UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño y descripción en VHDL de una IP para una interfaz SPI con técnicas de tolerancia a fallas para aplicaciones en sistemas espaciales

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Juan Misael Martínez Campistrano

DIRECTOR DE TESIS

Dr. Saúl de la Rosa Nieves

Ciudad Universitaria, Cd. Mx., 2024

Agradecimientos

Llegar a este punto de mi vida no ha sido fácil. A lo largo del camino me crucé con múltiples retos, pero gracias a las personas que me rodean y que han mostrado preocupación y apoyo en cada momento, he podido superar cada uno de ellos.

Quiero dar un agradecimiento especial a mis padres, Juan Martínez y Aurora Campistrano, por su amor y apoyo incondicional, por brindarme todos los recursos necesarios para terminar una carrera universitaria, y por todas esas veces en que se preocuparon por la comida que llevaría al día siguiente. Les agradezco de todo corazón por siempre creer en mí y por estar siempre presentes.

Al Dr. Saúl de la Rosa Nieves, porque, a pesar de todo, siempre fue muy comprensivo y atento durante el proceso de esta tesis. Le doy gracias también por permitirme formar parte del Laboratorio de Instrumentación Electrónica de Sistemas Espaciales (LIESE).

A mis amigos y compañeros de la Facultad de Ingeniería, los llevaré siempre en mi corazón y recordaré esta bonita etapa llena de risas y estrés.

A mi novia, Diana Bautista, por darme siempre la fuerza y alentarme a seguir día con día con esta tesis.

INDICE

1. INTRODUCCIÓN 7

1.2 Objetivos 7

1.2.1 Objetivo general **7**

1.2.2 Objetivos específicos **7**

1.3 Descripción de capítulos. 7

2. ESTADO DEL ARTE 9

2.1 Estado del arte de Protocolo de comunicación SPI en FPGA. 9

2.2 Estado del arte de Técnicas de tolerancia a fallas en FPGA. 12

3. MARCO TEORICO 15

3.1 Descripción estructural en VHDL 15

3.2 Efectos de la radiación espacial en semiconductores 16

3.2.1 SET 16

3.2.2 SEU 16

3.2.3 Single Event Latch-up (SEL) 16

3.2.4 SEB 17

3.2.5 SEGR 17

3.3 Técnicas de tolerancia a fallas para FPGA basadas en SRAM 17

3.3.1 La técnica de tolerancia a fallas de redundancia modular triple (TMR) 17

3.4 AXI4 19

3.5 Funcionamiento de bus SPI. 21

3.5.1 Descripción del funcionamiento 22

3.5.2 Modos de operación 22

4. DISEÑO DE CONCEPTO DEL MODULO SPI. 25

4.1 Determinación de las especificaciones 25

4.1.1 Propuesta del diseño del módulo SPI 25

4.1.2 Diseño Funcional 27

4.1.2.1 Funcionamiento de transmisión. 27

4.1.2.2 Funcionamiento de recepción 27

4.1.2.3 Funcionamiento de registro de control. 28

4.1.2.4 Funcionamiento de divisor de frecuencia y CS. 28

4.1.2.5 Funcionamiento del Reset. 29

5. DISEÑO DE CIRUCITOS SECUENCIALES Y COMBINACIONALES. 30

5.1 Circuitos básicos 30
5.1.1 Flip-Flop D. 30
5.1.2 Flip-Flop JK. 32
5.1.3 Multiplexor 4 a 1 33
5.2 Circuitos secuenciales. 35
5.2.1 Contador 1 (Datos). 35
5.2.2 Contador 2 (Filas de memoria FIFO_TX) 36
5.2.3 Contador 3 (Filas de memoria FIFO_RX) 38
5.2.4 Contador 4 (Divisor de frecuencia) 39
5.2.5 Contador 5 (QSCK2 Contador 2) 40
5.2.6 Contador 6 (Extraer dato de FIFO_RX) 41
5.2.7 Contador 7 (Ingresar dato a FIFO_RX) 41

6. DISEÑO DE BLOQUES FUNCIONALES PARA CONTROL Y DIRECCIONAMIENTO DE DATOS 43

6.1 Memoria FIFO 43
6.1.1 Memoria FIFO de Transmisión (FIFO_TX) 43
6.1.2 Memoria FIFO de Recepción (FIFO_RX) 48
6.2 Divisor de frecuencia. 52
6.3 Registros de desplazamiento 54
6.3.1 Transmisión (Paralelo -Serie) 54
6.3.2 Recepción (Serie - Paralelo) 55
6.3.3 Registros DR 57
6.3.3.1 Registro DR - escritura 57
6.3.3.2 Registros DR - lectura 58
6.4 Numero de Bits en cada dato (DT) 59
6.5 Chip Select (CS) 61
6.6 Polaridad y Fase del reloj (CPOL y CPHA) 65
6.6.1 Funcionamiento del reloj SCK 66
6.6.2 Solución al problema generado por señales CPOL y CPHA 69
6.7 Configuración en modo Esclavo. 70
6.8 Reset 78
6.9 Funcionamiento DR – FIFO – SALIDA. 81

6.10 Registros de control, datos y estado. 85
6.11 Tolerancia a fallas 86
6.12 AXI4 lite 90

7. EVALUACION DE RESULTADOS 91

7.1 Generación de espacio de pruebas. 91
7.2 Pruebas Físicas 91
7.2.1 Módulos internamente conectados. 92
7.2.2 Módulos externamente conectados. 95

8. CONCLUSIONES Y TRABAJO A FUTURO 102

9. ANEXOS 103

Anexo 1. Estilos de descripción en VHDL. 103
Anexo 2. Tablas de estados para diseño de contadores. 105
Anexo 3. Creación de una IP AXI4 Lite. 109
Anexo 4. Creación del entorno de pruebas. 119

10. REFERENCIAS. 125

1. INTRODUCCIÓN

El entorno en el que será utilizada la Interfaz Periférica Serial (SPI, por sus siglas en inglés) se encuentra llena de radiación ionizante elevada porque su aplicación será en sistemas espaciales. Debido a esto se tendrá un ingreso de partículas externas con alta energía al Arreglo de Compuertas Programables en Campo (FPGA, por sus siglas en inglés), las cuales podrían generar un cambio en las celdas de memoria de configuración causando alteraciones en la descripción de hardware de la interfaz SPI de manera permanente o cambiando uno o varios bits en cualquier parte del hardware mientras se encuentra en ejecución y en consecuencia tener datos finales erróneos.

Para poder mitigar los errores generados debido al ambiente espacial, se necesita instrumentar con técnicas de tolerancia a fallas a la interfaz. Estas técnicas pueden ser tanto simples como sofisticadas y para poder determinar qué tipo de técnica se aplicará, se tendrá que realizar un análisis de la importancia del componente junto con la probabilidad de error, esto para poder tener mayor confiabilidad en el sistema.

Es necesario diseñar esta interfaz SPI y describirla mediante VHDL al más bajo nivel de abstracción, permitiendo control y conocimiento total de todas las conexiones internas para poder ubicar los puntos más susceptibles. El estilo de descripción utilizado facilitara aplicar las técnicas de tolerancia a fallas debido a que se tiene conocimiento interno del diseño a nivel compuerta y poder asociar la interfaz axi4 lite con el SPI de una manera más fácil.

Las pruebas físicas se implementarán en el FPGA Basys 3 de Xilinx, con un oscilador dentro de la misma tarjeta de 100 MHz el cual es utilizado como reloj global. La transmisión entre dispositivos se ejecutará internamente y entre dos tarjetas con la ejecución del hardware en cada una de ellas.

1.2 Objetivos

1.2.1 Objetivo general

Diseñar una interfaz para recibir y transmitir datos mediante el protocolo de comunicación SPI. El sistema debe proporcionar la posibilidad de configurar parámetros controlables como son la frecuencia de transmisión, tamaño variable del dato a recibir y transmitir, y características propias del protocolo. El diseño debe permitir la instrumentación de técnicas de tolerancia a fallas para mitigar los efectos de la radiación espacial.

1.2.2 Objetivos específicos

- Diseñar una interfaz SPI y describirla mediante VHDL utilizando el mínimo de recursos en la descripción de hardware mediante el estilo de descripción tipo flujo de datos estructural.
- Agregar técnicas de tolerancia a fallas en la interfaz

1.3 Descripción de capítulos.

En este apartado se dará una descripción breve del contenido de cada capítulo.

Capítulo 1: Se presenta la problemática general del proyecto mediante un texto introductorio, los objetivos generales y específicos.

Capítulo 2: En este capítulo se revisan diferentes proyectos relacionados con el tema elegido. La primera parte se centra en diversos diseños de módulos SPI, y la segunda parte presenta investigaciones y aplicaciones de técnicas de tolerancia a fallas para sistemas espaciales.

Capítulo 3: Este capítulo contiene toda la información sobre los recursos necesarios para comenzar a diseñar y describir el proyecto. Comienza con una descripción del hardware en VHDL, continúa con las posibles causas de fallo en el hardware y finaliza con la estructura estándar del módulo SPI

Capítulo 4: Se menciona una descripción general del direccionamiento de datos y su funcionamiento de los componentes más importantes del módulo SPI.

Capítulo 5: Se describe y se muestra el diseño de los circuitos secuenciales y combinacionales utilizados dentro de toda la descripción de hardware.

Capítulo 6: La descripción de hardware de los componentes esenciales se muestra en este capítulo. En caso de haberse presentando alguna complicación, también se describe su solución.

Capítulo 7: Los resultados obtenidos, sean funcionales o no, se muestran mediante imágenes.

2. ESTADO DEL ARTE

El estado del arte es una investigación bibliográfica sobre los últimos avances de un área del conocimiento humano en específico con el fin de reunir el procedimiento de trabajos anteriormente realizados junto con los resultados obtenidos de cada uno. Principalmente, esta investigación es introductoria al tema elegido, así como también permite fundamentar la toma de decisiones y los resultados que se pueden obtener o superar del objetivo que se desea alcanzar.

El SPI es un estándar de comunicación síncrona entre dispositivos electrónicos. Existen diferentes modelos de este estándar ya que es muy variable al momento de diseñarlo, la frecuencia de transmisión, el número de bits que se pueden enviar en una trama, la funcionalidad full-dúplex, transmisión de datos comenzando por el Bit Mas Significativo (MSB, por sus siglas en inglés) o el Bit Menos Significativo (LSB, por sus siglas en inglés), entre otros, pueden variar dependiendo de las necesidades del proyecto en donde se aplicará.

Para sistemas embebidos en FPGA, que requieren altos niveles de confiabilidad y capacidad para adecuarse a aplicaciones a la medida, se requiere el diseño de las IP para diferentes interfaces, un caso especial es la interfaz SPI.

En el presente trabajo de tesis se describe el desarrollo de una IP para la interfaz SPI. El ambiente en donde se tiene pensada su aplicación está lleno de radiación ionizante elevada. Esta radiación en los FPGA puede generar desde un cambio temporal hasta uno permanente, se pueden generar cambios en los estados lógicos de bits no deseados o cambio en el mismo circuito descrito lo cual se interpretaría como un error. Para disminuir la probabilidad de errores es necesario aplicar técnicas de tolerancia a fallas al proyecto.

En este caso, el estado del arte se divide en dos partes. La primera corresponde a diferentes interfaces SPI ya sea en FPGA o microcontrolador, la segunda corresponde a las técnicas de tolerancia a fallas aplicados a sistemas espaciales.

2.1 Estado del arte de Protocolo de comunicación SPI en FPGA.

En [1] se describe la realización de un diseño de bajos recursos comparado con descripciones de código abierto en la red. Por desgracia, el autor no menciona el número de recursos lógicos utilizados, sin embargo, por ser una interfaz diseñada con el mínimo de recursos tiene únicamente lo necesario para la transmisión de datos y no más.

Se tienen parámetros configurables como son la frecuencia del CLK y SCK (señal de sincronización para la transferencia de datos entre dispositivos), y la configuración del reloj mediante CPOL y CPHA.

Un parámetro importante que utilizaron fue la frecuencia del reloj del sistema CLK el cual opera a 50 MHz mientras que la señal de sincronización del protocolo SPI llamada SCK

opera a 5 MHz. Para ser un protocolo de comunicación de alta velocidad, la transferencia de datos se da a una frecuencia muy baja ya que es 10 veces menor que la frecuencia del CLK, usualmente la frecuencia máxima del SCK es de 2 veces menor que el CLK el cual este no es el caso.

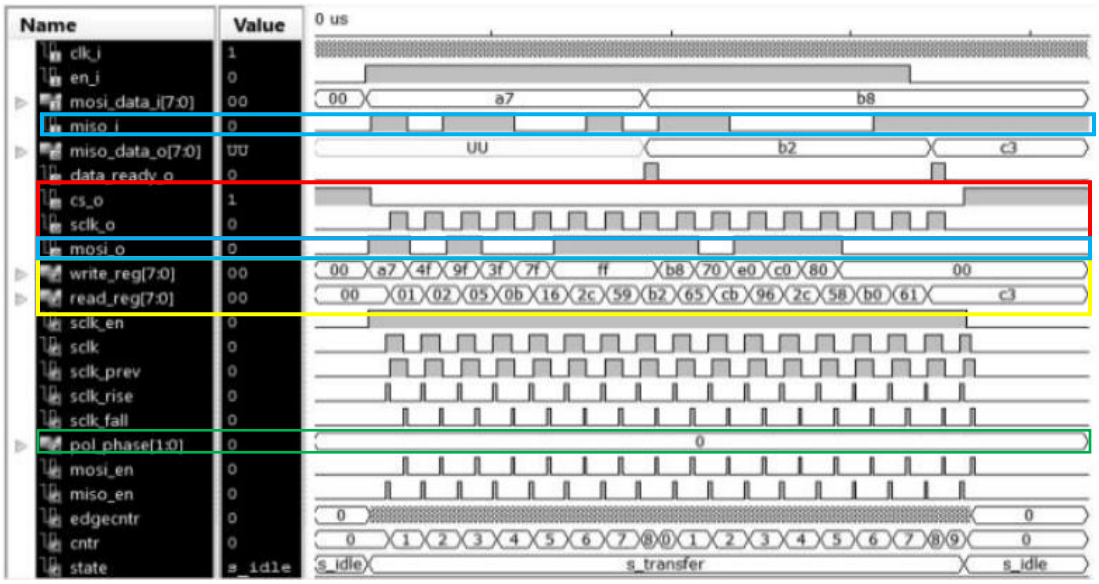


Figura 1. Resultados simulados tomados de [1]

La Figura 1 corresponde a la simulación de nuestra primera referencia bibliográfica [1], en donde se puede observar en el recuadro rojo dos señales muy importantes, el CS y el SCK, las cuales se encargan de indicar el comienzo y final de la transmisión y la sincronización de datos, respectivamente. En el recuadro amarillo se indican los valores que tienen los registros de corrimiento tanto de recepción como de transmisión. El recuadro verde muestra el valor de configuración de reloj que se utilizó, en este caso fue CPOL y CPHA igual a 0. Por último, en los recuadros azules se indica la entrada y salida serial de datos.

En [2] la interfaz SPI puede ser utilizada como maestro y como esclavo. Los registros utilizados son 53 en total, sin embargo, después de la síntesis se requieren 60, son pocos recursos para un protocolo tan complejo. Hablando de la frecuencia de operación, esta no es tan alta ya que el CLK opera a de 50 MHz, mientras que el SCK opera a 50 KHz, es decir, la transferencia de datos se da a una velocidad 100 veces menor que el CLK lo cual es muy poco eficiente.

Los resultados se compararon entre un microprocesador y un FPGA, la diferencia fue bastante grande, alrededor de 1 segundo o más, con una respuesta más atrasada en el microprocesador.

En [3] a pesar de no ser una aplicación en FPGA, el módulo SPI está muy bien detallado con respecto a sus componentes, por esto mismo se toma como ejemplo porque contiene de manera específica su composición interna.

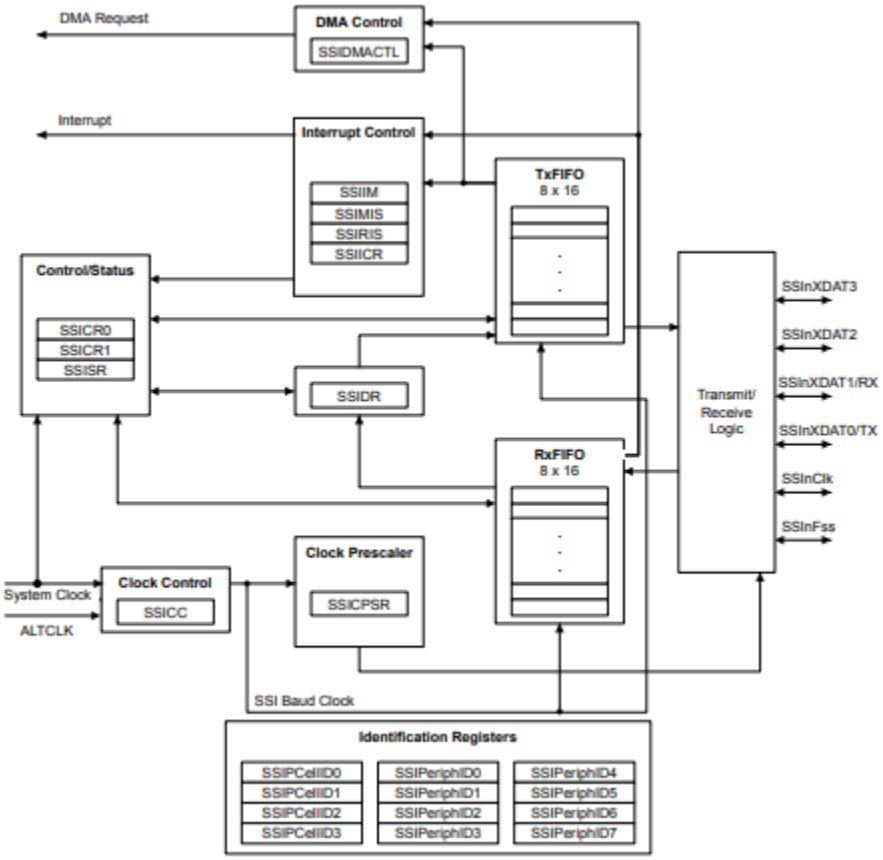


Figura 2. SPI en Tiva C TM4C1294NCPDT [3]

Este módulo no está diseñado para utilización en FPGA sino en un microcontrolador. Sus características son las siguientes:

- CPOL – CPHA
- Maestro- Esclavo
- Divisor de frecuencia.
    - Frecuencia del maestro (SCK) por lo menos 2 veces menor a la frecuencia del sistema
    - Frecuencia del esclavo (SCK) por lo menos 12 veces menor a la frecuencia del sistema
- Selección del tamaño de dato 4 – 16 bits
- Banderas de monitoreo
    - Empty FIFO Tx
    - Empty FIFO Rx
    - Full FIFO Tx
    - Full FIFO Rx

Puede alcanzar una frecuencia máxima del reloj SCK en modo maestro de 60 MHz y 10 MHz en modo esclavo. Es una desventaja que el esclavo trabaje a esta frecuencia como máximo, no se podría utilizar una comunicación full dúplex después de 10 MHz debido a que los datos recibidos por el maestro a mayor frecuencia se capturarían desfasados.

En [4] se quiere utilizar el protocolo de comunicación SPI a larga distancia, para poder lograr este objetivo es necesario combinarlo con SerDes (Serializador/Deserializador) el cual utiliza un medio de fibra óptica para la transferencia de datos, esto permitirá a SPI la transferencia datos a largas distancias. Los resultados obtenidos son correctos ya que se logró transmitir datos en un rango de frecuencias de 10 MHz hasta 50 MHz, se utilizaron dos diseños diferentes de SPI, el primero fue descrito en VHDL donde los resultados a 50 MHz tuvieron errores al momento de la transferencia mientras que en el segundo se utilizó una IP propia de Xilinx obteniendo resultados óptimos a la misma frecuencia. Las diferencias entre estos dos diferentes resultados son los recursos utilizados, el método menos confiable utiliza un menor número de recursos, mientras que la IP de Xilinx tiene un mejor rendimiento, pero los recursos se elevan.

En [5] se diseña el protocolo SPI en VHDL en estilo RTL. El funcionamiento tiene una frecuencia máxima de 100 MHz en modo maestro, sin embargo, se tiene la limitante de la frecuencia de operación del esclavo ya que no es posible la transferencia de datos en esta velocidad. Los retrasos detectados se dan en la transferencia del esclavo hacia el maestro ya que dependiendo de la distancia que existe entre ellos, disminuye la frecuencia máxima que puede utilizarse, esto debido a que la señal de reloj SCK tiene un mayor trayecto para poder actuar en el esclavo mientras que en el maestro el tiempo de propagación es significativamente menor, además de que el dato en la línea MISO tiene un retardo mayor por el mismo retraso del SCK agregando también la distancia entre dispositivos. Este problema no es resuelto completamente, pero hablan sobre la opción de agregar buffers a lo largo de las rutas de datos, agregando esclavos más rápidos o reducir la distancia entre dispositivos para poder aumentar la frecuencia máxima de operación sin errores.

Conclusiones del estado de arte de diseños de SPI.

Cada interfaz utilizada opera a diferentes frecuencias con respecto al reloj SCK, algunas tienen una eficiencia muy baja y otro llegan a operar a una alta velocidad sin ningún problema. Los aspectos importantes son la parte de la configuración mediante registros de control, como en [1] en el cual se tiene la posibilidad de variar la frecuencia de operación y observar en la simulación la trama correctamente funcionando con algunas otras señales necesarias para el diseño. Otra característica destacable es el problema del funcionamiento a frecuencias altas del SCK, ya que pueden presentarse desplazamientos de datos.

2.2 Estado del arte de Técnicas de tolerancia a fallas en FPGA.

Las técnicas de tolerancia a fallas ayudan a mitigar errores, en caso de los sistemas espaciales estos errores son debido al ambiente de radiación espacial en el que se encuentran.

En el artículo [6] se requiere lanzar el sistema al segundo punto de Lagrange Tierra-Luna (L2) utilizando un FPGA Kintex 7 de Xilinx. El objetivo de estas técnicas tolerantes a fallas es hacer uso de ellas con el mínimo de recursos ya que el sistema tiene recursos energéticos limitados, debido a esto se tiene un calentamiento, pero no un sistema de enfriamiento, por lo que es necesario ser eficiente en cuanto a la mitigación de errores.

Una parte importante para el funcionamiento correcto del sistema son los bits de configuración los cuales se protegerán con la técnica “scrubbing” la cual consiste en duplicar, comparar constantemente y reconfigurar bloques si es necesario. También se cuenta con un núcleo IP de mitigación de errores permitiendo la detección de errores de

hasta dos bits por palabra. Para eventos inesperados se utilizan técnicas tradicionales como es la redundancia modular triple (TMR, por sus siglas en inglés), la cual reduce significativamente la probabilidad de un error, el problema con ello es que solo enmascaran las fallas no las elimina.

En [7] se utilizan técnicas para disminuir el número de fallas lo mayor posible en un programa descrito con VHDL. Se consideran las fallas temporales y permanentes debido a los eventos que pueden ocurrir, ya sea el cambio de algún bit o daño permanente en la memoria configurable. También se clasificaron los errores que se pueden presentar desde el más sencillo donde solamente existe una falla transitoria, hasta el más complejo donde existen varias fallas permanentes junto con fallas transitorias que bloquean el sistema. Se utiliza una combinación de métodos para disminuir el porcentaje de error en todo el sistema, por ejemplo, la reescritura de código en memoria configurable no dañada junto con triple redundancia modular.

Estos métodos tienen buen resultado, pero no son 100% confiables ya que los errores más complejos necesitan una mayor investigación para los FPGA de bajo costo basados en SRAM como es el Virtex – 6 el cual se utilizó para este proyecto.

Por último, en [8] la misión necesita protección ya que trabajará en un microsatélite en órbita baja con componentes comerciales, de baja potencia y con alto rendimiento para las tareas a realizar. Dependiendo de la sensibilidad de los módulos del sistema se seleccionarán las técnicas a utilizar en cada uno de ellos, aumentando la confiabilidad para su operación en la órbita que se desea trabajar.

El sistema de protección consta de tres módulos. En el recuadro rojo se observa el primer módulo el cual triplica un área en específico para enmascaras una falla y mitigarla al momento, en el segundo módulo el cual se encarga de comparar las salidas del TMR el cual se observa en el recuadro amarillo, en dado caso de una falla doble ninguno de los módulos tendrá un correcto funcionamiento por lo que esto lleva al tercer módulo dentro del recuadro verde encargado de reconfigurar el sistema o áreas del sistema en específico cuando sea necesario. El esquema que representa los módulos se logra ver en la Figura 3.

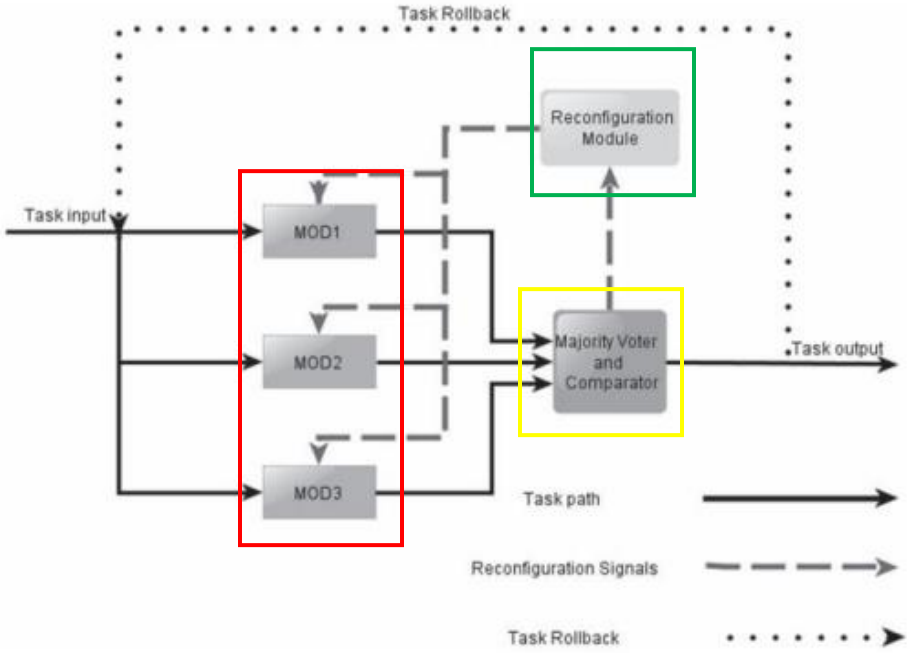


Figura 3. Esquema tolerante a fallas [8]

Conclusiones del estado del arte técnicas de tolerancia a fallas

Las técnicas de tolerancia a fallas que son aplicables en este tipo de sistemas pueden ir desde la más sencilla como es el TMR hasta algunas más complejas como son la reconfiguración del sistema. La técnica utilizada como modo de protección en caso de algún evento inesperado generado por el ambiente espacial es el TMR el cual ayuda a reducir el número de errores.

Hablando sobre la técnica, reconfiguración del sistema, esta es compleja ya que en caso de detectar alguna falla se reconfiguran algunos componentes o todo el sistema a modo que funcione sin errores nuevamente. En uno de los casos revisados se combinan la reconfiguración del sistema con el TMR para tener una mayor seguridad en cuanto a SEE

3. MARCO TEORICO

3.1 Descripción estructural en VHDL

VHDL es un lenguaje de descripción de hardware, su utilización primordial es describir circuitos digitales de una forma en que los humanos puedan implementar un diseño electrónico digital en un dispositivo electrónico programable.

VHDL presenta tres estilos de descripción de hardware:

- El primero está basado en describir cada componente del circuito con sus respectivas interconexiones, para tener conocimiento del más mínimo detalle junto con su funcionamiento siendo el más cercano a una implementación física, este estilo se conoce como “estilo estructural”.
- El segundo estilo es llamado “Flujo de datos”. Este estilo a su vez se presenta en dos formas, la primera describe el hardware mediante las funciones booleanas. La segunda forma describe el circuito mediante sentencias “WHEN-ELSE” el cual tiene un funcionamiento de condicionamiento más parecido al estilo comportamental o algorítmico.
- El tercer estilo consiste en describir el comportamiento requerido del circuito sin tener conocimiento interno de los componentes necesarios para su funcionamiento. Este estilo acorta el proceso de diseño ya que se encuentra en el nivel más alto de abstracción, este estilo es llamado “modo algorítmico”.

El estilo “flujo de datos”, el cual se caracteriza por describir señales y las conexiones con sus componentes mediante funciones booleanas se utilizará en el proyecto principalmente por dos razones. La primera va dirigida hacia el entorno en donde se será utilizada la interfaz debido a la radiación ionizante del espacio, esto puede generar cambios en las celdas de memoria encargada de las interconexiones del proyecto y así mismo cambiar alguna conexión o algún valor. Describir mediante funciones booleanas permite controlar a nivel compuerta la descripción del sistema y por lo tanto aplicar técnicas de tolerancia a fallas en circuitos perfectamente identificados. En el caso de una descripción algorítmica o flujo de datos donde se utilizan sentencias “WHEN-ELSE”, no es posible aplicar estas técnicas a nivel circuitos o componentes porque solo se conoce el comportamiento de la entrada y la salida más el diseñador no tiene control de los componentes que conforman el diseño.

La segunda razón tiene que ver con los recursos lógicos, en el caso del estilo “algorítmico” estos pueden ser superiores al de “flujo de datos” ya que en este se puede saber cuántos elementos lógicos se utilizan, pero no es posible controlar el diseño con menos recursos lógicos. En el caso del estilo “Flujo de datos” se conoce y se diseña cada componente o circuito lógico que se necesario utilizar haciendo un control más específico sobre el número de recursos lógicos.

3.2 Efectos de la radiación espacial en semiconductores

Para esta sección se consultó [10] en donde se habla sobre los diferentes tipos de fallas en FPGA debido a los SEE y técnicas para mitigar errores.

La principal influencia del entorno espacial en FPGA es principalmente los Eventos de Efectos Individuales (SEE, por sus siglas en inglés). SEE es el efecto de radiación resultante del impacto de partículas de altas energías (protones, iones pesados) en dispositivos electrónicos. Los diferentes SEE son los siguientes: Transitorio por Evento Único (SET, por sus siglas en inglés), Alteración por Evento Único (SEU, por sus siglas en inglés), Bloqueo por Evento Único (SEL, por sus siglas en inglés), Quemado por Evento Único (SEB, por sus siglas en inglés) y Ruptura de Puerta por Evento Único (SEGR, por sus siglas en inglés). Los dos primero son de estado transitorio y los restantes generan un daño permanente.

3.2.1 SET

El par electrón-hueco creado por la ionización da como resultado que el voltaje de salida de un dispositivo experimente una interferencia de pulso transitoria cuando la región sensible de un dispositivo CMOS, como un FPGA, es bombardeada por partículas de alto nivel de energía.

Para la lógica combinacional, el efecto de SET es temporal. Sin embargo, si la velocidad de la lógica combinacional es lo suficientemente rápida como para que el pulso transitorio se transmita a la lógica secuencial conectada con la lógica combinacional, el pulso podría capturarse y almacenarse para producir SEU. La probabilidad de ocurrencia depende del tiempo de llegada del pulso y de la relación entre el flanco ascendente y el flanco descendente (criterios para que se dispare un flip-flop).

3.2.2 SEU

Cuando una sola partícula de alto nivel de energía bombardea un Circuito Integrado (IC, por sus siglas en ingles), la ionización se producirá alrededor de la unión PN para crear una cierta cantidad de pares de electrones-huecos (portadores). Como resultado del efecto del campo eléctrico, la deriva y la recombinación de los portadores cambiarán la distribución y el movimiento de los portadores normales dentro del IC. Si el cambio es lo suficientemente grande, modificará el rendimiento eléctrico del dispositivo, de modo que el resultado será la falla del componente o circuito lógico (por ejemplo, la alteración de los datos en una unidad de almacenamiento, que se define como SEU). Por ejemplo, este fenómeno puede alterar los datos en una unidad de almacenamiento, como una memoria SRAM. Al pasar esto se provocarán cambios en la información almacenada, es decir, el valor almacenado en la unidad de almacenamiento experimentará SEU. Después del SEU, el valor almacenado defectuoso permanecerá hasta que se apague el sistema o se escriba un nuevo valor.

3.2.3 Single Event Latch-up (SEL)

El SEL es un fenómeno que puede ocurrir en dispositivos electrónicos, especialmente en circuitos integrados, cuando están expuestos a partículas cargadas de alta energía. Este se produce cuando una partícula cargada de alta energía impacta en una estructura semiconductor dentro del dispositivo, generando una corriente transitoria significativa, con posibilidad de desencadenar un mecanismo de realimentación positiva que resulta en un cortocircuito indeseado y persistente entre las conexiones de alimentación dentro del dispositivo. En otras palabras, el circuito se "bloquea" en un estado de cortocircuito.

Este cortocircuito puede causar daños irreparables en el dispositivo, como la destrucción de los componentes internos y la pérdida de funcionalidad.

3.2.4 SEB

Con partículas incidentes de alto nivel de energía, cuando una unión PN está polarizada en inversa, el proceso de deriva y aceleración de los portadores puede provocar la ruptura inversa de la unión PN, en la cual la fuente y el drenaje se conectan permanentemente hasta que el circuito eventualmente se quema.

3.2.5 SEGR

Cuando un dispositivo CMOS es bombardeado por una partícula cargada, se forma un pasaje de conducción de baja resistencia desde la rejilla hasta el sustrato a lo largo de la trayectoria de incidencia de la partícula, y se crea una corriente transitoria bajo el efecto del voltaje de la rejilla. Si la corriente transitoria es lo suficientemente alta, provocará una ruptura a lo largo del camino de corriente en la capa de óxido de la rejilla del dispositivo, formando un pasaje de conducción permanente desde la rejilla hasta el sustrato, lo que resultará en una falla total del dispositivo.

SEB y SEGR ocurren principalmente en dispositivos de potencia. La probabilidad de que ocurran SEB y SEGR en los FPGA es extremadamente baja.

3.3 Técnicas de tolerancia a fallas para FPGA basadas en SRAM

Para los FPGA basados en SRAM, las técnicas actuales que se pueden implementar en aplicaciones industriales para mejorar la confiabilidad del FPGA incluyen técnicas de mitigación dentro del diseño y técnicas de reconfiguración parcial. Estas dos técnicas pueden implementarse individualmente o en forma cooperativa, para obtener un mejor rendimiento de tolerancia a fallas de FPGA. En el presente trabajo las técnicas de interés son las que se encuentran dentro del diseño.

3.3.1 La técnica de tolerancia a fallas de redundancia modular triple (TMR)

La técnica TMR ha sido ampliamente implementada con resultado positivos, por lo mismo es muy recomendada y popular. Consiste en triplicar 1 o más módulos en su lógica combinacional y/o registros, agregando con un componente de votación por mayoría a la salida de cada módulo triplicado el cual emite el resultado final correcto. Como se muestra en la Figura 4, se presenta una falla en la lógica combinacional, en este caso el bit erróneo continuara alterando los resultados posteriores en donde sea utilizado. Para poder enmascarar este evento inesperado y tener un resultado final correcto la salida de cada módulo triplicado se ingresa al componente de voto mayoritario y el error deja de ser procesado, el detalle con el TMR es que no soluciona la falla, sino que solo mitiga los errores.

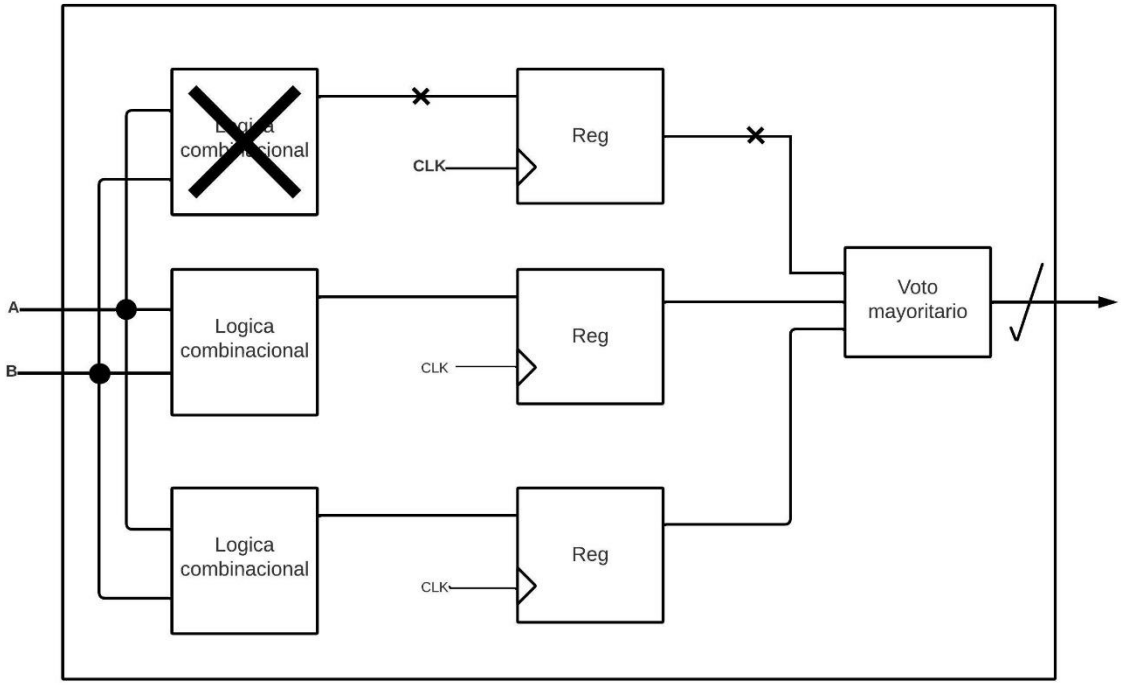


Figura 4. TMR

Con respecto a la Figura 4, la falla se presentó en la lógica combinacional pero también puede suceder en los registros, esto generaría el mismo resultado ya que el error es enmascarado en el módulo de Voto mayoritario.

Esta solución puede evitar errores en cierta parte del sistema ya que se protege tanto la lógica combinacional, así como los registros, es decir, se generan 3 bloques iguales con las mismas entradas de reloj y de datos, mientras que cada una de sus salidas son dirigidas a un componente de votación mayoritaria.

Existe una configuración de TMR más sencilla y es aplicable a los circuitos secuenciales. En la Figura 5 se muestra que solo se aplica la protección a los registros y a la lógica combinacional no, esto permite que al haber una falla en la lógica combinacional el dato defectuoso será introducido simultáneamente a los registros y no será posible detectar la alteración del dato ya que presentaran el mismo resultado dando un dato correcto en la salida mayoritaria

Este segundo esquema de protección es recomendable cuando el sistema ocupa más del 30% de los recursos del FPGA debido a que se requiere más hardware. En caso contrario, cuando es menor al 30% la utilización de recursos se opta por utilizar el TMR tanto para registros como para lógica combinacional.

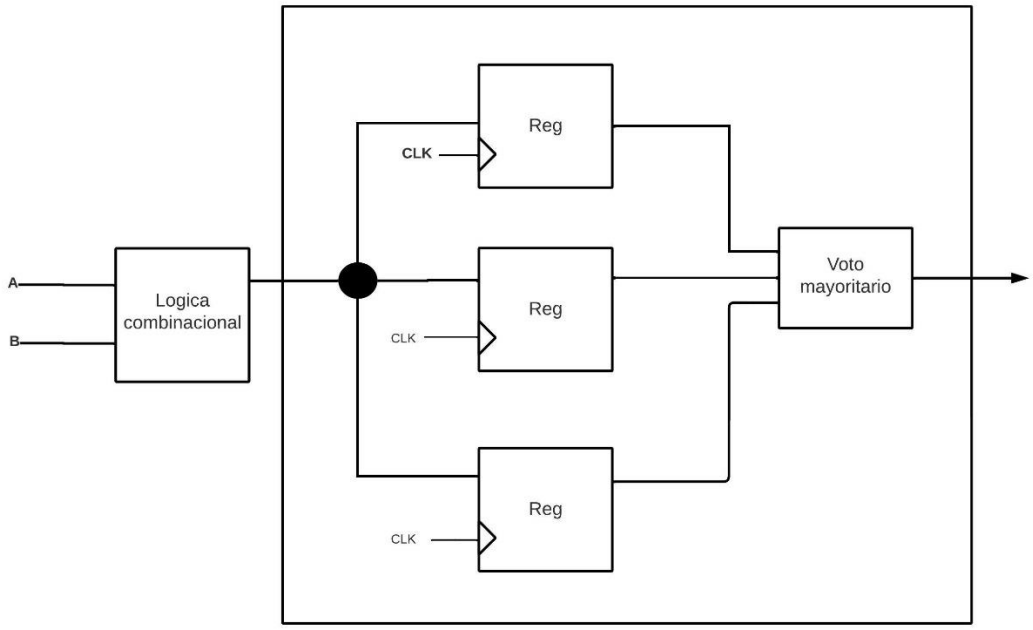


Figura 5. TMR a registros

3.4 AXI4

El bus de comunicaciones AXI4 es la cuarta generación de la especificación Arquitectura Avanzada de Bus de Microcontrolador (AMBA, por sus siglas en inglés), que permite la interconexión de bloques funcionales un Sistema en un Chip (SoC, por sus siglas en inglés). Las herramientas de desarrollo de Xilinx han utilizado este protocolo para facilitar la creación de diseños de SoC. El uso de estos buses tiene una variedad de ventajas, incluida una mejora en la productividad del diseño porque no es necesario aprender a usar varios protocolos de comunicaciones. Además, hay una gran variedad de IP Cores disponibles para su uso con esta conectividad.

“Existen tres tipos de interfaces AXI:

- AXI4 o AXI4-Full: Para realizar transferencias mapeadas en memoria con un rendimiento alto, permitiendo enviar ráfagas de datos en una misma transferencia.
- AXI4-Lite: Se trata de una simplificación del AXI4 para transferencias mapeadas en memoria, donde tan solo se pueden realizar transferencias simples, es decir, no permite el envío de ráfagas.
- AXI4-Stream: Para transferencias tipo *streaming* de alta velocidad donde un maestro transfiere datos a un esclavo. [9]

A continuación, se explicará con más detalle el funcionamiento de AXI4 y AXI4 Lite ya que funcionan de manera similar.

Estos protocolos requieren el uso de un canal de direcciones que indique la posición de memoria donde se desea realizar una lectura o una escritura. Esto permite la compartición de los canales de este tipo entre varios sistemas, como por ejemplo un uso típico que

consiste en compartir el canal de direcciones, cuya carga es menor que el canal de datos, y utilizar múltiples canales de datos.

Se dispone de los siguientes canales de transmisión:

- Write Data: Canal de escritura de datos para realizar una transferencia de datos de un maestro a un esclavo.
- Read Data: Canal de lectura de datos para transferir datos de un esclavo a un maestro. Este canal incluye los datos a transferir y una señal que indica el estado de la transferencia.
- Read Address: Canal de dirección de lectura. Incluye las señales necesarias para indicar la dirección de lectura de datos, y señales de control.
- Write Address: Canal de dirección de escritura. Incluye las señales necesarias para indicar la dirección de escritura de datos, y señales de control.
- Write Response: Canal de respuesta a las escrituras que indica el estado de la escritura, es decir, si se ha realizado correctamente o ha habido algún error

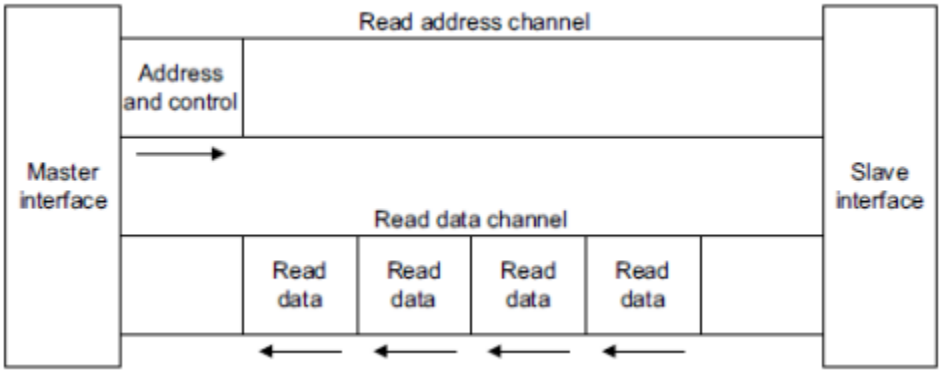


Figura 6. Lectura de datos en una interfaz mapeada en memoria.[11]

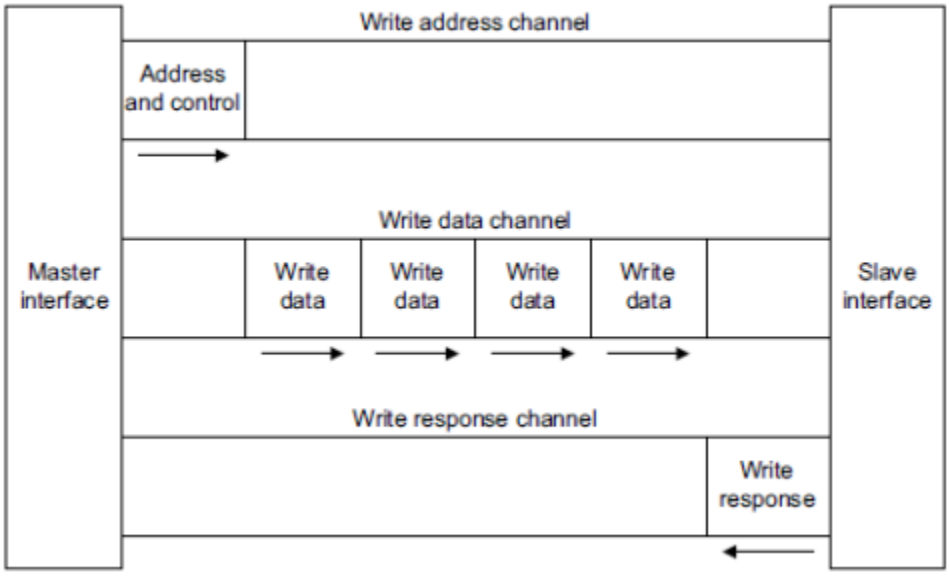


Figura 7. Escritura de datos de una interfaz mapeada en memoria.[11]

La Figura 6 y Figura 7 muestran una visión general tanto de una transferencia de lectura como una de escritura usando una interfaz mapeada en memoria, donde se pueden observar los cinco canales comentados anteriormente.

