After completing this module, you will be able to:

- Understand how to achieve greater design productivity using Vivado IP Integrator
- Understand how to rapidly create and reuse subsystem level IP with Vivado and IP Integrator
- Describe the embedded design flow in Vivado
- Explain how IP is added to extend processing system functionality
Outline

- **IP Integrator**
- Embedded System Development Flow in Vivado
  - Hardware Platform Creation
  - SDK Software Platform
- IP Catalog
- Summary

Challenges of Designing with Complex IP

- **Modern IP contains multiple complex interfaces**
  - Many signals, complex port mappings in RTL
  - Connectivity in overall design not apparent
- **Modern IP is highly parameterizable**
  - Designer must maintain consistency between interconnected IP blocks
  - Designer must understand interactions between blocks
- **Subsystems containing multiple IP blocks are often difficult to capture and reuse**
  - How to best capture IP for reuse
- **To help designers be more productive when integrating IP, Xilinx introduced Vivado IP Integrator.**
What is IP Integrator?

- A hierarchical IP integration tool for processor based and non-processor based systems
- A graphical and scriptable IP configuration and connection environment

Benefits of Vivado IP Integrator

- Interface Level Connectivity
  - Connect complex interfaces between IP in a single step
  - Users can create custom interface definitions
- Integration and Reuse of IP
  - Rapid creation of complex IP by packaging the contents of a diagram
- Automatic Generation of HDL
  - Instantiates all IP in a diagram and makes all the interconnections
- Take advantage of IP metadata
  - Propagate correct parameters to connected IP
  - Recognition of unique data types
- Processor based system support
Vivado IP Integrator User Interface

Outline

- IP Integrator
- **Embedded System Development Flow in Vivado**
  - Hardware Platform Creation
  - SDK Software Platform
- IP Catalog
- Summary
Embedded Design Architecture in Zynq

Embedded design with Zynq is based on:

- Processor and peripherals
  - Dual ARM® Cortex™-A9 processors of Zynq-7000 AP SoC
  - AXI interconnect
  - AXI component peripherals
  - Reset, clocking, debug ports
- Software platform for processing system
  - Bare Metal Applications or OS’s (e.g. Linux, FreeRTOS)
  - C language support
  - Processor services
  - C drivers for hardware
- User application
  - Interrupt service routines (optional)

The PS and the PL

The Zynq-7000 AP SoC architecture consists of two major sections

- PS: Processing system
  - Dual ARM Cortex-A9 processor based
  - Multiple peripherals
  - Hard silicon core
- PL: Programmable logic
  - Uses the same 7 series programmable logic
    - Artix™-based devices: Z-7010, Z-7015 and Z-7020 (high-range I/O banks only)
    - Kintex™-based devices: Z-7030, Z-7045, and Z-7100 (mix of high-range and high-performance I/O banks)
Embedded System Tools: Hardware

Hardware development tools
- IP Integrator
- IP Packager
- Simulation model generation
- Hardware netlist generation
- Hardware Manager to program FPGA
- Hardware debugging using Vivado analyzer

Embedded System Tools: Software

Eclipse IDE-based Xilinx Software Development Kit (XSDK)
- Board support package creation
- GNU software development tools
  - C/C++ compiler for the MicroBlaze and ARM Cortex-A9 processors (gcc)
  - Debugger for the MicroBlaze and ARM Cortex-A9 processors (gdb)
- TCF framework – multicore debug

Board support packages (BSPs)
- Stand-alone BSP
  - Free basic device drivers and utilities from Xilinx
  - NOT an RTOS
Embedded System Design using Vivado + XSDK

1. Launch Vivado
2. Invoke IP Integrator to create Block Diagram
3. Configure PS settings & configure
4. Add IP & configure

