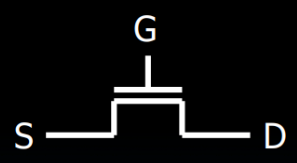
Transistors

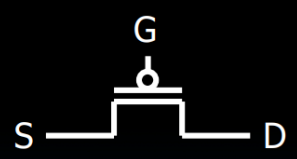
Three terminals : Drain ,Gate, Source

- **Three terminals: Drain, Gate, Source**
  - Switch action: Dan Garcia Says
    if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals

To remember:
n ("normal")
p (has a circle, like the top part of P itself)



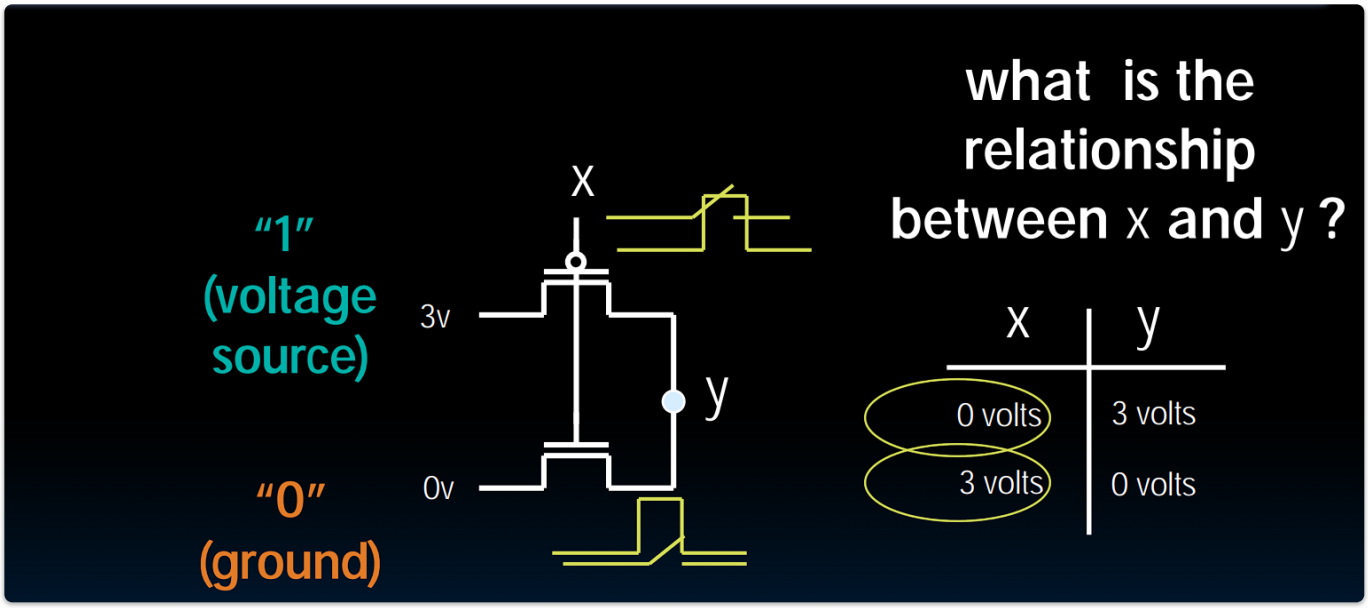
n-channel
open when voltage at G is low
closes when:
voltage(G) > voltage (S) + ε

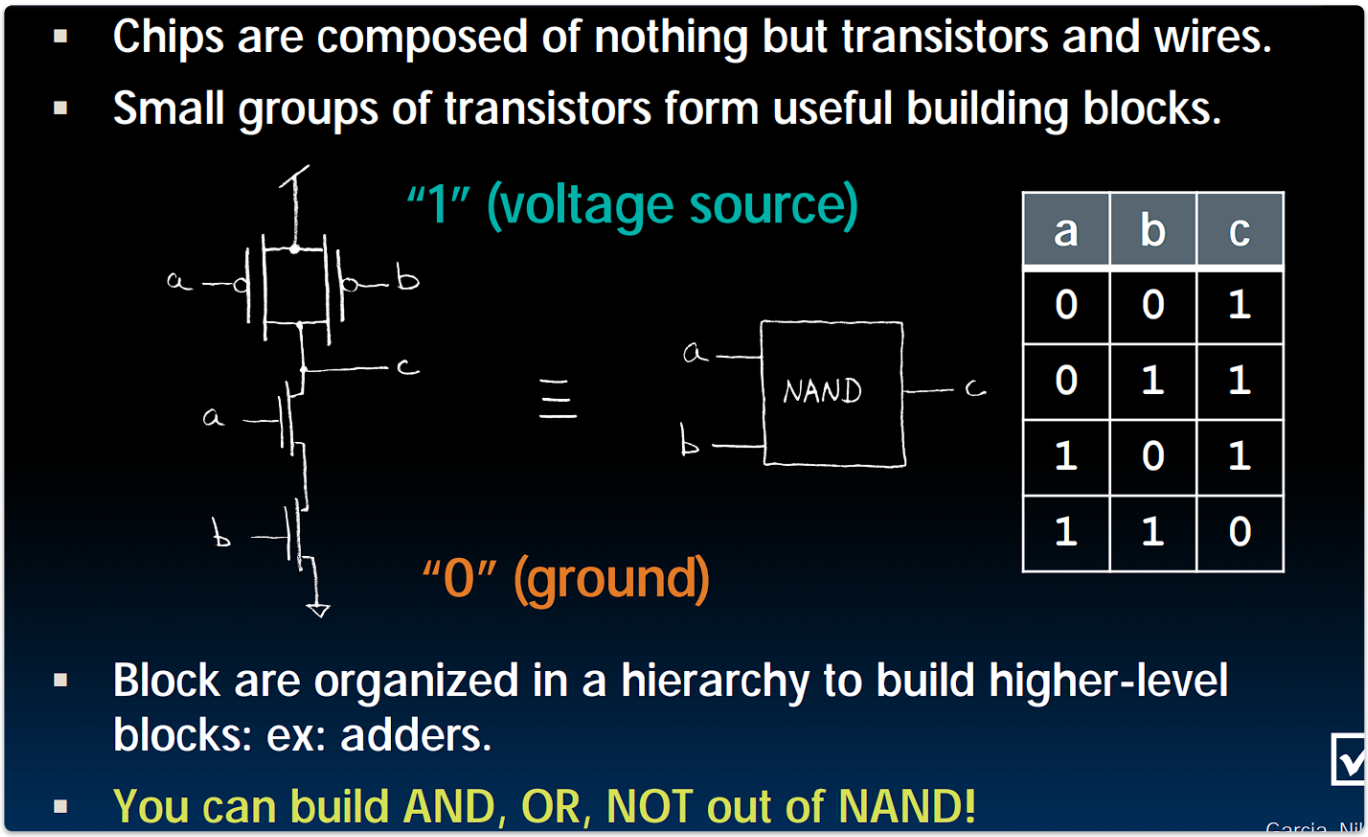


p-channel
closed when voltage at G is low
opens when:
voltage(G) > voltage (S) + ε

