Ryerson University

Faculty of Science

Department of Physics

Course Number	*COE 328*
Course Title	*Digital Circuits*
Semester/Year	*Fall 2021*
Instructor	*Dr. Reza Sedaghat*
TA Name	*Sajjad Rostami Sani*

Lab/Tutorial Report No.	*5*

Report Title	*Lab 5*

Section No.	*03*
Submission Date	*20-11-2021*
Due Date	*29-11-2021*

Student Name	Student ID	Signature*
Ahmad El-Gohary	501011852	Ahmad El-Gohary

Objective

The objective of this lab was to simulate the operation of a sequential circuit and to design a finite state machine that cycles through the digits of my student number using the assigned state diagram.

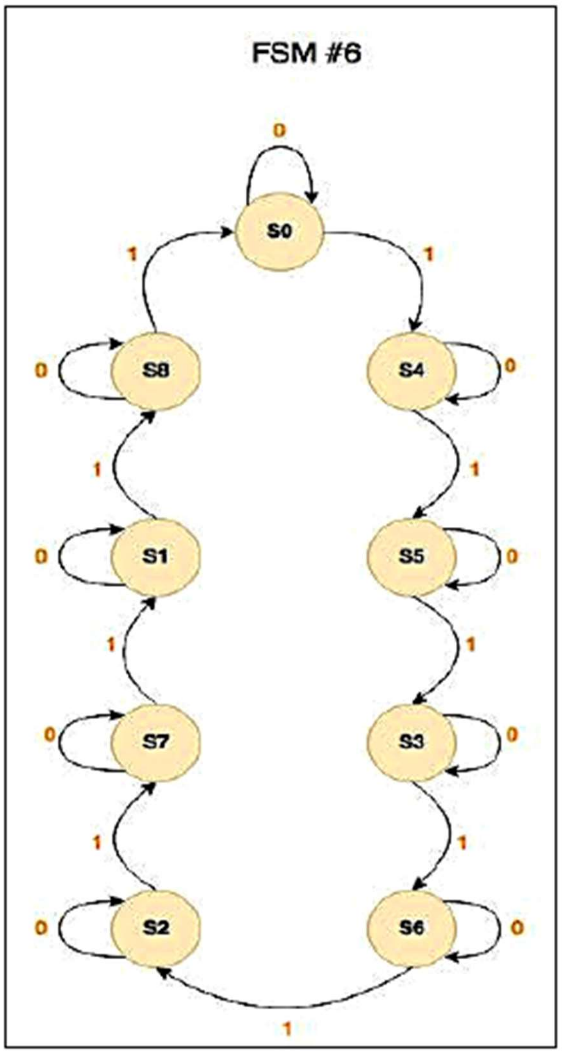
[This part of page is left blank intentionally]

Procedure

- To get the assigned state machine take the last 4 digits of my student number and divide it by the number of states to get the remainder.
- The remainder is the state number

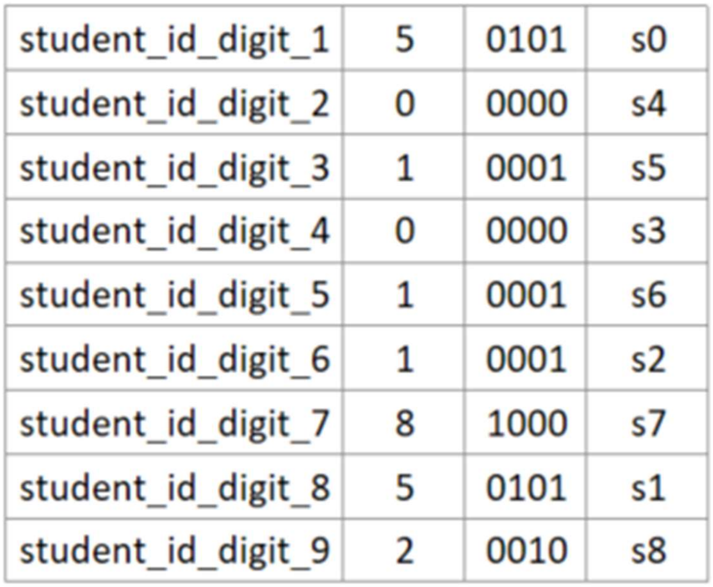
  1852/13 has a remainder of 6

  Assigned Finite State Machine:



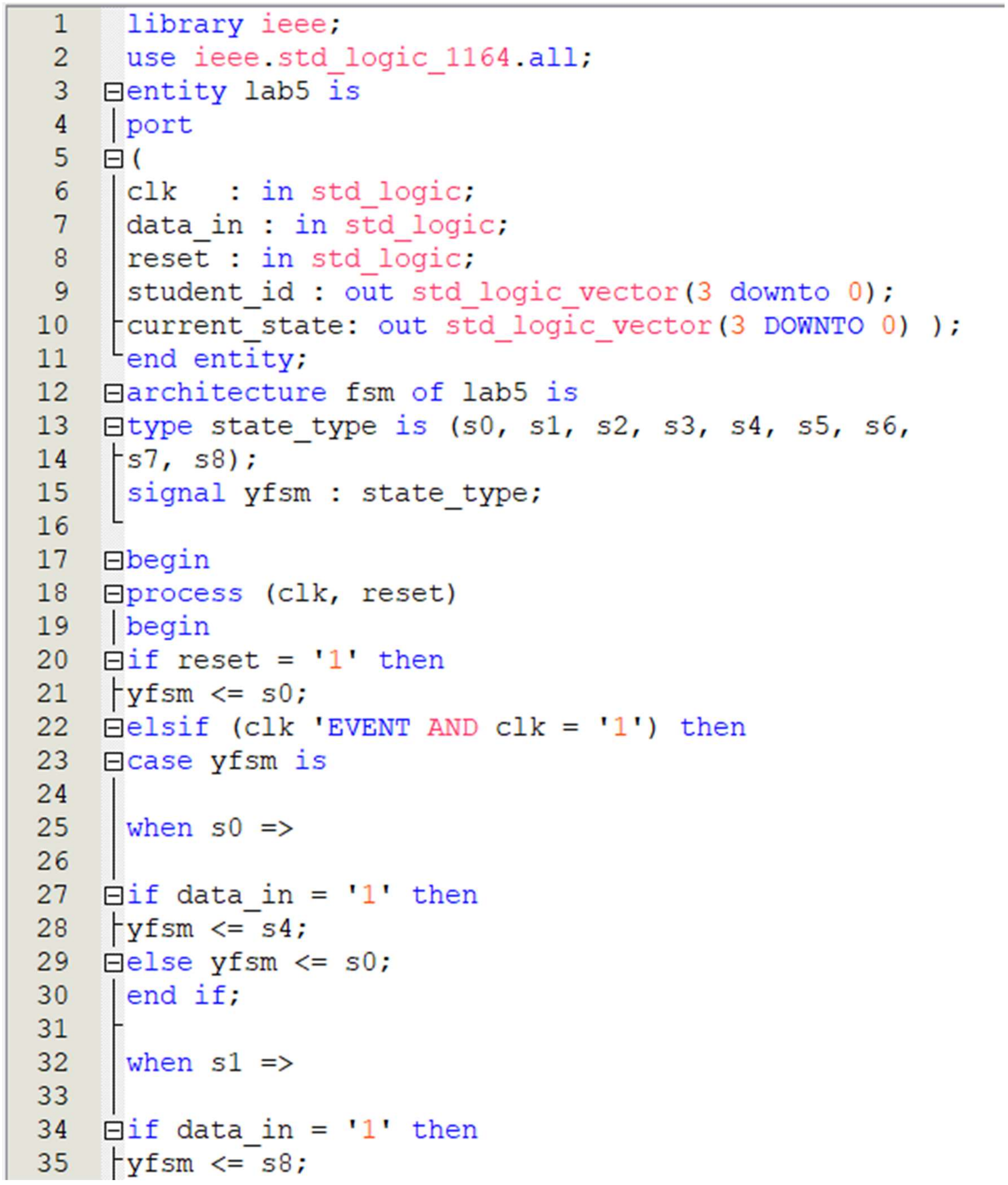
The machine shows that when data in is equal to 0 the state is assigned to itself but when data in is equal to 1 it is assigned to the state to its right for example: when data in for s0 is 1 s0 is assigned to s4.

