***Laboratorio 8 – Memoria Externa QSPI***

Objetivos del laboratorio son conocer:

* Saber que es una memoria
* Cargar información por SPI a un FPGA

1. Memoria Externa

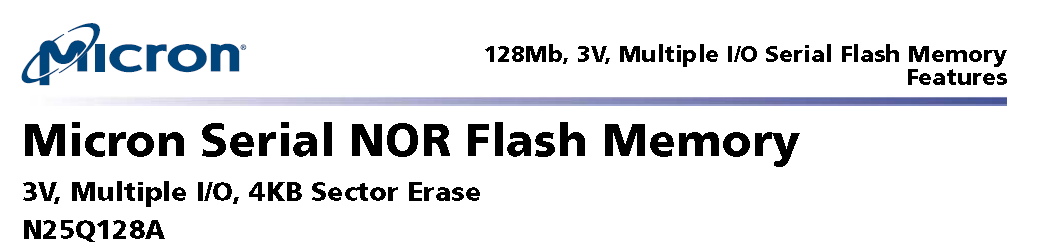
*1.1 – Introducción*

En este laboratorio utilizaremos la memoria externa de la Arty. Típicamente esta memoria es utilizada para, en el caso de un FPGA, programar la tarjeta. En todos los laboratorios anteriores al programar la tarjeta y desconectar la alimentación, la memoria de programación se pierde. Normalmente esta memoria QSPI que utilizaremos los FPGA lo utilizan (y también se puede utilizar el concepto para otros problemas) para inmediatamente al encender la tarjeta buscar la configuración en esta memoria flash externa.

La memoria QSPI de la Arty es de 16MB aproximadamente y la configuración de la Arty toma aproximadamente 2MB lo que queda en 14MB aproximadamente para alojar datos. La idea final es utilizar como utilizar una memoria no volátil en sus proyectos.

*1.2 – Memoria Flash N25Q128A*

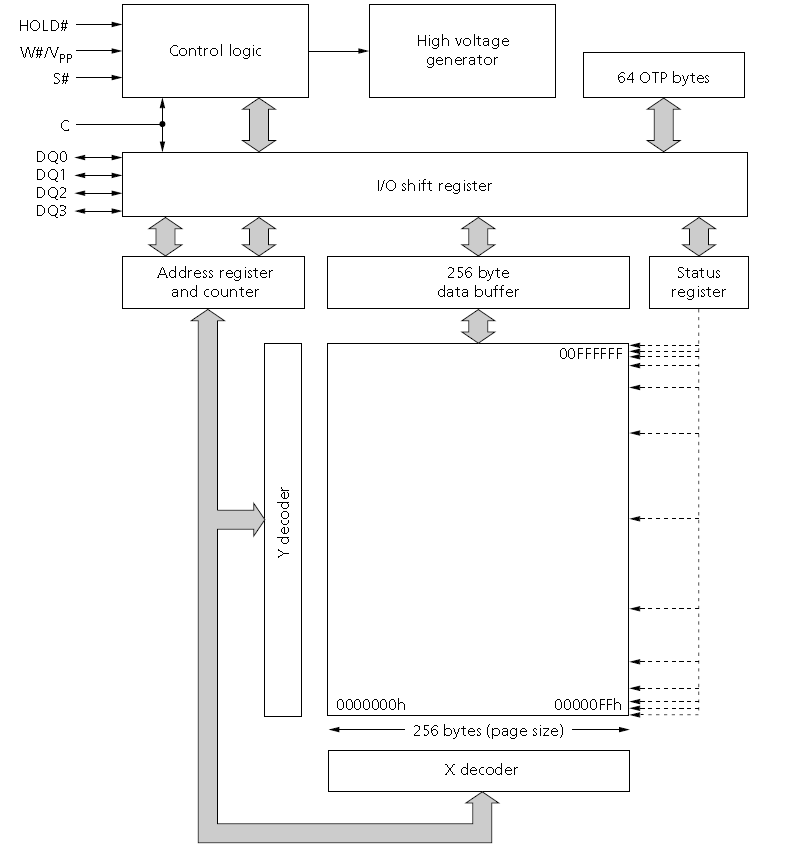
La memoria de la tarjeta Arty es una memoria de 3V de operación, no volátil que posee una arquitectura NOR de confección. Existen memorias flash basadas en NAND y NOR y su precio varía por prácticamente por velocidad de acceso, las memorias NAND son más baratas que las memorias NOR usualmente. Adicionalmente esta memoria tiene sectores de borrado de 4Kb lo que significa que podemos hacer un borrado masivo de cada 4Kb en vez de borrar byte a byte.



*Figura 1.1. Memoria Flash. La memoria flash de la Arty es una memoria NOR que opera a 3V*

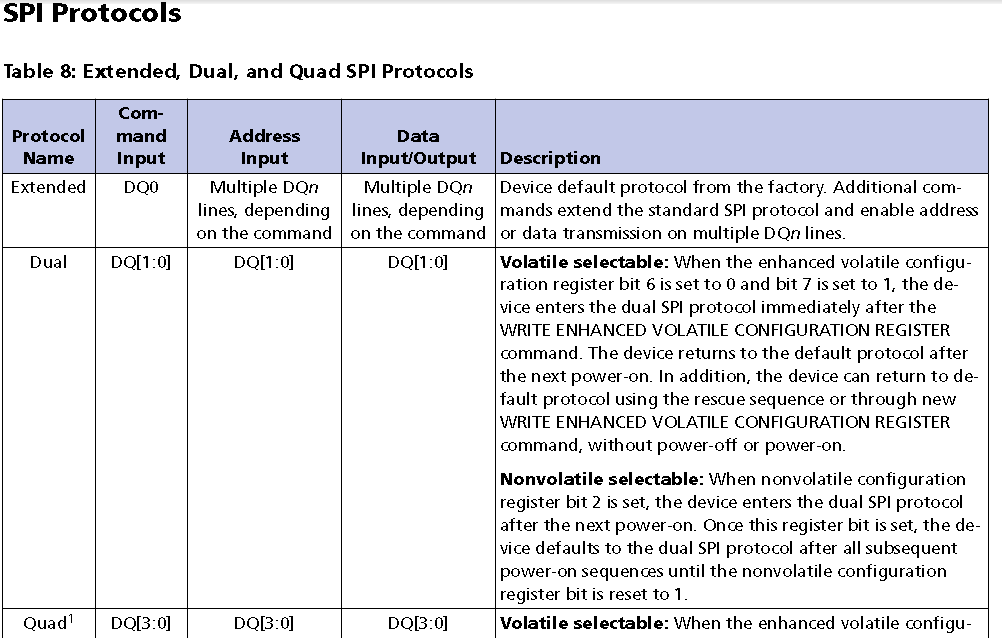
Si nos vamos a la página 13 del manual de la memoria flash <https://bit.ly/2SVhmNY> observaremos que tenemos señales como HOLD, W#, S, C, DQ[3:0] y que prácticamente utilizamos estas señales de control para realizar lecturas y escrituras, por ejemplo, escribiremos el registro que queremos leer y enviar la información y los registros de control procesarán dándonos de vuelta un un registro final el valor.

