Digital Lab 3:

Experiment1:

Control Circuits of Memory Data Scan and Output

Date: 2023/09/21

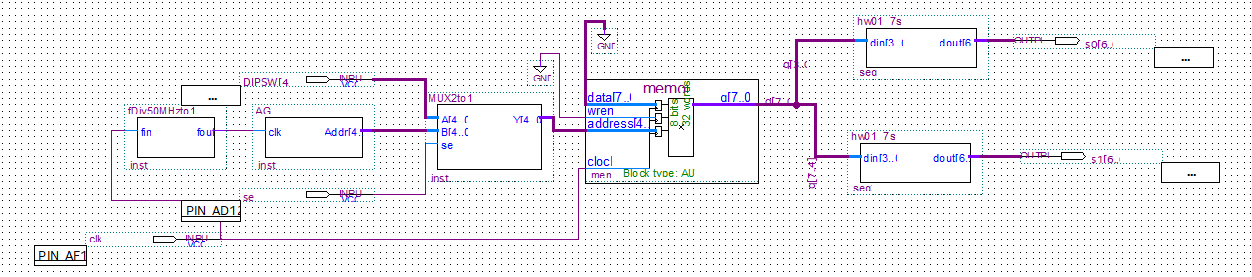
Class: 電機三全英班

Group: Group 11

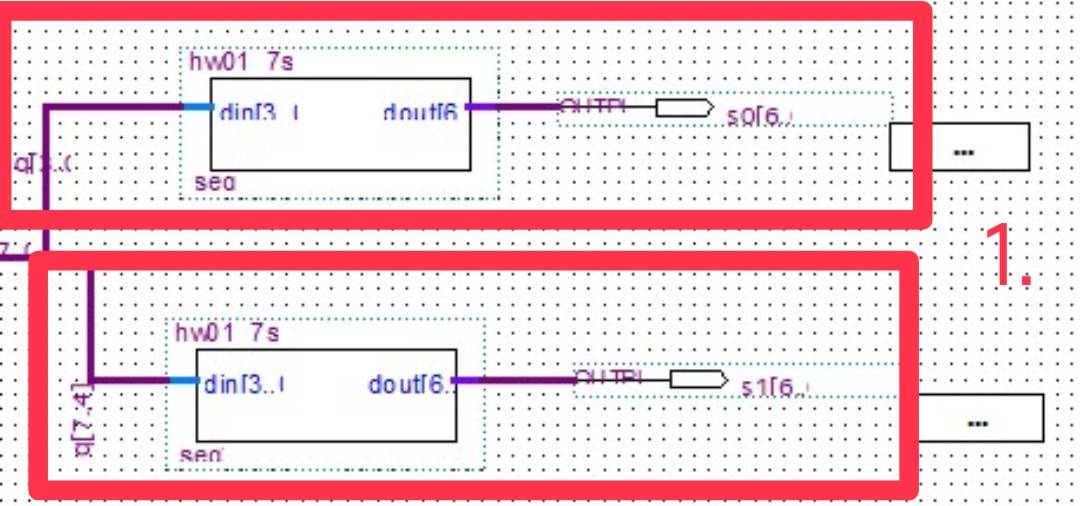
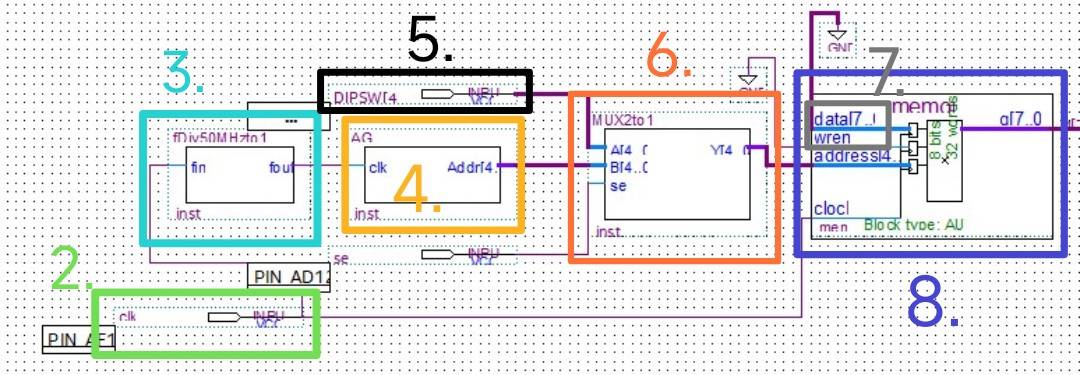
Name: B103105006 胡庭翊

