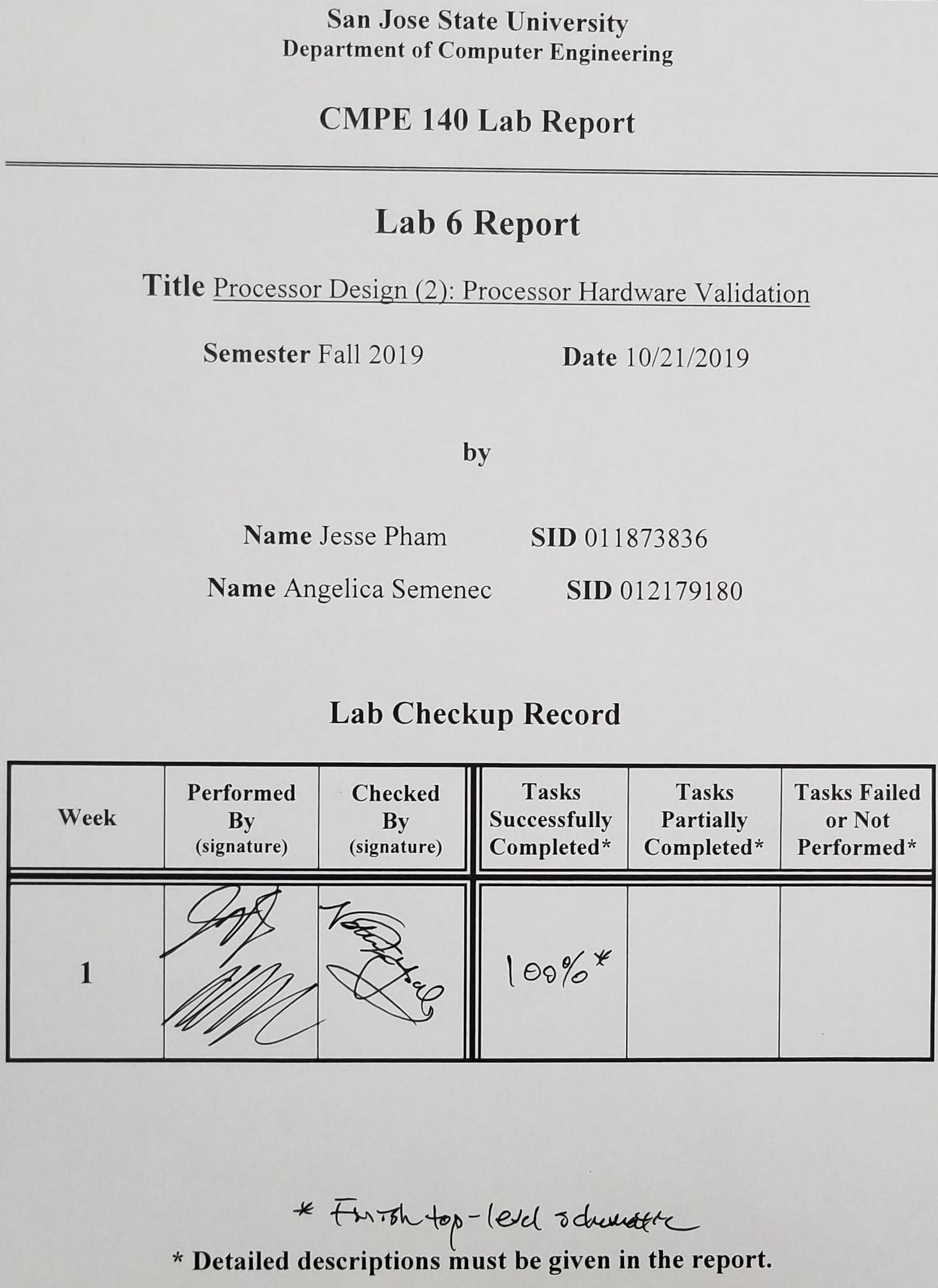
**Purpose**

The purpose of this lab is to validate the MIPS single-cycle processor through FPGA validation using the provided source files. This lab will utilize the single-cycle processor Verilog source code provided in assignment 5.

