Universidad Nacional Autónoma de México

Programa de Maestría y Doctorado en Ingeniería

Ingeniería Eléctrica - Sistemas Electrónicos

Diseño de la Arquitectura de un Procesador Dedicado a la Reconfiguración Electrónica de un Arreglo Reflectivo tipo Espirafase

T e s i s

Que para optar por el grado de:

Maestro en Ingeniería

P r e s e n t a

Oscar Iván Menéndez Rosas

T u t o r

Dr. Jorge Rodríguez Cuevas, Facultad de Ingeniería

México, D.F. Septiembre 2016.

Jurado Asignado:

Presidente:	Dr. Savage Carmona, Jesús.
Secretario:	Dr. De La Rosa Nieves, Saúl.
Vocal:	Dr. Rodríguez Cuevas, Jorge.
1er Suplente:	Dr. Martínez López, José Ismael.
2do Suplente:	Dr. Matatagui Cruz, Daniel.

Lugar donde se realizó la tesis: Ciudad Universitaria, México D.F.

Tutor de Tesis:
Dr. Jorge Rodríguez Cuevas.

_____

Firma

A un gran amigo,
quien probablemente habría
compartido el camino conmigo.
R. de J. Ruvalcaba S. †

Agradecimientos

La redacción de las líneas contenidas en este trabajo me resultó una tarea bastante laboriosa. Sin embargo, puedo asegurar que el párrafo más difícil de completar fue este, ya que simplemente no encuentro la combinación de palabras que expresen adecuadamente lo agradecido que estoy con mi madre. Conseguir este logro es resultado de una gran dedicación y disciplina, virtudes que la caracterizan y que yo trato de emular en mi vida. Estoy plenamente convencido de que sin su cariño, paciencia y apoyo no sería quien soy, ni habría llegado hasta donde me encuentro. Estas cualidades, entre otras, me dificultan imaginar una mejor figura materna, razón por la cual me considero una persona sumamente afortunada.

Indudablemente mi padre representa otro gran soporte para mi. El impulso para seguir adelante y una amplia gama de consejos y lecciones de vida son sólo algunas de sus aportaciones. Y a pesar de no haberme convencido de seguir sus pasos, lo encuentro muy orgulloso de haber tomado la decisión de caminar por la misma Universidad que él recorrió hace algunas generaciones. Por su parte, con mi hermano he compartido una gran cantidad de experiencias que me han hecho crecer y al mismo tiempo me complementan como ser humano. Además, como lo hacen los hermanos mayores, siempre ha estado al pendiente de mi. Y casi hermanos se encuentran mis primos, de quienes recibo un gran afecto. Especialmente el mayor, con quien además comparto aficiones que han reforzado nuestro vínculo. Por supuesto, siempre cariñosas y atentas conmigo se encuentran tanto tías como abuelos, quienes no dejan de recordarme la importancia de la familia.

Desde el punto de vista académico, debo a una gran cantidad de profesores la formación con la que cuento, ya que este camino comenzó hace muchos años y es gracias a ellos que estoy aquí. Dicha trayectoria se vio influenciada por mi ingreso a esta hermosa Universidad, que habría de convertirse en mi segunda casa y que me brindó la oportunidad de cruzarme con académicos notables que desarrollaron en mi el amor por el estudio con esa pasión al enseñar que les caracteriza, misma que les exhorto a que nunca abandonen.

Fue precisamente al llegar al final de mi carrera conocí a un profesor que me impulsó a ingresar al programa de posgrado y habría de convertirse en mi tutor en la maestría. Una vez

dentro del programa, aún cuando me tomó algo de tiempo aclarar mi tema de investigación, siempre conté con su apoyo y confianza. Así mismo, mi ingreso me llevó a formar parte del grupo de trabajo de Radiofrecuencia (RF) y microondas, donde me encontré con profesores y compañeros siempre atentos y dispuestos a cooperar como se hace en un buen grupo de trabajo. Fue así como logré despejar las dudas que finalmente me permitieron enfocar adecuadamente mis esfuerzos.

En el aspecto personal, me resulta imposible concebirme donde me encuentro sin las amistades. Especialmente aquellas que formé en mi infancia, tanto vecinos como compañeros de la escuela, que me han acompañado toda mi vida y con quienes comparto memorias invaluables. Un poco más tarde, y no por ello menos importantes, se encuentran los compañeros de la carrera, con quienes compartí este esfuerzo que implica llegar a convertirnos en Ingenieros. Indudablemente representaron un gran apoyo al cursar la carrera y, a la fecha, ese compañerismo se ha transformado en una grata amistad. Sin embargo, hacia el final de mi carrera, condiciones externas me alejaron del grupo de trabajo con el que solía rodearme, punto a partir del cual me encontré con una gran compañera y amiga, con quién formé un estupendo equipo y a quien agradezco haber compartido esa etapa conmigo.

Fue con ella con quien compartí cierta clase cuyo profesor habría de cambiar mi rumbo, ya que gracias a él me encontré con un excelente asesor para realizar mi servicio social. Durante dicha estancia conocí a una estupenda persona que resultó ser el líder de un equipo de trabajo integrado por compañeros sobresalientes. Tras una invitación, colaboré con ellos en algunos proyectos y compartí gratas experiencias. Antes de que el equipo se disolviera, encontré ahí el impulso para ingresar a la maestría y, a la fecha, cuento con el apoyo y amistad de los integrantes.

Al comenzar el segundo año de la maestría tuve la fortuna de reencontrarme con un amigo de la licenciatura que acababa de incorporarse al posgrado. Con él compartí una clase cuyo profesor habría de marcar de forma contundente el curso de mi tesis. De igual forma, gracias a él me reencontré con un excelente profesor de la licenciatura, a quién agradezco la amabilidad y confianza con la que me abrió las puertas de su laboratorio, donde encontré un estupendo grupo de compañeros con quienes he formado una grata amistad.

Al finalizar el segundo año, este trabajo llegaba a su etapa final. Fue entonces cuando los integrantes del jurado comenzaron a revisarlo, por lo que les agradezco el tiempo invertido en ello; especialmente a uno de ellos que, sin ser mi tutor, me impulsó continuamente a mejorarlo. Y precisamente en esta etapa, el mismo *maigo* con el que llevaba un año compartiendo laboratorio, me hizo una serie de observaciones que enriquecieron notablemente esta tesis.

Un poco más alejados, pero siempre atentos conmigo se encuentran algunas amistades de mis padres, quienes me tienen un gran afecto y me ofrecen un respaldo incondicional. Entre ellos se encuentra mi padrino, a quien debo mi primer trabajo, y algunos a quienes he llegado a considerar como mis propios tíos.

Ampliando un poco el panorama, considero que la presencia de todas estas valiosas personas, y aquellas que por traición de mi memoria he omitido, se la debo a la Fuerza Vital que impulsa todo aquello que está presente en nuestras vidas, buscando continuamente la forma de maravillarnos con todo lo que nos rodea en este tiempo y espacio que nos tocó compartir, razón por la cual estoy eternamente agradecido.

Finalmente, quiero agradecer al Consejo Nacional de Ciencia Y Tecnología (CONACYT) por el apoyo otorgado. De igual forma, por parte del grupo de trabajo, se agradece a la misma instancia el apoyo del proyecto 166106; así como al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) mediante los proyectos IN114116, IN115016 y IN117814.

Índice general

Resumen

Se presenta el diseño de la arquitectura de un procesador segmentado de cuatro etapas, cuyo objetivo es la reconfiguración electrónica de un arreglo reflectivo conformado por 367 radiadores de dos bits, lo que implica el uso de cuatro elementos conmutadores en cada radiador, es decir, 1468 líneas de control independientes.

El sistema se desarrolló en el lenguaje de descripción de hardware VHDL y es capaz de admitir cualquier dirección del haz principal formada por un ángulo de elevación entre 0° y 90°, así como un ángulo de azimut entre 0° y 360°, ambos con incremento de 1°. El cálculo de posición angular de cada anillo, antes de aproximarse a posiciones discretas, presentó un error promedio de 0.1° respecto a la calculada mediante el programa Octave.

Utilizando el programa Quartus II de Altera, la simulación del sistema reveló un tiempo de 97.68 [μs] para el cálculo secuencial de los 367 radiadores con una frecuencia de reloj de 50 [MHz]. Sintetizando el diseño en un arreglo de compuertas programables en campo FPGA de la familia Cyclone IV de Altera, una señal de control que se habilita al terminar el cálculo del arreglo permitió medir un tiempo de cálculo de 98 [μs].

Apoyándose en una tarjeta de desarrollo DE2-115 de Terasic, fue posible visualizar cualquier conjunto de 26 señales de control mediante los diodos emisores de luz con los que cuenta la tarjeta, verificando físicamente el correcto cálculo de las señales.

Finalmente, el diseño presenta potencial para aplicar el paralelismo a nivel de núcleos, con lo que se puede llegar a obtener un tiempo de cálculo de aproximadamente 300 [ns] con frecuencia de reloj de 50 [MHz]. Si la etapa más lenta del cauce así lo permite, es posible utilizar dispositivos capaces de trabajar a frecuencias de reloj más elevadas para disminuir este parámetro.

Introducción

Aviones de alto desempeño, naves espaciales, satélites y misiles son aplicaciones que demandan antenas con ciertas restricciones en tamaño, peso, costo, desempeño, instalación y perfil aerodinámico. Actualmente existen otras aplicaciones militares y comerciales, como la radio móvil o comunicaciones inalámbricas, que cuentan con especificaciones similares. Las antenas basadas en radiadores tipo microcinta cuentan con un bajo perfil que les permite satisfacer buena parte de ellas [1].

Por su parte, arreglos de radiadores microcinta alimentados por una fuente externa forman un sistema conocido como arreglo reflectivo, cuya característica más atractiva es la capacidad de generar un haz reconfigurable electrónicamente, la cual los hace idóneos para aplicaciones de radar en donde se requiere un rápido y preciso redireccionamiento del haz. Esto, aunado a la capacidad de obtener patrones de radiación altamente directivos, les permite incorporarse a una amplia variedad de sistemas de comunicaciones.

Tradicionalmente, los arreglos están conformados por miles de elementos radiadores, factor que a su vez representa un reto de diseño, ya que entre más elevado sea el costo del elemento unitario, más se eleva el costo del sistema. Esto ha provocado que múltiples grupos a nivel mundial adopten el reto de diseñar un desplazador de fase de bajo costo. Desde hace algunos años, el grupo de RF y Microondas de la Facultad de Ingeniería de la UNAM se incorporó a esta línea de investigación y ha desarrollado múltiples diseños que involucran el uso de anillos como elementos radiadores. A pesar del planteamiento de la reconfigurabilidad en distintas publicaciones del grupo, actualmente no se cuenta con un prototipo que presente dicha característica debido, en parte, a la falta de un sistema que sea capaz de realizar eficientemente la conmutación de las cargas reactivas en cada radiador para redirigir el haz principal.

Es en el marco de esta problemática que se pretende justificar el presente trabajo de tesis, ya que resulta clara la necesidad de contar con un sistema de control que permita obtener arreglos capaces de generar un haz redireccionable electrónicamente, característica que los hace tan atractivos. Cabe destacar que, como sucede frecuentemente en ingeniería, existen múltiples formas de resolver el problema. Sin embargo, tras una exploración de otras opciones, las atractivas ventajas y el gran potencial que presenta una en particular, condujeron al autor a evaluar el desempeño de la misma, no con ello descartando el uso de otra tecnología.

Objetivo

Diseñar la arquitectura de un procesador segmentado de cuatro etapas que permita realizar el control independiente de cada elemento radiador para lograr la reconfiguración electrónica de un arreglo reflectivo tipo espirafase conformado por 367 radiadores de dos *bits*.

Estructura del Trabajo

Este trabajo se divide en cuatro capítulos. El primero comienza con un breve recorrido histórico sobre las antenas hasta llegar a la los *arreglos reflectivos* basados en anillos resonantes tipo *espirafase*, abordando su principio de operación, estado del arte y tiempo de cálculo que presentan arreglos aplicados en sistemas de radar.

El capítulo dos aborda el concepto de *procesador* como elemento central de prácticamente cualquier sistema digital. Se presentan el surgimiento de la arquitectura *secuencial* y los principales atributos de un procesador, así como los niveles de aplicación del *paralelismo* y la emulación del mismo mediante la *segmentación encausada*. Finalmente, se muestra la *lógica programable* como una herramienta ideal para realizar el diseño y síntesis de la arquitectura de un procesador en los niveles de *abstracción* que resultan adecuados para obtener un sistema funcional, así como la *metodología de diseño* más apropiada.

En el capítulo tres se delimita el problema a resolver, presentando el cálculo de la *distribución de fase*, así como el arreglo en el cual se basará el diseño del sistema y los requisitos con los que debe cumplir. Se muestran las herramientas y el proceso de diseño del procesador, abordando las instrucciones que será capaz de ejecutar, así como la descripción del *hardware* necesario para tal efecto. Finalmente se proponen distintas técnicas para la *expansión de pines* del dispositivo a utilizar, así como la *escalabilidad* que presenta el diseño para minimizar el tiempo de cómputo.

Por último, en el capítulo cuatro se presentan los *resultados* del diseño propuesto. Utilizando el programa Quartus II se obtuvo el *tiempo de cálculo* a través de simulaciones, así como el *consumo* de recursos lógicos. Mediante el programa Octave se genera una simulación *gráfica* para verificar los valores obtenidos por el procesador. Se muestra la *síntesis* del diseño en una tarjeta de desarrollo DE2-115 de Terasic, la cual cuenta con FPGA de la familia Cyclone IV de Altera, donde se verifica *físicamente* tanto el tiempo de cálculo como la obtención de las líneas de control. Posteriormente, se propone una *tarjeta dedicada* que permita incorporar el sistema a un arreglo reflectivo, ya que la filosofía de propósito general de las tarjetas comerciales impiden el acceso a todos los pines disponibles del dispositivo. Finalmente se presenta una breve comparativa con dos soluciones alternativas para *concluir* así con las ventajas que presenta el sistema y los aspectos en los que podría mejorar a *futuro*.

Capítulo 1

Antecedentes sobre Antenas

Una antena es cualquier estructura que permite la transición de energía electromagnética entre una guía de onda y el espacio libre, como se muestra en la figura 1.1. La guía de onda o línea de transmisión se utiliza para transportar la energía desde la fuente hasta la antena y viceversa [1].

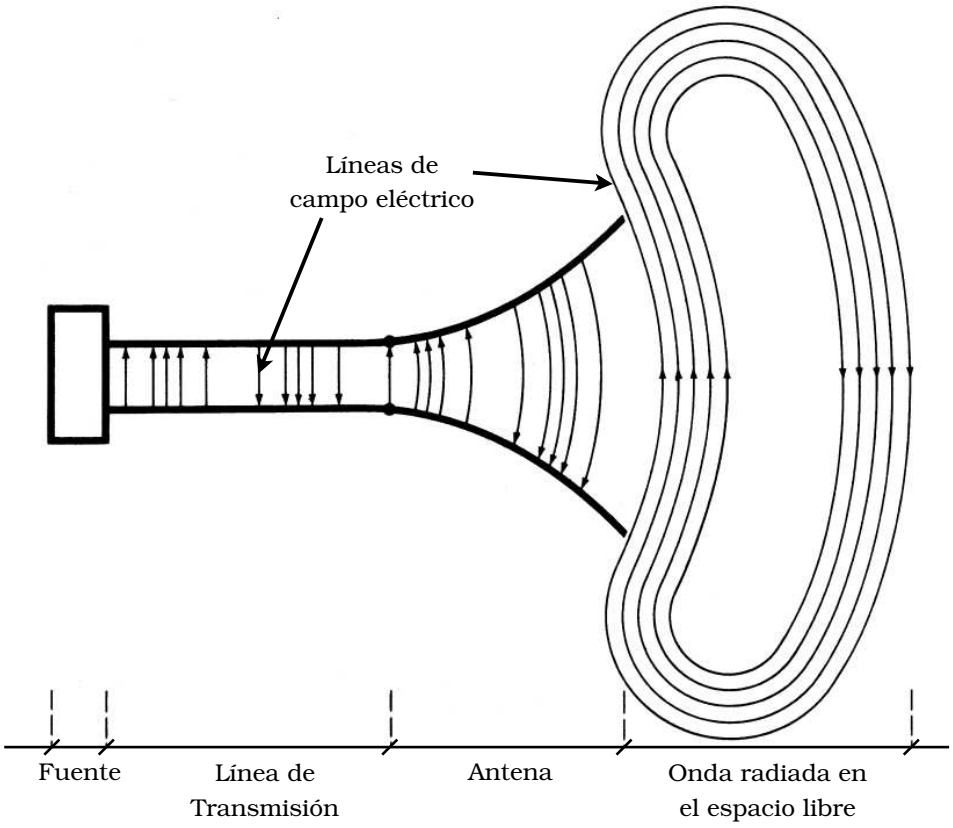


Figura 1.1: La antena como elemento de transición [1].

Por su parte, un arreglo reflectivo de elementos radiadores tipo microcinta es, en esencia, una antena, la cual tiene su fundamento en el fenómeno de la *radiación electromagnética.* Por tal razón, en este capítulo se presenta una breve reseña histórica que comienza con el descubrimiento del electromagnetismo y termina en el estado del arte de los arreglos reflectivos reconfigurables basados en anillos resonantes.

1.1. Descubrimiento del Electromagnetismo

La primera evidencia respecto a la relación entre la fuerza eléctrica y magnética se encuentra publicada en el *"Risetto dei foglietti universali di Trento"*, el 3 de agosto de 1802, bajo el nombre de *"Articolo sul galvanismo"*. El autor fue el jurista italiano Gian Romagnosi, quien reportó la deflexión de la aguja de una brújula en presencia de una corriente eléctrica circulando a través de un conductor (figura 1.2). Sin embargo, él mismo no le dio importancia al fenómeno y, por lo tanto, no hizo alusión alguna al descubrimiento del electromagnetismo [2].

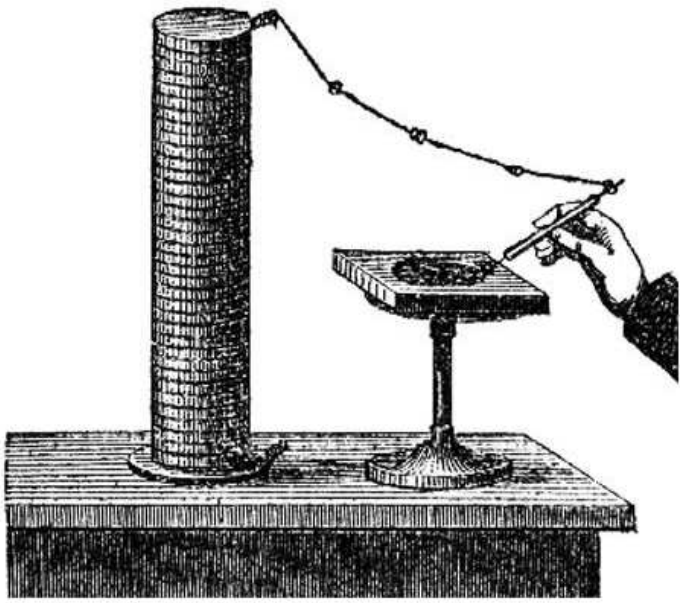


Figura 1.2: Experimento de Romagnosi en 1802 [2].

Por otro lado, en la Universidad de Copenhague, durante una clase impartida el 21 de abril de 1820, el mismo fenómeno fue observado por el físico danés Hans Christian Ørsted. Contrario a la concepción común de haber descubierto el electromagnetismo por casualidad, sus publicaciones científicas demuestran que se encontraba buscando dicho efecto, razón por la que se le adjudica su descubrimiento [3].

Para octubre de aquél año, la noticia del descubrimiento llegó al Instituto Real de Gran Bretaña, donde Michael Faraday llevaba siete años como asistente de Sir Humphry Davy. Aún cuando casi de inmediato repitieron y confirmaron los experimentos de Ørsted, en aquél mo-

mento Faraday se encontraba ocupado en sus investigaciones sobre metalurgia. Fue hasta el verano de 1831 que comenzó un intensivo periodo de experimentos sobre fenómenos electromagnéticos. Uno de ellos consistió en formar dos bobinas en los extremos opuestos de un anillo de hierro de aproximadamente 15 [cm] de diámetro, como se muestra en la figura 1.3. En una de ellas conectó una batería y en la segunda un galvanómetro, esperando medir una corriente inducida en el secundario. Para su sorpresa, observó que la aguja se desviaba únicamente al conectar o desconectar la batería, evento que que culminaría en el descubrimiento de la *inducción electromagnética* [4].

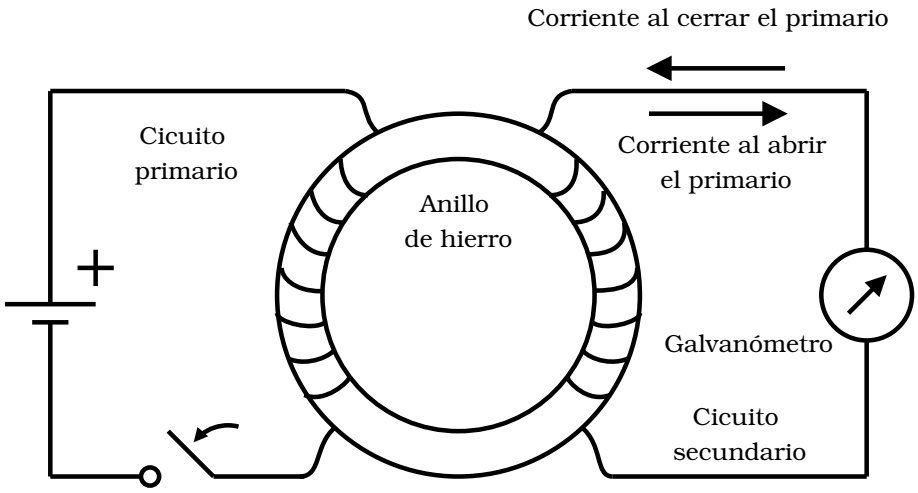


Figura 1.3: Experimento de Faraday sobre inducción electromagnética.

Cabe mencionar que, tras presentar su trabajo *"Experimental Researches in Electricity"* ante la Real Sociedad, Faraday escribió un documento en el cual declara que sus observaciones lo llevan a creer que la acción magnética es progresiva y requiere determinado tiempo de transmisión. También menciona que este comportamiento se podría comparar con las ondas que se generan al perturbar el agua (figura 1.4) o las que se producen en el aire como efecto del sonido: "Por lo tanto, me inclino a creer que la teoría vibratoria es aplicable a este fenómeno como lo es al sonido y, más probablemente, como a la luz" [5]. La extraordinaria precisión de su predicción nos llevan a citarlo como el primer científico que predijo la existencia de las *ondas electromagnéticas*.

1.2. Ondas Electromagnéticas

Como resultado de someter las hipótesis de Faraday a un análisis matemático riguroso, así como resumir y a su vez ampliar sustancialmente los diversos conceptos eléctricos conocidos hasta aquél momento, el físico matemático James Clerk Maxwell presentó en 1864, ante la Real Sociedad, las fórmulas matemáticas que modelaban la propagación de las hipotéticas ondas electromagnéticas a la velocidad de la luz [3].

Dos años después y cerca de 35 años antes del establecimiento de la radio comercial, el dentista americano Mahlon Loomis estaba al borde de descubrir el radio-telégrafo, cuando demostró la transmisión de señales entre dos montañas en el Blue Ridges, a una distancia de aproximadamente 20 millas. Conectando y desconectando el galvanómetro de una estación, obtenía una clara deflexión del galvanómetro en la otra estación. Loomis asumía que la tierra estaba rodeada de una capa de electricidad estática, que llamó *mar estático*, a través de la cual se propagaban las ondas eléctricas. El 30 de julio de 1872 obtuvo la patente americana por su *telégrafo aéreo*, la cual se considera la primera patente relativa a la telegrafía inalámbrica [3].

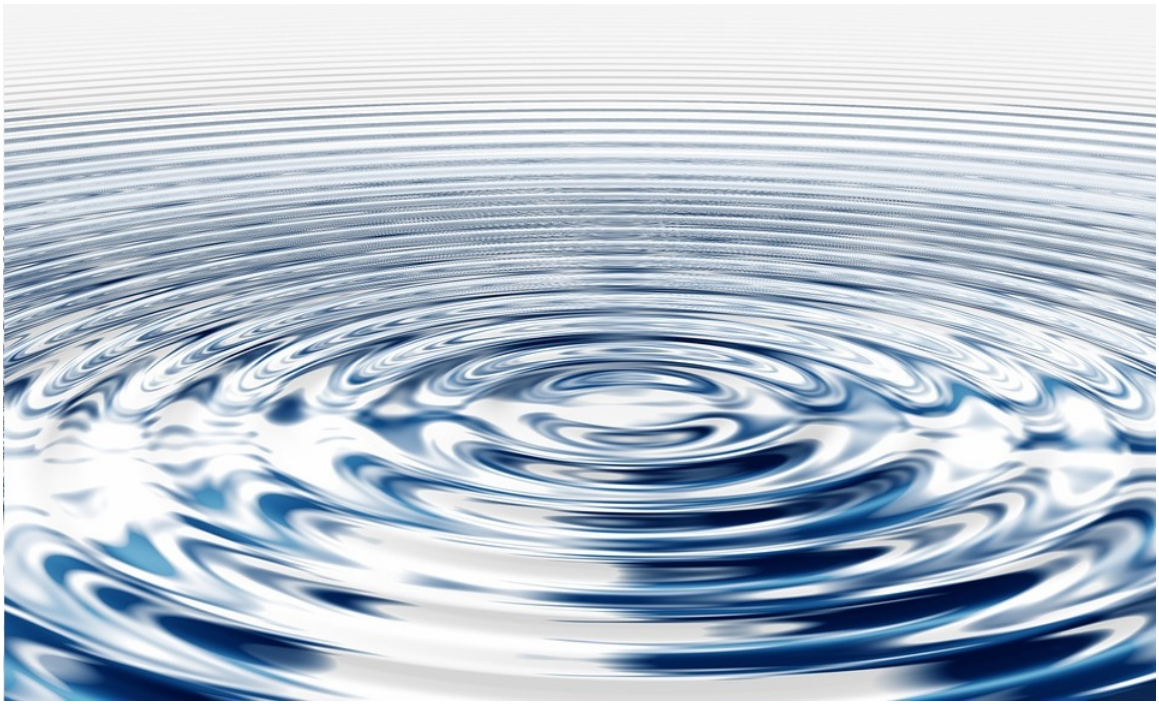


Figura 1.4: Analogía propuesta por Faraday respecto al comportamiento de las ondas electromagnéticas.

Por su parte, Maxwell continuó con su trabajo, y en 1873 publicó *"A Treatise on Electricity and Magnetism"* [6], en el cual formuló la teoría que sostiene que las ondas electromagnéticas poseen la misma naturaleza que la luz, difiriendo únicamente en la longitud de onda y, por lo tanto, cuentan con las mismas características de velocidad de propagación, polarización, reflexión y refracción. Mediante razonamiento matemático puro, afirmó que todos los fenómenos eléctricos y magnéticos podían reducirse a compresiones y movimientos en un medio que él llamó *ether*.

La polémica que generaron estas hipótesis provocó que diversos científicos comenzaran a realizar experimentos para validar la teoría de Maxwell. Incluso, la Academia de Ciencia en Berlín organizó una competencia en 1879, la cual ofrecía un premio por la investigación y experimentación que llevara a su validación.

1.3. Primeras Antenas

Diez años después aparecería el ganador del concurso de Berlín: Heinrich Hertz, quien durante un periodo de tres años de investigación y experimentación en el Instituto Politécnico de Karlsruhe, en Alemania, verificó el postulado de Maxwell de que las ondas electromagnéticas se propagan en el espacio. Para ello ideó múltiples dispositivos, entre los que destacan el dipolo Hertziano y, buscando conseguir una mayor directividad en el patrón de radiación, el cilindro reflector parabólico alimentado con un dipolo situado a lo largo del eje focal, como se muestra en la figura 1.5. Es debido al mérito de haber demostrado la existencia de las ondas electromagnéticas y por los múltiples dispositivos que desarrolló en tal camino, que se le considera el inventor de las *primeras antenas* [7].

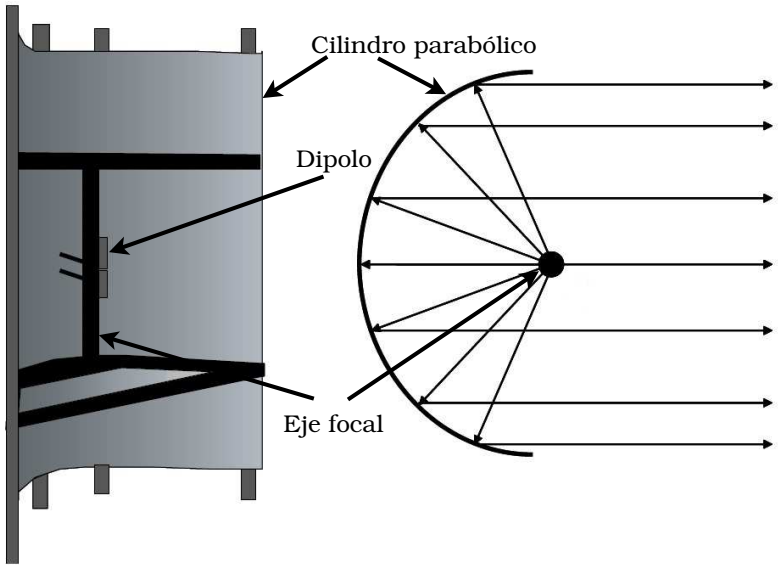


Figura 1.5: Reflector parabólico de Hertz [7].

La noticia de los descubrimientos de Hertz fue bien recibida por Sir Oliver Lodge y su amigo, el profesor G. F. FitzGerald, hombres que por su gran interés y profundo entendimiento del tema, eran probablemente los más calificados para apreciar plenamente su importancia. Aunque resulta cierto que FitzGerald no hizo un descubrimiento o invención propia, su relevancia como uno de los más respetables e influyentes físicos, le permitió jugar un papel importante al respaldar la teoría de Maxwell en su trabajo "*On the Electro-Magnetic Theory of Reflection and Refraction of light*"; así como estimular a Hertz en sus desarrollos y apoyar completamente a Lodge [3].

Por su parte, la gran contribución del profesor Lodge ha sido frecuentemente opacada por los logros aparentemente más dramáticos del joven Marconi. Mientras que los desarrollos de Lodge eran casi en su totalidad de carácter científico, Guglielmo Marconi se encontraba dedicado al desarrollo de la aplicación práctica, aspecto que atraía fuertemente la atención de la prensa y dejaba las contribuciones de Lodge virtualmente desapercibidas [5].

No obstante, cabe resaltar que en 1894, Lodge desarrolló un artefacto de extraordinaria sensibilidad capaz de detectar ondas electromagnéticas, el cual llamo *coherer*, que más tarde le permitiría demostrar, por primera vez, la posibilidad de enviar mensajes a través de las ondas Hertzianas cuando logró trasmitir señales de clave Morse. Tres años más tarde, con su amplio entendimiento sobre resonancia eléctrica, introdujo el principio de la *sintonía a determinada frecuencia* para un transmisor y su correspondiente receptor, mediante variaciones de la inductancia en circuitos resonantes [3].

Con tan profundo conocimiento de la teoría fundamental de las ondas electromagnéticas, así como su basta experiencia en la generación y detección de las mismas, podría resultar sorprendente que no procediera con el desarrollo comercial, lo cual le habría dado su lugar en la historia como el inventor de la radio. Sin embargo, considerando que se encontraba atendiendo arduas labores como profesor de física experimental en el *University Collage* en Liverpool, apenas podía esperarse que renunciara a dicha carrera para dedicarse a desarrollar comercialmente su trabajo. Más tarde habría de escribir, respecto a las demostraciones de Marconi de 1896, "fue una noticia dura para mi y para otros cuantos, pero mientras nosotros quedamos satisfechos con demostrar que podía hacerse, Marconi continuó entusiasta y persistentemente hasta convertirlo en un éxito práctico" [8].

Fue antes y durante los inicios de la radio de Marconi que se desarrollaron diversos tipos de antenas para frecuencias que incluso llegaban hasta microondas. Ejemplos de ellos son el reflector parabólico que Hertz construyó en 1888, para demostrar las propiedades ópticas de la luz, el cual trabajaba a 455 [MHz]. En 1889, basándose en la demostración de Hertz respecto a la reflexión y refracción de las ondas electromagnéticas, el profesor Lodge y su asistente el Dr. Howard, ampliaron este trabajo argumentando que también podrían concentrase y enfocarse al igual que la luz. Para demostrarlo, construyeron un par de lentes cilíndricos plano-convexos para una longitud de onda de 101 [cm].

Impresionado por las publicaciones de Lodge, el físico Chandra Bose, originario de la India, comenzó a realizar experimentos de reflexión, refracción y polarización de ondas electromagnéticas. Para 1897 reportó su trabajo, el cual implicó el uso de guías de onda, antenas tipo corneta, lentes dieléctricos, diversos polarizadores e, incluso, semiconductores a frecuencias que iban desde los 12 hasta los 60 [GHz]. Por su parte, el físico británico Ambrose Fleming demostró el uso de una guía de onda rectangular como alimentador de un lente de parafina trabajando a 15 [GHz], en el año 1900 [9].

En la década de 1890, el físico italiano Augusto Righi utilizó cilindros parabólicos para trabajar desde 1.4 hasta 11.5 [GHz]. Incluso, el mismo Marconi, estudiante de Righi, construyó un reflector parabólico para la transmisión de señales a 1.2 [GHz] en 1895 [10]. Sin embargo, la tecnología de aquél periodo limitaba la distancia a la que estas ondas podían detectarse, ya que era difícil obtener generadores de alta potencia para altas frecuencias. Esto provocó que la atención se centrara en longitudes de onda mayores, ya que estas favorecían la detección a largas distancias. Incluso, la *regla de oro* de aquélla época decía que el rango que podía alcanzarse con potencia adecuada era igual a 500 veces la longitud de onda.

Una vez alcanzado el límite de generación de potencia y optimizada la frecuencia de operación, fue la característica de la *directividad* o la capacidad de emitir y recibir ondas electromagnéticas provenientes de ciertas direcciones, la que permitió a Marconi establecer un servicio de comunicación trasatlántica confiable en 1907 [7].

De esta forma quedarían sentadas las bases para el desarrollo de la tecnología en antenas, a medida que se comprendía que representaban un elemento clave en un sistema de comunicación inalámbrico. Incluso podía apreciarse la importancia que ganaba el área de antenas en el hecho de que el primer artículo de la primera publicación de la revista "*Proceedings of the I.R.E*", del Instituto de Ingenieros de la Radio, se dedicó a dicho campo de conocimiento [11].

Desde la radio de Marconi hasta 1940, la tecnología de antenas se centró en principalmente en elementos radiadores basados en alambres: secciones largas de alambres, dipolos, hélices, rombos, abanicos, etc; y frecuencias que llegaban hasta la banda de UHF (300 a 3,000 [MHz]). No obstante, para ese año se habría de librar el obstáculo de la generación de alta potencia cuando, en Inglaterra, se desarrolló un magnetrón de 6 cavidades capaz de proveer 400 [W] de potencia promedio. Esto permitiría retomar el uso de ondas centimétricas y con ello, el desarrollo de lentes, guías de onda, cornetas y las recién aparecidas (1939) guías de onda ranuradas [12].

Por su parte, el inicio de la Segunda Guerra Mundial intensificó la necesidad de contar con sistemas de comunicaciones confiables y radares de alto desempeño, impulsando así una nueva era de desarrollo para las antenas.

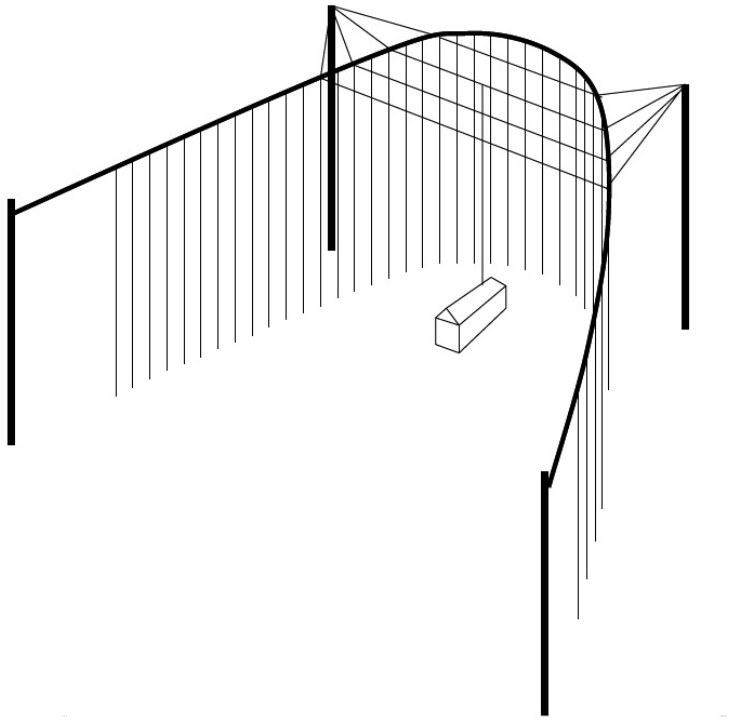


Figura 1.6: Reflector parabólico de Marconi [7].

1.4. Antenas en Arreglo

En general, un patrón de radiación específico no puede obtenerse mediante una antena de un elemento, ya que por lo regular poseen un patrón relativamente amplio y con bajos valores de directividad. Para diseñar antenas capaces de emitir la máxima radiación en una dirección determinada, es necesario incrementar el tamaño eléctrico de la antena. Esto se puede lograr incrementando las dimensiones del elemento radiador, lo cual suele presentar problemas mecánicos; o utilizando múltiples elementos para formar un *arreglo*, concepto propuesto por Sydney G. Brown y James E. Murray en 1889 [12].

Naturalmente, las primeras versiones utilizaban dipolos como elementos radiadores, ya que se trataba de los más utilizados y, por lo tanto, mejor comprendidos. De hecho, una de las primeras versiones data de 1916, cuando Marconi "con la idea utilizar longitudes de onda cortas combinadas con reflectores para ciertos propósitos militares", regresó a longitudes de onda de dos y tres metros generadas por *chispas* en aire comprimido. "Los reflectores empleados fueron hechos con segmentos de alambre sintonizados a la onda utilizada, dispuestos a lo largo de la curva descrita por una parábola cilíndrica y con un monopolo aéreo colocado en el eje focal (figura 1.6). El transmisor reflectivo fue *arreglado* de tal forma que pudiera ser girado, y los efectos a la distancia fueran estudiados en el receptor" [10].

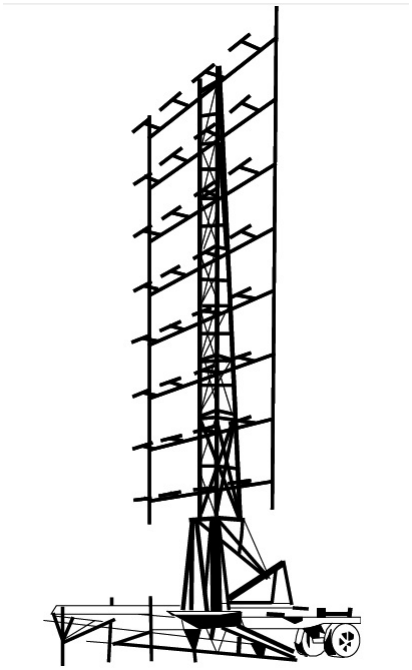


Figura 1.7: Antena del sistema de radar SCR-270 [7].

En 1939, se puso en servicio el radar SCR-270 (Figura 1.7), el cual consistía en un arreglo de ocho renglones de cuatro dipolos cada uno que trabajaba a la frecuencia de 100 [MHz]. Seis de estos radares se instalaron en Hawai con fines de detección temprana y fue uno de ellos el que detectó oportunamente el ataque de Pearl Harbor en Diciembre de 1941.

En la medida que los sistemas de radar aumentaban en tamaño, el incremento de peso provocaba una velocidad de rotación menor, lo que llevó a la investigación en exploración del haz "sin inercia". Los primeros trabajos se desarrollaron a inicios de la década de 1940 y trataban sobre reflectores esféricos, exploradores de haz libres de movimiento y arreglos con desplazadores de fase mecánicos.

La problemática que presenta una antena en arreglo radica en la alimentación de los radiadores: la distancia que existe entre los elementos y la fuente de alimentación introduce naturalmente una diferencia de fase en la onda que radian. Esto llevó al diseño de redes de alimentación calculadas para aplicar un cambio de fase adicional que permitiera incrementar la directividad. A estos sistemas se les conoce como *antenas en arreglo de fase*.

Introduciendo un sistema mecánico que permitiera modificar la longitud de las líneas de alimentación (algo análogo a la línea de transmisión variable) se podía modificar la fase radiada por cada elemento, logrando así redireccionar el haz sin mover físicamente la antena. A esta clase de elementos se les conoce como *desplazadores de fase mecánicos*. No obstante, en la década de 1950 serían reemplazados por su versión *electrónica*, la cual ofrecía un incremento en la velocidad de conmutación y, por lo tanto, en la velocidad de redireccionamiento del haz. En la década siguiente aparecieron aquellos conmutados *digitalmente*, con lo cual se incrementó aún más la flexibilidad del sistema, haciendo posible la reconfiguración electrónica de la antena mediante el uso de una computadora [7].

