**LAB REPORT**

**Laboratory exercise 5: Finite State Machines**

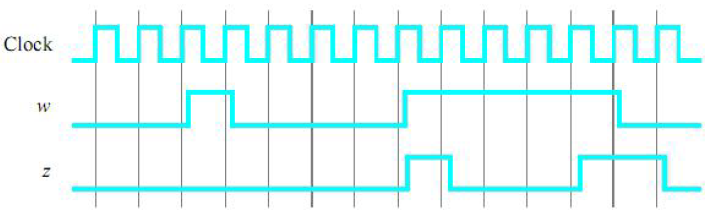
3220104688 杨佳昕

12 Dec, 2024

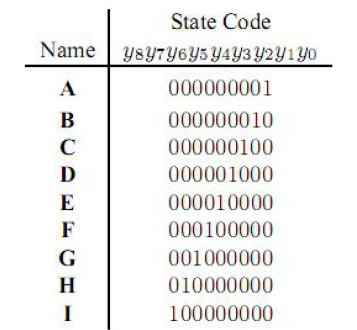
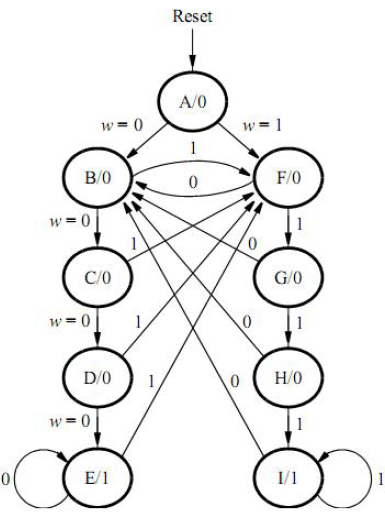
**1. Problem Description**

**Part I**

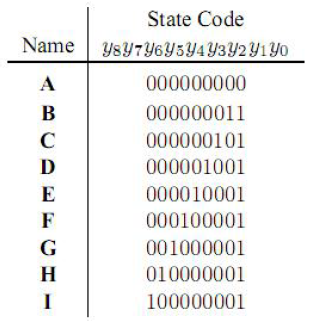
Design a finite state machine. Whenever w=1 or w=0 for four clock rising edges, the value of z is 1, otherwise, z is 0. After four rising edges, if w remains constant, the value of z has to be 1 until w changed.



A state diagram for this FSM is shown below. Match y(8 downto 0) to ledr(8 downto 0) and z to ledg.



Use sw0 as active-low reset, sw1 as w and key0 as clock input, and design the project. And what’s more, if changing the state code table to the table below, change the vhdl code to cover the functions above.



**Part II**

A suggested skeleton of the FSM vhdl code has given below. Fill the code to achieve the functions same as task 1 in Part I.

*library ieee;*

*use ieee.std\_logic\_1164.all;*

*entity part2 is*

*port(…define input and output ports…);*

*end part2;*

*architecture behavior of part2 is*

*…declare signals…*

*type state\_type is (A, B, C, D, E, F, G, H, I);*

*signal y\_Q, y\_D : state\_type; -- y\_Q is present state, y\_D is next state*

*begin*

*…*

*process(w, y\_Q)*

*begin*

*case y\_Q is*

*when A if (w = ‘0’) then y\_D <= B; else y\_D <= F; end if;*

*… other states…*

*end case;*

*end process;*

*process(clock)*

*begin*

*…*

*end process;*

*end behavior;*

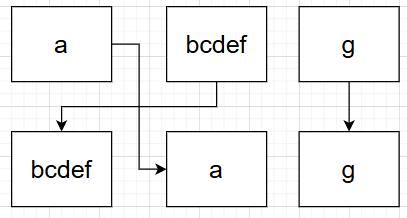
**2. Design Formulation**

**Part I**

The key point to both tasks are the logical relationship of A to I.

In task 1, every time the state is updated forward, y loops one bit forward. We can use ssl expressions.

In task 2, the last bit was kept still, and the other bits of y loop. Keep the last bit unchanged, throw the first bit to the second-to-last bit, and move other bits forward. The process is shown in the graph below.



Once the relationship is achieved, match each state to the next state with w value with case sentences.

