Labs 3 - VHDL Design

Submitted by **Meldrick Reimmer, Khoi Pham and Selma Boudissa**

Contents

1 Problem 1 **3**

2 Problem 2 **3**

3 Problem 3 **4**

1 Problem 1

```
if (clk 'event and clk='1') then
    -- output the data value at a specific bit
    -- correspond to value of add chosen
    case add is
    when " 0000 " => s <= data (0);
    when " 0001 " => s <= data (1);
    when " 0010 " => s <= data (2);
    when " 0011 " => s <= data (3);
    when " 0100 " => s <= data (4);
    when " 0101 " => s <= data (5);
    when " 0110 " => s <= data (6);
    when " 0111 " => s <= data (7);
    when others => s <= 'Z';
    end case ;
end if;
```

Figure 1: Multiplexer algorithm

The Multiplexer is the algorithm to pick out a bit value of a data pattern correspond to the value of another pattern which we call the data selector

Figure 2: Test bench

Observation: Watch the pattern of data "01101001", whenever the clock hit its max, check value of "add" and update value of s proportionally to the bit position of "data". (ie: when add = "0001", get the value of position 2 of data "01101001" which is 0 and assign it to s).

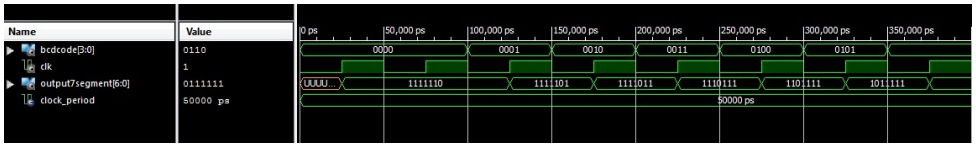
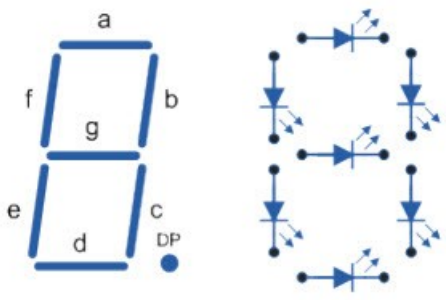


Figure 3: Test bench

2 Problem 2



Comment: The 7 segment display is a schematic to run a LED number display system based on a on/off predefined pattern. Notice the letter of the LED that will be used to point to in the program, the schematic algorithm is described as follow.

```
if (clk'event and clk='1') then
    case bcd is
        when "0000"=> output7Segment <="1111110";  -- 'display a led'
        when "0001"=> output7Segment <="1111101";  -- 'display b led'
        when "0010"=> output7Segment <="1111011";  -- 'display c led'
        when "0011"=> output7Segment <="1110111";  -- 'display d led'
        when "0100"=> output7Segment <="1101111";  -- 'display e led'
        when "0101"=> output7Segment <="1011111";  -- 'display f led'
        when "0110"=> output7Segment <="0111111";  -- 'display g led'
    end case;
end if
```

Figure 4: VHDL code

The binary rule in this schematic: "0" will turn on the LED while "1" turns it off The bits position in 8 bit pattern data correspond to the position of LED "a b c d e f g" that's how this system control the LED display. There is the last bit for the "point" but in 7 segment we don't activate it.

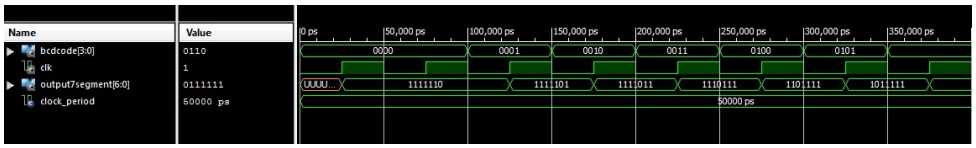


Figure 5: Test bench

Observation: The pattern of output changes according to the value of bcdcode (every time the clock hit max, ofcourse) followed by the above rule.

3 Problem 3

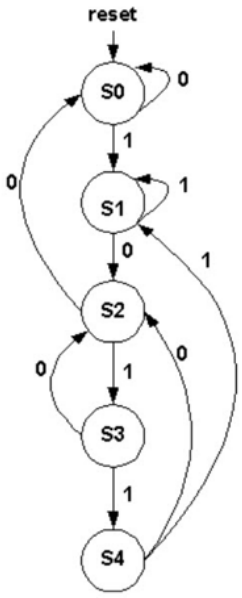
```
case current_s is
    when s0 =>
        if(input ='0') then
            next_s <= s0;
        else
            next_s <= s1;
        end if;

    when s1 =>;
        if(input ='0') then
            next_s <= s2;
        else
            next_s <= s1;
        end if;

    when s2 =>
        if(input ='0') then
            next_s <= s0;
        else
            next_s <= s3;
        end if;

    when s3 =>
        if(input ='0') then
            next_s <= s2;
        else
            next_s <= s4;
        end if;

    when s4 =>
        if(input ='0') then
            next_s <= s2;
        else
            next_s <= s1;
        end if;
end case;
```



Observation: Follow the graph of the exercise, I come up with my state transition program describe as above. The input is the condition for the program to set its next destination state.