***Laboratorio 5 – UART***

Objetivos del laboratorio son conocer:

* Saber que es una UART
* Utilizar el “softcore processor” Microblaze

*Para el caso de la Arty utilizaremos una UART, pero utilizando el puerto USB, el chip que tiene por delante*

1. UART

1.1 ¿Qué es una UART?

Una UART es común en sistemas embebidos para comunicaciones seriales. Sirve para comunicar, por ejemplo, microcontroladores u otros dispositivos en sistemas embebidos.

UART viene de Universal Asynchronous Receiver Transmitter. Como es universal podemos utilizarlos con diferentes capas físicas como RS232, RS485, CAN, etc. Como el UART es asíncrono (a diferencia de SPI e I2C), solo necesita los pines de transmisión (TX) y recepción (RX) más una señal de referencia o GND.

Se puede utilizar por ejemplo para comunicar a nivel TTL dos dispositivos (microcontroladores, FPGAs, DSPs) o cualquier integrado que tenga una UART como interfaz.

Se puede implementar utilizando VHDL y usando un chip transceiver que convierta estas señales a USB. Para este laboratorio utilizaremos el Microblaze (Xilinx Soft Core Processor), C y VHDL para implementar una UART.

En UARTs se habla de full dúplex y half dúplex, lo normal es tener un half dúplex, esto quiere decir que solamente tendremos o una recepción o una transmisión a la vez, a diferencia de full dúplex que puede enviar y recibir al mismo tiempo.

1.2 Descripción de la UART

Observemos la figura inferior. Tenemos primero que detectar el flanco de bajada y esperar la mita de medio bit, para esto necesitamos conocer los baudios.

Los baudios son la cantidad de bits que podemos transmitir por segundos. Teniendo este número ahora luego de caída la señal de arranque esperamos la mitad del tiempo. Asumamos unos baudios de 9600 que por lo general todos los equipos vienen configurados (si no se dice la velocidad) a estos baudios, la transición de un bit a otro se realizará en 104.16us y medio baudio a 52.08us.

Esperado medio baudio realizamos la lectura para saber que estamos en el bit de arranque, si es un ‘1’ ya de salida repetimos nuevamente la recepción, si es un ‘0’ es lo esperado y ahora en vez de esperar medio baudio esperamos el tiempo de un bit que es 104.16 us.

Todos los 8 bits se transmiten en este caso en Little endian (bit menos significativo es transmitido primero). Al llegar al bit de paridad en este caso tenemos paridad impar. Paridad impar simboliza que el conjunto de bits de datos más el bit de paridad son de resultado impar.

Finalmente existe un bit de parada (que puede ser 1 o 2) para retornar la recepción o transmisión a estado ocioso.



*Figura 1.1 – Ejemplo de una Transmisión/Recepción de una UART.*

Cabe aclarar que para que una UART funcione o trasmita/reciba datos necesitamos realizar un cruce de señales. Esto es, debemos cruzar las líneas de transmisión con recepción. En la Arty no nos preocuparemos por eso pues ya están en hardware conectadas apropiadamente, sin embargo, si en un equipo tenemos este tipo de problemas lo primero que habría que realizar sería invertir los cables.



*Figura 1.2 – Ejemplo de conexión de señales de una UART.*

1.3 – Microblaze Softcore Processor

Cuando poseemos un microcontrolador como por ejemplo un ARM Cortex M4, el núcleo es el procesador basado en ARM, las empresas como Texas Instruments que fabrican un microcontrolador lo hacen rodeando este procesador con periféricos específicos.

Microblaze es similar, sin embargo, como Microblaze es un softcore processor podemos tener los temporizadores que queramos, programables o fijos además de toda la lógica de un FPGA a nuestro alcance. Así que en vez de tener un procesador separado de todo se tiene unido al sistema.

Microblaze tiene más versiones, aquí utilizaremos la versión más pequeña (por el momento) llamada Microblaze MCS el cual para esta aplicación es suficiente.

El softcore processor tiene una interfaz llamada AXI que si queremos añadir periféricos lo podemos realizar, por ejemplo, con GPIO unidos al puerto AXI.



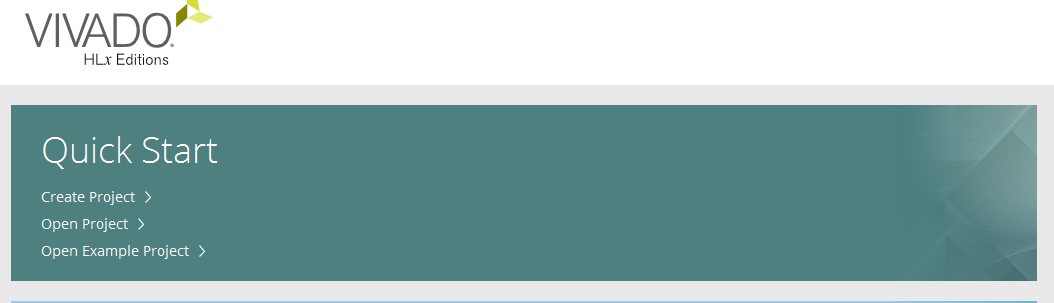
*Figura 1.3 – Microblaze SoftCore Processor.*

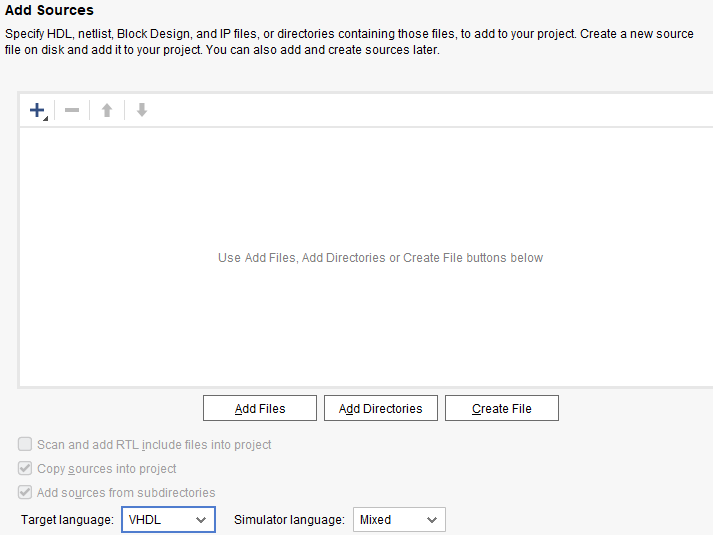
Una vez sintetizada, pasaremos al Xilinx SDK que nos dará la facilidad de programar en C la entidad que acabamos de describir en VHDL.

*2 - Creación del Proyecto de UART*

***NOTA: FAVOR CREAR EL PROYECTO EN UNA UBICACIÓN DONDE NO HAYA ESPACIOS, DE LO CONTRARIO EL XILINX SDK CREARÁ EL PROYECTO FUERA DE LA LOCALIDAD DEL PROYECTO.***

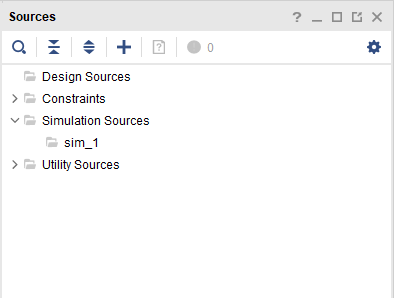
2.1 - Abrir Vivado en la pantalla principal y presionar en abrir proyecto





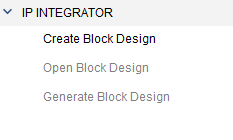
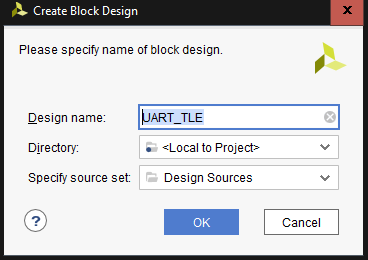
*Figura 2.1 – Creación del proyecto en Vivado. Asegurese de elegir VHDL!.*

2.2 – Crear del Project manager, un proyecto en blanco:



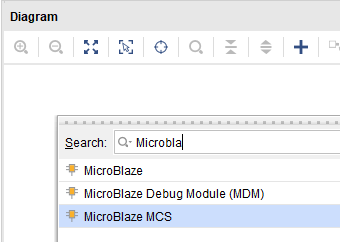
*Figura 2.2 – Archivos de trabajo de proyecto*

2.3 – De la ventana de Flow Navigator a su izquierda cree un diagrama de bloques.

