

# Sistemas Embebidos utilizando FPGAs

---

PEDRO IGNACIO MARTOS

PMARTOS@FI.UBA.AR



# Contenido

- ❑ **Componentes de un Sistema Embebido**
- ❑ Herramienta Vivado
- ❑ Ciclo de Desarrollo de un Sistema Embebido
- ❑ Creación de la Plataforma de Hardware
- ❑ Creación de la Plataforma de Software

# Componentes del Sistema Embebido

---

- ❑ Plataforma de Hardware
  - ❑ Procesador ARM.
  - ❑ Bus AXI.
  - ❑ Periféricos AXI.
  - ❑ Reset, temporización (clocks) y puerto de depuración.
- ❑ Plataforma de Software
  - ❑ Aplicaciones Bare Metal o utilizando Sistema Operativos (FreeRTOS, Linux) según procesador.
  - ❑ Soporte de lenguaje C (librerías standard).
  - ❑ Controladores de Hardware
- ❑ Aplicación del Usuario
  - ❑ Código Principal
  - ❑ Rutinas de Interrupción

# Processor System & Programmable Logic

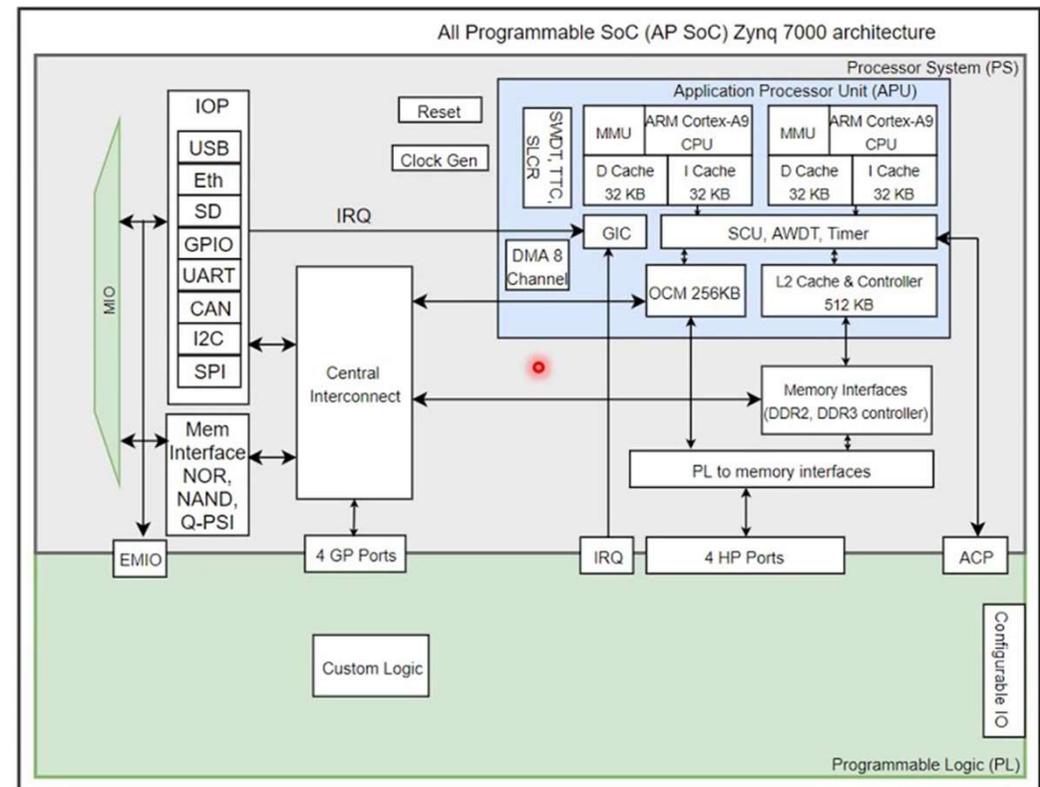
---

- ❑ Una FPGA utilizada para sistemas embebidos se compone de dos grandes bloques, el sistema de procesamiento (PS) y el bloque de lógica programable (PL)
- ❑ PS – Processing System:
  - ❑ Un procesador ARM de 32bits (Cortex A9) o 64bits (Cortex A53) en configuración single, dual o quad core dependiendo de la FPGA.
  - ❑ Periféricos Asociados (SPI, I2C, USB, Serie, etc)
  - ❑ El PS esta implementado en silicio (hard core)
- ❑ PL – Programmable Logic:
  - ❑ Bloques CLB
  - ❑ Block RAM
  - ❑ Multiplicadores por hardware
  - ❑ Bloques IOB de interconexión
  - ❑ Interface JTAG

# Processor System

El bloque PS tiene los siguientes elementos:

- ❑ Application Processor Unit (APU) con procesadores ARM
- ❑ I/O Peripherals (IOP)
  - ❑ Multiplexed I/O (MIO): Usb, Ethernet, GPIO, etc
  - ❑ Extended Multiplexed I/O /EMIO): Conecta el bloque de lógica programable a las interfaces MIO
- ❑ Controlador de memoria DDR
- ❑ Controlador DMA
- ❑ Temporizadores
- ❑ Controlador de Interrupciones
- ❑ Memoria (OnChip Memory – OCM)
- ❑ Inteface de Depuración SWO/JTAG

