Reading 08

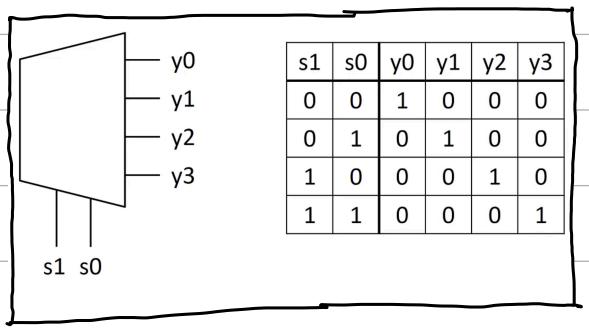
10.1 Decoders, Demuxes, and Muxes

4 way decoder

↳ 4 outputs

↳ 2 select-lines

↳ only 1 output is high (1) at a time

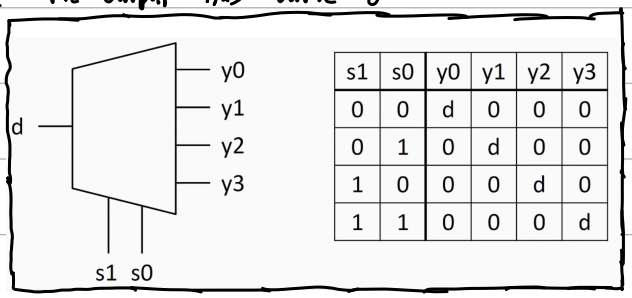


s1	s0	y0	y1	y2	y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

1-to 4 Demultiplexor (Demux)

↳ essentially same as decoder

↳ but the output has value d



s1	s0	y0	y1	y2	y3
0	0	d	0	0	0
0	1	0	d	0	0
1	0	0	0	d	0
1	1	0	0	0	d

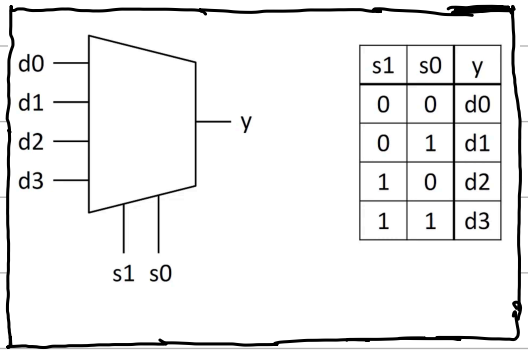
4-to-1 Multiplexor (Mux)

↳ 4 data inputs (d0, d1, d2, d3)

↳ to select lines (s0, s1)

↳ Single output (y)

↳

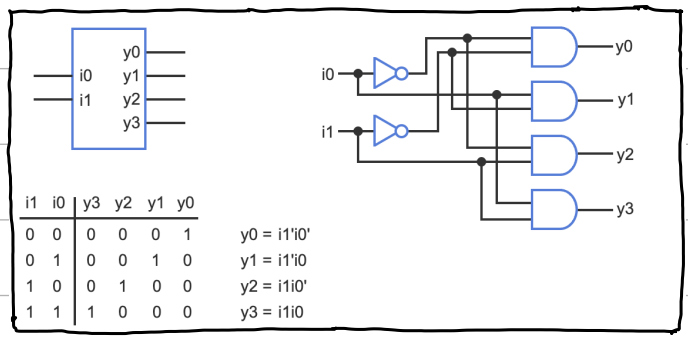


s1	s0	y
0	0	d0
0	1	d1
1	0	d2
1	1	d3

11.1 Decoders

- A *decoder* is a combinational circuit that converts N inputs to a 1 on one of 2^N outputs.
  - ↳ A 2x4 decoder ("2 to 4 decoder") converts two inputs to a 1 on exactly one of 4 outputs.

- **Ex** of decoder to circuit!

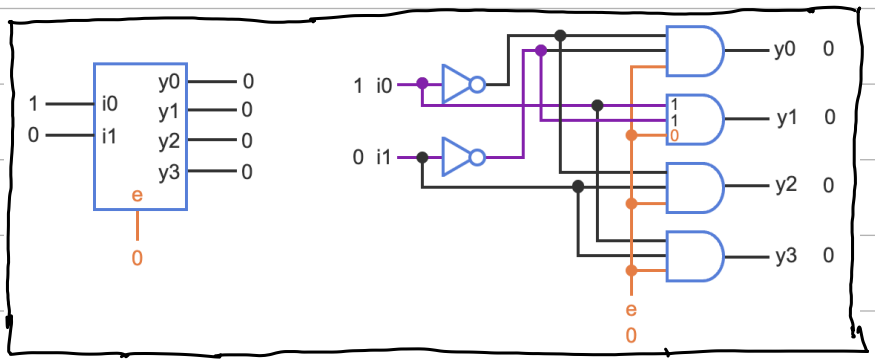


i1	i0	y3	y2	y1	y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

y0 = i1'i0'
y1 = i1'i0
y2 = i1i0'
y3 = i1i0

- Remember N inputs to 2^N outputs
- Remember each output requires 1 AND gate.
- Remember decoders do NOT require an OR gate.