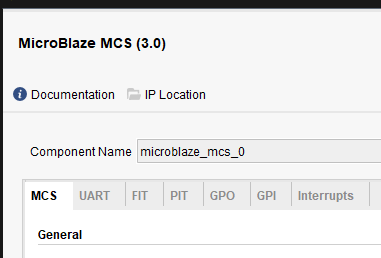
*Figura 2.3 – Creación de diagrama de bloques*

Del diagrama de bloques anterior creado, inserte el bloque de Microblaze MCS presionando ‘+’ y buscando el bloque Microblaze MCS que es el Microblaze de menor capacidad.



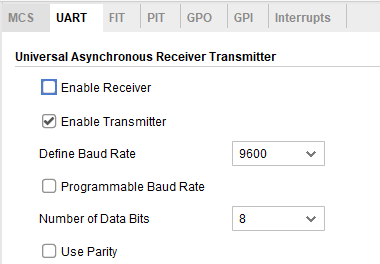
*Figura 2.4 – Inserción de Microblaze al proyecto*

Podemos utilizar el Microblaze normal, pero tomaría muchos componentes lógicos, si hace doble click verá la siguiente ventana.



*Figura 2.5 – Ventana general de Microblaze MCS.*

2.4 – Para la sección de UART tendremos entonces que habilitar solo el transmisor. Hacer click en OK luego de terminado de configurar Microblaze



*Figura 2.6 – Habilitando la UART.*



*Figura 2.7 – Microblaze MCS con su UART configurada. Si presiona el ‘+’ de lado donde dice ‘UART +’ en Microblaze podrá ver la descripción de pines de la UART.*

En este punto tenemos microblaze configurado para la UART, si queremos utilizar otro Microblaze podemos insertar uno más o los que queramos, pero recuerde que cada uno de estos procesadores consume memoria/bloques lógicos.

3.1 – Xilinx Development Kit (XSDK)

Xilinx Development Kit es una herramienta que tiene un IDE para poder utilizar solamente para un procesador de Xilinx, es decir, si queremos utilizar un microprocesador de Texas Instruments no se puede utilizar.

El XSDK es gratuito y ya instalado en el Vivado Webpack que ud. descargó e instaló. XSDK es basado en un proyecto open llamado Eclipse.

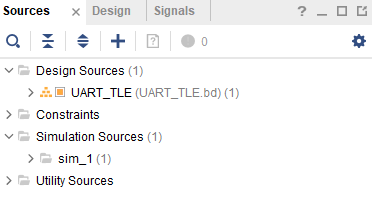


*Figura 3.1 – Xilinx SDK. Xilinx Software Development Kit*

Dato importante es saber que nosotros podríamos tener el softcore descargado en el FPGA, pero sin hacer lo que vamos a realizar en el XSDK simplemente el microprocesador estaría inactivo, solamente situado en memoria y consumiendo recursos. En orden para ejecutar su funcionamiento necesitamos utilizar el XSDK.

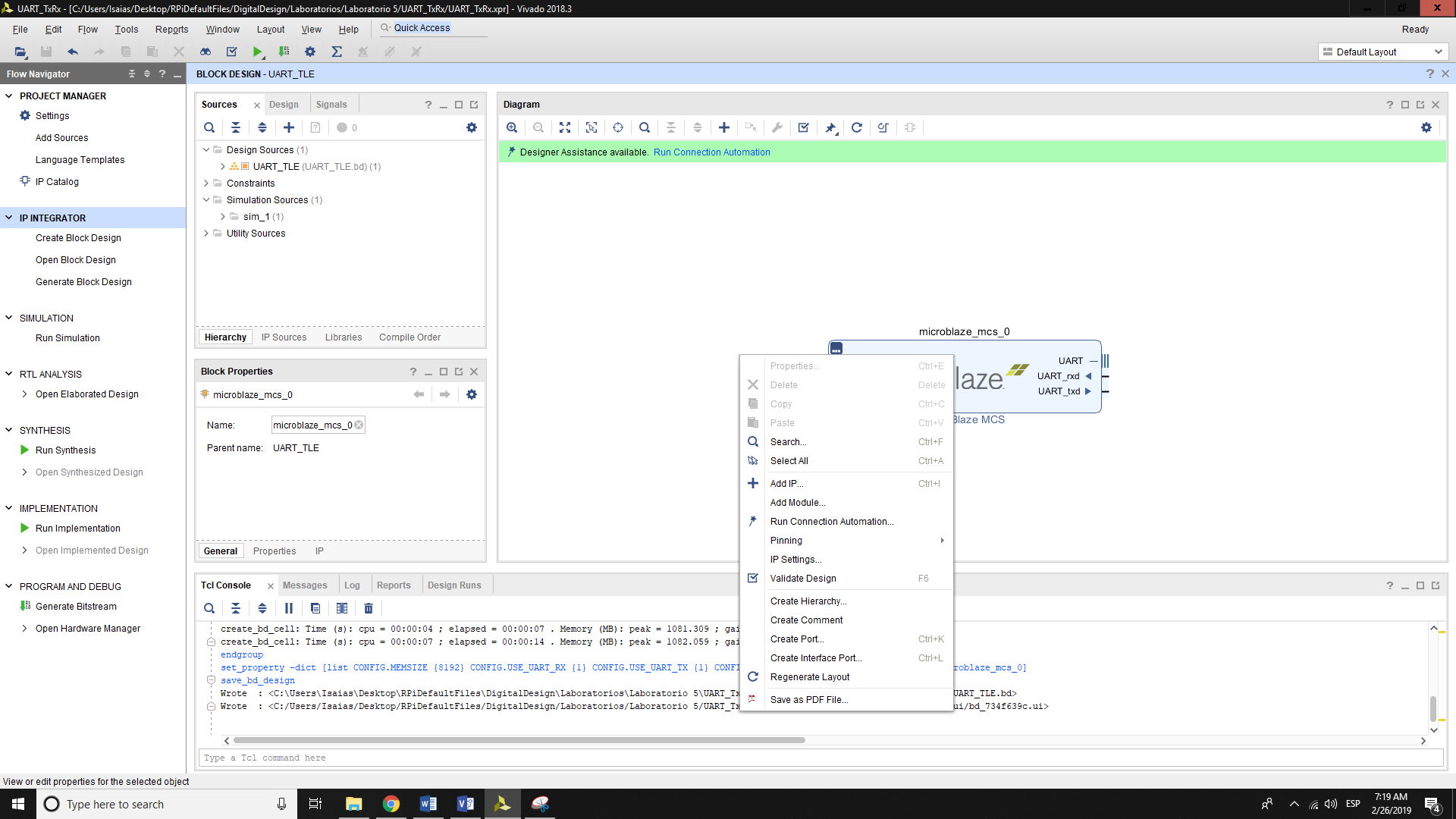
Con el XSDK también, pero no utilizaremos, podemos iniciar un proyecto con Microblaze y FreeRTOS que es un sistema operativo de tiempo real para sistemas embebidos.

