MÁSTER OFICIAL EN INGENIERÍA ELECTRÓNICA

SISTEMAS INTEGRADOS

BLOQUE 1: Introducción al Diseño de Sistemas Integrados

Curso 2013-2014

José Torres

Raimundo García

Julio Martos

Jesús Soret

Adrián Suárez

Pedro A. Martínez

Abraham Menéndez

DSDC Diseño de Sistemas Digitales y de Comunicaciones

VNIVERSITAT DE VALÈNCIA Escola Tècnica Superior d'Enginyeria

SISTEMAS INTEGRADOS

Tema 1.- Introducción al Diseño de Sistemas Integrados

Máster Oficial en Ingeniería Electrónica

Curso 2012-13

VNIVERSITAT DE VALÈNCIA Escola Tècnica Superior d'Enginyeria

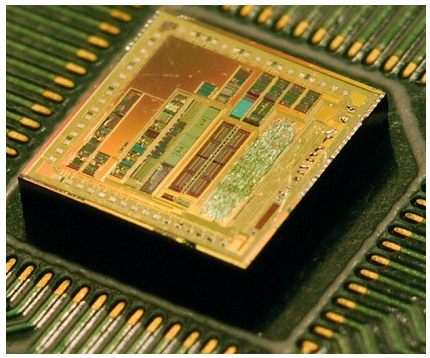
Introducción al Diseño de Sistemas Integrados

Índice

- Introducción.
- Sistemas Integrados en FPGAs.
- Codiseño Hardware/Software.

Introducción

- Sistemas Digitales Avanzados
  - Son aquellos en los que el núcleo del sistema es uno de los dispositivos de computación usual.
- Dispositivos Digitales Usuales
  - ASICs
  - FPGAs
  - Microcontroladores
  - Microprocesadores
  - DSPs
  - CPLDs



Introducción

- Método Software:
  - Se usan circuitos procesadores estándar (Microprocesadores, Microcontroladores o DSPs)
  - Se usan lenguajes de programación asociados a un juego de instrucciones fijo.
  - Funcionamiento de manera secuencial.

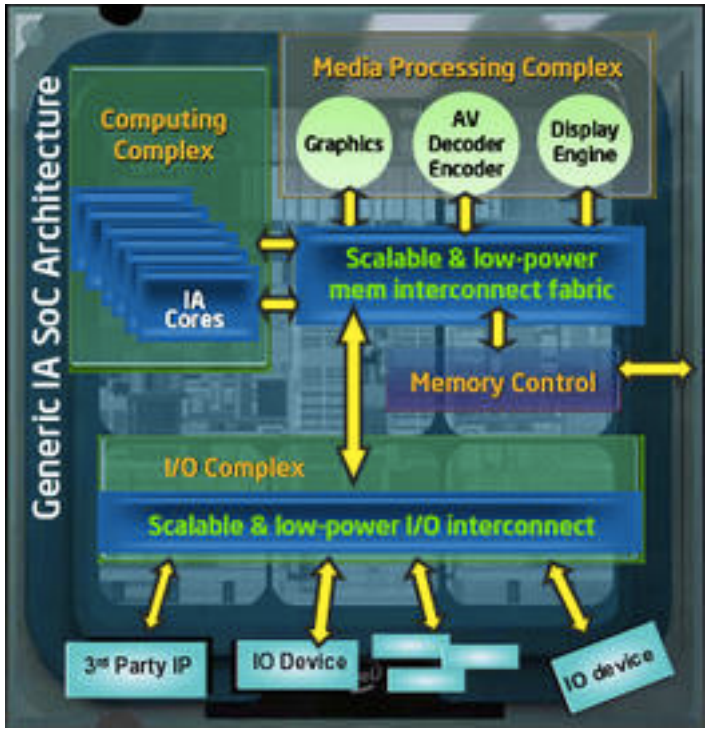
- Método Hardware:
  - Se usan circuitos hardware definibles por el usuario (CPLDs, FPGAs y ASICs)
  - Se usan los lenguajes de descripción hardware.
  - Funcionamiento de manera concurrente.

Introducción

- El método software se ve limitado por:
  - El procesado secuencial de las instrucciones.
  - La arquitectura concreta del procesador utilizado.
- El método hardware se ve limitado por:
  - La dificultad de diseñar mediante lenguajes de programación hardware.
  - El tiempo de realización de sistemas hardware específicos.
  - El coste de dichos sistemas.
- Actualmente existe una opción intermedia entre ambos métodos.

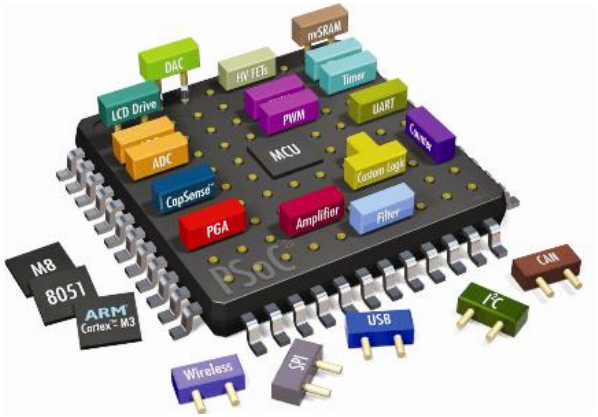
Introducción

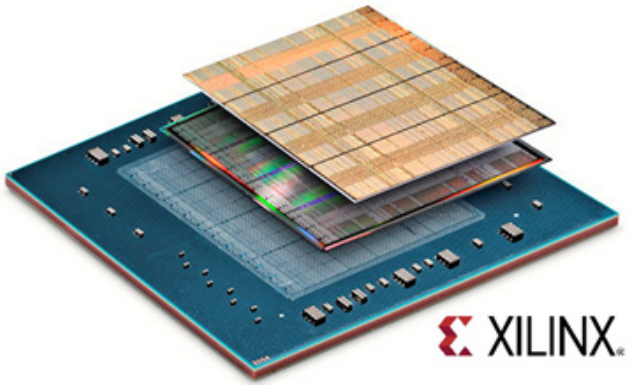
- Esta opción intermedia es la más usada en la actualidad y se denomina Sistema Integrado, Sistema Embebido o System on Chip.
- Un Sistema Integrado incluye en un único circuito los siguientes elementos:
  - Una o varias unidades de computación (Microprocesadores/DSPs/FPGAs…)
  - Una o varias unidades de memoria (RAM/FLASH…)
  - Diferentes circuitos de interfaz estándar (UART/SPI/I2C/Ethernet...)
  - Diferentes circuitos específicos para la aplicación (Gráficos/Automoción…)



Introducción

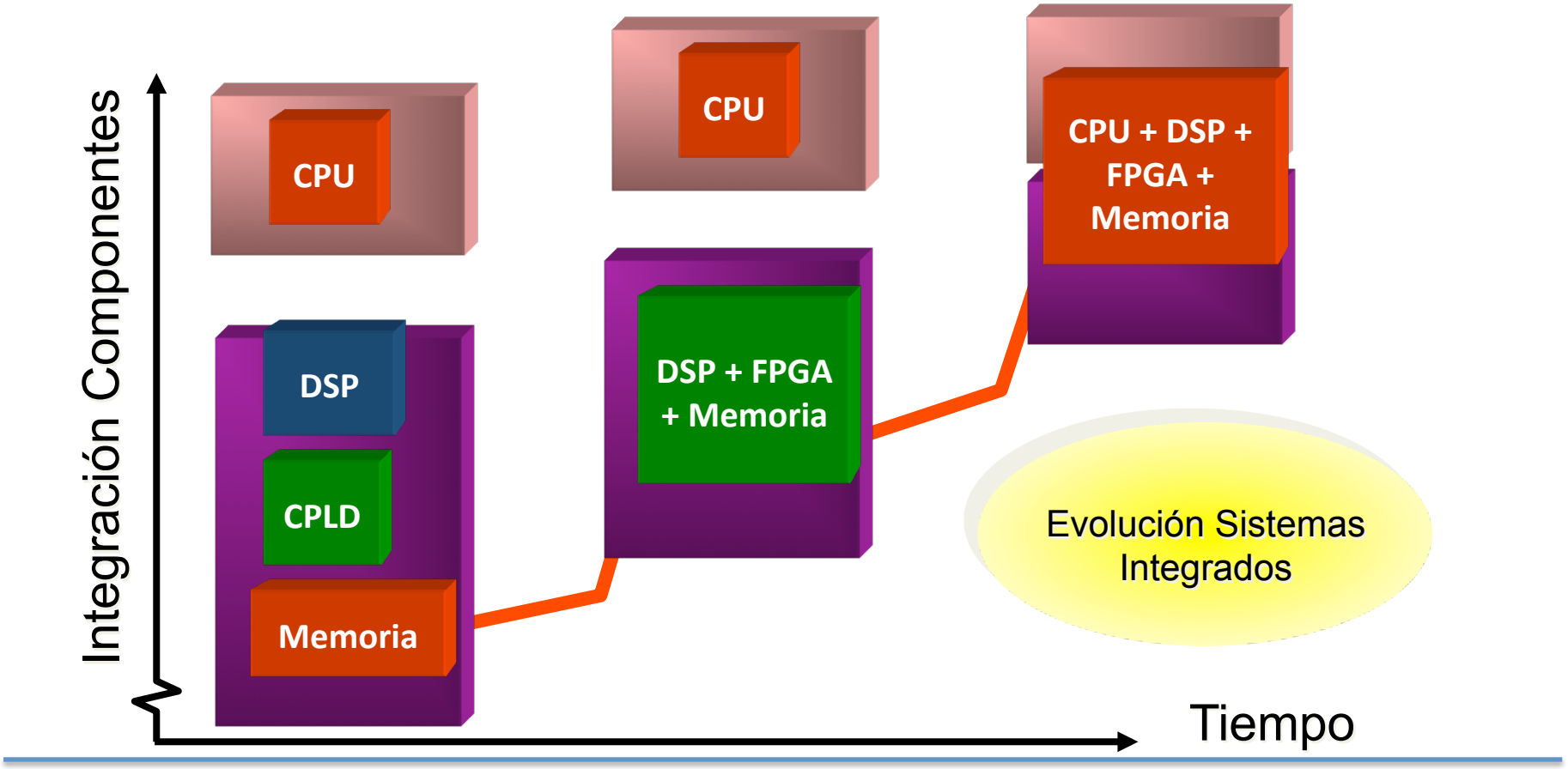
- Las opciones que tenemos actualmente para trabajar con un Sistema Integrado son:
  - Aquellos que ya se comercializan y que integran un microcontrolador, una parte digital programable, una parte analógica también programable y puertos de comunicación.
    - FIPSOC (Sidsa), Fusion (Actel), FPSC (Lattice) y PSoC (Cypress)
  - Aquellos que mediante una FPGA de gran capacidad lógica, permiten integrar un microprocesador y puertos de comunicación a medida.
    - Xilinx y Altera.





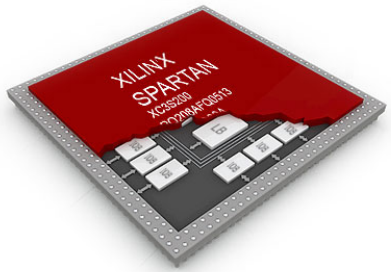
Introducción

- El aumento de la capacidad de integración en la fabricación de chips ha permitido el desarrollo de los Sistemas Empotrados.



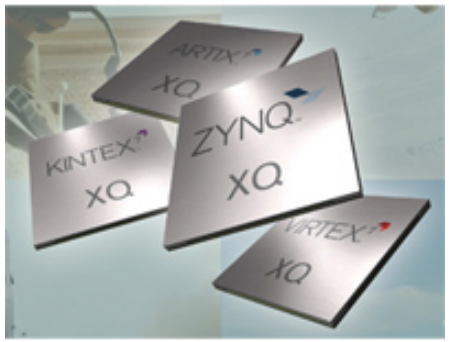
Introducción

- En el diseño de un Sistema Empotrado deben tenerse en cuenta los siguientes factores:
  - El "hardware" es diferente para cada aplicación.
  - Es posible necesitar un sistema operativo (RTOS)
  - Es conveniente utilizar un código compacto para reducir el tamaño de la memoria de programa.
  - Es necesaria la combinación de lenguajes de alto nivel (C) con lenguajes de bajo nivel (VHDL) para optimizar la velocidad de proceso.
- En esta asignatura nos vamos a centrar en los Sistemas Integrados basados en FPGAs, también llamados Systems on a Programmable Chip (SoPCs)



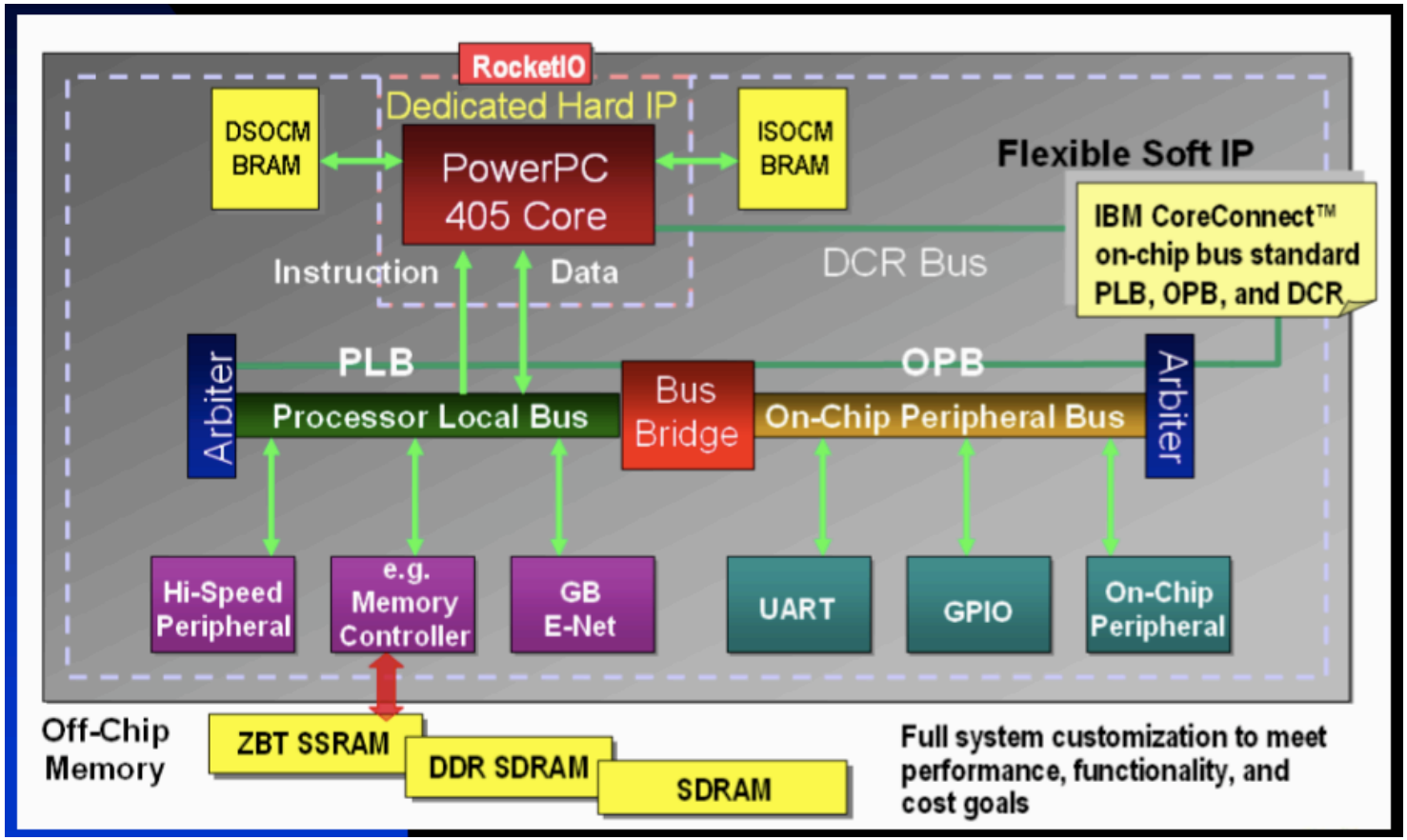
Sistemas Integrados FPGAs

- Xilinx, uno de los principales fabricantes de FPGAs, dispone de dos opciones para realizar un SoPC.
  - Bloque microprocesador hardware.
    - Power-PC de 32 bits en FPGAs Virtex 2 pro, Virtex 4 FX y Virtex 5 FXT.
    - ARM Dual-Core Cortex-A9 de 32 bits en Zynq.
  - Bloque microprocesador software.
    - Picoblaze de 8 bits en cualquier FPGA.
    - Microblaze de 32 bits para las familias Spartan, Virtex y las nuevas Artix y Kintex.



