Zynq Architecture -- Extend Embedded System into PL with Vivado

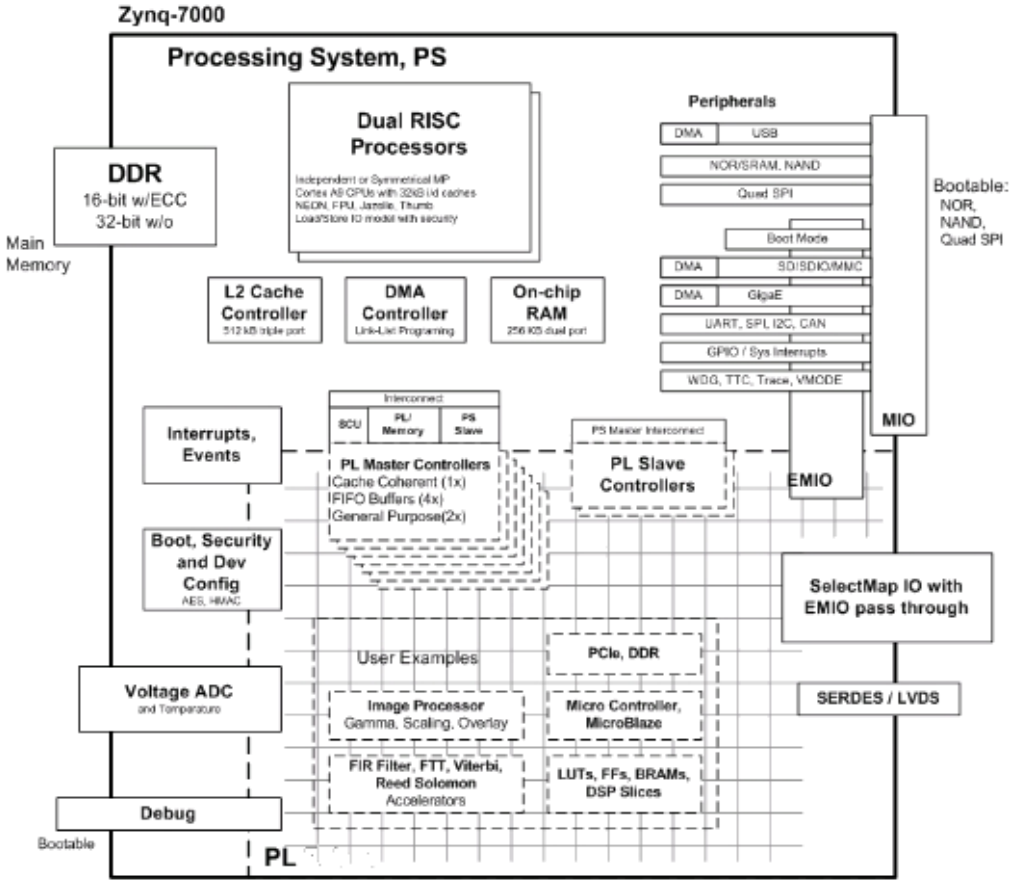
Zynq
Vivado 2015.2 Version

Outline

- PS-PL Interface
- PL Clocking Sources
- Zynq Resets
- Process of adding IP to PL

PS Components

- **The Zynq AP SoC processing system consists of the following blocks**
  - Application processing unit (APU)
  - I/O peripherals (IOP)
    - Multiplexed I/O (MIO), extended multiplexed I/O (EMIO)
  - Memory interfaces
  - PS interconnect
  - DMA
  - Timers
    - Public and private
  - General interrupt controller (GIC)
  - On-chip memory (OCM): ROM and RAM
  - Debug controller: CoreSight



