Escuela Politécnica Superior,
Grado en Informática

Asignatura: Diseño Automático de Sistemas

Práctica 4
Implementación de un softcore

26 de Abril de 2022

UNIVERSIDAD
NEBRIJA

UNIVERSIDAD
NEBRIJA

Pablo Sarabia Ortiz

Índice/Tabla de contenidos

1. Introducción

1.1. Presentación

Se va a desarrollar la implementación de un procesador *softcore*, sintetizado en la FPGA, para desarrollar los conocimientos vistos en clases teóricas. Se ha seleccionado Microblaze como el microcontrolador para implementar en la FPGA. Se van a utilizar las técnicas de codiseño software/hardware. Y utilizar el flujo de trabajo con las herramientas de Vitis y Vivado.

1.2. Bibliografía recomendada

Como contenido extra se puede leer la información adicional de Microblaze disponible en Xilinx, enlace. También se puede consultar la guía proporcionada por Digilent que indica el flujo de trabajo con Vitis y Vivado, enlace.

1.3. Objetivo

Se quiere mostrar como se puede hacer un diseño que conste de una parte software. El objetivo es obtener una mayor flexibilidad en nuestros diseños digitales y un tiempo de desarrollo más corto en nuestros diseños.

1.4. Versión de las herramientas

NOTA IMPORTANTE: Esta guía se ha realizado usando Vivado 2021.2, debería ser valido a partir de la versión 2018. En versiones anteriores se utilizaba Xilinx SDK, en general la guía es aplicable pero algunos pasos pueden variar.

2. Instalación de Vitis

Para poder programar el softcore (Microblaze) es necesario tener instalado la aplicación de Vitis. Para comprobar si está instalada lo más sencillo es hacer click en Tools->Launch Vitis IDE, mostrado en Figura 1.

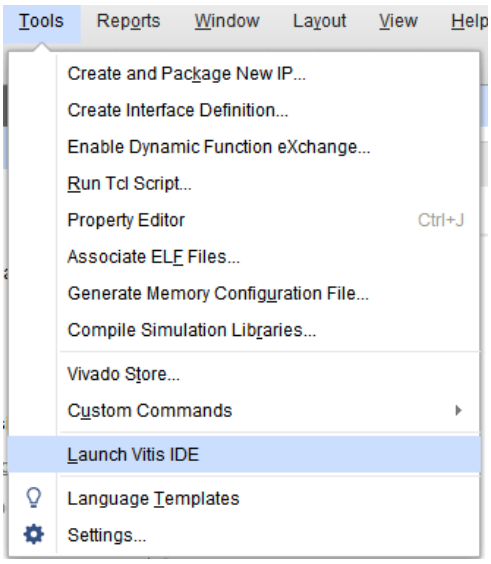

Figura 1. Lanzar Vitis IDE

Si al hacer click salta un error, no tienes instalado Vitis, para añadirlo basta con volver a ejecutar el instalador, el instalador se llama Add Design Tools 2021.2, véase Figura 2 .

Figura 2. Herramienta de instalación de nuevas herramientas de Vivado.

Una vez acabado este proceso deberíamos tener lista nuestras herramientas de trabajo.

3. Flujo de trabajo

Cuando se realiza un codiseño el flujo de trabajo es el siguiente:

1. Usamos Vivado para diseñar nuestro HW, añadimos nuestro diseño digital y las constraints necesarias.
2. Incluimos las IPs necesarias (CPU, RAM, UART, etc..).
3. Generamos el bitstream.
4. Exportamos el bitstream en formato ".xsa".
5. Usando Vitis generamos un proyecto de plataforma importando el HW.
6. Generamos el proyecto de aplicación.
7. Escribimos el código en C de nuestra aplicación para el microprocesador.
8. Compilamos y cargamos el programa mediante UART.

4. Diseñar el HW

Lo habitual si se ha seleccionado utilizar una FPGA es que se haya implementado un diseño digital. Lo primero sería dotarle de una interfaz para que se pueda comunicar con el procesador. En esta práctica solo se van a usar bloques IP proporcionados por Xilinx por simplicidad.

Lo primero es generar un bloque de diseño para ello creamos un nuevo proyecto con el target de Nexys 4 DDR e incluir las constrainsts como en prácticas anteriores. Una vez generado el proyecto en el menú de *Flow Navigator -> IP INTEGRATOR -> Create Block Design, véase* Figura 3.

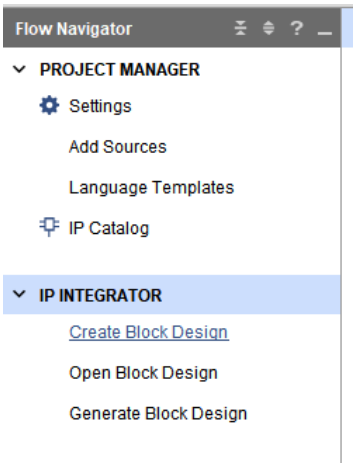

Figura 3. Creación del bloque de diseño.

Al generar nos pedirá introducir un nombre sin espacios, en este caso lo he nombrado como "microblaze_nexys", véase Figura 4.

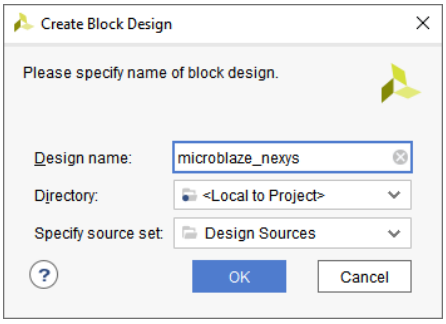

Figura 4. Añadir nombre al bloque de diseño.

A continuación, hacemos click en add IP y seleccionamos Microblaze en el desplegable, Figura 5.

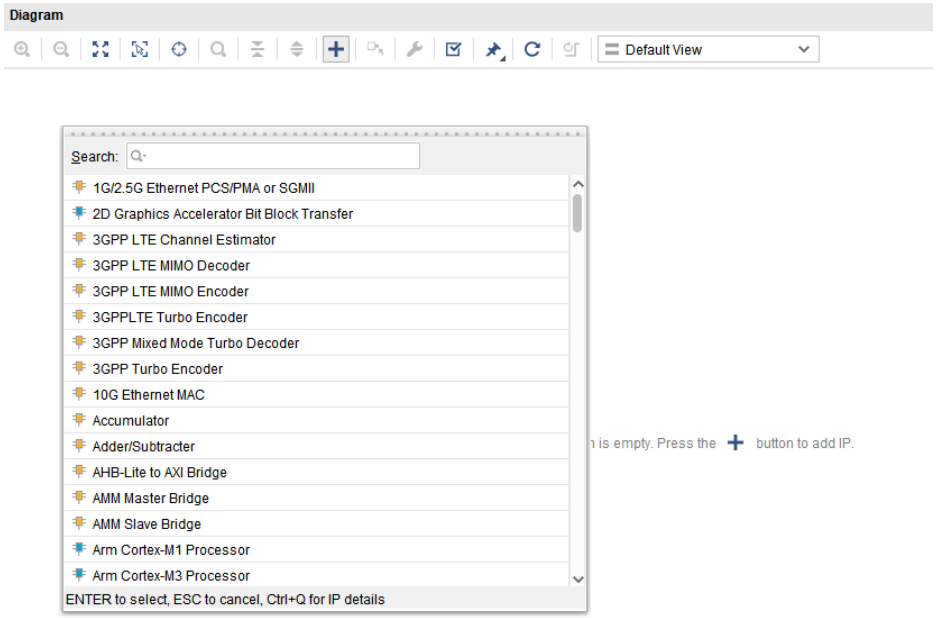

Figura 5. Añadir la IP de Microblaze.