3.2 – Debe haber terminado con una pantalla como sigue en el momento de haber creado el Microblaze MCS y configurado la UART antes de continuar

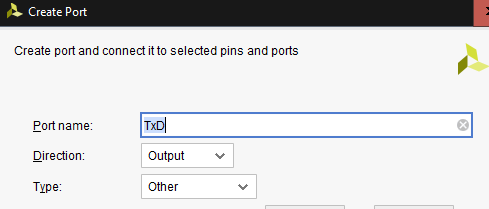
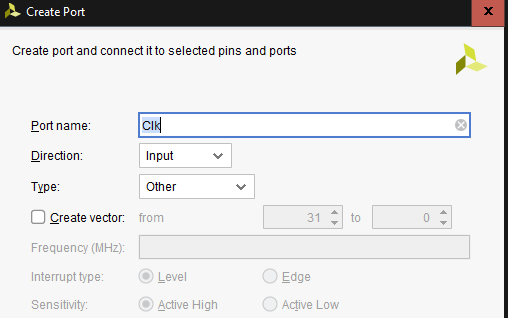


*Figura 3.2 – Sources de proyecto UART.*

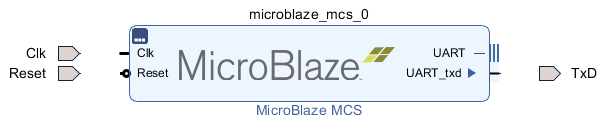
3.3 – Ahora añadiremos sobre el diagrama de bloques sus entradas y salidas. Para esto hacemos click con el botón derecho sobre el diagrama de bloques y creamos todas las señales siguientes con “Create Port…”



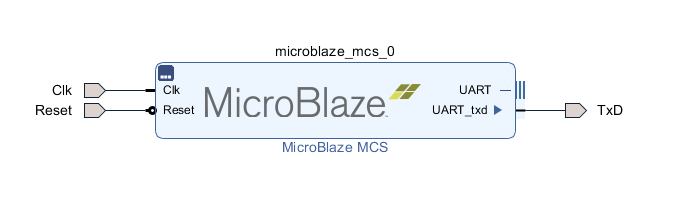
*Figura 3.3 – Empiece añadiendo los pines de entrada y salida y termine como se muestra en figuras siguientes.*



*Figura 3.4 – Tome en consideración que son 2 entradas y 1 salida*

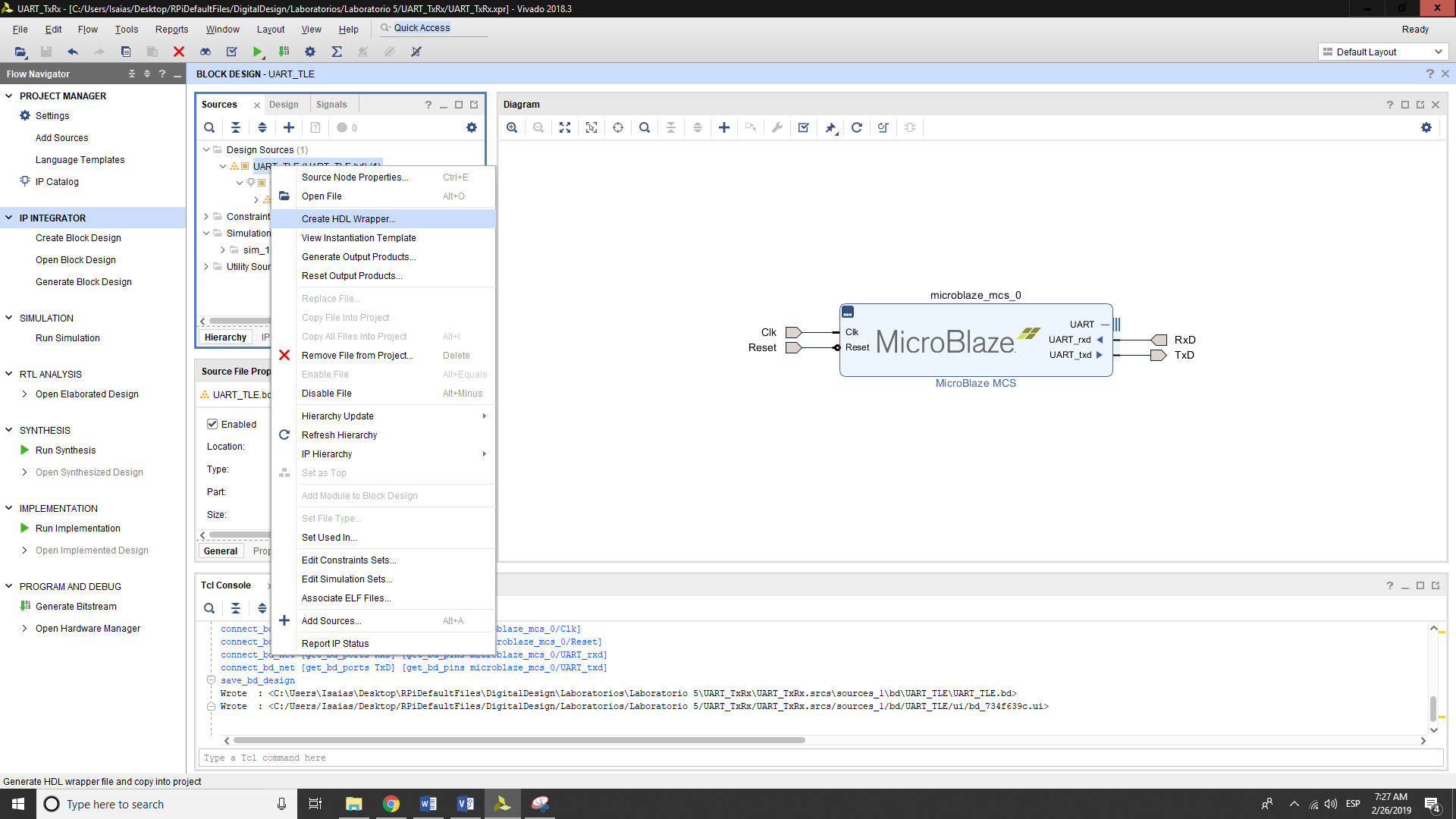


*Figura 3.5 – Diagrama de bloques de E/S en Microblaze. Termine el diagrama de bloques para tener una estrutura de entradas y salidas como se mue.*

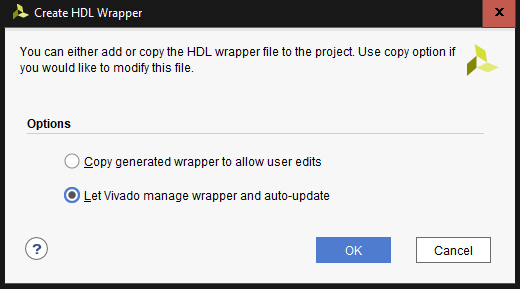
3.4 - Sitúese sobre los pines de E/S creados y haga click, manténgalo y cablee la entrada a su pin o la salida a su pin

*Figura 3.6 – Cableado de pines de Microblaze MCS.*

3.5 – Necesitamos crear una entidad superior de nuestro diagrama de bloques, por consiguiente, *primero asegúrese que tenemos en el proyecto la generación en VHDL* y luego hacemos click derecho en sources y nuevamente click en “Create HDL Wrapper …”.

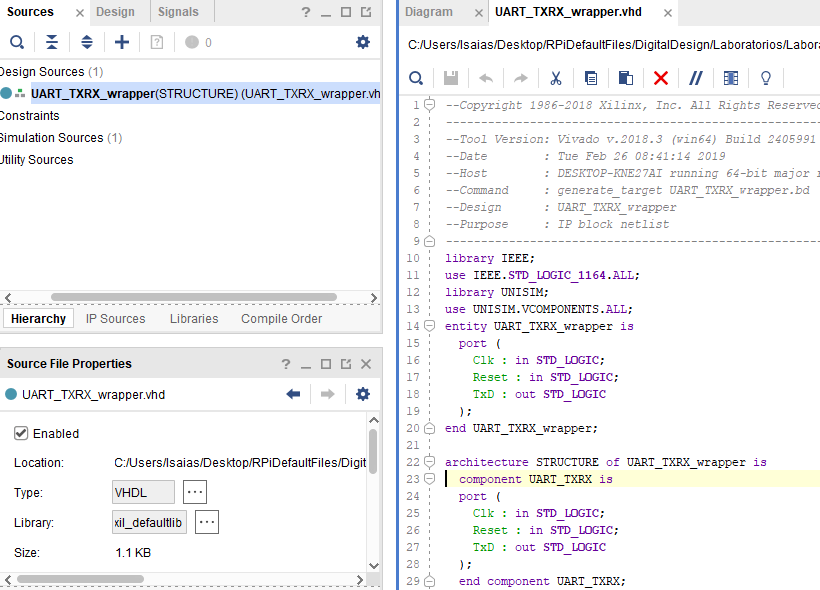


*Figura 3.7 – Creación de una entidad superior.*



*Figura 3.8 – Seleccionar para que Vivado administre los cambios.*

3.6 - En la siguiente figura, si se da doble click a UART\_TXRX\_wrapper.vhd, observamos que al añadir una nueva entidad superior el código no es tan diferente al estilo que hemos ido programando durante todos los laboratorios.



