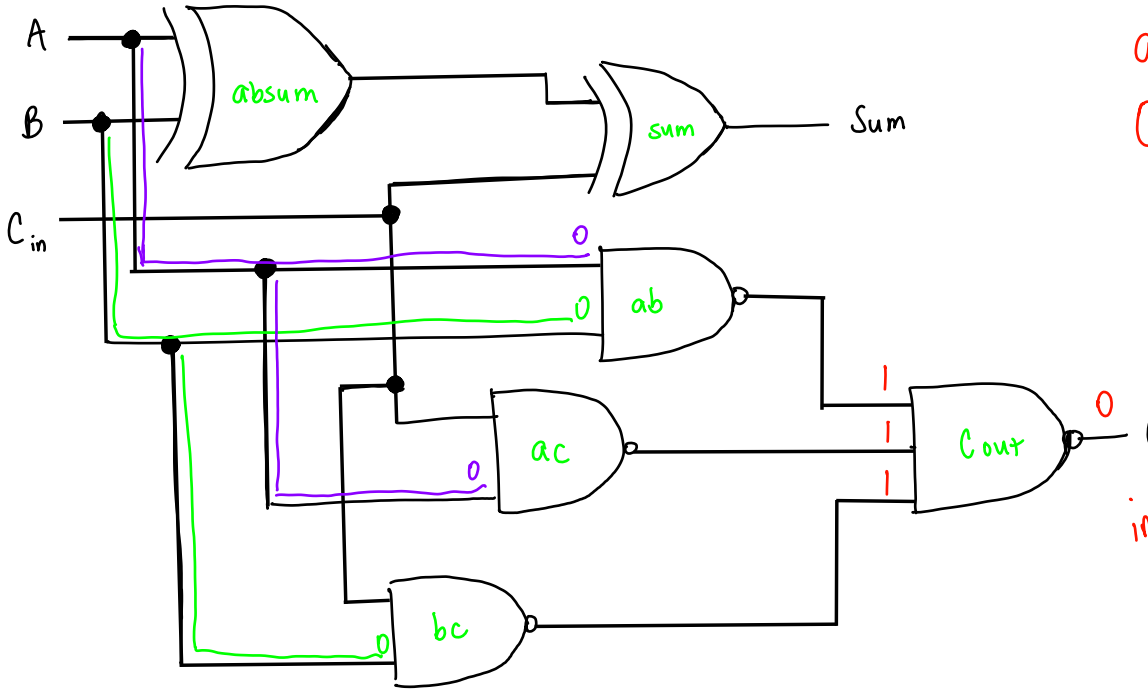
CASES BASED ON INPUT

CASE: 000

as long as there is at least one 0 input, output of nand is 1



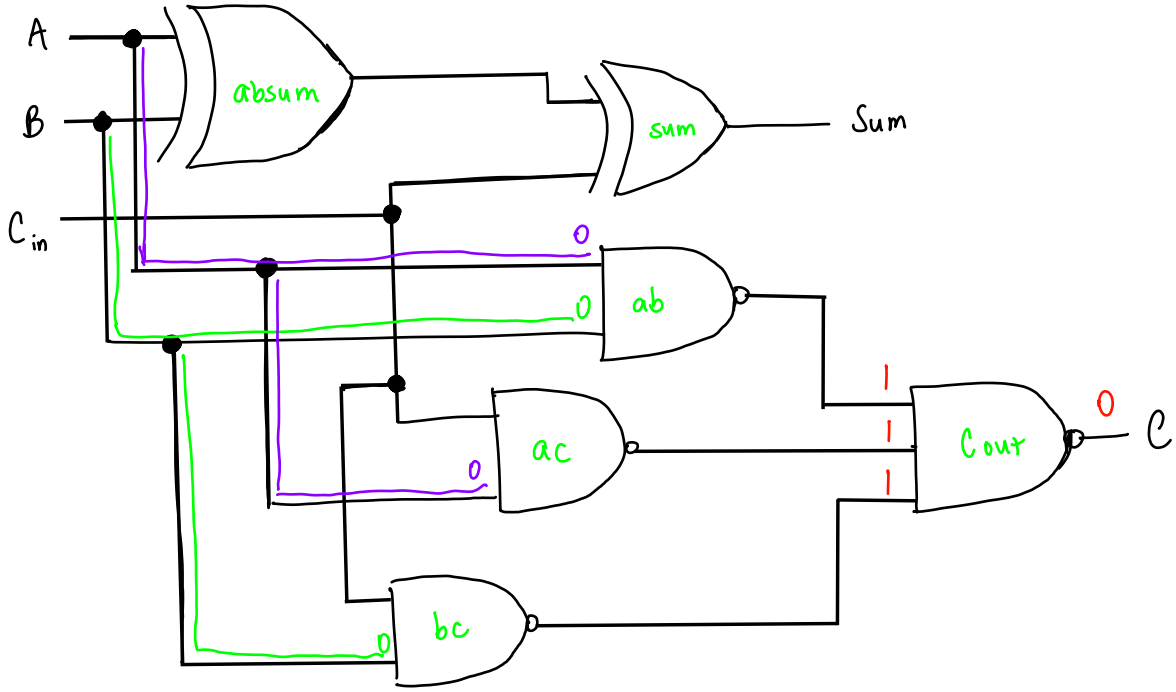