- Some decoders have an additional input called *enable input* that when 0 sets all outputs to 0s, and when 1 enables the decoder's normal behavior.
  - ↳ **Ex**

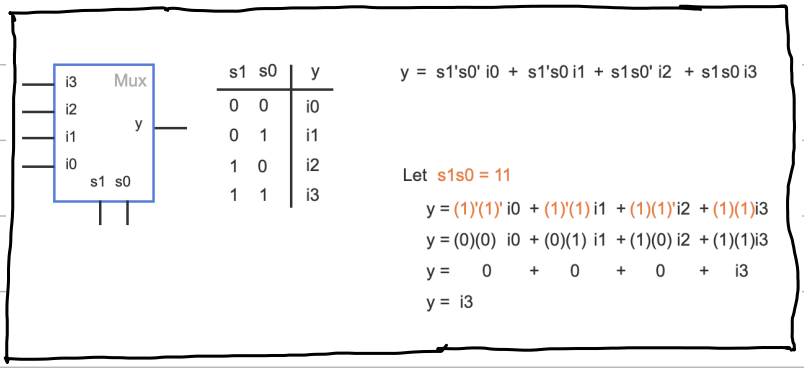


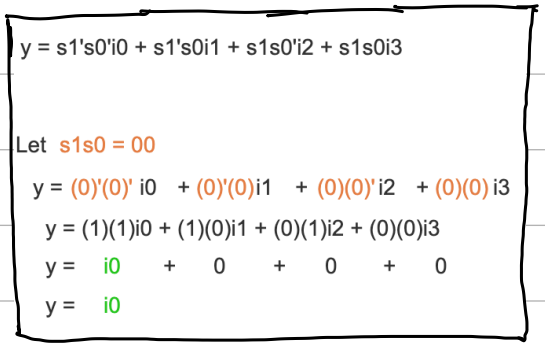
11.2 Muxes

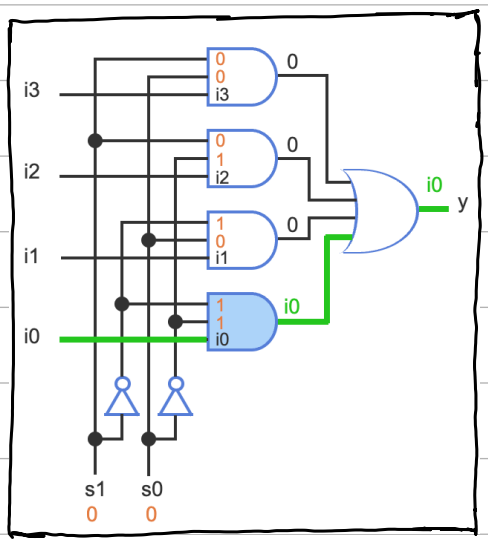
- A multiplexor is a combinational circuit that passes one of multiple data inputs through to a single output, selecting which one based on additional control inputs.
  - ↳ Mux is short for multiplexor
  - ↳ A mux's control inputs are called select lines.

- A 4x1 mux, "4 to 1" mux has 4 data inputs, one data output, and requires two select pins.

↳ Analogy: 4 lanes merge into 1 lane due to construction. A policeman (the select input) selects which one lane currently passes through by blocking the other lanes.

