中国科学院大学
University of Chinese Academy of Sciences

CS101

Systems Thinking

Modularization-1:
Combinational Circuits and Sequential Circuits

zxu@ict.ac.cn
zhangjialin@ict.ac.cn

Outline

- What is systems thinking?
- Three objectives of systems thinking
- Abstraction
- Modularization
  - Modularization and modules
  - Combinational circuits
    - Logic gates and combinational circuits
    - The information hiding principle
    - Adders
    - An adder-subtractor controlled by multiplexers
  - Sequential circuits
    - Types of memory cells and the D flip-flop
    - General organization of sequential circuit
    - A serial adder example
  - Instruction Set and Instruction Pipeline
  - Software Stack
- Seamless transition

These slides acknowledge sources for additional data not cited in the textbook

4.1 Modularization and modules

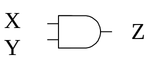
- Modularization is a systems thinking method, similar to divide-and-conquer in algorithmic thinking
  - Divide a system into multiple subsystems called modules
  - Compose modules into a system (higher-level abstraction)
- In a system with modularization
  - Two modules may be interconnected, but normally do not overlap
- Modularization is a special form of abstraction where the information hiding principle is followed
- Modularization, i.e., how to divide and compose a system, is an art, needing human imagination and creativity

- Understand how modularization works via a design journey
  - of higher and higher hardware subsystem abstractions
    - from designing gates to designing an instruction pipeline

4.1.1 Logic gates and combinational circuit

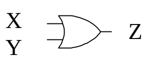
- Logic gates: electronic circuits realizing Boolean operators

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1



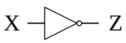
AND

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1



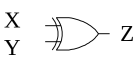
OR

X	Z
0	1
1	0



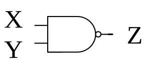
NOT

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0



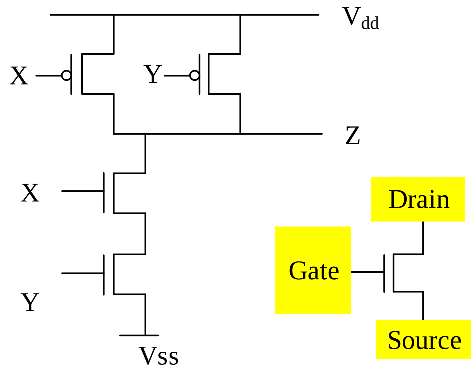
XOR

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



NAND

circle means NOT

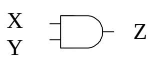


CMOS circuit for NAND

4.2.1 Logic gates and combinational circuit

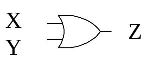
- Logic gates: electronic circuits realizing Boolean operators

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1



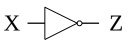
AND

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1



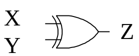
OR

X	Z
0	1
1	0



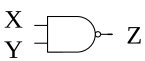
NOT

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

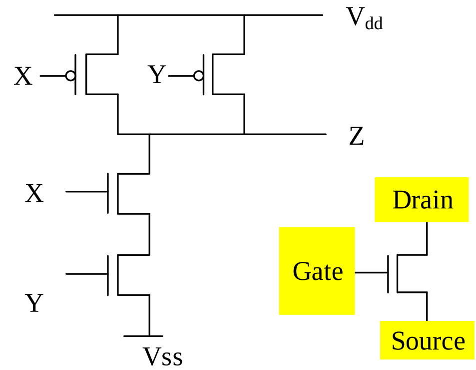


XOR

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



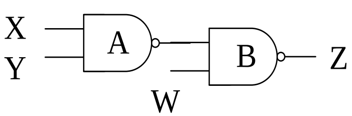
NAND

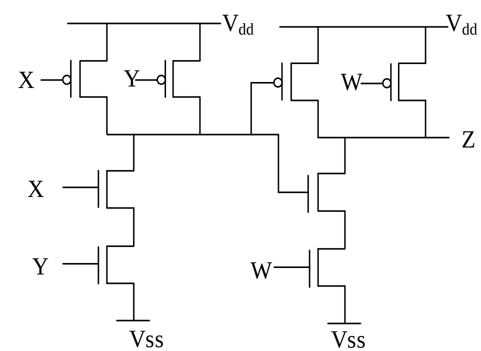


CMOS circuit for NAND

circle means NOT

$$Z = \overline{(\overline{X \cdot Y}) \cdot W}$$





4.2.1 Logic gates and combinational circuit

- Logic gates: electronic circuits realizing Boolean operators

AND

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

NOT

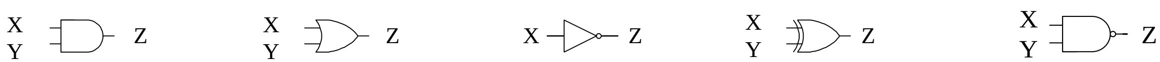
X	Z
0	1
1	0

XOR

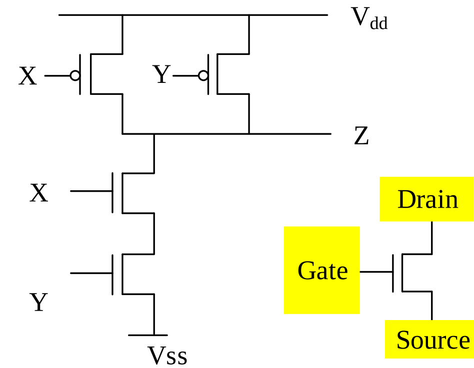
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

NAND

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



AND OR NOT XOR NAND

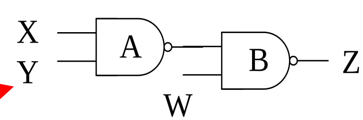


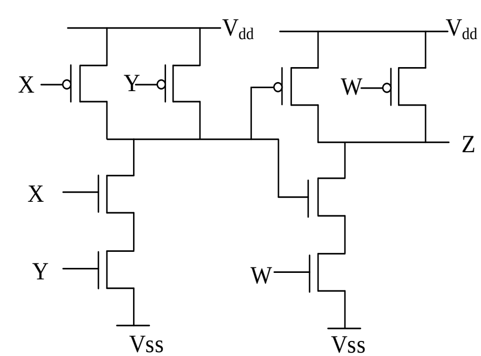
CMOS circuit for NAND

Vss: ground (0); Vdd: high voltage (1)

