Assignment 2
Submitted By :
Snehal Gupta
2016201

Question 1:

Architecture-

In 7 series FPGA, there is Clock Management Tile(CMT) which includes PLL and MMCM. Each CMT has one PLL and one MMCM .

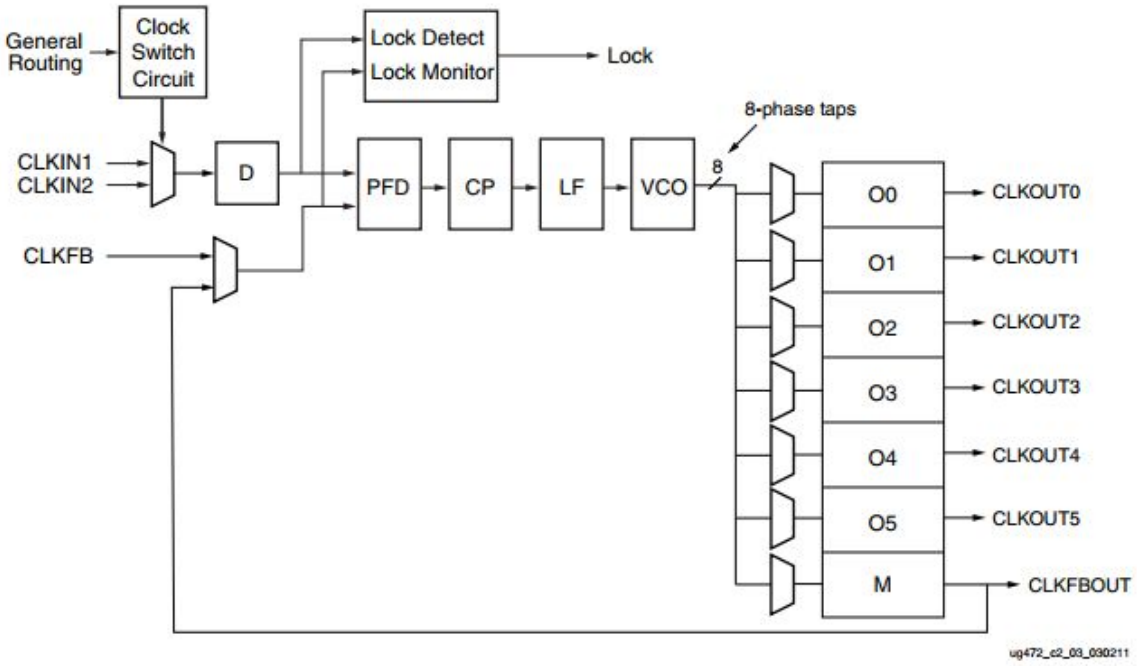


Figure 3-3: **Detailed PLL Block Diagram**

Explanation of PLL Diagram:

1. **D is input clock divider.**
2. **PFD stands for Phase Frequency Detector .The PFD detects the difference in phase and frequency between the reference clock and feedback clock inputs and generates an up or down control signal based on whether the feedback frequency is lagging or leading the reference frequency. The PFD outputs these up and down signals to a charge pump.**
3. **CP stands for Charge Pump which is responsible for integrating changed information from PFD.If the charge pump receives an up signal, current is driven into the loop filter. Conversely, if it receives a down signal, current is drawn from the loop filter.**

4. **LF stands for Loop Filter and is responsible for filtering CP signal and limits the rate of change of VCO.The loop filter converts the signals to a control voltage that is used to bias the VCO.**
5. **VCO :Based on the control voltage, the VCO oscillates at a higher or lower frequency, which affects the phase and frequency of the feedback clock.It can generate eight phases which are 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°.**
6. **O0 - O5 : These are output counters which divide the VCO frequency independently and can take any of the 8 phases.**
7. **M stands for Multiplier which is inserted in the feedback loop to increase the VCO frequency above the input frequency.Any one of the O0-O5 outputs can be used via the CLKFBIN path for feedback.**