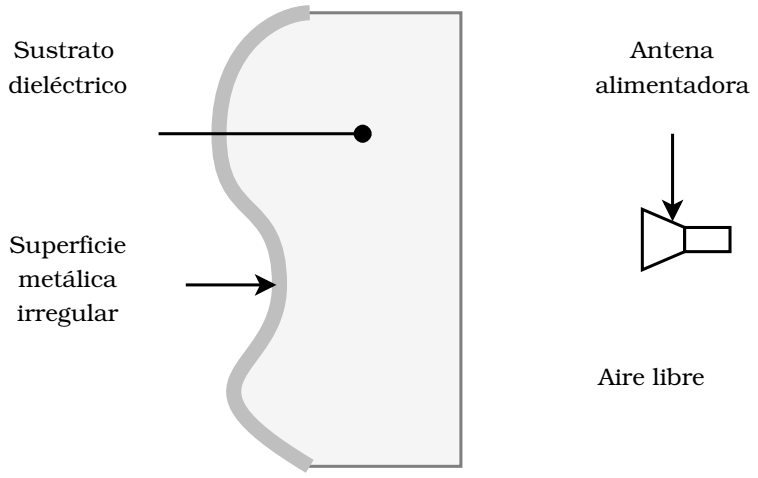


Figura 1.8: Antena reflectiva de superficie metálica irregular.

El paso siguiente habría de darse en la década de 1980 con la introducción del arreglo de fase activo, el cual buscaba eliminar la red de alimentación de los radiadores. Pese a que la función de dicha red es la de llevar la energía desde una fuente común hasta cada elemento, es indeseable debido a que introduce pérdidas y a menudo resulta complicada y voluminosa. Por su parte, en una antena en arreglo de fase activa, cada radiador es dotado con su propio transmisor, y únicamente es necesario conectarlos a la fuente de energía y a las señales digitales que controlan la dirección del haz [7].

1.5. Arreglos Reflectivos

Hasta 1963, las antenas de apertura eran catalogadas con base en dos teorías: la de arreglos y la óptica geométrica (reflectores y lentes). Sin embargo, en ese año se propuso una tercera clase de antena, llamada *arreglo reflectivo* [13], la cual combina la versatilidad del arreglo para generar una variedad de patrones de radiación junto con la ventaja que ofrecen los reflectores de evitar la incorporación de complicadas y costosas redes de alimentación, razón por la que también se les conoce como antenas *pasivas*.

La superficie reflectiva empleada en estas antenas se caracteriza por una *impedancia de superficie*, misma que puede sintetizarse para producir una variedad de patrones de radiación. Tal impedancia se puede obtener de forma exacta mediante el uso de un bloque dieléctrico terminado en una superficie metálica irregular, como se aprecia en la figura 1.8.

Por otro lado, es posible obtener una aproximación a la impedancia de superficie utilizando un *arreglo*, ya sea de dipolos o guías de onda abiertas, ambos terminados en corto circuito. En la figura 1.9 se muestra el arreglo fabricado en [13], el cual consta de 4 x 26 segmentos de guía de onda rectangular de diferente longitud.

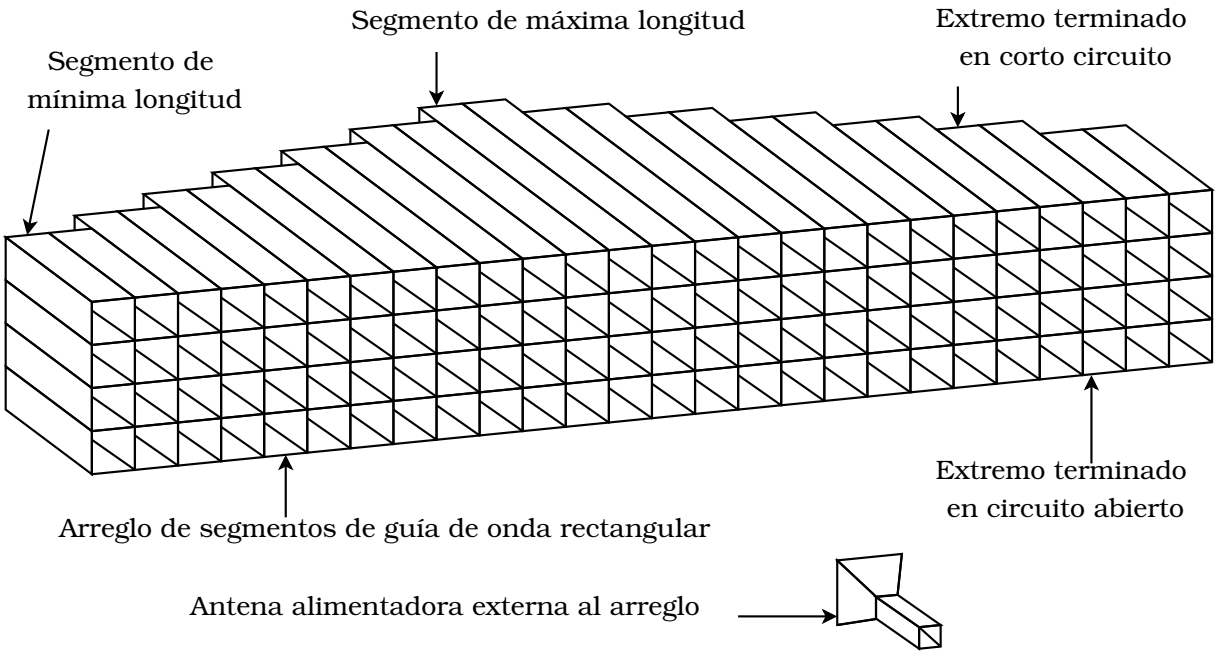


Figura 1.9: Arreglo reflectivo de 4 x 26 elementos guía de onda fabricado en [13].

Con ello, se logró demostrar la obtención de múltiples patrones de radiación que incluyen tanto haz amplio (*broad beam*) como haz angosto (*pencil beam*). De igual forma, se propuso la posibilidad de implementar el *escaneo electrónico* utilizando diodos que permitieran conmutar entre diferentes *posiciones del corto circuito* a lo largo de los segmentos de guía de onda. En este caso, se considera que cuatro posiciones por guía de onda resultaría adecuado.

No obstante, debido al uso de frecuencias de microondas relativamente bajas en aquél tiempo, los arreglos reflectivos basados en guía de onda resultaban ser antenas pesadas y voluminosas, razón por la cual no se continuó con el desarrollo de este concepto [14].

1.6. Antenas tipo Microcinta

Mientras tanto, otro avance en la tecnología de antenas sucedió con la aparición de un elemento radiador fundamentalmente nuevo: la antena tipo microcinta, la cual fue propuesta por Deschamps en 1953. Para 1955 fue archivada una patente en Fracia por Gutton y Baissinot. Sin embargo, su desarrollo se vio acelerado en la década de 1970 con la disponibilidad de sustratos de baja tangente de pérdidas, mejores modelos teóricos y el perfeccionamiento en los procesos de fotolitografía, lo que condujo a las primeras antenas prácticas fabricadas por Howell y Munson en 1972 y 1974 respectivamente [15].

Por lo regular, las antenas microcinta se les conoce como antenas tipo *parche*, ya que los elementos radiadores y las líneas de alimentación se graban sobre un sustrato dieléctrico, como se muestra en la figura 1.10. El parche de espesor t se sitúa a una distancia h de un plano de tierra. Entre ambos, se encuentra un sustrato de constante dieléctrica relativa ϵ_r. Usualmente se cumple que $t << \lambda_0$ y $0{,}003 \leq h \leq 0{,}05$, donde λ_0 es la longitud de onda en el espacio vacío [1].

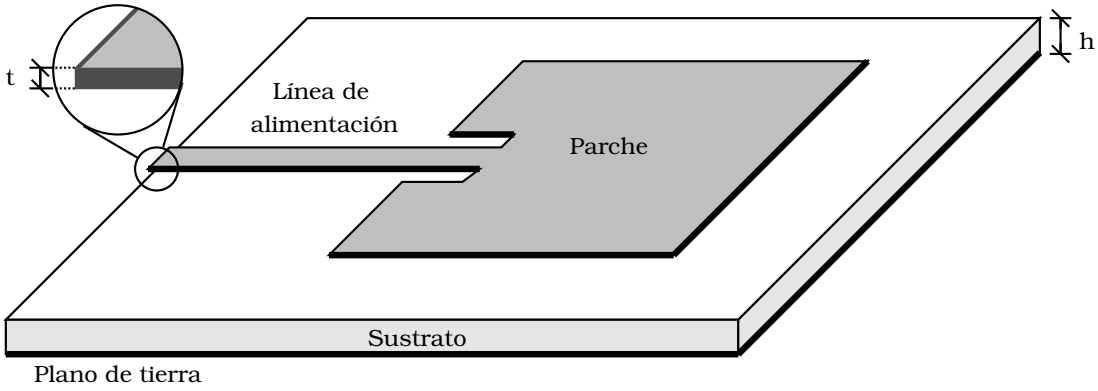


Figura 1.10: Antena tipo microcinta.

Los parches pueden ser prácticamente de cualquier forma, aunque se prefieren formas regulares como las que se muestran en la figura 1.11. Por su parte, las formas cuadrada, rectangular, dipolo y circular son las más utilizadas porque son sencillas de fabricar y analizar, además de presentar atractivas características de radiación, en espacial baja polarización cruzada [1].

Las antenas de microcinta operan mejor cuando el sustrato es *eléctricamente* grueso y con baja constante dieléctrica. Sin embargo, un sustrato delgado con una alta constante dieléc-

trica es preferible para líneas de transmisión y circuitería de microondas. Esto revela una de las paradojas asociadas con las antenas microcinta, ya que suele argumentarse que dichos elementos pueden integrarse fácilmente a otros circuitos en el mismo sustrato.

El problema radica en el hecho de que las antenas requieren campos *ligeramente enlazados* para conseguir la radiación de la energía al espacio libre, mientras que los circuitos necesitan campos *estrechamente enlazados* para prevenir radiación indeseada y acoplamiento mutuo [16]. Para conseguir un sistema funcional es necesario hacer un balance entre el buen desempeño de la antena y el del circuito.

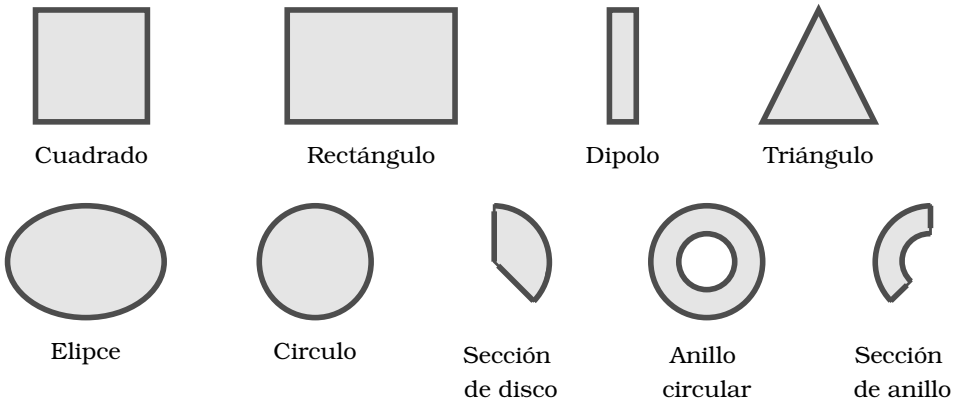


Figura 1.11: Diversas geometrías de parches [1].

La característica más atractiva de estas antenas es su bajo perfil, mismo que les permite adaptarse tanto a superficies planas como irregulares. Son de sencilla fabricación y de bajo costo utilizando la tecnología de circuitos impresos. Una vez que se han montado en una superficie rígida, son mecánicamente robustas. Dependiendo de la geometría que se utilice, pueden resultar muy versátiles en términos de frecuencia de resonancia, polarización, patrón de radiación e impedancia. Por último, añadiendo cargas entre el elemento y el plano de tierra, como diodos p-i-n o varactores, se pueden diseñar elementos *adaptables* capaces de variar la frecuencia de resonancia, impedancia, polarización y patrón de radiación [1].

Por otro lado, sus principales desventajas son su baja eficiencia, baja potencia, baja pureza en la polarización, radiación espuria debido a la alimentación de la señal y un elevado factor de calidad Q que repercute en un ancho de banda muy angosto, el cual típicamente resulta de una fracción o unidades de porcentaje [1].

No obstante, la necesidad de contar con antenas de bajo perfil ha provocado que, desde su aparición hasta la actualidad, se he hayan propuesto una gran variedad de diseños combinando distintas formas, técnicas de alimentación y geometrías. Tal desarrollo muy probablemente supere al de cualquier otro tipo de antena debido, en buena medida, a los sistemas que requieren antenas de bajo perfil, costo y peso, así como diversidad de polarización y facilidad tanto para formar arreglos, como para integrarse a circuitos. En la tabla 1.1 se muestran algunos de estos sistemas, los cuales incluyen tanto el sector militar como el civil [16].

Aplicación	Sistemas
Aviones	Radar, comunicaciones, navegación, altímetro, sistema de aterrizaje.
Misiles	Radar, camuflaje, telemetría.
Satélites	Comunicaciones, transmisión directa de TV, *sensado* remoto, radiómetros.
Barcos	Comunicaciones, radar, navegación.
Vehículos terrestres	Telefonía móvil, radio móvil.
Otros	Sistemas biomédicos, detección de intrusos, radioindentidicación (RFID).

Tabla 1.1: Algunas aplicaciones de las antenas microcinta [16].

1.6.1. Tipos de Alimentación

Existen múltiples formas de alimentar las antenas microcinta, siendo las cuatro más populares: *línea de microcinta, punta coaxial, acoplamiento por proximidad y acoplamiento por apertura* [1], mismas que se muestran en la figura 1.12.

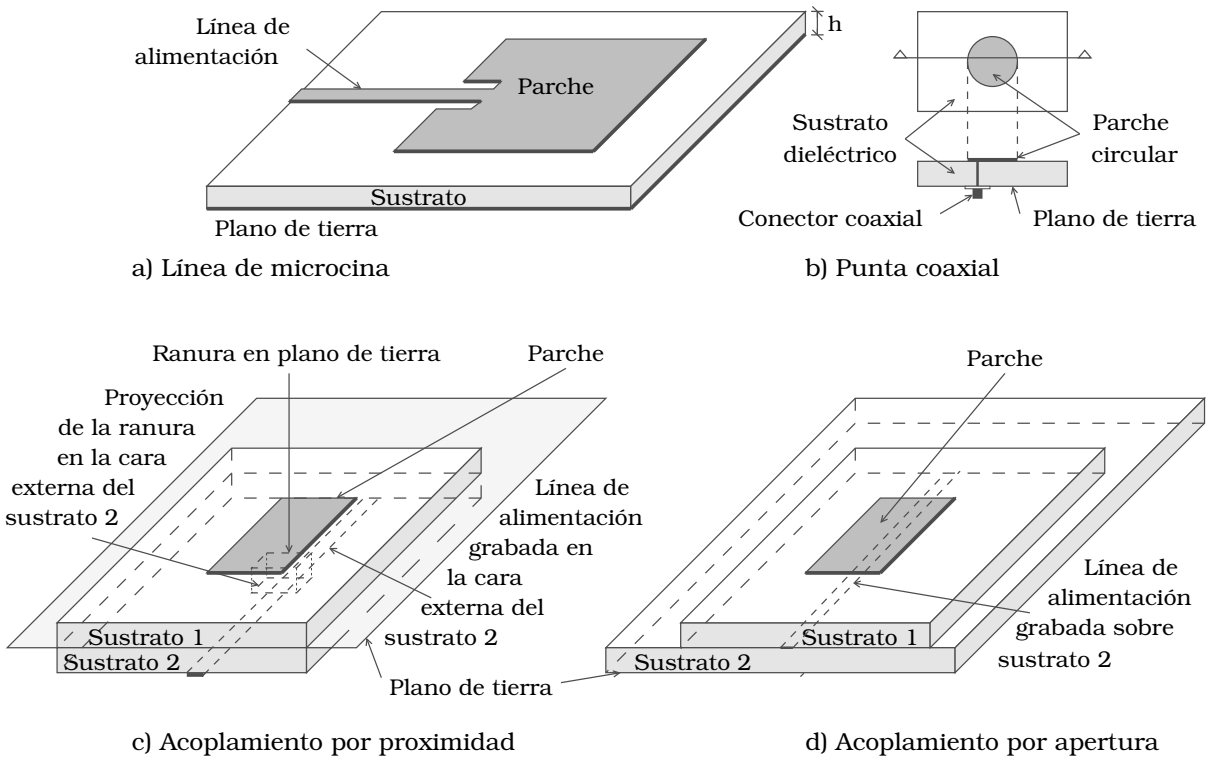


Figura 1.12: Distintos tipos de alimentación.

La alimentación por línea de microcinta es sencilla de fabricar y analizar, ya que solo debe acoplarse al parche controlando la posición de inserción. Sin embargo, a medida que el espesor del sustrato aumenta, también lo hacen las ondas superficiales y la radiación espuria, limitando el ancho de banda típicamente entre un 2 y 5 %.

La alimentación coaxial es ampliamente utilizada y se logra conectando el conductor central del cable con el radiador, mientras que el conductor externo se conecta con el plano de tierra (figura 1.12b). También resulta sencilla de fabricar y produce baja radiación espuria, aunque genera un ancho de banda angosto y resulta difícil de modelar, especialmente para sustratos con espesor $h > 0.02\lambda_0$.

Las formas anteriores poseen asimetrías inherentes que generan modos de alto orden y polarización cruzada. Para mitigar algunos de estos problemas surge la idea de alimentación por acoplamiento, la cual busca evitar el *contacto* con el radiador (figuras 1.12 c y d). El acoplamiento por apertura es el más difícil de fabricar y también produce ancho de banda angosto, aunque es más sencillo de analizar y produce radiación espuria moderada. Se forma con dos sustratos separados por un plano de tierra. En la cara externa del sustrato inferior se encuentra una línea de microcinta de alimentación, cuya energía se acopla al radiador a través de una ranura sobre el plano de tierra (figura 1.12c).

Por último, el acoplamiento por proximidad es el más sencillo de analizar y tiene la menor radiación espuria, aunque su fabricación también resulta complicada. La longitud de la línea de alimentación, así como la relación entre el ancho y largo del parche son parámetros que se utilizan para controlar el acoplamiento.

1.7. Arreglos Activos tipo Microcinta

De la misma forma que años antes se había optado por utilizar arreglos de elementos radiadores, como guías de onda, para conseguir mayor directividad, casi naturalmente surge la idea de utilizar arreglos de antenas microcinta para conseguir mayor control sobre el patrón de radiación, aumentar la ganancia y el manejo de potencia. Es así como nace el *arreglo de fase activo tipo microcinta*. Los elementos radiadores se pueden implementar con parches, o con ranuras, como se muestra en la figura 1.13.

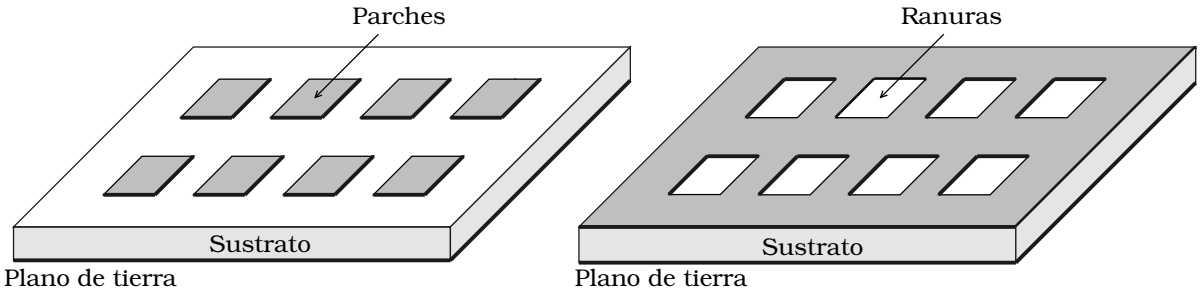


Figura 1.13: Arreglos tipo microcinta con parches y ranuras.

Al igual que el caso general de las antenas en arreglos de fase, los arreglos basados en elementos de microcinta requieren una alimentación coherente y un sistema de control que introduzca un desplazamiento, de fase o de tiempo, para obtener el redireccionamiento electró-

nico del haz [17]. En la figura 1.14 se presenta el diagrama de un módulo transmisor-receptor utilizado para compensar las pérdidas que introduce el desplazador de fase.

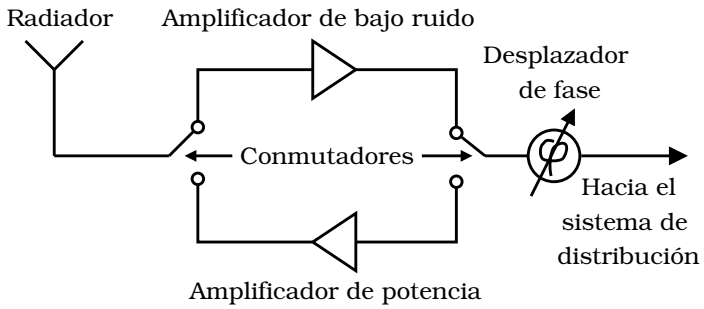


Figura 1.14: Módulo transmisor-receptor.

Este módulo se incorpora a cada elemento radiador del arreglo y suele tener un costo elevado, lo que resulta en un alto costo del sistema. Por esta razón, el desarrollo de estas antenas se han reservado casi exclusivamente para sistemas militares.

La libertad de diseño que ofrecen los arreglos de radiadores microcinta lleva a la existencia de una enorme diversidad de configuraciones. Para conseguir un buen desempeño, las técnicas de análisis se han tenido que mantener al ritmo de dicho desarrollo. Las condiciones de frontera implicadas en la mayoría de los casos impide el uso de métodos que ofrecen soluciones cerradas, por lo que se hace latente la necesidad de contar con herramientas que provean soluciones aproximadas. Desde 1970 hasta 1990 los métodos de análisis prominentes fueron el de ecuaciones integrales IE (*Integral Equations*) y la teoría geométrica de la difracción GTD (*Geometric Diffraction Theory*). Posteriormente, diferencias finitas en el dominio del tiempo FDTD (*Finite Difference Time Domain*), elemento finito FEM (*Finite Element Method*) y Método de Momentos MoM (*Method of Moments*) ganaron terreno en la solución de problemas con antenas [12].

Por otro lado, entre 1960 y 1980 se presentó un fuerte desarrollo en la arquitectura y tecnología de computadoras, el cual formó una plataforma sólida para la implementación de los métodos asintóticos que permitieron analizar y diseñar, con gran precisión, sistemas complejos de antenas que antes eran prácticamente intratables. El impacto de este desarrollo fue notable en el avance sobre la tecnología de antenas, ya que mientras en la primera mitad del siglo XX bien podía considerarse casi como una operación de “construir y probar”, a la fecha constituye un verdadero *arte* de ingeniería. Los métodos de análisis y diseño son tales, que el desempeño de la antena puede anticiparse con una notable precisión. De hecho, prácticamente todos los diseños de antenas pasan de la etapa inicial de diseño al prototipo sin necesidad de una etapa intermedia de pruebas. Si en el pasado el diseño de la antena solía considerarse como un asunto secundario en el sistema general, en la actualidad juega un papel crítico, ya que podría decirse con certeza que el éxito de muchos sistemas dependen del desempeño de la antena [12].

1.8. Arreglos Reflectivos tipo Microcinta

Así como el concepto de arreglo no tardó mucho en aplicarse a las antenas de microcinta, el concepto de arreglo reflectivo pronto se aplicaría a las antenas en arreglo de fase tipo microcinta. Con esto, análogo a la principal ventaja que presentaba el arreglo reflectivo original, se lograba una disminución del costo del sistema, ya que podía evitarse la inclusión de los módulos generadores de señal para cada elemento del arreglo. Por esta razón también se les conoce como *arreglos pasivos*.

Sin embargo, este sistema hereda la diferencia de fase debida a las diferentes trayectorias que recorre la señal de alimentación desde la fuente hasta cada elemento [14]. En la figura 1.15 se puede apreciar como la trayectoria t2 que recorre la onda es igual a la trayectoria t1 más un incremento Δt que puede llegar a representar múltiples veces la longitud de onda λ_0.

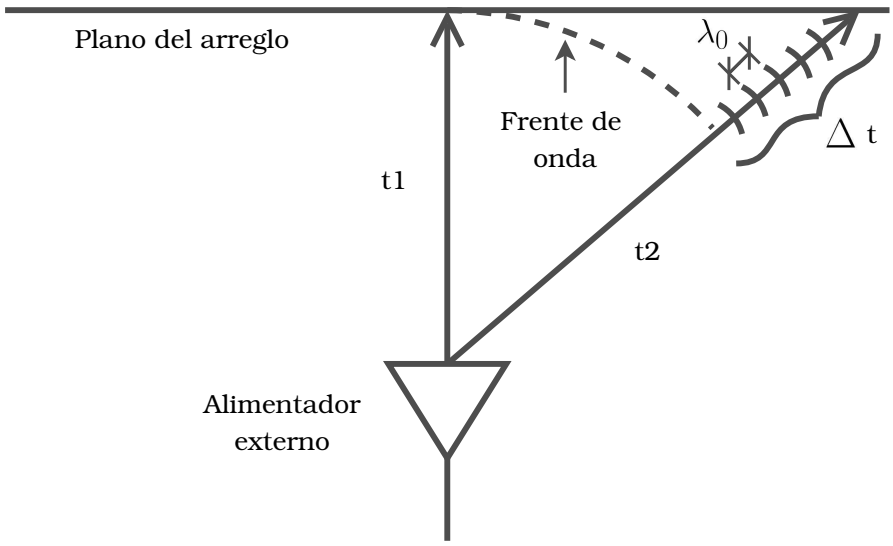


Figura 1.15: Diferencia de fase en un arreglo reflectivo [14].

Para compensar este problema, se incorpora un mecanismo de cambio de fase en los elementos del arreglo, el cual permite maximizar la radiación emitida en la dirección deseada. Variando dicho cambio de fase en forma programada, es posible conseguir el escaneo electrónico que se planteó desde la aparición del arreglo reflectivo. Para el caso de radiadores tipo microcinta, el desplazamiento de fase se puede conseguir de distintas formas, entre las que se encuentran:

- Parches idénticos con segmentos de línea de transmisión de longitud variable.
- Radiadores de tamaño variable.
- Elementos idénticos con rotación angular variable.
- Anillos resonantes cargados reactivamente.

1.9. Principio de Operación

La reconfigurabilidad de los arreglos reflectivos tiene su fundamento en el principio de cambio de fase de Fox [18]. En 1947, A. Garden Fox demostró que una onda de polarización circular (CPW) que se propaga sobre una guía de onda circular de longitud $\Delta\theta = 180°$ (o $\lambda/2$) se refleja con un cambio de fase igual a 2γ, donde γ representa la rotación angular de la guía de onda. De esta forma queda descrito un desplazador de fase en principio aplicado a guías de onda, el cual provee un cambio de fase continuo y acumulativo mediante un ajuste rotatorio, el cual involucra inercia angular que prácticamente lo descarta para aplicaciones de alta velocidad. No obstante, tal diseño posee la importante propiedad de permitir la variación de fase con una transmisión prácticamente del 100 % de la potencia incidente. Esta cualidad resulta importante en el rango de microondas donde la potencia es valiosa y no es tan sencillo recuperarla mediante amplificación.

Casi treinta años más tarde, H. Phelan [19] habría de proponer una ingeniosa forma de simular el giro mecánico por medios electrónicos: el arreglo *espirafase*, el cual está basado en elementos radiadores espirales multibrazo para polarización circular, conmutados electrónicamente mediante el uso de diodos p-i-n, como se muestra en la figura 1.16. Dicha conmutación equivale a la rotación del elemento γ grados, lo cual resulta en un cambio de fase adicional de 2γ en la CPW reflejada.

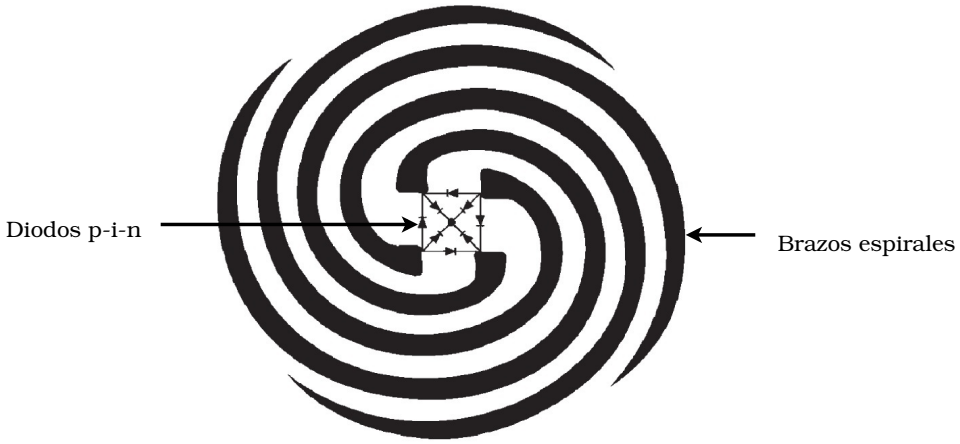


Figura 1.16: Desplazador de fase espiral de cuatro brazos [14].

En [20] se presentó un modulador de amplitud y fase para ondas milimétricas, el cual se muestra en la figura 1.17. La onda incidente de polarización vertical es transformada a una CPW a la salida del bloque polarizador. La sección de control, con base en el principio de cambio de fase de Fox, provee diferentes coeficientes de reflexión $\Gamma_\perp$ y $\Gamma_{||}$ para los dos modos de polarización lineal ortogonales entre si, cuyos vectores de densidad de campo eléctrico $\overrightarrow{E}$ son paralelos a los ejes PP' y QQ', respectivamente. Considérese la CPW incidente como:

$$E_i = E_0(\overrightarrow{e_x} + j\overrightarrow{e_y})e^{-j\beta z} \tag{1.1}$$

donde E_0 es la amplitud de la onda incidente, $\overrightarrow{e_x}$ y $\overrightarrow{e_y}$ los vectores unitarios en las direcciones $\overrightarrow{x}$ y $\overrightarrow{y}$ respectivamente, y $j = \sqrt{-1}$. En este caso, la onda reflejada puede expresarse como la suma de las CPWs:

$$\overrightarrow{E_r} = 0{,}5E_0e^{2j\gamma}(\Gamma_{||} - \Gamma_{\perp})(\overrightarrow{e_x} - j\overrightarrow{e_y})e^{j\beta z} + 0{,}5E_0(\Gamma_{||} + \Gamma_{\perp})(\overrightarrow{e_x} + j\overrightarrow{e_y})e^{j\beta z} \tag{1.2}$$

donde γ es el ángulo entre el eje $\overrightarrow{x}$ y el eje PP'.

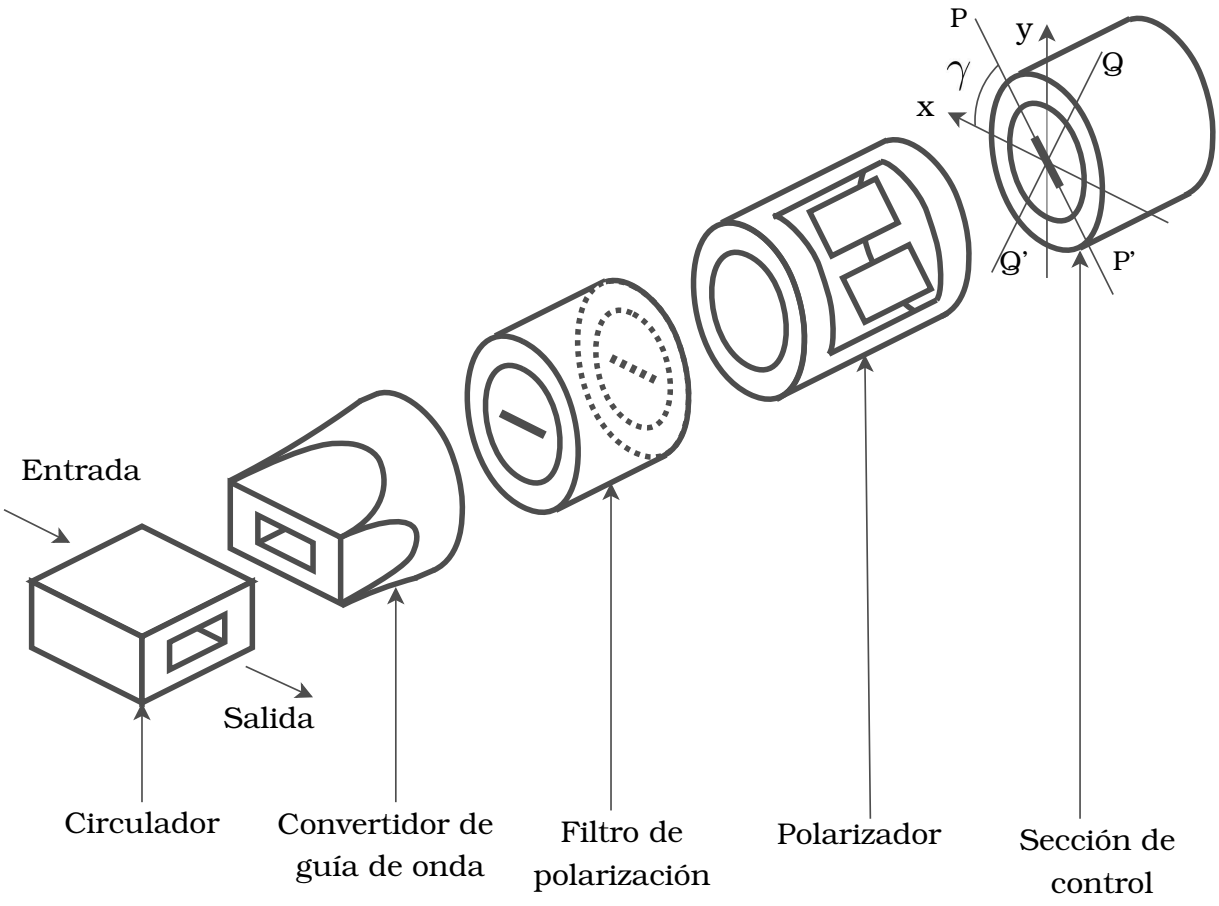


Figura 1.17: Modulador de amplitud y fase para ondas milimétricas [20].

La primera componente de la suma en la ecuación 1.2 es una CPW con la misma rotación del vector $\overrightarrow{E_i}$, la cual es la onda incidente. Esta componente puede controlarse modificando el ángulo γ, razón por la que se le conoce como *onda controlada*. La segunda componente de la suma es una CPW con dirección de rotación opuesta al vector $\overrightarrow{E_i}$. La fase de esta componte no se modifica con el cambio de γ, por lo que se le llama *onda no controlada*. La amplitud de la onda controlada encuentra su máximo cuando:

$$\Gamma_{\perp} = -\Gamma_{||} \tag{1.3}$$

lo que implica que la amplitud de la onda no controlada es igual a cero y el modulador presenta la mínima pérdida posible, determinada únicamente por la calidad de los elementos conmutadores.

La forma más simple de implementar la sección de control es mediante un dipolo de $\lambda/2$ terminado en corto circuito. Dicho elemento se graba en un diafragma y se instala en la sección transversal de una guía de onda circular de longitud $\lambda/4$ terminada en una pantalla metálica que representa el corto circuito.

Es posible utilizar minimotores para conseguir la rotación mecánica del desplazador y modificar así la fase de la onda reflejada. Sin embargo, la técnica resulta costosa, ya que contempla el uso de un motor por elemento radiador. En [21] se utilizó un minimotor a pasos ARSAPE 0820 y el tiempo de conmutación medido fue de aproximadamente 40 [ms], mismo que resulta insuficiente para la mayoría de las aplicaciones.

Una alternativa es el uso de la *simulación electrónica* del giro mecánico mediante la técnica espirafase, la cual mejora considerablemente las desventajas del análogo mecánico: menor costo y una mejora en el tiempo de conmutación de aproximadamente dos órdenes (decenas de nanosegundos). En la figura 1.18 se muestra el diafragma de control propuesto en [20] para implementar dicha técnica. El diseño consta de un anillo ranurado de diámetro promedio $\lambda/4$, que a su vez está cargado con segmentos de línea de transmisión (*stubs*) conectados en serie con el anillo. Para lograr la simulación electrónica del giro se conectan diodos p-i-n en paralelo con los stubs. De esta forma, idealmente pueden obtenerse tiempos de conmutación de aproximadamente 35 [ns].

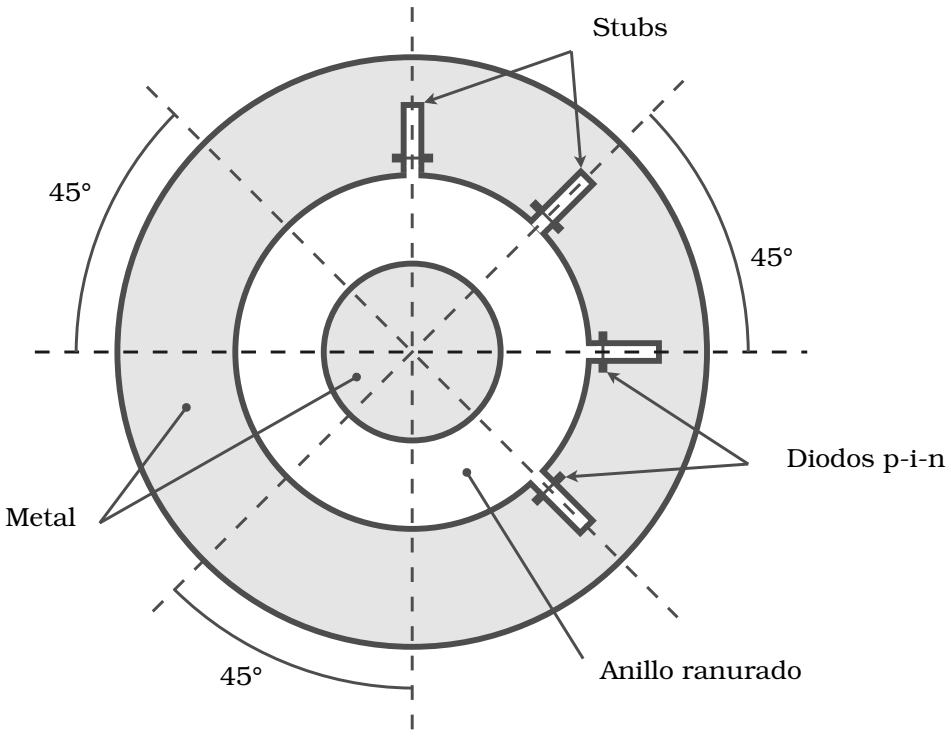


Figura 1.18: Diafragma de control de dos bits [20].

1.10. Estado del Arte

Aplicando el principio de espirafase a los anillos resonantes tipo microcinta, se propusieron en [22] desplazadores de fase de uno y dos bits. En la figura 1.19a se muestra el caso de un bit, en el cual un anillo ranurado de ancho w se forma mediante un círculo metálico interno y uno externo. Dicho anillo está cargado con dos stubs radiales, que a su vez cuentan con un diodo p-i-n conectado en paralelo. Finalmente, una pantalla metálica se coloca a una distancia $\lambda/4$ del diafragma. Añadiendo dos stubs con sus respectivos diodos, se forma un desplazador de dos bits (figura 1.19b). En tal diseño fue removido el círculo metálico central, ya que la frecuencia de resonancia serie no se ve afectada; no obstante, la distancia de la pantalla metálica debe ser mayor a $\lambda/4$ para compensar la reactancia inductiva del anillo metálico externo.

En ambos casos, para cualquier instante del tiempo, un diodo se encuentra en estado de alta impedancia, mientras que el resto se encuentran en baja impedancia, habilitándose así un único stub a la vez. Conmutando el diodo a permanecer en estado abierto se puede variar el cambio de fase sobre la onda reflejada, logrando así la simulación electrónica del giro.

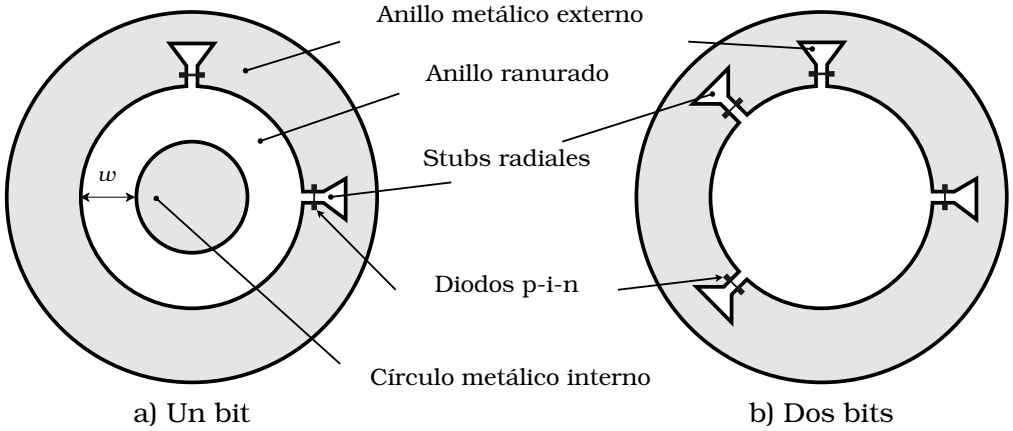


Figura 1.19: Desplazadores de fase tipo anillo ranurado [22].

En [23] se publicó el análisis de un desplazador de fase de dos bits que demuestra pérdidas de inserción menores a 0.5 [dB] en los cuatros estados, para la banda de frecuencia de 9.75 a 11.5 [GHz]. Conectando el desplazador a una sección de guía de onda ranurada WR-90, se determinó un tiempo de conmutación menor a 150 [ns]. Para disminuir las pérdidas de inserción debidas a los diodos, se propuso un circuito de polarización que utiliza resina de BenzoCycloButene (BCB), evitando así el proceso mecánico que implica el uso de sustratos metálicos gruesos propuesto en [24], con lo que se logra reducir considerablemente el costo. En la figura 1.20 se muestra el diafragma fabricado.

En [25] se investigó la posibilidad de utilizar desplazadores de fase *abiertos*, es decir, se propuso colocar un arreglo de diafragmas de control en un plano sin utilizar las secciones de guía de onda circular. Como resultado, mediante el uso del principio espirafase, pueden obtenerse arreglos altamente integrados y de bajo costo.

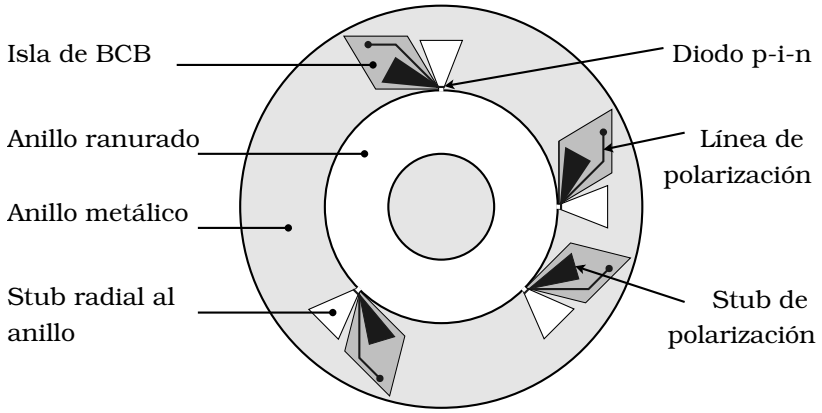


Figura 1.20: Desplazador de dos bits con circuito de polarización [23].

En [26] se presentó el diseño de un arreglo reflectivo como el que se muestra en la figura 1.21. La habilitación de distintos stubs para cada elemento del arreglo rompe la periodicidad del arreglo, misma que puede recuperarse definiendo un conjunto de elementos que se repiten dentro del arreglo infinito. A este bloque de radiadores se les llama *celda grande*. Las simulaciones demostraron ángulos de elevación de hasta 50 ° con coeficientes de conversión menores a -2 [dB] en la banda de frecuencia de 8.3 a 10.2 [GHz].

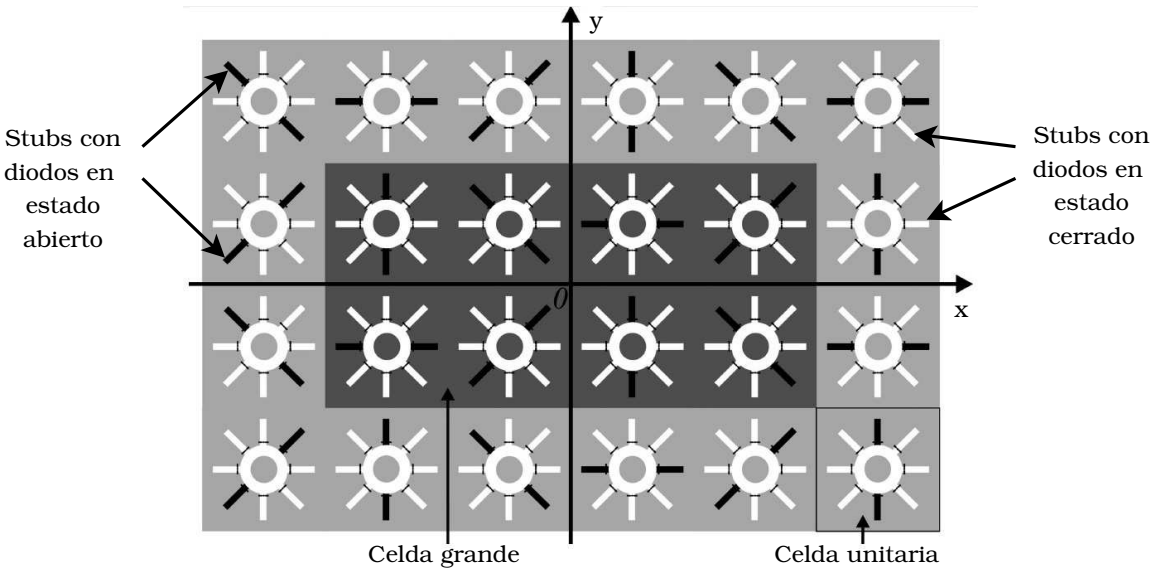


Figura 1.21: Arreglo, celda grande y elemento unitario [26].

Otra forma de generar desplazadores de fase reconfigurables electrónicamente es mediante el uso de anillos metálicos tipo parche cargados capacitivamente. En [27] se realizó el análisis de un arreglo reflectivo basado en estos elementos radiadores y se demostró la capacidad de redirigir el haz con ángulos de elevación de hasta 50°, con pérdidas de conversión menores a 2.7 [dB] para la banda de frecuencia de 8 a 18 [GHz]. En la figura 1.22 se muestra una sección del arreglo sobre la que incide y se refleja una onda de polarización circular.

A diferencia del anillo cargado con stubs, aquí se graban ranuras sobre un anillo metálico tipo parche y las secciones metálicas resultantes se conectan mediante diodos p-i-n. En la parte derecha de la figura 1.22 se muestran las cuatro posibles configuraciones que puede tomar este desplazador de dos bits. Para cualquier momento de tiempo, dos diodos permanecen en estado de no conducción (diodos en color blanco) mientras que el resto permanecen en estado de conducción (diodos en color negro).

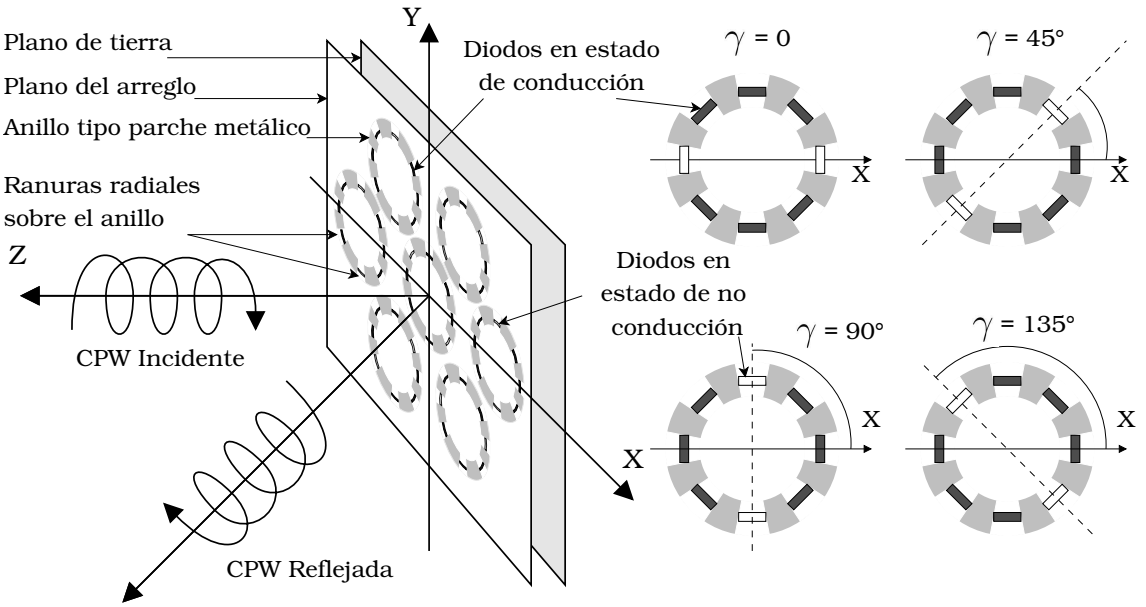


Figura 1.22: Arreglo de parches metálicos cargados capacitivamente.

En [28] se publicó el diseño de un arreglo de 367 anillos cargados con capacitores monolíticamente integrados, como se muestra en la figura 1.23. Se fabricaron cuatro arreglos fijos para ángulos de elevación θ_h de 0°, 20°, 40° y 60° y ángulo de azimut φ_h = 0°, demostrando eficiencia de apertura de 0.60, 0.51, 0.42 y 0.21 junto con niveles de polarización cruzada de -25.8, -24.0, -17.8 y -10.9 [dB] para 36.5 [GHz], respectivamente. La figura 1.23 muestra la configuración para reflejar el haz a la dirección normal (θ_h = 0°, φ_h = 0°).

Tal arreglo puede convertirse en una antena reconfigurable si se conectan conmutadores en serie con las cargas capacitivas para implementar la simulación electrónica del giro, como se muestra en la figura 1.24. Para cualquier instante de tiempo, todos los interruptores se encuentran en estado abierto excepto dos ubicados a 180 grados uno del otro, los cuales se encuentran en estado cerrado. Como resultado, el efecto de dos capacitores opuestos resulta dominante en cualquier momento.

En esta publicación se sugiere el uso de interruptores basados en la tecnología MEMS, (*MicroElectroMechanical Systems*), ya que presentan características como 1 [fF] de capacitancia en estado de no conducción y 2.2 [Ohm] de resistencia en estado de conducción, características que tendrían efectos despreciables en el desempeño del arreglo.

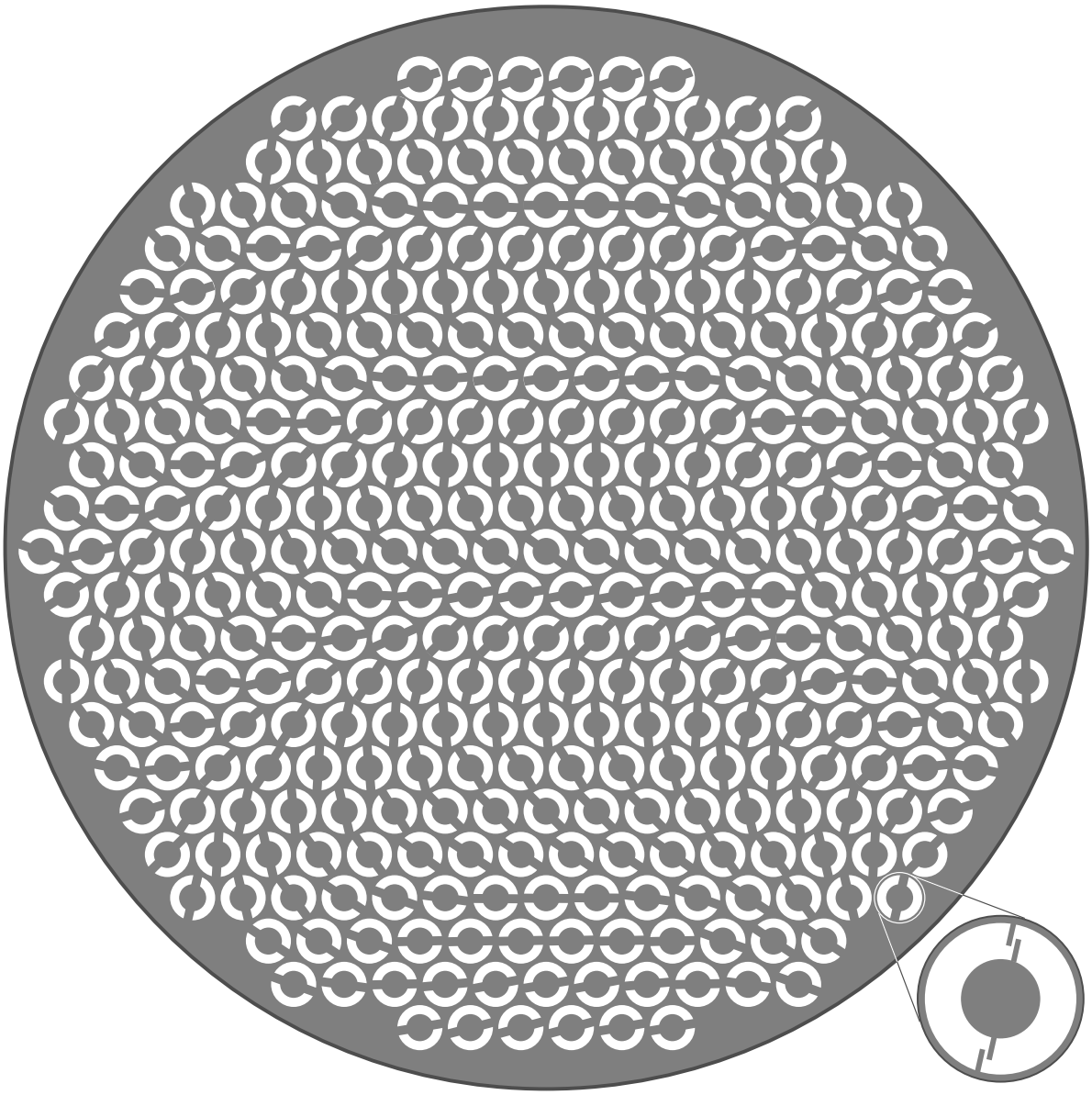


Figura 1.23: Arreglo que refleja a la dirección normal, formado por 367 anillos cargados con capacitores monolíticamente integrados.

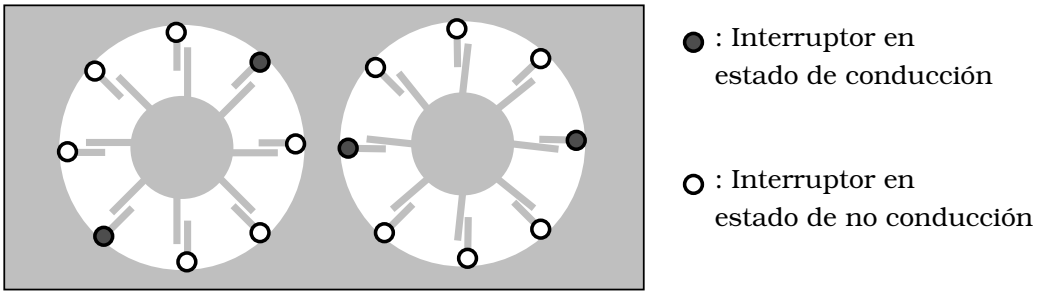


Figura 1.24: Desplazadores de fase reconfigurables electrónicamente [28].

1.10.1. Arreglos en Sistemas de Radar

Radares basados en arreglos de radiadores son especialmente útiles en sistemas que requieren de un redireccionamiento ágil del haz y contar con la posibilidad de ejecutar múltiples funciones simultáneamente. Algunos ejemplos de estas aplicaciones son la vigilancia satelital, sistemas de defensa aérea, defensa contra objetivos balísticos, radares multifunción para naves aéreas [29], radares anticolisión, comunicaciones y sistemas biomédicos [30].

Por su parte, una computadora es una parte crucial en este tipo de sistemas, ya que permite explotar la inherente flexibilidad de un arreglo mediante el control eficiente del radar y la planificación de sus operaciones. No obstante, el costo de obtener tal funcionalidad no es insignificante y es uno de los factores que eleva el costo del sistema. Algunas de las funciones que debe cumplir el computador son [29]:

- Proveer los comandos de direccionamiento para cada elemento radiador.
- Realizar el manejo de señales determinando la forma de onda, el número de observaciones, la tasa de transferencia de información, la potencia y la frecuencia a utilizar.
- Ejecutar el procesamiento digital de señales acorde con el modo de operación.
- Desplegar la información procesada al usuario.
- Monitorear de desempeño, la detección de averías, el almacenamiento de información y la generación de simulaciones.
- Realizar la gestión del radar mediante la asignación de prioridades a distintas tareas para lograr un buen compromiso entre las acciones a ejecutar y las capacidades del sistema.
- Proveer un medio para que el usuario pueda interactuar físicamente con el radar.

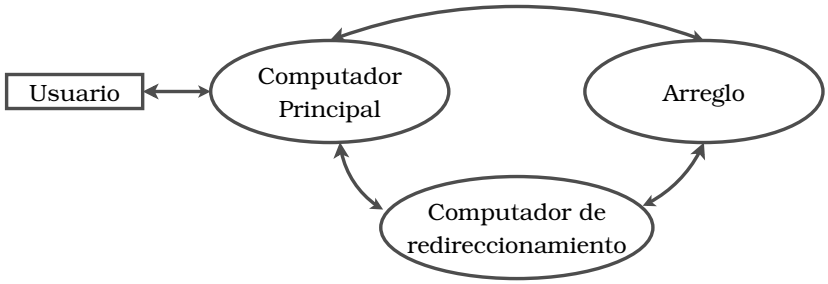


Figura 1.25: Sistemas de cómputo en un radar.

Aún cuando es posible utilizar una computadora de propósito general para efectuar todas los cálculos y acciones de control necesarios, es deseable utilizar una computadora separada de propósito específico para el direccionamiento del haz (figura 1.25). Ésta puede incorporarse al arreglo para minimizar el problema de comunicar un gran número de comandos hacia los elementos radiadores. De esta forma, el computador principal provee al computador dedicado los ángulos de elevación y azimut, y este último los traduce a las señales de control para cada radiador. Para el caso de arreglos con elementos dependientes de la frecuencia, este parámetro debe ser alimentado al computador de direccionamiento junto con los ángulos [29].

Una de las figuras de mérito de los radares es la frecuencia de repetición de pulso PRF (*Pulse Repetition Frecuency*), la cual determina la tasa a la que se repite el envío de un pulso [31]. El inverso de este parámetro es el periodo PRT (*Pulse Repetition Time*), el cual contempla el tiempo que toma el envío de un pulso más el tiempo de espera para la detección del eco como se muestra en la figura 1.26. Definiendo el tiempo de respuesta del computador de direccionamiento como Tiempo de Cálculo del Haz (TCH), entonces dicho parámetro debe ser menor al PRT para garantizar que el arreglo se encuentre correctamente configurado antes de enviar el pulso en la nueva dirección.

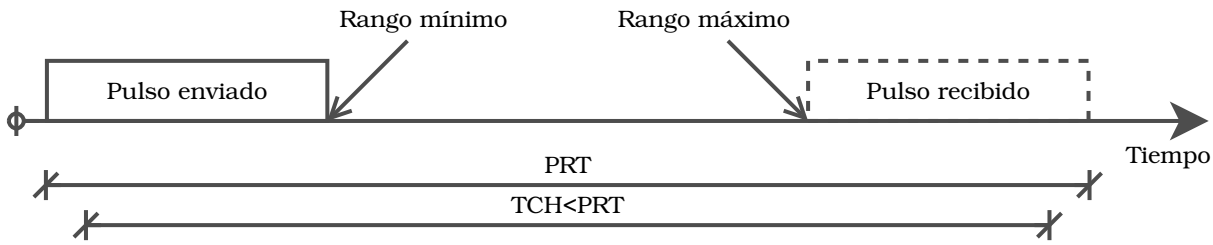


Figura 1.26: Diagrama típico del Tiempo de Repetición de Pulso.

En la tabla 1.2 se presenta el parámetro PRT de algunos radares fabricados comercialmente, así como su aplicación.

Nombre	Aplicación	PRT
RRP-117 *Seek Igloo*	Militar	4 y 0.9 [ms]
Primus 800	Clima	5.5, 2.7 y 2.1 [ms]
Variant	Militar	333 y 111 [μs]

Tabla 1.2: Tiempo de Repetición de Pulso de algunos radares comerciales.

Sin embargo, la capacidad de ejecutar múltiples funciones es una propiedad deseable en un radar. En la tabla 1.3 se presentan las funciones correspondientes al radar AN/FPS-85 [32] y la duración del los pulsos utilizados en cada una. En las funciones que utilizan *ráfagas*, se deducen PRTs de 22.125 [ms], 5.025 [ms] y 30 [μs].

Función	Duración del pulso/ráfaga
Búsqueda de largo alcance	250 [μs]
Búsqueda de corto alcance	10 [μs]
Seguimiento de largo alcance	250 [μs]
Seguimiento de corto alcance	1 [μs]
Seguimiento de rango extendido	40 pulsos de 125 [μs] / ráfaga de 1 [s]
Seguimiento Doppler intermedio	40 pulsos de 25 [μs] / ráfaga de 0.2 [s]
Seguimiento de precisión	40 pulsos de 5 [μs] / ráfaga de 1.2 [ms]
Calibración del arreglo	60 [μs]

Tabla 1.3: Funciones del radar de vigilancia satelital AN/FPS-85 [32].

Actualmente, una de las funciones más demandantes en sistemas de radar es la de multirastreo y búsqueda simultáneos, en la que los filtros de rastreo proveen mediciones de un objetivo mientras el radar puede continuar con la búsqueda de nuevos objetivos. De esta forma es posible seguir múltiples objetivos presentes en el volumen de escaneo. Para una precisión aceptable en condiciones en las que los objetivos se encuentran realizando maniobras, es necesario que la información de cada objetivo se actualice al menos a 2.5 [Hz], lo que equivale a 400 [ms] para cada objetivo [33].

Por su parte, en publicaciones referentes a tecnología en microondas se han reportado avances interesantes respecto al TCH. Por ejemplo, en [34] se presentó un arreglo reflectivo con elementos basados en cristales líquidos, el cual tiene la capacidad de reconfigurar el haz en 2 [s], lo que lo hace apto para aplicaciones que incluyen observación terrestre y climática.

En [35] se presentó un radar de apertura sintética SAR (*Synthetic Aperture Radar*) basado en un arreglo reflectivo diseñado para una plataforma de formación de imágenes, el cual utiliza la técnica de formación de haz digital DBF (*Digital Beam Forming*). El sistema trabaja a una frecuencia central de 35.75 [GHz] y utiliza dos frecuencias PRF: 5.15 y 5.3 [KHz], lo que implica un PRT de 194 y 189 [μs] respectivamente.

Partiendo de un tiempo TCH menor a 200 [μs] como parámetro de diseño para sistemas de radar dedicados a la formación de imágenes, en [36] se presentó un arreglo reflectivo de 160 x 160 elementos radiadores. Para que la fabricación de tal arreglo resultara factible, se diseño un elemento radiador de estructura simple que fuera fácilmente controlable, por lo cual se utilizó un parche de microcinta para implementar un desplazador de fase digital de un bit utilizando el diodo p-i-n como elemento conmutador. Utilizando únicamente 160 líneas de control, obtenidas mediante un FPGA, y un registro de corrimiento, los diodos se controlaron secuencialmente renglón por renglón. El tiempo de respuesta medido fue de tan sólo 28 [μs].

Cabe destacar un último trabajo que se publicó a tan sólo cinco meses antes de terminar el presente trabajo de tesis [37]. Se trata de un arreglo reflectivo reconfigurable de un bit formado por 10 x 10 elementos. Un microcontrolador se utiliza como sistema huésped, el cual se encarga de distribuir las señales de control a los conmutadores. Considerando un ciclo de operación de 62.5 [ns] y 40 operaciones para determinar el estado de un radiador, se obtiene un tiempo de conmutación de 2.5 [μs]. El sistema calcula de forma paralela 20 módulos, que a su vez calculan secuencialmente la información de cinco radiadores. Esto implica un tiempo de respuesta total de aproximadamente 12.5 [μs].

Con esto se pretende demostrar la importancia que tiene la implementación de un computador dedicado para liberar al computador principal de dicha labor, minimizando así el tiempo de cálculo involucrado, parámetro que a su vez resulta de gran importancia en prácticamente cualquier aplicación. Por tal razón, el presente trabajo propone el diseño de un procesador dedicado cuyo principal objetivo es obtener un tiempo de respuesta que mejore o se encuentre dentro del reportado en el estado del arte.

Capítulo 2

Fundamentos de la Arquitectura de un Procesador

El procesador es el elemento central de casi cualquier sistema digital, incluidos los complejos sistemas de cómputo modernos. De hecho, en su forma más básica, una computadora se forma con un microprocesador y una colección de elementos lógicos que lo habilitan para comunicarse con el mundo exterior, tanto para obtener datos, como para desplegar los resultados de las operaciones ejecutadas. La combinación de instrucciones (*software*) que se ejecuta en un microprocesador orquesta el comportamiento del conjunto de componentes (*hardware*) que se ha diseñado para dar solución a un problema [38].

Inicialmente, las computadoras fueron inventadas con el propósito de *computar*, término que está definido por la Real Academia del Español como cuenta o cálculo. Originalmente el término *computador* hacía referencia a una persona que resolvía cálculos matemáticos, y no fue sino hasta 1945 que el término se utilizó para referirse a las máquinas que realizaban dicha labor. Actualmente las computadoras son capaces de realizar una plétora de tareas reduciéndolas esencialmente a tareas lógico-aritméticas y de almacenamiento [39].

En este capítulo se realiza una breve revisión sobre el surgimiento de las computadoras, su transición de analógicas a digitales, el desarrollo de las máquinas que llevarían al concepto de almacenamiento del programa y, con ello, el surgimiento de la arquitectura secuencial que, tras algunas modificaciones, habría de convertirse en el fundamento de prácticamente todas las computadoras por venir. Se presentan los principales atributos de un procesador, así como los niveles de aplicación del paralelismo y la emulación del mismo mediante la segmentación encausada. Posteriormente se presenta la lógica programable como una tecnología que ofrece una amplia gama de dispositivos para implementar diseños basados en la descripción de hardware. Finalmente, se presenta la abstracción como una herramienta para manejar la complejidad que implica la descripción de un procesador, así como la metodología de diseño más apropiada.

2.1. Primeras computadoras

Una *calculadora mecánica* era un dispositivo capaz de realizar operaciones aritméticas básicas. La primera fue inventada por Blaise Pascal en 1642. La *Pascalina* podía sumar y restar dos números; aplicando dichas operaciones consecutivamente podía multiplicar y dividir. Treinta años más tarde, el matemático Gottfired Leibniz inventó la calculadora mecánica "*Step Reckoner*", la primera capaz de ejecutar las cuatro operaciones aritméticas: suma, resta, multiplicación y división [40].

El *Analizador Diferencial de Bush* es considerado como la primera computadora mecánica *analógica* de propósito general, desarrollada por Vannevar Bush en el MIT (*Massachusetts Institute of Technology*) a finales de la década de 1920. La representación de la información en una computadora analógica es compacta pero vulnerable al ruido. Un sólo capacitor puede almacenar una variable continua dentro de un computador, mientras que en una computadora *digital* son necesarios múltiples dispositivos capaces de mantener estados discretos. No obstante, las computadoras analógicas fueron reemplazadas por las digitales tras la Segunda Guerra Mundial [40].

Por su parte, una computadora digital es una máquina electrónica capaz de realizar procesar, almacenar y recuperar información. El término digital implica que la información dentro de la computadora está representada por variables que toman un número limitado de valores discretos. Dichos valores son procesados internamente por componentes que pueden mantener el mismo número de estados. La primera computadora electrónica, desarrollada a finales de la década de 1940, se utilizaba principalmente para cálculos numéricos. En este caso, los elementos discretos eran dígitos, que es precisamente donde surge el término *digital* [41].

Las primeras computadoras *digitales* utilizaban tubos al vacío para almacenar la información binaria, mismos que eran voluminosos y generaban una gran cantidad de calor. Sin embargo, en 1956 Schockley, Baarden y Brattan ganaron el Premio Nobel de física por la invención del *trasistor*, el cual habría de reemplazar los tubos al vacío para finales de la década, ya que eran pequeños y consumían poca potencia, lo que permitía obtener máquinas más rápidas, confiables y de menor tamaño [40].

A pesar de que actualmente se han desarrollado dispositivos electrónicos capaces de asumir entre seis y diez estados [42], prácticamente todos los sistemas de cómputo en la actualidad representan la información mediante la presencia o ausencia de carga eléctrica (dos estados) en transistores, lo que implica que su funcionamiento se basa en el sistema binario, el cual consta de dos dígitos: 0 y 1, números a los que se les conoce como *bits* (**b**inary dig**its**). Formando grupos de bits se pueden aplicar técnicas de codificación para representar información en forma de números decimales y letras. Ordenando adecuadamente estos grupos elementales se puede desarrollar un set de instrucciones que permita ejecutar prácticamente cualquier tarea de cálculo [41].

2.2. Representación de la Información

La información binaria en las computadoras digitales se almacena en memoria o en los registros del procesador. Un código binario es un grupo de *n* bits que representa hasta 2^n distintas combinaciones para codificar elementos [43].

Para representar los números enteros positivos, la convención más común es la notación *no signada*, en la cual el valor decimal de una cadena de n bits se obtiene como:

$$b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \cdots + b_2 2^2 + b_1 2^1 + b_0 2^0 \tag{2.1}$$

donde el punto binario (análogo al punto decimal) se asume a la derecha del bit menos significativo *lsb* (*least significant bit*).

Para representar enteros negativos, es necesario contar con una notación para dichos valores. En la aritmética ordinaria, un número negativo se indica mediante un signo negativo (-) y el número positivo mediante el signo positivo (+). Pero debido a la limitación del hardware con que cuentan, las computadoras deben representar todo mediante bits, incluido el signo del número. Es por esto que acostumbra representarse el signo mediante el bit más significativo *msb* (*most significant bit*), el cual se ubica en el extremo izquierdo. La convención dice que el cero se utiliza para representar el signo positivo, y el uno para el negativo [43].

La forma más natural de implementar el signo es de forma directa, es decir, representar el signo con el msb y el resto del de los bits codifican la magnitud. A esta convención se le conoce como *signo-magnitud*. Pese a que esta es la notación que se utiliza en la aritmética decimal, resulta inapropiada para utilizarse en aritmética computacional.

Una notación más adecuada es la que utiliza *complementos*. Las dos más comunes son complemento a *uno* y complemento a *dos*. En la primera, el número se obtiene invirtiendo el valor de cada bit, pero impone dificultades en los cálculos ya que el cero tiene dos representaciones: +0 y -0. Por su parte, el complemento a dos se obtiene sumando la unidad al complemento a uno. Esta notación es la más utilizada ya que la resta se implementa mediante la suma directa del primer operando con el complemento a dos del segundo operando [44].

El uso de números enteros es suficiente para diversas aplicaciones como son el conteo de procesos y direccionamiento de memoria. Sin embargo, el mundo es inherentemente análogo y es mejor modelado mediante números *reales* [38]. Para representarlos, es necesario manipular la posición del punto binario.

Al igual que el caso del signo, representar el punto físicamente implicaría utilizar un bit. Sin embargo, resulta más sencillo omitir esa representación y simplemente *asumir* su posición, para lo cual existen dos convenciones: *punto fijo* y *punto flotante*. En la primera se asume que el punto binario se encuentra *fijo* en una posición, mientras que en la segunda el punto se asume mediante la representación de un exponente y una mantisa [43].

La notación de punto fijo tiene implícito el punto binario entre los bits dedicados a la parte entera y los dedicados a la parte fraccionaria [51]. De esta forma, un número real no signado que utiliza n bits para la parte entera y m bits para la parte fraccionaria se representa de la siguiente forma:

$$b_{n-1}2^{n-1} + \cdots + b_2 2^2 + b_1 2^1 + b_0 2^0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-m}2^{-m} \tag{2.2}$$

Dado que el manejo del punto binario es únicamente una convención, la representación de reales negativos se maneja de la misma forma que los enteros, tanto para punto fijo como punto flotante, siendo la del complemento a dos la más utilizada.

2.3. Surgimiento del Almacenamiento de Programa

Cuando se completó en 1945, el computador e integrador numérico electrónico ENIAC (*Electronical Numerical Integrator And Computer*) operaba mucho más rápido que cualquier máquina anterior. Pero aún cuando era capaz de resolver problemas matemáticos complejos en tan sólo segundos, podía tomar días reconfigurar la máquina, ya que era virtualmente necesario *reconstruir* la máquina desconectando y reconectando un laberinto de cables para modificar su comportamiento [39].

Por su parte, antes de completar su trabajo en la ENIAC, John W. Mauchly y J. Presper Eckert concibieron una forma más sencilla de cambiar el comportamiento de la máquina, la cual ya se encontraba en construcción, por lo que su idea establecería las bases para la máquina sucesora: la computadora electrónica de variable discreta EDVAC (*Electronic Discrete Variable Automatic Computer*). En una descripción escrita en Septiembre de 1945 mencionaron "Una característica importante de este dispositivo sería que las instrucciones de operación y las tablas de funciones serían almacenadas en los mismos dispositivos de almacenamiento utilizados para los números" [39].

El concepto de almacenar tanto instrucciones como datos en una unidad común de almacenamiento habría de convertirse en una característica fundamental de la computadora automática universal UNIVAC (*UNIVersal Automatic Computer*) y de prácticamente cualquier computadora que precediera. El principio de *programa almacenado* fue la clave del éxito de la UNIVAC, ya que permitió la construcción de una computadora que contaba con capacidades mucho más generales que la ENIAC y que, aún así, necesitaba un menor número de tubos al vacío. Llevó también al establecimiento de la *programación* (más tarde *software*) como un aspecto al mismo tiempo separado pero igualmente importante que el diseño del hardware.

Aún cuando Eckert y Mauchly concibieron desde 1944 el principio de almacenamiento del programa, eran eran prácticamente desconocidos fuera del colegio *Moore School of Electrical Engineering*, de la Universidad de Pennsylvania. No obstante, el matemático húngaro John Von Neumann, a pesar de estar involucrado en múltiples proyectos, estaba suficientemente

intrigado con los desarrollos que tomaban lugar en el colegio Moore como para presentarse con Eckert y Mauchly e involucrarse en el proyecto. Fue así como, el 30 de junio de 1945 publicó su trabajo *"First Draft of a Report on the EDVAC"*, donde describía la máquina en términos lógicos más que en términos de los componentes que la integraban. Esto, aunado a su reputación internacional, provocaría que tal publicación fuera citada como el fundamento de la computación moderna [39].

A pesar de que la frase "arquitectura de Von Neumann" se encuentra tan arraigada como para ser suplantada, definitivamente Eckert y Mauchly, quienes demostraron profundo entendimiento sobre la naturaleza de la computación electrónica, bien merecen el tributo.

2.4. Arquitectura Secuencial

A las computadoras que realizan la ejecución de instrucciones cíclicamente, una a la vez, se les conoce como *computadoras secuenciales*, siendo la arquitectura de Eckert y Mauchly el ejemplo por excelencia. La versión actual de una máquina basada en la *arquitectura secuencial* cuenta con, al menos, las siguientes características [45]:

- Contiene tres sistemas de hardware: unidad de procesamiento central CPU (*Central Processing Unit*), una memoria principal y un módulo de Entrada/Salida, como se muestra en la figura 2.1.
- Cuenta con la capacidad de realizar el procesamiento secuencial de instrucciones.
- Ofrece una única ruta de comunicación entre la memoria y la unidad de control, aspecto que obliga la conmutación entre los ciclos de acceso a instrucciones y de acceso a datos.

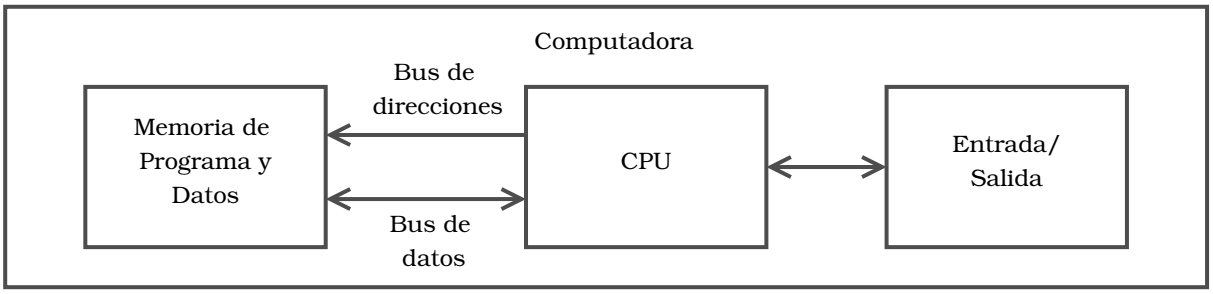


Figura 2.1: Estructura básica de una computadora secuencial.

A su vez, la CPU esta conformada por tres elementos principales (figura 2.2):

- La unidad lógico-aritmética ALU (*Arithmetic Logic Unit*) para manipular la información mediante operaciones.
- La unidad de control o *secuenciador*, que permite traer y ejecutar instrucciones.
- Un banco de registros para almacenar operandos y resultados.

La memoria permite almacenar las instrucciones y la información. Por su parte, la interfaz de entrada-salida permite la transferencia de información entre la computadora y una amplia gama de dispositivos externos como teclado, ratón o *mouse*, monitor, impresora, memorias externas, entre otros.

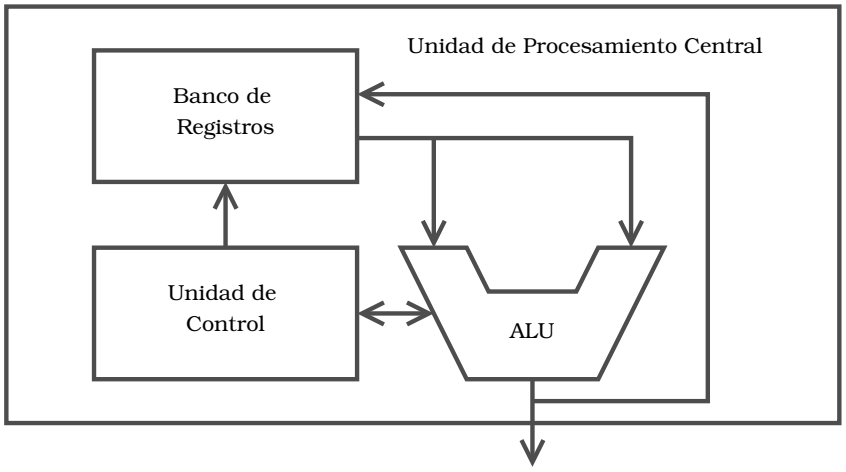


Figura 2.2: Estructura básica de la Unidad de Procesamiento Central.

Esta forma de organización resulta independiente de la tecnología de hardware que se utiliza: bien se puede decir que todos los componentes de cualquier computadora caen en alguna de estas categorías.

La arquitectura secuencial ejecuta instrucciones en un ciclo que se le conoce como *traer-decodificar-ejecutar*, el cual describe la forma de trabajar de la máquina. La iteración de un ciclo se lleva a cabo como sigue:

1.– La unidad de control *trae* la siguiente instrucción desde la memoria, utilizando un registro contador de programa para determinar su ubicación.

2.– La instrucción es *decodificada* para determinar así las operaciones a ejecutar.

3.– Si se requieren operandos, estos de transfieren de la memoria a los registros correspondientes en la unidad de control.

4.– La ALU ejecuta las operaciones necesarias y guarda el resultado en registros o memoria.

Sin embargo, el único bus de direccionamiento que conecta la CPU con la memoria (figura 2.1) representa su mayor desventaja, a tal grado que se le conoce como el *cuello de botella* de la arquitectura secuencial [45]. Esto se debe básicamente a que el procesador puede hacer un único acceso a memoria en un tiempo determinado, ya sea para leer una instrucción o para leer un dato.

Como solución a este problema surge la arquitectura *Harvard*, la cual propone el uso de memorias independientes para el programa y los datos, así como buses independientes para

acceder a ellos, con el fin de permitir la lectura simultánea de una instrucción y un dato [47]. En la figura 2.3 se muestra el esquema básico de esta arquitectura.

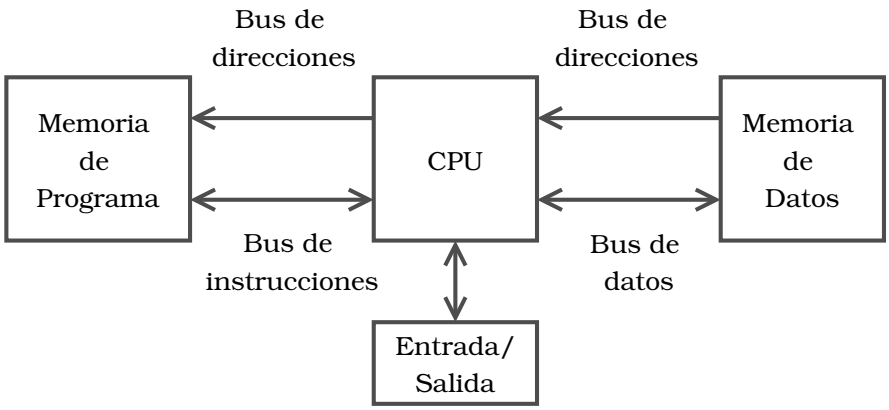


Figura 2.3: Arquitectura Harvard.

2.5. Clasificación del Paralelismo

El paralelismo en múltiples niveles es el motor que impulsa el diseño de las computadoras en la búsqueda de un mejor desempeño [46]. Existen básicamente dos tipos de paralelismo sobre los programas:

1.– *Paralelismo a nivel de datos*: se aplica cuando existen múltiples datos que pueden ser operados al mismo tiempo.

2.– *Paralelismo a nivel de tareas*: se produce cuando un trabajo genera varias tareas que se pueden operar amplia e independientemente en paralelo.

Por su parte, el hardware de la computadora puede explotar estos dos tipos de paralelismo en principalmente cuatro formas [46]:

1.– *Paralelismo a nivel de instrucciones*: explota el paralelismo a nivel de datos mediante técnicas como la segmentación encausada (*pipelining*) y la ejecución especulativa.

2.– *Arquitecturas tipo vector y unidades de procesamiento gráfico* GPU (*Graphics Processing Unit*): aprovechan el paralelismo a nivel de datos aplicando una misma instrucción en paralelo a una colección de datos.

3.– *Paralelismo a nivel de núcleos*: explota tanto el paralelismo a nivel de datos como a nivel de tareas en un modelo de hardware estrechamente acoplado que permite la interacción entre núcleos paralelos.

4.– *Paralelismo a nivel de peticiones*: aprovecha el paralelismo entre tareas que se encuentran en gran parte desacopladas, ya sean especificadas por el usuario o por el sistema operativo.

En 1966 Michael Flynn realizó un estudio acerca de los esfuerzos que se realizaban en la década de 1960 en cómputo paralelo y encontró una clasificación sencilla cuyas abreviaciones se continúan utilizando [46]. Observó el paralelismo en el flujo de instrucciones y datos determinando que toda computadora podía clasificarse en alguna de las siguientes categorías:

1.– SISD (*Single Instruction, Single Data*): estas computadoras manejan un *flujo único de instrucciones y un flujo único de datos*, como lo hacen las computadoras secuenciales basadas en uniprocesadores; permiten el uso del paralelismo a nivel de instrucciones.

2.– SIMD (*Single Instruction, Multiple Data*): manejan un *flujo único de instrucciones* ejecutadas por múltiples procesadores que utilizan su propio *flujo de datos*. Pueden hacer uso del paralelismo a nivel de datos aplicando la misma operación a múltiples datos en paralelo, para lo cual cada procesador cuenta con su propia memoria.

3.– MISD (*Multiple Instruction, Single Data*): distintos procesadores ejecutan un *flujo múltiple de instrucciones* sobre *el mismo flujo de datos*. No se ha construido ningún multiprocesador comercial de este tipo.

4.– MIMD (*Multiple Instruction, Multiple Data*): distintos procesadores ejecutan su propio *flujo de instrucciones* sobre su propio *flujo de datos*. Por lo general es más flexible que SIMD, pero es inherentemente más caro. Su objetivo es conseguir el paralelismo a nivel de tareas, aunque también puede conseguirlo a nivel de datos, peticiones y núcleos.

2.6. Atributos del Procesador

La arquitectura del procesador es fundamental para el diseño de sistemas digitales. La comprensión del diseño de la computadora le da forma al sistema digital mediante el uso de un microprocesador como el elemento central de control [38]. Este, a su vez se convierte en una plataforma programable donde se pueden implementar distintos algoritmos. Finalmente se diseña una lógica digital que rodea al microprocesador para asistirlo en distintas tareas.

La objetivo que persigue el diseñador de un procesador es determinar los *atributos* con los que debe contar y orientarlo a maximizar el desempeño con los recursos disponibles [46]. Dicha tarea abarca tres aspectos principales:

- El *diseño del hardware* del microprocesador: una vez que se formulan las especificaciones necesarias, se realiza el diseño lógico para seleccionar los elementos que conformarán el sistema. En ocasiones también se le conoce como *la implementación*.

- La *organización* se refiere a la forma en la que los componentes operan y se conectan para formar el sistema. Incluye aspectos de alto nivel como el diseño de la unidad de control, donde se implementan las operaciones lógico-aritméticas, los saltos en el flujo del programa y la transferencia de datos. También se le conoce como *microarquitectura*.

– El diseño del *conjunto de instrucciones* que el sistema será capaz de ejecutar. Aquí se determina la procedencia y destino de los datos (registros o memoria), los modos de direccionamiento, el tipo y tamaño de los operandos, los tipos de operaciones, el control de flujo y la codificación de las instrucciones. Usualmente se le conoce como *arquitectura*.

2.6.1. El Controlador

Uno de los componentes más importantes dentro de un procesador es *el controlador*, el cual se encarga de sincronizar y coordinar las acciones que deben ejecutarse [47]. En la figura 2.4 se presenta el esquema general.

Figura 2.4: Máquina de estados finitos como controlador de una arquitectura [47].

La forma más común de implementar este elemento es mediante una máquina de estados finitos FSM (*Finite State Machine*), la cual constituye una técnica especial de modelado de circuitos secuenciales. Tal modelo resulta muy útil en el diseño de sistemas cuyas tareas forman una secuencia bien definida, siendo la arquitectura del procesador un buen ejemplo de este tipo de sistemas [48].

En la figura 2.5 se muestra el diagrama a bloques básico de una FSM. Si la salida de la máquina depende tanto del estado presente como de la entrada actual, se le llama máquina *Mealy*; si únicamente depende de el estado actual, se le llama máquina *Moore* [48].

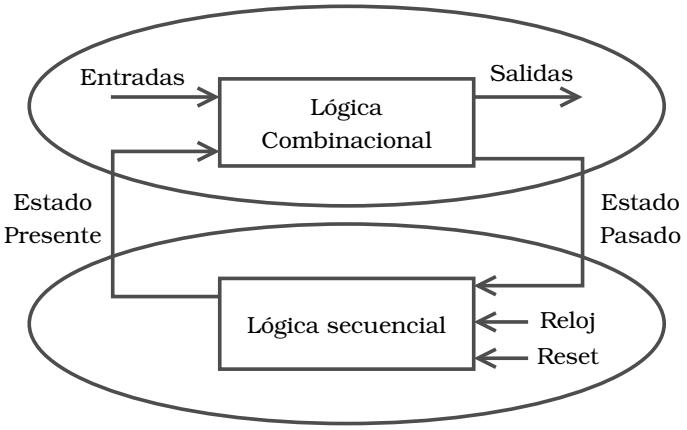


Figura 2.5: Diagrama a bloques de una máquina de estados finitos FSM [48].

Para el diseño de FSMs relativamente sencillas, el *diagrama de estados completo* resulta la forma de trabajo más práctica. No obstante, para máquinas de mayor complejidad existen otras herramientas más adecuadas como la carta ASM (*Algorithmic State Machine*) [49], la cual es sumamente práctica para visualizar la secuencia de acciones que involucra una instrucción.

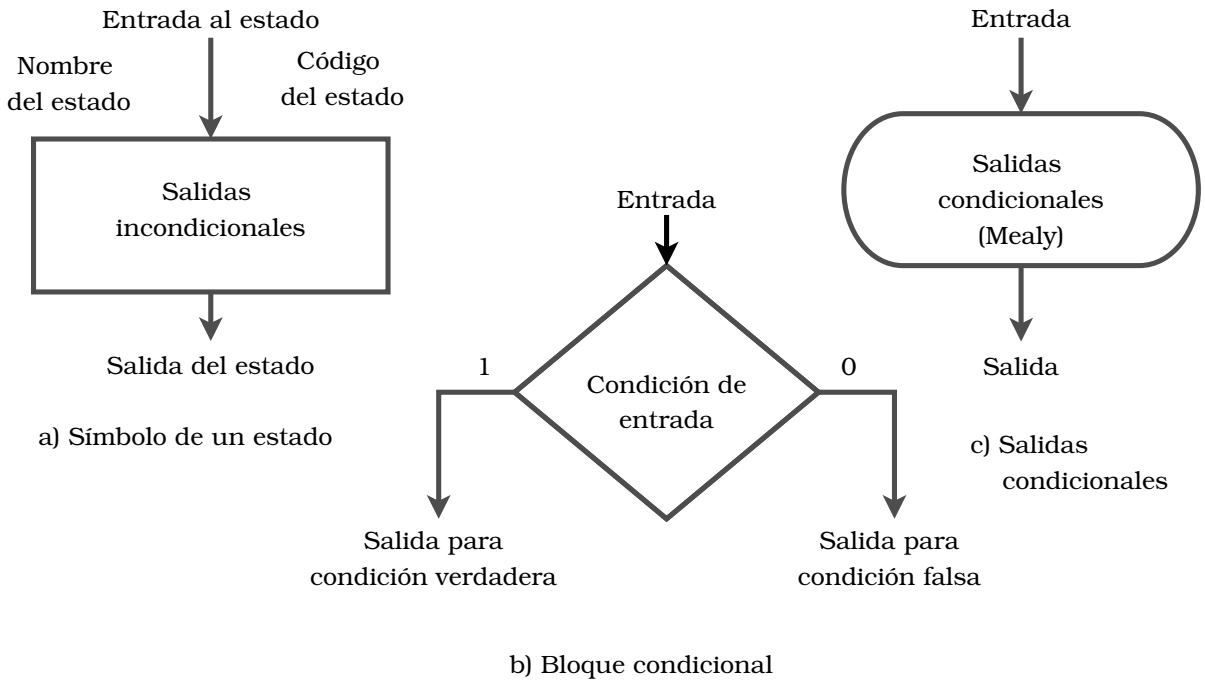


Figura 2.6: Elementos básicos de una carta ASM [49].

En la figura 2.6 se muestran los elementos básicos que se utilizan en una carta ASM, donde 2.6a representa el símbolo de un estado. En la parte superior derecha, fuera del rectángulo, se sitúa el código del estado, ya sea en hexadecimal o binario, mientras que el nombre se sitúa en la parte superior izquierda, aunque en ocasiones también se utiliza un globo a la izquierda del estado. Dentro del rectángulo se colocan las señales que se activarán de forma incondicional cuando la máquina llegue a dicho estado. Al siguiente ciclo de reloj, el flujo continua por la parte inferior. Este bloque se presenta tanto en las máquinas Moore como las Mealy, mientras que las salidas condicionales (figura 2.6c) se presentan únicamente en las máquinas Mealy y lo hacen precisamente a la salida de un bloque condicional (figura 2.6b).

2.6.2. El Secuenciador

Para el diseño del módulo de control de un procesador se requiere una FSM capaz de ejecutar algoritmos complejos. Como se vio en la sección anterior, se utilizan cartas ASM para describir la serie de acciones que deben realizarse en la ejecución de una instrucción. Los dispositivos capaces de implementar el control que requieren tales algoritmos se conocen como *secuenciadores* [47].

En la figura 2.7 se muestra el esquema básico del secuenciador para la arquitectura secuencial del 68HC11. La dirección de la instrucción siguiente puede provenir de dos fuentes: del registro PC o de una dirección externa, la cual puede corresponder a una dirección de salto por ejemplo. La *lógica interna* se encarga de evaluar las condiciones que determinan qué en-

trada será seleccionada por el multiplexor. De igual forma, genera las banderas que permiten seleccionar entre tres diferentes fuentes para la dirección externa [47].

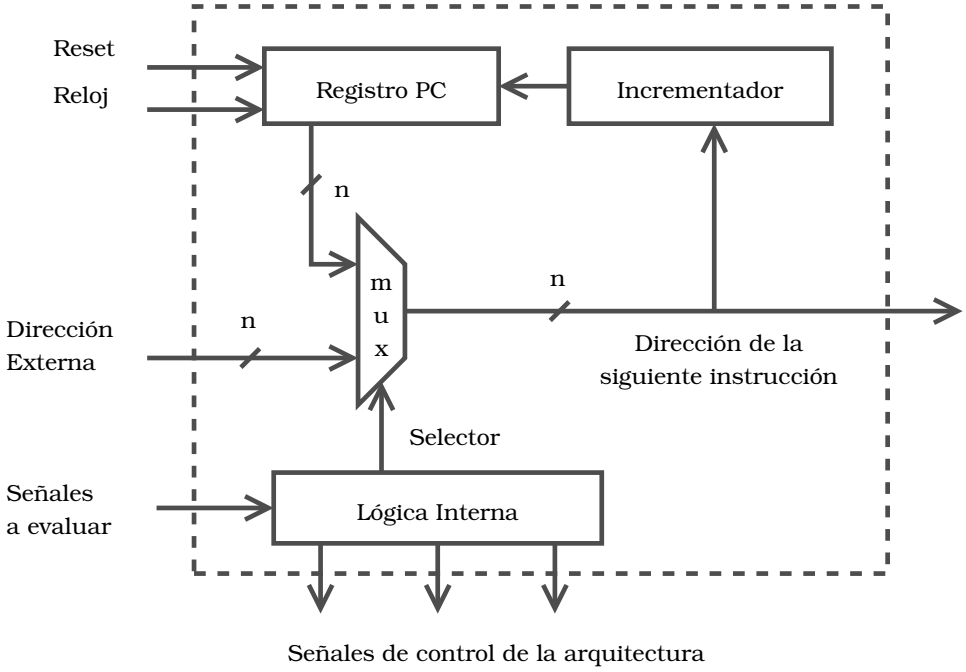


Figura 2.7: Esquema básico de un secuenciador [47].

2.7. Conjunto de Instrucciones

La multitud de tareas complejas que puede realizar un procesador se divide en secuencias de operaciones sencillas que manipulan datos individuales y toman decisiones en base a tales cálculos [38].

El diseño del hardware del procesador se encuentra estrechamente ligado con el conjunto de instrucciones que será capaz de ejecutar. Prácticamente todas las instrucciones de un microprocesador se pueden clasificar en:

- *Aritmético-lógicas*: permiten la modificación aritmética o lógica de los datos.
- *De transferencia*: permiten guardar la información y recuperarla cuando sea necesario.
- *De salto o control de flujo*: las cuales habilitan la ejecución de instrucciones en distintas secuencias.

Cada instrucción se representa mediante un código binario conocido como *código de la operación* (CoOp), el cual consiste en una cadena de bits que codifica las señales de control necesarias para que se ejecute determinada operación.

2.7.1. Tipos de Direccionamiento

Las instrucciones que puede ejecutar un procesador se clasifican por la forma en que acceden a los datos [47]. A continuación se presentan los seis tipos más comunes de direccionamiento, suponiendo una memoria de 64 KiloBytes y ancho de palabra de ocho bits.

- *Acceso Inherente:* no necesita operando, por lo que únicamente se utiliza una localidad de memoria para el CoOp (figura 2.8).

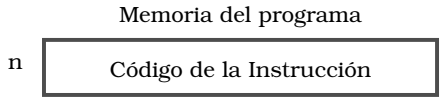


Figura 2.8: Direccionamiento por acceso inherente.

- *Acceso inmediato:* ocupa dos localidades de memoria, la primera para el CoOp de la instrucción, y la segunda para el *dato* a utilizar (figura 2.9).

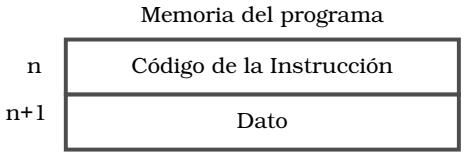


Figura 2.9: Direccionamiento por acceso inmediato.

- *Acceso directo:* utiliza la primer localidad de memoria para el CoOp y la siguiente para almacenar la parte baja de la dirección, asumiendo como cero la parte alta (figura 2.10).

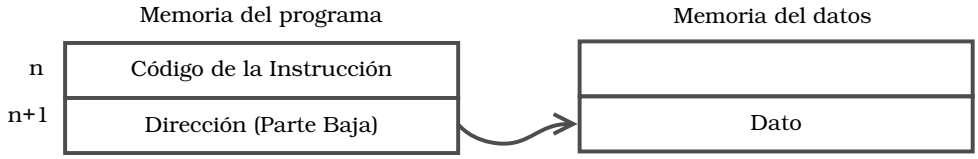


Figura 2.10: Direccionamiento por acceso directo.

- *Acceso extendido:* las dos localidades contiguas al CoOp forman la dirección a 16 bits (parte alta y parte baja) del dato a operar (figura 2.11).

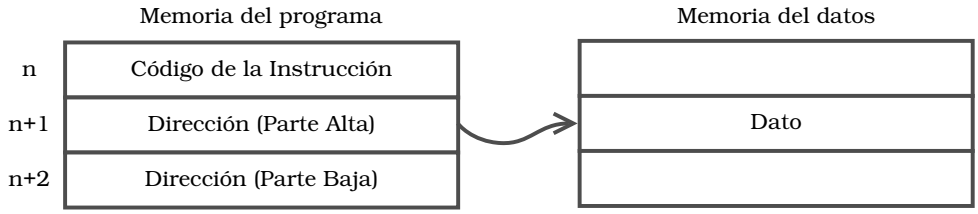


Figura 2.11: Direccionamiento por acceso extendido.

- *Acceso indexado:* la localidad adyacente al CoOp contiene un *desplazamiento* que, sumado al contenido de cierto registro, forma la dirección del dato (figura 2.12).

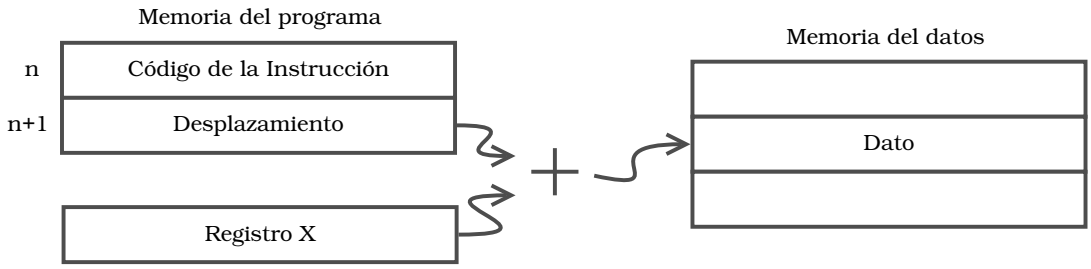


Figura 2.12: Direccionamiento por acceso indexado.

– *Acceso relativo:* este tipo de direccionamiento se utiliza para instrucciones de salto, donde la dirección de la siguiente instrucción se calcula sumando el *desplazamiento* al valor del registro Contador del Programa (figura 2.13).

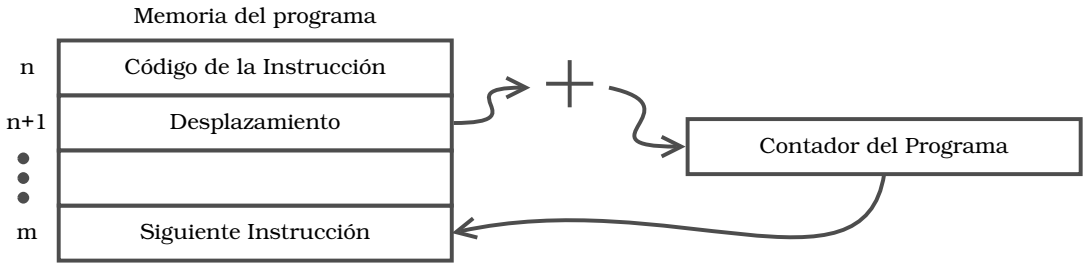


Figura 2.13: Direccionamiento por acceso relativo.

2.7.2. Filosofía de Diseño

Finalmente, cuando se diseña un conjunto de instrucciones, usualmente se elige entre dos filosofías de cómputo [38]. A la primera se le conoce como CISC (*Complex Instruction Set Computing*). El lineamiento básico es que el procesador debe contar con un conjunto de amplio de instrucciones de *complejidad elevada*, lo cual se traduce en una mayor cantidad de etapas por instrucción y, por tanto, una menor velocidad de ejecución, ya que cada estado de la carta ASM se ejecuta típicamente en un ciclo de reloj.

En la figura 2.14 se presenta el diagrama de flujo de la instrucción *Staa* con direccionamiento indexado, mediante el registro *x* (RegX), y desplazamiento 0x01. Dicho diagrama muestra los códigos de estado correspondientes a la arquitectura secuencial del microcontrolador 68HC11 bajo la filosofía CISC.

La instrucción Staa almacena el dato contenido en el acumulador *a* (AccA) en la localidad de memoria apuntada por el contenido del RegX, más el desplazamiento almacenado tras el CoOp. En el apéndice A se presenta el diagrama a bloques de tal arquitectura, así como la carta ASM correspondiente al diagrama de flujo de la figura 2.14. También se presentan dos instrucciones adicionales que pretenden ilustrar la *complejidad* de esta filosofía.

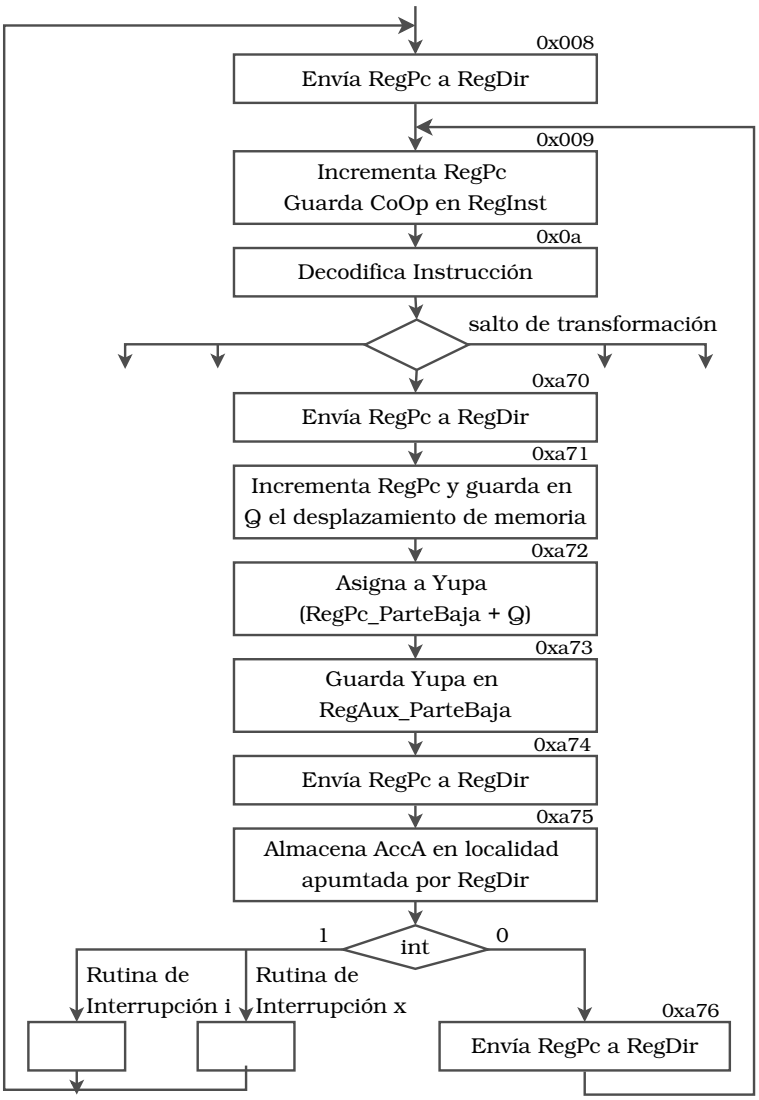


Figura 2.14: Diagrama de flujo de la instrucción Staa para la arquitectura CISC del 68HC11 [47].

Con fines demostrativos, dicha arquitectura se describió en el lenguaje VHDL (sección 2.9.2) para simular la ejecución de algunas instrucciones. En la figura 2.15 se muestra la simulación de la instrucción Staa indexada con desplazamiento 0x01, donde se puede apreciar que la ejecución efectivamente toma 10 ciclos de reloj, correspondientes a los 10 estados del diagrama 2.14. La línea *loc0x11* revela el contenido de la localidad de memoria 0x11 y demuestra la carga del dato reflejado en la línea *racca*.

Por su parte, el cómputo RISC (*Reduced Instruction Set Computing*) propone que las instrucciones deben contar con una *complejidad reducida*, lo cual se consigue buscando que todas tengan el mismo tamaño y, por lo tanto, el mismo número de etapas, permitiendo así aplicar técnicas de emulación del paralelismo. La ejecución de tareas complejas, como las que podría realizar una única instrucción tipo CISC, se consigue ejecutando una o varias tareas simples a mayor velocidad.

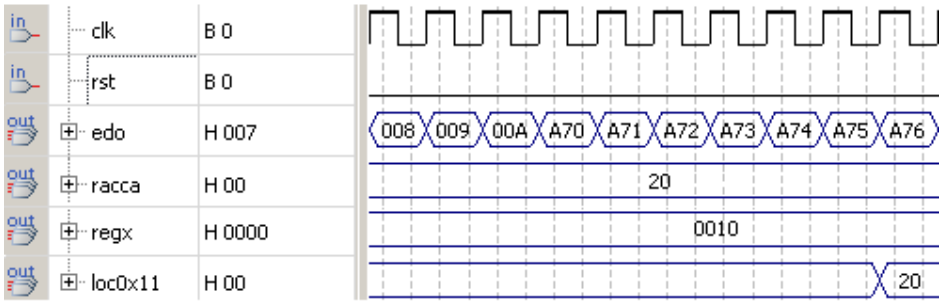


Figura 2.15: Simulación temporal en Quartus II de la instrucción Staa con direccionamiento indexado y desplazamiento 0x01.

En la figura 2.16 se presenta el diagrama de flujo que corresponde a cualquier instrucción de una arquitectura RISC de cuatro etapas. Cabe resaltar de dicha figura que la ejecución de todas las instrucciones toma únicamente cuatro ciclos de reloj.

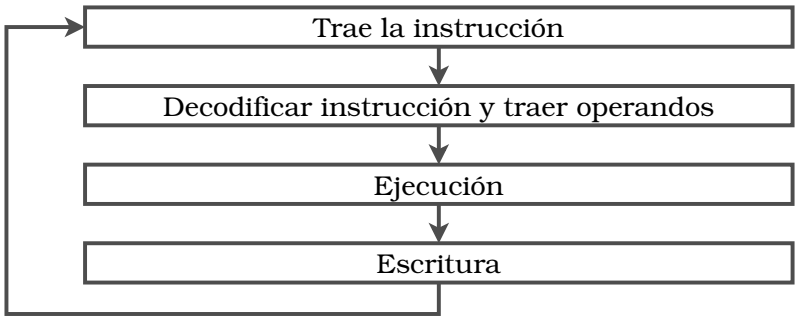


Figura 2.16: Diagrama de flujo para instrucciones tipo RISC de cuatro etapas.

De este último diagrama se comienzan a visualizar las ventajas que presenta el cómputo RISC sobre el CISC, ya que la ejecución de la instrucción Staa indexada que en su forma compleja toma 10 ciclos de reloj, en su forma reducida podría realizarse con tan sólo una instrucción de cuatro ciclos.

2.8. Segmentación Encausada

Como se vio en la clasificación de paralelismo por flujos de Flynn (sección 2.5), tres de las cuatro categorías (SIMD, MISD y MIMD) implican la implementación de múltiples procesadores o núcleos para conseguir el paralelismo en determinado nivel. Sin embargo, la técnica de *segmentación encausada* permite emularlo sin necesidad de replicar el hardware [47].

Esta técnica es análoga a una línea de producción, donde múltiples etapas realizan una parte del trabajo completo. Por ejemplo, en la línea de ensamble de una fabrica de autos, los trabajadores realizan pequeñas tareas para producir varios autos al día. En una línea bien

balanceada, un auto nuevo sale en el mismo tiempo que toma la ejecución de cualquier etapa, aspecto que no implica una reducción del tiempo que toma construir un auto, sino que refleja un incremento en el número de autos que se están fabricando al mismo tiempo y, por lo tanto, aumenta el número de autos fabricados.

Al igual que en la línea de ensamble de los vehículos, la segmentación encausada divide el trabajo en etapas cuya ejecución toman una fracción del tiempo de la que toma la instrucción completa. Dichas etapas se *acoplan* para formar un *cauce* en el cual las instrucciones entran por un extremo, se procesan por partes y salen por el otro extremo. Nuevamente, esta segmentación no disminuye el tiempo que toma ejecutar una instrucción, sino que incrementa el número de instrucciones que se procesan simultáneamente, incrementando así la tasa de entrada-salida de instrucciones.

En la figura 2.17 se muestra la estructura básica de un procesador segmentado de cuatro etapas, las cuales que se encuentran unidas por registros de acoplo [47]. El propósito de dichos registros es evitar la pérdida de la información de la etapa anterior para hacerla llegar a la etapa siguiente. Típicamente las etapas del ciclo del procesador secuencial (sección 2.4) corresponden con las primeras etapas del cauce: e1 = traer, e2 = decodificar y e3 = ejecutar. La cuarta etapa en la figura es la de *escritura*, en la cual se almacenan los datos resultantes de la etapa de ejecución, ya sea a memoria o a algún registro.

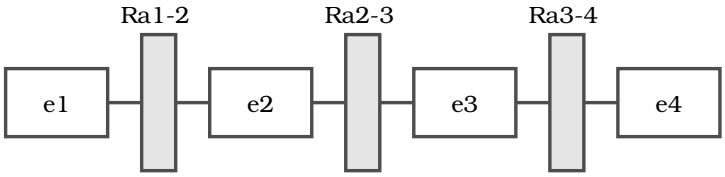


Figura 2.17: Estructura básica de una procesador segmentado [47].

En la figura 2.18 se presenta un diagrama de tiempos contra tareas para el cauce de la figura 2.17. Aquí se observa que la primera instrucción (i1) requiere cuatro ciclos de reloj para su ejecución. Sin embargo, una vez que la primera instrucción sale del cauce (t4), éste se encuentra lleno y, con cada ciclo de reloj se finaliza la ejecución de otra instrucción (i2,i3). De igual forma se aprecia que no se ha disminuido el tiempo de ejecución de cada instrucción, sino que se ha aumentado el desempeño del procesador, ya que el tiempo de salida de una instrucción respecto a la anterior es de un ciclo de reloj.

2.8.1. Desempeño

La forma más natural de medir el desempeño de un procesador es mediante el tiempo: la máquina que realiza la misma cantidad de trabajo en un menor tiempo es la más rápida [50]. Sin embargo, existen múltiples métricas definidas a partir de determinados segmentos de tiempo. Entre ellas, se encuentran dos medidas típicas del desempeño de un computador:

– *Tiempo de respuesta*: el cual se refiere al tiempo que existe entre el inicio y el final de una tarea, por lo que también se le conoce como *tiempo de ejecución*.

– *Rendimiento*: el cual muestra la cantidad de trabajo realizada en un tiempo determinado.

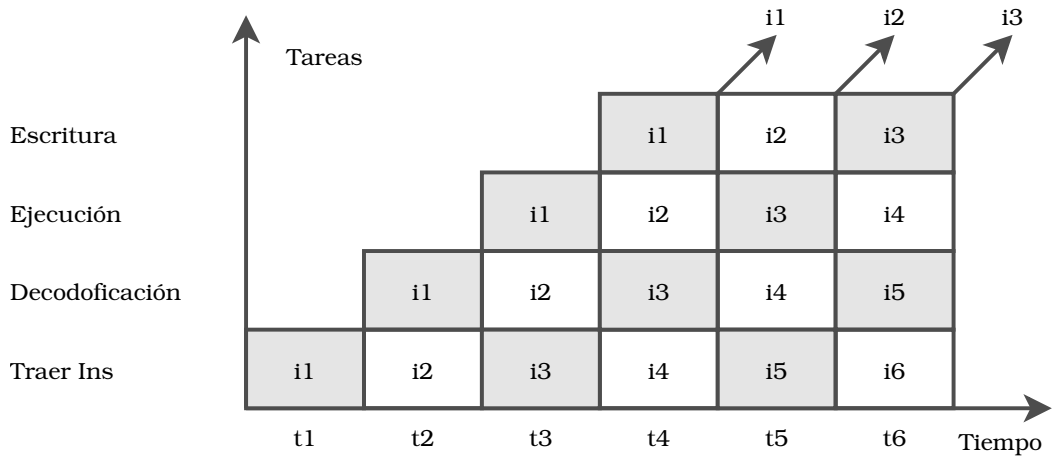


Figura 2.18: Diagrama de tiempo contra tareas [47].

Normalmente la reducción del tiempo de ejecución también mejora el rendimiento, mientras que el flujo inverso no necesariamente se cumple. Una forma de maximizar el desempeño implica minimizar el tiempo de ejecución de una determinada tarea. Por esta razón se puede relacionar el desempeño con el tiempo de ejecución de la siguiente forma:

$$\text{Desempeño}_x = \frac{1}{\text{Tiempo de ejecución}_x} \tag{2.3}$$

Prácticamente todas las computadoras utilizan un reloj que oscila a una frecuencia constante y determina el momento en que ocurren los eventos en el hardware. Estos intervalos discretos de tiempo se les llama *ciclos de reloj*. Los diseñadores también se refieren a la duración de estos ciclos como el *periodo del reloj* [50].

Una forma de definir el tiempo de ejecución en función de ciclos de reloj, así como de su periodo, se expresa en la siguiente fórmula:

$$\text{Tiempo de ejecución}_{CPU} = \text{Ciclos de ejecución}_{CPU} * \text{Periodo del reloj} \tag{2.4}$$

O, atendiendo a la frecuencia del reloj como inverso del periodo:

$$\text{Tiempo de ejecución}_{CPU} = \frac{\text{Ciclos de ejecución}_{CPU}}{\text{Frecuencia del reloj}} \tag{2.5}$$

Así como el desempeño en la línea de ensamble se determina por el tiempo que tarda en salir un nuevo auto de la línea, el desempeño del cause se determina por el tiempo que tarda en terminarse la ejecución de una instrucción respecto a la anterior.

Cabe destacar que uno de los aspectos más importantes en el diseño del cauce implica que el periodo de reloj debe durar, al menos, el mismo tiempo que toma la ejecución de la etapa

más lenta, ya que todas las etapas deben proceder al mismo ritmo [47]. Por esta razón, el trabajo del diseñador consiste en balancear el tamaño de las etapas para evitar o minimizar tiempos muertos dentro de las mismas.

Idealmente la mejora de tiempo es directamente proporcional al número de etapas: un cauce de *n* etapas es *n* veces más rápido. La siguiente fórmula muestra la mejora en el tiempo de ejecución de una instrucción:

$$\text{Tiempo de ejecución}_{encausado} = \frac{\text{Tiempo de ejecución}_{no\_encausado}}{\text{Número de etapas}} \tag{2.6}$$

Por lo tanto, el desempeño ideal de la arquitectura quedaría definido como:

$$\text{Desempeño}_{encausado} = \frac{\text{Número de etapas}}{\text{Tiempo de ejecución}_{no\_encausado}} \tag{2.7}$$

Sin embargo, en la realidad este desempeño no se alcanza debido a la existencia de situaciones de *riesgo* que obligan la detención o vaciado del cauce, disminuyendo así su rendimiento.

2.8.2. Riesgos

En el caso ideal, una arquitectura segmentada ejecuta instrucciones que son independientes una de otra, lo que implica que la siguiente instrucción no requiere el resultado de la anterior. Sin embargo, en la operación real, pueden presentarse dos tipos de riesgos [47].

El primero de ellos es la *dependencia de datos*, situación en la cual el resultado de una instrucción es utilizado por la instrucción siguiente. Cabe recordar que en el cauce presentado en la figura 2.17, las escritura a memoria o registros se produce hasta la última etapa, por lo que la siguiente instrucción accederá a un dato erróneo en la etapa de decodificación. Para ejemplificar el conflicto, supóngase el siguiente segmento de código, donde `rac` se refiere al registro acumulador:

> Carga `rac` con dato a.
> Multiplica `rac` por dato b.
> Suma dato c al `rac`.
> Resta `rac` menos dato d.

En la figura 2.19 se presenta un diagrama de múltiples ciclos de reloj que muestra la situación. Aquí puede apreciarse que la instrucción de multiplicación depende de la carga del dato *a* al `rac`, así como la instrucción suma depende del resultado de la multiplicación anterior y la resta depende del resultado de la suma. La *x* representa el valor inicial del `rac`.

El resultado que se espera obtener con el código es $rac = ((a * b) + c) - d$. No obstante, debido a que la carga de los operandos se realiza en la decodificación (e2) y la escritura a registros en la última etapa (e4), el resultado obtenido al final del código es $rac = (x * b) - d$, como se aprecia en diagrama de la figura 2.19.

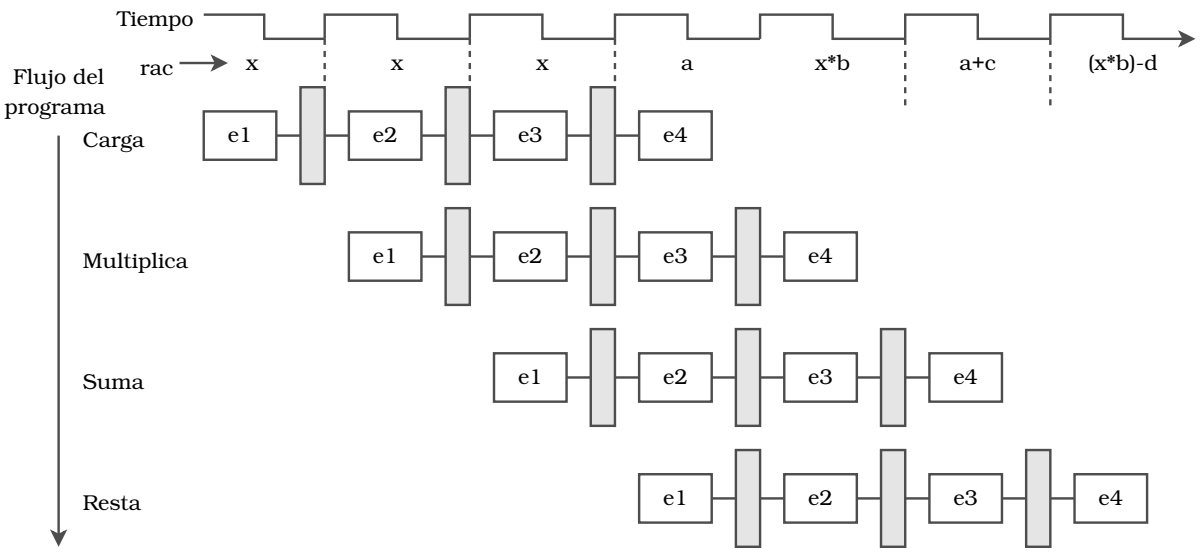


Figura 2.19: Diagrama de ciclos de reloj para el código de ejemplo.

Existen dos formas de resolver este tipo de riesgo. La primera de ellas es por *software* (figura 2.20), en la cual se introducen operaciones vacías (*nop*) entre las instrucciones que presentan riesgo, logrando así dar tiempo suficiente a que se actualice el registro con el dato correcto para la siguiente operación.

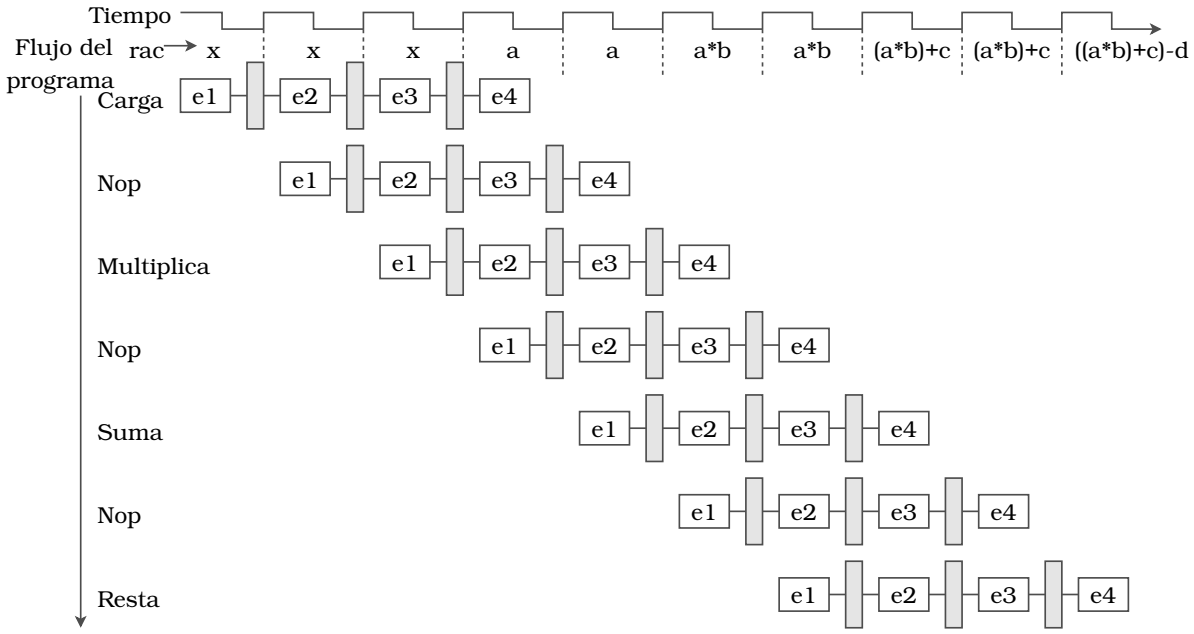


Figura 2.20: Diagrama de la solución por *software*.

La segunda opción es mediante *hardware*, donde es necesario implementar una *unidad de detenciones* que detecte las situaciones de riesgo y genere las mismas señales de control que produce la instrucción nop. En términos de los diseñadores, se dice que se *introduce una burbuja*.

En la figura 2.21 se muestra el diagrama de ciclos de reloj que ilustra la solución por detenciones, donde se aprecia la obtención del resultado esperado. La ventaja del uso de la unidad de detenciones es liberar al usuario de la necesidad de introducir operaciones nop en el código. Sin embargo, ambas opciones presentan una desventaja: el código aumenta su tiempo de ejecución, lo cual disminuye el desempeño del cauce. El código que originalmente tomaría siete ciclos de reloj toma diez.

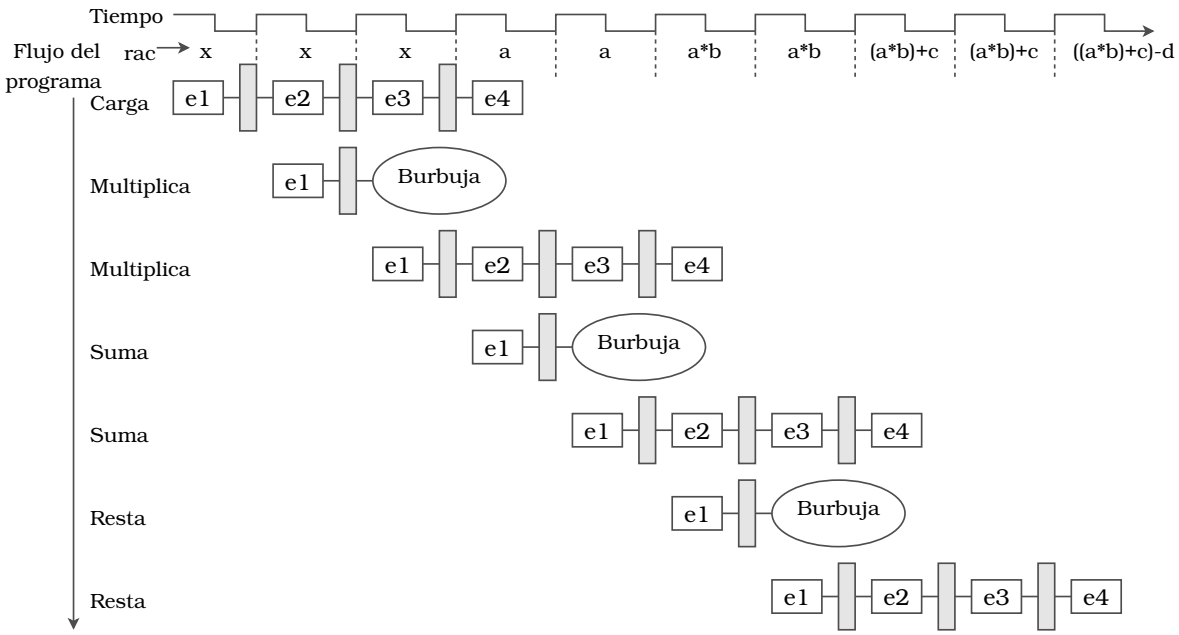


Figura 2.21: Diagrama de la solución por *hardware*.

Este impacto negativo puede compensarse si se atiende a la función de los registros de acoplamiento: almacenar la información generada en la etapa anterior para proveerla a la siguiente en el próximo ciclo de reloj. Esto implica que, tras ejecutarse la instrucción en la etapa tres, el resultado que se almacenará en el registro destino (rac en el código de ejemplo) se encuentra disponible en el registro de acoplo entre la etapa de ejecución y la de escritura. Mediante el uso de la técnica de *adelanto* es posible detectar el riesgo y proveer el dato correcto a la etapa de ejecución, logrando así eliminar el retraso que implica el uso de detenciones (figura 2.22).

Por otra parte, el segundo tipo de riesgo que existe en la operación de un procesador encausado es el *riesgo por saltos*, el cual es semejante al de dependencia de datos, pero se produce en la ejecución de instrucciones de control de flujo, como los saltos, los cuales se resuelven hasta la última etapa. Para ejemplificar la situación, se presenta el siguiente código de ejemplo:

```
                Resta rac con dato a.
                Salta a FlujoA si el resultado no es cero.
                Incrementa rac.
FlujoA          Divide rac por dato b.
```

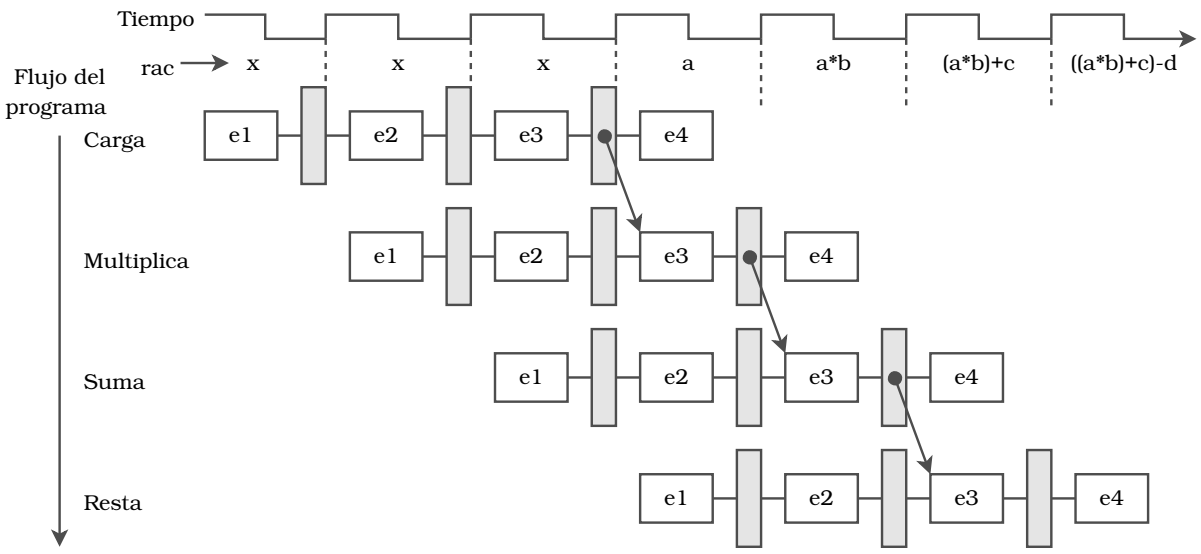


Figura 2.22: Diagrama de la solución utilizando la técnica de *adelanto*.

En este nuevo código de ejemplo se puede apreciar que, en caso de que la operación de resta resulte cero, el incremento no se ejecutaría. Sin embargo, debido a que el salto se resuelve hasta la cuarta etapa, la instrucción entra al cauce y termina ejecutándose como se puede apreciar el la figura 2.23.

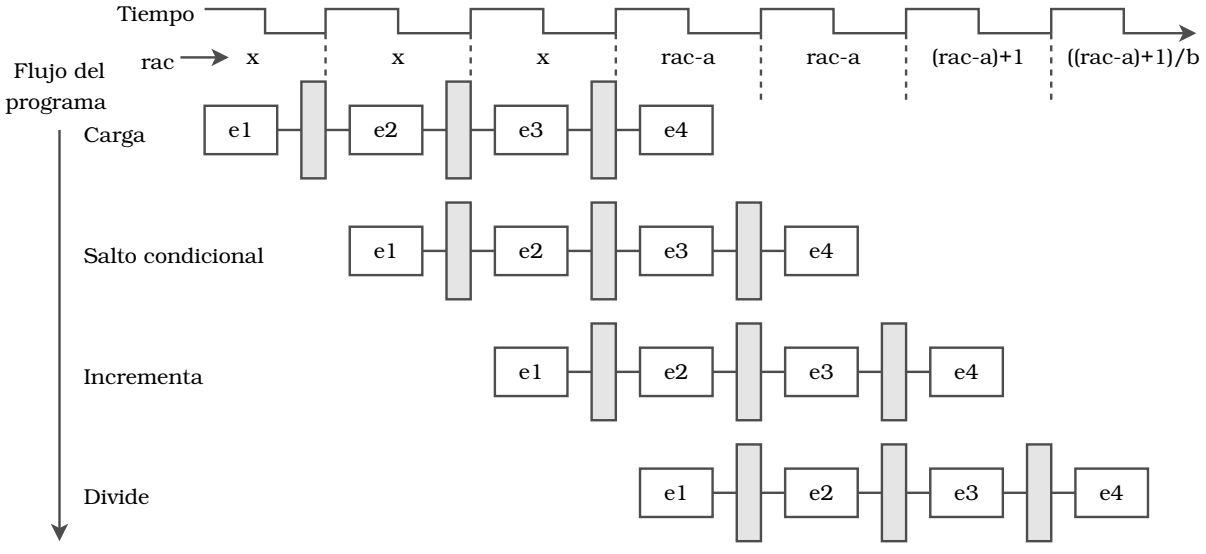


Figura 2.23: Ejemplo de riesgo por salto.

La forma de resolver este tipo de riesgos es mediante la técnica de detenciones, con lo cual se logra *esperar* la resolución del salto y vaciar el cauce en caso de que el salto se ejecute. En el código de ejemplo, la unidad de detenciones introduce una burbuja entre la instrucción de salto y el incremento, evitando así la ejecución del incremento. En ese caso, si el resultado es diferente de cero, el código obtendría el resultado correcto: $(rac - a)/b$.

Es por esta razón que una arquitectura segmentada debe contemplar el diseño de una Unidad de Detección de Riesgos que aplique ambas técnicas para detectar y solucionar los riesgos por dependencia de datos y aquellos debidos a instrucciones de control de flujo.

2.9. Lógica Programable

Al realizar el diseño de un sistema digital, por ejemplo la arquitectura de un procesador, es necesario tomar una decisión importante: la tecnología a utilizar para implementar el diseño. Aquí existen principalmente dos opciones: circuitos integrados de aplicación específica ASIC (*Application Specific Integrated Circuit*) o la lógica programable.

El uso de un ASIC provee soporte para diseños grandes y complejos con un gran desempeño. El costo de fabricación es sumamente elevado a menos que se produzcan por volumen. El tiempo de desarrollo es elevado y una modificación al diseño resulta costosa temporal y económicamente [44].

Por su parte, la lógica programable permite tiempos cortos para el diseño y una implementación sencilla mediante el uso de tarjetas de desarrollo, las cuales se encuentran disponibles en el mercado a precios razonables. Realizar modificaciones al diseño consume poco tiempo y no representa un gasto adicional, ya que únicamente se reprograma el dispositivo con el nuevo diseño. Esta tecnología ofrece al diseñador un amplio rango de dispositivos, que incluyen diferentes combinaciones de recursos, empaquetados, pines disponibles e, incluso, distintos fabricantes. Además, debido al éxito que ha tenido esta alternativa, los dispositivos se producen por volumen, lo cual los hace económicamente viables [49]. Por tales razones, se propone el uso de esta tecnología para el diseño del procesador dedicado.

2.9.1. Dispositivos Lógicos Programables

El PLD (*Programmable Logic Device*) fue introducido a mediados de la década de 1970. La intención era construir circuitos lógicos combinacionales que fueran reprogramables. Sin embargo, a diferencia de los microcontroladores que ejecutan un programa sobre *hardware fijo*, la programabilidad de los PLDs estaba orientada al nivel de hardware. En otras palabras, un PLD es un circuito integrado de propósito general cuyo hardware puede reconfigurarse para ejecutar determinada función.

Los primeros PLDs fueron los de lógica de arreglo programable PAL (*Programmable Array Logic*) y los arreglos de lógica programable PLA (*Programmable Logic Array*). Utilizaban únicamente compuertas lógicas, es decir, no utilizaban biestables (*flip-flops*), permitiendo únicamente la implementación de circuitos combinacionales. Para solucionar esa limitación, pronto aparecieron los PLDs *registrados* que incluían un flip-flop en cada salida del circuito. De esta forma, era posible implementar funciones secuenciales simples [48].

A comienzos de la década de 1980, se incorporó circuitería lógica adicional en cada salida del PLD. La nueva celda de salida, llamada *macrocelda*, contenía además del flip-flop, compuertas lógicas y multiplexores. Más aún, la celda era en sí programable, permitiendo múltiples modos de operación. Adicionalmente se incorporó una realimentación de la salida hacia el arreglo programable, lo que le dio una gran flexibilidad. Dicha estructura se le conoce como GAL (*Generic pAL*). A este conjunto de dispositivos se le conoce colectivamente como SPLD (*Simple PLD*).

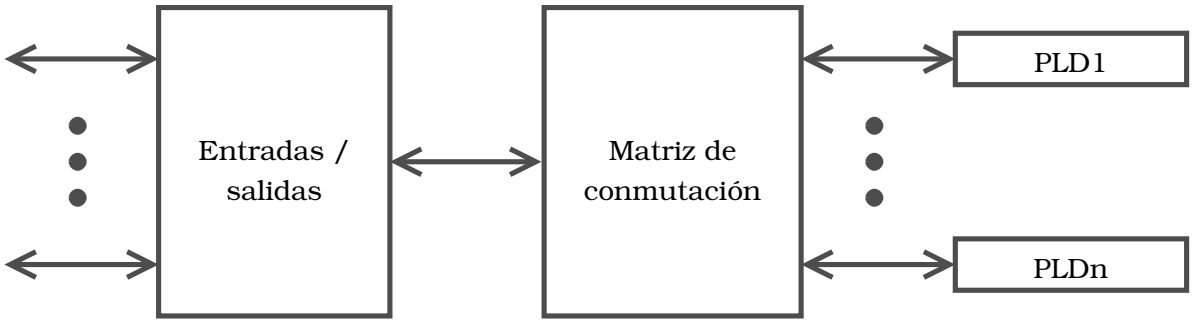


Figura 2.24: Estructura básica del dispositivo lógico programable *complejo* [48].

Más adelante, múltiples GALs serían fabricadas en el mismo circuito integrado, incorporando un esquema de interconexión más sofisticado, tecnología de fabricación más avanzada y funciones adicionales como el soporte para JTAG (*Jonit Test Action Group*) y la interfaz con varios estándares de niveles lógicos (1.8[V], 2.5[V], 5[V], etc). A este nuevo dispositivo se le conoce como CPLD (*Complex PLD*), el cual se volvió popular por su alta densidad, alto rendimiento y bajo costo [48]. En la figura 2.24 se muestra la estructura básica de estos dispositivos.

Para mediados de la década de 1980, Xilinx introdujo el arreglo de compuertas programables en campo FPGA (*Field Programmable Gate Array*), el cual difiere del CPLD en arquitectura, tecnología, funciones incorporadas, almacenamiento y costo. Está orientado a implementaciones de alto desempeño y permite el diseño de circuitos de gran escala. Una de las características más comunes de estos dispositivos es la reprogramación, la cual típicamente se consigue utilizando una memoria SRAM (*Static Random Access Memory*) para generar las interconexiones, aspecto que implica el uso de una memoria ROM (*Read Only Memory*) externa para cargar las conexiones al encender el dispositivo [48].

La arquitectura básica de un FPGA se ilustra en la figura 2.25. Consiste en un arreglo de bloques lógicos configurables CLBs (*Configurable Logic Blocks*) interconectados por un arreglo de matrices de conmutación. A diferencia de los PLDs que basan su funcionamiento en arreglos de compuertas *and* y *or*, los CLBs que forman el FPGA se basan en tablas de búsqueda LUTs (*Look-Up Tables*), ya que resultan elementos universales con los que se puede implementar cualquier función lógica [48].

Más aún, cuentan con un número mucho mayor de flip-flops, lo cual permite la construcción de circuitos secuenciales más sofisticados. Además de heredar el soporte JTAG y la

interfaz con distintos niveles lógicos de voltaje, cuentan con características como memoria SRAM, multiplicación de la señal de reloj mediante PLL (*Phase Locked Loop*) o DLL (*Delay Locked Loop*) y, en algunos casos, bloques de hardware dedicados a ciertas funciones intensivas como la multiplicación o el Procesamiento Digital de Señales [48].

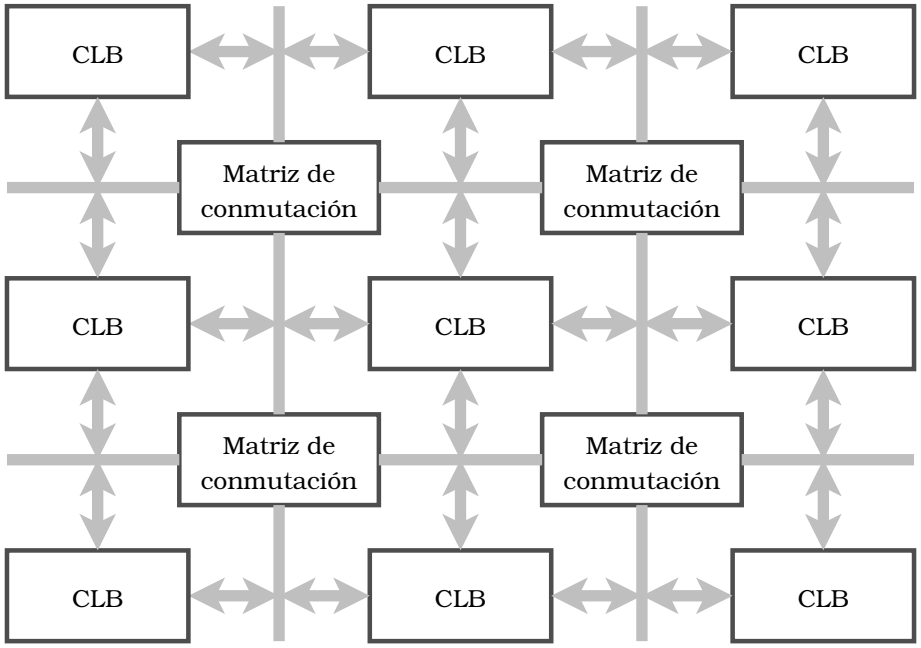


Figura 2.25: Estructura básica de un FPGA [48].

El nombre que se le da al bloque básico depende del fabricante, siendo CLB la nomenclatura de Xilinx, mientras que Altera inicialmente lo definió como elemento lógico LE (*Logic Element*) y posteriormente LAB (*Logic Array Block*). En la figura 2.26 se muestra la estructura básica del LAB en la familia Stratix III de Altera. Se encuentra formado por módulos lógicos adaptables ALM (*Adaptable Logic Module*), los cuales a su vez constan de un bloque combinacional formado por LUTs y funciones adicionales como sumadores, flop-flops y multiplexores.

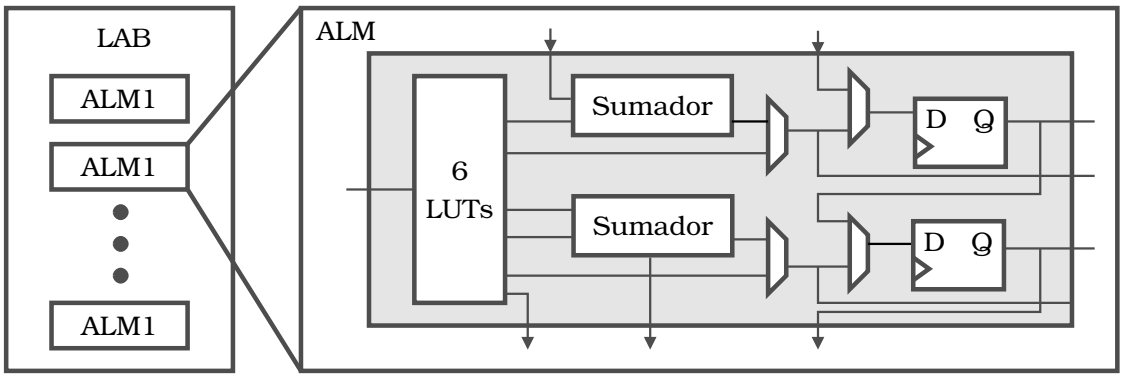


Figura 2.26: Estructura de un LAB y de un ALM en la familia StratixIII.

2.9.2. Descripción de Hardware

El proceso de encontrar un conjunto eficiente de compuertas para realizar una función determinada es una labor intensiva y propensa a errores, ya que requiere la simplificación manual de tablas de verdad y ecuaciones Booleanas, así como la traducción manual de máquinas FSM a compuertas lógicas. Sin embargo, para la década de 1990, los diseñadores descubrieron que resultaba mucho más productivo trabajar a un nivel de abstracción más elevado, especificando la función lógica y permitiendo que una herramienta de diseño asistido por computadora CAD (*Computer Aided Design*) produjera el arreglo de compuertas óptimo. Estas especificaciones las proporciona un lenguaje de descripción de hardware HDL (*Hardware Description Language*), siendo los dos más comunes Verilog y VHDL [51].

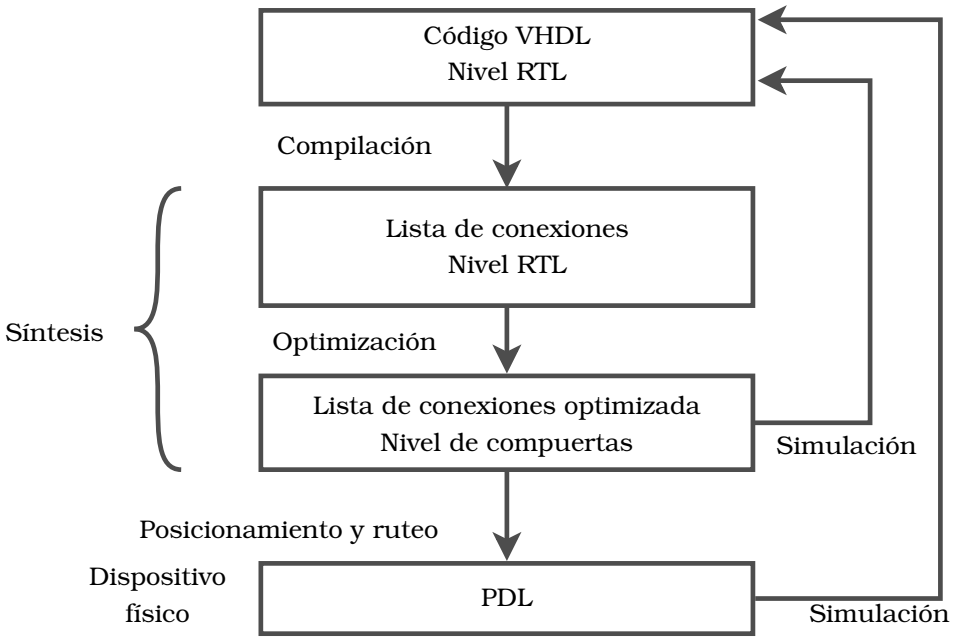


Figura 2.27: Flujo de diseño utilizando descripción de hardware [48].

Por su parte, VHDL significa *VHSIC Hardware Description Language*, donde VHSIC a su vez significa *Very High Speed Integrated Circuits*. Comenzó como iniciativa del Departamento de Defensa de los Estados Unidos en la década de 1980 y fue el primer lenguaje que se estandarizó por el Instituto de Ingenieros Eléctricos y Electrónicos IEEE (*Institute of Electrical and Electronics Engineers*) a través del estándar IEEE 1076. Posteriormente se añadió el estándar IEEE 1164 para incorporar sistemas lógicos multivalor [48].

Para el diseño del microprocesador dedicado la reconfiguración electrónica del arreglo se propone utilizar VHDL. La principal motivación radica en que dicho lenguaje pertenece a un estándar, por lo que resulta independiente de la tecnología con la que el fabricante genera su dispositivo, haciendo el diseño portable y reutilizable; además de ser aquél con el que el autor se encuentra más familiarizado.

El flujo de diseño se muestra en la figura 2.27. Se comienza escribiendo un código en VHDL. La síntesis comienza con la compilación. El archivo .hdl, que describe el circuito al nivel de transferencia de registros RTL (*Register Transfer Level*), se convierte en una lista de conexiones (*netlist*) a nivel de compuertas lógicas. El siguiente paso es la optimización de dicha lista, la cual se puede enfocar en velocidad de operación o la cantidad de recursos utilizados. A partir de esta etapa, se puede simular el funcionamiento del diseño. Finalmente, el programa de posicionamiento y *ruteo* (*fitter*) genera el plano o diseño físico a ser implementado en el dispositivo físico [48].

2.10. El manejo de la Complejidad: Abstracción

Los sistemas modernos digitales están construidos por millones o billones de transistores. No hay ser humano que sea capaz de comprender estos sistemas modelando las ecuaciones que describen el movimiento de los electrones en cada transistor y resolviéndolas simultáneamente. Por tal razón, debe aprenderse a manejar la complejidad si se pretende obtener el entendimiento necesario para llegar a construir la arquitectura de un procesador. La técnica que nos permite lograrlo es la *abstracción*, la cual implica esencialmente ocultar detalles de capas inferiores cuando es posible hacerlo [51].

Una computadora electrónica puede visualizarse como un conjunto de niveles de abstracción como se muestra en la figura 2.28. En el nivel más bajo encontramos la física: el movimiento de los electrones, el cual está descrito por la mecánica cuántica. En el siguiente nivel, encontramos que el sistema está construido por dispositivos electrónicos fundamentales, como son los diodos y los transistores, los cuales son estudiados por la física de semiconductores. La agrupación de estos dispositivos nos lleva un nivel más arriba, donde se visualizan los circuitos analógicos, los cuales son capaces de recibir y proveer un rango continuo de voltajes.

Arreglos de estos dispositivos electrónicos y/o circuitos analógicos dan paso a la generación de circuitos digitales como las compuertas lógicas, los cuales restringen los voltajes a valores discretos, dando paso a la codificación del 1 y el 0. Un nivel más arriba encontramos el diseño lógico, donde se utilizan los circuitos digitales básicos para construir estructuras más complejas como sumadores y memorias. Este nivel se liga con el de la arquitectura mediante el eslabón de microarquitectura, la cual combina los elementos lógicos de tal forma que se puedan ejecutar las instrucciones definidas por la arquitectura.

Por arriba de la arquitectura se considera el campo del software, donde encontramos primero el nivel de sistema operativo, el cual se encarga de manejar la ejecución de programas y de atender detalles de bajo nivel como el manejo de la memoria o el acceso a dispositivos de entrada y salida. Y finalmente encontramos el nivel de aplicaciones de software o programas, el cual hace uso de todas las facilidades que provee el sistema operativo para solucionar los problemas del usuario.

Figura 2.28: Niveles de abstracción [51].

Por su parte, cabe resaltar que en el diseño de circuitos de gran escala resulta impráctica la descripción mediante tablas de verdad, listas de minitérminos y ecuaciones lógicas. En este caso, es necesario utilizar metodologías de diseño estructuradas como la de *arriba hacia abajo* (*top-down*) para manejar de forma efectiva la complejidad. Dicha metodología implica un proceso en el que una función se especifica inicialmente en un alto nivel de abstracción y después se descompone en subfunciones de menor nivel, las cuales tienden a ser más concretas. Este proceso continúa hasta que se obtienen funciones manejables y módulos individuales que se pueden implementar mediante circuitos relativamente sencillos. Al proceso que comienza por las funciones más básicas para construir así la función más general se le conoce como *buttom-up* [52].

Con base en lo anterior, se plantea utilizar la metodología *top-down* para determinar los componentes básicos del *nivel lógico* con los que debe contar el procesador. Posteriormente se procederá *de abajo hacia arriba*, estructurando el nivel de *microarquitectura* para llegar finalmente a la generación de la *arquitectura* (figura 2.28).

Capítulo 3

Diseño del Procesador Dedicado

En este capítulo se presenta el diseño del procesador con el que se pretende dar solución al problema de la reconfigurabilidad, para lo cual se comienza con la selección del arreglo en el cual se basa el diseño, abordando sus principales características y presentando la forma en que se calcula teóricamente la posición angular de los elementos radiadores.

Posteriormente se presentan tanto las herramientas como el proceso de diseño de la arquitectura del procesador, desde el conjunto de instrucciones, hasta el hardware que será capaz de ejecutarlas. A continuación se abordan las posibles técnicas para la expansión de los pines disponibles del dispositivo a utilizar y, finalmente, se discute la escalabilidad del diseño mediante el paralelismo a nivel de núcleos para maximizar el desempeño del sistema.

3.1. Planteamiento y Delimitación del Problema

Un arreglo reflectivo reconfigurable electrónicamente se puede visualizar como un sistema formado por tres bloques principales (figura 3.1). El bloque de *generación* es el encargado de proveer las señales lógicas que gobiernan el funcionamiento de los elementos conmutadores. Estas señales están limitadas en voltaje y corriente a los niveles proporcionados por el dispositivo que las genera.

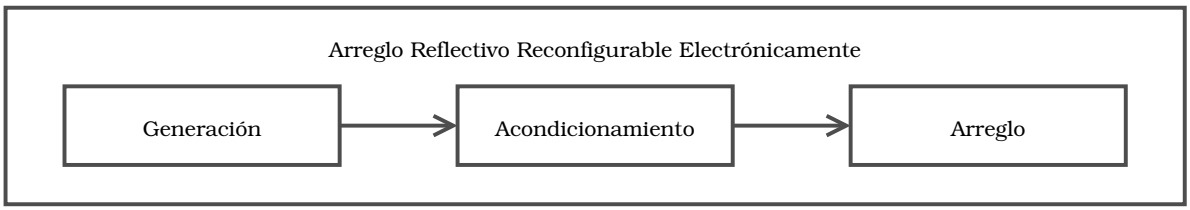


Figura 3.1: Esquema básico de un arreglo reflectivo reconfigurable electrónicamente.

Por su parte, los elementos conmutadores más utilizados son los diodos p-i-n, ya que se trata de una tecnología madura que presenta un buen desempeño en RF y Microondas. Sin embargo, estos requieren niveles de voltaje y corriente distintos a los que provee el dispositivo del bloque generador. Uno de los diodos que se utilizan en el grupo de trabajo es el HPND4005 [53], el cual se polariza en forma directa con máximo 1 [V], condición para la cual fluyen 20 [mA], mientras que polarizando en inversa a -30 [V] fluyen 100 [nA].

Otra tecnología prometedora es la de los MEMS, con la cual se pueden desarrollar interruptores con características sumamente atractivas, como voltajes de actuación de unidades de volt y consumo de corriente virtualmente cero. Aún cuando esta tecnología se encuentra en desarrollo, recientemente el grupo de trabajo ha logrado fabricar y accionar interruptores RF-MEMS. En caso de consolidarse el método de fabricación, estos interruptores podían integrarse monolíticamente al arreglo, diseñando sus características de voltaje y corriente para acoplarlas a los niveles lógicos de la etapa de generación, logrando así minimizar o incluso eliminar la etapa de acondicionamiento.

Para cualquier caso en que el conmutador no se acople a las características de las señales lógicas, será necesario implementar un bloque de *acondicionamiento* que realice dicha función. Una opción para implementar el acoplamiento de las señales es el uso de interruptores de un polo y dos Ttiros SPDT (*Single Pole, Double Throw*) controlados electrónicamente. De esta forma, la señal lógica proveniente del FPGA puede conmutar entre las dos fuentes de voltaje que requiere el diodo p-i-n: 1 [V] en polarización directa y -30 [V] en inversa, como se aprecia en la figura 3.2. Dichos dispositivos actualmente se encuentran disponibles en el mercado en el rango de unidades de dolar.

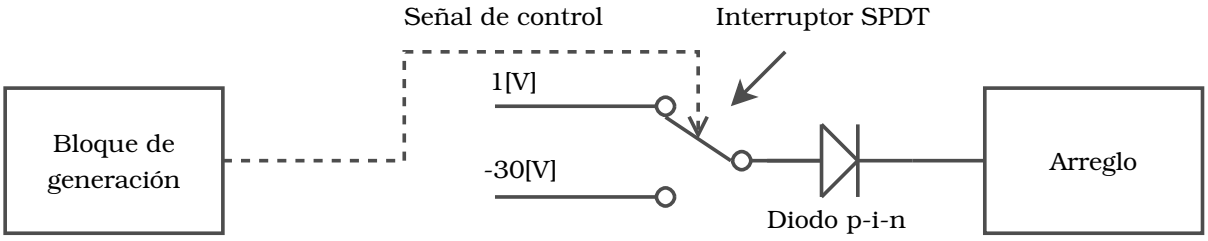


Figura 3.2: Interruptor un polo, dos tiros (SPTD).

3.1.1. La Distribución de Fase

Considérese el sistema de coordenadas que se muestra en la figura 3.3. La distribución de fase progresiva en la superficie de un arreglo reflectivo que produce un haz en la dirección (ϕ_h, φ_h) se calcula de la siguiente forma [14]:

$$\phi(x_i, y_i) = -k_0 sen(\theta_h) cos(\varphi_h) x_i - k_0 sen(\theta_h) sen(\varphi_h) y_i \tag{3.1}$$

donde k_0 es la constante de propagación en el vacío y (x_i, y_i) las coordenadas del elemento *i*.

Por otra parte, la fase del campo reflejado en cada elemento es igual a la fase del campo incidente que resulta de la propagación desde la antena alimentadora, más la fase introducida por cada celda:

$$\phi(x_i, y_i) = -k_0 d_i + \phi_R(x_i, y_i) \tag{3.2}$$

donde $\phi_R(x_i, y_i)$ corresponde al cambio de fase introducido por el elemento *i*, y d_i es la distancia que existe entre el centro de fase de la antena alimentadora y el elemento *i*, la cual se conoce como *distancia focal*. De las ecuaciones 3.1 y 3.2 se obtiene el cambio de fase necesario en cada elemento:

$$\phi_R(x_i, y_i) = k_0\{d_i - [x_i cos(\varphi_h) + y_i sen(\varphi_h)]cos(\theta_h)\} \tag{3.3}$$

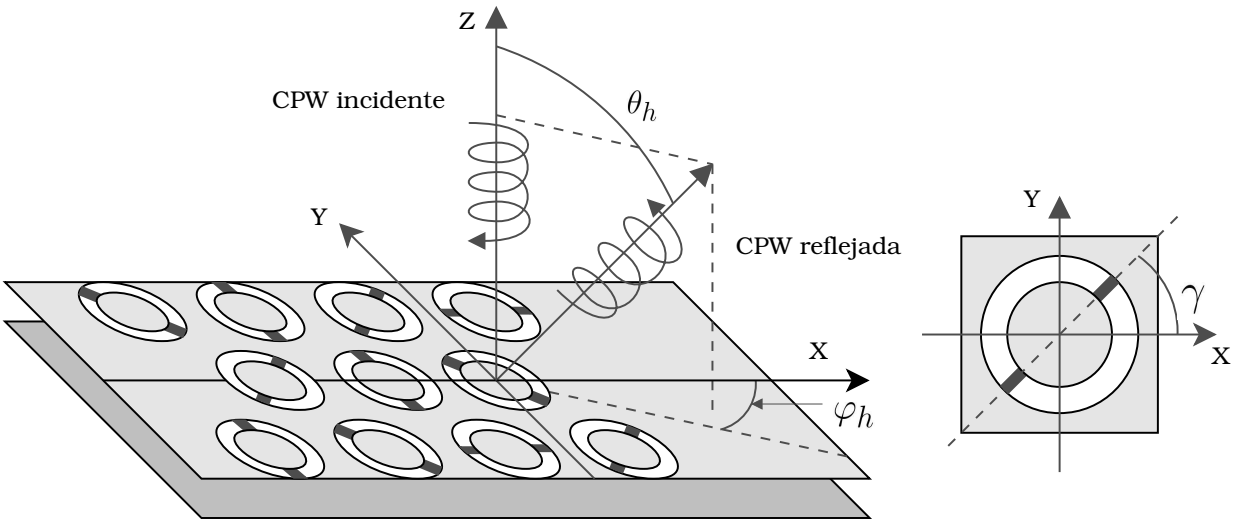


Figura 3.3: Esquema de un arreglo reflectivo.

Una onda CPW que incide sobre el elemento *i-ésimo* del arreglo se refleja con un cambio de fase igual a 2γ, donde γ representa la posición angular tal elemento (figura 3.3), misma que se obtiene de la siguiente forma:

$$\gamma_i = \frac{\phi_R(x_i, y_i)}{2} \tag{3.4}$$

El resultado de la ecuación anterior tiene un rango continuo de 0° a 180°. Sin embargo, la reconfiguración electrónica en el arreglo se obtiene introduciendo un cierto número de interruptores que permiten la conmutación entre posiciones angulares discretas. En la figura 3.4 se muestra la posición angular de las cargas en un anillo de dos bits, así como los *valores frontera* que definen los rangos de γ_i correspondientes a cada carga. El criterio es el siguiente:

- [0,22.5)° ∪ [157.5,180]° : carga ubicada a 0°.
- [22.5,67.5)°: carga ubicada a 45°.
- [67.5,112.5)°: carga ubicada a 90°.
- [112.5,157.5)°: carga ubicada a 135°.

Cabe destacar en este punto que uno de los parámetros más importantes para el diseño de la arquitectura es el mínimo *incremento en la dirección del haz* que se podrá manejar (Δ_h), el cual se forma por el mínimo incremento en los ángulos de elevación ($\Delta\theta_h$) y azimut ($\Delta\varphi_h$). La determinación de estos parámetros depende fuertemente de la aplicación y del *ancho del haz* que es capaz de producir el arreglo. Sin embargo, contemplando que el propósito inicial de este arreglo es académico, se emitió la recomendación de que es sistema soportara un incremento de 1° en cada parámetro, es decir:

$$\Delta\theta_h = \Delta\varphi_h = 1^\circ$$

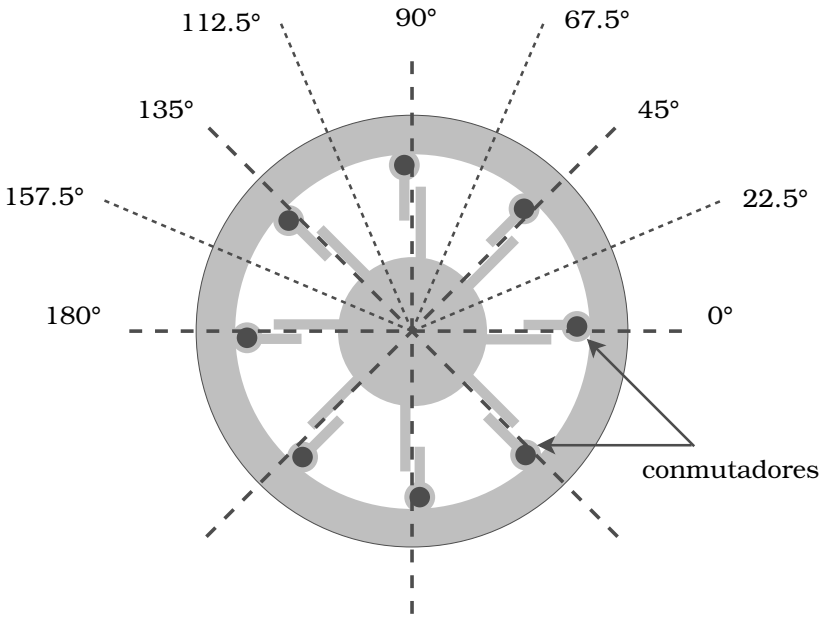


Figura 3.4: Posición angular de las cargas reactivas en un anillo de dos bits.

3.1.2. Selección del Arreglo

El diseño de la arquitectura se basó en el arreglo publicado en el 2015 por el grupo de trabajo de RF [28], el cual está conformado por 367 elementos radiadores distribuidos en una red triangular equilátera de 4.3 [mm] de lado (figura 3.5). Se fabricaron cuatro arreglos fijos configurados para ángulos de elevación θ_h de 0°, 20°, 40° y 60° y ángulo de azimut φ_h = 0° operando a 36.5 [GHz].

La disposición geométrica del arreglo y la antena alimentadora se muestra en la figura 3.6, donde se aprecia el centro de fase ubicado a 10.801 [cm] del centro del arreglo; y dada la distribución triangular de los elementos, el centro del mismo coincide con el de un anillo (coordenadas x = 0, y = 0), por lo cual se afirma que la *distancia focal* de dicho elemento es 10.801 [cm].

3.2. Herramientas de Diseño

Como se vio en la sección 2.9, la opción de diseñar un ASIC se encuentra fuera de los alcances de este proyecto, mientras que el uso de la lógica programable se presenta como una alternativa ideal para diseñar y probar la arquitectura de un procesador. Por esta razón, el diseño se realiza utilizando el lenguaje de descripción de hardware VHDL, ya que se trata de un estándar de la IEEE que permite el uso de PLDs de cualquier fabricante, tal como se mencionó en la sección 2.9.2. Por su parte, el programa que se utilizará para tal efecto es la versión estudiantil de Quartus II, el cual permite la realizar la síntesis del diseño en los dispositivos de Altera.

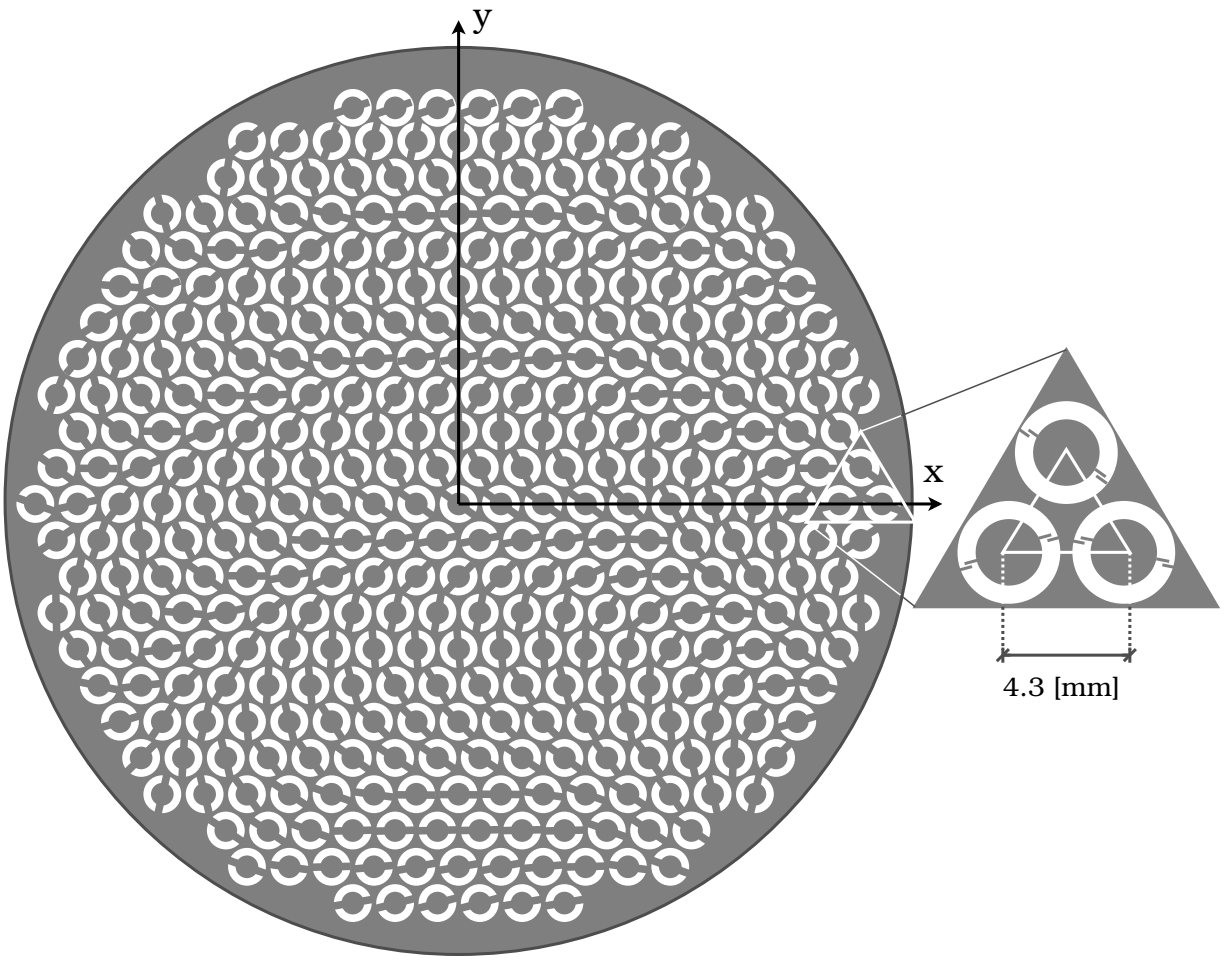


Figura 3.5: Arreglo de 367 anillos distribuidos en una red triangular y configurados para reflejar a la dirección normal.

