

EE230 – HW3 Report

PLL System Design

(Using VerilogA & Matlab)

Muhammad Aldacher

Student ID: 011510317

1. Schematic of the PLL system:

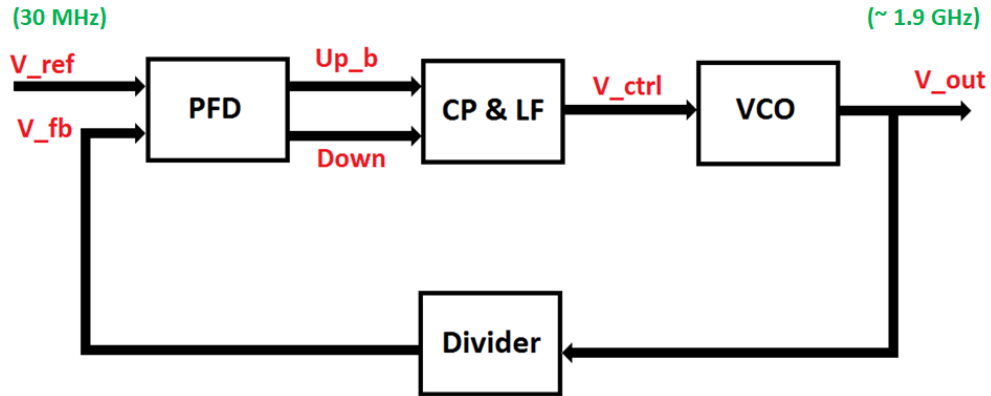


Fig. 1. PLL System Block Diagram

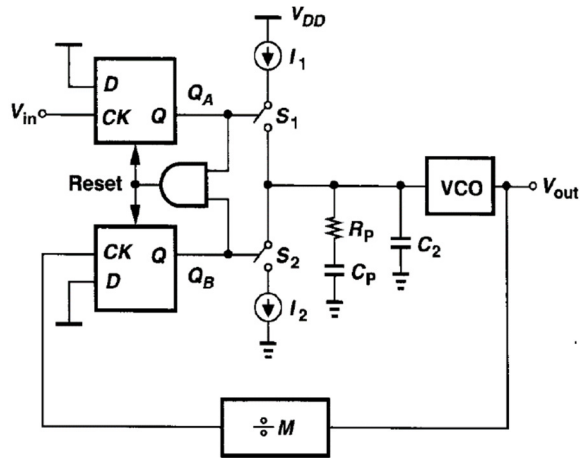
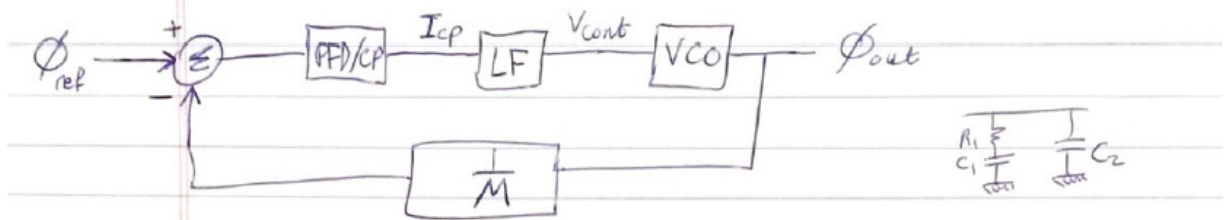


Fig. 2. Typical Charge Pump PLL System

Table 1. System Parameters Used

Parameter	Value
F_{REF}	30 MHz
F_{OUT}	1.9 GHz
$M_{Divider}$	64
I_{CP}	100 μ A
K_{VCO}	600 MHz/V
R_P	6.5 K Ω
C_P	100 fF
C_2	10 fF