immediate 0, w/o counting for carry

This also applies to CASE: 001

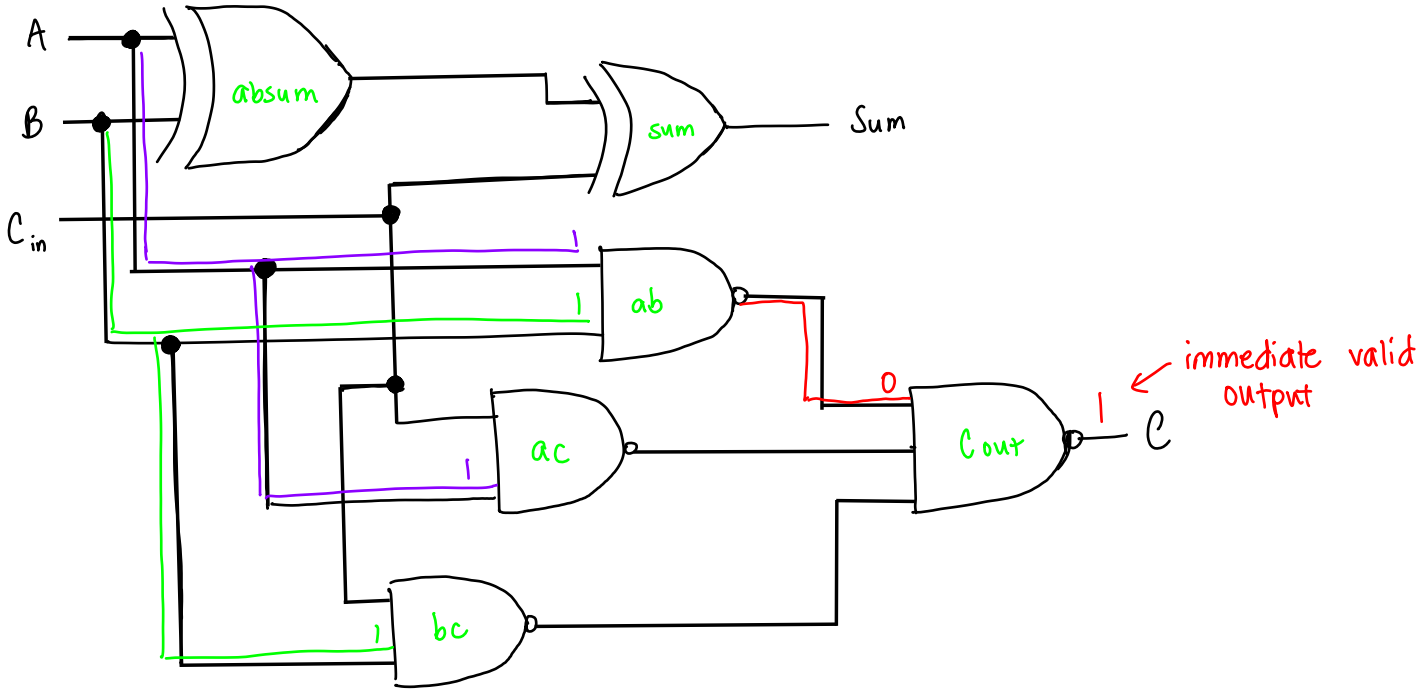
whether carry in is 0 or 1, it does not matter.