1. Block Diagram

Structure:

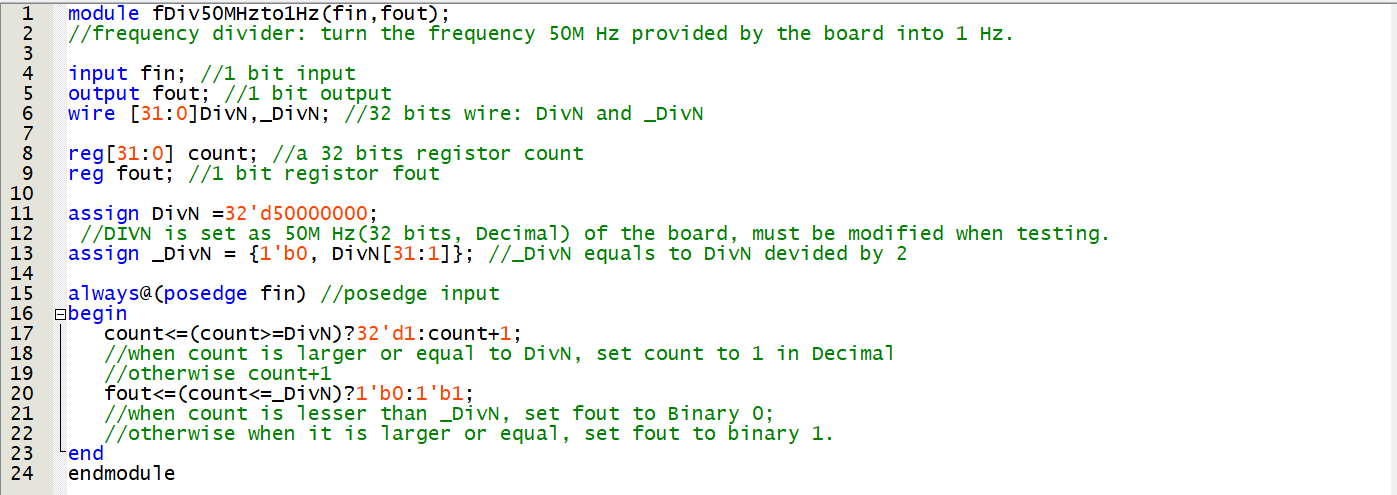


Function:



Descriptions:

1. 7 Segment Decoder: In this project, we use hexadecimal number to display and hence we need 8 bits for output. Since 4 bits is the largest number that a 7 segment display can modify, we here use two 7 segment display.
2. We use 50MHz that set by the board.
3. Frequency Divider: Convert 50MHz into 1Hz.
4. Address Generator: Using the fout of frequency divider, the clk follows a frequency of 1Hz to automatically generate the address of memory.
5. Use DIPSW to be the input of address.
6. 2-to-1 Multiplexer: Select the source of memory address: when sel is 0, the address of memory is decided by Dip Switch; when sel is 1, the address of memory is decided by address generator.
7. Since our experiment is control circuit of memory data scan and output, our wren and data should just be grounded.
8. Memory: This is the model generated by megafunction, the MUX should be its input.
9. Frequency Divider (FDiv)
10. Verilog Code and Comment