*Figura 3.9 – Entidad superior que envuelve a Microblaze en el proyecto.*

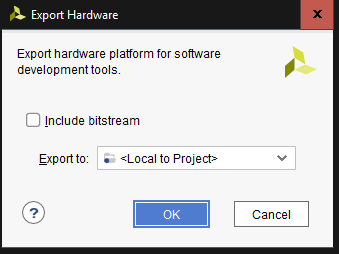
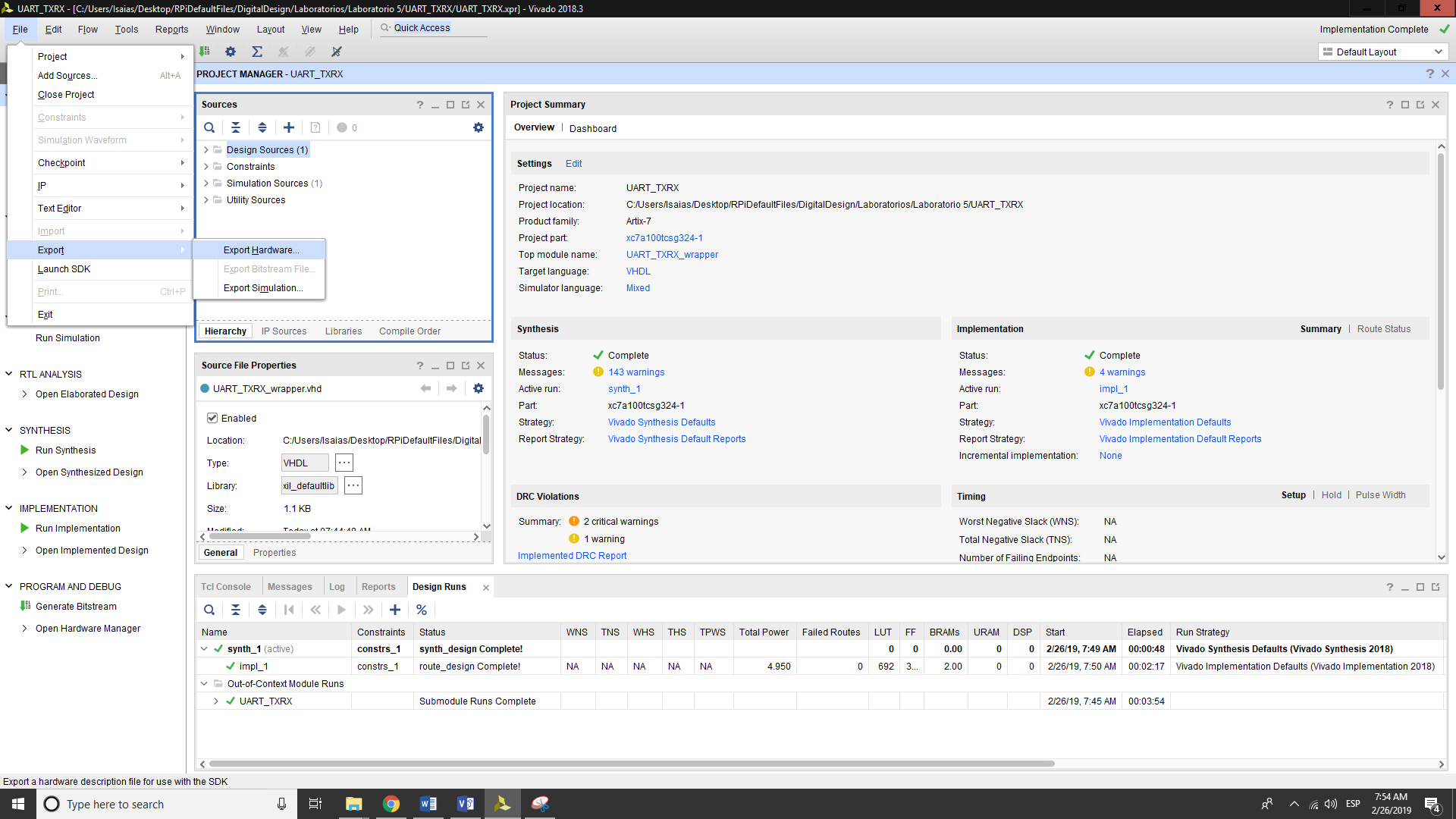
3.7 – Lo siguiente es generar la síntesis e implementación para corroborar el diseño, así que pasaremos por las diferentes partes de flujo realizada anteriormente. Primero realizaremos la corrida de síntesis, corrida de implementación y corrida de bitstream.



*Figura 3.10 – Los pasos de generación del bitstream.*

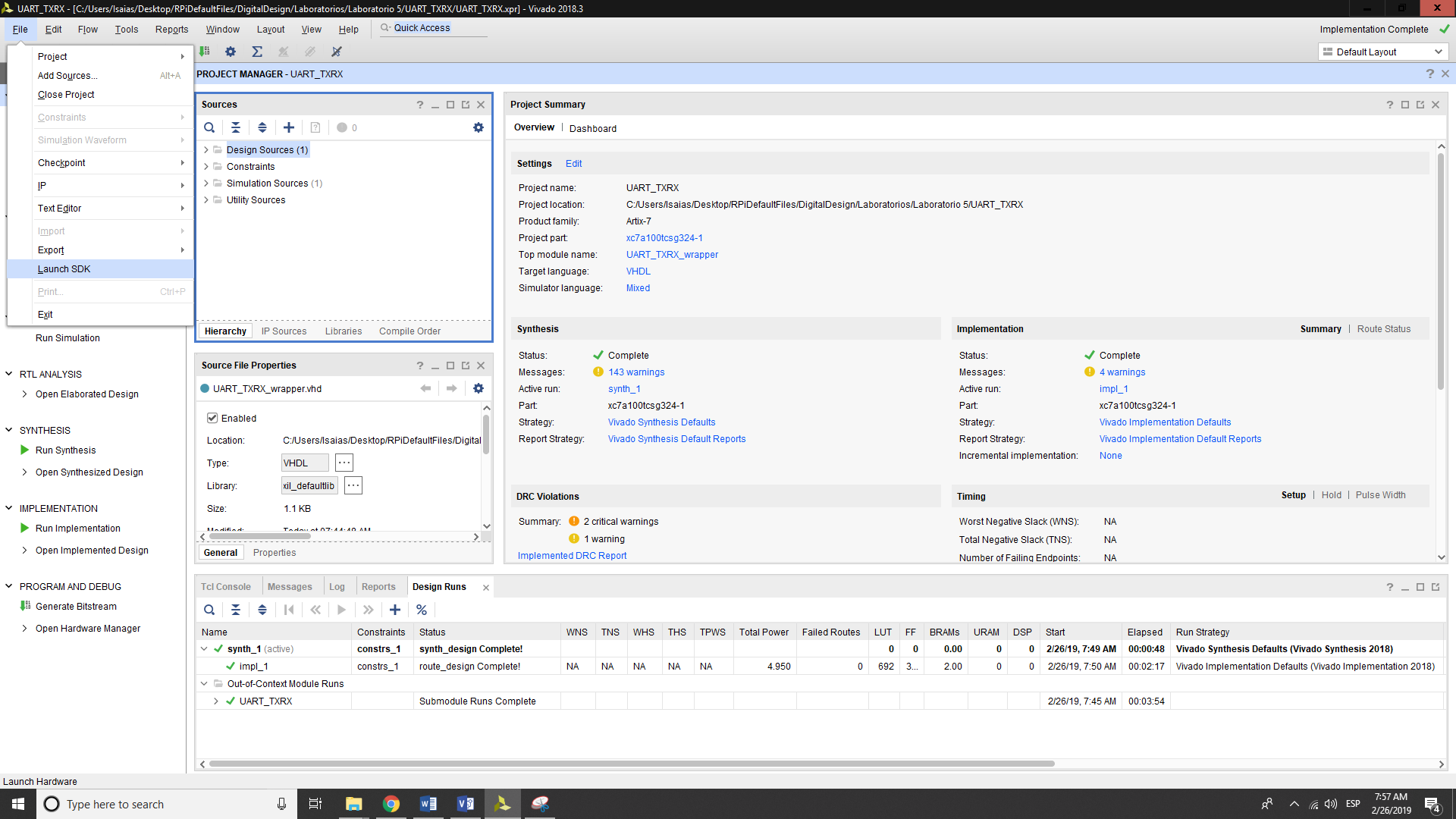
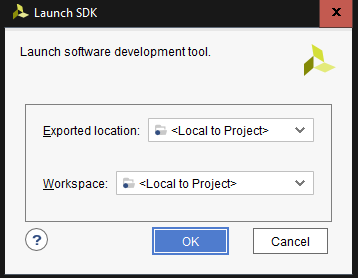
Una de las cosas que hace demorar el diseño de implementación es la memoria RAM utilizada en microblaze, así que esto demorará un poco. Podrá o no ver un error de generación de implementación debido a que no hemos definido las entradas y salidas (Vivado 2017 o menor), esto es normal y pasaremos al siguiente punto de nuestro proyecto por el momento. Lo siguiente es exportar el diseño al XSDK.

