WonBin Seo

Dr. Shannon Nicley,

ECE 813 Advanced VLSI Design

2 May 2025

8 bit Microprocessor Design with Cadence

ECE 813 Design Project Report

Michigan State University

Spring 2025

1. Project Overview

This project involves designing a complete 8-bit data path, including a register file (SRAM), an Arithmetic Logic Unit (ALU), and a barrel shifter, and a register file using Cadence Virtuoso design tools.

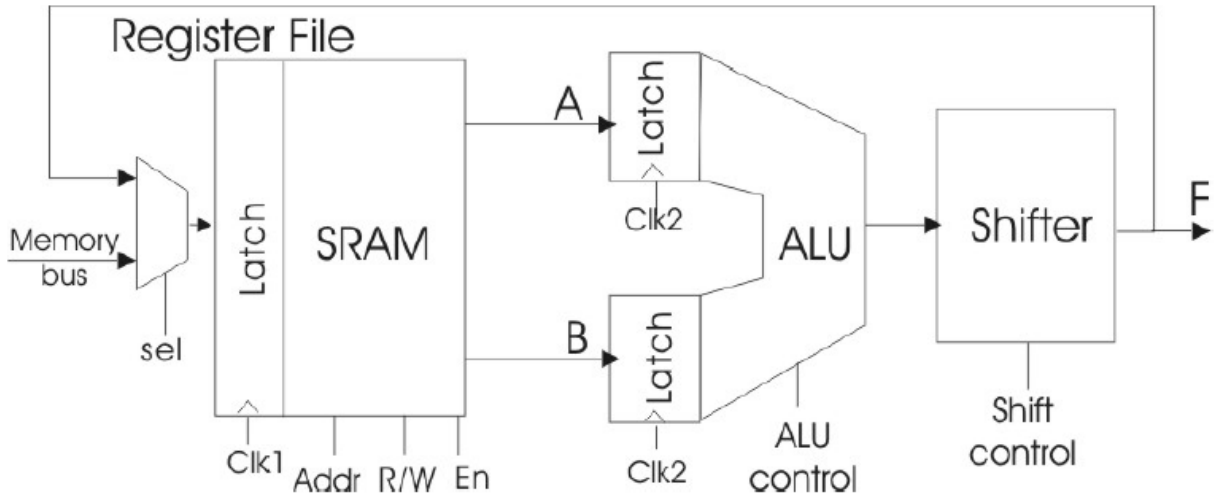

Figure 1. Structure of the 8-bit microprocessor data path

In this data path, we must design a complete 8-bit data path consisting of a register file (SRAM), an ALU (Arithmetic Logic Unit) and a Barrel shifter, as shown in Figure 1. The data path will be capable of multiple instructions determined by several control signals that comprise the operational code or "opcode".

The ALU can perform both arithmetic and bit-wise logical operations. The ALU will have two 8-bit data inputs and several control signals that specify the ALU function to be executed in a given clock cycle. The ALU input data is loaded into the latch at the rising edge of the ALU clock (clk2), and the output is generated after a delay due propagation through the combinational logic of the ALU and the Shifter. ALU will contain add/subtract and other logical calculations (Increment A, Increment B, A NAND B, A NOR B, A XOR B, INV A), using Manchester carry generation carry lookahead adder.

A shifter provides data manipulation capabilities. In this design, we will use barrel shifter that executing logical shifts. It will perform 7-bit logical shifts, which can be implemented by opcodes.

The register file is a block of memory that provides the data inputs to and stores outputs from the ALU. The register file at read/write address is specified by control bits included in the opcode. We will use multiport SRAM, has two read line and one write line.

From Figure 2, we can see the whole data path schematic have implemented in cadence tool.

