Organization of Digital Computer Lab
EECS112L/CSE 132L

Assignment 4
Pipeline MIPS

prepared by: Team Stressed
Student name:
Mansi Tyagi
Student ID:
23334840

Student name:
Erik Henriquez
Student ID:
57374677

Student name:
Kevin Chau
Student ID:
76934313

Student name:
Steven Chow
Student ID:
70916812

Student name:
Paul Dao
Student ID:
30658761

EECS Department
Henry Samueli School of Engineering
University of California, Irvine

March 13, 2016

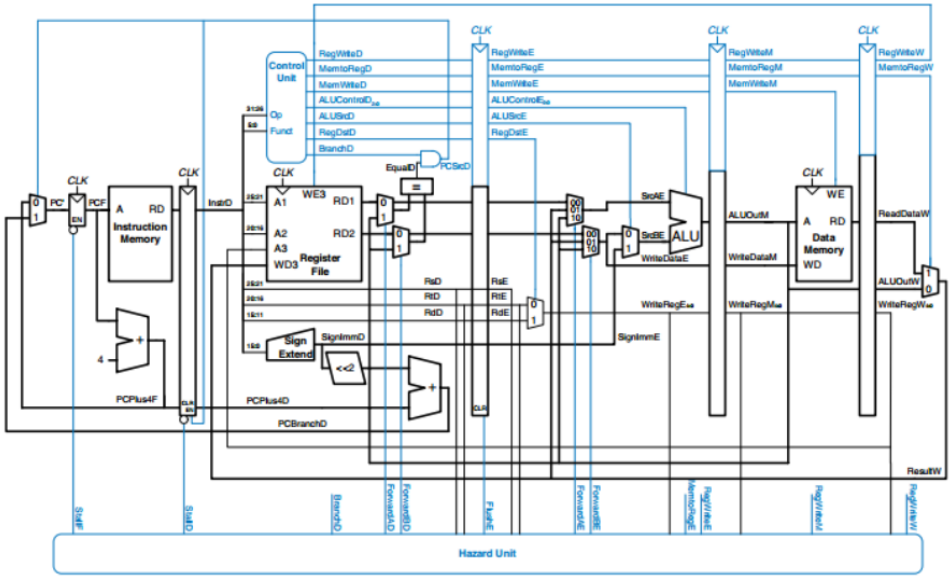
1 Summary of Processor Design

1.1 What We Learned

In this part of the assignment, we used our datapath from Lab 3, which implemented a complete single-cycle processor, and converted it into a pipelined processor. The processor we created pipelines the datapath and is capable of stalling and forwarding, and is capable of evaluating R, I, and J-Type instructions. In addition to simulating the processor, we also needed to synthesize it. We used Synopsis to determine the power consumption, area requirements, and clock frequency necessary to build our processor design. Although the use of Synopsis was initially very troublesome, we appreciated the fact that we were able to use tools that well-known corporations, like Intel and AMD, use to design and fabricate their own products.

1.2 Description

For our pipelined processor, we implemented all the components that we created for our single-cycle processor in lab 3. In order to create a pipelined processor, we created arrays which would function as the pipeline registers that are present between each stage. These registers are controlled by the rising clock edge. In each cycle, every stage would compute its instructions by using the values the components would receive from the pipeline register preceeding it. Since the registers are controlled by the rising clock edge, the previous values would be rewritten by the new ones, advancing each instruction by a stage each cycle. In addition to those, we also created a Hazard and Forwarding unit. The Hazarding unit detected when a load, structural, or control hazard were present. If they were, it would act accordingly to stall the next instruction as needed. The Forwarding unit detected when forwarding was necessary. If it was, it would forward the data to the necessary proceeding instructions. We also needed to add a comparator outside of the ALU and in the ID stage to reduce stalls when branches were taken. To control the comparator, we needed to incorporate two muxes to choose two of the four possible inputs. We also needed to include two more 3-to-1 muxes before the ALU to control when information needs to be forwarded to later instructions.



1.3 Testbench Architecture

Our testbench is the same as the one we used for Lab 3. We needed to add more cycles to accomodate the different cycles, since each instruction will now take 5 cycles, one for each stage, instead of one.

2 Sample Program

2.1 Instructions

We were given a list of 18 32-bit MIPS instructions, written in hexadecimal form. Before proceeding, we decoded these instructions into a human-readable format.