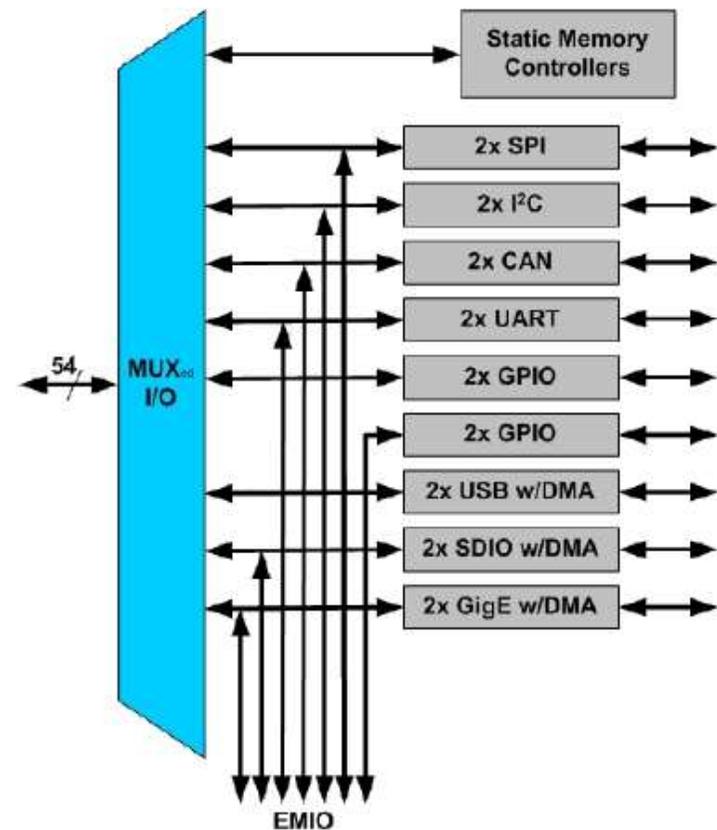


# Periféricos del PS

Hay 52 pines de I/O (MIO) que pueden mapearse

a:

- Dos USB 2.0 OTG/Device/Host
  - Dos GigE (10/100/1000)
  - Dos SDIO
  - Dos CAN
  - Dos UART
  - Dos SPI
  - Dos I2C
  - Cuatro bloques de 32 bits GPIO
- Los periféricos pueden accederse desde PL a través de la interface EMIO



# Contenido

- ❑ Componentes de un Sistema Embebido
- ❑ **Herramienta Vivado**
- ❑ Ciclo de Desarrollo de un Sistema Embebido
- ❑ Creación de la Plataforma de Hardware
- ❑ Creación de la Plataforma de Software

# Vivado

---

- ❑ El entorno Vivado es la herramienta para centralizar el proyecto
  - ❑ Permite desarrollar hardware específico para el sistema en el PL
  - ❑ Permite instanciar los bloques hardcore disponibles en el PS
  
- ❑ La herramienta IP Integrator (IPI) se utiliza para el diseño a nivel de sistema del hardware
  
- ❑ La herramienta SDK es un entorno basado en eclipse para desarrollar el software del sistema

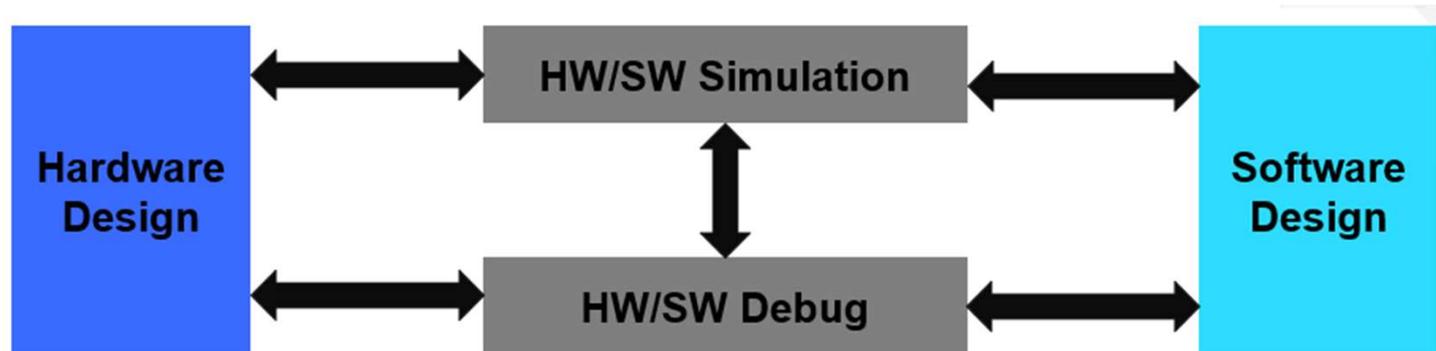
# Vivado

## Vivado / IP Integrator

- ❑ Entorno para configurar el PS y diseñar hardware en el PL
- ❑ Gestiona la descripción del hardware
- ❑ Se realizan las simulaciones de hardware

