EE 431 Final Lab

Pulse Width Modulation Driver

Report by:
Ethan Vosburg (evosburg@calpoly.edu)
Wyatt Tack (wtack@calpoly.edu)

December 5, 2024

Table of Contents

1 Introduction

Pulse width modulation (PWM) is a very common standard that is used when controlling peripherals or signals that are binary. For example, an LED is blessed driven either at full power or not power but through PWM signals we can change the perceived brightness of the LED while still only supplying either a high or low voltage. In this lab, we built a PWM driver that can take in a 4-bit value, and then output a PWM signal.

2 Background

Pulse wideth modulatio or PWM is a tequniue that is used when one needs to deal with a binary signal but would like to encode analog information. With this method, a period is chosen to transfer data on and then the duty cycle is modulated in order to send data. As seen in Figure 1. In this diagram, you can see a consistant period that is then changed thougough 3 differnet states.

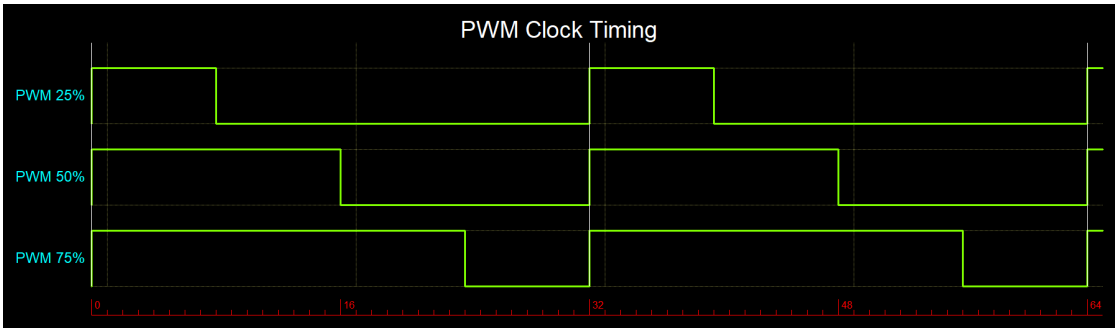

Figure 1: PWM Timing Examples

3 Design

The design process began with brainstorming on a whiteboard as shown in Figure 2 from here we determined that we needed to build a comparator, counter, and then use some basic logic gate to logicically combine these components. In our initial design, we made some errors in the specific gates that we needed but this was later resolved in the development.

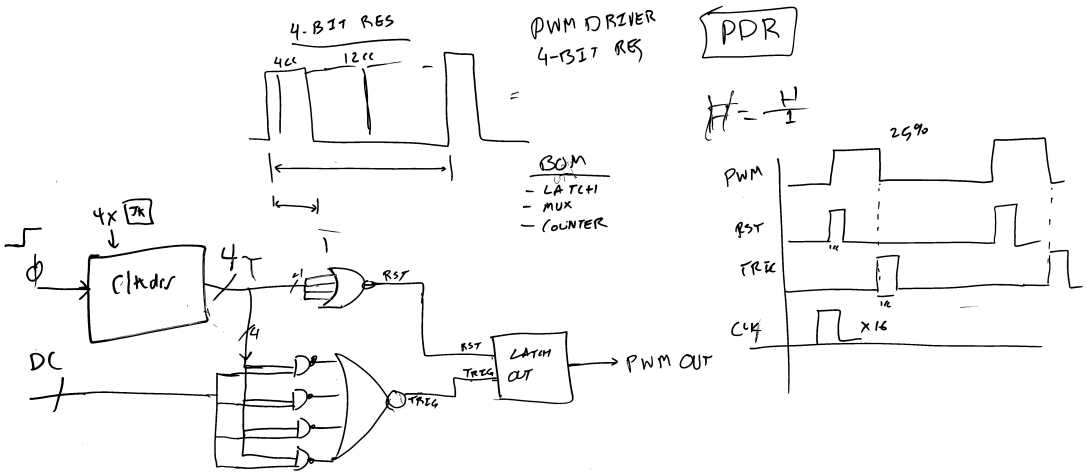

Figure 2: Initaial Brainstorm Design

3.1 Timing Diagram

From our brainstorming, the timing diagram remained the guiding principle. This is better shown in 3 which shows a proper clock diagram where we are using a reset signal and a comparator signal to drive some sort of data latch that would be able to output a PWM waveform. With this in mind, we could now start to work out the functional diagram that we would need to achieve the desired functionality.

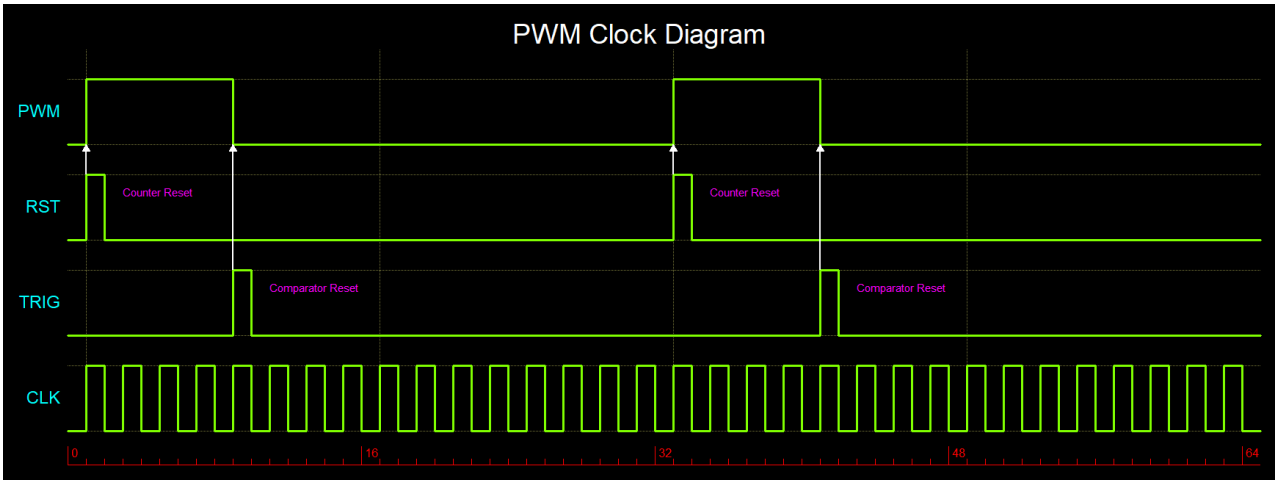

Figure 3: PWM Clock Diagram

3.2 High Level Design

The final design we went with used a 4-bit counter as a clock divider that constantly counted up and fed its output to a comparator which would then wait until the desired signal was output and then trigger the data latch. This worked for the rest of the pulse but to bring the signal up, we needed a trigger on the MSB of the 4-bit counter. This was achieved with a monostable circuit that would set the data latch after every full count. Together these two components would feed a data latch that simply went between V_{cc} and V_{ss}.