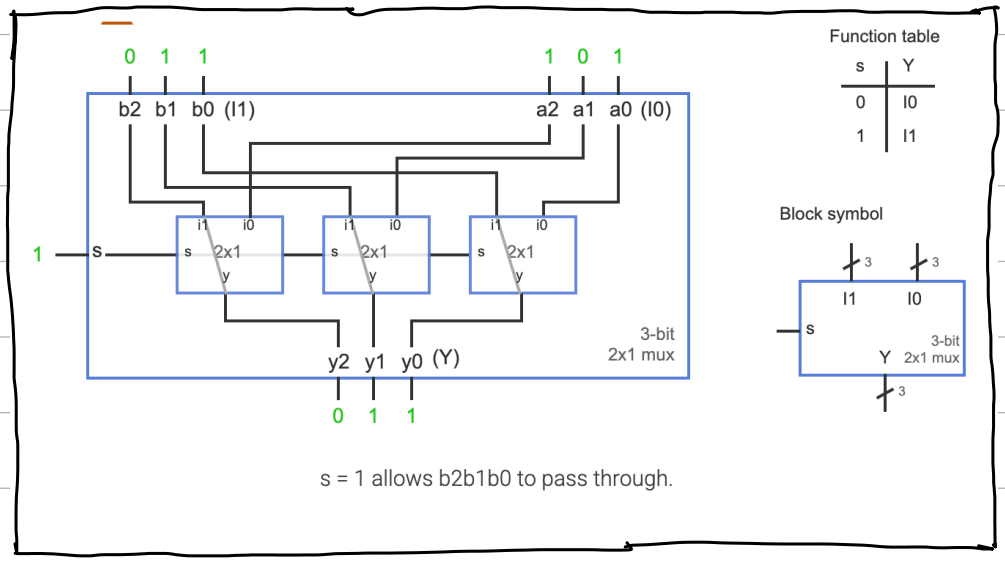






- For N data inputs, a mux requires $\log_2 N$ select inputs.
  - ↳ It also requires one AND gate per input, and an individual OR gate.

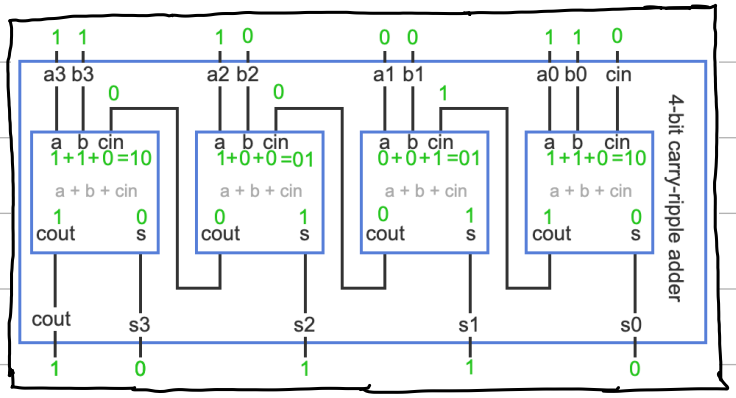
11.3 N-bit muxes

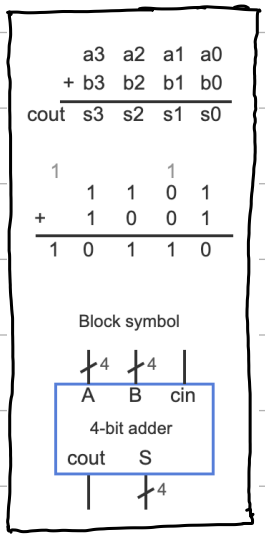


3 bit 2x1 Mux

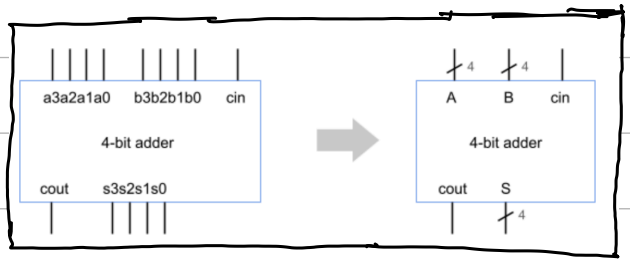
11.4 Adders

- An adder computes A + B, where A and B are N-bit numbers, such as 8-bit numbers. A carry-ripple adder mimics adding by hand, adding a digit's pair of bits and carry-in bit, and generating a sum and carry-out bit.





Notation:

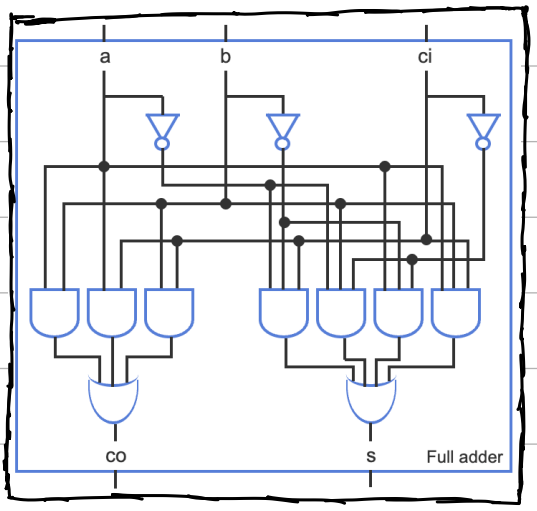


- Each digit's pair of bits and carry-in bit are added simultaneously.
  - ↳ As carry bits propagate from right to left, each digit's pair and carry-in are re-evaluated.

