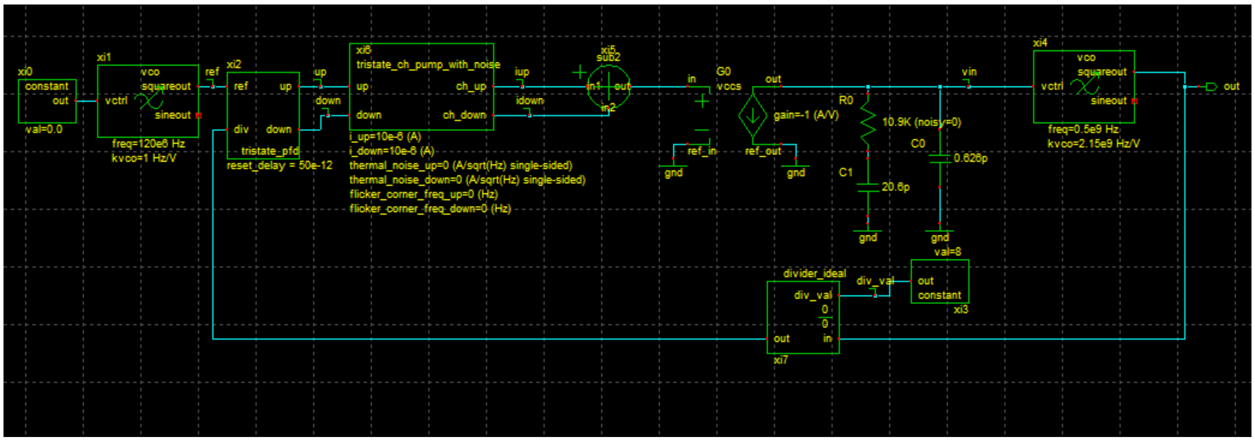
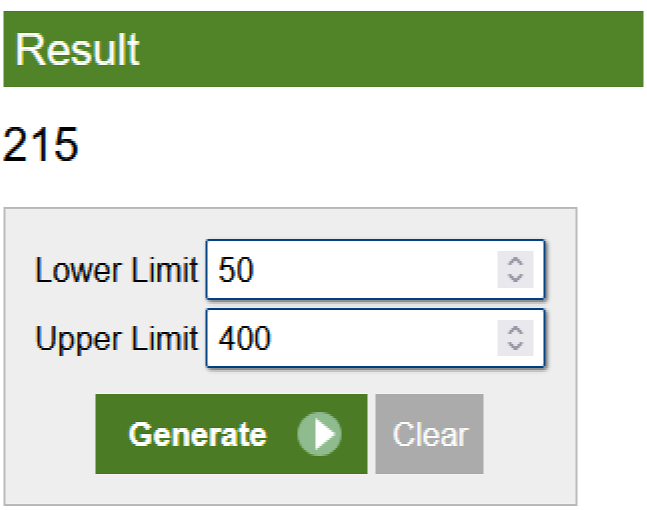
Lab6

