1. Introducción

El módulo de comunicación *Serial Peripheral Interface* Bootloader (SPI-Bootloader) es una unidad de diseño específico, que permite la comunicación entre la memoria flash serial IS25WP032D y el bus de comunicación con el microprocesador del proyecto "Procesador RISC-V en HV para aplicaciones médicas". La comunicación con la memoria se realiza a través del protocolo SPI, basado en la estandarización definida en *"KeyStone Architecture Serial Peripheral Interface (SPI)"* de la empresa Texas Instrument. Este módulo se desarrolla bajo el flujo de diseño de circuitos integrados digitales en las herramientas EDA de Synopsys con la tecnología XFAB de 0.18 μm.

2. Marco teórico

2.1. Protocolo de comunicación SPI

El protocolo SPI es un de comunicación serial que basa su funcionamiento en cuatro líneas de comunicación, divididas de la siguiente manera: una línea de reloj *(SCK)*, habilitador de esclavo*(SS)*, datos desde esclavo a maestro *(MISO)* y datos de maestro a esclavo*(MOSI)*.[2] La línea de habilitador de esclavo o *Slave Select* es una señal distinta por cada esclavo, caso contrario a las demás señales, las cuales comparten el mismo bus para todos los esclavos.[2]
En este tipo de comunicación se puede distinguir dos principales topologías. La primera de estas es de un módulo SPI maestro conectado a una única unidad SPI esclavo. La segunda se compone de un módulo SPI maestro conectado con múltiples unidades SPI esclavos. [2]
No existe un estándar general del protocolo, por lo que existen solo pautas generales o estandarizaciones de algunas empresas. Una de dichas estandarizaciones es la que se encuentra en *"KeyStone Architecture Serial Peripheral Interface (SPI)"* de la empresa Texas Instrument. Los detalles se pueden encontrar en el documento antes mencionado, sin embargo se hará una pequeña descripción de las principales especificaciones de importancia para el desarrollo de este proyecto. Algunas de las especificaciones son:

- Tiene registros de desplazamiento y *buffer* de 16 bits para los datos de transmisión y recepción.
- Generador de reloj en baudios de 8 bits.
- Pin de reloj serial, esclavo entra/maestro sale, esclavo sale/maestro entra y múltiples seleccionadores de esclavo.
- Programable la frecuencia de SPI, el largo del dato esperado a enviar o recibir, el modo de operación.
- Capacidad de operar a mas de 66 MHz.
- Programable tiempo de retardo entre operaciones.
- Capacidad de operar en configuración de 3 o 4 pines.
- Registros de configuración de SPI, manejo de interrupciones, retardos de SPI y datos.

2.2. Memoria IS25WP032D

La memoria IS25WP032D es una memoria flash serial, con un tamaño de 4 MB. Las principales características de la memoria se describen a continuación:

- Soporta protocolos de comunicación SPI, Fast, Dual, Dual I/O, Quad, Quad I/O, SPI DTR, Dual I/O DTR, Quad I/O DTR y QPI.
- Opera a una frecuencia máxima de 50 MHz en funcionamiento normal y hasta 133 MHz en lectura rápida.
- Soporta los modos 0 y 3 del SPI.
- Soporta más de 100000 ciclos de escritura y borrado.
- Lectura continua de 8/16/32/64 bytes.
- Tensión de alimentación de 1.65V a 1.95V.
- Un valor típico de 8μA de corriente de suspensión y de 4mA de corriente de lectura activa.

En cuanto al protocolo de la memoria esta cuenta con distintos comandos. Estos comandos permiten acceder a algunos registros de configuración de la memoria, realizar escritura y lectura en memoria, borrado de sectores de memoria, entre otros. Se hará énfasis en los comandos que permiten la escritura y la lectura en memoria.
Para realizar la escritura en memoria son necesarios dos instrucciones. La primera es la habilitación de escritura (*Write Enable Operation*) que permite habilitar el latch de escritura. El comando de habilitación de escritura se realiza con el valor 8'h06. El segundo comando es el de escritura (*Page Program Operation*) que permite la escritura de hasta 256 bytes de datos en áreas no reservadas de memoria. El comando de escritura se realiza con el valor 8'h02. Como parte del argumento del comando debe contener la dirección inicial de escritura. Si se escribe más de un byte, la memoria aumentará automáticamente la dirección dada para guardar el siguiente byte en la dirección correcta.
Para realizar la lectura en memoria es necesario solamente un comando: *Normal Read Operation*. Este comando permite realizar lectura en memoria. Para realizar este proceso se debe enviar el valor 8'h03 a la memoria, seguido de la dirección de memoria a leer. Similar al comando de escritura, la memoria aumentará internamente la dirección recibida y hará lectura del siguiente espacio de memoria, lo cual se repetirá hasta que se deshabilite la memoria (señal de SC del SPI). En la figura 1 se observa la secuencia de la lectura normal de memoria.

Para obtener mayor información sobre la memoria se debe observar [4].

2.3. Flujo de diseño en herramientas EDA Synopsys

La herramientas de diseño de circuitos digitales EDA de Synopsys permiten llevar a cabo algunas de las etapas del flujo de diseño por las cuales deben pasar un circuito para su implementación con celdas de una cierta tecnología. En este caso la tecnología es XFAB de 0.18 μm de baja corriente de fuga *(low leakage)*.
El flujo de diseño de circuito integrados digitales, observados en la figura 2, contempla las etapas denominadas síntesis lógica y síntesis física, las cuales se realizan en herramientas de diseño asistido

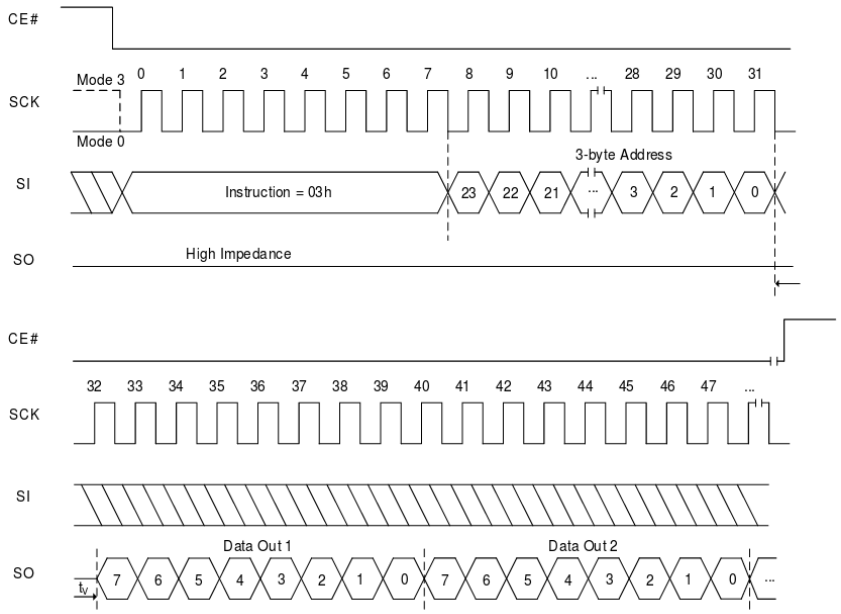

Figura 1: Secuencia de la lectura normal de memoria IS25WP032D. CITAR

por computadora(CAD), como las herramientas EDA de Synopsys. La síntesis lógica se realiza con la herramienta *Design Compiler*, la cual convierte el diseño escrito en lenguaje HDL en un mapeo de nodos a nivel de compuertas en una tecnológica en específico. La síntesis física se realiza con la herramienta *IC Compiler*, la cual realiza el posicionamiento y enrutamiento de las celdas de las tecnología según el el mapeo de nodos generado por la síntesis lógica. [3]

El uso de estas herramientas para el desarrollo del flujo es de vital importancia, ya que además de realizar la construcción del circuito en la tecnología especificada, nos brinda datos del área, potencia y temporizado.

3. Descripción funcional SPI Bootloader

El módulo SPI Bootloader es implementado en lenguaje de descripción de hardware Verilog. La unidad tiene como función realizar el proceso de carga de programa de instrucciones RISC-V y la escritura/lectura de la memoria IS25WP032D. Las principales características de la implementación se describen a continuación.

- Protocolo de comunicación SPI especifico para la memoria IS25WP032D.
- Soporta cuatro señales de interfaz SPI (MOSI, MISO, SCK y SS).
- Configurable el modo de actuación de la señal SS del SPI para la memoria IS25WP032D.
- Configurable dos modos de fase y polaridad del SCK del SPI para la memoria IS25WP032D.
- Configurable la frecuencia de operación del SCK del SPI para la memoria IS25WP032D.
- Configurable el primer bit de salida de las transacciones para la memoria IS25WP032D.
- Configurable el tamaño de paquete de recepción y envio del SPI.

