***Laboratorio 6 – UART y GPIOs***

Objetivos del laboratorio son conocer:

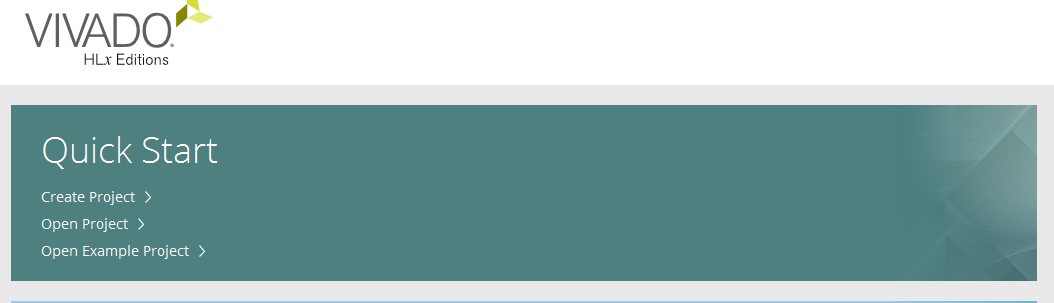
* Utilizar la UART para enviar y recibir mensajes
* Activar los LEDs dependiendo del mensaje enviado
* Saber el estado de los GPIOs dependiendo del mensaje a transmitir

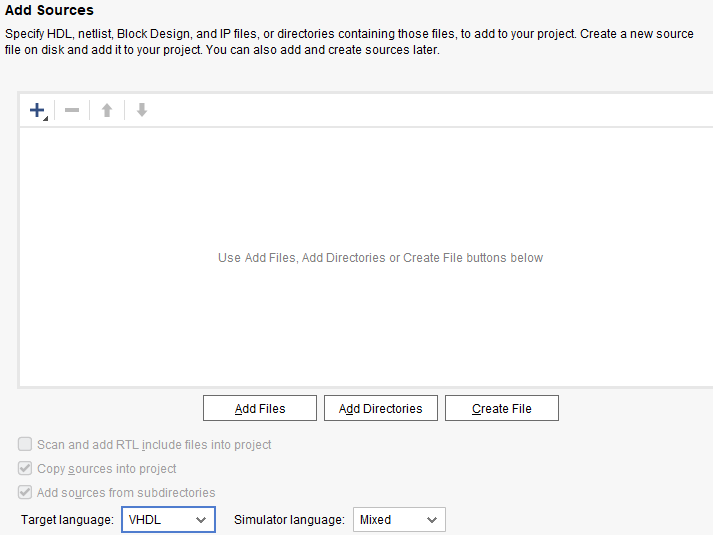
1. UART y GPIOs

*1.1 - Creación del Proyecto de UART y GPIOs y Configuración de Microblaze MCS*

***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.***

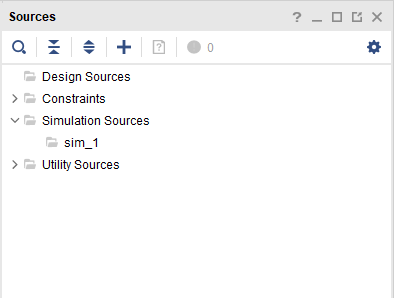
1.2 - Abrir Vivado en la pantalla principal y presionar en abrir proyecto





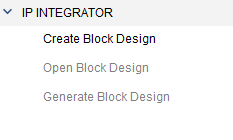
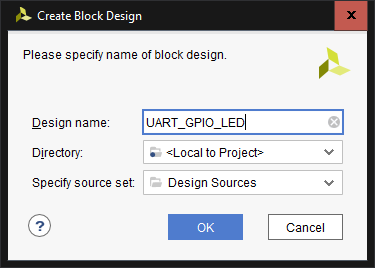
*Figura 1.1 – Creación del proyecto en Vivado. Asegurese de elegir VHDL!.*

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



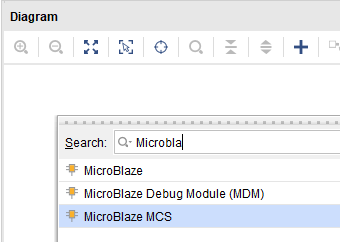
*Figura 1.2 – Archivos de trabajo de proyecto*

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

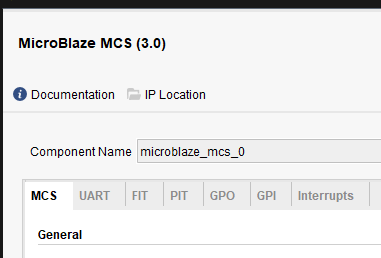
*Figura 1.3 – Creación de diagrama de bloques*

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.

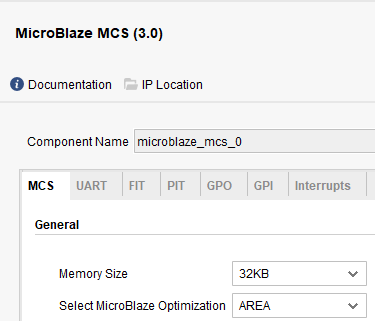


*Figura 1.4 – Inserción de Microblaze al proyecto*

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

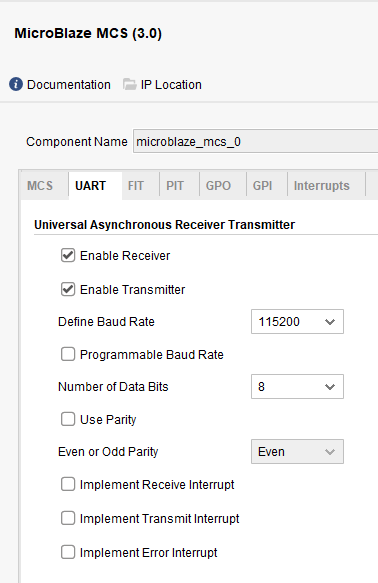


*Figura 1.5 – Ventana general de Microblaze MCS.*



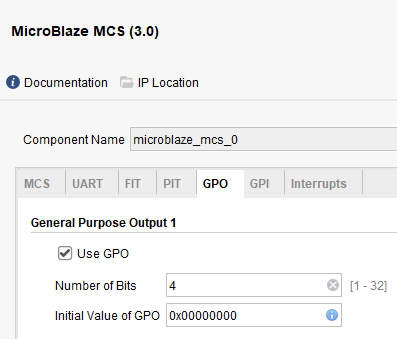
*Figura 1.6 – Ventana de Microblaze MCS, habilitar 32Kb de memoria de bloque.*

1.5 – Para la sección de UART tendremos entonces que habilitar tanto el transmisor como el receptor.



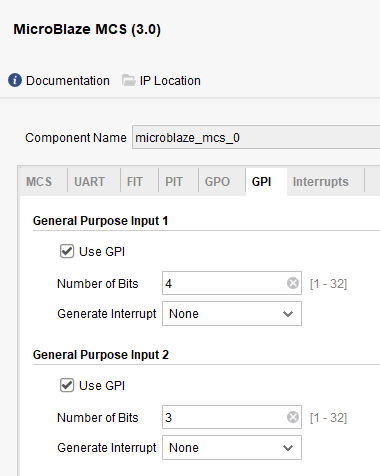
*Figura 1.7 – Habilitando la UART.*

1.6 – Para la sección de GPO habilitar 4 salidas, que corresponden a los LEDs.

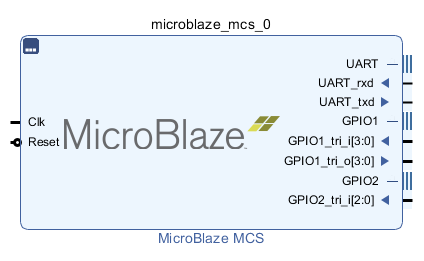


*Figura 1.8 – Habilitar las 4 salidas que corresponden a los LEDs.*

1.7 – Habilitar 4 entradas para la GPIO1 y 3 entradas para la GPIO2. Usaremos 4 switches y 3 botones debido a que uno corresponde al reset del microcontrolador.



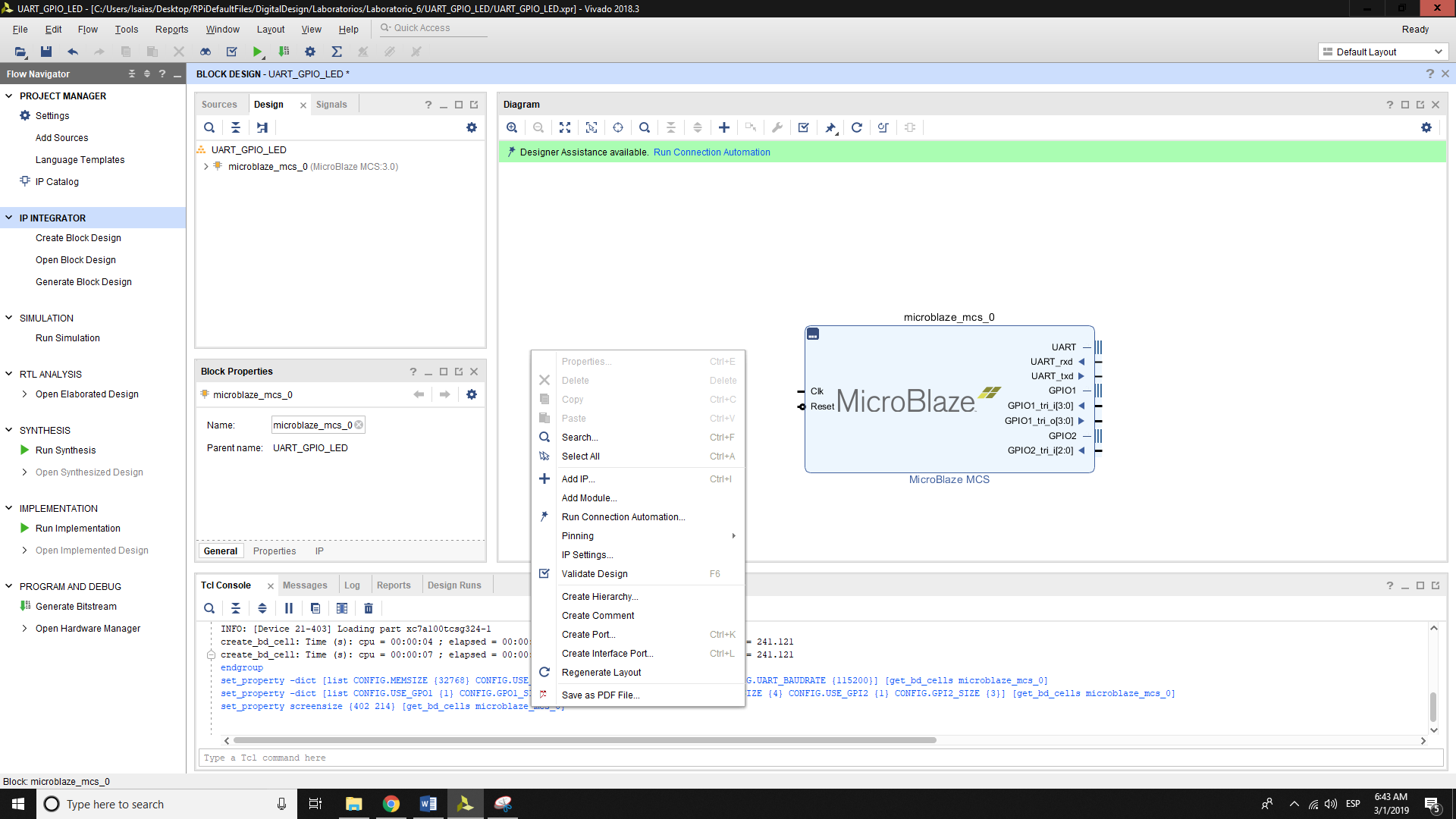
*Figura 1.9 – Habilitar las 4 entradas y 3 salidas.*



