MP1 Report

By: Nachuan Wang(nachuan3)

And

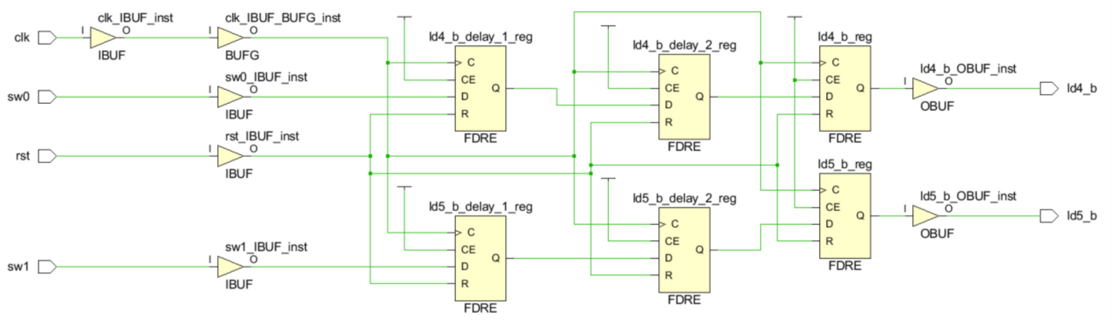
Liangyu Zhou(liangyu5)

Part A: Basic I/O

Assumptions:

For part one, we assume that the LEDs will be turned on after 3 clock cycles when set the switch. For reset, LEDs will immediately turn off after the synchronous reset is present.

Block Diagram:



Where FDRE stands for a single D-type flip-flop with clock enable and synchronous reset.

Entities/Modules:

We only have one module in this part, whose input is clk signal, synchronous reset and switch signal as control signal. The output of this module are two LEDs, which controlled by the switch signal.

Design process:

In RTL design, we firstly adopted 3 level flip flops to delay the switch signal for 3 clock cycles, then output on the LEDs. To simulate our design, we programmed our code into the Pynq board to test our design and proved to be correct.

Post-Implementation results:

Recourses	Utilization
FF	6
IO	6
BUFG	1

Power	Utilization
Clocks	0.001W
Signals	<0.001W
Logic	<0.001W
I/O	0.002W
PL static	0.106W

Timing	
Setup WNS	6.881ns

Setup total number of Endpoints	5
Hold WNS	0.256ns
Hold total number of Endpoints	4

Difficulties/Bugs:

The most challenging problem we encounter in this part is that we need to get familiar with Vivado and connected to device. Thanks to the tutorial video, we get through all the difficulties easily.

What we learnt from this

We learnt the design flow of FPGA based on Vivado including RTL design, simulation and constraint. And we also learnt how to assign some basic input and output port in Pynq board.

Vivado version:

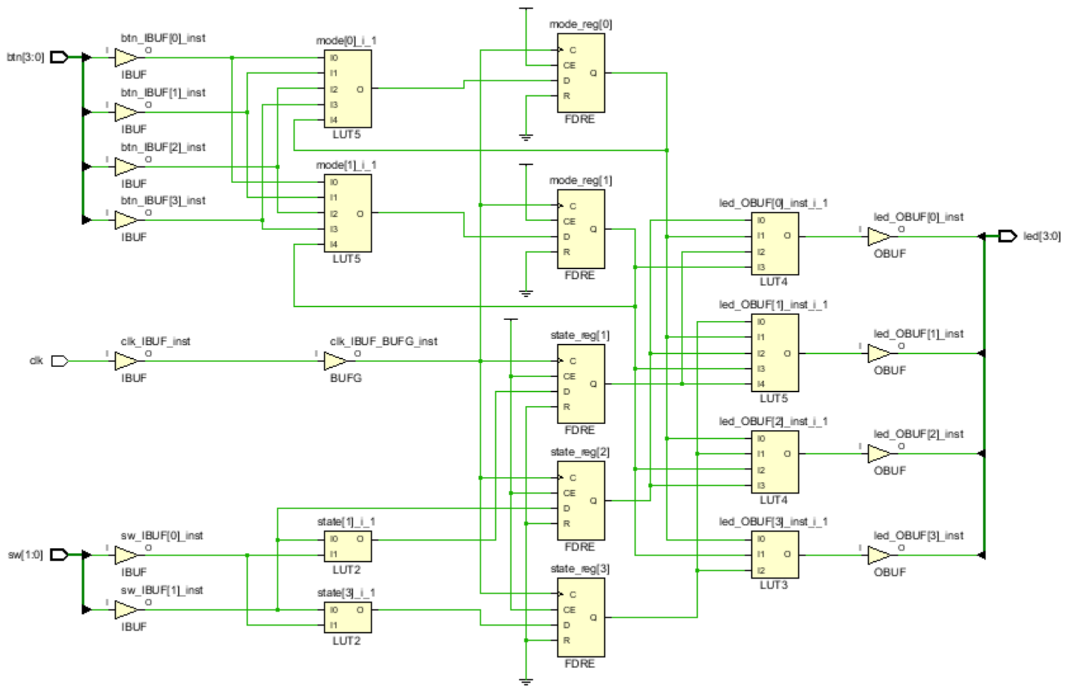
We used Vivado 2019.1 as tutorial video on our own personal computers with Windows.

