

LOW POWER, SMALL AREA ALL DIGITAL PHASE LOCKED LOOP (ADPLL)

By

Ammar Mohammad Ibrahim

Ammar Mohammad Hussein

Mohammad Abdel-Lateef Abdel-Tawab

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Electronics and Electrical Communications Department, Faculty of Engineering

Cairo University

Supervised by Dr. Hassan Mostafa

July, 2014

ABSTRACT

ADPLL

The objective of the thesis is to design an All Digital Phase Locked Loop (ADPLL) with low power. The design consists of three main blocks: Digitally Controlled Oscillator (DCO), Phase Detector (PD) and Loop Filter (LF). The DCO is considered as the heart of the ADPLL as it consumes the most power for the whole system. The design went through two different approaches, standard cells and custom cells .This design can be used in Clock and Data Recovery (CDR) system as an application.

This thesis presents a low power all digital phase locked loop (ADPLL) in 65 nm CMOS process with 1.2 V power supply. It operates in the frequency range of 100 - 300 MHz. The ADPLL uses a digitally controlled oscillator with two stages, fine tuning stage and coarse tuning stage. The source of oscillation for this DCO is the ring oscillator.

The proposed ADPLL uses also a phase-frequency detector (PFD) and shift registers for the loop filter. It achieved power consumption at 200 MHz of 0.6 mW and a lock time of 1 uS.

The last design step of the ADPLL is the layout, some modification applied to the layout to satisfy the required specifications, at the end of this thesis a comparison between the required and the achieved specifications in schematic and layout level.

Design considerations of the ADPLL circuit components and implementation using Cadence, Synopsys and Mentor tools are presented; the AMS tool is used frequently in the standard cells flow.

TABLE OF CONTENTS

List of Figures	
Acknowledgments	
Acronyms	
Chapter 1: Introduction	1
Chapter 2: Custom Cells Approach	5
2.1 DCO	
2.1.1 Ring Oscillator	6
2.1.2 Fine stage	11
2.1.3 Coarse stage	
2.1.4 Conclusion and final results	
2.2 PFD	24
2.3 Loop Filter	
2.4 Overall Design	
2.4.1 Extreme Reference Frequency	
2.4.2 Intermediate Reference Frequency	
2.4.3 Frequency step response	
2.4.4 Jitter calculation	
Chapter 3: Standard Cells Approach	
3.1 PFD	
3.2 Loop Filter	
3.3 PFD and Loop Filter	
3.3 Overall Design	
Chapter 4: Layout	
4.1 DCO	
4.1.1 DCV	
4.1.2 DCV2	
4.1.3 Ring Oscillator	45
4.1.4 Complete DCO	
4.2 PFD	
4.3 Loop Filter	
4.4 Overall Design	
4.5 Conclusion	51
References	
Appendix A: AMS Tutorial	53
Appendix B: Logic Synthesis	
Appendix C: Importing Synthesized Design into Cadence Composer	
Appendix D: Standard Cell Placement and Routing	
Appendix E: Power Calculation	

LIST OF FIGURES

Figure 1.1: Block diagram of the PLL	1
Figure 1.2: The Overall block diagram of the ADPLL	2
Figure 1.3: Jitter Illustration	
Figure 2.1: DCO symbol view	5
Figure 2.2: DCO internal structure	6
Figure 2.3: Ring oscillator block diagram	6
Figure 2.4: A Schematic view for the inverter	7
Figure 2.5: Simulation results of the inverter	
Figure 2.6: Simulation results for the ring oscillator without delay cells	
Figure 2.7: A schematic view of the HDC	9
Figure 2.8: The ring oscillator with HDC cells	
Figure 2.9: Simulation results for the ring oscillator with HDC cells	10
Figure 2.10: A schematic view for the DCV cell	11
Figure 2.11: The gate capacitance of NAND gate	12
Figure 2.12: Simulation results for the NAND based DCV Cell	13
Figure 2.13: A block diagram for the DCV building block	13
Figure 2.14: DCV array internal structure	
Figure 2.15: A symbol view for the DCV array	15
Figure 2.16: Period step Vs. frequency steps	16
Figure 2.17: A schematic view for the DCV2 cell	17
Figure 2.18: DCV2 block	
Figure 2.19: DCV2 array	19
Figure 2.20: A symbol view for the DCV2 array	20
Figure 2.21: Different output periods of the DCO	21
Figure 2.22: Different output waveforms of the DCO	22
Figure 2.23: Period steps versus code	22
Figure 2.24: Frequency steps versus code	23
Figure 2.25: PFD schematic	24
Figure 2.26: State diagram of the PFD	25
Figure 2.27: PFD simulation results	25
Figure 2.28: Digital control signals used to switch a set of varactors	26
Figure 2.29: Schematic view of the shift register	27
Figure 2.30: Reference = 100MHz	
Figure 2.31: DCO delay	
Figure 2.32: Reference= 250MHz	29
Figure 2.33: Effect of using counter	
Figure 2.34: Frequency counter	31
Figure 2.35: frequency step response	32
Figure 2.36: Eliminating the oscillations	
Figure 2.37: Eye-diagram of the ADPLL	
Figure 3.1: PFD schematic	
Figure 3.2: Loop Filter schematic	
Figure 3.3: PFD and Loop Filter	
Figure 3.4: Overall ADPLL (standard cells)	

Figure 3.5: AMS of the overall ADPLL without the counter	
Figure 3.6: AMS simulation of the overall ADPLL with counter	
Figure 3.7: AMS of the ADPLL with counter	
Figure 3.8: The standard cells of both PFD and Loop Filter	41
Figure 3.9: The transistor level of the standard D flip flop in cadence.	
Figure 4.1: DCV layout	
Figure 4. 2: DCV block layout	
Figure 4. 3: DCV array layout	
Figure 4. 4: DCV2 layout	44
Figure 4. 5: DCV2 block layout	45
Figure 4. 6: DCV2 array layout	45
Figure 4. 7: Ring Oscillator layout	45
Figure 4. 8: Final DCO layout	
Figure 4. 9: DCO operating range in layout	47
Figure 4. 10: PFD layout	47
Figure 4.11: PFD pre-layout simulation	
Figure 4.12: PFD post-layout simulation	
Figure 3.13: Shift Register layout	49
Figure 4.14: Pre-layout simulation of the Shift Register	49
Figure 4.15: Post-layout simulation of the Shift Register	
Figure 4.16: The overall layout	
Figure 4.17: Overall post layout simulation	

ACKNOWLEDGMENTS

First of all we would like to thank ALLAH for his mercy supporting us through this project. We wish to express sincere appreciation to **Dr. Hassan Mostafa** for his assistance in the preparation of this project. In addition, special thanks to our teaching assistance **Eng. Mahmoud Nagib Sawaby** for helping us with the needs and ideas of the standard cells approach and to **Eng. Mohammad Wagih Emam** for his time and effort for answering our questions and helping deal with Cadence toolkit. Thanks also to our friend **Eng. Islam Abdou** for installing the AMS tool in Cadence.

ACRONYMS

ADPLL	All Digital Phase Locked Loop
AMS	Analog Mixed Simulation
DCO	Digitally Controlled Oscillator
DCV	Digitally Controlled Varactor
DFF	D Flip Flop
DPLL	Digital Phase Locked Loop
HDC	Hysteresis Delay Cell
LF	Loop Filter
LPLL	Linear Phase Locked Loop
PD	Phase Detector
PFD	Phase Frequency Detector
PLL	Phase Locked Loop
SR	Shift Register
VCO	Voltage Controlled Oscillator

Chapter 1

INTRODUCTION

The PLL represents one of the most active topics in signal processing and communication theory. The initial ideas started as early as 1919 in the context of synchronization of oscillators. The theory of phase-locked loop was based on the theory of feedback amplifiers. The PLL contributed significantly to communications and motor servo systems. Due to the rapid development of integrated circuits (IC's) since the 1970's, PLLs are widely used in modern signal processing and communication systems, and it is expected that PLL will contribute to improvement in performance and reliability of future communication systems. The applications of PLLs include filtering, frequency synthesis, motor speed control, frequency modulation, demodulation, signal detection, frequency tracking and many other applications.

The PLL consists of three main blocks VCO, Loop Filter and Phase detector as shown in figure 1.1.

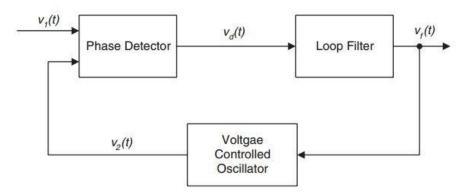


Figure 1.1: Block diagram of the PLL

There are many types of PLL according to the internal blocks and designing techniques as following:

- LPLL: Linear Phase Locked Loop which contains a VCO and RC circuit for the Loop Filter block and uses a multiplier to detect the phase difference between the reference frequency and the VCO output frequency.
- DPLL: Digital Phase Locked Loop was the very first digital PLL; it was in effect a hybrid device ONLY the phase detector was built as a digital block like EXOR.
- 3. ADPLL: All Digital Phase Locked Loop in which all the blocks are built as digital blocks.

ADPLL consists of the same three main blocks mentioned previously except for the VCO; it will be replaced by the DCO as shown in figure 1.2

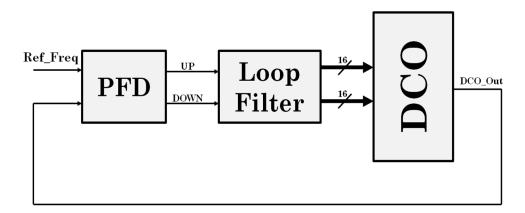


Figure 1.2: The Overall block diagram of the ADPLL

PLL in general has its own parameters such as:

- The operating frequency range: the range of frequencies that PLL can lock on them.
- **The lock time:** the time which PLL needs to lock on the reference frequency.
- **The Jitter:** undesired deviation from the true periodicity of an assumed periodic signal.

What is Jitter?

Jitter is the undesired deviation from true periodicity of an assumed periodic signal, Deviation (expressed in \pm ps) can occur on either the leading edge or the trailing edge of a signal. Jitter may be induced and coupled onto a clock signal from several different sources and is not uniform over all frequencies.

Period of ring oscillator vibrates in a random manner T=T+T where T is a random value. In high-quality circuits range of T is relatively small compared to T. This variation in oscillator period is called jitter. Local temperature effects cause the period of a ring oscillator to wander above and below the long-term average period when the local silicon is cold, the propagation delay is slightly shorter, causing the ring oscillator to run at a slightly higher frequency, which eventually raises the local temperature. When the local silicon is hot, the propagation delay is slightly longer, causing the ring oscillator to run at a slightly lower frequency, which eventually lowers the local temperature.

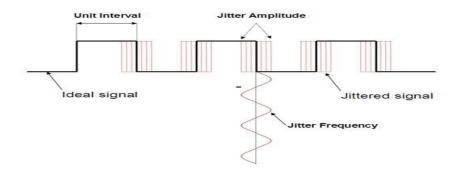


Figure 1.3: Jitter Illustration

We considered the DCO as the first design stage, because it consumes about 50% of the system power and covers the most area of the whole system area. The design went through two approaches the custom cells and the standard cells. The DCO is done in the custom approach however; the rest of the design went through both approaches.

The proposed ADPLL has the following specifications:

- Power $< 1 \ mW$.
- Area < 0.01 mm^2 .
- Frequency Range from 100 MHz to 300 MHz.
- Lock time < 10 μs.
- Peak to Peak Jitter < 20 *ps*.
- R.M.S. Jitter < 5 *ps*.

In the following chapters we are going to discuss the design steps in details for each block of the ADPLL to satisfy these requirements.

Frequently Asked Question about ADPLL:

- Why digital? What is the problem of the analog (linear) one?
 - Basically, the ADPLL consumes less power than the linear PLL.
 - ADPLL can be easily scaled down to another technology.
 - Linear PLL needs an off chip components such as capacitors and resistors (for the loop filter) which do not have a fixed and stable value because they may suffer from aging.
 - ADPLL covers less area than linear PLL.

Chapter 2

CUSTOM CELLS APPROACH

2.1 Digitally Controlled Oscillator

The digitally controlled oscillator (DCO) is considered the heart of the PLL as it controls the overall system performance and consumes the most power and area of the whole design. The proposed DCO follows a full custom design approach to make it easier to control its area and power. It consists of three main blocks:

- 1. A ring oscillator
- 2. A fine tuning stage (DCV Array)
- 3. A Coarse tuning stage (DCV2 Array)

Figure 2.1 is the symbol view of the DCO.

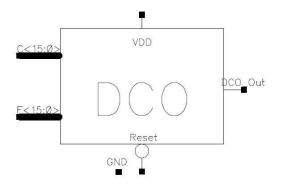


Figure 2.1: DCO symbol view

Another figure for the internal block diagram of the DCO is figure 2.2, figuring out its three main blocks.

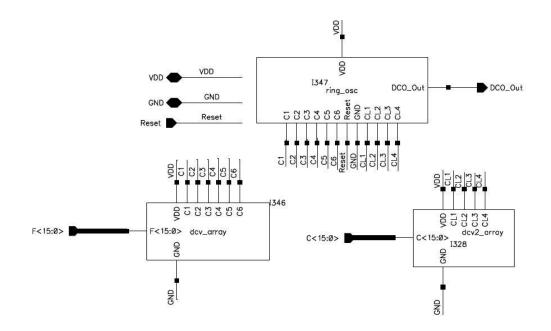


Figure 2.2: DCO internal structure

We will start our discussion by investigating the internal structure of the ring oscillator followed by the fine tuning stage (DCV Array) and finally the coarse tuning stage (DCV2 Array).

2.1.1 Ring Oscillator

The ring oscillator is the source of oscillation for the DCO. It consists of an odd number of inverters in a cascaded configuration with a feedback from the output to the input. Figure 2.3 is a block diagram for the proposed ring oscillator.

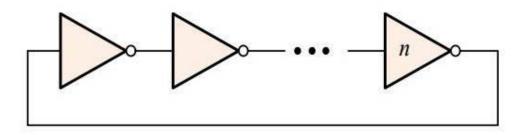


Figure 2.3: Ring oscillator block diagram

The design of the ring oscillator follows a full custom approach. The following is the schematic view (figure 2.4) of the basic cell for the ring oscillator, the inverter.

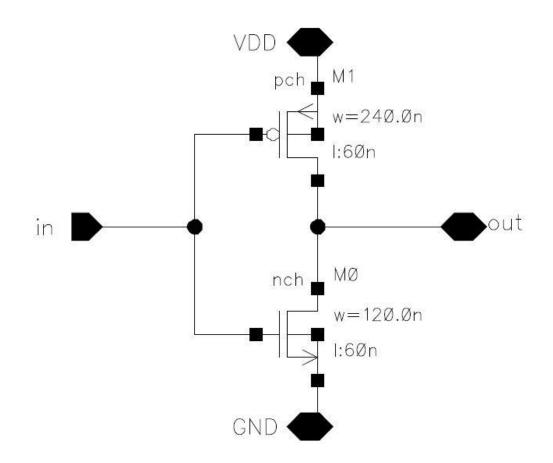


Figure 2.4: A Schematic view for the inverter

The simulation result for the inverter can be found in figure 2.5.

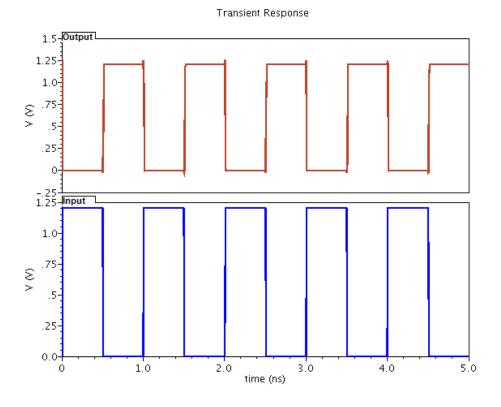


Figure 2.5: Simulation results of the inverter

Next step is to simulate the ring oscillator as cascaded inverters without any delay cells. Simulation result for this step is shown in figure 2.6.

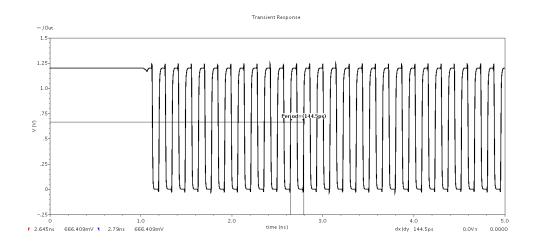


Figure 2.6: Simulation results for the ring oscillator without delay cells

The problem with the simulation results of this ring oscillator is that, the output frequency is in range of GHZ (Period = 144.5 ps) not MHZ and our lock range is from 100 MHZ to 300 MHZ. To solve this problem, we added delay cells to the internal nodes of the ring oscillator. The delay cells to be added to the ring oscillator are Hysteresis Delay Cells (HDC). Each HDC cell consists of two cross coupled inverters. The schematic view of the HDC cell can be found in the figure 2.7.

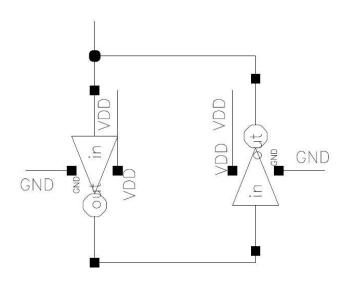


Figure 2.7: A schematic view of the HDC

The schematic view of the ring oscillator with HDC cells attached to it, is in figure 2.8.

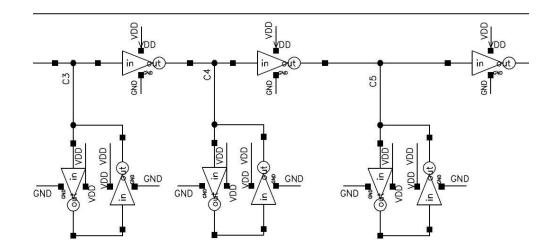
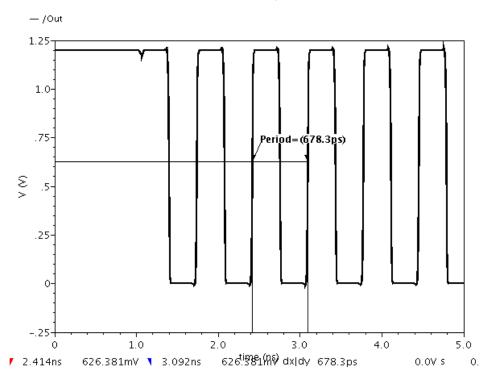


Figure 2.8: The ring oscillator with HDC cells

The simulation result for this modified ring oscillator is in figure 2.9 and it increased the period of oscillation from 144.5 ps to 678.3 ps.



Transient Response

Figure 2.9: Simulation results for the ring oscillator with HDC cells

2.1.2 A fine tuning stage (DCV Array)

For the proposed DCO, we use a fine tuning stage to give a step change in the period of oscillation of about 48 ps. The DCV cell is a NAND based delay cell. A schematic view of this DCV cell can be found in figure 2.10.

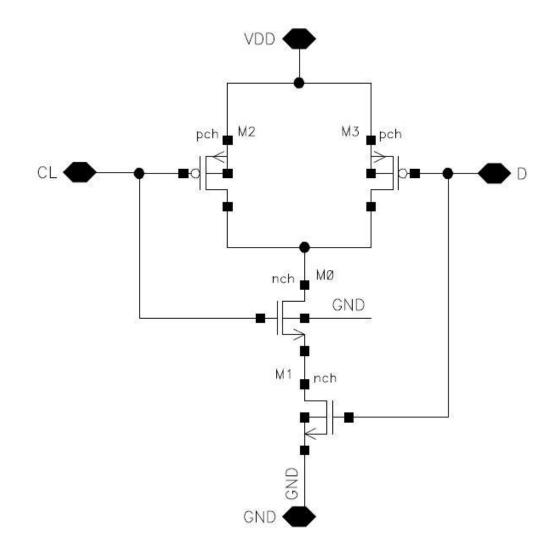


Figure 2.10: A schematic view for the DCV cell

The idea of operation of this cell is that, the gate capacitance seen from node CL (It refers to the load capacitance and it is connected to the output node of each inverter in the ring oscillator) can be changed according to the gate voltage applied to the node D (it refers to digital input bit of the DCO). The formula of

the resulting gate capacitance for this NAND based cell in both cases (when D is high and when it is low) is as follows in figure 2.11:

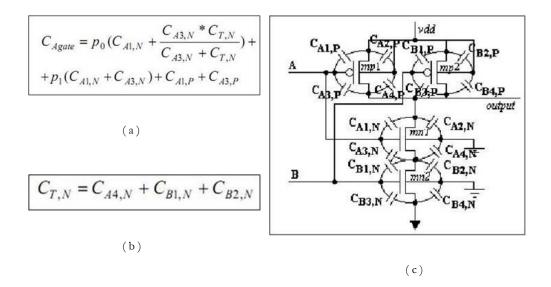


Figure 2.11: The gate capacitance of NAND gate

Where p0 is probability second input (B) to be equal the ZERO and p1 is probability second input to be equal the ONE (p0 + p1 = 1). The simulation result in figure 2.12 illustrates changing the gate capacitance with the gate voltage (CL) in two cases, when D is high and when it is low. From this result, it seems that we can achieve high capacitance for the case when the digital input bit is high (D=1) and we can get a low capacitance when it is low (D=0). Increasing the load capacitance for each node of the ring oscillator output means increasing the delay as the value of RC constant will be increased. For the ring oscillator and for a typical inverter, the propagation delay can be calculated from the following formula:

$$T_p = 0.69C_L \left(\frac{R_{\rm eqp} + R_{\rm eqn}}{2}\right)$$

Where CL is the output capacitance of the inverters of ring oscillator which is the gate capacitance of the delay cells. Reqp and Reqn are the equivalent resistances of the PMOS and NMOS transistors respectively.

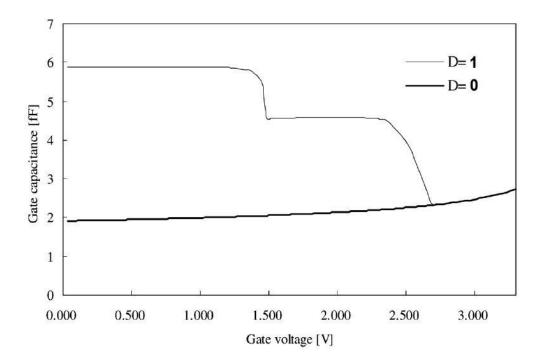


Figure 2.12: Simulation results for the NAND based DCV Cell

We use this NAND based DCV cell as a building element for a DCV block in the fine tuning stage. Every twelve DCV cells are connected to a single input which is the digital input bit (D). The output of this block consists of six nodes from C1 to C6. These nodes are connected to the corresponding outputs of six inverters in the ring oscillator. A block diagram for the DCV block is in figure 2.13.

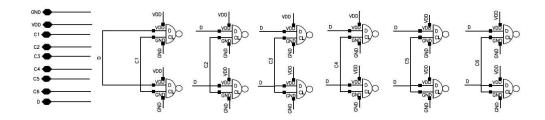


Figure 2.13: A block diagram for the DCV building block

We then use this DCV block to construct the DCV array consisting of sixteen DCV blocks. All outputs of the sixteen blocks (C1 to C6) are connected to the same six nodes of the ring oscillator but each input from the DCV blocks is connected to a different external digital bit so, for the DCV array of the fine tuning stage we have a digital word of sixteen bits. Figure 2.14 is the internal structure of the DCV array:

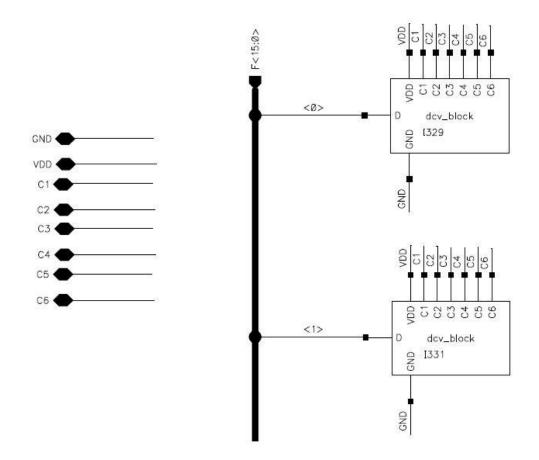


Figure 2.14: DCV array internal structure

A symbol view of this DCV array is shown below in figure 2.15:

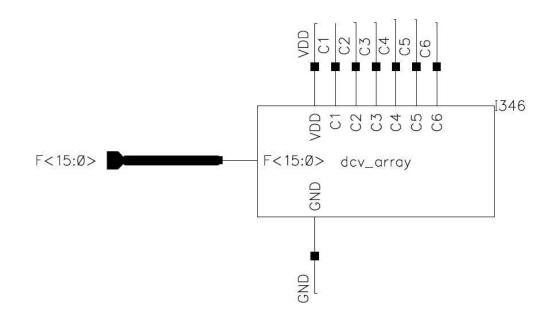


Figure 2.15: A symbol view for the DCV array

An important thing to notice here is that, although we achieved a fixed step in period of about 48ps for fine stage, the step in output frequency is not fixed because the relation between the frequency and period is not linear. For the same period step we get many frequency steps depending on the location of this period step in time access. For example, our DCO period range is from 3.3 ns to 10 ns consider adding a period step to the first period in range, from 3.3 ns to 3.348 ns (3.3ns+48ps), 3.3 ns corresponds to a frequency of 303 MHZ and 3.348 ns corresponds to a frequency of 298.686 MHZ so a period step of 48 ps from 3.3 ns to 3.348 ns causes a frequency step of 4.314 MHZ. Let's consider the same period step added to another period in another location in time access, for the last period of output oscillation from the DCO which is 10 ns, the period before this one is 9.952 ns (10 ns - 48 ps), 9.952 ns corresponds to a frequency of 100.482 MHZ and 10 ns corresponds to a frequency of 100 MHZ, so the same period step of 48 ps when added to the period 9.952 ns we get a frequency step of 0.482 MHZ. For these two cases we get two different frequency steps of 4.314 MHZ and 0.482 MHZ although the period step of fine stage is constant. Conclusion is that, although the period step for fine stage is constant we will get different frequency steps (not fixed) because the relation between frequency and

15

period is not linear. Figure 2.16 illustrates why we get different frequency steps for the same period step.

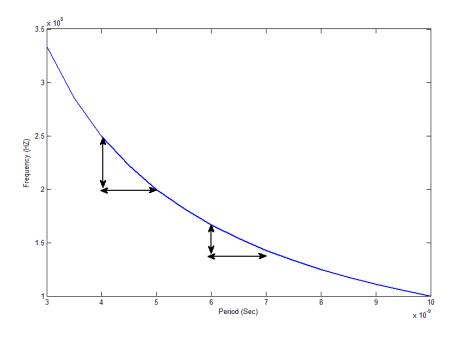


Figure 2.16: Period step Vs. frequency steps

2.1.3 A coarse tuning stage (DCV2 Array)

After using the fine tuning DCV array we achieved a relatively small step change in output period of the DCO and consequently a relatively small frequency steps. For the coarse stage we need to get larger frequency steps to reduce the lock time of the PLL, to achieve these larger frequency steps we have to use delay cells with larger period steps than the fine tuning stage (48 ps), so we used another delay cell to get this larger period step. The delay cell used for coarse tuning stage is also based on the NAND configuration but with a transmission gate in the beginning. The enable line for this transmission gate is the external digital input bit (D) and the input to it is the load capacitance node (CL) which is connected to the output of each inverter in the ring oscillator. Figure 2.17 is a schematic view for this delay cell.

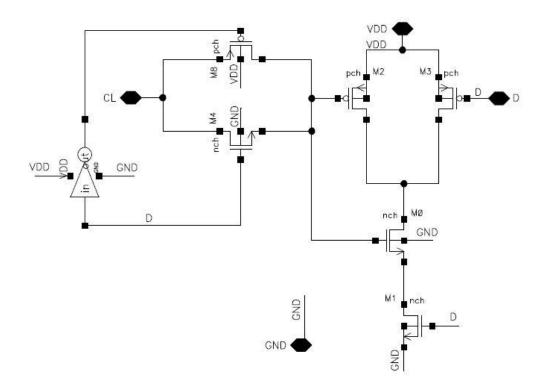


Figure 2.17: A schematic view for the DCV2 cell

When the external input is low (D=0), the CL node will be disconnected from the NAND cell and introduces low capacitance and consequently low delay. When the external input is high (D=1), the CL node is now connected to the NAND cell and can see the gate capacitance of it. The two cases of D=0 and D=1 here are different from those in fine tuning stage, as in fine tuning stage the CL node was connected to NAND cell in both cases that's why the period step in fine stage was relatively small. In coarse stage, the CL node is connected only when D=1 so we can say that, in coarse stage the node CL can see the capacitance of the NAND cell or it cannot see it, so the period step here is larger than the period step in fine stage. For coarse stage we achieved a period step of 380 ps. Another advantage for using the DCV2 cell is that, it helped us increase the largest output period of the DCO (10 ns) without affecting the smallest period (3.3 ns) by changing the sizing of the NAND cell in this DCV2 cell

We use this NAND based DCV2 cell as a building element for a DCV2 block in the coarse tuning stage. Every eight DCV2 cells are connected to a single input which is the digital input bit (D). The output of this block consists of four nodes from CL1 to CL4. These nodes are connected to the corresponding outputs of four inverters in the ring oscillator. These four inverters are following the six inverters used in the fine stage. A block diagram for the DCV2 block is in figure 2.18.

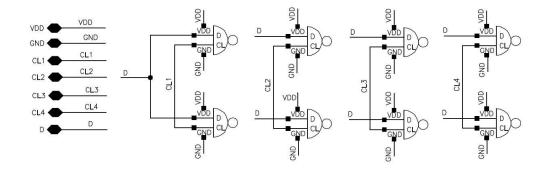


Figure 2.18: DCV2 block

We then use this DCV2 block to construct the DCV2 array consisting of sixteen DCV2 blocks. All outputs of the sixteen blocks (CL1 to CL4) are connected to the same four nodes of the ring oscillator but each input DCV2 blocks is connected to a different external digital bit so, for the DCV2 array of the coarse tuning stage we have a digital word of sixteen bits. The following figure is the internal structure of the DCV2 array (Figure 2.20).

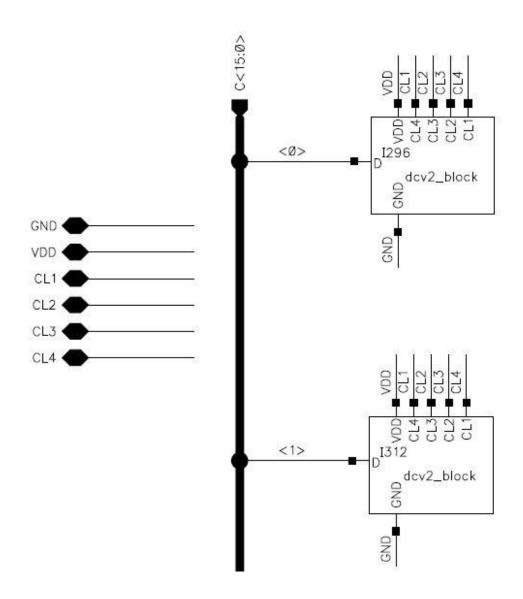


Figure 2.19: DCV2 array

A symbol view of this DCV2 array is shown in figure 2.20.