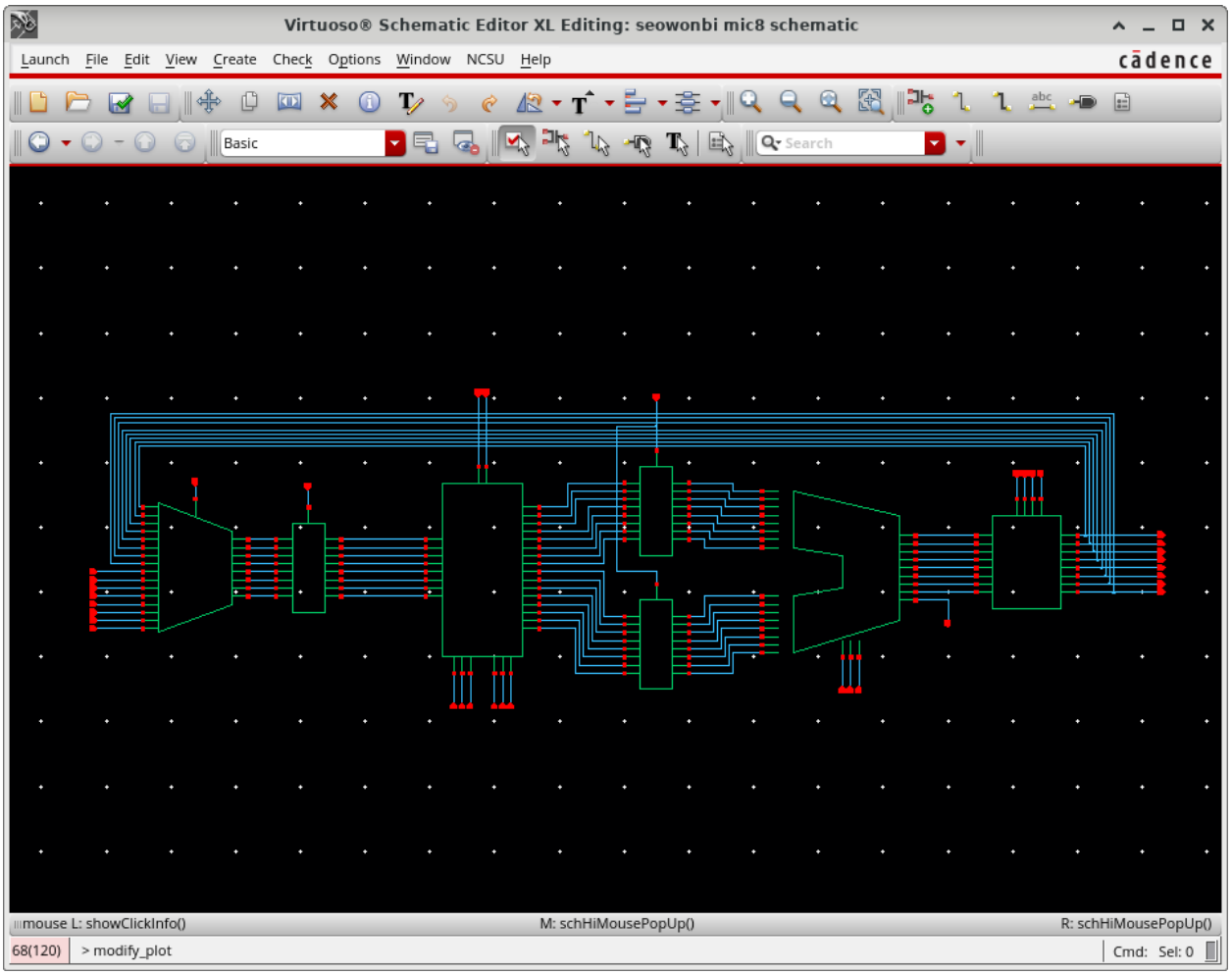

Figure 2. Schematic of the 8-bit Microprocessor

2. Design Methodology

In this project, we will confront many types of challenges. Firstly, the whole data path is compromised in limited area. Based on the design specifications provided by Cadence, the layout was optimized to minimize area usage. For this reason, we must find which component has the largest portion of the design, trying to reduce the amount of the area. Secondly, Divide the design into small pieces. For instance, the 8-bit architecture was modularized into 4-bit, 2-bit, and 1-bit components to facilitate design scalability and reuse. In the design of the ALU, Register File, and Shifter, we will use it effectively.

The preparation process began with a thorough understanding of the preceding lectures. After fully grasping the content, I proceeded with the design implementation. I aimed to ensure that all layouts complied with DRC (Design Rule Check) requirements.

In particular, I constructed a Logic Unit and used it to build a hierarchical structure. During this process, I rotated the unit where necessary to improve space utilization and routing efficiency. Additionally, a stepped routing structure was frequently employed to enhance spatial efficiency, as seen in minimized unused space.

To manage both the design project and exams concurrently, I established a parallel schedule alongside Exam 1 through Exam 3. During this period, I also implemented additional required components such as latches, NAND8, NOR8, and XOR8. Finally, I completed the report and simulation phases.

Figure 3 illustrates the efforts made through rotation to optimize space and routing, while Figure 4 shows an example of stepped routing used to maximize spatial efficiency.

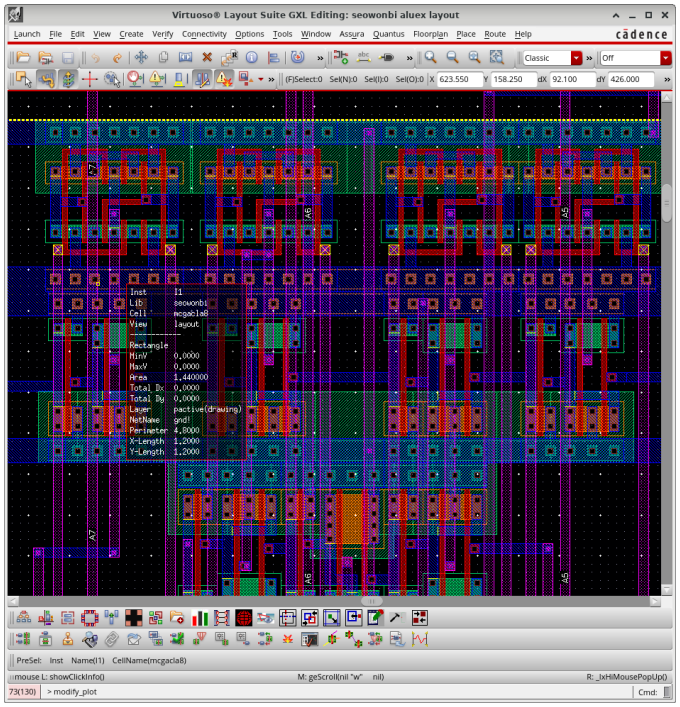

Figure 3. Rotating the Logical Units

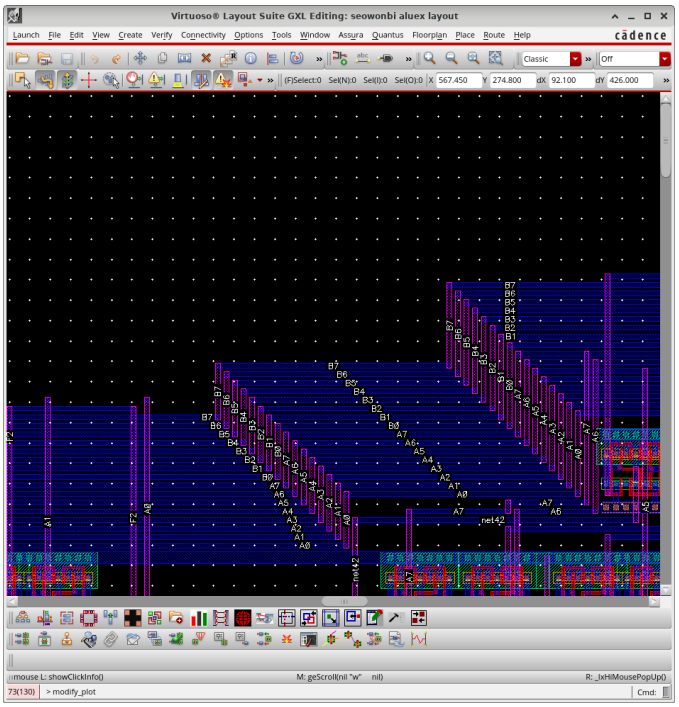

Figure 4. Stepped Structure

3. Design and Results

To implement this 8-bit microprocessor, I developed the key units—including an 8×8 SRAM, an adder, and various logic gates—through Exam 1 to Exam 3. Based on these units, I proceeded to design additional unit cells. Notable examples include the ALU, which incorporates an adder, and the barrel shifter, which was built using multiple 2-to-1 multiplexers. For 8x8 SRAM, we already discussed the whole thing in Exam3, we will talk about ALU, Shifter and latch.