Part 1

1-



2-



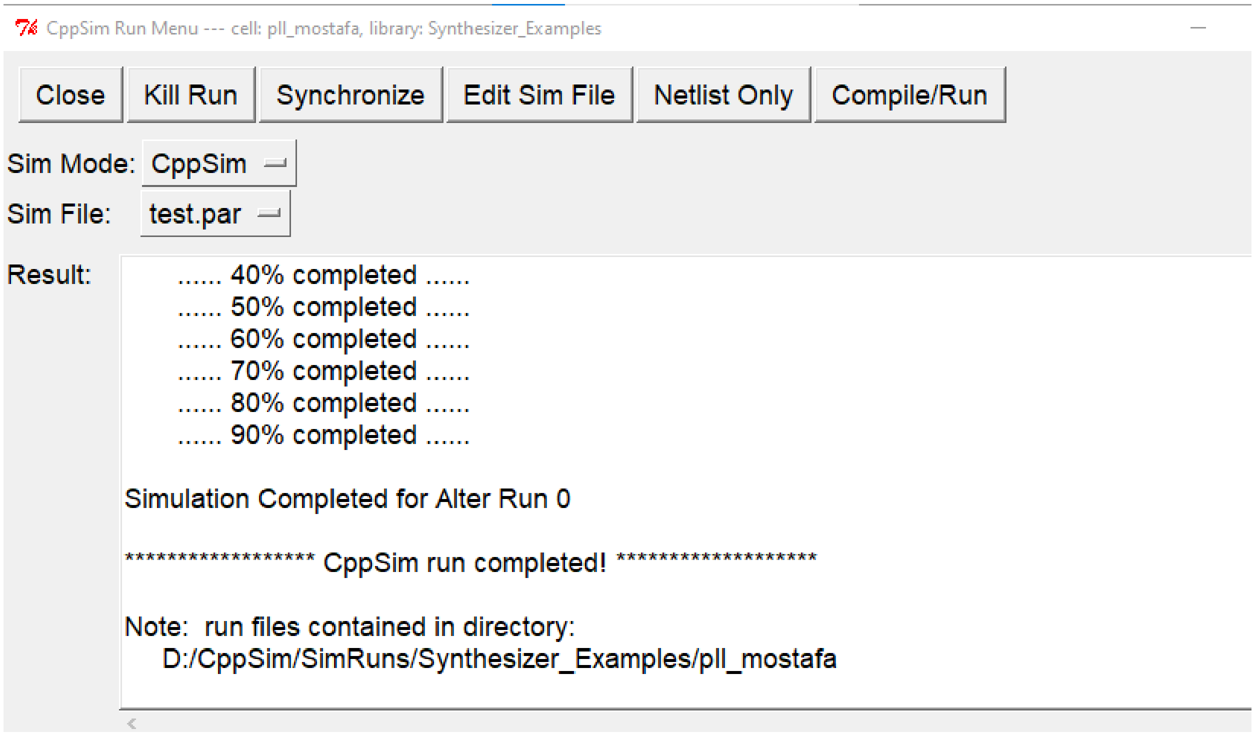
KVCO = (215 / 100)*1e9 = 2.15e9

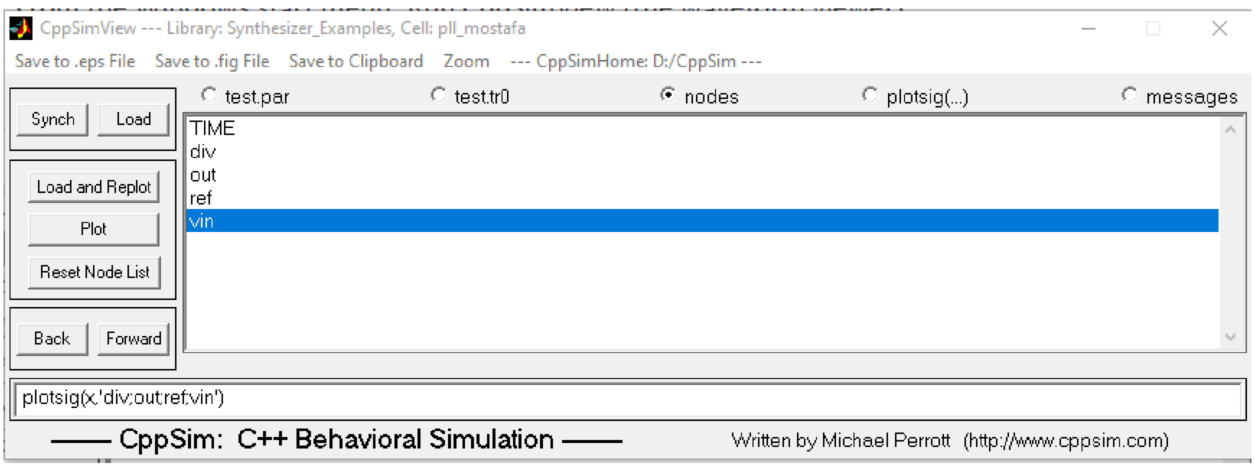
3- f(out) = f(in) + Kvco *vin

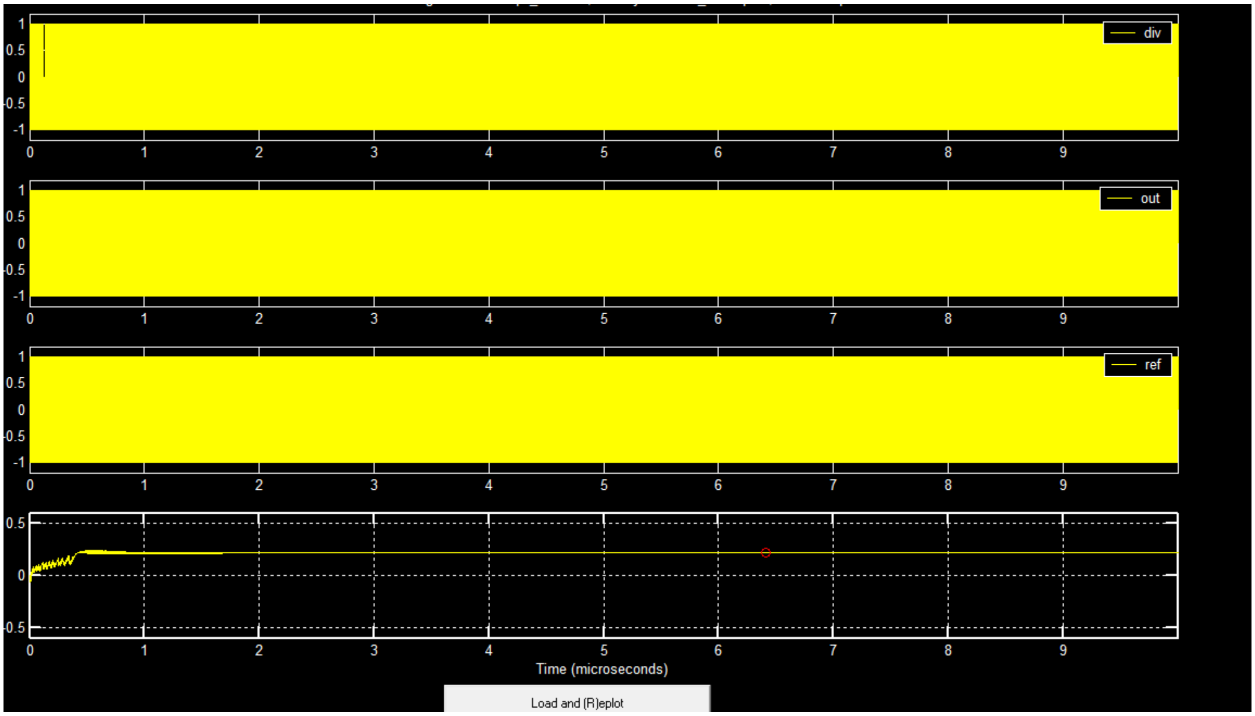
8*120e6 = 0.5e9 + 2.15e9*vin

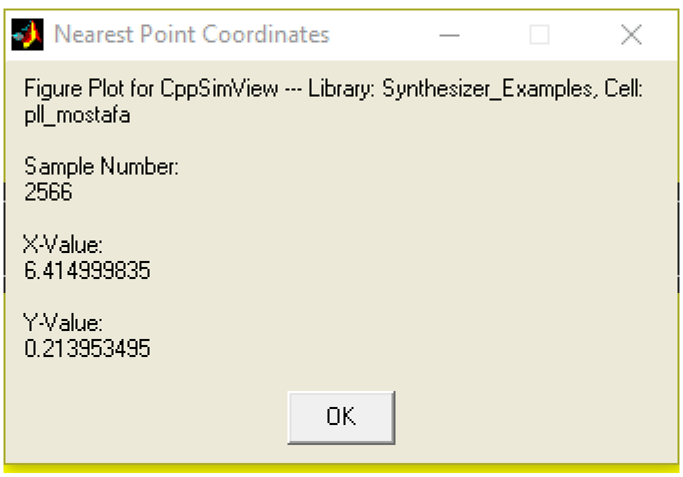
Vin = 0.2139 V

4-


CppSim Run Menu --- cell: pll_mostafa, library: Synthesizer_Examples
Close
Kill Run
Synchronize
Edit Sim File
Netlist Only
Compile/Run
Sim Mode: CppSim
Sim File: test.par
Result:
...... 40% completed ......
...... 50% completed ......
...... 60% completed ......
...... 70% completed ......
...... 80% completed ......
...... 90% completed ......
Simulation Completed for Alter Run 0
****************** CppSim run completed! *******************
Note: run files contained in directory:
D:/CppSim/SimRuns/Synthesizer_Examples/pll_mostafa