Con base en los niveles de abstracción presentados en la sección 2.10, el diseño del procesador dedicado sigue un flujo *de arriba hacia abajo* para determinar los bloques básicos en el nivel lógico y, posteriormente, se fluye *de abajo hacia arriba* para conseguir el funcionamiento deseado del sistema en el nivel de la arquitectura.

Por último, el formato numérico (sección 2.2) resulta ser un parámetro de diseño importante, ya que tiene un impacto considerable sobre el consumo de recursos lógicos que refleja el sistema. Buscando la aplicación del paralelismo a nivel de núcleos (sección 2.5), se procuró que tal consumo fuese el menor posible, razón por la cual se optó por utilizar el formato de punto fijo. El ancho de palabra se eligió de 17 bits, mismo que permite una precisión máxima de 15 bits para la parte fraccionaria (q15), dejando un bit para el signo y otro para la parte entera. Esto resulta especialmente útil en la representación de los valores de la función sinusoidal. Por su parte, la precisión numérica se eligió de 15 bits para la parte fraccionaria con base en el buen desempeño que mostró dicho parámetro en las pruebas realizadas en microcontroladores comerciales, mismas que se presentan en la sección 4.5.

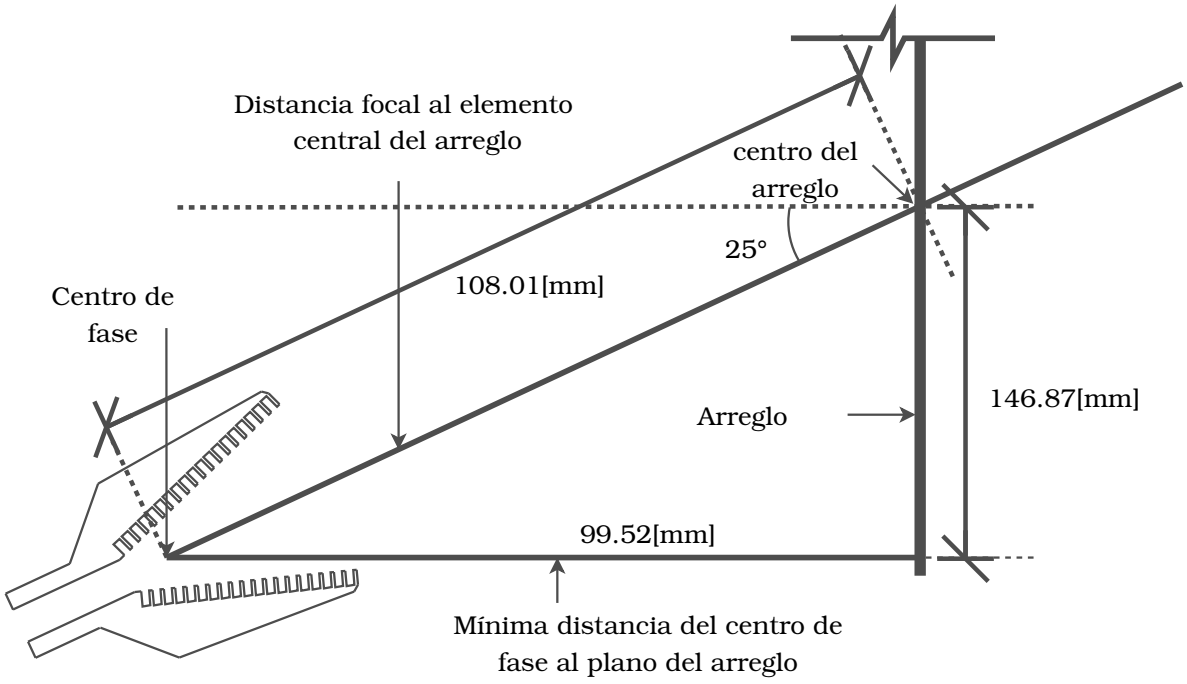


Figura 3.6: Disposición geométrica del arreglo reflectivo fabricado en [28].

3.3. Proceso de Diseño

Considerando que prácticamente cualquier aplicación se beneficia por un tiempo de cálculo pequeño (sección 1.10.1), se optó por el diseño de la arquitectura de un procesador que facilite la implementación del paralelismo a distintos niveles, buscando optimizar la arquitectura del procesador a nivel del hardware para obtener un cálculo eficiente de la posición angular de los elementos radiadores.

Por otra parte, en la sección 2.4 se mencionó que una computadora cuenta con tres bloques fundamentales: el procesador (CPU), la memoria y entrada/salida. Para el caso del sistema que se pretende diseñar, la estructura del bloque de entrada/salida es conceptualmente sencilla:

la entrada de los ángulos de elevación y azimut codificados en binario puro y la salida de las señales que gobernarán el comportamiento de los elementos conmutadores (en principio, un pin de salida por cada conmutador). En cuanto a la memoria, resulta conveniente que los bloques necesarios queden inmersos dentro de la unidad de procesamiento, haciendo más versátil el sistema, ya que una de las operaciones que más tiempo consumen en los sistemas de cómputo son la escritura y lectura de memoria externa [46].

De igual forma, se mencionó que una computadora es un dispositivo capaz de realizar una gran cantidad de tareas dividiéndolas en fragmentos elementales. Para dar solución a la reconfiguración electrónica, la tarea a ejecutar es la resolución de la ecuación 3.4. A partir de este punto, el diseño de la arquitectura capaz de realizar dicha tarea implica los siguientes pasos:

- Seleccionar el conjunto de instrucciones.
- Diseñar los bloques elementales de hardware que permitirán la ejecución de dichas instrucciones.

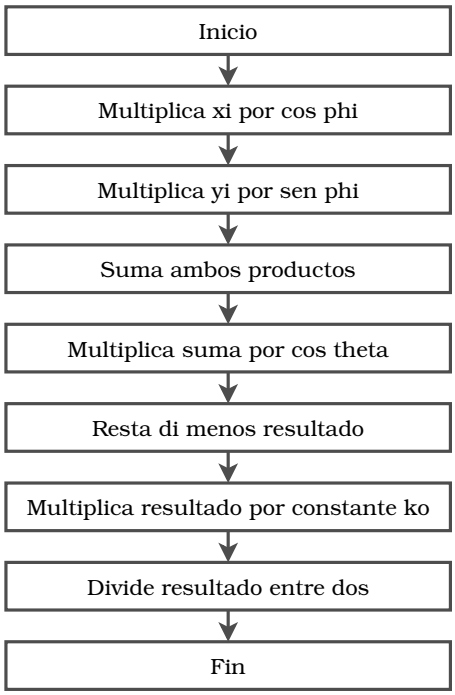


Figura 3.7: Diagrama de flujo del programa simplificado.

3.4. Determinación del Conjunto Instrucciones

Para seleccionar las instrucciones que ejecutará un procesador es necesario conocer las tareas que será capaz de realizar. Usualmente los procesadores comerciales se diseñan con una filosofía de propósito general, lo cual los habilita para realizar una amplia variedad de

funciones. Sin embargo, en este caso se utiliza la filosofía de propósito específico: la reconfiguración electrónica del arreglo. Para ello, se comienza con una inspección de la ecuación 3.3, lo cual podría sugerir que únicamente son necesarias operaciones aritméticas: suma, resta, multiplicación y división. La secuencia de instrucciones, o programa, que daría solución al problema se podría representar como en la figura 3.7.

Sin embargo, para resolver prácticamente cualquier problema, el procesador debe contar con instrucciones de *transferencia* que permitan almacenar los ángulos de entrada y resultados intermedios, así como instrucciones de *control de flujo* que permitan repetir el cálculo para todos los elementos del arreglo y para distintas combinaciones de ángulos de elevación y azimut. De otra forma, el sistema sería incapaz de cumplir con su objetivo.

Atendiendo dichas consideraciones, el esquema de un programa más funcional se muestra en a figura 3.8, donde se puede observar la carga de los datos necesarios para realizar el cálculo de todos los elementos del arreglo, así como la posibilidad de repetirlo con una nueva configuración mediante el regreso al inicio del programa. A partir de este diagrama, se propone el siguiente conjunto de instrucciones agrupado por categorías.

3.4.1. Control de Flujo

- *Salto condicional:* realiza la evaluación de alguna bandera o señal de control. Si es '0', el control del programa se mantiene en esta instrucción, mientras que al evaluarse '1' permite que el flujo del programa continue.
- *Salto incondicional*: El flujo del programa se modifica incondicionalmente.
- *Operación vacía*: es una instrucción que no realiza operación alguna. Se utiliza para hacer el manejo de riesgos por software (sección 2.8.2) y verificar así el funcionamiento de la arquitectura entes de implementar el control de riesgos por hardware.

3.4.2. Lógico-Aritméticas

- *Incrementa Registro*: ejecuta un incremento de una unidad al registro en cuestión. Se aplica en los registros de las coordenadas (x,y), así como el registro de distancia focal, para extraer los valores correspondientes a cada radiador de la memoria de elemento.
- *Corrimiento lógico*: realiza el corrimiento lógico de un registro. Se utiliza para ajustar el formato numérico y para implementar la división entre dos.
- *Multiplicación con corrimiento*: realiza la multiplicación de dos operandos y, dependiendo del formato de los mismos, realiza una operación de corrimiento que permite mantener el formato del resultado.
- *Suma*: ejecuta la suma de dos datos.
- *Resta*: realiza la resta de dos cifras.

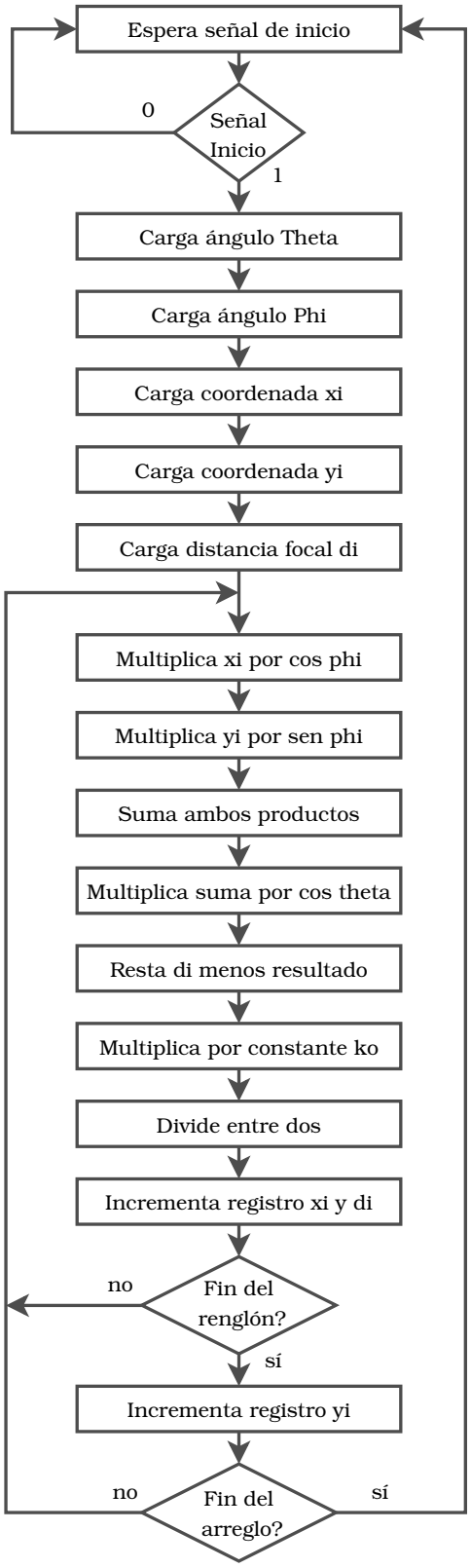


Figura 3.8: Diagrama de flujo del programa funcional.

3.4.3. Transferencia

– *Carga registro*: permite almacenar datos en los registros, como son los ángulos de elevación y azimut provenientes del exterior, así como datos de memoria mediante alguna forma de direccionamiento (sección 2.7.1).

– *Almacena registro*: permite almacenar el contenido de un registro, ya sea a la memoria de resultados o nuevamente a un registro.

3.5. Diseño del Hardware

El último parámetro de diseño del conjunto de instrucciones afecta fuertemente al diseño del hardware: la filosofía a seguir. En la sección 2.7.2 se mencionó que la RISC naturalmente permite la emulación del paralelismo mediante técnicas como la segmentación encausada, debido a que las instrucciones cuentan con una complejidad simplificada y con el mismo número de estados. Buscando el mejor desempeño posible, se optó por utilizar el cómputo RISC con el objetivo de implementar un procesador encausado.

A partir de este punto es posible comenzar con el diseño de los bloques elementales que conformarán el hardware de la arquitectura, para lo cual es necesario elegir el paradigma de descripción a utilizar. Existen tres principales [44], de los cuales el más adecuado para trabajar en el nivel de abstracción de micro-arquitectura (sección 2.10) es el de *descripción del comportamiento*. A su vez, en la conjunción de los bloques para formar la arquitectura del procesador resulta apropiado el uso del paradigma *estructural*. En el apéndice B se presenta el diseño de algunos bloques básicos descritos en VHDL.

Determinar todos los elementos con los que debe contar el procesador resulta un proceso complejo. Sin embargo, comenzando por el comportamiento básico de las etapas del cauce, es posible revelar los *bloques* necesarios para conseguir el comportamiento global del sistema. Los nombres de los mismos hacen referencia a la figura 3.9.

3.5.1. Traer la Instrucción

Naturalmente los primeros bloques surgen en esta etapa. Lo que se desea en principio es ejecutar una instrucción. La primera pregunta que podría surgir es ¿Dónde se encuentra dicha instrucción? La respuesta yace en la *memoria del programa*, donde se almacena el *CoOp* en la forma de un código binario. El siguiente bloque se intuye al momento de considerar que tras la ejecución de una instrucción, se desea continuar con otra. Para controlar la secuencia de instrucciones a ejecutar y, de esta forma, el comportamiento del procesador, se utiliza un *secuenciador* (sección 2.6.2).

3.5.2. Decodificación

Prácticamente el único parámetro que recibe esta etapa de la anterior es el CoOp de la instrucción en curso. Y aunque parezca simple, este parámetro es el que dará vida al resto de la arquitectura ¿Cómo? Mediante el bloque de control (*ctr*), el cual es básicamente un decodificador que genera todas las señales que gobernarán el comportamiento del procesador para ejecutar la acción indicada.

En cuanto a la memoria, generalmente se cuenta con una de propósito general (lectura/escritura). No obstante, la aplicación sugiere el uso de una memoria ROM para almacenar los parámetros propios de cada elemento radiador: sus coordenadas y su distancia focal. Estos datos se almacenan en el bloque *MemElem*.

Otra de las necesidades del sistema, como se aprecia en la ecuación 3.3, es el cálculo de funciones trigonométricas. Uno de los algoritmos más eficientes para realizar dicho cálculo es el CORDIC (*COrdinate Rotation Digit Computer*), el cual se basa en la rotación de un vector en un plano [54]. Sin embargo, este resulta costoso en cuanto al consumo de recursos lógicos. Para evaluar cuantitativamente este consumo, se utilizó la librería de megafunciones IP (*Intellectual Property*) de Quartus II [55] para sintetizar un bloque CORDIC capaz de calcular la función seno y coseno con una precisión de ocho bits para la parte fraccionaria (q8) y una latencia de dos ciclos de reloj para conseguir el resultado. Dicho bloque reflejó un consumo de 1,301 Elementos Lógicos para un FPGA de la familia Cyclone IV de Altera.

El uso de esta alternativa requiere evaluar tanto las necesidades como los recursos disponibles. En el caso del sistema que se planea diseñar, uno de los objetivos es obtener un bajo consumo de recursos lógicos, por lo que una alternativa más atractiva es el uso de una memoria para almacenar los valores de la función sinusoidal (*MemSeno*). El tamaño de esta memoria se define por el incremento de la dirección del haz Δ_h, el cual se propuso de 1° para ambos ángulos en la sección 3.1.1. Otro parámetro que impacta al tamaño de esta memoria es el rango de los ángulos, donde el mayor es el de φ_h: [0°,360°). Sin embargo, las propiedades de la función seno y coseno permiten que sea posible almacenar únicamente el rango de [0°,90°] para una de ellas y, codificando el cuadrante de φ_h, obtener el valor adecuado de ambas funciones a partir de los valores disponibles.

Al igual que sucedió con la memoria del programa, es necesario contar con un bloque que permita transitar por las direcciones disponibles. Para ello se implementa un banco de registros (*reg*), el cual además de contar con los necesarios para direccionar las memorias anteriores, contiene un registro acumulador y uno auxiliar dedicados al almacenamiento de los resultados intermedios.

Contemplando que en la etapa de ejecución únicamente se realizarán operaciones sobre dos datos en un determinado momento, es necesario utilizar multiplexores que permitan seleccionarlos de acuerdo a la instrucción en curso.

3.5.3. Ejecución

Una vez que los operandos están listos, en esta etapa se llevan a cabo las instrucciones. Para ello, se cuenta con la unidad lógico-aritmética (*alu*), la cual permite realizar las operaciones de suma, resta, multiplicación y corrimientos lógicos, así como la generación de banderas de estado. Para realizar modificaciones a los registros, los cuales son de ocho bits, se cuenta con el bloque *alr*. Tales unidades también permiten transportar un dato de la etapa dos a la cuatro sin realizar operación alguna sobre el mismo.

Suponiendo que para solucionar el problema fuera suficiente un programa como el que se mostró en la figura 3.7, la *alu* podría ser el único bloque de esta etapa. Sin embargo, como se evidenció en la sección 3.4, son indispensables las instrucciones de control de flujo, por lo que resulta necesario contar con un bloque de saltos que permita evaluar las banderas de estado para modificar el flujo del programa cuando sea necesario (*bch*).

3.5.4. Escritura

En esta etapa se realiza la transferencia de datos hacia los registros, tanto de resultados intermedios, como la codificación final de la carga reactiva en el anillo. También se obtiene la resolución de las instrucciones de salto y la dirección que modificar el flujo del programa en caso de que se ejecute.

Por otra parte, se contempla el uso de una memoria de anillos (*man*), la cual permite almacenar grupos de cuatro bits por anillo. Mapeando cada uno de estos bits a un pin de salida, se obtienen finalmente las líneas de control que gobiernan el comportamiento de cada diodo p-i-n presente en los 367 radiadores del arreglo.

3.6. Refinamiento y optimización del diseño

Una vez que se cuenta con el bosquejo de las etapas, es necesario acoplarlas con registros que permiten almacenar la información generada para su uso en estados y etapas posteriores, tal como se mencionó en la sección 2.8. Posteriormente se busca conseguir la ejecución de una instrucción, revelando así las modificaciones y ampliaciones a los bloques existentes, así como la necesidad de implementar nuevos elementos y señales de control.

Al ejecutar una instrucción exitosamente, se busca conseguir la ejecución de un conjunto de ellas, situación que tarde o temprano revela la presencia de los riesgos inherentes de la segmentación encausada (sección 2.8.2), por lo que es necesario implementar la unidad de detección de riesgos (*udr*), cuya función es evaluar las banderas que los revelan y generar las señales de control que permitan solucionarlos, ya sea utilizando adelantos o detenciones.

Dentro de este proceso de refinamiento y optimización surgió, por ejemplo, el bloque de codificación del cuadrante de φ_h (*ccp*). Inicialmente se había contemplado que esta codificación se realizara mediante una instrucción, de la misma forma que se implementaría si se utilizara la arquitectura fija de un microcontrolador comercial. Sin embargo, explotando la flexibilidad que provee la descripción de hardware, se consideró la posibilidad de codificar el ángulo φ_h mediante un bloque ubicado en la etapa de decodificación, logrando así determinar el cuadrante del ángulo durante la carga de dicho parámetro, ahorrando de esta forma instrucciones adicionales.

Una vez que se consiguió la ejecución de las instrucciones necesarias para calcular la posición angular de un elemento, se presentó la necesidad de aproximarla a alguna de las posiciones disponibles (sección 1.10). Esta situación resulta semejante a la del bloque codificador del cuadrante de φ_h, por lo que en lugar de implementar una instrucción adicional, se incorporó el bloque codificador de la posición angular (*cpa*). La salida de este bloque es la que finalmente se almacena en la memoria *man* de la cuarta etapa.

A partir de este punto, se contaba con una arquitectura capaz de calcular las cuatro señales que controlarían los conmutadores del primer anillo del arreglo. Y fue al momento de buscar la repetición del ciclo para el segundo anillo que se reveló la necesidad de contar con un bloque capaz de recordar cuál de los elementos del arreglo se estaba calculando en determinado momento. Dicho bloque es el contador de elementos (*cte*), y resulta útil para determinar el momento en el que se ha llegado al final de un renglón, así como al final del arreglo, para ejecutar las tareas de control de flujo pertinentes.

Cuando el proceso de refinamiento ha abarcado la ejecución de toda la secuencia de instrucciones de la figura 3.8, se obtiene finalmente la descripción del hardware del procesador segmentado de cuatro etapas (figura 3.9) que permite reconfigurar el arreglo reflectivo de 367 radiadores de dos bits.

3.7. Expansión de Pines de Salida

El resultado final que se obtiene por parte de la arquitectura es una línea de control por cada conmutador. Esto implica que, una vez acondicionada, la señal que llegará finalmente al conmutador proviene de un pin del dispositivo huésped; en este caso, el FPGA donde se sintetice el diseño.

El arreglo cuenta 367 radiadores de dos bits, lo que equivale a 1,468 conmutadores. Por su parte, los FPGAs que cuentan con mayor número de pines disponibles como salidas son:

- La familia Virtex UltraScale de Xilinx con hasta 1,456 [56].
- La familia Statrix 10 de Altera con hasta 1,640 [57], la cual tiene fecha de lanzamiento comercial para finales del año 2016.

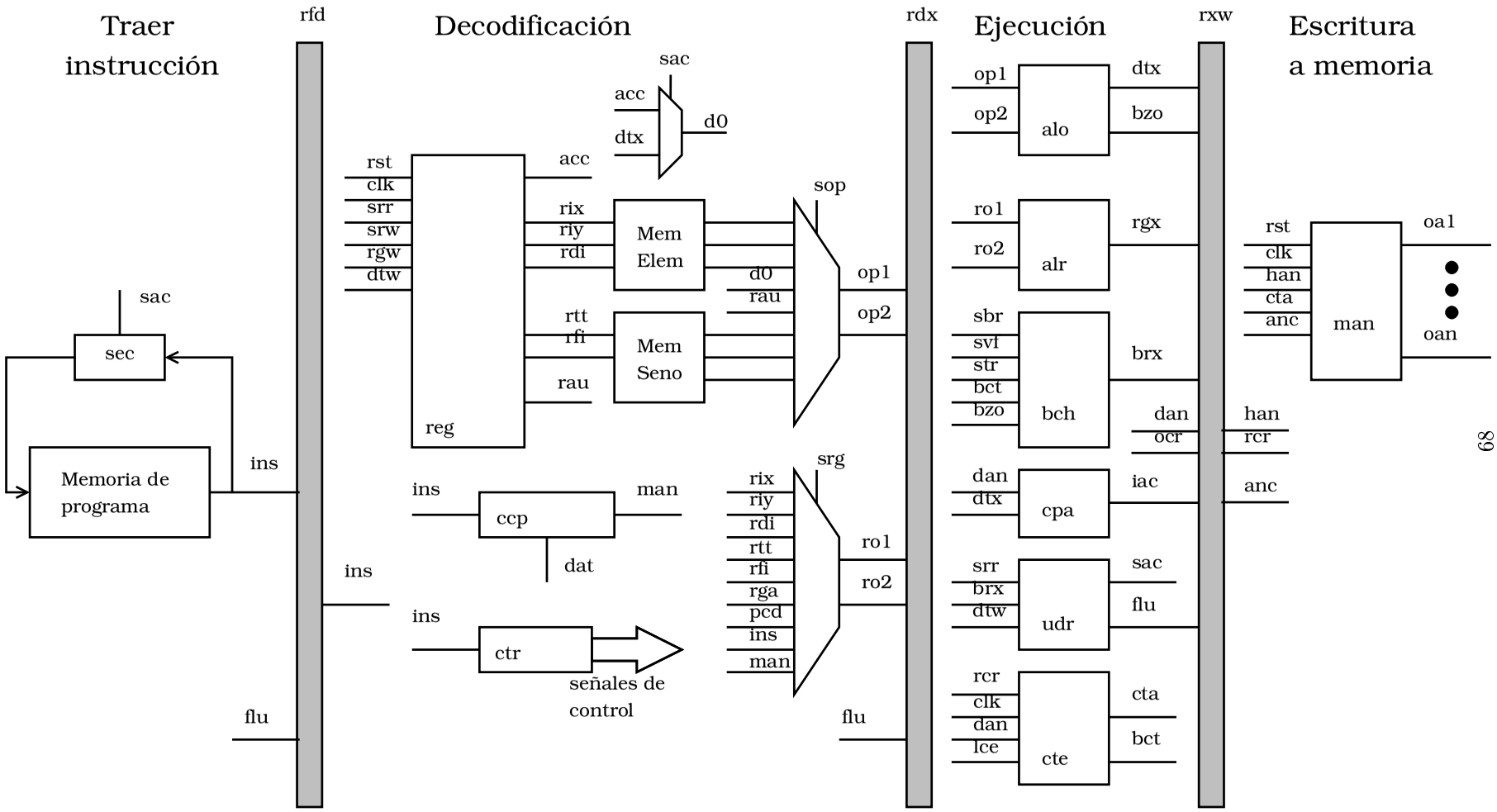


Figura 3.9: Diagrama a bloques del procesador diseñado.

Aún cuando 1,640 es una gran cantidad de pines disponibles, únicamente se pueden controlar arreglos con menos de 410 desplazadores de dos bits (410*4=1640), ya que es necesario restar los pines dedicados a la entrada de los ángulos y contemplar que, dependiendo del diseño, la herramienta de software que sintetiza el diseño en el FPGA puede llegar a determinar que no es posible asignar alguna salida a determinado pin. Un ejemplo de esta situación se ejemplifica en la sección 4.5.

Cabe destacar que esta limitante no es exclusiva de la solución propuesta en este trabajo. Prácticamente cualquier dispositivo requerirá el uso de alguna técnica para expandir los pines disponibles. Aún cuando los FPGAs son de los dispositivos que cuentan con mayor número salidas (pensando que el uso de la familia Statrix 10 sería suficiente para controlar el arreglo propuesto), arreglos de mayor número de elementos y/o un incremento en el número de bits del desplazador terminará por superar la oferta de pines de cualquier dispositivo. Por esta razón, se proponen las siguientes técnicas para la expansión de pines de salida.

3.7.1. Codificación Binaria

Como se aprecia en la figura 3.10, en todo momento el desplazador mantiene todos sus conmutadores en estado de no conducción excepto dos que se encuentran a 180° uno de otro (45° y 225° en la figura). En este caso, la misma línea de control puede encargarse del manejo de parejas de conmutadores opuestos. Por lo tanto, al igual que el desplazador con stubs radiales [22], únicamente se necesitan cuatro líneas de control (LCs).

Considérese que el diodo permanece en estado de baja impedancia cuando la línea de control está en nivel alto (LCna), y en alta impedancia cuando la línea presenta nivel bajo (LCnb), en la tabla 3.1 se observa que el comportamiento de las LCs para stubs radiales es el inverso para cargas capacitivas, lo que implica que al nivel del procesador únicamente es necesario negar las salidas para manejar ambos tipos de radiadores.

Impedancia de carga	Diodos/anillo	LCs	LCna	LCnb
Capacitiva	8	4	3	1
Inductiva	4	4	1	3

Tabla 3.1: Comportamiento de las líneas de control para anillos cargados capacitiva e inductivamente.

Por lo tanto, para desplazadores de dos bits, en cualquier instante de tiempo sólo una línea permanece en determinado nivel lógico, mientras que el resto de ellas permanecen en el nivel opuesto. Esto sugiere la posibilidad de implementar la codificación de los conmutadores a permanecer en estado de conducción mediante dos bits. Para recuperar las cuatro líneas de control basta con implementar un decodificador dos a cuatro para cada anillo, como se muestra en la figura 3.10. Debido a que el decodificador es un bloque lógico elemental, diversos fabricantes lo producen en masa y su costo se encuentra en el orden de centavos de dolar.

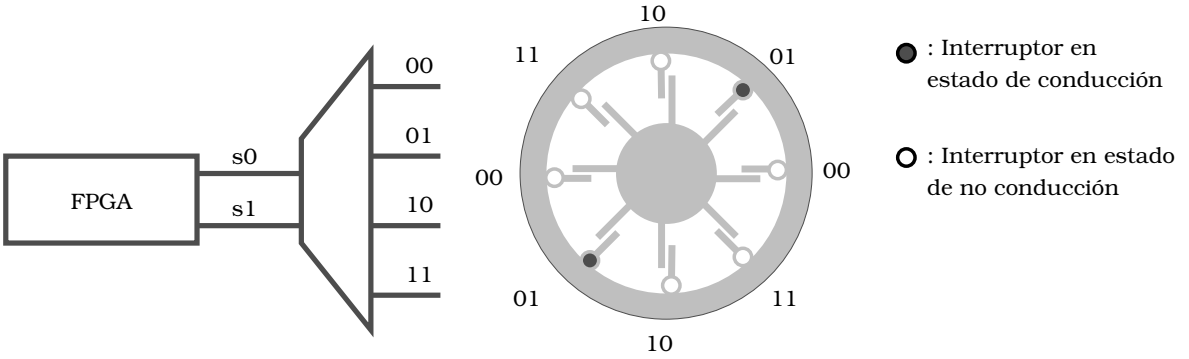


Figura 3.10: Decodificación del conmutador a permanecer en estado de alta impedancia.

Aplicando esta técnica en radiadores de dos bits, el mismo número de pines puede generar el doble de líneas de control. En general, la expansión de pines por codificación se puede expresar como:

$$lc_{(np)} = 2^n \tag{3.5}$$

donde lc corresponde a las líneas de control que se generan utilizando np pines, mientras que n es el número de bits del desplazador. Cabe destacar que el funcionamiento de esta técnica requiere que $n = np$, ya que todas las señales a la salida del decodificador, sin importar su tamaño, corresponden a las señales que gobiernan a los conmutadores de un único elemento radiador. Esto implica que no es posible, por ejemplo, utilizar un decodificador 4 a 16 para manejar cuatro anillos de dos bits.

En la figura 3.11 se muestra gráficamente el comportamiento para desplazadores de hasta cinco bits, donde puede apreciarse que esta alternativa resulta más eficiente mientras mayor es el número de bits del desplazador.

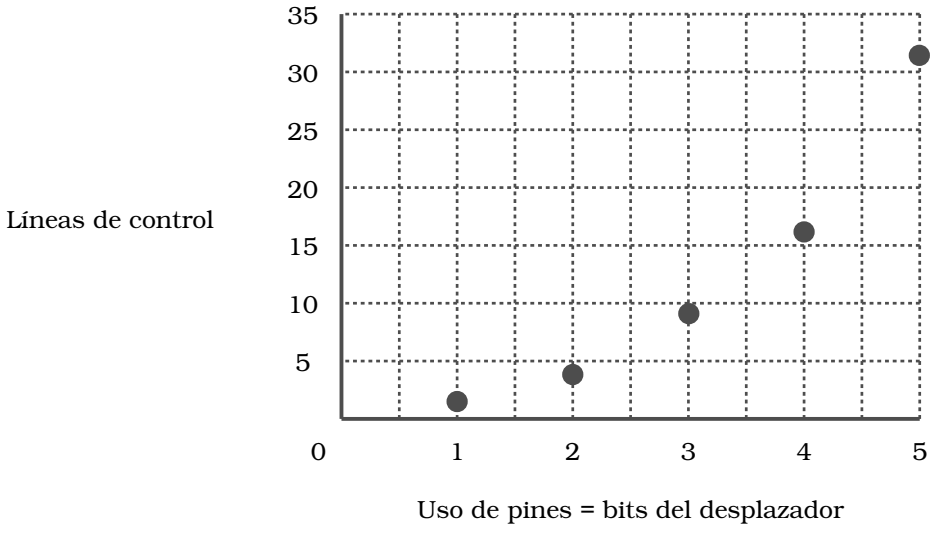


Figura 3.11: Comportamiento gráfico de la expansión de pines por codificación.

El diagrama a bloques de la etapa de generación de señales lógicas se presenta en la figura 3.12. El impacto que tiene esta técnica en el desempeño del sistema depende únicamente del tiempo de propagación de las señales a través del decodificador, el cual se encuentra en el orden de nanosegundos. Por ejemplo, el decodificador SN74LVC1G139 de Texas Instruments [58] tiene un tiempo de propagación máximo de 16.7 [ns] con alimentación de 1.8 [V].

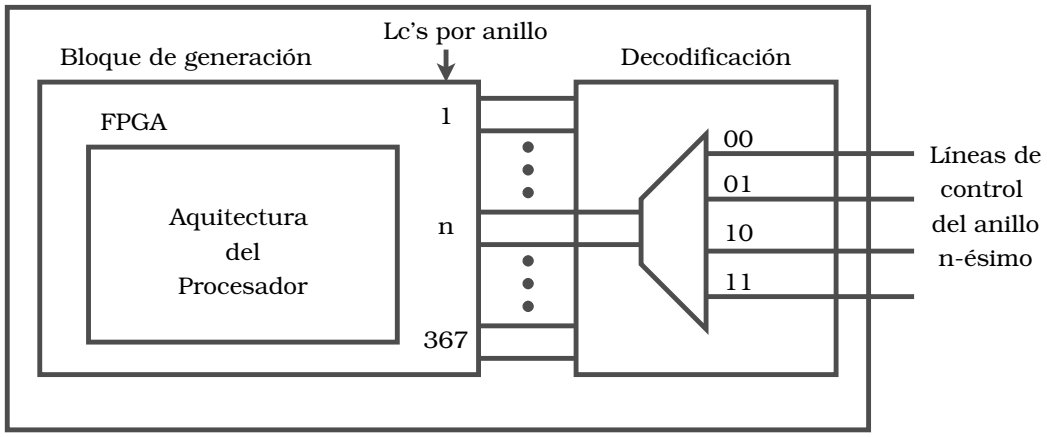


Figura 3.12: Bloque de generación con etapa de decodificación.

3.7.2. Registros de Corrimiento

Aún cuando la técnica de codificación resulta una interesante forma de expansión, se encuentra limitada a arreglos de máximo 820 elementos de dos bits para la familia Statrix 10. No obstante, los arreglos reflectivos pueden contener miles de ellos, razón por la cual es necesario contar con otra alternativa que permita ampliar aún más las capacidades del dispositivo.

Una de ellas es el uso de *registros de corrimiento*, de los cuales existe una variedad que implementa *entrada serie, salida paralelo*. Suponiendo un registro de 16 bits que utilice dos pines para la entrada serial del dato, entonces es posible obtener 16 líneas de control a partir de dos pines del FPGA.

Aunque esta opción podría parecer semejante al uso de un decodificador de 4 a 16, el registro tiene la ventaja de que las 16 líneas pueden corresponder a las líneas de control de cuatro radiadores de dos bits o a las 16 líneas de un radiador de cuatro bits.

Los registros de corrimiento *serie-paralelo* más comunes comercialmente son de 4, 8 y 16 bits. En general, la expansión de pines podría expresarse como:

$$lc_{(np)} = sp_{reg} \tag{3.6}$$

donde lc es el número de líneas de control que se obtienen con np pines que necesita el registro serie para comunicarse con el huésped, mientras que sp_{reg} es el número de salidas en paralelo del registro. Aquí se aprecia que las líneas de control son independientes del número de bits del desplazador.

En la figura 3.12 se muestra el comportamiento de la expansión utilizando dos pines del dispositivo huésped para comunicarse con registros de 4, 8 y 16 bits, donde se observa que la pendiente de expansión es mayor conforme mayor es el número de bits de la salida paralela del registro.

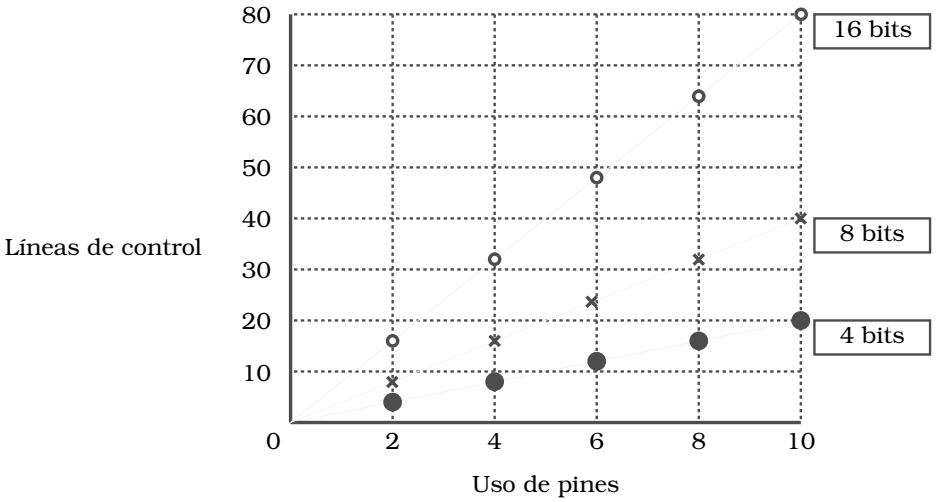


Figura 3.13: Expansión de pines mediante el uso de registros de 4, 8 y 16 bits.

Suponiendo el uso dos pines para establecer la comunicación serial con un registro de 16 bits y un FPGA de 206 pines disponibles, entonces con 103 registros se obtendrían ocho líneas más que las que se obtienen con un Statrix 10 sin expansión alguna, mientras que este último sería capaz de manejar un arreglo de 3,280 desplazadores de dos bits.

A diferencia de la técnica de codificación, el uso de registros implica un aumento en el tiempo de respuesta determinado por el retraso que genera la carga secuencial del registro. Éste queda determinado por la máxima velocidad de transferencia serial que se esté utilizando. En general:

$$t_{cr} = t_{cs} * n_{br} \tag{3.7}$$

donde t_{cr} es el tiempo de carga del registro, t_{cs} el periodo del reloj que gobierna la comunicación serial, y n_{br} el número de bits del registro. Utilizando una tasa de transferencia serial de 9,600 [bps] se carga un registro de 16 bits en 1.66 [ms], mientras que a una tasa de 128,000 [bps] se logra en 125 [μs].

3.7.3. Codificación más Registros

En caso de que ninguna de las opciones anteriores sea suficiente para determinado arreglo, cabe la posibilidad de implementar un sistema híbrido: cargar los registros de corrimiento con las señales codificadas en binario y utilizar decodificadores a la salida de éstos para recuperar las señales de control. Para tal solución:

$$lc_{(np)} = sp_{reg} * 2^n \tag{3.8}$$

donde lc son líneas de control que se obtienen a partir de np pines dedicados a la carga serial del dato al registro de corrimiento. Por su parte, sp_{reg} es el número de salidas paralelas del registro y n el número de bits del desplazador. En la figura 3.14 se aprecia el comportamiento gráfico de esta solución para anillos de dos bits y registros de 4, 8 y 16 bits.

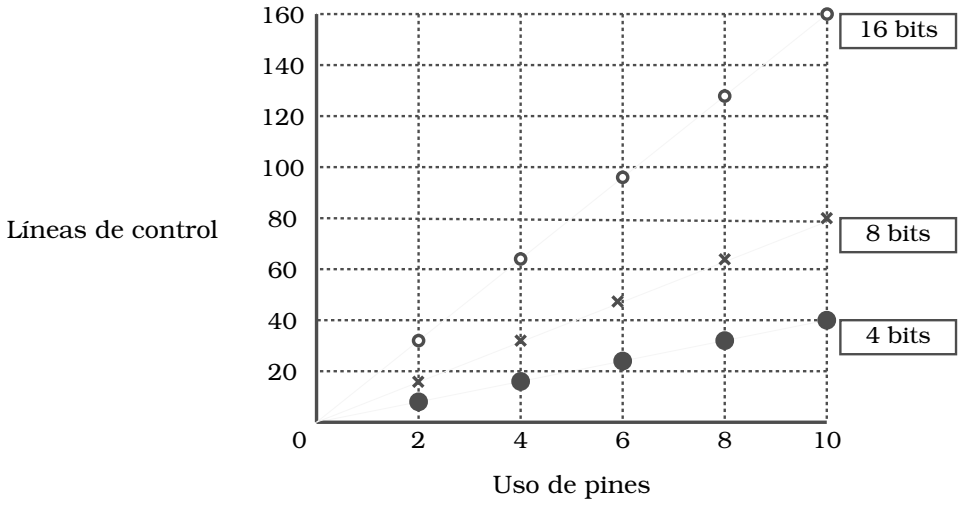


Figura 3.14: Expansión de pines para anillos de dos bits y registros de 4, 8 y 16 bits.