“AXI4-Lite es una versión reducida de AXI4, donde se simplifican las señales de los distintos canales, y tan solo se puede realizar una única transferencia con la misma dirección de memoria, es decir, no permite ráfagas de datos. Además, el tamaño del bus de datos máximo se reduce, pudiendo ser de tan solo 32 o 64 bits, mientras que en AXI4 puede ser de 32 a 1024 bits. “ [9]

3.5 Funcionamiento de bus SPI.

El estándar SPI es un protocolo de comunicación síncrono utilizado para la transferencia de datos entre dispositivos electrónicos. Fue desarrollado por Motorola en la década de 1980 y desde entonces se ha convertido en un estándar ampliamente adoptado en la industria.

Este bus emplea 4 señales para realizar comunicación con otro dispositivo. La conexión entre los dispositivos maestro y esclavo se muestra en la Figura 8.

Como nota adicional, el término “esclavo” en esta tesis será utilizado para hacer referencia al dispositivo o modulo que recibe las órdenes, no tiene ninguna connotación histórica o social. Esto se menciona para evitar algún mal entendido de discriminación.

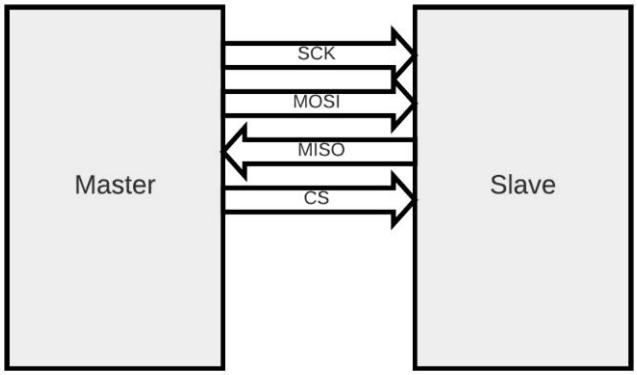


Figura 8. Conexión entre maestre y esclavo, SPI.

- SCK: Señal de reloj generada por el dispositivo maestro y empleada para sincronizar la transferencia del mensaje. Los datos se enviarán de manera secuencial y síncrona de forma que cada bit del mensaje permanezca en la línea de datos durante un ciclo de este reloj.
- MOSI: Línea de datos empleada para la transferencia de información del maestro al esclavo. Esta línea se conectará a todos los esclavos presentes en el bus de manera que el maestro pueda comunicar con todos ellos a través de una única línea.
- MISO: Conexión de datos utilizada para el envío de mensajes desde el esclavo hacia el maestro. Todos los dispositivos esclavos se conectan a la misma línea de modo que el maestro pueda recibir información de todos ellos a través de una sola entrada.
- CS: Línea de selección de esclavo gestionada por el dispositivo maestro. Cada

esclavo posee su propia entrada de selección mientras que el maestro posee una salida para cada uno de los esclavos conectados al bus. El maestro emplea esta línea para indicar al esclavo que se desea realizar con intercambio de información con él.

3.5.1 Descripción del funcionamiento

En este apartado se describirá el proceso para generar una transmisión entre un dispositivo maestro y un esclavo junto con los diferentes modos en los que puede realizarse.

Cuando se desea realizar una transmisión se realizarán de forma adecuada los siguientes pasos.

- Se debe seleccionar el esclavo con el que desea intercambiar la información colocando a nivel bajo el selector de chip (CS) correspondiente a dicho esclavo.
- El maestro debe activar el reloj de comunicación (SCK) a una frecuencia adecuada para la velocidad de transmisión deseada. Una vez activado el reloj, el maestro escribirá un bit en la MOSI en cada ciclo de reloj. En el flanco contrario al de escritura, el maestro leerá un bit de la línea MISO. Simultáneamente, el esclavo escribirá otro bit en la línea MISO en cada ciclo, leyendo el bit presente en la línea MOSI en el flanco de reloj contrario a la escritura.
- Cuando todos los bits hayan sido transmitidos en ambos sentidos, el maestro deshabilitará el reloj de comunicación (SCK).
- Finalmente, el maestro volverá a poner a nivel alto la línea de selección de chip para indicar al esclavo que la transferencia ha finalizado. En este momento el esclavo pondrá su salida de datos en alta impedancia para permitir transferencias de otros esclavos.

A continuación, se muestra un ejemplo de transmisión y recepción por parte del maestro.

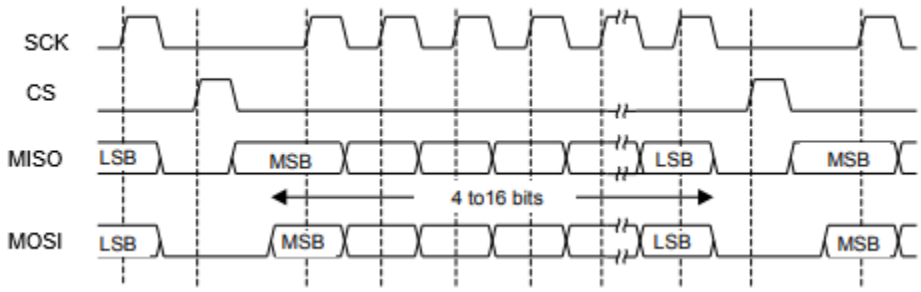


Figura 9. Transmisión de datos, Maestro.[3]

Se observa que los pasos requeridos se cumplen, en tanto se cambia de dato con el flanco negativo, mediante el flanco positivo se realiza la captura de bit. Es decir, se centra primero el dato y después se captura evitando errores en la transmisión.

3.5.2 Modos de operación

El bus SPI emplea 2 parámetros para definir el momento en el que se escriben o leen las líneas de datos, estos dos parámetros son

- CPHA: Define el retraso respecto al primer flanco de subida del reloj de sincronización.

  Cuando CPHA = 0: No hay retraso. El primer bit se leerá en el primer flanco de

subida del reloj, lo que implica que el bit debe ser escrito en la línea de datos previo al primer flanco de reloj, tanto en el caso del maestro como del esclavo.

Cuando CPHA = 1: El primer bit se leerá en el flanco de bajada del reloj sin importar el estado de reposo del SCK, lo que implica que el dato será capturado en cada flanco de bajada tanto para el maestro como para el esclavo.

- CPOL: Define el estado de reposo de la línea de reloj SCK, es decir, el nivel que tendrá la línea de reloj mientras que no se está realizando ninguna transferencia. Este parámetro puede tener 2 valores. Cuando CPOL = 0 la línea SCK permanecerá a nivel bajo mientras no se realicen transferencias y cuando CPOL = 1 la línea del reloj de sincronización se mantendrá a nivel alto cuando no se están efectuando operaciones de transmisión.

Para que el intercambio de información se realice de manera adecuada es necesario que el maestro y el esclavo tengan la misma configuración de estos parámetros. Analizando las posibles combinaciones, surgen 4 posibles modos de funcionamiento.

Modo 1: CPHA = 0 y CPOL = 0

En este modo la línea de reloj permanece a nivel bajo en reposo y la captura de datos se efectuará en flanco de subida.

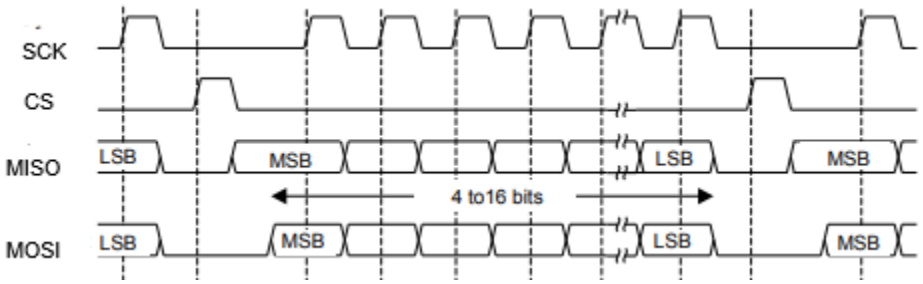


Figura 10. Transmisión de datos, CPHA = 0 y CPOL = 0 [3]

Modo 2: CPHA = 0 y CPOL = 1

Este modo se comporta de la misma manera que el anterior, con la diferencia del estado de reposo de la línea de reloj que permanecerá a nivel alto mientras que no se efectúen operaciones.

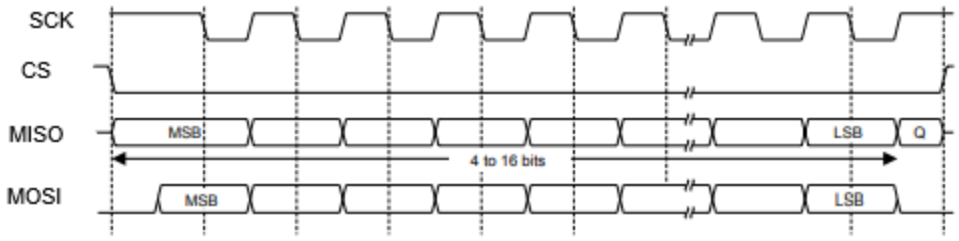


Figura 11. Transmisión de datos, CPHA = 0 y CPOL = 1 [3]

Modo 3: CPHA = 1 y CPOL = 0

Con esta configuración debe considerarse un retraso respecto a la línea de reloj SCK. La escritura se efectuará en flanco de subida, mientras que la lectura de datos se efectuará en los flancos de bajada del reloj, momento en el que el dato debe ser estable.

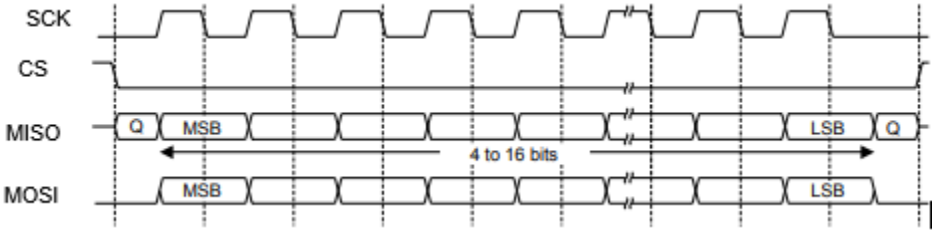


Figura 12. Transmisión de datos, CPHA = 1 y CPOL = 0 [3]

Modo 4: CPHA = 1 y CPOL = 1

Este modo es similar al anterior, variando de nuevo el estado de reposo de la línea SCK, que en este caso quedará fijada a nivel alto durante el reposo.

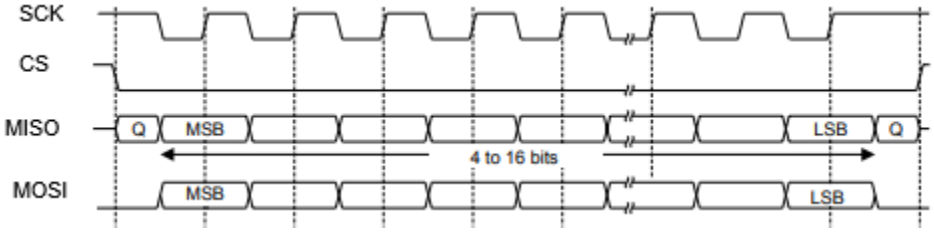


Figura 13. Transmisión de datos, CPHA = 1 y CPOL = 1 [3]

4. DISEÑO DE CONCEPTO DEL MODULO SPI.

4.1 Determinación de las especificaciones

La interfaz del protocolo de comunicación SPI debe de tener los siguientes parámetros

- Dos memorias FIFO, la primera dedicada a la transmisión de datos y la segunda a recepción
- Registros de control, de escrita y lectura.
- Divisor de frecuencia (SCK) con un máximo de 50 [MHz] y un mínimo de 1.5 [MHz]
- Control de la fase (CPHA) y polarización (CPOL) de reloj propio de la interfaz SPI (SCK)
- Líneas de control *Chip Select* para interactuar con dispositivos externos.
- Control en el número de bits máximo permitido
- Selección de modo maestro o esclavo
- *Reset* para utilización en caso de ser necesario.

4.1.1 Propuesta del diseño del módulo SPI

Se propone una interfaz con los siguientes elementos en general. Se observan las entradas y salidas que conforman a la interfaz mostrados en el primer o mayor nivel de abstracción con su respectivo número de bits de cada línea y su correspondiente nombre.

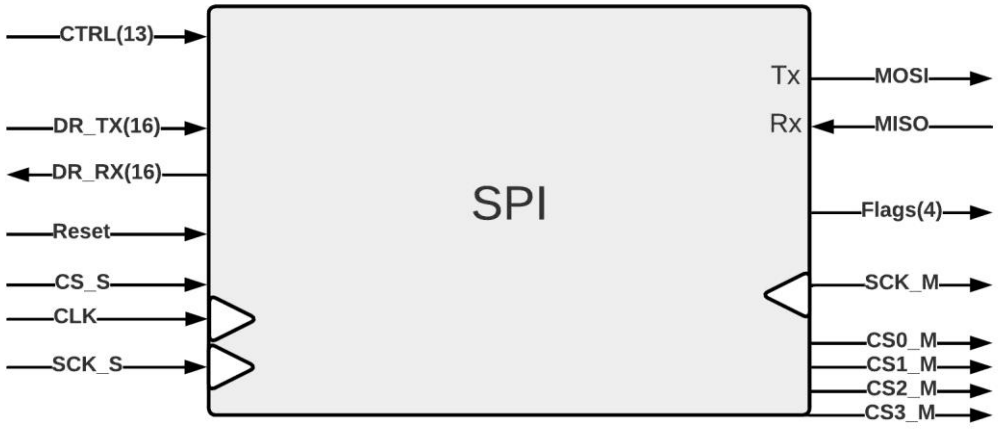


Figura 14. Primer nivel de abstracción, SPI.

A continuación, una descripción de cada una de las señales:

- CTRL → Esta señal tiene la principal tarea de escribir en los registros de control donde se encuentran la selección de modo y frecuencia de operación, numero de bits en el dato a transmitir y el seleccionador de esclavo (CS).
  Bits: 13
- DR_TX → Línea de datos que ingresan a la interfaz de manera paralela en un pulso de reloj (CLK) para poder ser trasmitidos a otro dispositivo.
  Bits: 16.

- DR_RX→ Línea de datos que salen de la interfaz de manera paralela en un pulso de reloj (CLK). El dato extraído anteriormente fue recibido a partir de la línea MISO.
  Bits: 16.

- Reset → Regresa a valores predeterminados todos los componentes.
  Normalmente los valores son '0'.
  Bits: 1.

- CS_*M → Línea de selección en caso de operar como maestro.
  El * indica un número del 0 al 3 debido a que estas señales tienen la misma función, solamente que se tienen 4 señales CS para utilizar debido a que el módulo SPI es multiesclavo.
  Bits: 1

- CS_S → Línea de selección en caso de operar como esclavo.
  Bits: 1.

- CLK → Reloj del sistema con el que funciona internamente la interfaz SPI.
  Bits: 1.

- SCK_M → Señal de reloj para transferencia de datos entre maestro y esclavo.
  Bits: 1

- SCK_S → Entrada de reloj propio del protocolo SPI en modo esclavo.
  Bits: 1

- MOSI → Flujo de datos del maestro al esclavo
  Bits: 1

- MISO → Flujo de datos del esclavo al maestro
  Bits: 1

- Flags → Banderas orientadas al monitoreo de las memorias FIFO correspondientes.
  - Empty_Rx→ Indica si la memoria FIFO de recepción se encuentra vacía.
  - Full_Rx → Indica si la memoria FIFO de recepción se encuentra llena.
  - Empty_Tx → Indica si la memoria FIFO de transmisión se encuentra vacía.
  - Full_Tx → Indica si la memoria FIFO de transmisión se encuentra llena.

  Bits: 4

4.1.2 Diseño Funcional

Después del nivel más alto de abstracción se obtiene el segundo nivel de abstracción mostrando los módulos funcionales que conforman el protocolo SPI.

En la Figura 15 se puede observar la utilización de registros DR, CTRL, el divisor de frecuencia, las memorias FIFO, registros de corrimiento y generador de CS. Todos estos componentes juegan un papel importante para su funcionamiento.

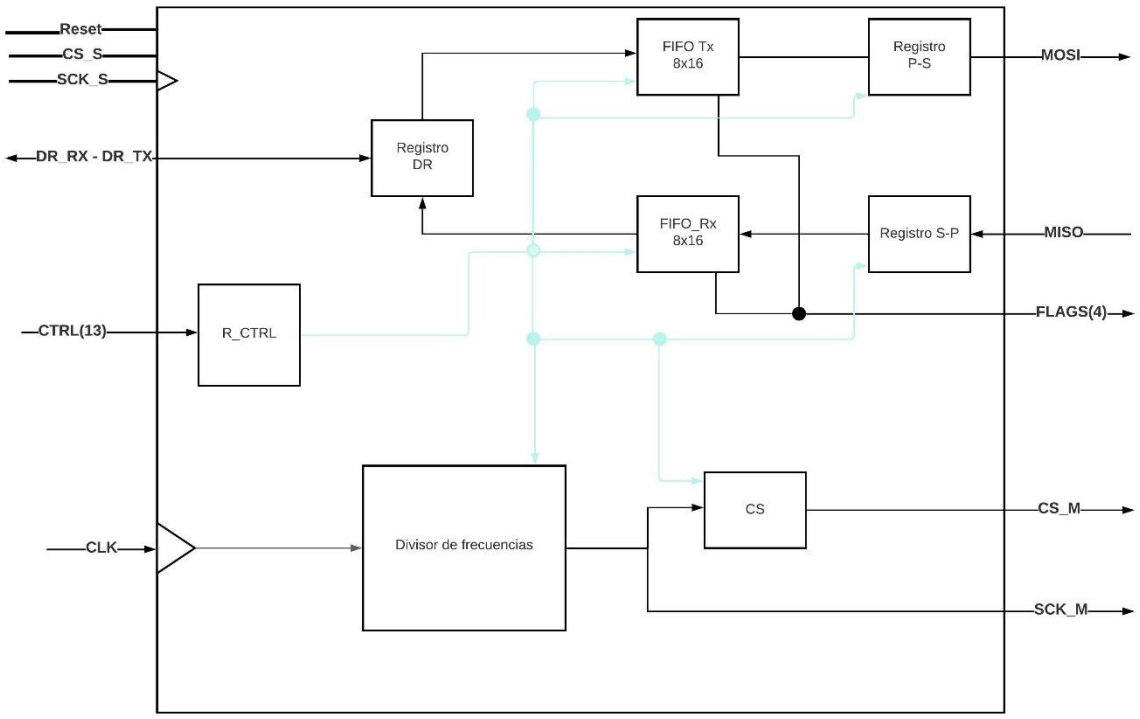


Figura 15 Segundo nivel de abstracción, SPI.

4.1.2.1 Funcionamiento de transmisión.

La transmisión de datos se ejecutará en varias etapas con la utilización de diferentes elementos dentro del módulo SPI. Para comprender como se da este proceso es necesario explicar paso a paso desde que se ingresa un dato hasta que la transferencia ha sido completada.

Primero se debe de ingresar una palabra con una longitud de 16 bits mediante el bus DR_TX hacia los registros DR. Posteriormente se almacenará en la memoria FIFO_TX la cual tendrá la condición de que, si se encuentra vacía el dato pasara a la siguiente etapa, si no es así solamente se almacenara. El dato será colocado en un arreglo de registros de corrimiento paralelo – serie (Registros P – S), se ingresarán de manera paralela de la memoria FIFO_TX a estos registros y se transmitirán en serie mediante la línea MOSI hacia otro dispositivo en sincronía con la señal SCK.

4.1.2.2 Funcionamiento de recepción

Al igual que en la transmisión, la recepción de datos se lleva a cabo de manera similar, pero con las etapas invertidas. Primero se comienza ingresando un dato serial de 16 bits mediante la línea MISO, el dato será almacenado en un arreglo de registros de

corrimiento serie paralelo (Registros S-P) en sincrónica con el SCK. Una vez terminada la recepción este dato se almacena en la memoria FIFO_RX, siempre y cuando no se encuentre llena. Por último, mediante una señal externa se indicará la extracción del dato inmediato siguiente de dicha memoria, el cual será direccionado a los registros DR para posteriormente leerlo fuera del módulo SPI.

4.1.2.3 Funcionamiento de registro de control.

Los registros de control almacenan una palabra de una longitud de 13 bits ingresada por el usuario, la cual dependiendo de su valor permitirá a la interfaz operar en diferentes modos. A continuación, se enlistarán los parámetros configurables mediante estos registros:

- Maestro – esclavo (MS): Configura a la interfaz en modo maestro o esclavo
- CPOL – CPHA: Permite elegir el valor de la polaridad y la fase de la señal de reloj SCK.
- Frecuencia de operación (CICLOS): Permite elegir la frecuencia de operación de la señal de reloj SCK. Esta operara en fracciones del CLK.
- Bits en cada palabra (DT): Configura el número de bits en cada palabra, tanto recibida como transmitida, será elegible dentro de un rango de 4 a 16 bits.
- Selección del CS (S_CS): Dado que la interfaz es multiesclavo, este registro permite configurar el esclavo con el que se desee intercambiar datos con tan solo elegir un CS.

Para realizar cambios en la interfaz se comienza introduciendo la palabra mediante el bus CTRL, esta se almacenará en los registros R_CTRL. Es importantes saber que las salidas de estos registros toman diferentes acciones dentro de la interfaz, dependiendo de su función, en la siguiente lista se muestran los elementos que modifican cada una de las señales de R_CTRL:

- MS: Divisor de frecuencia, CS, Registros P-S y Registros S-P
- CPOL – CPHA: Divisor de frecuencia, CS, Registros P-S, Registros S-P.
- CICLOS: Divisor de frecuencia.
- DT: FIFO_TX, FIFO_RX, CS, Divisor de frecuencia
- S_CS: CS

Al inicializar la interfaz se tendrá un valor de cero en R_CTRL, al momento de cambiar algún bit de las señales enlistadas anteriormente, los elementos que están ligados a ellas cambiaran su parte de su lógica interna para tener el comportamiento deseado.

4.1.2.4 Funcionamiento de divisor de frecuencia y CS.

El divisor de frecuencia es el encargado de generar la señal SCK con la ayuda de circuitos secuenciales y banderas. En el caso del CS, la señal principal es el SCK en complemento con banderas.

El proceso para generar el reloj de la interfaz SPI, denominado SCK, comienza con el ingreso de la señal de reloj CLK al divisor de frecuencias. Los pulsos de la señal CLK se cuentan regresivamente de ’n’ hasta 0, siendo ‘n’ un valor ingresado por el usuario mediante los registros de control, cuando la cuenta llega a 0 el nivel del SCK se invierte dando como

resultado otra señal de reloj con menor frecuencia. De esta manera la señal SCK es generada.

Posteriormente, en sincronía al SCK junto con otras condiciones se indica el comienzo y fin de la transmisión mediante el CS.

4.1.2.5 Funcionamiento del Reset.

La señal de Reset se ingresa por el usuario, dicha señal interviene en todos los flip-flop de la interfaz y los devuelve a su estado inicial. En la Figura 15 se observa que esta señal no se encuentra conectada a nada más que a la interfaz en si, esto debido a que interviene en todos los elementos sin importar su función.

A grandes rasgos su funcionamiento principal independientemente de la frecuencia de operación del SCK, la configuración del reloj, el número de bits a enviar en un dato se divide en dos maneras.

- Modo Maestro:
  - Escribir el dato a transmitir a los registros DR de escritura
  - El dato guardado en los registros DR se direccionará a la memoria FIFO de transmisión.
  - El dato será guardado en los registros de corrimiento P-S para poder transmitirlo y la FIFO permanecerá vacía hasta que se ingrese algún otro.
  - Una vez guardado se iniciará el proceso de intercambio de datos entre dispositivos generando de primera instancia el reloj SCK y manteniendo el CS en estado BAJO.
  - En cada pulso de reloj SCK se enviará un bit del dato. Al mismo tiempo que se trasmite se puede o no recibir datos desde el esclavo mediante la línea MISO.
  - Al terminar la transmisión, si es que se recibió un dato por parte del esclavo este es guardado en la memoria FIFO de recepción y posteriormente en cualquier momento que se desee puede ser extraído para su utilización.

- Modo esclavo:
  - Escribir el dato a transmitir a los registros DR de escritura
  - El dato guardado en los registros DR se direccionará a la memoria FIFO de transmisión.
  - Se esperará a el ingreso de la señal CS en estado BAJO.
  - Cuando el CS este de manera activa para poder trasmitir el dato el dato de la memoria FIFO de transmisión es guardado en los registros de corrimiento P-S.
  - En cada pulso de reloj SCK se enviará un bit del dato. Al mismo tiempo que se trasmite se puede o no recibir datos desde el maestro mediante la línea de recepción MOSI.
  - Al terminar la transmisión, si es que se recibió un dato por parte del maestro este es guardado en la memoria FIFO de recepción y posteriormente en cualquier momento que se desee puede ser extraído para su utilización.

5. DISEÑO DE CIRUCITOS SECUENCIALES Y COMBINACIONALES.

Los circuitos combinacionales descritos en esta sección son circuitos secuenciales que se utilizan en componentes que los requieren, como una memoria FIFO, un divisor de frecuencia, entre otros.

Los componentes internos están descritos en VHDL mediante estilo de descripción flujo de datos sin la utilización de sentencias WHEN_ELSE, es decir, en su mayoría cada parte del sistema este compuesto por sentencias booleanas y se conoce por completo como se encuentran configuradas las conexiones entre módulos, así como el interior de ellos.

5.1 Circuitos básicos

Los circuitos combinacionales mostrados a continuación son utilizados en diferentes puntos del programa. Se mostrará la configuración interna junto con una descripción de cada uno, también se incluirá una muestra de su funcionamiento. Se implementarán en el software QUARTUS II.

5.1.1 Flip-Flop D.

Un Flip Flop funciona con una configuración interna de compuertas, guardan un nuevo dato al momento en el que existe un flanco usualmente ascendente. El software utilizado no permite obtener el funcionamiento de un Flip-Flop con descripción flujo de datos en modo estructural, debido a esto son los únicos componentes descritos de manera logarítmica donde solamente se describe su comportamiento y no se conoce por completo la estructura interna.

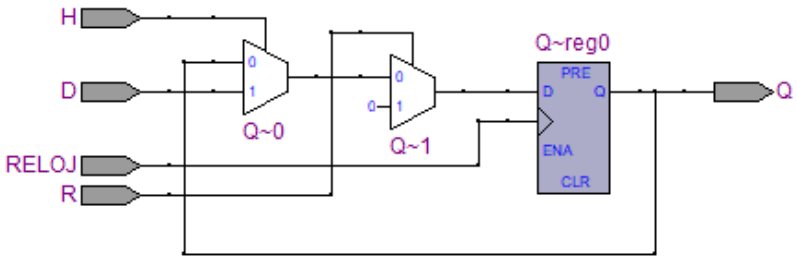


Figura 16 Flip Flop D

Se observa en la Figura 16, 4 entradas, 2 multiplexores, el registro D y una salida. La entrada H se encarga de darle paso al dato de entrada mediante un multiplexor, cuando H = ‘0’ no el valor de salida no tiene cambios, pero cuando H = ‘1’ el dato de entrada puede ingresar al registro en el siguiente pulso de reloj siempre y cuando R = ‘0’. En el caso de R= ‘1’ el registro presenta a la salida un ‘0’ sin importar el valor de la entrada o del habilitador.

```
Library IEEE;
Use IEEE.Std_logic_1164.all;

entity FFD is
port(CLK,D,R,H: in std_logic;
    Q: BUFFER std_logic);

end ENTITY;

architecture Behavioral of FFD is
begin
    PROCESS(CLK,R)
    BEGIN
    IF RISING_EDGE(CLK) THEN
        IF R='1' THEN
        Q<='0';
        ELSIF H='1' THEN
            Q<=D;
        ELSE
            Q<=Q;
        END IF;
    END IF;
    END PROCESS;
end Behavioral;
```

Figura 17. Descripción en VHDL de un Flip Flop D

Se puede observar a en la Figura 17 la descripción en VHDL de este elemento.

Este elemento tiene un funcionamiento síncrono, no genera cambios en la salida si no existe en la entrada de reloj un flanco ascendente y el valor del habilitador en alto. Este comportamiento se puede observar tanto en el esquema como en la descripción en VHDL.

Flow Status	Successful - Wed Jul 12 18:23:39 2023
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	SPI_PRINCIPAL
Top-level Entity Name	FFD
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1 / 18,752 (< 1 %)
Total combinational functions	1 / 18,752 (< 1 %)
Dedicated logic registers	1 / 18,752 (< 1 %)
Total registers	1
Total pins	5 / 315 (2 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 18. Reporte de compilación, Flip Flop D

En la Figura 18 se muestra el reporte de compilación del Flip Flop D.

5.1.2 Flip-Flop JK.

El Flip Flop JK tiene una pequeña similitud con el Flip Flop D, el habilitador y el *reset* funcionan de la misma manera, aunque en esta configuración debido al software no muestra los mismos multiplexores que en la Figura 16. La diferencia entre estos dos elementos radica en las entradas y el funcionamiento interno del registro, dependiendo del valor de las entradas J y K es el valor de salida.

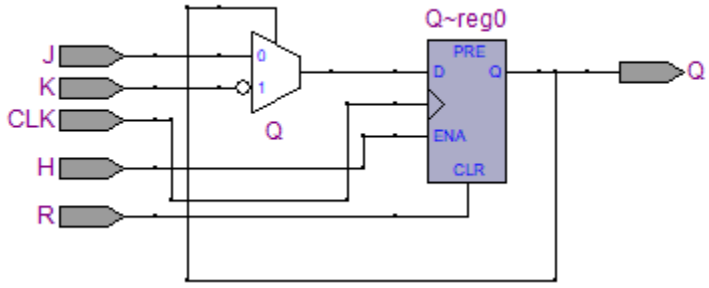


Figura 19. Flip Flop JK

En la Figura 20 se muestra la descripción en VHDL en modo algorítmico del FFJK.

```
LIBRARY IEEE;
USE IEEE.Std_logic_1164.all;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY FFJK IS
    PORT(J,K,CLK,H,R: IN STD_LOGIC;
    Q: BUFFER STD_LOGIC);
END FFJK;

ARCHITECTURE Behavioral OF FFJK IS
BEGIN
    PROCESS(CLK,R)
    BEGIN
        IF RISING_EDGE(CLK) THEN
        IF R='1' THEN Q<='0';
        ELSE
            IF H='1' THEN
                IF Q='0' THEN
                    IF (J='0') THEN Q<= '0';
                    ELSE            Q<='1';
                    END IF;
                ELSE
                    IF (K='0') THEN Q<= '1';
                    ELSE            Q<= '0';
                    END IF;
                END IF;
            END IF;
        END IF;
    END IF;
    END PROCESS;
END Behavioral;
```

Figura 20. Descripción en VHDL, flip-flop JK

El reporte de compilación muestra los elementos utilizados para poder efectuar el FFJK. Este reporte indica los elementos necesarios para generar el componente de la Figura 19.

Flow Status	Successful - Wed Jul 12 20:31:55 2023
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	SPI_PRINCIPAL
Top-level Entity Name	FFJK
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	2 / 18,752 (< 1 %)
Total combinational functions	2 / 18,752 (< 1 %)
Dedicated logic registers	1 / 18,752 (< 1 %)
Total registers	1
Total pins	6 / 315 (2 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 21. Reporte de compilación, FFJK

5.1.3 Multiplexor 4 a 1.

Un multiplexor depende de un seleccionador el cual se encarga dar acceso una entrada permitiéndole llegar a la salida.

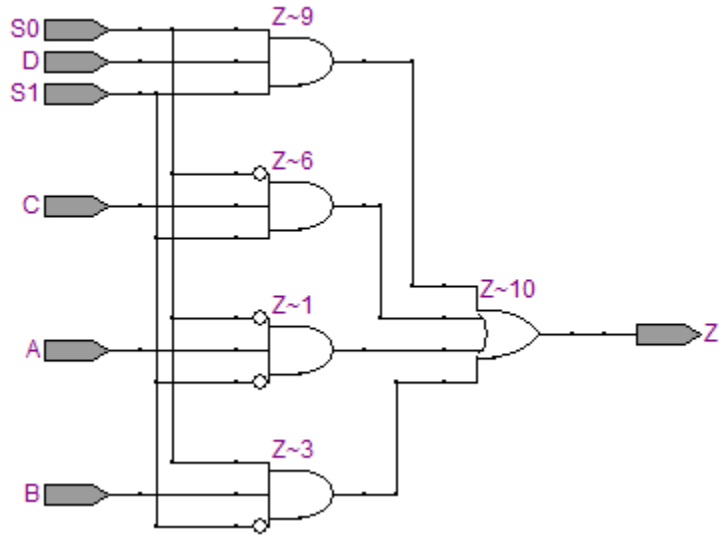


Figura 22. Multiplexor 4 a 1

Los dos bits del seleccionador S1 y S0 tienen una conexión mediante una compuerta AND con cada una de las entradas del multiplexor, esto genera que solamente una entrada pueda seguir el direccionamiento a la salida. Las salidas de las compuertas AND se

conectan a una compuerta OR, si cualquier entrada a esta última es ‘0’ la salida es cero y si alguna de ellas es ‘1’ la salida será ‘1’. En resumen, el seleccionador es el encargado de filtrar la entrada requerida mediante compuertas AND y la compuerta OR se encarga de reducir el número salidas a una.

```
Library IEEE;
Use IEEE.Std_logic_1164.all;

entity MUX4A1 is
port(A,B,C,D,S0,S1: in std_logic;
    Z: BUFFER std_logic);
end ENTITY;

architecture Behavioral of MUX4A1 is
begin

Z<= (A AND NOT S1 AND NOT S0) OR
    (B AND NOT S1 AND     S0) OR
    (C AND     S1 AND NOT S0) OR
    (D AND     S1 AND     S0);

end Behavioral;
```

Figura 23. Descripción en VHDL, Multiplexor 4 a 1

En la Figura 23 se encuentra la descripción en VHDL de un multiplexor 4 a 1. Con un mayor número de entradas y mayor número de bits en el seleccionador, pero con la misma configuración se puede generar un Multiplexor 16 a 1.

Flow Status	Successful - Thu Jul 13 12:43:03 2023
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	SPI_PRINCIPAL
Top-level Entity Name	MUX4A1
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	2 / 18,752 (< 1 %)
Total combinational functions	2 / 18,752 (< 1 %)
Dedicated logic registers	0 / 18,752 (0 %)
Total registers	0
Total pins	7 / 315 (2 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 24. Reporte de compilación, Mux4a1

En la Figura 24 se muestra el reporte de compilación donde se observa la utilización de dos elementos lógicos.

5.2 Circuitos secuenciales.

En esta sección se describirá el diseño, funcionamiento, interconexiones, entradas y salidas de cada circuito secuencial utilizado.

5.2.1 Contador 1 (Datos).

Este contador tiene un diseño de cuatro bits con una cuenta comenzando en cero y con la posibilidad de cambiar el fin de cuenta en un intervalo desde 4 hasta 15, es decir, la cuenta puede ir de 0 a 4, 0 a 5, 0 a 6, y así sucesivamente hasta 0 a 15. Se diseño este contador para cubrir los requerimientos de la interfaz ya que es necesario contar el número de bits que ingresan por la línea MISO y los que salen por la línea MOSI. Por ejemplo, si se desea tener una configuración de un dato de 6 bits, el contador tendrá una cuenta de 0 a 5, si la configuración se encuentra en 11 bits el contador tendrá una cuenta de 0 a 10 y así sucesivamente.

En resumen, el contador 1 servirá para contar los bits de recepción y transmisión que se reciban por las líneas MISO y MOSI, respectivamente.

Debido a que la tabla de verdad es muy grande solamente se describirá su funcionamiento y sus diferentes tipos de operación.

Se mostrarán 4 ejemplos de los 16 totales modos de trabajo del contador 1 en donde el reloj SCK_M1 hace referencia al uso de la interfaz en modo maestro con una configuración de CPHOL= 0 y CPHA= 0, este contador funciona de igual manera aun estando en modo esclavo y con una configuración de reloj diferente. La señal DT tiene una longitud de 4 bits con la función de indicar cuantos bits tendrá el dato a transmitir.

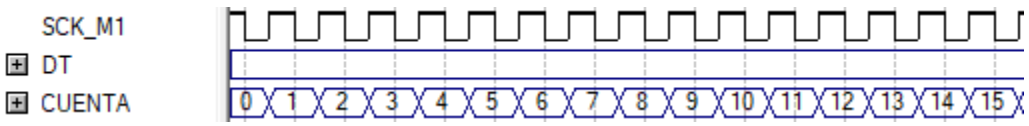


Figura 25. Contador 1, cuenta 0 – 15

El primer caso se le asigna un valor a DT igual a 0 en cada uno de sus bits, esto indica una cuenta comenzando en cero y terminando en quince (DT igual a cero es el predeterminado). De esta forma, al momento en que la cuenta llega a quince el valor siguiente es cero y vuelve a contar ascendentemente siempre y cuando el reloj esté en funcionamiento.

El funcionamiento mostrado en la Figura 25 es el mismo para un valor de DT de 0, 1, 2 y 3. Es decir, la cuenta tendrá una secuencia de 0 a 15 con estos valores de DT.

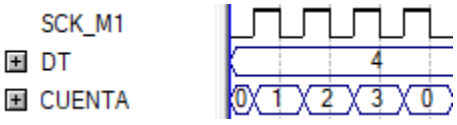


Figura 26. Contador 1, cuenta 0 a 3

En la Figura 26, cambia el valor de DT a 4 generando una cuenta de 0 a 3.

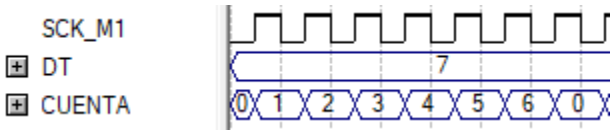


Figura 27. Contador 1, cuenta 0 a 6

De igual manera, en la Figura 27 se asigna el valor 7 a DT generando una cuenta de 0 – 6.

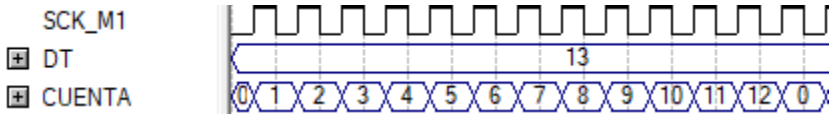


Figura 28. Contador 1, cuenta 0 a 12

Por último, en la Figura 28 al tener un valor de 13 en DT la cuenta comenzara en cero y terminara en doce.

Estos ejemplos indican el funcionamiento principal del contador 1. Para el número requerido de bits en el dato a transmitir, a la entrada DT se le asignara el valor de

Ecuación 1

$$DT = \#NumeroDeBits - 1$$

Es decir, al requiere un dato con 16 bits, son necesarios 16 pulsos de reloj dando lugar a un valor de DT igual a 15 para poder tener este comportamiento. Para un dato de 12 bits son necesarios 12 pulsos de reloj por lo que el valor de DT es igual a 11. Las tres excepciones de la Ecuación 1 es al momento en que DT tenga un valor de 0, 2 o 3 donde el número de bits a transmitir es de 16 por dato, por lo que la cuenta del contador para cada uno de estos valores es de 0 a15. Por propias especificaciones de la interfaz y fácil entendimiento de operación se optó por diseñarlo de esta manera.

5.2.2 Contador 2 (Filas de memoria FIFO_TX)

Para llevar un control del número de datos existentes en la memoria FIFO se realizó un contador ascendente/descendente de cero a ocho, siendo el cero la activación de una bandera indicando que se encuentra vacía la memoria y el ocho la activación de otra bandera indicando memoria llena. A diferencia del contador 1, el contador 2 tiene la propiedad de aumentar o disminuir en uno, en el caso de tener la cuenta en 0 no se puede disminuir aún más y al tener un valor de 8 no es posible aumentar a un valor más alto.

Figura 29. Entidad, contador 2.

El contador 2 está construido por cuatro Flip-Flop JK y una configuración de compuertas obtenidas a partir de la Tabla 4 ubicada en la sección de anexos. Contiene un habilitador el cual al tener un estado ALTO hace funcionar el contador, este habilitador se activa al momento en que ingresa o se extrae un dato de la memoria FIFO.

La señal QSCK2 es una señal encargada de indicar cuando y en qué dirección (asc/dsc) cambiara la cuenta. Si se encuentra en alto la señal el contador disminuirá ya que esto indica que un dato será extraído, en caso contrario aumentará si es que existe un dato para ingresar a la memoria. En el caso del habilitador, este activa al contador al momento en que QSCK2 se encuentra en estado ALTO o existe algún dato en los registros DR de transmisión listo para ser almacenado en la memoria FIFO_TX.

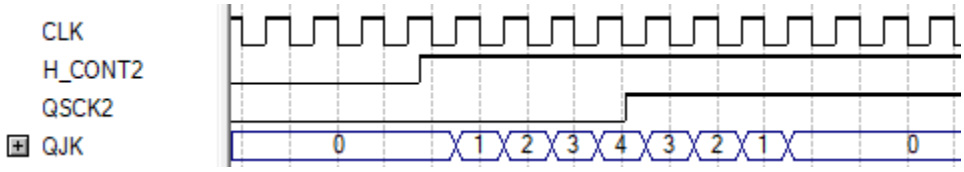


Figura 30. Simulación de contador 2

En la Figura 30 se observa la respuesta del contador. El habilitador H_CONT2 mientras se encuentre en estado BAJO restringe la actividad del contador y permanece en cero, si el contador se encontrara en algún otro valor permanecería sin cambios hasta que el habilitador tenga un valor ALTO. El QSCK2 tiene dos funciones. La primera función indica cuándo se desea ingresar un dato en la memoria, mientras que la segunda función controla la dirección del contador, ya sea en modo ascendente o descendente.

De igual manera, en la figura anterior se observa que el contador realiza una cuenta ascendente cuando QSCK2 se encuentra en estado BAJO y descendente con QSCK2 en estado ALTO. En términos del funcionamiento principal del contador, cuando se encuentra en modo ascendente quiere decir que hay datos ingresando a la memoria FIFO. Por el contrario, en modo descendente, los datos se extraen de la memoria.

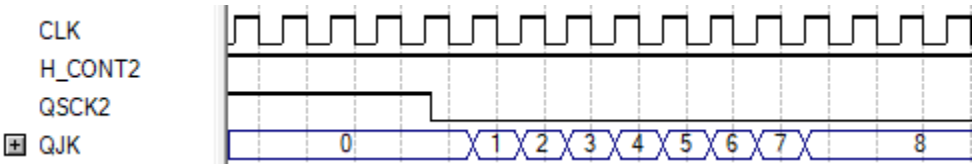


Figura 31. Simulación de contador 2, máximo y mínimo

En la Figura 31 se observan las dos excepciones del contador. Al encontrarse en el valor mínimo, estar habilitado y en modo descendente (H_CONT2 = '1', Q_SCK2 = '1') no cambia el valor y permanece en cero. La segunda excepción es en modo de operación ascendente (QSCK2 = '0'), al tener un valor de 8 la cuenta no sube o se reinicia porque la única opción disponible es disminuir ya que este es su valor máximo para el cual fue diseñado.

5.2.3 Contador 3 (Filas de memoria FIFO_RX)

El contador 3 tiene gran similitud con el contador 2 ya que su funcionamiento se basa en una cuenta en sentido tanto ascendente como descendente desde cero hasta ocho, indicando en cero y ocho la activación de banderas encargadas de monitorear si se encuentra vacía o llena la memoria FIFO, respectivamente. Existe una diferencia en la señal QSCK2, al tener un estado ALTO de esta señal el sentido de la cuenta será ascendente mientras que en estado BAJO será descendente, lo contrario al contador anterior.

El más alto nivel de abstracción puede observarse en la Figura 29, parecería ser el mismo contador, pero su funcionamiento y forma de operación son muy diferentes. La Tabla 5 de la sección de anexos se utilizo para el diseño general del contador en donde tiene un comportamiento inverso con respecto a QSCK2 en comparación al anterior.

Su principal función es llevar un conteo de los datos totales dentro de la memoria FIFO de recepción, aumentar la cuenta cuando un dato ingresa o disminuirla cuando se extrae.

Figura 32. Simulación del contador 3 - ascendente

En la Figura 32, se observan un conteo ascendente siempre y cuando el habilitador tenga un estado ALTO, en caso contrario el conteo no sufre ningún cambio como se puede ver de igual manera en la Figura 33 y 30 en el recuadro rojo. Con la señal QSCK2 en estado BAJO se puede observar que no hay cambios una vez se encuentre en cero, sin embargo, al estado contrario la cuenta es ascendente.

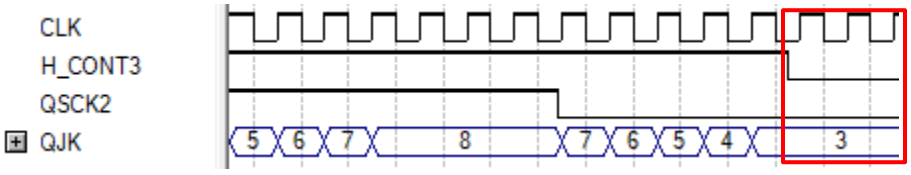


Figura 33. Simulación del contador 3 - descendente

En la Figura 33, el tope máximo de cuenta en modo ascendente es 8 por requerimientos del proyecto. De igual manera la cuenta ascendente comienza al momento en que QSCK2 se encuentra en estado BAJO.

