Name: Rohan Jhaveri
CSU Id: 830962238

Lab Report 7

Goal: To design two sequential logic circuits viz. pulse generator and pyramid counter.

Functions:

Pulse Generator: Take input clock clk1 and generate different output signals viz. clk2 which is twice the time duration, clk4 which is 4 times the time period of clk1. The other outputs p1, p2, p3, p4 and p23 are basically combinational logic of the clock. This circuit is operational when clr signal is low and enb signal is high.

Pyramid Counter: It works when enb is high and clr is low. It just has one input i.e. clock clk and has two outputs pulse1 and pulse2. Initially pulse1 and pulse2 both are zero. When the clk runs for 15 cycles, pulse 1 goes high momentarily. The next time when clk runs for 14 cycles, pulse one goes high momentarily once again, then for 13 clock runs and so on. Finally, when this comes down to 0 pulse 2 goes high, indicating a reset pulse and the entire operation starts again.

Steps:

1. Prepared a truth table for p1, p2, p3, p4 and p23 outputs on the basis of clk1, clk2, clk4. (Refer Prelab)

2. Made Verilog code for Pulse Generator and analyzed the output.

3. Made Verilog code for Pyramid Counter and analyzed the output.

Verilog Code:

Pulse Generator:

```
module pulsegen (clr, en, c1, c2, c4, p1, p2, p3, p4, p23);
        input clr, en, c1;
        output c2, c4, p1, p2, p3, p4, p23;
        reg c2, c4, p1, p2, p3, p4, p23;
        always@ (negedge c1)
        begin
                case (en)
                1: begin
                If (clr == 0)
                        c2 <= ~c2;
```

```
        else
                c2 <= 0;
        end
        2: begin
        if (clr == 1)
                c2 <= 0;
        end
        endcase
        end
always@ (posedge c2)
        begin
        case (en)
        1: begin
        If (clr == 0)
                c4 <= ~c4;
        else
                c4 <= 0;
        end
        2: begin
        if (clr == 1)
                c4 <= 0;
        end
        endcase
        end
always@ (c1, c2)
        begin
        p1 <= ((c2) && (~c1));
```

```
            p2 <= ((c2) && (c1));
            p3 <= ((~c1) && (~c2));
            p4 <= ((~c2) && (c1));
            p23 <= ((p2) || (p3));
            end
    endmodule
```

Pyramid Counter:

```
module pyramid (clk, enb, clr, pulse1, pulse2);
input clk, enb, clr;
output reg pulse1, pulse2;
reg p1, p2;
integer a = 15, b = 0;

always@ (posedge clk)
begin
    case (clr)
        1'b1: begin
        p1 = 1'b0;
        p2 = 1'b0;
        end
        1'b0: begin
            case(a)
                0: a = 15;
                1: p2 = enb;
                default: p2 = 1'b0;
            endcase
```

