SI100B
Introduction to Information Science and Technology (Electrical Engineering)

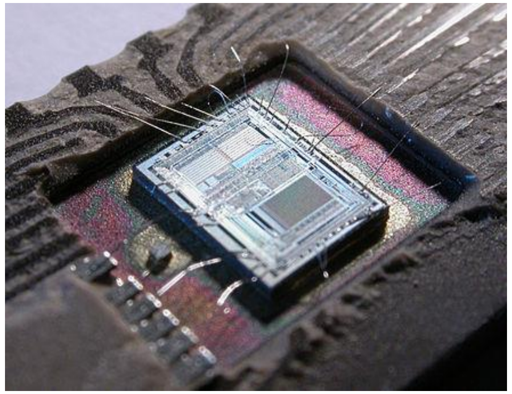
Lecture #3 (Digital) Combinational Logic Circuits

Instructor: Junrui Liang (梁俊睿)

Oct. 8th, 2021

The Theme Story

Devices (1)



Circuits (1/4)

Systems (2)

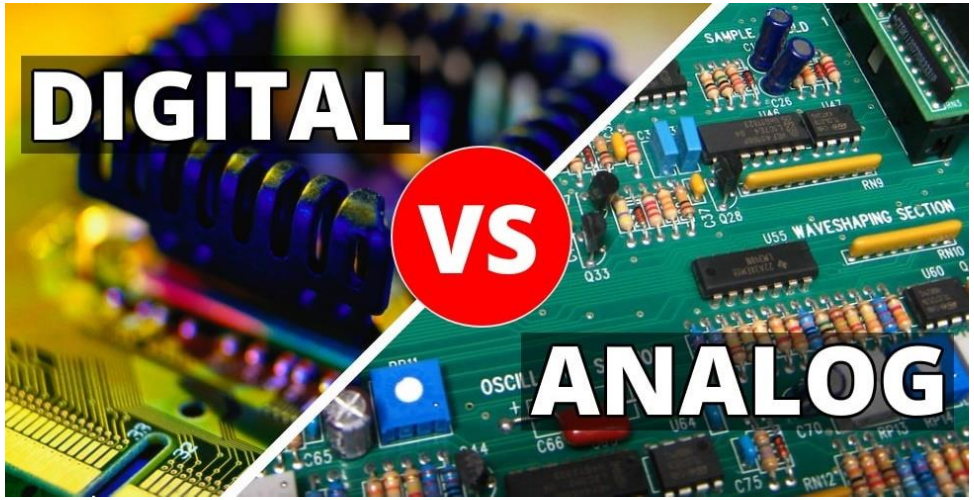
(Pictures are from the Internet)

Study Purpose of Lecture #3

- 哲学（bao'an）三问
  - Who are you?
  - Where are you from?
  - Where are you going?

To answer those questions throughout your life

- In this lecture, we ask
  - How many categories of circuits are there?
  - Why digital circuits 数字电路 won in computation & communication?
  - How to build combinational logic circuits 组合逻辑电路?



Lecture Outline

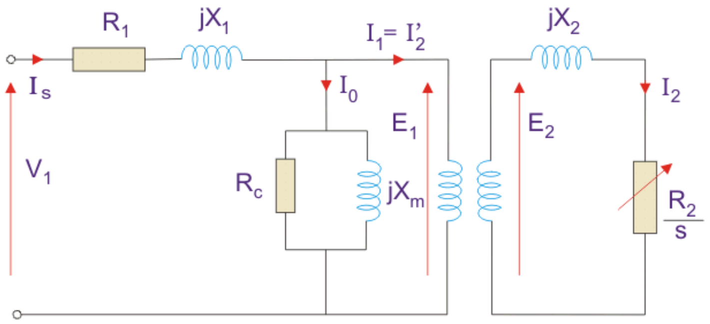
1. Circuit categories 电路种类
2. Boolean logic 布尔逻辑
3. Logic gates 逻辑门电路
4. Combinational logic circuit 组合逻辑电路
   - Example: majority circuit 投票选举电路实例

---------------- (break) ----------------

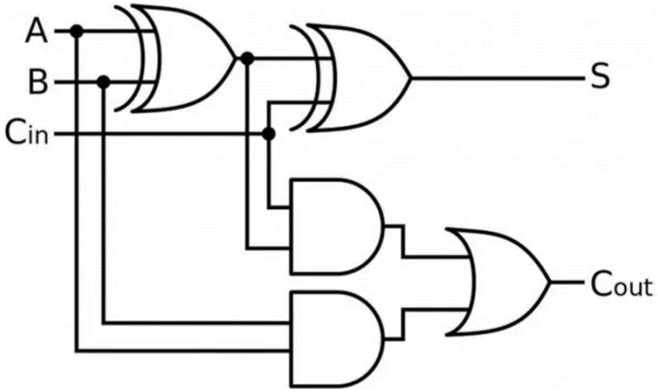
5. Combinational Logic Circuits Design 组合逻辑电路设计
   - Boolean algebra 布尔代数
   - Truth table 真值表
   - Karnaugh map 卡诺图
   - Example: decoder design 解码器设计实例