Instruction (HEX)	Instr-Type	OPCODE	RS	RT	RD	SHAMT	Func	Imm/Offset
20020005	Addi	8	0	2				5
2003000c	Addi	8	0	3				12
2067fff7	Addi	8	3	7				-9
00e22025	OR		7	2	4	0	37	
00642824	AND		3	4	5	0	36	
00a42820	Add		5	4	5	0	32	
10a7000a	Beq	4	5	7				10
0064202a	Slt		3	4	4	0	42	
10800001	Beq	4	4	0				1
20050000	Addi	8	0	5				0
00e2202a	Slt		7	2	4	0	42	
00853820	Add		4	5	7	0	32	
00e23822	Sub		7	2	7	0	34	
ac670044	SW	43	3	7				68
8c020050	LW	35	0	2				80
08000011	Jump	2	0	0				17
20020001	Addi	8	0	2				1
ac020054	SW	43	0	2				84

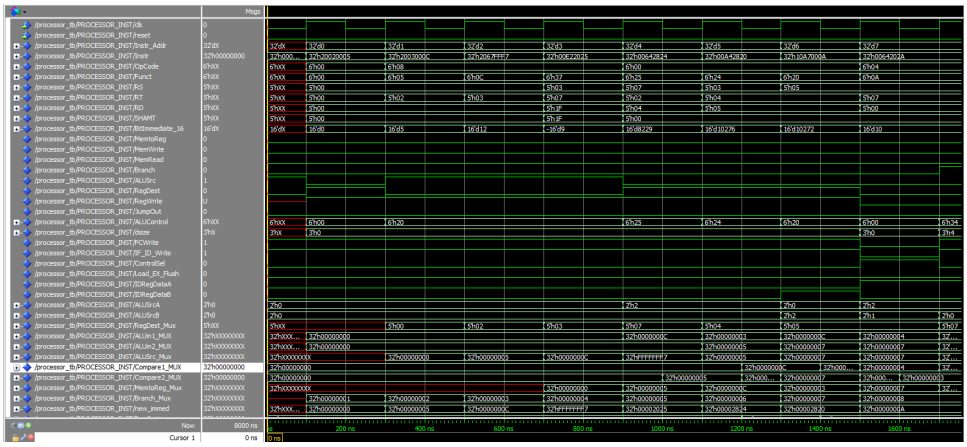
After decoding the instructions, we worked out the theoretical state of the Register File after every instruction. Of the two branch instructions in our sample program, one is successful and results in four instructions being skipped. There is also a jump instruction near the end of the program, which skips the final two instructions and moves the program counter into undefined memory, resulting in a "garbage" instruction.

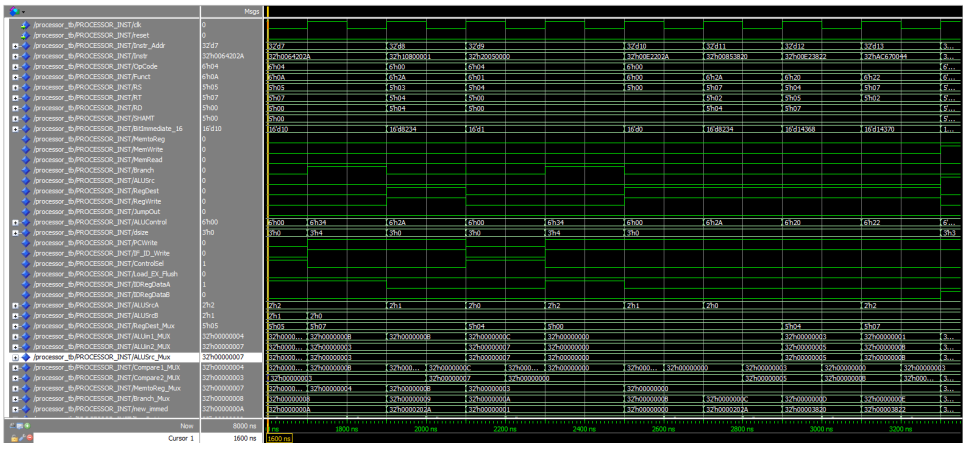
Register	Register Value (Initial)		1st Instr	2nd Instr	3rd Instr	4	5	6	7	8	9	10	11	12	13	14	15
0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0		5	5	5	5	5	5	5	5	5	5	5	5	5	5	3
3	0		0	12	12	12	12	12	12	12	12	12	12	12	12	12	12
4	0		0	0	0	7	7	7	7	0	0	0	0	0	0	0	0
5	0		0	0	0	0	4	11	11	11	11	11	11	11	11	11	11
6	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0		0	0	3	3	3	3	3	3	3	3	3	3	3	3	3
8	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
									No Branch	FALSE	BRANCH! Add 4	Skipped	Skipped	Skipped	Skipped	Store RT at RS+	Load RS+Offset