Value of s0 to s8 is assigned the value its respective student number digits according to the FSM.

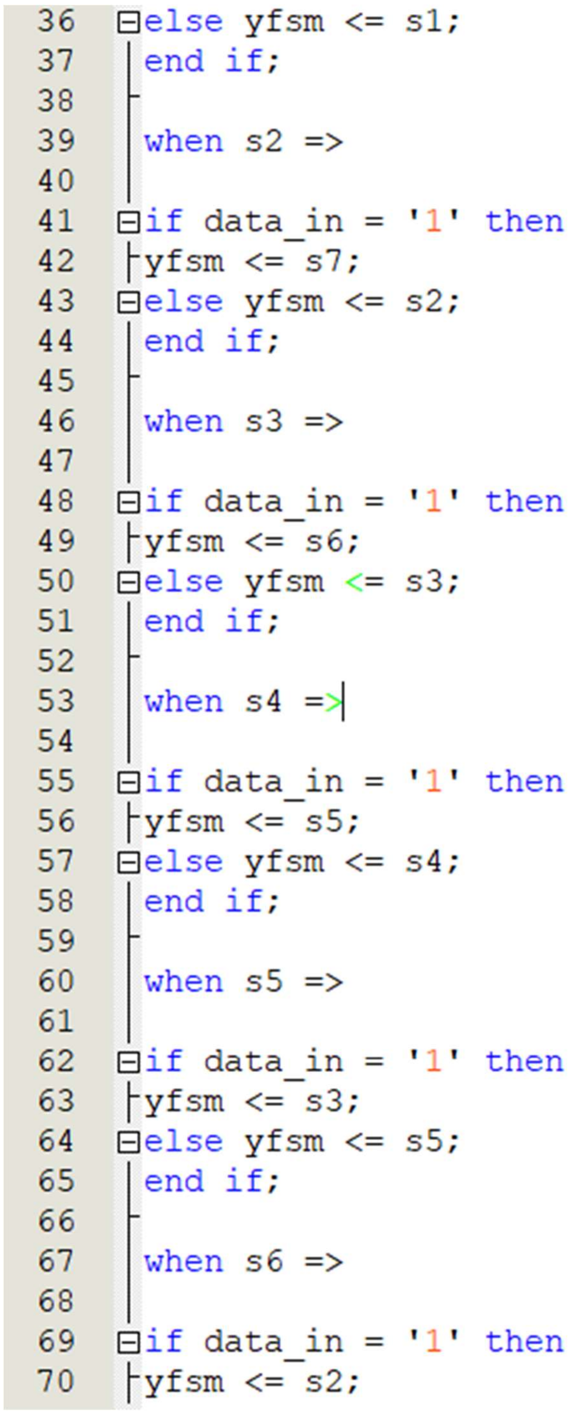


student_id_digit_1	5	0101	s0
student_id_digit_2	0	0000	s4
student_id_digit_3	1	0001	s5
student_id_digit_4	0	0000	s3
student_id_digit_5	1	0001	s6
student_id_digit_6	1	0001	s2
student_id_digit_7	8	1000	s7
student_id_digit_8	5	0101	s1
student_id_digit_9	2	0010	s8

