Adders, Encoders, and Decoders

More sophisticated applications of circuits

Akilesh Praveen - CMSC389E

UMD

March 5, 2020

Agenda

1. Announcements
   - Project 2
   - Project 1 Updates
2. Adders
   - Addition
   - Half Adders
   - Full Adders
3. Decoders & Encoders
   - Overview
   - Formal Definitions
   - Decoders
   - Encoders
4. Project 2
   - Project Tips

Announcements

Project 2

- Project 2 will be released Saturday (2/15), 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.

Project 1 Checkpoint

- https://ter.ps/proj1spr2020
- If you would like, upload your worldfile to this by **tonight**, Friday 2/14, and we will check it to make sure you're on the right track/ give you feedback for it. (This is **optional**)

Project 1 Extension/Grading

- We will be grading the project 1 **in class** next week. Attendance is mandatory for this (unless excused).
- For this reason, P1's deadline has been extended until **the beginning of class on 2/21**.

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?

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

Addition

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

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

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 `1`s as `TRUE`s and the `0`s 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?

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)

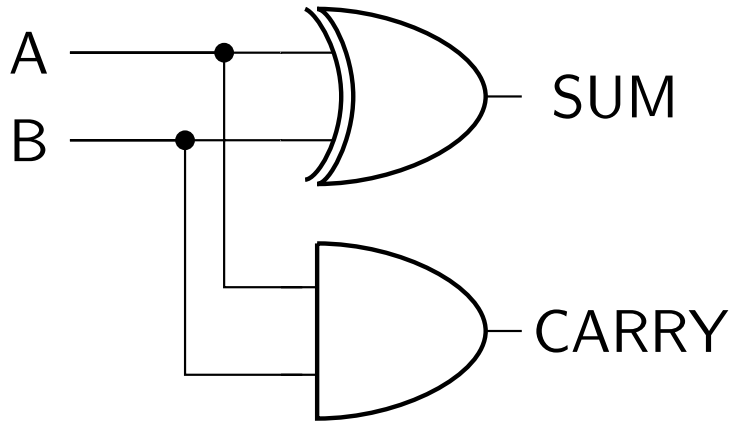
Half Adders

- 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

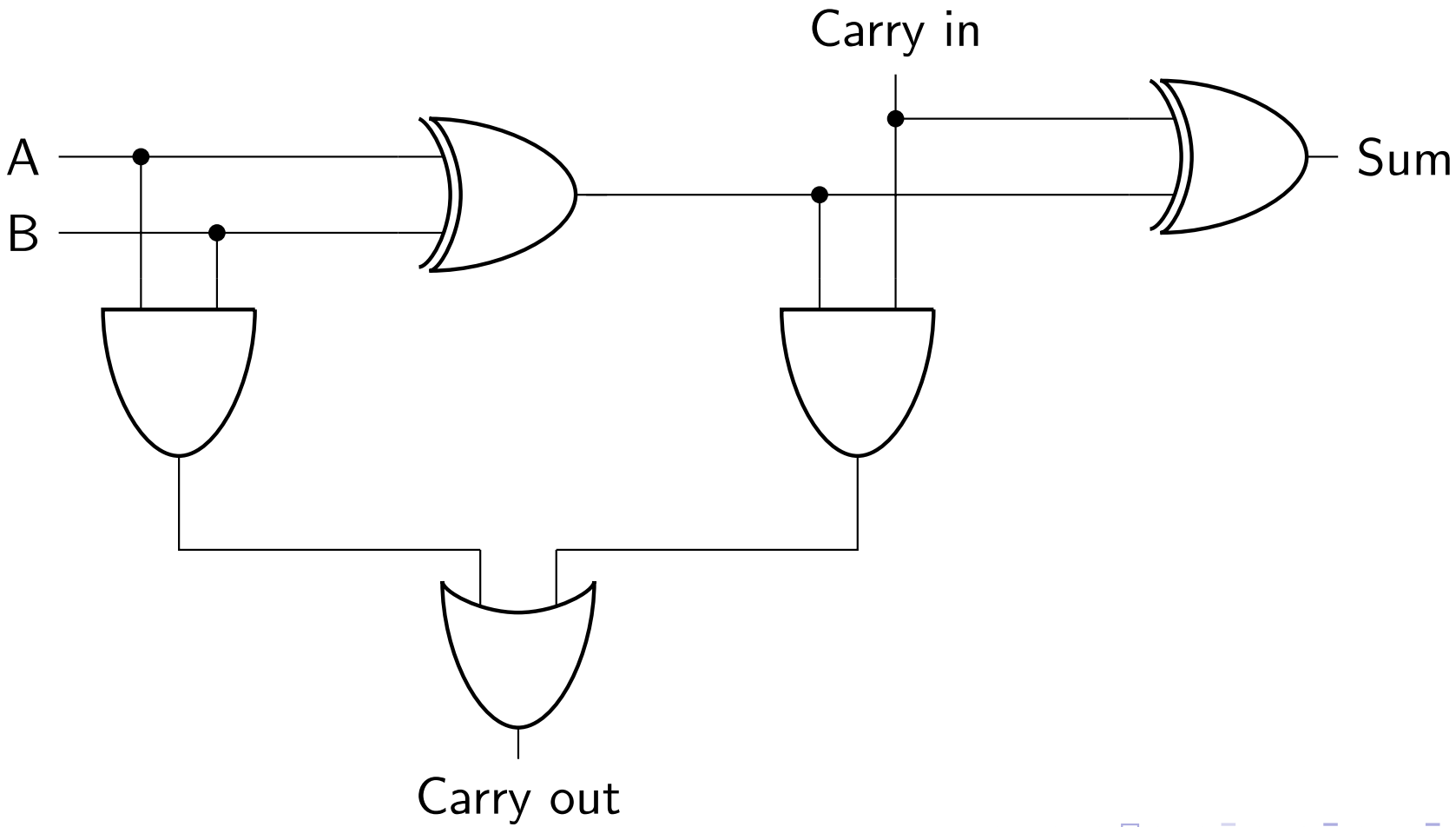


Full Adders

- 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.

