2-bit adder: add 2 2-bit numbers

Syntax (consistent w/ Verilog)

a group of bits ⇒ "bus" (not an array)

will have a LSB & MSB

represent as A[highest:lowest] left → right

MSB ↑ LSB ↑

so 2-bit bus is A[1:0] (just like binary number)

bit 0 ⇒ LSB

1 ⇒ MSB

MSB on left ↖ ↑ LSB on right

5-bit bus is A[4:0]

you can use A[0:4] but then LSB is on left

let A, B be 2-bit buses: A[1:0], B[1:0]

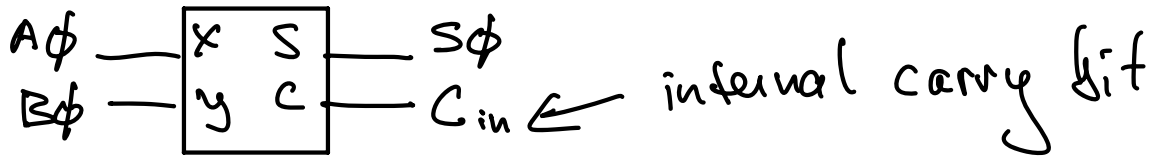
LSB of A is A0, MSB of A is A1, etc for B

construct S = A + B arithmetic

conceptually: in decimal we add digits, starting with LD (D = digit) and carrying

$$\begin{array}{r} {\scriptstyle 1}\ {\scriptstyle 1} \\ 75 \\ +\ 26 \\ \hline 101 \end{array}$$

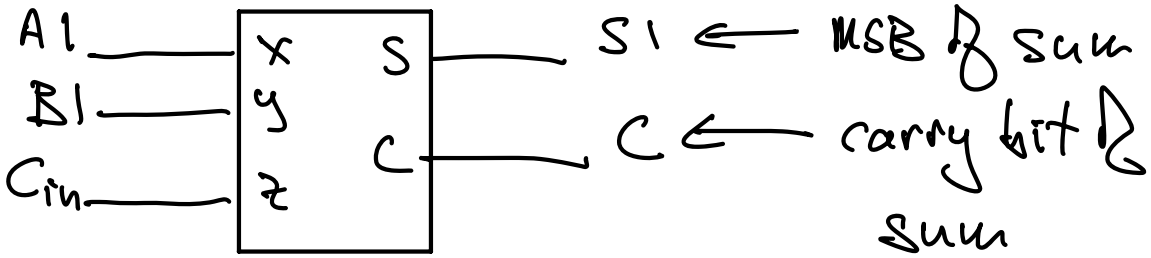
for A+B 1st add LSBs $A\phi + B\phi$



for A1 & B1 construct truth table

A1	B1	Cin	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	2
0	0	1	1
0	1	1	2
1	0	1	2
1	1	1	3

Sum is 2 bits



x	y	Cin	S	C
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$S = \bar{x}y\bar{C}_{in} + x\bar{y}\bar{C}_{in} + \bar{x}\bar{y}C_{in} + xyC_{in}$$

$$= \underbrace{(\bar{x}y + x\bar{y})}_{x \oplus y}\bar{C}_{in} + \underbrace{(\bar{x}\bar{y} + xy)}_{\overline{x \oplus y}}C_{in}$$