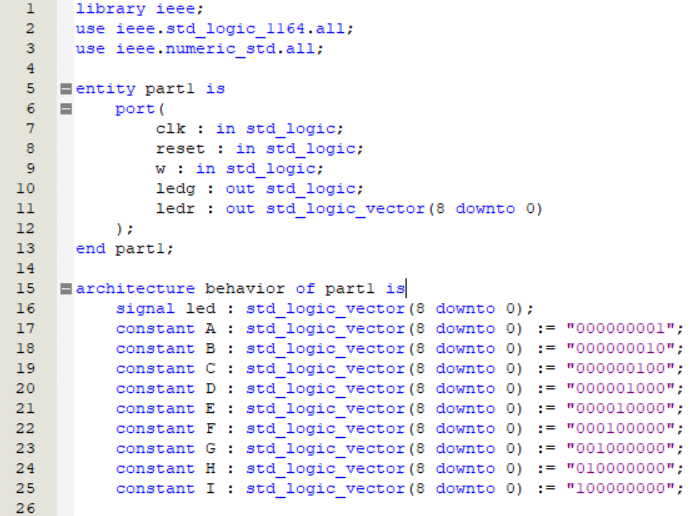
**Part II**

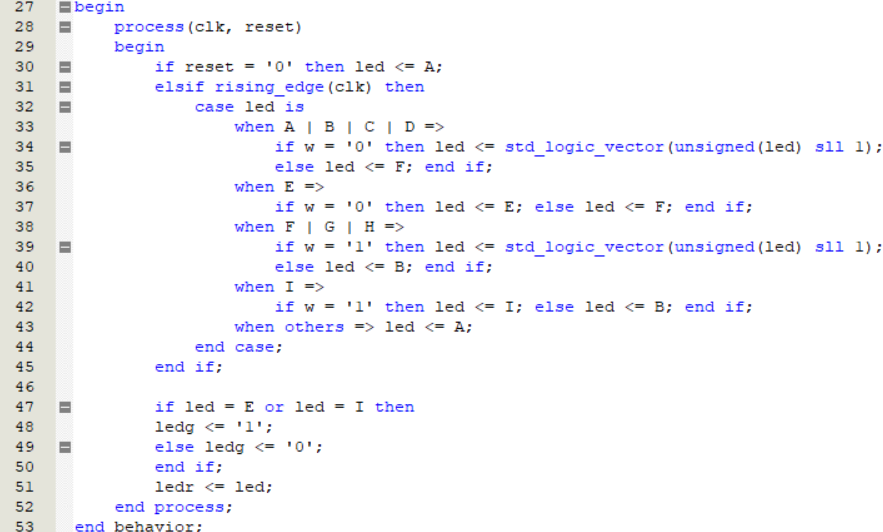
The first *process* is the same as the match part above. The second *process* is to refresh y\_Q value with y\_D value when *clock* or *reset* triggers. After the two parts, match the state with led values to determine whether led lights or not.