5.2.4 Contador 4 (Divisor de frecuencia)

El contador 4 es utilizado en el divisor de frecuencia para poder generar el reloj SCK a partir del CLK. Tiene un funcionamiento donde la cuenta tiene un comportamiento descendente y el inicio de esta no es desde cero si no cualquier otro valor menor o igual a 31 dependiendo la frecuencia deseada

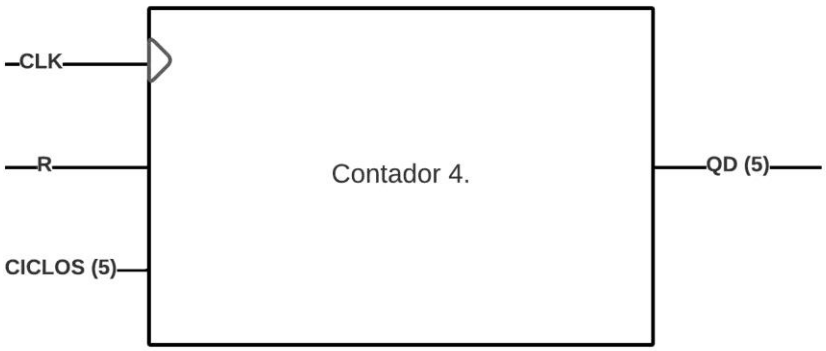


Figura 34. Entidad - contador 4.

En la Figura 34 se observa el más alto nivel de abstracción del contador 4. El reloj CLK es con el que funcionará internamente, tendrá también un *Reset* el cual regresará los valores del contador a cero. Por último, el inicio de cuenta se indica mediante CICLOS, si a esta señal se le asigna un valor de 25, la cuenta comenzara desde 25 $(11001)_2$ hasta cero, si el valor asignado es de 4 $(00100)_2$ la cuenta comenzara en 4 hasta cero. En el caso de tener un valor de cero, el contador no tendrá actividad ya que no se generará ningún cambio.

En la Tabla 6 de la sección de anexos se tiene la tabla de estados de este, esta tabla fue llenada a partir de la tabla de transición del Flip Flop D y utilizada para poder diseñar el contador mediante mapas de Karnaugh para cada entrada de Flip Flop D.

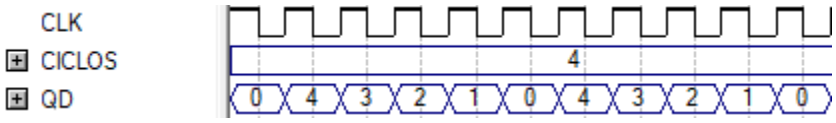


Figura 35. Simulación del contador 4 - Inicio de cuenta en 4.

En la Figura 35 se observa el valor de CICLOS en 4, QD son las salidas de los 5 Flip Flop del contador los cuales tienen una cuenta descendente donde su valor inicial es 4 y su valor final es 0.

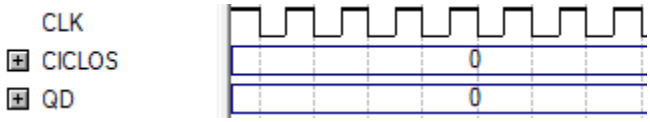


Figura 36. Simulación del contador 4 - Valor de CICLOS en 0.

En la Figura 36 debido al valor de CICLOS de cero, no existe como tal una cuenta descendente y en el diseño del contador este caso en particular se tomó en cuenta debido a que resulta útil su aplicación en el divisor de frecuencia.

5.2.5 Contador 5 (QSCK2 Contador 2)

Recordando la función de la señal QSCK2 del contador 2 encargada de indicar el sentido de la cuenta siendo ascendente en estado BAJO y descendente en estado ALTO. Para poder generar esta señal se tuvo que diseñar un contador de dos bits ya que depende de condiciones dentro de la memoria FIFO y algunos otros registros dentro de la interfaz SPI.

La señal QSCK2 funciona con normalidad en estado bajo incluso puede permanecer así durante un periodo largo de tiempo siempre y cuando no haya transmisión de datos. Para que esta señal cambie a estado ALTO es necesario que tanto en los registros de corrimiento como en la memoria FIFO_TX sea igual el dato, también es necesario que el contador 1 se encuentre en un valor de 1 $(0001)_2$ y la última condición es que el CS se encuentre en estado BAJO, en caso contrario QSCK2 no podrá cambiar su valor y permanecerá en estado bajo.

Para poder entender su diseño de una manera concreta se puede consultar la Tabla 7 en la sección de Anexos. La señal UNO indica cuando el contador 1 tiene como valor un 1 a su salida, la señal DSCK2 indica cuando el valor del dato inmediato de la memoria FIFO_TX es igual al del registro de corrimiento correspondiente, siendo A y B los estados actuales y por último B+ es la señal QSCK2 y A+ solamente es utilizada evitar una réplica de la señal QSCK2 cuando se cumplan las condiciones en más de un pulso de reloj (A+ y B+ indican el estado siguiente).

Su funcionamiento principal consta en una cuenta desde cero hasta dos esencialmente, la mayor parte del tiempo se encontrará en cero, cuando las condiciones anteriores se cumplan comenzará la cuenta, la señal QSCK2 en estado ALTO es la misma que tener en este contador un valor de “01” lo que generara una disminución en el contador 2, al siguiente pulso de reloj se quedara estático el contador en un valor de 2 $(10)_2$ hasta que el CS se encuentre de nuevo en estado ALTO generando un reincio de cuenta en el contador 5. En caso de un error donde el contador tengas sus dos valores de salida en ‘1’ el siguiente estado lo condicionara hacia 2 $(10)_2$.

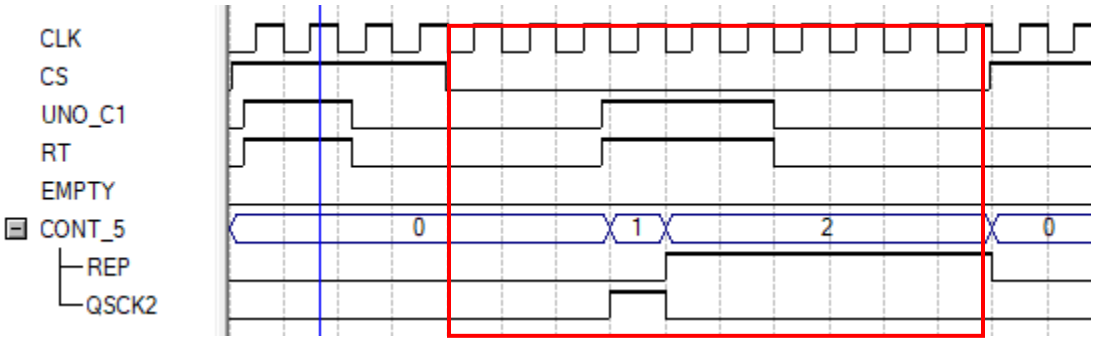


Figura 37. Simulación del contador 5

En la Figura 37 se logra observa en el recuadro rojo que al momento de tener las condiciones necesarias para poder iniciar la cuenta. Una vez que se cumplen las 4 condiciones (CS = ‘0’, UNO_C1 = ‘1’, RT= ‘1’, EMPTY = ‘0’) la cuenta comienza y se

mantiene en un valor de 2 hasta que el CS cambie de estado sin importar el estado de las demás señales.

Este funcionamiento ayuda a extraer un solo dato de la memoria FIFO_TX sin importar que tan largo sea el tiempo en que se cumplan tales condiciones.

5.2.6 Contador 6 (Extraer dato de FIFO_RX)

Este contador tiene un funcionamiento parecido al anterior, su principal diferencia son las señales externas de las cuales depende, en este caso solamente en una y en el anterior eran cuatro. De igual manera lleva una cuenta desde cero hasta dos. El contador permanece en con un valor de 0 siempre y cuando la señal externa, la cual identificaremos como EXT, se encuentre en estado BAJO. Una vez cambie de estado esta señal el contador comenzará una cuenta en donde al momento de tener un valor de 1 $(01)_2$ se indicará la extracción de un dato en la memoria FIFO_RX.

Cuando el contador indique la extracción durante un pulso de reloj, al siguiente pulso este contador tendrá un valor de 2 $(10)_2$ el cual tendrá el objetivo de no duplicar la instrucción de extraer dato. Después de este ciclo de cuenta el valor del contador 6 regresara a 0 $(00)_2$ esperando de nuevo el estado en ALTO de la señal EXT para un nuevo ciclo de cuenta.

En el Anexo 2 se observa la tabla del estado siguiente de este contador.

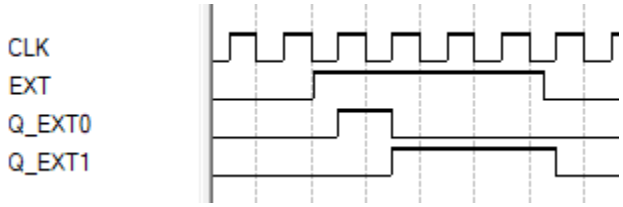


Figura 38. Simulación del contador 6.

La cuenta comienza una vez que la señal EXT se encuentra en estado ALTO al pulso de reloj siguiente. De igual manera, en el momento en que Q_EXT0 tenga un nivel de '1' se indicara la extracción de un dato en la memoria FIFO_RX, después de este estado se continuara a permanecer la cuenta en 2 (Q_EXT0 = 0, Q_EXT1 = 1). Este funcionamiento tendrá una gran importancia al momento de extraer e ingresar datos a la memoria ya que solamente se puede realizar una acción a la vez.

5.2.7 Contador 7 (Ingresar dato a FIFO_RX)

Al momento de ingresar un dato a la memoria FIFO de recepción es necesario un contador para evitar el duplicado del dato que se ingresará. Esto debido a la trama del protocolo SPI, se puede guardar el dato una vez que se haya recibido por completo, es decir, cuando el CS se encuentre en estado ALTO después el proceso de transmisión.

Otro aspecto para tomar en cuenta es el que no se puede extraer e ingresar un dato al mismo tiempo, por lo tanto, se le dará prioridad a la extracción en caso de que haya datos en la memoria FIFO de recepción en necesario su uso inmediatamente, mientras que el ingreso del dato lleva más tiempo y no puede ser en cualquier momento como se indica anteriormente.

Como en los contadores 5 y 6, este tiene el mismo funcionamiento, pero con un diseño diferente. En el Tabla 2 se puede ver su tabla de estados la cual fue utilizada para su diseño

con los Flip-Flop D. La señal EXT de la tabla hace referencia a la señal Q_EXT0 del contador 6.

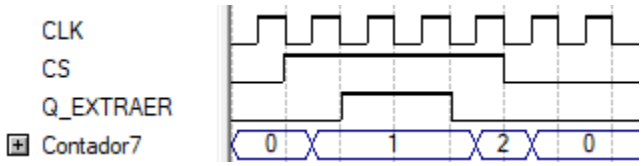


Figura 39. Simulación del contador 7 - Extrayendo dato al mismo tiempo que ingresando

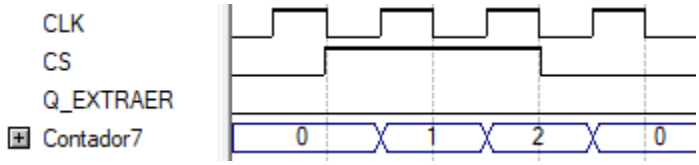


Figura 40. Simulación del contador 7 - Ingresando dato

En la Figura 39 y Figura 40 se observan los dos casos posibles en donde el contador puede operar.

- El primer caso muestra la situación en donde es momento de ingresar un dato cuando el CS cambia de estado BAJO a ALTO, pero inmediatamente se necesita extraer un dato ya existente en la memoria FIFO, se espera a ser extraído el tiempo que sea necesario y al siguiente pulso de reloj cuando la extracción haya terminado se ingresa el dato recibido desde la línea MISO.
- El segundo caso muestra la operación en donde el ingreso y la extracción no suceden al mismo tiempo y en cada pulso de reloj siempre que haya un estado ALTO del CS hay un aumento en el contador.

Cabe recalcar que el contador 6 y el contador 7 tienen la función de habilitar el contador 3 siempre que alguno de los dos tenga un valor de 1 $(01)_2$. Esto debido a que no todo el tiempo se van a ingresar datos y no todo el tiempo se van a extraer.

6. DISEÑO DE BLOQUES FUNCIONALES PARA CONTROL Y DIRECCIONAMIENTO DE DATOS

El protocolo SPI para una transferencia de datos utiliza cuatro líneas externas. Mientras que los componentes internos no son definidos como tal, se tiene libertar de poder manipular el direccionamiento de datos de diferentes formas, en este caso la guía a seguir es el diagrama mostrado en la Figura 15.

En esta sección se describirá el procedimiento del diseño de cada componente fundamental propuesto en la sección 4.

6.1 Memoria FIFO

6.1.1 Memoria FIFO de Transmisión (FIFO_TX)

Las memorias FIFO utilizadas para guardar los datos que se transmitirán o recibirán tienen la característica en donde solamente se puede leer o escribir, pero no ambas. La capacidad máxima de datos almacenados para la memoria tanto de transmisión como de recepción es de 8 palabras binarias con un mínimo de 4 bits y un máximo de 16 bits por palabra.

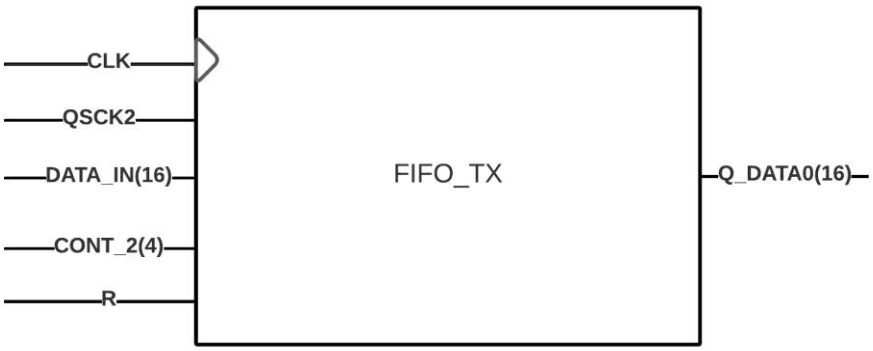


Figura 41. Memoria FIFO de transmisión - Entidad.

En la Figura 41 se observan las entradas y salidas de la memoria. Como entrada se tiene el reloj principal CLK, QSCK2 el cual está asociada al contador 2, de igual manera el contador 2 es utilizado para el conteo de datos, R como reset y por último DATA_IN la cual es el bus de datos de entrada de 16 bits. En la salida se tiene solamente Q_DATA0 el bus de datos de salida con 16 bits.

A continuación, se tendrá una explicación del funcionamiento de la memoria FIFO.

1. Si la memoria FIFO se encuentra vacía, el contador 2 tendrá un valor de cero. Así mismo si no hay dato a ingresar se quedara en cero.

2. Una vez ingresado el primero dato, el contador 2 aumentara en uno. Se pueden ingresar datos a la memoria FIFO de transmisión siempre y cuando la señal QSCK2 se encuentre en bajo.
3. Al momento de ingresar un dato a la memoria el contador 2 aumentará en uno, al extraer el dato disminuirá en uno dicho contador.
4. La memoria FIFO puede almacenar 8 datos en total, cuando la memoria se encuentra llena el contador 2 indirectamente indica que ya no pueden ingresar más datos hasta que se extraiga el siguiente.
5. En cuanto el dato ingrese a la memoria se iniciará el proceso de transmisión de datos como lo dicta el protocolo SPI. Si se ingresa un dato más mientras el primero aún no se termina de transmitir, este permanece en la memoria hasta haber completado él envió de dato anterior.
6. Cuando un dato es extraído de la memoria, todos los datos existentes dentro de esta son desplazados a la siguiente fila según corresponda. Es decir, el dato de la fila 6 es desplazado a la fila 5, el dato de la fila 5 es desplazado a la fila 4 y así sucesivamente. Cabe resaltar que la fila 0 contiene el siguiente dato a transmitir.
7. En caso de tener a la entrada un dato igual a cero no se ingresará a menos que el bit más significativo de los registros DR se encuentre en '1'.
8. Cuando se tiene simultáneamente el ingreso y extracción de un dato de la memoria, se prioriza la extracción mientras que el dato a ingresar permanece almacenado en los registros DR hasta que la extracción se complete.

La estructura utilizada en esta memoria debido a su aplicación es solamente un contador y 128 registros, la memoria está hecha para 8 palabras binarias de 16 bits cada una, es decir, es una memoria de 8x16. En este caso se utiliza el contador 2 para poder llevar un control del almacenamiento de datos.

En la Figura 42 se observa la configuración detallada de la memoria FIFO de transmisión. Los puntos suspensivos indican repetición de los registros y sus conexiones. El recuadro rojo representa todas las filas sin contar a la última, esto porque la configuración de compuertas es la misma y solo cambia el valor de DATA_IN y Contador2. Solamente se podrá habilitar una fila completa cuando el contador2 tenga algún valor de uno a siete. Por ejemplo, si el contador tiene un valor de seis quiere decir que hay cinco datos en la memoria y el siguiente dato a ingresar se guardara en la fila número seis. Al momento de extraer un dato, se activan los 128 registros de la memoria y los datos se recorren como se indica en el punto 6 de la explicación anterior.

De manera más técnica, la señal QSCK2 se utiliza como seleccionador, cuando se encuentra en estado alto se indica una extracción de un dato de la memoria, al extraer el dato esto también provoca un corrimiento en paralelo de los datos que se encuentran en la fila siguiente hacia la fila que se desocupó. QSCK2 con estado bajo ingresa un nuevo dato externo mediante DATA_IN pero solamente en la fila correspondiente al valor del contador 2, es decir, al tener un valor de 4 del contador solamente los registros de la fila 4 se habilitarán para poder almacenar los 16 bits de DATA_IN.

Por último, en el recuadro amarillo de la Figura 42 se tienen las mismas entradas para el ingreso de un nuevo dato mediante DATA_IN, en este caso ya no es necesario utilizar QSCK2 en estado alto porque no existe dato que se recorra a esta fila, en lugar de ello se limpian los registros.

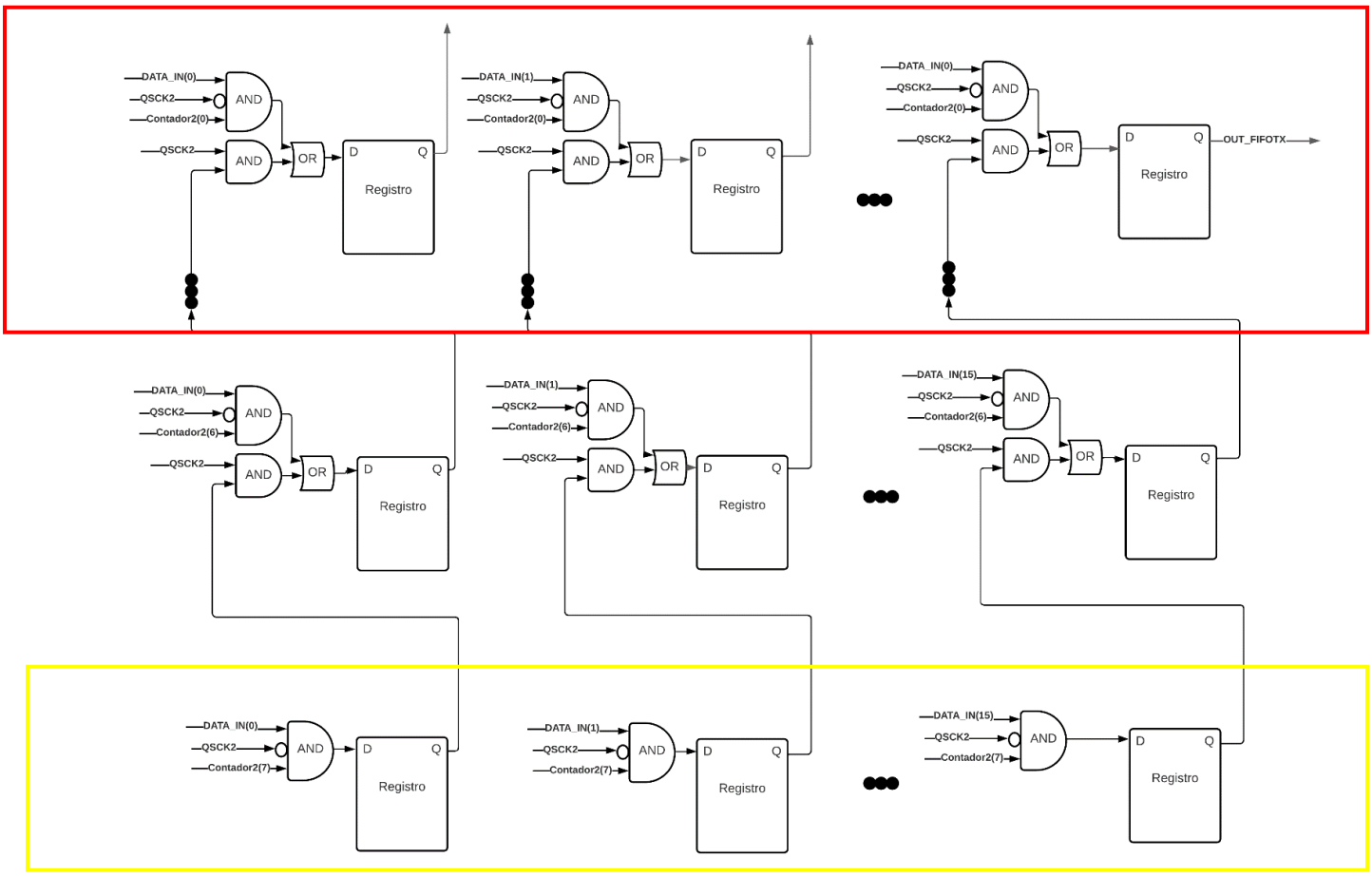


Figura 42. Memoria FIFO de transmisión

Todos los registros funcionan de manera síncrona al flanco de reloj ascendente con el reloj del sistema (CLK) donde al estar habilitados existen dos opciones de dato de entrada a cada registro. En caso de tener un dato de entrada junto con la señal QSCK2 en bajo este se guardará en la fila correspondiente marcada por el contador dos. Sin embargo, si QSCK2 se encuentra en alto el dato de la fila siete se desplazará en la fila seis, el dato de la fila seis se desplazará hacia la fila cinco y así sucesivamente hasta llegar a la fila cero. Siempre se ingresarán ceros a la fila siete cuando se extraigan datos de la memoria para limpiar los registros desocupados.

A continuación, en la Figura 43 se muestra la configuración de dos registros tomando como ejemplo el bit menos significativo ejemplificando esquemáticamente los demás registros con su respectivo dato de entrada y habilitador, es decir, como en la figura anterior, a este se le agrega el habilitador a los registros debido a que sin esto el dato no se podría retener por un intervalo de tiempo mayor a lo de un periodo del CLK.

Comenzando el análisis de la Figura 43 los recuadros rojos ya fueron explicados con anterioridad. El recuadro amarillo indica la habilitación del registro, cuando el contador2 tiene un valor de 6 y existe un dato de entrada se da dicha habilitación de toda la fila 6, si alguna de estas dos condiciones no se cumple, pero QSCK2 se encuentra en alto entonces

también el registro es habilitado. En el recuadro azul se tiene la misma configuración, el único cambio es que se necesita que el valor del contador2 sea igual a 7.

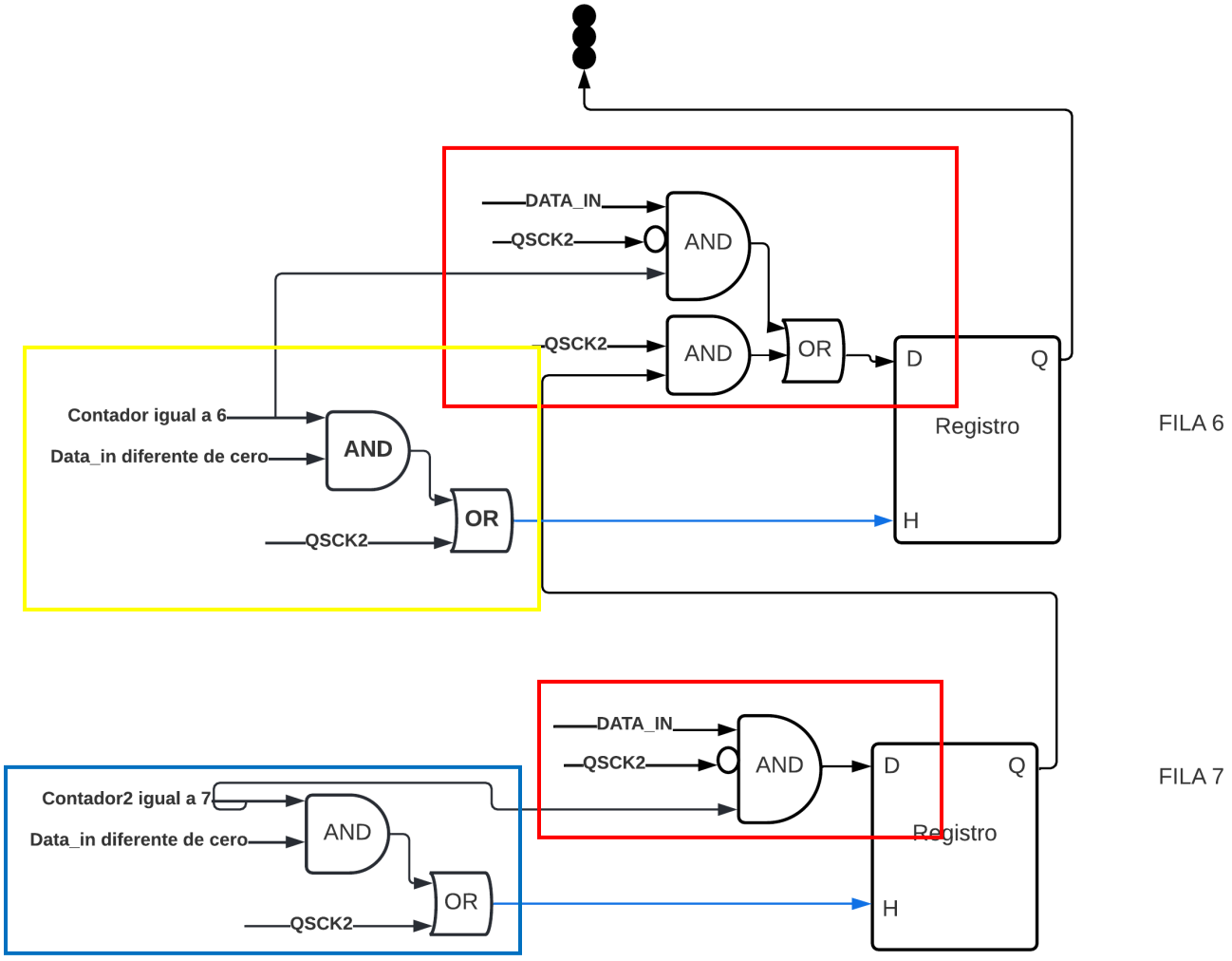


Figura 43. Registros menos significativos de la memoria FIFO de transmisión

A continuación, se presentan las simulación y análisis del comportamiento de este elemento.

En la

Figura 44 se explicará recuadro por recuadro, cada uno de ellos corresponde a una etapa diferente de la memoria. Comenzando por el recuadro rojo, DATA_IN como dato de entrada tiene un valor de 90 y al flanco de subida del CLK se guarda en la memoria FIFO en el primer espacio disponible el cual es FIFO_TX_0, en la salida Q_DATA0 es la salida de los 16 registros del dato inmediato por lo que se tiene el mismo valor.

En el recuadro amarillo el siguiente dato a ingresar es 344, se guarda el dato en el segundo espacio de la memoria el cual es FIFO_TX_1 sin modificar el primer dato. El contador 2 aumenta en 1 teniendo como resultado un valor de 2 indicando que el siguiente dato se guardara en la fila FIFO_TX_2.

Como se observa en el recuadro verde se siguen ingresando datos hasta el momento en que QSCK2 cambia a estado ALTO donde a pesar de tener un dato de entrada en DATA_IN se prioriza la extracción del ya existente en la memoria. Al mismo tiempo en que se extrae, los demás datos se recorren una fila hacia arriba. Como en este caso DATA_IN solamente es la señal de entrada se observa que el dato 1106 no es ingresado ya que se inactiva la posibilidad de guardar algún dato, por ende, este se pierde.

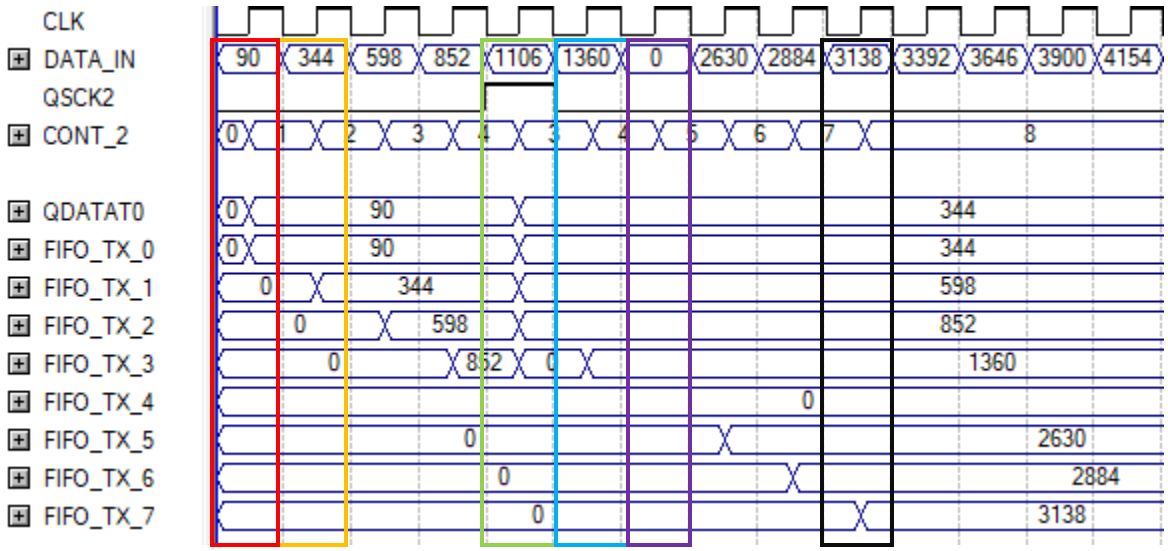


Figura 44. Memoria FIFO de transmisión - Ingreso de datos

En el recuadro azul de la

Figura 44 se ingresa un dato después de una extracción. Después, en el recuadro morado hay 0 en DATA_IN el cual también es válido para ser guardado por lo que el contador aumenta su de valor.

Por último, en el recuadro negro se ingresa un dato al último espacio disponible por lo que el contador aumenta en uno teniendo un valor final de 8. Después haber llenado la memoria, aunque existan más datos de entrada estos no ingresaran hasta que el dato inmediato sea extraído dejando un espacio libre.

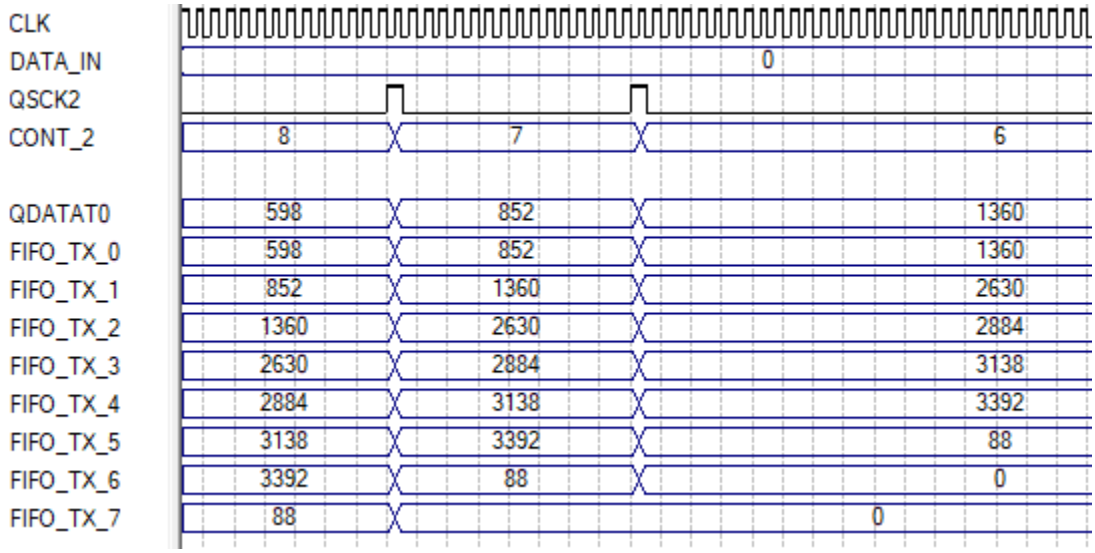


Figura 45. Memoria FIFO de transmisión - Extracción de datos.

En la Figura 45 se dejan de ingresar datos a la memoria, solamente se extraen en cada pulso de QSCK2. Se observa que en DATA_IN se tienen datos igual a 0, en este caso para representar el vaciado de la memoria se tomara como que no hay datos de entrada. Por lo tanto, al momento de recorrerse los datos a la fila siguientes el ultimo espacio anteriormente ocupado se limpia ingresando 0 al lugar correspondiente.

6.1.2 Memoria FIFO de Recepción (FIFO_RX)

La memoria FIFO de recepción tiene un funcionamiento similar a la FIFO de transmisión. La entrada CLK se refiere al ingreso del reloj del sistema, DATA_IN al ingreso del dato a la memoria, se tiene de igual manera el contador 3 para el conteo de datos dentro de la FIFO, R es la línea de *Reset*. La diferencia entre las dos memorias radica en la entrada Q_EXTRAER la cual se explica su funcionamiento en la sección 5.2.6, como su nombre lo dice está enfocada a extraer un dato en la memoria cada que el nivel de esta se encuentre en ALTO. Para el ingreso de un dato es utilizado QSCK2_RX el cual funciona con las salidas del Contador 6 y el contador 7, trabaja bajo la condición que no se esté extrayendo ninguno dato en ese instante, que haya un valor a ingresar en DATA_IN y que el contador 7 tenga un valor de 1.

Figura 46. Memoria FIFO de recepción - Entidad.

La complejidad de la memoria FIFO de recepción y el direccionamiento de datos es la misma que la de transmisión. Se utilizarán 8 arreglos de 16 registros con entrada y salida paralelos junto con la utilización del contador 3 en lugar del contador 2.

Su funcionamiento dentro del sistema comienza en el momento que se guarda un dato recibido desde otro dispositivo en los registros de corrimiento, al completarse el proceso de recepción se ingresa el dato a la memoria en la fila correspondiente y hasta que se requiera se extraerá el dato de manera paralela siendo direccionado a los registros de lectura DR para su utilización.

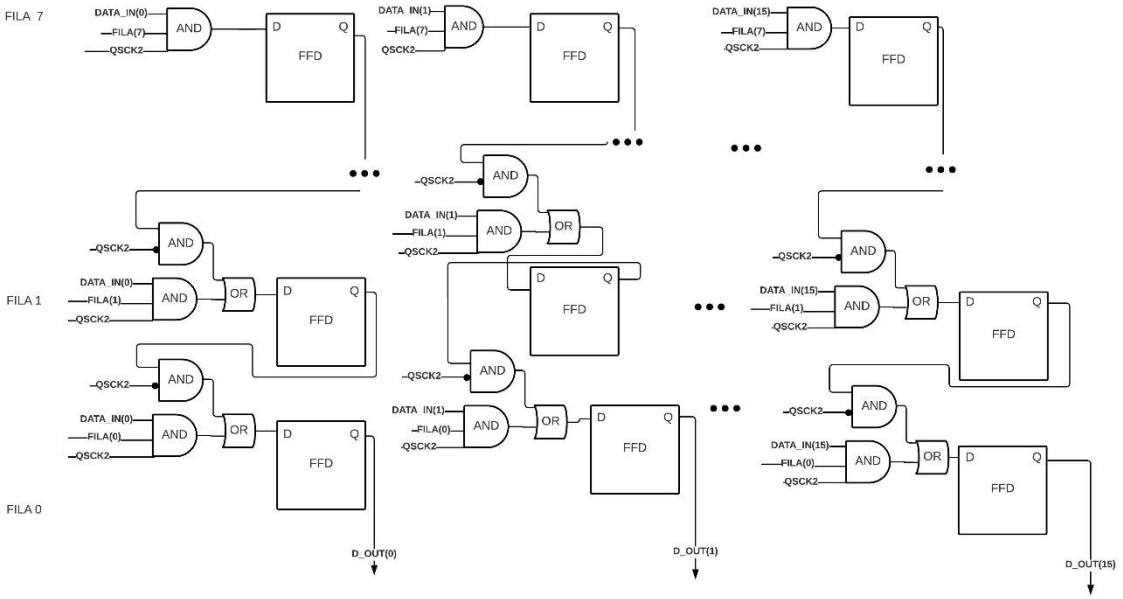


Figura 47. Memoria FIFO de recepción - Estructura interna

En la Figura 47 se tiene la misma configuración de compuertas con la única diferente que el estado de QSCK2 se utiliza al contrario, es decir, en lugar de ingresar datos con estado

bajo a la memoria, en recepción se ingresan con estado alto mientras que para extraer seria con estado bajo. Los demás elementos corresponden al contador 3 y el dato de entrada.

A continuación, se tendrá una explicación del funcionamiento de la memoria FIFO de recepción.

1. Si la memoria FIFO se encuentra vacía, el contador 3 tendrá un valor de cero.
2. Para poder ingresar un dato se tiene que completar la recepción de este. Una vez completado, la señal QSCK2_RX indicará el ingreso del dato a la memoria FIFO.
3. Al momento de ingresar un dato a la memoria el contador 3 aumentará en uno, al extraer el dato disminuirá en uno dicho contador.
4. Para poder habilitar la memoria es necesario que la señal Q_EXTRAER o QSCK2_RX se encuentren en estado ALTO con algunas otras condiciones como el que haya dato existente en la entrada.
5. La memoria FIFO puede almacenar 8 datos en total, cuando la memoria se encuentra llena el contador 3 indirectamente indica que ya no pueden ingresar más datos hasta que se extraiga el dato inmediato.
6. El dato extraído de la memoria será guardado en los registros DR de recepción durante un pulso de reloj, al siguiente pulso estos registros serán limpiados.
7. Los datos ingresados a la memoria permanecerán almacenados hasta el momento que la señal Q_EXTRAER se encuentre en estado ALTO. Solamente se ingresarán datos cuando QSK2_RX se encuentre de igual manera en estado Alto.
8. Cuando un dato es extraído de la memoria, todos los datos existentes dentro de esta son desplazados a la siguiente fila según corresponda. Es decir, el dato de la fila 6 es desplazado a la fila 5, el dato de la fila 5 es desplazado a la fila 4 y así sucesivamente. Cabe resaltar que la fila 0 contiene el siguiente dato a utilizar.
9. Se prioriza la extracción de datos ante el ingreso de algún otro.

Como se sabe, para mantener la memoria de manera estática sin que algún dato se pierda, cambie de lugar o modifique es necesario deshabilitarla. Estoy ayudará a modificar la memoria solamente al ingreso o extracción de datos en los momentos que se requiera.

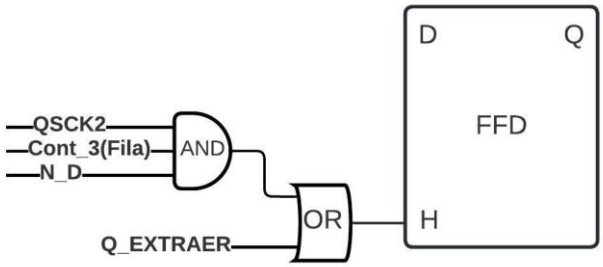


Figura 48. Registro de memoria FIFO de recepción - Habilitador

Los registros dedicados en la memoria FIFO de recepción tienen una configuración para ser habilitados de manera que puedan cambiar su valor solamente cuando sea necesario extraer un dato o cuando se necesite ingresar, en este segundo caso solamente se activara la fila correspondiente en donde el dato se ingresa mientras que los datos existentes dentro de la memoria permanecen estáticos. En la Figura 48 se indican las señales junto con su

configuración de compuertas que son necesarias para habilitar los flip-flop. El primer caso es un estado en alto de Q_EXTRAER y el segundo caso es el contador 3 indicando la fila correspondiente, un dato para ingresar a la memoria y QSCK2 de recepción en alto.

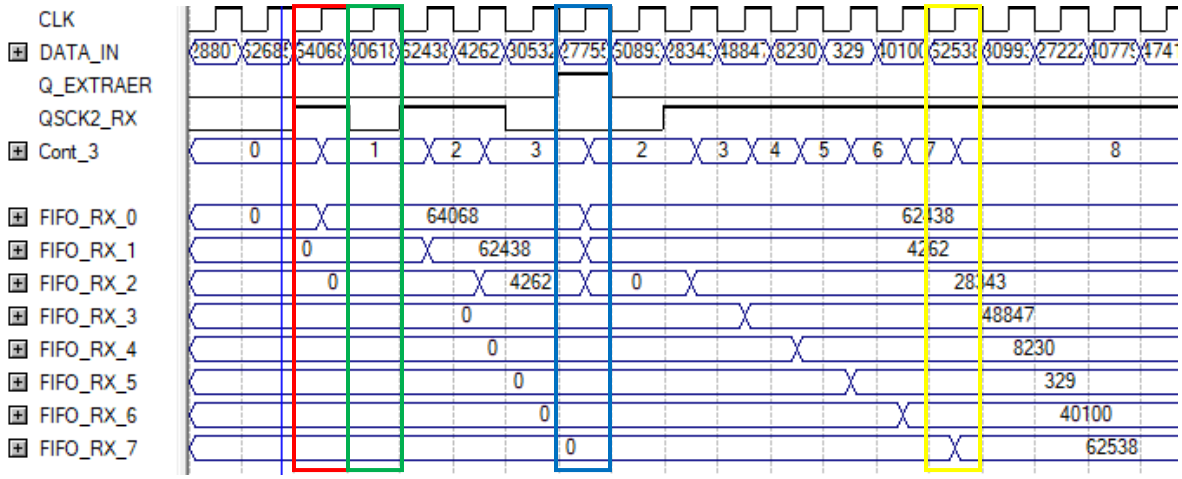


Figura 49. Memoria FIFO de recepción - Ingreso de datos

En la Figura 49 se observan varios recuadros de diferentes colores indicando diferentes casos en la memoria. En el recuadro rojo se ingresa el primer dato a la memoria, el contador 3 aumenta en 1. En el recuadro verde dado que la señal QSCK2_RX no se encuentra en alto, no se ingresa dato a la memoria. En el recuadro azul se extrae un dato dado que la señal Q_EXTRAER se encuentra en alto, los datos existentes en la memoria se recorren una fila hacia arriba ya que el dato de FIFO_RX_0 fue el extraído, también el contador 3 disminuye en 1. Por último, siguen ingresando datos hasta el punto de que la memoria no acepta más datos, en el recuadro amarillo se ingresa el último dato a la memoria mientras que el contador 3 aumenta a un valor de 8 el cual es su valor máximo.

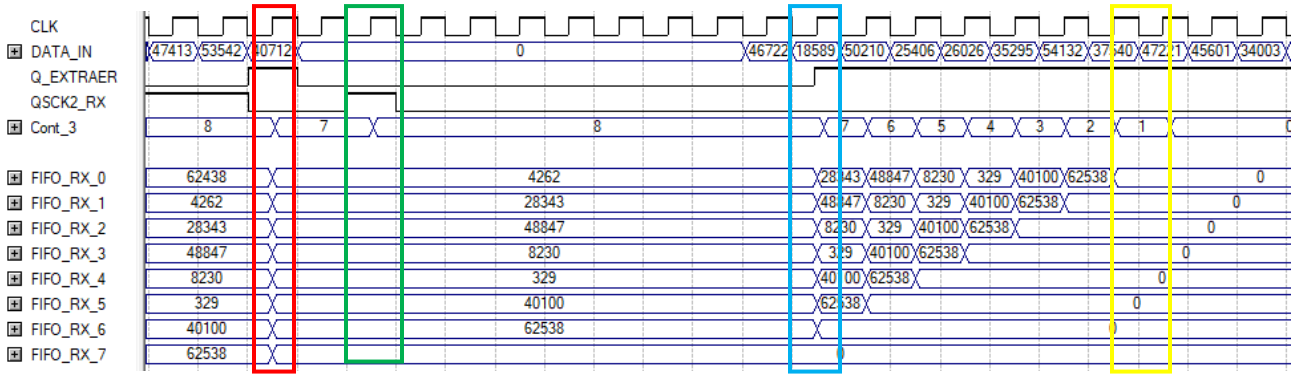


Figura 50. Memoria FIFO de recepción - Extracción de datos.

En la Figura 50 se muestra cómo se extraen los datos de la memoria. En el recuadro rojo se extrae un dato, los demás datos son recorridos a la siguiente fila y el contador 3 disminuye en 1. Se observa en el recuadro verde la falta de dato de entrada, pero con la señal QSCK2 en alto, no genera ningún cambio debido a que no hay nada que ingresar a

la memoria. En el recuadro azul se extrae de nuevo un dato hasta el momento en que Q_EXTRAER se encuentra en estado alto. Por último, en el recuadro amarillo parecería que la memoria se encuentra vacía, pero queda un dato igual a 0 por extraer. Después de extraer este último dato el contador permanece en 0 debido a que este es su valor mínimo.

6.2 Divisor de frecuencia.

En los sistemas digitales el reloj (CLK) marca la sincronía de todos los elementos del sistema. Algunas veces es necesario la utilización de un reloj de menor frecuencia para poder operar algunos elementos a una velocidad más baja, esto puede generarse mediante un divisor de frecuencia, la utilización del PLL, entre otras.

Al momento diseñar el divisor de frecuencia primero se tomó en cuenta el uso del PLL para poder tener mayor sintonización de frecuencias a utilizar por debajo de los 50 MHz, el problema de esta opción fue que el estándar SPI necesita generar un estado de reposo del reloj SCK mientras no haya transferencia de datos, esto no permitió el uso de PLL por lo que se eligió utilizar el divisor de frecuencias el cual no tendría este problema.

En este caso al elegir un divisor de frecuencia es necesario diseñar un contador para saber el momento en que cambiará de estado la señal de salida, esta será llamada SCK1. También un comparador será requerido para poder trabajar las diferentes frecuencias. Se tendrán dos salidas del divisor de frecuencia, una de ellas no tendrá pausas y será con las que operaran algunos elementos internos en modo maestro (SCK1) y la segunda salida (SCK) será dirigida al dispositivo esclavo, por ende, también tendrá un estado de reposo al no haber transmisión de datos.