En el diagrama debería verse lo siguiente, Figura 6.

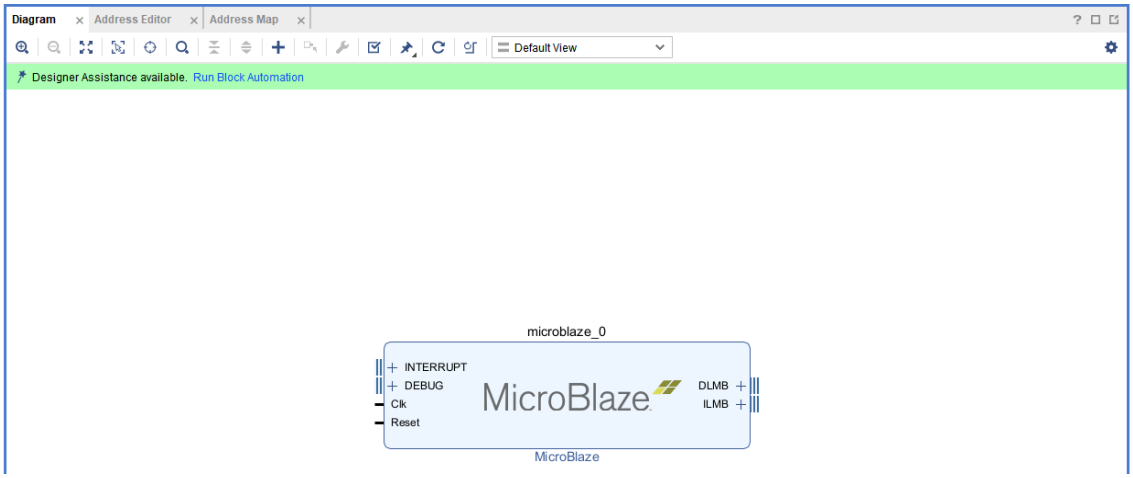

Figura 6. Diseño después de añadir el procesador Microblaze.

Ahora vamos a configurar el procesador. Haciendo click en Run Block Automation y configuramos el procesador hasta que se vea como en la siguiente imagen, Figura 7.

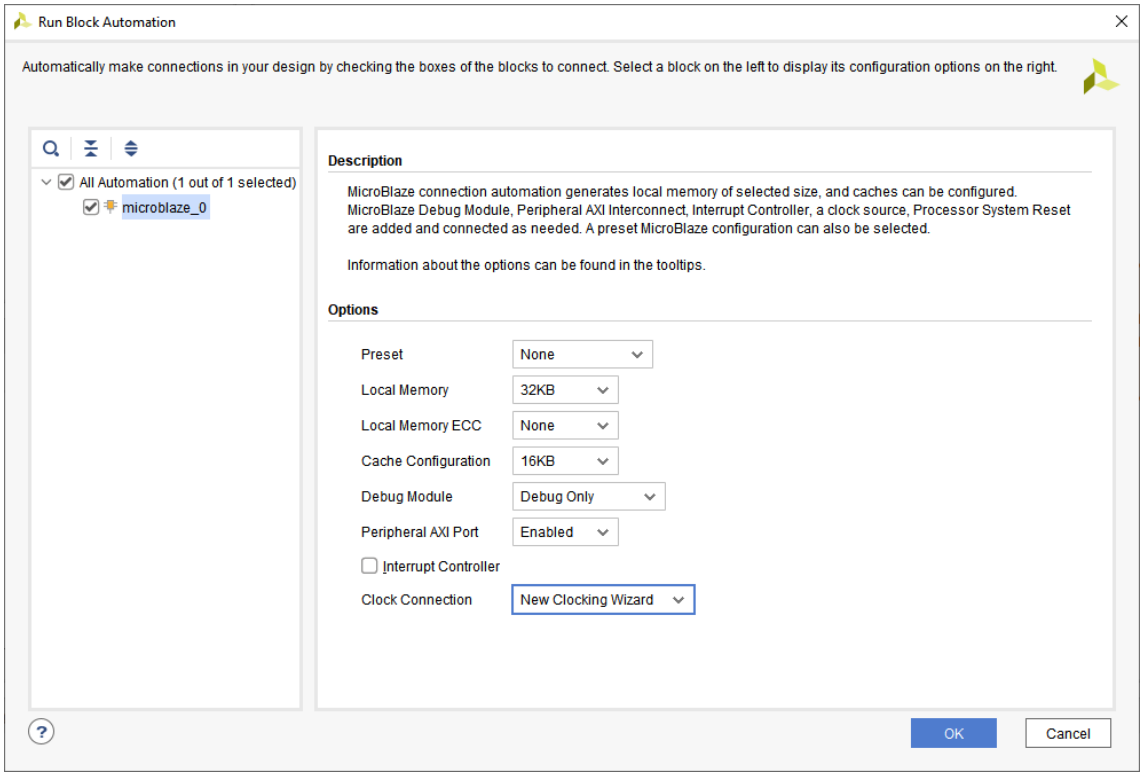

Figura 7. Configuración de Microblaze.

Después debemos hacer click en Run Block Automation, mensaje en verde en la parte superior del espacio de diseño. El resultado se debe ver tal y como se muestra en Figura 8 .

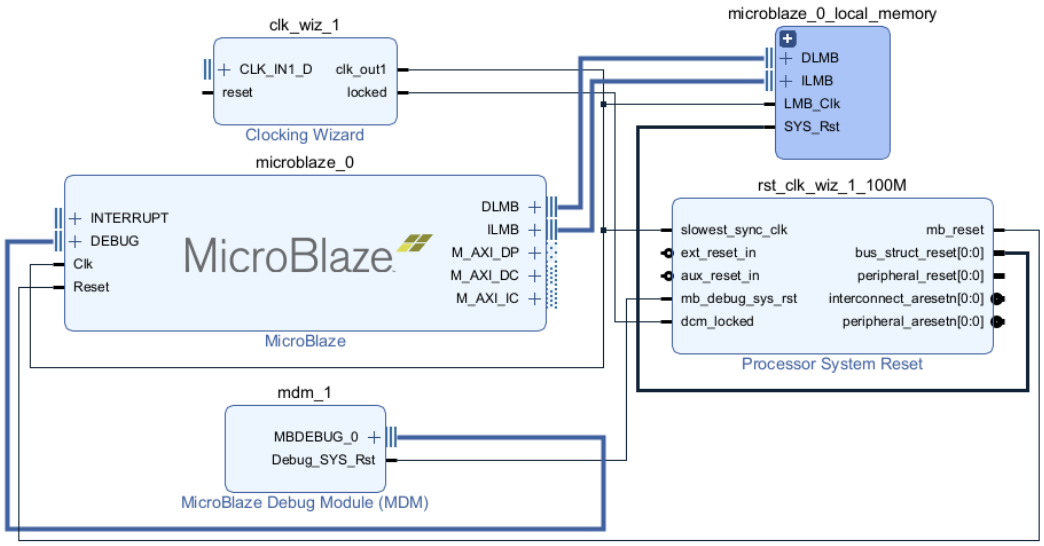

Figura 8. Diseño tras el Block Automation.

