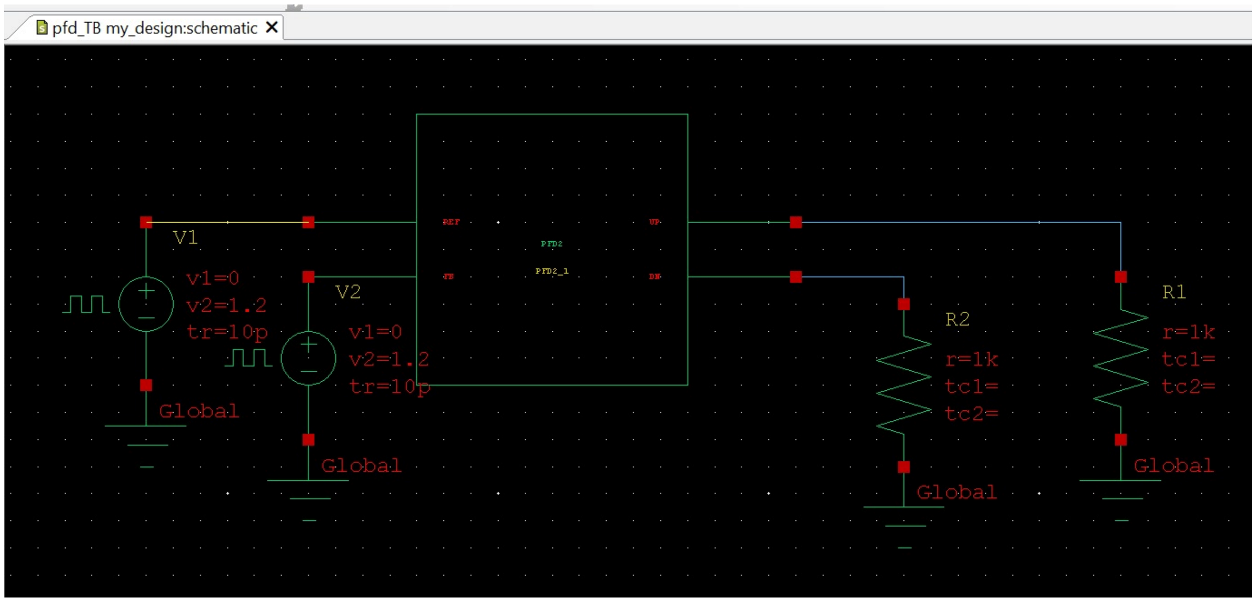
SIEMENS

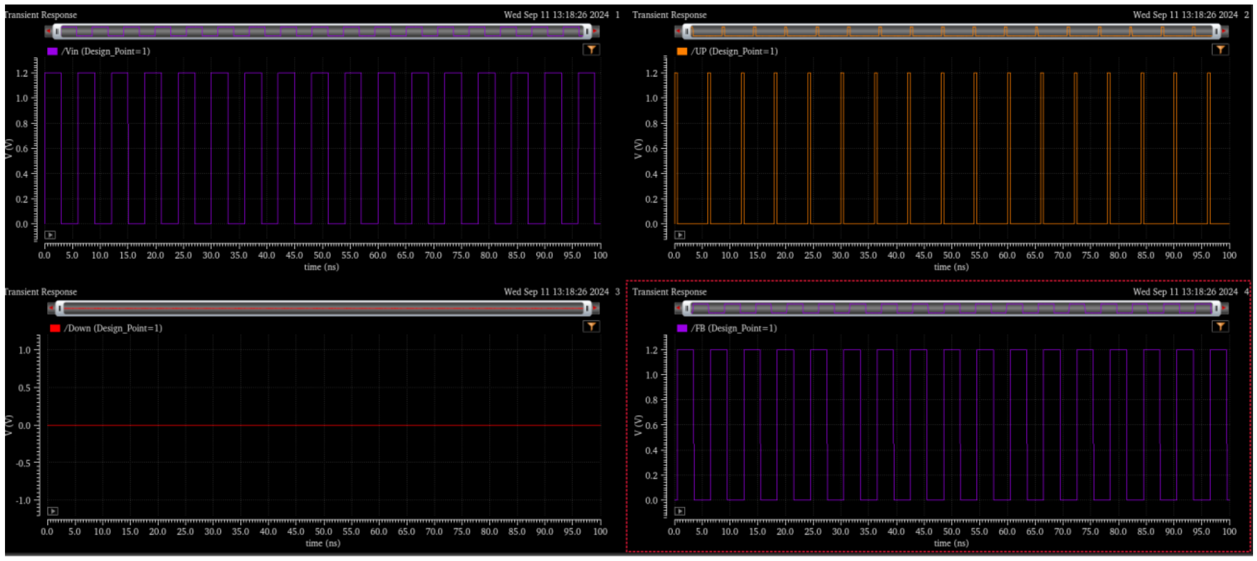
LAB-6

1. Your Verilog-A model for the PFD.

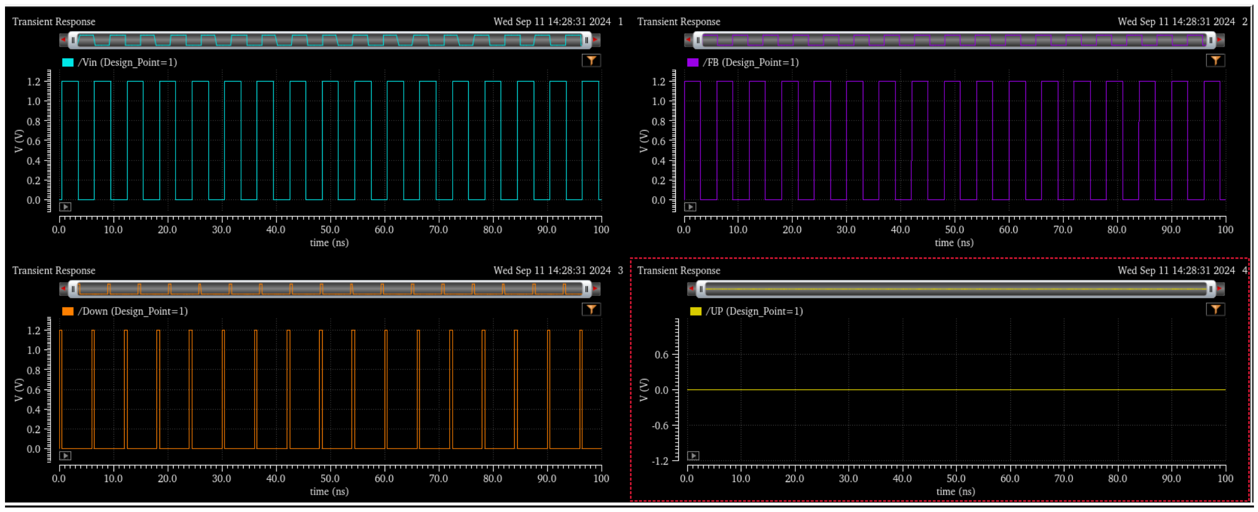
```
// AMS PLL Project: Phase Frequency Detector (PFD)

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

// REF: Reference signal
// FB: Feedback signal
// UP: Up signal (FB late)
// DN: Down signal (FB early)
module PFD2(REF,FB,UP,DN);
    // VDD and threshold voltage for digital signals
    parameter real VDD = 1.2;
    parameter real thresh = 0.6;
    // rise/fall/delay times of PFD output
    parameter real trise = 100p, tfall = 100p, td = 0;

    input REF,FB;
    output UP,DN;

    electrical REF,FB,UP,DN;

    // Internal UP and DN signals
    // * add line here *
    real DN_i,UP_i;
    analog begin

        // Check DN_i state when REF arrives
        // * add line here *
        @(cross(V(REF)-thresh,1))
            if(DN_i < thresh)
                UP_i=VDD;
            else begin
                UP_i = 0;
                DN_i = 0;
            end

        // Check UP_i state when FB arrives
        @(cross(V(FB)-thresh,1))
            // * add line here *
            if(UP_i < thresh)
                DN_i = VDD;
            else begin
        // * add line here *
        // * add line here *
        UP_i = 0;
        DN_i = 0;
      end

    V(UP) <+ transition(UP_i,td,trise,tfall);
    // * add line here *
    V(DN) <+ transition(DN_i,td,trise,tfall);
  end
endmodule
```

<u>2-Make a simple testbench to test the PFD. Deliver a snapshot clearly illustrating the PFD operation (similar to Fig. 6.13 and Fig. 6.14).</u>





The previous figure shows the affect when the Vin is faster than the Vrefernce (FB) the UP signal is high

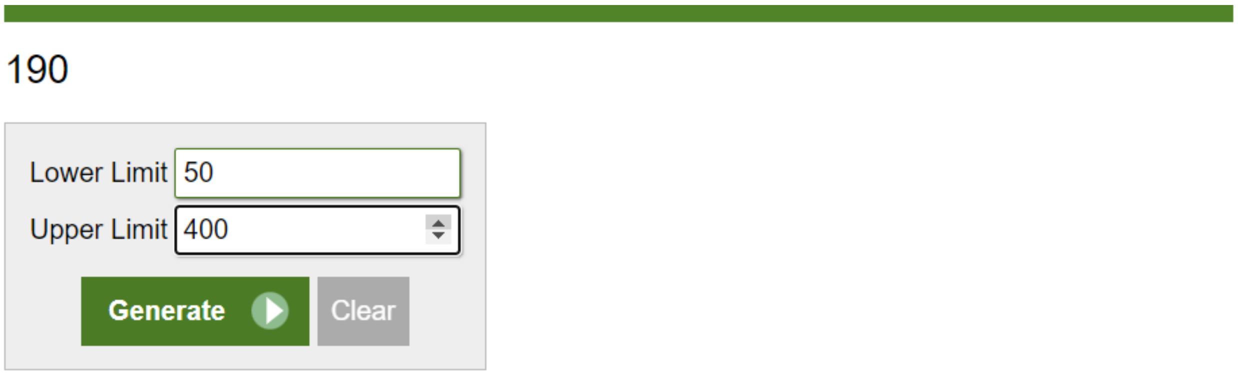


The previous figure shows the affect when the Vin is slower than the Vrefernce (FB) the DOWN signal is high

3-Your Verilog-A model for the CHP.

