

EECS 4321 Final Project : 8-bit Microcontroller

Yuxiang Chen (yc3096), Xinyi Chang(xc2323)

1. Introduction

This report gives a guideline to build a simplified 8-bit CPU with 6-bit instruction word and 3-bit opcode. IBM 90nm technology is used to implement this microcontroller. This design is optimized to a maximum frequency of 454 MHz. Along with the design specification, implementation, optimization and testing strategy will also be analyzed in this report. The project is implemented in a group of two people, and the project is saved in Yuxiang Chen's directory, library 'yc3096'.

2. Analytical Approach

This project is the final push of the entire microprocessor core design. Three more dataflow blocks are added to the design in addition to the shifters, adders, and SRAM blocks which completed in the previous projects. The three new blocks are 8-bits sensitive latch, 8-bits 3-to-1 multiplexer, and 8-bit bus driver. In order to minimum the voltage drop from 3-nmos multiplexer design, transmission gates are used along with buffers to reduce the noise. Bus driver and latches are implemented using the standard design with the optimized sizing.

In addition to these dataflow blocks, a 3-input 10-output instruction decoder are implemented as a static pseudo-NMOS PLA with the truth table showing in figure 2.1. Espresso tool is used for logic minimization. Figure 2.2 shows the optimized result from espresso tool. Figure 2.3 shows the schematic design of PLA.

After all components are designed, all blocks are connected together. Our final schematic design is shown in figure 2.4.

instr	Op	Sub	Mux_ctl	bus_en	mem_wr	mem_rd	shift_bypass	ld_bus	st_bus
NOP	000	0	000	0	0	0	1	0	0
LOAD	001	0	000	0	1	0	1	1	0
STORE	010	0	000	0	0	1	1	0	1
GET	011	0	001	0	0	1	1	0	0
PUT	100	0	000	1	1	0	1	0	0
ADD	101	0	010	0	0	1	1	0	0
SUB	110	1	010	0	0	1	1	0	0
SHIFT	111	0	100	0	0	0	0	0	0

Figure 2.1 Instruction Decoder Truth Table

```
35 # ESPRESSO Time was 0.00 sec, cost is c=8(0) in=22 out=18 tot=40
36 .i 3
37 .o 10
38 .ilb op2 op1 op0
39 .ob sub mux2 mux1 mux0 bus_en men_wr men_rd shifn ld_bus st_bus
40 .p 8
41 111 0100000000
42 100 0000110000
43 010 0000001001
44 001 0000010110
45 110 1010001000
46 011 0001001100
47 101 0010001100
48 --0 0000000100
49 .e
50 # WRITE Time was 0.00 sec, cost is c=8(0) in=22 out=18 tot=40
51
```

