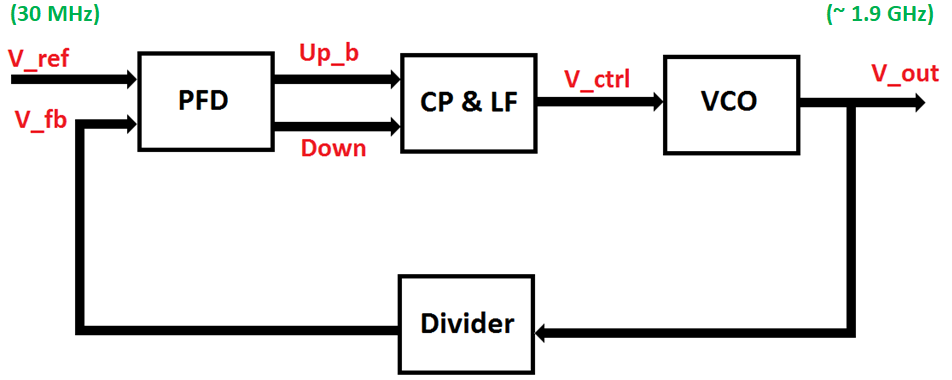
**EE230 – HW3 Report  
PLL System Design**  
(Using VerilogA & Matlab)

