UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INEGIERÍA

"DISEÑO DEL SISTEMA DE CONTROL DEL INSTRUMENTO TRACK-SIM PARA LA REPRODUCCIÓN DE TRAZAS LUMÍNICAS SOBRE LOS BLOQUES QUE CONFORMAN LA SUPERFICIE FOCAL DEL EXPERIMENTO JEM-EUSO"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – INSTRUMENTACIÓN

P R E S E N T A:

ISAI FAJARDO TAPIA

TUTOR: DR. GUSTAVO MEDINA TANCO

UNAM POSGRADO Ingeniería

CIUDAD UNIVERSITARIA NOVIEMBRE 2011

JURADO ASIGNADO:

Presidente:	Dr. Naser Qureshi
Secretario:	Dr. Gabriel Eduardo Sandoval Romero
Vocal:	Dr. Gustavo Adolfo Medina Tanco
1er suplente:	Dr. Oleg Viktorovich Kolokoltsev Filatov
2$^{do.}$ suplente:	Dra. Celia Angelina Sánchez Pérez

Lugares donde se realizó la tesis: Laboratorio de detectores, departamento de Física de Altas Energías, Instituto de Ciencias Nucleares-UNAM. Centro de Ciencias Aplicadas y Desarrollo Tecnológico-UNAM.

TUTOR DE TESIS:

Dr. Gustavo Adolfo Medina Tanco

_____

FIRMA

Este trabajo fue posible gracias al apoyo de beca para estudios de maestría por parte del Consejo Nacional de Ciencia y Tecnología (Conacyt), así como la red FAE del mismo Conacyt.

AGRADECIMIENTOS

A mis padres, por las enseñanzas, valores, inmenso amor y espíritu imperecedero con que forjaron mi camino.

A mi hermano por su ayuda, apoyo y revisiones. Que continúen los momentos de buena música.

A mi compañero y amigo en este ya largo camino: César.

A los amigos con quienes conformé un magnífico grupo de trabajo: Angélica, Avril, Cinzia y Héctor.

Al M. C. Miguel Enrique Patiño Salazar por la valiosa ayuda, apoyo y contribuciones.

A los que participaron con días y noches de esfuerzo: Aarón, Fidel, Héctor, Javier, Raúl y Verónica.

A quienes por diversos medios participaron de este trabajo: Alejandro, Arturo, Jesús y William.

Al posgrado de Instrumentación por los valiosos conocimientos.

A mi casa, la siempre gloriosa Universidad Nacional Autónoma de México.

Por mi raza hablará el espíritu.

Isai, Noviembre 2011.

El individuo ha luchado siempre para no ser absorbido por la tribu. Si lo intentas a menudo estarás solo, y a veces asustado. Pero ningún precio es demasiado alto por el privilegio de ser uno mismo.

Friedrich Nitzche.

CONTENIDO

ÍNDICE DE FIGURAS Y TABLAS

Figura/Tabla	Página

Figura/Tabla	Página

Figura/Tabla	Página

Figura/Tabla	Página

Introducción

La radiación ionizante y las fuentes de producción de ésta despertaron gran interés en la comunidad científica hacia finales del siglo XIX y principios del XX, de manera especial, el estudio de materiales radiactivos en el periodo de 1898 a 1912, fue de gran interés debido a que este campo ofrecía un camino directo al entendimiento de la naturaleza del átomo, cuya estructura era hasta entonces desconocida. Instrumentos como los electrómetros fueron comúnmente utilizados para medir el flujo pequeño de partículas provenientes de materiales radioactivos.[1] Sin embargo, las lecturas de los electrómetros debían ser corregidas debido a fugas, las cuales eran dependientes del tamaño del electrómetro y directamente proporcionales con el tiempo, pero notablemente no eran dependientes de la cantidad de carga dentro de las hojas del electrómetro. Estas fugas dejaron a la especulación sobre posibles fuentes de contaminación radioactiva que no habían sido descubiertas, o un flujo nuevo de partículas "etéreas".[2]

En términos simples, se puede asumir a los rayos cósmicos como partículas cargadas de alta energía, las cuales tienen su origen en el espacio exterior viajando casi a la velocidad de la luz y que golpean la Tierra desde todas direcciones. La mayoría de los rayos cósmicos se constituyen de núcleos de átomos completamente ionizados, que van desde los elementos más ligeros hasta los más pesados dentro de la tabla periódica, así como electrones y positrones de alta energía, y otras partículas subatómicas. Cuando se habla del término "rayos cósmicos", se pueden tener diversas "clasificaciones", sin embargo, en lo que todas coinciden o por lo menos en su mayoría, es sobre su

[1] Un electrómetro es un instrumento para medir la presencia de carga, que consiste de dos pequeñas hojas de metal suspendidas en un bulbo al vacío.

[2] Victor Hess estudió este fenómeno al colocar espectrómetros dentro de lagos (en donde las fugas desaparecieron). Finalmente, en 1912, Hess resolvió el problema al elevar dos cámaras ionizantes en globos a altitudes de 6 km. Mostró la existencia de un flujo de partículas provenientes del cielo con una intensidad que se incrementaba con la altitud. Gracias a este trabajo, Hess obtuvo el premio nobel de física en 1936 [1]. Su trabajo fue inmediatamente proseguido por estudios más detallados por parte de Kolhörster, quien mostró que el flujo de partículas se incrementaba rápidamente con la altitud, con un incremento en un factor de 10 a tan solo 10 km. Los rayos cósmicos se convirtieron entonces en la fuente de especulaciones por los siguientes 20 años, debido a su incremento exponencial con la altitud. Finalmente, Pfotzer mostró en 1936 que el flujo no se continuaba incrementando y alcanzaba un pico a alrededor de 15 km, después del cual decrecía rápidamente [2].

designación como “rayos cósmicos galácticos”, los cuales se originan en fuentes que se encuentran fuera del sistema solar, distribuidas a lo largo de la Vía Láctea. Sin embargo, algunas veces se incluye algunas otras clases de partículas energéticas, las cuales incluyen núcleos y electrones acelerados asociados con eventos energéticos dentro del Sol (denominadas partículas energéticas solares), así como partículas aceleradas en el medio interplanetario.

Debido a la gran especulación sobre la naturaleza de los rayos cósmicos (nombre introducido por la prensa alrededor de 1914), no hay ninguna definición científica sobre ésta frase, sólo la descripción popular: “cosas que llueven desde el cielo sin mojar”. La literatura científica ha adoptado tres variaciones de la frase: rayos cósmicos primarios, que hace referencia al flujo de partículas primarias externas a la atmósfera terrestre; cascadas de rayos cósmicos, que se designa como el flujo intermedio a lo largo de la atmósfera; y rayos cósmicos a nivel del mar, que se refiere al flujo final de partículas sobre la superficie terrestre.

En lo que se refiere a los rayos cósmicos de ultra alta energía, éstos colisionan con átomos en la alta atmósfera, produciendo una cascada de partículas secundarias, las cuales viajan como una lluvia a lo largo de la atmósfera hasta alcanzar la superficie terrestre. Éstos rayos cósmicos secundarios incluyen piones (los cuales rápidamente decaen para producir muones, neutrinos y rayos gama), así como electrones y positrones producidos por el decaimiento de muones e interacciones de rayos gama con los átomos de la atmósfera. El número de partículas que alcanzan la superficie de la Tierra se encuentra relacionado con la energía de los rayos cósmicos que golpean la atmósfera superior. Rayos cósmicos con energías de más de 10^{14} eV, se estudian con grandes arreglos de detectores distribuidos sobre varios kilómetros cuadrados a fin de muestrear las partículas producidas. Las frecuencias de estas lluvias varían desde 100/m^2/año para energías mayores a 10^{15} eV, a 1/km^2/siglo, para energías mayores a 10^{20} eV [2]. Por otra parte, los productos de interacciones de rayos cósmicos tales como neutrinos, son de interés y se estudian mediante grandes detectores localizados en profundas minas subterráneas y bajo el océano [2].

La mayoría de los rayos cósmicos que alcanzan la superficie terrestre son muones, con una intensidad promedio de cerca de 100/m^2/segundo. Aunque miles de rayos cósmicos pasan a través de nuestros cuerpos a cada minuto, los niveles de radiación resultante son relativamente bajos,

correspondiendo, a nivel del mar, a un pequeño porcentaje de la radiación natural de fondo. Sin embargo, la alta intensidad de los rayos cósmicos fuera de la atmósfera terrestre, constituye un peligro para astronautas, especialmente cuando el Sol se encuentra en el máximo de actividad, y el medio interplanetario se ve repentinamente lleno de partículas energéticas. Por otra parte, los rayos cósmicos también son dañinos para la instrumentación electrónica en el espacio, ya que impactos de iones pesados pueden causar fallas en los dispositivos semiconductores (Por ejemplo: cambios en bits de computadoras de a bordo y memorias), así como fallas en circuitos microelectrónicos.

Aunque la naturaleza de los rayos cósmicos es sin duda objeto de estudio de ciencia básica, y una de las preguntas sin respuesta dentro de la astrofísica moderna, tiene, sin embargo, repercusiones dentro de otras áreas del conocimiento en donde se incluyen aplicaciones y desarrollos tecnológicos. Por ejemplo, como he mencionado, todo el estudio realizado y vinculado con la detección de rayos cósmicos, se encuentra directamente relacionado a la predicción de fallas en circuitos integrados a nivel terrestres pero de mayor y especial interés en aeronaves y en sistemas electrónicos de aplicaciones espaciales. En particular, los rayos cósmicos que causan este tipo de daños son "hadrones", los cuales interactúan con la fuerza fuerte (también denominada "fuerza nuclear fuerte"), específicamente neutrones, protones y piones. Nunca se han medido valores experimentales del flujo de éstas partículas a nivel terrestre. Sin embargo, mediciones dispersas tomadas a lo largo de 50 años deben ser combinadas con estimaciones teóricas de variabilidad, a fin de obtener un punto de referencia del flujo de rayos cósmicos a nivel del mar. Por ejemplo, detectores de partículas en satélites han determinado que la composición primaria de los rayos cósmicos de baja energía consiste de 90% de protones y 10% de partículas alfa (núcleos de helio) [3], siendo el resto núcleos de iones pesados. Así mismo, no hay presencia de neutrones aislados en este flujo galáctico externo, dado que los neutrones son inestables fuera del límite del núcleo y tienen una vida media fuera del núcleo de 11 minutos [2] [4].

En cuanto a cuestiones tecnológicas se refiere, hay varios tipos de daños directos que las partículas energéticas pueden causar tanto a los equipos de vehículos espaciales así como a los de aviones.

Las partículas energéticas pueden ionizar átomos y desplazarlos dentro de sus estructuras cristalinas. Por ejemplo, los paneles solares de los vehículos espaciales sufren un fenómeno de

degradación en su rendimiento, debido al efecto acumulativo de daños derivados de estos desplazamientos nocivos provocados por las partículas energéticas. Un evento solar intenso de partículas energéticas puede en unos días, provocar la degradación equivalente a la del funcionamiento normal durante un año entero bajo el efecto únicamente de rayos cósmicos galácticos (GCR ó *galactic cosmic ray*). La ionización es generalmente el mecanismo dominante en la degradación del rendimiento de tarjetas y circuitos electrónicos. Las propiedades del teflón y de algunas pinturas utilizadas para regulación térmica pueden verse afectadas cuando se someten a altos grados de radiación [5]. Todo esto acorta el tiempo de vida de equipos y dispositivos.

El *efecto de un evento singular* (SEE) provoca daños por la ionización que una partícula produce cuando atraviesa un dispositivo microelectrónico. La creación de pares electrón-hueco debido al impacto de una partícula energética, puede alterar la respuesta normal de un circuito electrónico. Los SEU (*single event upsets*) son provocados especialmente por iones pesados procedentes de los rayos cósmicos primarios y/o secundarios, provocados por un protón muy energético en la atmósfera. Estos fallos pueden generar instrucciones erróneas en las computadoras de a bordo, o también *latch-ups*, que son estados anómalos de dispositivos electrónicos en los que se deja de responder a las señales de entrada. Los peores casos de SEE son los *burns out* (quemaduras), que son daños permanentes e irreversibles provocados por corrientes eléctricas parasitarias, las cuales son debidas a cortos circuitos provocados por la alteración de las estructuras cristalinas en dispositivos electrónicos (semiconductores principalmente). Algo muy importante a resaltar dentro de las fallas en dispositivos electrónicos, es referente a que los sistemas actuales son cada vez más vulnerables debido a la miniaturización, ya que es necesario tener menos carga para poder provocar un SEE [5].

Los electrones energéticos también producen diversos daños cuando penetran en vehículos y sistemas espaciales, debido a efectos de ionización y transferencia de su energía. En determinadas circunstancias esto puede provocar una descarga que interfiera con la instrumentación y los detectores, desviando la lectura de los mismos y desgastando sus materiales. La profundidad de penetración y la región donde puedan ocurrir los problemas dependerá de la energía de la partícula [5].

Ahora, para prevenir y disminuir estos problemas, se deben realizar estudios y análisis que eventualmente lleven a diseños de ingeniería de los materiales y dispositivos, con la finalidad de construirse de tal forma que puedan soportar el daño que las distintas partículas puedan causar. El grado de protección dependerá de la evolución en el largo plazo de la intensidad del flujo de partículas y el número de eventos solares, es decir, en la fase del ciclo solar en la que nos encontremos. Si dicha fase pudiera determinarse con precisión, sería de ayuda a la hora de especificar los requisitos de determinadas aplicaciones espaciales. Pero como la meteorología espacial, y especialmente el flujo de partículas es variable, hay también periodos en los que el funcionamiento debería ser interrumpido. Por ejemplo, el lanzamiento de vehículos espaciales durante periodos con intensos eventos de partículas solares, en especial si la órbita atraviesa zonas polares [5].

En cuanto al vuelo de aviones sobre rutas polares, también se ve afectado por las partículas energéticas y los efectos secundarios que éstas crean, alterando la atmósfera de la Tierra (especialmente la atmósfera polar, ya que se encuentra menos protegida por el campo magnético de la Tierra). Los numerosos protones de relativa baja energía que penetran hasta alturas de 50 a 70 km sobre el suelo, ionizan la atmósfera polar (ionosfera). A este fenómeno causado por las partículas solares energéticas se le denomina absorción de la cúpula polar, o PCA (*Polar Cup Absorption*), ya que este aumento de la ionización aumenta el fenómeno de absorción de ondas electromagnéticas de baja frecuencia, como las utilizadas en las comunicaciones de la aviación civil. De esto se tiene evidencia, pues durante eventos de partículas solares energéticas (por ejemplo en enero de 2005) se desviaron vuelos cercanos a los polos para que pudieran recuperar las comunicaciones con los centros de control [5].

Por otra parte, la magnetosfera de la Tierra no es siempre un escudo y puede llegar a ser peligrosa. El viento solar introduce continuamente energía en el sistema, que se libera en eventos explosivos llamados “subtormentas magnetosféricas”. El intenso flujo de electrones generado durante estos eventos en la magnetosfera provoca cargas en los satélites de comunicaciones. Estos eventos ocurren en condiciones solares tranquilas, cuando el viento solar de alta velocidad es particularmente eficiente transportando energía hacia la magnetosfera de la Tierra.

Debido a este tipo de problemas, así como a interés en el análisis del complejo entorno espacial y su impacto en los sistemas de aplicaciones espaciales, se desarrollaron modelos empíricos o cuasi-empíricos por parte de distintas organizaciones, generalmente independientes entre sí. En lo que respecta a los rayos cósmicos, el más conocido y utilizado es el modelo CREME (*Cosmic Ray Effects on Micro-Electronics*, efecto de los rayos cósmicos en la microelectrónica), que fue desarrollado por la NASA, y al que se puede acceder también a través del *Sistema de Información del Entorno Espacial* (SPENVIS) desarrollado por la ESA. Ambos disponen de interfaces de usuario a través de internet [5].

Así mismo, existe un modelo capaz de predecir el flujo de partículas de rayos cósmicos para todos los elementos de la tabla periódica, desde el hidrógeno hasta el uranio, y para energías comprendidas entre 1 y 10 000 MeV/nucleón. El espectro del flujo de energía se convierte a un espectro *lineal de transferencia de energía* (LET), que es el parámetro crucial para conocer el nivel de peligros espaciales para sistemas electrónicos. Éste representa un importante paso para calcular el número de SEUs [5].

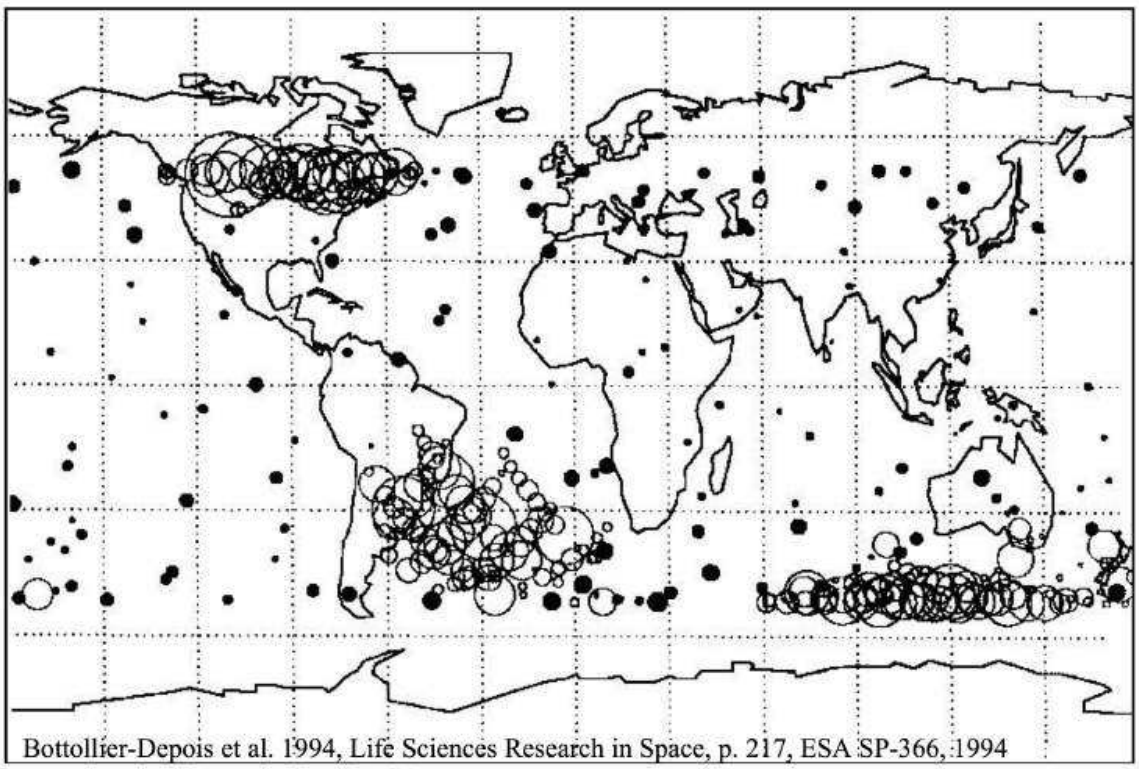
Por otra parte, las partículas energéticas son un riego potencial para la salud, ya que pueden llegar a dañar a las células. Cuando una partícula energética impacta contra una célula, deposita parte de su energía al interactuar con los electrones y las moléculas que forman la célula. Las consecuencias de esta interacción dependen de la especie y de la energía de la partícula (protón, ion, electrón, neutrón, fotón). Cualquier daño causado a las moléculas de la célula, especialmente al ADN, puede tener consecuencias para el futuro de la misma, su capacidad de división y el mantenimiento de su estructura. A su vez, en funcionamiento incorrecto de esta célula puede afectar al tejido u órgano al que pertenece.[3]

[3] Una célula dañada puede repararse a sí misma. Si no tiene éxito en esta labor, morirá. Si muchas células mueren entonces el órgano dejará de funcionar correctamente. Ahora, si la reparación no se hace totalmente bien, la célula puede reproducirse unas cuantas veces más, pero al hacerlo puede transferir los daños a las células hijas. De nuevo, el funcionamiento incorrecto de muchas de las hijas puede causar daños irreversibles al órgano. Un importante aspecto a resaltar es que las células dañadas que hayan sobrevivido pueden a su vez ser precursoras de células cancerígenas. Todos estos daños en tejido viviente dependen de la intensidad y tiempo de exposición a radiación, por tanto, la radiación cósmica constituye dos tipos de peligros para los seres vivos: dosis altas y dosis bajas.

Las dosis altas de radiación son una amenaza inmediata para la salud o incluso para la vida. Esto constituye un peligro para la tripulación de vuelos espaciales fuera de la magnetosfera terrestre. Los eventos de partículas energéticas solares están reconocidos como un riesgo crucial para vuelos interplanetarios (a la Luna o Marte). Por ejemplo, el gran evento solar del cuatro de agosto de 1972 sucedió durante el periodo de vuelos del Apolo a la Luna. Podrían haber tenido consecuencias mortales de haber sucedido justo en el momento del vuelo. Por lo tanto, la seguridad de los astronautas es un factor decisivo en los vuelos que se programen en el futuro [5].

En lo que respecta a las dosis bajas de radiación, éstas no acarrean consecuencias inmediatas pero son un riesgo a largo plazo. La tripulación de misiones espaciales e incluso la de aviones que viajan repetidamente a través de regiones de la Tierra, cuya atmósfera se encuentra bajo exposición intensa, como es el caso de los polos, se encuentran en esta situación.

Se han realizado estudios para determinar cuáles son las zonas que representan mayor riesgo para situaciones en que se tiene exposición a diferentes dosis de radiación. Como ejemplo, se muestra el mapa correspondiente a las dosis de radiación que fueron medidas a bordo de la estación espacial Rusa *MIR*, a través del experimento *Nausicaa* de la Agencia Espacial Francesa (CNES), durante el aumento de radiación provocado por un evento de partículas solares en octubre de 1989. El diámetro de los círculos indica la dosis de radiación recibida. La órbita de la *MIR*, la cual se situaba a una altitud de 420 km y con una inclinación de 51° con respecto al ecuador de la Tierra, llevaba a la estación a través de las regiones polares de Canadá, del Océano Pacífico y del sur de Australia [5]. Las dosis de radiación recibidas allí, tal y como muestra el diámetro de los círculos, están muy por encima de las de otras latitudes, con la excepción de la región débil del campo magnético del sur del Océano Atlántico (conocida como la anomalía del Atlántico Sur). Por otro lado, existen también zonas de aumento de radiación que no son debidas a eventos solares sino a la circulación de partículas cargadas en el campo magnético de la Tierra.



La dosis máxima de radiación registrada por la MIR/Nausicaa, mostradas en el mapa de círculos es de 2 mSv/h. Imagen tomada de [5].

Ahora, los efectos de la exposición a la radiación sobre la salud dependen de la cantidad de energía que absorban los tejidos (cuando más fuerte sea el flujo de las partículas mayor será la energía depositada), pero también depende del tipo de la partícula, su energía, y el órgano específico. Por ejemplo, los rayos X depositan energía de una forma relativamente uniforme en un volumen, mientras que la energía de los neutrones se deposita de forma más localizada, dependiendo de la reacción nuclear que se produzca en el tejido. Los neutrones tienen una mayor capacidad de crear lesiones que los protones de alta energía, electrones o rayos gamma.

Debido a estos efectos sobre la salud, las dosis de radiación recibidas por el personal a bordo de estaciones espaciales y aeronaves debe ser vigilada, ya que los tiempos de exposición a niveles bajos de radiación son acumulativos. La unidad del Sistema Internacional de Unidades (SI) utilizada para medir los efectos de la exposición prolongada a niveles bajos de radiación por la materia viva es el Sievert ó Sv (Rolf Sievert, Físico Sueco 1896 – 1966) el cual es equivalente a 1 J/Kg [5]. Representa la suma de las dosis de radiación ionizante absorbidas por diferentes órganos del

cuerpo humano, ponderados por*: la especie de la partícula (mayor peso para partículas alfa y núcleos pesados, neutrones, protones y finalmente fotones y electrones); y la sensibilidad del órgano expuesto a las radiaciones ionizantes*.

Algunos ejemplos de diferentes dosis de radiación se muestran a continuación.

Radiación de origen terrestre: La dosis normal debida a la radiactividad ambiente en la Tierra tiene una media de 2.4 mSv por año, con diferencias apreciables entre países. A nivel del mar la contribución de los rayos cósmicos es de aproximadamente 0.3 mSv [5] [6].

Radiografía médica: Una dosis de radiación recibidas durante una radiografía médica va desde 0.1 a varias decenas de mSv, dependiendo del tipo de radiografía [5].

Vuelos comerciales largos (a gran altura): La dosis típica recibida durante un vuelo transatlántico (Europa – América del Norte) debida a rayos cósmicos galácticos (GRC), es de 0.05 mSv. Puede aumentarse significativamente en el caso de eventos de partículas energéticas (se han contabilizado aumentos de hasta un factor de 10 en el caso de eventos solares muy intensos, pero estos eventos son muy poco frecuentes y tienen una duración muy corta como para influir en la dosis anual). Por ejemplo, a lo largo de los años, viajeros frecuentes o tripulaciones de cabina de vuelo pueden llegar a acumular dosis de unos pocos mSv.[4]

Vuelos interplanetarios: Un vuelo espacial hasta Marte implica una dosis de radiación de 1 Sv, que se atribuye principalmente a los rayos cósmicos, por lo que esta dosis no incluye la correspondiente

[4] Las compañías aéreas tienen ahora el requisito legal de comprobar que sus tripulaciones no reciben, al igual que cualquier otro trabajador, una dosis mayor de 100mSv cada 5 años, ni una dosis mayor a 50mSv anual. Así mismo, las trabajadoras embarazadas no deben acumular más de 1 mSv hasta el final de su embarazo, ya que el feto está más expuesto y es más vulnerable [5]. Es importante señalar que, de los casos mostrados aquí, Sievert consideraba el riesgo debido a exposiciones de bajo nivel de radiación como efectos estocásticos, por lo que valores por encima de 1Sv no son considerados.

a eventos solares, que pueden ser mucho mayores y constituir, en el momento, una seria amenaza para la vida si no se dispone de los escudos adecuados [5].

Por todo lo anterior, resulta de gran relevancia estudiar, conocer y entender la naturaleza de los rayos cósmicos, por lo que es de gran interés su detección en experimentos que se llevan a cabo actualmente, en especial, en aquellos que se encuentran en desarrollo para una operación futura. De éstos, uno de gran relevancia dentro del campo de la ciencia a nivel internacional es el experimento JEM-EUSO. Este experimento será el primero que intentará la medición de rayos cósmicos ultra-energéticos desde un observatorio en órbita con el fin de estudiar la naturaleza física detrás de estos fenómenos de tan altas energías, mediante un telescopio conectado a una bahía del Módulo Experimental Japonés en la Estación Espacial Internacional (ISS), el cual observará el lado nocturno de la atmósfera terrestre.

Con el objetivo de atacar estos problemas en la fase de prototipo de la superficie focal, en la Universidad Nacional Autónoma de México, se diseña un dispositivo denominado Track-Sim. El objetivo es la simulación física de los perfiles correspondientes a las trazas reales de partículas que se presentan a lo largo de la superficie focal del instrumento. Para esto, se utilizarán perfiles de intensidad simulados numéricamente correspondientes a diferentes partículas incidentes, con diferentes energías y geometrías y se los reproducirá sobre la superficie focal con fotones en el rango de 300 a 400 nm, que corresponden a las 3 líneas espectrales de fluorescencia del nitrógeno (N_2). Dicha reproducción de perfiles contará con todas las características espaciales y temporales que contendría una traza real, para lo cual, el presente trabajo, hace referencia al diseño del sistema de control del instrumento Track-Sim y la validación del mismo mediante el prototipo Track-Flat. El Track-Sim consistirá de un dispositivo compacto y robusto de forma que pueda ser utilizado por un usuario asistido por una computadora.

Este sistema se basa en la proyección de luz UV que utiliza como fuente un arreglo de LED's, los cuales, junto con un sistema óptico, permitirán la reproducción de las diversas trazas sobre los módulos foto-detectores (PDM), que son los módulos mediante los cuales se ensambla la superficie focal del telescopio. Se aborda el trabajo de hardware que controla la emisión de fotones por medio de LEDs, con miras al sistema final mediante la traducción de EAS (*Extensive Air Shower*), en

generación de pulsos eléctricos con las características requeridas por la óptica que forma la traza sobre la superficie focal, además de la generación de *background* que simule las condiciones de radiación de fondo presentes en el momento de funcionamiento del telescopio.

Es necesario resaltar que la investigación en las diferentes áreas de la ciencia no se puede concebir sin disponer de los instrumentos apropiados, por lo que la importancia del área instrumental representa una pieza clave, pues gracias a técnicas y desarrollos en instrumentación, es como se hace posible la creación de experimentos de diversas índoles. Así mismo, el proceso de construir dichos instrumentos ha demostrado ser un potente incentivo para el desarrollo y la innovación tecnológica, así como una herramienta efectiva para la transferencia de tecnología a la sociedad, por lo que resulta atractiva para las empresas y da una fuente de valor añadido para el país.

Sin embargo, la creación de instrumentación ha de ser ambiciosa y debe ser impulsada por la ciencia que necesita de instrumentación de punta para avanzar, para lo cual, deben plantearse incluso posibles cambios de paradigmas, a fin de formar el contexto en que las directrices de los futuros desarrollos se realicen.

Finalmente, dadas las características particulares sobre desarrollos tecnológicos e instrumentales, y considerando la situación actual del país, el desarrollo del área de instrumentación espacial es de una importancia relevante, pues permitirá en el futuro, la participación en misiones y proyectos de gran alcance, en donde sin duda, la formación de los recursos humanos en el área, cobrará vital importancia, ya que serán los futuros científicos e ingenieros instrumentistas, quienes podrán asumir la responsabilidad de liderar el desarrollo de instrumentación avanzada.

Capítulo 1

Detección de rayos cósmicos

La vida que aspira a lo alto se hace cada vez más dura; a medida que aumenta el frío, también crece la responsabilidad. Una cultura elevada es como una pirámide: sólo puede alcanzarse sobre una base muy ancha; la condición previa y fundamental es una mediocridad sana y fuertemente consolidada.

F. Nietzsche, "El Anticristo".

1.1 CHUBASCOS ATMOSFÉRICOS

La experimentación astrofísica cubre una gran variedad de campos de interés, muchos de ellos bien conocidos por su amplia difusión desde experimentación científica hasta instrumentos astronómicos de aficionados, los cuales se encuentran basados principalmente en las regiones ópticas y de radio, dispuesto así principalmente por el desarrollo histórico que han tenido éstas áreas y sus respectivos instrumentos para observación y detección. Sin embargo, existen otros equipos así como técnicas de observación y detección para fenómenos físicos de diferente índole como: rayos cósmicos, rayos *x*, rayos γ, observaciones en ultravioleta, ondas milimétricas, detección de neutrinos cósmicos, y hasta la búsqueda de ondas gravitacionales y materia oscura.

En lo que a los rayos cósmicos se refiere, existen diferentes tipos de éstos, los cuales se diferencian por su energía, desde los menos energéticos (provienen principalmente del sol), hasta los más energéticos que pueden tener su origen en el centro de la galaxia, como las explosiones de supernova y otras fuentes astrofísicas. Éstos rayos arriban constantemente a la tierra, siendo los más energéticos los únicos en ser capaces de alcanzar la superficie terrestre, sin embargo el arribo de éstos es muy bajo (aproximadamente uno por siglo sobre una superficie de 1 kilómetro cuadrado).

Los rayos cósmicos ingresan a la atmosfera terrestre y forman cascadas o chubascos (*Extensive Eir Shower* ó EAS), y se puede considerar, de manera simplificada, que una lluvia de partículas está compuesta de tres componentes, un componente hadrónico, uno muónico y uno electromagnético. En la figura 1.1 se muestra un esquema simplificado de una cascada de partículas originada por un rayo cósmico de naturaleza hadrónica. Uno de los primeros objetivos que se fijan en la detección de rayos cósmicos, es el estudio de la radiación cósmica que nos llega a la Tierra a través de ellos, la cual varia en muchos órdenes de magnitud (de 10^7 a 10^{20} eV) y, por consiguiente, es necesario usar distintos métodos experimentales dependiendo del intervalo de energía que se espera estudiar (figura 1.2). Cuando se trata de energías inferiores a 10^{15} eV (1 PeV) es factible la detección directa del núcleo primario. Para ello, se requiere transportar en globos aerostáticos o a

bordo de satélites los instrumentos de detección. Dependiendo del objetivo de estudio, las técnicas que se emplean son semejantes a las que se aplican en los experimentos de física nuclear y de partículas elementales. Para este caso, se procede a provocar la interacción del rayo cósmico con un medio material conocido y se estudian las características de los productos resultantes. Estos métodos van desde el uso de emulsiones hasta los más sofisticados que reconstruyen las trayectorias de las partículas que se crean en la colisión, con lo cual se puede medir la dirección del rayo cósmico primario.

Sin embargo, la detección directa pone de manifiesto la dificultad de que el número de rayos cósmicos primarios es tanto menor cuanto mayor sea la energía de estos, por lo que a muy altas energías, el ritmo de llegada es tan lento que la técnica de detección directa resulta inviable. Por ejemplo, para detectar unos mil rayos cósmicos con energías superiores a 1 PeV sobre una superficie de 10 m^2 se necesitaría más de un año. Para superar esto, se han desarrollado paralelamente diversas técnicas de detección indirecta basadas en la observación EAS, empleando detectores sobre la superficie terrestre. Estas técnicas cubren energías desde 10^{11} a 10^{21} eV.[5] Los métodos basados en detectores sobre la superficie terrestre pueden ocupar físicamente áreas muy extensas y por tanto tienen una eficiencia mucho mayor que los detectores a bordo de globos o satélites. Sin embargo, tienen como desventaja que la caracterización de la partícula primaria a partir de los datos de la cascada atmosférica es más difícil.

[5] Si la energía del rayo cósmico primario es superior a 10^{14} eV, la componente electromagnética de la EAS puede ser detectada a nivel del suelo. A energías inferiores se puede detectar la luz producida por efecto Cherenkov por las partículas de alta energía de la EAS.

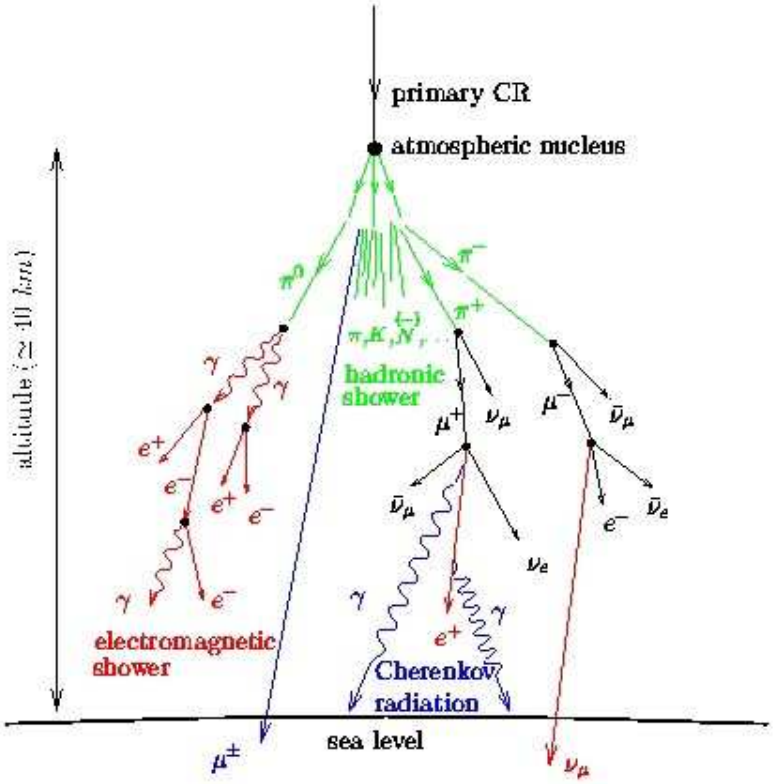


Figura 1.1 Representación de una lluvia de partículas o EAS. Imagen tomada de [7].

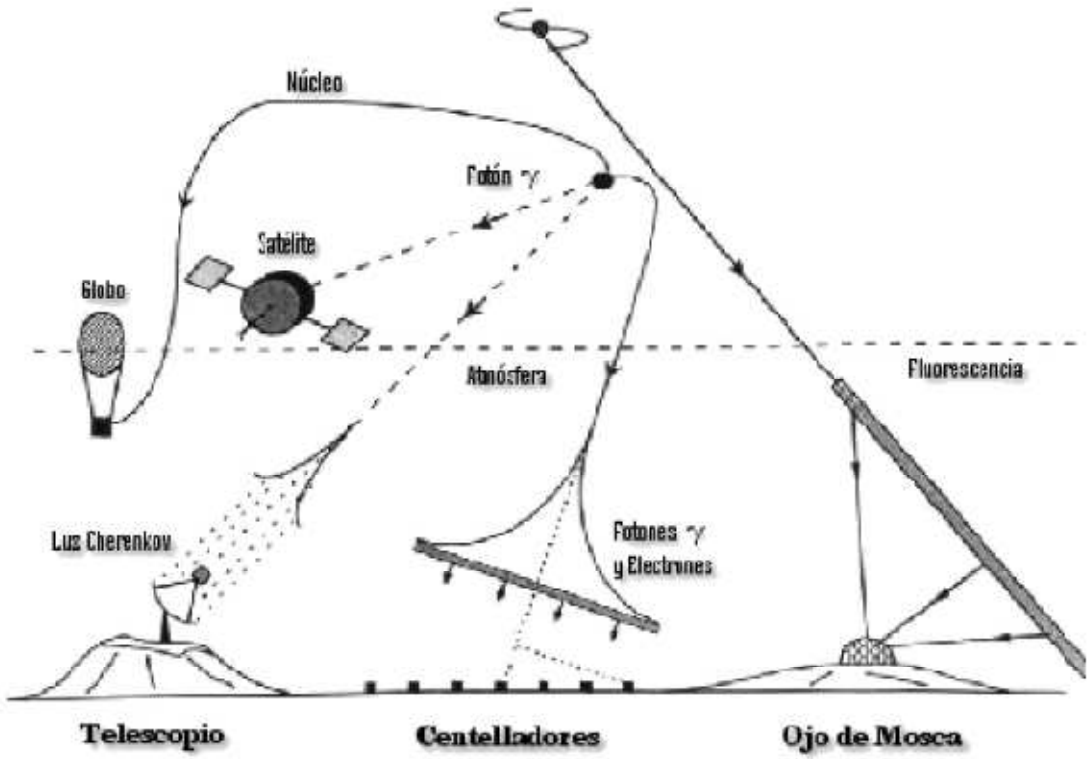


Figura 1.2. Métodos de detección de rayos cósmicos. Imagen tomada de [7].

Detectores de superficie en agua (Cherenkov)

Los detectores de agua consisten en un recipiente con paredes de polietileno reflectante, completamente cerrados y sellados para que no permitan el ingreso de luz, el cual almacena agua muy pura en su interior (figura 1.3). Cuando las partículas cargadas de las EAS pasan por el agua, producen una radiación que genera unos flashes de una tenue luz azul como una onda de choque[6]. La luz que produce la partícula resulta emitida dentro de los límites de una superficie de forma cónica, donde el vértice es el punto en que la partícula entró al detector y la directriz es la dirección de su movimiento. Esta luz rebota en las paredes reflectivas y, finalmente, es detectada por un detector de luz muy sensible denominado fotomultiplicador ó PMT (*Photomultiplier Tube*), el cual se halla en la parte superior del recipiente. Las señales obtenidas con el PMT son acondicionadas por una electrónica de adquisición de datos tipo *Front End* y posteriormente transmitidas a una estación central para su almacenamiento y procesamiento.

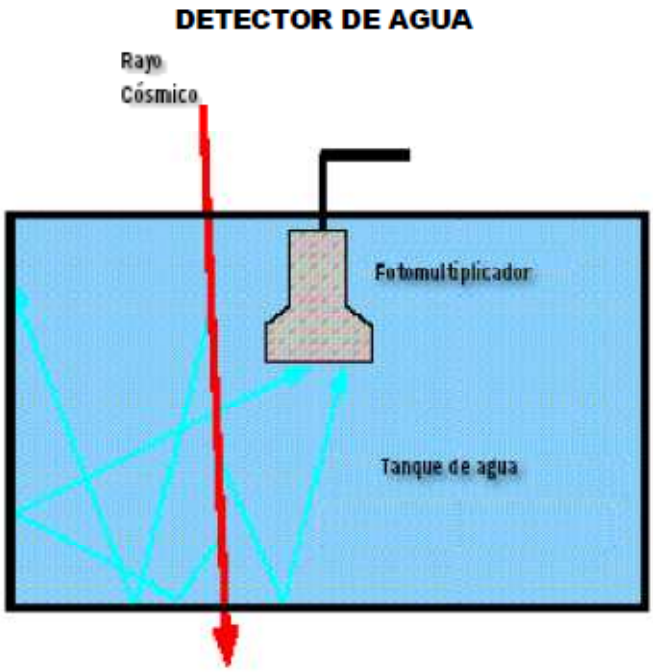


Figura 1.3 Detector Cherenkov de agua. Tomado de [7].

[6] El efecto Cherenkov se produce cuando una partícula cargada se mueve en un medio transparente con velocidad mayor que la que tendría la luz en dicho medio. La cantidad de luz Cherenkov producida es proporcional al número de partículas y al camino que estas recorren en el agua.

Detectores de centelleo

El detector de centelleo está constituido por un plástico especial denominado centellador. Al pasar las partículas cargadas de los rayos cósmicos por el centellador, excitan los átomos del plástico otorgándoles una cierta energía. Luego, los átomos excitados pierden esa energía emitiendo algunos fotones de luz. Enseguida, esa luz es detectada por un fotomultiplicador, el cual produce una señal eléctrica de corta duración (figura 1.4). Si se emplea un dispositivo lo suficientemente rápido se puede medir el instante de tiempo en que la partícula atravesó el plástico con una precisión del orden de 10^{-9} s (ns). En consecuencia, se trata de detectores capaces de medir la inclinación del disco de partículas que conforma la EAS, de donde se puede deducir la energía del rayo cósmico primario.[7]

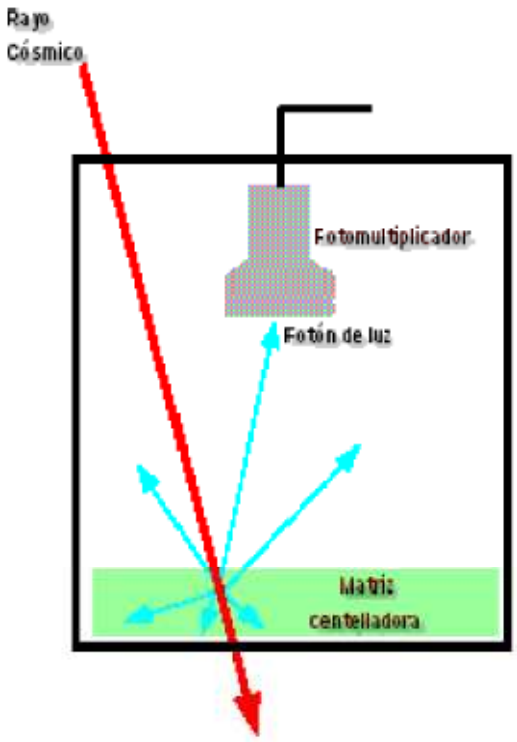
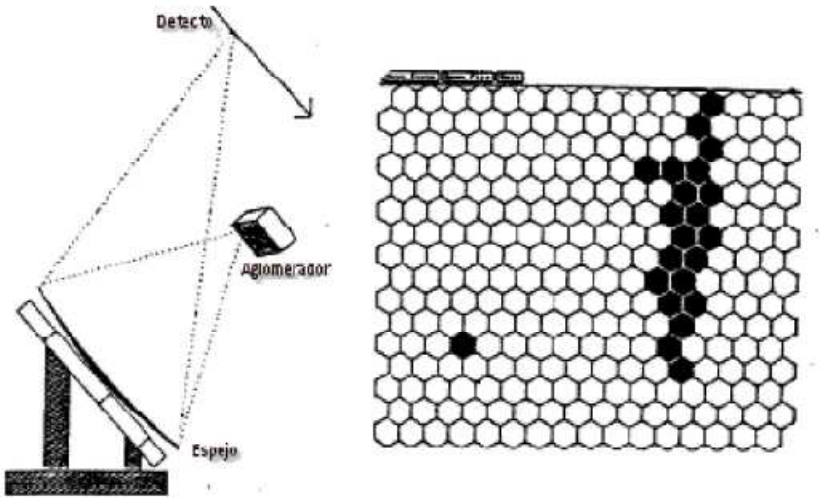


Figura 1.4 Detector de centelleo. Tomado de [7].

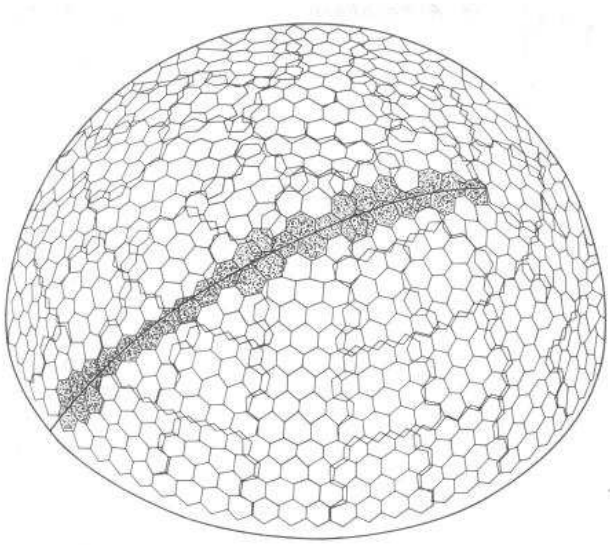
[7] Por otro lado, la comparación de los tiempos de arribo de las diferentes partículas a los distintos contadores permite medir la inclinación del disco, obteniendo, de este modo, la dirección de cada uno de los rayos cósmicos primarios que han sido detectados a su llegada.

Detectores de fluorescencia

El detector de fluorescencia (FD) consiste de una serie de colectores de luz (en la práctica PMTs), de pequeña apertura angular (~1.50°), dispuestos espacialmente sobre una estructura tipo panal de abeja con el objetivo de permitir una fina segmentación de una parte del cielo (figura 1.5). Cuando una lluvia de partículas se desarrolla en el cielo a cierta distancia del detector, ésta es observada como un punto de luz en movimiento con velocidad próxima a la de la luz. De acuerdo a este esquema, la lluvia en su desarrollo dispara una serie de PMTs de manera ordenada, tanto en forma espacial como temporal. Como las direcciones puntuales de cada uno de estos PMTs son conocidas, al igual que las respectivas señales colectadas (forma, amplitud y tiempo), utilizando esta información resulta posible reconstruir la geometría de la lluvia y a posteriori su perfil longitudinal, para finalmente evaluar la energía del rayo cósmico primario. Esta técnica fue introducida y desarrollada a mediados del siglo XX, como una nueva técnica para la detección de lluvias de partículas originadas por rayos cósmicos de alta energía, hasta establecerse como el experimento del detector *Fly's Eye* y su sucesor *HiRes*, desarrollados por la universidad de Utah (EE.UU.) [8].



a)



b)

Figura 1.5. a) Detector de fluorescencia. a) Esquema del detector y b) arreglo geométrico de correspondiente a pixeles de PMT en el cielo, así como la imagen de una lluvia en los pixeles oscuros. Imágenes tomadas de [7,8].

1.2 OBSERVATORIO PIERRE AUGER

El observatorio Pierre Auger fue diseñado para registrar las lluvias de partículas originadas por los rayos cósmicos con energía mayor a 10^{19} eV, cuando un rayo cósmico choca contra las moléculas de la atmósfera superior. Es un tipo de observatorio híbrido que consiste en una red de detectores de superficie Cherenkov de agua y un sistema de telescopios de fluorescencia atmosférica.

Como se resumió, la técnica de fluorescencia (FD ó *Fluorescence Detector*) se basa en la detección de la luz emitida por el nitrógeno atmosférico durante el desarrollo longitudinal de la lluvia, mientras que la técnica con detectores Cherenkov registra las partículas de la lluvia al nivel del suelo (SD ó *Surface Detector*), los cuales se encuentran posicionados entre sí a distancias regulares. La fluorescencia atmosférica es registrada utilizando telescopios ópticos localizados en puntos perimetrales del arreglo de superficie. El SD opera durante todo el tiempo, mientras que el FD es operativo bajo buenas condiciones climáticas (cielo despejado con poca cobertura de nubes) y baja contaminación lumínica (únicamente durante las noches con escasa luz de luna). Debido a esto, el tiempo de

funcionamiento del FD es de alrededor de 10 % del tiempo de funcionamiento del SD. Se espera entonces que al menos el 10 % de las lluvias detectadas sean híbridas, las cuales, resultan de gran utilidad pues son reconstruidas con mayor resolución y por lo tanto son útiles para efectuar la calibración cruzada en energía entre ambos detectores [9]. La figura 1.6 muestra una composición de los principales componentes del observatorio.

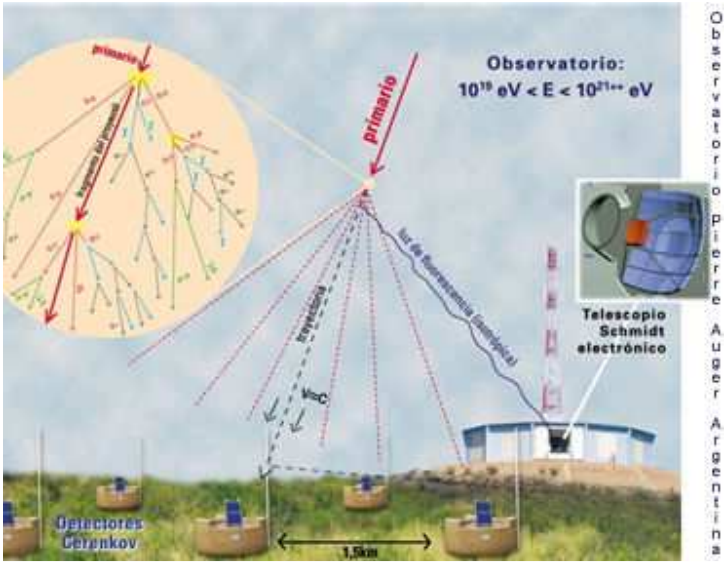


Figura 1.6. Principales componentes del observatorio Pierre Auger sitio sur. Los "tanques" marrones constituyen los detectores de superficie. El edificio "azul" es uno de los observatorios de fluorescencia instalados. Imagen tomada de [10].

El concepto de diseño original del *Observatorio Pierre Auger* contempla tener un sitio en cada hemisferio. El sitio sur se encuentra en la pampa amarilla, en las localidades de Malargüe y San Rafael (Provincia de Mendoza - Argentina), y en la actualidad su construcción se encuentra prácticamente concluida. El sitio norte será construido en el Estado de Colorado (EE.UU.) posterior a la construcción del sitio sur. La gran apertura del observatorio es lograda gracias al tamaño gigante del área de detección, (alrededor de 3000 km^2 en cada sitio), con lo que con ambos sitios se garantizará una cobertura completa del cielo. Las características más relevantes consideradas para definir la localización geográfica de los sitios son [10]:

- Latitud norte y sur comprendida entre 30° y 45° para tener una buena cobertura de todo el cielo.

- A aproximadamente 1400 m sobre el nivel del mar, para maximizar la eficiencia de detección de lluvias de partículas originadas por rayos cósmicos con energías mayores a 10^{19} eV.
- Suelo esencialmente plano y con escasa vegetación dentro de un área de ~3000 km^2.
- Clima seco con temperaturas moderadas, reducida cobertura del cielo, baja opacidad atmosférica por aerosoles, lejanía con fuentes artificiales de luz y polución humana, para permitir el funcionamiento del FD.

La construcción del *Observatorio Pierre Auger*, sitio sur, comenzó en el año 2001 con la fase de un arreglo de ingeniería. Durante los 6 meses de funcionamiento del *Arreglo de Ingeniería* unos 80 eventos híbridos fueron detectados. Actualmente, los 24 telescopios del FD se encuentran instalados y los cuatro ojos se encuentran en funcionamiento junto a un total de 1430 tanques de radiación Cherenkov del SD con un funcionamiento del detector híbrido que colecta a cada mes de observación un volumen de datos del orden de 10 GBytes [9]. La tabla 1.1 presenta las principales características del Observatorio Pierre Auger de acuerdo al status actual al 30 de Junio de 2010.

DATOS TÉCNICOS	
Detectores de Superficie (SD)	
Área cubierta:	3000 km^2.
Cantidad de tanques detectores:	1600.
Tipo de detectores:	Detectores Cherenkov en agua purificada de 120,000 litros, con tres tubos fotomultiplicadores cada uno.
Distancia entre detectores:	1.5 km.
Telescopios de fluorescencia	
Cantidad de telescopios:	4 por sitio con un total de 24 espejos.
Alcance:	20 km para cascadas de 10^{20} eV.
Espejos:	Esféricos de 3.6 m x 3.6 m con 30º × 30º de apertura y 440 tubos fotomultiplicadores cada uno.

Tabla 1.1. Principales características del sitio sur del observatorio Pierre Auger. Datos tomados de [10].

1.3 JEM-EUSO

El experimento JEM-EUSO (*Extreme Universe Space Observatory on Japanese Experiment Module*) pretende develar el origen astronómico de los rayos cósmicos más energéticos al observar desde la Estación Espacial Internacional (ISS) los efectos de éstos en la atmosfera terrestre, la cual representa el mayor detector que puede ser empleado. Este experimento, bajo diseño y construcción, será el primero en intentar la medición de rayos cósmicos ultra-energéticos desde un observatorio en órbita. Cuando una partícula de rayos cósmicos extremadamente-energéticos ó EECR (*Extremely-high Energy Cosmic Ray*) choca con un núcleo en la atmósfera terrestre y produce un EAS (que consiste en varios electrones, positrones y fotones), JEM-EUSO captura la luz fluorescente en el rango ultravioleta asociado al perfil de EAS, para estudiar su comportamiento y determinar la energía de la partícula primaria. El instrumento consiste en un telescopio de amplio campo de visión que detectará partículas con energía por encima de 10^{20} eV, al orbitar la Tierra cada 90 minutos aproximadamente, a una altitud de 400 km. La figura 1.7 muestra el principio del telescopio JEM-EUSO.

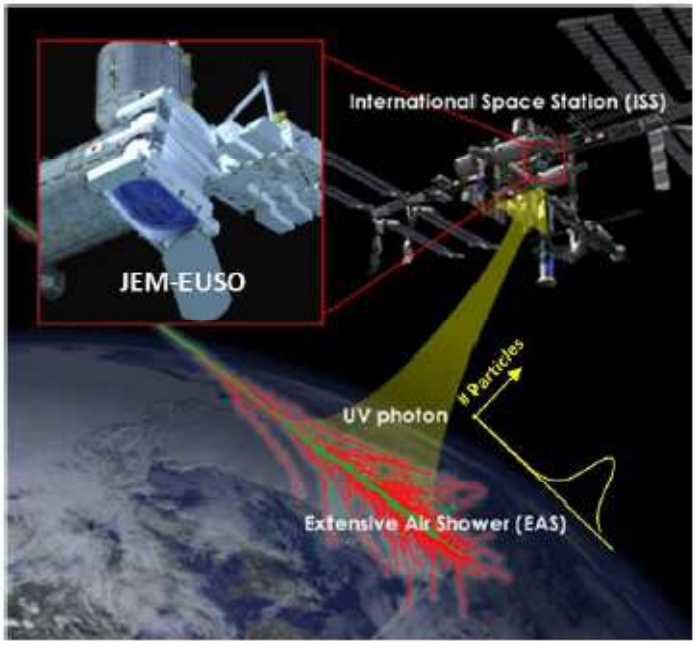
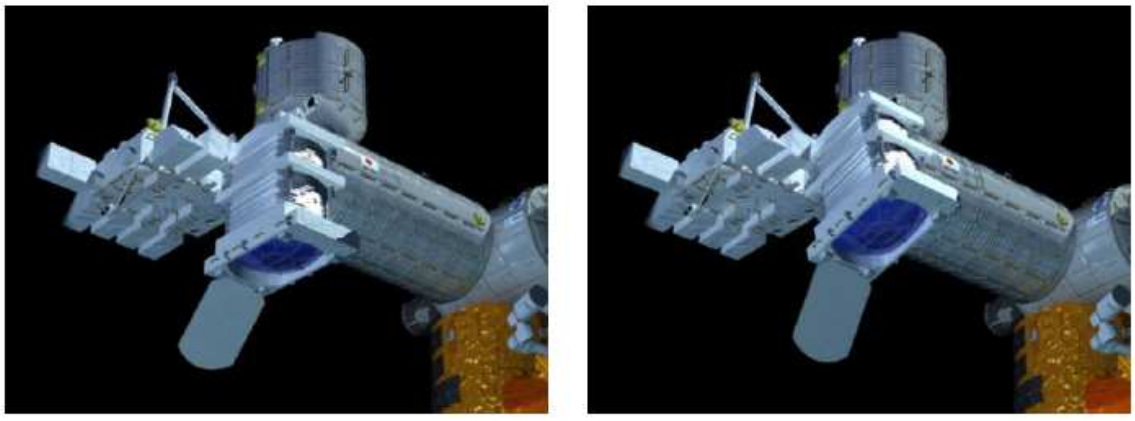


Figura 1.7. Principio del instrumento JEM-EUSO para la detección de partículas de alta energía. Imagen tomada de [11].

El telescopio JEM-EUSO tiene un campo de visión (FoV) de 60° (±30°) con una óptica compuesta por lentes de Fresnel, con la finalidad de detectar *tracks* de EAS con una resolución temporal de 2.5 μs y una resolución espacial de alrededor de 0.75 km (correspondiente a 0.1°) en modo *nadir*. Las imágenes segmentadas en tiempo permitirán determinar la energía y dirección de las partículas primarias, al utilizar cerca de 6000 fotomultiplicadores (PMTs) multiánodo en la superficie focal del detector, con miles de pixeles sensibles a la luz ultravioleta.

Es posible obtener un incremento en el área efectiva de detección al inclinar el telescopio a partir del modo nadir (figura 1.8) con lo que el umbral de energía se incrementa debido al incremento en la distancia de cada EAS y la absorción atmosférica. De estas características en el modo de operación del telescopio, se espera que los primero años de operación de la misión estén destinados a la observación en la región de bajas energías (modo nadir) para después dar paso a la región de altas energías (modo inclinado ó *tilted mode*).



a) *b)*

Figura 1.8. Ejemplificación de los modos de funcionamiento del telescopio JEM-EUSO en el Módulo Experimental Japonés (JEM/EF) en la ISS. a) Modo nadir y b) modo inclinado (tilted). Imagen tomada de [11].

El telescopio JEM-EUSO será capaz de reconstruir la dirección de arribo de los EECRs con precisión de unos cuantos grados, en un área terrestre con un radio de 250 km. Así mismo, la apertura instantánea de JEM-EUSO será mayor que la del Observatorio Pierre Auger Sur por un factor de entre 65 y 280, dependiendo del modo de operación *nadir* o *tilted* (figura 1.9). Se planea que la misión sea lanzada alrededor de 2015 – 2016, utilizando el cohete japonés H2B así como el vehículo de transporte HTV (*H-II transfer Vehicle*).

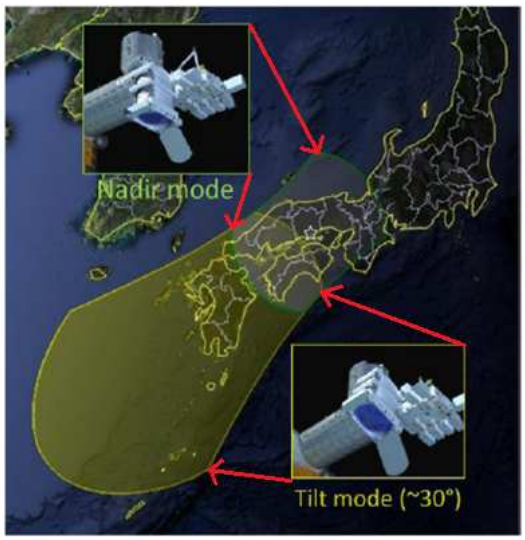


Figura 1.9. Área observada por el telescopio JEM-EUSO para una exposición para modos de observación nadir y tilt. Imagen tomada de [11].

1.3.1 Objetivos Científicos

Los rayos cósmicos constituyen la componente más energética de la radiación cósmica, la cual puede ser considerada como un "*canal de partículas*" que complementa a un "*canal electromagnético*", propio de la astronomía convencional. El principal objetivo de JEM-EUSO es iniciar un nuevo campo en astronomía y astrofísica que utilice el canal de partículas de energía extrema (10^{19} eV < E < 10^{21} eV) para lo cual, se planea que JEM-EUSO detecte más de mil eventos con energía mayor a 7×10^{19} eV en sus cinco años de operación.

Se pretende conocer el origen de EECRs, el ambiente de propagación desde la fuente hasta la Tierra, así como el mecanismo físico de aceleración de tales partículas. En los objetivos de la misión se encuentra el realizar observaciones como la detección de neutrinos y rayos gama de muy alta energía, la verificación de la teoría de la relatividad y efectos de gravedad cuántica a energías extremas, así como la supervisión sistemática de fenómenos atmosféricos como eventos luminosos transitorios ó TLE (*Transient Luminous Events*) en la atmósfera superior, resplandores nocturnos (*nightglows*), descargas de plasma y relámpagos.

1.3.2 Principio de observación

Como se ha mencionado, cuando un EECR choca con un núcleo atmosférico produce partículas secundarias, las cuales interactúan con los átomos de la atmósfera provocando una cascada de partículas. El número de partículas secundarias en una EAS se encuentra relacionado con la energía del EECR primario, por ejemplo, se pueden producir tanto como 10^{11} partículas en el máximo del desarrollo de una lluvia correspondiente a 10^{20} eV de un EECR [11]. Las partículas más dominantes dentro de una EAS son electrones que se mueven a través de la atmósfera, los cuales excitan a su paso los átomos y moléculas atmosféricos, en particular de nitrógeno (N_2), a diferentes niveles de energía "metaestables". Con un periodo de relajación corto, los electrones de esos niveles de energía regresan a su estado fundamental emitiendo una luz de fluorescencia característica. En el aire, los picos de esta luz caen en la banda ultravioleta (UV) con longitudes de onda entre 300 y 400 nm. La luz emitida es isotrópica y proporcional a la energía depositada en la atmósfera. Una EAS inducida por un EECR forma entonces una traza significativa de luz de fluorescencia a lo largo de su paso por la atmósfera, la cual depende de la energía y ángulo cenital del EECR primario. Por otra parte, también se presentan numerosas partículas secundarias con velocidades mayores a la de la luz en la atmósfera, las cuales emiten fotones Cherenkov. Parte de esos fotones serán difundidos isotrópicamente cuando toquen la superficie de la Tierra, mar o nubes [11].

Al mirar hacia la parte oscura de la atmósfera terrestre, el telescopio JEM-EUSO detectará la luz de fluorescencia como una traza con un perfil temporal y espacial de intensidad variable, tal y como se representa en la figura 1.10. Un EAS aparece como un pequeño disco luminoso cuando se lo mira de forma constante, el cual se mueve en línea recta a la velocidad de la luz. La luminosidad del disco varía de acuerdo a la intensidad y geometría entre un EAS y el telescopio. Mediante imágenes del movimiento de la traza durante algunos microsegundos, se puede determinar la dirección de arribo del EECR primario, mientras que la integral de la luz detectada proporciona información importante para determinar su energía. La forma de la cascada (en especial la posición del máximo de la lluvia a una profundidad con inclinación transversal) proporciona una indicación sobre la

naturaleza del EECR primario: por ejemplo, lluvias que se desarrollan profundo en la atmósfera indican que el EECR primario es un neutrino, lo cual se deduce debido a su sección transversal de muy baja interacción.

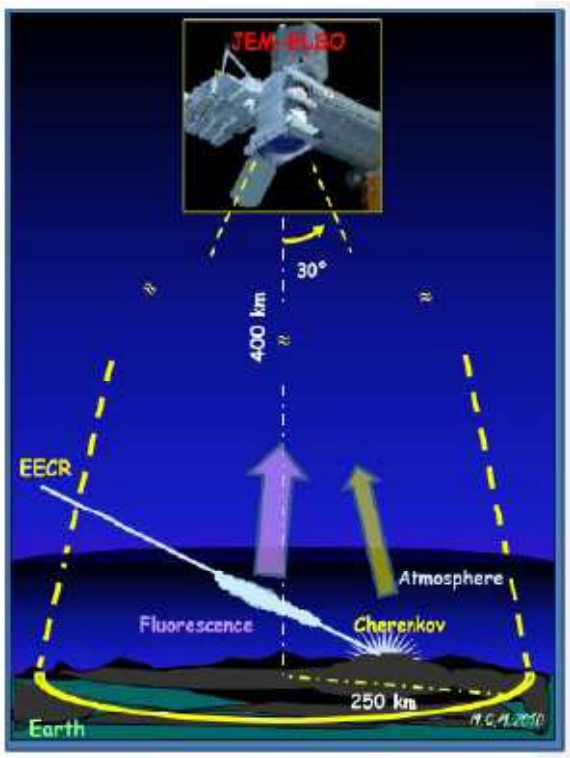


Figura 1.10. Principio de observación de JEM-EUSO. El telescopio detecta luz de fluorescencia y Cherenkov producidas por EAS a su paso por la atmósfera. Imagen tomada de [11].

1.3.3 El instrumento JEM-EUSO

El segmento de vuelo de la misión JEM-EUSO se conforma, básicamente, de un telescopio refractor. Este telescopio consiste en una cámara digital de alta velocidad de gran apertura y campo de visión amplio, que trabaja en longitudes de onda del ultravioleta cercano (330 – 400 nm) con capacidad de conteo de fotones individuales. Las principales partes que componen el detector son: el subsistema óptico, la superficie focal del detector, la electrónica y la estructura mecánica. Estos componentes se muestran en la figura 1.11. El subsistema óptico consiste de dos lentes de Fresnel y una lente difractiva de precisión, lo que proporciona una apertura de ±30° de campo de visión ó FoV (*Field of View*), de forma que la luz UV proveniente de EAS que ingresa, es enfocada sobre la superficie focal con

una resolución espacial de 0.1°. La superficie focal se encuentra compuesta por un arreglo de alrededor de 6000 fotomultiplicadores multiánodo (MAPMT) que convierten la energía de los fotones incidentes en pulsos eléctricos con una duración típica de 10 ns. La electrónica cuenta los pulsos eléctricos producidos por los MAPMT en periodos de 2.5 μs y almacena las cuentas en memoria. Cuando se encuentra una señal correspondiente a un evento de partículas, el sistema electrónico genera una señal de disparo (*trigger*) y transmite todos los datos de la imagen almacenada en memoria al centro de operación que se ubica en tierra.

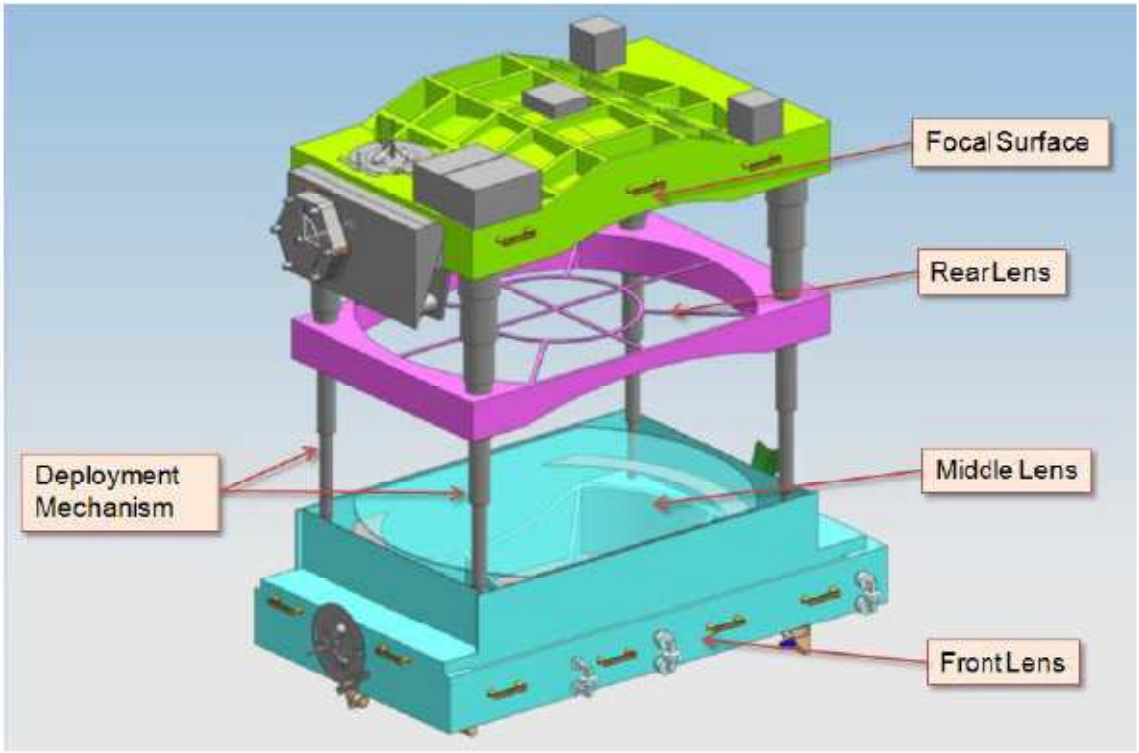


Figura 1.11. Vista esquemática del telescopio JEM-EUSO. Imagen tomada de [11].

En lo que concierne al monitoreo atmosférico, el telescopio JEM-EUSO utilizará una cámara infraroja y un Lidar (*Light Detection and Ranging*) con un láser ultravioleta a fin de observar las condiciones de la atmósfera en el FoV del telescopio y determinar el tiempo de observación efectivo así como incrementar la confiabilidad de los eventos detectados alrededor del umbral de energía. El papel del Lidar recae en observar las condiciones de nubes en diferentes puntos del campo de visión del telescopio y calibrar con precisión la relación entre altitud de nubes y su temperatura, que se obtiene a partir del análisis de las

imágenes de la cámara infraroja. Por otra parte, al utilizarse un láser con longitud de onda de 355 nm, la superficie focal del telescopio servirá también como receptor del Lidar.

En lo que respecta a la calibración del instrumento JEM-EUSO, ésta se realizará en dos etapas: calibración de pre-lanzamiento y calibración post-lanzamiento o en funcionamiento del instrumento.

La calibración de pre-lanzamiento, que es sobre la que se basa el presente trabajo, se lleva a cabo durante el diseño y construcción del telescopio. En específico, se realiza la calibración de los bloques que conforman la superficie focal, los cuales se encuentran constituidos por MAPMT. Con esta calibración se pretende optimizar el diseño de los algoritmos de *trigger* para la detección de eventos de EECR, y ayudar en la optimización de la transmisión de información desde la Estación Espacial Internacional. En el capítulo 2 se hará mención a detalle sobre esta calibración de pre-lanzamiento.

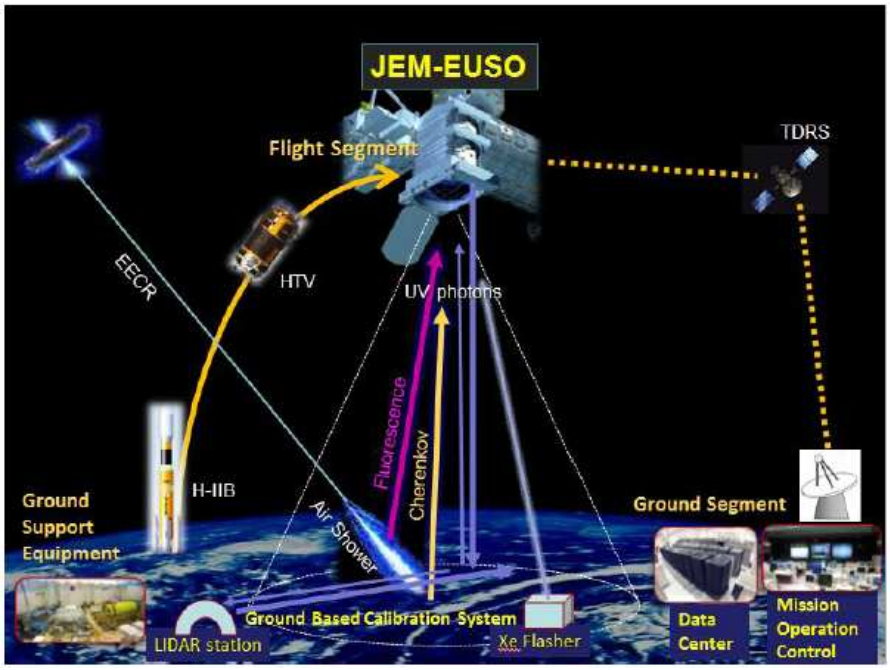
La calibración post-lanzamiento ó calibración durante el funcionamiento del telescopio se encuentra dividida en dos etapas: calibración abordo y calibración desde tierra.

El sistema de calibración abordo se compondrá de un set de tres LEDs con diferentes longitudes de onda (en el rango de 300 a 500 nm) que serán instalados en el cilindro del telescopio y servirán como fuentes de luz difusa. Cuando la superficie focal mide la luz proveniente de estos LEDs que pasa a través del sistema óptico así como la luz que se refleja en las paredes y cara interna de la tapa del telescopio, se calibrará la ganancia y la eficiencia de detección del instrumento.

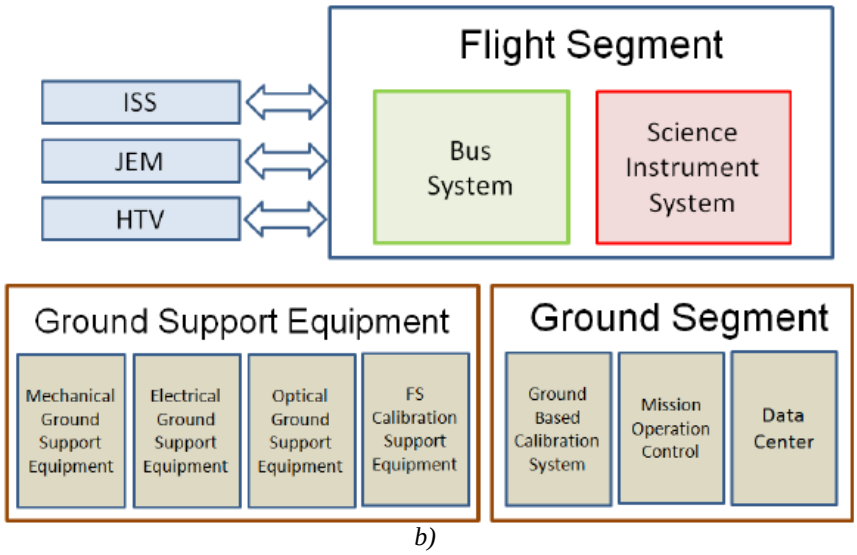
Para la calibración desde tierra, se utilizarán lámparas flash de Xenón en una docena de sitios alrededor del planeta como se muestra en la figura 1.12. Cuando el telescopio pase por encima de alguno de estos sitios una vez por día, detectará la luz proveniente de las lámparas y medirá la absorción en la atmósfera para la región UV de interés, con lo cual se tendrá la calibración en ese momento para esa zona en la atmósfera. Así mismo, con la intensión de estimar el error sistemático en la energía y dirección de arribo de los rayos

cósmicos primarios, el telescopio JEM-EUSO observará un láser UV disparado desde estaciones Lidar en tierra que simularán las EAS. Esta observación también permitirá estimar la transmitancia de la atmósfera como función de la altitud.

La figura 1.12 muestra, en resumen, una composición de toda la misión.



a)



b)

Figura 1.12. a) Vista conceptual de toda la misión JEM-EUSO. b) Componentes separados de la misión. Imagen tomada de [11].

1.3.4 Simulación del instrumento

En el contexto del proyecto JEM-EUSO, se ha desarrollado una herramienta de simulación "*end-to-end*" de la observación de EAS utilizando un detector espacial, la cual se denomina ESAF (EUSO *Simulation and Analysis Framework*). El código ESAF contiene todo el proceso de simulación y reconstrucción desde la interacción de las partículas primarias en la atmosfera terrestre hasta la reconstrucción final del evento.

Aunque el código se realizó teniendo en mente el diseño específico de EUSO, es lo bastante flexible como para permitir obtener resultados más generales que pueden ser útiles para cualquier proyecto futuro.

El código ESAF está escrito principalmente en C++ con algunas librerías externas en Fortran y su diseño orientado a objetos permite flexibilidad y modularidad en su uso. Se encuentra dividido en dos secciones independientes, simulación y reconstrucción de eventos, que comparten la infraestructura así como librerías del código. La estructura principal del programa y las interacciones entre las dos secciones principales se muestran en la figura 1.13.

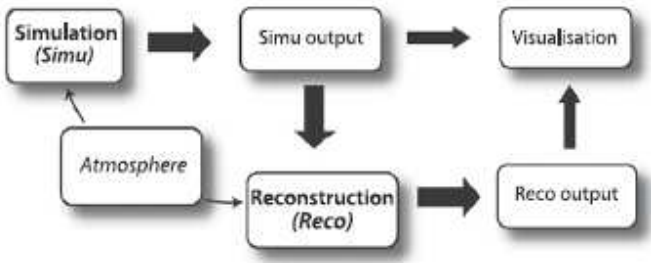


Figura 1.13. Estructura principal de ESAF. Imagen tomada de [12].

El módulo de simulación se encuentra estructurado en 6 secciones: simulación de lluvias producidas por flujos de UHECP; fluorescencia y fotones Cherenkov producidos en la atmósfera; propagación de luz desde el lugar de su producción y en dirección al telescopio, con simulaciones de las interacciones de fotones durante el proceso de propagación de los mismos; simulación de la óptica del telescopio (lentes y superficie focal); y simulación de la electrónica *Front-End* y niveles de disparo (*trigger*). La simulación se proporciona en un

archivo de ROOT con una estructura que contiene la misma información esperada de datos reales así como un conjunto de datos Monte-Carlo. El nivel de detalles puede ser configurado por el usuario (tipo de física, óptica, *trigger*, etc.). Proporciona fácil acceso al usuario con visualización de histogramas así como vistas en 2D y 3D para cada evento. La figura 1.14 muestra la estructura del módulo de simulación.

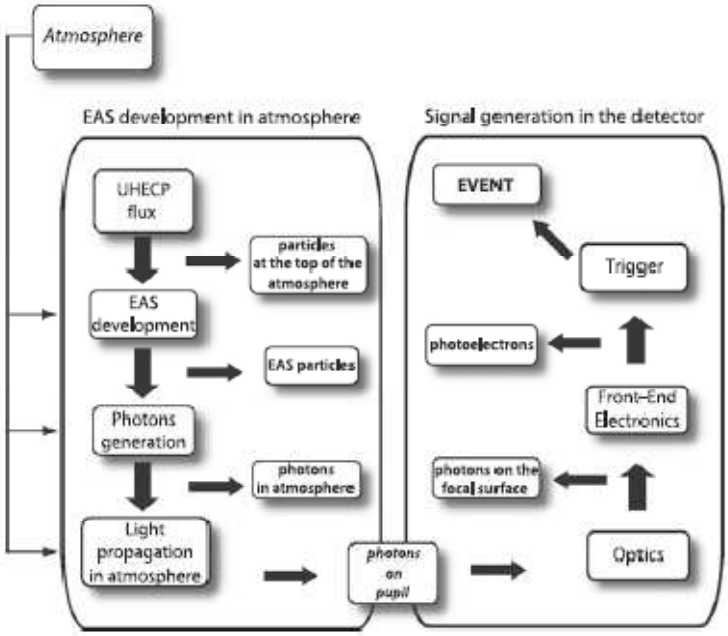


Figura 1.14. Estructura del módulo de simulación de ESAF. Imagen tomada de [12].

