Full Adder Circuit Optimization

ECE 09414 - 2 VLSI

Dr. Hanafi

Adam Fifth

Everett Jones

12/18/17

I. Abstract

The design of an optimized Full Adder circuit was sought after as a means to create a circuit that met the criteria of a Full Adder while also having the shortest delay possible. Through the use of 600 nm CMOS technology an optimized design for a Full Adder circuit was created. This was done by calculating delay times across multiple circuit designs before selecting the design with the smallest calculated delay. Each component was then sized and designed to be assembled into a functioning schematic and layout. The final design was theorized to have a delay of 1.59 ns, but through testing the device it was proven to be 1.36 ns which left an error of 23 percent between the two delays.

II. Introduction

To create an optimized Full Adder circuit the logic of the circuit had to be laid out first. This circuit involves three inputs and two outputs. The two outputs are the sum and carry of the inputs. As binary addition only has two states, 1 or 0, the carry bit is used as an input for the next full adder in the series or to denote an overflow. The sum denotes the number associated with the current place when the inputs are added together. This logic can be seen in Figure 1 as it is represented in the truth table.

Figure 1: Full Adder Truth Table. [1]

Input			Output	
A	B	Cin	Sum	Carry
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

With the truth table of the Full Adder done the logic for each of the outputs must be created. This is done with kmaps as to find the most efficient logic that can be used to create the circuit. The equations used being $S = A\overline{BC} + \overline{A}B\overline{C} + \overline{AB}C + ABC$ and $Cout = AB + AC + BC$. These equations show the necessary logic to have the Full Adder working correctly. The kmaps for the outputs can be seen in Figure 2.

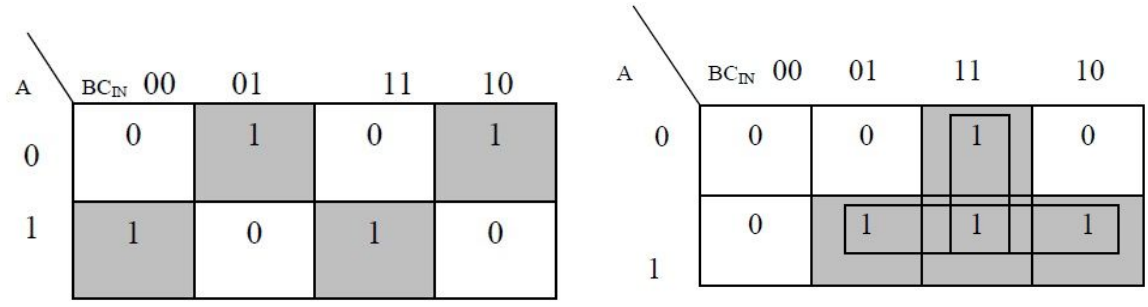

Figure 2: Right; Kmap for the Sum output. Left; Kmap for the Carry Out output.

From the previously gained logical expressions circuit models were made using CMOS logic which uses NANDs, NORs, and Inverters. From these initial models delay calculations were done to determine which would be the fastest to work with. This came down to a circuit comprised of two XORs and three NANDs. This circuit was to operate faster than any other developed circuit having a delay of only 1.5 ns theoretically.

III. Procedure

To create an optimized Full Adder the logic shown in Figure 1 and Figure 2 was initially created. These were the very first steps in creating the circuit. From here the logic was developed upon further, changing the logical expressions to find more ways to make the circuit. These various circuits were comprised of different amounts of NANDs, NORs, and Inverters to see which of them would be seen to have the shortest delay. Each of the worst case paths for these circuits can be seen in Figure 3. It can be seen that the fastest design that involved two XORs for the sum output is not on the list. This is because XORs do not truly exist in CMOS technology. The circuit was slowed, but only slightly by replacing the XORs with NANDs and inverters. These allowed for the same functionality of the XORs, but with the implementation of usable CMOS gates. Some of these gates may also seem as though they are near copies of one another, but the fastest circuits were tested with additional inverters to determine whether the additional steps would have the circuit operating with a smaller delay or not.

Figure 3: Worst case paths of various designs.

