

James Cook University
Electrical and Computer Engineering
EE4306 Assignment

Introduction

This assignment is worth 10% of the total of the course and covers VHDL programming and Asynchronous Sequential Logic.

The assignment is to be submitted by 10am on the 11th of September.

Task

Design an Phase Locked Loop IC to include an Exclusive OR type Phase Frequency detector and a divide by N network to permit a VCO to be controlled in 1 kHz steps from 11.0 MHz to 11.25 MHz. Complete the design for the PLL circuit and program this into a Lattice ispLSI2032 using the JCU-ECE EPLD development board to verify your EPLD design and using the VHDL programming language.

The divider is controlled using jumpers connected to pins 25 to 32 of the EPLD board (pin 25 is the least significant bit of the control word). The LED on pin 37 is to be used to indicate an out of lock condition with the VCO being a higher frequency than the 1 kHz Reference Frequency and the LED on pin 38 is to indicate an out of lock condition with the VCO being a lower frequency than the 1 kHz Reference Frequency. The Phase detector output and its complement are to appear on pins 3 and 4 respectively. The inputs from the VCO are to be at pin 9 and the Reference input frequency is to be on pin 10. The Reset pin is connected to pin 35 and active LOW, ie under normal conditions (no reset) it is HIGH. To facilitate testing, the frequency divider output is to be at pin 8.

Email your complete tested design, with the whole directory containing the Lattice Project Zipped as one file. Use a unique file name for this Zip file, as the email program results in another file of the same name to be overwritten by a file of the same name when receiving an attachment. So don't use the name
d get several of those and only one, the last one will be kept in the mail-box.
Ensure your name is included in the source code as well as in the file name sent.

The solution will be posted on the web shortly after the submission date.

C. J. Kikkert
21 Aug 00
V2

Solution

The phase detector logic signal flow graph is as per lecture notes and is:

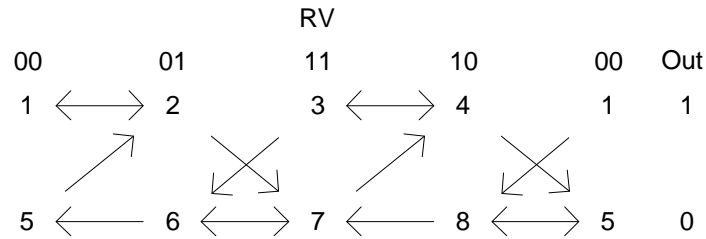


Figure 1.

The corresponding primitive flow table is:

RV				
00	01	11	10	Out
1	2		8	1
5	2	3		0
	6	3	4	1
1		7	4	0
5	2		8	0
1	6	3		1
	6	7	4	0
1		3	8	1

Table 1: Primitive Flow Table

In this table the Bold

Merging is possible between states:

1 and 8, 2 and 5, 3 and 6, 4 and 7, 6 and 8. All of these except 6 and 8 are needed. The merged flow table is thus:

RV				
00	01	11	10	Out
1	2	3	8	1
5	2	3	8	0
1	6	3	4	1
1	6	7	4	0

Table 2: Merged flow table

Note for normal operation, the sequence 1, 2, 3, 4, 1 etc will occur. Keeping the rows in this order, will ensure that only one variable changes during each change in input. This should minimise the possibility of a glitch occurring in the output

	RV				
fg	00	01	11	10	Out
00	1	2	3	8	1
01	5	2	3	8	0
11	1	6	3	4	1
10	1	6	7	4	0

Table 3: Transition Matrix

This results in the excitation matrix:

	RV				
fg	00	01	11	10	Out
00	00	01	01	00	1
01	01	01	11	00	0
11	10	11	11	10	1
10	00	11	10	10	0

Table 4: Excitation Matrix

This results in a Karnaugh map for F:

F	RV				
fg	00	01	11	10	Out
00	0	0	0	0	1
01	0	0	1	0	0
11	1	1	1	1	1
10	0	1	1	1	0

Table 5a Karnaugh map

$$\text{so that } F = f \bullet g + f \bullet V + f \bullet R + R \bullet V \bullet g$$

and a Karnaugh map for G:

G	RV				
fg	00	01	11	10	Out
00	0	1	1	0	1
01	1	1	1	0	0
11	0	1	1	0	1
10	0	1	0	0	0

Table 5b Karnaugh map

$$\text{so that } G = R' \bullet V + f' \bullet V + g \bullet V + f' \bullet g \bullet R'$$

The output is a 1 when fg is 00 or 11, as can be seen from the maps above, a separate Karnaugh map for the output coding is thus not required.

