Hardware Design Using EDK

Objectives

After completing this module, you will be able to:

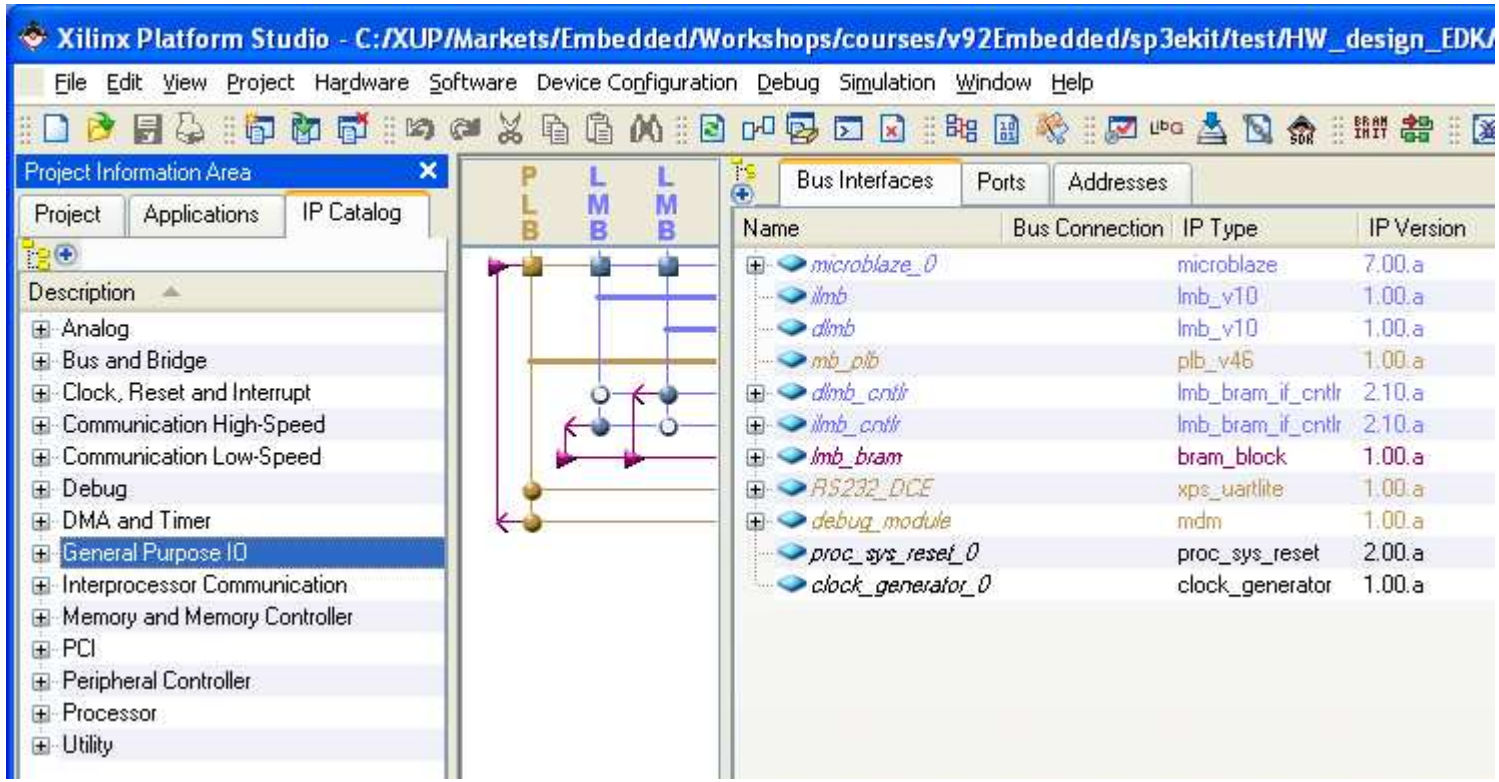
- Describe how to add hardware to an existing XPS project
- Discuss the function of Platform Generator (PlatGen)
- Utilize the integration between ISE™ and Xilinx Platform Studio (XPS) to enhance the design flow
- Utilize the Xflow in XPS
- Describe the steps involved in creating a submodule with XPS and integrating the submodule into a bigger system with ISE

Outline

- **Adding System Components**
- Generating the System netlists (PlatGen)
- Generating the Bitstream
  - Manually with ISE: Project Navigator Integration
    - Top Level
    - Submodule
  - Automatically from XPS: Xflow Integration

