VLSI Engineering Lab (Digital)

Experiment 4

Utkarsh Patel (18EC30048)

---

1 Objective

Design a circuit that takes 8-bit binary input and gives output "1" if the aggregate binary input is divisible by 3.

2 Theory

For a given 8-bit integer `in[7:0]`, it is required to check whether the decimal number constructed using only the first i MSBs is congruent to 0 (modulo 3). Hence, 8-bit binary output `out[7:0]` is needed to be generated such that

`out[i]` = 1, if the decimal constructed using `in[7:i]` is divisible by 3

= 0, otherwise

To generate `out[7:0]`, we used two D flip flops: S_1 and S_0 representing number after modulo 3.

- $S_1S_0 = 00$ denotes that the decimal constructed till current step is divisible by 3
- $S_1S_0 = 01$ denotes that the decimal constructed till current step leaves 1 as remainder when divided by 3
- $S_1S_0 = 10$ denotes that the decimal constructed till current step leaves 2 as remainder when divided by 3

We can construct a finite state machine using S_1 and S_0 such that it has only three states: 00, 01 and 10. Depending on the value of the current state and the value of the incoming bit, the FSM will transition into a new state, i.e., generate and store new remainder value.

Suppose at instant t, our aggregate input was X. Then, after taking the new bit I, the aggregate input changes as

$$X \coloneqq (X \ll 1) | I$$

This transition is same as $X \coloneqq 2X + I$. Following three cases arises:

Case 1: $S_1S_0 = 00$, X was divisible by 3, i.e., $X = 3k$. After transition, $X = 6k + I$.

- If $I = 0$, then $X = 6k = 3p$, hence new state is $S_1S_0 = 00$
- If $I = 1$, then $X = 6k + 1 = 3p + 1$, hence new state is $S_1S_0 = 01$

Case 2: $S_1S_0 = 01$, X leaves 1 as remainder when divided by 3, i.e., $X = 3k + 1$. After transition, $X = 6k + 2 + I$.

- If $I = 0$, then $X = 6k + 2 = 3p + 2$, hence new state is $S_1S_0 = 10$
- If $I = 1$, then $X = 6k + 3 = 3p$, hence new state is $S_1S_0 = 00$

Case 3: $S_1S_0 = 10$, X leaves 2 as remainder when divided by 3, i.e., $X = 3k + 2$. After transition, $X = 6k + 4 + I$.

- If $I = 0$, then $X = 6k + 4 = 3p + 1$, hence new state is $S_1S_0 = 01$
- If $I = 1$, then $X = 6k + 5 = 3p + 2$, hence new state is $S_1S_0 = 10$

Hence, the state transition table can built using this hypothesis.

Table 1. State transition table for registers S_1 and S_0

$S_1(t)$	$S_0(t)$	I	$S_1(t+1)$	$S_0(t+1)$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0

From Table 1, we can observe that the states for the two registers are given as

$$S_1(t+1) = \overline{S_1(t)} \cdot S_0(t) \cdot \bar{I} + S_1(t) \cdot \overline{S_0(t)} \cdot I$$

$$S_0(t+1) = \overline{S_1(t)} \cdot \overline{S_0(t)} \cdot I + S_1(t)\overline{S_0(t)} \cdot \bar{I}$$

3 Verilog codes

```
module modulo3(input wire[7:0] in, output reg [7:0] out);
    reg S1N, S1;
    reg S0N, S0;
    reg inp;
    integer i;
    integer val;
    always @ (in)begin
        S1N = 0;
        S0N = 0;
        out[7:0] = 8'b00000000;
        val = 0;
        for (i = 7; i >= 0; i = i - 1) begin
            inp = in[i];
            S1 = S1N;
            S0 = S0N;
            S1N = (~inp&~S1&S0)|(inp&S1&~S0);
            S0N = (inp&~S1&~S0)|(~inp&S1&~S0);
            if (S0N==0 && S1N==0) begin
                out[i] = 1;
                end
            else begin
                out[i] = 0;
                end
            val = 2 * val + out[i];
            $display("Input Bit = %b, Current Value = %3d, Output Bit = %b",in[i], val, out[i]);
        end
    end
endmodule


module tb_modulo3();
```

```
    reg[7:0] number;
    wire[7:0] out;
    modulo3 m1(.in(number), .out(out));
    initial begin
        $dumpfile("modulo3.vcd");
        $dumpvars();
        $monitor("Binary = %b, Decimal = %d, Output = %b\n", number, number, out);
        number = 8'b10101001;
        #200
        number = 8'b00000111; //7
        #200
        number = 8'b00101101; //45
        #200
        number = 8'b01100000; //96
        #200
        number = 8'b01100100; //100
        #200
        number = 8'b11111111; //255
        #200;
        $finish();
    end
endmodule
```