2. System Analysis:



$$LG = H_{open} \times \frac{1}{M}$$

$$= \frac{1}{M} \times \left(\frac{I_{cp}}{2\pi} \right) \times \left(\frac{1}{C_1 C_2 R_1} \frac{1 + s C_1 R_1}{s^2 + \left(\frac{C_1 + C_2}{C_1 C_2 R_1} \right) s} \right) \times \frac{K_{vco}}{s}$$

$$= \frac{I_{cp} K_{vco}}{2\pi M C_1 C_2 R_1} \frac{1}{s^2} \frac{1 + s C_1 R_1}{s + \left[\frac{C_1 + C_2}{C_1 C_2 R_1} \right]}$$

open loop gain = $\frac{\text{Const 1} + s C_1 R_1 \text{ Const 1}}{s^3 + \left[\frac{C_1 + C_2}{C_1 C_2 R_1} \right] s^2}$ → Get PM from this TF

$$H_{closed} = \frac{H_{open}}{1 + (H_{open} \times \frac{1}{M})}$$

$$= \frac{M \left(\frac{\text{Const 1} + s C_1 R_1 \text{ Const 1}}{s^3 + \left[\frac{C_1 + C_2}{C_1 C_2 R_1} \right] s^2} \right)}{1 + \frac{\text{Const 1} + s C_1 R_1 \text{ Const 1}}{s^3 + \left[\frac{C_1 + C_2}{C_1 C_2 R_1} \right] s^2}}$$

closed loop gain = $\frac{M \text{ Const 1} + s C_1 R_1 M \text{ Const 1}}{s^3 + \left[\frac{C_1 + C_2}{C_1 C_2 R_1} \right] s^2 + s C_1 R_1 \text{ Const 1} + \text{Const 1}}$

→ Get BW from this TF

$$H_{Open \text{ Loop Gain}}(s) = \frac{I_{cp} K_{vco}}{2\pi N} \frac{s R_1 C_1 + 1}{s^2 (s R_1 C_1 C_2 + (C_1 + C_2))}$$

$$H_{Closed \text{ Loop Gain}}(s) = \frac{I_{cp} K_{vco}}{2\pi} \frac{s R_1 C_1 + 1}{s^3 R_1 C_1 C_2 + s^2 (C_1 + C_2) + s R_1 C_1 \frac{I_{cp} K_{vco}}{2\pi N} + \frac{I_{cp} K_{vco}}{2\pi N}}$$