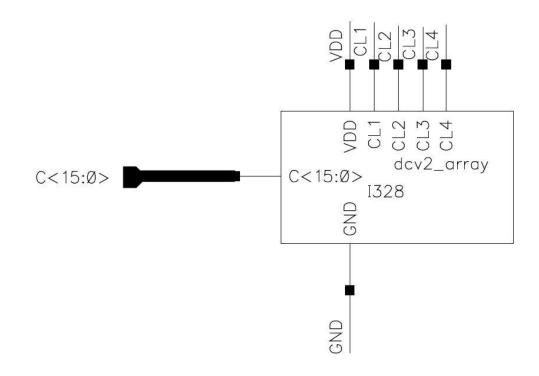


Figure 2.20: A symbol view for the DCV2 array

2.1.4 Conclusion and final results

In this section we introduced the proposed DCO which is consisting of three main blocks:

- 1. A ring oscillator
- 2. A fine tuning stage
- 3. A coarse tuning stage

The output period for this DCO ranges from 3.3 ns to 10 ns which is equivalent to a lock range from 100 MHZ to 300 MHZ. We used HDC cells to add a fixed delay to the ring oscillator and DCV cells for both fine and coarse tuning stages to add a programmable delay. The fine tuning stage gives a small step (48 ps) in period and consequently small steps in frequency. The coarse tuning stage gives large period step (380 ps) and consequently large steps in output frequency.

The following table in figure 2.21 shows different output periods of the DCO according to different values for the digital words of both fine and coarse tuning stages.

Control bits	Period	Frequency	Period step	Freq. step MHz
Fine Word Coarse Word	ns	MHz	ns	MHZ
0000000 0000000 0000000 0000000	3.289	304.878		
00000000 0000000 0000000 00000001	3.669	272.553	0.38	32.325
10000000 0000000 0000000 00000001	3.718	268.961	0.049	3.592
10000000 0000000 0000000 00000011	4.1	243.902	0.382	25.059
11000000 0000000 0000000 00000011	4.147	241.138	0.047	2.764
11000000 00000000 0000000 00000111	4.53	220.75	0.383	20.388
11100000 0000000 0000000 00000111	4.578	218.435	0.048	2.315
11100000 00000000 00000000 00001111	4.961	201.572	0.383	16.863
11110000 0000000 0000000 00001111	5.009	199.641	0.048	1.931
11110000 0000000 0000000 00011111	5.38973	185.538	0.38073	14.103
11111000 0000000 0000000 00011111	5.43695	183.927	0.04722	1.611
11111000 00000000 00000000 00111111	5.82148	171.778	0.38453	12.149
11111100 00000000 00000000 00111111	5.869	170.387	0.04752	1.391
11111100 00000000 00000000 01111111	6.25404	159.897	0.38504	10.49
11111110 00000000 00000000 01111111	6.29857	158.766	0.04453	1.131
11111110 0000000 0000000 11111111	6.68375	149.617	0.38518	9.149
11111111 0000000 0000000 11111111	6.73115	148.563	0.0474	1.054
11111111 00000000 00000001 11111111	7.1095	140.657	0.464	7.906
11111111 10000000 00000001 1111111	7.1607	139.651	0.0512	1.006
11111111 1000000 00000011 11111111	7.5419	132.593	0.3812	7.058
11111111 11000000 00000011 11111111	7.59375	131.687	0.05185	0.906
11111111 11000000 00000111 11111111	7.9758	125.379	0.38205	6.308
11111111 11100000 00000111 1111111	8.0248	124.614	0.049	0.765
11111111 11100000 00001111 11111111	8.41188	118.879	0.38708	5.735
11111111 11110000 00001111 11111111	8.4583	118.227	0.04642	0.652
11111111 11110000 00011111 11111111	8.84562	113.05	0.38732	5.177
11111111 11111000 00011111 11111111	8.8918	112.463	0.04618	0.587
11111111 11111000 00111111 11111111	9.2775	107.788	0.3857	4.675
11111111 1111100 00111111 1111111	9.3259	107.228	0.0484	0.56
11111111 1111100 01111111 1111111	9.7117	102.969	0.3858	4.259
11111111 1111110 01111111 1111111	9.7605	102.454	0.0488	0.515
11111111 1111110 11111111 1111111	10.146	98.561	0.3855	3.893
11111111 1111111 111111111111111111	10.194	98.097	0.048	0.464

Figure 2.21: Different output periods of the DCO

The simulation result in figure 2.22 is different output waveforms for the DCO.

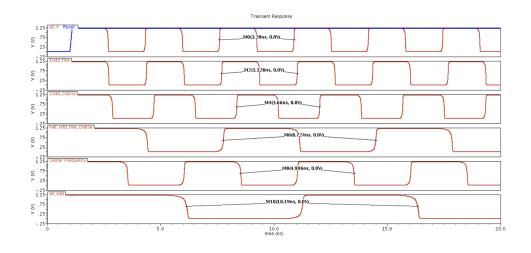


Figure 2.22: Different output waveforms of the DCO

Figure 2.23 is the period steps versus the digital code according to the above table, one coarse step change followed by one fine step change. From this figure, one can easily notice the coarse step is larger than the fine step. The start point of our range is 3.3 ns and the end point is 10 ns corresponds to the lock range for the proposed DCO (100 MHZ to 300 MHZ).

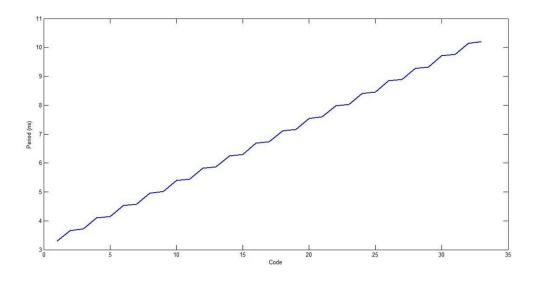


Figure 2.23: Period steps versus code

Figure 2.24 is the output frequency versus the 32-bit digital input word (16-bit word for fine stage and 16-bit word for coarse stage).

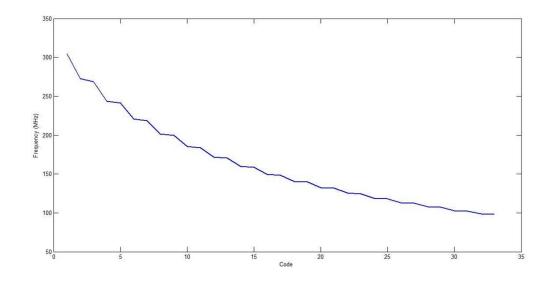


Figure 2.24: Frequency steps versus code

2.2 PFD

A phase detector is a circuit capable of delivering an output signal that is proportional to the phase difference between its two input signals Ref_Freq and DCO_Out as mentioned in figure 1.2. When the PLL moved into digital territory, digital phase detectors become popular, such as EXOR gate, the edge-triggered JK-flip flop, and the so-called phasefrequency detector (PFD). The PFD differs greatly from the other phase detector types as its name implies, its output signal depends not only on phase error but also on frequency error when the PLL has not yet acquired lock. The PFD is built from two D-flip flops, whose outputs are denoted UP and DOWN(DN) as shown in figure 2.25, these two signals are the digital representation of the phase/frequency error. The PFD can be in one of four states:

- UP=0, DN=0
- UP=1, DN=0
- UP=0, DN=1
- UP=1, DN=1

The fourth state is inhibited, however, by an additional gate. Whenever both flip flops are in the 1 state, a logic low level appears at their reset inputs, which reset both flip flops. We assign the symbols -1, 0, and 1 to these three states :

- UP=0, DN=0 \rightarrow state -1
- UP=1, DN=0 \rightarrow state 0
- UP=0, DN=1 \rightarrow state 1

The actual state of the PFD is determined by the positive-going transients of the signals Ref_Freq and DCO_Out, as explained by the state diagram in figure 2.26, a positive transition of Ref_Frq forced the PFD to go into its next higher state, unless it is already in the 1 state. In analogy, a positive edge of DCO_Out forces the PFD into its next lower state, unless it is already in the -1 state.

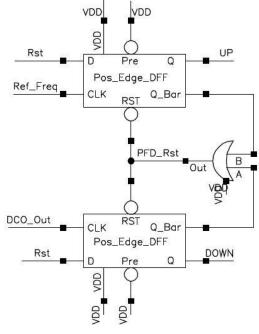


Figure 2.25: PFD schematic

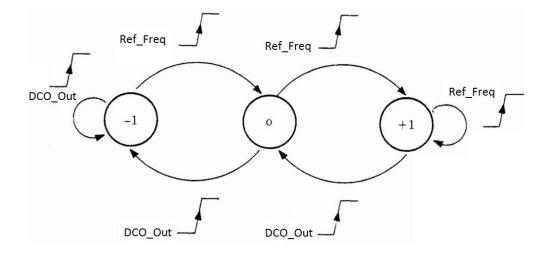


Figure 2.26: State diagram of the PFD

To see how the PFD works in a real PLL system, we consider the waveforms in figure 2.27, this figure shows the three cases:

- a) First 25ns shows the case where Ref_Freq leads, therefore the PFD toggles between states 0 and 1.
- b) If Ref_Freq lags as in the next 25ns, the PFD now toggles between states-1 and 0.
- c) The signals Ref_Freq and DCO_Out are 'exactly' in phase; both positive edges occur at the same time; hence the PFD will stay in state 0 forever.

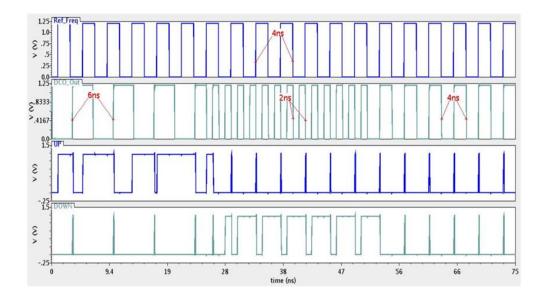


Figure 2.27: PFD simulation results

2.3 Loop Filter:

The Loop filter stage controls the output capacitance by changing the number of DCV cells that are turned on, as shown in figure 2.28 the digital control signals is used to increase/decrease the DCO frequency for a certain period of time by reducing /increasing the capacitance. If the input to the DCV is '1', it provides more capacitive load at the output. If more number of cells are on (input is '1'), then it acts as more capacitive load on the ring oscillator which reduces the DCO frequency.

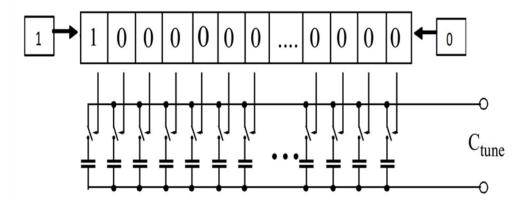


Figure 2.28: digital control signals used to switch a set of varactors

In order to control each of the fine DCV array and the coarse DCV array individually, we have used two 16-bits bi-directional loadable shift registers and here part of the schematic view for each shift register in figure 2.29

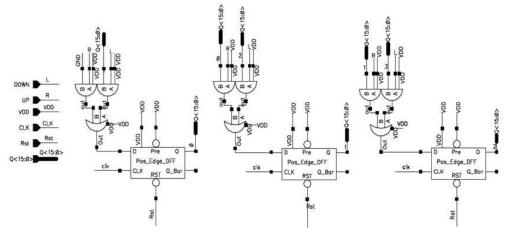


Figure 2.29: Schematic view of the shift register

Initially, 8 DCV cells of each array are on, this achieved by using asynchronous Reset and Preset signals. Depending on the up/down signals from PFD, the frequency is either increased or decreased. When phase and frequency acquisition starts, if the output of PFD is up, then the contents of the shift register are left shifted and bit '0' is pushed into Q<15> and hence the capacitive load decreases and the frequency increases. Likewise, if it is down, the contents of the shift register are right shifted and bit '1' is pushed into Q<0>. This reduces the frequency of the DCO as the capacitive loading at the output increases.

2.4 Overall Design

In this section the simulation results of the whole system will be introduced.

2.4.1 Extreme Reference frequency

First let the input (reference frequency) signal be an extreme, let's say the minimum frequency in the desired range (100-300MHz) as in figure 2.30.And by knowing

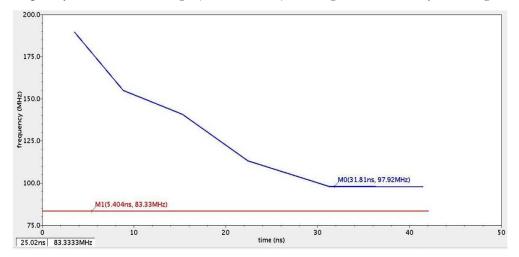
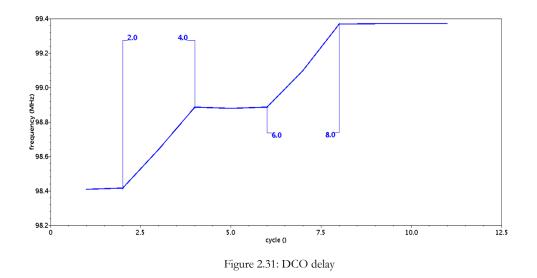


Figure 2.30: Reference = 100 MHz

that the DCO frequency initially equals to the center frequency (200MHz), then the shift register should get only DOWN pulses from the PFD, which activates all the DCV cells and introduces the lowest frequency .now what about the lock time ? Actually this depends on two factors, the:

- CLK used for the loop filter: this means the rate of changing in DCO frequency at a certain time. It's clear that we need to increase the frequency of this CLK to get smaller lock time.
- ii. DCO Delay : this means time needed by the DCO to change its frequency after one step delay as shown in figure 2.31, where one DCV cell is deactivated after two cycles, the effect of this step appears after exactly two cycles, this time is considered as the DCO delay, thus the CLK above in part (i) should take in consideration this delay to get the true UP/DN pulse after the new change in the DCO frequency.



2.4.2 Intermediate Reference frequency

Now let's consider this case, reference = 250MHz as below in figure 2.32.

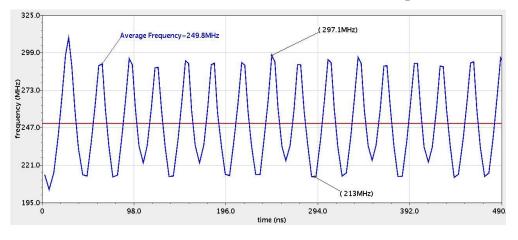
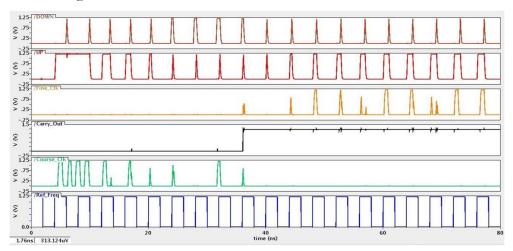


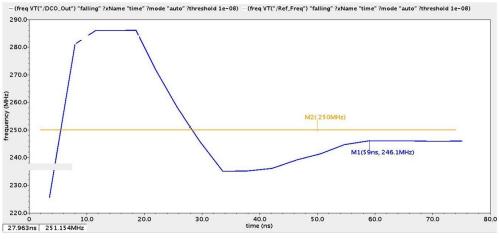
Figure 2.32: Reference= 250MHz

It's clear that average frequency is almost equals to the reference, but the problem that appears here due to the wide steps of the course stage (up to 31MHz), which results to these oscillations with a very high peak-to-peak value. Simply to solve this problem, the coarse stage must be stopped after being operating separately from the fine stage. This is applied using a 4bit Counter to count the maximum number of steps needed by the coarse stage, which is 16 at the worst case, after that a carry signal is used to turn off the coarse shift register and activates the fine one. The simulation results of this idea are

shown in figures 2.33a, 2.33b and 2.33c.



a) Control signals



b) Frequency response

Course stage										
Course SH_Reg	Fine SH_Reg	Frequncy (MHz)								
0000000 00000001	0000000 00001111	260.28								
0000000 00000011	00000000 00001111	236.85								

	Fine stage	
Course SH_Reg	Fine SH_Reg	Frequncy (MHz)
0000000 00000011	0000000 00000001	245.21
0000000 00000011	0000000 00000000	246.13

c) Digital words of Both SRs

Figure 2.33: Effect of using counter

In this example the reset signal of the whole system was designed to start the DCO oscillations at the center frequency, this is achieved by activating four DCV cells from each array, to calculate this frequency as explained in section 2.1:

Period (ps) = $3270 + \text{Coarse}_1$'s × (380) + Fine_1's × (48) = 5ns (200MHz).

Through the first stage the fine SR is stopped as shown in figure 2.33a and 2.33c, and at the end of this stage the DCO frequency was undecided between two frequencies around the reference corresponding to the digital words which clarified in figure 2.33c, after that the course stage stopped using the carry signal at 38n as shown in figure 2.33a.At the same time the fine stage started, this can be observed through the frequency response in figure 2.33b ,where the very small slopes appears after the 38ns.

2.4.3 Frequency step response

It was necessary to consider in our design that the Reference signal frequency may be changed during the loop, because this PLL targets a low power clock and data recovery system. But in order to detect this change, the loop filter won't be a simple shift registers as it now because a certain controlling circuit must be added. Actually we have replaced the coarse SH by a frequency counter circuit which is one of Digital Instruments that can be used to measure signal frequency and period, the basic idea is illustrated in the following figure:

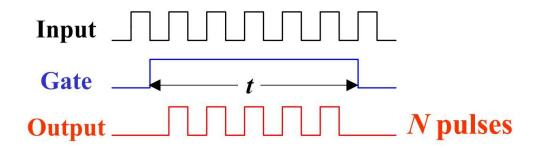


Figure 2.34: Frequency counter

We have implemented this idea by generating the Gate signal (figure 2.34) from the reference frequency, and the input clock source (time base) signal which used to trigger the counter from a ring oscillator which already discussed in section 2.1.1.

After counting the N pulses mentioned in figure 2.34, this number is therefore mapped to the 16 bit to control the coarse DCV array. Note that at the end of each counting period, the counted value should mapped synchronously and the counter should cleared. In the following the whole system simulation result after using the frequency counter :

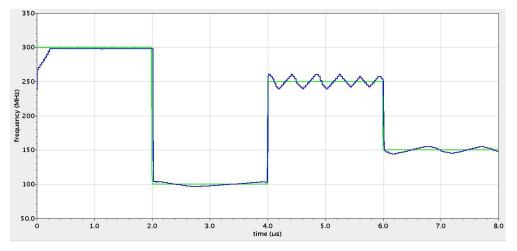


Figure 2.35: frequency step response

After this modification on the system to become capable of reacting with the step system response, the consumed power is increased from 0.25 to **0.6** mWatt due to the high frequency clock added. Also the lock time at the worst case does not exceed 300ns. Now there is one more enhancement needed

The last improvement needed, is to reduce the oscillation of the fine stage around the reference, this is achieved by slowing its clock frequency. Figure 2.36 shows the result after this modification.

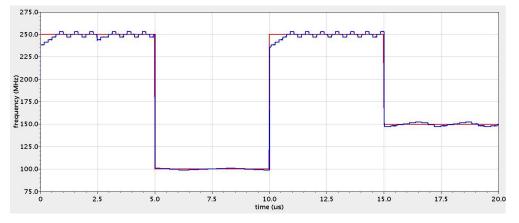


Figure 2.36: Eliminating the oscillations

As expected, slowing the fine SR led to the expansion of the time lock until the time of 1us as figure 2.36 shows.

2.4.4 Jitter calculation :

Figure 2.37 shows the eye-diagram of DCO output clock when locked at 300 MHz plotted using Cadence tools. In this eye-diagram, each and every cycle of the DCO output clock are overlapped on one clock period (after the DCO clock is locked to the reference) and the maximum deviation that can be obtained from the graph is measured as peak-to-peak jitter. Number of cycles that are taken into account are 100. The delay is measured at 50% voltage levels and the period jitter determines how noisy and stable the oscillator output signal is.

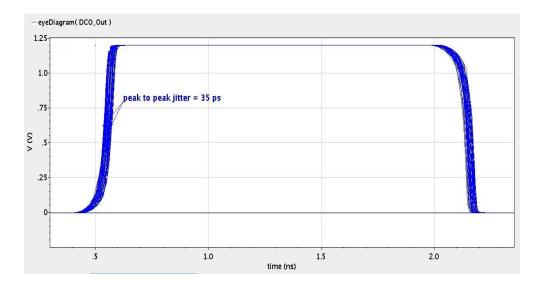


Figure 2.37: Eye-diagram of the ADPLL

The peak-to-peak jitter for this implementation when the feedback signal is locked at 300 MHz is 35 ps.

Chapter 3

STANDARD CELLS APPROACH

Standard Cells Approach means that the targeted block will be written with one of the Hardware Description Language HDL codes such as Verilog or VHDL, and then can be translated into an hardware circuit using Standard Cells library. Thanks to AMS we are able to simulate and test the targeted block with analog blocks in cadence environment.

One big advantage of using such approach is that the designer is not have to deal with the block gates at the transistor level and check the sizing of the logic gates . This approach also made the layout step very easy and effective in area.

3.1 PFD

As mentioned before, the Phase and Frequency Detector (PFD) will be used as a phase detector to detect the phase and the frequency difference between the reference signal and the output signal of the DCO.

First of all, we will write the code of our PFD using Verilog programming language, we will design a block which has:

- Two input ports (reference signal and DCO output).
- Two output ports (Up and Down).

The Up signal indicates that the system should increase the DCO frequency (i.e. the reference frequency is higher than the DCO frequency) and the Down signal indicates that the system should decrease the DCO frequency.

module dff (input d,clk,reset,output reg q); always@(posedge clk,negedge reset) if(~reset) q<=1'b0; else q<=d; endmodule module pfd (input refSignal,dcoSignal,output up,dn); wire intReset; assign intReset=~(up&dn); dff up_dff(1'b1,refSignal,intReset,up); dff dn_dff(1'b1,dcoSignal,intReset,dn); endmodule

And using Synplify PRO we translated our PFD into the corresponding standard cells schematic as shown in figure 3.1

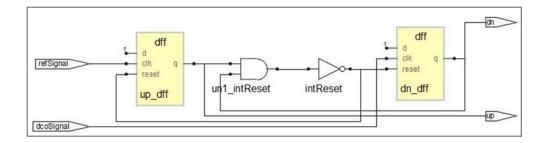


Figure 3.1: PFD schematic

This PFD can be imported as Verilog code in cadence also , see appendix A for more details.

3.2 Loop Filter

The Loop Filter used here is simply a 16 bits shift register ,following the same procedure of PFD ,we got the schematic view of the Loop Filter as shown in figure 3.2.

```
module loopFilter ( input up,dn,reset, output reg [15:0] q);
wire ored;
assign ored = up | dn;
always @(posedge ored, negedge reset)
if(~reset) //active low reset(level sensitive)
q <= 16'b11110000_00000000;
else if(up & ~dn)
q <= \{q[14:0],1'b0\};
else if(dn & ~up)
q <= \{1'b1,q[15:1]\};
endmodule
```

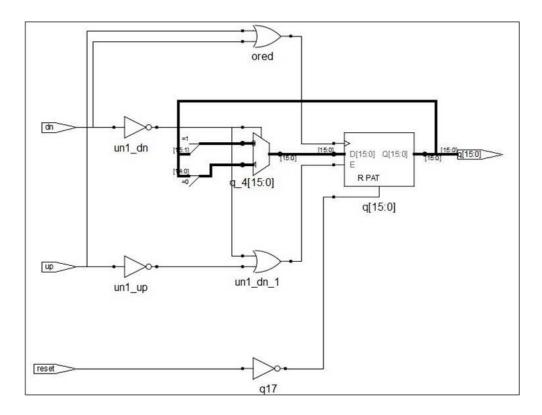


Figure 3.2: Loop Filter schematic

We used two Loop Filters, one as a Coarse and the other as a fine, the Coarse one is responsible for the large step change in the DCO output frequency while the fine one is responsible for the small step change.

3.3 PFD and Loop Filter

Now, we have PFD and Loop Filter as functional blocks, so we can connect them together as shown in figure 3.3.

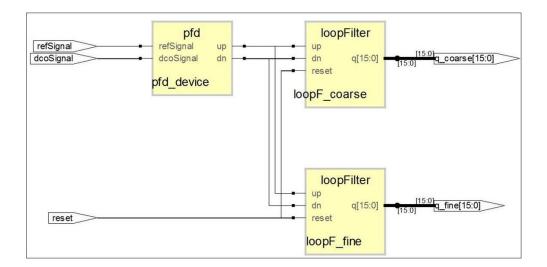


Figure 3.3: PFD and Loop Filter

Now, we have our functional block (PFD + Loop Filter) so we can import it to cadence environment and connect it with the custom designed DCO.

3.4 Overall Design

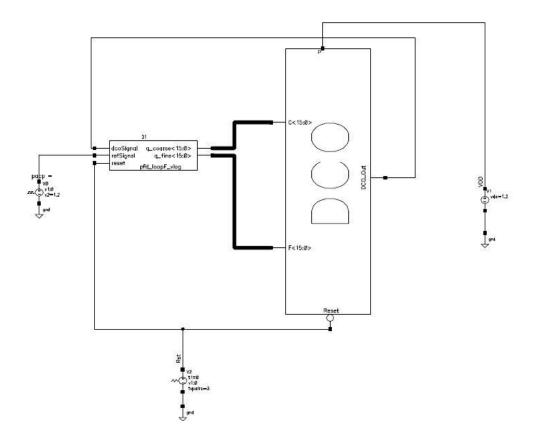


Figure 3.4: Overall ADPLL (standard cells)

Using AMS we can simulate analog and digital(functional) blocks together and check the functionality of our ADPLL and the result was as shown in figure 3.5

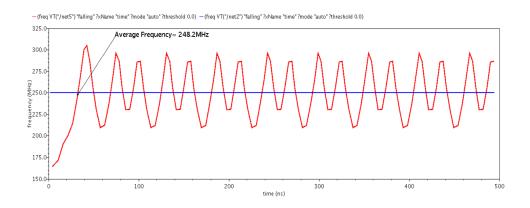


Figure 3.5: AMS of the overall ADPLL without the counter

As we can see that the DCO output frequency is oscillating around the reference frequency and that is due to the coarse large steps, so we need to stop the coarse shift register to eliminate the large oscillations.

4 bits counter is used for that purpose and when the counter carry bit activated the coarse shift register is turned off and the fine one is turned on as shown in figure 3.6

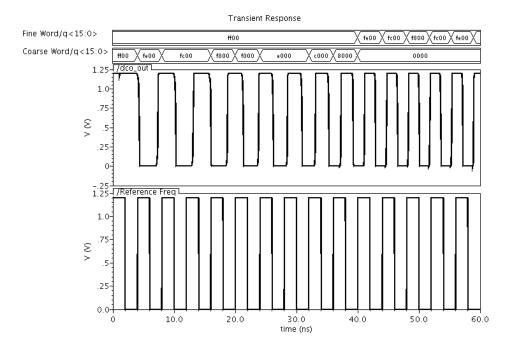


Figure 3.6: AMS simulation of the overall ADPLL with counter

And the result of the Overall ADPLL as a functional (behavioral) block was as shown in figure 3.7.

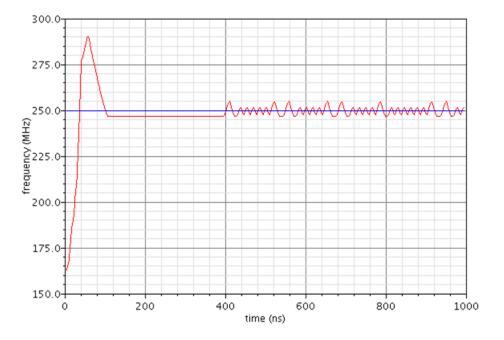


Figure 3.7: AMS of the ADPLL with counter.

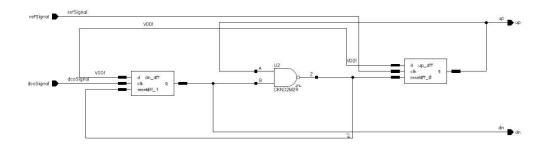
From the above figure we found that the lock time is less than 150 ns which means that our lock time restriction (lock time < 10 us) is satisfied.

But, the problem is, the above design assuming the reference frequency is fixed and will not exposed to a step change.

All the above simulation results is done considering the functional behavioral of the PFD and the Loop Filter (as code only).

We did the technology mapping using Design Compiler by converting the functional Verilog code of the PFD and the Loop Filter into a netlist to generate a mapped code which contains the needed standard cells to achieve the block functionality.

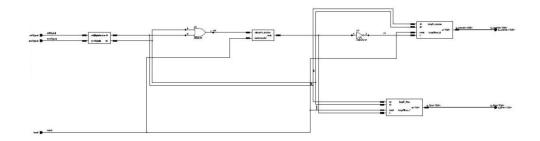
We extracted the transistor level schematic of the PFD and the Loop Filter successfully using the standard cells of UMC65nm digital kit as shown in figure 3.8.



a) Imported schematic of PFD in cadence

P-	-0	-d		┍⋑┈		-	-0		┍╴		┍═᠉─	₽ °	-6-
					5								-0
													-6-
					10			-D-				<u> </u>	- -
										<u> </u>	1 0-		
 							-			-			н

b) Imported schematic of Loop Filter in cadence



c) Imported schematic of PFD and Loop Filter together in cadence

Figure 3.8: The standard cells of both PFD and Loop Filter

As an example we took a snap shot of one standard block inside the PFD schematic in figure 3.8.a. above

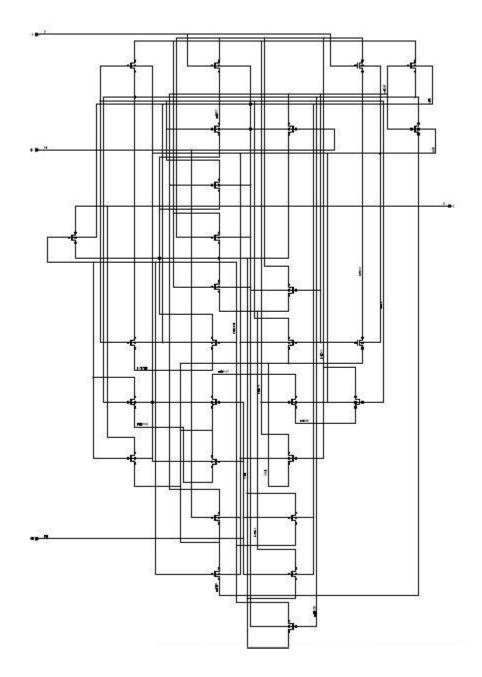


Figure 3.9: The transistor level of the standard D flip flop in cadence.