Design
NAND2-NAND2-NAND2-NAND2-NAND2-NAND2
INV-NAND3-INV-NOR4
INV-NAND3-INV-NOR4-INV-INV
INV-NAND2-NAND2-INV-NAND2-NAND2
INV-NAND2-NAND2-INV-NAND2-NAND2-INV-INV

For each of these worst case paths for each of the circuit the number of stages, logical effort, parasitic delay, and the overall delay had to be calculated in order to determine which of the paths would have the least delay. Each was tested by performing hand calculations on each of the paths to determine each of their characteristics to determine the overall delay. Each of the gates have a predetermined parasitic and logical delay. The parasitic delays of each of the gates can be added together to give the total parasitic delay while the same can be done for the logical effort. To determine the electrical effort the input and output capacitances were compared to give a value of 15RC for each of the worst case paths. The logical effort could be found by using the following equation: $F = G * B * H$. G being the logical effort, B being the branching of the path, and H being the electrical effort. With effort delay the number of stages could be determined, but if the number of stages was greater than this number the total number of stages found in the circuit was used as a substitute as the number calculated is a best case scenario. With the number of stages the effort delay could be calculated by using the following equation: $\hat{f} = F^N$, where N is the number of stages in the path. With this the delay could be calculated with $D = N\hat{f} + P$, where P is the parasitic delay of the path. Each of these variables and the delay associated with each of the worst case paths of each circuit can be seen in Figure 4.

Figure 4: The worst case paths with each of their associated delays and other values.

Design	N	G	P	D
NAND2-NAND2-NAND2-NAND2-NAND2-NAND2	6	5.62	10	32.86
INV-NAND3-INV-NOR4	4	5	9	28.8
INV-NAND3-INV-NOR4-INV-INV	6	5	11	28.43
INV-NAND2-NAND2-INV-NAND2-NAND2	6	3.16	10	26.5
INV-NAND2-NAND2-INV-NAND2-NAND2-INV-INV	8	3.16	12	29.05

With each of these circuits tested to determine which had the smallest delay the circuit that had replaced the two XOR path was determined to be the fastest path available. This path can be seen in red in Figure 4. As this circuit was seen to be the fastest it was selected to move further in modeling the circuit. First the circuit was modeled by using NAND, NOR, and inverters to have a functioning model to test. This can be seen in Figure 5 where the circuit was modeled in Cadence using a schematic. Each gate of the chosen path must also be sized using the following equation: $Cin = \frac{g_n Cout}{\hat{f}}$. This equation gives a value for the capacitance for each of the gates in the path and with this capacitance the the sizes can be determined. Ratios for each of the gate types was determined by comparing the capacitances the current flows through for each gate. For example an inverter has a 2:1 ratio. With this ratio the size of each of the gates could be determined by using the equation: $S = \frac{C}{T} * r_{p/n}$. $r_{p/n}$ being the ratio for either the pmos or the nmos for each of the gates and T being the sum of the ratio. For example T

for an inverter would be three. The sizes for each of these gates can be seen in Figure 6.

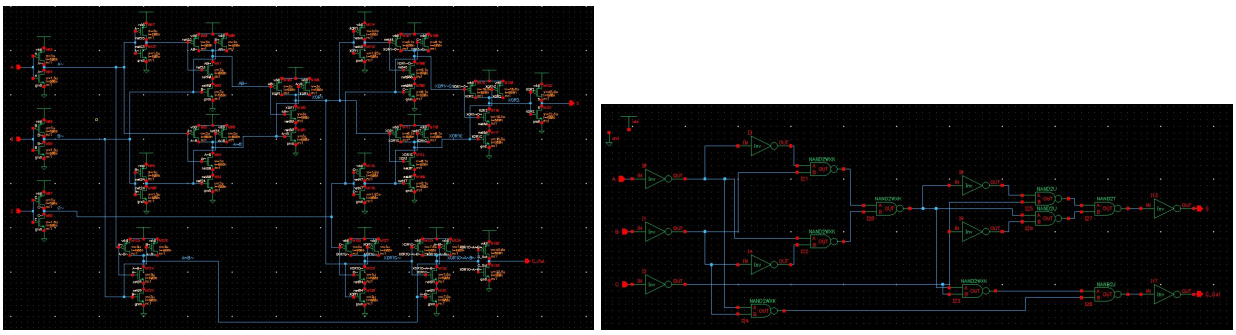

Figure 5: Left; The transistor level schematic of the Full Adder. Right; The symbol schematic of the Full Adder.

Figure 6: The gate sizes for each of the gates in the circuit.