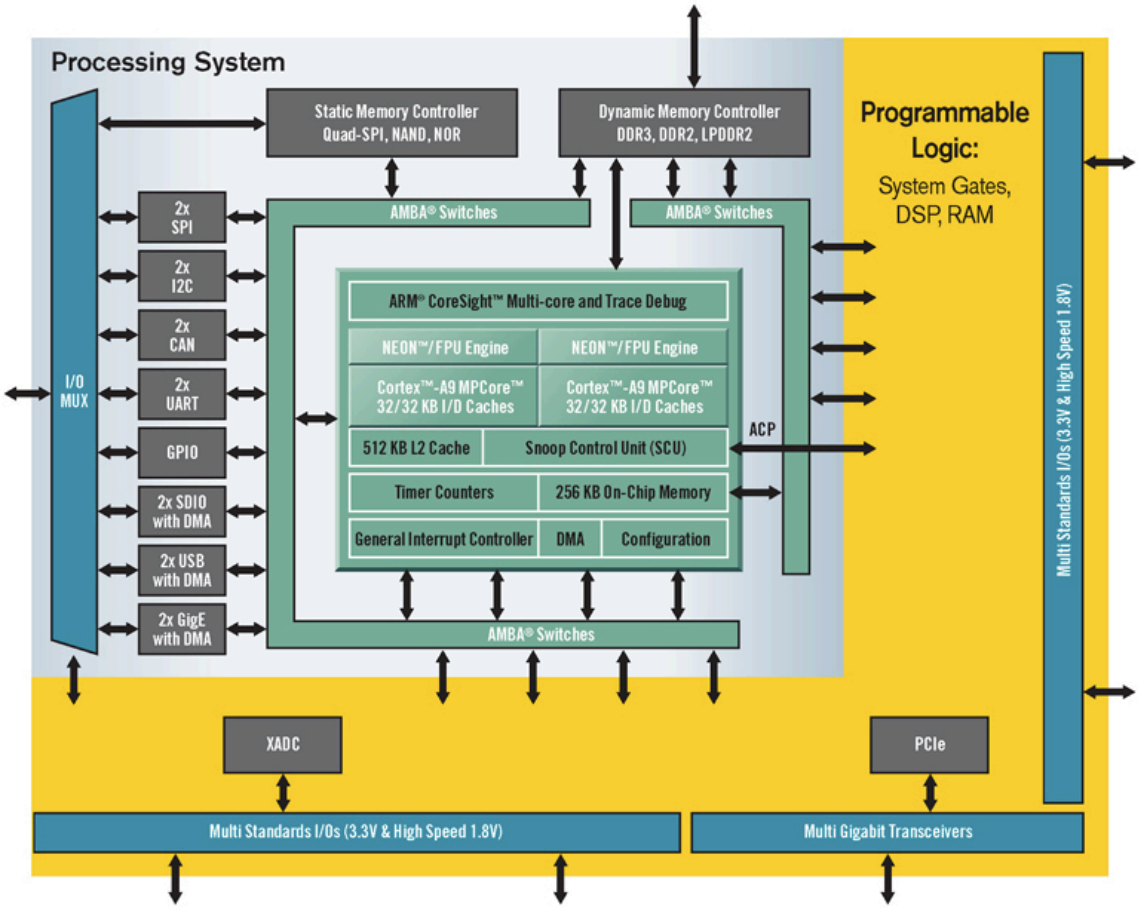
Sistemas Integrados FPGAs

- Arquitectura de un Sistema Empotrado “hardware” basado en Power-PC.



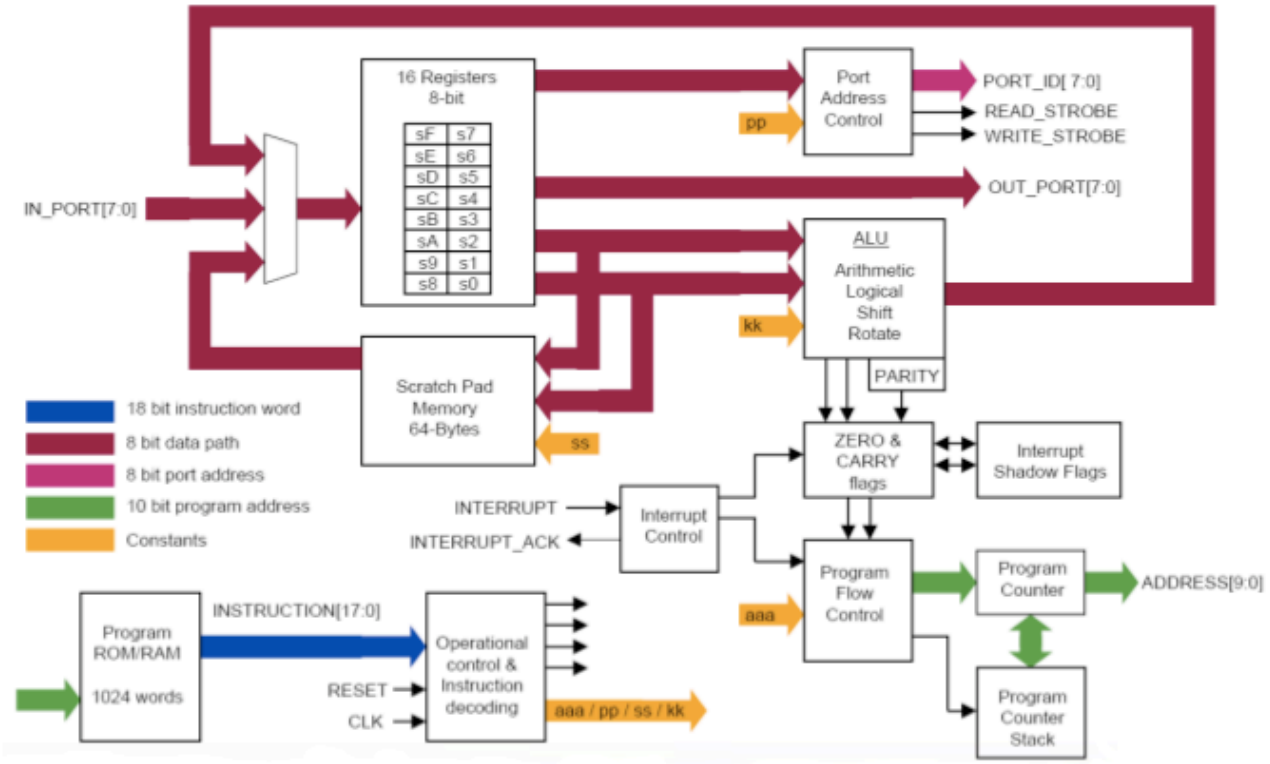
Sistemas Integrados FPGAs

- Arquitectura de un Sistema Empotrado “hardware” basado en ARM.



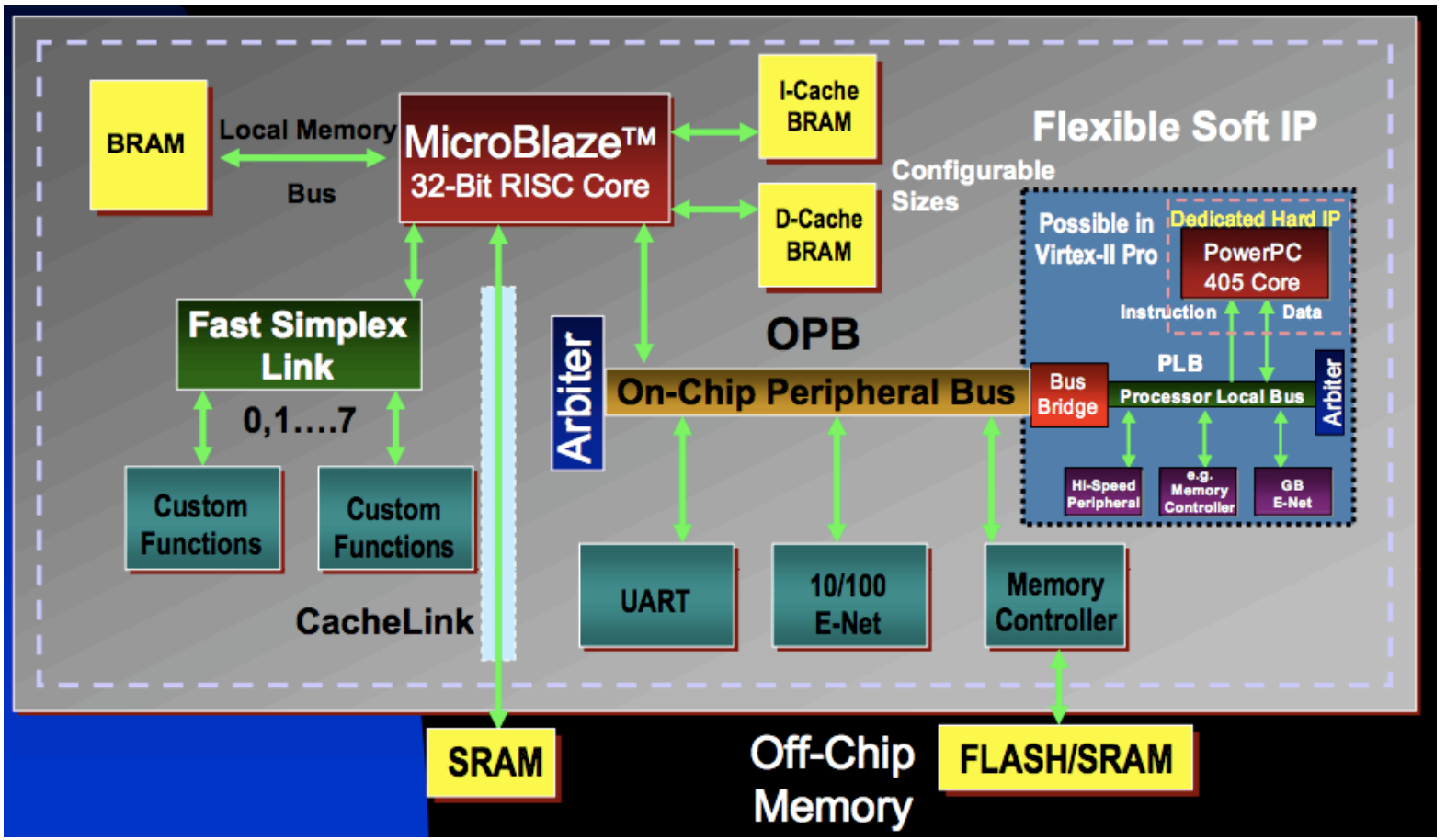
Sistemas Integrados FPGAs

- Arquitectura de un Sistema Empotrado “software” basado en PicoBlaze.



Sistemas Integrados FPGAs

- Arquitectura de un Sistema Empotrado "software" basado en MicroBlaze.



Sistemas Integrados FPGAs

- Las posibles opciones a la hora de escoger un bloque microprocesador software son:
  - Núcleo del microprocesador comercial diseñado para su implementación en FPGAs.
    - Ventajas:
      - Poder disponer de un microprocesador comercial ampliamente utilizado (8051, PIC,…)
      - Reducción del tiempo de diseño.
    - Inconvenientes:
      - Coste elevado.
      - Arquitectura del microprocesador fija.

Sistemas Integrados FPGAs

- Las posibles opciones a la hora de escoger un bloque microprocesador software son:
  - Núcleo del microprocesador diseñado a medida por el usuario.
    - Ventajas:
      - Poder disponer de una arquitectura totalmente a la medida de las aplicaciones.
      - Disponer del código fuente del microprocesador para realizar modificaciones.
      - Coste reducido.
      - Posibilidad de comercialización.
    - Inconvenientes:
      - Tiempo de diseño elevado.

Sistemas Integrados FPGAs

- Las posibles opciones a la hora de escoger un bloque microprocesador software son:
  - Núcleo del microprocesador prediseñado por el fabricante de FPGAs.
    - Ventajas:
      - Poder disponer de un microprocesador utilizado por muchos usuarios.
      - Intercambio de programas y periféricos de libre uso.
      - Reducción del tiempo de diseño.
      - Coste reducido.
    - Inconvenientes:
      - Limitaciones en la arquitectura.
- En esta asignatura trabajaremos con esta solución, concretamente con MicroBlaze.

Codiseño Hardware/Software

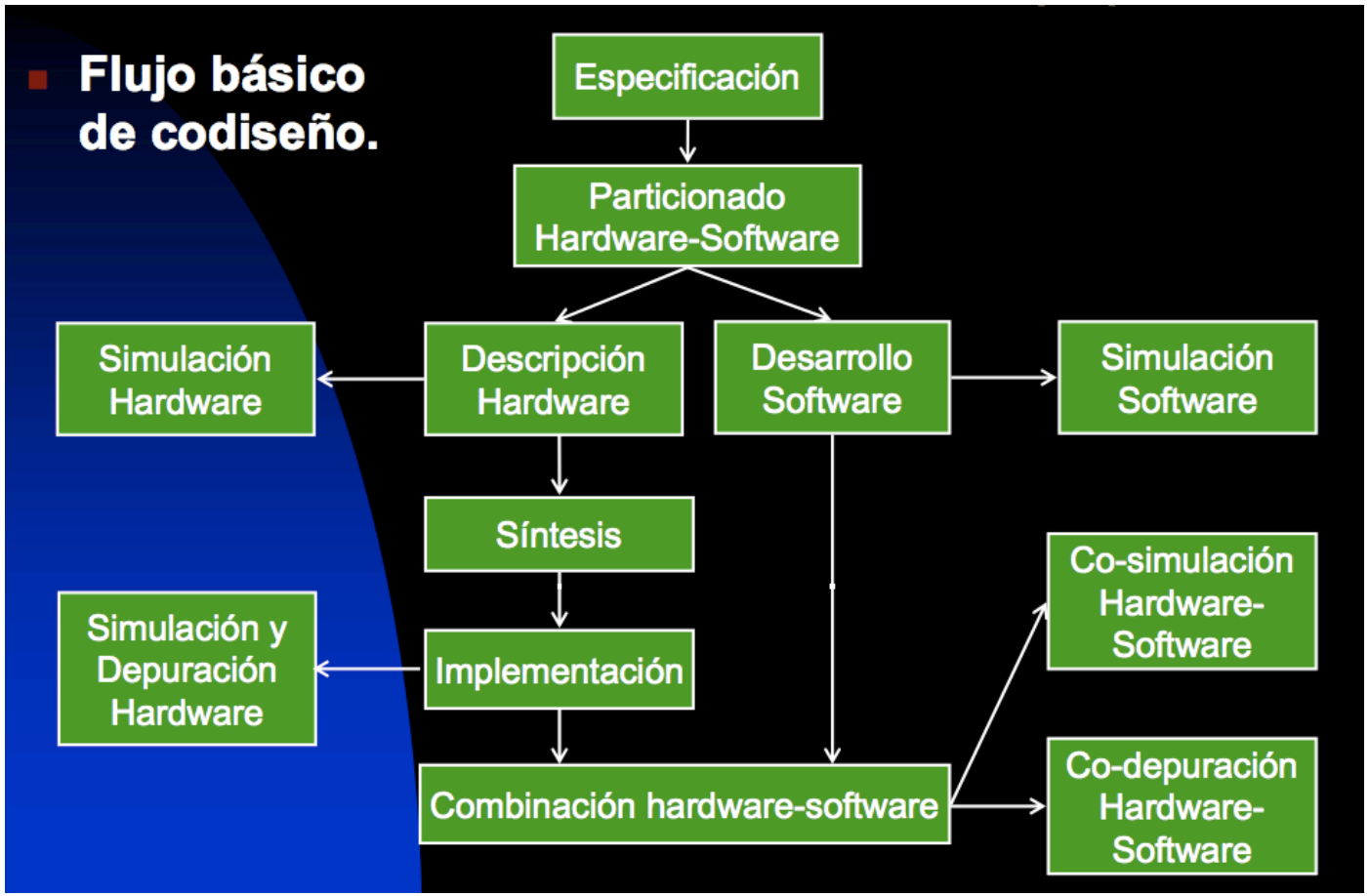
- Una vez entendido qué es un Sistema Integrado, debemos pararnos en comprender cómo se realiza el diseño de los mismos.
- Estos diseños presentan una complejidad elevada, es necesario trabajar en:
  - El hardware necesario, tanto el del propio microprocesador, como el de sus periféricos.
  - El programa que va a ejecutar el microprocesador.
  - El resto de componentes adicionales que se precisen para la aplicación.
- Este proceso se suele denominar Codiseño Hardware/Software y se puede definir dicho concepto:
  - El diseño concurrente por parte del mismo equipo de diseñadores de los componentes hardware y software de un Sistema Empotrado. Usando para ello metodologías y herramientas que consideren la interacción hardware/ software.

Codiseño Hardware/Software

- Los principales parámetros a considerar cuando realizamos Codiseño son:
  - Velocidad del diseño.
  - Área de uso.
  - Coste del desarrollo.
- Las soluciones hardware ofrecen buenas prestaciones en velocidad, aunque el coste y el área pueden ser elevados.
- Las soluciones software son más flexibles y de menor coste aunque más lentas.
- Las técnicas de Codiseño buscan el equilibrio entre ambas soluciones.

Codiseño Hardware/Software

- Flujo básico de Codiseño.



