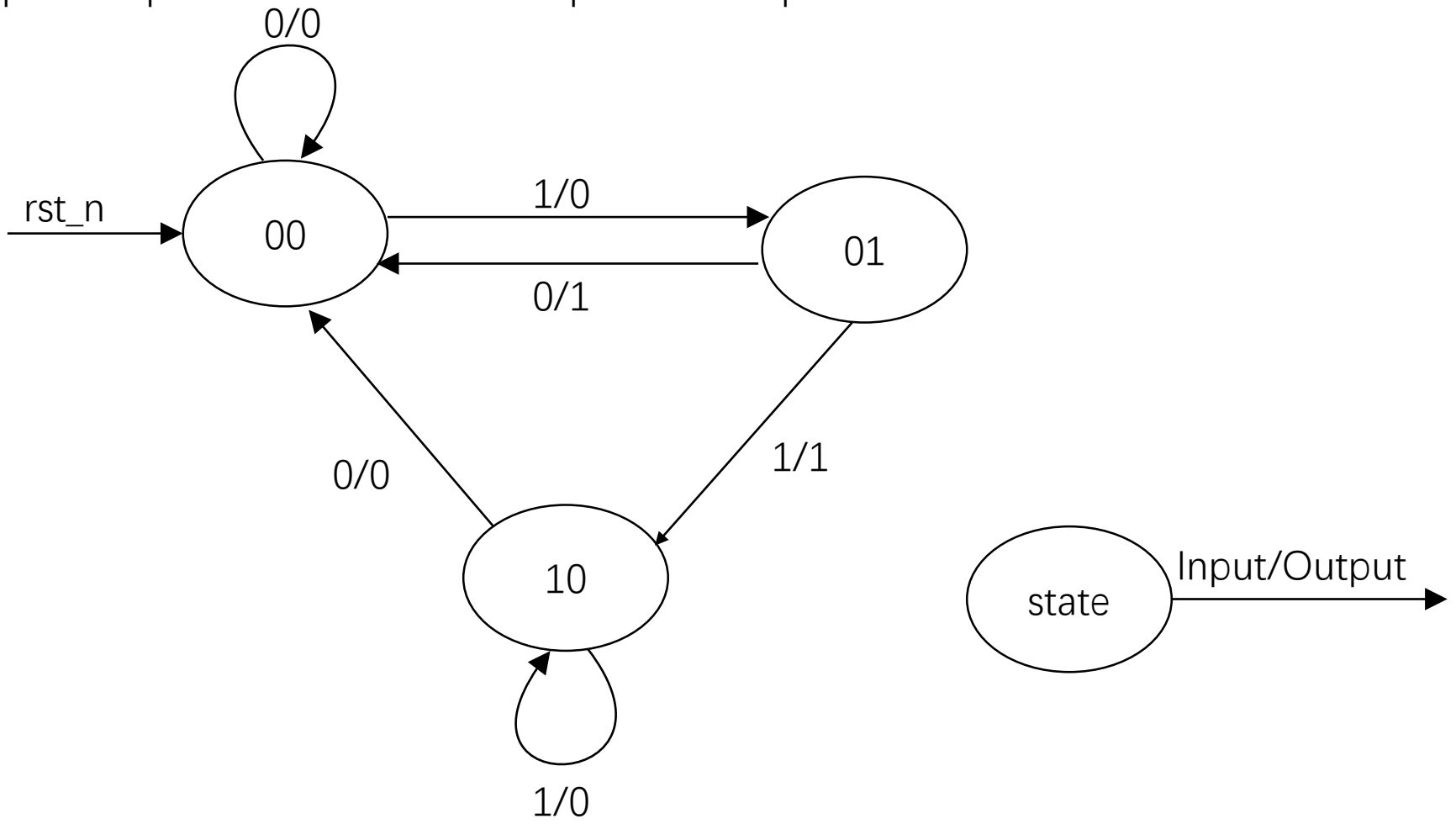
From a FSM to a Digital Circuit
&
Designing an Adder/Subtractor

Zhu Chen

The FSM Diagram

Function: Generate a positive pulse after the button is pushed. Keep 0 until the button is released and pushed again.