Four typical circuit categories

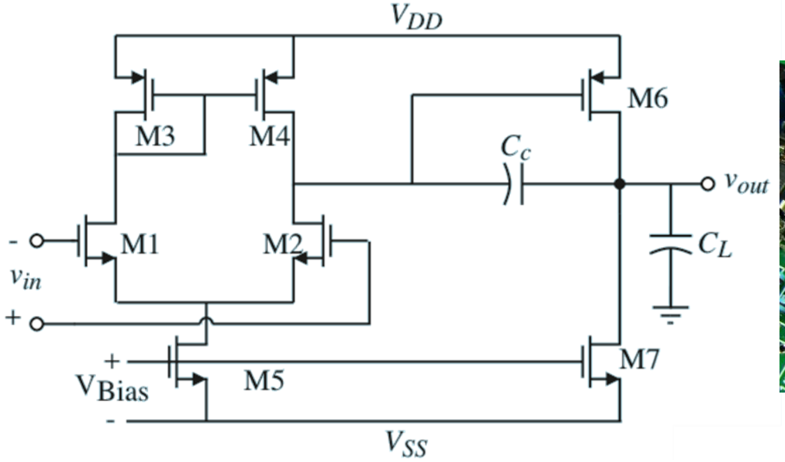
- Passive 被动/无源

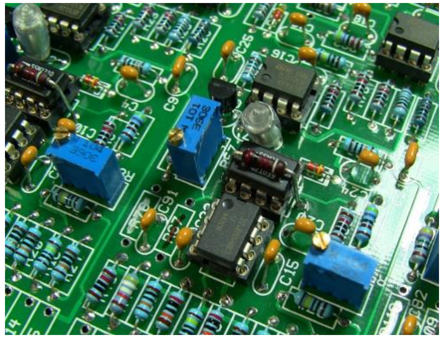


- Digital 数字

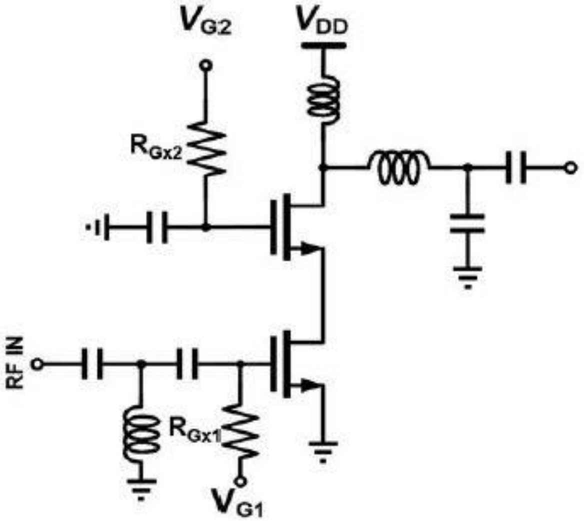


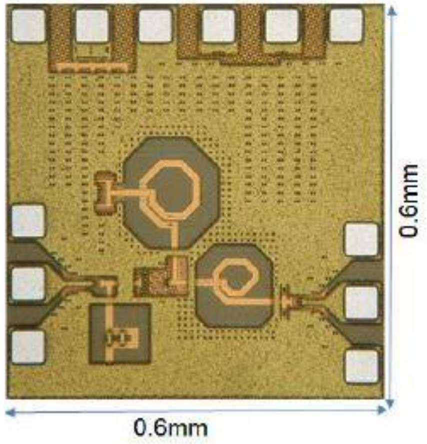
- Analog 模拟



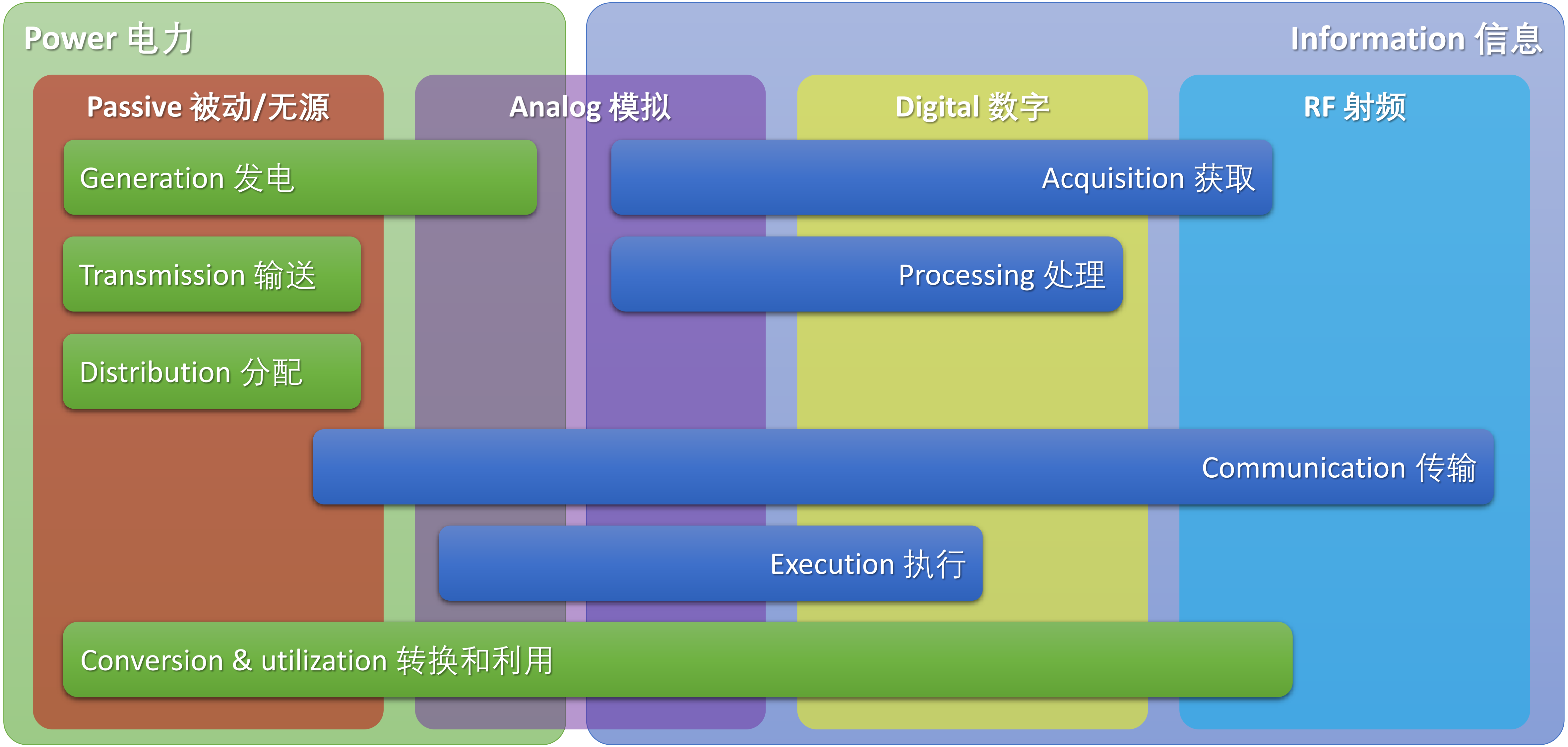


- RF 射频



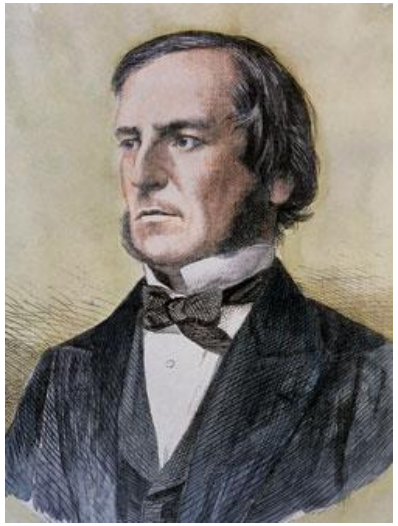


Electrical circuits for different application purposes

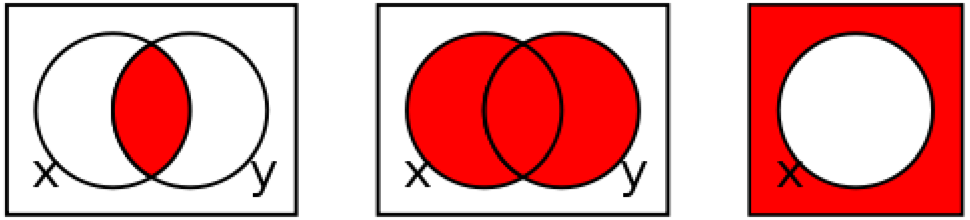


Boolean logic

- George Boole
  (1815 - 1864)
- Values
  - Boolean algebra allows only two values—0 and 1
  - The beauty of the *digital abstraction* is that digital designers can focus on 1's and 0's, ignoring whether the Boolean variables are physically represented with specific voltages, rotating gears, etc.



- Basic operations
  - AND (conjunction) 与 xy
  - OR (disjunction) 或 $x+y$
  - NOT (negation) 非 $\overline{x}$



x∧y　　x∨y　　¬x

Logic 0	Logic 1
False	True
Off	On
LOW	HIGH
No	Yes
Open switch	Closed switch

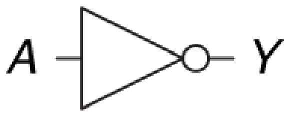
x	y	$x \wedge y$	$x \vee y$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	1