Codiseño Hardware/Software

- Flujo básico de Codiseño.
  - Especificación:
    - Definir la funcionalidad del sistema a desarrollar.
  - Particionado hardware/software:
    - Decidir que funciones se realizarán mediante rutinas software en el microprocesador empotrado y cuáles mediante circuitos hardware específicos.
    - Existen herramientas comerciales de coste elevado específicas para esta función.
  - Descripción hardware:
    - Elección del microprocesador y de la FPGA adecuados para la aplicación.
    - Diseño de periféricos mediante lenguajes HDL.
  - Desarrollo software:
    - Desarrollo de la aplicación en Lenguaje Ensamblador/C.
    - Compilación.

Codiseño Hardware/Software

- Flujo básico de Codiseño.
  - Simulación hardware:
    - Mediante programas simuladores a la herramienta usada.
  - Simulación software:
    - Mediante programas simuladores del lenguaje de programación usado.
  - Cosimulación hardware/software:
    - Herramientas tales como EDK, ChipScope.
  - Síntesis e implementación:
    - Se utiliza la herramienta específica del fabricante de la FPGA (ISE)

Codiseño Hardware/Software

- Ejemplo de Codiseño.
  - Control de un LCD de texto de 2 filas de 16 caracteres.

Power-On Initialization

The initialization sequence first establishes that the FPGA application wishes to use the four-bit data interface to the LCD as follows:

- Wait 15 ms or longer, although the display is generally ready when the FPGA finishes configuration. The 15 ms interval is 750,000 clock cycles at 50 MHz.
- Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.
- Wait 4.1 ms or longer, which is 205,000 clock cycles at 50 MHz.
- Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.
- Wait 100 µs or longer, which is 5,000 clock cycles at 50 MHz.
- Write SF_D<11:8> = 0x3, pulse LCD_E High for 12 clock cycles.
- Wait 40 µs or longer, which is 2,000 clock cycles at 50 MHz.
- Write SF_D<11:8> = 0x2, pulse LCD_E High for 12 clock cycles.
- Wait 40 µs or longer, which is 2,000 clock cycles at 50 MHz.

Codiseño Hardware/Software

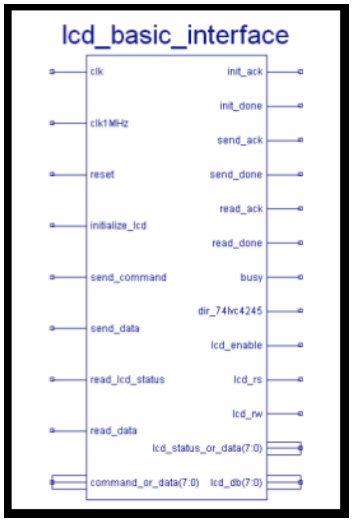
- Ejemplo de Codiseño.
  - Particionado Hardware/Software:
    - La lentitud de funcionamiento del LCD unida a la complejidad de la secuencia de operaciones a ejecutar para controlarlo lo hacen muy adecuado para ser manejado mediante un microprocesador.
    - Por otra parte, dicha lentitud obliga a un microprocesador a atender el control del visualizador durante largos periodos de tiempo, por lo que también sería adecuado descargar al microprocesador de esa tarea.
  - Primera opción (Software):
    - Realizar el control del visualizador totalmente por software, añadiendo el mínimo hardware externo necesario para la interfaz.
    - El software del microprocesador deberá encargarse de enviar y recibir instrucciones y datos al LCD (rutina bajo nivel) y del envío de mensajes (rutina alto nivel)
    - Este sería el caso de utilizar un microprocesador específico para toda la aplicación.

Codiseño Hardware/Software

- Ejemplo de Codiseño.
  - Segunda opción (Hardware):
    - Realizar el control del visualizador totalmente por hardware específico, añadiendo el mínimo software necesario para el LCD.
    - En este caso, el hardware controlará completamente el dispositivo.
    - Este sería el caso de utilizar una FPGA para el control del visualizador, lo cual implicaría bastante más coste y un área importante.
  - Tercera opción (Codiseño):
    - Realizar el control del visualizador mediante un hardware versátil que a la vez nos permita realizar otras funciones cuando no este en uso el LCD.
    - Podemos implementar mediante hardware (VHDL) la ejecución de las funciones de bajo nivel, como son el envío o recepción de una instrucción o dato al LCD.
    - Mediante “software” (implementar un microprocesador en el hardware, C) realizaríamos las rutinas de alto nivel, como el envío de mensajes de varios caracteres.

Codiseño Hardware/Software

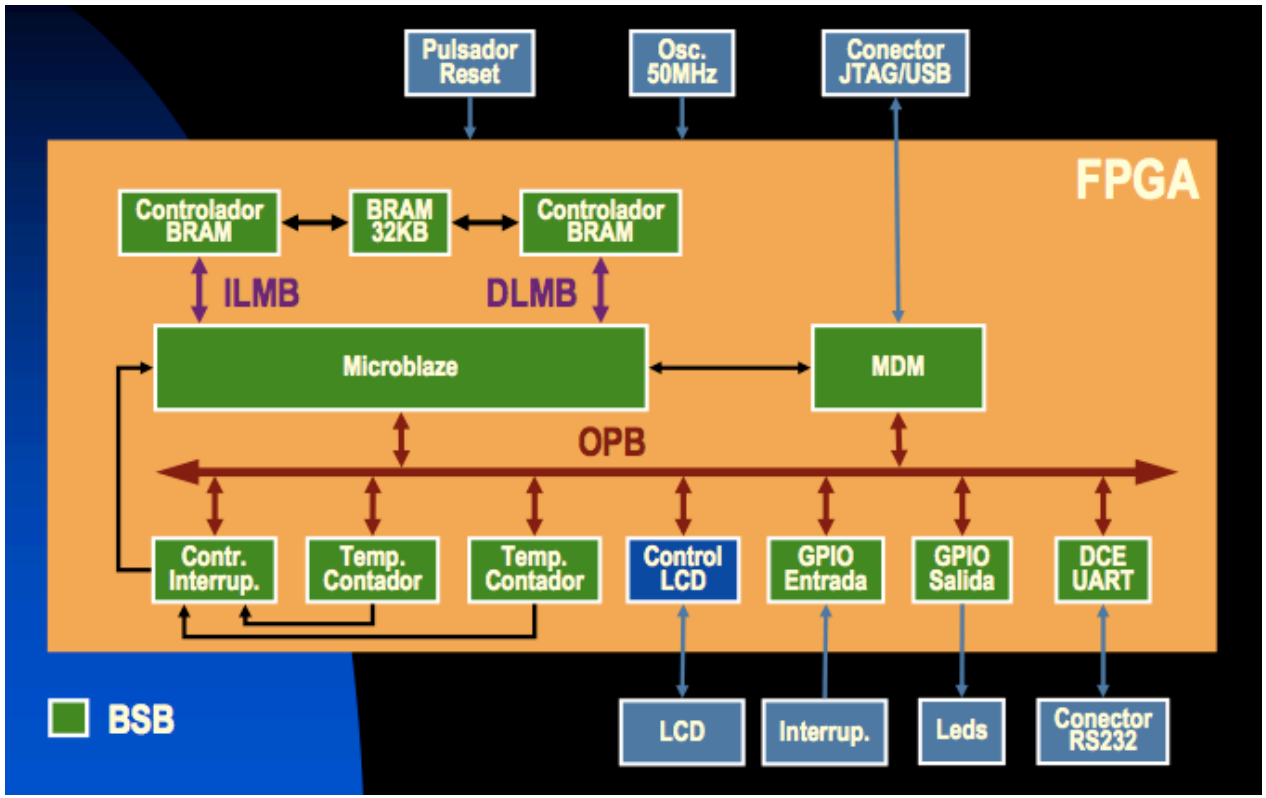
- Ejemplo de Codiseño.
  - Tercera opción (Codiseño):
    - Las tareas más críticas, sobre todo en tiempo, se implementan en hardware específico mientras que el resto se programan mediante C para ser ejecutadas por el microprocesador.
    - El hardware actúa como un coprocesador.
    - Es la mejor estrategia cuando se trabaja con diseños digitales avanzados.




```
void XromLCDPrint2Strings(char * line1, char * line2){

      int i=0;

            XromLCDSetLine(1);

            for(i=0; i<16; i++){
                  if(line1[i])
                        XromLCDPrintChar(line1[i]);
                  else break;
            }

            XromLCDSetLine(2);

            for(i=0; i<16; i++){
                  if(line2[i])
                        XromLCDPrintChar(line2[i]);
                  else break;
            }
      return;
}
```

Codiseño Hardware/Software

- Ejemplo de Codiseño.
  - Tercera opción (Codiseño):



Cuestiones

1

Ejercicio básico con XPS

El primer diseño con Microblaze se va a realizar sobre un Starter Kit Board que nos ofrecen Xilinx y Avnet y que contiene entre otros componentes una FPGA de la familia Spartan-6.

La finalidad de estas cuestiones es comprobar que se han asimilado los conocimientos que se pretenden transmitir a través de la realización de esta práctica. Deberías ser capaz de responder a todas las cuestiones que aparecen a continuación. De no ser así, se recomienda que revises el guión de la práctica y que repitas aquellas partes que no han quedado del todo claras. No dudes en consultar cualquier duda a tu profesor en caso de ser necesario.

1.- ¿Qué es el XPS? ¿Qué es EDK?

2.- ¿Qué es el BSB? ¿Es necesario utilizarlo siempre?

3.- ¿Para qué sirve el IP catalog? ¿Podríamos haberlo utilizado para hacer lo mismo que hace el BSB? ¿Podemos utilizar cualquier periférico de los que aparecen en la lista?

4.- ¿Para que sirve la opción "Generate bitstream"? ¿Se pueden ver los recursos de la FPGA ocupados antes de utilizar esta función?

5.- En lo referente al software, ¿por qué lanzamos SDK? ¿Cómo se generan las bibliotecas y los programas de test? ¿Por qué hay para un firmware empotrado 3 archivos en SDK? ¿Qué contiene cada uno de ellos?

6.- ¿Quién ha generado los programas de test? ¿En qué lenguaje están escritos? ¿Qué opción se utiliza para compilarlos? ¿Dónde se almacenan en la FPGA? ¿Se almacenan automáticamente? ¿Qué diferencia hay entre la "block RAM (BRAM)" y la "distributed RAM"?

7.- ¿Qué contienen los ficheros con extensiones .mhs y .mss? ¿Cuál es la función del fichero con extensión .ucf?

8.- ¿Qué hace la opción "Program FPGA"? ¿Hay que utilizarla cuando modifiquemos el hardware o el software? ¿Podemos programar sólo el hardware con esta opción o también el software?

9.- Si reiniciamos la FPGA, ¿estará esta programada con nuestro último diseño?

Ejercicio 1

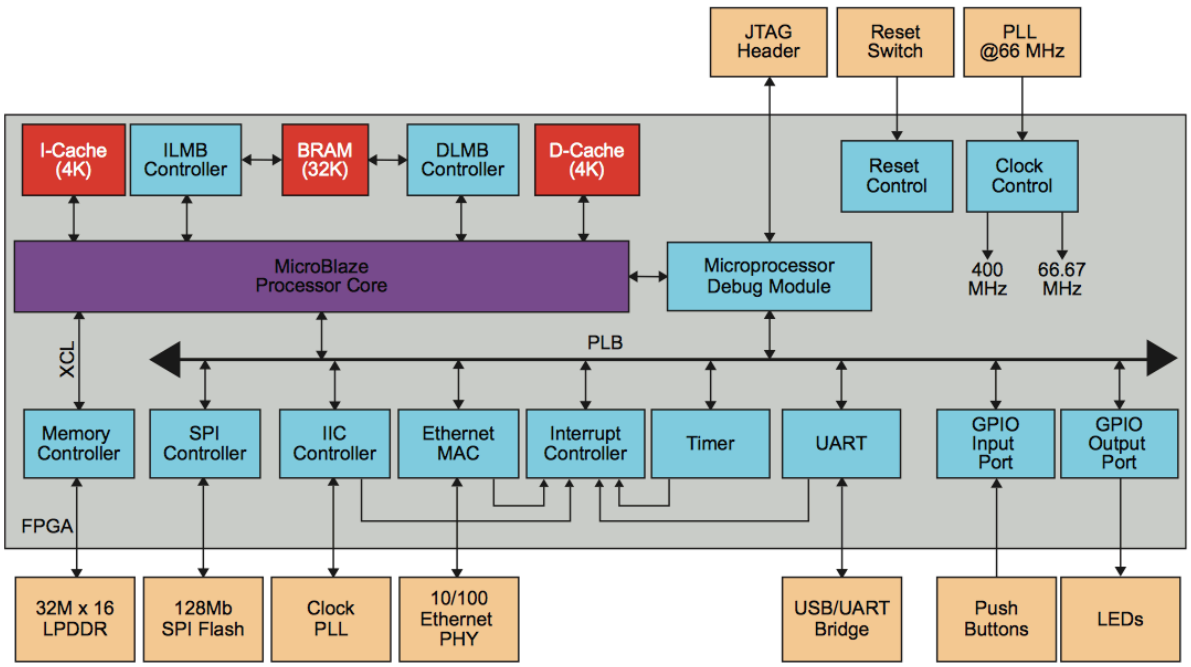
Ejercicio básico con XPS

El primer diseño con Microblaze se va a realizar sobre un Starter Kit Board que nos ofrecen Xilinx y Avnet y que contiene entre otros componentes una FPGA de la familia Spartan-6.

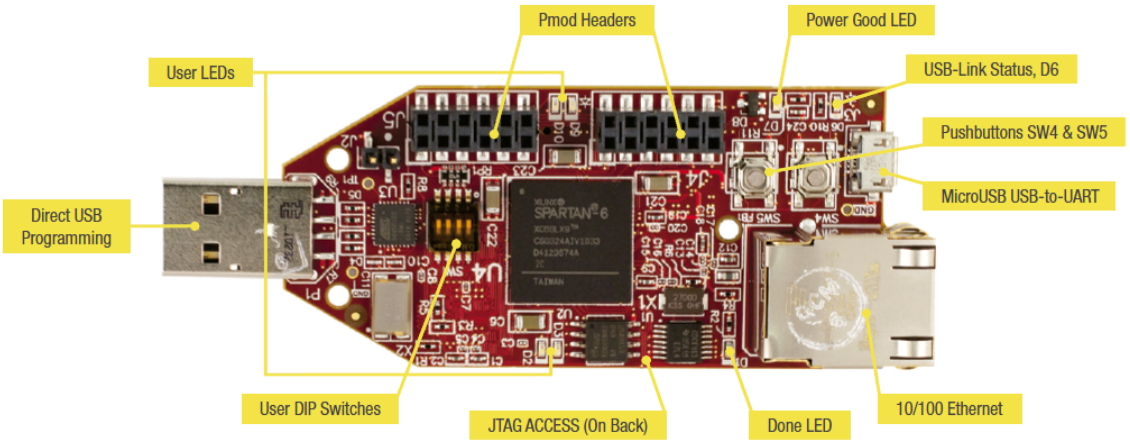
La finalidad de este ejercicio es realizar un diseño básico tipo "hardware" mediante un microprocesador embebido de 32 bits (Microblaze) y usando la herramienta XPS (Xilinx Platform Studio). Para ello, usaremos y conoceremos el BSB (Base System Builder), un asistente que nos ofrece la herramienta para realizar de forma rápida una aplicación.

Probaremos el correcto funcionamiento del "hardware" generando las aplicaciones "software" de test también generadas con BSB. Asimismo, exploraremos y utilizaremos el entorno XPS para programar la tarjeta y conoceremos la estructura de directorios y archivos fundamentales en cualquier diseño con Microblaze.

En este ejemplo crearemos una conexión entre la FPGA y periféricos exteriores empotrando Microblaze y usando el puerto RS-232. El diagrama de bloques que ilustra este ejemplo es el siguiente.