Chapter 4

LAYOUT

The last step of the design flow is the layout, first of all DCO is considered as the core of the ADPLL so as we started the design in the schematic scope with the DCO, we will start with the DCO in the layout scope and we believe that the range will be changed due to the capacitance and the resistance added by the layout (i.e. more delay).

4.1 DCO

The proposed DCO depends mainly on the delay cells (HDCs and DCVs) and because of the layout, the delay is not the same as schematic any more.

4.1.1 DCV

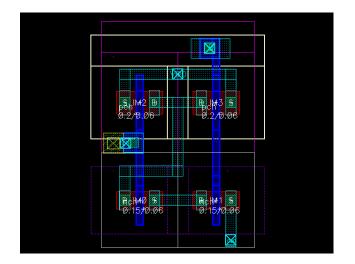


Figure 4.1: DCV layout

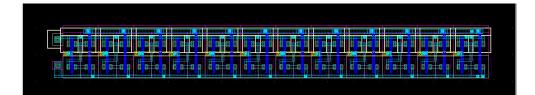


Figure 4. 2: DCV block layout

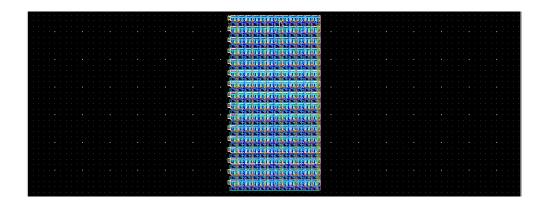
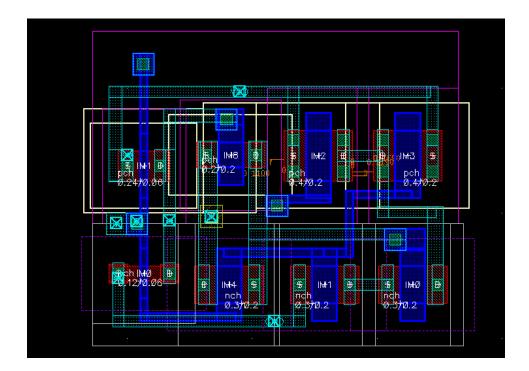


Figure 4. 3: DCV array layout



4.1.2 DCV2

Figure 4. 4: DCV2 layout

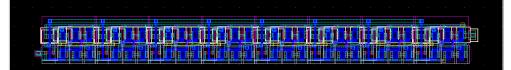


Figure 4. 5: DCV2 block layout.

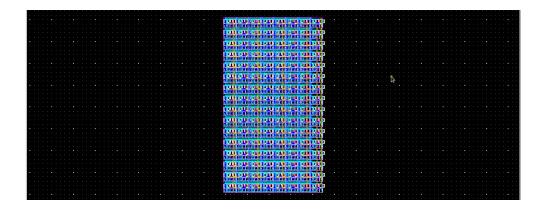


Figure 4. 6: DCV2 array layout

4.1.3 Ring Oscillator

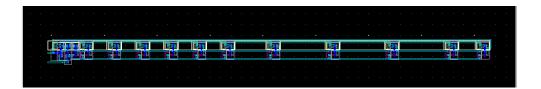


Figure 4. 7: Ring Oscillator layout

4.1.4 Complete DCO

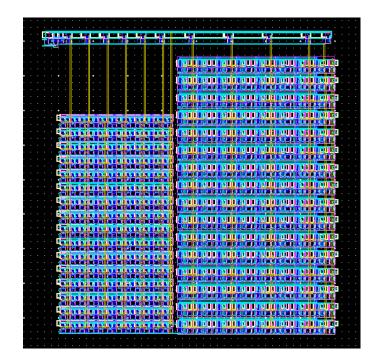


Figure 4. 8: Final DCO layout

The DCO range was from 3.33 ns to 10 ns of period (i.e. from 100 MHz to 300 MHz), but after constructing the layout directly without any modification to the sizing of DCVs or removing any fixed delay cells , the DCO range was from 11 ns to 16 ns (i.e. from 62.5 MHz to 90 MHz) which is out of our required range, so we had to remove the fixed delay cells (i.e. HDCs) but the range was still not satisfied, so we started to modify the length of the transistors in the DCV2 cells because the problem was with the lower bound of the required range (i.e. 100 MHz) and also increase the supply voltage source from 1.2v to 1.5v to satisfy the required range, and finally we have got 3.236 ns to 9.952 ns (i.e. 100.48 MHz to 309.02 MHz) as shown in figure 4.9.

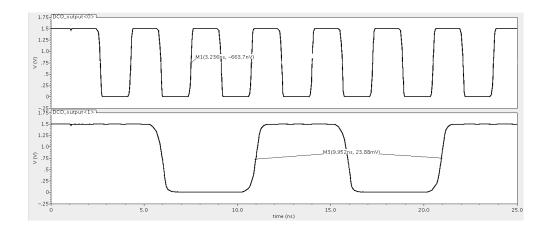


Figure 4. 9: DCO operating range in layout.

4.2 PFD

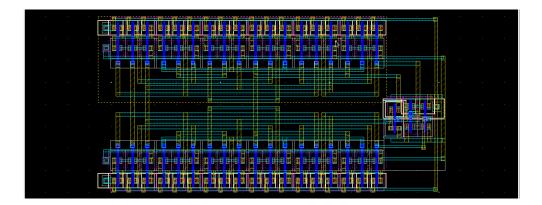


Figure 4. 10: PFD layout

The proposed PFD consists of two D flip flop and OR gate. At the first time, when simulating the layout directly with power supply voltage 1.2v, we found that the functionality of the PFD is not working correctly, so we decreased the supply voltage to 1v and the PFD worked properly as shown in figure 4.10 and figure 4.11.

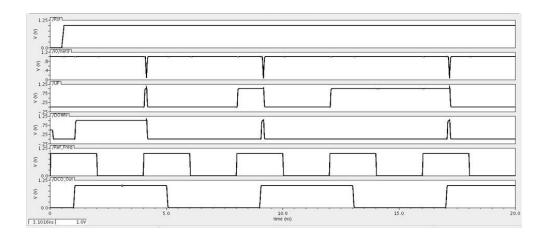


Figure 4.11: PFD pre-layout simulation.

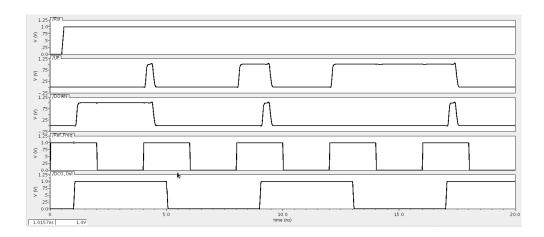


Figure 4.12: PFD post-layout simulation.

4.3 Loop Filter

As mentioned earlier, the proposed Loop Filter in ADPLL is a shift register, which contains D FF's and some combinational blocks.

• •	•	•	·	•	•	•	• •	•	٠	٠	٠	٠	٠	٠	٠	٠	٠	٠	•	• •	•	•	٠	•	•	•	•	•	•	•	٠	٠	•	٠	•	٠	٠	• •		•
• •																																								
	Ŧ	- 1-1	H	ų,	ij.	Ð	M	μŦ	١W	Į,	4	U,	þ	W.	W.	44	ηĄ	Ц		1		ΗĐ	ij.	61	19	ЦJ.		i i	٩Đ	W 1	府自	-	М		H	i i i		d d	1	
	bi.		pi an	içka	e <mark>pto</mark>	4	1	1972	P m	al a	ft.	-	a di ji	14	pine	ft	7	-1	a e	P		3					-		_	D.		ľ								
																		•			1							1	-											
• •																																								

Figure 3.13: Shift Register layout

The post-layout simulation result was different from the pre-layout simulation due to the delay produced by the layout capacitance and resistance as shown in figure 4.12 and figure 4.13, but that is not a big deal. As we can see in figure 4.12 in the time period 2.5ns to 5ns the shift occurs at the positive edge of the CLK signal because the up and down signal are different, but in the same time slot in figure 4.13 the shift occurred once due to the delay produced by the layout.

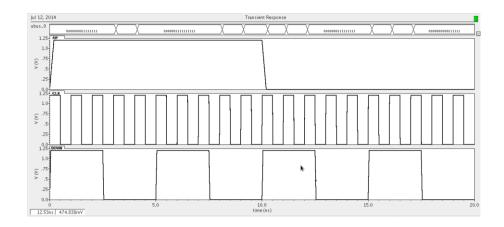
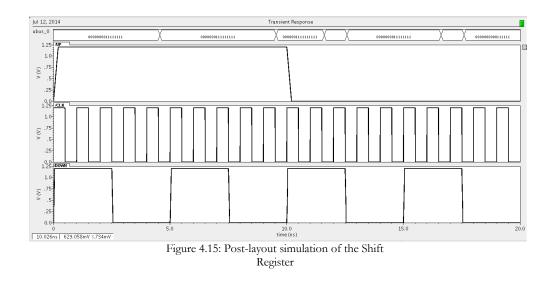


Figure 4.14: Pre-layout simulation of the Shift Register



4.4 Overall Design

After constructing the layout of each block in the system, the layout of the overall design can be constructed as shown in figure 4.16

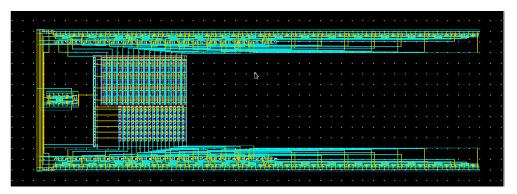


Figure 4.16: The overall layout

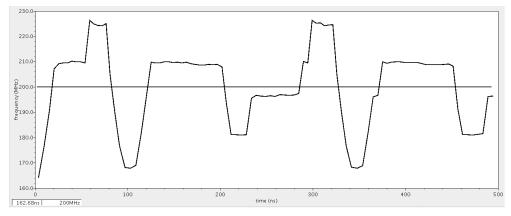


Figure 4.17: Overall post layout simulation

4.5 Conclusion

A comparison between the Required Specifications and Achieved Specifications is held as shown below

Required Specifications	Achieved Specifications
Power < 1 mW	0.6 mW
$Area < 0.01 mm^2$	From Layout= $0.0086mm^2$
Lock time $< 10 \mu s$	1 <i>µs</i>
P-to-P jitter < 20 ps	35 ps

References

- [1] Roland E. Best, "Phase Locked Loops, Design, simulation and applications 5th Edition" New York McGraw-Hill, 2003.
- [2] Duo Sheng, Ching-Che Chung and Chen-Yi Lee, "An Ultra-Low-Power and Portable Digitally Controlled Oscillator for SoC Applications" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 54, NO. 11, NOVEMBER 2007.
- [3] SALEH R. AL-ARAJI, ZAHIR M. HUSSAIN and MAHMOUD A. AL-QUTAYRI, "DIGITAL PHASE LOCK LOOPS, Architectures and Applications" Springer,2006.
- [4] Pao-Lung Chen, Ching-Che Chung, and Chen-Yi Lee "A Portable Digitally Controlled Oscillator Using Novel Varactors " IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 52, NO. 5, MAY 2005.
- [5] João Baptista Martins, Ricardo Reis and José Monteiro, "Capacitance and Power Modeling at Logic-Level".
- [6] CADENCE, "Virtuoso AMS Designer Environment Tutorials", 2008.
- [7] CADENCE, "Virtuoso AMS Designer Simulator User Guide", 2006.
- [8] CADENCE, "Virtuoso AMS Environment User Guide", 2006.
- [9] Alain Vachoux, "Top-Down Digital Design Flow" Version 6.0, October 2011.
- [10] Ahmed Ahmed, Hussein Mohamed, Khaled Ebrahim, Khaled Mohamed and Mohamed Sherif "All Digital Phase Locked Loop (ADPLL)", July 2013

Appendix A

AMS TUTORIAL

Through this tutorial you will learn how to simulate a system containing digital Verilog blocks, digital VHDL blocks and analog blocks. This tutorial is mainly divided into two parts, part 1 and part 2. For part 1, we are going to simulate a digital Verilog inverter with an analog inverter and compare the outputs of them, and then we put them in a cascaded configuration (analog inverter after digital Verilog inverter) to work together as a buffer. The idea of this cascaded configuration is to make sure that, the connect rules between the digital block and analog block are established correctly. For part 2, we are going to simulate another design contains three main blocks, a digital Verilog 4-bit counter, an analog 4-bit inverter and a digital VHDL 4-bit inverter. This design is organized as follows; an external clock signal and reset signal are applied to the Verilog 4-bit counter. The output of the Verilog 4-bit counter is labeled as $count_out < 3:0 >$ and is applied as an input to the next block which is the analog 4-bit inverter. The output of the analog 4-bit inverter is labeled as vhdl_inv<3:0> and is applied as an input to the last block which is VHDL 4-bit inverter. The output of the VHDL 4-bit inverter is labeled as vhdl_out<3:0>. We are interested in these signals, count_out<3:0>, vhdl_inv<3:0> and vhdl_out<3:0>. If everything is correct, the final output, vhdl_out<3:0> will be the same as the counter output count_out<3:0>. A block diagram for this design can be found in part 2 section.

Part1:

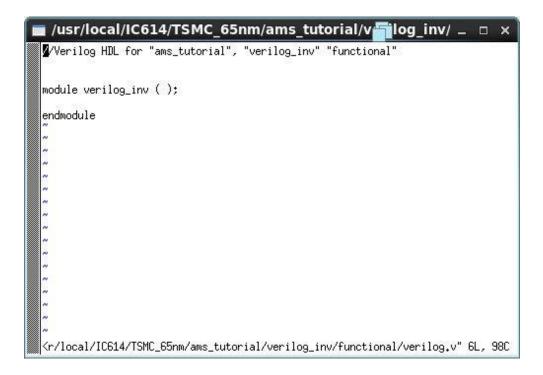
The following steps are to simulate both the digital Verilog inverter and the analog inverter and compare both outputs. Another configuration for these two inverters is to put them in cascade to work as a buffer. We use a library called ams_tutorial to include all circuits in this tutorial.

Firstly, we will make a cell view for the digital Verilog inverter.

Library Manager: WorkArea: /usr/local/IC614/TSMC_65nm	×
Elle Edit View Design Manager Help	cādence
New Library Open Ctrl+O Open (Bead-Only) Ctrl+R Category Cell	
Load Defaults	
Ogen Shell Window Ctrl+P	
Egit Ctri+X ambit ams_utortal analogLib basic	
CdSD#TechLib Cds_assertions Cds_inhcon Cds_spicelib connectLib functional	
ieee ncinternal ncmodels ncutis	
Messages LiB connectLib from the same file (defined earlier.) Warning: LiB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines LiB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines LiB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Insert UNDEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib in /usr/local/C614/TSMC_65nm/cds.lib in /usr/local/C614/TSMC_65nm/cds.lib in /usr/local/C614/TSMC_65nm/cds.lib to suppress this warning message.	
	1

	New 🚰	×
File		
Library	ams_tutorial	
Cell 🗢	verilog_inv	
View ->	functional	
Туре 🖚	Verilog	
Application		
Open with	Read HDL	
🔲 Always use t	his application for this type of file	
Library path file		
/usr/local/IC	614/TSMC_65nm/cds.lib	
	OK Cancel H	elp

After pressing the OK button, you'll get this menu



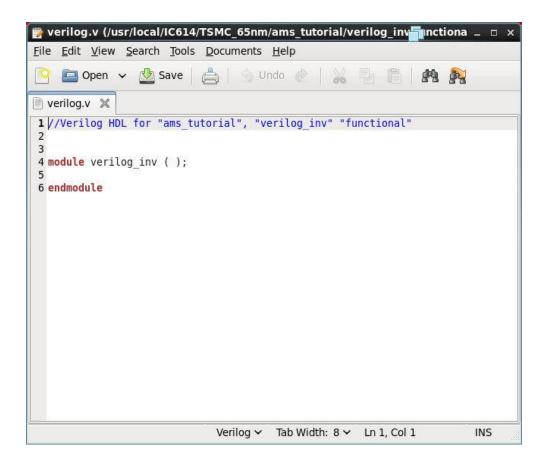
It is a little difficult to edit your code in this text editor so, we will use another text editor called "gedit" by typing the command **editor="gedit"** in the CIW window.

Ĉ	Virtuoso® 6.1.4 - Log: /root/CDS.log	×
<u>F</u> ile <u>T</u> ools <u>O</u> ptions So	nnet TSMC PDK Tools <u>H</u> elp	cādence
Loading schHDL.cxt Warning: ams.env Loading schematic.cxt Loading ahdlSck.cxt		
editor="gedit" "mouse L: 1 >	M:	R:

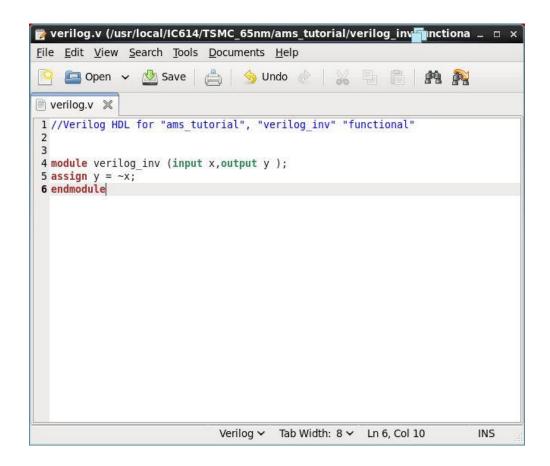
You can now see your cell view and double click on it to edit the code using gedit text editor

📓 🛛 Library Manage	er: WorkArea: /usr/local/IC614/TSMC_65nm	👘 💷 ×
Eile Edit ⊻iew Design Manager Help		cādence
	og_inv og_inv functional functional	
Messages LIB connectLib from the same file (defined earlier.) Warning: LIB connectLib from File /usr/local/C614/ToSMC_65 LIB connectLib from File /usr/local/C614/TosMC_65 linsert UNDEFINE connectLib before DEFINE connectLib in /usr/cal/C614/TSMC_65nm/cds.lib Or remove or comment out DEFINE connectLib in /usr/cal/C614/TSMC_65nm/cds.lib Or suppress this warning message.	ams/etc/connect_llb/cds.llb	

A new window will open up after double clicking your cell view as follows:



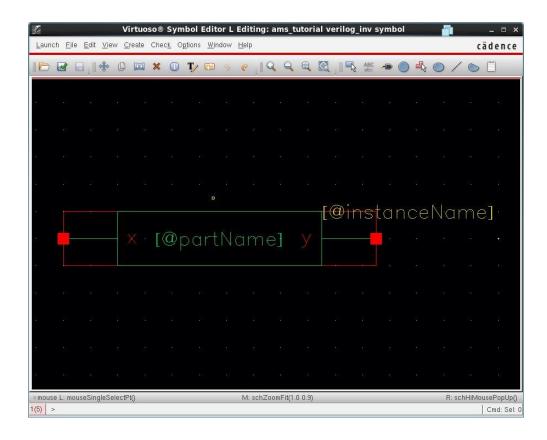
Edit the code and save it.



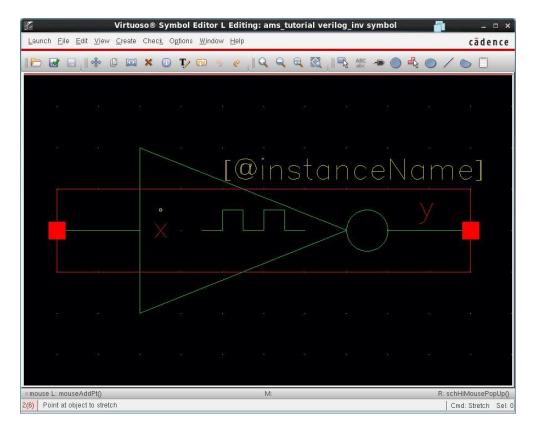
After closing it, you'll get a menu asking to create a symbol view for your circuit, press "yes".



You can use the created symbol as is or edit it.

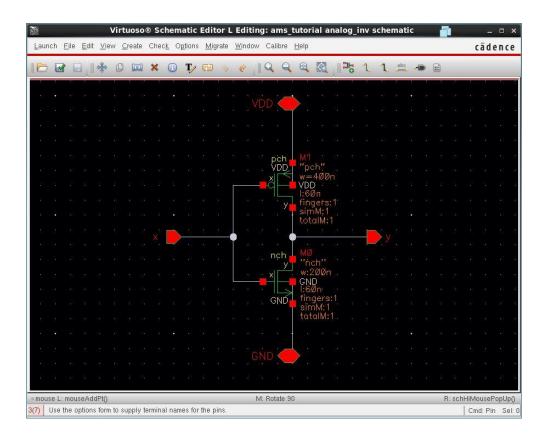


We will edit it to take the form of an inverter (optional step).



We put a something like square wave inside the symbol view to differentiate between it and the analog inverter. Now it is time to create the analog inverter.

	New I	×
File		-
Library	ams_tutorial	
Cell	analogLinv	
View	schematic	
Туре	schematic 🧧	
Application		-
Open with	Schematics L	
🔲 Always use	this application for this type of fi	le
Library path fil	e	
/usr/local/	IC614/TSMC_65nm/cds.lib	
	OK Cancel H	Help



5 Laund	ch <u>F</u>	ile	Edit	Vie					hem							Calibre			ai u	nand	·g_		cine	anne	ue	-				āde	
					-		Insta			<u>p</u> uon	<u> </u>		• <u></u>	indo Ø				e D	8		JF.	1.	1	ab	ç ,					aue	: 11 0
	•		9 9 9			1	<u>W</u> ire Wire Wire	(wi <u>d</u>	e)	W Sł	nift+V	v			/DI			•				4		30 20		19		8	2 10	-25	-
								Expre	ssion	P						1															
							<u>B</u> loc <u>C</u> ell Sold	view		В		•	F	rom (<u>C</u> ellv	iew		/i 'pcl													
							<u>N</u> ote Pa <u>t</u> cl	hcor				•		rom <u>F</u> rom <u>I</u>				v=4 /DD :60	ØØ												
					8) 8)	1	Pr <u>o</u> b M <u>u</u> lti	Shee				•				y	f S	ing simi	ers: M;1												
					10 20		CP <u>F</u>	NetS	iets	- -	×		•				•														
																nch y x		ncl v:2	ØØ⊓												
																GNĎ		GND :60 ing simi	n [.] ers												
																		ata													
	d.	12	8		84. 181	4 3	8	s: p	ŝ	10 10	8		24. 28.	33	91	Ð	a.	64). 181		91.	20 30	с. С	10	2	74 (†1	10	8	93 93	20 20	a a	40. 12.
mous	e L: m	ious	eAd	:Pt()											M: F	Rotate	90										F	l: sch	HiMo	ouseP	opU

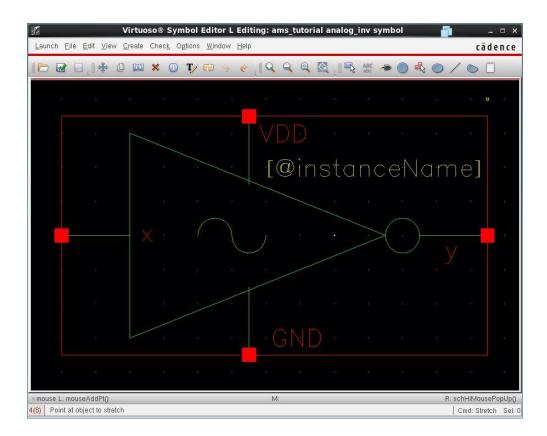
To create a symbol choose, **Create > cellview > from cellview** as follows

Press ok for the next window

	Cellview From Cellview 📑	×
Library Name	ams_tutorial	Browse
Cell Name	analog_inv	
From View Name	schematic	
To View Name	symbol	
Tool / Data Type	schematicSymbol	
Display Cellview	✓	
Edit Options	⊻	
	OK Cancel Defaults	Apply Help

	Sy	mbol Gene	eration Option	າຣ	×		
Library Name		Cell Name		View Name			
ams_tutorial	-	analog_in	7	symbol			
Pin Specificati	ons				Attributes		
Left Pins	x	List					
Right Pins	У				List		
Top Pins	ממע				List		
Bottom Pins	GND				List		
Exclude Inheri	ted Connection Pir	ns:					
🖲 None 🔾	All 🔾 Only thes	е:					
Load/Save 📃	Edit Attril	outes 🗔	Edit Labels	Edit P	roperties 🗔		
			ОК	Cancel A	Apply Help		

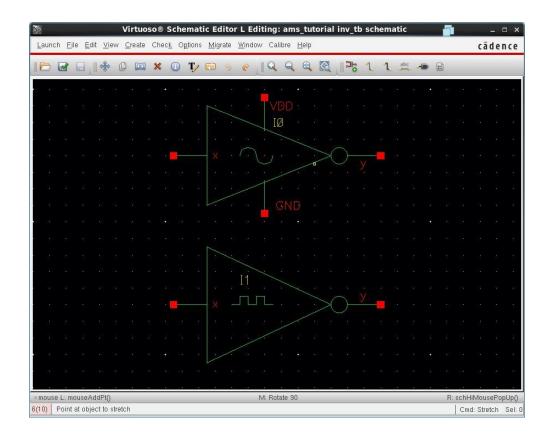
We will edit the symbol view to looks like the following (optional). We put a something like a sine wave inside the symbol to differentiate it from the digital inverter.



Next step is to make a test bench for a circuit containing both the digital Verilog inverter and the analog one.

	New 🚰	×
File		
Library	ams_tutorial	
Cell	inv_tb	
View	schematic	
Туре	schematic 🔽	
Application		
Open with	Schematics L 🛛 🧧	
🔲 Always use	this application for this type	of file
Library path fil	e	
/usr/local/	IC614/TSMC_65nm/cds.lib	
	OK Cancel	Help

ibrary	Category	Cell	View	
ms_tutorial		analog_inv	symbol	
hdlLib mbit ms_tutorial nalogLib asic ds_esperions ds_inhconn ds_sinhconn ds_spicelib onnectLib unctional see cinternal cutorial Examples Lib dilib di dib td ynopsys smc13nf torial		analog_nv inv_tb verilog_inv	symbol	
erilog ital_memory				



Put labels for input signal, Verilog output and analog output signals.

	Ad	d Wire Name	T ×
Wire Name	Net Expressio	on	
Names	input verilog	g_out analog_out	
Font Height	0.0625	Bus Expansion	🥑 off 🔾 on
Font Style	stick	Placement	🖲 single 🥥 multiple
Justification	lowerCenter	Purpose	🖲 label 🔾 alias
Entry Style	fixed offset	Bundle Display	🖲 horizontal 🔘 vertical
		Sh	ow Offset Defaults
AL Rotate)		
		Hide Ca	ncel Defaults Help

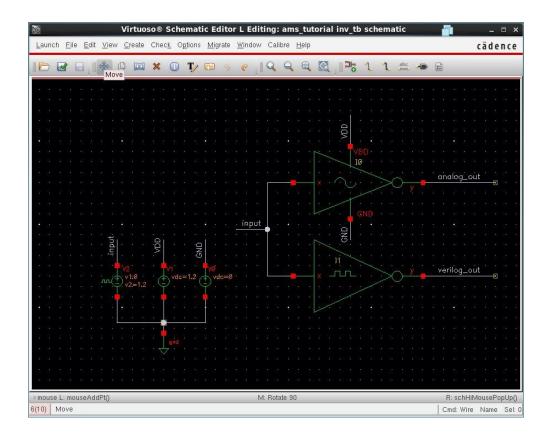
For the input source, a good choice is a periodic signal to test both, high state and low state. We use Vpulse as an input signal.

	Add	l Instance 🕤 🛛 🗙
Cell	vpulse	
View	symbol	
Names		
Array	Rows 1	L Columns 1
	🕰 Rotate 📃 🚺	Sideways 🚭 Upside Down
Frequenc	y name for 1/period	
Noise file	name	
Number o	f noise/freq pairs	0
DC voltag	le	
AC magni	itude	
AC phase	I	
XF magni	tude	
PAC mag	nitude	
PAC phas	se .	
Voltage 1		0 ¥
Voltage 2		1.2 ¥
Period		2n s
Delay tim	e	1
Rise time		
Fall time		
	•	lide Cancel Defaults Help

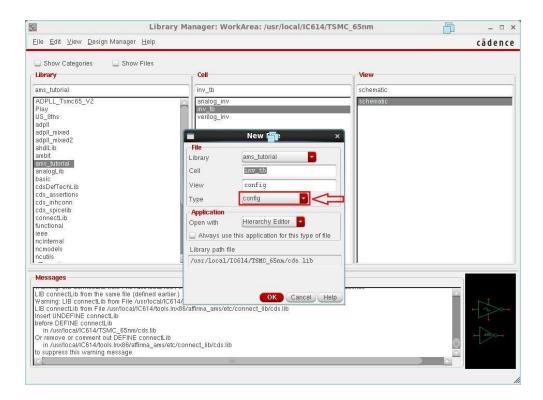
For this tutorial we use Vsupply = 1.2 v. We will use this value in connect rules also.

		Edit Object Properties 🛛 📑		×
Apply To	only curr	ent 🔽 instance 🔽		h
Show	syster	m 🗹 user 🗹 CDF		
Bro	wse	Reset Instance Labels Display		Ť
Property		Value	Display	
Library Na	ame	analogLib	off 🔽	
Cell Name)	vdc	off	
View Nam	ie	symbol	off 🔽	
Instance N	Vame	V1	off	
}		Add Delete Modify)	Ť
User Pro	perty	Master Value Local Value	Display	
Ivsignore)	TRUE	off	
CDF Par	ameter	Value	Display	
Noise file name			off	
Number of noise/fre	eq pairs	0	off	
DC voltage		1.2 ♥	off	
AC magnitude			off	
AC phase			off 🔽	
XF magnitude			off	
PAC magnitude			off 🔽	
PAC phase			off 🔽	
Temperature coeffic	cient 1		off 🔽	
Temperature coeffic	cient 2		off	
Nominal temperatur	re		off	
01	K) Ca	ancel Apply Defaults Previous	Next	Help

The final schematic after adding labels and sources looks like the following



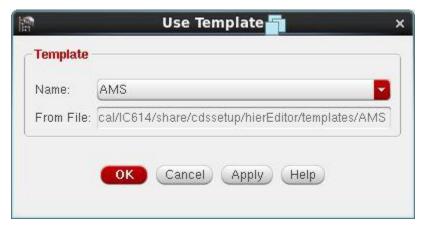
Close this schematic and create another cell view for it of type **config**. Make sure to choose type **config** (important).