También como mencionamos tenemos la capacidad de realizar lecturas varias consecutivas o en bloque del dispositivo.

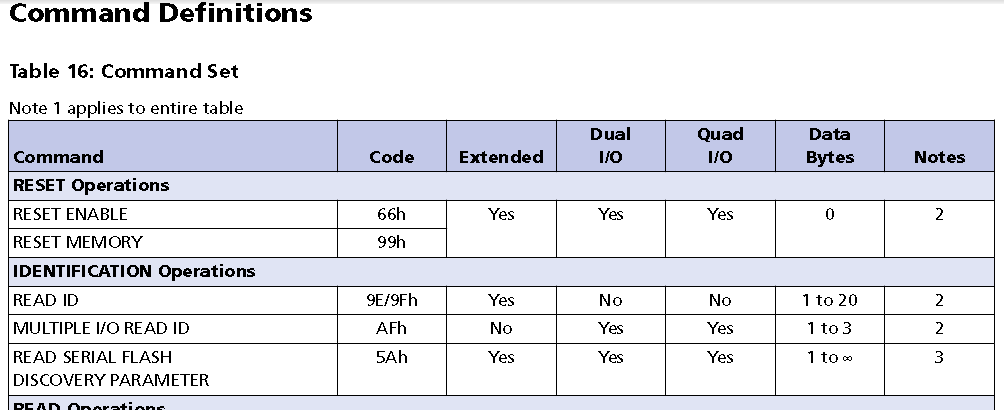


*Figura 1.2. Diagrama de bloques. Diagrama de bloques de la memoria flash QSPI.*

En la página 20 podemos observar los protocolos soportados. Nos da una descripción breve de las señales y el protocolo, por ejemplo, el protocolo SPI puede darse datos por medio de una interfaz dual DQ[1:0] o quad DQ[3:0], lo que quiere decir que podemos enviar un comando más completo.



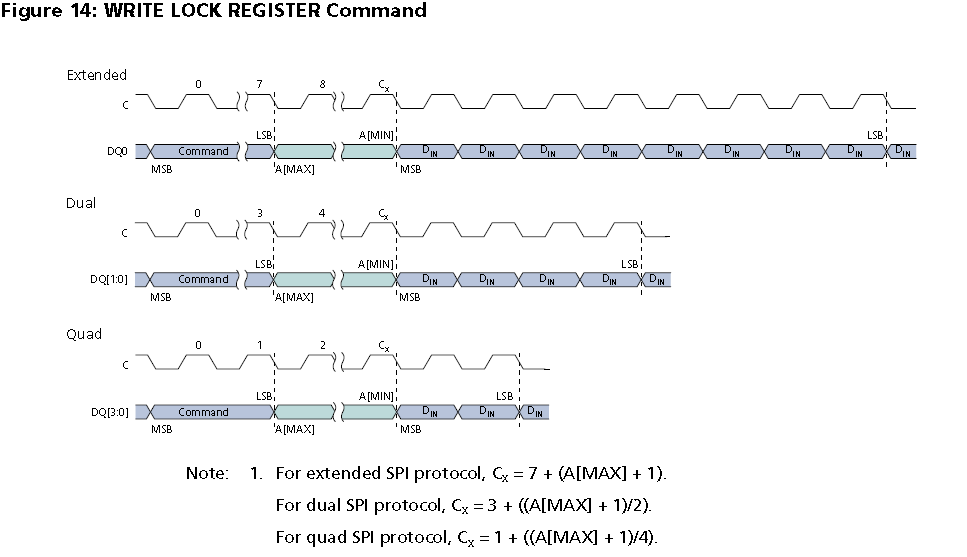
*Figura 1.3. Protocolo SPI de la memoria. Se pueden utilizar la interfaz dual como la quad para ingresar o adquirir datos.*



*Figura 1.4. Definición de Comandos. En la página 29 se muestran los comandos aceptados por el dispositivo.*

Si nos fijamos más a detalle de la lista de comandos observaremos que existe un código, este código representa el comando que queremos enviar, por ejemplo, para habilitar el reset enviaremos el comand 66h (66 hexadecimal), otro dato importante es verificar que el comando sea soportado. Nosotros utilizaremos el modo extendido por lo que observamos en esta columna que sí es soportado el comando; además se muestran otras condiciones como si soporta o no el comando dual o quad y bytes que hay o no que enviar.

Adentrandonos más en la página 36 también observamos los diagramas de tiempos de como tienen que ser enviados los comandos, por ejemplo en la figura inferior se muestra el comando de bloqueo de escritura y su secuencia de bits como debe de darse. Si navegamos a la página 56 veremos por ejemplo la explicación de un comando. El datasheet es extenso pero cuando trabajamos con diferentes integrados esta información es muy necesaria y deben de ser explícitos.



*Figura 1.5. Ejemplo del diagrama de tiempo de la memoria para bloqueo de escritura.*

*1.3 – Protocolo de Comunicación SPI*

Este protocolo es un protocolo embebido dentro de sensores y actuadores, es ampliamente utilizado como lo es la UART, digamos que SPI es un protocolo más moderno. La ventaja de SPI de una UART es que nos permite tener más equipos enlazados serialmente bajo el mismo bus de datos separados por una línea de control y es un bus de alta velocidad. Lo normal es encontrar este protocolo en memorias porque permite transmitir datos de manera muy rápida.

Tenemos 4 señales importantes.

MOSI = Master Out Slave In

MISO = Master In Slave Out

CLK = Reloj Síncrono

CS = Chip Select

Las señales comunes compartidas son MOSI, MISO y CLK. La señal de CS será única para cada integrado, es decir, si son 3, el maestro debe tener CS1, CS2, CS3 y cada señal debe ir a un esclavo, esta señal seleccionará que esclavo quiere cuestionar.

Para datos, ejemplo, si queremos leer, CS estará en estado bajo, enviaremos el stream de bytes acompañado del comando por MOSI y el esclavo responderá al maestro por medio de la línea MISO.

Existen varios modos de SPI:

* El modo estándar, que es solo un simple bit de datos transmitido en cada ciclo de reloj
* Modo dual, que permite transmitir dos bits en un ciclo de reloj
* Modo quad, que permite transmitir cuatro bits en un ciclo de reloj

Obviamente el Quad es 2 veces más rápido que el Dual y 4 veces más rápido que el estándar.

*1.4 – Protocolo de Comunicación SPI Explicado*

Para SPI, el protocolo de comunicación entre un maestro y un esclavo es como sigue. Las líneas representan si la información fluye del maestro al esclavo o visceversa y que debemos hacer para enviar datos.



*Figura 1.6. Ejemplo de comunicación entre un maestro y un esclavo. En este caso hemos dibujado el diagrama con respecto a detección del bit en flanco de subida (esto depende del esclavo).*

La manera de detección del flanco de subida depende del esclavo, se debe de verificar el datasheet para esto. Primeramente para seleccionar el esclavo, como en la figura anterior, el CS se ubica en estado bajo y seguido el maestro envía la secuencia de comandos, luego de enviar el comado y después de unos ciclos de reloj, el esclavo responderá con el grupo de bytes del comando asertado como vimos en la figura anterior.



*Figura 1.7. Conexión de múltiples esclavos a SPI.*

2 – Diseño y Estrategia del Bus SPI en Arty

*2.1 – Introducción*

La interfaz SPI a utilizar la diseñaremos con el Microblaze, sin embargo el microblaze no posee una interfaz SPI. Necesitaremos implementar el SPI utilizando VHDL en la lógica de VHDL e interfacearemos las dos para utilizar la memoria de SPI.

Usaremos también la UART para desplegar la implementación del diseño.

*2.2 – Concepto de Diseño en Diagrama de bloques*

En la figura inferior se muestra el concepto.



*Figura 2.1. Concepto del Diseño del Bus SPI e Interfaz a la PC.*

De la figura anterior, el microblaze lo usaremos como controlador y viene acompañado de sus señales de entrada de Reset y Clk. Implementaremos la lógica de VHDL de SPI y esta nos servirá para “conversar” con la memoria SPI de 16MB. Adicionalmente usaremos una UART para establecer comunicación con la PC y enviar la información de la memoria al computador.

3 – Notas Acerca de la Memoria Externa Flash

*3.1 – Introducción*

Existena algunas consideraciones al momento de escribir, primeramente utilizaremos la zona baja para esto, aquí descargaremos ele archivo de configuración de la memoria. La zona baja en este caso empieza en la posición 0x000000 y aproximadamente toma alrededor de 2MB, representado por el hexadecimal 0x1FFFFF.



*Figura 3.1. Zonificación aproximada de la memoria flash.*

Si notamos en la figura anterior existe una zona que denominamos una “zona de espacio”. Esta zona es necesaria debido a que según el XAPP951, que es un documento de aplicación de Xilinx nos informa que la memoria se escribirá hasta el punto que encuentre un break, por consiguiente, esto puede seguir escribiendo en otro bloque y pasaría de la sección 0x1FFFFF.

Dato importante es que el archivo \*.bit generado debe ser convertido a un archivo \*.mcs y este debe ser alojado en la memoria.

Seguido de la zona de espacio tenemos toda la memoria libre para datos de usuario, aquí alojaremos la data que necesitemos almacenar, así no entrará en conflicto con la configuración que tenemos cargada.

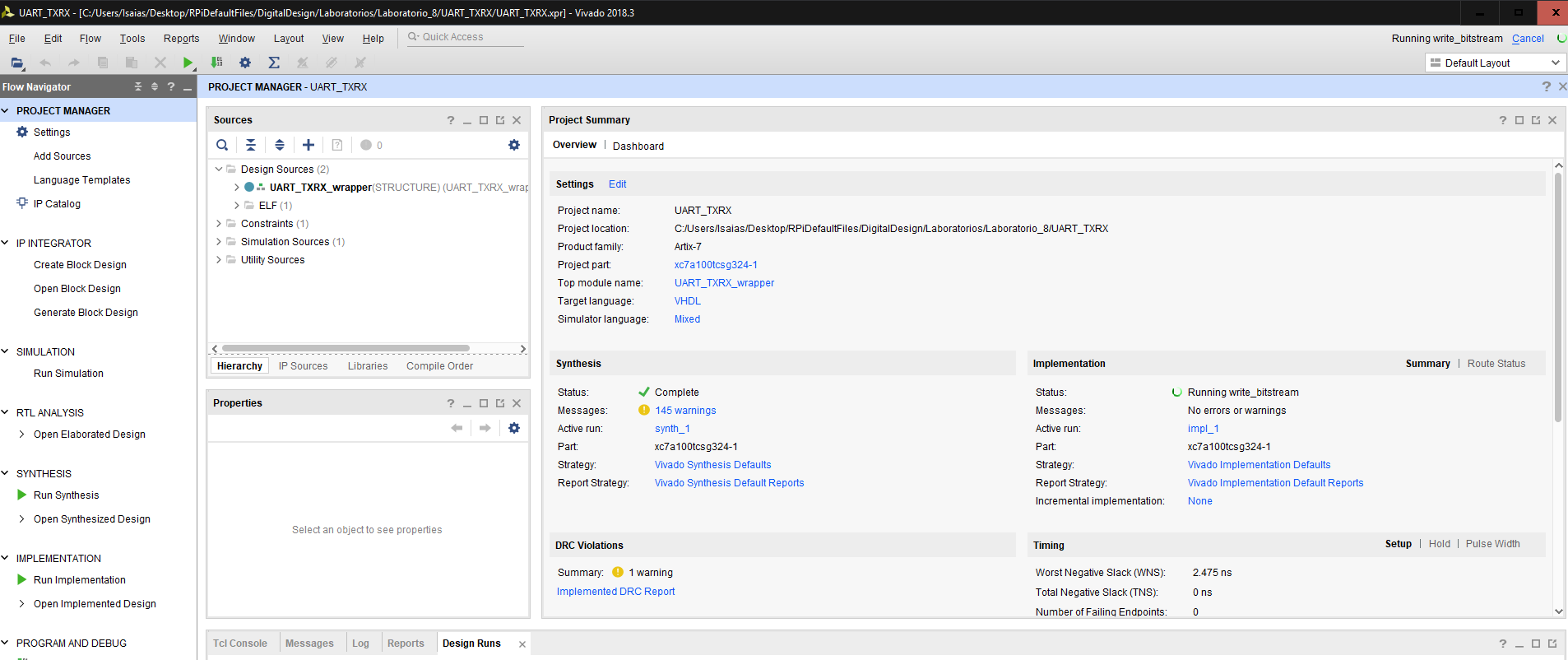
Si tenemos algún problema porque los datos han sido traslapados entre la configuración y los datos de usuario podemos realizar la implementación nuevamente de quemar el mcs en la flash, pero si es un dispositivo que se encuentra en campo lo normal será que no se deje esto de esta manera pues es un error grave y si es un sistema crítico puede tener malas consecuencias.

