# ECE 4730: Embedded Systems II

Project 2: Xilinx SDK

# Objectives

* Use the master.xdc
* Familiarity with Vivado Block Design
* MicroBlaze Softcore processor experience
* AXI GPIO experience
* Familiarity with Xilinx SDK
* Embedded programming in C
* Console i/o for FPGA

# Deliverables

* Instructor/TA-approved counter 5
* Your edited master.xdc file 3
* C code for the button-controlled counter 5
* Answers to questions 5
* Instructor/TA-approved button-controlled counter 7

Total: 25

# Procedure

Part 1- Block Design

1. Create this project like you created the previous one, only don’t add/create any source files.
2. Under ‘**Flow Navigator**’->’IP Integrator’, click on ‘Create Block Design’. Name your design in the popup and click ‘OK’.
3. Right click within the newly opened Block Diagram and select ‘Add IP’. Search ‘MicroBlaze’ and add it from the menu. Run Block Automation. Select the following configurations and click ‘OK’.

Preset: None

Local Memory: 64KB

Local Memory ECC: None

Debug Module: Debug & UART

Peripheral AXI Port: Enabled

𝤿 Interrupt Controller

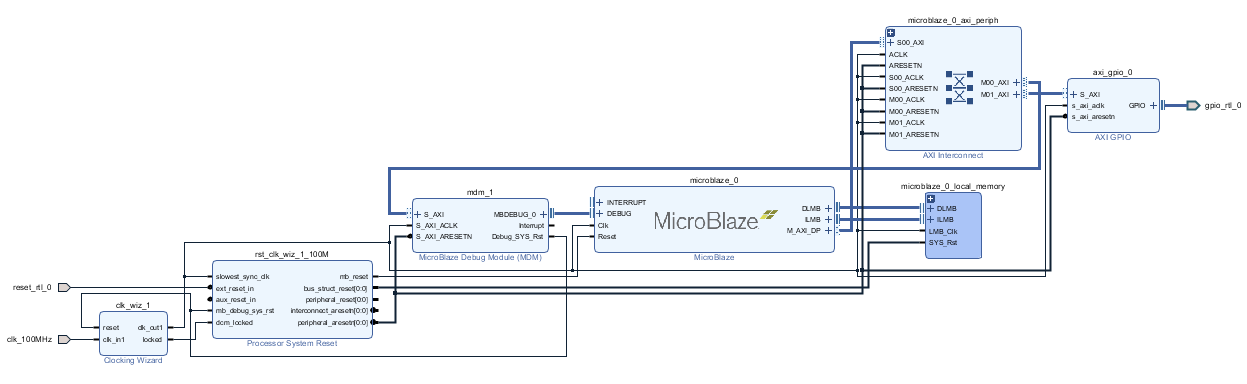
Clock Connection: New Clocking Wizard (100 MHz)

1. Run Connection Automation. Verify ‘All Automation’ is checked and click OK.
2. Double click on the ‘clk\_wiz\_1’ block to customize it. Under ‘Clocking Options’, change the ‘Source’ of the ‘Primary Input Clock’ from ‘Differential clock capable pin’ to ‘Single ended clock capable pin’. Click ‘OK’.
3. Now that the processor is set up, we need GPIO blocks to interact with the ZYBO inputs and outputs. Add ‘AXI GPIO’ to the design.
4. Customize the new block and set the configurations to the following:

🗹 All Outputs

GPIO Width: 4

Leave all other options unchanged for now.

1. Run Connection Automation, selecting ‘All’ again.
2. Right-click on a blank area to open the menu and select ‘Regenerate Layout’. The image should resemble the following:  
   
3. The GPIO block can be renamed under ‘**Block Properties**’ (found below ‘**Sources**’). Rename it ‘led’. Rename the output port ‘led’ as well.
4. Add a ‘Constant’ IP. Customize the block with the following:

Width: 1

Val: 1

Rename it “VCC”. Delete ‘reset\_rtl\_0’ and connect the VCC constant to the “ext\_reset\_in” pin on the “rst\_clk\_wiz\_1\_100M” IP.

1. The variable names of our ports and external interfaces are listed in the Design tab. In the following steps we will use those to modify the master.xdc file.

Part 2- the master.xdc

1. Download the master.xdc files from the following link:

<https://github.com/Digilent/digilent-xdc>

Place Zybo-Z7-Master.xdc in an easily-accessible location (it will be re-used in most labs) and then copy it into the current project subdirectory.