La tarjeta que se va a emplear para realizar los distintos ejercicios a lo largo de la asignatura está basada en una FPGA Spartan – 6, es la LX9 MicroBoard, y posee los siguientes módulos:



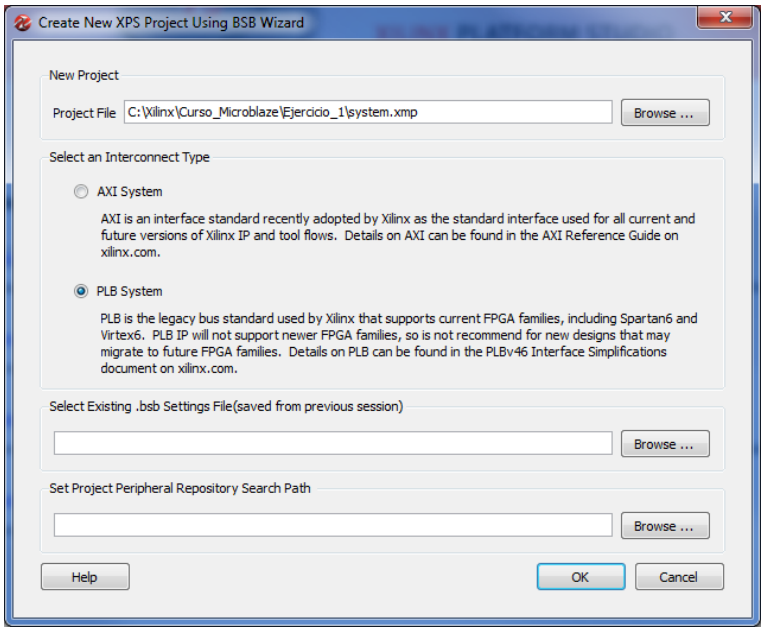
Los pasos a realizar en este primer ejercicio son los siguientes:

- Crear un Proyecto usando el Base System Builder de EDK
- Analizar el Proyecto creado e identificar tanto el microprocesador embebido como los periféricos
- Generar el Bitstream de la parte hardware
- Crear las aplicaciones para realizar un test de la memoria y los periféricos usando SDK
- Conectar la placa al ordenador y comprobar todo lo realizado sobre la misma

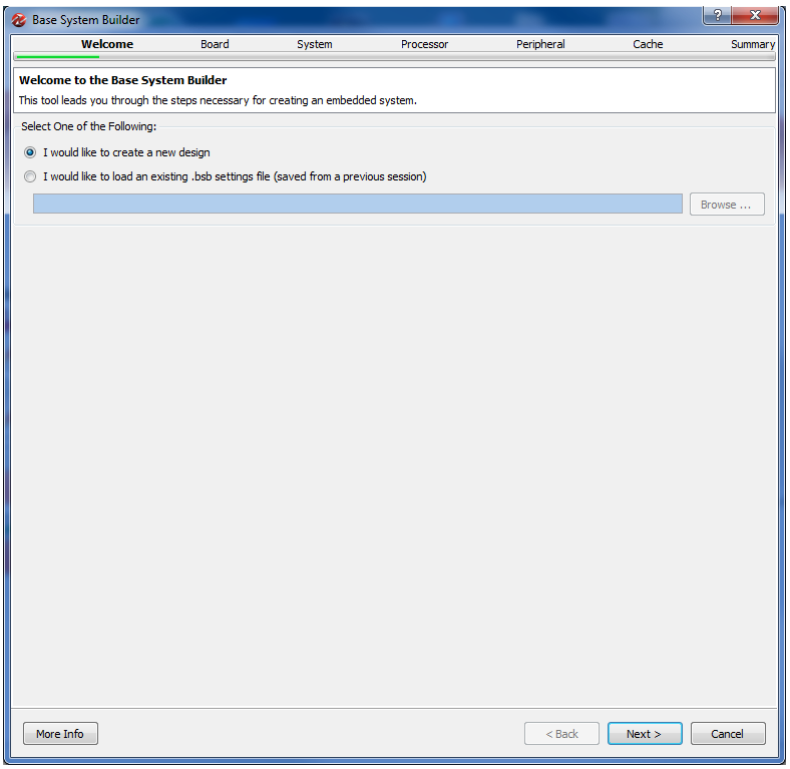
Creación sistema empotrado

Lo primero es iniciar XPS, buscando la versión instalada. Lo encontraréis en Xilinx Design Tools -> ISE Design Suite 14.4 -> EDK. Una vez iniciado elegir la opción Create New Project Using Base System Builder.

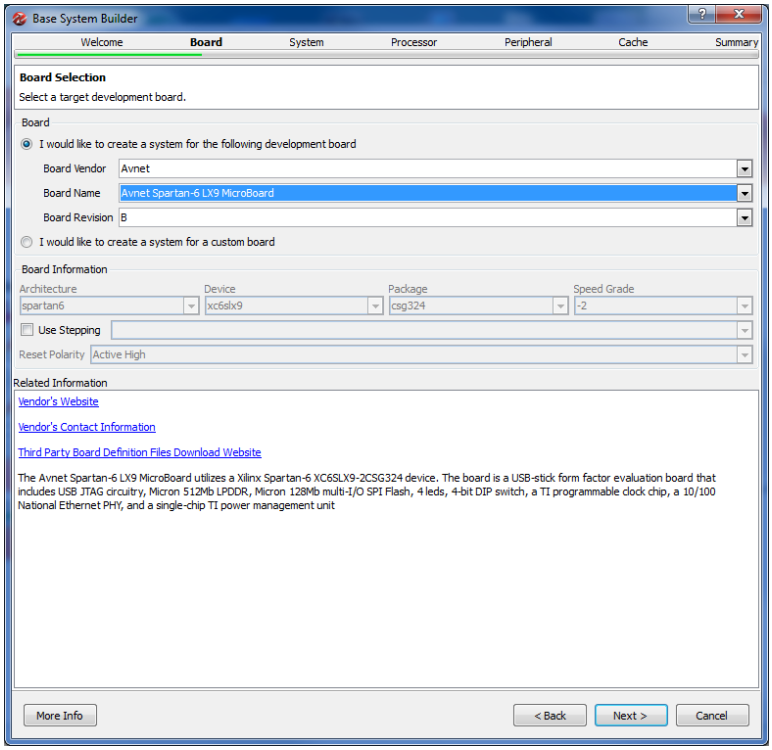
El siguiente paso es seleccionar el directorio donde vamos a diseñar nuestro sistema, le llamaremos Ejercicio_1 y dejaremos el nombre system.xmp para nuestro sistema empotrado. Usaremos el Bus PLB para trabajar en estos diseños.



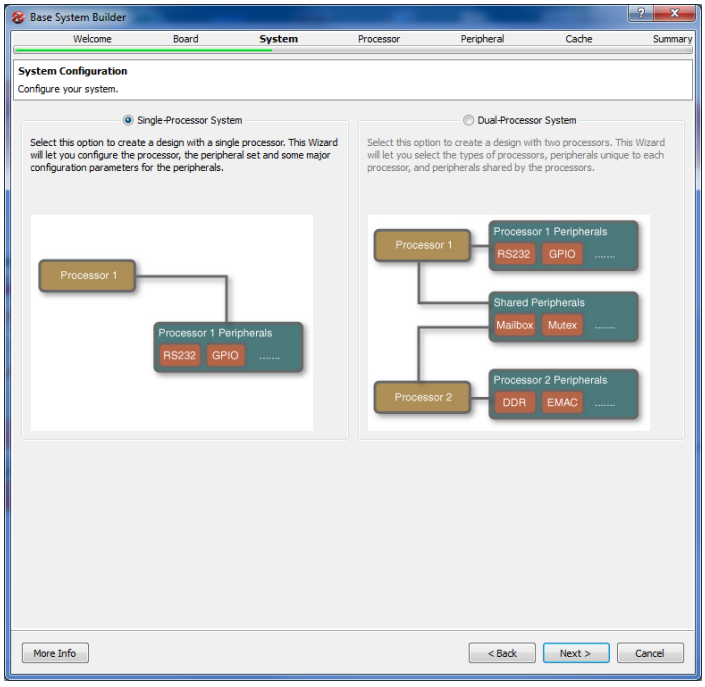
Seleccionamos la opción de crear un nuevo diseño con BSB.



El siguiente paso es definir la placa con la que vamos a trabajar, en nuestro caso definimos el fabricante (Avnet), el nombre de la tarjeta (Avnet Spartan-6 LX9 MicroBoard) y la revisión de la misma (B). En la parte inferior podéis leer una breve descripción de la placa y conocer los recursos de los cuales se dispone.



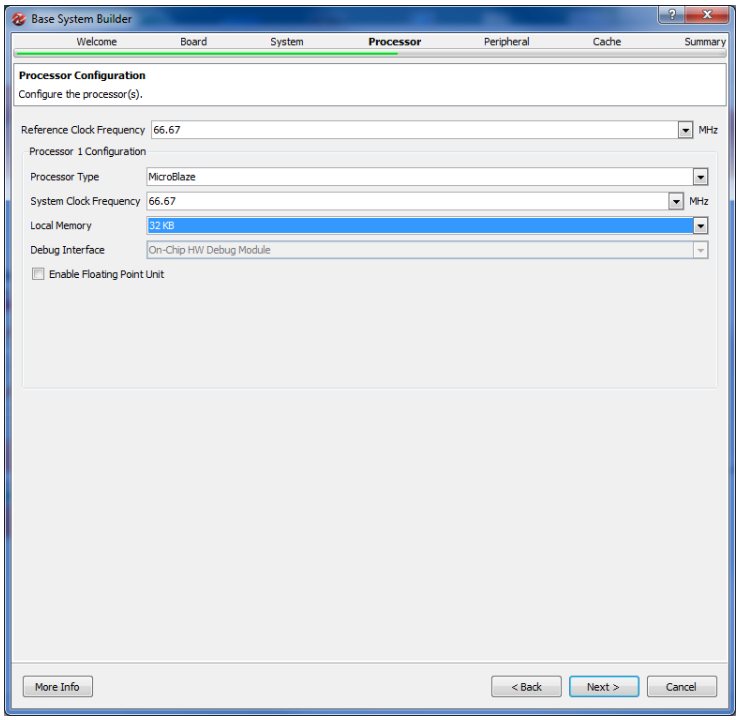
Seleccionamos el microprocesador con el que vamos a trabajar, en este caso Microblaze. Con esta FPGA sólo se puede trabajar con éste. Así mismo, podemos observar la descripción del procesador que vamos a empotrar en la FPGA.



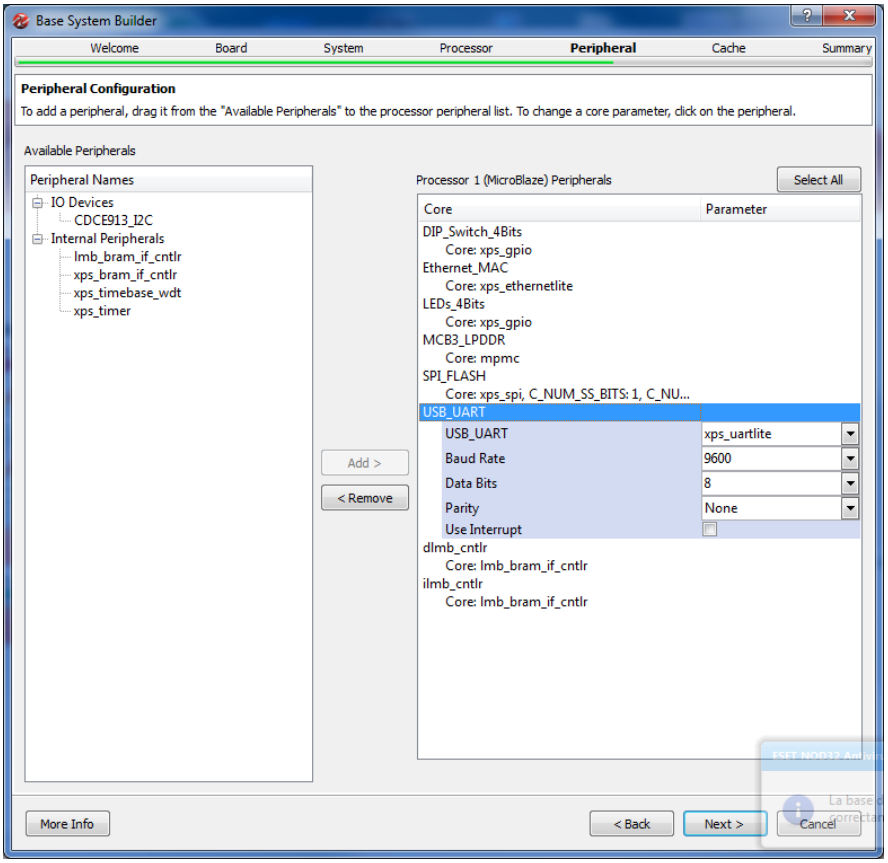
El siguiente paso nos permite ajustar las frecuencias del reloj global y del reloj del bus del microprocesador (usaremos 66.67 MHz que vienen por defecto). Seleccionamos el módulo de depuración hardware (nos permite depurar el sistema incorporando un analizador lógico interno) y escogemos un tamaño de memoria local de 32 KB.

Éste es el máximo tamaño de memoria local que se puede definir en esta FPGA ya que la herramienta EDK sólo permite valores potencia de 2.

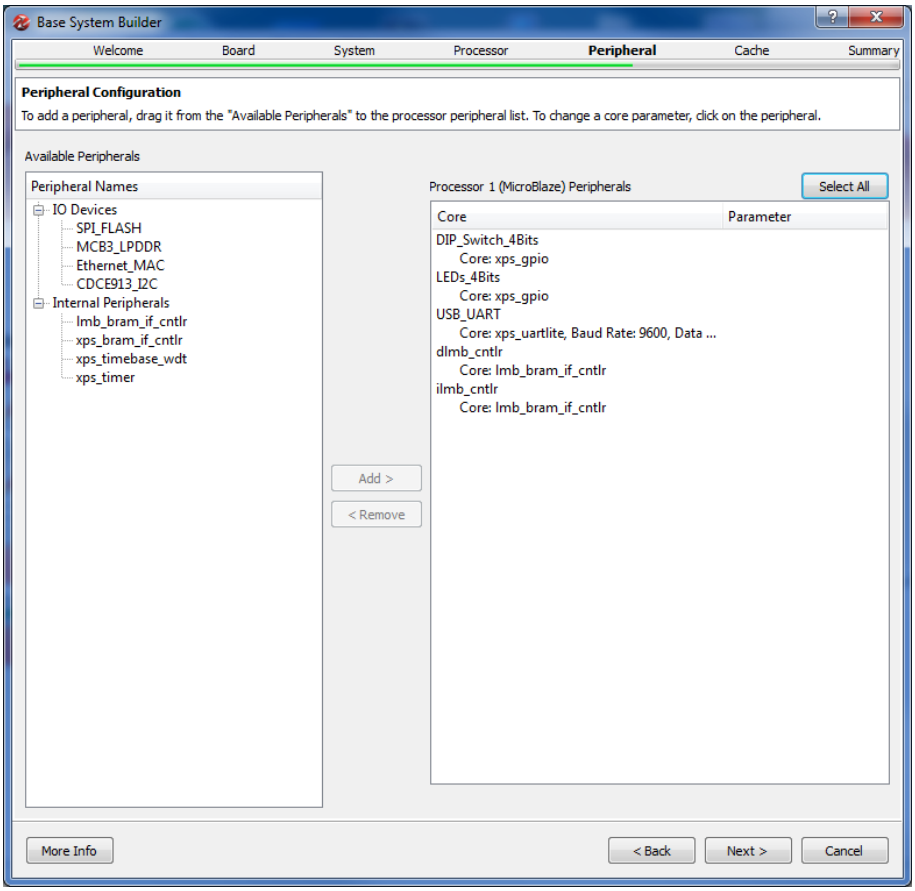
En este caso no utilizaremos Cache.



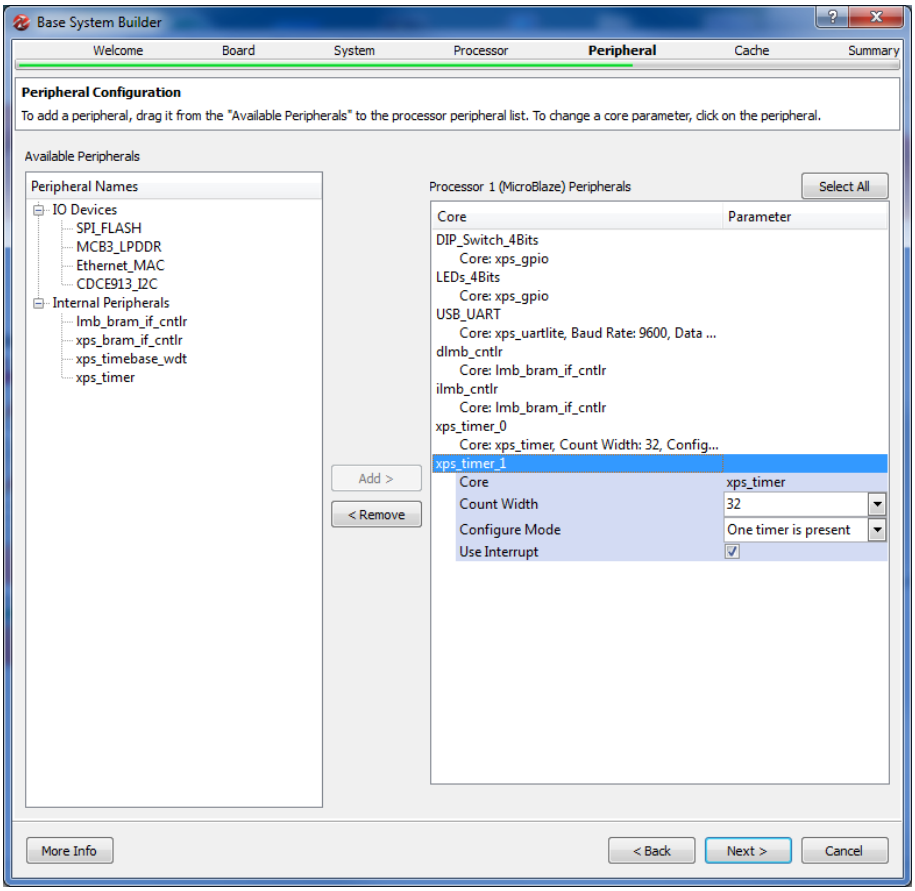
Configuramos el periférico del Puerto Serie (USB_UART) que es el que usaremos en esta aplicación con los valores de la siguiente figura. Si tenéis alguna duda sobre el periférico, podéis ir al datasheet directamente desde esta ventana. También seleccionaremos los LEDs para poderlos usar y los DIP_Switch.



