Homework

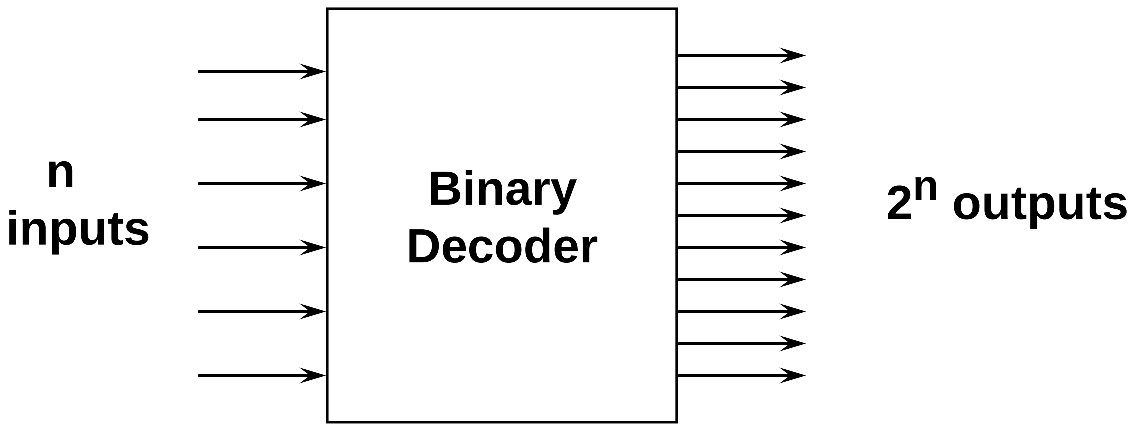
- Reading
  - Tokheim, Chapter 3, 4, and 6.1 - 6.3
  - Logisim Website
- Machine Projects
  - Continue on mp3
- Labs
  - Continue in labs with your assigned section

Combining Basic Logic Gates

- Decoders
- Encoders
- Selectors - Multiplexers
- ALUs
- Control Units
- Buses
- Simple computers