1. Find the “LEDs” section in the master file. Note that the variable name for the bus (“led”) follows the “get\_ports” command. Replace the corresponding bus name “led” with the name generated by your design “led\_tri\_o”. Uncomment those 4 lines.
2. Update the variable name in the “Clock signal” section to match your design (“clk\_100MHz”). (Note that the period for this clock signal is 8. Clock frequency is calculated by: 1GHz/period = 125MHz). Uncomment those 2 lines.
3. Add the new constraints file to your sources as done previously.
4. Open the context menu in the diagram and select ‘Validate Design’
5. Under ‘**Sources**->’Design Sources’, right click on the project and select ‘Create HDL wrapper’. Let Vivado manage wrapper and auto update and click ‘OK’.
6. Generate the Bitstream as done previously. An error may occur with the clock frequency. You can change the period in the XDC to correct it. Once generated, select ‘File’->’Export’->’Export Hardware’; include the bitstream as well.

Part 3- Bitstream and Programming

1. Launch the SDK (Software Development Kit) from the menu under ‘File’. Make the location local.
2. Once the SDK opens, select ‘File’ -> ‘New’ -> ‘Application Project’. Give a project name (eg. counter\_sw). Click ‘Next’ and select the ‘Empty Application’. Click ‘Finish’.
3. Verify you have counter\_sw, counter\_sw\_bsp, and led\_wrapper\_hw\_platform\_0 in your ‘Project Explorer’.
4. Find ‘project2a.c’ and save it in your project2 directory. Read through it and take note of the included header files. They will be helpful and informative.
5. Expand the ‘counter\_sw’ section, right click on ‘src’ and select ‘import’. In the popup, expand ‘General’, select ‘File System’, and click ‘Next’. Browse to import ‘project2a.c’. Click ‘Finish’.
6. The c program is ready to be compiled and placed on the FPGA. Program the hardware by selecting either the  icon, or ‘Xilinx Tools’ -> ‘Program FPGA’.
7. Select ‘Program’ with the defaults, take note of the warning (PS means processing system), then click ‘OK’.

The warning is meant to warn you that there is no ARM processor in the design. We already knew that because we programmed the FPGA with the MicroBlaze.

1. Now that the hardware is programmed, the C code can be run on it. Click ‘Run’ and select ‘Run Configuration’.
2. In the popup, double-click ‘Xilinx C/C++ Application (GDB)’ to create a configuration file. Name it. Go to the application tab. Browse for the project name ‘counter\_sw’. Set these configurations

𝤿 ‘Stop at main’

🗹 MicroBlaze

🗹 ‘Reset Processor’

𝤿 ‘Stop at program entry’

Click ‘Apply’ and then ‘Run’ to program the MicroBlaze processor.

1. To view the printed output, select ‘Xilinx’ -> ‘XSCT Console’. In the new console, type the commands ‘connect’, ‘target 5’, and finally ‘jtagterminal -start’.

Tip: ‘stop’ in the XSCT console will stop the program. Then you can re-run it.

1. If you can see the LEDs counting in binary from 0 to 15, then you did it. Good job!

Part 4- Your Turn

1. Why is the MicroBlaze called a soft processor?
2. Why did we need to change from a differential to single-input clock?
3. Block Automation Questions.
4. What does Connection Automation do?
5. AXI interface ?’s
6. ‘Reset\_rtl\_0\_0’: Why did we connect it to VCC? (the pin it’s connected to may help)
   1. What else could we have connected it to so we could reset the IP?
7. Why do we need the hardware exported before programming in the Xilinx SDK?
8. Describe the function of each of the following:
   1. Counter\_sw
   2. Counter\_sw\_bsp
   3. Led\_wrapper\_hw\_platform\_0
9. What does the while(1) expression do?
10. Why does the Xilinx SDK warn you that there’s no processing system (PS) in the design when the FPGA is programmed with the bitstream?
11. Add 8 bits of input to the GPIO block in your system. Connect them to the switches and push buttons on the Zybo. Use the SDK to implement the following pseudocode:

Var count

Switch (button)

0 -> count ++

1 -> count - -

2 -> count = switches

3 -> leds = count

Print(count)

1. Compare the value of counter in this project to the value in the previous project. Were they different? How? Why? How many clock cycles does it take to execute one iteration of the delay for-loop?
2. Which implementation was easier between this and the previous project? What are the advantages and disadvantages of each?
3. \*\*Dice game?