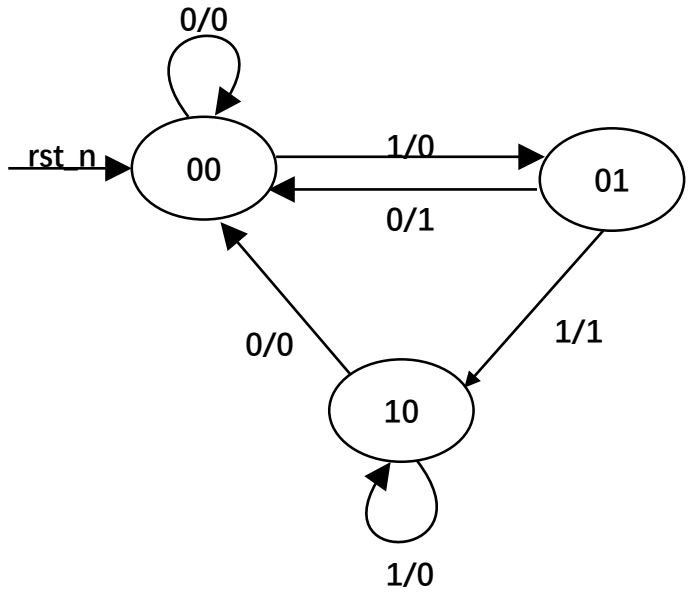


From http://www.allaboutcircuits.com/textbook/digital/chpt-11/finite-state-machines/

From FSM to Truth Table

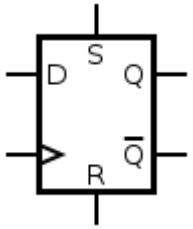
- List the input, current state, next state and output
  - Empty states represented by X, will be useful later

Current State		Input	Next State		Output
S_1	S_0	I	S_1'	S_0'	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	X	X	X
1	1	1	X	X	X

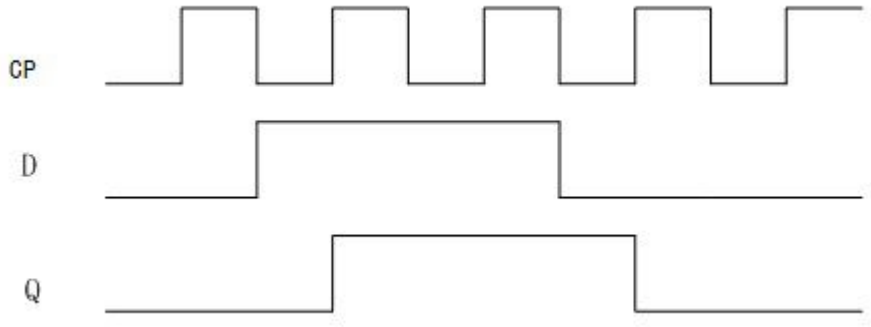


Represent the states

- Represent a state by the output of several flip-flops
  - Store the state value
  - Can be changed at the rising edge of the clock



- A D-Flip Flop
  - **Output = Input** at rising edge
  - $\bar{Q}$ is the complement of Q
  - Set S=R=0 to use it as expected
  - If S=R=1, output will always be 0



Simplifying the logical representations

- Simplify the circuit with a Karnaugh Map
  - Useful for 3 to 4 input variables (e.g., your next experiment)
  - Hamming Distance of neighbor columns/rows = 1 (e.g., 01 and 11)
  - Group 1's and X's together (1x2, 2x1, 2x2, 1x4, 4x1, etc.) and simplify
- S_0S_1 are inputs of the 2 D-Flip flops, $S_0'S_1'$ are the outputs

S_0'