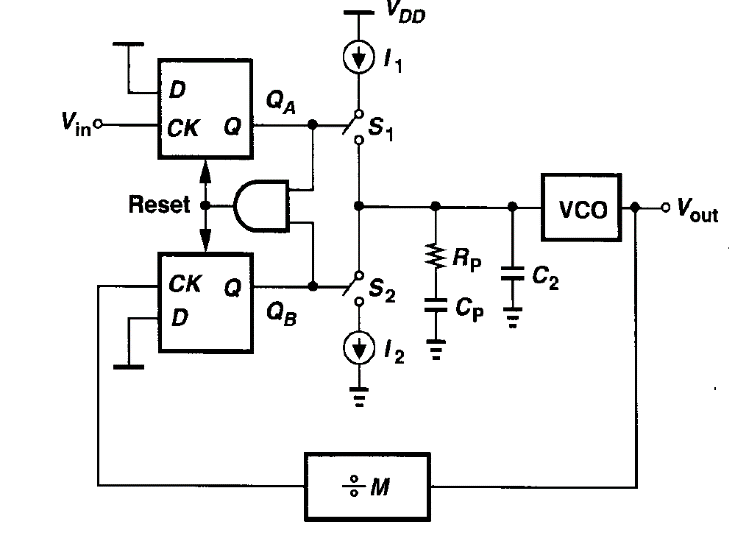
**Muhammad Aldacher**

**Student ID: 011510317**

1. **Schematic of the PLL system:**



**Fig. 1. PLL System Block Diagram**

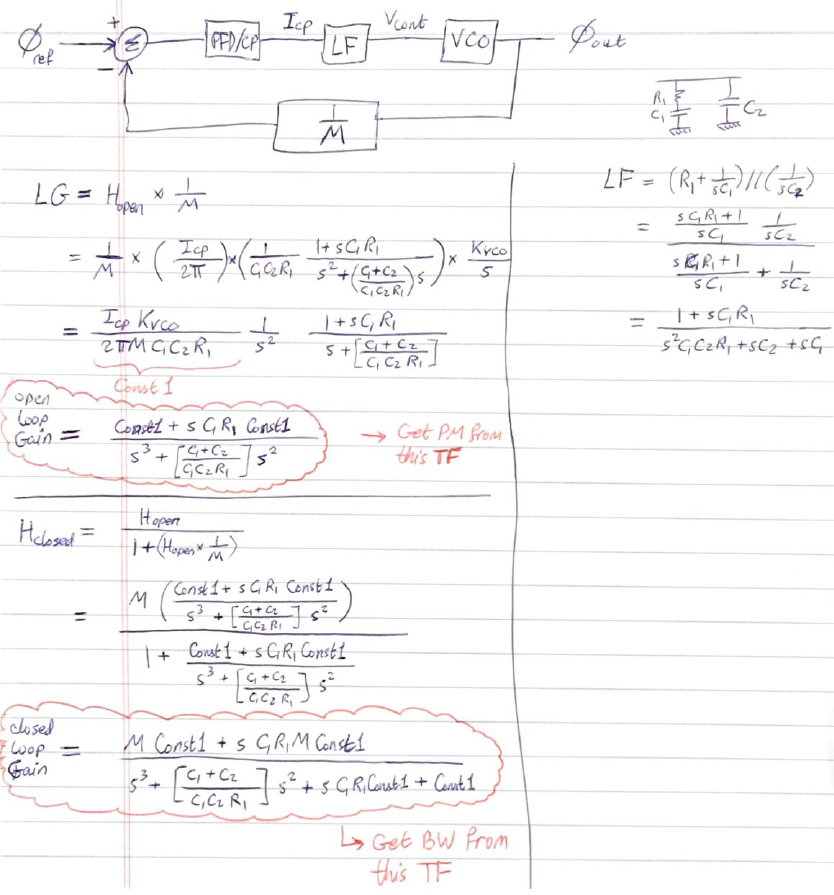
****

**Fig. 2. Typical Charge Pump PLL System**

Table 1. System Parameters Used

|  |  |
| --- | --- |
| Parameter | Value |
| FREF | 30 MHz |
| FOUT | 1.9 GHz |
| MDivider | 64 |
| ICP | 100 uA |
| KVCO | 600 MHz/V |
| RP | 6.5 KΩ |
| CP | 100 fF |
| C2 | 10 fF |

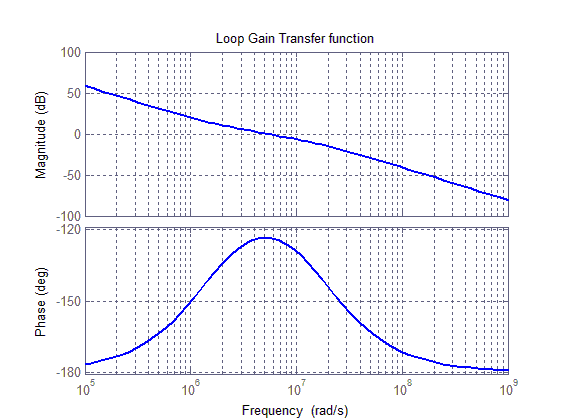
1. **System Analysis:**



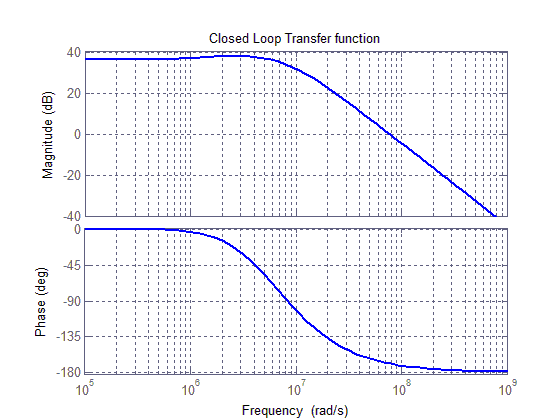
**=**

**=**

1. **System Stability (Matlab):**
   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 | fz | 0.245 MHz |
| Unity-Gain BW | fugb | 0.871 MHz |
| Pole | fp3 | 2.693 MHz |
| Max Phase Margin | PMMax | 56.44° |
| Phase Margin | PM | 56.38° |
| Closed-Loop BW | BW | 1.41 MHz |

* 1. **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)

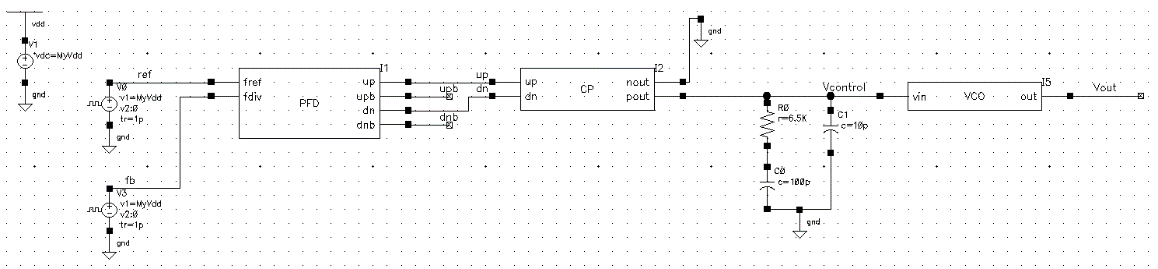
fp = wp3 / (2\*pi)

fc = wc / (2\*pi)