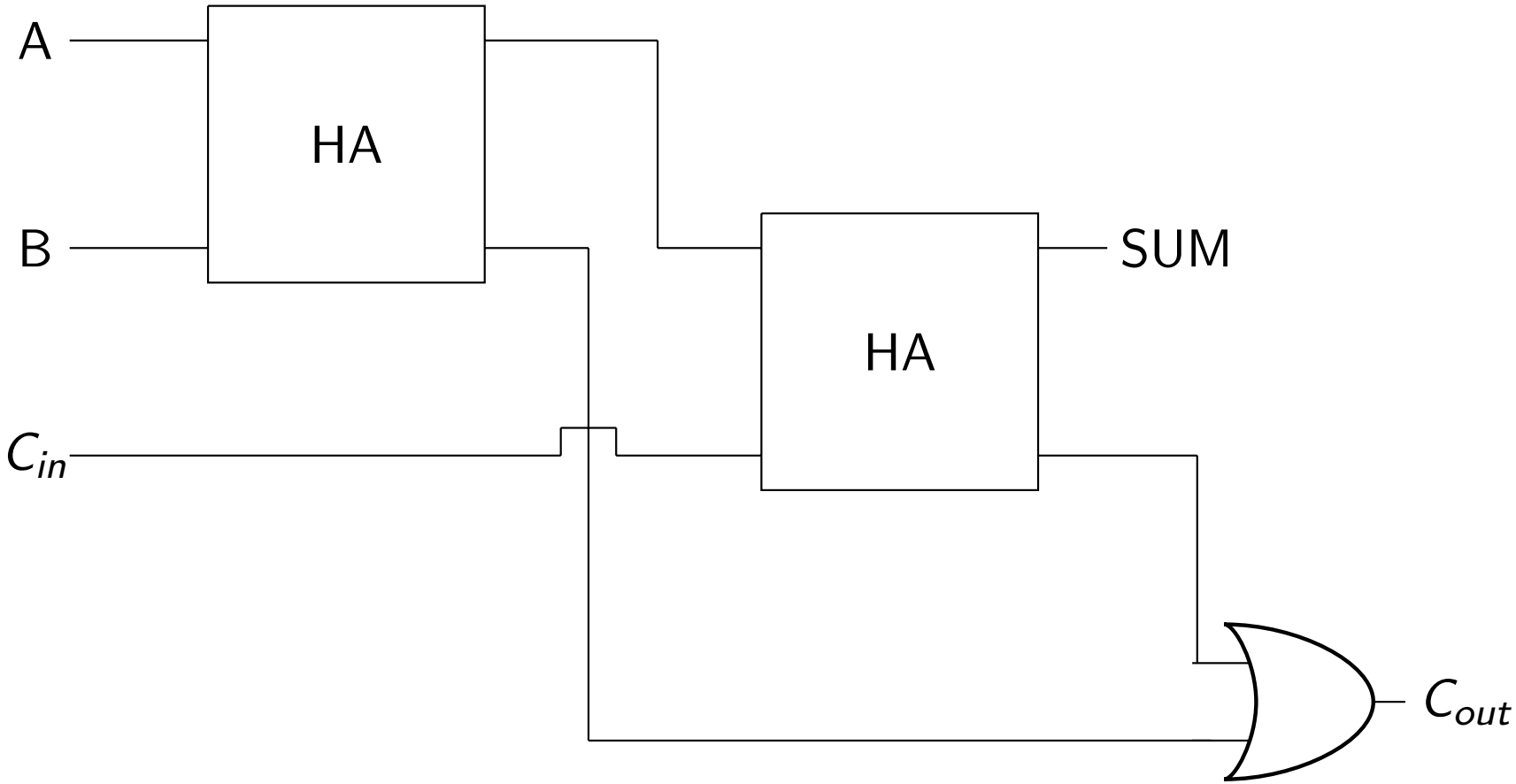


Full Adders

- 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

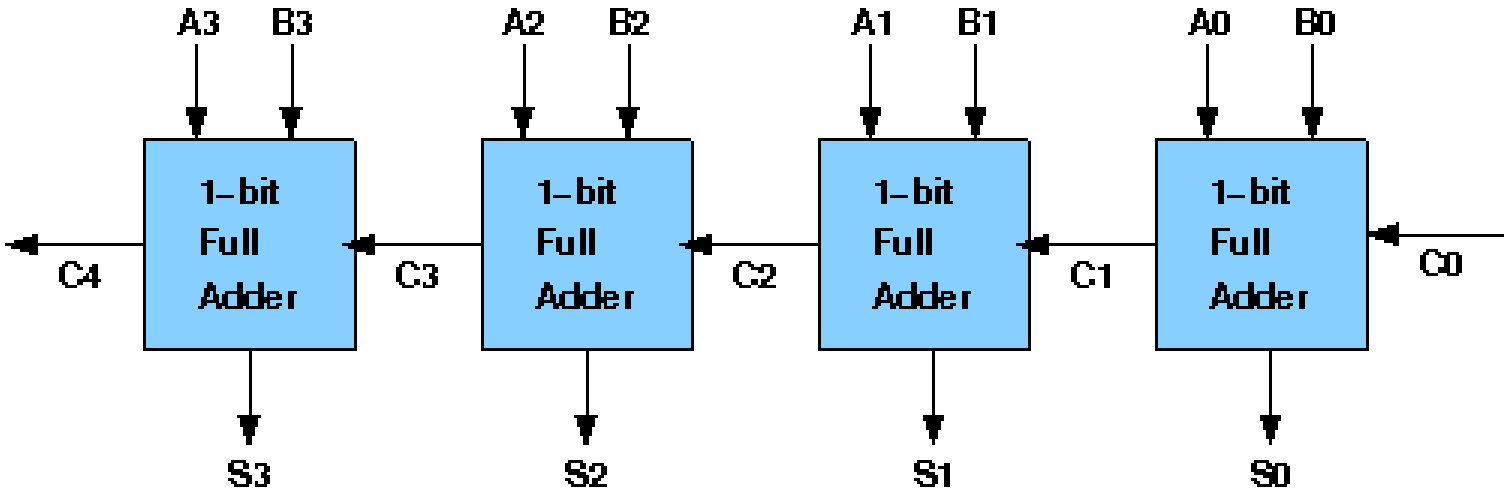


Image courtesy of IIT

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

Decoders & Encoders

Decoders & Encoders

- We've run into the issue of wanting to convert binary to decimal and vice versa
- It turns out that this is an essential problem in digital logic (this specific case more-so in computer science)
- Engineers have designed two logical circuits to accomplish these tasks
  - Decoders are circuits that can convert binary signals into a decimal representation of those signals
  - Encoders are circuits that can condense those decimal representations of signals back into binary
- These circuits can be generalized to different number systems as well, not just binary and decimal as we've specified

Decoders & Encoders

- **Decoders** $\Rightarrow$ Takes a 'denser' signal and converts it into an 'expanded' signal
  - In our case, the decoder is converting our denser binary signal and converting it into an expanded decimal signal
  - Less wires coming in, more wires coming out
- **Encoders** $\Rightarrow$ Takes an 'expanded' signal and converts it into a 'denser' signal
  - In our case, the encoder is converting our expanded decimal signal back into a denser binary signal
  - More wires coming in, less wires coming out

Decoders

- Decoders allow us to translate a denser signal into its expanded form
- Let's say we had the following 3-bit binary number below. How would you systematically go about converting it to a decimal number?

$$110$$

Decoders

- The easiest way to go about this digitally is to assign a wire for each possible output
- In this case, we will need 3 input wires, each representing a bit of the input binary number
- Therefore, we will need 7 output wires, each representing a decimal value from 0 through 7
- Given output wires O_0, O_1, ... O_7, let's find a way to efficiently wire 3-bit binary inputs I_0, I_1, and I_2 to produce those results