- A combinational circuit is comprised of interconnected gates without feedback wires
  - Any combinational circuit has a corresponding Boolean expression
  - Any Boolean expression has a corresponding combinational circuit
  - A combinational circuit can be shown as a logic diagram

$$Z = \overline{(\overline{X \cdot Y}) \cdot W}$$





CMOS circuit

4.2.1 Logic gates and combinational circuit

- Logic gates: electronic circuits realizing Boolean operators

AND

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

NOT

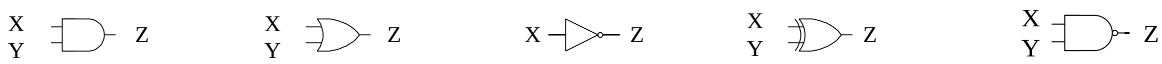
X	Z
0	1
1	0

XOR

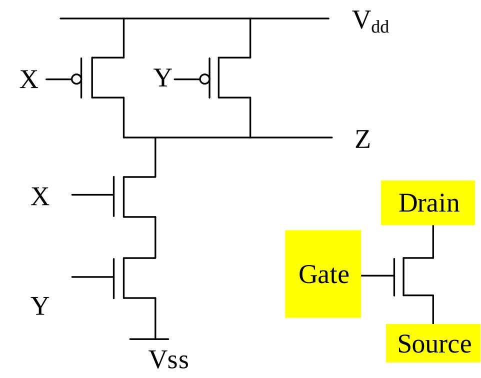
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

NAND

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



AND OR NOT XOR NAND

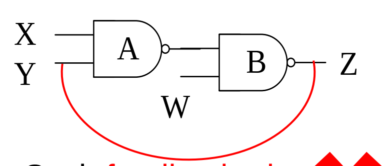


CMOS circuit for NAND

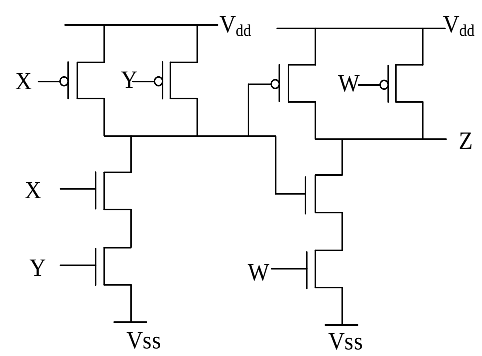
A combinational circuit is comprised of interconnected gates without feedback wires

- Any combinational circuit has a corresponding Boolean expression
- Any Boolean expression has a corresponding combinational circuit
- A combinational circuit can be shown as a logic diagram

$$Z = \overline{(\overline{X \cdot Y}) \cdot W}$$



Such feedback wire is not allowed

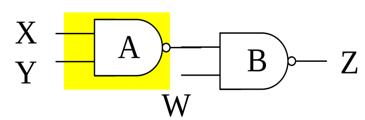


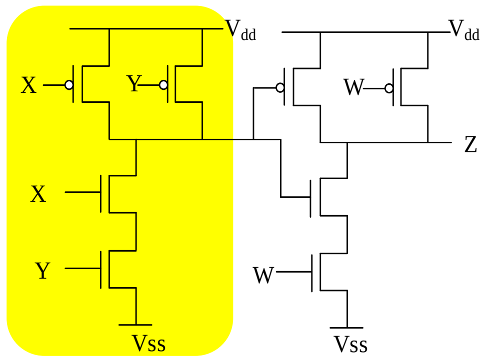
CMOS circuit

4.2.2 The information hiding principle

- A module only exposes its interface and visible behaviors, but hides internal details and internal behavior
- Three types of abstractions are shown of the same combinational circuit
  - Boolean expression
  - Logic diagram
  - CMOS circuit

$$Z = \overline{(\overline{X \cdot Y}) \cdot W}$$

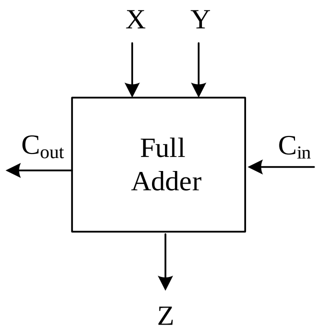




- The three yellow areas
  are different abstractions for the same thing
  - a 2-input-1-output NAND gate
- The Boolean expression and the logic diagram abstractions hide internal details of the CMOS implementation
- The former two are simpler and allow different implementations

4.2.3 Adders

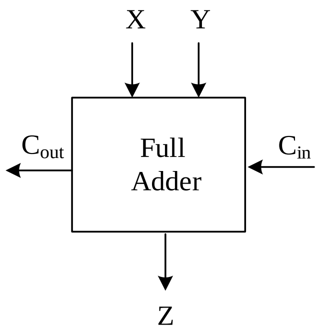
- Add two unsigned numbers X and Y to generate result sum Z
  - Consider the carry-in bit C_{in} and the carry-out bit C_{out}
  - Use the same algorithm we use when doing addition by pen and paper
- For one bit, design a full adder
  - Here, “full” means the adder considers carry-in and carry-out bits
  - **1-minute quiz**: given X, Y and C_{in}, what are the expressions of Z and C_{out}?



Full adder symbol

Full adder

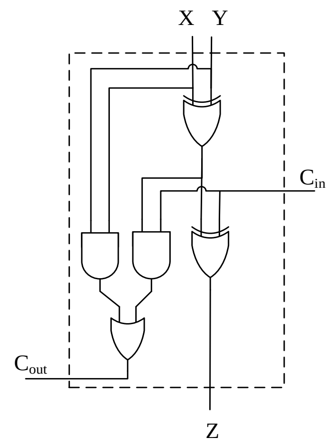
- Add two unsigned numbers X and Y to generate result sum Z
  - Consider the carry-in bit C_{in} and the carry-out bit C_{out}
  - Use the same algorithm we use when doing addition by pen and paper
- For one bit, design a full adder
  - Here, "full" means the adder considers carry-in and carry-out bits
  - **1-minute quiz**: given X, Y and C_{in}, what are the expressions of Z and C_{out}?
  - A: Derive the truth table from the manual addition method
    Then, derive the Boolean expressions for Z and C_{out}
  - $Z = X \oplus Y \oplus C_{in}$
  - $C_{out} = (X \cdot Y) + (X \oplus Y) \cdot C_{in}$