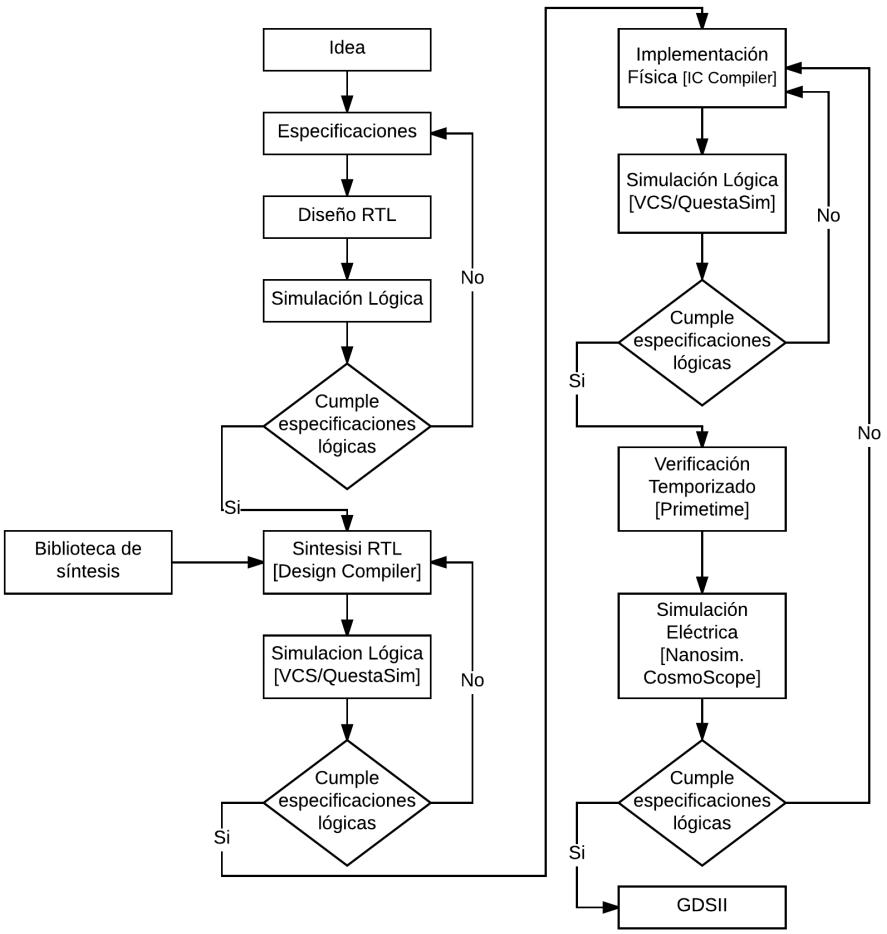

Figura 2: Flujo de diseño de circuito integrados digitales.

- Modo master de operación SPI.
- Capacidad de responder a tres comandos: escritura en memoria, lectura en memoria y *bootloader*.
- Capacidad de realizar transacciones con memoria externa de 4, 2 y 1 byte.

En la figura 3 se observa el diagrama de I/O del SPI Bootloader.
El SPI Bootloader tiene un canal *half duplex* síncrono que soporta la configuración de 4 pines del protocolo SPI para la comunicación con la memoria IS25WP032D y capacidad de manejo de las señales del bus de interconexión para el proyecto para el que se es desarrollado.
El SPI Bootloader soporta la transferencia continua de datos, respetando las características de transmisión de la memoria y del bus de interconexión. Tiene diferentes frecuencias de operación para la señal de reloj SCK, en base al reloj del sistema.
El módulo desarrollado tiene una configuración específica para el manejo de la memoria IS25WP032D, definida en el módulo *Control Top SPI Bootloader*. Sin embargo, el bloque SPI tiene un diseño adaptable para la comunicación con otros dispositivos como maestro.
Los registros con los que cuenta el módulo son de tamaños variables. Se encuentran registros de 1, 2, 8, 16, 32 y 64 bits. La razón de no contar con registros de un tamaño estandarizado se debe a que uno de los objetivos del módulo es tener el menor tamaño posible.
En cuanto a comportamiento de las señales, se tiene dos principales operaciones de importancia: la

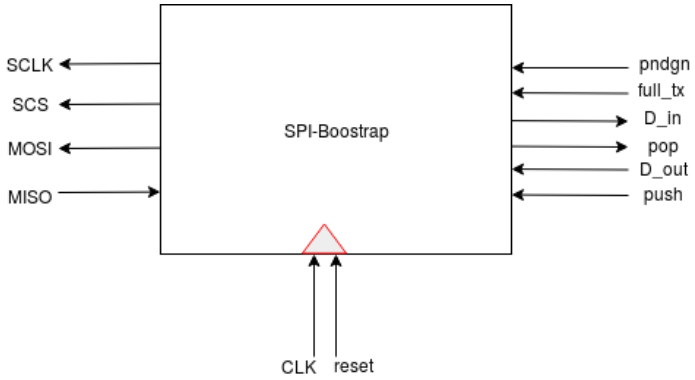

Figura 3: Diagrama de I/O de SPI Bootloader.

escritura en memoria y la lectura en memoria. La operación de *bootloader* es una secuencia de lecturas continuas a la memoria, por lo que es solamente necesario conocer el comportamiento de la lectura de memoria para entender dicha operación. En la figura 4 se observa la escritura en memoria, donde existe dos envíos continuos de datos a memoria: el comando *Write Enable Operation* y *Page Program Operation*, posterior a la recepción de la instrucción de escritura.

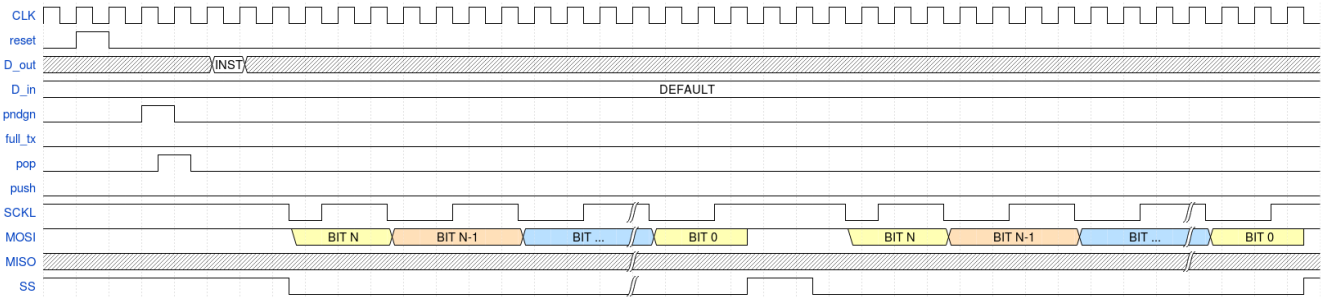

Figura 4: Diagrama de tiempo de escritura SPI Bootloader.

En la figura 5 se observa la lectura en memoria, donde se envía el comando *Normal Read Operation* y posterior a la lectura de datos se realiza el envío del dato al bus de interconexión.

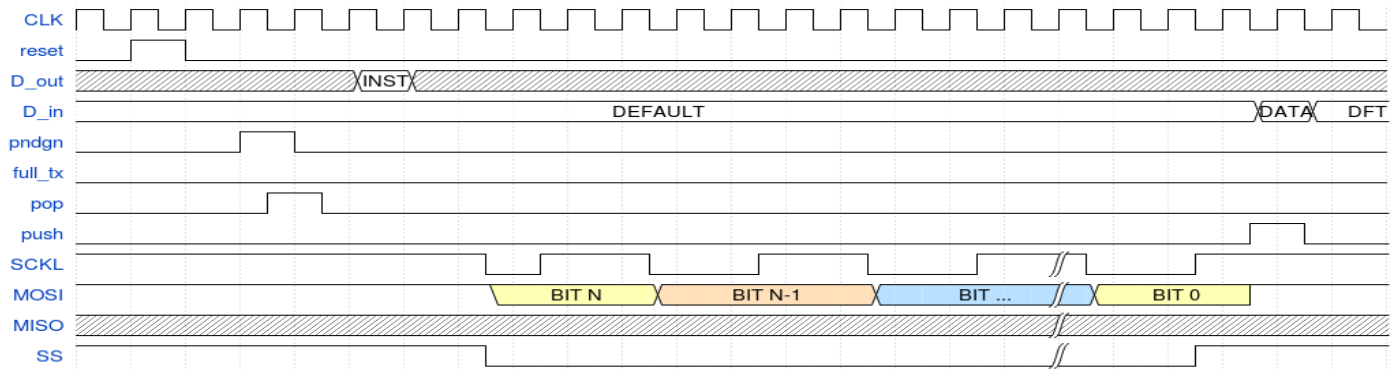

Figura 5: Diagrama de tiempo de lectura SPI Bootloader.

El diagrama de bloque de nivel superior para el SPI Bootloader se encuentra en la figura 6. A continuación se hará una descripción de los diferentes módulos mostrados en la figura.