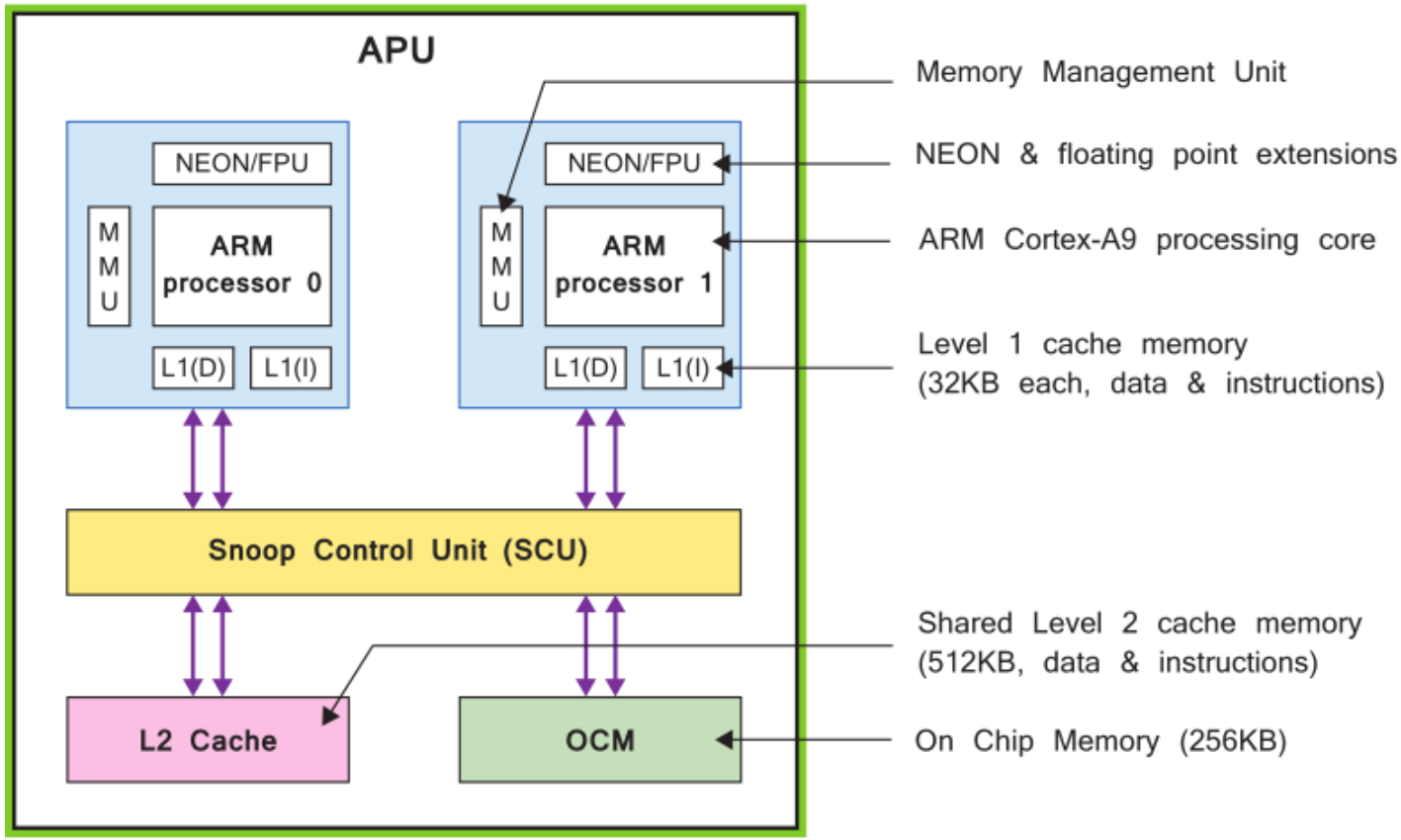
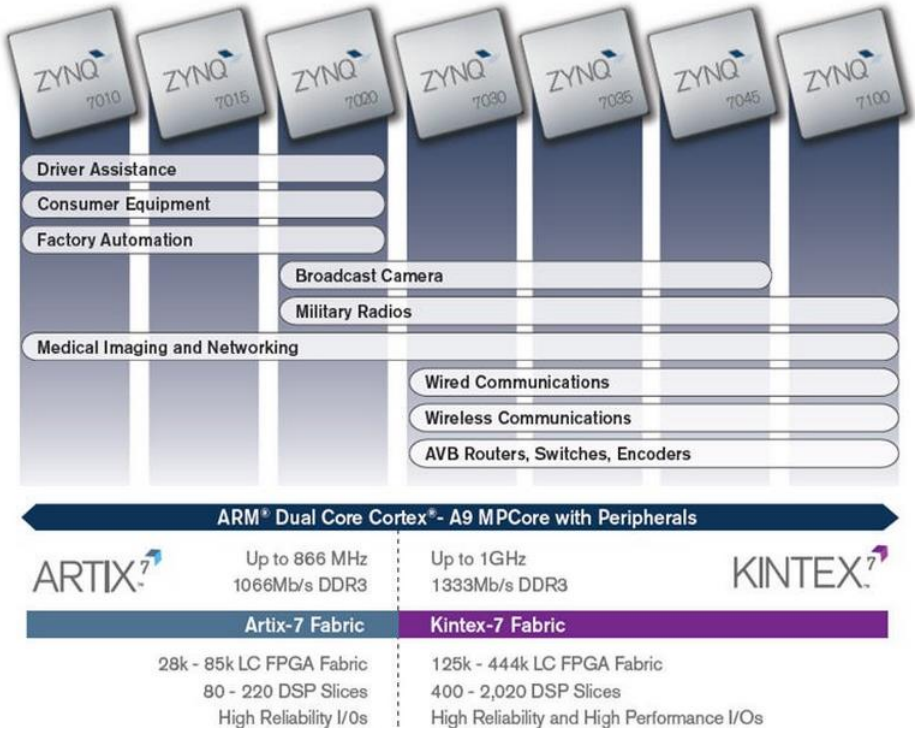


Figure 2.3: Block diagram of the application processing unit (simplified)

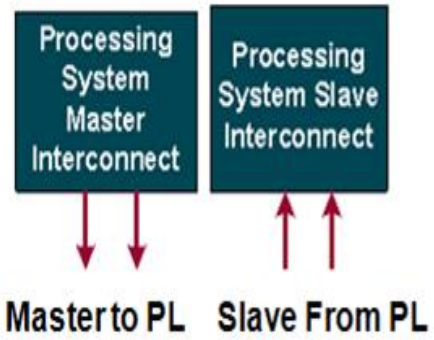
The PS and the PL

- **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, based on the following two FPGA fabrics
      - Artix™-based devices: Z-7010, Z-7015 and Z-7020 (high-range I/O banks only)
      - Kintex™-based devices: Z-7030, Z-7035, Z-7045, and Z-7100 (mix of high-range and high-performance I/O banks)



PS-PL Interfaces

- **AXI general-purpose ports (GP0-GP1)**
  - Two masters from PS to PL
  - Two slaves from PL to PS
  - 32-bit data width
  - Conversation and sync to processing system clock domain



- **Clock and resets**
  - Four PS clock outputs to the PL with enable control
  - Four PS reset outputs to the PL

PL Clocking Sources

- **PS clocks**
  - PS clock source from external package pin
  - PS has three PLLs (Phase-Lock-Loop) for clock generation
  - PS has four clock ports to PL
- **The PL has 7 series clocking resources**
  - PL has a different clock source domain compared to the PS
  - The clock to PL can be sourced from external clock capable pins
  - **Can use one of the four PS clocks as source**
- **Synchronizing the clock between PL and PS is taken care of by the architecture of the PS**
- **PL cannot supply clock source to PS**