En lo que se refiere al módulo de reconstrucción, éste lee el archivo ROOT que representa la simulación completa de la lluvia (EAS) junto con el *background*, y reconstruye los eventos tratándolos como si fueran datos reales del telescopio. Diferentes sub-módulos dentro del módulo de reconstrucción permiten realizar reconocimiento de patrones, identificación de señales de EAS por encima de la luz de estrellas, de la Luna y otras fuentes de luz de fondo, reconstrucción en 3D de la dirección de arribo y energía. En la figura 1.15 se muestra la estructura del módulo de reconstrucción.

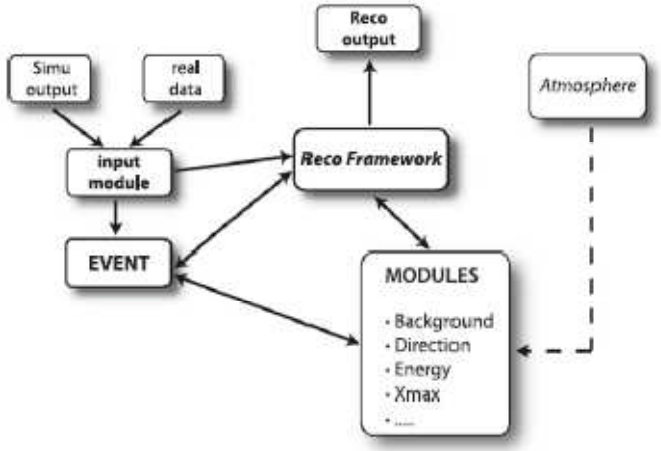


Figura 1.15. Estructura del módulo de reconstrucción de ESAF. Imagen tomada de [12].

El sistema de reconstrucción es la principal estructura que adquiere datos (reales o simulados) desde un módulo de entrada y construye la cadena de módulos necesarios para reconstruir los eventos. La reconstrucción de eventos se divide en diferentes tareas (por ejemplo, extracción de *background*, reconstrucción de dirección de arribo, etc.) y para cada tarea se encuentran disponibles diferentes módulos. El empleo de tales módulos permite reemplazar o excluir fácilmente un módulo determinado, y de esta forma, diferentes algoritmos o combinación de algoritmos pueden ser probados y comparados para el mismo evento. Por otra parte, al igual que en el módulo de simulación, los módulos de reconstrucción pueden ser configurados por el usuario, lo cual se realiza gracias al módulo que almacena los datos de toda la información relevante sobre la reconstrucción de eventos (*event container*), de tal forma que cada módulo puede accesar a esta información, leer los datos existentes y proporcionar sus propios resultados.

Capítulo 2

El instrumento Track-Sim

No hay otra manera de alcanzar la eternidad que ahondando en el instante, ni otra forma de llegar a la universalidad que a través de la propia circunstancia: el hoy y aquí.
E. Sabato, "La resistencia".

2.1 MOTIVACIÓN

La evolución espacial y temporal en intensidad del perfil de un EAS proporciona información muy útil sobre la energía e identidad de la partícula primaria, sin embargo, existen algunas dificultades en la realización práctica de la observación, y más concretamente en la catalogación de los eventos de interés, debido principalmente a que el ancho de banda para la transmisión a tierra de las señales es limitado. Es por esto que los algoritmos correspondientes a los diferentes niveles de *trigger* con que cuenta el telescopio, deben reducir el nivel de disparos en un factor de 10^6, con la intensión de elegir únicamente la información útil, lo que vuelve fundamental la validación de dichos algoritmos.

Al conocer las condiciones en que se llevan a cabo las observaciones y más aún, la detección de los eventos a los que los objetivos de la misión se enfocan, es necesario contar con una gran colección de datos mediante los cuales puedan llevarse a cabo los análisis que permitan determinar el tipo de evento detectado y su naturaleza, sin embargo el envío de una gran cantidad de datos desde el telescopio espacial, y más en concreto, desde la Estación Espacial Internacional, se convierte en un gran inconveniente debido a la limitación en ancho de banda para la transmisión de tal volumen de datos por periodo de observación de la misión JEM-EUSO. Aunado a esto, se tiene el problema inicial de identificar las señales que corresponden a eventos físicos de entre el ruido de fondo (*background*) en el que las observaciones tienen lugar, así como la "predilección" por partículas asociadas a física de mayor interés para la comunidad científica internacional en la actualidad (por ejemplo la detección de neutrinos de ultra alta energía). Estos problemas pusieron de manifiesto la necesidad de potenciar la detección de los eventos de interés mediante el diseño de algoritmos de *trigger*, los cuales permitan "seleccionar" de entre todos los posibles eventos que se detectan, aquellos que representen realmente eventos físicos que sean de interés científico. La filosofía de este "filtro" de datos en que se constituye el *trigger* del telescopio, se encuentra en el concepto mismo de todo el instrumento, y está basado en las propiedades estadísticas del flujo de fotones que arriban a la superficie focal del telescopio, así como en el análisis temporal del mismo.

Con el fin de combatir los problemas para la calibración de pre-vuelo del telescopio se plantea el desarrollo del instrumento Track-sim. La finalidad del simulador de trazas Track-Sim es el reproducir los perfiles correspondientes a las trazas realistas de partículas a lo largo de los bloques que conforman la superficie focal (FS) del telescopio JEM-EUSO, a partir de perfiles de intensidad simulados numéricamente, con la intensión de determinar la respuesta de la superficie focal. Para lograr esto se reproducirán espacial y temporalmente el flujo de fotones para las diferentes energías y geometrías posibles, en el rango de 300 a 400 nm que incluye a las 3 líneas espectrales de fluorescencia del nitrógeno (N_2) que el experimento pretende observar. Este sistema se basa en la proyección de luz UV que utilizará como fuente un arreglo de LED's, el cual junto con un sistema óptico, reproduce diversas trazas sobre los módulos foto-detectores (PDM) que son los módulos mediante los cuales se ensambla la superficie focal del telescopio.

2.2 OBJETIVOS Y METAS

El presente trabajo tiene como principal objetivo la elaboración y validación de una estrategia para el control de intensidad de perfiles de luz ultravioleta, los cuales servirán, en el futuro, como la base para la realización del instrumento Track-Sim. Se plantea, primeramente, realizar un prototipo reducido como modelo de validación. Para lograr esto se fijaron las siguientes metas:

1. Proposición y validación de una metodología para controlar la intensidad de fuentes de luz ultravioleta en número de fotones detectados por un dispositivo fotomultiplicador multiánodo.
2. Diseño y validación de la arquitectura del hardware para un prototipo de 8 canales independientes, correspondientes a 8 pixeles de un dispositivo MAPMT.
3. Diseño del hardware de control para la matriz del instrumento final Track-Sim.

2.2.1 Ancho de banda para transmisión a tierra de eventos físicos

Como cualquier nave o satélite, la Estación Espacial Internacional (ISS), en la que se situará JEM-EUSO, cuenta, dentro de sus diversos sistemas, con un sistema de comunicaciones que se enlaza con diversas estaciones en tierra, las cuales transmiten la información en forma bidireccional entre la ISS y los países que tienen presencia en ésta (principalmente Estados Unidos, la Unión Europea y Rusia). El sistema de comunicaciones se sitúa dentro de los sistemas más importantes con que cuenta la ISS debido a que de éste dependen muchos de los experimentos que se realizan a bordo, pero principalmente, depende el contacto con los astronautas que se encuentran dentro de la estación espacial.

Las comunicaciones entre la ISS y tierra se realizan mediante el uso varias estaciones en tierra, por lo que en diversas regiones del planeta se tienen estaciones que reciben y envían información entre las diferentes agencias espaciales y la ISS. Además de los segmentos de tierra, se tienen satélites de comunicaciones para realizar enlaces entre la ISS y las estaciones centrales. Sin embargo, existen limitaciones para el envío de información debido a cuestiones tecnológicas en el ancho de banda disponible así como la logística que una cuestión así representa, siendo un problema constante desde los primeros experimentos y vuelos espaciales que se realizaron. Aunado a esto, los problemas se han incrementado en años recientes al avanzar la tecnología de sensores e incrementarse la cantidad de información que se genera en, por ejemplo, sistemas satelitales que monitorizan actividades climáticas como gases atmosféricos, eventos nucleares, así como la creciente transmisión de telecomunicaciones en todo el mundo, lo que produce mayor cantidad de información de la que un sistema de enlace en cualquier satélite puede manejar en un tiempo razonable, representando un serio problema y un cuello de botella para las comunicaciones espaciales.

Aún cuando la ISS cuenta con diferentes comunicaciones de radio en diferentes frecuencias para realizar los enlaces con los sistemas en tierra, como por ejemplo las transmisiones rusas que tienen comunicación ocasional con las estaciones terrenas norteamericanas utilizando VHF-1 y VHF-2 (a 143.625 y 130.167 MHz respectivamente) [13], éstas no son suficientes. Existen otros sistemas como los que emplean antenas tipo dipolo como en el

caso de las naves Soyuz, el transbordador espacial y los sistemas de voz de la ISS así como sistemas de telemetría y seguimiento, o como el recientemente instalado sistema de internet a bordo de la estación espacial internacional utilizando el sistema de comunicación de alta velocidad en banda Ku [14], sin embargo, las diferentes necesidades de los experimentos presentes, así como los futuros, imponen limitaciones en los recursos de que se dispone.

Algunas opciones para solucionar el problema del ancho de banda en aplicaciones espaciales establecen el tener sistemas dedicados satélite-Tierra única y exclusivamente para determinadas aplicaciones, empero, los costos de construir e implementar tales sistemas son tan elevados que resultan inviables en la mayoría de los casos. Otras posibilidades se refieren a la construcción de redes de comunicaciones dedicas, lo que disminuye el costo de un sistema único dedicado, sin embargo aún resultan caras y sólo algunas aplicaciones hacen uso de tales redes (como por ejemplo la red de comunicaciones de la NASA denominada **Tracking and Data Relay Satellite System** ó **TDRSS**, en donde se ha puesto mayor énfasis en el sistema satelital a fin de disminuir la cantidad de estaciones terrenas).

Algunas otras soluciones involucran el desarrollo de tecnologías de comunicación óptica en espacio libre, en donde se busca incrementar la tasa de transferencia de datos desde la ISS con otras naves y sondas espaciales, utilizando redes de microsatélites así como telescopios receptores de seguimiento rápido, esto con el fin de incrementar la tasa de transferencia desde decenas de Kbps hasta 2.5Gbps [15].

Una solución que realmente aliviaría la congestión en el área de comunicaciones satelitales recae en el cambio y la creación de nuevos paradigmas en computación y electrónica. Algunos experimentos a bordo de la estación espacial internacional indican que estos problemas en la transmisión de datos pueden ser remediados utilizando chips de computadora más poderosos y complejos, que orbiten junto con los sistemas satelitales a fin de reducir tal cantidad de datos. Sin embargo, esto recae en el conocido dilema sobre la utilización de tecnología de punta en aplicaciones espaciales, en donde las tecnologías probadas desde tiempo atrás siguen siendo las prevalecientes en el ramo espacial, mientras

que los avances de punta permanecen, en su mayoría, fuera de estas aplicaciones. Si bien la mayor capacidad en cómputo a bordo de satélites con nueva tecnología resultaría en que sólo la información útil sería transmitida a Tierra, el principal inconveniente en tecnología electrónica y computacional es como funcionarían los elementos electrónicos de última generación en el difícil ambiente espacial. El mayor temor ha sido el impacto de partículas de alta energía con los transistores de los cada vez más densos dispositivos electrónicos, provocando por ejemplo, cambios en los estados lógicos que alteren los valores de cálculos individuales, lo que produciría resultados incorrectos en incluso la pérdida total del o los dispositivos en cuestión.

Debido a éstos problemas en la transmisión de datos desde la ISS, y tomando en consideración la gran cantidad de datos que se espera genere JEM-EUSO durante los periodos de funcionamiento, se han tomado diferentes decisiones a fin de disminuir la cantidad de datos que se transmitirán a Tierra de sólo los eventos físicos de interés. El problema de esto es determinar con precisión qué datos dentro de la gran cantidad de información generada son los de mayor relevancia científica.

2.2.2 Algoritmos de *trigger* del telescopio

El instrumento JEM-EUSO utiliza un método de selección de la información que colecta con la intención de obtener únicamente los datos relevantes. La manera de realizar esta selección se basa en algoritmos de *trigger* ó disparo, mediante los cuales, se pretende detectar la presencia de señales debidas a eventos físicos dentro de todo el ruido de fondo.

Debido a que el número de pixeles que contiene el telescopio es grande ($\sim 2\times10^5$) se ha desarrollado un esquema de *trigger* multinivel, particionando la superficie focal en módulos más pequeños denominados PDM (*Photo Detector Module*), los cuales son lo suficientemente grandes para contener parte importante de las trazas lumínicas producidas por las EAS. Cada PDM se acomoda y es identificado mediante coordenadas cartesianas como XPDM y YPDM, mientras que los pixeles dentro de cada PDM son nombrados como XY únicamente.

El sistema de *trigger* se conforma principalmente por dos partes principales organizadas en subniveles. Estas dos secciones de *trigger* trabajan sobre las propiedades estadísticas del flujo de fotones que ingresan al telescopio, con la intención de detectar eventos físicos contenidos dentro del ruido de fondo, a partir de un análisis de su posición y correlación temporal. En la tabla 2.1 se muestra una primera estimación de los factores de reducción del *trigger* para las señales de interés y se espera que en el futuro próximo estos valores sean mejorados.

Level		**Rate of signals/triggers at PDM level**	**Rate of signals/triggers at FS level**
1st level trigger (PDM)	Photon trigger	~9.2×10^{8} Hz	~1.4×10^{11} Hz
	Counting trigger	~7.1×10^{5} Hz	~1.1×10^{8} Hz
	Persistency trigger	~7 Hz	~10^{3} Hz
2nd level trigger (PDM cluster)		~6.7×10^{-4} Hz	~0.1 Hz
Expected rate of cosmic ray events		~6.7×10^{-6} Hz	~10^{-3} Hz

Tabla 2.1. Esquema de la capacidad de reducción de ruido de fondo. Tabla tomada de [11].

El primer nivel de trigger se implementa en un dispositivo FPGA dedicado (*Field Programmable Gate Array*) en cada módulo PDM, en donde se conecta a 9 celdas elementales ó EC (*Elementary Cell*) para manejar 36 MAPMTs con un total de 2304 canales. Este primer nivel de trigger consiste a su vez de tres subniveles organizados de la siguiente forma:

a) Primer subnivel: nivel de *trigger* a nivel ánodo de MAPMT que consiste básicamente de un discriminador analógico a fin de reconocer eventos correspondientes a fotoelectrones únicos (*single-photoelectron*) por cada ánodo de MAPMT.

b) Segundo subnivel: consta de un *trigger* digital a nivel pixel que accede a un contador y un comparador digitales. El tiempo de acceso se ha denominado GTU (*Gate Time Unit*) y tiene una duración de 2.5µs. Este nivel de *trigger* se activa cuando un número determinado de *single-photoelectron* que es registrados por una

cadena de ánodos a lo largo de un GTU sobrepasa un valor de umbral previamente establecido.

c) Tercer subnivel: Este subnivel de *trigger* es digital y se activa cuando persiste la actividad por encima del valor establecido del segundo subnivel de *trigger* en GTUs consecutivos en un PDM o parte de éste. Consiste en pixeles dedicados de conteo, agrupados en conjuntos de 2×2 ó 3×3 que son comparados con valores de umbrales previamente establecidos.

El sistema de *trigger* del telescopio será completamente programable en vuelo con el objetivo de operar en diferentes modos para identificar procesos físicos de diferente índole:

- Modo Estándar para detección de EECR.
- Modo lento.
- Modo rápido.
- Modo de *trigger* analógico (sub-modo).

En el modo estándar ó de EECR se buscarán señales que superen el segundo subnivel del primer nivel de *trigger* con tiempos de entre 30µs y 300µs. Los eventos que no cumplan con el periodo de duración mencionado serán ignorados.

El modo lento observará fenómenos atmosféricos como meteoros. Este modo de operación se basará en la actividad del tercer nivel de *trigger* que tenga duración que supere 300µs.

El modo rápido funcionará con una frecuencia de muestreo 8 veces superior a la del modo EECR. En este modo se buscarán señales en el tercer nivel de *trigger* con actividad menor a 30µs. Con este modo de operación se realizará la calibración de vuelo.

El modo de *trigger* analógico permitirá al instrumento dispararse cuando: i) se presenten fenómenos transitorios de baja duración (<<1 GTU) pero intensos (por ejemplo marcas de

luz Cherenkov o luz Cherenkov debida a neutrinos Tau), y ii) se presenten eventos con velocidad de propagación mucho menor que la luz (meteoros y descargas atmosféricas).

El segundo nivel de *trigger* se implementa a nivel clúster (conjuntos de 8 PDMs) en la denominada *Cluster Control Board* (CCB), donde un chip FPGA realiza el algoritmo concerniente al segundo nivel de *trigger* denominado LTT (*Linear Track Trigger*). Éste método se implementa en el circuito electrónico de PDM y sigue la siguiente estrategia:

a) Cuando se presenta un primer nivel de *trigger* los pixeles de un PDM se dividen en dos categorías (pixeles amarillos y blancos). Sólo se utilizan pixeles amarillos para integrar la señal del *track* con fines de *trigger*. Los pixeles blancos tienen muy poca señal o ausencia total de ésta con lo que son descartados y no se utilizan en los análisis de *trigger*.

b) Por encima del primer nivel de disparo, el algoritmo define una “caja”. Se integra a los pixeles amarillos dentro de la “caja” definida. La ubicación de la “caja” de análisis varía de GTU a GTU de acuerdo con el análisis de la dirección específica.

c) Después de definir la ubicación de la caja, se integra el contenido de los pixeles amarillos y el número total de fotoelectrones se compara con un umbral preestablecido. Esto depende fuertemente del valor promedio del ruido de fondo y de la razón de eventos falsos que es aceptable por el experimento.

La figura 2.1 muestra algunos esquemas del software de reconstrucción del sistema de *trigger* del telescopio.

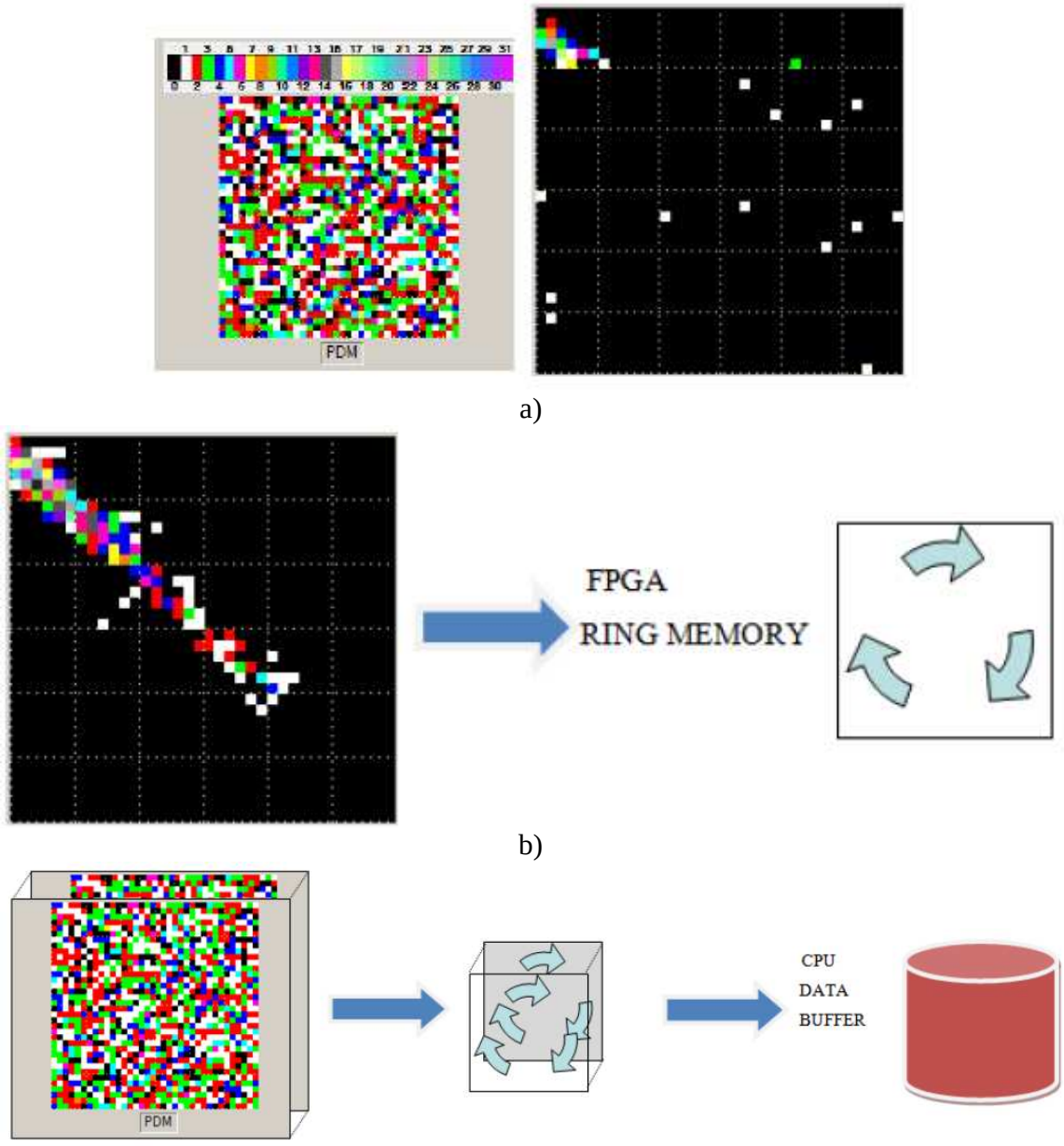


Figura 2.1. Reconstrucción de los eventos tomados por el sistema de trigger del telescopio. a) A la izquierda el ruido de fondo tomado en un PDM en donde los colores indican el número de fotoelectrones contados en cada pixel. A la derecha se muestra la actividad de persistencia en el segundo subnivel de trigger. b) Al final del tiempo de exposición, el instrumento almacena los valores en memorias. c) Lectura del contenido de memorias. Imagen tomada de [11].

2.3 PRINCIPIO DE FUNCIONAMIENTO DEL INSTRUMENTO TRACK-SIM

2.3.1 Distribución longitudinal para diferentes tipos de partículas

El objetivo del instrumento Track-Sim es reproducir los perfiles originados por las EAS, sobre los bloques PDM que conforman la superficie focal del telescopio JEM-EUSO. Dichos perfiles varían espacial y temporalmente, por lo que la forma de las trazas de fotones UV se determina por las características de la EAS que las producen. A su vez, el desarrollo de cada EAS está en función de los siguientes parámetros:

- Tipo de partícula primaria de EECR.
- Energía de la partícula primaria de EECR.
- Incidencia de las partículas primarias de EECR (dirección de desarrollo de la EAS).
- Cobertura de nubes y aerosoles en la atmósfera al momento de realizar la observación (procesos de absorción y dispersión de fotones).
- Reflexión en la superficie terrestre (importante para la detección de luz Cherenkov).

La simulación de EAS, y en específico, de los fotones que arriban al telescopio, se han realizado utilizando diferentes aproximaciones y herramientas como son CONEX, Saitama ó ESAF. De especial interés resulta el código ESAF debido a que se encuentra diseñado para la simulación del instrumento JEM-EUSO.

De las simulaciones realizadas con ESAF, la que resulta de mayor interés para el instrumento Track-Sim es la de los fotones que alcanzan la superficie focal del telescopio dado que es precisamente esto lo que se pretende reproducir.

Cabe mencionar que la cantidad de fotones que se producen durante una cascada de partículas en la atmosfera es demasiado grande (~10^{15} fotones para partículas primarias con energías del orden de 10^{20} eV) sin embargo sólo alrededor de 10^{4} fotones alcanzan el telescopio [12]. Esta reducción en el número de fotones se debe al ángulo sólido que subtiende el telescopio hacia la atmósfera terrestre, lo cual resulta ventajoso para realizar la simulación de la propagación de fotones dado el hecho de que para detectores espaciales,

estos se localizan fuera del medio de dispersión que constituye la atmósfera. Para el caso particular del telescopio JEM-EUSO a 400 km de altura, el ángulo sólido es de 3×10^{-11} sr [12]. La figura 2.2 muestra éste hecho.

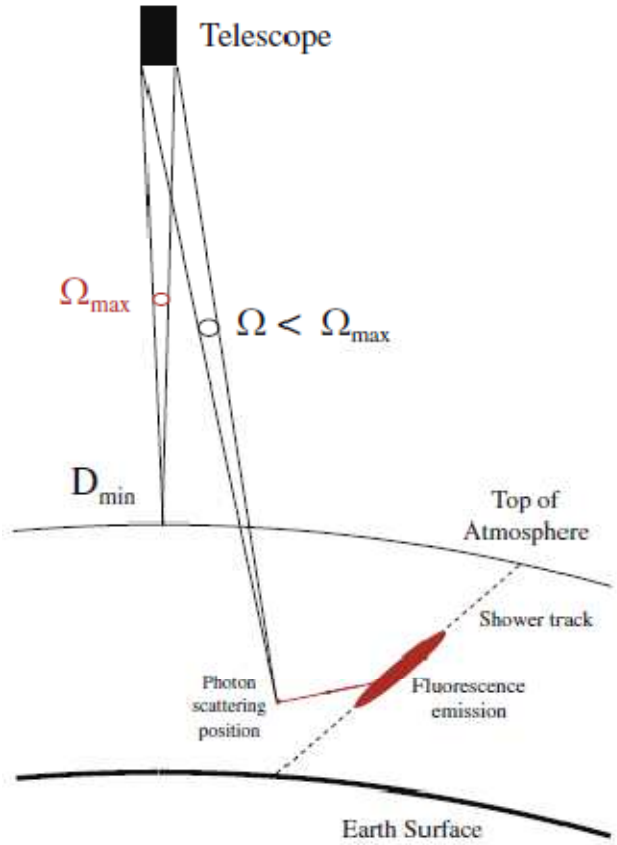


Figura 2.2 Distancia mínima entre el tope de la atmósfera (considerada a una altitud de 30 km) con el telescopio, y ángulo sólido del detector mediante el cual se reduce la cantidad de fotones que arriban al telescopio. Imagen tomada de [12].

De los 10^4 fotones que alcanzan el telescopio no todos llegan a la superficie focal, por lo que el número de fotones final que pueden ser detectados es del orden de algunos cientos. A continuación se muestra en la figura 2.3 un ejemplo realizado con ESAF de un histograma de la cantidad de fotones estimados que alcanzan la pupila del telescopio, el número de fotones que arriban a la superficie focal y el número de fotones que son detectados, entre otros parámetros.

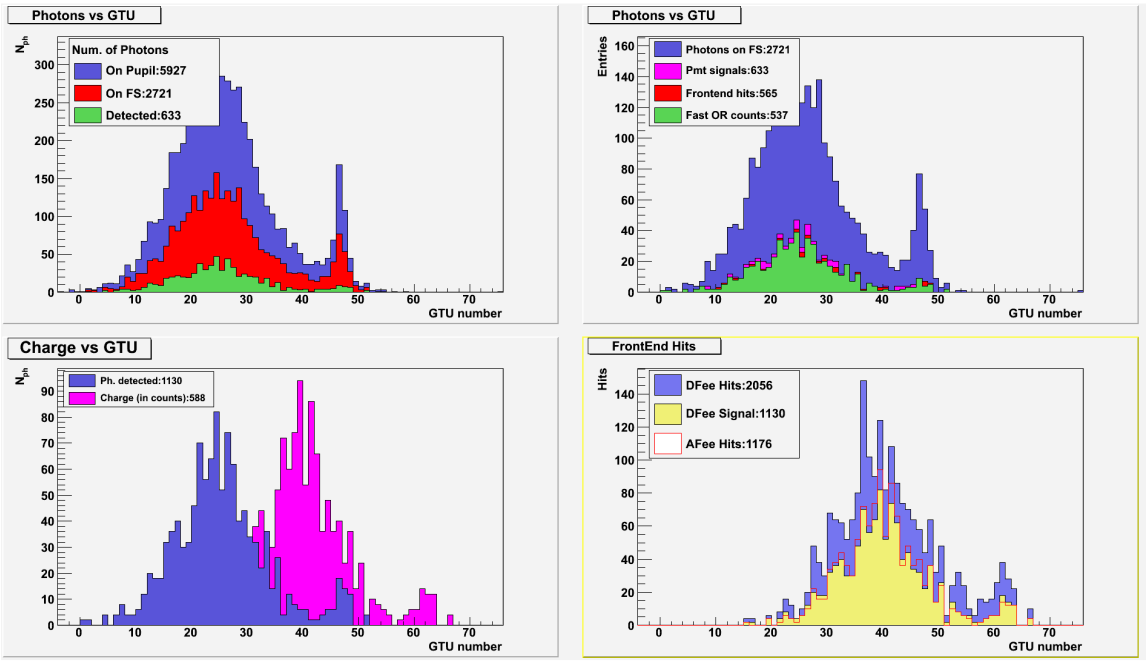
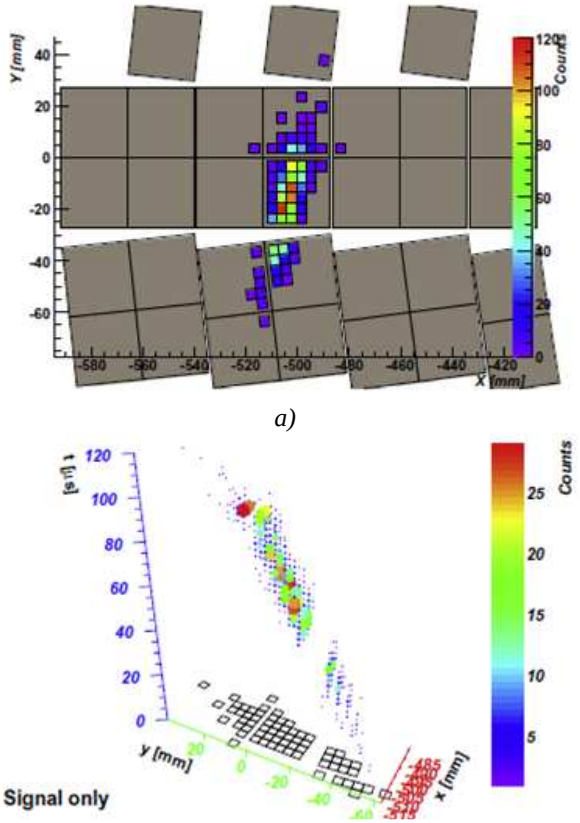


Figura 2.3. Histogramas de fotones estimados que llegan al telescopio en la pupila y superficie focal, así como fotones detectados, señales de cuentas en PMTs, electrónica Front-End ("detected", "frontend hits" y "fast OR counts") y carga en cuentas para protones con E = 10^{20}eV.

Los fotones que alcanzan la superficie focal se verán como un perfil que depende de la forma en que se desarrolla cada lluvia, pasando por un mínimo al comienzo y hasta que se alcance un máximo, para pasar nuevamente a un mínimo. La forma en que se observará esto sobre la superficie focal del telescopio aparece en la figura 2.4, en donde básicamente la electrónica realizará un conteo del número de fotones-electrones en cada pixel del arreglo de PMTs. Lo importante a destacar aquí son las características de las trazas a reproducir, las cuales, dependen del tipo de partícula de que se trate (protones, iones pesados, neutrinos, etc.), y que se forman por emisión de fotones UV en el intervalo de 0 a 300 fotones por GTU (recordando que cada GTU consta de 2.5µs) con una extensión longitudinal en la práctica (en la dirección temporal) de hasta 30cm, mientras que perpendicularmente puede tener hasta ~20 mm (sin considerar los fotones de "background" presentes en todo momento y que aparecen por toda la superficie focal).



a)

b)

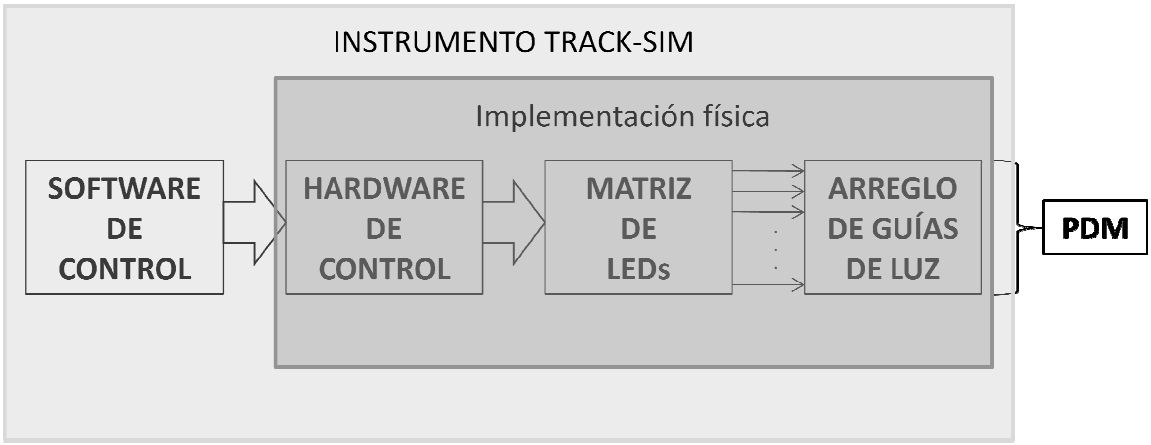
Figura 2.4. Simulación de una traza correspondiente a una EAS de 10^{20}eV en la superficie focal sin simulación de fotones de "background". (a) Vista en 2D de los pixeles iluminados por la traza (los cuadros grandes corresponden a cada MAPMT). (b) Vista 3D que incluye el tiempo contado en GTUs de 2.5μs. Los ejes x e y corresponden a las coordenadas xy de los pixeles mientras que el eje z corresponde al tiempo a partir de la primer señal generada por conteo de fotones. En el plano xy se proyecta el perfil de los pixeles con señal. Imagen tomada de [12].

Este tipo de perfiles son los que el instrumento Track-Sim debe generar sobre los bloques PDM. Para esto, el sistema de control partirá de los datos obtenidos de las simulaciones en ESAF para reproducirlos físicamente sobre los módulos fotodetectores (PDM) incluyendo el *background*.

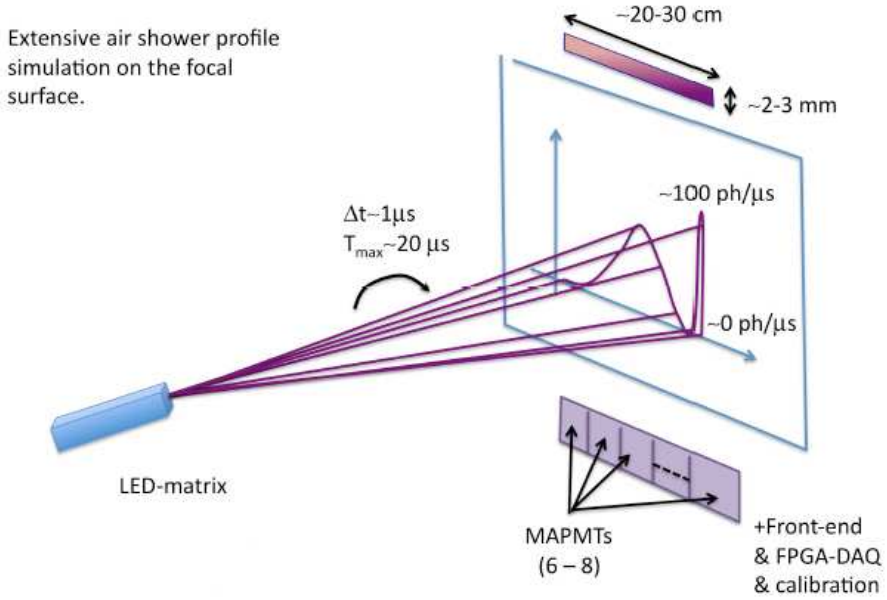
2.4 COMPONENTES DEL INSTRUMENTO TRACK-SIM

El instrumento Track-Sim se encuentra conformado por diferentes sistemas, los cuales se encuentran determinados por la función que cada uno desempeña, además de su dependencia entre sí. El producto final debe ser una cámara compacta y robusta, lista para ser usada por un consumidor final, y acompañada de una computadora portátil de control, la

cual contendrá un software de control e interfaz de usuario, así como una librería de trazas para proyección, manual de usuario y hoja de datos con información sobre la calibración del dispositivo. Un bosquejo del instrumento se muestra en la figura 2.5. A continuación se presentará en breve detalle los principales sistemas que conforman y se utilizan para la operación y calibración del instrumento Track-Sim.



a)



b)

c)

Figura 2.5 a) Esquema general del instrumento Track-Sim. b) Principio de operación para generación de perfiles. c) Bosquejo del instrumento final.

2.4.1 Sistema de control

El sistema de control, como su nombre lo indica, se encarga de operar el instrumento en todos los modos posibles de éste, así como brindar al usuario la interfaz de interacción y comunicación con el instrumento. Se encuentra constituido principalmente por dos partes: hardware y software.

El subsistema de hardware se compone de la electrónica que controla la fuente emisora de luz UV en el sistema óptico. El diseño y conformado del subsistema de hardware se presentarán en detalle en el capítulo 3, mientras que en el apéndice 4 se muestra lo concerniente a algunos avances sobre los algoritmos para el software de control, por lo que en esta sección sólo se realizará un breve bosquejo de este subsistema.

El hardware que controla la fuente de luz UV se conforma de un generador de pulsos múltiples, en donde se puede variar el ancho de los pulsos. En total se tienen 96 canales, cada uno capaz de generar pulsos con anchos diferentes y de operar todos a la misma y máxima frecuencia. La generación de dichas señales se realiza mediante un dispositivo FPGA, en el cual se descargarán los patrones que corresponderán a los diferentes perfiles a reproducir. La conexión entre el generador de pulsos y la fuente UV se realiza mediante

switches a base de dispositivos *buffer* y MOSFET, que son los que directamente manejan a la fuente UV.

En lo que respecta al software de control, su principal función es la de realizar la conversión del perfil correspondiente al número de fotones por GTU obtenido de ESAF, en un patrón binario al que se codificará dicho perfil para ser descargado en el hardware y posteriormente reproducirse como pulsos de ancho variable. Así mismo, se encarga de operar en diferentes modos al hardware (escritura, lectura, corrida de secuencias en forma continua y de una sola vez) y de fungir como interfaz entre el usuario y el instrumento. Este software operará en ambiente Linux y contendrá también una serie de librerías dentro de las cuales se ubicarán los archivos generados en ESAF para diferentes configuraciones de energías y geometrías, los cuales podrán ser seleccionados por el usuario para su reproducción sobre PDMs.

A continuación se presenta una simplificación del diagrama de flujo (figura 2.6) de la operación del sistema de control (hardware y software) el cual será comentado a detalle dentro del capítulo 3 y apéndice 4.

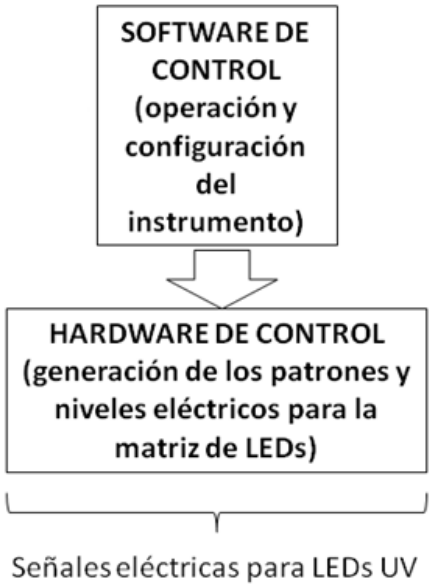
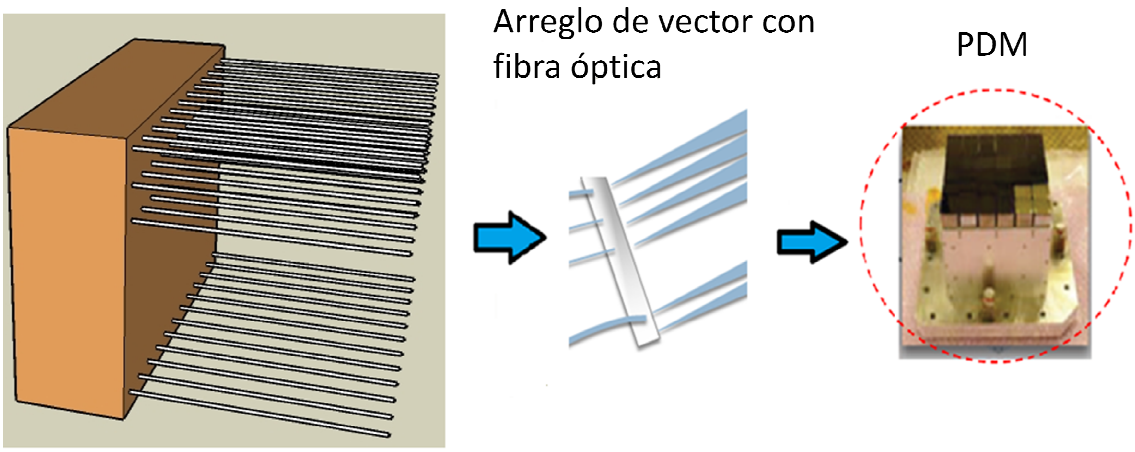


Figura 2.6. Diagrama de flujo simplificado de la operación del sistema de control.

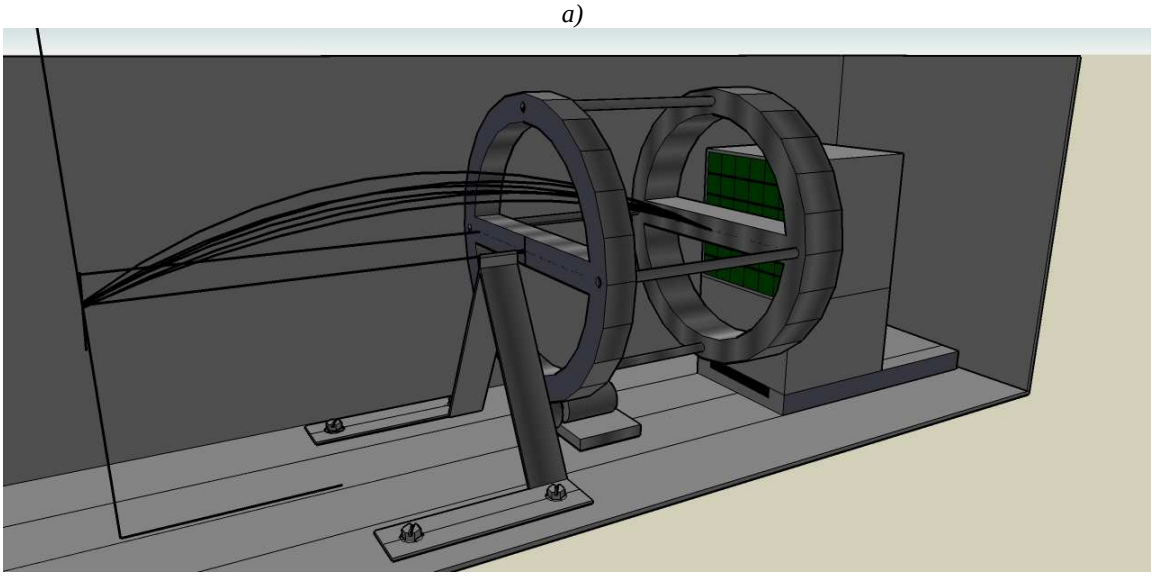
2.4.2 Sistema óptico y mecánico

El sistema óptico y mecánico del instrumento se constituye principalmente de la fuente emisora de luz UV, así como de los canales ópticos correspondientes en donde la fuente se acopla para tener una emisión sobre todos los pixeles de un PDM. Toda la estructura se implementará en forma de una matriz que encaja en el tamaño de un PDM, y que en el lado en que se ubican la fuente luminosa, tendrá un tamaño mayor (dado el tamaño de la fuente a utilizar) por lo que se terminará en una forma de embudo. Así mismo, cada canal contendrá una fibra óptica que transmite en el UV y mediante la cual se guiará la luz de cada LED a un pixel de cada MAPMT.

Para llegar al diseño final de la parte óptica y mecánica del dispositivo, se pasaron por diversos diseños conceptuales, primeramente con una estructura que giraría 90° conformada por un vector de 48 LEDs contenido en el diámetro interior de la circunferencia, junto con 8 LEDs de alineamiento. Toda esta estructura se posicionaría enfrente de un PDM. Otro diseño contemplaba el mismo principio de la circunferencia rotatoria con LEDs de alineamiento pero ahora pasaría a ser una matriz de 3 × 48 LEDs, mientras que un tercer diseño contemplaba una simplificación debido a cuestiones geométricas al momento de realizar la rotación de la circunferencia que contenía a la matriz de LEDs, por lo que se optó por una simplificación mecánica y que a la vez pasaba a tener 2 matrices: la primera de 3 × 48 (matriz horizontal) en donde el número de LEDs activos, esto es, encendidos durante la reproducción de un *track*, consistía de una sub-matriz de 3 × 7 LEDs; la segunda matriz no era propiamente una matriz, sino más bien un arreglo de tres líneas diagonales a 45°, una línea de 48 LEDs y otras dos de 47 LEDs, una a cada lado de la línea central de 48 LEDs. Cada línea diagonal presentaría una separación mayor entre los pixeles de salida (fibras ópticas) debido al factor $\sqrt{2}$ resultante de girar un vector horizontal a una posición diagonal de 45°, y en donde también se contemplaba tener la misma sub-matriz de 3 × 7 LEDs moviéndose a lo largo de toda la diagonal. La figura 2.7 muestra los bosquejos de estos diseños previos, mientras que la figura 2.8 muestra el diseño final.

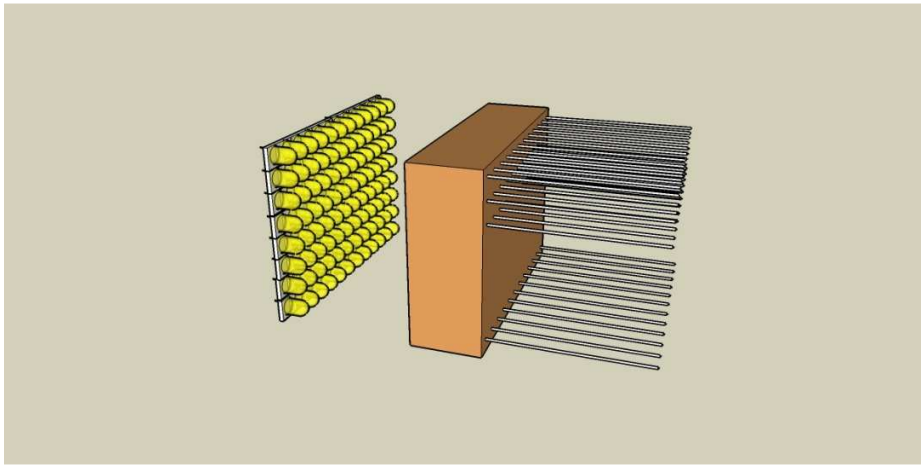


a)

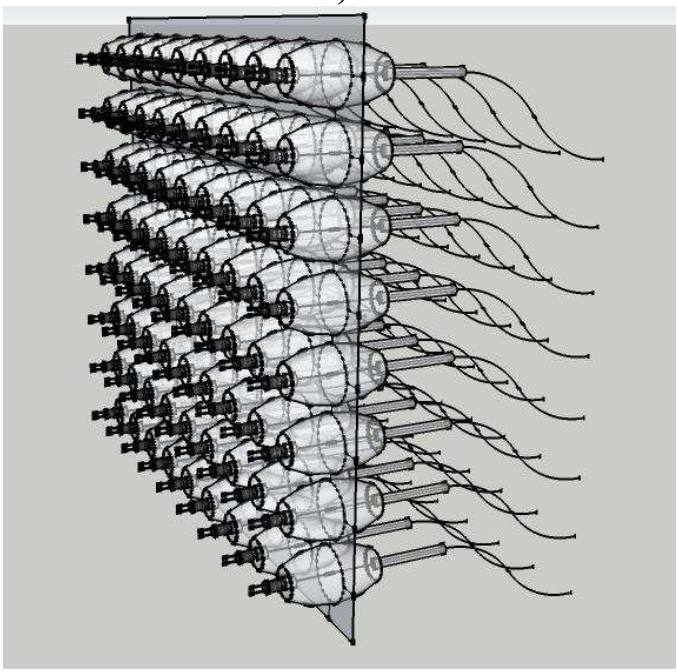


b)

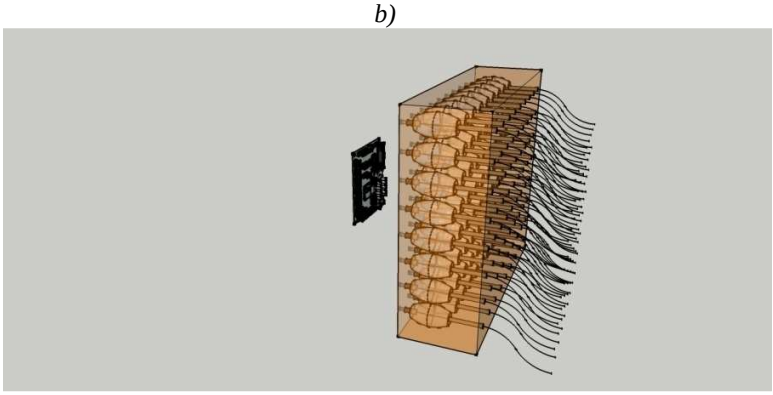
Figura 2.7. Primeros diseños del instrumento Track-Sim. a) Principio de funcionamiento y diseño conceptual del Track-Sim con sistema rotatorio. b) Sistema rotatorio conteniendo vector horizontal de LEDs dentro de la cámara de pruebas del instrumento y con PDM a probar.



a)



b)



c)

Figura 2.8. Concepto del diseño final para matriz de 48 × 48 LEDs. a) Matriz de LEDs con fibra óptica; b) Parte de la etapa de acoplamiento de LEDs con fibra óptica; c) Tarjeta de electrónica de control junto a una parte de la matriz de LEDs y fibra óptica.

2.4.3 Sistema de calibración

El sistema de calibración del instrumento Track-Sim tiene como función realizar la calibración del arreglo de LEDs dentro de la matriz con 48 × 48 fibras ópticas. El objetivo de este sistema será el de caracterizar y calibrar cada canal óptico del conjunto LED-fibra, para ajustar los parámetros requeridos y conseguir la emisión de fotones en el rango de interés. Este sistema se realiza utilizando un dispositivo MAPMT de 64 pixeles (propiamente el modelo H7546B de Hamamatsu) sobre el cual se proyecta la salida de cada pixel de matriz de la guía de luz. Para lograr esto se construye un sistema mecánico que básicamente realiza un recorrido sobre la MAPMT a fin de enfocar el o los canales de la guía a calibrar sobre los pixeles del dispositivo H7546B.

Aunado a este sistema mecánico de alineamiento, se tiene también un sistema de adquisición de datos para las señales provenientes del H7546B MAPMT, en donde se realiza el cálculo de la integración de la carga de cada pixel en que se mide la señal de salida, así como un conteo y discriminación por altura de pulso. Estos datos, posteriormente al procesamiento descrito en la etapa de hardware, son llevados a una PC mediante un dispositivo FPGA. En la PC se tiene un software que analiza los datos y determina los parámetros de ajuste para la calibración.

La figura 2.9 muestra la integración e interacción del sistema de calibración con el instrumento.

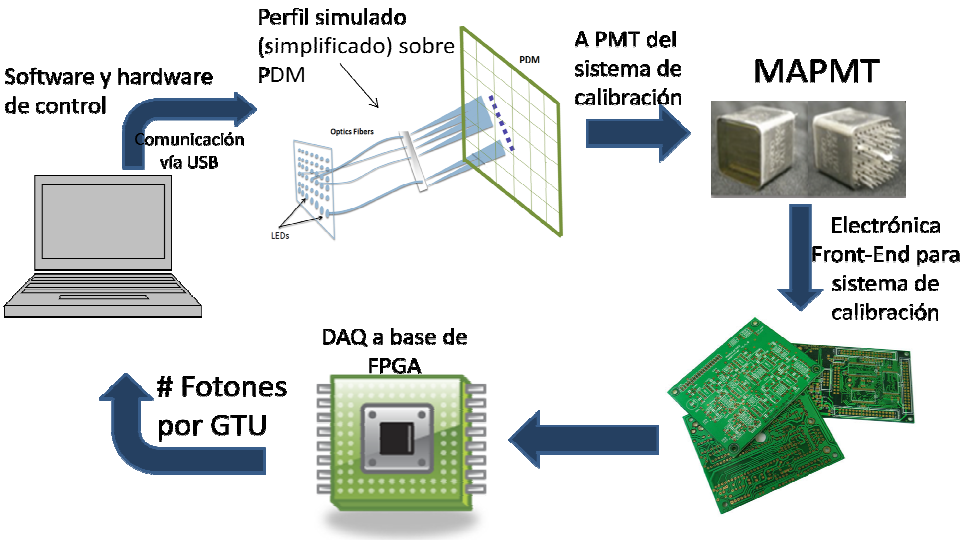


Figura 2.9. Diagrama de flujo del sistema de calibración.

2.4.4 Prototipo Track-Flat

Como primer aproximación para lo que será el instrumento Track-Sim, se llevó a cabo el diseño y construcción de un instrumento de menor tamaño pero que realice una validación de los conceptos para la realización del instrumento Track-Sim. Este primer prototipo se nombró Track-Flat, y su función principal es la de reproducir una traza de luz UV sobre 8 pixeles de un MAPMT (H7546B) a fin de llegar a tener un *track* plano. Esta traza plana realiza básicamente un recorrido de cada uno de los 8 pixeles de la PMT en forma secuencial, procurando que la emisión en cada pixel sea del mismo número de fotones, de ahí el nombre de Track-Flat. Así, se realiza una validación de la manera de operar un arreglo de 8 canales en forma pulsada para variar la cantidad de fotones a emitir dentro de 1 GTU (2.5µs). Así mismo se prueba la forma de las guías en que se albergan las fibras ópticas y los diferentes acoplamientos, a fin de verificar la forma de los canales, la emisión de cada canal y la calibración para compensar las variaciones en cada pixel.

De acuerdo con esto, se buscó la validación de las estrategias de diseño, así como la integración de las diferentes partes del dispositivo de forma funcional a fin de solucionar problemas que pudiesen presentarse en el diseño y desarrollo. Con esto, se planteó la

posibilidad de utilizar este dispositivo como ayuda en la calibración de un PDM por partes para su certificación dentro de la misión espacial UFFO[16].

Como complemento se decidió hacer más versátil al prototipo al agregarle diferentes niveles de intensidad, es decir, se tiene la opción de obtener Φ_1 fotones a la salida de los 8 pixeles, Φ_2 fotones, Φ_3 fotones, etc., en donde Φ_1, Φ_2, Φ_3,… representa una intensidad específica en la cual se emite un número de fotones constante para cada pixel. Por ejemplo, si se quiere la intensidad Φ_1 se emiten Φ_1 fotones en cada pixel (durante 1 GTU por pixel) si se elije la intensidad Φ_2 se emiten Φ_2 fotones en cada uno de los 8 pixeles, en donde el paso de un pixel al otro se realiza en tiempos de 1 GTU, y así para cada nivel de intensidad que se elija.

Para lograr esto se parte del modo de operar cada LED así como de la caracterización de los mismos. Esto se realizó utilizando como dispositivo transductor MAPMT de 64 pixeles H7546B de Hamamatsu, utilizando un solo pixel para efectos prácticos y de mayor control en la caracterización y calibración. La forma en que se varía la intensidad de un LED es al aplicar un pulso de un ancho determinado y una amplitud que alcanza el valor de encendido del dispositivo, que para este tipo de LED UV es de alrededor de 2.5V (el rango medido de forma experimental se encuentra desde alrededor de 2.5V a 3.3V). La emisión de luz en este tipo de LED (VAOL-5EUV0T4) se realiza, como ya se mencionó, al variar el ancho del pulso de voltaje aplicado a las terminales del LED, y más en concreto de la corriente, de forma que lo que se controla es el tiempo de emisión, a fin de obtener más o menos fotones controlando los anchos de pulso en que se enciende el LED, así como la intensidad de corriente. Por supuesto que la altura del pulso aplicado se trata de mantener en el mismo nivel para los diferentes LEDs a fin de evitar variaciones en la emisión LED a LED, sin embargo, hay ciertas diferencias entre cada dispositivo utilizado (desde los *switches* electrónicos que conmutan a los LEDs, hasta los mismo LEDs) que deben ser cuantificadas y, en la medida de lo posible, compensadas.

En lo que a la parte óptica se refiere, el prototipo Track-Flat integra una guía de luz en la que se contienen 8 fibras ópticas acopladas a 8 LED UV. La forma de la guía es tal que se

ajusta en su parte frontal a 8 pixeles de la MAPMT de JEM-EUSO así como a la MAPMT H7546B utilizada para la calibración. Esta guía de luz se encuentra fabricada de un plástico en color negro (ABSplus-P430), especial para la máquina de impresión 3D FORTUS 250mc en que se fabricó. El extremo en que se coloca la guía de luz con la MAPMT se encuentra pulido para tener la mejor homogeneidad posible en la luz de salida de las fibras ópticas. En la parte posterior de la guía de luz en que se encuentran los 8 LEDs, se agregó una tapa de soporte y sujeción con un diseño de recuadro que encierra a cada LED, así como una capa de material deformable, conocido como foami, y que es un material a base de etileno acetato de vinilo, el cual proporciona un mejor aislamiento óptico entre los LEDs a fin de evitar el *crosstalk* desde la emisión al inicio de los canales de la guía.

Por otra parte, la conexión de alimentación con los 8 LEDs de la guía de luz se realiza mediante un cable multiconductor blindado calibre 22 AWG, el cual se conecta a la electrónica de control mediante un conector DB9.

La generación de pulsos que conmuta a cada LED se realiza mediante un dispositivo FPGA que corre a una frecuencia de 100MHz. Los anchos de pulsos que pueden ser generados van desde los 10 ns hasta pulsos tan anchos como 1 GTU o más, sin embargo por cuestiones de requisitos así como de respuesta de la MAPMT, no se generan pulsos más allá de determinado ancho (esto será comentado en la sección 3.2 del capítulo 3). Por otra parte, los pulsos más cortos que se pueden alcanzar con los *switches* que conmutan a los LEDs van tan bajos como 12 ns (caso de los buffers *Schmitt Trigger*) y hasta 5 ns para el caso de los MOSFET 2N7002, sin embargo por cuestiones de oscilaciones y distorsiones en la respuesta de estos dispositivos (a pulsos cortos la forma del pulso oscila y se distorsiona grandemente ocasionando que la medición sea muy difícil de realizar al conectarse un LED) se optó por utilizar pulsos mayores, así como obtener un mejor control en la emisión (como se verá en la sección 3.2). La figura 2.10 muestra las dos partes principales del prototipo Track-Flat. En el siguiente capítulo se abundará más sobre la forma en que se controla el encendido de cada LED.

a) *b)*

Figura 2.10. Prototipo Track-Flat. a) Sección de control y b) guía de luz.

Capítulo 3

Diseño de la arquitectura y hardware de control del instrumento Track-Sim

Resignarse es una cobardía, es el sentimiento que justifica el abandono de aquello por lo cual vale la pena luchar, es, de alguna manera, una indignidad. La aceptación es el respeto por la voluntad de otro, sea éste un ser humano o el destino mismo. No nace del miedo como la resignación, sino que es más bien un fruto.

E. Sabato, "La resistencia".

3.1 DISEÑO BASE DEL HARDWARE PARA GENERACIÓN DE PULSOS

En esta sección se presenta la manera en que se realiza el encendido de LEDs y la variación de intensidad de los mismos, mediante el prototipo Track-Flat. Así mismo se presenta el diseño de la arquitectura del hardware de control, simulaciones y la implementación de validación para 8 canales. Todo, desde la parte de procesos, etapas lógico-digitales y analógicas, se comentará a detalle haciendo énfasis en las etapas clave, planteando la problemática, dificultades y la solución de forma viable y práctica.

Concepto general del diseño

Para conocer la respuesta a trazas realistas de partículas a lo largo del sistema en la superficie focal de JEM-EUSO, se debe reproducir un perfil transformado en foto-electrones como el mostrado en la figura 3.1. Como se debe reproducir el flujo en fotones (eficiencias cuánticas y de transmisión a través del sistema deben ser incluidas), por generalidad en cuanto a las diferentes energías y geometrías posibles, el rango dinámico en el eje vertical debe ser extendido al intervalo de 0-300 fotones por GTU (Gate-Time-Unit) de 2.5 μseg de duración. La extensión espacial sobre la superficie focal debe variar en la dirección longitudinal (dirección temporal) entre 0 y ~30 cm, mientras que perpendicularmente debe ser de ~20 mm. El sistema debe ser lo suficientemente versátil como para reproducir trazas de diferente geometría y energía, por ejemplo., variar la longitud de la traza manteniendo o no el número de GTUs constante. Por otra parte, la luz debe ser emitida entre 300 y 400 nm, idealmente, intervalo que incluye la fluorescencia del N_2.

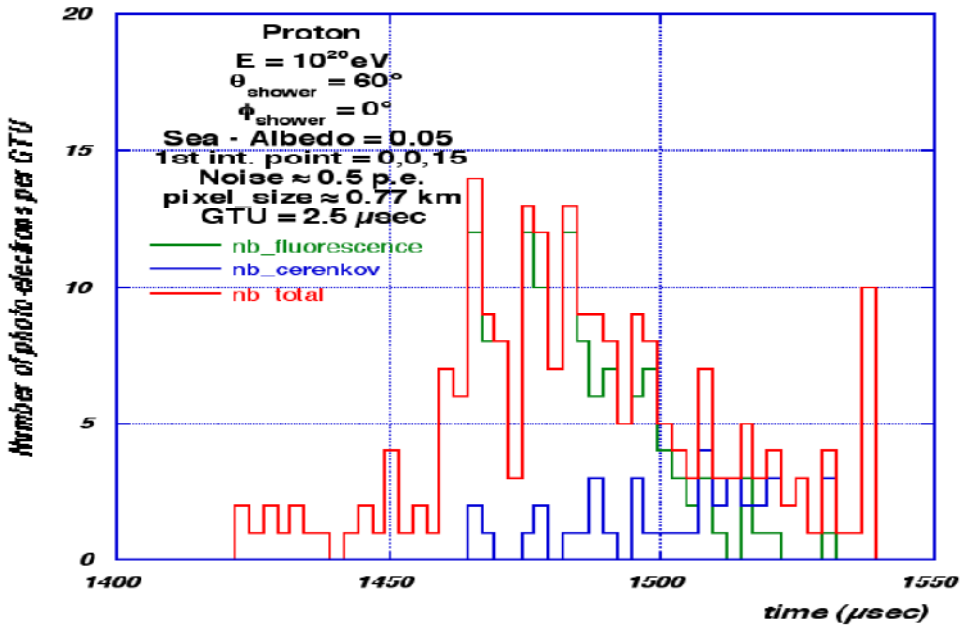


Figura 3.1. Perfil simulado de una traza sobre la superficie focal en foto-electrones.

De lo anterior se puede decir que el principal objetivo del instrumento Track-Sim es el de reproducir patrones de luz de intensidad variable, en diferentes geometrías sobre un plano *xy* para dimensiones de ~20 × 20 cm, con escalas de reproducción espacio-temporales de 2.5µs para la variación de intensidad. De esta manera, partiendo del objetivo general anterior, se plantea primeramente la realización, en forma controlada, de la emisión de luz con una fuente ultravioleta en el rango espectral de interés y dentro de los parámetros de intensidad requeridos. Las posibles opciones para lograr esto, considerando requisitos fundamentales como sencillez, robustez, versatilidad y bajo costo, son, en principio, las siguientes:

i. Sistema basado en MEMS.
ii. Sistema basado en matriz de LEDs.

MEMS

En este caso, un haz de luz sería redireccionado angularmente por MEMs. La fuente de luz podrían ser LEDs, o una lámpara UV. Las variaciones en el flujo podrían ser reproducidas variando la intensidad de la fuente o con un polarizador de rápido tiempo de reacción. Sin embrago, para que esta estrategia sea viable en el corto tiempo disponible, los MEMs

deberían venir ya en un sistema que simplifique considerablemente su implementación. Un ejemplo posible sería un proyector digital, del cual ya se pueda usar su sistema de proyección (MEMs y electrónica de *front-end*), así como protocolos de comunicación y parte del montaje mecánico. No obstante, al presente no es claro si el tiempo de reacción necesario (escalas de nanosegundos a microsegundos) puede ser alcanzado en estos dispositivos comerciales.

Matriz de LEDs

En este caso, se emplea un arreglo rectangular de LEDs, acoplados a fibras ópticas para formar una fuente de luz de dimensiones transversales más reducidas. Esta matriz, con una electrónica adecuada y programable, sería directamente la fuente de luz variable en intensidad y distribución espacial. Se debería adicionar al sistema emisor un sistema óptico adecuado para focalización y divergencia espacial del haz. Una traza típica puede llegar a tener del orden de los 100 GTUs o más en cualquier dirección del plano *xy* sobre la superficie focal, y en este caso, sobre un PDM. Esto involucraría tener una matriz cuadrada de alrededor de 2300 LEDs, por lo que se plantea como una solución un arreglo matricial de fibras ópticas originadas en los LEDs y terminadas de manera que se ajusten a los pixeles de un PDM. El montaje de fibras sería fijo, en cuyo caso el Track-Sim se simplifica en cuanto a cuestiones mecánicas y solo se presta cuidado en el control de los LEDs, así como los acoplamientos LED-fibra y salida de fibras a pixeles de PDM.

De estas dos opciones, sin lugar a dudas la que mejor se ajusta de manera práctica y técnica es la de una matriz de LEDs, ya que el manejo de MEMS utilizando dispositivos comerciales adecuados queda fuera de los requisitos fundamentales previamente mencionados (sencillez, robustez, versatilidad y bajo costo) y el diseño de dispositivos MEMS específicos para esta aplicación se aleja de los objetivos del presente trabajo, sin mencionar las dificultades técnicas y de infraestructura que esto representa.

Una vez establecida la selección de la opción a utilizar, se presenta a continuación el concepto básico para un diseño mediante una matriz de LEDs UV.

Existen diferentes maneras de variar la intensidad lumínica de un dispositivo LED, sin embargo, primeramente se revisará la forma en que se encuentra constituido un diodo LED y la manera en que opera, para posteriormente, considerando las principales características de los LEDs, se presenten las opciones mediante las cuales es posible operar a estos dispositivos, a fin de variar su intensidad a manera controlada. Para esto, las relaciones aquí presentadas, se harán bajo consideraciones ideales, es decir, se presenta un modelo con tendencia ideal de la unión P-N para diodos, y en específico dispositivos LEDs, pues dependiendo del tipo de material (y por ende del tipo de LED), el modelo sufre ciertas modificaciones, y dado que el tema central del presente trabajo no lo representan los dispositivos LEDs, lo que a continuación se presenta son únicamente los aspectos principales que permitan un entendimiento y sirvan de herramienta para el desarrollo central que es el control de la emisión de una fuente UV.

LEDs

Un LED es un diodo semiconductor que emite luz casi monocromática cuando se polariza en forma directa permitiendo que los electrones y los huecos en el dispositivo se recombinen y se libere energía en forma de fotones en un proceso denominado electroluminiscencia.

Este proceso de emisión de luz en un LED se presenta cuando un electrón pasa de la banda de conducción a la banda de valencia, perdiendo energía en el proceso en forma de un fotón. La longitud de onda del fotón emitido depende del material, y más en concreto, de la diferencia de energía entre la banda de conducción y la banda de valencia (banda prohibida).[8] El que esta energía perdida se manifieste de forma radiativa como un fotón

[8] En cualquier átomo se tienen niveles o bandas de energía, en cada una de las cuales los electrones que ahí se ubican poseen diferentes niveles de energía. En niveles muy cercanos al núcleo los electrones tienen una fuerza de atracción mayor a éste, mientras que en los niveles más alejados esta fuerza decrece permitiendo que los electrones se intercambien para combinarse con otros átomos, proceso que es de vital importancia en materiales semiconductores.

desprendido o como otra forma de energía (en forma vibratoria o calor, por ejemplo) va a depender principalmente del tipo de material semiconductor.

La banda de valencia es el nivel en donde se realizan las combinaciones de electrones, formando iones o incluso moléculas cuando se comparten con varios átomos, mientras que la banda de conducción representa el intervalo de energía electrónicas, ubicada por encima de la banda de valencia y que permite a los electrones ser acelerados ante la presencia de un campo eléctrico externo, creando de esta manera corrientes eléctricas.[9] La banda prohibida es la diferencia de energía entre la banda de valencia y la banda de conducción, representando la energía necesaria (E_g) para que un electrón pase de la banda de valencia a la banda de conducción. La figura 3.2 muestra las estructuras en que se encuentras las bandas de energía de valencia, conducción y prohibida.

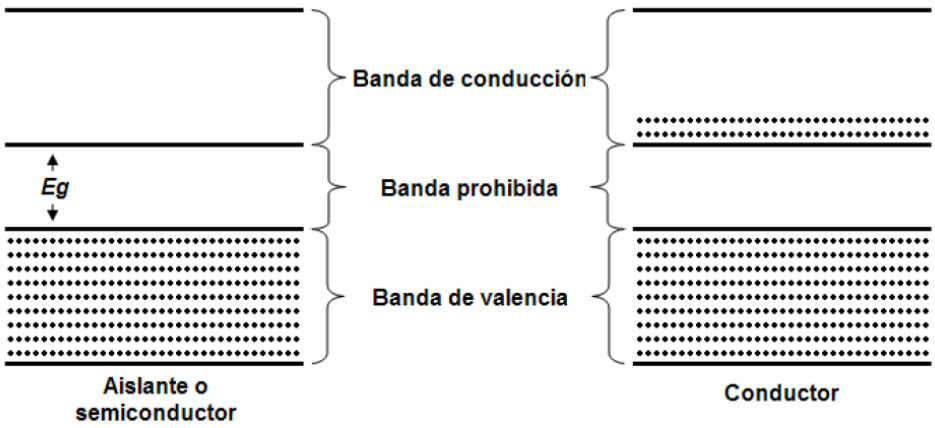


Figura 3.2. Bandas de energía en un material.

Al proceso en que los electrones pasan desde el estado energético correspondiente a la banda de conducción, a un hueco en la banda de valencia liberando energía se le denomina recombinación. La razón de recombinación se ve afectada por procesos de recombinación tanto radiativos como no radiativos, los cuales son los dos mecanismos básicos de recombinación en semiconductores. En un evento de recombinación radiativo, un fotón con

[9] En un semiconductor, los electrones pueden alcanzar esta banda cuando reciben suficiente energía, durante la polarización del dispositivo y/o por excitación térmica.

energía igual a la de la banda prohibida del semiconductor es emitido, como se muestra en la figura 3.3. En lo que respecta a un proceso no radiativo, la energía del electrón se convierte en energía vibratoria en el mallado de átomos (fonones). De esta forma, la energía de los electrones se convierte en calor, algo que no es deseable en dispositivos LED. Los procesos radiativos de alto nivel de excitación, de bajo nivel de excitación, así como procesos no radativos debidos a recombinación en superficies, recombinación Auger y defectos en la estructura cristalina, modifican la razón de recombinación [17], lo que resulta importante al momento de determinar la velocidad de encendido del dispositivo, ya que ésta se encuentra fuertemente ligada al proceso de recombinación.[10]

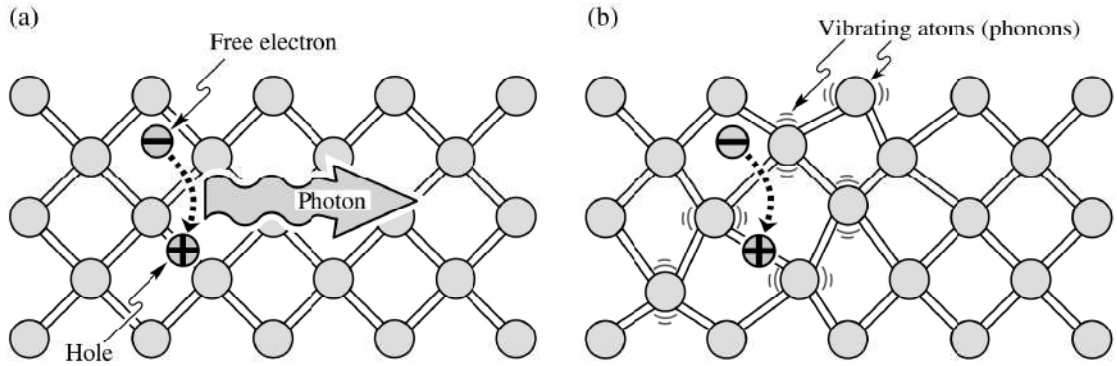


Figura 3.3. a) Recombinación radiativa de un par electrón-hueco acompañada de la emisión de un fotón con energía de $h\nu \approx Eg$. b) En eventos de recombinación no radiativa, la energía liberada durante la recombinación de pares electrón-hueco es convertida en fonones. Imagen tomada de [17].

Por otra parte, la eficiencia cuántica interna proporciona la relación de la cantidad de luz emitida dentro del semiconductor con respecto al número de carga que se recombina. Es importante señalar que no todos los fotones emitidos internamente logran salir del semiconductor debido a situaciones de escape de la luz y mecanismos de reabsorción en el sustrato entre otros.

[10] Otra consideración que afecta el proceso de emisión de luz en LEDs lo determina la pureza química del semiconductor, en donde se vuelve muy complicado fabricar materiales con niveles de impurezas menores al rango de partes por billón (10^9), y en donde algunos elementos pueden tener niveles profundos de impurezas y reducir de esta manera la eficiencia lumínica, de modo que se define una eficiencia cuántica en el semiconductor (eficiencia cuántica interna) debida a procesos radiativos y no radiativos.

Unión P-N en LEDs

Como cualquier diodo semiconductor, la base de un LED la constituye la unión de materiales extrínsecos P y N.[11] En el momento en el que los dos materiales se unen, los electrones y los huecos se combinaran en la región de unión, y como consecuencia, se creará una región con carencia de portadores. Esta región de iones positivos y negativos se denomina región de agotamiento o de transición, debido a la disminución de portadores en ella.

Bajo consideraciones en que todos los dopantes se encuentran completamente ionizados, la concentración de electrones libres (donadores) es $n = N_D$ y la concentración de huecos libres (aceptores) es $p = N_A$. En ausencia de portadores en la región de agotamiento, la única carga en esta región proviene de donadores y aceptores ionizados, los cuales forman una región de carga en donde se produce un potencial denominado voltaje de difusión V_D dado por

$$V_D = \frac{kT}{e} \ln \frac{N_A N_D}{n_i^2}$$

Donde n_i es la concentración intrínseca del material, y N_A y N_D las concentraciones de aceptores (huecos) y donadores (electrones) respectivamente. Este voltaje de difusión representa la barrera que los portadores libres deben superar para alcanzar las regiones neutras de conductividad opuesta (p ó n según sea el caso).

[11] Un material semiconductor que no tiene ningún tipo de impureza se dice que es un semiconductor intrínseco. Éste posee una banda prohibida lo suficientemente grande de tal forma que los electrones no puedan pasar de la banda de valencia a la banda de conducción, comportándose como un material aislante. Los más comunes de éste tipo de semiconductores los representan el silicio y el germanio. Si a un semiconductor intrínseco se le añade un pequeño porcentaje de "impurezas" (elementos trivalentes o pentavalentes), el semiconductor se denomina extrínseco, y se dice que está dopado, generándose semiconductores tipo N y semiconductores tipo P.

La ecuación para un diodo semiconductor con sección transversal *A* desarrollada por Shockley es

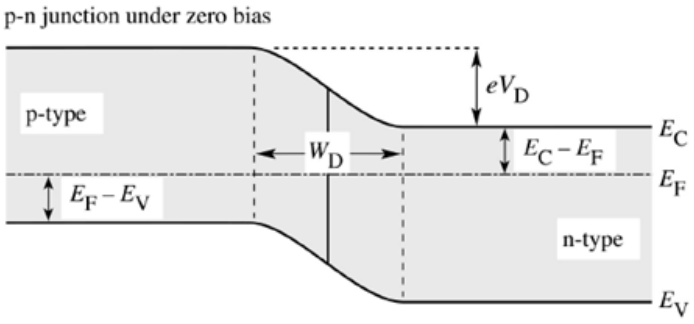
$$I = eA\left(\sqrt{\frac{D_p}{\tau_p}}\frac{n_i^2}{N_D} + \sqrt{\frac{D_n}{\tau_n}}\frac{n_i^2}{N_A}\right)\left(e^{eV/kT} - 1\right)$$

Donde $D_{n,p}$ y $\tau_{n,p}$ son las constantes de difusión de electrones y huecos y la vida de portadores minoritarios de electrones y huecos, respectivamente [17]. Esta ecuación puede reescribirse como

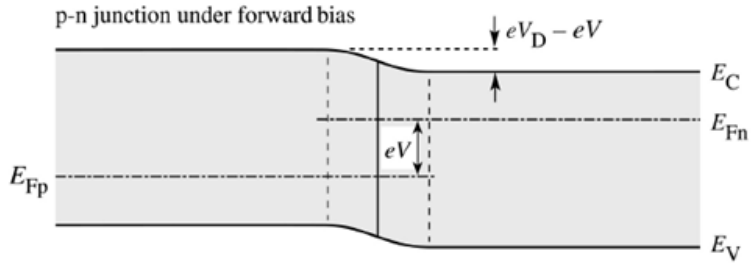
$$I = I_S\left(e^{eV/kT} - 1\right) \quad \text{siendo} \quad I_S = eA\left(\sqrt{\frac{D_p}{\tau_p}}\frac{n_i^2}{N_D} + \sqrt{\frac{D_n}{\tau_n}}\frac{n_i^2}{N_A}\right)\left(e^{eV/kT} - 1\right)$$

La cual representa la clásica ecuación de un diodo semiconductor, y que puede simplificarse para condiciones de polarización directa a sabiendas de que el voltaje en el diodo V>>kT/e (ó V_D >> kT/e), por lo que exp(eV/kT - 1)≈ exp(eV/kT). El voltaje en el cual la corriente se incrementa notablemente (cuando el voltaje del diodo se aproxima al voltaje de difusión) se denomina voltaje de umbral, y vale $V_{th} \approx V_D$.

