


# AMS Course

## PLL Project

### Result

79

Lower Limit	<input type="text" value="50"/>
Upper Limit	<input type="text" value="400"/>
<div><div>Generate </div><div>Clear</div></div>	

$$\underline{K_{vco} = 79/100 * 1e9 = 0.79 \text{ GHz/V}}$$

### 1- PFD Verilog-a model :

```
// 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 PFD(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 = 10p, tfall = 10p, td = 0;

    input REF,FB;
    output UP,DN;

    electrical REF,FB,UP,DN;

    // Internal UP and DN signals
    // *** add line here ***
    real DN_i=0 , UP_i=0;

    analog begin
```

```

analog begin

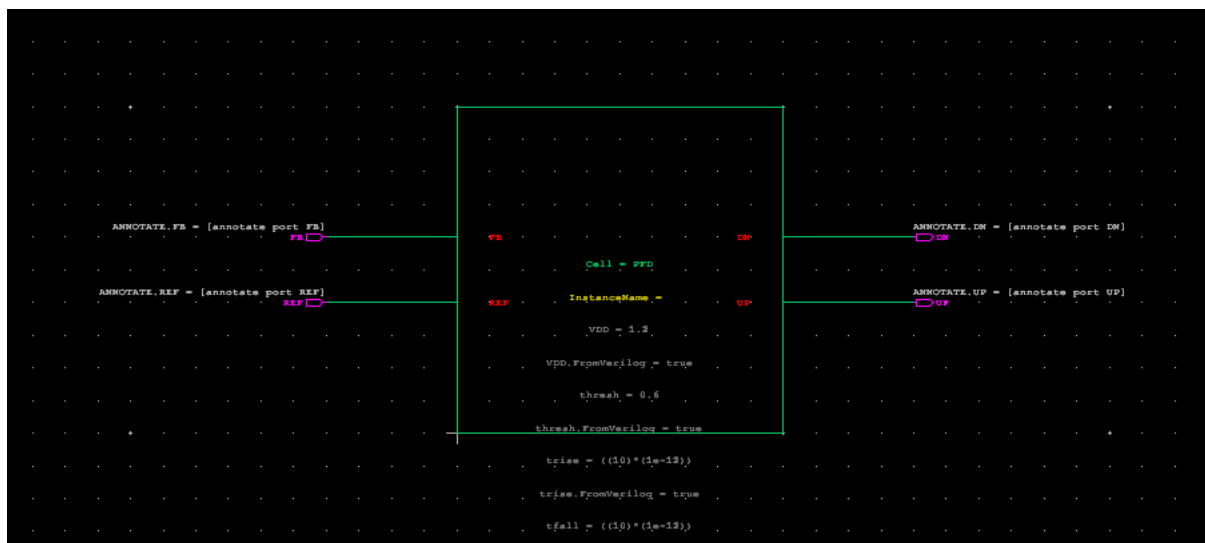
    // Check DN_i state when REF arrives
    // *** add line here ***
    @(cross(V(REF) - thresh,1))
        if(DN_i < thresh)
            // *** add line here ***
            UP_i=VDD;
        else begin
            UP_i = 0;
            DN_i = 0;
        end

    // 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

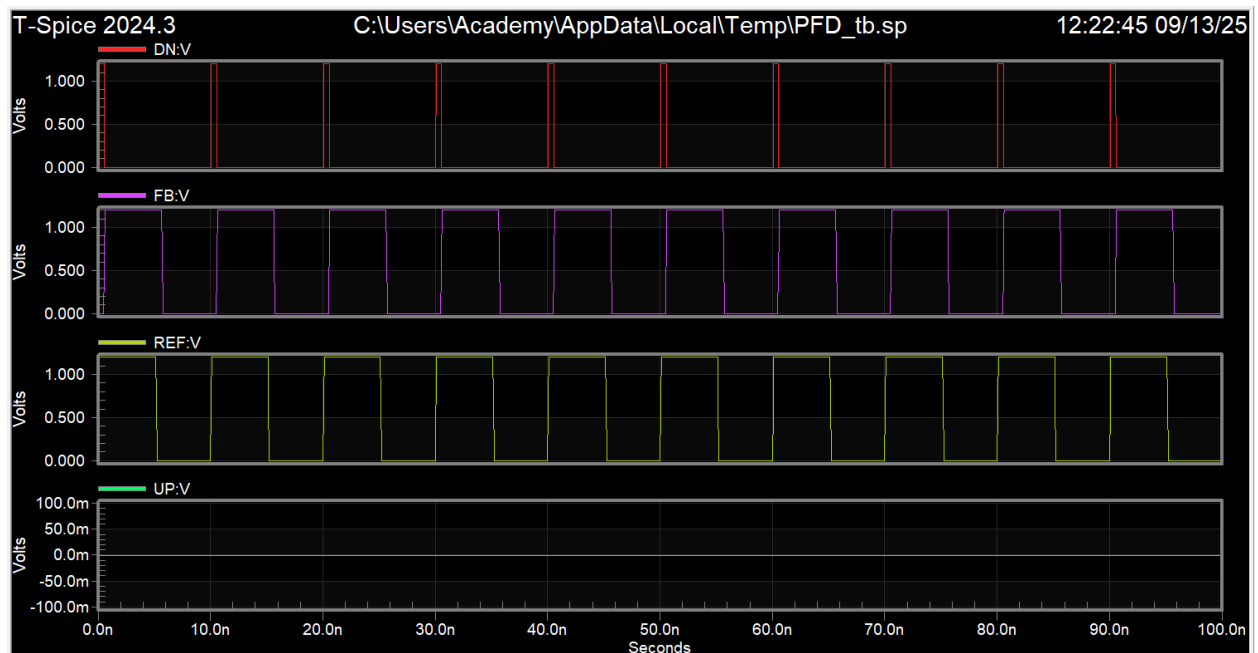
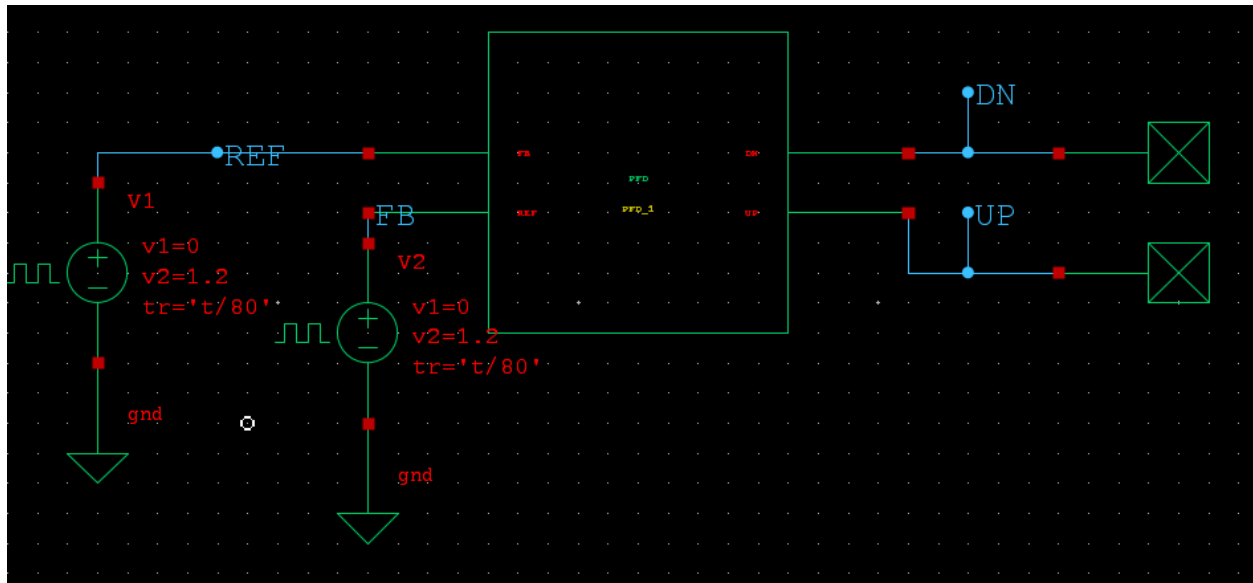
    V(UP) <+ transition(UP_i,td,trise,tfall);
    // *** add line here ***
    V(DN) <+ transition(DN_i,td,trise,tfall);

end
endmodule

```



## 2- PFD testbench and results:



From the charts we can see that we have down pulses due to the leading of REF signal towards the FB signal while the up signal is zero

### 3- CHP Verilog-a model:

```
// AMS PLL Project: Charge Pump (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=10p, tfall=10p, td=0;

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