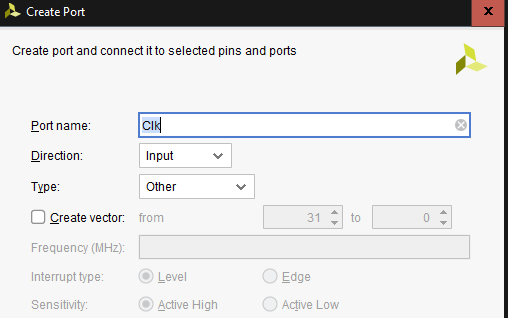
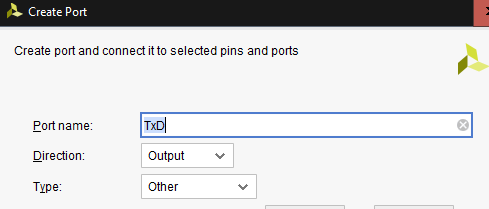
*Figura 1.9 – Microblaze MCS con su UART y GPIOs configuradas. Si presiona el ‘+’ de lado donde dice ‘UART +’ , ‘GPIO1 + ‘ y ‘GPIO2 +’ observará el bloque de Microblaze como se muestra en esta figura.*

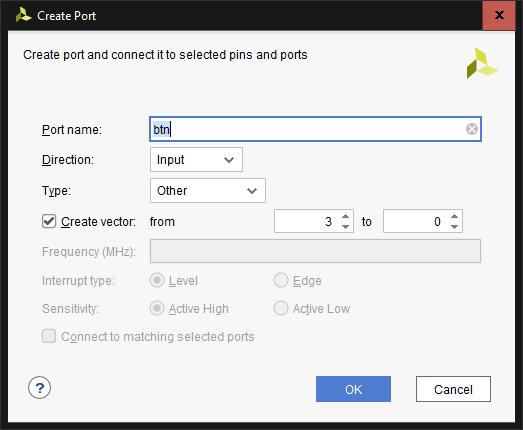
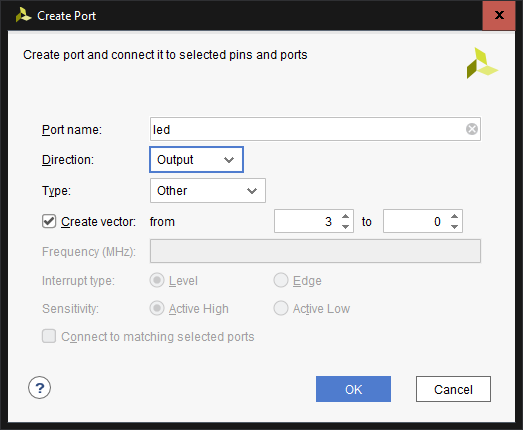
En este punto tenemos microblaze configurado para la UART y GPIOs, si queremos utilizar otro Microblaze podemos insertar uno más o los que queramos, pero recuerde que cada uno de estos procesadores consume memoria/bloques lógicos.

1.8 – 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…”



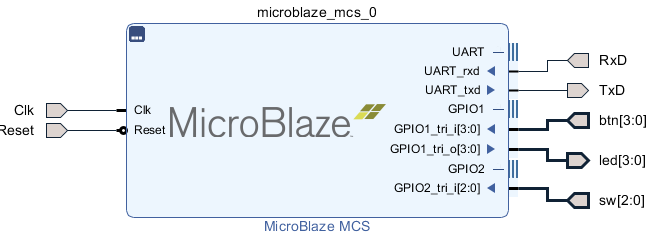
*Figura 1.10 – Empiece añadiendo los pines de entrada y salida con “Create Port…” y termine como se muestra en figuras siguientes.*

*Figura 1.11 – Tome en consideración cuales son entradas, cuales son salidas y cuales vectores como se muestra en estas figuras. Ejemplos de algunas señales y vectores.*

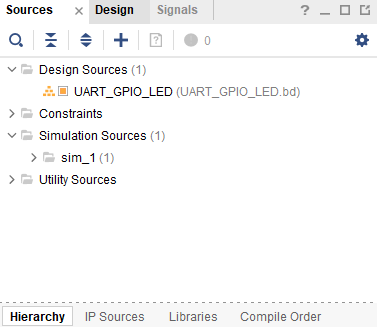
1.9 - Sitúese sobre los pines de E/S creados y haga click, manténgalo y cablee la entrada a su pin o la salida a su pin, debe terminar como sigue:



*Figura 1.12 – Cableado de pines de Microblaze MCS.*

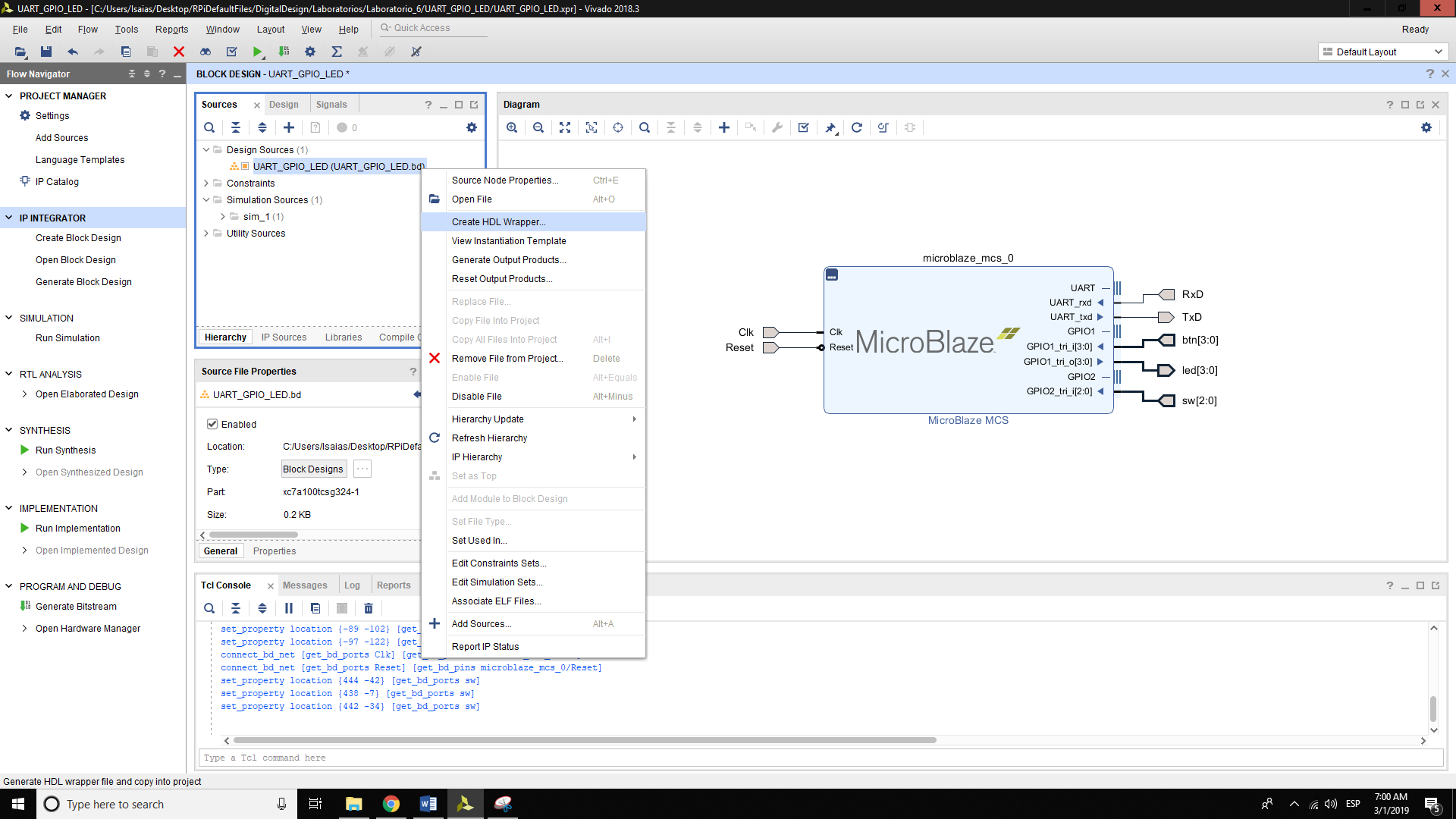
*2. – Diseño completo de UART y GPIOs, guía en pasos*

2.1 – Debe haber terminado con una pantalla como sigue en el momento de haber creado el Microblaze MCS y configurado la UART y GPIOs como en la figura 1.12 antes de continuar además de una memoria de 32Kb, en la ventana Sources debe mostrarse como sigue el diagrama de bloques.