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

// UP: Up signal
// DN: Dn signal
// IOUT: CHP current output
// Since the output is current, IOUT cannot be left
unconnected (o.c.) in the testbench
module CHP(UP,DN,IOUT);
    input UP,DN;
    inout IOUT;
    electrical UP,DN,IOUT;

    // ichp: CHP current
    parameter real ichp = 10u from [0:inf);
    // Threshold voltage for digital signals
    parameter real thresh=0.6;
    // rise/fall/delay times of CHP output
    parameter real trise=100p, tfall=100p, td=0;

    // Internal variable for CHP output current
    real IOUT_i = 0;

    analog begin

        // Generate events at UP and DN transitions
        @(cross(V(UP)-thresh,0))
            ;
        // * add line here *
        @(cross(V(DN)-thresh,0))
           ;
        // * add line here *

        if ((V(UP) > thresh) && (V(DN) < thresh))
            IOUT_i = -ichp;
        // * add line here *
        else if ((V(UP) < thresh) && (V(DN) > thresh))
            IOUT_i = ichp;
        else
            // * add line here *
            IOUT_i = 0;
        I(IOUT) <+ transition(IOUT_i,td,trise,tfall);

    end
endmodule
```

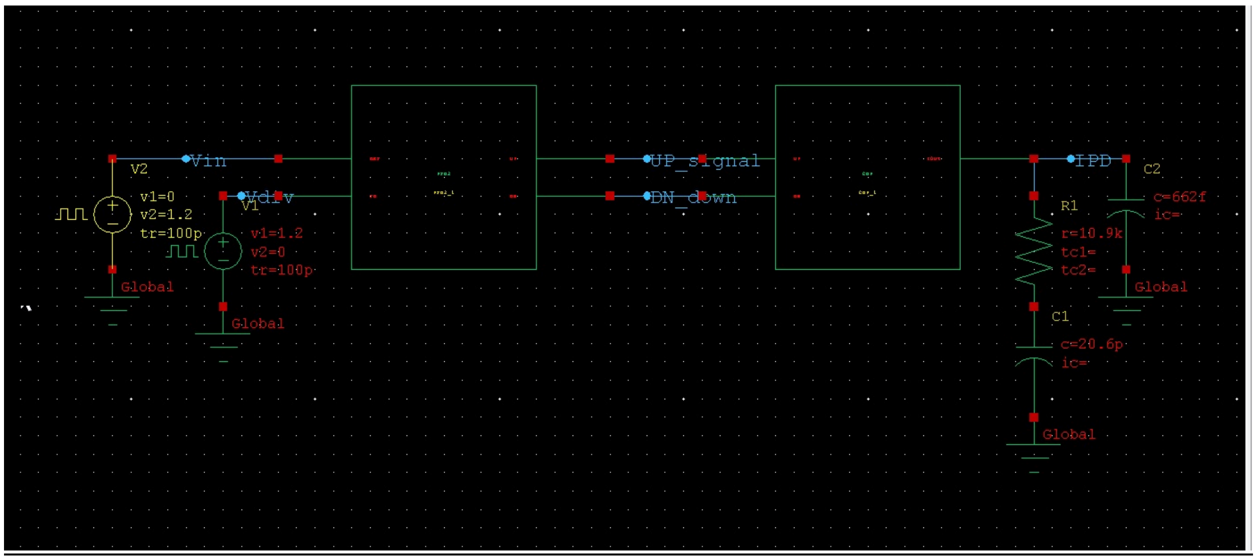
4-Make a simple testbench to test the CHP. Deliver a snapshot clearly illustrating the CHP operation (similar to Fig. 6.18).

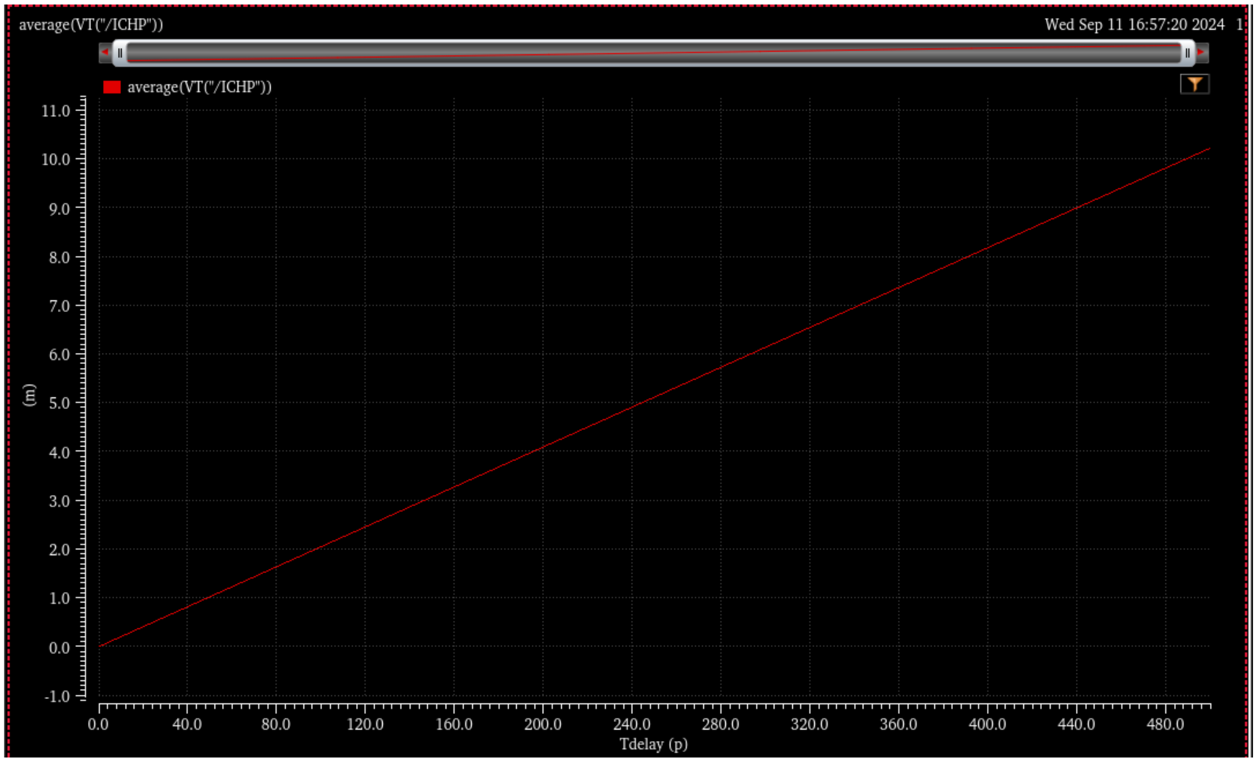


190

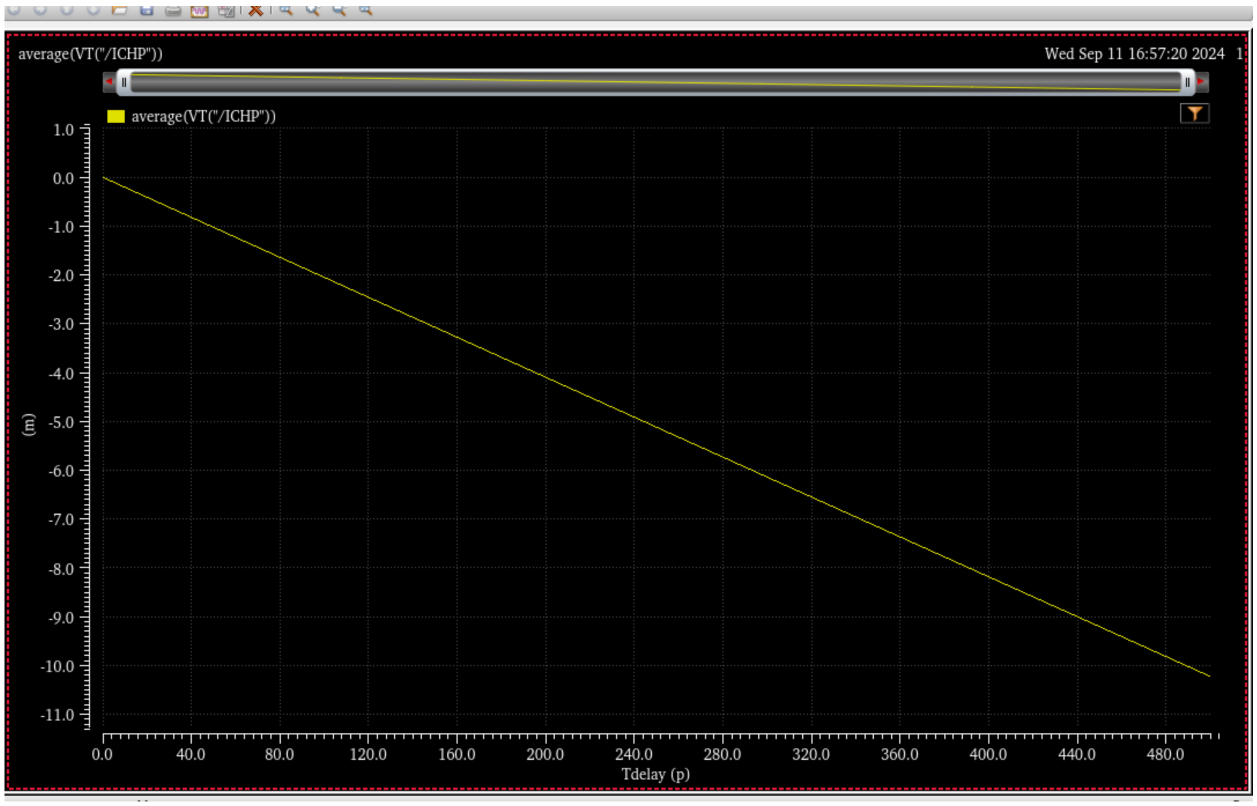
Lower Limit 50

Upper Limit 400