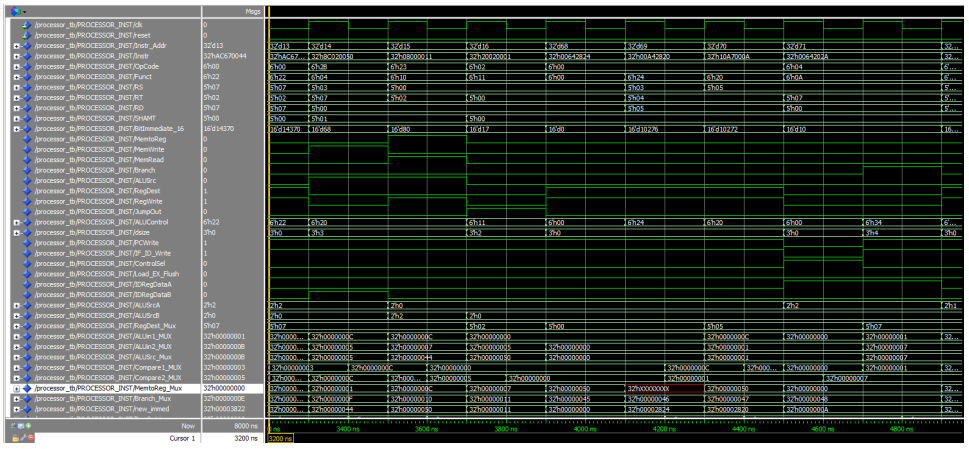
Register	16	17	18	
0	0	0	0	
1	0	0	0	
2	3	3	3	
3	12	12	12	
4	0	0	0	
5	11	11	11	
6	0	0	0	
7	3	3	3	
8	0	0	0	
9	0	0	0	
10	0	0	0	
	Jump to 17 * 4 =	Skipped	Skipped	END

2.2 Simulation Waveform

After designing and testing our Processor, we generated the following waveforms:







Each of the pipeline stages, hazards, forwarding, and stalls were correctly implemented. The instruction values shown in each cycle correlate to the instruction in the ID stage, while other various signals indicate the values for other instructions in different stages; for example, ALUresult outputs the evaluated ALU value from the EX stage. The pipeline registers indicate the transition of each instruction into the next stage.

3 Synthesis

Through synthesis, we were able to determine the power consumption, area requirements, and frequency of our processor design. Synopsis, a software suite provided by our university, was used to perform synthesis on our design. After adjusting the analysis, elaboration, and synthesis files and scripts, we synthesized our processor design and successfully generated various reports on our processor's performance.

3.1 Power Consumption

Our processor's total power consumption came out to 18.7 mW, down from 20.3 mW in our Single-Cycle Processor. From the power hierarchy we generated. Our Register File consumes approximately 57.7% of the power, which is down from the 74% it consumed in our single-clycle processor. The ALU was the only other component that consumed a significant amount of power. At 13.3%, it consumed 6.8% less than it did in our single cycle processor. All other components use three orders of magnitude less power.