Features-

1. **In PLL,the output frequency depends on M,D,O values. M and D values are defined for the entire PLL and O value is programmable for each of the six distinct output frequencies.**
2. **In order to allow greater frequency generation capability, there is cascaded routing connections between clock manager blocks that are in the same tile.**
3. **There is frequency lock between F(PFD) and CLKFB and phase lock between ClkFB and ClkIn.**
4. **The control loop of the PLL keeps ClkIn and ClkFBIN in phase regardless of the delay between ClkFBOUT and ClkFBIN.**

How VCO output frequency affects power consumption of FPGA :

In context of power consumption , when the charge pump receives an up signal, Since the current is driven into the loop filter and VCO needs to operate at high frequency , power consumption of FPGA increases . If it receives a down signal,current is drawn from the loop filter and VCO needs to operate at lower frequency, power consumption decreases.

Question 2:

1. **Clock Monitor functionality helps in monitoring the clock inputd to the MMCM/PLL .**
2. **We can monitor if the input frequency is out of range than the expected frequency.**
3. **We can detect clock stop.**
4. **We can detect glitches in the clock.**

It has 9 ports :

Input :

1. **1 port for reference clock which is used to monitor user clock.**
2. **4 ports for user clocks.**

Output :

1. **Clk_stop[3:0] : For stopping clock at the respective user clock.**
2. **Clk_glitch[3:0] : For indicating a glitch in the user clock.**
3. **Clk_oor[3:0] : For indicating out-of-range input frequency.**
4. **Interrupt : To interrupt the clock monitor feature.**

There are 32 clock monitor registers :

1. **4 to indicate if the clock frequency is greater than the specification.**
2. **4 to indicate if the clock frequency is lesser than the specification.**
3. **4 to indicate if there is a glitch in the input clock.**
4. **4 to stop a clock.**
5. **16 undefined.**

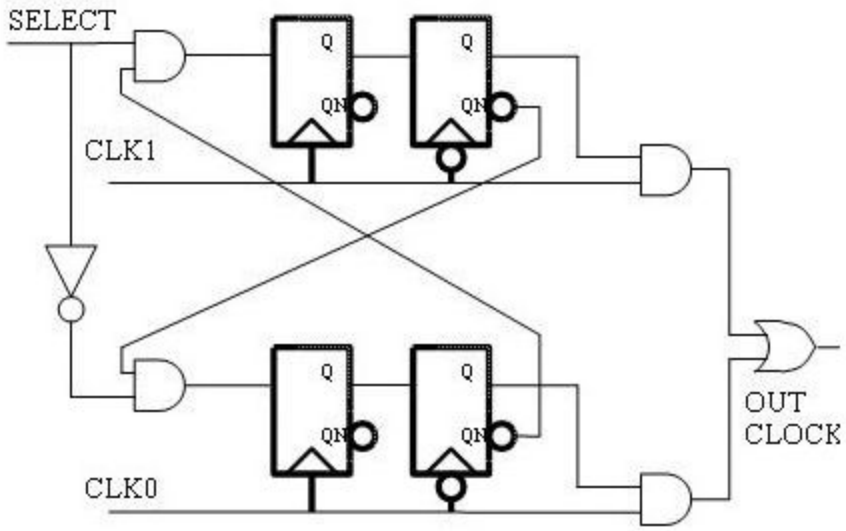
Question 3:

```
//dff.v
module dff(
   input clk1,
   input clk2,
   input sel,
   output out_clk
   );
   reg q1,q2,q3,q4,qn1,qn2,qn3,qn4;
   wire and1,and2,selc,f1,f2;
   not n1(selc,sel);
   and a1(and1,sel,qn4);
   and a2(and2,selc,qn2);
   and a3(f1,q2,clk1);
   and a4(f2,q4,clk2);
   or o1(out_clk,f1,f2);
   initial
   begin
     q1=1'b0; qn1=1'b1;
     q2=1'b0; qn2=1'b1;
     q3=1'b0; qn3=1'b1;
     q4=1'b0; qn4=1'b1;
```