S_0 \ S_1I	00	01	11	10
0	0	0	1	0
1	0	1	X	X

Cancel S_0

Cancel S_1

S_1'

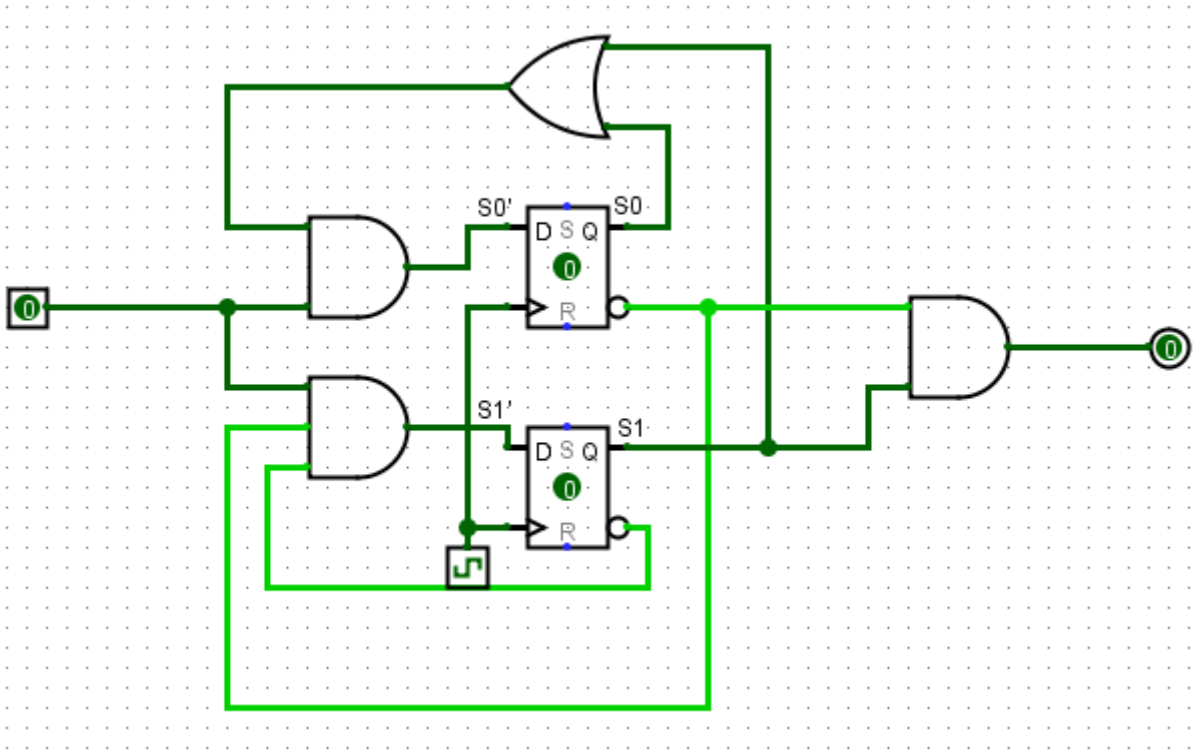
S_0 \ S_1I	00	01	11	10
0	0	1	0	0
1	0	0	X	X

Ignore redundant X

- $S_0' = S_0I + S_1I = (S_0 + S_1)I, S_1' = \overline{S_0}\,\overline{S_1}I, Y = \overline{S_0}S_1$