the Vin is faster than VFB so the output current for the charge pump is positive



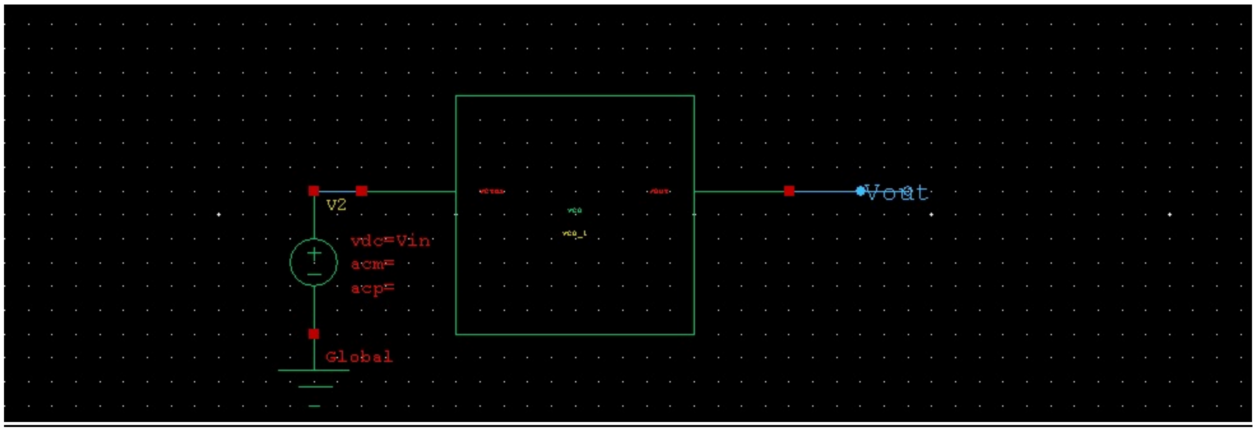
the Vin is slower than VFB so the output current for the charge pump is negative

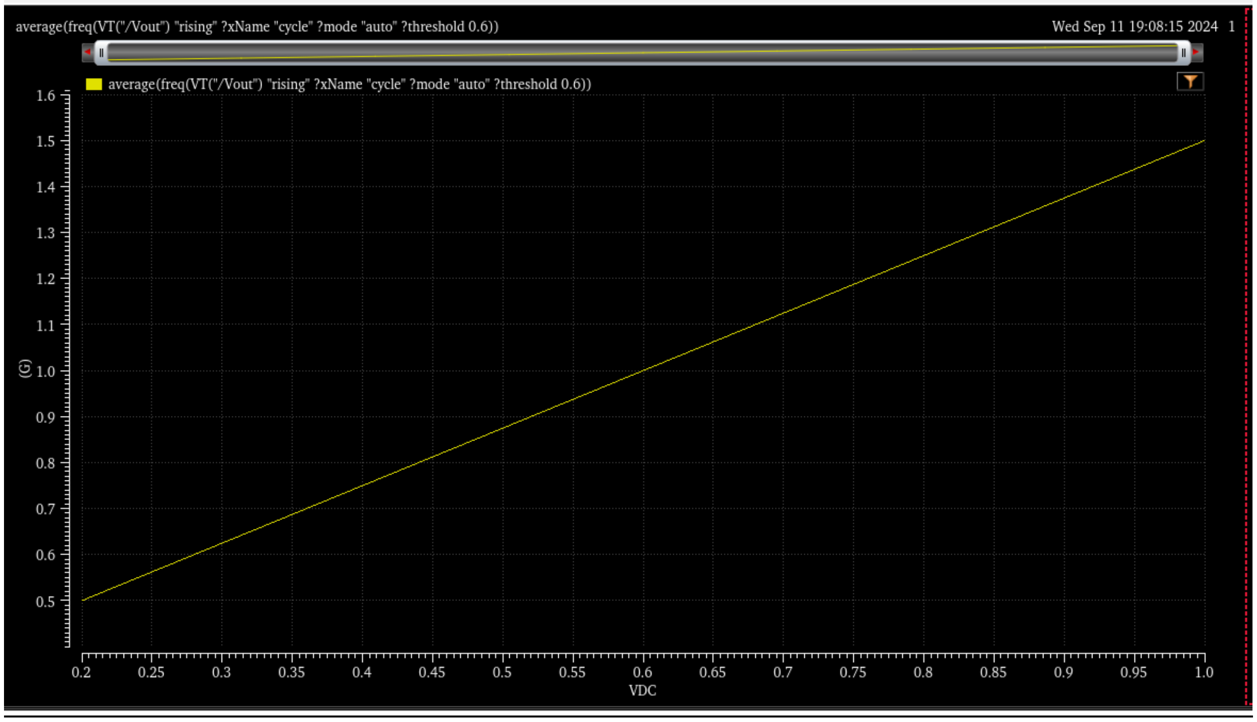
5-Your Verilog-A model for the VCO.

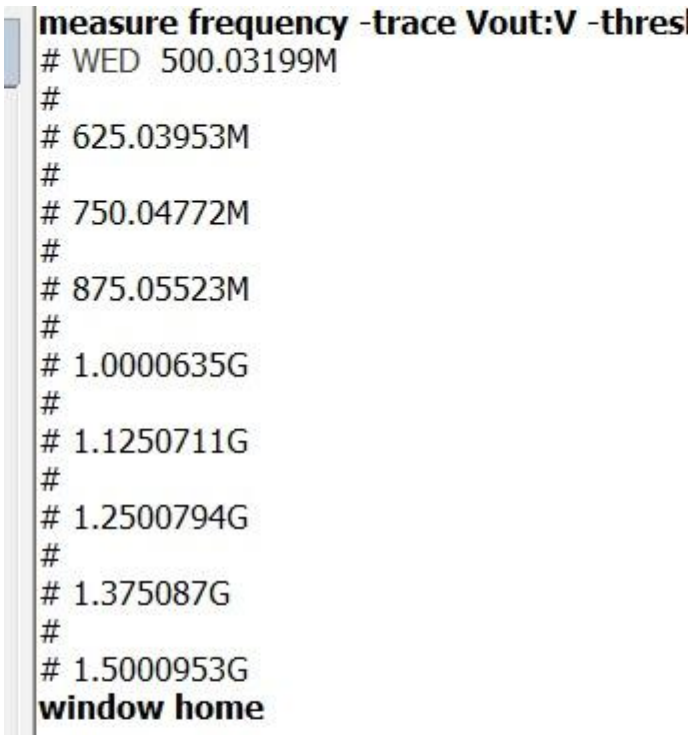
```
// AMS PLL Project: Voltage Controlled Oscillator (VCO)

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

module VCO(VCTRL,VOUT);

    parameter real VHIGH = 1.2;
    parameter real Vmin=0.2;
    parameter real Vmax=1 from (Vmin:inf);
    parameter real Fmin=0.5G from (0:inf);
    parameter real Fmax=2.02G from (Fmin:inf);
    parameter real trise=100p, tfall=100p, td=0;

    input VCTRL;
    output VOUT;
    voltage VCTRL,VOUT;


    // * add line here *
    real freq,phase,VOUT_i,sine;

    analog begin

        // compute the freq from the input voltage
        freq =((V(VCTRL) - Vmin)*(Fmax - Fmin) / (Vmax -
Vmin)) + Fmin;
        // bound the frequency
        // * add line here *
        if (freq > Fmax) freq = Fmax;
        if (freq < Fmin) freq = Fmin;

        // calculate the phase (modulo 2*pi)

        // * add line here *
        phase=2*`M_PI*idtmod(freq,0.0,1.0,-0.5);
        // generate the output
        sine = sin(phase);

        @(cross(sine,0))
            ;

        if (sine > 0)
```

```
                // * add line here *
                VOUT_i=VHIGH;

            else
                // * add line here *
                VOUT_i=0;