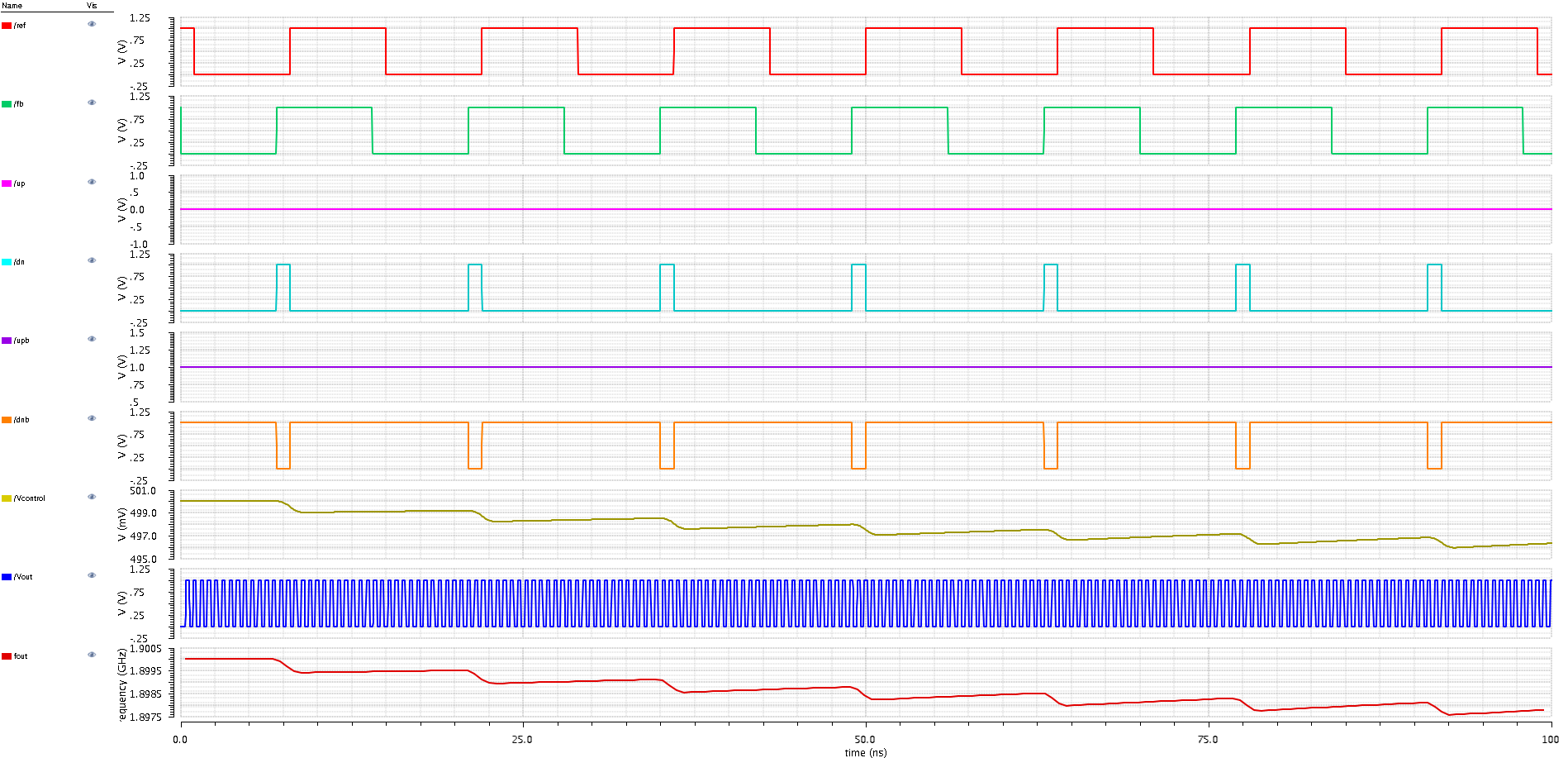
hold on; grid on; title('Closed Loop Transfer function')

set(findall(gcf,'type','line'),'linewidth',2)

