Adders, Encoders, and Decoders

More sophisticated applications of circuits

Akilesh Praveen & Ashwath Krishnan

UMD

January 20, 2020

Agenda

1. Announcements
   - Project 2
2. Adders
   - Addition
   - Half Adders
   - Full Adders

Announcements

Project 2

- Project 2 has been released, and can be found on the course website under 'Week 3' on the calendar.
- Everything you need to complete the project will be covered in today's lecture, with optional supplementary material provided on the course website under Week 3's resources section.

Some Reminders

- As the projects become more involved, we'd like to remind you to reach out and attend office hours if you are struggling.
- Additionally, here's a reminder that all projects can be group projects, but you'll have to let us know **4 days** prior to the project's due date if you'll be working in a group. (Max size of 3)

Adders

Addition

How would you add the following binary numbers?
11101 & 11011

Addition

- Let's visualize the computation as long addition
- How would you go about solving it this way?

$$\begin{array}{r} 11101 \\ +\,11011 \\ \hline \end{array}$$

Addition

- Let's visualize the computation as long addition
- How would you go about solving it this way?

$$\begin{array}{r} {\color{red}1} \\ 11101 \\ +\,11011 \\ \hline 0 \end{array}$$

- This phenomenon is known as a carry
- This occurs when the numbers we are adding exceed the limit imposed by the number system we are using
- Where else during this computation would this phenomenon occur?

Half Adders

- First, let's represent the 1s as `TRUE`s and the 0s as `FALSE`s
- Think back to last week when we talked about designing circuits that dealt with booleans
- The big question: can we perform the addition process that we just talked about using such gates?
- The answer: **absolutely**

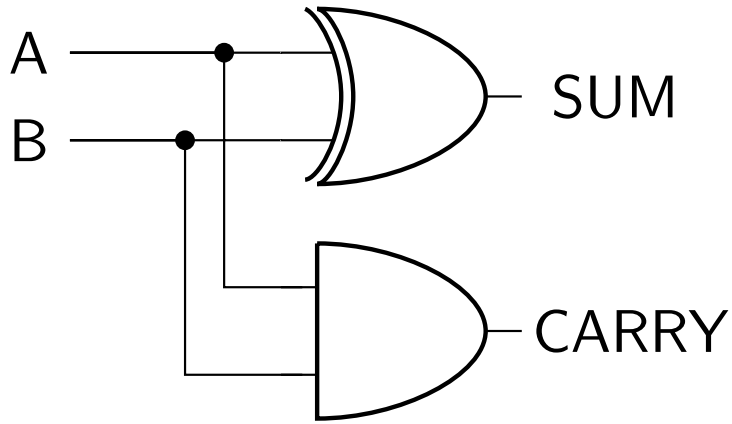
Half Adders

- Let's take this step by step.
- Looking at the addition example, what's the simplest operation we can take away from that?
- Let's see if we can build something that can take two one-bit numbers and add them, producing a carry if applicable
  - In terms of inputs, we will need two booleans
  - In terms of outputs, we will need two booleans (one for our sum, and one for the carry)
- Incidentally, we have been describing a **Half Adder**
- We have defined our inputs and outputs- try to design a Half Adder using the gates we've learned

Half Adders

- Below is the formal truth table and circuit diagram for an optimal Half-Adder
- In this case, we can see that the Sum and Carry can be represented by an `XOR` and `AND` gate, respectively