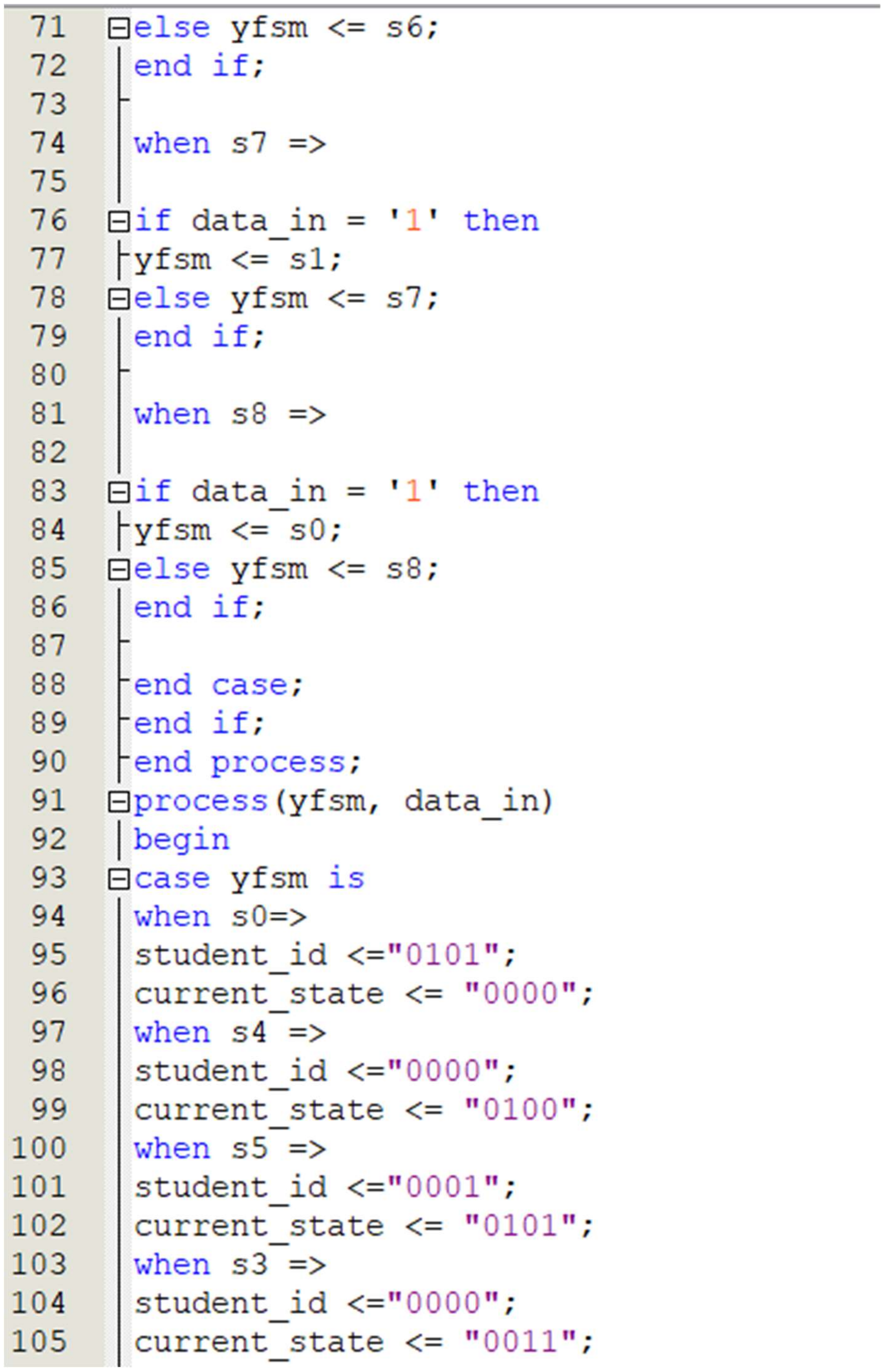
Implementing the previous information into VHDL code provided in the lab manual page 6:



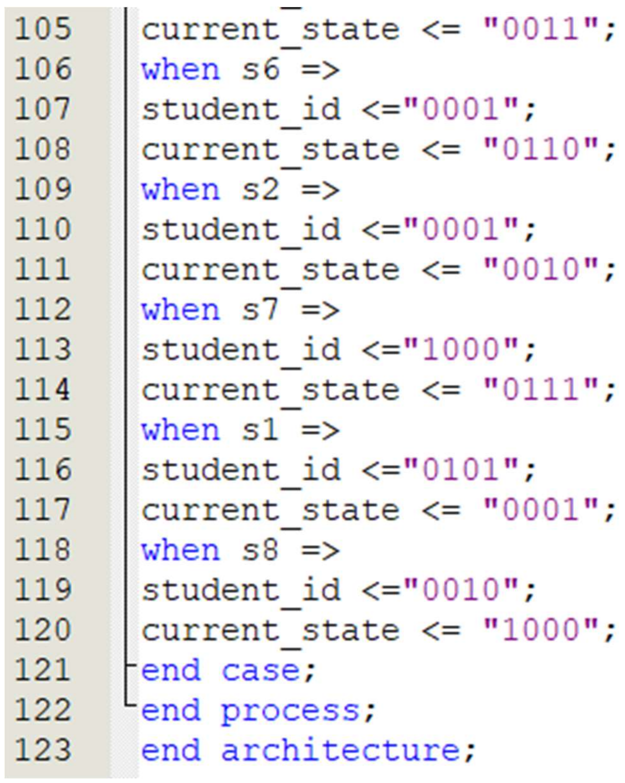
```
library ieee;
use ieee.std_logic_1164.all;
entity lab5 is
port
(
clk    : in std_logic;
data_in : in std_logic;
reset : in std_logic;
student_id : out std_logic_vector(3 downto 0);
current_state: out std_logic_vector(3 DOWNTO 0) );
end entity;
architecture fsm of lab5 is
type state_type is (s0, s1, s2, s3, s4, s5, s6,
s7, s8);
signal yfsm : state_type;

begin
process (clk, reset)
begin
if reset = '1' then
yfsm <= s0;
elsif (clk 'EVENT AND clk = '1') then
case yfsm is

when s0 =>

if data_in = '1' then
yfsm <= s4;
else yfsm <= s0;
end if;

when s1 =>

if data_in = '1' then
yfsm <= s8;
```



```
else yfsm <= s1;
end if;

when s2 =>

if data_in = '1' then
yfsm <= s7;
else yfsm <= s2;
end if;

when s3 =>

if data_in = '1' then
yfsm <= s6;
else yfsm <= s3;
end if;

when s4 =>

if data_in = '1' then
yfsm <= s5;
else yfsm <= s4;
end if;

when s5 =>

if data_in = '1' then
yfsm <= s3;
else yfsm <= s5;
end if;

when s6 =>

if data_in = '1' then
yfsm <= s2;
```

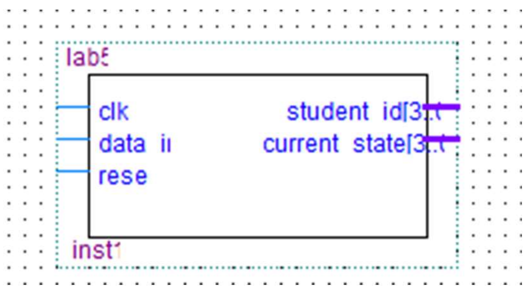


```vhdl
else yfsm <= s6;
end if;

when s7 =>

if data_in = '1' then
yfsm <= s1;
else yfsm <= s7;
end if;

when s8 =>

if data_in = '1' then
yfsm <= s0;
else yfsm <= s8;
end if;

end case;
end if;
end process;
process(yfsm, data_in)
begin
case yfsm is
when s0=>
student_id <="0101";
current_state <= "0000";
when s4 =>
student_id <="0000";
current_state <= "0100";
when s5 =>
student_id <="0001";
current_state <= "0101";
when s3 =>
student_id <="0000";
current_state <= "0011";
```

