VLSI Design Project

Design and Analysis of a 4x4 multiplier

Tejas Srivastava | 2021102017

1.) **Introduction**

The project mainly deals with design and performance analysis of a simple 4x4 multiplier circuit. The circuit is simulated in ngspice and power and time-delay analysis is done for pre-layout netlist as well as post-layout netlist. Layout is done in *magic* and the verification of the logic of the code is done in *verilog.* The pre-layout and post-layout results are compared and conclusions are drawn about the ideal and real life circuits found ICs present today.

2.) **Method/Approach**

For writing the netlist, extensive use of *subckt* is done to abstract individual building blocks of the circuit for the readability and robustness of the final multiplier. Firstly, subcircuits for basic gates like NAND, NOT, AND, OR are made and their functionality is verified. Next, using these basic gates, the building blocks of the 4x4 multiplier such as *half-adder, full-adder, 4-bit adder* are made and the final multiplier is just a simple combination of these blocks. The gate level circuit used for 4x4 multipllier is as shown in the figure :-

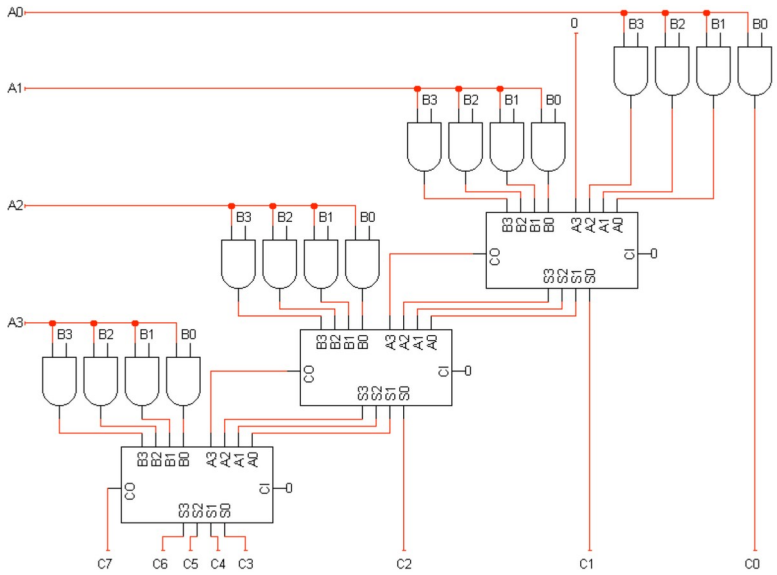


Figure 1 : Circuit Used for Multiplication

3.) **Multiplication Logic**

The basic idea is that we can break NxN multiplication into simpler order multiplication and get the result by proper addition of the obtained results. The circuit shown above is based on the ***add and shift*** method that we follow when we multiply two 4-bit numbers by hand. We know that 1 bit multiplication is simple an AND operation on the 2 bits and thus, AND gates produce partial products.

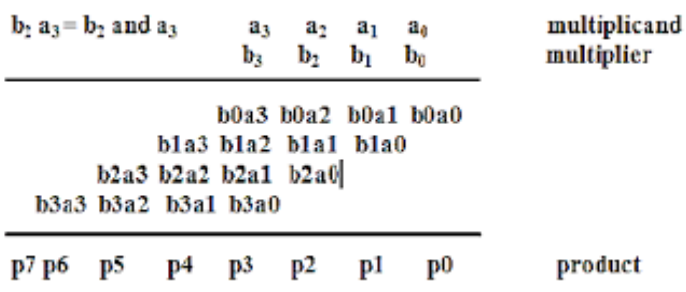


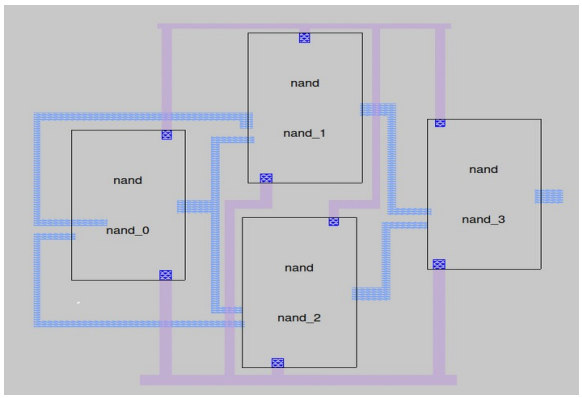
Figure 2 : Add and shift method

Combinations of these partial products when added properly give the multiplier result. This is why the above circuit contains only 4-bit adders for addition and AND gates for partial product. The algorithm is explained as follows :-