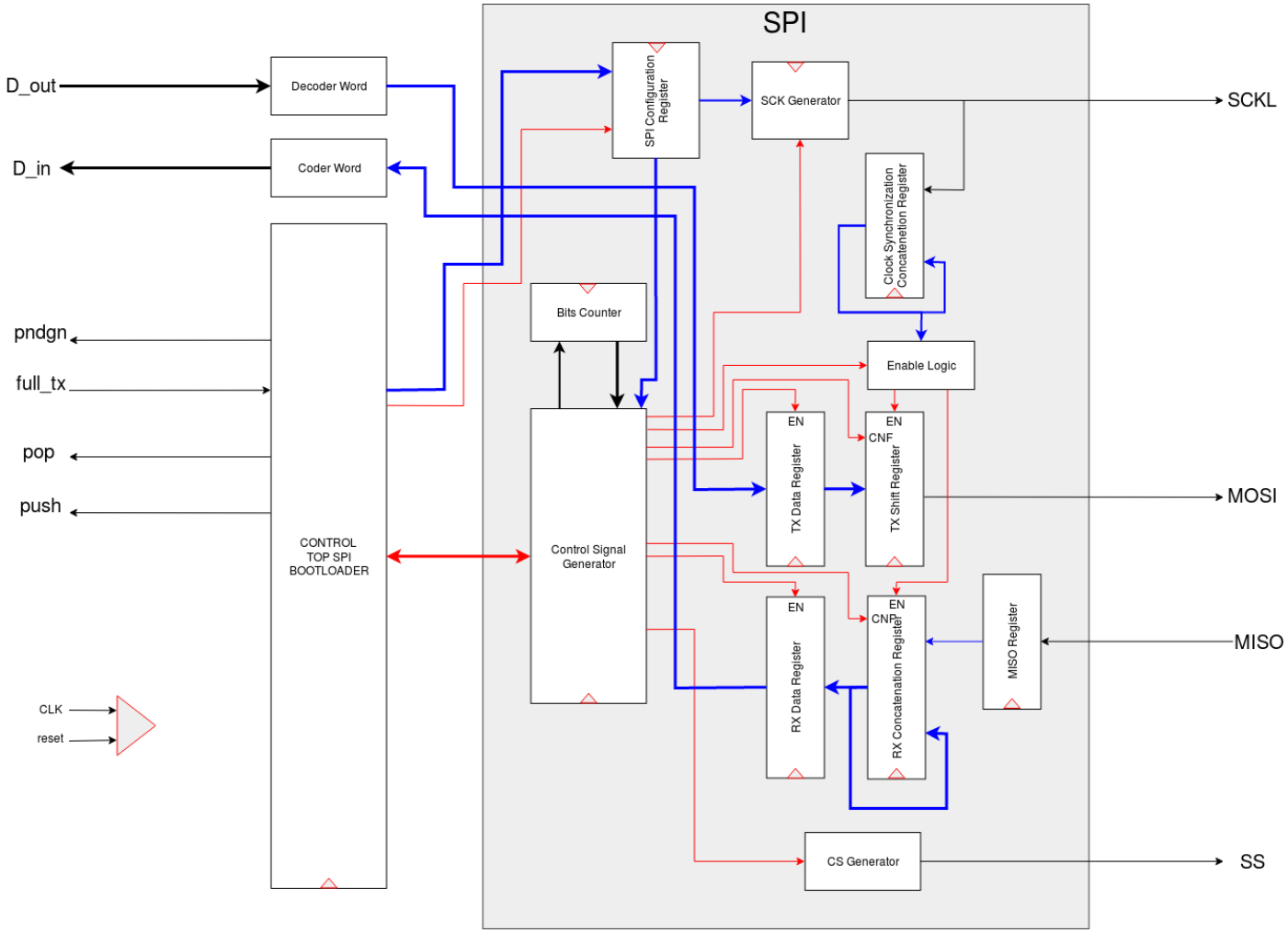

Figura 6: Diagrama de bloque del nivel superior para el SPI Bootloader.

3.1. SPI

El módulo SPI realiza la operación de comunicación con el protocolo SPI. Esta sección del módulo es configurable para adaptarlo a una variedad de aplicaciones, por lo que necesita de un controlador.
En la figura 7 se muestra un diagrama de I/O de la unidad. Las señales `flag_send` y `config_enable` son señales de control. La señal `data_config` es la señal que provee la configuración del SPI. La señal `data_send` contiene la palabra a enviar en el proceso de comunicación con el dispositivo esclavo. La señal `data_read` contiene el dato obtenido de la comunicación con el esclavo. La señal `end_send` indica la finalización de la operación de comunicación. Finalmente las señales `SCK, MISO, MOSI` y `SC` son las señales del interfaz SPI.
En la figura 8 se muestra el diagrama de tiempo de comunicación del módulo SPI con un dispositivo esclavo con una configuración particular. En la misma se muestra como se realiza la configuración del dispositivo y como se realiza el inicio de la transmisión de datos y la finalización del mismo.
De igual forma es necesario detallar los módulos internos del SPI, con el fín de definir de mejor manera la generación del reloj SCK, la sincronización de operaciones entre el reloj SCK y el reloj de sistemas, el registro de configuración y los valores almacenados, entre otros aspectos.

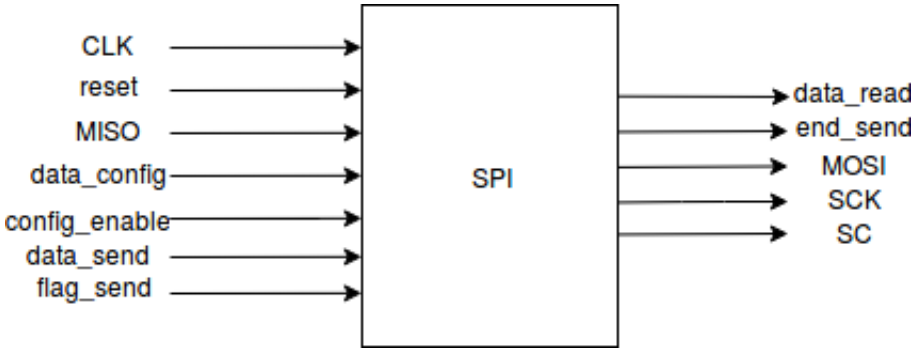

Figura 7: Diagrama de I/O de SPI.

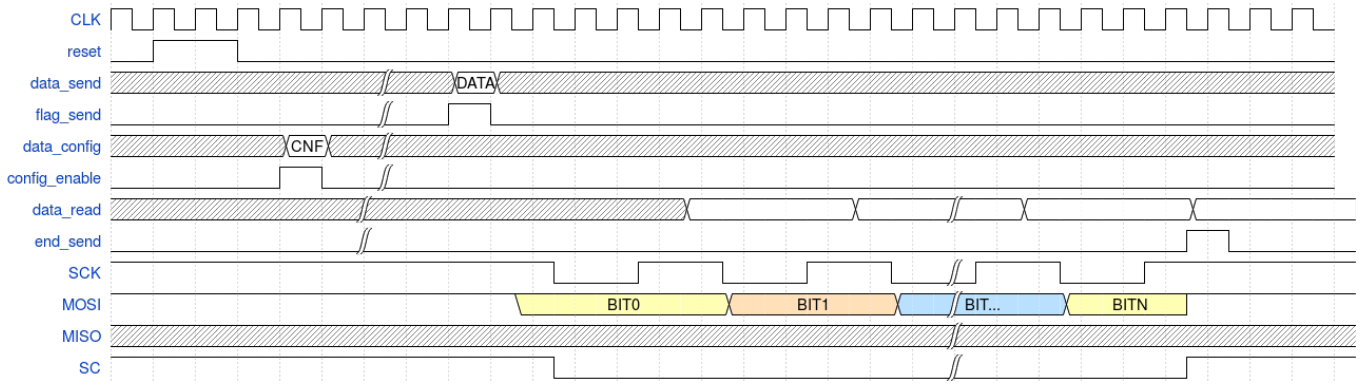

Figura 8: Diagrama de tiempo del proceso de comunicación con protocolo SPI del módulo SPI.

3.1.1. *SPI Configuration Register*

Este módulo es un registro de 16 bits con los valores de configuración del SPI. La configuración fue adaptada a las necesidades del proyecto, con el fin de hacer manejo de algunas capacidades del interfaz SPI de importancia para la comunicación con la memoria seleccionada. En la tabla 1 se puede observar los valores contenidos en este registro.

Tabla 1: Valores en registro *SPI Configuration Register*

15 8	7 6 4	3	2	1	0
SIZEWORD	POWERDOWN	CLKSCALE	CSHOLD	FBO	CLKMOD	ENABLE

La descripción de cada valor se observa a en la tabla 2.

3.1.2. *Control Signal Generator*

Este bloque es una máquina de estados que genera las señales de control para los diferentes módulos que componen la unidad SPI. En la figura 9 se observa el diagrama de la máquina de estados.
A continuación se realiza una descripción de los estados vistos en la figura 9.

- `START`: En este estado se esta valorando si se habilita el envió o recepción de alguna palabra por el interfaz SPI. Las señales de control mantienen todas las unidades en sus valores por defecto en el estado de no transmisión.

Tabla 2: Descripción de contenido de *SPI Configuration Register*

Valor de configuración	Tamaño (bits)	Descripción
ENABLE	1	1: Habilitado 0: Deshabilitado Este valor deja la máquina de estados del SPI en un estado de pausa, manteniendo el ultimo valor de las señales de control.
CLKMOD	1	0: Modo 0 1: Modo 3 Este valor determina el modo del reloj (fase y polaridad) para la transmisión del SPI.
FBO	1	0: Bit menos significativo 1: Bit más significativo Este valor determina si el primer bit de salida/entrada de las señales MISO y MOSI es el más o menos significativo.
CSHOLD	1	0: CS en uno lógico 1: CS en cero lógico Este valor determina si la señal de CS selecciona al esclavo en uno logico o cero lógico.
CLKSCALE	3	00X: Relación 1:4 010: Relación 1:8 011: Relación 1:16 1XX: Relación 1:32 Este valor determina la relación entre el reloj de sistema y el reloj generado para la comunicación SPI.
PWRDOWN	1	0: Modo habilitado 1: Modo deshabilitado Este valor coloca a la unidad en un estado de reposo de operación. Todas las señales se mueven a sus valores de reset.
SIZEWORD	8	Este valor tiene el dato de la cantidad de bits a enviar por transacción.

- `LOAD`: En este estado se genera la carga del paquete de datos a enviar.
- `OPER`: En este estado se esta realizando el proceso de transmisión/recepción del paquete de datos. Además se genera la señal de CS del interfaz SPI correcta según la configuración de *SPI Configuration Register*. Se esta evaluando constantemente si la cantidad de bits enviados es igual al valor que se encuentra en *SPI Configuration Register*.
- `END`: En este estado se genera la bandera de finalización de operación. Posteriormente se vuelve al estado donde se evalúa si se debe enviar un nuevo dato.