4 – Generación del Archivo

*4.1 – Generación del Bitstream*

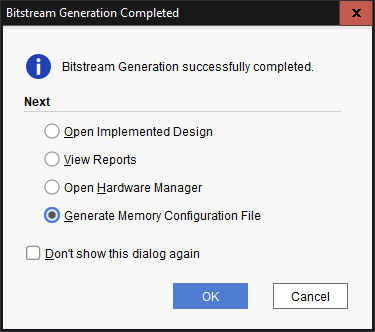
4.1 – Abrir el archivo de proyecto, por ejemplo el del laboratorio 5 o laboratorio 6 que son más completos y de mayor capacidad en memoria.

4.2 – Generar el bitstream como en laboratorios anteriores



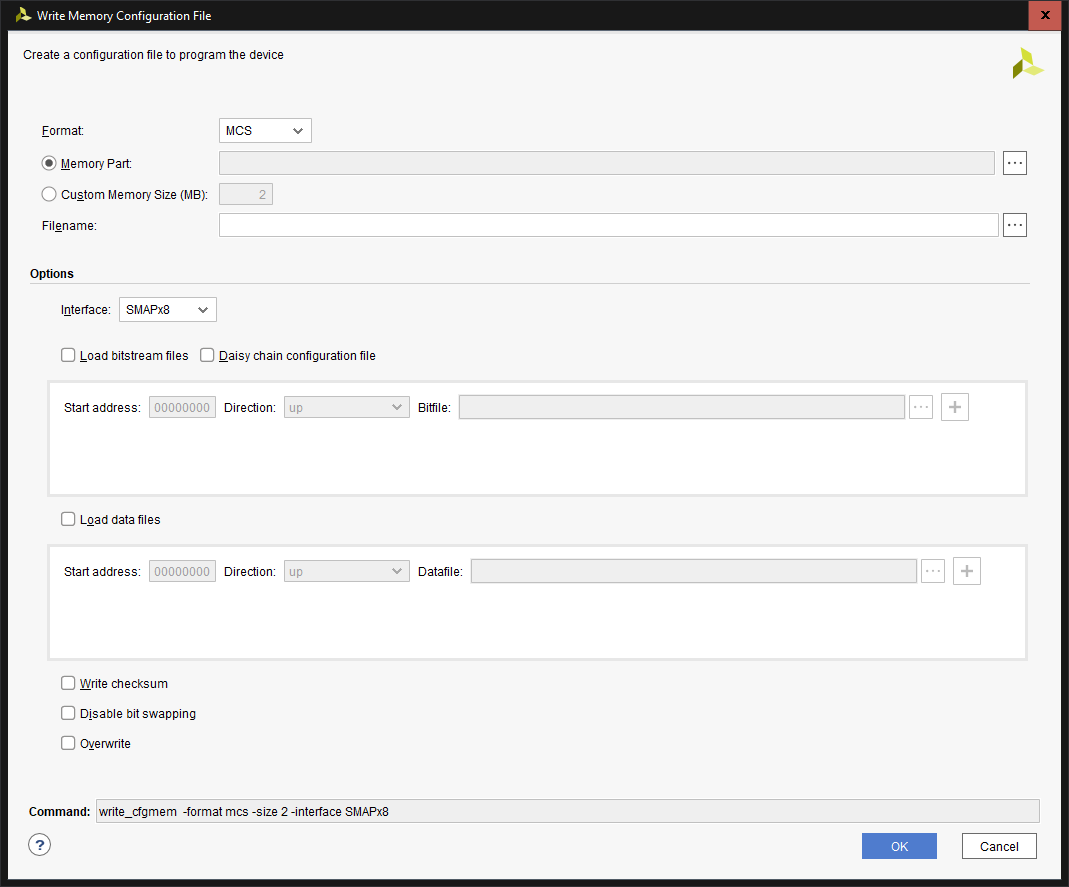
*Figura 4.1 – Generación del archivo bitstream.*

4.3 – Al completarse la generación seleccione y haga siguiente en generar el archivo de memoria



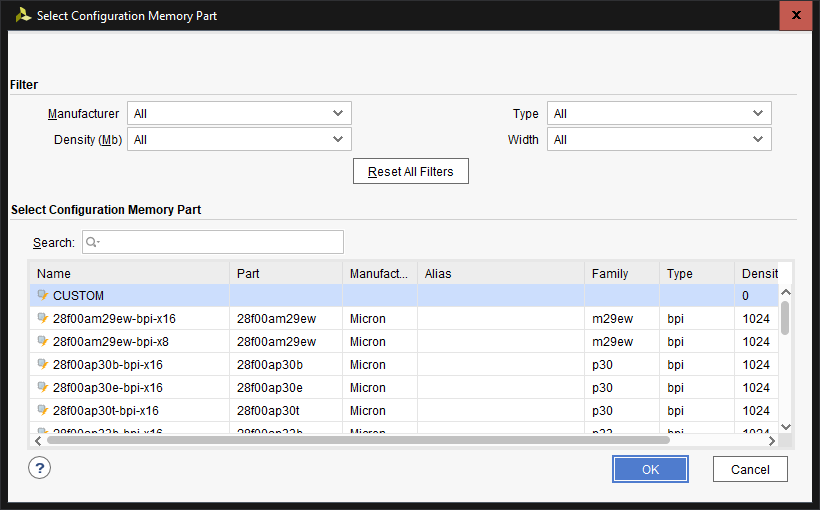
*Figura 4.2 – Seleccione la generación del archivo de memoria.*

4.4 – Seleccione la memoria como se muestra en la siguiente pantalla.