4 Output Log

```
Input Bit = 1, Current Value =   0, Output Bit = 0
Input Bit = 0, Current Value =   0, Output Bit = 0
Input Bit = 1, Current Value =   0, Output Bit = 0
Input Bit = 0, Current Value =   0, Output Bit = 0
Input Bit = 1, Current Value =   1, Output Bit = 1
Input Bit = 0, Current Value =   3, Output Bit = 1
Input Bit = 0, Current Value =   7, Output Bit = 1
Input Bit = 1, Current Value =  14, Output Bit = 0
Binary = 10101001, Decimal = 169, Output = 00001110
```

```
Input Bit = 0, Current Value =   1, Output Bit = 1
Input Bit = 0, Current Value =   3, Output Bit = 1
Input Bit = 0, Current Value =   7, Output Bit = 1
Input Bit = 0, Current Value =  15, Output Bit = 1
Input Bit = 0, Current Value =  31, Output Bit = 1
Input Bit = 1, Current Value =  62, Output Bit = 0
Input Bit = 1, Current Value = 125, Output Bit = 1
Input Bit = 1, Current Value = 250, Output Bit = 0
Binary = 00000111, Decimal =   7, Output = 11111010
```

```
Input Bit = 0, Current Value =   1, Output Bit = 1
Input Bit = 0, Current Value =   3, Output Bit = 1
Input Bit = 1, Current Value =   6, Output Bit = 0
Input Bit = 0, Current Value =  12, Output Bit = 0
Input Bit = 1, Current Value =  24, Output Bit = 0
Input Bit = 1, Current Value =  48, Output Bit = 0
Input Bit = 0, Current Value =  96, Output Bit = 0
Input Bit = 1, Current Value = 193, Output Bit = 1
Binary = 00101101, Decimal =  45, Output = 11000001
```

```
Input Bit = 0, Current Value =   1, Output Bit = 1
Input Bit = 1, Current Value =   2, Output Bit = 0
Input Bit = 1, Current Value =   5, Output Bit = 1
Input Bit = 0, Current Value =  11, Output Bit = 1
Input Bit = 0, Current Value =  23, Output Bit = 1
Input Bit = 0, Current Value =  47, Output Bit = 1
Input Bit = 0, Current Value =  95, Output Bit = 1
Input Bit = 0, Current Value = 191, Output Bit = 1
Binary = 01100000, Decimal =  96, Output = 10111111
```

5 Waveforms

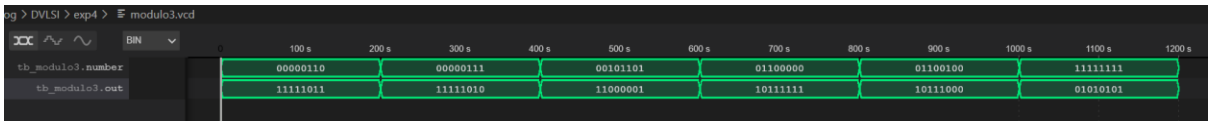

Fig. 1. The circuits takes as input 8-bit binary input `in[7:0]` and produces the desired output `out[7:0]` as per the algorithm described in Section 2: Theory.

6 Conclusion

- In this experiment, it was required to implement a circuit that takes 8-bit binary input and gives output "1" if the aggregate binary input is divisible by 3.

- For a given 8-bit integer `in[7:0]`, it is required to check whether the decimal number constructed using only the first i MSBs is congruent to 0 (modulo 3).

- Hence, 8-bit binary output `out[7:0]` is needed to be generated such that `out[i]` = 1, if the decimal constructed using `in[7:i]` is divisible by 3, otherwise `out[i]` = 0.

- To generate `out[7:0]`, we used two D flip flops: S_1 and S_0 representing number after modulo 3.
  - $S_1S_0 = 00$ denotes that the decimal constructed till current step is divisible by 3
  - $S_1S_0 = 01$ denotes that the decimal constructed till current step leaves 1 as remainder when divided by 3
  - $S_1S_0 = 10$ denotes that the decimal constructed till current step leaves 2 as remainder when divided by 3

- We can construct a finite state machine using S_1 and S_0 such that it has only three states: 00, 01 and 10. Depending on the value of the current state and the value of the incoming bit, the FSM will transition into a new state, i.e., generate and store new remainder value.

- Suppose at instant t, our aggregate input was X. Then, after taking the new bit I, the aggregate input changes as

$$X \coloneqq (X \ll 1) | I$$

- This transition is same as $X \coloneqq 2X + I$. Following three cases arises:
  - **Case 1**: $S_1S_0 = 00$, X was divisible by 3, i.e., $X = 3k$. After transition, $X = 6k + I$.
    - If $I = 0$, then $X = 6k = 3p$, hence new state is $S_1S_0 = 00$
    - If $I = 1$, then $X = 6k + 1 = 3p + 1$, hence new state is $S_1S_0 = 01$
  - **Case 2**: $S_1S_0 = 01$, X leaves 1 as remainder when divided by 3, i.e., $X = 3k + 1$. After transition, $X = 6k + 2 + I$.
    - If $I = 0$, then $X = 6k + 2 = 3p + 2$, hence new state is $S_1S_0 = 10$
    - If $I = 1$, then $X = 6k + 3 = 3p$, hence new state is $S_1S_0 = 00$
  - **Case 3**: $S_1S_0 = 10$, X leaves 2 as remainder when divided by 3, i.e., $X = 3k + 2$. After transition, $X = 6k + 4 + I$.
    - If $I = 0$, then $X = 6k + 4 = 3p + 1$, hence new state is $S_1S_0 = 01$
    - If $I = 1$, then $X = 6k + 5 = 3p + 2$, hence new state is $S_1S_0 = 10$

- From Table 1, we can observe that the states for the two registers as per current bit I are given as
  - $S_1(t+1) = \overline{S_1(t)} \cdot S_0(t) \cdot \bar{I} + S_1(t) \cdot \overline{S_0(t)} \cdot I$
  - $S_0(t+1) = \overline{S_1(t)} \cdot \overline{S_0(t)} \cdot I + S_1(t)\overline{S_0(t)} \cdot \bar{I}$

- Section 3 has the code used for implementing the required circuit.

- In the test bench module, the modulo3 function is checked for six different 8-bit binary input.

- From Fig. 1, it is observed that the module is producing correct result for all the 8-bit binary inputs.