3.1.3. *Bits counter*

Este bloque es un contador que es controlado por la máquina de estados del SPI. Este realiza la operación de suma si esta habilitado por la máquina de estados, si el *Clock Synchronozation Concatenetion*

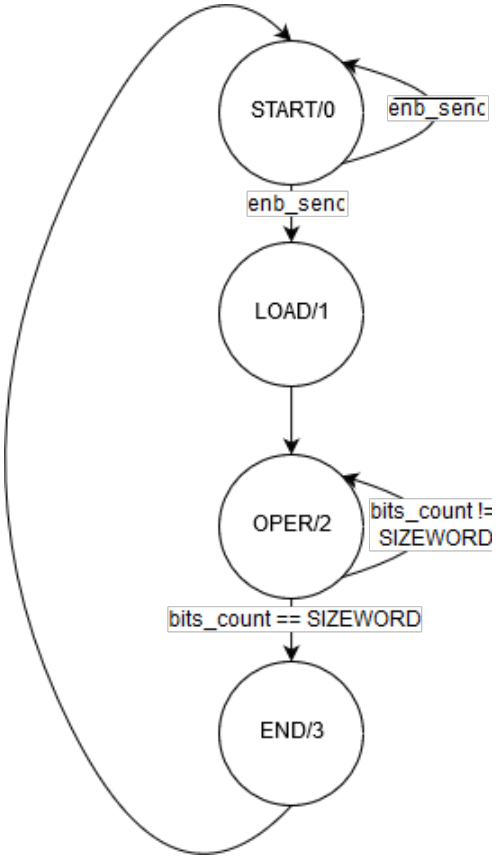

Figura 9: Diagrama de la máquina de estados de *Control Signal Generator*.

Register tiene un valor de 2'b10, se encuentra habilitado el valor `ENABLE` y deshabilitado `PWRDOWN` del *SPI Configuration Register*.

3.1.4. *MISO Register*

Este bloque es un registro para la señal de entrada MISO del SPI. Este registro es de importancia ya que permite tener un correcto controlador de la señal de entrada, con el cual se pueda disminuir el retardo de la señal debido al cableado u otro fenómeno.

3.1.5. *CS Generator*

Este es un bloque combinacional manejado por el *Control Signal Generator* del SPI y dependiente del valor de `CSHOLD` del registro de configuración. Su función es generar la señal de control *SS* del protocolo SPI.

3.1.6. *RX Concatenation Register*

Este bloque es un registro de concatenación de 32 bits de los valores recibidos de forma serial de la señal MISO. Su función es paralelizar los datos de entrada, al concatenar el nuevo dato de entrada serial a los 31 valores anteriores.
El registro recibe los valores de configuración `FBO` del *SPI Configuration Register*. Este valor permite determinar si el dato de entrante es un bit más o menos significativo.

3.1.7. *RX Data Register*

Este bloque es un registro de 32 bits que tiene como función obtener el dato de recepción de la comunicación SPI generado por el *RX Concatenation Register* al finalizar la transmisión. La señal de habilitación es dado por el módulo *Control Signal Generator*.

3.1.8. *TX Data Register*

Este bloque es un registro de 64 bits con función de capturar el dato que se enviará por la señal MOSI. Su señal de habilitación es dada por el módulo *Control Signal Generator*.

3.1.9. *TX Shift Register*

Este bloque es un registro de desplazamiento de 64 bits. Su función es serializar los datos de transmisión para la linea MOSI. Recibe el valor de configuración `FBO` del *SPI Configuration Register*, que determina si el desplazamiento se realiza hacia la izquierda o hacia la derecha. En cualquiera de los dos casos se introduce un uno lógico al final o inicio de registro con cada desplazamiento. El objetivo de tener alternancia en la forma de desplazamiento es para que el dato se serialice del bit más significativo al menos significativo o viceversa.

3.1.10. *Clock Synchronization Concatenation Register*

Este registro tiene un tamaño de 2 bits. Su función principal es realizar el muestreo de la señal SCK, el cual permitirá realizar la sincronización de las operaciones en los flanco de dicho reloj pero haciendo uso del reloj de sistema.
En la figura 10 se observa un diagrama de bloques de la lógica de este registro. Además, en la tabla 3 se muestra los distintos valores obtenidos y el valor de sincronización obtenido.

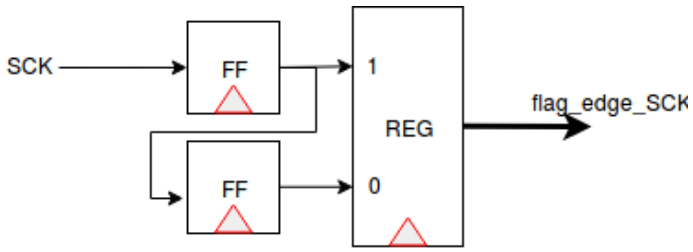

Figura 10: Diagrama de bloque de *Clock Synchronozation Concatenetion Register*.

Tabla 3: Relación de valores de *Clock Synchronozation Concatenetion Register* y valores de sincronización

Valor binario de registro	Valor de sincronización
2'b00	SCK en bajo
2'b01	Flanco negativo SCK
2'b10	Flanco positivo de SCK
2'b11	SCK en alto

3.1.11. *SCK Generator*

Este módulo se encarga de la generación del reloj del interfaz SPI a partir de la configuración de frecuencia de reloj obtenida de `CLKSCALE` del *SPI Configuration Register*. En la figura 11 se puede observar el diagrama de bloques para esta unidad.
Existe un bloque de lógica que esta en constante verificación del valor del contador *SCK Counter* y el valor de suma para el divisor de frecuencia para generar una frecuencia especifica, la cual es determinada por el valor de configuración `CLKSCALE`. Otro valor de configuración que es evaluada en esta lógica de comparación es `PWRDOWN` del *SPI Configuration Register*, que permite dejar en modo reposo a la unidad.

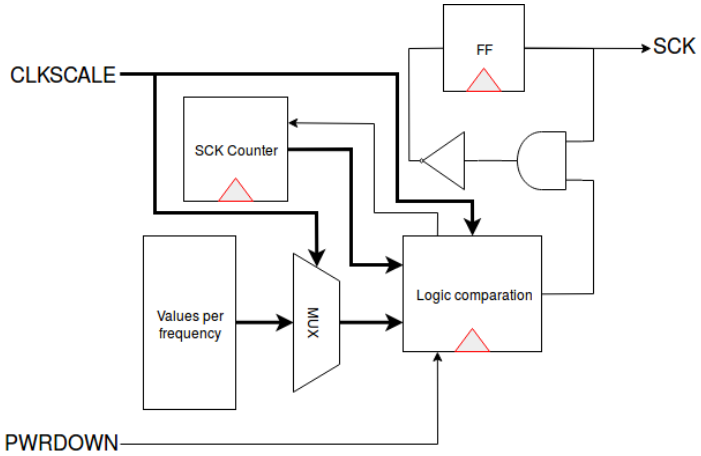

Figura 11: Diagrama de bloque de *SCK Generator*.

En la tabla 4 se muestran los valores de suma para el divisor de frecuencia para generar una frecuencia específica y la tasa de transmisión de información por cada frecuencia. La frecuencia de operación máxima del reloj SCK se determina a partir del mínimo de ciclos de reloj necesarios para configuración interna del módulo del reloj de sistema. El reloj de sistema es 20 MHz.

Tabla 4: Valores de suma para el divisor de frecuencia para generar una frecuencia específica y la tasa de transmisión de información por cada frecuencia.

Frecuencia SCK	Valor de suma a alcanzar	Tasa de transmisión
2.5 MHz	8	156.25 Kb/s
1.25 MHz	16	78.125 Kb/s
625 kHz	32	39.06 Kb/s
312.5 kHz	64	19.53 Kb/s

4. *Decoder/Coder Word*

Este módulo tiene como función generar la decodificación o codificación de la palabras a recibidas o a enviar en el bus de interconexión del microprocesador del proyecto “Procesador RISC-V en HV para aplicaciones médicas”. La palabra tiene una composición definida, la cual se observa en la tabla 5.
Además en la tabla 6 se observa la composición de las palabras y la instrucción correspondiente a cada palabra.

Tabla 5: Estructura de palabra a enviar/recibir del bus de interconexión del proyecto "Procesador RISC-V en HV para aplicaciones médicas"

Tamaño(bytes)	1	1	1	3	4
	Destino	Fuente	Código	Address	Data

Tabla 6: Palabras a enviar/recibir del bus de interconexión, composición e instrucción asociada.

	Index	Origen	Destino	Address	Data	Instrucción
0	6'h14	0x00	0x01	24'hXXXXXX	32'hXXXXXXXX	Inicio de bootstrap.
1	6'h18	0x01	0x00	24'hXXXXXX	32'hXXXXXXXX	Fin de bootstrap.
2	6'h10	0x00	0x01	ADDRESS	DATA	Escritura de 4 bytes.
3	6'h11	0x00	0x01	ADDRESS	{16'hXXXX,DATA}	Escritura de 2 bytes.
4	6'h13	0x00	0x01	ADDRESS	{24'hXXXXXX,DATA}	Escritura de 1 byte.
5	6'h00	0x00	0x01	ADDRESS	32'hXXXXXXXX	Lectura de 4 bytes.
6	6'h01	0x00	0x01	ADDRESS	32'hXXXXXXXX	Lectura de 2 bytes.
7	6'h03	0x00	0x01	ADDRESS	32'hXXXXXXXX	Lectura de 1 byte.
8	6'h10	0x01	0x00	{28'hXXXXXXX,4'bXX00}	DATA	Respuesta de lectura 4 bytes.
9	6'h11	0x01	0x00	{28'hXXXXXXX,4'bXX00}	{16'hXXXX,DATA}	Respuesta de lectura 2 bytes.
10	6'h13	0x01	0x00	{28'hXXXXXXX,4'bXX00}	{24'hXXXXXX,DATA}	Respuesta de lectura de 1 byte.

