Applying Simple CMOS Gates

Aidan Sharpe

I. Introduction

The humble adder is arguably the most important of all arithmetic devices found in a computer. Both subtractors and multipliers may be built using very little extra circuitry, since subtraction is the same as adding a negative, and multiplication is just repeated addition.

The simplest adding circuit is the half adder. It has two inputs, A and B, and two outputs, sum and carry. The sum output will be high if only one of the two inputs is high, and the carry output will be high if both inputs are high. More complex devices can be built upon this simple framework.

II. Background

The full adder is a device with three inputs—A, B and carry in (C_{in})—and two outputs—sum (S) and carry out (C_{out}). The sum bit will be high when the sum of A, B, and C_{in} modulo two is one. C_{out} will be high when the sum of A, B, and C_{in} is greater than one. This relationship is seen for all combinations of A, B, and C_{in} in table I.

TABLE I
Full Adder Truth Table

A	B	C_{in}	S	C_{out}
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

Typically, the full adder is thought of as two half adders with their carry outs "or-ed" together. The sum output of the half adder has the same truth table as an XOR gate, and the carry output has the same truth table as an AND gate. Therefore, a half adder can be modeled as an XOR gate and an AND gate. This simple representation of the full adder is seen in figure 1.

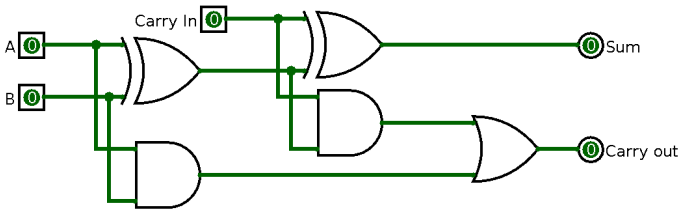

Fig. 1. Simple full adder schematic

.

The function of each gate in figure 1 makes the overall operation clear. When designing circuits in logic simulators, this topology is common. For example, figure 1 was constructed using Logisim, and the same topology is also used in many computers built in the hit sandbox game Minecraft.

III. Schematic Design

Previously, we designed a half adder schematic, seen in figure 2, containing four NAND gates. We purposefully left the carry out inverted, so building the full adder would be easier.

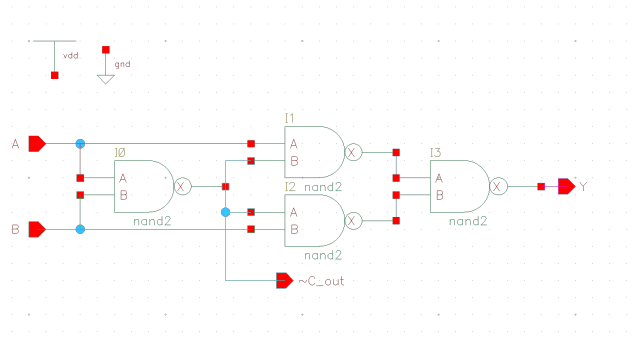

Fig. 2. Half adder schematic

Recall the full adder design from figure 1. We originally "or-ed" together the carry output from each of the two half adders. With our CMOS design, however, we cannot simply build AND gates and OR gates, as CMOS is restricted to inverting functions. By De Morgan's laws, however, we know that $A + B$ is the same as $\overline{A}\overline{B}$. With our half adder design, getting an inverted carry out is easier than getting a non-inverted carry out. Therefore, we can just add another NAND gate to have the same effect as "or-ing" the two non-inverted carry outs. This additional NAND gate is seen in our final schematic for the full adder seen in figure 3.

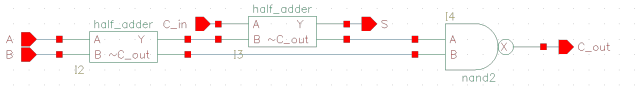

Fig. 3. Full adder schematic

This final design for the full adder is made from nine NAND gates total—four from each half adder and one for the carry out. With our final design, the last step was to size the gates using logical effort calculations to minimize delay.

IV. Logical Effort Calculations

There are two ways of calculating the logical effort for the full adder. One way would be to consider the two half adders as gates and to size them as a whole. The other way, the method we opted to choose, was to size all nine individual NAND gates.

The smallest size for a NAND gate has all n-channel MOSFETs and all p-channel MOSFETs at a width of $w = 2\lambda$.

V. Conclusion