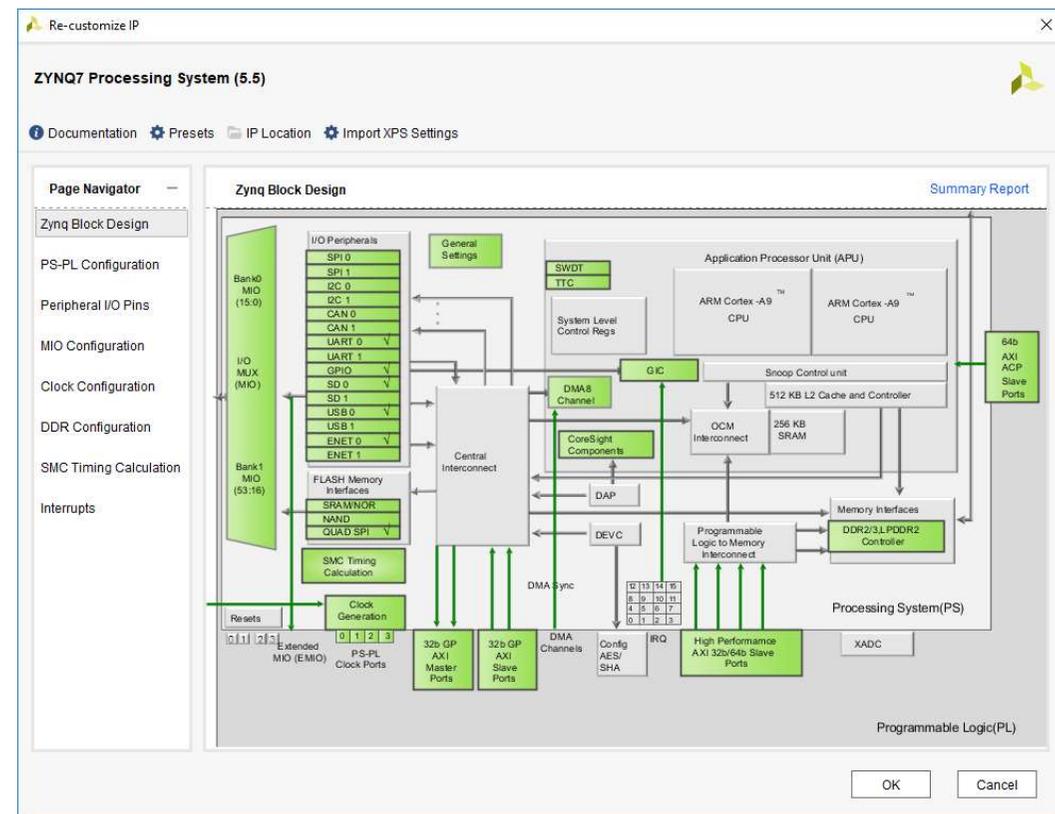
## SDK

- ❑ Entorno para gestionar el software
- ❑ Gestiona la descripción del sistema (BSP) a partir de la plataforma de hardware
- ❑ Genera la aplicación de software
- ❑ Realiza la depuración del software

