Department of Electrical and Computer Engineering

Subject Name: Digital ASIC Design

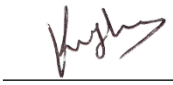
Subject code: Engi_5331

Instructor: Dr. Yushi Zhou

Project report on

Simulation of 1-bit Full Adder in Cadence

Submitted by: Signature

Kush Vyas- 0866563 

Vinaykumar Reddy Velma- 0681954 

CONTENTS

LIST OF FIGURES

List of Tables

Abstract:

Full adder is the functional building block and basic component in several architectures found in VLSI and DSP applications, Adder is a versatile component and mainly used in addition and multiplication as the basic processing element; Adder in a VLSI application is used in ALU design, Address generation in processors, Multipliers and so on. In this project we have considered 28Transistor CMOS design for 1-bit Full adder and these design is analyzed using **CADENCE** Design Suite tool for transistor level design and implementation by the sub-tool **Virtuoso**, **ADE** provided, performance is measured for 180nm technology and a comparative analysis of the adders is coated here with- Number of transistors/ Gate count, Delay, pre–layout and post–layout simulation results. Finally, comparison of the pre-layout and post-layout simulation results. And finding out the best and worst case delays for S and Cout.

Throughout the procedure of this project, the widths of NMOS is taken as 0.500µm, width of PMOS is taken as 1µm. The rise time and fall time for the pulsed voltage source input is taken as 10ps. The on time for the same is 2ns and time period is 4ns. The value of the load capacitor is taken as 94fF(last 2 digits of student's roll number + 40 fF) as per the mentioned in the project design specifications.

Keywords:

Cadence, Virtuoso, Truth table, Analog Environment (ADE), Area (Transistor Count), Layout, Extraction, Test bench, Metal, Contact, Stick Diagram, Diffusion, Hierarchy Editor, Config view, Propagation delay, Contamination delay.

1. Introduction

Summation (Addition) is the basic arithmetic operations and is used in VLSI systems as a full adder circuit extensively. It adds the binary numbers and is the building block for other operations such as subtraction , multiplication, division, etc. The overall performance of the system is mainly dependent on the adder performance. And the performance of adder depends on the number of transistors, its widths and load capacitance. Hence the performance enhancement of a 1-bit full-adder cell is to be done at the initial stages only.

In this report 28Transistor CMOS design is presented. The 28Transistor 1-bit Full adder is designed in CMOS design style. The 1 bit full adder cells are designed using CADENCE Design Suite and transistor level implementation or layout in other words is made using Virtuoso. The proposed Full adder Circuits have their own advantages and disadvantages. The adder discussed in this report is a mirror adder.

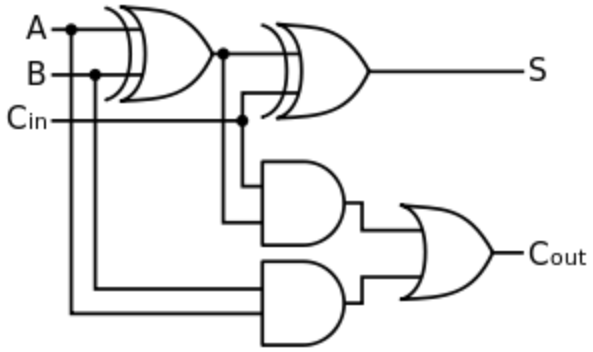


Fig.1: Gate level schematic for 1- bit full adder

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full-adder adds three one-bit numbers, often written as A, B, and Cin; A and B are the operands, and Cin is a bit carried in from the previous stages.

2. Purpose

The purpose of this project was to design and implement a 1-bit Full Adder at the schematic and layout level. This report outlines how both the functionality and the performance of the adder was characterized using Analog Design Environment (ADE) software using Cadence.

3. Design

Circuit Requirements

The project itself entailed a few basic requirements. These requirements were that the circuit could contain a maximum of 28 transistors, several contact pads, poly region, active region, p-plus, n-plus an n well. Upon reviewing the basic requirements, it was decided that the Full Adder would receive inputs sequentially so that the bit width of the adder (number of bits that it could add) could be scaled without increasing the number of input pads. It was also decided that a verification circuit would be added to the system so that characterization could be made easy. This verification circuit allowed for a simple check to see if the adder returned a valid 1-bit result.

4. Schematic of Full Adder

With the use of the Cadence software, schematic level designs were constructed for diagram listed above. The figures below show each schematic and width for the Full Adder. The width here was taken as 0.5µm for the NMOS and 1 µm for the PMOS. The reason behind taking this width was to meet the delay requirement. The width was to be taken such that the worst-case delay should not exceed 4ns. The experimental results show that the widths taken for this project, meets the requirement.

.

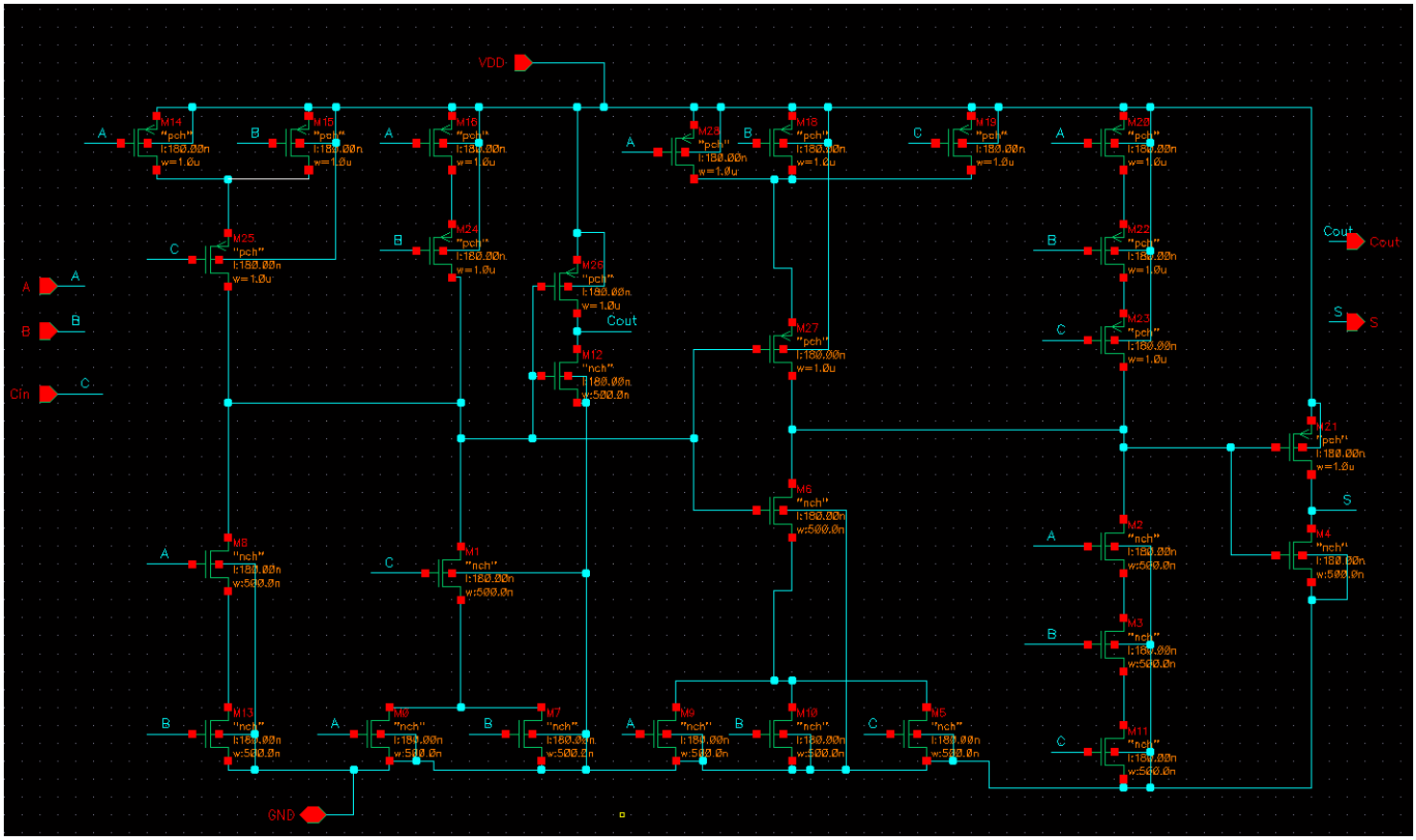


Fig.2: Schematic of Full Adder

1) Symbol

After making the schematic, the next task was to make the symbol of the schematic and use that symbol in the test bench to simulate the adder. After assigning proper names to the pins, the symbol would look like in the fig. 3 below.