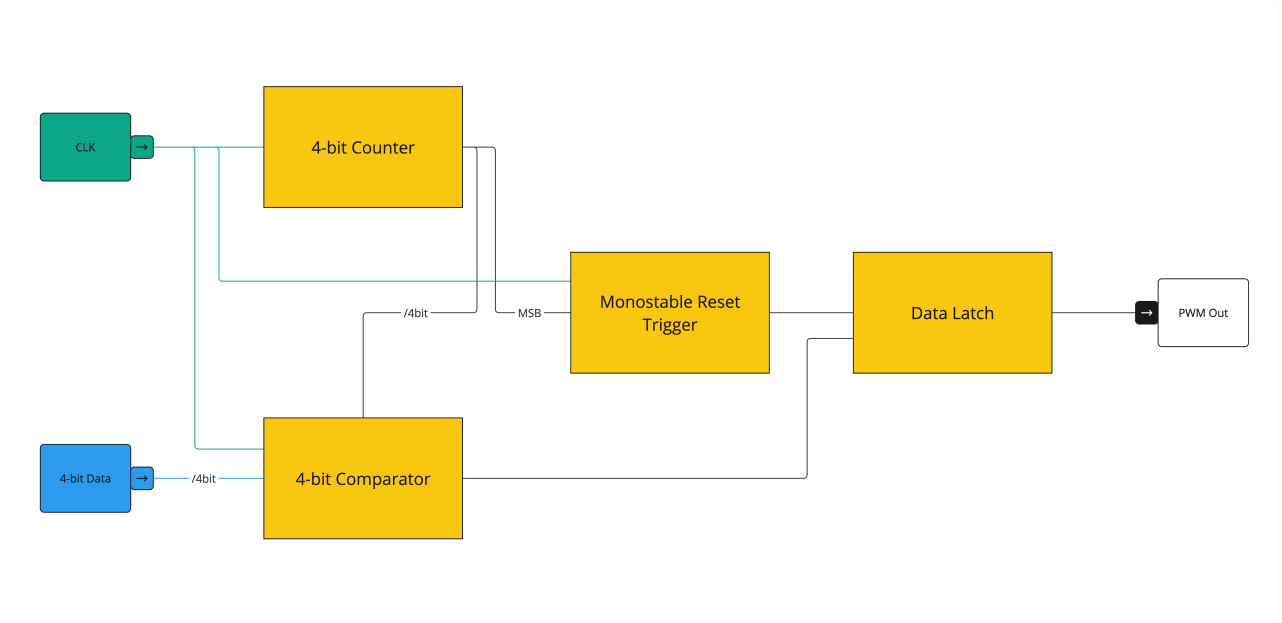

Figure 4: PWM Functional Overview

Pulling this all together, we get this final high-level block that takes power, data, and a clock and can output a PWM signal as shown in Figure 5.

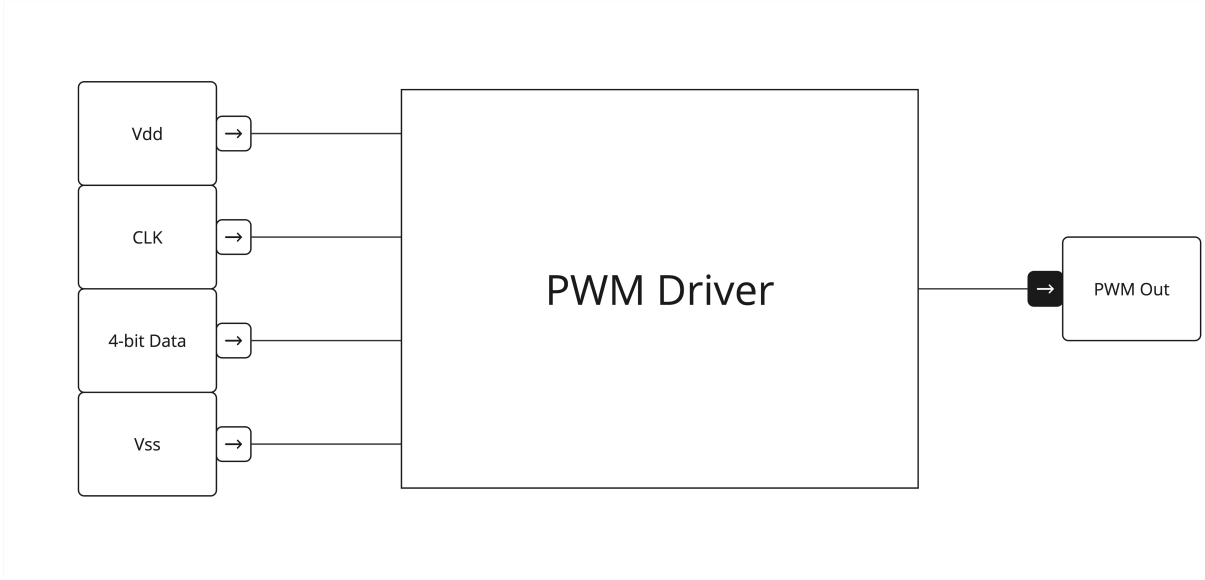

Figure 5: High Level PWM Block

4 Schematic

4.1 Flip Flips

In this design, we needed flip flops for storing the state of the output as well as for some noise mitigation that we found

