

San Jose State University  
Department of Computer Engineering

CMPE 140 Lab Report

Lab 6 Report

Title Processor Design (2): Processor Hardware Validation

Semester Fall

Date 03/17/2020

by


Name Thien Hoang  
(typed)

SID 012555873  
(typed)

Name Phat Le  
(typed)

SID 012067888  
(typed)

Lab Checkup Record

Week	Performed By (signature)	Checked By (signature)	Tasks Successfully Completed*	Tasks Partially Completed*	Tasks Failed or Not Performed*
1	 TH				

\* Detailed descriptions must be given in the report.

## I. Introduction

The purpose of this lab is to get familiar with the processor validation using FPGA devices. Specifically, the primary task is to perform hardware validation of the 32-bit single-cycle MIPS processor from the previous lab by drawing the diagram of the FPGA environment setup as well as filling up the test log table of the sample code.

## II. Design Methodology

Given the source code of the FPGA environment setup, the first task is to draw the top-level design diagram as shown in *Figure 1* below. According to the diagram, most of the outputs of the MIPS processor are connected through a multiplexer which is selected by actual switches to display in the 7-segments LEDs.

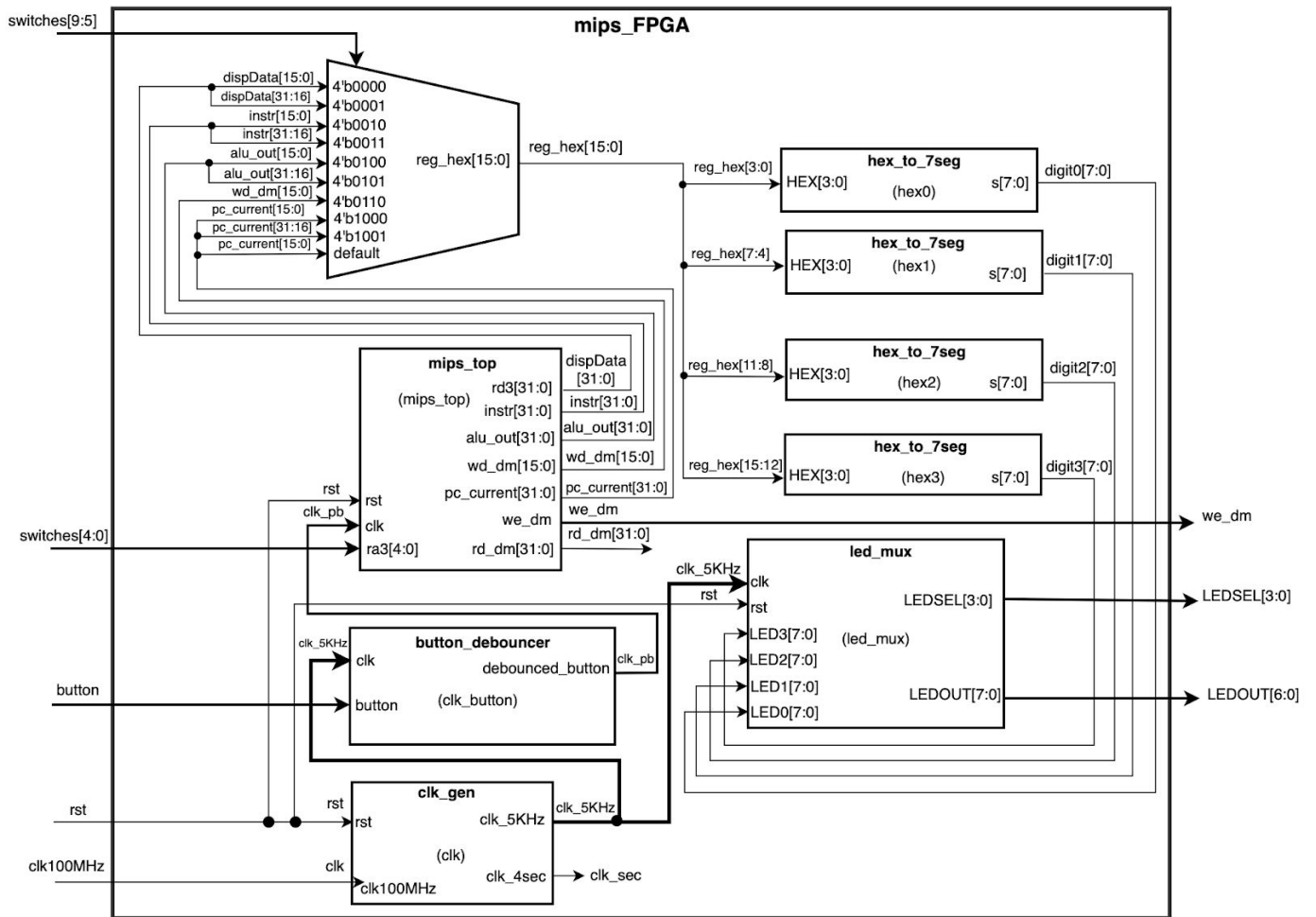


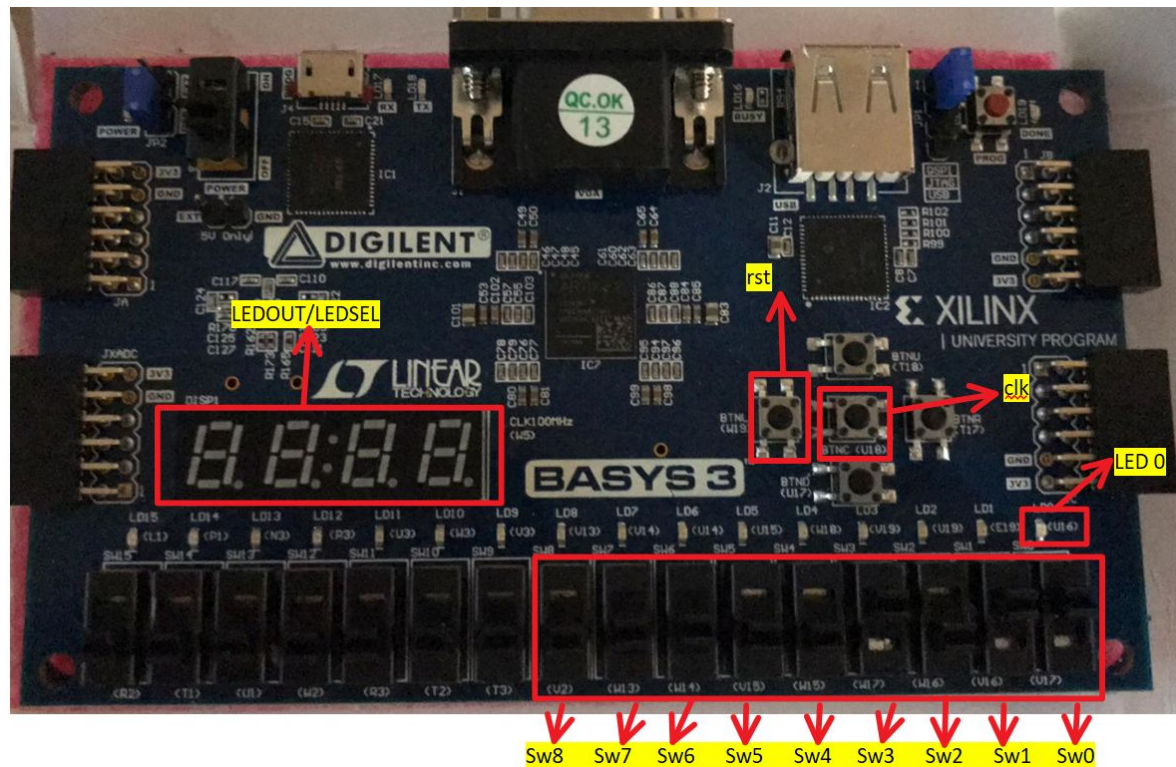
Figure 1: Top-level mips\_FPGA diagram

Another task is to record the log table from the same ASM source code from Assignment 2 and 5. Unlike the table from Assignment 2, the PC address value begins at 0 instead of 3000. Furthermore, the machine code at the address 18 and 20 are different from the previous

assignment as shown in *Table 1* below. Since the given FPGA environment does not design to directly read the data from the data memory, the test log table for this assignment will not include the memory content columns.

### III. FPGA Validation

For the FPGA validation, the observation of the constraint file was necessary to navigate and label the input switches and buttons as well as the output LEDs. Specifically, according to *Figure 2*, the center button was used as the manual clock (clk), and the left button as the reset signal (rst). Switches 0 to 4 were used to select the read address from the register file. Switches 5 to 8 were used as a select signal of the multiplexer to decide the type output display to the LEDs. The selection is decoded in the way that switch 5 would decide the display of the lower higher half word of value. Switches 6 to 8 would respectively decide the output of signals such as disp\_data, alu\_out, instr, wd\_dm, and pc\_current. Since the system design was not able to display the value at a specific address from data memory, we could only view the value when the write signal was enabled ( $we\_dm = 1$ ), which is when the LED0 was lit. Overall, the FPGA validation process was successful as outputs matched the expected values from the simulation waveforms.