x	$\neg x$
0	1
1	0

Basic logic gates

- NOT gate

NOT

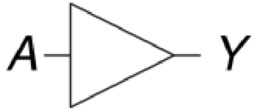


$Y = \bar{A}$

A	Y
0	1
1	0

- Buffer

BUF

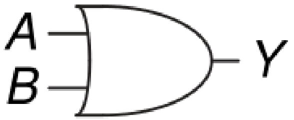


$Y = A$

A	Y
0	0
1	1

- OR gate

OR

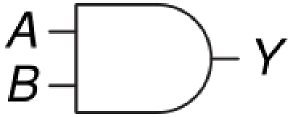


$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

- AND gate

AND



$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

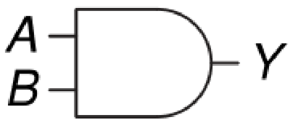
AND4

$Y = ABCD$

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

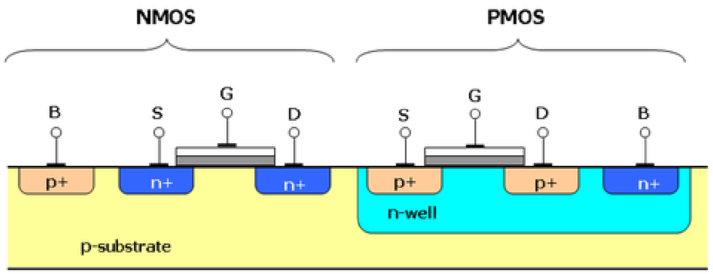
Logic gate implementation

AND

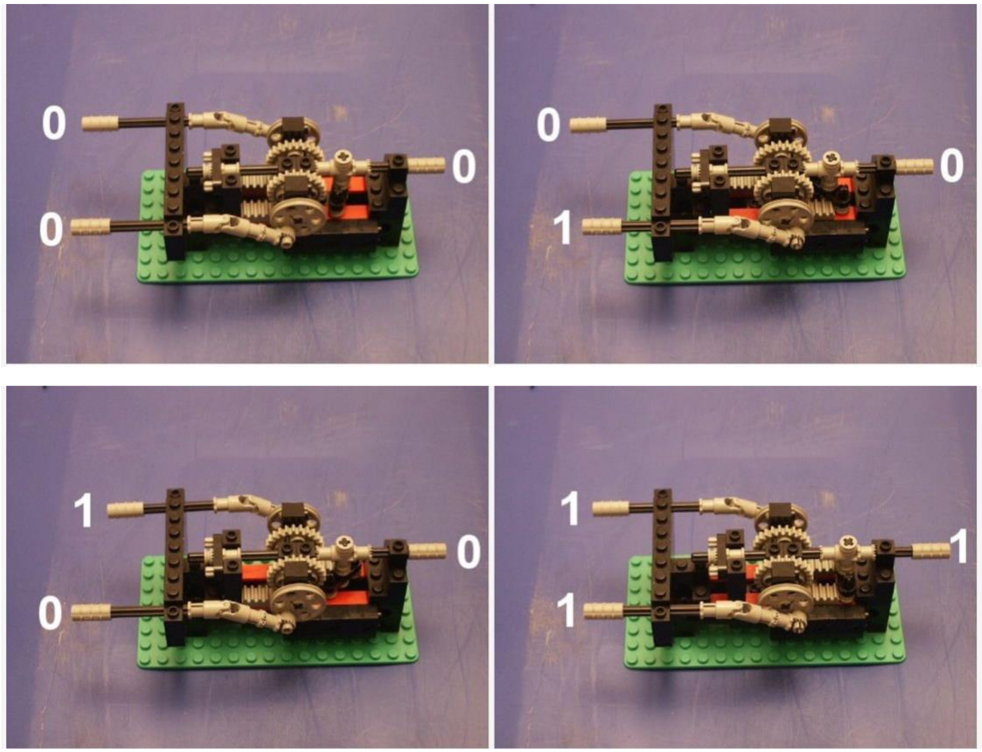


$Y = AB$

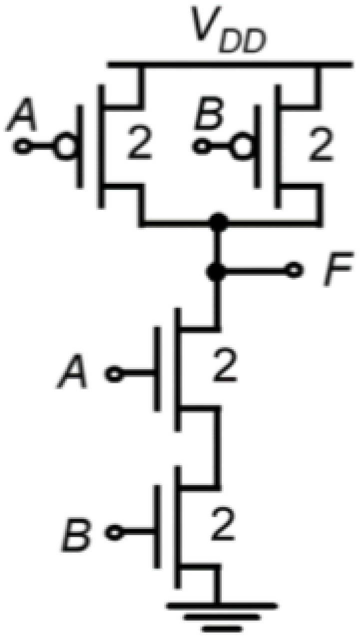
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