For the ALU, the adder was implemented in Exam 2 using a Manchester Carry Adder. However, implementing the subtractor and other logic functions presented challenges. To address subtraction, I implemented a 16-bit XOR unit, enabling the same adder to perform both addition and subtraction depending on the carry-in (Cin) signal. Furthermore, to support various bitwise logic functions (NAND8, NOR8, XOR8, INV8), I designed each as 16-to-8 modules. Then, using a 3-level multiplexer structure, I built an 8-to-1 MUX that could output signals Y0 to Y7 and Cout. The final ALU was composed of three adders (for add/subtract, increment A, and increment B) and four logic units, supporting full bitwise operation. We can find the logical functions through Figure 5 to 8.

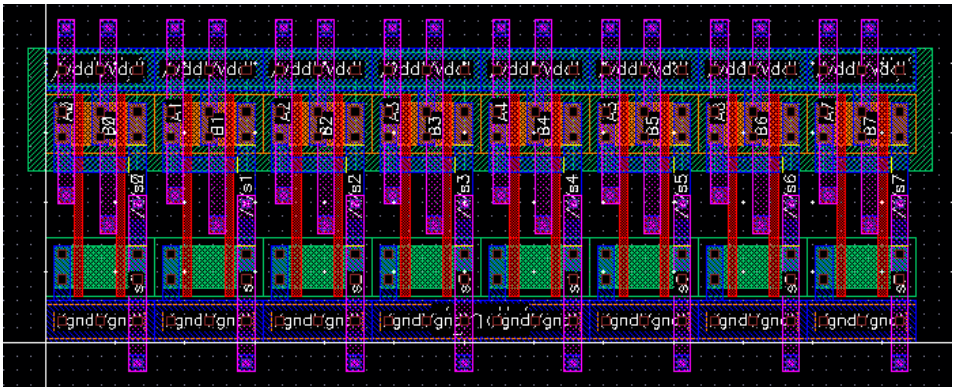

Figure 5. NAND8 Layout

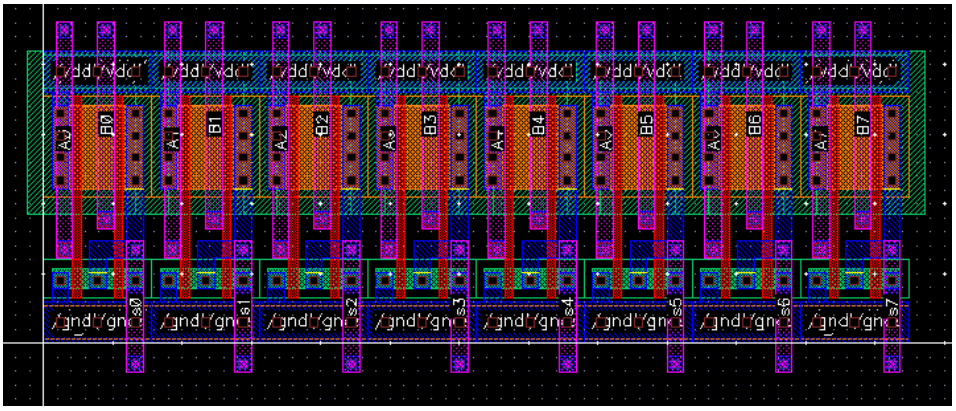

Figure 6. NOR8 Layout

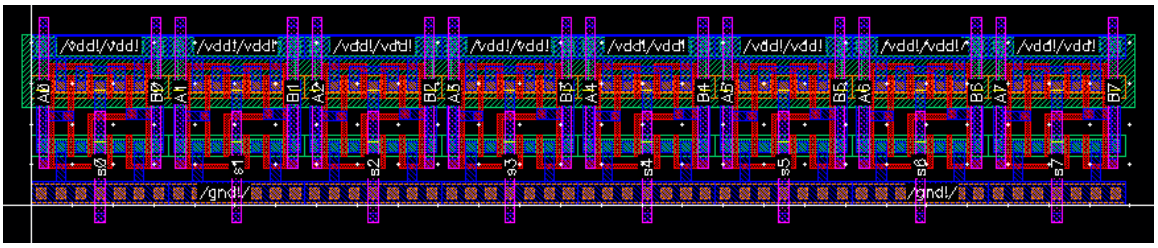

Figure 7. XOR8 Layout

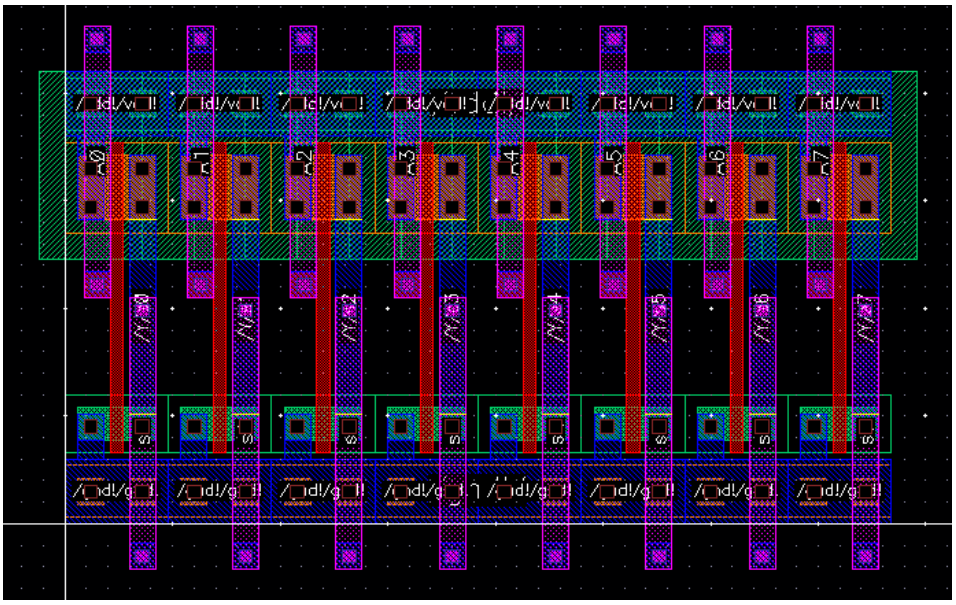

Figure 8. INV8 Layout

After that, we can find the ALU at Figure 9, which all logical functions and adders are combined.