No usaremos el controlador de I2C, tampoco Ethernet. La memoria FLASH externa que incorpora la placa con comunicación SPI tampoco la usaremos. Por tanto, nos debe quedar lo que veis en la siguiente imagen.



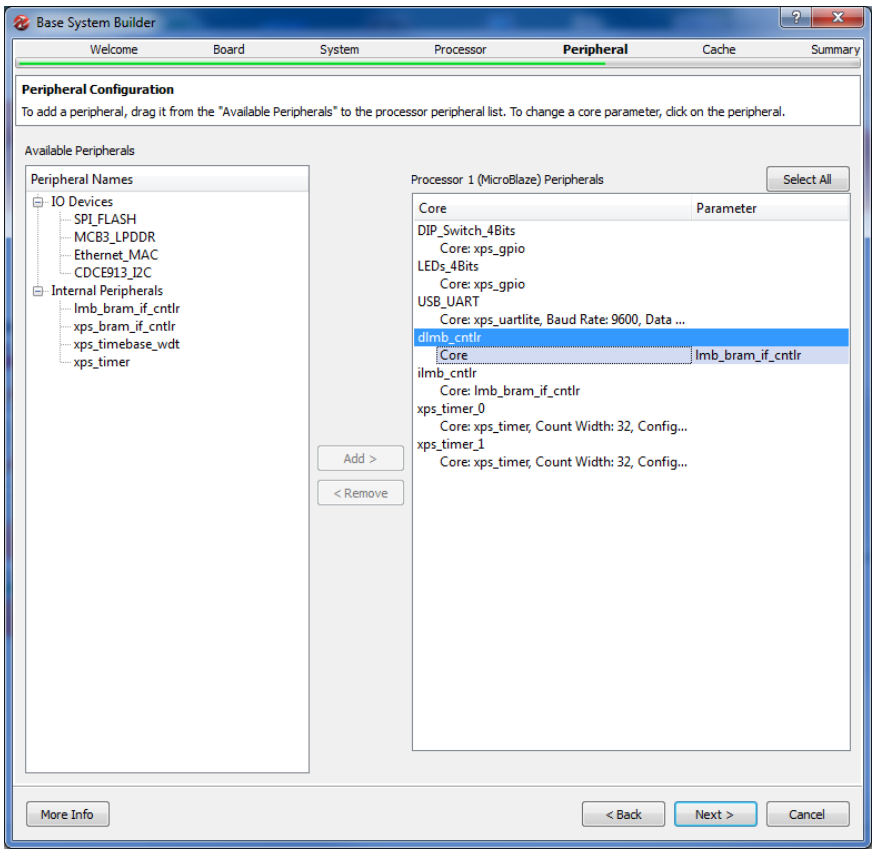
XPS nos da la opción de añadir un nuevo periférico, para lo cual pinchamos sobre Add Peripheral. Seleccionamos el periférico temporizador que viene predefinido (XPS_TIMER). Hacerlo dos veces para tener dos temporizadores. Seleccionamos los dos temporizadores con las opciones indicadas en la imagen anterior (uso de 1 sólo Timer y uso de interrupción en los mismos)



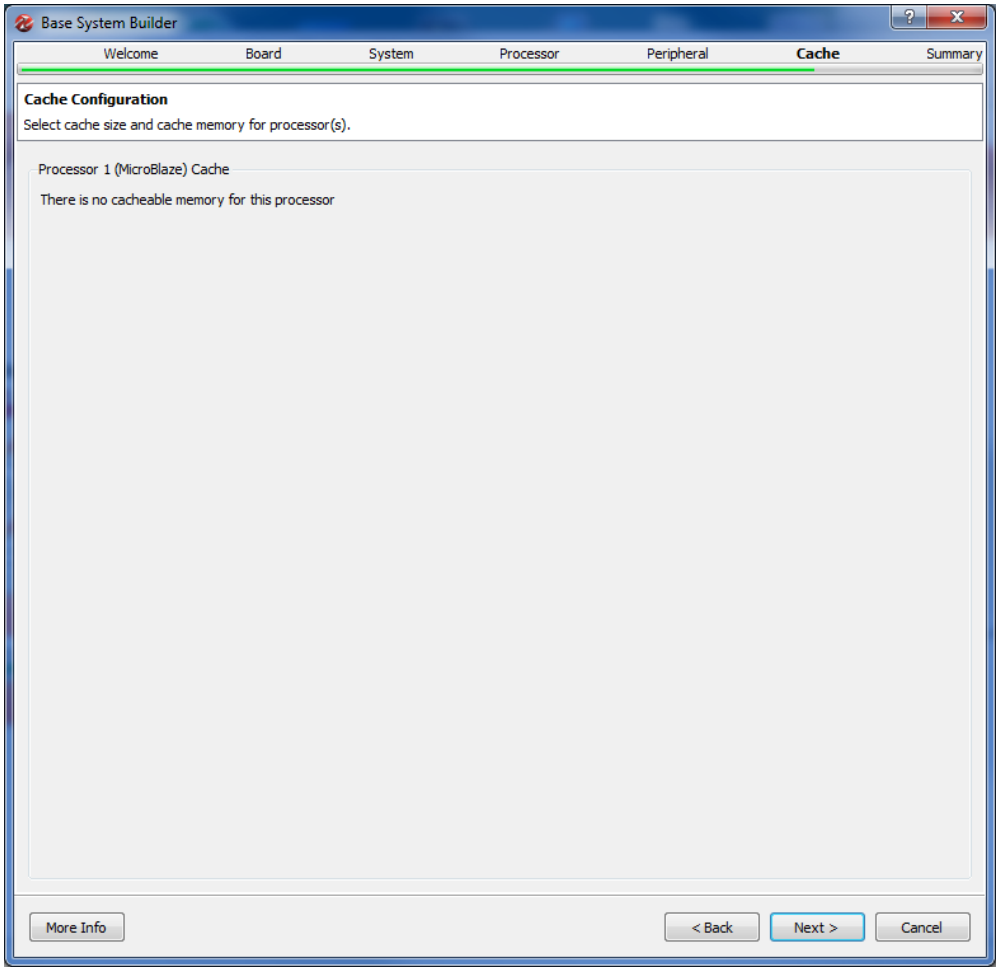
En este Ejercicio realizaremos un test rápido del sistema empotrado que estamos definiendo, para lo cual, usaremos el interfaz USB_UART que es el dispositivo de entrada y salida estándar.

Las aplicaciones software para configurar el test de memoria y el test de periféricos, utiliza los controladores por defecto de la memoria de instrucciones, de datos y de la pila.

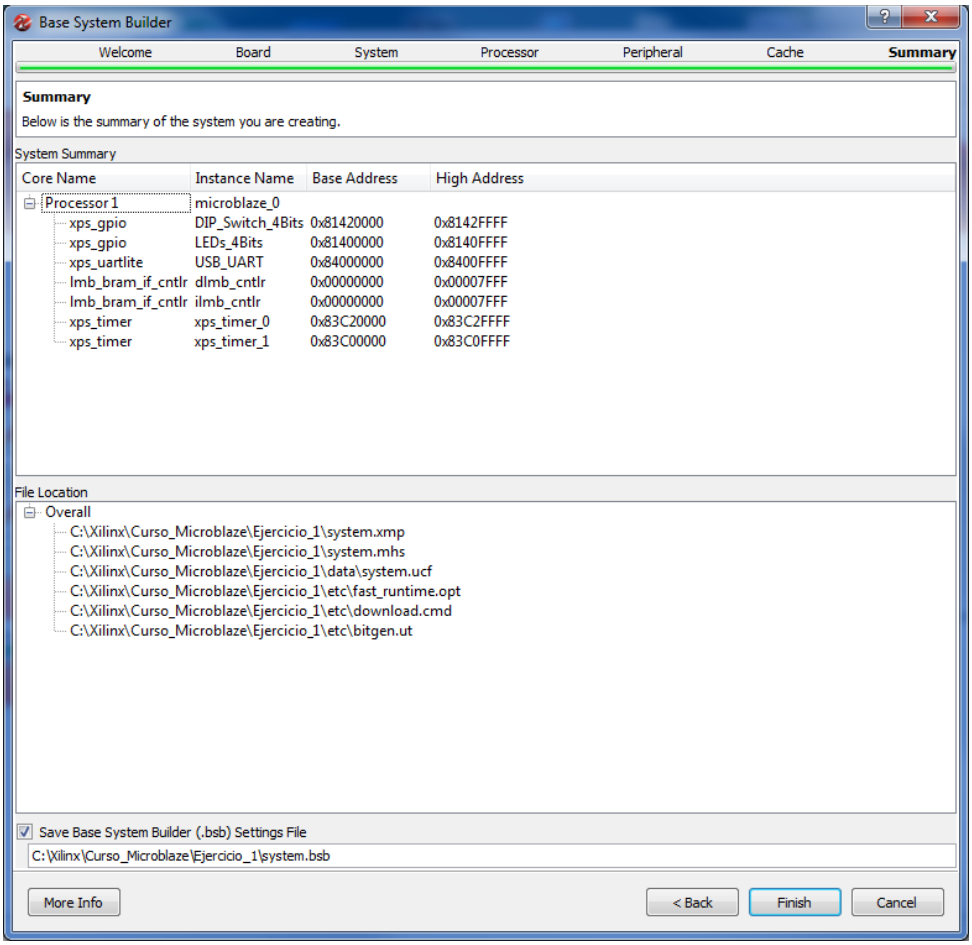
Estos controladores corresponden a la memoria interna y utilizan el bus local (LMB), son ilmb_cntrl y dlmb_cntrl. Por tanto, también hay que mantenerlos en el diseño que estamos realizando.



Confirmamos que no usamos la caché.

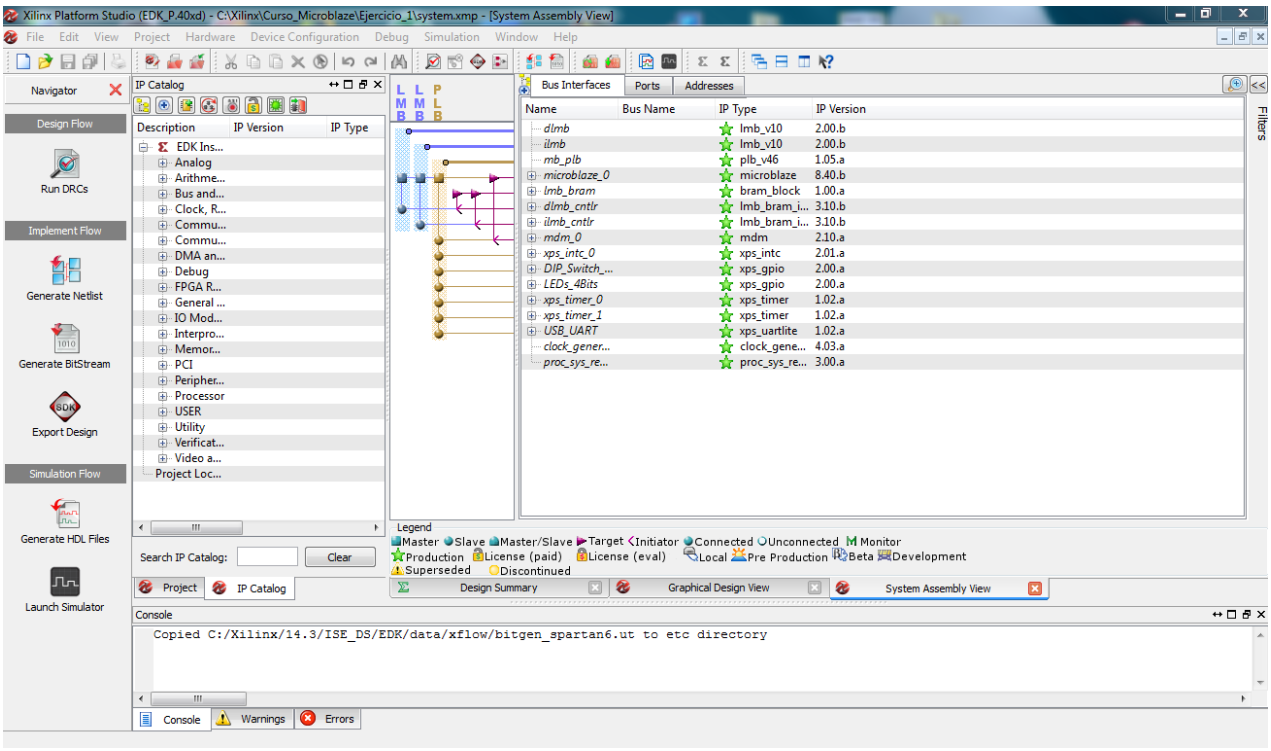


Revisamos que todo esté correcto y generamos el microprocesador embebido con las opciones escogidas. Podemos observar en éste resumen las direcciones de memoria asignadas a cada periférico.



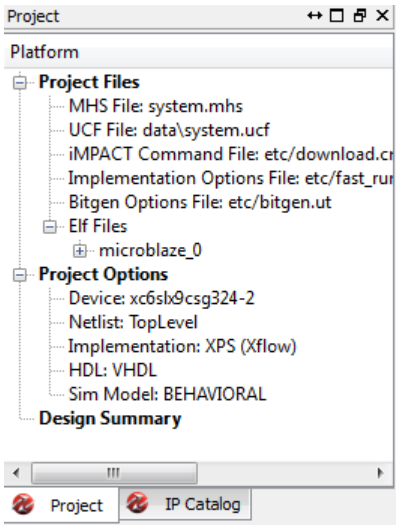
Descripción de XPS - EDK

Una vez terminada la generación hardware del procesador embebido aparece la ventana principal del programa XPS.

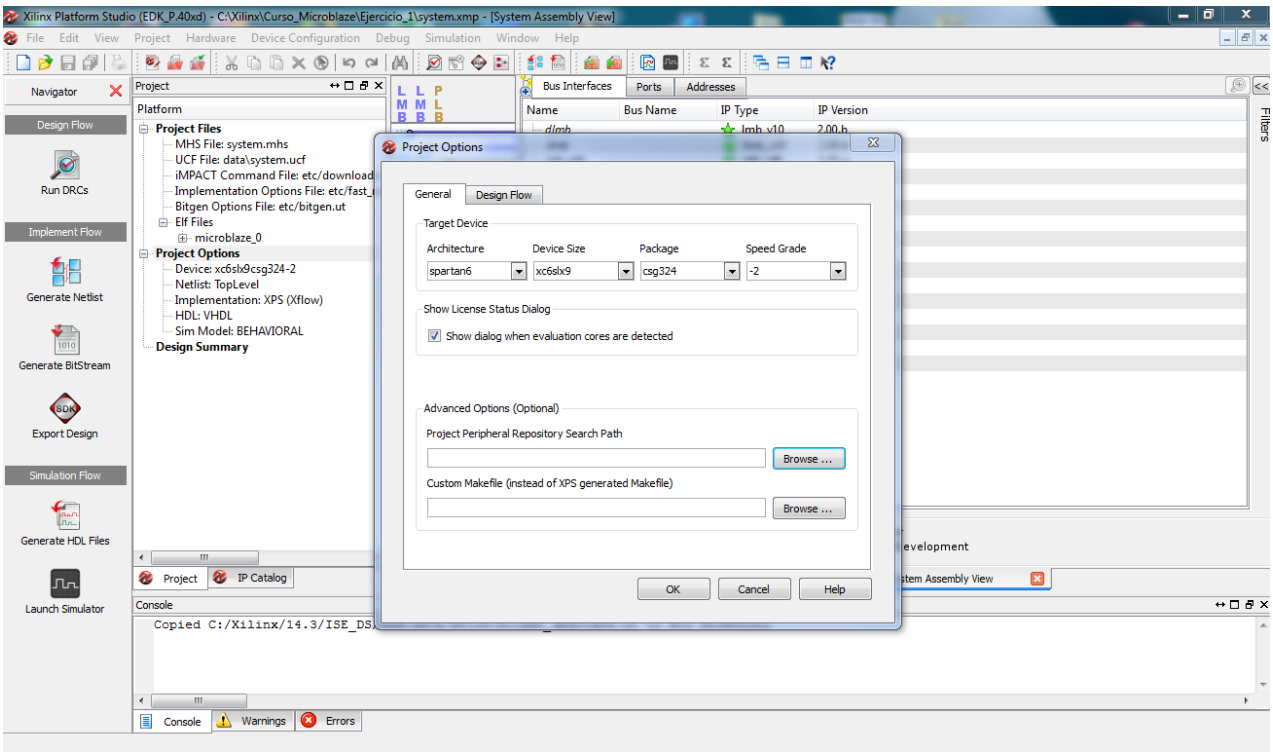


De la que podemos destacar en la parte izquierda el área de información del proyecto (Project Information Area), tenemos varias pestañas para poder trabajar con la aplicación creada.

- Con Project podemos acceder a los principales archivos del proyecto.