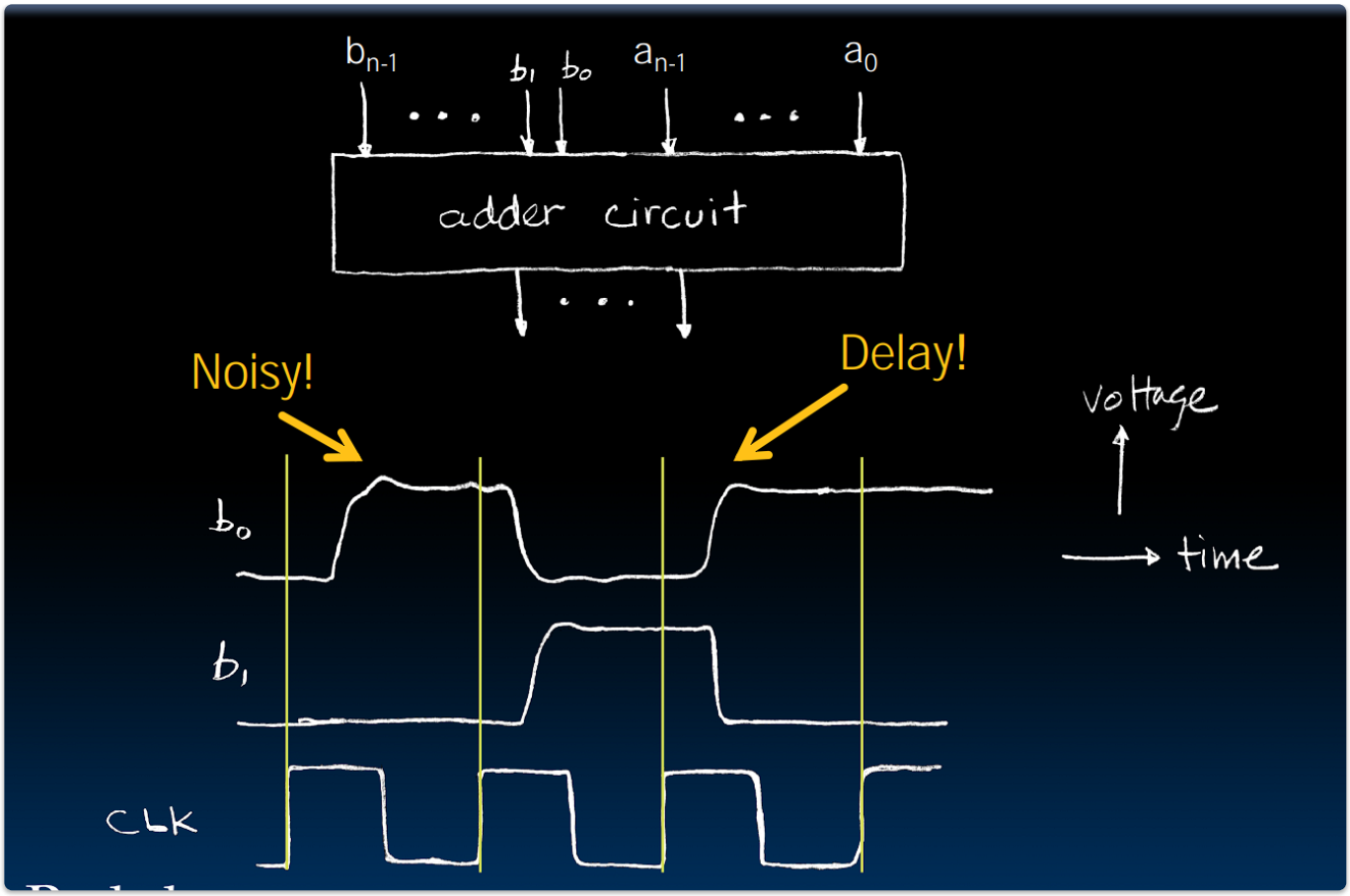
G LOW

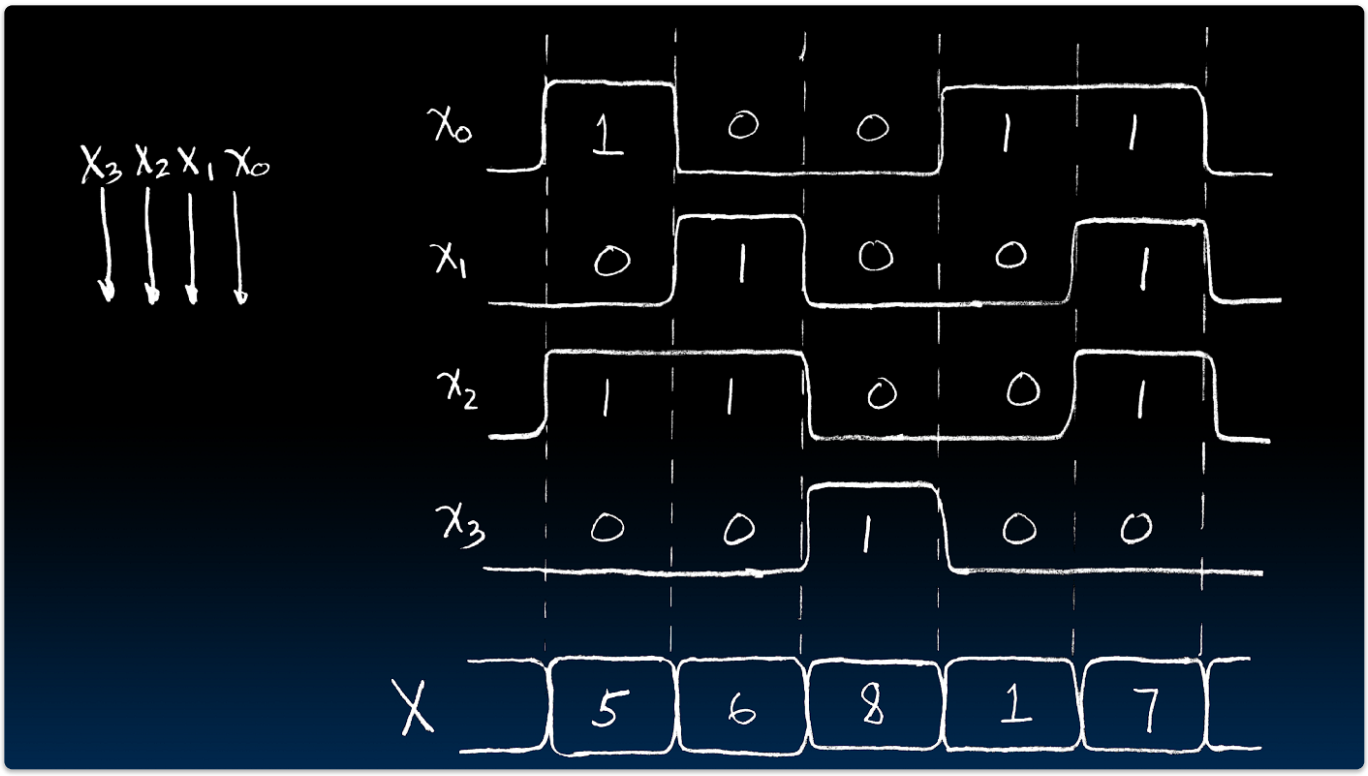
G HIGH



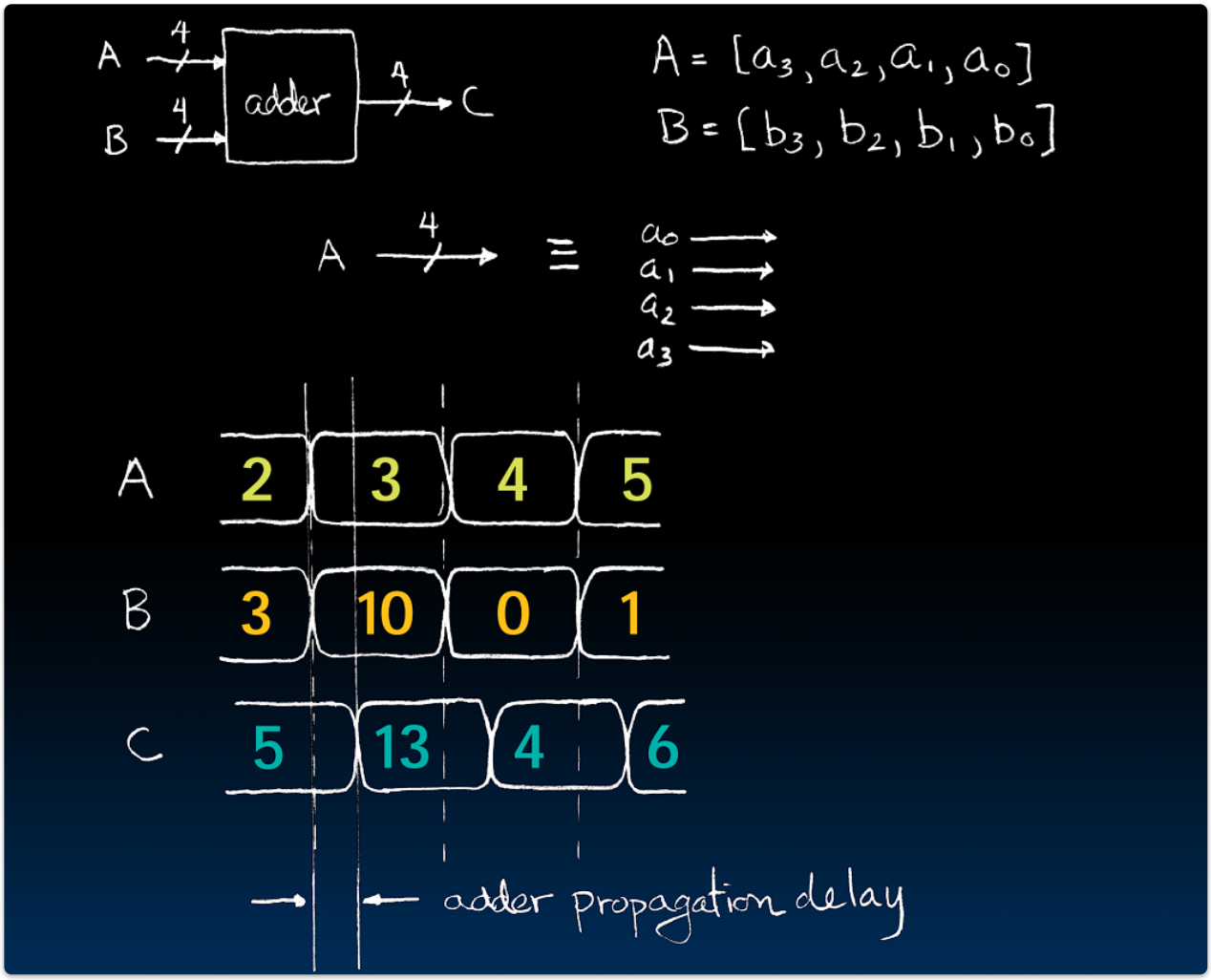


Signals and Waveform

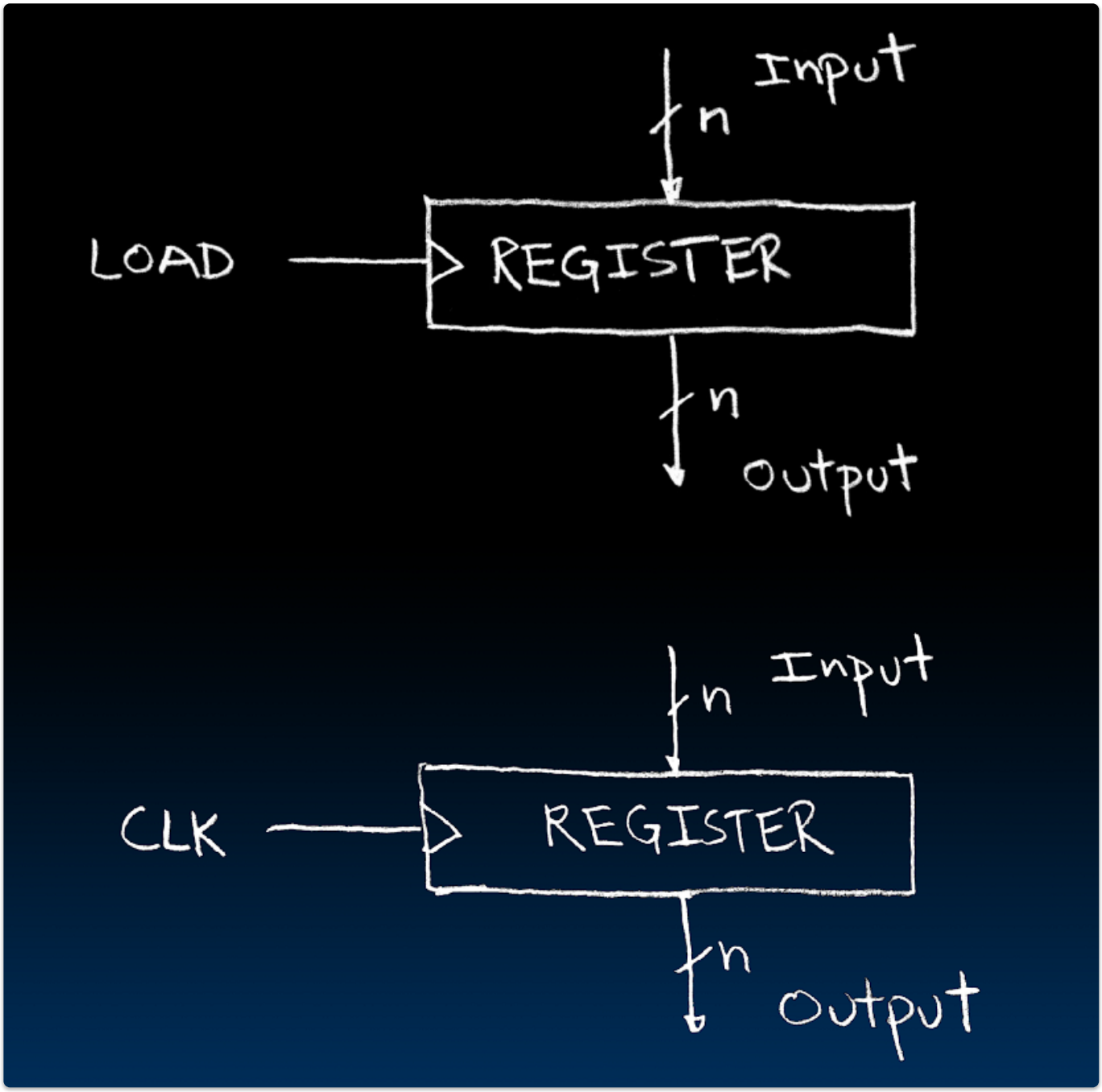




- A nibble adder which treat X_0 as the LSB and X_3 as MSB, thus gives the X below.