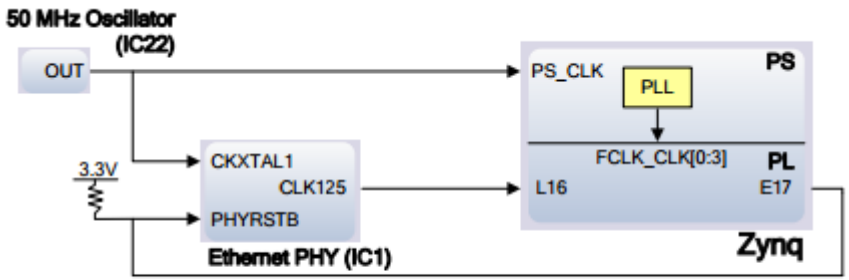


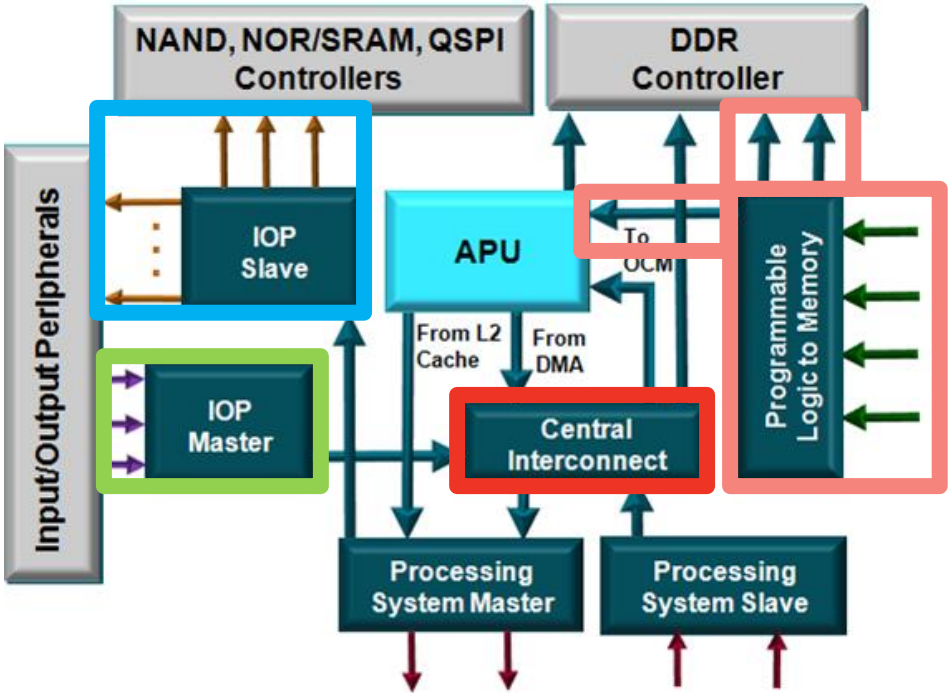
Figure 13. ZYBO clocking.

Zynq Resets

- **Internal resets**
  - Power-on reset (POR)
  - Watchdog resets from the three watchdog timers
  - Secure violation reset
- **PS resets**
  - External reset: PS_SRST_B
  - Warm reset: SRSTB
- **PL resets**
  - Four reset outputs from PS to PL
  - FCLK_RESET[3:0]

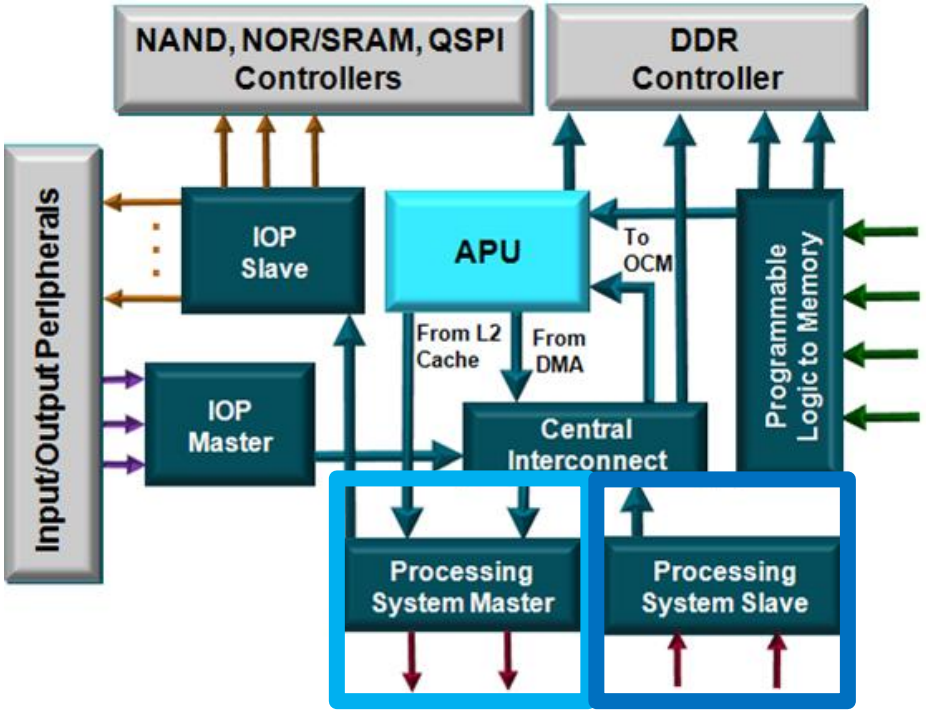
Processing System Interconnect (1)

- **Programmable logic to memory**
  - Two ports to DDR
  - One port to OCM SRAM
- **Central interconnect**
  - Enables other interconnects to communicate
- **Peripheral master**
  - USB, GigE, SDIO connects to DDR and PL via the central interconnect
- **Peripheral slave**
  - CPU, DMA, and PL access to IOP peripherals