4.1.1 D Flip Flop

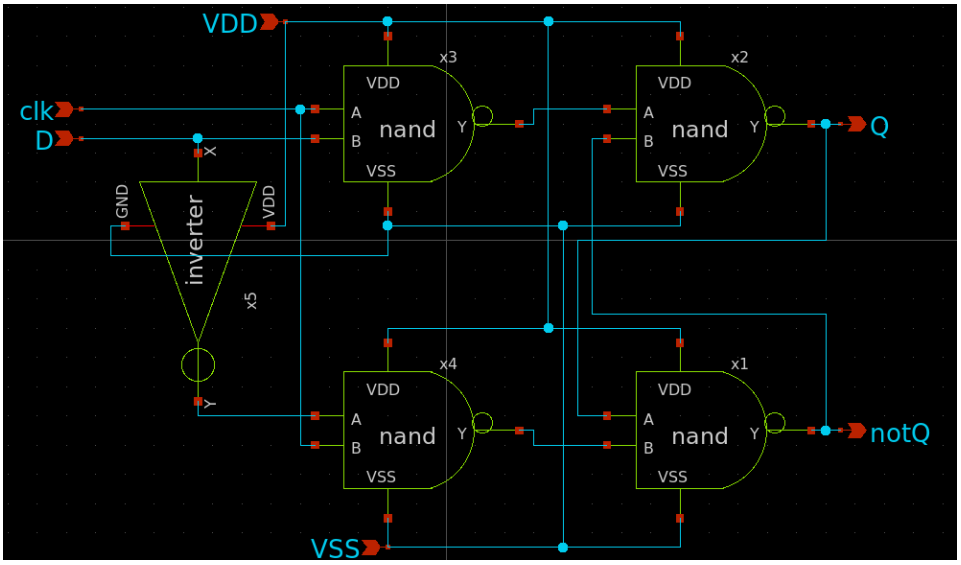

Figure 6: D Flip Flop Schematic

4.1.2 Edge Trigger D Flip Flop

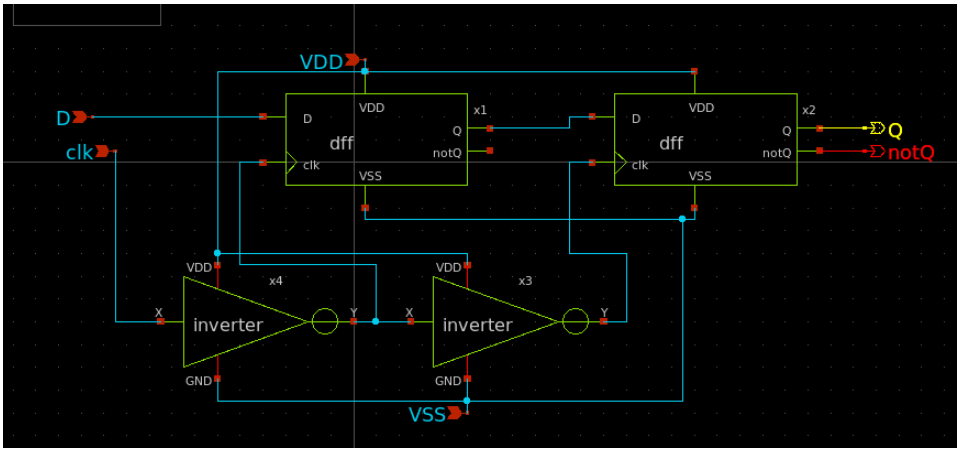

Figure 7: Edge Trigger D Flip Flop Schematic

4.2 Counter

With the construction of the two different flip-flop circuits, we could now put those to use in a 4-bit counter to make a clock divider. This was done by putting 4 D flip-flops in series and wiring the LSB to the clock line which would then continually slow down the clock signal.

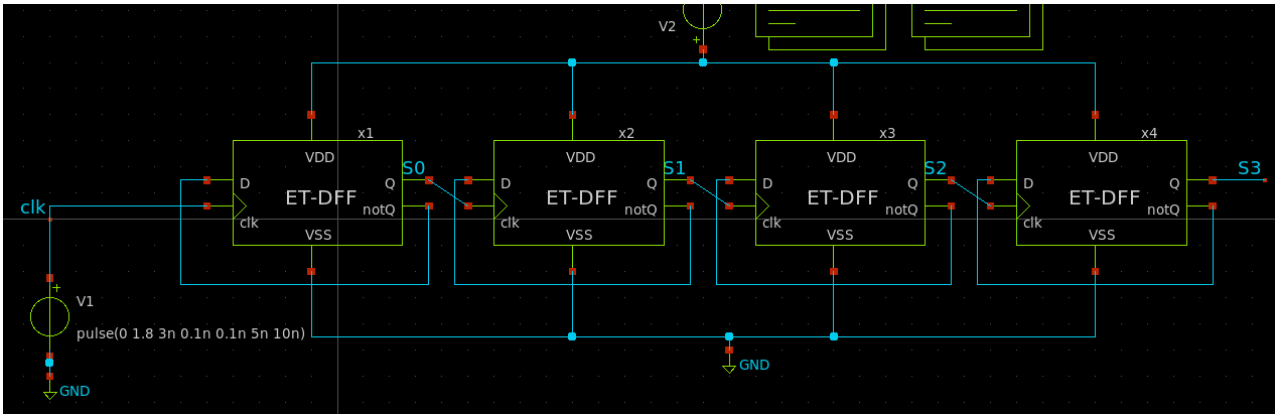

Figure 8: Counter Schematic

4.3 Comparator

With the clock divider taken care of, we could not look to the second largest part of the circuit, the comparator. The comparator was built up of XNOR gates, which acted as equivalence evaluators for each bit. We could then combine the outputs from these evaluators with gates to get a single-bit output that would tell us if the two signals are the same. The XNOR schematic can be seen in Figure 9 and with the final comparator schematic in Figure 10.