Full adder symbol

Cin	X	Y	Z	Cout
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

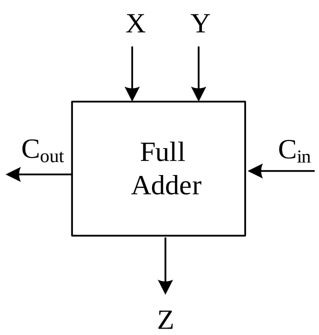
Truth table



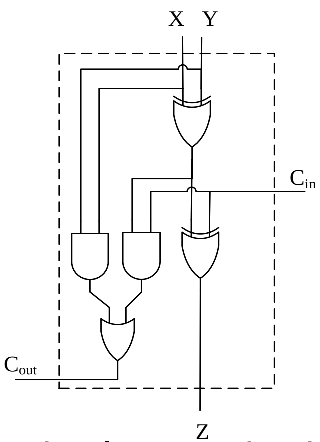
Implementation by gates

Ripple-carry adder

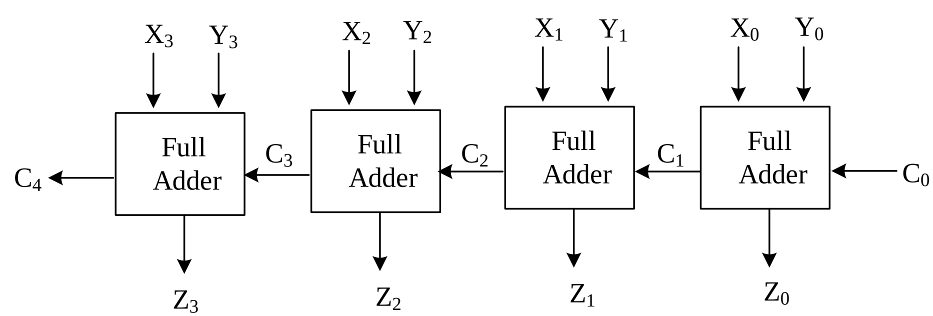
- Add two unsigned numbers X and Y to generate result sum Z
  - Consider the carry-in bit C_{in} and the carry-out bit C_{out}
  - Use the same algorithm we use when doing addition by pen and paper
- For n-bit, design an n-bit ripple-carry adder (assuming n=4 in example)
  - Use X+Y=1011+1001 = 10100 to verify that the 4-bit adder works correctly
  - **1-minute quiz**: How much time to do the addition? Use gate delay as the unit



Full adder symbol



Its implementation by gates



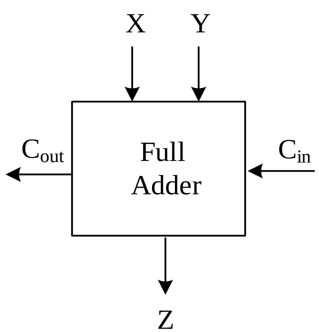
A 4-bit ripple-carry adder

Ripple-carry adder

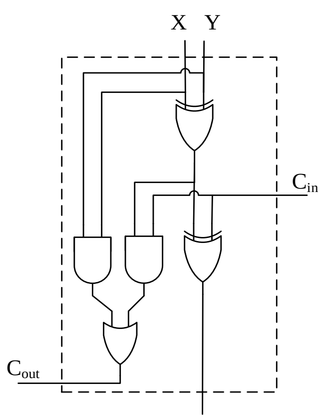
- Add two unsigned numbers X and Y to generate result sum Z
  - Consider the carry-in bit C_{in} and the carry-out bit C_{out}
  - Use the same algorithm we use when doing addition by pen and paper

- For n-bit, design an n-bit ripple-carry adder (assuming n=4 in example)
  - **1-minute quiz**: How much time to do the addition? Use gate delay as the unit
  - Answer 1: total delay ~ 3n; about 12 gate delays when n=4
    - Because each full adder needs 3 gate delays to generate carry-out

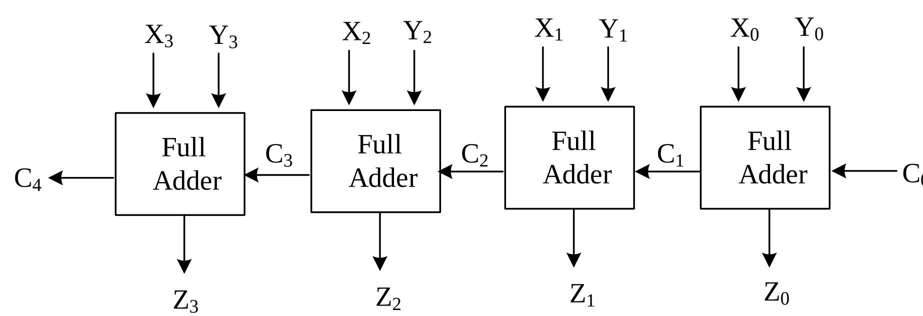
  - Is this answer correct?



Full adder symbol



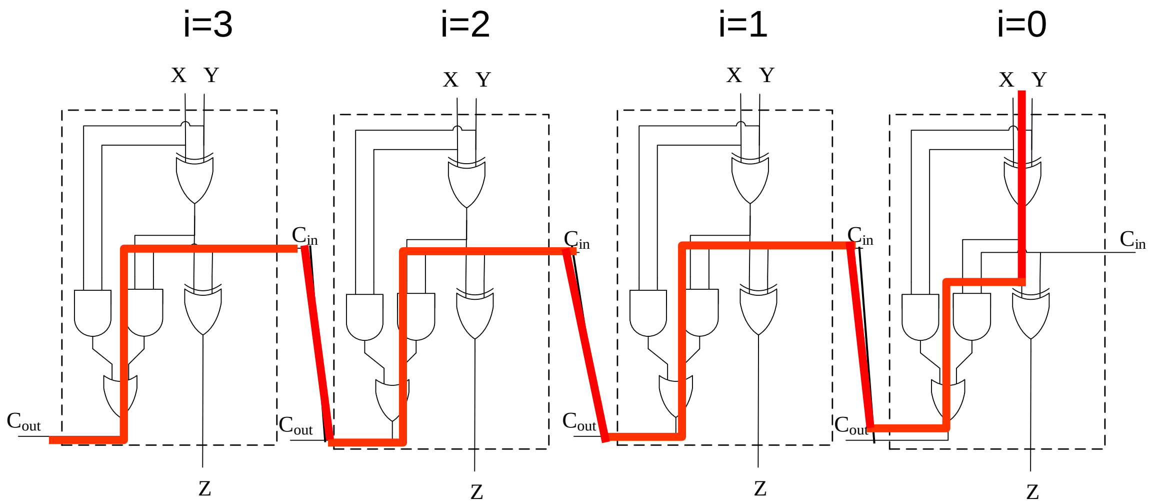
Its implementation by gates



A 4-bit ripple-carry adder

Ripple-carry adder

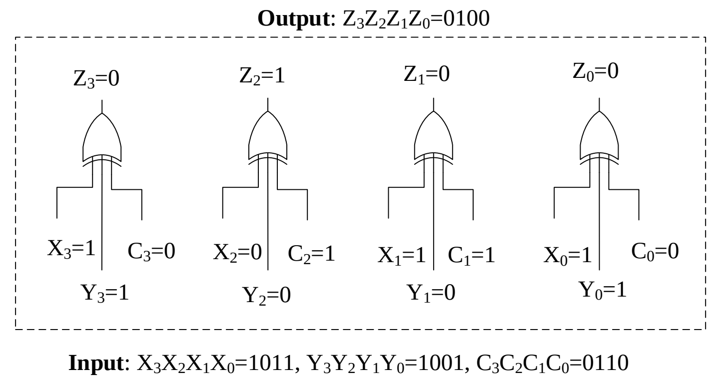
- Add two unsigned numbers X and Y to generate result sum Z
  - Consider the carry-in bit C_{in} and the carry-out bit C_{out}
  - Use the same algorithm we use when doing addition by pen and paper
- For n-bit, design an n-bit ripple-carry adder (assuming n=4 in example)
  - **1-minute quiz**: How much time to do the addition? Use gate delay as the unit
  - Answer 2: 2n+1. 9 gate delays for n=4
    - Only C1 needs 3 gate delays, and another Ci needs only 2 additional gate delays.
    - Why? Because for $1 \leq i < n,\ X_i \oplus Y_i$ is already generated when C_1 becomes available
      - Red line shows the longest path