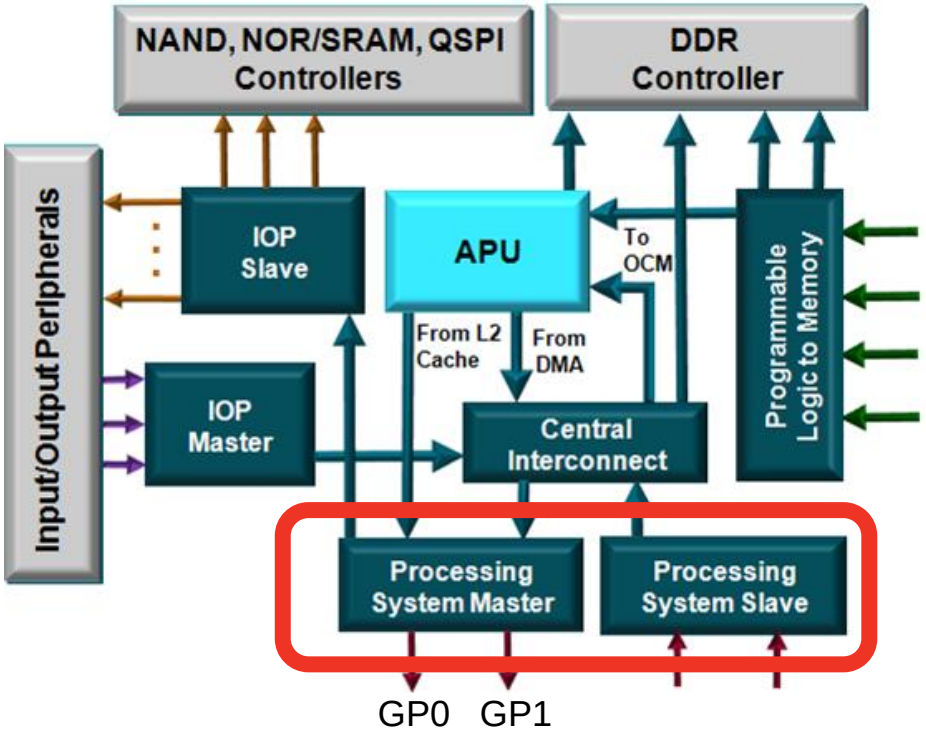
Processing System Interconnect (2)

- **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



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)

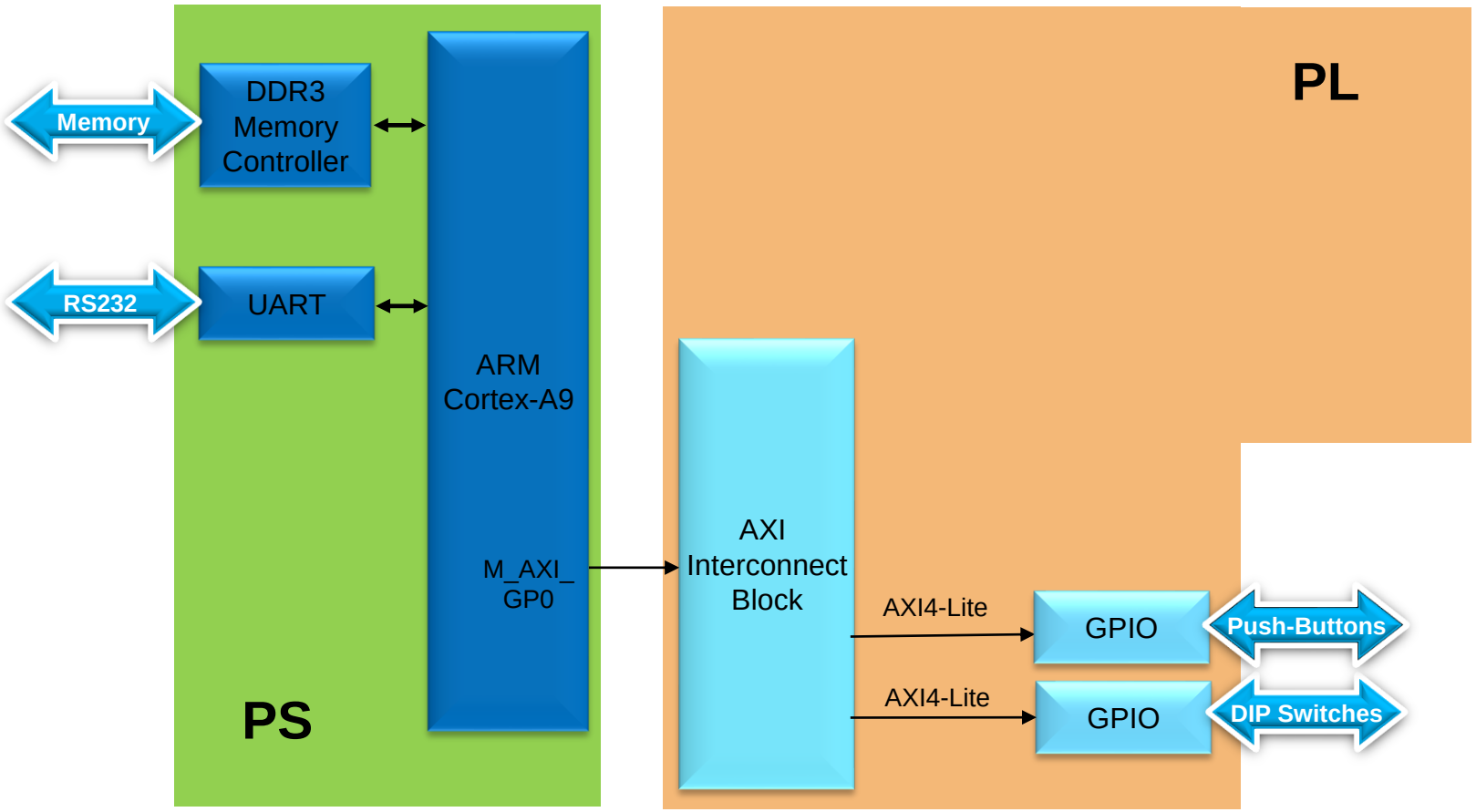


Process of Adding IP to PL

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**

ARM Cortex-A9 based Embedded System Design Lab2: Adding IPs in PL



Procedure

- Open the project in Vivado
- Add and configure GPIO peripherals in the system in the IP Integrator
- Add external ports
- Generate the bitstream and export to SDK
- Create a TestApp application in SDK
- Verify the functionality in hardware