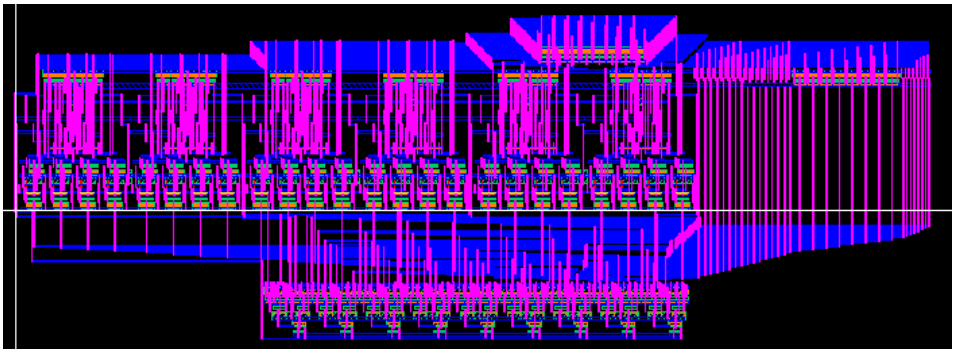

Figure 9. ALU Layout

Several challenges arose during the construction of the ALU. Due to the large size of the adder, routing became a significant issue. Initially, I placed the multiplexers at the center of the layout, but this led to highly tangled wiring. Later, I revised the design to follow the schematic from the original proposal. This adjustment resolved the routing issues; however, it resulted in less optimal area efficiency.

The barrel shifter was designed to support logical shifts of up to 7 bits in an 8-bit environment. It

consists of five stages of multiplexers and implements four opcodes to support both left and right logical shifts. The first and last stages handle the direction of the shift based on a ‘left’ signal, while the middle three stages perform 4-bit, 2-bit, and 1-bit shifts respectively. This configuration enables logical shifts of up to 7 bits in either direction. From Figure 10, we can find the whole design of the Shifter.

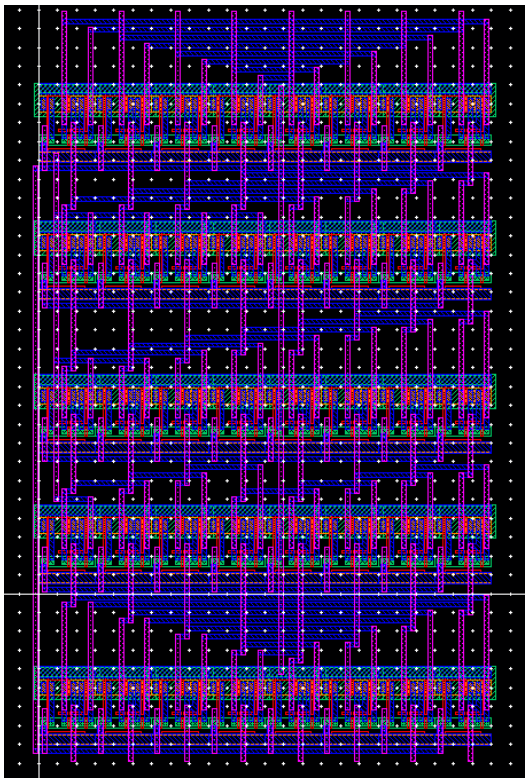

Figure 10. Logical Barrel Shifter

Additional supporting units were also developed. To implement an 8-bit latch that stores and transfers data based on clk1 and clk2 signals, I first designed a 1-bit D latch using NAND2 gates. This latch is enabled when clk = 1, allowing data to be transferred. Although typical D latches provide both Q and $\overline{Q}$ outputs, only Q was used in our design, and $\overline{Q}$ was omitted. This basic latch was then expanded into an 8-bit latch. From Figure 11 to Figure 13, we can find the latch and latch8s’ schematic and layout design.