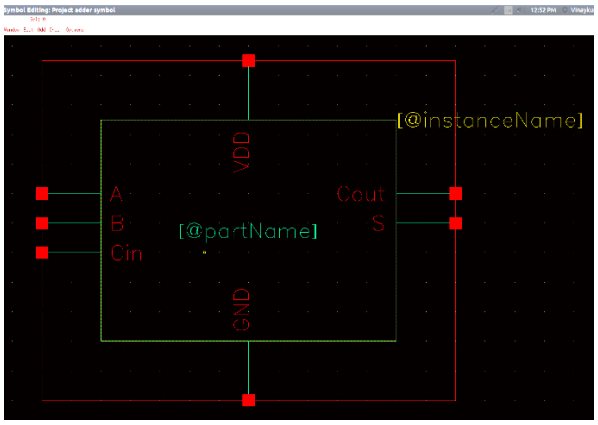


Fig.3: Symbol

5. Simulation of the test bench

After putting the symbol in the schematic editor, all the inputs and outputs were to be connected appropriately. That means all the inputs were connected to different voltage sources and outputs were connected to the load capacitor. And the circuit was connected to VDD and GND to complete the circuit. Here, all the signals and VDD are taken as 1.8V.

The simulation aim was to get the output that follows the truth table of a 1-bt Full Adder. Also, it was required to calculate the propagation and contamination delay for each combination of input. Generally, there are $8(2^3 = 8)$ possible combination of inputs as given in the table 1 below.

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1: Full adder truth table

However, if constant voltage is given to all the inputs, there will not be a rising or falling transition of output. Thus, we cannot find the delay from that approach. To get the transition, at least one of the input has to be in the form of pulse. In such approach, only 4 combinations of inputs will fulfill

all the condition of inputs in above table. But, in case of delay, it will not be sufficient conditions. We need to add more cases to observe the delay to find out, in which case the delay in maximum and minimum.

To do that, firstly, one of the input is given pulsed input and the rest are given constant voltages, then the second input is given pulsed input and constant voltage for the rest. After that, 2 of the inputs are given pulsed input and then all the inputs are given pulsed input. By doing that, we can see delay in all the cases. And compare the results. Fig 4. in the test bench shows B input is given pulsed voltage with 10ps of rise time and fall time.

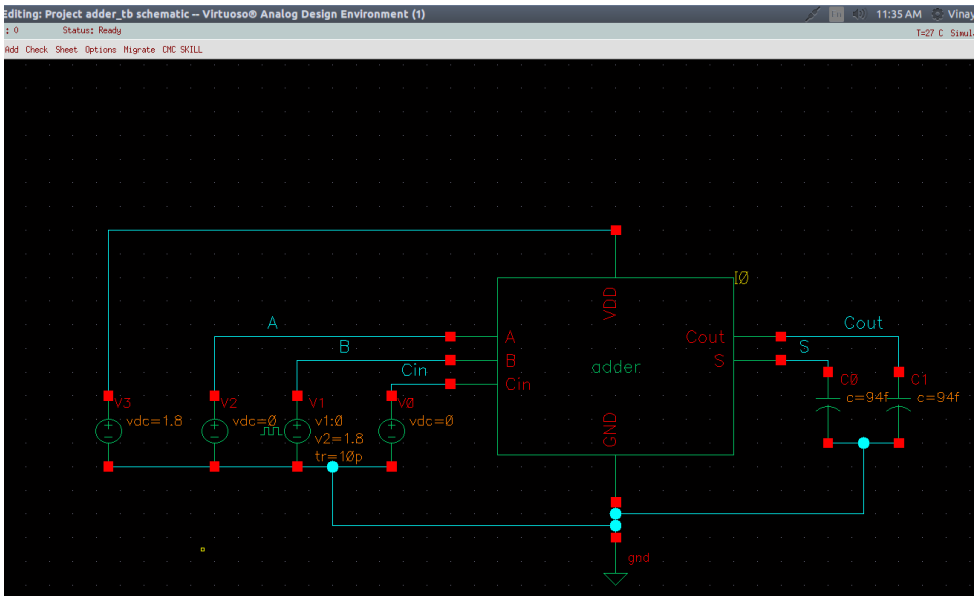


Fig.4: Pre-layout simulation test bench

Following this procedure, we can give 4(2^2=4) possible constant voltages to A and Cin. The waveform for this case will look like in the fig 5 below.

It can be observed from the waveform that there is no propagation of Cout, so there is no delay of Cout in this specific case.

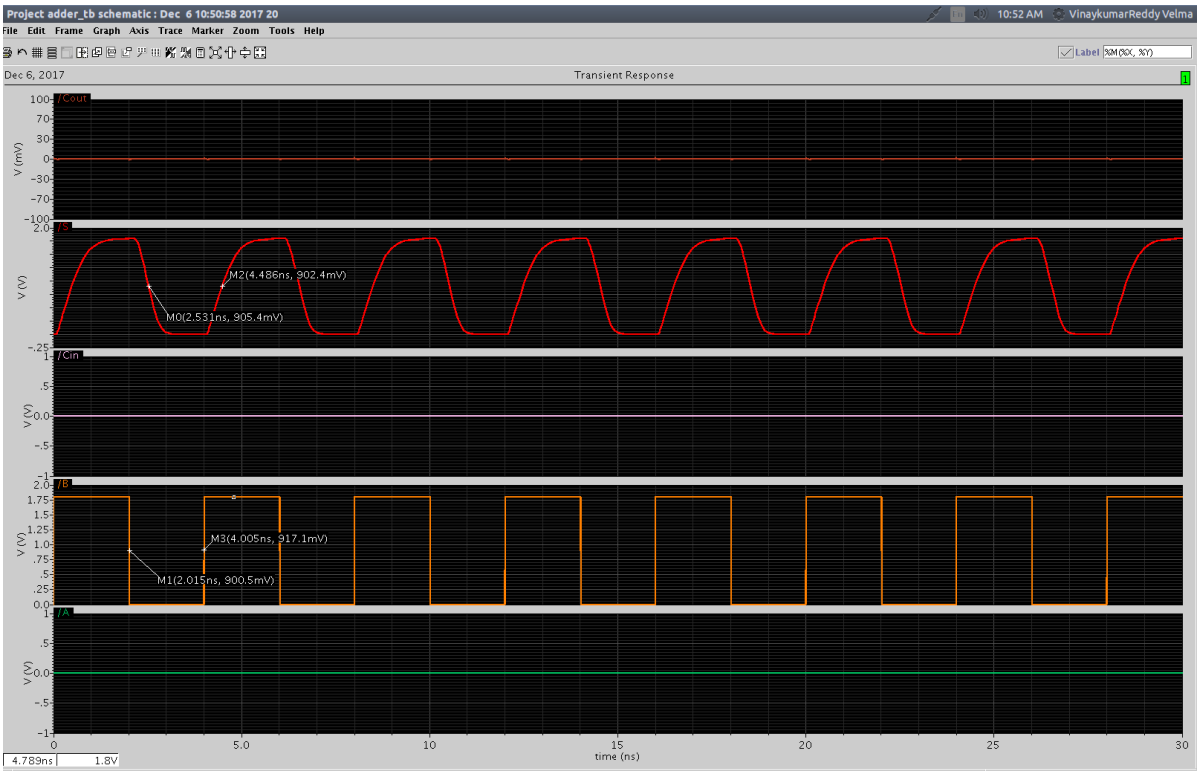


Fig.5: B= Vpulse, A=GND, Cin=GND waveform

Here, this checks 2 condition at the same time namely, **000** & **010**. And since there is a pulsed input, we can get the pulse at the output and we can find propagation delay from that pulse. It is to be noted that, in case **010,** Sum= 1, Cout= 0. So we will not get any transition for the Sum but not Cout. Such cases with no pulse transition for S and Cout are omitted from the tables.

And following this procedure, we can not only cover all cases of inputs but also find delay in all the conditions and compare them. Here, instead of inserting graphs for all the cases, the data is represented in the form of tables. The separate tables of Sum and Cout is shown below. Table 2 shows the delays of S with different combination of inputs while Table 3 shows the same thing for Cout. It is to be noted that in some cases, there we no pulse observed which is why the number of cases in both Sum and Cout may vary. All the time mentioned in the tables is in nano second.

A	B	Cin	τ_{pdf}(ns)	τ_{pdr}(ns)	τ_{pd}(ns)=(τ_{pdf} + τ_{pdr})/2
Vpulse	GND	GND	0.543	0.495	0.548
Vpulse	GND	1.8V	0.492	0.616	0.399
Vpulse	1.8V	GND	0.529	0.613	0.431
Vpulse	1.8V	1.8V	0.413	0.515	0.401
GND	Vpulse	GND	0.516	0.481	0.536
GND	Vpulse	1.8V	0.592	0.452	0.655
1.8V	Vpulse	GND	0.569	0.632	0.450
1.8V	Vpulse	1.8V	0.432	0.525	0.411

GND	GND	Vpulse	0.480	0.463	0.518
GND	1.8V	Vpulse	0.446	0.579	0.385
1.8V	GND	Vpulse	0.493	0.615	0.401
1.8V	1.8V	Vpulse	0.449	0.541	0.415
Vpulse	Vpulse	Vpulse	0.463	0.498	0.465

Table 2: Propagation delay for Sum

A	B	Cin	τ_{pdf}(ns)	τ_{pdr}(ns)	τ_{pd}(ns)=(τ_{pdf} + τ_{pdr})/2
Vpulse	GND	1.8V	0.499	0.470	0.485
Vpulse	1.8V	GND	0.442	0.496	0.469
GND	Vpulse	1.8V	0.446	0.482	0.464
1.8V	Vpulse	GND	0.457	0.507	0.482
GND	1.8V	Vpulse	0.418	0.473	0.446
1.8V	GND	V pulse	0.465	0.500	0.483
Vpulse	Vpulse	Vpulse	0.338	0.433	0.386
Vpulse	Vpulse	1.8V	0.438	0.491	0.46
Vpulse	Vpulse	GND	0.350	0.501	0.426
Vpulse	GND	Vpulse	0.405	0.492	0.448
Vpulse	1.8V	Vpulse	0.404	0.487	0.445
GND	Vpulse	Vpulse	0.348	0.426	0.387
1.8V	Vpulse	Vpulse	0.373	0.446	0.41

Table 3: Propagation delay for Cout

After the simulation is done, the next step is to prepare the layout.

6. THE LAYOUT AND EXTRACTED VIEWS OF FULL ADDER

The schematic of the adder is a little complex compared to the NOR or NAND logic gates. But an easy approach to do the layout is to take reference of the stick diagram. It can reduce the complexity and provide an efficient way to carry out layout.

The reference schematic is in fig. 2 and it will be difficult to represent the stick diagram for whole schematic, in only 1 figure. Which is why, the schematic is divided into 6 parts out of which, 2 are inverters to invert the Sum and Cout.

The stick diagram for different segments are drawn in fig. 6 The diagram follows the stick diagram rules that metals should be shown in blue color, poly is to be represented in red, N diffusion region should be shown in light green and P diffusion should be shown in yellow color.