Major Steps

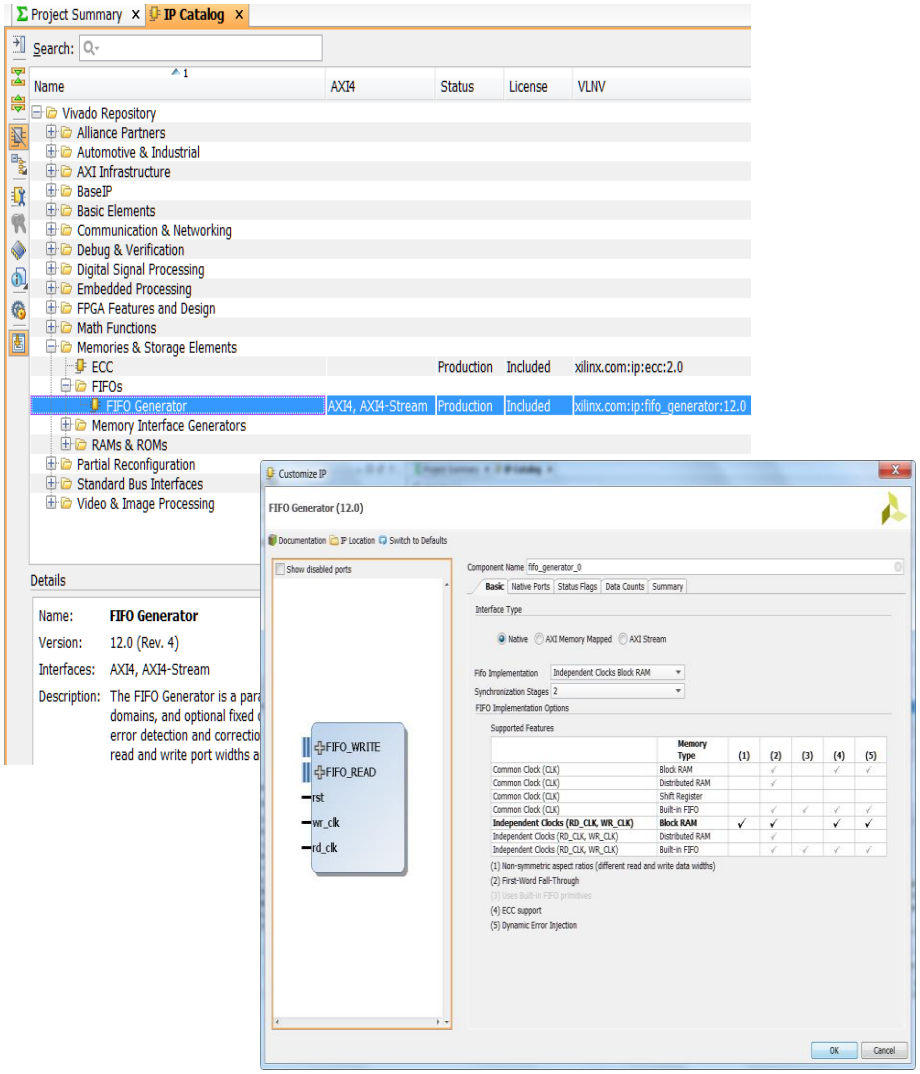
- The GP Master interface of the PS was enabled. GPIO peripherals were added from the IP catalog and connected to the Processing System through the 32b Master GP interface.
- The peripherals were configured and external FPGA connections were established. Pin location constraints were made using IP Integrator Automation, and also manually, to connect the peripherals to push buttons and DIP switches.
- A TestApp application project was created and the functionality was verified after downloading the bitstream and executing the program.

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
  - Traffic Generator, Performance monitor

Vivado IP Catalog

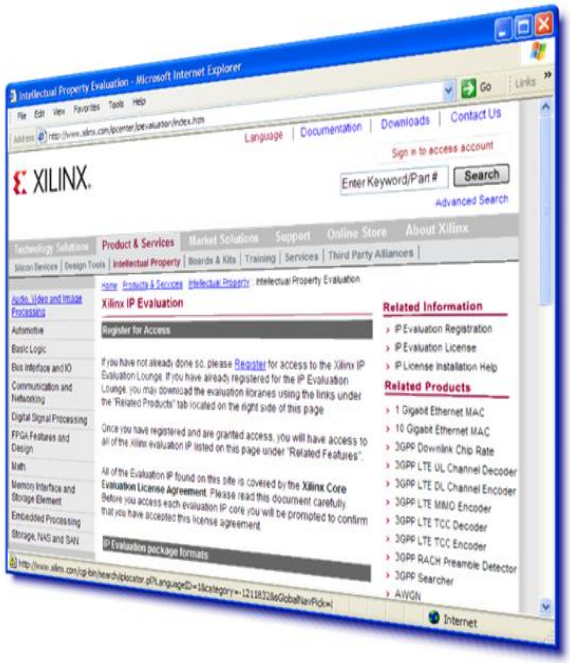
- **Integrated IP Support**
  - Instant access to IP customization
  - Vivado IP GUI look and feel
  - Support for Vivado synthesis and implementation
  - Selectable IP output products
  - Full Tcl support



IP Cores Included as Evaluation

- AXI CAN controller
- AXI USB2 device
- Video IP
- Telecoms/ Wireless IP

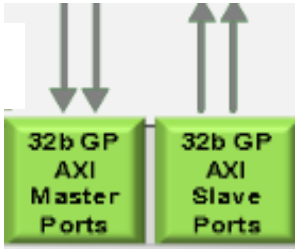
Standard Bus Interfaces				
S/PDIF	AXI4, AXI4-Stream	Production	Purchase	xilinx.com:ip:spdif:2.0
SD Card Host Controller	AXI4	Production	Purchase	logicbricks.com:logicbricks:logisdhc:0.0
RapidIO				
Serial RapidIO Gen2	AXI4, AXI4-Stream	Production	Purchase	xilinx.com:ip:srio_gen2:4.0
PCI Express				
AXI Memory Mapped To PCI Express	AXI4	Production	Included	xilinx.com:ip:axi_pcie:2.6
7 Series Integrated Block for PCI Express	AXI4-Stream	Production	Included	xilinx.com:ip:pcie_7x:3.1
DisplayPort				
DisplayPort	AXI4, AXI4-Stream	Pre-Produ...	Purchase	xilinx.com:ip:displayport:6.0
PCI				
32-bit Initiator/Target for PCI (7-Series)		Production	Purchase	xilinx.com:ip:pci32:5.0
64-bit Initiator/Target for PCI (7-Series)		Production	Purchase	xilinx.com:ip:pci64:5.0



Xilinx developed, delivered, and supported Evaluation IP installs with a 90-day evaluation license