            V(VOUT) <+ transition(VOUT_i,td,trise,tfall);

            // bound the time step
            // * add line here *
            $bound_step(0.1/freq);

        end
    endmodule
```

6. Make a simple testbench to test the VCO. Deliver a snapshot clearly illustrating the VCO operation (similar to Fig. 6.11, but a single tuning curve is enough, i.e., no corner sim required). Add a marker to show the tuning voltage corresponding to the required output frequency.





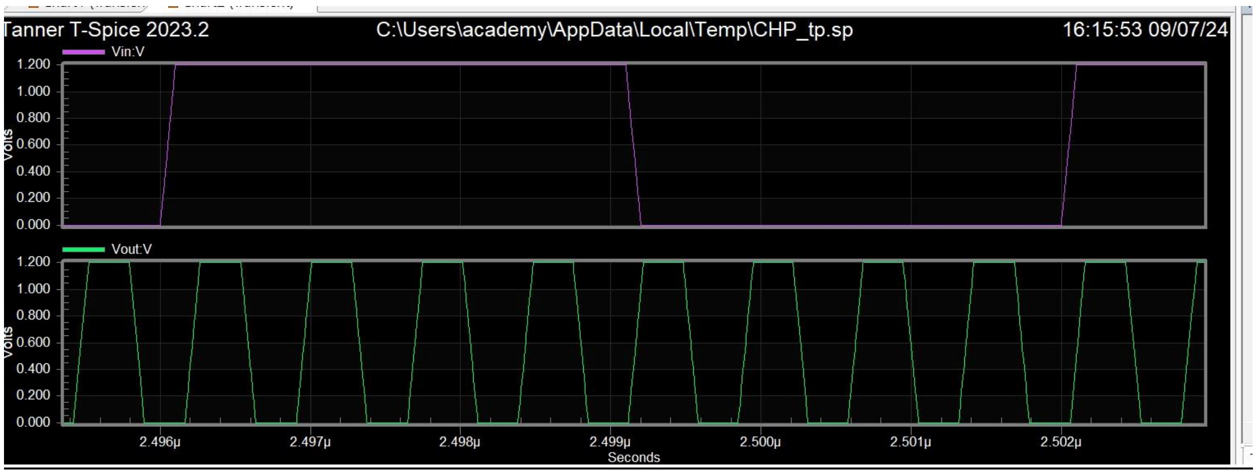


```
measure frequency -trace Vout:V -thres
# WED  500.03199M
#
# 625.03953M
#
# 750.04772M
#
# 875.05523M
#
# 1.0000635G
#
# 1.1250711G
#
# 1.2500794G
#
# 1.375087G
#
# 1.5000953G
window home
```

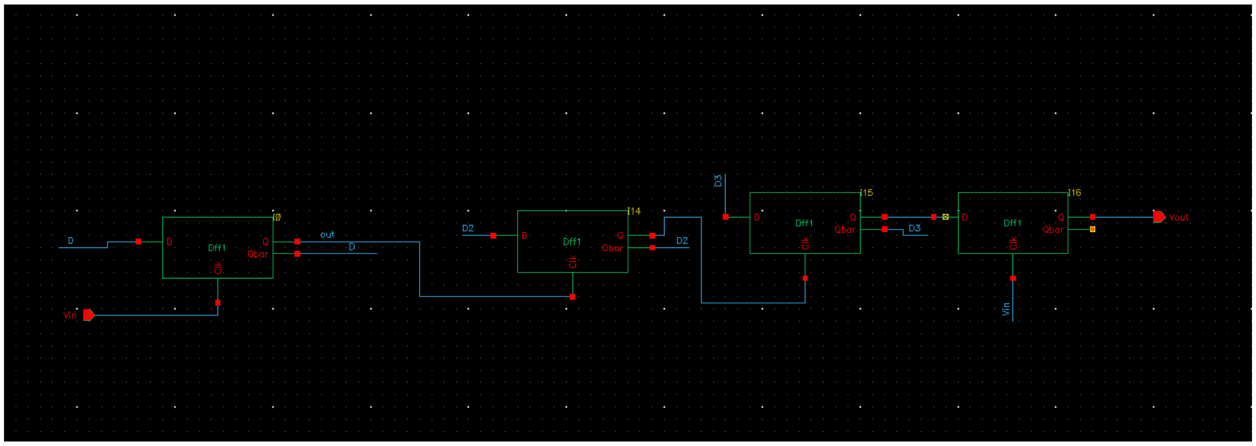
7. Your Verilog-A model for the divider.

```
// AMS PLL Project: Frequency Divider

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

module divider(VIN,VOUT);

    output VOUT; voltage VOUT;  // output
    input VIN; voltage VIN;     // input (edge triggered)
    parameter real vh=1.2;      // output voltage in high
state
    parameter real vl=0;        // output voltage in low
state
    parameter real vth=0.6; // threshold voltage at input
    parameter integer ratio=8 from [2:inf); // divider
ratio
    parameter real tt=10p from (0:inf); // transition time
of output signal
    parameter real td=0 from [0:inf);   // average delay
from input to output

    // *** add line here ***
    real out_value=0,count;

    analog begin

        @(cross(V(VIN) - vth, 1)) begin
            if (count==floor(ratio/2))  begin
                out_value = vh;
            // *** add line here ***
            end
            else if (count==floor(ratio))   begin
                out_value = vl;
                count=0;
                // *** add line here ***

            //else
                //count=count+1;
            end
            count = count + 1;
    end

