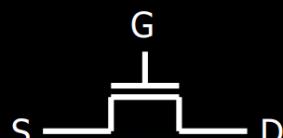


Transistors

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

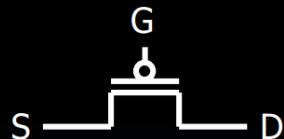
To remember:
n ("normal")
p (has a circle,
like the top
part of P itself)



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



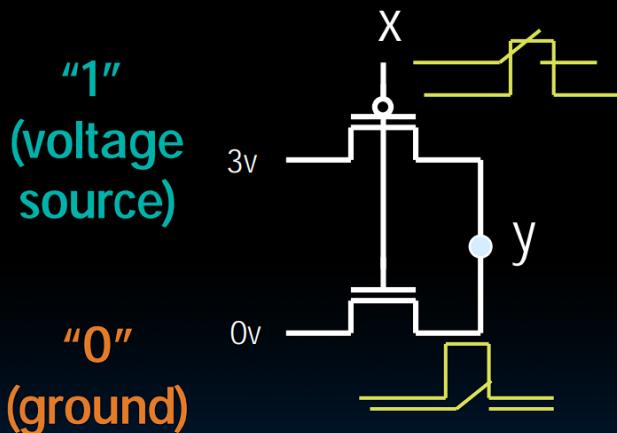
G LOW
G HIGH



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

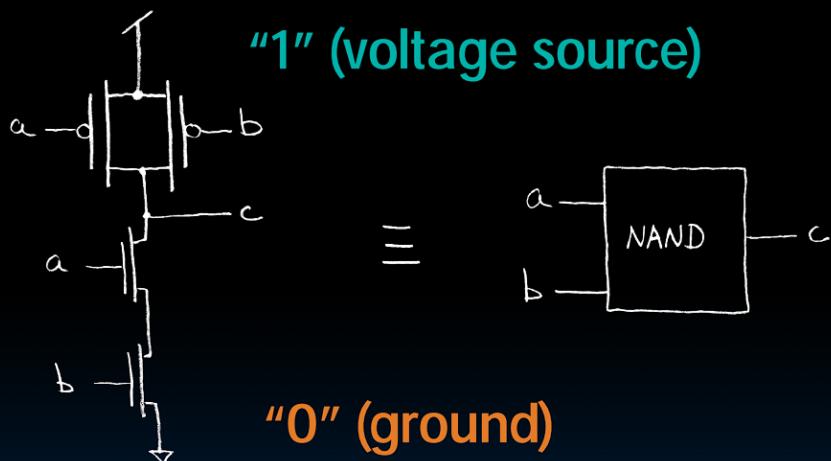


what is the relationship between x and y ?



| X | y |
|---------|---------|
| 0 volts | 3 volts |
| 3 volts | 0 volts |

- Chips are composed of nothing but transistors and wires.
- Small groups of transistors form useful building blocks.