Figure 2.2 Espresso Optimized Truth Table

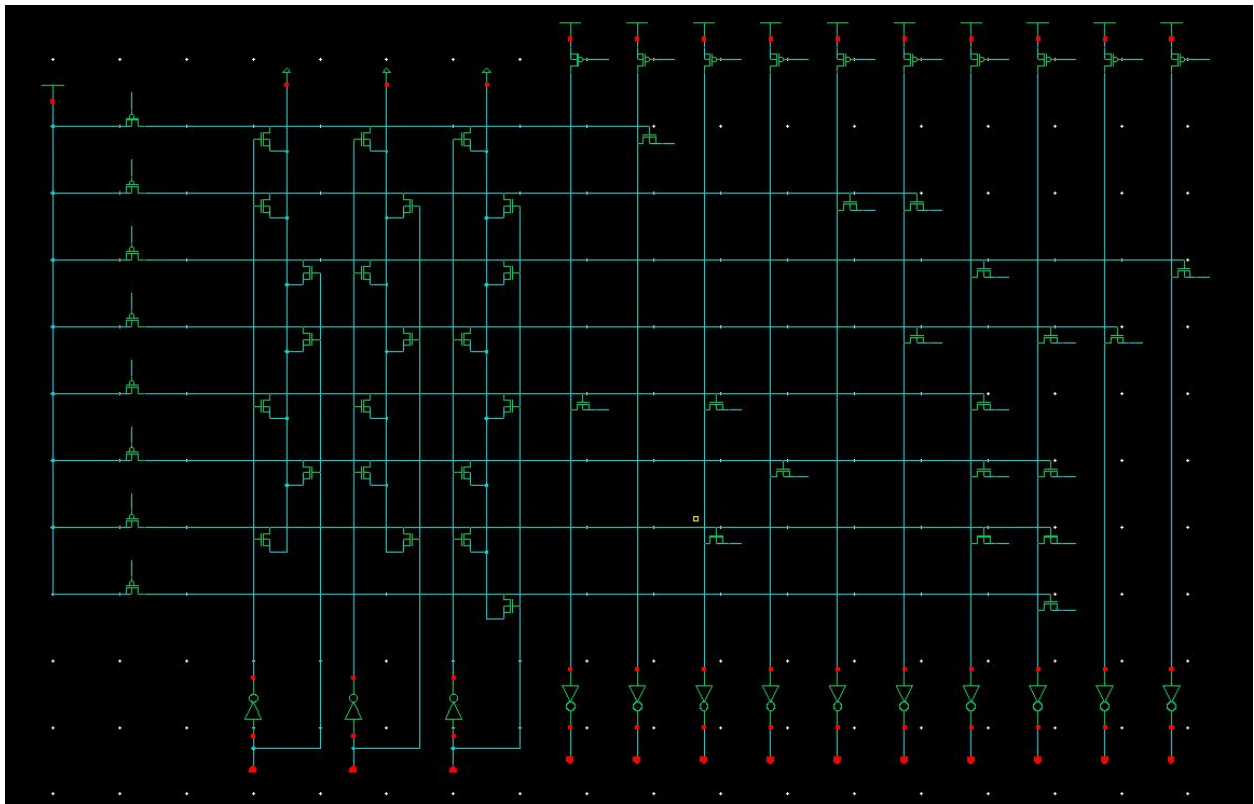


Figure 2.3 Schematic Design of Instruction Decoder (PLA)

