ECE451Lab6

Huanjia Liu (ID:831993149)

Nov 13, 2020

Contents

1 Objective

During this lab, we are asked to use Verilog to achieve pulse clock and pyramid counter.

The objective of this lab is to apply sequential logic design techniques to design and build two different blocks, a clock generator and a pyramid counter.

2 Introduction

2.1 Pulse Clock

The function of this circuit is to take an input clock signal and output the following signals. Here are three inputs: Enable, Clear and Clk. and 7 output pins:

clk2: An output clock signal whose period is twice the input clock signal.

clk4: An output clock signal whose period is four times the input clock signal.

p1, p2, p3 and p4: Four different pulse signals whose pulse width is half the cycle period of the input clock signal and whose period is twice the input clock signal. In addition, the cycle period of these pulse signals are equally divided into four sub-phases (sub-periods) and each sub-phase must be occupied by only one of the four pulses.

p23: Another pulse signal whose period is the same as signals p1, p2, p3 and p4, but the width and the location of the pulse are the second and third sub-phases.

The relationship between inputs and outputs is shown in the figure below.

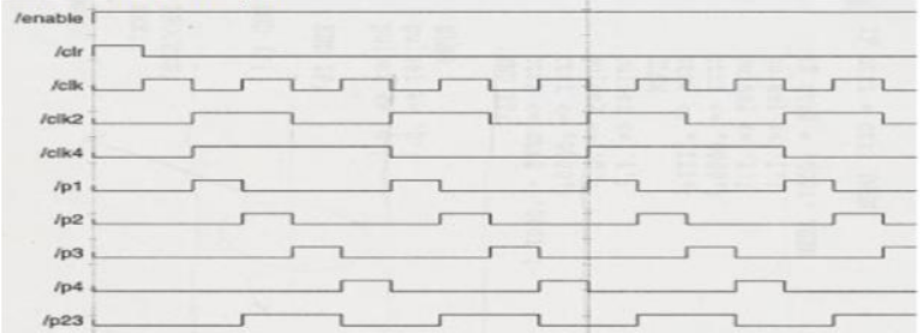

Figure 1: Relationship between inputs and outputs

2.2 Pyramid counter

A counter that counts from 0 to 15, then pulses a control signal, pulse1, which reconfigures the counter to count from 0 to 14 (or 1 to 15) and pulses the control signal again, then counts 0 to 13 (or 2 to 15) and pulses the control signal, etc. When the count goes from 0 to 1 (or 14 to 15) a second control signal, pulse2, is pulsed. The relationship between the input signal and the output signals is illustrated below.

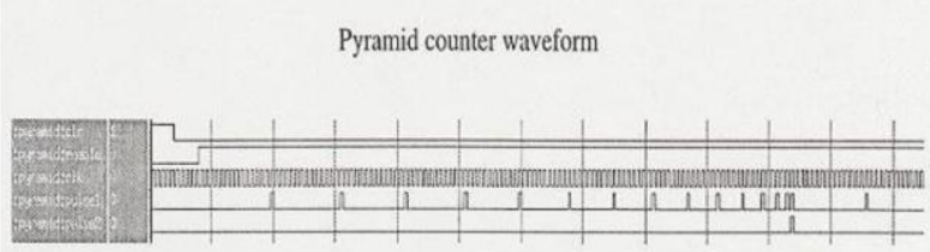

Figure 2: Relationship between inputs and outputs

3 Procedure and analysis

3.1 Pulse clock

Clk2	Clk	P1	P2	P3	P4	P23
0	0	0	0	1	0	1
0	1	0	0	0	1	0
1	0	1	0	0	0	0
1	1	0	1	0	0	1

From the truth table above, we can get write down the boolean equation below.

$$P1 = Clk2*Clk'$$

$$P2 = Clk2*Clk$$

$$P3 = Clk2'*Clk'$$

$$P4 = Clk2'*Clk$$

$$P23 = Clk2'*Clk' + Clk2*Clk$$

Then we can write down the Verilog code below, here we generate two sub module called c_clk2 and c_clk4 for creating signals Clk2 and Clk4. Also the register variable **real_enable** is for synchronous.(If enable signal becomes active in the middle of a clock period the circuit is not enabled until the next rising clock edge).

```
module lab6(clk, enable, clear, clk2, clk4, p1, p2, p3, p4, p23);
input clk, enable, clear;
output p1, p2, p3, p4, p23, clk2, clk4;
reg p1, p2, p3, p4, p23;

c_clk2 ins1(.clk(clk), .clk2(clk2));
c_clk4 ins2(.clk2(clk2), .clk4(clk4));

reg real_eanble;
```

```
always@(posedge clk)
begin
   if(enable)
   begin
      real_eanble = 1;
   end
   else
   begin
      real_eanble = 0;
   end
end


always@(clk)
begin
if(real_eanble)
   begin
      p1 <= (!clk)&(clk2);
      p2 <= (clk)&(clk2);
      p3 <= (!clk)&(!clk2);
      p4 <= (clk)&(!clk2);
      p23 <= ((clk)&(clk2))|((!clk)&(!clk2));
   end
if(clear)
begin
   p1 <= 0;
   p2 <= 0;
   p3 <= 0;
   p4 <= 0;
```

```
   p23 <= 0;
end
end

endmodule



module c_clk2(clk, clk2);
input clk;
output reg clk2;

always@(negedge clk)
   begin
   clk2 = ~clk2;
   end
endmodule



module c_clk4(clk2, clk4);
input clk2;
output reg clk4;

always@(posedge clk2)
   begin
   clk4 = ~clk4;
   end
endmodule
```

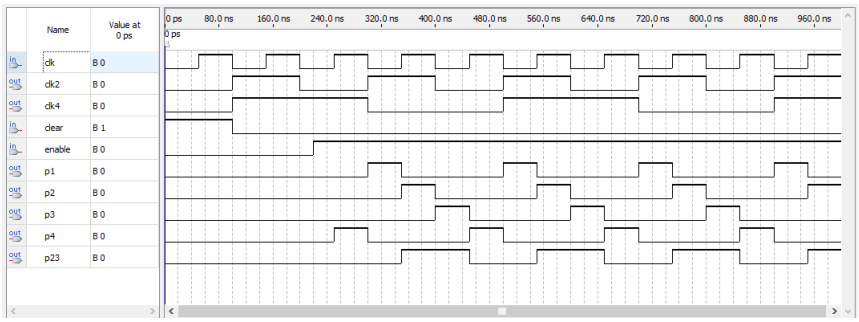

Figure 3: Waveform of pulse clock

3.2 Pyramid counter

Here is the Verilog code for my Pyramid counter.

```
module lab62(clk, clear, pulse1, pulse2, count1, count2);

input clk, clear;

output reg pulse1, pulse2;


output reg[3:0] count1, count2;


always@(posedge clk)

begin

if(~clear)

begin

   if(count1 == (4'b1111 - count2 ))

   begin

      count1 = 0;

      count2 = count2 + 1'b1;

      pulse1 = 1;

   end

   else

   begin
```

```
      count1 = count1 + 1'b1;
      pulse1 = 0;
      pulse2 = 0;
   end
   if(count2 == 15)
   begin
      pulse2 = 1'b1;
      count2 = 0;
   end
end
else
begin
   count1 = 0;
   count2 = 0;
   pulse1 = 0;
   pulse2 = 0;
end
end

endmodule
```

And here is my waveform of Pyramid counter. Works pretty good.

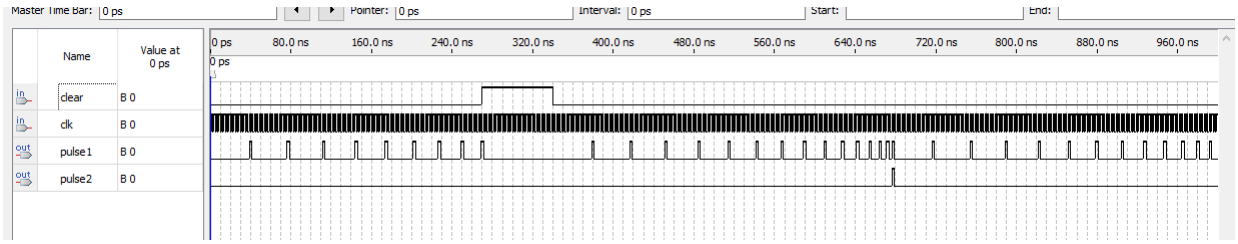

Figure 4: Waveform of Pyramid counter

4 Questions

4.1 Describe how you would implement a hierarchical design in Verilog.

In this lab, I implement the hierarchical design by writing sub modules. For example, in pulse clock, I write two modules to generate Clk2 and Clk4, thus I can instantiate them in my main code directly, and don't need to worry about the detail anymore.

5 Conclusion

Now we can get the conclusion, I finish this lab successfully, and achieve all the objective. During the lab, I have learned about some knowledge about synchronous, and I applied it using reg variable **real_enable** in my pulse clock, it can enable the circuit until the next rising clock edge, even I set enable signal becomes active in the middle of a clock period.