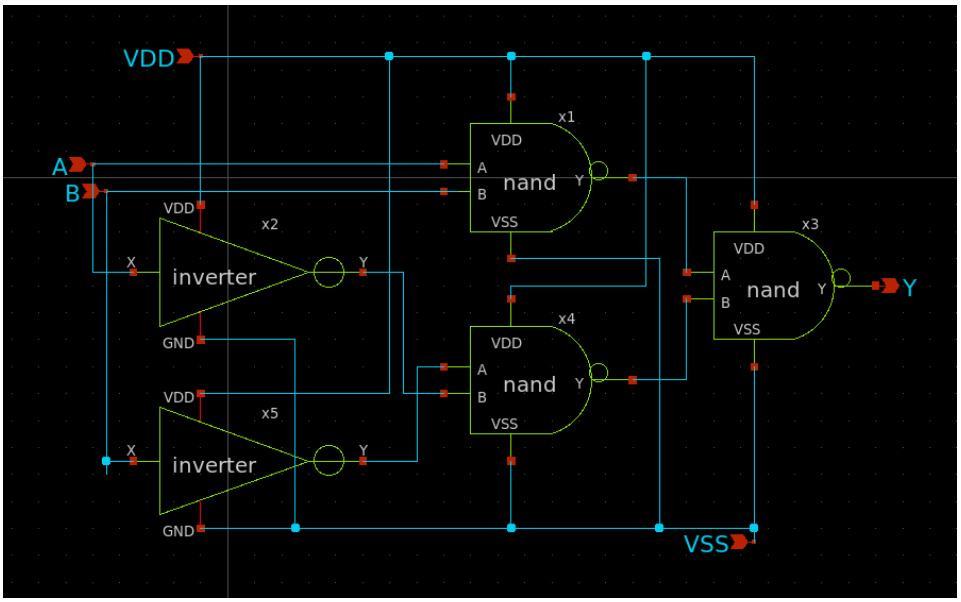

Figure 9: XNOR Schematic

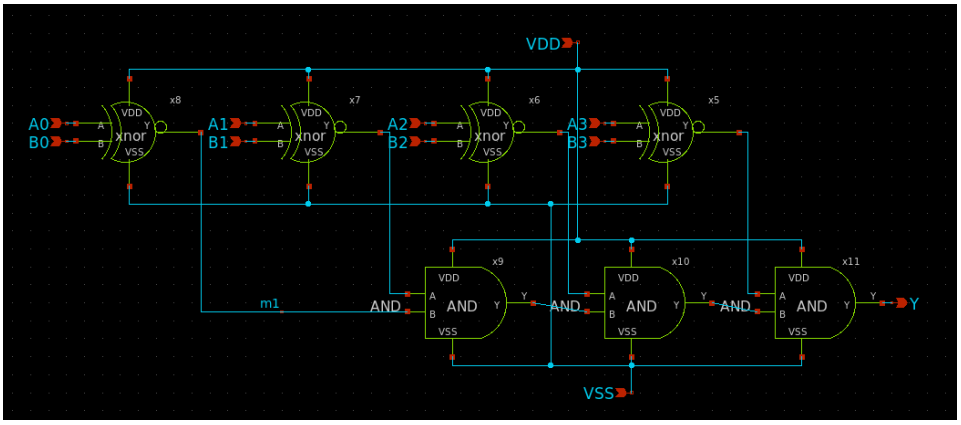

Figure 10: Compartor Schematic

4.4 PWM Driver

Adding all of these components together, we get the final PWM driver schematic. There were a few notable inclusions that we had to add while going through the design process. Since we were no longer dealing with the prefect world, we needed to account noise that might be present in the signals that we were passing. In order to handel this, we used a D Flip Flop acting as a filter to trim away unwanted noise. This can be seen in Figure 11.

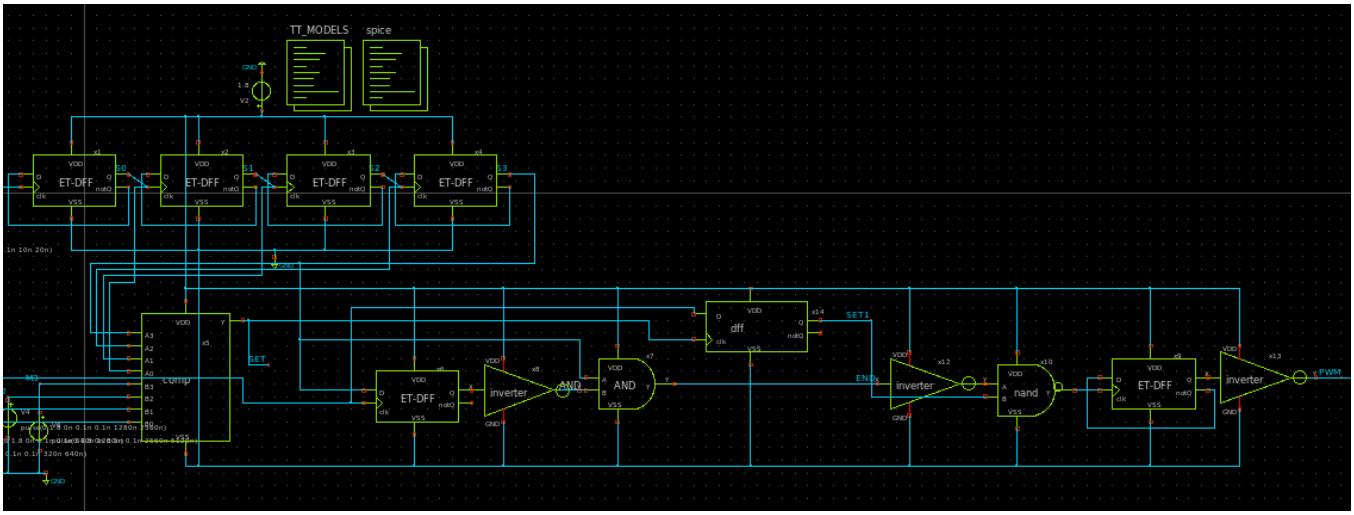

Figure 11: PWM Schematic Before Bubbeling

We noticed that we would be able to combint and push bubbles to make the schamatic smaller, more efficient, and use better thought out so we went through and optimized the placment of the gates and the type of gates. For example, we fond that in one place in out circuit, we were using a AND gate connected to an inverter, replacing this with NAND gate was an easy simplification. The final design that we setteled on can be seen in Figure 12.