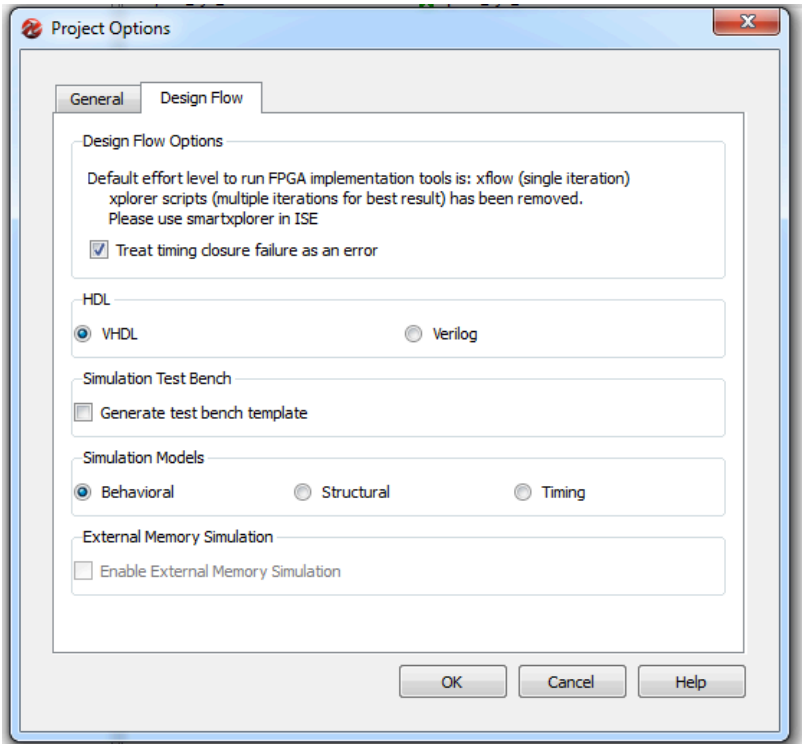


Esta pestaña, también, nos permite acceder a las opciones principales del proyecto pinchando dos veces sobre Project Options, opción también disponible en Project -> Project Options.

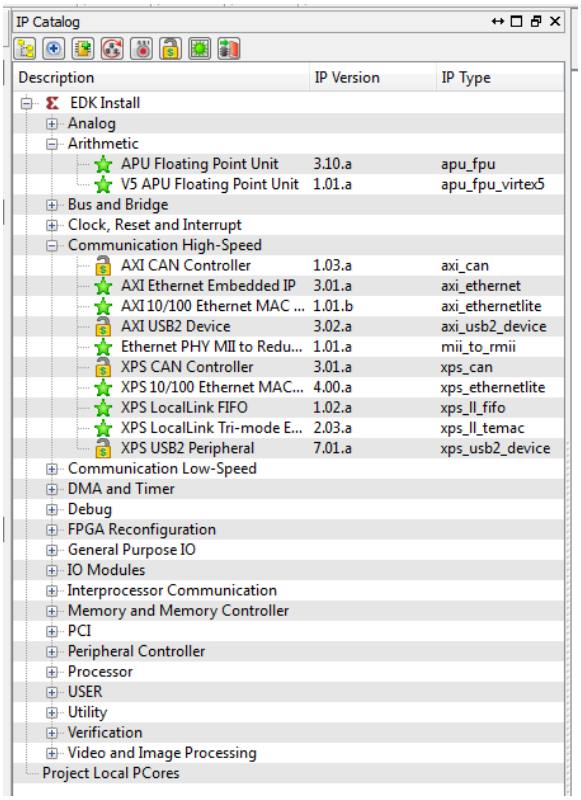


Esta nueva ventana tiene tres pestañas, la primera nos da las opciones del proyecto relacionadas con el dispositivo elegido y el directorio de definición de periféricos de usuario (repository).

La segunda pestaña nos da las opciones del proyecto relacionadas con la generación de archivos HDL y la simulación.

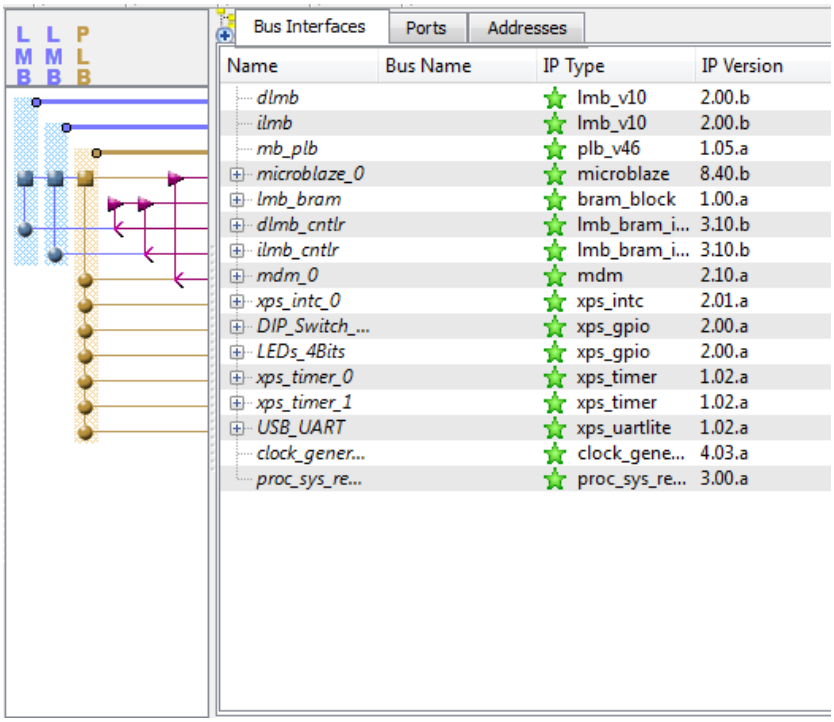


- Con IP Catalog podemos añadir cualquier periférico predefinido en XPS. Echad un vistazo a los periféricos incluidos.

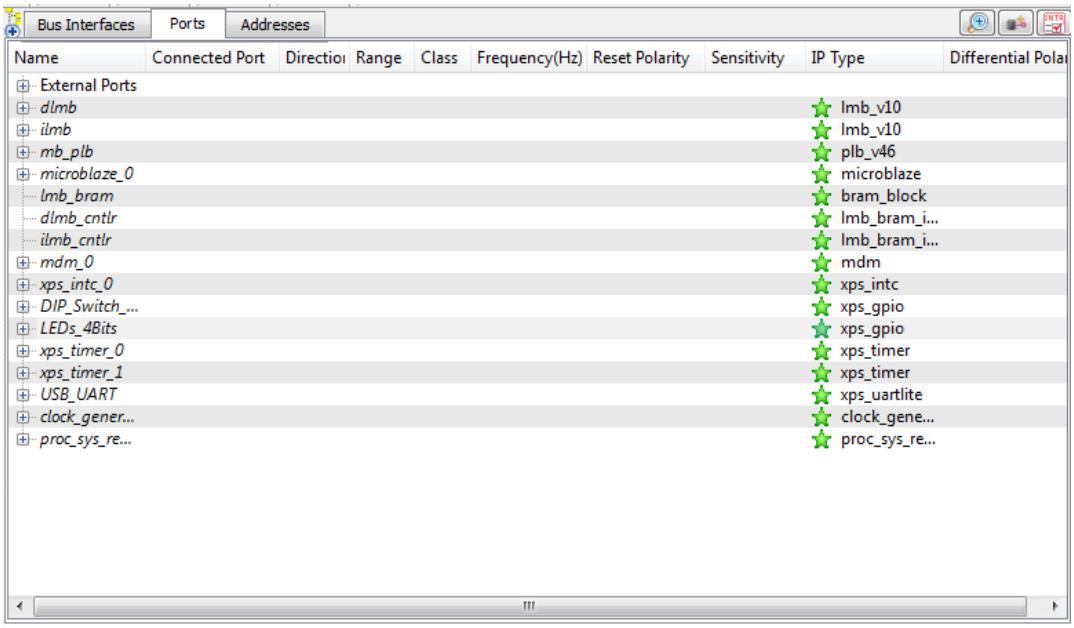


En la parte derecha de la ventana principal tenemos la vista del sistema (System Assembly View).

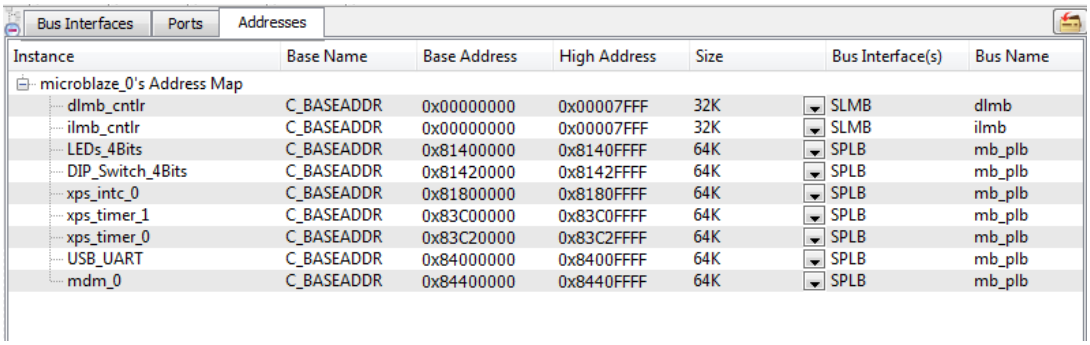
- La opción Bus Interface nos permite ver las conexiones entre los distintos bloques que forman el sistema empotrado que hemos definido.



- La opción Ports nos permite ver los distintos terminales (entradas y salidas) de cada bloque del sistema empotrado.

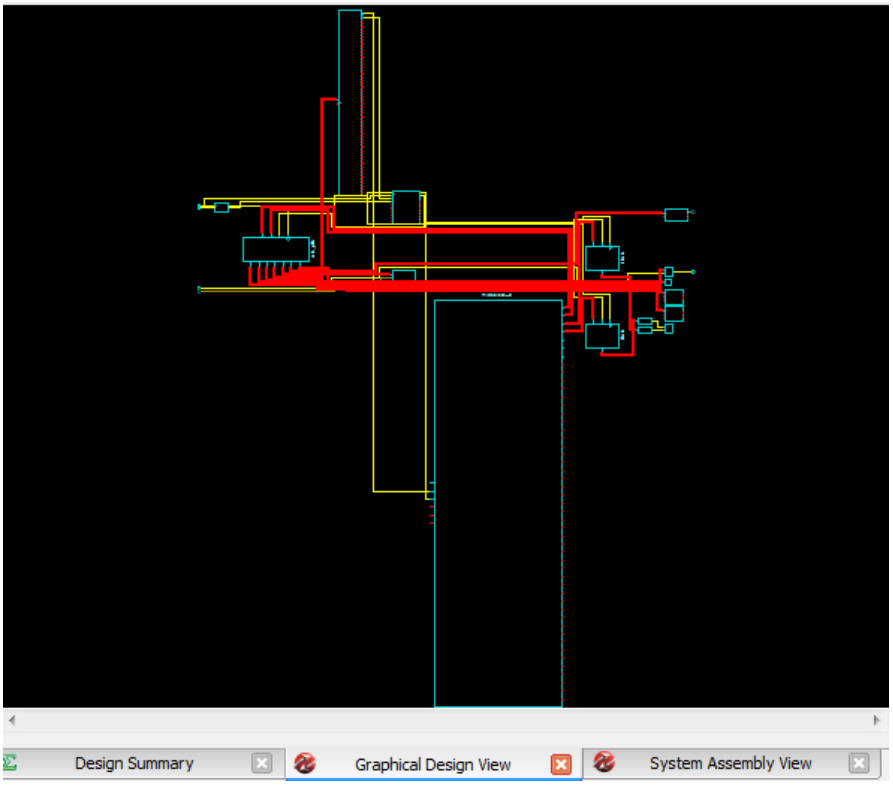


- La última opción es Addresses, que nos permite ver las direcciones asignadas a los distintos periféricos que forman el sistema empotrado.

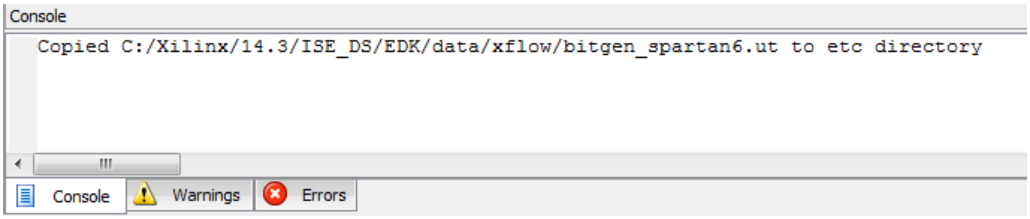


Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name
microblaze_0's Address Map						
dlmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB	dlmb
ilmb_cntlr	C_BASEADDR	0x00000000	0x00007FFF	32K	SLMB	ilmb
LEDs_4Bits	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb_plb
DIP_Switch_4Bits	C_BASEADDR	0x81420000	0x8142FFFF	64K	SPLB	mb_plb
xps_intc_0	C_BASEADDR	0x81800000	0x8180FFFF	64K	SPLB	mb_plb
xps_timer_1	C_BASEADDR	0x83C00000	0x83C0FFFF	64K	SPLB	mb_plb
xps_timer_0	C_BASEADDR	0x83C20000	0x83C2FFFF	64K	SPLB	mb_plb
USB_UART	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb_plb
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb_plb

También tenemos una pestaña más denominada Graphical Design View en la que se puede observar el esquemático del hardware creado.



En la zona inferior tenemos la zona de la consola para ver los mensajes, errores, etc… (Console Window).



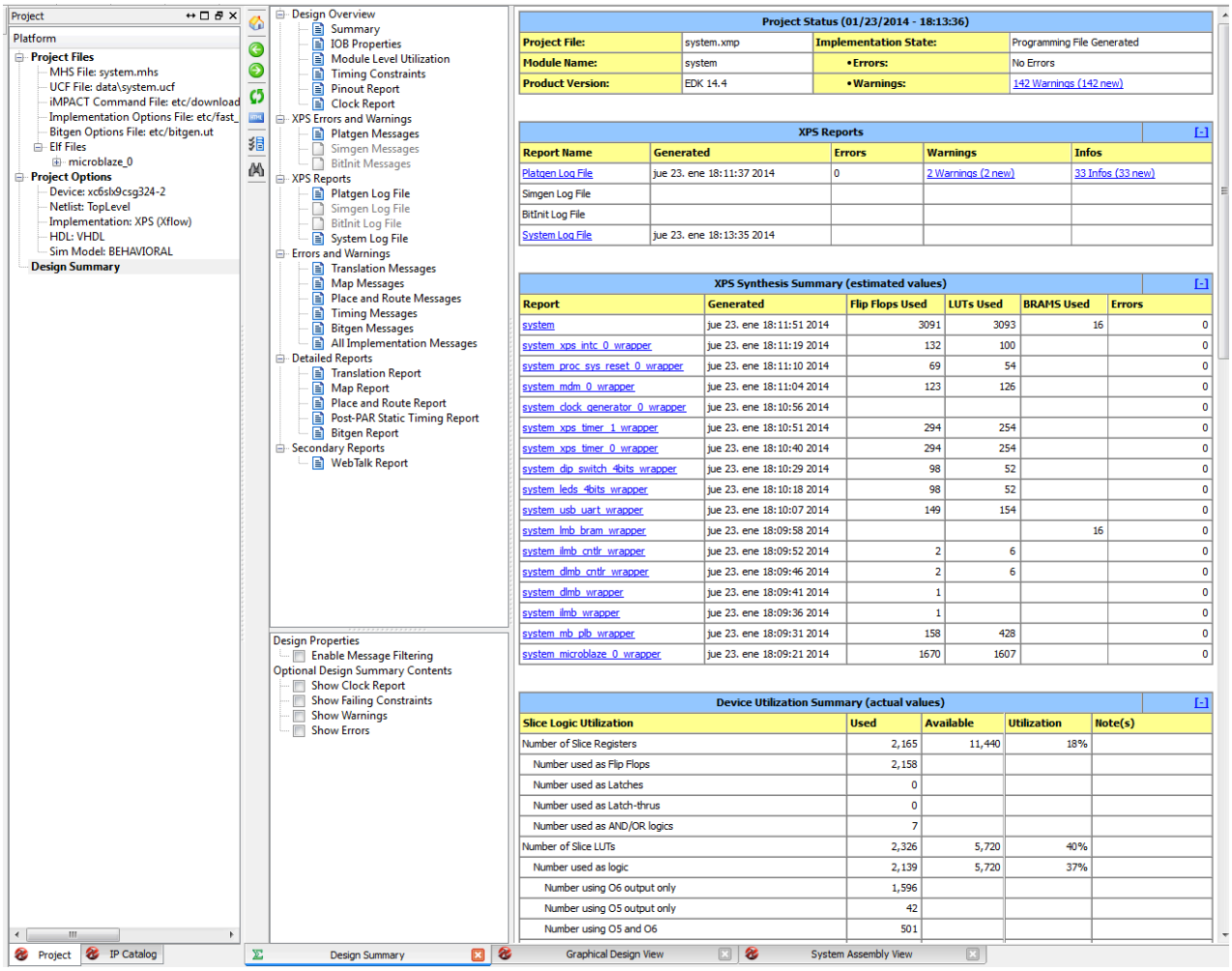
Descripción archivos

Los archivos generados más importantes que debemos conocer son:

- System.mhs (Microprocessor Hardware Specification): Definición del hardware del sistema empotrado, accesible a través de la pestaña Project del área de información.
- System.mms (Microprocessor Software Specification): Definición del software del sistema empotrado.
- System.ucf (User Constraints File): Definición de restricciones del sistema empotrado, tanto temporales como de asignación de terminales.