Ahora vamos a configurar el bloque que genera las señales de reloj. El procesador va a funcionar a 100 MHz y la memoria funcionará al doble de frecuencia 200 MHz.. Para configurar el rekoj

hacemos doble click en clk_wiz_1 y lo configuramos para que la entrada del reloj sea sys_clock, el reloj propio de la FPGA, Figura 9.

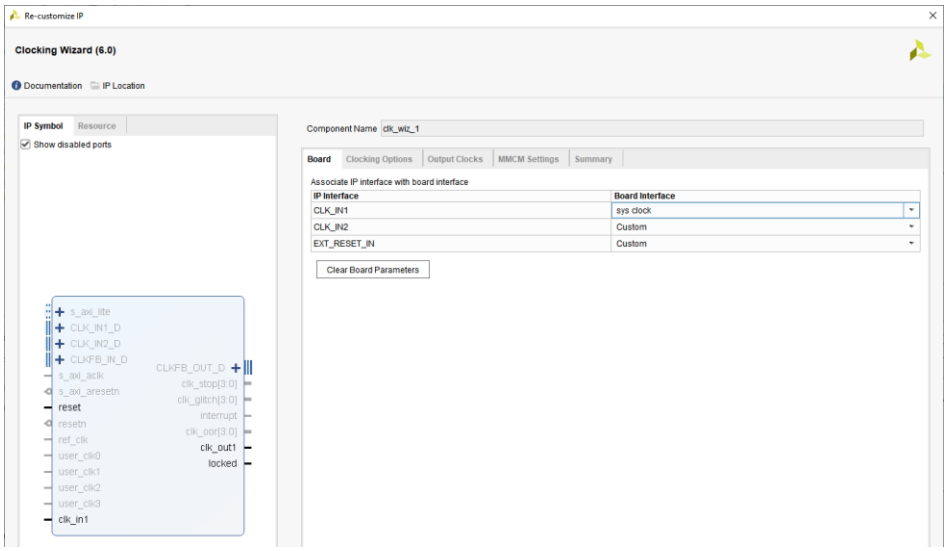

Figura 9. Configurar la entrada del reloj.

A continuación en la pestaña de Output Clocks, vamos a generar las dos señales que necesitamos tal y como se muestra en . Además marcamos el reset como active Low (Pestaña Reset Type en la zona inferior derecha).

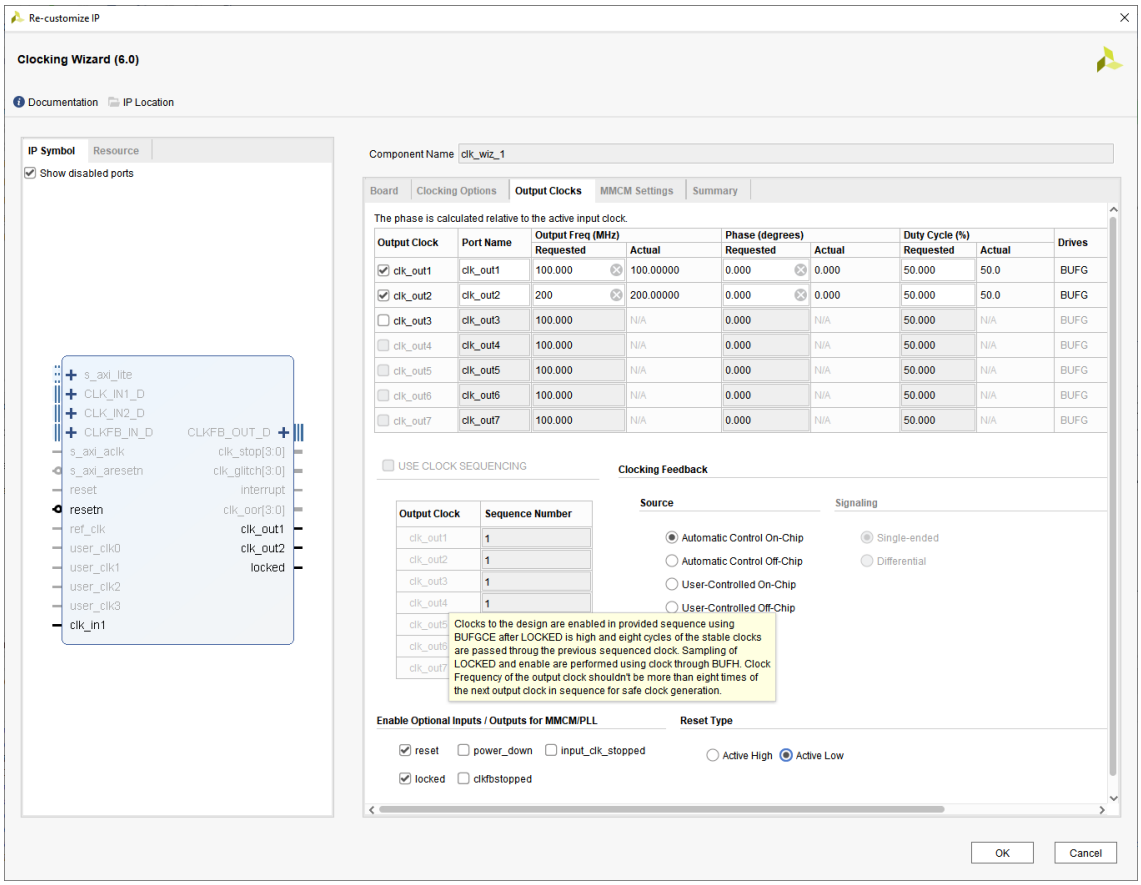

Figura 10. Configurar la salida 2 a 200 MHz y el reset a nivel bajo.

El siguiente paso es añadir la interfaz UART que nos permitirá cargar el programa y seguir su

funcionamiento desde el terminal. La ip se llama “AXI Uartlite”. Una vez añadida hacemos click en Run Connection Automation y seleccionamos todas las conexiones disponibles, como se muestra en Figura 11.

