Simple CMOS Gates

Aidan Sharpe

I. Introduction

In this lab, we designed and analyzed several simple CMOS gates. Specifically, we worked with an inverter, a transmission gate, and a two-input NAND gate. For each of the gates, a simple workflow was implemented. First, a schematic was created. This was done to create a simple, high-level model of the circuit. Next, the schematic was pulled into a simulation tool to determine delay characteristics. To enable rapid reusability of the design, a schematic symbol was then created. Finally, we designed a physical layout, and checked it to ensure proper spacing and that it matched the schematic.

II. Inverter

The first, and simplest gate we created was a CMOS inverter. This device is made from one n-channel MOSFET (NMOS), and one p-channel MOSFET (PMOS). Its schematic is seen in figure 1. Looking closely, the width of the NMOS is 120nm and the width of the PMOS is 240nm. This was done to match the resistances of the two devices.

The equivalent channel resistance of an NMOS is $R = \frac{R_0}{k}$, and for a PMOS, $R = \frac{2R_0}{k}$ [1]. The constant k is the ratio of the width of the MOSFET to the width of a unit transistor for that technology. For example, the unit transistor for 45nm is 120nm. Therefore, a PMOS will always have twice the resistance of an NMOS for the same width, and if the resistances are to match, then the PMOS must be twice as wide as the NMOS.

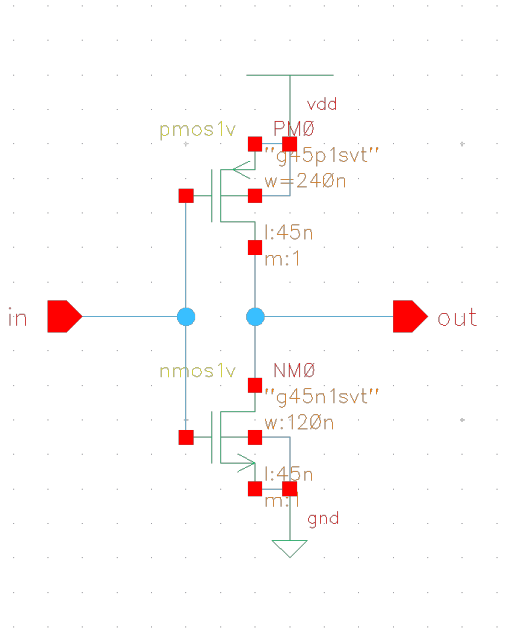

Fig. 1. The schematic for the inverter

With the schematic done, we then analyzed the delay characteristics of the inverter. Specifically, we looked at rise time t_r, fall time t_f, rising propagation delay t_{pdr}, falling propagation delay t_{pdf}, and average propagation delay t_{pd}. The rise time t_r is simply the time it takes the output to rise from 20% to 80%, and the fall time t_f is the time it takes for the output to fall from 80% to 20%. The rising propagation delay t_{pdr} is the time between the input change reaching 50% and a rising output change reaching 50%. The rising propagation delay t_{pdf} is the same but for a falling output change. The average propagation delay is simply the arithmetic mean of the rising and falling propagation delays:

$$t_{\text{pd}} = \frac{t_{\text{pdr}} + t_{\text{pdf}}}{2} \tag{1}$$

For our inverter, we simulated a pulsed input with a period of 2[ns], seen as the green trace in figure 2. Visually, the easiest delay to recognize is t_{pdr} and t_{pdf}. They are simply the smallest difference between the green and red traces at 0.5[V]. Specifically, t_{pdf} is the time between a green trace rising edge reaching 0.5[V] and its corresponding red trace falling edge reaching 0.5[V]. Likewise, t_{pdr} is the time between a green trace falling edge reaching 0.5[V] and its corresponding red trace rising edge reaching 0.5[V].