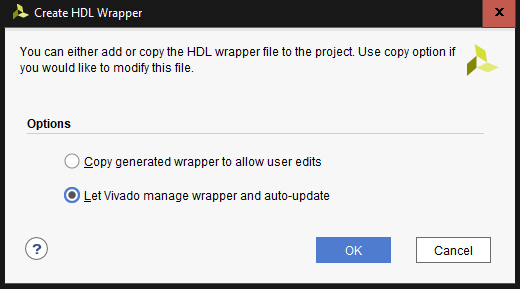


*Figura 2.1 – Sources de proyecto UART y GPIO.*

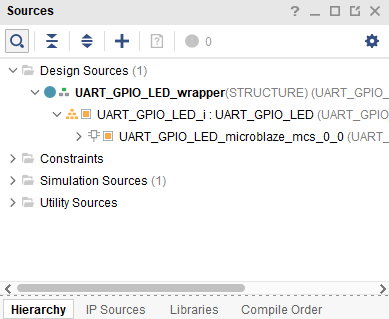
2.2 – Necesitamos crear una entidad superior de nuestro diagrama de bloques, por consiguiente, *primero asegúrese que tenemos en el proyecto la generación en VHDL* y luego hacemos click derecho en sources y nuevamente click en “Create HDL Wrapper …”.



*Figura 2.2 – Creación de una entidad superior.*

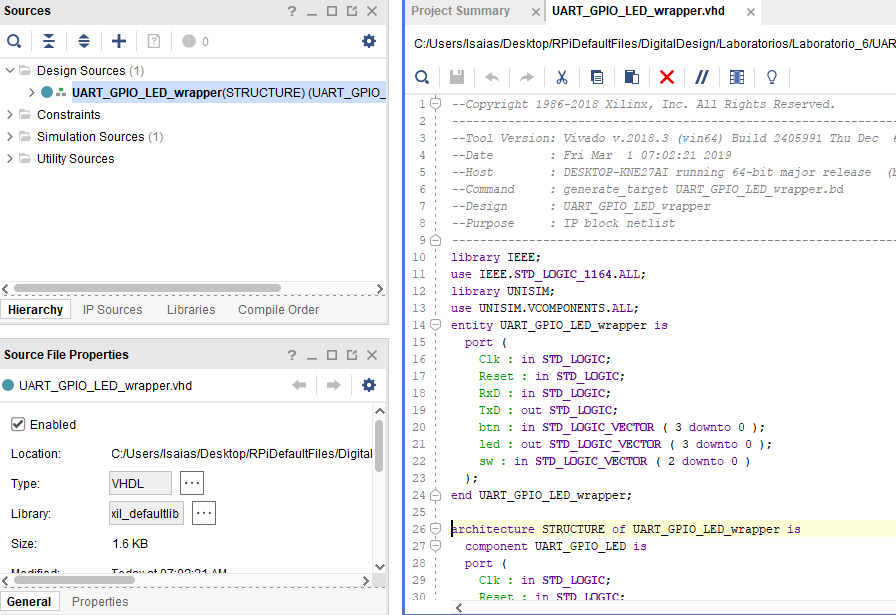


*Figura 2.3 – Seleccionar para que Vivado administre los cambios.*



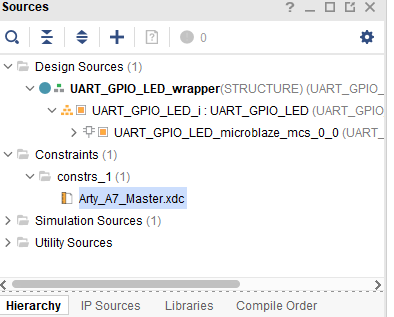
*Figura 2.4 – Envolvente (wrapper) de VHDL para Microblaze MCS.*

2.3 - En la siguiente figura, si se da doble click a UART\_GPIO\_LED\_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.



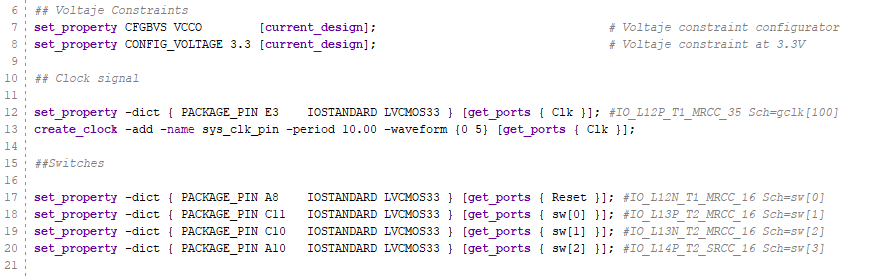
*Figura 2.5 – Entidad superior que envuelve a Microblaze en el proyecto.*

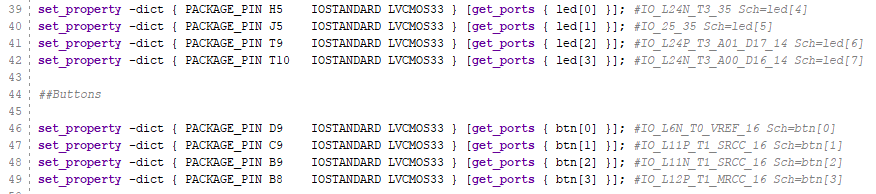
2.4 - Lo seguido es realizar el posicionamiento de pines como lo hemos hecho anteriormente. Para esto agregue el archivo Arty\_A7\_Master.xdc de los grupos de laboratorios anteriores.



*Figura 2.6 – Adición del archivo maestro Arty\_A7\_Master.xdc.*

2.5 – Modifique las líneas especificas listadas en la figura inferior que nos ayudarán a generar la salida específica necesaria por el puerto serial y sensado/control de las GPIOs.

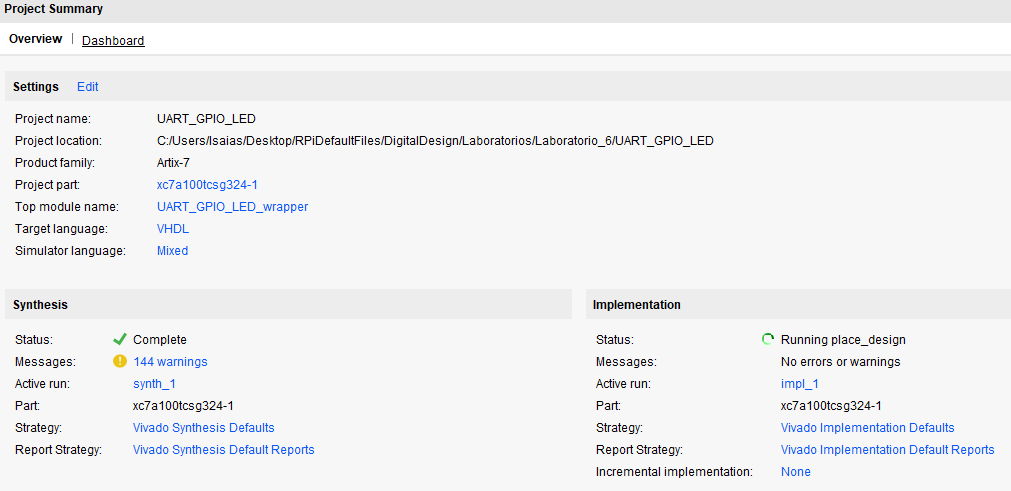






*Figura 2.7 – Modificación del archivo Arty\_A7\_Master.xdc.*

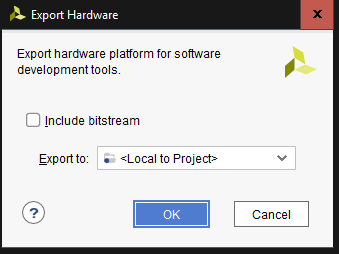
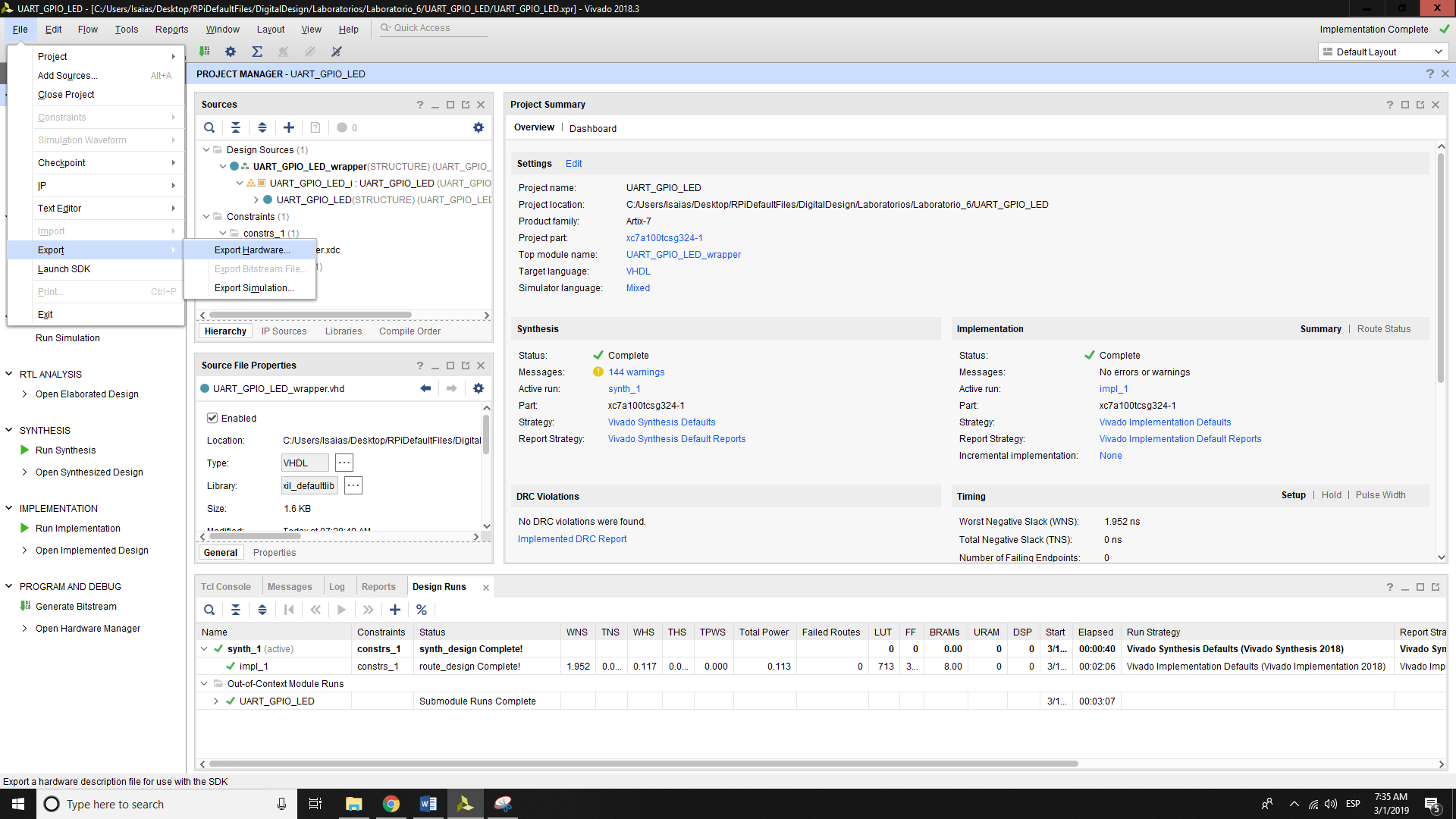
2.6 – Lo siguiente es generar la síntesis e implementación para corroborar el diseño, así que pasaremos por las diferentes partes de flujo realizada anteriormente. Primero realizaremos la corrida de síntesis, corrida de implementación, no genere el bitstream aún.