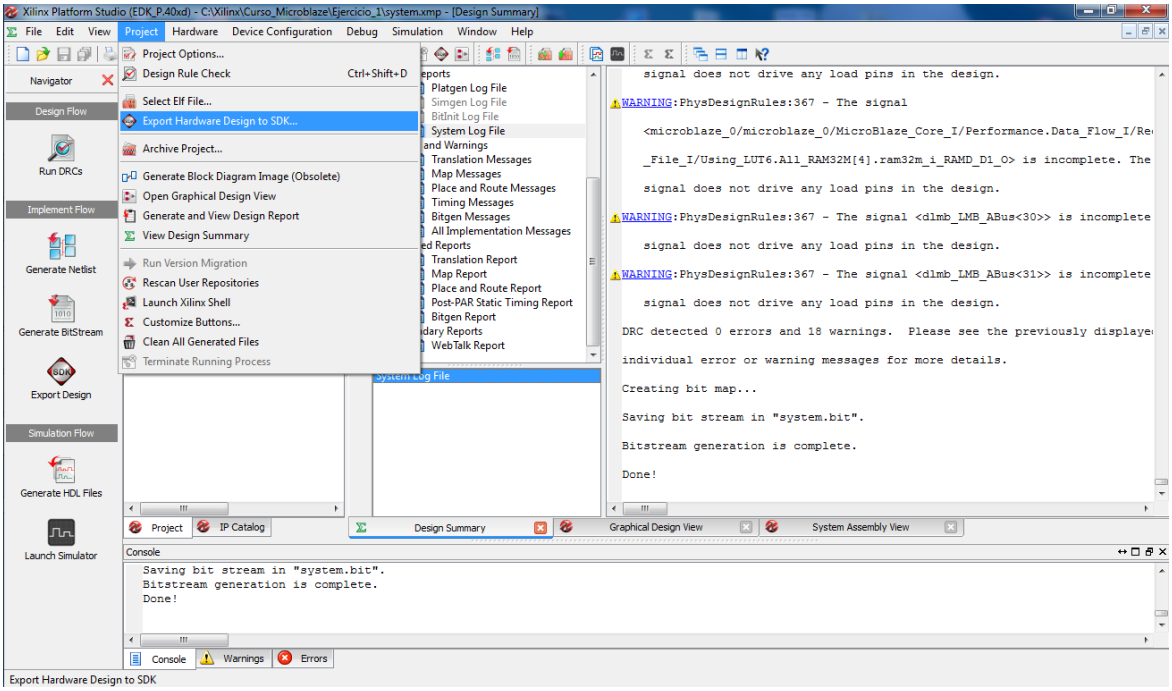
Implementación del sistema empotrado - EDK

La implementación hardware del sistema empotrado se realiza ejecutando la opción Hardware -> Generate Bitstream. Una vez terminada la implementación, tarda unos cuantos minutos, podemos ver los recursos que ha ocupado este diseño en la FPGA entrando en Design Summary dentro de la pestaña Project.

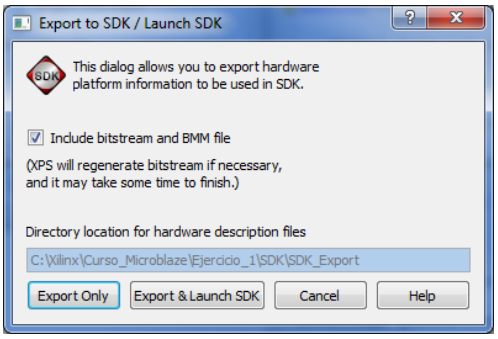


Diseño del software - SDK

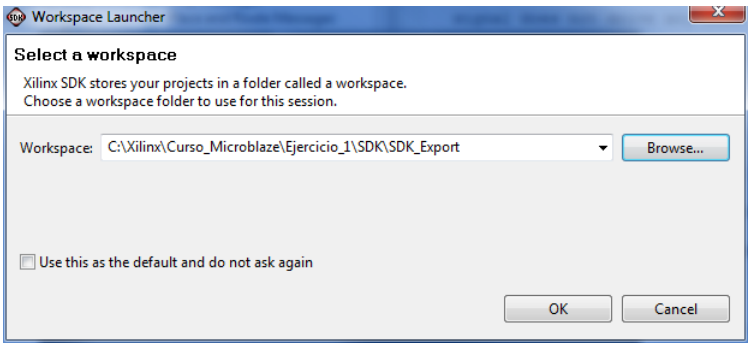
Una vez definido el hadware a medida e interno de la FPGA, ya podemos programar el microprocesador embebido (MicroBlaze). Para ello, las opciones referidas al desarrollo del software para el sistema empotrado son accesibles a través de la opción Project -> Export Hardware Design to SDK…



Se generará una carpeta denominda SDK dentro de la carpeta del Proyecto en la que se introducirá la exportación del hardware generado con EDK. Para abrir el entorno SDK se debe seleccionar la opción Export & Launch SDK.

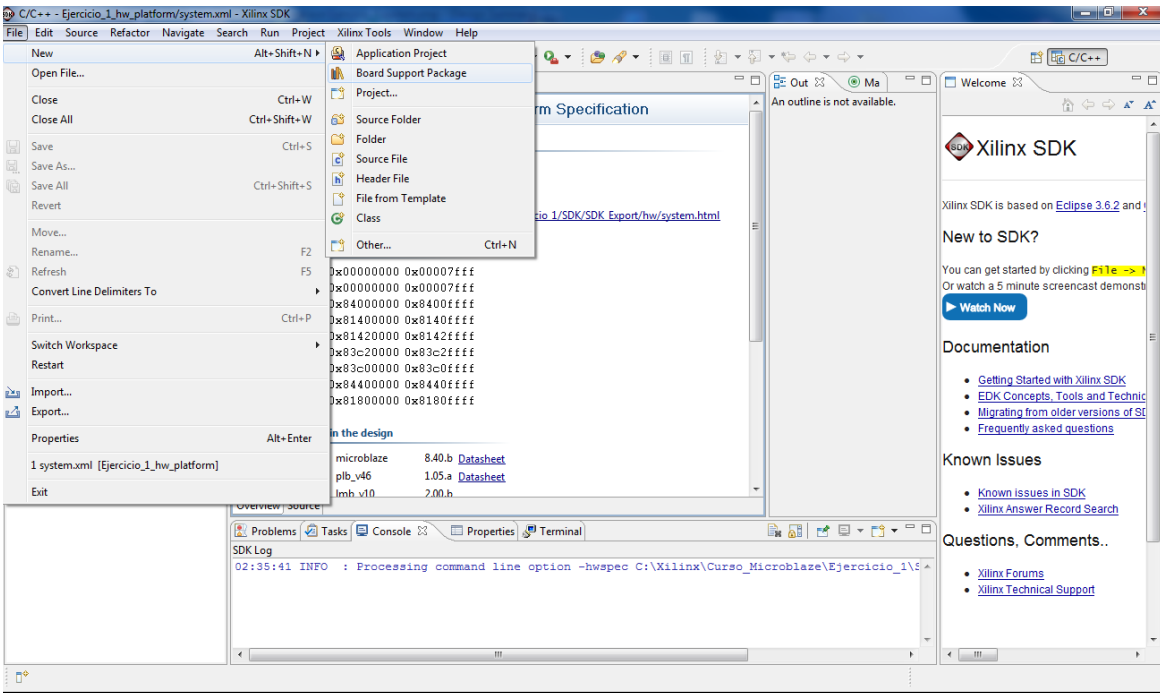


A continuación se selecciona el área de trabajo que deseamos emplear, eligiendo la carpeta SDK_Export dentro de la carpeta del Proyecto que hemos generado.

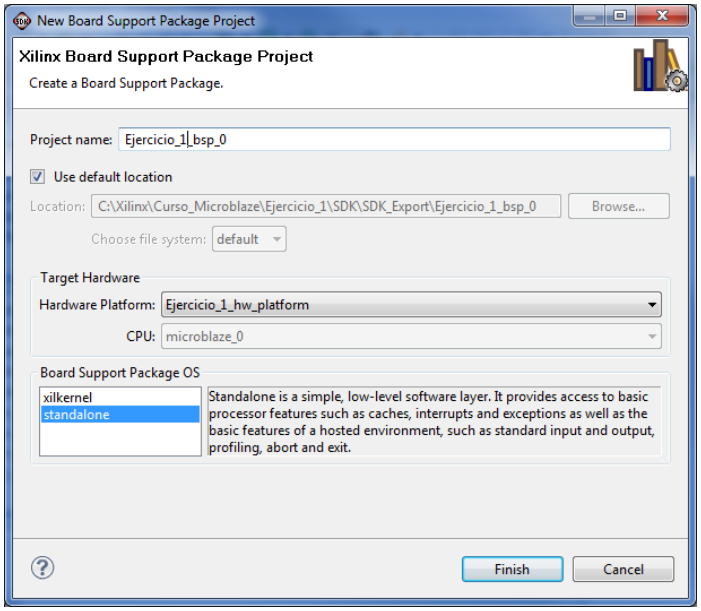


Cuando arranca SDK, lo único que tenemos es la parte hardware creada y exportada (Ejercicio_1_hw_platform.

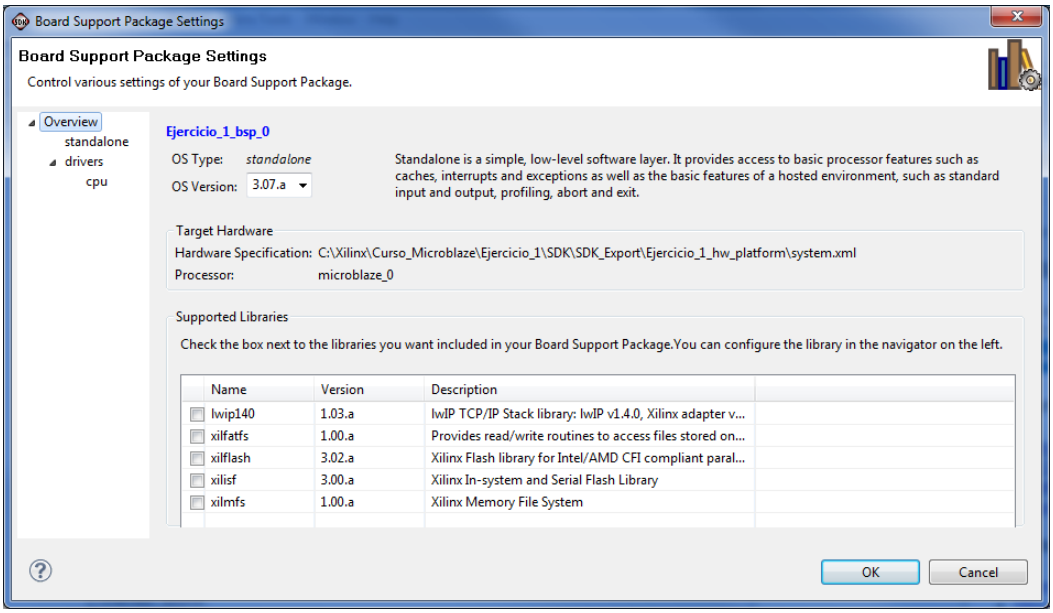
La generación de las bibliotecas y de los ficheros de descripción de la PCB usada (BSP: Board Support Package) se realiza mediante la ejecución de la opción File -> New -> Board Support Package. Hacerlo para generar los programas de test del microprocesador empotrado.



Podemos cambiar el nombre por defecto del bsp que vamos a crear para identificarlo con nuestro Ejercicio. La plataforma hardware debe coincidir con la que acabamos de exportar. Y como no usamos ningún sistema operativo, dejáremos la opción standalone.



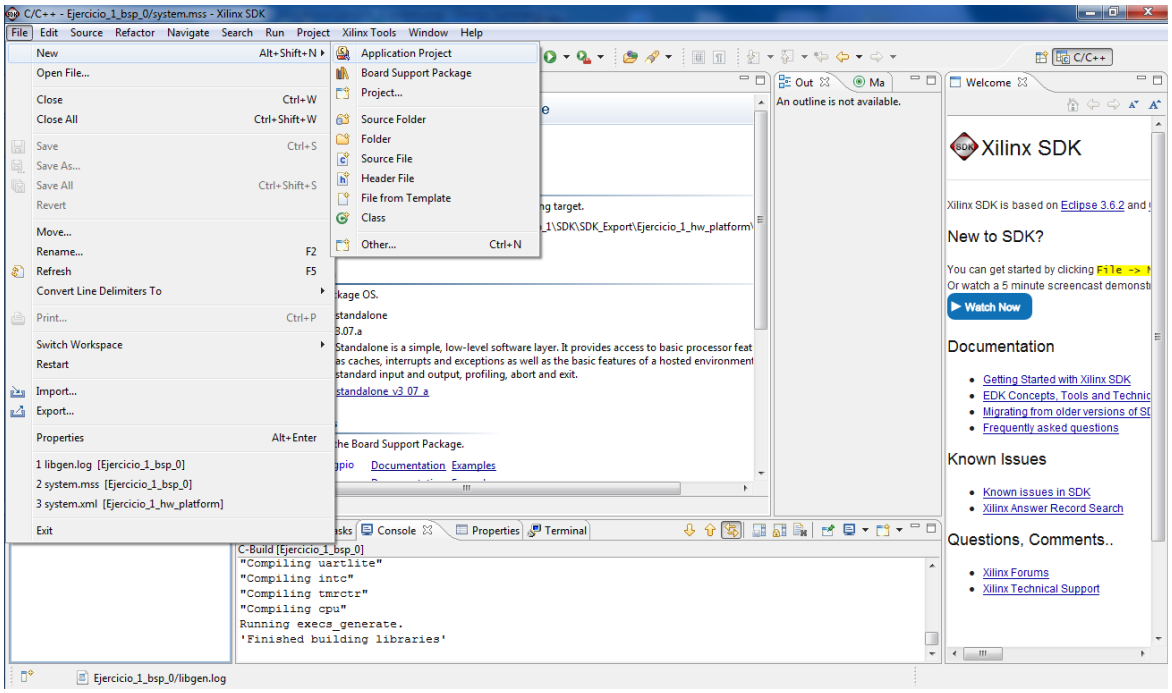
En la siguiente ventana que nos aparece (Overview) podemos observar y cambiar las opciones referidas al desarrollo software de nuestro sistema empotrado; estas son los parámetros de la CPU, el sistema operativo y las bibliotecas que se van a usar.



En la pestaña Drivers tenemos las opciones de los programas que manejan los distintos periféricos.

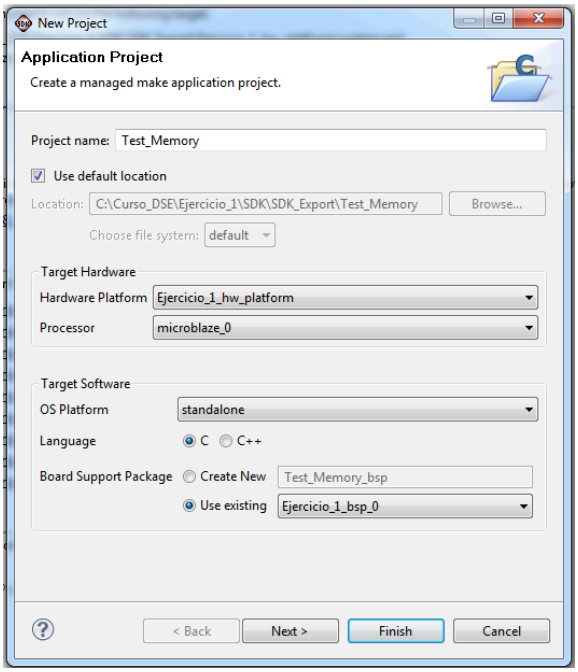
Ahora ya estamos en condiciones de crear y compilar cualquier aplicación software. Por tanto, vamos a generar la aplicación TestApp_Memory que realiza una comprobación de la memoria del sistema. De forma que se guarde en la memoria interna de la FPGA (BRAM).

Para ello, creamos una nueva aplicación para nuestro sistema empotrado. File -> New -> Application Project.