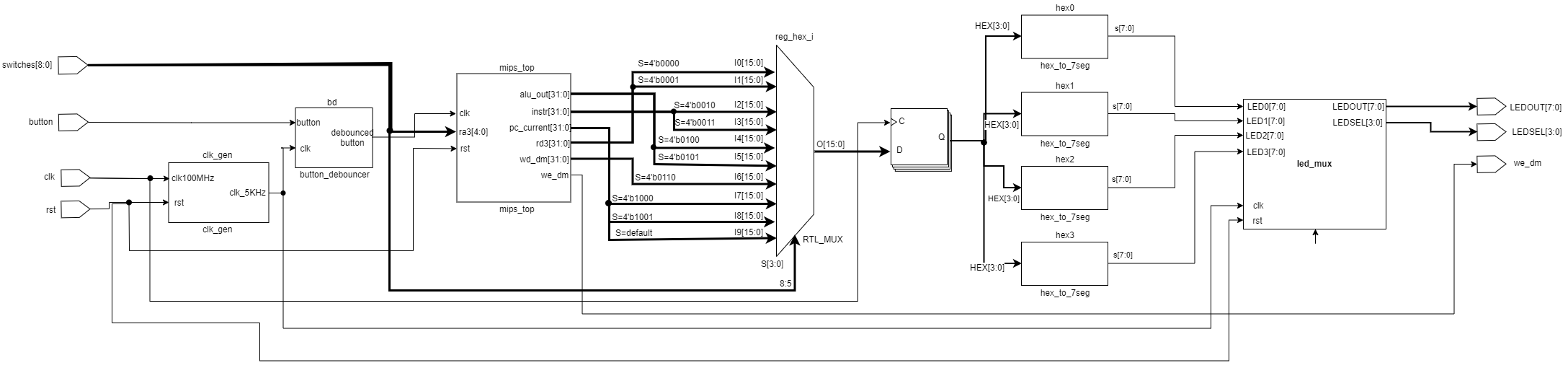
**Design Methodology**

The source code for the processor design and input MIPS instructions as hexadecimal values were provided in the given lab. The processor was assembled in Lab 5. In this lab, design sources for the FPGA were given to be incorporated into the existing design along with the constraints file that directed the on-board peripherals to the specific signals generated by the code. The 7-segment display was used to show the 4-byte values corresponding of the input instructions, program counter, and registers. The switches were used to display the register data, the instruction, the alu output, the data to be written to memory, and the current program counter value. One of the on-board LEDs was used to display when a write to memory instruction was fetched and decoded. A clock generator module is used to create the clock signal, which is passed to a button debouncer. When the corresponding button is pressed (clock or reset), the program will either reset the program counter to the first instruction or the clock will fetch the next instruction.

**Tasks Successfully Completed**

* Created the FPGA program using the given Verilog modules
* Successfully generated the bitstream and program the FPGA board
* Validated the program’s output by comparing it to the expected output generated in the Lab 2 Test Log

**Diagrams**

****

*Figure 1: The above diagram shows the top-level module, which connects all of the devices to their respective signals on the Basys3 FPGA board.*

**Validation Table**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adr | Expected Machine Code | Actual Machine Code | PC | Registers | | | | | Memory Content | |
| $v0 | $v1 | $a0 | $a1 | $a3 | [80] | [84] |
| 00 | 20020005 | 20020005 | 0x0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 04 | 2003000c | 2003000c | 0x4 | 5 | c | 0 | 0 | 0 | 0 | 0 |
| 08 | 2067fff7 | 2067fff7 | 0x8 | 5 | c | 0 | 0 | 3 | 0 | 0 |
| 0c | 00e22025 | 00e22025 | 0xc | 5 | c | 7 | 0 | 3 | 0 | 0 |
| 10 | 00642824 | 00642824 | 0x10 | 5 | c | 7 | 4 | 3 | 0 | 0 |
| 14 | 00a42820 | 00a42820 | 0x14 | 5 | c | 7 | b | 3 | 0 | 0 |
| 18 | 10a7000a | 10e5000a | 0x18 | 5 | c | 7 | b | 3 | 0 | 0 |
| 1c | 0064202a | 0064202a | 0x1c | 5 | c | 0 | b | 3 | 0 | 0 |
| 20 | 10800001 | 10040001 | 0x20 | 5 | c | 0 | b | 3 | 0 | 0 |
| 24 | 20050000 | 20050000 |  |  |  |  |  |  |  |  |
| 28 | 00e2202a | 00e2202a | 0x28 | 5 | c | 1 | b | 3 | 0 | 0 |
| 2c | 00853820 | 00853820 | 0x2c | 5 | c | 1 | b | c | 0 | 0 |
| 30 | 00e23822 | 00e23822 | 0x30 | 5 | c | 1 | b | 7 | 0 | 0 |
| 34 | ac670044 | ac670044 | 0x34 | 5 | c | 1 | b | 7 | 7 | 0 |
| 38 | 8c020050 | 8c020050 | 0x38 | 7 | c | 1 | b | 7 | 7 | 0 |
| 3c | 08000011 | 08000011 | 0x3c | 7 | c | 1 | b | 7 | 7 | 0 |
| 40 | 20020001 | 20020001 |  |  |  |  |  |  |  |  |
| 44 | ac020054 | ac020054 | 0x44 | 7 | c | 1 | b | 7 | 7 | 7 |
| 48 | 08000000 | 08000000 | 0x48 | 7 | c | 1 | b | 7 | 7 | 7 |