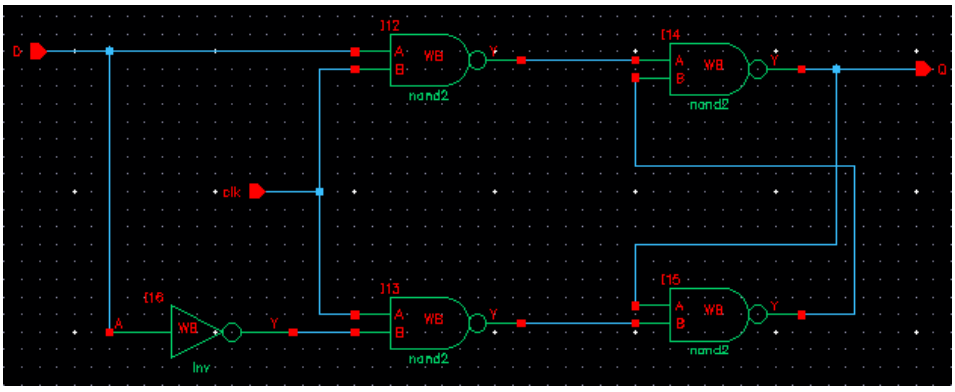

Figure 11. 1-bit Latch Schematic

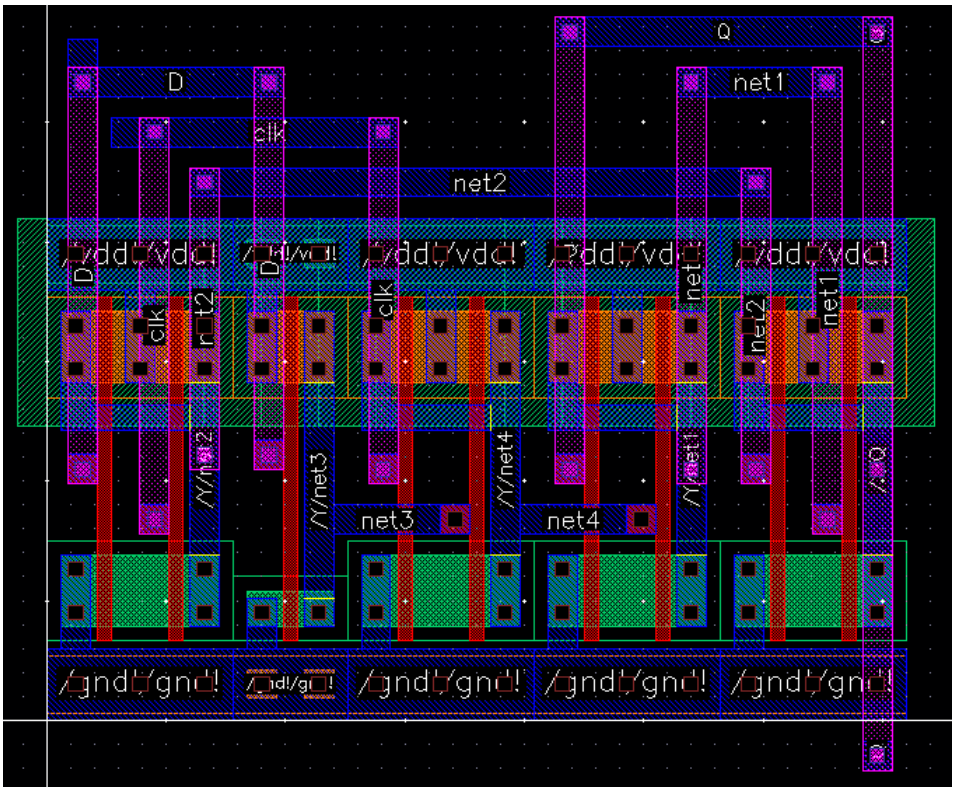

Figure 12. 1-bit Latch Layout

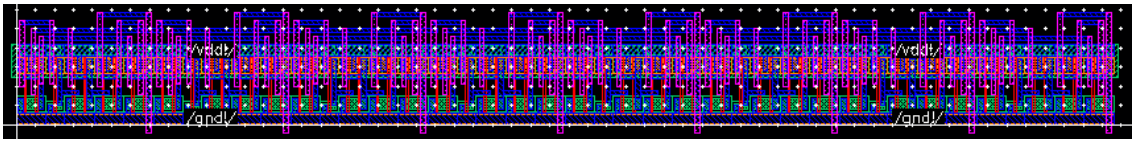

Figure 13. 8-bit Latch Layout

Finally, I implemented a 16-to-8 MUX using eight 2-to-1 MUXes. Based on the rfs signal, the MUX selects either the md0–md7 signals when rfs = 1, or the Y0–Y7 signals when rfs = 0. Figure 14 shows a schematic and a layout of 16-to-8 MUX.

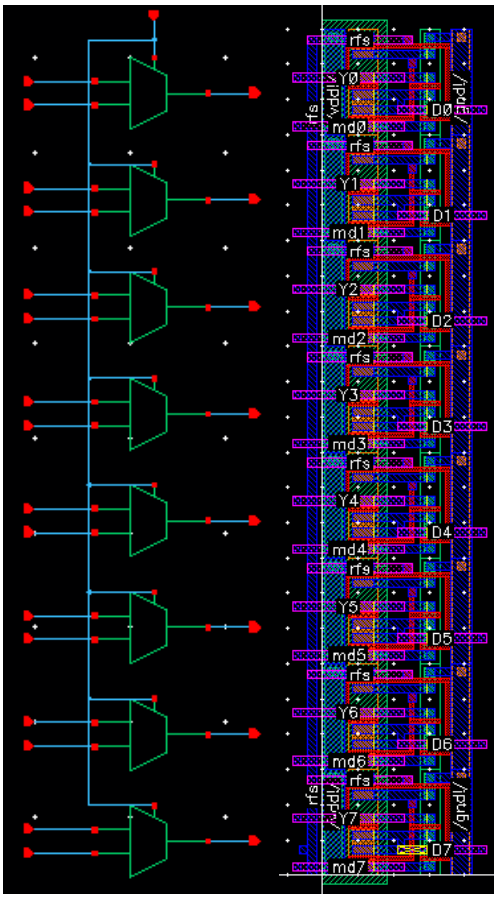

Figure 14. Schematic and Layout of the 16-to-8 MUX.

Now, we will have to talk about the delay that has come from the post-layout simulation. Regarding simulation, we measured the critical arithmetic delay and critical logical delay within the ALU, as well as the overall critical delay in the combined ALU and shifter structure. Prior to running the simulation, we predicted that the subtract operation in the arithmetic path—passing through the XOR8 unit—would have the longest delay. For the logic path, the XOR operation was expected to be the most critical due to the relatively large size of the logic gate and its high propagation delay.

Table 1 and Figure 15 present the measured critical delays within the ALU, which confirm our initial predictions. As shown, the longest delays indeed correspond to the subtract and XOR operations. Table 2 and Figure 16 summarize the critical delay in the combined ALU and shifter design. The results show that the subtract and XOR operations remain the most critical, while there is little difference between the delay of a direct pass-through versus an actual shift operation within the shifter. Although the ALU performs arithmetic operations involving carry propagation, the shifter contributes relatively little to the overall delay. This is expected because the barrel shifter is implemented using a logarithmic MUX network with minimal gate depth, resulting in nearly constant and small propagation delay across different shift operations.

