Final Project – Coin bank

110062322 FEN-YU HSIEH

Contents

- Circuit diagram of my design and explanation

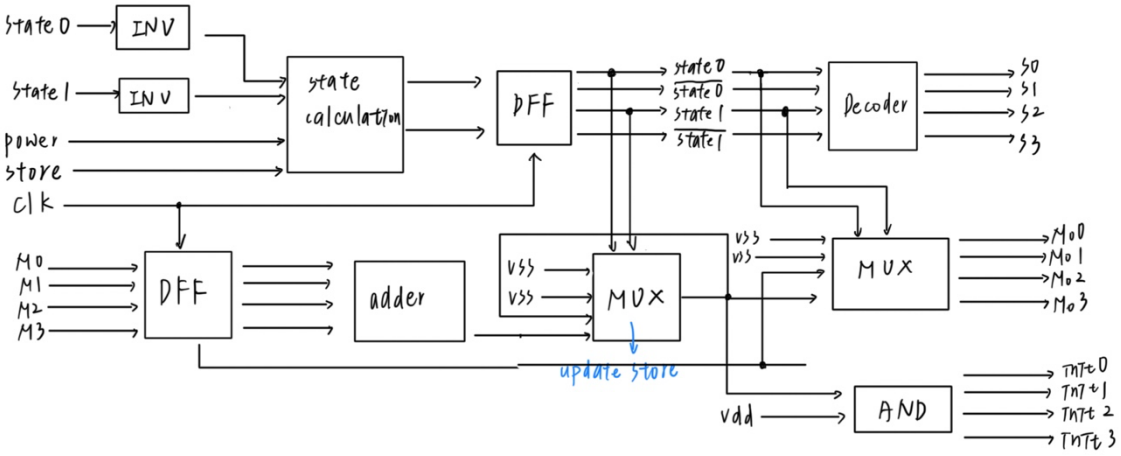

Figure 1

Figure 1 illustrates the general framework of my assignment. There are additional detailed operations, such as adding buffers in certain parts. However, the overall process follows these steps:

First, a state calculation is performed, utilizing basic logic circuits such as AND and OR gates to determine whether the next state0 and state1 will be 0 or 1. After computation, the results are stored in a D flip-flop (DFF). When the clock triggers, the outputs state0 and state1 are generated. These values are then passed into a decoder, which decodes them into S0, S1, S2, and S3.

For the inputs M0, M1, M2, and M3, they are first stored using a DFF before being processed by an adder. However, a multiplexer (MUX) is placed after the adder to control when the adder's output should be received. The stored value is updated only when both state0 and state1 are 1.

Finally, the outputs Mo0, Mo1, Mo2, and Mo3 are selected based on the current state to determine which value should be displayed. Meanwhile, init0, init1, init2, and init3 are displayed according to the stored values.

- Pre-sim waveform

Figure 2 and 3 show the waveform of my pre-simulation.

First, looking at the red line, it indicates the moment when power is enabled. At the next positive clock edge, the state transitions to 01.

Next, the blue line highlights the case when store is enabled, and the input value

mo = 1. At the next positive clock edge, the state transitions to 10. Additionally, mo0 is also set high because, in state 10, it displays the previously input value.

After another clock cycle, the state transitions to 11. At this point, the value from the adder is stored into store, causing init0 to go high. Since mo0 displays the stored value, it is also in a high state.

Lastly, the purple line marks the moment when m2 is set high. However, since there is no store signal, it does not affect the output values.