Suponiendo dos pines para comunicarse con un registro de ocho bits, entonces podrían manejarse cuatro anillos de dos bits por cada dos pines del dispositivo huésped, como se muestra en la figura 3.15. En este caso, un FPGA de 102 pines puede proveer ocho líneas de control menos que un Statrix 10 sin expansión, mientras que este último sería capaz de manejar un arreglo de 6,560 radiadores utilizando esta técnica.

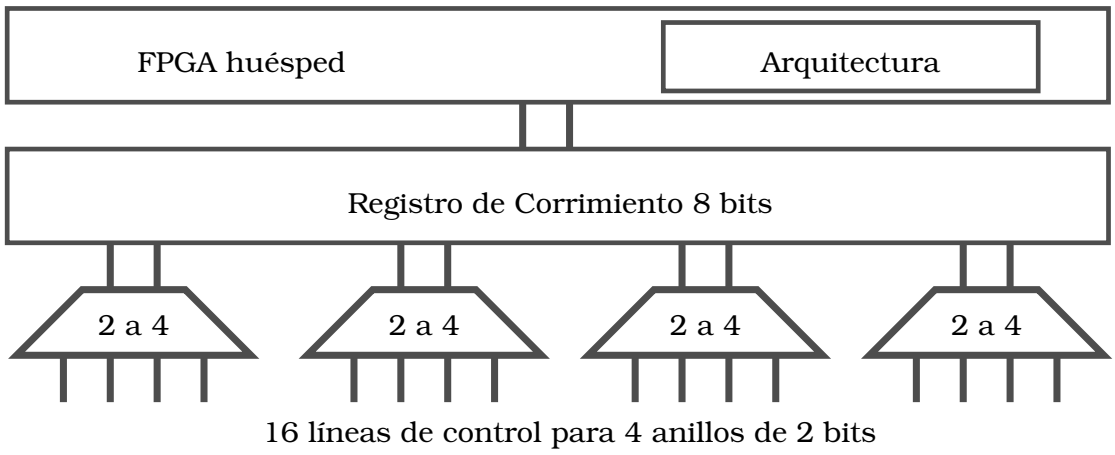


Figura 3.15: Expansión mediante registro de ocho bits y codificación a dos bits.

El impacto en el tiempo de respuesta que implica esta solución debe considerar la suma del retraso generado por la carga serial de los registros más el tiempo de propagación que introducen los dispositivos decodificadores.

3.7.4. Registros de Corrimiento Conectados en Serie

En caso de que el sistema híbrido no fuese suficiente para obtener la cantidad de líneas de control necesarias, aún hay otra opción. Existen registros de corrimiento que son capaces de manejar entrada serial y proveer salida tanto paralela como serial. Conectando la salida serie de uno a la entrada de otro, es posible emular un registro entrada serie, salida paralela de $n*m$ donde n es en número de registros conectados en serie y m el número de bits de la salida paralela de cada registro.

En la figura 3.16 se presenta un esquema para esta forma de expansión. Utilizando únicamente dos pines del FPGA es posible manejar un arreglo de registros conectados en serie. Considerando los tres registros de 16 bits visibles en la figura, se obtendría una expansión de 48 líneas de control a partir de dos pines, aunque es posible expandir dicho número mediante la conexión de más registros.

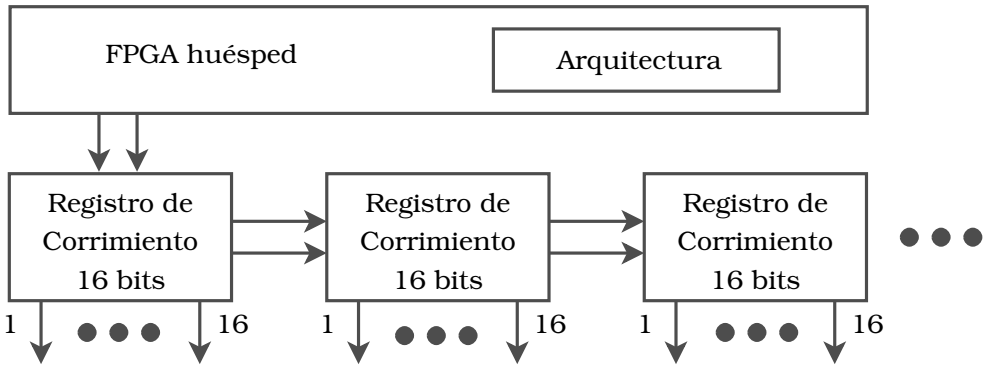


Figura 3.16: Expansión de pines por registros conectados en serie.

Lógicamente esta solución tiene un costo: el incremento en el tiempo de respuesta, ya que atendiendo a la fórmula 3.7 y una tasa de transferencia de 128,000 [bps], el tiempo que tomaría cargar un arreglo de registros capaz de generar el mismo número de líneas de control que el Statrix 10 sería de aproximadamente 12.81 [ms].

3.8. Escalabilidad en Tiempo de Respuesta

La creciente oferta de elementos lógicos disponibles dentro de los FPGAs ofrece la posibilidad de escalar el diseño para mejorar el tiempo de respuesta. Para conseguirlo en necesario apoyarse en el concepto de paralelismo a nivel de núcleos que se definió en la sección 2.5.

Considérese el procesador encausado que se presentó en en la figura 3.9 como un núcleo capaz de manejar el arreglo total (*nat*). Replicando dicho núcleo y asignándole a cada uno el cálculo de la mitad del arreglo, como se muestra en la figura 3.17, se genera un sistema *multinúcleo* que es capaz de reducir el TCH aproximadamente a la mitad.

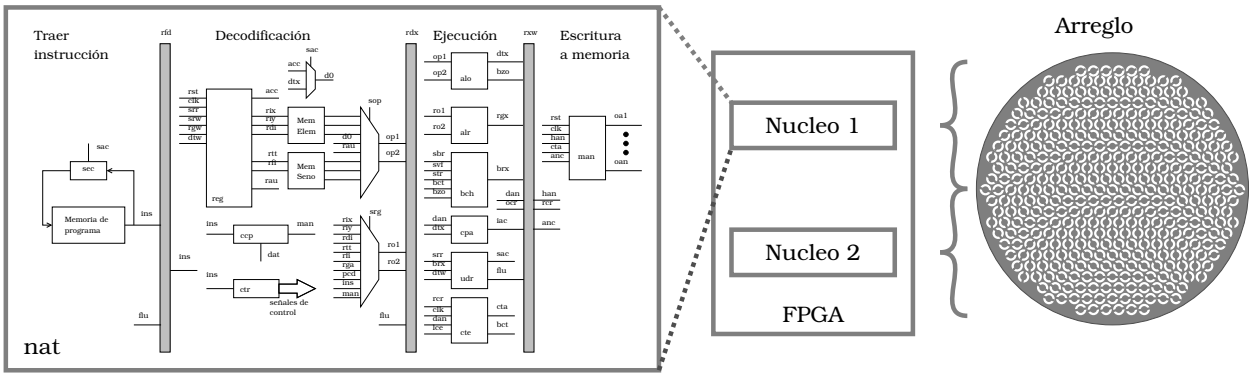


Figura 3.17: Manejo del arreglo con dos núcleos.

El número de núcleos que se pueden implementar depende de los recursos con los que cuenta el FPGA. En general, si se implementan n núcleos, se puede obtener una reducción en tiempo igual a:

$$T_n = T_1/n \tag{3.9}$$

donde T_n es el tiempo de respuesta con n núcleos y T_1 el tiempo de respuesta utilizando sólo uno. El comportamiento normalizado de la mejora en tiempo utilizando la filosofía multinúcleo se muestra en la figura 3.18.

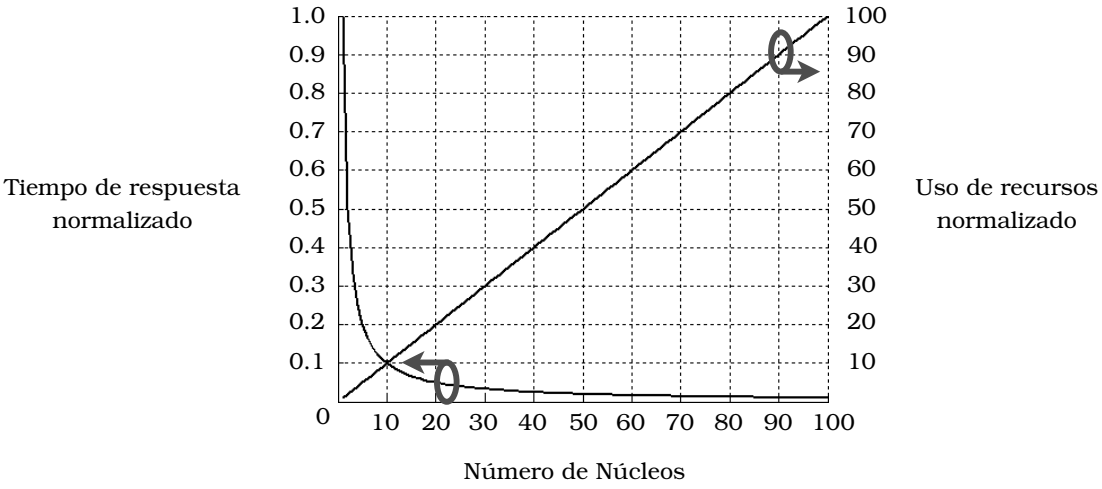


Figura 3.18: Escalabilidad mediante la técnica multinúcleo.

Naturalmente, esta técnica encuentra su límite al llegar al uso de un núcleo por anillo, explotando el paralelismo a nivel de núcleos al máximo y alcanzando así el mejor desempeño posible. No obstante, la desventaja de esta técnica radica en el consumo de recursos. La figura 3.18 revela el crecimiento normalizado del consumo de recursos del dispositivo, el cual es teóricamente lineal, aunque el comportamiento puede variar debido al proceso que ocupa la herramienta Quartus II para sintetizar el diseño, así como por las modificaciones que sufran los bloques de *MemElem* de cada núcleo. En cualquier caso, el límite quedará impuesto por las cantidad de recursos lógicos del dispositivo a utilizar.

Capítulo 4

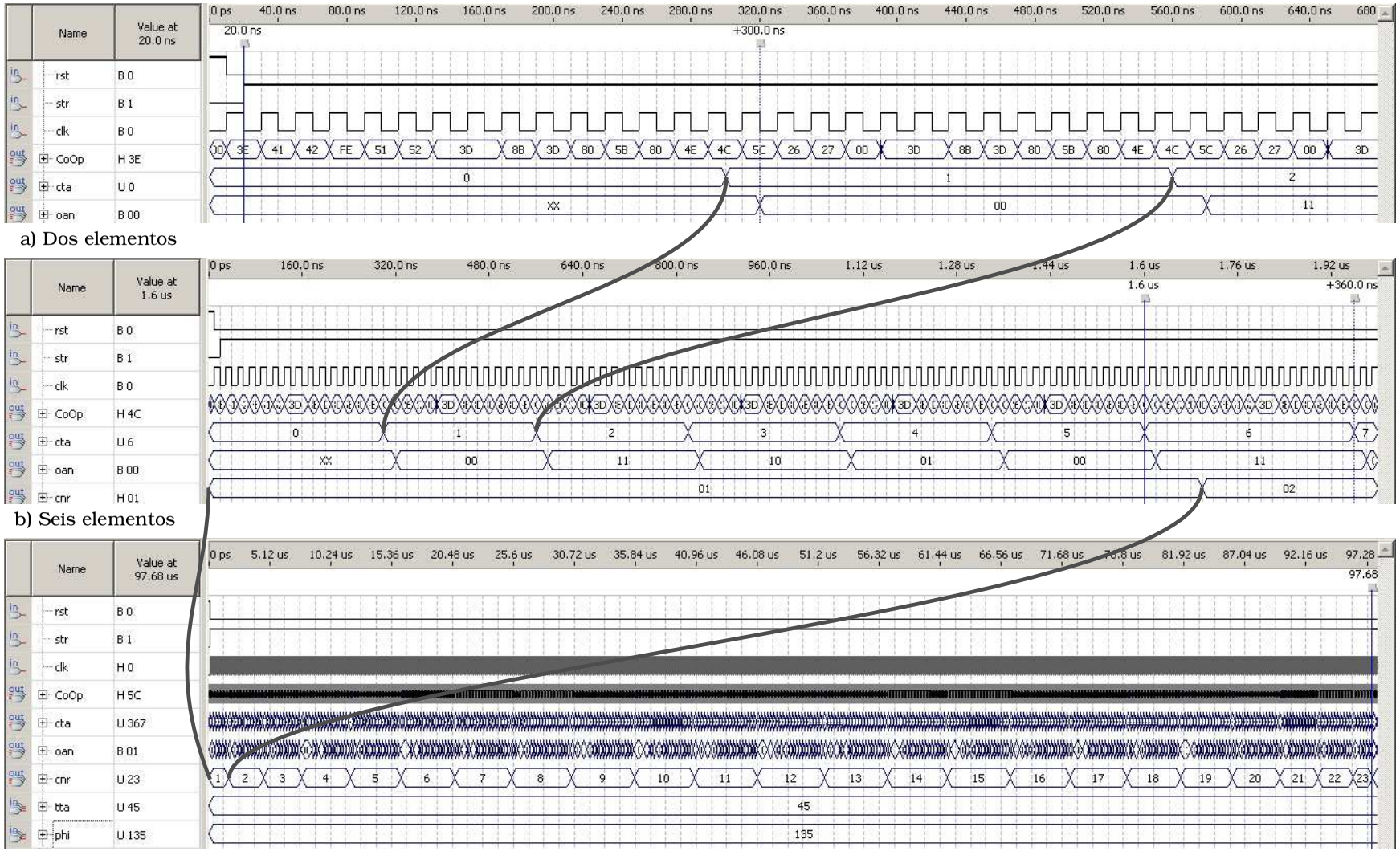
Resultados

En este capítulo se muestran los resultados del diseño de la arquitectura del procesador dedicado. Se presentan las simulaciones en el ambiente de Quartus II que permiten verificar el valor binario de las señales de control, así como el tiempo de ejecución del sistema. Mediante el uso del programa libre *Octave* se obtiene una visualización gráfica de todos los elementos. La verificación física de las señales de control se realiza sintetizando el diseño en una tarjeta de desarrollo DE2-115 de Terasic, la cual cuenta con un FPGA de la familia Cyclone IV de Altera. Buscando la incorporación del sistema a un arreglo reflectivo, se presenta una propuesta de tarjeta dedicada para tal efecto. Finalmente, se presentan las conclusiones y el trabajo a futuro.

4.1. Simulación en Quartus II

Naturalmente, la primera forma de verificación es la simulación que ofrece el ambiente de Quartus II, ya que el diseño de cada bloque y su funcionamiento, tanto individual como parte del sistema, se comprobó de la misma forma. En la figura 4.1 se muestra la simulación para la dirección de haz $\theta_h = 45$ y $\varphi_h = 135$, donde:

- *rst*: señal de reestablecimiento.
- *str*: señal de inicio.
- *clk*: señal de reloj de 50 [MHz].
- *CoOp*: código de la instrucción.
- *cta*: contador de elementos.
- *oan*: resultado codificado a dos bits.
- *cnr*: contador de renglones del arreglo.
- *tta*: ángulo de elevación θ_h en decimal.
- *phi*: ángulo de azimut φ_h en decimal.



a) Dos elementos

b) Seis elementos

c) 367 elementos

Figura 4.1: Simulación del arreglo configurado para reflejar a la dirección $\theta_h = 45$, $\varphi = 135$.

4.1.1. Tiempo de Respuesta

En la figura 4.1a se muestra que, una vez generado el pulso de inicio (*str*) a los 20 [ns] y con *rst* = *'0'*, el primer anillo se calculó en 15 instrucciones, las cuales a una frecuencia de reloj de 50 [MHz], equivalen a 300 [ns]. El cálculo del siguiente elemento no incluye las operaciones correspondientes a la carga de los ángulos θ_h y φ_h, por lo que toma únicamente 13 instrucciones, equivalentes a 260 [ns] . Por su parte, la figura 4.1b revela que el cálculo del séptimo elemento toma 18 instrucciones que equivalen a los 360 [ns] visibles a través de los cursores. Esta diferencia se debe a que el séptimo elemento es el primero del segundo renglón, lo que implica una demora debido a la ejecución de instrucciones de control adicionales al finalizar el cálculo del sexto elemento.

Finalmente, la figura 4.1c revela que el tiempo que le toma al procesador calcular los 23 renglones del arreglo (línea *cnr*) es de 97.68 [μs]. No obstante, esta pareciera ser la única información rescatable de tal figura, ya que las líneas de reloj, código de operación, resultado codificado y contador de elementos se encuentran completamente saturadas de transiciones. Sin embargo, está última línea (*cta*) revela un valor de 367 a través de la columna de valor a los 97.68 [μs] (*Value at 97.68 [μs]*), con lo que se muestra que el contador de elementos llegó al final del arreglo. De la misma forma, se observa que el código correspondiente al último elemento es *01*, lo que implica una posición angular de 45° para tal anillo.

Utilizando una señal de control que toma el valor de '1' cuando se ha completado el cálculo del último anillo, se midió en osciloscopio un tiempo de cálculo de 98 [μs] correspondiente al valor Δt en la figura 4.2.

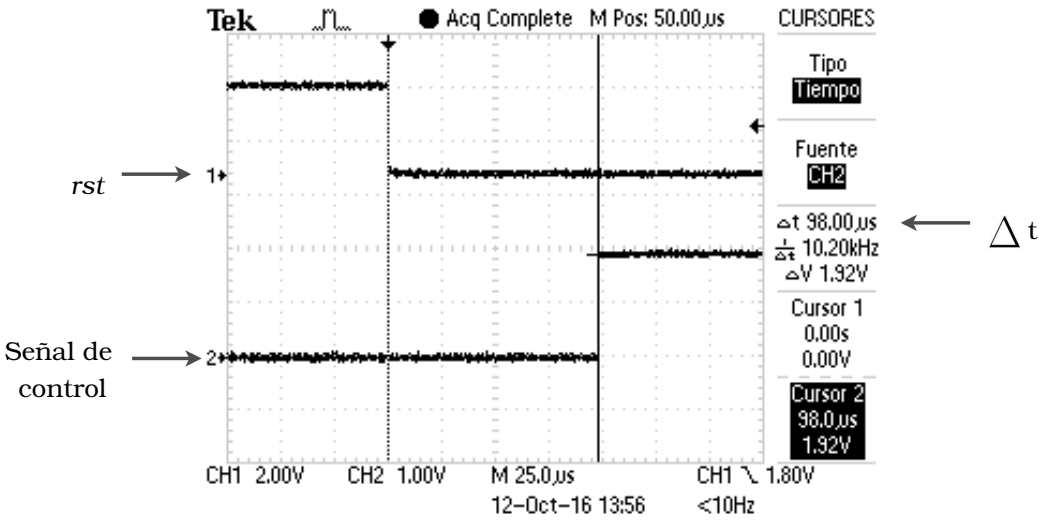


Figura 4.2: Medición del tiempo de cálculo mediante osciloscopio.

Cabe destacar que el uso de otro FPGA que sea capaz de trabajar a una frecuencia de reloj más elevada mejoraría el tiempo de cálculo, siempre y cuando se atienda la directriz de que el tiempo que toma la ejecución de la etapa más lenta sea menor al nuevo periodo de reloj (sección 2.8.1). Por ejemplo, la familia Stratix 10 de Altera [59] puede trabajar a 1 [GHz].

4.2. Comprobación gráfica

En la figura 4.1a se muestra la obtención del código correspondiente a los primeros dos elementos del arreglo. La línea *oan* revela que la posición angular del primer elemento será 0°, mientras que el segundo será 135°. Así mismo, la figura 4.1b presenta, de forma aún legible, la posición de los primeros seis elementos correspondientes al primer renglón: 0°, 135°, 90°, 45°, 0° y 135° respectivamente. Sin embargo, en la figura 4.1c se aprecia que la gran cantidad de transiciones que toman lugar en 97.68 [μ s] hace que tal información no sea visible.

Lógicamente, el simulador ofrece la posibilidad de aumentar o alejar la vista (*zoom*). Sin embargo, la comprobación de los resultados por este medio resulta altamente ineficiente. Por esta razón, se optó por realizar una simulación que permitiera comprobar gráficamente el resultado. Para ello, se programó la ecuación 3.4 en Octave y se generó una imagen que muestra gráficamente la posición angular de cada elemento en color azul, como se muestra en la figura 4.3.

Posteriormente, se exportan los resultados de la simulación de Quartus II como un archivo de texto, y se grafican sobre la figura anterior en color rosa. Cada línea que se muestra en la figura corresponde a la posición angular de los diodos que permanecen en estado de conducción en un anillo. De esta forma, la verificación resulta sencilla: si ambos métodos llevaron al mismo resultado para determinado elemento, se observa una línea en esa posición (rosa porque fue la última en generarse), ya que el resultado obtenido por la arquitectura sobrescribe al de Octave. Por el contrario, donde se muestran dos líneas, los métodos obtuvieron resultados distintos.

En la figura 4.3 se muestra el cálculo realizado para la reflexión del haz a la dirección θ_h=45° y φ_h=135°. Dicha representación muestra concordancia con el resultado obtenido en la figura 4.1b. Sin embargo, también se puede apreciar que existen errores entre los dos métodos de cálculo. Para ubicarlos, considérese el sistema de coordenadas normalizado $\overline{x}$,$\overline{y}$ que se muestra en la figura 4.3. En este caso, los renglones con coordenada $y = \pm 1, \pm 3, \pm 5$,etc tienen una coordenada x que es múltiplo de 0.5. Por ejemplo, el elemento que se encuentra en la parte superior derecha del central tiene coordenadas (0.5,1). El primer renglón del arreglo tiene coordenada y=11, mientras que el último tiene y=-11.

Bajo esta convención, el elemento (2.5,5) presenta un error, ya que Octave aproximó la posición a 45° (azul), mientras que la arquitectura lo hizo a 0° (rosa). Examinando los valores obtenidos antes de la aproximación, se observó que Octave calculó 22.774°, mientras que la arquitectura obtuvo 22.4056°. Recordando que el valor frontera que determina si se aproxima a 0° o a 45° es de 22.5° (figura 3.4), se explica claramente el origen del error. Algo semejante ocurre con los elementos de coordenadas (-1,4), (0,4), (-9,-2), (-4,-4), (3.5,-5) y (-7,-6).

Una forma de aproximar el resultado obtenido por la arquitectura al de Octave es incrementando la precisión numérica. Sin embargo, una inspección detallada revela que el error que

existe entre la posición angular real y la aproximación que hicieron ambos métodos es casi la misma. La posición real del elemento es 22.4056°, por lo que el error de aproximación con Octave fue de $|45-22.4056| = 22.5944$, mientras que con la arquitectura fue de $|0-22.4056| = 22.4056$, con lo que se muestra que ambos métodos se alejan $22.5 \pm 0.2°$ de la posición original, razón por la cual se considera que no es necesario incrementar la precisión numérica.

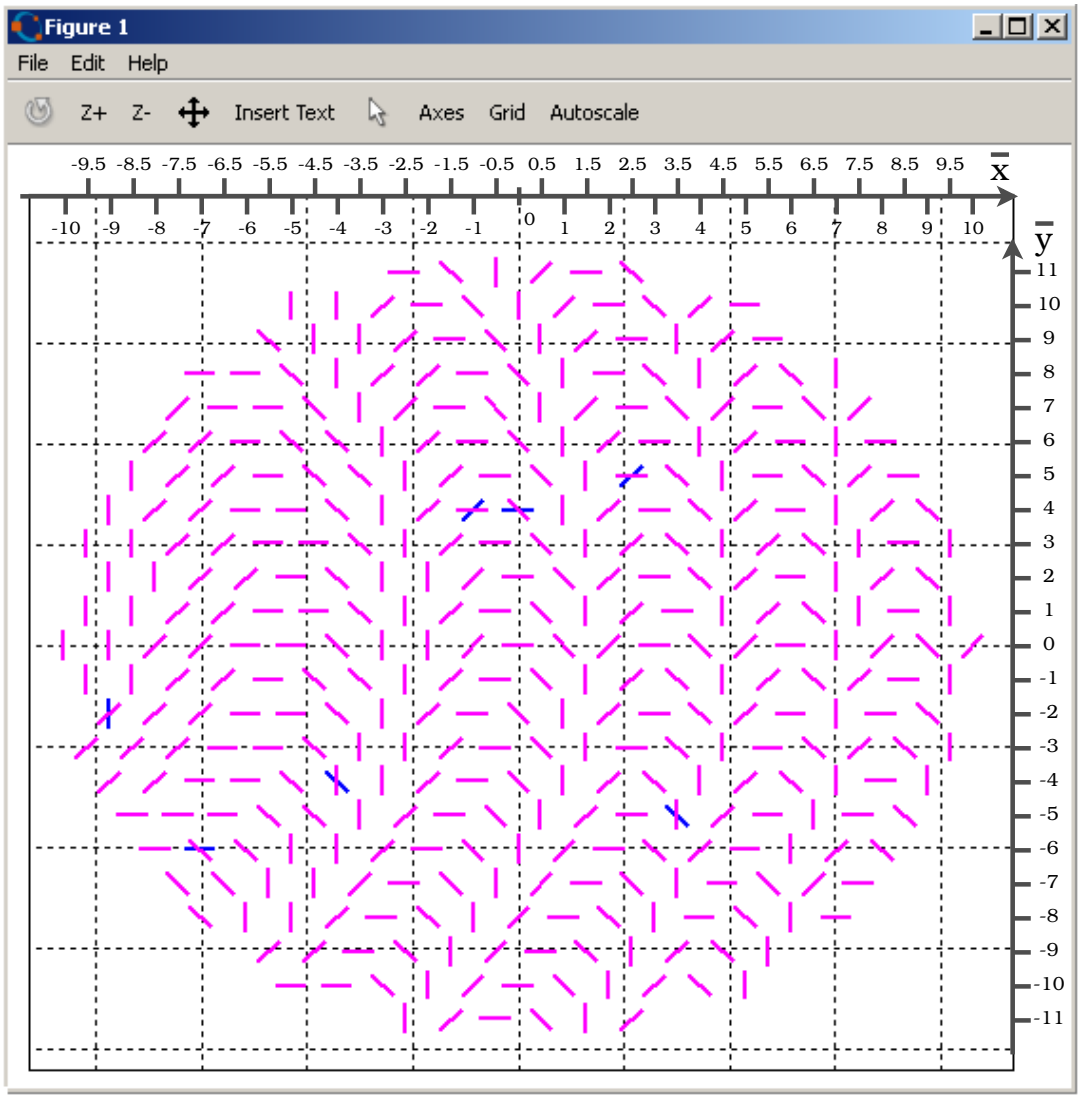


Figura 4.3: Simulación gráfica del arreglo configurado a $\theta_h = 45$ y $\varphi_h = 135$.

Tomando como punto de partida la fase que introducen los elementos radiadores a la onda reflejada (sección 3.4), se pueden realizar algunas manipulaciones algebraicas para revelar nuevamente el ángulo de elevación y azimut con el que originalmente se calculó la posición de cada elemento. Apoyándose nuevamente en Octave, fue posible recuperar la dirección original a partir de la posición angular *continua* de los elementos (antes de la aproximación). Sin embargo, al efectuar la aproximación a alguna de las cuatro posiciones disponibles, la distribución de fase sobre el eje horizontal (x) y el vertical (y) revela una progresión no uniforme,

razón por la cual no se puede determinar, mediante dicha manipulación algebraica, la contribución de la posición angular de cada elemento (una variable) a la dirección del haz principal (dos variables).

Por su parte, en [25] se menciona que el arreglo de desplazadores de dos bits provee únicamente cuatro cambios de fase. Por lo tanto, un cambio de fase linealmente distribuido debe aproximarse a posiciones discretas. Esta aproximación conduce a errores de fase y, por consecuencia, a la reducción de la ganancia del arreglo. Arreglos de dos bits muestran una pérdida de ganancia adicional de 0.91 [dB], mientras que desplazadores de tres bits la presentan de 0.22 [dB].

En [29] se menciona que el máximo error de direccionamiento $\Delta\theta_o$ debido a la cuantización se calcula de la siguiente forma:

$$\Delta\theta_o/\theta_B = \pi/(4 * 2^B) \tag{4.1}$$

donde θ_B es el ancho del haz y B el número de bits del desplazador. Para dos bits, el máximo error de direccionamiento resulta 0.196.

Finalmente, la forma de obtener el error entre la dirección deseada y la real involucra la resolución del sistema mediante un método asintótico como MoM, considerando así el acoplamiento mutuo entre los elementos radiadores y los efectos de borde. La obtención del patrón de radiación por este medio queda fuera del alcance de este trabajo.

4.3. Verificación Física

La última forma de verificar el funcionamiento del procesador diseñado consiste en la visualización física de las señales lógicas que, tras la etapa de acondicionamiento, finalmente controlarán a los conmutadores. Las pruebas de la arquitectura se desarrollaron en la tarjeta de evaluación DE2-115 de Terasic [60], la cual cuenta un FPGA EP4CE115F29C7N de la familia Cyclone IV, así como todos los elementos que requiere el FPGA para su configuración, interruptores mecánicos para la entrada de información y diodos emisores de luz LED (*Light Emitting Diode*) para representar las señales de control, entre otros periféricos.

Mapeando adecuadamente las entradas y salidas en la interfaz de planeación de pines (*pin planner*) del ambiente Quartus II, es posible introducir al sistema la dirección deseada codificando los ángulos en binario puro y asignando siete interruptores mecánicos para θ_h y nueve para φ_h. Por su parte, las señales de control se pueden visualizar mediante los 26 LEDs con los que cuenta la tarjeta. Si se contempla una señal de control por cada interruptor, únicamente se observan las señales de 6.5 anillos en un tiempo determinado. Sin embargo, si se realiza una codificación a dos bits (sección 3.7.1), es posible visualizar cualquier grupo de 13 anillos, como se muestra en la figura 4.4.

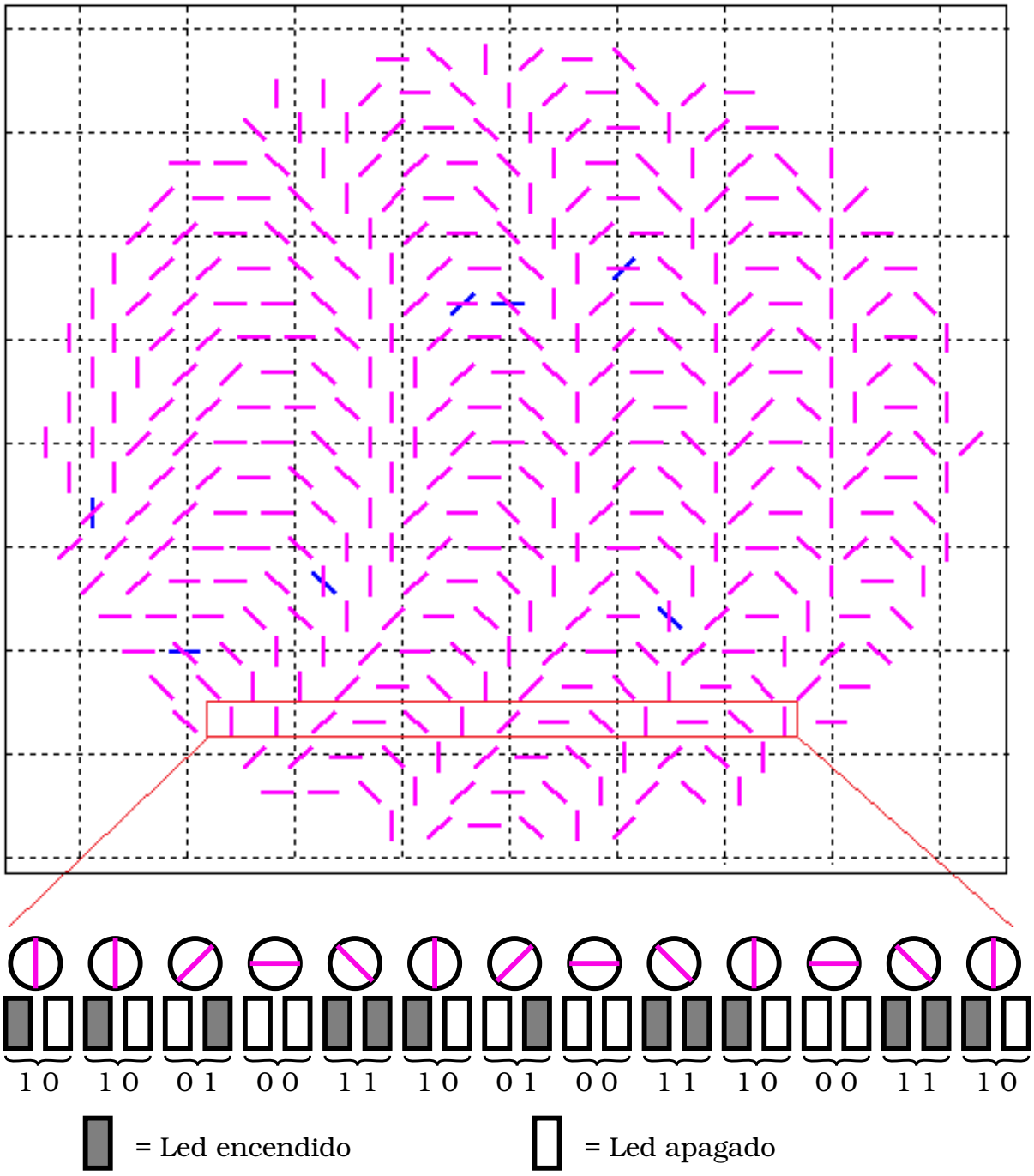


Figura 4.4: Visualización de señales en los LEDs de la tarjeta DE2-115 para $\theta_h = 45^\circ$ y $\varphi_h = 315^\circ$.

4.3.1. Uso de Recursos Lógicos

La síntesis de la arquitectura en el FPGA Cyclone IV reveló, a través del software Quartus-II, un consumo de 2,670 elementos lógicos. Por otra parte, si se desea implementar el paralelismo

por multinúcleo (sección 3.8), el mínimo número de LEs sería:

$$LE = n * r$$

donde n es el número de núcleos a implementar y r el número LEs utilizados por núcleo: 2,670 para el procesador diseñado.

Cabe destacar que este parámetro de consumo resulta importante para descartar el uso de dispositivos con una capacidad menor a la que se desea implementar. No obstante, contar con un dispositivo con el exacto número de LEs, o su equivalente, no garantiza que sea posible la síntesis, ya que al aproximarse al uso del 100 % de recursos, el programa puede encontrar problemas para implementar el diseño, de la misma forma que sucedió en la síntesis de la función lógica presentada en la sección 4.5.

4.4. Tarjeta Dedicada

Como se pudo observar en la sección 4.3, el uso de una tarjeta de desarrollo comercial permite al diseñador sintetizar determinada descripción en cuestión de minutos, debido a que cuenta con los elementos necesarios para el correcto funcionamiento del FPGA. Esto ahorra el tiempo involucrado en el diseño de un circuito impreso que cumpla dicha función.

No obstante, la desventaja de tales tarjetas es su filosofía de propósito general, la cual prácticamente las descarta para el uso de una aplicación específica como la de un arreglo reflectivo. Un ejemplo bastante ilustrativo es precisamente la tarjeta DE2-115, la cual permite el acceso únicamente a 40 pines de los 528 con los que cuenta el FPGA. El resto de ellos se encuentran ocupados por una amplia variedad de periféricos que carecen de utilidad para la aplicación. Incluso, los 26 pines que manejan los LEDs que se utilizaron para visualizar las señales (sección 4.3) son inaccesibles para su uso en un arreglo real.

Por esta razón, es necesario diseñar una *tarjeta dedicada* que permita aprovechar todos los pines disponibles del FPGA. Actualmente, el grupo de trabajo tiene contemplado el diseño de un arreglo reflectivo de 100 elementos con desplazadores de dos bits cargados inductivamente, lo que implica 400 LCs. Para tal arreglo, y buscando en contar con un sistema capaz de manejar arreglos de mayor tamaño, se propone el diseño esquematizado en la figura 4.5.

La tarjeta CoreEPCE10 de WaveShare, contiene un FPGA de la familia Cyclone IV con 164 pines disponibles de entrada/salida IO (In/Out). Buscando mantener al mínimo el retardo debido a la carga serial del dato (sección 3.7.2), se propone el uso de registros de ocho bits conectados a 160 de estos pines para obtener 680 líneas de control, las cuales se pueden expandir a futuro con el arreglo de registros conectados en serie que se muestra en el inferior de la figura 4.5. Finalmente, si se atiende a la recomendación de utilizar interruptores SPDT para la etapa de acoplamiento (sección 3.1), estos pueden incluirse en el diseño de la tarjeta.

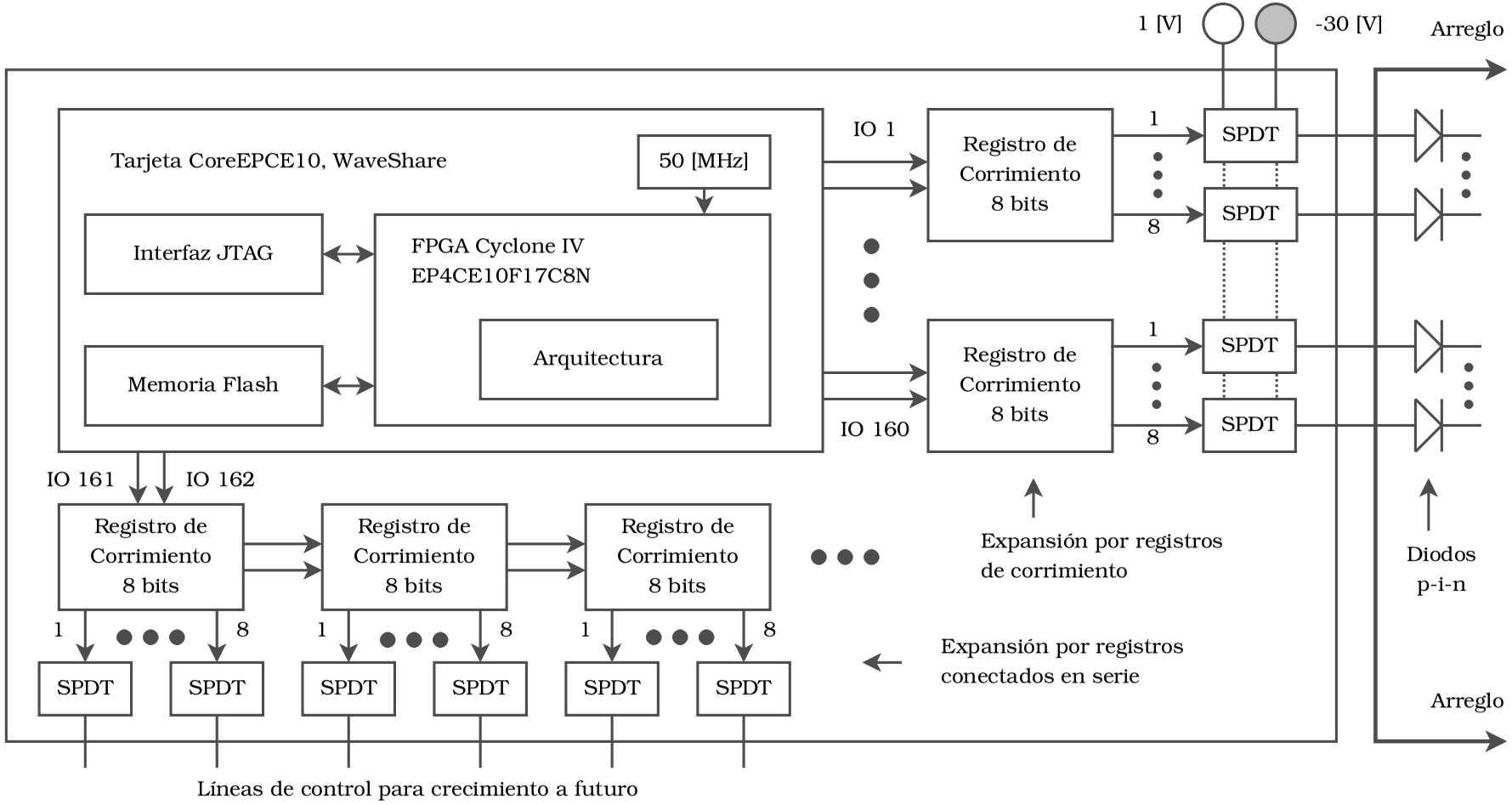


Figura 4.5: Diagrama de la tarjeta dedicada.

4.5. Comparativa con Otras Soluciones

Como se mencionó en la introducción del trabajo, existen otras opciones para dar solución al problema de la reconfiguración electrónica de un arreglo reflectivo. Con el fin efectuar una comparación que pretende ser demostrativa más que exhaustiva, se evaluaron dos soluciones adicionales: el uso microcontroladores comerciales y la implementación de una función lógica. En la tabla 4.1 se presentan características en las que poseen mayor fortaleza marcadas con una $\surd$.

Solución	Pines disponibles	Incremento ángulo	Flexibilidad	Tiempo de respuesta	Escalabilidad Multinúcleo
MCU comercial	—	$\surd$	—	$\surd$	—
Función lógica	$\surd$	—	—	$\surd$	—
Arquitectura de un procesador	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Tabla 4.1: Comparativa con dos alternativas.