La figura 3.4 muestra lo anteriormente dicho para una unión P-N.



a)



b)

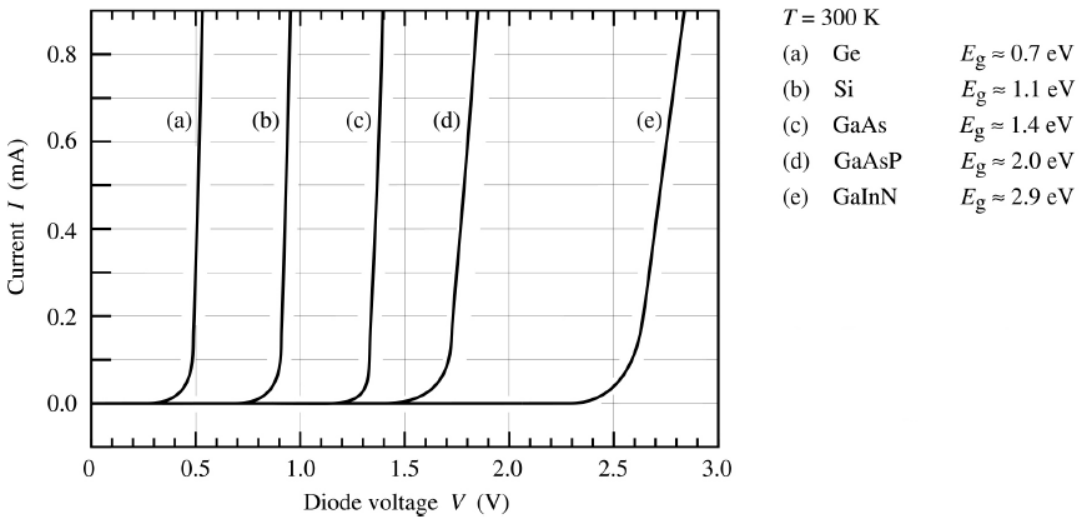
Figura 3.4. Unión P-N en (a) sin voltaje de polarización y (b) polarización directa, en donde los portadores minoritarios se difunden dentro de la región neutral donde se recombinan, y se reduce el ancho de la capa de la región de agotamiento W_D. Imagen tomada de [17].

Una buena aproximación para determinar el voltaje de difusión (y por consiguiente el voltaje de umbral) es dividir la energía para pasar la banda prohibida (pasar de la banda de valencia a la banda de conducción) entre el valor de la carga elemental[12]

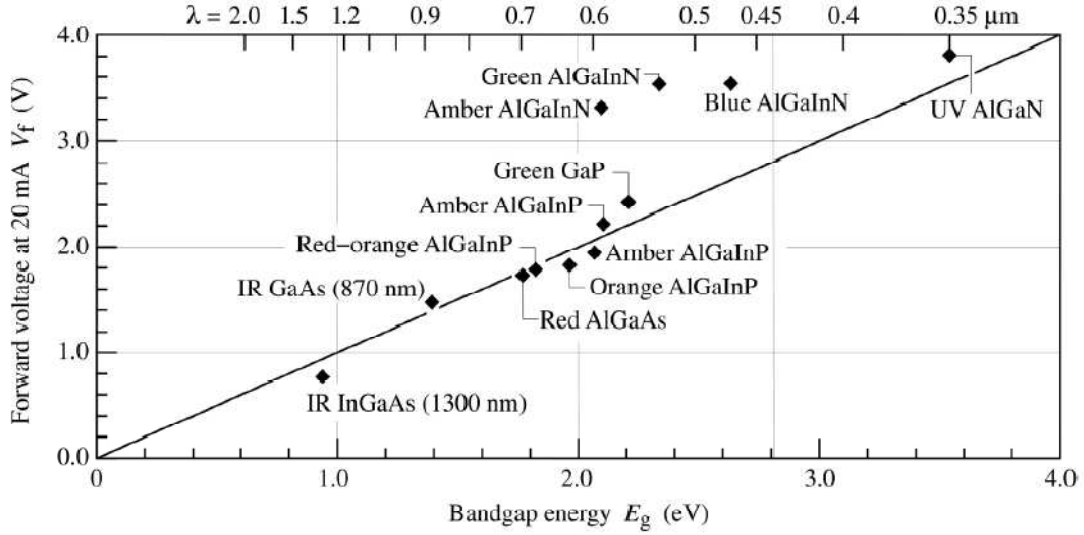
$$V_{th} \approx V_D \approx \frac{E_g}{e}$$

La figura 3.5 presenta algunas de las curvas características para diferentes LEDs de diferentes materiales con sus valores típicos de encendido.

[12] Aún cuando esta es una buena aproximación, no considera efectos como resistencias parásitas (resistencia serie y paralelo del dispositivo), pérdidas de inyección no adibáticas, dispersión por emisión de fonones, entre otros, los cuales pueden ser significativos bajo diferentes circunstancias y dependiendo del dispositivo en cuestión, así como del modo de operarlo [17].



a)



b)

Figura 3.5. a) Características corriente – voltaje a temperatura ambiente para uniones P-N de diferentes materiales y b) Voltajes de polarización directa (encendido) contra energía de la banda prohibida (bandgap) para LEDs fabricados con diferentes materiales. Imágenes tomadas de [17].

Emisión de luz en LEDs

Cuando un electrón pasa de la banda de valencia a la banda de conducción y deja un hueco en la banda de valencia se crea un par electrón – hueco. La energía necesaria para producir esto (E_g) puede ser suministrada por un incremento en la temperatura, un fotón, un electrón, u otra partícula energética que excite el sistema. El electrón en la banda de conducción que pasa a un estado "excitado" tenderá a regresar a su estado de equilibrio al encontrar un

hueco (recombinación), pasando en el proceso de un estado de mayor energía a uno de menor en la banda de valencia, emitiendo energía en el proceso igual a $h\nu \approx E_g$.

Para un diodo ideal, cada electrón inyectado en la región activa del dispositivo generará un fotón. Por conservación de la energía, la energía de cada electrón inyectado será igual a la energía del fotón generado, de forma que el voltaje aplicado al LED multiplicado por el valor de la carga elemental es igual a la energía del fotón emitido ($eV = h\nu$). Existen sin embargo diversos efectos que provocan cambios en el voltaje ideal del diodo provocando diferencias en la aproximación anterior.

Durante el proceso de recombinación radiativa, la emisión de luz se da mediante un mecanismo de emisión espontánea, el cual se encuentra dentro de los mecanismos desarrollados por Einstein dentro de la teoría de transiciones ópticas.[13] La figura 3.6 muestra esquemáticamente estos procesos.

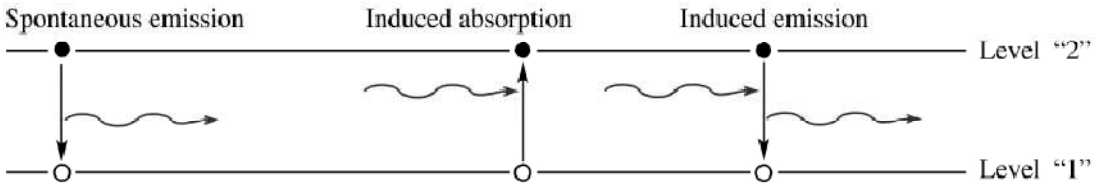


Figura 3.6. Representación de emisión espontánea (izquierda), absorción inducida o estimulada (centro) y emisión estimulada o inducida (derecha) en un modelo atómico de dos niveles de energía. Imagen tomada de [17].

De manera ideal, se emite un fotón por cada electrón de corriente que se inyecta en la región activa de un LED, por lo que la eficiencia cuántica correspondería a la unidad. Esta eficiencia cuántica interna se define como

[13] La emisión espontánea ocurre sin un estímulo externo, produciendo un fotón. De esta forma, la razón de emisión inducida es proporcional a la densidad de fotones o densidad de radiación. Por su parte, la emisión estimulada se presenta cuando una excitación provoca la generación de fotones. Así mismo, Einstein mostró que la emisión estimulada y la absorción estimulada son procesos complementarios [17].

$$\eta_{\text{int}} = \frac{\#\ de\ \ fotones\ emitidos\ desde\ la\ \ región\ activa\ \ por\ \ segundo}{\#\ \ de\ electrones\ inyectados\ dentro\ del\ \ LED\ \ por\ \ segundo} = \frac{P_{\text{int}}/(h\nu)}{I/e}$$

Donde P_{int} es la potencia óptica emitida desde la región activa e I es la corriente inyectada [17].

En un LED ideal todos los fotones emitidos desde la región activa salen al espacio libre, lo que representa una eficiencia de extracción unitaria. Sin embargo, en un LED real, no toda la potencia emitida desde la región activa sale del dispositivo, por lo que algunos fotones nunca abandonan el semiconductor y son atrapados. Esto se debe a diferentes mecanismos como la reabsorción en el sustrato (asumiendo que el sustrato absorbe en la misma longitud de onda), incidencia en la superficie de los contactos metálicos (ánodo y cátodo) y reflexión interna total. Por esto, se define una eficiencia de extracción como

$$\eta_{extracción} = \frac{\#\ \ de\ \ fotones\ emitidos\ en\ \ el\ \ espacio\ libre\ \ por\ \ segundo}{\#\ \ de\ \ fotones\ emitidos\ desde\ la\ \ región\ activa\ \ por\ \ segundo} = \frac{P/(h\nu)}{P_{\text{int}}((h\nu)}$$

Donde P hace referencia a la potencia óptica emitida en el espacio libre [17].

De lo anterior, la eficiencia cuántica externa se define [17] como

$$\eta_{ext} = \frac{\#\ \ de\ \ fotones\ emitidos\ en\ \ el\ \ espacio\ libre\ \ por\ \ segundo}{\#\ \ de\ electrones\ inyectados\ dentro\ del\ \ LED\ \ por\ \ segundo} = \frac{P/(h\nu)}{I/e} = \eta_{\text{int}}\eta_{extracción}$$

La cual determina la relación del número de fotones que salen del dispositivo con respecto al número de cargas inyectadas en el mismo.

Otra cantidad de utilidad en estos dispositivos es la eficiencia de potencia, la cual se define como

$$\eta_{pot} = \frac{P}{IV}$$

Donde el producto *IV* es la potencia eléctrica y *P* la potencia óptica emitida en el espacio libre.

Capacitancia de transición y difusión

Como en la mayoría de los dispositivos semiconductores, existen efectos capacitivos los cuales pueden ser despreciados en operación a bajas frecuencias debido a que la reactancia capacitiva es alta a frecuencias bajas ($X_C = 1/2\pi fC$), sin embargo, a frecuencias altas, esto no puede ignorarse debido a que el valor de X_C se volverá lo suficientemente pequeña debido al valor alto de *f*, lo que puede introducir una trayectoria de corto circuito a través del dispositivo.

En el diodo semiconductor p-n existen dos efectos capacitivos que deben considerarse, sobre todo hacia frecuencias de operación altas. Ambos efectos de capacitancia se presentan tanto para la región de polarización directa como para la región de polarización inversa, pero dado que una de ellas siempre es mucho mayor a la otra en cada región, se consideran los efectos de sólo una para cada región.

En la región de polarización inversa se presenta en mayor medida la capacitancia de transición (C_T) o de región de agotamiento, mientras que en la región de polarización directa se tiene la capacitancia de difusión (C_D) o de almacenamiento.

En la región de polarización inversa existe una región de agotamiento (libre de potadores) que en esencia se comporta como aislante entre las capas de cargas opuestas. Cuando se incrementa el voltaje de polarización inversa, la capacitancia de transición disminuirá (el ancho *d* de la región de agotamiento se vuelve mayor) como se observa en la figura 3.7,

recordando que la capacitancia para un capacitor de placas paralelas de área *A*, separadas una distancia *d* se define por $C = \varepsilon A/d$, donde ε es la permitividad del dieléctrico.[14]

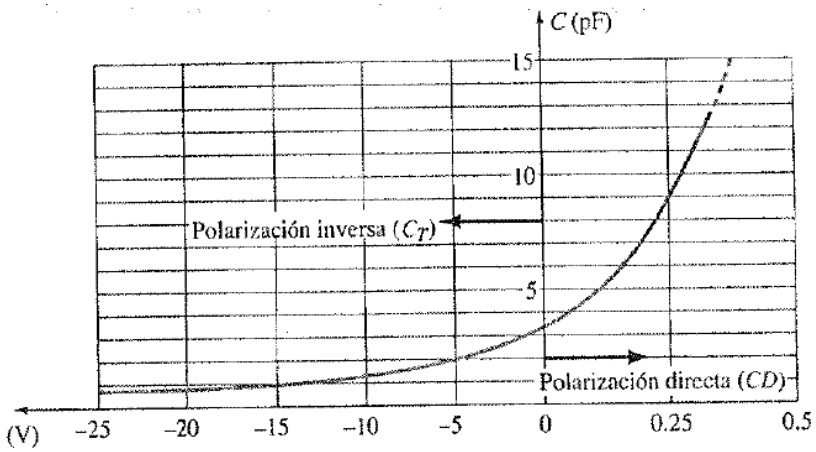


Figura 3.7. Representación de los efectos de capacitancia de transición y difusión como función del voltaje aplicado en un diodo de silicio. Imagen tomada de [18].

El efecto de capacitancia de transición también se presenta en la región de polarización directa pero es mucho menor y despreciable junto al efecto de capacitancia de difusión, en donde a niveles crecientes de corriente se presentarán niveles crecientes de capacitancia de difusión, que depende de la velocidad a la cual se inyecte la carga en las regiones cercanas fuera de la región de agotamiento [18]. Por otra parte, cuando se incrementa la capacitancia de difusión al incrementar la corriente, se reduce el valor de resistencia asociada, por lo que la constante de tiempo ($\tau = RC$) resultante tiende a mantenerse constante. Los efectos capacitivos de transición y difusión se esquematizan con un capacitor en paralelo con el diodo semiconductor como se aprecia en la figura 3.8.

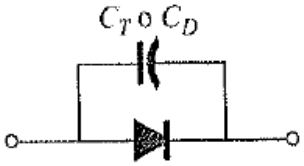


Figura 3.8. Circuito equivalente para efectos capacitivos en un diodo semiconductor. Imagen tomada de [18].

[14] El hecho de que la capacitancia dependa del potencial de polarización inversa representa una clase especial de diodos semiconductores que se aplican en diferentes sistemas electrónicos (diodos de capacitancia variable o varicap).

Tiempo de recuperación inverso

Durante el estado de polarización directa una gran cantidad de electrones se mueve del material de tipo *n* al material de tipo *p* y una cantidad de huecos pasa del material tipo *p* al material tipo *n*, lo cual es un requisito para que exista la conducción. Los electrones en el material tipo p así como los huecos que se difunden en el material tipo n establecen un gran número de portadores minoritarios en cada material. Si el voltaje aplicado en las terminales del dispositivo se invierte con objeto de establecer una polarización inversa, se observaría, de manera ideal, que el diodo pasaría de un estado conductivo a uno no conductivo, sin embargo, debido a la cantidad de portadores minoritarios en cada material, la corriente a través del diodo se invierte y se mantiene en ese nivel durante cierto intervalo de tiempo, denominado tiempo de almacenamiento (t_s), y que es el tiempo que requieren los portadores minoritarios para regresar a su estado de portadores mayoritarios en el material opuesto. Este comportamiento puede apreciarse en la figura 3.9.

El diodo permanecerá en ese estado de circuito cerrado con una corriente en inversa ($I_{inversa}$) que estará determinada por los parámetros de la red. Posterior al tiempo t_s la corriente reducirá su nivel hasta llegar al estado de no conducción, tiempo que se denomina como tiempo de transición (t_t) con lo que el tiempo de recuperación inverso será la suma de t_s y t_t ($t_{rr} = t_s + t_t$) y que es un valor de especial interés para aplicaciones de conmutación rápida.

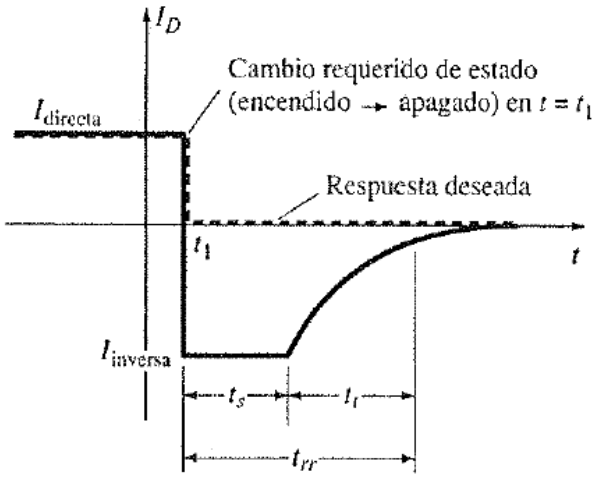


Figura 3.9. Definición del tiempo de recuperación inverso en un diodo semiconductor. Imagen tomada de [18].

Características de modulación en LEDs

Como primera aproximación al comportamiento de un LED en aplicaciones de modulación, se tratará a este tipo de dispositivos mediante aproximaciones lineales, las cuales desde luego no son el reflejo fiel de lo que ocurre en situaciones reales, pero representan un buen acercamiento para entender su comportamiento.

Una característica muy importante al momento de utilizar un LED en aplicaciones de conmutación rápida son el tiempo de levantamiento y de caída del dispositivo. Para comprender estas importantes características de un LED se recurre a un circuito RC excitado con una función escalón, en donde es conocida la respuesta en que la salida de voltaje se incrementa con

$$V_{out}(t) = V_0\left(1 - e^{-\frac{t}{\tau_1}}\right)$$

Con una constante de tiempo $\tau_1 = RC$. Cuando el voltaje aplicado regresa a cero, la salida de voltaje decae como

$$V_{out}(t) = V_0 e^{-\frac{t}{\tau_2}}$$

De donde se sabe que para un circuito *RC* simple $\tau_1 = \tau_2$. Los tiempos de levantamiento y decaimiento se definen como el tiempo en que el nivel de voltaje crece del 10% del nivel de voltaje final al 90% del mismo, para el tiempo de levantamiento, y del 90% al 10% para el tiempo de decaimiento. Esto se aprecia en la figura 3.10.

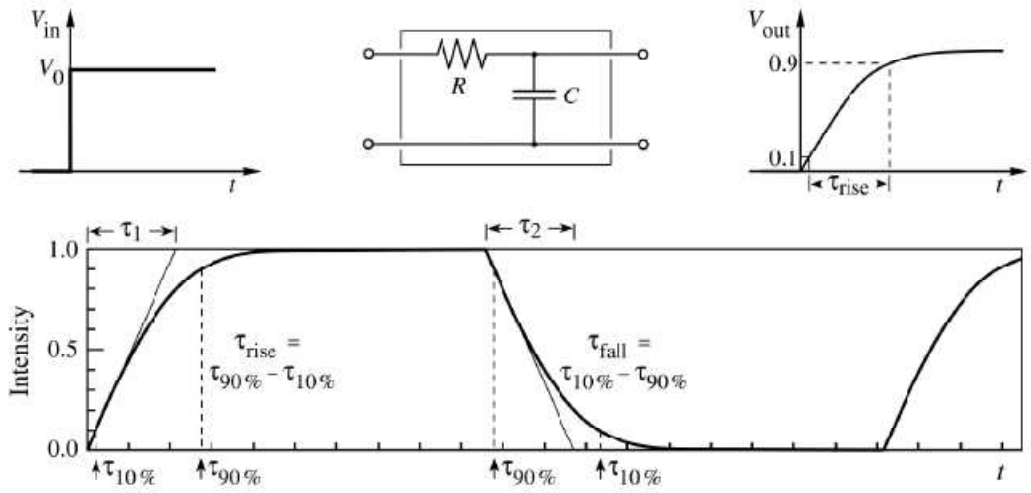


Figura 3.10. Circuito RC con excitación de un escalón y sus tiempos de levantamiento y decaimiento. Imagen tomada de [17].

La función de transferencia para este circuito es

$$H(i\omega) = \frac{1}{1 + i\omega\tau}$$

Si se utiliza el modelo de un circuito *RC* para aproximar la respuesta de un LED, con una función escalón como fuente de excitación, la potencia óptica de salida del LED y su función de transferencia se definirán como

$$P_{out}(t) = P_0\left(1 - e^{-\frac{t}{\tau_1}}\right) \qquad H^2_{LED}(i\omega) = \frac{1}{1 + i\omega\tau}$$

Esto se puede esquematizar como se muestra en la figura 3.11.

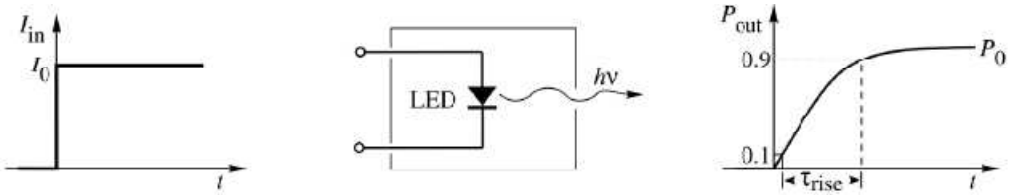


Figura 3.11. Potencia óptica de salida como función del tiempo para un LED con tiempo de levantamiento τ_r.. Imagen tomada de [17].

Las consideraciones anteriores son una buena aproximación para determinar los tiempos de levantamiento y decaimiento, sobre todo cuando se desconocen los parámetros importantes que tienen gran influencia sobre la respuesta no lineal de un LED, como lo son las resistencias serie y paralelo, la capacitancia de transición y de difusión, procesos de recombinación no radiativa entre otros.

Cuando se modula un LED, es importante tener en cuenta la capacitancia *C* intrínseca del mismo (transición y difusión), la cual como ya se ha mencionado, depende de la corriente y de la región de polarización en que se opere, lo que principalmente repercute en los tiempos de levantamiento y decaimiento, en donde la capacitancia del LED *C*, así como la resistencias del mismo y la del circuito que lo opera, determinarán el valor de τ. Si se reduce la capacitancia del diodo LED, se incrementa el ancho de banda, Por ejemplo, como se menciona en E. Freud Schumbert, Hunt N. E. J., Malik R. J., Micovic M., and Miller D. L. "Temperature and modulation characteristics of resonant-cavity light-emitting diodes" *IEEE J. Lightwave Technology* 14, 1721 (1996), en donde para LEDs de comunicaciones, la corriente que se inyecta en la región activa es pequeña, de modo que el tiempo de vida espontáneo de recombinación es quien limita la frecuencia de modulación del dispositivo.

Para LEDs de alta capacitancia el principal problema lo representa la capacitancia de difusión, la cual resulta difícil de reducir, sin embargo puede lograrse mediante la introducción de defectos que actúen como "*luminescence killers*" (E. Freud Schumbert *et al*., 1996) para poder operar en el orden de GHz, sin embargo la salida de luz se ve fuertemente disminuida, mientras que para LEDs de capacitancia pequeña, el tiempo de levantamiento y de decaimiento se encuentran mayormente limitados por el tiempo de recombinación espontánea (E. Freud Schumbert *et al*., 1996).

Afortunadamente para el LED elegido, la capacitancia que se manifiesta no perjudica la forma de operar como se verá posteriormente.

Tecnicas de control de intensidad para LEDs

Variación de intensidad en LEDs por nivel de polarización desde la fuente de alimentación

Existen diferentes formas de variar la intensidad de un dispositivo LED, pero la mayoría realizan al final lo mismo: variar la polarización aplicada al LED. La primera de ellas que se comentará es mediante la variación de la señal de alimentación, de forma que se cambie el nivel de corriente y voltaje en el LED. La figura 3.12 muestra esquemáticamente y de manera simplificada la forma de realizar esto.

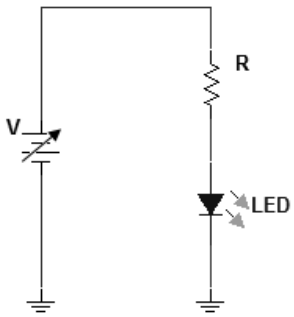


Figura 3.12. Variación de intensidad de un LED mediante la variación de la fuente de alimentación.

Esta es la manera más sencilla y básica de cambiar la intensidad de un LED, y es como más adelante se mostrará la verificación de la respuesta de la curva I-V del LED.

El problema que presenta esta solución es que requiere un número *n* de fuentes independientes variables para controlar *n* LEDs, en donde para un arreglo de algunos pocos LEDs puede ser una opción viable, sin embargo, dadas las características del arreglo a manejar, esta forma de variar la intensidad de los LEDs resulta impráctica, además de que no toma en cuenta un parámetro importante que más adelante será comentado y que se refiere al detector a utilizar (MAPMT), el cual presenta situaciones de saturación de luz para fuentes luminosas que operan de forma continua, la cual, es precisamente la forma de operar del circuito de la figura 3.12.

Pasando por alto en este momento la situación de la saturación de la PMT, la forma en que se operara al LED con una fuente variable permitiría mover el punto de operación en la curva del LED para obtener la intensidad óptica correspondiente, siendo posible además de

la variación en intensidad, el realizar el cambio correspondiente del nivel de polarización del LED entre GTUs. Esta forma de operación en intensidades diferentes se planteó dentro las etapas y diseños previos del Track-Sim que fueron comentados en la sección 2.4. La operación de calibración por intensidad operaría de la siguiente manera.

Primeramente se contempla que el movimiento de un *track* típico sobre la superficie focal, el cual se constituye en un área que abarca entre 9 y 25 pixeles (típicamente 21 pixeles conocidos de simulaciones previas como las mostradas en [12]), formados en un arreglo matricial de 3 × 7, que a su vez constituye la matriz base a formar con LEDs UV. Este arreglo en principio se mueve en una dirección específica determinada por la dirección de desarrollo del *track* y puede ocurrir para cualquier ángulo acimutal sobre la superficie focal. La figura 3.13 muestra un arreglo de 21 pixeles independientes, los cuales conforman, de manera aproximada, el *track* que se proyecta sobre la superficie focal de JEM-EUSO.

Para generar el patrón de intensidad variable de este arreglo "básico" de pixeles independientes, se parte del control por intensidad variable de una sub-matriz de 21 LEDs, acoplados a fibra óptica y controlados mediante la variación de fuentes independientes. Esta sub-matriz de 21 LEDs independientes es parte de la matriz horizontal descrita en la sección 2.4 de 3 × 48 LEDs. Propiamente no son 21 LEDs independientes, sino una señal lumínica de 21 niveles de intensidad independientes que recorren la matriz de 3 × 48 (y 3 × 68 para el caso del diseño de la matriz diagonal).

Para visualizar de manera muy sencilla el funcionamiento de la reproducción de un *track*, se realiza una analogía con el comportamiento de un anuncio publicitario de LEDs en que la propaganda se mueve de un lado a otro sobre una matriz que cubre todo el panel de reproducción del que se compone al anuncio. La única diferencia es que en el caso del Track-Sim la velocidad de reproducción es muy alta y en un rango espectral reducido dentro del cercano ultravioleta, además de tener la posibilidad de moverse en cualquier dirección (para el caso del diseño final de 48 × 48 LEDs) o en tres direcciones para el caso del primer diseño del arreglo giratorio.

Figura 3.13. Vista en 2D del arreglo que representa la sub-matriz con 21 niveles independientes de intensidad y una representación de un conglomerado de pixeles que conforman un track.

Una forma práctica de lograr lo anterior, mediante la variación del voltaje de polarización en cada LED, se realizaría transportando el patrón deseado mediante un circuito lógico combinacional, el cual permitiría realizar el recorrido de los 21 pixeles independientes por la matriz en la dirección deseada con pasos de 1 GTU (2.5µs) mientras que la intensidad de los LEDs se varía cambiando el nivel de su voltaje de polarización a través de circuitos convertidores de frecuencia a voltaje (CFV), los cuales sería controlados por señales de diferentes frecuencias generadas en un sistema a base de FPGA para el control.

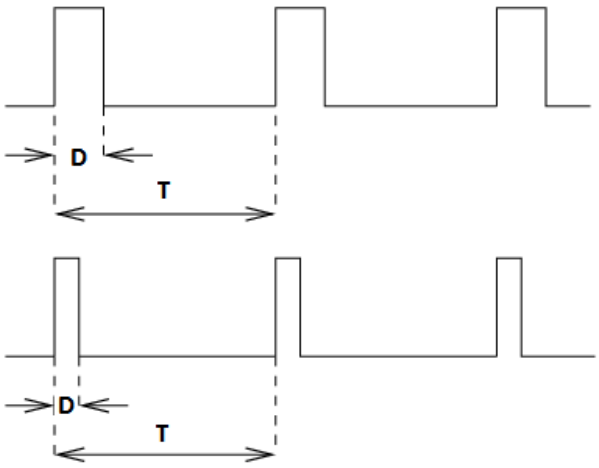
Para lograr esto se requiere de varios circuitos convertidores de frecuencia a voltaje, con un nivel de voltaje diferente a la salida de cada uno. Estos circuitos se conectarían a un arreglo de *switches* analógicos o multiplexores analógicos de alta velocidad (para realizar la conmutación entre niveles de intensidad en tiempos menores a 1 GTU, y cuya salida se conectaría directamente a cada LED a controlar. Serían necesarios tantos circuitos convertidores de frecuencia a voltaje como niveles de intensidad se requiriera; de la misma forma sería necesario generar la cantidad requerida de señales de diferente frecuencia para alimentar a cada circuito CFV. Los *switches* o multiplexores analógicos estarían controlados por las mismas señales que operan sobre el circuito combinacional.

Como se ha manifestado, para tener diversidad de niveles de intensidad se necesita de varios circuitos CFV que los produzcan, lo cual representa una primer complicación pues si se requiere de *n* niveles diferentes para cada uno de los 21 pixeles independientes, la cantidad de circuitos CFV se incrementa, volviéndose en una solución complicada e impráctica. Otro aspecto que complicó esta realización es el hecho de que la PMT presenta una saturación para determinados niveles de luz, y dado que la forma de operar del LED al

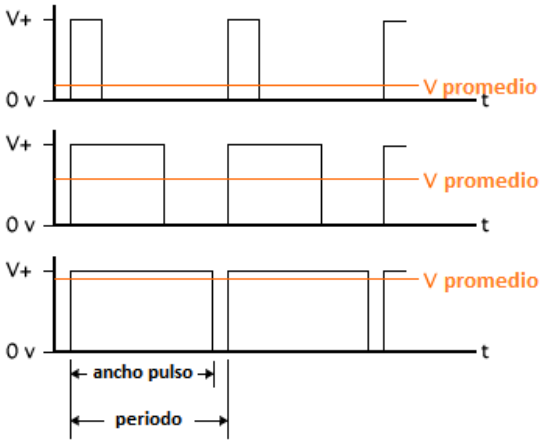
modificar su voltaje de polarización es no pulsada, esto es, se varía la corriente y voltaje en el LED pero éste se mantiene encendido durante bastante tiempo, se producen niveles de intensidad muy grandes para utilizar un dispositivo PMT. Este tipo de complicaciones volvió a esta forma de variación de intensidad en LEDs inviable para los propósitos que se pretenden.

Variación de intensidad por PWM

Esta forma de variar la intensidad de un LED es bastante conocida y es la que más se utiliza para este propósito. La manera en que se realiza una modulación PWM para intensidad de LEDs proporciona un nivel de corriente promedio en el LED, la cual es proporcional a la intensidad lumínica que se emite. La figura 3.14 esquematiza la forma en que opera una señal PWM.



a)



b)

Figura 3.14. Representación de una señal PWM para variar la intensidad de un LED. a) Definiciones del periodo T y ciclo de trabajo D. b) Variación del nivel promedio al variar el ancho de pulso ó ciclo de trabajo.

La modulación PWM es una técnica en la que el ciclo de trabajo de una señal periódica se modifica para dos fines principalmente: a) para transmitir información a través de un canal de comunicaciones ó b) para controlar la cantidad de potencia que se envía a una carga. La segunda opción es la que se aplica en la modulación de intensidad en LEDs, lo cual se realiza al controlar el valor promedio del voltaje (y corriente) que alimenta a la carga (LED) mediante el encendido y apagado de un *switch* que conecta la fuente de alimentación a la carga. Así, mientras más tiempo permanezca cerrado dicho *switch*, mayor será la potencia que se suministra a la carga. Lo anterior se realiza al controlar el ancho de pulso o ciclo de trabajo de la señal utilizada. Para el tren de pulsos típico utilizado como el que se observa en la figura 3.14 se obtiene el nivel promedio como

$$\bar{v} = \frac{1}{T}\int_0^T v(t)dt$$

Si *v(t)* es una señal que varía únicamente entre la referencia 0V y un nivel máximo V_0, la expresión anterior queda como

$$\bar{v} = \frac{1}{T}\int_0^{DT} V_0 dt = V_0 D$$

Recordando que *D* se define en el rango entre 0 y 1.

La manera de operar la variación de intensidad para el arreglo de 21 niveles independientes es mediante un circuito combinacional el cual, al igual que en el caso anterior de variación por nivel de polarización desde la fuente, llevaría a cabo el recorrido del *track* en la matriz de 3 × 48 y 3 × 68 a intervalos de 1 GTU entre pixeles, y al mismo tiempo, recibe las señales PWM y las pasa hacia los circuitos de manejo de LEDs, resultando el circuito combinacional, en una especie de decodificador – multiplexor con 204 salidas para la matriz diagonal (3 × 68) y 144 salidas para la matriz horizontal (3 × 48). Así mismo, se obtienen 21 canales PWM independientes (3 × 7) más 6 canales PWM extra para poder realizar algunas simulaciones de "background" mediante LEDs colocados estratégicamente para este fin, y 7 bits de control para direccionamiento. Toda esta arquitectura se implementaría en un dispositivo FPGA.

El mayor inconveniente de este sistema es claramente la gran cantidad de líneas de entrada/salida necesarias para cumplir con la reproducción sobre las matrices horizontal y diagonal junto a los 6 LEDs de *background* y los bits de control, lo que para el FPGA a utilizar es un problema no pequeño, pues aunque existen chips PFGA con 1000 pines o más, la utilización de tales dispositivos los vuelve costosos, la disponibilidad no es inmediata y el manejo de tal cantidad de pines y líneas de reloj, polarización y voltajes de referencia hacen que el diseño del PCB que contenga dicho chip sea todo un problema en sí a resolver. Por otra parte, la utilización de circuitos externos que hagan la función de multiplexores lleva el problema de tener múltiples pines en el FPGA y dispositivos de alta velocidad independientes en los cuales se debe evitar, en lo posible, problemas de retrasos y sincronización, claro está, partiendo del hecho de que se dispone de dichos circuitos y dispositivos para realizarlos.

Estos inconvenientes vuelven la solución de manejo de señales PWM complicada, además de que no se garantiza que dichos niveles PWM no produzcan saturación en la PMT, ya que pruebas realizadas con un LED pulsado mediante una señal PWM mostraron que para

determinados anchos de pulso, la respuesta de la PMT se muestra saturada (esto se revisará posteriormente).

Variación de intensidad mediante variación de anchos de pulso: PWV (Pulse Width Varying)

Se planteó un nuevo esquema de operación dadas las problemáticas que presentan los diseños anteriores, así como los inconvenientes de saturación que se vislumbraban en la PMT, pero principalmente debido a un rediseño en el proyecto Track-Sim. En este rediseño se mantiene la variación de intensidad de los LEDs para el régimen de emisión de N fotones dentro de 1 GTU (rango de 0 – 300 fotones/GTU) de ser posible emitiendo fotones individuales (*single photo-electron* en la PMT) o en paquetes pero siempre dentro del rango establecido. Se presenta la opción de reducir la cantidad de canales que conectan a cada LED, mediante el direccionamiento matricial para acceder al o los LED(s) que se desea encender por determinado tiempo, con lo cual se reduce enormemente la cantidad de salidas a utilizar mediante el dispositivo de hardware y es posible direccionar la totalidad de pixeles de un PDM.

Para lograr la variación en intensidad, se divide cada GTU en partes correspondiente al número de pixeles a iluminar durante dicho GTU, con lo cual se divide el rango dinámico de emisión de cada LED. Por ejemplo, se divide un GTU en 4 (encendido de 4 LEDs) y se van dando tiempos de encendido mediante el direccionamiento de dichos LEDs en la matriz hasta alcanzar el número de fotones que a lo largo de todo el GTU se debe emitir sobre el pixel en cuestión, dependiendo de la lluvia que se pretenda simular. Se realiza la misma acción para los GTUs subsecuentes hasta completar el número de GTUs correspondientes a la lluvia que se simula.

Este principio de funcionamiento es muy simple en su concepción inicial, además de que se opera en el régimen de mayor interés que es el de emisión por cantidad de fotones por pixel por GTU, con lo cual, además, se incrementa la cantidad de pixeles a la totalidad del PDM. La matriz que se recorre puede ahora ajustarse al tamaño de la matriz de cada lluvia (de

entre 9 y 25 pixeles típicamente) por lo que se ajusta mejor la simulación a las condiciones reales de funcionamiento. La intensidad de cada LED es, en principio, proporcional ó, en el peor de los casos, sigue una relación bien definida respecto al ancho de pulso con el que se enciende, lo cual será probado más adelante.

La figura 3.15 muestra la forma en que se opera mediante la emisión por tiempo de encendido de LEDs (encendido instantáneo ó PWV), semejante a lo que se realiza en una señal PWM pero sin el promedio que ésta produce debido a que la señal PWV no es periódica ($T \rightarrow \infty$).

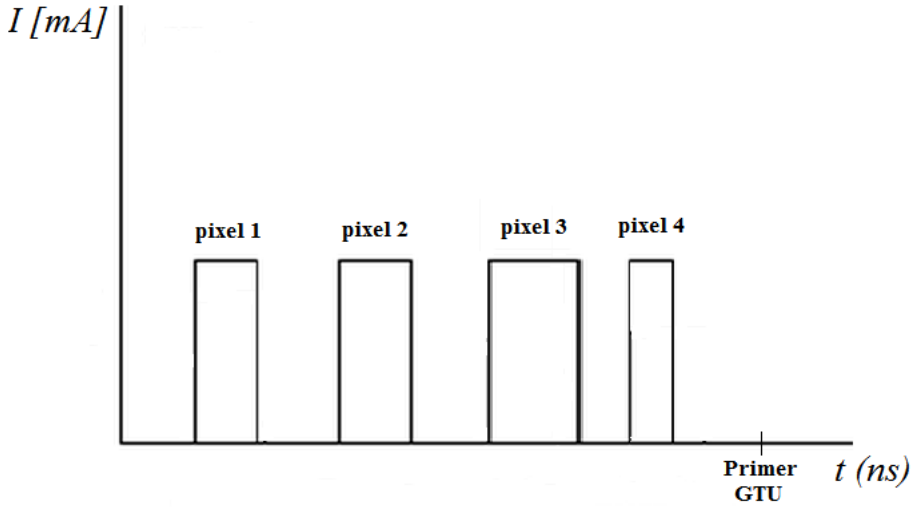


Figura 3.15. Variación de intensidad de LEDs mediante emisión por anchos de pulso variables ó PWV, con división de GTUs en pixeles a encender (en este ejemplo 4) por pulsos de corriente. Los pulsos de corriente mostrados son idealizados para fines ilustrativos.

Circuitos para encendido de LEDs

Ya se ha discutido sobre la forma en que se realizará el control del encendido de LEDs, sin embargo, hasta este punto no se ha comentado nada sobre el circuito que opera directamente a los LEDs y que se vuelve necesario debido a que no es posible conectar directamente los LEDs a los pines de salida del FPGA, pues este último entrega muy poca corriente (del orden de los µA) propiciando que el LED no encienda, además de que el

nivel de salida en voltaje se encuentra limitado a 3.3V y se corre el riesgo de dañar al mismo FPGA.

La forma en que se implementa un circuito sencillo para encender los LEDs es mediante el uso de componentes discretos como transistores, los cuales permiten operar al LED a velocidades de conmutación como la requerida. Antes de entrar en detalles sobre el circuito de encendido del LED UV se comentará un poco sobre las pruebas que se realizaron a éste para determinar su respuesta en cuanto a velocidad de conmutación, y más que ésta, ante su respuesta para pulsos cortos en tiempo.

La primer prueba realizada al LED UV candidato fue, además de conocer la longitud de onda en que emite (espectro de emisión) fue conocer si podía responder a velocidades como las requeridas en el diseño y a pulsos lo más cortos posibles que garanticen el encendido del dispositivo. Estos requisitos de conmutación en el LED se obtienen del diseño base del Track-Sim: frecuencia espacio – temporal de 1 GTU (2.5 µs ó 400 kHz) y pulsos tan "cortos" como sea posible pero que permitan encender y controlar el dispositivo.

Para llevar a cabo esta prueba, se utilizó el arreglo que se muestra en la figura 3.16 utilizando un generador de funciones arbitrarias Tektronix AFG 3252, un osciloscopio digital de fósforo MSO 3032 y el LED UV VAOL-5EUV0T4, en donde los parámetros de operación fueron los siguientes:

- Tren de pulsos de amplitud entre 3 V y 3.2 V, f = 400kHz.
- Los anchos de pulso se variaron desde 250 ns hasta 4 ns.

De esta prueba se observó que el LED puede encender para pulsos cortos (del orden de 5 ns) sin embargo, debido a que el pulso se distorsiona de forma significativa cuando se introduce el LED al circuito, no se alcanza la amplitud necesaria para encender al LED, por lo que en la práctica, valores por debajo de 20 ns pueden encender el LED pero la deformación del pulso provoca que las mediciones resulten difíciles de determinar (por ejemplo amplitud y ancho del pulso). Aunado a esto, no es posible conocer la limitación en

potencia (corriente) que entrega el generador AFG 3252, por lo que la respuesta del LED ante el pulso generado puede estar limitada de inicio por esta situación.

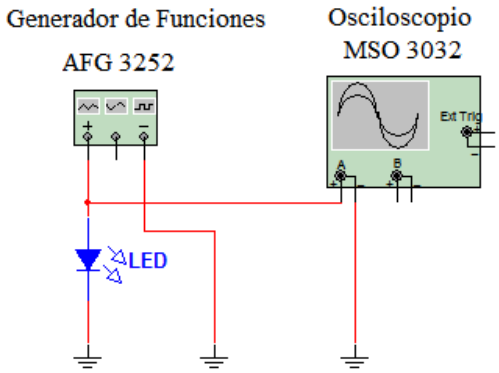


Figura 3.16. Circuito de primer aproximación para encender un LED UV del tipo VAOL – 5EUV0T4 y verificar su comportamiento para pulsos cortos a la frecuencia nominal de 1 GTU (400kHz).

De esta primer prueba se conoce que el LED puede operar a pulsos cortos (~5 ns) siempre que las características de dichos pulsos garanticen el nivel de encendido del LED, por lo que se procede ahora a realizar pruebas mediante el circuito de encendido, en donde la salida en potencia no es un problema de inicio, pero si la respuesta del dispositivo a utilizar.

Básicamente la operación que se necesita es la de un *switch* electrónico, de ahí que los transistores son la mejor solución. El requisito fundamental para el *switch* que se selecciones es la velocidad de respuesta, pues esto puede limitar el diseño si no se obtienen pulsos tan cortos como se requieran. Otro parámetro importante es la potencia que puede entregar el dispositivo y considerando que se pueden tener operando varios LEDs durante un mismo pulso, el *switch* seleccionado debe proporcionar la potencia indispensable para manejar la cantidad de LEDs en cuestión. Dadas estas primeras dos restricciones se examinan las alternativas posibles.

En la actualidad existe una gran variedad de transistores disponibles en el mercado, sin embargo sólo hay dos tipos básicos de tecnologías: TBJ y FET. Estos dos tipos de transistores tienen a su vez múltiples derivados, los cuales dependen del tipo de aplicación principalmente. Para esta aplicación la elección del tipo de transistor se basó

principalmente en la rapidez de conmutación, y desde luego con capacidad de suministrar potencia, por lo que dadas las características de operación y construcción de los transistores, los tipo FET ofrecen mejores rendimientos, principalmente debido a que en los transistores TBJ, tanto electrones como huecos contribuyen a la conducción y la presencia de huecos y su menor movilidad causa que los TBJ requieran mayor corriente y tiempo para conmutar, resultando en menor ancho de banda. Así mismo los transistores FET son deseables dadas sus características como baja resistencia de canal (R_{on}) y bajo voltaje de saturación entre drenaje (*drain*) y fuente (*source*), por lo que la disipación de potencia en el transistor es baja (lo que implica menores perdidas), la impedancia de entrada grande en la terminal de compuerta (*gate*) y como se ha mencionado en contraposición a la mayoría de los transistores TBJ, tienen rápida velocidad de conmutación.

Dentro de los transistores FET que existen, se han elegido a los MOSFET de tipo incremental en tecnología TrenchMOS, ya que, además de contar con las buenas características que se han comentado sobre los transistores FET, éstos mantienen la corriente en cero en el estado de corte, a diferencia de los MOSFET tipo decremental y FETs, en donde, en estado de corte, la corriente entre drenaje y fuente (I_D) no es cero sino que tiene un valor $I_D = I_{DSS}$. Por otra parte, este tipo de transistores MOSFET se encienden con voltajes positivos entre *gate* y *source*, a diferencia de los otros transistores FET y MOSFET, debido a su construcción, la cual, presenta una ausencia de canal entre las terminales *drain* y *source*. [15].

El transistor utilizado para el circuito de encendido de LEDs es el 2N7002F, el cual cuenta con las siguientes características principales:

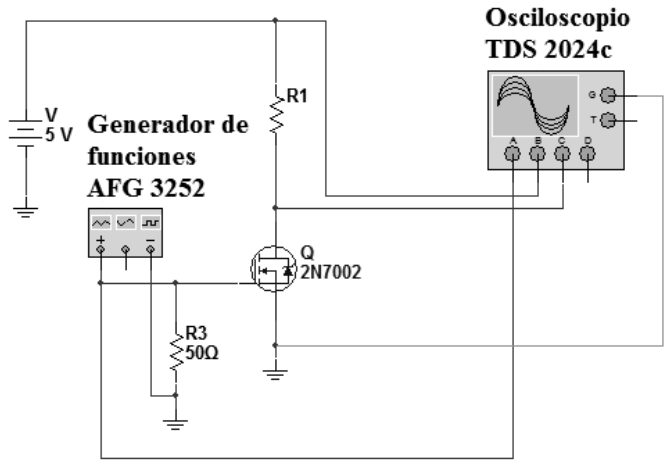
- $V_{DS} \leq 60$ V.
- $R_{DSon} \leq 2\ \Omega$.
- $I_D \leq 475$ mA.
- $P_{tot} \leq 0.83$ W.

[15] Debido a esto, se tienen efectos en el modelo del dispositivo, ya que éste no obedece a la ecuación de Shockley. Para más detalles sobre MOSFETs incrementales y la tecnología TrenchMOS ver glosario y apéndice 3.

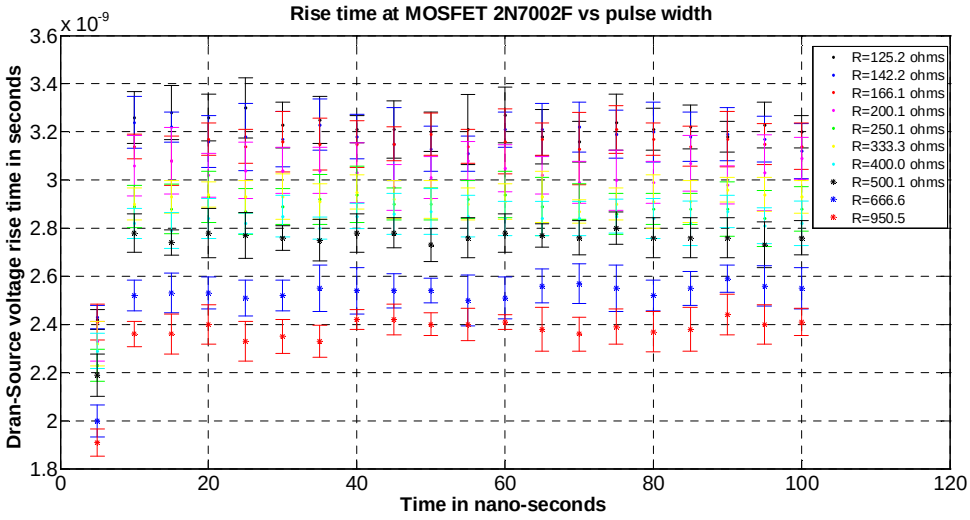
- t_{on} típico de 2.5 ns.
- t_{off} típico de 11 ns.

El encapsulado de este transistor es tipo SOT23 de montaje superficial. La figura 3.17 muestra una implementación del circuito de prueba del transistor con que se midió su tiempo de conmutación. El generador de funciones se configuró para un tren de pulsos con los siguientes parámetros:

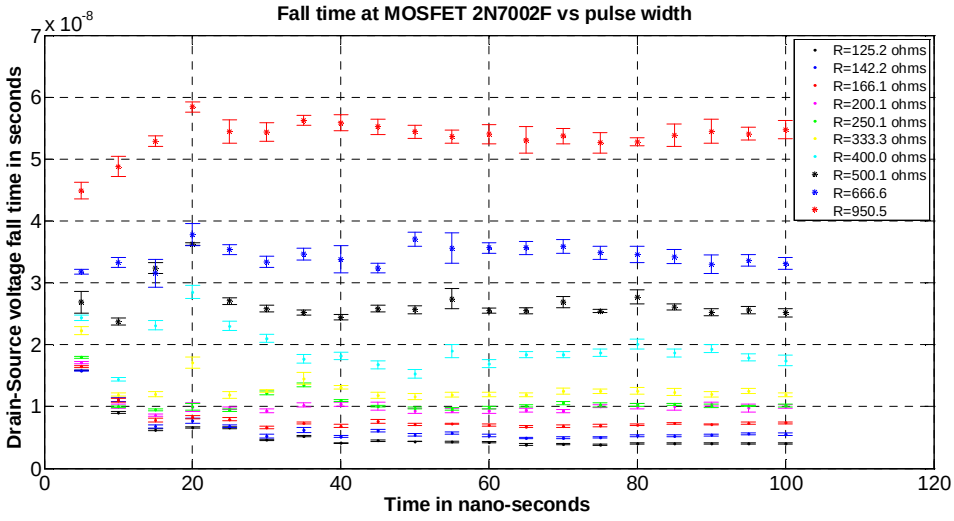
- Amplitud de 5 V.
- Frecuencia de 400 kHz.
- Anchos de pulso desde 5ns.
- Tiempo de levantamiento y decaimiento de la señal del generador de 2.5 ns.



a)



b)



c)

Figura 3.17. a) Circuito de prueba del transistor 2N7002F para encendido y apagado, semejante a como se indica en las pruebas de su hoja de datos. b) Valores de tiempos de levantamiento en voltaje drain-source (V_{GS}) para diferentes corrientes de drain. c) Valores de tiempos de decaimiento en voltaje drain-source (V_{GS}) para diferentes corrientes de drain. Se utilizaron 10 muestras de la señal de voltaje drain-source para cada ancho de pulso desde 5ns y hasta 100ns.

De las figuras 3.17*b* y 3.17*c* se observa que los valores resultantes del tiempo de levantamiento y decaimiento para este transistor se encuentran alrededor del rango que especifica el fabricante en su hoja de datos.

La forma en que se implementó el circuito para la realización de pruebas es el que se muestra en la figura 3.18.

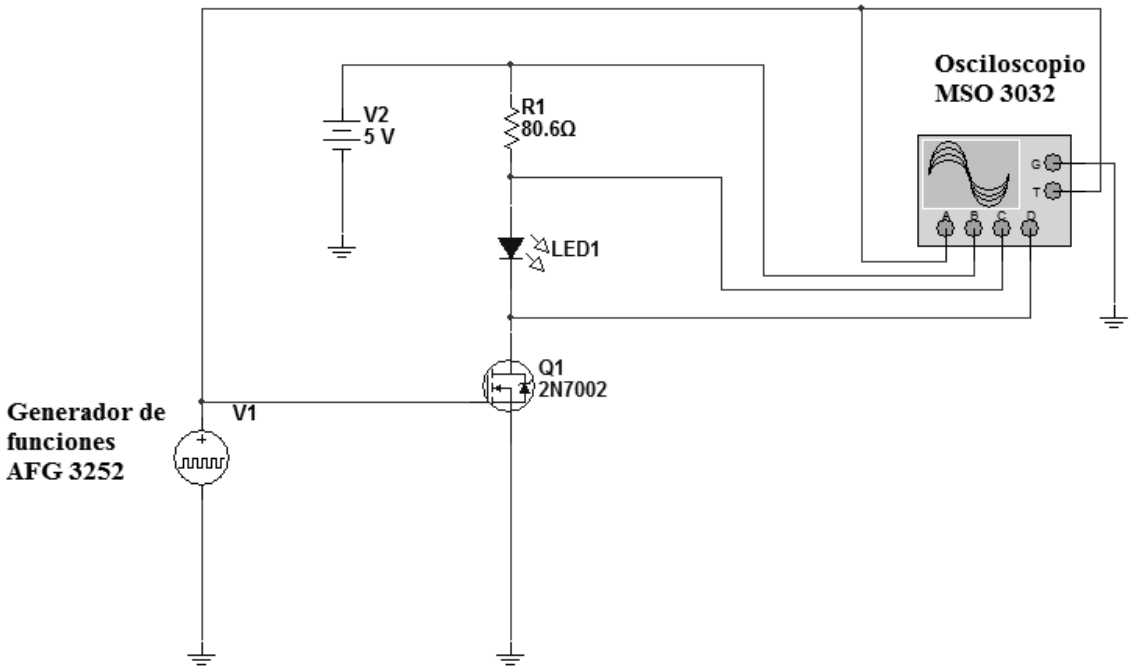
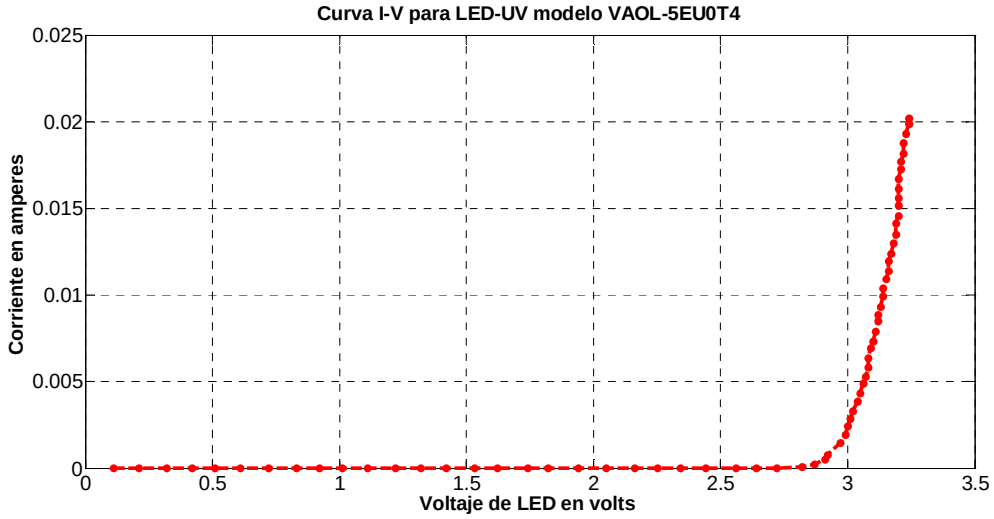
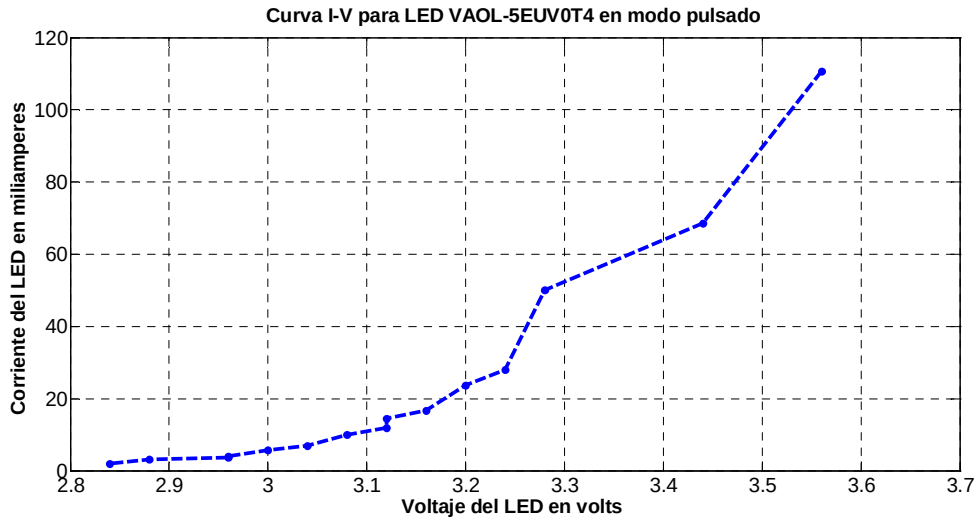


Figura 3.18. Circuito de encendido de LED mediante transistor MOSFET.

En este circuito de la figura 3.18 se pudo verificar el encendido del LED y levantar la curva I-V en modo pulsado para una forma de onda de tren de pulsos con amplitud de 3.3 V, frecuencia de 400 kHz y ancho de pulso (ciclo de trabajo) de 50 ns como se aprecia en la figura 3.19*b*, para comparar el comportamiento con la curva I-V en corriente directa pero no pulsada. (Figura 3.19*a*).



a)



b)

Figura 3.19. Comparación de curvas I-V para el LED UV modelo VAOL – 5EUV0T4. a) Curva I-V en DC en modo no pulsado. b) Curva I-V en modo pulsado mediante transistor MOSFET para el rango de encendido indicado por el fabricante.

Se observa que el modo pulsado es muy semejante a la curva obtenida del LED para DC sin pulsar. En las siguientes secciones se mostrarán resultados obtenidos del encendido del LED UV en términos de intensidad respecto al ancho de pulso aplicado al circuito para esta configuración con MOSFET, así como para otro circuito implementado en el prototipo Track-Flat, en donde se valida y modifican algunos parámetros del diseño.

Diseño a partir de direccionamiento de matriz de LEDs

El rediseño de la manera de operar el Track-Sim pasó de tener dos arreglos matriciales de 3 × 48 y 3 × 68, a una matriz que contempla un PDM completo de 48 × 48 pixeles, lo cual podría pensarse que complicaría la manera de manejar tal cantidad de LEDs pero por el contrario, simplificó la operación al realizarse ahora como el acceso a las coordenadas de una matriz *i*, *j*, de m × n elementos. La figura 3.20 muestra esquemáticamente la forma en que se opera a los LEDs dentro de la matriz de 48 × 48 elementos.

Como se observa en la figura 3.20, se requieren 96 (2 × 48) salidas del FPGA para manejar el mismo número de *switches*, los cuales deberán operar a la velocidad de conmutación que

permita variar de forma controlable la emisión por número de fotones de los LEDs, con corrientes máximo de la nominal de cada LED (20 mA).

El funcionamiento y la forma de operar son como sigue. Se divide en 2 secuencias de 48 × m y 48 × n al *track* a simular. Estas secuencias son cargadas cada una en las dos memorias RAM (columnas y filas). La manera en que van apareciendo los 1s y 0s lógicos en los respectivos pines de salida del sistema realiza las aperturas y cierres de los *switches* de encendido.

El tiempo de cerrado de los *switches* determina el tiempo de encendido de los LEDs y por consiguiente la intensidad instantánea, mientras que la secuencia de encendido (LEDs a encender) determina la forma de la lluvia. El cerrado de los *switches* para obtener la posición, se realiza, de manera conjunta, al reproducir el contenido de las dos memorias, utilizando para esto un circuito secuencial para cada una, el cual consiste de un contador *Up/Down* más una sección combinacional, que es la que determina propiamente que pixeles se requiere iluminar. Las otra memoria RAM operan a los *switches* a la frecuencia que permita obtener los anchos de pulso más pequeños para las intensidades más bajas (el ancho de los pulsos a utilizar así como pruebas en el circuito de encendido serán mostradas en la sección 3.2). El valor de 1s lógicos se repite (un uno lógico a la salida determina cierre y encendido de LEDs) el número de veces necesarias para alcanzar determinado ancho de pulso (intensidad).

La forma en que se conectan los LEDs en la matriz se muestra en la figura 3.21, en donde cada *switch* puede ser un transistor MOSFET, un buffer en circuito integrado o ambos, con cuyo cierre se produce la emisión en el o los LEDs correspondientes. Como se observa, los LEDs para una misma fila o columna se encuentran conectados en forma paralela en ánodo y cátodo (los ánodos se conectan a filas y los cátodos a columnas) por lo que, en el peor caso un solo transistor debe ser capaz de manejar la corriente de varios LEDs en un mismo instante de tiempo, lo cual depende completamente de la geometría de la lluvia así como del nivel de *background* presente en todo momento, pero con la intensión de determinar la capacidad de cada transistor en corriente, se considera el caso típico de un *track* con 21

niveles de intensidad independientes (matriz de 3 × 7) presentes en un mismo GTU, en donde, en el peor de los casos, se tendría a un transistor manejando 7 LEDs en un mismo instante de tiempo, y si se considera la corriente máxima nominal para cada LED de 20 mA, se tiene que el transistor debe manejar una corriente de 7 × 20 mA = 140 mA. El transistor selecciona 2N7002F se encuentra en capacidad de cumplir estos requerimiento, ya que puede manejar corrientes de hasta 475 mA nominales, y picos de 1.9 A durante pulsos de 10 µs. Sin embargo, corrientes de 20 mA en un LED significan intensidades demasiado grandes para el dispositivo MAPMT, por lo que en realidad, como se mostrará en pruebas posteriores en el presente capítulo, intensidades de corriente del orden de 5 mA o menores (dependiendo de la resistencia en serie utilizada con el LED) son las empleadas para brindar un buen control.

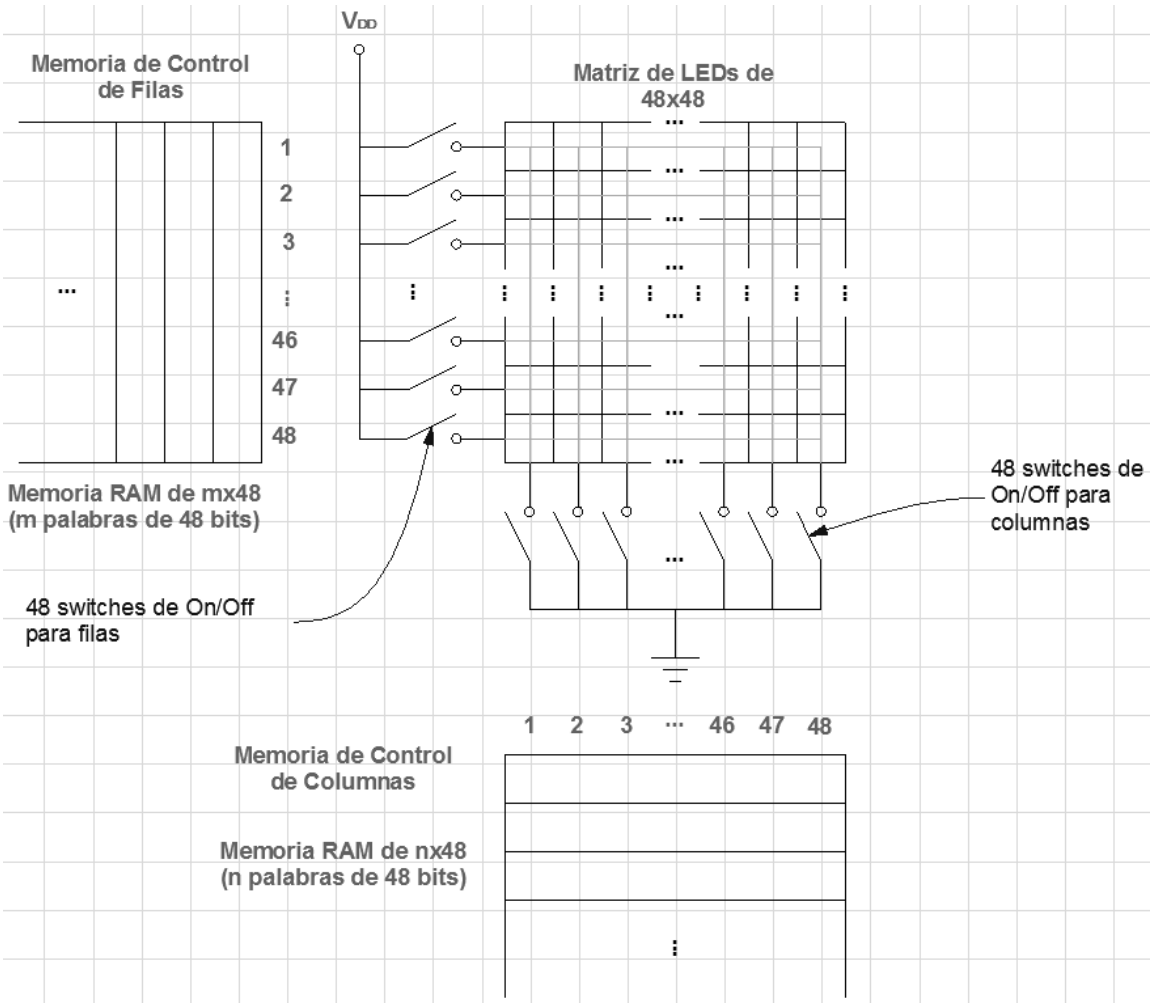
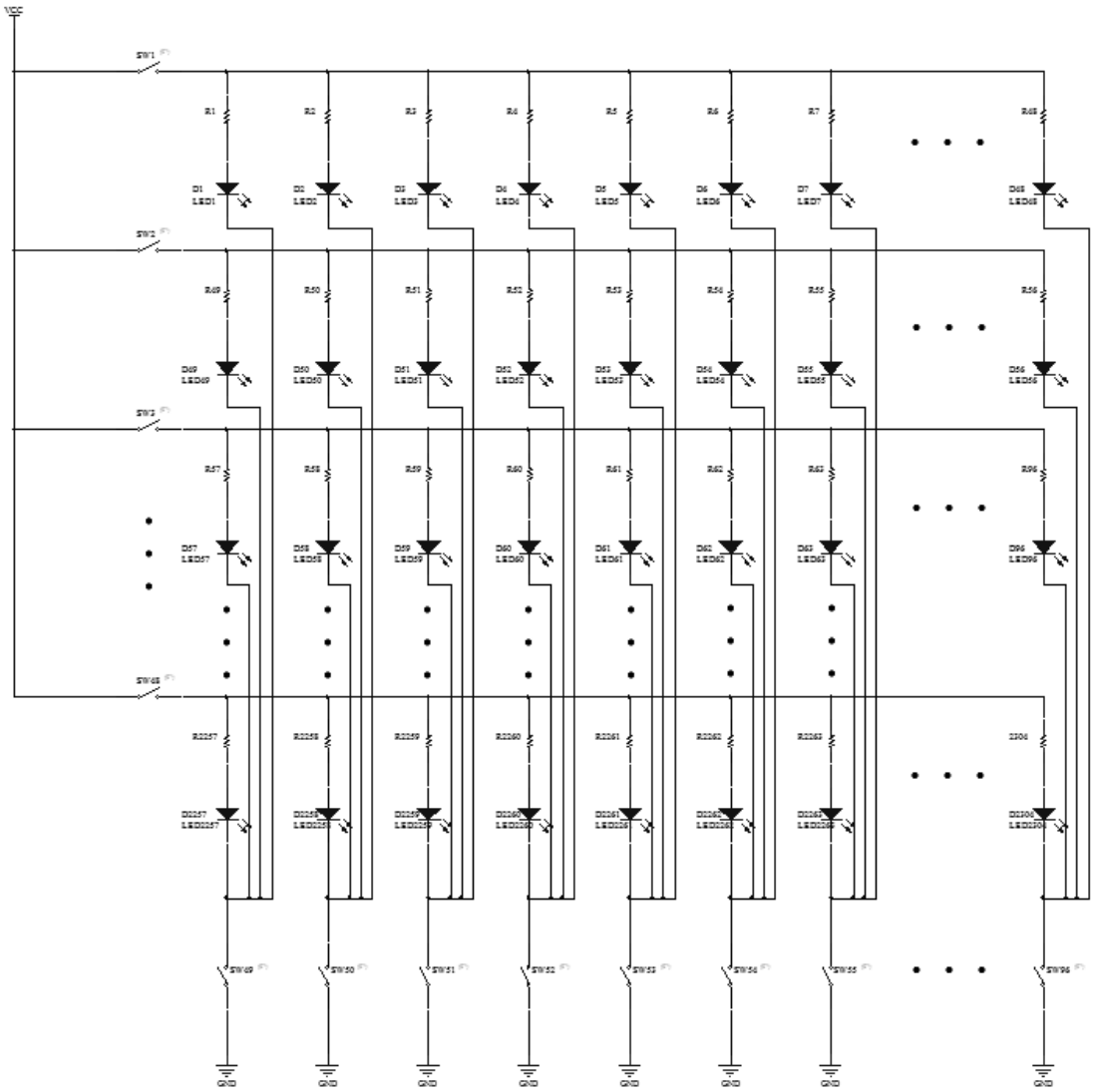
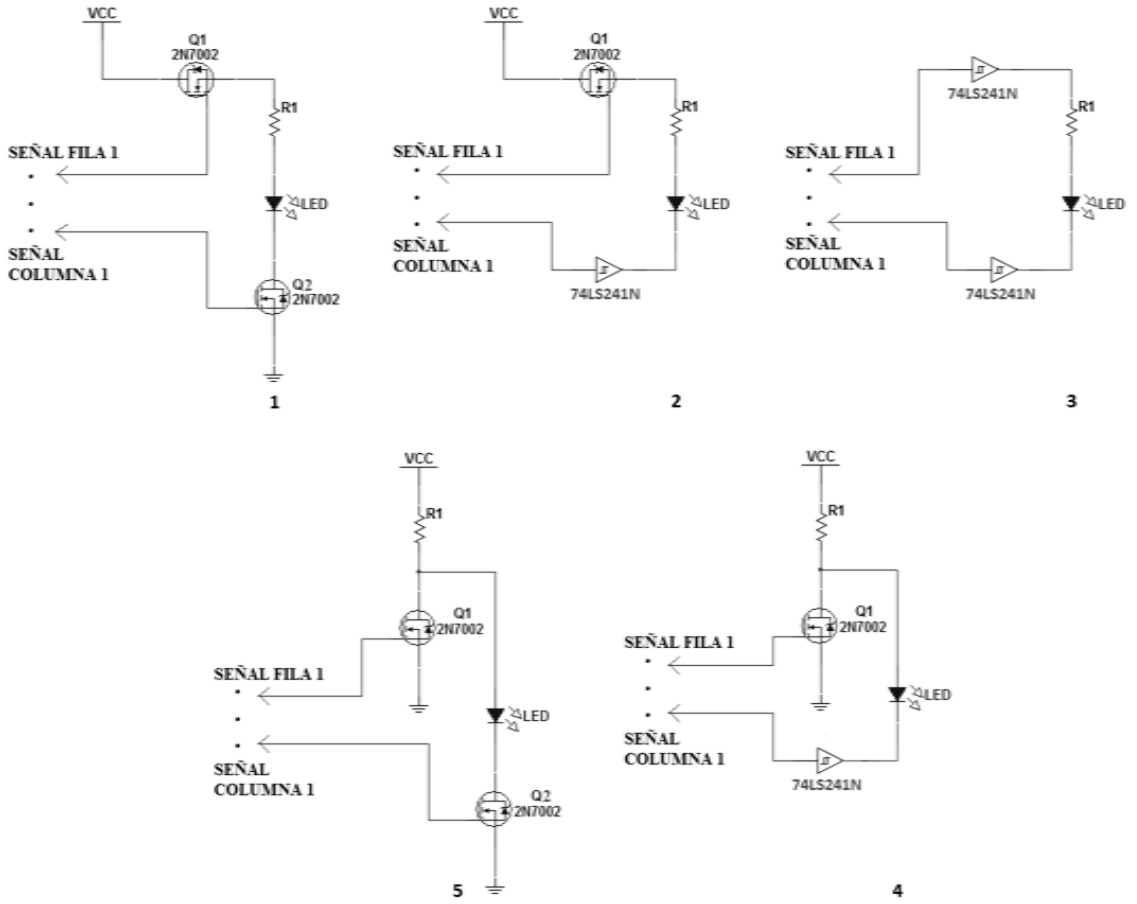


Figura 3.20. Encendido por direccionamiento de la matriz de 48 × 48 LEDs para toda una estructura PDM.



a)



b)

Figura 3.21. a) Conexión de los LEDs en la matriz de 48 × 48 elementos. b) Configuraciones posibles a detalle con transistores MOSFET y buffers como "drivers" de LEDs en filas y columnas (en este caso se muestra un solo LED).

3.1.1 Simulación del diseño base de hardware

Arquitectura del diseño base

Para ser capaces de reproducir los *tracks* sobre un PDM completo se requerirán de dos memorias RAM, las cuales representan el corazón del hardware a implementar. A su vez, para operar estas memorias es necesario utilizar elementos de hardware auxiliares, los cuales, en conjunto, serán un sistema completo capaz de realizar la tarea del simulador Track-Sim. A continuación se describirá con más detalle el hardware de control que se implementa.

La figura 3.22 muestra a grandes rasgos la arquitectura principal (parte de la arquitectura completa) que se implementa, toda ella, dentro de un dispositivo FPGA. Se aprecian las dos memorias RAM que controlan filas y columnas de la matriz con su entrada para cargar datos en 8 bits, compartida entre las dos memorias pero activa para cada una en diferente momento. Así mismo se encuentran 2 secciones ilustradas a cada lado que realizan la reproducción de un *track* plano (principio del prototipo Track-Flat), el cual es un *track* simple cuya intensidad es la misma para los pixeles que recorre y no varía en todo el tiempo que dura dicho *track*, únicamente se desplaza por la matriz (este tipo de *track* será referido a mayor detalle dentro de la sección 3.2). Estos bloques de hardware constan de un contador binario (lado derecho) con un reloj cuya frecuencia es de 400 kHz (2.5µs) el cual, básicamente incrementa o decrementa el valor de su cuenta para colocar a su salida una señal con frecuencia fija y anchos de pulso también fijos, que pasan a un multiplexor digital habilitado para dejar pasar las señales del bloque contador del *track* plano y no las que se encuentran en la memoria RAM de columnas. El bloque que se ubica a la izquierda se encarga de mandar señales de conexión para tres filas, de forma que se mantengan habilitadas durante el desarrollo del *track* plano; de esta forma se cierra el circuito para una matriz de 3 × 48 elementos y es mediante el cierre y apertura de los switches que controlan las columnas que se hace el recorrido del *track* en sentido horizontal, ya sea de izquierda a derecha o de derecha a izquierda, así como el envío de los tiempos de encendido para generar la intensidad del *track* plano. El control en la selección de entrada de los multiplexores así como de los datos que ingresan a cada memoria y la forma en que debe operar toda la arquitectura de hardware, se realizan mediante un bloque de control y comunicación, el cual establece comunicación con el software de control, quien es responsable de fungir como coordinador principal de todas las funciones para operar al instrumento Track-Sim. Este bloque de comunicaciones, como los demás, se encuentra formado de secciones lógicas secuenciales y combinacionales, a partir de bloques funcionales utilizando máquinas de estados.

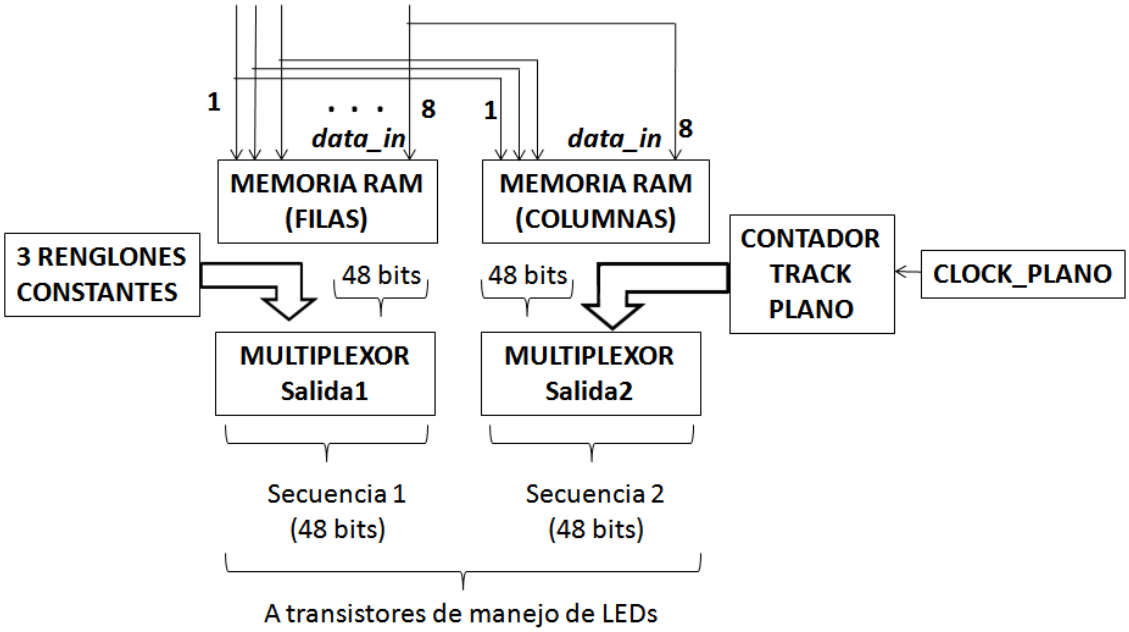


Figura 3.22. Arquitectura base del hardware de control del instrumento Track-Sim.

La figura 3.23 muestra la arquitectura con más detalle del hardware que se implementó, en donde aparecen a la izquierda el bloque de control y comunicación y a la derecha la arquitectura completa dentro del FPGA.

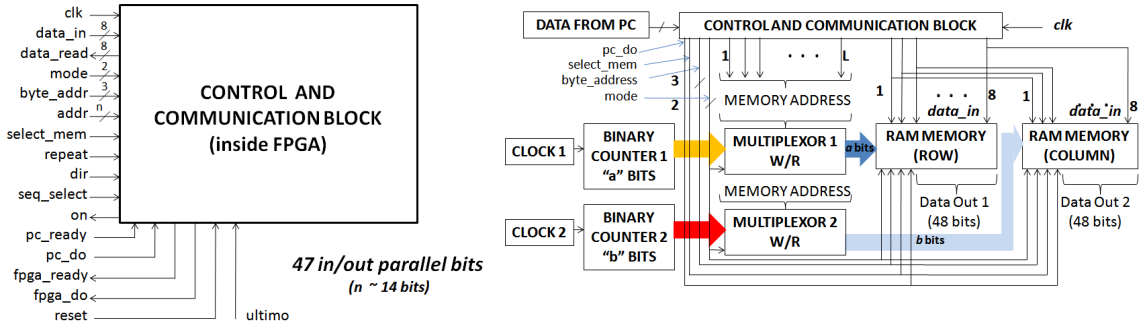


Figura 3.23. Bloque de comunicación y control (izquierda) y arquitectura creada e implementada dentro del dispositivo FPGA (derecha).

El bloque de control y comunicación, así como la arquitectura completa serán comentadas en mayor detalle en la sección 3.3.

Selección del dispositivo FPGA

La elección del dispositivo a utilizar se realizó tomando en cuenta diversos factores que surgieron a partir del planteamiento de la arquitectura así como de parámetros preestablecidos en el diseño base. Estos factores y parámetros son los siguientes:

- Un total de 143 pines como máximo disponibles para el usuario de los cuales: 96 pines de salida son utilizados como canales para las señales que manejan los *switches* (MOSFETS) y 47 pines de entrada-salida utilizados por el bloque de comunicación y control (1 pin de reloj, 8 pines de datos de entrada a memorias, 8 pines de lectura de datos, 2 pines de entrada para modo de operación, 3 pines de entrada para seleccionar byte de datos de entrada a memorias, 14 bits de datos para direccionar memorias, 1 bit de entrada para selección de memoria, 1 bit de control "*repeat*" para opciones de operación, 1 bit de dirección de reproducción de *track*, 1 bit para elegir entre *track* en memorias o *track* plano, 1 bit de salida de estado "*on*", 1 bit bandera de entrada indicador denominado "*pc_ready*", 1 bit bandera indicador de entrada denominado "*pc_do*", 1 bit bandera de salida indicador denominado "*fpga_ready*", 1 bit bandera de salida indicador denominado "*fpga_do*", 1 bit de entrada para "*reset*" y 1 bit de entrada bandera denominado "*ultimo*").
- Capacidad de manejar frecuencias para obtener pulsos de 10 ns o menores.
- Tiempos de levantamiento y decaimiento cortos así como retrasos cortos (menores a 5 ns).
- Capacidad de almacenamiento de la arquitectura a implementar.