```
    analog begin
```

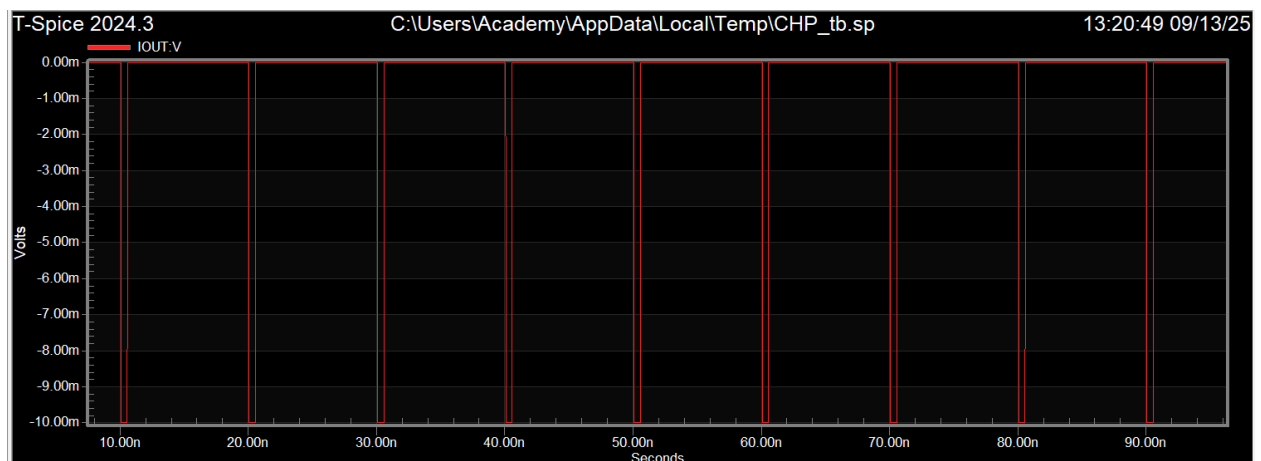
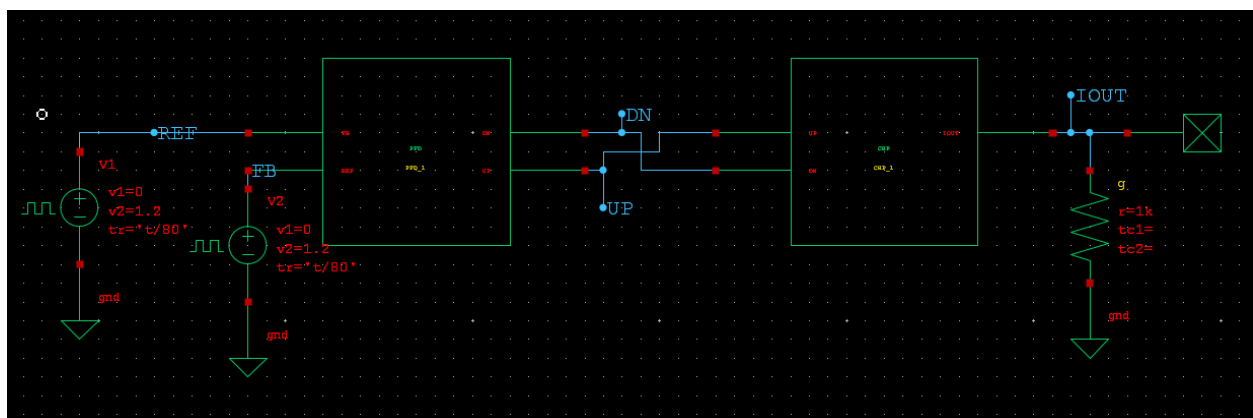
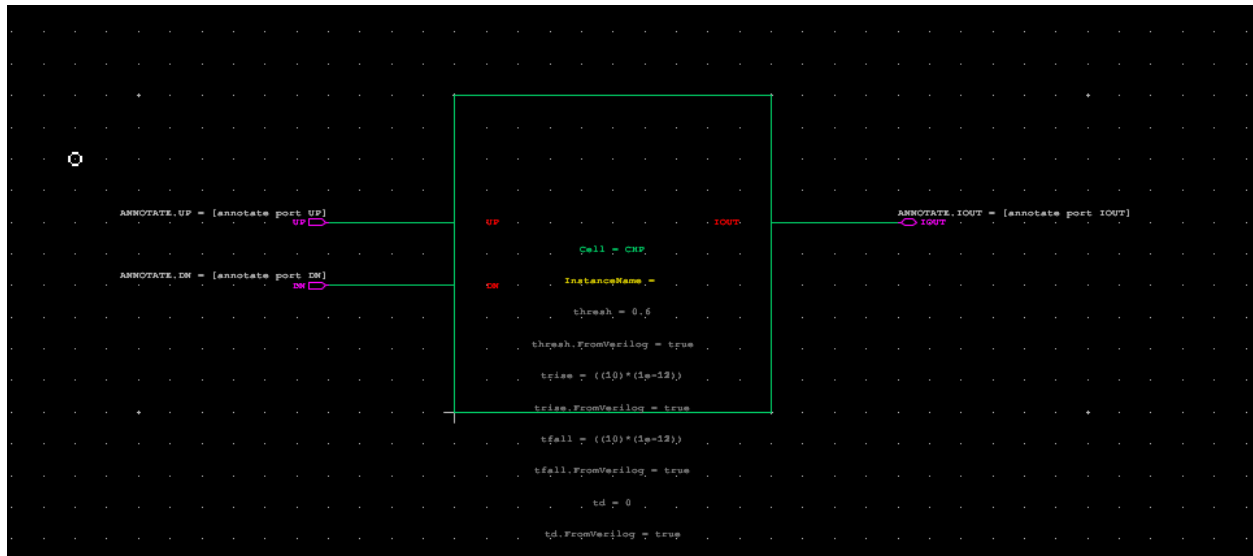
```
        // Generate events at UP and DN transitions
        @(cross(V(UP)-thresh,0))
        ;
        // *** add line here ***
        @(cross(V(DN)-thresh,0))
        ;
        // *** add line here ***

        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;

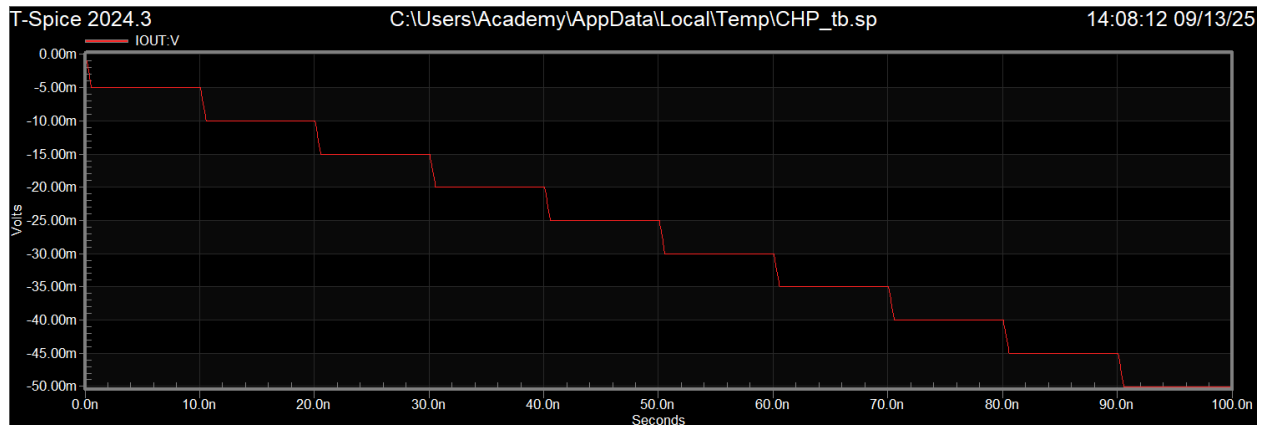
        I(IOUT) <+ transition(IOUT_i,td,trise,tfall);
```

```
    end
endmodule
```

#### 4- CHP testbench and results:



(in case of resistive load )



In case of capacitive load

Since feedback leads input voltage , the output current is negative .

## 5- VCO Verilog-a model :

```
// 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=1.465 from (Vmin:inf);
    parameter real Fmin=0.5G from (0:inf);
    parameter real Fmax=1.5G from (Fmin:inf);
    parameter real trise=10p, tfall=10p, td=0;

    input VCTRL;
    output VOUT;
    voltage VCTRL,VOUT;

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

    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 < Fmin) freq = Fmin;
        if (freq > Fmax) freq = Fmax;
```

```

// calculate the phase (modulo 2*pi)
// *** add line here ***
phase = 2*M_PI*idtmmod(freq,0,1,-0.5);

// generate the output
sine = sin(phase);

@(cross(sine,0))
    ;

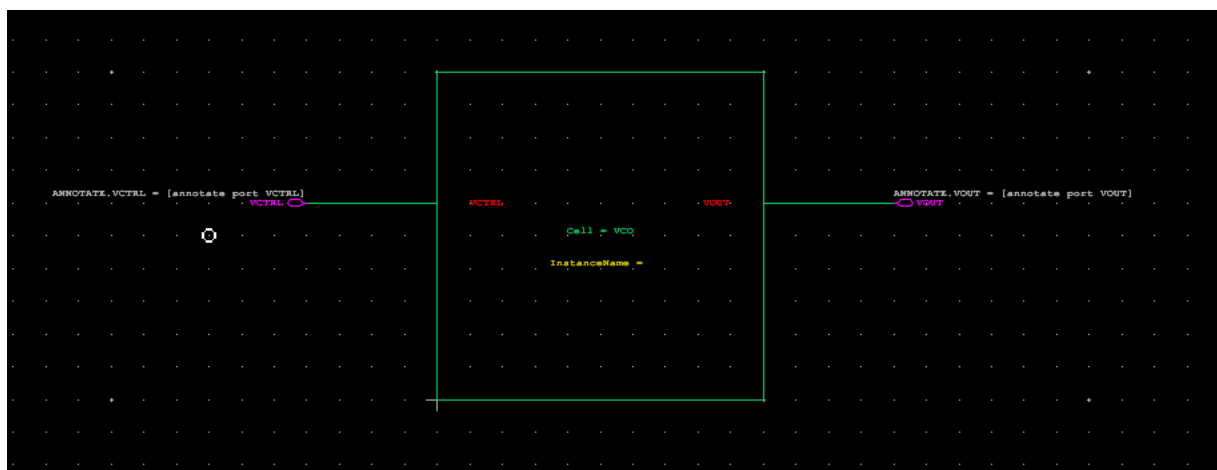
if (sine > 0)
    // *** add line here ***
    VOUT_i = VHIGH;
else
    // *** add line here ***
    VOUT_i = 0;