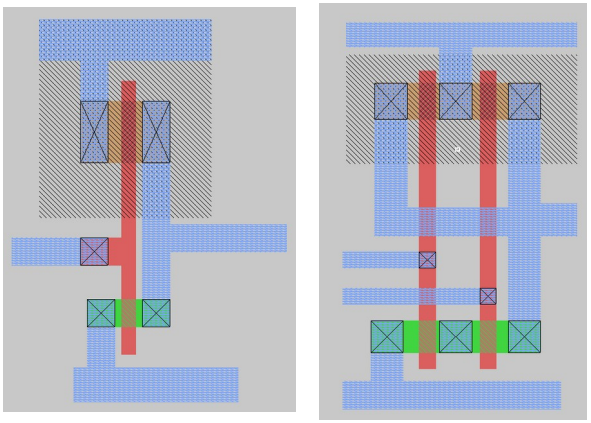
4.) **Layout in Magic**

The layout for the same circuit is done in magic. Extensive use of *getcell* is done to do as much abstrction as possible. For example, one cell is made for each of the basec gates used to build adder circuits, which then are used to make the multiplier circuit. The only 2 basic gates made without using another cell are NAND and NOT. The gates for OR, AND, XOR etc are then made from these and circuits for adders are then made from these gates.

For Example, XOR gate is made as a combination of 4 nand gates as shown below



XOR gate made from 4 NAND gates



(a) NOT gate (b) NAND gate

5.) **Time Delays : Worst Case**

Time delays are calculated using .measure command in ngspice with trig and targ given appropriately.

Time delays at each output node are calculated for about 50 possible inputs and the worst case delay is noted down for each input sequence. The delays are combined in a *delays.txt* file. The highest delay among all of them is found out to be

$$td\_6\_hl = 5.079905e\text{-}11$$

With close approximation, this can be said as the worst case delay, which gives us a minimum clock frequency of $f = 1.96 \times 10^{10}$ *Hz*

Note: *In the delays.txt file, for each input combination, it is not necessary that output change occurs in all the output nodes, which is why not all output nodes are present in each input combination.*

6.) **Power Analysis**

For doing power analysis, we attach a 0V DC source at each of the output nodes of the multiplier, and measure current flowing through the source by plotting it. We observe a small value of leakage current even the output is OFF. The average (approximate) of this small current is multiplied by the Vdd value to get the power of the circuit.

For measuring power at each node, we give such an input sequence which after 1 time period, causes change in all of the output nodes. That is if input sequence changes from (A0A1A2A3, B0B1B2B3) to (a0a1a2a3, b0b1b2b3), then the output changes from out to out'. *(where c' denotes 1s complement of c)*

One such input sequence is found out to be : -

State 1 : 1001 x 1011 = 01100011 = 99 = c
State 2 : 1101 x 1100 = 11011100 = 156 = c'

Power is measured this way for both pre-layout and post-layout circuit and comparision is made. The observation is noted in the table below :-

Node	Pre-Layout	Post-Layout
P0	57nW	1300nW
P1	102.5nW	200nW
P2	100nW	100nW
P3	105nW	72nW
P4	57nW	250nW
P5	60nW	210nW
P6	51nW	75nW
P7	62nW	350nW

Observation : It is observed clearly that more power is required by the post-layout circuit rather than the pre layout ciruit. The reason for this is that when we do layout in magic, we connect all the parasatic capacitances to the mosfets. The power/energy used by them is not reflected in the ideal (pre-layout) circuit.