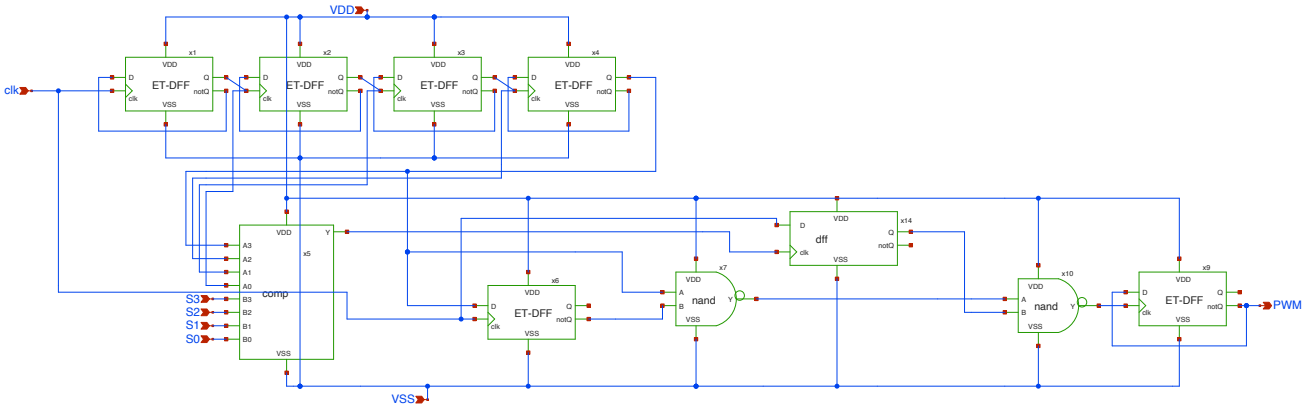

Figure 12: Inverter Layout in Xschem

5 Layout

5.1 Flip Flops

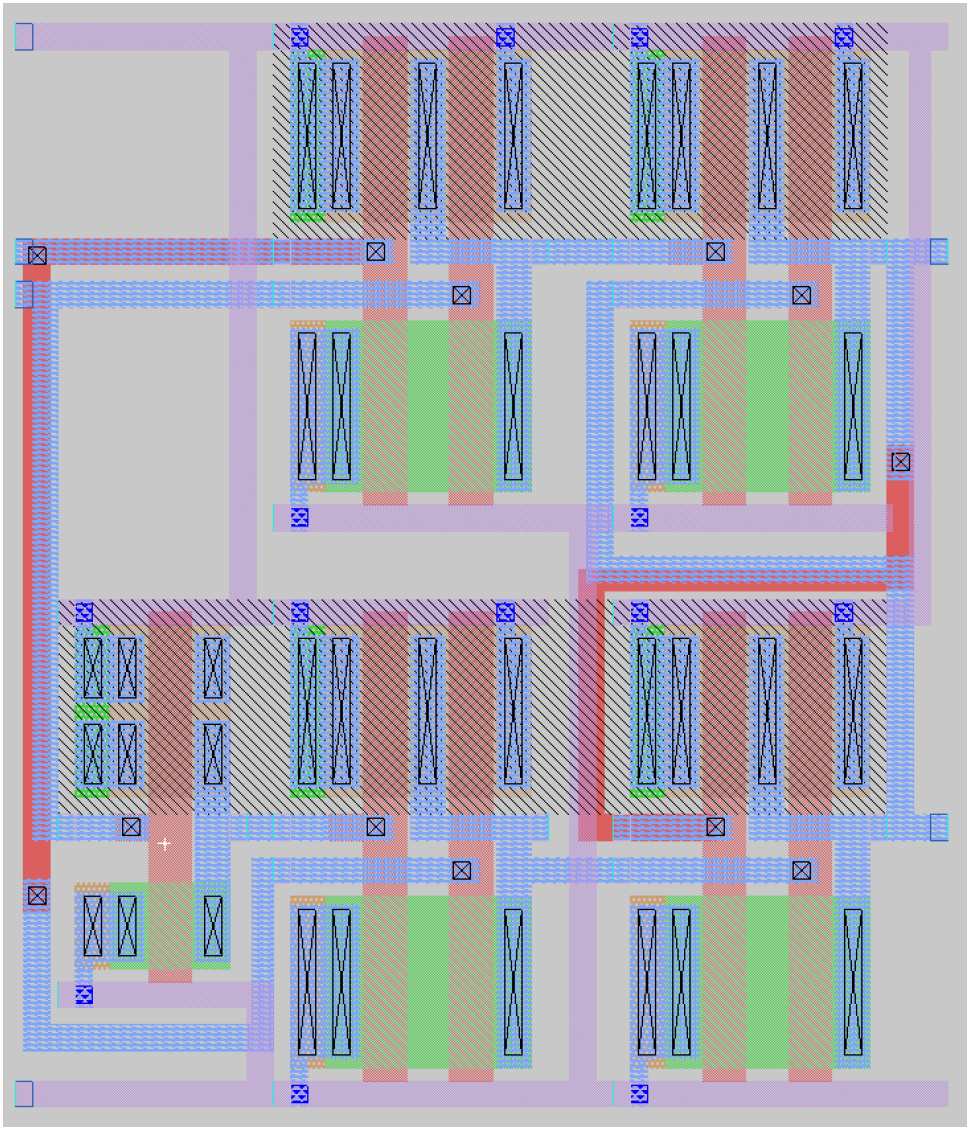

Figure 13: D Flip Flop Layout

Figure 13 shows the D Flip Flop layout. Four NAND gates make up a 2x2 grid on the right side of the image with an inverter on the bottom left.

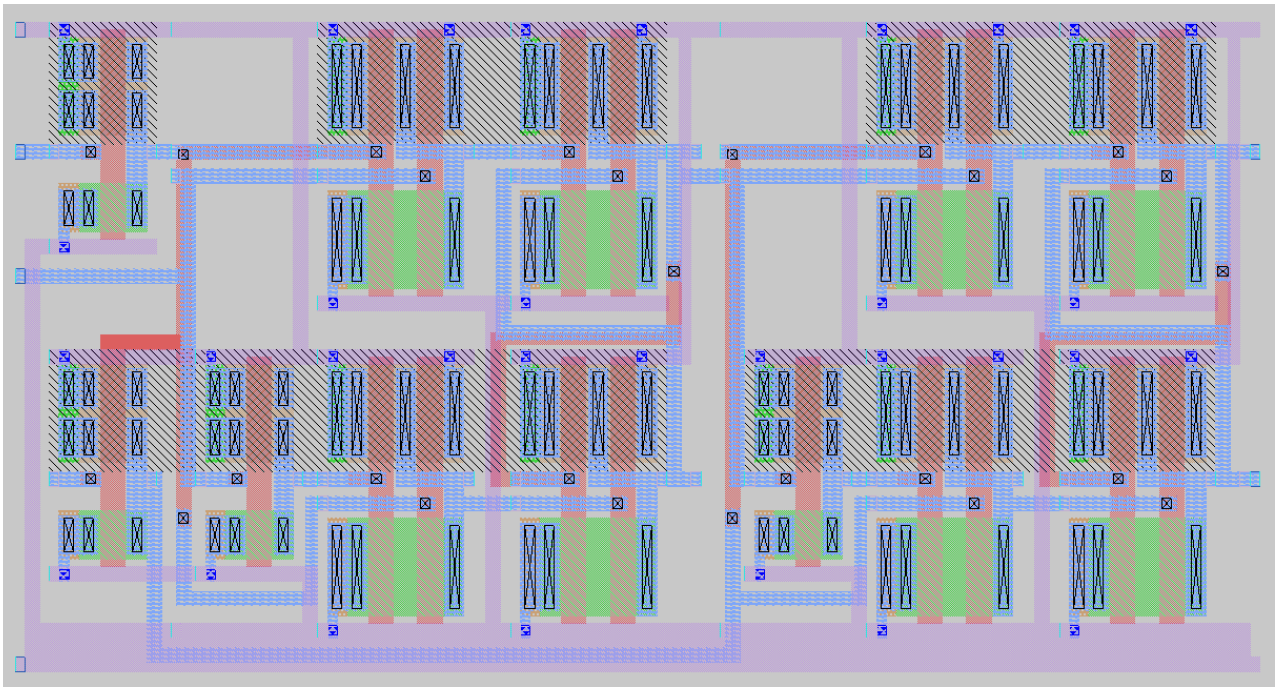

Figure 14: Edge Trigger D Flip Flop Layout

Figure 14 shows an Edge Trigger D Flip Flop which becomes the back bone of the counter. This shows a 2x2 grid of D Flip Flops that are all connected together with two inverters on the far left of the layout.