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

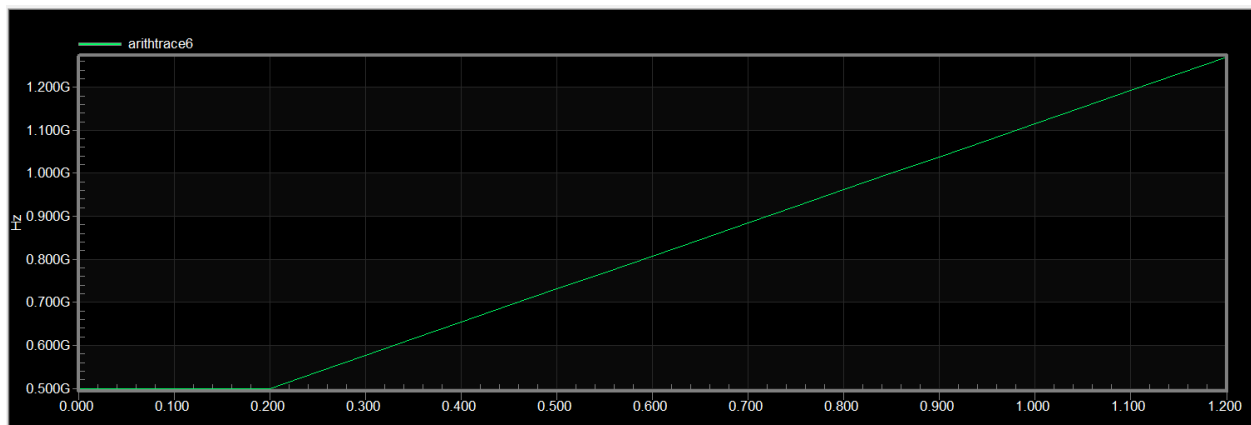
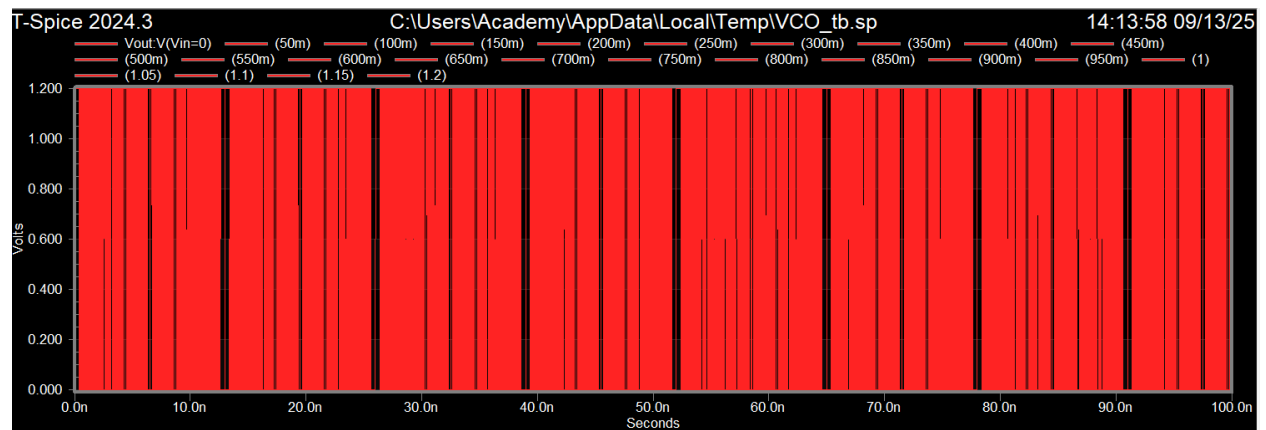
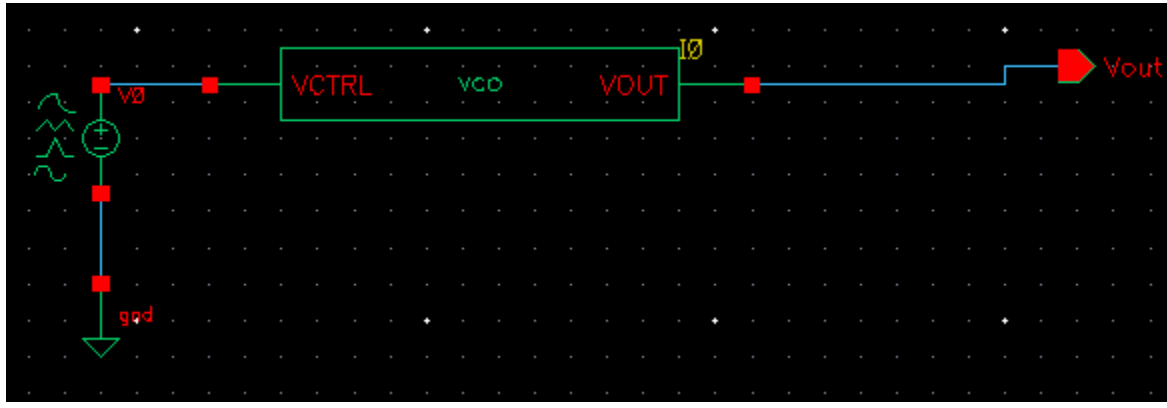
// bound the time step
// *** add line here ***
$bound_step(0.1/freq);
end
endmodule

```

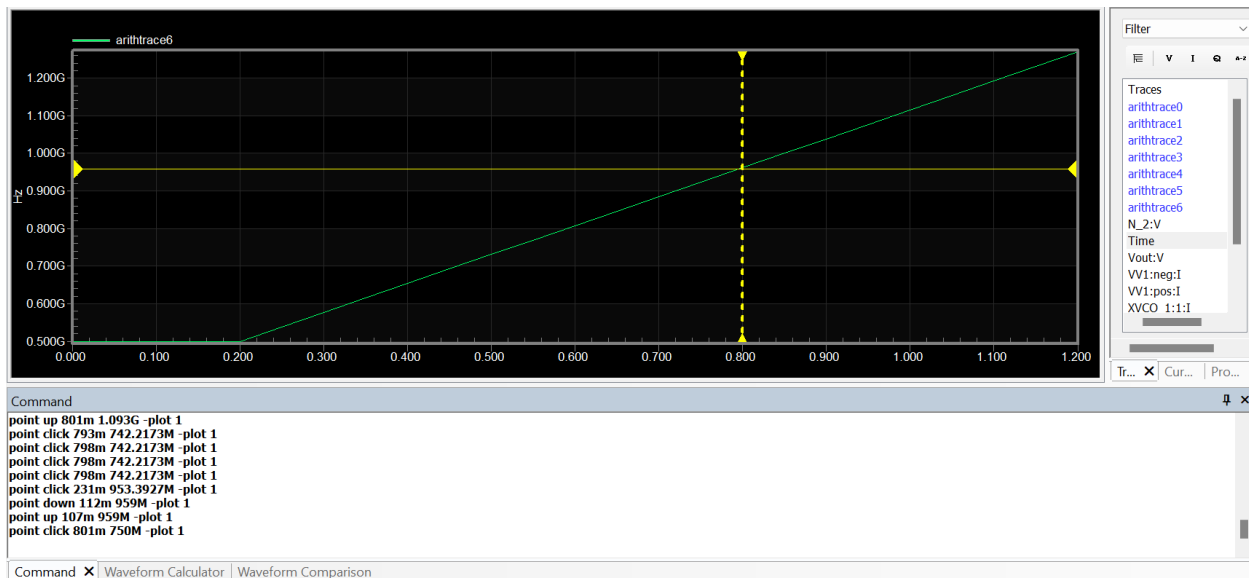
---



## 6- VCO testbench and results:







At the desired output frequency ( $8 \times 120\text{M} = 960\text{ MHz}$ ), the  $V_{ctrl}$  value is nearly  $0.79\text{ V}$

## 7- DIVIDER Verilog-a model :