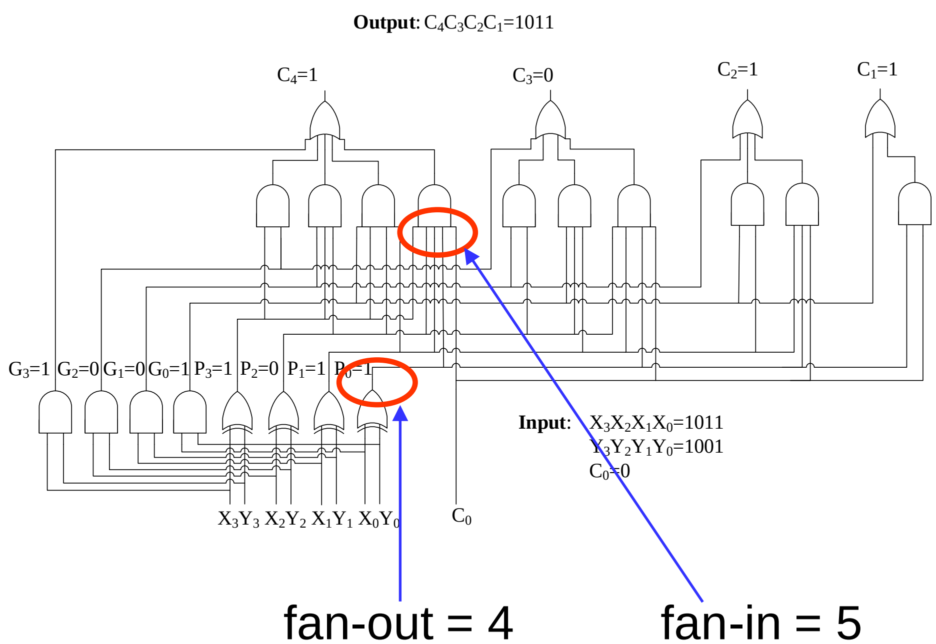


Expand to show the gate-level details

A much faster adder

- Instead of compute the carry bits one by one in *n* steps, we can compute all the *n* carry bits in parallel in one step
  - This step needs 3 gate delays
- Afterwards, the *n* sum bits are computed in parallel in one step
  - This step needs 1 gate delay
- 4 gate delays in total
  - Compared to 2*n* -1 gate delays for ripple-carry adder
- Constrained by fan-in and fan-out in practice
  - A gate can only safely receive a few inputs
  - A gate output cannot drive too many wires





4.2.4 A combinational circuit can compute multiple functions via selection by control signals

- An adder-subtractor controlled by a multiplexer (MUX in short)
  - Control signal S to select addition (S=0) or subtraction (S=1)
  - Subtraction is adding negative, e.g., 5-5 = 5+(-5)
    - The negative value of a number X is the two's complement of X
  - Let X = Y = 5, i.e., $X_3X_2X_1X_0 = Y_3Y_2Y_1Y_0 = 0101$
    - Then, X – Y = 5 + (-5) = 5 + Two's complement of 5

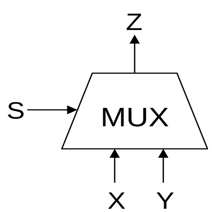
$$\rightarrow\ 0101 + \left(\overline{0}\,\overline{1}\,\overline{0}\,\overline{1} + 0001\right)$$

$$= \ 0101 + \ 1011$$

$$= 10000 = C_4\, Z_3\, Z_2\, Z_1\, Z_0$$

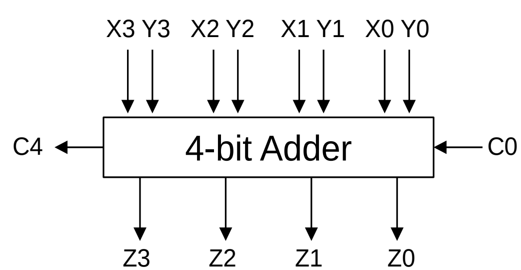
$$Z = \begin{cases} X & \text{if } S = 0 \\ Y & \text{if } S = 1 \end{cases}$$

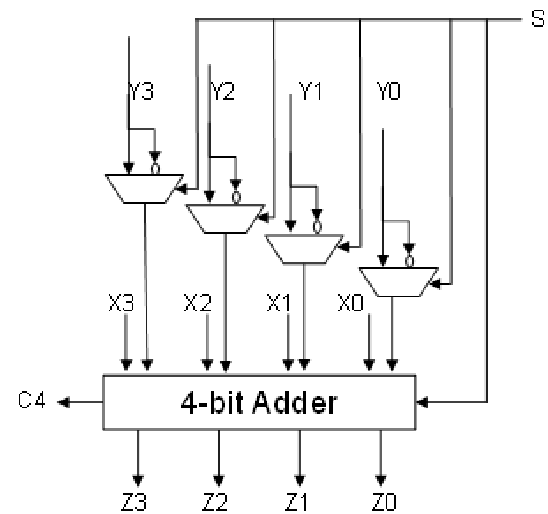
S	X	Y	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



S	Z
0	X
1	Y

A multiplexer





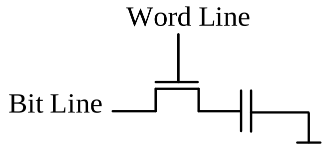
A two's complement 4-bit adder and an adder-subtractor

4.3 Sequential circuits

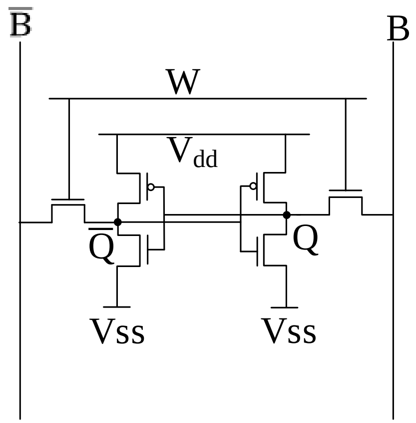
- **Sequential circuit = combinational circuit + state circuit**
  - With states, a system can execute multi-step computational processes
  - Each step computes two types of values
    - The current output values
    - The next state values
  - Both are computed from
    - The current input values
    - The current state values
- Correspondences to abstractions in logic thinking
  - A combinational circuit is equivalent to a Boolean expression
  - A sequential circuit is equivalent to an automaton
- States in hardware circuits are implemented by two types of basic circuits
  - Memory cells
  - flip-flops: logic circuits with feedback wires
    - We discuss only the D flip-flop