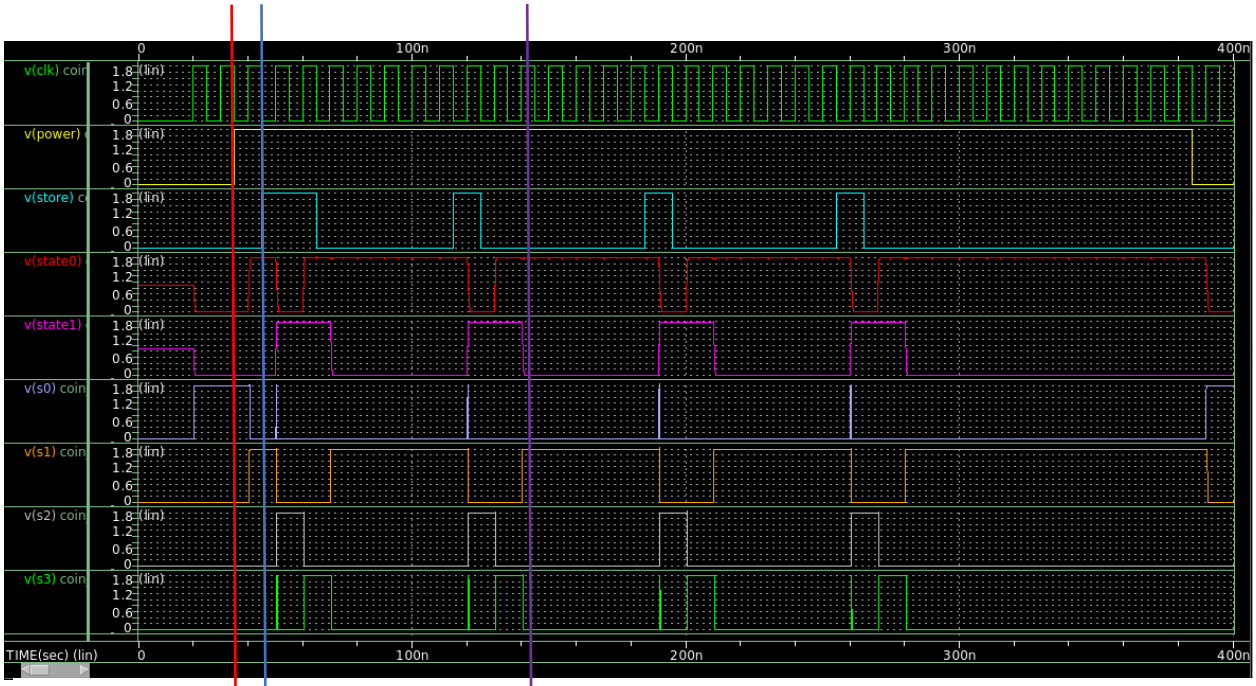

Figure 3

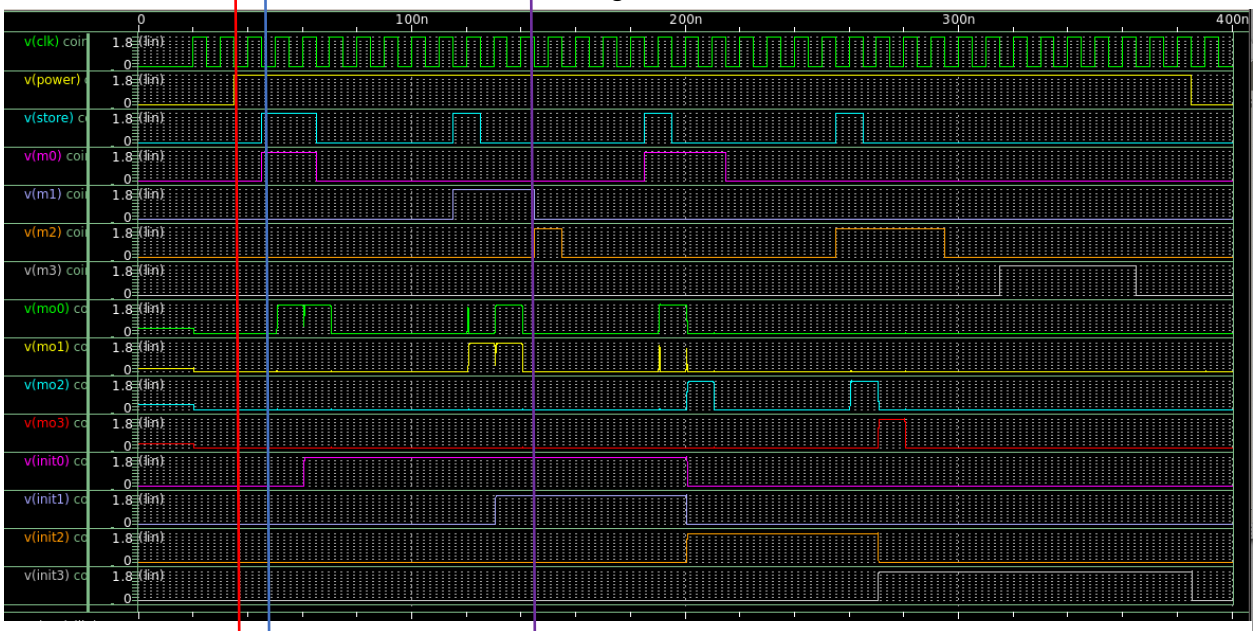

Figure 2

- DRC summary report