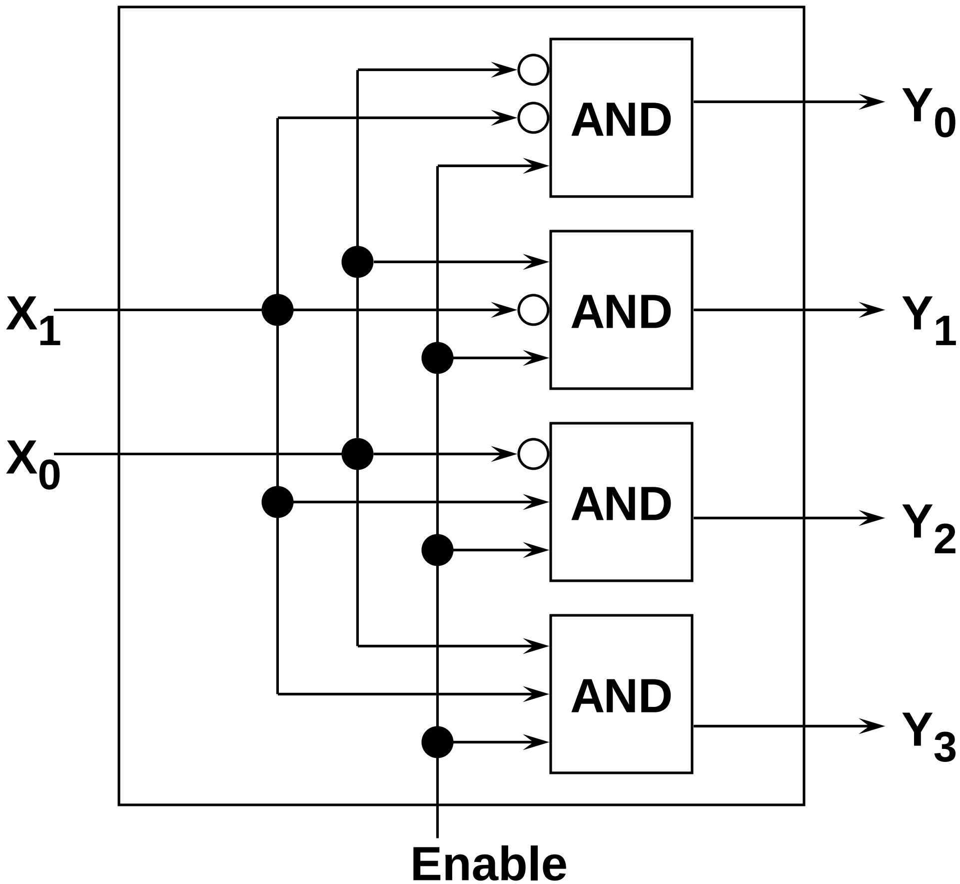
Binary Decoder

- Logic with n input lines and 2^n output lines
- Only one output is a 1 for any given input

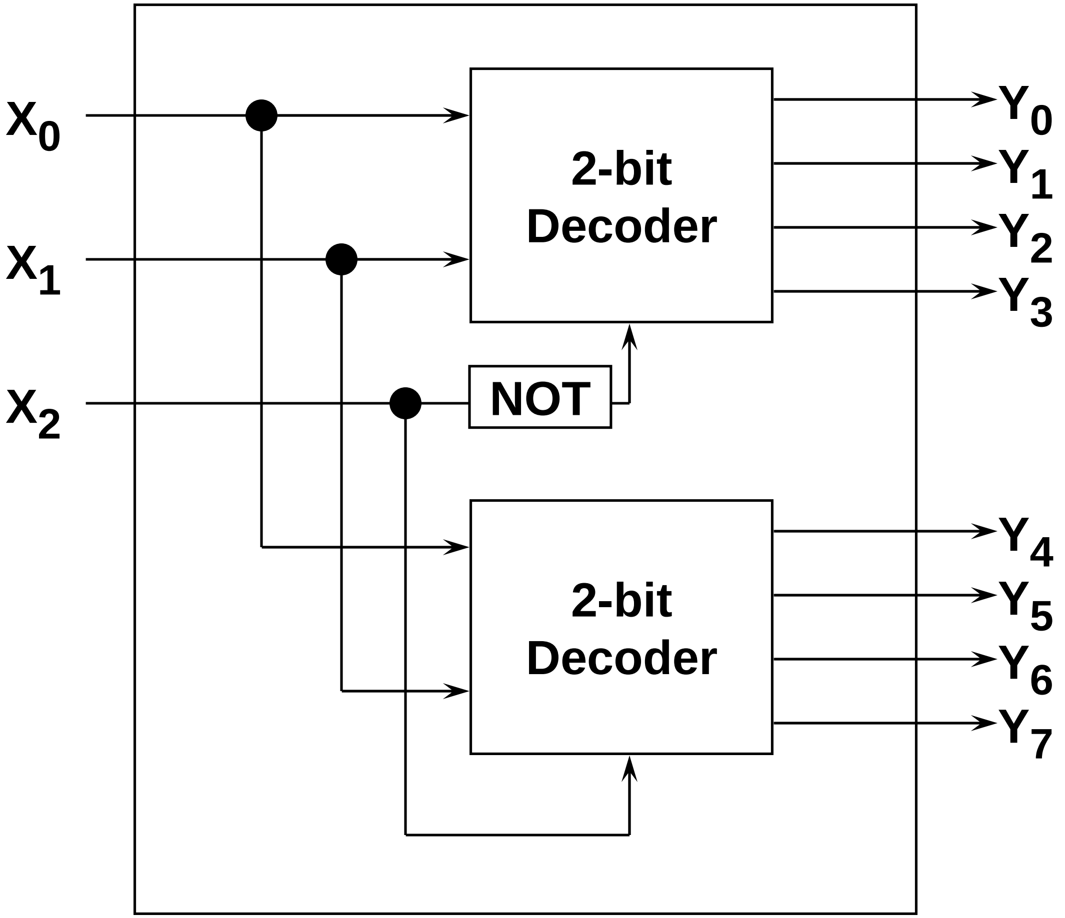


Building a Binary Decoder

- Start with a 2-bit decoder:

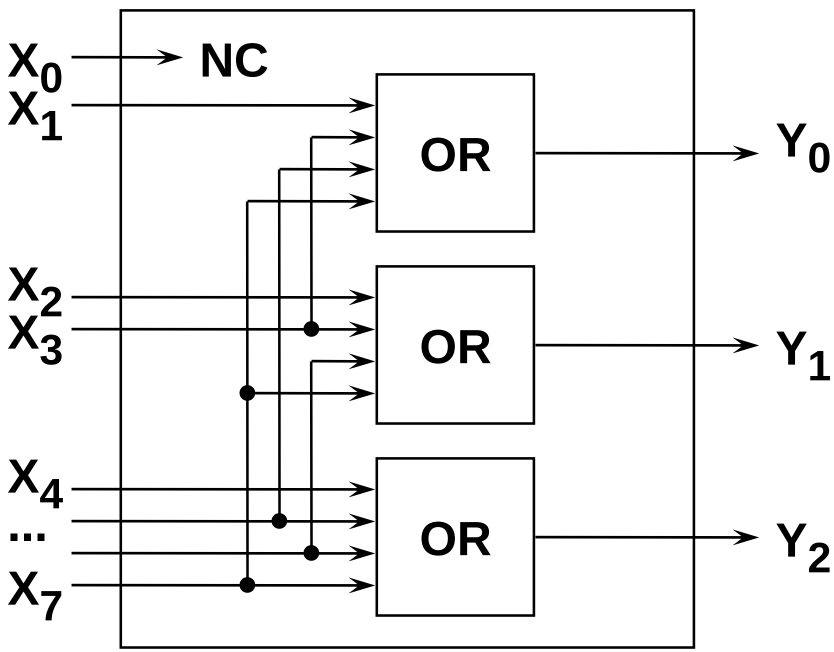


Then Add Two to Make Three...



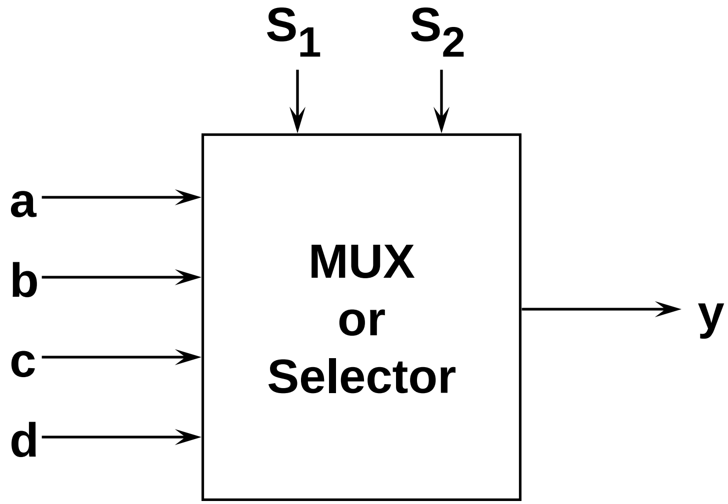
Developing an Encoder

- If we can decode, then we need to encode
- Encode from 1 out of n into a binary weighted form
- A keyboard encoder does this