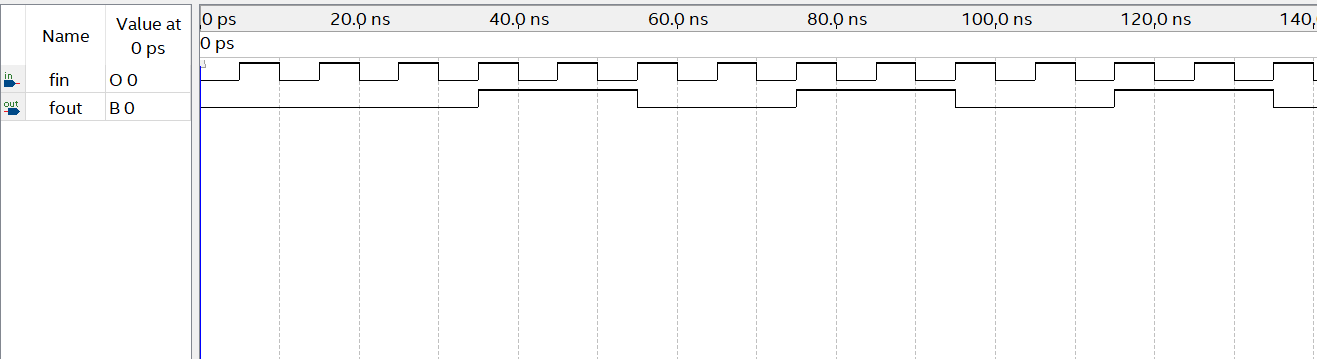


1. Simulation

First we need to adjust the value of DivN in verilog code into a better-simulated value.



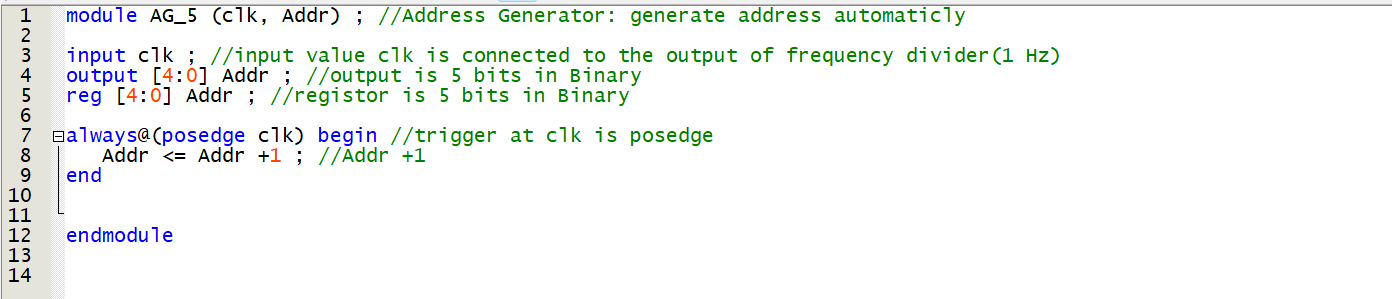
This value would make the output (fout) be 1/4 of the input (fin).



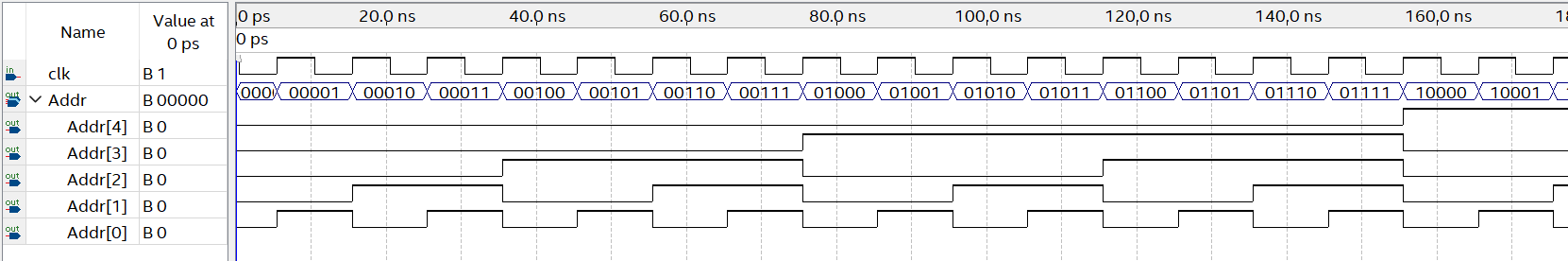
We can see that after count became 3 in the first period, since count = 3 is larger than \_DivN, fout changed from 0 to 1, and hence the frequency divider started to work.

The later-on waveform of fout is 1/4 of fin’s, which showed that the frequency divider is well-functioned.