       V(VOUT) <+ transition(out_value, td, tt);
    end
endmodule
```

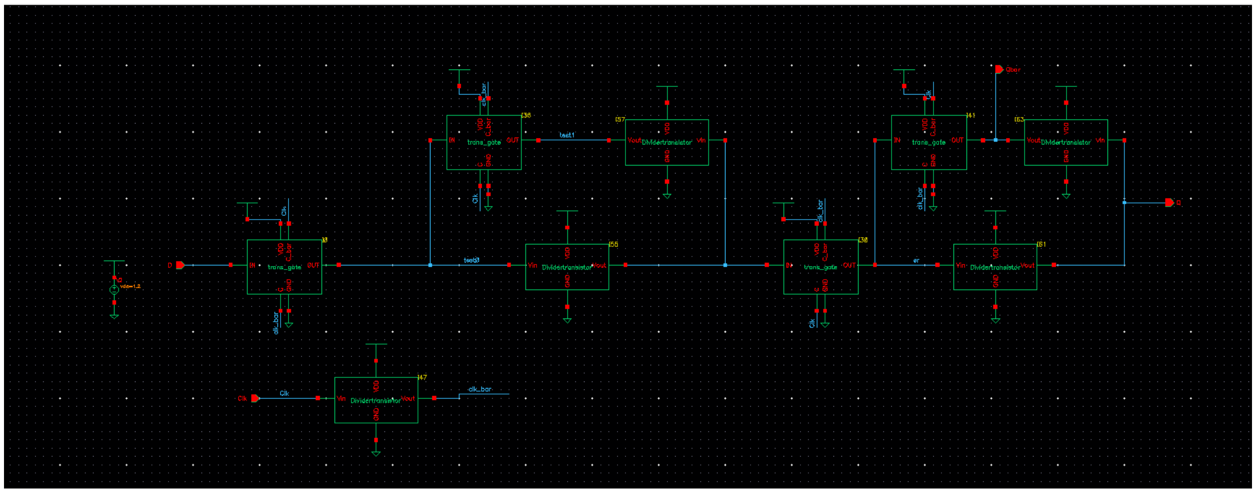
8. Make a simple testbench to test the divider. Deliver a snapshot clearly illustrating the divider operation (similar to Fig. 6.20).



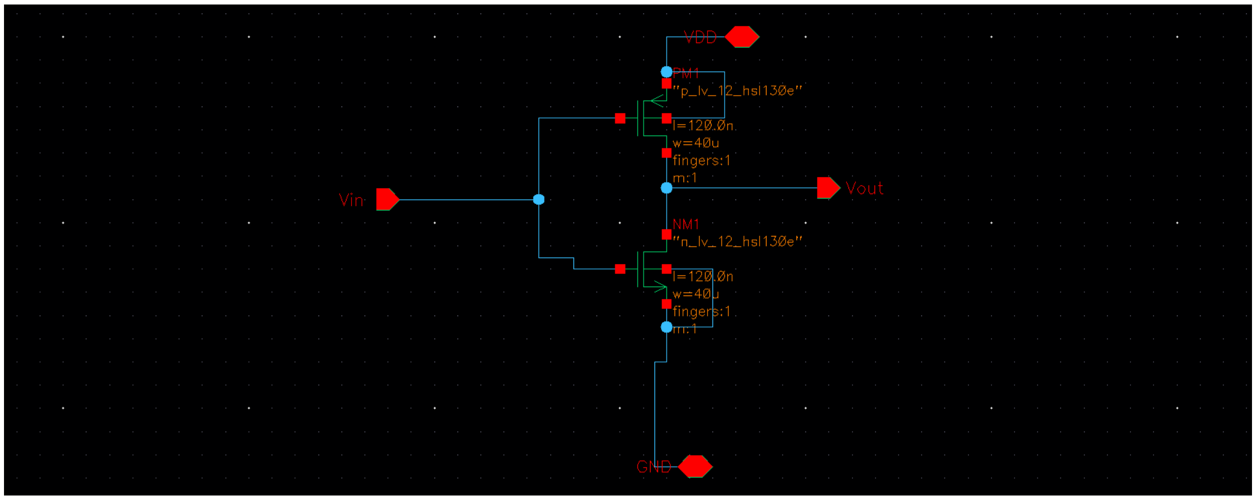
9. Design the divider at the transistor level. Deliver a snapshot clearly illustrating the transistor level schematic of the DFF that you used in the design.



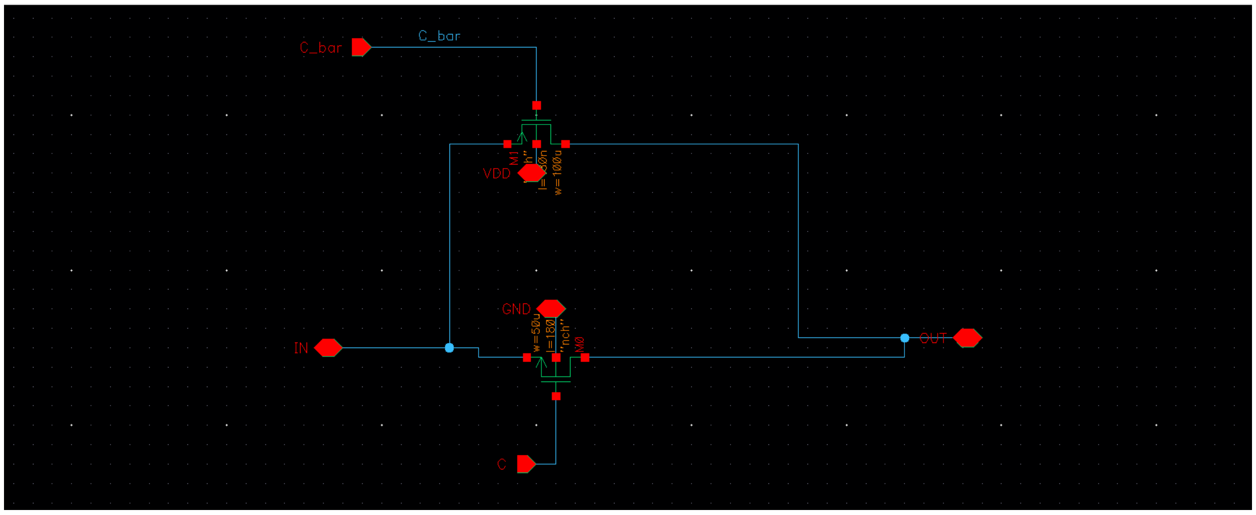
the divider schematic composed of 3 Dff to divide the frequency by 3 and one timing DFF to adjust the output with the clock.



The schematic of DFF

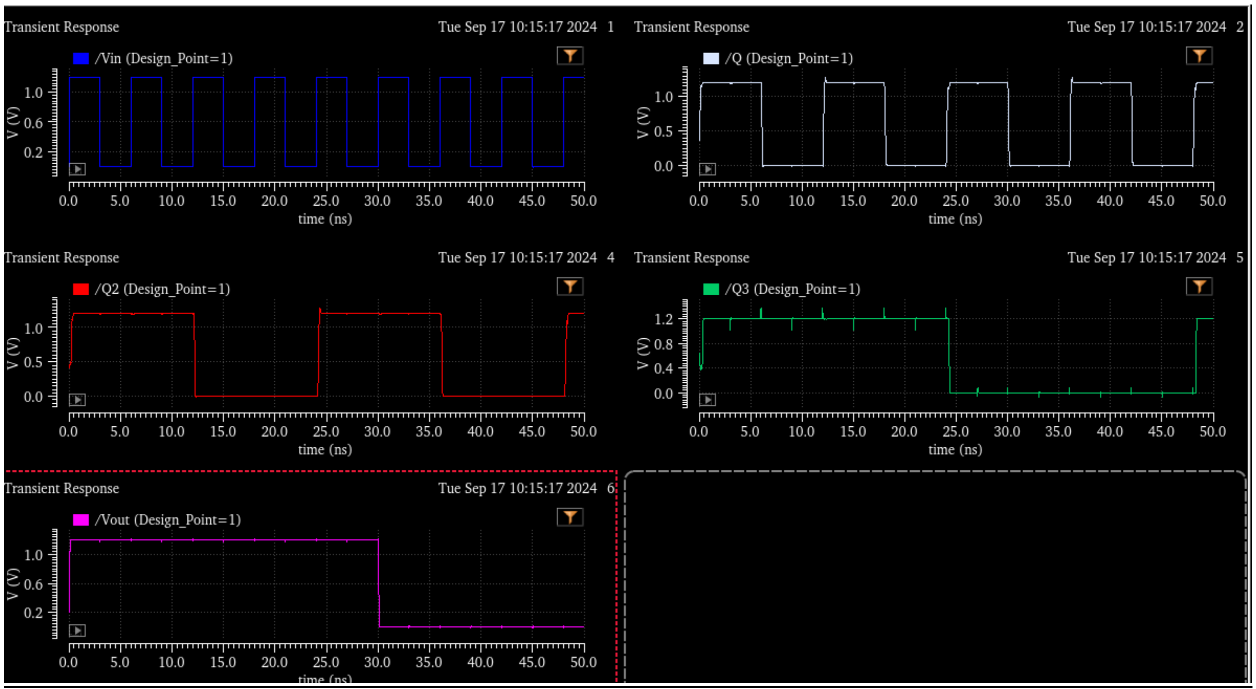


The inverter used in the DFF



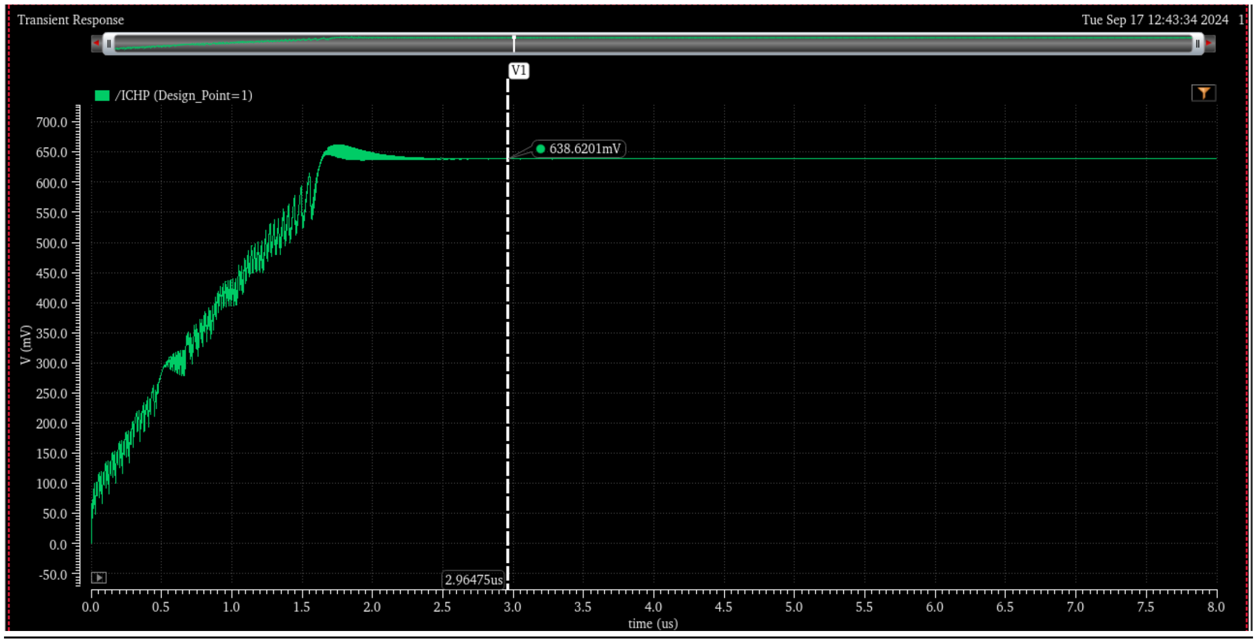
The transmission gate used in the DFF

<u>10. Using the same previous divider testbench, test your transistor level divider. Deliver a snapshot clearly illustrating the divider operation (similar to Fig. 6.20).</u>



<u>11. Integrate the PLL as shown in Fig. 6.2 using Verilog-A models only. Simulate the PLL using config view and hierarchy editor. Deliver a snapshot clearly illustrating the VCO control voltage (similar to Fig. 6.7). Add a cursor to show the value of the</u>

control voltage after lock.



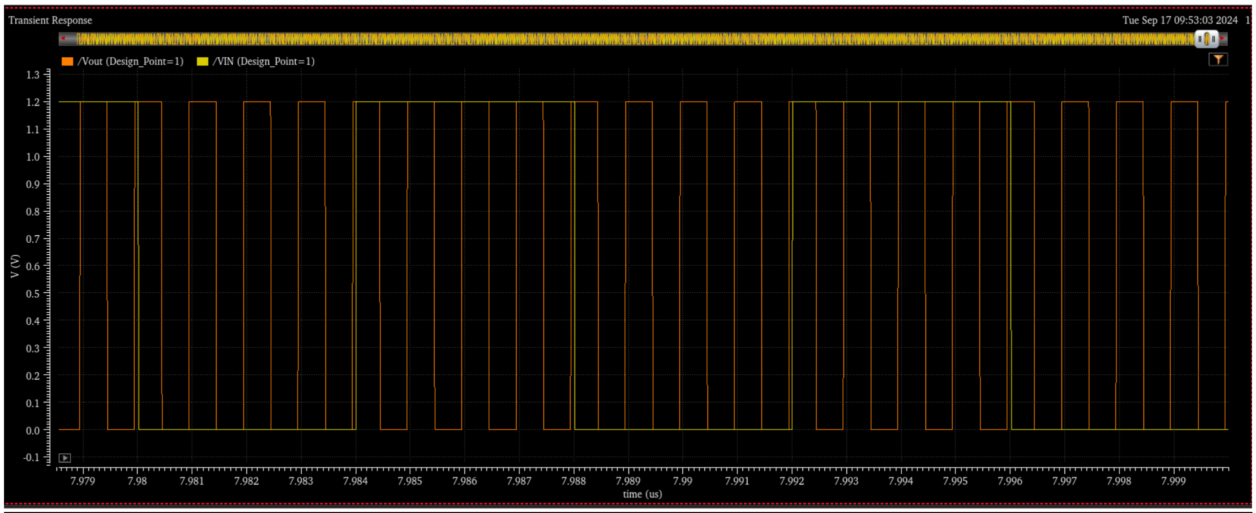
12. Given kvco, analytically calculate the control voltage at which the PLL should achieve lock.