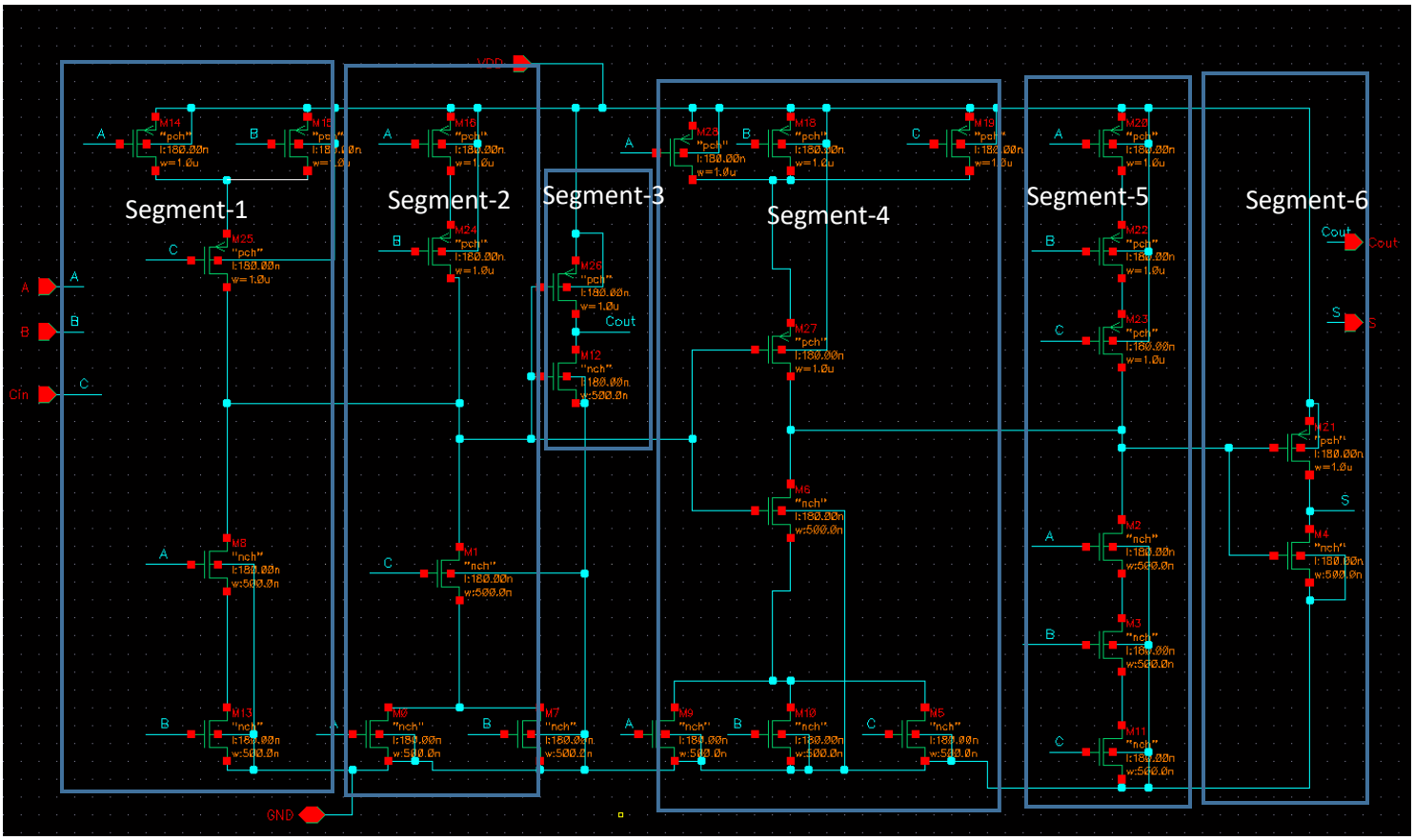


Fig. 6: Segmentation of Schematic

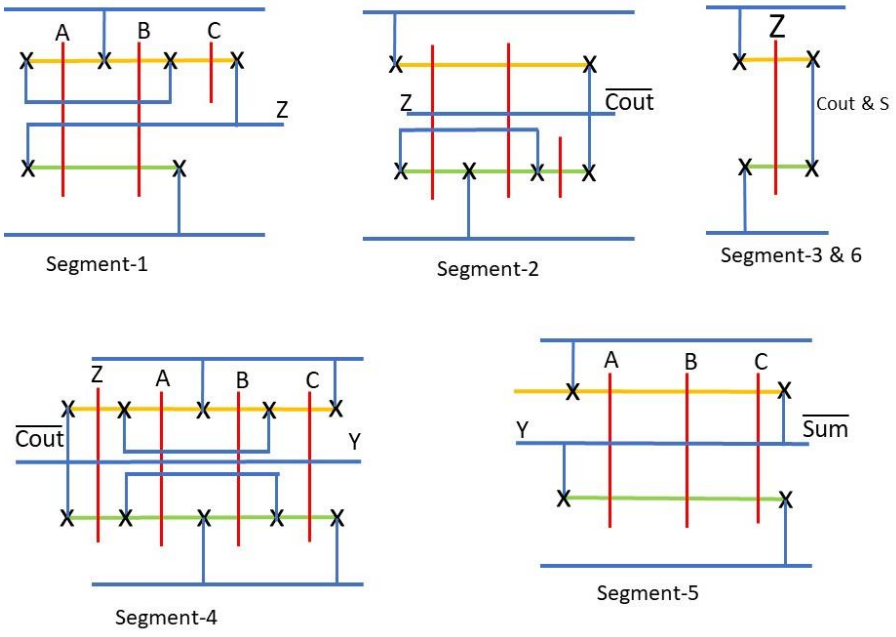


Fig.7: Stick diagram of different segments

The transistors are generally defined by at least four physical masks. They are active (also called diffusion, diff, OD or RX), n-select (also called n- implant, or n-plus), p-select (also called p-implant, or p-plus), and polysilicon (also called poly, poly g, PO, or PC). The active mask defines all the areas where either n- or p- type diffusion is to be placed or where the gates of transistors are to be placed. The overlap area of the poly and active area defines the gates of transistors. select surrounds active regions where n type diffusion is required. P-select surrounds active regions where p type diffusion is required. Further, to form PMOS, or sometimes even to form NMOS, n-well or p-well are required, which should be larger than the active regions. To connect the body to the corresponding voltage level, n+ or p+ region should be placed in the body.

After placing appropriate metals, contacts, pins and all regions, the layout will look like in fig. 8.

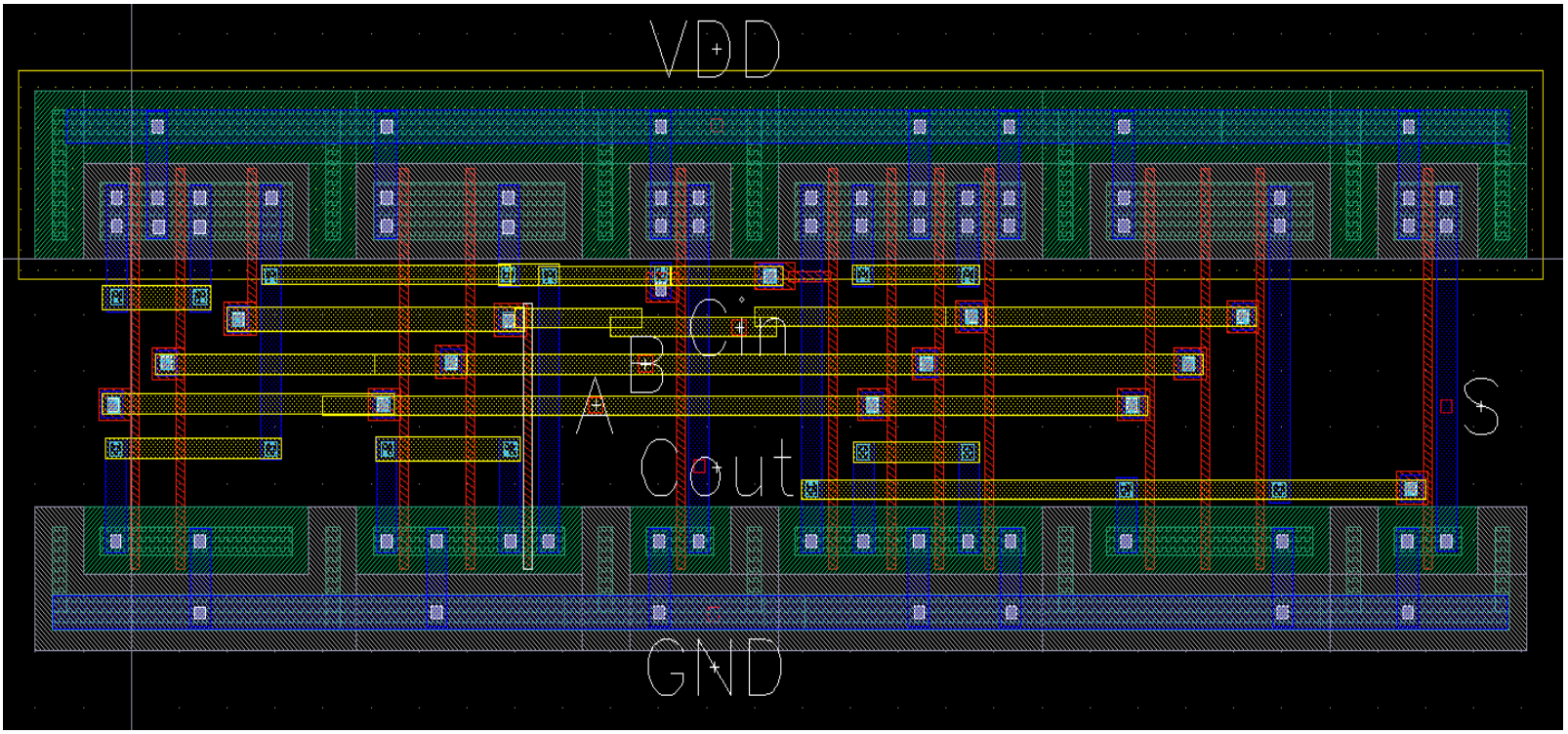


Fig. 8: Layout of Full Adder

After finishing the layout, the next step is to check if the design is following the DRC rules or not. These rules are defined by the foundry. If there is any violation of DRC rules, it will show which rule is being violated.

5. DRC Results

These rules are defined by the foundry. If there is any violation of DRC rules, it will show which rule is being violated. The design rule verification step checks that all polygons and layers from the layout database meet all of the manufacturing process rules. A complete set of design rules for TSMC 180 nm PDK can be found in "/CMC/kits/cmosp18/doc".
Secondly, a successful DRC check does not mean that the designed layout represents the drawn schematic. It just checks that all the rules of the design are being followed. And does not compare it with the schematic.

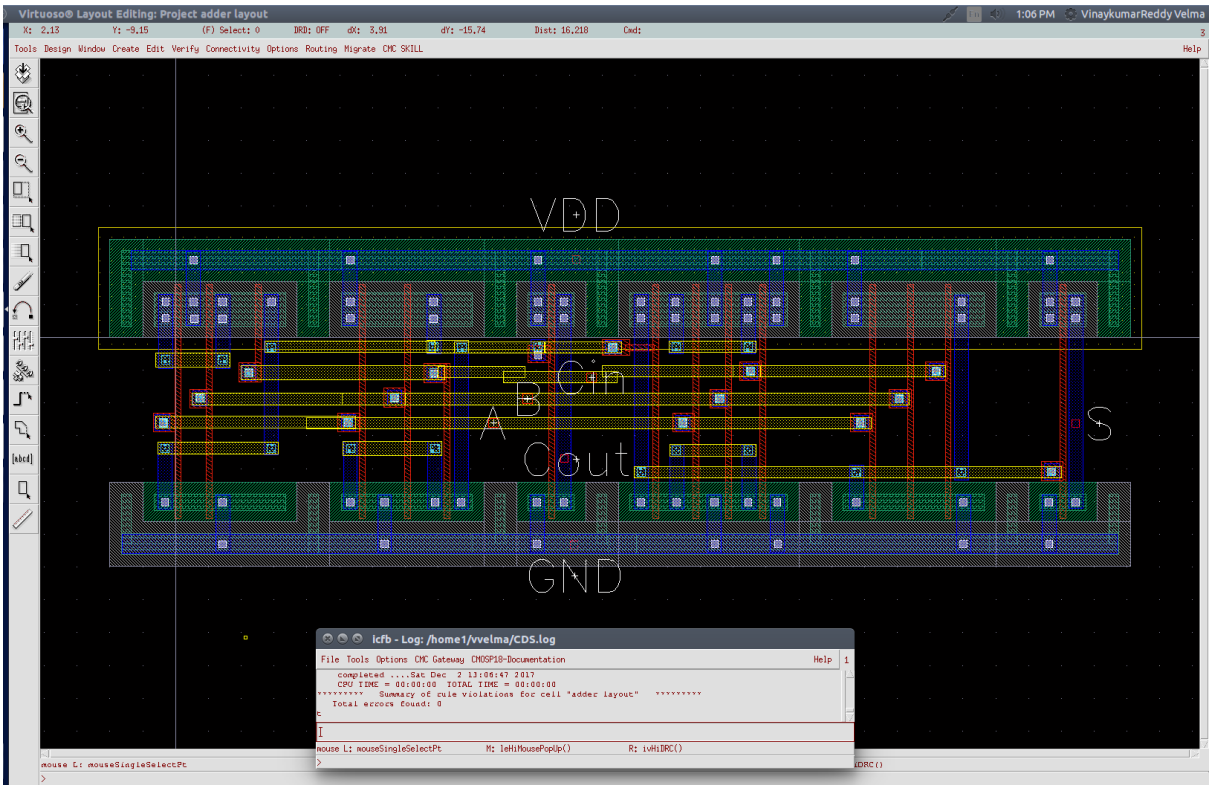


Fig. 9: DRC result

In addition to that, as per the given parameters, the circuit area should not increase 500µm^2. And since we are using the 180nm process area cannot be found with the help of stick diagram. So measuring the boundaries of the layout, we can find out the total area which is denoted in the fig. 10 below.