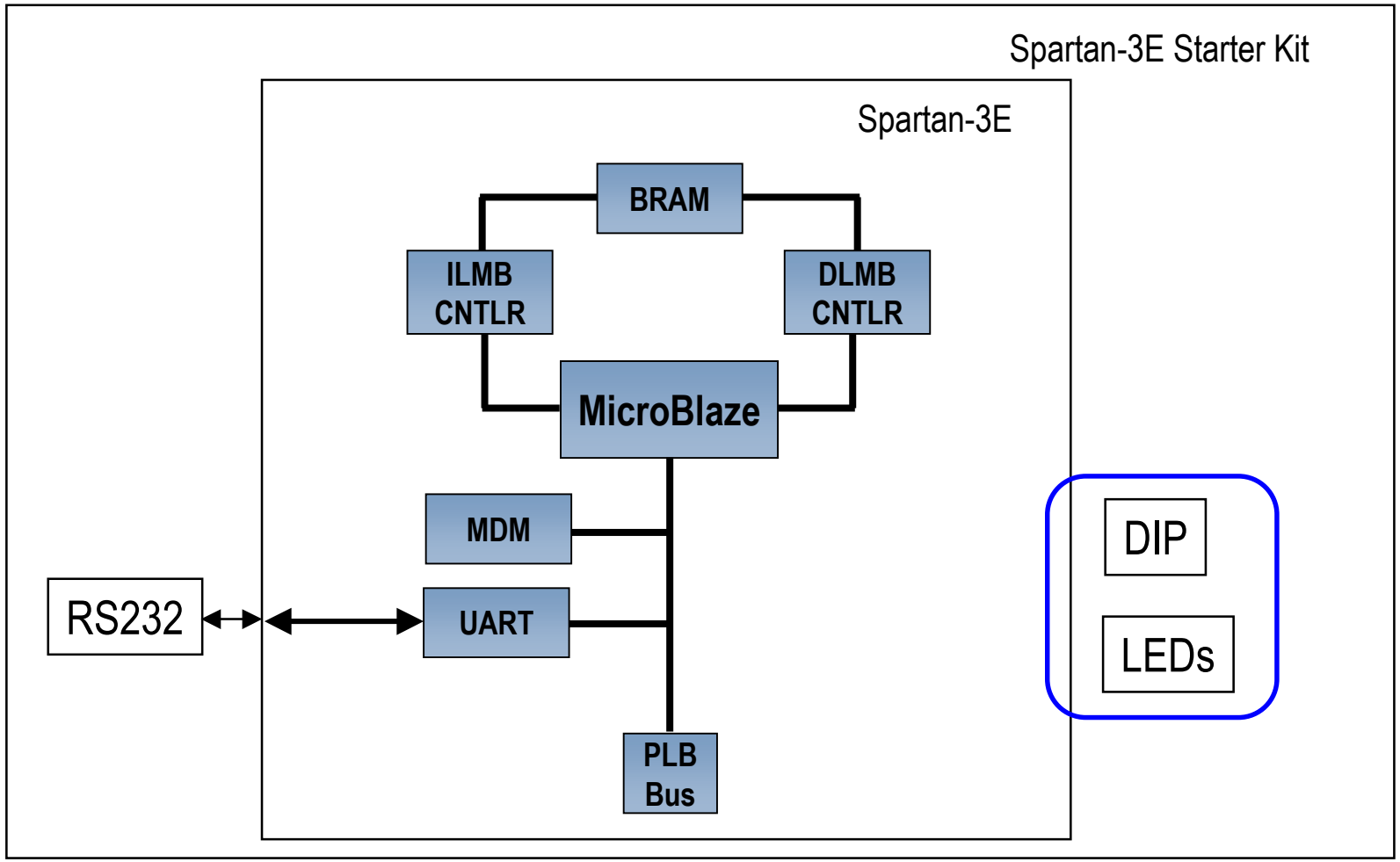
Embedded Design

Initial System created with Base System Builder targeting Spartan-3E Starter Kit



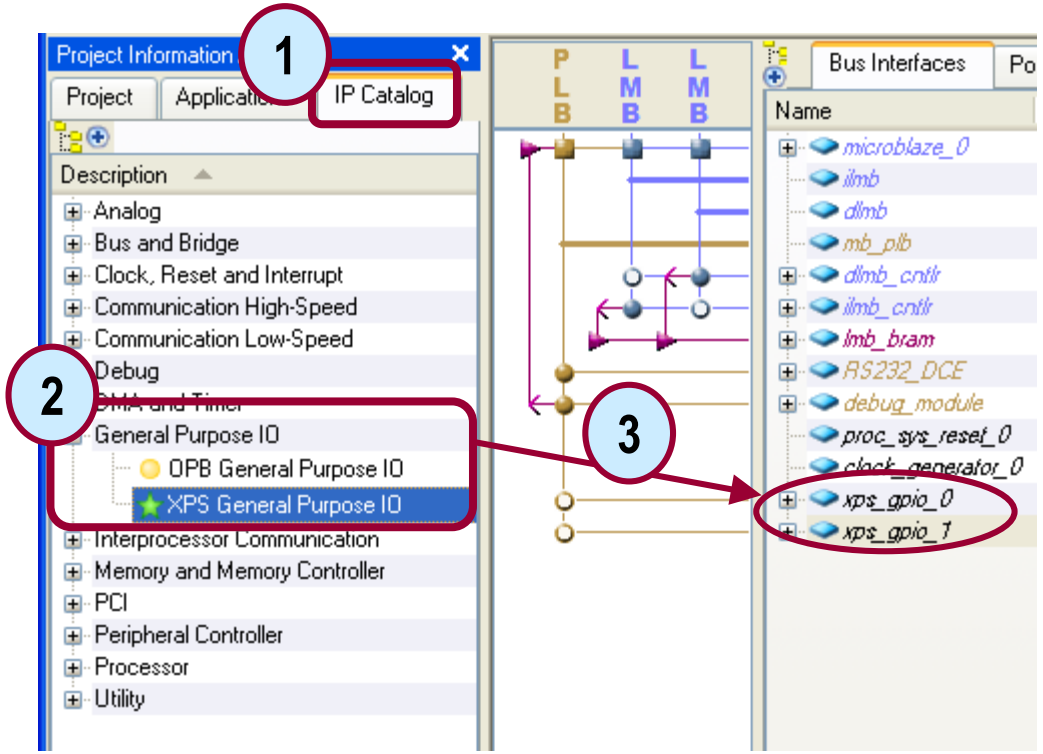
Embedded Design

Add GPIO Peripherals to connect to on-board DIP Switches and LEDs



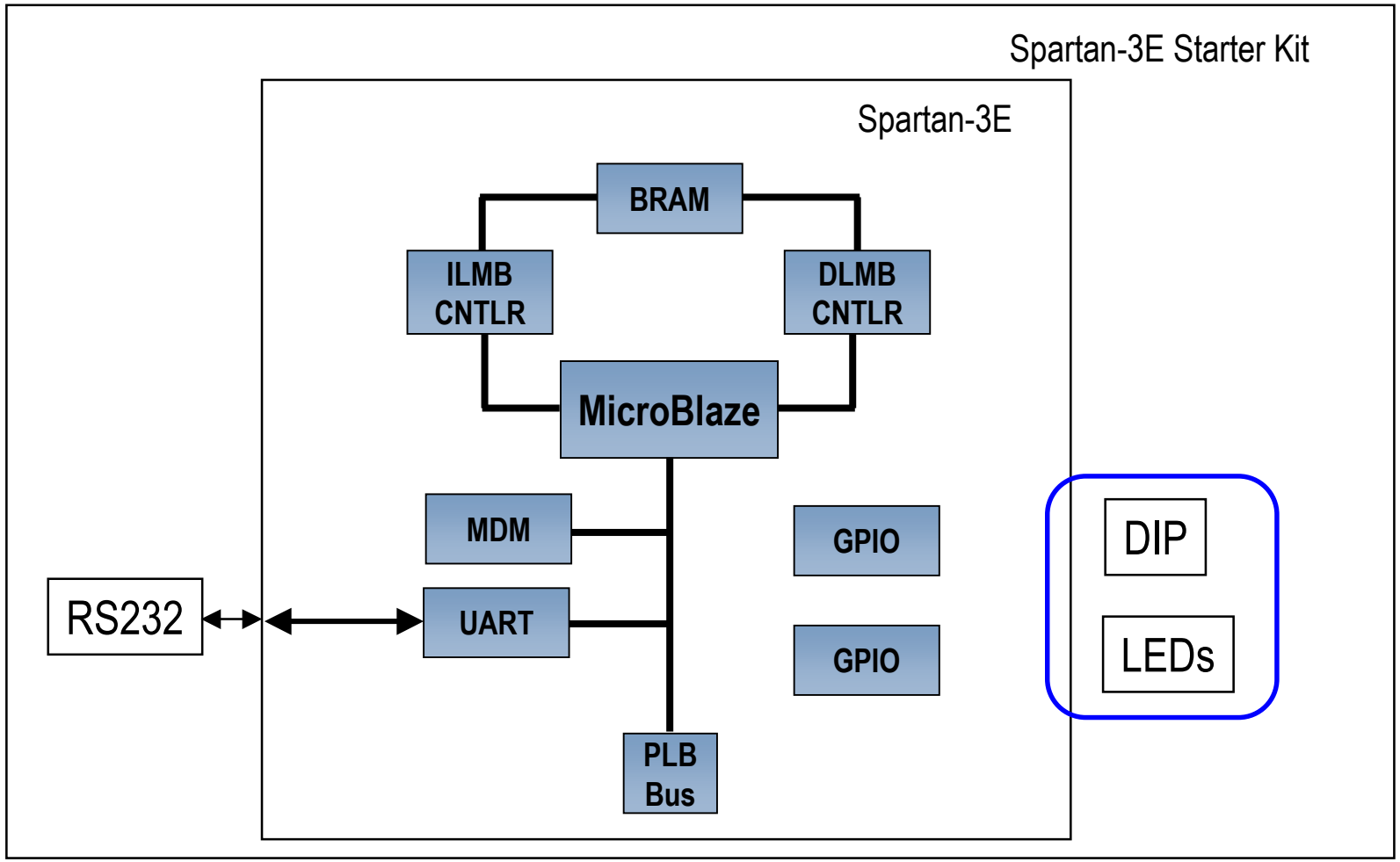
Adding IP to Design

1. To add hardware in a new, empty project or to an existing project, select **IP Catalog** tab in XPS
2. Expand group(s) of IP in the left window
3. Select an IP and drag it to the System Assembly View window or double-click on the selected IP to be included into the system MHS file