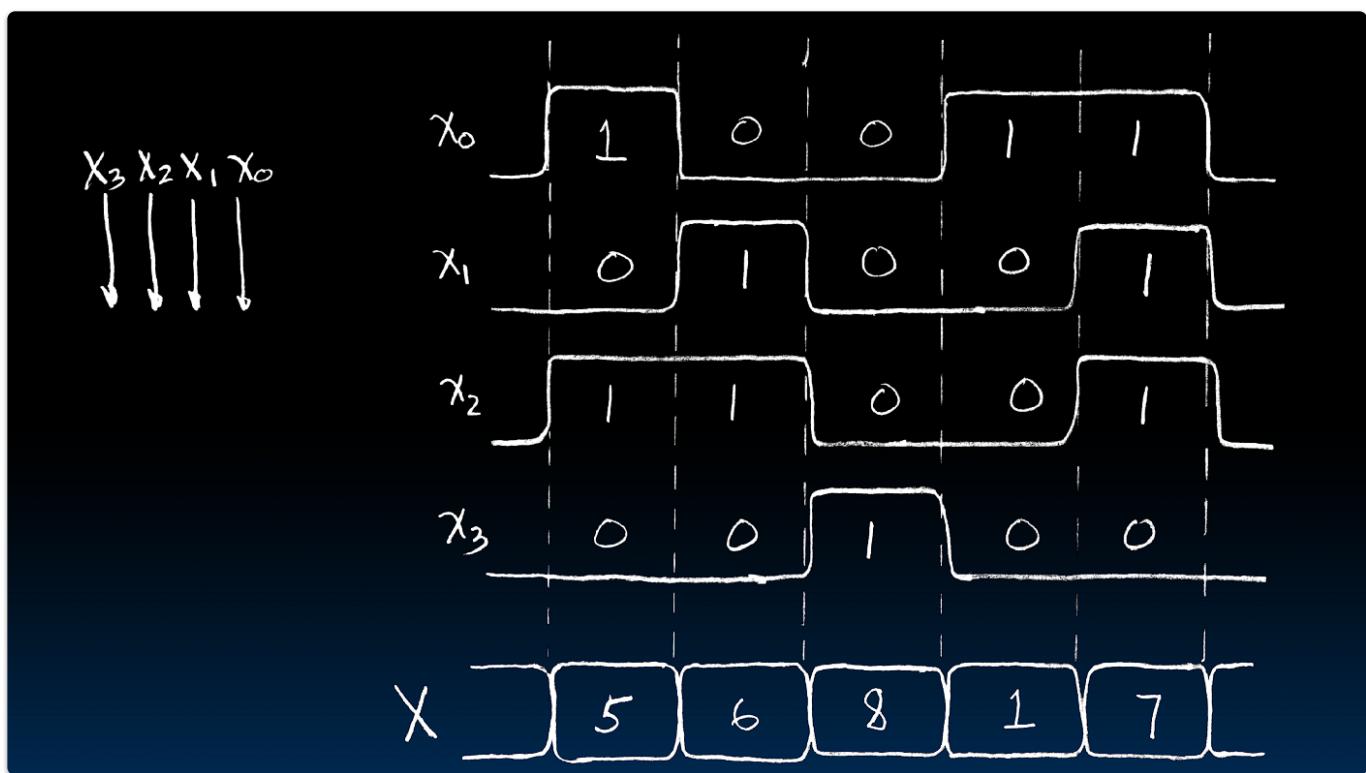
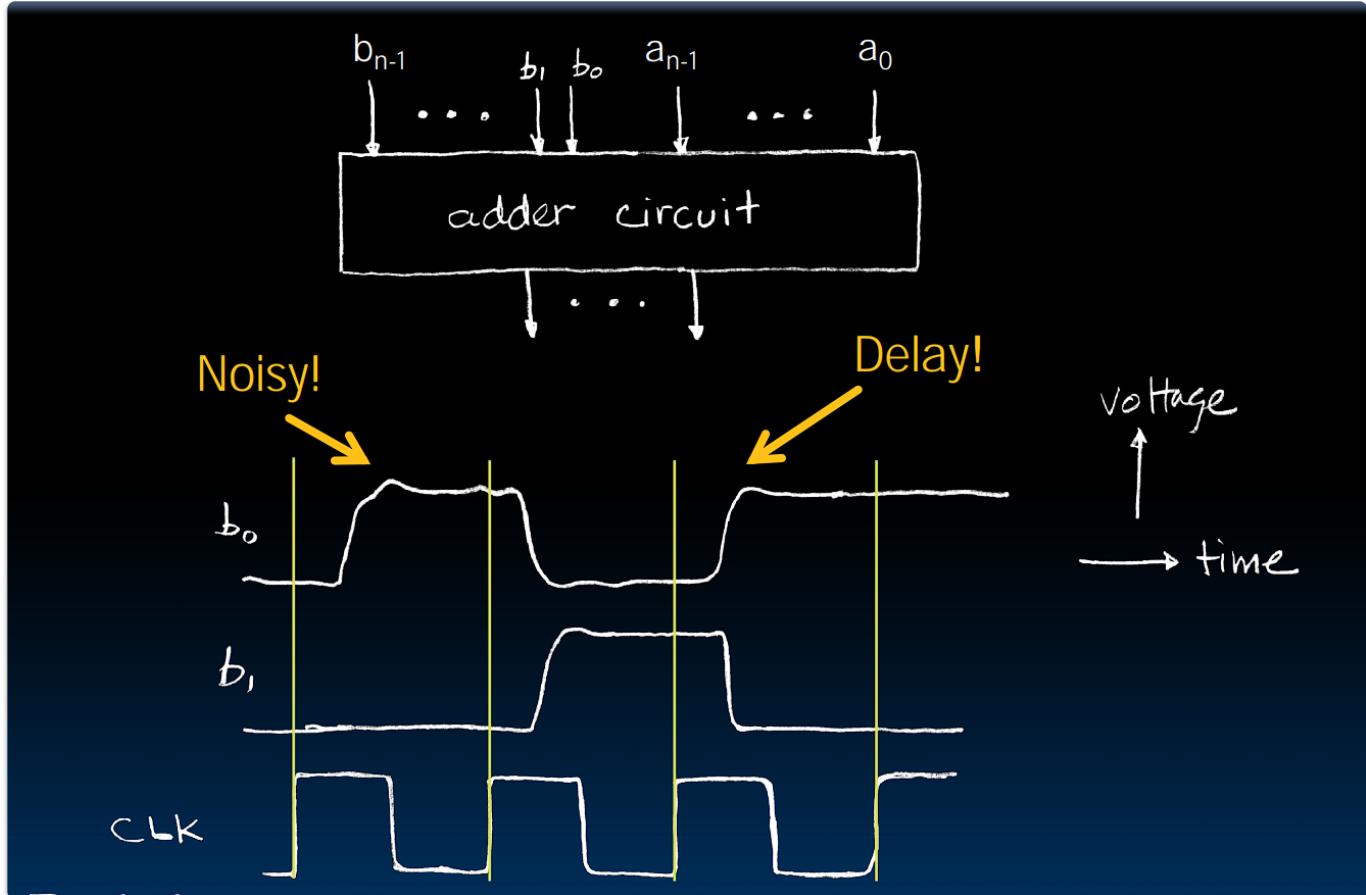
| a | b | c |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- Blocks are organized in a hierarchy to build higher-level blocks: ex: adders.
- You can build AND, OR, NOT out of NAND!



Garcia-Nil

Signals and Waveform

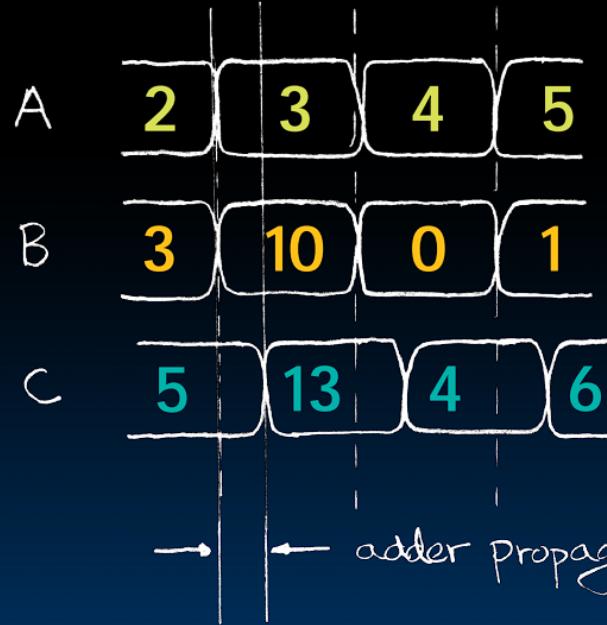
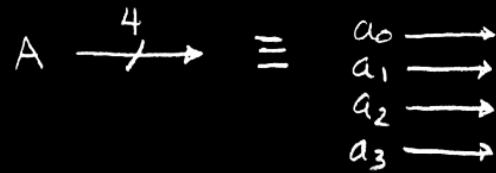


- A nibble adder which treat X_0 as the LSB and X_3 as MSB, thus gives the X below.

