EE 443
Digital and Computer Analysis and Design Laboratory

Lab *#8*: *The Complete Data Path*
04/2813

Prepared by:

Matt Van Veldhuizen

Lab Participation

Provide the percent participation in lab by each lab partner (normally, 50%, 50% for 2 partners):

Lab member name	**Percent participation**
Matt Van Veldhuizen	50%
Karl Ott	50%

Does your solution work the way it's supposed to work? ☐YES ☐NO[1]

[1]If your answer is NO, please explain in your report.

Instructor/TA comments and grading:

Change Figures!!!!!

1. Objective and Background

This point of this lab was to take all of the components that were created in the past labs and complete the data path for the 16-bit RISC style microprocessor.

2. Equipment

Altera Quratus II

3. Procedure

To complete the data path two more components where needed, the PC (program counter) register and CONTROL module. The PC was designed to have three control signals, CLK (clock), EN (enable), and RST (to reset the PC to 0). In addition the PC had a 16-bit input and output. The Control module was used to look at the operation code and function code within each instruction and generate the appropriate signals a table was made to list the control signals with their names, a description and the instructions in which they are active can be seen in Table 2. Once these last two components were created the other components from the past labs were included within the project. The next step was to design a hierarchical organization of the block diagram for the data path, this sketch can be seen in Figure 1. After sketching the diagram the next step was to generate symbols for all of the components and make a Block Diagram connecting each component together as described in the sketch. This can be seen in Figure 2. The next step was to hard code some values in the data path. First of which was register 0 ($0) which should always read zero. The other values that were hard coded where in the data memory component, where some data was placed at the address 0x100 and 0x102. The last bit that was hard coded was in the instruction memory component. Where a program was to hard coded. This program was to move the data stored at addresses 0x100 and 0x102 to some registers. Perform any arithmetic operation on the data and place that result into register R1. Store the data from R1 into the memory location 0x104. Then read that data from R1 so that it shows up on output. Lastly add a couple more instructions showing off the processor. A complete list of all of the actions can be found in Table 1. Once all of this was done the program was compiled and tested.

4. Results

The chip resources used for the entire data path are 1% of the total logic elements, 4% total pins and 2% total memory bits which can be seen in Figure 4 and had a maximum delay of 42.3ns. Meaning the clock frequency for this device would be between 23.4MHz 25.6MHz. The smallest device that this processor can run on is a Cyclone II EP2C5T144C6 . Which uses 5% of the total logic elements, 24% of the total pins, and 7% f the total memory bits of the chip resources which can be seen in Figure 5 and has a maximum delay of 46.8ns, which means that the clock frequencies are 21.3MHz and 22.1MHz.

5. Discussion and Questions

The final result of this lab was a working 16-bit RISC microprocessor. We had some trouble with the block diagram part of the lab. Once all of the components were connected, we found several bugs with the data memory component, but these were pretty easy to fix. Once these bugs were fixed, we found another bug where the data from the memory was not being written back to the registers. Once this was fixed everything magically works.

6. Conclusion

This lab was to take all of the components that were created in the past labs and complete the data path for the 16-bit RISC style microprocessor.