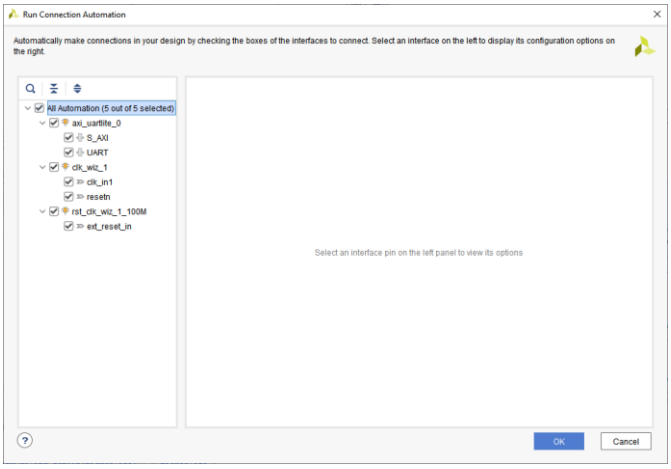

Figura 11. Ventana del Run Connection Automation.

Tras su ejecución el diagrama debería verse tal y como se muestra en la Figura 12.

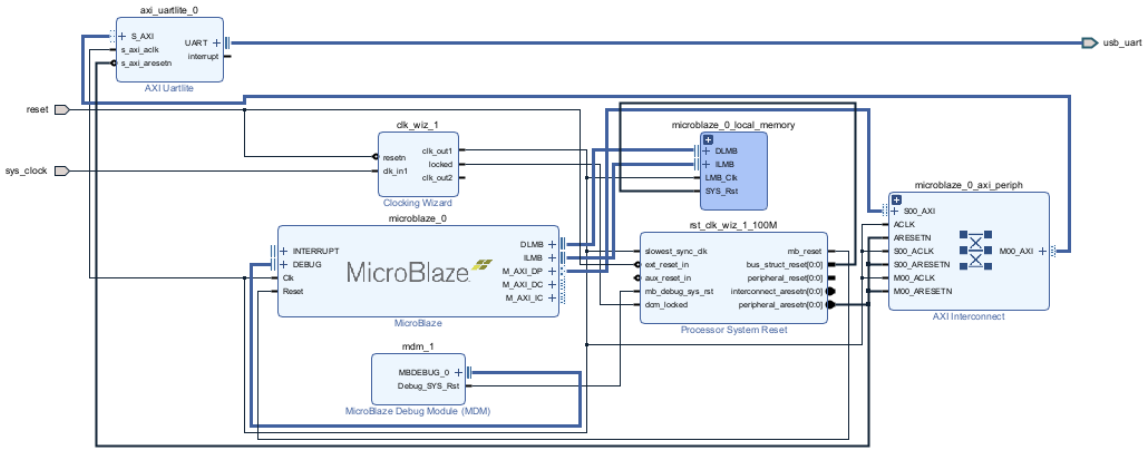

Figura 12. Diagrama del bloque de diseño tras el autoconnect.

Por ahora tenemos un microcontrolador con interfaz UART pero seguimos necesitando una memoria para ejecutar el programa. Para ello vamos a utilizar la RAM DDR2 que contiene la propia placa de desarrollo. Realizar un controlador de memoria RAM no es algo trivial, por tanto vamos a utilizar el controlador de memoria que nos proporciona Vivado. El controlador se llama MIG (Memory Interface Generator). Para ello lo añadimos al diseño como una IP más, buscamos mig en el desplegable de las IPs y lo añadimos al diseño. El resultado debería ser el mostrado en la Figura 13.

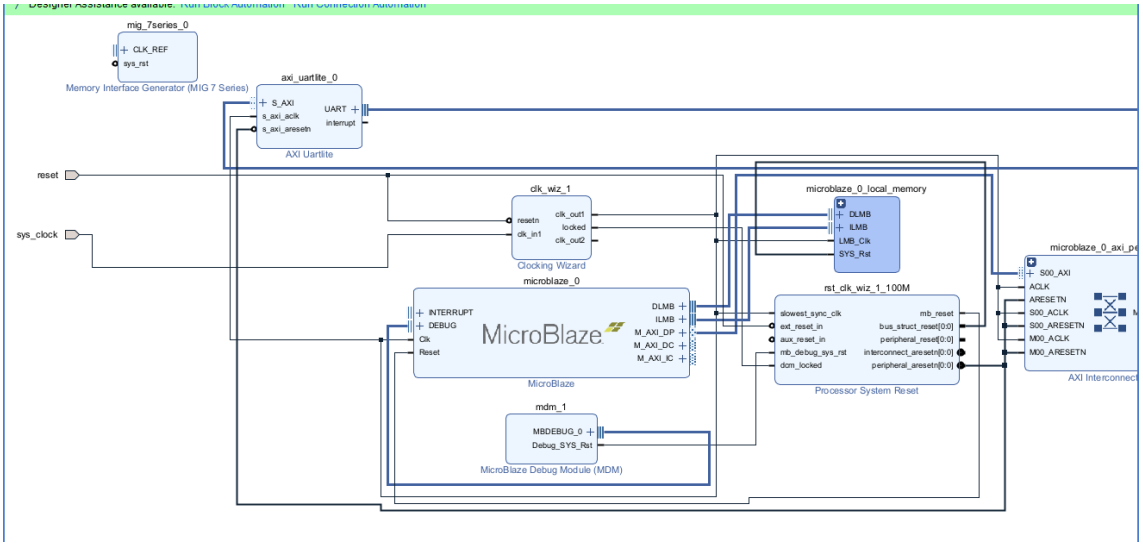

Figura 13. Diagrama tras añadir el controlador de memoria MIG.

Una vez añadido haremos click en Run Block Automation y tras ejecutarse nos aparecerá el siguiente error, mostrado en la Figura 14. Este error es normal y no nos debemos preocupar.

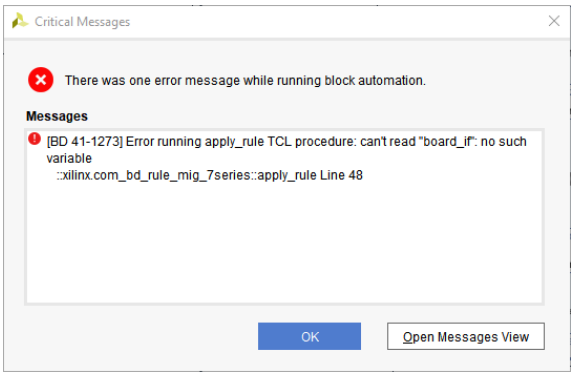

Figura 14. Error tras ejecutar el Run Block Automation.

Ahora volvemos a ejecutar el Run Connection Automation, pero esta vez la ventana que nos aparece nos basta con seleccionar solo el controlador de la memoria: “mig_7series_0”, como se muestra en la Figura 15.