A
B
4
4
adder
4
C
A = [a3, a2, a1, a0]
B = [b3, b2, b1, b0]
A
4
≡
a0
a1
a2
a3
A
2
3
4
5
B
3
10
0
1
C
5
13
4
6
adder propagation delay

Register



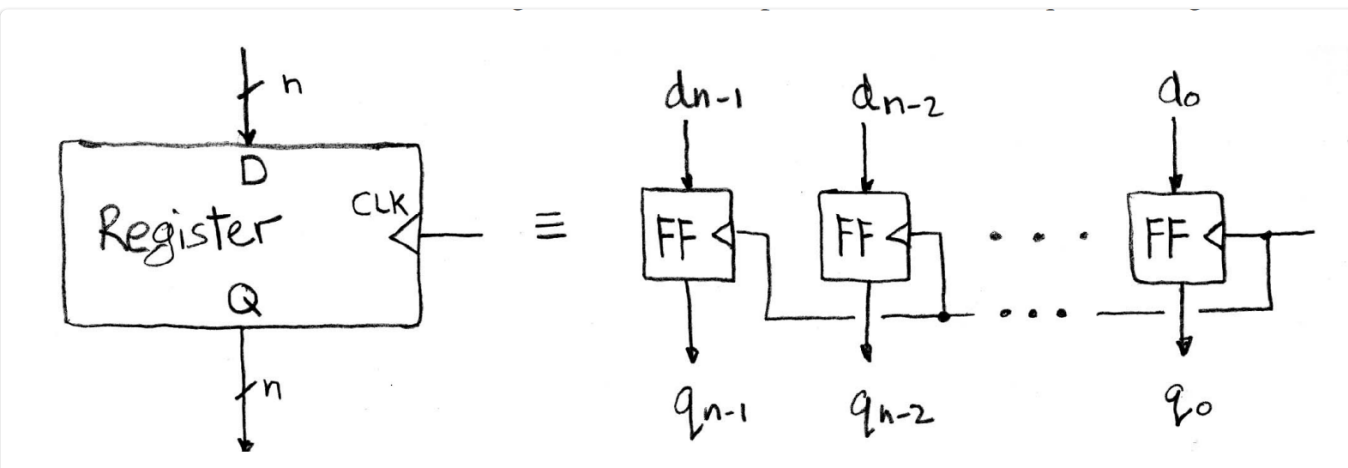
clk enables the register to store the input.

Clock control pulse of our circuits.