*Figura 4.4 – Ventana de selección de memoria. Haga click en “…”*

4.5 – Como no existe ninguna memoria referente a la que tiene el Arty, si buscamos en la siguiente página, habrá que cancelar estas ventanas y abrir el hardware manager

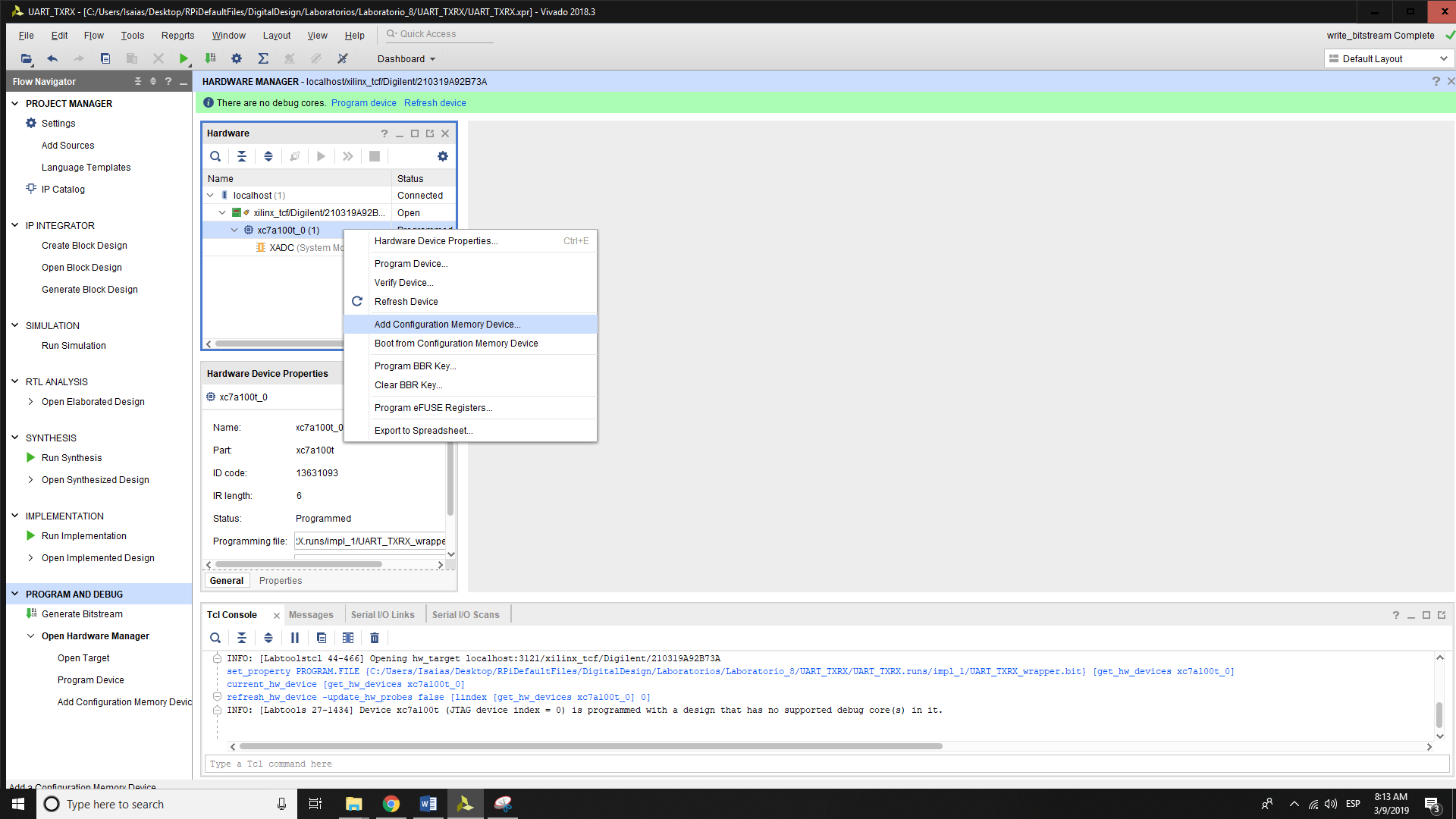


*Figura 4.5 – Ventana de selección de memoria. La memoria deseada no existe*

4.5 – Abrir el Hardware Manager

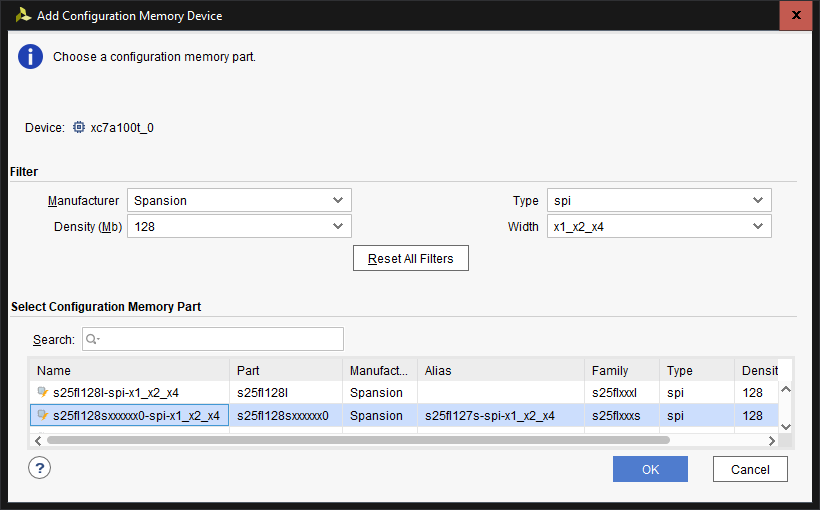
4.6 – Conectarse a la tarjeta de Xilinx

4.7 – Sobre la siguiente ventana seleccionar el dispositivo de memoria a agregar



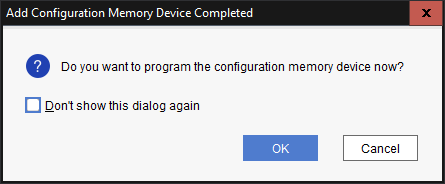
*Figura 4.6 – Agregar el dispositivo de memoria*