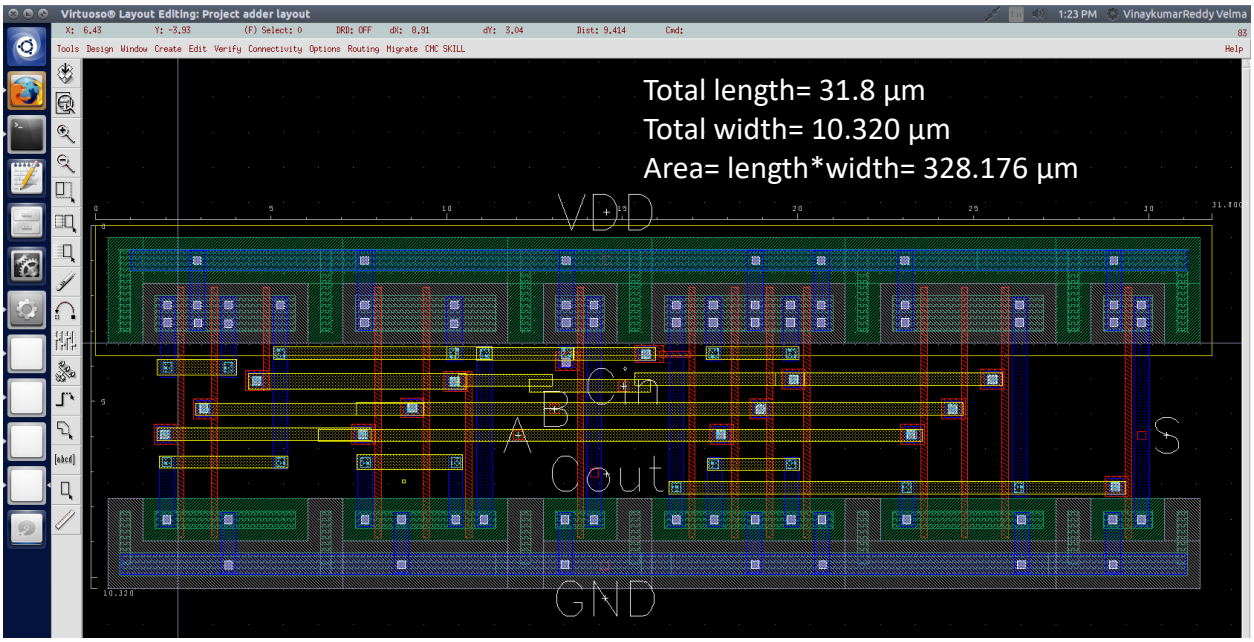


Fig. 10: Circuit area calculation of layout

Extraction

This process will identify the drawn transistors, metals, contacts and pins from layout. In addition to that it will identify the hidden capacitors from the layout as well which causes the propagation delay. If the DRC rules are not followed, the software will not extract the file with an error of violation of DRC rules. So it can be done only after a successful DRC. This process also generates a netlist of data for the layout which can be compared with the schematic while doing the LVS.

Extraction is a hand-off step back to the circuit designer, who will simulate the circuit based upon the extraction file. Sometimes, there are specific requirements from the circuit designers to the layout engineer. For example, the circuit designer may be more interested in the parasitic capacitance due to the wire coupling. The layout engineer should switch the parasitic resistance before the extraction. Feedback will be given by the circuit designer after the post-layout simulation to the layout engineer if there is a need of modification for the layout. As you can see, the extracted netlist is a good vehicle that link the design engineer and layout engineer.

The extracted view of the layout can be seen as in fig. 11.

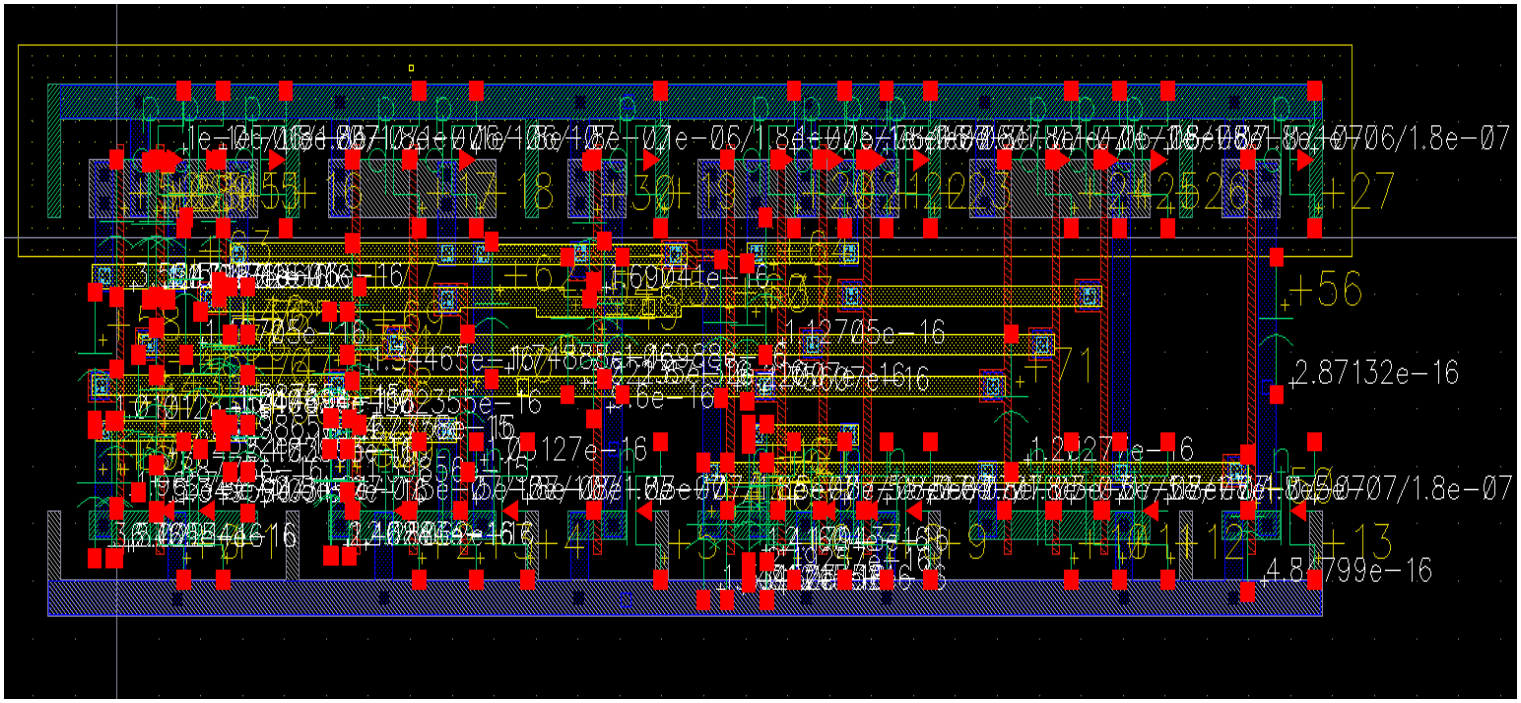


Fig. 11: Extracted view

7. LVS Results

LVS verification is checking that the design is connected correctly. The schematic is the reference circuit and the layout (extracted netlist) is checked against it. The following are verified:

- Electrical connectivity of all signals, including input, output, and power signals to their corresponding devices.
- Device sizes: transistor width and length, resistor sizes, capacitor sizes. Identification of extra components and signals that have not been included in the schematic; floating nodes would be an example of this.