3. System Stability (Matlab):

3.1 Bode Plots:

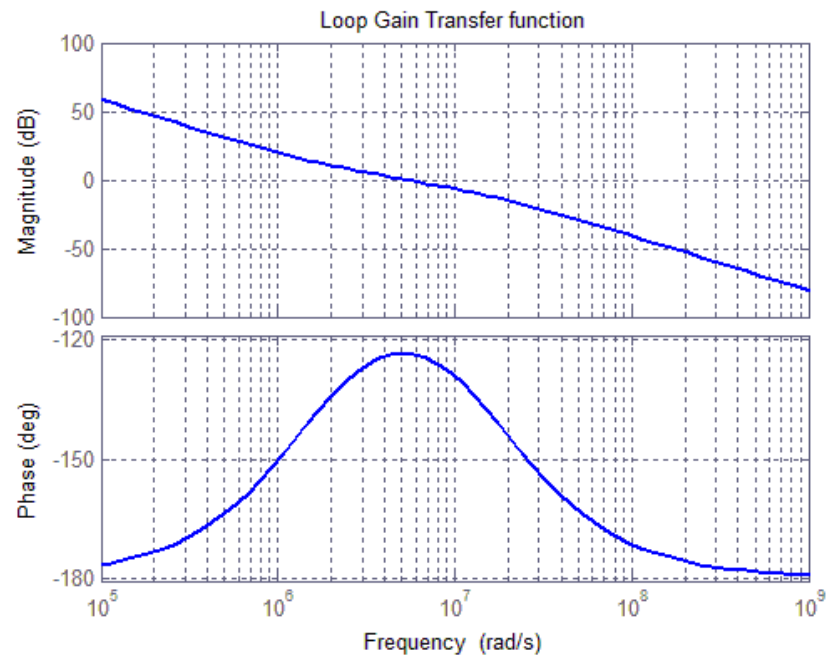


Fig. 3. Open-Loop Bode Plots

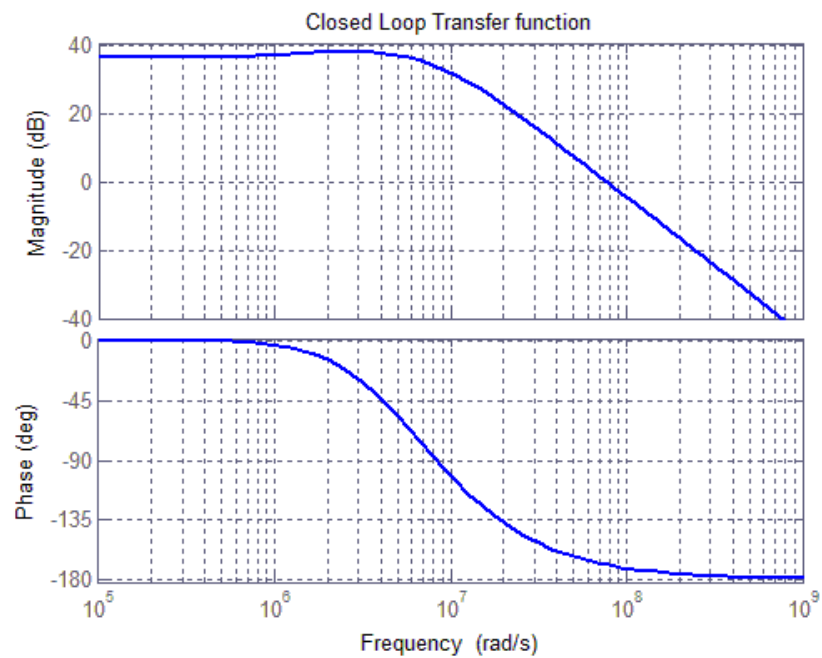


Fig. 4. Closed-Loop Bode Plots

Table 2. Open-Loop & Closed-Loop Parameters

Parameter		Value
Zero	f_z	0.245 MHz
Unity-Gain BW	f_{ugb}	0.871 MHz
Pole	f_{p3}	2.693 MHz
Max Phase Margin	PM_{Max}	56.44°
Phase Margin	PM	56.38°
Closed-Loop BW	BW	1.41 MHz

3.2 Code:

```

clc; close all;
figure
Icp = 100e-6;      Kvco = 2*pi*600e6;      M = 64;
R1 = 6.5e3;        C1 = 100e-12;          C2 = C1/10;
%-----
% Open Loop Gain:
wz = 1 / (R1*C1)
w_ugb = wz * sqrt((C1/C2)+1);
wp3 = (C1 + C2) / (R1*C1*C2);
PM_Max = (atan(w_ugb/wz) - atan(w_ugb/wp3)) * 360 / (2*pi)
Const1 = (Icp*Kvco) / (2*pi*M*C1*C2*R1);

H = tf([C1*R1*Const1 Const1],[1 ((C1+C2)/(C1*C2*R1)) 0 0]);
bode(H)
[Gm,Pm,Wgm,Wpm] = margin(H);
Pm
hold on; grid on; title('Loop Gain Transfer function')
set(findall(gcf,'type','line'),'linewidth',2)
%-----
% To find the zero & pole accurately:
b = [C1*R1*Const1 Const1];
a = [1 ((C1+C2)/(C1*C2*R1)) 0 0];
fvtool(b,a,'polezero')
[b,a] = eqtflength(b,a);
[z,p,k] = tf2zp(b,a)           %<-- z = zero, p = pole

gain = db2mag(0);
wc = getGainCrossover(H,gain)   %<-- wc = unity-gain BW
%-----
% Closed Loop Gain:
figure
H = tf([(C1*R1*Const1*M) (M*Const1)],[1 ((C1+C2)/(C1*C2*R1)) (C1*R1*Const1) Const1]);
bode(H)
[Gm,Pm,Wgm,Wpm] = margin(H);

BW = (bandwidth(H))/(2*pi)
fz = wz / (2*pi)
fp = wp3 / (2*pi)
fc = wc / (2*pi)
hold on; grid on; title('Closed Loop Transfer function')
set(findall(gcf,'type','line'),'linewidth',2)