4.8 – Seleccione el dispositivo de memoria de la Arty, que es un SPI de 3.3V de Spansion de 128Mb de densidad



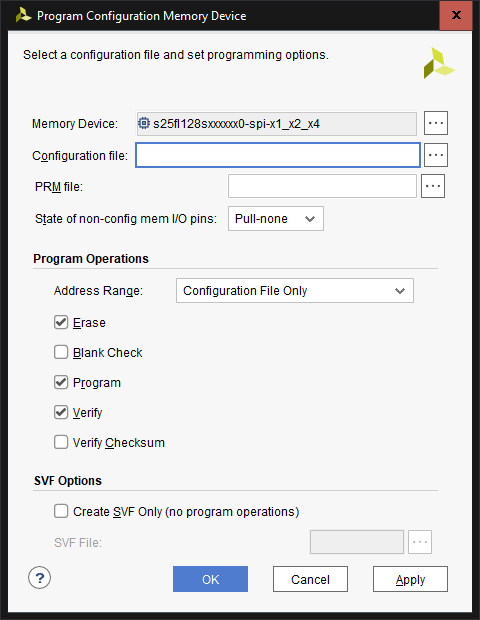
*Figura 4.6 – Seleccione el dispositivo de 3.3V de 128Mb de NOR Flash.*

4.9 – Nos preguntará luego si deseamos realizar la programación del dispositivo en la memoria y le damos aceptar.



*Figura 4.7 – Ventana de configuración a memoria.*

4.10 – Como no tenemos el archivo \*.mcs, si buscamos el archivo de configuración en la ventana emergente en “Configuration File” “…”, lo siguiente es generar el bitstream.

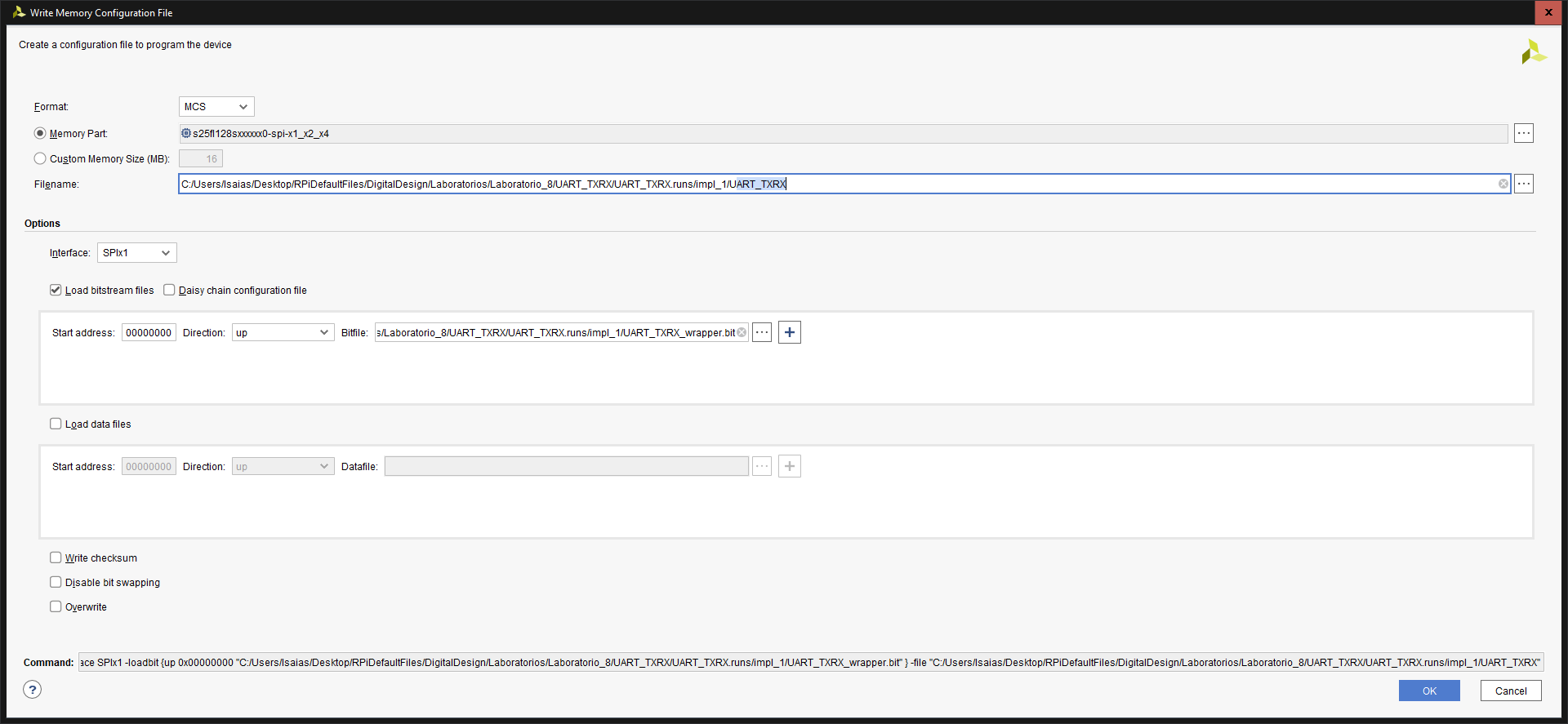


*Figura 4.8 – Ventana de selección del \*.mcs.*

4.11 – Genere nuevamente el bitstream nuevamente.

4.12 – Finalizado la generación del bitstream, generar el archivo de configuración \*.mcs nuevamente.