$$\text{Output} = f \text{ xor } g.$$

The unlock light high (ULH) should be ON in states 5 and 7 and the unlock light low (ULL) should be ON in states 6 and 8. The transitional states are all don't cares. The output coding maps for ULH and ULL are as follows:

ULH	RV			
fg	00	01	11	10
00	0	X	X	0
01	1	0	X	X
11	X	0	0	X
10	X	X	1	0

Table 6a Karnaugh map for UnLock High Indicator

The resulting output coding is $ULH = f' \cdot g \cdot V' + f \cdot g' \cdot V$

G	RV			
fg	00	01	11	10
00	0	1	1	0
01	1	1	1	0
11	0	1	1	0
10	0	1	0	0

Table 6a Karnaugh map for UnLock Low Indicator

The resulting output coding is $ULL = f' \cdot g' \cdot R + f \cdot g \cdot R'$

This results in the VHDL code for these equations as:

```

PhDet_P: process (R,V,F,G)          --Phase Detector Process
begin
    F <= ((F and G) or (F and V) or (F and R) or (R and V and G)) and Res;
    G <= ((not R and V) or (not F and V) or (G and V) or (not F and G and not R)) and Res;
    PDO <= F xor G ;
    CPDO <= not PDO ;
    ULH <= (not F and G and not V) or (F and not G and V);  --PLL unlocked VCO High
    ULL <= (not F and not G and R) or ( F and G and not R);  --PLL unlocked VCO Low
end process PhDet_P;

```

Divide by N network:

The divide by N network can very easily be written in VHDL. The difference between the upper value of the counter and the lower value of the counter is the frequency divider ratio. One way of operating the counter is to start at zero and count to the required number between 11000 and 11250. That number must be calculated using an addition of 11000 and the switch setting. It is unlikely that that will fit into the EPLD with the limited resources available. If the last 8 bits of the counter are zero at the lowest number, the 8 least significant bits can simply be gated in thus avoiding the addition. However that is not the situation in this application.

The other way of doing this is to start at say the number that is set by the switches and end at the largest number. In the design below, we start at the largest number and count down till the number set by the switches. This then does not involve any addition.

The waveforms for the divider and the phase detector are shown in the two plots below. The first one is a large scale view and the second one is a close-up