Hardware Configuration
IP Integrator

Vivado
ZedBoard

Vivado

SDK

13. Program bitstream & .elf into Zynq

11. Create software platform
12. Create and build Application project => .elf

Outline

- IP Integrator
- Embedded System Development Flow in Vivado
  - Hardware Platform Creation
  - SDK Software Platform
- IP Catalog
- Summary
Zynq Customization Processing System

- Zynq Block Design
- PS-PL Configuration
- Peripheral I/O Pins (Table View)
- MIO Configuration
- Clock Configuration
- DDR Configuration
- SMC Timing Calculation
- Interrupts

PS-PL Configuration

- GP Master enabling
- GP Slave enabling
- HP Slave enabling
- ACP Slave enabling
- PS-PL Cross Trigger  
  - For system debugging
Peripheral I/O Pins

Use MIO Configuration to select, de-select various Peripherals
Clock Configuration

Clock Configuration
- Input frequency can be set
  - Processor, DDR
- All IOP clock frequencies can be set
- PL fabric clocks can be enabled and configured
- Set Timers

Exporting to SDK

Export hardware first
- The Hardware Description File (.hdf) format file containing all the relevant information will be created and placed under the *.sdk directory
- Include bitstream if generated

Launch SDK
- Software development is performed with the Xilinx Software Development Kit tool (SDK)

The SDK tool will then associate user software projects to hardware
Outline

- IP Integrator
  - Embedded System Development Flow in Vivado
    - Hardware Platform Creation
    - SDK Software Platform
- IP Catalog
- Summary

Xilinx Software Development Kit (XSDK)

- Full-featured software design environment
- Separate tool from Vivado – can install standalone for SW teams
- Based on popular Eclipse open-source IDE
- Used for software applications only; hardware design and modifications are done in Vivado
- Well-integrated environment for seamless debugging of embedded targets
- Sophisticated software design environment with many options and features with support for
  - Multiple processors
  - Multiple software platforms
  - Multiple software applications
- Fully Featured C/C++ code editor and error navigator
SDK Workbench Views

1. C/C++ project outline displays the elements of a project with file decorators (icons) for easy identification
2. C/C++ editor for integrated software creation
3. Code outline displays elements of the software file under development with file decorators (icons) for easy identification
4. Problems, Console, Properties views list output information associated with the software development flow

Software Management Settings

- Software is managed in three major areas
  - Compiler/Linker Options
    - Application program
  - Software Platform Settings
    - Board support package
  - Linker Script Generation
    - Assigning software to memory resources
Software Development Flow

- Create hardware platform project
  - Automatically performed when SDK tool is launched from Vivado project
    - The hardware should have been exported
- Create BSP
  - System software, board support package
- Create software application
- Create linker script
- Build project
  - compile, assemble, link output file `<app_project>.elf`

Outline

- IP Integrator
- Embedded System Development Flow in Vivado
  - Hardware Platform Creation
  - SDK Software Platform
- IP Catalog
- Summary
Vivado IP Catalog

- The IP Catalog contains a collection of IP that can be used to assemble a system.
- Supported by IPI
- Each IP block has its own configuration parameters.
- Most of the IP are free, some require licenses.
- Stored as source code in the install directory.
  - Always synthesized with the latest tools.
  - Some proprietary source code is encrypted.
- IP Core search path:
  - Current project directory.
  - MyProcessorIPLib directory (user defined).
  - Repository Directory listed using:
    - Project → Project Options → Device and Repository Search tab
  - `%XILINX_INSTALL%/Vivado\2014.2\data\ip` (Tools installation directory)

IP Peripherals
Included as Source (Free)

- Bus and bridge controllers:
  - AXI to AXI connector
  - Local Memory Bus (LMB)
  - AXI Chip to Chip
  - AHB-Lite to AXI
  - AXI4-Lite to APB
  - AXI4 to AHB-Lite
- Debug cores:
  - Integrated Logic Analyzer
- DMA and Timers:
  - Watchdog, fixed interval
- Inter-processor communication
- External peripheral controller Memory and memory controller
- High-speed and low-speed communication peripherals:
  - AXI 10/100 Ethernet MAC controller
  - Hard-core tri-mode Ethernet MAC
  - AXI IIC
  - AXI SPI
  - AXI UART
- Other cores:
  - System monitor
  - Xilinx Analog-to-Digital Converter (XADC)
  - Clock generator
  - System reset module
  - Interrupt controller
Communicating with PL

