# ECE 4730: Embedded Systems II

Project 3: Custom IP, multiplier

# Objectives

* Add board specification files to Vivado
* Use the Zynq processor in block design
* Implement hardware multiplication in a custom IP
* Access the IP via software

# Deliverables

* Instructor-approved multiplication IP 5
* Your edited C code 6
* Your edited Verilog Code 7
* Terminal output 7

Total: 25

# Procedure

Part 1- Zynq7 configurations

1. Download additional board specification files from:

<https://github.com/Digilent/vivado-boards/archive/master.zip>

1. Extract the .zip. Find the ‘New Boards’ directory and place it in C:\Xilinx\Vivado\data\boards\ board\_files. Now the board specs will be loaded upon starting Vivado.
2. Open Vivado now. Create a new project as before, except in the ‘Default Part’ window. Click on ‘Boards’ and select ‘Zybo Z7-10’. We will use the ‘Board’ tab from now on.
3. Create a Block design. Add the ‘Zynq7 Processing System’ IP. Run Block automation with the following configurations:

🗹 ‘Apply Board Preset’

1. Re-customize the processing system. In the ‘Peripheral I/O Pins’ tab, uncheck all pins except for ‘UART 1’.
2. Click on ‘Zynq Block Design’ tab. Notice that the ‘Application Processing Unit (APU)’ (the ARM processor) is connected to ‘UART 1’ through a ‘Central Interconnect’ block. Click ‘OK’.
3. The PS (processing system) is now ready.

Part 2- Custom IP Creation

Multiply IP specifications:

* Each register contains 32 bits.
* Register 0 is factor 1.
* Register 1 is factor 2.
* Register 2 is the product.

1. Go to the menu bar. Click on ‘Tools’ -> ‘Create and Package IP’. A window called ‘Create and Package New IP’ opens; click ‘Next’. Select ‘Create a new AXI4 peripheral’ and click ‘Next’.
2. A window will appear to name your IP. You can change the defaults as you like. Don’t change the IP location (we recommend naming the IP “multiply”).
3. The next window asks about the interface configs. Leave the default values:

Interface type: Lite

Interface mode: Slave

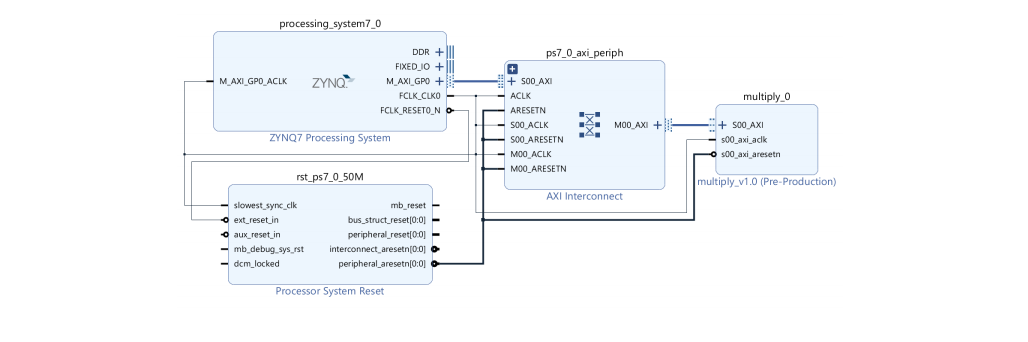
Data Width: 32

Number of Registers: 4

1. Select ‘Edit IP’ and click ‘Finish’. Another window will open to finish our custom IP.
2. The backbone of the IP will be generated, so we only need to add our custom functionality.
3. Examine the ‘Package IP’ tab.
4. Under ‘**Sources**’, expand ‘multiply\_v1\_0.v’ and open the ‘multiply\_v1\_0\_S00\_AXI.v’ file.
5. Examine the Verilog code. There will be questions on the functions. Comment out code that writes to ‘slv\_reg2’ (i.e. ‘slv\_reg2 <=’).
6. Insert the code found in “mult\_ip.txt” under the line ‘// Add user logic here’.
7. In the ‘Package IP’ tab, click on ‘Review and Package’; select ‘repackage’. This packages the IP to be imported by our original block design. You may decide to close the project.

Part 3- Finishing the Hardware

1. Add ‘multiply’ or ‘multiply\_v1.0’ to the block design. Run All Connection Automation. Regenerate the layout. Yours should resemble the following:



1. Create the HDL wrapper. Let Vivado manage wrapper and auto update. This creates the top module.
2. Generate the bitstream. There should be four critical messages. Ignore them.
3. Export the hardware and bitstream as done previously.

Part 4- SDK

1. Open the SDK; select ‘File’ -> ‘New’ -> ‘Application Project’. In the first window, name the project and leave the defaults. Click ‘Next’. Create a ‘Hello World!’ project. This step creates the necessary files for the PS on the ZYBO board.
2. Expand ‘multiply\_test’. Under ‘src’, open ‘helloworld.c’ and examine the generated code.
3. Edit the ‘helloworld.c’ source file to test your multiplier according to the specifications in Part 3. Print the values in the registers to the terminal using printf.

*Hint: xparameters.h, xil\_io.h, and multiply.h (or the name of your custom IP) should be included and will be useful. You will find functions to read to and write from registers in multiply.h*

1. Program the FPGA as done previously. Next is programming the processor with C code.
2. Under ‘multiply\_test’ expand ‘binaries’. Right click on ‘multiply\_test.elf’-> ‘Run As’ and click on ‘Launch on Hardware (GDB)’. This creates a configuration file and launches the code.
3. To see output, we need to connect to the board’s UART. Using either PuTTY or the ‘SDK Terminal’ inside of the SDK.
4. Discover the COM port associated with the board (use Window’s “Device Manager”). Enter that as the COM port.
5. If everything is correct, you will see text being printed to the serial console. Demonstrate your progress to the instructor.

Part 5- Your Turn

1. Why did we comment out code writing to slv\_reg2?
2. What do the following functions do?
   1. One
   2. Two
3. What is the purpose of the tmp reg in the Verilog code? What happens if the register is removed?\*
4. There are two possible errors producing incorrect results from the IP. What are they?
5. Correct these two errors in your IP, editing the Verilog code, and demonstrating test cases in the C appliction.
6. Thoughts: create an external port in the multiply IP.