*Figure 2: Digilent Basys 3 FPGA Board environment setup*

#### **IV. Conclusion**

In sum, We are getting a lot of knowledge in this lab such as displaying multiple values on the 7-segment LEDs using Basys 3 FPGA switches as well as implementing a toggle using a switch so that it can display both lower and higher output values. This lab also helps understand the hierarchical design, and it also gives us a lot of valuable information to work in the next lab.

#### **V. Successful Task**

- We were able to create a block diagram that illustrates the environment validation setup.
- We were able to create a validation record table that verifies the execution of each instruction and compares it with the log file in Assignment 2.

## VI. APPENDIX

mipstest.asm				
<pre> # mipstest.asm # Test the following MIPS instructions. # add, sub, and, or, slt, addi, lw, sw, beq, j #      Assembly      Description      Address      Machine main:  addi \$2, \$0, 5 # initialize \$2 = 5      3000      20020005       addi \$3, \$0, 12      # initialize \$3 = 12      3004      2003000c       addi \$7, \$3, -9      # initialize \$7 = 3      3008      2067fff7       or   \$4, \$7, \$2      # \$4 &lt;= 3 or 5 = 7      300c      00e22025       and  \$5, \$3, \$4      # \$5 &lt;= 12 and 7 = 4      3010      00642824       add  \$5, \$5, \$4      # \$5 = 4 + 7 = 11      3014      00a42820       beq  \$5, \$7, end      # shouldn't be taken      3018      10a7000a       slt  \$4, \$3, \$4      # \$4 = 12 &lt; 7 = 0      301c      0064202a       beq  \$4, \$0, around   # should be taken      3020      10800001       addi \$5, \$0, 0 # shouldn't execute      3024      20050000 around: slt  \$4, \$7, \$2      # \$4 = 3 &lt; 5 = 1      3028      00e2202a       add  \$7, \$4, \$5      # \$7 = 1 + 11 = 12      302c      00853820       sub  \$7, \$7, \$2      # \$7 = 12 - 5 = 7      3030      00e23822       sw   \$7, 68(\$3)      # [80] = 7      3034      ac670044       lw   \$2, 80(\$0)      # \$2 = [80] = 7      3038      8c020050       j    end              # should be taken      303c      08000c11       addi \$2, \$0, 1 # shouldn't execute      3040      20020001 end:    sw   \$2, 84(\$0)      # write adr 84 = 7      3044      ac020054       j    main              # go back to beginning      3048      08000c00 </pre>				

Table 1. Validation Record Table MIPS Processor

Adr	Expected Machine Code	Actual Machine Code	PC	Register				
				\$v0(\$2)	\$v1(\$3)	\$a0(\$4)	\$a1(\$5)	\$a3(\$7)
00	2002005	0x2002005	00004	00000005	0	0	0	0
04	2003000c	0x2003000c	00008	00000005	0000000c	0	0	0
08	2067fff7	0x2067fff7	0000c	00000005	0000000c	0	0	00000003
0c	00e22025	0x00e22025	00010	00000005	0000000c	00000007	0	00000003
10	00642824	0x00642824	00014	00000005	0000000c	00000007	00000004	00000003
14	00a42820	0x00a42820	00018	00000005	0000000c	00000007	0000000b	00000003
18	10a7000a	0x10e5000a	0001c	00000005	0000000c	00000007	0000000b	00000003
1c	0064202a	0x0064202a	00020	00000005	0000000c	00000000	0000000b	00000003
20	10800001	0x10040001	00028	00000005	0000000c	00000000	0000000b	00000003

24	20050000	0x20050000	SKIP/ NO EXECUTION					
28	00e2202a	0x00e2202a	0002c	00000005	0000000c	00000001	0000000b	00000003
2c	00853820	0x00853820	00030	00000005	0000000c	00000001	0000000b	0000000c
30	00e23822	0x00e23822	00034	00000005	0000000c	00000001	0000000b	00000007
34	ac670044	0xac670044	00038	00000005	0000000c	00000001	0000000b	00000007
38	8c020050	0x8c020050	0003c	00000007	0000000c	00000001	0000000b	00000007
3c	08000c11	0x08000c11	00034	00000007	0000000c	00000001	0000000b	00000007
40	20020001	0x20020001	SKIP/ NO EXECUTION					
44	ac020054	0xac020054	00048	00000007	0000000c	00000001	0000000b	00000007
48	08000c00	0x08000c00	00000	00000007	0000000c	00000001	0000000b	00000007