	A+B	A-B	A+1	B+1	A NAND B	A NOR B	A XOR B	INV A
Delay	25.60793	28.509	24.12966	18.3457	11.3519	7.9484	12.8547	7.92731

Unit: ns

Table 1. Arithmetic and Logical Delay of the ALU

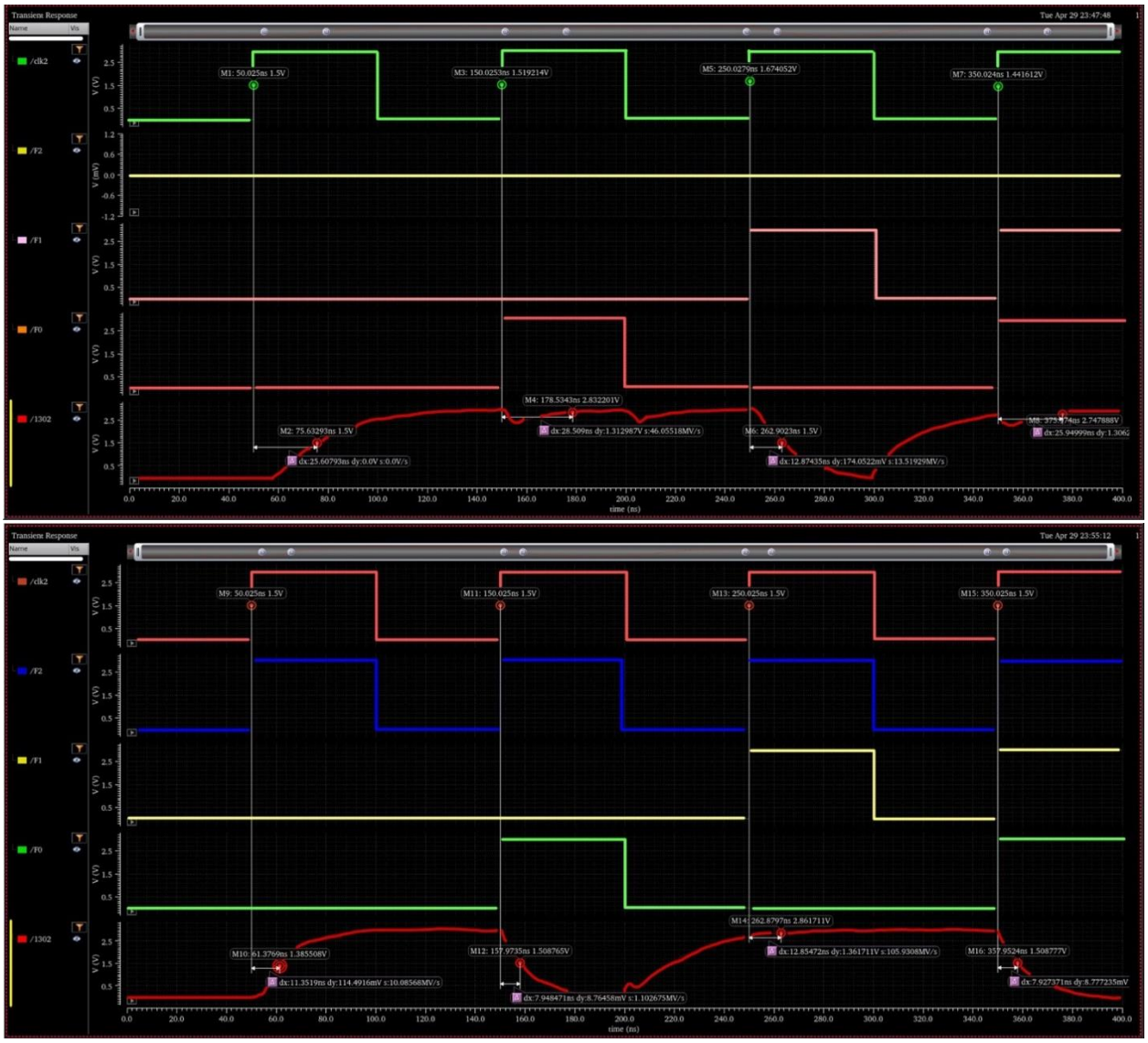

Figure 15. Arithmetic and Logical Delay of the ALU

	A-B	A XOR B
Pass (F6 = 0)	31.10923ns	30.89078ns
Pass (F6 = 1)	28.94617ns	22.76173ns

