MP1 Report

By: Nomaan Dossaji (ndossa2)

And

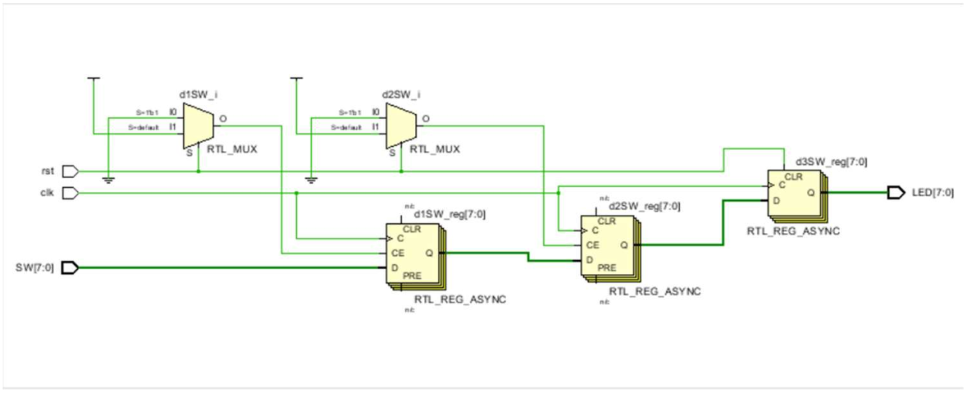
Jay Patel (jdpatel4)

Part 1

Assumptions

For part 1 we assumed that the reset button would black out all of the LEDs right away, instead of waiting 3 clock cycles since resets usually occur as soon as possible without any lag. However, the LEDs will turn back on after 3 clock cycles to resume normal functionality.

Block Diagram



Modules

We only needed to create one module for part 1. Our inputs were the clock, reset button, and the 8 switches. The outputs are the 8 LEDs. The purpose of this module is to reflect what the switches show after 3 clock cycles.

Design Process

We delayed the output by three clock cycles by adding three flip flops before outputting the value to the LEDs. To verify this was done correctly, we checked the simulation.

Post-Implementation Results

Resource	Utilization
LUT	1
FF	24
IO	18

	Power
Clocks	.001W
Signals	<.001W
Logic	<.001W
I/O	.006W

Setup		Hold		Pulse Width	
WNS	8.609ns	WHS	.171ns	WPWS	4.5ns
TNS	0 ns	THS	0 ns	TPWS	0
Failing Endpoints	0	Failing Endpoints	0	Failing Endpoints	0
Number of Endpoints	16	Number of Endpoints	16	Number of Endpoints	25

Difficulties/Bugs:

Our main difficulty for part 1 was downloading and using Vivado. Figuring out which version, how to setup the software, and connect it to the FPGA took us a while. The most difficult part was getting Vivado to program the FPGA. It almost took us an hour to figure it out. Once we had figured out the version, drivers, and software needed, we were able to get the part done pretty quickly.

What we learned

We learned how to setup basic functionality through Verilog with the FPGA. We learned how to use the Vivado software as well.

Vivado Version

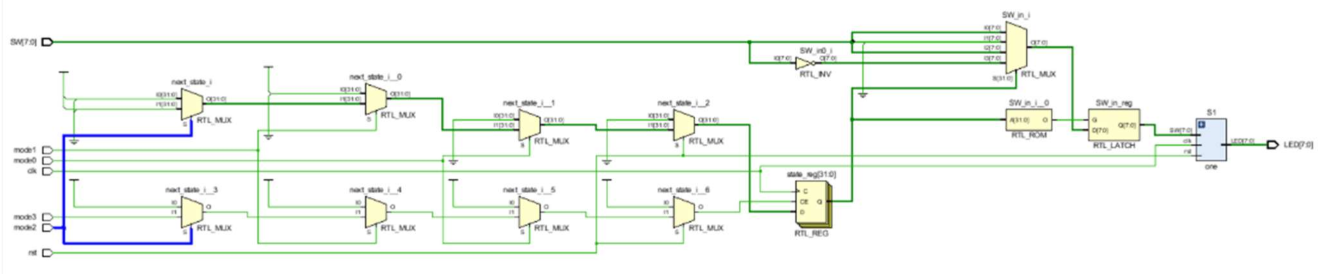
We used Vivado 2018.2 on our personal computers on Windows.

Part 2

Assumptions

We assumed there is a 3 clock cycle delay, similar to part 1. We also assumed that pressing two buttons at same time would make the hardware do the mode with lower priority first. We also assumed that pressing the reset makes the state machine go in mode 0 and would black out all the LEDs.

Block Diagram



Modules

We used the module from part1 as a submodule for part2 to simply output the LED configuration. We created an additional module to determine the mode, read the switches, and decide what the LED configuration should be. This module then passes the LED configuration to the module from part1.

