ECE 212 – Digital Circuits II

Lab10 – MIPS Multicycle Processor (Part2)

Otieno Maurice (Scribe)

Alex Villalba

https://github.com/otienomaurice1/ece212_alex_maurice.git

Introduction:

In this lab we designed the data path for the multicycle processor. The code for major components was designed earlier and the only task we had was connecting this component together. The major components of the multicycle processor include the memory file, register file, the alu, the pc register, the instr register, the data register, the alu register and the alu source registers. Most of the register files were created as instances of either the flopr or flopenr modules while most of the multiplexers were instances of mux2, mux3 and mux4 multiplexers.

The pc register reads from the jump path, the alu register and the alu via a mux3.
The memory file receives the address to read from either the alu register via the instruction or data mux2. The memory file reads into the instruction and memory registers. The instruction register's output is the instruction, whose value runs the Datapath. The data register reads into the register file via a mux2.The register file reads into the alu source registers and the alu source registers read into the alu via the source multiplexers. This multi-cycle processor has the ability to compute lw, sw, j, beq,sub,and,add,slt,addi,and or. The signals that control the registers and the multiplexers in the Datapath are inputs from the control unit that was designed in lab 9 earlier.

The top-level diagram for the multicycle diagram is shown in.

Also shown is the table detailing the expected signals as the program runs.