- A half adder is a circuit that adds two bits and generates a sum and carry-out bit.
- A full adder is a circuit that adds three bits and generates a sum and carry-out.
  - ↳ can be designed from a truth table:

ci	a	b	co	s
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

```
  0   10   11
  0    1    1
+0   +1   +1
---  ---  ---
00   10   11
```



$$co = ci'ab + cia'b + ciab' + ciab$$
$$co = ab + cia + cib$$
$$s = ci'a'b + ci'ab' + cia'b' + ciab$$

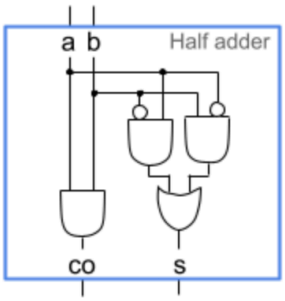
- An N bit carry-ripple adder is constructed with N full adders.

- An incrementer adds 1 to a number.
- A half adder adds 2 bits (a,b) and is sufficient for an incrementer.
  - ↳ half adder truth table equations and circuits:

a	b	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

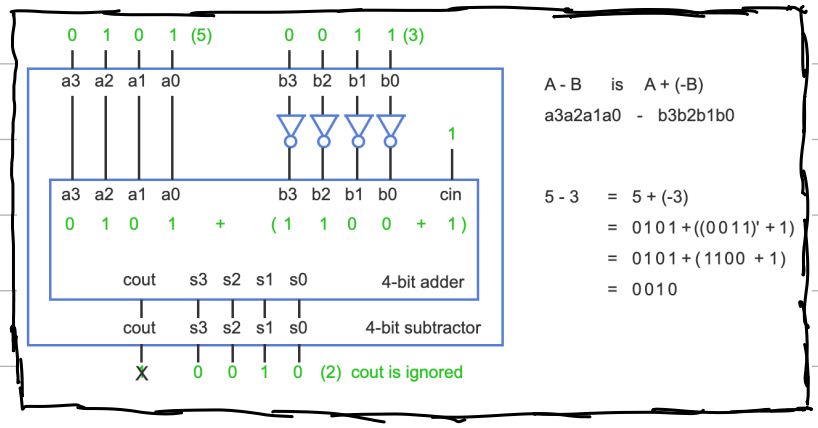
co = ab

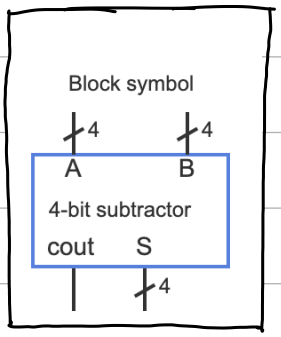
s = a'b + ab'



11.5 Subtractors

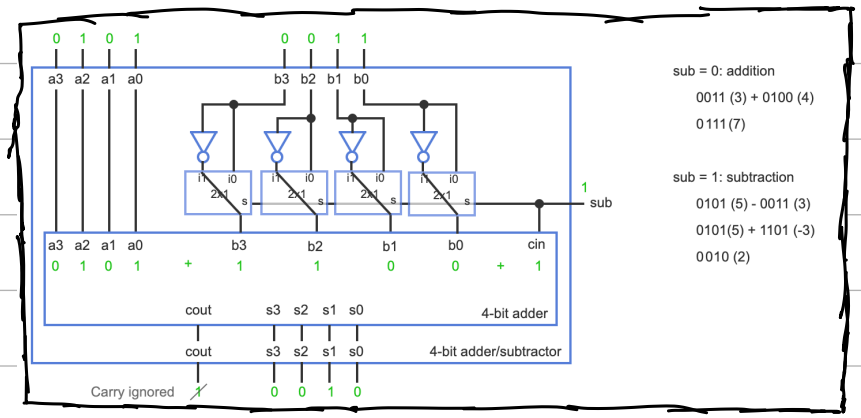
- A subtractor computes A - B, where A and B are N-bit numbers, such as 8-bit numbers.
  - ↳ If numbers are represented using two's complement representation, a subtractor can be built using an adder.
  - ↳ Inverting B's bits and setting the adder's carry-in to 1 add B's complement to A.

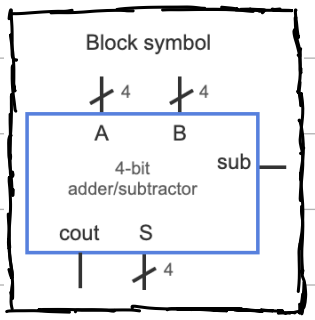




- Because two's complement representation performs subtraction by complementing and adding, a single adder circuit can perform either addition or subtraction, saving circuit size.

- A 4 bit adder/subtractor :





- Configuring the adder/subtractor to perform the following operation : 7 - 2

  a3a2a1a0 = 0111

  b3b2b1b0 = 0010

  sub = 1