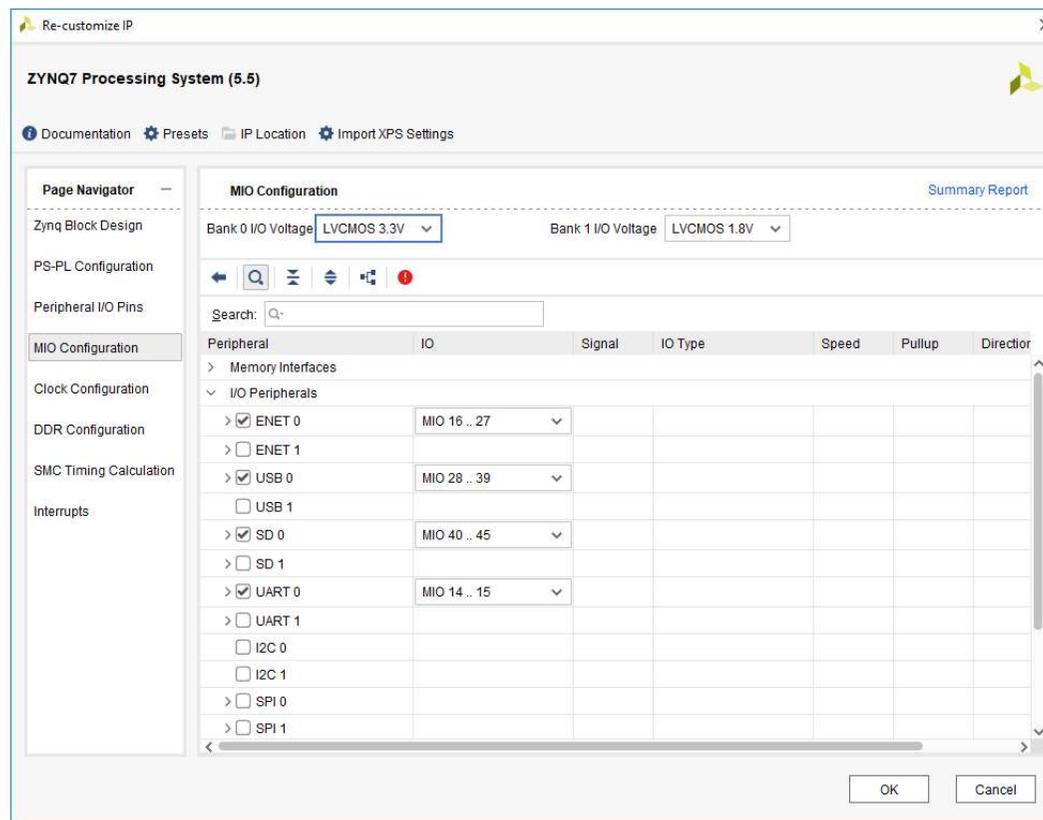


# Configuración del PS – Procesador

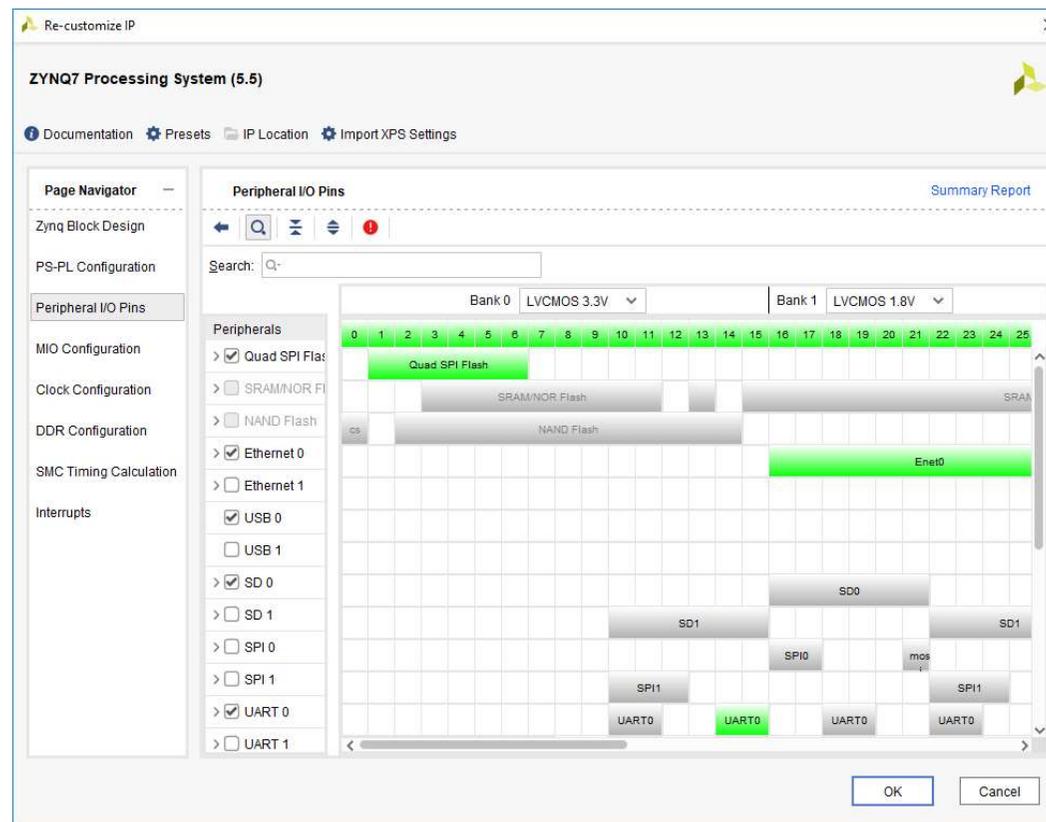
- ❑ Conexiones del Procesador
- ❑ Interface PS-PL
- ❑ Pines de I/O de los periféricos
- ❑ Configuración del ruteo de periféricos
- ❑ Configuración del reloj
- ❑ Configuración del controlador DDR
- ❑ Interrupciones



# Configuración del PS – Pines de I/O

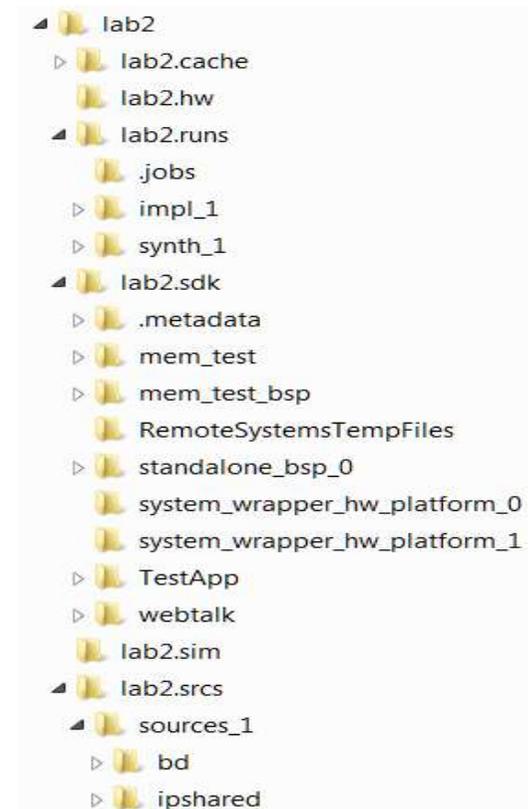


# Configuración del PS – Periféricos



# Archivos del proyecto

- ❑ Raiz: Proyecto (xpr), archivos de log
- ❑ .SRCS: código fuente en HDL, archivos generados por IPI
- ❑ .SIM: archivos de simulación de hardware
- ❑ .RUNS: archivos de implementación
- ❑ **.SDK: Archivos del software**
- ❑ .CACHE: archivos temporales



# Contenido

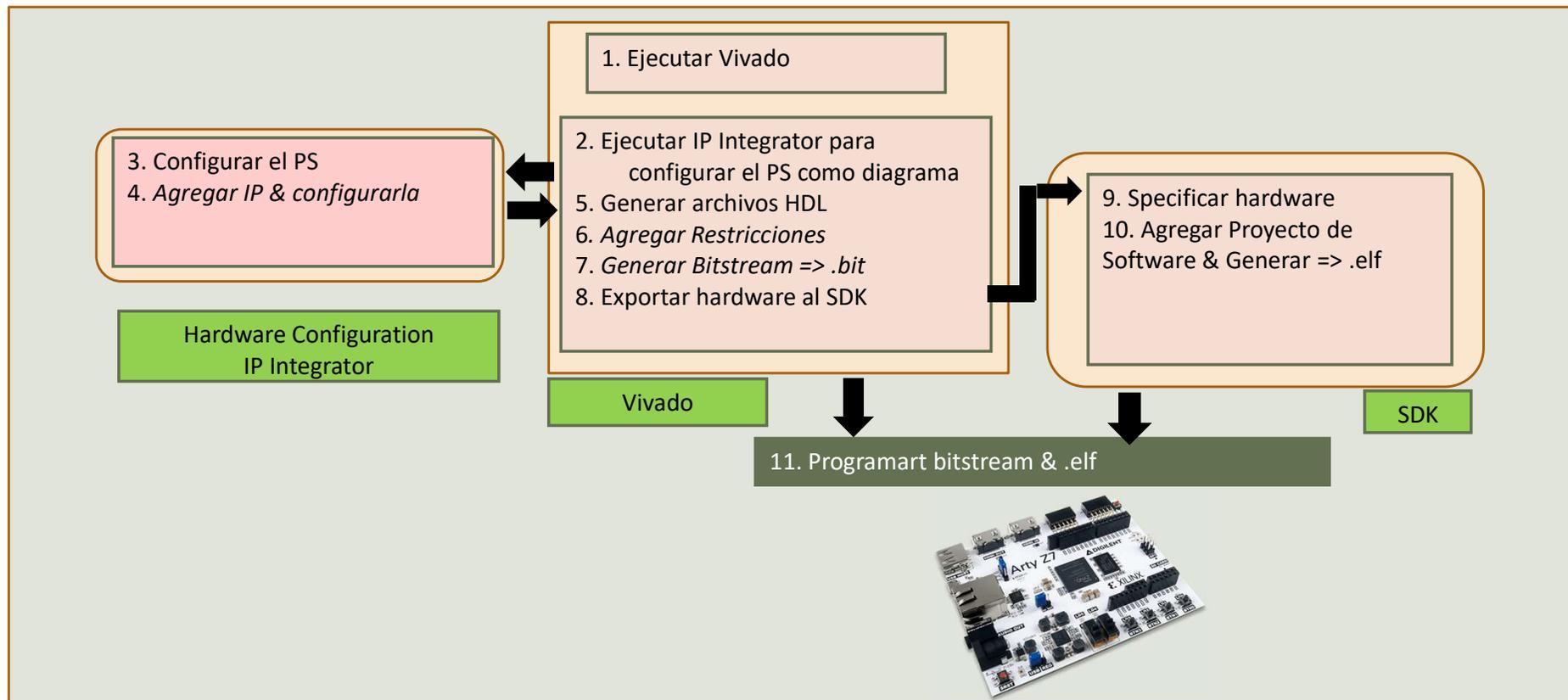
- ❑ Componentes de un Sistema Embebido
- ❑ Herramienta Vivado
- ❑ **Ciclo de Desarrollo de un Sistema Embebido**
- ❑ Creación de la Plataforma de Hardware
- ❑ Creación de la Plataforma de Software

# Ciclo de Desarrollo

---

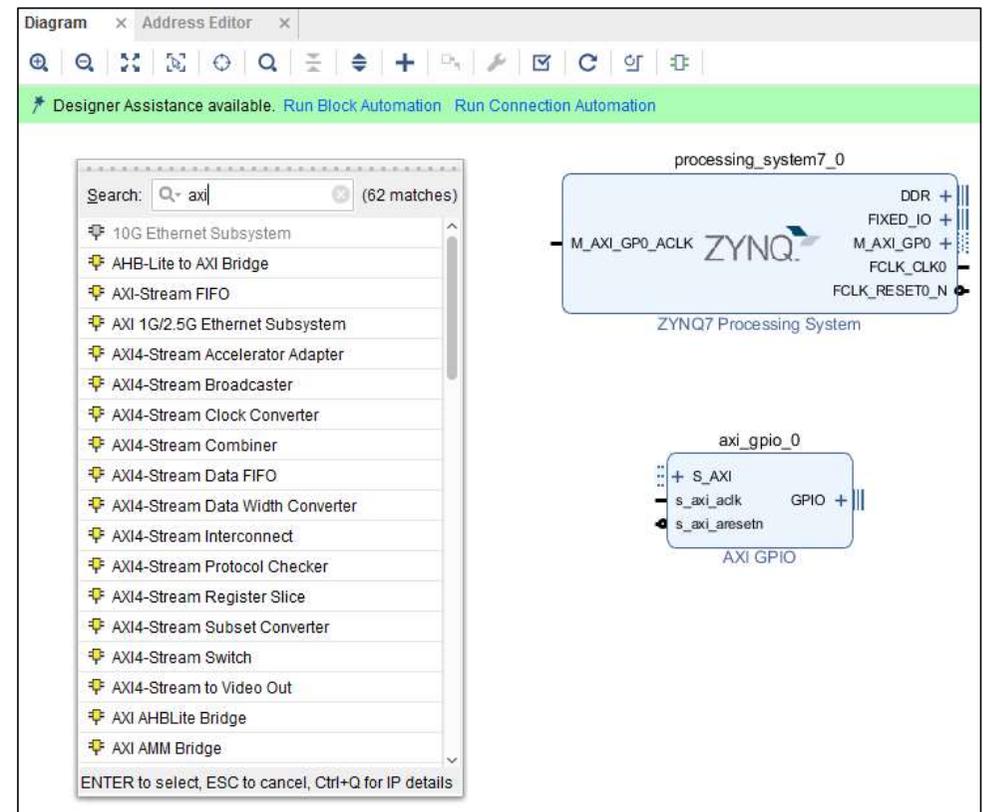
- Crear un proyecto en Vivado
- Lanzar la herramienta IP Integrator
- Diseñar la parte de hardware del PS
- Generar los archivos HDL de la descripción de hardware del PS
- Diseñar los bloques de hardware adicionales
- Exportar la descripción de hardware al SDK
- Crear el paquete BSP que corresponde a la descripción de hardware
- Compilar el software (aplicación + middleware)
- Configurar la FPGA con el archivo bitstream
- Ejecutar el software en el PS (archivo .ELF)