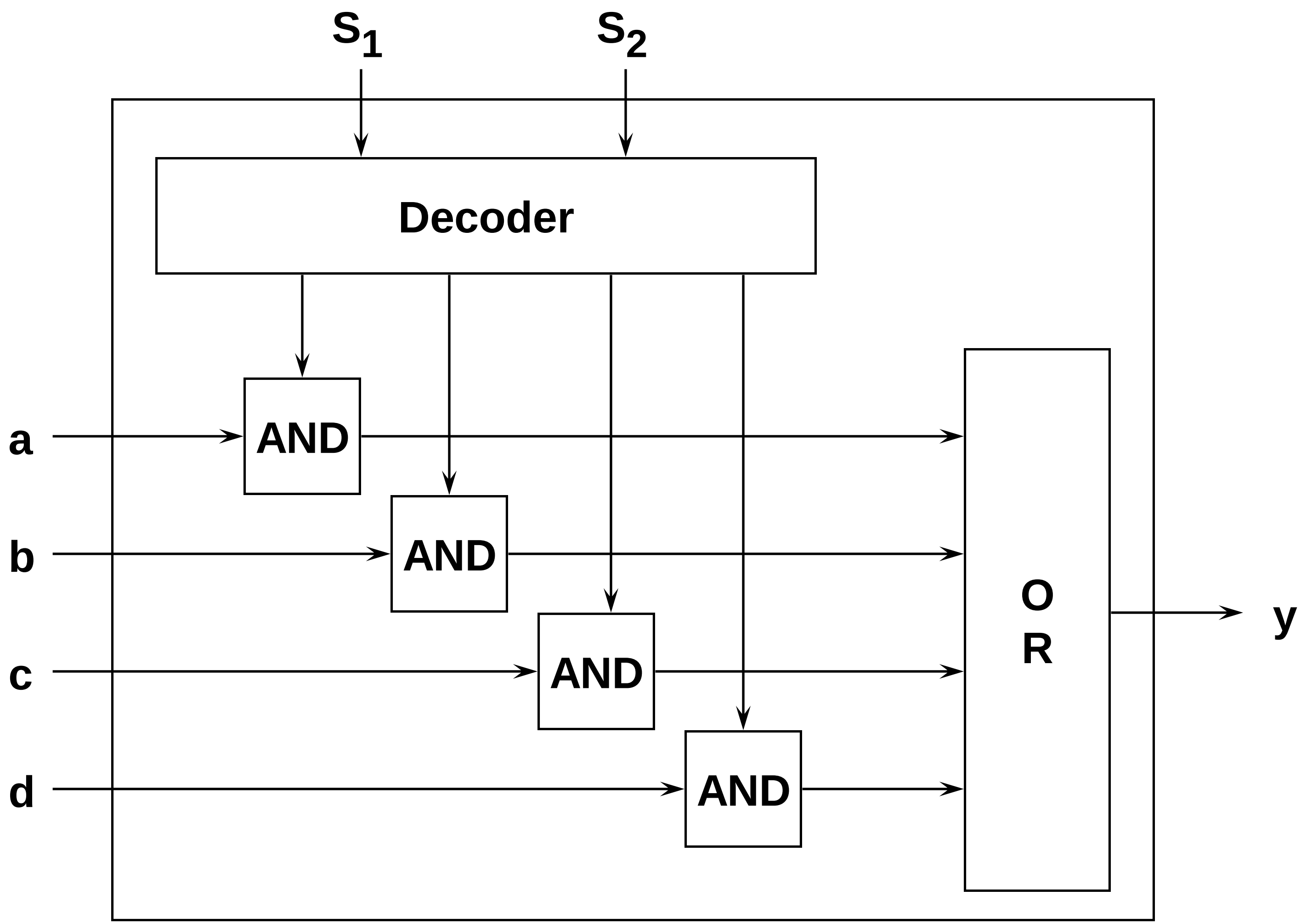
Next Comes a Selector

- Like a switch; also called a multiplexer or MUX