Half Adder		
In	Sum	Carry
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

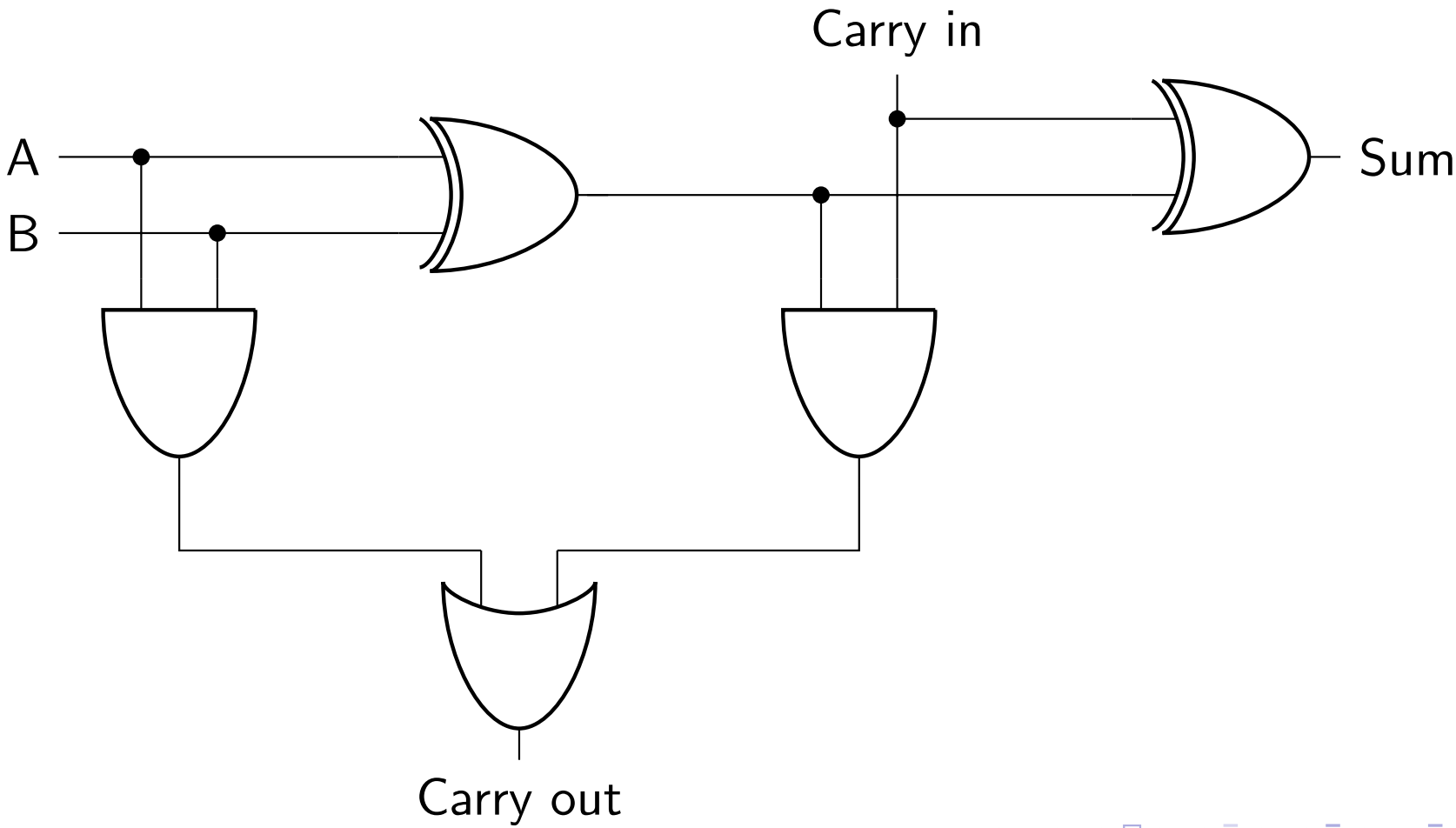


Full Adders

- The Half Adder is great, but limited in functionality
- Let's work on 'scaling' it for use with multi-digit addition.
- How do we accomplish such a task?
- All we need to do is create a circuit that can:
  - Add two one-bit numbers and produce a sum and carry
  - In case the previous addition resulted in a carry, be able to add in that third number as well
- Essentially, we need a logical circuit that can add three one-bit numbers. We can then use this circuit for each digit in our addition.