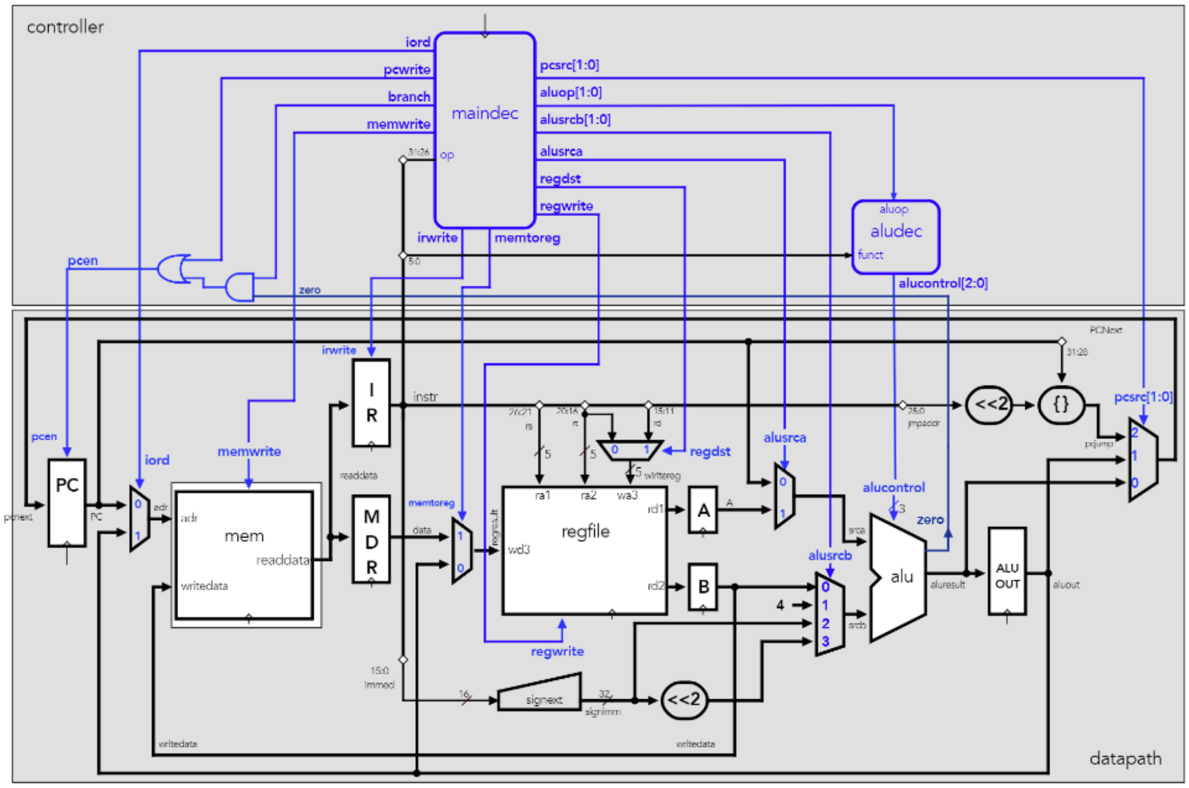
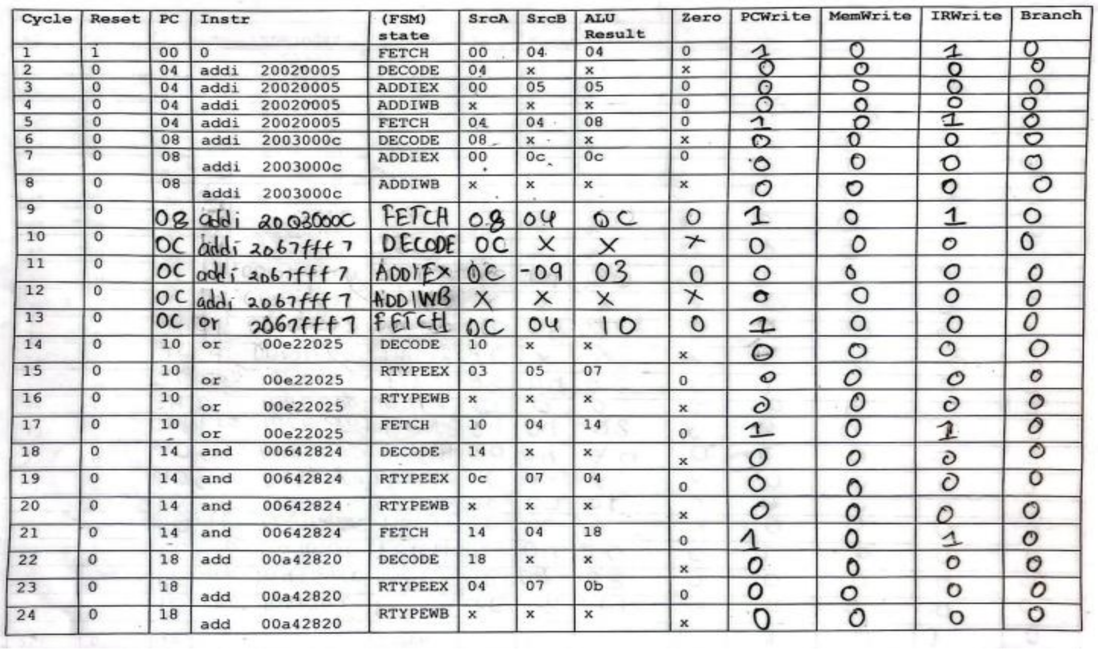