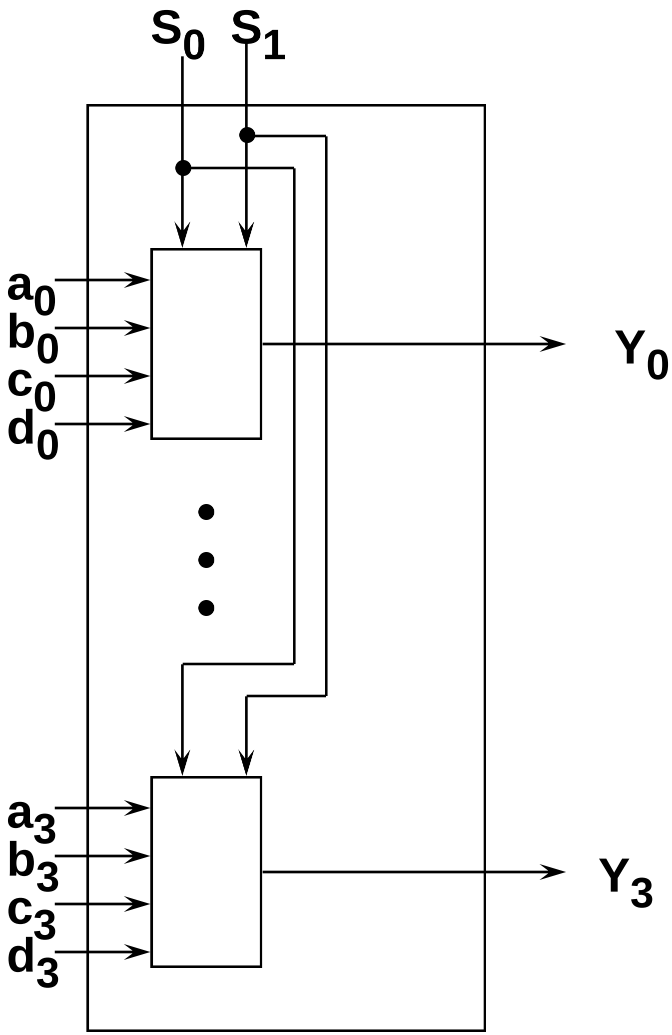


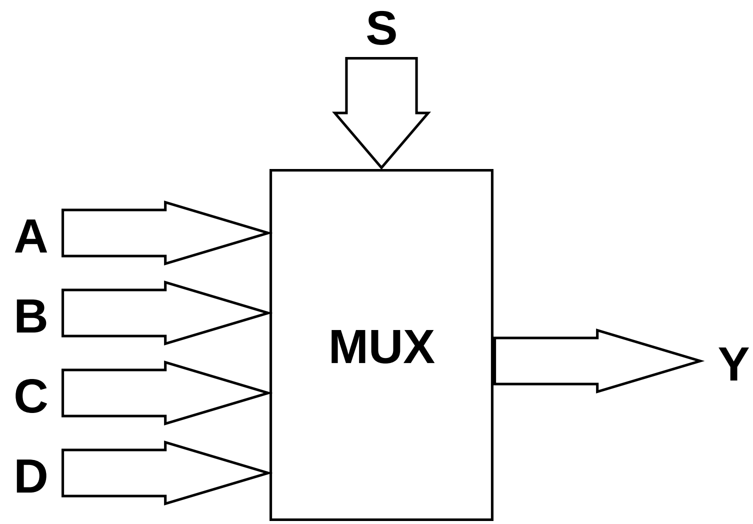
- Again, build it up from simple basic logic gates