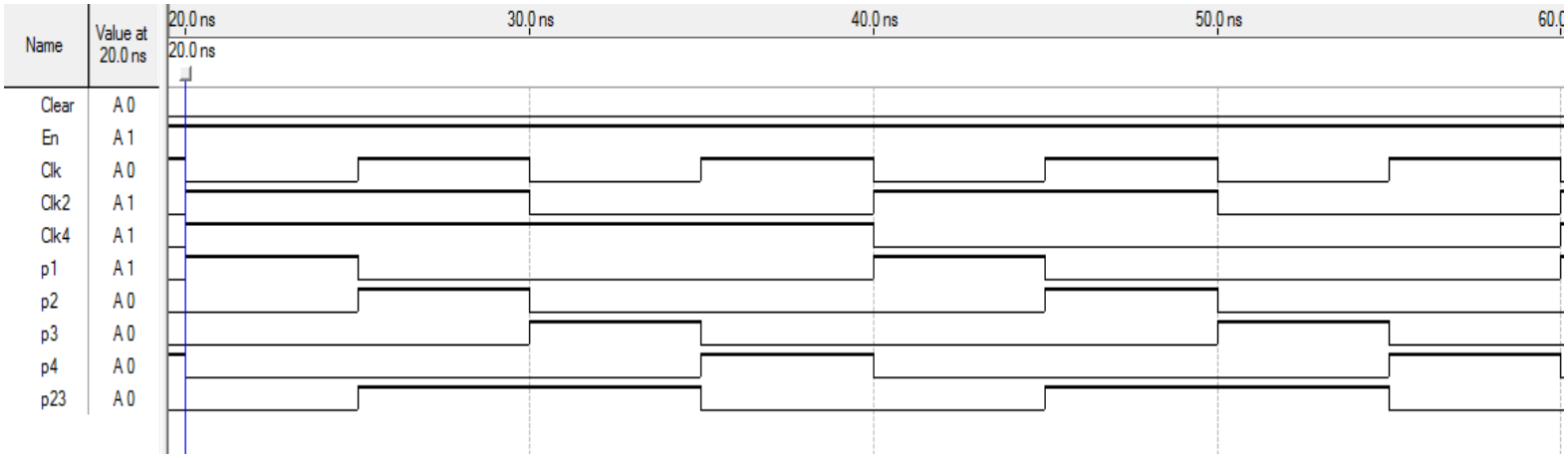
Name: Rohan Jhaveri
CSU Id: 830962238

```
                b = b+1;
                case(b)
                        a: begin
                        p1 = enb;
                        a = a-1;
                        b = 0;
                        end
                        default: p1 = 1'b0;
                endcase
        end
    endcase
    end
    always@ (clk) begin
    pulse1 = clk && p1;
    pulse2 = pulse1 && p2;
    end
endmodule
```

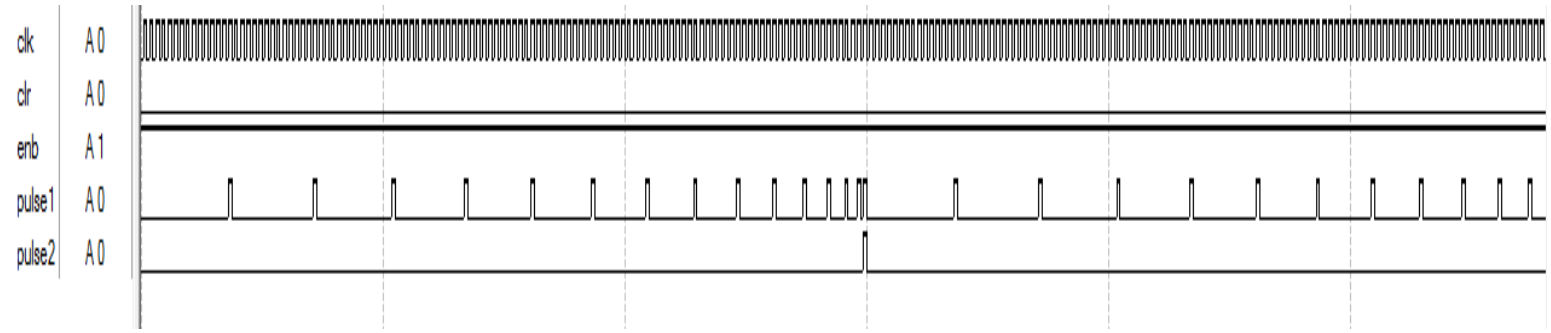
Name: Rohan Jhaveri
CSU Id: 830962238

Results and Graphs:

Pulse Generator:



Pyramid Counter:



Conclusion:

This lab thought us how to design an output from the clock signal and manipulating its frequencies. Then using these outputs to generate other outputs hieratically. Also, we learnt various aspects of Verilog to generate pulse generator and pyramid counter.

Questions:

Describe how you would implement a hierarchical design in Verilog.
-> To implement hierarchical design in Verilog, we divide the code into small modules which forms the hierarchy. The higher level modules call the modules underneath it and depends on their values. Hierarchical design helps in simplifying the code complexity and makes debugging easy as each module is a separate entity of code.