5.2 Counter

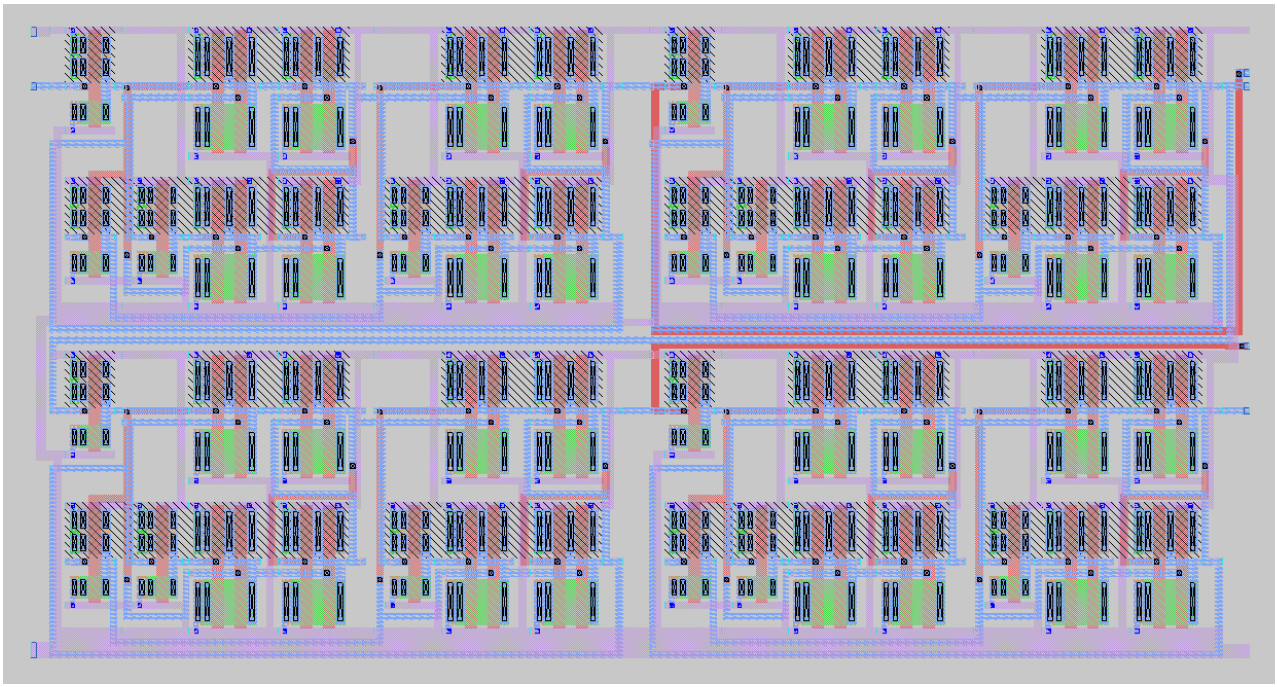

Figure 15: Counter Layout

Figure 15 shows the layout for the counter. If you look closley, you can see the 4 Edge Trigger D Flip Flops that are being wired together in a 2x2 grid. These are operating as a clock divider outputting a pulse every 16 clock cycles but it is also outputting

5.3 Comparator

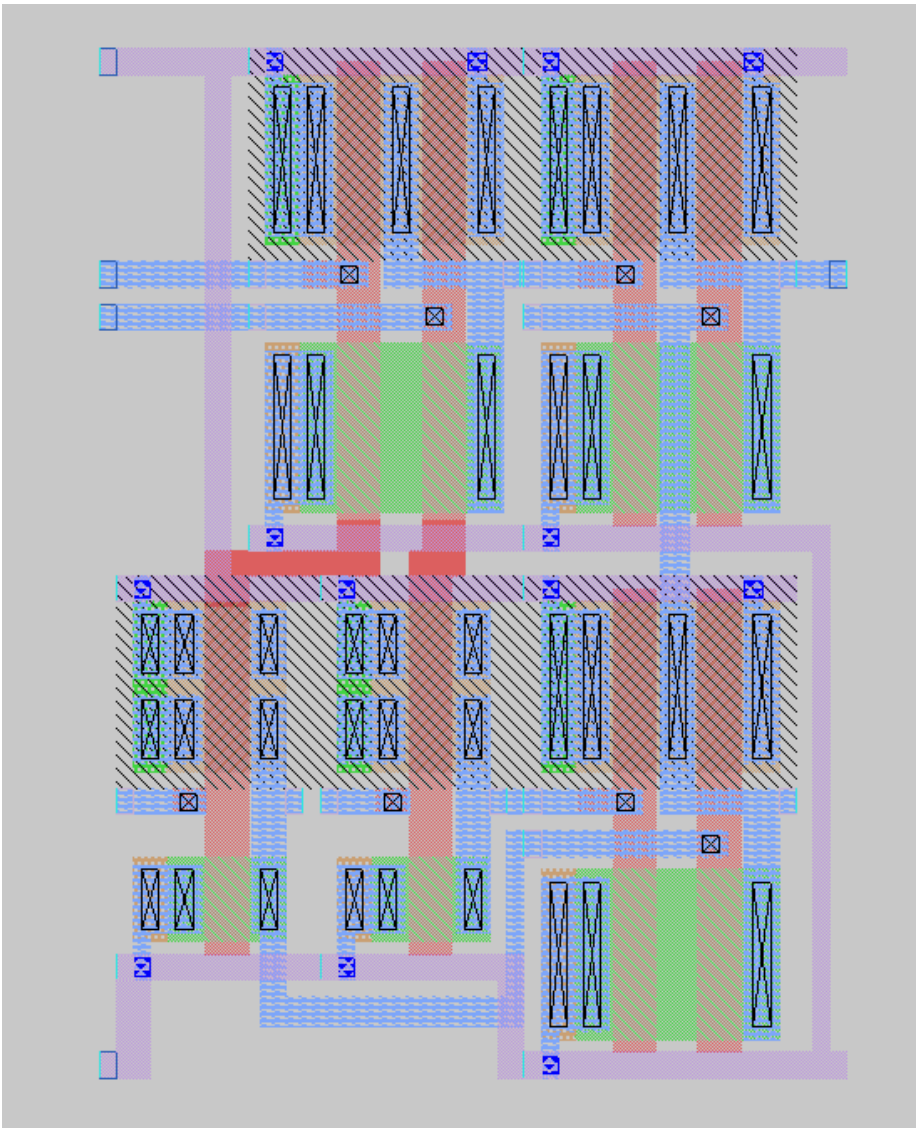

Figure 16: XNOR Layout

Figure 16 shows the layout for an XNOR gate. We can see 3 NAND gates with 2 on the top of the design and then one on the bottom right of the design with two inverters to the left of the lower XNOR.

