Introduction to Hardware/Software Co-Design on Zynq SoC

OBJECTIVES

- Create a Hardware/Software System using the ZYBO Board or the ZYBO Z7-10 Board.
- Create a Block Based Design in Vivado 2019.1.
- Use of the AXI GPIO peripheral to control LEDs via software.
- Creating a software application in SDK.

REQUIRED FILES (ZYBO, ZYBO Z7-10)

Definition files:	▪ For ZYBO and ZYBO Z7-10 Boards: `vivado-boards-master.zip`. ▪ (old manual method) for ZYBO Board: `ZYBO_zynq_def.xml`.
XDC files:	▪ ZYBO (`ZYBO_Master.xdc`), ZYBO Z7-10 (`Zybo-Z7-Master.xdc`)
Software file	▪ `LED_test_tut_1Cp.c`

BOARD SETUP FOR HARDWARE/SOFTWARE CO-DESIGN IN VIVADO (ZYBO, ZYBO Z7-10)

- Every board includes different interfaces and components (e.g.: slide switches, push buttons, LEDs, USB-UART, etc.) the Zynq-7000 PS and PL interact with. This information is specified in `.xml` definition files. Vivado already includes this information for a large set of boards, except for the ZYBO or ZYBO Z7-10. To set this up:
  - ✓ Load .xml file into PS (old, only tested for ZYBO Board): For the ZYBO Board, we can load the `ZYBO_zynq_def.xml` file into a project (when configuring the PS). However, this needs to be done for every project.
  - ✓ Files installed in Vivado: Preferred, one-time set-up procedure. You need to place the definition files (available in `vivado-boards-master.zip`) for the ZYBO and ZYBO Z7-10 boards into the Vivado installation directory:
    - Vivado 2019.1: Go to `{Vivado installation folder}\2019.1\data\boards\board_files\` and paste the folders `\zybo` and `\zybo-z7-10` contained in `vivado-boards-masters.zip`. Restart Vivado if it was open.

FIRST HARDWARE/SOFTWARE DESIGN IN VIVADO

- This experiment is taken from the Zynq Book Tutorial → Exercise 1 (A, B, C). Refer to that material for step-by-step instructions. Here, we provide abridged instructions for two approaches:
  - ✓ 1.A. Load .xml file into PS: We manually indicate the Zynq-7000 device, the LED ports, and the Processing System (PS) `.xml` definition file. A few changes were made to the procedure in the Zynq Book Tutorial.
  - ✓ 1.B. Files installed in Vivado: A few extra details are provided. It works for the ZYBO or ZYBO Z7-10 Board.
- To create a hardware/software design in Vivado, we will create a 'Block Design Project' where we can instantiate and interconnect reconfigurable hardware components and the hard-wired Processing System (PS).

1A. CREATING A BLOCK DESIGN PROJECT IN VIVADO - ZYBO BOARD (LOAD .XML FILE INTO PS)

- Create a new Vivado project `first_zynq_design`.
  - ✓ Make sure the default language is VHDL, so that the system wrapper is created in VHDL.
  - ✓ At Default Part, go to Parts and select the **ZYNQ XC7Z010-1CLG400** device.
- Once created, add the `ZYBO_Master.xdc` file into the project (paste it in `/first_zynq_design` and add to project)
- Click on Create Block Design.
- Instantiate the Zynq PS and the AXI GPIO peripheral (right click on the canvas and select Add IP).
- Click on Run Block Automation. This will create the external connections for the DDR and FIXED_IO interfaces.
- Configure AXI_GPIO: Double-click on it, select All Outputs, and set GPIO Width to 4.
- Click on Run Connection Automation. Check both S_AXI and GPIO.
- Double click on ZYNQ PS:
  - ✓ Look at the I/O Peripherals list and Flash Memory Interfaces in the Zynq Block Design figure. Those are all the peripherals the Zynq-7000 chip can interface with. However, the ZYBO Board only contains a handful of them.
  - ✓ Load the `ZYBO_zynq_def.xml` file (Import XPS Settings). This will indicate the peripherals and interfaces the ZYBO Board can interface with. Note that a ✓ appears next to the available ones (Fig. 1). If this file is not loaded, the software application might not work as expected when using a certain peripheral (e.g. UART).
- Create the VHDL wrapper (Sources Window → right-click on the top-level system design → Create HDL Wrapper)
- We now need to manually indicate our 4 output ports:
  - ✓ Open the VHDL wrapper and modify the name of the 4-bit output port (usually it is `gpio_rtl_tri_o`) to `led`. Do the same in the port map statement.