Dados estos requisitos se realizó una búsqueda de diferentes dispositivos, de entre los cuales se utilizó el FPGA ProASIC3E A3PE1500 en encapsulado PQG208 de Actel, el cual cuenta con las siguientes características:

- Sistema compuesto por 1 500 000 compuertas.
- 38 400 flip-flop tipo D.
- 270 kbits de memoria RAM (de 1024 bits).

- 1 kbit de memoria flashROM.
- 2 circuitos PLL (para manejo de señales de reloj).
- 8 bancos (conjuntos de pines) dedicados a pines de entrada/salida.
- 147 pines de entrada/salida disponibles para el usuario, de los cuales 65 pueden ser utilizados como pares diferenciales.
- Operación a una frecuencia máxima de 350 MHz. La tarjeta incorpora un oscilador de 40 MHz y tiene cabida para un segundo oscilador así como una entrada mediante un conector SMA para una fuente externa como puede ser un generador de funciones o de pulsos.

Además de las características arriba mencionadas y que cumplen con los requisitos previamente establecidos, se trabajó con este dispositivo principalmente debido a que ya se contaba con él.[16] El sistema consta de una tarjeta PCB prototipo de 6 capas, la cual cuenta con acceso a todos los pines del encapsulado por parte del usuario y con el mínimo de hardware asociado para su funcionamiento. Otra opción es adquirir un chip por separado y elaborar la tarjeta PCB a medida, sin embargo el trabajo que implica tal desarrollo es un tema propio de investigación y queda fuera de los alcances del presente trabajo. Por otra parte, el tamaño de la tarjeta y la forma en que se encuentra constituida la vuelven una herramienta de fácil utilización por lo que su incorporación a un prototipo se vuelve bastante accesible.

Ahora se procederá a mostrar los resultados de las simulaciones realizadas en la arquitectura creada.

Simulaciones realizadas en Modelsim

Las simulaciones del funcionamiento del hardware se realizaron utilizando la herramienta de simulación digital Modelsim® de Mentor Graphics.[17] El primer paso fue desarrollar el código para la creación del hardware, lo cual se realizó utilizando un lenguaje de

[16] Esta tarjeta en un principio se utilizaría en otros experimentos y pruebas.
[17] Software especializado en simulación de sistemas basados en FPGA así como ASIC para lenguajes de descripción de hardware como VHDL, Verilog y System C.

descripción de hardware o HDL (*Hardware Description Language*) denominado VHDL.[18] El código escrito en VHDL para el instrumento Track-Sim se encuentra en el apéndice 2A.

Una vez creado el código, se procede a su compilación utilizando para esto la plataforma Modelsim, así como las librerías del dispositivo FPGA seleccionado (en este caso el ProASIC3E A3PE1500 de Actel). Una vez compilado el código se crea un archivo con las señales que servirán como entradas a la arquitectura a simular, considerando para esto los tiempos en que se espera funcione la arquitectura, los niveles lógicos de condiciones iniciales, cambios en los niveles lógicos de las señales, así como la señal de reloj con la frecuencia a la que se operará. Teniendo esto listo se procede a la simulación para un determinado tiempo de duración.

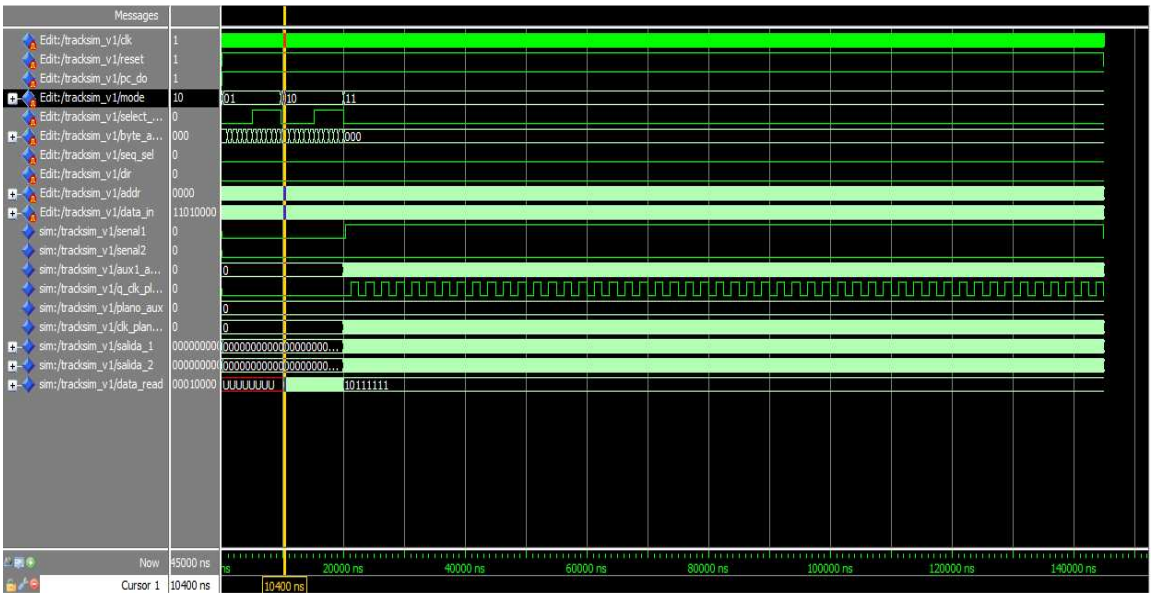
Las condiciones con las cuales se realizó la simulación son las siguientes:

- Empleo de la matriz completa de 48 × 48.
- Escritura y lectura de ambas memorias (simulación de un *track*[19] creado con fines ilustrativos para memorias de tamaño 48 × 16 operando ambas a la misma frecuencia).
- Reloj principal con una frecuencia de 40 MHz (reloj que incorpora la tarjeta ProASIC3/E Starter Kit).
- Simulación del *track* plano.

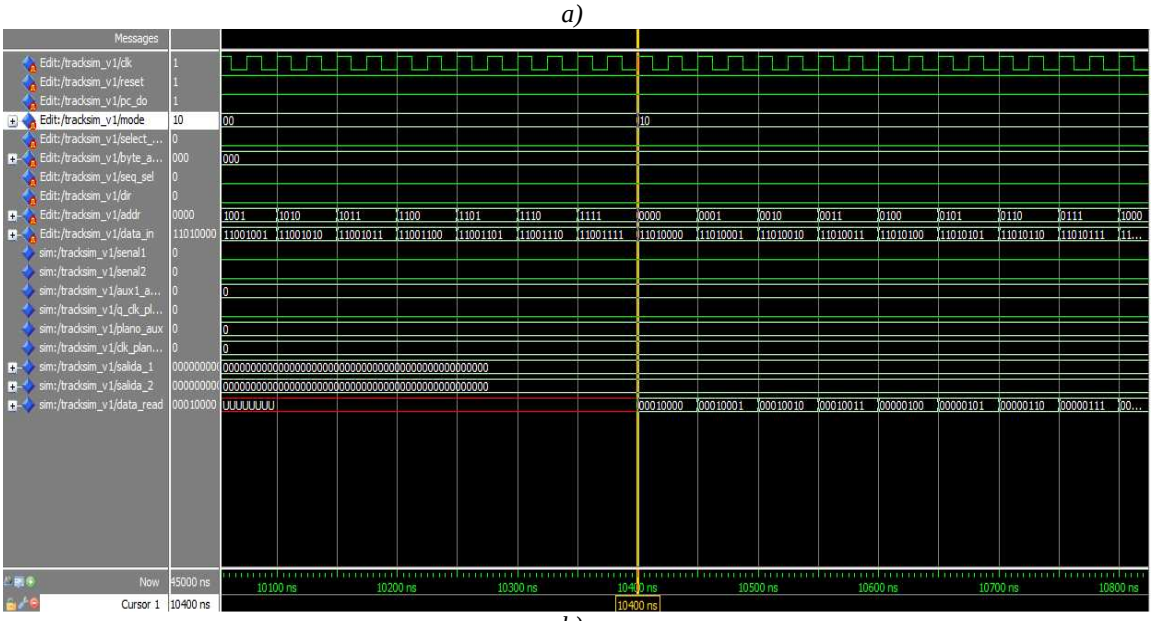
La figura 3.24 muestra los resultados que se obtienen de la simulación en Modelsim para el *track* en la matriz completa, mientras que la figura 3.25 muestra una simulación realizada para un *track* plano en una matriz de 3 × 48.

[18] VHDL es un acrónimo de la combinación de VHSIC (Very High Speed Integrated Circuit) y HDL. El VHDL es un lenguaje utilizado para describir y formar hardware en circuitos digitales definido por la IEEE.

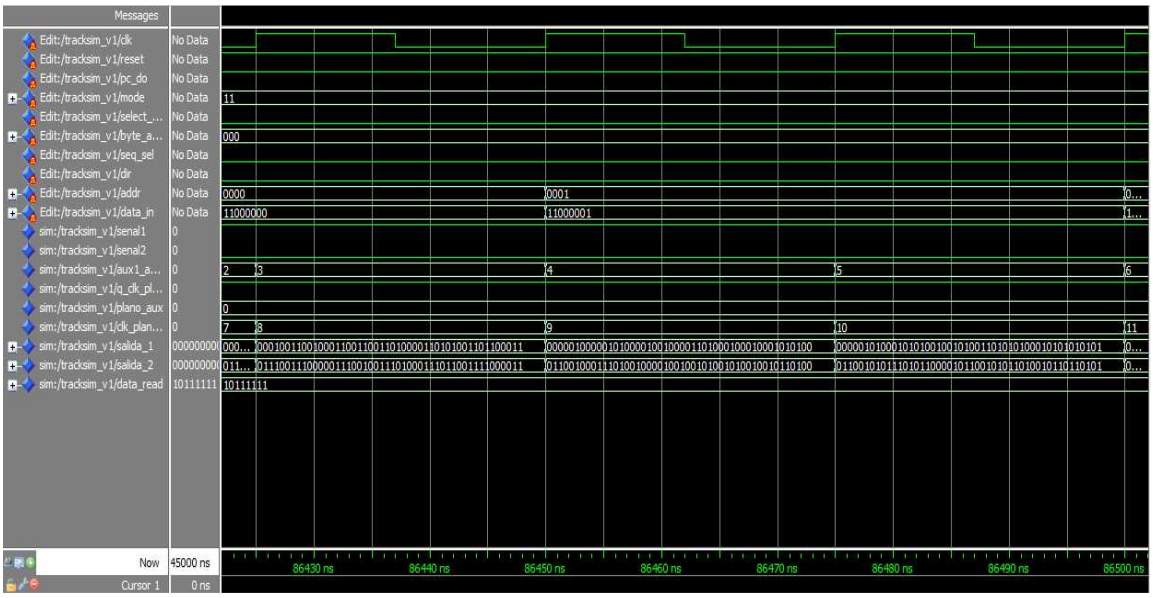
[19] No es un track propiamente dicho lo que se crea para simular la escritura y posterior lectura en las memorias RAM sino diferentes secuencias de 48 bits que se almacenan en 16 localidades de memoria, que es el tamaño de ambas memorias utilizado para efectos de prueba.


Messages
Edit:/tracksim_v1/clk
Edit:/tracksim_v1/reset
Edit:/tracksim_v1/pc_do
Edit:/tracksim_v1/mode
Edit:/tracksim_v1/select_...
Edit:/tracksim_v1/byte_a...
Edit:/tracksim_v1/seq_sel
Edit:/tracksim_v1/dir
Edit:/tracksim_v1/addr
Edit:/tracksim_v1/data_in
sim:/tracksim_v1/senal1
sim:/tracksim_v1/senal2
sim:/tracksim_v1/aux1_a...
sim:/tracksim_v1/q_clk_pl...
sim:/tracksim_v1/plano_aux
sim:/tracksim_v1/clk_plan...
sim:/tracksim_v1/salida_1
sim:/tracksim_v1/salida_2
sim:/tracksim_v1/data_read
Now
Cursor 1
20000 ns
40000 ns
60000 ns
80000 ns
100000 ns
120000 ns
140000 ns
10400 ns

a)

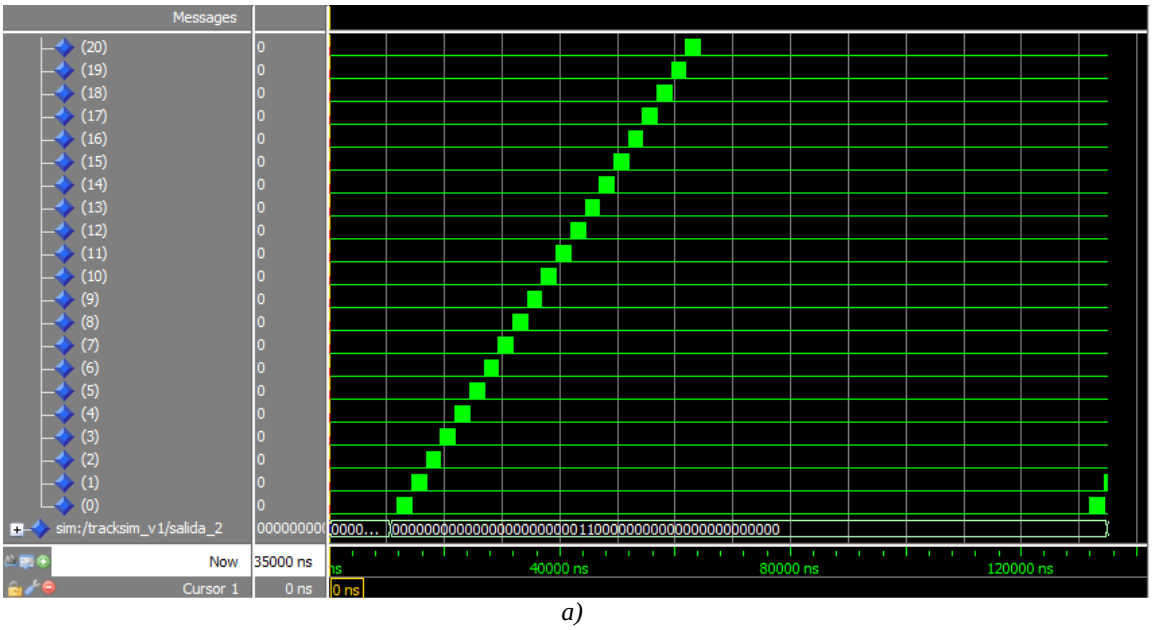

Messages
Edit:/tracksim_v1/clk
Edit:/tracksim_v1/reset
Edit:/tracksim_v1/pc_do
Edit:/tracksim_v1/mode
Edit:/tracksim_v1/select_...
Edit:/tracksim_v1/byte_a...
Edit:/tracksim_v1/seq_sel
Edit:/tracksim_v1/dir
Edit:/tracksim_v1/addr
Edit:/tracksim_v1/data_in
sim:/tracksim_v1/senal1
sim:/tracksim_v1/senal2
sim:/tracksim_v1/aux1_a...
sim:/tracksim_v1/q_clk_pl...
sim:/tracksim_v1/plano_aux
sim:/tracksim_v1/clk_plan...
sim:/tracksim_v1/salida_1
sim:/tracksim_v1/salida_2
sim:/tracksim_v1/data_read
Now
Cursor 1
10100 ns
10200 ns
10300 ns
10400 ns
10500 ns
10600 ns
10700 ns
10800 ns
10400 ns

b)

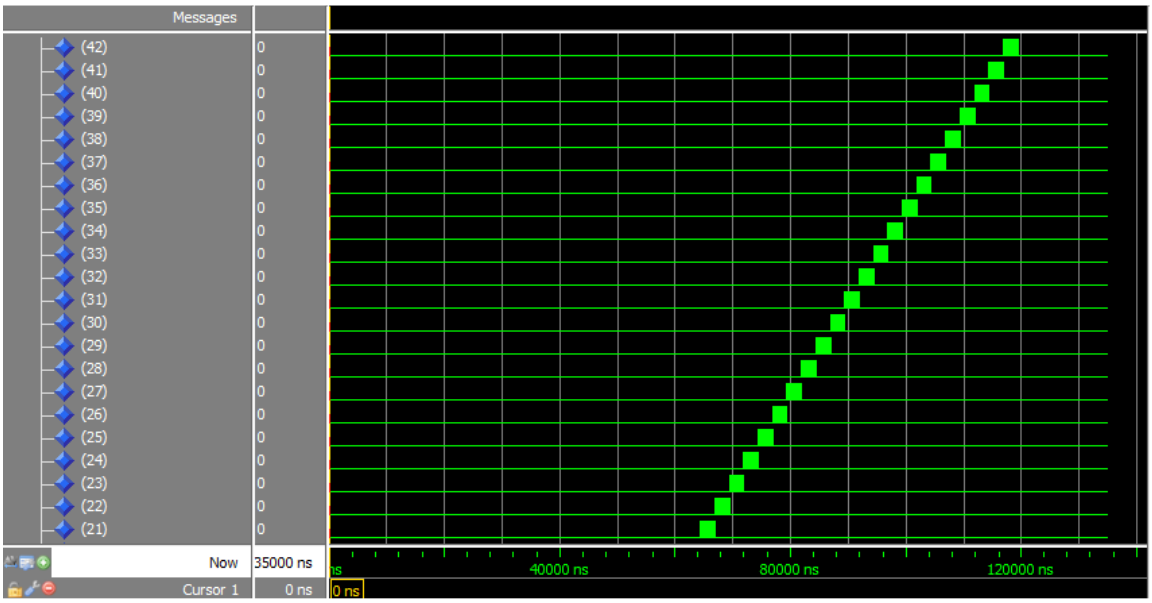


c)

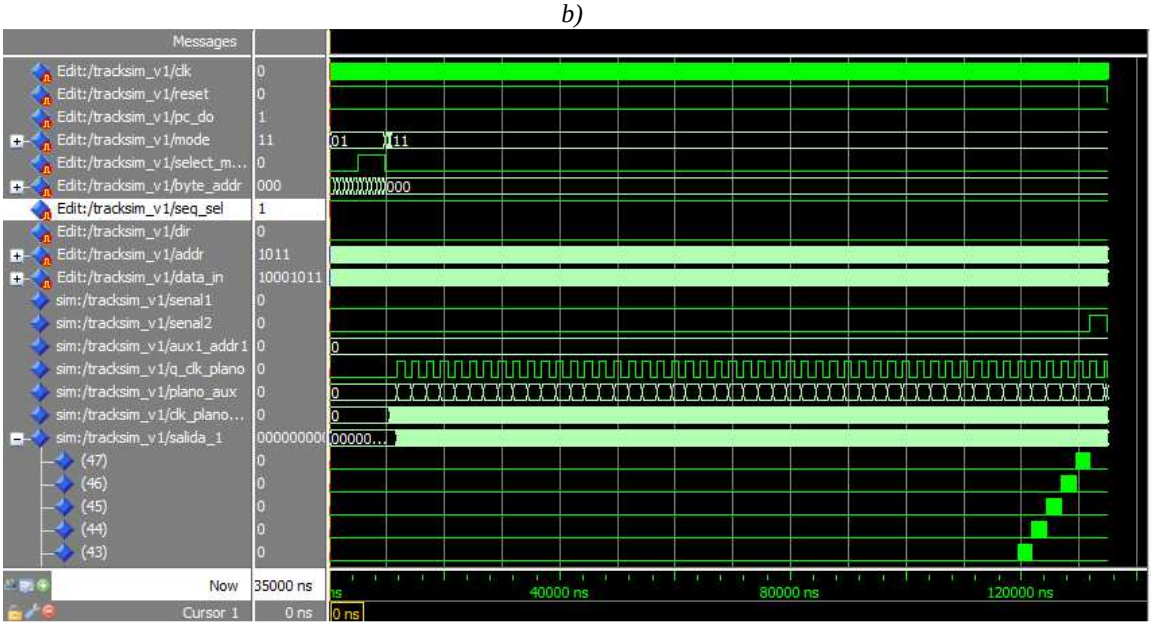
Figura 3.24. Simulaciones realizadas para la arquitectura creada en el dispositivo FPGA ProASIC3E de Actel. a) Aspecto de la simulación completa en 140 µs. b) Primeros 10 800 ns de la escritura en cada una de las memorias. c) Lectura y reproducción del contenido de ambas memorias RAM en sincronía con la señal de reloj "clk" de 40 MHz (25 ns).



a)



b)



c)

Figura 3.25. Simulación de un track plano dentro de las opciones de simulación del Track-Sim. a) Primeros 21 canales del track plano; cada separación entre trenes de pulsos es de 2.5 µs. b) Siguientes 22 canales. c) Últimos 5 canales (canal 43 a 47).

3.2 VALIDACIÓN DEL HARDWARE DE GENERACIÓN DE PULSOS: PROTOTIPO TRACK-FLAT

El prototipo Track-Flat es un dispositivo creado para reproducir y validar en forma de una traza plana de luz UV, a lo largo de 8 pixeles para una PMT multiánodo (MAPMT), las diferentes etapas que involucran el proceso de simulación del Track-Sim: desde la arquitectura de hardware para generación de pulsos, circuitos de manejo de LEDs, forma de realizar el control de intensidad, y hasta las partes ópticas como lo es la guía de luz para focalizar sobre 8 pixeles, pasando por las características ópticas de los LEDs a utilizar, el acoplamiento LED-fibra óptica, la fibra misma, la salida de la fibra y su acoplamiento con la MAPMT así como la detección en la PMT. En principio se busca que la intensidad en cada pixel sea la misma, de ahí el nombre de Track-Flat, por lo que la salida de cada guía de luz debe emitir con la misma cantidad de fotones. Además de emitir en la misma intensidad en cada pixel, se pensó el prototipo para tener opción de generar varios niveles de intensidad, esto es, cada nivel de intensidad emite como *track* plano (con la misma intensidad en cada pixel) pero con diferente nivel de intensidad entre los niveles disponibles, siento éstos elegibles por el usuario. Las principales características del prototipo Track-Flat son:

- Diferentes niveles de intensidad seleccionables por el usuario (misma intensidad en cada pixel para cada nivel elegido).
- Posibilidad de generar la traza en sentido horizontal de derecha a izquierda o de izquierda a derecha.
- Dos modos de operación continuo (el *track* se repite de manera indefinida hasta que es parado por el usuario) y de una sola reproducción (se reproduce el *track* una sola vez hasta que el usuario decide volver a reproducirlo).

A fin de simplificar el manejo del prototipo, éste ha sido dividido en dos secciones, control y óptica (mostradas en la figura 2.10 del capítulo anterior), las cuales consisten principalmente de lo siguiente:

- Gabinete con electrónica de control, el cual genera los pulsos en 8 canales de las diferentes intensidades y realiza el manejo de los LEDs.
- Guía de luz a base del conjunto LED-fibra óptica que emite luz UV en 8 pixeles de la MAPMT (modelo H7546B de Hamamatsu).

Diseño base del Track-Flat

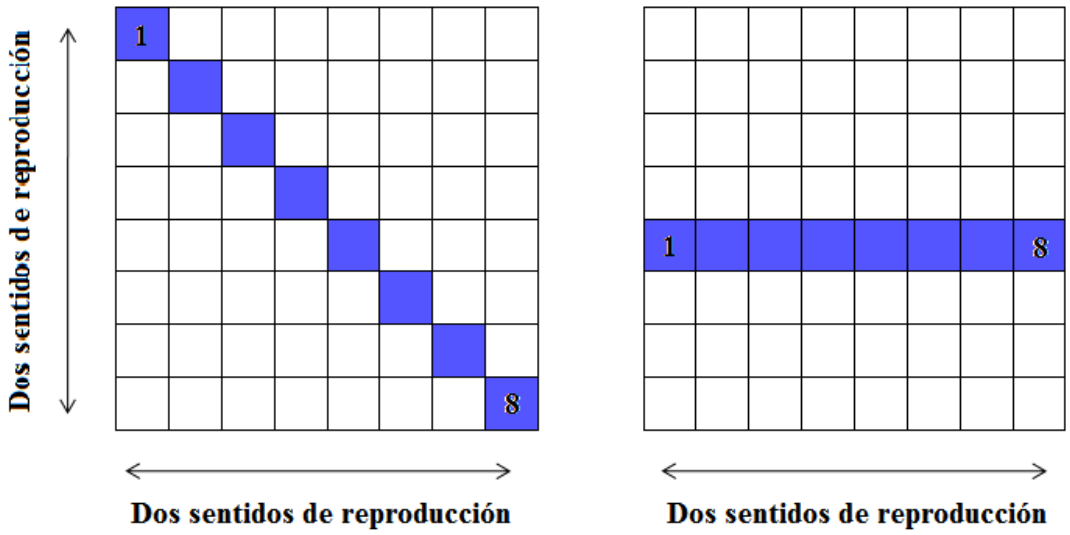
Los requerimientos básicos para el prototipo Track_Flat son:

- 8 pixeles contiguos de una MAPMT (diseñado para el modelo utilizado por JEM-EUSO pero probado con el modelo H7546B).
- Distanciamiento espacio-temporal para encendido entre pixeles de 1 GTU (2.5 μs).
- En principio el mismo tiempo de encendido para cada LED para lograr la misma intensidad en número de fotones.

La figura 3.26 muestra gráficamente la forma en que se reproduce el *track* plano en 8 pixeles de una MAPM. Por simplicidad principalmente en la parte óptica (guía de luz), se escogen 8 pixeles adyacentes en forma horizontal para recibir la traza plana, como se aprecia en la figura 3.26*b*.

Para lograr los niveles de intensidad se utilizará el método descrito en la sección 3.1 mediante variación de anchos de pulso por emisión instantánea ó PWV, y utilizando los elementos de la electrónica que serán empleados en el Track-Sim.

Comenzando por la generación de los pulsos que controlan la intensidad de los LEDs, se utiliza una arquitectura semejante a la implementada para el Track-Sim, sin embargo en una versión reducida pero que lleva a cabo básicamente la misma función. La principal distinción con la arquitectura del Track-Sim se refiere a que en el Track-Flat el control no se realiza desde una computadora con un programa especial, sino que en el mismo instrumento se incorporan etapas a partir de *switches* y botones que el usuario opera.



a) *b)*

Figura 3.26. Reproducción de una traza plana en 8 pixeles adyacentes de una MAPMT. a) Traza en 8 pixeles adyacentes en dirección y sentido diagonal de una MAPMT de 64 pixeles. b) Simplificación en 8 pixeles adyacentes horizontales en dos sentidos (izquierda a derecha y derecha a izquierda). Los pasos entre pixeles (pixeles en color violeta) se realiza en intervalos de 1 GTU.[20]

Hardware del dispositivo: diagrama de flujo y a bloques de la arquitectura

El hardware que se creó para el prototipo Track-Flat permite generar un único pulso en cada uno de los 8 canales con un ancho que puede elegirse de entre 10 valores posibles, distanciados temporalmente 2.5 µs entre canales y de la misma amplitud. Para realizar esto se parte de los requerimientos y se genera el diagrama de flujo que aparece en la figura 3.27, donde se comienza con el modo de funcionamiento, continuo o de una sola reproducción, para posteriormente agregar la elección del ancho de pulso o intensidad deseada, así como el sentido de la reproducción del *track*.

Con esta base del diseño, se hace posible bosquejar la arquitectura e implementar el diseño, pasando al diagrama de bloques de la figura 3.28. Esta arquitectura se compone de dos partes principales: la primera formada por tres bloques secuenciales y la segunda por 2

[20] Por simplicidad se omite la separación existente entre pixeles de la MAPMT pero ésta es considerada en el diseño óptico de la guía de luz, a fin de ajustarse al máximo sobre cada pixel en que se emite. Para más información sobre esto ver el trabajo de tesis "Diseño de un Sub-Sistema Opto-electrónico de un simulador de trazas para la calibración en tierra de la superficie focal del observatorio espacial JEM-EUSO", César Tavera Ruiz, maestría en instrumentación, Facultad de Ingeniería, UNAM 2011.

bloques de lógica combinacional. La sección secuencial se encuentra formada por un generador de reloj de 2.5 µs y 2 contadores binarios, mientras que la sección combinacional se forma a partir de un circuito "multiplexor-decodificador" y un multiplexor de salida para toda la arquitectura. Cada uno de estos bloques realiza las siguientes acciones.

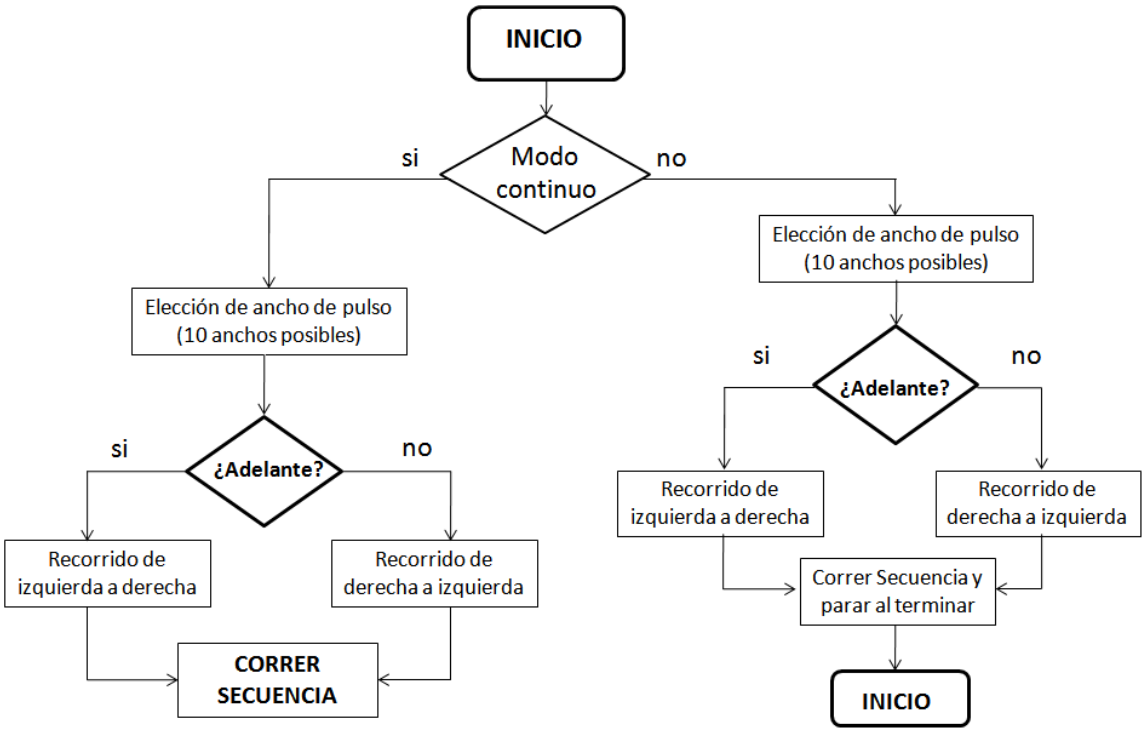


Figura 3.27. Diagrama de flujo de la operación del Track-Flat.

Reloj de 10 ns: Clock1

Este bloque sólo se agrega en el diagrama para que la arquitectura se visualice de manera más sencilla, pero básicamente se trata de la misma señal de reloj de 100 MHz (10 ns) del FPGA.

Reloj de 2.5 µs: Clock2

Este bloque genera, a partir de la señal de reloj del FPGA, un reloj con un periodo de 2.5 µs (frecuencia de 400 kHz) que servirá para realizar el cambio entre LEDs y por consiguiente entre pixeles, a intervalos de 1 GTU. Incorpora una entrada para habilitación (*reset*) de

manera que comienza a funcionar, al igual que todos los demás bloques, a partir de que se le indica. La señal de reloj generada es utilizada por uno de los contadores binarios.

Contadores binarios

Uno de los dos contadores binarios es incorporado en un mismo bloque, junto con el reloj de 2.5 µs a fin de hacer más eficiente el código y la síntesis del mismo en el hardware dentro del FPGA, sin embargo, para fines explicativos se manejan como dos contadores por separado (como aparece en la figura 3.28). En conjunto los dos contadores realizan las siguientes acciones. Uno de ellos, el que se incorpora en el bloque del reloj de 2.5 µs y que se define como *Binary Counter 1*, tiene como función el realizar un conteo, tanto para generar la señal de 2.5 µs, así como para contar periodos del reloj de 100 MHz del FPGA que genera valores en una señal utilizada como sincronía, a fin de generar los anchos de pulso que se van a enviar a los LEDs.

En lo que respecta al otro contador binario, denominado *Binary Counter 2*, éste tiene como función el determinar el paso entre cada canal de salida para cada uno de los 8 LEDs, en los dos sentidos, izquierda a derecha y derecha a izquierda. Al igual que el reloj de 2.5 µs, este contador incorpora una entrada de habilitación.

Circuito combinacional "multiplexor-decodificador"

Este bloque es el que se encarga propiamente de generar los pulsos con ancho variable o PWV. Básicamente depende del valor que vea en sus entradas para proceder a poner en su salida el respectivo o los respectivos valores de ancho de pulso, haciendo esto en sincronía con las señales que recibe de los contadores binarios y de las señales de control enviadas por el usuario. De forma general, puede verse a este circuito como una función booleana con diversas salidas y diversas entradas que puede ser representada en una tabla de verdad.

Multiplexor de salida

El multiplexor de salida tiene como función el servir de *switch* para habilitar las señales de los 8 canales y dejar de operar cuando así se requiera. Cuando se habilita para operar las señales, pone en los pines de salida del FPGA los pulsos que se generan en el circuito combinacional "multiplexor-decodificador" hasta que se deshabilitan éstos y entonces se pone en los pines de salida un valor lógico de cero, con lo que en el circuito de manejo de LEDs, se "abren" los *switches* a base de transistores MOSFET y/o buffers para apagar los LEDs.

Como se comentó en los bloques secuenciales, estos circuitos combinacionales presentan entradas de habilitación, y la principal de ellas es controlada por el usuario mediante el botón de *reset*.

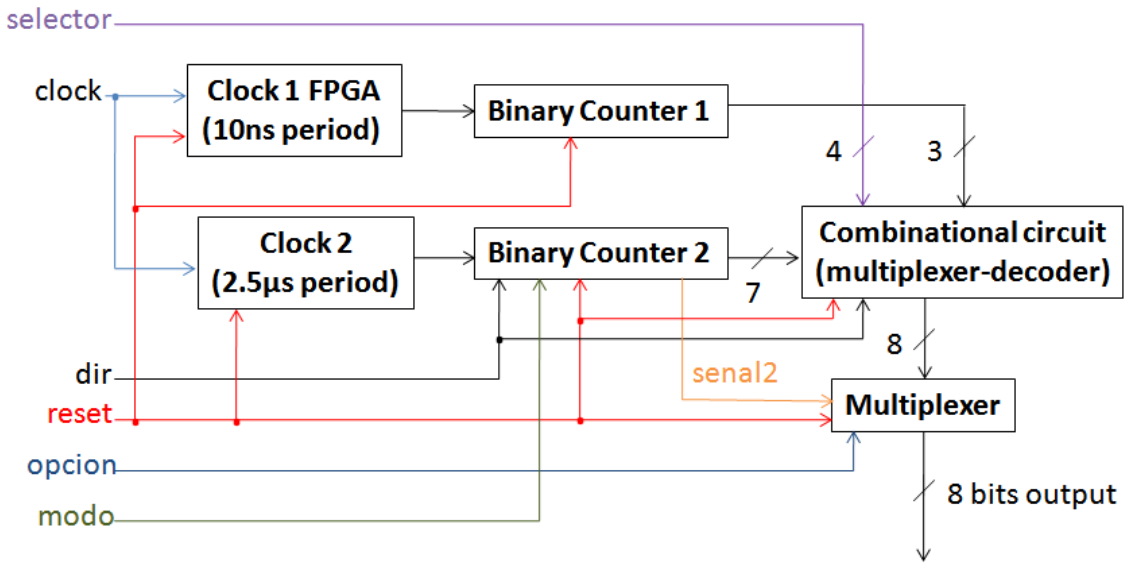
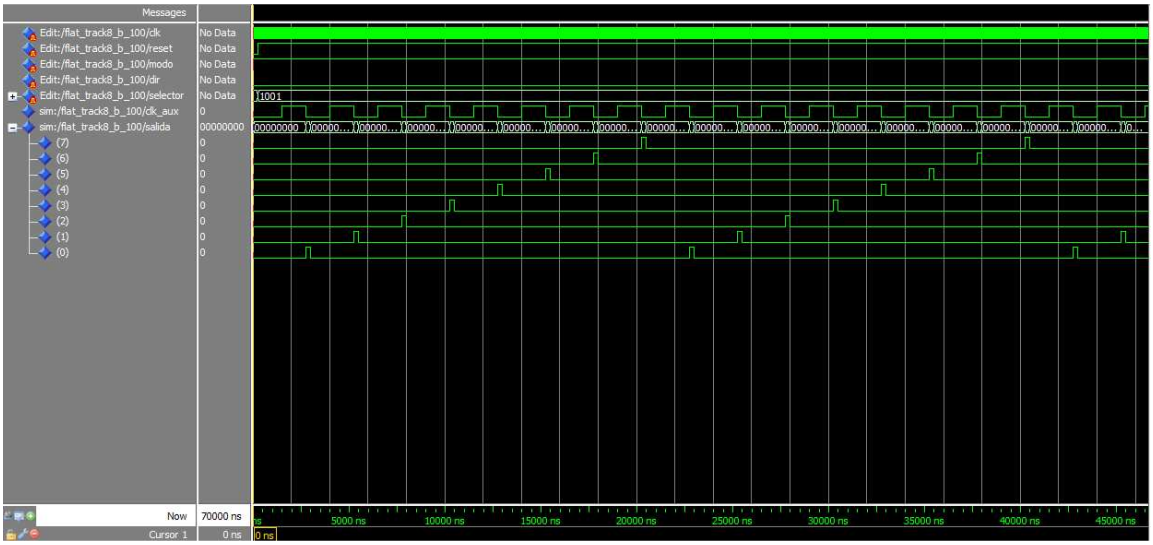


Figura 3.28. Arquitectura del prototipo Track-Flat.

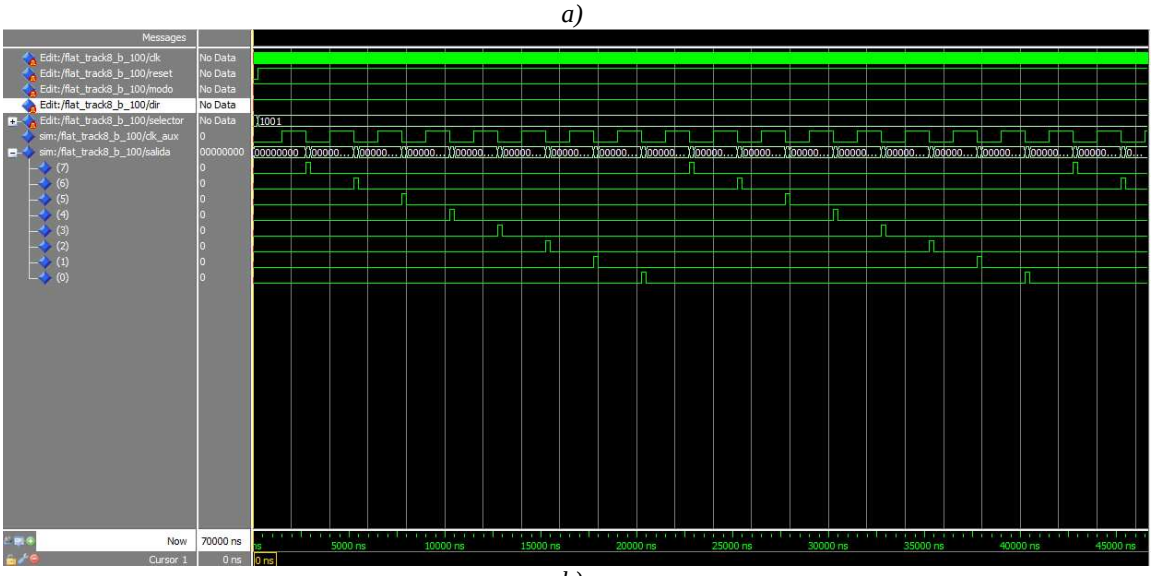
El código en VHDL creado para esta arquitectura puede encontrarse en el apéndice 2B.

Simulación del diseño en VHDL

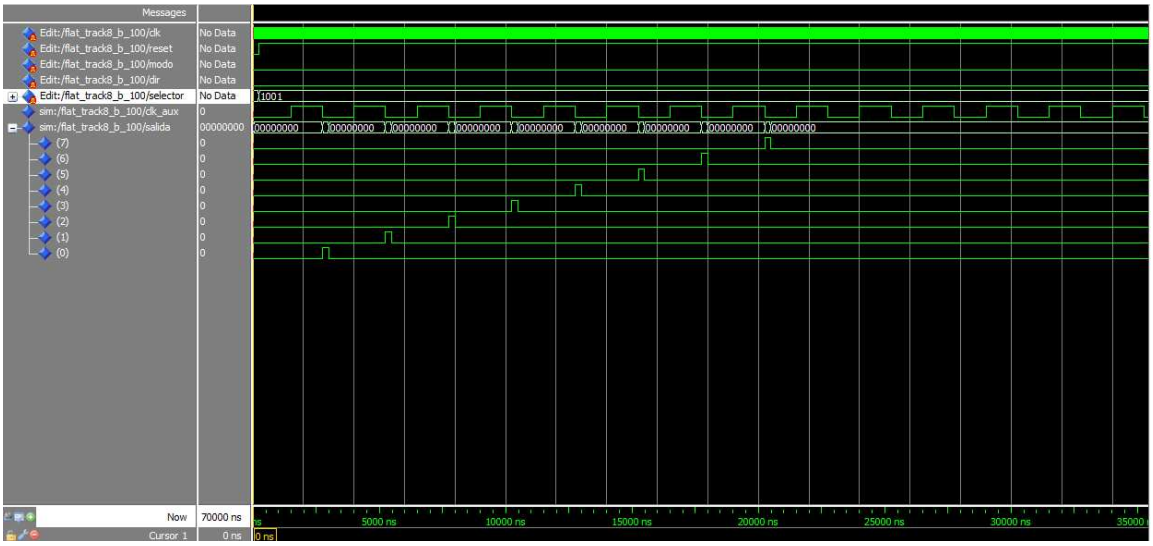
De la misma forma que se realizaron simulaciones a la arquitectura del Track-Sim, se simula el funcionamiento de la arquitectura creada para el Track-Flat utilizando el mismo software Modelsim®. Su creó también un archivo de entradas con las señales y sus diferentes valores que toman en determinados instantes de tiempo. La figura 3.29 muestra los resultados que se obtuvieron en los dos modos de operación: continuo y de una sola reproducción.



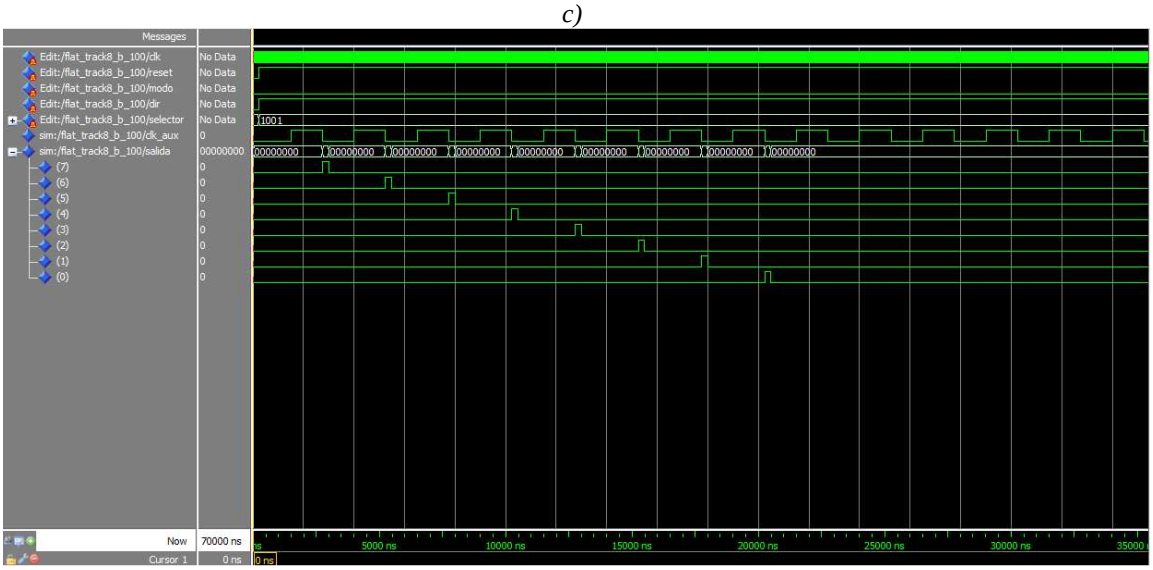
a)



b)



c)



d)

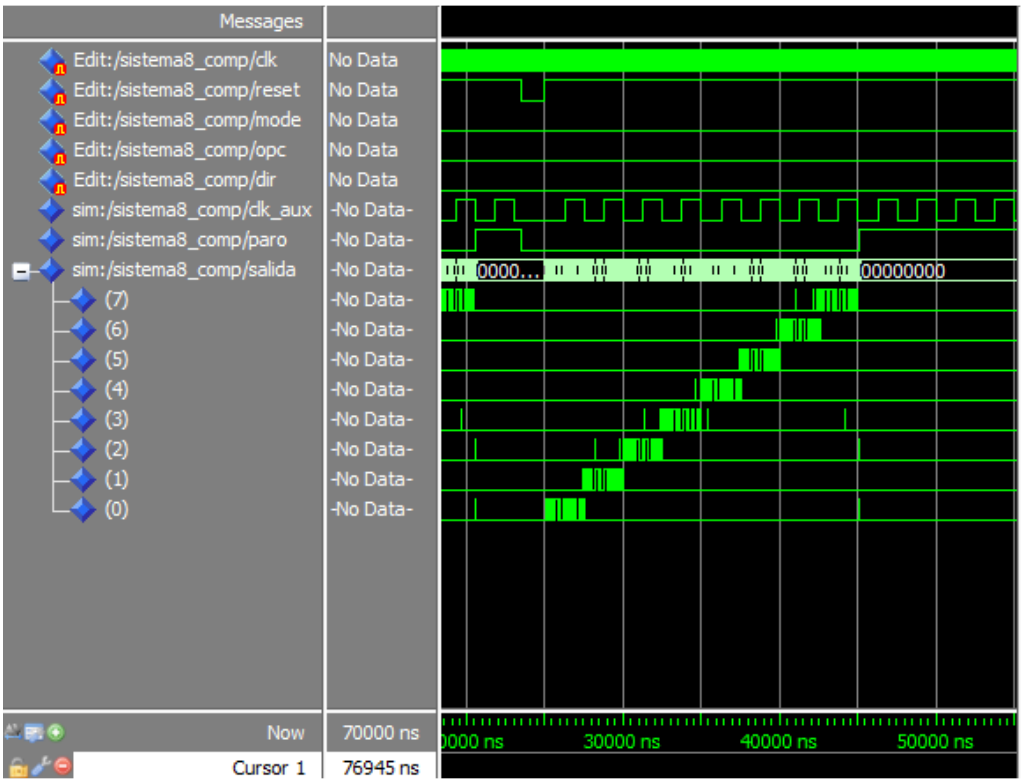
Figura 3.29. Simulaciones de la arquitectura implementada dentro del FPGA Spartan 3E de Xilinx. a) Simulación para el modo de operación continuo con pulsos que van del canal 1 al canal 8 (de izquierda a derecha). b) Simulación en modo continuo para pulsos que van del canal 8 al canal 1 (de derecha a izquierda). c) Simulación para modo de una sola reproducción con pulsos que van del canal 1 al canal 8 (de izquierda a derecha). d) Simulación de salida de una sola reproducción para pulsos que van del canal 8 al canal 1 (de derecha a izquierda). Las simulaciones mostradas son para 250 ns de ancho de pulso.

Implementación en dispositivos FPGA Actel y Xilinx

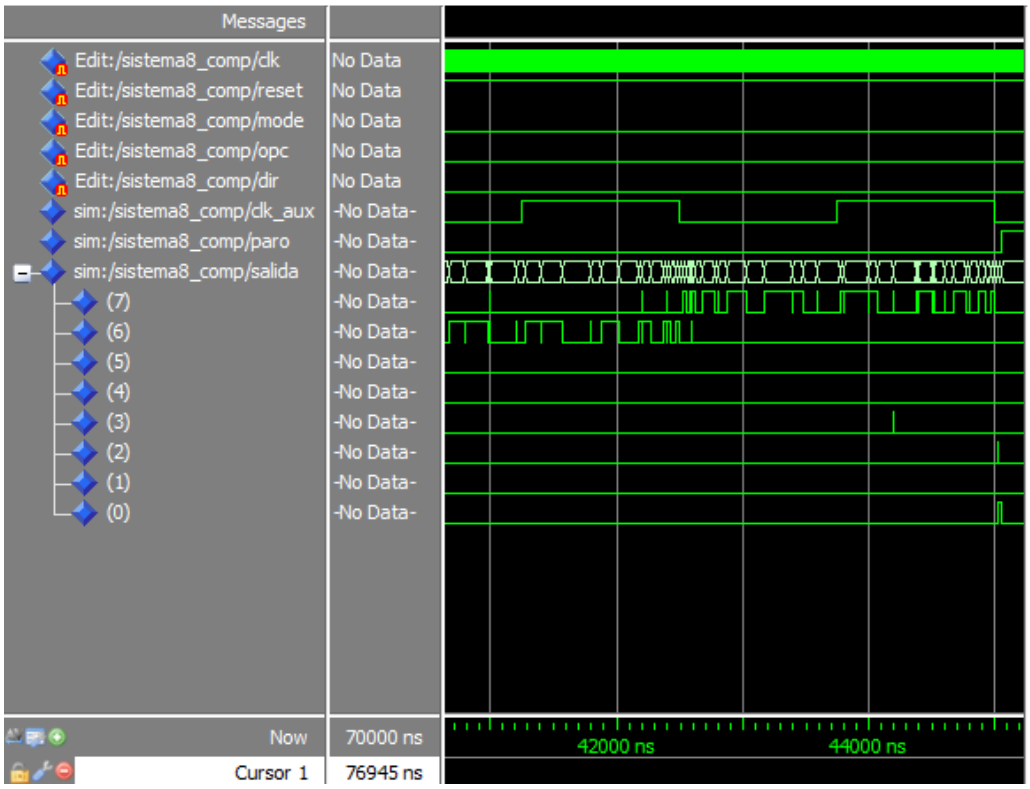
La implementación física de la arquitectura se realizó, primero, en el mismo FPGA destinado para el Track-Sim, la tarjeta prototipo ProASIC3/E Starter Kit y posteriormente en una tarjeta Basys2 con un FPGA Spartan 3E de Xilinx, que es la utilizada en el prototipo Track-Flat final. Ambos circuitos fueron probados y se realizaron mediciones como se muestra a continuación.

Implementación y pruebas en FPGA ProASIC3E A3PE1500 de Actel: tarjeta Starter Kit

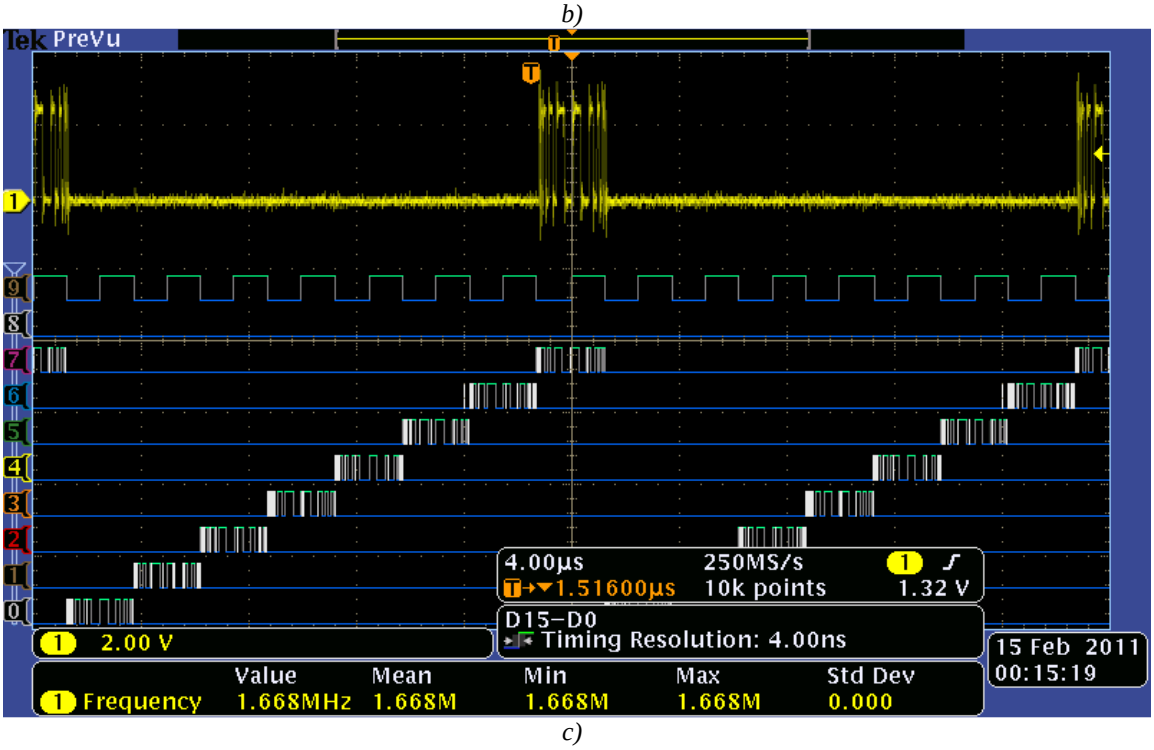
Para esta primer implementación se realizó la generación de trenes de pulsos en cada uno de los 8 canales, variando el ancho de pulso para cada canal conforme transcurría el tiempo de 2.5 µs en que estaba activo el canal en cuestión. De esta prueba se observó primeramente el comportamiento de la simulación en Modelsim y posteriormente, se utilizó un osciloscopio de señal mixta MSO 3032. La figura 3.30 muestra estas pruebas.



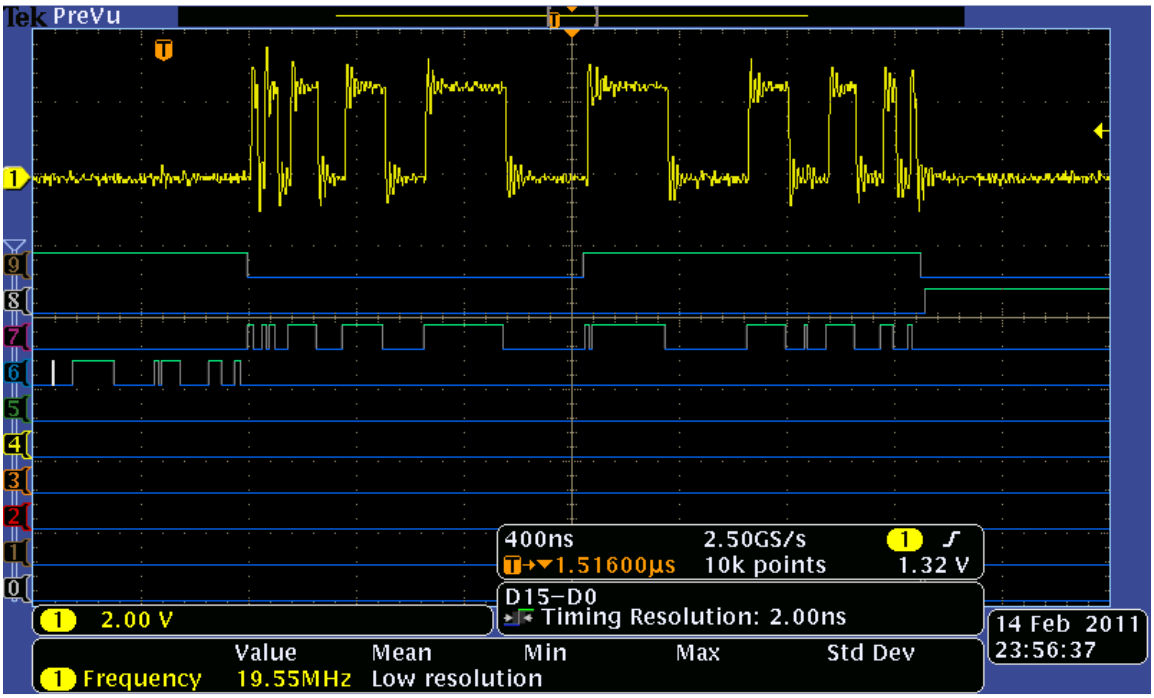
a)


Messages
Edit:/sistema8_comp/clk
Edit:/sistema8_comp/reset
Edit:/sistema8_comp/mode
Edit:/sistema8_comp/opc
Edit:/sistema8_comp/dir
sim:/sistema8_comp/clk_aux
sim:/sistema8_comp/paro
sim:/sistema8_comp/salida
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
No Data
-No Data-
Now
70000 ns
Cursor 1
76945 ns
42000 ns
44000 ns

b)


Tek PreVu
4.00µs
250MS/s
1.51600µs
10k points
1.32 V
2.00 V
D15–D0
Timing Resolution: 4.00ns
15 Feb 2011
00:15:19
Value
Mean
Min
Max
Std Dev
Frequency
1.668MHz
1.668M
1.668M
1.668M
0.000

c)



d)

Figura 3.30. Pruebas en FPGA ProASIC3E A3PE1500 de Actel. Se realizaron simulaciones en modo continuo en donde se observa la forma en que varía el ancho de pulso durante los 2.5 µs que dura la presencia de señal en cada canal como se aprecia en a) y b). En c) y d) se muestran las señales medidas en el osciloscopio, en específico, el canal 8, bajo las mismas condiciones que la simulación, donde es de destacar la gran cantidad de ruido montado sobre la señal que se obtiene de los pines del FPGA, así como los sobrepasos durante los cambios de estado.

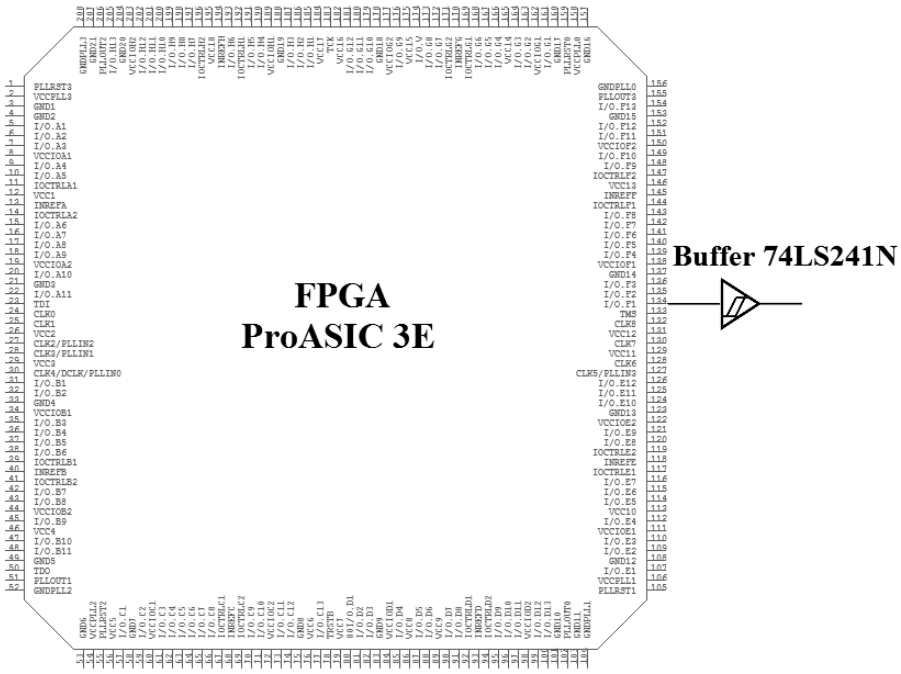
Se observa de éstas pruebas la presencia de *gltches* pero más importante, el ruido montado en la señal que sale de cada uno de los canales, como aparece en la señal del canal 8 en color amarillo en la parte superior de 3.34*d* y 3.34*e*. Además del ruido en las señales de cada canal, se ve la respuesta inherente al sistema que representan los transistores MOSFET de que se compone el FPGA, el cual, a primera vista, es muy semejante a la respuesta de los sistemas de segundo orden del tipo subamortiguados.[21]

[21] La forma de la respuesta a la salida de cada canal de este FPGA, y en general de todos los circuitos que operen con transistores en conmutación de sus estados (encendido y apagado o corte y saturación, respectivamente) produce este tipo de respuesta, la cual es intrínseca a la naturaleza del transistor MOSFET así como de otros transistores, debido a la dinámica del sistema, provocada por la constitución física de estos dispositivos, que son analizados en detalle de acuerdo a los diferentes modelos que han sido desarrollados para esto. El transistor MOSFET obedece a diferentes modelos, los cuales, dependen, para aplicaciones prácticas, principalmente de la frecuencia de operación, siendo por tal motivo, los más difundidos dentro de los cursos de electrónica. Otros efectos como los acoplamientos entre etapas y con otros dispositivos, y por ejemplo, para el caso de circuitos integrados, el de la escala de integración, en donde entran en juego limitaciones de composición, dimensiones y cercanía (acoplamientos) entre transistores, utilizan un modelado y diseño más complejos comenzando por el modelo desarrollado por

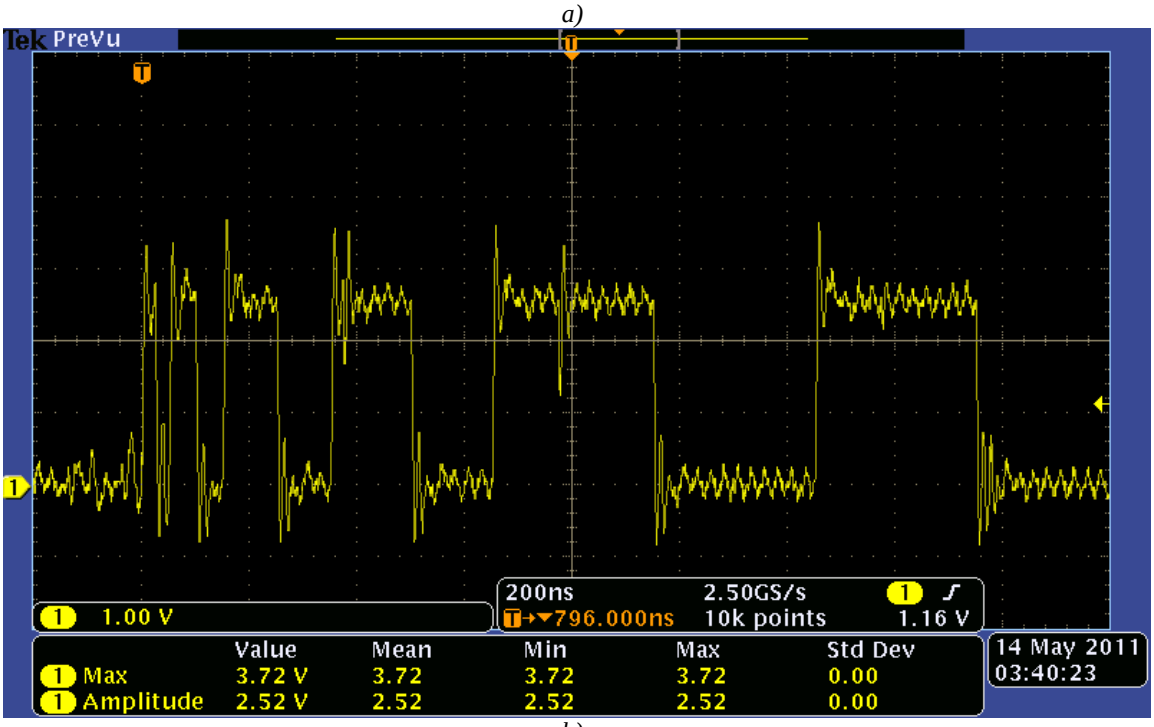
La forma en que se presentan las señales por parte del FPGA ProASIC3E A3PE1500 de Actel no es deseable debido a que las oscilaciones montadas en la señal que maneja a los LEDs puede traer inconvenientes al momento de variar la intensidad (esto será tratado con detalle más adelante en esta sección). Las oscilaciones de mayor tamaño serán amplificadas en la etapa del transistor MOSFET por lo que el LED "verá" directamente estas oscilaciones. Dado este hecho importante, se optó por reducir este comportamiento de las señales a la salida de este FPGA, en primera instancia, para eliminar las oscilaciones presentes en los cambios de estado y el ruido de alta frecuencia montado en la señal.

La primera modificación fue realizada para reducir las oscilaciones presentes en los cambios de estado, y consistió en utilizar un circuito que de alguna forma eliminara o atenuara en gran medida estas oscilaciones, tanto en tiempo como en amplitud. El circuito utilizado es un comparador de ventana con histéresis muy utilizado en diferentes aplicaciones, denominado *Schmitt Trigger* ó comparador Schmitt. A fin de implementar una solución robusta y compacta se decidió por utilizar un circuito integrado que brindará la configuración *Schmitt Trigger*, y que cumpliera además con los tiempos y anchos de pulso esperados a ser utilizados. Basado en estos requerimientos, se seleccionó un circuito *buffer* con comparador Schmitt integrado a su entrada, con lo cual se ganó además en el incremento en capacidad de corriente por parte de cada canal. El circuito integrado utilizado es el SN74LS241N de Texas Instrument. Este *buffer* de ocho entradas y salidas es capaz de alcanzar anchos de pulso de un mínimo de 10 ns (medido en laboratorio) con retrasos entre entrada y salida de 10 ns y capacidad de corriente en nivel lógico de alto de 15 mA y en nivel lógico bajo de 24 mA. La figura 3.31 muestra la implementación de este circuito y las mejoras en la forma de onda. Las mediciones fueron realizadas antes y después del circuito buffer con el osciloscopio MSO 3032.

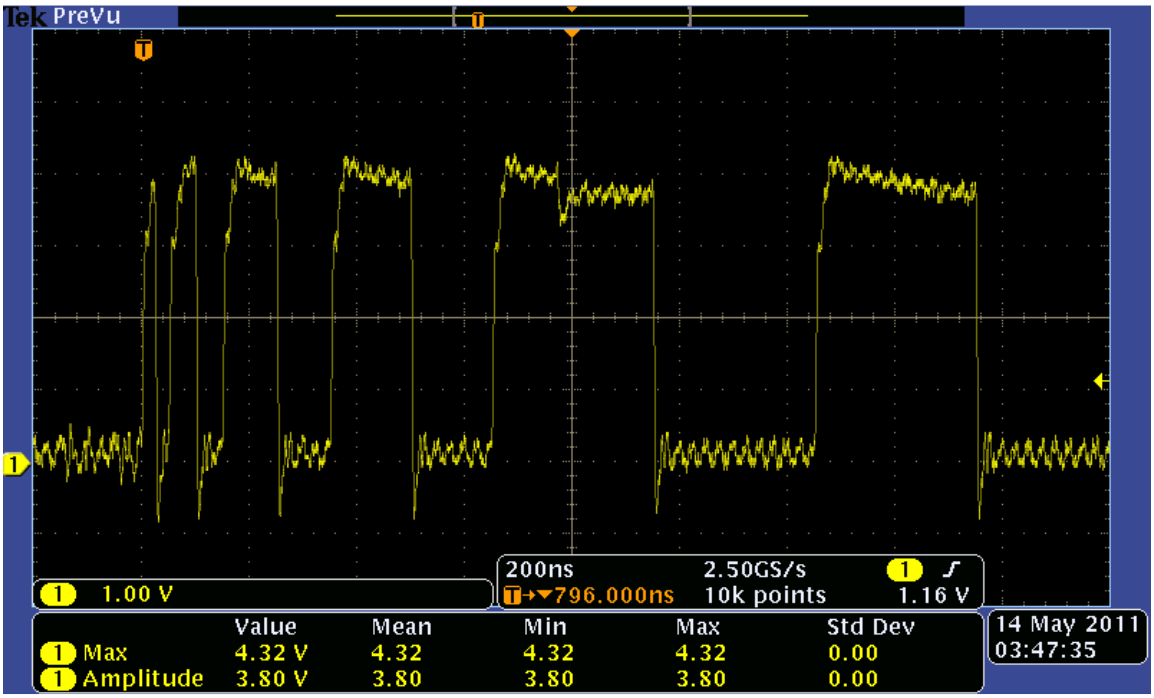
Shockely hasta modelos más recientes como el desarrollado por Phillips y la Universidad de Pensilvania denominado PSP (para más detalles ver [19], [20], [21], [22], [23], [24]).


FPGA
ProASIC 3E
Buffer 74LS241N

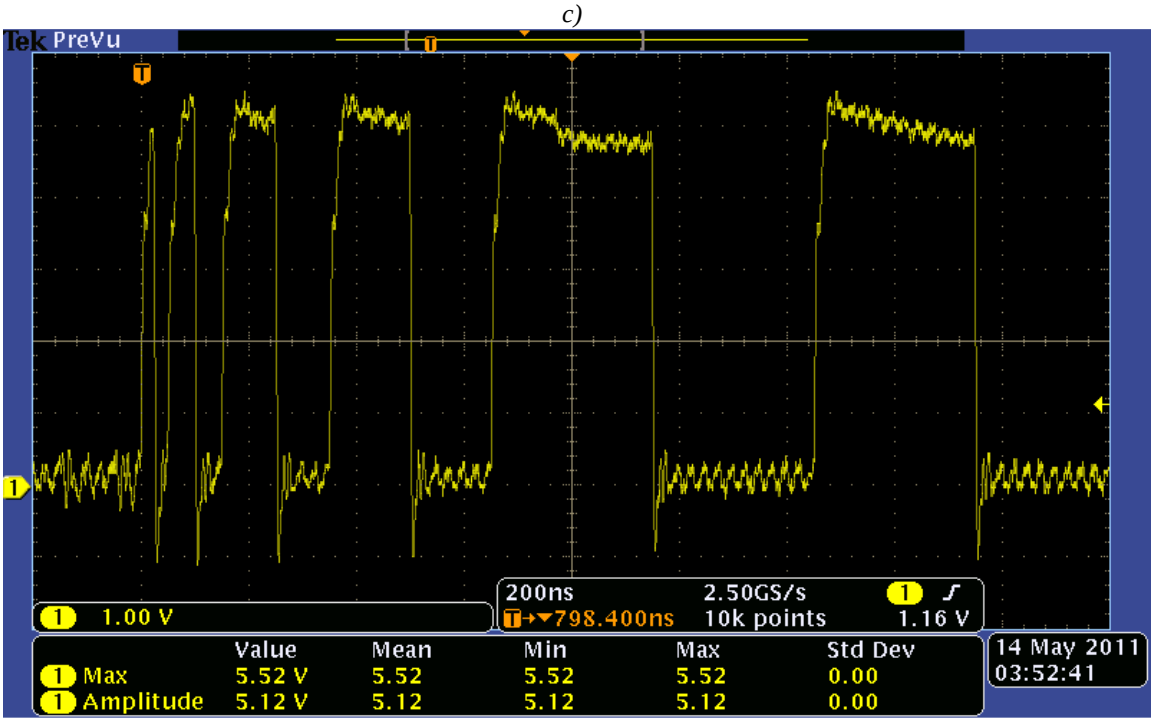
a)


Tek PreVu
1.00 V
200ns
2.50GS/s
796.000ns
10k points
1.16 V
Value
Mean
Min
Max
Std Dev
Max
3.72 V
3.72
3.72
3.72
0.00
Amplitude
2.52 V
2.52
2.52
2.52
0.00
14 May 2011
03:40:23

b)



c)

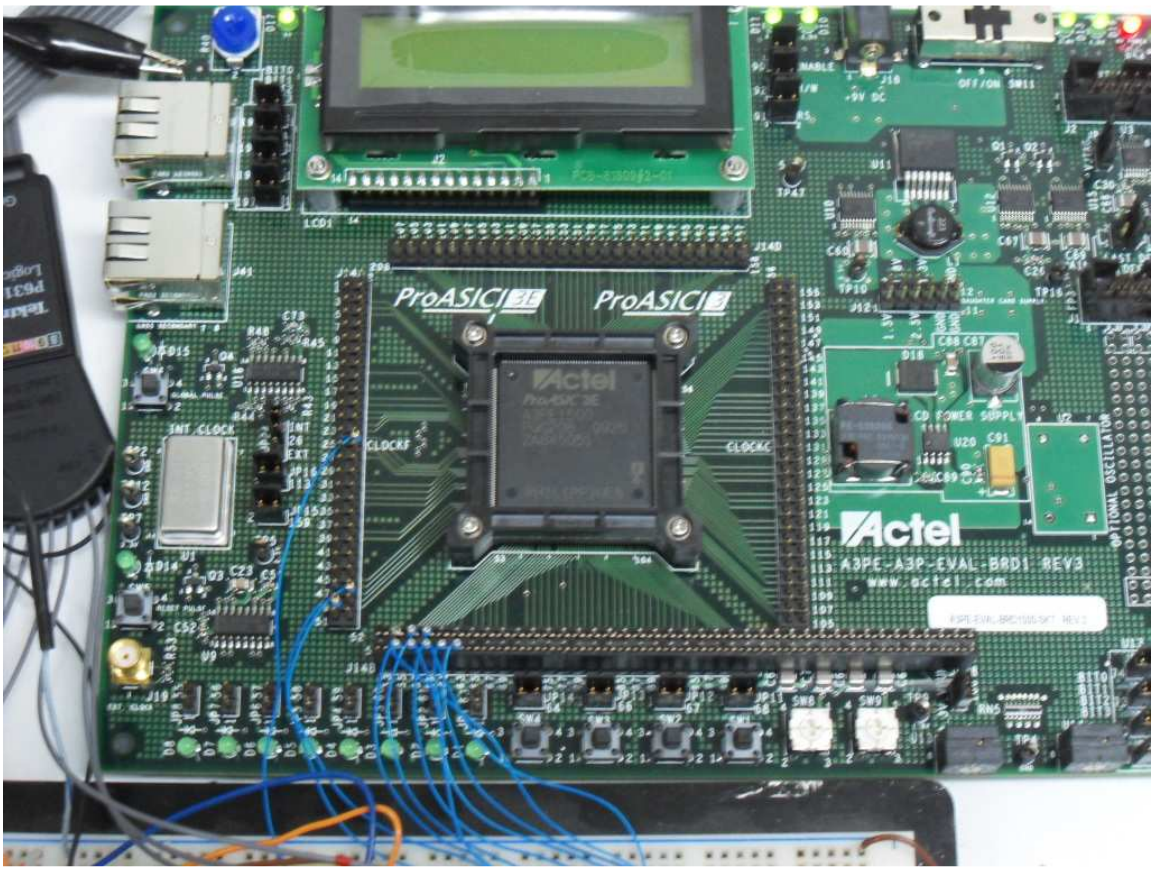


d)

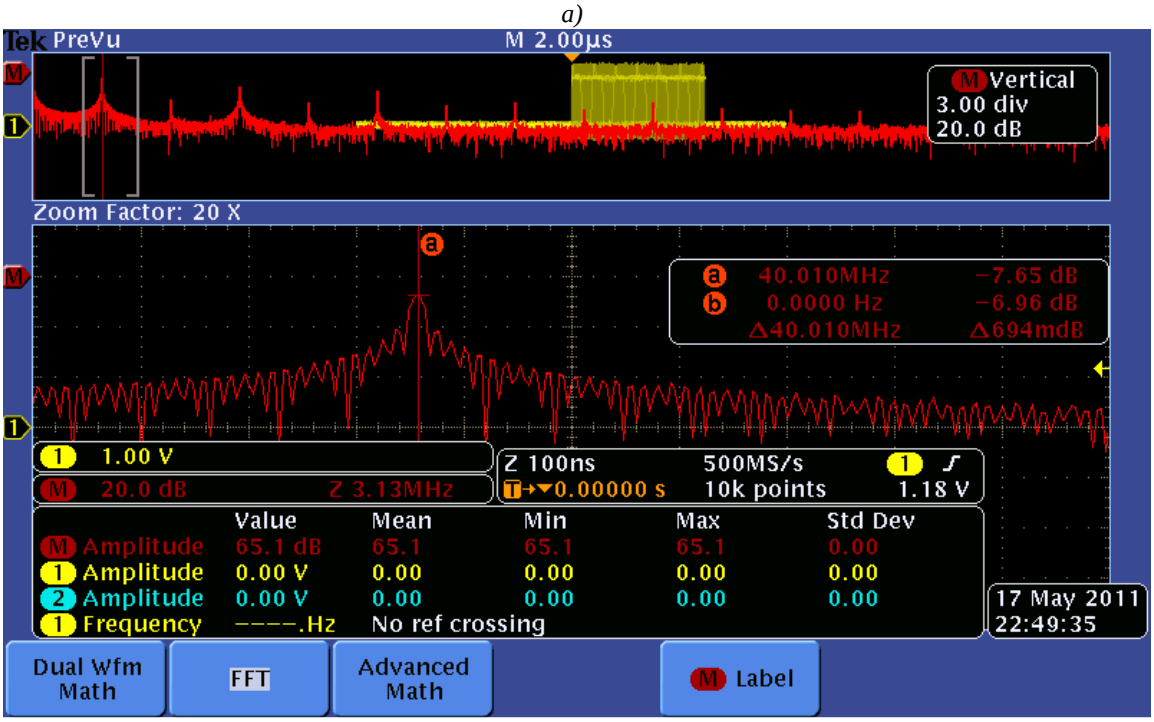
Figura 3.31. Implementación de circuito buffer Schmitt Ttrigger para eliminar oscilaciones (comportamiento subamortiguado) en las salidas del FPGA ProASIC3E A3PE1500 de Actel. a) Configuración del circuito. b) Forma de onda medida directamente de un canal (pin) del FPGA antes del circuito buffer SN74LS241N. c) Salida del circuito buffer Schmit Trigger SN74LS241N polarizado a 5 V y sin carga a su salida (sólo osciloscopio). d) Salida del mismo circuito buffer pero polarizado con 6 V.

Como se observa de la implementación con el circuito *buffer*, las oscilaciones en los cambios de estado fueron reducidas o eliminadas prácticamente, sin embrago aún persistió el ruido de alta frecuencia montado en la señal. Para determinar que ocasiona este ruido se realizaron mediciones con el osciloscopio MSO 3032 a través de FFT, en donde se determinó, como lo muestra la figura 3.32, que la fuente de estas oscilaciones es el reloj mismo de 40 MHz que incorpora la tarjeta ProASIC3E. La eliminación de este ruido de alta frecuencia resulta más complicado que la eliminación de las oscilaciones mediante el circuito *Schmitt Trigger*, pues utilizar un filtro que logre quitar la mayor parte de estas oscilaciones resulta en la práctica complejo, dado que el orden requerido debe ser elevado si se desea un mejor resultado, y aunado al gran número de canales a utilizar en el diseño final del Track-Sim se convierte en una solución inviable. La mejor solución es ir directamente a la fuente que produce el problema, esto es, el reloj de la tarjeta.