Kvco=1.9Ghz and Fin= 166.7Mhz and Fout=8*166.7Mhz & fmin=0.5Ghz & Vmin=0.2$\rightarrow$ vctrl=(Fout-Fmin/Kvco)+0.2=0.638v

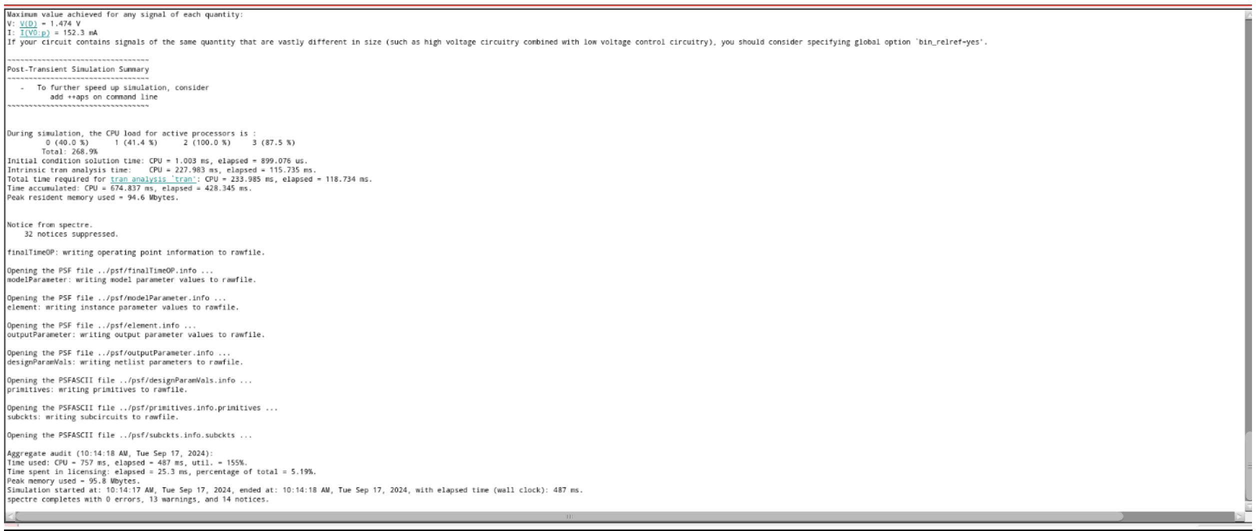
13. Compare the control voltage simulated value with the analytically calculated value. Comment.

They are equal.

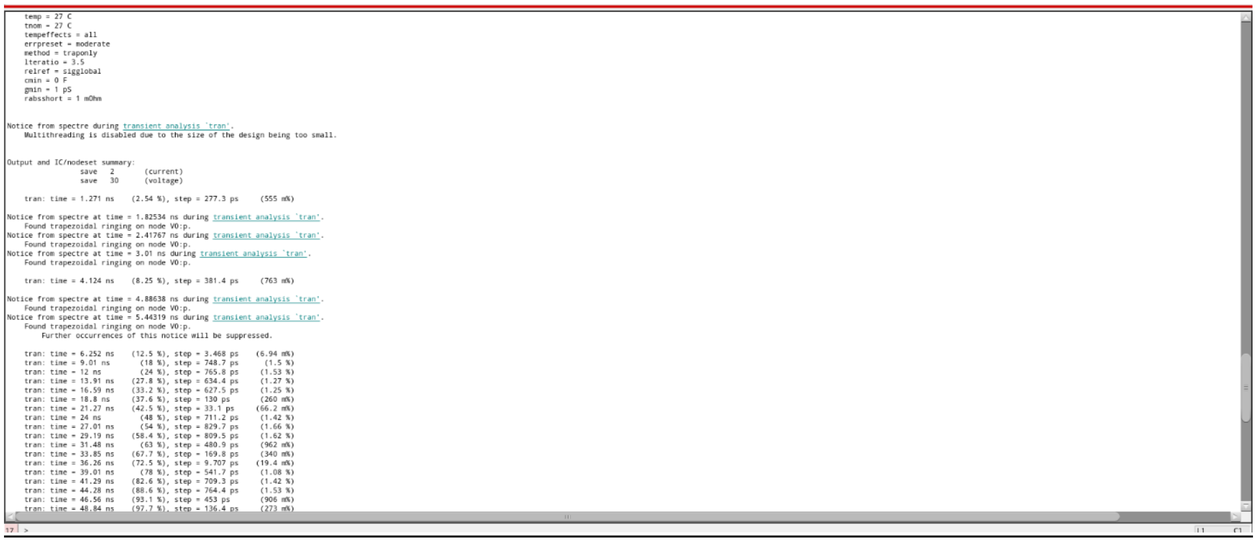
14. Integrate the PLL as shown in Fig. 6.2 using Verilog-A models only. Simulate the PLL using config view and hierarchy editor. Deliver a snapshot clearly illustrating the PLL operation (similar to Fig. 6.8). Add “A/B marker” to indicate the reference and output frequencies.



15. Deliver a snapshot of the log file of the previous simulation clearly illustrating the success of the simulation and the simulation time.



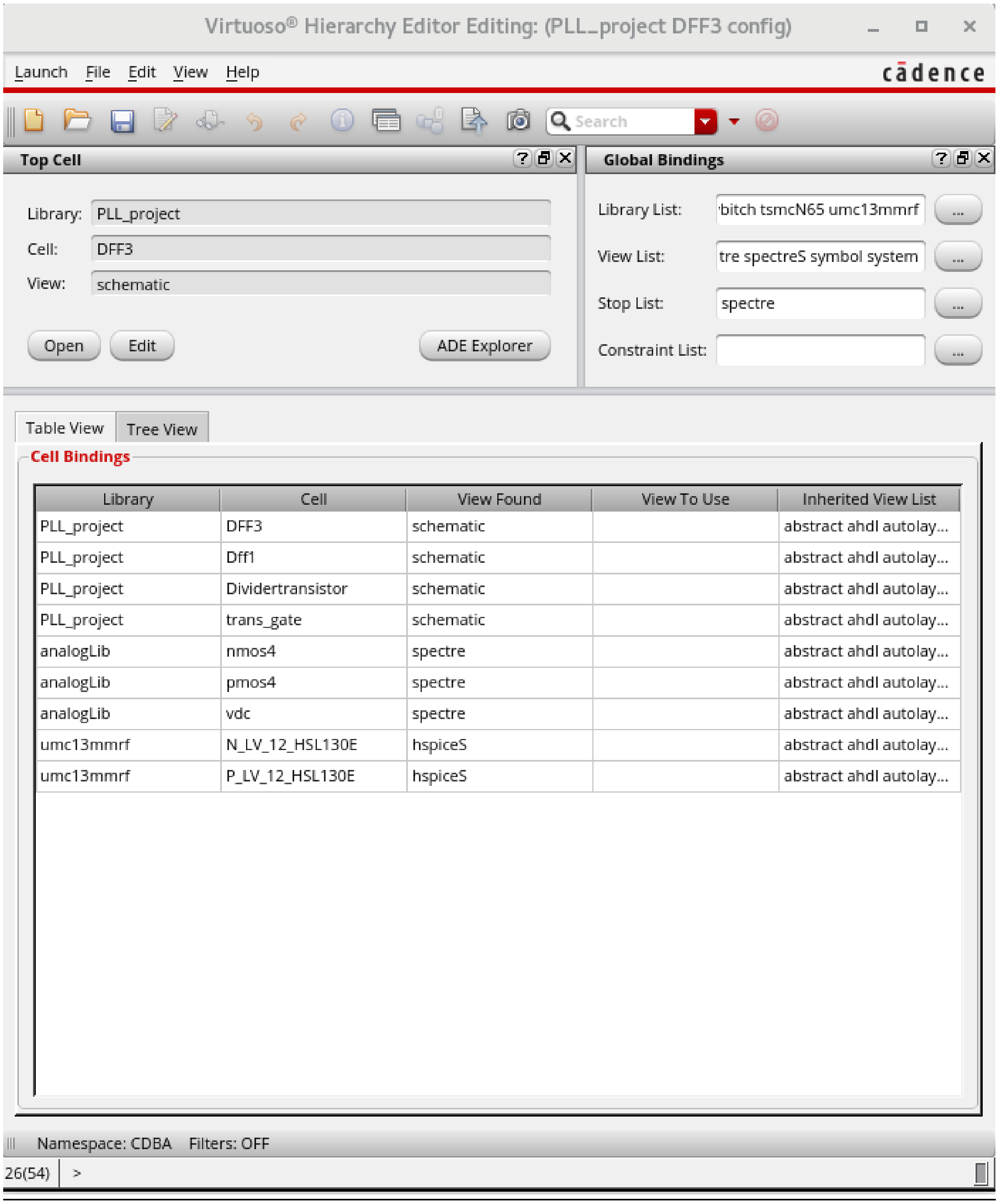
```
Maximum value achieved for any signal of each quantity:
V: V(D) = 1.474 V
I: I(V0:p) = 152.3 mA
If your circuit contains signals of the same quantity that are vastly different in size (such as high voltage circuitry combined with low voltage control circuitry), you should consider specifying global option `bin_relref=yes'.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Post-Transient Simulation Summary
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   -  To further speed up simulation, consider
         add ++aps on command line
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