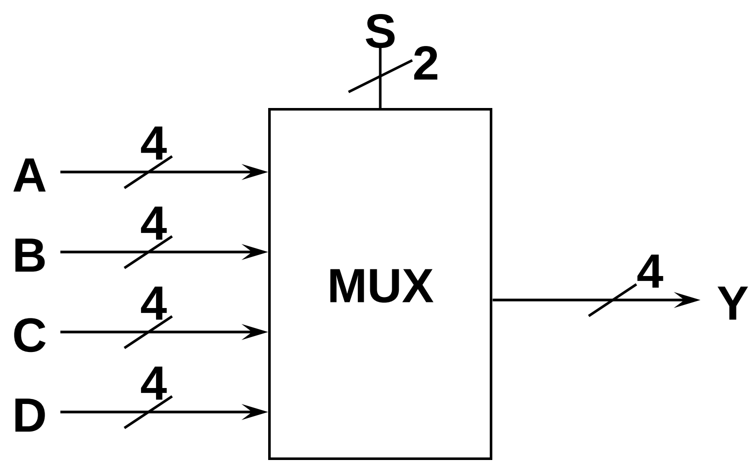
A 1-bit Selector



A 4-bit Selector

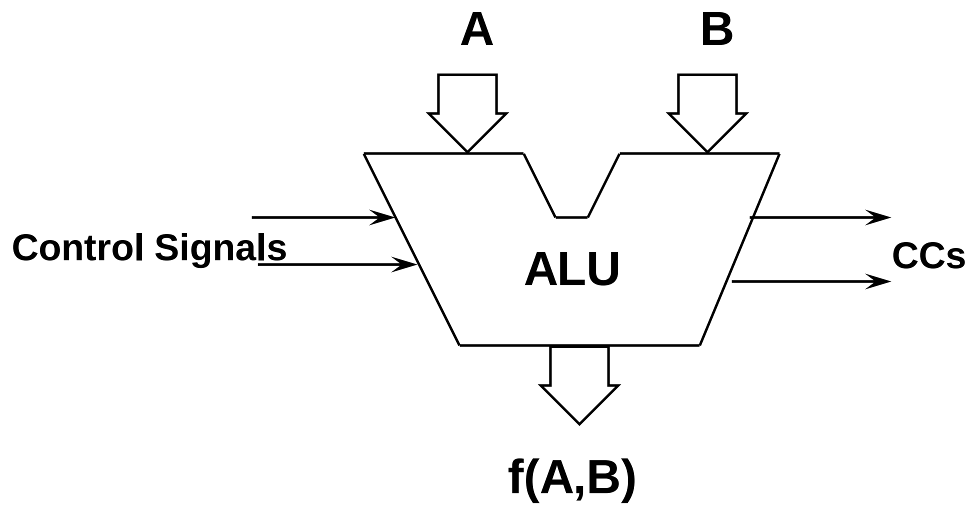






The ALU Is Next

- Logical and arithmetic operations
- Variations in
  - Base
    - Binary
    - Decimal
    - BCD
  - Implementation
    - Serial
    - Parallel
    - Pipelined



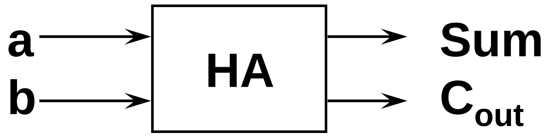
Simple Example - Binary Adder

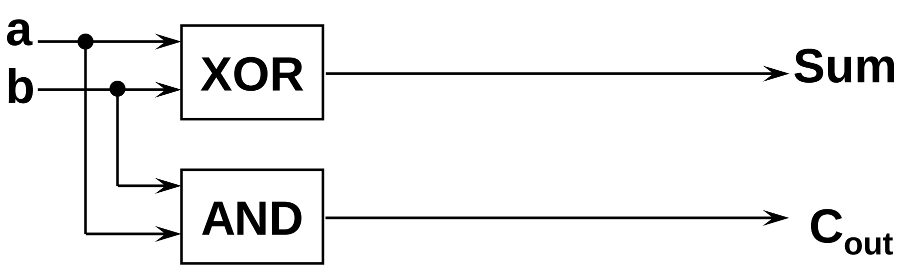
- Develop a half-adder (HA)
- Use two HA’s to build a full-adder (FA)

The Half-Adder

$$\text{Sum} = (\bar{a} \bullet b) + (a \bullet \bar{b}) = (\bar{a} + \bar{b}) \bullet (a + b)$$