Figura 51. Divisor de frecuencia - Entidad.

En este divisor de frecuencia se tiene como entrada el reloj del sistema (CLK), el Chip Select (CS), CPOL, CPHA, el reset (R) y el número de ciclos el cual servirá para dictar la frecuencia deseada tanto para SCK como para SCK1. También se tienen dos entradas extras (QVAN y QF) muy importantes ya que con estas señales se monitorea el final de la transmisión de datos para poder pausar el reloj en el momento correcto.

Funcionamiento del divisor de frecuencia.

1. El reloj CLK ingresa a un contador.
2. El contador utilizado es el CONTADOR 4 visto en la sección 5.2.4.

3. La entrada CICLOS será utilizada para indicar el comienzo de la cuenta del contador descendente. Cuando la cuenta llegue a 0, el estado de las salidas cambiará y el contador reiniciará su cuenta.
4. Las salidas del reloj tendrán el mismo valor a diferencia de la pausa de SCK cuando no haya transmisión de datos.
5. Para cada configuración de reloj (CPOL y CPHA) la pausa de SCK tendrán un momento diferente.

Para poder obtener el número de CICLOS dada la frecuencia deseada, se utilizará la siguiente ecuación.

$$\#CICLOS = \frac{Frecuencia(CLK)}{2 * Frecuencia(SCK)} - 1$$

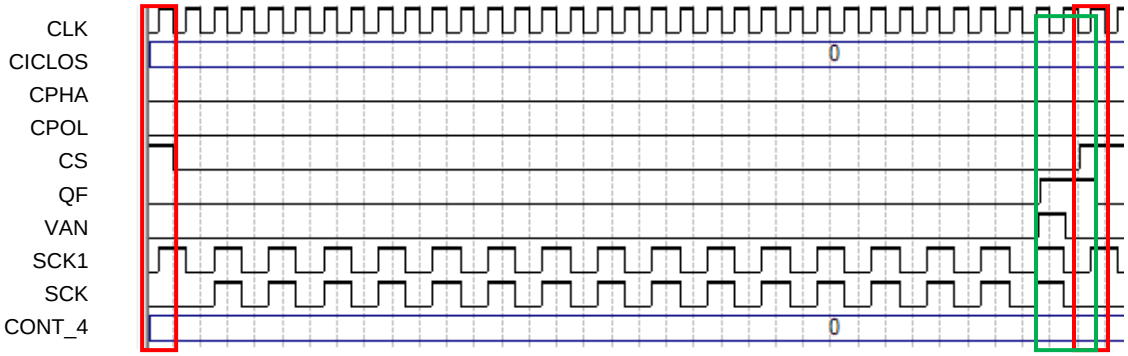


Figura 52. Divisor de frecuencias – Ciclos 0

En la Figura 52 se observa en los recuadros rojos al CS en estado alto, esto indica que el reloj de salida SCK estará deshabilitado mientras que el SCK1 continua periódicamente. En el recuadro verde se observa las señales VAN y QF cambiando a estado alto al momento en que el ultimo flanco de subida del reloj, estas dos señales se activan al momento en que el contador 1 se encuentra en el último número antes de regresar a un valor de 0. Es decir, si se configura para transmitir 16 bits, al tener un valor de 15 en el contador 1 estas dos banderas se activan, si se configura para 7 bits las banderas se activan con un valor de 6. VAN y QF ayudan a parar el reloj para que pueda tener un comportamiento diferente según corresponda con los valores de CPOL y CPHA.

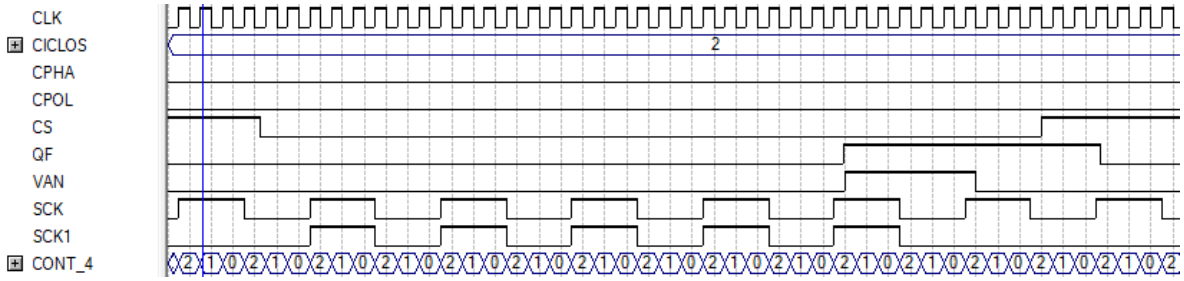


Figura 53. Divisor de frecuencia - Ciclos 2

En la Figura 53 se observa el valor de CICLOS igual a 2, este valor es cargado al contador 4 cada vez que llega a cero ya que su cuenta es descendente. En el caso de que sea 0 como en la Figura 52 el contador permanecerá estático, mientras que si tiene algún otro valor este será cargado al contador una vez llegue a 0.

Los pulsos tanto de SCK como de SCK1 son más largos debido a que se espera que transcurran 3 pulsos de reloj CLK que deben de pasar para un cambio en la salida del divisor de frecuencia. En este caso la frecuencia resultante es de 16 MHz aproximadamente.

6.3 Registros de desplazamiento

Los registros de desplazamiento están conformados por un conjunto de flip-flop, normalmente del tipo D. Estos registros son utilizados para almacenar datos o transferirlos dentro del sistema mediante un desplazamiento. No es necesario una declaración de estados para un desplazamiento de datos, con la configuración correspondiente de compuertas a la entrada de cada registro puede ser funcional dependiendo del tipo de desplazamiento.

Existen cuatro tipos de registros de desplazamiento de los cuales solamente tres de ellos son útiles para el proyecto:

- Paralelo – Paralelo.
- Paralelo – Serie.
- Serie – Paralelo.

Los registros de desplazamiento pueden tener diferentes longitudes. En este caso se utilizarán 16 flip-flop para poder almacenar 16 bits los cuales son necesarios tanto para los datos de entrada como los datos de salida.

Los registros de desplazamiento de entrada y salida son registros con una ubicación intermedia entre las memorias FIFO tanto de recepción como de transmisión y las líneas MISO y MOSI, respectivamente. Estos registros son utilizados para guardar un dato y enviarlo a memoria al completarse el proceso o enviar un dato bit a bit por completo.

6.3.1 Transmisión (Paralelo -Serie)

En la transmisión se utilizan registros de desplazamiento Paralelo - Serie. Su funcionamiento básico es ingresar el dato de manera paralela en un pulso de reloj para después desplazar el dato a lo largo del arreglo de registros. Es necesario un multiplexor a la entrada de cada registro debido a que se debe seleccionar el dato a ingresar según convenga.

Los registros de desplazamiento utilizados en la interfaz tienen una configuración inusual para el dato de entrada en paralelo debido a su funcionamiento. Un aspecto que dificulta la implementación es la falta de pulso de reloj SCK en modo esclavo para poder cargar el dato, como se muestra en la Figura 9 el dato a transmitir tiene que estar posicionado a la salida antes del primer pulso de reloj propio de la interfaz. En este caso, para el modo maestro no existe ningún problema ya que puede trabajarse internamente con el SCK1 y el SCK ya que los dos operan a la misma frecuencia. El problema existe al momento de utilizar la interfaz en modo esclavo, la frecuencia del reloj SCK interno no siempre será la misma que la del reloj del maestro, esto genera el problema en el primer bit del dato a transmitir en la línea MISO.

La manera en que se resolvió este problema se explica a continuación.

Comenzando con el proceso, se tiene un dato en la memoria FIFO de transmisión. Posteriormente el bit de mayor significancia se guarda en un registro que funciona con el CLK, este dato se posiciona en la línea de transmisión (MISO) mediante un multiplexor 2 a 1 hasta el momento en que el contador 1 tenga un valor diferente de 0. Una vez transmitido el primer bit, en la línea MISO es seleccionada la salida de los registros de desplazamiento paralelo-serie donde se encontrarán los otros 15 bits.

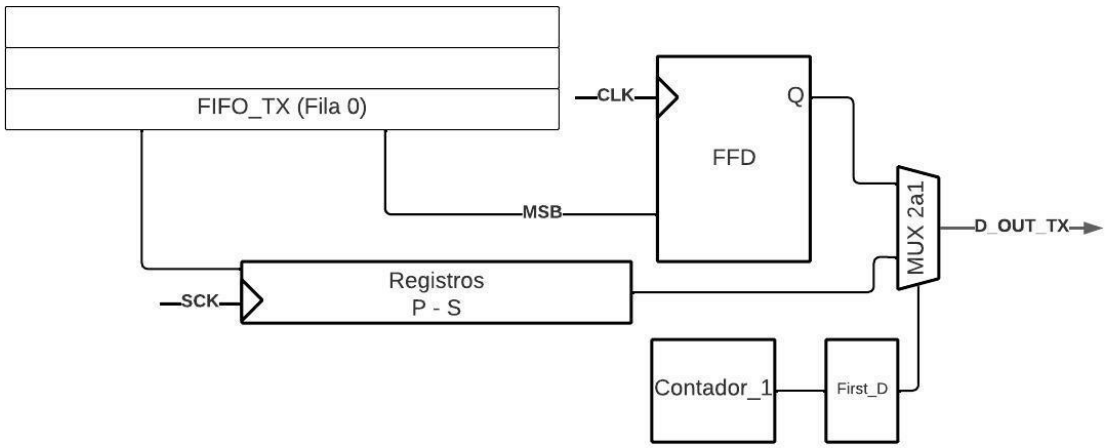


Figura 54. Transmisión - Dato de salida.

En la Figura 54 se muestra esquemáticamente lo anteriormente explicado. El primer bit para transmitir es el que tiene mayor significancia, se almacena en el FFD y mediante el multiplexor se le da acceso para poder llegar a la línea de salida D_OUT_TX. Una vez transmitido, el multiplexor selecciona la salida de los registros de desplazamiento los cuales tienen los demás bits.

El funcionamiento del First_D utiliza la cuenta del contador 1, solamente indica cuando el valor es diferente cero o termina la transmisión de datos. Se tiene que utilizar el mismo reloj utilizado en los registros de desplazamiento para poder tener sincronía.

Debido al problema antes mencionado para el modo esclavo, esto genero la utilización de esta configuración para el modo maestro dado que se priorizo utilizar la mínima cantidad de recursos en cuanto a registros.

6.3.2 Recepción (Serie - Paralelo)

En la recepción de datos se utilizan registros de desplazamiento Serie – Paralelo, es decir, los datos ingresan de manera serial y se extraen paralelamente. El ingreso de datos es mediante la línea MISO, con cada pulso de reloj SCK se ingresa un bit hasta llegar a completar la recepción de un dato.

No es necesaria la utilización de un multiplexor para el primer bit ni para ningún otro y no necesita de tantas complicaciones con en el caso de transmisión.

En la Figura 55 se muestra la configuración de los registros de desplazamiento utilizados para la recepción de datos. Su funcionamiento comienza una vez que el primer pulso de

reloj SCK, ya sea esclavo o maestro, llegue a los registros. Con el primer pulso de reloj se guardará el primer bit, con el segundo pulso será guardo el segundo bit mientras que el primero será desplazado al segundo registro, este proceso se repetirá hasta completar la recepción de datos.

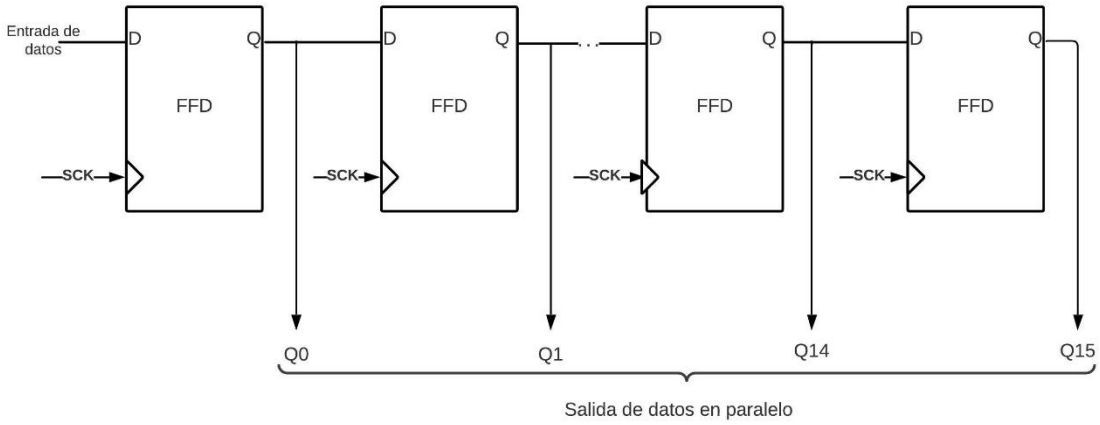


Figura 55. Recepción - Registros Paralelo-Serie

Al momento de tener un dato completo en los registros de desplazamiento el reloj SCK pausará su funcionamiento debido a que el intercambio de datos entre esclavo y maestro termino. El siguiente paso es el almacenamiento en la memoria FIFO de recepción en la fila correspondiente. Se mostrará un ejemplo como se puede ver a continuación.

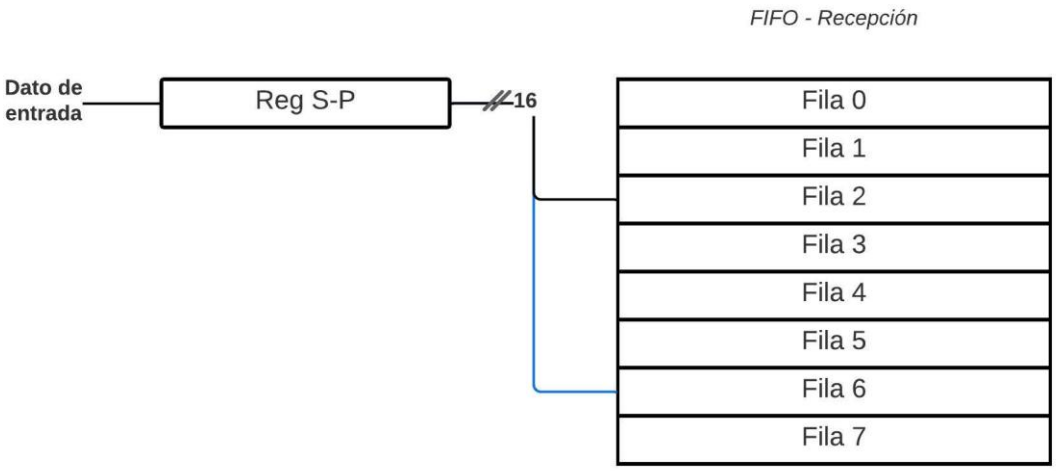


Figura 56. Recepción - Desplazamiento de datos a FIFO

En la Figura 56 se muestra el dato de entrada a los registros de desplazamiento, la salida de estos registros debido a que es de manera paralela se tienen 16 bits en total los cuales serán almacenados en algún lugar de la memoria FIFO. Se muestran dos ejemplos para este proceso.

En el primer ejemplo se tienen 2 datos en la memoria los cuales ocupan la Fila 0 y 1, el siguiente dato a almacenar será ingresado a la Fila 2. El segundo ejemplo (Línea azul) indica un total de 6 datos almacenados, es decir, de la Fila 0 hasta las 5 existe un dato por lo que el espacio inmediato disponible es la Fila 6.

Este proceso es fácil de entender debido a que el funcionamiento de los registros de desplazamiento es básico. Estos registros funcionan con el reloj SCK lo cual no hace posible limpiarlos una vez extraído el dato porque no existe otro pulso después de completarse la recepción y el siguiente pulso es para poder ingresar el MSB del siguiente dato. Este proceso desplaza el dato anterior para colocar al nuevo lo cual permite una sustitución en lugar de una limpieza en los registros.

6.3.3 Registros DR

Los registros DR son registros de desplazamiento con entrada y salida Paralelos como se puede observar en la Figura 57. Principalmente se utilizan poder escribir o leer algún dato de la memoria FIFO, ya se de transmisión o recepción, respectivamente.

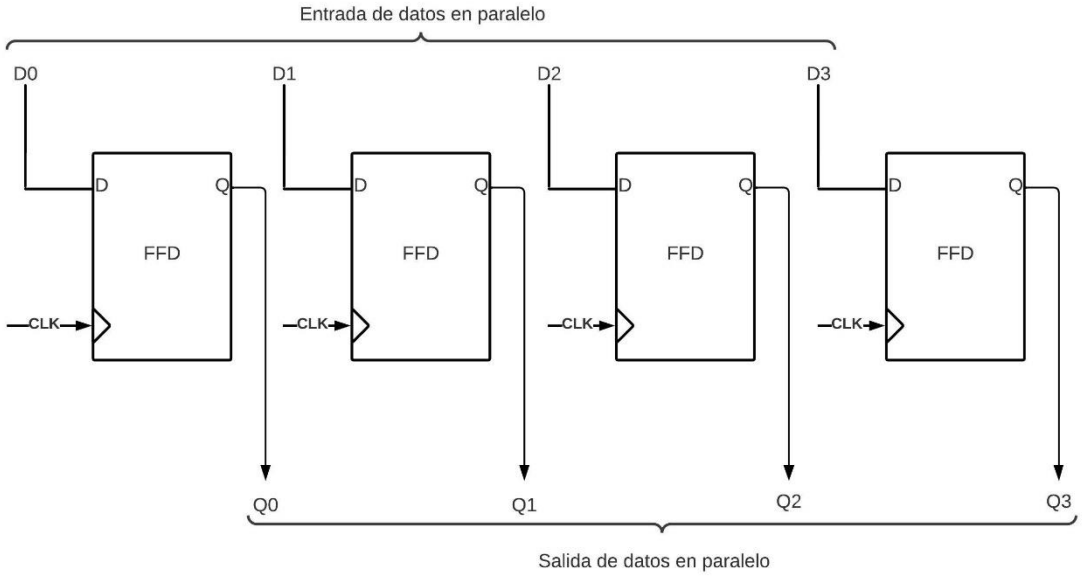


Figura 57. Registros DR - Registros de desplazamiento P-P

Como recordaremos, en la Figura 15 se observa un bloque con el nombre DR entre los datos de entrada y salida y las memorias FIFO. Se tiene un total de 32 registros, 16 para transmisión y los 16 restantes para recepción.

6.3.3.1 Registro DR - escritura

Para poder almacenar un dato en necesario utilizar los registros de escritura. La mayoría de los casos se podrá ingresar inmediatamente el dato a la memoria FIFO, en los casos donde se tendrá que esperar al menos un pulso de reloj extra del CLK es cuando se extrae

un dato de la memoria, en este caso no se puede ingresar dato por lo que es guardado para no perderlo hasta terminar este proceso.

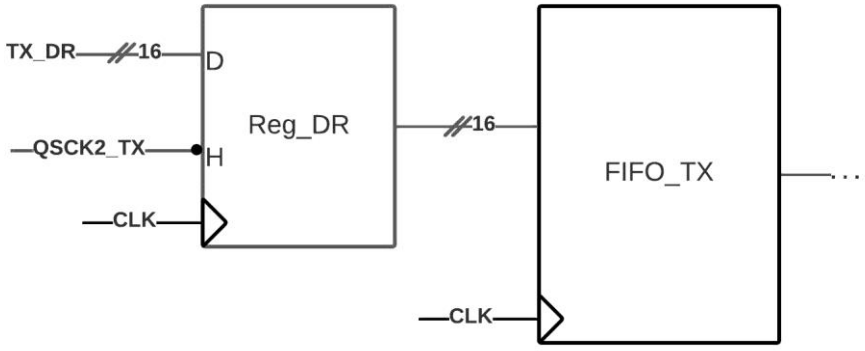


Figura 58. Transmisión - Registros DR de escritura

Los registros de escritura se encuentran entre la entrada del dato a transmitir el cual tiene una longitud de 16 bits, y la memoria FIFO de transmisión. En cada pulso de reloj puede almacenarse un dato diferente en Reg_DR. Si existe ya un dato en Reg_DR al siguiente pulso de reloj se guarda en la memoria en su fila correspondiente.

En el caso de extraer un dato de la memoria, como se vio en la sección 5.2.2 , la señal QSCK2 debe de encontrarse en estado alto para deshabilitar Reg_DR hasta que termine la extracción. En cada ciclo de reloj del CLK se puede ingresar 1 dato a los registros DR, estos podrán ser almacenados con un límite de 8 por lo que los datos se perderán en caso de seguir ingresándolos a la memoria y esta se encuentre llena.

6.3.3.2 Registros DR - lectura

El funcionamiento de estos registros DR de lectura es el mismo que de los de escritura, la diferencia radica en el momento en que se pueden ingresar datos a estos. Aunque exista un dato para ingresar este no tendrá acceso hasta el momento en que la señal EXT del contador 6 sea activada. Esta es la diferencia importante aparte del direccionamiento de datos el cual es inverso, es decir, en lugar de ingresar datos a la memoria FIFO estos serán extraídos de ella.

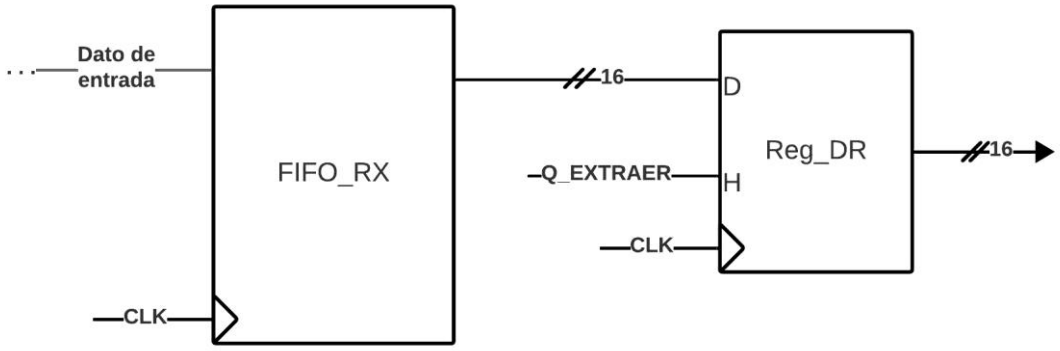


Figura 59. Recepción - Registros DR de lectura

De acuerdo con la Figura 59 para poder ejemplificar su funcionamiento se suponen 3 datos almacenados en la memoria FIFO. El dato siguiente para extraer de la memoria está dirigido a Reg_DR, sin embargo, por más pulsos de reloj que ingresen no será posible leerlo hasta el momento en que la señal Q_EXTRAER del contador 6 se active. Al tener un estado alto en la señal Q_EXTRAER el dato inmediato de la memoria FIFO es desplazado a Reg_DR, este permanece solamente un pulso de reloj y después estos registros regresan a un valor de 0.

6.4 Numero de Bits en cada dato (DT)

Un requerimiento importante en la interfaz es el poder configurar el número de bits en cada dato que se transmita. Dentro de la descripción en VHDL se nombra DT a este parámetro el cual tiene una longitud de 4 bits, es posible asignarle un valor desde 0 hasta 15 el cual generara que los bits del dato a transmitir sean entre un rango de 4 hasta 16. En la Tabla 1 se observa el valor que debe de tener DT para configurar el numero de bits del dato.

Tabla 1. Valores de DT

DT	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Bits	**16**	**16**	**16**	**16**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**

Para poder lograr esta configuración se tienen 16 multiplexores 16 a 1 tanto para transmisión como para recepción. En ambos casos, al asignarle un valor de 0 – 3 a DT cada dato tendrá un total de 16 bits por motivos de diseño, a partir de un valor de 4 para DT la longitud del dato coincidir con dicho valor.

El proceso de almacenado para la transmisión comienza al momento en que se escribe en los registros DR y posteriormente se guardan en la FIFO_TX, los multiplexores encargados de realizar la funciona de DT se encuentran entre estos dos componentes. De manera predeterminada la longitud del dato es igual a 16 bits, al momento de almacenarse en la FIFO_TX no se generan cambios. El caso de DT = 12, al ingresar a DR un dato igual a 0x34A2 este será almacenado en la memoria FIFO_TX como 0x4A20 truncando el valor hasta los 12 bits y almacenándolo de izquierda a derecha comenzando con el MSB. En el caso de DT = 8 el mismo dato sería almacenado como 0xA200.

La configuración de los multiplexores para realizar la función de DT en el caso de recepción cambia al momento de almacenar el dato. Su conexión dentro de la interfaz se ubica entre

los registros encargados de capturar el dato recibido mediante la entrada MISO y la memoria FIFO_RX. De igual manera que en transmisión, para un valor de DT en donde el dato tenga una longitud de 16 bits este no sufre cambios al momento de ser almacenado en la memoria. Sin embargo, en cualquier otro caso solamente se trunca el valor hasta los bits indicados y se almacena de forma normal. Por ejemplo, utilizando el mismo dato que en transmisión (0x34A2) para DT = 12 seria almacenado como 0x04A2, para DT = 8 seria almacenado como 0x00A2 y así sucesivamente en todos los casos.

De manera más explícita se muestra en la Tabla 2 como se almacenaría un dato en la memoria con valor de 0xFFFF tanto para transmisión como recepción en cada uno de los casos de DT.

Tabla 2. Datos almacenados en las memorias FIFO según DT.

DT	TX	RX
0	0XFFFF	0xFFFF
1	0XFFFF	0XFFFF
2	0XFFFF	0XFFFF
3	0XFFFF	0XFFFF
4	0XF000	0X000F
5	0XF800	0X001F
6	0XFC00	0X003F
7	0XFE00	0X007F
8	0XFF00	0X00FF
9	0XFF80	0X01FF
10	0XFFC0	0X03FF
11	0XFFE0	0X07F
12	0XFFF0	0X0FFF
13	0XFFF8	0X1FFF
14	0XFFFC	0X3FFF
15	0XFFFE	0X7FFF

Cabe resaltar que estos multiplexores se encuentran entre los registros DR y la memoria FIFO en caso de trasmisión y entre los registros de desplazamiento y la memoria FIFO para el caso de recepción. En otras palabras, se ubican antes del ingreso de la memoria FIFO.

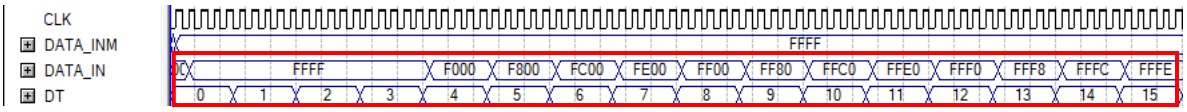


Figura 60. DT - Simulación Transmisión

DATA_INM es el valor que de salida de los registros DR de escritura y DATA_IN es el valor a ingresar en la memoria, en este caso se tiene para la simulación un dato igual a 0xFFFF para poder ver el punto al que se quiere describir. En el recuadro rojo de la Figura 60 los valores de DT de 0, 1, 2 y 3 permiten tener una configuración de 16 bits lo cual permite el desplazamiento del dato sin hacerle ningún cambio más que el guardarlo desde los registros DR a la memoria FIFO. En los demás valores de DT el valor de DATA_IN cambia y tiene el comportamiento deseado, comparando los valores de esta simulación con los de la Tabla 2 todos son correctos.

En el caso de recepción todas las entradas tendrán el mismo valor, como se puede apreciar en el ejemplo de la Figura 61. La diferencia con el caso de transmisión radica en que al momento de guardar los datos en la memoria se guardan de manera trunca

En este caso QD_IN es el dato recibido externamente y DATA_IN es el dato por ingresar a la memoria FIFO_RX

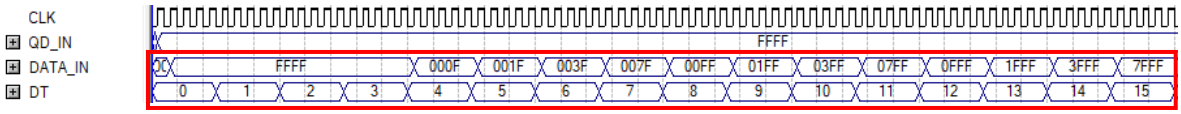


Figura 61. DT - Simulación de recepción

En la Figura 61 en el recuadro rojo se observa el comportamiento esperado, comparando estos resultados con los de la Tabla 2 todos coinciden dando lugar a un buen funcionamiento. También se observa que el dato es truncado, es decir, solamente se utilizan los bits indicados por DT como es el ejemplo de DT = 12, en DATA_IN no se almacenan los 4 bits más significativos de 0xFFFF.

Algo que agregar en este apartado es la importancia que tiene esta configuración para poder cambiar el número de bits en cada dato, permite una configuración desde 4 hasta 16 bits. Este parámetro aparte de influir en la forma en que se guardan los datos también afecta al CS ya que esta señal debe de mantenerse en estado bajo, y el número de pulsos del SCK cambia según se necesite con respecto a el valor de DT. Por ejemplo, para un valor de DT igual a 10, el CS deberá mantenerse en bajo durante 11 pulso de reloj SCK y posteriormente pasar a estado alto. En pocas palabras, DT también influye en el tiempo que deberá estar en estado bajo el CS y los pulsos de SCK.

6.5 Chip Select (CS)

El CS es una línea primordial para el funcionamiento interno y la transmisión de datos. Indica el momento en que se inicia y detiene el intercambio de datos, también es el encargado de seleccionar el esclavo con el cual se requiere comunicación.

La línea de selección tiene 2 posibles estados, el primero es ‘0’ el cual indica una transferencia de datos y la segunda es ‘1’ la cual la interfaz permanece en estado de reposo donde ningún dato es transferido.

En modo maestro el módulo SPI puede tener varias líneas de selección para intercambiar datos con algún esclavo con el que se desee comunicar con tan solo activar su línea

correspondiente. Sin embargo, cuando se encuentra en modo esclavo existe una línea de selección única de entrada.

El CS es generado internamente en modo maestro mientras que en modo esclavo se recibe como una entrada, debido a esto en modo esclavo no existe un diseño para generar este componente ya que se utiliza como una señal externa. Para generar el comportamiento deseado del CS es necesario utilizar la sincronía del reloj SCK1, por ende, también se necesita el divisor de frecuencia. También es necesario agregarle una modificación para que pueda funcionar según corresponda a los valores que tengan las entradas CPOL y CHPA ya que dependiendo del valor que tengan en conjunto el momento en que el estado del CS cambie es diferente para cada caso.

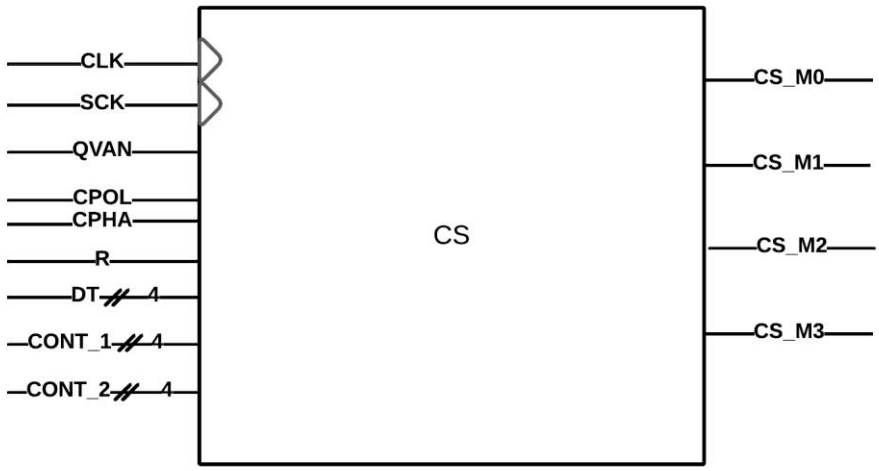


Figura 62. Chip Select - Entity.

En la entidad del CS se observan entradas ya analizadas como son el Contador 1 y 2 o QVAN vista en el apartado 6.2, se incluyen tanto el CLK como el SCK y otros parámetros necesarios. Todas son muy importantes para poder generar el comportamiento necesario del CS aunque internamente algunas sean utilizadas para condiciones de inicio como se analizara más adelante.

Recordando, el estado en reposo del CS es en alto, es decir, cuando no se transfiere ningún dato el CS permanece en ‘1’. Este estado se mantiene gracias a el contador 1, contador 2 y a la no existencia de dato en los registros DR. Es necesario que los dos contadores de encuentren en cero y no haya dato de entrada para que se mantenga un estado alto ya que esto indica que no hay nada que transmitir y la memoria está vacía.

Para poder cambiar a estado bajo es necesario que alguna de las señales anteriores tenga un cambio, que exista dato en la entrada o que la memoria tenga algún dato guardado (contador 2 diferente de cero). En el caso del contador 1 no es necesario que cambie de valor ya que para que este funcione se necesita un estado bajo del CS.

Al terminar la transferencia de datos el CS se cambiará a estado alto nuevamente, esto se dará a partir de un estado alto en la bandera QVAN, después esta bandera pasa a estado bajo una vez que el CS se encuentra en modo reposo, esto ayuda a mantener el CS en alto

hasta la existencia de otro dato. En caso de que exista un dato en la memoria se repite el mismo proceso de transmisión.

En la Figura 63 se observa de manera esquemática lo anteriormente explicado. El CS solamente permanece en estado alto cuando QVAN es igual a '1' o cuando no existe ningún dato en la memoria para transferir.

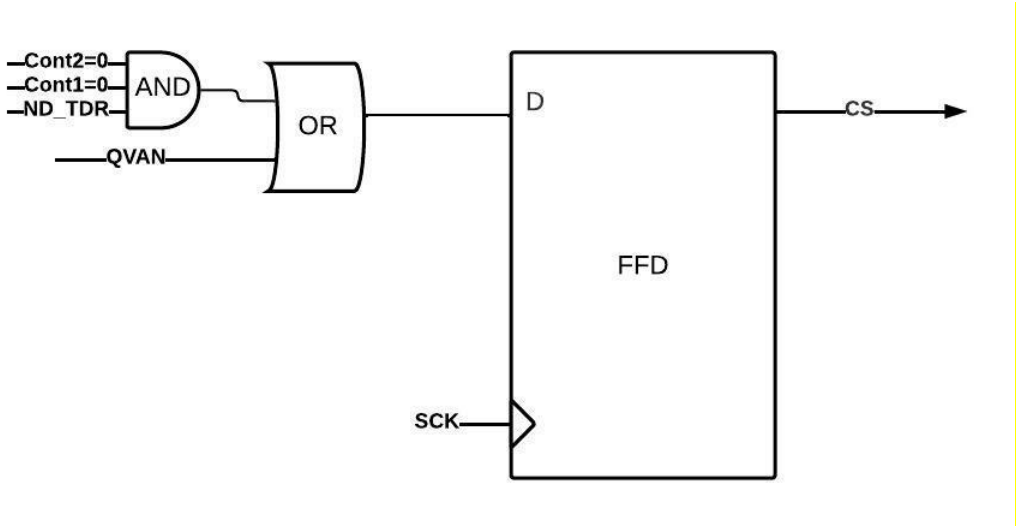


Figura 63. CS - Generación de estado alto y bajo.

La interfaz trabaja como multiesclavo por lo que se tendrán 4 líneas de selección a la salida, es decir, 4 CS. Para poder seleccionar el CS requerido se utilizó un demultiplexor 1 a 4 y se le agrego de igual manera una condición de inicio el cual se observa en las líneas rojas de la Figura 64, es decir, el contador 1 y 2 deberán de tener un valor de cero lo cual generará un estado alto para cada uno de los CS desde el momento en que inicia el programa. El seleccionador S_CS podrá controlarse mediante registros de control.

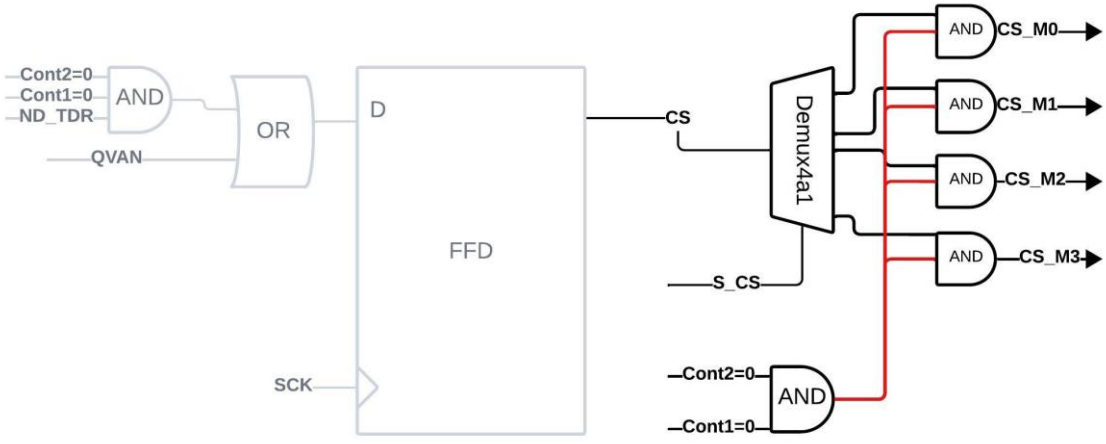


Figura 64. CS - Multiesclavo.

Su funcionamiento genera buenos resultados como se muestra a continuación.

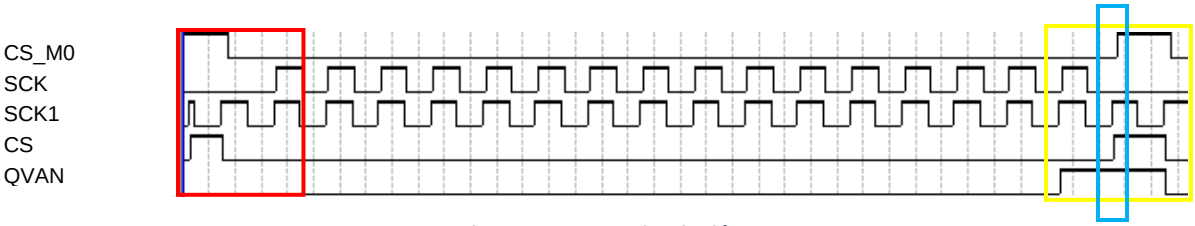


Figura 65. CS - Simulación

En la Figura 65 en el recuadro rojo se observa la inicialización del programa en donde el CS_M0 comienza en estado alto, el reloj SCK1 se estabiliza después de un pequeño instante y el CS comienza en '0', pero inmediatamente cambia a '1' ya que esta señal es síncrona al reloj CLK. En el recuadro amarillo se observa tanto al CS_M0 como al CS con el mismo comportamiento, esto nos indica que todo funciona como se espera. En el caso de la bandera QVAN se observa que cambia a un estado alto después del último flanco de subida del SCK indicando el envío del último bit del dato a transmitir. Por último, en el recuadro azul se tiene a QVAN en estado alto lo que permite que al siguiente pulso del SCK1 el CS cambie a estado alto indicando el término de su actividad. Así es como se genera el comportamiento del CS. Hasta este punto el CS tiene un funcionamiento correcto, pero se detectó una falla cuando se transmite el último dato almacenado en la memoria FIFO_TX antes de quedar vacía dicha memoria. Como se observa en la Figura *66* en el recuadro rojo, la señal CS_M0 tiene un adelanto con respecto al CS lo cual no debe de pasar, esto genera que la señal SCK evite tener un estado de reposo en cero y los siguientes envíos de datos se den con errores.

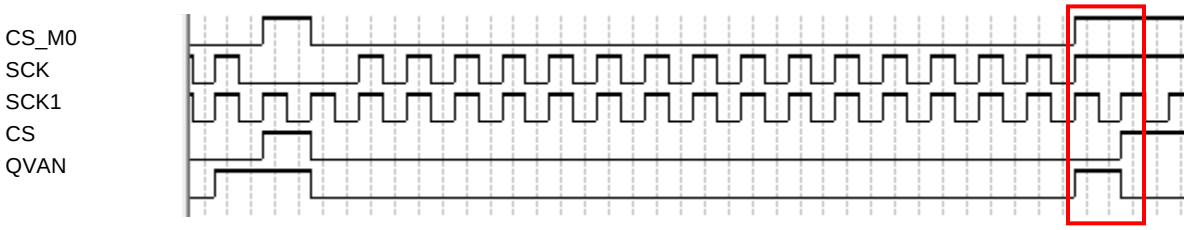


Figura 66. CS - Simulación con memoria FIFO vacía

Antes de presentar la manera en cómo se resolvió esta falla es necesario conocer con más detalle del por qué se genera. El CS_M0 necesita tener un estado en alto siempre que no exista una transmisión sino esto puede generar un error, cuando se inicializa el sistema o se reinicia la interfaz el CS tarda un pequeño instante en operar de manera correcta como se observa en la Figura 67 en el recuadro verde y la señal SCK1 genera un pequeño pulso debido a esta falla. Entonces, por mantener este estado en alto desde el comienzo en CS_M0 se da la falla, esto aplica para las demás salidas CS_M1, CS_M2 Y CS_M3.

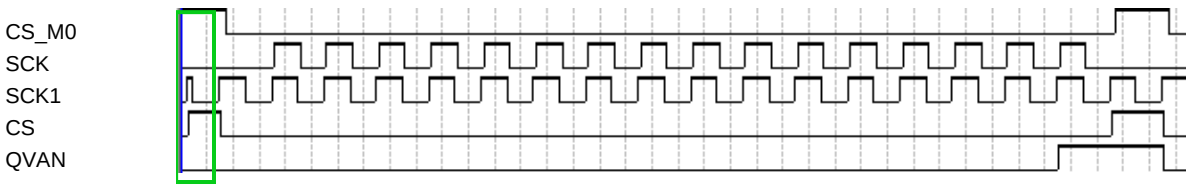


Figura 67. CS - Adelanto

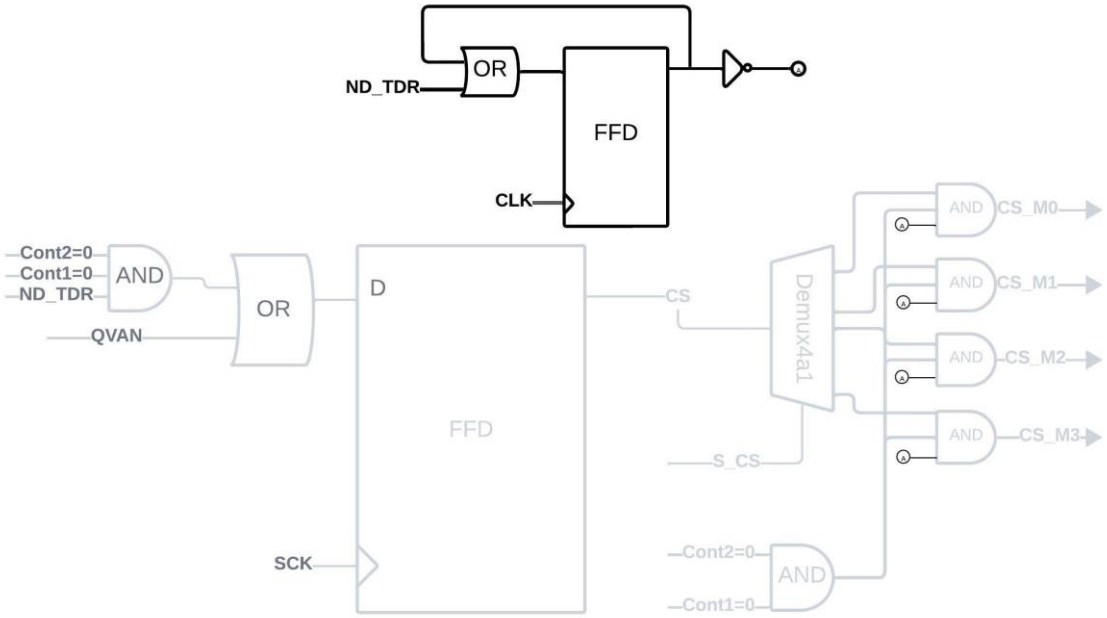


Figura 68. CS - CS_INICIO

La manera de resolverlo fue agregando un flip-flop D el cual se puede observar en la Figura 68, este tiene la función de generar una bandera que indica un '1' cuando se comienza la primera transmisión y permanece en este estado hasta que se reinicia la interfaz. Esto permite dos cosas, la primera es que al inicializar la interfaz se tenga un estado alto del CS_M0 sin importar la sincronía del CS con el CLK y la segunda es que la falla antes detectada se resuelva.

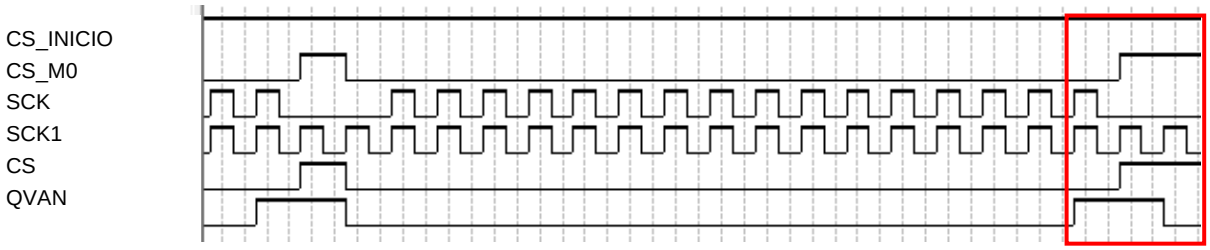


Figura 69. CS - Simulación sin el problema del CS

En la Figura 69 se muestra el problema resuelto, gracias a este registro que se agregó se obtiene el comportamiento deseado como se observa en el recuadro rojo, esto aplica desde el CS_M0 hasta el CS_M3.

En la siguiente sección se mostrará el funcionamiento del CS con diferentes configuraciones de CPOL y CPHA, ya que hasta este punto se tomó como predeterminado un valor de CPOL = '0' y CPHA = '0'.

6.6 Polaridad y Fase del reloj (CPOL y CPHA)

Dos de los parámetros más importantes para el módulo SPI son CPOL y CPHA encargados de cambiar la configuración del reloj. El comportamiento que debe de tener el reloj SCK depende de estos dos parámetros, el reloj puede mantener su estado de reposo en alto o

en bajo y pueden capturarse los datos con flanco de subida o bajada. Estas dos características son modificables a partir de los dos parámetros antes mencionados.