Embedded Design Progress

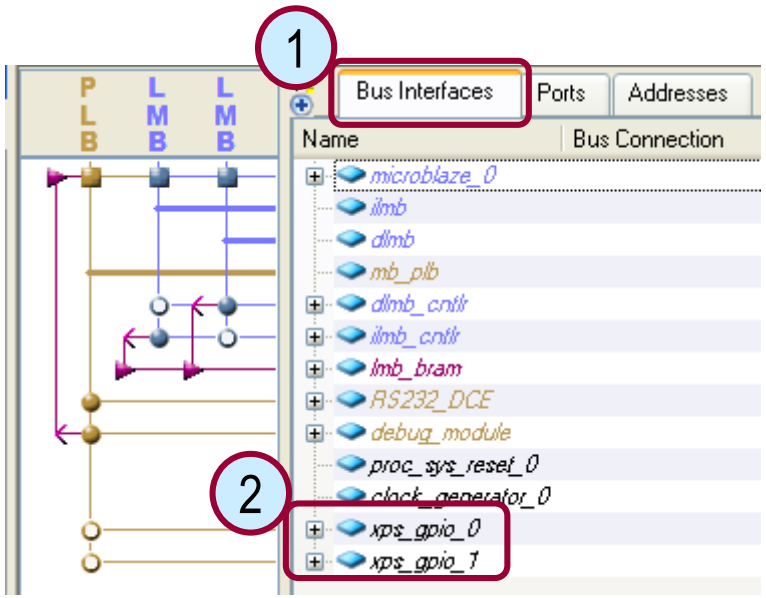
GPIO Peripherals Added to System

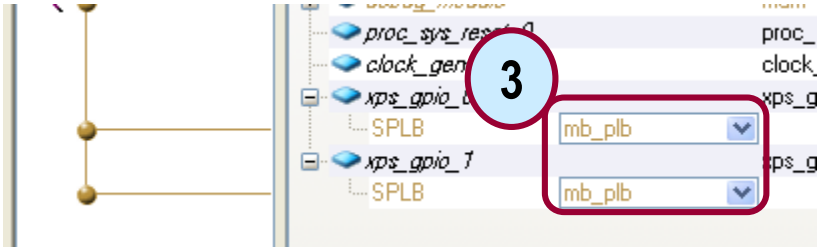


Making Bus Connections

MicroBlaze communicates with external peripheral devices using busses

1. Select Bus Interfaces tab
2. Expand Peripherals in System View
3. Click under Bus Connection column, and select a bus instance to which it needs to connect

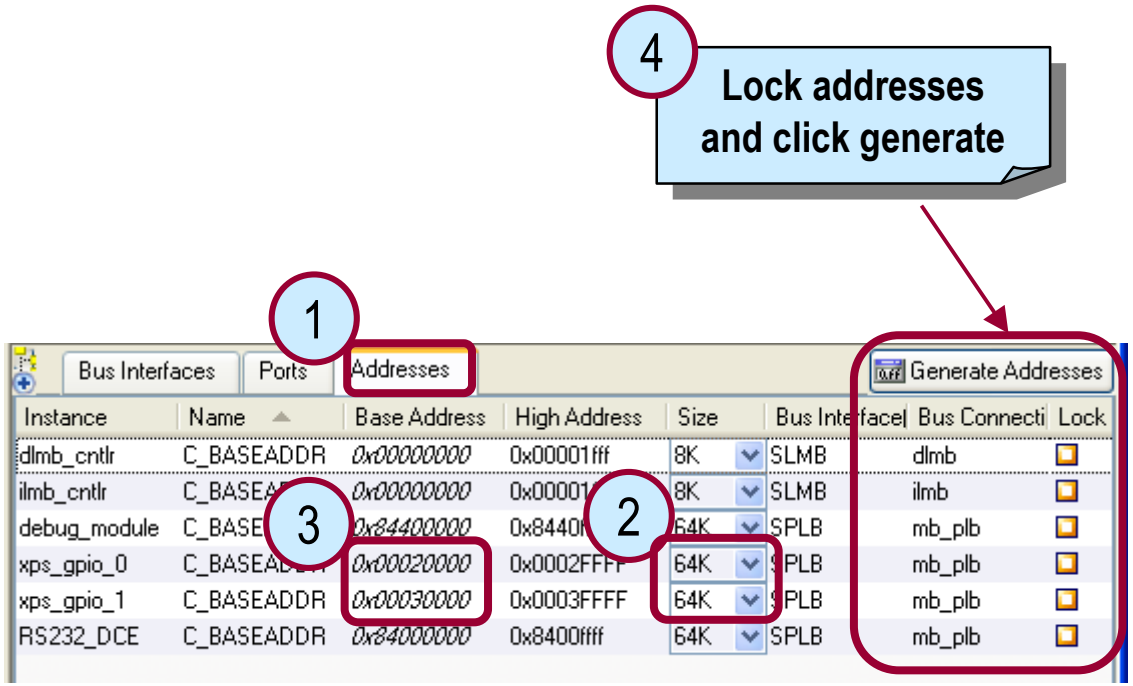




Assigning Addresses

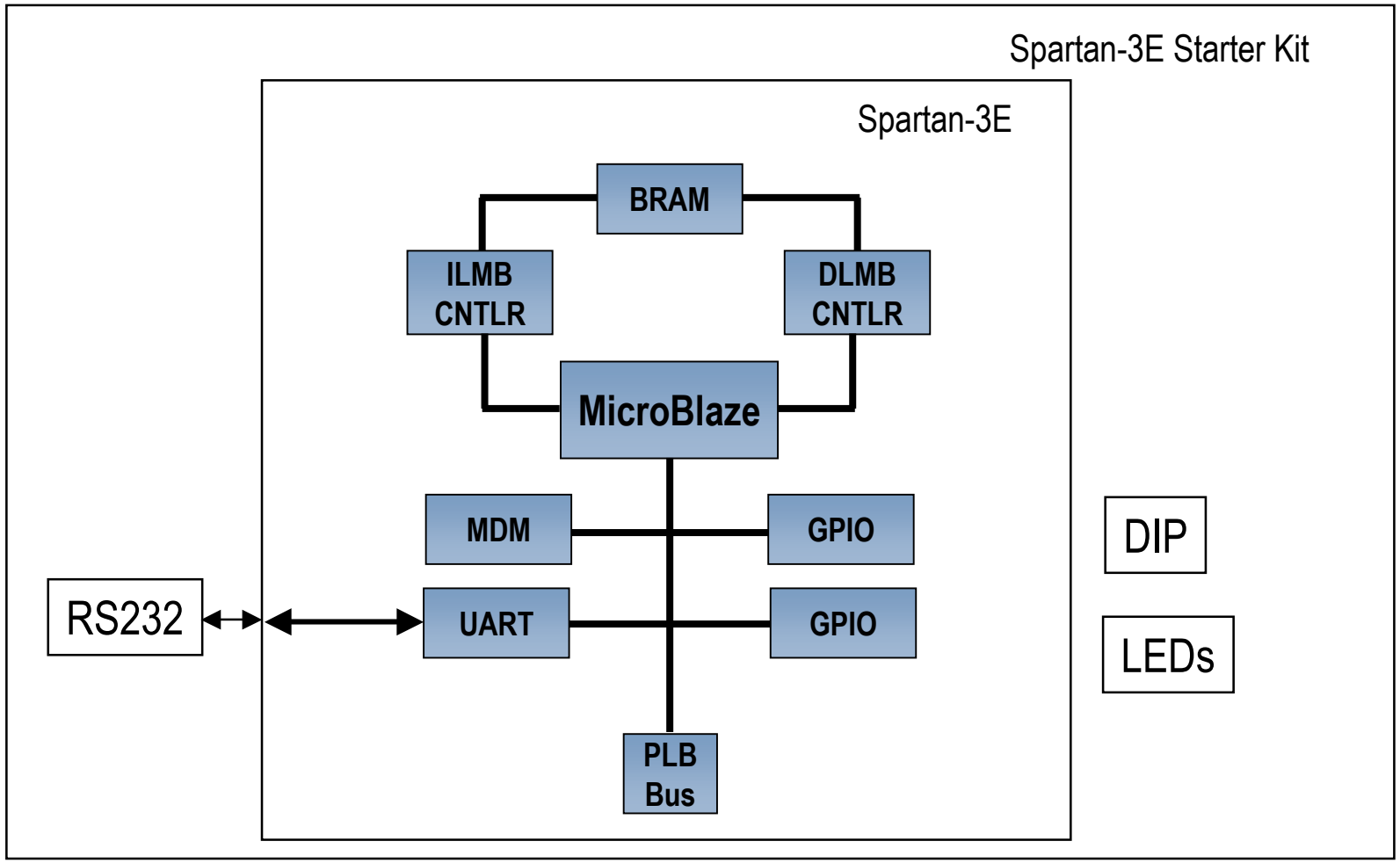
MicroBlaze communicates with external devices through registers or memories at specific address ranges