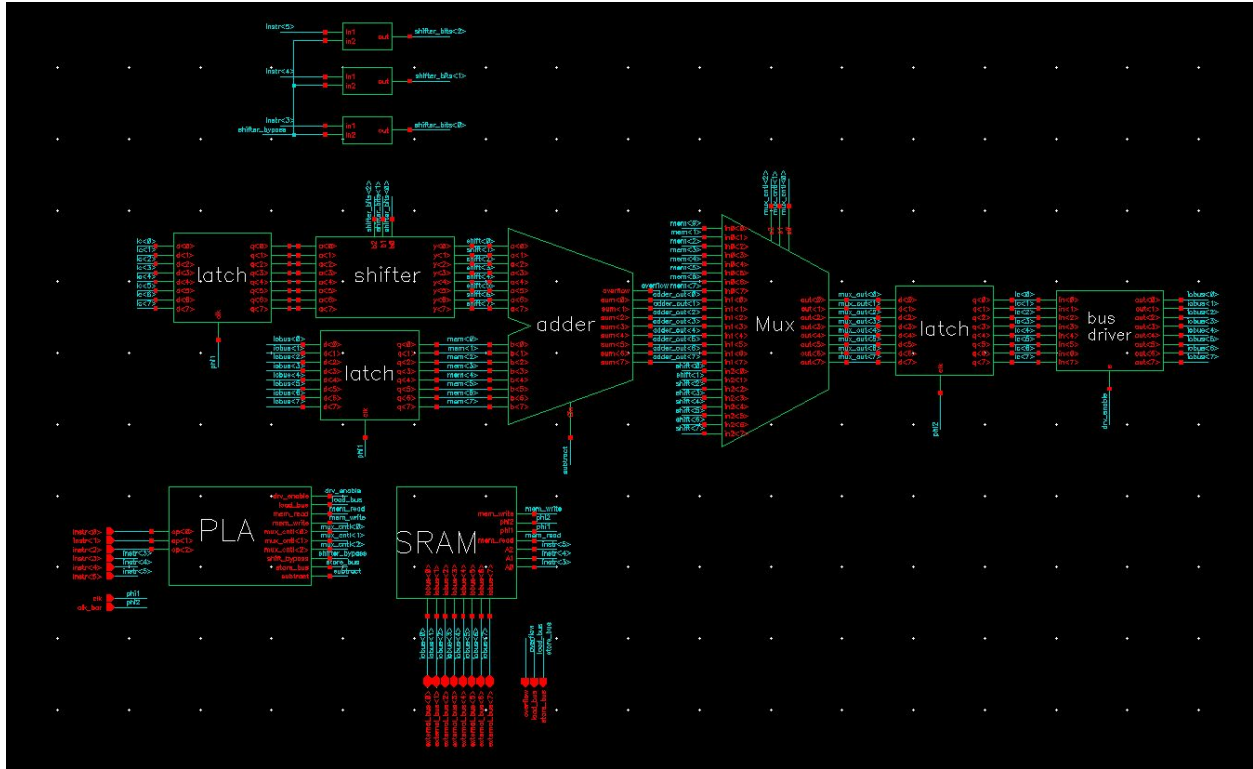


Figure 2.4 Overall CPU Schematic

3. Layout Design

To design a PLA with a compact area, we followed the pseudo-nMOS PLA layout structure shown in the W&H textbook. Figure 3.1 shows our layout design of the instruction decoder.

After finish all components, we connected all of the blocks together. Figure 3.2 shows the floor plan of layout design considering the layout hierarchy, and figure 3.3 shows our final CPU layout. The CPU is DRC and LVS clean as shown in figure 3.4.