```

4. System Simulations:

4.1 Open-Loop Test Bench:

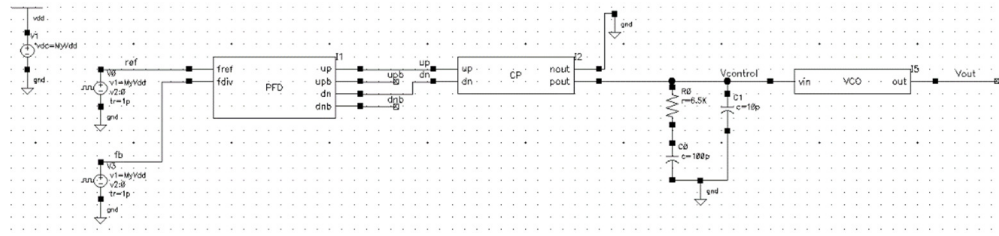


Fig. 5. Test Bench for PFD, CP, LF, & VCO in an Open Loop

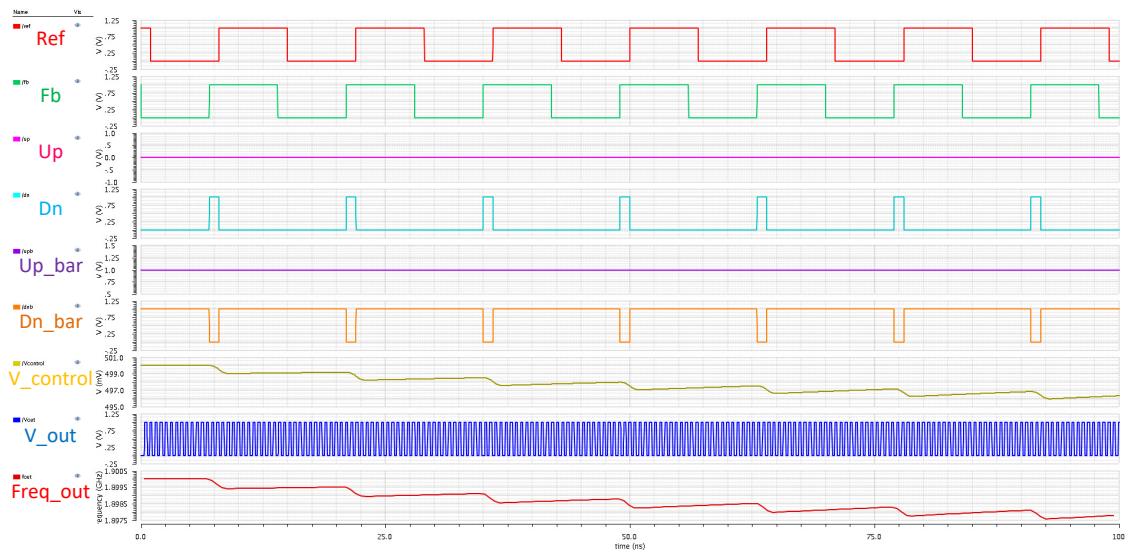


Fig. 6. Waveforms when Feedback signal is leading (with respect to Reference)

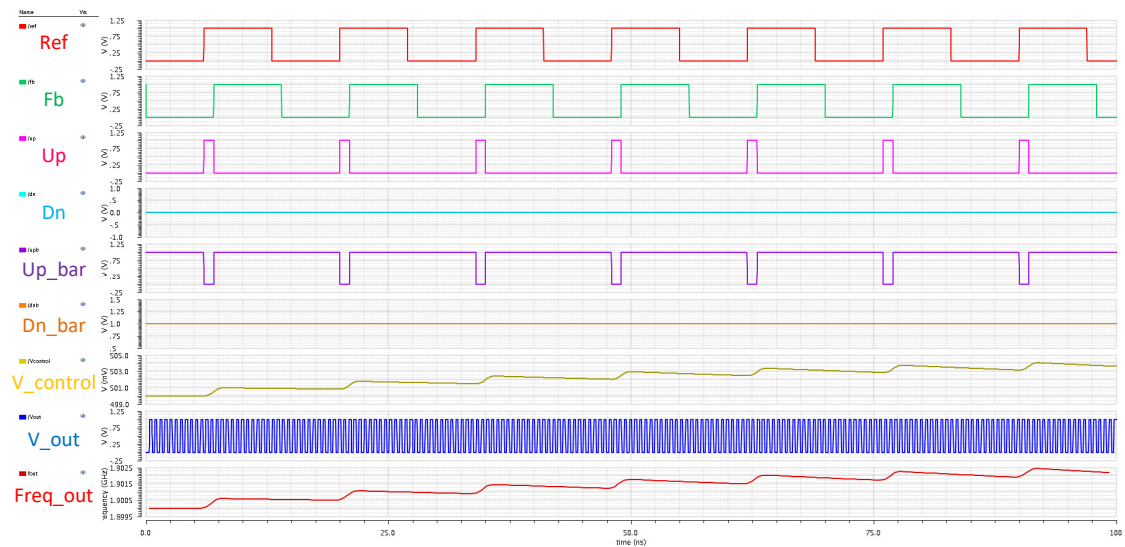


Fig. 7. Waveforms when Feedback signal is lagging (with respect to Reference)