You will get a menu like the following

1	Virtuoso® Hierarchy Editor	_ 🗆 X
<u>File Edit View</u>		cādence
	New Configuration	×
Top Cell	Top Cell	? 5 >
Library:		
Cell:	Library: ams_tutorial	
View:	Cell: [inv_tb	
	View:	
Open Edit	Global Bindings	
	Library List:	
Table View Cell Bindings	View List:	
	Stop List:	ed Lib List
	Constraint List:	Ed LID LIST
	Description	
	Description	
	OK Cancel Use Template Help	
Namespace.	A	
(14) >		

Press Use Template and choose AMS for Name field from the following menu.



Press **OK** and choose **view** as **schematic** for the section **Top Cell** like the following.

	New Configuration	
Top Cell		
Library:	ams_tutorial	-
Cell:	inv_tb	-
View:	schematic	-
Global Bin	dings	
Library Lis	t myLib	
View List:	avioral functional schematic veriloga vhdl vhdlams wreal	
Stop List:	spectre	
Constraint	List:	
Descriptio	n	
AMS netli Note: Pleas	onfig view template for OSS-based and Cellview-based Verilog sters in ADE. e remember to replace Top Cell Library, Cell, and View with the actual names used by your design.	1-
	OK Cancel Use Template Help	

After pressing **OK** you will get the following menu. Choose **File>Save**

1	Virtuos	o® Hierarchy	Editor: New Co	nfigur	ation (S	ave I	leeded	1 -	o x
<u>File</u>	dit <u>V</u> iew <u>P</u> lugi	ns <u>H</u> elp						cāde	nce
- 🖻 <u>O</u> pe	v Config en en (<u>R</u> ead-Only)	9 C	a □ ① 4	? & ×	Global	Bindinç	ĮS	C	? ð ×
<mark>, ∏</mark> Sav Sav Sav	ve (Needed) ve <u>A</u> s ve As <u>V</u> HDL ve As Verilo <u>g</u> pulate Library		ו		Library View Li: Stop Lis Constra	st: t:	myLib eriloga vh spectre	dl vhdlams wreal	
Sav	ve Cell <u>T</u> able Dal ve <u>D</u> efaults se Window	a ew			Seneta				
	Library	Cell	View Found	View	To Use	Inherite	ed View Lis	Inherited Lib Lis	st
	ams_tutorial	analog_inv	schematic			spectr	e spice	myLib	
200	ams_tutorial	inv_tb	schematic			spectr	e spice	myLib	
	ams_tutorial	verilog_inv	functional			spectr	e spice	myLib	
	analogLib	vdc	spectre				e spice	myLib	
	analogLib	vpulse	spectre			1.12 5.12	e spice	myLib	
	tsmcN65	nch	spectre				e spice	myLib	
	tsmcN65	pch	spectre			spectr	e spice	myLib	
III Nam	espace: CDBA	Filters: OFF							
7(14) >									

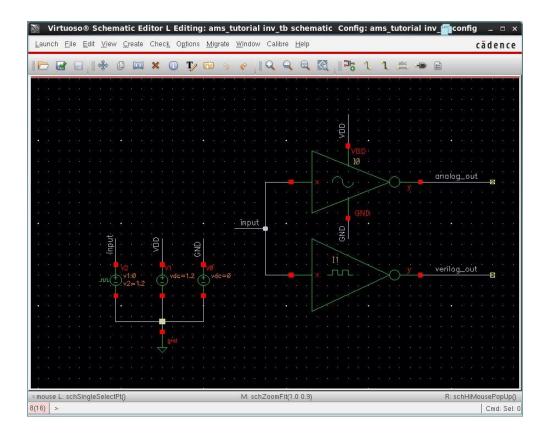
Notice that, **inv_tb** is now having two views, **config** and **schematic**.

Show Categories Show Files		Library Manager: WorkArea: /usr/lo	cal/IC614/TSMC_65nm	- -
Jurary Cell Wew ms_tutorial inv_tb config ADPLL_Tsmc65_V2 analog_inv config lay inv_tb config schematic inv_tb config inv_tb inv_tb config analog_inv inv_tb config inv_tb inv_tb config analog_inv inv_tb config invalue inv_tb config analog_inv inv_tb config invalue inv_tb config schematic invalue config invalue invalue config incontal invalue config incontal invalue config incontal <t< th=""><th>jle <u>E</u>dit <u>V</u>iew <u>D</u>esign Manager <u>F</u></th><th>lelp</th><th></th><th>cādenc</th></t<>	jle <u>E</u> dit <u>V</u> iew <u>D</u> esign Manager <u>F</u>	lelp		cādenc
ADPLL_Temc65_V2 Vary Vary Vary Vary Vary Vary Vary Vary] Show Categories 🗌 Show F Library		View	
ADPLL_Temc65_V2 Vary Vary Vary Vary Vary Vary Vary Vary	ams tutorial	inv th	config	
Play Import Import Import IS_8ths: dpll Import				
ddpl_mixed ddpl_m	Play			
	US_8ths			
udpli_mised2 wholl.bit imbit	adpll	5/05/04		
undulta imbit immodulta im				
Imbit India Utorial India Utorial	aupii_mixeuz abdii ib			
InalogLib vasic	ambit			
In discrete connect. Ib from the same file (defined earlier.) Aessages Aessages Aessages Aessages IB connectLib from the same file (defined earlier.) IB connectLib from the same file (defined earlier.) Aessages IB connectLib from the same file (defined earlier.) IB connectLib from the sa	ams_tutorial			
Ide Define Luib ids_assertions ids_signed bit in / unctional eee eee bit in / unctional eee HB connectLib from the same file (defined earlier) ************************************				
dds_asertions dds_pio(elib connectLib incritional eee dessages Aessage				
dds_inhoonn dds_spicelib connectLib unctional eeee eeee iccinternal iccoudels iccutils HE connectLib from the same file (defined earlier.) HB connectLib from the same file (defined earlier.) HB connectLib from File /usr/0.cal/C614/TSMC_65nm/cds.lib Line 18 redefines HB connectLib from File /usr/0.cal/C614/TSMC_65nm/cds.lib Line 18 redefines HB connectLib rowspice in the connectLib rowspice in the connectLib in /usr/local/IC614/TSMC_65nm/cds.lib in /usr/local/IC614/TSMC_65nm/cds.lib in /usr/local/IC614/TSMC_65nm/cds.lib in /usr/local/IC614/TSMC_65nm/cds.lib o suppress this warning message.				
connectLib unctional esee esee Hereal cicriternal cic	cds_inhconn			
unctional eee eee iccintermal iccmodels iccmodels iccutils HB connectLib from the same file (defined earlier) Varning: LB connectLib from File /usr/local/C614/TSMC_85mm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/TSMC_85mm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/TSMC_85mm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/Tools.lix88/affirma_ams/etc/connect_lib/cds.lib iset UNDEFINE connectLib Oremover or comment out DEFINE connectLib In /usr/local/C614/Tools.lix88/affirma_ams/etc/connect_lib/cds.lib in suppress this warning message.	cds_spicelib			
eee icintemal icinotedis icinutia				
IB connectLib from the same file (defined earlier.) Available in the same file (defined earlier.) IB connectLib from the same file (defined earlier.) Saming: LIB connectLib from File /usr/local/C614/TSMC_85nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/ADOIs.Inx86/affirma_ams/etc/connect_lib/cds.lib sert UNDEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib) remove or comment out DEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib) suppress this warning message.				
IB connectLib from the same file (defined earlier) Aversages IB connectLib from the same file (defined earlier) Savring: LIB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/ToSMC_65nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/ToSMC_65nm/cds.lib sefUNDEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib Dr remove or comment out DEFINE connectLib in /usr/local/C614/ToSIs.ms&/affirm_ams/etc/connect_lib/cds.lib s uppress this warning message.				
Aessages HB connectLib from the same file (defined earlier) Yarning: LIB connectLib from File /usr/local/C614/T5MC_85nm/cds.lib Line 18 redefines HB connectLib from File /usr/local/C614/T5MC_85nm/cds.lib Line 18 redefines HB connectLib from File /usr/local/C614/T5MC_85nm/cds.lib sefU NDEFINE connectLib in /usr/local/C614/T5MC_85nm/cds.lib or remove or comment out DEFINE connectLib in /usr/local/C614/tools.lmx86/affirma_ams/etc/connect_lib/cds.lib suppress this warning message.	ncmodels			
IB connectLib from the same file (defined earlier) Avrinig: LIB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines sert UNDEFINE connectLib effore DEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib or remove or comment out DEFINE connectLib in /usr/local/A014/rols.lin&6/affirm_ams/etc/connect_lib/cds.lib suppress this warning message.	ncutils			
IB connectLib from the same file (defined earlier) Avrinig: LIB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/TSMC_65nm/cds.lib Line 18 redefines sert UNDEFINE connectLib effore DEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib or remove or comment out DEFINE connectLib in /usr/local/A014/rols.lin&6/affirm_ams/etc/connect_lib/cds.lib suppress this warning message.		<u> </u>	10	
Varning: LB connectLib from File /usr/local/C614/ToSMC_65nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/tools.lnx86/affirma_ams/etc/connect_lib/cds.lib sert UNDEFINE connectLib effore DEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib)r remove or comment out DEFINE connectLib in /usr/local/C614/tools.lnx86/affirma_ams/etc/connect_lib/cds.lib s uppress this warning message.	Messages			
Varning: LB connectLib from File /usr/local/C614/ToSMC_65nm/cds.lib Line 18 redefines IB connectLib from File /usr/local/C614/tools.lnx86/affirma_ams/etc/connect_lib/cds.lib sert UNDEFINE connectLib effore DEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib)r remove or comment out DEFINE connectLib in /usr/local/C614/tools.lnx86/affirma_ams/etc/connect_lib/cds.lib s uppress this warning message.	I B connect in from the same file (defi	ned earlier)	nerodome Ento I Todomito	
sert UNDEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.lib)r remove or comment out DEFINE connectLib in /usr/local/C614/Arols/Ins8/affirma_ams/etc/connect_lib/cds.lib o suppress this warning message.	Warning: LIB connectLib from File /usi	/local/IC614/TSMC_65nm/cds.lib Line 18 redefine		
efore DEFINE connectLib in /usr/local/G614/TSMC_65nm/cds.lib or remove or comment out DEFINE connectLib in /usr/local/IC614/Ools.lnx86/affirm_ams/etc/connect_lib/cds.lib suppress this warning message.	IB connectLib from File /usr/local/IC6	14/tools.lnx86/affirma_ams/etc/connect_lib/cds.lib		
in /usr/local/IC614/TSMC_65nm/cds.lib)r remove or comment out DEFINE connectLlb in /usr/local/IC614/dois1h.ms8/dfitma_ams/etc/connect_lib/cds.lib o suppress this warning message.				
or remove or comment out DEFINE connectLib in /usr/local/C614/holos.Inx88/affirma_ams/etc/connect_lib/cds.lib s uppress this warning message.	in /usr/local/IC614/TSMC_65nm/cd	s.lib		
o suppress this warning message.	Or remove or comment out DEFINE co	onnectLib		
		na_ams/etc/connect_lib/cds.lib		
	o suppress this warning message.			

Double click on config and press OK from the following menu

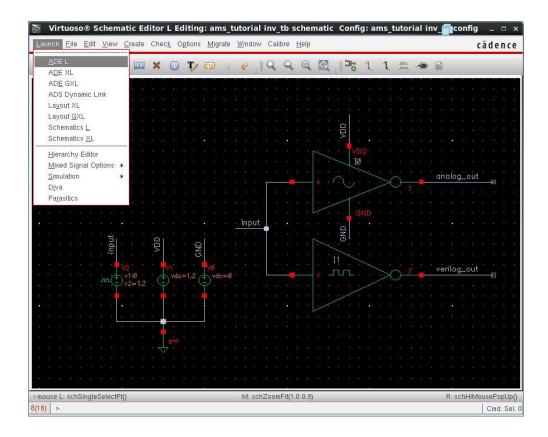
Open Configuration or Top	llView ×
Open for editing	
Configuration "ams_tutorial inv_tb config"	🔾 yes 🧕 no
Top Cell View "ams_tutorial inv_tb schematic"	🖲 yes 🔾 no
OK	Cancel Help

It will open the config view of your circuit. Choose check and Save.



Then Launch > IDE L to start simulating your circuit

(This must be done from the **config** view not schematic view)



Then choose Setup>Simulation/Directory/Host.. from the following menu

S <u>e</u> ssion	Get <u>up Analyses V</u> ariables <u>C</u>	utputs <u>S</u> imulation <u>R</u> esults <u>T</u> i	ools <u>H</u> elp	cādence
	🖞 <u>D</u> esign	Analyses		? 5 ×
Design V	Simulator/Directory/Host	Ope - Frable	Arguments	the second se
Nan	Tur <u>b</u> o/Parasitic Reduction			
	Model Libraries			On
	<u>T</u> emperature			9
	Stim <u>u</u> li			lb-s
	Simulation <u>Files</u>			
	MATLAB/Simulink			>
	<u>E</u> nvironment			
		Name/Signal/Expr -	Value Plot Save :	Save Options 🔤 🚺
mouse L:		Plot after simulation: Auto M:	Plotting mode: R	eplace

Choose simulator as ams and press OK

	rtuoso® Analo Setun Analyses V Choosing Si	ariables Oi		n Results 1	- Tools Help		o _ □ × cādence
Design Val	Simulator Project Directory Host Mode Host Remote Directory	spectre ADSsim aps hspiceD spectre UltraSim ams spectreVe UltraSimVe	rilog	tributed	faults Apply		?
			Outputs Name/Sigr	al/Expr ~	Value Plot		? P × e Options W
mouse L:				V:		a model (Cally	R:
9(17) Simu	llator/Directory/Host			402	Status: Ready	/ T=27 C	Simulator: spectre

The next step is to edit the connection rules. Choose Setup>Connect Rules

45	Virtuoso® Analog Design	Environment (1) - ams	_tutorial inv_tb 👘fig	_ = ×
S <u>e</u> ssion	Set <u>up A</u> nalyses <u>V</u> ariables <u>O</u> l	utputs <u>S</u> imulation <u>R</u> esults <u>T</u> r	ools <u>H</u> elp	cādence
Design V Nar	Simulator/Directory/Host	ion	Arguments	? ₿ × • AC • CC • Trans • • • • •
mouse L		Outputs Name/Signal/Expr - Plot after simulation: Auto M:	Value Plot Save Save O	ptions O
9(17) Co	onnect Rules	Status: Ready	T=27 C Simulator: ams(Spect	tre)Mode: batch

	ams1: Select Connect Rules 🛛 🛅	×
st of Connect	Rules Used in Simulation	
Туре	Rule Name Details	
Built-in	ConnRules_18V_full_fast Lib:connectLib View:connect	
<[NUI	
Enable	Disable Delete Rename Copy Up Down Custo	mize
Built-in and G	Customized rules	
Rules Name	e connectLib.ConnRules_18V_full_fast View	2
Description	This is the description for ConnRules_18V_full_fast Customize.	
	1 🔜 🗛 Add	
User-defined	rules for ncvlog,ncelab,ncsim	
Library	Browse	
Cell	Add	
View		
	OK Cancel Apply	Hel

You will get a menu looks like the following.

For this tutorial we use Vsupply=1.2v which is not included in the attached rules so we will choose any one and edit it using the **Customize** button. We will set **Vsup** to **1.2v** instead of **1.8v** and **vthi** to 0.8v (2*Vsup/3) and **vtlo** to 0.4v (Vsup/3). Do not forget to press the button **Change** after changing any value of these values. Vthi means threshold value for high logic and vtlo means threshold value for high logic and vthe forbidden zone.

	an	s1: Customize Built-in R	ules 📊	
Description		description for ConnRules_1	18V_full_fast	
Module E2R R2E_2 ER_bidir L2E_2_CP Bidir_2_CF	Mode	vdeita=`Vsup/64 v vsup=1.6 vdeita=`Vsup/i vdeita=`Vsup/64 vtol=`Vdeita vsup=1.6 vlo=0 tr=0.2n tf=0.	× vtol=`Vdelta/4 ttol=`Tr/20 64 tr=`Tr/20 tf=`Tr/20 rout=200 a/4 ttol=`Tr/20 tf=`Tr/20 tf=`Tr/2 a/4 ttol=`Tr/20 tr=`Tr/20 tf=`Tr/2 2n rlo=200 rhi=200 rx=40 rz= r=0.2n tf=0.2n rlo=200 rhi=201	
Mode Parameters		View connect mod	lule View defines	
Parameter	r Vsup	Value 1.8	â	
	vlo tr	D		
	tf rlo rhi	20 20		
Parameter	vsup	Value 1.2	Change	
Direction1		Discipline1		
Direction2	-	Discipline2		
Connect Reso	lutions	OK Cancel)	Apply Disciplines He	el

am	s1: Customize Built-in Rules
Chis is the c le Declarations	description for ConnRules_18V_full_fast
Mode PF PF	Parameter/Values vsup=1.2 vali=1.2 vali=0.0 vio=0 a=0.2h a=0.2h no=200 vdelta=Vsup/64 vtol=Vdelta/4 ttol=Tr/20 vsup=1.2 vdelta=Vsup/64 tr=Tr/20 tf=Tr/20 rout=200 vdelta=Vsup/64 vtol=Vdelta/4 ttol=Tr/20 tr=Tr/20 tf=Tr/2 vsup=1.2 vlo=0 tr=0.2n tf=0.2n rlo=200 rhi=200 rx=40 rz= vsup=1.2 vthi=1.2 vtlo=0.6 tr=0.2n tf=0.2n rlo=200 rhi=201 rx=40 vsup=1.2 vthi=1.2 vtlo=0.6 tr=0.2n tf=0.2n rlo=200 rhi=201 rx=40 rz= vsup=1.2 vthi=1.2 vtlo=0.6 tr=0.2n tf=0.2n rlo=200 rhi=200 rhi=201 rx=40 rz= vsup=1.2 vthi=1.2 vtlo=0.6 tr=0.2n tf=0.2n rlo=200 rhi=201 rx=40 rz= vsup=1.2 vthi=1.2 vtlo=0.6 tr=0.2n rlo=200 rhi=200 rk=40 rz= vsup=1.2 vthi=1.2 vtlo=0.6 tr=0.2n rto=0.2n rto
r	
rx rz vthi vtlo vdelta vtol	40 10M 1.2 0.6 Vsup/64 Vdelta/4
vthi	Value 0.8 Change
	Discipline1
	le Declarations Mode PF PF r r rz vthi vtlo vdelta vtol

escription Th	nis is the o	lescription for ConnRules_18▼_full_fast
Connect Module	Declarations	
Module E2R R2E_2 ER_bidir L2E_2_CPF Bidir_2_CPF		Parameter/Values vsup=1.2 vun=0.0 vi0=0.0 vi0=0 u=0.2n u=0.2n no=200 1 vdelta=`Vsup/64 vtol=`Vdelta/4 ttol=`Tr/20 vsup=1.2 vdelta=`Vsup/64 tr=`Tr/20 tr=`Tr/20 rout=200 vdelta=`Vsup/64 vtol=`Vdelta/4 ttol=`Tr/20 tr=`Tr/20 tr=`Tr/2 vsup=1.2 vlo=0 tr=0.2n tf=0.2n rlo=200 rhi=200 rx=40 rz= vsup=1.2 vthi=0.8 vtlo=0.6 tr=0.2n tf=0.2n rlo=200 rhi=201
Mode (Parameters	•	View connect module View defines
Parameter	14	
	vthi	0.8
	vtlo vdelta vtol ttol	0.6 `Vsup/64 `Vdeita/4 `Tr/20
$\leq [$	1212	
Parameter	vtlo	Value 0.4 Change
Direction1		Discipline1
Direction2	-	Discipline2

Then choose analysis and we will simulate the circuit for 10ns

💼 Choosing Analyses Virtuoso® 🔤 nalog Design 🗙
Analysis 🛛 💿 tran 🔾 dc 🔾 ac 🥥 envlp
Transient Analysis
Stop Time 10n
Accuracy Defaults (errpreset - Spectre Only)
🗹 conservative 🔲 moderate 🛄 liberal
Transient Noise
Enabled 🥑 Options
OK Cancel Defaults Apply Help

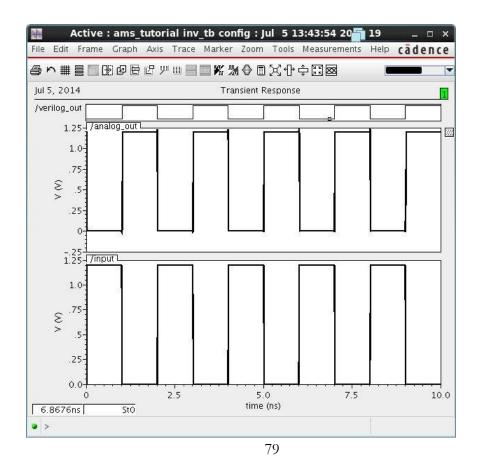
Then select the signals to be plotted Outputs>To Be Plotted>Select On Schematic

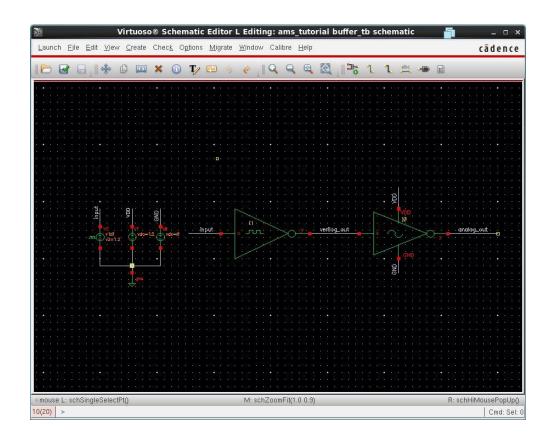
Contraction of the second seco		1) - ams_tutorial inv_tb 👘fi	and sheringer, shere
S <u>e</u> ssion Set <u>up A</u> nalyses <u>V</u> ariables Design Variables Name Value	Outputs Simulation Delete Delete To Be Saved To Be Saved Save All Mane/Signal/	Results Tools Help	Cādence
	Plot after simulation	n: Auto Plotting mode: Rep	
9(17) To Be Plotted	M:	:: Ready T=27 C Simulator: ams()	R: Spectre)Mode: batch

We are interested in these signals, input, analog_out and Verilog_out.

Virtuoso® Analog Design S <u>e</u> ssion Set <u>up</u> <u>A</u> nalyses <u>V</u> ariables <u>O</u>	Environment (1) - ams utputs <u>S</u> imulation <u>R</u> esults <u>T</u>		nv_tb	fīng _ □ × cādence
Design Variables Name ← Value	Analyses Type → Enable 1 tran	Arg. Inservative	uments	
	Outputs Name/Signal/Expr - 1 input	Value Plo	t Save	no
	2 analog_out 3 verilog_out	⊻ ⊻		no 🚺
mouse L:	Plot after simulation: Auto M:	Plot	iing mod	e: Replace
9(17) Select On Schematic	and the second	T=27 C S	imulator:	ams(Spectre)Mode: batcl

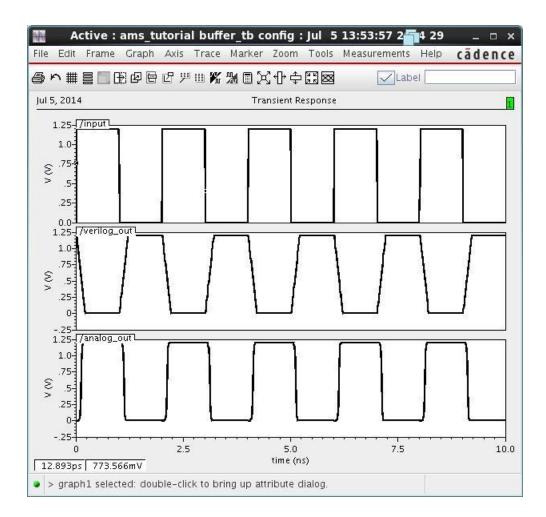
Then Run Simulation. The following are the results from both the analog inverter and the Verilog digital inverter and they are the same.





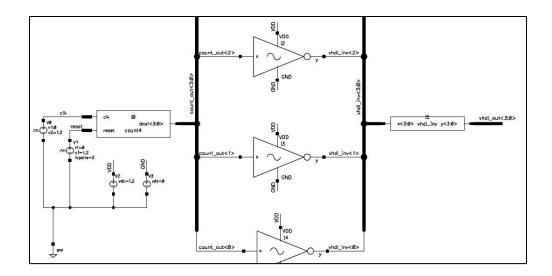
We will now try the cascaded configuration of both inverters to act as a buffer.

The simulation result can be found in the following figure. One can simply notice that, the analog_out signal is the same as the input signal. (The operation of a typical buffer).



Part2:

In this part we are going to simulate a design consisting of 3 main blocks, digital Verilog 4-bit counter, analog 4-bit inverter and digital VHDL 4-bit inverter. The block diagram of this circuit is as follows



The analog inverters are taken **from Part 1** and the codes for Verilog counter and VHDL inverter is as follows

```
module count4(input clk,reset,output[3:0] dout);
reg[3:0] count_reg;
wire[3:0] count_next;
//state register
always@(posedge clk,posedge reset)
    if(reset)
    count_reg <= 0;
    else
    count_reg <= count_next;
//next state logic
assign count_next = count_reg+1;
//output logic
assign dout = count_reg;
endmodule</pre>
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity vhdl_inv is
port (x: in STD_LOGIC_VECTOR (3 downto 0);
y: out STD_LOGIC_VECTOR (3 downto 0));
end;
architecture behavioral of vhdl_inv is
begin
y <= not x;
end;
```

We will start by importing both the Verilog counter and the VHDL inverter into virtuoso. The following figure is to import the Verilog counter.

8	Library Manager: Work	Area: /usr/local/IC614/TSM	IC_65nm	🗂 _ 🗆 ×
<u>File E</u> dit <u>V</u> iew		-		cādence
and the second	New			
Show Catego	In the second seco			
Library	Import Export	EDIF200	View	
ams_tutorial	Refresh	Verilog	config	
ADPLL_Tsmc65 Play		- Spice	config schematic	
US_8ths adpli	Bookmarks	<u>D</u> EF		
adpll_mixed adpll_mixed2	📓 <u>1</u> ams_tutorial buffer_tb config	<u>L</u> EF		
aupii_iiiixeuz ahdiLib	🛥 2 ams_tutorial inv_tb schematic	Stream Netlist View		
ambit	🛥 <u>3</u> ams_tutorial buffer_tb schematic	Netlist view		
ams_tutorial analogLib	🛓 4_ams_tutorial inv_tb config			
Dasic	🛥 <u>5</u> ams_tutorial analog_inv schematic			
dsDefTechLib	📓 <u>6</u> ams_tutorial verilog_inv symbol			
cds_inhconn	7 ams_tutorial verilog_inv functional			
ods_spicelib connectLib	* 8 ADPLL_Tsmc65_V2 Overall_Test schematic			
functional	🛥 <u>9</u> adpll dcv2_tb schematic			
ieee ncinternal	🛥 0 ADPLL_Tsmc65_V2 dcv_tb schematic			
ncmodels	<u>C</u> lose Data			
	Exit	- Log: /root/CDS.log		_ □ ×
Messages —	Eile Tools Options Sonnet TSMC PDK Tools He	elp	cā	idence
LIB connectLib	in /usr/local/IC614/TSMC_65nm/cds.lib			
LIB connectLib	Or remove or comment out DEFINE connectLib in /usr/local/IC614/tools.lnx86/affirma :			
Insert UNDEFIN	to suppress this warning message.	ans/ecc/connect_110/cds.110		
in /usr/local	<			
Or remove or d in /usr/local				D
	mouse L:	M:		R:
	1 2			

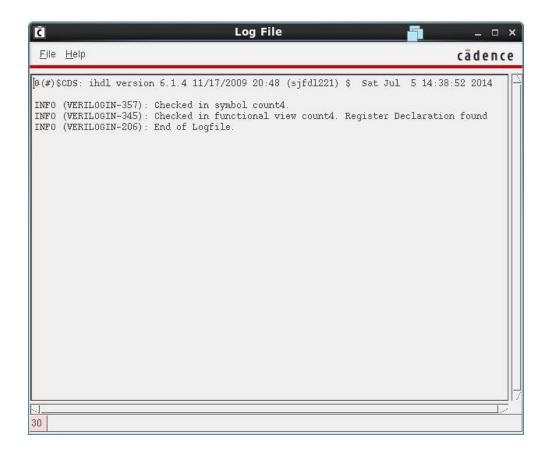
Choose Target Library and Verilog file as follows

	Verilog In 👘	
Import Options Glob	oal Net Options Schematic Generation	Options
File Filter Name		
/ count4.v		
vhdl_inv.vhd		
/root/Desktop/ams_tu	itorial/	
Target Library Name	ams_tutorial	Browse
Reference Libraries	basic	DIOWSE
	-	
Verilog Files To Import	<pre>'root/Desktop/ams_tutorial/count4.</pre>	▼ Add
-f Options		Add
-v Options		Add
-y Options		Add
Library Extension		
Library Pre-Compilation	n Options	
Pre Compiled Verilog Lik	prary	
HDL View Name	hdl	
Target Compile Library N	lame	Browse
Compile Verilog Library	Only 🔲	
Ignore Modules File		Add
Import Modules File		Add
Import Structural Modules	As schematic	
- Structural View Names	s ————	
	Cancel Defaults Apply Lo	ad)(Save)(He

You will get a menu like the following after pressing **OK**. You may press **Yes** to view the logfile.



The logfile looks like the following. Reporting that your import process has been completed.



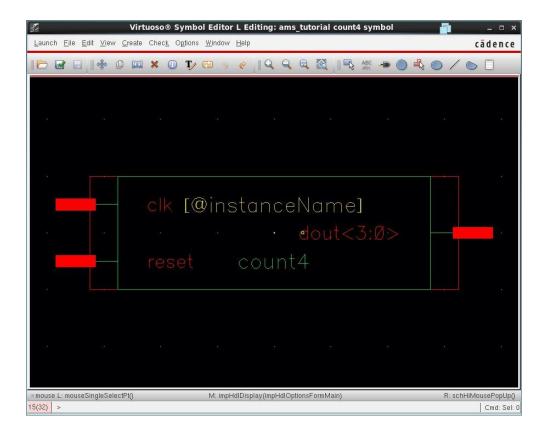
Now check your target library (**ams_tutorial** in out example) to find the Verilog counter **count4** with two views, **functional** (The Verilog code) and **symbol**.