4.1. *Control Top SPI Bootloader*

Este módulo se encarga de identificar la instrucción proveniente del bus de interconexión, generar las señales de control para realizar dichas operaciones y generar las señales de control para generar la palabra de salida hacia el bus. En la figura 12 se muestra la descripción de los estados de la máquina de estados. Esta se puede dividir en cuatro grandes secciones de operación: *identificación*(estados 0 al 4), *bootloader*(estados 5 al 10), *escritura* (estados 11 al 16) y *lectura*(estados 17 al 20).
En cada estado se realizan las operaciones pertinentes para ejecutar las distintas instrucciones esperadas. En la sección de operación de *identificación* se esta realizando la evaluación de si ingresa un nueva dato a la FIFO. En la sección de operación de *bootloader* se realiza la operación de carga del programa de instrucciones. En la sección de *escritura* se realiza la operación de escritura a la memoria dada una dirección específica y el dato. Finalmente, la sección de operación de *lectura* realiza la lectura de un dato en la memoria IS25WP032D dada una dirección específica.

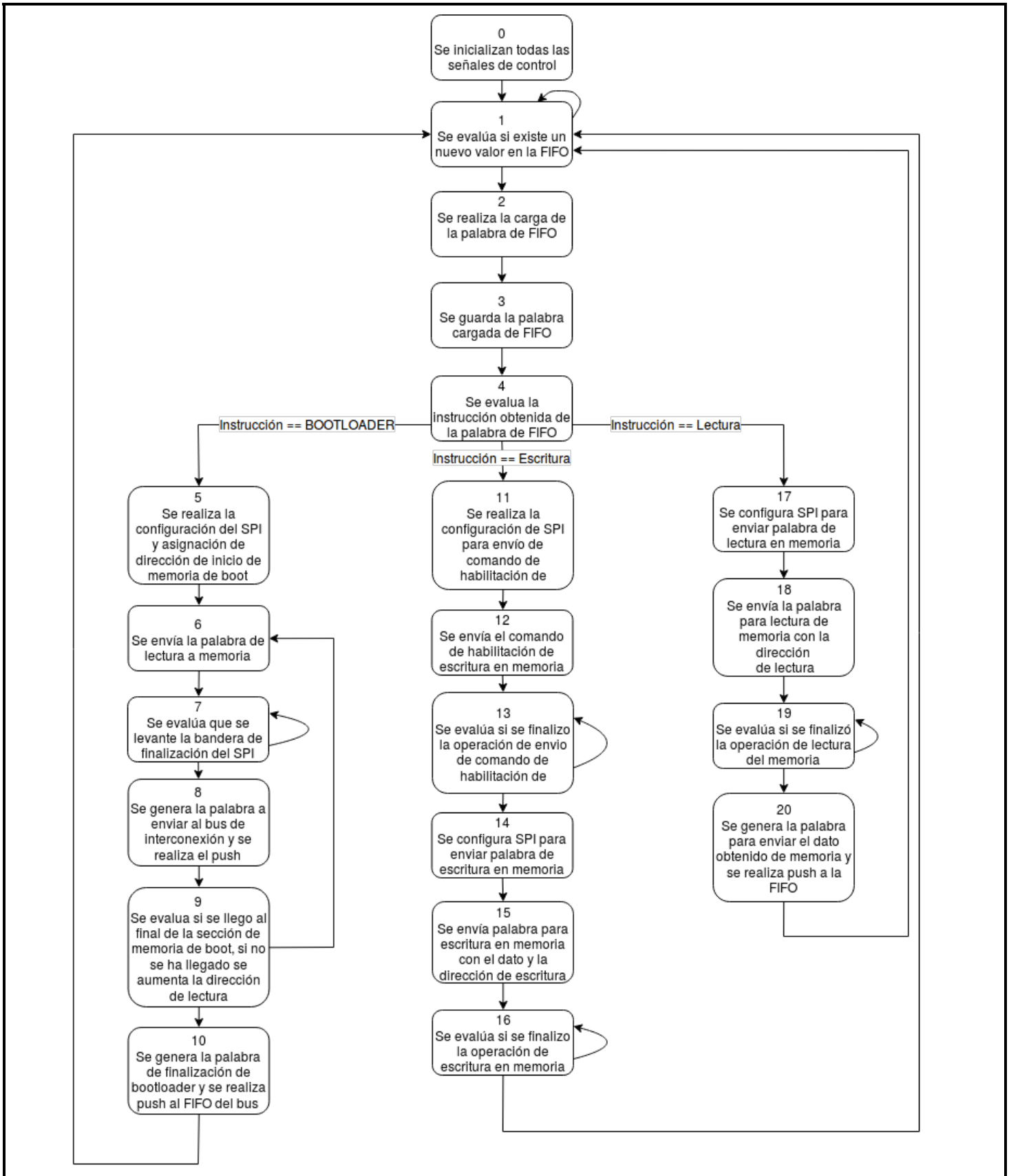

Figura 12: Descripción de estados de la máquina de estados de *Control Top SPI Bootloader*.

5. Flujo de diseño de la unidad SPI Bootloader en herramientas EDA de Synopsys

Como se observo en la figura 2, se requiere un flujo de diseño para llevar la unidad SPI Bootloader de una idea a un circuito fabricable. Para ello es necesario realizar una descripción del diseño en un lenguaje de descripción de hardware, para posteriormente realizar las distintas simulaciones y síntesis necesarias.
El diseño RTL es el primer paso en el flujo de diseño. Este permite realizar una descripción en hardware de la unidad a implementar. El siguiente paso es realizar la síntesis lógica y su correspondiente verificación mediante simulaciones y los reportes que generan las herramientas. Finalmente, se realiza la síntesis física de la unidad y su correspondiente verificación.

5.1. Diseño RTL de SPI Bootloader

El diseño RTL de la unidad SPI Bootloader se realizó a partir del diagrama modular observado en la figura 6. Se generaron dos archivos: el primero contiene la descripción de la unidad encargada de la comunicación por SPI y el segundo contiene la instanciación de la unidad de comunicación SPI y la máquina de estados de control general de la unidad. En la figura 13 se logra observar dicha relación de archivos.

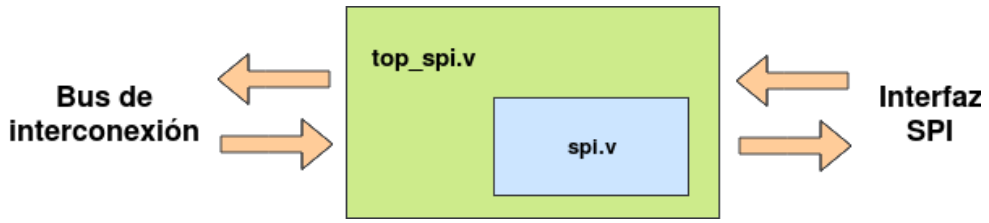

Figura 13: Relación de archivos Verilog de descripción de SPI Bootloader.

Ambos códigos se encuentran parametrizados, es decir, contienen una variable local constante que permite definir atributos del módulo que pueden ser modificados al llamar la unidad en otro módulo. Es importante conocer esos valores, dado a que estos les da adaptabilidad al diseño de una forma sencilla. En la tabla 7 se muestran los valores parametrizados de los módulos generados.
Los diseños RTL deben ser verificados previo a realizar las síntesis lógica y física. Esto permite encontrar errores en la implementación y tener un marco de referencia para verificación post síntesis. La primera verificación se realiza con simulaciones a partir de *testbench* compilados en la herramienta VCS de Synopsys. Se realizan tres pruebas principales al archivo *top* de la implementación del SPI Bootloader.
La primera prueba es que reciba la instrucción de *bootloader* y realice la operación. En la figura 14 se muestra la simulación obtenida. Se le determina al módulo que el programa de instrucciones es de solo 16 bytes. En la figura 14.a se muestra la simulación completa, donde se realiza la lectura de 4 bytes por operación de comunicación con la memoria. En la figura 14.b se observa la operación de comunicación con memoria de solo 4 bytes. Aquí podemos corroborar que las señales del interfaz SPI (*MOSI*,*MISO*, *SCK* y *SC*) tienen un comportamiento similar al descrito en la figura 5. Finalmente, en la figura 14.c se observa cuando se envía el dato recuperado de memoria a la FIFO del bus de interconexión.

Tabla 7: Parámetros de los módulos en Verilog para la implementación de SPI Bootloader.

top_spi		spi	
Parámetro	Descripción	Parámetro	Descripción
FIFO_DATA	Valor de la cantidad de bits de la palabra esperada de la FIFO.	DATA_IN	Valor de la cantidad de bits de la palabra de entrada al SPI.
INDEX	Valor de la cantidad de bits del código de instrucción.	DATA_OUT	Valor de la cantidad de bits de la palabra de salida del SPI.
ORIGEN	Valor de la cantidad de bits del código de periférico de origen.		
DESTINO	Valor de la cantidad de bits del código de periférico de destino.		
ADDR	Valor de la cantidad de bits del valor de dirección.		
DATA	Valor de la cantidad de bits del valor de datos.		
START_ADDR	Dirección de inicio del programa de instrucciones en memoria IS25WP032D.		
END_ADDR	Dirección final del programa de instrucciones en memoria IS25WP032D.		

La segunda prueba es que el módulo reciba la instrucción de escritura. En este caso la escritura de 2 bytes. En la figura 15 se muestra la simulación obtenida. De igual manera se observa que dicho resultado cumple las especificaciones mostradas en la figura 4 en cuanto a comportamiento de las señales.

Finalmente, la figura 16 muestra la prueba donde el módulo recibe la instrucción de lectura de 4 bytes. Como en los casos anteriores, el comportamiento de las señales cumple las especificaciones planteadas en la figura 5.