4.2 VCO & Divider Test Bench:

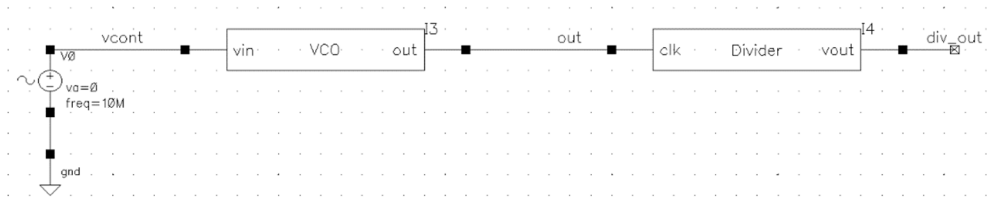


Fig. 8. Test Bench for VCO & Divider

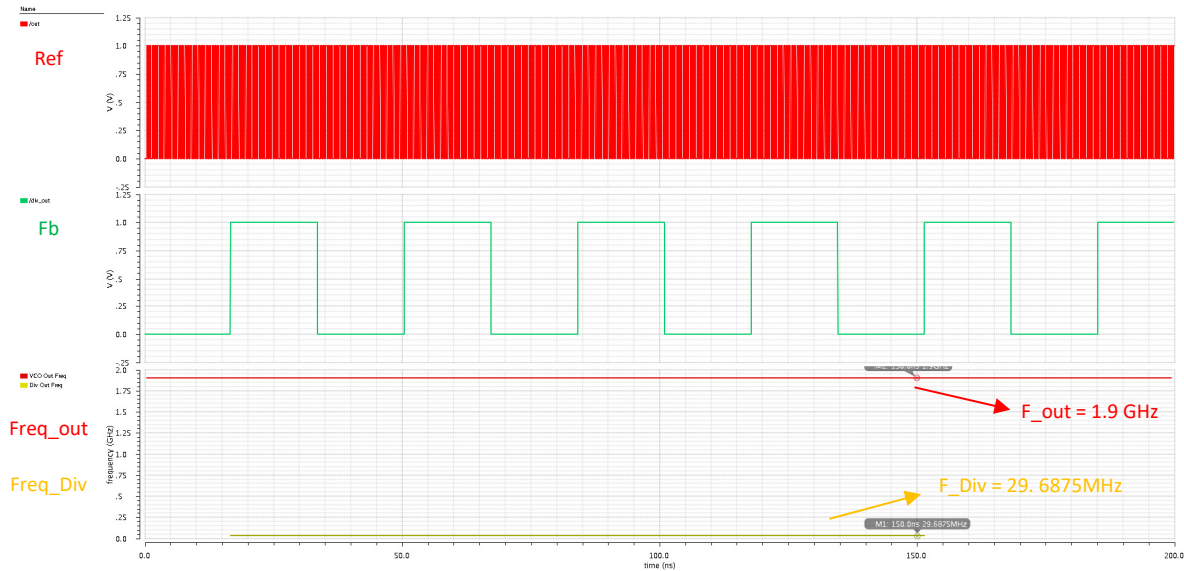


Fig. 9. Output Waveforms of VCO & Divider

4.3 Whole PLL Test Bench:

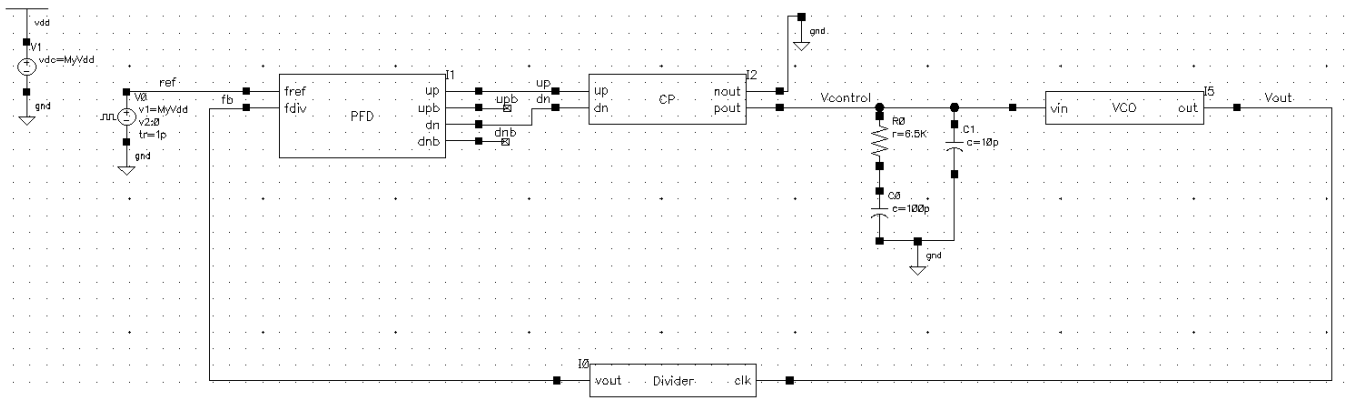


Fig. 10. Test Bench for the Whole PLL System

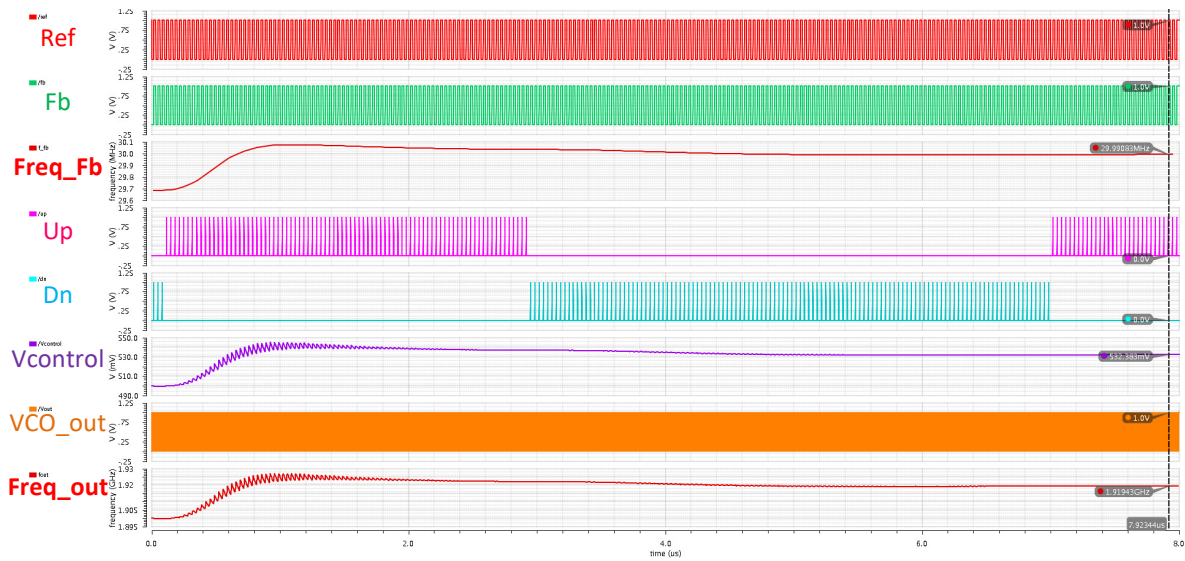


Fig. 11. Output Waveforms of the PLL when locking

5. VerilogA codes:

5.1 PFD code:

```
// VerilogA for EE230_PLL_VerilogA, PFD, verilogA
```

```
`include "constants.vams"
```

```
`include "disciplines.vams"
```

```
module PFD(up,dn,upb,dnb,fref,fdiv);
```

```
parameter real vtrans=0.5;
```

```
parameter real delay = 0;
```

```
parameter real ttime = 1p;
```

```
parameter real clk_threshold = 0.5;
```

```
input fref;
```

```
input fdiv;
```

```
output up,upb,dn,dnb;
```

```
electrical fref,fdiv,up,upb,dn,dnb;
```

```
real fv_rst, fr_rst, reset, upr,upbr,dnr,dnbr;
```

```
analog begin
```

```
    @(cross(V(fref) - clk_threshold, +1))
```

```
    begin
```

```
        upr = 1; upbr = 0; end
```

```
    @(cross(V(fdiv) - clk_threshold, +1))
```

```
    begin
```

```
        dnr = 1; dnbr = 0; end
```



```

        if(dnr == upr) begin
            upr = 0; dnr = 0; upbr = 1; dnbr = 1; end

        V(up) <+ transition(upr,delay,ttime);      V(dn) <+ transition(dnr,delay,ttime);
        V(upb) <+ transition(upbr,delay,ttime);    V(dnb) <+ transition(dnbr,delay,ttime);
    end
endmodule