Full Adders

- Below is a full adder that meets our previously outlined specifications exactly.

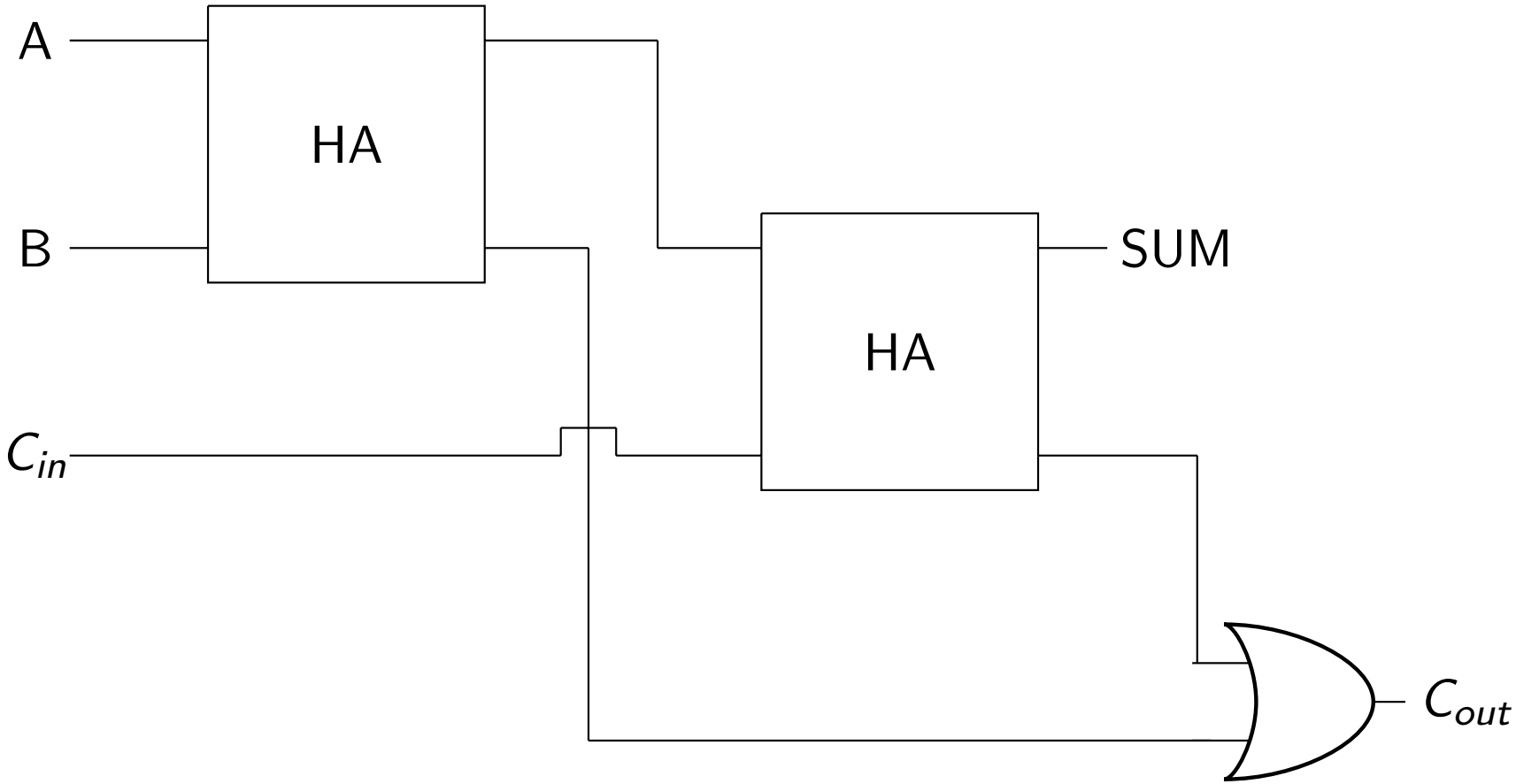


Full Adders

- However, it's important to note that there's an easier way to think about the implementation of a full adder
- Notice that the Half Adder already has half the functionality of the Full Adder that we need to design
  - Specifically, it can add two one-bit numbers and produce a sum and carry. All we need to add is the ability to carry a bit into the addition.
- That being said, let's try and employ the use of Half Adders within our Full Adder construction

Full Adders

- Take a look at this Full Adder construction, built using Half Adders.