En la sección 3.5.2 se observan los cuatro casos posibles de la configuración del reloj SCK. Para poder ver la importancia de estos dos parámetros se mostrará de primera instancia el funcionamiento del reloj y posteriormente los registros relacionados a esta.

6.6.1 Funcionamiento del reloj SCK

Para practicidad de las siguientes 4 simulaciones se utilizó una configuración de DT igual a 4, es decir, 4 bits en cada dato a transmitir o 4 pulsos de reloj SCK, aunque para cualquier valor de DT las bases del funcionamiento es el mismo. Esto solamente para poder ejemplificar de una mejor manera los flancos en que debe de cambiar el CS y capturar el cada bit del dato.

- CPOL = ’0’ y CPHA = ‘0’

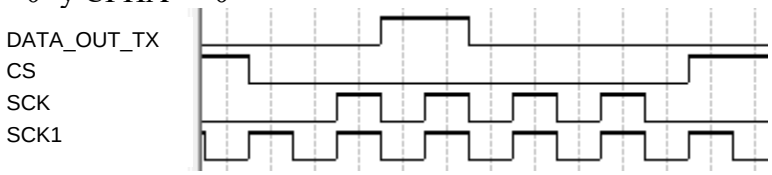


Figura 70. CPOL y CPHA - Simulación 00

En la Figura 70 se observa la utilización del reloj SCK1 en donde internamente se encuentra conectado a los flip-flop encargados de generar el comportamiento del CS. Es utilizado el flanco de subida para el cambio de estado del CS, tanto para bajo como para alto. También se observa que el dato de salida cambia de bit con flanco de bajada ya que este tiene que ser centrado y ser capturado con flanco de subida. Cabe resaltar que el dato enviado es 0x04 en las 4 diferentes configuraciones de reloj.

- CPOL = ’0’ y CPHA = ‘1’

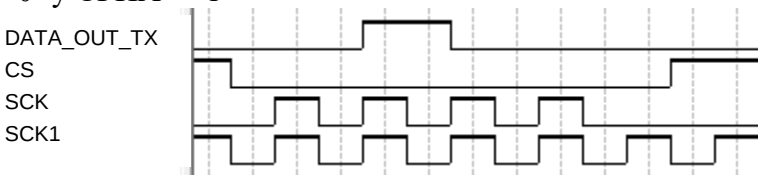


Figura 71. CPOL y CPHA - Simulación 01

El reloj SCK se mantiene un estado de reposo en ‘0’ como se observa en la Figura 71. Ahora el flanco de subida no es requerido para capturar algún bit que se transmita o se reciba sino el flanco de bajada es el utilizado para dicha captura. El CS cambia de nivel con flanco de bajada de SCK1.

- CPOL = ’1’ y CPHA = ‘0’

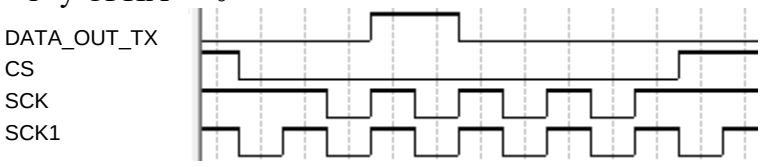


Figura 72. CPOL y CPHA - Simulación 10

De igual manera que al tener un valor de CPOL = ‘0’ y CPHA = ‘1’, la captura de bits y el cambio de estado del CS en esta configuración se da con flanco de bajada. El estado de reposo del reloj es en nivel alto.

- CPOL = ’1’ y CPHA = ‘1’

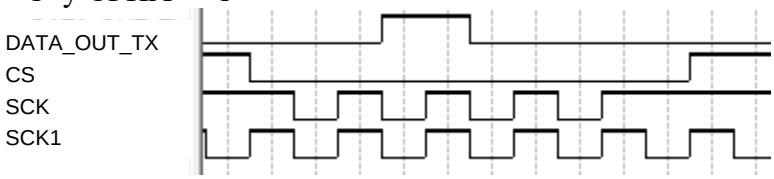


Figura 73. CPOL y CPHA - Simulación 11

Por último, en esta configuración del reloj el SCK se encuentra en estado de reposo con un nivel alto y la captura de bits junto con el cambio de estado del CS se da con flanco de subida.

Ahora, para poder generar el comportamiento del reloj dependiendo del valor de CPOL y CPHA se utilizó un multiplexor el cual se encarga de habilitar el flip-flop encargado de la señal SCK y en cada entrada del multiplexor es necesaria una configuración de compuertas para cada uno de los 4 casos.

En la Figura 74, se observa en el recuadro rojo la configuración del multiplexor diseñado para generar el estado de reposo del SCK según corresponda a los valores de CPOL y CPHA.

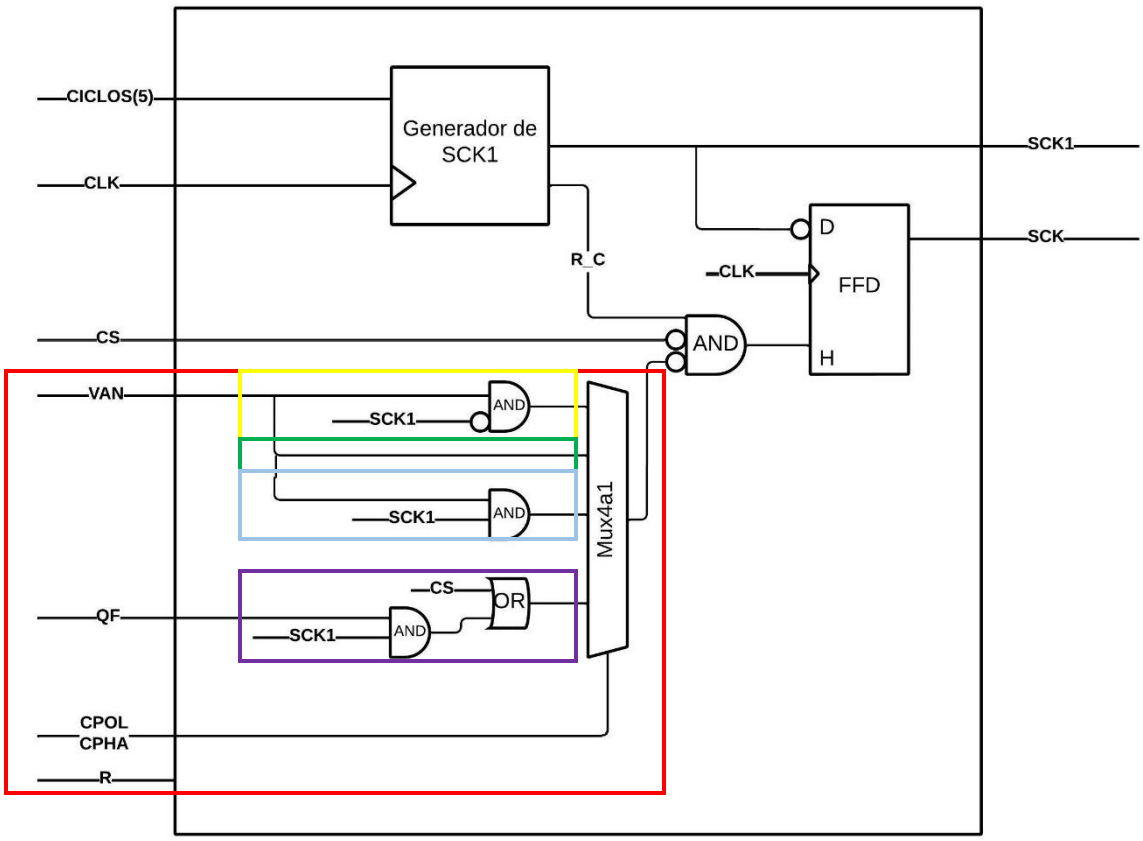


Figura 74. Divisor de frecuencia - Configuración para reposo de SCK

Para los valores de CPOL = ‘0’ y CPHA = ‘0’, se tendrá estado de reposo en bajo del SCK al momento en que el reloj SCK1 tenga un estado bajo y la señal VAN se encuentre en alto. Esta configuración se puede observar en el recuadro amarillo.

Para los valores de CPOL = ‘0’ y CPHA = ‘1’ como se observa en el recuadro verde, solamente es necesario un valor en alto de la señal VAN para poder pausar el reloj SCK y así generar el estado de reposo en bajo. Esto debido a que el reloj SCK tiene el nivel bajo el cual es requerido al momento en que VAN cambia a estado alto.

Para los valores de CPOL = ‘1’ y CPHA = ‘0’, como se observa en el recuadro azul, el SCK1 junto con la señal VAN se espera a que tengan un estado alto para poder deshabilitar el registro. El resultado genera el estado de reposo en alto

Pol último, para los valores de CPOL = ‘1’ y CPHA = ‘1’ se muestra su configuración en el recuadro morado, el estado de reposo del SCK debe de ser en alto por lo que se espera que SCK1 junto con la señal QF tengan un estado alto o simplemente el CS debe tener un estado alto para obtener el estado de reposo deseado.

Así es como el comportamiento del reloj se pausa al final de la transmisión, para habilitarlo de nuevo en cualquiera de las cuatro configuraciones solamente es necesario en un estado bajo del CS el cual se da al iniciar una transmisión.

6.6.2 Solución al problema generado por señales CPOL y CPHA

Hasta este punto se ha mostrado mediante simulaciones el funcionamiento de la señal SCK, del CS y de algunos otros componentes cuando las señales CPOL y CPHA tienen diferentes valores. Para lograr este comportamiento en las señales se requiere de una modificación en la configuración de compuertas que ingresan a los flip-flops que utilizan al SCK y SCK1 como señal de entrada de reloj. La principal causa de esta acción es la necesidad de operar algunas partes del módulo SPI con flancos de subida o de bajada dependiendo del valor que tengan CPOL y CPHA. Se tomaron en cuenta dos alternativas para este proceso. La primera fue la utilización de multiplexores entre el reloj SCK y su negación, esto para poder elegir en el mismo flip-flop si los cambios se darían con flanco de bajada o de subida. El resultado no fue óptimo debido a los retrasos de tiempo que se pueden generar por multiplexar señales de reloj. La segunda alternativa y la cual se eligió para aplicarla en modulo SPI fue la de duplicar los flip-flop que utilizan el SCK, uno de ellos con entrada de reloj SCK y el otro con la misma señal, pero negada y a la salida de cada par de flip-flops utilizar un multiplexor para poder seleccionar la señal correspondiente.

Después se realizó un análisis sobre las 4 diferentes configuraciones posibles y el resultado indica que al momento de aplicar una compuerta XNOR a CPOL y CPHA y a la salida se obtiene un ‘1’ cuando el flanco con el que deben de operar de SCK y SCK1 debe de ser de subida. En caso contrario los registros deben de funcionar con flanco de bajada cuando a la salida de la compuerta XNOR se obtenga un ‘0’. La compuerta XNOR se utilizó en lugar de una XOR para tener como referencia el flanco que se utiliza, es decir, ‘1’ para flanco de subida y ‘0’ para flanco de bajada. Para fines prácticos al resultado de la compuerta XNOR se denominará con el nombre PF por la Polaridad y Fase del reloj.

Los componentes que requerirán duplicar sus flip-flops debido a COL y CPHA son los siguientes:

- Contador 1
- Señal VAN
- CS
- Registros de desplazamiento (TX)
- Señal First_D (TX)
- Registros de desplazamiento (RX)

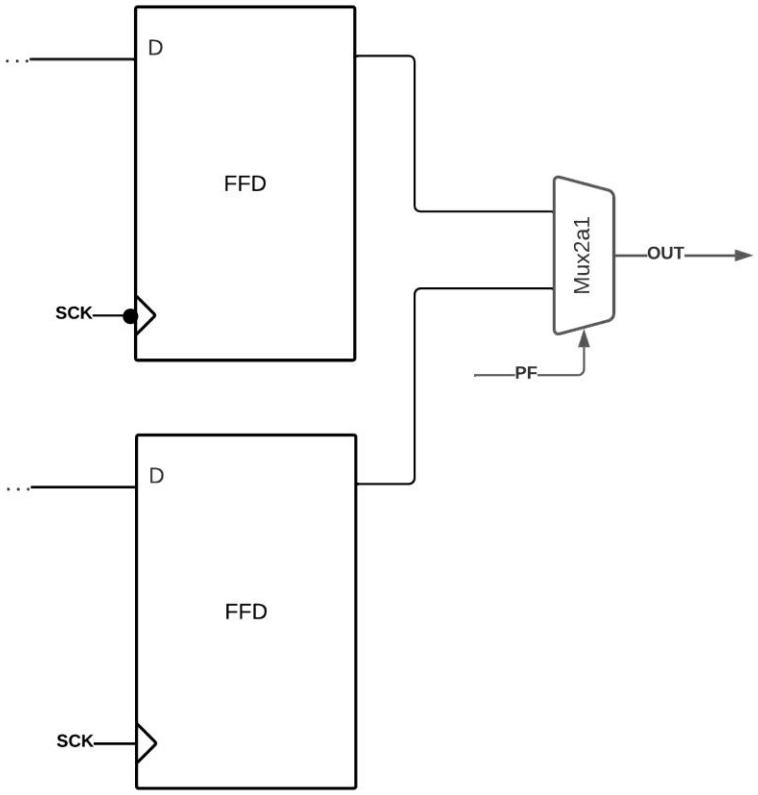


Figura 75. Configuración de registros (PF)En la Figura 75 se muestra el ejemplo de un flip-flop duplicado, todos los flip-flop en esta situación tienen la misma configuración. Se observa que el flip-flop superior tiene como entrada de reloj SCK negado y el inferior con SCK, como dato de entrada tienen el mismo valor, a la salida valor de PF seleccionara el dato según se requiera mediante un multiplexor, de igual manera los flip-flops que funcionen con flanco de subida serán habilitados con un valor de PF igual '1' y viceversa. Cada registro de los componentes anteriormente mencionados fue duplicado debido a que su funcionamiento depende del flanco de reloj del SCK, por ejemplo, para el contador 1 se utilizan 4 flip-flops y al aplicar este nuevo requerimiento se tendrán 8 en total.

6.7 Configuración en modo Esclavo.

Anteriormente el análisis ha sido descrito desde el punto de vista en modo maestro de la interfaz. Para que la interfaz funcione en modo esclavo se realizaron algunas modificaciones en diferentes partes del módulo SPI y en otras se agregaron algunas señales para un correcto funcionamiento.

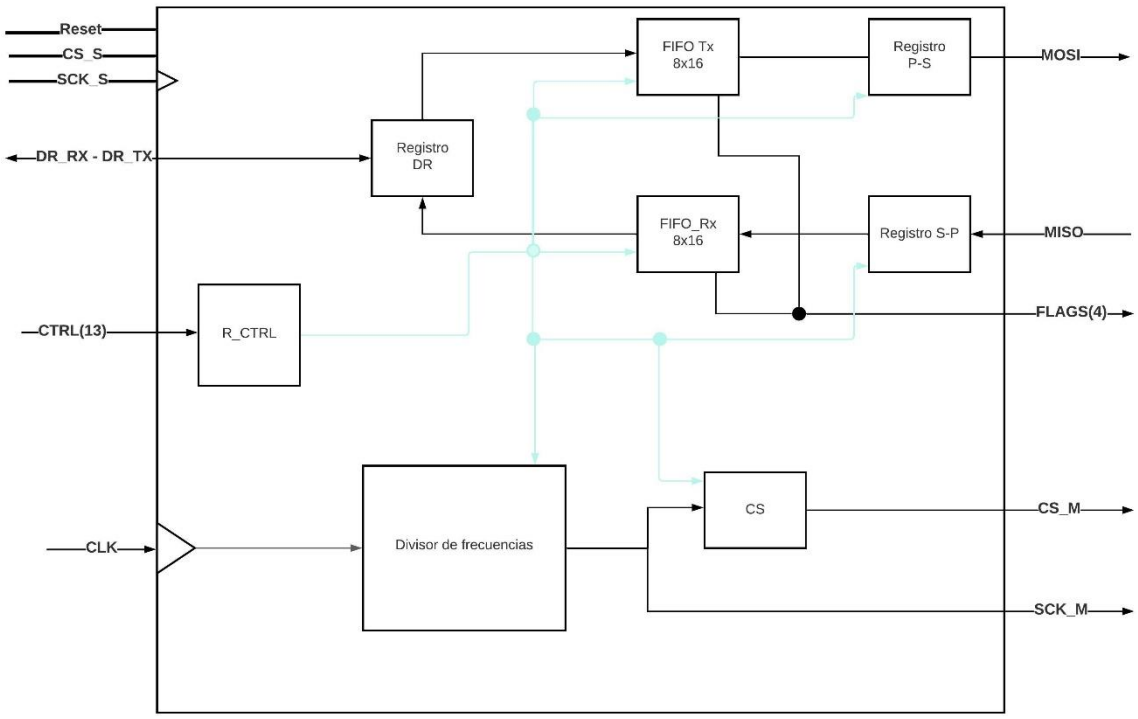


Figura 76. Concepto general en modo maestro.

En la Figura 76 se muestra el diseño general de los componentes y su conexión entre sí. Los componentes son identificables fácilmente por su nombre, en el caso de R_CTRL son los registros de control donde como entrada tiene señales como CPOL, CPHA, DT, S_CS, CICLOS. A todo este diseño se le realizan modificaciones para que pueda tener un funcionamiento tanto en modo maestro como en modo esclavo y este sea elegible mediante los registros de control.

En la sección 6.6.2, se nombraron 6 componentes que necesitaban duplicar los flip-flop para poder hacer configurable el módulo mediante CPOL y CPHA. Ahora se procederá a realizar un ajuste similar para poder agregar la entrada MS la cual permitirá elegir operar el módulo SPI como maestro o esclavo.

Los flip-flop de los componentes de la siguiente lista a pesar de ser duplicados anteriormente se volverán a duplicar para que la interfaz pueda ser elegible a utilizar en modo esclavo. Esto debido a que la señal de reloj SCK a utilizar en modo esclavo no será la misma que en modo maestro lo cual hace necesaria esta modificación ya que si se utiliza un multiplexor entre relojes se pueden generar retrasos en el resultado final.

Los registros de los siguientes componentes se vuelven a duplicar.

- Registros de desplazamiento RX
- Registros de desplazamiento TX
- FIRST_D TX
- Contador 1

Se observa que en comparación con los componentes de la sección anterior que se requirieron duplicar hace falta el CS y VAN, esto porque son señales necesarias para

generar el CS en modo maestro y como en modo esclavo no son necesarias se omitieron en este proceso.

Recodando, la señal MS indica con ‘0’ la utilización de la interfaz en modo maestro y con ‘1’ en modo esclavo. Esta es utilizada para poder seleccionar el CS del maestro o el esclavo mediante un multiplexor 2 a 1. Fuera de este caso particular, la señal MS tiene la función de seleccionar junto con PF la salida requerida según corresponda de los flip-flop que se duplicaron.

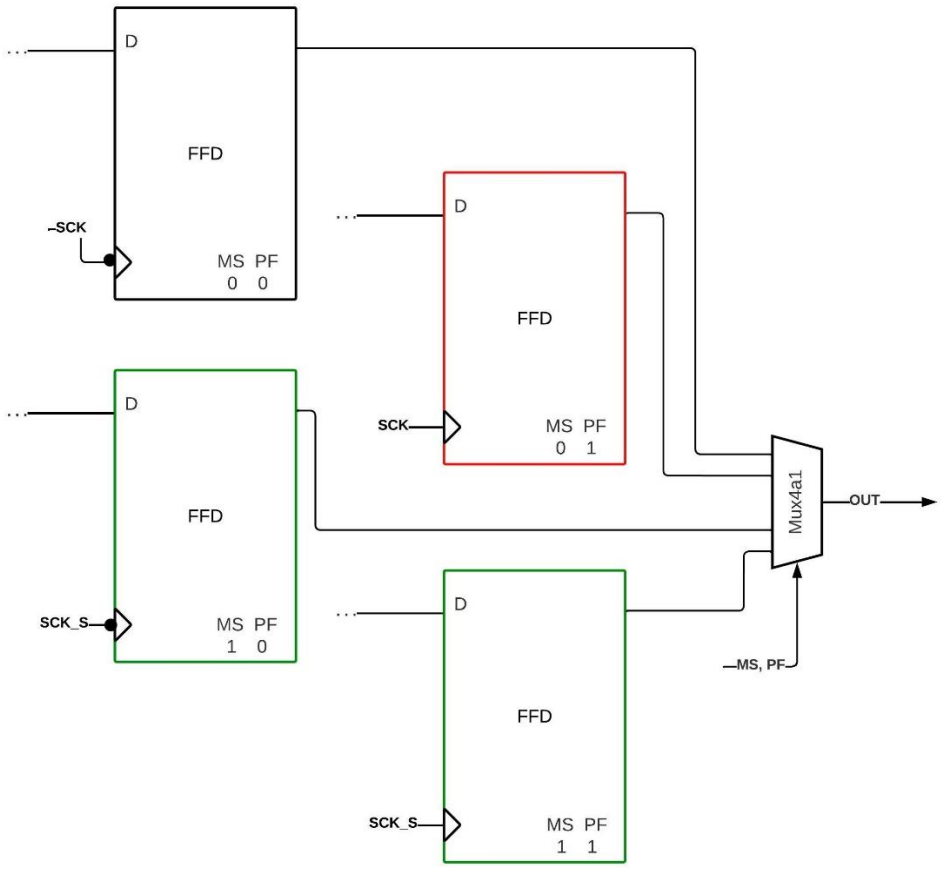


Figura 77. MS - Cuadruplicación de registros.

En la Figura 77 se presenta visualmente como quedaría cada registro con la última duplicación requerida. Se observa un flip-flop en color rojo el cual es el principal, este se utiliza con PF= ‘1’ en modo maestro. También se observa uno en color negro el cual es su duplicación, es decir, el cual se utiliza con PF = ‘0’ en modo maestro. Ahora, fue necesario realizar una duplicación de estos dos flip-flop antes mencionados ya que esos son utilizados cuando la interfaz se encuentra operando en modo maestro y se utiliza el SCK generado por la propia interfaz SPI, mientras que los flip-flop que se pueden observar en color verde utilizan una señal de reloj externa la cual identificaremos como SCK_S, es decir, estos flip-flop son utilizados en modo esclavo, realizan la misma acción, pero con una señal de reloj externa. Por último, la salida de cada flip-flop es dirigida a un multiplexor para ser seleccionada dependiendo de los valores de MS y PF.

Poniendo como ejemplo los registros de desplazamiento RX, en donde se reciben los datos desde otro dispositivo, este utiliza 16 flip-flops para su operación, al momento de duplicarlos debido a los valores de CPOL y CPHA el número total de flip-flops asciende a 32. Ahora, se vuelven a duplicar estos flip-flops para poder utilizar la interfaz en modo esclavo dando un total de 64 flip-flop para este componente.

En general se cuatriplicarían los flip-flops de los componentes de la última lista mencionada Algo muy importante a tener en cuenta es que la señal SCK_S se recibe desde fuera de la interfaz y no es posible predecir cuándo tendrá funcionamiento y cuando estará en modo reposo, debido a esto se presentó un problema con el contador 1 cuando se da un reinicio a la interfaz. La falla se presenta al momento de un reinicio, el reset puede durar un pulso de reloj del CLK como mínimo para reiniciar la interfaz, el contador 1 en modo esclavo debido a que utiliza como entrada de reloj el SCK_S y su frecuencia de operación es menor al CLK puede que no se reinicie ya que no asegura que haya un del flanco de subida del SCK_S en el contador 1 mientras el reset este en '1'. Esto da lugar a que la cuenta comience en cualquier otro valor que no sea 0 cuando comienza la siguiente transmisión. Es decir, al momento de no reiniciar el contador 1 junto con toda la interfaz la cuenta será incorrecta después de un reinicio lo cual generaría un almacenamiento de datos en cada una de las memorias con errores.

Para poder resolver el problema es necesario comenzar la cuenta del contador 1 en 0 al momento en que se comience una transmisión cuando se trabaja en modo esclavo. Por lo tanto, se agregará una condición antes de los multiplexores de los registros dedicados al modo esclavo para poder establecer un valor de cero al inicio de cada transmisión como se puede ver en la Figura 78 mediante los recuadros con signo de interrogación.

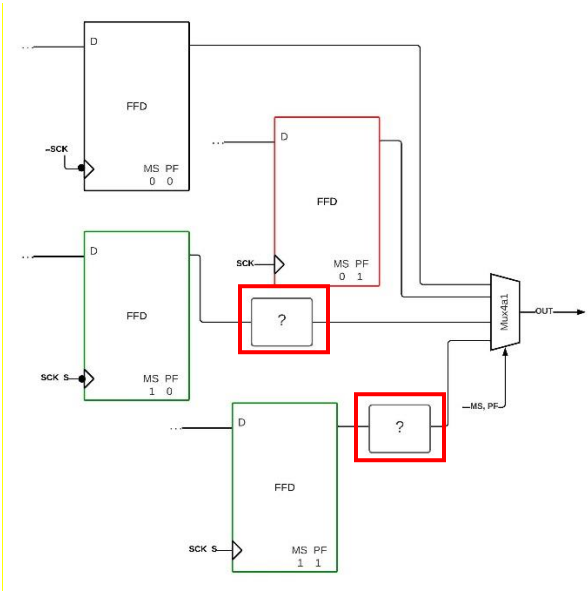


Figura 78. MS - Contador 1 condicionando

En la Figura 79 se muestra la configuración del contador 1, en el rectángulo superior se observa cómo está conformado internamente cada flip-flop, en este se muestra que se

tiene la cuadruplicación aplicada. La salida de cada uno va dirigida a la configuración de compuertas encargadas de realizar el conteo.

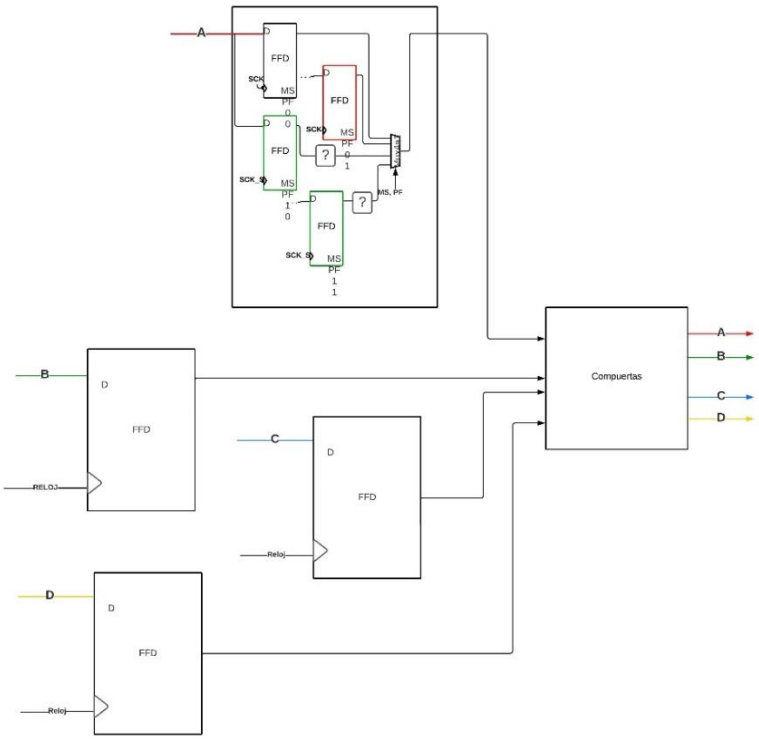


Figura 79. Contador 1 - Flip Flop cuatriplicados

Para tener un valor de 0 en el contador 1 es necesario forzar los valores de cada flip-flop a 0 hasta el primer flanco de captura de bit del SCK_S, esto con el fin de que el siguiente estado sea igual a 1 del contador 1. En la Figura 80 se muestra con un recuadro rojo el tiempo en el que las entradas al contador 1 deben de ser cero cuando se utilice la configuración de reloj PF=’1’ y en azul cuando PF=’0’.

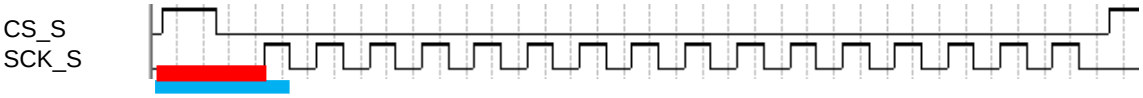


Figura 80. MS - Condición de cero.

Ahora, para poder generar este comportamiento fueron necesarios 3 flip-flops, uno con el CLK como entrada de reloj el cual tiene gran importancia y los dos restantes con SCK_S.

Es necesario asegurar el inicio de cuenta en cero al iniciar la transmisión, de esto se encargará el primer registro el cual se muestra en la Figura 81. Este tiene la función de mantenerse en estado alto cuando el CS_S se encuentra en alto o el contador 1 en modo esclavo tenga un valor diferente de cero.

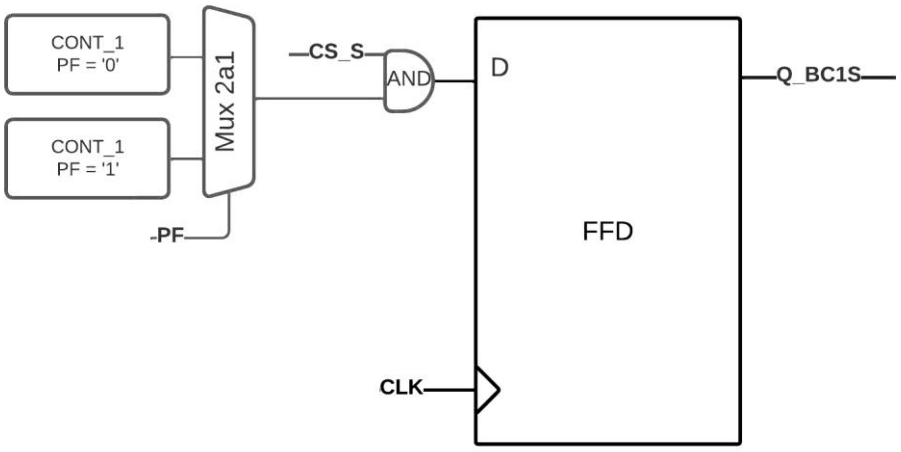


Figura 81. MS - Registro 1 de condición para contador 1.

En la Figura 81 se observan dos recuadros con la leyenda “CONT_1”, esto hace referencia al valor del contador 1. En este caso se utilizan 4 flip-flop para contador 1 con PF = ‘0’ y otros 4 para PF = ‘1’. Una vez aclarado esto, el funcionamiento general de la Figura 81 indica cuando existe un error en la cuenta, es decir, al momento en que el CS_S se encuentre en alto el contador 1 debe de tener un valor de 0 y si esto no sucede es porque hubo un error o un reinicio de la interfaz.

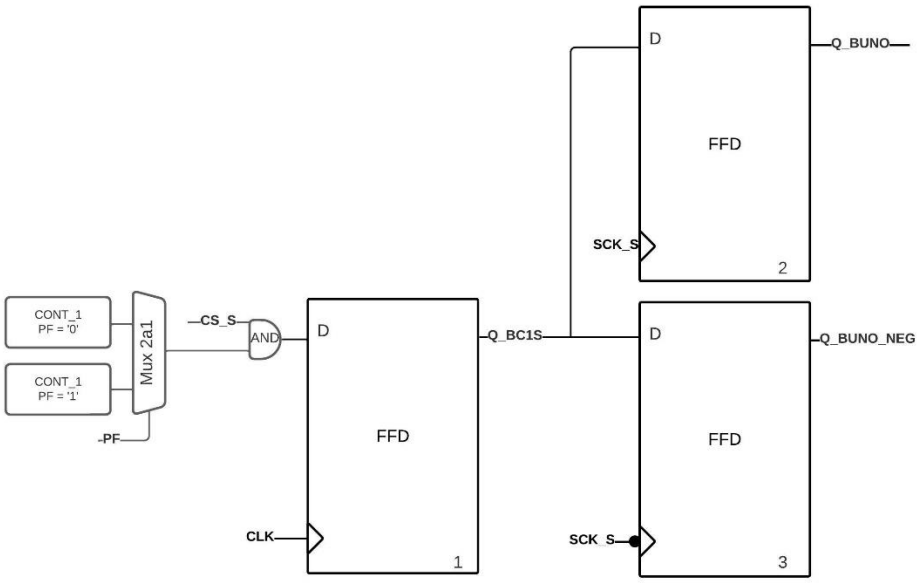


Figura 82. MS - Condición del contador 1.

Para solucionar este problema, al último diseño le agregaron otros dos flip-flop los cuales funcionan como banderas. Estos flip-flop son utilizados en caso de reiniciar la interfaz, tienen la función de asegurar un valor de cero del contador 1 aunque la salida de cada flip-flop de este contador será diferente de 0.

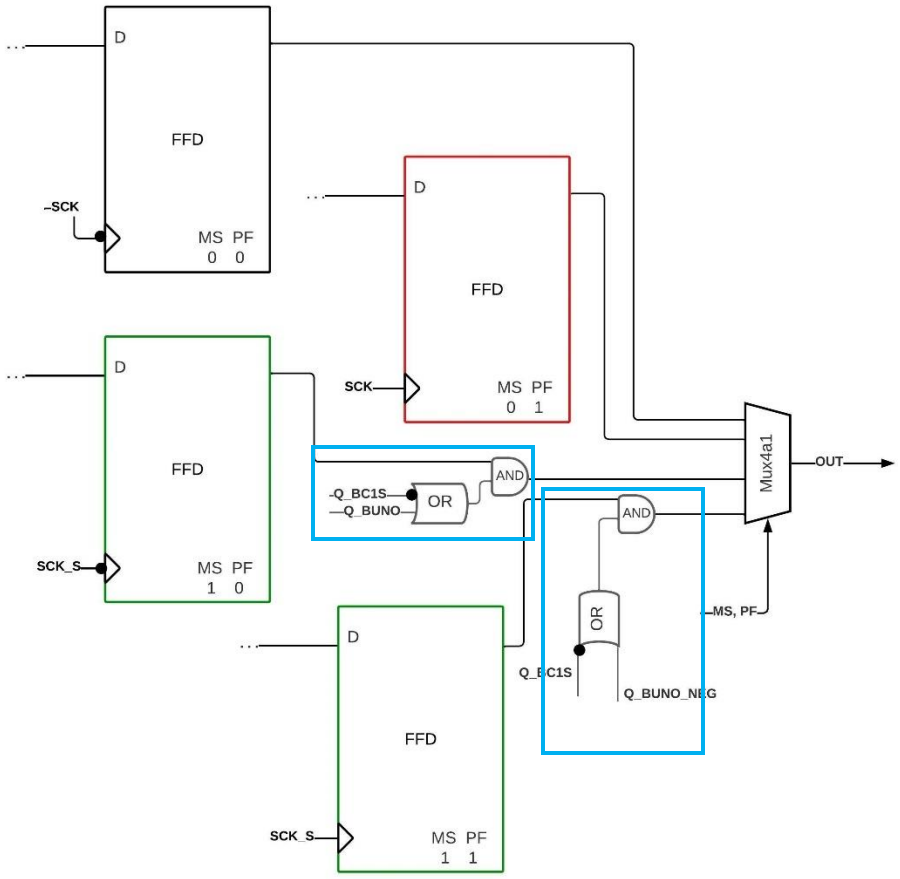


Figura 83. MS - Registro de contador 1.

En la Figura 83 se observa que se agregaron las señales anteriores de tal manera que el reinicio de la interfaz no generara ningún error en la cuenta. Q_BC1S debido a que solamente es utilizada por si existe algún error en la cuenta se utiliza su negación y Q_BUNO junto con Q_BUNO_NEG tienen un valor en alto siempre y cuando todo se encuentre funcionando en tiempo y forma.

Una vez analizado esto e implementado en la descripción de hardware se procede a mostrar el funcionamiento de la interfaz. Primero se mostrarán simulaciones en donde el funcionamiento sea correcto para posteriormente visualizar el error y su solución.

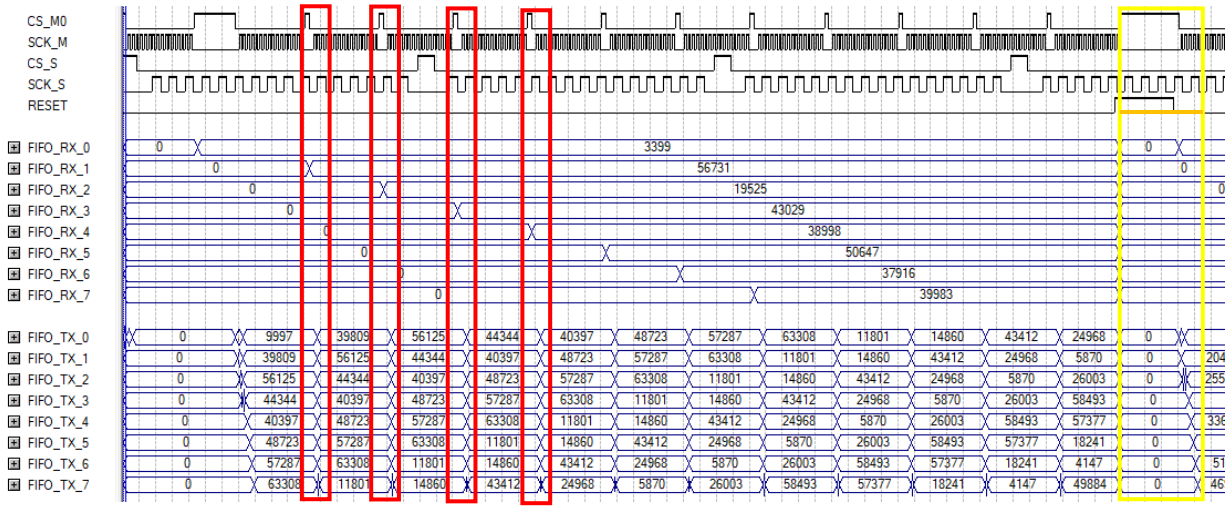


Figura 84. MS - Modo maestro

Al momento de simular la interfaz, se introduce manualmente una señal de CS_S y SCK_S para verificar su funcionamiento. En la Figura 84 se muestran las memorias FIFO de transmisión y recepción, el CS y el SCK tanto para maestro como para esclavo solamente para observar la diferencia en frecuencia entre los dos modos de operación. Los recuadros rojos indican cuando se comienza y termina una trama, también se observa en la memoria de transmisión el desplazamiento de datos y en la memoria de recepción el llenado de la memoria. Esta simulación corresponde al modo maestro por lo que se utilizó el CS_M0 y SCK_M. Por último, el recuadro amarillo muestra un reinicio, el cual indica una limpieza total en las memorias y otros componentes.

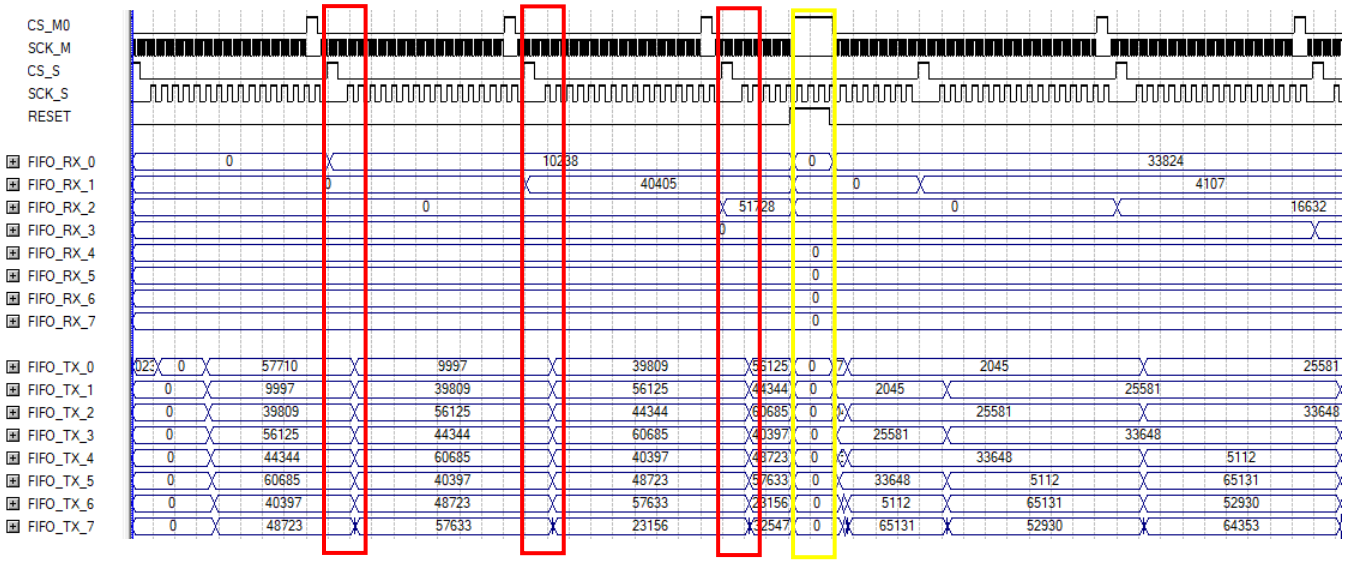


Figura 85. MS - Esclavo

Ahora se tiene en la Figura 85 la simulación de la interfaz operando en modo esclavo, se puede seguir generando las señales SCK_M y CS_M o pueden ser deshabilitadas, en este caso se muestran dichas señales para poder ver su comportamiento cuando la interfaz se encuentra en modo esclavo. Se observa un buen funcionamiento al almacenar y extraer

datos en las memorias, en los recuadros rojos se observa el momento en que en la memoria de transmisión se extrae el dato y en la de recepción de almacena. De igual manera se logra que todos los datos en las memorias y las señales tengan un valor de 0 en caso de reiniciar la interfaz como se observa en el recuadro amarillo.

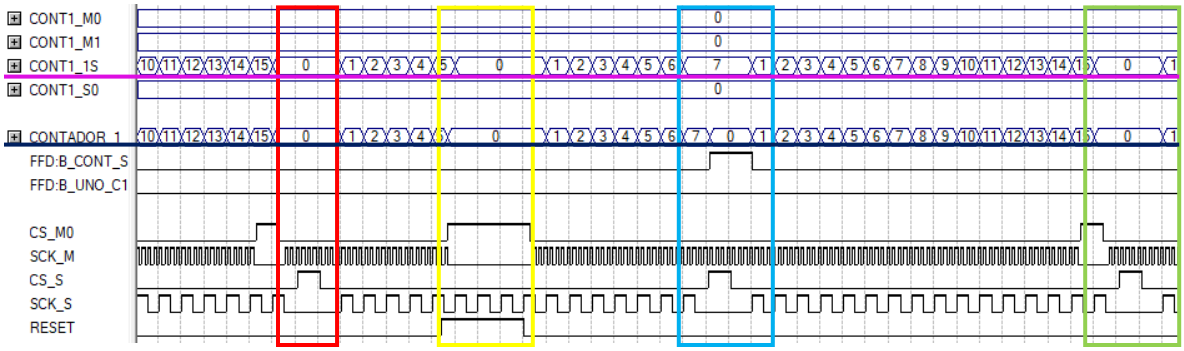


Figura 86. MS - Reinicio del contador 1 en modo esclavo.

Por último, al momento de implementar el diseño que soluciona el error del contador 1 en modo esclavo ya no se tendrá ningún problema al momento de reiniciarlo. En la Figura 86 se muestra la simulación de la interfaz en modo esclavo dando un reinicio y viendo el comportamiento del contador 1. El recuadro rojo indica el momento en donde este contador opera con normalidad, se realiza una trama y no existe problema alguno. El recuadro amarillo indica el momento en que se reinicia la interfaz por completo, todas las señales regresan a su valor predeterminado. En este caso como el SCK_S y CS_S son señales externas siguen operando con normalidad, pero el contador 1 después del reinicio su valor regresa a cero lo cual es correcto hasta ahora, este se puede observar en la línea purpura.

En el recuadro azul se observa el instante en donde se termina la transmisión y comienza una nueva, en este momento el contador 1 debe de tener un valor de cero, pero termina en 7. Gracias a que el problema se solucionó, al momento en que el CS_S se encuentra en estado alto, los valores del Contador 1 son forzado a cero y se puede continuar con una transferencia de datos sin ningún problema. Cabe destacar que el valor de CONT1_1S subrayado en purpura es el valor de salida de los flip-flop en modo esclavo mientras que el valor de CONTADOR_1 subrayado en negro es el valor final corregido. Después del recuadro azul la cuenta continua correctamente y se observa en el recuadro verde de que el problema fue resuelto.

6.8 Reset

Cuando el programa se inicializa los flip-flop de la interfaz tienen de manera predeterminada un valor de ‘0’ en su mayoría, estos valores van cambiando conforme se vaya utilizando el módulo SPI al transmitir un dato o cambiando algún valor de los registros de control. Entonces, el reinicio del programa genera una limpieza total de todas las modificaciones en estos valores y los devuelve a el estado predeterminado inicial.

En esta sección se analizará el componente de reinicio. Primero se debe de tener en cuenta que la opción de reiniciar la interfaz con un pulso de duración de un periodo del CLK no es viable dado que se llegan a utilizar hasta 3 señales de reloj las cuales operan a diferentes frecuencias cada una. Esto dio lugar a reiniciar la interfaz en partes.

Si se reinicia todo al mismo instante el flip-flop encargado de generar la señal SCK1 (señal de sincronía en modo maestro) permanecerá en cero mientras dure el reinicio y los flip-flop que operan con esta señal no tendrán ningún cambio, por eso en la primera etapa del Reset se reiniciaran los flip-flop que operan con esta señal tanto con flanco de subida como de bajada. La segunda etapa permite el Reset de los registros faltantes, es decir, los que operan con el CLK.

Mientras estas dos etapas se llevan a cabo, los flip-flop que operan con el SCK_S y SCK puede que se no regresen a sus valores predeterminados porque el reinicio se puede dar mientras no se tenga transmisión de datos y por ende no se tengan flancos de subida o bajada de dichas señales sino solamente un estado alto o bajo.

Las señales de reloj de mayor interés son el CLK, SCK1, en el caso de la señal del esclavo SCK_S junto con el SCK se tendrá un caso particular ya que no se tiene control sobre ellas ya que dependen de que haya transmisión al momento del reinicio.