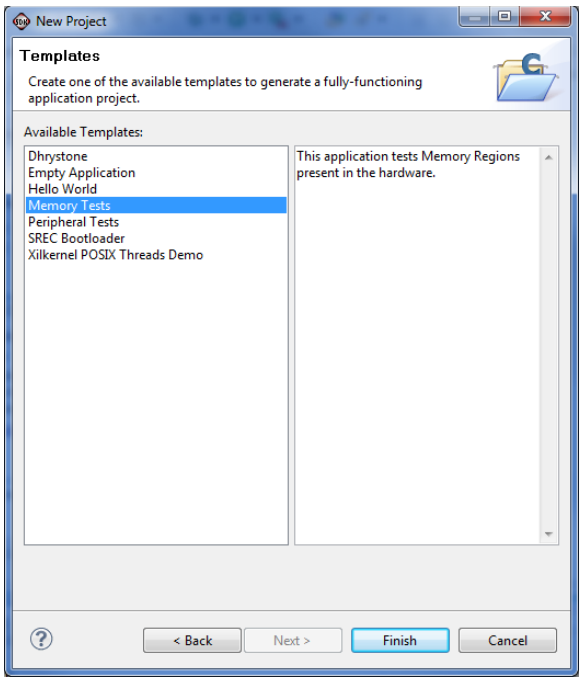


Le damos el nombre Test_Memory, dejamos la plataforma hardware creada (Ejercicio_1_hw_platform), usamos el Lenguaje C y seleccionamos el bsp existente y creado anteriormente (Ejercicio_1_bsp_0)

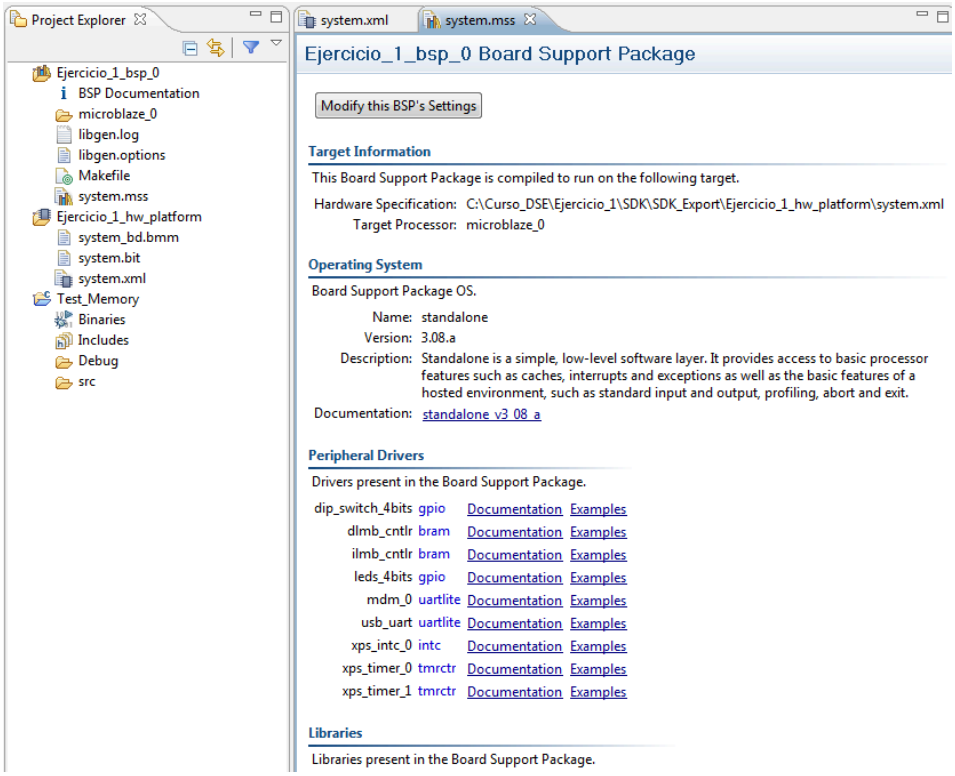
Y, cuidado, no finalizamos. Debemos darle a Next.



En la siguiente pantalla, tenemos varias aplicaciones ya creadas y definidas por Xilinx para poder probar nuestro sistema. Escogemos Memory Tests y finalizamos.

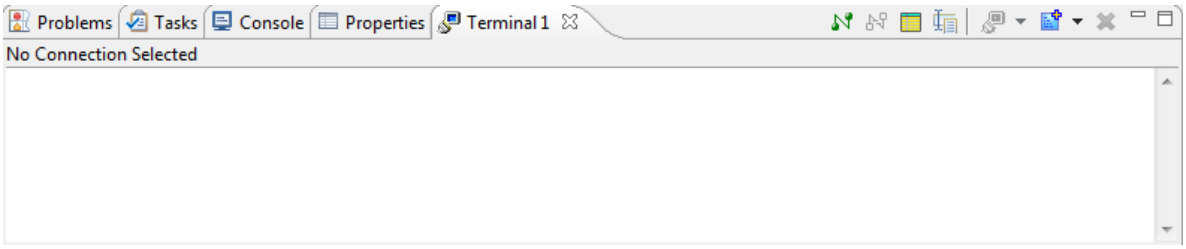


SDK automáticamente compilará todo el sistema y nos informará si hay algún error. Debemos tener algo así.

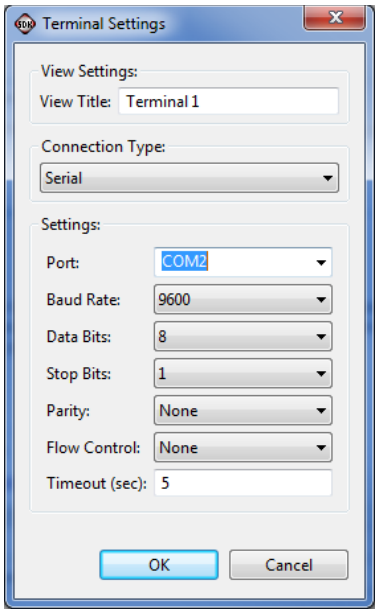


Antes de programar la FPGA y poner en funcionamiento el sistema empotrado que hemos diseñado, vamos a ejecutar el Terminal para tener acceso al puerto USB_UART (periférico que hemos diseñado como entrada/salida estándar de nuestras aplicaciones software). Conectar el cable Serie y el cable USB principal de alimentación de la FPGA. Verificar los drivers de ambas conexiones.

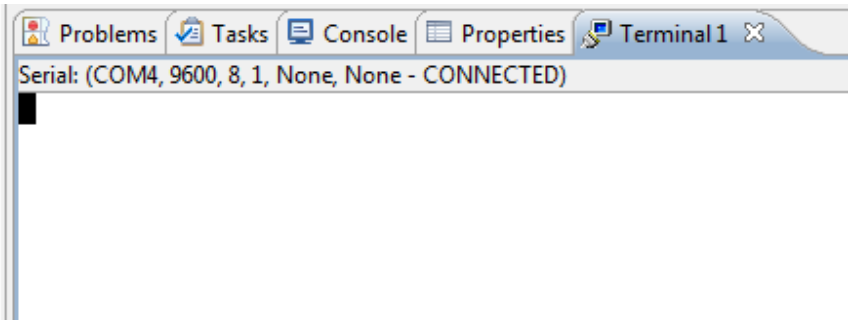
En la parte inferior aparece la pestaña Terminal mediante la cual se tendrá acceso al tráfico serie que transcurre desde el PC a la FPGA y viceversa. Si no aparece esta pestaña ir a Window -> Show View -> Terminal.



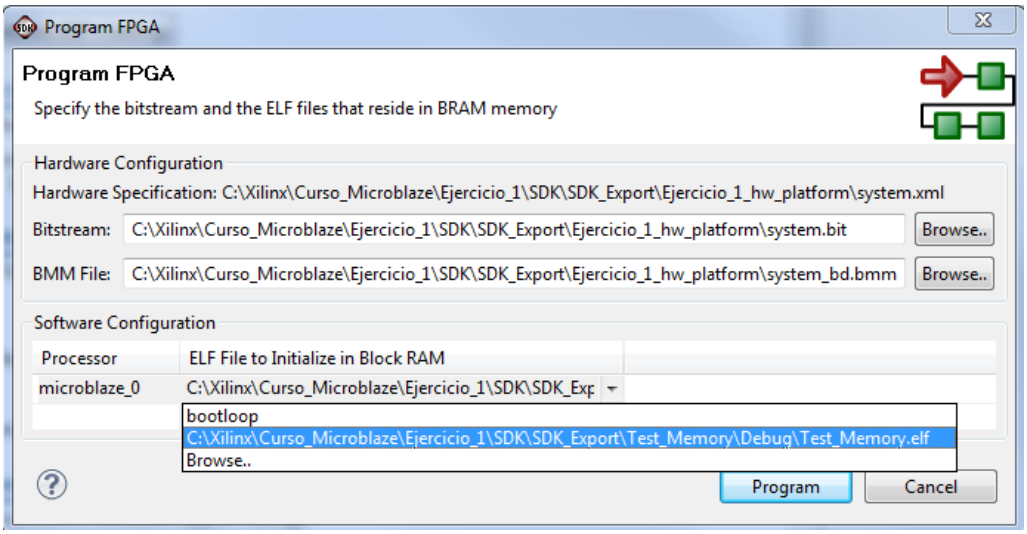
Para configurar la conexión clicamos en el tercer icono que aparece en la parte superior derecha del Terminal y se selecciona Connection Type Serial y el puerto al que se ha asignado la FPGA. Esta opción se reconfigurará cuando conectemos la FPGA al PC.



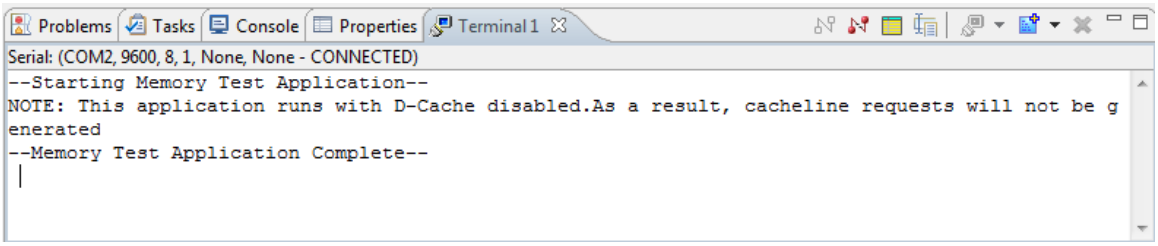
Nos debe aparecer lo siguiente.



Para programar la FPGA, ejecutamos la opción Xilinx Tools -> Program FPGA. Seleccionamos la ubicación del archivo ELF, y se hace clic en Program.

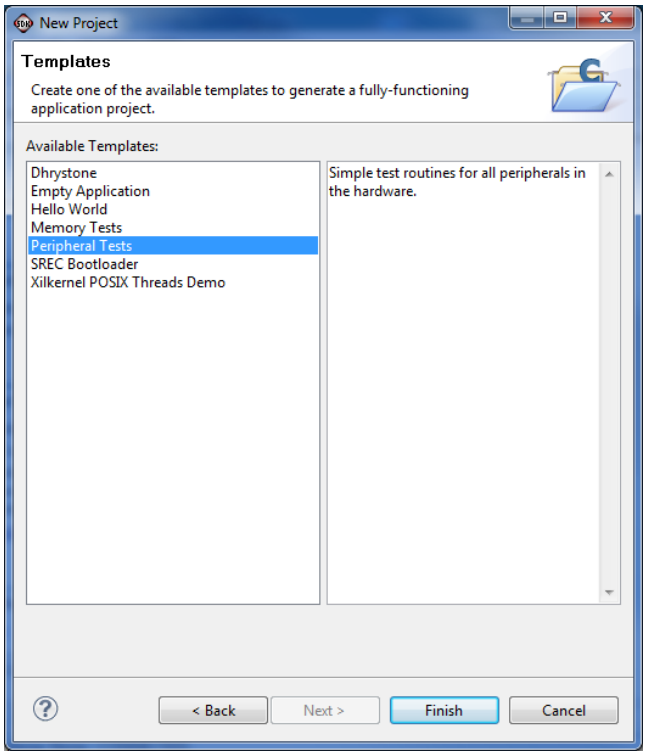


Si observamos la pestaña de Terminal veremos los mensajes enviados por el microprocesador Microblaze empotrado en la FPGA al ejecutar la aplicación software TestApp_Memory que hemos configurado en la FPGA.



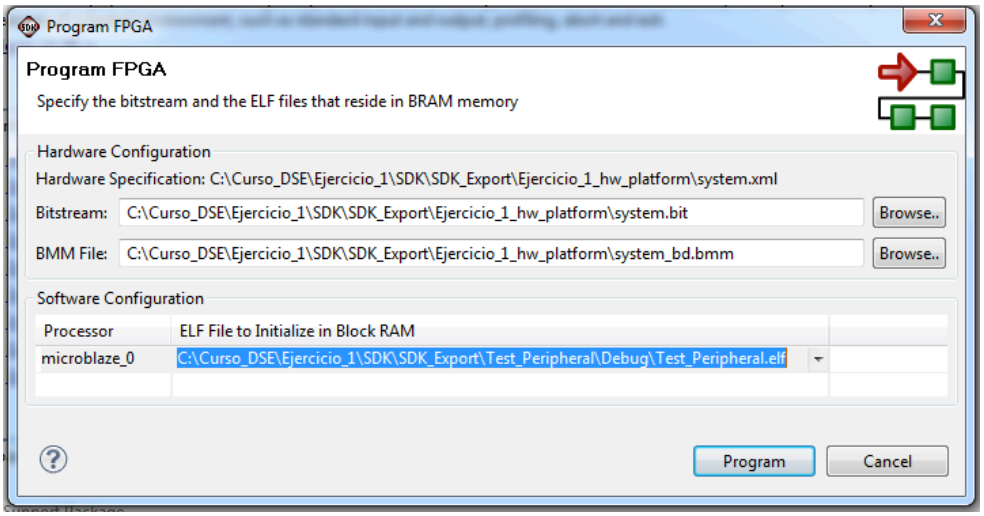
Podemos probar la otra aplicación software de ejemplo que posee SDK. Esta aplicación es Test_Peripheral. Para poder probarla debemos seleccionar de nuevo, File -> New -> Application Project.

Le damos el nombre Test_Peripheral y seleccionamos el bsp existente y creado anteriormente (Ejercicio_1_bsp_0). Hacemos clic en Next y seleccionamos Pheripheral Tests.

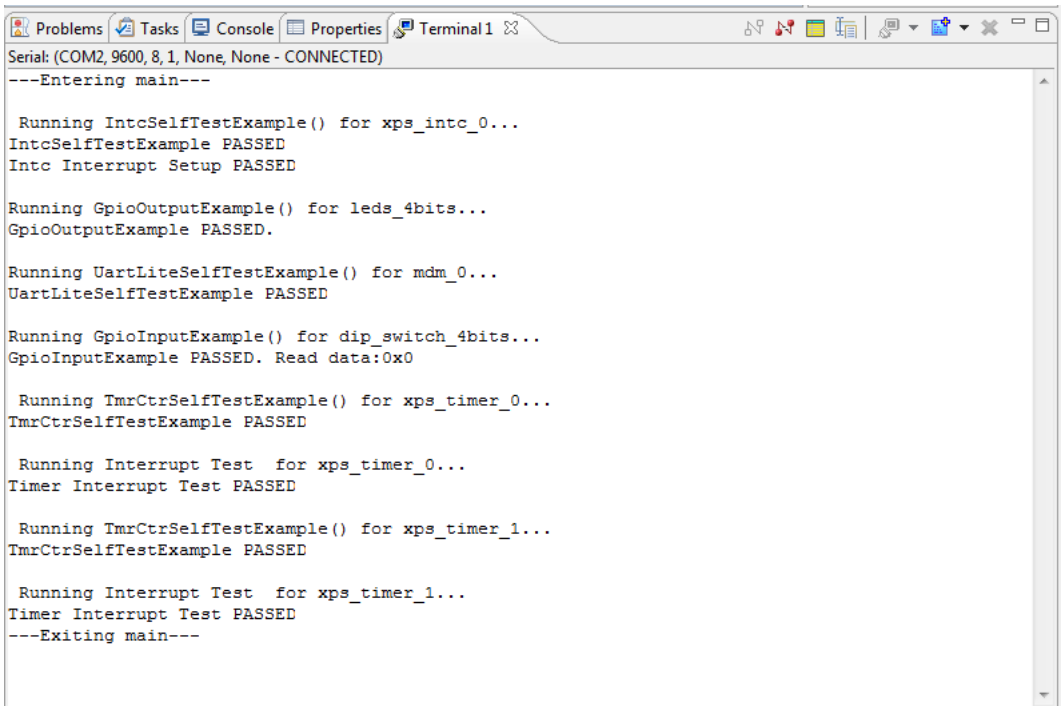


Para programar el proyecto de test de periféricos realizamos lo mismo que con la anterior aplicación. Pueden colgar de un mismo hardware y de un mismo BSP múltiples aplicaciones software. Decidiremos cual programar en cada caso en el paso final.

De nuevo debemos seleccionar Xilinx Tools -> Program FPGA. Seleccionamos la ubicación del archivo ELF, y se hace clic en Program.



En Terminal podemos observar los mensajes enviados por el microprocesador Microblaze empotrado en la FPGA al ejecutar la aplicación software de test de periféricos.



En la placa veremos como se recorren los LEDs. Si queremos comprobar que todo funciona, podemos modificar el valor de los interruptores y volver a lanzar el programa. Veremos como en lugar de obtener un valor de 0x0, tendremos el valor colocado por nosotros.