3.8 – Ahora exportaremos el diseño de hardware de alto nivel para que pueda utilizarse luego en el Xilinx Software Development Kit, así, valla a ‘File/Export/Export Hardware…’



*Figura 3.11 – Exportación del diseño de hardware.*

3.9 – Exportado el hardware lo siguiente es utilizar el Xilinx Software Development Kit.

*Figura 3.12 – Como lanzar Xilinx SDK.*

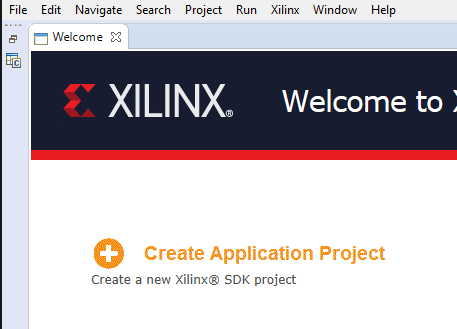
3.10 - El XSDK demorará un poco así que no desespere y tenga paciencia.



*Figura 3.13 – Ventana de XSDK.*

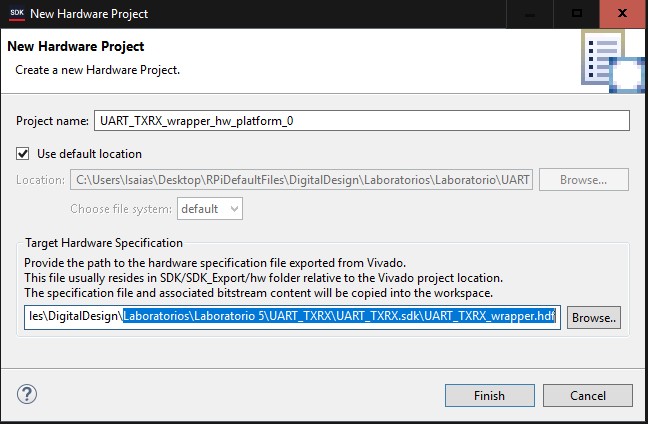
4 – Creación del proyecto en Xilinx SDK

4.1 – Seleccione luego de estar en la ventana inicial un nuevo proyecto de aplicación para iniciar.



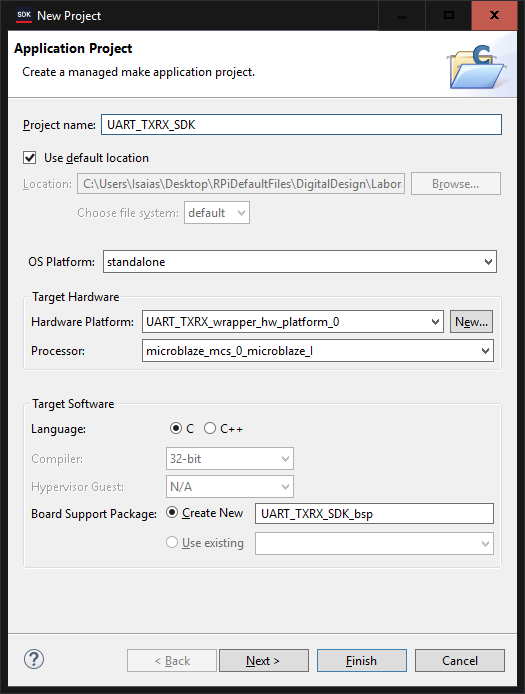
*Figura 4.1 – Ventana de XSDK.*

4.2 – Si todo está en orden no es necesaria esta ventana, de lo contrario, si su SDK ha generado en otra ubicación el proyecto, esta es la manera de buscarlo y que archivo debe encontrar.



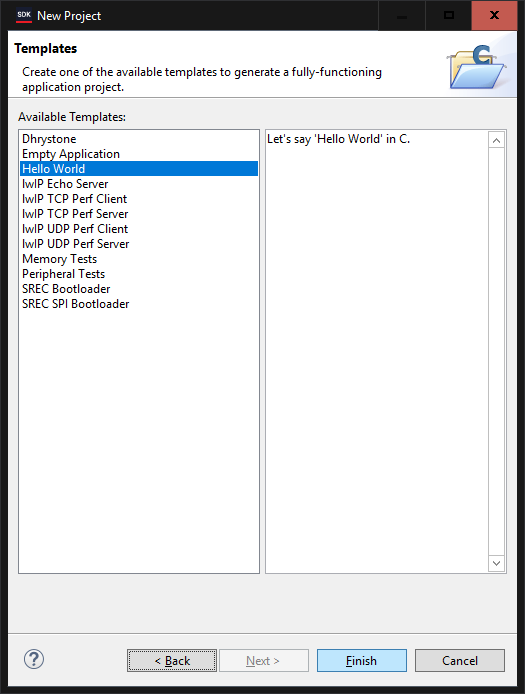
*Figura 4.2 – Ventana de Nuevo Proyecto de Hardware. Si no encuentra XSDK el paquete creado, esta será la manera de encontrarlo.*

4.3 – Utilice los siguientes parámetros como diseño del proyecto. Ud. Podría por ejemplo utilizar C++ pero recuerde que C tiene menor plantilla de espacio lo que consume menos memoria que es escasa en este sistema.



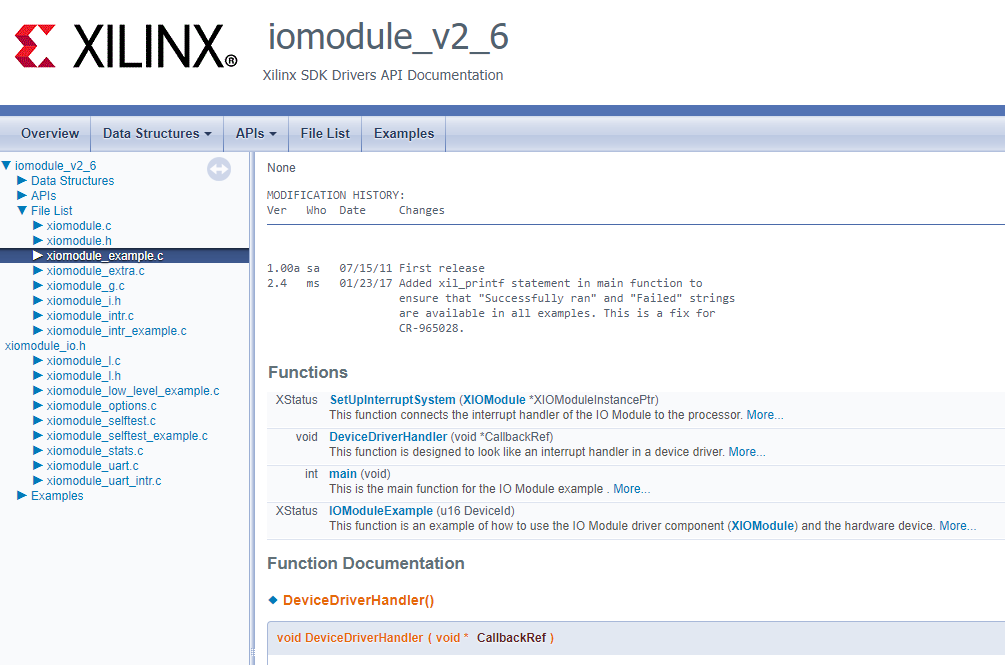
*Figura 4.3 – Ventana de nuevo proyecto de aplicación. Elegir C, asegurarse que el hardware es el que ud ha creado en Vivado y que es la instancia de microblaze correcta.*

4.4 – Haga click en siguiente y observe que se pueden crear varios proyectos, hasta un proyecto en blanco, nosotros empezaremos con un “esqueleto” llamado “Hello World” que es la estructura básica y de allí modificaremos para utilizar a nuestro objetivo.



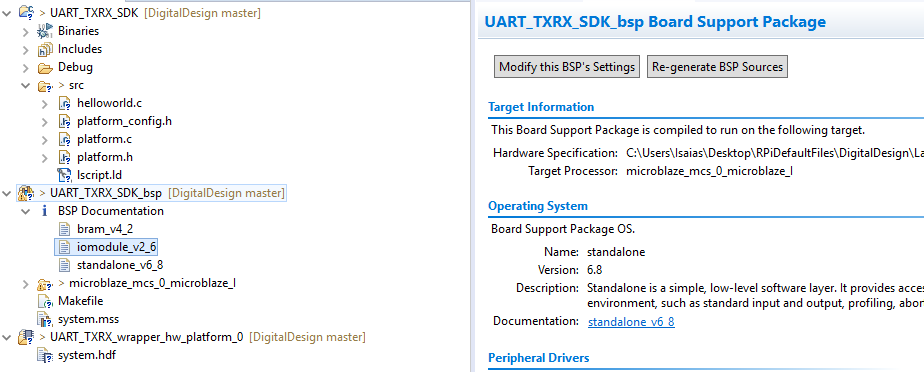
*Figura 4.4 – Ventana de nuevo proyecto, selección de aplicación. Elegir Hello World como base para la construcción de nuestra aplicación final.*

4.5 – Se nos deben de haber creado las siguientes carpetas que nos indican, por ejemplo, los BSP o “board support packages”, al abrir el vínculo (doble click al desplegar el BSP) que son librerías para utilizar periféricos



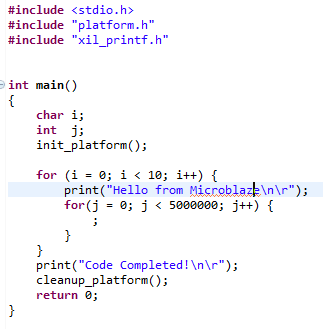
*Figura 4.5 – Documentación de un Board Support Package. Los BSP son entre algunas cosas, funciones, ejemplos, código que podemos reutilizar sin crear la rueda (sin empezar de cero). Nos sirven para empezar a crear aplicaciones.*

4.6 – Del archivo principal haga doble click en el archivo fuente “helloworld.c”. Es desde aquí que empezaremos a modificar nuestra aplicación final.



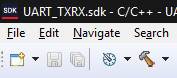
*Figura 4.6 – Carpetas creadas de proyecto. Nos basaremos en el archivo helloworld.c*

4.6 – Modifique el código para como sigue en la siguiente figura, imprimiremos 10 veces la palabra por el puerto serial.



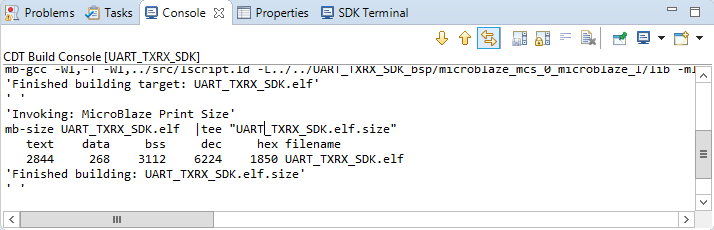
*Figura 4.7 – Archivo HelloWorld.c. Modificar el archivo para imprimir algunos caracteres por el puerto serial.*

4.7 – Guarde todos los cambios y presionar el martillo para que pueda genera el código 



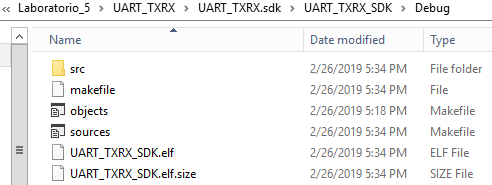
*Figura 4.8 – Ubicación de la herramienta build. La herramienta build sirve para compilar el proyecto.*

4.8 – En la parte inferior del proyecto, la pestaña de consola, debe ver algo parecido a la imagen inferior. Allí se muestra que la compilación ha terminado con éxito y que el archivo .elf, el cual utilizaremos más adelante ha de ocupar en la memoria alrededor de 6 Kb.



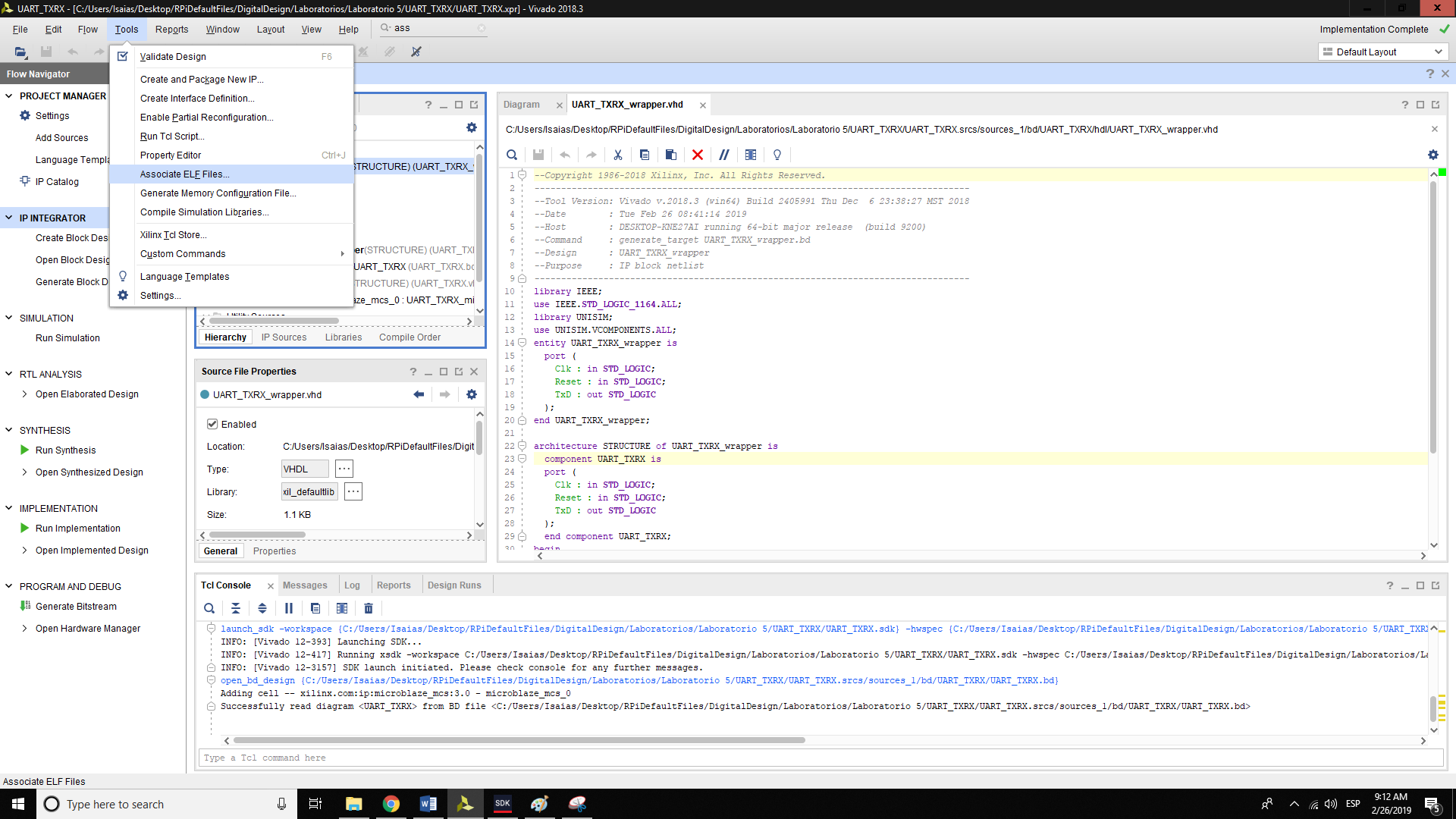
*Figura 4.9 – Finalización exitosa de proyecto. Note el tamaño de la aplicación que es menor al especificado en Microblaze MCS, de lo contrario nos dará error.*

4.9 - Como obserav, nos generó un archivo .elf de 6Kb, que será el que quemaremos en el FPGA en la siguiente dirección:



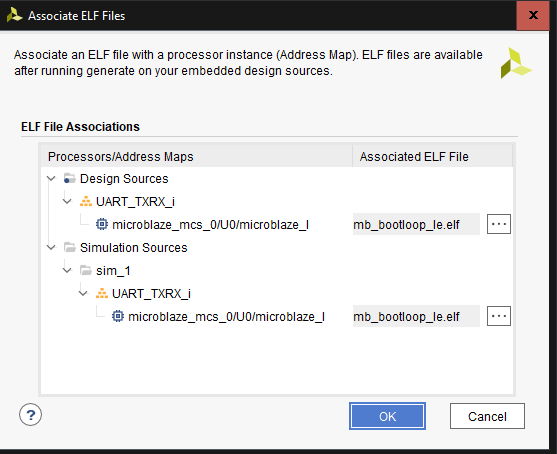
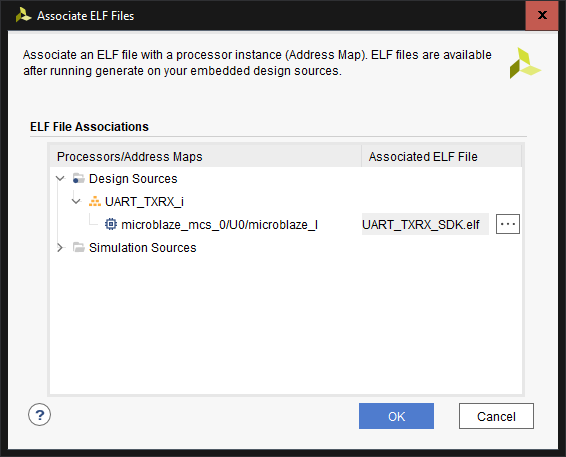
*Figura 4.10 – Ubicación del archivo .elf. El archivo .elf es necesario para construir la aplicación final del FPGA.*

4.10 – Lo seguido es asociar a nuestro diseño este archivo para generar el bitstream luego. Primeramente nos vamos a Vivado y en “Tools/Associate ELF Files…” lo asociamos de la carpeta de la figura 4.10.



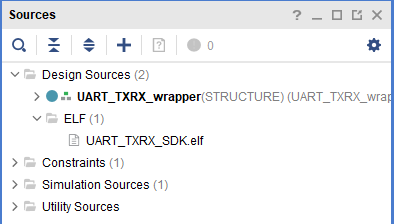
*Figura 4.11 – Asociación del archivo ELF. El archivo ELF generado en XSDK nos servirá para generar el bitstream final del proyecto.*

4.10 – Se abrirá una ventana que nos dará dos listas de .elf asociados. No nos interesa las fuentes de simulación sino las fuentes de diseño, por consiguiente, asocie el .elf de la figura 4.10 en Design Sources

*Figura 4.12 – Asociación del archivo ELF. El archivo ELF generado en XSDK nos servirá para generar el bitstream final que tiene la aplicación a ejecutar.*

4.11 – La ventana final debe estar como sigue:

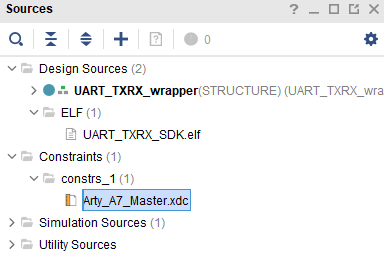


*Figura 4.13 – Adición del archivo .elf al proyecto. Vista final del archivo a ejecutar.*

5. Posicionamiento de Pines

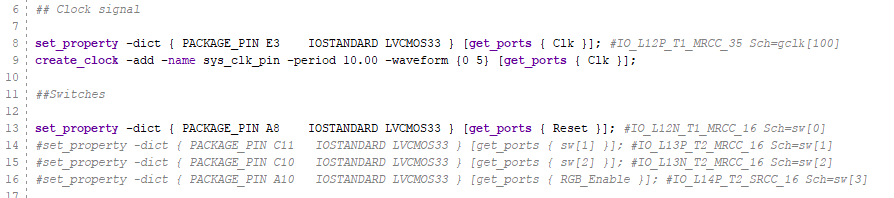
5.1 - Lo seguido es realizar el posicionamiento de pines como lo hemos hecho anteriormente.

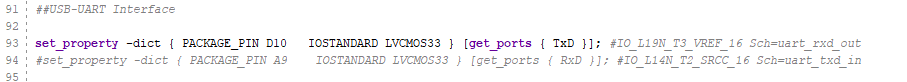
Para esto agregue el archivo Arty\_A7\_Master.xdc de los grupos de laboratorios anteriores.



*Figura 5.1 – Adición del archivo maestro Arty\_A7\_Master.xdc.*

5.2 – Modifique las líneas especificas listadas en la figura inferior que nos ayudarán a generar la salida específica necesaria por el puerto serial.

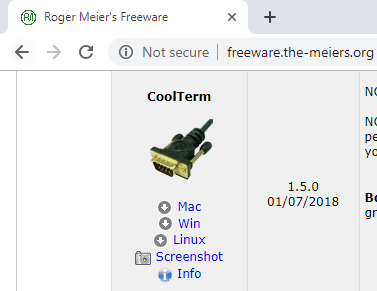




*Figura 5.2 – Modificación del archivo Arty\_A7\_Master.xdc.*

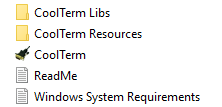
5.3 – Salve los cambios de la ventana anterior y como siguiente paso genere el bitstream de la manera común de otros laboratorios. Finalizada la generación del bitstream cancele la ventana que se despliega.

5.4 – Descargue CoolTerm de la página que muestra la figura siguiente.



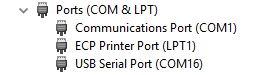
*Figura 5.3 – CoolTerm. Emulador de terminal serial multiplataforma.*

5.5 – Descomprima CoolTerm en una carpeta que pueda ubicar rápidamente e inicie el ejecutable CoolTerm.