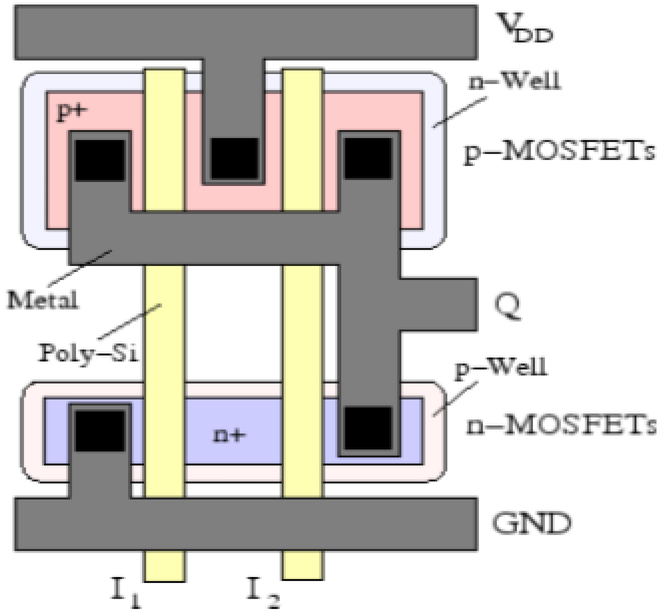
- Mechanical way



- Electrical way



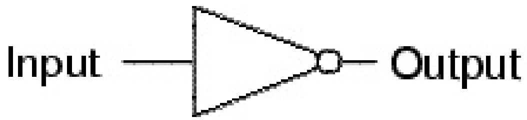
Schematic



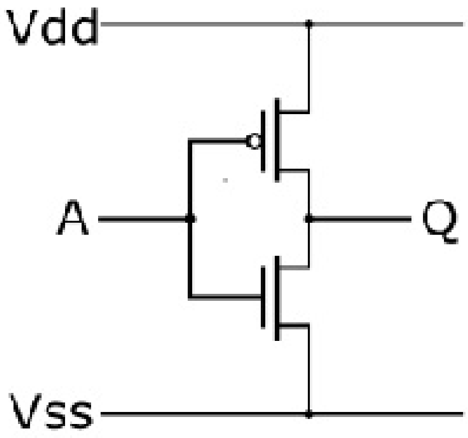
Physical layout

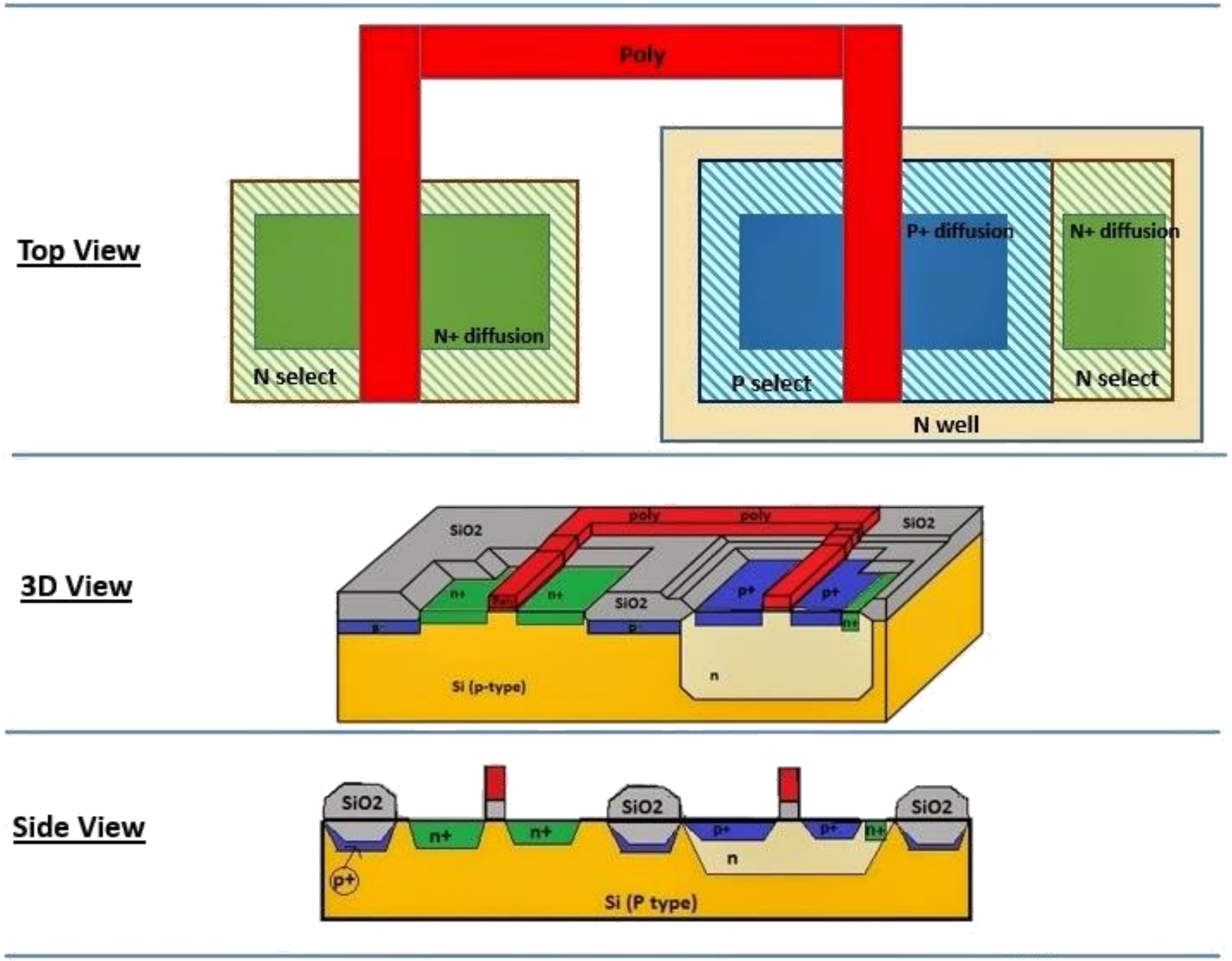
A 3D-view of the CMOS implementation

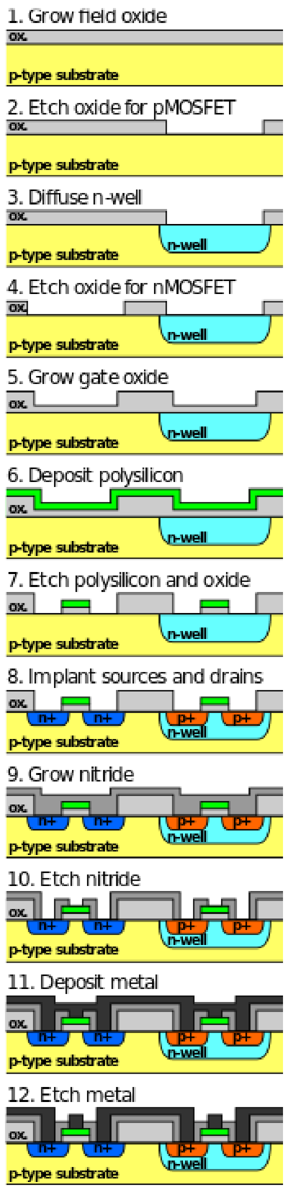
INVERTER



Input	Output
1	0
0	1

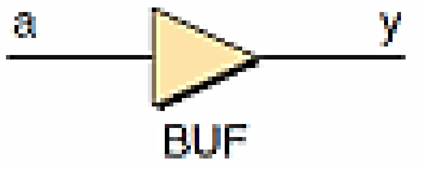






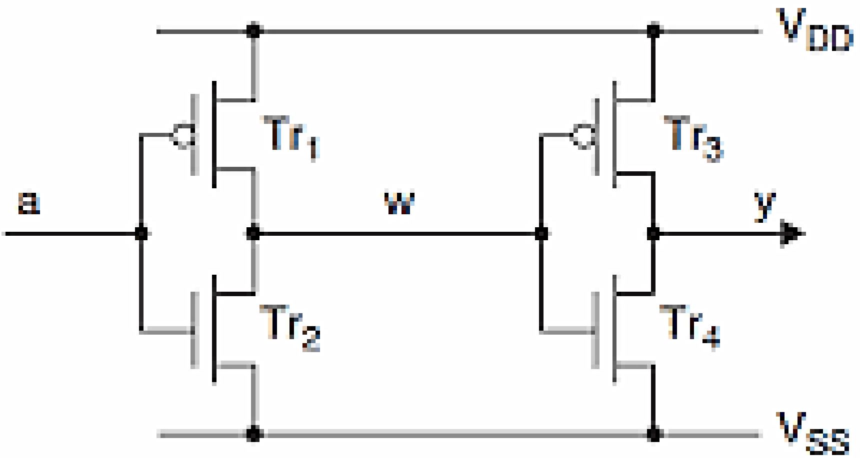
CMOS implementation of a buffer

- Why we need a buffer?
  - Logical (virtual) representation



a	y