7. Attachments

(See my drawing)
Figure 1: Block Diagram Hierarchy Sketch

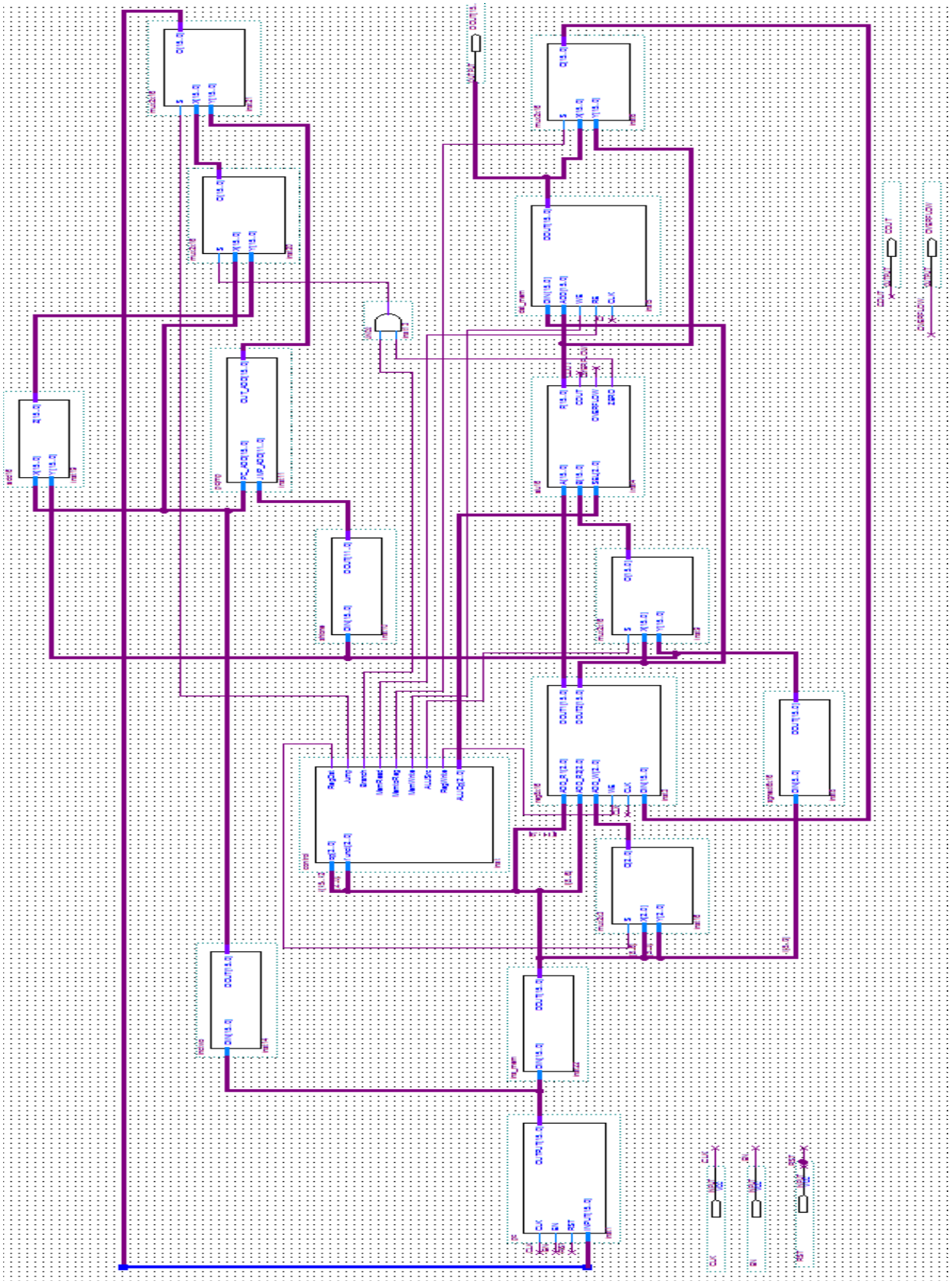

Figure 2: Block Diagram

Table 1: Program

Action	Instruction	Machine Code
R2 <- 0x100	lw R2, R7(0)	1011:111:010:000000
R3 <- 0x102	lw R3, R7(2)	1011:111:011:000010
R1 = R2 + R3	add R1, R2, R3	0000:010:011:001:010
0x104 <- R1	sw R1, R7(4)	1111:111:001:000100
print R1	lw R1, R7(4)	1011:111:001:000100
R4 = R2 - R3	sub R4, R2, R3	0000:010:011:100:110
if R1 == R4 branch	beq R1, R4, 1	0100:001:100:000001
R1 = R1 - R4	sub R1, R1, R4	0000:001:100:001:110
0x106 <- R1	sw R1, R7(6)	1111:111:001:000110

Flow Status	Successful - Wed May 01 19:25:26 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	lab8
Top-level Entity Name	m16dp
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	253 / 33,216 (< 1 %)
Total combinational functions	253 / 33,216 (< 1 %)
Dedicated logic registers	23 / 33,216 (< 1 %)
Total registers	23
Total pins	21 / 475 (4 %)
Total virtual pins	0
Total memory bits	8,208 / 483,840 (2 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 3: Data Path Chip Resources

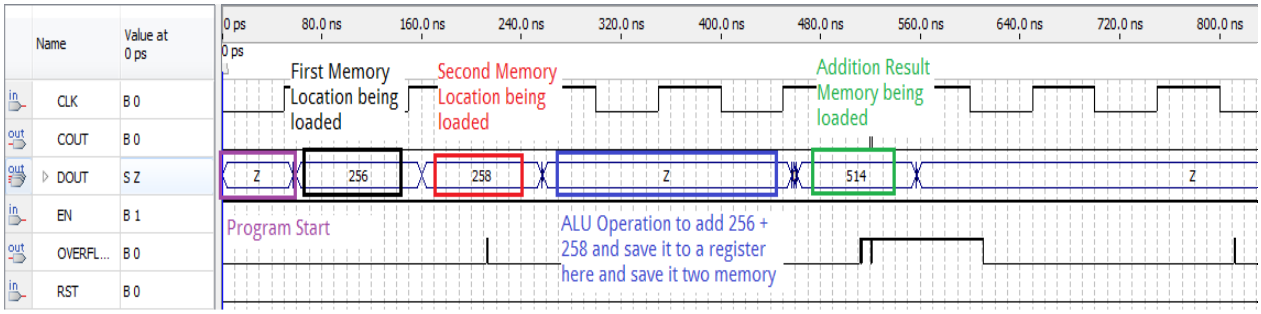

Figure 4: Data Path Simulation

Flow Status	Successful - Wed May 01 19:59:58 2013
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	lab8
Top-level Entity Name	m16dp
Family	Cyclone II
Total logic elements	253 / 4,608 (5 %)
Total combinational functions	253 / 4,608 (5 %)
Dedicated logic registers	23 / 4,608 (< 1 %)
Total registers	23
Total pins	21 / 89 (24 %)
Total virtual pins	0
Total memory bits	8,208 / 119,808 (7 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP2C5T144C6
Timing Models	Final

Figure 5: Data Path on Device EP2C5T144C6 Chip Resources

Table 2: Bus Control Signals for MIPS-16

Control Signal	**Description**	**Asserted (or Value) in:**
RegDst	Register Destination	Writebacks to registers
Branch	Branch Taken/Not Taken	When branching occurs
MemRead	Memory Read	Read from memory
MemtoReg	Memory Location to Register	Value from memory or ALU
ALUOp	ALU Operation Code	R and I type instructions
MemWrite	Memory Write	When writing to memory
ALUSrc	ALU Source Selection	R and I type instructions
RegWrite	Register Write	Writing into a register