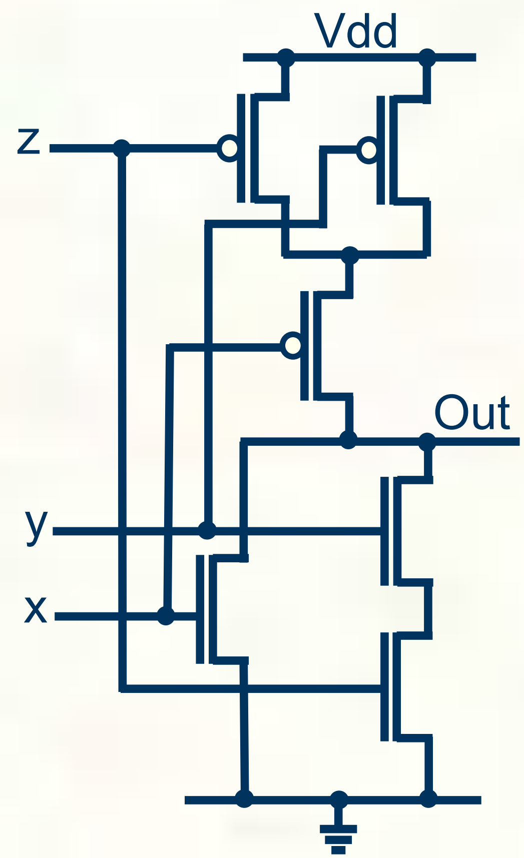
Combinational Logic

Multi-input gates - (x+(y*z))’

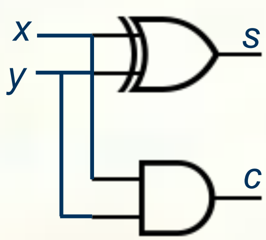
x	*y*	*z*	*out*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0



Very easy with inverting gates

Half Adder

- 1-bit addition ($x + y$)
  - outputs: sum s and carry c
  - $s = xy' + x'y$ (x XOR y or $x \oplus y$)
  - $c = xy$



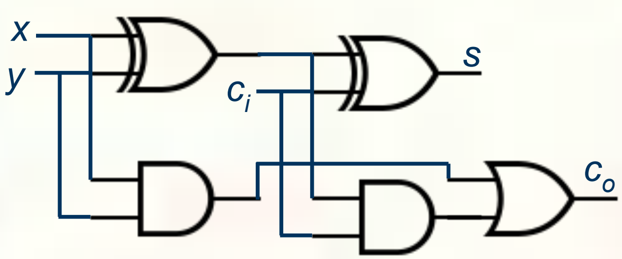
x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full 1-bit adder

- Need a carry in c_i

$s = x'yc_i' + xy'c_i' + x'y'c_i + xyc_i$
$\quad = c_i'(x'y+xy') + c_i\,(xy+x'y')$
$\quad = c_i'(x'y+xy') + c_i\,(x'y+xy')'$
$\quad = c_i \oplus (x \oplus y)$

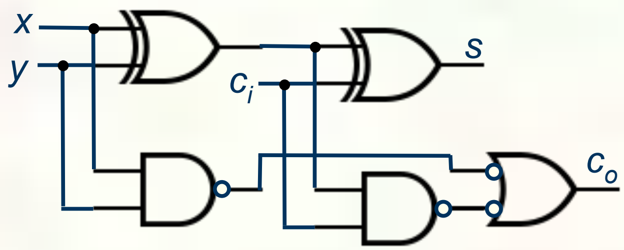
$c_o = x'yc_i + xy'c_i + xyc_i' + xyc_i$
$\quad = c_i\,(x'y+xy') + xy(c_i'+c_i)$
$\quad = c_i\,(x \oplus y) + xy$