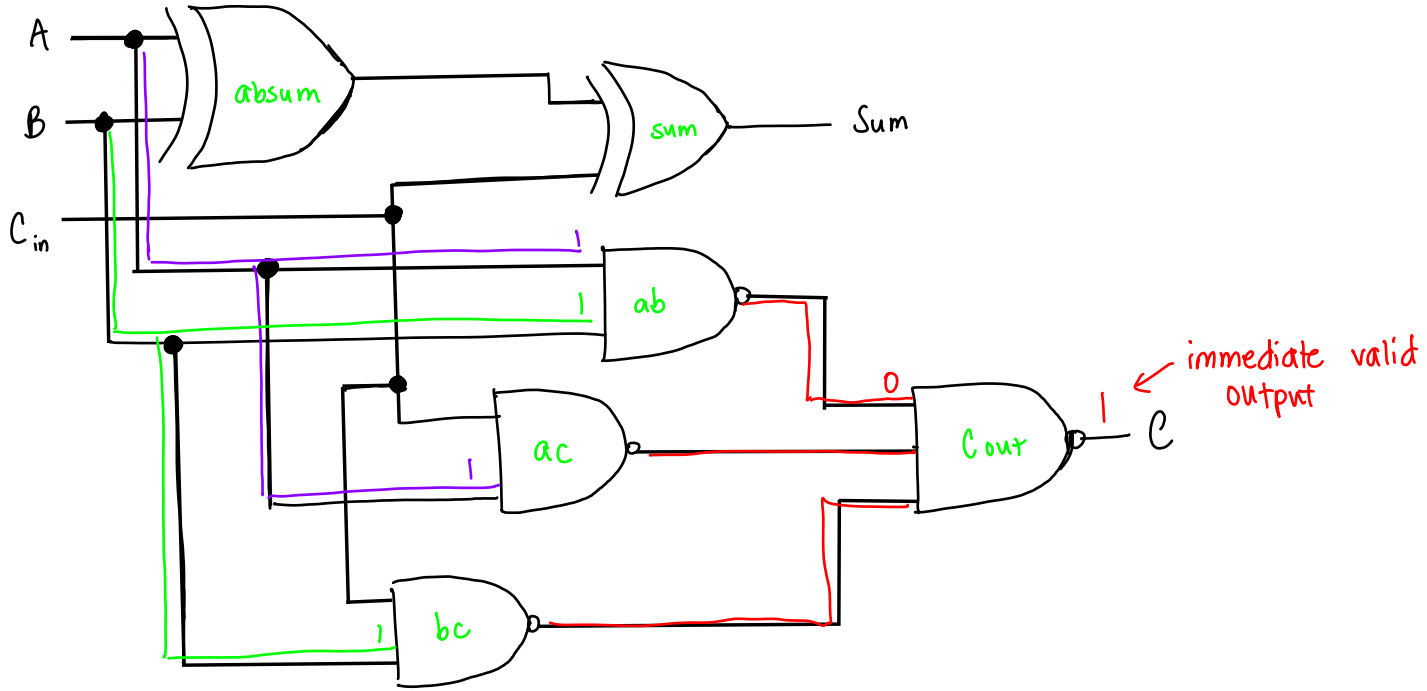
Carry out will remain 0.

automatic and no delay.