- Clocks control pulse of our circuits
- Voltages are analog, quantized to 0/1
- Circuit delays are fact of life
- Two types of circuits:
  - Stateless Combinational Logic (&,|,~)
  - State circuits (e.g., registers)

Register Details: Flip-Flops

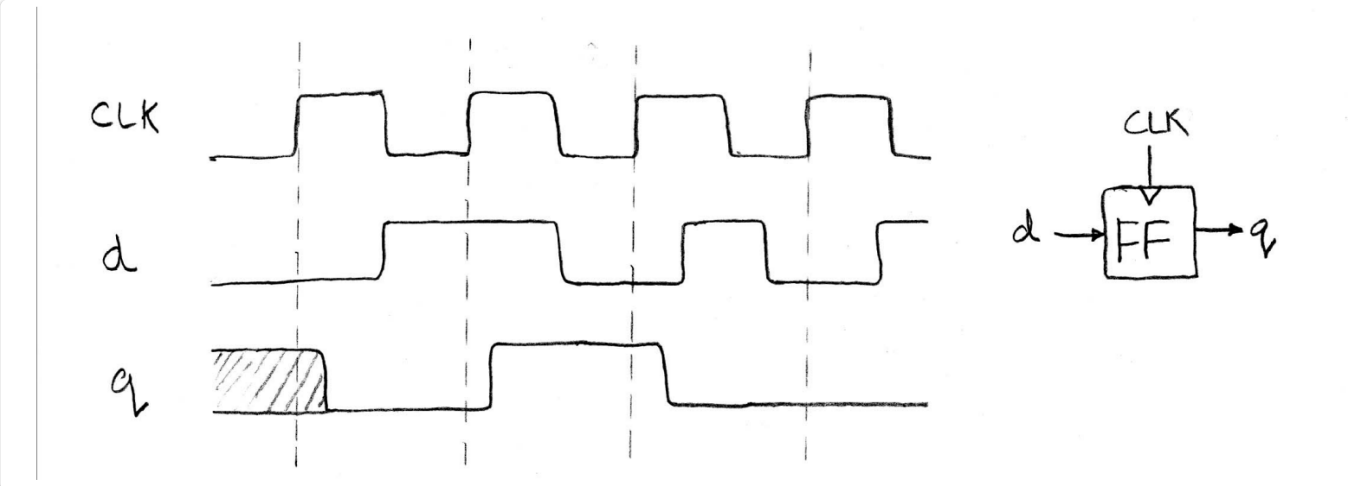
n-bit wide register is nothing but n instance of **flip-flop**.



Input bits is d_i and output is q_i , d stands for data, q stands for quiet a.k.a. stable

edge-triggered d-type flip-flop

Only consider positive edge-triggered.

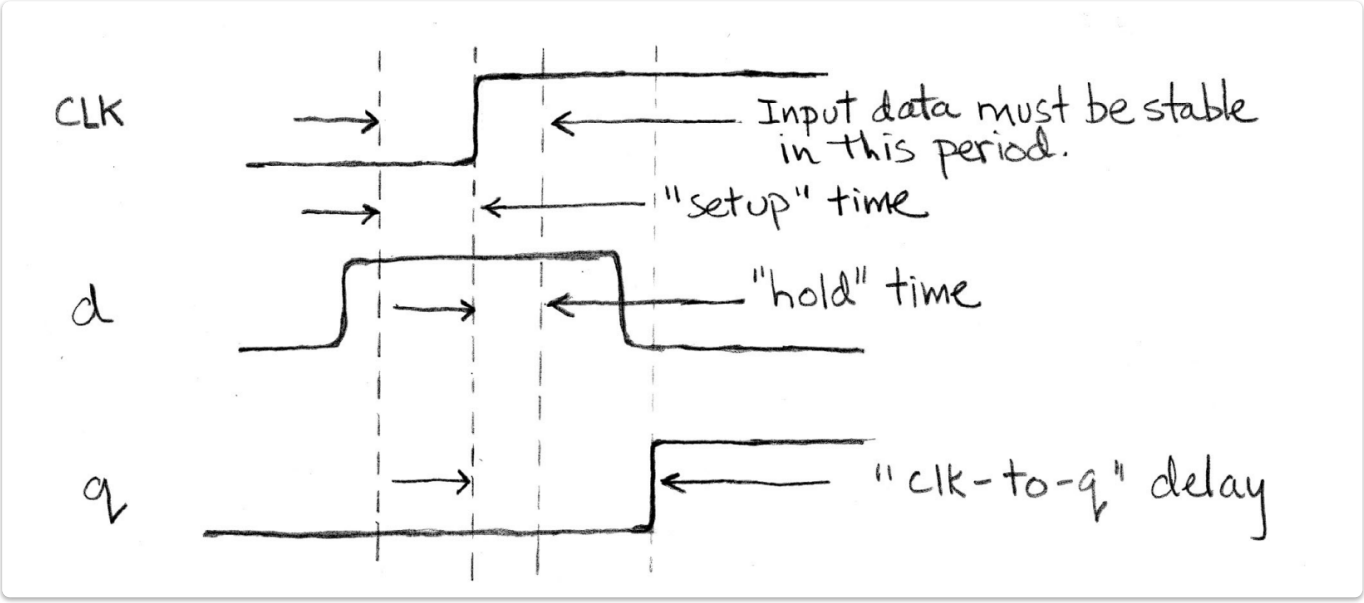


On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored.

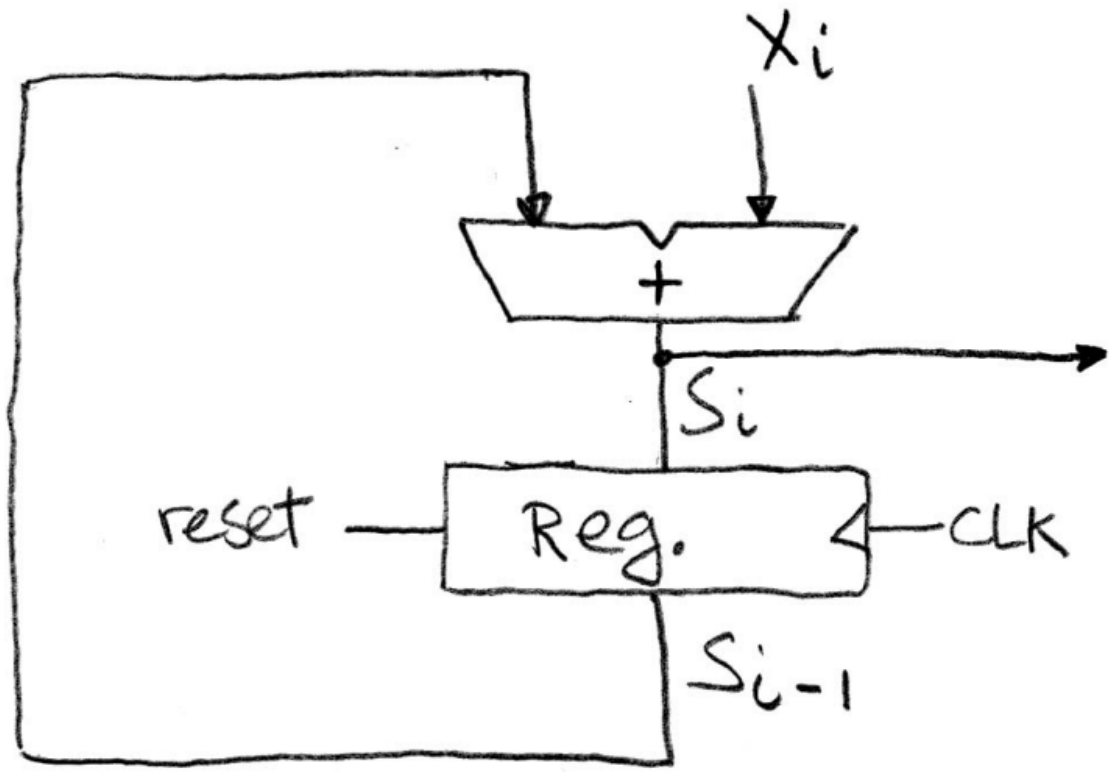
> *limitations*
> *ff cannot change their outputs instantaneously.*
> *Time is need to transfer inputs internally*

Therefore, the input d should be stable before the rising edge and remain stable for a short amount of time after the edge.
These two called: **setup time** and **hold time** .
Setup time mainly prevent sampling during the input is at rising edge.
During this time window, the input shall **not change** .
Once the flip-flop captures the new input, it also takes a small amount time to transfer the new value to output.

This delay is called *clk-to-q* delay.