Component	Size Ratio
Nand2J	7.5:7.5
Nand2K	3:3
InvS	45:22.5
Nand2T	16.5:16.5
Nand2U	8:8
InvV	3.9:1.95
Nand2W	3:3
Nand2X	3:3
InvY	3:3
InvZ	3:1.5

With each of these sizes calculated the layouts of each of the gates must be made. Each gate had its own layout made and tested to ensure that it functioned as it should be. Some of the gates were the same as others in the circuit which allowed for duplicates to be made. These layouts can be seen in Figure 7. Each of these layouts were tested individually using Design Rule Check to make sure they followed all rules in developing layouts for 600 nm design. These extracted views were then tested using Layout VS Schematic to compare these extracted layouts to the schematic associated with each gate. This ensure that the netlists of the schematic and the layout match. By further testing each layout with inputs and outputs each can be seen to work as intended. The extracted layouts can be seen in Figure 8.

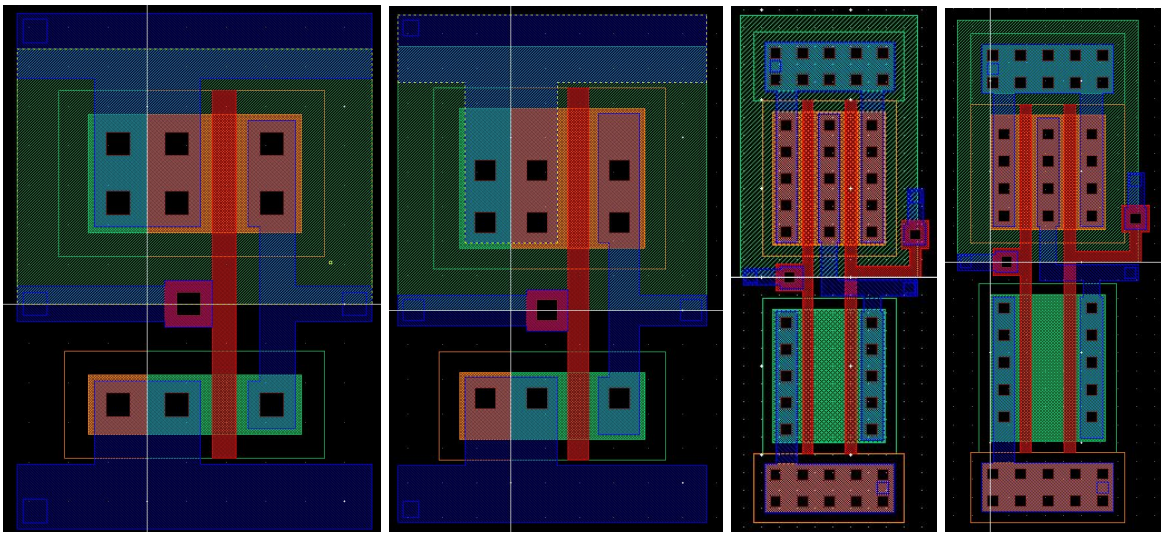

Figure 7: Layouts from left to right; InvIN, InvV, NAND2J ,NAND2U.

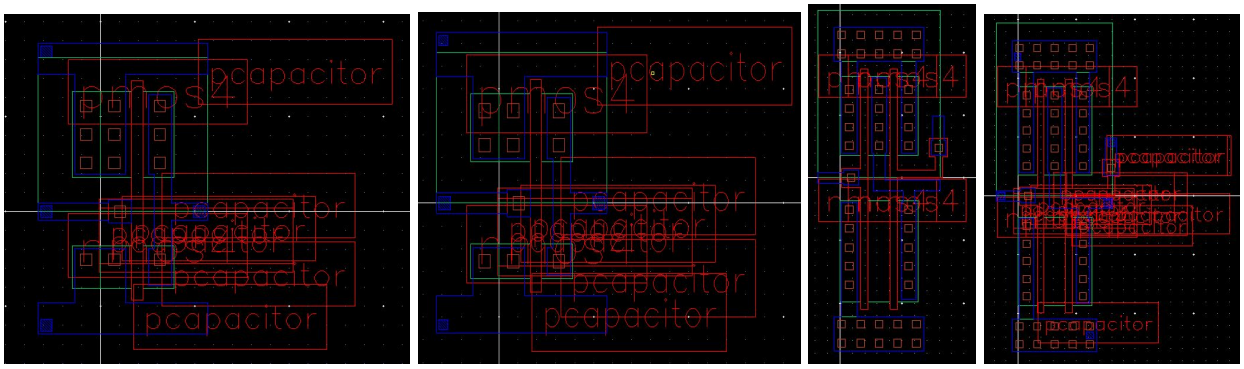

Figure 8: Extracted layouts from left to right; InvIN, InvV, NAND2J ,NAND2U.