1. Select **Addresses** filter
2. Click in the size column and select desired size
3. Enter base address
   - XPS will calculate the high address from base address and size entries
4. Instead of entering base address, lock addresses of instances for which you don't want XPS to change address and then click Generate Addresses button



Hardware Design Progress

GPIO instances are now connected to PLB bus, with Base/High Addresses Assigned



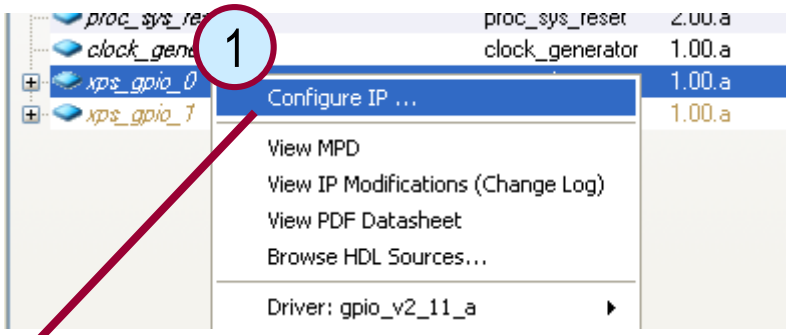
Parameterize IP Instances

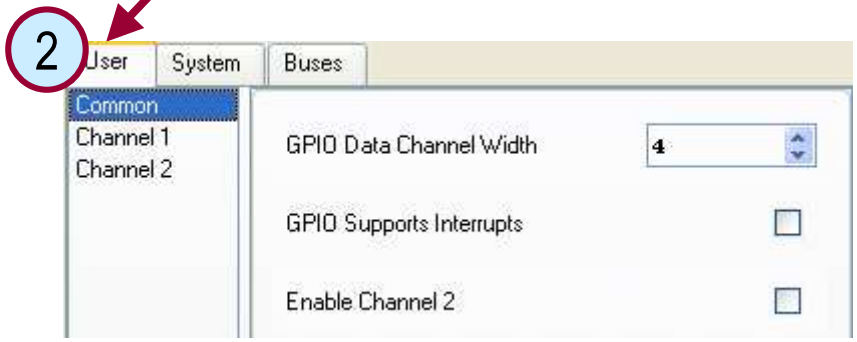
Set a GPIO to a 4-bit input to connect to the 4 DIP Switches on the Board

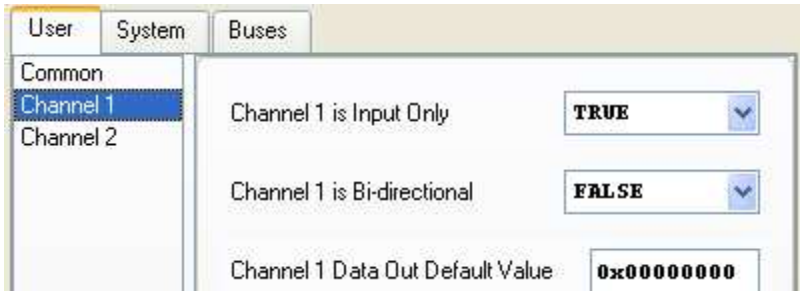
1. Double click the instance or right click on the instance and select **Configure IP** to list the configurable parameters