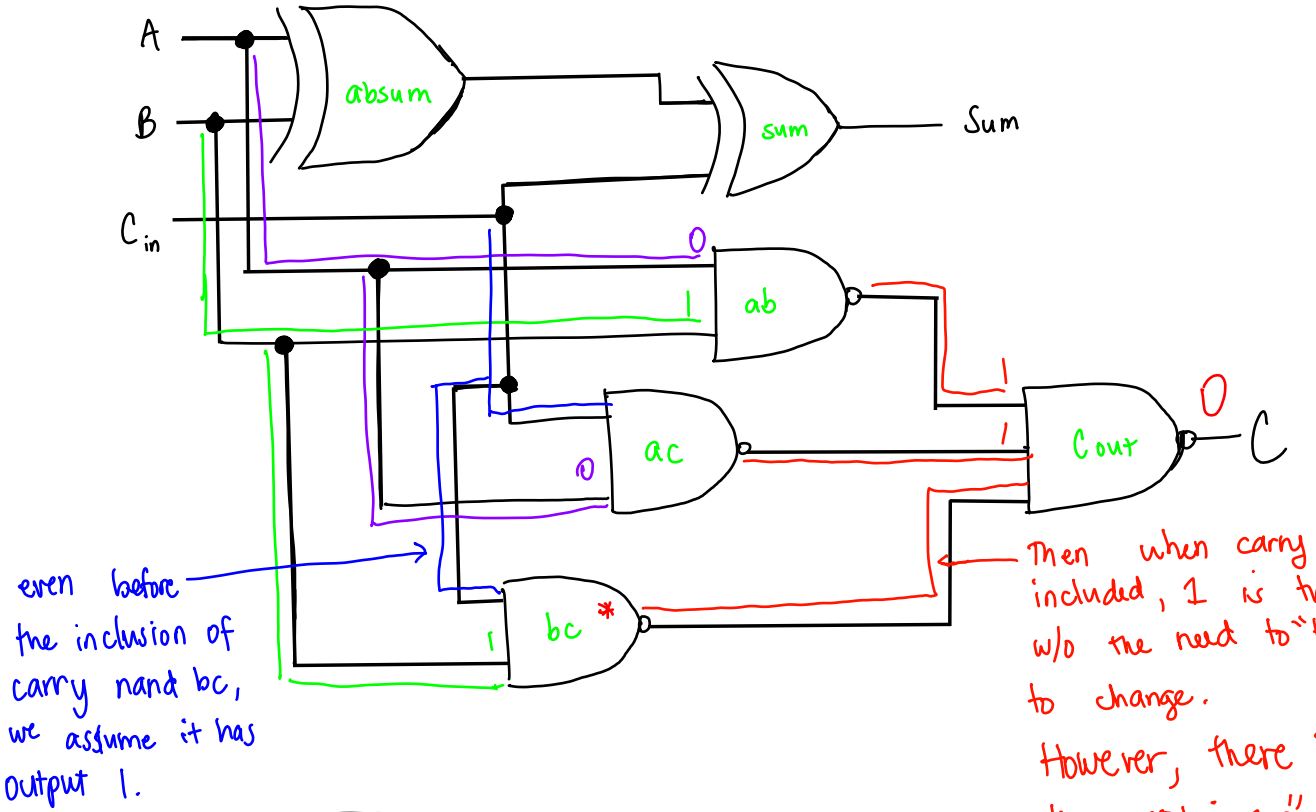
CASE 110 and 111





CASE 010 and 100

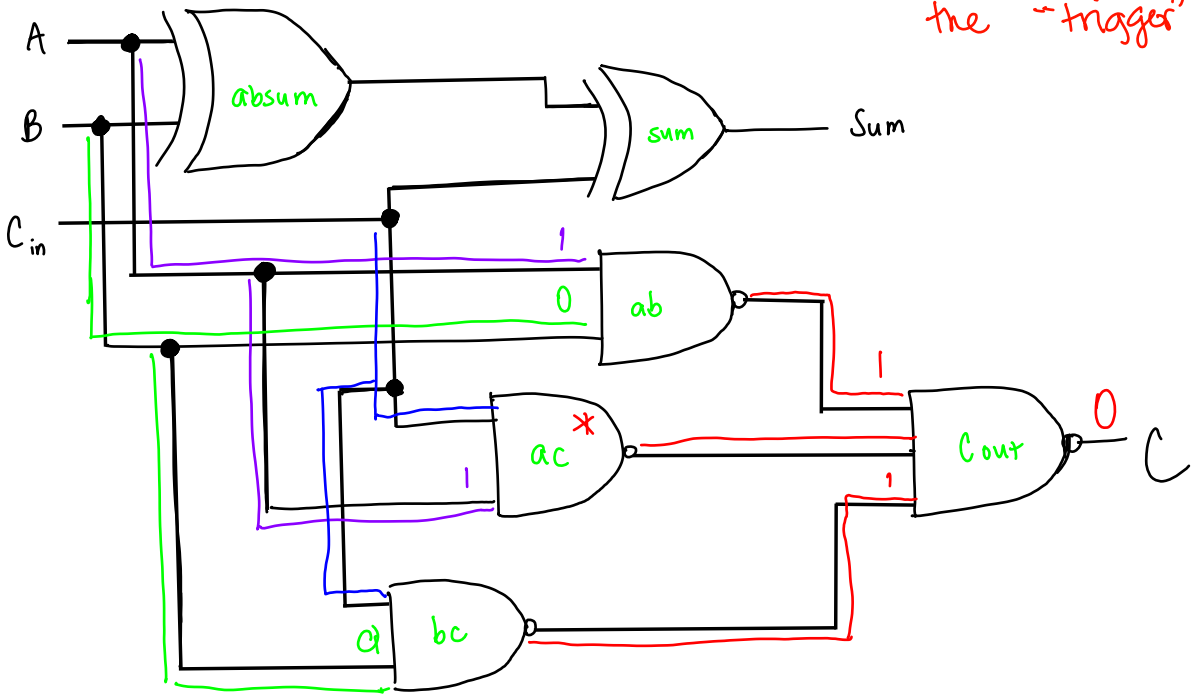
(working under the assumption of a "phantom 1", or that the natural state of a nand is 1)