CppSimView --- Library: Synthesizer_Examples, Cell: pll_mostafa
Save to .eps File
Save to .fig File
Save to Clipboard
Zoom
--- CppSimHome: D:/CppSim ---
Synch
Load
Load and Replot
Plot
Reset Node List
Back
Forward
test.par
test.tr0
nodes
plotsig(...)
messages
TIME
div
out
ref
vin
plotsig(x,'div;out;ref;vin')
CppSim: C++ Behavioral Simulation
Written by Michael Perrott (http://www.cppsim.com)

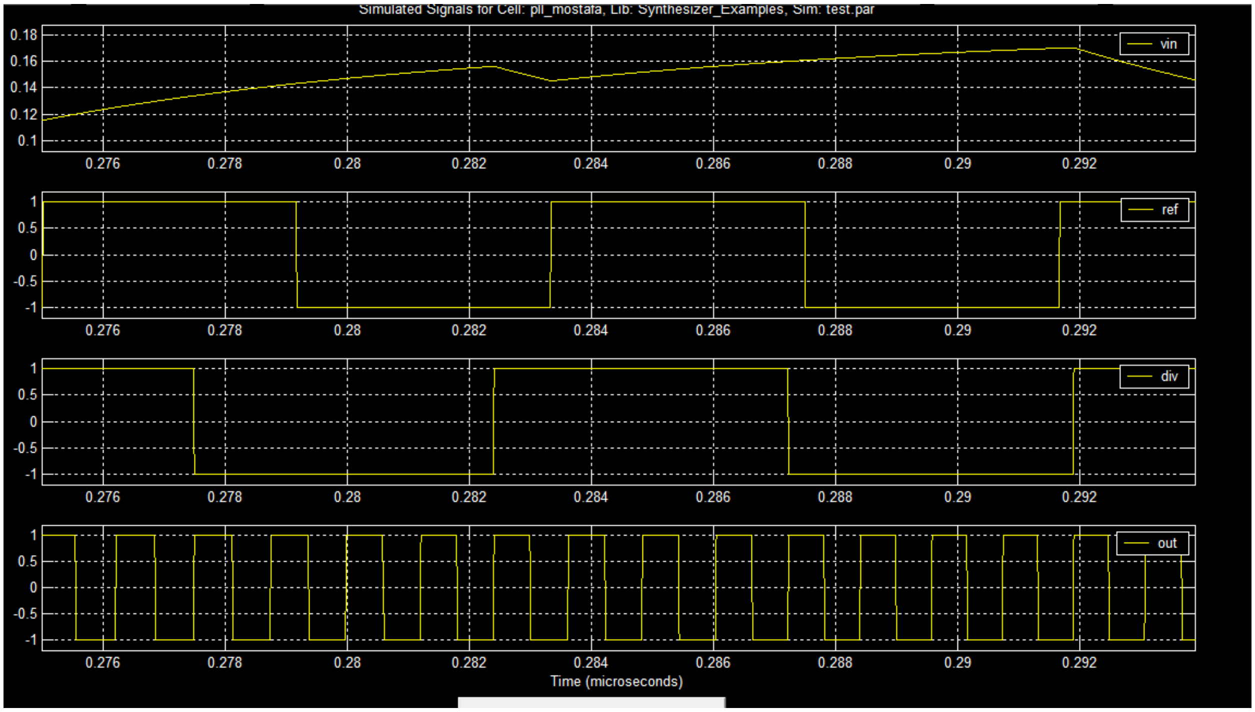




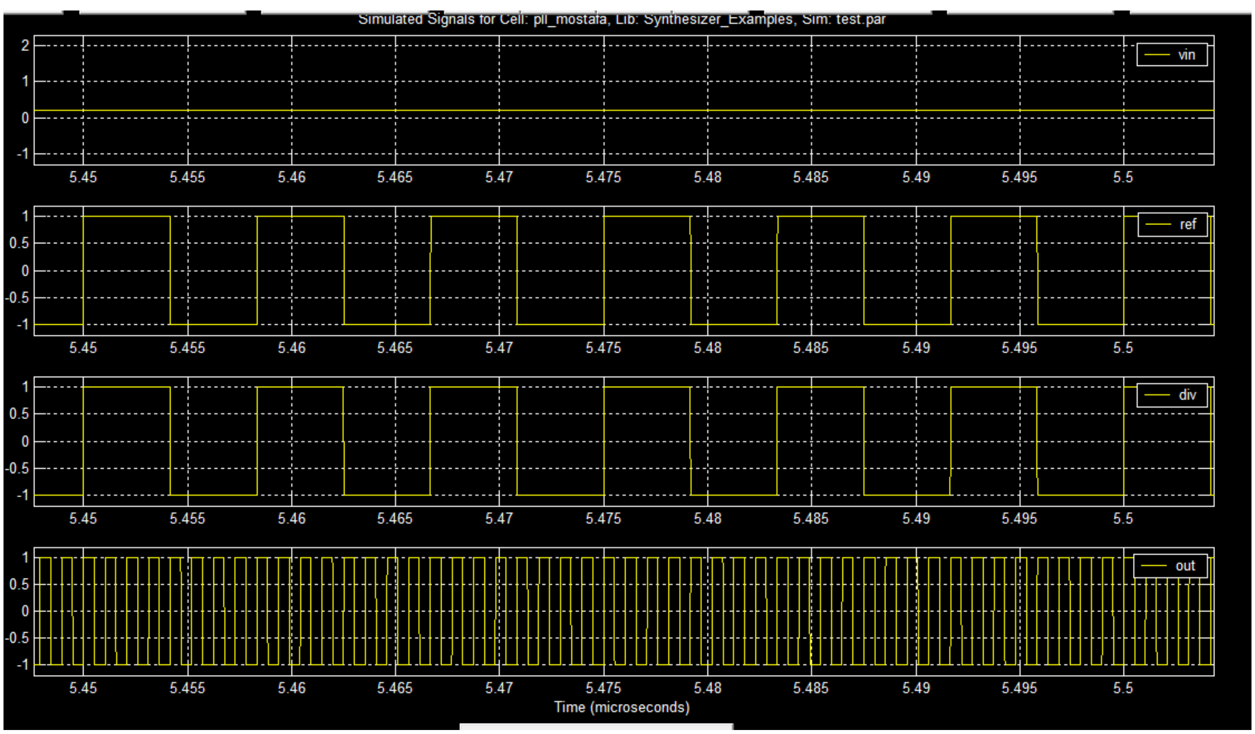
	Analytically	CppSim
Vin	0.2139	0.21395

We can observe that the value of Vin(Vctrl) from simulation is exactly the same as the value obtained analytically .