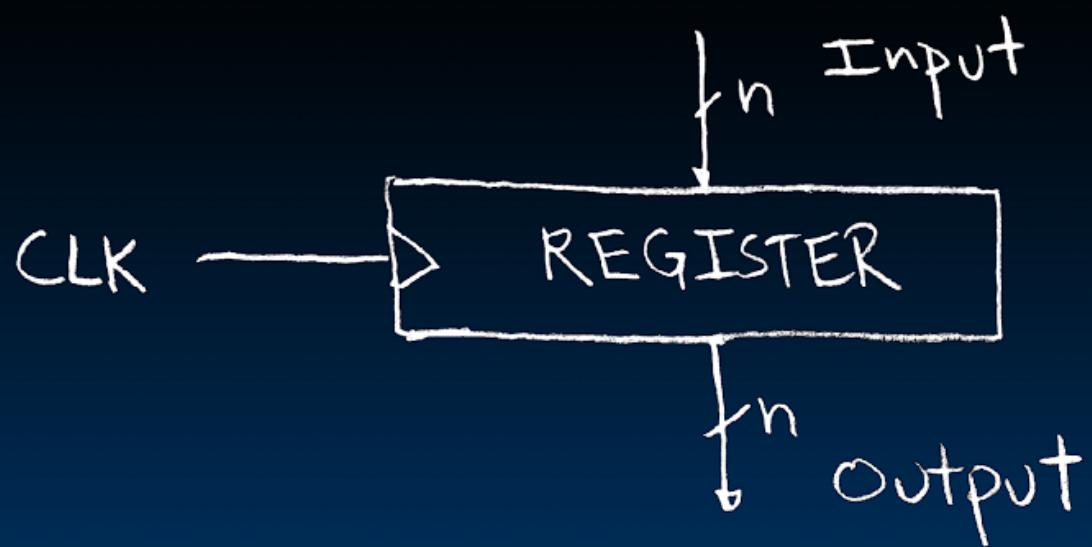
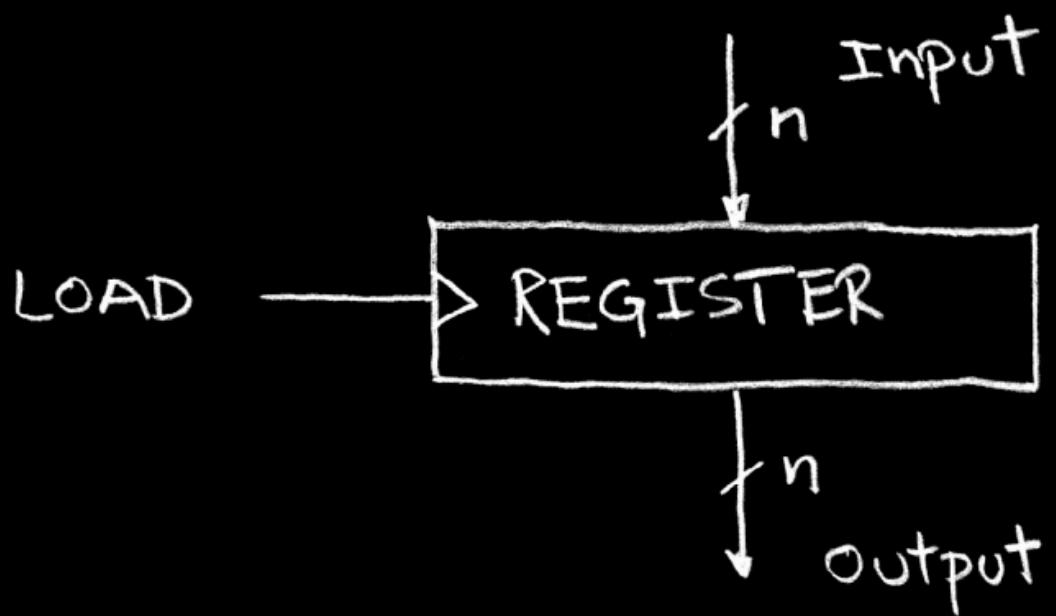


$$A = [a_3, a_2, a_1, a_0]$$

$$B = [b_3, b_2, b_1, b_0]$$



Register



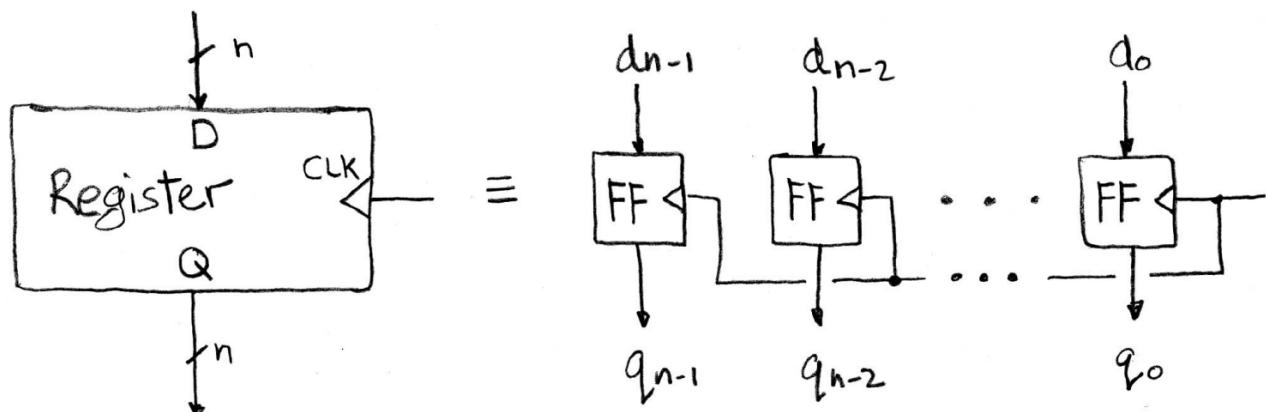
clk enables the register to store the input.

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 ($\&$, \mid , \sim)
 - State circuits (e.g., registers)

Register Details: Flip-Flops

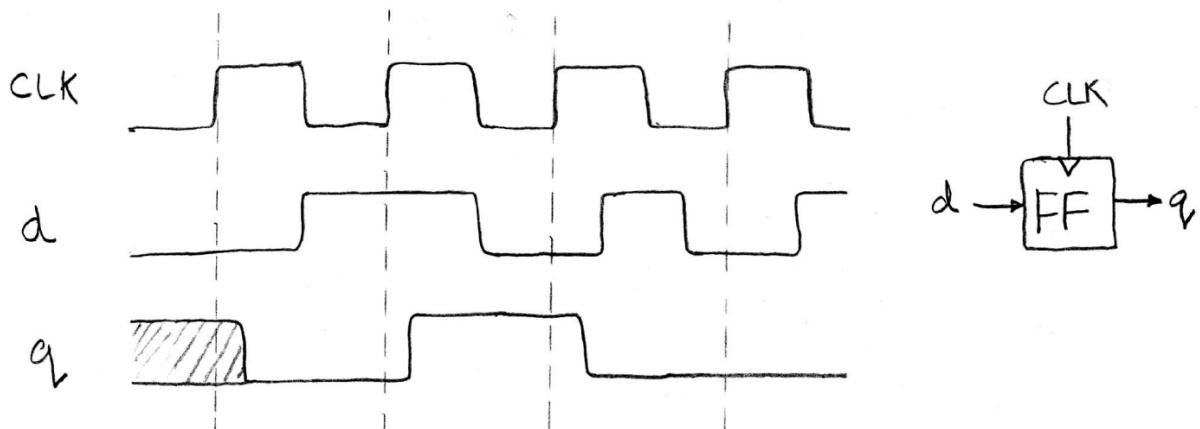
n-bit wide register is nothing but n instance of flip-flop.