A successful LVS shows that the drawn layout matches with the schematic and widths of all transistors, contacts are same in schematic and layout. The log files are created after the LVS completes. This log file shows that how many nodes, transistors and contacts are there in schematic and extracted. Even if the LVS fails, the log shows where and how many mismatches are there in the layout as compared to the schematic.

This result shows that the layout which is designed represents the desired schematic.

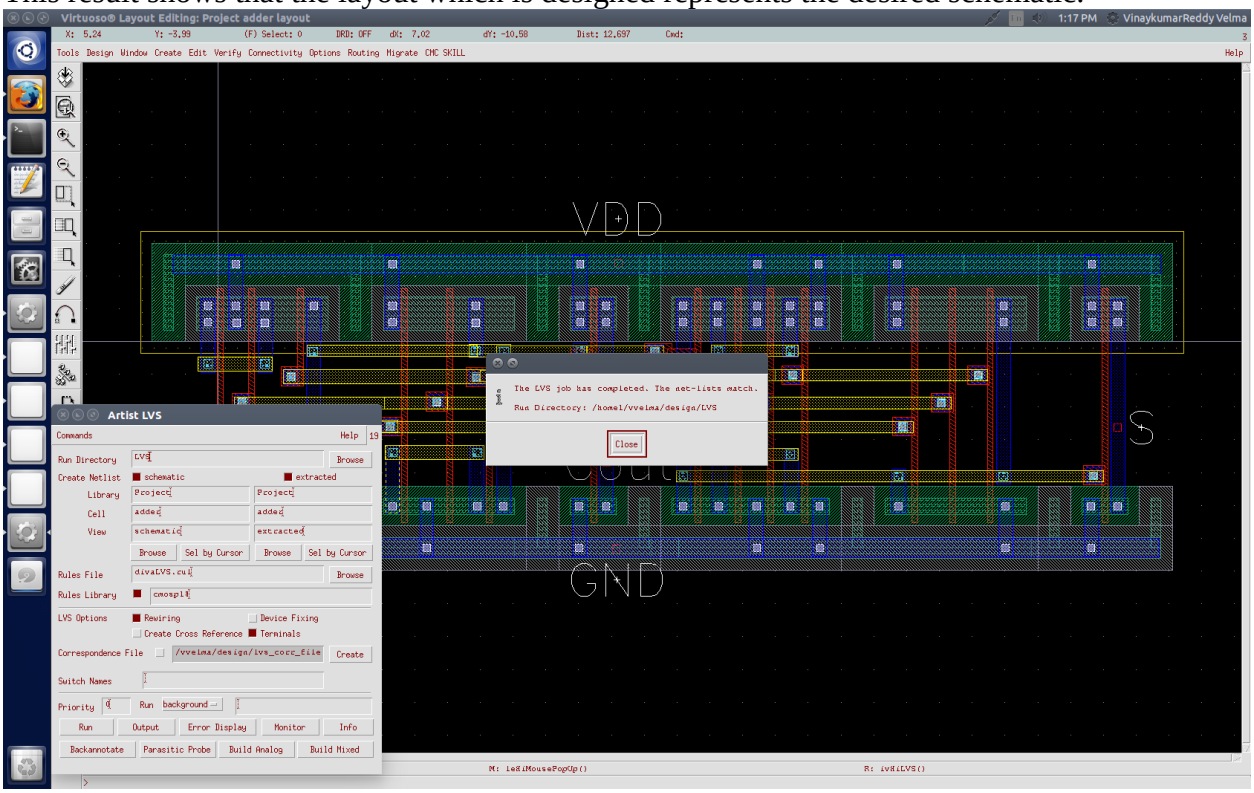


Fig. 12: LVS Result

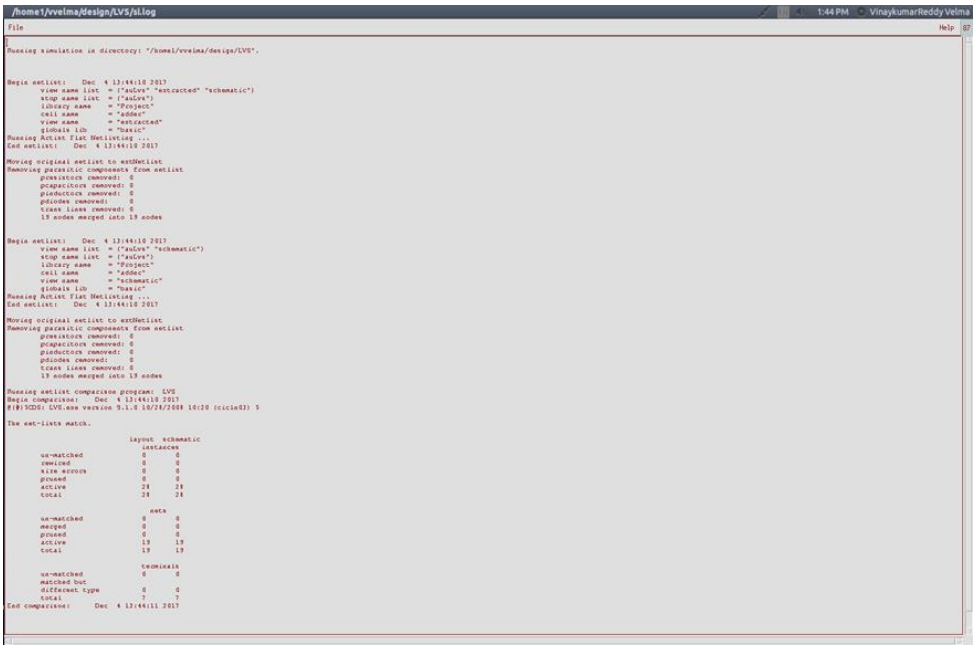


Fig.13: Si log file

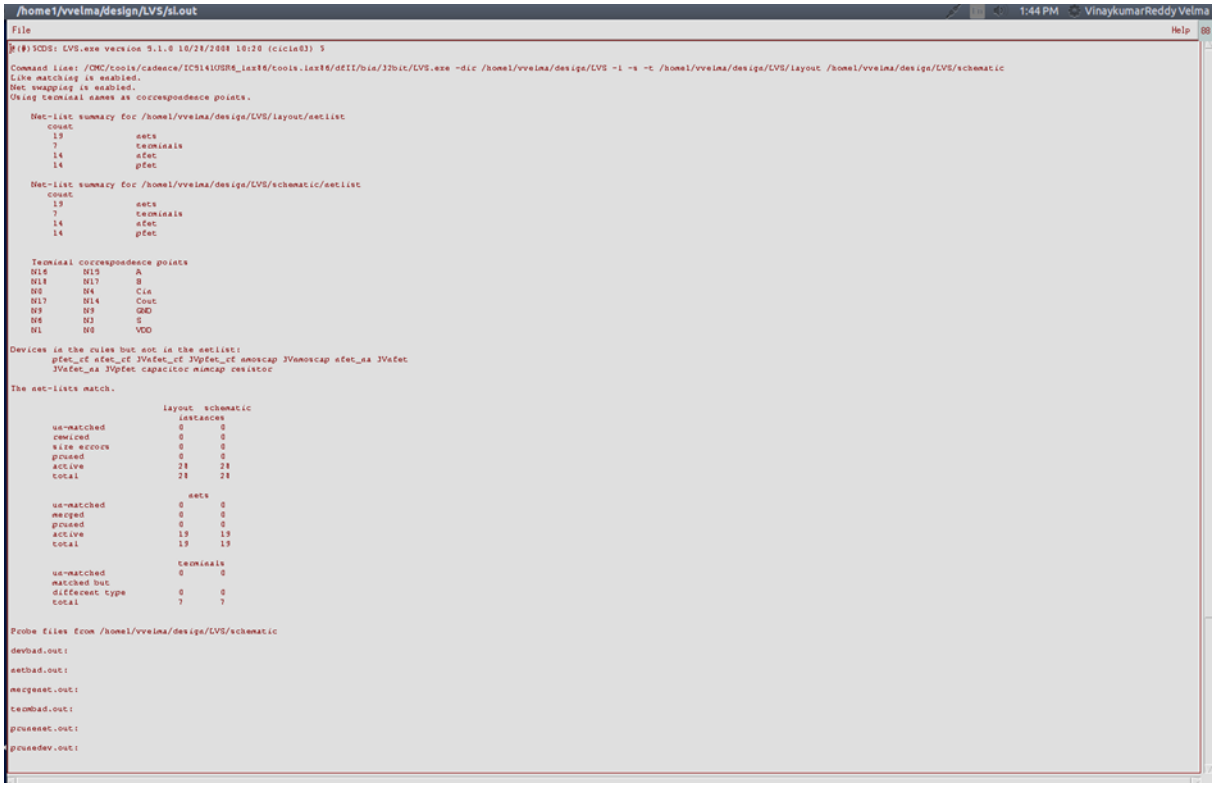


Fig. 14: Si out file

8. POST LAYOUT SIMULATION

Once the LVS is done, next step is to verify that the extracted circuit gives the same result as the schematic. It is the last and important step in a design.