*Figura 2.8 – Los pasos de generación de síntesis e implementación.*

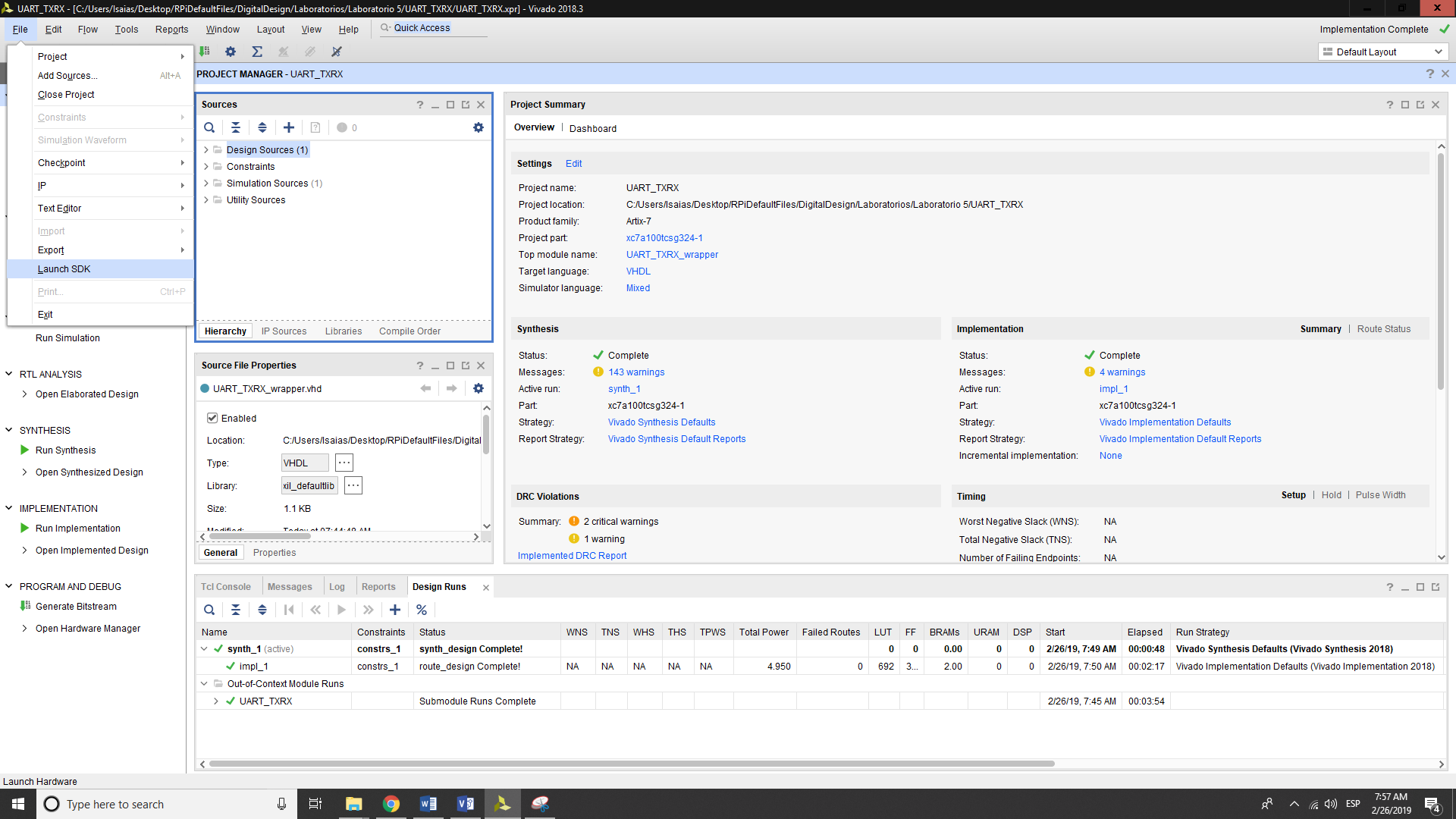
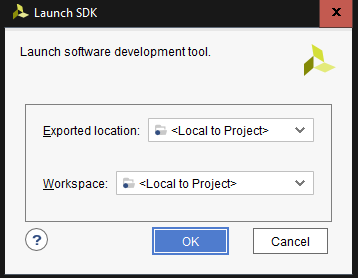
3 – Xilinx Development Kit (XSDK)

3.1 – 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 3.1 – Exportación del diseño de hardware.*

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

*Figura 3.2 – Como lanzar Xilinx SDK.*

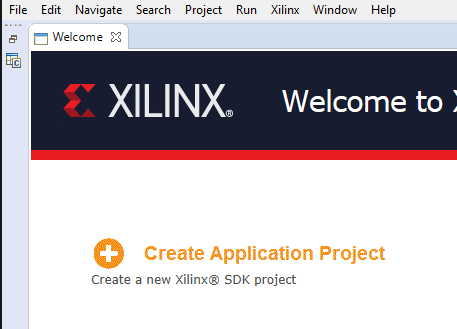
3.3 - El XSDK demorará un poco así que no desespere y tenga paciencia.



*Figura 3.3 – Ventana de XSDK.*

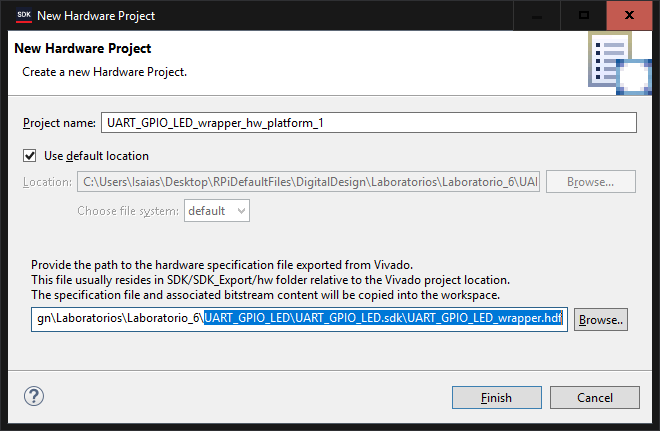
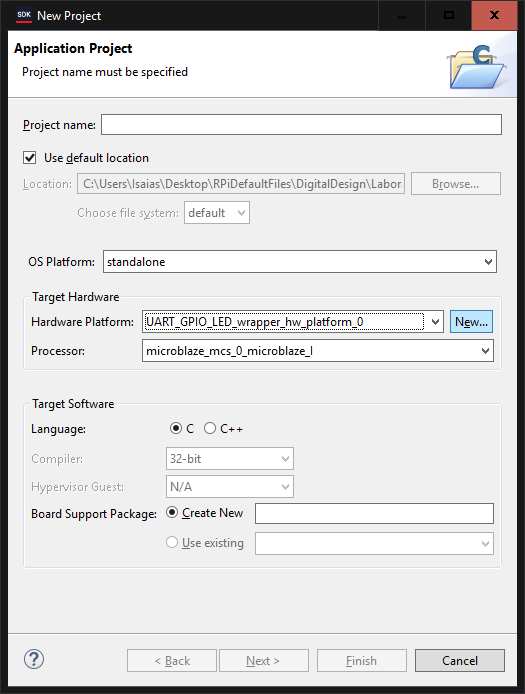
4 – Creación del proyecto en Xilinx SDK

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



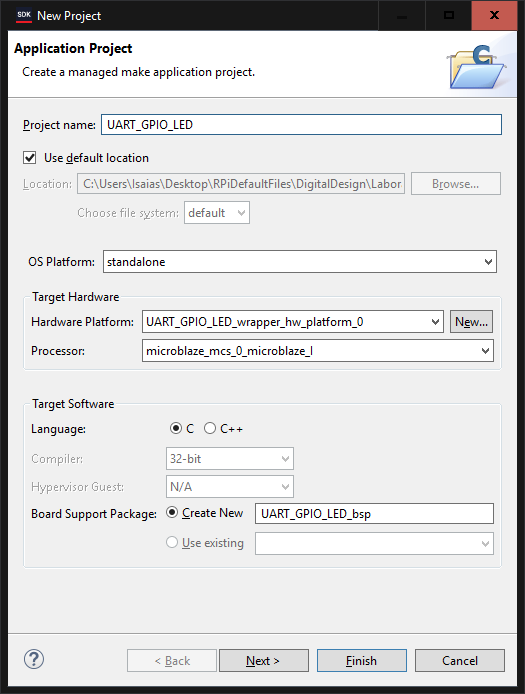
*Figura 4.1 – Ventana de XSDK.*

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.