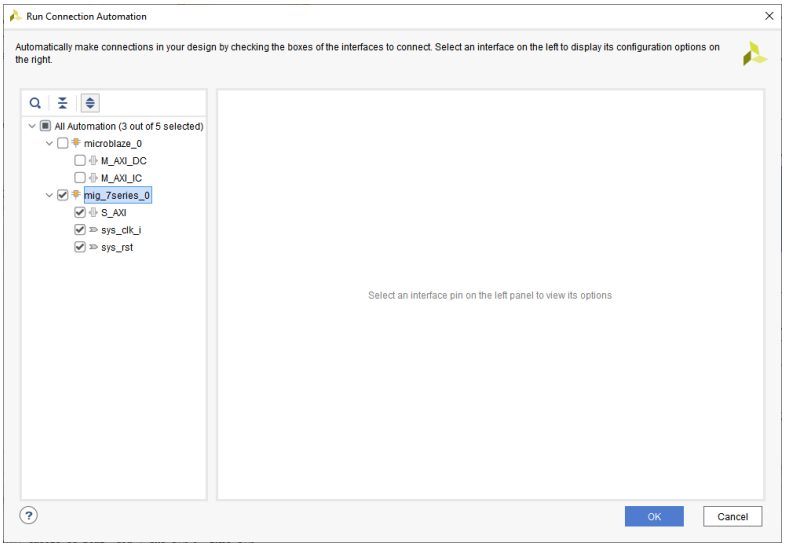

Figura 15. Ventana del Run Connection Automation.

El resultado tras hacer la conexión debería verse como se muestra en la Figura 16.

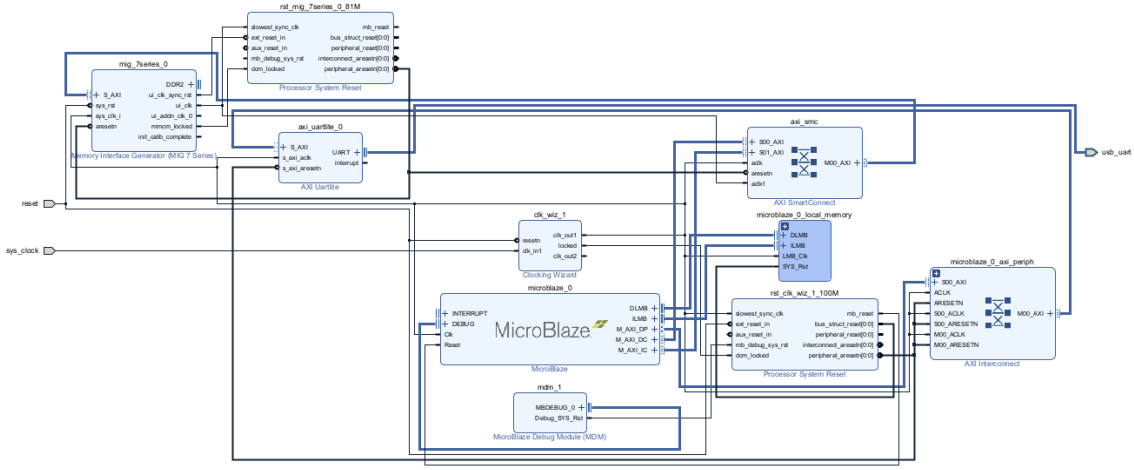

Figura 16. Resultado tras conectar MIG a nuestro diseño.

Había comentado al inicio de la práctica que la RAM funcionaba a 200 MHz y si nos fijamos en el diagrama del bloque de diseño el reloj de la RAM está conectado a la red de reloj que proviene del clk_out1, que se corresponde con el reloj de 100 MHz. Para solucionarlo basta con hacer click derecho sobre el pin "sys_clk_i" del bloque "mig_7series" y en el menú desplegable hacer click en "Disconnect pin". Una vez lo hemos desconectado podemos rutar manualmente haciendo click en la salida "clk_out2" del bloque y arrastrando hasta la entrada "sys_clk_i" del bloque MIG, resaltado en naranja en la Figura 17.

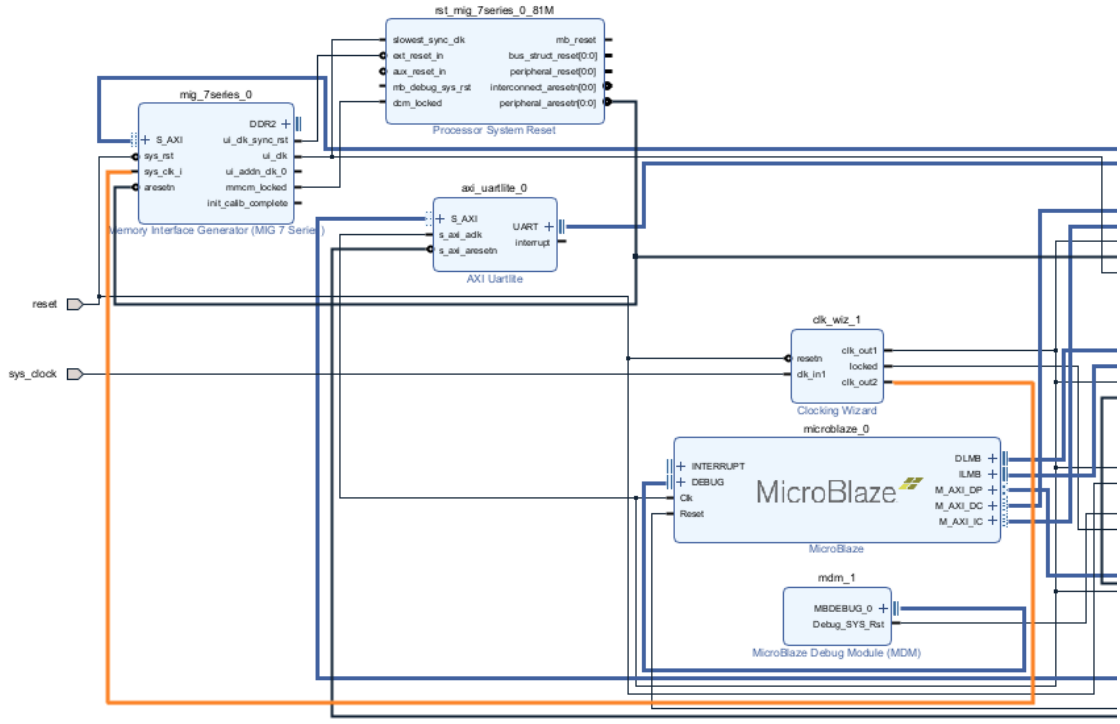

Figura 17. Conexionado entre el clk_wiz y el módulo MIG.

A continuación vamos a reorganizar el esquema para que sea más claro, para ello hacemos click en la parte superior del diagrama donde dice: “Regenerate Layout”, Figura 18.

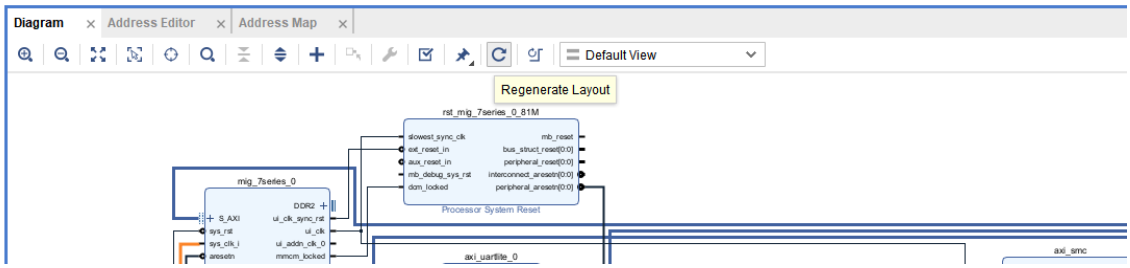

Figura 18. Regenerar el diagrama para visualizar mejor.