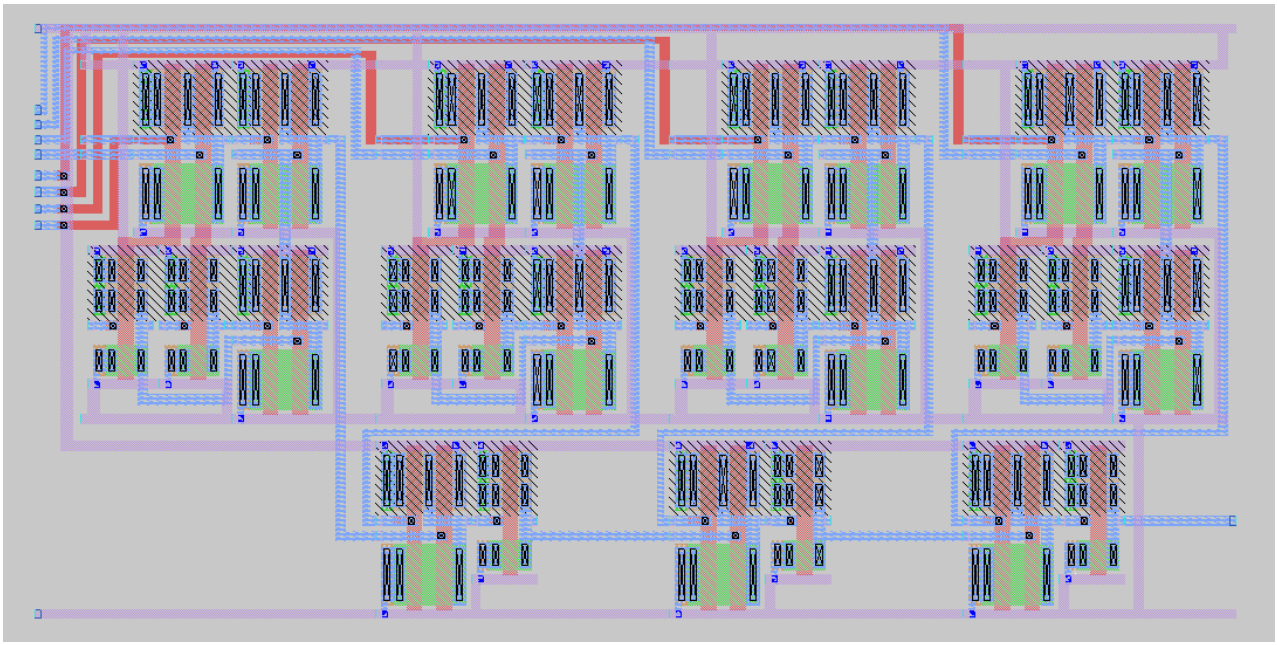

Figure 17: Comparator Layout

Figure 17 shows the layout for the comparator which uses 4 XNOR gates as equivilence logic and then combines them together with 3 AND gates at the bottom of the layout.

5.4 PWM Driver

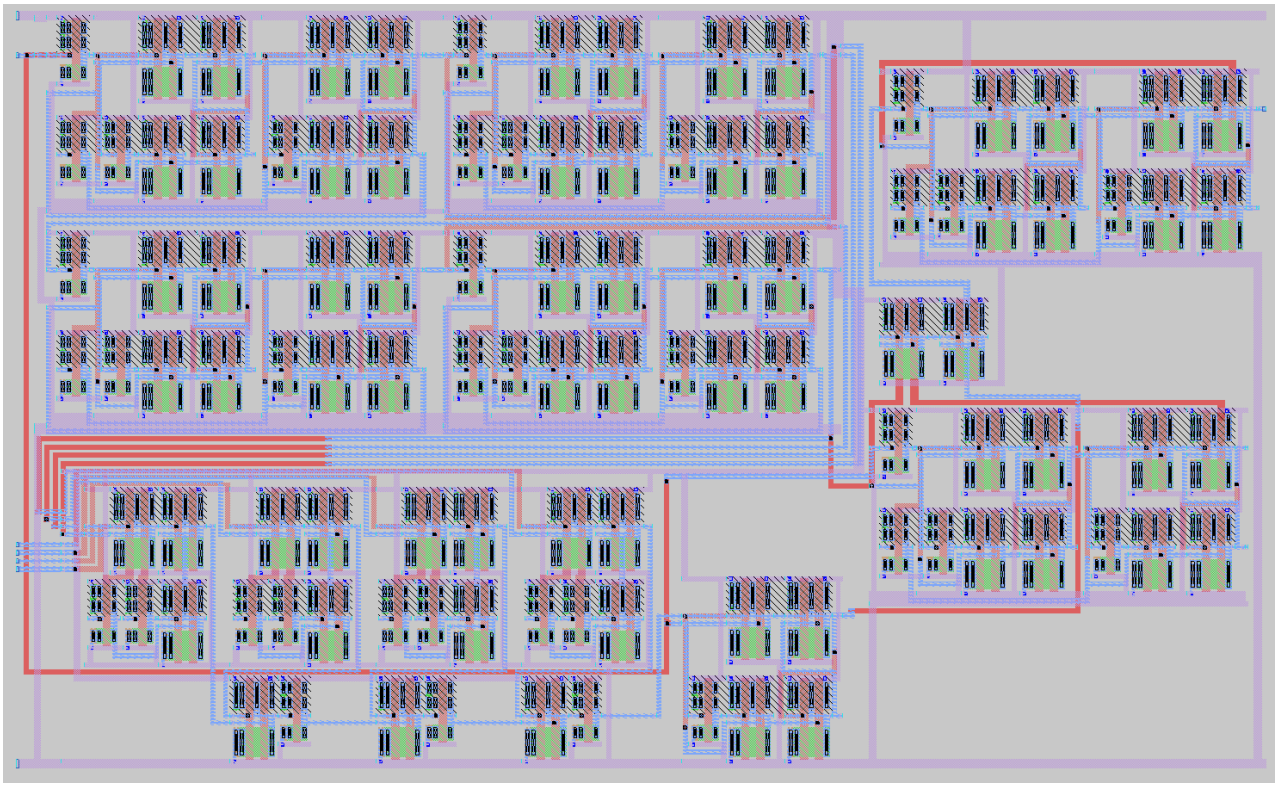

Figure 18: Full PWM Driver Layout

Figure 18 shows the final completed layout. In the top left, the counter is seen as it takes in a clock signal and outputs bits to the comparator and then the MSB to the Mono stable circuit shown in the top right. The comparator is located on the bottom left of the layout with the 4-bit input and the bits out from the counter. This leaves the data out latch on the lower right of the schematic with some supported gates located north of it where the signal is eventually outputted.

6 Verification

6.1 Simulation

After simuting our Xschem schematic in Figure 19 and the Magic layout in Figure 20 we can see that the two circuits are operating idetically.