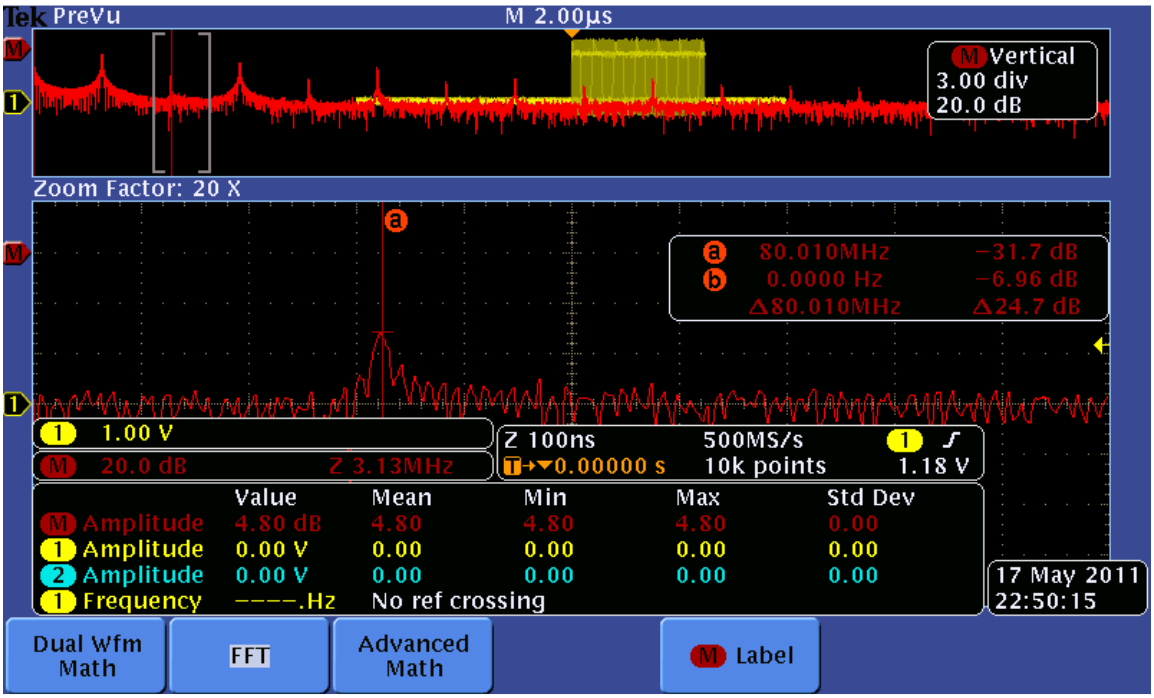
Se cambió la señal del reloj por una fuente externa conocida utilizando el generador de funciones AFG3252 con una señal senoidal, utilizando una amplitud de 0 a 3.3V, a los mismos 40 MHz, con lo que se observó que las oscilaciones de alta frecuencia en la señal de salida de los canales del FPGA desaparecieron. La conclusión es que cambiando la señal de reloj por una señal más “limpia” se elimina el problema del ruido de alta frecuencia montado en las señales a la salida del FPGA.


ProASIC3E
ProASIC3
Actel
A3PE-A3P-EVAL-BRD1 REV3
www.actel.com

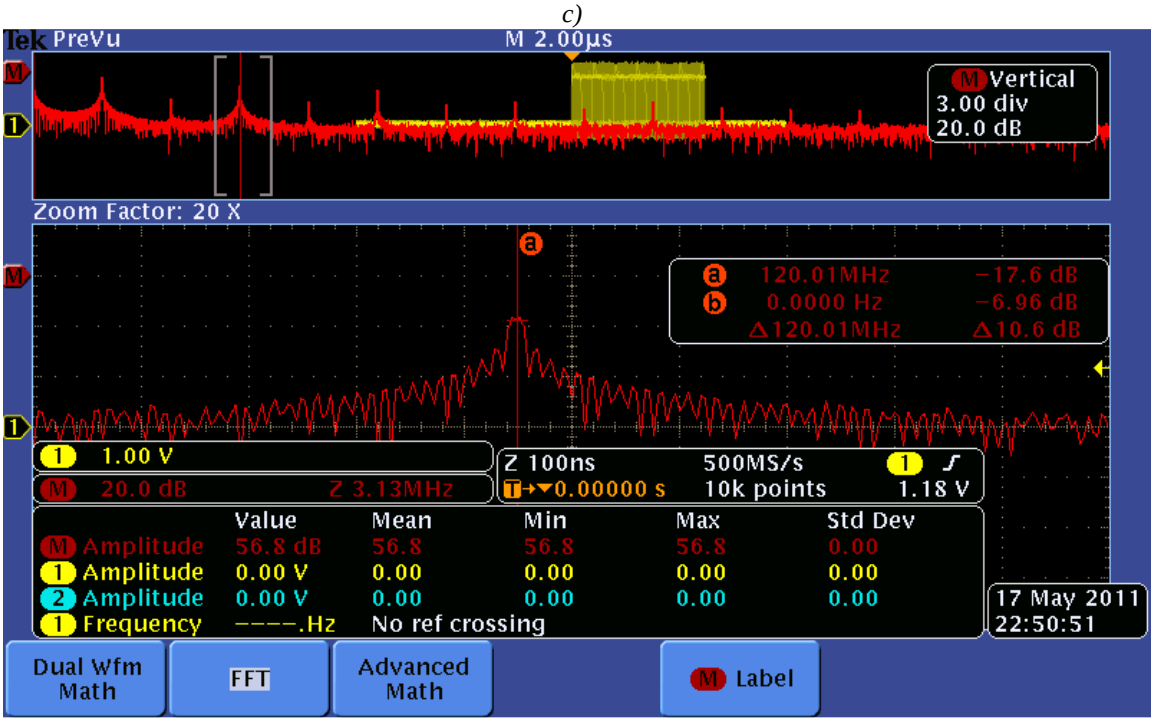
a)


Tek PreVu
M 2.00µs
M Vertical
3.00 div
20.0 dB
Zoom Factor: 20 X
a 40.010MHz −7.65 dB
b 0.0000 Hz −6.96 dB
Δ40.010MHz Δ694mdB
1 1.00 V
M 20.0 dB Z 3.13MHz
Z 100ns 500MS/s
T 0.00000 s 10k points
1.18 V
Value Mean Min Max Std Dev
M Amplitude 65.1 dB 65.1 65.1 65.1 0.00
1 Amplitude 0.00 V 0.00 0.00 0.00 0.00
2 Amplitude 0.00 V 0.00 0.00 0.00 0.00
1 Frequency -----.Hz No ref crossing
17 May 2011
22:49:35
Dual Wfm Math
FFT
Advanced Math
M Label

b)



c)



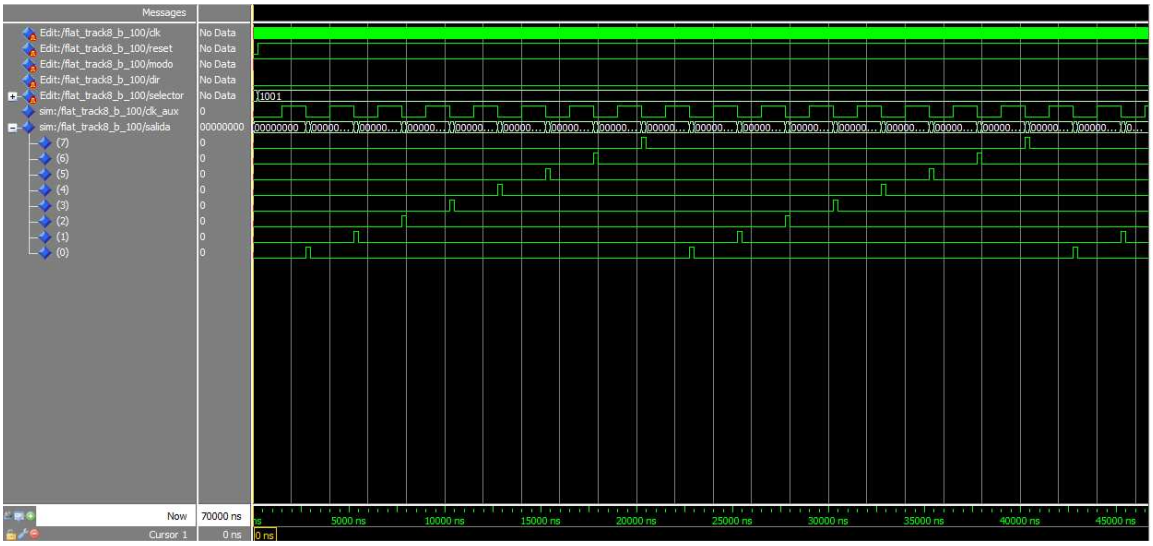
d)

Figura 3.32. Medición del ruido de alta frecuencia a la salida de los pines del FPGA A3PE1500 de Actel. a) Tarjeta ProASIC Starter Kit de Actel con FPGA A3PE1500 y canales conectados para su operación y medición mediante conexiones tipo wire wrap. b) Espectro de frecuencia de la señal del canal 8 en los pines de salida del FPGA con la componente fundamental de 40 MHz. c) Segundo armónico de 80 MHz. d) Tercer armónico de 120 MHz.

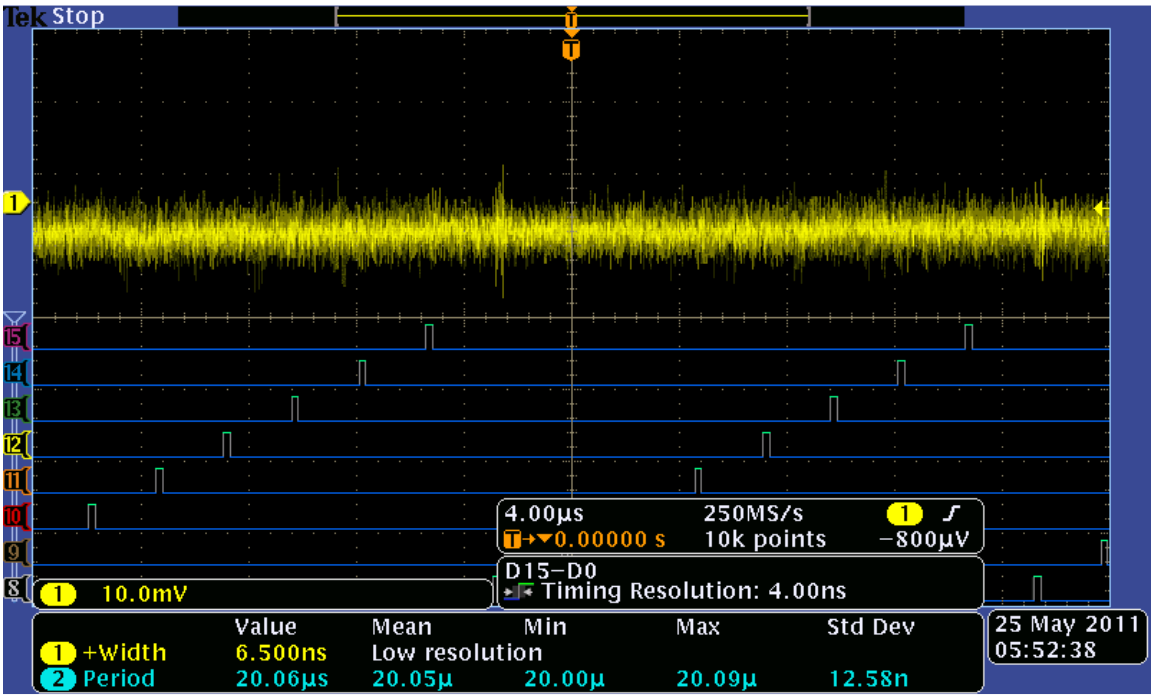
Implementación en tarjeta Basys2 con FPGA Spartan 3E de Xilinx

La implementación del prototipo Track-Flat se realizó en un dispositivo FPGA de menor capacidad y tamaño dado lo reducido que resulta. Se utiliza para este fin la tarjeta Basys2 que incorpora un FPGA Spartan3E de Xilinx de 100 000 compuertas en encapsulado TQ144. La tarjeta es de dimensiones bastante reducidas lo que permite que sea fácilmente incorporada en un gabinete que alberga a otras tarjetas con circuitos de control, volviendo al prototipo compacto y modular.

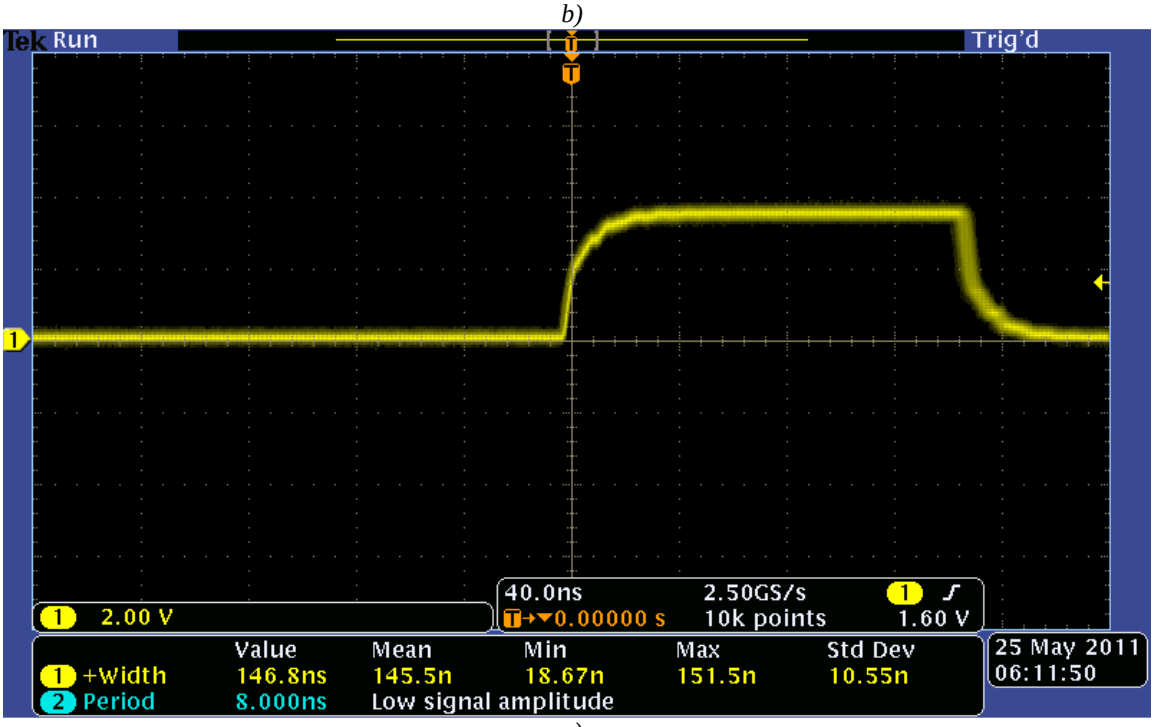
Al igual que con la tarjeta ProASIC Starter Kit, se realizaron simulaciones así como mediciones con osciloscopio para observar el comportamiento. Esto se puede observar en la figura 3.33, donde se aprecia que a diferencia de la tarjeta con el FPGA A3PE1500 de Actel, el FPGA Spartan3E presenta señales a su salida más “limpias” y consistentes de acuerdo a lo esperado. Tampoco se presentan *glitches* y no se presentan problemas con la señal de reloj. Aún y cuando las señales provenientes del FPGA son mejores que las obtenidas con la tarjeta de Actel, la utilización del circuito buffer *Schmit Trigger*, además de proporcionar más corriente a cada canal, sirve como etapa de acoplamiento entre los transistores MOSFET y el FPGA, teniéndose también la oportunidad de obtener una amplitud mayor a los 3.3 V máximos que se brindan.



a)



b)



c)

Figura 3.33. Implementación de la arquitectura del prototipo Track-Flat en su tarjeta destino Basys2 con un dispositivo Spartan3E de Xilinx. a) Simulación en Modelsim para modo continuo. b) Medición en modo continuo en osciloscopio. c) Pulso de 150 ns de ancho medido directamente a la salida de la tarjeta Basys2.

Otra ventaja de utilizar esta tarjeta es que contiene un oscilador con 3 diferentes frecuencias de operación, las cuales son seleccionables mediante un jumper para los valores de 25, 50 y

100 MHz. El valor de frecuencia a utilizar es de 100 MHz, con lo que se pueden obtener en principio anchos de pulso mínimos de 10 ns.

Diseño del circuito de encendido

Circuito de encendido a base de transistor MOSFET

El control de intensidad se realizará utilizando la emisión por duración de encendido de LEDs ó PWV, mediante un circuito a base de *switches* conformados por transistores MOSFET y/o buffer. El circuito a base de MOSFET tiene la configuración que se mostró en la figura 3.18 pero con variaciones durante las pruebas físicas del circuito, en primer lugar, del valor de la resistencia entre la fuente de polarización y el ánodo del LED, para encontrar el valor óptimo de acuerdo al voltaje de polarización a utilizar y a la respuesta obtenida en la PMT. Otra modificación que será comentada es la de un arreglo atenuador de la intensidad del LED en el circuito sin utilizar medios ópticos, lo cual será referido más adelante como una posible solución a la saturación en la respuesta de la PMT.

La primer verificación sobre el comportamiento del circuito del encendido se realizó mediante una simulación del circuito de la figura 3.34*a*, utilizando el software basado en SPICE Multisim®. Esta simulación muestra un primer acercamiento al comportamiento del circuito. Se anexó la etapa del buffer con comparador Schmitt entre el generador de pulsos y la terminal de compuerta del transistor. Las condiciones de las señales de pruebas del generador fueron las siguientes:

- Amplitud entre 0 y 3.3 volts.
- *Rise time* de 2.5 ns y fall time de 2.5 ns (de las características del generador AFG3252 utilizado en la implementación física).
- Ancho de pulso de 20 ns.
- Señal tren de pulsos con periodo de 2.5 µs.

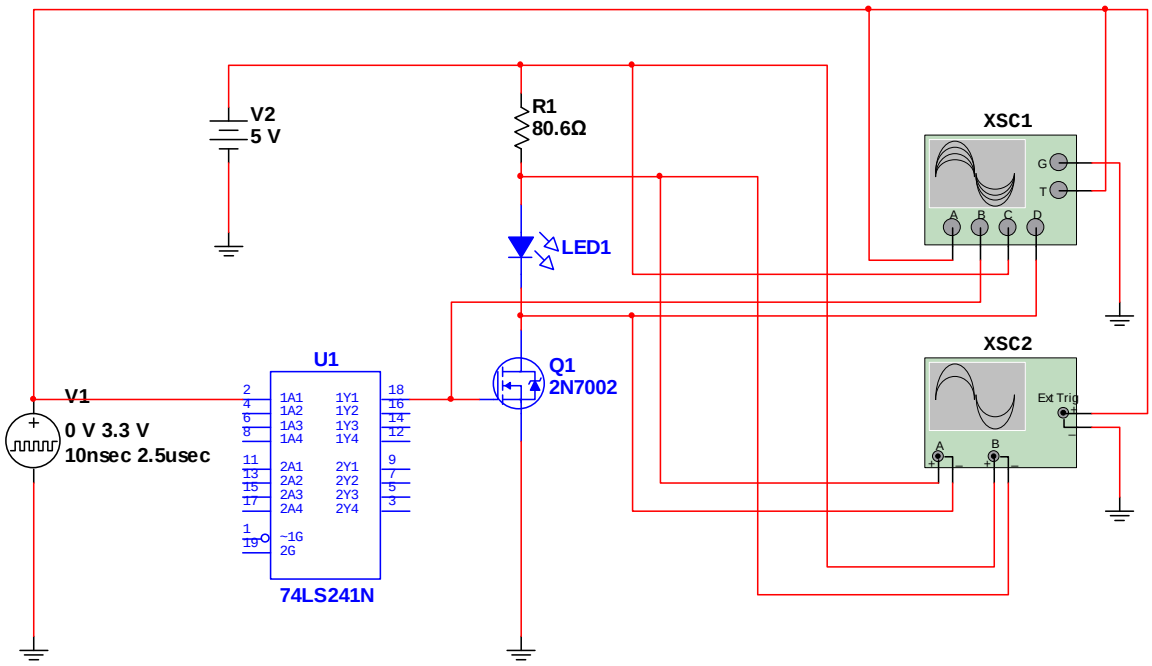
Se miden los voltajes en el LED (V_{LED}), la caída en la resistencia (V_R) y la diferencia de potencial entre drenaje y fuente (V_{DS}). Se obtienen los resultados mostrados en 3.34, de donde lo más importante a resaltar es ver que el LED en conjunto con el MOSFET presentan un tiempo de encendido un tanto mayor que el enviado desde el generador de funciones, lo cual se debe a los tiempos de encendido (tiempos de levantamiento y decaimiento) de ambos dispositivos en conjunto, mismos que a su vez, son función de la corriente que circula por ambos dispositivos, así como del voltaje aplicado mediante la fuente de polarización de todo el circuito (5 V en este caso) y el pulso de encendido aplicado entre compuerta y fuente (V_{GS}). [22] De las simulaciones realizadas también resulta importante destacar que el dispositivo es capaz de encender y apagarse en tiempos bastante menores que un GTU, lo que indica que es posible encender y apagar (forma de onda en azul) cada LED del orden de 50 o más veces dentro del intervalo temporal de un GTU, por lo que en principio, esperando las pruebas del número de fotones emitidos por el LED,[23] es posible cumplir con los requisitos ya comentados aquí para un rango dinámico de ~300 fotones/GTU.

Por otra parte, el valor de la resistencia de 80.6 Ω se seleccionó para manejar un valor de corriente de ~20 mA, que es el valor nominal en que se suele operar a este LED. [24]

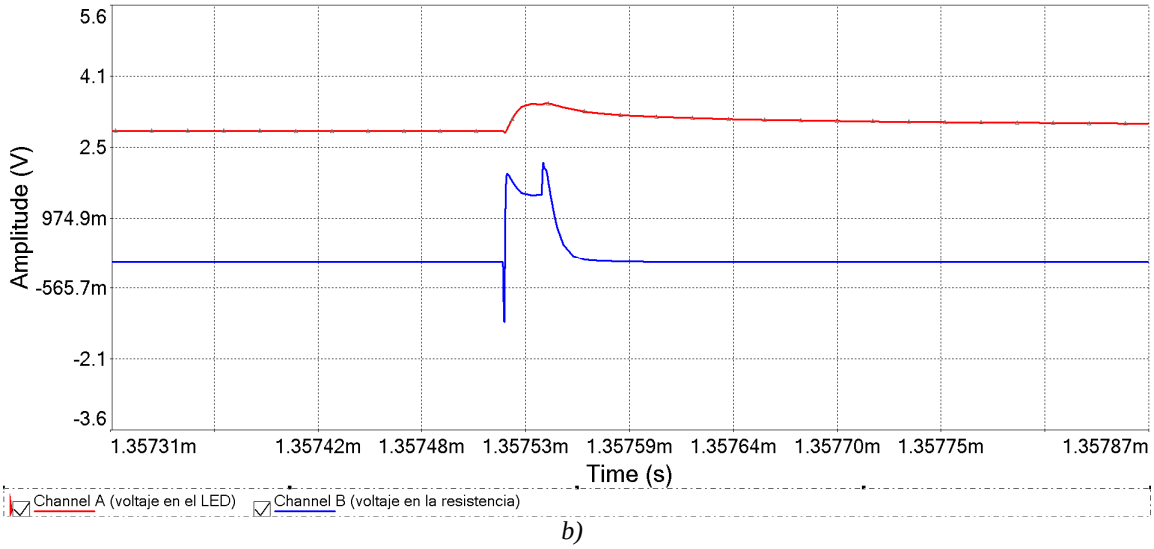
[22] Como se mencionó al inicio de este capítulo la capacitancia de difusión (presente en polarización directa de un LED) es función de la corriente en el dispositivo, lo cual a su vez tendrá efecto en el tiempo de encendido del LED. Para el caso del MOSFET, el tiempo de encendido y apagado del dispositivo depende de la diferencia de potencial aplicada entre las terminales *gate* y *source* ó V_{GS}, la cual actúa directamente en la apertura del canal, así como de las dimensiones físicas del dispositivo como longitud y ancho del canal, tamaño del canal generado en MOSFETs de enriquecimiento como el utilizado aquí, espesor de la capa de óxido metálico y otros parámetros (para más detalles consultar [27]).

[23] Resultados de pruebas mostradas a detalle en César Tavera Ruiz, *et al*., 2011.

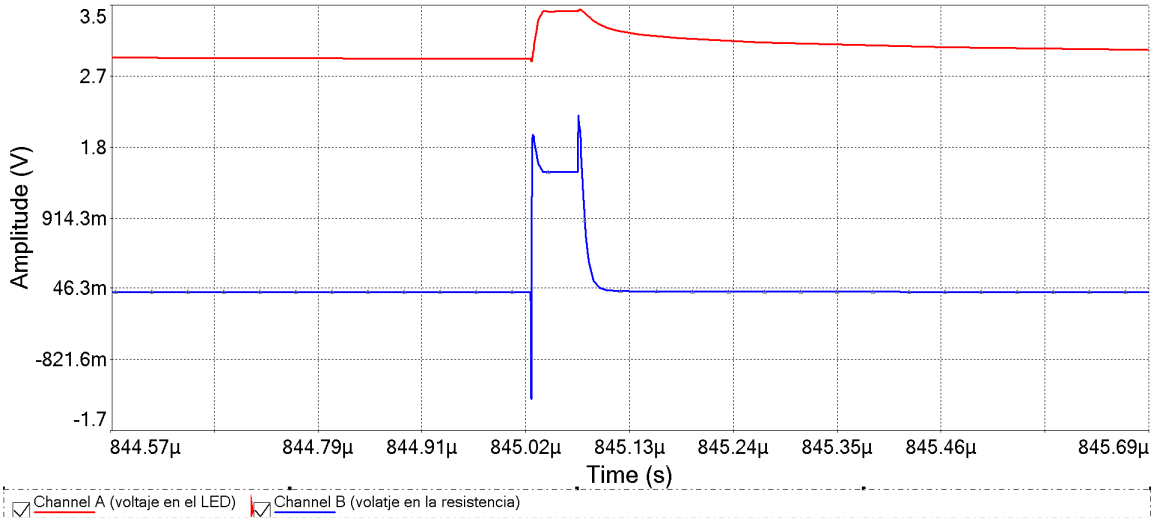
[24] Pueden utilizarse valores de corriente en directa de más de 20 mA (hasta 30 mA como máximo para este LED) sin embargo, es posible dañar a los LEDs con corrientes por encima de su valor máximo, además de que el valor de 20 mA es comúnmente utilizado en dispositivos LED y el reportado por el fabricante de este LED para sus pruebas de intensidad relativa (ver datasheet del LED utilizado VAOL-5EUV0T4, que se ubica dentro del apéndice 1).


V2
5 V
R1
80.6Ω
XSC1
G
T
A
B
C
D
LED1
U1
Q1
2N7002
XSC2
Ext Trig
V1
0 V 3.3 V
10nsec 2.5usec
74LS241N

a)


5.6
4.1
2.5
974.9m
-565.7m
-2.1
-3.6
Amplitude (V)
1.35731m
1.35742m
1.35748m
1.35753m
1.35759m
1.35764m
1.35770m
1.35775m
1.35787m
Time (s)
Channel A (voltaje en el LED)
Channel B (voltaje en la resistencia)

b)



c)

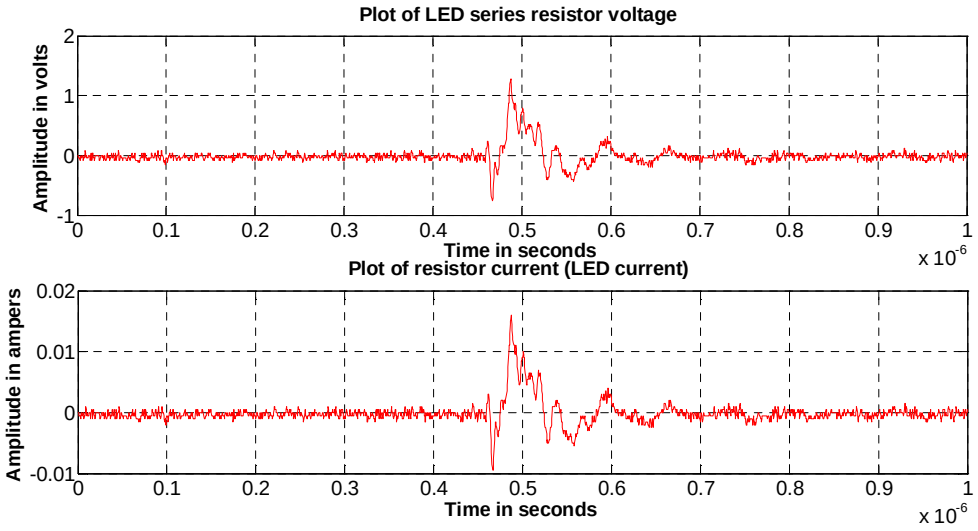
Figura 3.34. Simulaciones de conmutación del LED con el circuito a base de MOSFET. a) Configuración base del circuito con R1 = 80.6 Ω. El circuito SN74LS241N se polariza a 5V. b) Resultados obtenidos para simulación con ancho de pulso de 20 ns, la señal en rojo es el voltaje en el LED, mientras que la señal en azul es el voltaje en la resistencia R1. c) Resultado de simulación para ancho de pulso de 50 ns. La señal en rojo (parte superior de 3.34c) es el voltaje en el LED, mientras que la señal en azul (parte baja) es la diferencia de potencial en la resistencia R1.

Ahora se muestra en la figura 3.35 los resultados de las pruebas físicas del circuito, en donde se utilizó una resistencia variable (potenciómetro) ajustada a un valor de 80.1 Ω, de tal forma que se obtenga, como máximo, un valor alrededor de la corriente nominal del LED de 20 mA. Las mediciones aquí se realizaron con un osciloscopio digital TDS 2024 y una fuente de alimentación programable modelo Array 3631A operando a 5V DC. Al mismo tiempo que se realizaron estas mediciones del LED en el circuito, se conectó el LED apuntando hacia una PMT de un pixel con el fin de obtener su comportamiento en intensidad. Este experimento con la PMT será revisado posteriormente ya que se obtienen resultados muy significativos así como problemas en la respuesta de la PMT debidas a saturación de la misma.

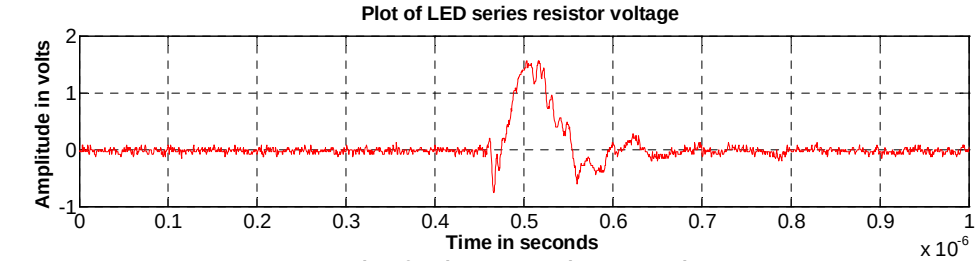
Del comportamiento de encendido del circuito de la figura 3.35 *a* se puede observar que es semejante a lo obtenido en la simulación, sin embargo, los anchos de pulso alcanzados aquí son aún menores (el ancho que se mide es el ancho a media altura o HWMH) siendo posible llegar hasta 10 ns sin problemas, sin embargo, para la intensidad detectada en la PMT, los anchos mínimos presentan ciertos inconvenientes como se verá más adelante. Además, la forma de los pulsos medidos es más distorsionada que en las simulaciones, por

lo que se tienen presentes ciertas oscilaciones en los cambios abruptos de la señal, las cuales se amortiguan y desaparecen. Este comportamiento de oscilaciones será medido al utilizar el dispositivo MAPMT para determinar cómo afecta a la intensidad del LED, ya que si tiene impacto representativo en la misma, se deberá encontrar la forma de eliminarla, atenuarla al máximo o cuantificarse para contemplar estos efectos en la calibración.

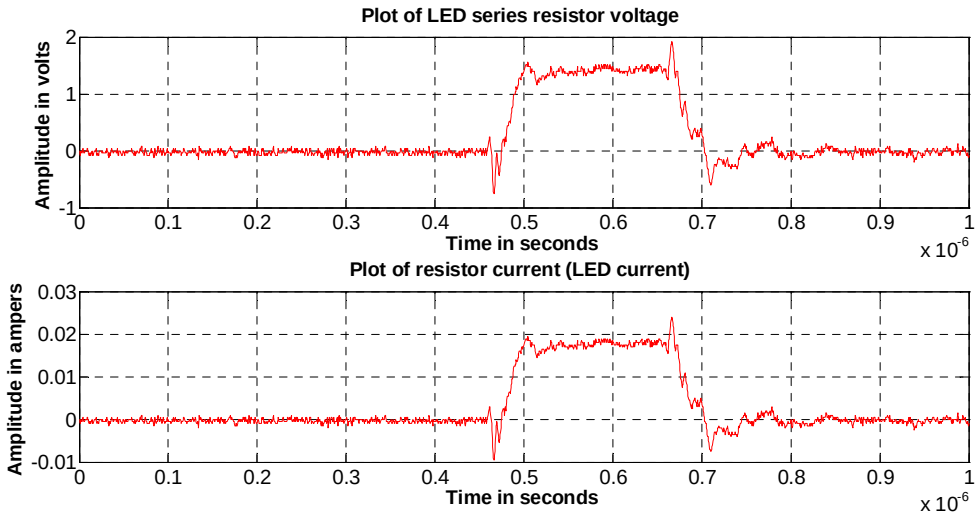
Por otra parte, la forma de onda del voltaje en la resistencia al ser dividida entre el valor utilizado de 80.1 Ω proporciona la corriente del LED. Esta forma de obtener la corriente será utilizada para determinar el comportamiento de la variación en intensidad con respecto a los anchos de pulso en análisis posteriores.



a)



b)



c)

Figura 3.35. Pruebas en el circuito de encendido de LEDs. a) Forma de onda adquirida con el osciloscopio TDS 2024 en la resistencia R1 (ver figura 3.34a) de 80.1 Ω, para un pulso de 20ns proveniente desde el generador de funciones AFG 3252. b) Forma de onda adquirida en R1 para un pulso del generador de 50 ns. c) Pulso de voltaje y corriente en resistencia serie del LED para 200ns de ancho de pulso.

Comportamiento en intensidad del LED con el circuito de encendido a base de MOSFET

Al realizar las primeras pruebas al circuito de encendido de LEDs, puede verse a simple vista que, al modificar el tiempo de encendido, la intensidad del LED cambia (a mayor tiempo de encendido mayor intensidad) por lo que se decidió medir directamente la intensidad para cuantificar su comportamiento. Para esto se utilizó la PMT multi-ánodo (MAPMT) de 64 pixeles, sin embargo, se observó que después de determinado valor de

ancho de pulso, la respuesta de la MAPMT tiende a no cambiar y la corriente que consume crece, pareciendo a primera instancia que se satura debido a que la luz producida por el LED es demasiada. Esto, en principio, vuelve difícil de cuantificar la intensidad del LED además de que es riesgoso aumentar el ancho de pulso de encendido e incrementar por consiguiente la corriente de la MAPMT a valores que puedan provocar un daño permanente en este costoso dispositivo. [25] Debido a esto, se decidió comparar y utilizar primeramente otro dispositivo PMT, para verificar el comportamiento del mismo circuito de encendido, y por otro lado, no correr riesgo de dañar a la MAPMT.[26]

Utilizando entonces la PMT de un solo pixel, y la configuración (con R1 = 80.1 Ω) del circuito de la figura 3.34, se varió el ancho con el generador AFG3252, desde 20 ns hasta 200 ns, en pasos de 10 ns. Se hizo un primer levantamiento de datos utilizando el osciloscopio TDS 2024C de la corriente del LED y de la respuesta de la PMT de acuerdo al arreglo de la figura 3.36*a*, en donde se adquirió un pulso tanto del voltaje en R1 como de la salida de la PMT, por cada ancho de pulso utilizado. La forma de cuantificar y obtener la relación de intensidad en el LED mediante el circuito y el método de control de la emisión por emisión instantánea (PWV) se realizó al comparar la carga medida en la PMT respecto de la carga inyectada al LED, integrando los pulsos obtenidos de salida de la PMT y de corriente en el LED.

Para obtener la carga medida en la PMT se realizó el siguiente procedimiento. Primero se conectó una carga de 50 Ω en las terminales de salida de la PMT y se midió y adquirió el pulso de voltaje en las mismas. Una vez adquiridos los datos de voltaje en función del tiempo a la salida de la PMT, en el análisis de los datos se divide cada pulso de voltaje (correspondiente a un ancho de pulso diferente) entre el valor de la carga de 50 Ω para obtener la corriente en función del tiempo. Posteriormente se encuentra el área bajo la

[25] En el peor de los casos puede quemarse la MAPMT perdiéndose el dispositivo definitivamente, sin embargo, durante la primera prueba, se limitó la corriente en la fuente de alto voltaje de modo que al incrementarse ésta a determinado valor, el voltaje de alimentación se reducía, protegiendo así al dispositivo MAPMT.

[26] Esta MAPMT fue utilizada en un principio en el experimento BATATA y fue reasignada para el grupo de trabajo del instrumento Track-Sim para su uso, sin embargo, se indicó que produjo un comportamiento un tanto "indeseado" dentro del otro experimento (BATATA), pero en principio opera de forma "normal".

curva para transformar la corriente en carga eléctrica. En lo que respecta a la corriente del LED, la forma de onda de cada pulso medido en la resistencia R1 se divide por el valor utilizado de 80.1 Ω para encontrar la forma de onda del pulso de corriente, y posteriormente se integra éste para encontrar la carga inyectada al LED.

La carga que brinda la PMT no representa la intensidad que emite el LED ni la que recibe la PMT (en este caso el número de fotones que arriban al fotocátodo) pues todavía debe realizarse un escalado que depende de parámetros como la eficiencia cuántica y la eficiencia de conteo de la PMT, respuesta dependiente de la longitud de onda de la PMT (más estrictamente, la convolución de la curva de respuesta del LED en longitud de onda con la curva de respuesta de la PMT, la cual a su vez, también dependiente de la longitud de onda), así como el valor de carga del *single-photoelectron* o valor de carga correspondiente a la detección de un solo fotón, con lo cual, sería entonces posible conocer la intensidad de la fuente LED (emisión del número de fotones de acuerdo al ancho de pulso utilizado) y la intensidad detectada por la PMT (número de fotones), sin embargo, todos estos parámetros sólo escalan al resultado de la carga medida en la PMT, pero lo importante por ahora, es determinar la forma en que varía la intensidad respecto del control utilizado para la misma, buscando siempre que ésta sea realice de forma lineal. [27]

Es importante señalar que previo al tratamiento de los datos provenientes de la PMT y de la resistencia del circuito, se realiza un corrimiento para ponerlos en fase e identificar que la salida en carga de la PMT corresponde a los tiempos de emisión del LED[28], calculándose, como primer aproximación, la carga en la PMT correspondiente al tiempo de duración del pulso mayor de corriente en el LED y despreciándose la porción restante como se aprecia

[27] Por ahora se busca únicamente conocer como varía la intensidad de LED mediante la técnica de control propuesta y circuitos utilizados para ello, pues en principio, se trata operar al LED de forma lineal, pero más adelante serán detallados los aspectos mediante los cuales puede obtenerse una respuesta no lineal que no es ocasionada por el control y comportamiento del LED, sino por el detector utilizado para estas mediciones.

[28] La PMT puede presentar salida en carga en ausencia de luz, debido al desprendimiento de electrones por excitación térmica, pero siendo en principio pequeña comparada con la carga debida a luz, además, la frecuencia a la que se producen es baja dentro de la cámara oscura construida para la realización de la parte óptica del experimento, con un "rate" de ~1 ó 2 pulsos por minuto. Dentro de dicha cámara oscura es donde se manejan fotomultiplicadores u otros dispositivos detectores y/o emisores de luz sensibles, ó que requieran de condiciones de oscuridad mínimas para operarse.

en la figura 3.36*b*. El resultado de la comparación de cargas en el LED y en la PMT se muestra en la figura 3.36*c*.

Como se mencionó dentro de la descripción de operación de LEDs en la sección 3.1, de manera ideal, cada carga inyectada a un dispositivo LED, producirá un fotón. En la práctica esto no es cierto, sin embargo brinda una muy buena aproximación para determinar el comportamiento del control de intensidad de un LED con el circuito y método propuestos. De este primer resultado (figura 3.36*c*) se observa que el comportamiento es bastante lineal en una región determinada, lo cual es lo más deseable para cuestiones de control. De forma simplificada, lo que se busca es que la respuesta de intensidad del LED respecto del control sobre el mismo, sea de la forma

$$\Phi \ \alpha \ i$$

ó

$$\Phi \ \alpha \ PW$$

Es decir, la intensidad Φ, en número de fotones, es proporcional a la corriente en el LED *i* ó la intensidad Φ es proporcional al ancho de pulso *PW*, con lo que se obtendría la intensidad del LED en función del ancho de pulso (y por tanto de corriente, de forma intrínseca). Para determinar bien el rango y la forma de operar del método de variación de intensidad PWV, así como del circuito que lo realiza, se llevaron a cabo más tomas de datos, utilizando, primeramente, la PMT de un solo pixel y posteriormente la MAPMT para verificar el comportamiento del método en ambas. A continuación se comentan los resultados de estos análisis, los cuales, se encuentran divididos de acuerdo a la PMT utilizada, así como al circuito de manejo de LEDs. Se referirá cuando se modifique el circuito o algún parámetro importante del arreglo experimental.

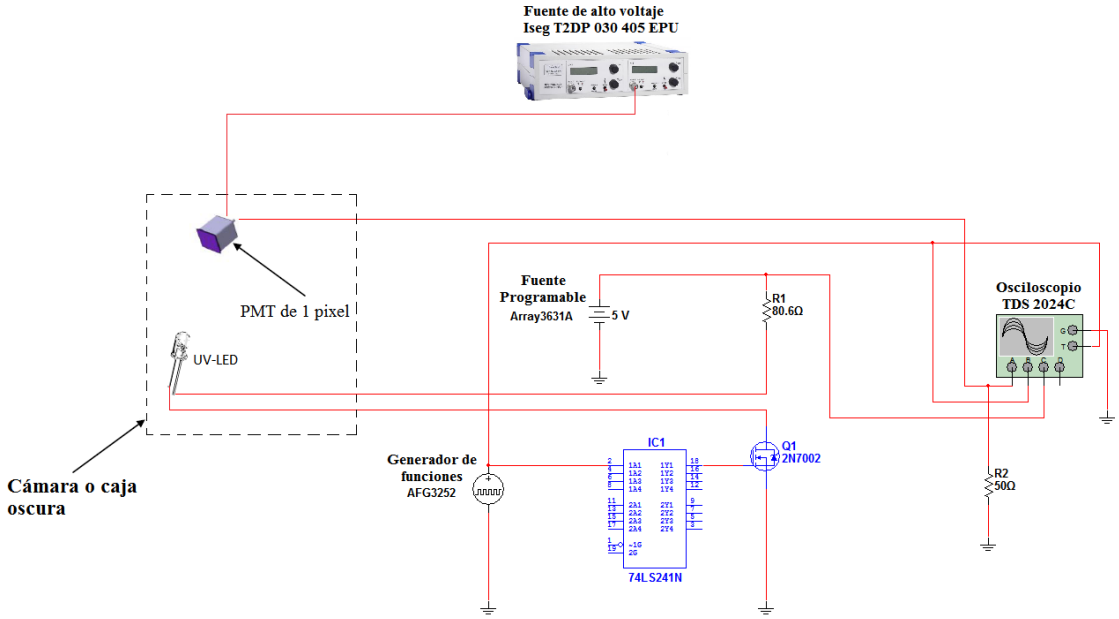
Pruebas con circuito a base de MOSFET y PMT de un pixel

Como se observa en la figura 3.36, el control de intensidad presenta hasta determinado rango de anchos de pulso, un comportamiento lineal. Sin embargo, en la siguiente prueba, se evalúa este comportamiento para un rango de anchos de pulso mayores, por dos motivos: el primero debido a que se pretende conocer hasta que rango el control es lineal, y segundo para verificar la respuesta de la PMT.

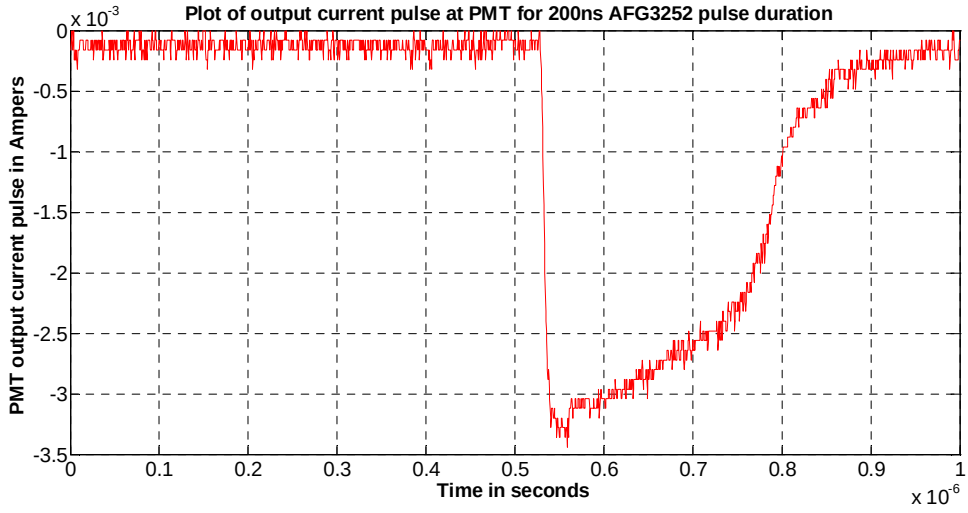
La figura 3.36 muestra los resultados obtenidos para un rango de anchos de pulso de hasta 200 ns para la PMT de un pixel y una comparación (figura 3.36*d*), con la respuesta del mismo circuito pero con la MAPMT para anchos de pulso de hasta 400 ns.

De estas pruebas, se observa que efectivamente hay una relación lineal entre el ancho de pulso utilizado y la carga, o en su defecto, si se conoce el valor de *single-photoelectron* para este dispositivo PMT de un pixel, el número de fotones detectados. La región lineal y los anchos de pulso necesarios para lograr determinar esta región son función directamente de la intensidad de corriente en el LED, así como del mismo dispositivo fotomultiplicador como se verá en la sección de pruebas y resultados.

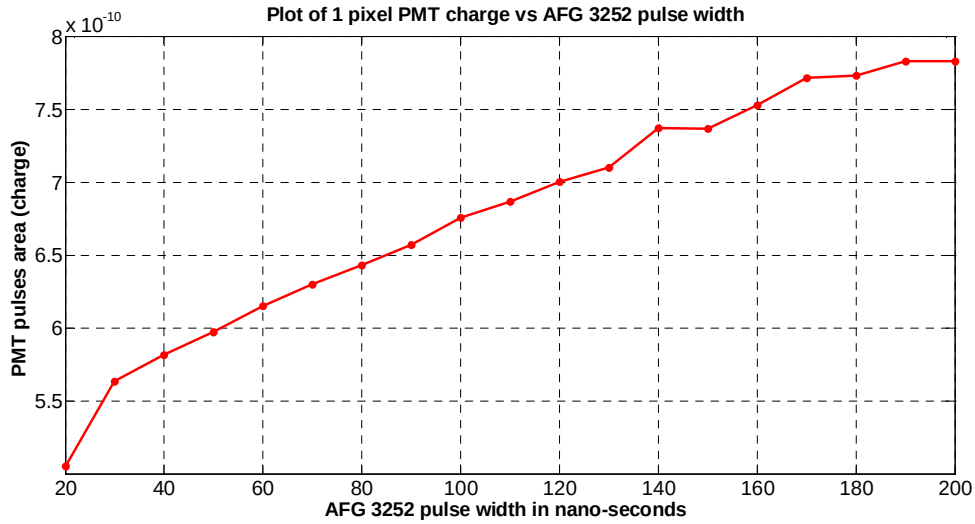
La PMT se polariza utilizando una fuente de alto voltaje con su salida ajustada a -920 V, con anchos de pulso desde 20 ns y hasta 200 ns en pasos de 10 ns. Con la resistencia serie en el LED, a valores mayores a 200 ns de ancho de pulso, la respuesta de la PMT de un pixel comienza a cambiar la forma en el pulso de salida, reduciendo su amplitud y perdiendo su forma quasicuadrada. Con este valor de resistencia serie con en el LED (80.1 Ω) se espera una intensidad de corriente nominal en el LED de alrededor de 20 mA. De estas pruebas se verifica la validez del método PWV para controlar la intensidad del LED de forma lineal en otro dispositivo fotomultiplicador además del MAPMT que se utilizará en pruebas y prototipos.


Fuente de alto voltaje
Iseg T2DP 030 405 EPU
PMT de 1 pixel
UV-LED
Cámara o caja oscura
Fuente Programable
Array3631A
5 V
R1
80.6Ω
Osciloscopio
TDS 2024C
Generador de funciones
AFG3252
IC1
74LS241N
Q1
2N7002
R2
50Ω

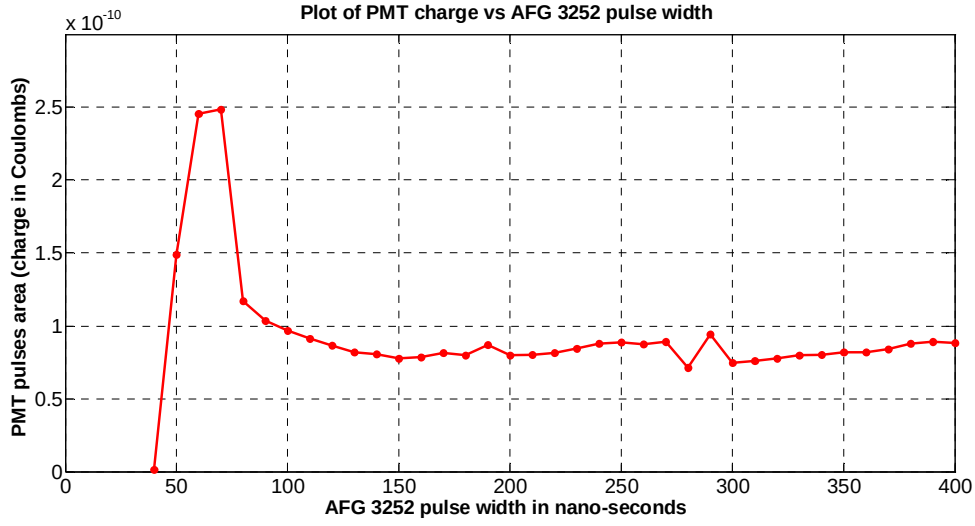
a)


Plot of output current pulse at PMT for 200ns AFG3252 pulse duration
x 10^-3
PMT output current pulse in Ampers
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time in seconds
x 10^-6

b)



c)



d)

Figura 3.36. Arreglo experimental y resultados para la PMT de un pixel y el LED UV. a) Arreglo experimental montado en la cámara oscura. b) Pulso de salida en la PMT debido a la duración del pulso mayor de corriente en el LED (200 ns). c) Carga a la salida de la PMT en función en el ancho de pulso en el LED para el rango de anchos de pulso desde 20 ns y hasta 200 ns. d) Salida en carga generada por la MAPMT en función del ancho de pulso aplicado desde 40 ns y hasta 400ns.

Circuito de encendido mediante circuito buffer Schmitt Trigger como driver de LED

Se diseñó un circuito a base de un driver en circuito integrado con entrada *Schmitt Trigger*. El circuito es bastante sencillo y consta de una resistencia en serie con el LED a la salida del buffer *Schmitt Trigger*. La función de la resistencia es obtener un valor de corriente para controlar la emisión del LED al mandar pulsos de corriente constante cuyo ancho varía. Al

mantener la intensidad de corriente constante y variar el tiempo en que esa corriente circula por el LED, lo que se hace realmente es variar la cantidad de carga inyectada al LED, lo cual significa variar directamente el proceso de recombinación en el dispositivo LED de forma que se modifica la emisión de fotones. En principio, como se comento en la sección 3.1, por cada portador de carga que se recombina se emite un fotón, sin embargo, la relación no es 1:1 (ver sección 3.1 LEDs).

Las pruebas realizadas a este circuito demuestran que, para cargas resistivas, es posible alcanzar anchos de pulso tan pequeños de entre 12 ns y 15 ns, sin embargo, al colocar al LED y la resistencia en serie como carga, la velocidad de respuesta del circuito cambia completamente (voltaje en el LED), debido a que el comportamiento de esta carga se manifiesta como la respuesta típica de un circuito RC (ver sección 3.1 LEDs).

Para determinar el comportamiento del control de intensidad mediante este circuito, se realizaron pruebas con diferentes valores de resistencia de acuerdo al esquema de la figura 3.40. Dependiendo del valor de la resistencia en serie con el LED, es posible realizar un control proporcional (lineal) de la emisión en mayor o menor grado, con lo cual se confirma nuevamente la validez del método PWV para el control de intensidad. Los resultados de las pruebas con este circuito se muestran en la sección de *Pruebas y resultados* del prototipo Track-Flat.

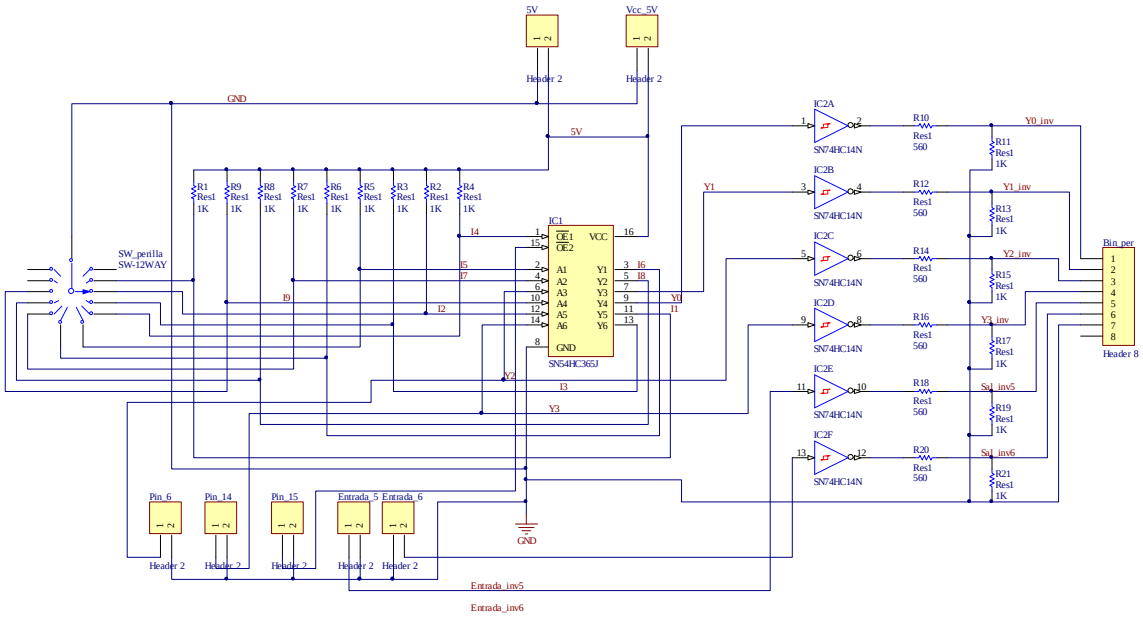
Circuitos de control

Para tener control sobre la arquitectura en el FPGA, se elaboraron diferentes circuitos, los cuales básicamente se encargan de mandar estados lógicos a los pines definidos como entradas en la tarjeta Basys2. A continuación se describe cada uno de estos circuitos.

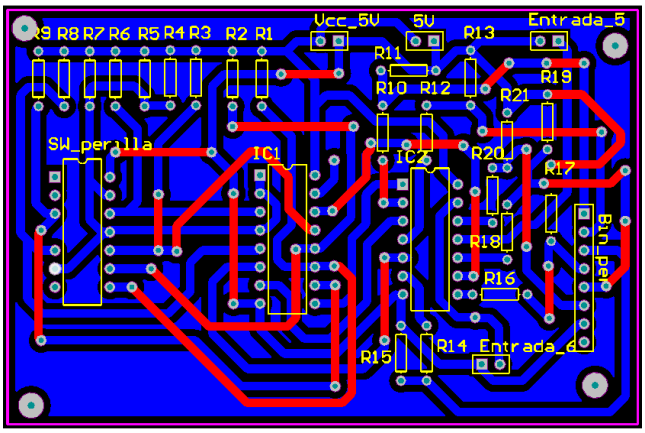
Circuito selector de anchos de pulso

Este circuito se encarga de permitir al usuario elegir entre las 10 opciones de anchos de pulso opcionales mediante una perilla o *switch* rotatorio. El circuito implementado toma las

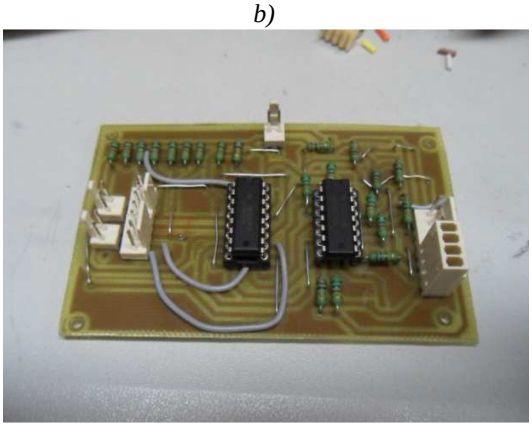
diferentes posiciones decimales del *switch* rotatorio y las convierte a valores binarios, los cuales sirven de entrada a la parte del circuito combinacional dentro del FPGA que elige el ancho de pulso a generar. El circuito consta básicamente de dos circuitos integrados, el CD74HC147 que es un codificador decimal a binario, y un circuito integrado buffer inversor 74HC14, el cual incorpora una entrada *Schmitt Trigger* a fin de eliminar espúreos o rebotes indeseados originados por la conmutación del *switch* rotatorio. A su salida se agregan divisores de voltaje para adecuar los niveles altos a valores de 3.3 V para que puedan conectarse a los pines de entrada del FPGA. Todas las etapas de esta tarjeta se polarizan a 5 V. La figura 3.37 muestra el esquemático del circuito, PCB y la tarjeta fabricada.



a)



b)



c)

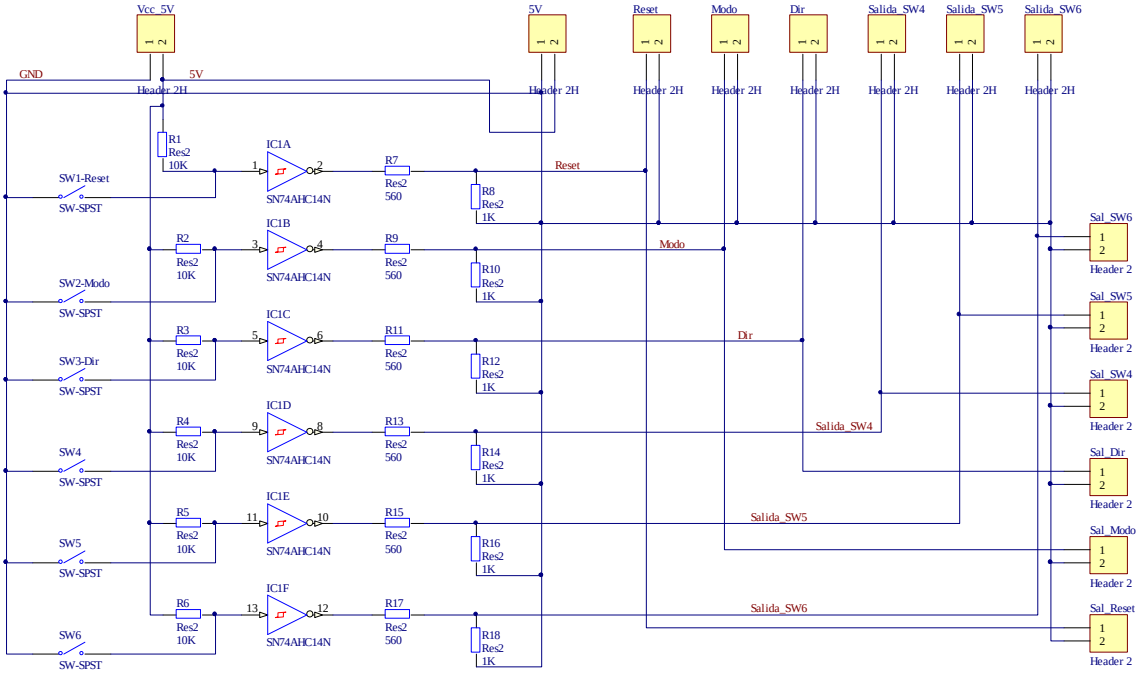
Figura 3.37. Circuito de control para selección de anchos de pulso. a) Esquemático. b) PCB diseñado utilizando la plataforma Altium Designer®. c) Tarjeta fabricada con ayuda de la máquina de prototipos Protomat S40 de LPKF Laser & Electronics.

Circuito de control para modos de operación

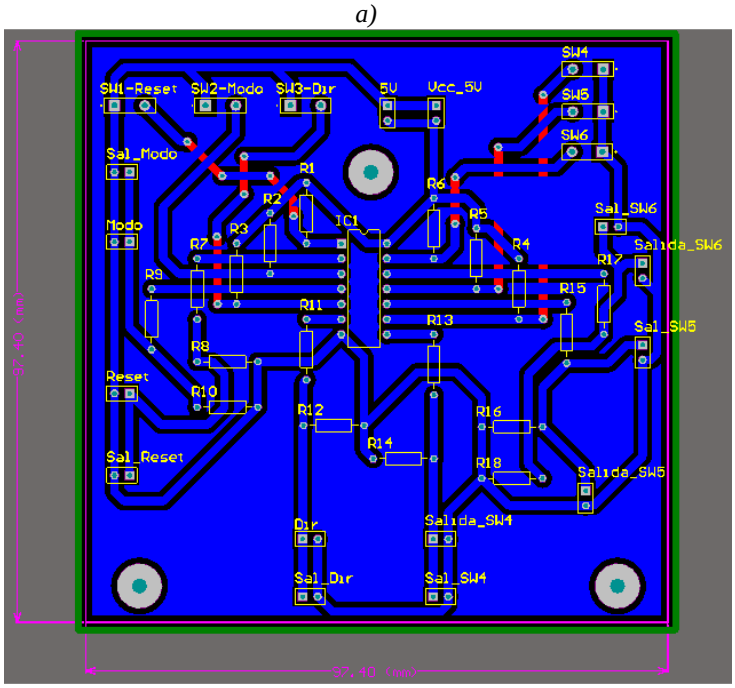
Este circuito proporciona estados lógicos a tres entradas en la tarjeta Basys2, cuyos valores son utilizados por la arquitectura interna para determinar el modo y forma de operación del Track-Flat:

a) Modo continuo o de una sola reproducción.
b) Dirección de reproducción de la traza (izquierda a derecha o de LED 1 a LED 8 y derecha a izquierda o de LED 8 a LED 1).
c) *Reset* para comenzar con la reproducción de la traza o parar y deshabilitar el prototipo.

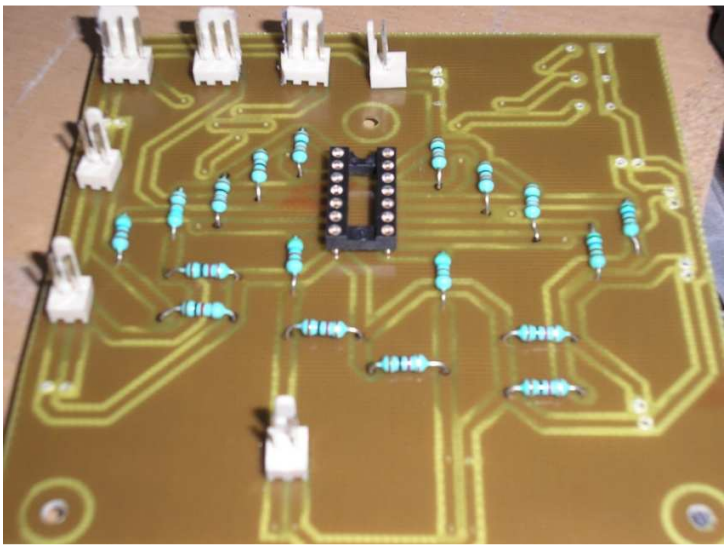
El circuito es bastante sencillo ya que solo se trata de un buffer inversor 74HC14 con un divisor de voltaje a su salida para manejar niveles lógicos de 3.3 V como entrada en alto al FPGA. La figura 3.38 muestra este circuito.



a)



b)

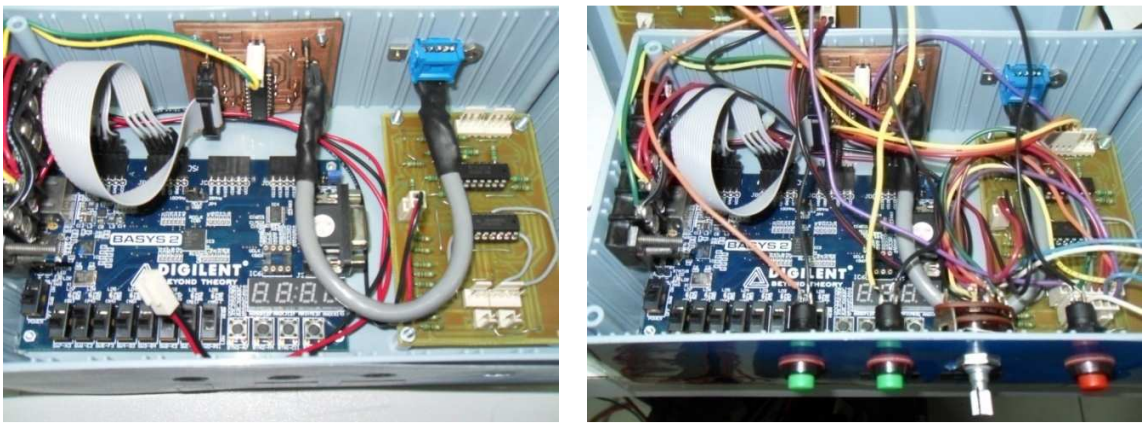


c)

Figura 3.38. Circuito de switches de control y operación. a) Diagrama esquemático. b) Diseño del PCB en Altium Designer®. c) Tarjeta construida utilizando la máquina de prototipos PCB Protomat S40.

Implementación en *casing* (gabinete) y pruebas

Una vez que se concluyeron todos los circuitos del prototipo de forma separada y se hicieron las pruebas respectivas para verificar su correcto funcionamiento, se procedió a la integración dentro de un gabinete. Con la intensión de obtener un producto compacto, así como para facilitar la implementación de las tarjetas PCB construidas, se escogió un gabinete fabricado en plástico ABS con unas dimensiones de 19.3 × 11.5 × 7.5 cm. La figura 3.39 muestra el prototipo implementado.



a)

b)

Figura 3.39. Sistema de control o caja de control del prototipo Track-Flat. a) Proceso de ensamblado. b) Prototipo final.

Funcionamiento del Track-Flat

Para operar el prototipo Track-Flat se siguen los siguientes pasos:

1. Se conecta el cable multiconductor de 8 hilos que alimenta a los 8 LEDs ubicados en la guía de luz con la caja de control mediante un conector tipo DB9 ubicado en la parte posterior como se muestra en la figura 3.39*b* (figura inferior derecha).

2. Se alinea la guía óptica con la MAPMT de forma que la parte más estrecha de la guía óptica quede apuntando a los 8 pixeles a utilizar en el dispositivo fotomultiplicador. Para esto se utiliza una marca ubicada en la guía de luz y que debe coincidir con el borde de la MAPMT. Todo este procedimiento se realiza de

manera cuidadosa, tratando de mantener en todo momento la coplanaridad tanto en la posición horizontal como en la vertical entre la guía de luz y la MAPMT.

3. Se conecta la alimentación mediante el conector tipo *plug* a la fuente de alimentación de DC de 5 V (positivo al centro) así como el conector USB (tipo mini-USB) a una fuente con alimentación USB de 5V ó al puerto USB de una PC. El conector de alimentación vía USB se encarga de polarizar a la tarjeta Basys2 así como de reprogramarla en caso de ser necesario, mientras que el conector tipo *plug* de 5 V polariza a las tarjetas de control así como a la etapa de manejo de LEDs. Esto se aprecia en la figura 3.39*b* (abajo izquierda).

4. La operación básica se realiza mediante los botones colocados en la parte frontal de la caja de control del prototipo. El botón rojo con la etiqueta de *reset*, se encarga de correr las secuencias al ser presionado (botón hacia adentro), una vez configurados los demás parámetros, como se muestra en la tabla 3.1.

Botón/Switch	Posición	Función
Reset	Presionado	El dispositivo se encuentra apagado (no se reproduce ninguna traza).
	No presionado	El dispositivo reproduce la traza configurada.
Mode	No presionado	Reproducción de una sola vez (la traza se reproduce una vez y después se detiene el dispositivo).
	Presionado	Reproducción continua (la traza generada es reproducida indefinidamente hasta que el usuario detiene la ejecución mediante el botón de *reset* al ponerlo en la posición hacia afuera o no presionado).
Dir	No presionado	La traza se reproduce desde el LED1 hacia el LED8 (de izquierda a derecha visto desde la parte frontal de la MAPMT).
	Presionado	La traza se reproduce desde el LED8 hacia el LED1 (de derecha a izquierda visto desde la parte frontal del dispositivo MAPMT).
Switch selector de intensidad	1	Ancho de pulso mínimo
	2	xxxxxx
	3	xxxxxx
	4	xxxxxx
	5	xxxxxx
	6	xxxxxx
	7	xxxxxx
	8	xxxxxx
	9	xxxxxx
	10	Ancho de pulso máximo

Tabla 3.1. Configuración para puesta en marcha del prototipo Track-Flat.

5. Una vez que se ha encendido y se ha configurado la forma de operar el dispositivo se puede comenzar a utilizarlo al presionar el botón *reset*. La MAPMT debe estar polarizada al nivel requerido, así como conectados los equipos osciloscopio, módulos NIM y/o sistemas de adquisición de datos.

Pruebas y resultados

Las primeras pruebas que se realizaron al prototipo armado consistieron en verificar que se obtuvieran las señales de encendido de LEDs de acuerdo a los parámetros de configuración deseados. Posteriormente se conectó el cable multiconductor entre la caja de control y la guía de luz y se verificó que los pulsos encendieran a todos los LEDs, observando primeramente, a simple vista, que al cambiar el ancho de pulso se reflejaba esto en un cambio de intensidad a la salida de la guía de luz. Realizadas estas primeras pruebas observacionales, se procedió a ubicar a la MAPMT en posición con la guía de luz para realizar mediciones a la salida de esta última.

El arreglo experimental para realizar las pruebas y mediciones se observa en la figura 3.40. Se resalta la parte que conforma al prototipo Track-Flat y se especifica al resto de quipos, instrumentos y dispositivos utilizados para la toma de datos. Los equipos utilizados de acuerdo al arreglo de la figura 3.40 son los siguientes:

- Cámara oscura donde se monta la parte óptica del experimento.
- Dispositivo fotomultiplicador de 64 pixeles Hamamatsu H7546B.
- Fuente de alto voltaje de dos canales ISEG T2DP 030 405 EPU con dos salidas configurables en polaridad (+/-) de hasta 3 kV a 4 mA.
- Generador de funciones Tektronix AFG3252.
- Osciloscopio digital Tektronix TDS 2024.
- Fuente de alimentación programable de salida triple Array 3631A.
- Cables coaxiales BNC LEMO de 5, 6, 10 y 16 ns de retraso.
- Multímetro de banco Fluke 8846A.
- PC para adquisición de datos.

Las primeras pruebas se realizan en modo continuo, comenzando con los anchos de pulso más pequeños y determinando la intensidad en el pixel de referencia de la MAPMT en número de fotones. El procedimiento que se realizó es el siguiente:

1. Se configura la generación de pulsos en la caja de control de acuerdo a los parámetros a probar.
2. Se ejecuta la traza.
3. Se visualiza la salida del pixel de referencia de la MAPMT en un canal (canal 1) del osciloscopio TDS 2024C, en donde se tiene configurado como *trigger* en el canal 2 al pulso de salida de la caja de control.
4. Se adquieren ambas formas de onda para determinar la carga del pulso de salida de la MAPMT y el ancho de pulso a media altura aplicado al LED. A partir de la carga de salida de la MAPMT se conoce el número de fotones detectados, al hacer el cociente del valor *de la carga del pixel de referencia entre el valor de la carga de un single-photoelectron*, previamente determinado, para esta PMT, en 1.98 pC ó ~2pC.[29]
5. Se toman varios pulsos para un mismo ancho de pulso a fin de obtener estadística representativa y determinar el comportamiento del prototipo Track-Flat.[30]

De las primeras pruebas, se observa que valores de resistencias en serie con LED muy pequeñas ($R \rightarrow 0$), proporcionan el comportamiento de la figura 3.41, donde se observa que para los anchos de pulso más cortos, la respuesta de cada canal es casi completamente lineal, pero conforme se incrementa la intensidad de los LEDs (ancho de pulso), se presentan efectos, los cuales, a primera instancia, parecen ser debidos a saturación. La saturación puede tener dos fuentes: la del dispositivo en la emisión (LED) o la del dispositivo receptor (MAPMT). Para determinar cuál es la fuente que produce la saturación se analizan los dos casos que posiblemente originan esta condición.

[29] Este fue el valor obtenido de *single-photoelectron* para el pixel de referencia de la MAPMT utilizada como se muestra en César Tavera *et al.,* 2011.
[30] César Tavera *et al.,* 2011.

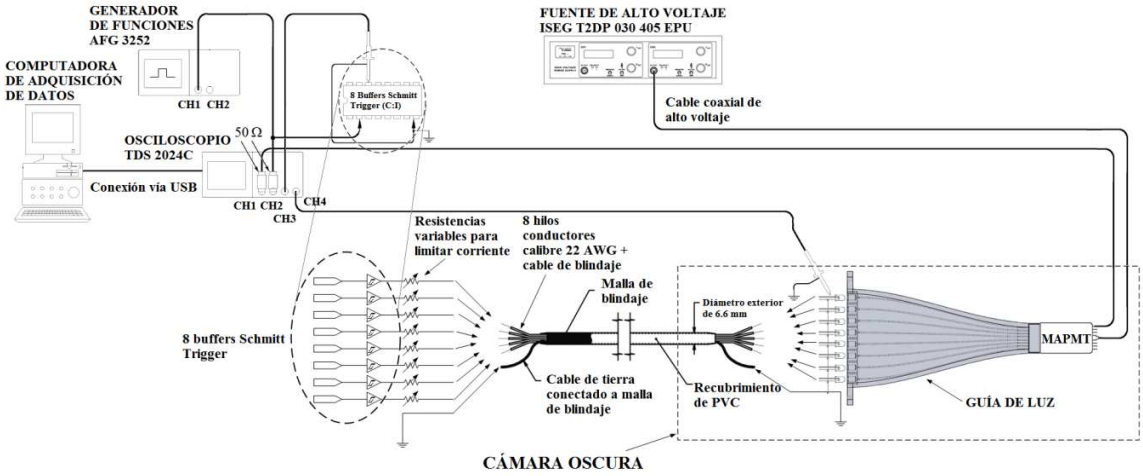
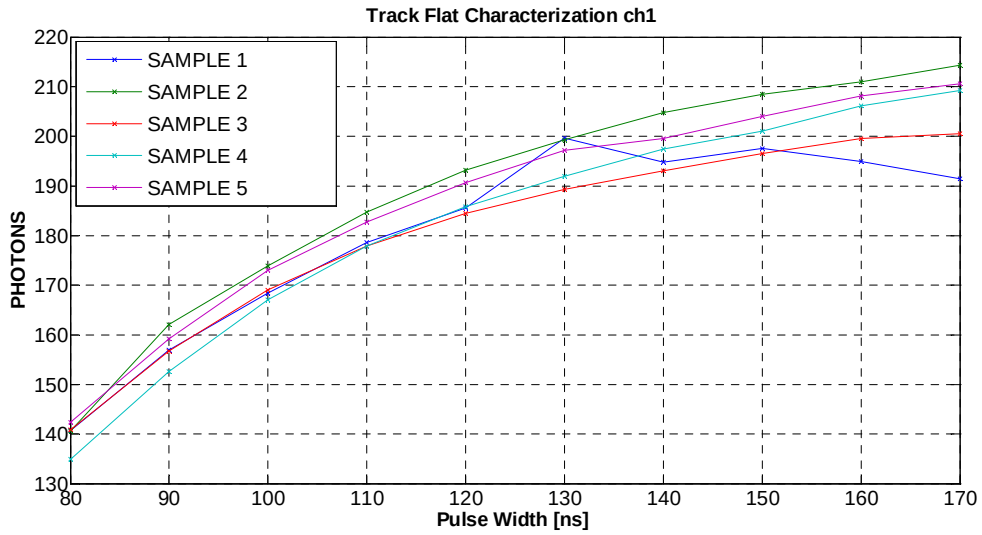
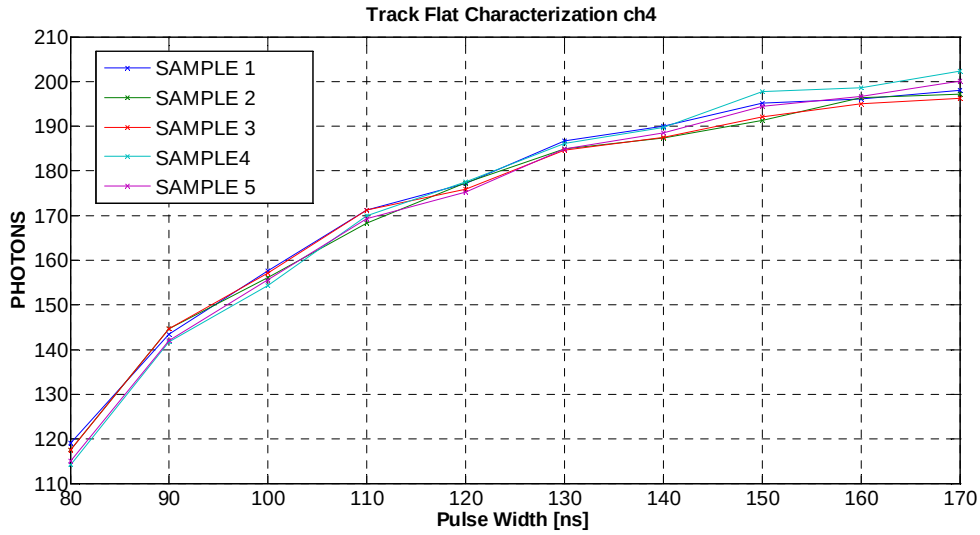


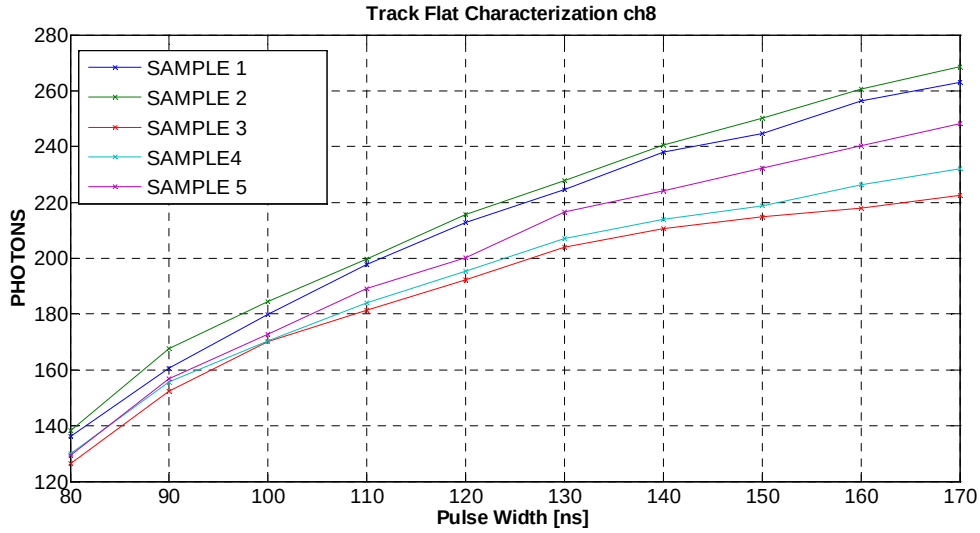
Figura 3.40. Montaje experimental durante las pruebas del prototipo Track-Flat.



a)



b)



c)

Figura 3.41. Primeros resultados del prototipo Track-Flat. a) Canal 1 con LED1 del prototipo Track-Flat con pixel de referencia del dispositivo fotomultiplicador. b) Canal 4 con LED4 y el mismo pixel de referencia de MAPMT. c) Respuesta del canal 8 con LED8 y pixel de referencia. Más detalles de cómo se obtuvieron estas gráficas puede consultarse en César Tavera et al., 2011.