The first step in this is to make a test bench just like the pre-layout simulation but this time, 2 symbols are taken, and they are given same inputs and their outputs are taken through the load capacitor.

The next step is to use the hierarchy editor tool of the cadence to make the config window as in fig. 15. Here, one of the symbol is assigned to the schematic circuit and the other symbol is assigned as the extracted. Thus, this two symbols represent the same thing but they are not entirely the same this. The aim here, is to verify if both the results are giving the same outputs with similar delays or not.

And the procedure to get the output waveform remains the same as done during the pre-layout simulation.

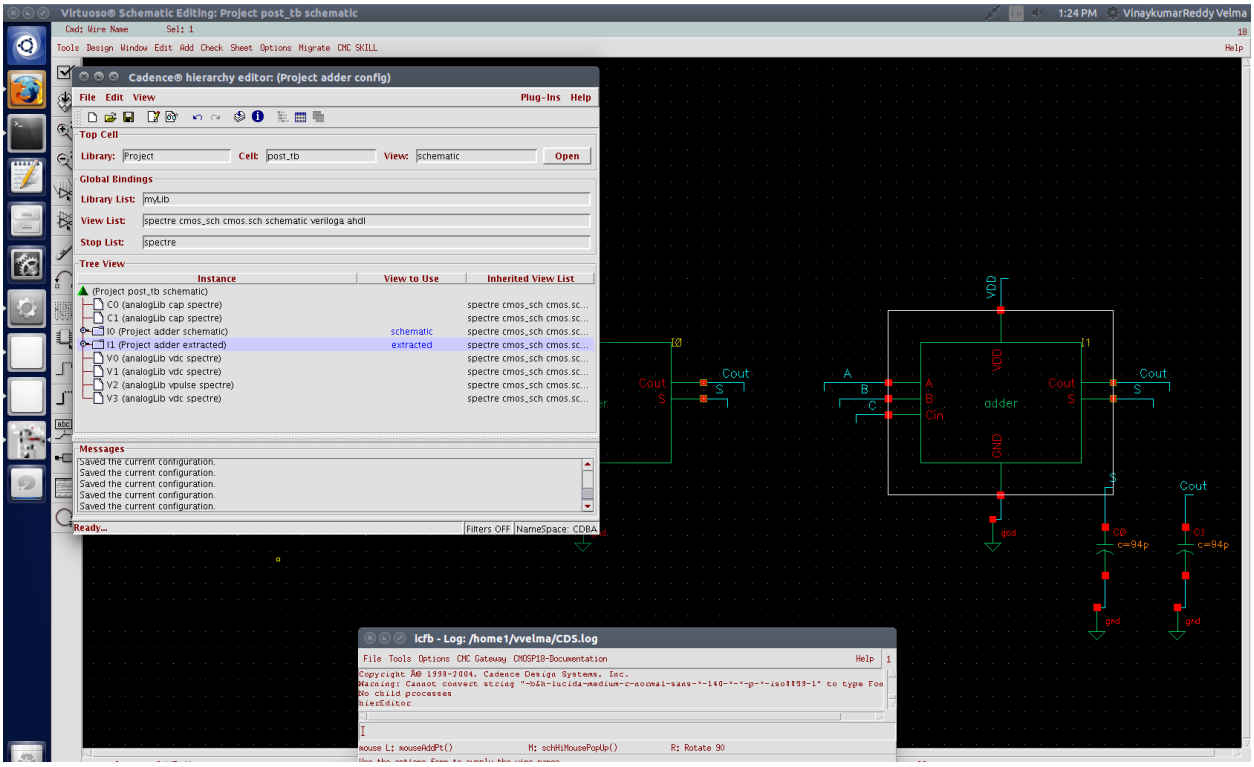


Fig.15: Config window

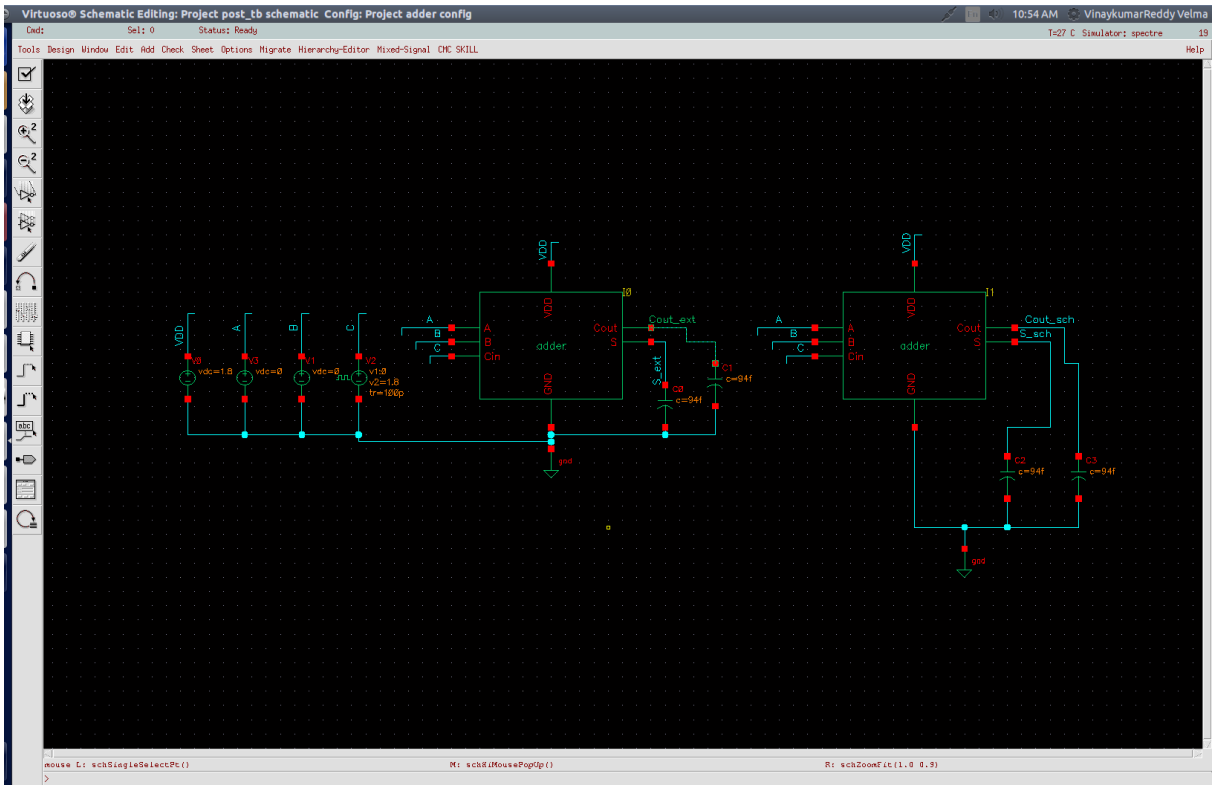


Fig.16: Post layout test bench

In the post layout simulation, the obtained waveform for one of the case looks like in fig. 16. In the figure, there are 2 waveforms of Sum and 2 for Cout. One of them represents the waveform of Sum generated by the schematic and the other from the extracted symbol. And same goes for carry. The results can be shown in the form of table. Here, with the help of markers, the points to find the propagation delay can be located and we can find the delay for all cases. Table 4 shows the delays in all cases for Sum and Cout.