7.) **Time Delay: Pre-Layout vs Post-Layout**

Similar to the previous input sequenct, if we give the sam input sequence for delay analysis, we get delay values for each of the output node. (since a transition occurs in each of the output bit for c -> c'). Again the state transition (or input sequences) used here are :-

***State 1** : 1001 x 1011 = 01100011 = 99 = c*
***State 2** : 1101 x 1100 = 11011100 = 156 = c'*

The following table tabulates the observed time delay values: -

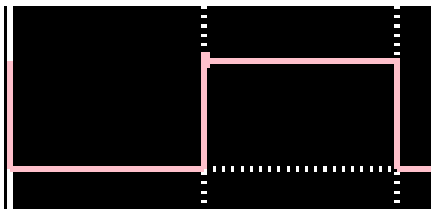
Delays Pre Layout :-

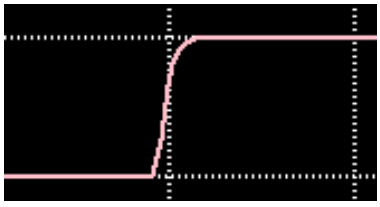
Node	td_{LH} (in sec)	td_{HL}(in sec)
P0	3.60e-12	3.62e-12
P1	1.18e-11	9.11e-12
P2	1.18e-11	9.11e-12
P3	3.81e-11	2.46e-11
P4	3.72e-11	4.35e-11
P5	2.12e-11	2.99e-11
P6	2.49e-11	4.63e-11
P7	2.39e-11	2.38e-11

Delays Post Layout :-

Node	td_{LH} (in sec)	td_{HL}(in sec)
P0	1.17e-9	1.30e-9
P1	3.36e-9	3.06e-9
P2	2.63e-9	3.04e-9
P3	9.31e-9	1.08e-8
P4	8.01e-9	8.13e-10
P5	7.42e-9	1.61e-8
P6	1.10e-9	2.11e-8
P7	7.88e-9	2.21e-8

Observation : It can be clearly seen that the time delay values for Post Layout circuit are quite larger that those in the pre layout table. This is simply due to the fact that when we start considering all the capacitances for out netlist, their charging/discharging time also gets included in the time delay values and thus we see a marked increase in delays.

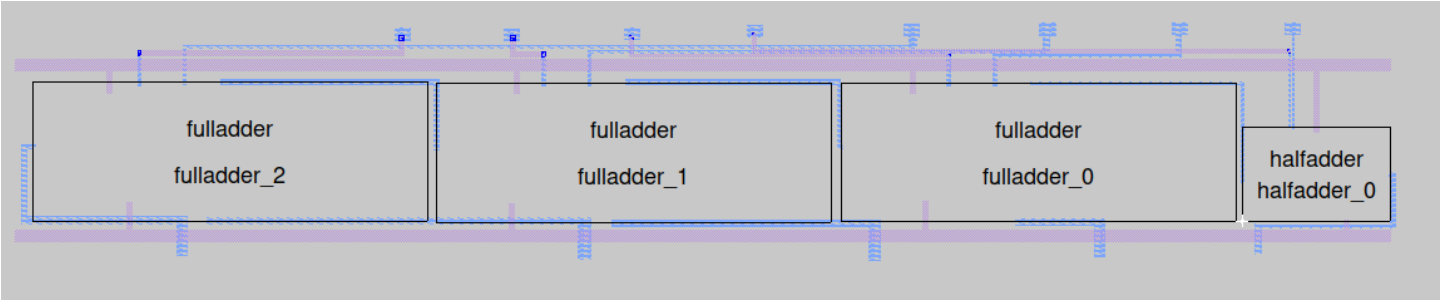




Typical Transition : Pre-Layout (no capacitances considered) Post Layout (all capacitances considered)