RECRLAB@OU

- ✓ Open the XDC file and uncomment the lines for the LEDs. The port name is already `led` and it matches (as it must) the name provided in the top VHDL wrapper file.

- Synthesize your design. The first time, Vivado might undo your modifications to the VHDL wrapper file (this is a software bug). If it does, modify it for a second time and Synthesize again.
- Implement and generate the bitstream.
- Export hardware (include bitstream) and launch SDK (Software Development Kit).

1.B. CREATING A BLOCK DESIGN PROJECT IN VIVADO - ZYBO OR ZYBO Z7-10 BOARD (INSTALLED IN VIVADO)

- Create a new Vivado project: `first_zynq_design`.
  - ✓ Make sure the default language in VHDL, so that the system wrapper is created in VHDL.
  - ✓ At Default Part, go to Boards, and select the **Zybo** (or **Zybo Z7-10**) board.
- Once project is created, add the `ZYBO_Master.xdc` (or the `ZYBO-Z7-Master.xdc`) file into the project: paste it in `/first_zynq_design` and add to project.
- Click on Create Block Design.
- Instantiate the Zynq PS (right click on the canvas and select Add IP).
  - ✓ Click on Run Block Automation. This will create the external connections for the DDR and FIXED_IO interfaces.
- Instantiate the AXI GPIO peripheral (right click on the canvas and select 'Add IP').
  - ✓ Click on Run Connection Automation. Check both S_AXI and GPIO. On GPIO option → Select Board Part Interface, select ***leds_4bits***.
- Double click on ZYNQ PS:
  - ✓ Look at the I/O Peripherals list and Flash Memory interfaces in the Zynq Block Design (Fig. 1). The ZYBO Board can only interface with a handful of them (and they are marked by a ✓).

