CMSC 250: Adder Circuits

Justin Wyss-Gallifent

February 25, 2023

1 Introduction

Let's now pull together everything we know and build a circuit out of our three gates that adds two binary numbers. When we add two binary numbers bit-by-bit we could simply be adding two bits or we could be adding three bits when there is a carry bit.

2 Half-Adders

First let's just see how to add two bits. There are four possibilities. In each we'll write the result as two bits for future reference. These results are the *sum bit* and *carry bit*, the latter because we'll need to carry it, as we'll see:

p		q		*carry*	*sum*
0	+	0	=	0	0
0	+	1	=	0	1
1	+	0	=	0	1
1	+	1	=	1	0

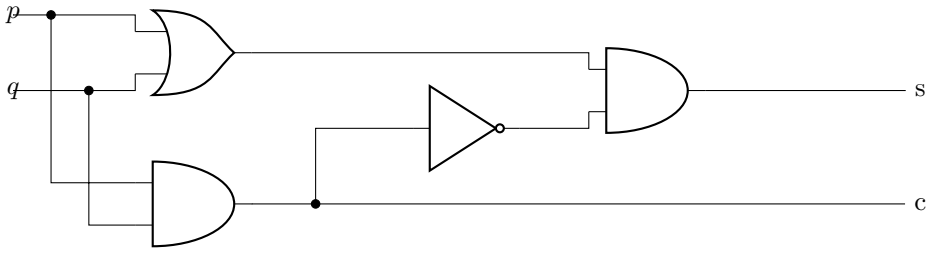
Observe that the result in the 1's place is the same as XORing the starting bits and the result in the 2's place is the same as ANDing the starting bits.

It follows that with an XOR circuit and an AND gate we can add two bits. Since we only have $\wedge$, $\vee$, and $\sim$ we first observe that:

$$p \,\mathrm{XOR}\, q = (p \vee q) \wedge \sim (p \wedge q)$$

We can therefore build a circuit that will do the job.

Definition 2.0.1. A *half-adder* adds two bits and produces a *sum bit* and a *carry bit*. One way to design this circuit is as follows:



Take a second to trace through this to see that s and c will contain the sum bits (the $\oplus$) and the carry bit (the $\wedge$).

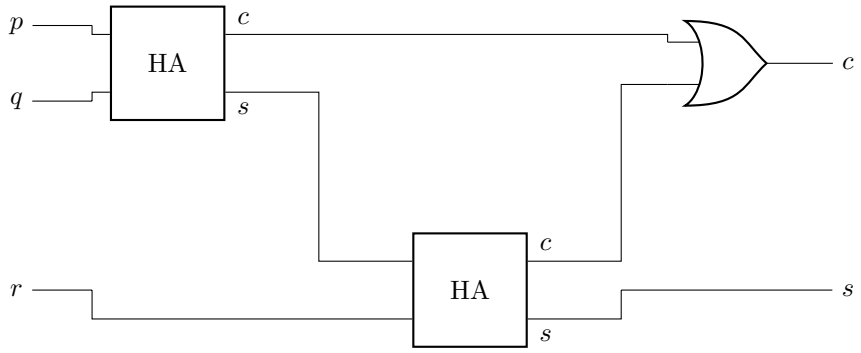
3 Full-Adders

Now let's see how to add three bits. There are eight possibilities. In each we'll write the result as two bits for future reference.

p	$+$	q	$+$	r		*carry*	*sum*
0	+	0	+	0	=	0	0
0	+	0	+	1	=	0	1
0	+	1	+	0	=	0	1
0	+	1	+	1	=	1	0
1	+	0	+	0	=	0	1
1	+	0	+	1	=	1	0
1	+	1	+	0	=	1	0
1	+	1	+	1	=	1	1

While we could get really technical and figure out how to draw a circuit for this, interestingly it can be built out of two half-adders and an OR gate.

Definition 3.0.1. A *full-adder* adds three bits and produces a *sum bit* and a *carry bit*. One way to design this circuit is as follows:



It's far less obvious than the half-adder that this does what is intended. Basically this circuit makes two claims. Here the functions s and c return the sum and carry bits respectively.

1. $s(p, q, r) = s(s(p, q), r)$ which equals $(p \oplus q) \oplus r$.
2. $c(p, q, r) = c(p, q) \vee c(s(p, q), r)$ which equals $(p \wedge q) \vee ((p \oplus q) \wedge r)$.

This can be verified by simply writing down a truth table which contains the two logical expressions above.

p	q	r	$p \oplus q$	$(p \oplus q) \oplus r$	$(p \oplus q) \wedge r$	$p \wedge q$	$((p \oplus q) \wedge r) \vee (p \wedge q)$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	1	0	0	0
0	1	1	1	0	1	0	1
1	0	0	1	1	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	1
1	1	1	0	0	1	1	1

We can see that the column $(p \oplus q) \oplus r$ equals the value of the sum bit and the column $((p \oplus q) \wedge r) \vee (p \wedge q)$ equals the value of the carry bit.

4 An Adder Circuit for n Bits

The key to building a circuit which adds binary numbers is to observe that we start by adding the rightmost bits, and there are only two of those, so we can use a half-adder.

Each successive addition involves three bits, one of which is the carry bit from the previous sum and which might be 0.

The very last addition takes the carry bit and just dumps it on the left. We can therefore proceed as follows to add the binary numbers represented by $a_n ... a_1$ and $b_n ... b_1$ to get $d_{n+1} d_n ... d_1$.

In total then we require 1 half-adder and $n - 1$ full adders: