
Creando y Agregando IP Custom

Temario

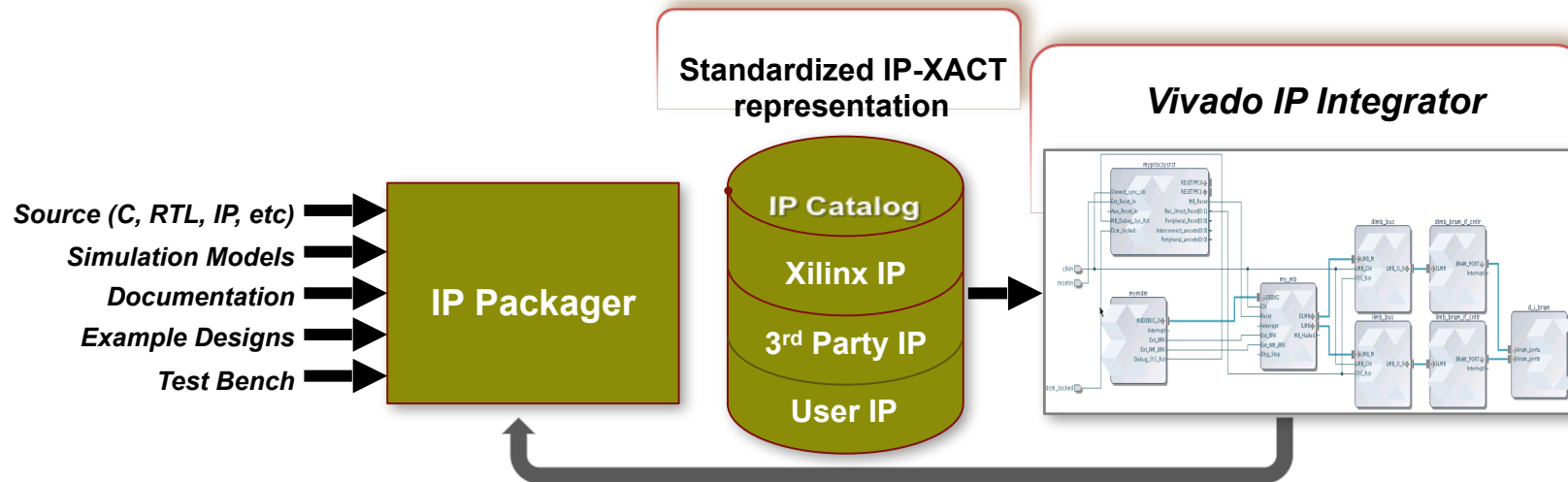
➤ ***Crear y Empaquetar IP***

- Crear IP
- Empaquetar IP

➤ Resumen

Reusando tu IP

- IP de muchas fuentes pueden ser empaquetadas y disponibles en Vivado
- Todas las IP disponibles en el catálogo de IP de Vivado puede ser usada para crear diseños
- Cualquier diagrama de IP Integrator puede ser rápidamente empaquetado como una IP compleja única



Crear y Empaquetar IP

➤ El asistente Create and Package IP permite:

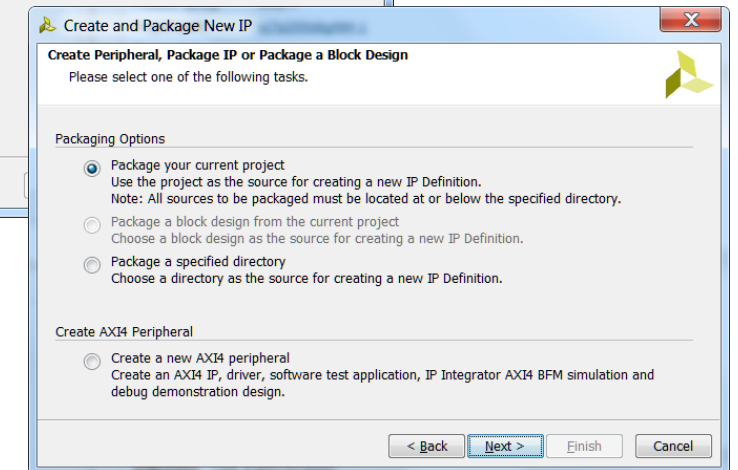
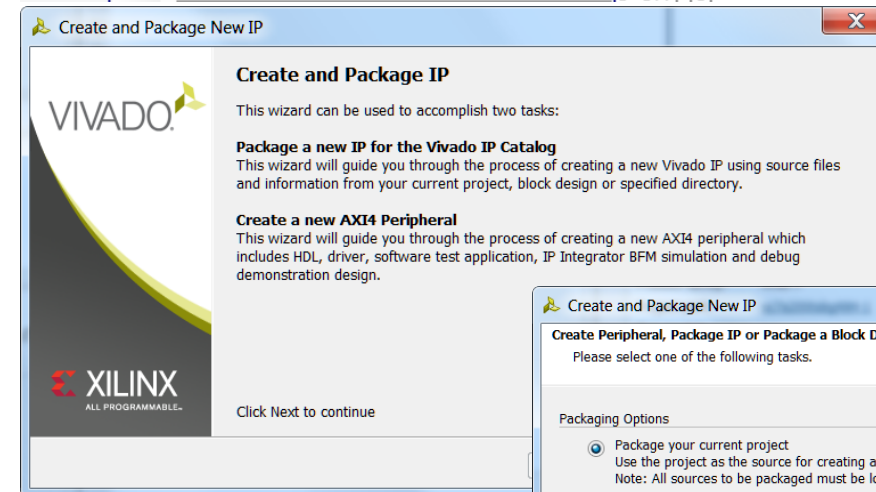
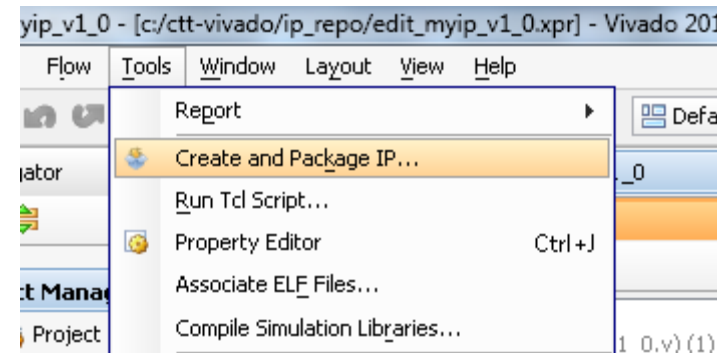
- **Crear** una plantilla para un nuevo periférico
- **Empaquetar** el proyecto actual
 - Sólo si el proyecto está abierto
- **Empaquetar** un proyecto existente

➤ El packager permite incluir la IP en el catálogo de IP para distribución

➤ Usa el formato IP-XACT

➤ Incluye un conjunto de archivos completo

- Código fuente, Restricciones, Test Benches (archivos de simulación), documentación.