With each of the gates having a layout and extracted view that were fully tested to ensure that they worked properly the final step was taken. Each of the individual layouts was brought together to make a standard cell snap together together layout. This layout using metal bars at the top and bottom of the standard cell layouts of each of the gates in order to connect them. Utilizing both metal one and metal two these bars connected each of the inputs and outputs of the gates. The snap together layout can be seen in Figure 9 and the extracted view can be seen in Figure 10.

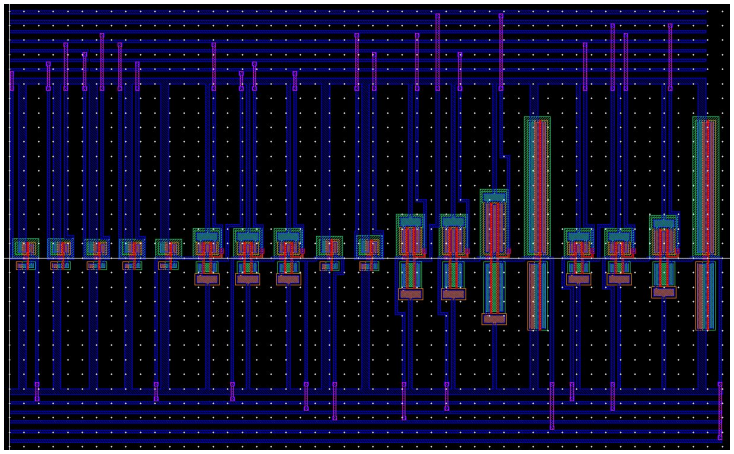

Figure 9: Standard Cell Stand Together Layout of the Full Adder Circuit.

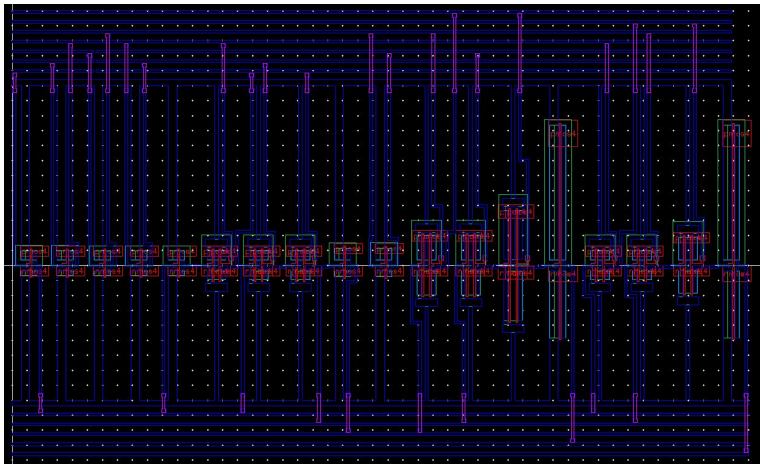

Figure 10: Extracted view of the Standard Cell Snap Together Layout of the Fuller Adder Circuit.

IV. Results

The schematic of the Full Adder circuit was tested by using bit inputs and putting in bitstreams that covered every possible combination of bits entering the Full Adder. These bit streams followed the truth table perfectly, though they caused minor issues in the graph of the outputs as two bits changing state at once caused the Carry output and the Sum output to flip when they were not meant to. The circuit does fix itself just a moment later once it has time to react to both of the changes in state. This can be seen in Figure 11.