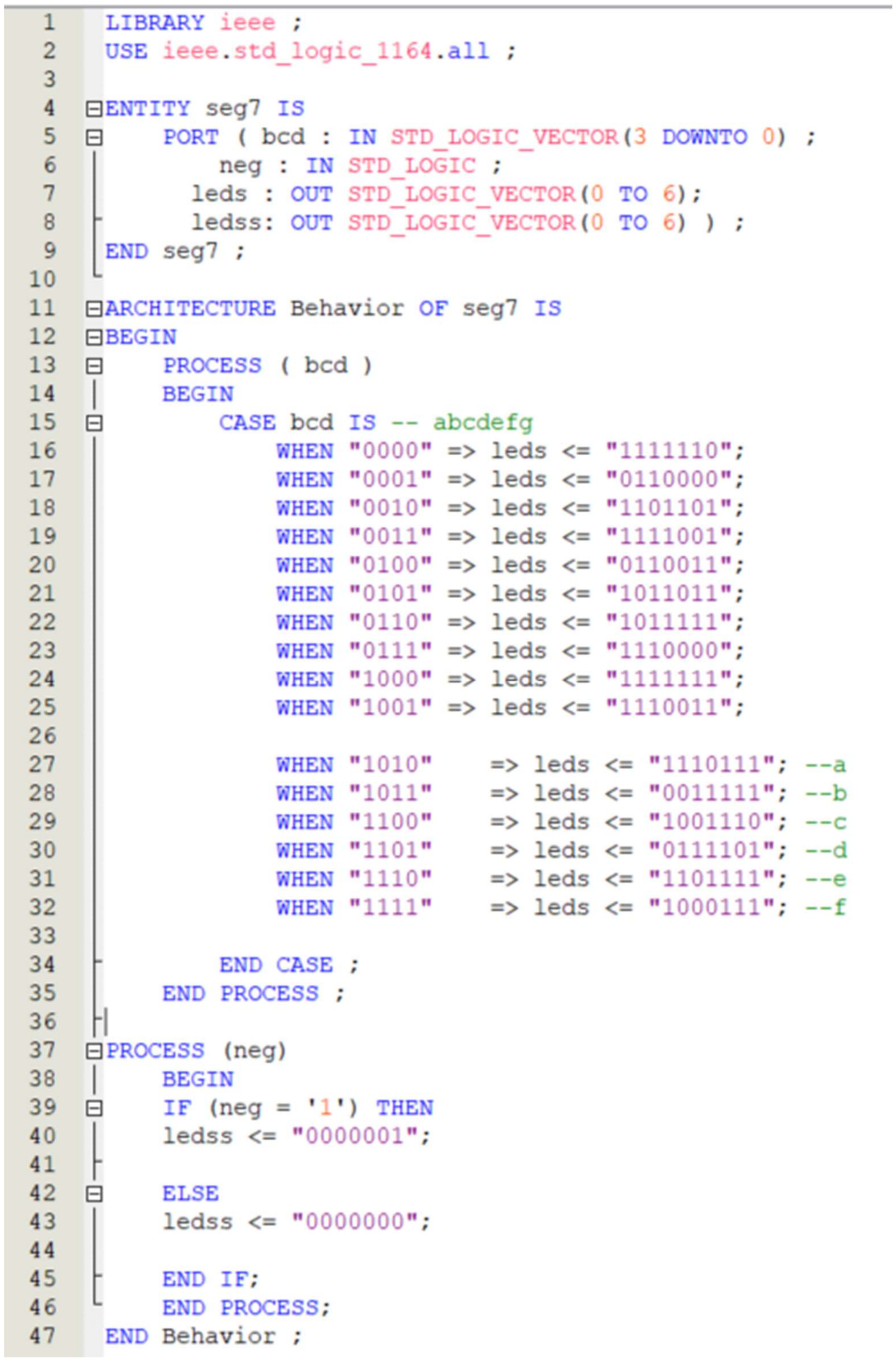


```
current_state <= "0011";
when s6 =>
student_id <="0001";
current_state <= "0110";
when s2 =>
student_id <="0001";
current_state <= "0010";
when s7 =>
student_id <="1000";
current_state <= "0111";
when s1 =>
student_id <="0101";
current_state <= "0001";
when s8 =>
student_id <="0010";
current_state <= "1000";
end case;
end process;
end architecture;
```

Converting the previous VHDL code into a block diagram:

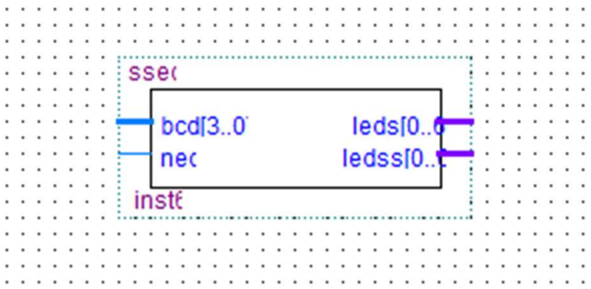


7 segment display code from lab 3:

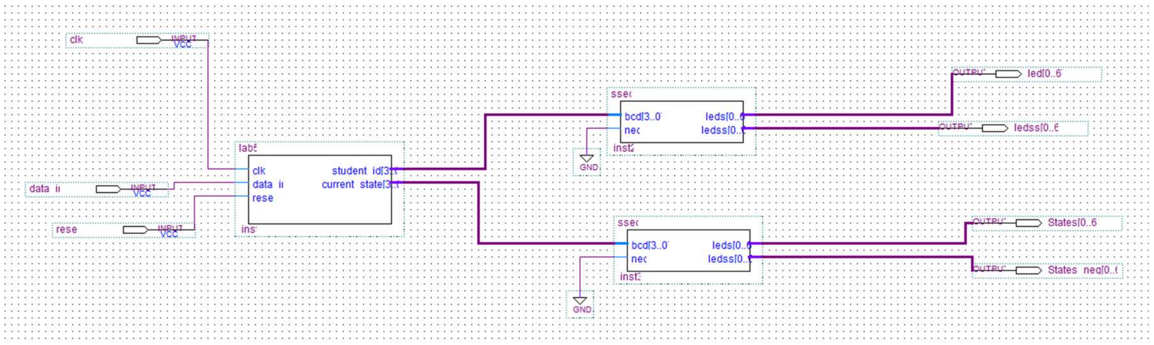


```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY seg7 IS
    PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
        neg : IN STD_LOGIC ;
      leds : OUT STD_LOGIC_VECTOR(0 TO 6);
      ledss: OUT STD_LOGIC_VECTOR(0 TO 6) ) ;
END seg7 ;

ARCHITECTURE Behavior OF seg7 IS
BEGIN
    PROCESS ( bcd )
    BEGIN
        CASE bcd IS -- abcdefg
            WHEN "0000" => leds <= "1111110";
            WHEN "0001" => leds <= "0110000";
            WHEN "0010" => leds <= "1101101";
            WHEN "0011" => leds <= "1111001";
            WHEN "0100" => leds <= "0110011";
            WHEN "0101" => leds <= "1011011";
            WHEN "0110" => leds <= "1011111";
            WHEN "0111" => leds <= "1110000";
            WHEN "1000" => leds <= "1111111";
            WHEN "1001" => leds <= "1110011";

            WHEN "1010"    => leds <= "1110111"; --a
            WHEN "1011"    => leds <= "0011111"; --b
            WHEN "1100"    => leds <= "1001110"; --c
            WHEN "1101"    => leds <= "0111101"; --d
            WHEN "1110"    => leds <= "1101111"; --e
            WHEN "1111"    => leds <= "1000111"; --f

        END CASE ;
    END PROCESS ;

PROCESS (neg)
    BEGIN
    IF (neg = '1') THEN
    ledss <= "0000001";

    ELSE
    ledss <= "0000000";

    END IF;
    END PROCESS;
END Behavior ;
```

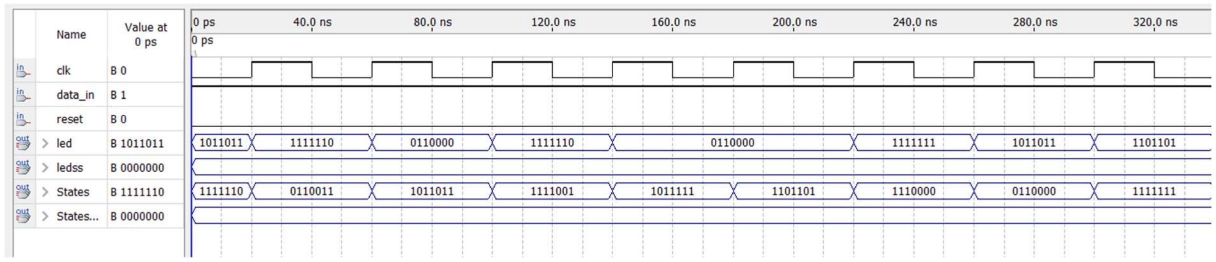
Converting the previous VHDL code into a block diagram:



Connecting the 7-segment display with the FSM in a circuit:



Waveform for the sequential circuit:



References

Brown, S. D., & Vranesic, Z. G. (2009). *Fundamentals of Digital Logic with VHDL Design*. New York, United States: McGraw-Hill Education.