Temario

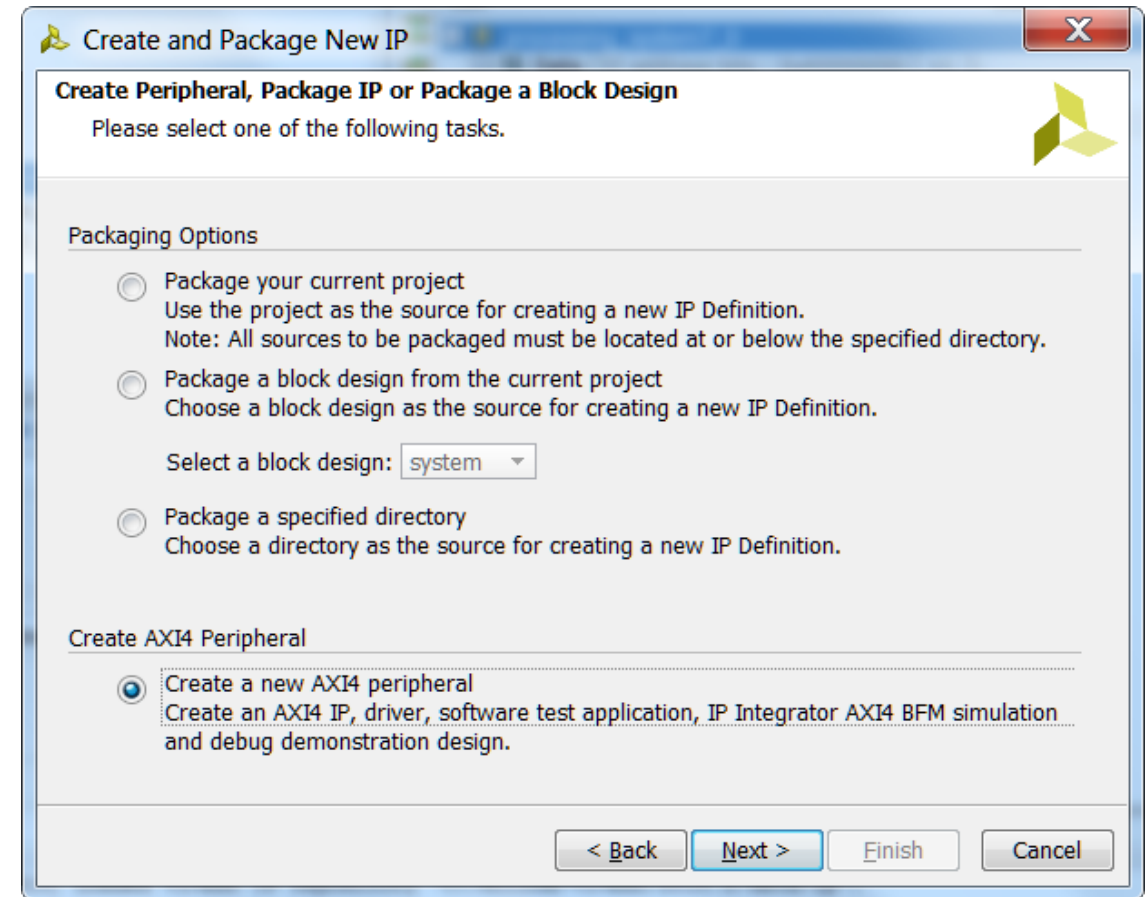
➤ ***Crear y Empaquetar IP***

- ***Crear IP***
- Empaquetar IP

➤ Resumen

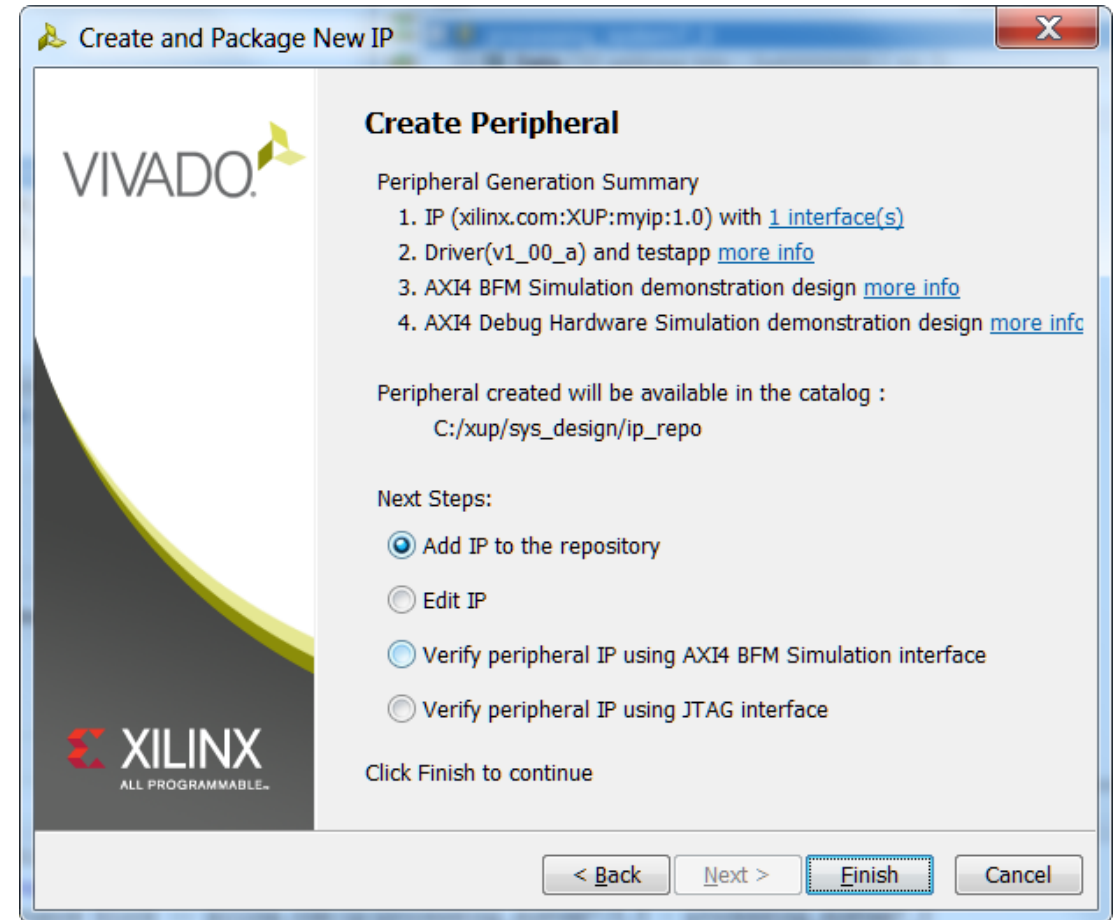
Crear IP personalizada

- **Asistente Create and Package IP**
- **Genera plantilla HDL para**
 - Slave/Master
 - AXI Lite/Full/Stream
- **Opcionalmente genera**
 - Software Driver
 - Sólo para AXI Lite e interfaz Full slave
 - Test Software Application
 - Ejemplo AXI4 BFM



Crear IP personalizada

- **Agregar IP al repositorio**
- **Editar IP**
- **Verificar**
 - BFM Simulation Example Design
 - JTAG



Generar plantilla para AXI Lite

➤ Implementación HDL de una interfaz AXI

- 32 bit de ancho de datos

➤ El usuario especifica la cantidad de registros requeridos (minimum 4)

➤ Leer/Escribir hacia/desde los registros implementados

➤ La lógica de usuario puede ser conectada fácilmente

➤ La lógica de usuario puede ser un diseño jerárquico

```
case ( axi_awaddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB] )
2'h0:
  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-1; byte_inc
    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
      // Respective byte enables are asserted as per write strobes
      // Slave register 0
      slv_reg0[(byte_index*8) +: 8] <= S_AXI_WDATA[(byte_index*8) +: 8];
    end
2'h1:
  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-1; byte_inc
    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
      // Respective byte enables are asserted as per write strobes
      // Slave register 1
      slv_reg1[(byte_index*8) +: 8] <= S_AXI_WDATA[(byte_index*8) +: 8];
    end
2'h2:
  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-1; byte_inc
    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
      // Respective byte enables are asserted as per write strobes
      // Slave register 2
      slv_reg2[(byte_index*8) +: 8] <= S_AXI_WDATA[(byte_index*8) +: 8];
    end
2'h3:
  for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-1; byte_inc
    if ( S_AXI_WSTRB[byte_index] == 1 ) begin
      // Respective byte enables are asserted as per write strobes
      // Slave register 3
      slv_reg3[(byte_index*8) +: 8] <= S_AXI_WDATA[(byte_index*8) +: 8];
    end
end
```


HDL AXI Lite

➤ Conectar la lógica de usuario a los registros, o modificar el diseño

```

if (slv_reg_wren)
begin
  case (
    axi_awaddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB]
    2'h0:
      for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-1; byte_index = byte_index+1 )
        if ( S_AXI_WSTRB[byte_index] == 1 ) begin
          // Respective byte enables are asserted as per write strobes
          // Slave register 0
          slv_reg0[(byte_index*8) +: 8] <= S_AXI_WDATA[(byte_index*8) +: 8];
        end
      2'h1:
        for ( byte_index = 0; byte_index <= (C_S_AXI_DATA_WIDTH/8)-1; byte_index = byte_index+1 )
          if ( S_AXI_WSTRB[byte_index] == 1 ) begin
            // Respective byte enables are asserted as per write strobes
            // Slave register 1
            slv_reg1[(byte_index*8) +: 8] <= S_AXI_WDATA[(byte_index*8) +: 8];
          end
    end
  end
end

```

Address

Register

Data

Plantilla Generada para AXI Full

➤ HDL AXI Full Interface

- Interfaz de datos de 32 bits

➤ Soporte para transacciones en ráfaga implementado

- Especifica el tamaño del espacio de memoria
- Hasta 1024 Bytes

➤ Código de ejemplo para implementar un bloque de memoria

- La lógica de usuario puede conectarse o reemplazarse en esta sección

```

case (S_AXI_AWBURST)
  2'b00: // fixed burst
    // The write address for all the beats in the transaction are fixed
    begin
      axi_awaddr <= axi_awaddr;
      //for awsize = 4 bytes (010)
    end
  2'b01: //incremental burst
    // The write address for all the beats in the transaction are increments by 4
    begin
      axi_awaddr[C_S_AXI_ADDR_WIDTH - 1:ADDR_LSB] <= axi_awaddr[C_S_AXI_ADDR_WIDTH - 1:ADDR_LSB];
      //awaddr aligned to 4 byte boundary
      axi_awaddr[ADDR_LSB-1:0] <= {ADDR_LSB{1'b0}};
      //for awsize = 4 bytes (010)
    end
  2'b10: //Wrapping burst
    // The write address wraps when the address reaches wrap boundary
    if (aw_wrap_en)
      begin
        axi_awaddr <= (axi_awaddr - aw_wrap_size);
      end
    else
      begin
        axi_awaddr[C_S_AXI_ADDR_WIDTH - 1:ADDR_LSB] <= axi_awaddr[C_S_AXI_ADDR_WIDTH - 1:ADDR_LSB];
        axi_awaddr[ADDR_LSB-1:0] <= {ADDR_LSB{1'b0}};
      end
    end
  default: //reserved (incremental burst for example)
    begin
      axi_awaddr <= axi_awaddr[C_S_AXI_ADDR_WIDTH - 1:ADDR_LSB] + 1;
      //for awsize = 4 bytes (010)
    end
end

```

Archivos creados

➤ component.xml

- Descripción IP XACT

➤ bd

- Archivo tcl Block Diagram

➤ drivers

- SDK y archivos de software (c code)
- Funcionalidad de Lectura/Escritura registro/memoria
- Código simple SelfTest()

➤ hdl

- Verilog/VHDL

➤ xgui

- GUI tcl file

```
XStatus LED_IP_Reg_SelfTest(void * baseaddr_p)
{
    .....

    xil_printf("*****\n\r");
    xil_printf("* User Peripheral Self Test\n\r");
    xil_printf("*****\n\n\r");

    /*
     * Write to user logic slave module register(s) and read back
     */
    xil_printf("User logic slave module test...\n\r");

    for (write_loop_index = 0 ; write_loop_index < 4; write_loop_index++)
        LED_IP_mWriteReg (baseaddr, write_loop_index*4, (write_loop_index+1)
            READ_WRITE_MUL_FACTOR);

    for (read_loop_index = 0 ; read_loop_index < 4; read_loop_index++)
        if ( LED_IP_mReadReg (baseaddr, read_loop_index*4) != (read_loop_in
            +1)*READ_WRITE_MUL_FACTOR) {
            xil_printf ("Error reading register value at address %x\n", (int)
                baseaddr + read_loop_index*4);
            return XST_FAILURE;
        }
}
```

- [illegible]

Temario

➤ Crear y Empaquetar IP

- Crear IP
- ***Empaquetar IP***

➤ Resumen

Empaquetar la IP (Package)