Figure 1



Cycle	Reset	PC	Instr	(FSM) state	SrcA	SrcB	ALU Result	Zero	PCWrite	MemWrite	IRWrite	Branch
1	1	00	0	FETCH	00	04	04	0	1	0	1	0
2	0	04	addi 20020005	DECODE	04	x	x	x	0	0	0	0
3	0	04	addi 20020005	ADDIEX	00	05	05	0	0	0	0	0
4	0	04	addi 20020005	ADDIWB	x	x	x	0	0	0	0	0
5	0	04	addi 20020005	FETCH	04	04	08	0	1	0	1	0
6	0	08	addi 2003000c	DECODE	08	x	x	x	0	0	0	0
7	0	08	addi 2003000c	ADDIEX	00	0c	0c	0	0	0	0	0
8	0	08	addi 2003000c	ADDIWB	x	x	x	x	0	0	0	0
9	0	08	addi 2003000C	FETCH	08	04	0C	0	1	0	1	0
10	0	0C	addi 2067fff7	DECODE	0C	X	X	X	0	0	0	0
11	0	0C	addi 2067fff7	ADDIEX	0C	-09	03	0	0	0	0	0
12	0	0C	addi 2067fff7	ADDIWB	X	X	X	X	0	0	0	0
13	0	0C	or 2067fff7	FETCH	0C	04	10	0	1	0	0	0
14	0	10	or 00e22025	DECODE	10	x	x	x	0	0	0	0
15	0	10	or 00e22025	RTYPEEX	03	05	07	0	0	0	0	0
16	0	10	or 00e22025	RTYPEWB	x	x	x	x	0	0	0	0
17	0	10	or 00e22025	FETCH	10	04	14	0	1	0	1	0
18	0	14	and 00642824	DECODE	14	x	x	x	0	0	0	0
19	0	14	and 00642824	RTYPEEX	0c	07	04	0	0	0	0	0
20	0	14	and 00642824	RTYPEWB	x	x	x	x	0	0	0	0
21	0	14	and 00642824	FETCH	14	04	18	0	1	0	1	0
22	0	18	add 00a42820	DECODE	18	x	x	x	0	0	0	0
23	0	18	add 00a42820	RTYPEEX	04	07	0b	0	0	0	0	0
24	0	18	add 00a42820	RTYPEWB	x	x	x	x	0	0	0	0

25	0	18	add	00a42820	FETCH	18	04	1c	0	1	0	1	0
26	0	1C	beq	1097000a	DECODE	1C	28	44	X	0	0	0	0
27	0	1C	beq	1097000a	BEQ EX	0B	03	08	0	0	0	0	1
28	0	1C	beq	1097000a	FETCH	1C	04	20	0	1	0	1	0
29	0	20	slt	0064202a	DECODE	1C	X	X	X	0	0	0	0
30	0	20	slt	0064202a	RTYPEEX	0C	07	01	0	0	0	0	0
31	0	20	slt	0064202a	RTYPEWB	X	X	X	X	0	0	0	0
32	0	20	slt	0064202a	FETCH	20	04	24	0	1	0	1	0
33	0	24	beq	10800001	DECODE	24	04	28	X	0	0	0	0
34	0	24	beq	10800001	BEQEX	0	0	0	1	0	0	1	1
35	0	28	beq	10800001	FETCH	28	04	2C	0	1	0	0	0
36	0	2C	slt	00e2202a	DECODE	2C	X	X	X	0	0	0	0
37	0	2C	slt	00e2202a	RTYPEEX	03	05	01	0	0	0	0	0
38	0	2C	slt	00e2202a	RTYPEWB	X	X	X	X	0	0	0	0
39	0	2C	slt	00e2202a	FETCH	2C	04	30	0	1	0	1	0
40	0	30	add	00853820	DECODE	30	x	x	x	0	0	0	0
41	0	30	add	00853820	RTYPEEX	01	0b	0c	0	0	0	0	0
42	0	30	add	00853820	RTYPEWB	x	x	x	x	0	0	0	0
43	0	30	add	00853820	FETCH	30	04	34	0	1	0	1	0
44	0	34	sub	00e23822	DECODE	34	x	x	x	0	0	0	0
45	0	34	sub	00e23822	RTYPEEX	0c	05	07	0	0	0	0	0
46	0	34	sub	00e23822	RTYPEWB	x	x	x	x	0	0	0	0
47	0	34	sub	00e23822	FETCH	34	04	38	0	1	0	1	0
48	0	38	sw	ac670044	DECODE	38	x	x	x	0	0	0	0
49	0	38	sw	ac670044	MEMADR	0c	44	50	0	0	6	0	0
50	0	38	sw	ac670044	MEMWR	x	x	x	x	0	1	0	0
51	0	38	sw	ac670044	FETCH	38	04	3c	0	1	0	1	0
52	0	3c	lw	8c020050	DECODE	3c	x	x	x	0	0	0	0
53	0	3c	lw	8c020050	MEMADR	00	50	50	0	0	0	0	0
54	0	3c	lw	8c020050	MEMRD	x	x	x	x	0	0	0	0
55	0	3c	lw	8c020050	MEMWB	x	x	x	x	0	0	0	0
56	0	3c	lw	8c020050	FETCH	3c	04	40	0	1	0	1	0
57	0	40	J	08000011	DECODE	40	04	44	X	0	0	0	0
58	0	44	J	08000011	JEX	X	X	X	0	0	0	0	0
59	0	44	J	08000011	FETCH	44	04	48	0	1	0	1	0
60	0	48	SW	ac020054	DECODE	48	X	X	X	0	0	0	0
61	0	48	SW	ac020054	MEMADR	00	54	54	0	0	0	0	0
62	0	48	SW	ac020054	MEMWR	X	X	X	X	0	1	0	0