2. Enter new values
   - Override defaults

*Take similar steps for the other GPIO

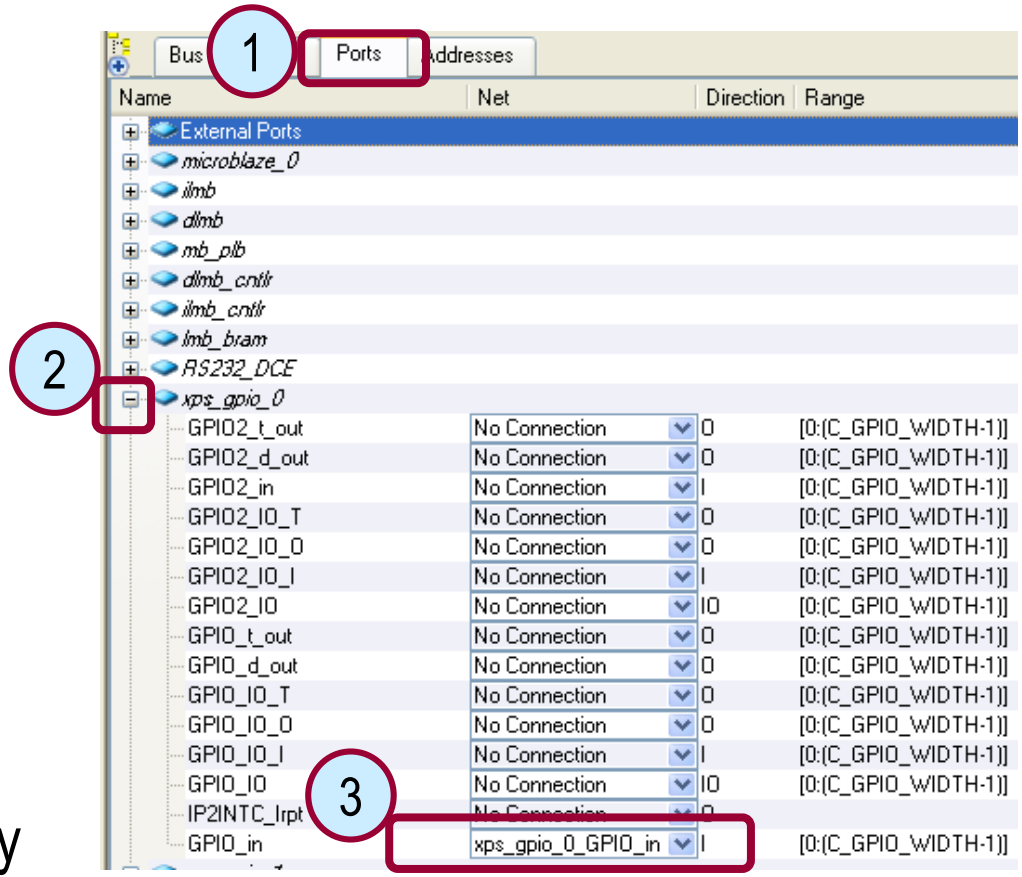


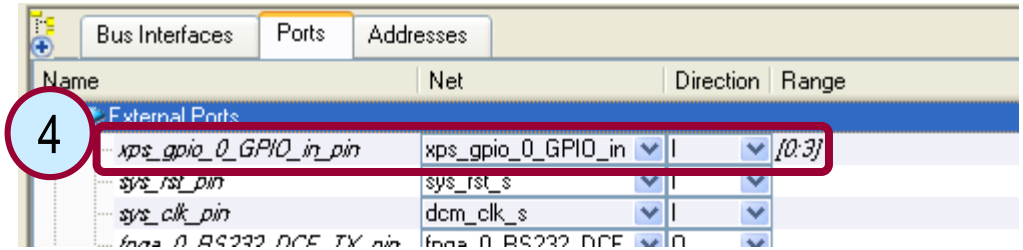




Connecting Ports

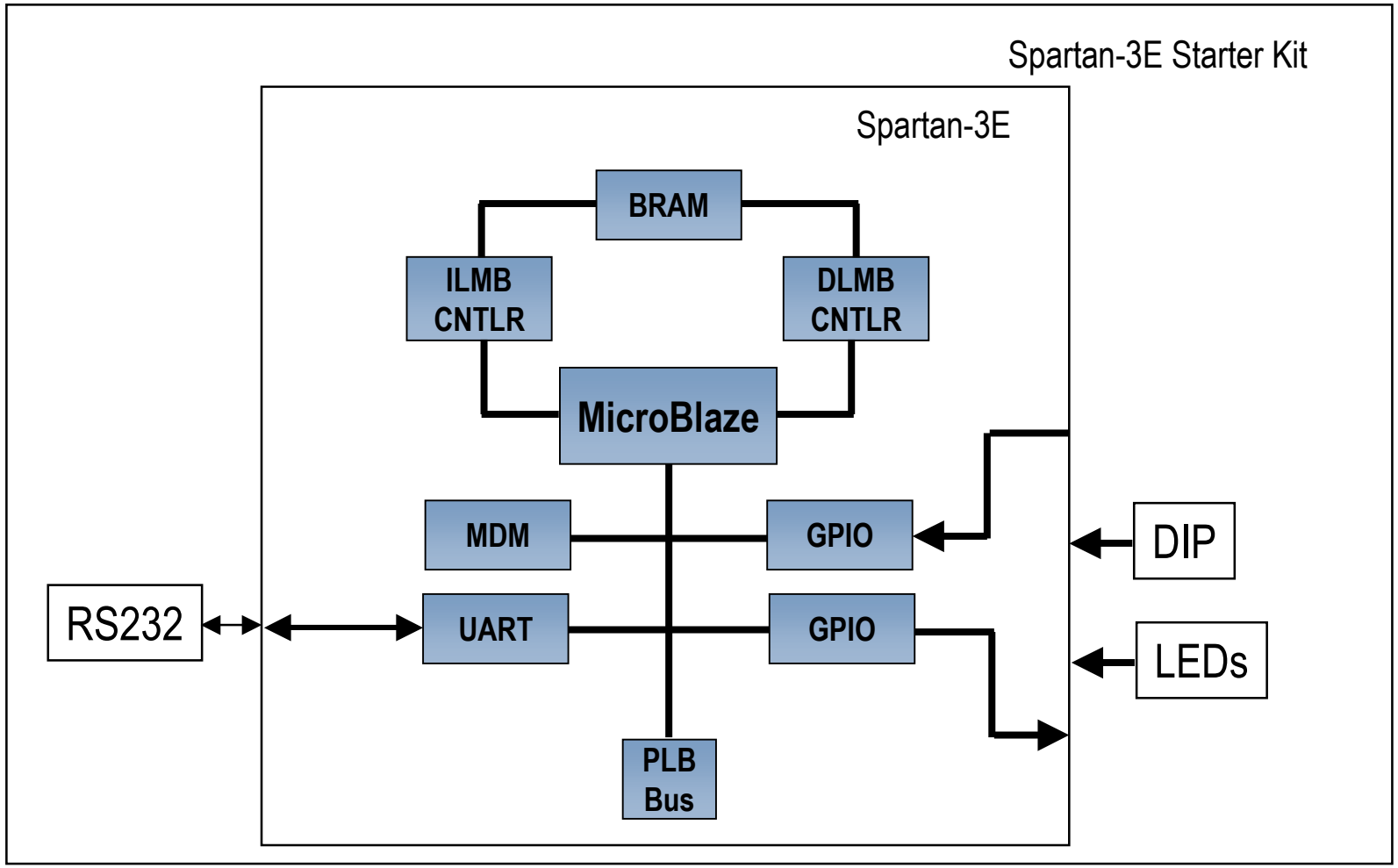
1. Select **Ports** filter
2. Click on plus sign to see available ports
3. Click under the Net column and select appropriate signal
   - –If the port is external in the design then make it external
4. Verify the external pin entry in the External Ports section