➤ Empaquetar el proyecto actual

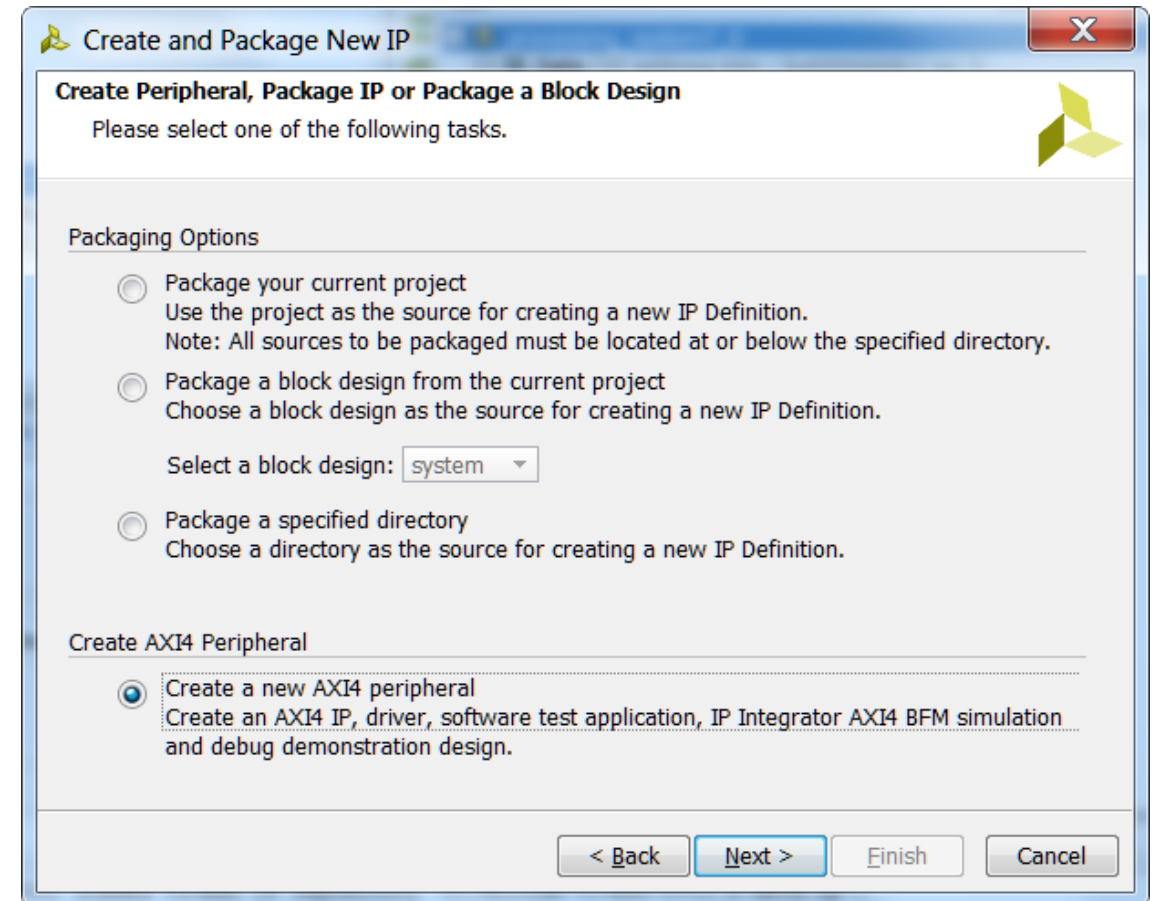
- Debe estar abierto!
- Empaquetar el HDL generado

➤ Empaquetar un directorio (otro proyecto/ archivo fuente)

- Opción para empaquetar como un library core
- El core puede ser referenciado por otra IP
- El Core no está disponible standalone (No será visible en el catálogo IP)

➤ Crear un nuevo periférico AXI4

- También necesitará ser empaquetado
- Los pasos de empaquetado son similares en los tres casos.



IP-XACT

➤ Formato XML para describir una IP usando meta-data

- Puertos
- Interfaces
- Parámetros configurables
- Archivos, documentación

➤ IP-XACT sólo describe información de alto nivel acerca de la IP, no descripciones de bajo nivel, por lo tanto no reemplaza ni el HDL ni el Software.

➤ Habilita conexión automática, configuración e integración

➤ Habilita integración de IPs de terceras partes

- (Y exportar la propia IP)



IP Packager

- **Automáticamente analiza el proyecto y los archivos para determinar los parámetros**
- **Reumen inicial**
- **Identifica**
 - Archivos
 - Fuentes HDL, Testbenches, Documentación,
 - Parametros
 - Configurable
 - Puertos
 - Interfaces
 - Compatibilidad
- **Crea GUI Layout for IPI**

IP Packager

➤ Modificar configuración

- Propiedades
- Compatibilidad
- Archivos
- Parámetros personalizados
- Puertos
- Interfaces
- Dirección y Memoria
- IP y seguridad

➤ Los proyectos pueden ser actualizados – por ej. se pueden agregar archivos fuente

- Los cambios serán reflejados en el packager

➤ Review and package

The screenshot shows the 'Package IP - axi_lite_slave' window with the 'IP Identification' tab selected. The left sidebar lists various configuration categories, with 'IP Identification' highlighted. The main area displays the 'Identification' section, which includes fields for Vendor, Library, Name, Version, Display Name, Description, Vendor Display Name, Company Url, Categories, Root Directory, and Xml File Name. The 'Categories' field is set to '/Basic_Elements'. The 'Root Directory' is 'c:/xup/embedded/labs/led_ip' and the 'Xml File Name' is 'c:/xup/embedded/labs/led_ip/component.xml'. At the bottom, there is a checkbox for 'Show advanced information'.

Project Summary x axi_lite_slave.v x Package IP - axi_lite_slave x

IP Identification

0 errors 0 warnings 0 info messages
Fill in and modify the information fields below that will be used to identify your IP. Set the categories in which your IP will appear in the IP Catalog by modifying the Taxonomy.