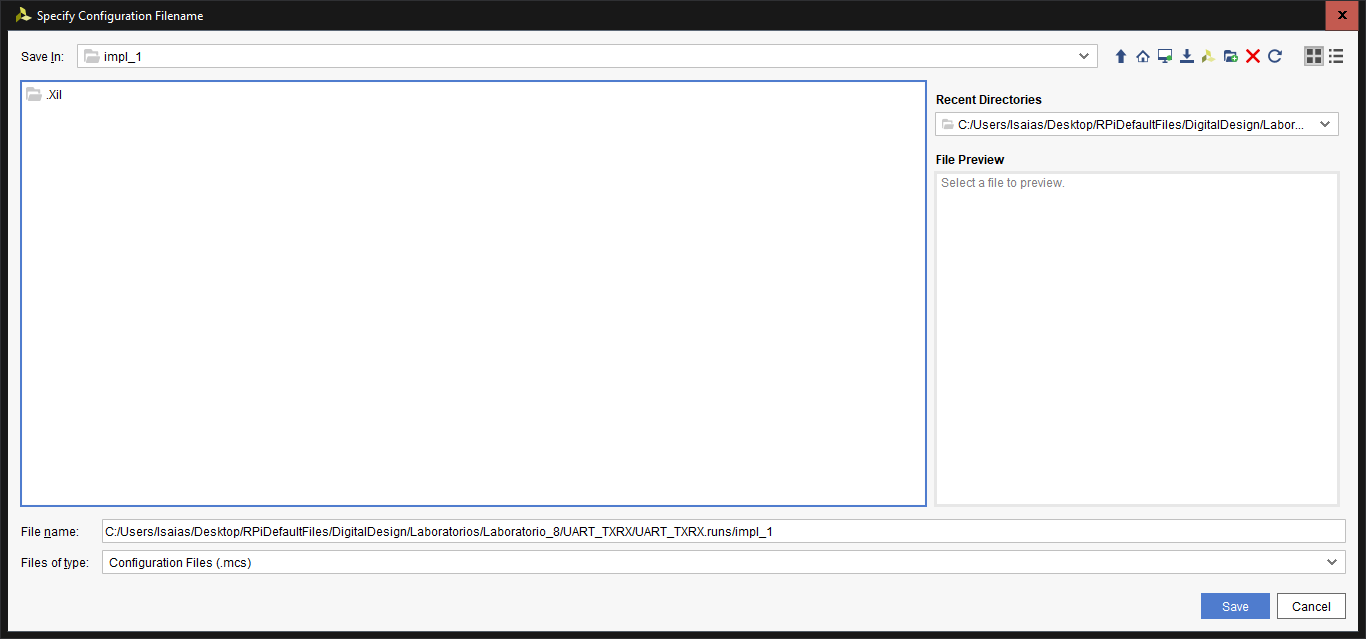
4.13 - Si observa la siguiente ventana notará que detecta la memoria deseada. Selecciónela.



*Figura 4.9 – Ventana de Búsqueda del archivo de configuración. Seleccione la memoria.*

4.14 – Selecciones “…” en Filename y guardar el archivo generado dentro de la carpeta de proyecto en:

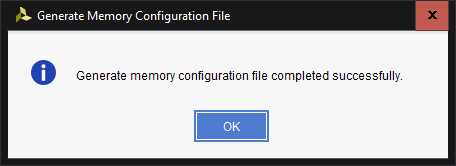
<Local Poject>/<Projec Name>.runs/impl\_1



*Figura 4.10 – Guardar el archivo en la carpeta del proyecto en impl\_1, por ejemplo.*

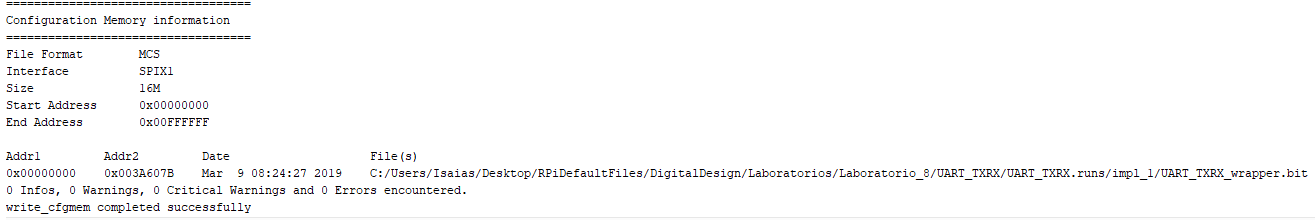
4.15 – Terminar en la pantalla como sigue, es decir, seleccionar le SPI, buscar el bit file en impl\_1 y especificar las rutas.

*Figura 4.11 – Salvar archivos. Salvados los archivos en impl\_1, también busque el bitfile y genere el mcs basado en el bitstream*

*Figura 4.12 – Ventana de resultado satisfactorio. Finalmente debe aparecer una ventana con los siguientes mensajes satisfactorios que denotan que se implemento el archivo de flash correctamente.*

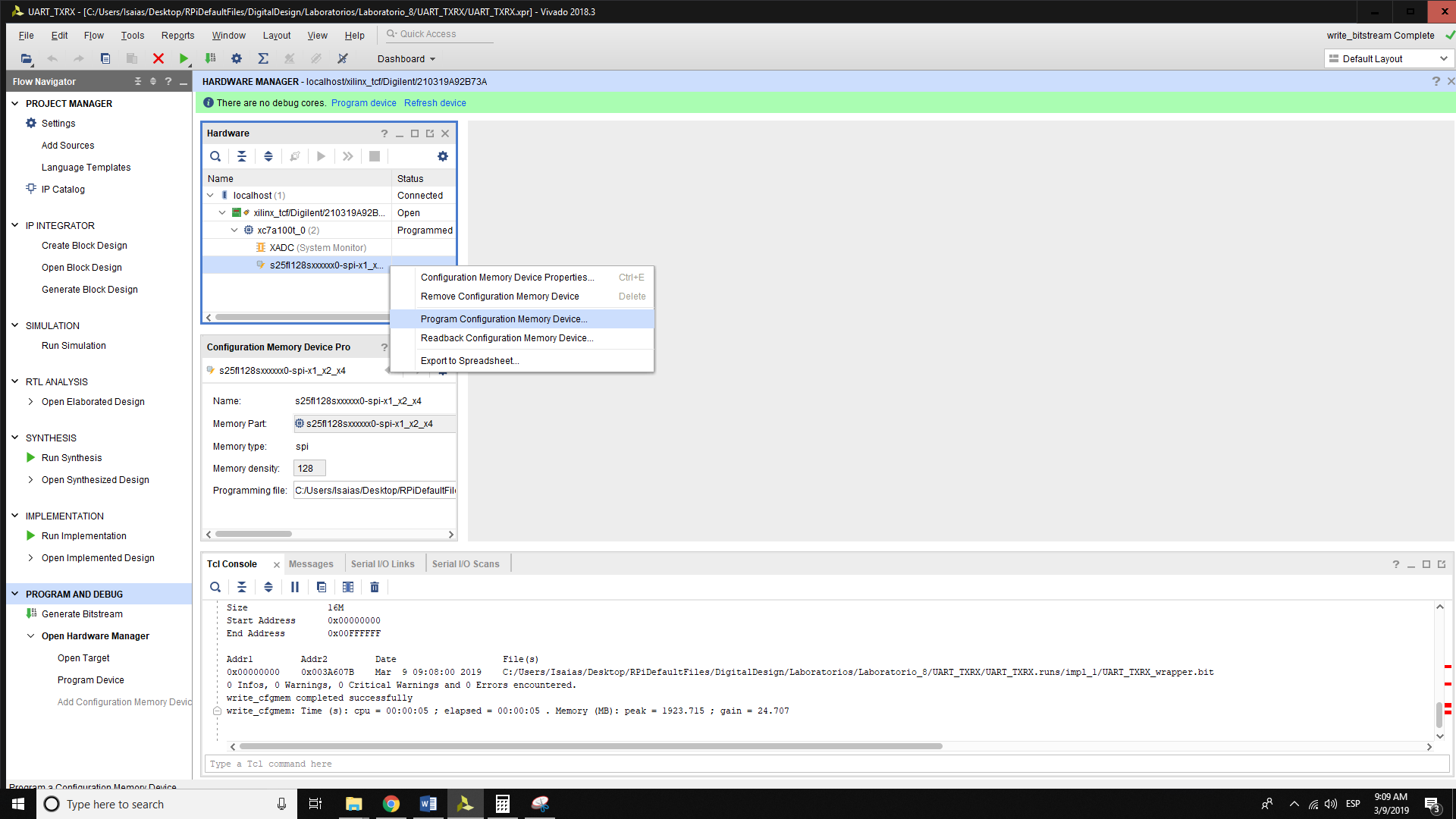
4.15 – Si observa la consola en la ventana inferior de vivado sale información general del archivo \*.mcs creado.