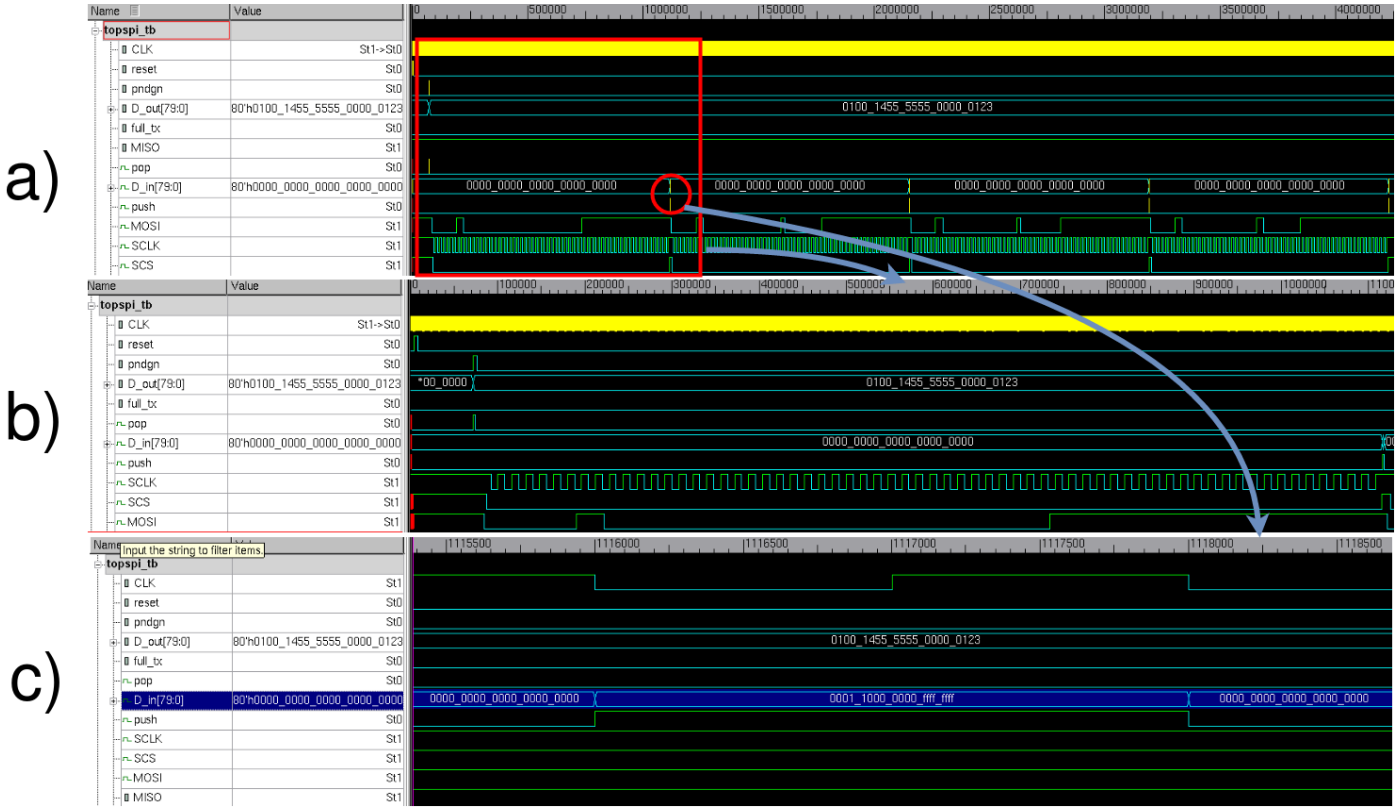

Figura 14: *Testbench* de la instrucción de *bootloader* de la implementación del SPI Bootloader.

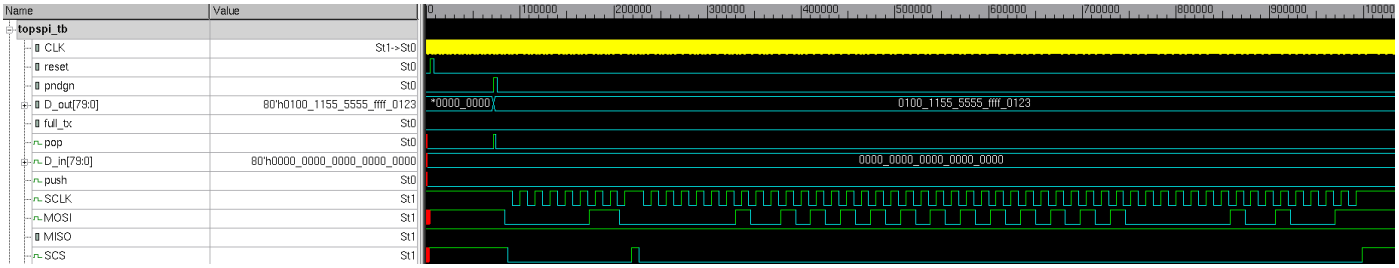

Figura 15: *Testbench* de la instrucción de escritura de la implementación del SPI Bootloader.

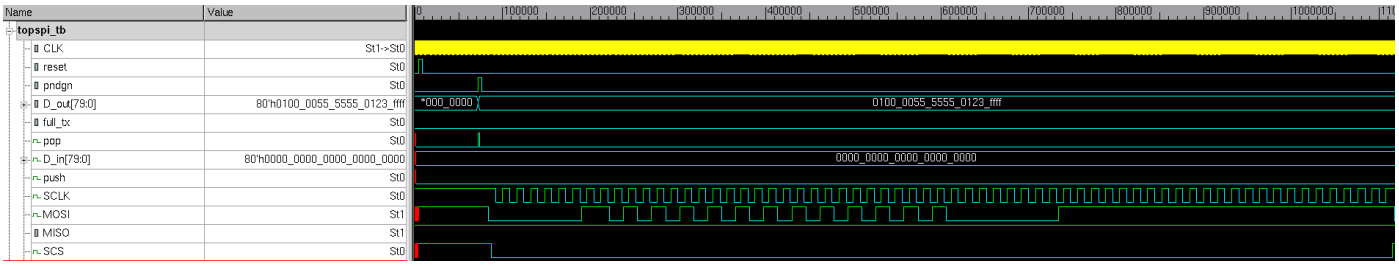

Figura 16: *Testbench* de la instrucción de lectura de la implementación del SPI Bootloader.

5.2. Síntesis lógica de SPI Bootloader

Como se ha descrito, la síntesis lógica convierte el diseño descrito en HDL a un mapeo de nodos a nivel de compuertas en una tecnología especifica. Para realizar la síntesis lógica se realizo una estructura

de directorios de trabajo, donde se pudieran separar los distintos archivos necesarios o generados por dicha síntesis. En la figura 17 se muestra la estructura del directorio de trabajo.

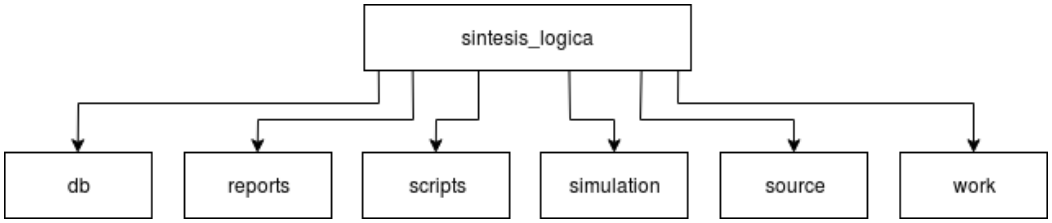

Figura 17: Estructura del directorio de trabajo para la síntesis lógica de SPI Bootloader.

Cada directorio esta destinado a contener archivos con contenido específico. A continuación se detallan los contenidos de las carpetas:

- ***db***: En esta carpeta se almacenan los archivos generados por la síntesis como lo son el mapeo de nodos (.v), la bases de datos sintetizadas (.ddc) y las restricciones de diseño de Synopsys (.sdc).
- ***reports***: En esta carpeta se almacenan los reportes generados por la síntesis lógica de área, potencia, listado de celdas, temporizado y *Quality of Result*. Este último es una sintesis de los resultados de los demás reportes.
- ***scripts***: Este directorio contiene los *scripts* necesarios para realizar las síntesis lógica. Los *scripts* permiten realizar el proceso de manera más rápida, en relación a realizar dichos pasos en el interfaz gráfico o introduciendo uno a uno cada comando. Hay dos *scripts* para esta síntesis; el primero tiene un trazado general de los comandos de síntesis y el segundo las limitaciones de temporizado.
- ***simulation***: Esta carpeta contiene los archivos necesarios para generar la simulación post síntesis lógica.
- ***source***: Este directorio contiene los archivos de descripción de hardware en Verilog (.v).
- ***work***: Este directorio contiene algunos archivos auxiliares para generados en la síntesis lógica.

Los *scripts* utilizados, como se menciono, contiene la secuencia de comandos que permite realizar la síntesis lógica y las limitaciones de temporizados del diseño. En la figura 18 se muestra el flujo del diseño y las limitantes de temporizado.

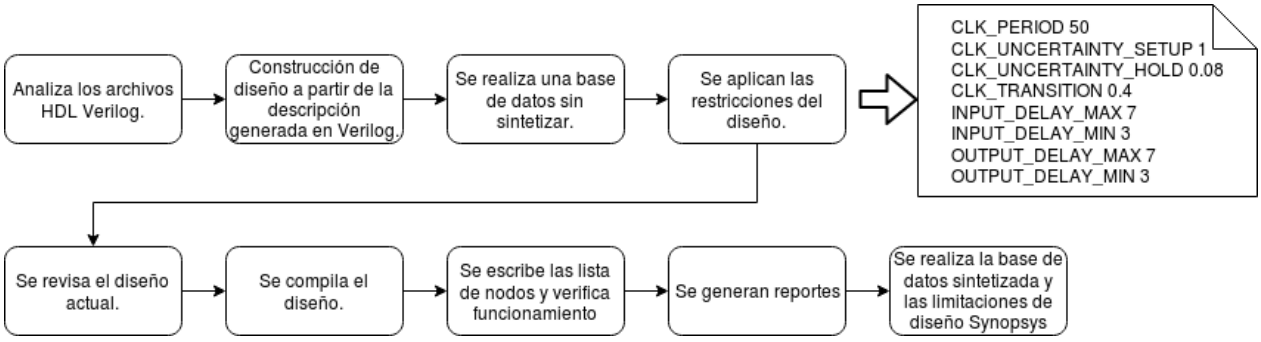

Figura 18: Flujo de scripts y limitaciones de diseño para la síntesis lógica del SPI Bootloader.