Before lock :

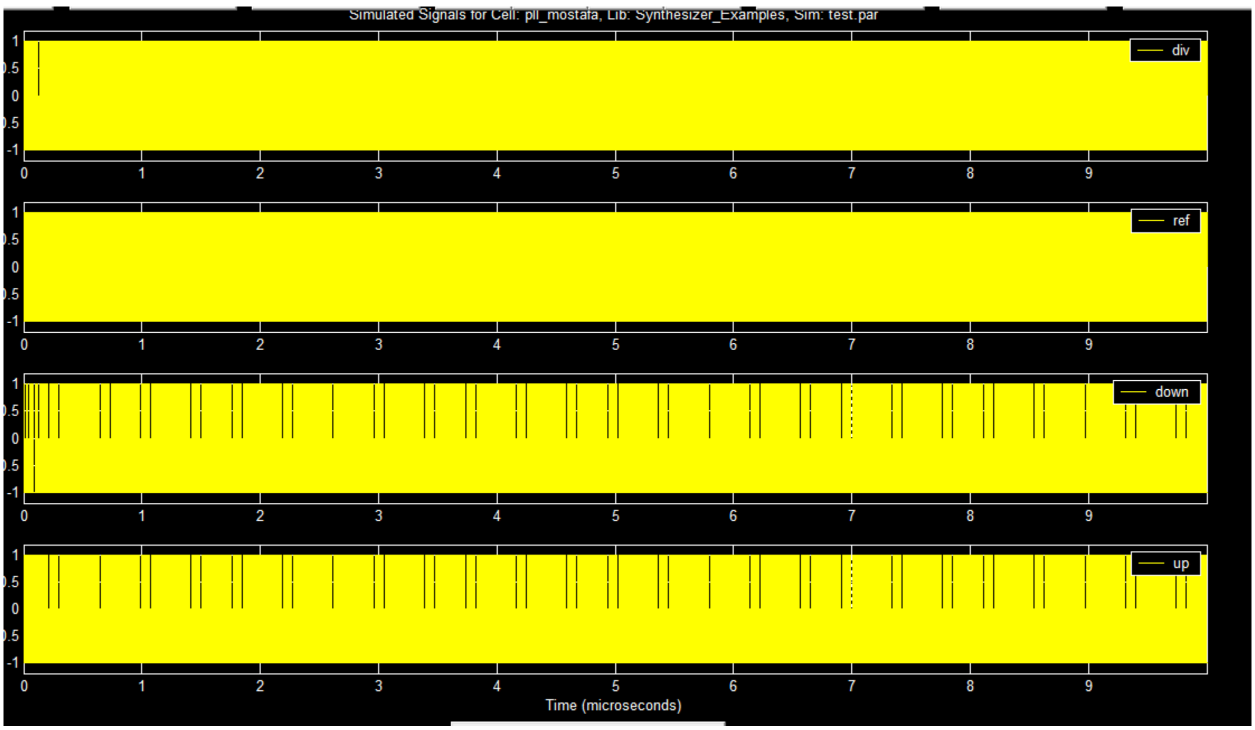


After lock:

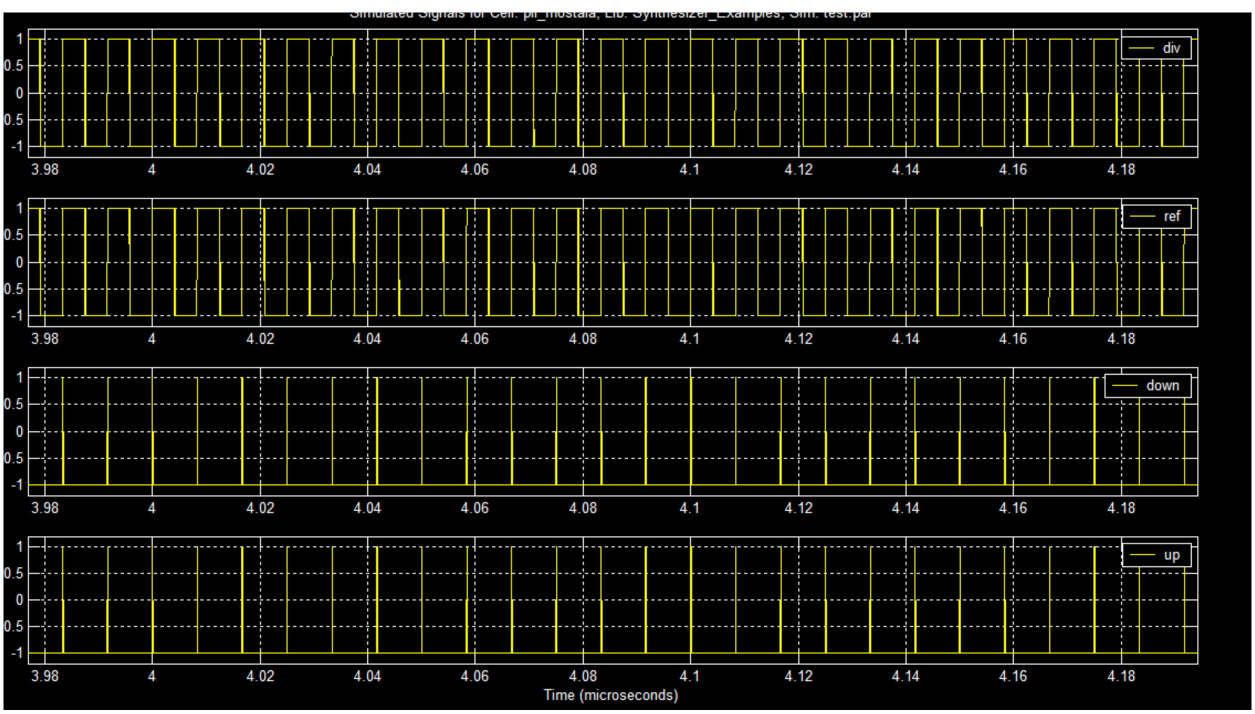


Part 2

1-

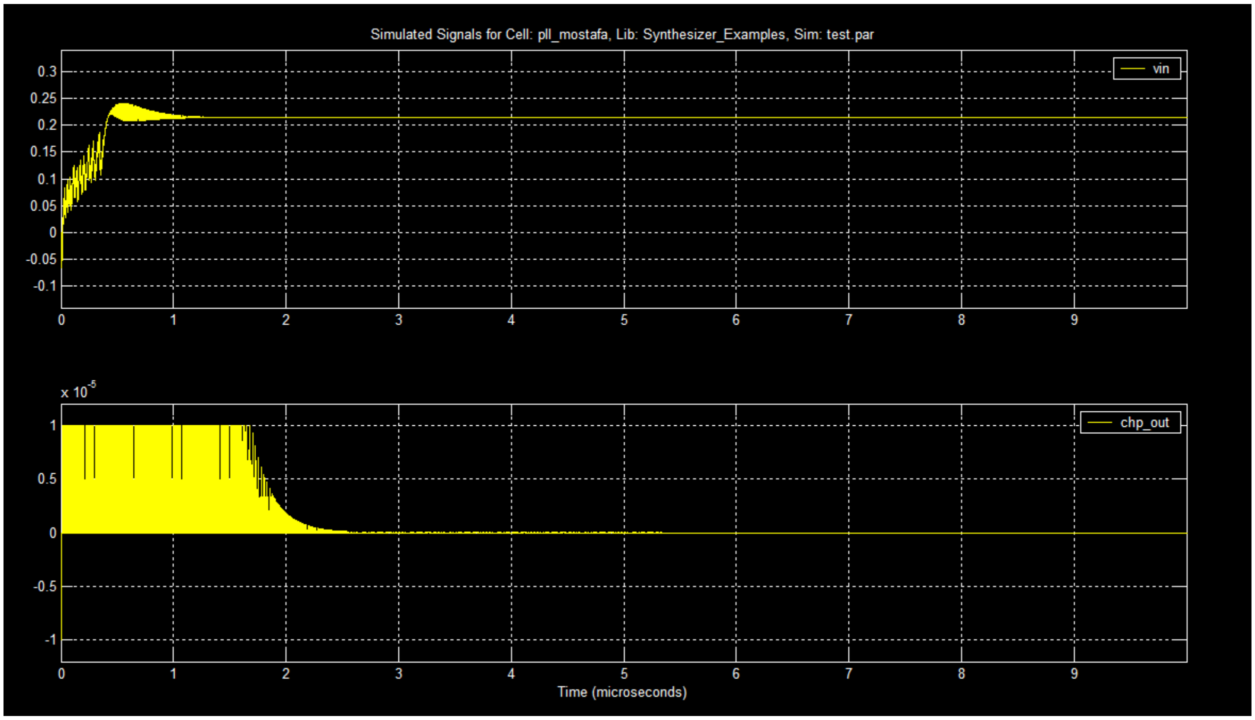


Simulated Signals for Cell: pll_mostafa, Lib: Synthesizer_Examples, Sim: test.par
div
ref
down
up
Time (microseconds)

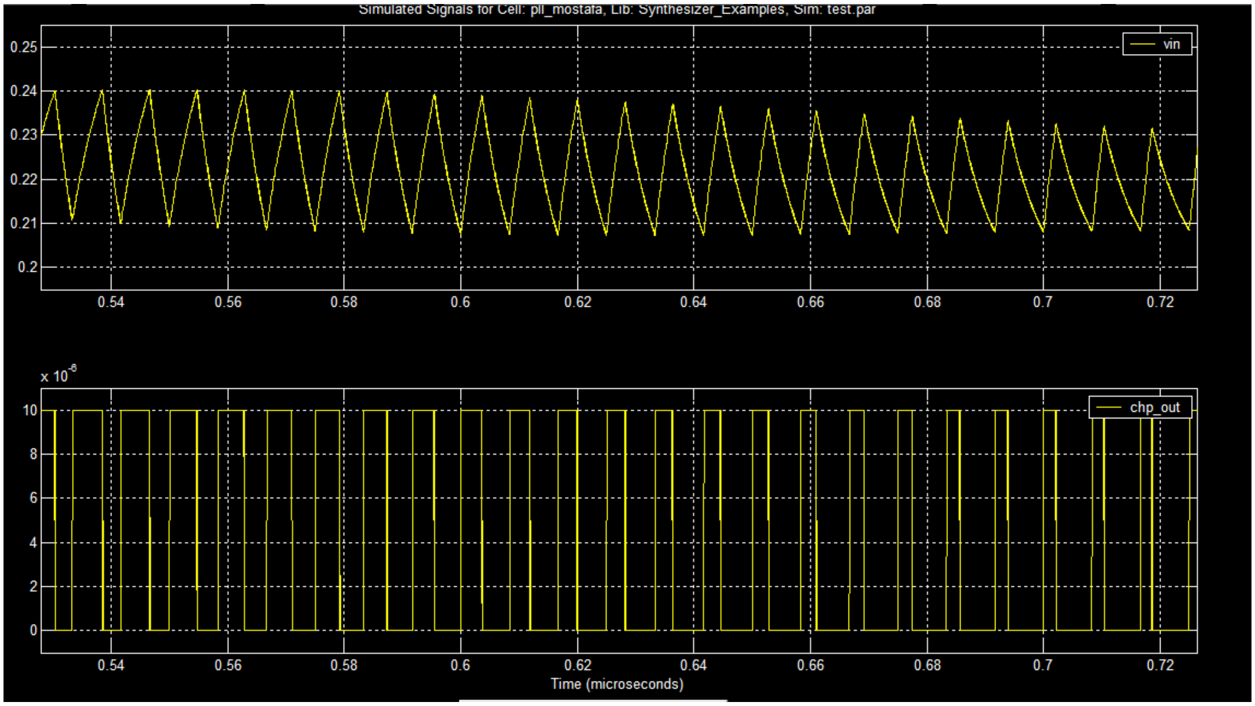


div
ref
down
up
Time (microseconds)