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.

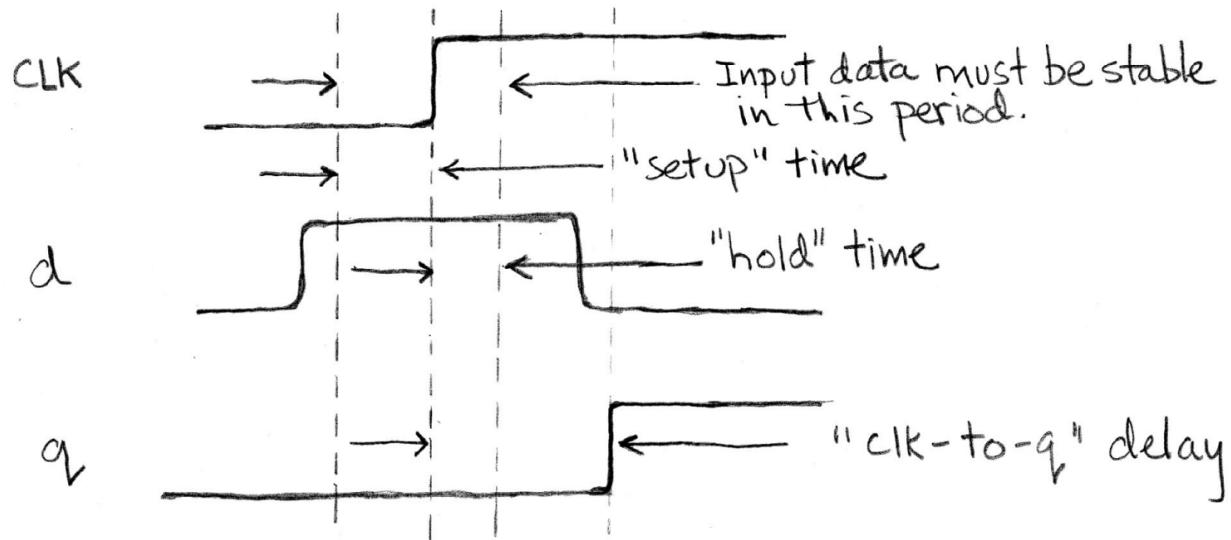
These two called: **setup time** and **hold time**.

Setup time mainly prevent sampling during the input is at rising edge.

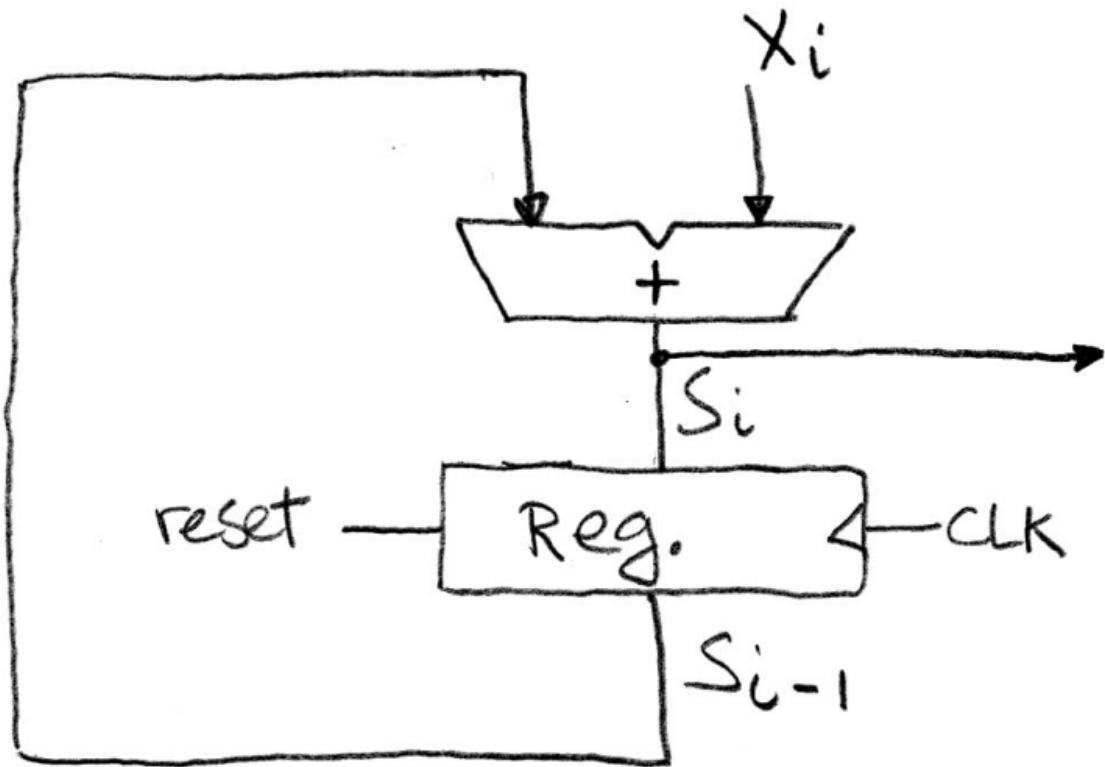
During this time window, the input shall **not change**.

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.

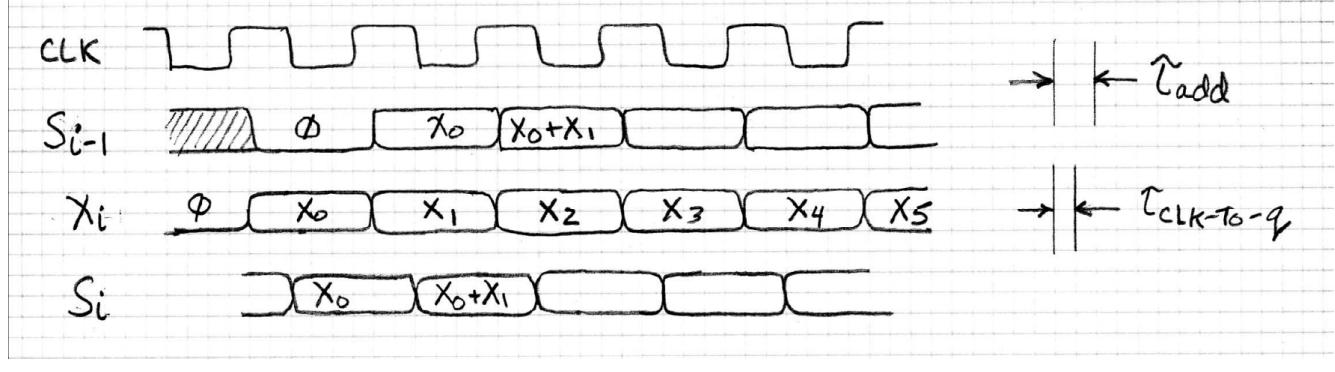


Accumulator



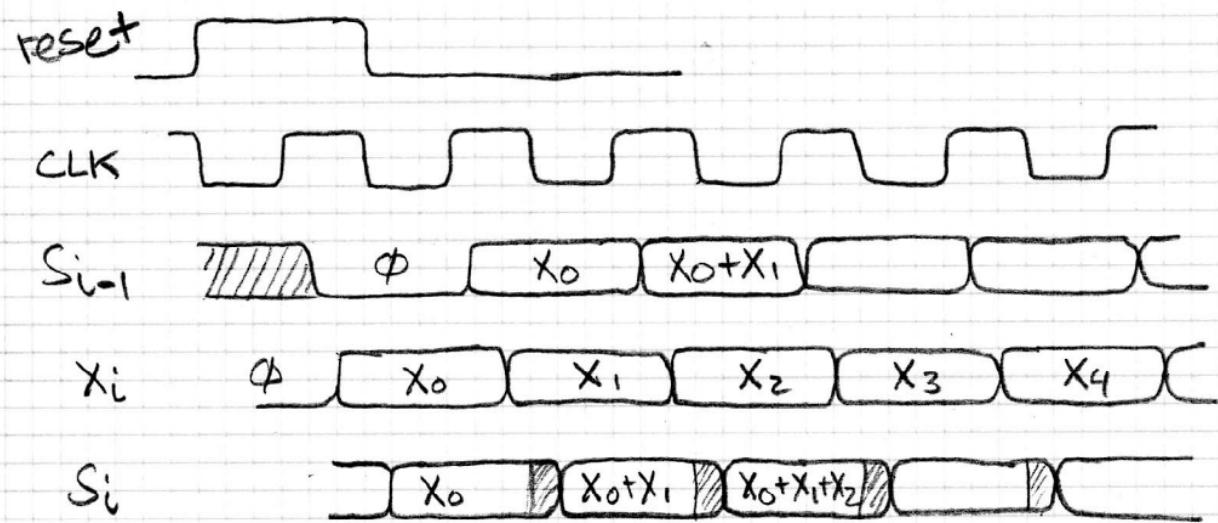
The output of the circuit is labeled S_i , and the output of the register is labeled S_{i-1} to remind us that the register delays the signal for 1 cycle. So if the output of the circuit is holding the result of the i^{th} iteration, then the register holds the result of the $i^{th} - 1$ iteration.

Below is the detailed waveforms for a few iterations.



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.

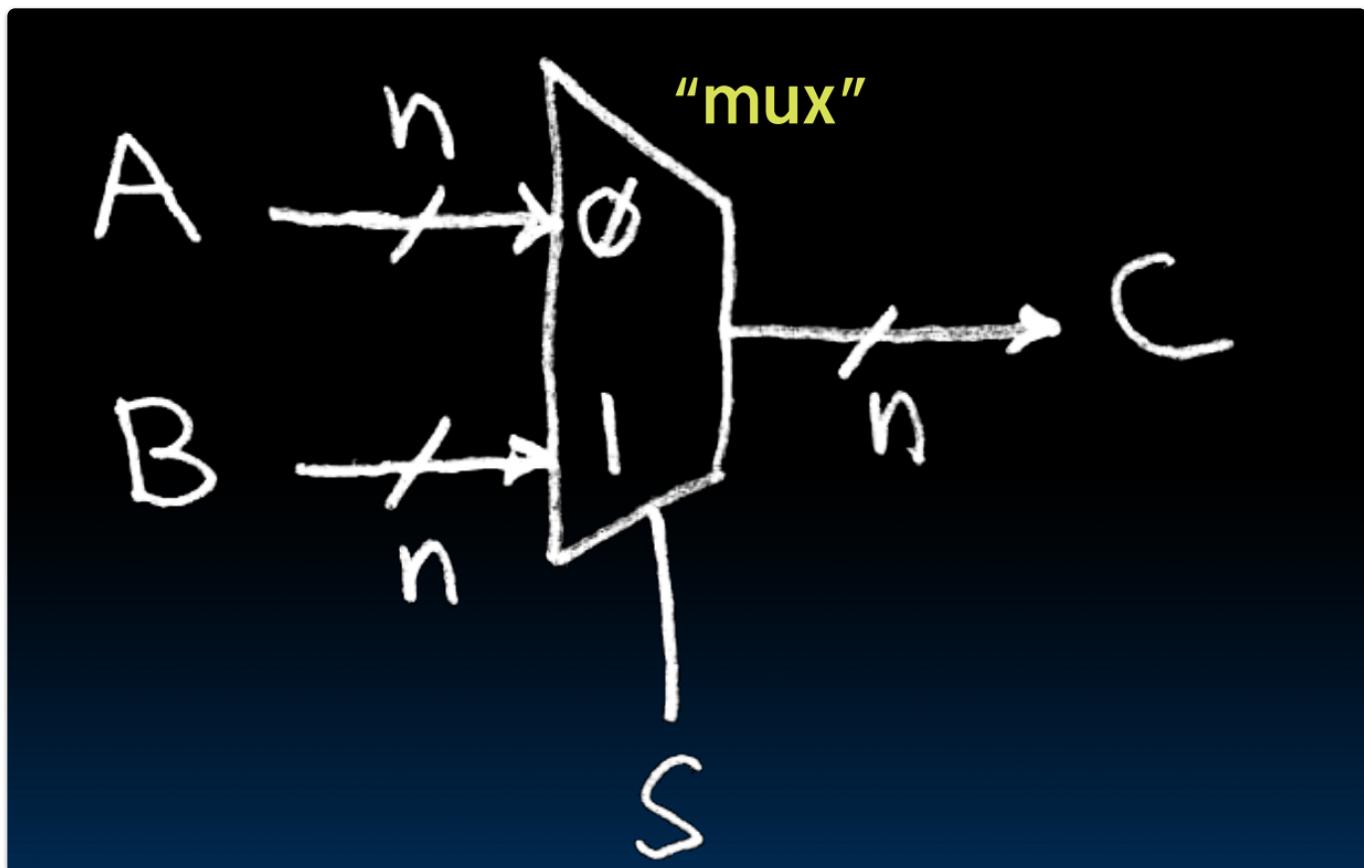
In practice X may not necessarily arrive at the same time as the feedback value, S_{i-1} . The waveforms below show X arriving a little bit later than S_{i-1} .