1. Address Generator (ADDG)
2. Verilog Code and Comment



1. Simulation

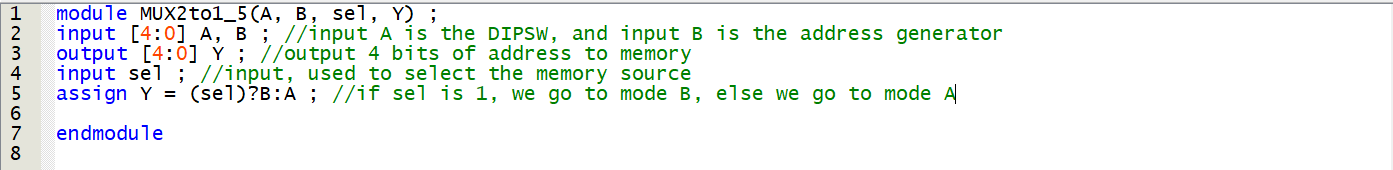
The address generator triggered when clk is in posedge, it makes its output Addr+1.

The Addr[4], Addr[3], Addr[2], Addr[1], Addr[0] represent the 5 bits of the binary address.

The output satisfied the cycle of binary numbers, so the Address

Generator is functioned normally.

1. 2-to-1 Multiplexer (2to1 MUX)
2. Verilog Code and Comment

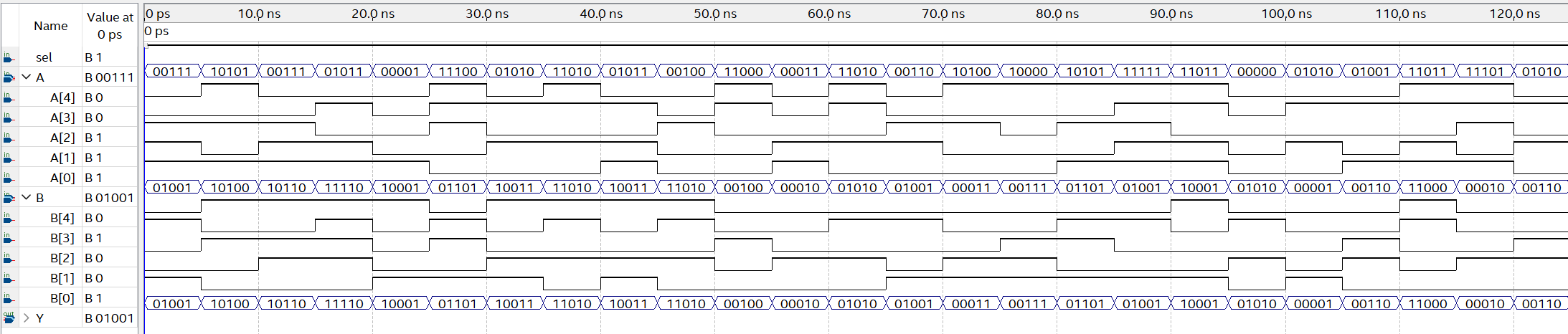


1. Simulation



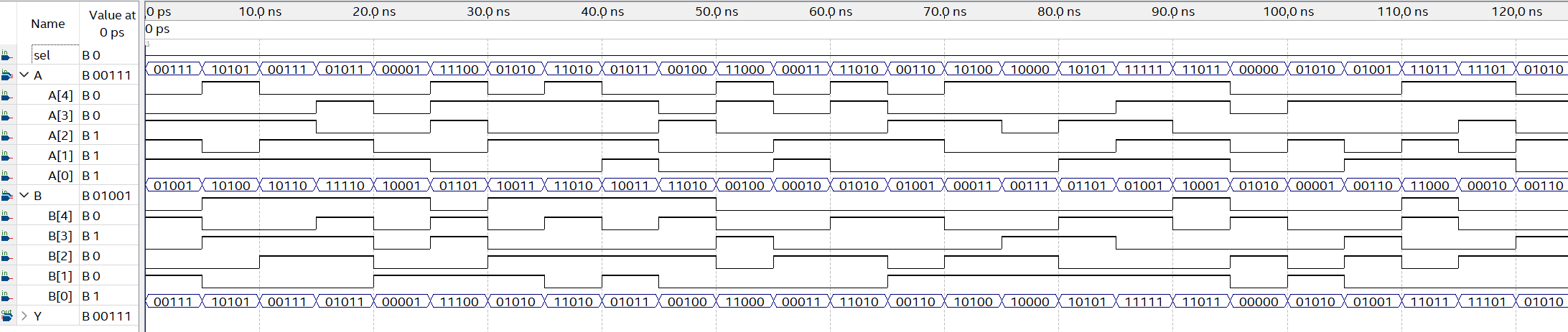
Set the address of A and B randomly.