Posterior a realizar la síntesis lógica, se realizo simulaciones al mapeo de nodos generados para las tres pruebas hechas a la descripción de hardware original. En los tres casos la simulación arroja los mismos resultados vistos en 14, 15 y 16.
También se debe valorar los resultados de los reportes, los cuales dan una línea preliminar de la implementación del diseño. Además el reporte de temporizado permite conocer si el diseño cumple con las restricciones y si tienes problemas de slack. En la tabla 8 se muestra un resumen de los datos obtenidos en los reportes. A partir de los datos obtenidos se deduce que no existen problemas de temporizado y que el mayor consumo de área se da por celdas no combinacionales.

Tabla 8: Resumen de datos de los reportes de la síntesis lógica del SPI Bootloader.

Reporte	Criterio	Valor
Área	Número de puertos	291
	Número de celdas	1571
	Número de celdas combinacionales	1111
	Número de celdas secuenciales	459
	Área combinacional	15736.72 μm^2
	Área no combinacional	24017.21 μm^2
	Área total	42054.41 μm^2
Potencia	Potencia dinámica	13.86 μW
	Potencia de fuga	3.77 nW
	Potencia interna	0.7419 mW
	Potencia total	0.7558 mW
Temporizado	Slack de peor ruta	9.46ns

5.3. Síntesis física de SPI Bootloader

La síntesis física realiza el posicionamiento y enrutamiento de las celdas a partir del mapeo de nodos generado en la síntesis lógica. Para realizar dicha síntesis se utilizo la estructura de directorios de trabajo que se muestra en la figura 19, muy similar a la utilizada en la síntesis lógica.

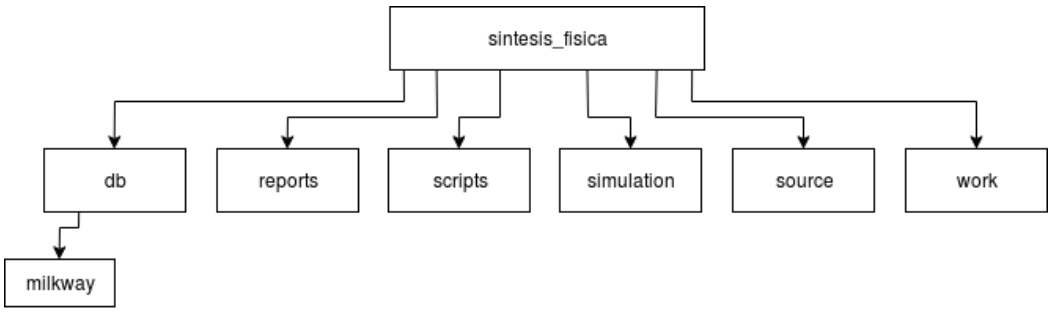

Figura 19: Estructura del directorio de trabajo para la síntesis física de SPI Bootloader.

A pesar de poseer una estructura de directorio similar a la de la síntesis lógica, los archivos necesarios para la síntesis física son distintos. A continuación se hace una pequeña descripción del contenido de cada directorio.

- ***db***: En este directorio se almacenan los archivos finales generados por la síntesis física en formatos .v(netlist a nivel de compuertas), .sdc (especificaciones de temporizado), .spef(capacitancias

parásitas), entre otros (cada archivo contiene información relevante a la síntesis física). Además, acá se crea el ***milkway*** el cual consiste en una seria de librerías que contienen información acerca del diseño.

- ***reports***: En esta carpeta se almacena los reportes generados por la herramienta *IC Compiler* sobre la síntesis física.
- ***scripts***: En este directorio se encuentra el *script* necesario para ejecutar la sintesis física. Para el caso de la síntesis física implementada solo se utiliza un script.
- ***simulation***: En esta carpeta se almacena los archivos post síntesis física y el *testbench* para realizar la simulación comportamental post síntesis.
- ***source***: En este directorio se almacenan los archivos base para generar la síntesis física. Dichos archivos son los archivos generados post síntesis lógica, los cuales deben ser trasladados a este directorio de la síntesis física.
- ***work***: Este directorio contiene algunos archivos auxiliares para generados en la síntesis física.

El *script* necesario para realizar la síntesis física contiene instrucciones para realizar el flujo básico de construcción de *layout* y para guiar a la herramienta a través de las necesidades de la implementación, con el fin de obtener un *layout* optimizado. En la figura 20 se observa el flujo de comandos que se encuentran en el *script* utilizado.

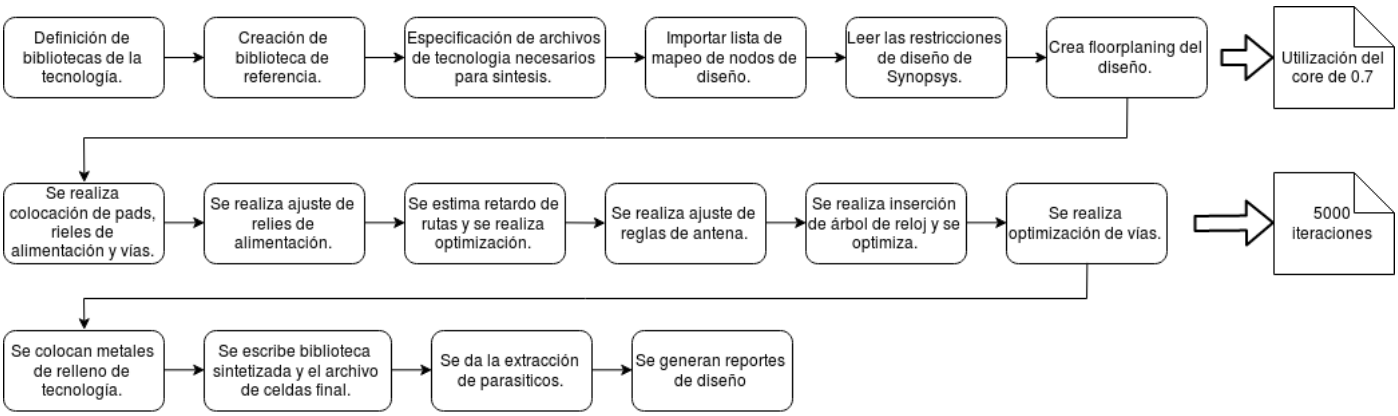

Figura 20: Flujo del scripts para la síntesis física del SPI Bootloader.

Una vez realizada la síntesis física se realizaron las simulaciones en los tres casos de interés con capacitancias parásitas, obteniendo resultados casi idénticos a los mostrados en las figuras 14, 15 y 16. A partir de lo anterior se determina que el diseño cumple con el comportamiento esperado en su implementación en layout. Posteriormente, se realiza una revisión de los valores finales obtenidos en los reportes, los cuales marcan la pauta sobre las características finales del layout. En la tabla 9 se muesta un resumen de los datos del reporte.
Es posible ver como los valores de los reportes varían de la síntesis lógica a la síntesis física. Esto debido a que en la primera síntesis se realiza un mapeo de nodos y no toma en cuenta muchas variables físicas que si son de importancia en la síntesis física, lo cual provoca esa variabilidad. Además, en cada síntesis se realizan optimizaciones distintas, las cuales cambian en proporción distinta el ahorro de área, celdas y potencia.
Analizando los valores, se puede observar como la cantidad de celdas disminuye en la síntesis física. En

Tabla 9: Resumen de datos de los reportes de la síntesis física del SPI Bootloader.

Reporte	Criterio	Valor
Área	Número de puertos	170
	Número de celdas	928
	Número de celdas combinacionales	630
	Número de celdas secuenciales	297
	Área combinacional	15812.66 μm^2
	Área no combinacional	22606.06 μm^2
	Área total	40733.38 μm^2
Potencia	Potencia dinámica	0.212 mW
	Potencia de fuga	4.65 nW
	Potencia interna	0.5434 mW
	Potencia total	0.7554 mW
Temporizado	Slack de peor ruta	5.49ns

la figura 21 se muestra una comparativa entre la cantidad total de celdas, las celdas combinacionales y las celdas no combinacionales en cada síntesis. En resumen, los resultados de la síntesis física muestra que se disminuyo el numero de celdas.