Table 1. Expected Instruction Trace

Results

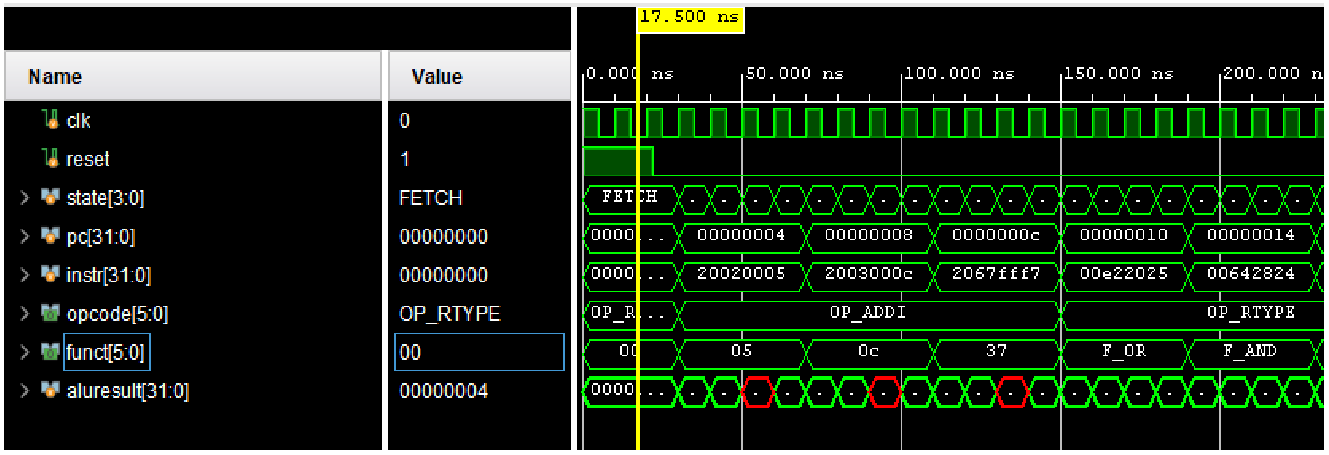


Figure 2screenshot 1 showing beginning of the program

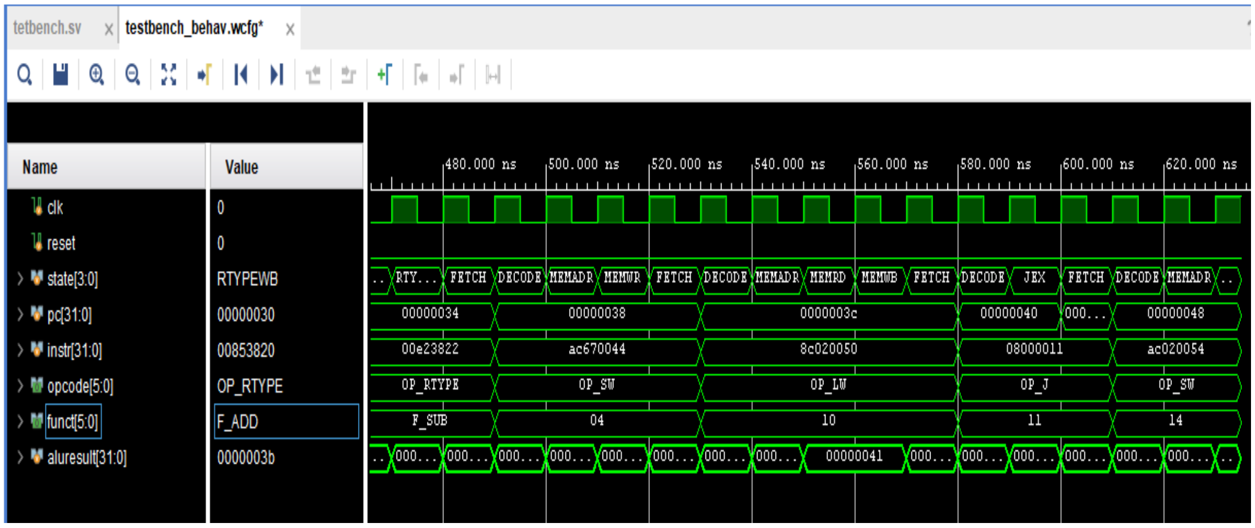
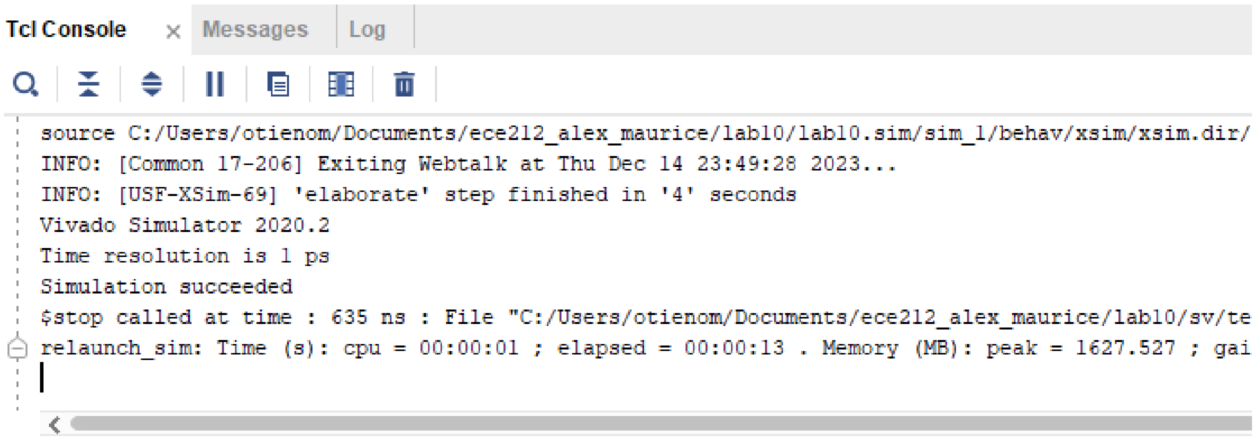


Figure 3 screenshot 2 showing end of program



```
source C:/Users/otienom/Documents/ece212_alex_maurice/lab10/lab10.sim/sim_1/behav/xsim/xsim.dir/
INFO: [Common 17-206] Exiting Webtalk at Thu Dec 14 23:49:28 2023...
INFO: [USF-XSim-69] 'elaborate' step finished in '4' seconds
Vivado Simulator 2020.2
Time resolution is 1 ps
Simulation succeeded
$stop called at time : 635 ns : File "C:/Users/otienom/Documents/ece212_alex_maurice/lab10/sv/te
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:13 . Memory (MB): peak = 1627.527 ; gai
```

Figure 4 tcl console screenshot showing success of the mips multicycle processor

Conclusion:

The mips multicycle Datapath was much more challenging to design. We ran into several problems dealing with bit widths when doing port declarations and connections. This made the debugging process a little painful. The process was however, simplified by the fact that most of the individual designs were pre-coded.

Time spent on lab

6 hrs.