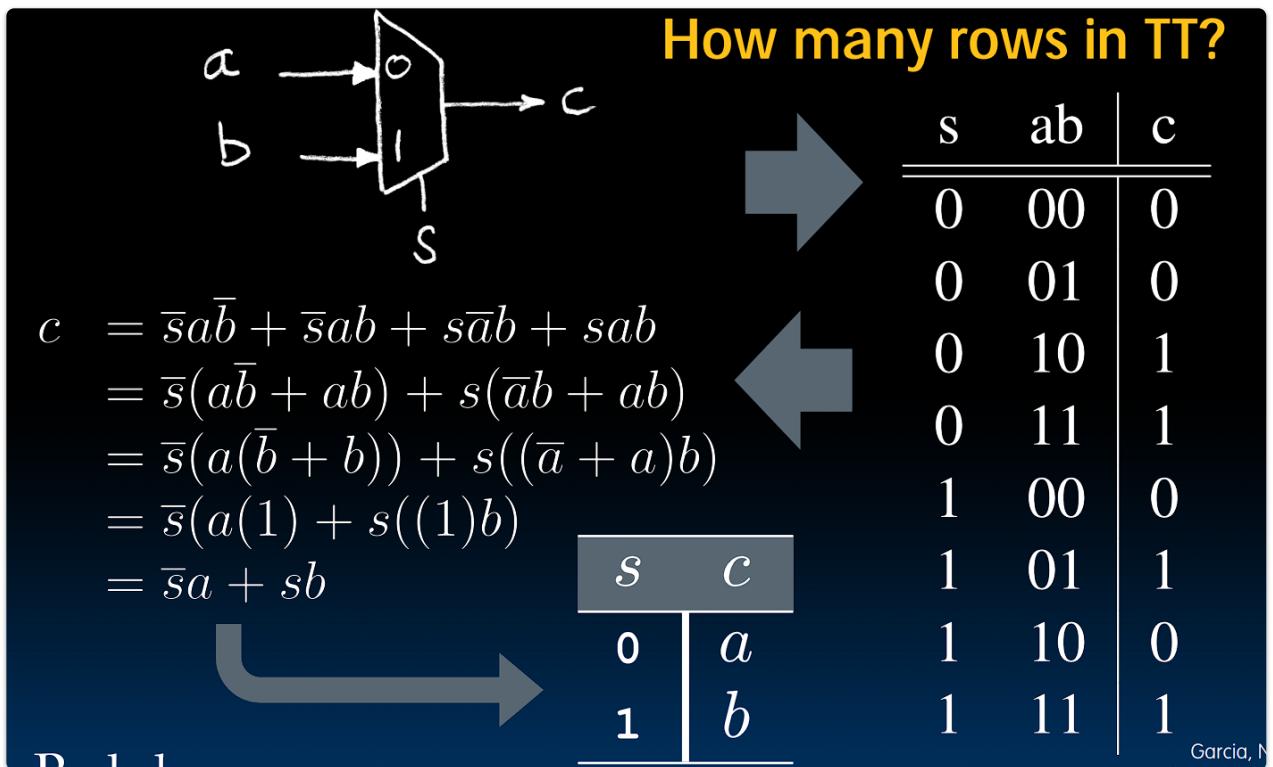
Therefore on each cycle there is a small time period where the adder has inconsistent inputs. For instance, when the register first captures X_0 , for a small time period the X input still has X_0 , therefore the adder begins to compute $X_0 + X_0$! However, this erroneous calculation is quickly aborted 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

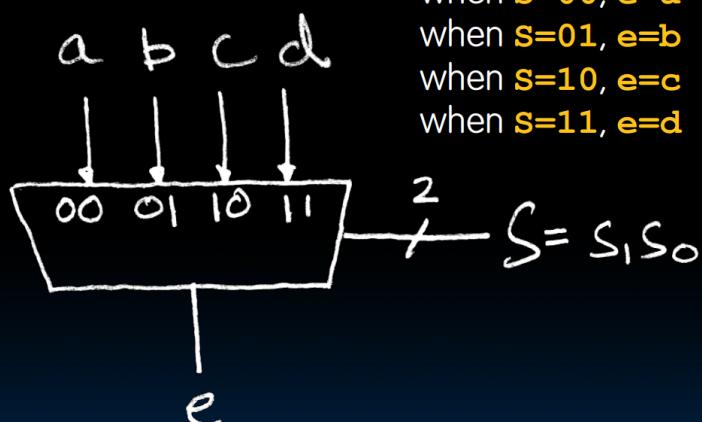


- 2-to-one, n bit wide.



4-to-1 Multiplexor?

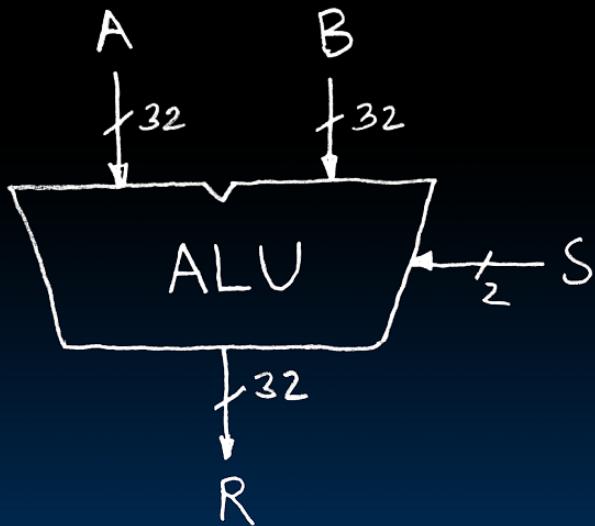
- How many rows in the Truth Table?



| $s_1 s_0$ | c |
|-----------|-----|
| 0 0 | a |
| 0 1 | b |
| 1 0 | c |
| 1 1 | d |