4.3.1 Types of memory technology

- Non-volatile memory (NVM): content is retained when power is off
  - ROM (read-only memory)
  - Read-write NVM
- Volatile memory: content is lost when power is off
  - DRAM (dynamic random access memory)
    - Simple and inexpensive
    - Needs to constantly refresh its content (once every 7.8-128 μs)
    - Because the capacitor leaks electricity after charging
  - SRAM (static random access memory)
    - More complex but avoid refreshing overhead
    - Faster but more expensive than DRAM



DRAM cell
1 transistor and
1 capacitor



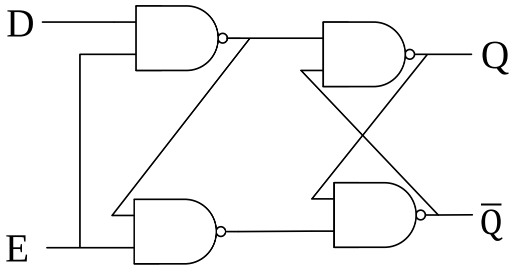
SRAM cell
6 transistors

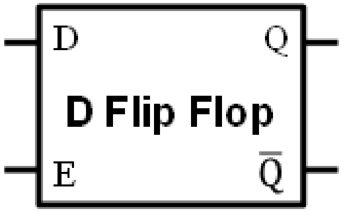
Type	Latency	Price $/GB
Register	100s ps	N/A
SRAM	100s ps ~ 10 ns	$100's ~1000s /GB
DRAM	10s ~ 100 ns	$2~4 /GB
NVM: Main Memory	100 ns ~ 10 μs	$6 / GB
NVM: SSD	10 μs ~ 1 ms	$0.1~0.2 / GB
Hard Disk: HDD	> A few ms	$0.02 / GB

4.3.2 D flip-flop

- We can also use gates to form state circuits
  - With feedback wires
- A common type is the delay flip-flop, or D flip-flop
  - 2-input-2-output, where the 2 states (outputs) are negation of each other
    - Input D is data input; Input E (enable signal) is often the clock signal CLK
  - Functionality: when enabled, Q is D delayed one clock cycle
    - When E=0, Q remains the same; when E=1, the new Q will be the current D

E	D	Q	Q_{next}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

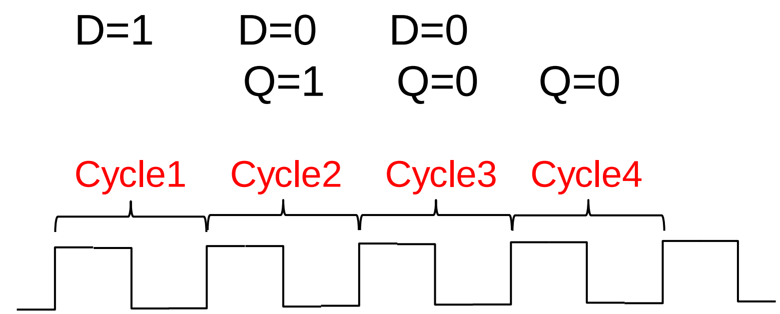




D flip-flop: Truth table Internal logic diagram Symbol

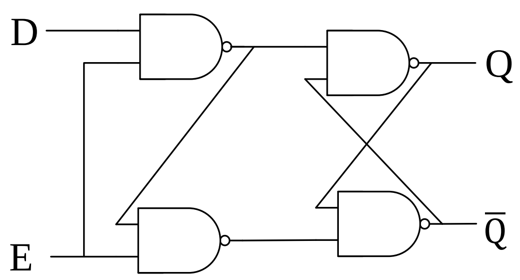
4.3.2 D flip-flop

- A common type is the delay flip-flop, or D flip-flop
  - 2-input-2-output, where the 2 states (outputs) are negation of each other
    - Input D is data input; Input E (enable signal) is often the clock signal CLK
  - Functionality: when enabled, Q is D delayed one clock cycle