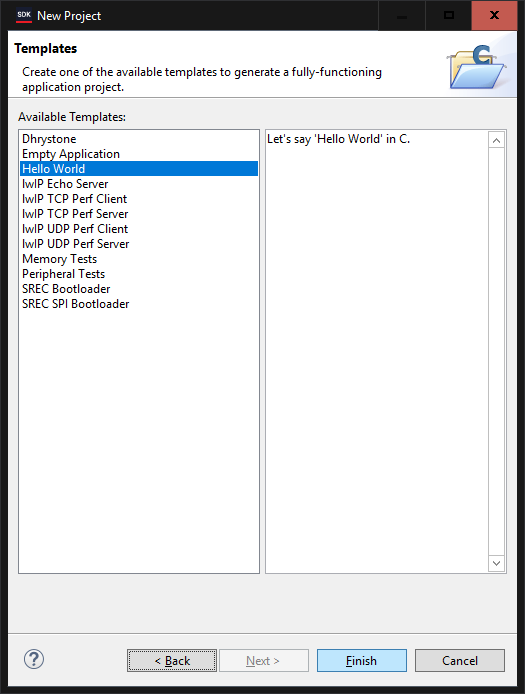
*Figura 4.2 – Ventana de Nuevo Proyecto de Hardware. Si no encuentra XSDK el paquete creado, esta será la manera de encontrarlo, presionar “New” y buscar en el folder de proyecto bajo la carpeta .sdk el resultado de hardware.*

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.



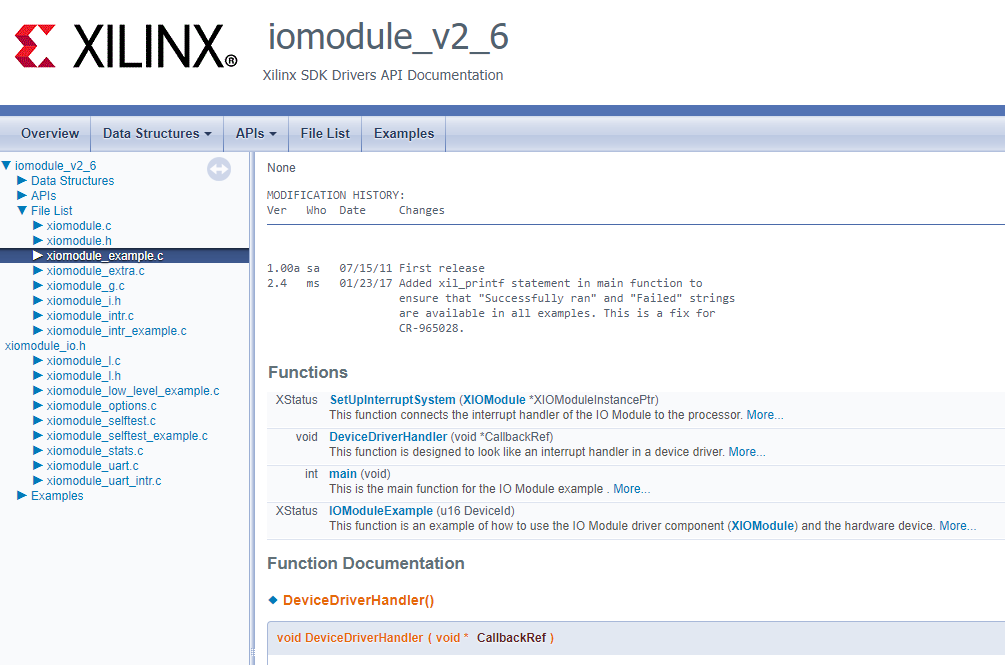
*Figura 4.3 – Ventana de nuevo proyecto de aplicación. Elegir C, asegurarse que el hardware es el que ud ha creado en Vivado y que es la instancia de microblaze correcta.*

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.



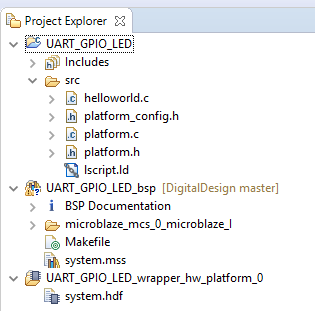
*Figura 4.4 – Ventana de nuevo proyecto, selección de aplicación. Elegir Hello World como base para la construcción de nuestra aplicación final.*

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*

4.7 – Modifique el código de “helloworld.c” para como sigue en la siguiente figura, realizaremos envío de datos y recepción por el puerto serial.

*Listado 4.1 – Archivo HelloWorld.c. Modificar el archivo para realizar envío de datos y sensado de datos para visualizar en la computadora por medio del puerto serial.*

#include <stdio.h>

#include <string.h>

#include "xparameters.h"

#include "xil\_cache.h"

#include "xiomodule.h"

int main()

{

int done = 0;

int byte\_count = 0;

int execute\_cmd;

char command[6];

char led\_cmd[3];

char led\_num[2];

u8 tmp\_rx\_buf;

u8 rx\_buf[40];

u32 button\_data = 0;

u32 switch\_data = 0;

u32 led\_data = 0;

u32 data;

XIOModule iomodule;

Xil\_ICacheEnable();

Xil\_DCacheEnable();

// Initialize module

data = XIOModule\_Initialize(&iomodule, XPAR\_IOMODULE\_0\_DEVICE\_ID);

data = XIOModule\_Start(&iomodule);

//data = XIOModule\_CfgInitialize(&iomodule, NULL, 1);

xil\_printf("CFInitialize returned (0 = success) %d\n\r", data);

print("---Enter a command---\n\r");

// While loop until user enters "finish" command

while(done == 0)

{

execute\_cmd = 0;

memset(rx\_buf, 0, sizeof(rx\_buf));

byte\_count = 0;

// Build up message from UART Terminal

while (execute\_cmd == 0)

{

// Read UART data

while ((data = XIOModule\_Recv(&iomodule, &tmp\_rx\_buf, 1)) == 0);

rx\_buf[byte\_count] = tmp\_rx\_buf;

if (rx\_buf[byte\_count] == '\n')

execute\_cmd = 1;

byte\_count++;

}

// Build the command arrays

memcpy(command, &rx\_buf[0], 6);

memcpy(led\_cmd, &rx\_buf[0], 3);

memcpy(led\_num, &rx\_buf[4], 2);

// Read the button & switch status

button\_data = XIOModule\_DiscreteRead(&iomodule, 1);

switch\_data = XIOModule\_DiscreteRead(&iomodule, 2);

// Execute the received command

if (strcmp(led\_cmd, "led") == 0)

{

led\_data = led\_num[1] - 48;

if ((led\_num[0] - 48) == 1)

led\_data += 10;

XIOModule\_DiscreteWrite(&iomodule, 1, led\_data);

}

else if (strcmp(command, "button") == 0)

xil\_printf("Button Status: %d\n\r", button\_data);

else if (strcmp(command, "switch") == 0)

xil\_printf("Switch Status: %d\n\r", switch\_data);

else if (strcmp(command, "finish") == 0)

done = 1;

else

print("Command is invalid, re-enter below:\n\r");

// Debugging print outs below

// print("Command: ");

// print(command);

// print("\n\r");

//

// print("LED\_cmd: ");

// print(led\_cmd);

// print("\n\r");

//

// print("LED\_num: ");

// print(led\_num);

// print("\n\r");

}

print("---Exiting main---\n\r");

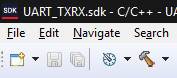
Xil\_DCacheDisable();

Xil\_ICacheDisable();

return 0;

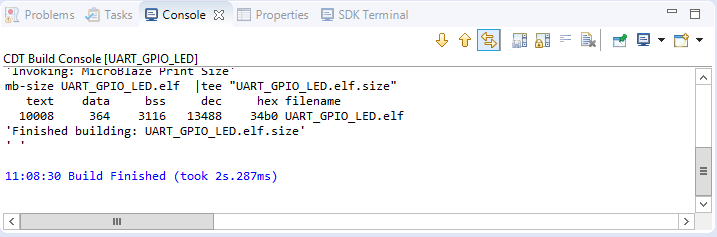
}

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



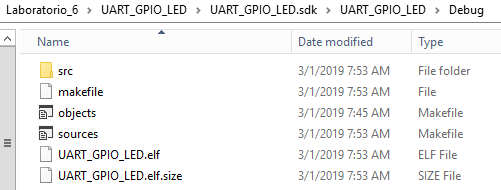
*Figura 4.7 – Ubicación de la herramienta build. La herramienta build sirve para compilar el proyecto.*

4.9 – 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 13488 bytes.