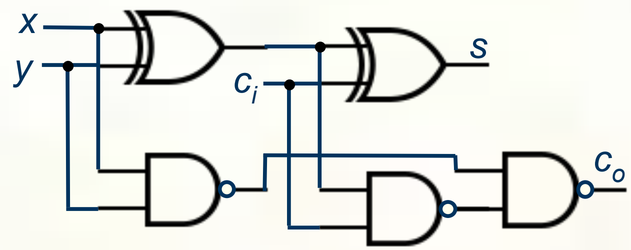


x	y	c_i	s	c_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

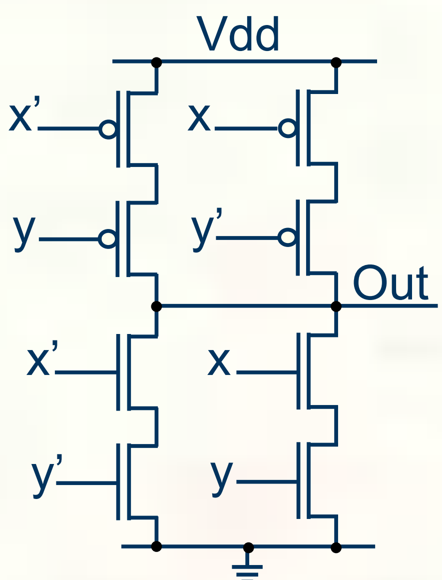
CMOS adder

For the carry, we can first use De Morgan's laws



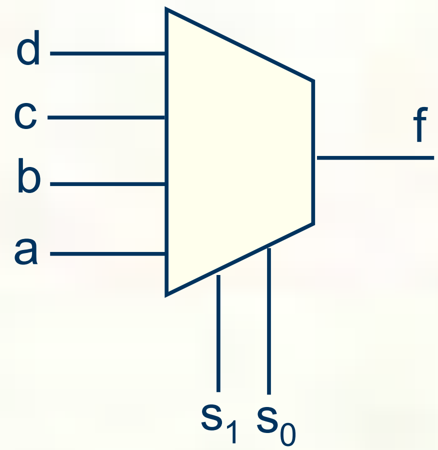


For the **xor**, recall that a **xor** b = ab' + a'b and (a **xor** b)' = ab + a'b'

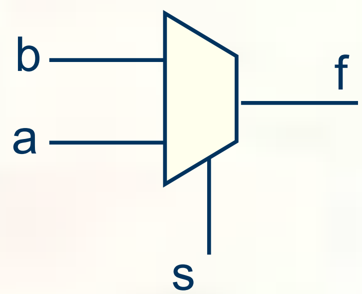


Multiplexor

- A multiplexor (selector or mux) selects one of *n* inputs to be output



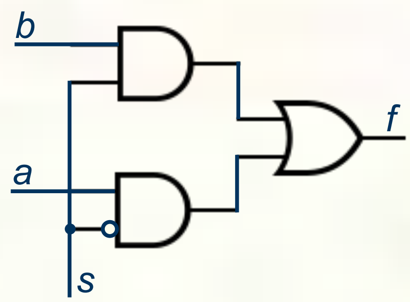
- Let's make it easy and start with a 2-input mux
  - when s = 0, f = a
  - when s = 1, f = b



2 input Multiplexor design

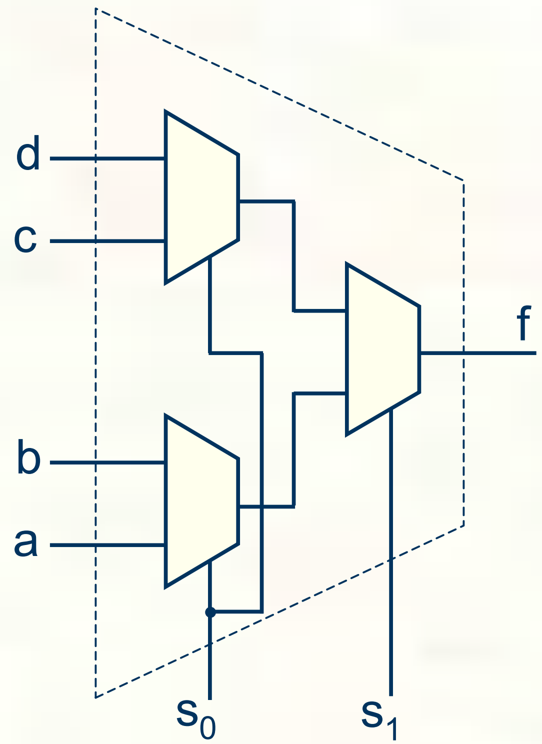
s	*a*	*b*	*f*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$f = s'ab' + s'ab + sa'b + sab$$
$$= s'a(b'+b) + sb(a'+a)$$
$$= s'a + sb$$



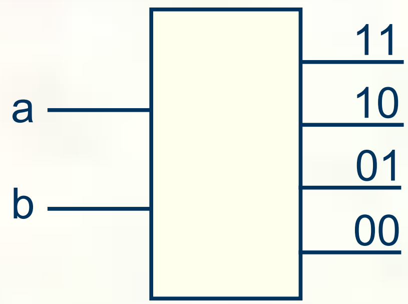
Making bigger muxes

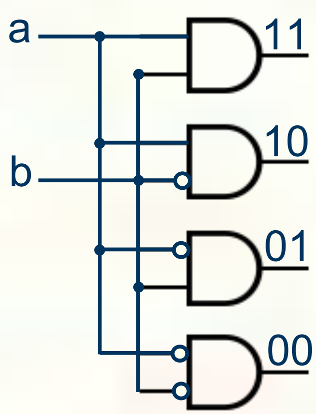
You can design them directly, or just glue together smaller muxes:



Decoders

- Decode an *n*-bit input to set output line 2^n-1





a	*b*	*00*	*01*	*10*	*11*
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Programmable Logic Array (PLA)

- Since all Boolean functions can be expressed in sum of products form, we can implement a set of n input functions systematically in hardware using a PLA