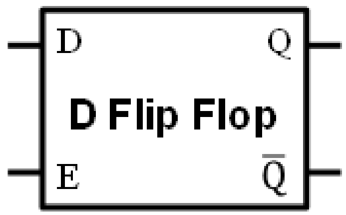


E	D	Q	Q_{next}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

D flip-flop: Truth table



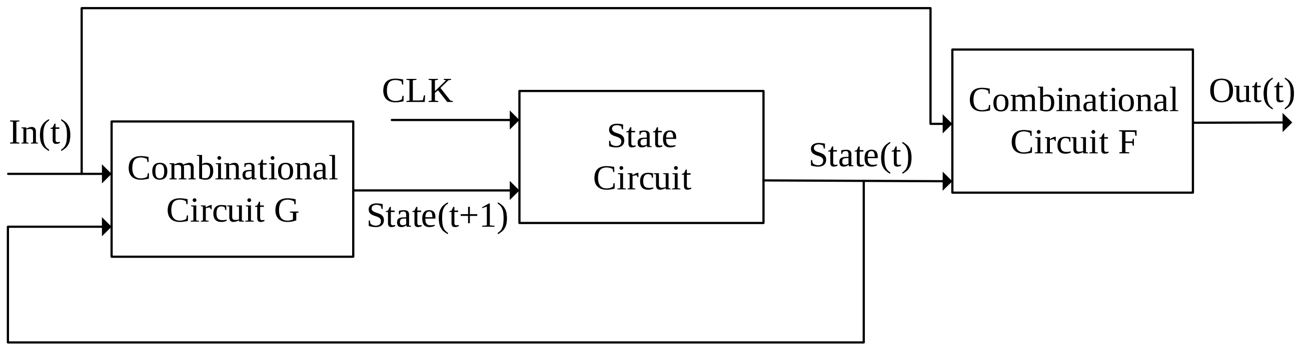
Internal logic diagram



Symbol

4.3.3 General organization of sequential circuits

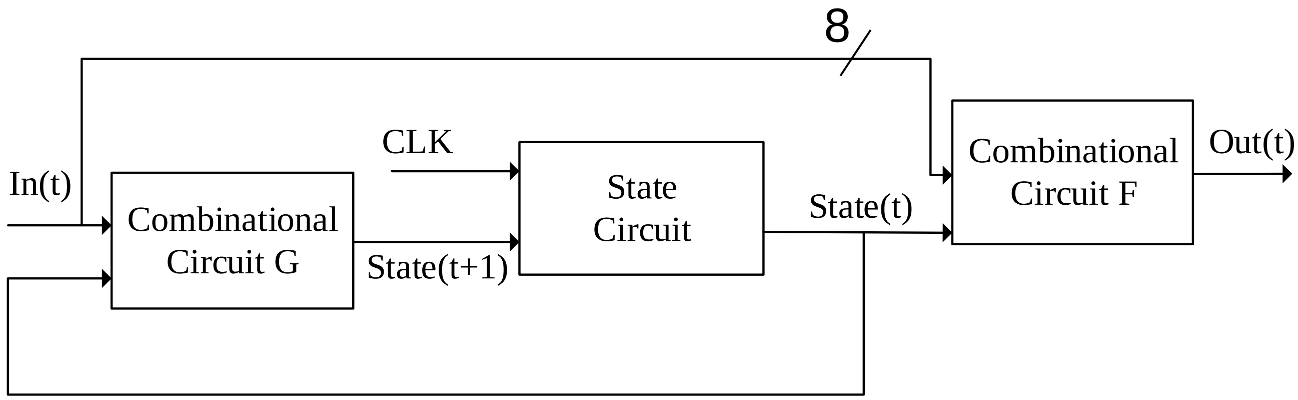
- Any sequential circuit can be organized as shown below
  - Comprised of two combinational circuits and a state circuit
  - Driven by a clock signal CLK
  - The state circuit consists of one or more D flip-flops
  - The output circuit F: Out(t) = F(In(t), State(t))
  - The next-state circuit G: State(t+1) = G(In(t), State(t))
- This is the logic diagram for a sequential circuit
  - Also called synchronous sequential circuit
    - Because the sequential circuit is synchronized by the clock signal



4.3.3 General organization of sequential circuits

- Any sequential circuit can be organized as shown below
  - Comprised of two combinational circuits and a state circuit
  - Driven by a clock signal CLK
  - The state circuit consists of one or more D flip-flops
  - The output circuit F: Out(t) = F(In(t), State(t))
  - The next-state circuit G: State(t+1) = G(In(t), State(t))

  - An input or output may have multiple bits



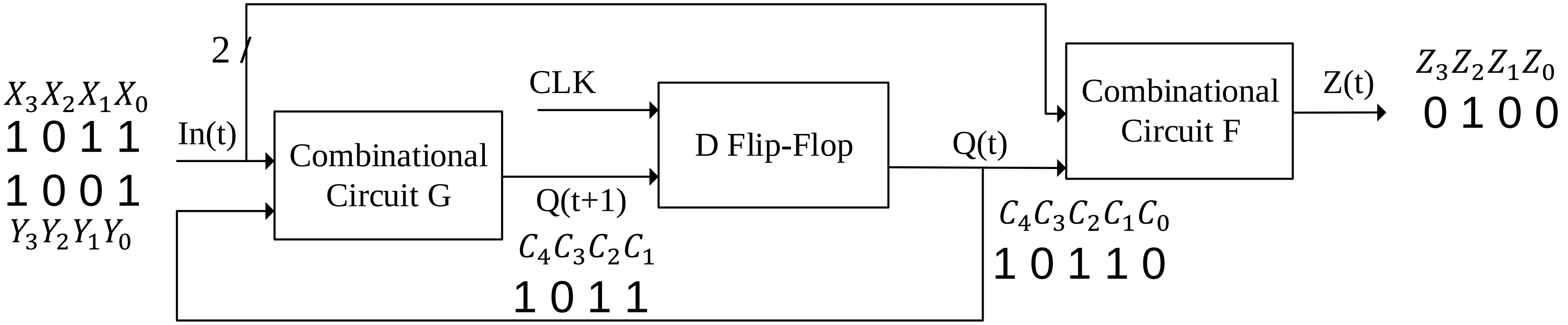
We will go through the process of designing a 4-bit serial adder

4.3.4 Design a 4-bit serial adder

- To perform $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$ in 4 steps
  - Each step performs a full addition
- First, **design** the adder
- Second, **verify** the correctness of the adder
- Use of an example is helpful in both design and verification
  - X + Y = 11_{10} + 9_{10}
    = 1011_2 + 1001_2 = 10100_2
    = 20_{10} = 4_{10} and overflow
    = 0100_2 and overflow
  - Therefore, $Z_3Z_2Z_1Z_0 = 0100$ and the carry-bit $C_4 = 1$

Design process

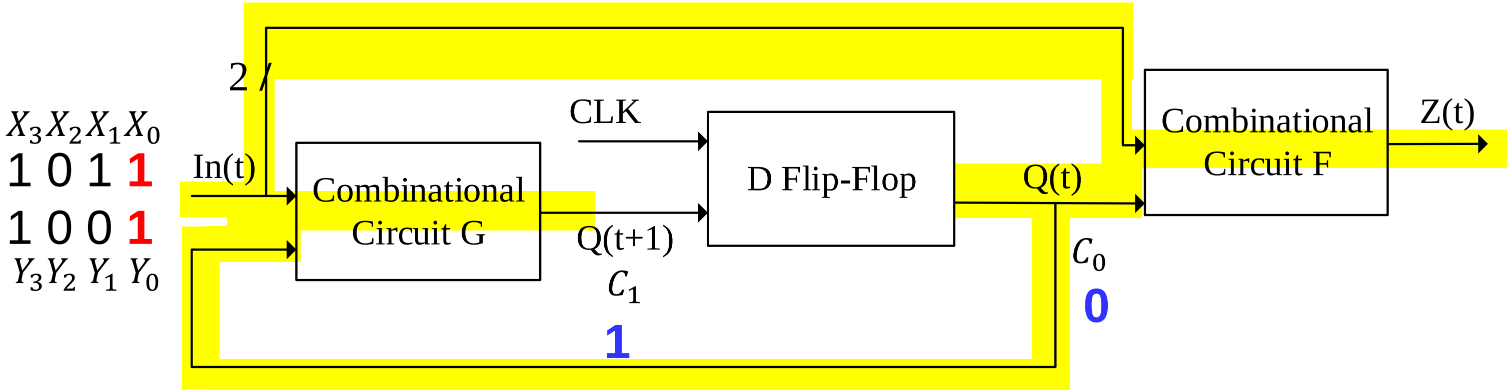
- Perform $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$ in 4 steps
  - Example: X + Y = 11_{10} + 9_{10} = 1011_2 + 1001_2 = **10100_2** = 20_{10} = "4_{10} and overflow"
- Design process
  1. Q: how many bits (or D flip-flops) are needed for the state circuit?
     - A: 1 bit to hold the carry bit. Only one D flip-flop is needed. Denote the state as Q.
  2. Draw the logic diagram for the sequential circuit
     - How?
     - Some students directly draw it by applying the principle of serial addition
       - Those students can skip the next 5 slides
     - Some prefer to understand behavior of the sequential circuit (the figure below) by executing X + Y (serial addition) using the paper+pencil method