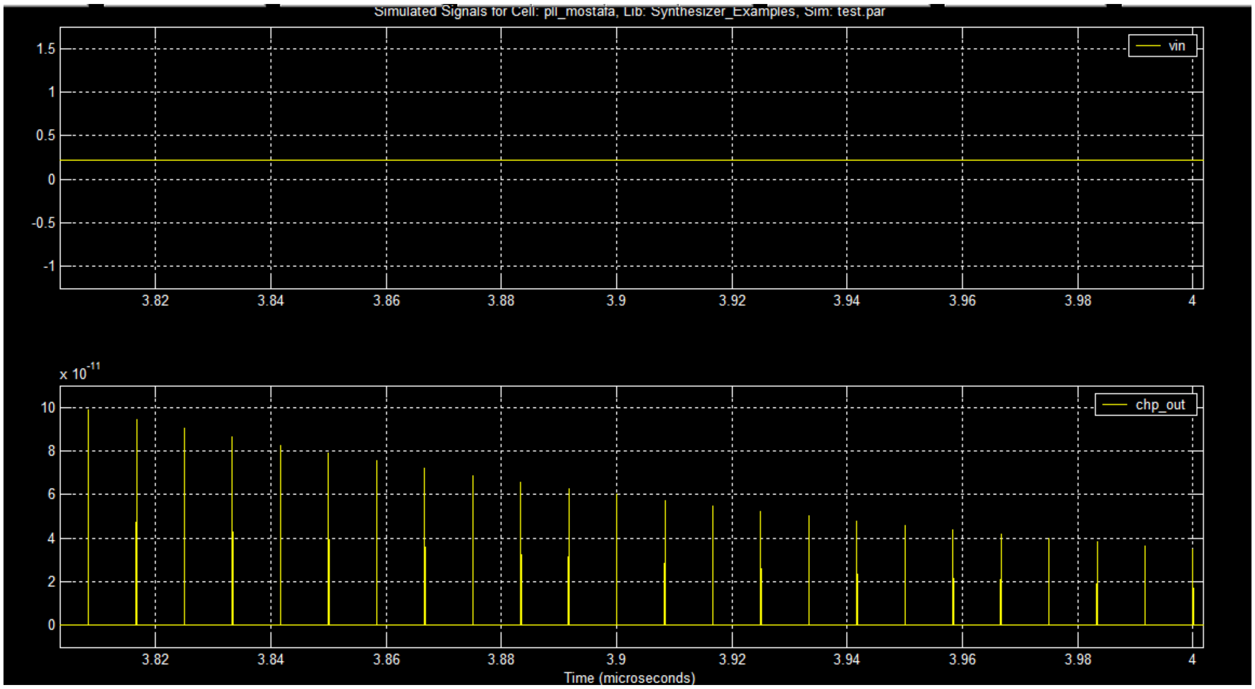
2-



Before lock:

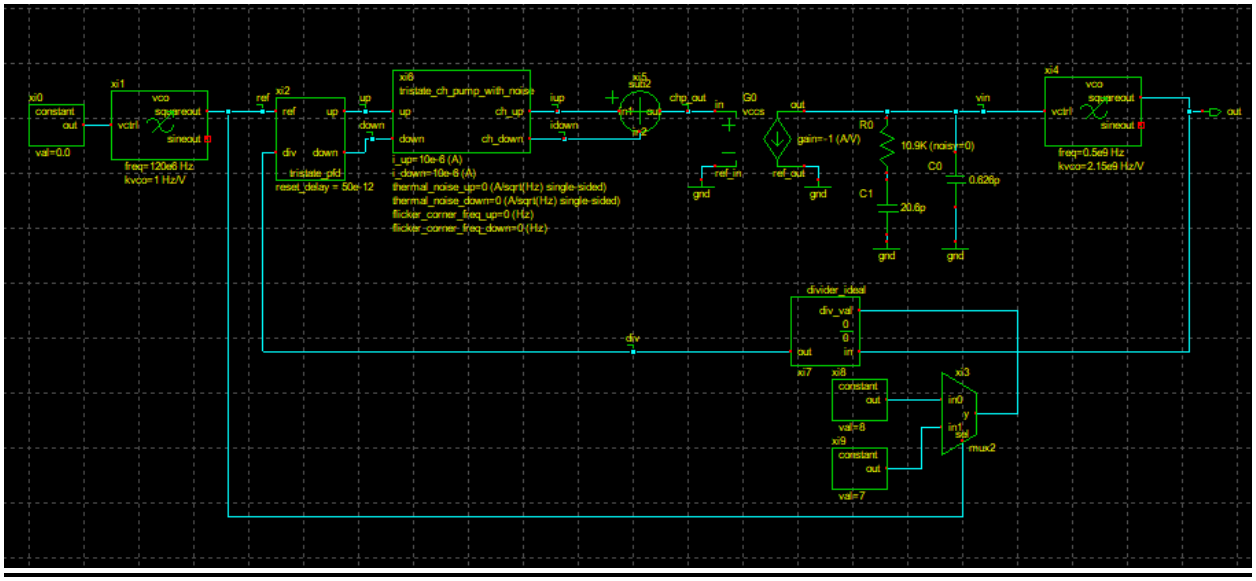


After lock:

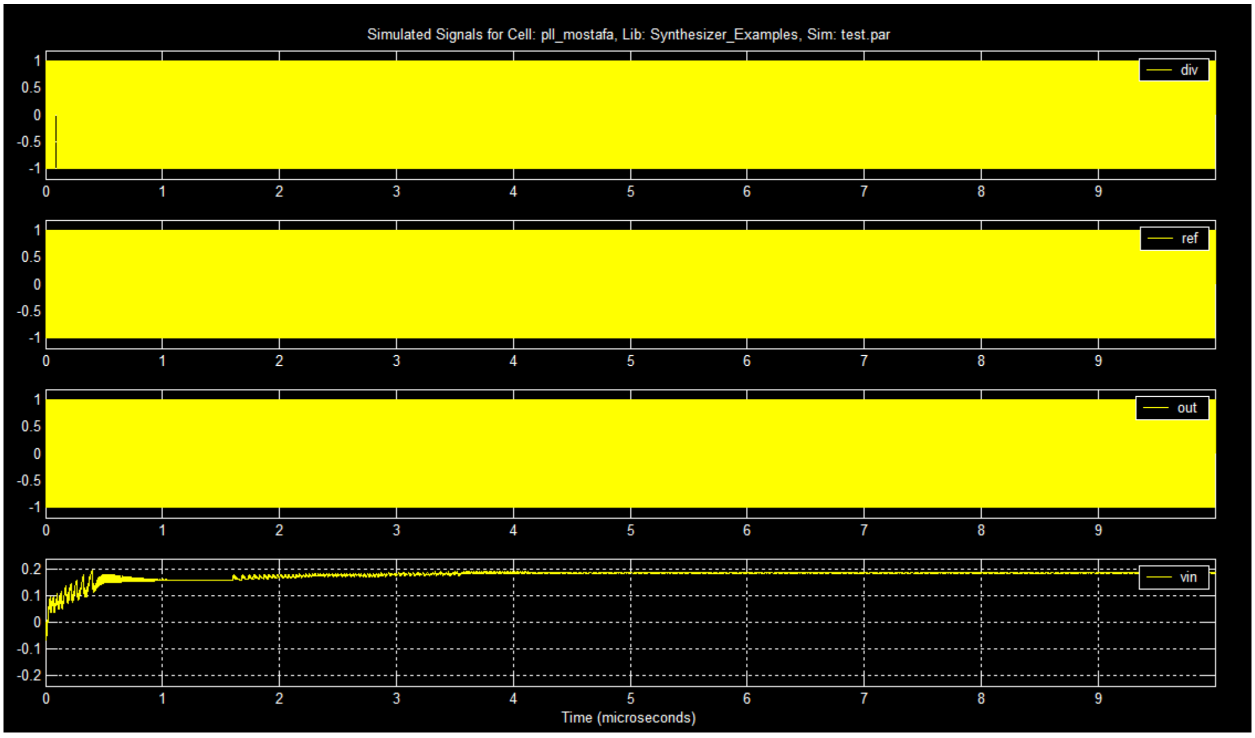


Simulated Signals for Cell: pll_mostafa, Lib: Synthesizer_Examples, Sim: test.par
vin
1.5
1
0.5
0
-0.5
-1
3.82
3.84
3.86
3.88
3.9
3.92
3.94
3.96
3.98
4
x 10^-11
chp_out
10
8
6
4
2
0
Time (microseconds)

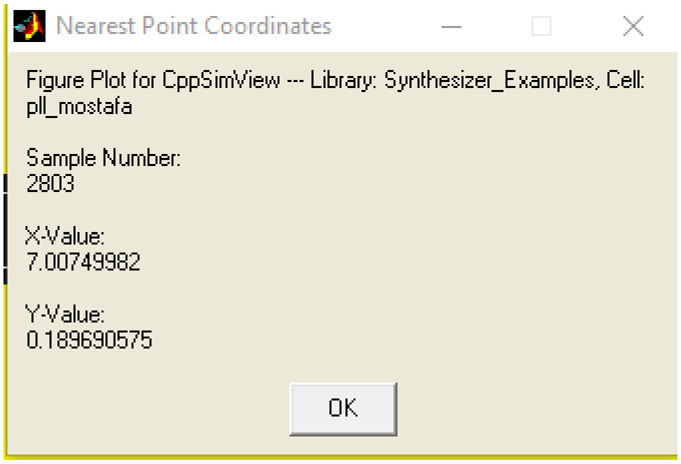
Part 3



constant
val=0.0
vco
freq=120e6 Hz
kvco=1 Hz/V
tristate_pfd
reset_delay = 50e-12
tristate_ch_pump_with_noise
i_up=10e-6 (A)
i_down=10e-6 (A)
thermal_noise_up=0 (A/sqrt(Hz) single-sided)
thermal_noise_down=0 (A/sqrt(Hz) single-sided)
flicker_corner_freq_up=0 (Hz)
flicker_corner_freq_down=0 (Hz)
gain=-1 (A/V)
10.9K (noisy=0)
20.6p
0.626p
freq=0.5e9 Hz
kvco=2.15e9 Hz/V
divider_ideal
val=8
val=7
mux2



Simulated Signals for Cell: pll_mostafa, Lib: Synthesizer_Examples, Sim: test.par
div
ref
out
vin
Time (microseconds)