```
`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=0;
```

```

analog begin

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

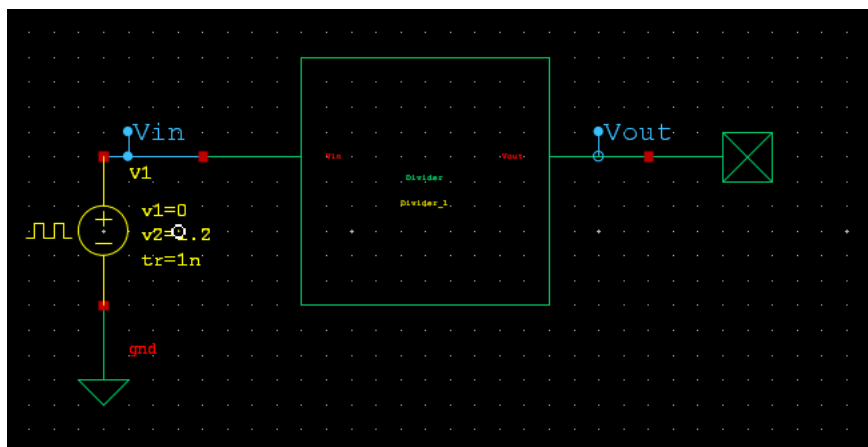
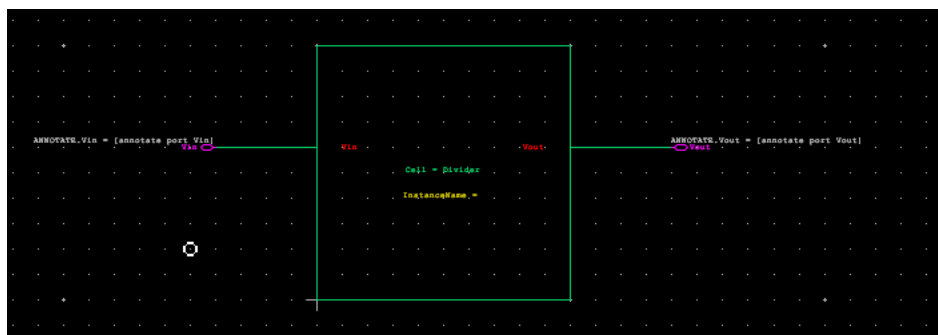
    V(VOUT) <+ transition(out_value, td, tt);