Design process

- Perform $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$ in 4 steps
  - Each step is a clock cycle
  - Each step performs a full addition
- Use the paper+pencil method
  - Step 1 (clock cycle 1) is shown in bold. Note that initial state Q(0) = **C_0 = 0**
  - Given C_0 = 0, X_0 = 1, Y_0 = 1, circuits F, G should output Z_0 = 0, C_1 = Q(1) = 1

	1 0 1 **1**	= X
+	1 0 0 **1**	= Y
	0	= C
	0	= Z

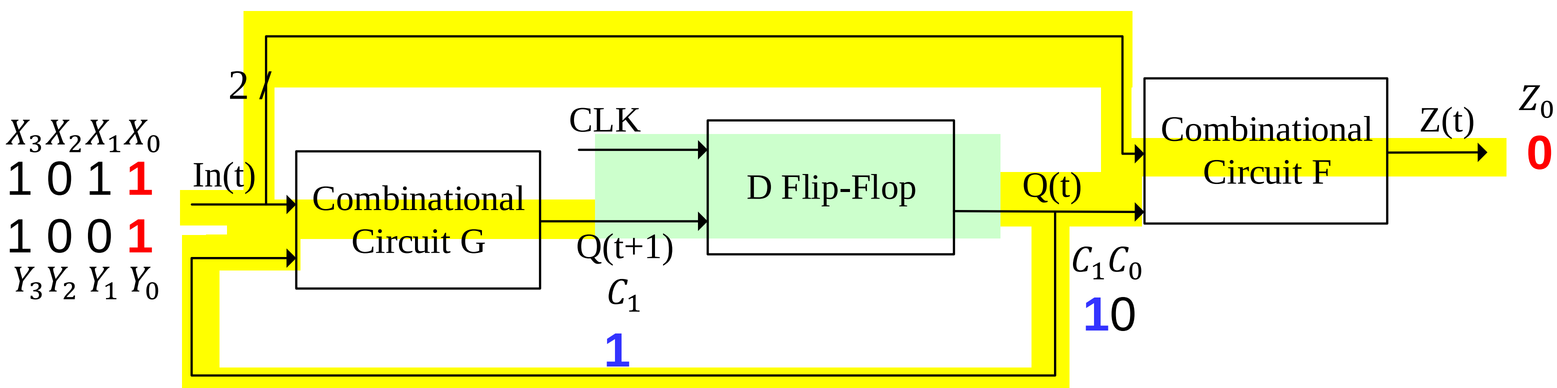


Design process

- Perform $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$ in 4 steps
  - Each step takes a clock cycle to complete
  - Each step performs a full addition
- Use the paper+pencil method
  - Step 1 (clock cycle 1) is shown in bold. Note that Q(0) = **C_0 = 0**
    - Given C_0 = 0, X_0 = 1, Y_0 = 1, circuits F, G should output Z_0 = 0, C_1 = Q(1) = 1
    - At the final moment of clock cycle 1 (and the beginning of cycle 2), the D flip-flop output Q(t) takes the value of the D flip-flop data input Q(t+1)

1 0 1 **1** = X
\+ 1 0 0 **1** = Y

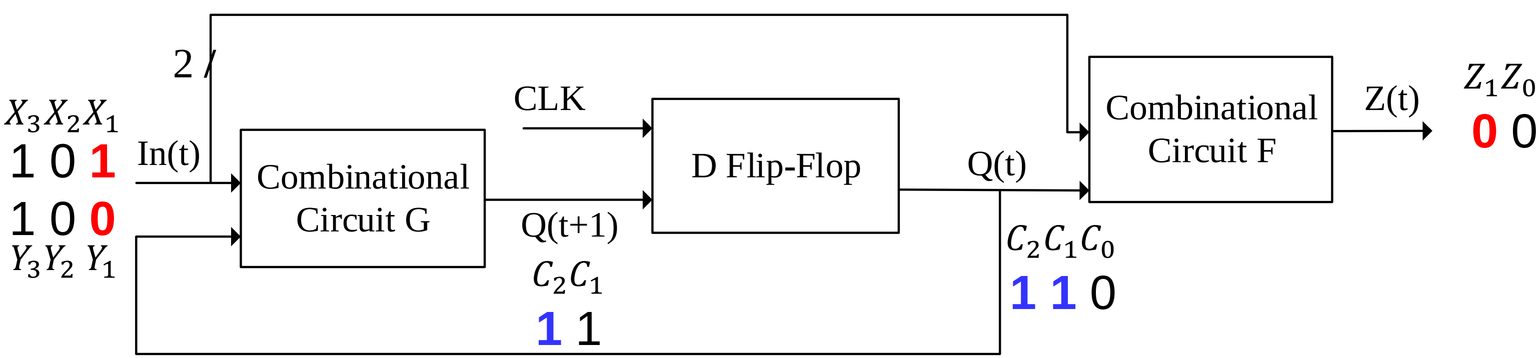
1 0 = C
0 = Z



Design process

- Perform $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$ in 4 steps
  - Each step performs a full addition
- Use the paper+pencil method
  - Step 2 (clock cycle 1) is shown in bold. Note that Q(1) = **C_1 = 1**

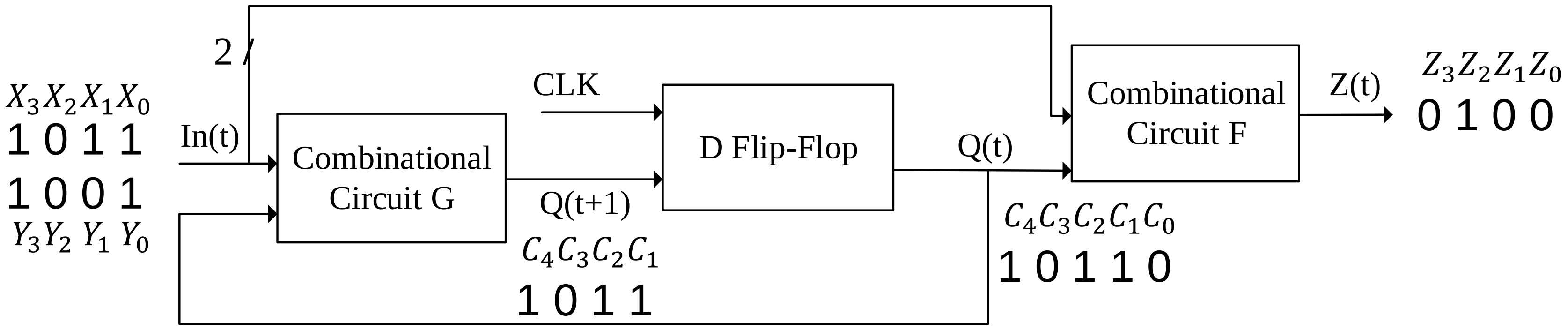
	1	0	**1**	1	= X
+	1	0	**0**	1	= Y
		1	**1**	0	= C
			0	0	= Z