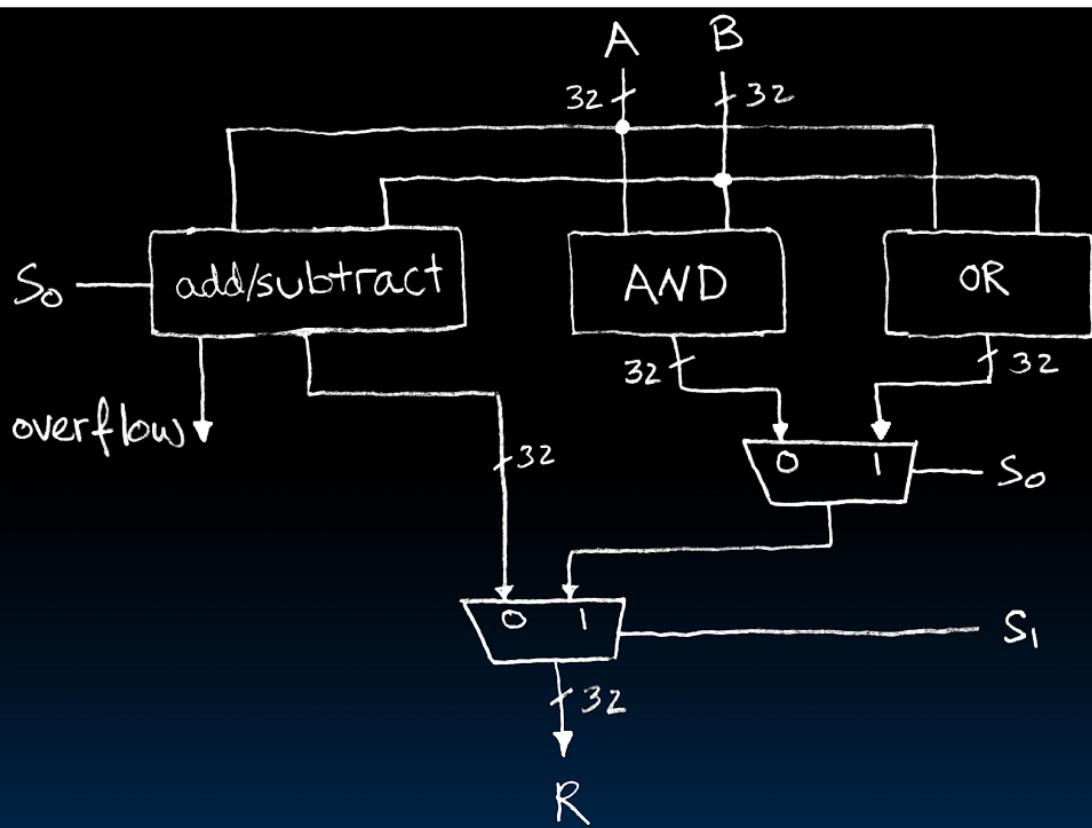
$$e = \bar{s}_1 \cdot \bar{s}_0 a + \bar{s}_1 s_0 b + s_1 \bar{s}_0 c + s_1 s_0 d$$

ALU



when $S=00$, $R=A+B$
 when $S=01$, $R=A-B$
 when $S=10$, $R=A \& B$
 when $S=11$, $R=A | B$

We can implement muxes hierarchically.



Adder/Subtractor

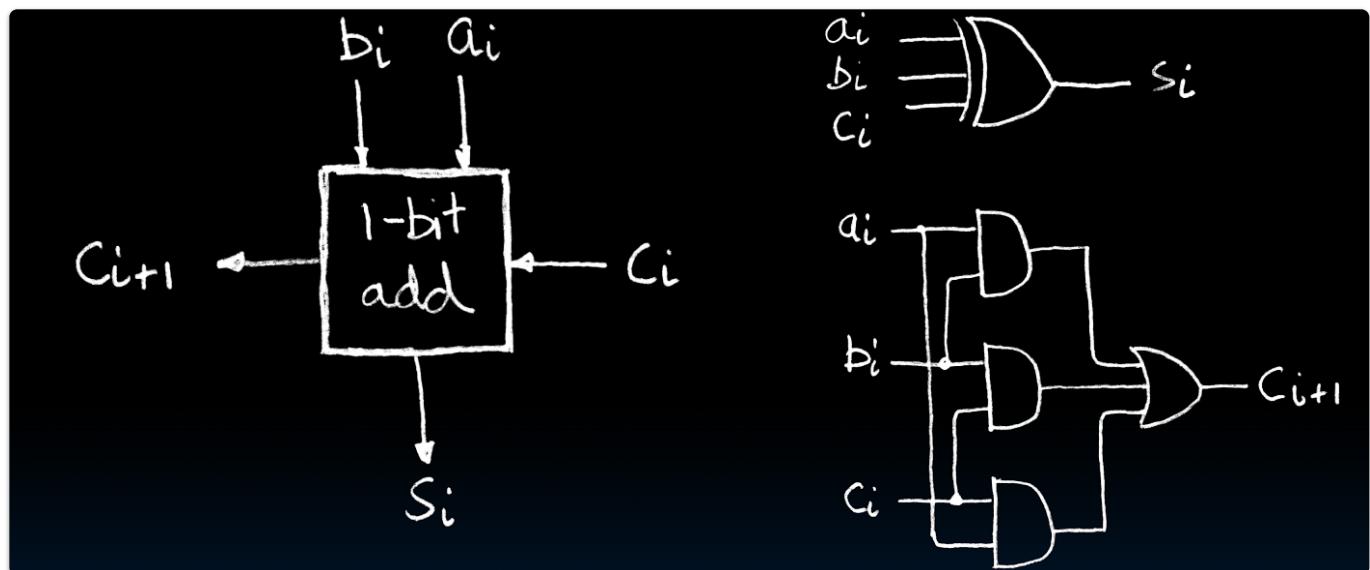
$$+ \begin{array}{cc|c|c} a_3 & a_2 & a_1 & a_0 \\ b_3 & b_2 & b_1 & b_0 \\ \hline s_3 & s_2 & s_1 & s_0 \end{array}$$

| 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 |

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

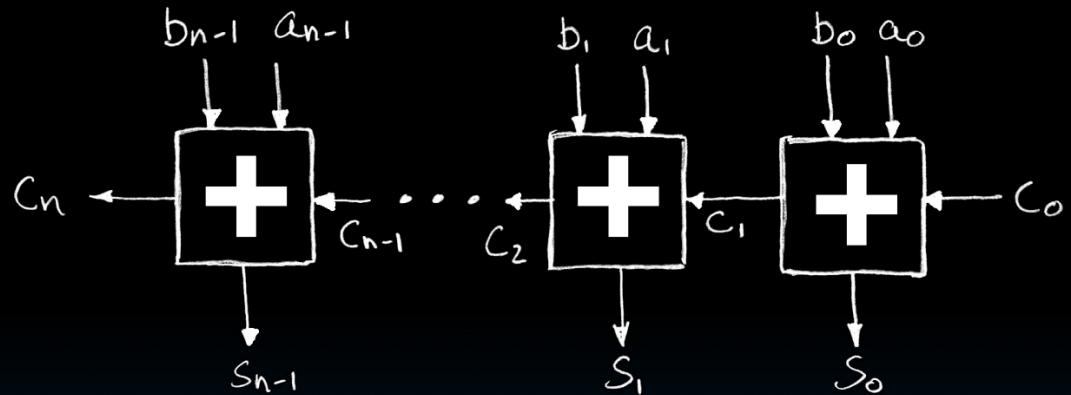
- MAJ : 函数结果为 a, b, c 中占多数的 0 or 1。

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 $c_n \text{ XOR } c_{n-1}$.