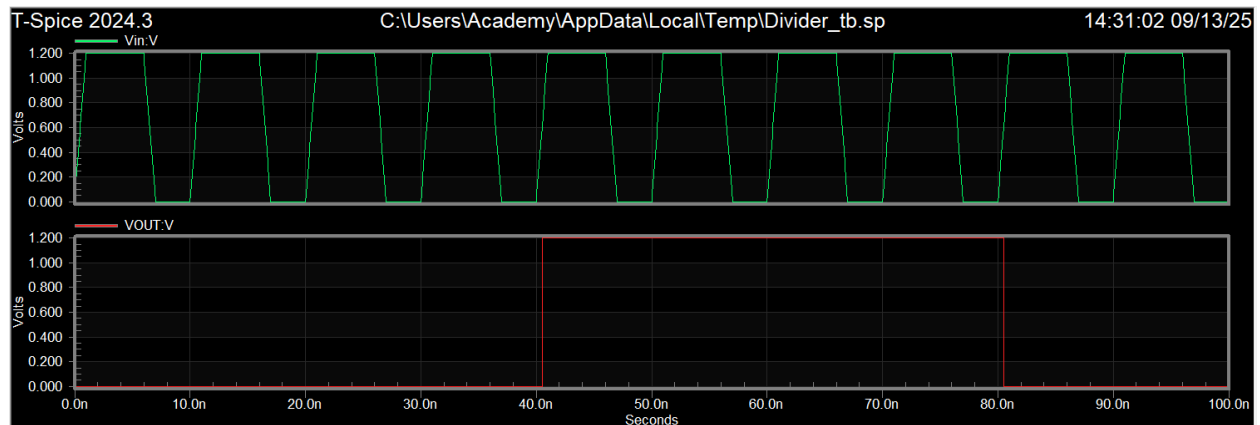
end
endmodule

```

---

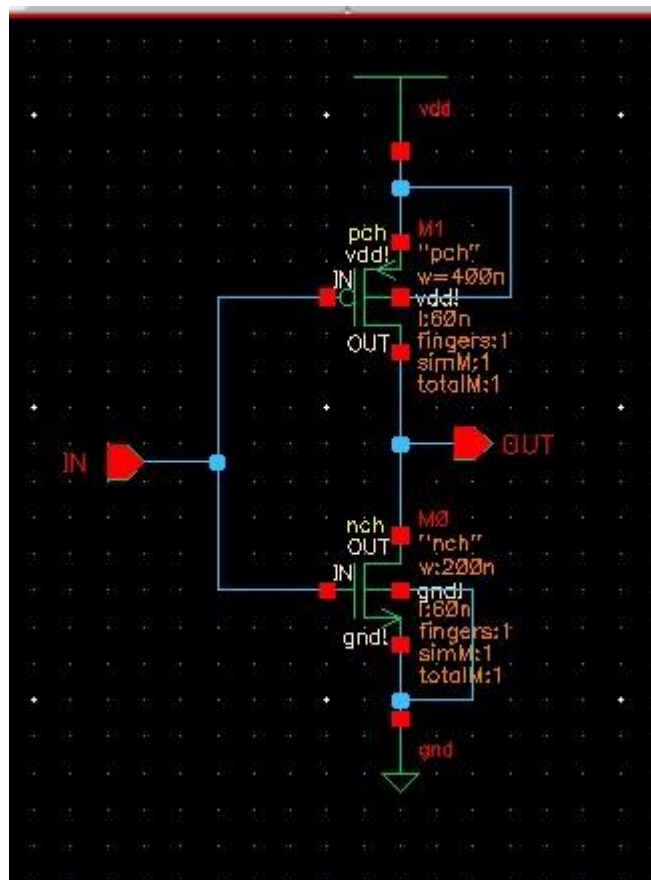
## 8- Divider testbench and results :



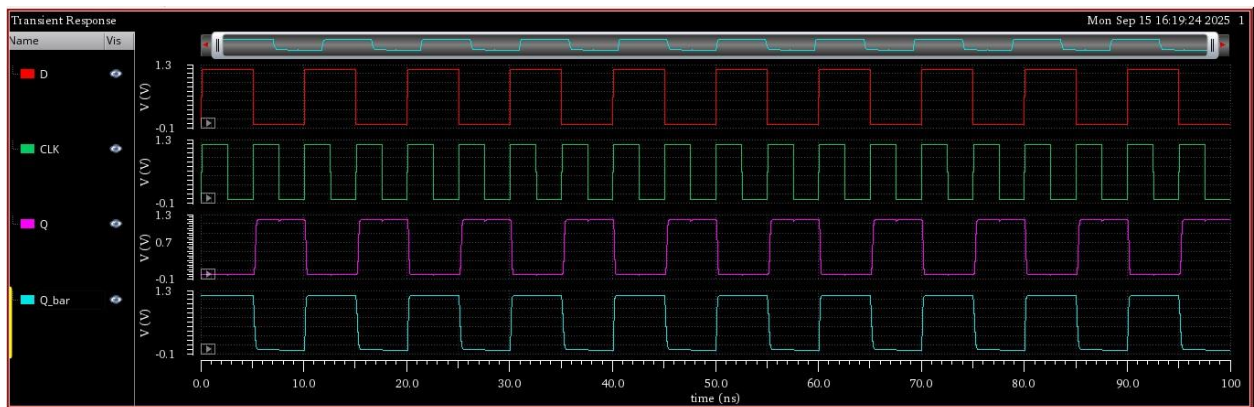
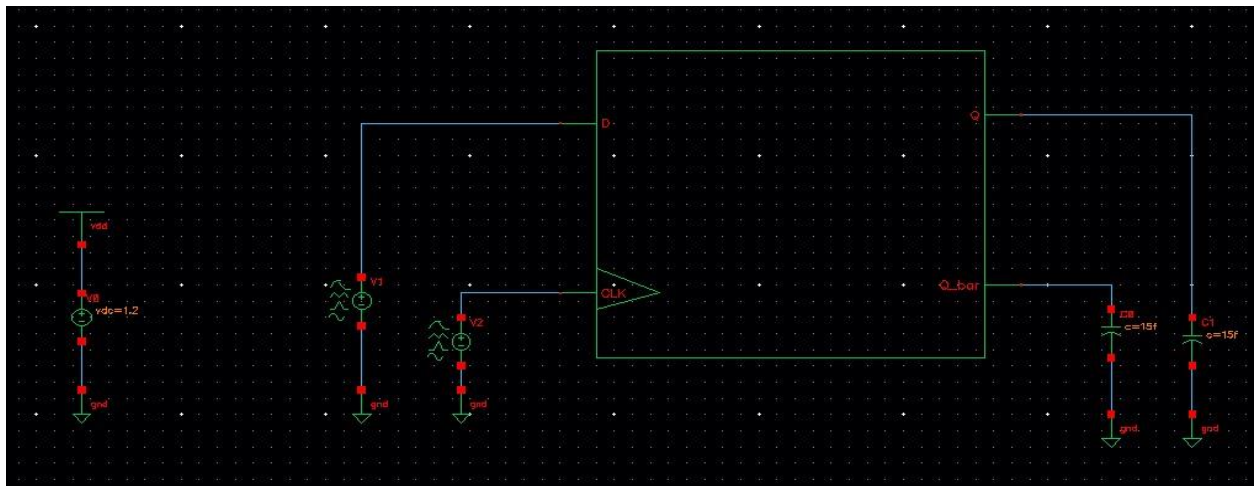
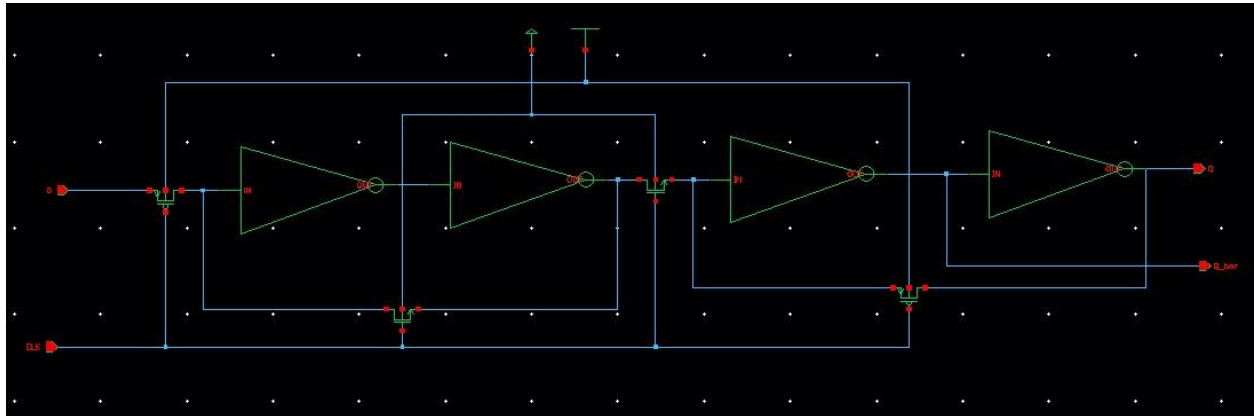


9-

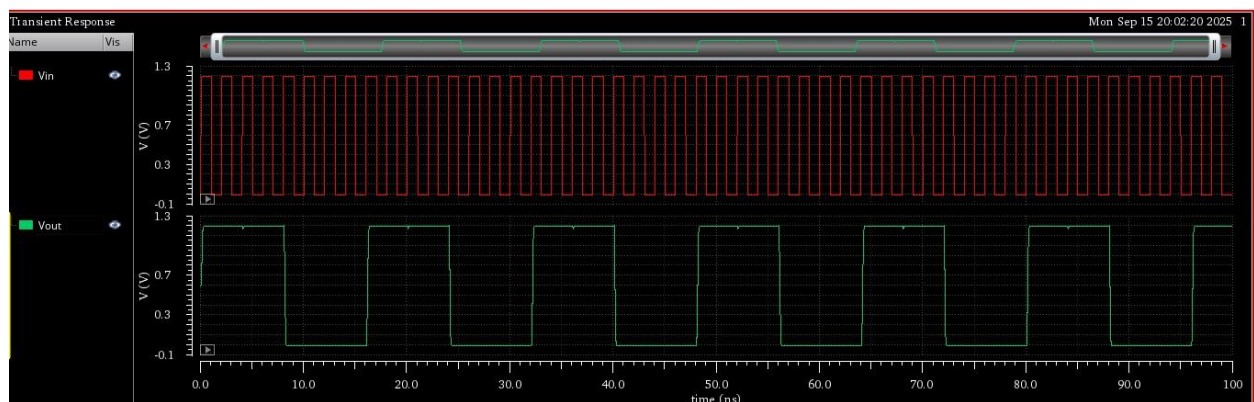
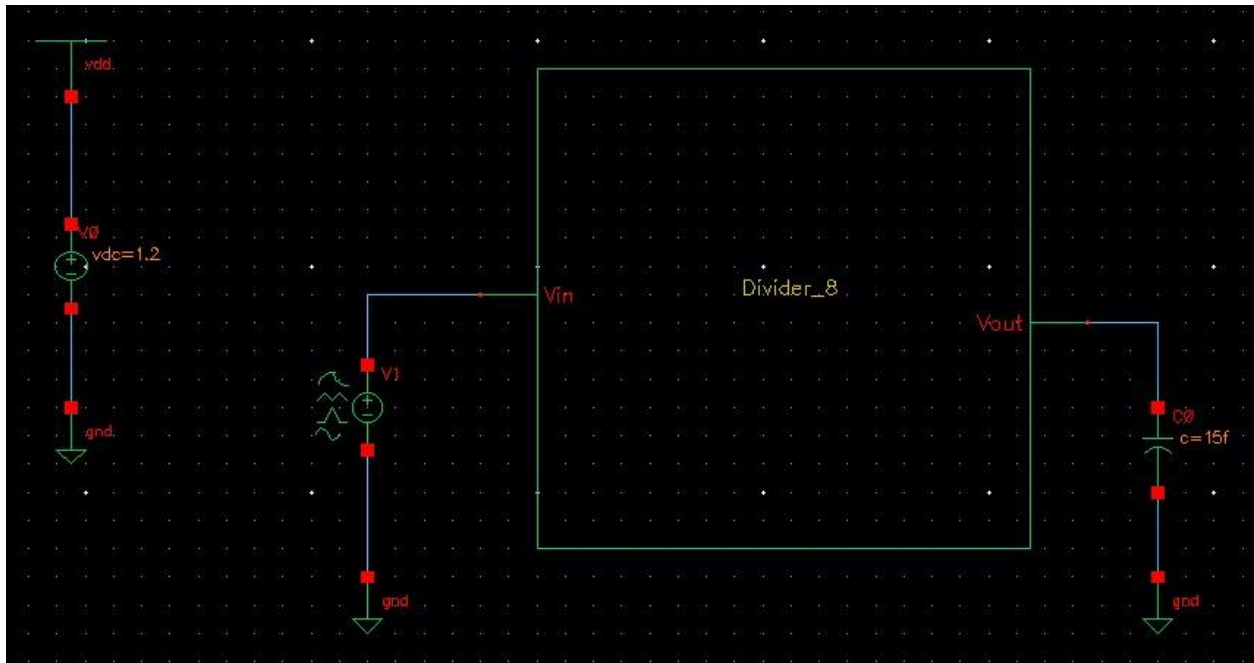
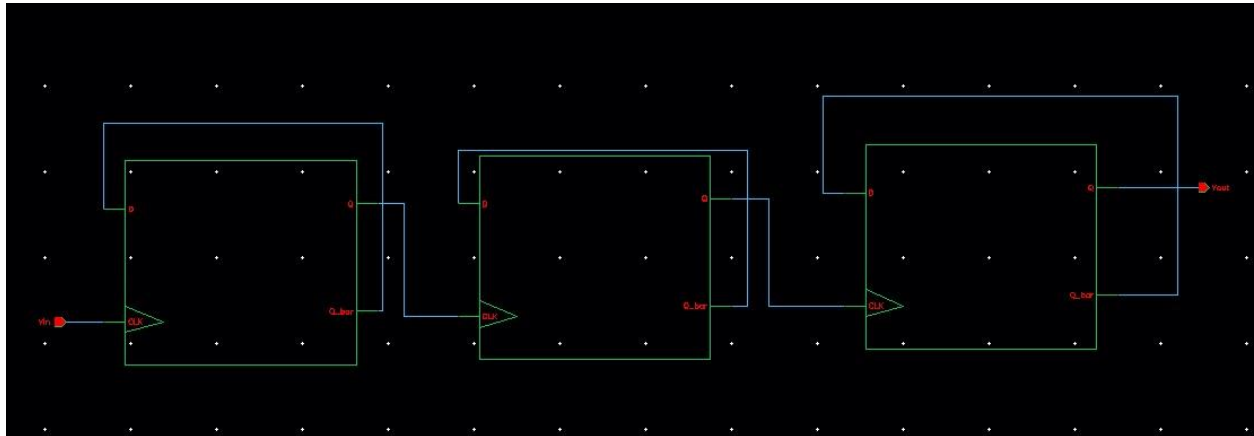
## Inverter design



## D\_ff design and testbench :



## 10- transistor level divider design and testbench :





## 11- pll integration using Verilog\_a models only:

**Top Cell**

Library: pll  
Cell: verilog\_a\_model\_test  
View: schematic

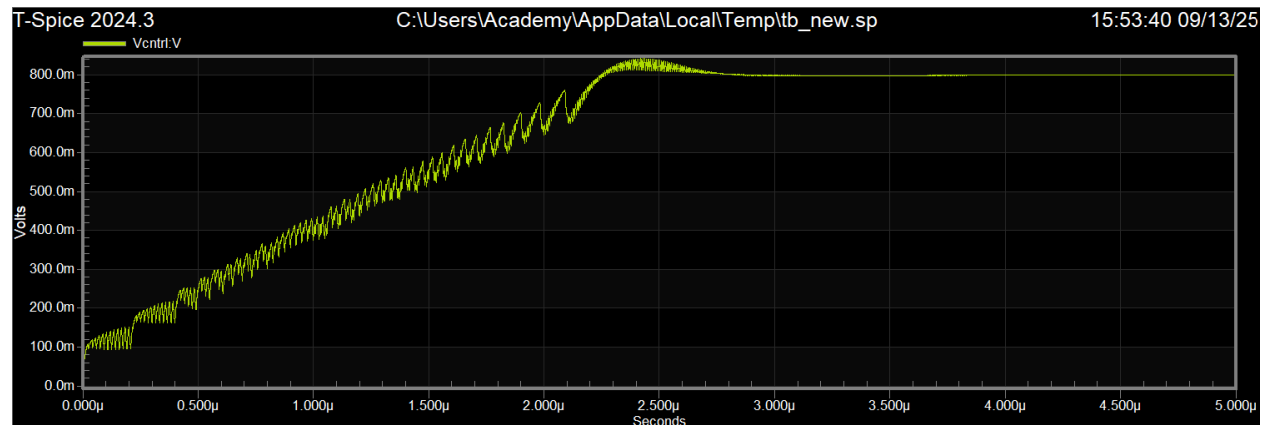
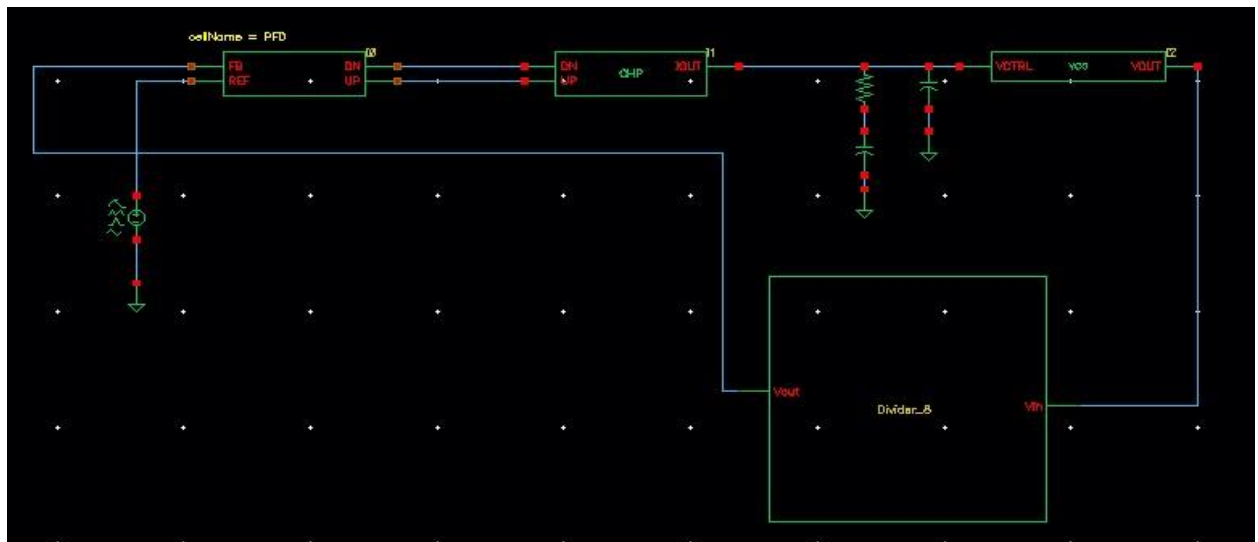
Open Edit ADEL

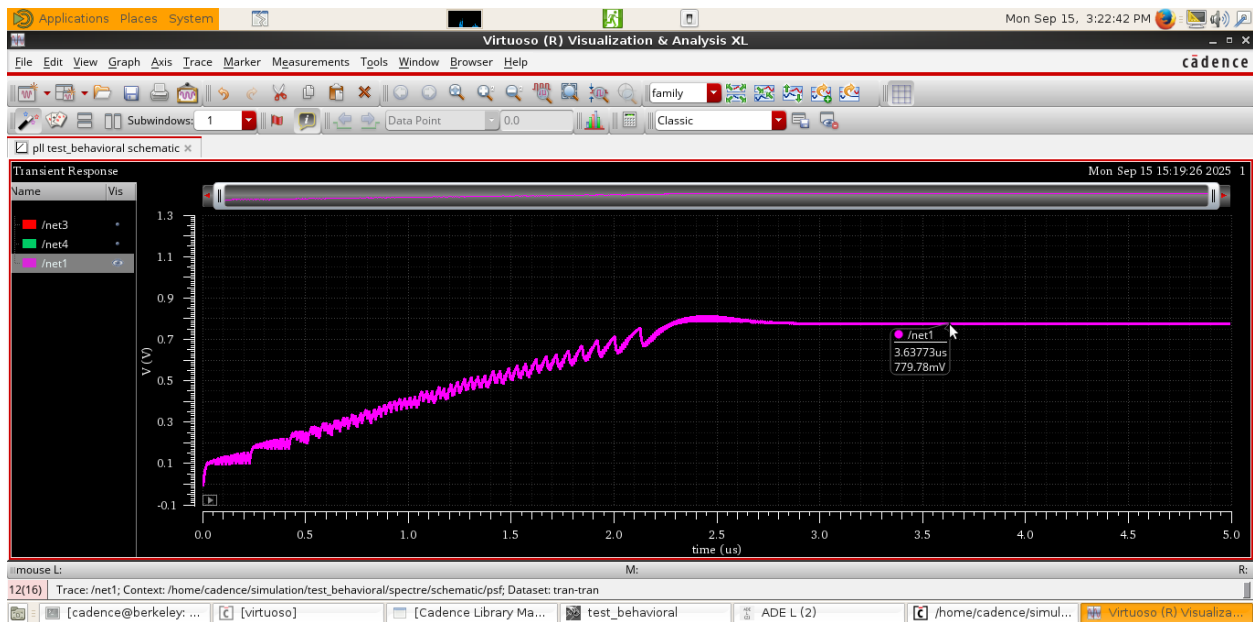
**Global Bindings**

Library List: ahdLib analogLib pll tsmcN65 functional basic US\_8ths  
View List: ahdl behavioral functional hspiceS schematic symbol  
Stop List: spectre  
Constraint List:

**Cell Bindings**

Library	Cell	View Found	View To Use	Inherited View List
analogLib	cap	spectre	spectre	ahdl behavioral functional hspiceS sche...
analogLib	res	spectre	spectre	ahdl behavioral functional hspiceS sche...
analogLib	vpulse	spectre	spectre	ahdl behavioral functional hspiceS sche...
pll	CHP	veriloga	veriloga	ahdl behavioral functional hspiceS sche...