🕺 Lil	orary Manager: WorkArea: /usr/lo	cal/IC614/TSMC_65nm	i _ = ×
<u>File E</u> dit <u>V</u> iew <u>D</u> esign Manager <u>H</u> elp			cādence
Show Categories Show Files Library ams_tutorial ADPLL_Tsmc65_V2 Play US_6ths adpll adpll_mixed adpl_mixed2 ahdiLib ambit amalogLib basic cds_offech_Lib cds_estions cds_inhcom cds_sicelib connect.ib functional leee ncinternal ncmodels ncmtlis	Cell count4], analog_inv buffer_tb cds_globals count4 inv_tb verilog_inv	Tunctional symbol	
Messages			
LIB connectLib from the same file (defined Warning: LIB connectLib from File /usr/locz LIB connectLib from File /usr/locz/l/C614/t0 Insert UNDEFINE connectLib before DEFINE connectLib in /usr/local/C614/T8MC_85nm/cds.lib Or remove or comment out DEFINE connect in /usr/local/C614/tools.lnx86/affirma_ar to suppress this warning message.	I//C614/TSMC_65nm/cds.lib Line 18 redefine ools.Inx66/affirma_ams/etc/connect_lib/cds.lib	8	

Double click on functional to check the code.

jle <u>E</u> dit <u>V</u> iew <u>D</u> esign Manager <u>H</u> elp			cāden
🗌 Show Categories 🛛 🔲 Show Files			
Library	Cell	View	
· · · · · · · · · · · · · · · · · · ·			
ams_tutorial	count4	functional	
ADPLL_Tsmc65_V2	analog_inv	functional	
Play JS 8ths	buffer_tb	symbol	
dpll	cds_globals count4		
dpll mixed	iny th		
dpll_mixed2	verilog.v (/usr/local/IC614/TSMC_6	5nm/ams_tutorial/c🔚nt4/func 💷 🗆	×
hdlLib	ile Edit View Search Tools Docume	ants Help	
mbit ms tutorial			
nalogLib	🍳 📠 Open 🐱 🎂 Save 📇	🇓 Undo 🙋 🛛 💥 🖏 👘 🛉 🏘	~
asic			
dsDefTechLib	verilog.v 🗶		
ds_assertions ds_inhconn	1// Created by ihdl		
	2 module count4(input clk,reset,ou	tnut[3:0] dout):	
onnectLib	3 reg[3:0] count reg;	cpuc[5.0] dduc/,	
Inctional	4 wire[3:0] count next;		
	5		
cmodels	6 //state register		
nutile	7 always@(posedge clk,posedge reset	t)	-
	<pre>8 if(reset)</pre>		-
	<pre>9 count_reg <= 0;</pre>		
	0 else		
/arning: LIB connectLib from File /usr/local/IC614	<pre>1 count_reg <= count_next; 2 //next state logic</pre>		
IB connectLip from File /usi/local/ICb14/tools.inx	3 assign count next = count reg+1;		
	4 //output logic		
in /usr/local/IC614/TSMC_65nm/cds.lib 1	5 assign dout = count reg;		~
Pr remove or comment out DEFINE connectLib		Width: 8 V Ln 1, Col 1 INS	
in /usr/local/IC614/tools.Inx86/affirma_ams/etc/	veniog 🗸 Tab		
Suppress and warming message.	100		

Double click on symbol view to check the symbol. You may need to edit it, we will just leave it as is.



Next step is to import the VHDL 4-bit inverter.

	Library Manager: Work	Area: /usr/local/IC614/	TSMC_65nm	_ □
ile <u>E</u> dit <u>V</u> iew		-		cāden
] Show Catego	New ▶ 			
Library	Import •	EDIF200	View	
ams_tutorial	Export •	Verilog	symbol	
DPLL_Tsmc65 lav		V <u>H</u> DL	functional symbol	
s_8ths dpll	Bookmarks •	- Spice DEF	symbol	
dpll_mixed dpll_mixed2 adLib mbit ms_tutorial aalogLib asic dsDefTechLib s_assertions ds_spicelib onnectLib nctional ee cinternal	1 ams_tutorial count4 symbol 2 ams_tutorial count4 functional 3 ams_tutorial buffer_tb config 4 ams_tutorial buffer_tb schematic 5 ams_tutorial buffer_tb schematic 6 ams_tutorial inv_tb config 7 ams_tutorial analog_inv schematic 9 ams_tutorial verilog_inv symbol 9 ams_tutorial verilog_inv functional 0 ADPLL_TSmc65_V2 Overall_Test schematic	LEF Stream Netlist View		
cmodels cutils	<u>C</u> lose Data Exit	- Log: /root/CDS.log	×	
lessages		elp	cādence	
arning: LIB c IB connectLib	to suppress this warning message. INPO (VERILOGIN_GUI-13): Verilog Import proc INPO (VERILOGIN_GUI-15): Verilog Import comp Warning: ams.env L: awvMouseSingleSelectPICB() >	leted. Look at logfile,	./verilogIn.log, for process exe	

Choose your target library (ams_tutorial in our example) and VHDL file. Leave all other fields as they are.

8	VHI	DL Import 📑	× 🗇 – • ×
<u>F</u> ile <u>E</u> dit ⊻iew <u>D</u> esign Man	Import Options Schematic Gen	eration Ontions	cādence
Show Categories :	File Name	Target Library Name 3 ams_tutorial vhdl inv.vhd	
ams_tutorial	vhdl_inv.vhd	2	
ADPLL_Tsmc65_V2 Play US_6ths adpl1 mixed	1	Add >> «< Remove	
adpll_mixed2 ahdlLib ambit ams tutorial	/root/Desktop/ams_tutorial/*.v	zhd	
analogLib basic	Import Structural Architectures As	schematic	
cdsDefTechLib	Reference Libraries	basic US_8ths ieee std	
cds_assertions cds_inhconn	Symbol View Name	symbol	
cds_spicelib connectLib	Overwrite Existing Views	⊻	
functional	Case Sensitive Symbol Matching	✓	
ncinternal	User Specified Standard Libraries		
ncmodels noutils	Maximum Number of Errors	10	X
Messages <u>File</u> Tools	Compile VHDL Views After Import Compiler Options		cādence
LIB connectLib	VHDL WORK Library Name		pr process exe
Insert UNDEFIN before DEFINE in /usr/local	Summary File	./vhdlin.summary	
Or remove or c	Compatibility Option		
to suppress this mouse L: aw	v93 Option		R: vhdlHilmport)
	Power		
	Net Name	Value (1)	

After pressing **OK** you will get the following menu, press **Yes** to check the logfile.



VHDL ToolBox Log File	_ = ×
Elle <u>H</u> elp	cādence
NFO (VHDLIN-284): VHDL In Run Summary (#)\$CDS: vhdlin version 6.1.4 11/17/2009 20:49 (sjfnl007) \$ Sat Jul 5 14:46:25 2014	-
NFO (VHDLIN-238): Processing VHDL source file: /root/Desktop/ams_tutorial//vhdl_inv.vhd. NFO (VHDLIN-264): Done. NFO (VHDLIN-244): Vhdl Design Unit: vhdl_inv : Entity NFO (VHDLIN-243): Created symbol view of type symbol. NFO (VHDLIN-245): Created entity view of type Vhdl NFO (VHDLIN-255): Vhdl Design Unit: vhdl_inv.behavioral : Architecture NFO (VHDLIN-245): Created behavioral view of type Vhdl	
<pre>NFO (VHDLIN-255): -> (/root/Desktop/ams_tutorial//vhdl_inv.vhd,9) Behavioral expr</pre>	ession.
NFO (VHDLIN-229): Number of file(s) processed in this round is 1	
NFO (VHDLIN-231): Number of file(s) successfully imported in this round is 1	

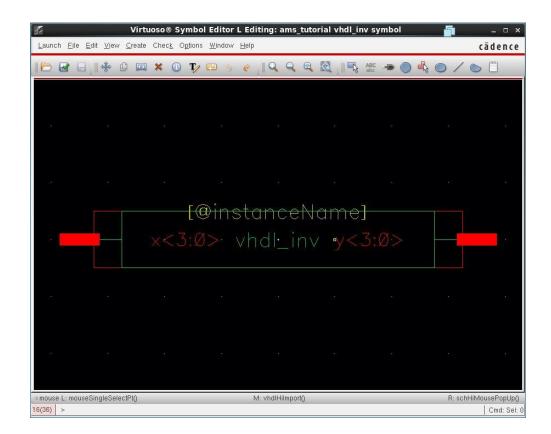
Now you can check your cell, named **vhdl_inv**.

🔲 Show Categories 🛛 🔲 Show F			
Library	Cell	View	
ams_tutorial	vhdl_inv	symbol	
ADPLL_Tsmc65_V2 Play US_6ths adpll_mbxed adpl_mbxed adpl_mbxed2 adpl_mbxed2 addLib ambit malogLib basic cdspeffechLib cds_assertions cds_plicelib connectLib functional isee ncintermal ncmdels	analog_inv buffer_tb cds_globals count4 inv_tb verilog_inv vhdl_inv	behavioral entity symbol	
Messages			
LIB connectLib from the same file (de Warning: LIB connectLib from File /us/ LIB connectLib from File /us/local/C Insert UNDEFINE connectLib before DEFINE connectLib in /us/local/C614/TSMC_65nm/c Or remove or comment out DEFINE c in /us/local/C614/Atools.inx68/affin o suppress this warning message.	r/local/IC614/TSMC_65nm/cds.lib Line 18 redefines 614/tools.lnx86/affirma_ams/etc/connect_lib/cds.lib ds.lib onnectLib		

Double click on behavioral view and entity view to check the code.

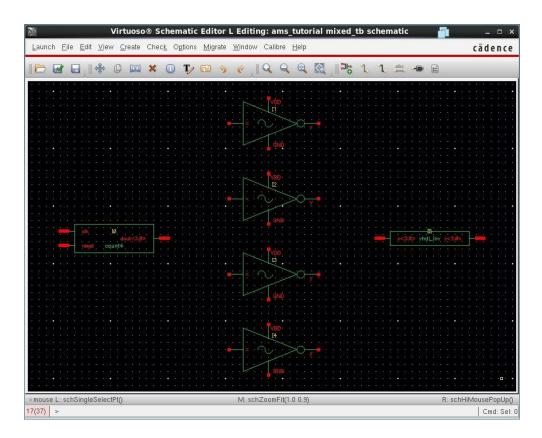
ile <u>E</u> dit <u>V</u> iew <u>S</u> earch <u>T</u> ools <u>D</u> oo	uments <u>H</u> elp	
🍳 🔄 Open 👻 🆄 Save 📋	Undo 🗞 📓 🛍 🏘 🕅	
(sjfnl007) \$ 2 on Sat Jul 5 14:46:25 201	n version 6.1.4 11/17/2009 20:49	
3 5 <mark>architecture</mark> behavioral of vh 6 begin 7 y <= not x; 8 end;	dl_inv is whdl.vhd (/usr/local/IC614/TSMC_65nm/ams_tutorial/vhdl_mr/entity File Edit View Search Tools Documents Help	() _ □ ×
VHDI	<pre> whdL.vhd % 1 Created by @(#)\$CDS: vhdlin version 6.1.4 11/17/2009 20:49 (sjfnl007) \$ </pre>	4
criticuens critils lessages	3 4 5 library IEEE; 6 use IEEE.STD_LOGIC_1164.all; 7 entity whol inv is	
IB connectLib from the same file (defined e avaning: LIB connectLib from File /usr/local /C614Aou sert UNDEFINE connectLib fore DEFINE connectLib fore DEFINE connectLib in /usr/local/IC614/ToMC_65nm/Cds1ib in /usr/local/IC614/hools.inx68/affirm_am suppress this warning message.	<pre>ICG14/ 9 y: out STD_LOGIC_VECTOR (3 downto 0)); 10 end; 11 Lib</pre>	
C	VHDL Tab Width: 8 Ln 4. Col 1	INS

You can edit the symbol. We will use it as it is.



Next step is to make the test bench for the block diagram mentioned in the beginning of this section. We will name it **mixed_tb** (optional).

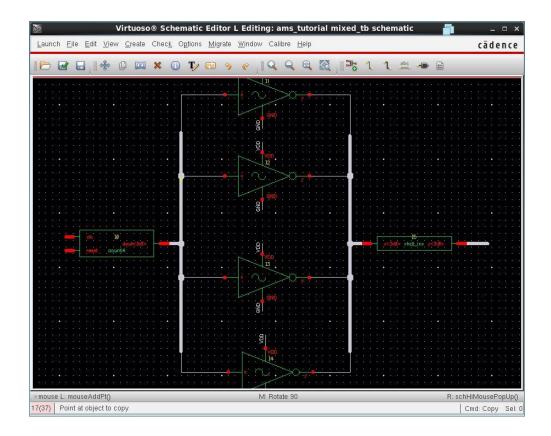
	New 🚘	×
File		
Library	ams_tutorial	
Cell	mixed_tb	
View	schematic	
Туре	schematic 🗧	
Application		
Open with	Schematics L	
🔲 Always use	this application for this ty	pe of file
Library path fil	e	
/usr/local/1	C614/TSMC_65nm/cds.l	ib
	OK Canc	el Help



Choose **create>wire (wide)** to create buses.

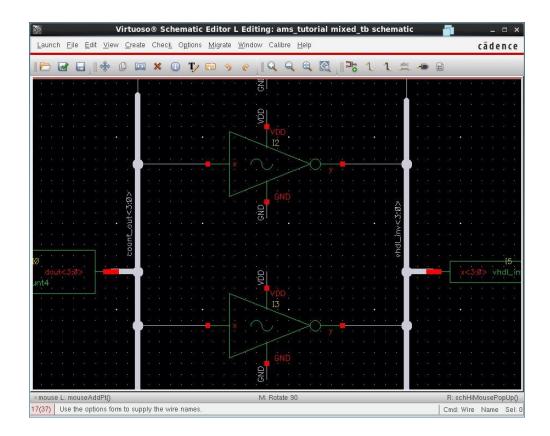
aunch <u>File E</u> dit <u>V</u> iew <u>Create</u> Chec <u>k</u> O <u>p</u> ti	ns <u>Migr</u> ate <u>W</u> indow Calibre <u>H</u> elp	cādenc
🗁 🛃 🔒 🛊 🕻 🍡 Instance	୬ ୧ । ସ୍ୟୁ ସ୍ୟୁ 🛛 🎝 ୪	l 1 些 🛥 🔒
	V V	
	Shift+W	
	Line Line	
≝ Wire Na <u>m</u> e		
Net Expression		
≫ . <u>P</u> in		
	áno l	
<u>C</u> ellview	te a and at any site a and a and a site	
<u>S</u> older Dot		
<u>N</u> ote		
Patchcord	a a construction of the second se	
Pr <u>o</u> be	, = × · · · · > ○ → • · · · · · ·	
MultiSheet		THE SOLE NOTION TO A THE ACCURATE A SUB-
CPE NetSets		15
dout<3:0>		
	, Noo	
	L B	
	i and is another and at any property and its and another and	
	an a maging a star and a star a star a star a star	
	· · · · · · · · · · · · · · · · · · ·	
	na ana ang 🖂 n 🔓 GND na ang ang ang ang	
nouse L: schSingleSelectPt()	M: schZoomFit(1.0 0.9)	R: schHiMousePopU

Connect these blocks together as follows



Add labels for buses by pressing l.

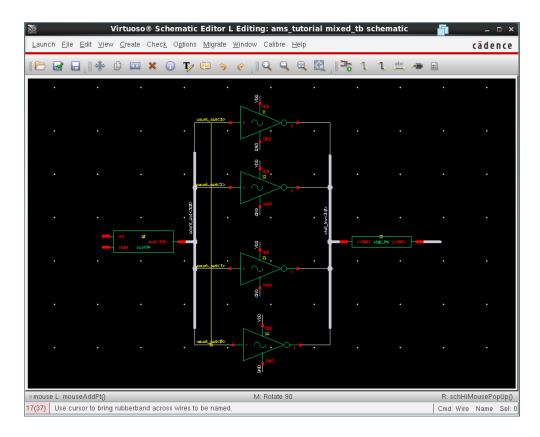
	Ado	l Wire Name	1 ×
Wire Name	Net Expression	n	
Names	count_out<3:0	> vhdl_inv<3:0>	
Font Height	0.0625	Bus Expansion	🖲 off 🔾 on
Font Style	stick	Placement	🧕 single 🥥 multiple
Justification	lowerCenter	Purpose	🖲 label 🔾 alias
Entry Style	fixed offset	Bundle Display	🖲 horizontal 🔘 vertical
		Sh	ow Offset Defaults
AL Rotate)		
		(Hide) Ca	ncel Defaults Help

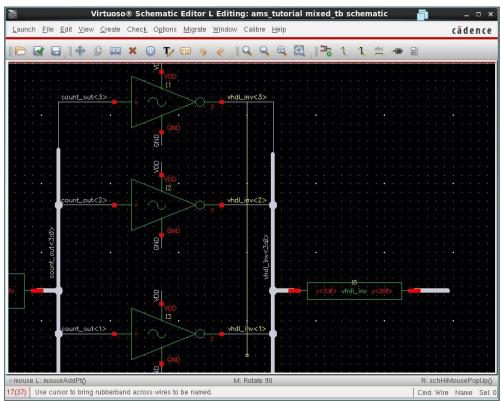


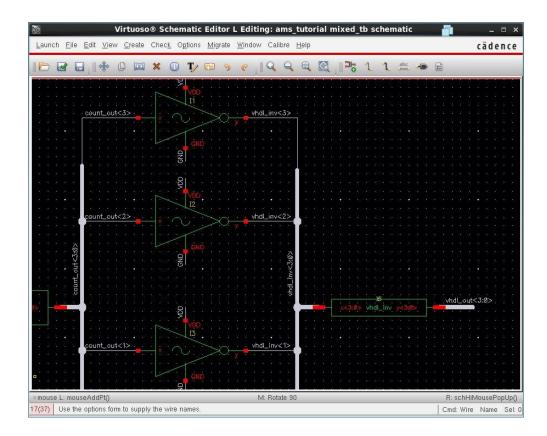
Add labels for the wires connected to the buses.

	Ad	d Wire Name	1 ×
Wire Name	Net Expressio	in	
Names	count_out<3:0)> [6
Font Height	0.0625	Bus Expansion	🔾 off 🖲 on
Font Style	stick	Placement	🔾 single 💽 multiple
Justification	lowerCenter	Purpose	🧕 label 🔾 alias
Entry Style	fixed offset	Bundle Display	🖲 horizontal 🔘 vertical
		Sh	ow Offset Defaults
AL Rotate)		
		Hide Ca	ncel Defaults Help

After pressing **OK**, left click on the first wire and move to the last wire across the wires between them and finally left click on the last wire.







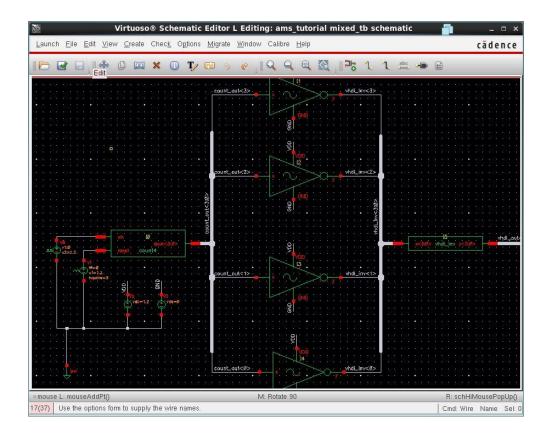
Add sources to your design clk, reset, VDD and GND. For the clk we use Vpulse with period 2ns.

	Ado	i Instance 📊 🛛 🗙 🗙
Cell	vpulse	
View	symbol	
Names	[
Array	Rows	1 Columns 1
	AL Rotate	Sideways 🛛 🚭 Upside Down
Frequent	cy name for 1/period	
Noise file	e name	
Number	of noise/freq pairs	0
DC volta	ge	
AC magr	nitude	
AC phas	e	
XF magn	itude	
PAC mag	gnitude	
PAC pha	ise	
Voltage ⁻	1	0 V
Voltage á	2	1.2 ¥
Period		2n s
Delay tim	18	
Rise time		
Fall time		
		Hide Cancel Defaults Help

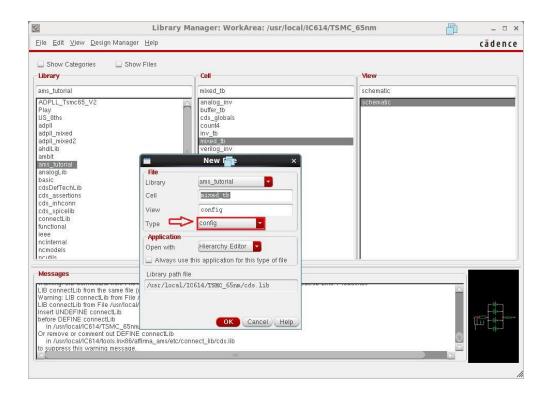
For the reset signal, we use Vpwl with the following configuration for an active high reset.

	Ad	d Instance 💼		×
Library	analogLib		Browse	6
Cell	vpwl			
View	symbol			
Names				
Array	Rows	1 Columns	1	
	🕰 Rotate 🔰 🔄	Sideways 🛛 🗲 Ups	ide Down	
Frequenc	y name for 1/period			
Number o	f pairs of points	3		
Time 1		0 s		
Voltage 1		1.2 V		
Time 2		ln s		
Voltage 2		1.2 V		
Time 3		1.1n s		
Voltage 3		0 8		
Noise file	name			
Number o	f noise/freq pairs	0		
DC voltaç	le			
AC magn	itude			
AC phase	1			
) (T		Hide Cancel De	faults He	elp)

The final schematic after adding sources and labels is as follows



Check and Save then close this schematic and create a config view for it as we did for Part 1.



		(? # ×	Global	Binding	IS	(7 5
						·		
ams_tutorial				Library	List:	myLib		6
mixed_tb				View I	et.	ariloga vh	di uhdiams ureal	16
cohomotic			_	VICW L	JI.	stillega viti	ar vitalans wiear	0
schematic			-	Stop Li	st:	spectre		16
Edit				Constra	int List:			
ams tutorial		functional			spectr	e spice	mvLib	
cano_catoria		Jonomouro			opeen	c opiec	inyens	
ams_tutorial	count4				1.05*	e spice	myLib	
ams_tutorial	count4 mixed_tb	schematic			spectr	e spice	myLib	
ams_tutorial ams_tutorial	count4 mixed_tb vhdl_inv	schematic behavioral			spectr spectr	e spice e spice	myLib myLib	
ams_tutorial ams_tutorial analogLib	count4 mixed_tb vhdl_inv vdc	schematic behavioral spectre			spectr spectr spectr	e spice e spice e spice	myLib myLib myLib	
ams_tutorial ams_tutorial analogLib analogLib	count4 mixed_tb vhdl_inv vdc vpulse	schematic behavioral spectre spectre			spectr spectr spectr spectr	e spice e spice e spice e spice	myLib myLib myLib myLib	
ams_tutorial ams_tutorial analogLib	count4 mixed_tb vhdl_inv vdc	schematic behavioral spectre			spectr spectr spectr spectr spectr	e spice e spice e spice	myLib myLib myLib myLib myLib	
	mixed_tb schematic Edit	mixed_tb schematic Edit View Tree View ndings Library Cell	mixed_tb schematic Edit View Tree View ndings Library Cell View Found	mixed_tb schematic Edit View Tree View ndings Library Cell View Found View	mixed_tb View Li schematic Stop Lis Edit Constra View Tree View Indings Library Cell View Found	mixed_tb View List: schematic Stop List: Edit Constraint List: View Tree View ndings Library Library Cell View Found View To Use	mixed_tb View List: sriloga vh schematic Stop List: spectre Edit Constraint List: Constraint List: View Tree View Minute Ibinary Cell View Found View To Use	mixed_tb View List: priloga vhdl vhdlams wreal schematic Stop List spectre Edit Constraint List: Tree View View Tree View View To Use

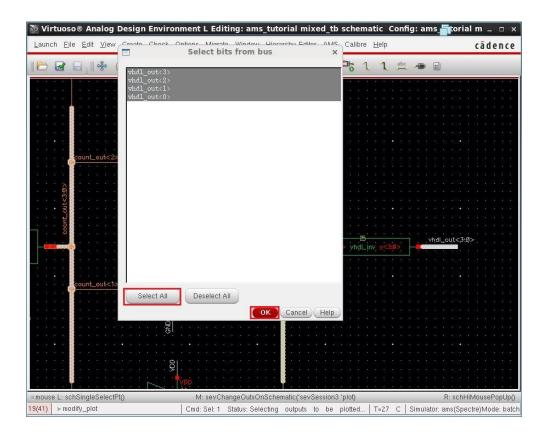
Choose the **ams simulator** as we did for **Part 1** and edit **connect rules** then select signal to be plotted.

Virtuoso® Analog Design			
Session Setup <u>A</u> nalyses <u>V</u> ariables Design Variables Name Value	Outputs Simulation Delete Delete To Be Sayed Image: Sayed Save All Image: Sayed All Outputs Name/Signal/	0 10n conservative Select On Schemati Select From HED Add To <u>R</u> emove From	· · · · · · · · · · · · · · · · · · ·
	Plot after simulation	in: Auto 🎴 Plottii	ng mode: Replace
mouse L: 20(42) Remove From	M: Statu:	::Ready T=27 C Sir	R: mulator: ams(Spectre)Mode: batch

We are interested in plotting these signals, count_out<3:0>, vhdl_inv<3:0> and vhdl_out<3:0>.

	Select bits from b
<pre>count_out<3> count_out<2></pre>	
count_out<1> count_out<0>	
Select All	Deselect All

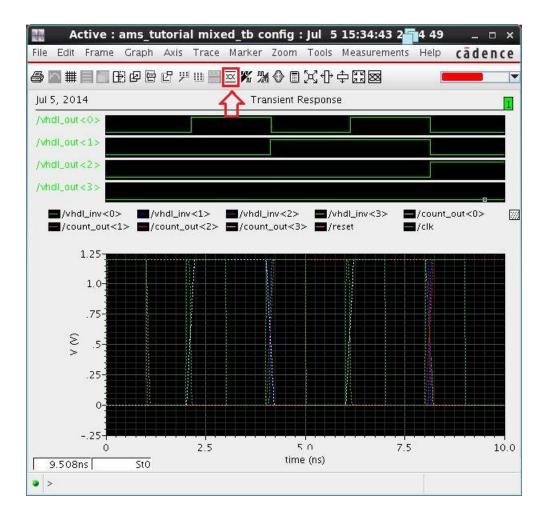
	Select bits from b
vhdl_inv<3> vhdl_inv<2> vhdl_inv<1> vhdl_inv<0>	
Select All	Deselect All



Then Run Simulation.

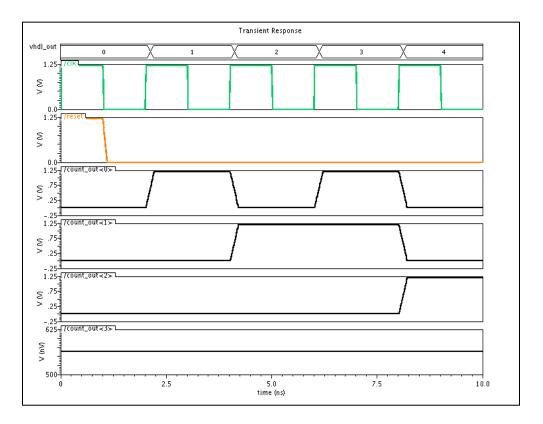
10n conservative	Argum Piot		Save Options	
	_	Save	Save Options	
- Value	_	Save	Save Options	
- Value	_	Save		
	V		no	
	<u> </u>		no	- 1
			no	
	¥		no	
			no	
			no	
	V		no	
			no	
	V		no	
	V		no	
	V		no	
			no	
	V		no	
				Image: Constraint of the sector of the se

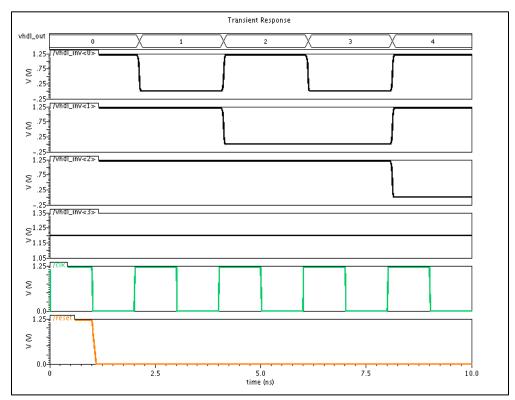
From the results figure, we can select the digital signals of vhdl_out<3:0> and convert them to a bus.



Bit 🗸	Signal	Results Directory	
) (lsb)	/vhdl_out<0>	/root/simulation/mi	
1	/vhdl_out<1>	/root/simulation/mi	
2	/vhdl_out<2>	/root/simulation/mi	
3 (msb)	/vhdl_out<3>	/root/simulation/mi.	
alpha Sort			
alpha Sort Bus Name Radix Hex			

Simulation results are as follows





For more details, please check the following videos: http://youtu.be/QzHU-FyISIo http://youtu.be/AN641gYyFj4 http://youtu.be/SLAEHGnrwuE

Conclusion

In this tutorial we have learned how to use AMS simulator to simulate digital Verilog codes with digital VHDL code and analog blocks. This tutorial assumes that, you have the AMS tool integrated with Cadence Virtuoso. For further information, please check the video notes of this tutorial in the reference page.

Appendix B

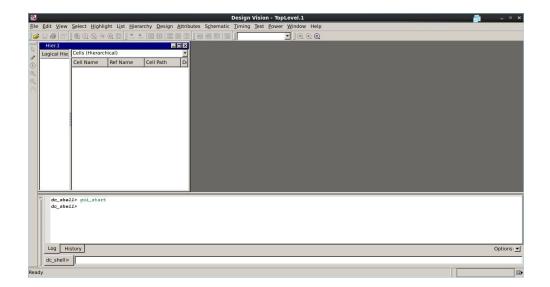
LOGIC SYNTHESIS

This tutorial presents the main steps to perform the logic synthesis of a digital Verilog 4-bit counter with the Synopsys Design Vision and Design Compiler tools. We divide this tutorial into two parts, part 1 and part 2. For part 1, we make all the steps using the Design Vision graphical environment. For part 2, we make the same steps using a TCL (Tool Command Language) script.

Part 1:

First step is to start the Design Vision graphical environment. In command window type:

dc_shell and press enter then type gui_start. You should get the following menu:



Next step is to edit setup menu to add your target library and link library; the libraries to which your design will be mapped. To do this step choose **File-> Setup** and from the following menu remove default libraries and add yours.

Categories	Defaults	
- Defaults Variables	Search path:	
	Physical library:	
	Link library: * KIT/synopsys/uk65lscllmvbbr_120c25_tc.db	
	Target library: * KIT/synopsys/uk65lscllmvbbr_120c25_tc.db	•••
	Symbol library: * your_library.sdb	
	Synthetic library:	1011
	* = required	

The following step is to analyze your Verilog code to check it and see if it is synthesizable. Choose **File-> Analyze** from the main menu and using the **Add** button select all your Verilog sources to be analyzed then press **OK**.