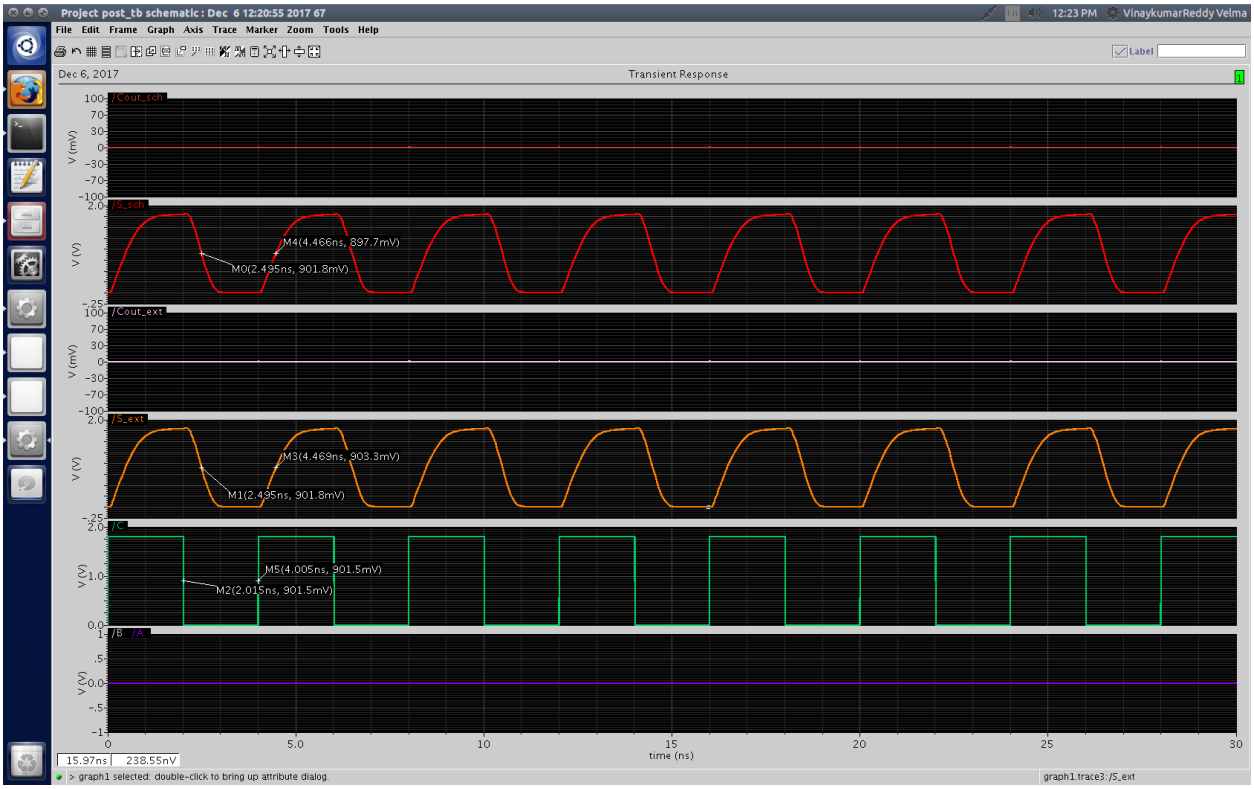


Fig.17: Waveform of a post-layout simulation for A=GND, B= GND and C=Vpulse

			Extracted			**Schematic**		
A	B	Cin	τ_{pdr}(ns)	τ_{pdf}(ns)	τ_{pd}(ns)	τ_{pdr}(ns)	τ_{pdf}(ns)	τ_{pd}(ns)
Vpulse	GND	GND	0.493	0.543	0.518	0.496	0.543	0.520
Vpulse	GND	1.8V	0.615	0.491	0.553	0.616	0.491	0.554
Vpulse	1.8V	GND	0.614	0.530	0.572	0.614	0.530	0.572
Vpulse	1.8V	1.8V	0.517	0.414	0.466	0.514	0.414	0.464
GND	Vpulse	GND	0.480	0.516	0.498	0.480	0.566	0.523
GND	Vpulse	1.8V	0.452	0.594	0.523	0.593	0.470	0.532
1.8V	Vpulse	GND	0.628	0.544	0.586	0.630	0.544	0.587
1.8V	Vpulse	1.8V	0.524	0.432	0.478	0.524	0.432	0.478
GND	GND	Vpulse	0.464	0.480	0.472	0.461	0.480	0.471
GND	1.8V	Vpulse	0.580	0.447	0.514	0.578	0.445	0.512
1.8V	GND	Vpulse	0.579	0.530	0.555	0.614	0.530	0.572
1.8V	1.8V	Vpulse	0.541	0.449	0.495	0.524	0.432	0.478
Vpulse	Vpulse	Vpulse	0.496	0.464	0.480	0.496	0.464	0.480

Table 4: Delay of Sum for extracted and schematic view

			Schematic			Extracted		
A	B	Cin	τ_{pdr}(ns)	τ_{pdf}(ns)	τ_{pd}(ns)	τ_{pdr}(ns)	τ_{pdf}(ns)	τ_{pd}(ns)
Vpulse	GND	1.8V	0.499	0.468	0.484	0.497	0.468	0.483
Vpulse	1.8V	GND	0.496	0.443	0.470	0.494	0.443	0.469
GND	Vpulse	1.8V	0.478	0.446	0.462	0.478	0.446	0.462
1.8V	Vpulse	GND	0.506	0.461	0.484	0.508	0.466	0.487
GND	1.8V	Vpulse	0.472	0.419	0.446	0.460	0.419	0.440
1.8V	GND	Vpulse	0.496	0.443	0.470	0.494	0.443	0.469
Vpulse	Vpulse	Vpulse	0.430	0.340	0.385	0.430	0.340	0.385
Vpulse	Vpulse	GND	0.352	0.503	0.427	0.35	0.51	0.43
Vpulse	GND	Vpulse	0.405	0.494	0.449	0.408	0.493	0.45
Vpulse	1.8V	Vpulse	0.403	0.487	0.445	0.403	0.488	0.445
1.8V	Vpulse	Vpulse	0.375	0.446	0.41	0.378	0.447	0.412
Vpulse	Vpulse	1.8V	0.44	0.489	0.464	0.443	0.467	0.455
GND	Vpulse	Vpulse	0.345	0.426	0.385	0.348	0.43	0.389

Table 5: delay of Cout for Schematic and extracted view

9. Conclusion

It can be seen from both the tables that the propagation delay in pre-layout and post-layout simulation are almost the same. The minor difference can be because of the parasitic that might be introduced during the layout. And the delays in the schematic for pre-layout and post-layout are matching with each other with minor deviations.

a. Contamination Delay

From the above Table 5 we can say that the best-case delay for Sum is when Cin is Vpulse and A and B are GND. In this case, the delay can be called as contamination delay. And the least delay in case of Cout is when, all the inputs are Vpulse. And the delay in that case is 0.385ns. That is the contamination delay for Cout.

b. Propagation Delay

The worst-case delay is when A is Vpulse, B is 1.8V (Constant) and C is GND. The delay in this case is 0.587ns and it can be called as propagation delay for Sum. And for Cout, the worst-case delay is when A is Vpulse, B is GND and Cin is 1.8V(Constant). The delay in this case is 0.483ns. So it the propagation delay for Cout.

10. References

[1] Neil H. Weste and David Money Harris. “CMOS VLSI Design- A circuit and system perspective”. Fourth Edition, 2010
[2] A. Kabbani and T. Khan, _ELE754 Lab Manual," 2009
[3] Smith, Michael John Sebastian. “Application-Specific Integrated Circuits”. Vol. 7. Reading, MA: Addison-Wesley, 1997.
[4] Zhuang, Nan, and Haomin Wu. "A new design of the CMOS full adder." IEEE journal of solid-state circuits 27.5 (1992): 840-844.
[5] Manjunath K M, Abdul Lateef Haroon P S, Amarappa Pagi, Ulaganathan J. “Analysis of various full-adder circuits in Cadence” National conference on power system & industrial automation (NCPSIA 2015).