En la Figura 87 se muestran las señales de entrada y salida del componente de reinicio. Se tienen como entrada dos señales de reloj por lo anteriormente mencionado, el reset el cual es una señal externa y el CS generado en modo maestro. Por el lado de las salidas se tienen 3 señales dedicadas para reiniciar un grupo de flip-flop cada una.

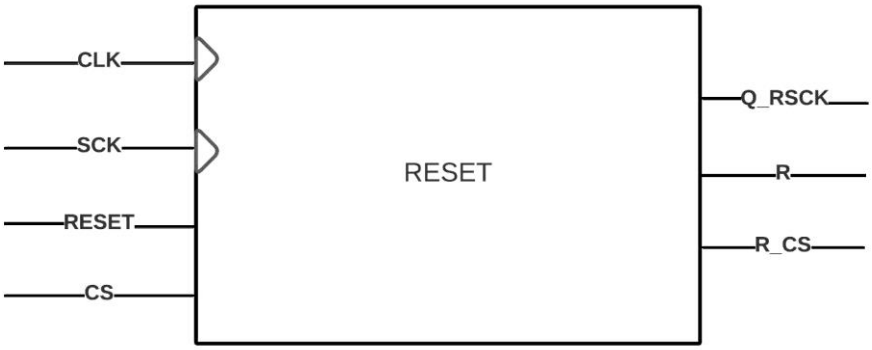


Figura 87. Reset - Entidad.

La señal Q_RSCK será la encarga de realizar la primera etapa de reinicio. La señal R tendrá la tarea de la segunda etapa y por último R_CS asegurará un valor de cero en el contador 1 al inicio de cada transmisión.

A continuación, se comenzará analizando la configuración de compuertas del Reset para posteriormente ver su aplicación dentro de la interfaz y el impacto que tiene.

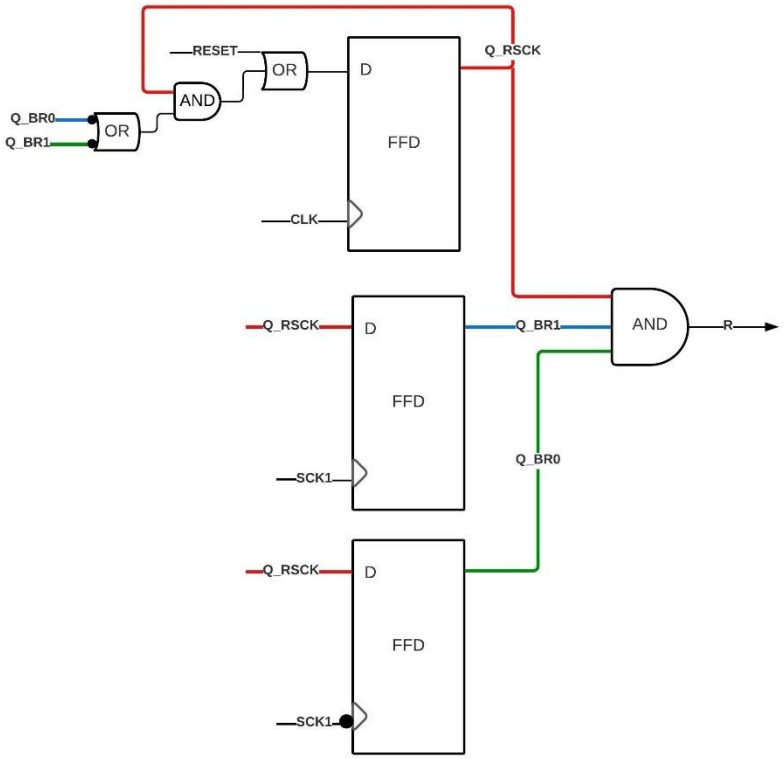


Figura 88. Reset - Configuración de registros

En la Figura 88 se muestra la configuración de registros para poder reiniciar el sistema. En este caso el registro superior el cual funciona con CLK, tienen la función de capturar el pulso de entrada de RESET y mantenerlo en estado alto el tiempo que sea necesario, aunque la señal RESET regrese a estado bajo. Una vez Q_RSCK se encuentre en estado alto se introduce a los dos registros restantes, estos funcionan como bandera indicando cuando ya se reiniciaron los flip-flop que operan con flanco de subida y posteriormente los de flanco de bajada o viceversa. Los registros que utilizan al CLK se reinician con R, este proceso se da cuando ya se reiniciaron los flip-flop que operan con SCK1, antes no.

Esta condición de R es necesaria ya que el reloj SCK1 funciona con el CLK, entonces al momento de reiniciar este registro no se podrán reiniciar los que funcionan con esta señal de reloj debido a que permanecerá en estado bajo siempre que el reset este activado. Por esta razón los registros que utilicen el CLK como entrada de reloj son los últimos en reiniciarse.

A continuación, se muestran los dos ejemplos de su funcionamiento.

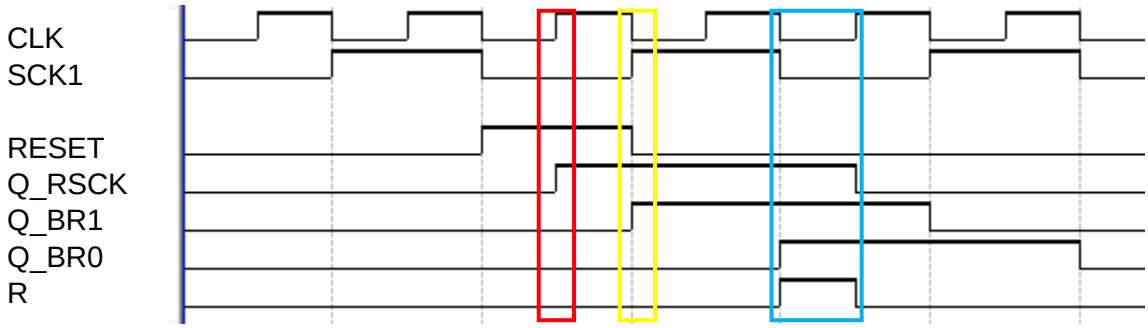


Figura 89. Reset - Simulación 1

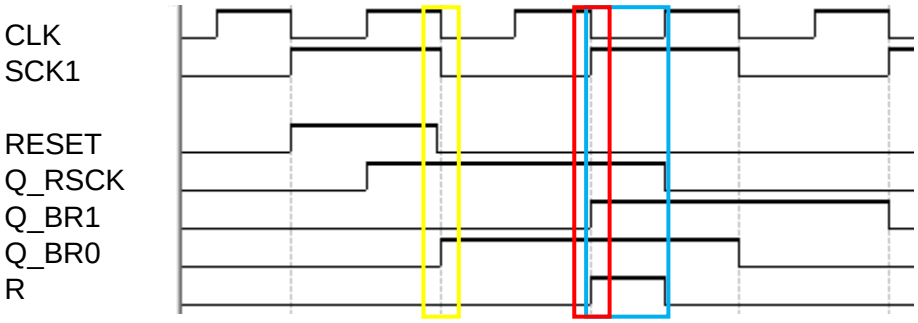


Figura 90. Reset - Simulación 2

En la Figura 89 se muestra el primer caso que puede ocurrir. En el recuadro rojo se captura la señal de entrada RESET en Q_RSCK, esta tendrá un estado en alto hasta que la primera etapa de reinicio se complete. La señal Q_BR1 indica con '1' cuando aquellos flip-flop que utilizan el flanco de subida del SCK o SCK1 han sido reiniciados, de igual manera Q_BR0 indica con un '1'cuando los flip-flop que utilizan el flanco de bajada de SCK o SCK1 han sido reiniciados. Por último, en el recuadro azul el valor de R se encuentra en estado alto lo cual quiere decir que la segunda etapa de reinicio está en ejecución, y se reinician los flip-flop que utilizan el CLK como reloj de entrada.

En resumen, la señal R_QSCK reinicia los registros que utilizan como fuente de reloj SCK y SCK1 tanto para flanco de subida como de bajada, y R para poder reiniciar los que utilizan al CLK. Q_BR0 y Q_BR1 son banderas que indican que el reinicio de flip-flop correspondientes a sido completado.

6.9 Funcionamiento DR – FIFO – SALIDA.

Una vez analizado cada componente de la interfaz y su configuración interna de compuertas se procederá a ver el funcionamiento general. No se tendrá visualización directa de cada componente, pero las más importantes se podrá ver en diferentes modos de operación.

Se comenzará con un ingreso aleatorio de datos a los registros DR mediante el bus TDR. También se visualizará el almacenamiento de datos en la memoria tanto en la memoria FIFO_TX como en la FIFO_RX, el funcionamiento de reloj SCK y la generación del CS, de igual manera se tendrá una interacción con la interfaz cambiando los parámetros DT, CICLOS, CPOL, CPHA y MS.

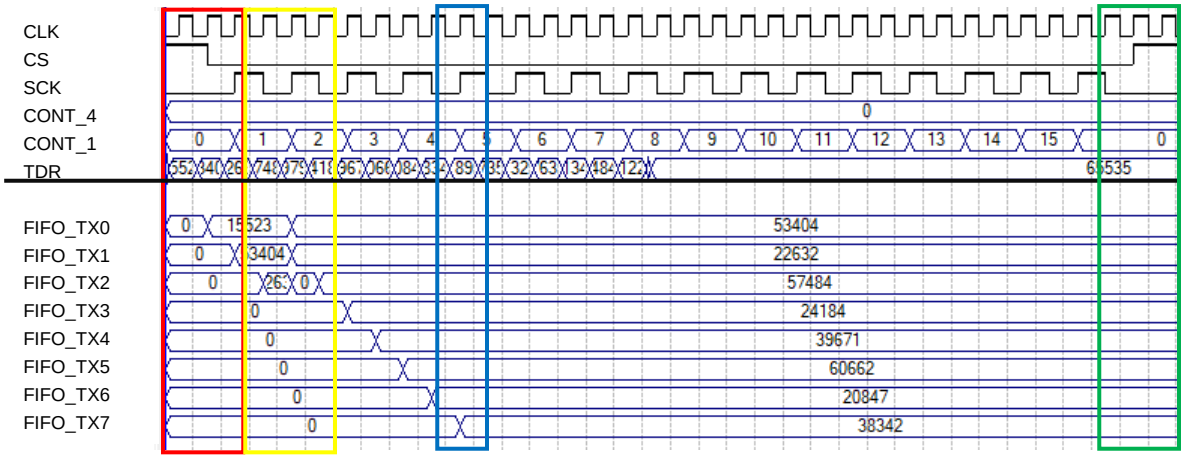


Figura 91. Funcionamiento general - FIFO TX

Se comienza la primera prueba con los valores predeterminados de los registros de control en donde se utiliza la interfaz en modo esclavo, se opera a 50 MHz, 16 bits en cada dato, CPHA = '0' y CPOL = '0', como se logra ver en la Figura 91 el funcionamiento de la interfaz con todos los componentes interconectados. La interfaz comienza sus operaciones desde el primer momento en que se ingresa un dato en TDR, sin este primer dato el CS y SCK se mantienen en estado de reposo lo cual no genera actividad en la interfaz. Después de ingresado el primer dato, se almacena en los registros DR de transmisión, como se observa en el recuadro rojo, posteriormente se guarda en la memoria FIFO_TX (primera fila) y al mismo instante se comienza con la transferencia de datos.

En el recuadro amarillo se continúa ingresando datos a la FIFO_TX mientras se da una transferencia. En el recuadro azul se indica una memoria llena por lo que ya no se ingresan más datos. Por último, en el recuadro verde se observa el final la transmisión, y el reloj SCK junto con el CS y el contador 1 regresan a sus valores iniciales.

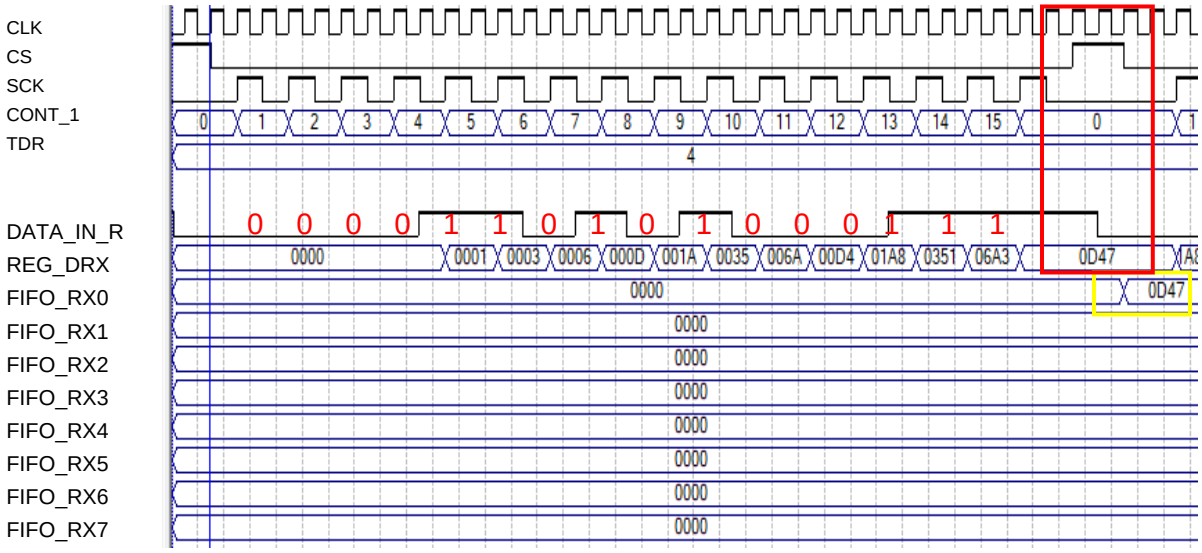


Figura 92. Funcionamiento general - Dato de entrada en la recepción y dato de salida en transmisión

En la Figura 92 se muestra el funcionamiento de recepción. Se ingresa un dato serial mediante DATA_IN_R el cual tiene un valor de 0x0D47. En REG_DRX se almacena de

manera serial el dato hasta completar la recepción y tener el dato esperado como se observa en el recuadro rojo. Por último, en el recuadro amarillo es almacenado en la memoria para posteriormente ser extraído y utilizado.

Un caso particular en la memoria FIFO de recepción es cuando las señales de ingreso y extracción de dato suceden al mismo tiempo. En la Figura 92 se observa que se ingresa un dato a la memoria justo antes de que el CS comience la siguiente transmisión. En la Figura 93 se observa la extracción de un dato de la memoria FIFO_RX mediante la señal EXT como se observa en el recuadro rojo, en ese mismo instante se debe de ingresar el dato a la memoria. La prioridad la tiene la extracción para después ingresar el dato al siguiente pulso de reloj del CLK (recuadro amarillo).

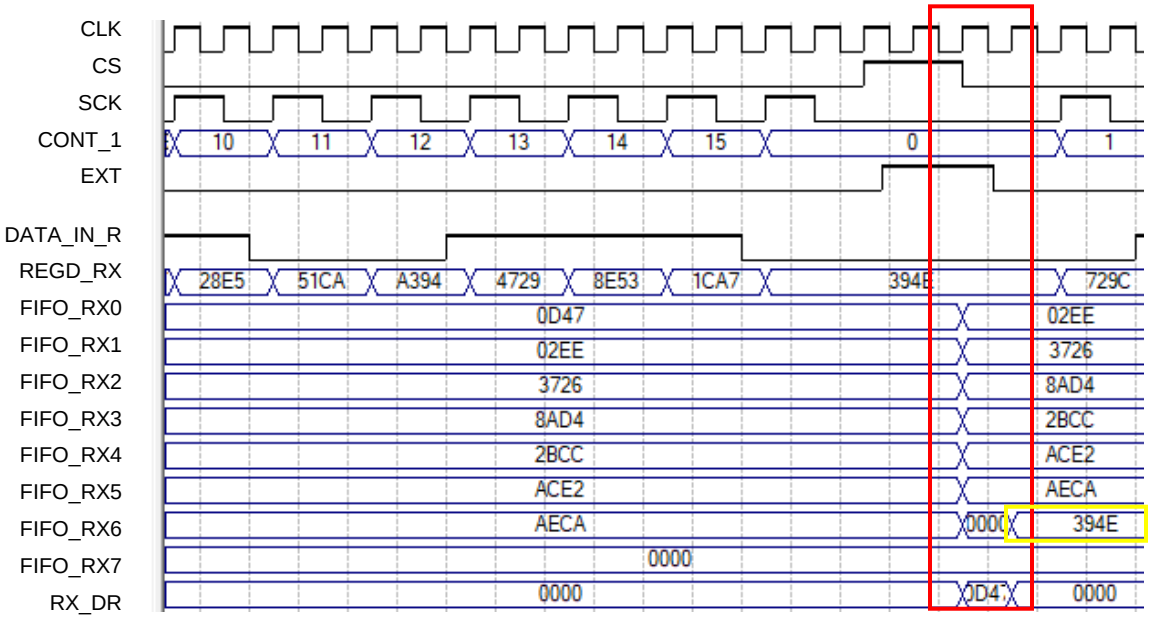


Figura 93. Funcionamiento general - Recepción ingreso y extracción de datos.

En la Figura 94 es cambiando DT a un valor de 4 bits, CPOL = ‘1’, CPHA = ‘0’, y CICLOS = 3. Con esto se observará el correcto funcionamiento que se tiene aun cambiando estos parámetros configurables. Al modificar el numero de bits de cada dato afecta de primera instancia al CS y SCK_M como se observa en el recuadro rojo. La memoria FIFO tanto de transmisión como de recepción guardan valores muy altos y muy bajos, respectivamente. Esto es debido al valor de DT como se vio en la sección 6.4, en la memoria de transmisión los datos son almacenados en los registros de mayor significancia mientras que en recepción se almacenan en los de menor. Cuando un dato (en este caso) no tiene valores de ‘1’ en sus 4 bits menos significativos se almacena en la memoria FIFO como un 0, esto no es lo mismo a no tener dato, el valor del dato es igual a 0 como se observa en el recuadro amarillo.

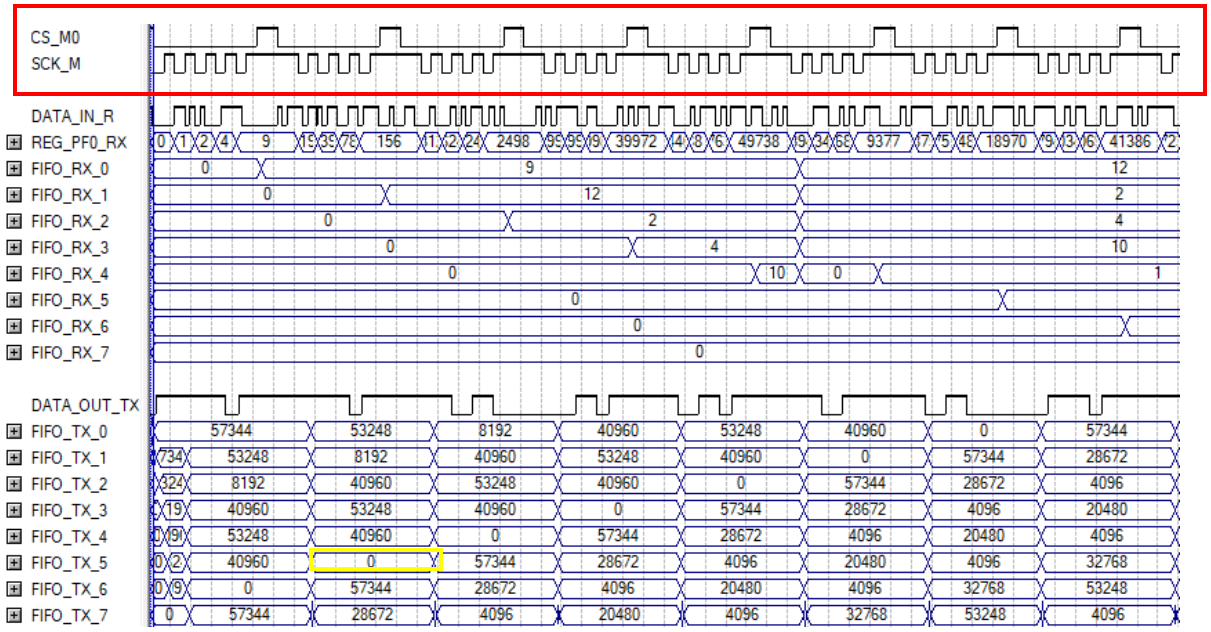


Figura 94. Funcionamiento general - Parámetros modificados.

En segundo lugar la velocidad de operación del SCK se redujo de 50 MHz a 12 MHz y una configuracion de reloj CPOL y CPHA de “10”. En este caso el SCK tiene su estado de reposo en alto y un total de 4 pulsos de reloj para la tranmisión.

En estos dos ejemplos es posible ver un buen funcionamiento en todos los componentes aunque los parametros configurables cambien. Solamente se mostraron estas dos simulaciones ya que serian muy repetitivas las pruebas, anterioremente se observa un funcionamiento individual de cada parte del programa y la unificacion de toda la interfaz no tiene ninguno problema ni cambio

Por ultimo el resultado de los recursos utilizados en el programa se podra ver a continuación en la Figura 95 donde se tiene el reporte de compilación.

Flow Status	Successful - Mon Oct 02 20:42:59 2023
Quartus II Version	9.1 Build 350 03/24/2010 SP 2 SJ Web Edition
Revision Name	SPI_PRINCIPAL
Top-level Entity Name	SPI_PRINCIPAL
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	No
Total logic elements	855 / 18,752 (5 %)
Total combinational functions	788 / 18,752 (4 %)
Dedicated logic registers	452 / 18,752 (2 %)
Total registers	452
Total pins	62 / 315 (20 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 95. Reporte de compilación final.

Los registros utilizados totales son 452, en su mayoría FFD. Para las simulaciones, en el software se utilizó una tarjeta de la familia CYCLONE II solamente para poder describir la interfaz en VHDL.

6.10 Registros de control, datos y estado.

Se utilizan registros para poder almacenar bits que permiten configurar la interfaz, escribir o leer datos y visualizar el estado de algún componente mediante banderas. La configuración utilizada para estos grupos de registros de desplazamiento es paralelo – paralelo.

Los registros de control en este caso se utilizan para cambiar los parámetros de entrada de la interfaz como son S_CS, DT, CICLOS, CPOL, CPHA y MS, con estos puedes generar un cambio interno solamente al cambiar cualquier bit de los antes mencionados.

Los registros DR de escritura sirven de igual manera para escribir, pero estos no generan un cambio interno ya que los datos ingresados a estos registros serán transmitidos desde la interfaz a otro dispositivo.

En el caso de las banderas y los registros DR de lectura, consiste en realizar una visualización de los bits de cada una de ellas para poder tomar una decisión o acción. En el caso de los registros DR de lectura también se puede tomar el mismo valor para procesarlo.

Para el caso de los registros de control, pueden cambiar de valor siempre y cuando el CS se encuentre en estado alto, en caso de haber una transmisión no es posible actualizar el valor de estos registros hasta que se termine trama. En el caso de los registros DR el dato permanece un ciclo de CLK y no se puede escribir en ellos cuando QSCK2 o EXT se encuentran en alto en el caso de los de escritura y lectura, respectivamente. Para las banderas, se depende del valor que tenga el contador 2 y 3, y siempre se encuentran habilitados.

Estos grupos de registros tendrán una gran importancia más adelante ya que tendrán una conexión directa con la Interfaz AXI4.

6.11 Tolerancia a fallas

El segundo objetivo más importante de este proyecto es realizar una IP con técnicas de tolerancia a fallas, el primero es poder generar la interfaz SPI a nivel compuerta en VHDL. Como se vio anteriormente, la redundancia modular triple (TMR) puede ser aplicada en diferentes puntos de la descripción, así como en compuertas, registros o las dos al mismo tiempo. El punto malo de esto es la utilización de recursos de la tarjeta ya que es necesario triplicar desde una compuerta hasta un componente completo

La técnica utilizada que se integró al proyecto fue la TMR a los registros solamente, ya que estos almacenan datos y si alguno de ellos es alterado mientras se encuentra en un conteo, es almacenado, cambia alguna bandera o genera el cambio en alguno de ellos puede repercutir gravemente en el funcionamiento total de la interfaz.

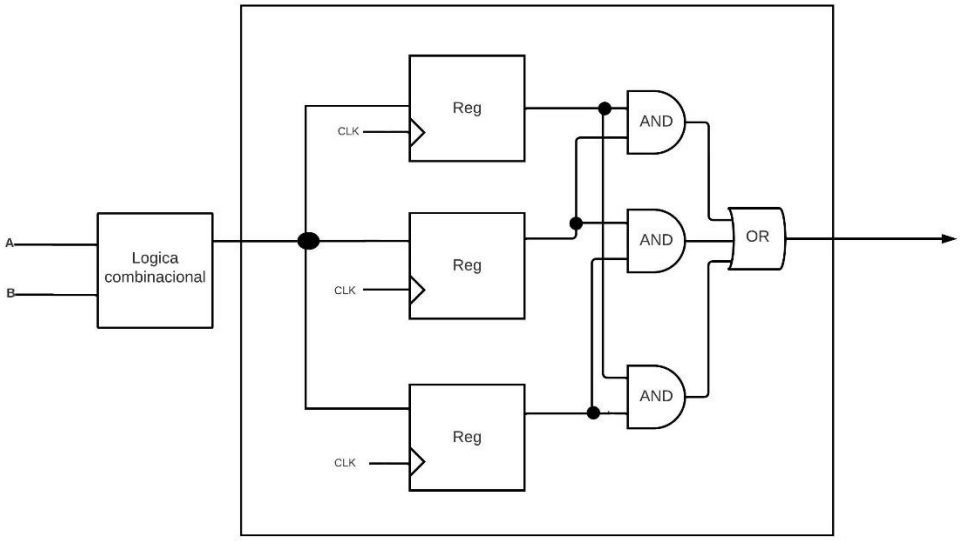


Figura 96. TMR - Registros.

En la Figura 96 se observa aplicada a un registro la TMR, después de la salida de los registros se observan 4 compuertas las cuales son lo que conforma el “voto mayoritario”.

Desde la Figura 97 hasta la Figura 99 se muestran los ejemplos de las diferentes fallas que pueden ocurrir. Las líneas rojas indican un dato incorrecto, mientras que las verdes indican un funcionamiento correcto. Como se sabe, este modelo de protección solo es funcional si solamente uno de los tres registros falla porque si existen dos fallas al mismo tiempo entonces el dato incorrecto es el que tendría mayoría por lo que generaría un error a la salida. Es decir, el dato correcto por no tener la mayoría, el sistema lo interpreta como un error.

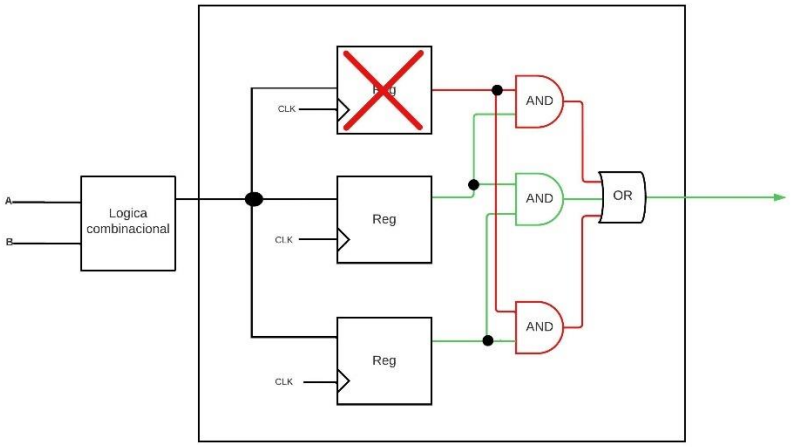


Figura 97. TMR a registro - falla en el primer registro

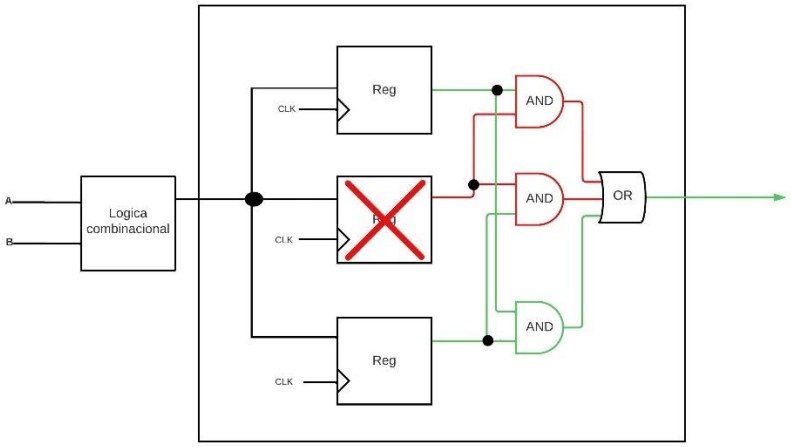


Figura 98. TMR a registro - falla en el segundo registro

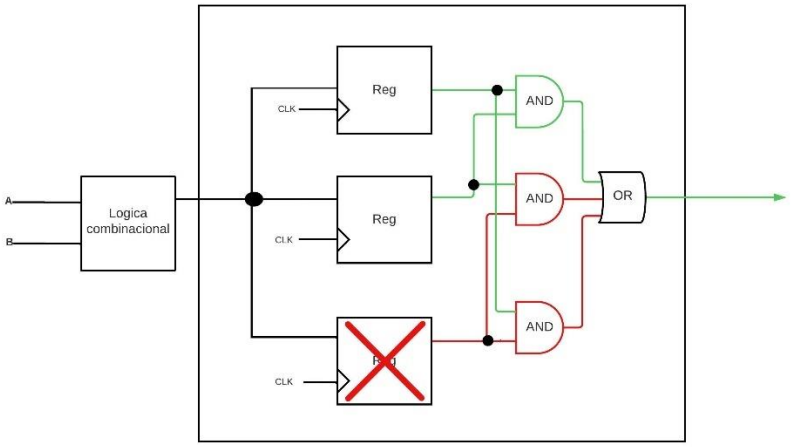


Figura 99. TMR a registro - falla en el tercer registro

Para lograr aplicar esta triple redundancia a la interfaz fue necesario modificar solamente los archivos de los flip flops D y JK. A continuación, se mostrará la arquitectura del código de cada uno de ellos junto con el esquema generado por el software. En uno de estos casos se utilizó la descripción en modo algorítmico y en el otro a nivel de compuertas.

```
Library IEEE;
Use IEEE.Std_logic_1164.all;

entity FFD is
port(CLK,D,R,H: in std_logic;
    Q: BUFFER std_logic);

end ENTITY;

architecture Behavioral of FFD is
SIGNAL Q0,Q1,Q2: STD_LOGIC;

begin
    PROCESS(CLK,R)
    BEGIN
    IF RISING_EDGE(CLK) THEN
        IF R='1' THEN
        Q0<='0'; Q1<='0'; Q2<='0';
        ELSIF H='1' THEN
            Q0<=D; Q1<=D; Q2<=D;
        ELSE
            Q0<=Q0; Q1<=Q1; Q2<=Q2;
        END IF;
    END IF;
    END PROCESS;
Q<=(Q1 AND Q2)OR
   (Q2 AND Q0)OR
   (Q0 AND Q1);

end Behavioral;
```

Figura 100. TMR - FFD con redundancia modular triple

```
LIBRARY IEEE;
USE IEEE.Std_logic_1164.all;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY FFD_JK_TMR IS
    PORT(J,K,CLK,H,R: IN STD_LOGIC;
    Q: BUFFER STD_LOGIC);
END ENTITY;

ARCHITECTURE Behavioral OF FFD_JK_TMR IS
SIGNAL Q0,Q1,Q2: STD_LOGIC;

COMPONENT FFJK IS
    PORT(J,K,CLK,H,R: IN STD_LOGIC;
    Q: BUFFER STD_LOGIC);
END COMPONENT;

 BEGIN
FFJK0: FFJK PORT MAP(J=>J,K=>K,CLK=>CLK,H=>H,R=>R,Q=>Q0);
FFJK1: FFJK PORT MAP(J=>J,K=>K,CLK=>CLK,H=>H,R=>R,Q=>Q1);
FFJK2: FFJK PORT MAP(J=>J,K=>K,CLK=>CLK,H=>H,R=>R,Q=>Q2);

Q<=(Q1 AND Q2)OR
   (Q2 AND Q0)OR
   (Q0 AND Q1);
END Behavioral;
```

Figura 101.TMR - FFJK con redundancia modular triple

En la Figura 100 se triplica el registro en modo algorítmico y en la Figura 102 se observan los registros y compuertas generados por el mismo software, el resultado es el esperado, la diferencia radica en los multiplexores que aparecen del lado izquierdo ya que estos son los encargados de la habilitación y el reinicio descritos en el programa. En resumen, si se generan los registros y compuertas tal cual se describen.

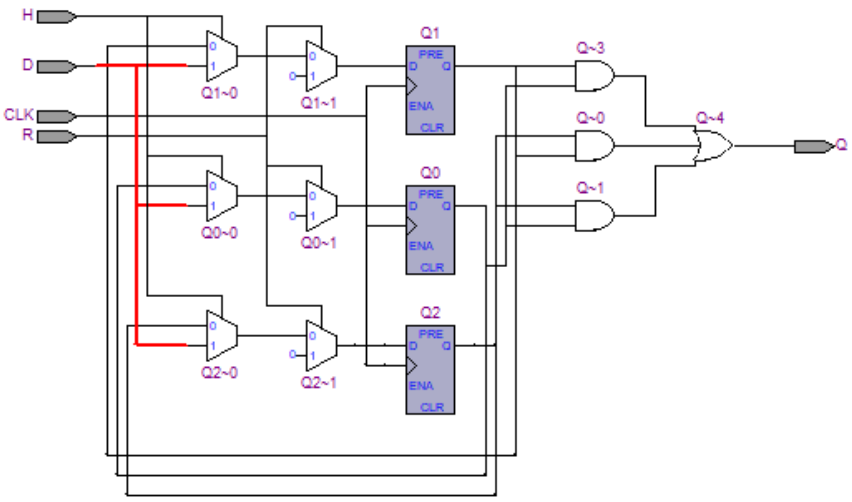


Figura 102. TMR - FFD generado por el software

En el caso del FFJK de la Figura 101 se tiene una descripción diferente en donde se declara el componente del FFJK y se triplica utilizando la función “Port map” para después unir sus salidas e ingresarlas al bloque de votación mayoritaria. En la Figura 103 se muestra en bloques verdes el FF generado y su configuración interna, mientras que fuera de estos bloques se tienen las entradas, salidas y las compuertas que corresponden al voto mayoritario.

En otras palabras, ya sea en modo algorítmico o estructural se tienen los mismos bloques sin cambios en los registros y compuertas. Son los mismos, así que cualquiera de las dos opciones a utilizar para poder implementar el TMR es correcta.

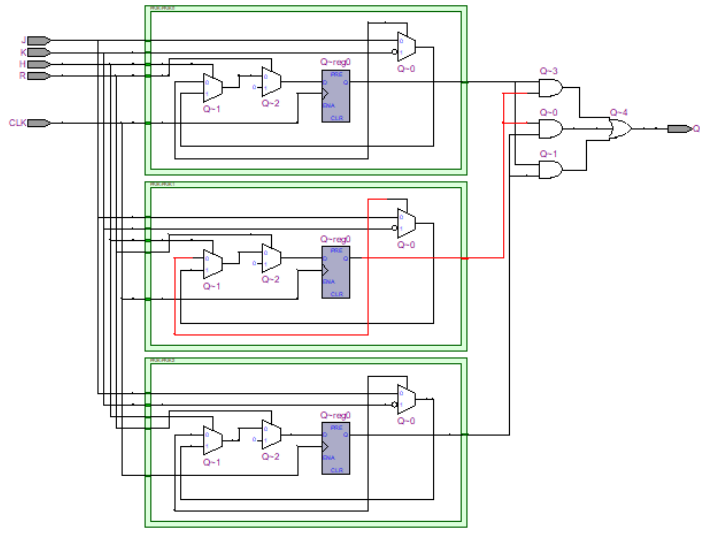


Figura 103. TMR - FFJK generado por el software.

6.12 AXI4 lite

AXI4 es una interfaz de comunicación para conectar diferentes bloques de hardware en un sistema. Esta interfaz es utilizada para facilitar la comunicación entre diversos bloques y/o el procesador. Por ejemplo, se puede conectar periféricos como UART, USB, ADC/DAC, SRAM, entre otros.

Se eligió AXI4 Lite, la cual es una versión reducida de AXI4, ya que es adecuada para aplicaciones como la presentada en este trabajo de tesis ya que los datos que se necesitan transferir se dan de uno por uno con una longitud pequeña. Otro punto que ayudo a la elección de la utilización de este estándar fue gracias a que en el software “VIVADO Design Suite” se puede crear de una manera sencilla una IP con AXI4 Lite integrado y debido a que la descripción de esta interfaz de comunicación ya viene optimizada para solamente agregar la descripción en VHDL y utilizar dicha IP como se requiera.

AXI4 Lite trabaja con direcciones de memoria, es decir, se le debe de indicar la dirección de memoria para poder modificar un dato. Para poder lograr esto es necesario realizar tres etapas

1. La creación de la IP de nuestro módulo SPI: En este punto se interconectan las entradas y salidas deseadas del módulo hacia la interfaz AXI4 Lite.
2. Creación de hardware para utilizar la IP: VIVADO cuenta con un procesador prediseñado llamado MicroBlaze descrito en VHDL, este dará lugar junto con otros componentes al espacio de trabajo donde se integrará la IP.
3. Desarrollo de software: Se utilizará el software VITIS Desing Suite de Xilinx para el desarrollo de software el cual tomará lugar mediante lenguaje de programación C.

Para poder generar la IP AXI4 Lite con nuestro proyecto integrado se utilizó el software VIVADO Design Suite de XILINX. Como se explicó anteriormente, la interfaz SPI se describió en Quartus II versión 9.1 para poder desarrollarla de una manera más rápida. Si este proceso se hubiera realizado en VIVADO hubiera sido al menos 5 veces más tardado debido al sintetizar el proyecto toma de 10 hasta 30 minutos en terminar este proceso y la simulación se tiene que realizar mediante descripción en VHDL lo cual en Quartus II es de manera grafica dando un tiempo mucho menor en cada simulación.

Una vez funcionando correctamente el proyecto en Quartus II, se integrará completamente a la interfaz AXI4 lite en VIVADO. Los siguientes pasos mostrados a continuación se pueden utilizar para poder integrar cualquier proyecto a la interfaz AXI4 Lite, en este caso es para poder integrar nuestro modulo:

La creación de nuestra IP AXI4 Lite con el módulo SPI integrado se muestra en el Anexo 3. También fue necesario la creación de un entorno de trabajo para poder realizar pruebas y obtener resultados, este proceso se encuentra en el Anexo 4.

7. EVALUACION DE RESULTADOS

7.1 Generación de espacio de pruebas.

La IP AXI4 Lite con el módulo SPI integrado esta completada. Ahora, para corroborar los resultados es necesario crear un espacio de trabajo en donde se pueda utilizar dicha IP y visualmente ver los datos que se transmiten y reciben. Para esto se ha diseñado otra IP para poder visualizar su funcionamiento en una tarjeta de trabajo la cual es la Basys 3. Esta nueva IP contiene un decodificador BCD a 7 segmentos y 16 multiplexores 4 a 1 los cuales sirven para que en los displays de 4 segmentos de la tarjeta se observen los valores alfanuméricos en base hexadecimal.

7.2 Pruebas Físicas

Para poder realizar las pruebas se tuvieron dos diseños de bloques, el primero consta de una interconexión de dos módulos SPI dentro de la misma tarjeta como se muestra en la Figura 149. En el segundo se utilizan dos tarjetas de trabajo las cuales contienen una IP de nuestro modulo SPI dentro, una de ellas será configurada en modo maestro y la otra en modo esclavo.

En ambos casos los resultados serán mostrados en la tarjeta de trabajo. En los displays de 7 segmentos, mostrados en rojo en la Figura 104, se visualizarán los datos transmitidos y recibidos tanto del maestro como del esclavo. Los Switchs enmarcados en amarillo serán utilizados para poder elegir el dato que se quiera ver en los displays. Algunos botones que se encuentran enmarcados en blanco tendrán la función de extraer datos de la memoria de recepción tanto del maestro como del esclavo o reiniciar a los valores predeterminados ambos módulos. Por ultimo los puertos Pmod de entrada/salida enmarcados en verde tendrán asociados las señales principales de SPI: SCK, CS, MISO, MOSI.

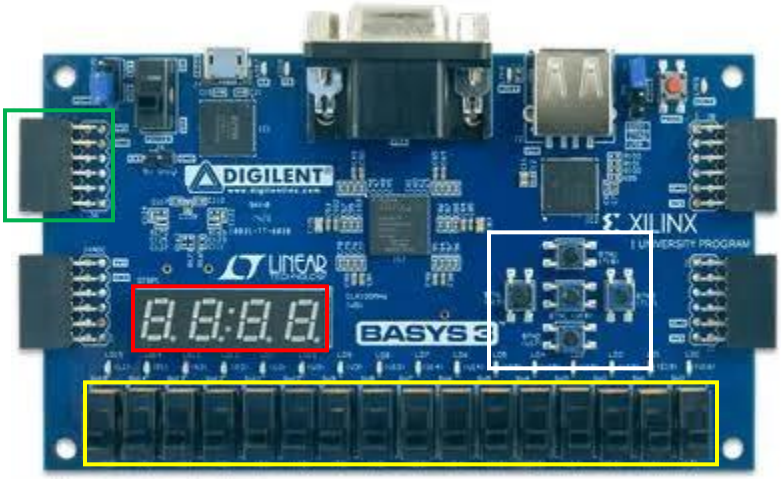


Figura 104. Basys 3

7.2.1 Módulos internamente conectados.

Se realizó este tipo de prueba ya que una transmisión de datos mediante SPI no solo es entre dos dispositivos externos sino también puede ser dentro del mismo. Las pruebas que a continuación se muestran se realizaron variando la frecuencia de operación, las señales CPOL, CPHA y DT.

La máxima frecuencia de transmisión es de 50 MHz la cual se utilizará en este caso para realizar las pruebas. También el maestro recibirá un 0x7910 como dato del esclavo y el esclavo recibía 0x8596 del maestro. Las imágenes siguientes muestran los resultados obtenidos

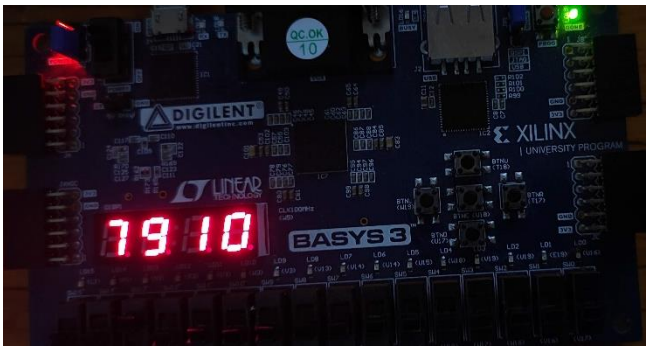


Figura 105. Maestro - Cpol = 0 Cpha = 0, 16 bits.

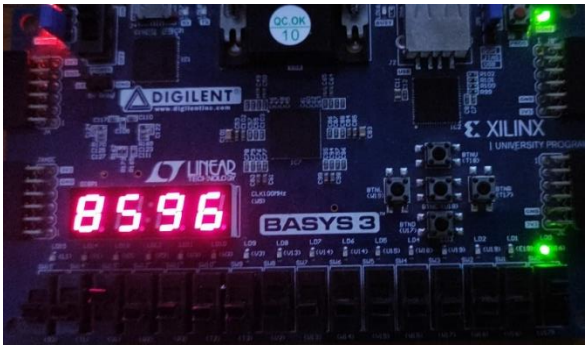


Figura 106. Esclavo - Cpol = 0 Cpha = 0, 16 bits.

Figura 107. Maestro - Cpol = 0 Cpha = 1, 16 bits.

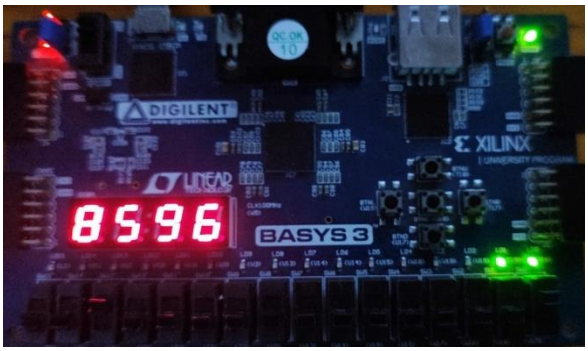


Figura 108. Esclavo - Cpol = 0 Cpha = 1, 16 bits.

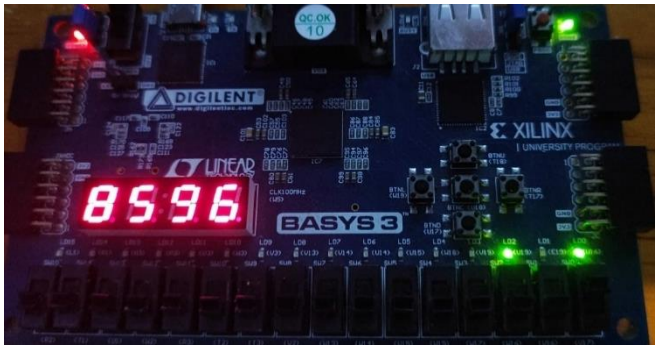


Figura 109. Esclavo - Cpol = 1 Cpha = 0, 16 bits.

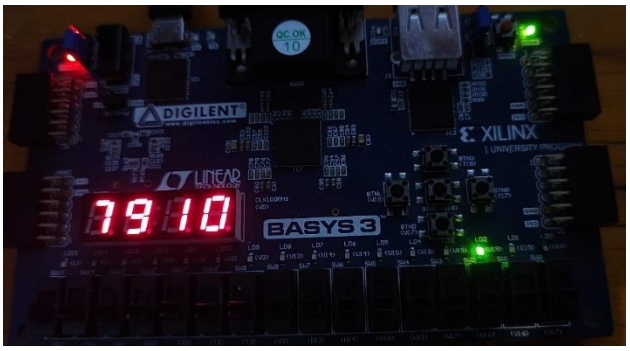


Figura 110. Maestro - Cpol = 1 Cpha = 0, 16 bits.