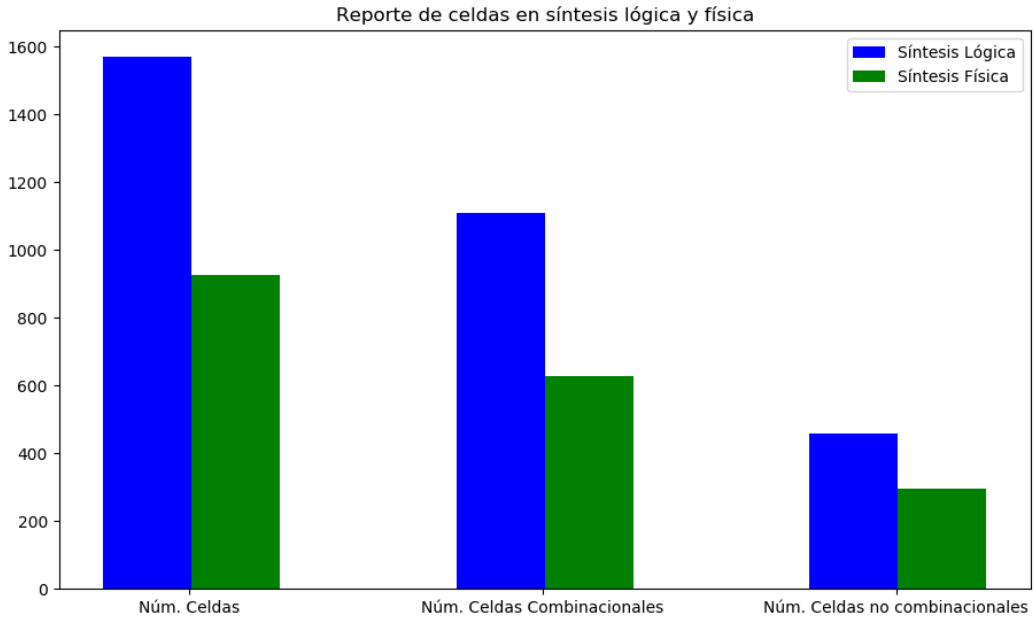

Figura 21: Comparación de los reportes de celdas de las síntesis del SPI Bootloader.

Un análisis similar se puede realizar en el aspecto de área, donde los valores de las síntesis física también disminuyen. En la figura 22 se observa una comparativa entre la cantidad total de área, el área combinacionales y el área no combinacionales en cada síntesis.
Finalmente, en el apartado de potencia se obtienen valores casi idénticos. En cuanto al *slack*, se obtiene un valor positivo, lo que permite concluir que el diseño no tiene problemas de temporizado.

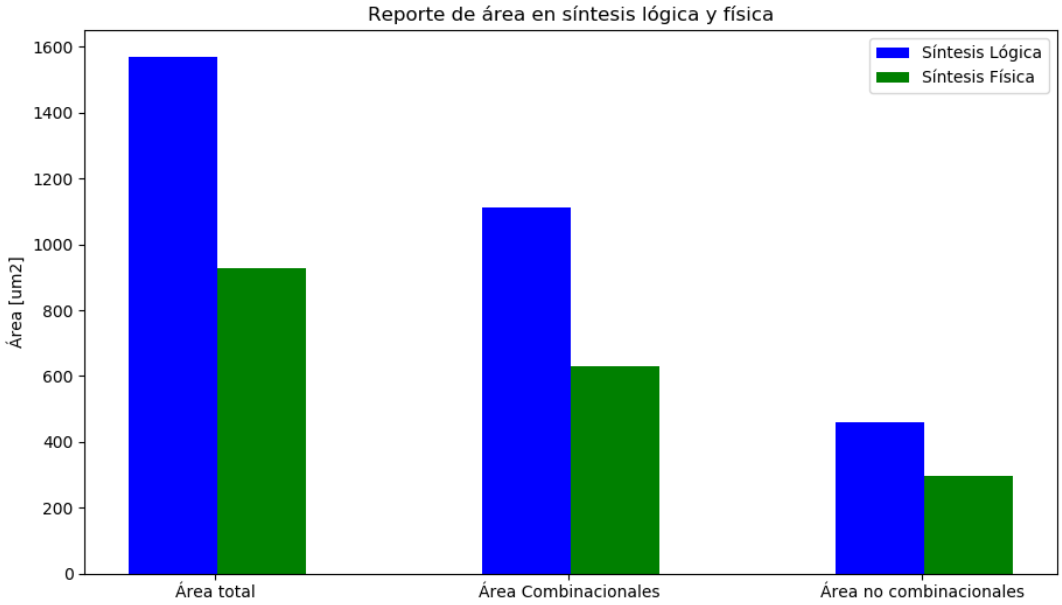

Figura 22: Comparación de los reportes de área de las síntesis del SPI Bootloader.

Otro aspecto que se puede evaluar es las dimensiones geométricas finales del *layout*. En la figura 23 se observa el *layout* final del SPI Bootloader. Dado que se dimensiona en un área cuadrada y con el resultado del área total de los reportes, se puede estimar que las dimensiones del layout son 201.82 x 201.82 μm.

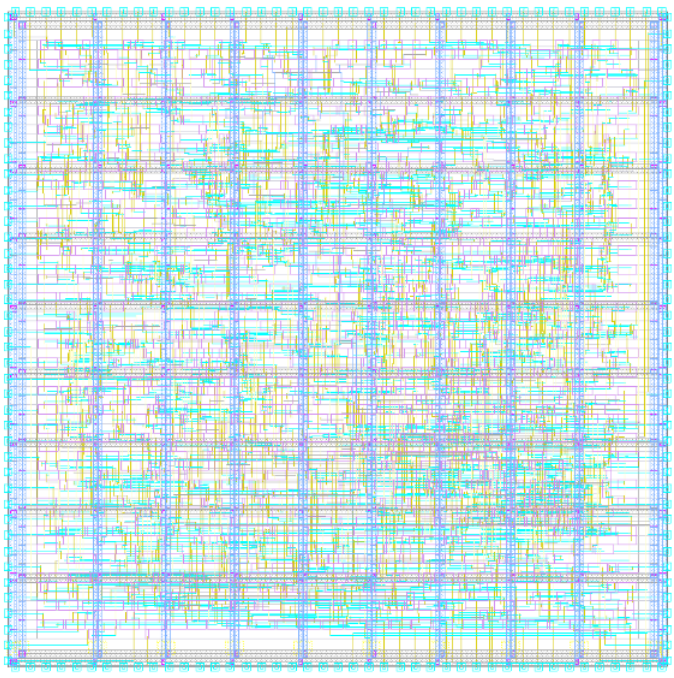

Figura 23: Layout del SPI Bootloader.

La herramienta *IC Compiler* también permite obtener un análisis térmico del layout final. En la figura 24 se muestra el análisis para el diseño implementado. Dicho análisis también permite intuir los sectores que consumirán mayor potencia.

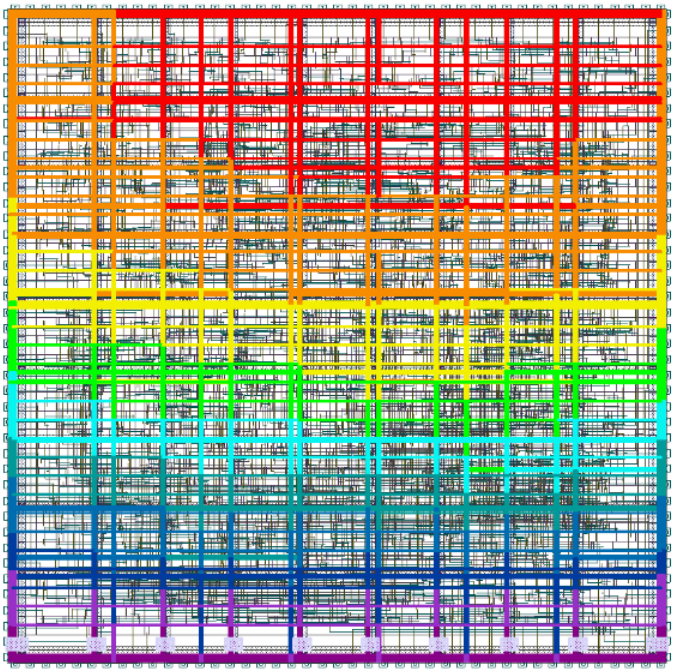

Figura 24: Análisis térmico del layout del SPI Bootloader.

6. Conclusiones

- El protocolo SPI no tiene una única forma de implementación y a menudo se diseña para una aplicación en específico. Sin embargo existen pautas principales especificadas por distintas empresas, que permite tener una linea principal de diseño. En el caso del SPI Bootloader se utiliza como base las especificaciones del *"KeyStone Architecture Serial Peripheral Interface (SPI)"* de la empresa Texas Instrument.

- Es necesario crear un sincronizador entre el reloj del sistema y el reloj generado para la comunicación SPI. El sincronizador permite trabajar bajo un solo dominio de reloj y realizar las operaciones necesarias de cada reloj.

- Tener un nivel de jerarquía de los archivos que describen el RTL permite mantener un mayor orden de diseño y permite realizar correcciones de manera más sencilla.

- En la implementación actual del SPI Bootloader se realizaron simulaciones para los tres casos de interés, sin embargo es necesario realizar una mayor cantidad de simulaciones que puedan llevar el diseño a un mayor nivel de estrés para comprobar su funcionalidad.

- Los resultados entre las síntesis física y la síntesis lógica pueden ser distintos debido a que en la primera síntesis se realiza un mapeo de nodos y no toma en cuenta muchas variables físicas que si son de importancia en la síntesis física, lo cual provoca esa variabilidad. Además, en cada síntesis

se realizan optimizaciones distintas, las cuales cambian en proporción distinta el ahorro de área, celdas y potencia.

Referencias

[1] Weste, N.& Harris, D. (2010). *CMOS VLSI Design: A Circuits and Systems Perspective* Boston: Addison-Wesley, Cuarta Edición.

[2] Leens Frédéric. (2009). *An Introduction to I2C and SPI Protocols*. IEEE Instrumentation and Measurement Magazine.

[3] Valverde, J. (2016). *Tutorial para flujo de diseli de circuitos integrados digitales con tecnología IBM 8RF (CMOS 0.13 μm)*. DCILAB Instituto Tecnológico de Costa Rica.

[4] Integrated Silicon Solution. (2018). *SERIAL FLASH MEMORY WITH 133MHZ MULTI I/O SPI QUAD I/O QPI DTR INTERFACE*. DATA SHEET.