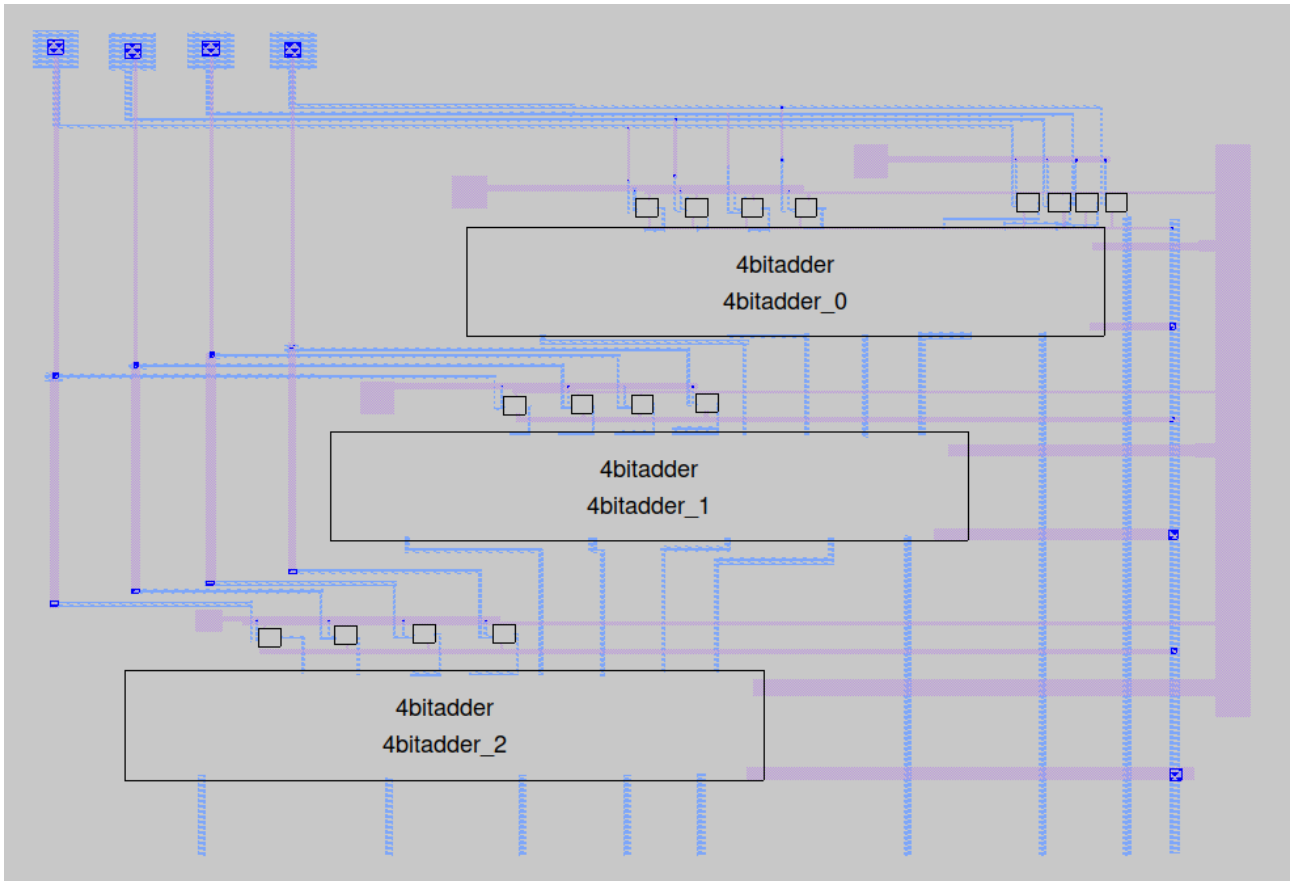
8.) **Final Layout**

Images of 4 bit adder and final layout are shown as below :-

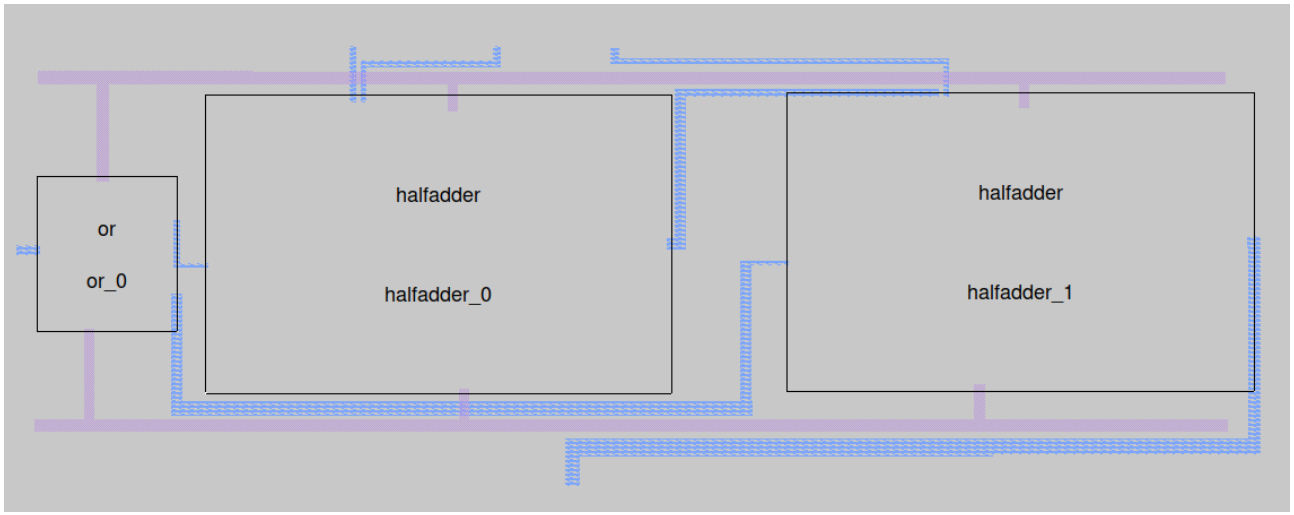


4-bit adder in magic

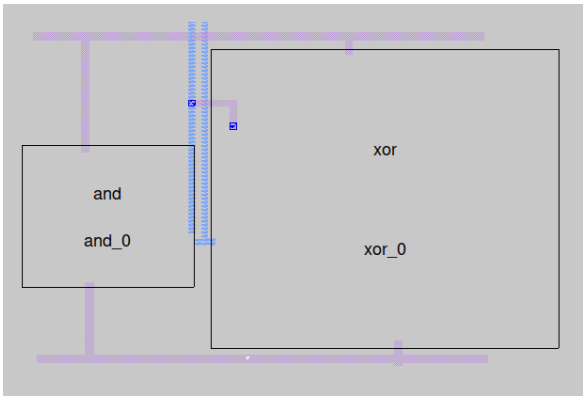
The following is the layout of the final 4x4 multiplier circuit :-