pll	Divider	veriloga	veriloga	ahdl behavioral functional hspiceS sche...
pll	PFD	veriloga	veriloga	ahdl behavioral functional hspiceS sche...
pll	VCO	veriloga	veriloga	ahdl behavioral functional hspiceS sche...
pll	verilog_a_model_test	schematic		ahdl behavioral functional hspiceS sche...





$$12 - F_{out} = K_{vco} * V + c$$

For  $V_{min} = 0.2$  ,  $F_{min} = 0.5$  GHz ,  $K_{vco} = 0.79$  GHz/V

$$C = 342e6$$

Second substitution when  $F_{out}$  is the required output frequency (960MHz)

$$960e6 = 0.79 \text{ GHz/V} * V_{ctrl} + c$$

$$V_{ctrl} = 0.7822 \text{ V}$$

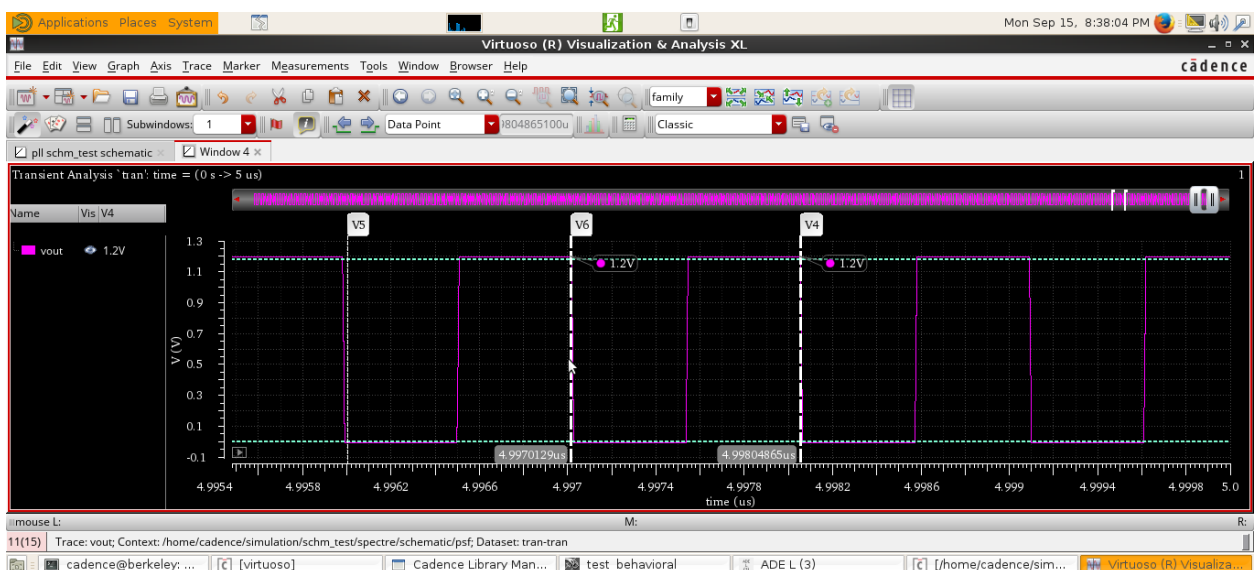
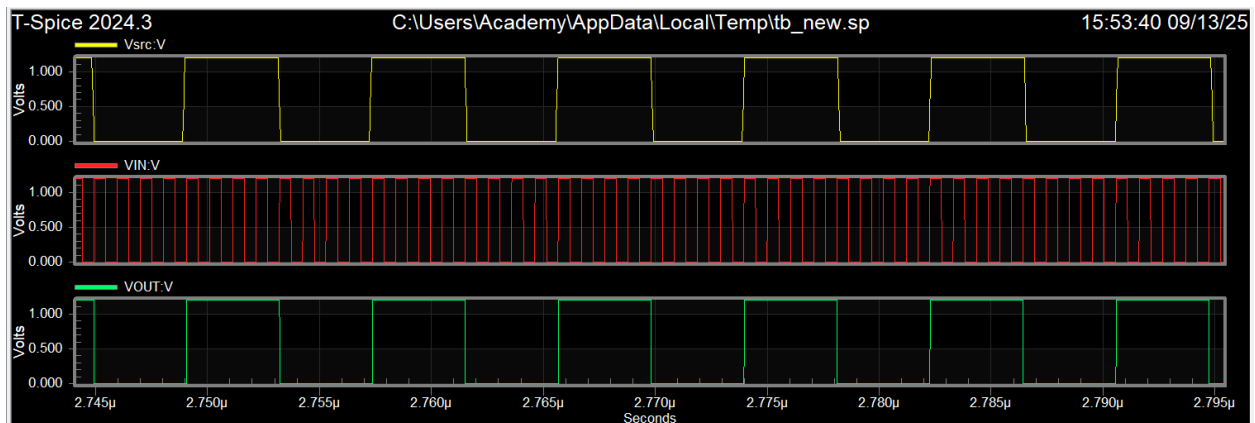
13-

Vctrl simulated	Vctrl analytically
0.779 V	0.782 V

Voltage 1	0 V
Voltage 2	1.2 V
Period	8.3n s
Delay time	
Rise time	10p s
Fall time	10p s
Pulse width	4.167n s
Temperature coefficient 1	

there is slightly difference between the two values as the pulse width and period in our simulation are not 100% accurate.

14 -



$$\text{Output frequency} = 1/(4.99804865\text{us}-4.9970129\text{us}) = 965.48 \text{ MHz}$$



(not exactly 960MHz because our source period is not accurate 8.3ns)

We can see that Vref locks on Vsource with frequency 120MHz and the output frequency is 960MHz .

Note that we had a problem in tanner and needed to switch wiring names so here VOUT represents the feedback and VIN represents the output .

15-