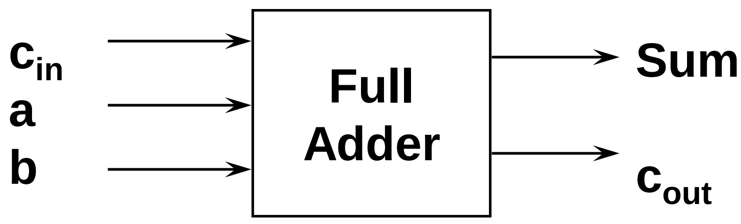
$$\text{Carry} = a \bullet b$$



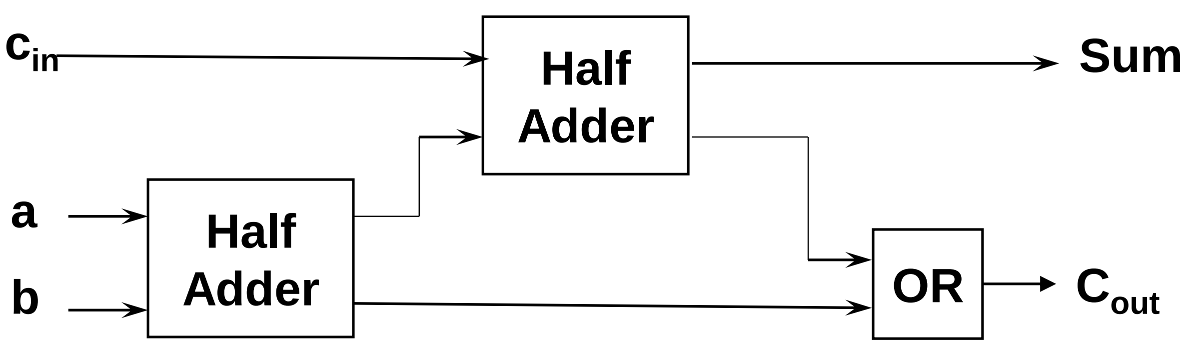


a	b	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

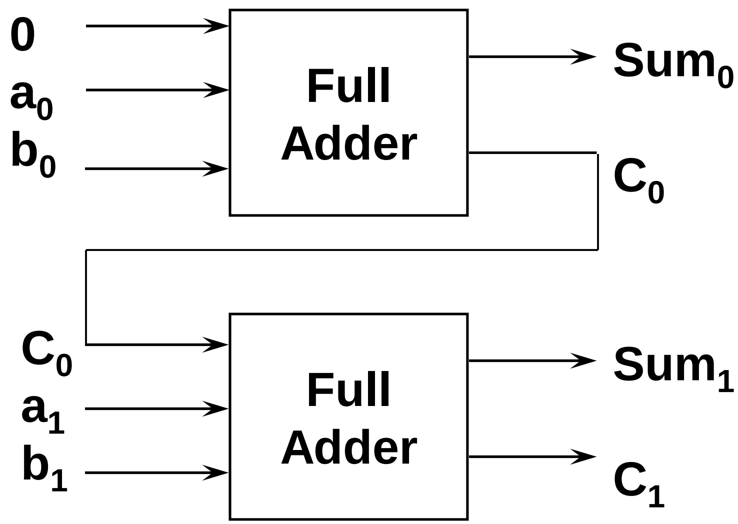
From HA to FA



a	b	c_{in}	Sum	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



Using Full Adders for Addition



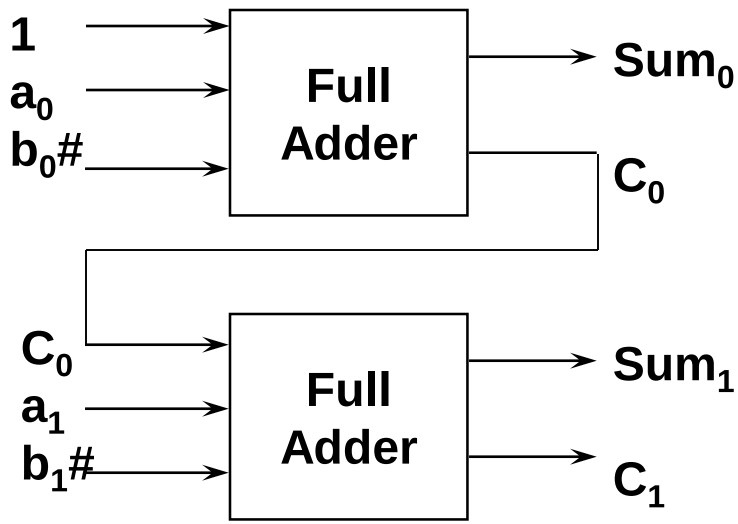
a	b	C_{in}	Sum_0	C_0
LSB				
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
MSBs		C_{n-1}	Sum_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

```
    0 0        0 0          1 1        1 1
+   1 1      + 0 1        + 0 1       +1 1
-------      -----       ------     ------
(0) 1 1    (0) 0 1      (1) 0 0    (1) 1 0
```

Note: The carry flag value after the addition represents the N+1 bit value in the result

Using Full Adders for Subtraction

Difference = a - b
= a + (-b)
= a + (~b + 1)
= a + ~b + 1



(1) 0 0	0 0	(0)1 1	1 1
- 1 1	+ 0 1	- 0 1	+1 1
0 1	(0) 0 1	1 0	(1) 1 0

a	b#	C_0	Sum	C_1
LSB				
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1
MSBs		C_{n-1}		C_n
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

Note: The carry flag value after the ~ and addition is the opposite of the subtract borrow condition

Configurable Add/Subtract ALU