- **Processing system master**
  - Two ports from the processing system to programmable logic
  - Connects the CPU block to common peripherals through the central interconnect

- **Processing system slave**
  - Two ports from programmable logic to the processing system

- **Slave PL peripherals address range**
  - 4000_0000 and 7FFF_FFFF (connected to GP0) and
  - 8000_0000 and BFFF_FFFF (connected to GP1)

Add IP in the PL

- **Configure GP ports from PS Customization GUI**
- **PS-PL Configuration**
  - E.g. Enable M_AXI_GP0/1 or S_AXI_GP0/1
- **Ports will then be available on Zynq Block Diagram**
- **Connect the added IP to the appropriate port**
- **Assign address to the added IP, if unmapped**
- **Configure the IP, if necessary**
- **Make external connections, if needed**
  - Add external ports/interfaces if the added IP is interacting with external devices
Connecting IP

- Add IP from the IP Catalog
- Click and drag to find connections
- Valid connections Highlighted

- Designer Assistance, Connection automation
  - If Board Support available, IP can be connected to external pins
  - Or manually create and connect (external) ports

Designer Assistance: Block Automation, Connection Automation

- Block Automation
  - PS configuration
- Connection Automation
  - Can automatically connect IP blocks
  - Automatically insert required blocks
  - E.g. Add BRAM; Automation will insert and connect BRAM controller and Reset logic
- If Board Support is available, IP can automatically be connected to top level ports
**Parameterize IP Instances**

- Double-click or right click the instance and select **Customize Block** to open the configurable parameters dialog box (refer to the datasheet if needed).
- Default values are shown.
  - Customize the parameters that you want.

**Assign Addresses**

- Peripherals in the Zynq™ AP SoC PS have fixed addresses and do not appear in the address map.
- For PS peripherals added and connected manually the addresses will appear as unmapped.
- Click on the **Auto Assign Addresses** button.

The address will be generated and appear as the generated addresses of the added IP.
Bitstream Generation

- After defining the system hardware, the next step is to create hardware netlists if the system hardware has logic in PL
- Generate the output products
  - IP level files are generated
- A HDL wrapper for the block diagram must be generated
  - Additional logic can be added to the HDL, or the processor system can be used as a sub-block in a HDL design
- If the system contains hardware in the PL, the bitstream must be generated
  - The block diagram must be open before synthesise and implementation can be carried out
- The PL (FPGA) must be programmed before application can be downloaded and executed

Outline

- IP Integrator
- Embedded System Development Flow in Vivado
  - Hardware Platform Creation
  - SDK Software Platform
- IP Catalog
- Summary
Summary

- Vivado is an IP Centric design environment, allowing multiple ways to add/manage IP
  - Automatic RTL generation
  - Rapid assembly, packaging, and reuse of IP subsystems
  - Processor based system design support

- IPI is a System Level design tool that increases productivity, allowing designs to be completed faster

- The PS Configuration wizard permits access to several configurable features of PS

- The Xilinx Software Development Kit (XSDK) is a comprehensive software development environment for software applications

PS functionality can be extended by instantiating peripherals in PL

Adding IP in PL involves
- Enabling interface(s) in PS
- Selecting IP from the IP Catalog and configuring IP for desired functionality
- Connecting the (PL) IP to the PS using IP Integrator
- Assigning address
- Connecting IP ports to ports of other peripherals and/or to external pins

HDL Wrapper is needed for IP Integrator Block

Bitstream must be generated when PL has any IP

The FPGA must be programmed with the generated hardware bitstream before an application can be run