# Ciclo de Desarrollo



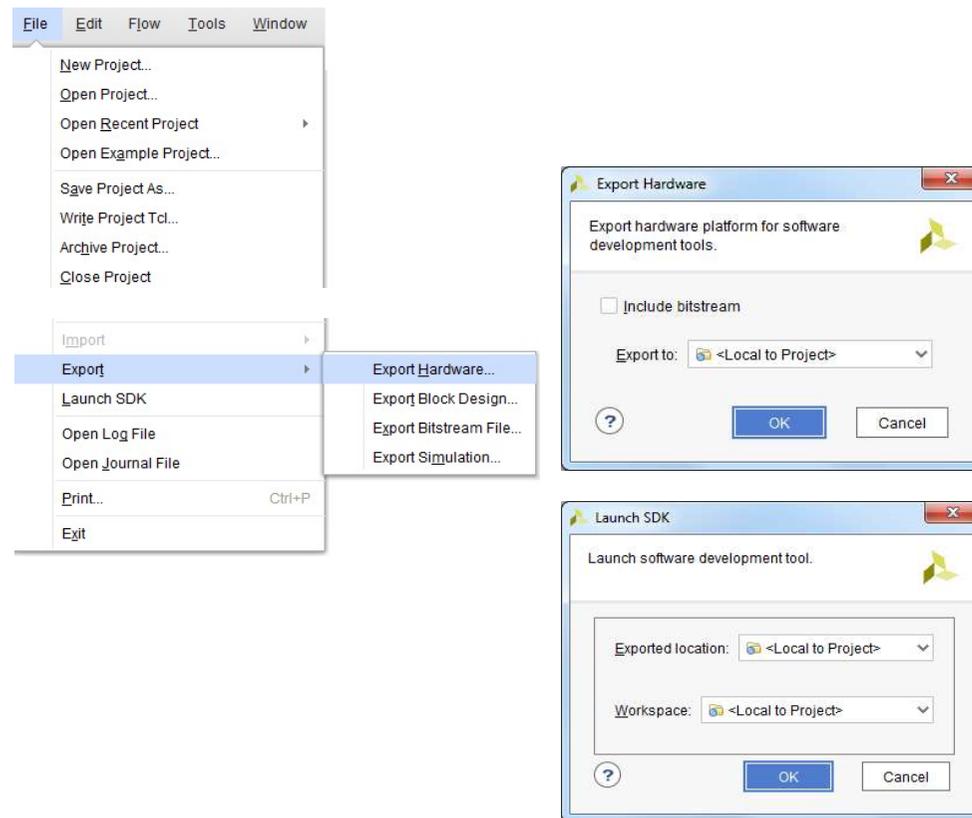
# IP Integrator

- ❑ El PS se configura a partir de un diagrama en bloques
- ❑ Se agrega la IP desde el catalogo
- ❑ Se pueden crear jerarquías de bloques
- ❑ Se puede importar IP desarrollada en Vivado con la herramienta IP Packager
- ❑ Se conectan entre si los puertos de cada bloque



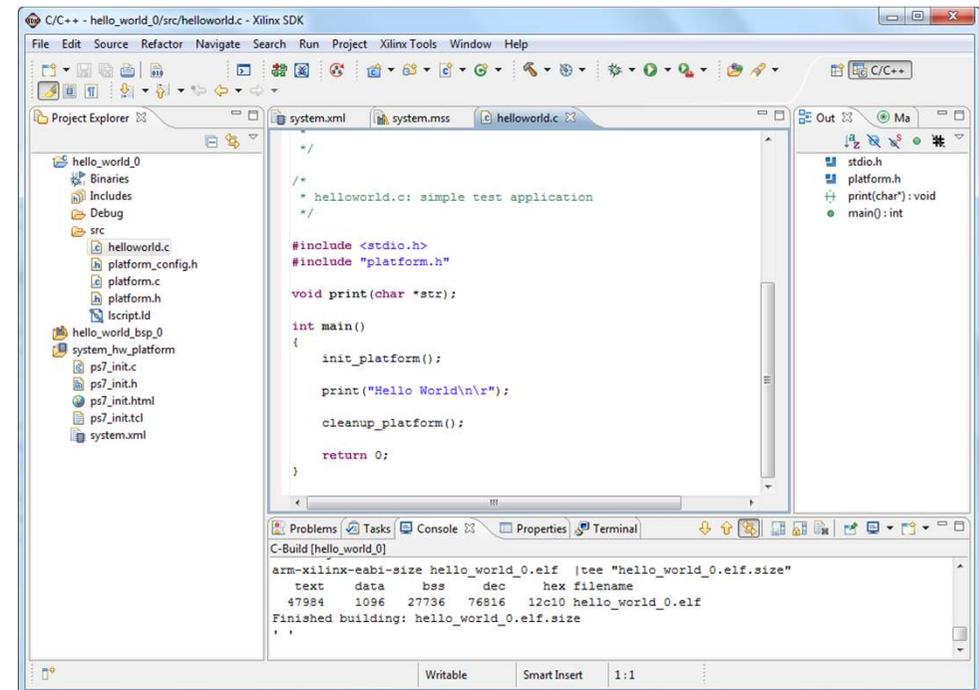
# Exportar al SDK

- ❑ Cuando se exporta el hardware:
  - ❑ Se genera y se guarda toda la información relevante en el directorio .SDK
  - ❑ Se genera el bitstream
  - ❑ La herramienta SDK asocia el proyecto de software al hardware exportado