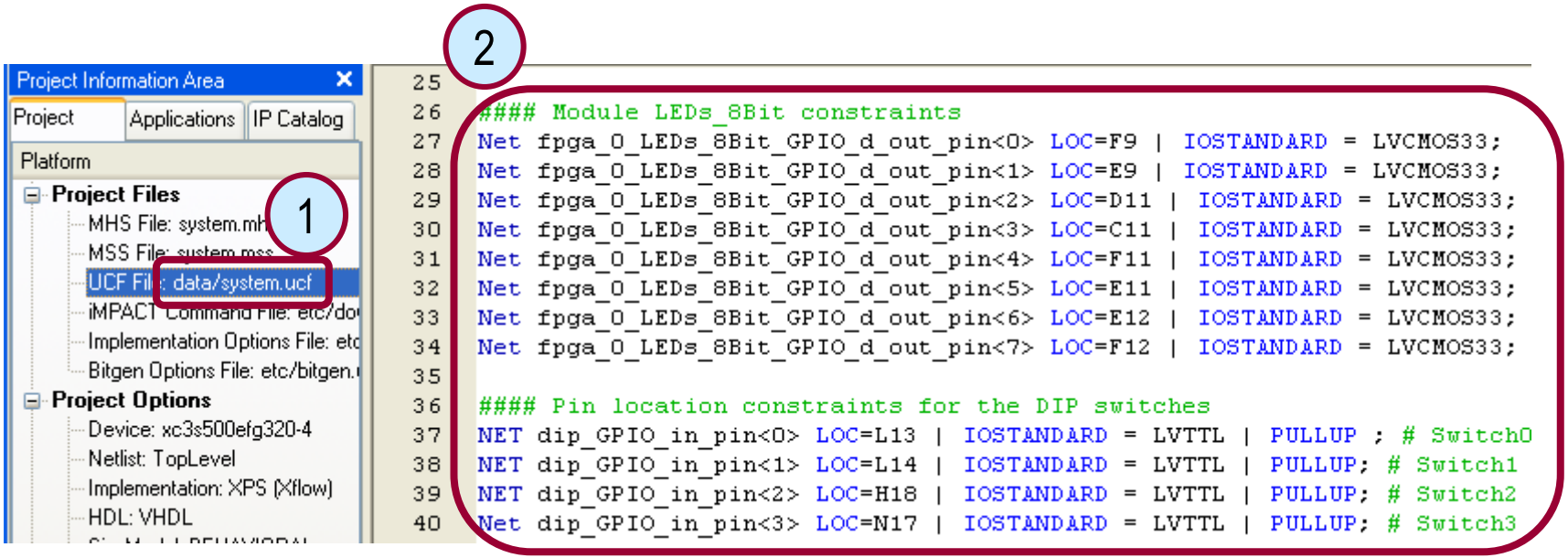
Hardware Design Progress

External Port Connections for both GPIO instances have been established



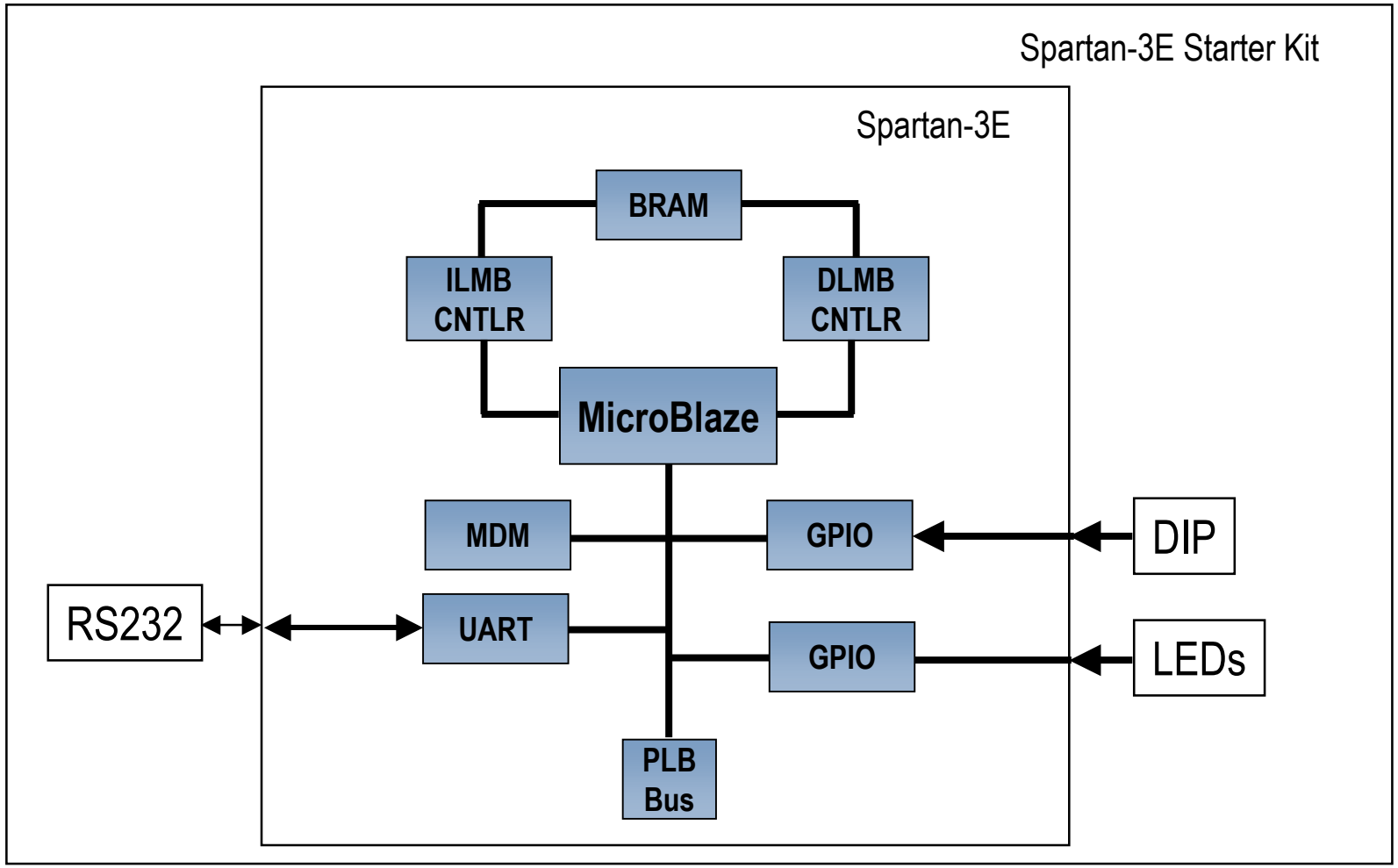
Make Pin Assignments

1. Double-click the system.ucf under the **Project** tab
2. Enter the pin location constraints (refer to the board user manual)



Hardware Design Progress

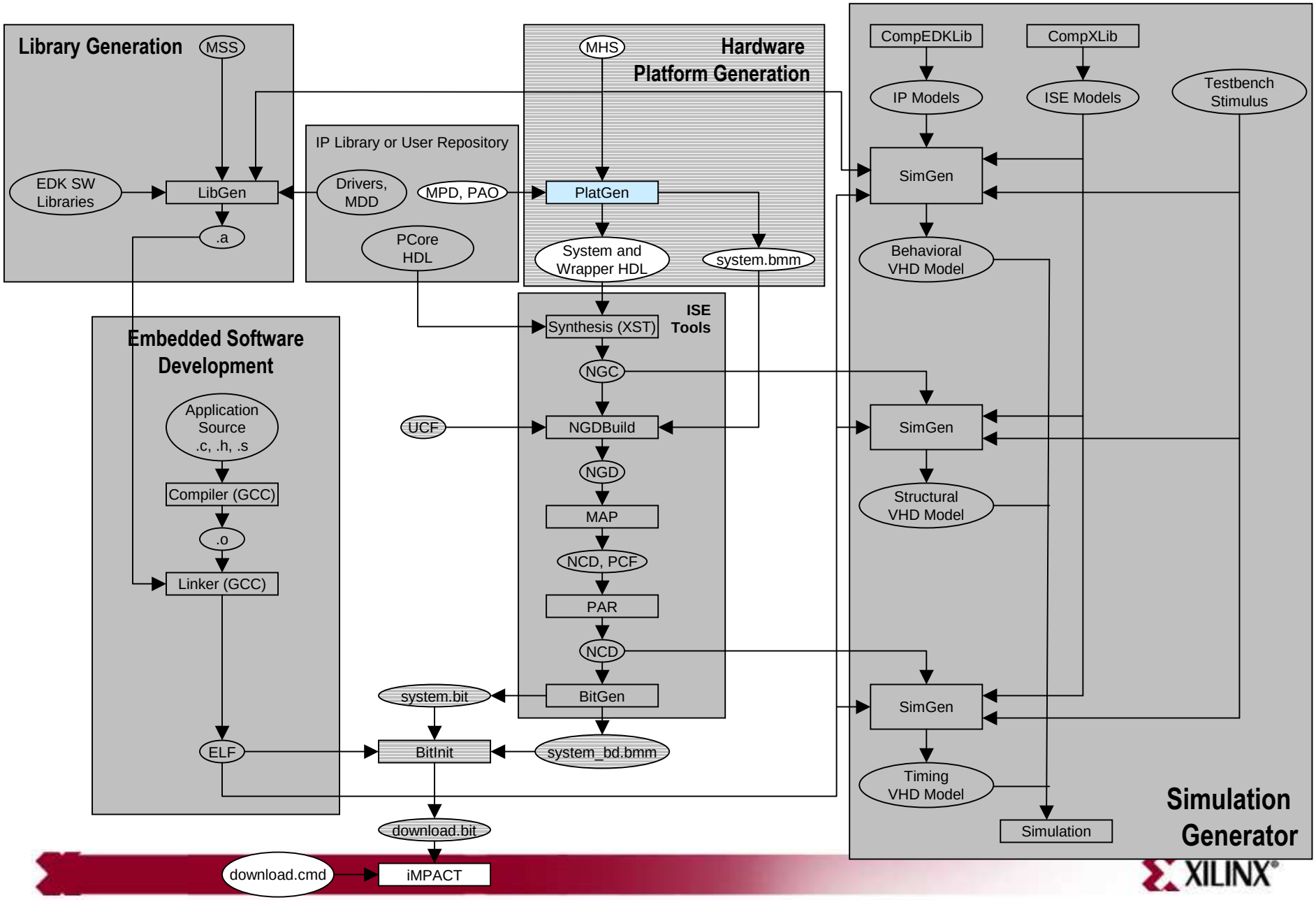
The GPIO instances are connected to the external DIP switches and LEDs on the board



Outline

- Adding System Components
- **Generating the System Netlists (PlatGen)**
- Generating the Bitstream
  - Manually in ISE: Project Navigator Integration
    - Top Level
    - Submodule
  - Automatically from XPS: Xflow Integration

Hardware Creation Flow



Hardware Design

- After defining the system hardware and connectivity, the next step is to create hardware netlists with the Platform Generator (PlatGen)
- PlatGen inputs the following files:
  - Microprocessor Hardware Specification (MHS) file
  - Microprocessor Peripheral Definitions (MPD) file
- PlatGen constructs the embedded processor system in the form of hardware netlists (HDL and implementation netlist files)

Hardware Design Files

MHS and MPD

Microprocessor Hardware Specification (MHS) File

```
BEGIN opb_uartlite
 PARAMETER INSTANCE = RS232_Uart
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_BAUDRATE = 115200
 PARAMETER C_DATA_BITS = 8
 PARAMETER C_ODD_PARITY = 0
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_CLK_FREQ = 100000000
 PARAMETER C_BASEADDR = 0x40600000
 PARAMETER C_HIGHADDR = 0x4060ffff
 BUS INTERFACE SOPB = opb
```