$$= (x \oplus y)\bar{C}_{in} + \overline{(x \oplus y)}C_{in}$$

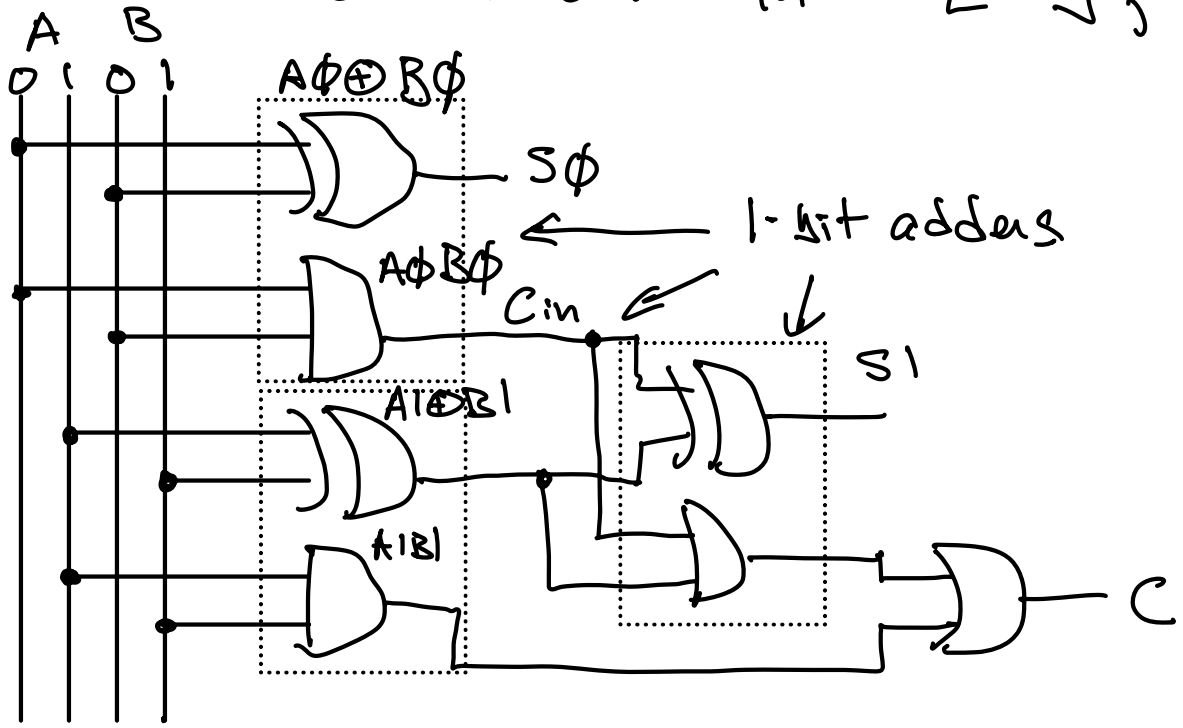
$$= x \oplus y \oplus C_{in}$$

$$C = xy\bar{C}_{in} + \bar{x}yC_{in} + x\bar{y}C_{in} + xyC_{in}$$

$$= xy(\bar{C}_{in} + C_{in}) + (\bar{x}y + x\bar{y})C_{in}$$

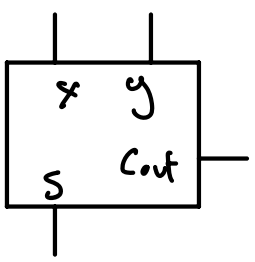
$$= xy + (x \oplus y)C_{in}$$

now can draw network for $S[1:0]$, C

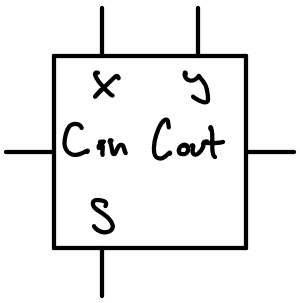


another way to show it:

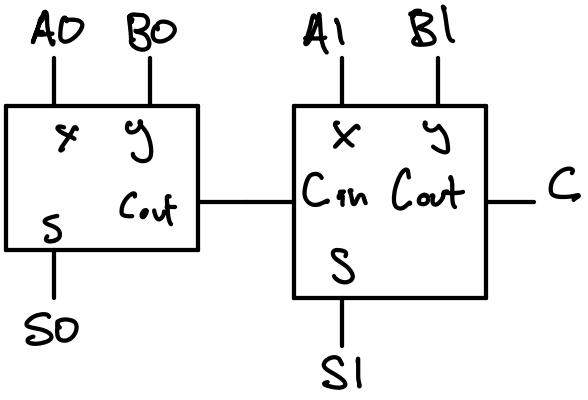
"half" adder:



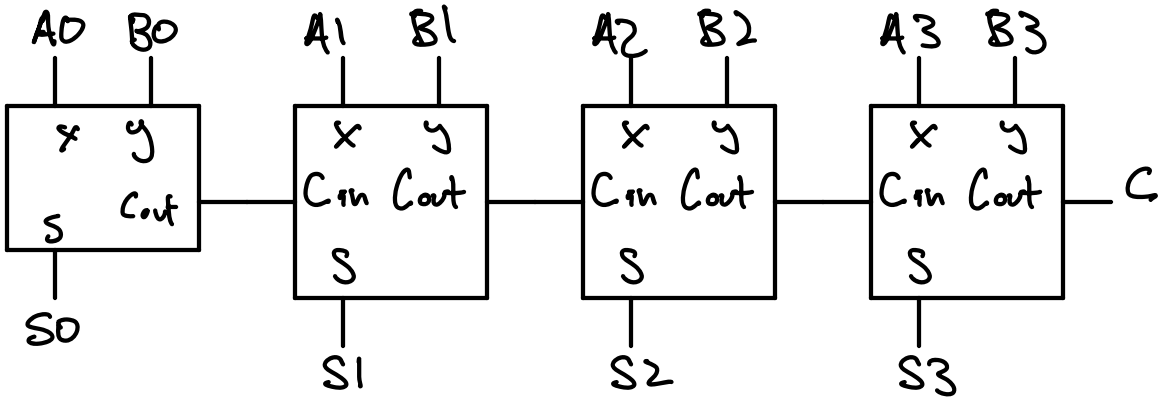
"full" adder:



Then 2-bit adder:



How would you construct 4-bit adder?



Can construct n-bit adder w/ 1 half, n-1 full adders

Caveat: last full adder has to wait for previous adders to finish

⇒ this is an example of "sequential" logic

⇒ there are ways to construct an adder that is not sequential, but has more parallelism

parallel logic would involve more gates

=> often the case that #gates N * total time to complete circuit T is ~ constant

=> can decrease T by increasing N

or " N " " T

digital uncertainty principle!