During simulation, the CPU load for active processors is :
         0 (40.0 %)      1 (41.4 %)      2 (100.0 %)     3 (87.5 %)
        Total: 268.9%
Initial condition solution time: CPU = 1.003 ms, elapsed = 899.076 us.
Intrinsic tran analysis time:    CPU = 227.983 ms, elapsed = 115.735 ms.
Total time required for tran analysis `tran': CPU = 233.985 ms, elapsed = 118.734 ms.
Time accumulated: CPU = 674.837 ms, elapsed = 428.345 ms.
Peak resident memory used = 94.6 Mbytes.


Notice from spectre.
    32 notices suppressed.

finalTimeOP: writing operating point information to rawfile.

Opening the PSF file ../psf/finalTimeOP.info ...
modelParameter: writing model parameter values to rawfile.

Opening the PSF file ../psf/modelParameter.info ...
element: writing instance parameter values to rawfile.

Opening the PSF file ../psf/element.info ...
outputParameter: writing output parameter values to rawfile.

Opening the PSF file ../psf/outputParameter.info ...
designParamVals: writing netlist parameters to rawfile.

Opening the PSFASCII file ../psf/designParamVals.info ...
primitives: writing primitives to rawfile.

Opening the PSFASCII file ../psf/primitives.info.primitives ...
subckts: writing subcircuits to rawfile.

Opening the PSFASCII file ../psf/subckts.info.subckts ...

Aggregate audit (10:14:18 AM, Tue Sep 17, 2024):
Time used: CPU = 757 ms, elapsed = 487 ms, util. = 155%.
Time spent in licensing: elapsed = 25.3 ms, percentage of total = 5.19%.
Peak memory used = 95.8 Mbytes.
Simulation started at: 10:14:17 AM, Tue Sep 17, 2024, ended at: 10:14:18 AM, Tue Sep 17, 2024, with elapsed time (wall clock): 487 ms.
spectre completes with 0 errors, 13 warnings, and 14 notices.
```



```
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    errpreset = moderate
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS
    rabsshort = 1 mOhm


Notice from spectre during transient analysis `tran'.
    Multithreading is disabled due to the size of the design being too small.


Output and IC/nodeset summary:
                save   2       (current)
                save   30      (voltage)

    tran: time = 1.271 ns    (2.54 %), step = 277.3 ps     (555 m%)

Notice from spectre at time = 1.82534 ns during transient analysis `tran'.
    Found trapezoidal ringing on node V0:p.
Notice from spectre at time = 2.41767 ns during transient analysis `tran'.
    Found trapezoidal ringing on node V0:p.
Notice from spectre at time = 3.01 ns during transient analysis `tran'.
    Found trapezoidal ringing on node V0:p.

    tran: time = 4.124 ns    (8.25 %), step = 381.4 ps     (763 m%)

Notice from spectre at time = 4.88638 ns during transient analysis `tran'.
    Found trapezoidal ringing on node V0:p.
Notice from spectre at time = 5.44319 ns during transient analysis `tran'.
    Found trapezoidal ringing on node V0:p.
        Further occurrences of this notice will be suppressed.