Design process

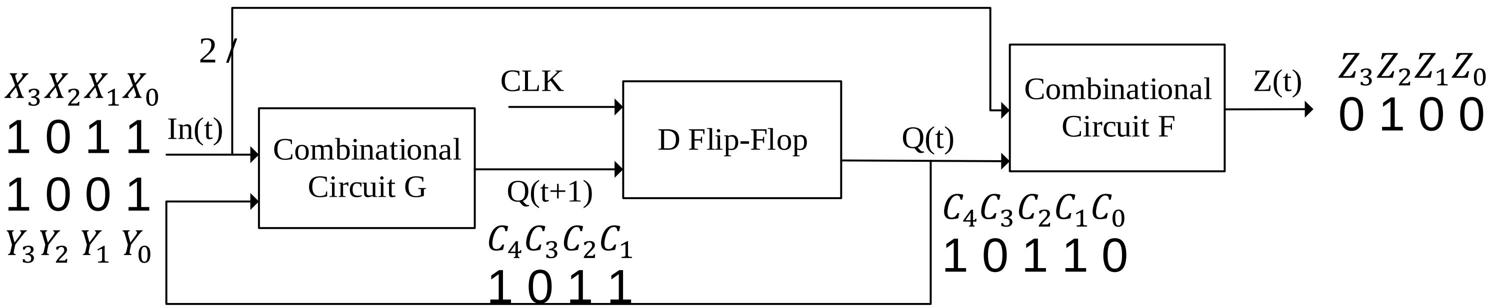
- Perform $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$ in 4 steps
  - Each step performs a full addition
  - Continue to do Step 3 and 4

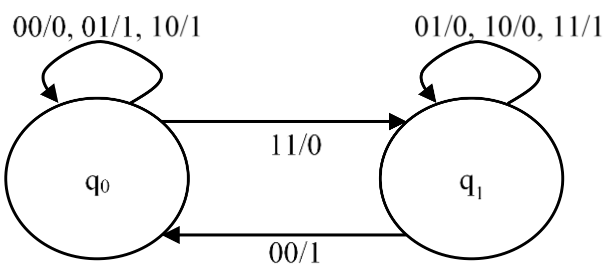
```
    1 0 1 1 = X
+   1 0 0 1 = Y
-----------------
  1 0 1 1 0 = C
    0 1 0 0 = Z
```



Design process

- Draw the logic diagram for the sequential circuit
  - Derive the state-transition diagram



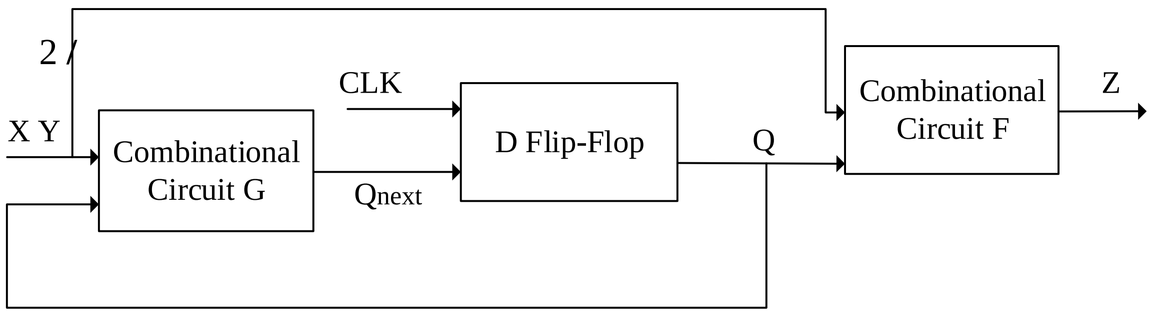


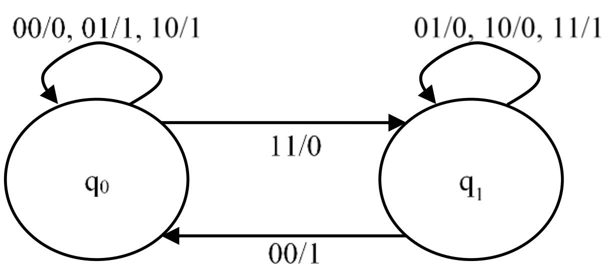
q_0 denotes C=0
q_1 denotes C=1

	1 0 1 1	= X
+	1 0 0 1	= Y
	1 0 1 1 0	= C
	0 1 0 0	= Z

Design process

- Derive the state-transition diagram (remember that Q denotes carry C)
- Derive the truth table and Boolean expressions for F, G





q_0 denotes C=0
q_1 denotes C=1

Q	X	Y	Z	Q_{next}
q_0	0	0	0	q_0
q_0	0	1	1	q_0
q_0	1	0	1	q_0
q_0	1	1	0	q_1
q_1	0	0	1	q_0
q_1	0	1	0	q_1
q_1	1	0	0	q_1
q_1	1	1	1	q_1

$$Z = F(X, Y, Q) = X \oplus Y \oplus C$$

$$Q_{next} = G(X, Y, Q) = (X \cdot Y) + (X \oplus Y) \cdot Q$$

Verification process

- Given $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, $C_0 = 0$, verify the resulting sequential circuit, noting that $\mathrm{F{:}}\ \mathrm{Z} = \mathrm{X} \oplus \mathrm{Y} \oplus \mathrm{C}$; $\mathrm{G{:}}\ \mathrm{Q_{next}} = (\mathrm{X} \cdot \mathrm{Y}) + (\mathrm{X} \oplus \mathrm{Y}) \cdot \mathrm{Q}$
- Step 1:
  - $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0 = \mathbf{0}$
  - $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = \mathbf{1}$
- Step 2:
  - $Z_1 = X_1 \oplus Y_1 \oplus C_1 = 1 \oplus 0 \oplus 1 = \mathbf{0}$
  - $C_2 = (X_1 \cdot Y_1) + (X_1 \oplus Y_1) \cdot C_1 = (1 \cdot 0) + (1 \oplus 0) \cdot 1 = \mathbf{1}$
- Step 3:
  - $Z_2 = X_2 \oplus Y_2 \oplus C_2 = 0 \oplus 0 \oplus 1 = \mathbf{1}$
  - $C_3 = (X_2 \cdot Y_2) + (X_2 \oplus Y_2) \cdot C_2 = (0 \cdot 0) + (0 \oplus 0) \cdot 1 = \mathbf{0}$
- Step 4:
  - $Z_3 = X_3 \oplus Y_3 \oplus C_3 = 1 \oplus 1 \oplus 0 = \mathbf{0}$
  - $C_4 = (X_3 \cdot Y_3) + (X_3 \oplus Y_3) \cdot C_3 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = \mathbf{1}$

The final result is $Z_3Z_2Z_1Z_0 = 1011$, with $C_4 = 1$ indicating overflow