Identification

Vendor : xilinx.com

Library : user

Name : axi_lite_slave

Version : 1.0

Display Name : axi_lite_slave_v1_0

Description : axi_lite_slave_v1_0

Vendor Display Name :

Company Url :

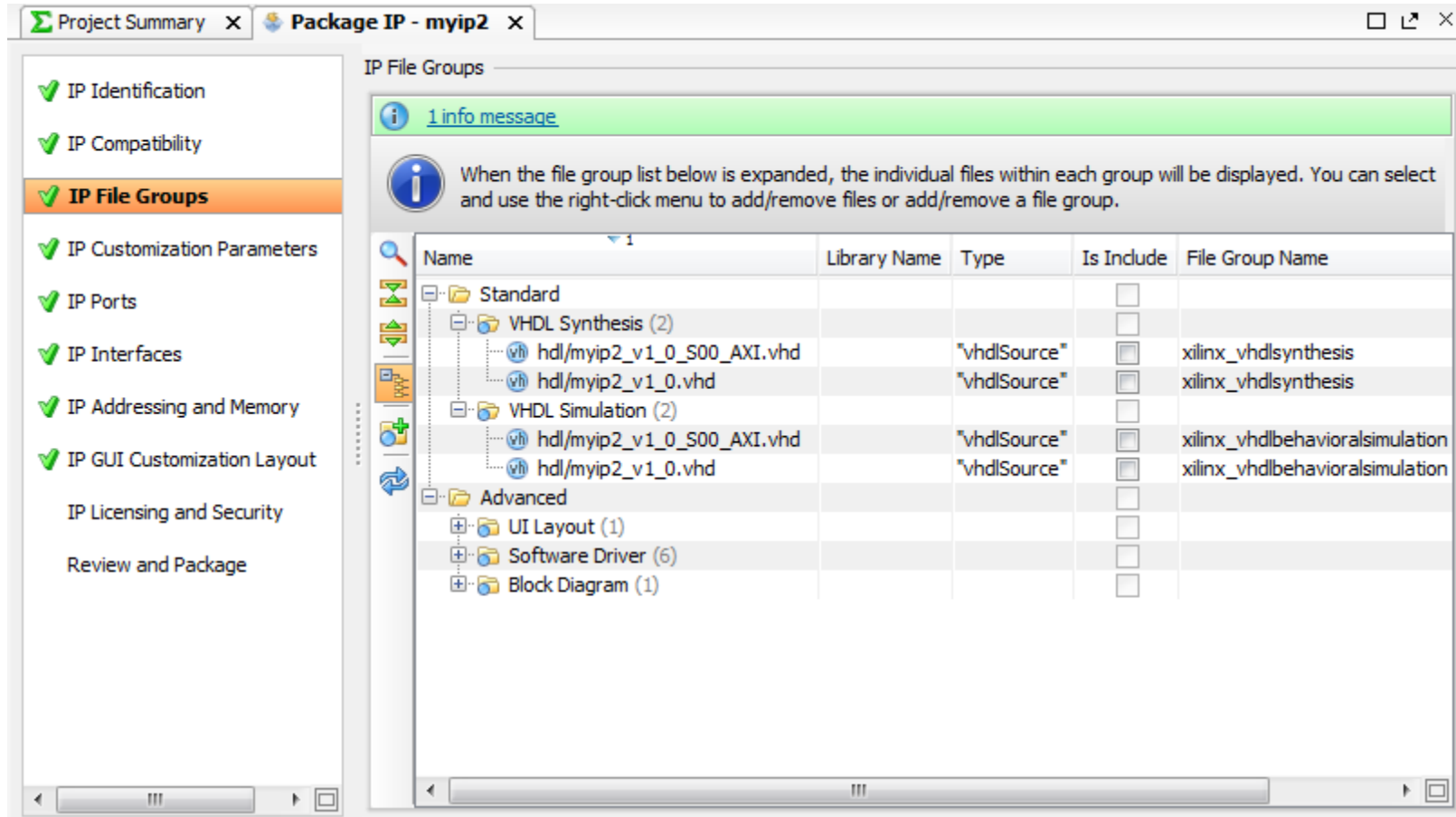
Categories : /Basic_Elements

Root Directory : c:/xup/embedded/labs/led_ip

Xml File Name : c:/xup/embedded/labs/led_ip/component.xml

☐ Show advanced information

Personalizando la IP para Reusarla en IP Packager

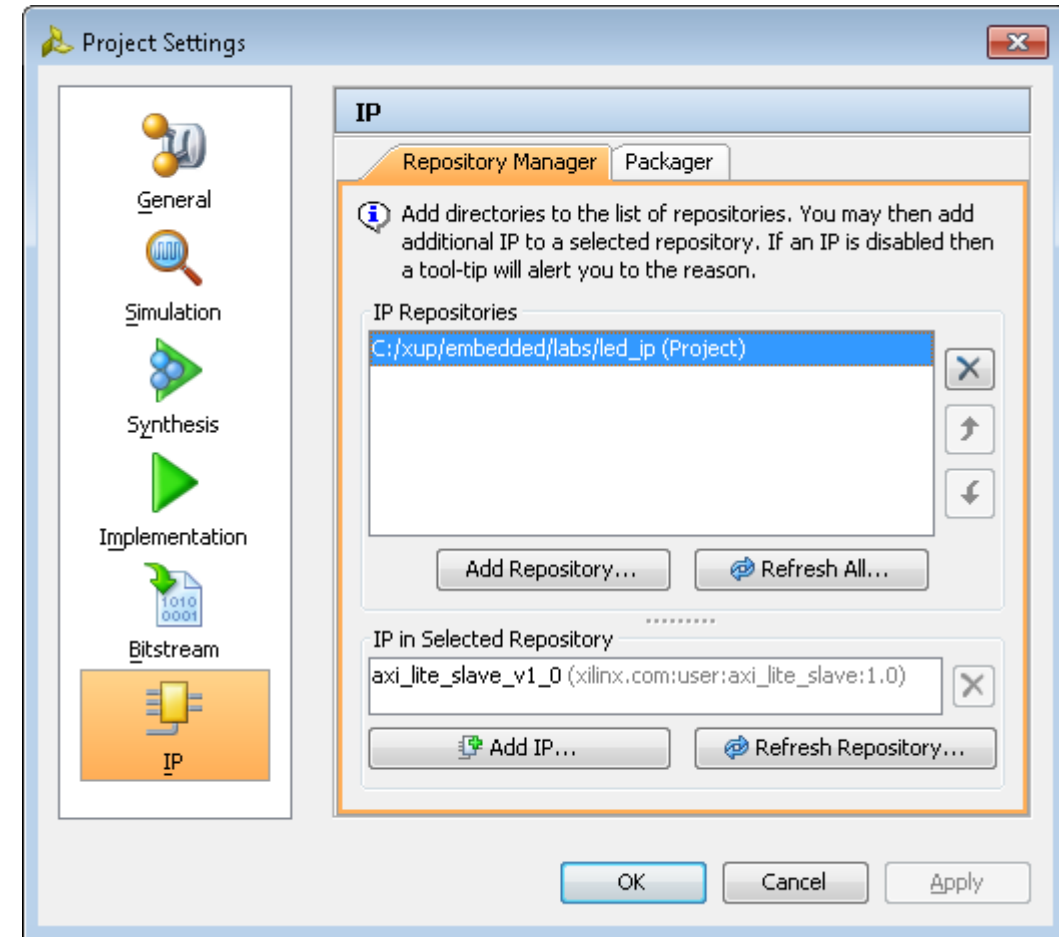


Opciones para
seleccionar

Agregar, Editar o
modificar la configuración
por defecto

Repositorio IP

- El catálogo IP de Vivado puede ser extendido agregando repositorios IP adicionales.
- Todos los IP son mostrados de la misma manera (IP de terceras partes, su propio IP, e IP de Xilinx)
- Packager crea el archivo .xml para la IP
- Especifica el directorio del repositorio de IP
 - Puede ser hecho automáticamente desde el packager
 - Acceso desde la configuración del proyecto, o desde el catálogo IP
 - Muestra la IP que ha sido encontrada
- Muestra la IP en el repositorio para uso en IPI



Temario

➤ Crear y Empaquetar IP

- Crear IP
- Empaquetar IP

➤ **Resumen**

- **Custom IP can be imported using IP Packager**
- **Puede ser incluida en el repositorio de IP para ser reusada entre proyectos**
- **El asistente Create and Package soporta la creación y el empaquetado de IP compatible con AXI Lite, Full, y Stream**
 - Maneja el protocolo en la interfaz
 - Provee plantillas para agregar funcionalidades HDL
 - Empaqueta dentro del catálogo de IP