    tran: time = 6.252 ns    (12.5 %), step = 3.468 ps    (6.94 m%)
    tran: time = 9.01 ns       (18 %), step = 748.7 ps      (1.5 %)
    tran: time = 12 ns         (24 %), step = 765.8 ps     (1.53 %)
    tran: time = 13.91 ns    (27.8 %), step = 634.4 ps     (1.27 %)
    tran: time = 16.59 ns    (33.2 %), step = 627.5 ps     (1.25 %)
    tran: time = 18.8 ns     (37.6 %), step = 130 ps       (260 m%)
    tran: time = 21.27 ns    (42.5 %), step = 33.1 ps     (66.2 m%)
    tran: time = 24 ns         (48 %), step = 711.2 ps     (1.42 %)
    tran: time = 27.01 ns      (54 %), step = 829.7 ps     (1.66 %)
    tran: time = 29.19 ns    (58.4 %), step = 809.5 ps     (1.62 %)
    tran: time = 31.48 ns      (63 %), step = 480.9 ps     (962 m%)
    tran: time = 33.85 ns    (67.7 %), step = 169.8 ps     (340 m%)
    tran: time = 36.26 ns    (72.5 %), step = 9.707 ps    (19.4 m%)
    tran: time = 39.01 ns      (78 %), step = 541.7 ps     (1.08 %)
    tran: time = 41.29 ns    (82.6 %), step = 709.3 ps     (1.42 %)
    tran: time = 44.28 ns    (88.6 %), step = 764.4 ps     (1.53 %)
    tran: time = 46.56 ns    (93.1 %), step = 453 ps       (906 m%)
    tran: time = 48.84 ns    (97.7 %), step = 136.4 ps     (273 m%)
17 >
```

16. Replace the Verilog-A view of the divider with the schematic (transistor level) view.

Deliver a snapshot of the new configuration in hierarchy editor.



Library: PLL_project
Cell: DFF3
View: schematic

Library List: bitch tsmcN65 umc13mmrf
View List: tre spectreS symbol system
Stop List: spectre
Constraint List:

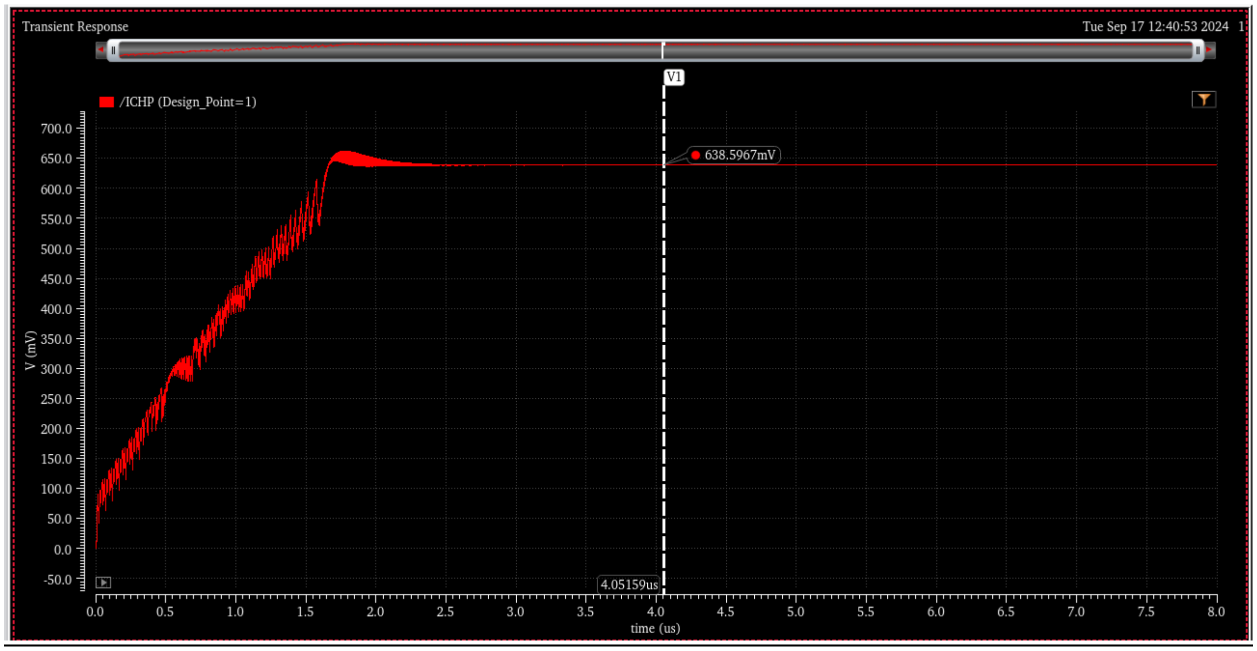
Library	Cell	View Found	View To Use	Inherited View List
PLL_project	DFF3	schematic		abstract ahdl autolay...
PLL_project	Dff1	schematic		abstract ahdl autolay...
PLL_project	Dividertransistor	schematic		abstract ahdl autolay...
PLL_project	trans_gate	schematic		abstract ahdl autolay...
analogLib	nmos4	spectre		abstract ahdl autolay...
analogLib	pmos4	spectre		abstract ahdl autolay...
analogLib	vdc	spectre		abstract ahdl autolay...
umc13mmrf	N_LV_12_HSL130E	hspiceS		abstract ahdl autolay...
umc13mmrf	P_LV_12_HSL130E	hspiceS		abstract ahdl autolay...

Namespace: CDBA Filters: OFF
26(54) >

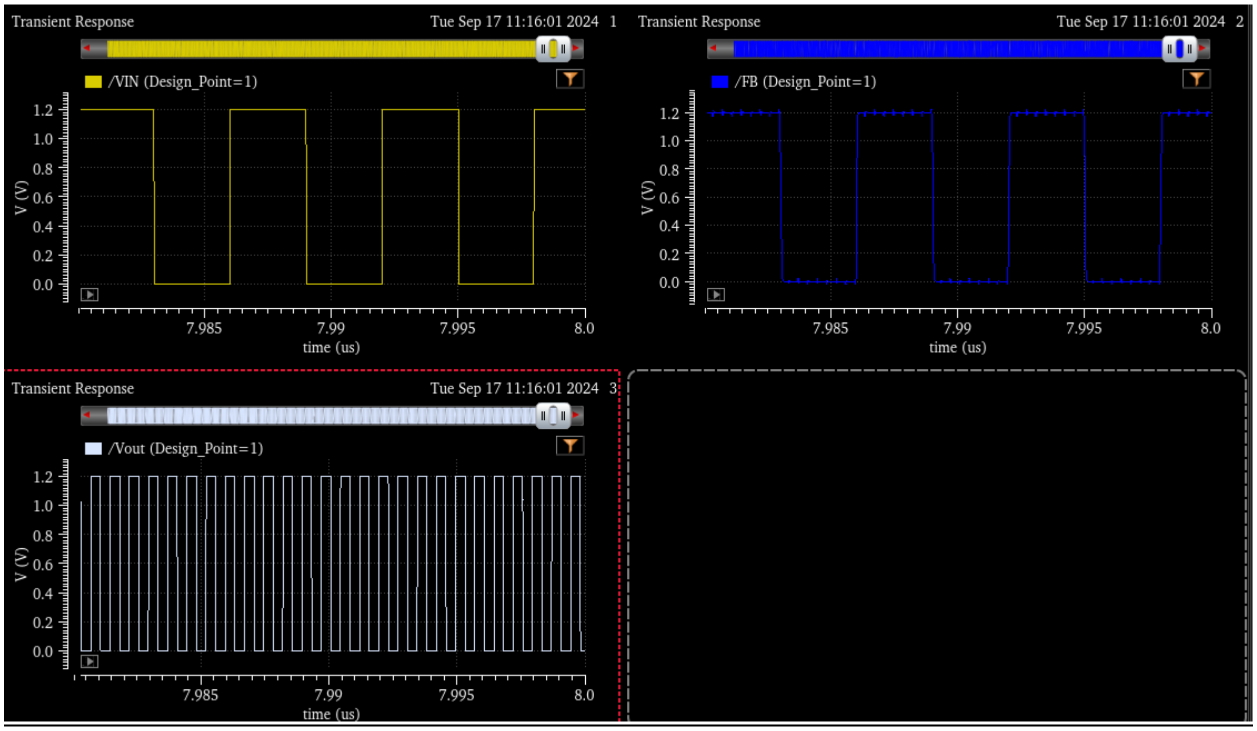
Config view

<u>17. Integrate the PLL as shown in Fig. 6.2 using Verilog-A models of all blocks except the divider. Use the schematic view of the divider. Simulate the PLL using config view</u>

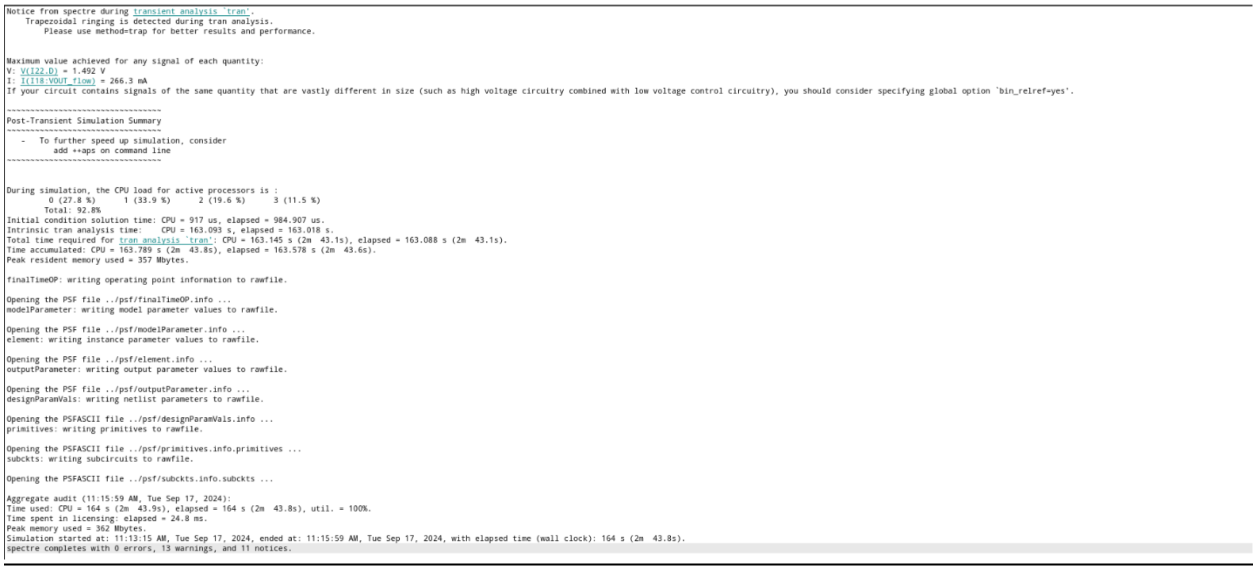
and hierarchy editor. Deliver a snapshot clearly illustrating the VCO control voltage (similar to Fig. 6.23).



18. Integrate the PLL as shown in Fig. 6.2 using Verilog-A models of all blocks except the divider. Use the schematic view of the divider. Simulate the PLL using config view and hierarchy editor. Deliver a snapshot clearly illustrating the PLL operation (similar to Fig. 6.24). Add “A/B marker” to indicate the reference and output frequencies.



<u>19. Deliver a snapshot of the log file of the previous simulation clearly illustrating the success of the simulation and the simulation time.</u>



```
Notice from spectre during transient analysis `tran'.
    Trapezoidal ringing is detected during tran analysis.
        Please use method=trap for better results and performance.


Maximum value achieved for any signal of each quantity:
V: V(I22.D) = 1.492 V
I: I(I18:VOUT_flow) = 266.3 mA
If your circuit contains signals of the same quantity that are vastly different in size (such as high voltage circuitry combined with low voltage control circuitry), you should consider specifying global option `bin_relref=yes'.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Post-Transient Simulation Summary
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   -   To further speed up simulation, consider
          add ++aps on command line
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



During simulation, the CPU load for active processors is :
         0 (27.8 %)      1 (33.9 %)      2 (19.6 %)      3 (11.5 %)
        Total: 92.8%
Initial condition solution time: CPU = 917 us, elapsed = 984.907 us.
Intrinsic tran analysis time:    CPU = 163.093 s, elapsed = 163.018 s.
Total time required for tran analysis `tran': CPU = 163.145 s (2m  43.1s), elapsed = 163.088 s (2m  43.1s).
Time accumulated: CPU = 163.789 s (2m  43.8s), elapsed = 163.578 s (2m  43.6s).
Peak resident memory used = 357 Mbytes.

finalTimeOP: writing operating point information to rawfile.

Opening the PSF file ../psf/finalTimeOP.info ...
modelParameter: writing model parameter values to rawfile.

Opening the PSF file ../psf/modelParameter.info ...
element: writing instance parameter values to rawfile.

Opening the PSF file ../psf/element.info ...
outputParameter: writing output parameter values to rawfile.

Opening the PSF file ../psf/outputParameter.info ...
designParamVals: writing netlist parameters to rawfile.

Opening the PSFASCII file ../psf/designParamVals.info ...
primitives: writing primitives to rawfile.

Opening the PSFASCII file ../psf/primitives.info.primitives ...
subckts: writing subcircuits to rawfile.

Opening the PSFASCII file ../psf/subckts.info.subckts ...

Aggregate audit (11:15:59 AM, Tue Sep 17, 2024):
Time used: CPU = 164 s (2m  43.9s), elapsed = 164 s (2m  43.8s), util. = 100%.
Time spent in licensing: elapsed = 24.8 ms.
Peak memory used = 362 Mbytes.
Simulation started at: 11:13:15 AM, Tue Sep 17, 2024, ended at: 11:15:59 AM, Tue Sep 17, 2024, with elapsed time (wall clock): 164 s (2m  43.8s).
spectre completes with 0 errors, 13 warnings, and 11 notices.
```

<u>20. Compare the simulation time of (13) and (17). Comment.</u>

Simulation time of Verilog A model is: 757ms

Simulation time of transistor level model is: 164s

The time taken by transistor level model is much higher .