Decoders

- Let's start with a truth table

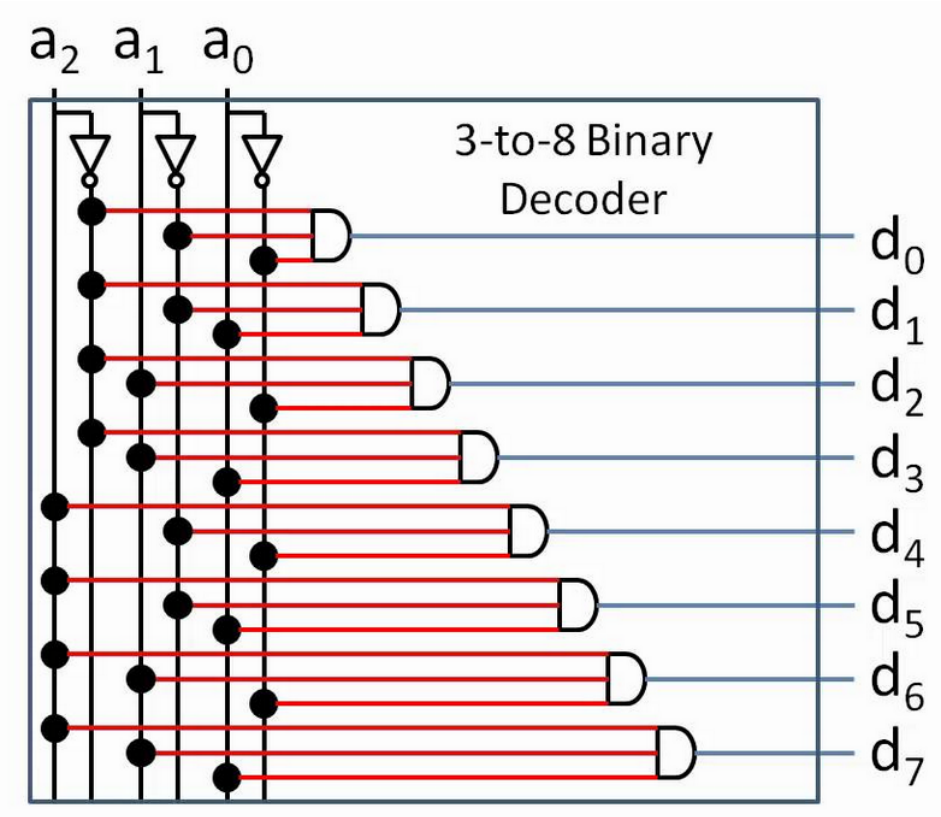
Table I Truth table for 3 to 8 decoder

Enable pin	Input lines			Output lines							
E	I_2	I_1	I_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Note: 'x' denotes don't care condition

Decoders

- It turns out, we can represent this pretty systematically
- Notice the clever use of input buses, which we went over briefly last week



Encoders

- Think of encoders as the complement to decoders- they do everything a decoder can do, but backwards
- Going back to our previous example, let's assume we had 7 input wires, each signifying the decimal numbers 0-7
- Our goal is, based on which wire (0-7) is on, to produce the corresponding binary representation as output along the 3 output wires

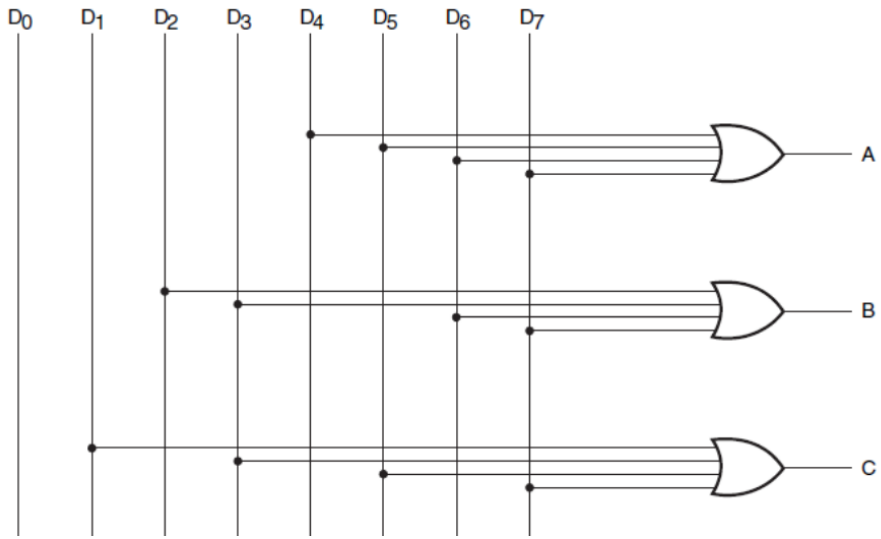
Encoders

1 2 4

- What's so special about these 3 numbers to us?
- The key is that each number 0-7 can be represented as a unique additive combination of 1, 2, and 4
  - e.g. $1 = 1$, $2 = 2$, $3 = 2 + 1$
- This is a principle of the binary number system itself
- By leveraging this fact, we can simply match each unique additive combination of 1, 2 and 4 to its corresponding input wire

Encoders

- Below is an encoder's circuit and truth table- Notice that it looks similar in design to the decoder that we discussed



D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Project 2

Project 2

- For this project, you won't be working with decoders and encoders so much as you'll be working with adders.
- We're moving on to the 'Arithmetic' portion of your ALU now (congrats, the logic processing is good to go!)
- All you have to do now, is take care of how the ALU handles basic adding, and you should be set for this project
- Remember the project guidelines and don't start too late!
- As always, these slides and supplemental resources can be found on the course website