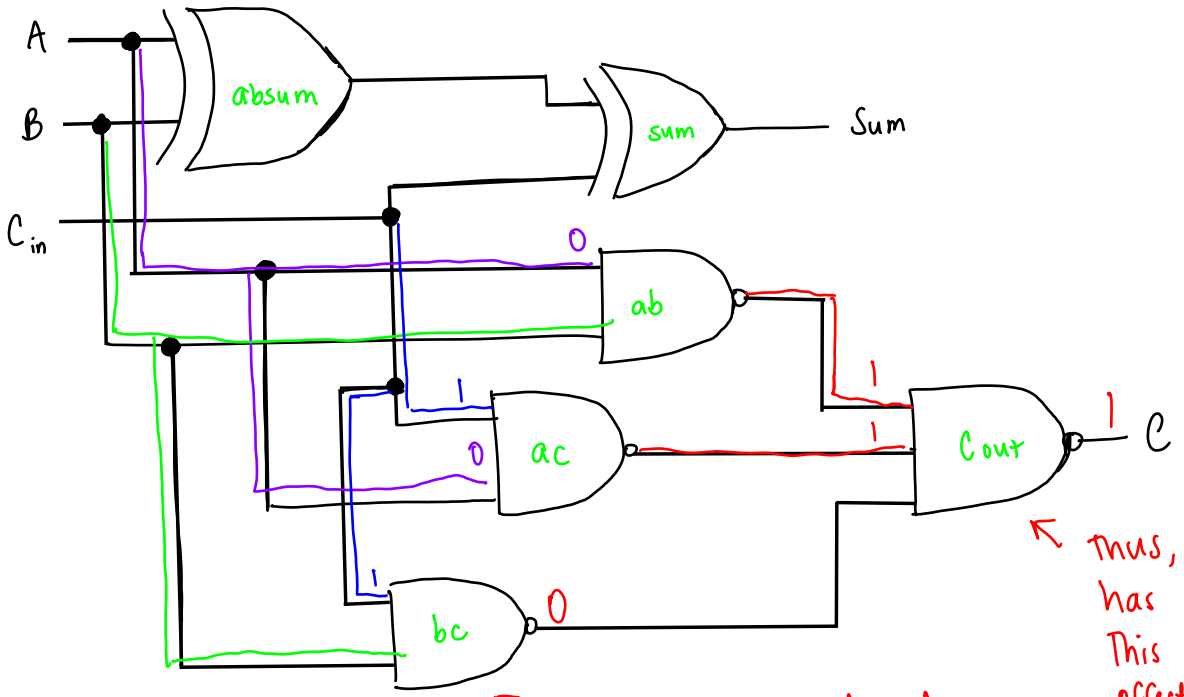
even before the inclusion of carry nand bc, we assume it has output 1.

Then when carry in is finaly included, 1 is the valid output w/o the need to "trigger" bc nand gate to change.

However, there is still a bit of delay, including the "trigger" for Cout nand.