0	0
1	1

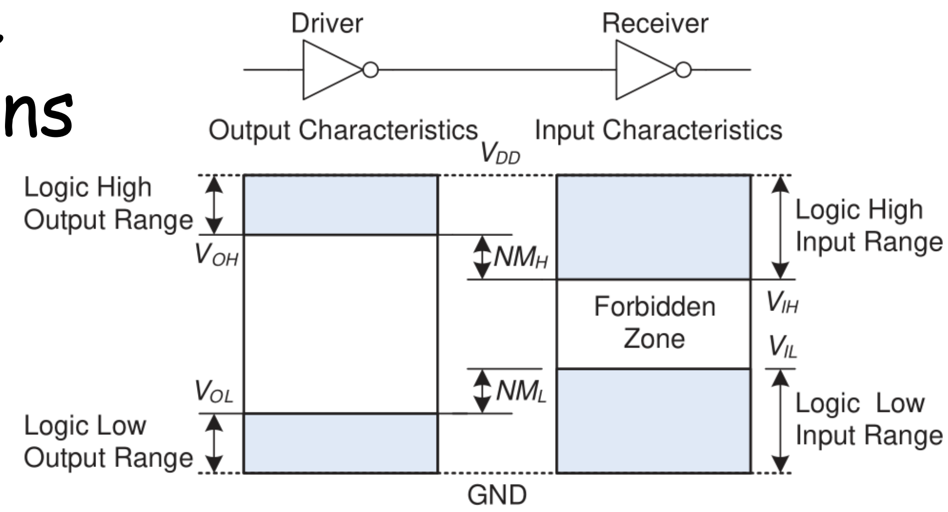
  - Physical implementation



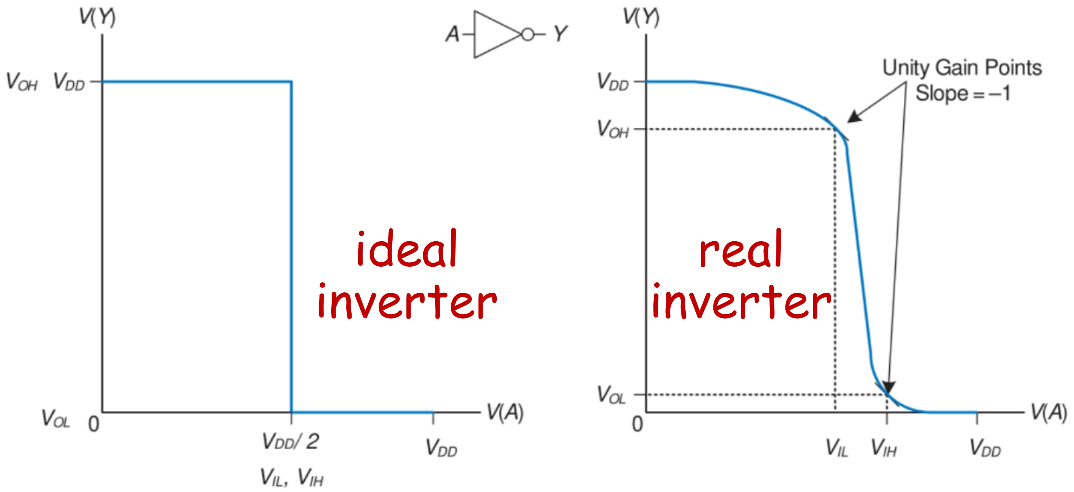
- From the logical point of view, a buffer might seem useless 逻辑上没有意义
- From the analog point of view, the buffer might have desirable characteristics such as the ability to deliver large amounts of current to drive a motor or many gates 物理上增强驱动能力
- This is an example of why we need to consider multiple levels of abstraction to fully understand a system; the digital abstraction hides the real purpose of a buffer.