f(out) -f(in) = Kvco * vin

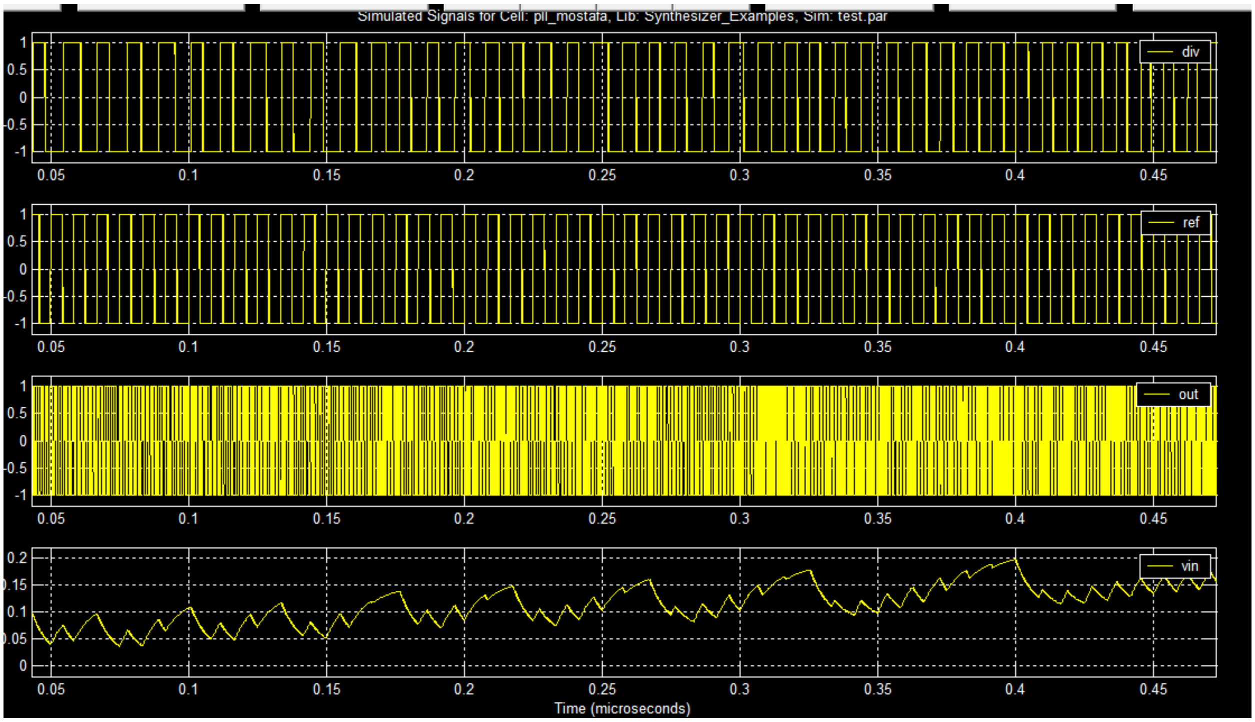
7.5*120e6 = 0.5e9 + 2.15e9*vin

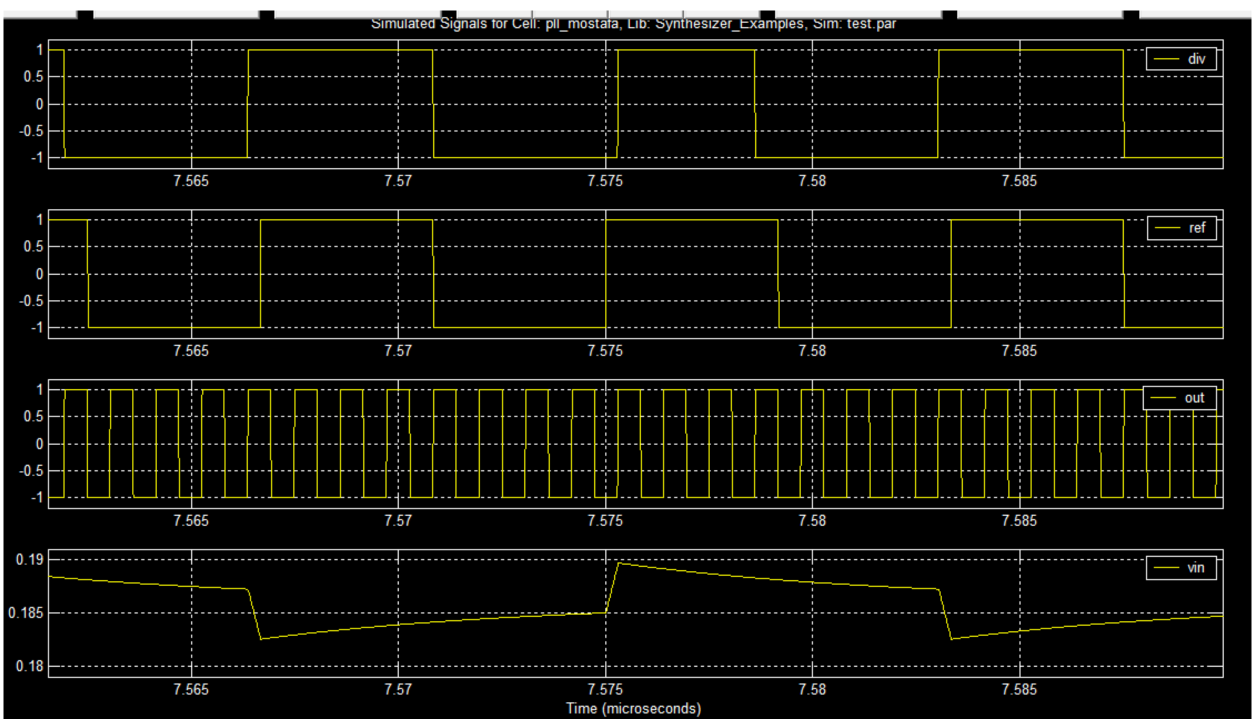
Vin = 0.186 V

	Analytically	CppSim
Vin	0.186	0.18969

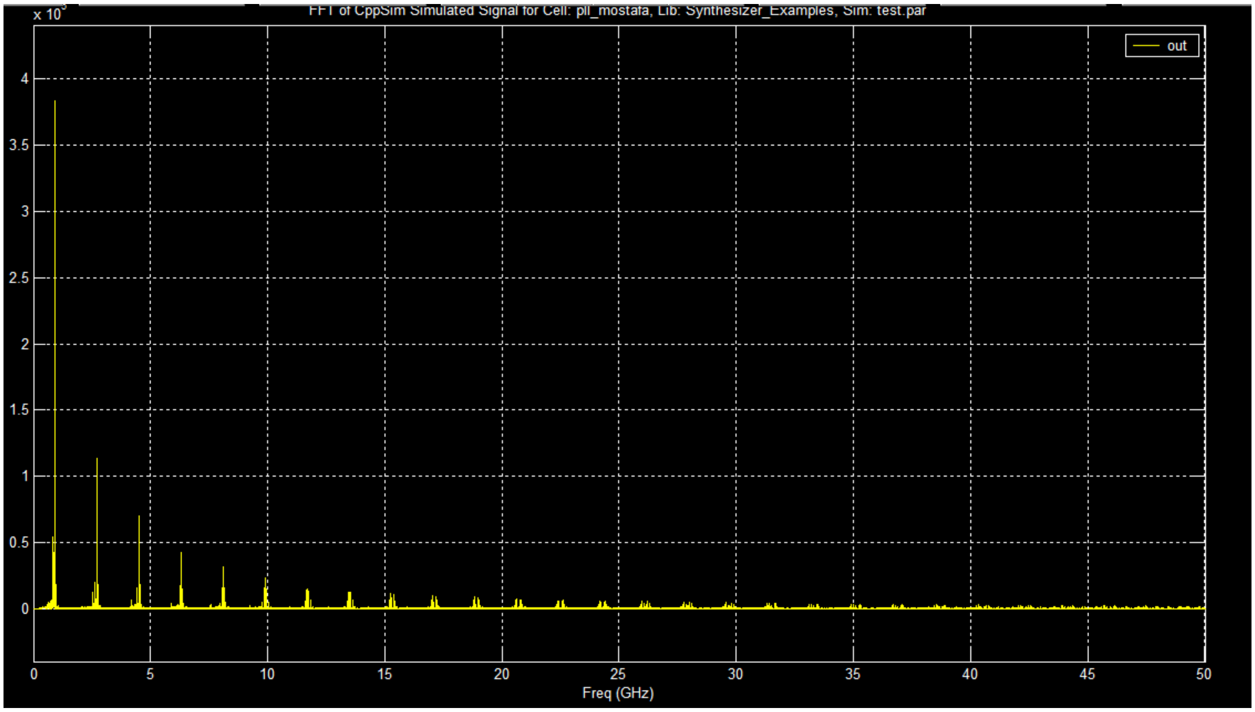
We can observe that the value of Vin(Vctrl) from simulation is near the value obtained analytically but not exactly the same because the Vin value isn't stable due to changing the divider value between 7 and 8 .

Using mux to divide over two different values using the same divider made Vin not stable but gave us the effect of using fractional N divider so we substituted with N = 7.5 in the equation .


Simulated Signals for Cell: pll_mostafa, Lib: Synthesizer_Examples, Sim: test.par
div
ref
out
vin
Time (microseconds)


Simulated Signals for Cell: pll_mostafa, Lib: Synthesizer_Examples, Sim: test.par
div
ref
out
vin
Time (microseconds)

<u>3-</u>



1- We can see the main tone at 0.96 GHZ which aligns perfectly with the expected locked VCO frequency:

fvco=N*fref=8*120 MHz=960 MHz

2- there is additional odd harmonics with decreasing power with high frequencies .
These are harmonics of the square-wave VCO output and because we are alternating between two divider values which introduces some unwanted spurs