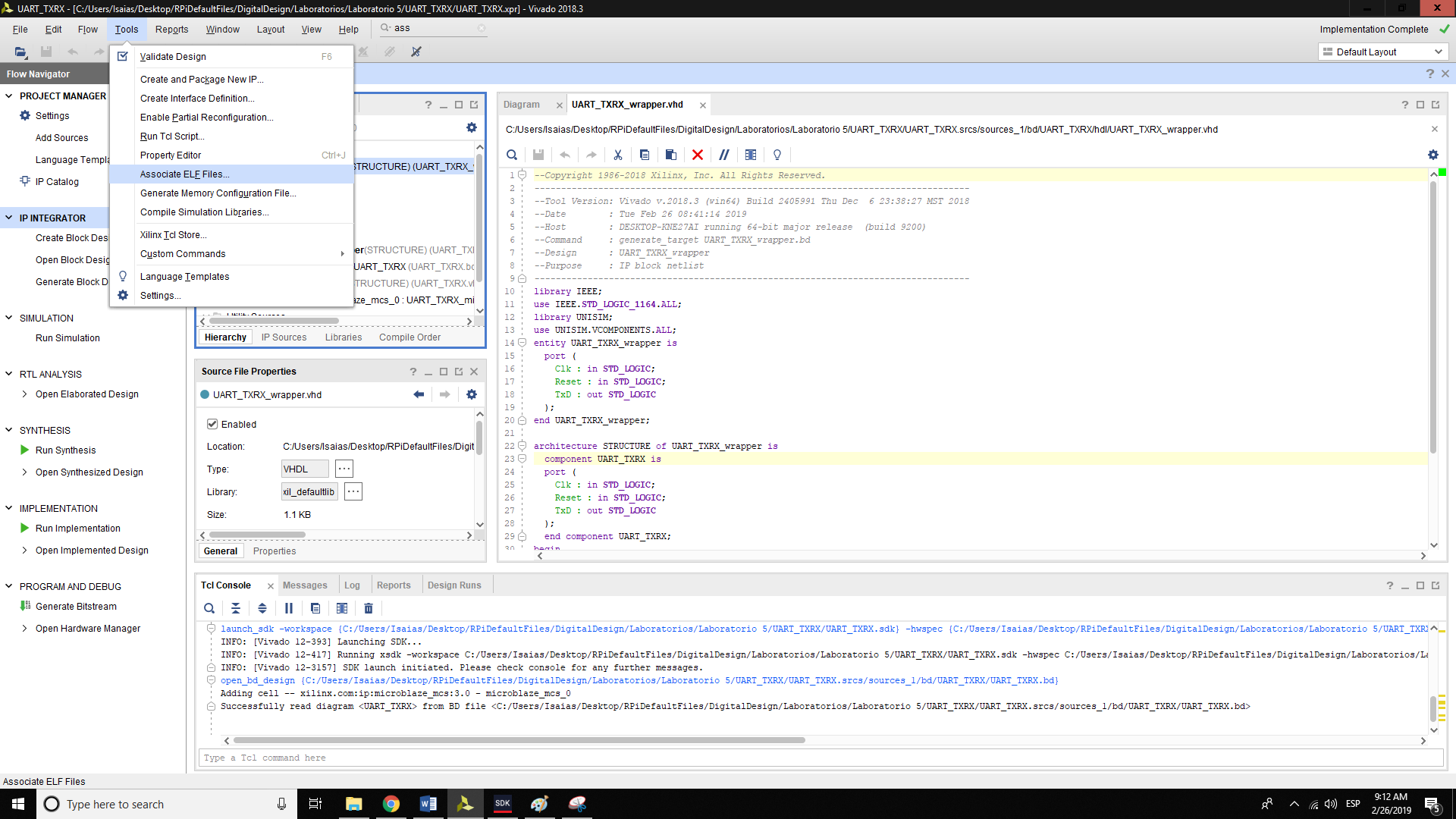
*Figura 4.8 – Finalización exitosa de proyecto. Note el tamaño de la aplicación que es menor al especificado en Microblaze MCS, de lo contrario nos dará error.*

4.10 - Como observa, nos generó un archivo .elf de casi 14Kb, que será el que quemaremos en el FPGA de la siguiente dirección:



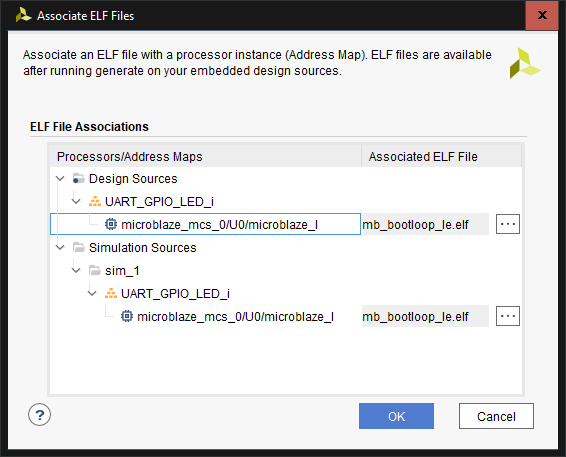
*Figura 4.9 – Ubicación del archivo .elf. El archivo .elf es necesario para construir la aplicación final del FPGA.*

4.11 – 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.

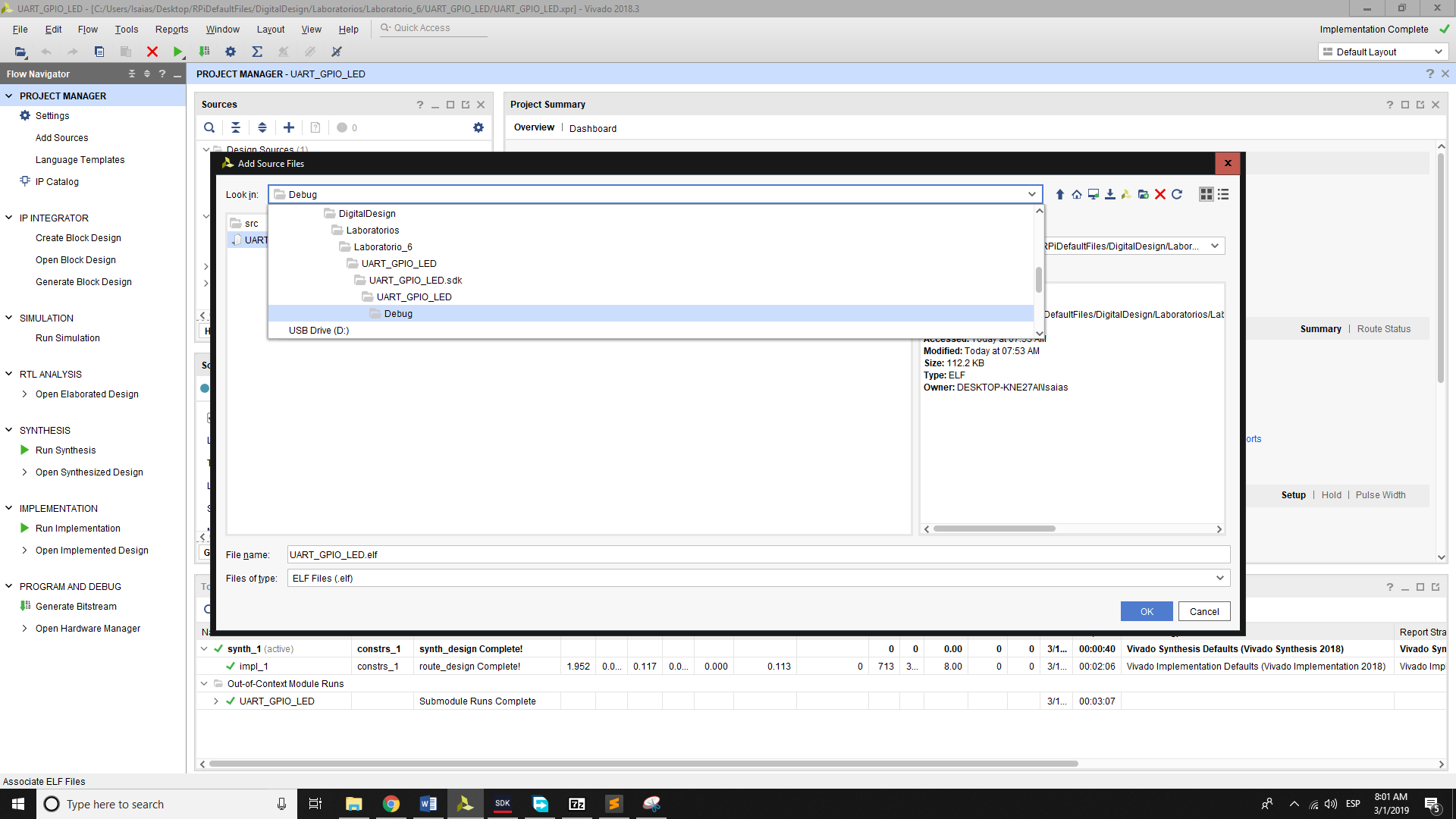


*Figura 4.10 – Asociación del archivo ELF. El archivo ELF generado en XSDK nos servirá para generar el bitstream final del proyecto.*

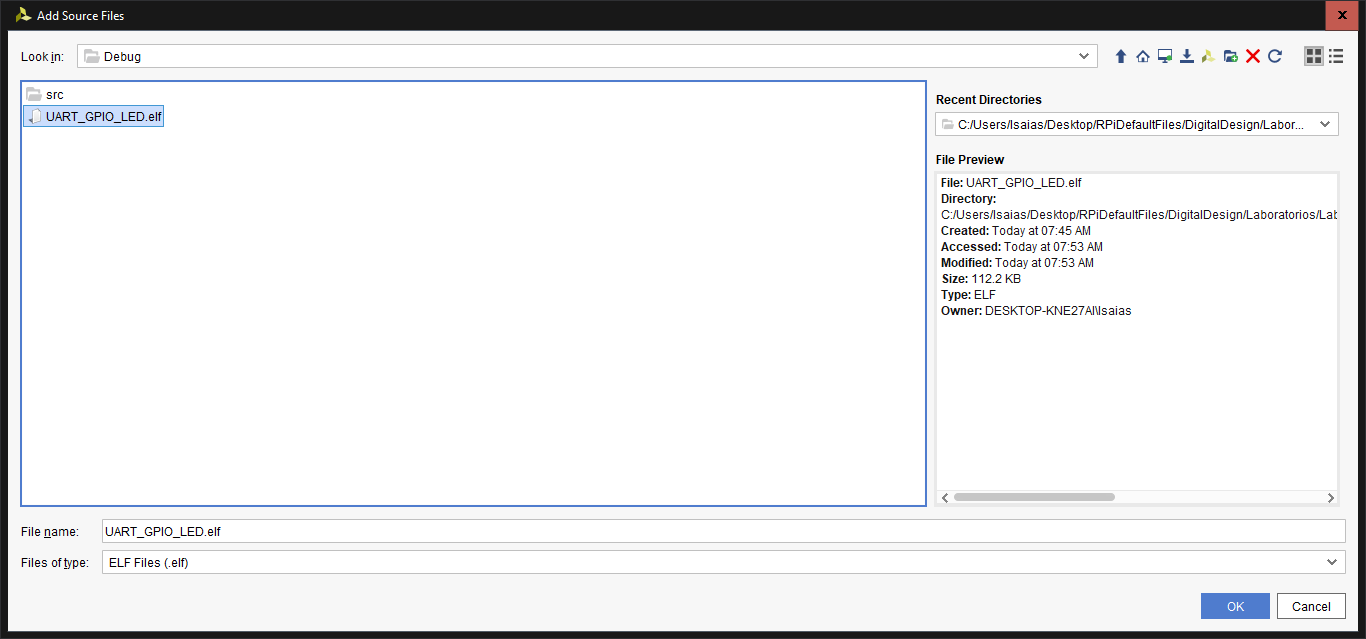
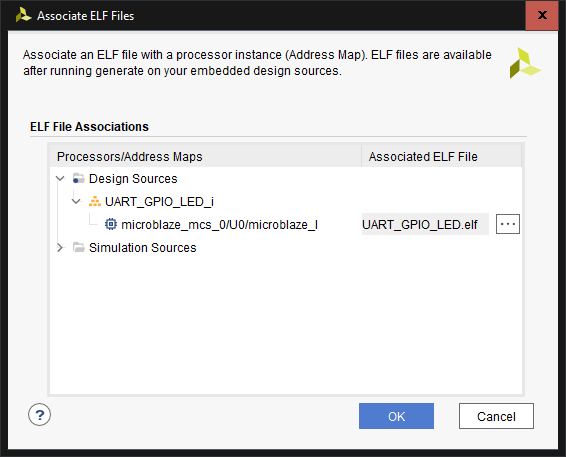
4.12 – Se abrirá una ventana que nos dará dos listas de .elf asociados. No nos interesa las fuentes de simulación sino las fuentes de diseño, por consiguiente, asocie el .elf de la figura 4.10 en Design Sources