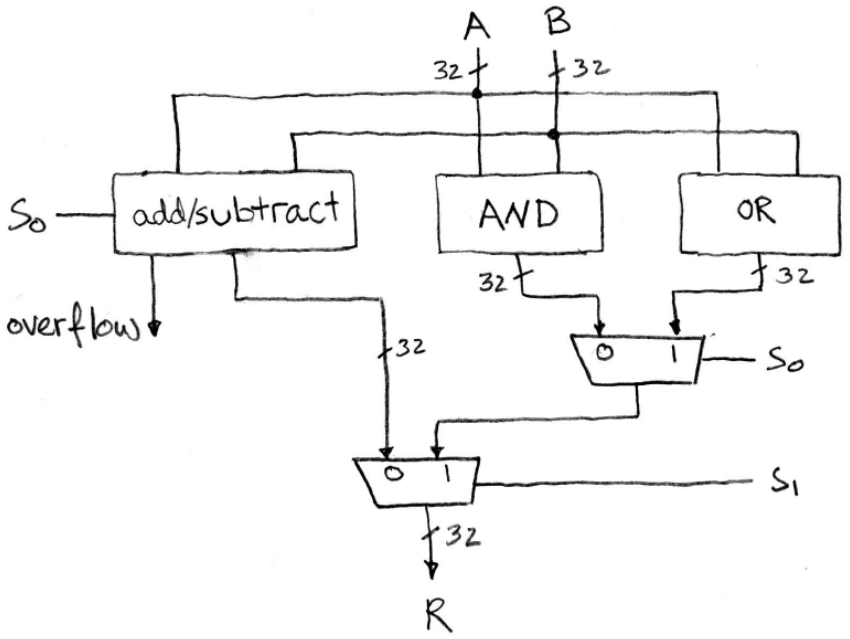
Connect the circuit

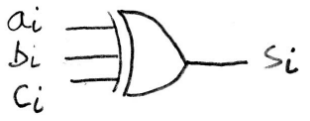
- $S_0' = S_0 I + S_1 I = (S_0 + S_1) I, S_1' = \overline{S_0}\,\overline{S_1} I, Y = \overline{S_0} S_1$

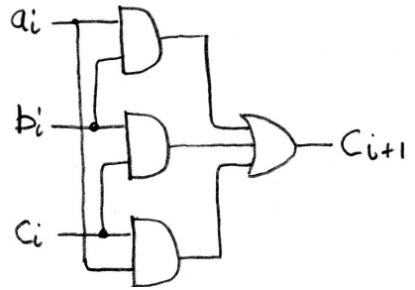


Designing an Adder/Subtractor

- An adder/subtractor is a circuit that is:
  - An adder when $S_0 = 0$
  - An subtractor when $S_0 = 1$
  - Required in ALU (top figure)
- Can be a cascade of 1-bit adders
- 1-bit adders(bottom figure):
  - Input: a, b, c_0 (carry of last adder)
  - Output: s, c_1 (sum and carry)
  - Design by truth table. $s = a \text{ xor } b \text{ xor } c$



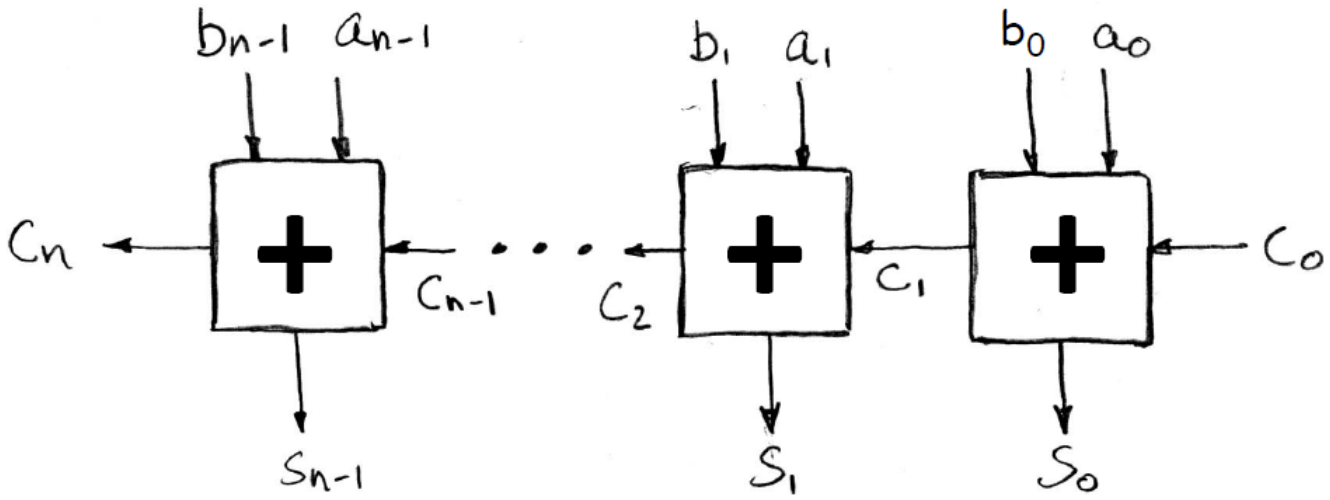




Cascade of 1-bit adders into an adder

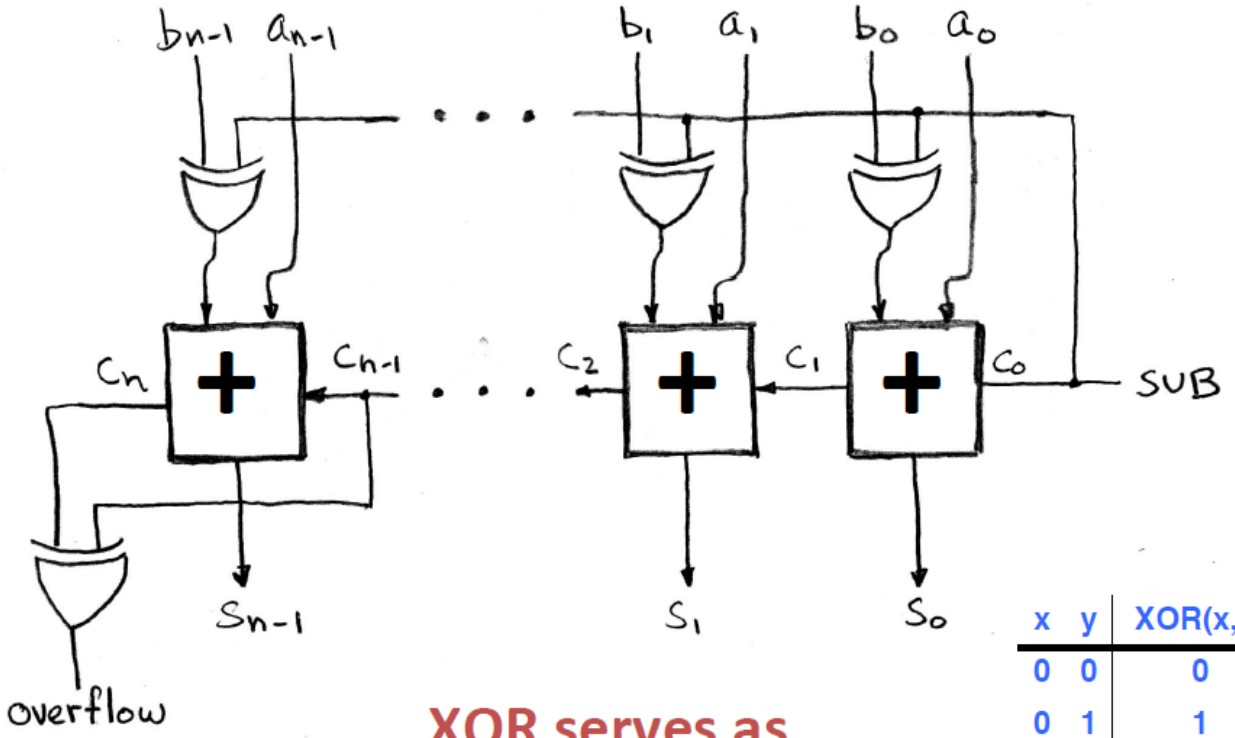
- Rather intuitive
- C_n to represent overflow

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



Cascade of 1-bit adders into a subtractor

- How come?