```

5.2 Charge Pump code:

```

// VerilogA for EE230_PLL_VerilogA, CP, verilogA
`include "constants.vams"
`include "disciplines.vams"

module CP(pout,nout,up,dn);
    parameter real cur = 100u; //output current 100uA
    parameter real threshold=0.5;
    input up, dn;
    output pout, nout;
    electrical pout, nout, up, dn;
    real out;

    analog begin
        @(initial_step) out = 0.5;

        if((V(dn)>0.5) && (V(up)<0.5)) begin
            out = -cur; end
        else if((V(dn)<0.5) && (V(up)>0.5)) begin
            out = cur; end
        else out = 0.0;

        I(pout, nout) <+ -transition(out, 0.0, 10n, 10n);
    end
endmodule

```

5.3 VCO code:

```

// VerilogA for testlib, VCO, VerilogA
`include "constants.vams"
`include "disciplines.vams"

module VCO ( vin , out ) ;

    input vin ; output out ;
    electrical vin ; electrical out ;

```



```

parameter real Vmin=0; // Minimum input voltage
parameter real Vmax=Vmin+1 from (Vmin : inf) ; // Maximum input voltage
parameter real Fmin= 1.6e9 from ( 0 : inf) ; // Minimum output frequency
parameter real Fmax=2.2e9 from (Fmin : inf) ; // Maximum output frequency
parameter real Vamp = 1 from [ 0 : inf) ; // Output sinusoid amplitude
parameter real ttol=1u/Fmax from ( 0 : 1 /Fmax ) ; // Crossing time tolerance
parameter real vtol = 1e-9; // Voltage

parameter integer min_pts_update=32 from [ 2 : inf) ;

parameter real tran_time = 10e-12 from ( 0 : 0.3 /Fmax ) ;
parameter real jitter_std_ui = 0 from [ 0 : 1 ) ;

// Internal Variables
real freq, phase, jitter_rad, dPhase, phase_ideal;
integer n ;
integer seed ;

analog begin
    @( initial_step )
    begin
        seed = 671;
        n = 0 ;
        dPhase = 0 ;
        jitter_rad = jitter_std_ui * 2 * `M_PI ;
    end

    // compute the freq from the input voltage
    freq = ( (V(vin) - Vmin) * (Fmax - Fmin) / (Vmax - Vmin) ) + Fmin ;

    $bound_step (1/( min_pts_update * freq ) ) ;
    if ( freq > Fmax) freq = Fmax ;
    if ( freq < Fmin) freq = Fmin ;
    phase_ideal = 2 * `M_PI * idtmod(freq, 0.0, 1.0, -0.5);
    phase = phase_ideal + dPhase ;

    @( cross(phase_ideal + `M_PI/2, +1, ttol, vtol) or cross(phase_ideal - `M_PI/2, +1, ttol, vtol))
    begin
        dPhase = $rdist_normal(seed,0,jitter_rad); end

    @( cross(phase + `M_PI/2, +1, ttol, vtol) or cross(phase - `M_PI/2, +1, ttol, vtol))
    begin
        n = ( phase >= -`M_PI/2)&&(phase < `M_PI / 2 ) ; end

    // generate the output
    V( out ) <+ transition (n?Vamp:0 , 0 , tran_time);

```

```
end  
endmodule
```

5.4 Divider code:

```
// VerilogA for EE230_PLL_VerilogA, Divider, veriloga  
`include "constants.vams"  
`include "disciplines.vams"  
  
module Divider(clk,vout);  
  
parameter real vtrans=0.5;  
parameter real delay = 0;  
parameter real ttime = 1p;  
parameter real clk_threshold = 0.5;  
parameter real N = 64;  
  
input clk; output vout;  
electrical clk,vout;  
real counter,v_out;  
  
analog begin  
  
    @(initial_step) begin  
        v_out = 0; counter = 0; end  
  
    @(cross(V(clk) - clk_threshold, +1))  
    begin  
        if(counter < ((N/2)-1)) begin  
            counter = counter +1; v_out = 0; end  
        else if(counter < (N-1)) begin  
            counter = counter +1; v_out = 1; end  
        else if (counter == (N-1)) begin  
            counter = 0; v_out = 0; end  
    end  
  
    V(vout) <+ transition(v_out,delay,ttime);  
end  
endmodule
```