	Analyze Desig	×
Fi <u>l</u> e names	in analysis order:	
	m/Desktop/count4_new/count4.v	A <u>d</u> d
		D <u>e</u> lete
		•
		*
<u>F</u> ormat:	VERILOG (v)	•
<u>W</u> ork library	/: WORK	*
	Create new library if it does n	ot exist Cancel

Next step is to elaborate your design. The elaboration phase performs a generic pre-synthesis of the analyzed model. It essentially identifies the registers that will be inferred.

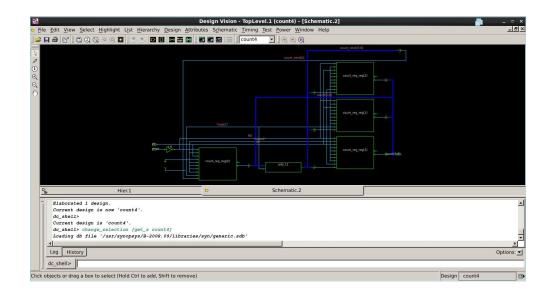
To do this step choose **File -> Elaborate** from the main menu and from the elaboration menu select the top level file of your design and press **OK** as seen in the following figure.

	Elat	oorate Desig	×
<u>L</u> ibrary:	DEFAULT		•
<u>D</u> esign:	count4(ve	rilog)	-
Parameters:	Name	Value	
<u> <u> </u> <u> R</u>eanalyze </u>	out-of-date		
		ОК	Cancel

After pressing OK in the elaboration menu, the main menu will be updated with your design which is count4 in this tutorial.

24	and the second s	Design Vision - TopLevel.1 (count4) - [Hier.1]	📑 💷 ×
& File Edit View	Select Highlight List Hierarchy	Design Attributes Schematic Timing Test Power Window Help	X
S	@ @ @ @ @ * * ▷ .	🗖 🗃 🖬 📓 🔚 🔤 count4 💽 💽 🕘 📵	
Logical Hiera	Cells (All)		×
G==> co	unt4 Cell Name Ref Name	Cell Path Dont Touch	
0	Dcount_reg **SEQGEN*	count_reg_r undefined	
0	D-count_reg **SEQGEN*	count_reg_r undefined	
Q	D-count_reg **SEQGEN*	count_reg_r undefined	
Eth		count_reg_r undefined	
	Dadd_11 *ADD_UNS_		
	D-U1 **logic_1**	U1 undefined	
	D-U2 **logic_0**	U2 undefined	
	DI_0 GTECH_NOT	I_0 undefined	
Elabora	ted 1 design.		-
Current	design is now 'count4'.		_
dc_shel			
	design is 'count4'. 1> change_selection [get_s count	41	
	db file '/usr/synopsys/B-2008.1		-
4			1 1
Log His	tory		Options: -
dc_shell>			
dc_snell>	J		
Ready			Design count4

You can display the elaborated schematic by selecting your design in the hierarchy window and then clicking the **Create Design Schematic** icon **D** or by right click on your design entity then choose **Schematic View**. The following is the elaborated schematic view for our design.



It is a good idea to save your design to this point. Choose **File -> Save as** and from the following window choose a name and click **Save**.

	Save Design As 📑	×
Look in:	🔄 /home/eslam/Desktop/count4_new/ 💽 🗢 🖻) 💣 🔝 🏢
🗀]) count4	.v	
File <u>n</u> ame:	count4_elaborated.ddc	<u>S</u> ave
File type:	Database Files (*.ddc *.ddc.gz *.db *.db.gz *.gdb 💌	Cancel
<u>F</u> ormat:	Auto	SYNOPSYS'
☑ <u>S</u> ave	all designs in hierarchy	

You can type this command in Design Vision to generate a report of the hierarchy of the design, *report_hierarchy*. You can also use this command to report all used cells and operating voltage of your design, *report_design*.

Now we are going to define our design constraints like area and clock speed. For this step select your design in the hierarchy window then click the **Create Symbol View** icon **U**. You will get a window like this one.

2	Design Vision - Topl	.evel.1 (count4) - [Symbol.1]		🗐 💶 🖛 🗙
File Edit View Select Highlight List Hierarch				X
] 🗳 🖬 😂 📴] 0. 0. 0. 0. 0. 🗐] * * 🔟		•] • • •		
A.	Operating Environment Optimization Constraints			
3	Optimization Directives			
⊙ eí cí				
	clk	count4		
		Count4	count[3:0]	
	rocot			
	reset			
B Hier.1	0	Symbol.1		
dc_shell> Current design is 'count4'.				<u> </u>
dc_shell> change_selection [get_s co				
<pre>dc_shell> change_selection [get_s co dc_shell> ::cmenus::create_schematic</pre>				
Loading db file '/usr/synopsys/B-20	08.09/libraries/syn/generic.sdb'			
dc_shell> change_selection [get_s c.	1k]			
Log History				Options: 💌
dc_shell>				
Ready				Port clk

Now select your clock signal from the symbol view and choose **Attributes -> Specify Clock** to define the clock period and its duty cycle. For our example we choose a clock period of 10 ns and duty cycle of 50% like the following figure.

ort name:		
Perio <u>d</u> : 10]
Edge	Value	Add <u>e</u> dge pair
Rising Falling		0 5 Rem <u>o</u> ve edge pair
		Invert <u>w</u> ave form

Next step is to define the constraints on design area by choosing **Attributes -> Optimization Constraints -> Design Constraints** from Design Vision main window. We choose area to be equal to zero to get the minimum area for our design.

2	Design Vision - To	pLevel.1 (count4) - [Symbol.1]			_ • ×
File Edit View Select Highlight List Hierarch	hy Design Attributes Schematic Timin	ng Test Power Window Help			X
] 😂 🖬 😂] 🔍 Q Q 🔍 Q 🖾] ≛ ≛ 🔟	Specify Clock	·			
	Operating Environment	•			
1	Optimization Constraints	Design Constraints			
(1)	Optimization Directives	<u>Timing Constraints</u>			
••• • •		Derive Constraints			
	clk				
		count4			
			count[3:0]		
\rightarrow	reset				
Pas Hier.1	0	Symbol.1			
dc_shell> change_selection [get_s co	ouet 41				
dc_shell> ::cmenus::create_schematic					-
Loading db file '/usr/synopsys/B-20 dc_shell> change_selection [get_s c.					
dc_shell> create_clock -name "clk"		lk }			
1					
dc_shell>					<u> </u>
Log History					Options: 💌
dc_shell>					
Ready				Port clk	D

🗖 🛛 Design (Constants
Current design: count	4
Optimization constra	ints
	Constraint value: Unit:
Max ar <u>e</u> a :	0
Max <u>dynamic</u> power:	
Max <u>l</u> eakage power:	
Max total power:	
– Design rules –	
Max fanout:	
Max <u>t</u> ransition:	
ОК	Cancel <u>A</u> pply

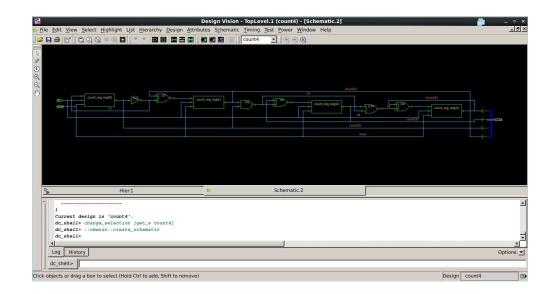
Next step is to compile your design. After this step your design will be mapped to real standard cells from your target technology library. From the main window of Design Vision, choose **Design -> Compile Design**, you will get a menu like the following figure, click **OK**.

	Compile		>
 Mapping options Map design Exact map Map effort: medium Area effort: medium Design rule options Fix design rules and optimiz Optimize mapping only Fix design rules only Fix hold time only 		tions Incremental Allo <u>w</u> bound Auto <u>u</u> ngrou & Area	lary condition
	ОК	Cancel	Apply

As seen in the following figure, your design is now mapped to real standard cells.

ile Edit View Select Hi	oblight List Hie	rarchy Design		<pre>gn Vision - TopLevel.1 (count4) - [Hier.1] ematic Timing Test Power Window Help</pre>	
Logical Hierarchy	Cells (All)				
==> count4	Cell Name	Ref Name	Cell Path	Dont Touch	
	Dcount_reg	DFQRM2RA	count_reg_r	undefined	
	Dcount_reg	DFQRM2RA	count_reg_r	undefined	
	D count_reg	DFQRM2RA	count_reg_r	undefined	
	⊕count_reg	DFQRM2RA	count_reg_r	undefined	
	1D-U3	XOR2M2RA	U3	undefined	
	1D-U5	XNR2M2RA	U5	undefined	
	1D-U7	XNR2M2RA	U7	undefined	
	1D-U9	ND2M2R	U9	undefined	
	DU10	NR2B1M2R	U10	undefined	
	DU11	INVM2R	U11	undefined	
Optimization Con 1 Current design is dc_shell> change_s	'count4'.	s count4]			
Log History					Option
dc_shell>					

You can check the schematic view of your design now to see it in terms of standard cells.



A good idea is to save your design now using File -> Save As and choose an appropriate name, for use we name it count4_clk10ns_mapped.ddc.

To report your design constraints to check if there are any violations on your design constraints, choose **Design -> Report Constraints**. Continue like the following figure.

Repo	rt Constraints 📑 🛛 👘					
Report for						
Current design: count4						
Report options						
🖵 No <u>l</u> ine split	└ Verb <u>o</u> se					
Show only maximum capacitance	Show only minimum <u>c</u> apacitance					
F Show only maximum delay	F Show only minimum delay					
Show only maximum leakage power	F Show only minimum porosity					
Show only maximum dynamic power Show only maximum transition						
F Show only maximum fano <u>u</u> t	F Sho <u>w</u> only maximum area					
Show only cell degradation	Show only connection class					
Show only multiport nets	Show only critical range					
Show all violators	Significant digits: 2					
Output options						
☑ To report <u>v</u> iewer						
To file: count4_clk10ns_mapped_allvie	ol.rpt <u>B</u> rowse					
Append to file						
	OK Cancel Apply					

You will get a report like the following.

2				Vision - TopLevel.1 (co				💼 – • ×
				s Schematic Timing Tes		w Help		-82
		• • • • •		count4	•] • • •			
X	28				#6			
() Rep () Des () Vez () Dat	port : constraint -all_violat ign : count4 rzion: B-2000.09 to : Thu Jin -2000.09 to : Thu Jin -2000.09 to : Thu Jin -2000.09 max_area Design 	0:59 10:48:40 2014 Required Area 0.00		Slack -47.16 (VIOLATED)	(
200	*** End Of Report	Hier.1	D	Schematic.2		Report.1		
-	Design	Area	Area	Slack				2
	count4	0.00		-47.16 (VIOLA	TED)			
	dc_shell>							
-	Log History							Options: •
	dc_shell>							
Ready							Design	count4

This report says that, area is violated and this is expected because we set the max area earlier to zero to achieve a minimum area, now this minimum area can be seen under the field of **Actual Area** in this report and it is 47.16 um^2 in our case. **Slack** equals to **Required Area** minus the **Actual Area** achieved by Design Vision, one would like always to get a positive slack. Positive slack means that, constraint is met but negative slack means constraint is violated.

Choose **Design -> Report Area** to check the overall area of your design.

	Report Area 📑	×
-Report for		
Current design: Current instance:	count4	
-Report options — V No <u>l</u> ine split		
-Output options — To report <u>v</u> iewe	ır	
	4_clk10ns_mapped_area.rpi <u>B</u> row	se
	OK Cancel <u>A</u> pp	oly

After clicking **OK** you will get a report for your design area like the following figure.

			Design Vis	ion - TopLevel	.1 (count4) - [Repor	t.2 - Area]			
	ile <u>E</u> dit <u>V</u> iew <u>S</u> elect <u>H</u> ighligh	nt L <u>i</u> st <u>H</u> ierarchy	Design Attributes Sche	ematic <u>T</u> iming	Test Power Window	Help			_ & ×
] 🛋	H & K] Q Q Q Q	0 1 * * D I		count4	• • • • •				
	XIII				糸				
1	Date : Thu Jul 10 20:02:								-
00	Library(s) Used:								
23	uk651scl1mvbbr_120c25_	tc (File: /mnt)	hgfs/UMC65/G-9LT-LOGI	C_MIXED_MODE6:	5N-LL_LOW_K_UMK65LS	CLLMVBBR-LIBRAF	Y_TAPE_OUT_KIT-Ver.E	02_PB/UMK65LSCLLMVBBR_B02_1	APEOUTKI
	Number of ports:	6							
	Number of nets:	12							
	Number of cells:	10							
	Number of references:	6							
	Combinational area:	15.480000							
	Noncombinational area:	31.680000	1223						
	Net Interconnect area:	undefined (F	lire load has zero net	area)					
	Total cell area:	47.160000							
	Total area:	undefined							-
	4							1	<u>ار</u>
	Pg Hier.1	0	Schematic.2		Report.1		Report.2		
	Noncombinational area:	31.680000							
	Net Interconnect area:	undefined	(Wire load has zero	net area)					
	Total cell area:	47.160000							
	Total area:	undefined							
	dc_shell>								-
									÷
	Log History								Options: 💌
	dc_shell>								
	1u 								1-
Rea	dy							Design count4	

To check timing of your design, choose Timing -> Report Timing Path.

Repo	rt Timing Paths 👘 🗧
From: pin Through: pin To: pin	Selection[1] Selection[2] Selection[3]
Report options	Ma <u>x</u> imum path delay:
Max paths per group: 1	Minimum pat <u>h</u> delay:
Delay type: max Sort by: Filter Fil	eport timing loops ustify paths with input vector ind true path tath delay threshold: 0 Enable asynchronous arcs Show net trans <u>i</u> tion time Show net capacitance
i snow dont_touch, size_only attributes for her	
Output options	
To file: count4_clk10ns_mapped_timing.rpt	Browse
Append to file	
	OK Cancel Apply

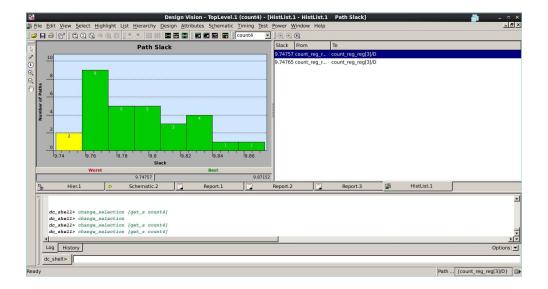
You will get a timing report like the following one; from this figure we notice that our timing slack is positive and equals to 9.75 ns i.e. timing constraint is met and we can reduce the required clock period by 9.75 ns. Now we know that min clock period for our design is, 10 ns - 9.75 ns = 0.25 ns.

88100000011≛≛			count4	• • • •			
				96			
count_reg_reg[1]/Q (DFQRM2RA)	0.11	0.11 r					
U9/Z (ND2M2R)	0.03	0.14 f					
U10/Z (NR2B1M2R)	0.04	0.17 r					
U3/Z (XOR2M2RA)	0.08	0.25 f					
count_reg_reg[3]/D (DFQRM2RA)	0.00	0.25 f					
data arrival time		0.25					
clock clk (rise edge)	10.00	10.00					
clock network delay (ideal)	0.00	10.00					
count_reg_reg[3]/CK (DFQRM2RA)	0.00	10.00 r					
library setup time	0.00	10.00					
data required time		10.00					
data required time	ant na ini dal ani ak da ani ali ini da ani ak ani ani da	10.00					
data arrival time		-0.25					
slack (MET)		9.75					
B Hier.1 D	Schematic.2		Report.1		Report.2	Report.3	
data arrival time		-0.25					
slack (MET)		9.75	-				
dc_shell>							
						 	0-1
Log History							Opti

A good way to visualize your timing paths and determine the critical path can be done using **Timing -> Path Slack.** Choose OK from the following menu.

🗖 🛛 Path Slack 🚰 🛛 🛛 🗧
From: pin Selection[1] Through: pin Selection[2] To: pin Selection[3]
Nworst paths: 10 * Max paths: 50 * Group name: • Delay type: max • T Enable preset clear arcs Include hierarchical pins
 Number of bins: 8 ★ ✓ Value range per bin: <= Slack <= ✓ Lower bound strict ✓ Upper bound strict
Histogram title: Path Slack X-axis title: Slack Y-axis title: Number of Paths
OK Cancel Apply

You will get a histogram for slacks of all timing paths in your design like the following one.



In this histogram, the path with the smallest slack is the worst (critical) path in your design. In our example, the worst paths have a slack of 9.74 ns. When selecting any path from this histogram you can see the equivalent path on your circuit by moving to your schematic view. The selected path above is shown in schematic as follows.

Eile Edit View Select Highlight List Hierarchy Des ■ ■ ● □ □ □ □ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		er <u>W</u> indow Help		• :
			(1994)	
			AT THE	
P_B Hier.1 © Schematic.2	Report.1 Rep	ort.2 Report.3	📸 HistList.1	J
<pre>dc_shell> change_selection [get_s count4] dc_shell> change_selection dc_shell> change_selection [get_s count4] dc_shell> change_selection [get_s count4] </pre>				•

Other reports can be useful for you like resources report, you can get it from

D	esign -	> Re	port]	Design	Resources	as	follov	wing.
---	---------	------	--------	--------	-----------	----	--------	-------

Re	port Design Reso 🔤 es 🛛 🗙 🗙
-Report for	
Current design: Current instance:	count4
-Report options No <u>l</u> ine split -Output options —	Show resources for subdesigns
To report viewe	۲ ۱
☑ To file: unt4_c	lk10ns_mapped_resources. Browse
🔽 App	e <u>n</u> d to file
	OK Cancel <u>A</u> pply

In the following steps, we will export the netlist file and all needed files to be used in the following design stages like automatic place and route using SOC Encounter.

Before exporting the netlist file you have to type the following command in Design Compiler command line.

change_names -rules verilog -hierarchy -verbose

This command is to apply some verilog naming rules to your design before exporting the gate level netlist file.

Rie Edit View Select Highlight List Hierarchy De	Design Vision - TopL				urces]			×
			M	-				
<pre></pre>	/ add_11	1						4
No implementations to report No multiplexors to report								
Design Type Object	New Name							
count4 cell count_reg_reg[0] count4 cell count_reg_reg[1]	<pre>count_reg_reg_0_ count_reg_reg_1_ count_reg_reg_2_ count_reg_reg_3_ ages'. (CMD-041) 1_depth'. (CMD-041)</pre>							
***** End Of Report *****								
Re Hier.1 D Schematic.2	Report.1	Report.2		Report.3	100	HistList.1	Report.4	
count4 cell count_reg_reg[3] Information: Defining new variable 'var_ Information: Defining new variable 'max_ Information: Defining new variable 'coll 1	model_depth'. (CMD-041)	(CMD-041)						<u>.</u>
dc_shell>			-					
Log History							 	Options: 💌
dc_shell> change_names -rules verilog -hid	erarchy -verbose	<u> </u>						
Ready							Path	{count_reg_reg[3]/D}

Now, save your netlist verilog file using **File -> Save As** then type an appropriate name and change the format to verilog like the following figure.

Save Design As 👘	×
Look in: 🔄 /home/eslam/Desktop/count4_new/) 💣 🔠 🇰
count4_elaborated.ddc	
File name: count4_clk10ns_mapped.v	<u>S</u> ave
File type: Database Files (*.ddc *.ddc.gz *.db *.db.gz *.gdb	Cancel
Eormat: VERILOG (v)	SYNOPSYS'
Save all designs in hierarchy	

The output gate level netlist file from this step is as follows.

```
module count4 ( clk, reset, count );
output [3:0] count;
input clk, reset;
wire n1, n2, n3;
wire [3:0] count_next;
DFQRM2RA count_reg_reg_0_ (.D(n1), .CK(clk), .RB(reset), .Q(count[0]));
DFQRM2RA count_reg_reg_1_ (.D(count_next[1]), .CK(clk), .RB(reset), .Q(
    \operatorname{count}[1]);
DFQRM2RA count_reg_reg_2_ (.D(count_next[2]), .CK(clk), .RB(reset), .Q(
    count[2]);
DFQRM2RA count_reg_reg_3_ (.D(count_next[3]), .CK(clk), .RB(reset), .Q(
    count[3]);
XOR2M2RA U3 ( .A(count[3]), .B(n2), .Z(count_next[3]) );
XNR2M2RA U5 ( .A(count[2]), .B(n3), .Z(count_next[2]) );
XNR2M2RA U7 (.A(count[1]), .B(n1), .Z(count_next[1]));
ND2M2R U9 (.A(count[1]), .B(count[0]), .Z(n3));
NR2B1M2R U10 (.NA(count[2]), .B(n3), .Z(n2));
INVM2R U11 ( .A(count[0]), .Z(n1) );
endmodule
```

This file will be used in appendix C and is also used as an input to the automatic place and route tool, SOC Encounter to generate the layout for your design.

Next step is the post synthesis timing data extraction. In this step we extract the standard delay format file which can be used in post synthesis simulation. To get this file, type the following command in the command line of Design Vision.

write_sdf -version 2.1 coun4_clk10ns_mapped_vlog.sdf

You can fine this file and all report files you've generated earlier in your working directory.

The following command is to generate a file includes all your design constraints in TCL format and will be used in standard cell placement and routing (SOC Encounter).

write_sdc -nosplit count4_clk110ns_mapped.sdc

For more details, please check the following video:

http://youtu.be/pD85Hnsi2cc

Part 2:

The same steps of part 1 can be done by using this TCL script. The code is self documented, please read it carefully and edit all needed fields according to your design.

Design Setup (change library files according to your technology)
set link_library /home/eslam/Desktop/synopsys/uk65lscllmvbbr_120c25_tc.db
set target_library /home/eslam/Desktop/synopsys/uk65lscllmvbbr_120c25_tc.db
#Analyze
analyze -format verilog {/home/eslam/Desktop/synopsys/count4.v}
#elaborate (count4 is the name of the top level module)
elaborate count4 -architecture verilog -library DEFAULT -update
write -hierarchy -format ddc -output /home/eslam/Desktop/synopsys/count4.ddc
#timing & area constraints (clk is the clock name in my verilog file-edit according to
your design (ns))
create_clock -name "clk" -period 10 -waveform { 0 5 } { clk }
set_max_area 0
#compile design
compile
#export design (reports and netlist and timing files)
write -hierarchy -format ddc -output
/home/eslam/Desktop/synopsys/count4_clk10ns_mapped.ddc
#generate design reports
report_constraint -nosplit -all_violators > /home/eslam/Desktop/synopsys/allviol.rpt
report_area > /home/eslam/Desktop/synopsys/area.rpt
report_timing > /home/eslam/Desktop/synopsys/timing.rpt
report_resources -nosplit -hierarchy > /home/eslam/Desktop/synopsys/resources.rpt
report_reference -nosplit -hierarchy > /home/eslam/Desktop/synopsys/references.rpt
report_hierarchy > hierarchy.rpt
report_design > design.rpt
#add some verilog naming rules before exporting the gate level netlist file
change_names -rules verilog -hierarchy -verbose
write -hierarchy -format verilog -output
/home/eslam/Desktop/synopsys/count4_clk10ns_mapped.v
write_sdf -version 2.1 count4_mapped_vlog.sdf
write_sdc -nosplit count4_vlog.sdc
puts "Finished"

For more details, please check the following video: http://youtu.be/gH6ZMh3IQBI

Conclusion

Using this tutorial we've learned how to convert our functional verilog code into a gate level netlist file, i.e. a file containing standard cells from our technology library and learned how to define design constraints like timing and area constraints and how to analyze timing paths using a timing histogram and finally how to generate all needed reports and files which can be used in next design steps like automatic place and route.

Appendix C

IMPORTING SYNTHESIZED DESIGN INTO CADENCE COMPOSER SCHEMATIC VIEW

In this tutorial we are going to simulate a digital 4-bit Verilog counter to test its function first without any timing analysis then we will use the synthesized version (output of Design Compiler) of this counter and import it into CADENCE Composer as a schematic view containing all standard cells needed for this counter from our technology library, so in this part we will simulate the 4-bit counter in two levels of design:

- 1. Pre-synthesis functional Verilog simulation using AMS simulator
- 2. Post-synthesis transistor level simulation using Spectre simulator

1. Pre-synthesis functional Verilog simulation using AMS simulator

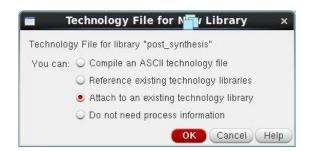
The following is the digital 4-bit Verilog code used for this tutorial:

module count4(input clk,reset,output[3:0] count);
reg[3:0] count_reg;
wire[3:0] count_next;
//state register
always@(posedge clk,negedge reset)
if(~reset)
count_reg <= 0;
else
count_reg <= count_next;
//next state logic
assign count_next = count_reg+1;
//output logic
assign count = count_reg;
endmodule</pre>

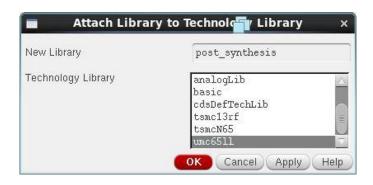
For this tutorial, we will make new work library and name it post_synthesis as following:

Library Manager: WorkArea: /usr/local/IC614/TSMC_65nm	📅 _ 🗆 ×
Elle Edit View Design Manager Help	cādence
New Image: Library Open Ctrl+O Cell View Open (Bead-Only) Ctrl+R Category Copen Witty Cell	
Load Defaults	
Ogen Shell Window Ctrl+P	
Exit Ctrl+X	
ams_mapped ams_tutoral ams_tutoral basic cdsDeffechLib cds_assetions cds_nhconn cds_spicelib connectLib functional leee ncintemal ncmortels	
Messages LIB connectLib from the same file (defined earlier.) Warning: LIB connectLib from File /usr/local/C6114/TSMC_65nm/cds.lib Line 18 redefines LIB connectLib from File /usr/local/C614/tosls.lib/cds.lib Insert UNDEFINE connectLib before DEFINE connectLib in /usr/local/C61470kC_65nm/cds.lib Or remove or comment out DEFINE connectLib in /usr/local/C61470kC_65nm/cds.lib to suppress this warning message.	
	,

New	Library 🛅	
Library		
Name post_synthesis 🧹		
Directory a /usr/local/IC614/TSM	1C_65nm/ 🧧 🦕	E 💣 🎟 🎟
Assura CCI Calibre NangatePfdLoopF Play Techfile UMK65FDKLLC000000A_B10 adpll_mixed adpll_mixed	ams_mapped ams_tutorial drc_65n inv_hdl_tb.run1 inv_hdl_tb_run1 inv_hdl_tb_run1 inv_hdl_tb_run2 Vs_65nm models nand2.run1	nand2_run1 nand2_test.ru nand2_test.ru notgate_tb_ru notgate_tb_ru notgate_tb_ru notgate_tb_vu fd_loopF_vi skill test stsmc13rf
File type: Directories		
Design Manager		
Use NONE Use No DM		
OK Apply	Cancel	Help



We should attach this work library to our technology library which is **umc65ll** in our tutorial.



We make similar steps to those done in appendix A to import this Verilog code into cadence as following:

	New	kArea: /usr/local/IC614/TSMC_	_65nm
jle <u>E</u> dit <u>V</u> iew	<u>O</u> pen		cādeno
_ Show Catego Library	import Export Refresh Make <u>R</u> ead Only	EDIF200 Verilog V <u>H</u> DL	View
post_synthesis	Bookmarks	- <u>S</u> pice DEF	
ams_tutorial analogLib basic ddsDeffechLib cds_assertions cds_inhconn dds_spicelib connectLib functional ieee ncinternal ncmodels nccutis post_synthesis dtExamples fLib	1 ams_mapped pfd_loopF schematic 2 adpll_mixed2 allDesign config 3 adpll_mixed2 allDesign schematic 4 adpl dco2_tb schematic 5 ams_mapped loopFilter_0 schematic 7 ams_mapped pfd schematic 8 adpl_mixed all_vlog schematic 9 adpll ring_osc_tb schematic 0 adpll ring osc_nohdc schematic	LEF Stream Netlist View	
sdilib std 🍙	<u>C</u> lose Data		
	Exit	6.1.4 - Log: /root/CDS.log	
Messages -	Eile Tools Options Sonnet Help		cādence
IB connectLit	Loading umc6511/libInitCus done! .oaded umc6511/libInit.il successfully INFO (TECH-180011): Design library 'po 	7	xt 'oxf_cb' from library 'umc6 hed to technology library 'umc R

	Verilog In 📑	
Import Options Glob	al Net Options Schematic Generation Opt	ions
File Filter Name		
/ count4.v		
count4.v~		
count4_mapped.v count4_mapped.v~		
/home/eslam/Desktop/	count4/	
Target Library Name	post_synthesis	Browse
Reference Libraries	basic	
Verilog Files To Import	'home/eslam/Desktop/count4/count4.v	Add
-f Options		Add
-v Options		Add
-y Options		Add
Library Extension		
Library Pre-Compilation		-
Pre Compiled Verilog Lik		
HDL View Name	hdl	
Target Compile Library N	lame	Browse
Compile Verilog Library	Only	
<u></u>		Add
Ignore Modules File		
Import Modules File		Add
Import Structural Modules	As schematic	
-Structural View Names		
	Cancel Defaults Apply Load	(Save)(H

After ending this process, you will get a functional and symbol view for your counter.

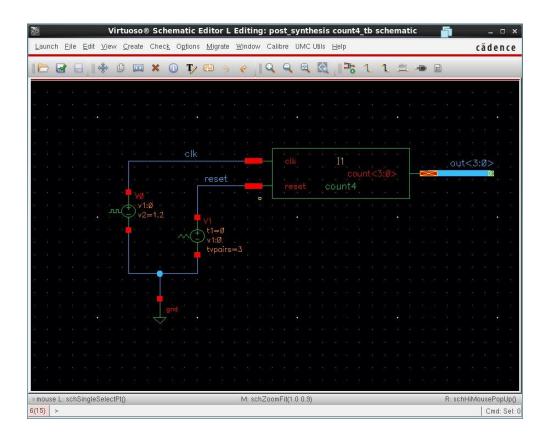
📓 Library Manager: WorkArea: /usr/local/IC614/TSMC_65nm 📑 _ 🗆 >			
<u>File Edit View D</u> esign Manager <u>H</u> elp			cādence
Show Categories 🗌 Show Files			
Library	Cell	View	
post_synthesis	count4] []	
ams_tutorial analogLib basic cdsDefTechLib cds_assertions cds_inhconn cds_spicelib connectLib functional ieee ncinternal ncmodels ncutils post_svinhesis rfLxe_ncutils rfLib sdilib	count4	functional symbol	
Messages			
LIB connectLib from the same file (defined earlie Warning: LIB connectLib from File /usr/local/C614/tools.lr Insert UNDEFINE connectLib before DEFINE connectLib in /usr/local/C614/TSMC_65nm/cds.llb Or remove or comment out DEFINE connectLib in /usr/local/IC614/tools.lnx86/affirma_ams/ete	14/TSMC_65nm/cds.lib Line 18 rede xx86/affirma_ams/etc/connect_lib/cd	rfines s.lib	

Double click on functional view to check your code.