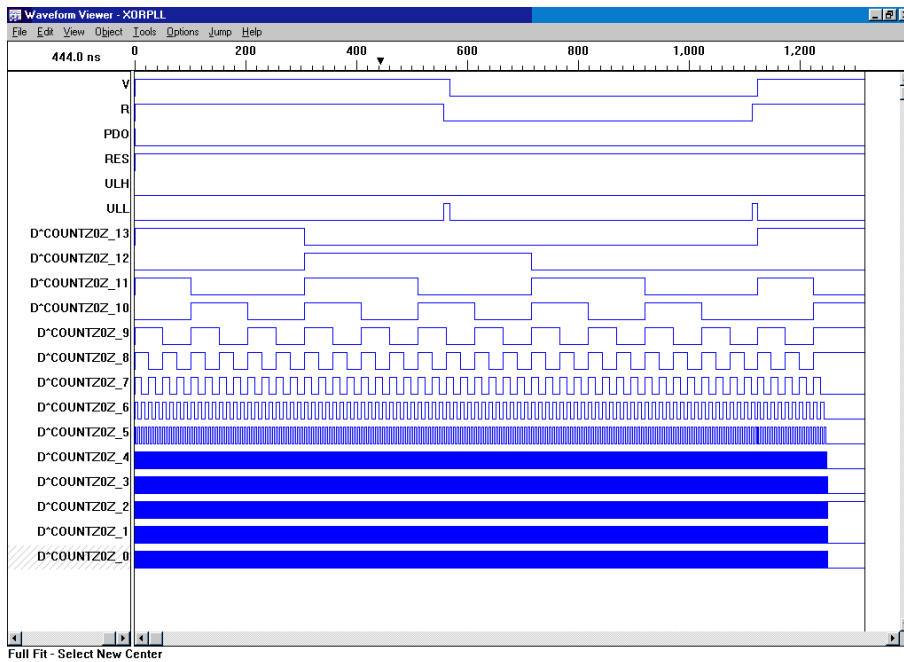


Fig 7a. Full test vector time plot

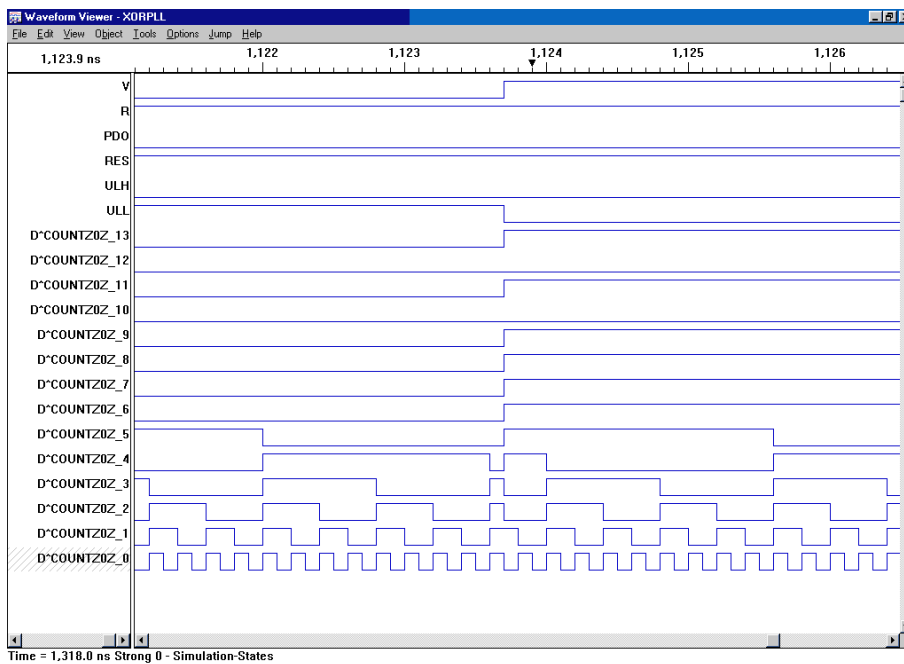


Fig 7b. Close-up test vector time plot

The value for the switches was set to 00001111. It can be seen that the divider works as expected, with the counter loading the value of 11200 (decimal) when 0F (hex), which corresponds to the switch setting is reached.

```
-- Solution to Assignment 1 for EE4306
--Author C. J. Kikkert Sep 2000
--James Cook University
```

```
library ieee;
use ieee.std_logic_1164.all;
```

```
entity XORPLL is
```

```
port(Clk,R,Res : in std_logic;
```

```
      V : inout std_logic;
```

```
      ULL, ULH, PDO, CPDO : inout std_logic;
```

```
      NLoad : in integer range 0 to 255); -- NLoad is the number at which the counter stops
```

```
                                     -- for Div by 11200 set NLoad
```

```
to 255 NotLoad = 0;
```

```
attribute lock: string;
```

```
attribute lock of PDO : signal is "3"; -- pin allocation for all inputs and outputs
```

```
attribute lock of CPDO : signal is "4";
```

```
attribute lock of Res : signal is "7";
```

```
attribute lock of clk : signal is "9";
```

```
attribute lock of V : signal is "8";
```

```
attribute lock of R : signal is "10";
```

```
attribute lock of ULH : signal is "37";
```

```
attribute lock of ULL : signal is "38";
```

```
attribute lock of NLoad : signal is "25,26,27,28,29,30,31,32";
```

```
end;
```

```
architecture XORPLL_arch of XORPLL is
```

```
    signal Count : integer range 0 to 11250; -- counter for frequency divider max 11250
```

```
    signal F, G : std_logic; -- feedback variables of asynch
```

```
sequential logic
```

```
begin
```

```
    PhDet_P: process (R,V,F,G) --Phase Detector Proces
```

```
begin
```

```
    F <= ((F and G) or (F and V) or (F and R) or (R and V and G)) and Res;
```

```
    -- and Res used to enable waveform viewing, without getting
```

```
ambiguities
```

```
    G <= ((not R and V) or (not F and V) or (G and V) or (not F and G and not R)) and Res;
```

```
    PDO <= F xor G ;
```

```
- Output from phase detector
```

```
    CPDO <= not PDO ;
```

```
- Complementary Output
```

```
    ULH <= (not F and G and not V) or (F and not G and V); --PLL unlocked VCO High
```

```
    ULL <= (not F and not G and R) or ( F and G and not R); --PLL unlocked VCO Low
```

```
end process PhDet_P;
```

```
    Count_P: process (Clk)
```

```
    --Divide by N Circuit
```

```
begin
```

```
    if rising_edge(Clk) then
```

```

to max count.
--
count
    if Count <= NLoad then          --if Count =< value preset by switches, load
        Count <= 15;                --for testing divide by 16
        Count <= 11250;              --11250 value to load, maximum

        V <= '1';                    --load max value of counter
    else                             --now flip output V if needed.
        if (Count <= 5567) then      --ideal 5561 uses 7 GLB, 5567 uses 6 GLB's
        --
        if (Count <= 7) then        --for testing only divide by 16 max
            V <= '0';
        end if;
        Count <= Count - 1;         --down counter is used,
    end if;
end if;
end process Count_P;
end XORPLL_arch;

```