*Figura 4.13 – Información de la generación del archivo MCS.*

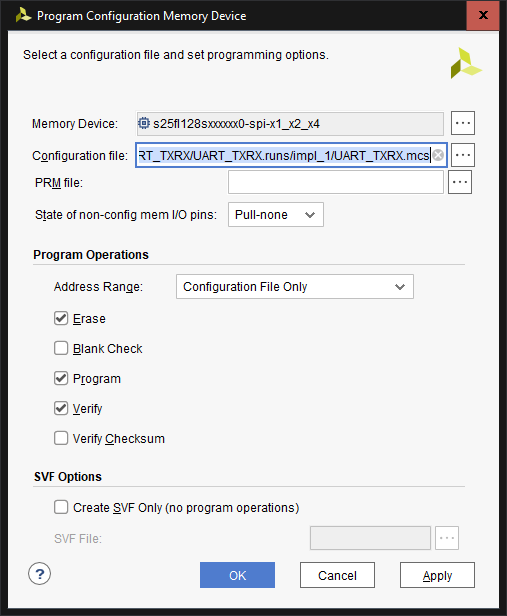
Si observamos un poco más a detalle la consola, en nuestro caso tenemos alrededor de 4MB de información, un poco más de lo detallado anteriormente de 2MB, es por esto que necesitamos dejar la mayor cantidad de espacio adicional para no traslapar datos.

4.16 – Regresar al hardware manager y realizar la programación de la memoria, hacer click derecho sobre la memoria y seleccionar la opción de programar el dispositivo de memoria.



*Figura 4.14 – Seleccionar programar el dispositivo de memoria.*

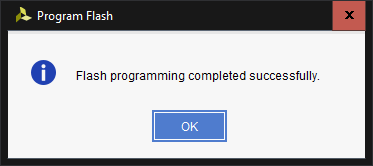
4.17 – Seleccionar el archivo \*.mcs creado para programar la memoria flash



*Figura 4.15 – Archivo \*.mcs y programación de memoria. Programar la memoria bajo el archivo creado.*

4.18 – Finalmente el archivo se descargará en la memoria flash como paso 1 y como paso 2 generará verificaciones para observar que no haya ningún problema al escribir.





*Figura 4.16 – Escritura del archivo MCS en la memoria. La finalización debe completar correctamente*

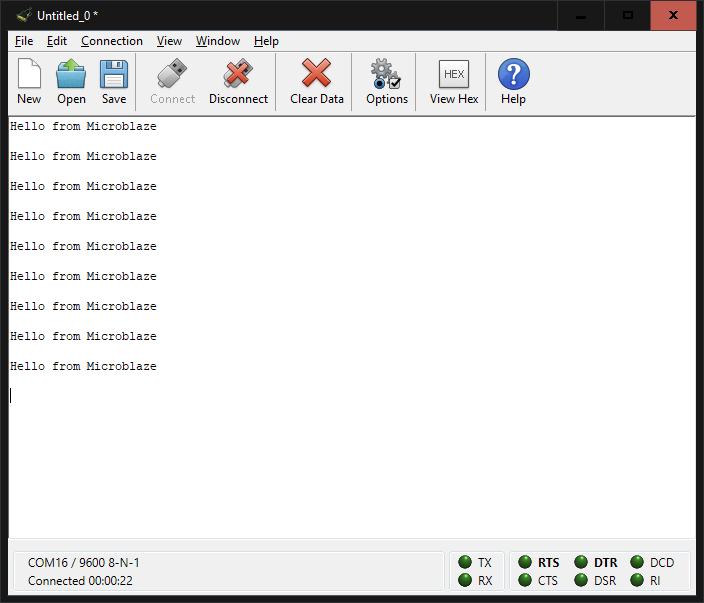
4.19 – Finalizada la generación del archivo apague la tarjeta.

4.20 – Inicie CoolTerm y conecte la tarjeta

4.21 – Abrir el puerto de comunicaciones respectivo y observar que se despliegan los mensajes sin necesidad de volver a programar.

4.21 – Presione el botón de Reset de la tarjeta (Se encuentra cerca del cable USB. Etiquetado como DONE O PROG).

4.22 – Observe que la programación no se pierde. NOTA: Sea paciente. La configuración toma aproximadamente 1 segundo en cargar de la memoria.



*Figura 4.17 – Comprobación del archivo de Microblaze retenido en la memoria Flash.*

5 - Evaluación

* 50% - Generación del archivo \*.mcs
* 50% - Demostrar que el FPGA no pierde la configuración