Ahora vamos a indicarle a Vivado que las conexiones de la memoria RAM son externas para ello debemos hacer click derecho sobre el pin DDR2 en el bloque mig_7series y en el menú contextual hacer click en “Make external”, como se muestra en la Figura 19.

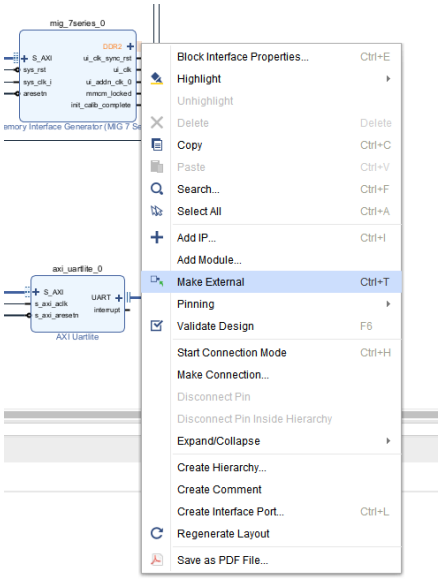

Figura 19. Hacer el pin de la RAM externo.

Una vez realizado este proceso vamos al menú izquierdo del *Block Design* y en la pestaña *Board*, buscamos la interfaz "DDR2 SDRAM", como se muestra en la Figura 20.

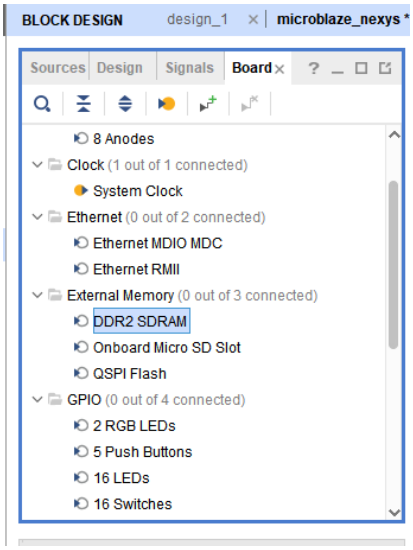

Figura 20. Interfaces de la placa disponibles.

Arrastramos la interfaz hasta el controlador "mig_7series" que aparecerá en verde, mostrado en la Figura 21 .

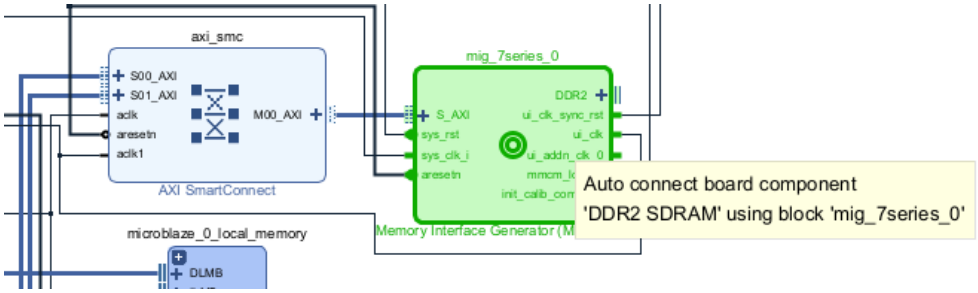

Figura 21. Proceso de añadir la interfaz de la RAM al controlador MIG.

Nos debe aparecer un mensaje indicando que se ha conectado correctamente el componente. El siguiente paso es añadir las constraints de la placa. Las constraints son el fichero que indica a Vivado los puertos del diseño a que pines van. Primero añadimos las constraints como hemos hecho en prácticas anteriores, es el fichero xdc que proporcioné en la primera práctica.

En este diseño vamos a tener los siguientes pines externos: reset, sys_clock, usb_uart_rxd y usb_uart_txd. Debemos descomentar las líneas que se corresponden a esos pines y escribir el nombre de los puertos del diseño entre los corchetes tras get_ports. El botón de reset lo vamos a conectar al pin C12 (puedes consultar el pin donde pone PACKAGE_PIN XXX) donde XXX es el nombre del puerto. El usb_uart_rx va conectado al pin C4 y el usb_uart_tx va al pin D4.

En este punto hemos finalizado el diseño HW de sistema así que vamos a proceder con los pasos necesarios para exportarlo y poder programarlo.

5. Exportar el diseño HW

Para exportar el diseño HW lo primero que vamos a hacer es validar que el diseño es correcto. Para ello hacemos click en la barra superior del diseño de bloques en el botón "Validate Design", Figura 22.

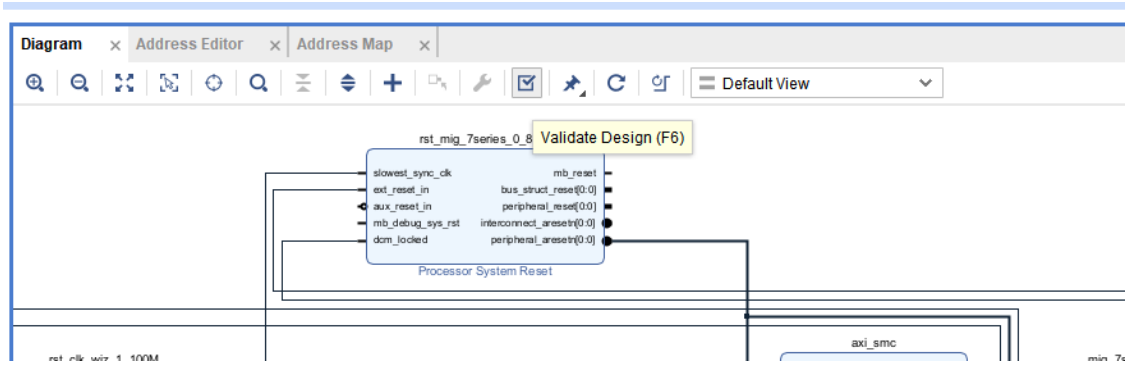

Figura 22. Validar diseño.

Después vamos a generar el *HDL Wrapper* que consiste en un fichero VHDL que permite al sintetizador sintetizar nuestro diseño. Si deseásemos utilizar algún diseño digital propio también nos permitiría conectarlo como si de un componente de VHDL más se tratase. Para generarlo es necesario en la pestaña de sources hacemos click derecho sobre el diseño de bloques y en el desplegable seleccionamos *Create HDL Wrapper*, Figura 23.

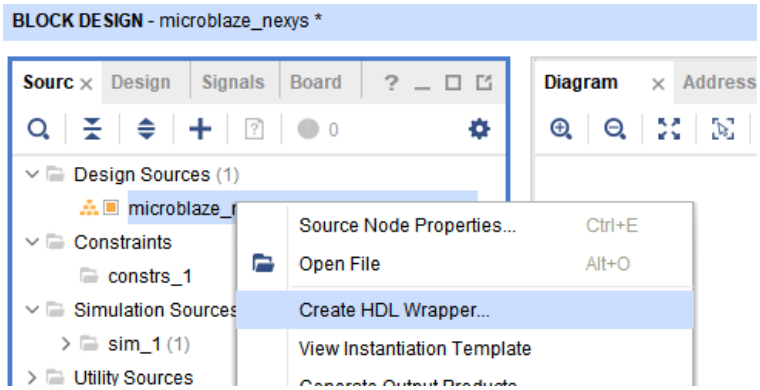

Figura 23. Create HDL Wrapper de nuestro diseño de bloques.

A continuación vamos a generar el bitstream para programar en la FPGA, para ello hacemos click en el menú izquierdo en *Generate Bitestream*. Este proceso lleva bastante tiempo ya que está sintetizando y rutando todas las IPs que hemos incluido.

Una vez finalizado el proceso, se abrirá una ventana con diferentes opciones para visualizar nuestro diseño final, podemos hacer click en cancelar. Ya tenemos el bitstream que vamos a cargar en la FPGA.

Por último para poder desarrollar SW para el microblaze que hemos integrado en nuestro sistema necesitamos exportar el diseño HW. Hacemos click en File->Export->Export Hardware, Figura 24.

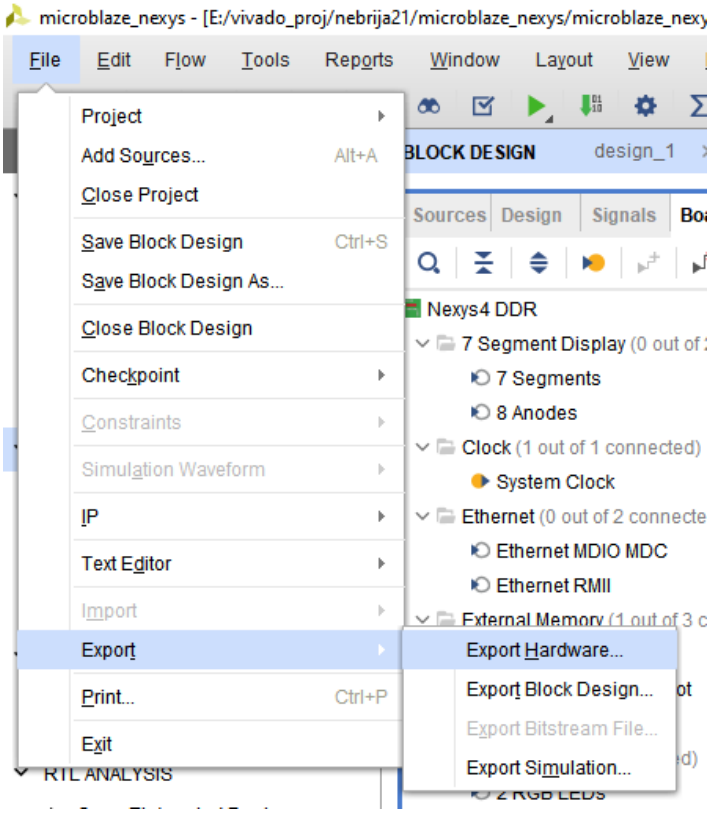

Figura 24. Exportar el diseño de HW.

A continuación vamos a seleccionar en la ventana emergente seleccionamos el include bitstream, tal y como se muestra en Figura 25.

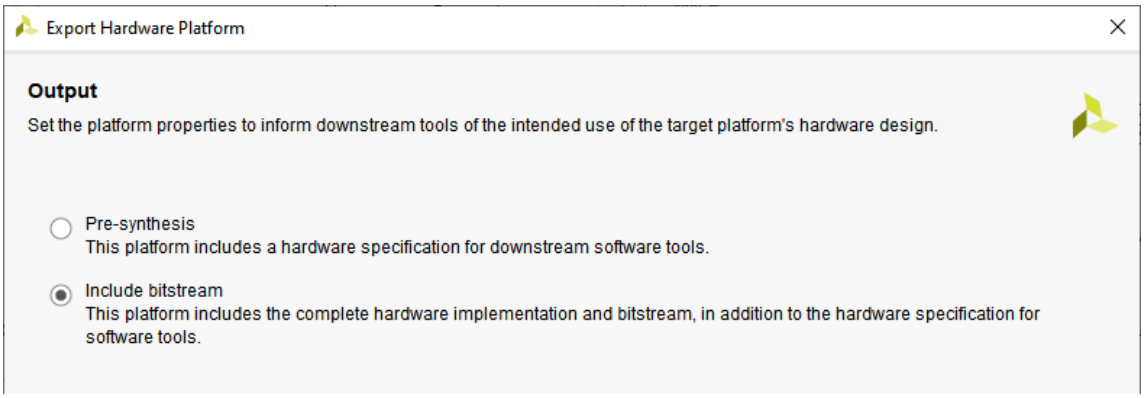

Figura 25. Seleccionar bitstream en el diálogo.

6. Diseño de la aplicación SW

A partir de ahora vamos a trabajar en Vitis, podemos lanzar el IDE desde el propio Vivado en Tools->Launch Vitis IDE. Debemos crear un proyecto de aplicación (File-> New -> Platform

Project).

Se debe elegir la plataforma, en este punto debemos cargar el fichero *.xsa que hemos exportado con anterioridad. Se debe elegir como sistema operativo *standoalone* y procesador microblaze_0.

Una vez tengamos la plataforma creada podemos crear un proyecto de aplicación (File-> New -> Application Project), seleccionamos el proyecto de plataforma que hemos generado con anterioridad, le damos un nombre a la aplicación y entre la lista de plantillas seleccionamos Hello World.

La aplicación Hello World tiene dentro de /src el fichero helloworld.c donde se encuentra la función main de la aplicación. Podemos comprobar que contiene 5 líneas de código para escribir por el serial.

```
int main()
{
    init_platform();
    print("Hello World\n\r");
    print("Successfully ran Hello World application");
    cleanup_platform();
    return 0;
}
```

A continuación, vamos a compilar el código haciendo click Project -> Build All. Si todo es correcto continuamos con la práctica.

7. Cargar HW/SW en la FPGA

Para cargar el bitstream tenemos que ir a Vivado, y hacer click en la pestaña Program and Debug -> Open Hardware Monitor. Ahí deberemos hacer click en open target y después en Program device. Con esto la FPGA tiene el diseño de Hardware que diseñamos con anterioridad. Ahora debemos ir a Vitis y hacer click en Window-> Console para visualizar la consola.

Una vez tenemos la consola en la parte inferior derecha hacemos click en Open Console View-> 4. Command Shell Console, Figura 26.

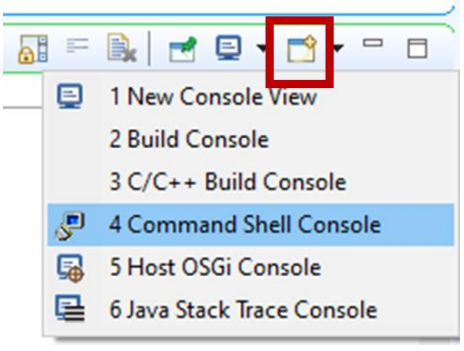

Figura 26. Crear nueva consola serial.