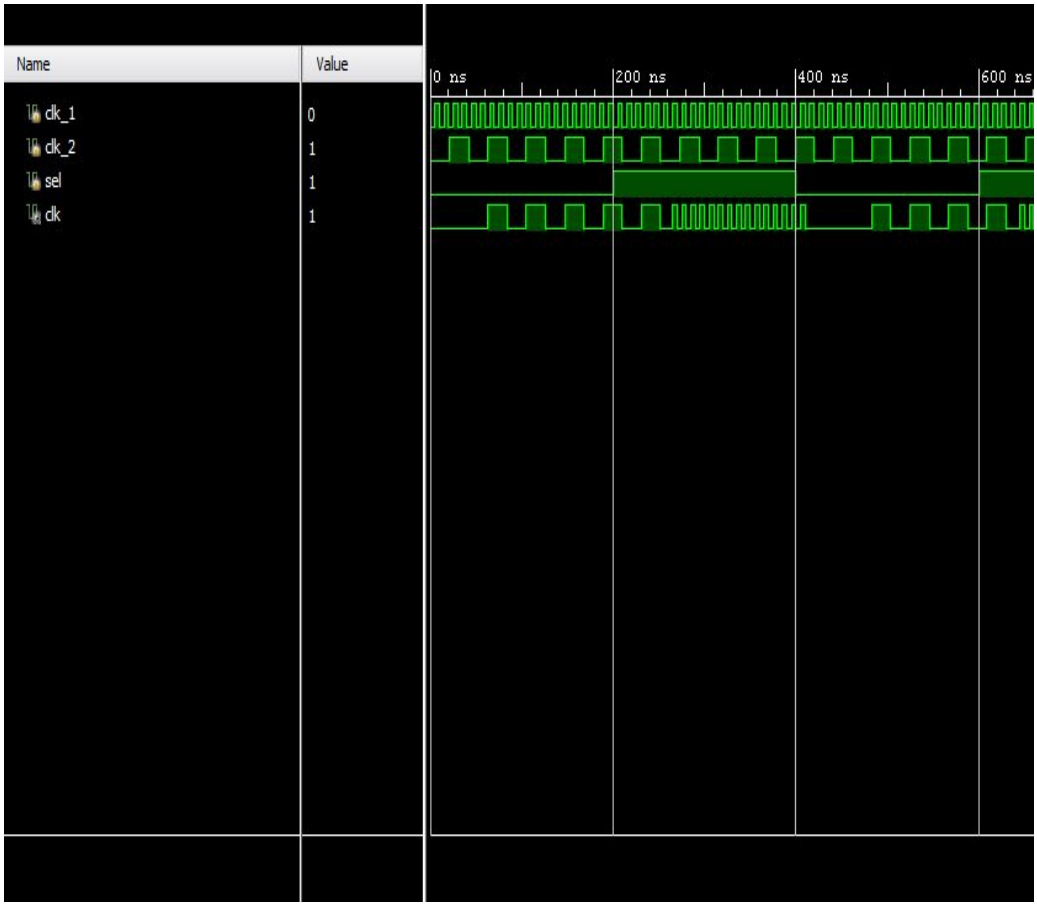
```
    end

    always @ (posedge clk1)
    begin
       q1=and1;
       qn1= ~and1;
    end
    always @ (negedge clk1)
    begin
    q2=q1;
    qn2= ~q1;
    end
    always @ (posedge clk2)
       begin
          q3=and2;
          qn3= ~and2;
       end
    always @ (negedge clk2)
       begin
          q4=q3;
          qn4= ~q3;
       end

endmodule

//testb.v
module testb;
    reg clk_1;
    reg clk_2;
    reg sel;
    wire clk;
    dff ans(clk_1,clk_2,sel,clk);

   initial
       begin
       clk_1 = 0;
       clk_2 = 0;
       sel = 0;
       end
```

```
  always #5 clk_1 = ~clk_1;
  always
  begin
  #1;
  #20 clk_2 = ~clk_2;
  end
  always #200 sel = ~sel;

endmodule
```