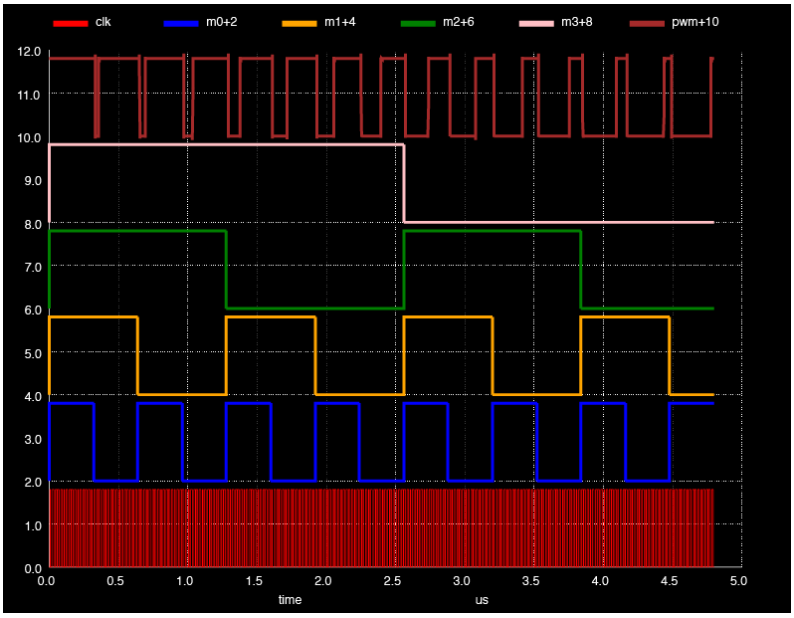

Figure 19: Xschem PWM Driver Simulation

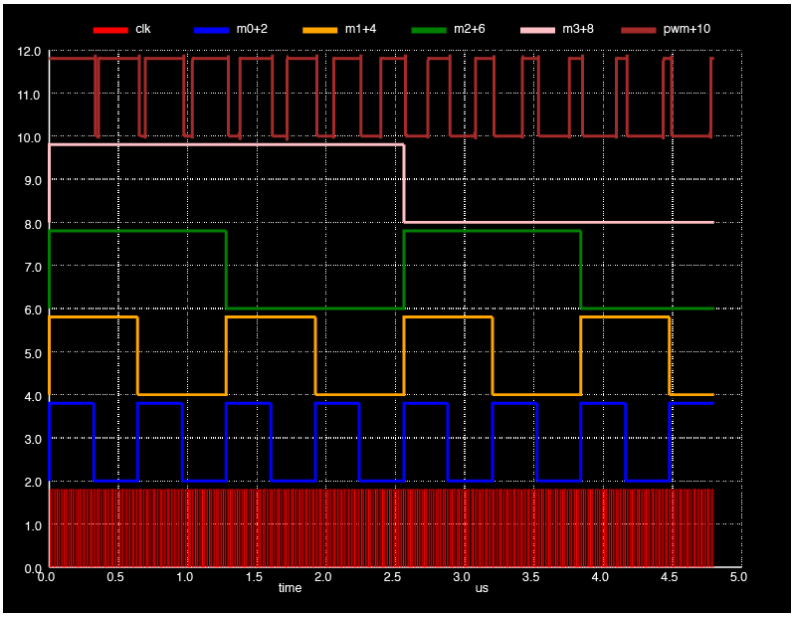

Figure 20: Xschem PWM Driver Simulation

6.1.1 DRC

For the DRC, as shown in Figure 21 we can see that the DRC Manager in Magic shows that this design is clear of design rule violations. To conform to the manufacturing capabilities of the node we are designing in, there cannot be any errors.

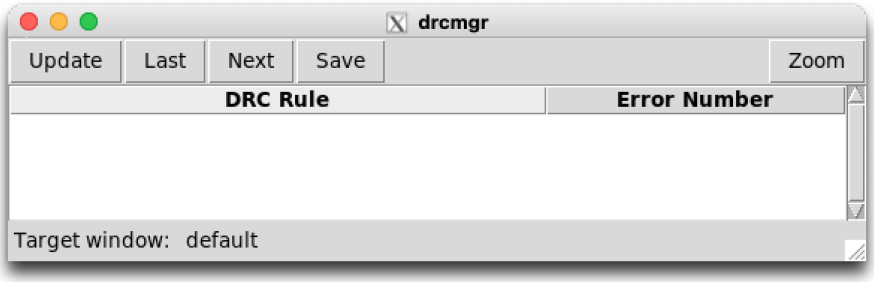

Figure 21: DRC Manager in Magic

6.2 LVS

For LVS, as shown in Listing 1

```
Circuit magic_pwm.cir contains 348 device instances.
    Class: sky130_fd pr nfet_01v8 instances: 174
    Class: sky130_fd_pr_pfet_01v8 instances: 174
Circuit contains 181 nets.
Contents of circuit 2: Circuit: 'xschem_pwm.cir'
Circuit xschem_pwm.cir contains 348 device instances.
    Class: sky130_fd_pr_nfet_01v8 instances: 174
    Class: sky130_fd_pr_pfet_01v8 instances: 174
Circuit contains 181 nets.

Circuit 1 contains 348 devices, Circuit 2 contains 348 devices.
Circuit 1 contains 181 nets, Circuit 2 contains 181 nets.

Final result:
Circuits match uniquely.
.
Logging to file "comp.out" disabled
LVS Done.
```

Listing 1: LVS Check Output

6.3 Size

Figure 22 shows the final area of the layout as well as the transistor count.

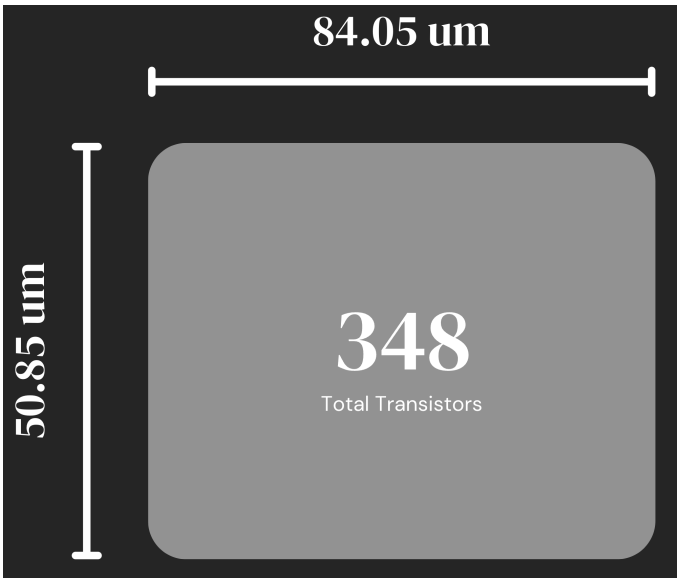

Figure 22: Final Layout Size

7 Discussion

In this lab, we progressed from concept to a working layout that was LVS-checked (Layout Versus Schematic). This process taught us how to design and implement not just schematics but also the underlying ideas. We started by identifying a problem and then breaking it down into manageable components. From there, we determined what we needed to create and, using a bottom-up philosophy, gradually assembled the final driver. Finally, we verified that we had created the correct device by using LVS to ensure our layout matched the working schematic.

With this lab experience, we have acquired all the building blocks necessary to advance and tackle increasingly complex designs. The processes, architecture, and basic gate layout can be scaled up to create larger designs. For example, we could extend the PWM driver to 32 bits or develop different circuits that can communicate with our driver.

Although this lab was successful, some aspects were quite challenging. One major challenge was translating our initial concepts into effective designs. In Xschem and Magic, we needed to create designs that were noise-resistant. Another difficulty was troubleshooting when things did not go as planned. We had to analyze netlists and interpret them to resolve issues with the layouts when the results were not as expected.

Overall, this was a unique and satisfying lab experience that solidified our understanding of layout development in the VLSI world. With this knowledge and practice, we now possess the skills needed to continue our journey in electrical engineering.