Ceng 342

|  |  |
| --- | --- |
| to: | preston cook |
| from: | david langerman, samuel patzer |
| subject: | lab 7 |
| date: | April 30, 2016 |
| cc: | zhao, yanxiao |
|  |  |

Intro: This lab taught us to build a simple MIPS processor using VHDL code. The processor would be implemented with a finite state machine with data path. The user can input an instruction using the slide switches 8 bits at a time. When 4 are loaded, the processor executes the instruction. Subsequently, the result is displayed based on 4 inputs from the slide switches. This will either be the current value of the program counter, or the contents of one of the 32 registers.

Design: The design that we implemented uses the de-bounced button, and an ALU as components to the processor. The ALU shares a clock with the mpu, and works as a separate finite state machine. The mpu first loads 4 sets of 8 bits as an instruction. Then, the mpu passes the instruction to the ALU, which then executes whatever the instruction is. A register file defined as a 32 length array of 32 bit std\_logic\_vectors is common to all components. When the ALU finishes with an instruction, a done bit is set, and the ALU returns to its idle state. The MPU then sits waiting for the user to input which register to show the contents of. When the user enters the register to display, the output is displayed 8 bits at a time on the LEDs.

Testing and Results:

We tested each of the instructions using the following commands:

Add – 00000001 01001010 0101000 00100000

Sub – 00000001 01001010 0101000 00100010

And – 00000001 01001010 0101000 00100100

Or – 00000001 01001010 0101000 00100101

Beq – 00010001 01001010 0000000 00000000

Conclusion:

We observed a fully functional simple MIPS processor that was able to save any result to a register. When we first got to the lab we were confused on how to set the registers so that we could test the instructions, we soon figured out how to set the registers and were on our way testing. After though testing of the processor we deemed it worthy and we knighted as Sir MIPS.

Appendix A- Code

Appendix B – Testbench

Appendix C – Test Images

Appendix D—Constraints