GP Ports

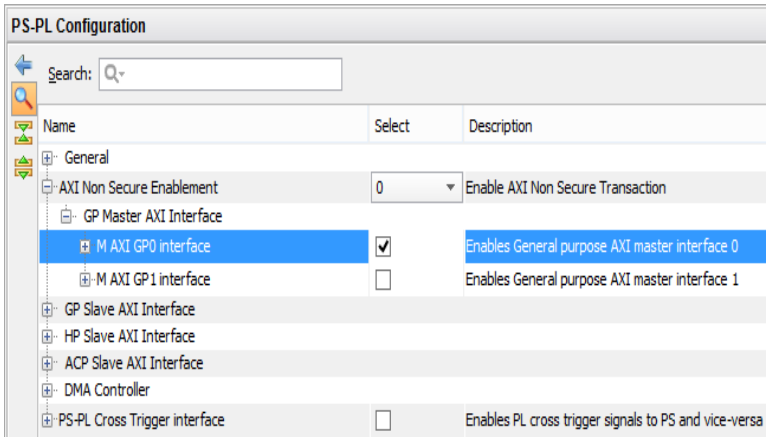
- **By default, GP Slave and Master ports are disabled**

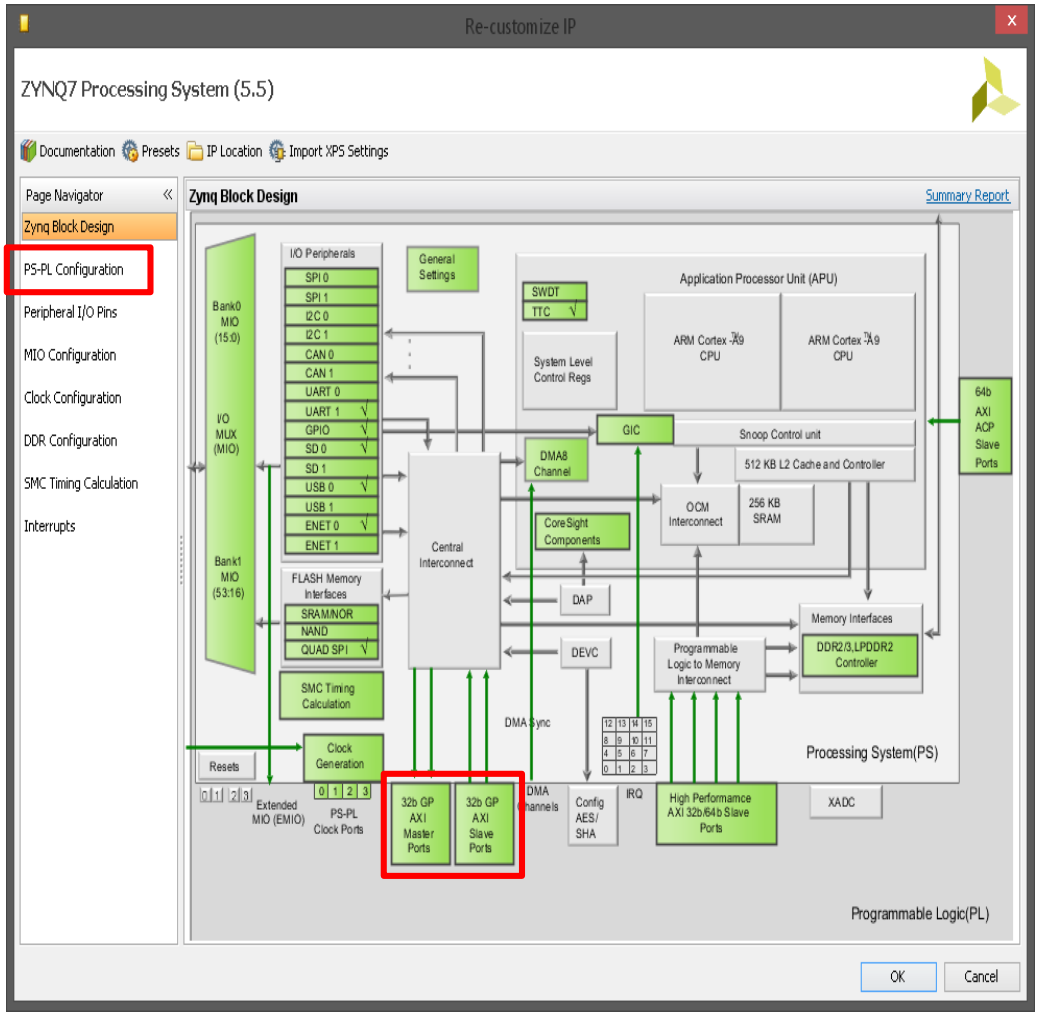


- **Enable GP Master and/or Slave ports depending on whether a slave or a master peripheral is going to be added in PL**
- **axi_interconnect block is required to connect IP to a port with different protocols**
  - Automatically convert Protocols
  - Can be automatically added when using Block Automation in IPI

Configuring GP Ports

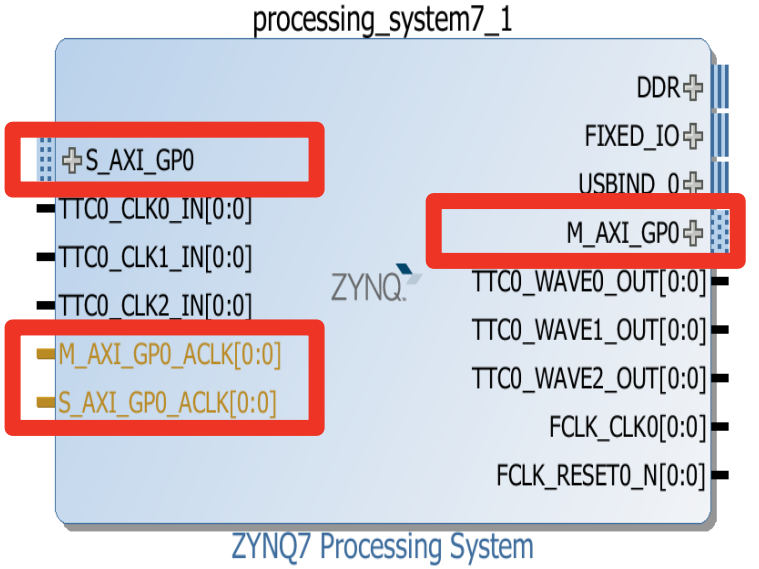
- **Click on the menu or green GP Blocks to configure**