¿Saturación en la fuente de emisión LED?

Como ya se explicó dentro del apartado 3.1 del presente capítulo, los dispositivos LEDs producen fotones en el proceso de recombinación radiativa (en polarización directa en el dispositivo). La forma en que se opera a los LEDs es precisamente al inyectar pulsos de corriente de una duración determinada en polarización directa. Por tal motivo, la emisión se

presenta cuando hay corriente de encendido en un LED y mientras mayor sea el tiempo en que se mantiene la corriente en el LED (ancho de pulso) se tiene una mayor emisión de fotones, con una respuesta en potencia óptica semejante a la de la figura 3.11, donde existe un valor al cual tiende la potencia óptica emitida por un LED a medida que se incrementa el tiempo de encendido de éste. Sin embargo, aún y cuando el LED tiene un límite de emisión en potencia (y por consiguiente en emisión en número de fotones), la saturación en un dispositivo LED es difícil que se produzca con los valores de corriente y tiempos de encendido como los utilizados, por lo que el comportamiento que se observa en 3.41 se encuentra aún distante de la saturación del dispositivo, además de que es probable que se dañe el dispositivo debido a la intensidad de corriente que se haga circular por éste antes de alcanzar la saturación.

¿Saturación en MAPMT?

Un dispositivo fotomultiplicador como el utilizado, produce un pulso de corriente en cada uno de los ánodos de salida que incorpora cuando el pixel asociado es iluminado por una fuente con una longitud de onda que se encuentre dentro del rango espectral en el que el material fotosensible del fotocátodo responde (fotones con la energía necesaria para desprender electrones del fotocátodo).

De manera resumida, un fotomultiplicador se compone y opera de la siguiente manera. Se tienen tres principales etapas en este tipo de dispositivos: receptor de luz (fotocátodo), multiplicación de electrones y colector de electrones; todas ellas funcionando dentro de un tubo al vacío (figura 3.42). La primer etapa se compone de una sección sensible a la luz denominada fotocátodo, que emite electrones cuando inciden sobre él fotones con una energía (longitud de onda) suficiente para producir el desprendimiento de los electrones del material del que se encuentra fabricado. La siguiente etapa consiste de realizar una multiplicación de los electrones emitidos por el fotocátodo, mediante un campo eléctrico que acelera estos electrones y los dirige hacia etapas de colección en forma de ánodos, los cuales reciben el nombre de dinodos. La energía de los electrones incidentes provoca la emisión de un número mayor de electrones secundarios que son dirigidos hacia un segundo

dinodo (cada etapa de dinodos se encuentra en presencia de un campo eléctrico que se genera mediante un arreglo de divisores de tensión). El número de dinodos utilizados así como su disposición varía en cada fotomultiplicador, de acuerdo a características como el tamaño del fotocátodo, composición en uno o varios pixeles, entre otras. La última etapa consiste de un receptor de electrones denominado ánodo, donde se obtiene la señal de salida del dispositivo fotomultiplicador. La figura 3.42 muestra un esquema básico del funcionamiento de un fotomultiplicador. Las cargas capturadas en el ánodo son pasadas a instrumentos y/o etapas externas para su análisis y procesamiento en forma de una corriente eléctrica, o de voltaje cuando se coloca una carga entre éste y el nivel de referencia o tierra.

Estos dispositivos poseen características y parámetros que definen su comportamiento y respuesta. Las principales son: eficiencia cuántica (eficiencia de detección en el fotocátodo), sensibilidad espectral, eficiencia de conteo de pulsos (eficiencia en la multiplicación de electrones), ganancia (en función del campo eléctrico producido por el voltaje de polarización) y corriente de consumo. Cada dispositivo fotomultiplicador responde de manera diferente, sin embargo, los pulsos de salida en todos los casos comparten más o menos las mismas características principales: rápida velocidad de respuesta (tiempos de levantamiento y decaimiento del orden de 1ns), amplitud y ancho de pulso para el mismo número de fotones detectados.

Los dispositivos fotomultiplicadores se encuentran dentro de los dispositivos más sensibles para el conteo de fotones debido a su velocidad de respuesta y gran sensibilidad, sin embargo deben ser manipulados en ambientes con condiciones de iluminación mínimas para garantizar su correcto funcionamiento.[31] Por otra parte, durante su operación, es necesario mantenerlos en lugares oscuros donde la intensidad de luz a detectar es baja, debido a que intensidades considerables de luz pueden saturar e incluso dañar permanentemente el dispositivo. Por este motivo, la saturación en el dispositivo MAPMT se vuelve el mejor candidato a la respuesta obtenida en las curvas de 3.41, lo cual se probó como se describe a continuación.

[31] Debido a que los fotomultiplicadores son muy susceptibles a pequeñas intensidades lumínicas, es necesario dejarlos en reposo durante algún tiempo antes de operarlos, si es que fueron expuestos a luz intensa.

Las pruebas realizadas en laboratorio con el dispositivo MAPMT, mostraron que la respuesta de los pulsos en el fotomultiplicador se "deforman" y reducen en amplitud para intensidades cada vez mayores en la fuente luminosa utilizada (LED UV). Para casos en que el LED se operaba de forma no pulsada mediante una fuente de corriente directa, la respuesta de la MAPMT se reduce al mínimo pero la corriente que consume el fotomultiplicador crece considerablemente. Esto se debe a la gran cantidad de fotones que llegan al fotocátodo y desprenden electrones, provocando una gran cantidad de éstos últimos en el proceso de multiplicación. Si la intensidad de la fuente emisora es demasiada (muchos fotones llegando al MAPMT), se puede llegar al momento de saturar el fotocátodo de forma que ya no se emiten más electrones, e incluso se puede llegar a la destrucción de éste.

En diferentes pruebas se utilizaron varios valores de resistencias en serie con el LED, de forma que diferentes intensidades de corriente en tiempos de encendido determinados se generaron. Como se aprecia en la figura 3.43, a menores intensidades de corriente (valores mayores de resistencia serie en el LED, y que en la figura 3.43 van de ~11 mA a ~2.65 mA), vuelven al control más "fino" y permite crecer la región lineal así como la cantidad de fotones detectados en la MAPMT. Esto implica que al hacer más grande los valores de la resistencia serie en el LED, el control mediante PWV y utilizando el circuito buffer *Schmitt Trigger* como *driver* para el LED, se puede lograr un mejor control lineal. Sin embargo, no puede llevarse a valores demasiado grandes de resistencia serie debido a que los anchos de pulso requeridos para lograr el encendido del dispositivo LED son también mayores, reduciendo la posibilidad de encender todos los pixeles que requieran de una determina intensidad durante el mismo GTU mediante el arreglo matricial en el Track-Sim.

De estos resultados, se determina que la saturación se produce en el MAPMT antes de que ocurra en el LED. Así mismo, se establece que el control de intensidad por PWV extiende la región proporcional con intensidades de corriente menores, siendo en principio, para esta aplicación, el valor temporal del GTU y el número de pixeles a encender la limitante.

Otro aspecto de fundamental importancia y del que se deriva que la saturación ocurre en el dispositivo MAPMT, se refiere al número de fotones detectados a mayores valores de resistencia serie. Esto se debe a que la carga inyectada con intensidades de corriente menores para los mismos incrementos temporales, es menor, permitiendo una emisión de fotones desde el dispositivo en incrementos menores para los mismos incrementos temporales que con corrientes mayores, lo cual, se deriva también del comportamiento *RC* del circuito, dado que la constante $\tau = RC$ se vuelve mayor. De ahí que el control en la región lineal de detección de fotones crezca. Sin embargo, se ha observado que valores demasiado pequeños de corriente (*R* grande) involucran tiempos de encendido prolongados a lo largo de un GTU, con lo cual, aún y cuando se incrementa la región lineal y el rango dinámico que es capaz de detectar el MAPMT, se limita, en el caso del Track-Sim, la posibilidad de encender varios LEDs dentro de un GTU con el mismo circuito de control. Resultados de estas pruebas, así como del tamaño de la menor matriz característica que viaja en una lluvia (aproximadamente 3 × 3 pixeles), se puede, en primera instancia, determinar que para estos 9 LEDs a encender en un GTU, el ancho máximo permitido es de ~278 ns, y de la figura 3.43*b* se sabría que una resistencia de ~350 Ω sería la forma de conseguirlo, para un número de hasta ~200 fotones detectados con este dispositivo MAPMT, sabiendo desde luego que el número de fotones emitidos por el LED es mayor. De esta misma aproximación se obtiene una corriente de ~4 mA en el LED, donde 4 mA = $(V_{SchTrg} - V_{LED})/R$, con V_{SchTrg} = 4.5V, V_{LED} = 3.1V y R = 350Ω.

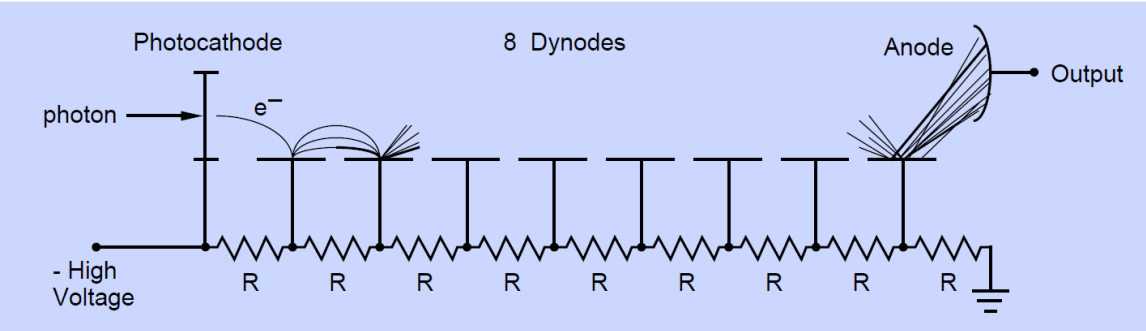
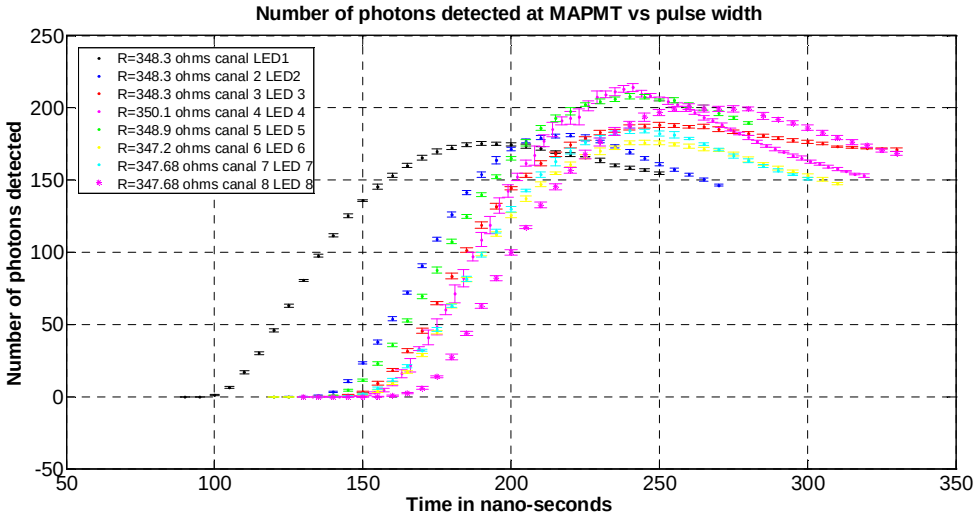
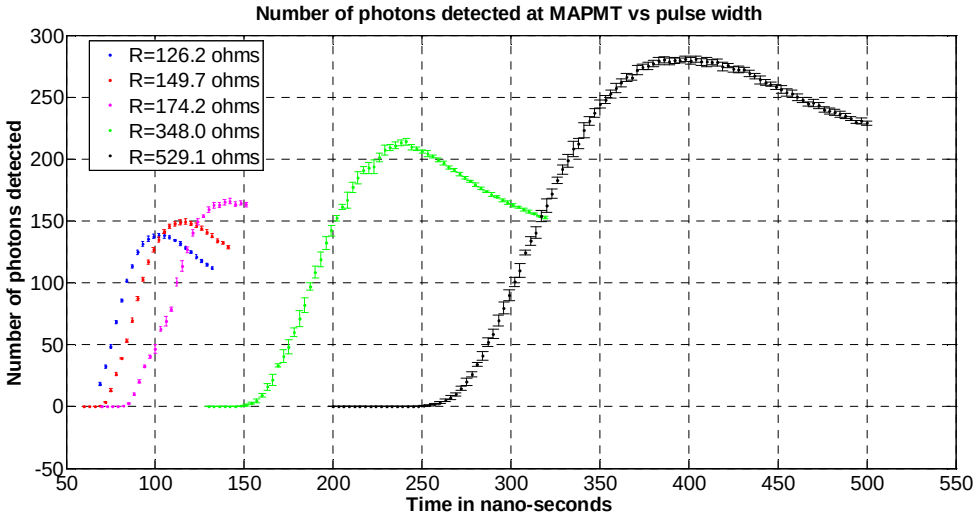


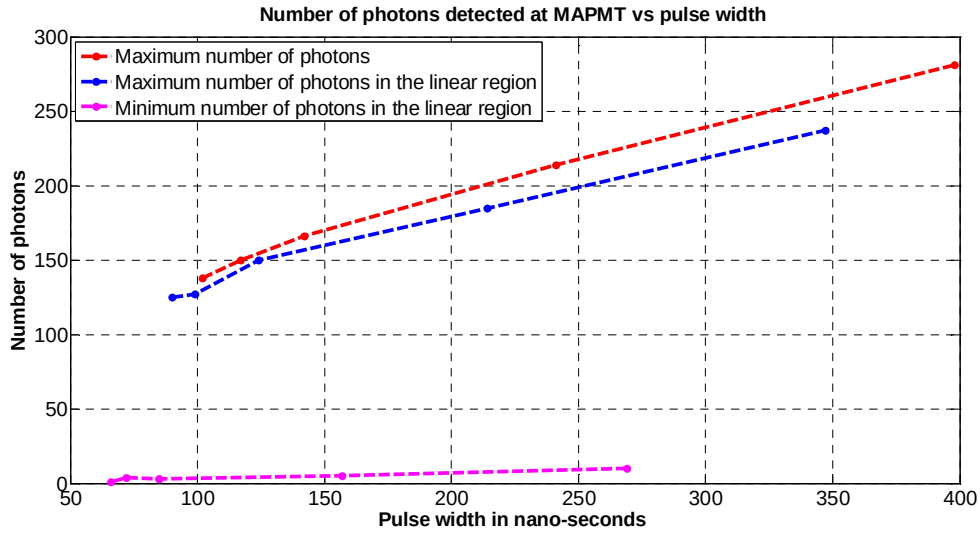
Figura 3.42. Operación y composición de un dispositivo fotomultiplicador con 8 etapas de dinodos. Imagen tomada de [26].


Number of photons detected at MAPMT vs pulse width
R=348.3 ohms canal LED1
R=348.3 ohms canal 2 LED2
R=348.3 ohms canal 3 LED 3
R=350.1 ohms canal 4 LED 4
R=348.9 ohms canal 5 LED 5
R=347.2 ohms canal 6 LED 6
R=347.68 ohms canal 7 LED 7
R=347.68 ohms canal 8 LED 8
Number of photons detected
Time in nano-seconds

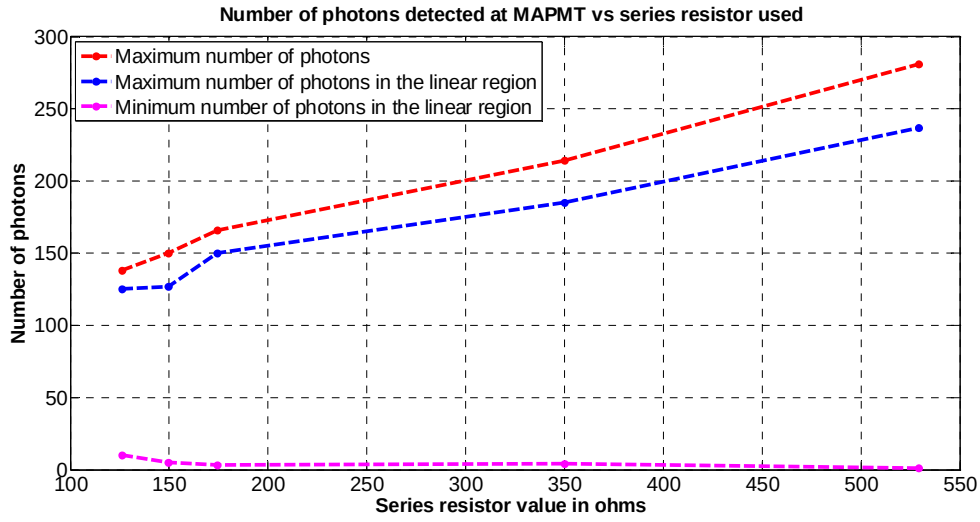
a)


Number of photons detected at MAPMT vs pulse width
R=126.2 ohms
R=149.7 ohms
R=174.2 ohms
R=348.0 ohms
R=529.1 ohms
Number of photons detected
Time in nano-seconds

b)



c)



d)

Figura 3.43. Respuesta en MAPMT para diferentes valores de resistencia serie en el LED para el circuito del arreglo experimental de la figura 3.40. a) Número de fotones detectados en cada canal de la guía óptica para el mismo valor de resistencia serie de 350Ω con el LED. b) Número de fotones detectados por el pixel de referencia en la MAPMT para el mismo canal de la guía óptica (canal 4) con diferentes valores de resistencia serie en el LED. c) Número máximo, máximo en la región lineal y mínimo en la región lineal de fotones detectados en el dispositivo MAPMT en función del ancho de pulso. d) Número máximo, máximo lineal y mínimo lineal de fotones detectados en el dispositivo MAPMT en función del valor de resistencia serie del LED.

3.3 DISEÑO DEL HARDWARE FINAL: GENERACIÓN DE PULSOS – MATRIZ DE LEDS

Como se ha comentado en la sección 3.1.1, la configuración final del dispositivo Track-Sim consiste en una matriz de 48 × 48 LEDs, los cuales son acoplados a una guía de luz que termina de tal forma que se adapta a una estructura PDM. A continuación se expone la forma en que se desarrolló la arquitectura del hardware que controla la matriz de LEDs, mostrando y resaltando su interacción con el programa de control.

Matriz de LEDs

La arquitectura creada es la que se describió dentro de la sección 3.1.1 y que se muestra en la figuras 3.22 y 3.23. Como se comentó, esta arquitectura es la que realiza todo el proceso de reproducción de los patrones asociados a cada secuencia que direcciona a la matriz de LEDs. Sin embargo, aún cuando ya se estableció la forma de operar de dicho hardware para controlar los *switches* que manejan directamente a los LEDs, falta establecer la manera completa de operar, lo que involucra en especial al bloque de control y comunicación de la arquitectura de hardware dentro del dispositivo FPGA. Este bloque realiza tres funciones principales:

- Escritura ó recepción de datos enviados desde el software de control.
- Lectura ó transmisión de datos hacia el software de control.
- Reproducción de las secuencias en memorias RAM.

Estas tres funciones son las tareas básicas que el conjunto hardware – software debe ser capaz de realizar. La figura 3.44 muestra el diagrama de flujo con estas funciones para el hardware de control.

Este diagrama de flujo se encuentra estrechamente relacionado con el propio correspondiente al programa de control, sin embargo, en esta sección sólo se comenta respecto del primero (para detalles sobre los algoritmos del programa de control consultar

apéndice 4). Como se observa, las acciones básicas mencionadas arriba son ejecutadas por el hardware de acuerdo a la evaluación de condiciones provenientes del programa de control. Cada condición se compone de un patrón de bits que es reconocido y validado por el bloque de control y comunicación.

Primeramente, y aquí viene una condición muy relacionada y dependiente del flujo de datos en el programa de control, se ejecuta una condición dentro del programa para verificar que el hardware se encuentre encendido. Una vez cumplido esto dentro del programa, el resto compete únicamente a procesos dentro del hardware.

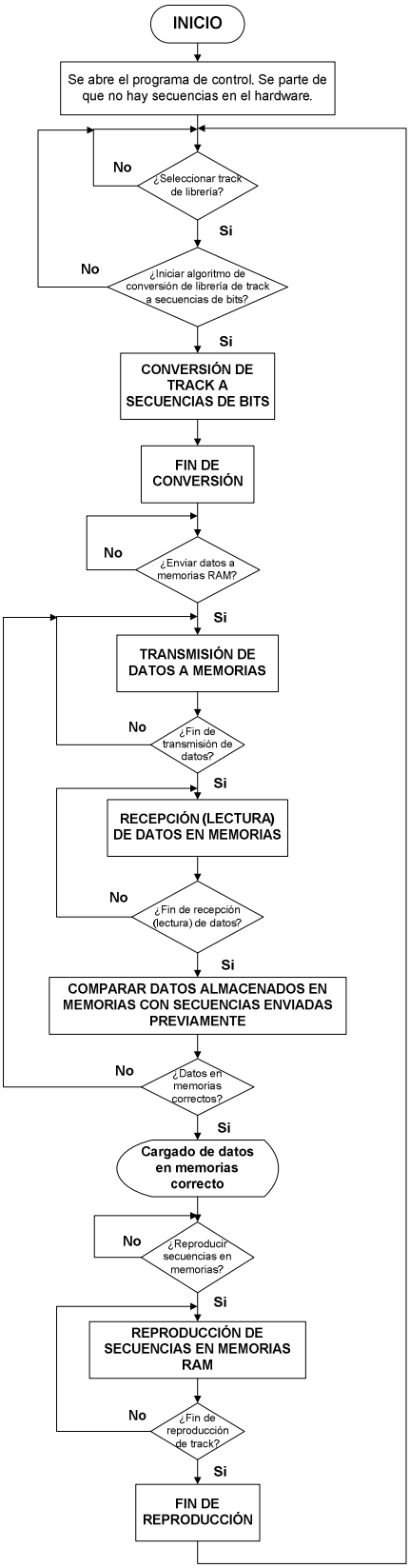
El primer paso dentro del funcionamiento del hardware (una vez encendido) es el de cargar las secuencias de una traza en las memorias RAM. Por la misma naturaleza de una memoria RAM, cada vez que el sistema es apagado (desenergizado) el contenido de las memorias RAM es borrado, por lo que cada vez que se apague el sistema, antes de realizar la reproducción de cualquier traza, primeramente se siguen los pasos de:

a) Seleccionar la traza a reproducir de la librería en el programa.
b) Cargado de datos en memorias (función de escritura en hardware).
c) Transmisión de datos almacenados en memorias (función de lectura).
d) Reproducción de las traza en forma de secuencias de bits en memorias.

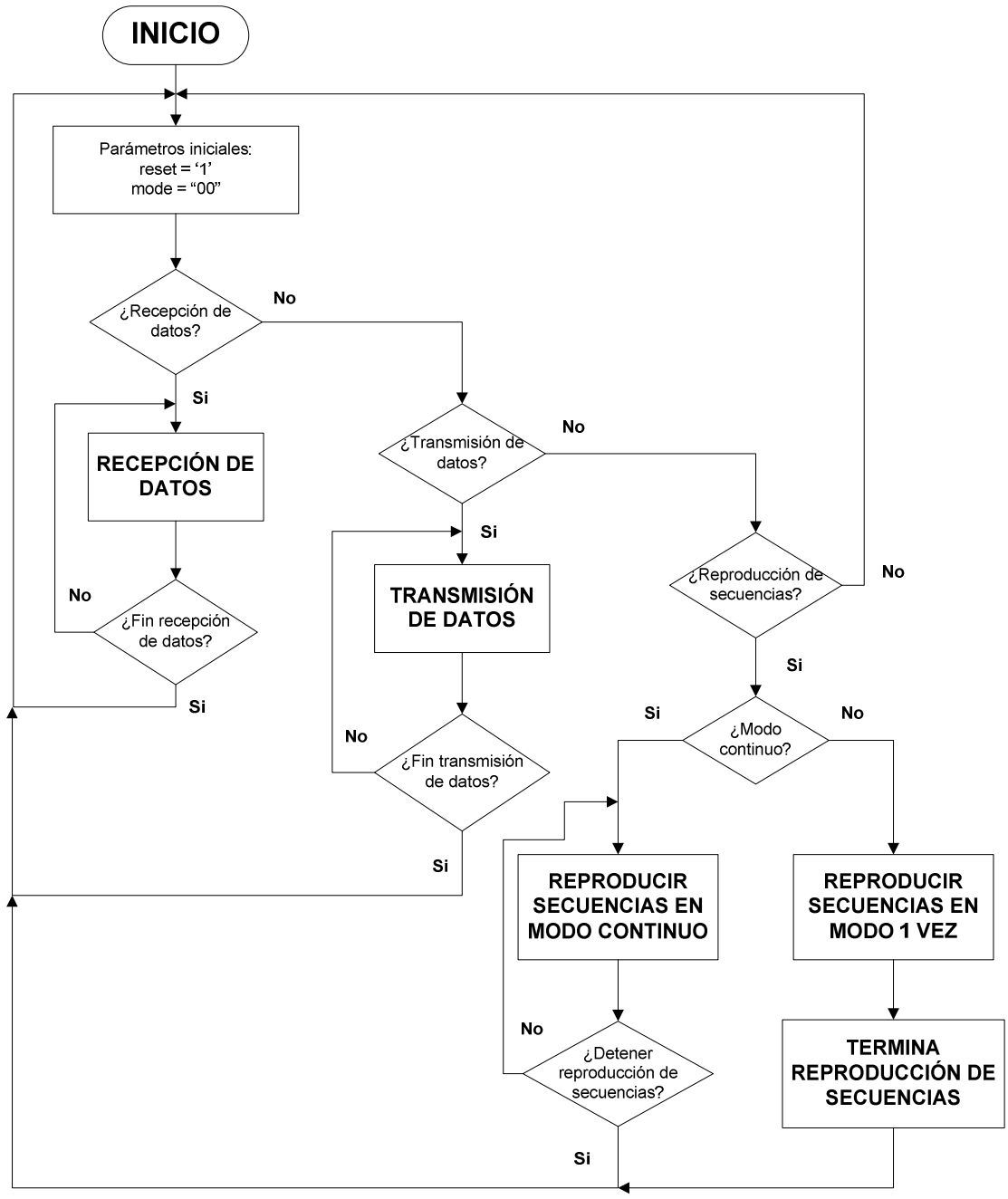
Estos pasos se muestran en la figura 3.44*a*.

Por lo que respecta a los procesos de la figura 3.44*b*, se representa el flujo de datos cubriendo los posible escenarios para cualquier modo de operación del instrumento Track-Sim, considerando desde que se enciende el dispositivo para comenzar a trabajar con él y se reproduce la primer traza (cumpliendo cada una de las tres funciones básicas del hardware como aparecen en 3.44*b*, de izquierda a derecha), hasta que se tiene alguna traza cuya reproducción se ha llevado a cabo y se desea cargar otra traza para su reproducción, volver a reproducir la traza ó simplemente se desea leer la traza actualmente en memorias.

Cada una de las tres funciones básicas ubicadas en el diagrama de la figura 3.44*b* se encuentra a su vez ramificada en diversos procesos de acuerdo a la función que realizan, y, en el caso de la reproducción de las secuencias en memorias, se considera también la configuración deseada. Para entender mejor cada etapa de escritura, lectura y reproducción de secuencias, se presentan a continuación las máquinas de estados utilizadas por el bloque de control y comunicación en cada proceso.


INICIO
Se abre el programa de control. Se parte de que no hay secuencias en el hardware.
No
¿Seleccionar track de librería?
Si
No
¿Iniciar algoritmo de conversión de librería de track a secuencias de bits?
Si
CONVERSIÓN DE TRACK A SECUENCIAS DE BITS
FIN DE CONVERSIÓN
No
¿Enviar datos a memorias RAM?
Si
TRANSMISIÓN DE DATOS A MEMORIAS
No
¿Fin de transmisión de datos?
Si
RECEPCIÓN (LECTURA) DE DATOS EN MEMORIAS
No
¿Fin de recepción (lectura) de datos?
Si
COMPARAR DATOS ALMACENADOS EN MEMORIAS CON SECUENCIAS ENVIADAS PREVIAMENTE
No
¿Datos en memorias correctos?
Si
Cargado de datos en memorias correcto
No
¿Reproducir secuencias en memorias?
Si
REPRODUCCIÓN DE SECUENCIAS EN MEMORIAS RAM
No
¿Fin de reproducción de track?
Si
FIN DE REPRODUCCIÓN

a)



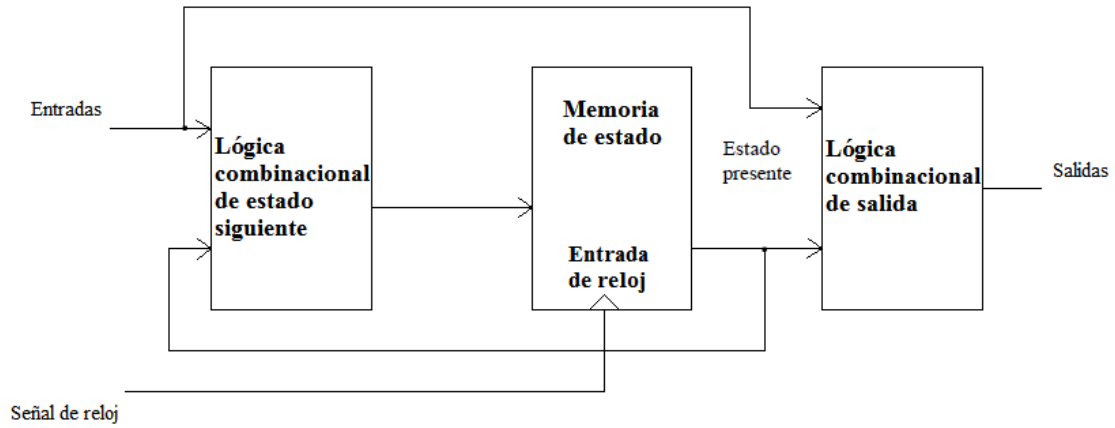
b)

Figura 3.44. Diagrama de flujo del hardware de control. a) Secuencia de funcionamiento desde que se arranca de cero el sistema (encendido del sistema hardware-software). b) Procesos que se ejecutan en el hardware de acuerdo a cada una de las funciones básicas que realiza.

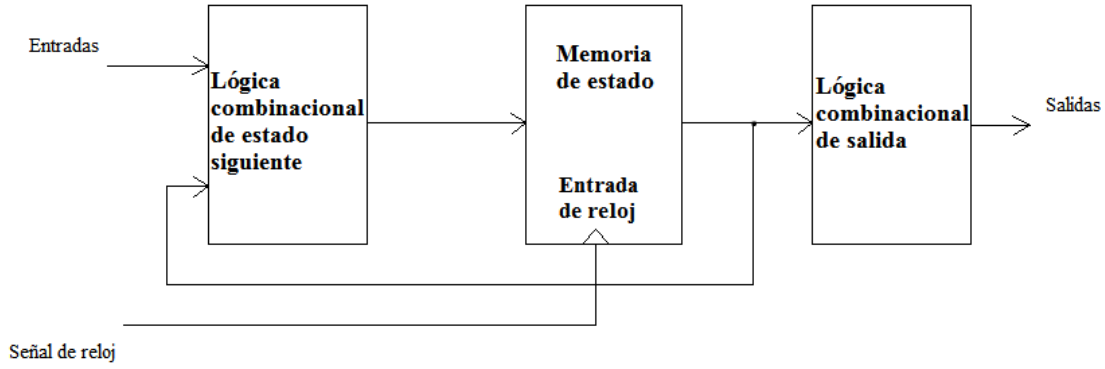
Máquinas de estados finitos (FSM)

En primer lugar se comentará brevemente sobre lo que es una máquina de estados finitos. Una máquina de estados finitos o FSM (*Finite State Machine*) es una técnica de modelado de circuitos lógicos secuenciales. El modelo más general de un circuito lógico secuencial tiene entradas, salidas y estados internos, y tradicionalmente se han diferenciado dos modelos básicos de este tipo de circuito, los modelos de Mealy y Moore, cuya diferencia radica en la forma en que la salida es generada. En el modelo de Mealy, la salida es función del estado presente y de la entrada mientras que en el modelo Moore la salida es función únicamente del estado presente. Ambos modelos se refieren a la constitución de una máquina de estados finitos (FSM). La figura 3.45 muestra los diagramas de máquinas de estados finitos en configuración de Mealy y Moore.[32]

[32] En el modelo Moore las salidas del circuito lógico secuencial se encuentran sincronizadas con el reloj debido a que dependen únicamente de las salidas de los flip-flops, los cuales se encuentran sincronizados con el reloj. Para el modelo Mealy las salidas pueden cambiar si las entradas cambian durante un ciclo de reloj e inclusive pueden presentar valores "falsos" momentáneos debido al retraso entre los cambios de las entradas y los cambios en las salidas de los flip-flops. Para evitar estas dificultades y sincronizar el modelo Mealy, las entradas del circuito secuencial deben estar sincronizadas con el reloj y las salidas deben presentarse inmediatamente después del flanco que marca el cambio en el reloj. Las entradas son cambiadas en el flanco "inactivo" del reloj para garantizar que las entradas a los flip-flops se estabilizan antes de que ocurra el flanco activo del reloj. De esta manera, la salida de una máquina de estados finitos tipo Mealy es el valor que se encuentra presente inmediatamente después del flanco activo del reloj.



a)



b)

Figura 3.45. Diagrama a bloques de dos máquinas de estados. a) Modelo de Mealy. b) Modelo de Moore.

Para el diseño del hardware de control se utiliza una máquina de estados global, la cual constituye la arquitectura del bloque de control y comunicación, sin embargo, para su mejor exposición y comprensión al momento del diseño, se ha separado a esta máquina de estados finitos en 4 máquinas: la primera que representa de forma global los principales estados de cada proceso, y las otras tres que contienen la forma de operar entre y de los estados correspondientes a las tres funciones básicas de escritura, lectura y reproducción de secuencias en memorias RAM. La figura 3.46*a* muestra la estructura de la máquina de estados finitos global, en donde aparecen los tres principales estados que se ejecutan en el hardware (Recepción de datos o escritura, Transmisión de datos o lectura y Correr secuencia o reproducción de secuencias en memorias RAM). Se observa que la asignación de estados depende de tres bits denominados *mode* (2 bits) y *reset*, los cuales son entradas

físicas al FPGA en la sección del bloque de control y comunicación; a su vez, el bit de *reset* se dirige a otros bloques de la arquitectura creada dentro del FPGA como las memorias RAM, los contadores, relojes, etc., ya que éste representa la habilitación global del sistema.

En lo que respecta a la figura 3.46*b*, ésta representa el diagrama de estados para el estado *Recepción de datos* (escritura) donde se observan los procesos que se realizan durante el envío de datos de secuencias desde el programa de control hacia el hardware. Este diagrama de estados cuenta con tres estados o procesos básicos que realizan el cargado de datos en memorias como se describe a continuación.

- *RECEPCION DE DATOS* (en FPGA): los datos en forma de secuencias de bits correspondientes a una traza, se encuentra listos para ser mandados a las memorias RAM. El programa de control lo hace saber al hardware al poner *reset* = '1' y *mode* = "01" (manteniendo desde el programa *pc_ready* = '0', *pc_do* = '0' y *ultimo* = '0'), ante lo cual, el sistema en el FPGA para al estado "RECEPCIÓN DE DATOS" y responde con *fpga_ready* = '1'. Cuando el programa confirma que el hardware se encuentra en modo de recibir datos ó escritura (al ver *fpga_ready* = '1'), envía los datos en la configuración de bits siguiente. Primero se envían 3 bits que indican la posición del byte de datos (*byte_addr*) a continuación se envían 14 bits de la dirección de localidades de memorias (*addr*) y el bit de selección de memoria (*select_mem*). Posterior al envío de estos primeros 18 bits de configuración, se envían 8 bits de datos de secuencias en los 8 pines del bloque de control y comunicación denominados *data_in*. Posterior al envío de los 8 bits de datos se pone desde el programa de control *pc_do* = '1', con lo que se guardan los 8 bits de datos en la posición, localidad y memoria respectiva establecidos por los 18 bits de configuración. Instantes después de poner *pc_do* = '1' desde el programa, se manda la configuración *reset* = '1', *pc_ready* = '1', *ultimo* = '0', ante lo cual, el sistema de hardware pasa al estado de RECIBE DATO. Cualquier otra condición de los bits *reset* y *pc_ready* durante este estado, mantiene o regresa al estado inicial de la máquina de estados (INICIO), o envía al estado de ULTIMO DATO.

- *RECIBE DATO*: en este estado se establece que se han guardado los bits en la memoria y localidad correspondiente a la configuración que se estableció durante el estado de RECEPCION DE DATOS. En este estado el sistema en el FPGA pone *fpga_ready* = '0' con lo que se le indica al programa que se ha guardado el dato y el hardware está listo para guardar otro dato o finalizar el proceso de escritura. El programa, al ver *fpga_ready* = '0', manda la configuración *reset* = '1', *pc_ready* = '0' y pc_do = '0' para pasar al estado de RECEPCION DE DATOS (se mantiene *pc_do* = '1'). De otra forma, se mantiene en este estado o se va al estado de INICIO (interrupción por software o hardware para abortar el proceso de recepción de datos en las memorias RAM).

- *ULTIMO DATO*: se pasa del estado RECIBE DATO a este estado cuando se mandan desde el programa de control *reset* = '1', *pc_ready* = '1' y *ultimo* = '1' (manteniendo *pc_do* = '0'). En este estado se le indica al hardware que se ha enviado el último byte de datos. El sistema de hardware responde ante esto con *fpga_ready* = '0', con lo que se le informa al programa que el hardware está listo para otra tarea (lectura, reproducción de secuencias o escritura nuevamente).

Ahora, el diagrama de estados de la figura 3.46*c* detalla el procedimiento que realiza la arquitectura del bloque de control y configuración para llevar a cabo una lectura de los datos almacenados en las memorias RAM. La descripción del diagrama de estados es la siguiente.

- *TRANSMISION DE DATOS* (desde sistema FPGA a programa de control): los datos de la configuración del track almacenados en las memorias RAM serán enviados al programa. Se cae en este estado al poner desde el programa *reset* = '1' y *mode* = "10" (se pone a su vez *pc_do* = '0'). Al pasa el hardware a este estado, pone *fpga_ready* = '1' para informarle al programa que se puede pasar a la lectura de los datos en el estado siguiente de ENVIA DATO, terminar el envío de datos al pasar a ULTIMO DATO TR, mantenerse en este estado o pasar al estado de INICIO por interrupción (*reset* = '0'). Mientras se mantiene en este estado mediante *reset* = '1',

pc_ready = '0' y *ultimo* = '0' ó con *reset* = '1', *pc_ready* = '0' y *ultimo* = '1', se colocan los bits respectivos en la configuración adecuada para leer el byte de datos que se desea, de la siguiente forma. Primeramente el programa envía los 3 bits que indican la posición del byte de datos al colocar la configuración de estos tres bits en *byte_addr*, los 14 bits para direccionar la localidad de memoria que se colocan en *addr* y el bit de selección de memoria (*select_mem*). Posterior al envío de los 18 bits de configuración, el programa pone *pc_do* = '1' ante lo cual el sistema en el FPGA pone el byte de datos en los pines asignados a *data_out*. Una vez que el programa ha puesto *pc_do* = '1', con lo cual el byte de datos se encuentra disponible para leerse en los pines respectivos, se manda desde el software *reset* = '1', pc_ready = '1' y *ultimo* = '0' (manteniendo *pc_do* = '1' para leer el byte de datos), con lo que se pasa al estado de ENVIA DATO.

- *ENVIA DATO*: en este estado, se indica que la lectura del dato en la posición, localidad y memoria respectiva. Aquí el sistema del FPGA pone *fpga_ready* = '0' para indicarla al programa que se ha puesto la información del byte de datos para ser leído. Al ver *fpga_ready* = '0', el programa, una vez leído el byte de información, coloca *reset* = '1', *pc_ready* = '0' y *pc_do* = '0' (manteniendo *ultimo* = '0') con lo que se va al estado de TRANSMISION DE DATOS para prepararse para otra lectura de datos o finalizar con este proceso. Cualquier otra combinación de los bits de *reset*, *pc_ready* y *pc_do* mantiene en este estado (ENVIA DATO) o manda al estado de INICIO por interrupción del proceso.

- *ULTIMO DATO TR*: se pasa a este estado desde el estado TRANSMISION DE DATOS cuando se manda desde el programa *reset* = '1', *pc_ready* = '1' y *ultimo* = '1' (manteniendo *pc_do* = '0'). En este estado se pone *fpga_ready* = '0' para indicarle al programa que se está listo para abandonar el proceso de transmisión de datos desde el sistema FPGA hacia el programa. Al ver *fpga_ready* = '0', el programa manda *reset* = '1'y *pc_ready* = '0' ó simplemente *reset* = '0' (por interrupción) para pasar al estado INICIO y terminar con el proceso de transmisión

de datos (se mantiene *pc_do* = '0'). La otra combinación de los bits *reset* = '1' y *pc_ready* = '1' mantendrá al hardware en este estado.

En lo que respecta al diagrama de estados de la figura 3.46*d*, éste muestra la forma en que se realiza la reproducción de la traza en forma de secuencias de bits almacenadas en las memorias RAM. La operación es como sigue.

- *CORRER SECUENCIAS*: se llega a este estado cuando el programa, desde el estado *INICIO*, coloca *reset* = '1' y *mode* = "11". En este estado, la arquitectura del sistema pone *fpga_ready* = '1', con lo cual, se da aviso al programa de que se está listo para comenzar con la reproducción de las secuencias en las memorias. Desde el programa, antes de seleccionar el pasar en la arquitectura a modo de reproducción de secuencias, se selecciona si la reproducción será de forma continua o de una sola vez, ya sea al mandar *reset* = '1', *mode* = "11" y *repeat* = '0' ó *repeat* = '1'. Con *reset* = '0', o con *reset* = '1' y *mode* ≠ "11" se pasa al estado INICIO.

- *UNA VEZ*: en este estado se ha elegido la forma de reproducción de una sola vez y se espera a que se finalice la traza mediante *senal1* = '1' (secuencias en memorias RAM) o *senal2* = '1' (para la traza plana). De otra forma se espera en este estado o se va al estado INICIO por interrupción. En este estado se mantiene *fpga_ready* = '1'.

- *CONTINUO*: en este estado se realiza una reproducción de forma continua de la traza previamente elegida (traza elegida de librería o traza plana), la cual se detiene hasta que el usuario así lo determine para pasar al estado de INICIO. De igual forma que en el caso de la reproducción de una sola vez, el sistema pone *fpga_ready* = '1'.

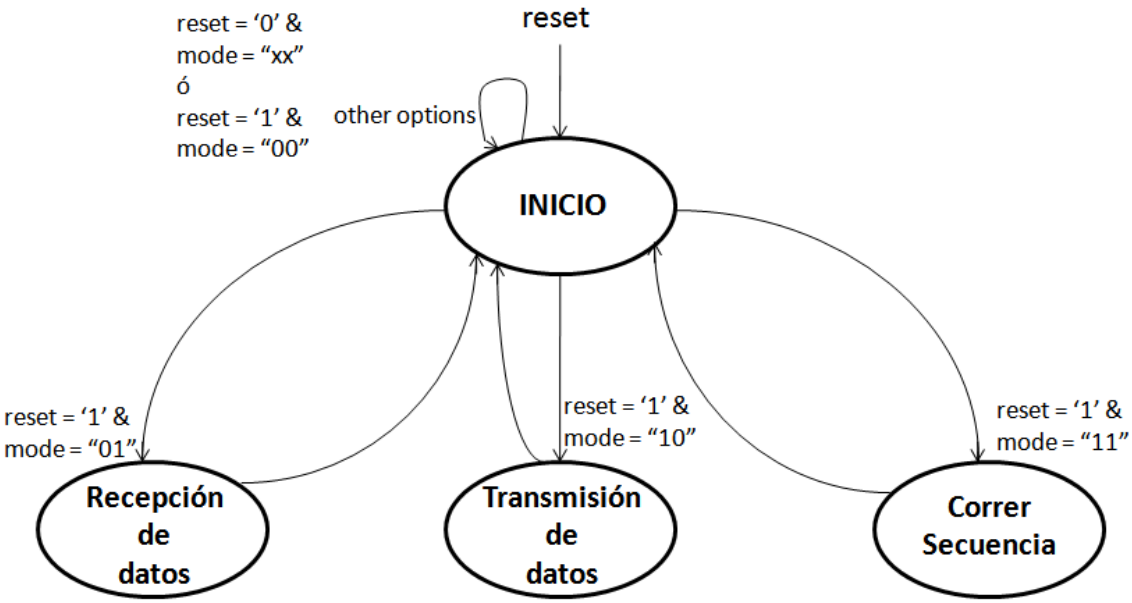
- *Track en memorias*: se elige reproducir el *track* contenido en memorias RAM. Dependiendo de la configuración seleccionada, se pasa al estado siguiente de *Fin de secuencia* si se eligió la reproducción de una vez ó se mantiene reproduciendo la secuencia en memorias (con posibilidad de cambiar algunos parámetros) de forma

indefinida hasta que el usuario decida detener la ejecución de manera “normal” o mediante la interrupción global de *reset*.

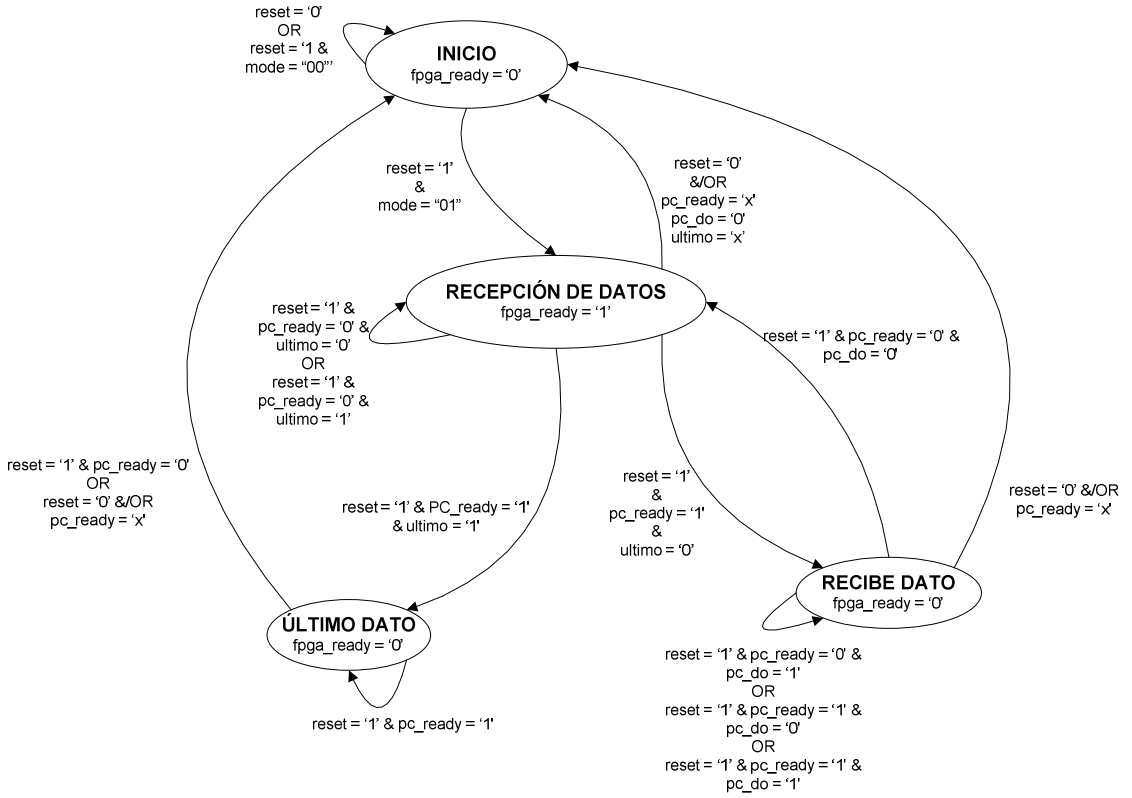
- *Track plano*: se reproduce el *track* plano (*track-flat*) de intensidad constante. De igual manera que en el estado de *Track en memorias*, se pueden cambiar algunos parámetros como el sentido de reproducción (adelante o atrás) la reproducción de una sola vez o de forma continua, así como traza en memorias o traza plana. Si se eligió la reproducción de forma continua, o se pasa al siguiente estado de *Fin de secuencia* (si se eligió la reproducción de una sola vez), se termina de pasar por los 48 pixeles en forma horizontal que cubren a un PDM.

- *FIN SECUENCIA*: se pasa a este estado si se eligió la reproducción de una sola vez de la traza y cuando se ha terminado la reproducción de la misma. Se mantiene en este estado hasta que usuario determina la acción a realizar mediante *reset* = ‘1’ y *mode* ≠ “11” ó por interrupción con *reset* = ‘0’.

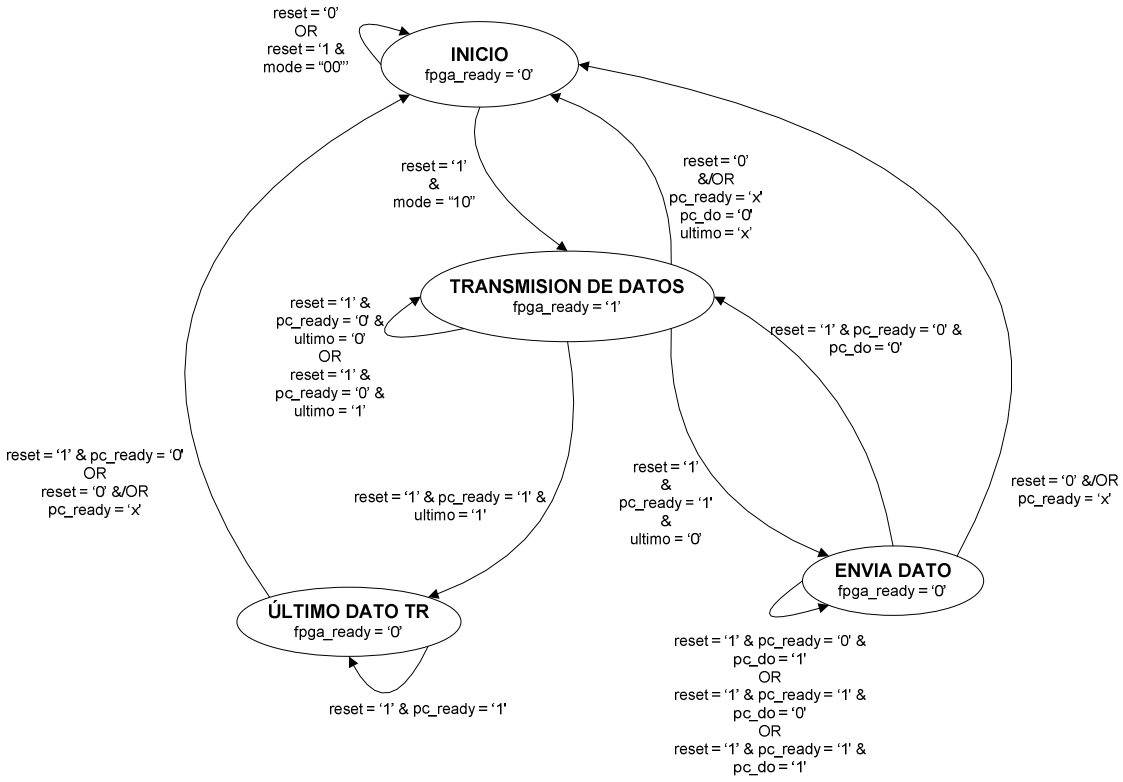
Es importante notar que en cualquier momento y proceso que se encuentre realizando todo el sistema (hardware y software), al presentarse una acción interrupción mediante *reset* = ‘0’, se interrumpen todos los procesos y se regresa al estado inicial. De esta forma se deshabilitan todas las funciones del hardware y se espera a que se regrese a las condiciones de operación “normal” nuevamente, con lo cual, cualquier proceso interrumpido de escritura, lectura o reproducción de secuencias, se retoma desde el principio, siendo el usuario quien decide la acción a realizar. La interrupción global por parte de la señal de *reset* es ocasionada únicamente por el usuario, ya sea de forma manual en hardware o mediante la interfaz de usuario del programa de control.


reset = '0' &
mode = "xx"
ó
reset = '1' &
mode = "00"
reset
other options
INICIO
reset = '1' &
mode = "01"
reset = '1' &
mode = "10"
reset = '1' &
mode = "11"
Recepción de datos
Transmisión de datos
Correr Secuencia

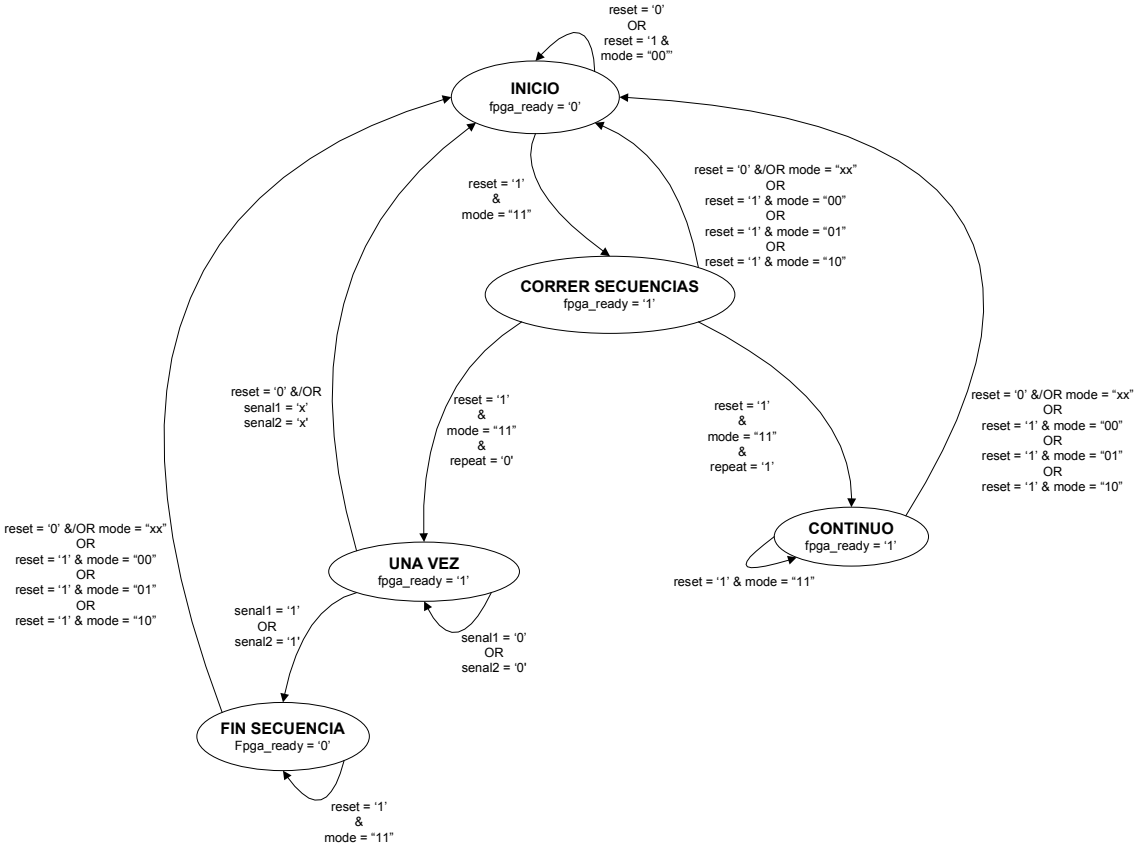
a)


reset = '0'
OR
reset = '1 &
mode = "00"'
INICIO
fpga_ready = '0'
reset = '1'
&
mode = "01"
reset = '0'
&/OR
pc_ready = 'x'
pc_do = '0'
ultimo = 'x'
RECEPCIÓN DE DATOS
fpga_ready = '1'
reset = '1' &
pc_ready = '0' &
ultimo = '0'
OR
reset = '1' &
pc_ready = '0' &
ultimo = '1'
reset = '1' & pc_ready = '0' &
pc_do = '0'
reset = '1' & pc_ready = '0'
OR
reset = '0' &/OR
pc_ready = 'x'
reset = '1' & PC_ready = '1'
& ultimo = '1'
reset = '1'
&
pc_ready = '1'
&
ultimo = '0'
reset = '0' &/OR
pc_ready = 'x'
RECIBE DATO
fpga_ready = '0'
ÚLTIMO DATO
fpga_ready = '0'
reset = '1' & pc_ready = '1'
reset = '1' & pc_ready = '0' &
pc_do = '1'
OR
reset = '1' & pc_ready = '1' &
pc_do = '0'
OR
reset = '1' & pc_ready = '1' &
pc_do = '1'

b)


reset = '0'
OR
reset = '1 &
mode = "00""
INICIO
fpga_ready = '0'
reset = '1'
&
mode = "10"
reset = '0'
&/OR
pc_ready = 'x'
pc_do = '0'
ultimo = 'x'
TRANSMISION DE DATOS
fpga_ready = '1'
reset = '1' &
pc_ready = '0' &
ultimo = '0'
OR
reset = '1' &
pc_ready = '0' &
ultimo = '1'
reset = '1' & pc_ready = '0' &
pc_do = '0'
reset = '1' & pc_ready = '0'
OR
reset = '0' &/OR
pc_ready = 'x'
reset = '1' & pc_ready = '1' &
ultimo = '1'
reset = '1'
&
pc_ready = '1'
&
ultimo = '0'
reset = '0' &/OR
pc_ready = 'x'
ÚLTIMO DATO TR
fpga_ready = '0'
ENVIA DATO
fpga_ready = '0'
reset = '1' & pc_ready = '1'
reset = '1' & pc_ready = '0' &
pc_do = '1'
OR
reset = '1' & pc_ready = '1' &
pc_do = '0'
OR
reset = '1' & pc_ready = '1' &
pc_do = '1'

c)



d)

Figura 3.46. Máquinas de estados de los procesos que se llevan a cabo en el hardware de control. a) Máquina de estados general del hardware de control y que constituye la arquitectura del bloque de control y comunicación. b) Máquina de estados para la recepción de datos en el hardware de control desde el programa. c) Máquina de estados para la transmisión de datos desde el hardware de control hacia el programa. d) Máquina de estados para la reproducción de las secuencias previamente cargadas en las memorias RAM del hardware de control.

Las simulaciones de la arquitectura fueron mostradas en las figuras 3.24 y 3.25, así como el código VHDL de toda la arquitectura completa puede verse en el apéndice 2A.

Circuito MOSFET en configuración para arreglo de columnas y filas

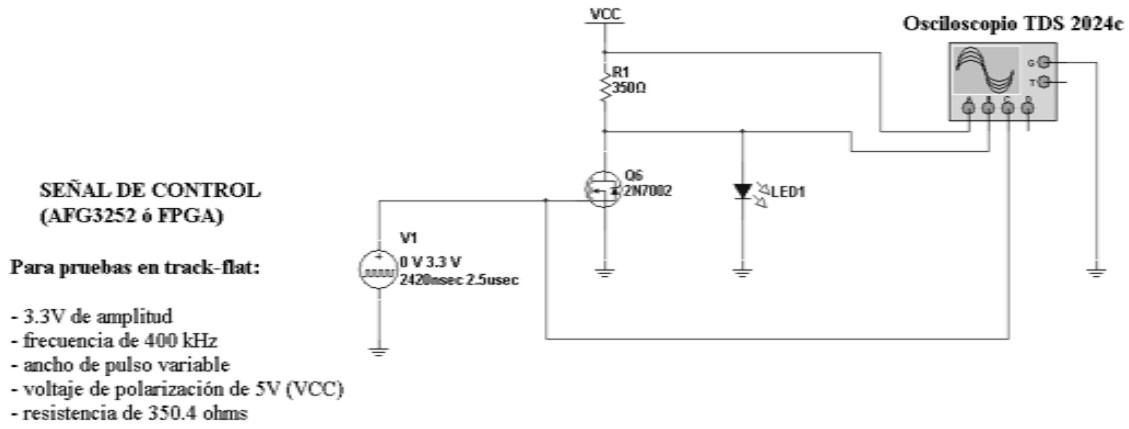
De las configuraciones presentadas en la figura 3.21, así como de las pruebas realizadas utilizando el circuito *buffer Schmitt Trigger* y el MOSFET 2N7002F, se sabe que cualquiera de estas dos alternativas de circuitos son viables para realizar el control PWV. Por otra parte, se conoce que el LED se comporta como un circuito RC, el cual se carga hasta alcanzar el valor nominal del voltaje de encendido, que para este modelo de LED es

de alrededor de 3.1 V. Así mismo, el LED comienza a emitir fotones por encima de ~2.5 V, de forma que conforme se incremente el ancho de pulso y el LED se encuentre por encima de este nivel, habrá mayor número de fotones emitidos y detectados en el dispositivo MAPMT, hasta alcanzar el valor en que éste se satura (aún y cuando el pulso de control se vuelve cero), dado que el voltaje y por consiguiente la corriente en el LED se decrementan en un tiempo *RC*, donde *R* es la resistencia limitadora en serie con el LED y *C* es la capacitancia equivalente del LED.

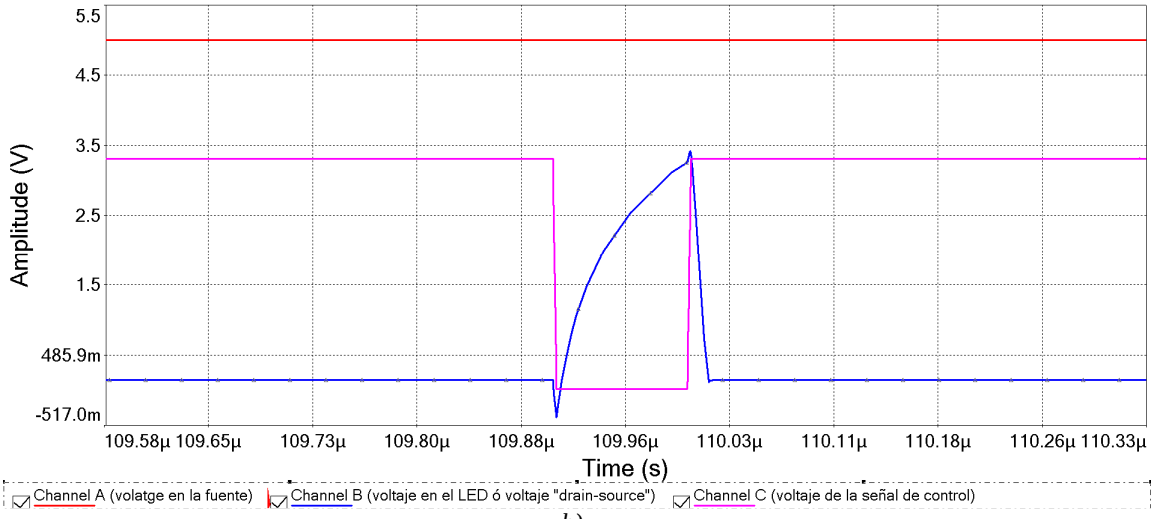
Ahora, teniendo en mente el encendido de varios LEDs dentro de un GTU (dependiendo de la lluvia a reproducir) y sabiendo que cada LED se mantiene encendido con tiempos que involucran el ancho de pulso utilizado así como la constante τ ($\tau = RC$), se vuelve evidente que antes de poder emitir un pulso para encender otro LED con el mismo circuito que controla a ambos, se debe esperar a que el LED recién encendido se apague, es decir, transcurra un tiempo de por lo menos τ posterior al pulso de excitación para estar en condiciones de mandar otro pulso con el primer LED ya apagado. Este comportamiento de los dispositivos LEDs puede representar una clara desventaja en cuanto al número de LEDs que pueden lograr ser encendidos con el mismo circuito durante el mismo GTU. Lo ideal sería que la forma de onda en los LEDs se tratase de un pulso rectangular, con tiempos de levantamiento y decaimiento muy pequeños. Una forma de mejorar la respuesta de un LED involucra el tratar de volver el voltaje en el LED lo más cuadrado posible, ya sea reduciendo el tiempo de levantamiento, decaimiento o ambos. El tiempo de levantamiento es más difícil de modificar, debido a que se trata de la naturaleza intrínseca del LED al momento de recibir un pulso de excitación mediante el circuito de control, sin embargo, para el tiempo de decaimiento es posible llevarlo a un valor mucho más pequeño. Esto se consigue mediante el mismo circuito de control utilizando un MOSFET como se aprecia en la figura 3.47. Con esta configuración el LED se apaga al encender el MOSFET, correspondiendo ahora el tiempo de apagado del LED al tiempo de encendido del MOSFET, el cual, como se verificó para este transistor, es de entre 2.5 ns y 3 ns. De esta forma, se considera que el LED se apaga de manera prácticamente "instantánea". El encendido se mantiene en un valor de τ, sin embargo, al reducir el tiempo de apagado en tal grado, es posible encender otro LED casi de forma inmediata mediante el mismo circuito

de control, lo que a su vez se traduce en más LEDs que pueden ser encendidos dentro de un mismo GTU.

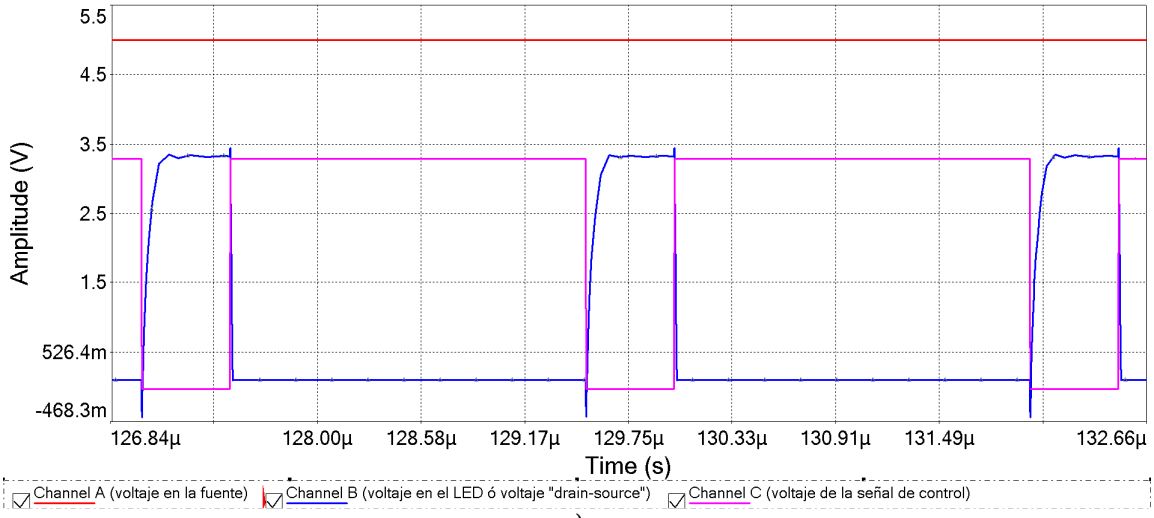
Para el prototipo Track-Flat lo anteriormente citado no representa ninguna dificultad, sin embargo, para el modelo del Track-Sim en donde se enciende más de un LED dentro de un GTU en acceso de forma matricial, si es de vital importancia el ser capaz de encender los LEDs requeridos para lograr alcanzar el número de fotones requerido. Por consiguiente, de las posibles opciones presentadas en la figura 3.21, el arreglo 4 en donde aparece una combinación de transistor MOSFET y *buffer Schmitt Trigger* aparece como la configuración óptima para el control matricial de encendido en el Track-Sim. Este circuito se armó y probó para determinar su comportamiento en cuanto al encendido del LED junto con el dispositivo sensor MAPMT. Los resultados se muestran en la figura 3.48.



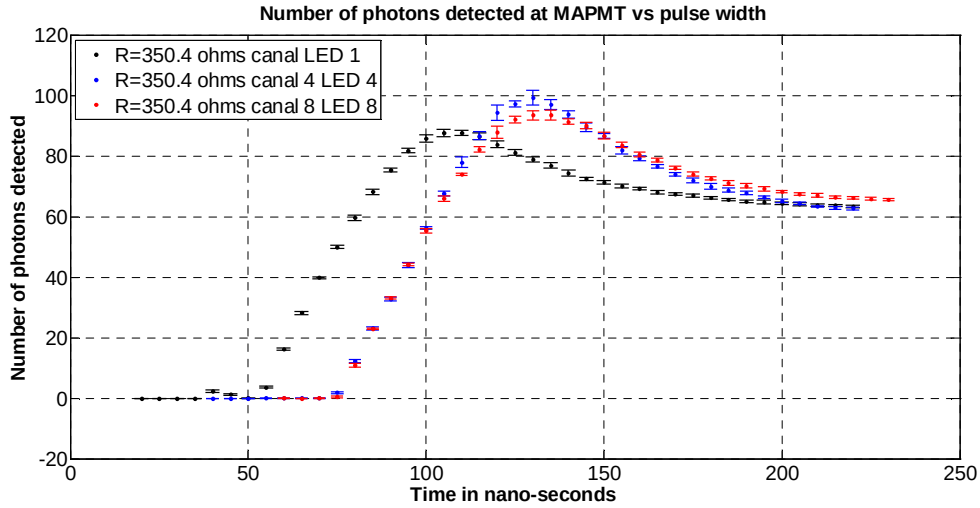
a)


5.5
4.5
3.5
2.5
1.5
485.9m
-517.0m
Amplitude (V)
109.58µ 109.65µ 109.73µ 109.80µ 109.88µ 109.96µ 110.03µ 110.11µ 110.18µ 110.26µ 110.33µ
Time (s)
Channel A (volatge en la fuente)
Channel B (voltaje en el LED ó voltaje "drain-source")
Channel C (voltaje de la señal de control)

b)

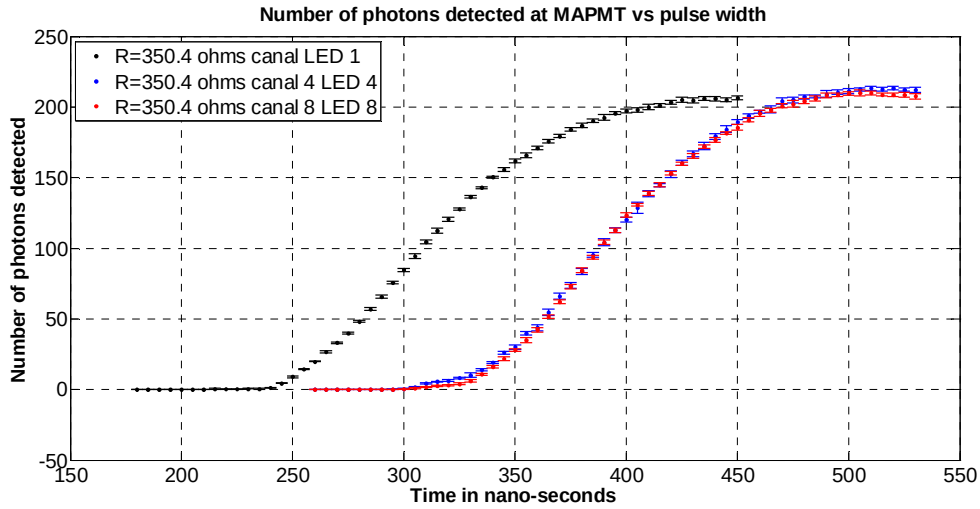

5.5
4.5
3.5
2.5
1.5
526.4m
-468.3m
Amplitude (V)
126.84µ 128.00µ 128.58µ 129.17µ 129.75µ 130.33µ 130.91µ 131.49µ 132.66µ
Time (s)
Channel A (voltaje en la fuente)
Channel B (voltaje en el LED ó voltaje "drain-source")
Channel C (voltaje de la señal de control)

c)

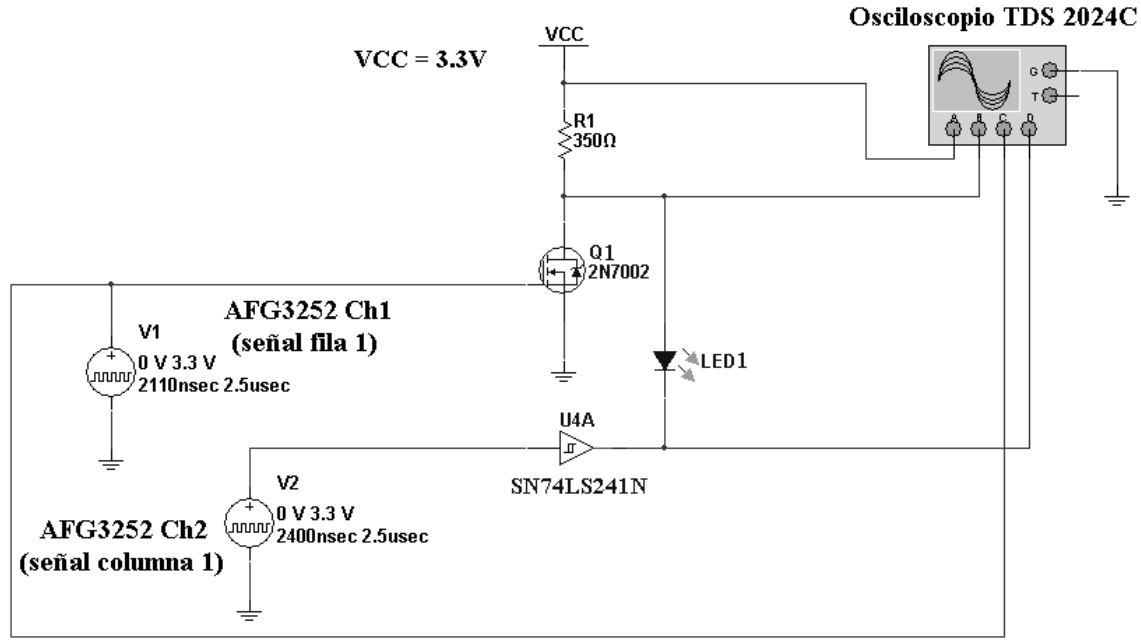


d)

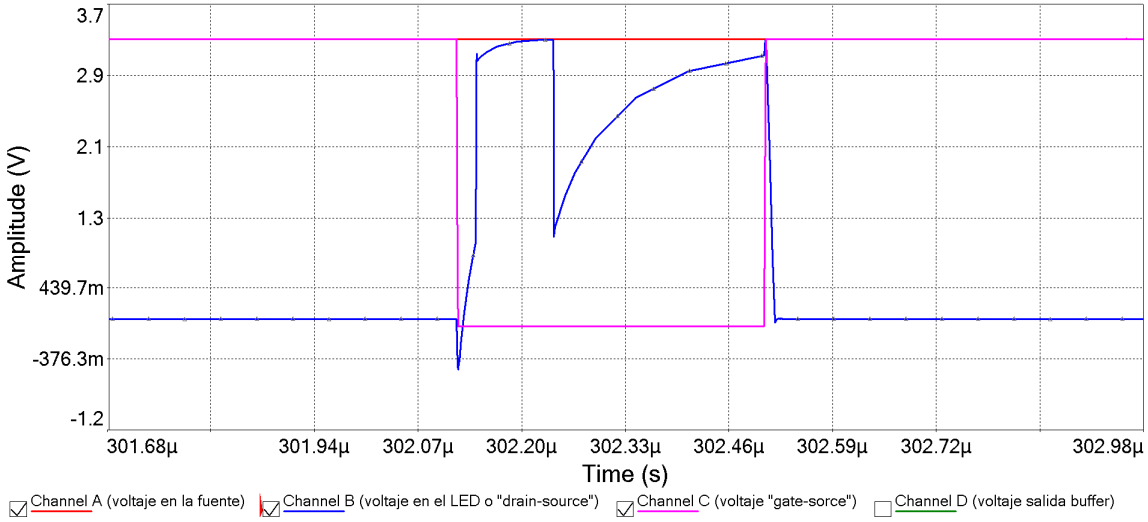
Figura 3.47. Circuito de encendido a base de MOSFET con tiempo de decaimiento reducido. a) Configuración del circuito. b) Simulación del circuito en a) para un ancho de pulso de encendido en el LED de 100ns. c) Simulación del circuito en a) para un ancho de pulso de encendido en el LED de 500ns. d) Pruebas con guía óptica y MAPMT para tres canales (canales 1, 4 y 8).



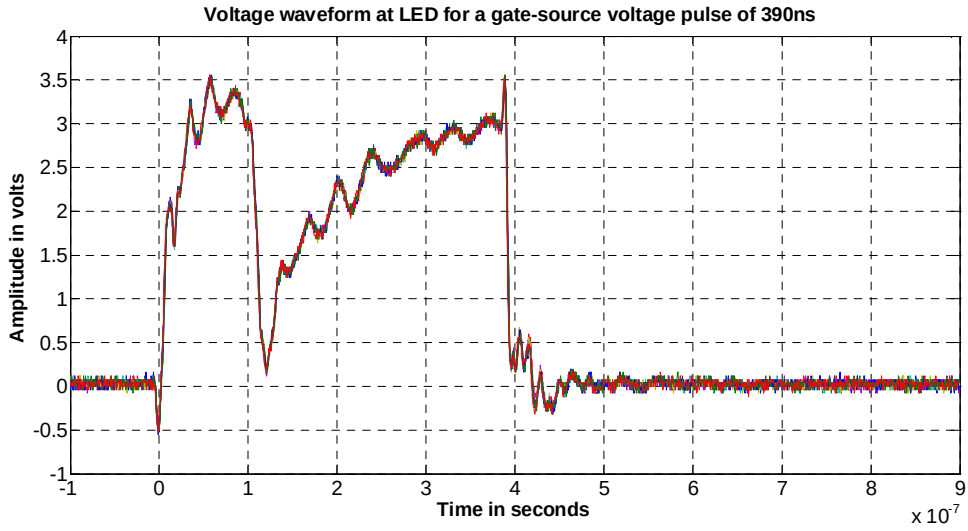
a)


Osciloscopio TDS 2024C
VCC
VCC = 3.3V
R1
350Ω
Q1
2N7002
AFG3252 Ch1
(señal fila 1)
V1
0 V 3.3 V
2110nsec 2.5usec
LED1
U4A
SN74LS241N
V2
0 V 3.3 V
2400nsec 2.5usec
AFG3252 Ch2
(señal columna 1)

b)


3.7
2.9
2.1
1.3
439.7m
-376.3m
-1.2
Amplitude (V)
301.68µ
301.94µ
302.07µ
302.20µ
302.33µ
302.46µ
302.59µ
302.72µ
302.98µ
Time (s)
Channel A (voltaje en la fuente)
Channel B (voltaje en el LED o "drain-source")
Channel C (voltaje "gate-sorce")
Channel D (voltaje salida buffer)

c)



d)

Figura 3.48. a) Intensidad en número de fotones detectados por MAPMT para tres canales de la guía óptica (canales 1, 4 y 8) para el circuito 4 mostrado en 3.21. b) Circuito implementado. c) Simulación del circuito. d) Señales de control aplicadas en transistor MOSFET y en buffer para encender/apagar al LED.