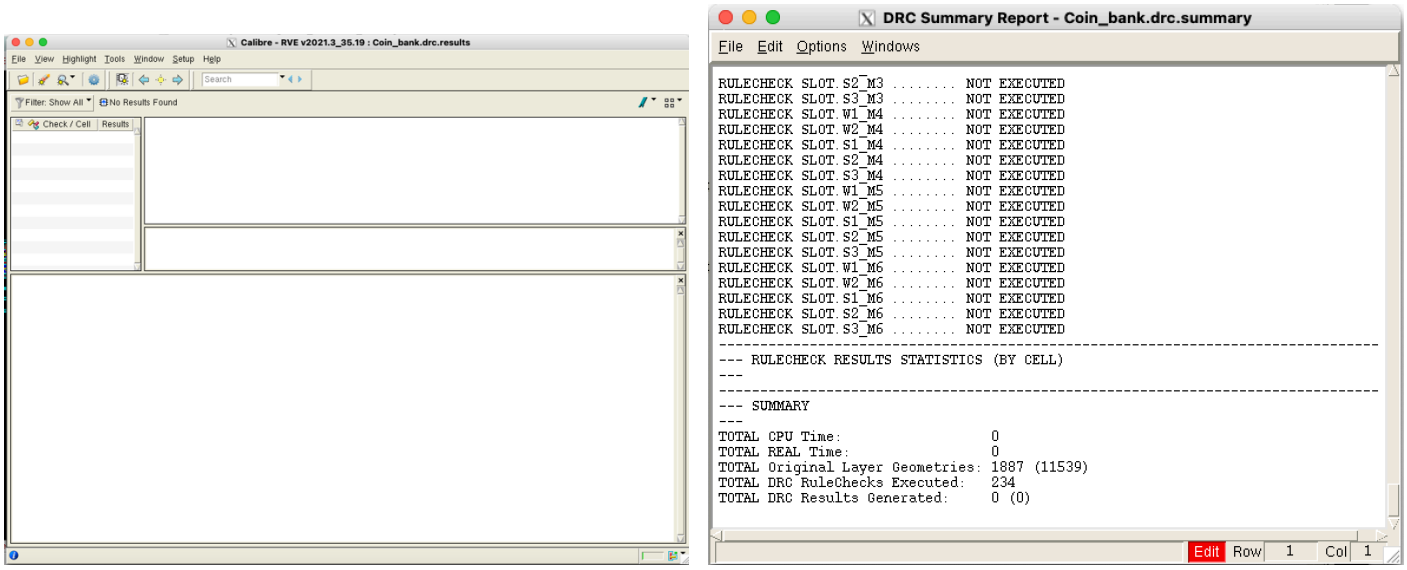

Figure 4

Figure 4 is my DRC Summary Report, which shows that there are no violations or non-compliant issues detected.