1. **System Simulations:**
   1. **Open-Loop Test Bench:**



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



Ref

Fb

Up

Dn

Up\_bar

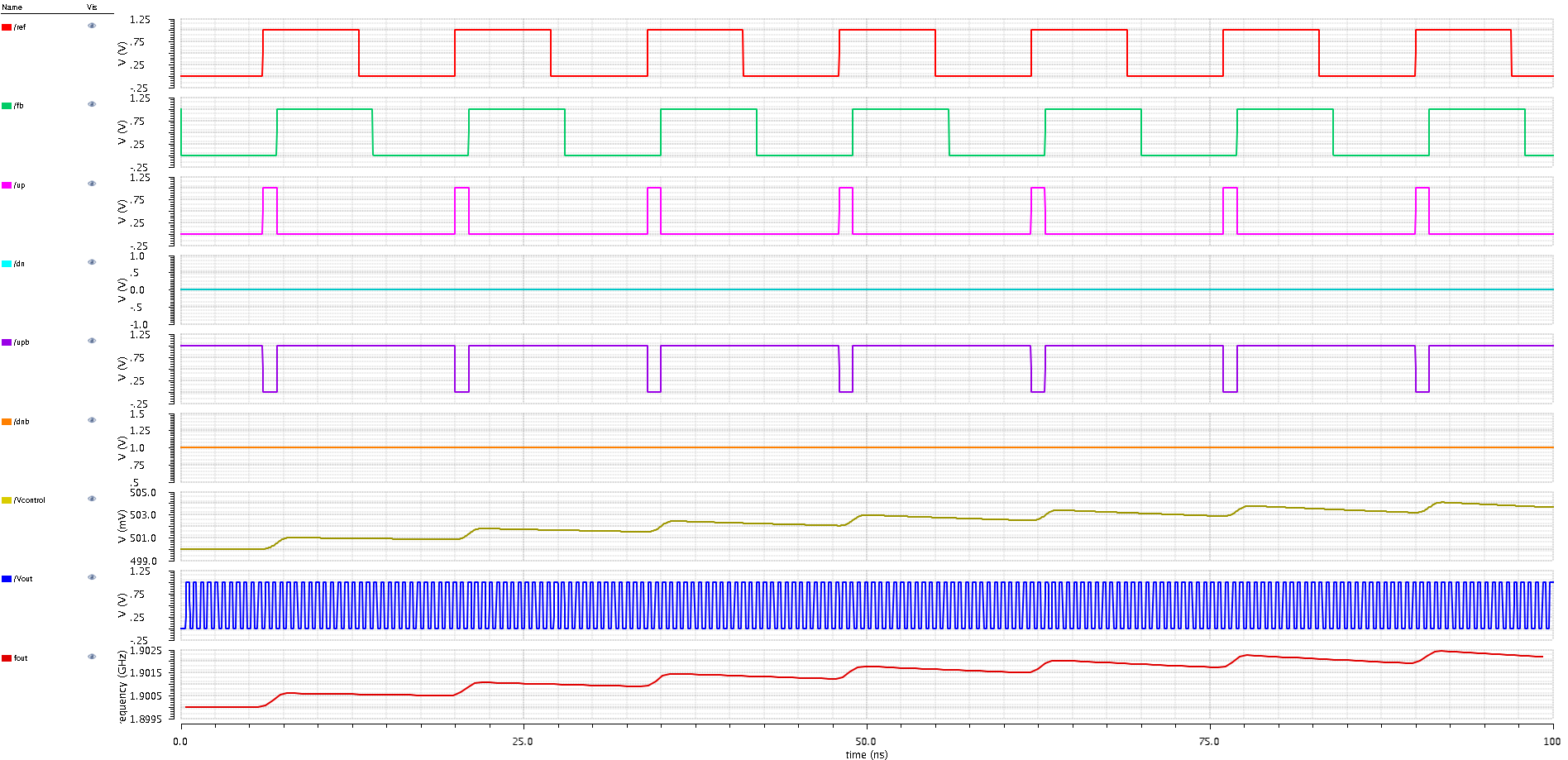
Dn\_bar

V\_control

V\_out

Freq\_out

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



Ref

Fb

Up

Dn

Up\_bar

Dn\_bar

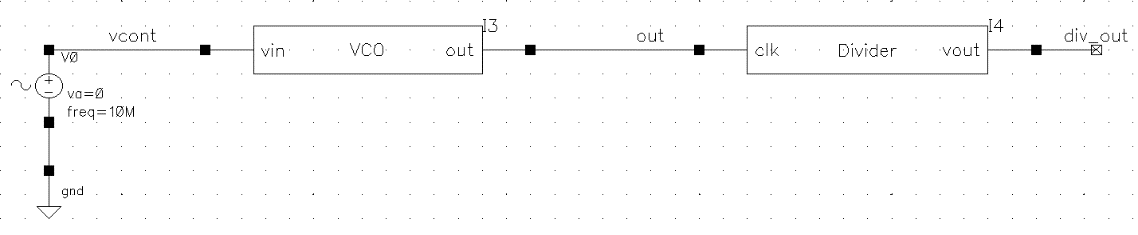
V\_control

V\_out

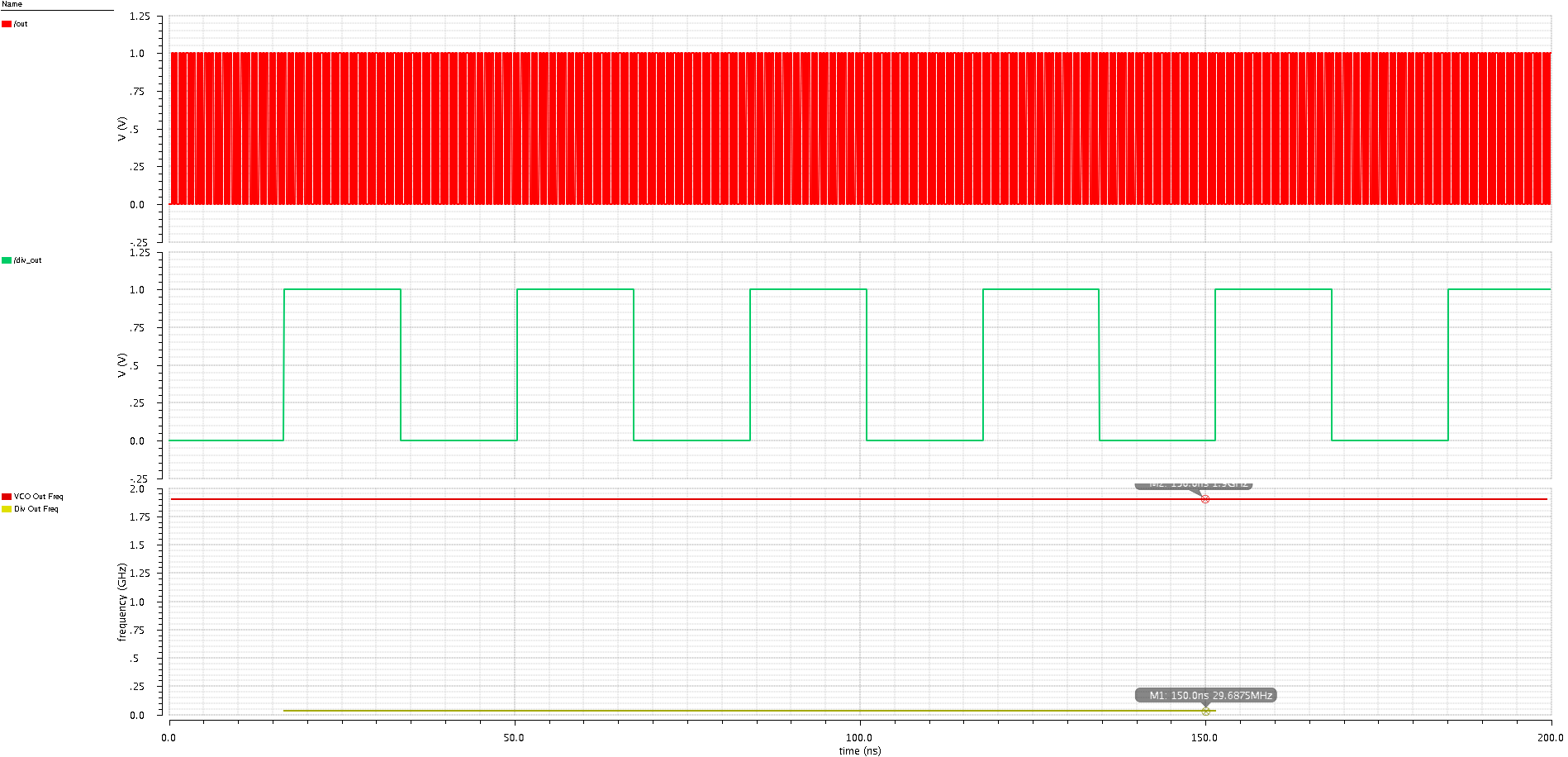
Freq\_out

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

* 1. **VCO & Divider Test Bench:**



**Fig. 8. Test Bench for VCO & Divider**



F\_Div = 29. 6875MHz

F\_out = 1.9 GHz

Ref

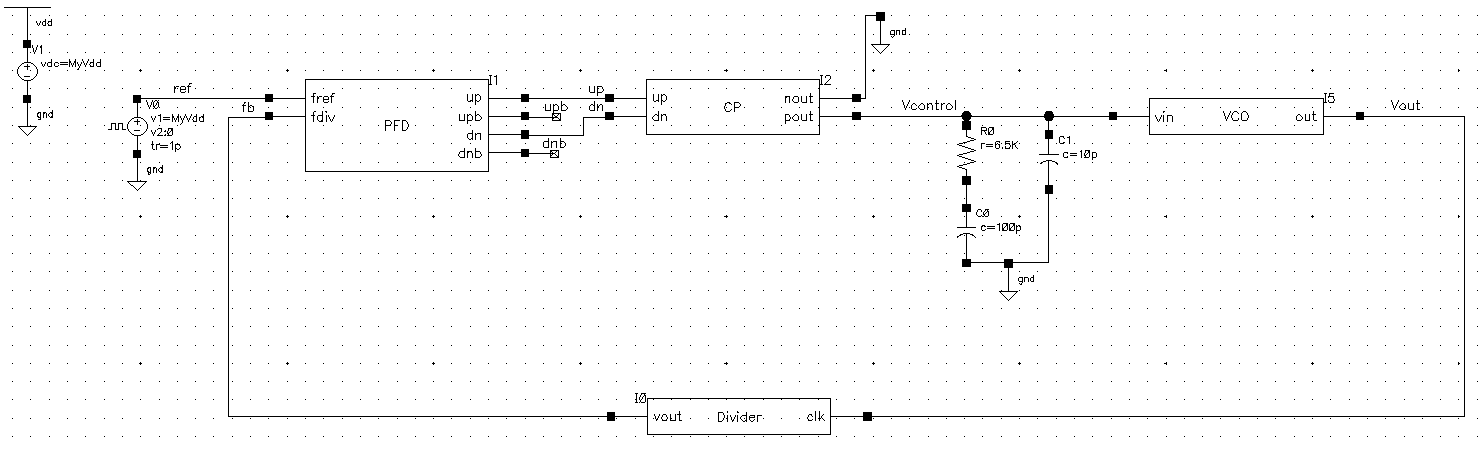
Fb

Freq\_out

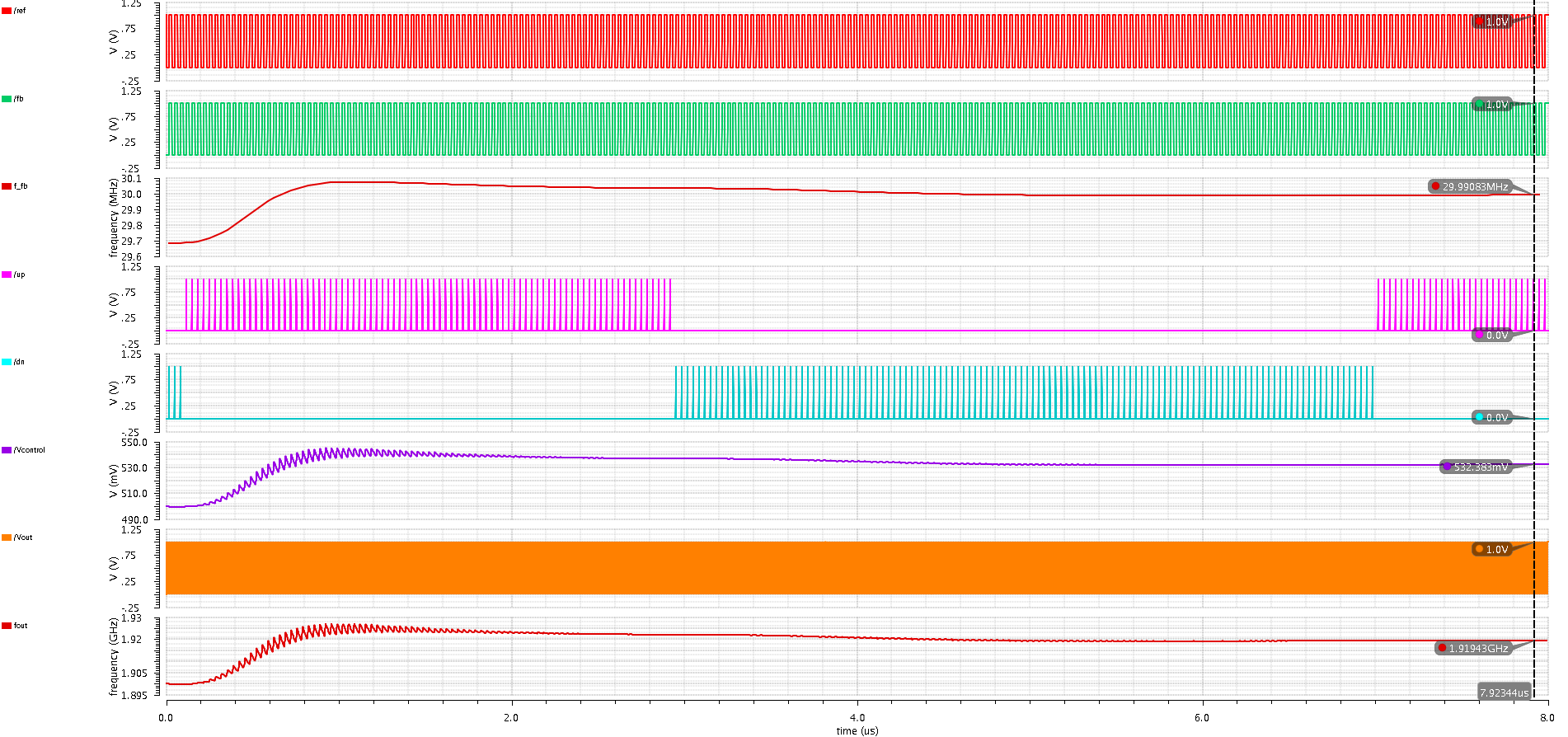
Freq\_Div

**Fig. 9. Output Waveforms of VCO & Divider**

* 1. **Whole PLL Test Bench:**



**Fig. 10. Test Bench for the Whole PLL System**



Ref

Fb

**Freq\_Fb**

Up

Dn

Vcontrol

VCO\_out

**Freq\_out**

**Fig. 11. Output Waveforms of the PLL when locking**

1. **VerilogA codes:**
   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

* 1. **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

* 1. **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

* 1. **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