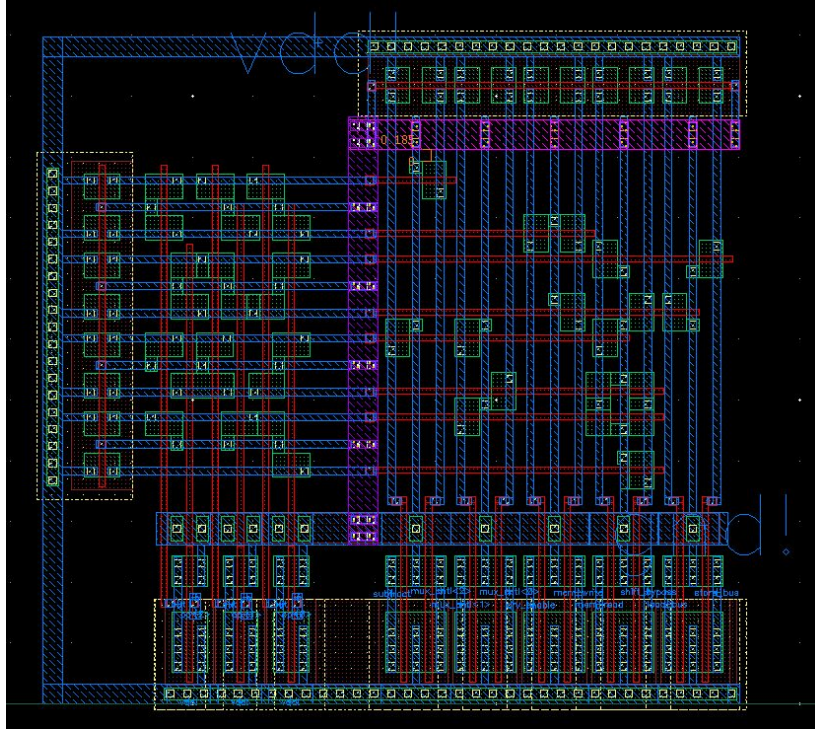


Figure 3.1 Instruction Decoder



Figure 3.2 Floor Plan of Layout

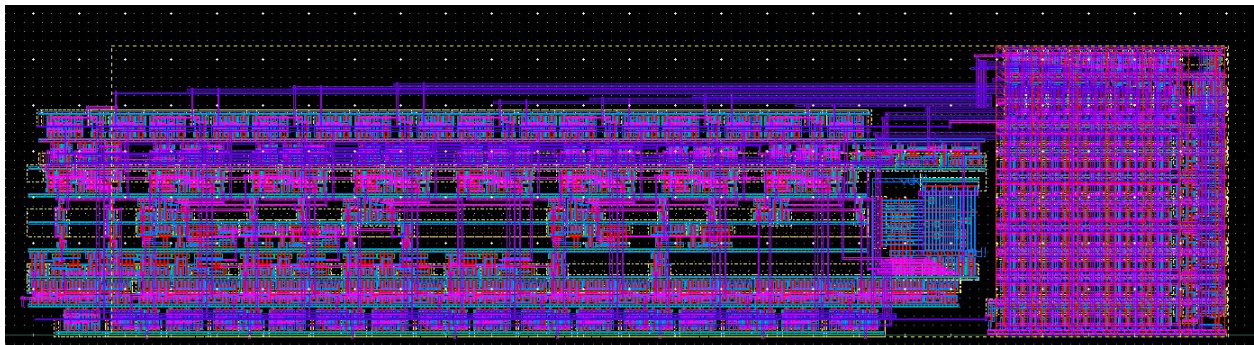


Figure 3.3 Overall CPU Layout

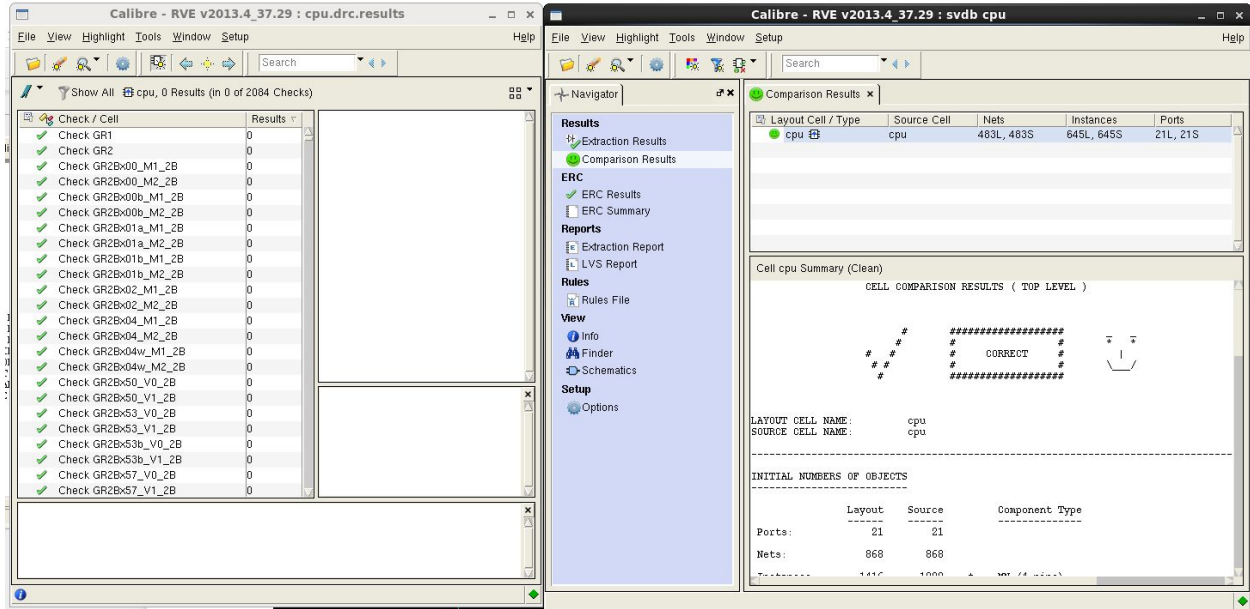


Figure 3.4 DRC and LVS Report

4. Ultrasim Verification

To verify the functionality of the CPU, the test vector in figure 4.1 are used. Figure 4.2 and figure 4.3 shows the schematic result and layout result respectively. The results show that our design is functionally correct.