# Herramienta SDK

- ❑ Crea el proyecto de software
- ❑ Crear el archivo BSP (con middleware y librerías)
- ❑ Crea la aplicación
- ❑ Crear el linker script
- ❑ Compila el proyecto
- ❑ Genera el archivo .ELF



```

C/C++ - hello_world_0/src/helloworld.c - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
Project Explorer
hello_world_0
  Binaries
  Includes
  Debug
  src
    helloworld.c
    platform_config.h
    platform.c
    platform.h
  IscriptId
  hello_world_bsp_0
    system_hw_platform
    ps7_init.c
    ps7_init.h
    ps7_init.html
    ps7_init.tcl
    system.xml
system.xml
system.mss
helloworld.c
Out
stdio.h
platform.h
print(char*): void
main(): int
*/
/*
 * helloworld.c: simple test application
 */
#include <stdio.h>
#include "platform.h"

void print(char *str);

int main()
{
    init_platform();

    print("Hello World\n");

    cleanup_platform();

    return 0;
}
Problems Tasks Console Properties Terminal
C-Build [hello_world_0]
arm-xilinx-eabi-size hello_world_0.elf |tee "hello_world_0.elf.size"
text data bss dec hex filename
47984 1096 27736 76816 12c10 hello_world_0.elf
Finished building: hello_world_0.elf.size

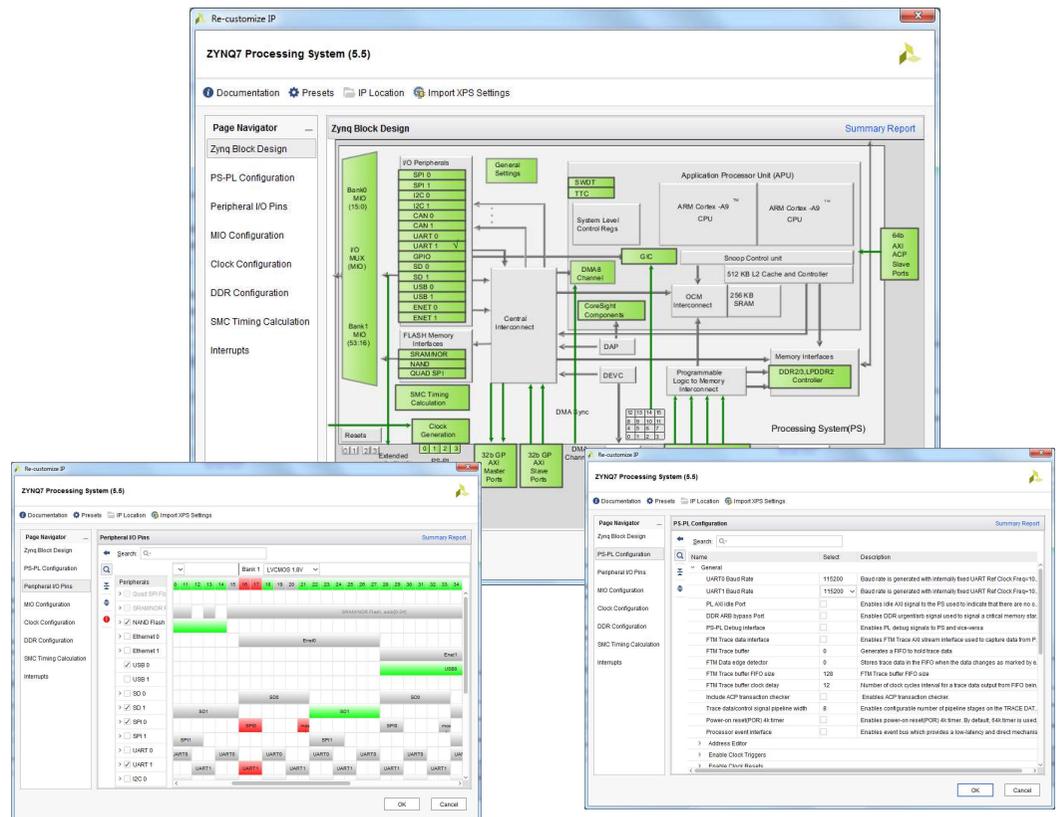
```

# Contenido

- ❑ Componentes de un Sistema Embebido
- ❑ Herramienta Vivado
- ❑ Ciclo de Desarrollo de un Sistema Embebido
- ❑ **Creación de la Plataforma de Hardware**
- ❑ Creación de la Plataforma de Software

# Configuración del PS

- ❑ La herramienta IP integrator se utiliza para la configuración de:
  - ❑ El procesador ARM
  - ❑ Los periféricos
  - ❑ El controlador de memoria DDR
  - ❑ La memoria interna (OnChip Memory)
  - ❑ La distribución de los pines de I/O entre periféricos del PS y lógica en el PL
  - ❑ Los componentes del PS se configuran mediante registros mapeados en memoria



# Configuración de la temporización

- ❑ Configuración de la frecuencia del procesador y del controlador DDR
- ❑ Configuración de las interfaces de I/O (I/O ports)
- ❑ Configuración de las señales de reloj en PL
- ❑ Timers

The screenshot displays the 'Clock Configuration' tool interface. On the left is a 'Page Navigator' with options: Zynq Block Design, PS-PL Configuration, Peripheral I/O Pins, MIO Configuration, Clock Configuration (selected), DDR Configuration, SMC Timing Calculation, and Interrupts. The main area is titled 'Clock Configuration' and has a 'Summary Report' link. It features two tabs: 'Basic Clocking' (active) and 'Advanced Clocking'. At the top, 'Input Frequency (MHz)' is set to 50.000000 and 'CPU Clock Ratio' is set to 6:2:1. Below is a search bar and a table of clock configurations.