] Show Categori	<mark> ≫ verilog.v (/usr/local/IC614/TSMC_65n post_sy _ □ × </mark> <u>File Edit View Search Tools D</u> ocuments <u>H</u> elp		
ibrary	🍳 🔄 Open 🗸 🕭 Save 🛛 🚔 🔄 Undo 🛷 🔶 🗸	View	
ost_synthesis		functional	
Ims_tutorial inalogLib isaic dsDefTechLib ids_assettions :ds_inhconn dds_spicelib onnectLib unctional eee icrinternal icrinderla icrinternal icrintern	<pre>verilog.v % // Created by ihdl 2 module count4(input clk,reset,output[3:0] count); 3 reg[3:0] count_reg; 4 wire[3:0] count_next; 5 //state register 6 always@(posedge clk,negedge reset) 7 if(~reset) 8 count_reg <= 0; 9 else 10 count_reg <= count_next; 11 //next state logic 12 assign count = count_reg+1; 13 //output logic 14 assign count = count_reg; 15 endmodule 16</pre>	functional symbol	
dessages	Verilog ~ Tab Width: 8 ~ Ln 1. Col 1 INS		
/arning: LIB conr IB connectLib fro sert UNDEFINE efore DEFINE co in /usr/local/ICI)r remove or com			

Now, it's time to make a test bench for this code to generate simulations results for it.

	New 🚰 🛛 🛛 🗙
File	
Library	post_synthesis
Cell	count4_tb
View	schematic
Туре	schematic 🔽
Application	
Open with	Schematics L
🔲 Always use	e this application for this type of file
Library path fil	le
1 13 3 4	IC614/TSMC 65nm/cds.lib



For this code we use an active low reset signal, this is how to get this reset signal using library sources:

	Ac	ld Instance 🛅	
Library	analogLib		Browse
Cell	vpwl		
View	symbol		
Names			
Array	Rows	1 Colur	nns 1
	🕹 Rotate 📄 🚺	🖌 Sideways 🔵 🚄	Upside Down
Frequenci	/ name for 1/period		
	pairs of points	3	
Time 1	hand at hourse	0 s	
Voltage 1		0	
Time 2		1n s	_
Voltage 2		0	
Time 3		1.1n s	
Voltage 3		1.2 ¥	
Noise file r	name		
	name noise/freq pairs	0	
	noise/freq pairs		
Number of	noise/freq pairs e		
Number of DC voltage	noise/freq pairs e		
Number of DC voltage AC magnit	noise/freq pairs e ude		

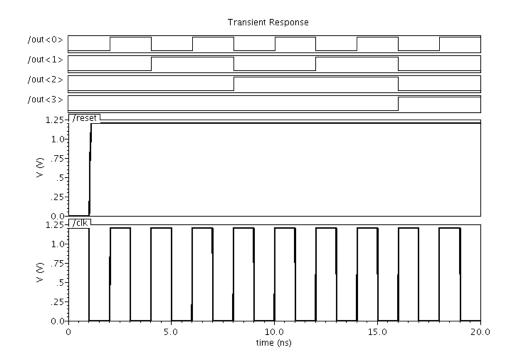
Use similar steps to that in appendix A to create new config view for your test bench and don't forget to change the simulator to AMS and edit the connection rules.

	New 👘 🛛 🗙
File	
Library	post_synthesis
Cell	count4_tb
View	config
Туре	config 🔽
Application	
Open with	Hierarchy Editor 🧧
🔲 Always use	e this application for this type of file
Library path fi	le
	IC614/TSMC 65nm/cds.lib

Simulator	ams	·
Project Directory Host Mode	ADSsim aps hspiceD spectre	C distributed
Host	UltraSim 	
Remote Directory	ams spectreVerilog UltraSimVerilog	

You must start the simulation from this config view as we noticed in appendix A.

The simulation results for this Verilog 4-bit counter are in the following figure:



2. Post-synthesis transistor level simulation using Spectre simulator

Now, we are going to import our synthesized Verilog code (standard cells description of our counter – output of Design Compiler) into CADENCE composer and simulate it in the transistor level using Spectre simulator. We use similar steps as we did to import the functional Verilog code but the only different part in import menu is that, you must specify your standard cells library (for us, it is **umc65stdcells)** as shown in the following figure :

	Verilog In 🔤	
Import Options Glo	bal Net Options Schematic Generation C	ptions
File Filter Name		
/		
count4.v count4.v~		
count4_mapped.v count4_mapped.v~		
1 <u>2</u>		
/home/eslam/Desktop	/count4	
Target Library Name	post_synthesis	Browse
Reference Libraries	basic umc65stdcells	
Verilog Files To Import	alam/Desktop/count4/count4_mapped.⊽	Add
-f Options		Add
-v Options		Add
anne teoreanna		
-y Options		Add
Library Extension		
Library Pre-Compilatio	n Options	
Pre Compiled Verilog Lil	orary	
HDL View Name	hdl	
Target Compile Library N	Jame	Browse
Compile Verilog Library		Biomse
<u></u>		
Ignore Modules File		Add
Import Modules File		Add
Import Structural Modules	As schematic	
Structural View Name	S	
	Cancel) Defaults) Apply) Load	1)(Save)(He

Transistors used in each standard cell in **umc65stdcells** are taken from the technology library we've included earlier, the **umc65ll**. You can have a look on the mapped code of this counter in the following figure:

```
module count4 ( clk, reset, count );
output [3:0] count;
input clk, reset;
wire n1, n2, n3;
wire [3:0] count_next;
DFQRM2RA count_reg_reg_0_(.D(n1), .CK(clk), .RB(reset), .Q(count[0]));
DFQRM2RA count_reg_reg_1_(.D(count_next[1]), .CK(clk), .RB(reset), .Q(
     count[1]) );
DFQRM2RA count_reg_reg_2_( .D(count_next[2]), .CK(clk), .RB(reset), .Q(
     count[2]) );
DFQRM2RA count_reg_reg_3_ ( .D(count_next[3]), .CK(clk), .RB(reset), .Q(
     count[3]));
XOR2M2RA U3 ( .A(count[3]), .B(n2), .Z(count_next[3]) );
XNR2M2RA U5 ( .A(count[2]), .B(n3), .Z(count_next[2]) );
XNR2M2RA U7 ( .A(count[1]), .B(n1), .Z(count_next[1]) );
ND2M2R U9 ( .À(count[1]), .B(count[0]), .Z(n3) );
NR2B1M2R U10 ( .NA(count[2]), .B(n3), .Z(n2) );
INVM2R U11 ( .A(count[0]), .Z(n1) );
endmodule
```

After ending this process successfully, there will be a schematic view added to

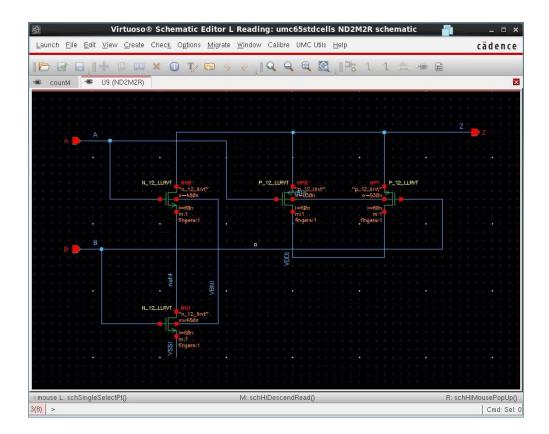
your cell views of the counter.

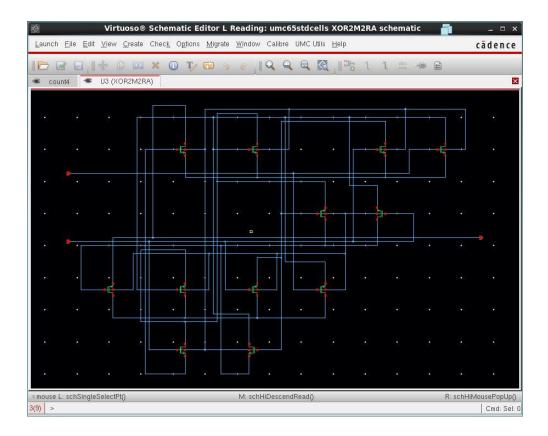
ile <u>E</u> dit <u>V</u> iew <u>D</u> esign Manage	r <u>H</u> elp		cāden
Show Categories 🗌 Show	w Files	, View	
iost_synthesis	count4	schematic	
unctional	count4	functional	
866	County	schematic	
ncinternal		symbol	
ncmodels		10 N	
ocutils			
iost_synthesis fExamples			
Examples Lib			
dilib			
td			
ynopsys			
smc13rf smcN65			
ancinos atorial			
mc65II			
mc65stdcells			
erilog			
rital_memory			
lessages			
and the second			
	/usr/local/IC614/TSMC_65nm/cds.lib Line 18 rec /IC614/tools.lnx86/affirma_ams/etc/connect_lib/c		
isert UNDEFINE connectLib	neeranteerannoorainnia_anoretereenneer_nore	20.110	
efore DEFINE connectLib			
in /usr/local/IC614/TSMC_65nm	i/cds.lib		
r remove or comment out DEFINE	E connectLib ffirma.ams/etc/connect.lib/cds.lib		
suppress this warning message.			
11			

Double click on this schematic view to check the internal structure of your counter in terms of your standard cells.

-	-			 	~			-			~	23								 -
0			÷		0	T/		¢		1 9				EO 1	1	abc				
				-					ПĒ		÷.,			7		-1-				
								•						1	•					
								•		·			÷	·	÷	·	·		·	
				 •			a		1											
									T.											
		·												_∐_				·		
									-											
									•		-₽ ¥									
									-											
		<u>.</u>	×.	 					.											
											_	1								
									÷		÷÷									
									ŀ		·									

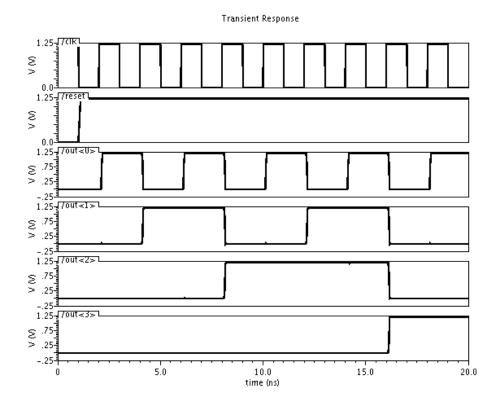
You can also check the internal structure of each standard cell in your counter by pressing **shift+E** on this cell. The following are the internal structures of two standard cells used in this counter:





Now, it is time to test this schematic. An important notice here is that; don't forget to connect power nodes (VDD and GND) and bulk nodes of transistors (VBN and VBP). We did not make this step when simulating the functional Verilog code because there were no transistors in this schematic and we were just simulating the code.

The simulation results for the transistor level schematic of the counter are in the following figure. One can easily notice, the same output as the functional Verilog code is achieved.



For more details please check this video: http://youtu.be/FLjRAzKSvxc

3. Conclusion

In this tutorial, we have simulated a digital Verilog 4-bit counter in two different design levels, one is to simulate the pre-synthesis Verilog code to test its functionality only and another one is to simulate the post-synthesis code after being mapped to real standard cells from your technology library. Both simulations give the same results.

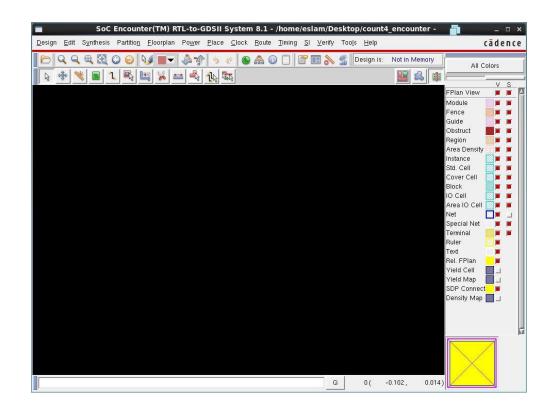
Appendix D

STANDARD CELL PLACEMENT AND ROUTING

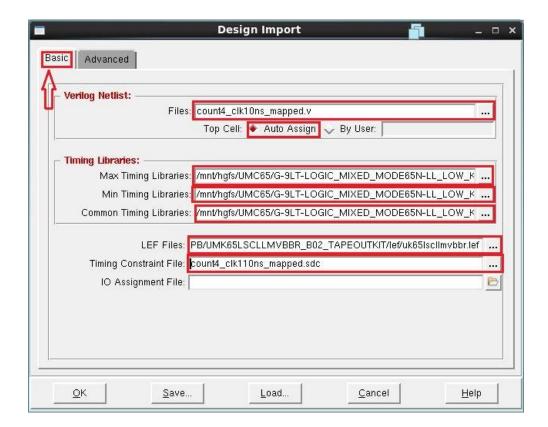
In this tutorial we will learn how to import the gate level netlist file (which we get from SYNOPSYS Design Vision) into CADENCE SOC Encounter to do the placement and routing of the standard cells used in our example, the 4-bit counter. You may import the final layout from this step into CADENCE layout editor to check it through DRC, LVS and PEX and make the post layout simulation. The tutorial is divided into two parts, part 1 and part 2. For part 1 we make all the steps using the graphical environment and for part 2, we make the same steps using a TCL (Tool Command Language) script.

Part 1:

The starting window for SOC Encounter looks like the following figure.



Choose **Design -> Import Design**, the following window will show up. Fill it with your gate level netlist in the field of **Verilog Netlist** then choose your **Timing Libraries** as follows, for **Max Timing Libraries** choose the file with worst case conditions and for **Min Timing Libraries**, choose the file with best case conditions and finally **for Common Timing Libraries**, choose the file with typical case conditions. For **LEF files** you need to add the LEF files from your technology kit. For **Timing Constraints File**, add the SDC file which we got from Design Vision in logic synthesis step.



Now move to the **Advanced tab** and choose **Power** to add names for your power nets, make sure to type proper names like in you LEF file. To know the power net names in your technology kit you can check the LEF file and search for "power" to know the name for power net and ground net. For our technology library, net names are VDD and VSS. You may need to save this configuration to use it again instead of inserting all fields from the beginning, to do this step choose **Save** and the **OK**.

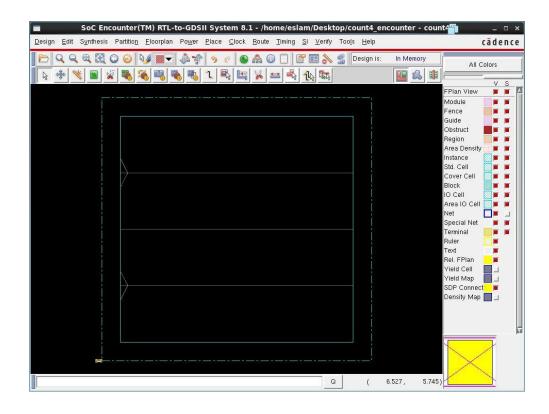
	D	esign Import		_ = ×
Basic Advanced Delay Calculation GDS ILM IPO/CTS OpenAccess Power RC Extraction RTL SI Analysis Timing Yield MMMC	Po Gro	esign Import wer Nets: VDD und Nets: VSS Rate Scale Factor:	1.0	
<u>2</u>	1 <u>S</u> ave	Load	Cancel	Help

It is a good idea to save your design using **Design -> Save Design As -> SoCE**.

Next step is to specify the floorplan for your design, choose Floorplan -> Specify Floorplan. Define an aspect ratio of 1 and core utilization of 85% which means that 15% of the core area will be free for possible future cell replacements. Choose core to die boundary large enough to hold the power rings, we choose it as 0.6 microns. This is enough for this design as there will be one power ring and one ground ring of 0.1 micron and spacing between them of 0.1 micron also.

	Specify Floor	plan 📑	_ 0
Basic Advance	d		
– Design Dimensi	ions		
	Size 🗸 Die/IO/Core	Coordinates	
	by: 🔶 Aspect Ratio:		0 97297297
	s by. 👻 Aspect Natio.	 Core Utilization: 	Contraction of the local division of the loc
		Cell Utilization:	0.885135
	🗸 Dimension:	Width:	7.4
		Height:	7.2
🗸 Die Size I	oy:	Width:	8.6
		Height:	8.4
Core Margins	by: 💊 Core to IO Bou	indary	
	Core to Die Bo	undary	
	Core to Left: 0.6	Core to Top:	0.6
С	ore to Right: 0.6	Core to Bottom:	0.6
IO Box Calcu	lation Use: 🛛 🗸 M	lax IO Height 🔶 Mir	n IO Height
Floorplan Ori	gin at: 🔶 Lo	wer Left Corner 💊 C	Center Unit: Micron
<u>o</u> k	Apply	<u>C</u> ancel	Help

Your floor plan will look like the following one.



Choose Floorplan -> Connect Global Nets to connect power nets to your design. From the following menu type VDD in pin name and VDD in the field of **To Global Net** then click on **Add to List**. Make similar steps for VSS then **Apply** and **Close** this window.

	Global Net Connections 📑 💶 🗆 🛪
Connection List	Power Ground Connection Connect Pin Tie High Tie Low Instance Basename: Pin Name(s) VDD Net Basename: Scope Single Instance: Under Module: Under Region: Ibx 0.0 Iby: 0.0 ury: 0.0 Apply All To Global Net VDD Override prior connection Verbose Output Add to List Update Delete
<u>Apply</u>	k Reset Close Help

In the command window you will get a report of these connected power pins like the following figure.

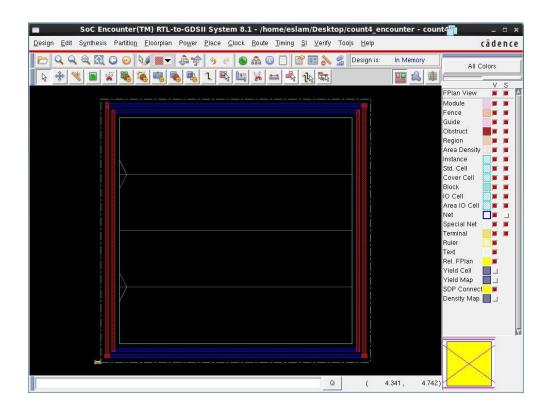
Saving route Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> Writing Netlist "/home/eslam/Desktop/count4_encounter/count4-clk10n s-import.enc.dat/count4.v" Saving configuration Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving floorplan Saving plocement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveVieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1>	<u>File</u>		and the second second second second	host:/hor Terminal		, in the second s	incertor in							
<pre>*** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> Writing Netlist "/home/eslam/Desktop/count4_encounter/count4-clk10n s-import.enc.dat/count4.v" Saving configuration Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving Drc markers Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1></pre>	and the second	- and the second se	0.000		- <u>-</u> p									ĵ,
<pre>*** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> Writing Netlist "/home/eslam/Desktop/count4_encounter/count4-clk10n s-import.enc.dat/count4.v" Saving configuration Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/siFix.option' Saving floorplan Saving floorplan Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encount</pre>				e (cou=0	:00:00.	0 real	=0:00:	00.0	mem=19	6.5M)	***			1
Creating constraint file encounter 1> Writing Netlist "/home/eslam/Desktop/count4_encounter/count4-clk10n s-import.enc.dat/count4.v" Saving configuration Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter													**	1
encounter 1> Writing Netlist "/home/eslam/Desktop/count4_encounter/count4-clk10n s-import.enc.dat/count4.v" Saving configuration Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1	2000-2007 C													
<pre>s-import.enc.dat/count4.v" Saving configuration Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1> enc</pre>					/home/e	slam/D	esktop	/coun	t4 end	ounte	er/co	unt4-	clk10	n
Saving preference file /home/eslam/Desktop/count4_encounter/count4-clk10ns-impor t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1> enco					1.000 CT 1000 CT 1000	1993 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 -			100					1
<pre>t.enc.dat/enc.pref.tcl Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1> encounter</pre>	Saving co	nfigur	ation	8										1
Saving SI fix option to '/home/eslam/Desktop/count4_encounter/count4-clk10ns-imp ort.enc.dat/siFix.option' Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Creating constraint file encounter 1> encounter 1>	Saving pr	eferen	ce file	/home/es	slam/De	sktop/	count4	enco	unter/	count	4-cl	k10ns	-impo	r
ort.enc.dat/siFix.option' Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1> encounte	.enc.dat	/enc.p	ref.tcl			1003								1
Saving floorplan Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1	Saving SI	fix o	ption to	/home,	/eslam/	Deskto	p/coun	it4_en	counte	r/cou	int4-	clk10	ns-im	p
Saving Drc markers No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1				on'				C20040						
<pre> No Drc file written since there is no markers found. Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1</pre>														1
Saving placement *** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1> encounte														1
<pre>*** Completed savePlace (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1</pre>				n since t	there i	s no m	arkers	foun	d.					
Saving route *** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> en counter 1> 10 new pwr-pin connections were made to global net 'VDD'.	J 1					900 85								1
*** Completed saveRoute (cpu=0:00:00.0 real=0:00:00.0 mem=196.5M) *** *** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> en counter 1> 10 new pwr-pin connections were made to global net 'VDD'.				e (cpu=0	:00:00.	0 real	=0:00:	00.0	mem=19	6.5M)	***			
*** Completed saveYieldMap (cpu=: 0:00:00.0 real=0:00:00.0 mem=: 0.000M) *** Creating constraint file encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> en counter 1> 10 new pwr-pin connections were made to global net 'VDD'.				(12)		10								1
Creating constraint file encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> en counter 1> 10 new pwr-pin connections were made to global net 'VDD'.														1
encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> encounter 1> en counter 1> 10 new pwr-pin connections were made to global net 'VDD'.	2000-2007 Co. 10				u=: 0:0	0:00.0	real=	0:00:	00.0 m	iem=:	0.00	0M) *	**	
counter 1> 10 new pwr-pin connections were made to global net 'VDD'.														
	A DESCRIPTION OF A DESC	A DESCRIPTION OF A DESC	And the second	And the second	the second s	AND REAL PROPERTY OF THE PARTY OF	A DESCRIPTION OF THE OWNER OWNER	A DOWNER WATCHING TO A DOWN	the second s	A REAL PROPERTY AND	the state of the s	unter	1> e	1
10 new gnd-pin connections were made to global net 'VSS'.										t VD	· · 0			
	0 new gn	d-pin	connect	lons were	e made	to glo	bal ne	t VS	5'.					

Next step is to add power rings to your design. To do this, choose Power ->

Power Planning -> Add Rings and the following window will show up. From this window, choose the width of your power rings and the spacing between them.

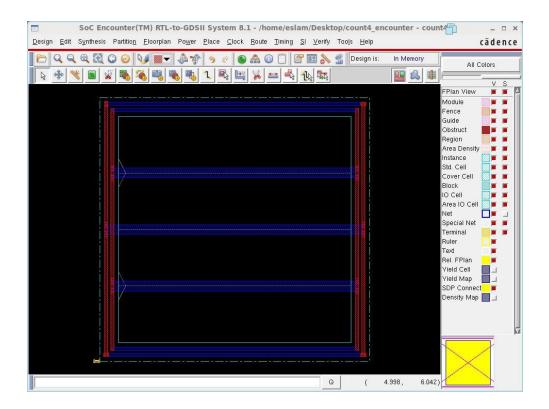
	VSS VDD					
Ring Typ	e					
	ring(s) conto					
🔶 A	round core b	boundary	🗸 Along	/O boundary		
	clude select					
	: ring(s) arou	ind				
	ach block					
	ach reef	an dama in Mana				
		er domain/fenc I block and/or				
V C						
C						
S	lusters of sel	lected blocks a				
Ξ	lusters of sel J With share	lected blocks and ring edges			Maura	tick [
L User	lusters of sel J With share defined coor	lected blocks a ad ring edges rdinates:	and/or groups		MouseC	lick
L User	lusters of sel J With share defined coor	lected blocks and ring edges	and/or groups		MouseC	lick
User	lusters of sel J With share defined coor	lected blocks a ad ring edges rdinates:	and/or groups		MouseC	lick
User	lusters of sel 1 With share defined coor Core ring	lected blocks a ad ring edges rdinates:	and/or groups		MouseC	lick
User	lusters of sel J With share defined coor Core ring nfiguration	lected blocks s ad ring edges rdinates: Block r Bottom:	and/or groups	of core rows	MouseC	lick
User	lusters of sel 1 With share defined coor Core ring nfiguration Top:	lected blocks s ad ring edges rdinates: Block r Bottom:	ing Left:	of core rows	MouseC	lick
User	lusters of sel g With share defined coor Core ring Infiguration Top: ME1 H = 0.1	lected blocks s rdinates: ✓ Block r Bottom: ME1 H	ing Left:	of core rows Right: ME2 V	<u>MouseC</u>	lick
User User Ring Cor ayer: Vidth: Spacing:	lusters of sel g With share defined coor Core ring Infiguration Top: ME1 H = 0.1	lected blocks a rdinates: Block r Bottom: ME1 H 0.1 0.1	Ing Left: ME2 V	Right ME2 V =	-	lick

Your floorplan will now look like the following one.



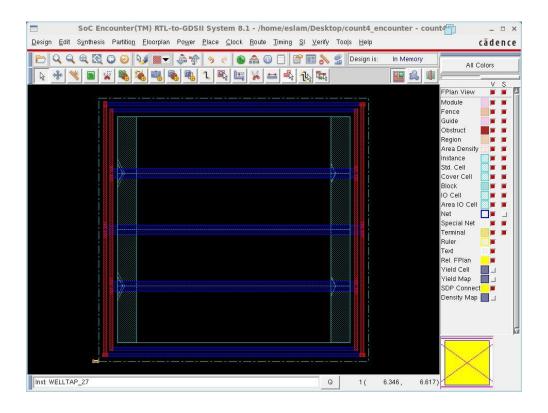
Save your design to this point using **Design -> Save Design AS -> SoCE** and choose an appropriate name, we choose count4_clk10ns-pring.enc.

Now, it is possible to route the power grid. Select **Route -> Special Route** then clock **OK**. After this step you will get the following.



Next step is to add well taps to your design so that your VDD and GND are connected to substrate and n-wells respectively. This is to help tie them to your VDD and GND levels so that they don't drift too much. Choose **Place -> Physical Cells -> Add Well Tap.** Choose the well tap from your technology library and click **OK**.

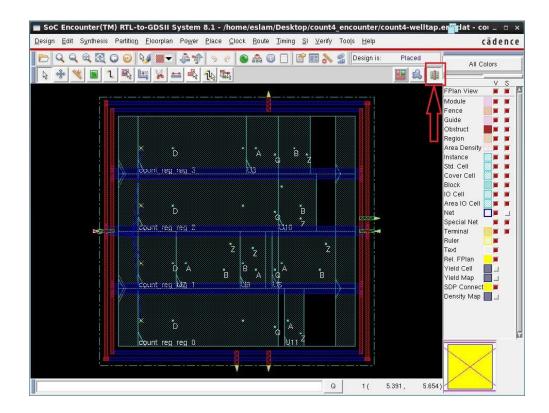
_	Add Well Tap	Instances 🛅	_ = ×
Cell Name WT3R			Select
Max. Gap betweer	n Cells along Row	10	
Offset from Start of	Row for First Cell i	n Row 0.0	
	ws to Skip for Next art Adding Cells 1	Row of Cells 0	
Fill Area	raw	-	lly
<u></u> K	Apply	<u>C</u> ancel	Help



Next step is placing the standard cells. Choose **Place -> Standard Cells** and check the following menu then click **OK**.

		Place			_ = >
🔶 Run Full I	Placement 🗸 R	un Incremental Pla	icement 🗸 Ru	n Placement In Fl	loorplan Mode
-Optimization	Options	ation			
	-Place Optimizati	2007 B			
umber of Thre	ad(s): 1 5	et Multiple CPU			
<u>0</u> K	<u>A</u> pply	Mode	Defaults	<u>C</u> ancel	<u>H</u> elp

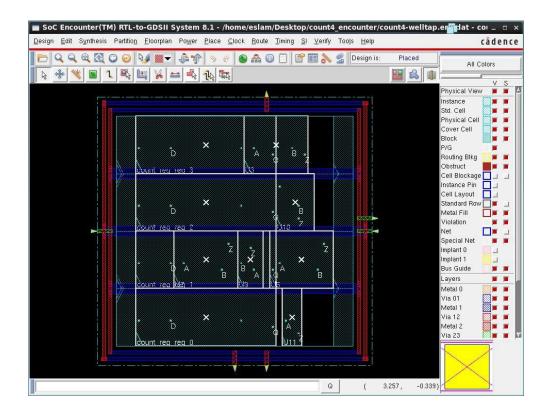
Your standard cells are now placed and you can check them.



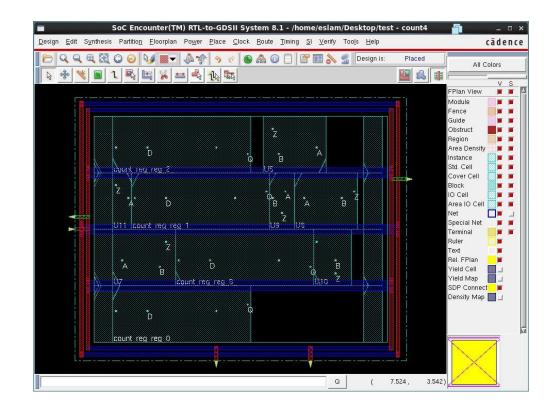
To check placement, choose Place -> Check Placement and click OK.

		Place		.	_ = ×
🔶 Run Full P	lacement 🕹 Ri	un incremental Pla	cement 💸 Ru	in Placement in F	loorplan Mode
	<mark>Options</mark> Place Optimiza Place Optimizatio				
Number of Threa		et Multiple CPU			
<u>o</u> k	Apply	Mode	Defaults	<u>C</u> ancel	Help

After clicking OK we got these violations, there was an overlapping between some standard cells.



To solve these violations we edited the aspect ratio again from Floorplan -> Specify Floorplan and the problem had been solved.



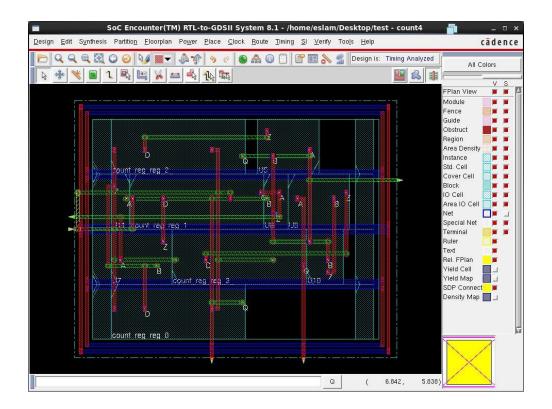
Check the file named .checkPlace in your work directory to make sure that there are no violations.