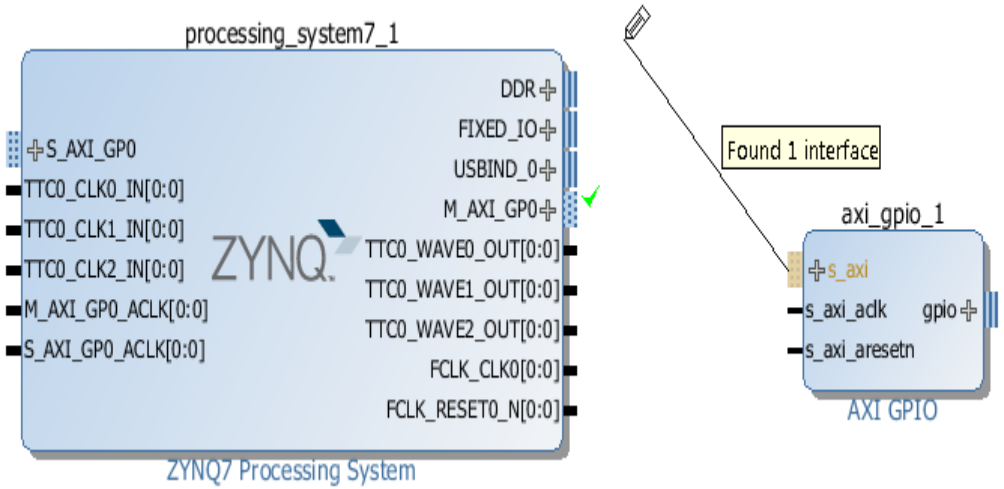
Add IP in the PL

- **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



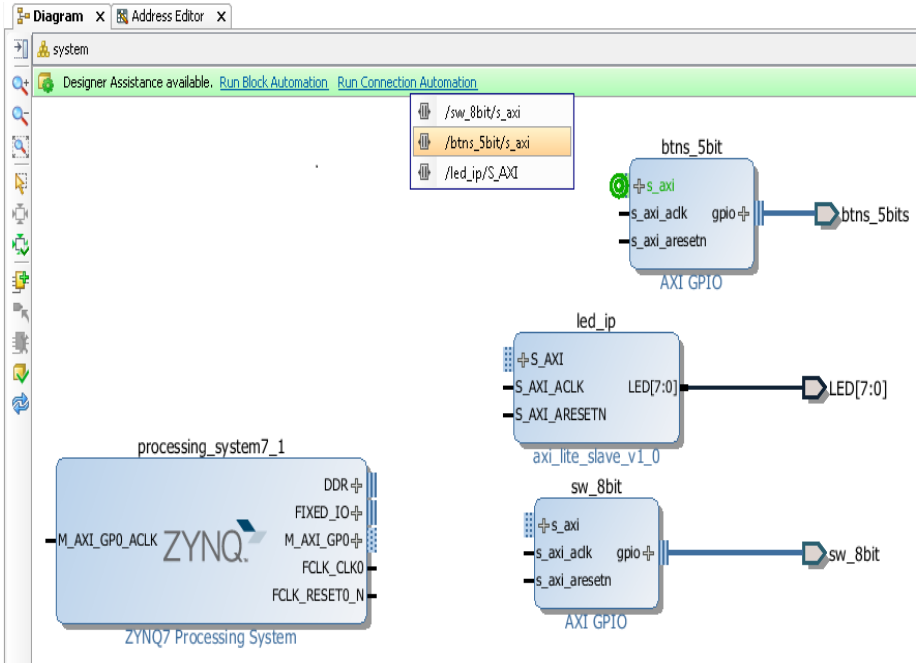
Connecting IP

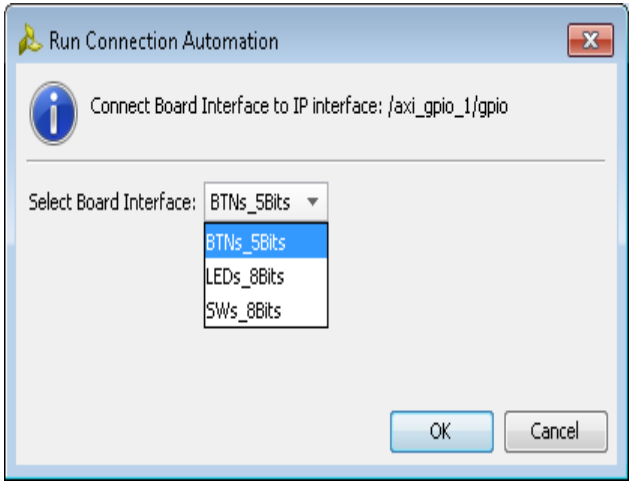
- **Add IP from the IP Catalog**
- **Click and drag to find connections**
- **Valid connections Highlighted**
- **Designer Assistance, Connection automation**
  - If Board Support available, IP can be connected to external pins
- **Or manually create and connect (external) ports**