MHS overrides MPD

Microprocessor Peripheral Definitions (MPD) File

```
## Bus Interfaces
BUS_INTERFACE BUS = SOPB, BUS_STD = OPB, BU

## Generics for VHDL or Parameters for Veri
PARAMETER C_BASEADDR = 0xFFFFFFFF, DT = std
PARAMETER C_HIGHADDR = 0x00000000, DT = std
PARAMETER C_OPB_DWIDTH = 32, DT = integer
PARAMETER C_OPB_AWIDTH = 32, DT = integer
PARAMETER C_DATA_BITS = 8, DT = integer, RA
PARAMETER C_CLK_FREQ = 125_000_000, DT = in
PARAMETER C_BAUDRATE = 9600, DT = integer,
```

MPD contains all of the defaults

PlatGen

PlatGen Generated Directories

- project_directory
  - hdl directory
  - implementation directory
  - synthesis directory

- **HDL** directory
  - *system*.[vhd|v] file (if top level)
  - *system_stub*.[vhd|v] file (if submodule)
  - *peripheral_wrapper.[vhd|v]* files
- **Implementation** directory
  - *peripheral_wrapper.ngc* files
  - *system.ngc* file
  - *system.bmm* file
- **Synthesis** directory
  - *peripheral_wrapper.[prj|scr] files*
  - *system*.[prj|scr] files

PlatGen Memory Generation

- Platform Generator generates the necessary banks of memory and the initialization files for the block RAM block (bram_block). The block RAM block is coupled with a block RAM controller
- Current block RAM controllers for MicroBlaze include the following:
  - PLB block RAM controller (xps_bram_if_cntlr)
  - OPB block RAM controller (opb_bram_if_cntlr)
  - LMB block RAM controller (lmb_bram_if_cntlr)

PlatGen Memory Sizes

- Memory sizes

Architecture	Memory Size (kBytes) 32-bit byte-write	Memory Size (kBytes) 64-bit byte-write
Spartan™-II	2, 4	4,
Spartan-IIE	2, 4, 8, 16	4, 8, 16, 32
Spartan-3	8, 16, 32, 64	16, 32, 64, 128
Spartan-3e	8, 16, 32, 64	16, 32, 64, 128
Virtex™	2, 4, 8, 16	4, 8, 16, 32
Virtex-E	2, 4, 8, 16	4, 8, 16, 32
Virtex-II	8, 16, 32, 64	16, 32, 64, 128
Virtex-II PRO	8, 16, 32, 64	16, 32, 64, 128
Virtex-4	2, 4, 8, 16, 32, 64, 128	4, 8, 16, 32, 64, 128, 256
Virtex-5	4, 8, 16, 32, 64, 128, 256	8, 16, 32, 64,128, 256, 512

- Memory must be built on 2^n boundaries
  - Let **I** be the unsigned number formed by the starting address and **S** be the size of the memory. If **I/S** is the integer, then the memory is built on the $\mathbf{2^n}$ boundary
  - 1-KB (1024) memory at 0x4000 (16384) is at the 2^n boundary (16384/1024 = 16); whereas, 1 KB (1024) at 0x4100 (16640) is not (16640/1024 = 16.25)

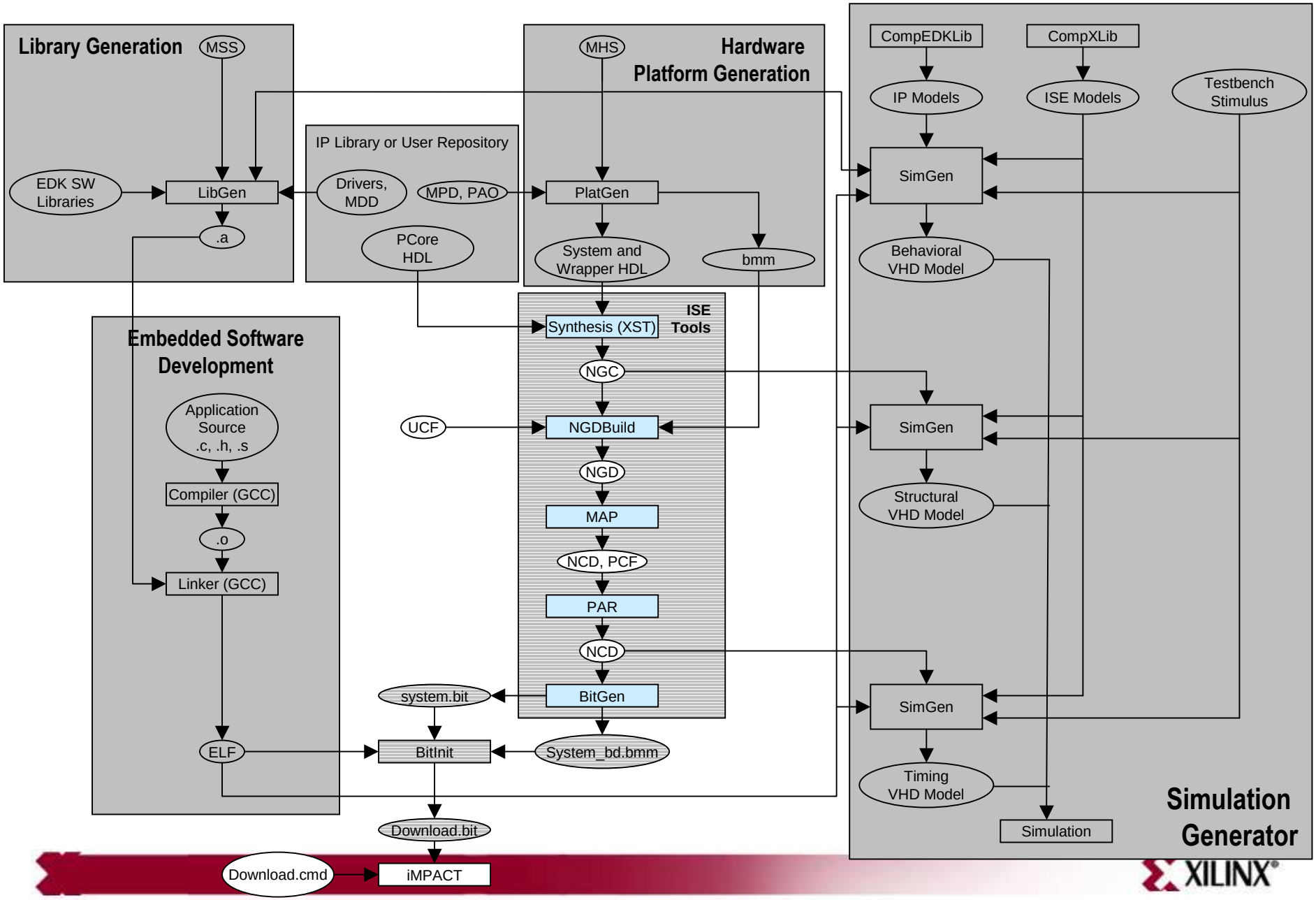
Block Memory Map

- A Block RAM Memory Map (BMM) file contains a syntactic description of how individual block RAMs constitute a contiguous logical data space
- PlatGen has the following policy for writing a BMM file:
  - If PORTA is connected and PORTB is not connected, the generated BMM will be from PORTA point of reference
  - If PORTA is not connected and PORTB is connected, the generated BMM will be from PORTB point of reference
  - If PORTA is connected and PORTB is connected, the generated BMM will be from PORTA point of reference

Outline

- Adding System Components
- Generate the System Netlists (PlatGen)
- **Generate the Bitstream**
  - Manually in ISE: Project Navigator Integration
  - Automatically with XPS: Xflow Integration

Hardware Implementation Flow



Outline

- Adding System Components
- Generate the System Netlists (PlatGen)
- **Generate the Bitstream**
  - **Manually in ISE: Project Navigator Integration**
  - Automatically with XPS: Xflow Integration