The physical considerations

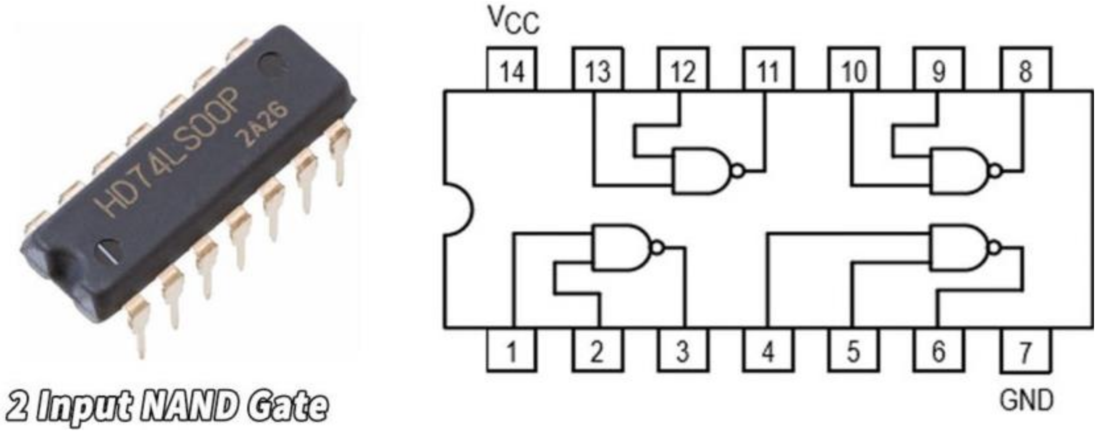
- Noise margins



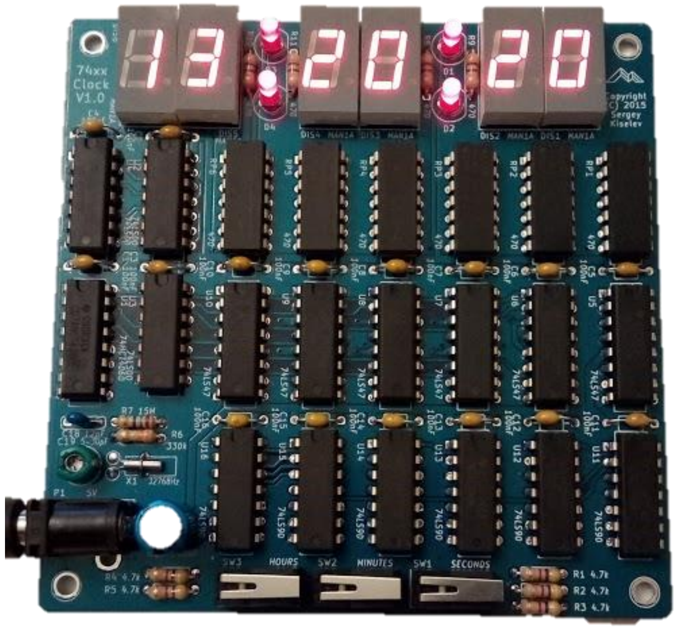
- DC transfer characteristics



- Physical chip



- A digital clock built with 74LSxx chips



Other logic gates

异或门

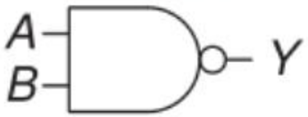
XOR

$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

与非门

NAND

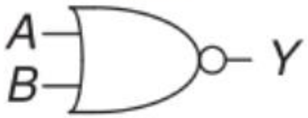


$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

或非门

NOR

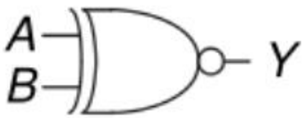


$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

同或门

XNOR

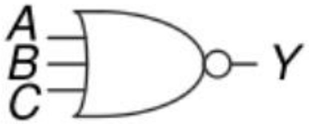


$Y = \overline{A \oplus B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

三输入或非门

NOR3

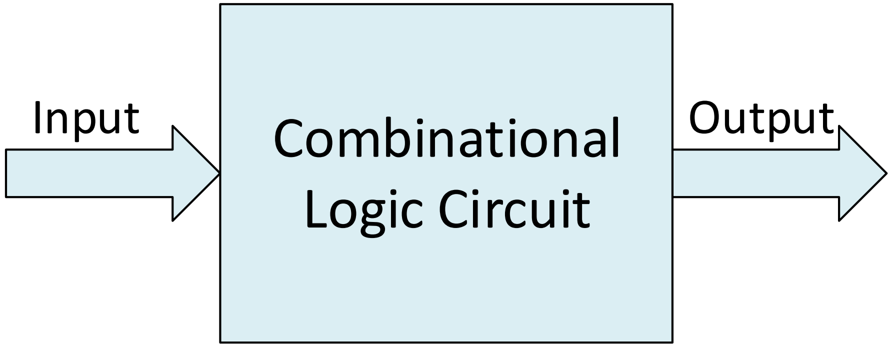


$Y = \overline{A + B + C}$

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

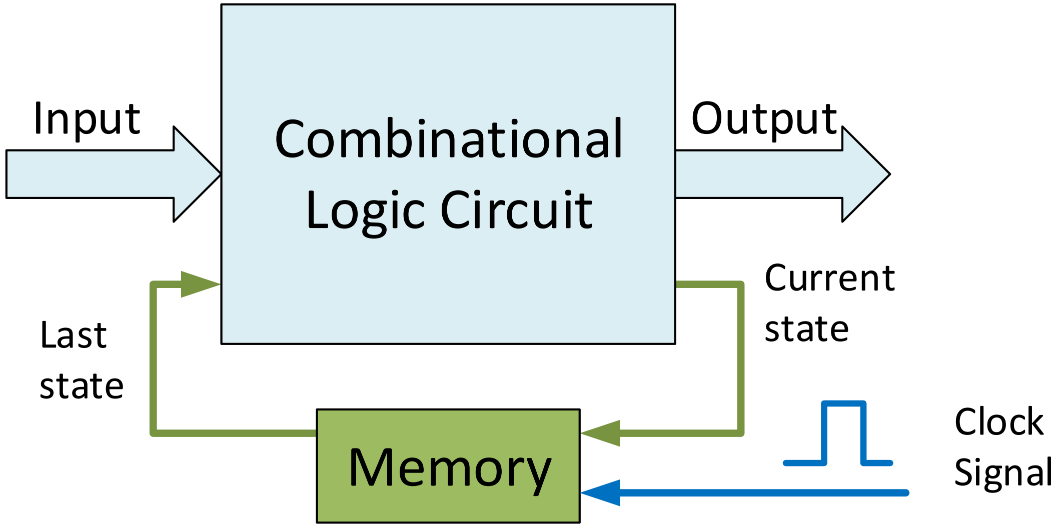
Digital electronics

- Combinational logic circuit



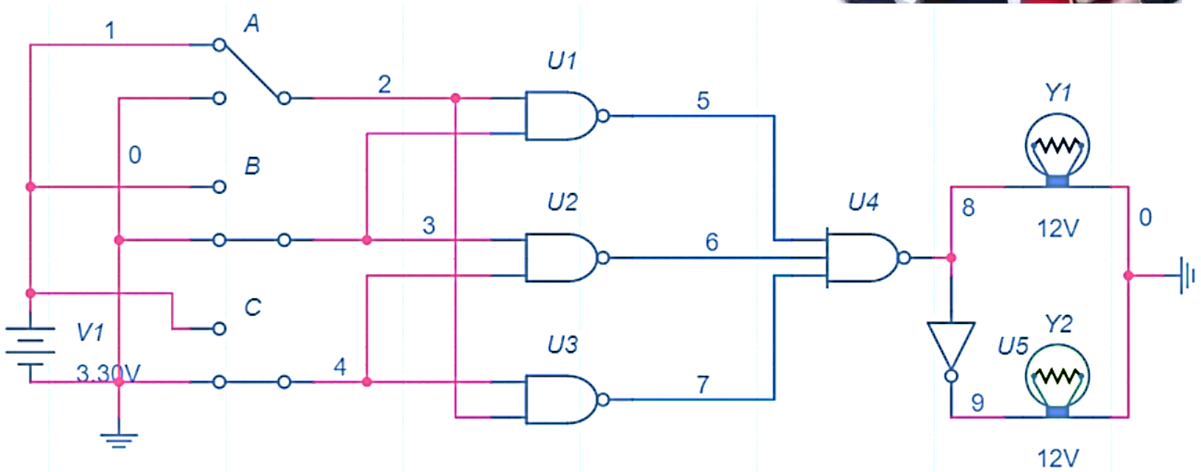
  - Output depends on
    - Present input values
  - No memory

- Sequential logic circuit



  - Output depends on
    - Present inputs
    - The history of past inputs
  - Have memory

Circuit implementation



- 3 input majority function
  - Truth table

S1	S2	S3	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

  - Logic expression

$$Y = AB + BC + AC$$

$$= \overline{\overline{AB}\;\overline{BC}\;\overline{AC}}$$

Boolean algebra

- Axioms 公理

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

- Theorems of one variable

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

- Theorems of two variables

	Theorem		Dual	Name
T6	$B \bullet C = C \bullet B$	T6′	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	T7′	$(B + C) + D = B + (C + D)$	Associativity
T8	$(B \bullet C) + (B \bullet D) = B \bullet (C + D)$	T8′	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Distributivity
T9	$B \bullet (B + C) = B$	T9′	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	T10′	$(B + C) \bullet (B + \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = B \bullet C + \overline{B} \bullet D$	T11′	$(B + C) \bullet (\overline{B} + D) \bullet (C + D) = (B + C) \bullet (\overline{B} + D)$	Consensus