Table 2. Critical Delay of the ALU + Shifter

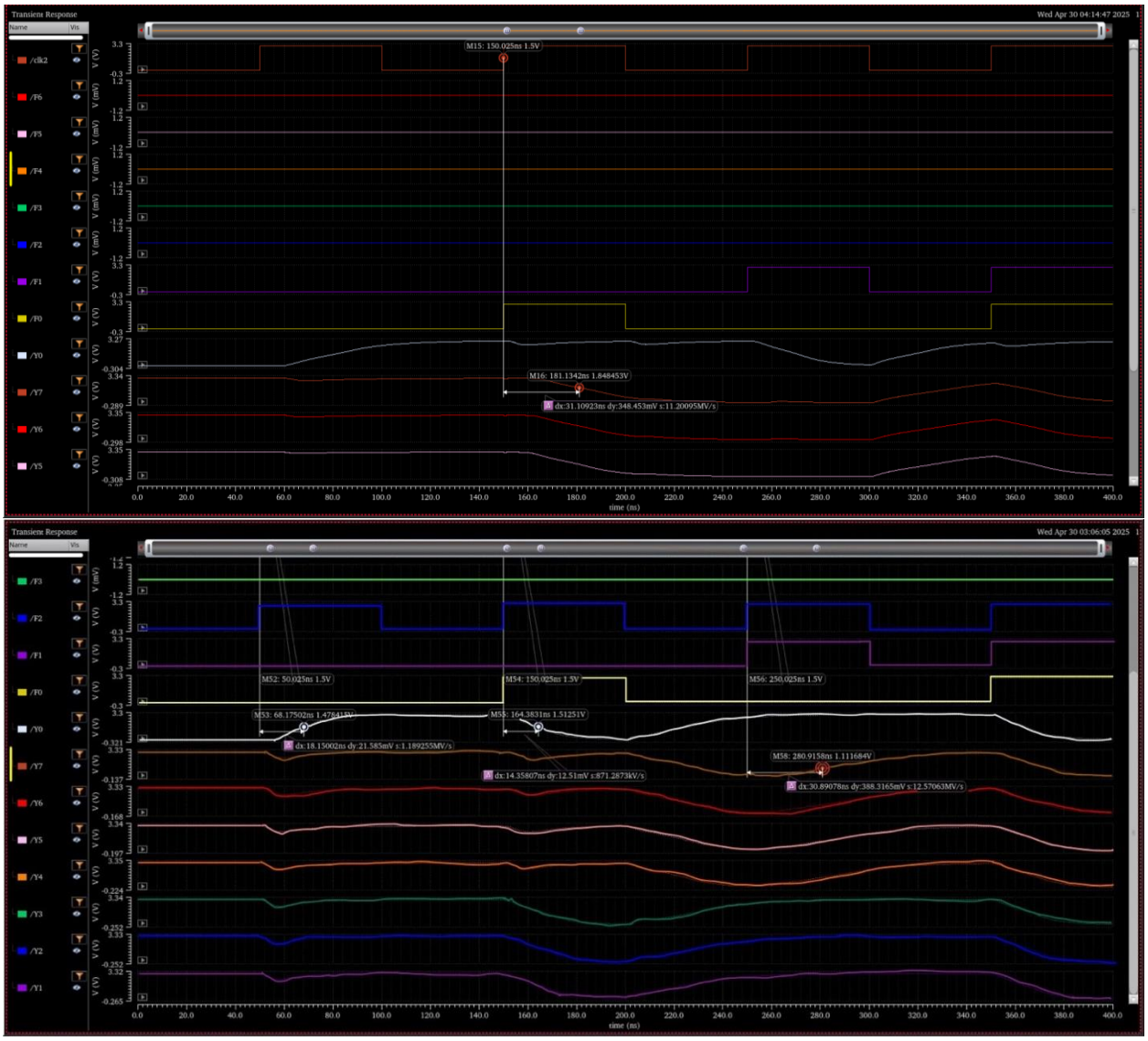

Figure 16. Critical Delay of the ALU + Shifter

Through the tables and figures, we can confirm that the subtract operation, the XOR gate, and the subtract operation passing through the shifter represent the most critical delay paths.

Then, we can now have the size of the layout and number of transistors. For Figure 17, we can find the whole design and its size. The layout resulted in a width of 1182.3 and a height of 855.3, indicating a relatively large design. This aspect will be addressed further in the improvement section.

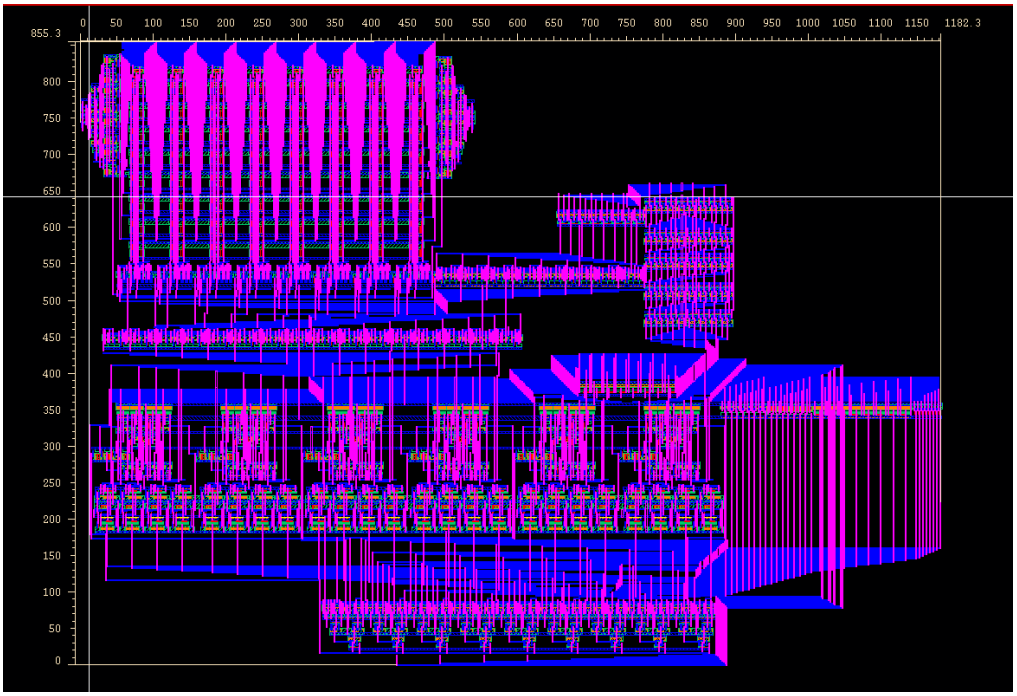

Figure 17. Size of the whole Layout

For this design, there are 1961 pmos and 2169 nmos, so we use a total of 4130 transistors.

Figure 18 shows the result of the LVS, which contains the number of transistors.

Figure 18. The number of transistors and LVS

Table 3 shows the ALU truth table (for Exam 2), while Table 4 presents the truth table for the shifter. Along with the shifter's truth table, Figure 19 illustrates its operation. Given an input of 11111111, we can observe how the logical shift right (LSR) and logical shift left (LSL) functions behave.

F2	F1	F0	Operation
0	0	0	A+B
0	0	1	A-B
0	1	0	Increment A
0	1	1	Increment B
1	0	0	A NAND B
1	0	1	A NOR B
1	1	0	A XOR B
1	1	1	NOT A

Table 3. Truth Table of the ALU

Left	F5	F4	F3	Operation
0	0	0	0	Pass
0	0	0	1	LSR1
0	0	1	0	LSR2
0	0	1	1	LSR3
0	1	0	0	LSR4
0	1	0	1	LSR5
0	1	1	0	LSR6
0	1	1	1	LSR7
1	0	0	0	Pass
1	0	0	1	LSL1
1	0	1	0	LSL2
1	0	1	1	LSL3
1	1	0	0	LSL4
1	1	0	1	LSL5
1	1	1	0	LSL6

1	1	1	1	LSL7

Table 4. Truth Table of the Shifter

Figure 19. Simulation of the Shifter

Now, we can find the final schematic and symbol of the 8-bit Microprocessor from Figure 20 and Figure 21.