#	opcode(instr<2:0>)	Instruction	Instr<5:3>	Test Vector
1	001	LOAD	111	addr 7 = 10000110
2	011	GET	111	Acc = addr 7
3	111	SHIFT	001	Acc << 1
4	100	PUT	110	addr 6 = Acc
5	101	ADD	110	Acc = Acc + addr 6
6	001	LOAD	010	addr2 = 0000001
7	110	SUB	010	Acc = Acc - addr2
8	110	SUB	010	Acc = Acc - addr2
9	100	PUT	001	addr1 = Acc
10	001	LOAD	110	addr 6 = 10000000
11	111	SHIFT	111	Acc << 7

12	110	SUB	110	Acc = Acc - addr 6
13	100	PUT	011	addr3 = Acc
14	010	STORE	010	bus = addr2
15	010	STORE	011	bus = addr3

Figure 4.1 Test Vectors



Figure 4.2 Schematic Simulation Result

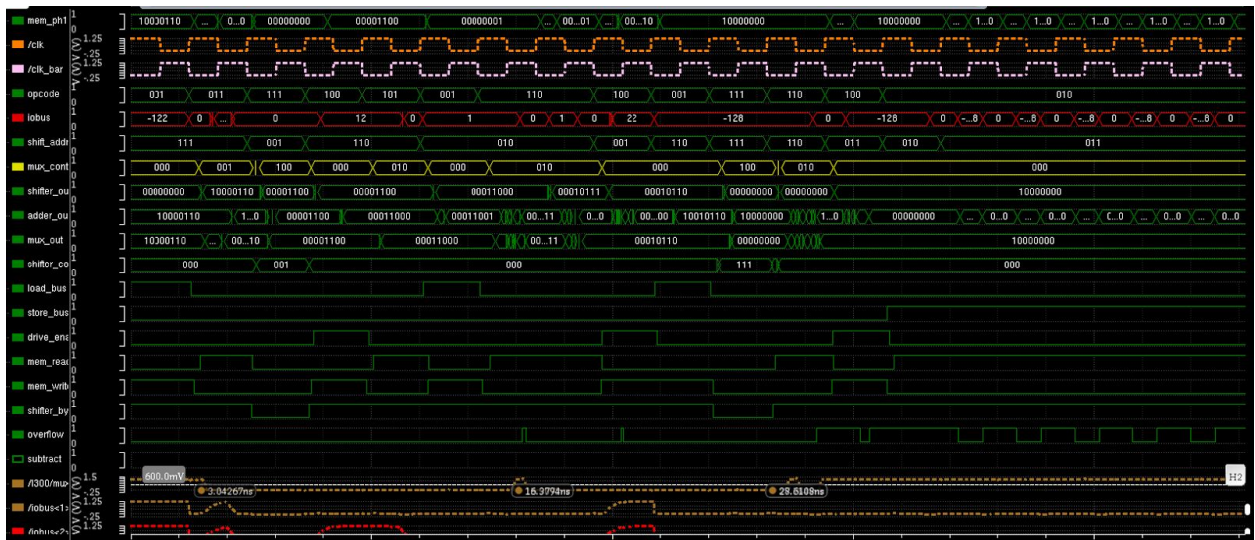


Figure 4.3 Lay Simulation Result

5.Delay

The largest delay of our design is coming from subtract 'acc' from the value from SRAM. The critical path delay is from instruction 11 to 12, when running "00000000" - "10000000" (carry in = 1). The worst delay of schematic is (11.86199 ns - 11.01 ns = 852 ps) as shown in figure 5.1. The worst delay of layout is 2.21 ns.

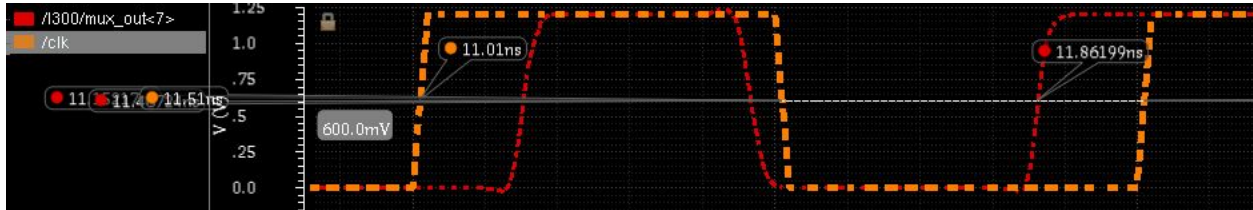


Figure 5.1 Delay for Schematic



Figure 5.2 Delay for Layout

6. Power

Figure 6.1 and 6.2 show the average power dissipation in 13 instructions. The schematic simulates an average power dissipation of **1.38 mW** and the extracted layout simulates an average power dissipation of **1.978 mW**.