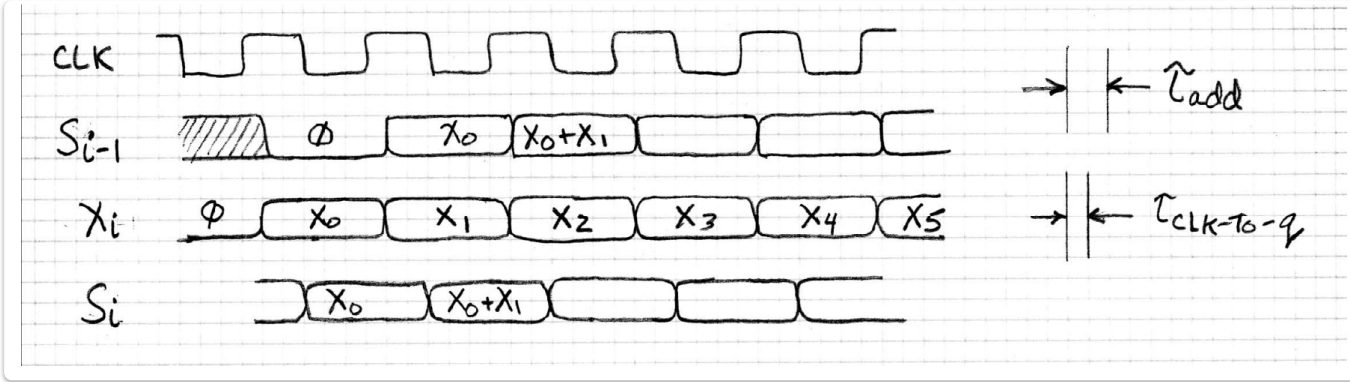


Accumulator



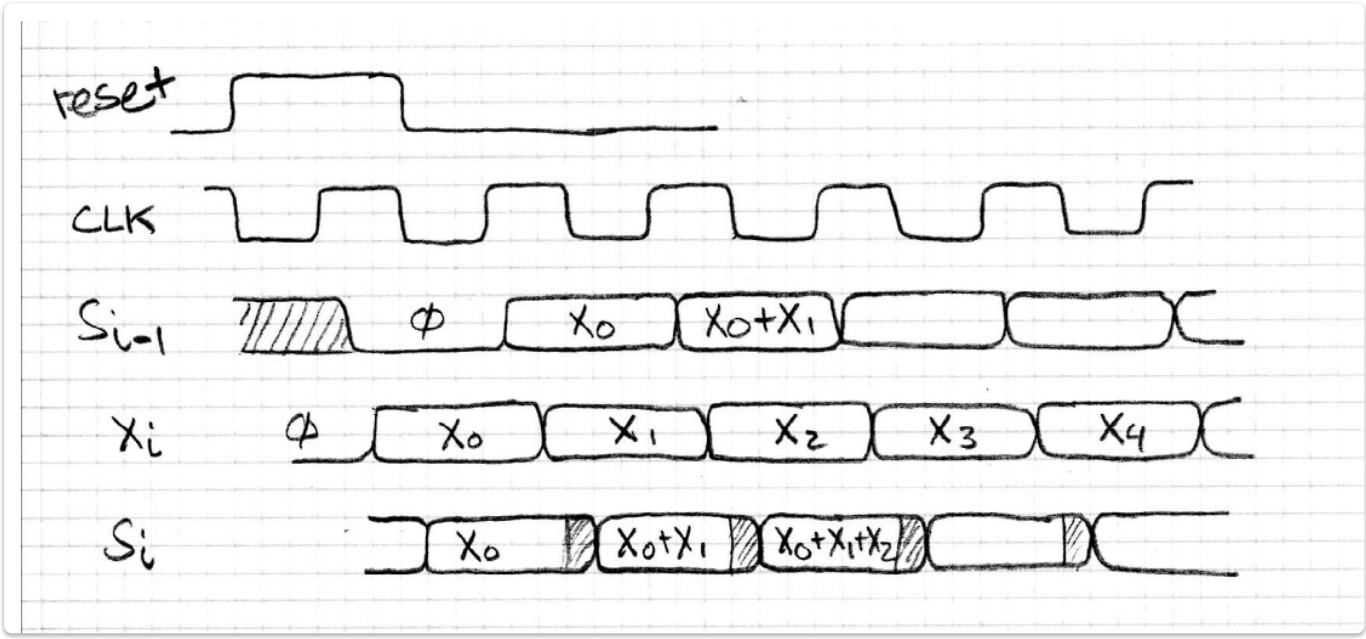
The output of the circuit is labeled S_i, and the output of the register is labeled S_{i-1} to remind us that the register delays the signal for 1 cycle. So if the output of the circuit is holding the result of the ith iteration, then the register holds the result of the $i^{th} - 1$ iteration.

Below is the detailed waveforms for a few interations.



Start by looking at the timing of the change on the output of the register S_{i-1}. This follows the positive-edge of the clock after a small delay (the clk-to-q time of the flip-flops used to implement the register). We assume that the input X is applied at precisely the same time. The two values move through the adder together and after a small delay (the adder propagation delay τ_{add}) a new result appears at the output of the adder, S_i. Then all is quiet until the rising edge of the clock. At that time the output value is transferred to the register and the whole process repeats.

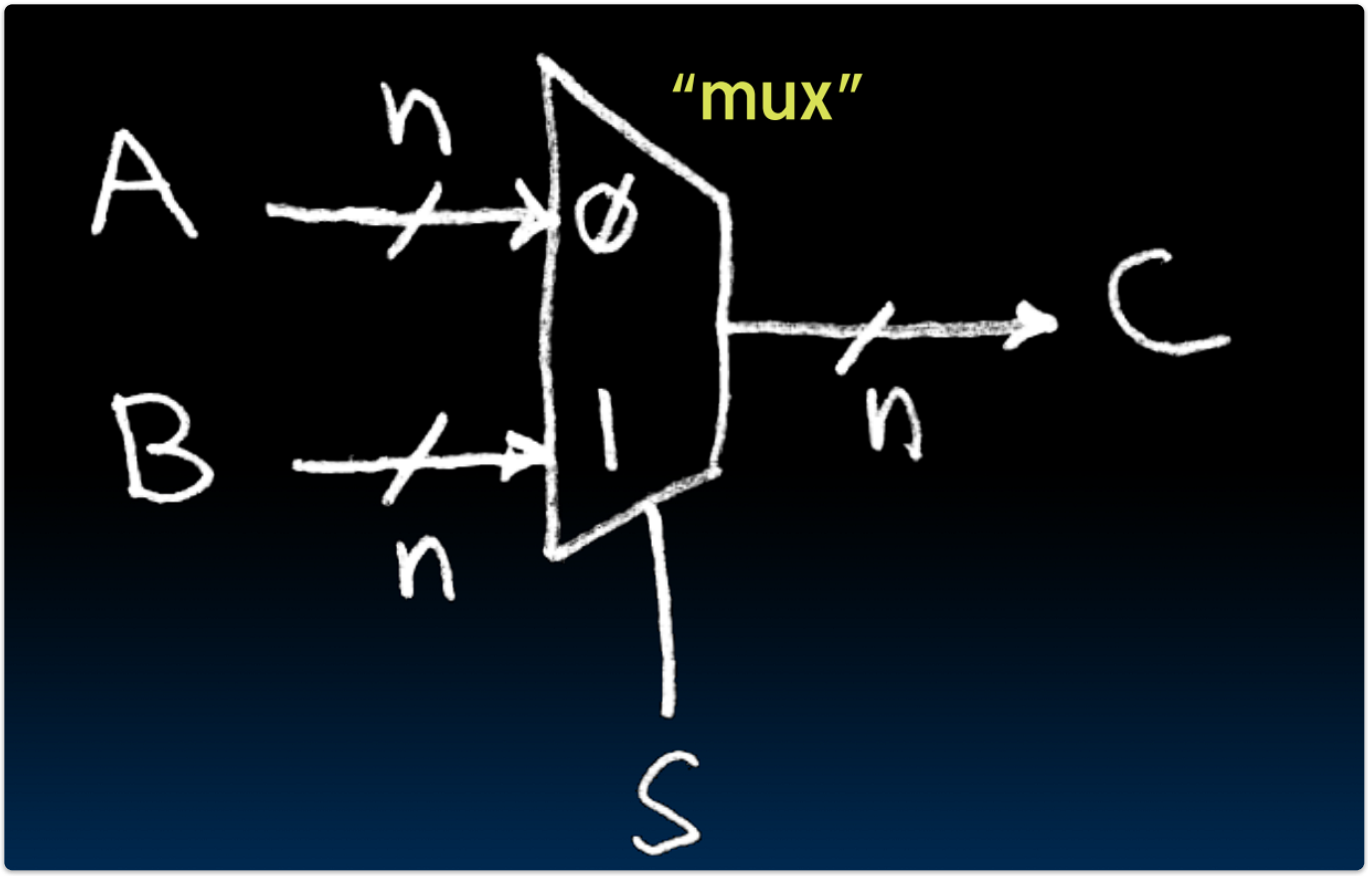
In practice X may not necessarily arrive at the same time as the feedback value, S_{i-1}. The waveforms below show X arriving a little bit later than S_{i-1}.