🍘 count4.checkPlace [Read Only] (~/Desktop/test) - 🔤 dit 💷 🗆 🗙
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>S</u> earch <u>T</u> ools <u>D</u> ocuments <u>H</u> elp
💁 🔄 Open 👻 🖄 Save 🛛 📇 👘 🗳 Undo 🗞 🛛 🖓 🖷 📳 🔶 👻
📄 count4.checkPlace 💥
<pre>1 ####################################</pre>
Plain Text 🗸 Tab Width: 8 🗸 Ln 1, Col 1 INS

Now choose **Timing -> Optimize** to make the timing optimization for Pre-CTS.

Click $\boldsymbol{O}\boldsymbol{K}$ in the following menu.

Pre-CTS	🔷 Post-CTS		V Post-Route	
Optimization Type-				
🛛 Setup		🔟 Hold		
🗢 Incremental				
🔶 Design Rules Vi	olations			
🍯 Мах Сар				
📕 Max Tran				
🔟 Max Fanout				
Include SI SI C	Satis 1			



Now choose Clock -> Design Clock then click Gen Spec and select your CLKBUF cells.

Clock Spe	ecification Files:		 Gen Spec	
Results D	irectory: clock_r	eport		

CKBUFM1R CKBUFM20R CKBUFM22RA CKBUFM24R CKBUFM26RA CKBUFM28RA	Add	CKBUFM1R CKBUFM20R CKBUFM22RA CKBUFM2RA CKBUFM2RA CKBUFM2R CKBUFM32R	
CKBUFM32R CKBUFM3R CKBUFM40R CKBUFM4R CKBUFM6R CKBUFM6R CKINVM12R CKINVM18	Delete	CKBUFM3R CKBUFM40R CKBUFM4R CKBUFM4R CKBUFM6R CKBUFM6R	

Then clock **OK**.

Now choose **Clock -> Display -> Display Clock Tree**. From the following menu you can display the clock phase delay.

	Display Clock T 🔁 💶 >
a	ock Selection
٠	All Clock(s)
~	Selected Clock
R	oute Selection
V	Pre-Route
٠	Clock Route Only
Ŷ	Post-Route
Di	splay Selection
*	Display Clock Tree
	All Level
	✓ Bottom Level (non-gated clock tree only)
	Selected Level (non-gated clock tree only)
	1
~	Display Clock Phase Delay
Ŷ	Display Min/Max Paths
	OK Apply Cancel Help

Now choose **Timing -> Optimize** then select **Post-CTS** and click **OK**.

V Post-Route		Optimization	ġ.		_ 🗆 🗙
Hold	− <mark>Design Stage</mark> ✓ Pre-CTS	Post-CTS	~	Post-Route	
Hold	Optimization Type				
	👅 Setup	1	Hold		
	🗸 Incremental				
	🔶 Design Rules Violation	s			
	🍺 Мах Сар				
	📕 Max Tran				
	🔟 Max Fanout				
	I Include SI SI Option:	. 1			
		ан			

Open your terminal window and check the WNS field (Worst Negative Slack) which means the slack of the critical path in your design.

Now choose **Timing -> NanoRoute -> Route**. From the following menu check the required fields and click **OK**.

		Nano	Route			_ 0
Routing Phase						
📕 Global Route						
👅 Detail Route	Start Iteration	default	End Iteration	default		
ost Route Optimization 🚊	Optimize Via 🔟 🤇	Optimize Wire				
Concurrent Routing Feat	Ires					
📕 Fix Antenna	_ Insert Dioc	les	Diode Cell Nam	e	7	
		Congestion	Timin	g		
Timing Driven	Effort 5				S.M.A.R.T.	
SI Driven				2		
Post Route SI	SI Victim File					
🔟 Litho Driven						
Post Route Litho Repaired	air					
Routing Control						
Selected Nets Only	Bottom Layer	default	Top Layer	default		
ECO Route						
_ Area Route	Area			è	Select Area a	nd Route
Job Control						
 Auto Stop 						
lumber of Thread(s) For M	ultiple Threaded: 1					
Number of Thread(s) Fo	r Superthreaded: 1					
Number of Host(s) Fo	r Superthreaded: 0	_				
Set Multiple CPU						
OK	[]			1.2.2		1144
<u>O</u> K <u>A</u> pply	Aţtribute	Mode	<u>S</u> ave	Load	<u>C</u> ancel	Help

Next step is the post route timing optimization. Choose **Timing -> Optimize** and choose **Post-Route** then **OK**.

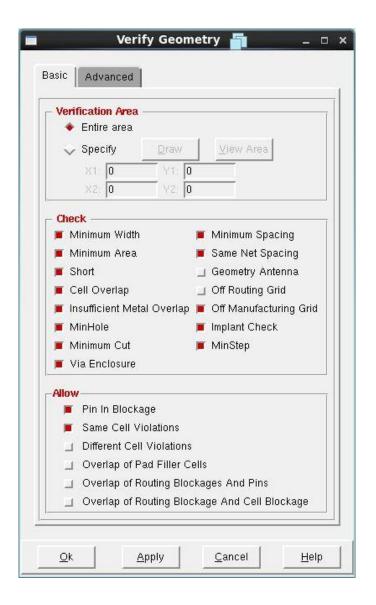
	Optim	ization		_ 0 :
Design Stage V Pre-CTS	🗸 Post-C	TS	Post-Route	
	V 1 03FC	10	• I Ust-Mudie	
Optimization Type				
👅 Setup		🔟 Hold		
🗸 Incremental				
Design Rules Vid	plations			
👅 Мах Сар				
Max Tran				
I Max Fanout				
Include SI SI C	ptions			
		-	10 - 10 - 10 - 10 - 10 - 10 - 10 - 10 -	
<u>о</u> к <u>А</u> рг	oly Mode	and the second se		

Next step is to add filler cells. Choose **Place -> Physical Cells -> Add Filler** and select filler cells from your technology library. Filler cells will fill remaining holes in the rows and ensure the continuity of power/ground rails and N+/P+ wells.

The following steps are to check your design. To check connectivity, choose **Verify -> Verify Connectivity** then press **OK** in the following window.

	Verify Connectivity 📑 🛛 🗛
- Net Type	ī
All	
🗢 Regular Only	
🕹 Special Only	
Nets	
🔶 All	
🥪 Selected	
💊 Named:	
Check Open Connectivity Loop Geometry Loop	 UnConnected Pin Unrouted Net Antenna Weakly Connected Pin Geometry Connectivity Keep Previous Results
Connect Pad Speci	al Ports
Verify Connecti	vity Report: count4.conn.rpt File
Error: 1000	Warning: 50
<u>k</u>	Apply <u>C</u> ancel <u>H</u> elp

To check geometry of your design, choose **Verify -> Verify Geometry** then click **OK**.



You can now generate some useful reports. Choose **Design -> Report -> Netlist Statistics** and check your command window to see this report file.

You can also check the number of gates used in your design by choosing **Design** -> **Report -> Gate Count** and click OK in the next window.

	Gate Co	unt Report		×
	lls and Blocks IlsOnly			
To File Co Desig				
		Cancel	Help	

In command window, type *report_timing* to get a timing report for your design.

To export a standard delay format file type the following in command window write_sdf count4_clk10ns_pared.sdf -version 2.1

The P+R netlist may be different from the imported netlist as cells may have been added or replaced during clock tree synthesis (CTS) and various timing optimization phases. To export this netlist choose **Design -> Save -> Netlist** and click **OK** in the following window.

	Save Netlist 🛅	and.	×
📕 Include Interm	ediate Cell Definition		
🔟 Include Leaf C	Cell Definition		
Netlist File: count4	1.v		0
<u></u> K	<u>C</u> ancel	Help	

The next step is to export this layout to a file which can be used in virtuoso layout editor, choose **Design -> Save -> GDS/OASIS**. Choose an appropriate **Map File**. The library name is the design library name in virtuoso. Choose merge files and merge it with the .gds files from your technology library.

🖬 🛛 GDS/OASIS Export 📑 💷 🛛 🛪
Output Format 🔹 🔶 GDSII/Stream 🗸 OASIS
Output File count4_clk10ns_pared.gds
Map File streamOut.map 🖻
Library Name DesignLib
Structure Name count4
Attach Instance Name to Attribute Number
Attach Net Name to Attribute Number
👅 Merge Files 🛛 /gds/uk65lscllmvbbr.gds 🔹 🖻 🔄 🔲 Uniquify Cell Names
☐ Stripes 1
🔟 Write Die Area as Boundary
Write abstract information for LEF Macros
Units 2000 🛶
Mode ALL
<u>OK</u> <u>Apply</u> <u>Cancel</u> <u>H</u> elp

For more details please check this video:

http://youtu.be/udPMw9_rZL0

Part 2:

The same steps of part 1 will be done here using this TCL script.

Importing the Design
loadConfig Default.conf
setDrawView fplan
fit
saveDesign count4-import.enc
#Floorplanning the Design
floorPlan -r 1 0.85 0.6 0.6 0.6 0.6
saveDesign count4-fplan.enc
#Power Planning
clearGlobalNets
globalNetConnect VDD -type pgpin -pin VDD -inst * -module {} -verbose
globalNetConnect VSS -type pgpin -pin VSS -inst * -module {} -verbose
addRing \
-around core \
-nets {VSS VDD} \
-center 1 \
-width_bottom 0.1 -width_right 0.1 -width_top 0.1 -width_left 0.1 \
-spacing_bottom 0.1 -spacing_right 0.1 -spacing_top 0.1 -spacing_left 0.1 \
-layer_bottom ME1 -layer_right ME2 -layer_top ME1 -layer_left ME2 \
-bl 1 -br 1 -rb 0 -rt 0 -tr 0 -tl 0 -lt 1 -lb 1
#placing well taps
addWellTap -cell WT3R -maxGap 10 -skipRow 1 -startRowNum 2 -prefix WELLTAP
#special route
sroute \
-connect { blockPin corePin floatingStripe } \
-blockPin { onBoundary bottomBoundary rightBoundary } \
-allowJogging 1
saveDesign count4.enc
#Placing the standard cells
setPlaceMode -timingDriven true
placeDesign -prePlaceOpt
setDrawView place checkPlace
optDesign -preCTS -outDir /home/eslam/Desktop/encounter
saveDesign count4-placed.enc
#Synthesizing a Clock Tree
createClockTreeSpec -output count4_spec.cts \

-bufferList CKBUFM12R CKBUFM16R CKBUFM1R CKBUFM20R CKBUFM22RA CKBUFM24R CKBUFM26RA CKBUFM2R CKBUFM32R CKBUFM3R CKBUFM40R \ CKBUFM48R CKBUFM4R CKBUFM6R CKBUFM8R CKINVM12R CKINVM16R CKINVM1R CKINVM20R CKINVM22RA CKINVM24R CKINVM26RA \CKINVM2R CKINVM32R CKINVM3R CKINVM40R CKINVM48R CKINVM4R CKINVM6R CKINVM8R clockDesign -specFile pfd_loopF_spec.cts \ -outDir /home/eslam/Desktop/encounter optDesign -postCTS -outDir /home/eslam/Desktop/encounter saveDesign count4-cts.enc #Routing the Design setNanoRouteMode -routeWithTimingDriven true -routeTdrEffort 5 routeDesign optDesign -postRoute -outDir /home/eslam/Desktop/encounter saveDesign count4-routed.enc **#Design Finishing** addFiller \ -cell { FIL16R FIL1R FIL2R FIL32R FIL4R FIL64R FIL8R FILE16R FILE32R FILE3R FILE4R FILE64R FILE6R \FILE8R FILEP16R FILEP32R FILEP64R FILEP8 } \ -prefix FIL setDrawView place saveDesign count4-filled.enc #Checking the Design verifyConnectivity -type all -report connectivity.rpt verifyGeometry -report geometry.rpt #Generating Reports reportNetStat reportGateCount -outfile gateCount.rpt summaryReport -outdir /home/eslam/Desktop/encounter #Design Export write_sdf -version 2.1 -precision 4 count4_pared.sdf saveNetlist -excludeLeafCell count4_pared.v streamOut count4_pared.gds \ -mapFile streamOut_me_pinOnly.map \ -libName count4_pared \ -merge uk65lscllmvbbr.gds

For more details, please check the following video:

http://youtu.be/NEfx3igkzME

Conclusion

Through this tutorial, we started with a gate level netlist (The file we got from SYNOPSYS Design Vision) and followed all the steps to place and route the standard cells in this netlist file. The final output from this tutorial is a layout file and timing constraints files which can be used for post layout simulations.

Appendix E

POWER CALCULATION

The method to measure power using Cadence Spectre is described in this tutorial, the 2-bit inverter in the below figure is used as an example to show how power measurement is done in cadence spectre .

The equations we need to apply to Calculate the average power consumed are :

The Instantaneous Power : **P** (t) = Power supply voltage (VDD) * current drawn from power supply at time Then the Average Power : Average Power = $\frac{1}{Time Period} \int_{Time Period}$ Instantaneous Power

The following changes needs to be done for the measurement of the power drawn from the power supply .

1. Changes to the Existing Schematic :

- On the top-level of the schematic, add a Vdc source (from the analog library) and connect its positive terminal to the VDD.
- Select the Vdc source (a white box appears around the selected item), and press Q, an edit object properties window will appear Type he power supply value (which is 3v in this example) across DC Voltage and press OK.

Important note :This addition of the Vdc source has to be done only to the toplevel of the design schematic and **SHOULD NOT** be done for each of the blocks in the project.

Now, Save the sheet (check and save) and go to the analog-environment window to perform the simulation.

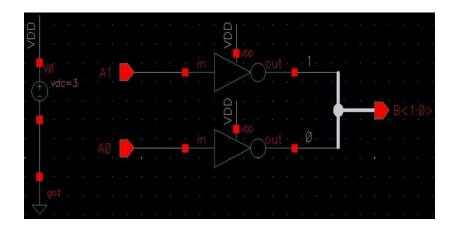


Figure 1

2. Simulation

- Make all the necessary set-up for the simulation
- To plot the current drawn from the VDD, select Output-> To Be Plotted - > Select on Schematic in the analog-environment window and then select the -ve terminal of the Vdc source in the schematic (because the current plotted for a certain node is the input to this node).
- A circle appears in the schematic as shown in figure 2, make sure the circle appears. If it does not appear, then you are plotting the voltage and not the current.



Figure 2

 Simulate the circuit and the plot of the output current from the VDD will be as shown in figure 3

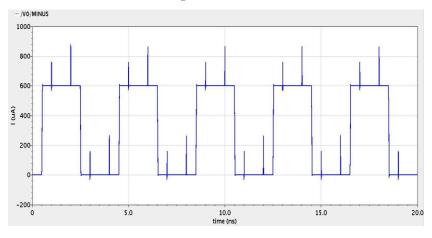


Figure 3

- Select the tools -> calculator from the analog-environment window
- To create TRAN current expression, check the (it) choice and then reselect the -ve terminal of the Vdc source in the schematic, after that select the function "INTEG" from the built-in functions. Figure 4 clarify this step.

<u>File To</u>	ols <u>V</u> iew <u>O</u> ptio	ns <u>C</u> onstant	ts <u>H</u> elp							cā	der
										_	
 Result 	ilts Dir: /root/sim	ulation/Powe	r_T/spectre/scl	hematic/psf							
_	vf vdc vdc										
_				1 1 11	0 3- 4						
● Off 〇	Family 🔾 Wave	🗹 Clip	App	end							
- 1.50	Pop Bo		Mare I Man		- 6						
- 2	🏅 Pop 📑		∭≋ M+ ∣	ME E+ EE	9						
- 2 [🎦 Pop 📭		∭≋ M+	ME E+ EE	9	Ĉ					
	Ъ Pop 📝		∦≋ M +	ME E+ EE	9	e			7	8	9
All	1999	-		ME E+ EE					7	8	9
All	bandwidth	dBm	fourEval	gp	intersect	nfmin	psd	\$21	7	8	9
All 1/x 10**x	bandwidth	dBm delay	fourEval	gp gpc_freq	intersect ipn	nfmin overshoot	psdbb	s22		-	-
All 1/x 10**x Rn	bandwidth clip compare	dBm delay deriv	fourEval freq_ freq_jitter	gp gpc_freq gpc_gain	intersect ipn ipn∀RI	nfmin overshoot peak	psdbb pzbode	s22 san		-	-
All 1/x 10**x Rn a2d	bandwidth clip compare compression	dBm delay deriv dft	fourEval freq freq_jitter frequency	gp gpc_freq gpc_gain groupDelay	intersect ipn ipn∀RI kf	nfmin overshoot peak peakToPeak	psdbb pzbode pzfilter	s22 san sett		5	6
All 1/x 10**x Rn a2d abs	bandwidth clip compare compression compressionVRI	dBm delay deriv dft dftbb	fourEval freq freq_jitter frequency ga	gp gpc_freq gpc_gain groupDelay gt	intersect ipn ipnVRI kf In	nfmin overshoot peak peakToPeak period_jitter	psdbb pzbode pzfilter real	s22 san sett sin		5	6
All 1/x 10**x Rn a2d abs acos	bandwidth clip compare compression compressionVRI conjugate	dBm delay deriv dft dftbb dnl	fourEval freq_ freq_jitter frequency ga gac_freq	gp gpc_freq gpc_gain groupDelay gt gumx	intersect ipn ipnVRI kf In Ioadpull	nfmin overshoot peak peaktoPeak period_jitter phase	psdbb pzbode pzfilter real riseTime	s22 san sett sin sinl	4	5	6
All 1/x 10**x Rn a2d abs acos acos	bandwidth clip compare compression VRI conjugate convolve	dBm delay deriv dft dftbb dnl dutyCycle	fourEval freq freq_jitter frequency ga gac_freq gac_gain	gp gpc_freq gpc_gain groupDelay gt gumx harmonic	intersect ipn ipnVRI kf In Ioadpull log10	nfmin overshoot peak peakToPeak period_jitter phase phaseDeg	psdbb pzbode pzfilter real riseTime rms	s22 san sett sin sinl slev	4	5 2 ±	6
All 1/x 10**x Rn a2d abs acos acos asin	bandwidth clip compare compression compressionVRI conjugate convolve cos	dBm delay deriv dft dftb dnl dutyCycle evmQAM	fourEval freq_ freq_jitter frequency gac_freq gac_gain gainBwProd	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq	intersect ipn kf In Ioadpull Iog10 Isb	nfmin overshoot peak peakToPeak period_jitter phase phaseDeg phaseDegUnwrapped	psdbb pzbode pzfilter real riseTime rms rmsNoise	s22 san sett sin sin slev spe	4 1 0 use	5 2 ±	6 3
All 1/x 10**x Rn a2d abs acos acos acos acos asin asin	bandwidth clip compression compression VRI conjugate convolve cos cosh	dBm delay deriv dft dft dft dft dft dty Cycle evmQAM evmQpsk	fourEval freq freq_jitter frequency ga gac_freq gac_gain gainBwProd gainMargin	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo	intersect ipn kf In Ioadpull Iog10 Isb Ishift	nfmin overshoot peak peakToPeak period_jitter phase phaseDeg phaseDegUnwrapped phaseMargin	psdbb pzbode pzfilter real riseTime rms rmsNoise rn	s22 san sett sin sin slev spe spe	4 1 0 use	5 2 ±	6
All 1/x 10**x Rn a2d abs acos acosh asin asinh atan	bandwidth clip compression compression compressionVRI conjugate convolve cos cosh cross	dBm delay deriv dft dftbb dnl dutyCycle evmQAM evmQpsk exp	fourEval freq freq_itter frequency gac_gain gainBwProd gainMargin getAscilWave	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo iinteg	intersect ipn ipnVRI kf In Ioadpull Iog10 Isb Ishift mag	nfmin overshoot peak peakToPeak period_jitter phaseDeg phaseDeg phaseDeg phaseNargin phaseNoise	psdbb pzbode pzfilter real riseTime rms rmsNoise rn root	s22 san sin sin slev spe spe spe	4 1 0 use	5 2 ±	6 3
All 1/x 10**x Rn a2d abs acos acos acos asin asinh atan atanh	bandwidth clip compare compression VRI conjugate convolve cos cosh cross d2a	dBm delay deriv dft dftbb dnl dutyCycle evmQAM evmQpsk exp eyeDiagram	fourEval freq_jitter frequency ga_gac_gain gainBwProd gainBwProd gainBwProd gainAargin getAsciiWave gmax	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo iinteg iinteg	intersect ipn ipnVRI kf loadpull log10 Isb Ishift mag nc_freq	nfmin overshoot peak peakToPeak period_jitter phaseDeg phaseDegUnwrapped phaseDegUnwrapped phaseMargin phaseNoise phaseRad	psdbb pzbode pzfilter real riseTime rms rmsNoise rm root rshift	s22 san sin sin slev spe spe spe sqr	4 1 0 use	5 2 ±	6 3
All 1/× 10**× Rn a2d abs acos acosh asinh atanh atanh atanh average	bandwidth clip compare compression compression vRI conjugate convolve cos cos cross d2a dB10	dBm delay deriv dftb dftbb dnl dutyCycle evmQAM evmQpsk evp eyeDiagram fallTime	fourEval freq_litter freq_litter frequency gac_gain gainBavProd gainMargin getAsciiWave gmax gmin	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonic harmonic Freq histo iinteg iinteg	intersect ipn ipnVRI kf ln loadpull log10 lsb lshift mag nc_freq nc_gain	nfmin overshoot peak peakToPeak period_jitter phase phaseDegUnwrapped phaseNargin phaseNoise phaseRad phaseRad phaseRad	psdbb pzbode pzfilter real riseTime rms rmsNoise rn root rshift s11	s22 san sett sin sin spe spe spe sqr ssb	4 1 0 use	5 2 ±	6 3
All 1/x 10**x Rn a2d abs acos acos acos asin asinh atan atanh	bandwidth clip compare compression VRI conjugate convolve cos cosh cross d2a	dBm delay deriv dft dftbb dnl dutyCycle evmQAM evmQpsk exp eyeDiagram	fourEval freq_jitter frequency ga_gac_gain gainBwProd gainBwProd gainBwProd gainAargin getAsciiWave gmax	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo iinteg iinteg	intersect ipn ipnVRI kf loadpull log10 Isb Ishift mag nc_freq	nfmin overshoot peak peakToPeak period_jitter phaseDeg phaseDegUnwrapped phaseDegUnwrapped phaseMargin phaseNoise phaseRad	psdbb pzbode pzfilter real riseTime rms rmsNoise rm root rshift	s22 san sin sin slev spe spe spe sqr	4 1 0 use	5 2 ±	6 3

Figure 4

The calculator window will appear as shown in figure 5 .This simulation (for the 2-bit inverter) is done from 0n to 20ns. Let's find the average power consumed by this circuit in this period, thus the integration should performed in this period (figure 5).In the Signal text box, multiply the current waveform by 3 and divide by 20ns (figure 5).Press OK.

Pop P	Virtuoso (R) Visualization & Analysis XL calculator	_		- 5	×
vt vf vdc vs op vn sp vswr hp zm it if idc is opt mp vn2 zp yp gd data Off Family Wave ✓ Clip Image: Append Image: Append <td< td=""><td>Eile Tools View Options Constants Help</td><td></td><td>cā</td><td>der</td><td>1 c e</td></td<>	Eile Tools View Options Constants Help		cā	der	1 c e
it if idc is opt mp vn2 zp yp gd deta Off Family Wave ✓ Clip Append Image: Clip Image: Clip <td>Results Dir: /root/simulation/Power_T/spectre/schematic/psf</td> <td></td> <td></td> <td></td> <td></td>	Results Dir: /root/simulation/Power_T/spectre/schematic/psf				
TT("/V0/MINUS")					
All 7 8 integ 4 5 Signal IT("/V0/MINUS" 3/20n 1 Initial Value 0 1 Final Value 20n user 1	🔾 Off 🔾 Family 🔾 Wave 🗹 Clip 🦏 🐗 Append 🔤 📑				
All 7 8 9 integ 4 5 6 Signal T("/V0/MINUS" 3/20n 1 2 3 Initial Value 0 - 0 1 0 Final Value 20n - User 1 User 1 User 1 User 1 User 1	IT("/V0/MINUS")		_		_
integ 4 5 6 Signal (TC/V0/MINUS*3/20n 1 2 3 Initial Value 0 0 ± Final Value 20n user 1 user	🕇 ┛ 🚯 Pop 💯 😰 🏽 🗱 🕅 + ME E+ EE 🥱 🧭				
Signal IT("/Y0/MINUS" 3/20n Initial Value 0	All	7	8	9	7
Initial Value 0	integ	4	5	6	-
Final Value 20n - user 1 user 1	Signal IT("/V0/MINUS" 3/20n	1	2	3	-
user i user i		0	±		+
	Final Value 20n	use	er 1	use	r 2
user 3 user		use	er 3	use	r 4
	QK Apply Defaults Close				
OK Apply Defaults Close	Successful evaluation	11			-

Figure 5

• The expression for power calculation appears in the result text-box. Press "EVAL" from the keypad in the calculator. The average power consumed by this circuit will be displayed in the result textbox, as shown in figure 6.

						is XL calculator				1.12		
Eile To	ols <u>V</u> iew <u>O</u> ptio	ns <u>C</u> onstant	ts <u>H</u> elp						_	ca	der	n c
Resu	ilts Dir: /root/sim	ulation/Powe	r_T/spectre/scl	nematic/psf								
200	vf Ovdc O if Oidc O											
Off O	Family 🔾 Wave	Clip		end								
6.7E-6												
	Pop Bitt	02 Boo	Mase I auto	MELE. E	- 1 -							
2	Pop		∭≋ M+	ME E+ E	9							
2	Pop 📴		∦≋ M+	ME E+ EE	= 🥱	¢	_		7	8	9	6
/x	bandwidth	dBm	fourEval	gp	intersect		psd	s21	7	8	9	
/× 0**× 8n	bandwidth clip compare	dBm delay deriv	fourEval freq freq_litter	gp gpc_freq gpc_gain	intersect ipn ipn∀RI	overshoot peak	psdbb pzbode	s22 san			-	
/× 0**× Rn i2d ibs	bandwidth clip compare compression compressionVRI	dBm delay deriv dft dftbb	fourEval freq freq_jitter frequency ga	gp gpc_freq gpc_gain groupDelay gt	intersect ipn ipn∀RI kf In	overshoot peak peakToPeak period_jitter	psdbb pzbode pzfilter real	s22 san sett sin		5	6	
I/X IO""X A2d Abs Acos Acos	bandwidth clip compare compression	dBm delay deriv dft dftbb dnl dutyCycle	fourEval freq freq_jitter frequency ga gac_freq gac_gain	gp gpc_freq gpc_gain groupDelay gt gumx harmonic	intersect ipn kf In loadpull log10	overshoot peak peakToPeak period_jitter phase phaseDeg	psdbb pzbode pzfilter real riseTime rms	s22 san sett sin sinl slev	4	5 2 ±	6 3	
I/X I O""X An A2d Abs Acos Acos Acosh Asin	bandwidth clip compare compression compression VRI conjugate convolve cos	dBm delay deriv dft dftbb dnl dutyCycle evmQAM	fourEval freq_ freq_jitter frequency gac_freq gac_gain gainBwProd	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq	intersect ipn kf In Ioadpull Iog10 Isb	overshoot peak peakToPeak period_jitter phase phaseDeg phaseDeg phaseDegUnwrapped	psdbb pzbode pzfilter real riseTime rms rmsNoise	s22 san sett sin sin slev spe	4 1 0 use	5 2 ±	6 3 use	
I/X IO""X A2d Abs Acos Acosh Asin Asinh Atan	bandwidth clip comprare compression compressionVRI conjugate convolve cos cosh cross	dBm delay deriv dft dftbb dnl dutyCycle evmQAM evmQpsk exp	fourEval freq_ freq_litter frequency ga gac_freq gac_gain gainBwProd gainMargin getAsciiWave	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo	intersect ipn kf In loadpull log10	overshoot peak peakToPeak period_jitter phase phaseDeg phaseDegUnwrapped phaseMargin phaseNoise	psdbb pzbode pzfilter real riseTime rms rmsNoise rn root	s22 san sett sin sinl slev	4 1 0 use	5 2 ±	6 3	1040
All All All All All All All All All All	bandwidth clip compare compression VRI conjugate convolve cos cosh cross d2a	dBm delay deriv dft dftbbb dftbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbbb dftbbbb dftbbbb dftbbbb dftbbbbb dftbbbbbbb dftbbbbbbbbbb	fourEval freq_ifter frequency ga_gac_gain gainBwProd gainBwProd gainAargin getAsciiWave gmax	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo iinteg iinteg	intersect ipn ipnVRI kf In loadpull log10 Isb Isb Ishift mag nc_freq	overshoot peak peakToPeak period_jitter phase phaseDeg phaseDegUnwrapped phaseMargin phaseNoise phaseRad	psdbb pzbode pzfilter real riseTime rms rmsNoise rn root rshift	s22 san sett sin sin slev spe spe spe spe	4 1 0 use	5 2 ±	6 3 use	
/X 0""X In IZd Ibs Icos Icosh Isinh Isinh Isinh Isinh Isinh Isinh	bandwidth clip compression compression compressionVRI conjugate convolve cos cosh cosh cross d2a dB10	dBm delay deriv dftb dftbb dnl dutyCycle evmQAM evmQpsk evp gyeDiagram fallTime	fourEval freq_litter frequency gac_gain gainBavProd gainMargin getAsciiWave gmax gmin	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo iinteg iinteg iint	intersect ipn kf In log10 Isb Ishift mag nc_freq nc_gain	overshoot peak peakToPeak period_jitter phase phaseDeg phaseDegUnwrapped phaseMargin phaseNoise phaseRad phaseRad	psdbb pzbode pzfilter real riseTime rms rmsNoise rn root rshift s11	s22 san sett sin sin slev spe spe spe spe spe spe	4 1 0 use	5 2 ±	6 3 use	
I/X IO""X A2d Abs Acos Acosh Asin Asinh Atan	bandwidth clip compare compression VRI conjugate convolve cos cosh cross d2a	dBm delay deriv dft dftbbb dftbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbb dftbbbb dftbbbb dftbbbb dftbbbb dftbbbbb dftbbbbbbb dftbbbbbbbbbb	fourEval freq_ifter frequency ga_gac_gain gainBwProd gainBwProd gainAargin getAsciiWave gmax	gp gpc_freq gpc_gain groupDelay gt gumx harmonic harmonicFreq histo iinteg iinteg	intersect ipn ipnVRI kf In loadpull log10 Isb Isb Ishift mag nc_freq	overshoot peak peakToPeak period_jitter phase phaseDeg phaseDegUnwrapped phaseMargin phaseNoise phaseRad	psdbb pzbode pzfilter real riseTime rms rmsNoise rn root rshift	s22 san sett sin sin slev spe spe spe spe	4 1 0 use	5 2 ±	6 3 use	

Figure