Part B: Basic I/O with FSM

Assumptions:

For part B, we assume that the LEDs will be turned on immediately after the switch changes, as well as the reset signal. And we also adopted a synchronous reset as we did in part A. And we further assume that there is no priority among different buttons.

Block Diagram:



Entities/Modules:

We only have one module in this part, whose input is clk signal, synchronous reset, switch signal as control and button signal as mode. The output of this module are 4 LEDs, which controlled by the switch signal.

Design process:

In RTL design, we firstly designed the state machine by defining the state transfer using buttons. Secondly, we designed an always block to generate outputs according to the mode and state. Then we designed a testbench to simulate our design. And after verify our design, we burnt the bitstream into our board to show the actual result.

Post-Implementation results:

Resources	Utilization
FF	5
IO	11
BUFG	1
LUT	5

Power	Utilization
Clocks	0.001W
Signals	<0.001W
Logic	<0.001W
I/O	0.004W
PL static	0.106W

Timing	
Setup WNS	6.738ns
Setup total number of Endpoints	2
Hold WNS	0.264ns
Hold total number of Endpoints	2

Difficulties/Bugs:

The most challenging problem we encounter in this part is that we need to design a state machine as well as how to verify this design by testbench.

What we learnt from this

We learnt how to design an FSM and use it to control the output state of FPGA.

Vivado version:

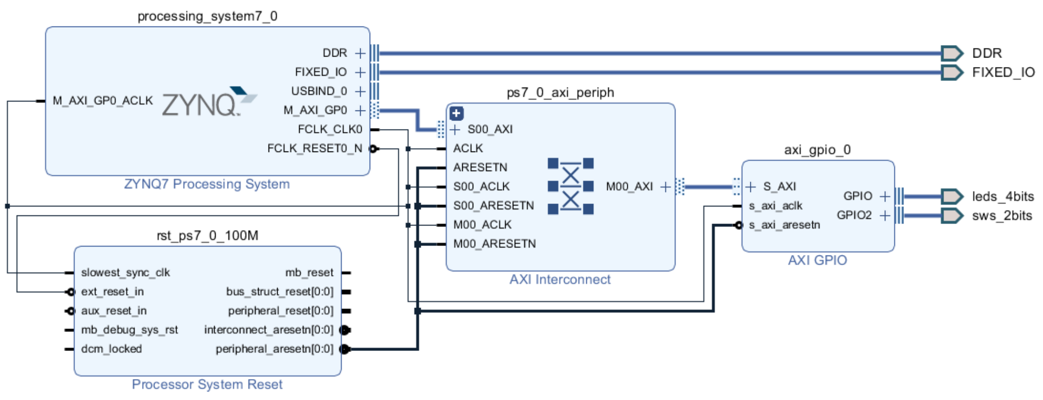
We used Vivado 2019.1 as tutorial video on out own personal computers with Windows.

Part C: Embedded SoC

Assumptions:

For part C, we actually did the same thing as in part B except for we implement the function on software side. So we share the same assumptions as in part B.

Block Diagram:



Entities/Modules:

We initialized a Zynq processor to build the software platform. Then we generate a AXI _GPIO block to drive the Led and Switch ports. And the system automatically generated the axi_periph block and clock block.

Design process:

In this part, we firstly create a block design to instantiate a Zynq processor. And we created a Axi_GPIO block and connect them to Led and Switch port. After that, we use vivado software to connect all the blocks. After building all the hardware infrastructure, we launch SDK to program the software. We designed a state machine function to illustrate state transfer, and in the main loop, we change circuit state by reading Switch port, and we output the Led by writing to Led port through AXI_GPIO. Finally we run the program on board and verify the result.

Post-Implementation results:

Resources	Utilization
LUT	434
LUTRAM	60
FF	591
IO	6
BUFG	1

Power	Utilization
Clocks	0.003W

Signals	0.002W
Logic	0.001W
I/O	0.001W
PS7	1.256
PL static	0.137W

Timing	
Setup WNS	3.305ns
Setup total number of Endpoints	1276
Hold WNS	0.071ns
Hold total number of Endpoints	1276

Difficulties/Bugs:

The most challenging problem we encounter in this part is that this is the first time we learnt how to build a functional Soc system on a Pynq board. I actually encountered several problems that deal with exporting hardware files to build SDK environment. I solved these problems by searching in Xilinx platform.

What we learnt from this

We learnt how to design an FSM on software and learnt how to build a Soc on Pynq board. And we also learnt how to connect led and switch with Zynq processor with axi_gpio module.

Vivado version:

We used Vivado 2019.1 as tutorial video on our own personal computers with Windows.