Figure 6.1 Power Dissipation Simulation for Schematic

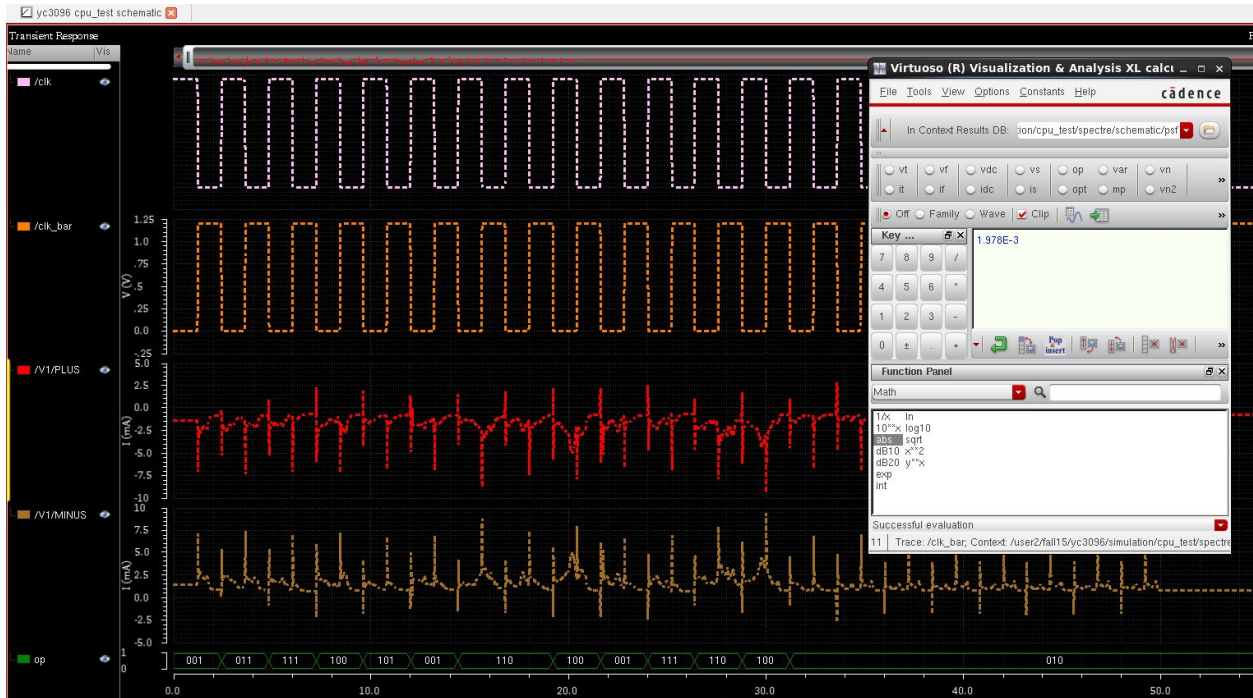


Figure 6.2 Power Dissipation Simulation for Layout

7. Conclusion

Overall, we successfully implemented a 8-bits CPU with the expected functionality. The design achieved a **maximum frequency of 454 MHz** in extracted layout. The overall layout size is 128.33(width) * 31.605(height). We didn't use the standard cell provided by the course because we want to achieve a SRAM with better stability and drivability. The trade off is larger area in overall SRAM cell(24.98 * 31.605). Besides that, we think the design is in great density.

To further optimize our design. The adder could be sized more reasonable. The adder can be easily speed up by 16% performance by sizing up the gate with large fan-out, especially in our Sklansky architecture. However, due to timing constraint of the project, we didn't further optimize the adder.