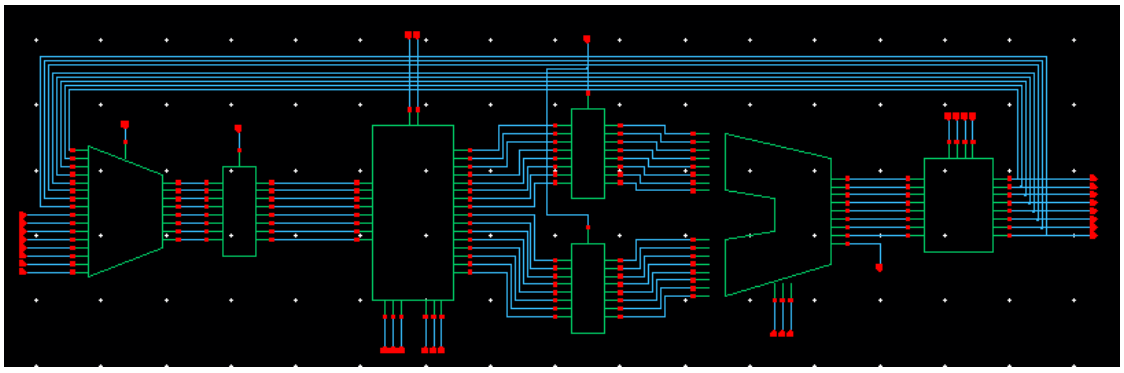

Figure 20. Schematic of the 8-bit Microprocessor

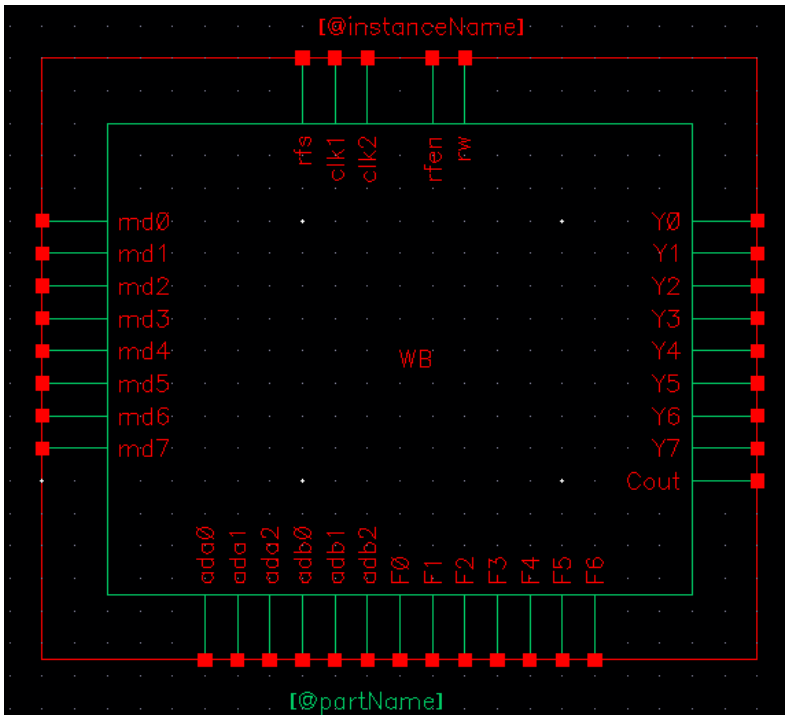

Figure 21. Symbol of the 8-bit Microprocessor

4. Conclusions

In this project, we successfully designed and implemented an 8-bit microprocessor by constructing key components such as the ALU, barrel shifter, and SRAM. Throughout the design process, special attention was given to layout optimization, hierarchical structuring, and DRC compliance. Simulation results met our expectations, with subtract and XOR operations exhibiting the longest delays due to their structural complexity. The combined ALU and shifter critical path also aligned with our predictions, and overall functionality was verified through truth tables and waveform analysis.

During the LVS process, most issues were related to drain/source connections and net mismatches. However, the most critical problem involved the declaration of VDD and GND. Interestingly, LVS still reported a successful match even when VDD and GND were not explicitly re-declared—likely because LVS primarily checks connectivity rather than functional correctness. To resolve this, I created new VDD and GND jumpers to match the number of terminals. This adjustment allowed the post-layout simulation to produce correct and reliable results.

While the design met its functional goals, the overall area was relatively large, as seen from the layout dimensions. Several factors contributing to the large layout area have been identified and summarized below.

a) **Adder size**: The adder was significantly large. This inefficiency primarily stemmed from stacking gates vertically across layers, rather than organizing them in a more compact, sequential layout. The inefficiency was realized too late in the process. Routing all inputs and outputs externally would have been a more effective approach.
b) **Limited fine-grained area control**: Although efforts were made to comply with DRC constraints, precise control over the area at a detailed level was not fully achieved.
c) **Design and gate-level limitations**: There were inherent constraints in the basic gate designs. While multiple references were reviewed, time constraints prevented full exploration and optimization.
d) **SRAM structure understanding**: While the functional behavior of the SRAM was understood (at Exam3), the internal structural design was not fully grasped. Although lecture materials and references guided the implementation, a more efficient structure likely exists.
e) **Metal layer usage for I/O**: Better application of metal layers for signal input/output could have improved overall layout efficiency.
f) **Design Prediction**: Hierarchical design seemed to be the core principle of this project, and in retrospect, the low-level design did not fully support that hierarchy as effectively as it could have. Had I considered the wiring and interconnections more carefully with the next stage of the design in mind, I believe a more efficient and streamlined layout could have been achieved.
g) **Deeper Reflection on the Design Process**: Each time I attempted a design, I spent considerable time anticipating the layout and predicting simulation results. Despite this effort, I often struggled to come up with more efficient design alternatives. This experience highlighted the importance of developing a habit of continuous reflection and deeper analysis, especially through revisiting lectures and thoroughly thinking through each design decision.

5. **Feedback on the Final Project**

Due to the significant time investment required, I often felt overwhelmed working alone on this project. My ideas were limited, and the only sources of additional information were the lectures and online searches. In hindsight, if I had been able to consult someone with prior experience in similar design projects, the process would likely have been more manageable and efficient.

6.**Advice for Future Students**

Anyone planning to take this course should be prepared to approach it with strong commitment and a sense of responsibility. Time management is absolutely essential, as the workload demands significant time investment. However, the moment you pass LVS, all the challenges feel worth it—it’s an incredibly rewarding experience. The course offers substantial theoretical insight, and the hands-on design work encourages deep, independent thinking. If you're looking to strengthen your design skills, I wholeheartedly recommend taking this course.