Circuit implemented :



Simulated waveform :



Explanation :
Frequency of clock-1 : 0.2 Hz
Frequency of clock-2 : 0.05 Hz
Phase shift of clock-2 wrt clock-1 : 72 degree
In the waveform, there are three stages :

1. **0ns-200ns : Initially , Q and QN of all the d flip-flops have been set to 0 and 1 respectively and select line is 0.Since the select line is 0, the top input of or gate remains 0 during this interval. At the first positive edge of clock2, q of bottom left d flip flop becomes 1 and it becomes input for bottom right d flip-flop.At the first negative edge of clock2 , this input is passed on to the output of bottom right d flip-flop and thus value of AND of 1 and clk2 becomes clk2 and is displayed at the output . Till 200ns , since the select line and qn of top right flip flop doesnt change,there is no change in the output of flip-flops and hence , clock2 gets displayed at output.**
2. **200ns-400ns : After 200ns , select line changes to 1 . But,since the value of qn of bottom right flip-flop is still zero, the final output wont be affected.This change will affect the output of bottom left d flip-flop only**

when positive edge of clock2 is encountered. So,the output remains same unless positive edge of clock2 is encountered.At the positive edge of clock2, q of bottom left d flip-flop becomes 0.At the negative edge, 0 is passed onto the output of bottom right d flip-flop . so,bottom input to or gate becomes 0. Now qn of bottom right d flip-flop becomes 1. So, the input to top-left d flip-flop becomes 1.After this,at the positive edge of clock1 , this value is passed to output of top-left flip-flop . At negative edge the value is further passed to output of top-right flip-flop. So,the value of AND of 1 and clock 1 is clock1 and it gets displayed at the output.Till 400 ns,since the select line doesnt change,there is no change in the output of flip-flops and hence,clock-1 gets displayed at the output.

3. **400ns-600ns : After 400ns , select line changes to 0. But since the value of qn of top-right flip-flop is still zero, the final output wont be affected. When the positive edge of clock1 is encountered, value 0 is passed to the output of top-left flip-flop . After this,when negative edge of clock1 is encountered,This value is further passed to the output of top right d flip-flop which changes qn of top-right d flip-flop and hence the input of bottom left d flip-flop. Now the upper input to OR gate has become zero and the value of bottom input to OR gate is zero since the previous stage(200ns-400ns).So,0 is displayed at the output unless positive edge of clock2 is encountered.When the positive edge is encountered , the input value 1 is passed onto the output of bottom left d flip-flop. When the negative edge is encountered , this value is further passed to the output of bottom right d flip-flop . The value of AND of 1 and clock2 becomes clock2 and final output becomes clock2 . Till 600ns, since there is no change in the select line,no change is reflected at the final output.**

Question 4:

```
//sqr.v
`timescale 1ns / 1ps
module sqr(
x_in,
x_out,
clk
   );
```

```
input [15:0]x_in;
output [15:0]x_out;
input clk;

wire en;
cordic_0 obj1 (clk,1'b1,x_in,en,x_out);

endmodule
```

Simulated waveform :

The three inputs given are :

Binary Equivalent	Decimal Equivalent	Hexadecimal Equivalent
11100001	225	e1
01111001	121	79
01100100	100	64

11111111	**255**	**ff**

For the corresponding inputs , correct outputs in decimal representation should be :

1. **225 - 15**
2. **121 - 11**
3. **100 - 10**
4. **255 - 15.96**

Outputs obtained are :

Hexadecimal equivalent	Decimal Equivalent
0f	**15**
0b	**11**
0a	**10**
0f	**15**

Hence, the square root functionality of ip catalog gives exact output for perfect squares , but for other integers , it shows only the integer part on output because of use of unsigned integers.