```
generate netlist...
Loading seCore.cxt
Initializing the control file using cp:
  cp /usr/local/cadence/IC617/tools.lnx86/dfII/etc/si/control.spectre /home/cadence/simulation/test_behavioral/spectre/schematic/netlist/control
Copying Spectre source file 'spectre.inp'
Copying Spectre command file 'spectre.sim'
Begin Incremental Netlisting Sep 15 14:45:05 2025
End netlisting Sep 15 14:45:05 2025
Loading monte.cxt

Netlisting Statistics:
  Number of components:    8

  Elapsed time:           4.0s (2.00/s)
Errors: 0  Warnings: 0
...successful.
compose simulator input file...
...successful.
start simulator if needed...
...successful.
Loading paraplots.cxt
simulate...
INFO (ADE-3071): Simulation completed successfully.
reading simulation data...
...successful.
```

#### Post-Transient Simulation Summary

- To further speed up simulation, consider
  - add ++aps on command line
  - add +cktpreset=sampled on command line for ADC/DAC simulation
  - add +cktpreset=pll on command line for PLL simulation

```
Initial condition solution time: CPU = 0 s, elapsed = 1.92881 ms.
Intrinsic tran analysis time:   CPU = 10.716 s, elapsed = 10.7218 s.
Total time required for tran analysis 'tran': CPU = 10.756 s, elapsed = 10.7851 s.
Time accumulated: CPU = 42.508 s, elapsed = 46.7172 s.
Peak resident memory used = 145 Mbytes.
```