Full Adders

- If full adders are better than half adders, why do we use them?
- During addition of any size, you may notice that there is always one addition 'operation' that will never have a 'carry in'
  - This will always occur in the least significant bit
- Since a half adder is fundamentally simpler than a full adder, we always use it when we can
- Now that we have the ability to add two one bit numbers, with or without a carried '1' coming into the addition, let's put these together (literally) and create a multiple digit adder.

Full Adders

- Here's an example of how we can chain adders to increase the number of bits that we can perform addition with
- Note that the Full Adder handling `A0` and `B0` can be replaced with a Half Adder

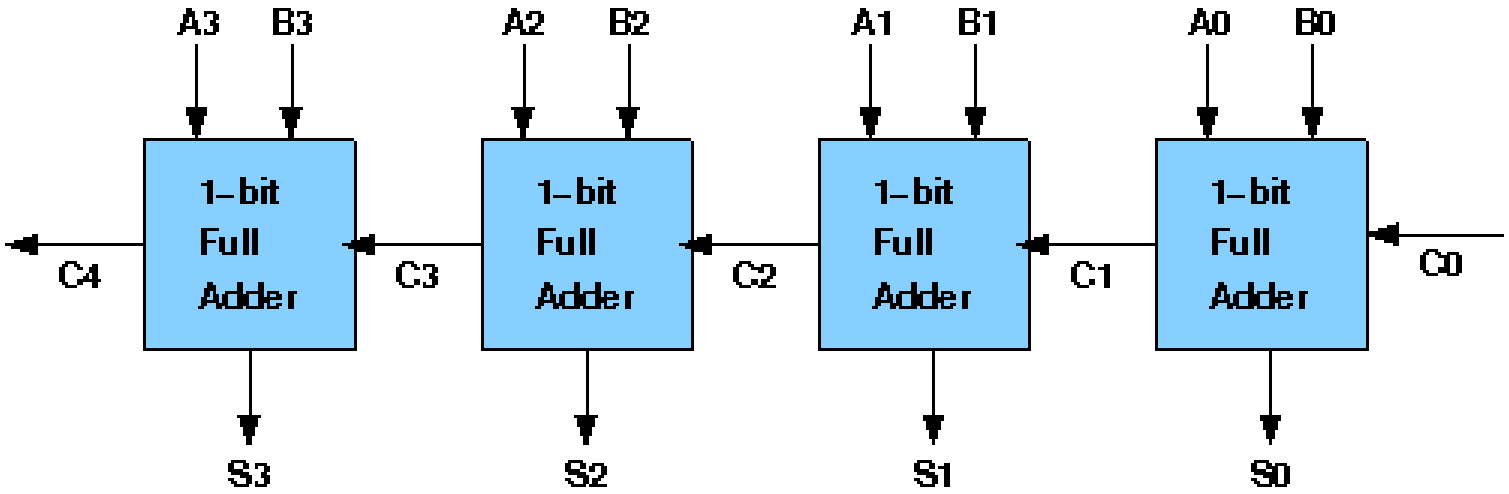


Image courtesy of IIT

Full Adders

- Okay, now we can add simple n-bit numbers. But then how can computers show us numbers that aren't in binary?
- We need a way to translate these binary signals into numbers that we can use (or that a computer can display for us)
- In other words, we need to somehow convert the outputs to base-10