*Figura 4.11 – Asociación del archivo ELF. El archivo ELF generado en XSDK nos servirá para generar el bitstream final que tiene la aplicación a ejecutar.*

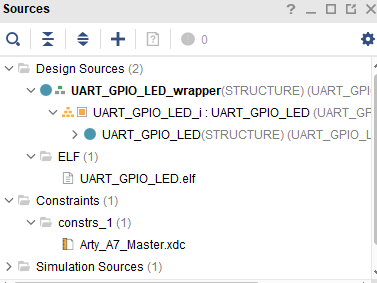


*Figura 4.12 – Buscar el archivo .elf. En la carpeta de proyecto navegar a <proyecto>.sdk/<proyecto\_en\_SDK>/Debug*

*Figura 4.13 – Encontrar el archivo .elf y asociar.*

4.11 – La ventana final debe estar como sigue:

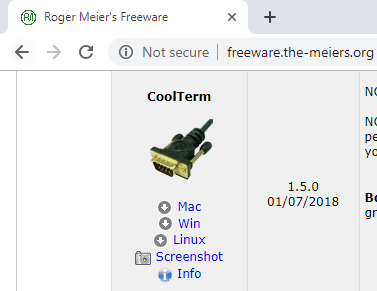


*Figura 4.13 – Adición del archivo .elf al proyecto. Vista final del archivo a ejecutar.*

4.12 – 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.

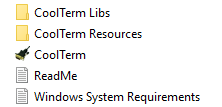
5 – Demostración del Código y CoolTerm

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



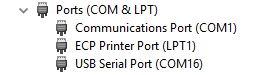
*Figura 5.1 – CoolTerm. Emulador de terminal serial multiplataforma.*

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



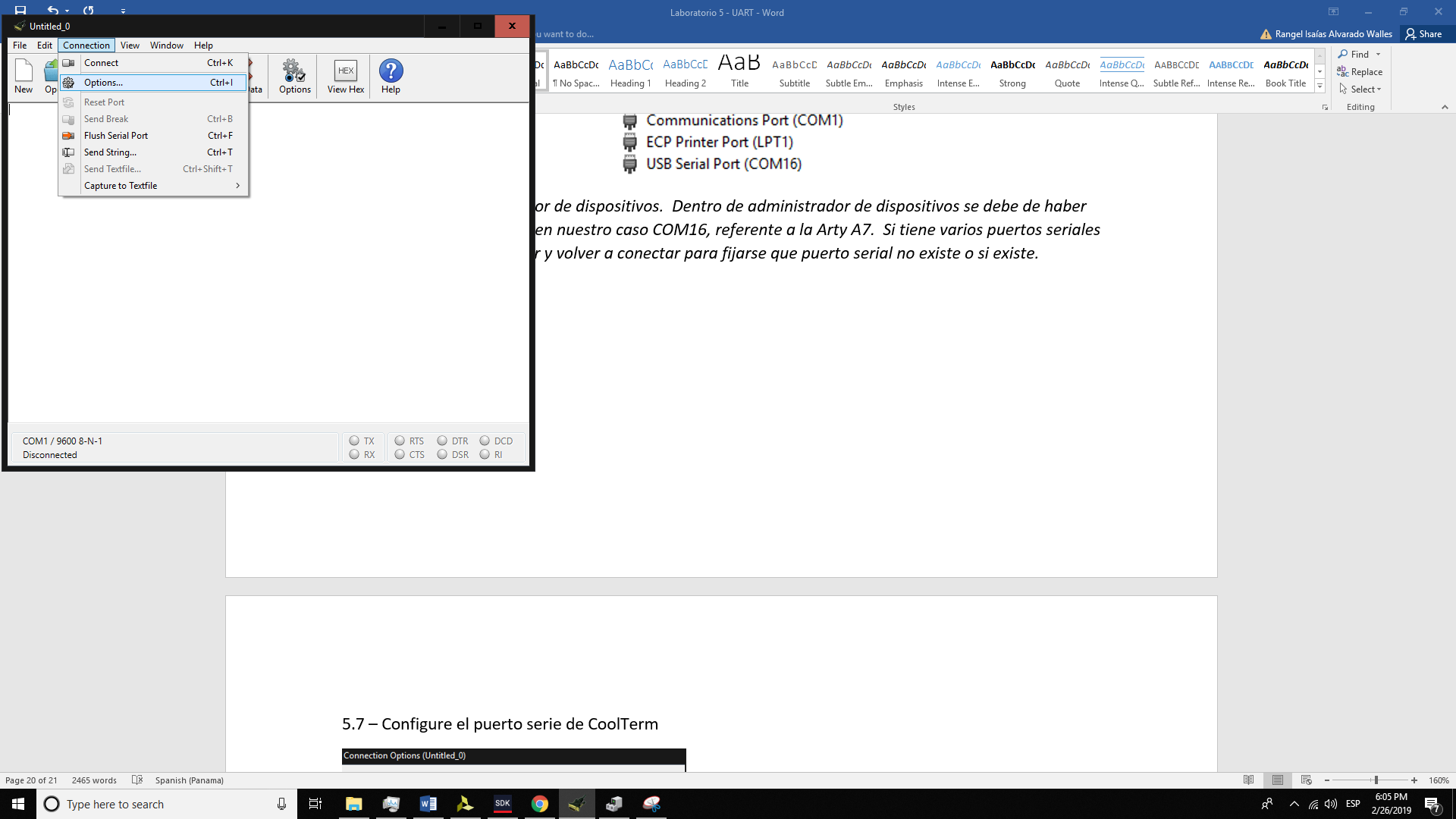
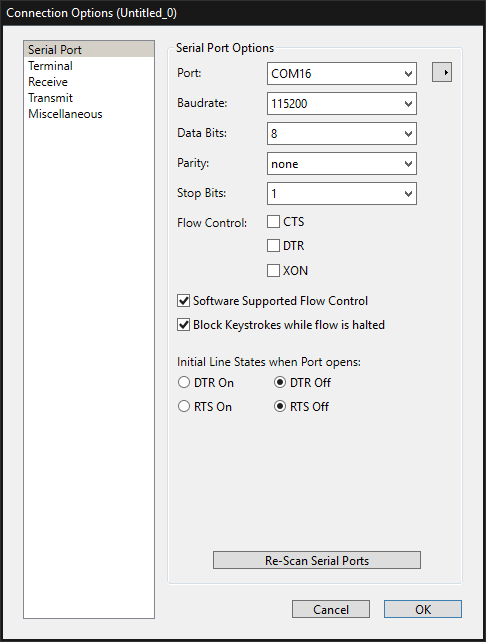
*Figura 5.2 – Ejecutable de CoolTerm. Inicie la aplicación. La ventaja de CoolTerm es que no es un instalable y puede borrarse cuando lo desee.*

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



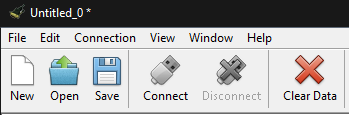
*Figura 5.3 – Administrador de dispositivos. Dentro de administrador de dispositivos se debe de haber creado un USB Serial Port, en nuestro caso COM16, referente a la Arty A7. Si tiene varios puertos seriales puede desconectar y volver a conectar para fijarse que puerto serial no existe o si existe.*

5.4 – 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 5.4 – 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.

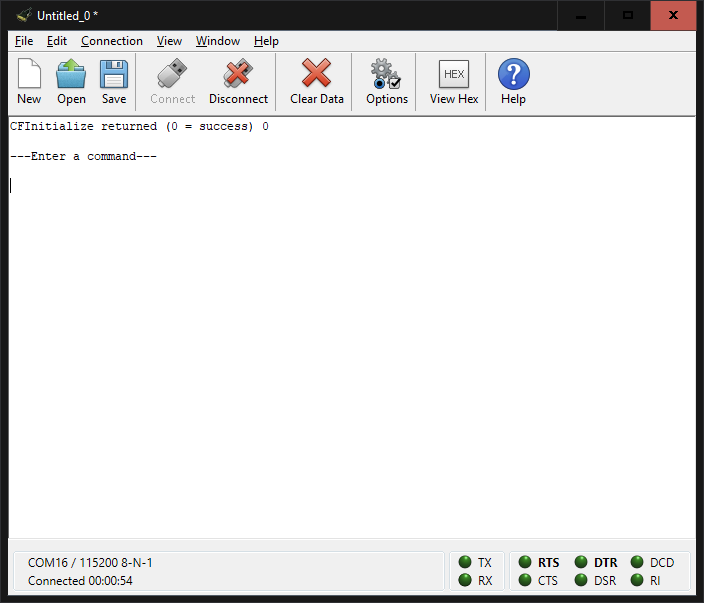


*Figura 5.5 – Presione el botón Connect para abrir el puerto serial (en la PC) de la Arty A7.*