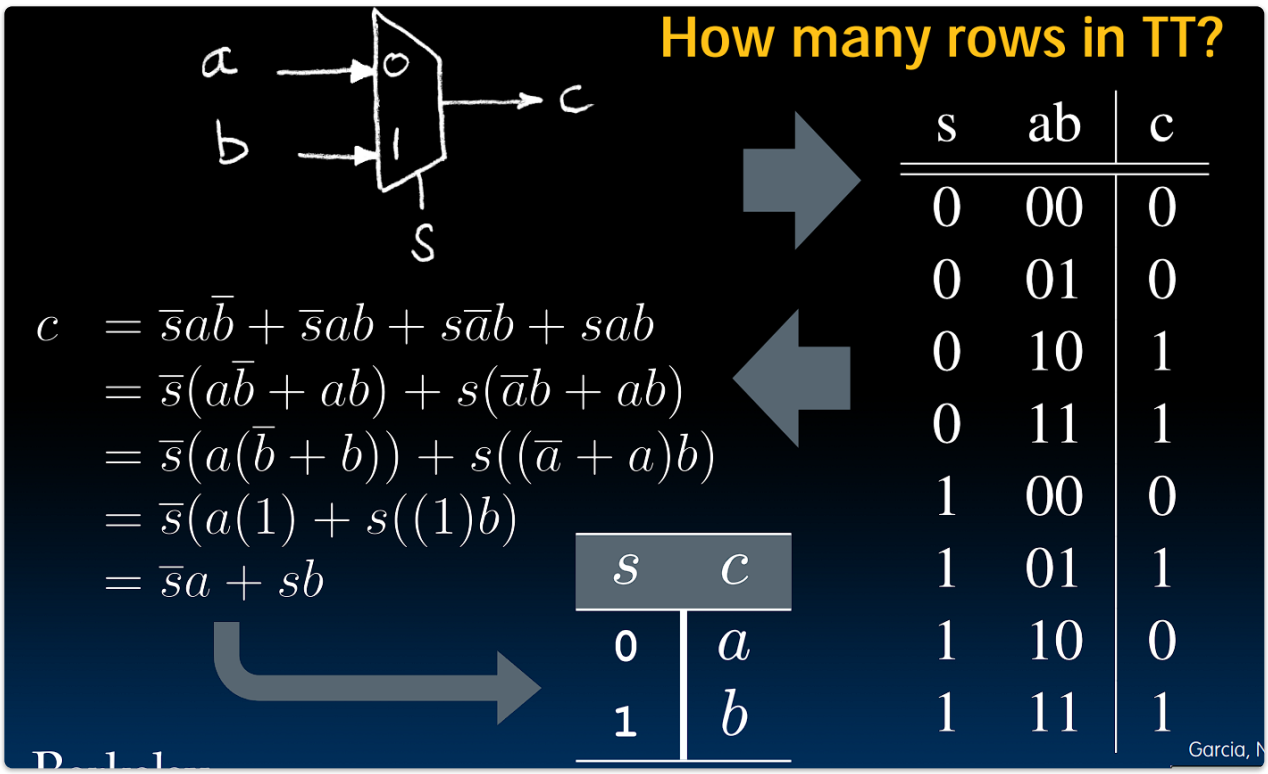
Therefore on each cycle there is a small time period where the adder has inconsistent inputs. For instance, when the register first captures X_0, for a small time period the X input still has X_0, therefore the adder begins to compute $X_0 + X_0$! However, this erroneous calculation is quickly aborted when the X input changes to X_1. Unfortunately, the aborted computation will probably make it through the adder, creating a sort of instability at the output. However, the instability in S_i has no effect on S_{i-1}, as it captures its value from S_i before it goes bad. This sort of arrival mismatch and subsequent output instability is common in many circuits. In properly designed circuits, the instability never happens around the rising-edge of the clock and therefore gets ignored by the registers and down-stream circuitry.

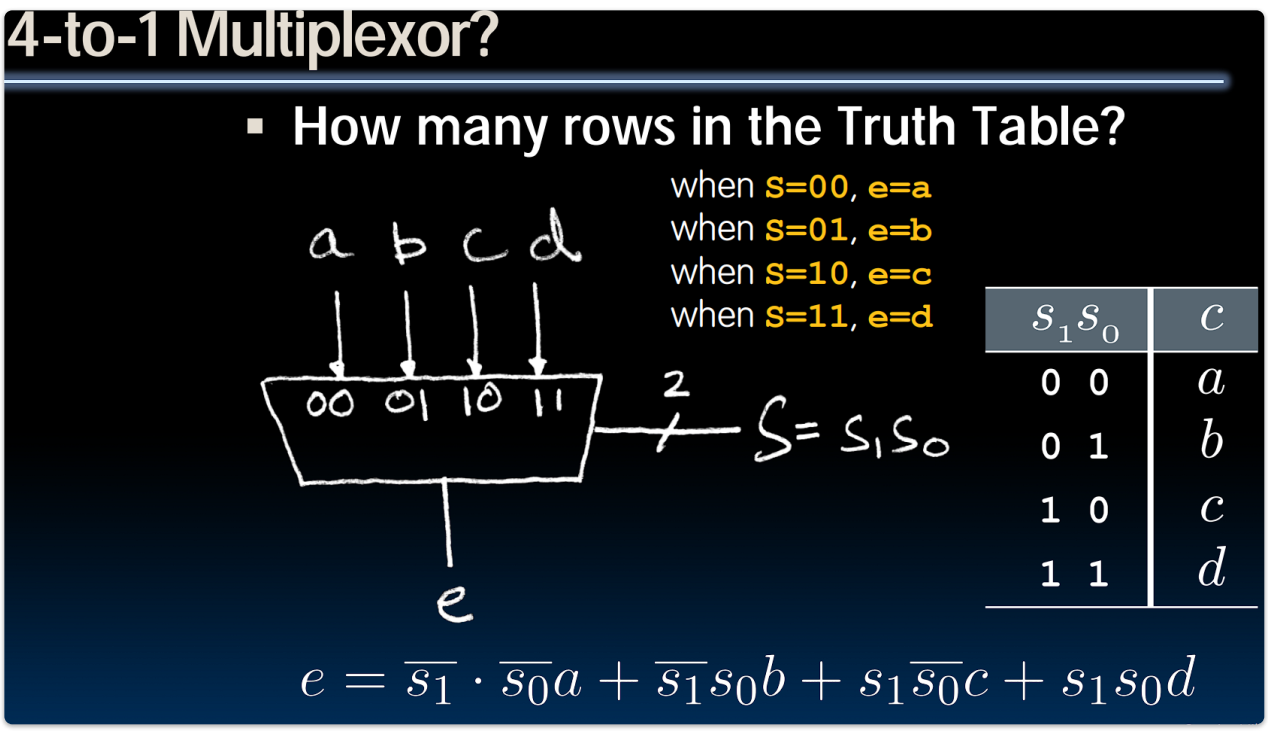
Pipelining -- Adding registers to improve Performance

Data Multiplexors

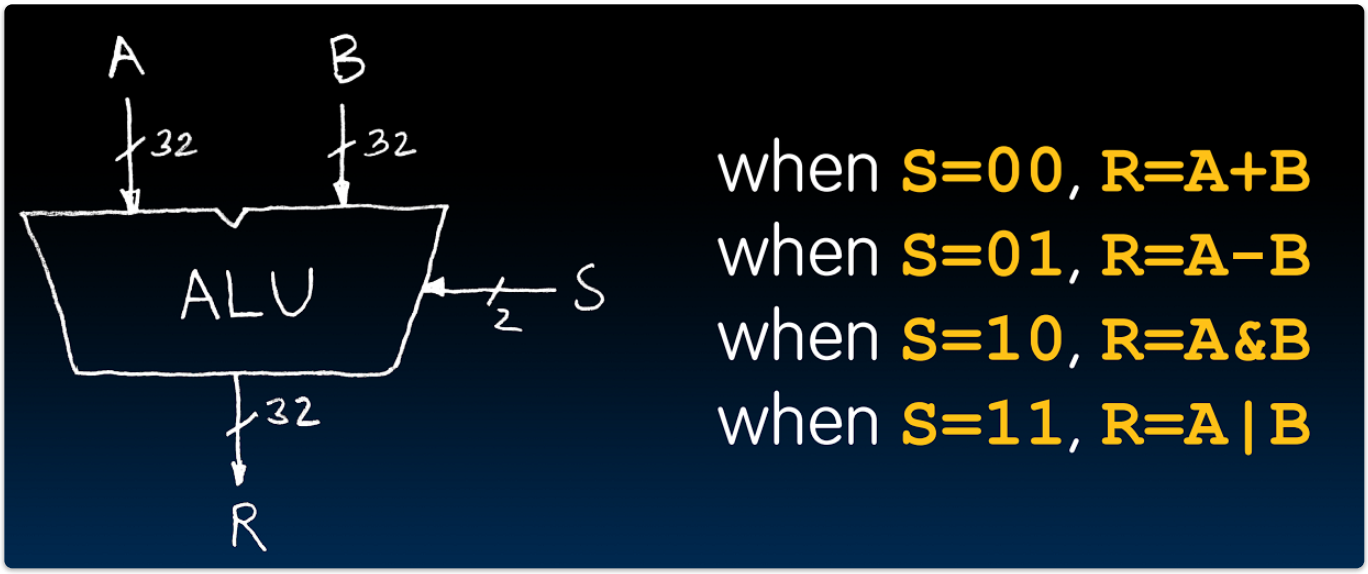


- 2-to-one , n bit wide.