When sel = 1:



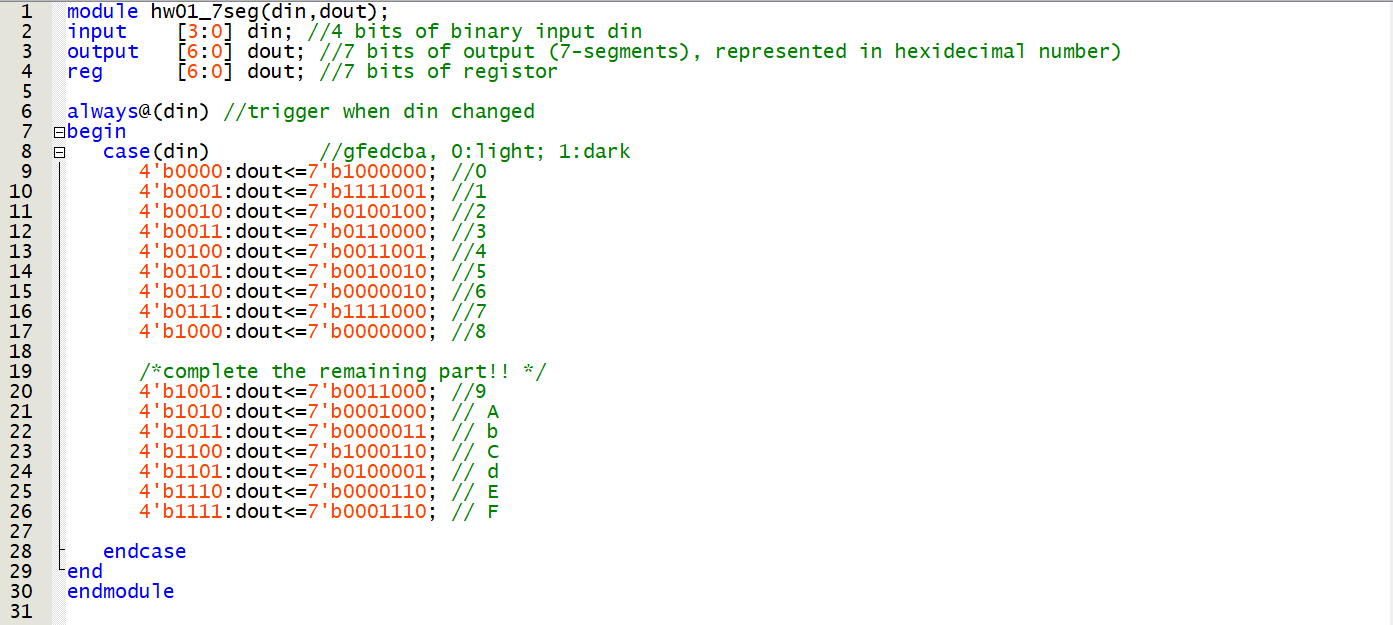
The output Y follows directly as B.

When sel = 0:

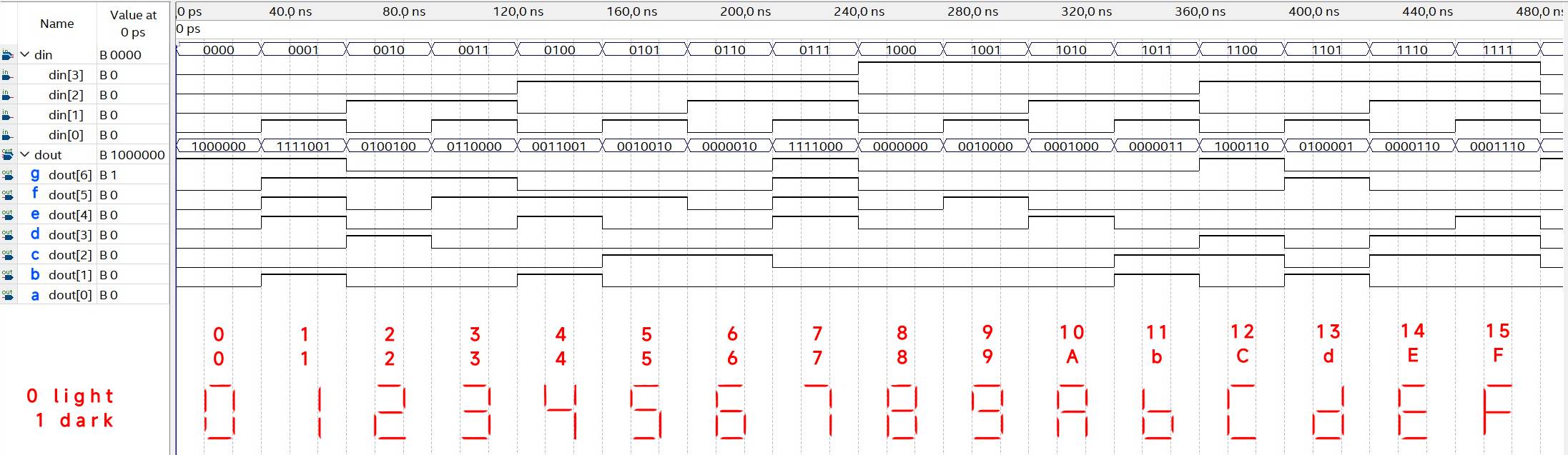


The output Y follows directly as A.

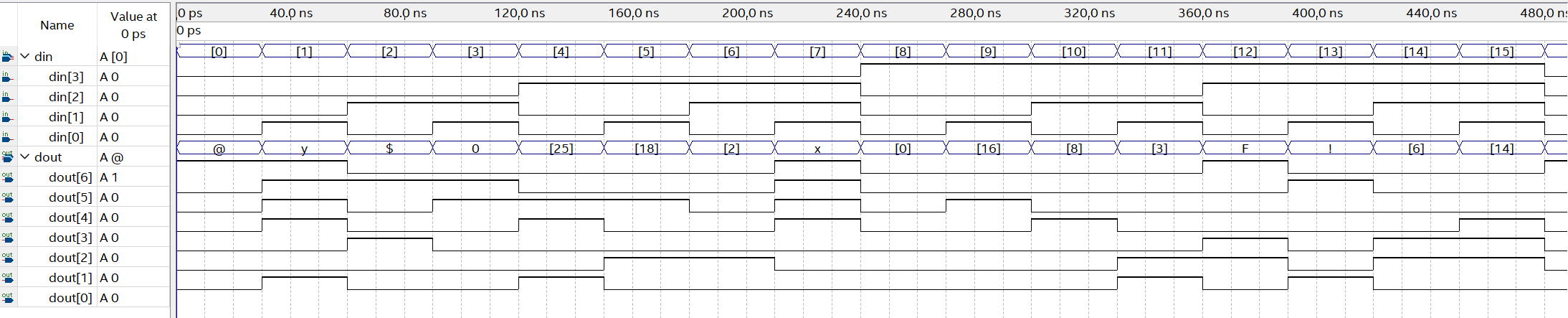
1. 7 Segment Decoder
2. Verilog Code and Comment



1. Simulation



Convert the binary value into ASK-II code to check if the decoder is correct.



1. Reflection

In this electrical engineering experiment, I encountered Quartus, Verilog, and FPGA boards for the first time. The experiment aimed to use the Quartus software to input Verilog code from the manual and create a 50MHz to 1Hz frequency divider, 2-to-1 multiplexer, address generator, and 7-segment decoder. Additionally, we simulated waveforms and integrated pre-written memory, creating a block diagram and configuring the pin planner. The ultimate goal was to design a memory data scanning output control circuit and successfully program it into the FPGA board.

Due to my limited familiarity with the software, we couldn't complete the experiment during our class time. As a result, I devoted my own time to understand Quartus and a little verilog and also attended extra lab sessions. Quartus revealed its powerful features, although it appeared somewhat complex at first. Despite the intricate steps and the vast amount of knowledge to acquire, I believe this will be an exciting experience (if there are no conflicting exams that week).

Verilog and FPGA were also a new territory for me. While I had a brief introduction to Verilog during my freshman year, this was my first hands-on experience. This semester, I am also taking a course on practical digital systems. As a novice in hardware description languages, I realize that self-studying Verilog is crucial for my understanding and behavior in this semester.