**5.5 – 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.**

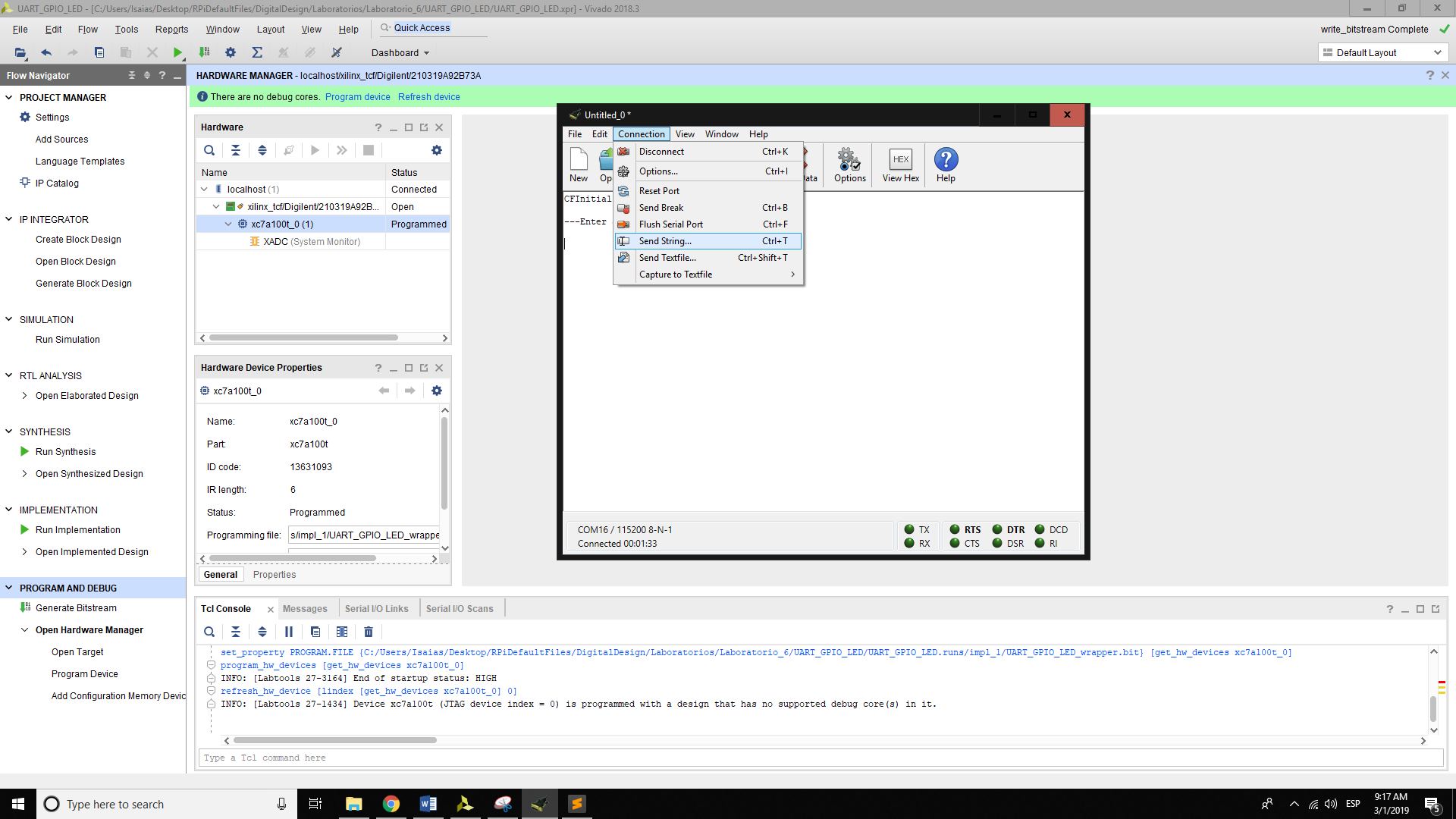
5.6 – Descargue el bitstream como lo ha realizado en laboratorios anteriores.

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

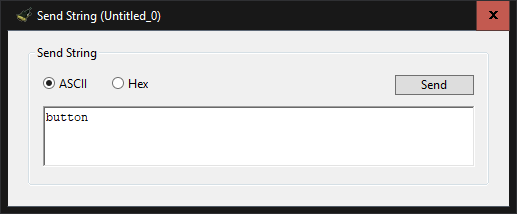


*Figura 5.8 – Vistazo final a la aplicación de comunicación del puerto serial.*

5.8 – En “Connection/Send String” Ingrese los posibles commandos, cambie el estado de los switches y observe lo que sucede, el estado decimal debe transformarse a binario que corresponde para switches, botones y LEDs.



*Figura 5.9 – Trate de enviar un comando con Send String.*



NOTA: Para enviar el string debe escribir button y presionar ENTER, luego el botón Send.

*Figura 5.10 – Envíe el String. Recuerde añadir los caracteres de final de línea y retorno de carro con ENTER, luego presione Send.*

5.9 – Los comandos permitidos son los siguientes:

NOTA:

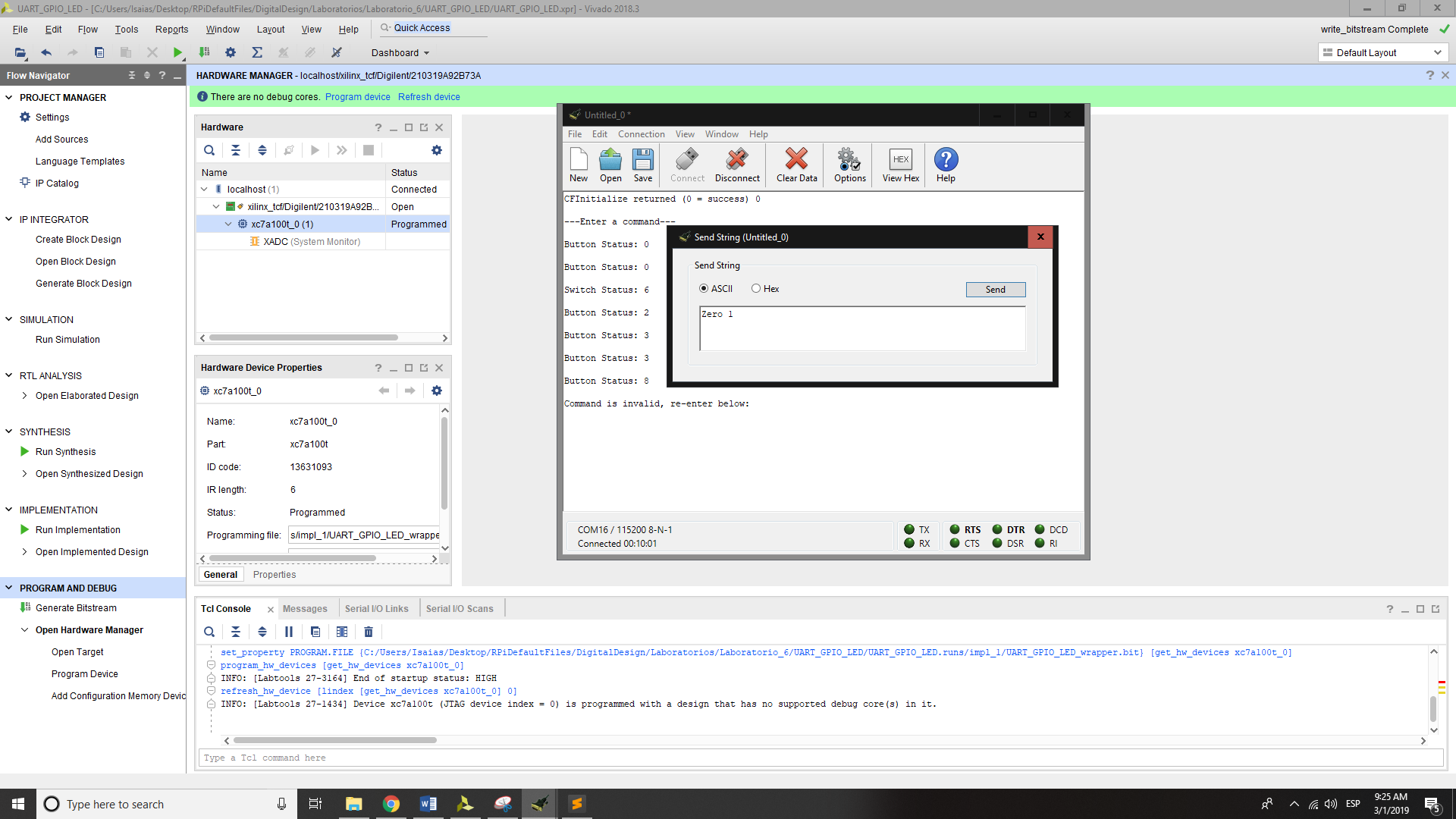
XX = dos dígitos ASCII entre 00 a 15.

<ENTER> es la tecla, no escribir el ASCII sino presionarlo para que entienda los caracteres 0D 0A ASCII.

button<ENTER>

switch<ENTER>

led xx<ENTER>



*Figura 5.11 – Los diferentes comandos y sus respuestas. Al tener un comando no dentro de la tabla de comandos dará un resultado inválido.*

6 - Evaluación

* 50% - Terminar el proyecto anterior
* 25% - Modificar los baudios según la tabla dada a continuación y demostrar en el FPGA

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Reset (SW) | Baudrate |
| 1 | AHIR . BHAVINI | 2 | 38400 |
| 2 | ALVARADO VILLEGAS TSAREV BRYANT | 3 | 19200 |
| 3 | AVILA WONG CHRISTIAN | 0 | 900 |
| 4 | CABALLERO CHECA CARLOS MIGUEL | 3 | 115200 |
| 5 | CASTRO INAUDY JORGE LUIS EFRAIN | 0 | 300 |
| 6 | CEBALLOS SAYAS MARVIN HELAM | 3 | 600 |
| 7 | CEDEÑO PONCE CLAUDIA CECILIA | 2 | 57600 |
| 8 | GUEVARA ROMERO SOFIA | 1 | 38400 |
| 9 | LEMUS ESCOBAR STHEFANIE ROXXANE | 1 | 900 |
| 10 | LUGO DURAN GUILLERMO RAY G. | 2 | 1200 |
| 11 | MARIN ARJONA NOHELY ENITH | 0 | 19200 |
| 12 | NUÑEZ MARTINEZ NOHELYS NAZARETH | 1 | 600 |
| 13 | RODRIGUEZ ALVARADO REMIGIO | 3 | 57600 |
| 14 | RODRIGUEZ MARTINEZ YARELIS DEL CARMEN | 2 | 600 |
| 15 | SANTAMARIA GUERRA VICTOR MANUEL | 1 | 115200 |
| 16 | TEJADA WILSON ANDREA RACHELL | 2 | 57600 |
| 17 | VARELA RETTALLY JAROD NESSIM | 3 | 900 |

* 25% - Agregar un comando que haga parpadear un LED.