```
finalTimeOP: writing operating point information to rawfile.
modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.
```

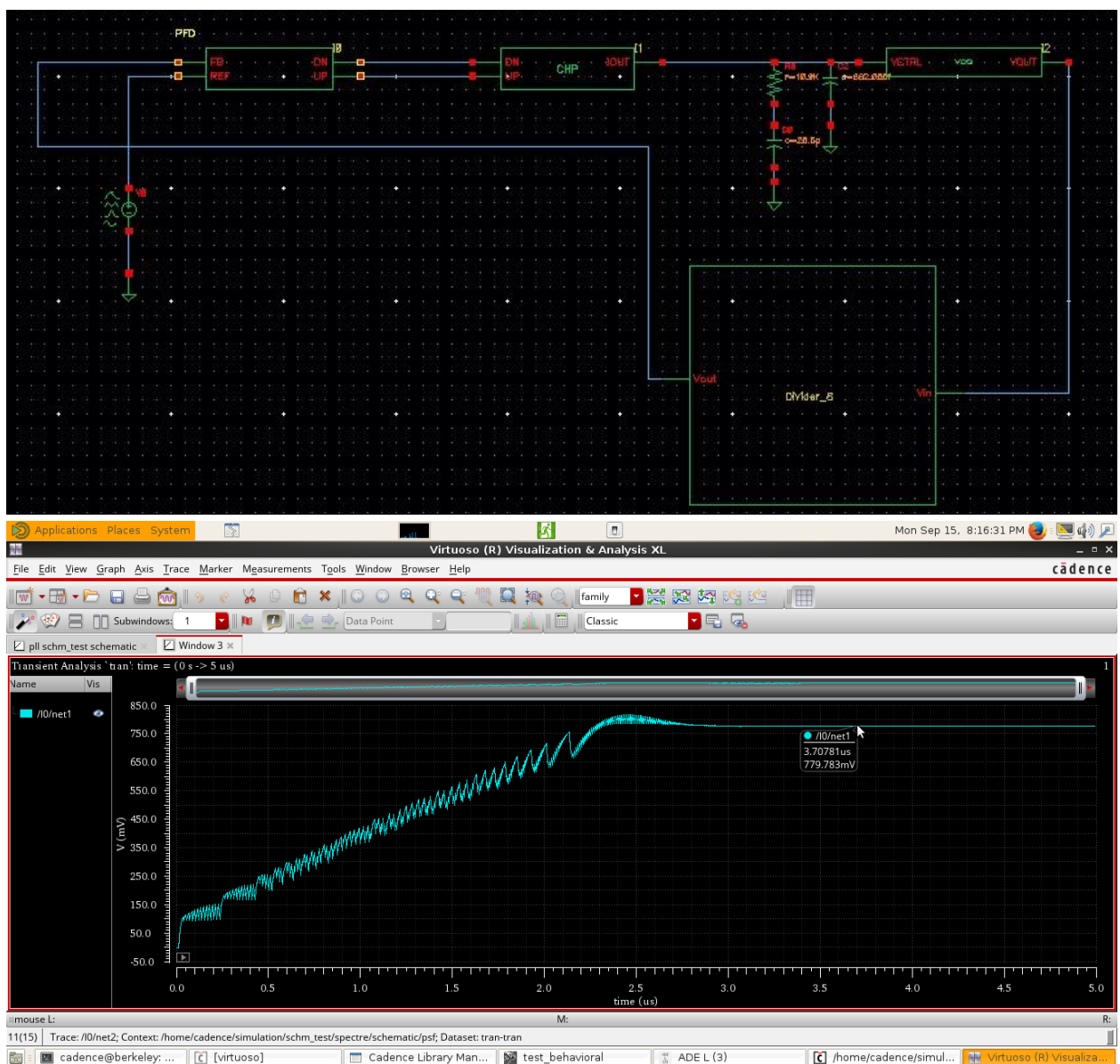
16-

Top Cell					Global Bindings				
Library: <input type="text"/>					Library List: ahdLib analogLib pll tsmcN65				
Cell: <input type="text"/>					View List: behavioral ahd functional hspiceS schematic spectre				
View: <input type="text"/>					Stop List: spectre				
<input type="button" value="Open"/> <input type="button" value="Edit"/>					Constraint List: <input type="text"/>				
<input type="button" value="ADE L"/>									

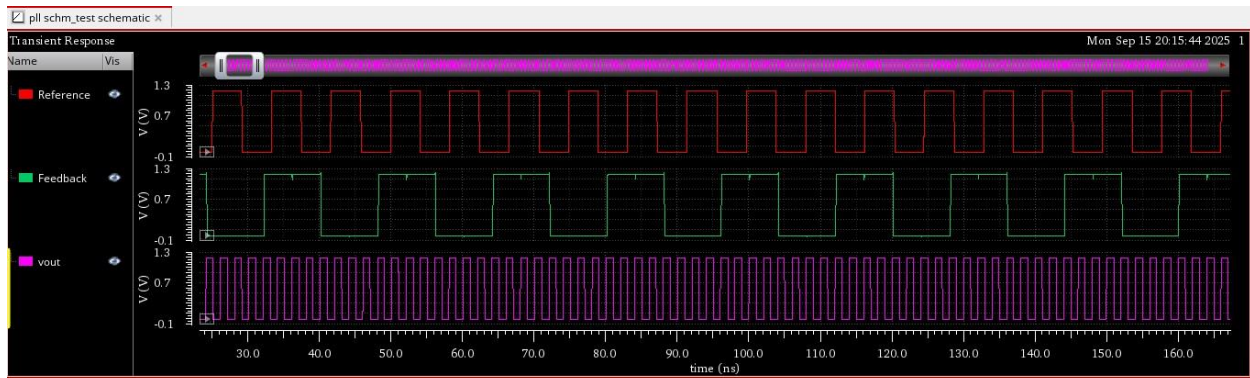
Cell Bindings				
Library	Cell	View Found	View To Use	Inherited View List
analogLib	vsources	spectre		behavioral ahd functional hspiceS sche...
pll	CHP	veriloga	veriloga	behavioral ahd functional hspiceS sche...
pll	Dff	schematic		behavioral ahd functional hspiceS sche...
pll	Divider2	schematic		behavioral ahd functional hspiceS sche...
pll	PFD	veriloga	veriloga	behavioral ahd functional hspiceS sche...
pll	VCO	veriloga	veriloga	behavioral ahd functional hspiceS sche...
pll	inverter	schematic		behavioral ahd functional hspiceS sche...
pll	schem_test	schematic		behavioral ahd functional hspiceS sche...
pll	test_behavioral	schematic		behavioral ahd functional hspiceS sche...
tsmcN65	nch	spectre		behavioral ahd functional hspiceS sche...

17-

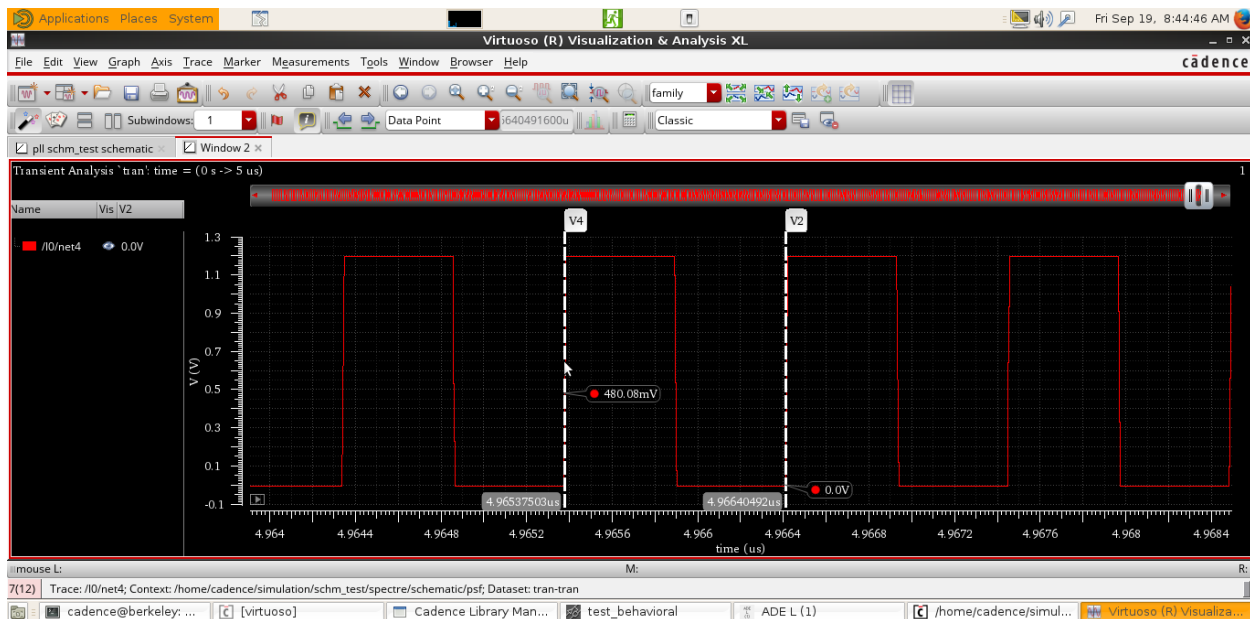
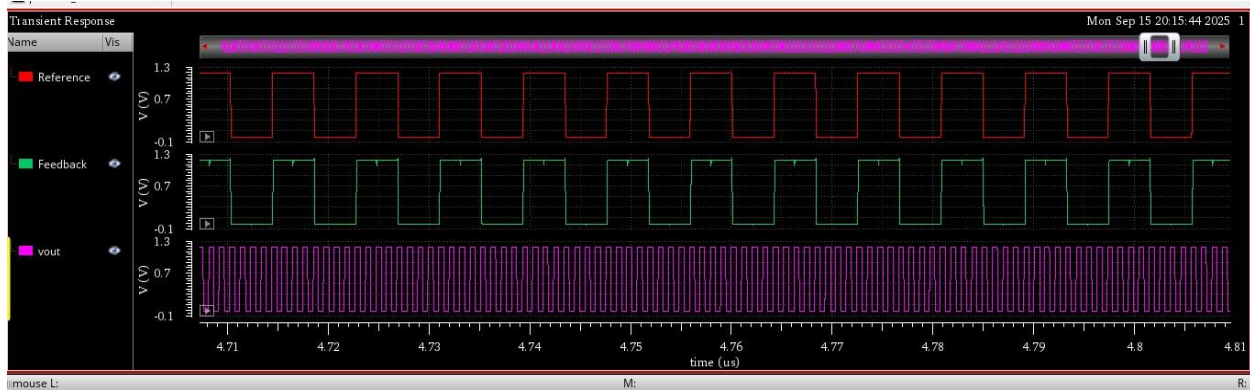


18-

Before lock:

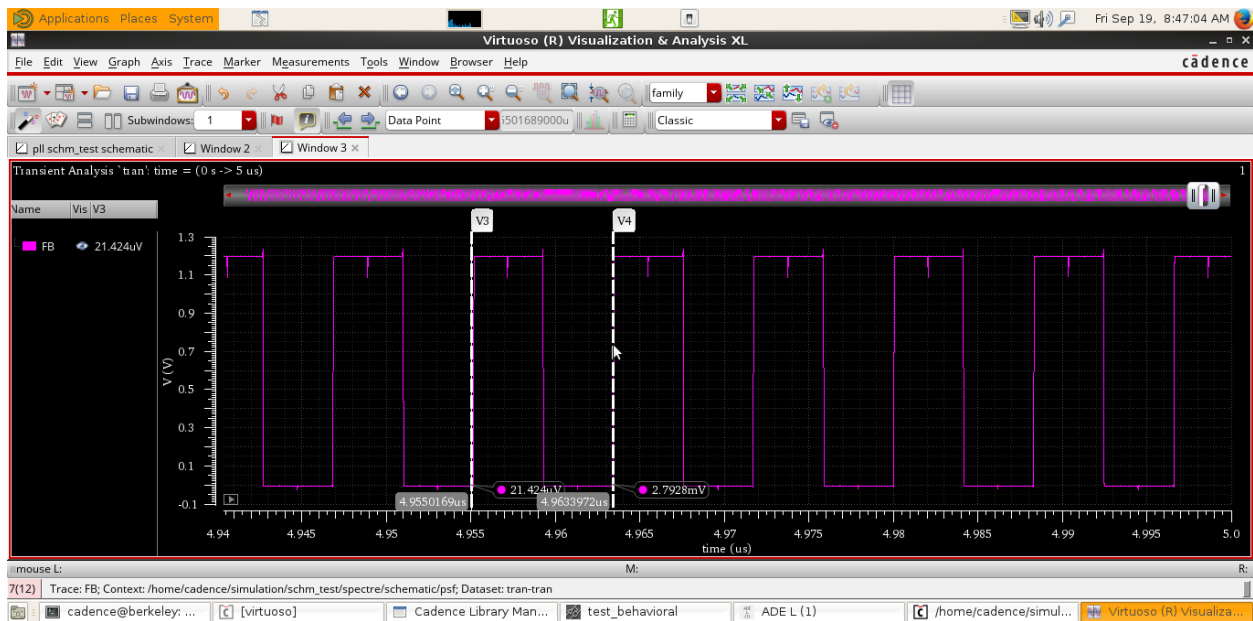


After lock:



Output frequency =  $1/(4.96640492\mu\text{s} - 4.96537503\mu\text{s}) = 970.48 \text{ MHz}$

(not exactly 960MHz because our source period is not accurate 8.3ns)



Feedback lock frequency =  $1/(4.9633972\mu\text{s} - 4.9550169\mu\text{s}) = 119.4 \text{ MHz}$  (not exactly 120MHz ).



19-

```
generate netlist...
Initializing the control file using cp:
  cp /usr/local/cadence/IC617/tools.lnx86/dfII/etc/si/control.spectre /home/cadence/simulation/schm_test/spectre/schematic/netlist/control
Copying Spectre source file 'spectre.inp'
Copying Spectre command file 'spectre.sim'
Begin Incremental Netlisting Sep 15 20:13:00 2025
End netlisting Sep 15 20:13:00 2025
```

```
Netlisting Statistics:
  Number of components: 23

  Elapsed time: 0.0s
Errors: 0 Warnings: 0
  ...successful.
compose simulator input file...
  ...successful.
start simulator if needed...
  ...successful.
simulate...
INFO (ADE-3071): Simulation completed successfully.
reading simulation data...
  ...successful.
```

---

#### Post-Transient Simulation Summary

- To further speed up simulation, consider
  - add ++aps on command line
  - add +cktpreset=sampled on command line for ADC/DAC simulation
  - add +cktpreset=pll on command line for PLL simulation

---

```
Initial condition solution time: CPU = 36 ms, elapsed = 37.554 ms.
Intrinsic tran analysis time: CPU = 161.912 s, elapsed = 162.161 s.
Total time required for tran analysis 'tran': CPU = 161.956 s (2m 42.0s), elapsed = 162.207 s (2m 42.2s).
Time accumulated: CPU = 163.308 s (2m 43.3s), elapsed = 163.619 s (2m 43.6s).
Peak resident memory used = 234 Mbytes.
```

```
Notice from spectre.
29875 notices suppressed.
```

```
finalTimeOP: writing operating point information to rawfile.
modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.
```

---

20-

Verilog-A (behavioral model )	Transistor level model
10 s	163.6 s

As expected, the transistor-level simulation requires more time compared to the behavioral-level simulation