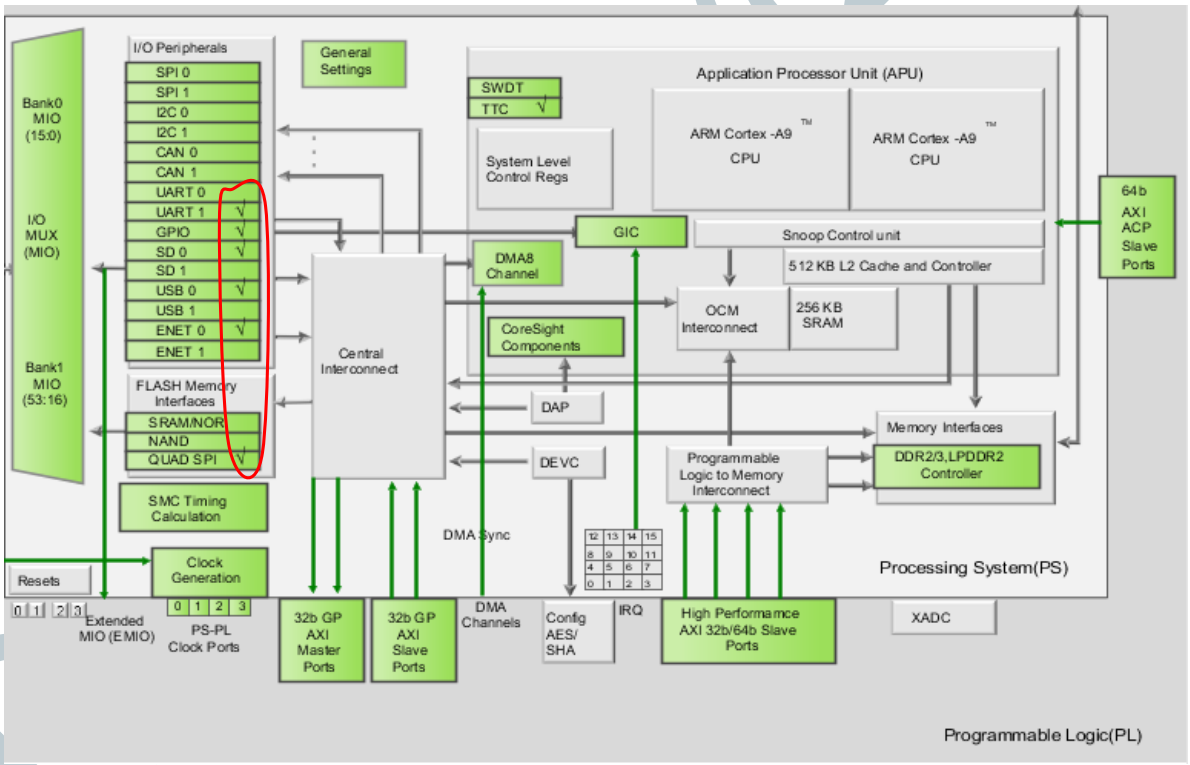

Figure 1. Zynq Block Design showing the I/O peripherals and FLASH Memory Interfaces supported by the ZYBO Board (marked with ✓)

- Create the VHDL wrapper (Sources Window → right-click on the top-level system design → Create HDL Wrapper)
- We included 4 output ports (***leds_4bits***), which now we need to manually indicate in the .xdc file:
  - ✓ Open the VHDL wrapper. You will see that the name of the 4-bit output port is `leds_4bits_tri_io`.
  - ✓ Open the XDC file and uncomment the lines for the 4 LEDs. Then, rename the 4 ports as `leds_4bits_tri_io[3..0]`. The name of the LED ports must match the name given in the top VHDL wrapper file.

- Synthesize your design.
- Implement and generate the bitstream.
- Export hardware (include bitstream) and launch SDK (Software Development Kit).

2. SOFTWARE APPLICATION IN SDK

- Create a new software project:
  - ✓ Select File → New → Application project.
  - ✓ Enter `LED_test` in the Project Name field. Keep Target Software → Language → C.
  - ✓ Select Empty Application.

- Load the software application:
  - ✓ Copy the `LED_test_tut_1Cp.c` (obtained from the Zynq Book Tutorials) on the folder: `first_zynq_design/first_zynq_design.sdk/LED_test/src`
  - ✓ Right-click on the `LED_test` project and select Refresh to update the files.

- The software routine controls the LEDs using software instructions by writing on a particular peripheral register (refer to the Zynq Book Tutorials for a comprehensive explanation).

- Build the software application. By default, SDK will build the software project automatically and create an output .elf file. This is the file to be loaded into the instruction memory.

- Test the software application and the hardware design on the ZYBO Board or ZYBO Z7-10 Board:
  - ✓ Connect your ZYBO Board to the USB port of your computer.
  - ✓ Download the bitstream on the PL: Xilinx Tools → Program FPGA.
  - ✓ Go to the SDK Terminal and click on Connect to a Serial Port. Configure the port with the same baud rate as the PS (11500 by default) and the proper COM port (available after the PL is programmed). This will allow us to receive messages from the ZYBO Board.
    - UART: This peripheral is connected to the PS. We can use the xil_printf instruction to print out messages. Feel free to modify the software by printing out messages. These messages will appear in the SDK Terminal.
  - ✓ Select the project `LED_test`. Right click and select Run As → Launch on Hardware (GDB).
    - You should see the LEDs blinking ($0110 \leftrightarrow 1001$) and the message ECE5736: GPIO output test on the SDK Terminal.
    - Sometimes, when trying the software for the first time, it might not work. Go to Run → Run Configurations: `LEDtest.elf` and uncheck: Run ps7_nit, ps7_post_config. Then select run the software again.

RECRLab@OU