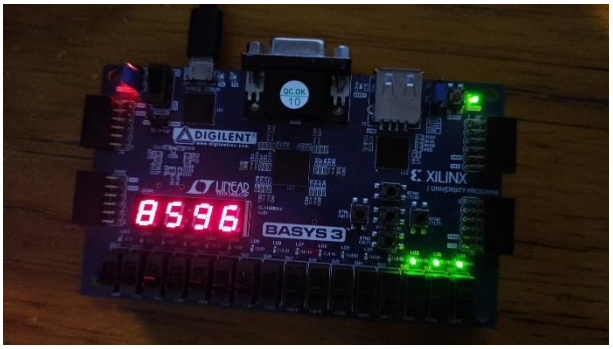


Figura 111. Esclavo - Cpol = 1 Cpha =0, 16 bits.

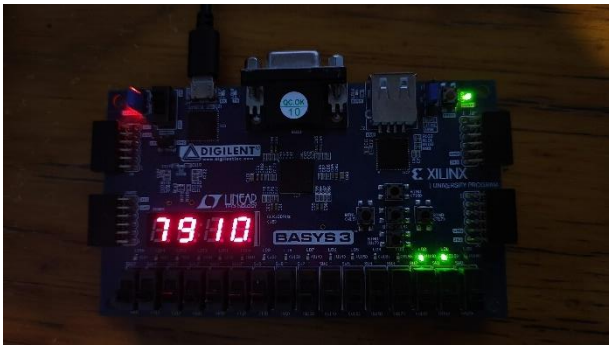


Figura 112. Maestro - Cpol = 1 Cpha = 0, 16 bits.

Las señales CPOL y CPHA se han variado y el funcionamiento ha sido correcto. Ahora, el número de bits que se ha transmitido por dato es de 16, a continuación, se cambiara a diferentes valores en donde el valor original se truncara y solamente se tomaran en cuenta los bits correspondientes menos significativos.

En la Figura 113 del dato original 0x8596 se transmite solamente 0x0016 debido a que se toma en cuenta solo los 5 bits menos significativos.

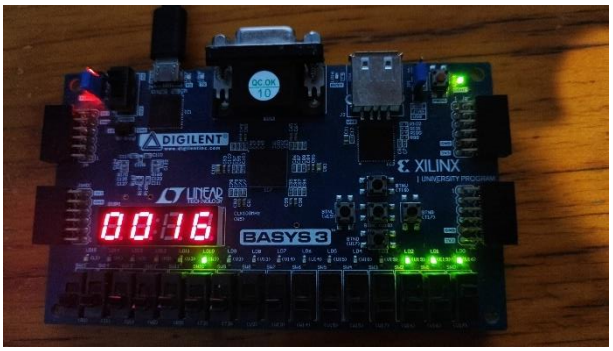


Figura 113. Esclavo - Cpol = 1 Cpha = 1, 5 bits.

De igual manera como en el caso anterior del dato 0x7910 se transmite 0x0010 por la misma razón.

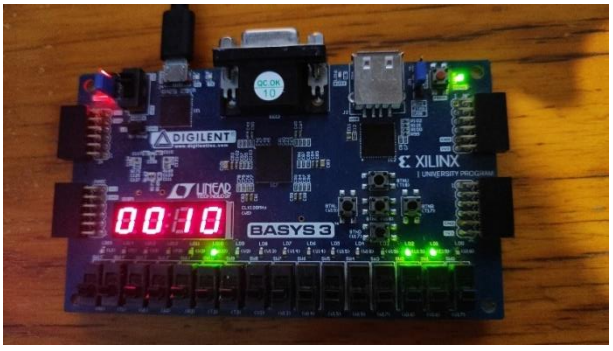


Figura 114. Maestro - Cpol = 1 Cpha = 1, 5 bits.

Los 9 bits menos significativos de 0x8596 son 0x0196, en la Figura 115 se observa que el esclavo recibe un dato correcto ya que se utiliza un valor de DT = 9, es decir el dato tiene una longitud de 9 bits.

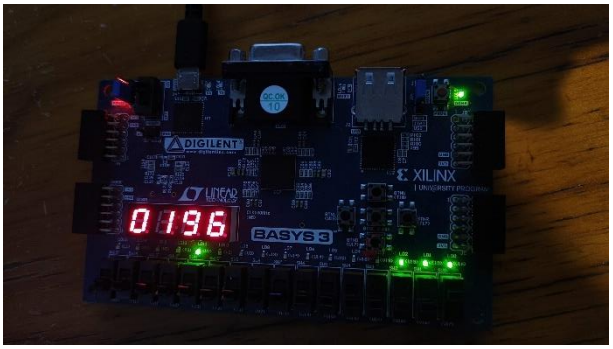


Figura 115. Esclavo - Cpol = 1 Cpha =1, 9 bits.

En la Figura 116 el maestro recibe 0x0110 lo cual es correcto ya que los 9 bits menos significativos de 0x7910 son 0x0110.

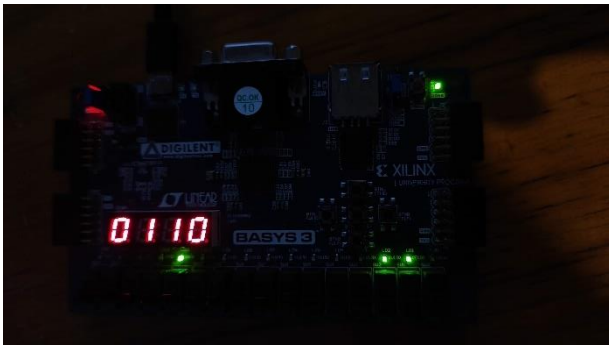


Figura 116. Maestro - Cpol = 1 Cpha = 1, 9 bits.

Los resultados son positivos con una conexión interna en la tarjeta de dos módulos SPI a 50 MHz con diferentes configuraciones tanto de reloj como de datos. Se realizaron pruebas a diferentes frecuencias y los resultados continuaron siendo correctos.

7.2.2 Módulos externamente conectados.

La segunda parte de las pruebas corresponden a una transferencia de datos entre dos FPGA con los mismos componentes que el anterior en cada tarjeta, pero con una sola IP SPI en lugar de dos. El intercambio de datos se dará entre dos tarjetas conectadas externamente, una en modo esclavo y la otra en modo maestro.

La asignación de señales a los puertos Pmod para ambas tarjetas son los siguientes.

Figura 117. Basys 3 - Puertos Pmod.

- JA10 → SCK_M
- JA9 → CS_M
- JA8 → MISO
- JA7 → MOSI
- JA4 → SCK_S
- JA3 → CS_S

Una vez asignados los pines se procede a realizar transferencia de datos entre los dos dispositivos. En este caso los resultados obtenidos tienen una gran diferencia en comparación con el diseño de bloque anterior, esto se debe por el tiempo de propagación porque la distancia recorrida de las señales principales del protocolo SPI es mayor principalmente por la distancia que existe entre tarjetas.

A continuación, se mostrarán los resultados obtenidos:

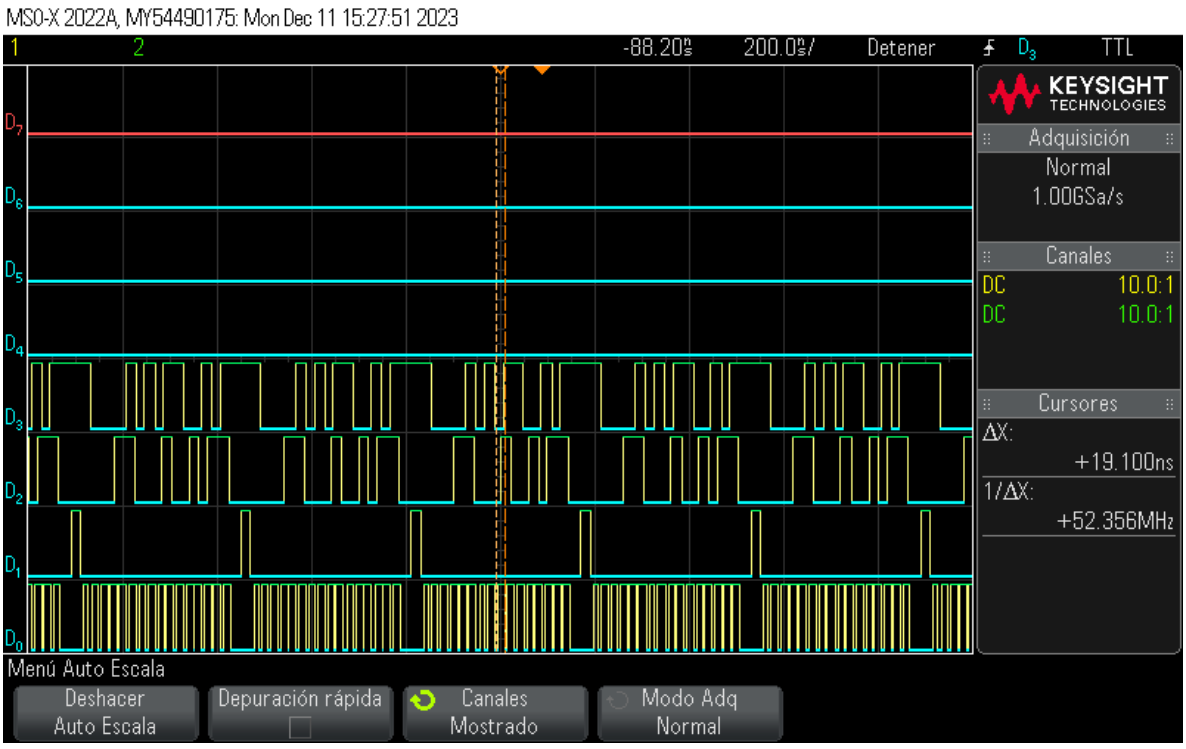


Figura 118. Analizador lógico - Transferencia continua.

En la Figura 118 se observan una transferencia de datos continuos, es decir, una trama tras otra. La línea superior son los bits mandados por el esclavo hacia el maestro, la siguiente línea del maestro al esclavo y las restantes representan al CS y el SCK, respectivamente.

En las siguientes imágenes el reloj SCK presenta una distorsión debido a que el sistema no da tiempo de reaccionar correctamente, es decir, la frecuencia utilizada es más alta de lo que se puede soportar.

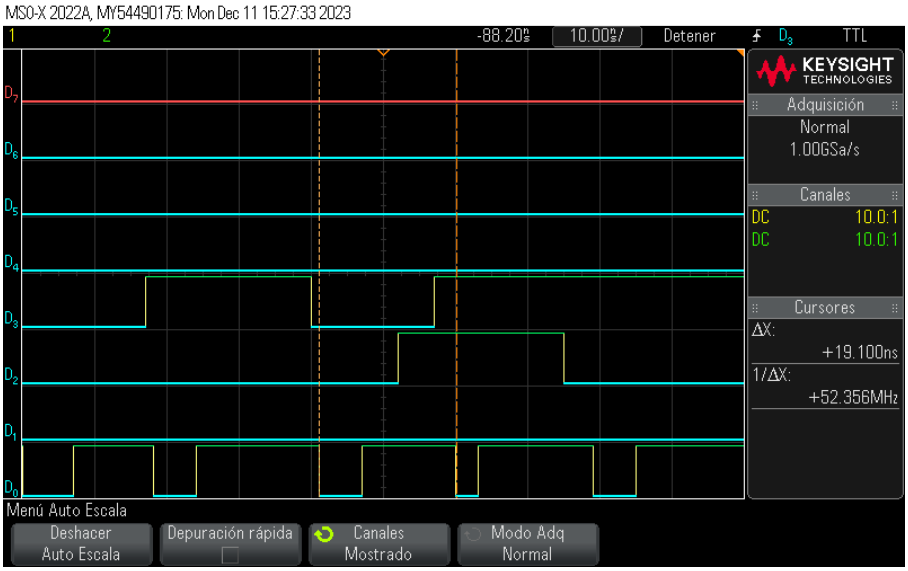


Figura 119. Frecuencia - Cpol = 0, Cpha =1.

En la Figura 119 se mide el tiempo entre flancos de bajada para poder obtener el periodo de señal el cual de espera sea de 20 [ms] dando como resultado una frecuencia de 50 [MHz]. Los resultados reales indican una frecuencia de 52.3 [MHz], muy similar a lo esperado pero el ciclo de trabajo no es del 50 %.

En las siguientes imágenes se muestra el comportamiento que tienen las señales principales de SPI pero en especial la señal de reloj SCK, variando los parámetros CPOL y CPHA. En cada una de ellas el ciclo de trabajo tiene una variación lo cual no es lo ideal.

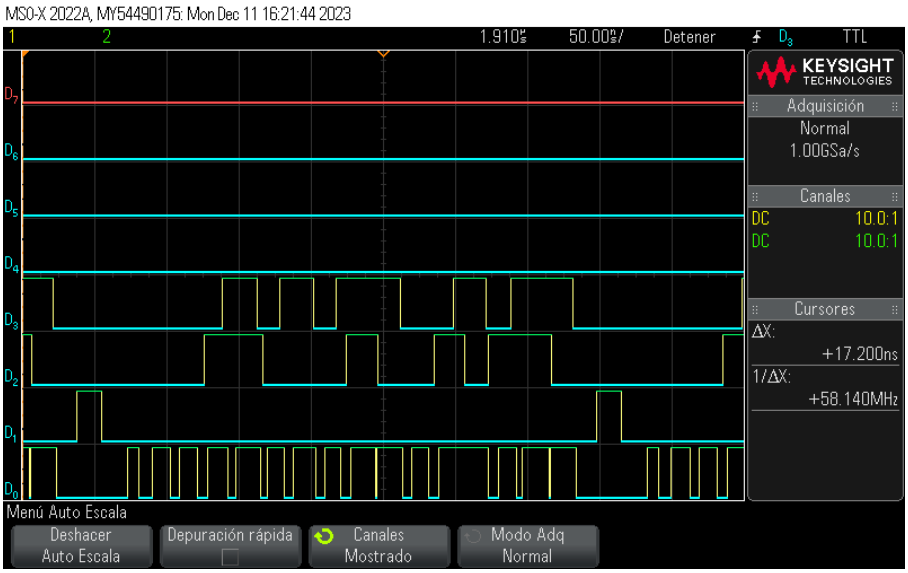


Figura 120. Pruebas - Cpol = 0, Cpha = 0

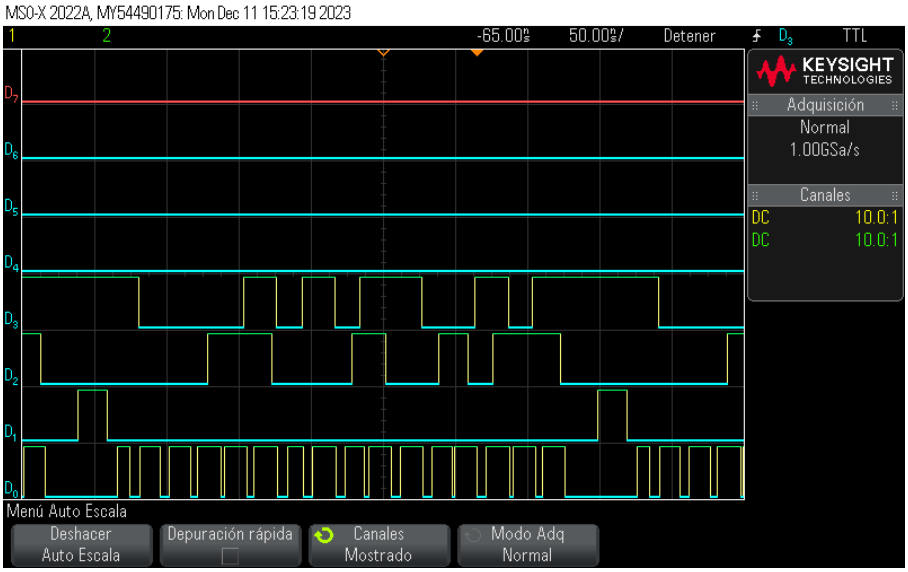


Figura 121. Pruebas - Cpol = 0, Cpha = 1

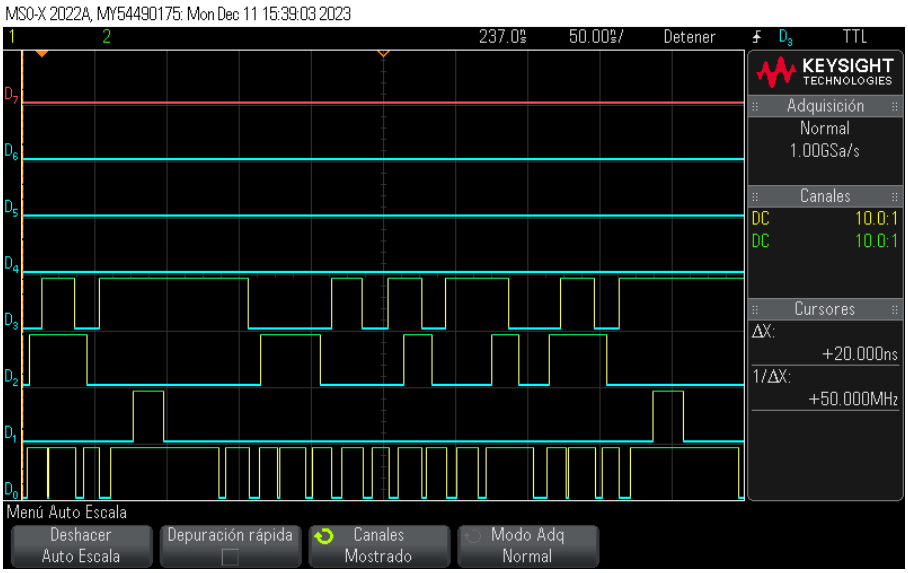


Figura 122. Pruebas - Cpol = 1, Cpha = 0.

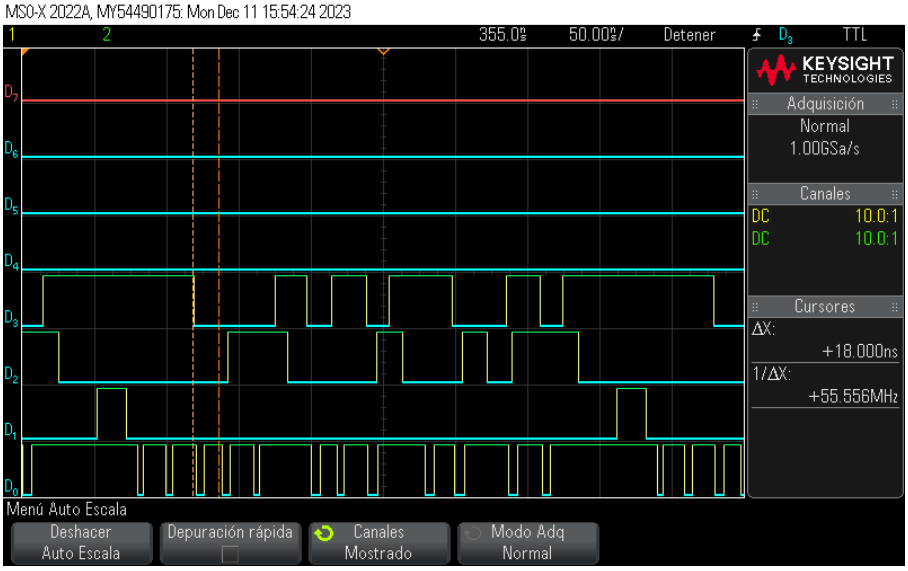


Figura 123. Pruebas - Cpol = 1, Cpha = 1

Después de observar las señales en el analizador lógico se procedió a realizar una transferencia de datos entre dos tarjetas. Esto dio resultados positivos en el caso en donde el CPOL y CPHA son contrarios, es decir, cuando se tiene CPOL = ‘1’ y CPHA = ‘0’ o viceversa, el dato transmitido del maestro al esclavo llega correctamente en cualquiera de los dos casos. En caso contrario, cuando CPOL y CPHA tienen los mismos valores los datos llegan desplazados y se toman como erróneos.

Al momento de tener una frecuencia de operación igual o menor a 25 MHz en el SCK el maestro puede transmitir correctamente datos en cualquiera de las 4 configuraciones del reloj sin ningún problema. Sin embargo, para que el esclavo realice una transmisión correcta es necesaria una frecuencia de 12.5 [MHz] o menor.

En la Figura 124 se muestra una trama de datos a 25 MHz. Tanto el maestro como el esclavo envían el mismo dato, en este caso, se observa que en el canal D2 existe un retraso en comparación al canal D3, en otras palabras, D2 corresponde al dato de que envía el esclavo y existe un retraso en esta señal dando como resultado un error.

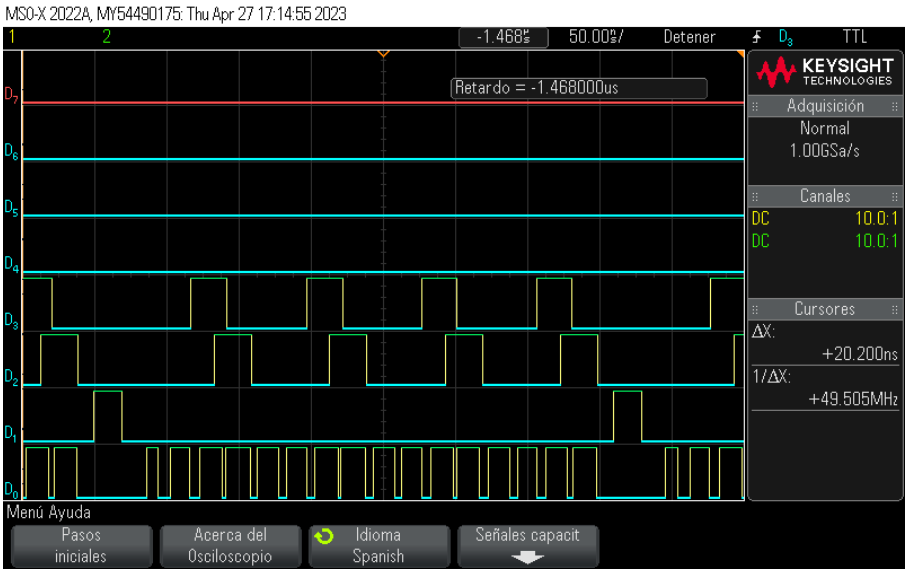


Figura 124. Retraso de tiempo del esclavo.

Tabla 3. Funcionamiento correcto de IP.

	CPOL=0 CPHA=0	CPOL=0 CPHA=1	CPOL=1 CPHA=0	CPOL=1 CPHA=1
50 MHz (M)	×	✓	✓	×
50 MHz (S)	×	×	×	×
25 MHz (M)	✓	✓	✓	✓
25 MHz (S)	×	×	×	×
16.66 MHz (M)	✓	✓	✓	✓
16.66 MHz (S)	×	×	×	×
12.5 MHz (M)	✓	✓	✓	✓
12.5 MHz (S)	✓	✓	✓	✓
10 MHz (M)	✓	✓	✓	✓
10 MHz (S)	✓	✓	✓	✓

En la Tabla 3 se muestra la configuración de reloj en las filas y en las columnas la frecuencia de transferencia de datos tanto del maestro como del esclavo. Con un “×” se representa

una transferencia de datos con malos resultados, a diferencia de "✓" en donde los datos se transfieren de manera correcta y sin errores. Se observa que el modo maestro funciona correctamente a 50 MHz solamente con dos configuraciones de reloj, a una frecuencia menor no tiene errores. En modo esclavo la máxima frecuencia de transferencia de datos sin errores es de 12.5 MHz, esto debido a los tiempos de propagación de las señales en el módulo esclavo, en especial la del SCK, por lo que el retraso aumenta en comparación del maestro.

Los resultados finales son muy parecidos con los teóricos del módulo SPI de [3], este caso es muy representativo ya que el diseño SPI de este proyecto se basó en el del microcontrolador TIVA TM4C1294 debido a las características que tiene como son la frecuencia máxima de operación, las memorias FIFO, entre otros.

8. CONCLUSIONES Y TRABAJO A FUTURO

El propósito de esta tesis fue el desarrollo de una IP SPI integrada en AXI4 Lite con técnicas de tolerancia a fallas descrita en VHDL utilizando solamente funciones booleanas. Se consiguió describir por completo el módulo SPI al más bajo nivel, lo cual ayudo con el proceso de agregar el TMR en cada elemento sin dejar atrás la optimización de recursos utilizados. Por otro lado, el conocer el funcionamiento a detalle de todas las señales ayudo a generar la IP con facilidad.

Como en todo proyecto, se presentaron retos en cada fase de desarrollo dando la posibilidad de depurar y tener un sistema cada vez más completo. Uno muy significante fue la utilización de las 3 señales de reloj en diferentes partes de la interfaz, es decir, las señales CLK y el SCK tanto para modo maestro como para modo esclavo, causaron el uso de un mayor número de recursos lógicos y generaron de igual manera una disminución en la velocidad de transferencia.

Otro gran reto fue la implementación de la interfaz AXI4 Lite ya que la información sobre su uso es escasa, por lo que se tuvo que recurrir a un método de pruebas para diseñar la interfaz. Al realizar este proceso y manejar de una mejor manera el software VIVADO fue posible tener un buen resultado. Además, el conocimiento básico de descripción de hardware en VHDL ayudo mucho para la integración del módulo SPI en la interfaz AXI4 Lite.

Las técnicas de tolerancia a fallas fueron aplicadas en diferentes partes de la interfaz SPI, en este caso no fueron probadas físicamente ya que es necesario adentrarse a temas externos de inyección de fallas. Para probar las técnicas de tolerancia e tuvo que cambiar alguna compuerta dentro de dicho bloque haciendo referencia a una posible falla, esto dio resultados positivos.

Después de resolver todos los problemas presentados durante el desarrollo de esta IP y tener resultados que cumplen con los requerimientos establecidos los cuales se comprobaron primero en las simulaciones, y posteriormente se realizaron las pruebas físicas correspondientes. El intercambio de datos con el SPI desarrollado a una frecuencia baja no tiene ningún problema y funciona correctamente sin errores ni desplazamientos inesperados, el problema viene cuando se llega a la máxima frecuencia, 50 MHz, ya que los tiempos de propagación generan que el intercambio de datos en 2 de las 4 configuraciones de reloj tenga desplazamientos indeseables.

Trabajo Futuro

Queda resolver los problemas ocurridos a altas velocidades, provocados por los tiempos de propagación en el hardware descrito, mediante el análisis y optimización del diseño.

9. ANEXOS

Anexo 1. Estilos de descripción en VHDL.

En la primera parte del programa donde se declaran todas las entradas y salidas a utilizar, esto omite las señales internas del circuito entre componentes.

Por ejemplo:

```
ENTITY Mux2a1 IS
PORT(a: IN BIT;
        b: IN BIT;
        sel: IN BIT;
        c: OUT BIT);
END Mux2a1;
```

La segunda parte del programa ligada a la entidad donde se describe el comportamiento o los componentes con sus respectivas interconexiones, esta es llamada Arquitectura la cual da lugar a cualquier modo de descripción antes mencionado e incluso una combinación entre ellos. Para ser más claro sobre las diferencias y similitudes se dará un ejemplo para cada uno a partir de la entidad de un multiplexor 2 a 1.

En el modo estructural la sintaxis no es tan clara como se logra ver a continuación

```
ARCHITECTURE Estructural OF Mux2a1 IS
COMPONENT comp_not IS
        PORT(a: IN BIT;
                b: OUT BIT);
END COMPONENT;

COMPONENT comp_and IS
        PORT(a: IN BIT;
                b: IN BIT;
                c: OUT BIT);
END COMPONENT;

COMPONENT comp_or IS
        PORT(a: IN BIT;
                b: IN BIT;
                c: OUT BIT);
END COMPONENT;

SIGNAL x,y,z: bit;

BEGIN
```

```
Not1: not PORT MAP(sel,x);
And1: and PORT MAP(b,sel,y);
And2: and PORT MAP(a,x,z);
Or1: or PORT MAP(y,z,c);

END ARCHITECTURE;
```

Se diseñan las compuertas a utilizar como son AND, OR y NOT, esas mismas compuertas se utilizan para describir el multiplexor 2 a 1. Mediante la leyenda PORT MAP se manda a llamar la compuerta deseada y dentro del paréntesis se asocian las entradas con sus respectivas salidas. Este modo es poco utilizado debido a su complejidad y larga codificación, cada compuerta a utilizar ocupa mucho espacio hablando de código haciendo demasiado extenso un programa más complicado que un mux 2 a 1.

El modo “flujo de datos” se puede utilizar de dos maneras como se muestra a continuación.

```
ARCHITECTURE Flujo_De_Datos OF Mux2a1 IS

BEGIN
        c<=(a and (not sel)) or (b and sel); (Flujo de datos en moto estructural)

        c <= a WHEN sel ‘0’ ELSE b; (Flujo de datos con sentencias WHEN-ELSE)

END ARCHITECTURE;
```

En la arquitectura de este estilo de descripción se puede generar un multiplexor 2 a 1 en tan solo una línea de código, a comparación del estructural que es más extenso y difícil de entender.

Por último, el modo “Algorítmico” se centra en el comportamiento que se desea tener y no en los componentes, por lo mismo para circuitos pequeños puede parecer más complicada la descripción, sin embargo, para circuitos más grandes ayuda a disminuir el trabajo necesario para el comportamiento requerido.

```
ARCHITECTURE Algorítmico OF Mux2a1 IS

BEGIN
        PROCESS(a, b, sel)

BEGIN
        IF (sel=’1’) THEN

        c<=b;
ELSE
        c<=a;

END IF;

END PROCESS;

END ARCHITECTURE;
```

En este estilo de descripción no se utilizan compuertas, solamente se describe el comportamiento que tendrán las salidas con respecto a las entradas ya que el software utilizado se encarga de crear la estructura interna para dicho funcionamiento. En programas de mayor magnitud es necesario incluir señales internas, pero el estilo de descripción tiene la misma estructura.

Anexo 2. Tablas de estados para diseño de contadores.

Tabla 4. Tabla del estado siguiente - Contador 2

	Q3	Q2	Q1	Q0	QSCK2	Q3+	Q2+	Q1+	Q0+	J3	K3	J2	K2	J1	K1	J0	K0
0									1		X		X		X	1	X
1					1						X		X		X		X
2				1				1			X		X	1	X	X	1
3				1	1						X		X		X	X	1
4			1					1	1		X		X	X		1	X
5			1		1				1		X		X	X	1	1	X
6			1	1			1				X	1	X	X	1	X	1
7			1	1	1			1			X		X	X		X	1
8		1					1		1		X	X			X	1	X
9		1			1			1	1		X	X	1	1	X	1	X
10		1		1			1	1			X	X		1	X	X	1
11		1		1	1		1				X	X			X	X	1
12		1	1				1	1	1		X	X		X		1	X
13		1	1		1		1		1		X	X		X	1	1	X
14		1	1	1		1				1	X	X	1	X	1	X	1
15		1	1	1	1		1	1			X	X		X		X	1
16	1					1				X			X		X		X
17	1				1		1	1	1	X	1	1	X	1	X	1	X

Tabla 5. Tabla del estado siguiente - Contador 3

	Q3	Q2	Q1	Q0	QSCK2	Q3+	Q2+	Q1+	Q0+	J3	K3	J2	K2	J1	K1	J0	K0
0									1		X		X		X	1	X
1					1						X		X		X		X
2				1				1			X		X	1	X	X	1
3				1	1						X		X		X	X	1
4								1	1		X		X	X		1	X
5			1		1				1		X		X	X	1	1	X
6			1	1			1				X	1	X	X	1	X	1
7			1	1	1			1			X		X	X		X	1
8		1					1		1		X	X			X	1	X
9		1			1			1	1		X	X	1	1	X	1	X
10		1		1			1	1			X	X		1	X	X	1
11		1		1	1		1				X	X			X	X	1
12		1	1				1	1	1		X	X		X		1	X
13		1	1		1		1		1		X	X		X	1	1	X
14		1	1	1		1				1	X	X	1	X	1	X	1
15		1	1	1	1		1	1			X	X		X		X	1
16	1					1				X			X		X		X
17	1				1		1	1	1	X	1	1	X	1	X	1	X

Tabla 6. Tabla del estado siguiente - Contador 4

	R	Q4	Q3	Q2	Q1	Q0	Q4+	Q3+	Q2+	Q1+	Q0+	D4	D3	D2	D1	D0
0											1					1
1						1				1					1	
2					1					1	1				1	1
3					1	1			1					1		
4				1					1		1			1		1
5				1		1			1	1				1	1	
6				1	1				1	1	1			1	1	1
7				1	1	1		1					1			
8			1					1			1		1			1
9			1			1		1		1			1		1	
10			1		1			1		1	1		1		1	1
11			1		1	1		1	1				1	1		
12			1	1				1	1		1		1	1		1
13			1	1		1		1	1	1			1	1	1	
14			1	1	1			1	1	1	1		1	1	1	1
15			1	1	1	1	1					1				
16		1					1				1	1				1
17		1				1	1			1		1			1	
18		1			1		1			1	1	1			1	1
19		1			1	1	1		1			1		1		
20		1		1			1		1		1	1		1		1
21		1		1		1	1		1	1		1		1	1	
22		1		1	1		1		1	1	1	1		1	1	1
23		1		1	1	1	1	1				1	1			
24		1	1				1	1			1	1				1
25		1	1			1	1	1		1		1	1		1	
26		1	1		1		1	1		1	1	1	1		1	1
27		1	1		1	1	1	1	1			1	1	1		
28		1	1	1			1	1	1		1	1	1	1		1
29		1	1	1		1	1	1	1	1		1	1	1	1	
30		1	1	1	1		1	1	1	1	1	1	1	1	1	1
31		1	1	1	1	1										
32	1															
33	1					1										
34	1				1											
35	1				1	1										
36	1			1												
37	1			1		1										
38	1			1	1											
39	1			1	1	1										
40	1		1													
41	1		1			1										
42	1		1		1											
43	1		1		1	1										
44	1		1	1												
45	1		1	1		1										
46	1		1	1	1											
47	1		1	1	1	1										
48	1	1														
49	1	1				1										
50	1	1			1											
51	1	1			1	1										
52	1	1		1												
53	1	1		1		1										
54	1	1		1	1											
55	1	1		1	1	1										
56	1	1	1													
57	1	1	1			1										
58	1	1	1		1											
59	1	1	1		1	1										
60	1	1	1	1												
61	1	1	1	1		1										
62	1	1	1	1	1											
63	1	1	1	1	1	1										

Tabla 7. Tabla del estado siguiente - Contador 5

	UNO	DSCK	CS	A	B	A+	B+
0							
1					1	1	
2				1		1	
3				1	1	1	
4			1				
5			1		1		
6			1	1			
7			1	1	1		
8		1					
9		1			1	1	
10		1		1		1	
11		1		1	1	1	
12		1	1				
13		1	1		1		
14		1	1	1			
15		1	1	1	1		
16	1						
17	1				1	1	
18	1			1		1	
19	1			1	1	1	
20	1		1				
21	1		1		1		
22	1		1	1			
23	1		1	1	1		
24	1	1					1
25	1	1			1	1	
26	1	1		1		1	
27	1	1		1	1	1	
28	1	1	1				
29	1	1	1		1		
30	1	1	1	1			
31	1	1	1	1	1		

Tabla 8. Tabla de estado siguiente - Contador 6

			D1	D0
A	B	EXT	A+	B+
		1		1
	1			
	1	1	1	
1				
1		1	1	
1	1			
1	1	1	1	

Tabla 2. Tabla de estado siguiente - Contador 7

A	B	CS	EXT	A	B
			1		
		1			1
		1	1		1
	1				
	1		1		
	1	1		1	
	1	1	1		1
1					
1			1		
1		1		1	
1		1	1	1	
1	1				
1	1		1		
1	1	1		1	
1	1	1	1	1	

Anexo 3. Creación de una IP AXI4 Lite.

i. Se selecciona “Create Project”

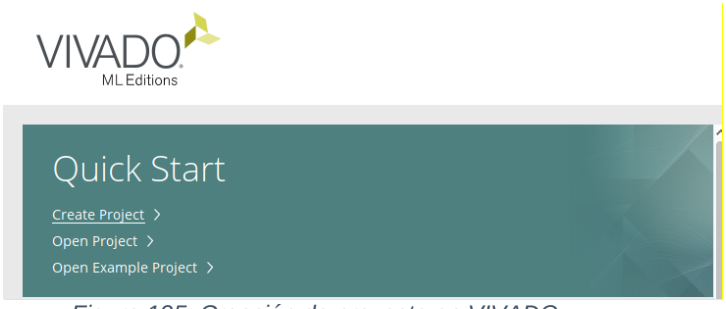


Figura 125. Creación de proyecto en VIVADO

ii. En la ventana que será desplegada seleccionaremos “Next” y posteriormente se visualizará la ventana de la Figura 126 en donde se nombrará al proyecto y seleccionará en donde almacenarlo. Una vez listo se procede a dar clic de nuevo en “Next”.

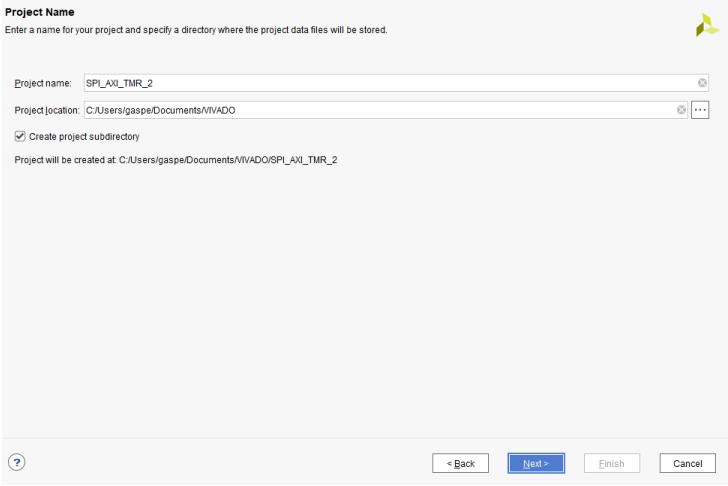


Figura 126. Nombrar al proyecto en VIVADO

En los siguientes 3 pasos no se mostrarán capturas de pantalla ya que no se les hace ningún cambio.

iii. En la siguiente vista se tendrá seleccionado por defecto “RTL Project” el cual es el que utilizaremos, es la única opción que debe de estar seleccionada y posteriormente se da clic en “Next”.

iv. En la vista “add sources” no es necesario añadir los archivos en VHDL que se crearon para el proyecto, en este paso solamente se da clic en “Next”

v. En la vista “Add constrains (optionals)” se puede agregar un archivo con extensión. XDC para la asignación de pines o solamente dar clic en “Next”.

vi. La siguiente vista en la ventana es “Default Part” en donde se seleccionará la tarjeta de trabajo utilizada para poder realizar las pruebas reales de la interfaz y se dará clic en “Next”. En la Figura 127 se observa un ejemplo de la selección de la tarjeta “BASYS 3” la cual será utilizada en este caso.

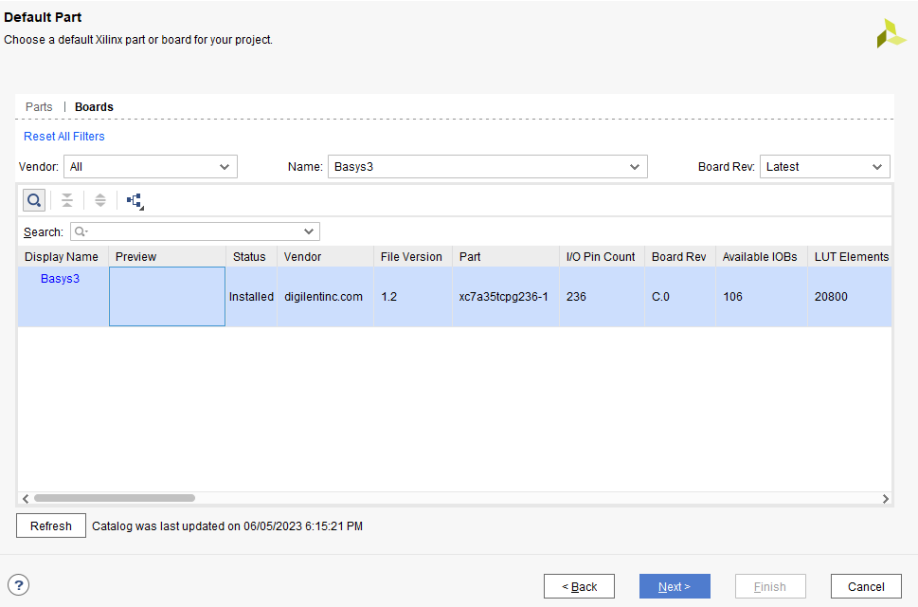


Figura 127. Selección de tarjeta de trabajo

vii. Por último, se mostrará un resumen de los pasos realizados anteriormente y se selecciona “Finish”. Se abrirá una nueva ventana el cual es el espacio de trabajo ya que el proyecto ha sido creado.

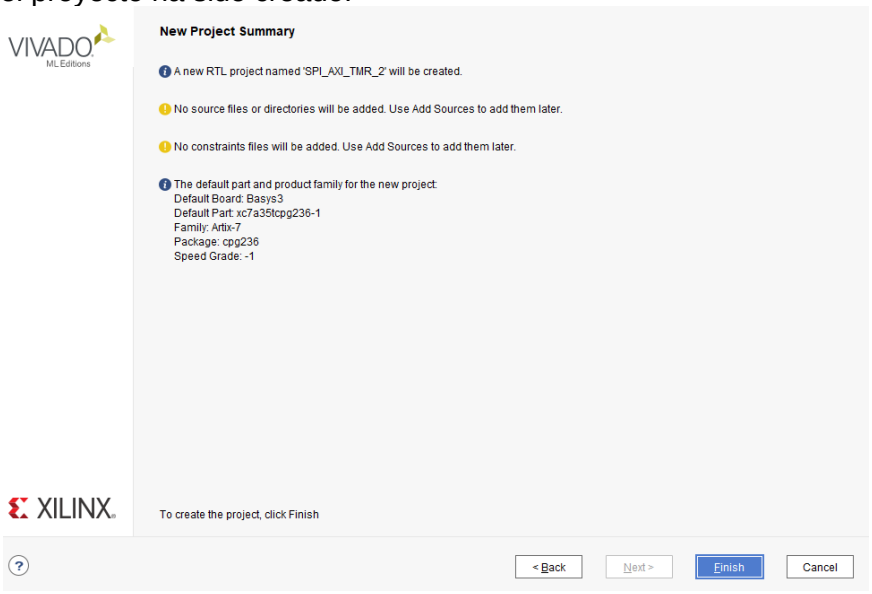


Figura 128. Resumen de creación del proyecto.

Una vez creado el proyecto en VIVADO se procede a crear una IP AXI lite como se logra ver a continuación:

i. Seleccionamos “Tools - Create and Package New IP”. Esto nos desplegará una ventana nueva.

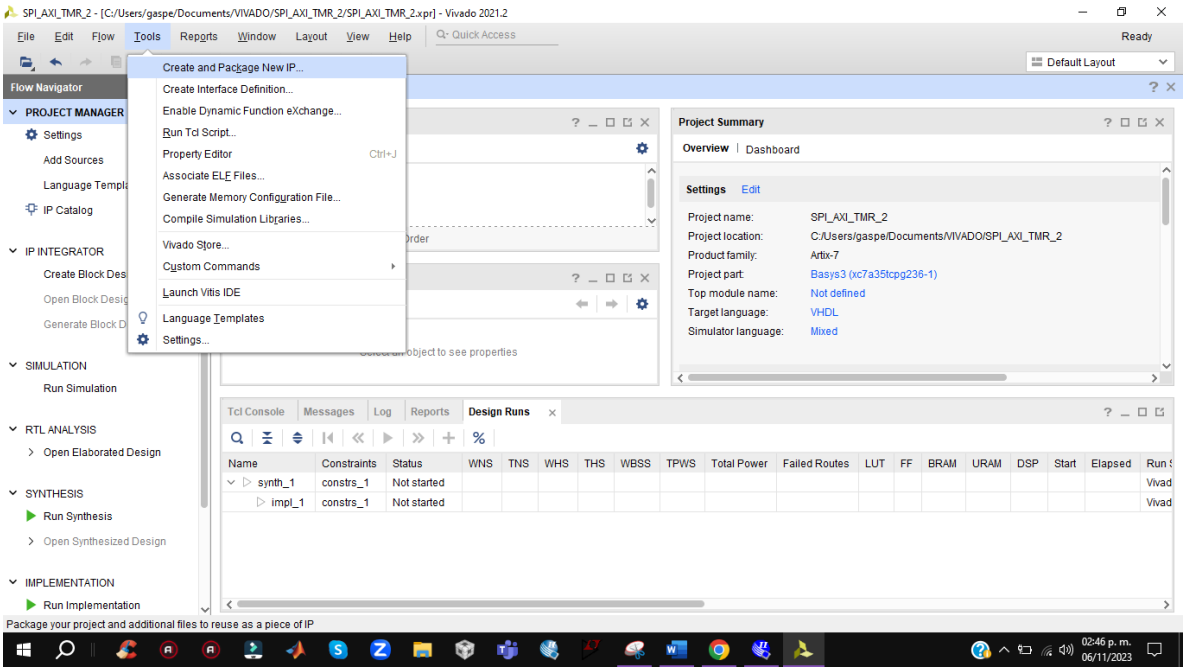


Figura 129. Espacio de trabajo en VIVADO

ii. En la nueva ventana se seleccionará “Next” y se nos mostrará el contenido de la Figura 130. Seleccionaremos “Create a new AXI4 Peripheral” y seleccionamos “Next”.