La gran variedad de microcontroladores comerciales y su gran poder de procesamiento lógicamente los proponen como posible solución para el problema de la reconfiguración de un arreglo. Cuentan con una gran capacidad de cómputo, lo que se refleja en un buen tiempo de respuesta y buena precisión numérica gracias a la unidad aritmético-lógica con la que cuentan prácticamente todos los MCUs, así como la capacidad de manejar aritmética de punto flotante en algunos casos, mediante la unidad de punto flotante FPU (*Floating Point Unit*). En cuanto a sus desventajas se encuentra, en general, una pobre disponibilidad de pines de salida, así como la falta de flexibilidad debida a que la arquitectura de su unidad de procesamiento central está diseñada por el fabricante y se encuentra fija a nivel de hardware.

Las opciones disponibles comercialmente de estos dispositivos son amplias, ya que existen múltiples fabricantes (Texas Instruments TI, Micro Chip, Atmel, etc) y cada uno cuenta con distintos modelos. Buscando evaluar cuantitativamente el desempeño de microcontroladores de TI, se realizó un programa en ensamblador para el TMS320F28017, dispositivo que cuenta con 20 o 22 pines disponibles, dependiendo del empaquetado. Para el cálculo de los 367 elementos del arreglo propuesto (sección 3.1.2), las pruebas demostraron un TCH de aproximadamente 530 [μs]. Utilizando el ARM de TI MSP432P401R se obtuvo un TCH de 102 [ms].

Una alternativa al uso de los microcontroladores es la descripción de hardware, la cual ofrece una ventaja que atiende a la tecnología disponible para implementar un diseño: los dispositivos lógicos programables. Algunos de ellos, como los FPGAs, cuentan con una gran cantidad de pines disponibles que actualmente superan los 1,400 en familias como la Stratix X de Altera o Virtex UltraScale de Xilinx.

Basadas en la descripción de hardware, existen dos alternativas esencialmente distintas. La primera de ellas es la implementación de una función lógica. El comportamiento es básicamente el de una *memoria*, donde los ángulos de entrada concatenados forman la *dirección* donde se encuentra la cadena de 1468 bits que corresponden a las señales de control que configuran el arreglo para reflejar a la dirección de entrada.

El tiempo de respuesta de esta alternativa es el mínimo posible, ya que implica únicamente el retardo de las señales al propagarse por los elementos lógicos del dispositivo. Sin embargo, el costo del excelente tiempo de respuesta se refleja en dos aspectos: el elevado consumo de recursos y una gran sensibilidad al mínimo incremento de la dirección del haz Δ_h. Dicho comportamiento se puede visualizar en la forma de obtener el numero de funciones *nf*:

$$nf = \frac{(\theta_{max} - \theta_{min})}{\Delta\theta_h} * \frac{(\varphi_{max} - \varphi_{min})}{\Delta\varphi_h} \tag{4.2}$$

donde $[\theta_{min}, \theta_{max}]$ es el rango del ángulo de elevación y $\Delta\theta_h$ su mínimo incremento, mientras que $[\varphi_{min}, \varphi_{max}]$ es el rango del ángulo de azimut y $\Delta\varphi_h$ su mínimo incremento. Aquí puede apreciarse que entre menor sea el incremento de los ángulos, mayor será el número de funciones necesarias y, por lo tanto, mayor el consumo de recursos. Por ejemplo, si se contemplan rangos de θ_h = [0,64]°, φ_h = [0,360)° e incrementos $\Delta\theta_h$ = 4 y $\Delta\varphi_h$ = 3, entonces *nf* = 1,905, ya que la dirección θ_h = 0° es la dirección normal al arreglo independientemente del valor de φ_h.

Suponiendo el uso de un Cyclone IV EP4CE22F17C6 de 22,320 LEs para implementar tal función lógica, 11 pines para codificar la dirección de entrada al sistema y 134 pines conectados a un diodo p-i-n c/u, entonces la *memoria* a implementar sería de 1,905 *palabras* de 134 bits c/u. En la figura 4.6 se muestra el reporte de síntesis generado por Quartus II, donde se revela un consumo del 93 % de los recursos disponibles.

Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	20,834 / 22,320 (93 %)
Total pins	145 / 154 (94 %)

Figura 4.6: Uso de los recursos lógicos de un Cyclone IV implementando una función lógica.

En este punto quizá surja una observación respecto al reporte del total de pines en la figura 4.6: hay nueve pines sin utilizar que desperdician un 4 % del total disponible. Este es un problema que debe considerarse en cualquier diseño que implique el uso de PLDs: el programa que se utilice (Quartus II de Altera en este caso) no siempre logrará sintetizar un diseño que consuma un número de recursos cercanos al 100 % de la capacidad del dispositivo. En la figura 4.7 se muestra el error que arroja el programa cuando se intenta aumentar el número de salidas a 136 en el diseño anterior.

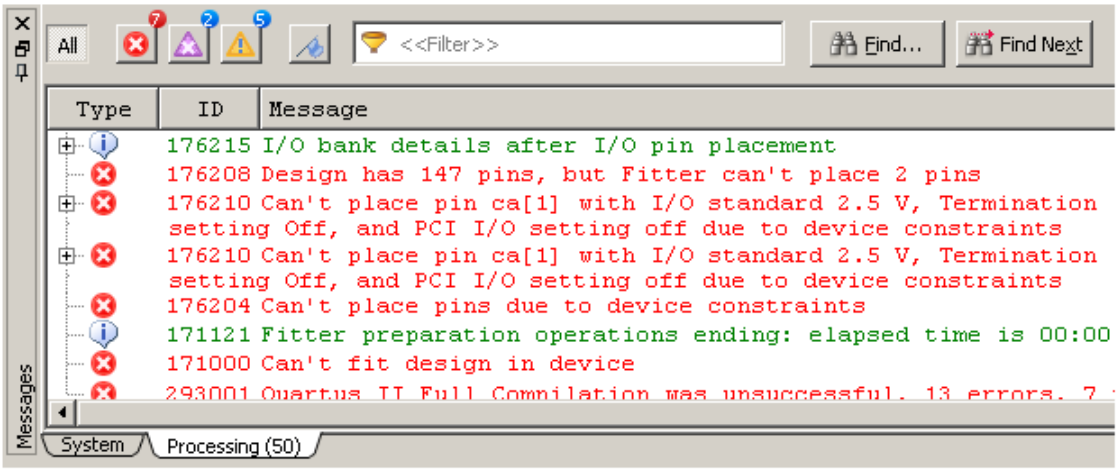


Figura 4.7: Error al asignar dos pines más de salida al diseño de la figura 4.6.

De este ejemplo se puede apreciar que el sistema no es fácilmente escalable, ya que el simple hecho de incrementar el rango de θ_h a 68° provocaría un incremento a nf = 2,024, lo cual supera la cantidad de recursos lógicos disponibles en un 4 %, como se muestra en la figura 4.8. Tras presentarse esta situación, Quartus II arroja un error como el que se muestra en la figura 4.9.

Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	23,236 / 22,320 (104 %)
Total pins	145 / 154 (94 %)

Figura 4.8: Uso de recursos excedido al aumentar rango de θ_h a 68°.

Contemplando los parámetros utilizados en el diseño del procesador dedicado (rangos de θ_h = [0,90]°, φ_h = [0,360)° e incremento de 1° para ambos), entonces el número de funciones sería 32,311, mientras que utilizando un incremento de 0.1°, nf = 3,239,101.

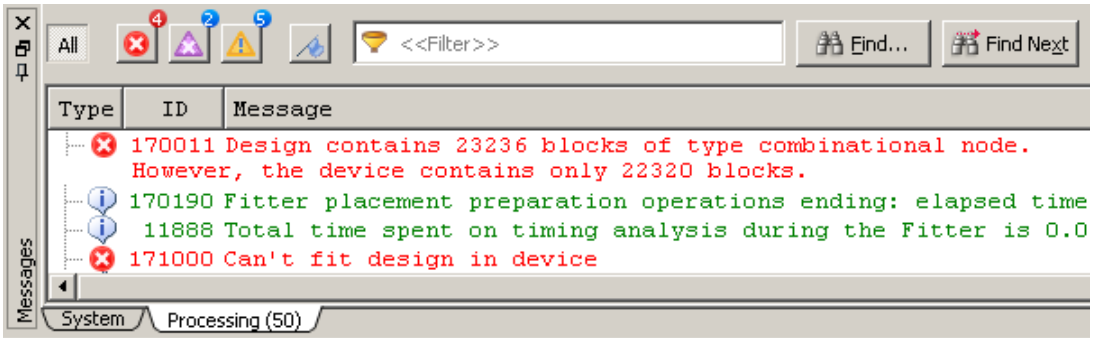


Figura 4.9: Error debido a cantidad de recursos excedida.

La segunda solución implica explotar el potencial que provee la descripción de hardware para diseñar la arquitectura de un procesador *dedicado*. La arquitectura diseñada y los resultados obtenidos demuestran que el sistema posee un excelente tiempo de respuesta y la posibilidad de mejorarlo mediante la implementación de múltiples núcleos. La libertad que ofrece la herramienta HDL para describir cada componente provee al sistema con una enorme flexibilidad para suprimir instrucciones y hacer más eficiente el cálculo, lo cual no es posible al utilizar la arquitectura de un microcontrolador comercial.

En cuanto al incremento de los ángulos, el uso de la decodificación por cuadrantes, permite mantener al mínimo la memoria de la función sinusoidal: [0°,90°]. De esta forma, la disminución de los incrementos impacta únicamente en el crecimiento de esta memoria. Por ejemplo, $\Delta\theta_h$=$\Delta\varphi_h$=1° implica el uso de 90 localidades, mientras que un incremento de 0.1° representa crecimiento a 900 localidades, lo cual resulta considerablemente menor que el crecimiento de 3,206,790 (3,239,101 - 32,311) localidades que se presenta en el caso de la función lógica.

El diseño de un procesador mediante HDL también goza de las ventajas que proporciona la tecnología de PLDs, ya que es factible implementar el diseño en un FPGA con un gran número de pines disponibles. Esto se demuestra en el hecho de que es posible utilizar un sólo FPGA para generar las 1,468 líneas de control de cada interruptor en el arreglo de 367 elementos. Sin embargo, las técnicas de expansión de pines ofrecen la posibilidad de manejar arreglos de mayor dimensión, aunque esta técnica sea igualmente aplicable a las otras alternativas.

Finalmente, en la tabla 4.2 una comparación del procesador dedicado contra el sistema implementado en [37].

	Ciclo de reloj	No. Operaciones	LC	TCH
Sistema publicado en [37]	62.5 [ns]	40	100	12.5 [μs]
Procesador dedicado	20 [ns]	13.3 promedio	1,468	97.68 [μs]

Tabla 4.2: Comparativa con el sistema implementado en [37].

De esta forma se pretende evidenciar que efectivamente existen múltiples formas de implementar el control de un arreglo de radiadores. Por ejemplo, existe una amplia variedad de microcontroladores comerciales que no fueron evaluados en este trabajo. Así mismo, la solución a ciertas limitantes, que en un principio se pretende aplicar a cierta solución, puede resultar aplicable a las otras, como sucedió con la expansión de pines. Y, en última instancia, podría pensarse en implementar un sistema híbrido que aproveche las fortalezas de cada alternativa. Sin embargo, la elección de una solución específica indudablemente quedará determinada por la aplicación que se le dará al sistema (académica, comercial, militar, etc) y los parámetros que se especifiquen para tal efecto.

4.6. Conclusiones y Trabajo a Futuro

El desarrollo de este trabajo llevó al diseño de un procesador capaz de realizar el cálculo de la posición angular de los 367 desplazadores de fase de dos bits del arreglo propuesto, el cual implica la obtención de 1,468 líneas de control independientes. Tal diseño se generó en el lenguaje VHDL y admite cualquier dirección del haz principal formada por un ángulo de elevación $\theta_h = [0, 90]$ y azimut $\varphi_h = [0, 360)$, ambos con un incremento $\Delta\theta_h = \Delta\varphi_h = 1°$.

Por su parte, la posición angular de cada anillo presentó un error promedio de 0.1° antes de aproximarse a posiciones discretas, lo que implica un promedio de 2 % de anillos con posición distinta respecto a la calculada mediante el programa Octave. Sin embargo, ambos métodos conducen a errores muy semejantes cuando se aproxima a posiciones discretas (sección 4.2). La solución del sistema mediante un método asintótico como el MoM se plantea como la mejor forma de estimar el impacto de este error.

Utilizando el programa Quartus II de Altera, las simulaciones revelan un tiempo de 97.68 [μs] para el cálculo secuencial de los 367 radiadores a una frecuencia de reloj de 50 [MHz]. Generando una señal de control que se habilita al terminar el cálculo del arreglo, se midió un tiempo de 98 [μs]. Dicho tiempo se encuentra dentro del rango establecido por el estado del arte presentado en la sección 1.10.1. No obstante, el sistema muestra potencial para aplicar el paralelismo a nivel de núcleos, con lo que se puede llegar a obtener un tiempo de cálculo de aproximadamente 300 [ns] utilizando la misma frecuencia de reloj. Si la etapa más lenta del cauce así lo permite, es posible utilizar dispositivos que sean capaces de trabajar a frecuencias de reloj más elevadas, mejorando aún más este parámetro.

La síntesis del diseño se realizó utilizando una tarjeta de desarrollo DE2-115 de Terasic, la cual incorpora un FPGA de la familia Cyclone IV de Altera. Cualquier grupo de 26 señales de control, tanto *directas* como codificadas, se pueden visualizar en los LEDs con los que cuenta la tarjeta, verificando así el funcionamiento físico del sistema.

Actualmente el grupo de trabajo contempla la fabricación de un arreglo de 100 anillos de dos bits cargados con stubs radiales (400 líneas de control). Con base en tal proyecto, se propuso el esquema de una tarjeta dedicada que ofrece la posibilidad de generar 640 líneas de control utilizando registros de corrimiento de ocho bits y 160 pines de los 164 que ofrece el FPGA. Más aún, el sistema contempla la posibilidad de expandir tal número utilizando la conexión de registros en serie.

Para el bloque de acoplamiento, se encontró una atractiva solución en los interruptores SPDT controlados electrónicamente, ya que permiten la conmutación de los voltajes necesarios por el diodo p-i-n mediante la señal de control que proviene directamente del FPGA. No obstante, aún existe la necesidad de contar una fuente que sea capaz de proveer las condiciones de voltaje y corriente requeridas por los conmutadores: 1,468 para carga inductiva y 2,936 para carga capacitiva en el arreglo de 367 elementos.

Apéndice A

Arquitectura CISC del microcontrolador 68HC11

En la figura A.2 se presenta la arquitectura CISC del microcontrolador 68HC11. Por su parte, en la figura A.1 se presenta la carta ASM correspondiente a la instrucción Ldaa con acceso directo, mientras que en la figura A.3 se presenta la instrucción Staa y en la figura A.4 la instrucción Addd, ambas utilizando direccionamiento indexado mediante el registro X.

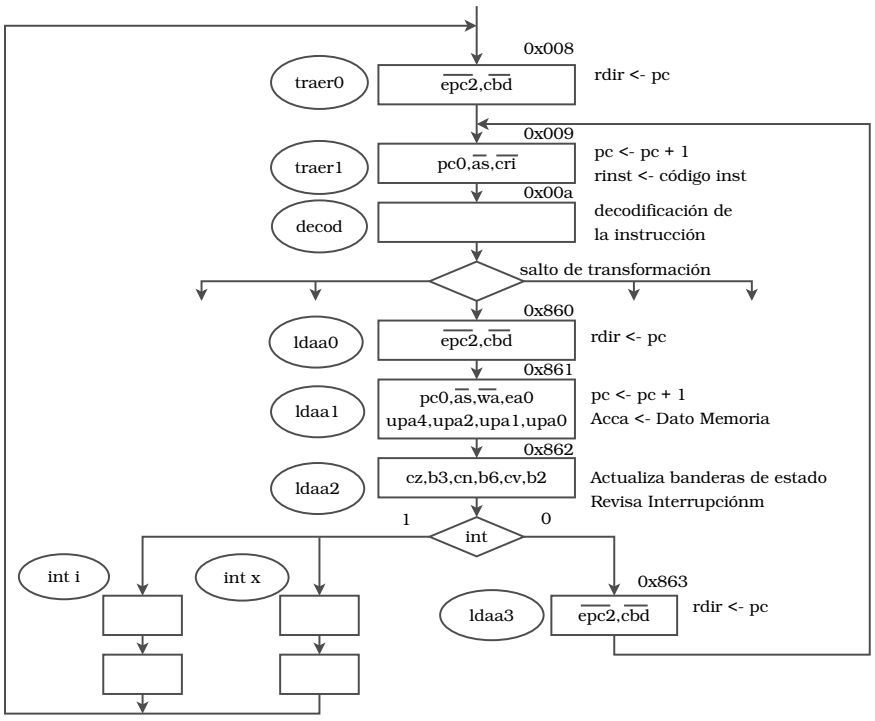


Figura A.1: Carta ASM de la instrucción Ldaa con acceso directo, de la arquitectura CISC del microcontrolador 68HC11 [47].

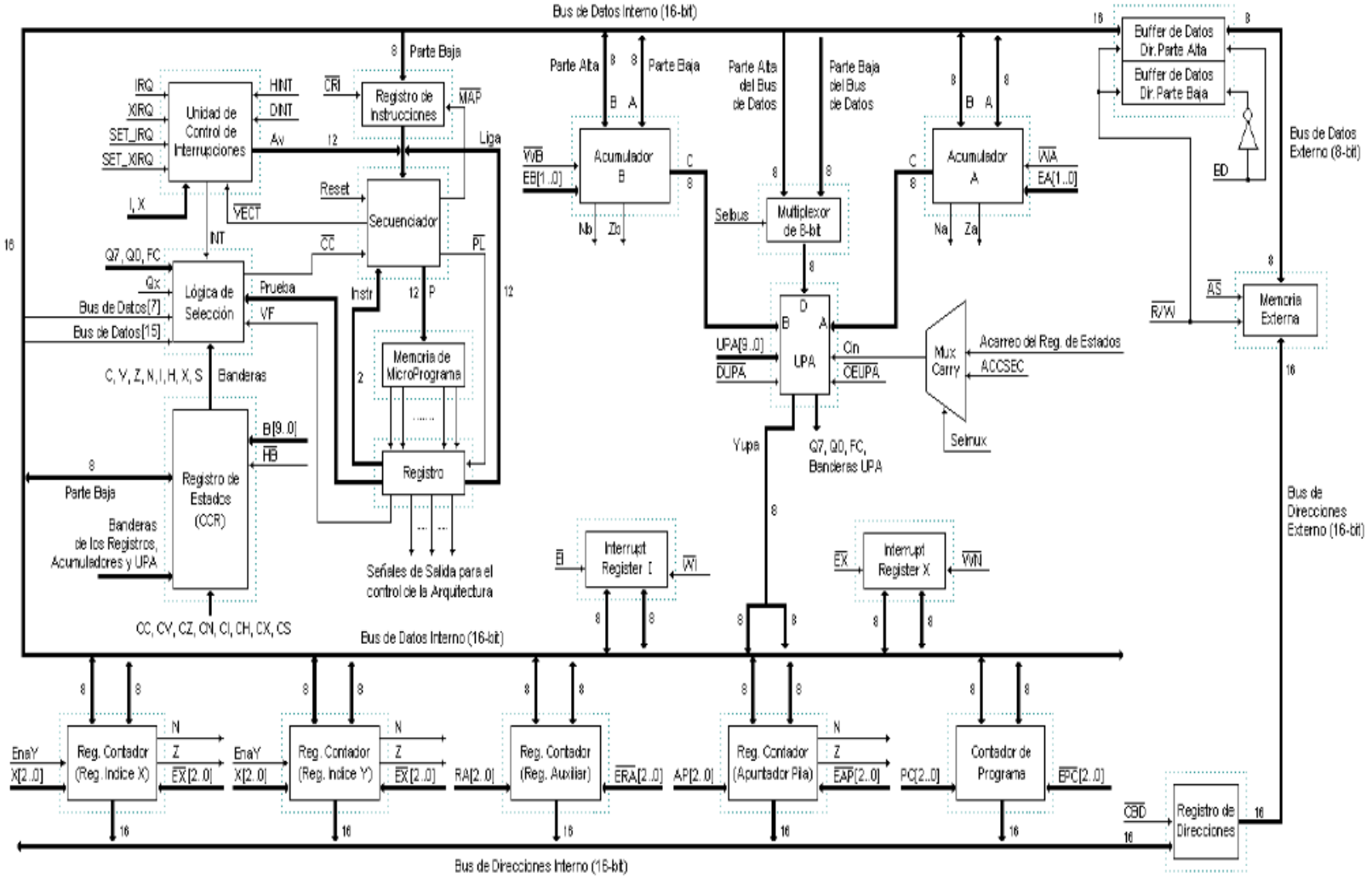


Figura A.2: Diagrama de la arquitectura secuencial del microprocesador 68HC11 [47].

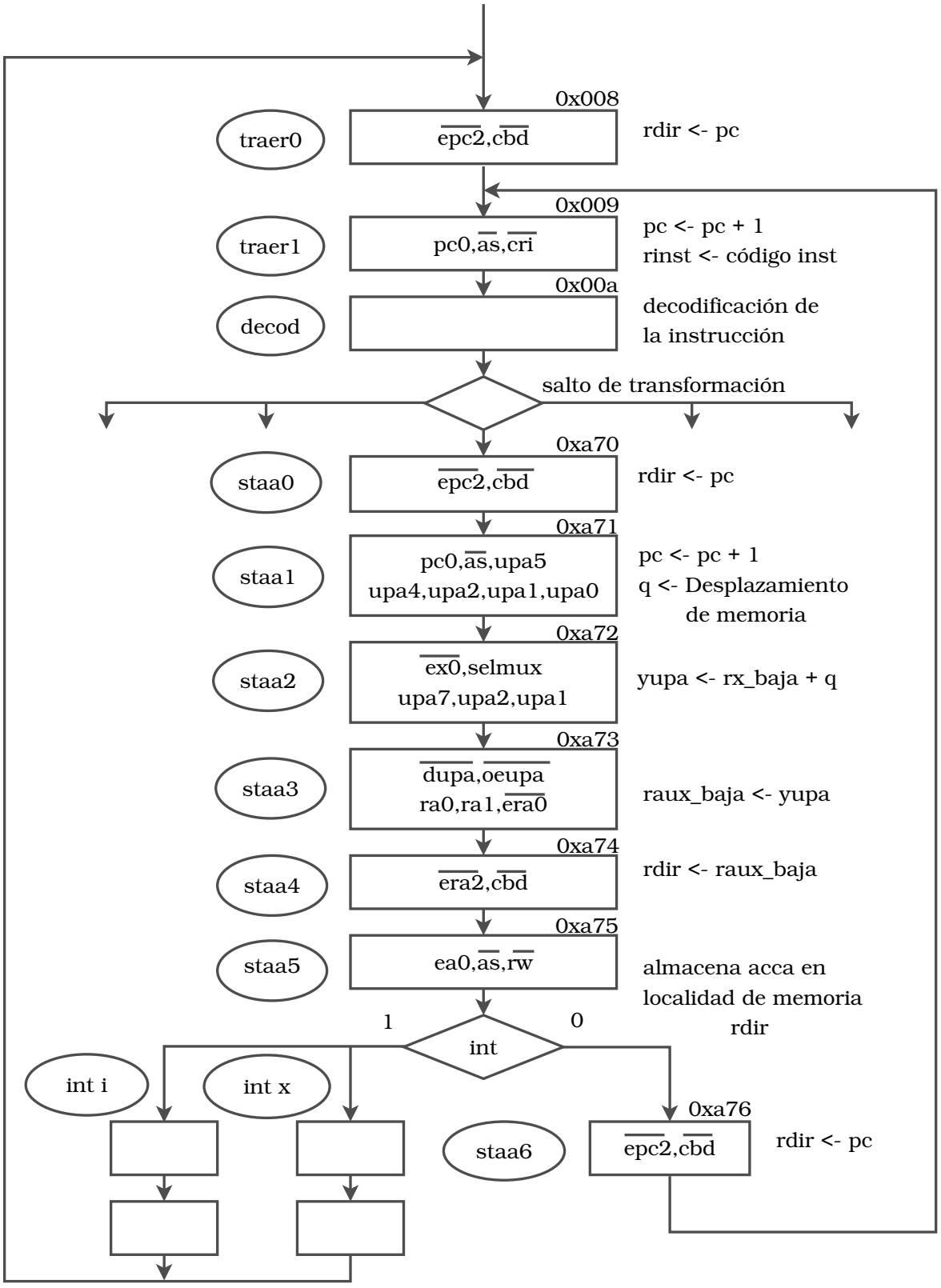


Figura A.3: Carta ASM de la instrucción Staa, indexada mediante el registro X, de la arquitectura CISC del microcontrolador 68HC11 [47].

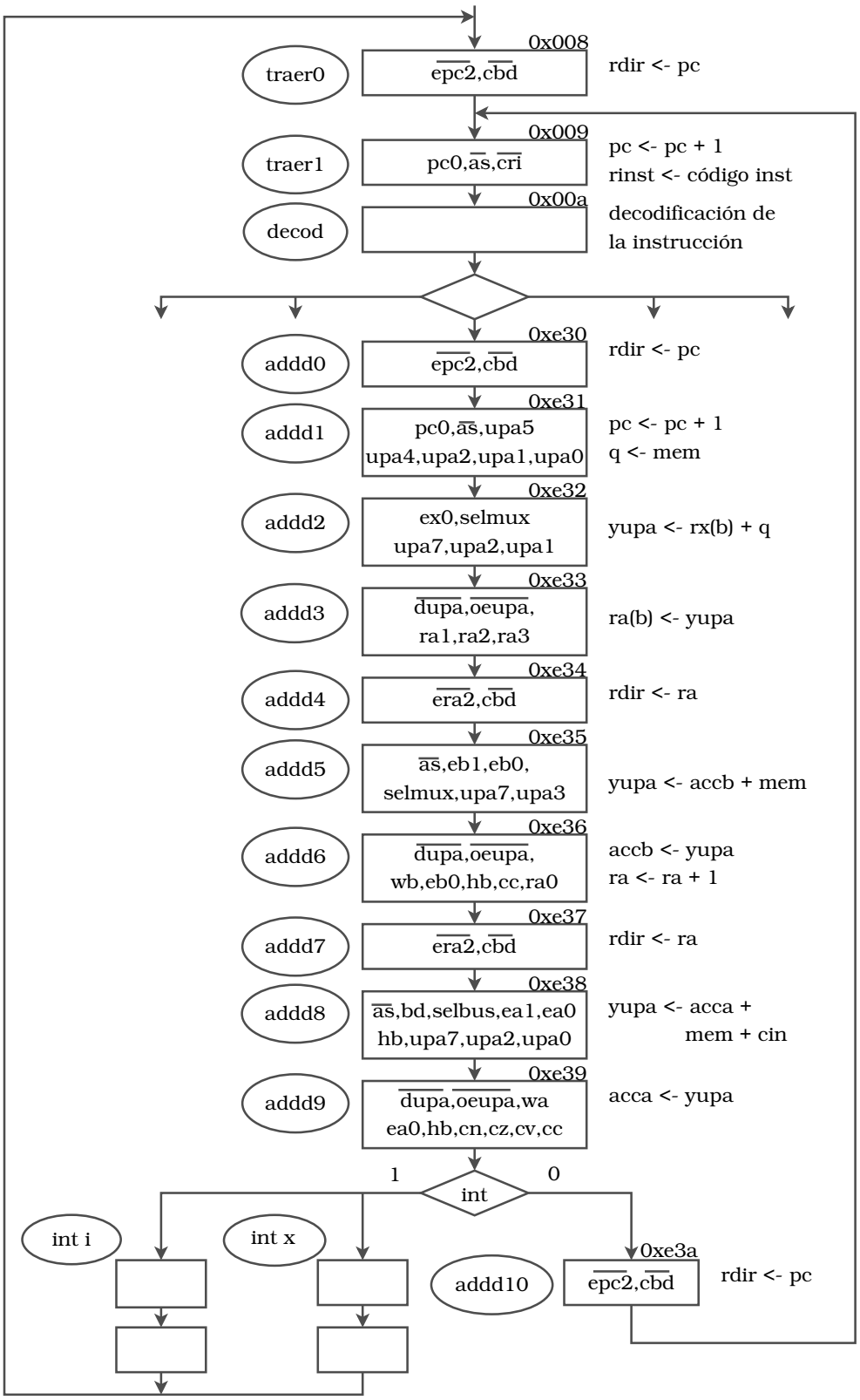


Figura A.4: Carta ASM de la instrucción Addd, indexada mediante el registro X, de la arquitectura CISC del microcontrolador 68HC11 [47].

Apéndice B

Bloques elementales del micoprocesador

A continuación se presenta una descripción general de algunos de los bloques más elementales que se utilizan en la arquitectura, descritos en VHDL.

B.1. Sumador de 8 bits

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity sumador is
  Port(
    o1 : in std_logic_vector(7 downto 0); --operando1
    o2 : in std_logic_vector(7 downto 0); --operando2
    ci : in std_logic; --acarreo de entrada
    rs : out std_logic_vector(7 downto 0); --resultado
    co : out std_logic --acarreo de salida
    );
end sumador;
architecture arq of sumador is
  signal su: std_logic_vector(8 downto 0);
begin
  su  <= ('0'&o1) + ('0'&o2) + ci;
  rs  <= su(7 downto 0);
  co  <= su(8);
end arq;
```

B.2. Complemento a dos

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity comp2 is
  Port(
    ops : in std_logic_vector(7 downto 0);
      --OperandoPos
    ong : out std_logic_vector(7 downto 0)
      --OperandoNeg
    );
end comp2;
architecture arq of comp2 is
begin
  ong <= not(ops) + 1;
end arq;
```

B.3. Contador de 8 bits

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity cont is
  Port(
    rst : in std_logic; --reset
    clk : in std_logic; --reloj'
    cta : out std_logic_vector (7 downto 0) --cuenta
    );
end cont;
architecture arq of cont is
  signal aux: std_logic_vector (7 downto 0);
begin
  process(rst,clk)
  begin
    if rst = '1' then
      aux <= x"00";
    elsif rising_edge(clk) then
      aux <= aux+1; --au;
    end if;
    cta <= aux;
```

```
  end process;
end arq;
```

B.4. Multiplexor 4 a 1 de 8 bits

```
library ieee;
use ieee.std_logic_1164.all;
entity z1 is
  Port(
     sel : in std_logic; --señal de selección
     en1 : in std_logic_vector(7 downto 0);
        --entrada 1
     en2 : in std_logic_vector(7 downto 0);
        --entrada 2
     en3 : in std_logic_vector(7 downto 0);
        --entrada 3
     en4 : in std_logic_vector(7 downto 0);
        --entrada 4
     sal : out std_logic_vector(7 downto 0)
        --salida
  );
end z1;
architecture arq of z1 is
begin
  with sel select
    sal <= en1 when "00",
           en2 when "01",
           en3 when "10",
           en4 when "11";
end arq;
```

B.5. Codificador binario de 4 a 2 bits

```
library ieee;
use ieee.std_logic_1164.all;
entity codi is
  Port(
    en1 : in std_logic_vector(3 downto 0);
      --entrada 1
    sal : out std_logic_vector(1 downto 0)
      --salida
```

```
  );
end codi;
architecture arq of codi is
begin
  process(en1)
  begin
  sal <= "00";
    case en1 is
      when "0001" => sal <= "00";
      when "0010" => sal <= "01";
      when "0100" => sal <= "10";
      when "1000" => sal <= "11";
      when others => null;
    end case;
  end process;
end arq;
```

B.6. Decodificador de 2 a 4 bits

```
library ieee;
use ieee.std_logic_1164.all;
entity deco is
  Port(
    sel : in std_logic_vector(1 downto 0); --selección
    sal : out std_logic_vector(3 downto 0) --salida
  );
end deco;
architecture arq of deco is
begin
  process(sel)
  begin
    case sel is
      when "00" => sal <= "0001";
      when "01" => sal <= "0010";
      when "10" => sal <= "0100";
      when "11" => sal <= "1000";
    end case;
  end process;
end arq;
```

B.7. Registro de 8 bits con reestablecimiento asíncrono

```
library ieee;
use ieee.std_logic_1164.all;
entity reg is
  Port(
    rts: in std_logic; --señal de reestablecimiento
    clk: in std_logic; --'senal de reloj'
    ren: in std_logic_vector(7 downto 0);
      --valor a la entrada
    rsl: out std_logic_vector(7 downto 0)
      --valor a la salida
    );
end reg;
architecture arq of reg is
begin
  process (rts,clk,ren)
  variable rau : std_logic_vector (7 downto 0)
  := x"14"; --variable auxiliar que almacena el valor
  -- del registro. Puede contar con un valor inicial.
  begin
    if rts = '1' then
      rau := x"14";
    elsif rising_edge(clk) then
      rau := ren;
    end if;
    rsl <= rau;
  end process;
end arq;
```

B.8. Memoria Sólo Lectura (ROM) de 256 localidades de 8 bits

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity z1 is
Port (
    dir: in std_logic_vector (7 downto 0);
    ins: out std_logic_vector (7 downto 0)
    );
end entity;
architecture arq of z1 is
type mt is array (255 downto 0) of
      std_logic_vector (7 downto 0);
  constant mi : mt :=
  (
    0 to 200  => x"77",--delcaración de un bloque
    201      => x"01",--declaración de una localidad
    202      => x"0a",
    203      => x"b7",
    204 to 255=> x"ff"
  );
begin
  process(dir)
  begin
    ins <= mi(to_integer(unsigned(dir)));
  end process;
end arq;
```

Bibliografía

[1] C. A. Balanis, *Antenna Theory: Analysis and Design*. USA: Wiley, 1997.

[2] R. de Andrade Martins, "Romagnosi and volta's pile: Early difficulties in the interptretation of voltaic electricity," *Nuova Voltania*, 2000.

[3] A. A. Huurdeman, *The worldwide history of telecommunications*. New Jersey, USA: Wiley, 2003.

[4] John and M. Gribbin, *Faraday in 90 minutes*. India: Universities Press, 2003.

[5] G. R. M. Garratt, *The Early History of Radio*. London, U.K.: The Institutiom of Engineering and Technology, 2006.

[6] J. C. Maxwell, *A treatise on Electricity and Magnetism*. London: Clarendon Press, 1893.

[7] H. J. Visser, *Array and Phased Array Antenna Basics*. West Sussex, England: Wiley, 2005.

[8] O. J. Lodge, *Past Years - An Autobiography*. London: Hodder and Stoughton, 1931.

[9] D. T. Emerson, "The work of jagadis chandra bose: 100 years of millimeter-wave research," *IEEE Transactions on Microwave Theory and Techniques*, December 1997.

[10] J. Ramsay, "Highlights of antenna history," *IEEE Comunications Magazine*, September 1981.

[11] M. I. Pupin, "A discussion on experimental tests of the radiation law for radio oscillators," *Proc. Inst. Radio Engineers*, 1913.

[12] C. A. Balanis, "Antenna theory: A review," *Proceedings of the IEEE*, January 1992.

[13] D. G. Berry, R. G. Malech, and W. A. Kennedy, "The reflectarray antenna," *IEEE Transactions on Antennas and Propagation*, November 1963.

[14] J. Huang and J. Encinar, *Reflectarray antennas*. Honoken, NY: Wiley, 2008.

[15] R. Garg, P. Bhartia, I. Bahl, and A. Ittipiboon, *Microstrip Antenna Design Handbook*. London, U.K.: Artech House, 2001.

[16] D. A. Pozar, "Microstrip antennas," *Proceedings of the IEEE*, January 1992.

[17] R. J. Mailloux, *Phased Array Antenna Handbook*. , USA: Artech House, 2005.

[18] A. G. Fox, "An adjustable wave-guide phase changer," *Proceedings of the I.R.E.*, December 1947.

[19] H. R. Phelan, "Spiraphase reflectarray for multitarget radar," *Microwave Journal*, July 1977.

[20] A. E. Martynyuk, N. A. Martynyuk, and S. N. Khotiaintsev, "Millimeter-wave amplitude-phase modulator," *IEEE Transactions on Microwave Theory and Techniques*, June 1997.

[21] A. E. Martynyuk, J. Martínez-López, J. Rodríguez-Cuevas, and Y. K. Sydoruk, "Wideband reflective array based on loaded metal rings," *IEEE Microwave Symposium Digest*, June 2005.

[22] A. E. Martynyuk and Y. K. Sidoruk, "Low-loss phase shifters for ka band phased array," *IEEE International Conference on Phased Array Systems and Technology*, May 2000.

[23] A. E. Martynyuk, A. G. Martinez-Lopez, and J. Martínez-López, "2-bit x-band reflective waveguide phase shifter with bcb-based bias circuits," *IEEE Transactions on Microwave Theory and Techniques*, March 2006.

[24] A. V. Chenakin, O. E. Martynyuk, and V. I. Skachko, "A new hybrid technology for millimeter-wave integrated circuits," *IEEE MTT-S International Symposium*, June 1997.

[25] A. E. Martynyuk, J. Martínez-López, and N. A. Martynyuk, "Reflective passive phased array whith open polarization phase shifters," *IEEE International Symposium on Phased Array Systems and Technology*, October 2003.

[26] A. E. Martynyuk and J. Rodríguez-Zamudio, "Polarization phase shifters and spiraphase-type phased arrays," *MSMW Symposium Proceedings*, June 2007.

[27] A. E. Martynyuk, N. A. Martynyuk, and J. Martínez-López, "Reflective phased array based on split metal rings with p-i-n diode switches," *First European Conference on Antennas and Propagation*, November 2006.

[28] J. Silva-Montero, J. Martínez-Lopez, J. Rodríguez-Cuevas, and A. E. Martynyuk, "Spiraphase-type reflectarray for large reflection elevation angles," *IEEE Transactions on Antennas and Propagation*, October 2015.

[29] M. I. Skolnik, *Introduction to Radar Systems*. USA: McGraw-Hill, 1980.

[30] M. Yi, Y. Hong, W. Lee, and J. So, "Digitized millimeter-wave beam-forming metal reflectarray antenna," *The 2015 International Workshop on antenna Technology*, March 2015.

[31] T. Jeffrey, *Phased-Array radar Desgin: Application of Radar Fundamentals*. USA: Scitech, 2009.

[32] J. E. Reed, "The an/fps-85 radar system," *Proceedings of the IEEE*, March 1969.

[33] F. Neri, *Introduction to Radar Systems*. USA: Artech House, 2006.

[34] G. Pérez-Palomino, M. Barba, and J. A. Encinar, "Design and demonstration of an electronically scanned reflectarray antenna at 100 ghz usin multiresonant cells based on liquid crystals," *IEEE Transactions on Antennas and Propagation*, August 2015.

[35] A. Patyuchenko, C. Tienda, M. Younis, S. Bertl, P. López-Dekker, and G. Krieger, "Concept of a multi-beam reflectarray digital-beam forming synthetic aperture radar," *IEEE International Symposium on Phased Array Systems and Technology*, October 2013.

[36] H. Kamoda, T. Iwasaki, J. Tsumochi, T. Kuki, and O. Hashimoto, "60-ghz electronically reconfigurable large reflectarray using single-bit phase shifters," *IEEE Transactions on Antennas and Propagation*, July 2011.

[37] H. Yang, F. Yang, S. Xu, Y. Mao, M. Li, X. Cao, and J. Gao, "A 1-bit 10x10 reconfigurable reflectarray antenna: Design, optimization and experiment.," *IEEE Transactions on Antennas and Propagation*, April 2016.

[38] M. Balch, *Complete Digital Design*. USA: McGraw Hill, 2003.

[39] P. E. Ceruzzi, *A History of Modern Computing*. USA: MIT Press, 2003.

[40] G. O'Regan, *A Brief History of Computing*. London: Springer, 2012.

[41] M. M. Mano and C. R. Kime, *Logic and Computer Design Fundamentals*. USA: Prentice Hall, 2004.

[42] A. Hellemans, "Sis-state memristor opens door to weird computing," *IEEE Spectrum*, 2014.

[43] M. M. Mano, *Computer System Architecture*. USA: Prentice Hall, 2004.

[44] I. Grout, *Digital Systems Design with FPGAS*. USA: Newnes, 2008.

[45] L. Null and J. Lobur, *The essentials of Computer Organization and Architecture*. USA: Jones and Bartlett, 2003.

[46] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. USA: Morgan Kaufmann, 2012.

[47] J. Savage-Carmona, G. Vázquez, and N. E. Chávez-Rodríguez, *Diseño de Microprocesadores*. México: FI UNAM, 2015.

[48] V. A. Pedroni, *Circuit Design with VHDL*. USA: MIT Press, 2004.

[49] R. F. Tinder, *Engineering Digital Design*. USA: Elsevier Science, 2000.

[50] J. L. Hennessy and D. A. Patterson, *Computer Organization and Design*. USA: Morgan Kaufmann, 1994.

[51] S. L. H. David M. Harris, *Digital Design and Computer Architecture*. USA: Morgan Kaufmann, 2007.

[52] J. D. I. B. D. C. Victor P. Nelson, H. Troy Nagle, *Digital Logic Circuit Analysis and Design*. USA: Prentice Hall, 1995.

[53] *AV01-0593ENO: HPND-4005 Data Sheet*. Avago Technologies, 2006.

[54] J. E. Stine, *Digital Computer Arithmetic Datapah Design Using Verilog HDL*. USA: Kluwer, 2004.

[55] *Introduction to Quartus II*. Altera Corporation, Version 4.1 Rev.1, June 2004.

[56] *DS890: UltraScale Architecture and Product Overview*. Xilinx Inc., Version 2.9, September 2016.

[57] *Stratix 10 Product Table*. Altera Corporation, 2015.

[58] *SCES602D: SN74LVC1G139 Data Sheet*. Texas Instruments, 2015.

[59] *Stratix 10 Device Overviwe*. Altera Corporation, 2015.

[60] *DE2-115 User's Manual*. Terasic Technologies Inc., 2013.