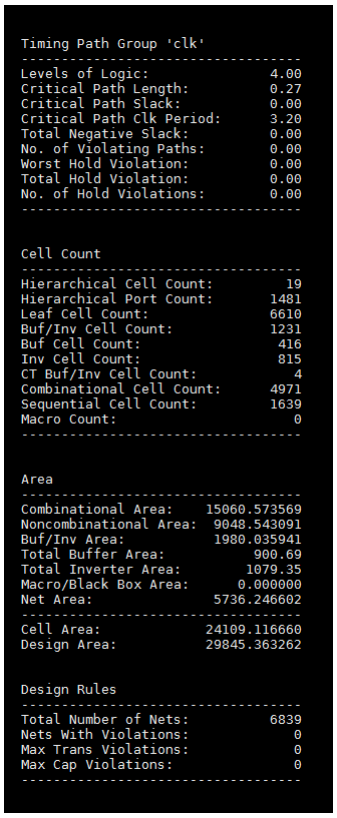
Hierarchy	Switch Power	Int Power	Leak Power	Total Power	%
processor	99.941	218.646	1.84e+10	1.87e+04	100.0
Ram1 (ram)	1.002	0.000	0.000	1.002	0.0
A1 (alu)	11.497	24.335	2.44e+09	2.48e+03	13.3
r51 (alu_DW01_addsub_0)	0.835	2.167	1.58e+08	161.334	0.9
C1 (Comparator)	0.265	0.900	9.25e+07	93.691	0.5
F (forward)	0.682	2.073	1.21e+08	124.049	0.7
BC (BranchComparator)	0.119	0.404	1.34e+08	134.718	0.7
eq_34 (BranchComparator_DW01_cmp6_0)	0.119	0.404	1.34e+08	134.718	0.7
NormAdder (adder_addr_size32_1)	0.175	0.658	6.29e+07	63.730	0.3
add_18 (adder_addr_size32_1_DW01_add_0)	0.175	0.658	6.29e+07	63.730	0.3
R1 (regfile_NBIT32_NSEL5)	76.005	54.208	1.06e+10	1.08e+04	57.7
C1 (controller)	0.244	0.154	1.16e+08	116.262	0.6
H (hazard)	0.178	1.179	8.42e+07	85.542	0.5
IMEM_1 (synth_imem)	0.823	3.900	1.25e+08	129.497	0.7
PCAdder (adder_addr_size32_0)	7.99e-02	0.280	4.92e+07	49.593	0.3
add_18 (adder_addr_size32_0_DW01_add_0)	7.99e-02	0.280	4.92e+07	49.593	0.3
ProgCnt (PC)	2.794	22.869	5.33e+08	558.163	3.0

3.2 Area

After reading the generated area reports, we could not determine the units being used in the report. However, Synopsys reported that our Design area would be 25485.282135, down from 82228.100300 in our Single-Cycle Processor, with a Combinational Area of approximately 10926, which is 5942 less than before. These results are from our own Data Memory component, not using Professor Yaghini's SRAM. Were it tested with the Professor's SRAM, the area would likely be half the size.

3.3 Required Frequency

According to the generated timing report, our Critical Path Length was .27, which is a significant improvement over the 3.18 ns taken by our Single-Cycle Processor. This allowed us to decrease our clock period from 3.2 ns to 1 ns and still have no negative slack. This results in a frequency of approximately 100 MHz.



```
Timing Path Group 'clk'
-----------------------------------
Levels of Logic:               4.00
Critical Path Length:          0.27
Critical Path Slack:           0.00
Critical Path Clk Period:      3.20
Total Negative Slack:          0.00
No. of Violating Paths:        0.00
Worst Hold Violation:          0.00
Total Hold Violation:          0.00
No. of Hold Violations:        0.00
-----------------------------------


Cell Count
-----------------------------------
Hierarchical Cell Count:         19
Hierarchical Port Count:       1481
Leaf Cell Count:               6610
Buf/Inv Cell Count:            1231
Buf Cell Count:                 416
Inv Cell Count:                 815
CT Buf/Inv Cell Count:            4
Combinational Cell Count:      4971
Sequential Cell Count:         1639
Macro Count:                      0
-----------------------------------


Area
-----------------------------------
Combinational Area:    15060.573569
Noncombinational Area:  9048.543091
Buf/Inv Area:           1980.035941
Total Buffer Area:           900.69
Total Inverter Area:        1079.35
Macro/Black Box Area:      0.000000
Net Area:               5736.246602
-----------------------------------
Cell Area:             24109.116660
Design Area:           29845.363262


Design Rules
-----------------------------------
Total Number of Nets:          6839
Nets With Violations:             0
Max Trans Violations:             0
Max Cap Violations:               0
-----------------------------------
```

4 Known Issues

Due to our unfamiliarity with SystemVerilog, we chose not to use the testbench to preload instruction memory. For the Questasim simulation, we read the example program instructions from a file. For synthesis, we created a second version of instruction memory that has instructions preloaded into memory in order to synthesize the Processor properly.

5 Conclusion

As a complete pipeline processor, our project now supports the full MIPS instruction set. It is capable of all three types of MIPS instructions: R-type, I-type, and J-type. Unlike our single cycle processor, however, it now has a better throughput and can implement forwarding, stalling, and hazard detection. Through synthesis, we were able to determine the area, power and frequency properties our processor design would have if fabricated.