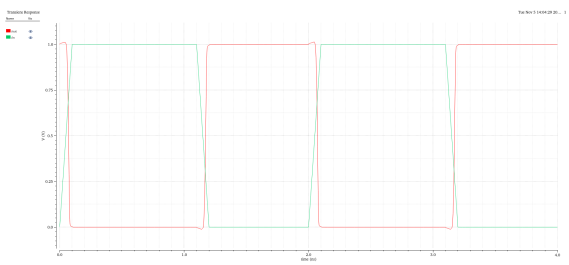

Fig. 2. Simulated inverter input and output signals

Instead of eyeballing the time, we loaded the outputs of the simulation into a calculator tool to precisely calculate the time. This tool was used to calculate all five delay characteristics. The values of the delay characteristics are seen in table I. It is important to note that the rise and fall times of the input signal are identical, but as seen in the table, the rise and fall times for the output are different. This effect is quite odd. The total output capacitance is $3C_0$ regardless of whether the output is rising or falling, and we picked our k values such that the resistance was R_0 for both the pull-up and pull-down network. Therefore, regardless of the edge, the time constant should be $\tau = 3R_0C_0$. Regardless, we still have a difference between the rising and falling edges. The difference likely comes from an assumption we made. We assumed that the resistance for a PMOS is twice that of an NMOS of the same size, but this is merely an approximation to account for holes having lower mobility than electrons[1].

TABLE I
INVERTER DELAY CHARACTERISTICS

t_r	12.33[ps]
t_f	11.98[ps]
t_{pdr}	22.69[ps]
t_{pdf}	21.45[ps]
t_{pd}	22.07[ps]

With these characteristics in mind, we next created a symbol for the inverter. Without going into too much detail, the symbol allows the entire schematic and layout to be packaged into a repeatable block with a known visual indicator. The final symbol is seen in figure 3. In the future, we may place inverters just as we placed individual MOSFETs when creating the inverter schematic.

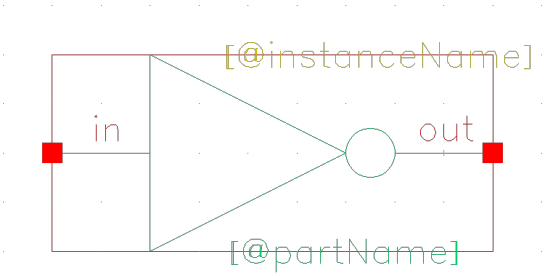

Fig. 3. Inverter schematic symbol

Finally, we created the phyical layout for the inverter. The layout for the MOSFETs is premade with editable parameters. The layout was created with consideration for the width of and spacing between layout elements. Another important factor was the number of contacts between elements. For example, we used three contacts between the metal1 input and the polysilicon gate material to ensure a high quality and low resistance connection. We also used five contacts between ground and the p-substrate and between power and the n-well to maximize the contact area. Finally, we added pins to connect power, ground, input, and output nets to the design. The completed layout is seen in figure 4.

To ensure that our layout met the design rules imposed by the technology library, we used a design rule checker. Our initial design had some small errors, such as the n-well being too close to the contacts to power, but the design rule checker enabled us to rapidly resolve these issues. We also ran another check to ensure that the connections in the layout matched the connections in the schematic. After adding the proper pins, this check also passed. With both checks completed, our inverter was finished.

III. Conclusion

The workflow implemented enables the rapid re-use of circuits. This concept is important as it is not feasible to layout every MOSFET and every connection in a larger design. By building up a design from previously designed components, keeping track of the characteristics of each piece along the way, it is much easier to create complex digital circuits.

This process is the essence of VLSI. It is not possible for people to build a CPU with billions of transistors one MOSFET at a time. Very large scale designs are only made possible by slowly building up a subsystem by using repeatable elements, and abstracting away the fine details.

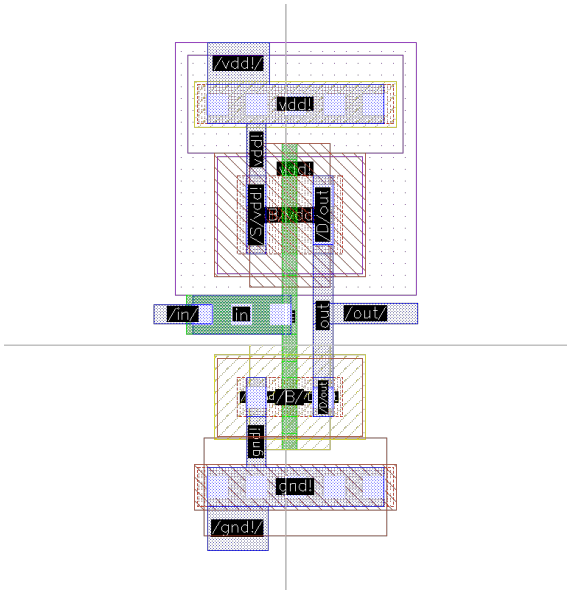

Fig. 4. Inverter layout

References

[1] David Money Harris Neil H. E. Weste. *CMOS VLSI Design a Circuits and Systems Perspective, Fourth Edition.* Pearson, 2011.