Designer assistance; Block Automation, Connection Automation

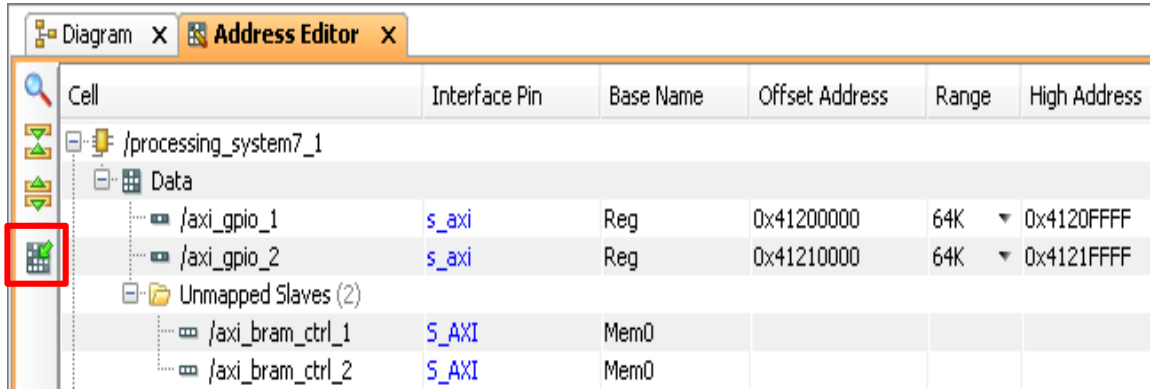
- Block, Connection
- Can automatically connect IP blocks
- Automatically insert required blocks
- E.g. Add BRAM; Automation will insert and connect BRAM controller and Reset logic
- If Board Support is available, IP can automatically be connected to top level ports





Assign Addresses

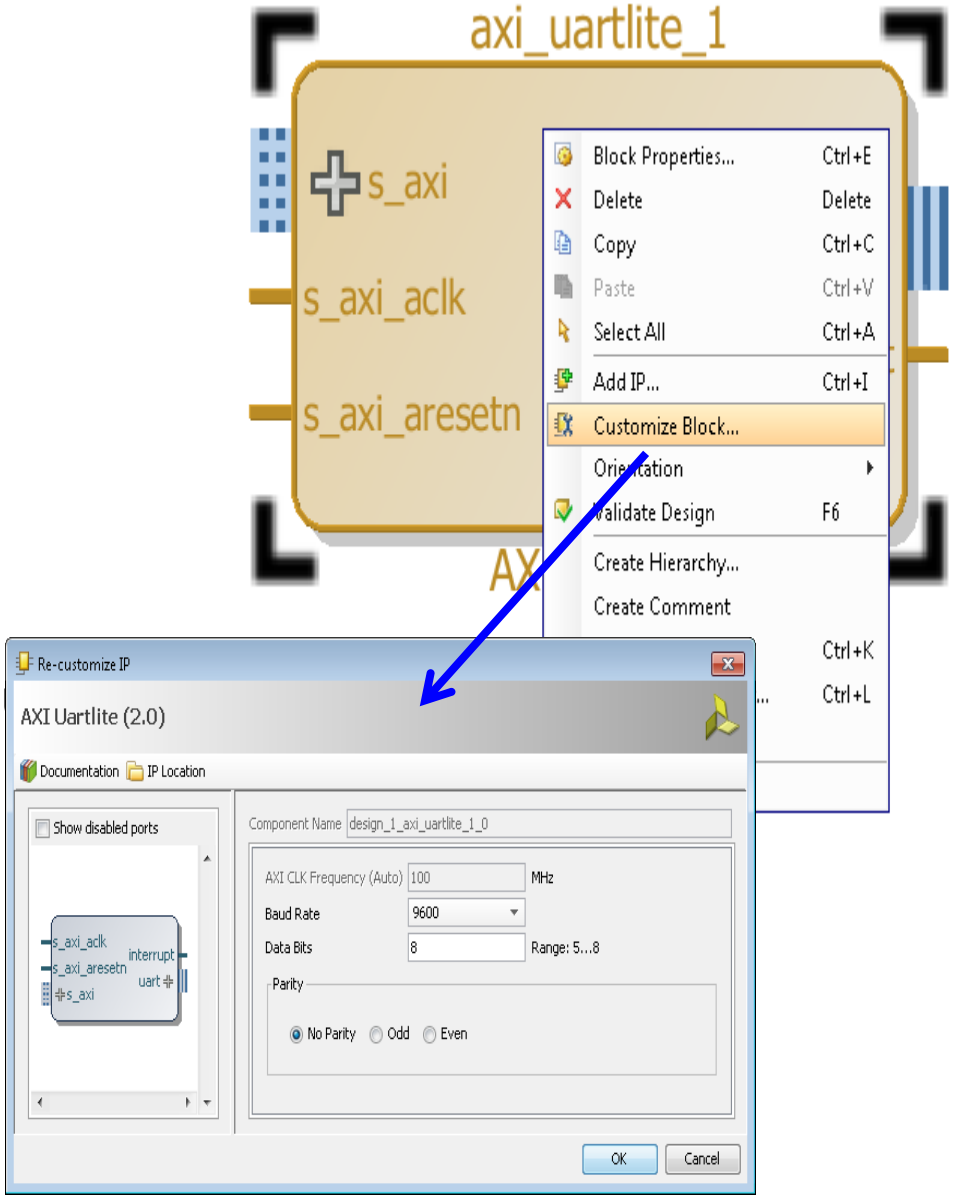
- Peripherals in the Zynq™ AP SoC PS have fixed addresses and do not appear in the address map when an IP is added to the system
- For PS peripherals Click on the Auto Assign Addresses button



- The address will be generated and show the generated addresses of the added IP
- The fixed addresses of the configured peripherals of the PS

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 yo want



Simulation, Synthesis, and Implementation

- Simulation is verifying the functionality of the design .
  There are two kinds of simulation**:**
  1- behavioral (functional) simulation which is done pre-fit (before synthesis)
  2- Timing simulation which is done post-fit (after synthesis) to ensure that it

  has achieved the required timing.
- Synthesis is to implement the design by translating the design into a next level of abstraction: from RTL level to gate level.
- Implementation is to translate netlist into the placed and routed FPGA design.
- From SW designers' view, Simulation process is something like the debugging process, while the Synthesis and implementation process is something like the compile-link-make process. The Synthesis process is the key point of EDA technology, which makes the Automatic process possible.

Bitstream Generation

- **After defining the system hardware, the next step is to create hardware netlists if the system hardware has logic in PL**
- **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
- **The design and block diagram must be open before synthesise and implementation can be carried out**
- **If the system contains hardware in the PL, the bitstream must be generated**
- **The PL (FPGA) must be programmed before application can be downloaded and executed**

Create a TestApp Application in SDK

- Export hardware platform to SDK
- Create a test application to test the newly added IPs
- The FPGA must be programmed with the generated hardware bitstream before an application can be run