Debemos en la ventana que se abre seleccionar conexión serie, le damos el nombre que deseos y la codificación por defecto, véase Figura 27.

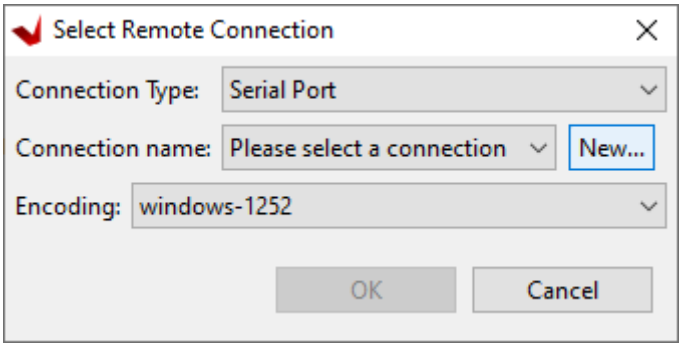

Figura 27. Crear terminal serial.

Al hacer click en new debemos configurar la conexión serial seleccionando el puerto COM4 y el baudrate de 9600, Figura 28.

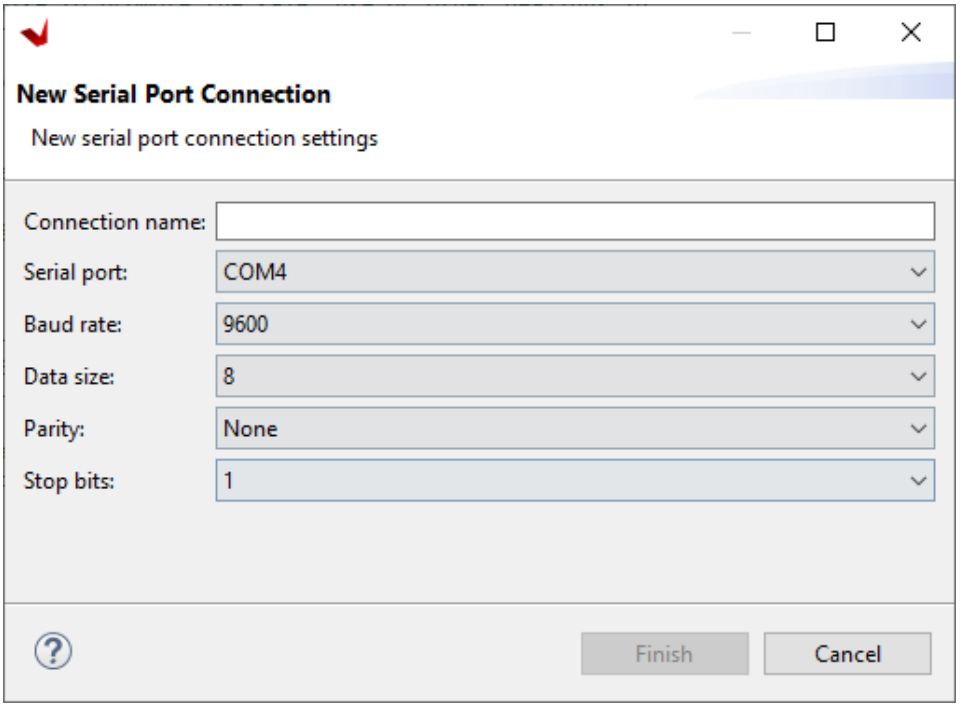

Figura 28. Parámetros de la conexión serie.

Si la placa está funcionando correctamente la consola debería aparecer como “(CONNECTED)”. En este punto vamos a cargar al micro el programa de hola mundo que hemos compilado. Para ello hacemos click derecho en nombreApp_system Run As-> Launch Hardware, tal y como se muestra en la Figura 29.

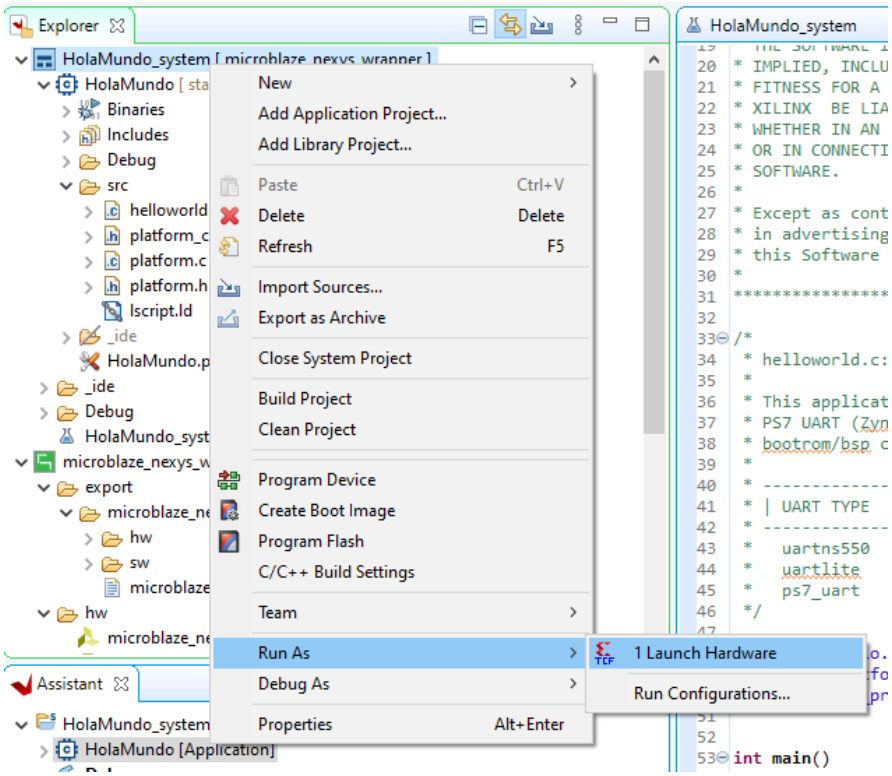

Figura 29. Escribir el código a través del serial.

Se abrirá una ventana con una barra de carga y una vez finalice la carga podremos ver en la consola serial lo siguiente, Figura 30.

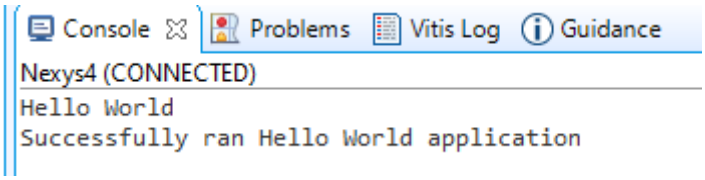

Figura 30. Hola mundo.

8. Evaluación y entrega

8.1. Grupos

Se utilizarán los mismos grupos que en la anterior práctica.

8.2. Puntuación

Este hito se corresponde con un **20%** del valor del proyecto final. Para su evaluación basta con enseñar el código funcionando. El código será revisado por el profesor en la siguiente sesión de prácticas. Para la revisión es necesario llevar el proyecto funcionando y con una visualización de las señales de interés clara que permita ver el funcionamiento del sistema.