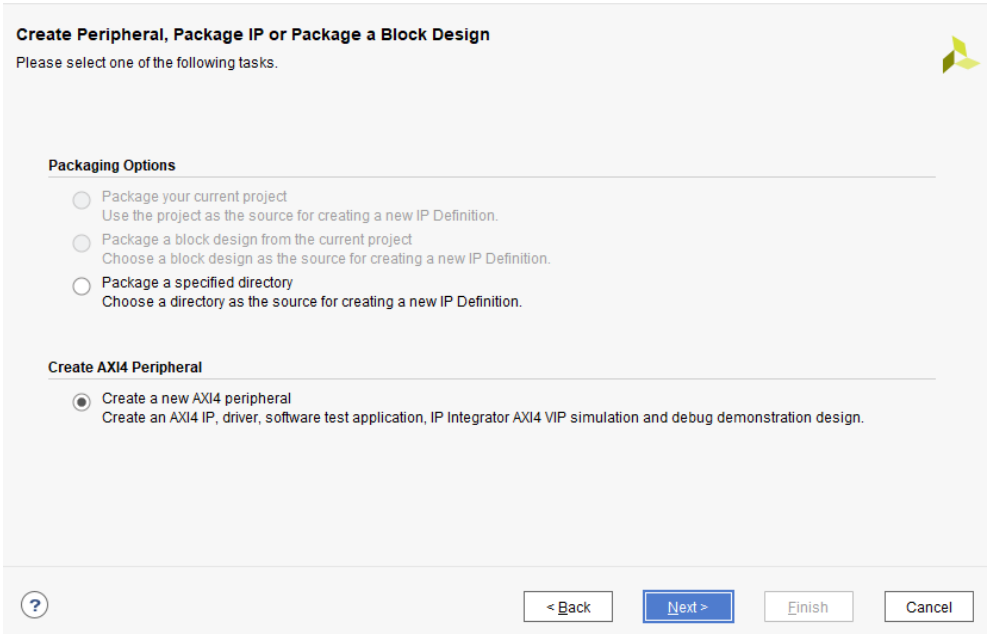


Figura 130. Creación de IP AXI.

iii. En la siguiente vista se nombra a la IP y se puede o no describir brevemente su función o algún comentario formal o informal.

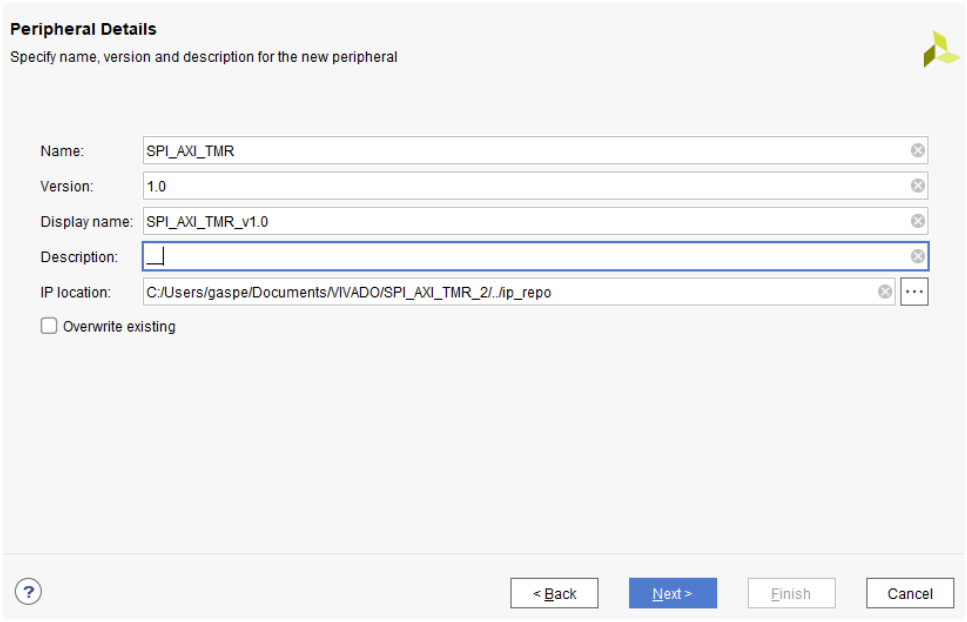


Figura 131. Nombre y descripción de la IP.

iv. Después se muestra la configuración de la IP donde se pueden cambiar parámetros muy importantes los cuales en este caso no se modificarán ya que los que vienen por defectos son los que necesitamos, seleccionamos “Next”. En la Figura 132 tenemos seleccionados AXI Lite y la utilización de 4 registros.

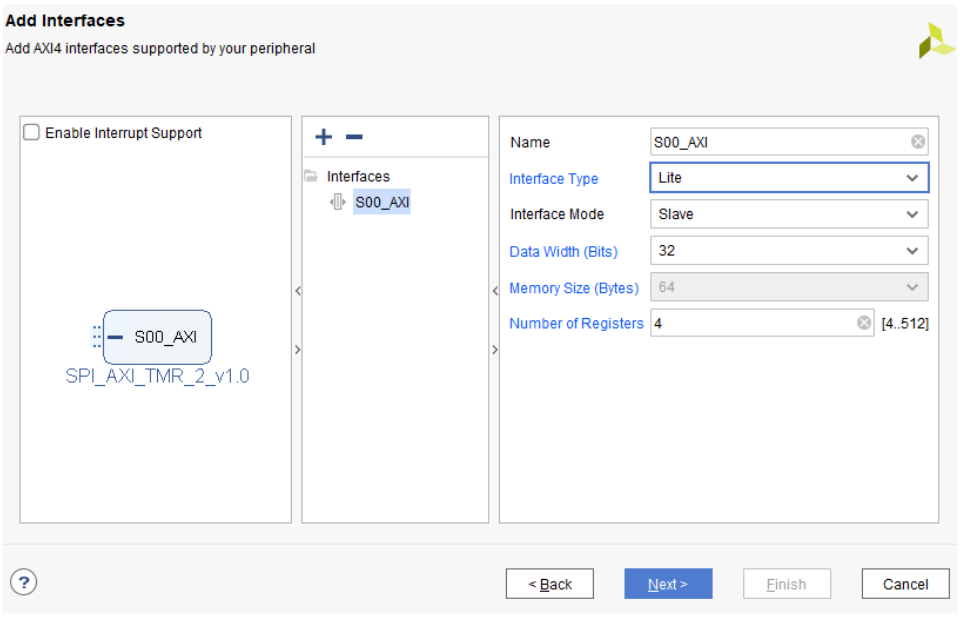


Figura 132. Parámetros de la IP

v. Por último, seleccionamos “Edit IP” en la última vista y seleccionamos “Finish”.

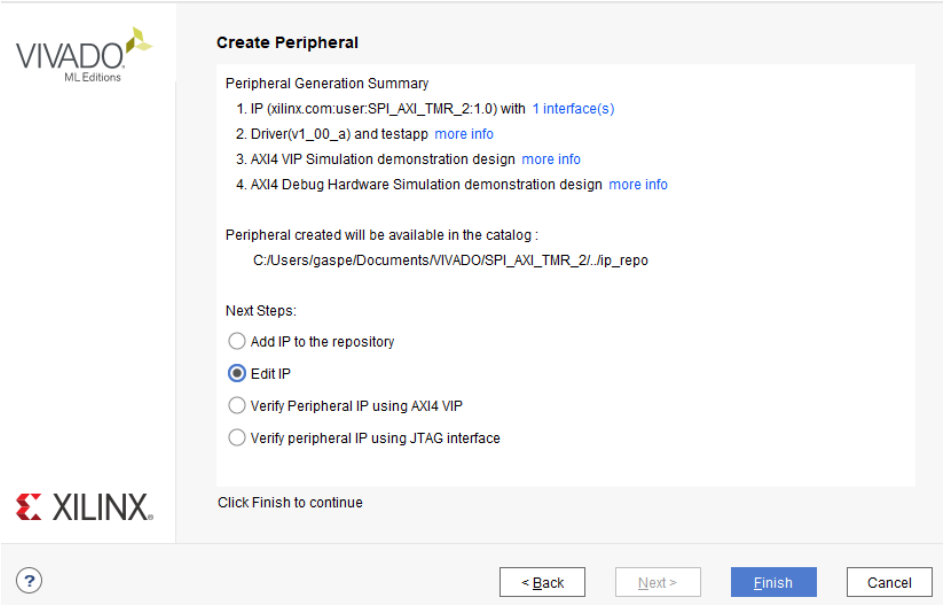


Figura 133. AXI4 lite - Edit IP

vi. Una vez creada la IP AXI4 Lite se desplegará una nueva ventana muy parecida a la Figura 129 en donde se tendrán dos archivos en VHDL los cuales son los archivos para poder agregar la descripción de la interfaz SPI.

A continuación, se mostrarán los pasos para poder generar la IP AXI4 con SPI integrado:

i. Se tienen 2 archivos en VHDL, el primero es el programa principal en donde se vinculan las entradas y salidas de AXI4 Lite con el proyecto en general. El segundo archivo es la descripción en VHDL de la interfaz AXI4. En la Figura 134 se muestran dichos archivos.

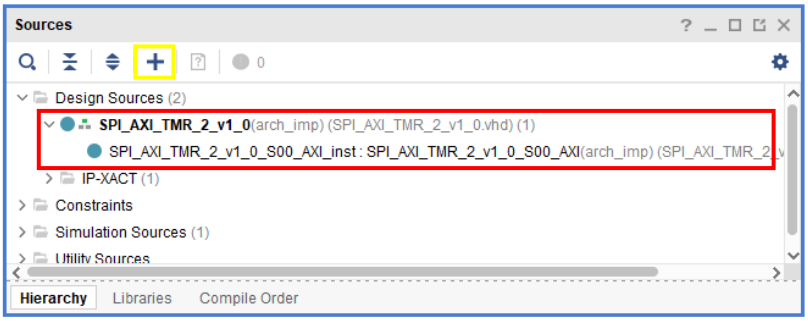


Figura 134. Archivos de AXI4 Lite

ii. El archivo principal de la interfaz SPI_AXI_TMR_2_v1_0 contiene en la entidad las entradas y salidas del proyecto en VHDL creado en VIVADO y la conexión con las entradas y salidas de la interfaz AXI4 mediante señales. En este segundo archivo de AXI4 “SPI_AXI_TMR_2_v1_0_S00_inst…” es en donde se tienen las líneas de descripción de la interfaz AXI4 Lite y también en donde se agregarán las entradas, salidas, señales, componentes y descripción en VHDL del módulo SPI en su correspondiente sección.

iii. Dentro del archivo “SPI_AXI_TMR_2_v1_0_S00_inst…” se tiene la primera sección en donde las entradas y salidas del módulo SPI se deben de escribir.

```
entity SPI_AXI_TMR_2_v1_0_S00_AXI is
    generic (
        -- Users to add parameters here

        -- User parameters ends
        -- Do not modify the parameters beyond this line

        -- Width of S_AXI data bus
        C_S_AXI_DATA_WIDTH  : integer    := 32;
        -- Width of S_AXI address bus
        C_S_AXI_ADDR_WIDTH  : integer    := 4
    );
    port (
        -- Users to add ports here

        -- User ports ends
        -- Do not modify the ports beyond this line

        -- Global Clock Signal
        S_AXI_ACLK  : in std_logic;
        -- Global Reset Signal. This Signal is Active LOW
```

Figura 135. Sección para agregar entradas y salidas del módulo SPI a AXI4 Lite.

iv. En la Figura 136 se muestra el lugar en donde las señales del módulo SPI tienen lugar.

```
    signal aw_en    : std_logic;

-- Users to add signals and components here

        -- User signal and components ends


begin
    -- I/O Connections assignments
```

Figura 136. Sección para agregar señales del módulo SPI a AXI4 Lite

v. Por último, en la Figura 137 se observa el espacio en donde se agregará toda la descripción en VHDL del módulo SPI, es decir, lo descrito en la Arquitectura del proyecto de colocará en esta parte.

```
    end process;


    -- Add user logic here

    -- User logic ends

end arch_imp;
```

Figura 137. Sección para agregar descripción en VHDL de SPI a AXI4 Lite

vi. Una parte importante para poder tener conexión con la interfaz AXI4 Lite son los registros de lectura y escritura los cuales se pueden observar en la Figura 138.

```
494     process (slv_reg0, slv_reg1, slv_reg2, slv_reg3, axi_araddr, S_AXI_ARESETN, slv_reg_rden)
495     variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
496     begin
497         -- Address decoding for reading registers
498         loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
499         case loc_addr is
500           when b"00" =>
501             reg_data_out <= slv_reg0;
502           when b"01" =>
503             reg_data_out <= slv_reg1;
504           when b"10" =>
505             reg_data_out <= DR_RX;
506           when b"11" =>
507             reg_data_out <= FLAGS;
508           when others =>
509             reg_data_out  <= (others => '0');
510         end case;
511     end process;
```

Figura 138. Asociación de SPI con registros de AXI4 Lite

En esta imagen se observan el bus “slv_reg0” y “slv_reg1” en azul, las cuales son las señales que tendrán el papel de ligar la interfaz AXI4 Lite con el módulo SPI y poder escribir en ellas. Mientras que “DR_RX” y “FLAGS” tienen el mismo papel, pero estas se utilizan para leer datos y monitorear banderas. En este caso DR_RX Y FLAGS corresponde al módulo SPI y se liga a la interfaz AXI4, mientras que slv_reg0 y slv_reg1 corresponde a la interfaz AXI4 y se utilizara como entradas al módulo SPI.

vii. Ahora el bus SLV_REG0 tomo el lugar de la señal CTRL para poder escribir en el módulo SPI mediante la interfaz AXI4 Lite. Para esto se utilizará otro software una vez terminado la IP.

```
558 ------------REGISTROS DE CONTROL--------------------------------
559 MS_C:      FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(0),  R=>R, Q=>MS        ,H=>CS_MS);
560 CPHA_C:    FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(1),  R=>R, Q=>CPHA      ,H=>CS_MS);
561 CPOL_C:    FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(2),  R=>R, Q=>CPOL      ,H=>CS_MS);
562 CICLOS_0:  FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(3),  R=>R, Q=>CICLOS(0) ,H=>CS_MS);
563 CICLOS_1:  FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(4),  R=>R, Q=>CICLOS(1) ,H=>CS_MS);
564 CICLOS_2:  FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(5),  R=>R, Q=>CICLOS(2) ,H=>CS_MS);
565 CICLOS_3:  FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(6),  R=>R, Q=>CICLOS(3) ,H=>CS_MS);
566 CICLOS_4:  FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(7),  R=>R, Q=>CICLOS(4) ,H=>CS_MS);
567 DT_0:      FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(8),  R=>R, Q=>DT(0)     ,H=>CS_MS);
568 DT_1:      FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(9),  R=>R, Q=>DT(1)     ,H=>CS_MS);
569 DT_2:      FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(10), R=>R, Q=>DT(2)     ,H=>CS_MS);
570 DT_3:      FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(11), R=>R, Q=>DT(3)     ,H=>CS_MS);
571 S_CS0:     FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(12), R=>R, Q=>S_CS(0)   ,H=>CS_MS);
572 S_CS1:     FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(13), R=>R, Q=>S_CS(1)   ,H=>CS_MS);
573 ----PRUEBA DATOS DE ENTRADA-------------------------------
574 SW_14 :    FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(14), R=>R, Q=>SW14      ,H=>CS_MS);
575 SW_15 :    FFD PORT MAP(CLK=>CLK , D=>SLV_REG0(15), R=>R, Q=>SW15      ,H=>CS_MS);
```

Figura 139. AXI4 Lite - Registros de control

viii. En la Figura 140 se muestra la modificación del módulo IP en donde se hace la conexión directa del módulo SPI con la interfaz AXI4 Lite.

```
874  --REGISTROS DE ESCRITURA TX FIFO
875  TX_DR: REG_16 PORT MAP(CLK=>CLK, I=>SLV_REG1(15 DOWNTO 0),R=>R,O=>QTDR,H=>H_TDR);
876
877  --REGITROS DE LECTURA RX FIFO
878  RX_DR: REG_16 PORT MAP(CLK=>CLK, I=>RDR, R=>R  ,O=>W_DR, H=>H_RDR);
879
880  --ASOCIACION DE BANDERAS Y SALIDA DE DATOS RX A REGISTROS DE AXI4 Lite.
881  DR_RX(15 DOWNTO 0)<=W_DR;
882  FLAGS(0)<=EMPTY_TX;
883  FLAGS(1)<=FULL_TX;
884  FLAGS(2)<=EMPTY_RX;
885  FLAGS(3)<=FULL_RX;
```

Figura 140. AXI4 Lite - Registros de escritura, lectura y banderas.

ix. En el archivo “SPI_AXI_TMR_2_v1_0” se agregan las mismas entradas y salidas tanto para la entidad como en el componente, en las siguientes imágenes se puede visualizar mejor:

```
5  entity SPI_AXI_TMR_2_v1_0 is
6      generic (
7          -- Users to add parameters here
8
9          -- User parameters ends
10         -- Do not modify the parameters beyond this line
11
12
13         -- Parameters of Axi Slave Bus Interface S00_AXI
14         C_S00_AXI_DATA_WIDTH    : integer   := 32;
15         C_S00_AXI_ADDR_WIDTH    : integer   := 4
16     );
17     port (
18         -- Users to add ports here
19
20         -- User ports ends
21         -- Do not modify the ports beyond this line
22
23
24         -- Ports of Axi Slave Bus Interface S00_AXI
25         s00_axi_aclk    : in std_logic;
```

Figura 141. Sección para agregar entradas y salidas a AXI4 Lite

```
51     -- component declaration
52     component SPI_AXI_TMR_2_v1_0_S00_AXI is
53         generic (
54         C_S_AXI_DATA_WIDTH  : integer   := 32;
55         C_S_AXI_ADDR_WIDTH  : integer   := 4
56         );
57         port (
58
59         -- Users to add parameters here
60
61         -- User parameters ends
```

Figura 142. Sección para agregar entradas y salidas a la declaración del componente AXI4 Lite

x. En el mismo archivo se vinculan las entradas y salidas del segundo archivo con las del primero, en este caso no se visualizan las señales de la interfaz SPI porque se muestra el lugar en donde deben de ir.

El nombre de la señal en la entidad se asociará con “=>” al nombre del componente seguido de una coma “,” como se muestra a continuación:

```
87  begin
88
89  -- Instantiation of Axi Bus Interface S00_AXI
90  SPI_AXI_TMR_2_v1_0_S00_AXI_inst : SPI_AXI_TMR_2_v1_0_S00_AXI
91      generic map (
92          C_S_AXI_DATA_WIDTH  => C_S00_AXI_DATA_WIDTH,
93          C_S_AXI_ADDR_WIDTH  => C_S00_AXI_ADDR_WIDTH
94      )
95      port map (
96          -- Users to add parameters here
97
98          -- User parameters ends
99          S_AXI_ACLK  => s00_axi_aclk,
100         S_AXI_ARESETN   => s00_axi_aresetn,
101         S_AXI_AWADDR    => s00_axi_awaddr,
```

Figura 143. Asociación de entradas y salidas de la entidad con el componente.

xi. Por último, no olvidar agregar los archivos en donde se encuentra la descripción en VHDL de los componentes utilizados como son los de Flip-Flop, multiplexores, divisor de frecuencia, entre otros.

Para poder empaquetar la IP y poder utilizarla es necesario continuar con pequeños pasos como los siguientes:

i. Seleccionamos la pestaña “PACKAGE IP – SPI_AXI_TMR_2”, damos clic en File Groups à Merge changes from File Groups Wizard.

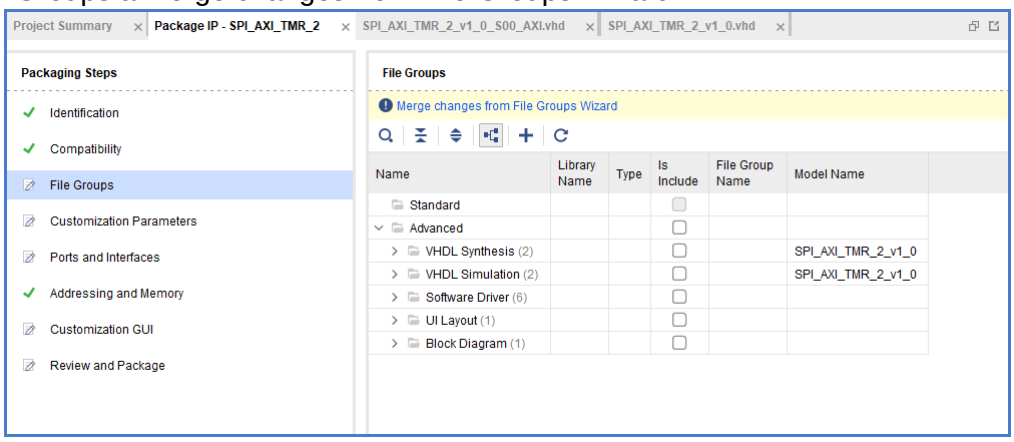


Figura 144. Package IP (1)

ii. El siguiente paso damos clic en “Customization Parameters” y después en “Merge Change from Customization Parameters Wizard”

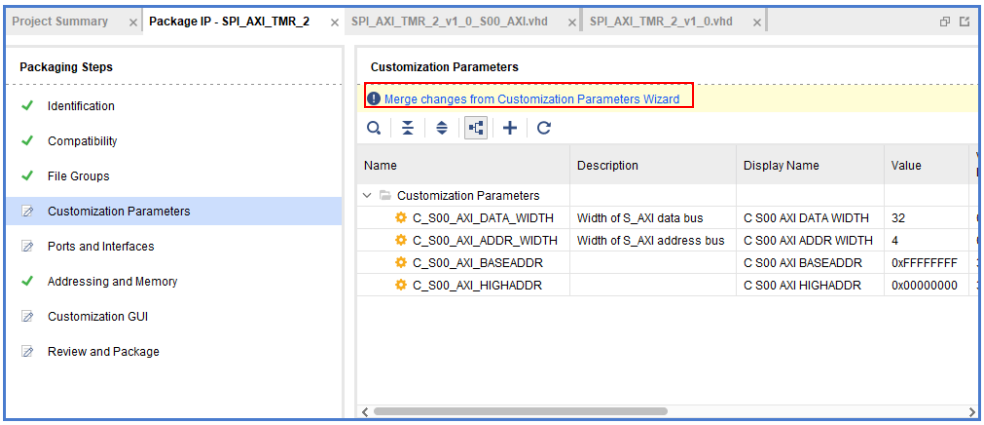


Figura 145. Package IP (2)

iii. Se verifica que los cambios hayan sido aplicados dando clic en “Ports and Interfaces” y visualizando el diagrama de la IP con sus entradas y salidas.

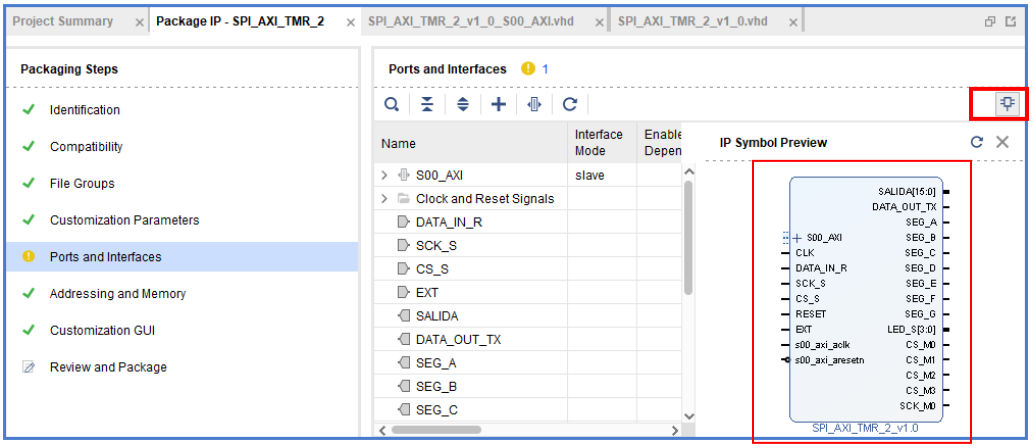


Figura 146. . Package IP - Verificación de empaquetamiento

iv. Al final dar clic en “Review and Package” y dar un clic en “Re-Package IP”, cerramos la ventana para poder seguir trabajando en la pantalla principal (Figura 129).

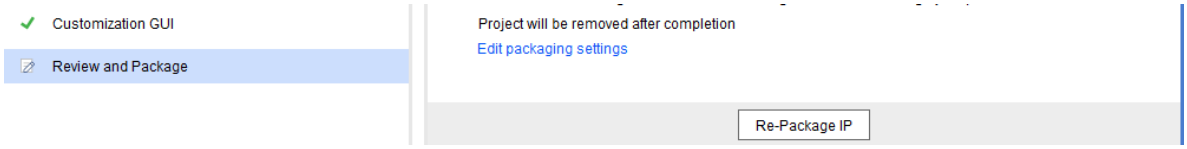


Figura 147. Package IP - Actualizar IP - Re empaquetar

Hasta este punto la creación de la IP fue completada, aún falta generar el entorno de trabajo y la aplicación de Vitis en el proyecto. Esto se mostrará en el siguiente Anexo.

Anexo 4. Creación del entorno de pruebas.

Es necesario realizar algunas acciones dentro del software VIVADO para poder hacer uso de la IP. Este procedimiento se muestra a continuación:

Al terminar la creación de la IP con el módulo SPI integrado, el software se redirecciona al espacio de trabajo principal el cual nos permitirá crear un diseño de bloques, este servirá para hacer uso de nuestra IP.

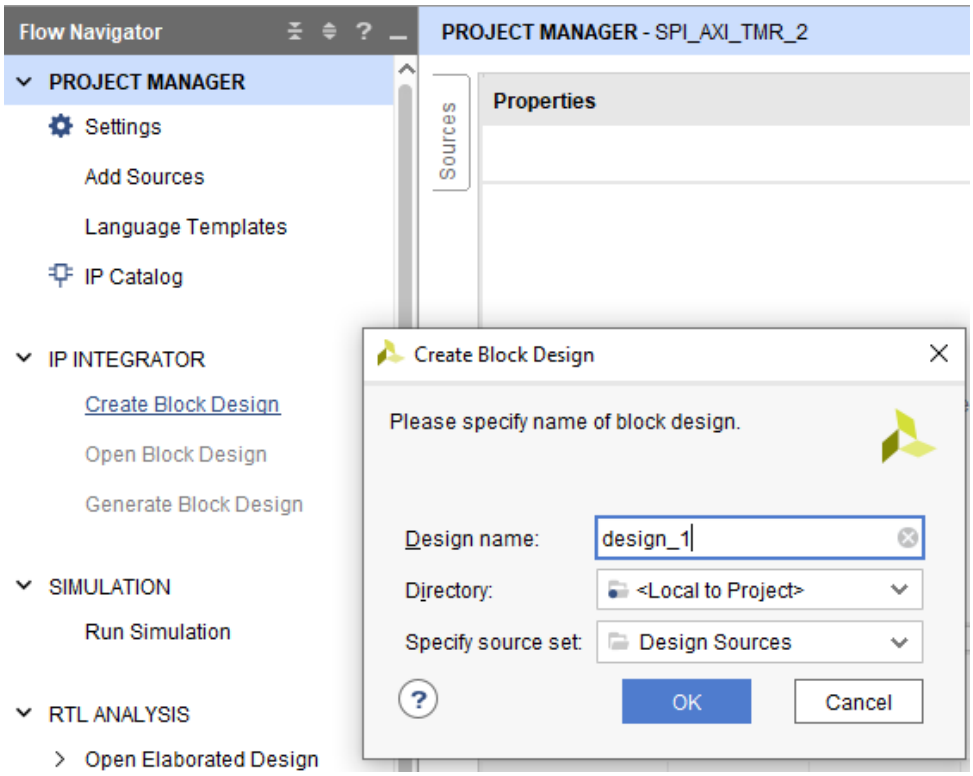


Figura 148. Creación de un bloque de trabajo

En la Figura 148 se muestra la creación de dicho espacio de trabajo siguiendo la siguiente ruta:

IP INTEGRATOR → ”Create Block Desing” → OK

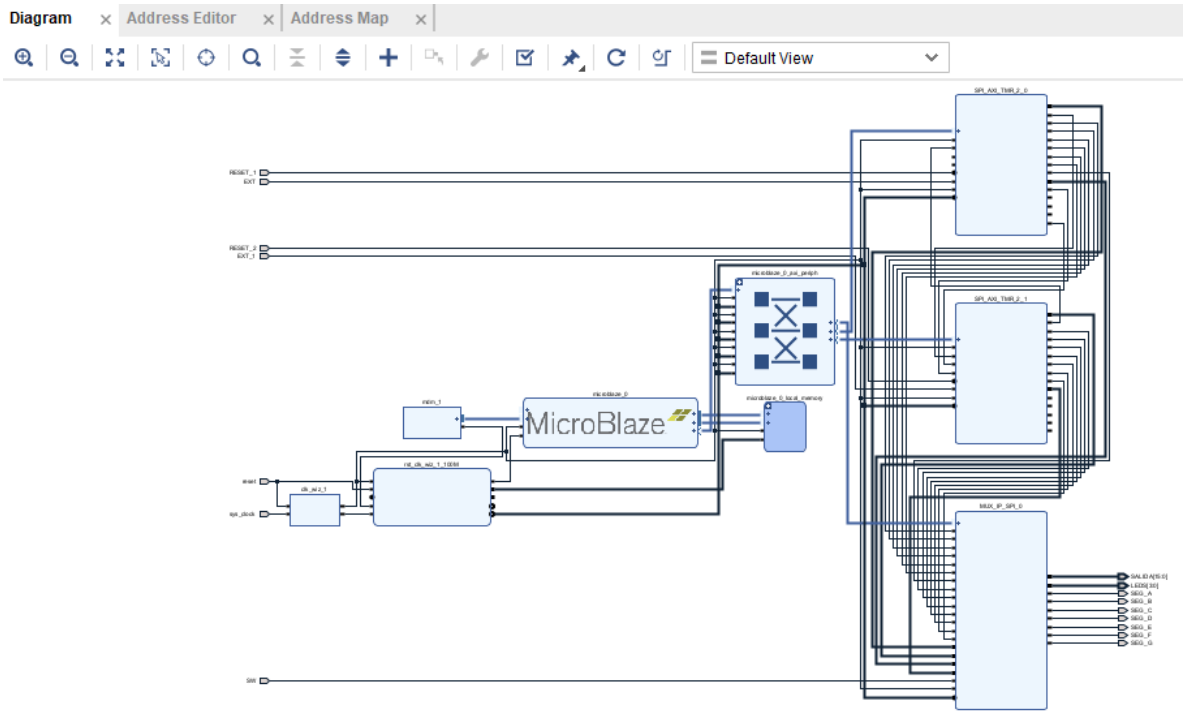
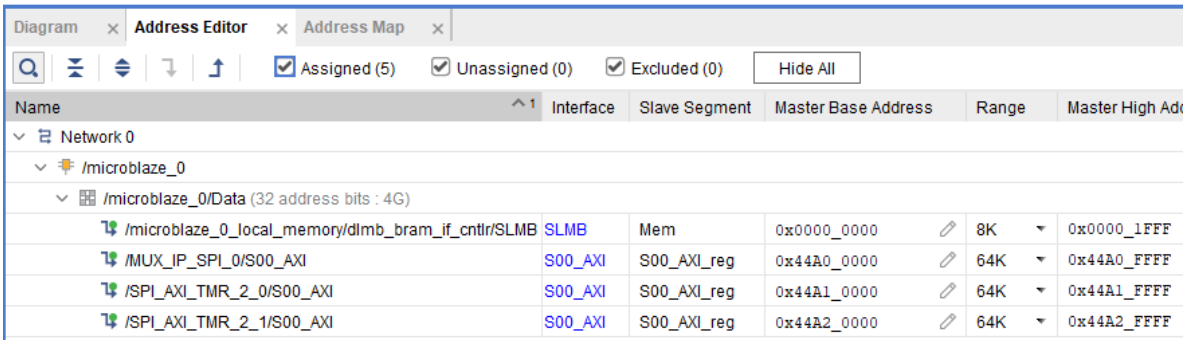


Figura 149. Diseño de bloque creado

Es necesario ya tener lista la IP de nuestro SPI y la nueva IP para las pruebas. En la Figura 149 se agregó el procesador MicroBlaze, en este punto VIVADO de manera automática agrega los módulos necesarios para hacer uso de nuestra IP junto con sus conexiones. Después, se agregan dos IP SPI una se configurará como maestro y la otra como esclavo interconectándose entre sí, es decir, dentro de la misma tarjeta de trabajo se tendrá una transferencia de datos entre estas dos IP. Por último, se agrega la IP encargada de mostrar los resultados en la tarjeta.



Name	Interface	Slave Segment	Master Base Address	Range	Master High Ad
Network 0					
/microblaze_0					
/microblaze_0/Data (32 address bits : 4G)					
/microblaze_0_local_memory/dlmb_bram_if_cntlr/SLMB	SLMB	Mem	0x0000_0000	8K	0x0000_1FFF
/MUX_IP_SPI_0/S00_AXI	S00_AXI	S00_AXI_reg	0x44A0_0000	64K	0x44A0_FFFF
/SPI_AXI_TMR_2_0/S00_AXI	S00_AXI	S00_AXI_reg	0x44A1_0000	64K	0x44A1_FFFF
/SPI_AXI_TMR_2_1/S00_AXI	S00_AXI	S00_AXI_reg	0x44A2_0000	64K	0x44A2_FFFF

Figura 150. Direcciones de memoria de cada IP.

Cuando se agrega una IP al diseño de bloque se le asigna una dirección de memoria, en este caso a la primer “SPI_AXI_TMR_2_0” su dirección es 0x44A1_0000 y a “SPI_AXI_TMR_2_1” se le asigna 0x44A2_0000, la cual mediante esta dirección se podrán modificar los valores de los registros de control y escribir un dato que se desea transmitir.

Recordando del anexo anterior, las señales de AXI4 Lite asociadas al módulo SPI son desde “slv_reg0” hasta “slv_reg3” y a cada señal se le asigna una dirección de memoria de cómo se indica a continuación:

- “Slv_reg0” 0x0000 + 0X44A1_0000
- “Slv_reg1” 0x0004 + 0X44A1_0000
- “Slv_reg2” 0x0008 + 0X44A1_0000
- “Slv_reg3” 0x000C + 0X44A1_0000

Con dicha dirección de memoria se puede escribir o leer algún dato de la IP. Por ejemplo, los registros de control o registros de escritura se pueden modificar escribiendo a “slv_reg0” y “slv_reg1”, respectivamente.

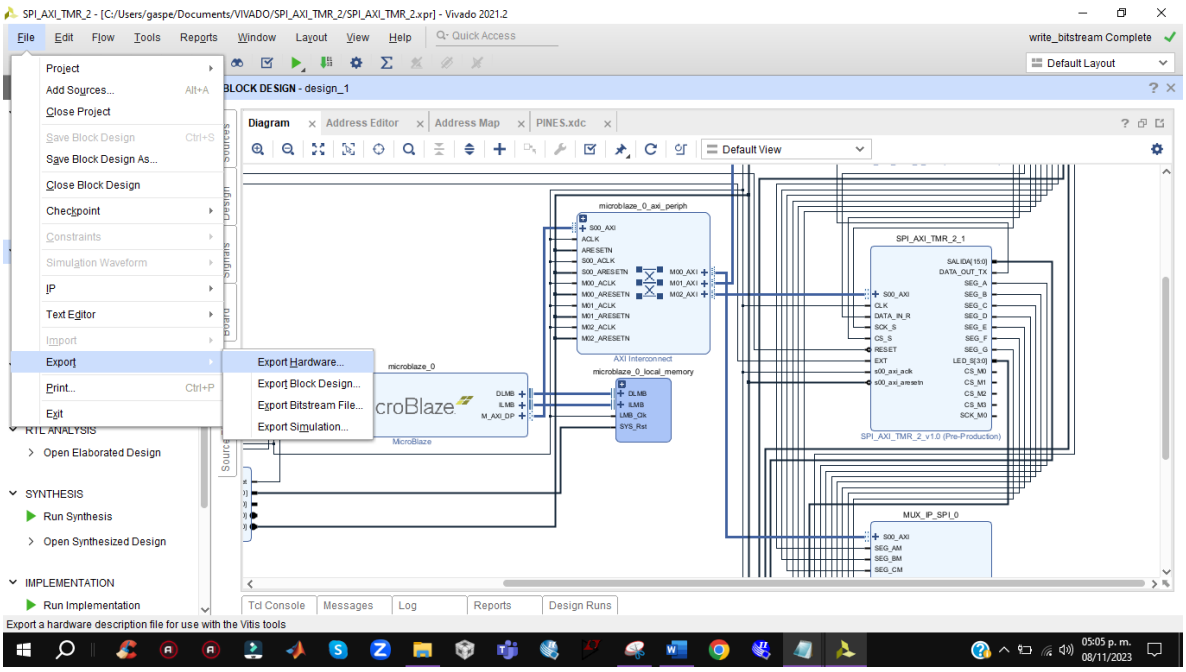


Figura 151. Exportar Hardware

El software ayuda a conectar automáticamente las IP creadas por el usuario con el MicroBlaze. Después de tener correctamente conectado el diseño se siguen los siguientes pasos:

- Generar el diseño de bloque.
- Cree el envolvente/wrapper HDL del sistema
- Sintetizar, Implementar y generar el Bitstream. Para este punto es necesario asignar los pines mediante un archivo. XDC
- Exportar el Hardware como se muestra en la Figura 151, incluyendo el Bitstream.

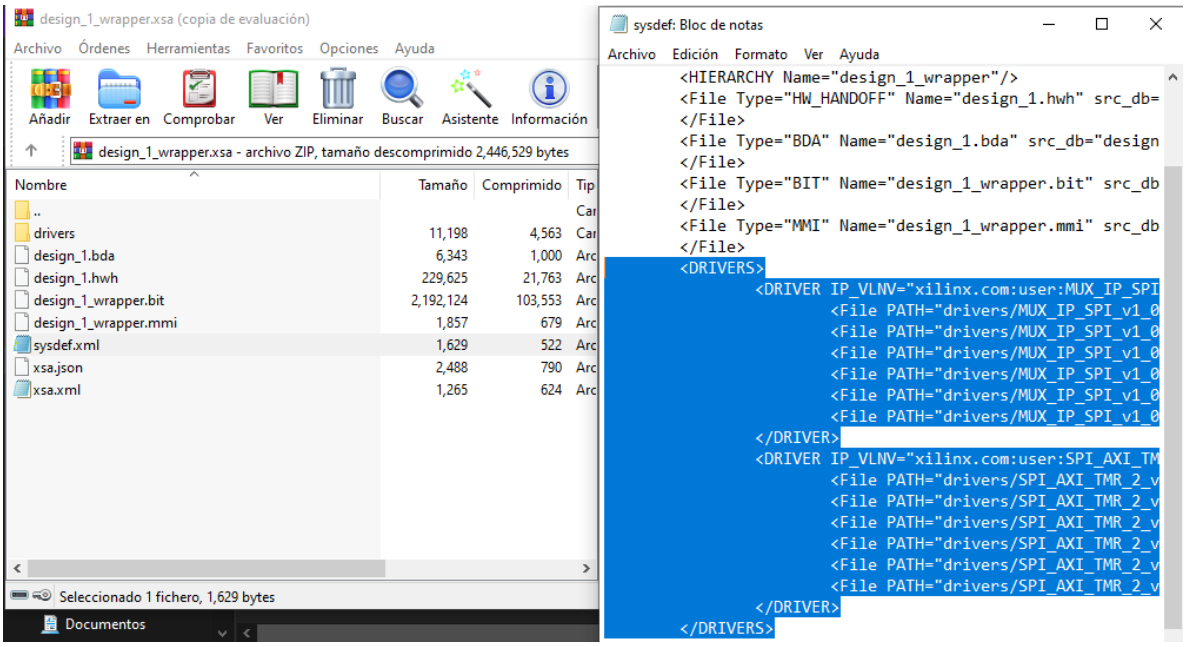


Figura 152. Error en archivo Wrapper

En el directorio del proyecto se crea un archivo llamado “design_1_wrapper.xsa” el cual es necesario abrirlo con WinRar y posteriormente abrir “sysdef.xml” para borrar el texto sombreado mostrado en la Figura 152. Si este paso no es ejecutado, el proyecto en el Software VITIS generará un error al momento de cargar el programa a la tarjeta de trabajo y no se podrá utilizar de ninguna manera el proyecto creado.

Como últimos pasos

- Crear proyecto en VITIS añadiendo el archivo “design_1_wrapper.xsa”.
- Programar tarjeta (Xilinx → Program Device → Program)
- Construir depuración, así como verificar una correcta codificación en C (Build Debug).
- Depurar (Debug)

En la Figura 153 se muestra el codigo generado para poder escribir en los registros de control mediante la interfaz AXI4 Lite. Se observan las direcciones de memoria de cada “slv_reg” de la linea 3 hasta la 11 asignandoles un nombre para poder identificarlos. En el diseño de bloques las IP SPI se encuentran conectadas entre si, y en la linea 22 y 23 del código se asignan los valores correspondientes a los registros de control en donde una IP operara en modo esclavo y la otra en modo maestro, es decir, “slv_reg0” o “Data_reg0” tanto para esclavo como para maestro corresponde a los registros de control. Por ultimo, se escribe el dato en “Slv_reg1” o “Data_reg1” 0x2222 en el maestro para poder transmitirlo en el esclavo y 0x8888 en el esclavo para transmitirlo al maestro.

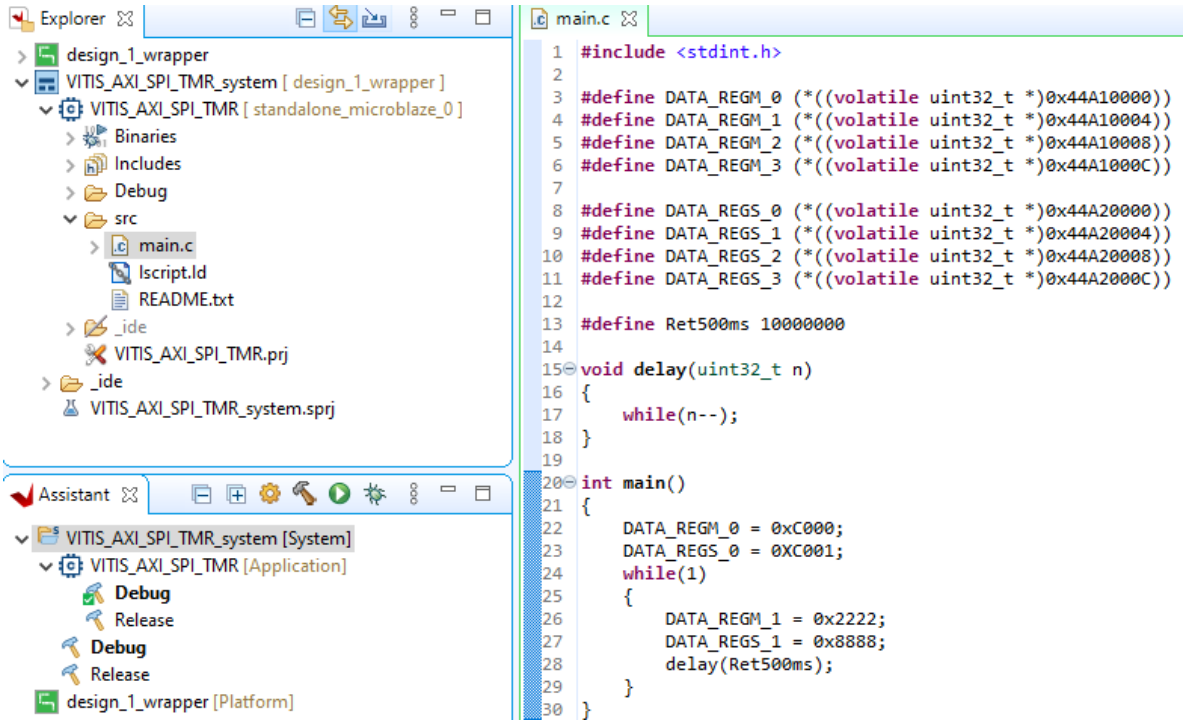


Figura 153. VITIS - Código en C.

10. REFERENCIAS.

[1]. Aykenar M.B., Soysal G., (Octubre 2020) "Design and Implementation of a Lightweight SPI Master IP for Low Cost FPGAs" ,IEEE, pags 4.

[2]. Pahlevi R.R., Putrada A.G., Abdurohman M., (Mayo 2018) "Fast UART and SPI Protocol for Scalable IoT Platform", VI Congreso Internacional de Tecnologías de la Información y la Comunicación (ICoTIC), 6 pags.

[3]. https://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf?ts=1688565458518&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTM4C1294NCPDT

[4]. Hobden P., Srivastava S., (Diciembre 2018) "Low Cost FPGA Implementation of a SPI over High Speed Optical SerDes", Simposio internacional IEEE 2018 sobre sistemas electrónicos inteligentes (iSES), 6 pags.

[5]. Raj M., (Julio 2017), "100 MHz High Speed SPI Master: Design, Implementation and Study on Limitations of using SPI at High Speed", International Journal on Recent and Innovation Trends in Computing and Communication, 4 pags.

[6]. Harten L., Jordans R., Pourshaghaghi H., (Septiembre 2017), "Necessity of Fault Tolerance Techniques in Xilinx Kintex 7 FPGA Devices for Space Missions: A Case Study", 2017 Euromicro Conference on Digital System Design, 8 pags.

[7]. Espinosa J., Andres D., Ruiz J.C., Gil P., (Agosto 2012) "TOLERATING MULTIPLE FAULTS WITH PROXIMATE MANIFESTATIONS IN FPGA-BASED CRITICAL DESIGNS FOR HARSH ENVIRONMENTS", 22nd International Conference on Field Programmable Logic and Applications (FPL), 8 pags.

[8]. Perez Celis J.A., Ferrer Pérez J.A, Santillán Gutierrez S.D., de la Rosa Nieves S., (Marzo 2016) "Simulation of Fault-Tolerant Space Systems Based on COTS Devices With GPSS", IEEE Systems Journal, 6 pags.

[9]. Pavón J., (2015) "MOVIMIENTO EFICIENTE DE DATOS PARA UN SOC BASADO EN ZYNQ", Universidad de Alcalá Escuela Politécnica Superior.

[10]. Mengfei Yang, Hua, Feng, & Gong, (2017), "Fault-Tolerance Techniques for Spacecraft Control Computers", John Wiley & Sons Singapore, 343 pags

[11]. AXI4 en Xilinx
Recuperado de: https://www.xilinx.com/products/intellectual-property/axi.html