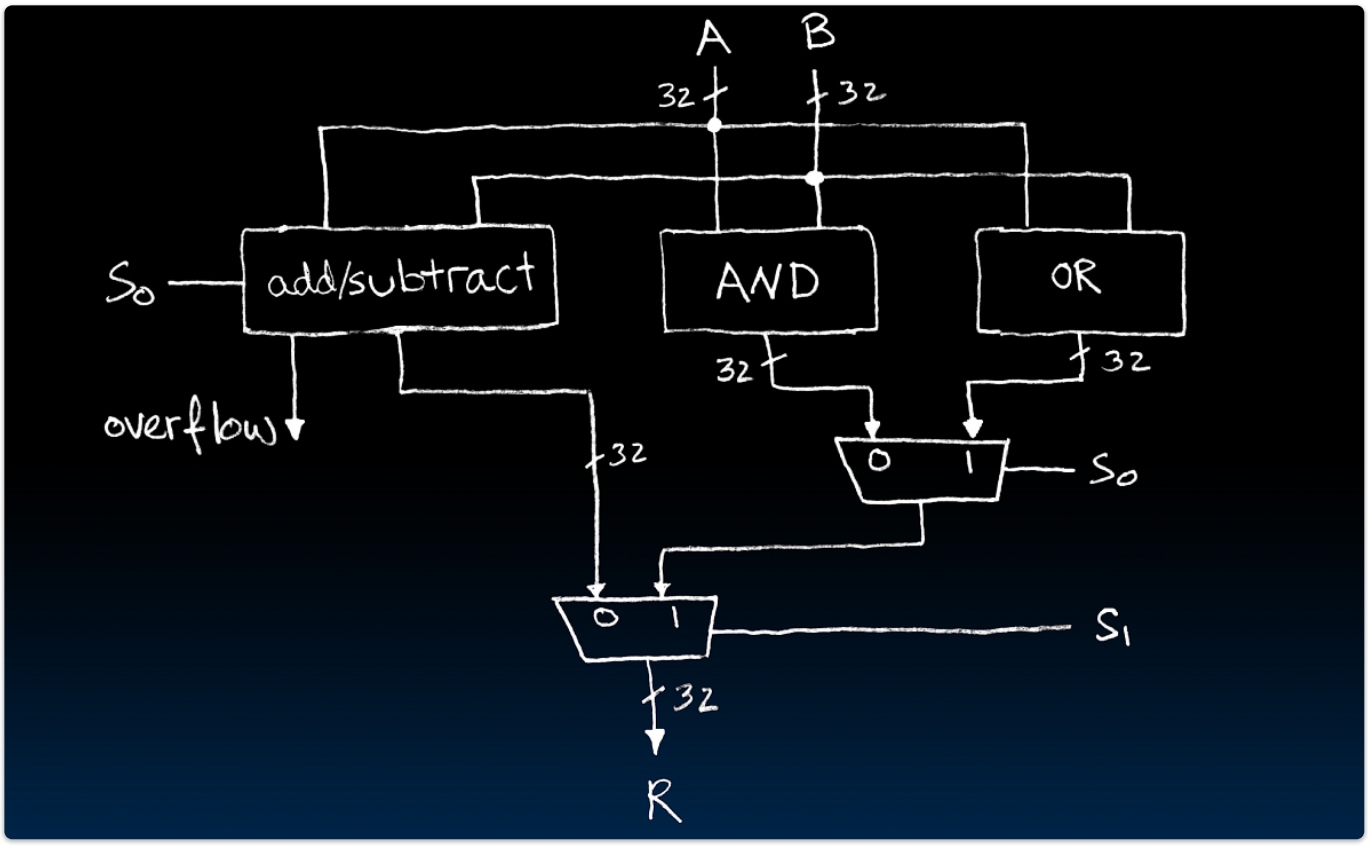




ALU



We can implement muxes hierarchically.



Adder/Subtracter

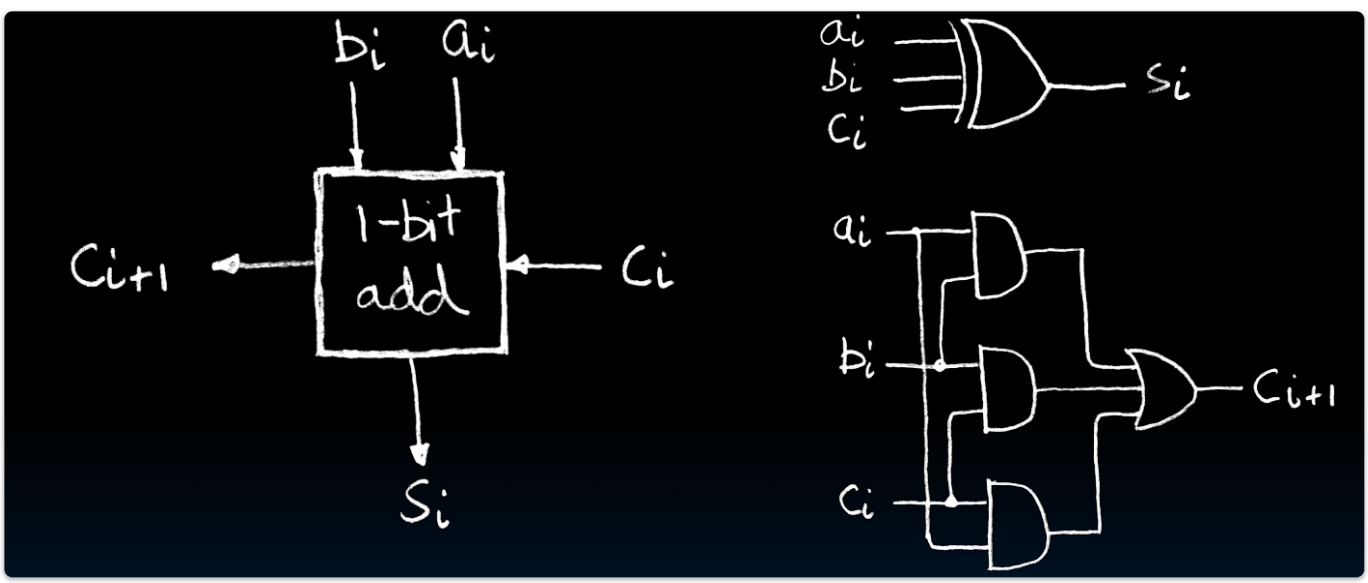
$$
\begin{array}{rcccc}
 & a_3 & a_2 & \boxed{a_1} & a_0 \\
+ & b_3 & b_2 & \boxed{b_1} & b_0 \\
\hline
 & s_3 & s_2 & \boxed{s_1} & s_0
\end{array}
$$

a_i	b_i	c_i	s_i	c_{i+1}
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

$$
\begin{aligned}
s_i &= \mathrm{XOR}(a_i, b_i, c_i) \\
c_{i+1} &= \mathrm{MAJ}(a_i, b_i, c_i) = a_i b_i + a_i c_i + b_i c_i
\end{aligned}
$$