| | | | |
|--|---|---|--|
| $\begin{array}{r} 11 \\ "-1" \\ \hline 10 \\ "-2" \\ \hline 01 \\ "1" \\ \hline 00 \\ "0" \end{array}$ | $\begin{array}{c} \pm \\ + \\ \hline \end{array}$ | $\begin{array}{c} \# \\ + \\ \hline \end{array}$ | $\begin{array}{r} 1 \\ 11 \\ +11 \\ \hline 10 \end{array}$ |
| | $\begin{array}{r} c_1 \\ a_1 \\ \hline a_0 \\ b_1 \\ b_0 \\ \hline c_2 \quad s_1 \quad s_0 \end{array}$ | $\begin{array}{r} c_0 \\ \hline s_1 \\ s_0 \end{array}$ | |
| | $\begin{array}{r} 10 \\ +10 \\ \hline 100 \end{array}$ | $\begin{array}{r} 10 \\ +11 \\ \hline 101 \end{array}$ | |
| | $\begin{array}{r} 01 \\ +01 \\ \hline 10 \end{array}$ | $\begin{array}{r} 01 \\ +10 \\ \hline 11 \end{array}$ | $\begin{array}{r} 01 \\ +11 \\ \hline 100 \end{array}$ |
| | $\begin{array}{r} 00 \\ +00 \\ \hline 00 \end{array}$ | $\begin{array}{r} 00 \\ +01 \\ \hline 01 \end{array}$ | $\begin{array}{r} 00 \\ +10 \\ \hline 10 \end{array}$ |
| | $\begin{array}{r} 00 \\ +11 \\ \hline 11 \end{array}$ | | |

+ 00 01 10 11
 "0" "1" "-2" "-1"

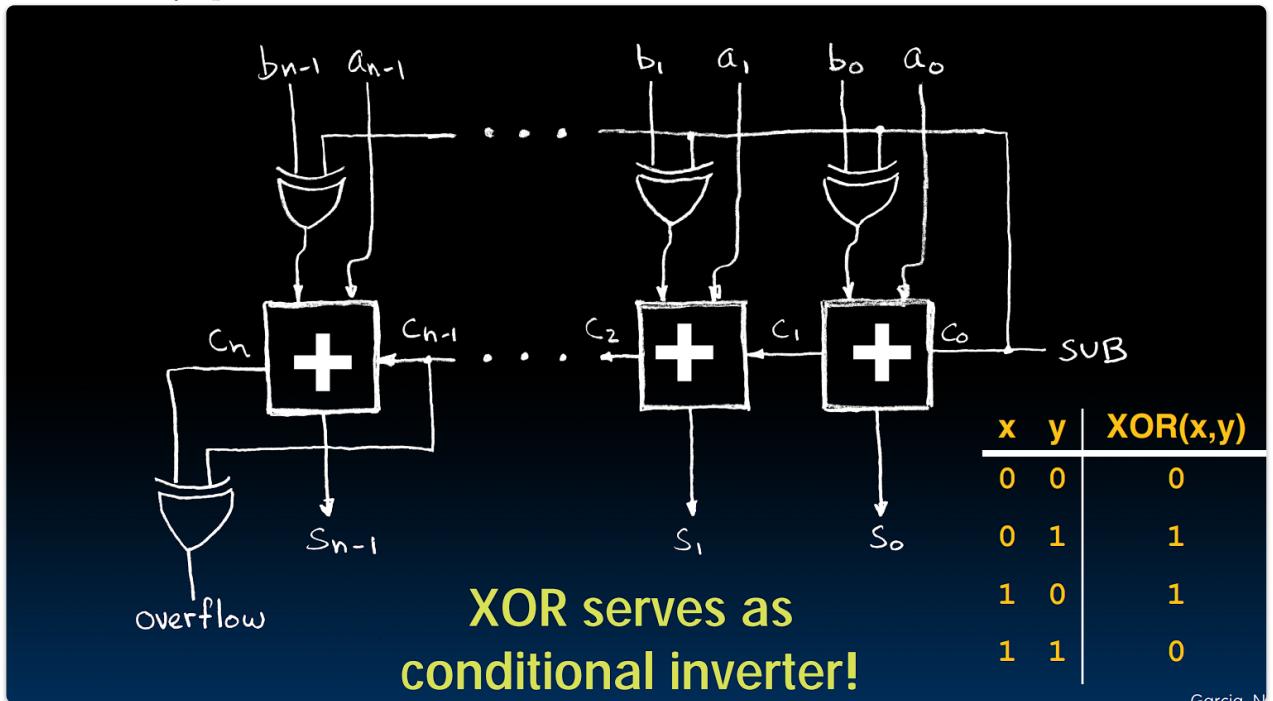
- Let's add
 - First unsigned
 - Then signed (Two's Complement)
 - When do the lowest 2 bits of sum not represent correct sum?
 - Is there a pattern of when this happens?
Hint: Check out the **carry-bit** and the sum-4s-column-bit
- Highest adder
 - $C_{in} = \text{Carry-in} = c_1, C_{out} = \text{Carry-out} = c_2$
 - No C_{out} or $C_{in} \rightarrow$ NO overflow!
 - C_{in} and $C_{out} \rightarrow$ NO overflow!
 - C_{in} , but no $C_{out} \rightarrow A, B \text{ both } > 0, \text{ overflow!}$
 - C_{out} , but no $C_{in} \rightarrow A \text{ or } B \text{ are } -2, \text{ overflow!}$

What operation is this?

overflow = $c_n \text{ XOR } c_{n-1}$ ✓

Subtractor

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



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