**Test Log From Assignment 2**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Adr | Expected Machine Code | Actual Machine Code | PC | Registers | | | | | Memory Content | |
| $v0 | $v1 | $a0 | $a1 | $a3 | [80] | [84] |
| 400000 | 20020005 | 20020005 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 400004 | 2003000c | 2003000c | 4 | 5 | c | 0 | 0 | 0 | 0 | 0 |
| 400008 | 2067fff7 | 2067fff7 | 8 | 5 | c | 0 | 0 | 3 | 0 | 0 |
| 40000c | 00e22025 | 00e22025 | c | 5 | c | 7 | 0 | 3 | 0 | 0 |
| 400010 | 00642824 | 00642824 | 10 | 5 | c | 7 | 4 | 3 | 0 | 0 |
| 400014 | 00a42820 | 00a42820 | 14 | 5 | c | 7 | b | 3 | 0 | 0 |
| 400018 | 10a7000a | 10a7000a | 18 | 5 | c | 7 | b | 3 | 0 | 0 |
| 40001c | 0064202a | 0064202a | 1c | 5 | c | 0 | b | 3 | 0 | 0 |
| 400020 | 10800001 | 10800001 | 20 | 5 | c | 0 | b | 3 | 0 | 0 |
| 400024 | 20050000 | 20050000 | - | - | - | - | - | - | - | - |
| 400028 | 00e2202a | 00e2202a | 28 | 5 | c | 1 | b | 3 | 0 | 0 |
| 40002c | 00853820 | 00853820 | 2c | 5 | c | 1 | b | c | 0 | 0 |
| 400030 | 00e23822 | 00e23822 | 30 | 5 | c | 1 | b | 7 | 0 | 0 |
| 400034 | ac670044 | ac670044 | 34 | 5 | c | 1 | b | 7 | 7 | 0 |
| 400038 | 8c020050 | 8c020050 | 38 | 7 | c | 1 | b | 7 | 7 | 0 |
| 40003c | 08100011 | 08000c11 | 3c | 7 | c | 1 | b | 7 | 7 | 0 |
| 400040 | 20020001 | 20020001 | - | - | - | - | - | - | - | - |
| 400044 | ac020054 | ac020054 | 44 | 7 | c | 1 | b | 7 | 7 | 7 |
| 400048 | 08100000 | 08000c00 | 48 | 7 | c | 1 | b | 7 | 7 | 7 |

**Discussion/Conclusion**

When the program was initially stepped through on the FPGA board, the value of register $a1 displayed the value “8” after the instruction at pc value 0x14 was executed. This was due to an error in the hex to led file where the display for the letter “b” and the number “8” were set to the same value. This code was adjusted so that the resulting hex value displayed would show a lower-case “b” rather than an upper-case “B”, which resembled the number “8”.

After the program fetched and decoded a write to memory instruction, one of the on-board LEDs would light up to signal that that instruction was active. The alu\_out shows the memory address to be written to and the wd\_dm shows the data that is to be written to memory. This would be checked before the instruction was clocked, unlike the values of the register files. The input read addresses and output data corresponding to the addresses were asynchronous to the register file’s clock signal and so the value of wd\_dm would be available before the instruction was finished executing.

Each of the register values could be checked after the execution of the instruction occurred. Only the registers listed in the above log tables were checked for the appropriate value and then validated using the Test Log from Lab 2.

This lab provided an understanding of how the registers values and other necessary values could be checked to verify that the program was functioning properly in the single-cycle MIPS architecture.