Component	Clock Source	Requested Frequ...	Actual Frequency(...)	Range(MHz)
Processor/Memory Clocks				
CPU	ARM PLL	650	650.000000	50.0 : 667.0
DDR	DDR PLL	525	525.000000	200.000000 : 534.000...
IO Peripheral Clocks				
SMC	IO PLL	100	100.000000	10.000000 : 100.000000
QSPI	IO PLL	200	10.000000	10.000000 : 200.000000
ENET0	IO PLL	1000 Mbps	10.000000	
ENET1	IO PLL	1000 Mbps	10.000000	
SDIO	IO PLL	50	50.000000	10.000000 : 125.000000
SPI	IO PLL	166.666666	166.666672	0.000000 : 200.000000
> CAN				
PL Fabric Clocks				
<input checked="" type="checkbox"/> FCLK_CLK0	IO PLL	100	100.000000	0.100000 : 250.000000
<input type="checkbox"/> FCLK_CLK1	IO PLL	50	10.000000	0.100000 : 250.000000

# Contenido

- ❑ Componentes de un Sistema Embebido
- ❑ Herramienta Vivado
- ❑ Ciclo de Desarrollo de un Sistema Embebido
- ❑ Creación de la Plataforma de Hardware
- ❑ **Creación de la Plataforma de Software**

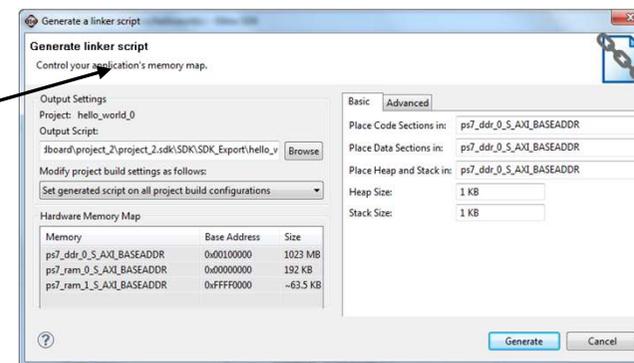
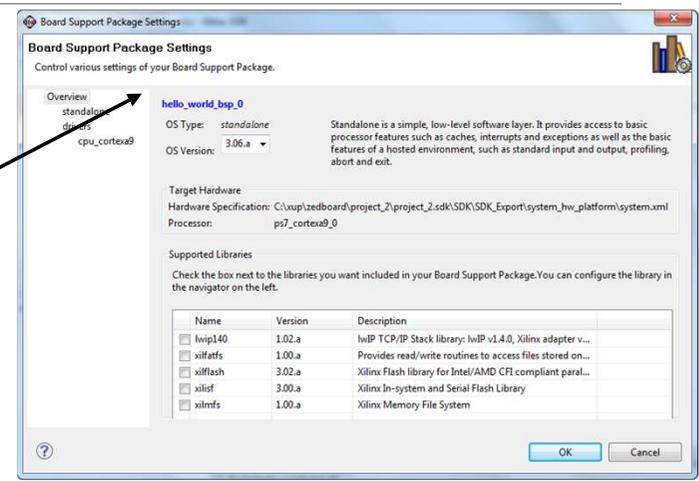
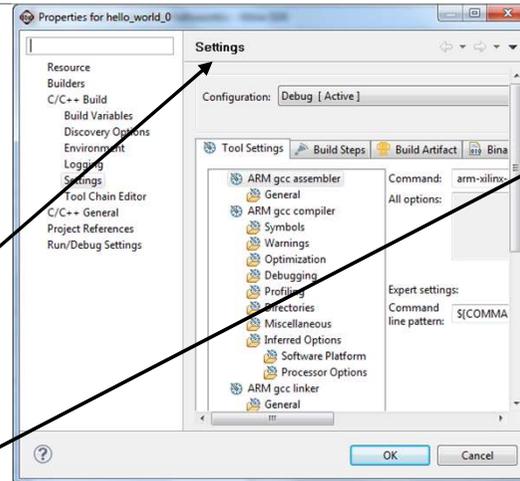
# Características del SDK

---

- ❑ Entorno de desarrollo de software basado en Eclipse.
- ❑ Esta separado de Vivado, puede ser instalado en forma independiente para equipos de software.
- ❑ Solo se utiliza para el software, cualquier modificación al hardware debe hacerse en Vivado y reexportar.
- ❑ Permite hacer depuración del software desde el entorno
- ❑ El mismo entorno puede trabajar con distintas aplicaciones de un mismo o de distinto tipo de procesador.
- ❑ Se utiliza C y C++ como lenguajes de programación.

# Generación del software

- ❑ El software se gestiona desde tres aspectos:
- ❑ **Compilado:** contiene las opciones que afectan a la generación del código ejecutable de la aplicación
- ❑ **BSP:** contiene las opciones del middleware asociado al hardware importado.
- ❑ **Linker script:** contiene las opciones de gestión de la memoria y enlazado de bibliotecas de la aplicación.



# Práctica 1

- ❑ En esta práctica se implementará el hardware de un sistema básico utilizando las herramientas Vivado y IP Integrator.
- ❑ Se utilizará la herramienta SDK para crear una aplicación de software a partir de una plantilla.
- ❑ La aplicación de software se ejecutará en el hardware para verificar su funcionalidad.