Como se observa de la figura 3.47, para el mismo valor de resistencia de 350 Ω en serie con el LED, se pueden encender al LED desde anchos de pulso menores en comparación con el circuito a base de *buffer Schmitt Trigger*, sin embargo la intensidad en número de fotones se vuelve menor. Esto sucede debido a que la salida del circuito *Schmitt Trigger* es menor que 5V (aproximadamente 4.4V), lo que se manifiesta en menor intensidad de corriente en el LED y por consiguiente una extensión en el rango de emisión antes de que el MAPMT se sature, como ocurre con valores mayores de resistencia, mientras que con el circuito de la figura 3.47 al alimentarse directamente el LED mediante la resistencia serie de 350 Ω de la fuente de 5V, la intensidad de corriente es mayor (equivalente a tener una resistencia menor) por lo que la intensidad es menor al alcanzar el valor de saturación en el MAPMT. A pesar del hecho de que se tengan menores intensidades en número de fotones, lo cual puede ser resuelto al cambiar el valor de resistencia serie a un mayor valor para así incrementar la intensidad detectada en el MAPMT, la parte importante la constituye el hecho de que se puede conseguir apagar al LED en un tiempo bastante pequeño.

En lo que respecta a la figura 3.48, se observa que el comportamiento de intensidad en número de fotones detectados en el MAPMT es muy similar al previamente expuesto mediante el circuito *Schmitt Trigger* solo como *driver* de LED, aunque en este caso se

presenta un corrimiento en ancho de pulso para alcanzar la saturación en el fotomultiplicador, del cual, 100 ns corresponden a un pulso en alto a la salida del buffer a partir de que el transistor MOSFET se encuentra en estado de corte y el LED enciende, utilizado con fines de verificar el apagado del LED, ya sea a través del transistor MOSFET y/o del buffer, de forma que, el corrimiento real, resulta en aproximadamente 50 ns, y que es consecuencia del tiempo en que la salida del buffer pasa de alto a bajo (*fall time* del *buffer*) así como del tiempo que tarda en alcanzar el valor de encendido el LED (τ) cuyo valor depende de la impedancia total que se manifiesta en el circuito resistencia-LED-*buffer* cuando el *buffer* se encuentra en estado lógico bajo (cero). Por otra parte, dependiendo del valor de la resistencia serie utilizada con cada LED, es posible alcanzar valores de intensidad mayores, claro está, utilizando a su vez, anchos de pulso mayores, sin embargo, es posible alcanzar el rango dinámico deseado ya sea con un valor de resistencia mayor o al encender el LED más de una vez con valores de resistencia menores, siendo preferible, en cuanto al algoritmo de conversión de *tracks* a patrones de bits se refiere, el encender cada LED una sola vez o la menor cantidad de veces dentro de un GTU.

Otra ventaja de la dependencia en la intensidad en número de fotones respecto del valor de resistencia utilizado, permite manejar los niveles de intensidad si necesidad de recurrir a medios ópticos, siendo posible, "adecuar la salida de cada canal respectivo a fin de compensar situaciones debidas a, por ejemplo, la geometría de los canales ópticos, acoplamientos LED-fibra óptica y guía óptica-MAPMT e incluso diferencias en los mismos LEDs así como en las señales eléctricas enviadas a los éstos.

De otros efectos observados, se verificó que las oscilaciones presentadas en la forma de onda, tanto en corriente como en voltaje en el LED, no presentan repercusiones sobre la intensidad registrada por el dispositivo MAPMT, pues en ningún caso se presentan inconsistencias que refieran a esta situación como la causa, aún utilizando diferentes configuraciones y dispositivos de conmutación (MOSFETs y buffers).

Finalmente, con estas pruebas se comprueba tanto la validez del método de control de intensidad en LEDs mediante PWV, los circuitos de manejo de LEDs, así como el

funcionamiento en un primer prototipo de 8 canales y la estrategia del control matricial mediante un sistema basado en FPGA para el diseño final del instrumento Track-Sim.

Capítulo 4

Conclusiones

***Se puede admitir la fuerza bruta, pero la razón bruta es insoportable*.**

Oscar Wilde.

En el presente trabajo se ha expuesto lo concerniente al sistema de control del instrumento Track-Sim y la validación en pequeña escala del hardware en forma del prototipo Track-Flat, lo que abre la posibilidad de la realización final de la herramienta con que se complementan las necesidades de calibración en tierra del telescopio espacial JEM-EUSO.

En lo que respecta al principal objetivo referente a la creación y validación de la estrategia con la que se controla la intensidad de perfiles de luz ultrvioleta, junto con las metas de la arquitectura de hardware propuestas, los resultados mostraron que es viable utilizar dispositivos LED como fuente luminosa variable, cumpliendo el rango dinámico de interés de 0 a 300 fotones detectados, mediante la metodología de variación de intensidad por anchos de pulso variable (PWV).

Los puntos cumplidos y la problemática resuelta en torno a éstos son los siguientes.

Control de intensidad en LEDs mediante PWV

El control de intensidad de una fuente de luz ultravioleta se realizó controlando la emisión de dispositivos LED mediante PWV, al diseñar y probar el control de un solo LED, verificando su respuesta mediante un dispositivo MAPMT, y su expansión posterior en 8 canales (8 LEDs).

Se verificó que ante variaciones de corriente menos intensas, se tiene la posibilidad de aumentar el control de la emisión (respuesta eléctrica del LED en forma de circuito RC), así como incrementar y superar en cierta medida, la problemática de saturación por parte del dispositivo detector MAPMT. Obviamente, como en muchas otras situaciones, se trata de un *trade-off* entre los beneficios que se obtienen al decrementar la intensidad de corriente en el LED, para incrementar el rango de emisión, así como la ampliación de la región de control lineal del método, pero al mismo tiempo, se incrementa el tiempo requerido para alcanzar dicho rango dinámico de intensidad, pudiendo comprometer la cantidad de LEDs que pueden ser encendidos dentro un mismo intervalo temporal ó GTU. Es entonces donde entran en juego las opciones y restricciones que el método, la forma de realizar la detección

por parte del telescopio y la propia física que se observará, permiten, siendo la principal de ellas, la distribución de la intensidad por unidad temporal de un GTU, lo cual, significa, que es posible realizar una acumulación de fotones durante la duración de un GTU y no de forma "instantánea" en cada pixel. De acuerdo con esto, no es necesario decrementar demasiado la intensidad de corriente en el LED e incrementar el tiempo de encendido, a fin de lograr la intensidad máxima del rango dinámico en una sola exposición de ancho de pulso, sino que se puede encender y apagar a cada LED durante tiempos menores, siempre y cuando se cumpla que, al finalizar el GTU del que se trate, se estará obteniendo el nivel de intensidad requerido durante ese lapso y esa posición. De esta última cuestión, así como de la posibilidad de reducir por parte del hardware, la cantidad de conexiones independientes en cada LED, se toma parte para plantear el direccionamiento matricial sin dejar de lado el control en LEDs individuales, lo cual, no sería posible, si la emisión y detección en los pixeles que conforman el conglomerado que viaja en forma de *track* durante el desarrollo de una lluvia, se presentasen en los mismos intervalos de tiempo, como se refirió en la sección 2.2.2 del capítulo 2 respecto al algoritmo de *trigger*.

Respecto de los anchos de pulso a utilizar, es importante señalar que, aún cuando no se requiera de pulsos "cortos" (<10ns), si es necesario utilizar pasos o incrementos en el ancho de pulso tan cortos como sea posible en la práctica, debido a que la resolución en intensidad es directamente función de dichos incrementos, así como de la resistencia en serie con el LED, y más propiamente dicho, de la pendiente de la recta de ajuste en la región lineal para la curva de intensidad típica emisión-detección observada con el dispositivo MAPMT. En la práctica, la resolución en intensidad se encuentra limitada por el incremento en anchos de pulso que la arquitectura de hardware permite, y como se mencionó en el capítulo 3, para la tarjeta prototipo ProASIC 3E el límite se encuentra marcado por una frecuencia de reloj máxima de 350 MHz ó ~2.9 ns. De las pruebas realizadas utilizando un generador de pulsos con incrementos de 3 y 5 nanosegundos, se observó que, por ejemplo, para el caso de la resistencia de ~175 Ω, en la región lineal, para una pendiente de 3.4 fotones/ns, si se utilizan incrementos de 5ns, se obtienen incrementos en intensidad de ~17 fotones detectados, lo que se mejora a ~10 fotones detectados para el caso de incrementos temporales de 3ns. Para el caso de la resistencia serie de ~530Ω, la pendiente que se

obtiene es de 2.7 fotones/ns, con incrementos en intensidad de ~14 fotones para incrementos temporales de 5 ns y de ~8.2 fotones para 3 ns de resolución temporal. De esto, se demuestra que conforme se vuelve menor la intensidad de corriente en un LED (valores de resistencia mayores), el control resulta mejor, sin embargo, como ya se mencionó, se paga el precio de utilizar anchos de pulso mayores, con la clara desventaja, en el número de LEDs que pueden ser encendidos dentro de un mismo GTU.

Validación con hardware de 8 canales

El hardware se planteó, diseñó y probó para que sea sencillo y flexible, pero a su vez, robusto y confiable, cumpliendo con las características que el método de control de intensidad PWV requiere, a partir de hardware reconfigurable a base de dispositivos FPGA.

El prototipo Track-Flat validó la arquitectura que crea las señales PWV en 8 canales independientes (8 LEDs), mostrando que, tanto las simulaciones realizadas como las comprobaciones medidas en laboratorio, son consistentes, con lo que es posible realizar un diseño que se apegue a la realidad con una muy buena aproximación a partir del código en el lenguaje de descripción de hardware (en este caso VHDL), antes de pasar a la implementación en el dispositivo, lo cual es una enorme ventaja, ya que permite ahorrar tiempo en el proceso de diseño, además de garantizar la correcta operación, así como permitir la mejor detección de errores al momento de desarrollar el código.

También se validó el comportamiento de los LEDs en diferentes configuraciones de los dispositivos que se utilizan como *switches*. En los regímenes en que se operan los *switches* que alimentarán a los LEDs, la frecuencia de operación no es la verdadera limitante, sino los tiempos de encendido y apagado que los dispositivos a utilizar deben cumplir, pues las señales que realizarán los encendidos y apagados en los LEDs, no son periódicas, sino pulsos que se manifiestan en los patrones que un *track* requiera. Sin embargo, no se llegará a utilizar anchos de pulso menores a 30 ó 40 ns, debido a la propia naturaleza de la carga LED (respuesta en forma de un circuito RC), de modo que hasta que no se alcanza el nivel requerido para la emisión (alrededor de 2.5V en las terminales del LED) y se mantiene por

encima de éste, no habrá fotones detectados. Más aún, dependiendo del valor de resistencia serie que se coloque en el LED, se requerirá de un pulso de mayor ancho para comenzar a emitir luz, siendo en todos los casos, pulsos que van de algunas decenas a algunos cientos de nanosegundos, por lo que, para los dispositivos seleccionados como *switches*, los tiempos de encendido y apagado (*rise time* y *fall time*, respectivamente), no se ven comprometidos.

Se observó que el valor de la resistencia en serie con el LED depende enteramente de la guía óptica y su acoplamiento con el dispositivo MAPMT, para el caso del prototipo Track-Flat, y de la guía final y su acoplamiento con la estructura PDM en el caso del Track-Sim. Se observó que tanto los materiales utilizados, como la manera de acoplar la guía óptica con el dispositivo detector de luz, tienen un gran impacto en la curva de respuesta de intensidad contra ancho de pulso, pudiendo generar tanto corrimientos temporales como en intensidad en dicha curva, con alteraciones de canal a canal de manera considerable. Para pequeñas variaciones es posible realizar correcciones mediante el ajuste de la propia resistencia en serie con el LED, como en el caso de la obtención de un *track* plano, en donde, ante importantes dificultades en alineamiento entre la guía óptica y el dispositivo MAPMT, tras diversos ajustes en alineación, se pueden compensar las diferencias persistentes al modificar el valor de resistencia, siempre y cuando, tales diferencias no sean grandes. En el mejor de los casos, se busca que todos los canales emitan con la misma intensidad para el mismo ancho de pulso, es decir, se tenga una sola recta de ajuste para todos los canales en la región lineal de la curva de intensidad, lo cual fue probado con el prototipo Track-Flat al buscar que todos los canales emitan de forma casi idéntica con el mismo ancho de pulso utilizado (para más detalles ver César Tavera, et al., 2011).

Hardware para control de matriz en el instrumento Track-Sim

La arquitectura creada para el modelo final, permitirá la ejecución de cualquier tipo de *track* a simular dentro de una estructura PDM con aproximadamente 2300 pixeles (matriz de 48 × 48 LEDs). Se planeó la arquitectura a fin de maximizar la cantidad de recursos dentro del dispositivo FPGA en forma de memorias RAM para cumplir con la capacidad de

almacenamiento de hasta los *tracks* más extensos, así como permitir utilizar mayores resoluciones temporales o incrementos de ancho de pulso, siendo éstos últimos función del reloj del sistema en el FPGA.

Dado este requisito, la comunicación paralela se volvió la opción más factible a fin de aprovechar los recursos en su mayoría para el proceso de almacenamiento, con lo que se establece también la forma en que debe realizarse el programa de control, y más precisamente, la interfaz que hará la comunicación entre el programa y el hardware del sistema FPGA. Como tal interfaz debe ser confiable, versátil y robusta, la comunicación USB resulta una buena opción, a partir de lo cual se dicta que, dicha interfaz de comunicación, debe realizar, en términos simples, una operación de transferencia de datos de forma serial (saliendo mediante USB desde el programa de control) a paralelo (llegando al sistema FPGA). Esta interfaz puede ser realizada mediante diferentes opciones como microcontroladores, CPLDs, e inclusive sistemas existentes como los dispositivos NI de puertos digitales, que por un lado ya incorporan comunicación USB y por el otro puertos paralelos, sin embargo, esta es una de las opciones que quedan abiertas en lo futuro y que, dependiendo de las facilidades, costo y tiempo, se optará por la mejor solución. Dentro de estas tres opciones, quizá por tiempo la que mejor se encuentra posicionada es utilizar un sistema existente de NI, tanto por costo, simplicidad y robustez, además de que es posible realizar, de igual manera, el programa de control utilizando la herramienta de desarrollo LabVIEW® de NI, la cual, en su versión Linux, ha mostrado indicios de buen rendimiento bajo Ubuntu, así como la facilidad de desarrollo de la interfaz gráfica de usuario en un programa ejecutable sin necesidad de tener LabVIEW instalado, sin embargo, esto se encuentra abierto a cualquier plataforma, tanto de software como de hardware.

Por otra parte, de las simulaciones realizadas, se observaron, dos problemáticas:

1. Ruido de alta frecuencia y oscilaciones en las señales provenientes del FPGA de la tarjeta ProASIC 3E, situación que se resolvió al cambiar directamente el reloj del sistema por uno más estable y, en el caso de las oscilaciones, al introducir un circuito *Schmitt Trigger*.

2. Aparición de *glitches*. Esta situación apareció como la principal diferencia entre los dos dispositivos FPGA utilizados, y se encuentra relacionada con el proceso de síntesis en las herramientas de cada fabricante de FPGA. El entendimiento y la correcta elección de los parámetros en las herramientas de diseño, así como a partir de un buen diseño de las máquinas de estados eliminan esta situación.

Trabajo futuro

Para completar el sistema de control del instrumento Track-Sim, queda pendiente la implementación del programa de control. Para esto se diseñaron los diferentes algoritmos, tanto de procesos de comunicación y control, así como de conversión de los *tracks* que conformarán las librerías, los cuales pueden ser revisados dentro del apéndice 4.

Estos algoritmos permitirán, una vez elegida la herramienta mediante la cual se programará el software de control, llevar a la práctica todos los procesos que operarán al instrumento Track-Sim. Para ello, fueron tomados en consideración las principales funciones que se espera realice el usuario con el instrumento, sin embargo, dado que se cuenta únicamente con las etapas diseñadas antes de pasar a la implementación, se está en la posibilidad de agregar y modificar etapas y funcionalidades, como lo es, por ejemplo, el incluir una etapa de realimentación desde el sistema de calibración como aparece en la figura 2.9 de la sección 2.4.3, la cual, podría pensarse como una etapa automatizada e intermedia antes de realizar la ejecución de cualquier *track*, sin embargo, tales consideraciones deben de sujetarse a disponibilidad completa de dicho sistema de calibración, el cual, pensando en la operación del instrumento Track-Sim completo, requerirá de un sistema de adquisición multicanal, capaz de leer la cantidad de señales obtenidas de un PDM, en incluso, la disposición del mismo PDM para efectos de prueba, ó, en su defecto, quizá un sistema de calibración reducido, el cual contenga un mínimo de canales que permitan, de manera confiable, conocer el estado de generación de fotones del instrumento, o implementar dicha interfaz con el sistema de calibración para una calibración previa al funcionamiento del instrumento. Otra opción es agregar alguna etapa de automatización de pruebas, en que se coloque al bloque PDM dentro de la cámara que permita su alineación con el instrumento

Track-Sim. Todas estas son sólo algunas opciones que pueden agregar funcionalidad al instrumento y que pueden incluirse dentro del programa de control, sin embargo, como se ha mencionado, una de las directrices que ha fundamentado y permeado en todo momento el presente trabajo, recae en la realización de un sistema práctico, sencillo y fácil de utilizar pero a la vez versátil dentro de la aplicación para la que fue concebido.

Finalmente, es importante señalar que, además de la importante contribución en cuanto a la calibración de los algoritmos de *trigger* del telescopio JEM-EUSO, lo cual permitirá, por una parte, la evaluación en cuanto a cuestiones de reducción en la transmisión de datos desde la Estación Espacial Internacional, y por otra la, una contribución en la detección fiable y eficiente de EECRs, la realización de un instrumento como lo es el Track-Sim o el Track-Flat como versión reducida, constituye en sí la creación de una herramienta instrumental jamás realizada para la simulación de trazas reales de partículas de rayos cósmicos, por lo que este trabajo representa un desarrollo único en su tipo. De esta forma se brinda la posibilidad de simular, bajo las condiciones reales esperadas, el comportamiento de detectores, tanto para desarrollos de observatorios terrestres como espaciales, así como sistemas afines multicanal que utilicen fotomultiplicadores.

Apéndices

Apéndice 1: LED datasheet (VAOL-5EUV0T4)

VCC | OPTOELECTRONICS
lighting:theway

UV LED LAMP

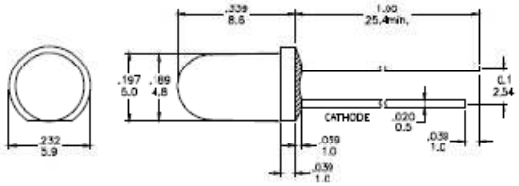
VAOL-5EUV0T4

Feature
- Low Power Consumption
- I.C. compatible

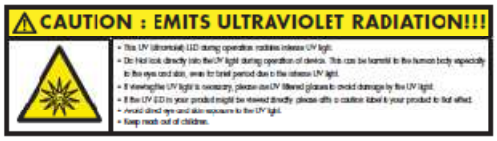
Applications
- Disinfection and Sterilization
- Adhesive Curing
- Leak Detection
- Authentication

Description
- These LEDs are Based on InGaN Material Technology
- Emitted color: Purple (UV)
- Water Transparent Lens

Package Dimension



*Tolerance : ± 0.01 / 0.25 Unit : ± inch / mm



Absolute Maximum Ratings at Ta=25°C

Symbol	Parameter	Max.	Unit
PD	Power Dissipation	120	mW
VR	Reverse Voltage	5	V
IAF	Average Forward Current	30	mA
IPF	Peak Forward Current (Duty=0.1, 1kHz)	100	mA
—	Derating Linear From 25°C	0.4	mA/°C
Topr	Operating Temperature Range	–20 to + 80	°C
Tstg	Storage Temperature Range	–20 to + 100	°C
Lead Soldering Temperature [1.6mm (0.063inch) From Body] 260°C For 5 Seconds.			

Electrical / Optical Characteristics and Curves at Ta=25°C

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Unit
VF	Forward Voltage	IF= 20 mA	2.8	3.0	3.6	V
IR	Reverse Current	VR= 5 V			50	μA
Λ θ	Half Intensity Angle	IF= 20 mA	10	15	20	Deg
IV	Luminous Intensity	IF= 20 mA	--	200	--	mcd.
λ p	Peak Wavelength	IF= 20 mA	400	405		nm

VCC | OPTOELECTRONICS
lighting:theway

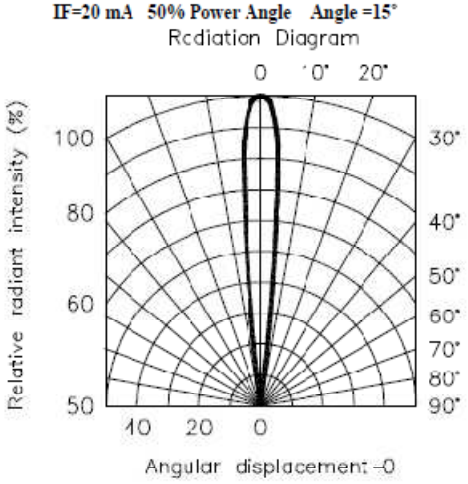
Electrical Characteristics at Ta=25℃

Symbol	Iv		VF		λp	
Parameter	Luminous Intensity		Forward Voltage		Peak Wavelength	
Condition	IF=20mA		IF=20mA		IF=20mA	
Unit	mcd		V		nm	
Binning	Grade	Range	Grade	Range	Grade	Range
	BIN10	125~175	P0	2.8~3.0	U6	400~405
	BIN11	175~245	P1	3.0~3.2	U7	405~410
			P2	3.2~3.4		
			P3	3.4~3.6		

Intensity: Tolerance of minimum and maximum = ± 15%
Vf: Tolerance of minimum and maximum = ± 0.05v
NOTE:
1. Static electricity and surge damages the LED. It is recommend to use a anti-static wrist band or anti-electrostatic glove when handing the LEDs. All devices, equipment and machinery must be properly grounded.

Radiation Diagram

IF=20 mA 50% Power Angle Angle =15°

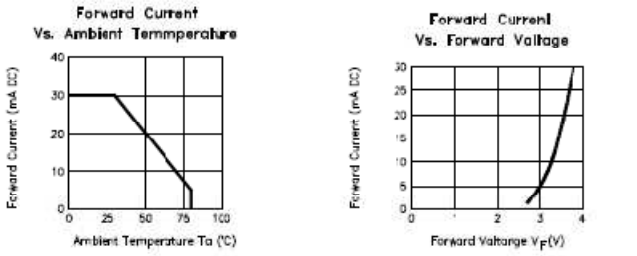


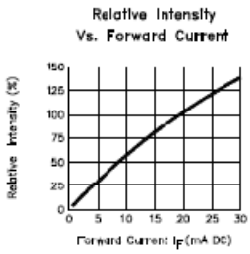
VCC | OPTOELECTRONICS
lighting:theway

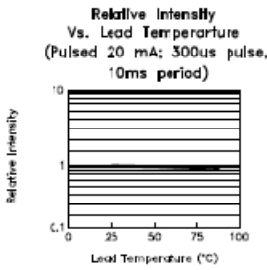
UV

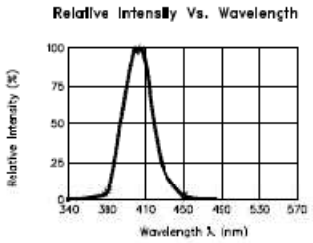
Typical Electro-optical Characteristic Curves

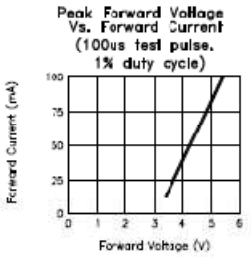
(25°C Free Air Temperature Unless Otherwise Specified)











Apéndice 2A: Código VHDL de la arquitectura del instrumento Track-Sim

```
--- tracksim_v1.0.vhd

---------- SISTEMA PREVIO DE TRACKSIM CON MEMORIAS RAM DE TAMAÑO 16 x 48 --------
--

 ----------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
--use IEEE.numeric_std.all;
 ----------------------------------------------------------------
 ENTITY tracksim_v3 IS
 GENERIC (bits: INTEGER := 8;                              -- # de bits por palabra
          words1: INTEGER := 16;                           -- # de palabras de
la(s) memoria(s) ram  1 (del reloj de 40 MHz(25ns): 2.5us/25ns = 100, que por 8
LEDs da 800)
          words2: INTEGER := 16;                           -- # de palabras de
la(s) memoria(s) ram  2 (del reloj de 40 MHz(25ns): 2.5us/25ns = 100, que por 8
LEDs da 800)
          num_leds: INTEGER := 48;                         -- # de LEDs a manejar
por lado de la matriz (48 para Tracksim completo)
          cero: STD_LOGIC_VECTOR :=
"000000000000000000000000000000000000000000000000";  -- para el # de bits de
salida de cada memoria (48)
          plano: STD_LOGIC_VECTOR :=
"000000000000000000000000110000000000000000000000");-- para habilitar líneas para
track plano

    PORT (clk : IN STD_LOGIC;
          reset : IN STD_LOGIC;                            -- reset general (manual
y/o por software)
          --wr_ena : IN STD_LOGIC;
          mode : IN STD_LOGIC_VECTOR(1 downto 0);          -- 2 bits de control de
modo de operación
          byte_addr : IN STD_LOGIC_VECTOR(2 DOWNTO 0);     -- 3 bits para
direccionar posición de ram (selección de submemorias)
          select_mem : IN STD_LOGIC;                       -- 1 bit para selección
de memoria (conjunto de submemorias)
          seq_sel: IN STD_LOGIC;                           -- 1 bit para selección
de secuencia (secuencia en memorias o track plano)
          repeat: IN STD_LOGIC;                            -- bit para seleción
entre reproducción continua o 1 vez
          dir: IN STD_LOGIC;
          addr: IN STD_LOGIC_VECTOR (3 DOWNTO 0);          -- 4 bits de
direccionamiento para 16 localidades
          data_in: IN STD_LOGIC_VECTOR (bits - 1 DOWNTO 0);
          pc_do : IN STD_LOGIC;                            -- bit de flag de
control desde PC
          pc_ready : IN STD_LOGIC;                         -- bit de falg de
control desde PC
          ultimo: IN STD_LOGIC;                            -- bit que indica si se
ha cargado el último dato

    ----- Salidas -----
```

```
          fpga_ready: OUT STD_LOGIC;                                          --
Salida del FPGA a la PC
          salida_1: OUT STD_LOGIC_VECTOR (num_leds - 1 DOWNTO 0);             --
Salida de la memoria ram horizontal (48 bits de salida)
          salida_2: OUT STD_LOGIC_VECTOR(num_leds - 1 DOWNTO 0);              --
Salida de memoria ram vertical (48 bits de salida)
          data_read: OUT STD_LOGIC_VECTOR(bits - 1 DOWNTO 0));                --
Salida para lectura de datos en memorias

 END tracksim_v3;

 Architecture descripcion of tracksim_v3 is

    Type vector_array_1a is array (0 to words1 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_1a: vector_array_1a;

    Type vector_array_1b is array (0 to words1 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_1b: vector_array_1b;

    Type vector_array_1c is array (0 to words1 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_1c: vector_array_1c;

    Type vector_array_1d is array (0 to words1 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_1d: vector_array_1d;

    Type vector_array_1e is array (0 to words1 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_1e: vector_array_1e;

    Type vector_array_1f is array (0 to words1 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_1f: vector_array_1f;

    Type vector_array_2a is array (0 to words2 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_2a: vector_array_2a;

    Type vector_array_2b is array (0 to words2 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_2b: vector_array_2b;

    Type vector_array_2c is array (0 to words2 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_2c: vector_array_2c;

    Type vector_array_2d is array (0 to words2 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_2d: vector_array_2d;

    Type vector_array_2e is array (0 to words2 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_2e: vector_array_2e;

    Type vector_array_2f is array (0 to words2 - 1) of
         STD_LOGIC_VECTOR (bits - 1 downto 0);
    Signal memory_2f: vector_array_2f;
```

```
    TYPE estados IS (inicio, recepcion, recibe_dato, ultimo_dato, transmision,
envia_dato, ultimo_dato_tr, correr_sec, continuo, una_vez, fin_secuencia); -- Se
definen los estados

    Signal correr : std_logic;
    Signal mem_data_out_1a : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_1b : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_1c : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_1d : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_1e : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_1f : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_2a : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_2b : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_2c : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_2d : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_2e : std_logic_vector(bits - 1 downto 0);
    Signal mem_data_out_2f : std_logic_vector(bits - 1 downto 0);

    Signal q_clk_plano: std_logic;                                          --
Reloj para contador2
    Signal clk_plano_aux: integer range 0 to 51;                            --
Cuenta del divisor de frecuencia (clock2)
    Signal plano_aux : integer range 0 to num_leds;                         --
Lleva la cuenta del contador2
    Signal senal1 : std_logic;                                              --
bit de ayuda para saber fin de modo 1 vez
    Signal senal2 : std_logic;                                              --
bit de ayuda para saber fin de modo 1 vez
    Signal track_plano : std_logic_vector (num_leds - 1 downto 0);          --
Salida hacia el multiplexor
    Signal track_plano_out : std_logic_vector(num_leds -1 downto 0);        --
Salida del reproductor de track plano (secuencia B) hacia multiplexor
    Signal aux1_addr1 : integer range 0 to words1 - 1;                      --
Lleva la cuenta del contador 1

    signal parar : std_logic;                                               --
Para detener el modo continuo (modo 1 vez)

    SIGNAL presente : estados;                                              --
Señal para manjo de estados

 Begin

 ---------- Asignación de estados ----------
estado:
process (clk, reset, mode, repeat, dir, seq_sel, pc_do, pc_ready, ultimo, senal1,
senal2)
    begin
      if (reset ='0') then
         presente <= inicio;
      elsif (clk = '1' and clk'event) then
         case (presente) is
              when inicio => if (mode = "00") then                  -- no hace
nada y se queda a la espera del modo de operación
                                presente <= inicio;
                             elsif (mode = "01") then               -- modo
recepción de datos desde PC
                                presente <= recepcion;
                             elsif (mode = "10") then               -- modo
transmisión de datos a PC
                                presente <= transmision;
                             else                                   -- mode =
"11" para correr secuencia
```

```
                                presente <= correr_sec;
                            end if;

---------- ASIGNACIÓN DE SALIDAS PARA MODO RECEPCIÓN DE DATOS (mode = "01") -----
-----

            when recepcion => if (reset = '0') then                -- Si se
resetea el sistema se vuelve a inicio (reset manual o por software)
                                presente <= inicio;
                              elsif (pc_ready = '0' and ultimo = '0') then  --
se espera para ir al siguiente estado
                                presente <= recepcion;
                              elsif (pc_ready = '0' and ultimo = '1') then  --
se espera para ir al siguiente estado
                                presente <= recepcion;
                              elsif (pc_ready = '1' and ultimo = '0') then  --
"pc_ready = '1' and ultimo = '0'", se indica ir a "recibe_dato"
                                presente <= recibe_dato;
                              else                                         --
"pc_ready = '1' and ultimo = '1'", se indica ir a "ultimo_dato"
                                presente <= ultimo_dato;
                              end if;

          ----- Asignación de estados para recibe_dato -----

            when recibe_dato => if (reset = '0') then                -- Si se
resetea el sistema se vuelve a inicio (reset manual o por software)
                                  presente <= inicio;
                                elsif (pc_ready = '0' and pc_do = '0') then
-- se espera para ir a dato_sig
                                  presente <= recepcion;
                                elsif (pc_ready = '0' and pc_do = '1') then
-- se espera para ir a dato_si
                                  presente <= recibe_dato;
                                elsif (pc_ready = '1' and pc_do = '0') then
-- se va a dato_sig
                                  presente <= recibe_dato;
                                else                                     --
pc_ready = '1' and pc_do = 1, espera para ir a dato_sig
                                  presente <= recibe_dato;
                                end if;

          ----- Asignación de estados para ultimo_dato -----

            when ultimo_dato => if (reset = '0') then                -- Hasta
que no se presente un "reset", se permanece en modo continuo
                                  presente <= inicio;
                                elsif (pc_ready = '0') then
                                  presente <= inicio;                -- Fin de
la carga de datos
                                else                                     --
pc_ready = 1 se queda a la espera de indicación de fin de carga de datos
                                  presente <= ultimo_dato;
                                end if;


---------- ASIGNACIÓN DE ESTADOS PARA MODO TRANSMISIÓN DE DATOS (mode = "10") ---
-------

            when transmision => if (reset = '0') then                -- Si se
resetea el sistema se vuelve a inicio (reset manual o por software)
                                  presente <= inicio;
```

```
                                elsif (pc_ready = '0' and ultimo = '0') then
-- Espera por pc_ready = 1 para pasar a envía_dato
                                    presente <= transmision;
                                elsif (pc_ready = '0' and ultimo = '1') then
                                    presente <= transmision;
                                elsif (pc_ready = '1' and ultimo = '0') then
-- pc_ready = '1' and ultimo = '0', espera para ir a "envia_dato" (pc_do en cero)
                                    presente <= envia_dato;
                                else                                    --
pc_ready = '1' and ultimo = '1', va a "ultimo_dato_tr"
                                    presente <= ultimo_dato_tr;
                                end if;

          ----- Asignación de estados para envia_dato -----

              when envia_dato => if (reset = '0') then                    -- Si
se resetea el sistema se vuelve a inicio (reset manual o por software)
                                   presente <= inicio;
                                elsif (pc_ready = '0' and pc_do = '0') then    -
- Se va a dato siguiente en modo transmisión
                                    presente <= transmision;
                                elsif (pc_ready = '0' and pc_do = '1') then    -
- Permanece en "envia_dato" hasta ver 0 en pc_do
                                    presente <= envia_dato;
                                elsif (pc_ready = '1' and pc_do = '0') then    -
- Se va a ultimo_dato_tr (se informa al FPGA)
                                    presente <= envia_dato;           -- Se
mandó el último dato (se informa a PC)
                                else                                        --
reset = 1 & pc_ready = 1 & pc_do = 1 (espera para pasar a "dato_sig_tr" ó
"ultimo_dato_tr")
                                    presente <= envia_dato;
                                end if;

          ----- Asignación de estados para ultimo_dato_tr -----

              when ultimo_dato_tr => if (reset = '0') then        -- Si se
resetea el sistema se vuelve a inicio (reset manual o por software)
                                       presente <= inicio;
                                   elsif (pc_ready = '0') then    -- Se va a
"inicio" (fin de la transmisión de datos)
                                       presente <= inicio;
                                   else                           -- pc_ready =
1, espera por instrucción para ir a "inicio" (con esto marca el fin de la
transmisión de datos)
                                       presente <= ultimo_dato_tr;
                                   end if;


---------- ASIGNACIÓN DE ESTADOS PARA MODO DE CORRER SECUENCIA (mode = "11") ----
------

              when correr_sec => if (reset = '0') then        -- Si se resetea el
sistema se vuelve a inicio (reset manual o por software)
                                   presente <= inicio;
                                elsif (mode = "11" and repeat = '0') then
-- Espera por instrucción desde PC para correr la secuencia (posterior a la
configuración del tipo de secuencia en el software de control)
                                   presente <= una_vez;
                                elsif (mode = "11" and repeat = '1') then
                                   presente <= continuo;
                                else                                        --
pc_do = 1 and repeat = 1, se va a "contunuo"
```

```
                                    presente <= inicio;
                                end if;

          ----- Asignación de estados para reproducción 1 vez -----

              when una_vez => if (reset = '0') then
                                    presente <= inicio;
                                elsif (seq_sel = '0') then                          -
- Track en memorias
                                    if (senal1 = '1') then
                                       presente <= fin_secuencia;                   -
- Se finaliza la reproducción de la traza
                                    else
                                       presente <= una_vez;
                                    end if;
                                else                                                -
- "seq_sel = 1", se finaliza la reproducción del track plano
                                    if (senal2 = '1') then
                                       presente <= fin_secuencia;
                                    else
                                      presente <= una_vez;
                                    end if;
                                end if;


              when fin_secuencia => if (reset = '0') then
                                       presente <= inicio;
                                    elsif (mode = "11") then                        -
- permanece en "fin de secuencia"
                                       presente <= fin_secuencia;
                                    else                                            -
- Se va a inicio (se indica desde la PC con "mode = "00"")
                                       presente <= inicio;
                                    end if;

          ----- Asignación de estados para reproducción continua -----

              when continuo => if (reset = '0') then
                                    presente <= inicio;
                               elsif (mode = "11") then                     --
Espera para ir a fin de secuencia
                                    presente <= continuo;
                               else                                         -- Se
va a inicio (se endica desde la PC indicado por mode = "00" u otro valor)
                                    presente <= inicio;
                               end if;

         end case;
      end if;
end process estado;

---------- ASIGNACIÓN DE SALIDAD EN LOS ESTADOS ----------

salidas_estados:
process (presente)
    begin
         case (presente) is

              when inicio => fpga_ready <= '0'; parar <= '0';

         ---------- Salidas para modo recepción de datos ----------

              when recepcion => fpga_ready <= '1';
```

```
                                -- parar <= '0';

              when recibe_dato => fpga_ready <= '0';
                                  -- parar <= '0';

              when ultimo_dato => fpga_ready <= '0';
                                  -- parar <= '0';

          ---------- Salidas para modo transmisión de datos ----------

              when transmision => fpga_ready <= '1'; -- parar <= '0';

              when envia_dato => fpga_ready <= '0';

              when ultimo_dato_tr => fpga_ready <= '0';

             --  when dato_sig_tr => fpga_ready <= '1';

          ---------- Salidas para modo correr secuencia ----------

              when correr_sec => fpga_ready <= '1'; parar <= '0';

              when una_vez => fpga_ready <= '1'; parar <= '0';

              when fin_secuencia => fpga_ready <= '0'; parar <= '1';

              when continuo => fpga_ready <= '1'; parar <= '0';

          end case;
end process salidas_estados;

 ---------- Memorias RAM ----------
 memorias_ram:
    process (clk, reset, mode, select_mem, byte_addr, pc_do)
    Begin
       if (reset = '0') then
           correr <= '0';
           data_read <= "00000000";                                          --
cero en bits de lectura hacia la PC
       else
         if (pc_do = '1') then                                               --
bit (flag) desde PC para indicar escritura
           if (mode = "00") then                                             --
modo de paro (no hacer nada)
              correr <= '0';
              data_read <= "00000000";                                       --
cero en bits de lectura hacia la PC
           elsif (mode = "01") then
-- modo escritura en memoria(s)
                data_read <= "00000000";                                     --
cero en bits de lectura hacia la PC
              if (select_mem = '0' and byte_addr = "000") then
                 if (clk'event and clk = '1') then
                     memory_1a(conv_integer(unsigned(addr))) <= data_in;     --
guarda en memory_1a
                 end if;
              elsif (select_mem = '0' and byte_addr = "001") then
                 if (clk'event and clk = '1') then
                     memory_1b(conv_integer(unsigned(addr))) <= data_in;     --
guarda en memory_1b
                 end if;
              elsif (select_mem = '0' and byte_addr = "010") then
                 if (clk'event and clk = '1') then
```

```
                        memory_1c(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_1c
                    end if;
                elsif (select_mem = '0' and byte_addr = "011") then
                    if (clk'event and clk = '1') then
                        memory_1d(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_1d
                    end if;
                    elsif (select_mem = '0' and byte_addr = "100") then
                    if (clk'event and clk = '1') then
                        memory_1e(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_1e
                    end if;
                elsif (select_mem = '0' and byte_addr = "101") then
                    if (clk'event and clk = '1') then
                        memory_1f(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_1f
                    end if;
                elsif (select_mem = '1' and byte_addr = "000") then
                    if (clk'event and clk = '1') then
                        memory_2a(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_2a
                    end if;
                elsif (select_mem = '1' and byte_addr = "001") then
-- (select_mem = '1' and byte_addr = "000")
                    if (clk'event and clk = '1') then
                        memory_2b(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_2b
                    end if;
                elsif (select_mem = '1' and byte_addr = "010") then
                    if (clk'event and clk = '1') then
                        memory_2c(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_2c
                    end if;
                elsif (select_mem = '1' and byte_addr = "011") then
                    if (clk'event and clk = '1') then
                        memory_2d(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_2d
                    end if;
                elsif (select_mem = '1' and byte_addr = "100") then
                    if (clk'event and clk = '1') then
                        memory_2e(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_2e
                    end if;
                elsif (select_mem = '1' and byte_addr = "101") then
                    if (clk'event and clk = '1') then
                        memory_2f(conv_integer(unsigned(addr))) <= data_in;      --
guarda en memory_2f
                    end if;
                else                                                             --
los demás casos no importan
                        correr <= '0';
                end if;
            elsif (mode ="10") then
-- modo de lectura de memoria(s)
                if (select_mem = '0' and byte_addr = "000") then
                    if (clk'event and clk = '1') then
                        data_read <= memory_1a(conv_integer(unsigned(addr)));    --
guarda en memory_1a
                    end if;
                elsif (select_mem = '0' and byte_addr = "001") then
                    if (clk'event and clk = '1') then
```

```
                    data_read <= memory_1b(conv_integer(unsigned(addr)));    --
guarda en memory_1b
                end if;
            elsif (select_mem = '0' and byte_addr = "010") then
                if (clk'event and clk = '1') then
                    data_read <= memory_1c(conv_integer(unsigned(addr)));    --
guarda en memory_1c
                end if;
            elsif (select_mem = '0' and byte_addr = "011") then
                if (clk'event and clk = '1') then
                    data_read <= memory_1d(conv_integer(unsigned(addr)));    --
guarda en memory_1d
                end if;
                elsif (select_mem = '0' and byte_addr = "100") then
                if (clk'event and clk = '1') then
                    data_read <= memory_1e(conv_integer(unsigned(addr)));    --
guarda en memory_1e
                end if;
            elsif (select_mem = '0' and byte_addr = "101") then
                if (clk'event and clk = '1') then
                    data_read <= memory_1f(conv_integer(unsigned(addr)));    --
guarda en memory_1f
                end if;
            elsif (select_mem = '1' and byte_addr = "000") then
                if (clk'event and clk = '1') then
                    data_read <= memory_2a(conv_integer(unsigned(addr)));    --
guarda en memory_2a
                end if;
            elsif (select_mem = '1' and byte_addr = "001") then
-- (select_mem = '1' and byte_addr = "000")
                if (clk'event and clk = '1') then
                    data_read <= memory_2b(conv_integer(unsigned(addr)));    --
guarda en memory_2b
                end if;
            elsif (select_mem = '1' and byte_addr = "010") then
                if (clk'event and clk = '1') then
                    data_read <= memory_2c(conv_integer(unsigned(addr)));    --
guarda en memory_2c
                end if;
            elsif (select_mem = '1' and byte_addr = "011") then
                if (clk'event and clk = '1') then
                    data_read <= memory_2d(conv_integer(unsigned(addr)));    --
guarda en memory_2d
                end if;
            elsif (select_mem = '1' and byte_addr = "100") then
                if (clk'event and clk = '1') then
                    data_read <= memory_2e(conv_integer(unsigned(addr)));    --
guarda en memory_2e
                end if;
            elsif (select_mem = '1' and byte_addr = "101") then
                if (clk'event and clk = '1') then
                    data_read <= memory_2f(conv_integer(unsigned(addr)));    --
guarda en memory_2f
                end if;
            else                                                             --
los demás casos no importan
                    correr <= '0';
            end if;
                correr <= '0';
        else                                                                 --
"11" modo de reproducción de secuencia(s)
                correr <= '1';
```

```
                    data_read <= "00000000";                                    --
cero en bits de lectura hacia la PC
            end if;
          end if;
        end if;
    end process memorias_ram;

 ---------- Reloj para track plano ----------

clock_plano:
process (clk, reset, mode)

begin

        if (reset = '0') then
            clk_plano_aux <= 0;
            q_clk_plano <= '0';
        else
            if (mode = "11") then
                if (clk'event and clk = '1') then
                    clk_plano_aux <= clk_plano_aux + 1;
                    if (clk_plano_aux = 50) then
                        q_clk_plano <= not q_clk_plano;                         --
Se cambia la señal q_clk2 después de cierto # de ciclo de reloj (división de
frecuencia en este caso por 100)
                        clk_plano_aux <= 1;                                     --
Se reestablece el valor de la cuenta a uno (con esto se evitan medios periodos de
menor duración al inicio)
                    end if;
                end if;
            else
                clk_plano_aux <= 0;
                q_clk_plano <= '0';
            end if;
        end if;
end process clock_plano;

---------- Contador para track plano ----------

contador_plano:
process (q_clk_plano, reset, mode, seq_sel, dir, plano_aux)       -- Contador de
ciclos del reloj para obtener nueva frecuencia

begin

if (reset = '0') then
    plano_aux <= 0;                                                 -- Reset para
reinicio a cero
    senal2 <= '0';
else
    if (mode = "11") then
        if (seq_sel = '0') then
            plano_aux <= 0;
            senal2 <= '0';
        elsif (q_clk_plano'event and q_clk_plano = '1') then
           if (repeat = '0') then                                           -- Para
reproducir una vez
              if (dir = '0') then
                 if (plano_aux = num_leds) then
                     plano_aux <= 1;                                        -- Reinicia a 1
después de completada una cuenta (16 GTUs, 1 GTU por LED)
                    senal2 <= '1';                                          -- Señal
auxiliar para determinar el fin para modo una vez
```

```
                else
                    plano_aux <= plano_aux + 1;                    -- Cálculo de
avance ascendente
                end if;
             else
                if (plano_aux = 0) then
                    plano_aux <= num_leds;                         -- Reinicia a
16 después de completada una cuenta (16 GTUs, 1 GTU por LED)
                elsif (plano_aux = 1) then
                       plano_aux <= num_leds;
                       senal2 <= '1';                                -- Señal
auxiliar para determinar el fin para modo una vez
                else
                       plano_aux <= plano_aux - 1;                 -- Cálculo de
avance descendente
                end if;
             end if;
          else                                                     -- Para
reproducción continua
             if (dir = '0') then
                if (plano_aux = num_leds) then
                    plano_aux <= 1;                                -- Reinicia a 1
después de completada una cuenta (16 GTUs, 1 GTU por LED)
                    --senal2 <= '1';                               -- Señal
auxiliar para determinar el fin para modo una vez
                else
                    plano_aux <= plano_aux + 1;                    -- Cálculo de
avance ascendente
                end if;
             else
                if (plano_aux = 0) then
                    plano_aux <= num_leds;                         -- Reinicia a
16 después de completada una cuenta (16 GTUs, 1 GTU por LED)
                elsif (plano_aux = 1) then
                       plano_aux <= num_leds;
                       --senal2 <= '1';                              -- Señal
auxiliar para determinar el fin para modo una vez
                else
                       plano_aux <= plano_aux - 1;                 -- Cálculo de
avance descendente
                end if;
             end if;
          end if;
       end if;
    else
        plano_aux <= 0;                                            -- Reset para
reinicio a cero
        senal2 <= '0';
    end if;
end if;

end process contador_plano;

---------- Reproductor de track plano (secuencia B) ----------

reproductor_track_plano:
process(clk, reset, mode, seq_sel, plano_aux)

begin

if (reset = '0') then
    track_plano <= cero;
else
```

```
    if (mode = "11") then
      if (seq_sel = '0') then
          track_plano <= cero;
      else                                                    -- Aquí se
realiza el proceso de selección de cada salida
          if (clk'event and clk = '1') then
              if (plano_aux = 1) then
                 track_plano(47 downto 1) <=
"00000000000000000000000000000000000000000000000";     -- 47 ceros
                 track_plano(0) <= '1';
              elsif (plano_aux = 2) then
                    track_plano(47 downto 2) <=
"000000000000000000000000000000000000000000000000";   -- 46 ceros
                    track_plano(0) <= '0';
-- 1 cero
                    track_plano(1) <= '1';
              elsif (plano_aux = 3) then
                    track_plano(47 downto 3) <=
"00000000000000000000000000000000000000000000000";    -- 45 ceros
                    track_plano(1 downto 0) <= "00";
-- 2 ceros
                    track_plano(2) <= '1';
              elsif (plano_aux = 4) then
                    track_plano(47 downto 4) <=
"0000000000000000000000000000000000000000000000";     -- 44 ceros
                    track_plano(2 downto 0) <= "000";
-- 3 ceros
                    track_plano(3) <= '1';
              elsif (plano_aux = 5) then
                    track_plano(47 downto 5) <=
"0000000000000000000000000000000000000000000000";     -- 43 ceros
                    track_plano(3 downto 0) <= "0000";
-- 4 ceros
                    track_plano(4) <= '1';
              elsif (plano_aux = 6) then
                    track_plano(47 downto 6) <=
"0000000000000000000000000000000000000000000000";     -- 42 ceros
                    track_plano(4 downto 0) <= "00000";
-- 5 ceros
                    track_plano(5) <= '1';
              elsif (plano_aux = 7) then
                    track_plano(47 downto 7) <=
"0000000000000000000000000000000000000000000000";     -- 41 ceros
                    track_plano(5 downto 0) <= "000000";
-- 6 ceros
                    track_plano(6) <= '1';
              elsif (plano_aux = 8) then
                    track_plano(47 downto 8) <=
"0000000000000000000000000000000000000000000000";     -- 40 ceros
                    track_plano(6 downto 0) <= "0000000";
-- 7 ceros
                    track_plano(7) <= '1';
              elsif (plano_aux = 9) then
                    track_plano(47 downto 9) <=
"0000000000000000000000000000000000000000000000";     -- 39 ceros
                    track_plano(7 downto 0) <= "00000000";
-- 8 ceros
                    track_plano(8) <= '1';
              elsif (plano_aux = 10) then
                    track_plano(47 downto 10) <=
"0000000000000000000000000000000000000000000000";     -- 38 ceros
                    track_plano(8 downto 0) <= "000000000";
-- 9 ceros
```

```
                track_plano(9) <= '1';
            elsif (plano_aux = 11) then
                track_plano(47 downto 11) <=
"00000000000000000000000000000000000000000";           -- 37 ceros
                track_plano(9 downto 0) <= "0000000000";
-- 10 ceros
                track_plano(10) <= '1';
            elsif (plano_aux = 12) then
                track_plano(47 downto 12) <=
"00000000000000000000000000000000000000000";           -- 36 ceros
                track_plano(10 downto 0) <= "00000000000";
-- 11 ceros
                track_plano(11) <= '1';
            elsif (plano_aux = 13) then
                track_plano(47 downto 13) <=
"00000000000000000000000000000000000000000";           -- 35 ceros
                track_plano(11 downto 0) <= "000000000000";
-- 12 ceros
                track_plano(12) <= '1';
            elsif (plano_aux = 14) then
                track_plano(47 downto 14) <=
"00000000000000000000000000000000000000000";           -- 34 ceros
                track_plano(12 downto 0) <= "0000000000000";
-- 13 ceros
                track_plano(13) <= '1';
            elsif (plano_aux = 15) then
                track_plano(47 downto 15) <=
"00000000000000000000000000000000000000000";           -- 33 ceros
                track_plano(13 downto 0) <= "00000000000000";
-- 14 ceros
                track_plano(14) <= '1';
            elsif (plano_aux = 16) then
                track_plano(47 downto 16) <=
"00000000000000000000000000000000000000000";           -- 32 ceros
                track_plano(14 downto 0) <= "0000000000000000";
-- 15 ceros
                track_plano(15) <= '1';
            elsif (plano_aux = 17) then
                track_plano(47 downto 17) <=
"00000000000000000000000000000000000000000";           -- 31 ceros
                track_plano(15 downto 0) <= "00000000000000000";
-- 16 ceros
                track_plano(16) <= '1';
            elsif (plano_aux = 18) then
                track_plano(47 downto 18) <=
"00000000000000000000000000000000000000000";           -- 30 ceros
                track_plano(16 downto 0) <= "00000000000000000";
-- 17 ceros
                track_plano(17) <= '1';
            elsif (plano_aux = 19) then
                track_plano(47 downto 19) <= "00000000000000000000000000000";
-- 29 ceros
                track_plano(17 downto 0) <= "000000000000000000";
-- 18 ceros
                track_plano(18) <= '1';
            elsif (plano_aux = 20) then
                track_plano(47 downto 20) <= "0000000000000000000000000000";
-- 28 ceros
                track_plano(18 downto 0) <= "0000000000000000000";
-- 19 ceros
                track_plano(19) <= '1';
            elsif (plano_aux = 21) then
```

```
                track_plano(47 downto 21) <= "00000000000000000000000000";
-- 27 ceros
                track_plano(19 downto 0) <= "00000000000000000000";
-- 20 ceros
                track_plano(20) <= '1';
            elsif (plano_aux = 22) then
                track_plano(47 downto 22) <= "00000000000000000000000000";
-- 26 ceros
                track_plano(20 downto 0) <= "00000000000000000000";
-- 21 ceros
                track_plano(21) <= '1';
            elsif (plano_aux = 23) then
                track_plano(47 downto 23) <= "00000000000000000000000000";
-- 25 ceros
                track_plano(21 downto 0) <= "00000000000000000000";
-- 22 ceros
                track_plano(22) <= '1';
            elsif (plano_aux = 24) then
                track_plano(47 downto 24) <= "00000000000000000000000000";
-- 24 cero
                track_plano(22 downto 0) <= "00000000000000000000";
-- 23 ceros
                track_plano(23) <= '1';
            elsif (plano_aux = 25) then
                track_plano(47 downto 25) <= "00000000000000000000000000";
-- 23 ceros
                track_plano(23 downto 0) <= "00000000000000000000";
-- 24 ceros
                track_plano(24) <= '1';
            elsif (plano_aux = 26) then
                track_plano(47 downto 26) <= "00000000000000000000000000";
-- 22 ceros
                track_plano(24 downto 0) <= "00000000000000000000";
-- 25 ceros
                track_plano(25) <= '1';
            elsif (plano_aux = 27) then
                track_plano(47 downto 27) <= "00000000000000000000000000";
-- 21 ceros
                track_plano(25 downto 0) <= "00000000000000000000";
-- 26 ceros
                track_plano(26) <= '1';
            elsif (plano_aux = 28) then
                track_plano(47 downto 28) <= "00000000000000000000000000";
-- 20 ceros
                track_plano(26 downto 0) <= "00000000000000000000";
-- 27 ceros
                track_plano(27) <= '1';
            elsif (plano_aux = 29) then
                track_plano(47 downto 29) <= "00000000000000000000000000";
-- 19 ceros
                track_plano(27 downto 0) <= "00000000000000000000";
-- 28 ceros
                track_plano(28) <= '1';
            elsif (plano_aux = 30) then
                track_plano(47 downto 30) <= "00000000000000000000000000";
-- 18 ceros
                track_plano(28 downto 0) <= "00000000000000000000";
-- 29 ceros
                track_plano(29) <= '1';
            elsif (plano_aux = 31) then
                track_plano(47 downto 31) <= "00000000000000000000000000";
-- 17 ceros
```

```
                    track_plano(29 downto 0) <= "00000000000000000000000000000000";
-- 30 ceros
                    track_plano(30) <= '1';
              elsif (plano_aux = 32) then
                    track_plano(47 downto 32) <= "0000000000000000";
-- 16 ceros
                    track_plano(30 downto 0) <=
"00000000000000000000000000000000";                 -- 31 ceros
                    track_plano(31) <= '1';
              elsif (plano_aux = 33) then
                    track_plano(47 downto 33) <= "000000000000000";
-- 15 ceros
                    track_plano(31 downto 0) <=
"00000000000000000000000000000000";                 -- 32 ceros
                    track_plano(32) <= '1';
              elsif (plano_aux = 34) then
                    track_plano(47 downto 34) <= "00000000000000";
-- 14 ceros
                    track_plano(32 downto 0) <=
"00000000000000000000000000000000";                 -- 33 ceros
                    track_plano(33) <= '1';
              elsif (plano_aux = 35) then
                    track_plano(47 downto 35) <= "0000000000000";
-- 13 ceros
                    track_plano(33 downto 0) <=
"00000000000000000000000000000000";                 -- 34 ceros
                    track_plano(34) <= '1';
              elsif (plano_aux = 36) then
                    track_plano(47 downto 36) <= "000000000000";
-- 12 ceros
                    track_plano(34 downto 0) <=
"00000000000000000000000000000000";                 -- 35 ceros
                    track_plano(35) <= '1';
              elsif (plano_aux = 37) then
                    track_plano(47 downto 37) <= "00000000000";
-- 11 ceros
                    track_plano(35 downto 0) <=
"00000000000000000000000000000000";                 -- 36 ceros
                    track_plano(36) <= '1';
              elsif (plano_aux = 38) then
                    track_plano(47 downto 38) <= "0000000000";
-- 10 ceros
                    track_plano(36 downto 0) <=
"00000000000000000000000000000000";                 -- 37 ceros
                    track_plano(37) <= '1';
              elsif (plano_aux = 39) then
                    track_plano(47 downto 39) <= "000000000";
-- 9 ceros
                    track_plano(37 downto 0) <=
"00000000000000000000000000000000";                 -- 38 ceros
                    track_plano(38) <= '1';
              elsif (plano_aux = 40) then
                    track_plano(47 downto 40) <= "00000000";
-- 8 ceros
                    track_plano(38 downto 0) <=
"00000000000000000000000000000000";                 -- 39 ceros
                    track_plano(39) <= '1';
              elsif (plano_aux = 41) then
                    track_plano(47 downto 41) <= "0000000";
-- 7 ceros
                    track_plano(39 downto 0) <=
"00000000000000000000000000000000";                 -- 40 ceros
                    track_plano(40) <= '1';
```

```
            elsif (plano_aux = 42) then
                  track_plano(47 downto 42) <= "000000";
-- 6 ceros
                  track_plano(40 downto 0) <=
"00000000000000000000000000000000000000000";        -- 41 ceros
                  track_plano(41) <= '1';
            elsif (plano_aux = 43) then
                  track_plano(47 downto 43) <= "00000";
-- 5 ceros
                  track_plano(41 downto 0) <=
"0000000000000000000000000000000000000000";       -- 42 ceros
                  track_plano(42) <= '1';
            elsif (plano_aux = 44) then
                  track_plano(47 downto 44) <= "0000";
-- 4 ceros
                  track_plano(42 downto 0) <=
"000000000000000000000000000000000000000000";      -- 43 ceros
                  track_plano(43) <= '1';
            elsif (plano_aux = 45) then
                  track_plano(47 downto 45) <= "000";
-- 3 ceros
                  track_plano(43 downto 0) <=
"0000000000000000000000000000000000000000000";     -- 44 ceros
                  track_plano(44) <= '1';
            elsif (plano_aux = 46) then
                  track_plano(47 downto 46) <= "00";
-- 2 ceros
                  track_plano(44 downto 0) <=
"00000000000000000000000000000000000000000000";    -- 45 ceros
                  track_plano(45) <= '1';
            elsif (plano_aux = 47) then
                  track_plano(47) <= '0';
-- 1 cero
                  track_plano(45 downto 0) <=
"000000000000000000000000000000000000000000000";   -- 46 ceros
                  track_plano(46) <= '1';
             elsif (plano_aux = 48) then
                  track_plano(47) <= '1';
                  track_plano(46 downto 0) <=
"0000000000000000000000000000000000000000000000";  -- 47 ceros
            else
                  track_plano <= cero;
            end if;
          end if;
       end if;
    else
        track_plano <= cero;
    end if;
end if;

end process reproductor_track_plano;

----- Asignación de salidas de track plano -----

track_plano_out(0) <= clk AND track_plano(0);                          --
Asignación de salidas
track_plano_out(1) <= clk AND track_plano(1);
track_plano_out(2) <= clk AND track_plano(2);
track_plano_out(3) <= clk AND track_plano(3);
track_plano_out(4) <= clk AND track_plano(4);
track_plano_out(5) <= clk AND track_plano(5);
track_plano_out(6) <= clk AND track_plano(6);
track_plano_out(7) <= clk AND track_plano(7);
```

```
track_plano_out(8) <= clk AND track_plano(8);
track_plano_out(9) <= clk AND track_plano(9);
track_plano_out(10) <= clk AND track_plano(10);
track_plano_out(11) <= clk AND track_plano(11);
track_plano_out(12) <= clk AND track_plano(12);
track_plano_out(13) <= clk AND track_plano(13);
track_plano_out(14) <= clk AND track_plano(14);
track_plano_out(15) <= clk AND track_plano(15);
track_plano_out(16) <= clk AND track_plano(16);
track_plano_out(17) <= clk AND track_plano(17);
track_plano_out(18) <= clk AND track_plano(18);
track_plano_out(19) <= clk AND track_plano(19);
track_plano_out(20) <= clk AND track_plano(20);
track_plano_out(21) <= clk AND track_plano(21);
track_plano_out(22) <= clk AND track_plano(22);
track_plano_out(23) <= clk AND track_plano(23);
track_plano_out(24) <= clk AND track_plano(24);
track_plano_out(25) <= clk AND track_plano(25);
track_plano_out(26) <= clk AND track_plano(26);
track_plano_out(27) <= clk AND track_plano(27);
track_plano_out(28) <= clk AND track_plano(28);
track_plano_out(29) <= clk AND track_plano(29);
track_plano_out(30) <= clk AND track_plano(30);
track_plano_out(31) <= clk AND track_plano(31);
track_plano_out(32) <= clk AND track_plano(32);
track_plano_out(33) <= clk AND track_plano(33);
track_plano_out(34) <= clk AND track_plano(34);
track_plano_out(35) <= clk AND track_plano(35);
track_plano_out(36) <= clk AND track_plano(36);
track_plano_out(37) <= clk AND track_plano(37);
track_plano_out(38) <= clk AND track_plano(38);
track_plano_out(39) <= clk AND track_plano(39);
track_plano_out(40) <= clk AND track_plano(40);
track_plano_out(41) <= clk AND track_plano(41);
track_plano_out(42) <= clk AND track_plano(42);
track_plano_out(43) <= clk AND track_plano(43);
track_plano_out(44) <= clk AND track_plano(44);
track_plano_out(45) <= clk AND track_plano(45);
track_plano_out(46) <= clk AND track_plano(46);
track_plano_out(47) <= clk AND track_plano(47);

---------- Contador 1 (memorias RAM) ----------

contador1:
process(clk, reset, mode, seq_sel, dir)

begin

if (reset = '0') then
    aux1_addr1 <= 0;                                        -- Reset para
reinicio a cero
    senal1 <= '0';
else
    if (mode = "11") then
        if (seq_sel = '1') then
            aux1_addr1 <= 0;
            senal1 <= '0';
        elsif (clk'event and clk = '1') then
            if (dir = '0') then
                if (aux1_addr1 = words1 - 1) then
                    aux1_addr1 <= 1;                                --
Reinicia a 1 después de completada una cuenta
                    --senal1 <= '1';
```

```
                elsif (aux1_addr1 = words1 - 2) then              --
                       aux1_addr1 <= aux1_addr1 + 1;              --
                       senal1 <= '1';                             --
                else
                    aux1_addr1 <= aux1_addr1 + 1;                           --
Cálculo de avance ascendente
                end if;
            else
                if (aux1_addr1 = 0) then
                    aux1_addr1 <= words1 - 1;                               --
Reinicia a 16 después de completada una cuenta
                   -- senal1 <= '1';
                elsif (aux1_addr1 = 1) then
                       aux1_addr1 <= words1 - 1;
                    --senal1 <= '1';
                elsif (aux1_addr1 = 2) then                       --
                       aux1_addr1 <= aux1_addr1 - 1;              --
                       senal1 <= '1';                             --
                else
                    aux1_addr1 <= aux1_addr1 - 1;                           --
Cálculo de avance descendente
                end if;
            end if;
        end if;
    else
        aux1_addr1 <= 0;                                                    -- Reset
para reinicio a cero
        senal1 <= '0';
    end if;
end if;

end process contador1;

---------- Salida de memorias ram (con multiplexor para los diferentes modos de
funcionamiento) ----------
mem_data_out_1a <= memory_1a(aux1_addr1);
mem_data_out_1b <= memory_1b(aux1_addr1);
mem_data_out_1c <= memory_1c(aux1_addr1);
mem_data_out_1d <= memory_1d(aux1_addr1);
mem_data_out_1e <= memory_1e(aux1_addr1);
mem_data_out_1f <= memory_1f(aux1_addr1);
mem_data_out_2a <= memory_2a(aux1_addr1);
mem_data_out_2b <= memory_2b(aux1_addr1);
mem_data_out_2c <= memory_2c(aux1_addr1);
mem_data_out_2d <= memory_2d(aux1_addr1);
mem_data_out_2e <= memory_2e(aux1_addr1);
mem_data_out_2f <= memory_2f(aux1_addr1);
--clk_aux <= q_clk2;
--paro <= parar;

---------- Multiplexores de salida ----------

salida_1 <= cero when reset = '0' else
          cero when reset = '1' and mode = "00" else
          cero when reset = '1' and mode = "01" else
          cero when reset = '1' and mode = "10" else
          mem_data_out_1a & mem_data_out_1b & mem_data_out_1c & mem_data_out_1d &
mem_data_out_1e & mem_data_out_1f when reset = '1' and mode = "11" and seq_sel =
'0' and parar = '0' else   -- se concatenan las memorias (horizontales)
          track_plano_out when reset = '1' and mode = "11" and seq_sel = '1' and
senal2 = '0' else
          cero;
```

```
salida_2 <= cero when reset = '0' else
          cero when reset = '1' and mode = "00" else
          cero when reset = '1' and mode = "01" else
          cero when reset = '1' and mode = "10" else
          mem_data_out_2a & mem_data_out_2b & mem_data_out_2c & mem_data_out_2d &
mem_data_out_2e & mem_data_out_2f when reset = '1' and mode = "11" and seq_sel =
'0' and parar = '0' else   -- se concatenan la memorias (verticales)
          plano when reset = '1' and mode ="11" and seq_sel = '1' and senal2 =
'0' else
          cero;

end descripcion;
```

Apéndice 2B: Código VHDL de la arquitectura del prototipo Track-Flat

```
-- flat_track8_b_100.vhd

---------- SISTEMA PARA ENCENDIDO DE LÍNEA DE 8 LEDS ----------

-- Reloj de 100MHz => pasos de 10ns

----------------------------------------------------------------------------------
-
-- Company:
-- Engineer:
--
-- Create Date:    18:16:12 05/23/2011
-- Design Name:
-- Module Name:    flat_track8_b_100 - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
-
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.ALL;


-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

ENTITY flat_track8_b_100 IS
 GENERIC (bits: INTEGER := 8;                                    -- # de bits por
palabra
         -- valor1 : std_logic_vector := "00001111";
         -- valor2 : std_logic_vector := "11110000";
          num_leds: INTEGER := 8;                                -- # de LEDs a
manejar
             salida_un_led: STD_LOGIC_VECTOR := "00001000";
             a: INTEGER := 4;
             b: INTEGER := 5;
             c: INTEGER := 6;
             d: INTEGER := 7;
             e: INTEGER := 8;
             f: INTEGER := 9;
```

```
            g: INTEGER := 10;
            h: INTEGER := 11;
            i: INTEGER := 12;
            j: INTEGER := 13;
         -- words1: INTEGER := 800;                              -- # de palabras
de la memoria rom  1 (del reloj de 40 MHz(25ns): 2.5us/25ns = 100, que por 8 LEDs
da 800)
          cero: STD_LOGIC_VECTOR := "00000000");               -- # cero a la
salida (deshabilita). 8 ceros

    PORT (clk : IN STD_LOGIC;
          reset : IN STD_LOGIC;
          modo : IN STD_LOGIC;
          dir : IN STD_LOGIC;
          selector: IN STD_LOGIC_VECTOR (3 DOWNTO 0);          -- 4 bits para
seleccionar el ancho de pulso deseado
          clk_aux: OUT STD_LOGIC;
             clk_salida: OUT STD_LOGIC;
          salida: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));             -- Salida del
multiplexor (8 bits de salida)

 END flat_track8_b_100;

architecture descripcion of flat_track8_b_100 is

-- TYPE estados IS (inicio, continuo, una_vez, fin_secuencia);  -- Se definen los
estados

signal salida_aux : std_logic_vector(bits -1 DOWNTO 0);      -- Señal auxiliar
para asignar pulsos de salida
--signal salida_un_led: std_logic_vector(bits - 1 DOWNTO 0);   -- Para un solo
pulsar un led a la vez

signal q_clk2: std_logic;                                    -- Reloj para
contador2
signal clk2_aux2: integer range 0 to 126;                     -- Cuenta del
divisor de frecuencia (clock2)
signal aux2 : integer range 0 to 8;                          -- Lleva la cuenta
del contador2
signal senal2 : std_logic;

signal parar : std_logic;                                    -- Para detener el
modo continuo (modo 1 vez)

-- signal presente : estados;                                   -- Señal para
manjo de estados


begin

---------- Reloj 2 ----------

clock2:
process (clk, reset)

begin

       if (reset = '0') then
          clk2_aux2 <= 0;
          q_clk2 <= '0';
       else
          if (clk'event and clk = '1') then
             clk2_aux2 <= clk2_aux2 + 1;
```

```
            if (clk2_aux2 = 125) then
               q_clk2 <= not q_clk2;                              -- Se cambia la
señal clk1 después de cierto # de ciclo de reloj (división de frecuencia en este
caso por 100)
               clk2_aux2 <= 1;                                    -- Se reestablece
el valor de la cuenta a uno (con esto se evitan medios periodos de menor duración
al inicio)
            end if;
         end if;
      end if;
end process clock2;


---------- Contador 2 ----------

contador2:
process (q_clk2, reset, modo, dir, aux2)                         -- Contador de
ciclos del reloj para obtener nueva frecuencia

begin

if (reset = '0') then
    aux2 <= 0;                                          -- Reset para reinicio a cero
    senal2 <= '0';
else
    if (q_clk2'event and q_clk2 = '1') then
        if (modo = '0') then                            -- Para reproducción de 1 vez
           if (dir = '0') then
               if (aux2 = num_leds) then
                   aux2 <= 1;                                    -- Reinicia a 1
después de completada una cuenta (8 GTUs, 1 GTU por LED)
                   senal2 <= '1';                                -- Señal auxiliar
para determinar el fin para modo una vez
               else
                   aux2 <= aux2 + 1;                             -- Cálculo de avance
ascendente
               end if;
           else
               if (aux2 = 0) then
                   aux2 <= num_leds;                             -- Reinicia a 8
después de completada una cuenta (8 GTUs, 1 GTU por LED)
               elsif (aux2 = 1) then
                   aux2 <= num_leds;
                   senal2 <= '1';                                -- Señal auxiliar
para determinar el fin para modo una vez
               else
                   aux2 <= aux2 - 1;                             -- Cálculo de avance
descendente
              end if;
           end if;
        else                                            -- Para reproducción continua
(modo = '1') se quita senal2 <= '1'
           if (dir = '0') then
               if (aux2 = num_leds) then
                   aux2 <= 1;                                    -- Reinicia a 1
después de completada una cuenta (8 GTUs, 1 GTU por LED)
                  -- senal2 <= '1';                             -- Señal auxiliar
para determinar el fin para modo una vez
               else
                   aux2 <= aux2 + 1;                             -- Cálculo de avance
ascendente
               end if;
           else
```

```
                if (aux2 = 0) then
                    aux2 <= num_leds;                          -- Reinicia a 8
después de completada una cuenta (8 GTUs, 1 GTU por LED)
                elsif (aux2 = 1) then
                    aux2 <= num_leds;
                   -- senal2 <= '1';                           -- Señal auxiliar
para determinar el fin para modo una vez
                else
                    aux2 <= aux2 - 1;                          -- Cálculo de avance
descendente
               end if;
            end if;
         end if;
     end if;
end if;

end process contador2;


---------- Generador de secuencias de pulsos con anchos variables ----------

pulsos:
process(reset, selector, dir, clk2_aux2, aux2, q_clk2)

begin

if (reset = '0') then
salida_aux <= cero;
else
   if (selector = "0000") then                          -- PULSO DE 20ns DE ANCHO
      if (dir = '0') then                                                      --
Sentido ascendente
         if (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 1 and q_clk2 = '0') then
            salida_aux(0) <= '1';                                              --
Salida de un pulso de 20ns para LED1
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 20ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
```

```
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;
         end if;
      else                                                                     --
Sentido descendente (dir = '1')
         if (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 1 and q_clk2 = '0') then
```

```
            salida_aux(0) <= '1';                                                   --
Salida de un pulso de 25ns para LED1
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 25ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 25ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 25ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 25ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 25ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
```

```
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= a and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 25ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;                                                --
Cero para otros casos (único pulso)
         end if;
      end if;
   elsif (selector = "0001") then                    -- PULSO DE 50ns DE ANCHO
      if (dir = '0') then                                                      --
Sentido ascendente
         if (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 1 and q_clk2 = '0') then
            salida_aux(0) <= '1';                                              --
Salida de un pulso de 50ns para LED1
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 50ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 50ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED4
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '1';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED5
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '1';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED6
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '1';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;
        end if;
      else                                                                   --
Sentido descendente (dir = '1')
        if (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 50ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= b and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 50ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "0010") then                    -- PULSO DE 80ns DE ANCHO
     if (dir = '0') then                                                     --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 75ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
         elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 75ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 75ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 75ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 75ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 75ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;
         end if;
      else                                                                     --
Sentido descendente (dir = '1')
         if (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 1 and q_clk2 = '0') then
            salida_aux(0) <= '1';                                              --
Salida de un pulso de 75ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= c and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 75ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "0011") then                    -- PULSO DE 100ns DE ANCHO
      if (dir = '0') then                                                    --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 100ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED4
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '1';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED5
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '1';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED6
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '1';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;
        end if;
      else                                                                   --
Sentido descendente (dir = '1')
        if (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 100ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= d and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 100ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "0100") then                    -- PULSO DE 120ns DE ANCHO
      if (dir = '0') then                                                    --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 125ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED4
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '1';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED5
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '1';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED6
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '1';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;
        end if;
      else                                                                   --
Sentido descendente (dir = '1')
        if (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 125ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= e and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 125ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "0101") then                    -- PULSO DE 150ns DE ANCHO
     if (dir = '0') then                                                     --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 150ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED4
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '1';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED5
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '1';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED6
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '1';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;
        end if;
      else                                                                   --
Sentido descendente (dir = '1')
        if (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 150ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= f and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 150ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "0110") then                    -- PULSO DE 170ns DE ANCHO
      if (dir = '0') then                                                    --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 175ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
         elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 175ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 175ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 175ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 175ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 175ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;
         end if;
      else                                                                     --
Sentido descendente (dir = '1')
         if (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 1 and q_clk2 = '0') then
            salida_aux(0) <= '1';                                              --
Salida de un pulso de 175ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= g and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 175ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "0111") then                    -- PULSO DE 200ns DE ANCHO
      if (dir = '0') then                                                    --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 200ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED4
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '1';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED5
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '1';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED6
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '1';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;
        end if;
      else                                                                   --
Sentido descendente (dir = '1')
        if (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 200ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= h and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 200ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
        end if;
      end if;
   elsif (selector = "1000") then                    -- PULSO DE 220ns DE ANCHO
      if (dir = '0') then                                                    --
Sentido ascendente
        if (clk2_aux2 > 0 and clk2_aux2 <= i and aux2 = 1 and q_clk2 = '0') then
              salida_aux(0) <= '1';                                          --
Salida de un pulso de 225ns para LED1
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED2
              salida_aux(1) <= '1';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED3
              salida_aux(1) <= '0';
              salida_aux(2) <= '1';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
```

```
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED4
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '1';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED5
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '1';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED6
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '1';
              salida_aux(6) <= '0';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED7
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '1';
              salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
              salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED8
              salida_aux(1) <= '0';
              salida_aux(2) <= '0';
              salida_aux(3) <= '0';
              salida_aux(4) <= '0';
              salida_aux(5) <= '0';
              salida_aux(6) <= '0';
              salida_aux(7) <= '1';
        else
              salida_aux <= cero;
        end if;
      else                                                                   --
Sentido descendente (dir = '1')
        if (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 1 and q_clk2 = '0')
then
```

```
            salida_aux(0) <= '1';                                                   --
Salida de un pulso de 225ns para LED1
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 225ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 225ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 225ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i  and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 225ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= i and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                                   --
Salida de un pulso de 225ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
```

```
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= i and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= i and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 225ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;                                            --
Cero para otros casos (único pulso)
         end if;
      end if;
   elsif (selector = "1001") then                    -- PULSO DE 250ns DE ANCHO
      if (dir = '0') then                                                  --
Sentido ascendente
         if (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 1 and q_clk2 = '0') then
            salida_aux(0) <= '1';                                          --
Salida de un pulso de 250ns para LED1
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 2 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 3 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 4 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 5 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 6 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;
         end if;
      else                                                                     --
Sentido descendente (dir = '1')
         if (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 1 and q_clk2 = '0') then
            salida_aux(0) <= '1';                                              --
Salida de un pulso de 250ns para LED1
```

```
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 2 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED2
            salida_aux(1) <= '1';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 3 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED3
            salida_aux(1) <= '0';
            salida_aux(2) <= '1';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 4 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED4
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '1';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 5 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED5
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '1';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
        elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 6 and q_clk2 = '0')
then         -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                          --
Salida de un pulso de 250ns para LED6
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '1';
            salida_aux(6) <= '0';
            salida_aux(7) <= '0';
```

```
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 7 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED7
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '1';
            salida_aux(7) <= '0';
         elsif (clk2_aux2 > 0 and clk2_aux2 <= j and aux2 = 8 and q_clk2 = '0')
then          -- Cero para otros casos (único pulso)
            salida_aux(0) <= '0';                                              --
Salida de un pulso de 250ns para LED8
            salida_aux(1) <= '0';
            salida_aux(2) <= '0';
            salida_aux(3) <= '0';
            salida_aux(4) <= '0';
            salida_aux(5) <= '0';
            salida_aux(6) <= '0';
            salida_aux(7) <= '1';
         else
            salida_aux <= cero;                                                --
Cero para otros casos (único pulso)
         end if;
      end if;
   end if;
end if;
end process pulsos;

---------- Multiplexor de salida ----------

salida <= cero when reset = '0' else
          salida_aux and salida_un_led when reset = '1' and senal2 = '0' else
          cero;

---------- Salida reloj auxiliar de 2.5us ----------

clk_aux <= q_clk2;

clk_salida <= clk;

end descripcion;
```

Apéndice 3: Tecnología TrenchMOS

La tecnología TrenchMOS, desarrollada por Phillips Semiconductors (división ahora llamada NXP), representa una mejora sustancial sobre otros dispositivos MOSFET. Esta nueva forma de componentes MOSFET de potencia, se compone de un tipo nuevo de estructuta en el transistror, la cual provee de una trayectoria más directa y eficiente para el flujo de corriente a lo largo del dispositivo semiconductor, así como baja resistencia de canal $R_{ds}(on)$.