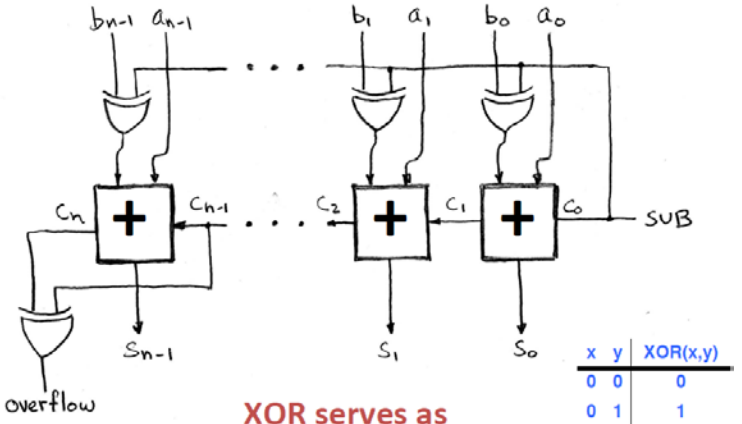
x	y	XOR(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

XOR serves as conditional inverter!

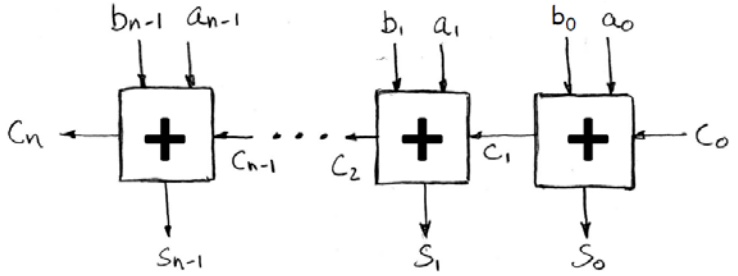
Cascade of 1-bit adders into a subtractor

- A, B are n-bit numbers with signs
- $\tilde{A}$ is the 2's complement, $\bar{A}$ is 1's complement
- Setting $SUB = 1$ to get $\bar{B}$
  - b_i xor $SUB = b_i$ when $SUB = 0$
  - b_i xor $SUB = \bar{b_i}$ when $SUB = 1$
- Subtract to addition: $A - B = A + \tilde{B}$
  - $-B = \tilde{B}$ considering only the lowest (n-1) bits:
    $(-B)_{n-1} = (2^{n-1} + (-B))_{n-1} = (2^{n-1} - 1 - B) + 1$

  Where $(\cdot)_{n-1}$ represents taking the lowest (n-1) bits



Overflow of n-bit adder



- Let A, B be the equivalent inputs to an n-bit adder
- If $A > 0, B > 0$
  - $a_{n-1} = b_{n-1} = 0 \Rightarrow c_n = 0$
  - If overflow, $c_{n-1} = 1$
- If $AB < 0$
  - Overflow won't happen since $|A + B| \leq |A|$

Overflow of n-bit adder

- If $A < 0, B < 0$, we actually add their complements $\bar{A}, \bar{B}$
  - $a_{n-1} = b_{n-1} = 1 \Rightarrow c_n = 1$
  - If overflow, $c_{n-1} = 0$ (a bit tricky here)
    - If $(A + B)_{n-1} = 2^{n-1}$, then $A + B = -2^n$, represented by $\mathrm{0b}100 \ldots 0$, not overflow
    - Overflows when $(A + B)_{n-1} > 2^{n-1}$
      $\Rightarrow 2^{n-1} - \tilde{A}_{n-1} + 2^{n-1} - \tilde{B}_{n-1} > 2^{n-1} \Rightarrow \tilde{A}_{n-1} + \tilde{B}_{n-1} < 2^{n-1} \Rightarrow c_{n-1} = 0$
- To sum up, the addition overflows if and only if c_n xor c_{n-1}=true

The End