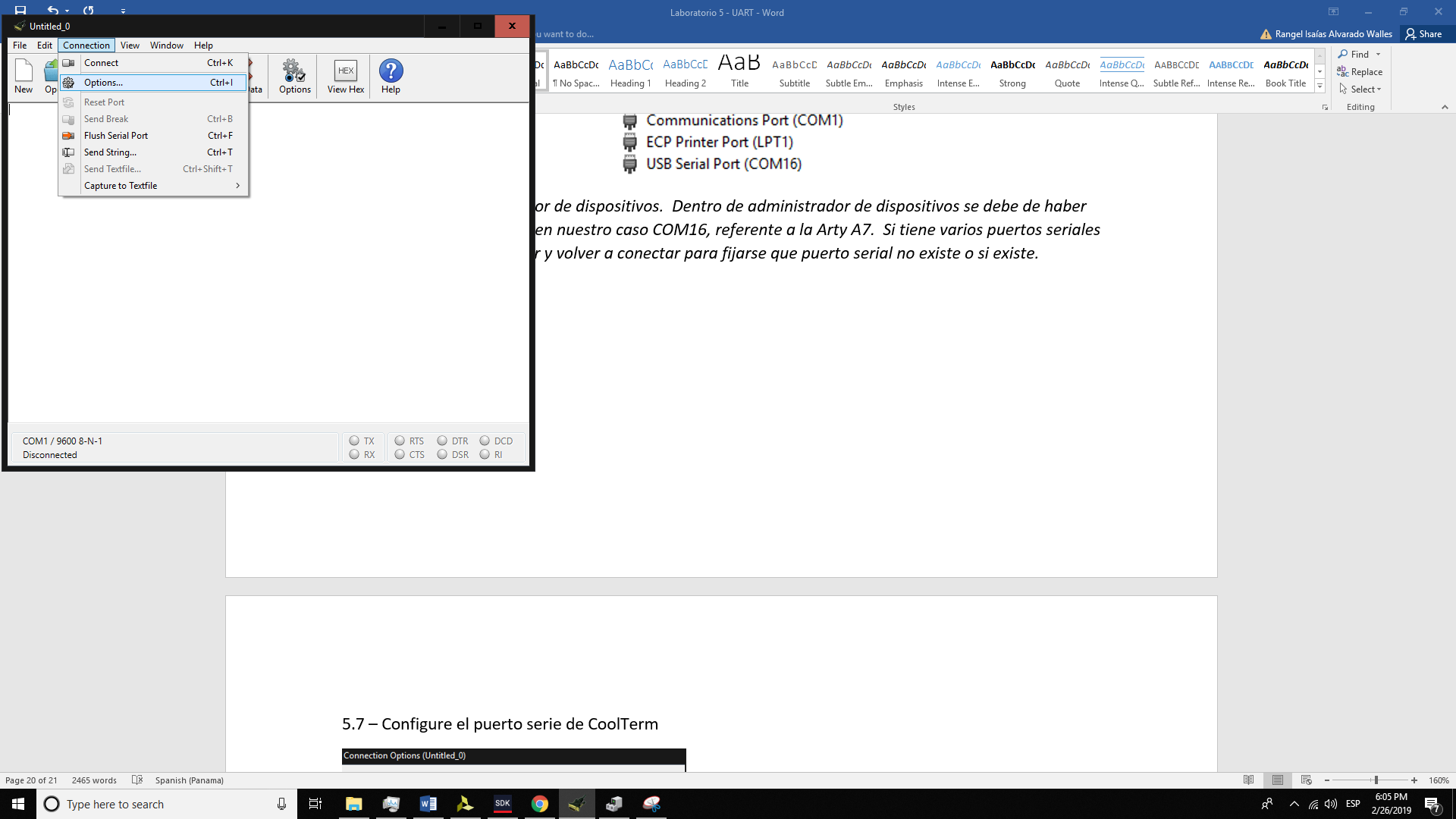
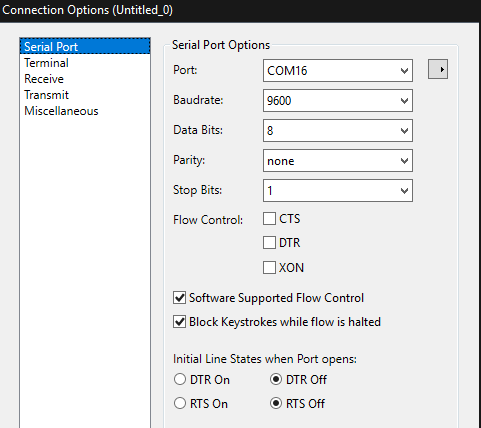
*Figura 5.4 – Ejecutable de CoolTerm. Inicie la aplicación. La ventaja de CoolTerm es que no es un instalable y puede borrarse cuando lo desee.*

5.6 – Conecte la tarjeta Arty A7 a su computador para que pueda crearse el puerto serial virtual. Encuentrelo en administrador de dispositivos.



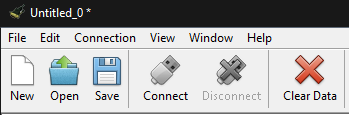
*Figura 5.5 – Administrador de dispositivos. Dentro de administrador de dispositivos se debe de haber creado un USB Serial Port, en nuestro caso COM16, referente a la Arty A7. Si tiene varios puertos seriales puede desconectar y volver a conectar para fijarse que puerto serial no existe o si existe.*

5.7 – Configure el puerto serie de CoolTerm. Si no se encuentra, en la ventana Options, en la parte inferior hay un botón que dice “Re-Scan Serial Ports”.

*Figura 5.6 – Configuración de CoolTerm. Configurar el puerto serial a los mismos baudios que la UART del Microblaze, de lo contrario no tendrá comunicación.*

COM16 es el puerto serial creado por la tarjeta Arty A7, en su caso debe revisar que puerto serial ha creado para proceder. Presione OK y luego proceder al siguiente paso.

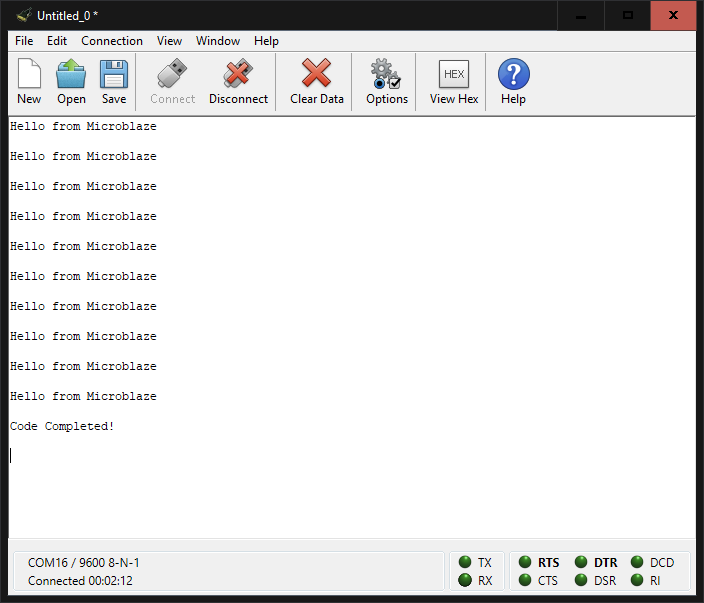


*Figura 5.7 – Presione el botón Connect para abrir el puerto serial (en la PC) de la Arty A7.*

**5.8 – El siguiente paso es verificar que el switch de Reset en la Arty A7 (configurado en hardware como el SW0) debe estar en dirección ON (cercano al header), de lo contrario Microblaze no actuará.**

5.9 – Descargue el bitstream como lo ha realizado en laboratorios anteriores.

5.10 – Observe nuevamente CoolTerm, debe ver una salida como sigue dependiendo del mensaje ingresado:



*Figura 5.8 – Vistazo final a la aplicación de comunicación del puerto serial.*

6 - Evaluación

* 50% - Terminar el proyecto
* 50% - Modificar los baudios según la tabla dada a continuación y demostrar en el FPGA