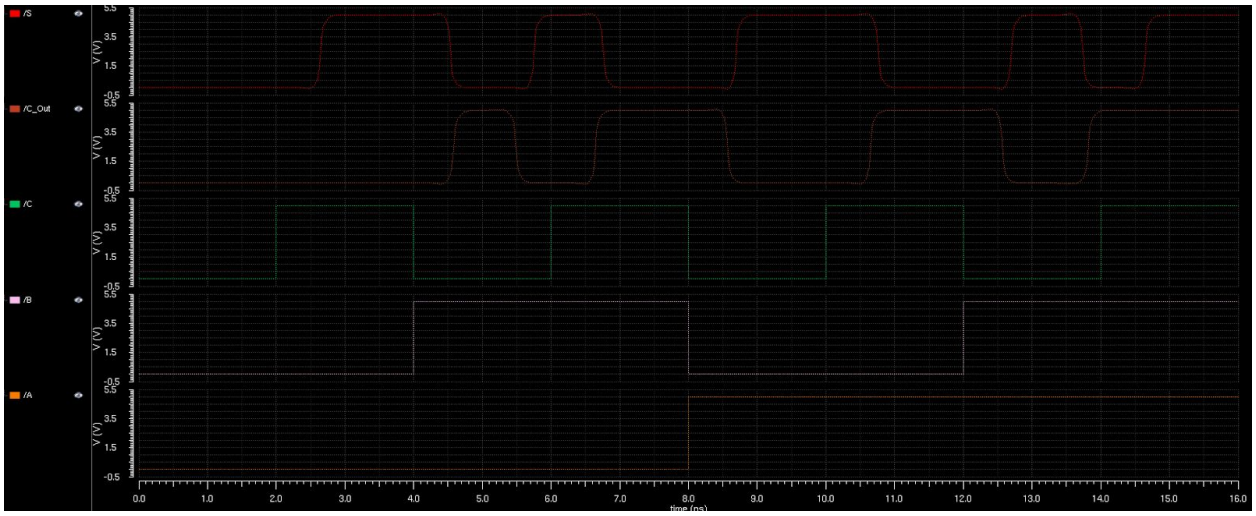

Figure 11: Graph of the inputs and outputs of the schematic view when tested.

After the schematic was tested each of the gates were tested. This was done to ensure that each part going into the Standard Cell Snap Together Layout was working properly prior to its assembly. Each of these tests checked if the NAND gates were working properly and if the Inverters were working properly. The two graphs in Figure 12 show that the NAND logic and the Inverter logic work just as they should. When the inverter receives a value of one it outputs a value of zero and vice versa. The NAND gate only reacts to both inputs being one which results in the output being a zero whereas any other time it outputs a one.

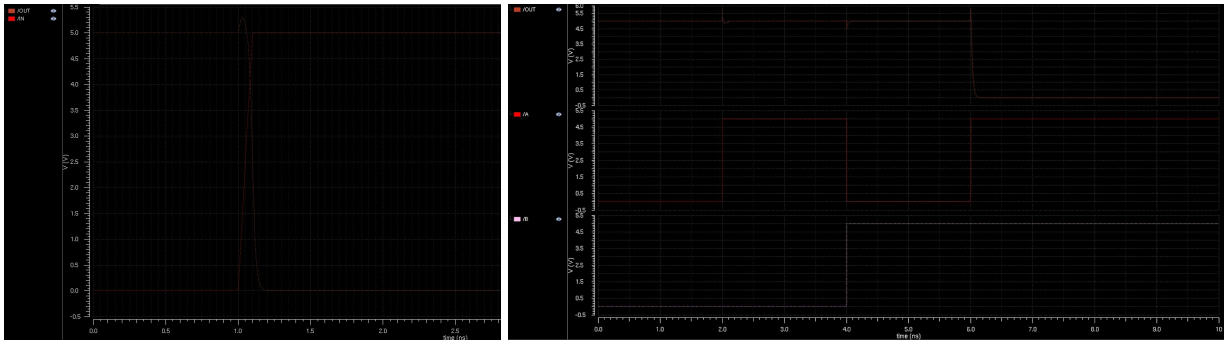

Figure 12: Left; The graph of a tested Inverter Layout. Right; The graph of a tested NAND Layout.

The Standard Cell Snap Together Layout was to be tested, but through an odd error that states: the layout has been updated and a new extracted view is necessary. The test could not properly be conducted. Attempts to remedy this error were made and steps to go back as far as the schematic view to check and save the work as well as performing an LVS on the extracted view which produced a valid netlist. All other tests validate that the final layout should work properly under testing, but generating a netlist through ADE seems to cause this error.

Though the final layout could not be tested properly the delay was gather from the schematic view which proper gate sizes assigned. This delay was measured to be 1.36 ns when compared to 1.59 ns from the calculated delay. This leaves the error to be around 23 percent which is well within the threshold of 20 to 30 percent error.

V. Conclusions

From this project it can be seen that the optimized Full Adder that was designed worked just as it should. It performed within the bounds of error and kept a relatively low delay with the actual delay being only 1.36 ns. Each of the gates were sized in accordance to the hand calculations made for the circuit and the layouts reflected these sizes. While the final Snap Together layout could not be tested properly to a persistent error that stems from some unknown source, the schematic test and the individual layout tests show that the functionality of the final layout should be correct.