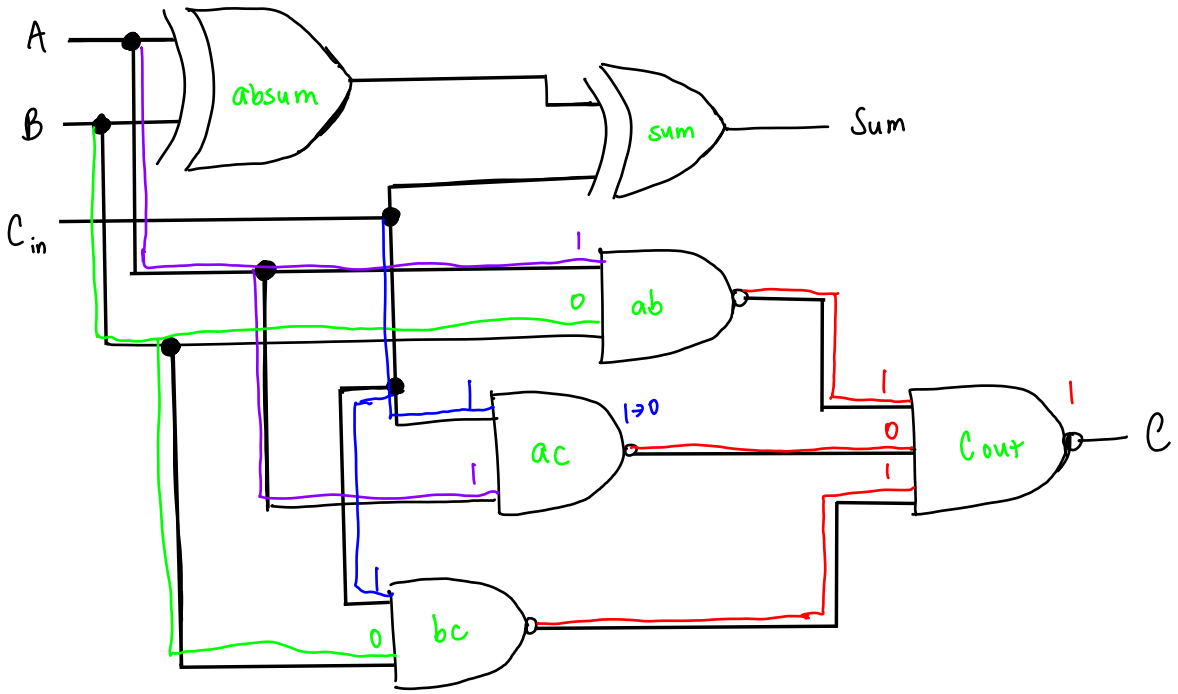
CASE 011

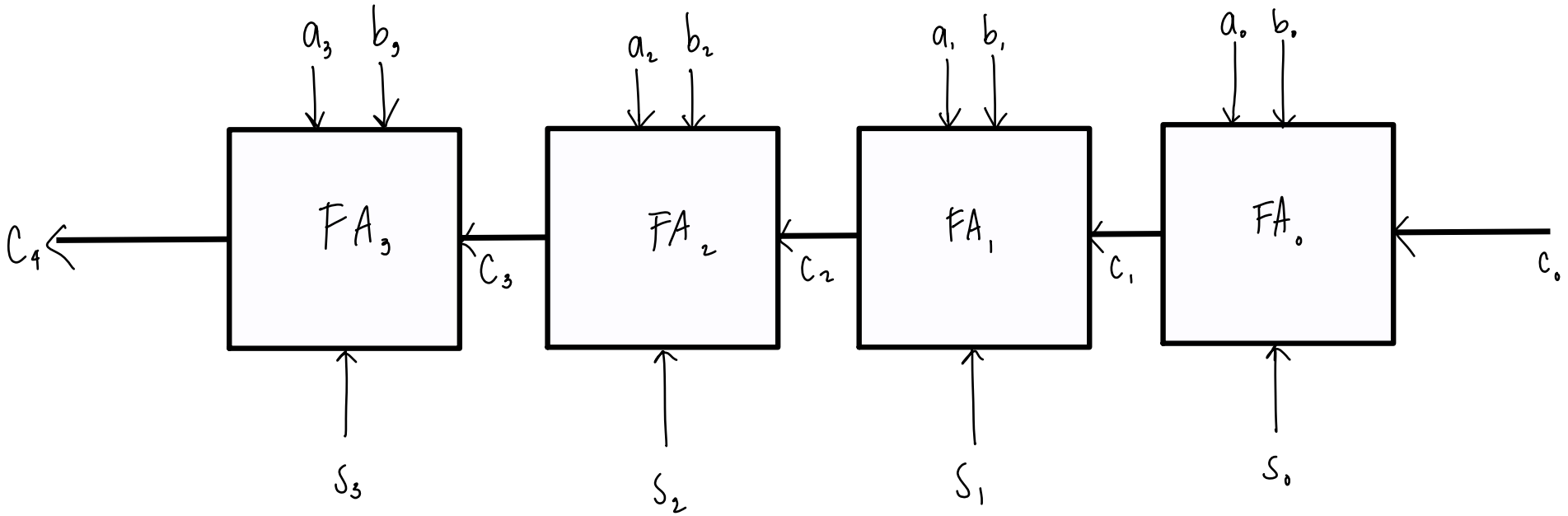


↖ needs carry in and needs to be "triggered" from 1 to 0

↖ thus, Cout depends or has to wait for nand bc. This delay will have the ripple adder effect

CASE 101





Parallel Adder → the addition of all bits is happening parallelly; FAs are connected parallelly

↳ Assumption: application of input bits ⇒ immediate availability of all sum bit and carry bit at the output.

Reality: valid output is actually available AFTER CERTAIN DELAY

↓

PROPAGATION DELAY = t_p ($ns \rightarrow ms$)

- amount of time for output (in this case) to be valid

(CSC 157.01)

Delay depends on the propagation delays of the internal circuit of that block and the individual logic gates.