- LVS passing message

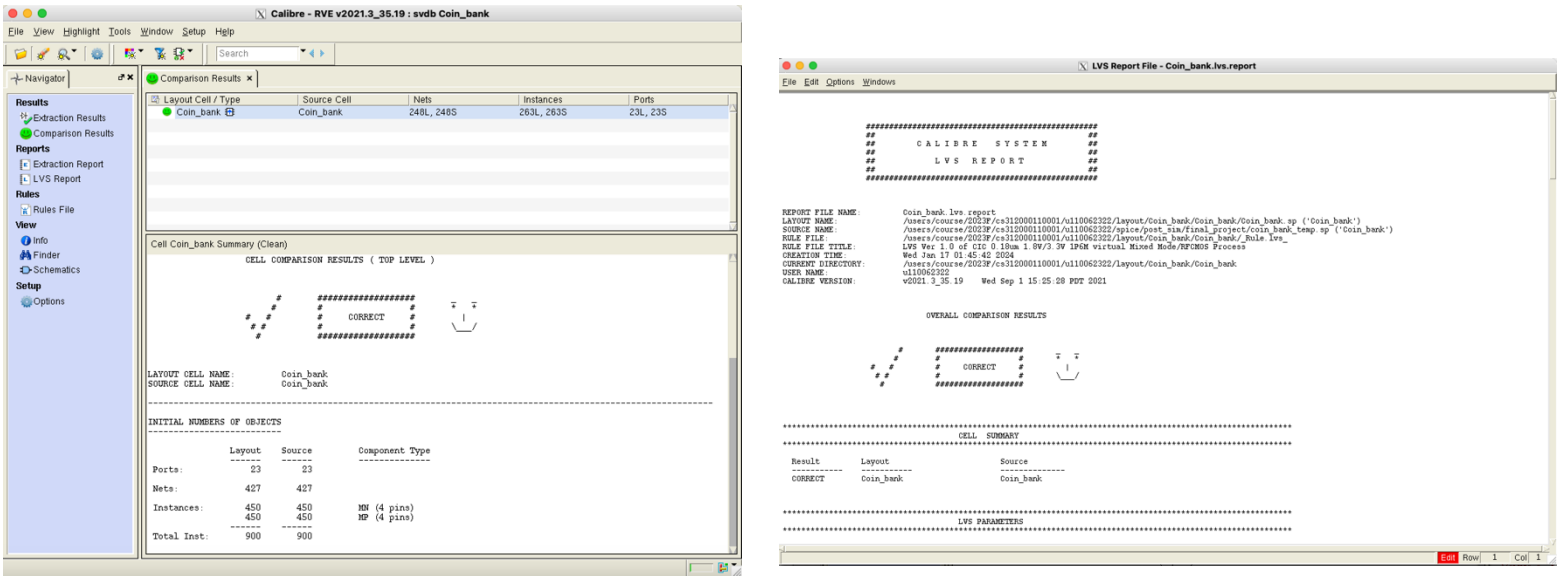

Figure 5

Figure 5 is my LVS passing message, which shows they are all CORRECT。

- Layout

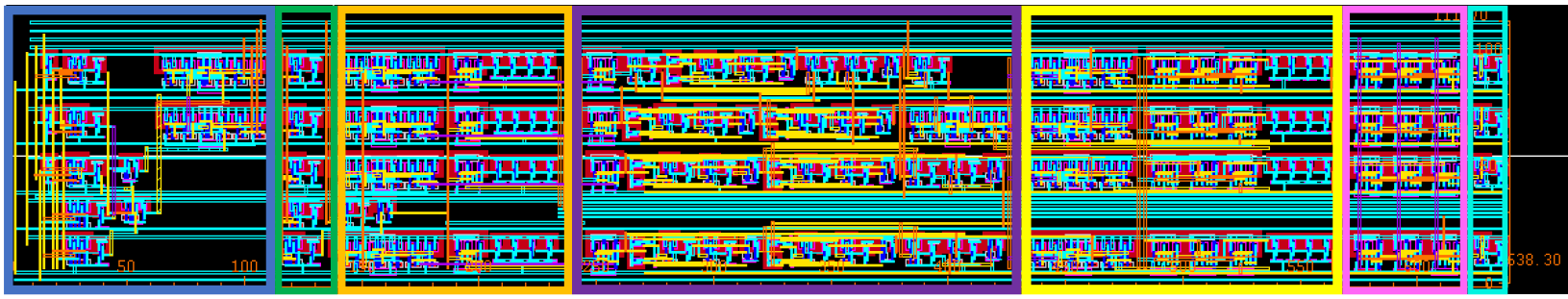

Figure 6

Figure 6 shows my layout design.

- The blue block is where state0 and state1 are computed.
- These states are then connected to the green block, which functions as the decoder for S0, S1, S2, and S3.
- The orange block stores the input values mo~m3.
- The purple block represents the adder.
- The yellow block handles the store process.
- The pink block corresponds to the output mo0~mo3.
- The light blue block represents init0~init3.

I first designed the fundamental components such as AND, OR, MUX, buffer, etc., and instantiated them as reusable instances. Finally, in the coin_bank file, I simply connected all these instances together.