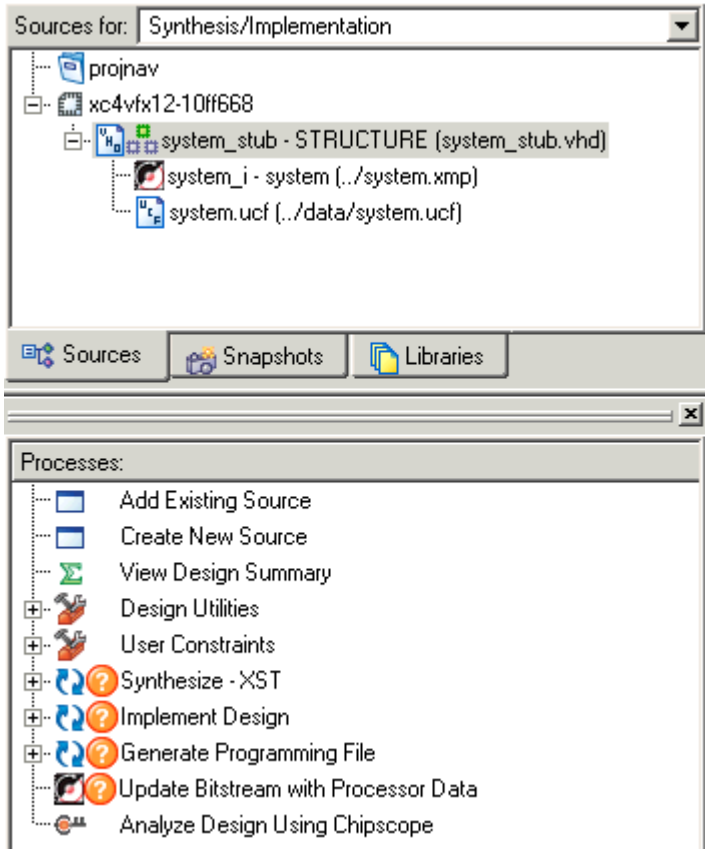
Manual ISE Flow

User generates bitstream in ISE

- The processor system (.xmp) can be added and connected in an ISE project
  - XPS can be invoked from ISE
- Benefits include
  - Add additional logic to the FPGA design
  - Synthesize the design by utilizing ISE™-supported synthesis tools
  - Control the FPGA implementation flow by using ISE
    - Timing and constraints entry
    - Implementation tool flow control
    - Point tool control
      - FPGA Editor tool
      - Constraints Editor tool
      - ChipScope™ Pro tool

Instantiate Processor System in ISE

- Two ways to use the XPS and ISE tools to process embedded systems:
  - Top-Down
    - Invoke ISE and create a top-level project
    - Then create a new embedded processor source to include in the top-level design. This automatically invokes XPS, where you develop your embedded sub-module
  - Bottom-Up
    - Invoke XPS and develop your embedded processor design as a sub-module
    - Later, invoke ISE and add the embedded sub-module as a source to include in your top-level ISE project.



Outline

- Adding System Components
- Generating the System Netlists (PlatGen)
- Generate the Bitstream
  - Manually in ISE: Project Navigator Integration
  - **Automatically with XPS: Xflow Integration**

Hardware Implementation

Automated Approach

- Xflow – Automatically implements hardware and generates the bitstream
  - Input files → .ngc netlists, system.bmm file, system.vhd, .ucf
  - Output Files → system.bit, system_bd.bmm

  - Xflow calls the ISE™ Implementation tools using fast_runtime.opt file
    - NGDBuild, MAP, PAR, and TRACE are executed
  - Xflow then calls the BitGen program using bitgen.ut file
    - BitGen generates the bit file system.bit
    - BitGen also generates the back-annotated system_bd.bmm BMM file, which contains the physical location of the block RAMs

Automatic ISE Flow

XPS generates bitstream using Xflow

- Benefits:
  - Independent design of the processor system
  - One GUI for performing all design work

- Limitations:
  - No direct control of synthesis and implementation options
  - No point-tool support
  - The embedded system design must be the top level of the design

Xflow

Required XPS Directory Structure

- project_directory
  - Code/TestApp directory [optional]
  - data directory
  - etc directory
  - pcores
  - synthesis

- Code/TestApp directory
  - <application>.c
- data directory
  - <system>.ucf
- etc directory
  - bitgen.ut
  - download.cmd
  - fast_runtime.opt
  - BSDL files
- pcores directory
  - User IP
  - Customized block RAM controllers

Controlling Xflow

- A file called fast_runtime is in the etc directory
- This is what it looks like:

```
# Options for Translator
# Type "ngdbuild -h" for a detailed list of ngdbuild command line options
Program ngdbuild
-p <partname>;          # Partname to use — picked from xflow commandline
-nt timestamp;          # NGO File generation. Regenerate only when
                        # source netlist is newer than existing NGO file (default)
-bm <design>.bmm;       # block RAM memory map file
<userdesign>;           # User design — pick from xflow command line
<design>.ngd;           # Name of NGD file. Filebase same as design filebase
End Program ngdbuild
```

Knowledge Check

- What are some of the advantages of using ISE™ and XPS integration?

- What are some of the advantages of using Xflow and XPS integration?

Answers

- What are some of the advantages of using ISE™ and XPS integration?
  - Add additional logic to the FPGA design
  - Synthesize the design by utilizing ISE-supported synthesis tools
  - Control the FPGA implementation flow by using ISE
- What are some of the advantages of using Xflow and XPS integration?
  - One GUI to perform all design work
  - Simple push-button flow

Knowledge Check

- What is the smallest memory size that PlatGen can generate for a Spartan™-IIE device?

- Why is the address 0xFFFF_B100 NOT a valid BASEADDR for a Local Memory Bus (LMB) block RAM controller?

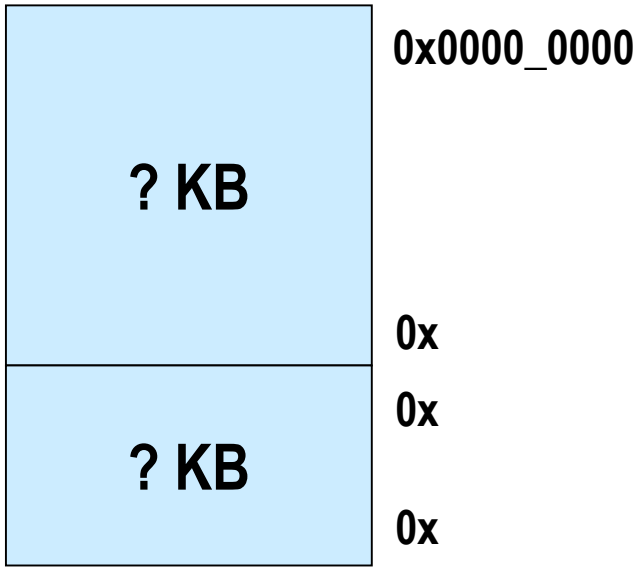
- What will the BAUDRATE for the peripheral be:
  - If the MPD file has the following parameter: C_BAUDRATE = 9600
  - If the MHS file has the following parameter: C_BAUDRATE = 115200

Answers

- What is the smallest memory size that PlatGen can generate for a Spartan™-IIE device?
  - **2 KB**
- Why is the address 0xFFFF_B100 NOT a valid BASEADDR for a Local Memory Bus (LMB) block RAM controller?
  - **It is not on a 2n boundary**
- What will the BAUDRATE for the peripheral be:
  - If the MPD file has the following parameter: C_BAUDRATE = 9600
  - If the MHS file has the following parameter: C_BAUDRATE = 115200
    - **The BAUDRATE will be 115200**

Knowledge Check:
Memory Space

- How do you build a 48-KB OPB BRAM memory space for a MicroBlaze™ processor in a Spartan™-3E device?



Answers:
Memory Space

- How do you build a 48-KB OPB BRAM memory space for a MicroBlaze™ processor in a Spartan™-3E device?

Where Can I Learn More?

- Tool documentation
  - ***Embedded System Tools Guide*** $\rightarrow$ ***Xilinx Platform Studio***
- Support Website
  - EDK Website: www.xilinx.com/edk