***NB:*** Reset is set as active low, so when reset = 0, the whole system changes to the original state.

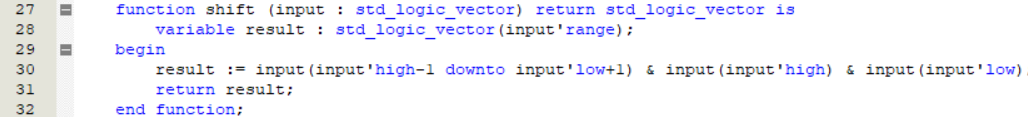
**3. Design Entry**

**Part I**



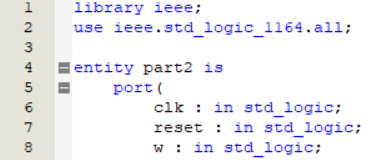


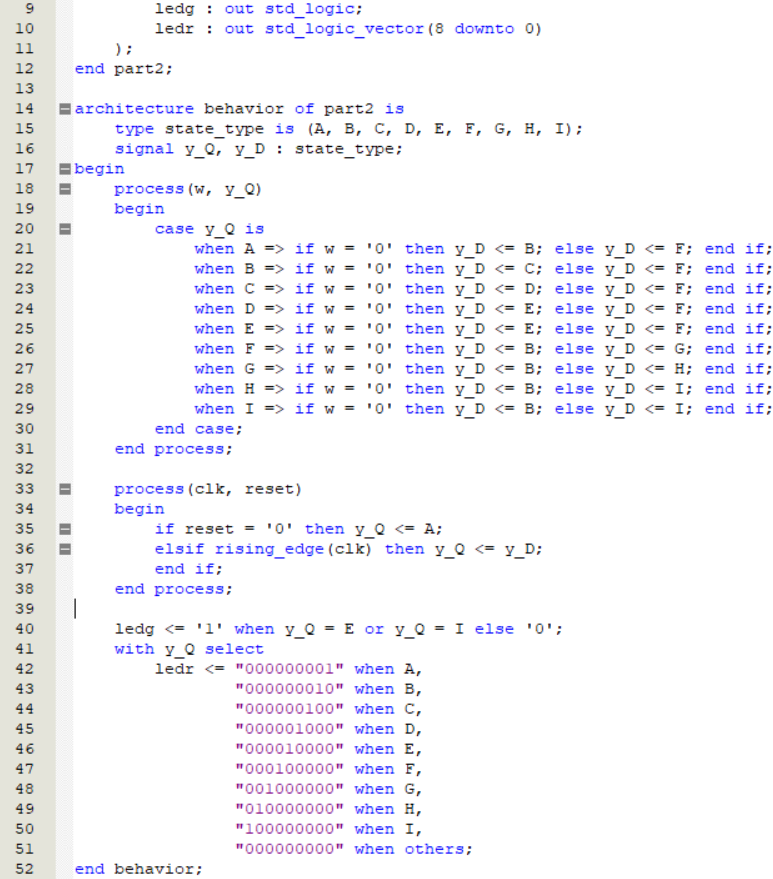
When the state code table changes, we just add a function *shift* to keep the last bit still and loop the other bits:



Other codes are similar to those above.

**Part II**

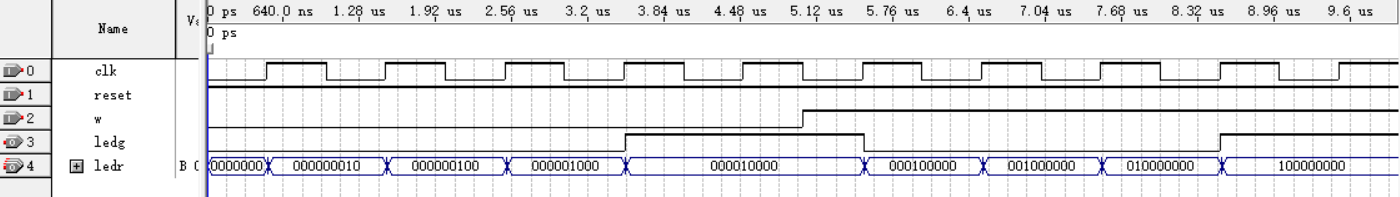




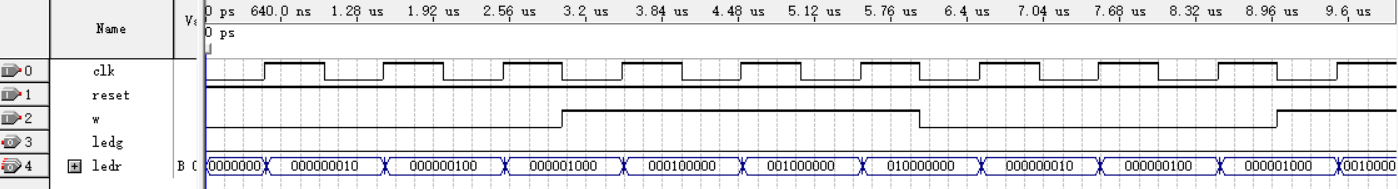
**4. Simulation and Synthesis Results**

**Part I**

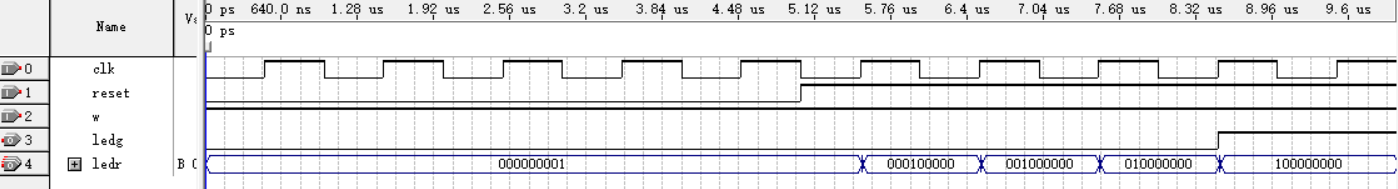
Ledg triggers:



Unable to trigger:

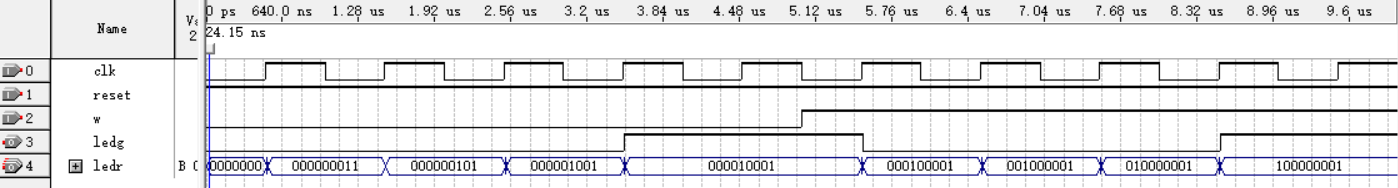


Reset:

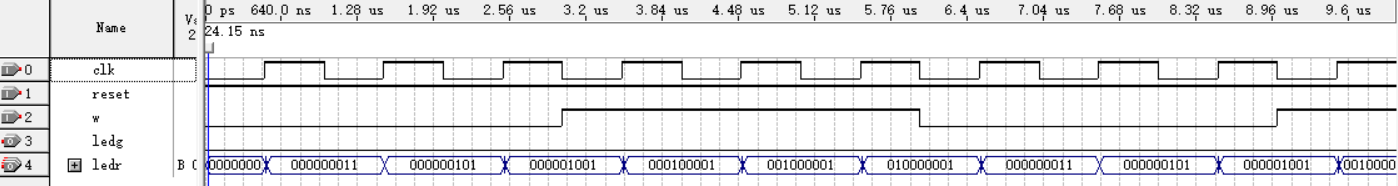


When state code table changes:

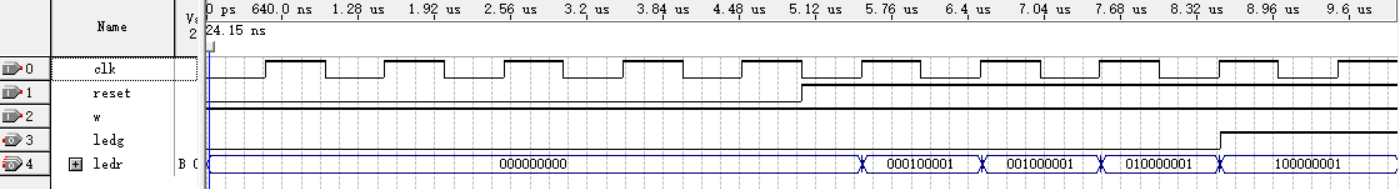
Ledg triggers:



Unable to trigger:

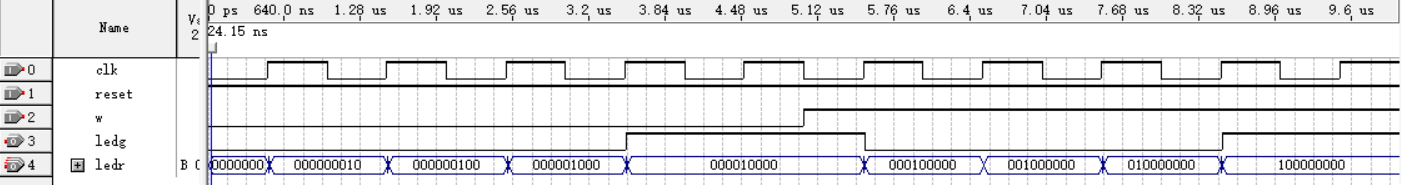


Reset:

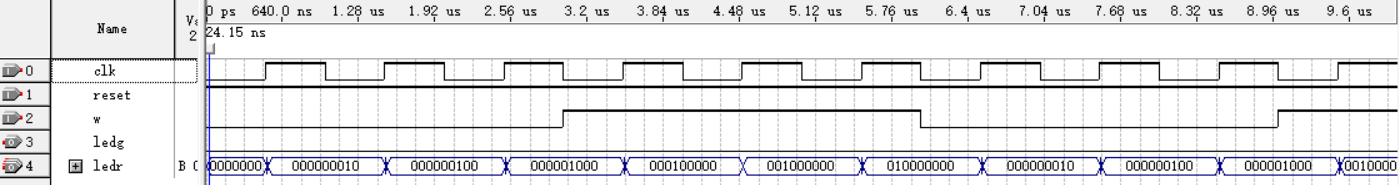


**Part II**

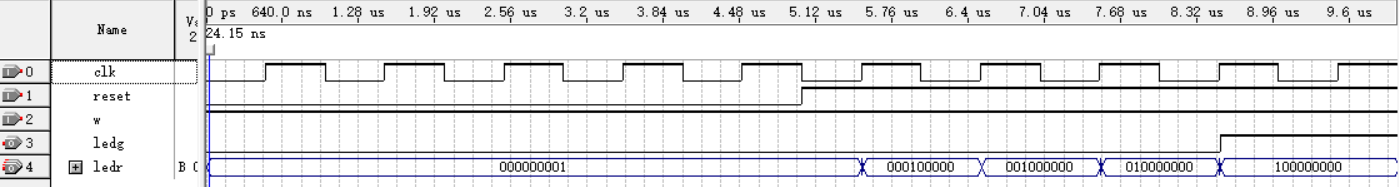
Ledg triggers:



Unable to trigger:

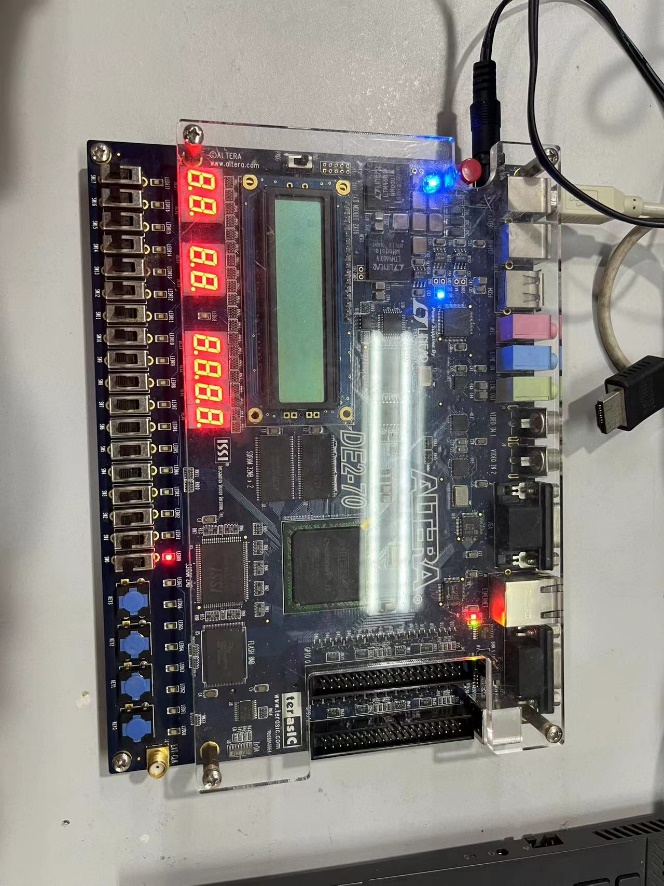


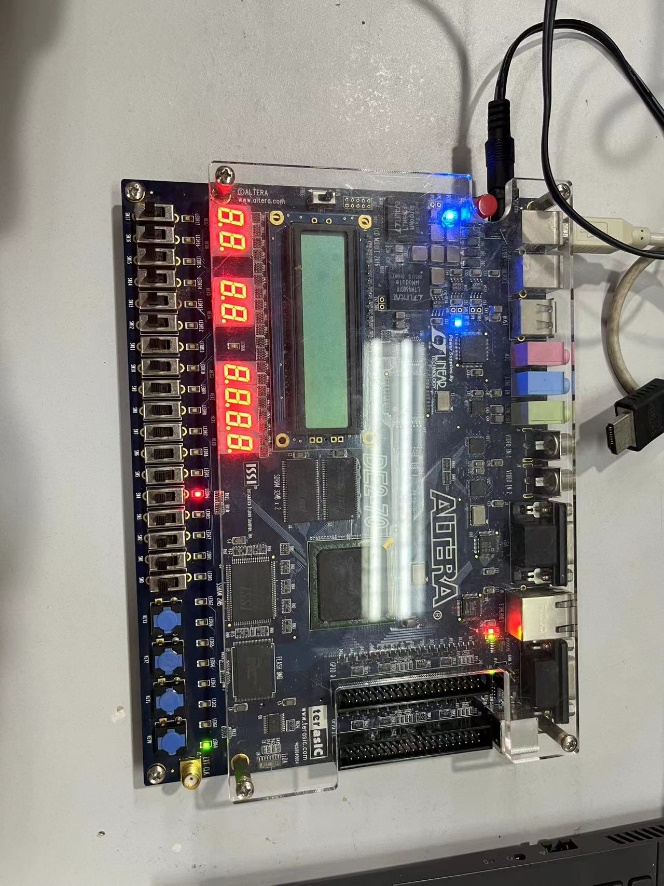
Reset:

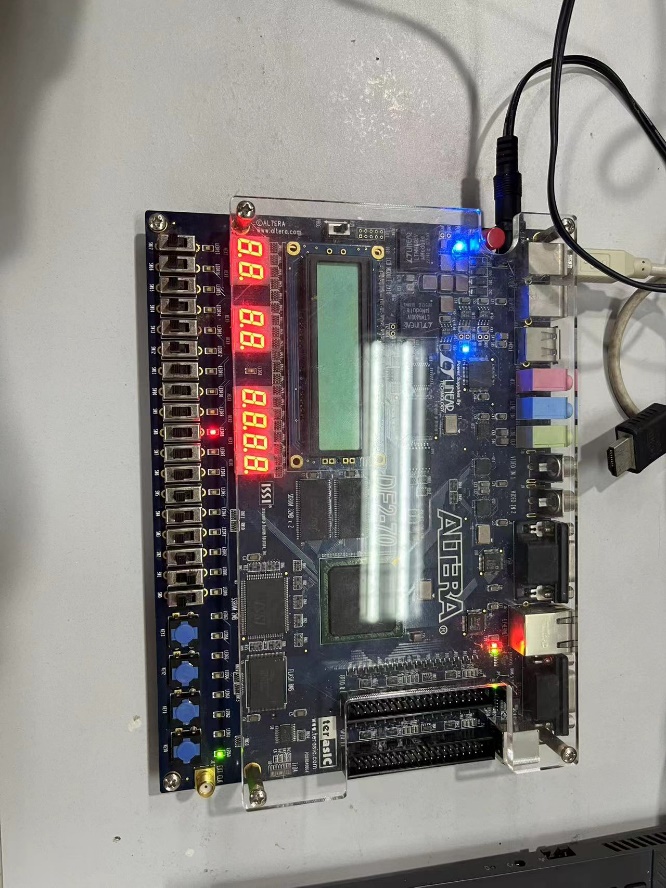


**5. Experimental Results**

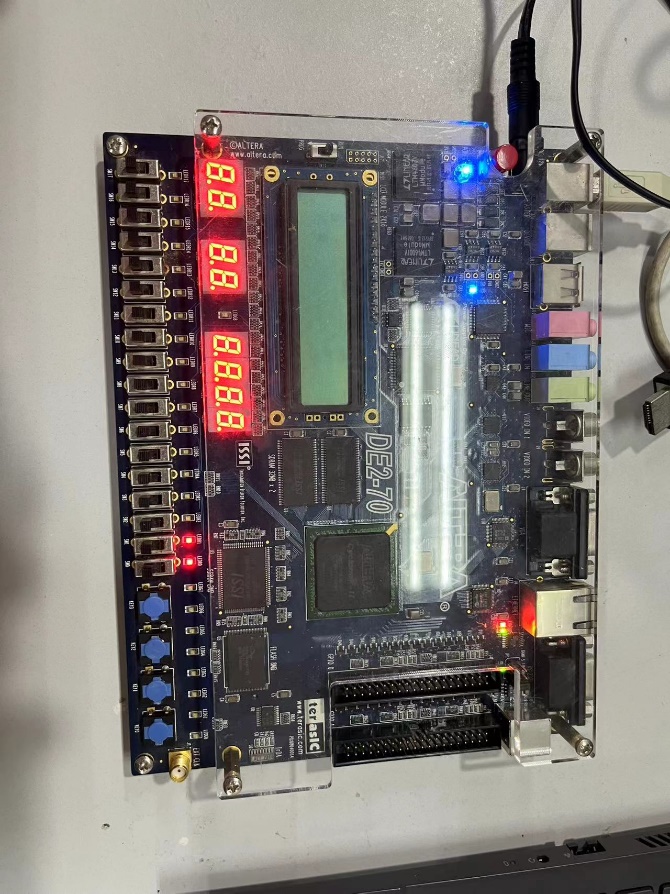
**Part I**

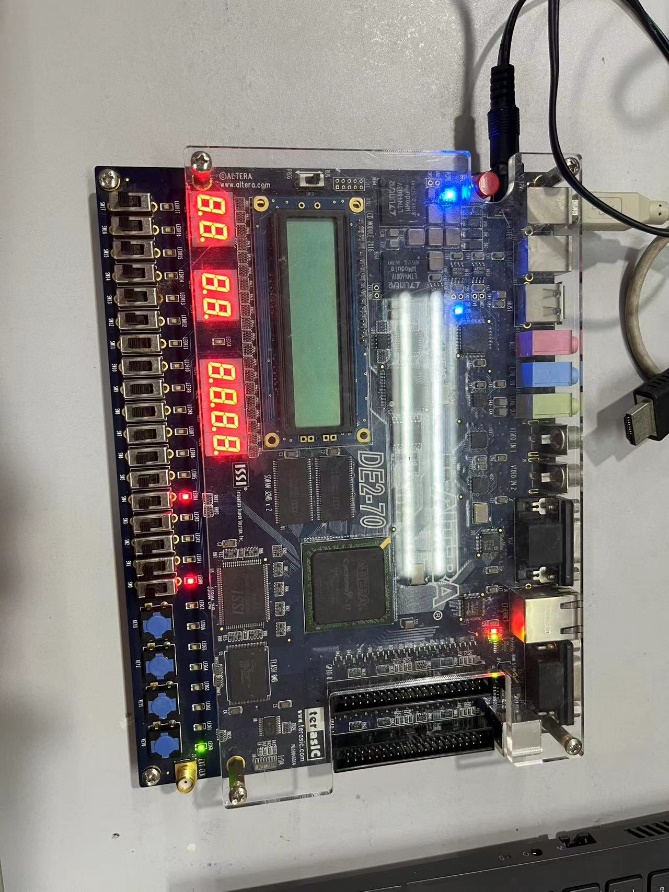


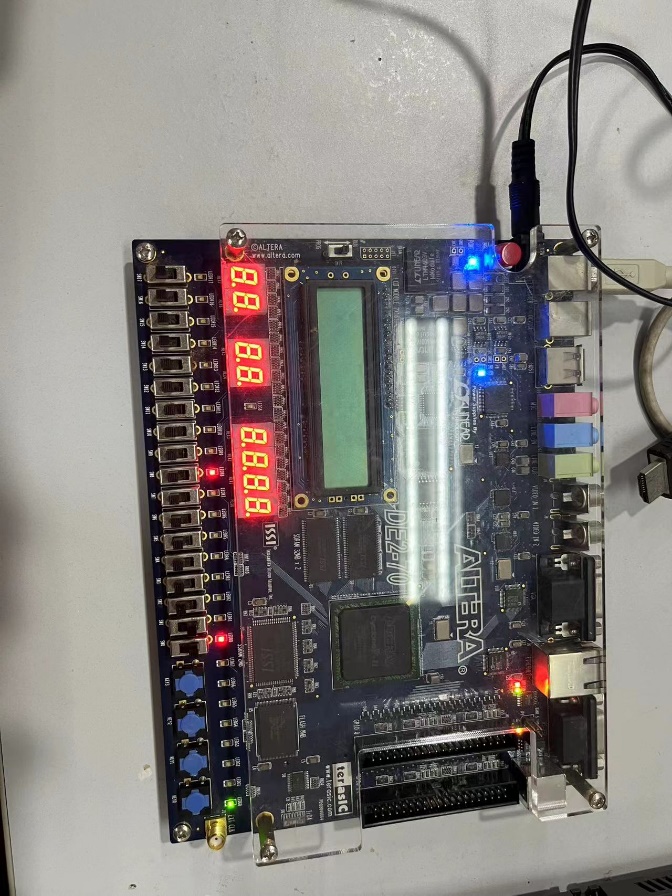




**Part II**

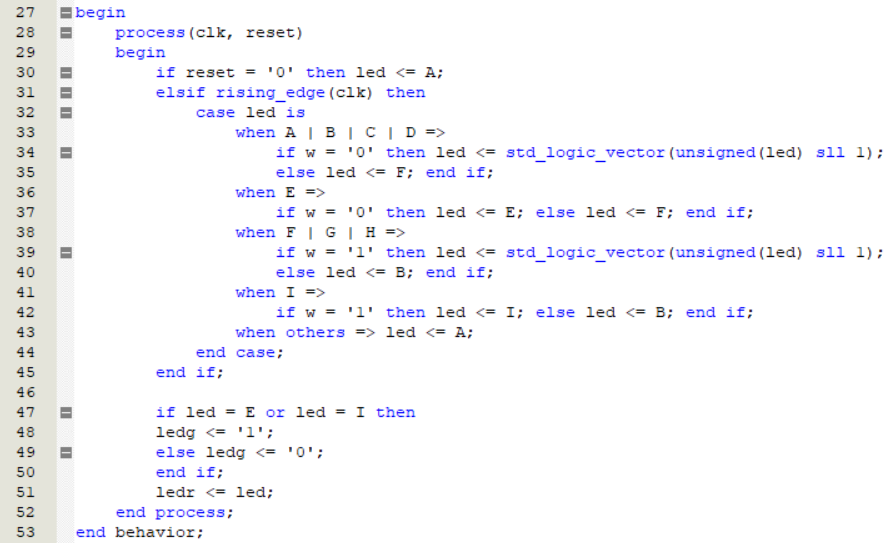




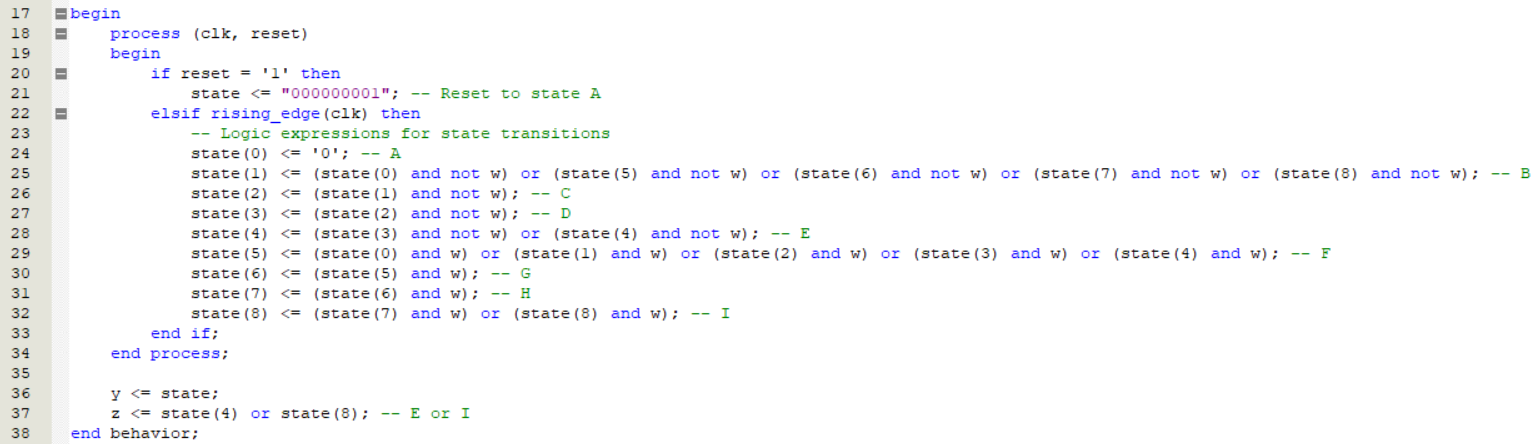


**6. Discussion and Conclusion**

Part I asks to write a logical expression that drives the flip-flop input ports. However, in the code given above:



This is not logical expression such as Boolean expressions. The correct code of this part is:



The uniqueness of this code is the processing logical of variables in process expressions. The value of any variable will not change until the next process triggered. Thus, although we declared *state(0)* <= *‘0’* and *state(1)* <= *state(0) and not w*, before the second process triggered, the value *state(0) = 1* is successfully transferred to *state(1)* expression. And when the second process arrives, *state(0)* is refreshed to 0.

However, I had to notice that both codes above achieved the same function, e.g., state transfer. It seems that the first expression (regarded wrong because of no logic) is more easy to read and more ‘logical’ in human eyes.