4x4 Multiplier in Magic



Full Adder in magic

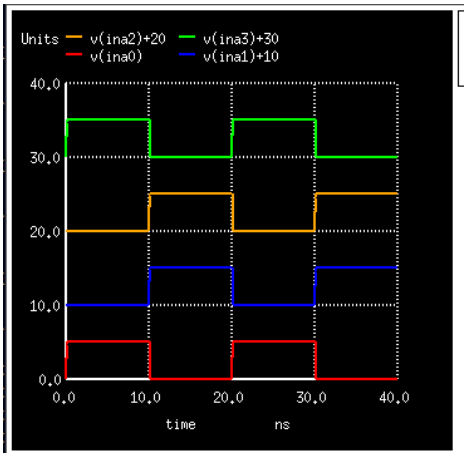


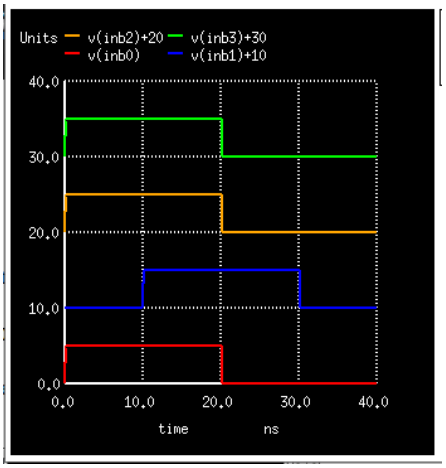
Half Adder in magic

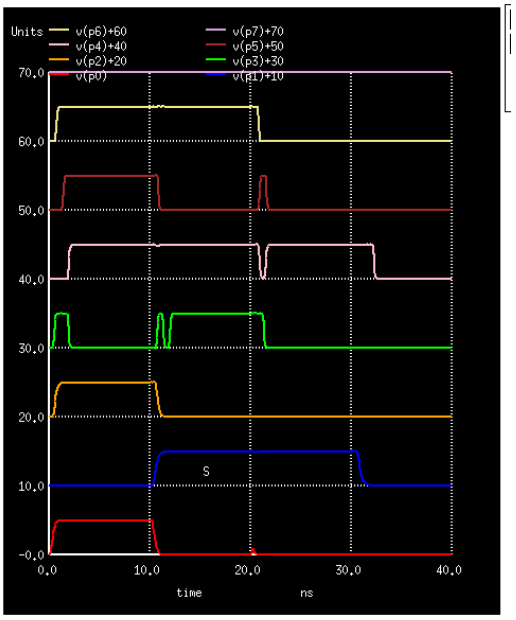
Where further gate level layout has been previously shown in section 4.

9.) **Graphs of output/input**

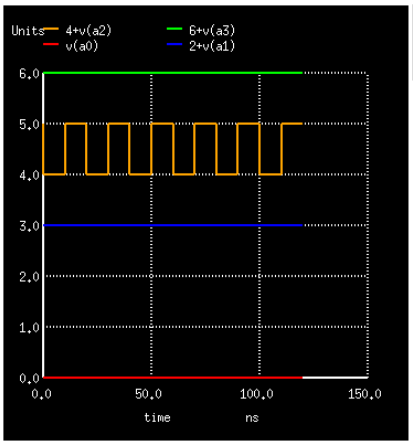
The following graphs depict the variation of transient output vs transient input for one test input :-

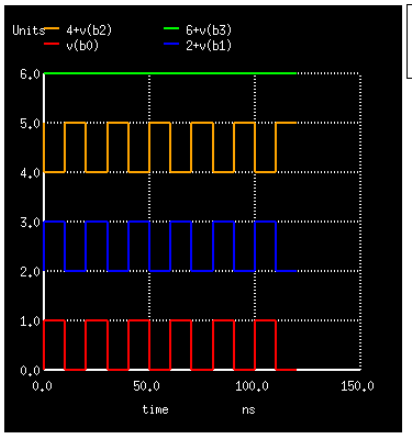


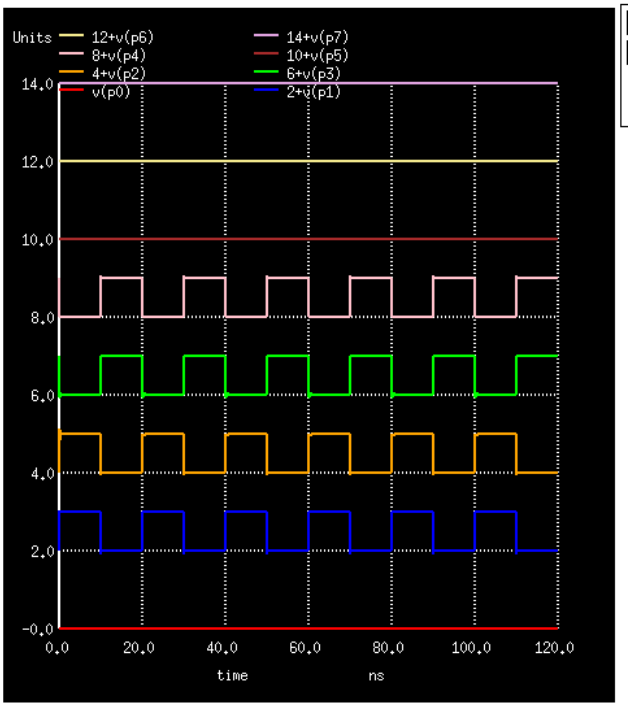




Input Output Graphs of Post Layout Circuit







Input Output Graphs of Post Layout Circuit

Note : run caller.spice to run simulations for ngspice (pre layout) and run multiplier_checker.spice in magic folder to run simulations for post layout circuit