Design Process

Determining the 3 cycle clock delay was done the same way part 1 was completed. We verified in the same way as well (with simulation). Along with this, we verified the modes were completed correctly by testing a various setup of switches and going through each mode to see if the LEDs were shifted in the correct way.

Post-Implementation Results

Resource	Utilization
LUT	9
FF	26
IO	22

	Power
Clocks	.001W
Signals	<.001W
Logic	<.001W
I/O	.009W
Static	.106W

Setup		Hold		Pulse Width	
WNS	8.202ns	WHS	.122 ns	WPWS	4.5ns
TNS	0 ns	THS	0 ns	TPWS	0
Failing Endpoints	0	Failing Endpoints	0	Failing Endpoints	0
Number of Endpoints	26	Number of Endpoints	26	Number of Endpoints	27

Difficulties/Bugs

After completing part 1, we had figured out how to setup the board and do some basic Verilog. We used what we had created in part one and built an additional module off that. Because we had completed part 1 beforehand, we did not encounter any bugs. The only difficulty we encountered was forgetting to add reset as a functionality.

What we learned

We learned to use state machines with part 2. We wrote System Verilog code to decide which mode should the FPGA change to when a button was pressed. We had had experience doing this through various previous classes such as ECE 411 and ECE 385.

Vivado Version

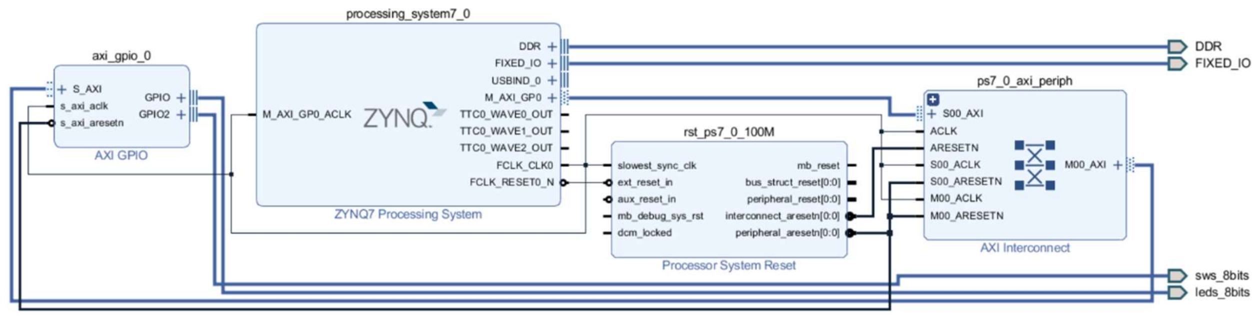
We used Vivado 2018.2 on our personal computers on Windows.

Part 3

Assumptions

For part 3 we assume that the changes in switches should be reflected without any delay. We also assume that the change in switch should not affect which mode is being shown and simply keeps going through the loop. Our final assumption is we are allowed to include libraries and use any functions that do not require external downloads.

Block Diagram



The diagram above shows our implementation for part c. For this part, we first instantiated the ZYNQ and AXI GPIO modules, and then connected them through the Vivado software. We then programmed our led_blkl.c file, which had the desired functionality, onto the ZYNQ7 Processing System.

Modules

For part 3 we were required to instantiate the AXI to GPIO and processor blocks. After that we coded in software. With C programming, we were able to read from the switches and write to the LEDs within the main function using GPIO functions.

Design Process

We were able to achieve the one second delay by using a function called usleep. usleep's parameters take the number of microseconds you want the board to sleep for. Since we want the board to have a one second break between each mode, we chose one million for the parameter.

Post-Implementation Results

Resource	Utilization
LUT	440
LUTRAM	60
FF	665
IO	16

	Power
Clocks	.003W
Signals	.002W
Logic	.001W
I/O	<.001W
PS7	1.529W
Static	.144W

Setup		Hold		Pulse Width	
WNS	4.234ns	WHS	.022 ns	WPWS	4.020ns
TNS	0 ns	THS	0 ns	TPWS	0
Failing Endpoints	0	Failing Endpoints	0	Failing Endpoints	0
Number of Endpoints	1420	Number of Endpoints	1420	Number of Endpoints	730

Difficulties/Bugs

We found this part to be straight forward when following the "Getting Started with Vivado – Part 2 (Video)" located on the course website. We did not encounter any bugs or difficulties because of this video and worked on our first try.

What we learned

Part 3 of the MP taught us how to read from the switches and write to the LED's through software. We were able to use GPIO functions to access the LEDs and switches.

Vivado Version

We used Vivado 2018.2 on our personal computers on Windows.