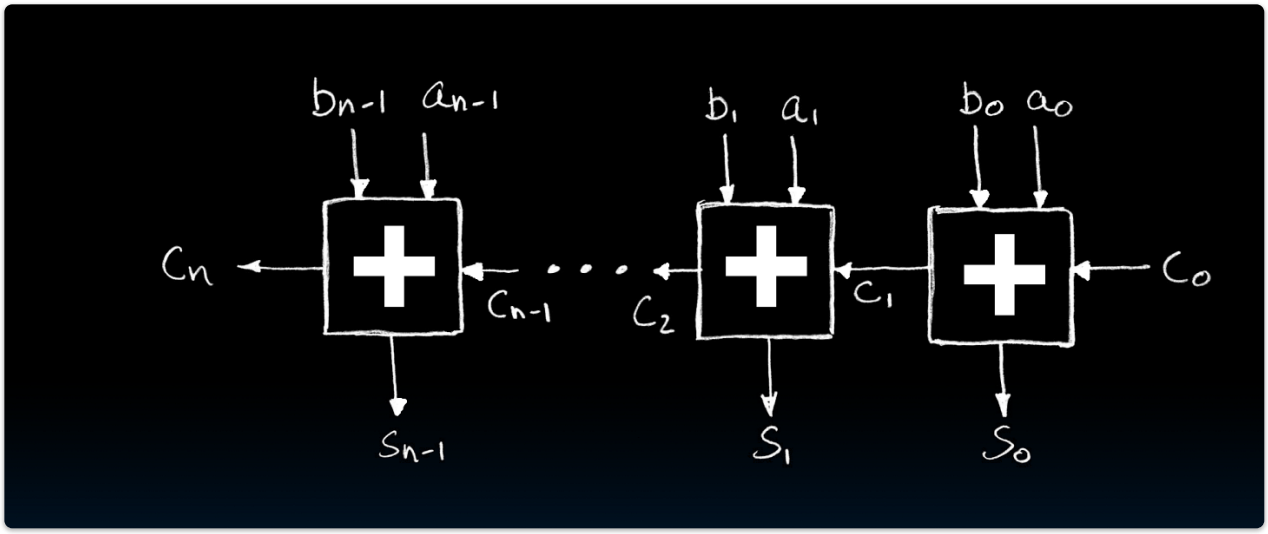
- MAJ :函数结果为a，b，c 中占多数的0or1。

1-bit adder



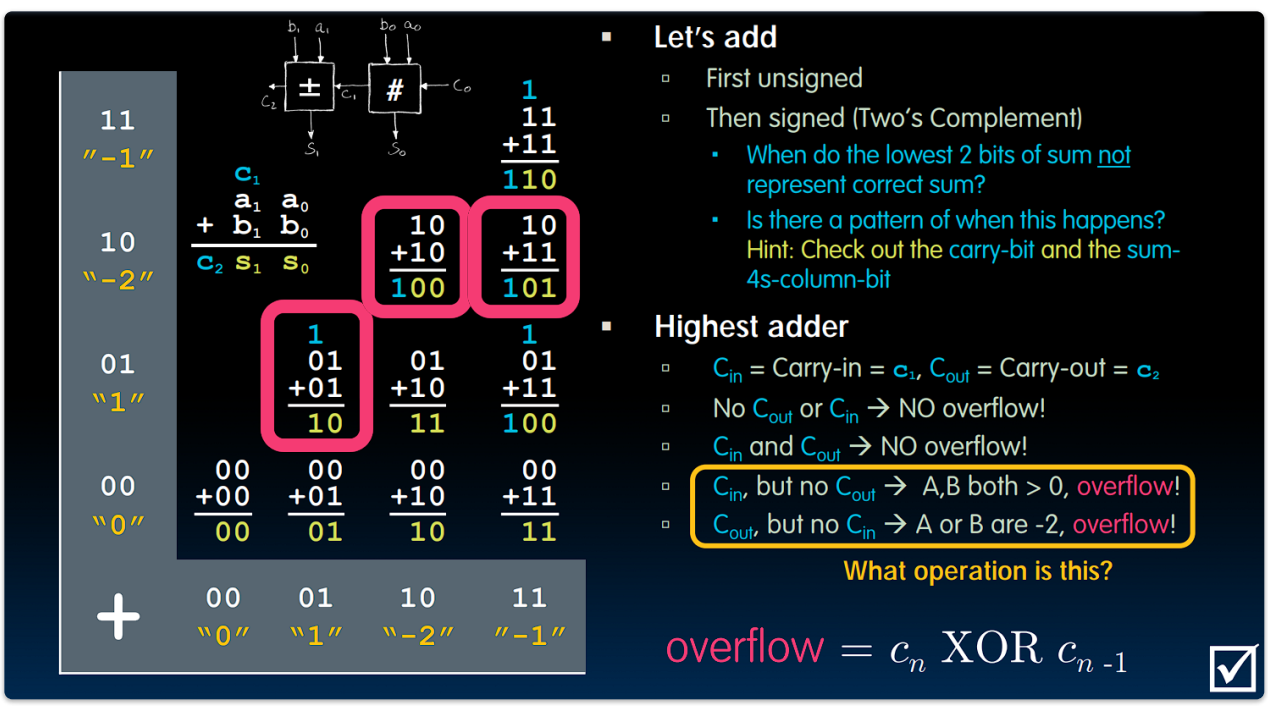
N-bit adder

- Just cascade N 1-bit adder.



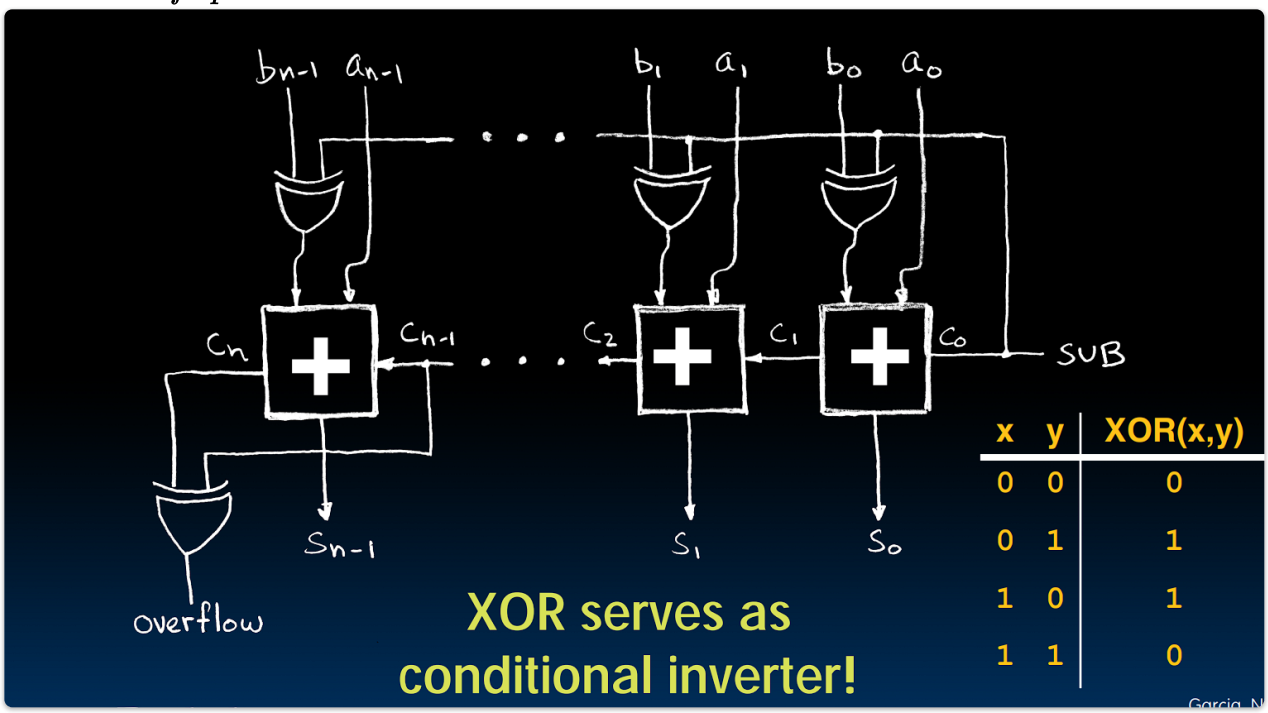
Overflow

- When doing unsigned operations, carry bit from MSB means **overflow** .
- When doing **signed** operations, overflow depends on $c_n \; XOR \; c_{n-1}$.



Subtractor

- $A - B = A + (-B)$
- $-B$ = B_{filp}+1



When **Sub** line is 1 , then b_i will flip over, and **Sub** will work as $LSB's$ carry bit.