Esta tecnología ofrece a los diseñadores electrónicos características como baja disipación de potencia, mayor capacidad de corriente en un chip del mismo tamaño que los demás dispositivos semejantes y existentes, ó un chip de menor tamaño pero con la misma disipación de potencia. Por esto, este tipo de tecnología ofrece gran ventaja en situaciones en que las aplicaciones requieren de dispositivos MOSFET de potencia con buenas prestaciones.

La forma de un dispositivo con tecnología TrenchMOS utiliza una estructura vertical para realizar un dispositivo FET. De esta forma, se tiene la ventaja fabricar un dispositivo con un canal más corto. En dispositivos estándar, la corriente fluye a través de regiones de área relativamente pequeña, por lo que los valores de resistencia en el canal crecen, porvocando una gran reducción en la eficiencia de todo el dispositivo. Para disminuir este inconveniente, los dispositivos en tecnología TrenchMOS se basan en una estructura vertical para el flujo de corriente en lugar de un flujo horizontal, como en los dispositivos MOS estándar. Como se muestra en la figura *a*, este tipo de dispositivos tienen la terminal de fuente (source) en la parte superior del dispositivo, mientras que la terminal de drenaje (drain) se localiza en la parte inferior.

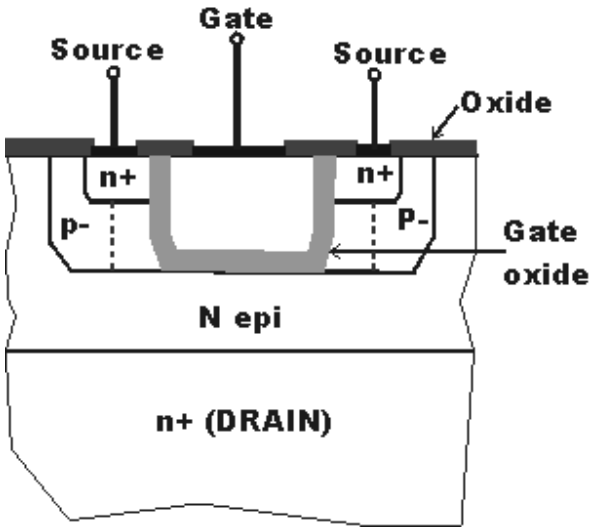


Figura a) Estructura de un transistor de efecto de campo en tecnología TrenchMOS. Imagen tomada de [27].

Aún cuando la tecnología TrenchMOS utiliza estructuras verticales, ésta se diferncia de otros tipos de tecnología MOS basadas en el mismo tipo de estructuras verticales, por ejemplo, como la estructura V-groove. Con la tecnología trenchMOS, la compuerta se forma mediante una "fisura" de grabado en la estructura del semiconductor, a la cual se le agrega posteriormente una capa aislante de óxido metálico y finalmente se rellena con polisilicio. La estructura resultante se encuentra optimizada para la operación de dispositivos MOSFET de potencia, en donde la porfundidad de la estructura es de vital importancia para la operación y la resistencia menor respecto de otros dispositivos FET, la cual, resulta en ordenes de 8, 14, 18 y 24 mΩ, con voltajes de operación de entre 5 y 10V, y tolerancia a picos de hasta 55V ó 60V. Este valor pequeño de resistencia implica que no se requiere de disipadores de calor en muchos casos, permitiendo que los dispositivos sean fabricados en empaquetados de montaje superficial.

Apéndice 4: Diseño del programa de control del instrumento Track-Sim

El programa de control representa la segunda parte de todo el sistema de control. La función principal es la de realizar toda la administración y operación del instrumento Track-Sim. En este capítulo se presenta el algoritmo desarrollado para la traducción de EAS simulados en patrones de bits, los cuales, serán descargados en el hardware que actúa, mediante la guía de luz, sobre los bloques PDM de la superficie focal de JEM-EUSO. Así mismo, se presenta el algoritmo de control que realiza la comunicación con el hardware.

4.1 TRADUCCIÓN DE SIMULACIONES DE EAS EN PATRONES DE GENERACIÓN DE PULSOS ELÉCTRICOS EN TIEMPO Y ESPACIO

La función del algoritmo de conversión de *tracks* es la de transformar los archivos de librerías que contienen las simulaciones correspondientes a EAS en patrones de bits que indican la apertura y cierre de *switches* para el encendido de LEDs. Los archivos de librerías con resultados de las simulaciones contienen, básicamente, la información del número de fotones que arriban a la superficie focal, y en este caso, a la sección que corresponde a un PDM. Con el fin de simplificar el algoritmo de conversión se organiza la información de cada archivo de simulación de la siguiente manera.

Los archivos ASCII de las librerías contendrán la información en 4 columnas organizadas como se muestra en la tabla 4.1.

x	***y***	**Número de fotones (Φ_{xyn})**	**GTU**
4	2	43	1
7	5	66	1
6	9	113	2
• • •	• • •	• • •	• • •
46	48	52	94
x_n	y_n	ϕ_{xyn}	L

Tabla 4.1. Organización de la información dentro de cada archivo ASCII con resultados de simulaciones de EAS obtenido de ESAF.

Las columnas *x* y *y* determinan la posición de un pixel dentro de un PDM, mientras que la tercer columna (Φ_{xy}) contiene directamente la cantidad de fotones en dicho pixel. La cuarta columna indica el número de GTU en que se presenta la cantidad de fotones en los pixeles conforme se desarrolla cada *track*.

A partir de esta información en cada archivo se ejecuta el algoritmo de conversión, sin embargo, antes de exponer dicho algoritmo, se hará referencia sobre la obtención de determinados parámetros resultados de las pruebas presentadas en el capítulo anterior sobre el método de control de intensidad en LEDs.

De la figura 3.20 ó 3.21*a* del capítulo anterior se establece que se tienen igual número de *switches* fila y *switches* columna en un valor de 48, respectivamente, a fin de cubrir una estructura PDM por completo. Si se controla la intensidad de cada pixel mediante el tiempo de encendido del LED respectivo a través de los *switches*, y encendiendo uno a la vez, a partir del direccionamiento matricial, se sabe que mediante el cierre de la pareja de *switches xy* para el pixel en cuestión, durante el tiempo necesario, se alcanzará la intensidad Φ requerida. Entonces, se requiere que ambos *switches* se cierren durante el mismo intervalo temporal a fin de alcanzar el valor Φ, el cual, puede ser obtenido en un solo cierre de ambos *switches* o mediante *n* cierres de éstos, lo cual, depende del valor de Φ, así como del valor seleccionado de *R* en serie con el LED. Para obtener el tiempo de cierre de un par de

switches que permita alcanzar el valor Φ, se parte de la respuesta típica para una curva como las mostradas en la figura 3.43*b* dentro del apartado de pruebas y resultados en el capítulo 3. Como se explico anteriormente, a mayores valores de resistencia, el rango dinámico que se puede alcanzar para Φ se hace más grande, y es posible obtener familias de curvas con diferentes valores de *R* en serie con el LED, las cuales, permitan alcanzar mediante uno solo o varios pulsos de encendido en el LED el valor Φ. Sin embargo, no es posible utilizar anchos de pulso demasiado "grandes" (con *R* mayores) para obtener el rango dinámico, ya que se requiere encender varios LEDs dentro de un mismo GTU, por lo que, pensando en el máximo del rango de ~300 fotones por pixel por GTU, un valor razonable que cumpla con los requisitos para una matriz característica de 3 × 3 pixeles que viaja en forma del *track* sobre la superficie focal, sería un comportamiento como el de la curva para R ~ 175 Ω que se muestra en la figura 3.43*b*, ya que si, en el peor de los caso, cada pixel requiriese del máximo de intensidad de 300 fotones, sería posible encender dos veces cada pixel para conseguirlo (2500ns/18 ~ 138 ns, con alrededor de 150 fotones para pulsos con ancho de ~ 125 ns).

Por otra parte, es importante señalar que el encendido entre LEDs no se realiza de forma instantánea, ya que existen tiempos muertos debidos a los tiempos tanto de levantamiento como de decaimiento de los *switches* involucrados, sin embargo, estos tiempos no representan dificultades para conseguir las intensidades máximas en, por lo menos, la matriz de 3 × 3 pixeles, debido a que se trata de tiempos pequeños menores a 20 ns en total.[33] .

Una vez sea elegida una curva con un valor de resistencia fijo, lo cual depende, para cada canal, de las compensaciones que deban realizarse mediante el valor de resistencia serie con el LED a fin de obtener, en la medida de lo posible, los mismos valores de intensidad para los mismos anchos de pulso utilizados, se determina la curva de fotones detectados por el dispositivo MAPMT en función del ancho de pulso y se obtiene el rango lineal como se muestra en la figura 4.1.

[33] Los *switches* utilizados en el encendido de los LEDs encienden y apagan en tiempos menores a 20 ns.

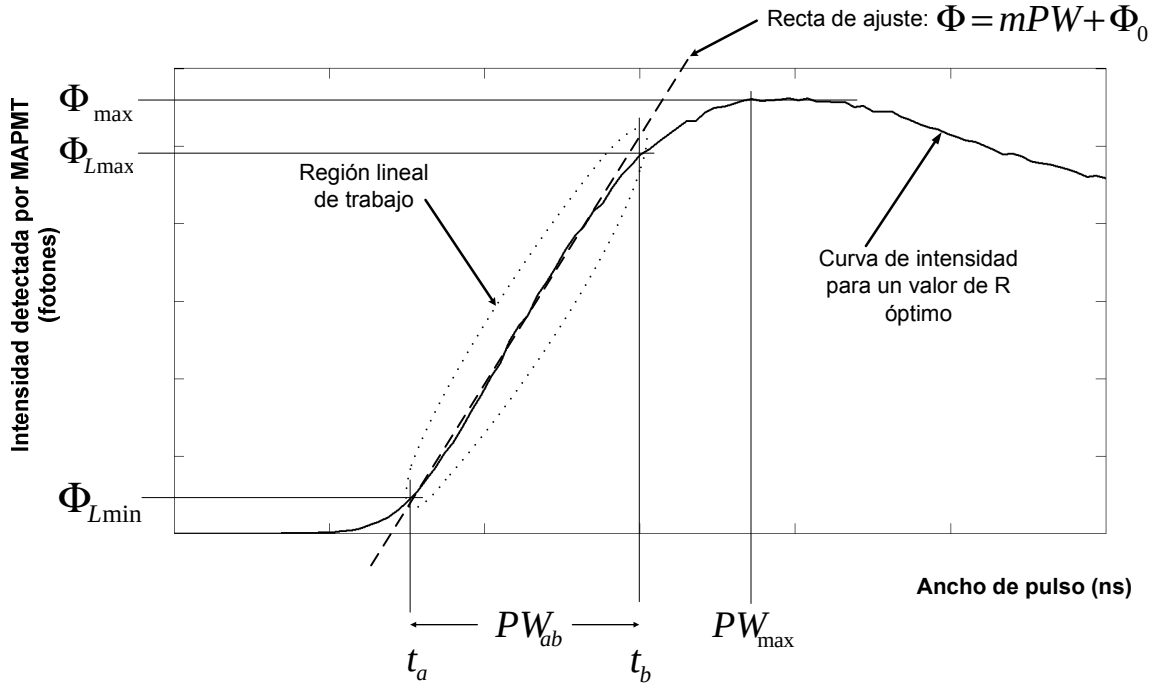


Figura 4.1. Curva de fotones detectados en el dispositivo MAPMT en función del ancho de pulso y recta de ajuste en la región lineal para un valor de resistencia serie con el LED.

De esta figura se obtiene la región lineal, la cual se caracteriza por una curva de la forma

$$\Phi = mPW + \Phi_0 \qquad (1)$$

Esta ecuación representa la recta de ajuste en la región lineal, siendo Φ la intensidad en número de fotones, m la pendiente, PW el ancho de pulso y Φ_0 la ordenada al origen. Una vez seleccionado el valor de R que mejor se ajuste al modelo de la guía óptica final, así como sea factible el cumplir con el rango dinámico de intensidad, se determina la región lineal en la curva característica como se muestra en la figura 4.1 para llegar a una ecuación de la forma de (1), la cual será válida únicamente para el intervalo definido como lineal de

$$PW_{ab} = t_b - t_a[ns] \qquad (2)$$

ó

$$PW_a = t_a[ns]$$
$$PW_b = t_b[ns] \qquad (3)$$

Entonces, para el rango lineal, la ecuación (1) puede reescribirse únicamente cómo

$$\Phi = mPW \qquad (4)$$

Esta recta es la que proporciona la relación de control en la emisión del conjunto LED-guía óptica y que debe ser reproducible, dentro de sus intervalos de confianza, en todos los canales en la versión definitiva del Track-Sim.

La resolución en (4) depende completamente de los incrementos ΔPW, los cuales, se encuentran en función del reloj utilizado en el FPGA, teniéndose desde luego mejor resolución con frecuencias de reloj más altas.[34]

Algoritmo de conversión de *tracks* de EAS simulados en patrones de bits

El algoritmo de conversión de *tracks* simulados sobre la superficie focal, tiene como finalidad la producción de dos matrices binarias a partir de archivos ASCII, los cuales, contienen datos acomodados en la forma mostrada por la tabla 4.1. El tamaño de cada matriz será de $x \times M$ y de $y \times N$, con los valores de M y N determinados por la intensidad (en número de fotones) y extensión en GTUs de cada *track*, mientras que los valores de x y de y se encuentran determinados por el tamaño de la matriz que cubre a un PDM completo y que es de 48 × 48 (x = 48, y = 48). Así mismo, M y N representan el número de localidades de memoria RAM en el hardware (filas y columnas), cada una con 48 bits. Como se mencionó en el capítulo 3, un '1' indica el encendido de LEDs, mientras que un '0' indica apagado.

La figura 4.2 muestra el algoritmo de conversión de *tracks* en dos memorias de 48 × M y 48 × N.

[34] La resolución en la emisión será la que permita el sistema FPGA en cuanto a la frecuencia de reloj que puede manejar, siendo el límite, para el caso de la tarjeta ProASIC 3E, de 350MHz (~2.8ns), por lo que, para los casos en que se requiera de una emisión cuyo valor en fotones no pueda ser alcanzado con la resolución del sistema en cuestión, se realizará una aproximación por redondeo para obtener el valor más cercano posible, dando siempre el margen de error debido a dicho redondeo.

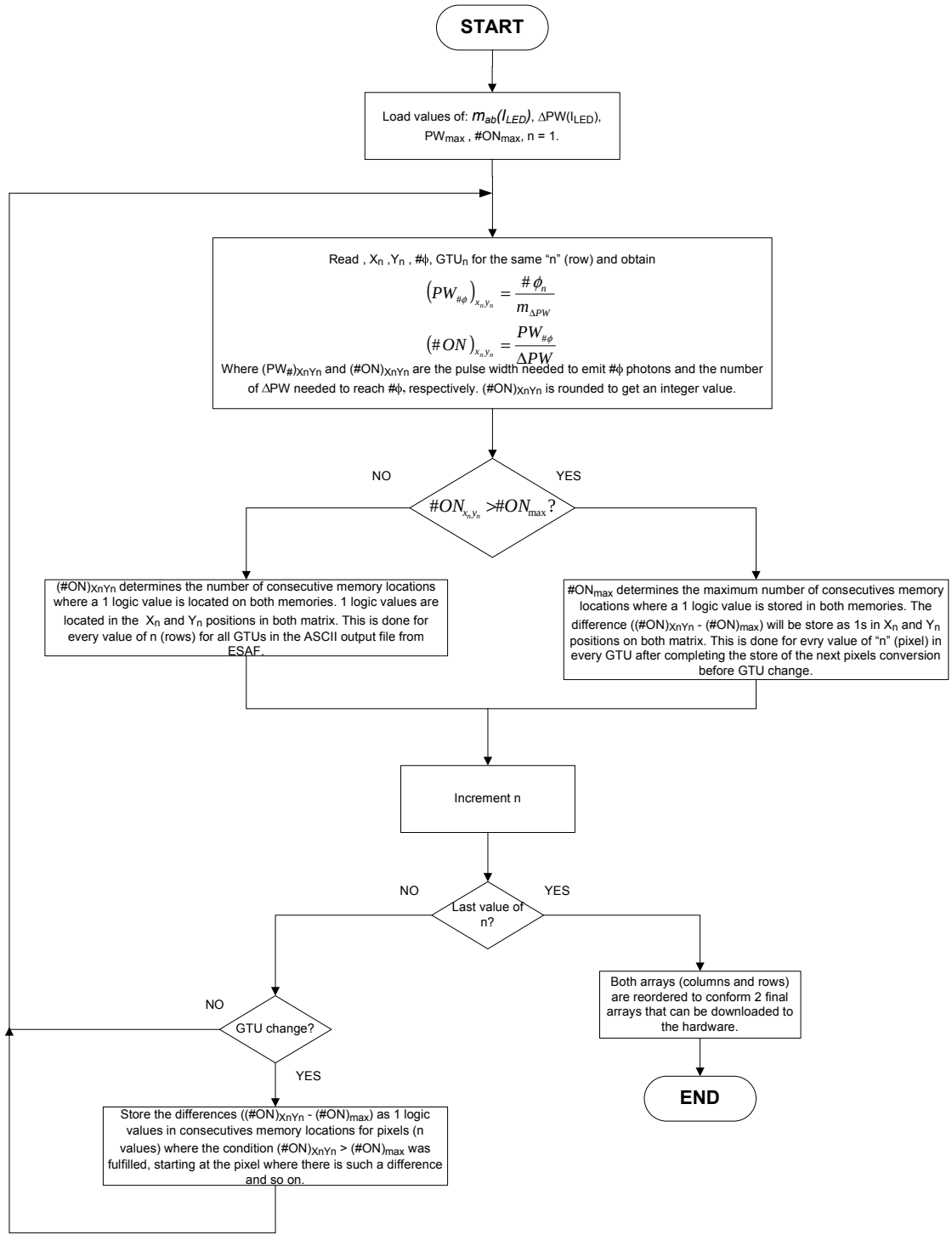


Figura 4.2. Algoritmo de conversión de "tracks" en memorias con patrones de bits de 48 × M y 48 × N (columnas y filas respectivamente).

Este algoritmo opera de la siguiente manera. Una vez se establecen los valores de las relaciones de (2), (3) y (4), se obtiene a partir de (2) y (3), la región de operación lineal, de

donde los parámetros m_{ab}, PW_{max} y $\#ON_{max}$ quedan determinados. Esto se realizará para cada canal de la guía óptica final al momento de la calibración, con ayuda del dispositivo MAPMT. El valor de ΔPW se define a partir del reloj en el hardware utilizado para la producción de pulsos, siendo, en el caso de la tarjeta destino ProASIC 3E, el valor de 350MHz la mejor resolución (~2.9 ns), aunque pasos ó anchos del orden de 5 ns (reloj de 250MHz) muestran ser factibles, aunado a la ventaja de que se trata de un múltiplo directo del valor de un GTU para procesos internos en el dispositivo FPGA.

Una vez cargados los valores de los parámetros mencionados, se lee el archivo producido por ESAF desde la primer fila (n = 1). A continuación se determinan los valores de ancho de pulso requerido ($PW_{\Phi xyn}$) para lograr la intensidad y, a partir de ahí, el número de encendidos ($\#ON_{XYn}$) requeridos para lograr la intensidad deseada como

$$\left(PW_{\Phi_{xyn}}\right)_{x_n y_n} = \frac{\Phi_{xy_n}}{m_{\Delta PW}}$$

$$\left(\#ON\right)_{x_n y_n} = \frac{PW_{\Phi_{xyn}}}{\Delta PW}$$

Con estos parámetros se determina el ancho de pulso total requerido para obtener hasta la máxima intensidad por pixel de ~300 fotones por GTU. Esto por supuesto depende del valor de la resistencia en serie con cada LED, pero con la intensión de facilitar la realización del algoritmo de conversión sin dejar de lado el cumplimiento de los requisito de diseño, se planteó utilizar un valor de resistencia tal que, de no cumplir en una sola exposición de luz la intensidad deseada, se repita la emisión en ese pixel posterior a la emisión en los otros pixeles que requieran encenderse dentro del mismo GTU, la menor cantidad de veces posibles (2 de forma idónea). Por ejemplo, para el valor de ~175 Ω, se tiene, como se muestra en la figura 3.43*b* del capítulo 3, que dicho valor de resistencia en su región lineal, presenta un rango dinámico que va de 0 a ~150 fotones, por lo que se buscaría, para alcanzar el máximo de ~300 fotones, encender, con anchos de ~125ns máximo durante dos ocasiones diferentes y distanciadas temporalmente lo suficiente, al LED en cuestión. Si el ancho de pulso requerido no sobrepasa el ancho máximo de la

región lineal de la curva de respuesta para un canal (PW_{max}), el número de encendidos obtenido será directamente el número de incrementos en ancho de pulso (ΔPW) en que se mantendrá en alto la salida del hardware para producir el ancho de pulso necesario. Por el contrario, si el ancho de pulso requerido sobrepasa el ancho máximo de la región lineal (PW_{max}), se almacenará la cantidad de 1s correspondiente a $\#ON_{max}$ (número máximo de encendidos en la región lineal) en las localidades de memoria de ambas memorias RAM ($x \times M$, $y \times N$), para posteriormente, cuando se terminen de encender por primera vez en el mismo GTU todos los LEDs que requieran de una intensidad específica, se regresa a aquellos que requieran de mayor tiempo de encendido para lograr la intensidad deseada. Los importante aquí es que la intensidad requerida por GTU, debe cumplirse dentro del tiempo que dura un GTU, sin importar que todos los fotones lleguen de manera consecutiva hasta alcanzar la intensidad requerida, ya que pueden estar dispersos a lo largo de todo el GTU, pero siempre y cuando se detecten dentro del mismo GTU.[35]

Todo este proceso se realiza para cada fila de datos del archivo ASCII con resultados de las simulaciones. Al termino del procesamiento en cada fila en dicho archivo, se pasa a la siguiente fila (se incrementa "n"). Si no es la última fila se evalúa si hay cambio de GTU, resultando para un cambio de GTU en que se almacenan los valores pendientes de la diferencia $\#ON_{XYn} - \#ON_{max}$ en los pixeles en que fueron evaluados, como 1s en la cantidad correspondiente a dicha diferencia de localidades de memoria subsecuentes ($x \times M$, $y \times N$), comenzando por el primer pixel en donde se cumplió que $\#ON_{XYn} > \#ON_{max}$ y así subsecuentemente hasta el último pixel en donde el número de encendidos supera al máximo de la región lineal. Posteriormente se regresa a la evaluación de los parámetros $\#ON_{XYn}$ y $\#ON_{max}$ para la nueva fila de datos del archivo de resultados de simulaciones. De igual forma, si no hay cambio de GTU, se calculan los nuevos valores de $\#ON_{XYn}$ y $\#ON_{max}$ para la siguiente fila de datos en el archivo para comenzar con el procesamiento nuevamente. En el caso de tratarse de la última fila del archivo, se concluye con el algoritmo de conversión.

[35] Este es uno de los requisitos de diseño del instrumento Track-Sim, el cual, abre la posibilidad de utilizar el control por intensidad PWV trabajando en la región de comportamiento lineal del conjunto LED - MAPMT.

Es importante señalar que tanto las posiciones "*x*" y "*y*" en cada memoria las determinan directamente los valores en dichas columnas que se leen dentro de archivo de resultados de simulaciones de EAS, mientras que la cantidad de localidades de memoria subsecuentes con valores de '1' lógico en ambas memorias (encendido del LED), la determina la resolución temporal en el hardware (ΔPW), el valor de resistencia (R) en serie con el LED a utilizar y la intensidad Φ en el pixel respectivo. Este último valor (Φ) depende únicamente de los resultados de simulación de EAS, mientras que la resolución temporal y la resistencia limitadora de corriente con el LED, dependen del hardware utilizado así como del análisis de respuesta del LED en el dispositivo MAPMT, respectivamente.

4.2 GENERACIÓN DE BACKGROUND DE INTENSIDAD VARIABLE

El *background* o fotones de fondo presente en las observaciones realizadas por JEM-EUSO es una pieza importante al momento de realizar la detección de EAS. Dado que las señales de EAS son de baja intensidad, el estudio preciso de los niveles de *background* esperados durante la vida de la misión se vuelve fundamental, y más aún, el poder realizar reproducciones durante la calibración en tierra en los componentes de la superficie focal de los niveles de *tracks* en conjunto con el *background*, le da una mayor importancia al instrumento Track-Sim.

En lo que respecta a la generación de *background*, éste se presenta en un *track* como un incremento en el número de fotones respecto de un perfil esperado, es decir, análogo a lo que sucede a una señal, por ejemplo de voltaje, a la cual se le agrega un *offset* en la dirección positiva. Sin embargo, el perfil de fotones de *backgroun*d no es un valor constante durante el tiempo en que se realiza una observación de EAS, ya que dicho *backgroun*d, además de variar con el tiempo (GTUs), lo hace también dependiendo de las condiciones de observación esperadas por JEM-EUSO. Por ejemplo, se espera determinado número de fotones de *background* cuando se realicen observaciones en la atmósfera terrestre mientras se pasa por encima de algún océano, un desierto, una zona de selva tropical, áreas urbanas, noches de Luna llena, cobertura de nubes e inclusive niveles de *backgroun*d debidos a otros fenómenos que se presentan en la misma atmósfera terrestre,

como lo son los micrometeoritos que puedan emitir en el ultravioleta, eventos luminosos transitorios (TLE), descargas de plasma y relámpagos, principalmente, los cuales, forman parte también de los estudios que se planea realice JEM-EUSO [11]. Estos valores de *background* se suman a las distribuciones temporales de *tracks* observados por la superficie focal del detector como se aprecia en la figura 4.3, lo que produce un "enmascaramiento" de los *tracks* de luz debidos a EAS, principalmente para aquellos *tracks* de intensidades más bajas.

Ahora, para poder simular las señales de *backgound* junto con los perfiles de *tracks*, es preciso producir el incremento en intensidad de la señal resultante, esto es, reproducir la suma de fotones de *background* y fotones debidos a *tracks*. Por otra parte, el código ESAF cuenta con un módulo que permite simular el *background* esperado durante diferentes condiciones de observación, por lo que se vuelve posible incluir dentro del archivo de resultados de simulaciones, los niveles de *background* junto con los niveles de fotones de *tracks* debidos a EAS. De esta forma, el mismo archivo que contiene la información de un *track* específico, incluiría además la cantidad de fotones debidos a *background*, por lo que, la conversión de la información de *tracks* a secuencias binarias, utilizaría el mismo algoritmo presentado en la sección 4.1.

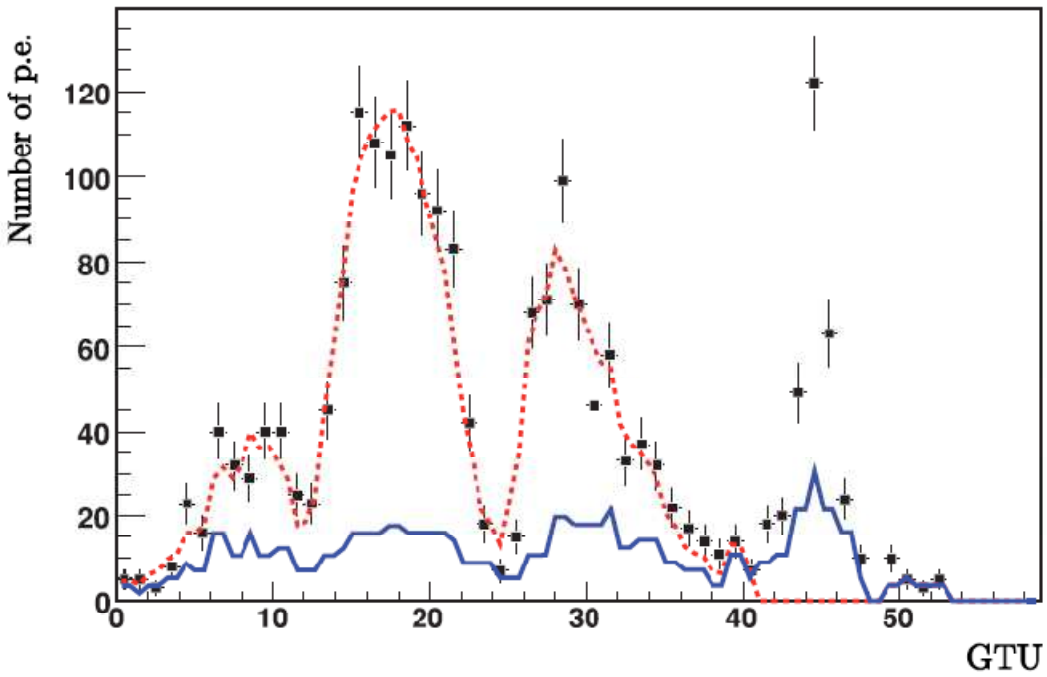


Figura 4.3. Distribución temporal del número de foto-electrones detectados por la superficie focal. Los puntos con barras de error son el número de foto-electrones observados, mientras que la curva en azul representa el background estimado y la curva en rojo representa la suma de la luz de fluorescencia (tracks) y de background. El pico al final representa la marca Cherenkov. Imagen tomada de [12].

Otra opción es la de obtener un archivo con únicamente la cantidad de fotones debidos a *background* con la misma estructura mostrada en la tabla 4.1 del archivo ASCII que contiene los resultados de simulaciones de EAS, con lo cual, sería necesario realizar un algoritmo que, a partir de éstos dos archivos, obtenga un tercer archivo con la suma de fotones debidos a perfiles y *background*. Ésta última opción depende de la factibilidad de obtener de ESAF un único archivo de resultados de *tracks* con la contribución de fotones de *background*, lo cual se encuentra en discusión y revisión con el grupo de simulación.

4.3 SOFTWARE DE CONTROL: INTERFAZ DE USUARIO

En este último apartado se presenta propiamente lo que es el diseño de los procesos del programa de control, los cuales involucran directamente la interacción mediante una interfaz de usuario. Las secciones anteriores representan parte del programa de control, sin embargo, en lo que a acciones de control propiamente se refiere, éstas se ejecutan mediante los diferentes modos de funcionamiento y comunicación que el programa realiza sobre el

hardware. Estas etapas consisten básicamente en el gerenciamiento de las ya mencionadas y descritas dentro de la sección 3.3 en el capítulo anterior. Sin embargo, aquí se tratan desde el punto de vista de quien ejecuta y decide las acciones: el conjunto usuario-máquina. Se expone a continuación la forma en que se constituyen los algoritmos de control y comunicación con el hardware, así como la forma de interactuar del usuario dentro del programa de control, lo que se manifestará al final, como una interfaz sencilla y fácil de manejar por parte del usuario.

El algoritmo principal del sistema de control es el que se presentó en la figura 3.44 *a)* en su versión simplificada y para efectos explicativos dentro de los procesos que se ejecutan en hardware. Para su mejor exposición y comprensión, se mostrará primeramente de forma reducida en sus principales etapas, para posteriormente exponer las mismas a mayor detalle. La figura 4.4 muestra el algoritmo de control general, en donde se resaltan las principales etapas: conversión de *tracks* a secuencias de bits, transmisión de datos a hardware (escritura en memorias RAM), recepción de datos de memorias (lectura de memorias RAM), comparación de datos en memorias (comparar datos leídos con datos del resultado de conversión) y reproducción de secuencias (lectura del contenido de memorias RAM para generar *track*).

Este algoritmo general opera de la siguiente manera. Parte del hecho de que se abre el programa ó se enciende el hardware. De esta manera no hay conversión de datos dentro del programa y/o tampoco se tiene *track* almacenado en el hardware listo para reproducirse. Otra opción es que ya se tenga previamente un *track* en memorias RAM, tras haber sido descargado previamente por el usuario. Si no se tienen datos en memorias RAM y se desea cargar la información de un *track* en el hardware para posteriormente reproducirse, se siguen los siguientes pasos:

1. Se pregunta al usuario por seleccionar un *track* dentro de las librerías disponibles así como iniciar el algoritmo de conversión presentado en la sección 4.1.

2. Después de que el usuario selecciona el *track* a reproducir, y una vez que se ha terminado la conversión de los datos del archivo simulado en ESAF a secuencias de bits que serán descargados en el hardware, se pregunta al usuario si se procede a la descarga de la información del *track* convertido hacia las memorias RAM del hardware de control.

3. Se realiza la transmisión de datos hacia el hardware de control, la cual constituye otro algoritmo de comunicación que se presentará más adelante. Al término de dicha descarga en el hardware se realiza una comprobación de los datos enviados a las memorias RAM, mediante una lectura de las mismas seguido de una comparación con los datos resultado del algoritmo de conversión. Si la descarga se realizó con éxito (todos los datos enviados a memorias coinciden con los datos del resultado del algoritmo de conversión), se avisa al usuario, de lo contrario, se descargan nuevamente los datos para evitar errores.

4. Cuando la revisión de los datos descargados sea correcta, se pregunta al usuario si desea realizar la reproducción de las secuencias en memorias (reproducción del *track* sobre PDM). Dependiendo de los parámetros de reproducción que el usuario establezca, al término de la reproducción del *track* se le informa al usuario (FIN DE REPRODUCCIÓN) o se para hasta que el usuario lo decida.

5. Posteriormente el programa actualiza los parámetros iniciales y regresa y espera hasta que el usuario determine la siguiente acción a realizar.

La otra opción que se tiene, después de una conversión y ejecución de *track*, es que ya se tengan datos cargados en las memorias RAM así como datos de conversión de algún archivo en librerías, ante lo cual, el sistema le da la opción al usuario de seleccionar un nuevo *track* de librerías y realizar todo el proceso de conversión y ejecución, o bien, cargar nuevamente los datos de la última conversión en las memorias RAM y/o reproducir el *track* que se encuentra almacenado en el hardware.

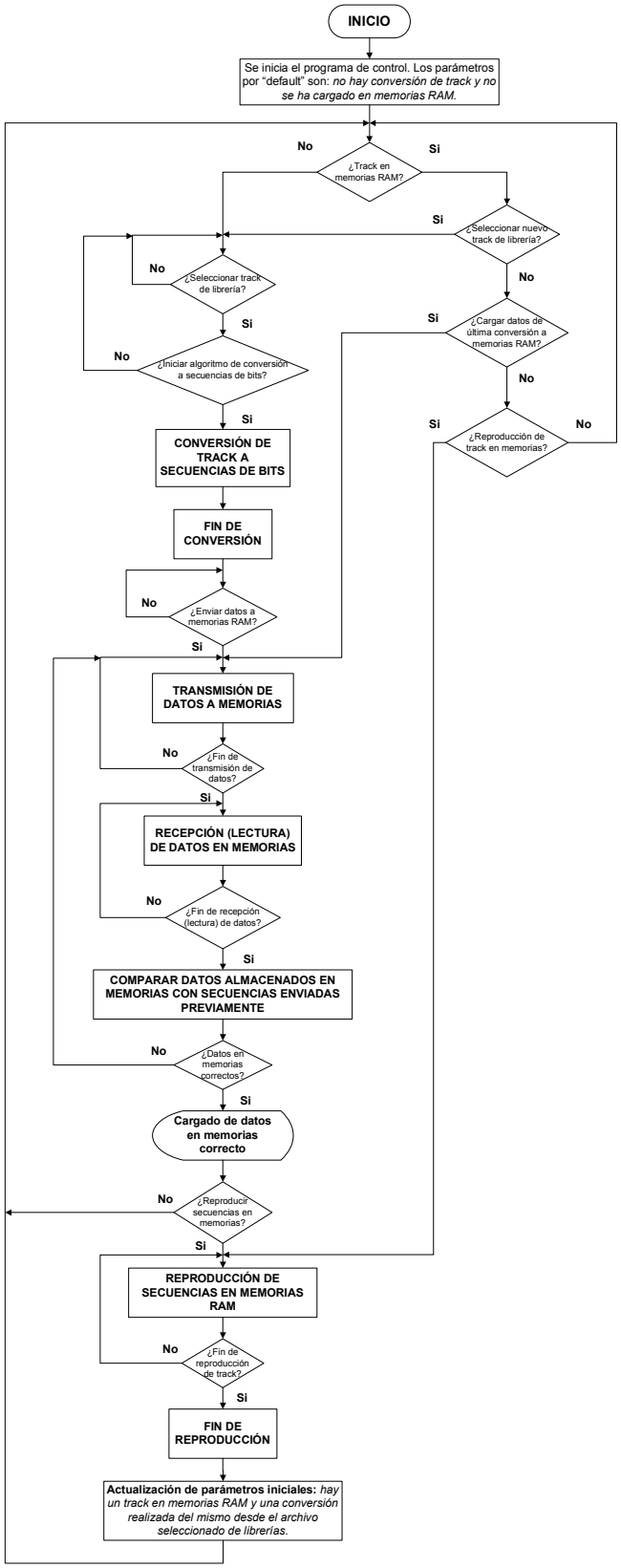


Figura 4.4. Algoritmo general de control y operación del programa de control del instrumento Track-Sim.

Ahora, se presenta el desarrollo de los algoritmos utilizados para la comunicación en las modalidades de *transmisión de datos a memorias*, *recepción (lectura) de datos en memorias* y *comparación de datos en memorias con secuencias enviadas previamente*. En lo que respecta a la reproducción de secuencias en memorias RAM, no se trata propiamente de un algoritmo, sino más bien de la ejecución en hardware mediante parámetros especificados por el usuario, del *track* en memorias RAM.

Algoritmo de transmisión de datos a memorias

El algoritmo que se encarga del envío de información de *tracks* hacia el hardware se muestra en la figura 4.5. Este algoritmo se sincroniza con la máquina de estados dentro del hardware mostrada en 3.46*b* del capítulo anterior. La forma de operar desde el punto de vista del programa de control es la siguiente. Posteriormente a la conversión de los datos de simulación de un *track* en secuencias de bits, es posible enviar el par de arreglos obtenidos a la memoria RAM correspondiente. Lo que realiza este algoritmo de escritura en memorias es leer cada una de las filas de datos de los arreglos obtenidos (arreglo o matriz de datos de memoria RAM que controla *switches* columna y arreglo o matriz de datos de memoria RAM que controla *switches* fila), y enviarla a la memoria respectiva, es decir, copia los patrones obtenidos como resultado de la conversión dentro de las memorias RAM para que, en conjunto, reproduzcan el *track* seleccionado de librerías. Como se aprecia en la figura 4.5, se tienen bits de control que le indican al hardware la acción a realizar y cuando ejecutarla. Así mismo se tiene una realimentación mediante el bit *fpga_ready*, el cual, informa cuando un proceso se ha completado y se está en posibilidad de seguir con otro.

La figura 4.5 muestra dentro de cada proceso, una breve descripción a fin de resultar más fácil de entender. Todos los procesos mostrados en 4.5 serán ejecutados por el programa de control, el cual llevará cuenta de cada acción realizada, siendo el usuario quien decida sobre los procedimiento principales de comunicación y control, pero a su vez, no intervendrá en acciones como la determinación de la finalización de envío de datos en una y/o en ambas memorias RAM, lo cual será realizado de forma autónoma por el programa.

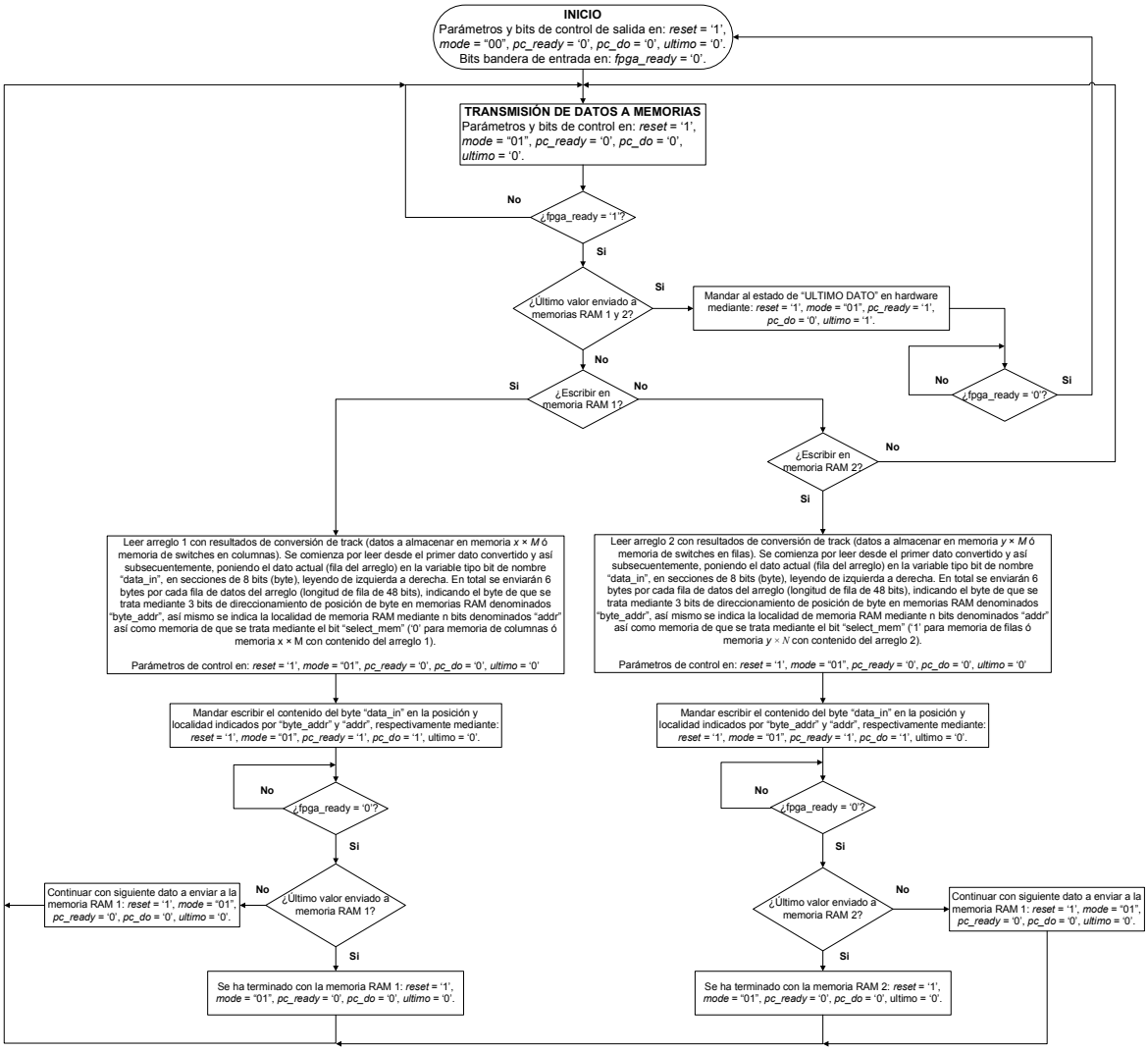


Figura 4.5. Algoritmo de comunicación para el modo de transmisión de datos a memorias RAM.

Algoritmo de recepción de datos desde memorias (lectura de memorias RAM)

La figura 4.6 muestra el algoritmo que se ejecuta para realizar la comunicación y lectura de datos desde las memorias RAM. Este algoritmo es muy similar al utilizado para escribir datos en memorias, siendo la principal diferencia el colocar al hardware en modo de lectura, así como el pasar los datos de memorias a dos arreglos para posteriormente tener la posibilidad de compararlos con los arreglos resultantes del algoritmo de conversión. El resultado de la similitud de este algoritmo con el de la figura 4.5 es consecuencia del gran parecido de ambos procesos (escritura y lectura de memorias RAM), en prácticamente la misma arquitectura de hardware, lo cual, también se refleja en la máquina de estados

presentada en 3.46*c* dentro de la sección 3.3 del capítulo anterior. De igual forma que en el caso del algoritmo para la transmisión de datos a memorias RAM, se incluye en los procesos de la figura 4.6 una breve descripción para un mejor entendimiento.

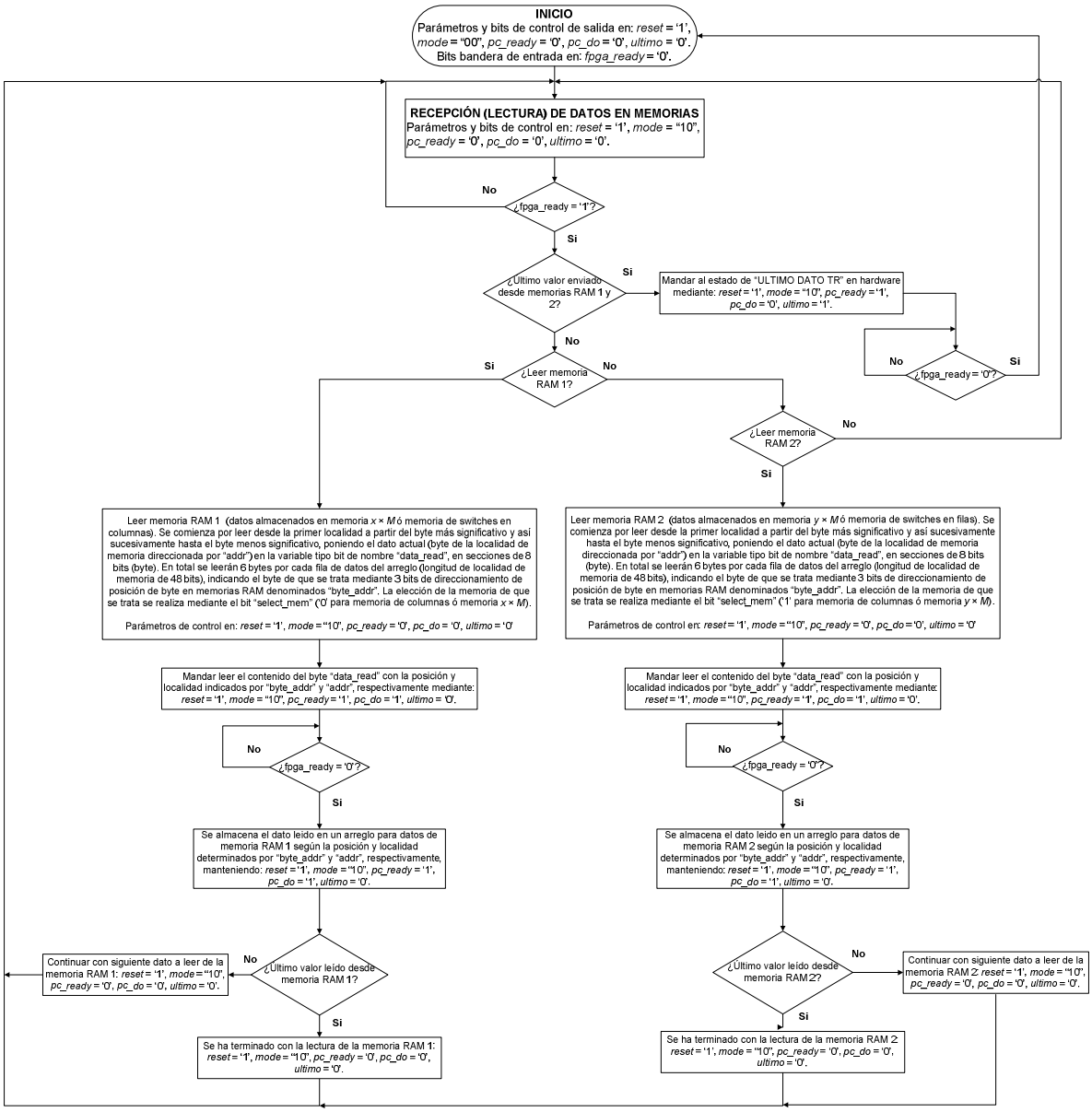


Figura 4.6. Algoritmo de recepción (lectura) de datos en memorias RAM.

Algoritmo de comparación de datos en memorias con secuencias enviadas previamente

Cada vez que se realiza una acción de envío de datos a memorias RAM (escritura en memorias), al terminar con el cargado de datos en hardware, se realiza una acción de verificación de los datos enviados a memorias y los datos obtenidos después de la conversión de un *track* en secuencias de bits. Básicamente se realiza la comparación de los patrones de 1s y 0s, los cuales, deben de coincidir entre cada una de las posiciones enviadas a las memorias y los arreglos resultantes del algoritmo de conversión. En resumen, los dos arreglos que se obtienen después de la conversión de un *track* a secuencias de bits, deben de ser iguales a los datos organizados de la misma manera dentro de ambas memorias RAM. De esta forma, el algoritmo radica en únicamente realizar una comparación de cada uno de los arreglos con su respectivo arreglo leído de memorias. Esto se muestra en la figura 4.7.

Reproducción de secuencias en memorias RAM

La sección que se encarga de realizar la reproducción de las secuencias de bits en memorias (reproducción del *track* a simular) es quizá la parte más sencilla dentro del programa de control. Básicamente el usuario determina los parámetros bajo los cuales se simulará el *track,* para posteriormente comenzar con la reproducción, la cual, se detiene hasta que el usuario así lo desea (reproducción en modo continuo), o cuando se termina de reproducir el *track* (reproducción en modo de una sola vez). Los parámetros que el usuario podrá ajustar son los siguientes.

- *repeat*: con el valor de este bit se determina si la reproducción del *track* en memorias se realiza de forma continua (*repeat* = ‘1’) o en una sola vez (*repeat* = ‘0’).
- *dir*: con este bit el usuario tiene la posibilidad de reproducir el *track* original en forma inversa, es decir, el perfil se crea partiendo de los niveles de intensidad en número de fotones desde el último GTU y hacia el GTU inicial ó GTU cero. Con

dir = ‘0’ se realiza la reproducción en sentido “normal” u original conforme se espera se presente el *track*, mientras que con *dir* = ‘1’, la reproducción se realiza de forma invertida.

- *seq_sel*: el estado de este bit decide por ejecutar el *track* almacenado en memorias mediante *seq_sel* = ‘0’, o ejecutar un *track* plano en 3 filas centrales del PDM con *seq_sel* = ‘1’.

De acuerdo a la configuración de estos parámetros se realiza la reproducción deseable. Posteriormente a la configuración establecida por el usuario, y una vez que el usuario determina comenzar con la reproducción, el programa envía *reset* = ‘1’ y *mode* = “11” para poner al hardware en modo de reproducción de secuencias y realizar la reproducción del *track*, durante la cual, el usuario no puede cambiar los parámetros de configuración sino hasta que se para la reproducción o es detenida por el mismo usuario. La figura 4.8 muestra los procesos realizados dentro de la etapa de reproducción de secuencias en memorias RAM.

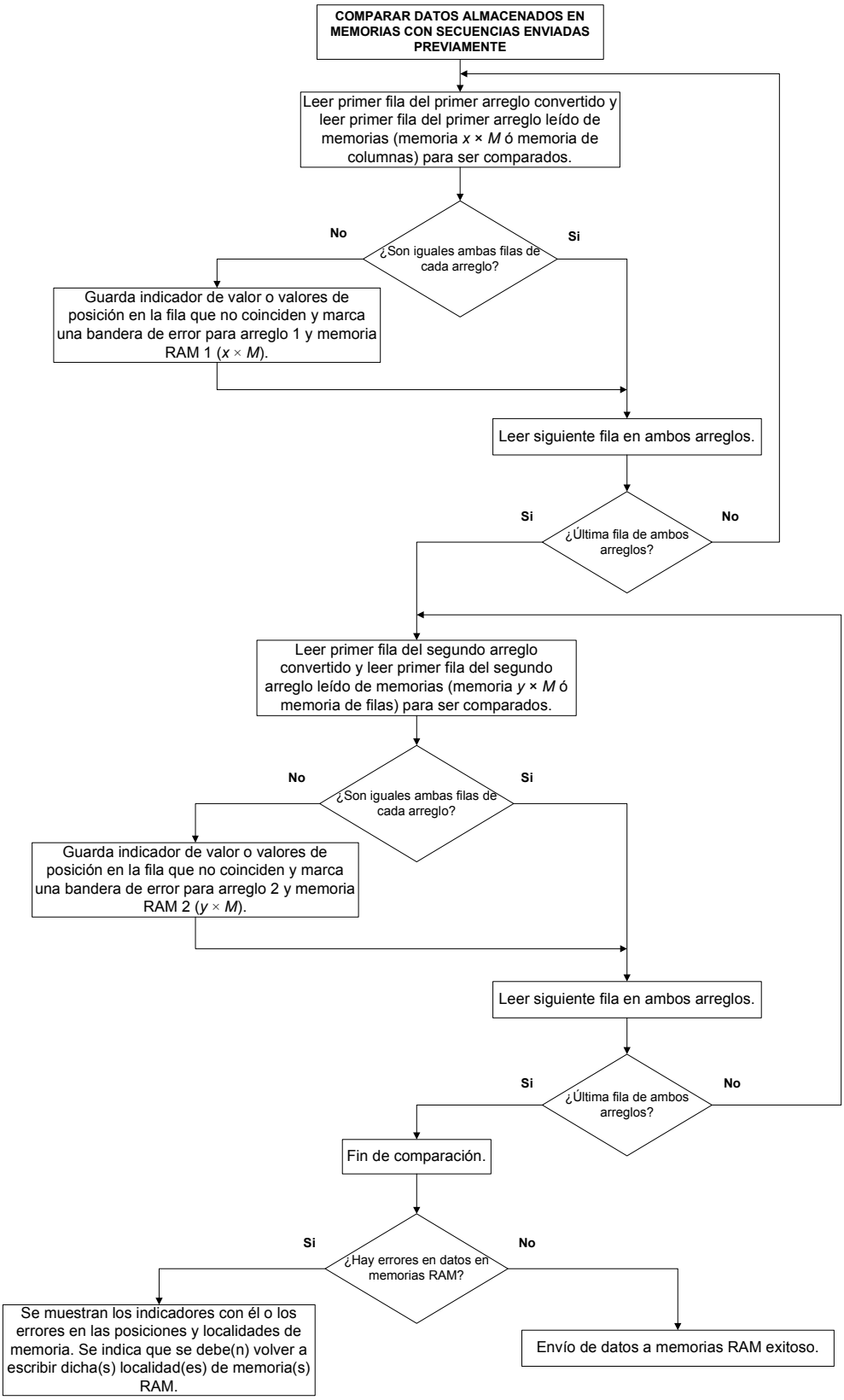


Figura 4.7. Algoritmo de comparación de datos en memorias con secuencias enviadas previamente.

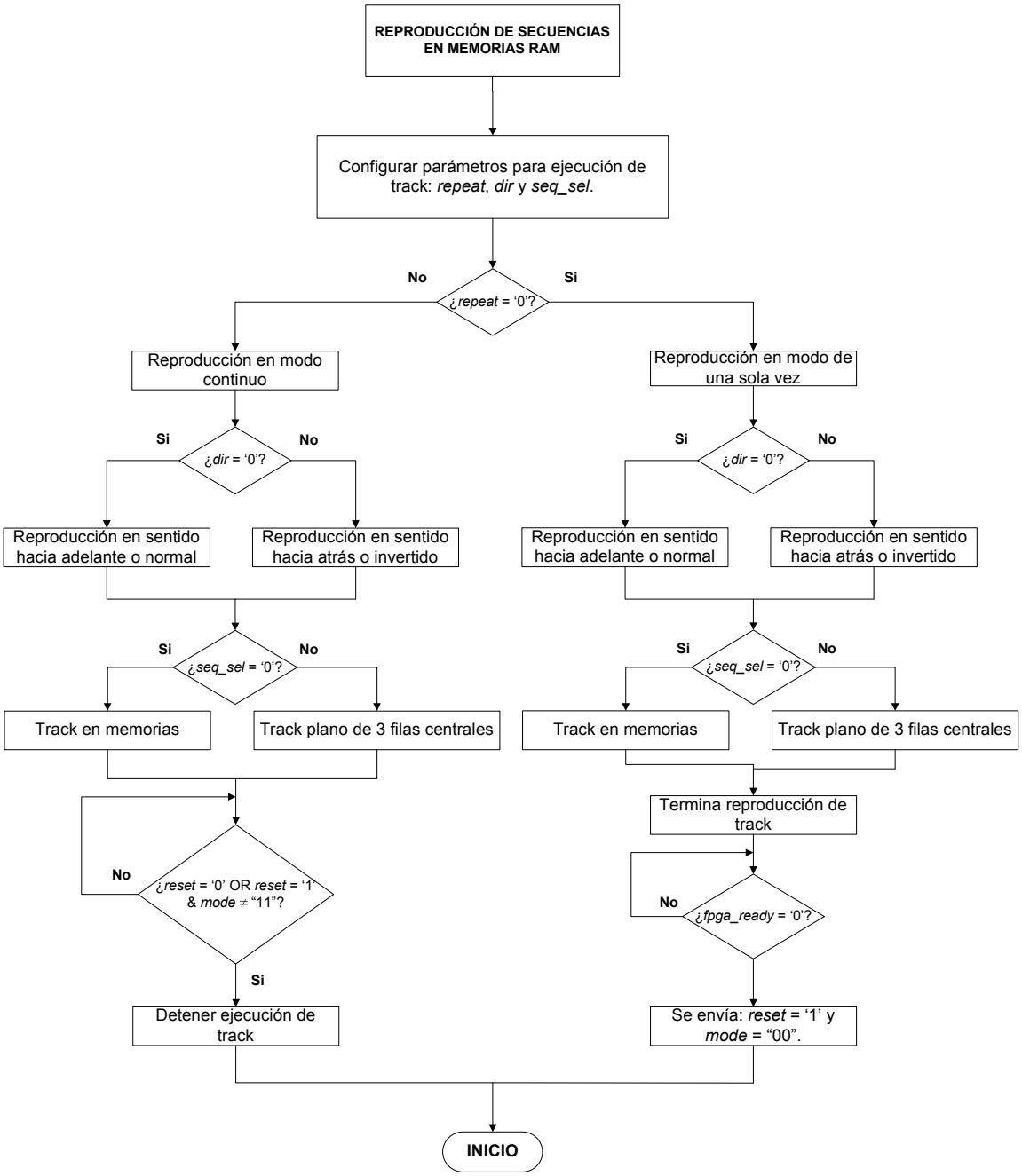


Figura 4.8. Proceso de reproducción de secuencias en memorias RAM.

Es importante señalar que cuando no se está ejecutando el algoritmo de transmisión de datos a memorias, el de recepción de datos de memorias o el de comparación (verificación) de datos en memorias, así como la reproducción de secuencias en memorias RAM, los bits de control *pc_ready*, *pc_do* y *ultimo* se mantienen todos en '0'. El bit de control *reset* se mantiene en '1' a menos que el usuario mande un "*reset*" al hardware desde el programa o

lo ejecute en el hardware mismo. Los bits que indican el modo de funcionamiento (*mode*) se mantienen en determinada combinación dependiendo del proceso que se encuentre realizando desde el programa (comúnmente en estado de espera ó *mode* = "00"). Mientras no se realice ningún proceso y se esté a la espera de que el usuario decida, *mode* se mantendrá en "00". Para los demás bits no cuenta mucho su estado, pero se mantendrán en cero todos (*data_in*, *byte_addr*, *addr*, *select_mem*, *repeat*, *dir* y *seq_select*). Si se presenta una interrupción mediante *reset* = '0', el programa retornará al estado inicial, colocando todos los bits de control en cero (*data_in* = '0', *byte_addr* = "000", *addr* = "000…0", *select_mem* = '0', *repeat* = '0', *dir* = '0' y *seq_select* = '0') y a partir de ahí, será el usuario quien decida la acción a realizar, recordando que una interrupción se presenta únicamente por parte del usuario mediante el bit de *reset*, y puede ser desde el programa o directamente en el hardware. Si la interrupción se realiza por hardware, el programa estará habilitado para saber cuando esto ocurre, mediante un bit de realimentación hacia el software.

Finalmente, a partir de los procesos expuestos en el presente capítulo, es posible pasar directamente al desarrollo del programa de control, el cual, se planea se ejecute dentro de un entorno Linux y se comunique mediante USB al hardware que controla y actúa directamente sobre la matriz de LEDs. Dicho programa constará de una interfaz de usuario gráfica y sencilla de manejar, con únicamente las opciones necesarias pero al mismo tiempo versátil.

Glosario

ABS plástico: Es un plástico muy resistente utilizado principalmente en la industria automotriz y de aparatos electrónicos. ABS significa Acrilonitrilo Butadieno Estireno. Proporciona resistencia a ataques químicos y estabilidad a altas temperaturas.

Antena tipo dipolo: Es una antena sencilla utilizada para la transmitir o recibir ondas de radiofrecuencia. La versión más sencilla del dipolo, consiste en dos elementos conductores rectilíneos colineales de igual longitud, alimentados en el centro, y de radio mucho menor que el largo. La longitud del dipolo es la mitad de la longitud de onda de la frecuencia de resonancia del dipolo, y puede calcularse como 150/frecuencia(MHz) para un resultado en metros.

Apertura de un telescopio: Representa la parte principal de un telescopio, la cual determina la cantidad de luz que entra al telescopio y forma la imagen primaria. Es importante para determinar la razón focal así como el poder de resolución del telescopio.

ASIC: Acrónimo que significa Circuito Integrado de Aplicación Específica. Consiste de un circuito integrado hecho a la medida para un determinado uso o función específica. Los sistemas ASIC más avanzados incluyen a los denominados Sistema en un Chip ó SoC (*System on a Chip*), los cuales son equivalentes a una microcomputadora con memoria Ram, ROM, EEPROM, Flash, líneas de E/S (entrada/salida), etc., pero diseñados para realizar una aplicación específica.

Banda Ku: La banda Ku se refiere a un segmento del espectro electromagnético dentro de la región de microondas, definido entre 12 y 18 GHz, utilizado principalmente en las comunicaciones satelitales y dividido por regiones geográficas en el mundo.

CCB: Acrónimo de Cluster Control Board. Se refiere a una sección de la electrónica del telescopio JEM-EUSO.

CCMOS: Se refiere a un tipo de lógica CMOS que utiliza una señal de reloj a fin de permitir que la operación del circuito sea independiente de la superposición de fase, con lo cual se evita un diseño largo y tedioso que tenga en cuenta la fase e inversión del reloj.

CPLD: Acrónimo de Complex Programmable Logic Device ó Dispositivo Complejo Lógico Programable, los cuales se refieren a dispositivos PLD o Dispositivos Lógicos Programables pero a un mayor nivel de integración, formado por múltiples bloques PLD y comunicados mediante una matriz de interconexiones programables.

CYTOP: Es un tipo especial de polímero de estructura amorfa o no cristalina con excelentes niveles de transmisión de luz en el rango visible – UV, además de contar con buena resistencia mecánica y contra químicos, repelente a aceite, agua y humedad.

Electroluminiscencia: Proceso de emisión de luz mediante la aplicación de una fuente de energía eléctrica [18].

ESAF: Acrónimo que significa *EUSO Simulation and Analysis Framework*. Representa una herramienta escrita para la simulación "end-to-end" de EAS en el contexto del experimento EUSO (Extreme Universe Space Observatory).

Flip-flop: Se refiere a la unidad mínima de información o elemento básico de memoria capaz de almacenar 1 bit. La información contenida en un conjunto de flip-flops es llamada "estado del circuito". Los flip-flops están formados por compuertas lógicas y siempre presentan 2 salidas, una válida y otra negada. Las entradas del flip-flop dependen del tipo de flip-flop del que se trate. También se les denomina "biestables" debido a que pueden permanecer en uno de sus dos estados posibles durante un tiempo indefinido en ausencia de perturbaciones.

FoV: Acrónimo de Field of View o campo de visión (también denominado Field of Vision), que se refiere a una región observable ya sea de forma angular o lineal en

determinado momento. En el caso de JEM-EUSO, el FoV de 60° representa el área angular vista por el instrumento.

Fonones: Son modos de vibración dentro de una estructura cristalina como la red atómica de un sólido y determinan muchas de las propiedades físicas de los materiales, principalmente dentro de la física del estado sólido, en donde condicionan la conductividad térmica y eléctrica. Debido a que también se analizan desde el punto de vista cuántico, se les ha dado ciertas propiedades de partículas (cuasipartículas), incluyendo el nombre, el cual surge en conjunción de sus propiedades de longitud de onda larga dentro de los sólidos, lo que da lugar al sonido.

FPGA: Acrónimo que significa *Field Programable Gate Array* o arreglo de compuertas programable en el campo. Este dispositivo electrónico contiene bloques lógicos los cuales pueden ser interconectados de forma *ad hoc* a la aplicación utilizando un lenguaje de descripción de hardware especializado, con la gran ventaja de ser reprogramables. Pueden desempeñar funciones tan sencillas como el de una compuerta lógica hasta complicados sistemas embebidos y sistemas en un chip o SoC [30].

FSM: Una máquina de estados finitos o *Finite State Machine* se refiere a una máquina de estados con estados bien definidos y delimitados. Básicamente consiste en un método o modelo que define el comportamiento de un sistema con entradas y salidas, en donde las salidas dependen no solo de las señales de entrada actuales sino también de las anteriores. En los sistemas digitales se tienen dos tipos principales de FSM: el modelo de Mealy y el modelo de Moore.

Fuerza fuerte: La fuerza fuerte o interacción nuclear fuerte, es una de las cuatro fuerzas o interacciones fundamentales existentes que el modelo estándar de física de partículas establece para explicar las interacciones entre las partículas conocidas. La fuerza fuerte es la responsable de mantener unidos a los nucleones de un átomo (protones y neutrones) venciendo a la repulsión electromagnética entre los protones y uniendo a los neutrones con los protones y entre sí.

H2B: Cohete japonés que en principio, será el que lleve a órbita al telescopio JEM-EUSO.

HTV: Acrónimo de *H-II transfer Vehicle* el cual realizará la maniobra de acoplamiento con la estación espacial internacional

IDAQ: Acrónimo de *Integrated Data Aquisition.*

LED-UV: LED o diodo emisor de luz ultravioleta.

MAPMT: Acrónimo de *Multi-Anode Photomultiplier* o fotomultiplicador multiánodo.

Microcontrolador: Un microcontrolador consiste de un circuito integrado que contiene una computadora en un chip. Básicamente contiene una unidad central de proceso, memoria y periféricos de entrada y salida.

Microsatélite: Son satélites artificiales de dimensiones y masa reducida que van de algunas decenas de kilogramos a cientos de kilogramos. Se utilizan en diferentes aplicaciones de baja razón de transmisión de información y/o en redes de microsatélites.

MOSFET: Acrónimo de *Metal Oxide Field Effect Transistor* o transistor de efecto de campo de óxido metálico. Es un tipo de transistor de efecto de campo que en la actualidad sirve de base para la creación de la gran mayoría de circuitos integrados y procesadores, así como para aplicaciones de electrónica de potencia.

Mpc: Acrónimo de *Megaparsec*, unidad de longitud empleada en astronomía. Se define como la distancia a la que una unidad astronómica (aproximadamente la distancia media entre la Tierra y el Sol) subtiende un ángulo de un segundo de arco (paralaje de 1 segundo de arco).

Muón: Partícula elemental que pertenece al grupo de los leptones en la familia de los Fermiones (sigue la estadística de Fermi) con un espín de 1/2. Posee carga eléctrica negativa como el electrón pero su masa es 200 veces mayor.

Neutrinos: Son un grupo de partículas subatómicas pertenecientes a la familia de los fermiones que no poseen carga eléctrica y con espín de 1/2. Poseen muy poca masa la cual sólo ha podido ser estimada más no mediada de forma precisa.

Neutrino Tau: Es una partícula elemental perteneciente al grupo de los leptones, con un espín de 1/2 y una masa pequeña, del orden de 8 millones de veces más pequeña que la del electrón, por lo cual, se mueve a una velocidad cercana a la de la luz.

Nightglows: Se refiere a los resplandores nocturnos observados en la región espectral de 300 a 400nm, referentes a la emisión en moléculas de oxígeno en la banda espectral Herzberg I [28], a una altura de 95 km entre la mesosfera y la termosfera. Se ha reportado que esta emisión tiene una fuerte correlación con la línea verde de 557.7nm del átomo de oxígeno [11]. La "rayas" de la emisión en verde (con un ancho de 40 km) has sido descritas con un movimiento a partir del punto de observación desde tierra, y se piensa que estas rayas son producidas por ondas gravitacionales formadas en la troposfera y propagadas hacia la atmósfera alta. Esta propagación de gravedad puede afectar la energía y el momento angular transferido desde la mesosfera y la termosfera [11].

Nadir: Se define como nadir a la posición hacia debajo de nuestros pies, localizada en una recta imaginaria desde nuestra ubicación en la superficie terrestre y que pasa por el centro de la Tierra. En sentido contrario sobre esa misma línea imaginaria (por encima de nuestra cabeza) se ubica el Cenit.

Partículas alfa: Son núcleos de helio completamente ionizados, es decir, sin electrones. Se conforman de 2 protones y 2 neutrones, con carga eléctrica de $+2q_e$ y una masa de 4 uma (unidades de masa atómica). Se producen en reacciones nucleares o desintegración

radiactiva de núclidos (agrupaciones de protones y neutrones denominados nucleones) que transmutan en elementos más ligeros mediante la emisión de dichas partículas alfa.

Pión: Se denomina Pion a un conjunto de 3 partículas subatómicas π^0 π^+ π^-. Estas partículas representan al mesón (bosón ó hadrón con espín entero, el cual responde a la interacción fuerte) más ligero.

Polisilicio: Es un material que consiste de pequeños cristales de silicio, utilizado en la construcción de celdas solares, y a partir de silicio amorfo, para dispositivos de películas delgadas. También es convertido a silicio de cristales simples, el cual es utilizado en la industria microelectrónica.

PWM: Acrónimo que significa *Pulse Width Modulation* o modulación por ancho de pulso.

Radiación Cherenkov: Es una radiación de tipo electromagnético, la cual se produce por el paso de partículas cargadas en un medio dieléctrico con velocidades superiores a la de la luz en dicho medio.

Rayos cósmicos: Los rayos cósmicos son partículas subatómicas, núcleos de hidrógeno y núcleos pesados de alta energía, con energías que van desde 10^9 eV hasta los más energéticos de 10^{20} eV. Al colisionar con las partículas de la atmósfera terrestre producen partículas secundarias de menor energía, las cuales producen a su vez más partículas en lo que se conoce como una lluvia de partículas ó EAS (*Extended Air Shower*). Algunos rayos cósmicos son producidos en el Sol, sin embargo, los de mayor energía, se producen fuera del sistema Solar e inclusive fuera de la Galaxia [29].

Rayos x: Parte del espectro electromagnético con energías en el rango de 1 a 150 keV ó entre 10nm y 0.1nm de longitud de onda [29]. Es una radiación ionizante con un origen a partir de fenómenos extranucleares, a nivel de la órbita electrónica, producidos fundamentalmente por la desaceleración de electrones.

Rayos γ: Radiación electromagnética con energías por encima de 0.1 MeV y longitud de onda menor a 0.1Å ó 10^{-11}m [29]. Se producen en procesos radiactivos o procesos subatómicos como la aniquilación electrón-positrón, así como fenómenos astrofísicos violentos como los GRB (*Gamma Ray Busrt*), los cuales son "flashes" de rayos gama de gran magnitud con duraciones que van de unos segundos a unas pocas horas.

ROOT: Se trata de una herramienta para el procesamiento de datos así como simulación de procesos físicos principalmente en el área de física de altas energías. Surgió en el CERN como herramienta de apoyo y ha sido difundido a otros campos de la física.

RTL: Acrónimo de **Register Transfer Level** o nivel de transferencia de registro, el cual se refiere a un nivel de abstracción empleado para la descripción de circuitos digitales síncronos. En un diseño RTL el comportamiento de un circuito se define en términos del flujo de las señales (o transferencia de datos) entre los registros de hardware y las operaciones lógicas realizadas por estas señales. La utilización de diseños a nivel RTL permite crear representaciones de alto nivel y detalle de los circuitos digitales.

Ondas milimétricas: Las ondas milimétricas constituyen una región de las microondas de mayor frecuencia y menor longitud de onda del orden de milímetros. Las microondas oscilas entre 300MHz y 300GHz, con longitudes de onda de entre 1m y 1mm.

Neutrinos cósmicos: Se piensa que los neutrinos cósmicos son, al igual que la radiación de fondo de microondas, remanentes del Big Bang. Son neutrinos de baja energía a los que se les asocia a la materia oscura.

Netlist: Un *netlist* o lista de conexiones, es la primera forma de describir un circuito mediante un lenguaje, que consiste en dar una lista de componentes, con entradas y salidas pero sin interconexiones. No se trata de un lenguaje de alto nivel, por lo que no describe cómo funciona el circuito, sino que simplemente se limita a describir los componentes que posee y las conexiones entre ellos [30].

Materia oscura: Se denomina materia oscura a la materia que no radia de forma electromagnética de manera que no puede ser observada directamente mediante telescopios y técnicas actuales de observación, sin embargo, se deduce o se hace hipotética su existencia a partir de los efectos gravitacionales que ejerce sobre la materia visible como estrellas y galaxias, además de fungir como posible explicación de las anisotropías del fondo cósmico de microondas del universo. Para más detalles ver [29].

Ondas gravitacionales: Una onda gravitacional se refiere a una ondulación o fluctuación de la curvatura del espacio-tiempo, producida por la aceleración de un cuerpo masivo o por sistemas de objetos que gravitan entre sí y que se transmiten a la velocidad de la luz.

TrenchMOS: Tipo de tecnología de construcción de transistores MOSFET y MOSFET de potencia, la cual logra un mejor rendimiento en cuanto a consumo de potencia, incrementa la capacidad de corriente y reduce las dimensiones del dispositivo, siendo posibles encapsulados de montaje superficial.

VHDL: Acrónimo de la combinación de VHSIC (*Very High Speed Integrated Circuit*) y HDL (*Hardware Description Language*). Representa un tipo de lenguaje de descripción de hardware empleado para el diseño de circuitos digitales, y que se encuentra definido por el IEEE (*Institute of Electrical and Electronics Engineering*) bajo la ANSI/IEEE 1076-1993. La ANSI es la *American National Estándar Institute* o Instituto Estadounidense de Estándares y que se encarga de supervisar el desarrollo de estándares para productos, servicios y sistemas en los Estados Unidos y que, por lo general, son aceptados en muchas partes del mundo.

VHF: Acrónimo de *Very High Frecuency* o frecuencia muy alta. Se refiere a la banda del espectro electromagnético en el rango de 30MHz a 300MHz.

Wire wragp: Tecnología utilizada para la conexión de dispositivos electrónicos sin la necesidad de realizar placas de circuito impreso, utilizada para la fabricación a gran escala durante la década de los años 60s y 70s. En la actualidad se utiliza principalmente para la

realización de prototipos. Se pueden realizar de forma automática o manual, mediante herramientas dedicadas para esto.

Referencias

[1] Judith A. Irwin, 2007, *Astrophysics, decoding the cosmos*. John Wiley and Sons, Ltd.

[2] R. A. Mewaldt, 1996, *Cosmic Rays*. *Macmillan Encyclopedia of Physics*.

[3] Hannu Karttunen, Pekka Kröger, Hikki Oja, Markku Poutanen, Karl J. Donner. *Fundamental Astronomy*. Springer 5th edition.

[4] R. A. Mewaldt, October 7, 2009, *Solar and Galactic Sources of Precipitating Energetic Particles*, HEPPA 2 Workshop, Boulder CO.

[5] Neutron Monitor Database (NMD-B). *Impact: Technological and biological effects of cosmic rays*. Posted March 24, 2009 - 9:40am by Nicolas Fuller. http:// http://www.nmdb.eu/?q=node/137.

[6] Sievert. http://www.sievert-system.org/WebMasters/en/index.html.

[7] Astro Cosmos, *Cómo detectar rayos cósmicos*. Pagina web. http://www.astrocosmo.cl/astrofis/astrofis-03_14.htm

[8] University of Utah, *High Resolution Fly's Eye*. http://www.cosmic-ray.org/.

[9] Diego Gabriel Melo. *El Detector de fluorescencia del Observatorio Pierre Auger: Reconstrucción de lluvias de partículas, análisis de los primeros datos y extensión híbrida del detector de fluorescencia a energías* $\leq 10^{18}$ *eV*. Tesis de doctorado.

[10] Observatorio Pierre Auger. http://www.auger.org.ar/argentina/index.shtml

[11] JEM-EUSO collaboration, *Report on the Phase A Study 2010*.

[12] C. Berat, S. Bottai, D. De Marco, S. Moreggia, D. Naumov, M. Pallavicini a,b, R. Pesce, A. Petrolini, A. Stutz, E. Taddei, A. Thea, *Full simulation of space-based extensive air showers detectors with ESAF*, Astroparticle Physics 2010.

[13] Zarya, Soviet, Russian and International Space Flight site. http://www.zarya.info/Frequencies/FrequenciesISS.php.

[14] NASA web site. http://www.nasa.gov/home/hqnews/2010/jan/HQ_M10-011_Hawaii221169.html.

[15] H. Hemmati. Status *of Free-Space Optical Communications Programme at JPL*. IEEE Aerospace Conference. Proceedings, 3, 101 - 105 (2000).

[16] Il H. Park, *UFFO, Space Heritage for JEM-EUSO*, JEM-EUSO AMS meeting at Torino, Oct. 25 – 29, 2010.

[17] E. Fred Schubert, 2003, *Ligth-Emitting Diones*. Cambridge University press.

[18] Robert L. Boylestad, Lous Nashelsky, 2003, *Electronic Devices and Circuit Theory*. Pearson Education.

[19] Z. Zhu, A. Kathuria, S.G. Krishna, M. Mojarradi, B. Jalali-Farahani, H. Barnaby, W. Wu, and G. Gildenblat, *Design applications of compact MOSFET model for the extended temperature range (60-400K)*. Electronics Letters, Vol. 47, No. 2, 2011, 141-142.

[20] Z. Zhu, G. Gildenblat, *Symmetrically linearised charge-sheet model for extended temperature range*. Electronics Letters, 45 (2009), 346-348.

[21] G. Gildenblat, Z. Zhu, C.C. McAndrew, *Surface potential equation for bulk MOSFET*. Solid-State Electronics, 53 (2009) 11–13.

[22] X. Li, C. C. McAndrew, W. Wu, S. Chaudhry, J. Victory, G. Gildenblat, *Statistical modeling with the PSP MOSFET model*. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol. 29, 2010, pp. 599–606.

[23] G. Gildenblat, X. Li, H. Wang, W. Wu, R. van Langevelde, A.J. Scholten, G.D.J. Smit and D.B.M. Klaassen, *Introduction to PSP MOSFET Model*, Presented at the 2005 Workshop on Compact modeling in Annaheim CA, May 2005; Technical Proceedings, pp. 19-24.

[24] Z. Zhu, G. Gildenblat, C. C. McAndrew, and I.-S. Lim, *Modeling the Frequency Dependence of MOSFET Gate Capacitance*. 2011 IEEE Conference on Microelectronic Test Structures, April 4-7, Amsterdam, The Netherlands, pp. 13-18.

[25] Volnei A. Pedroni, *Circuit Design with VHDL*, 2004 MIT Press, Cambridge Massachusets, London, England.

[26] Stanford Rserch System, *Signal Recovery with PMTs*, Application Note 4. www.thinkSRS.com.

[27] TrenchMOS or Trenchgate MOS overview, Radio-Electronics site. http://www.radio-electronics.com/info/data/semicond/fet-field-effect-transistor/trenchmos-gate-transistors.php. Consultado al 13-10-2011.

[28] Alain Jenouvrier, Marie-France Me´rienne, Bernard Coquart, Michel Carleer, Sophie Fally, Ann Carine Vandaele, Christian Hermans, and Reginald Colin, *Fourier Transform Spectroscopy of the O_2 Herzberg Bands*. Journal of Molecular Spectroscopy 198, 136-162 (1999).

[29] Richard A. Matzner, *Dictionary of Geophysics, Astrophysics and Astronomy*. CRC Press 2001.

[30] Fernando Pardo, José A. Boluda, *VHDL Lenguaje para síntesis y modelado de circuitos*. Segunda Edición, Alfaomega 2004.