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = (\overline{B_0} + \overline{B_1} + \overline{B_2} \ldots)$	T12′	$\overline{B_0 + B_1 + B_2 \ldots} = (\overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots)$	De Morgan's Theorem

Generalized procedures of combinational circuit design

- Improved equation minimization

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	
1	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	T3: Idempotency
2	$\overline{B}\,\overline{C}(\overline{A}+A)+A\overline{B}(\overline{C}+C)$	T8: Distributivity
3	$\overline{B}\,\overline{C}(1)+A\overline{B}(1)$	T5: Complements
4	$\overline{B}\,\overline{C}+A\overline{B}$	T1: Identity

- Sum of product 积(与)的和(或)
  (Multiple AND terms ORed together)

  1. $ABC + \overline{A}B\overline{C}$
  2. $AB + \overline{A}B\overline{C} + \overline{C}\overline{D} + D$
  3. $\overline{A}B + C\overline{D} + EF + GK + H\overline{L}$

- Product of sum 和(或)的积(与)
  (Multiple OR terms ANDed together)

  1. $(A + \overline{B} + C)(A + C)$
  2. $(A + \overline{B})(\overline{C} + D)F$
  3. $(A + C)(B + \overline{D})(\overline{B} + C)(A + \overline{D} + \overline{E})$

Generalized procedures of combinational circuit design

- Interpret the problem and set up its truth table
- Write the AND (product) term for each case where output = 1
- Combine the terms in SOP (sum of product) form
- Simplify the output expression if possible
- Implement the circuit for the final, simplified expression

- Equation minimization example

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$	
1	$\overline{B}\,\overline{C}(\overline{A} + A) + A\overline{B}C$	T8: Distributivity
2	$\overline{B}\,\overline{C}(1) + A\overline{B}C$	T5: Complements
3	$\overline{B}\,\overline{C} + A\overline{B}C$	T1: Identity

Karnaugh map (K-maps)

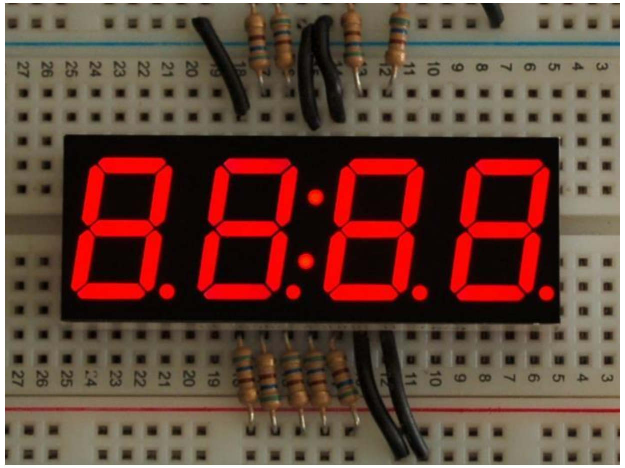
- A graphical method for simplifying Boolean equations
  - Invented in 1953 by Maurice Karnaugh at Bell Labs
  - – Adjacent squares share all the same literals **except one**
    相邻格的输入值仅有一位变化
  - The K-map also "wraps around."
    左右环接

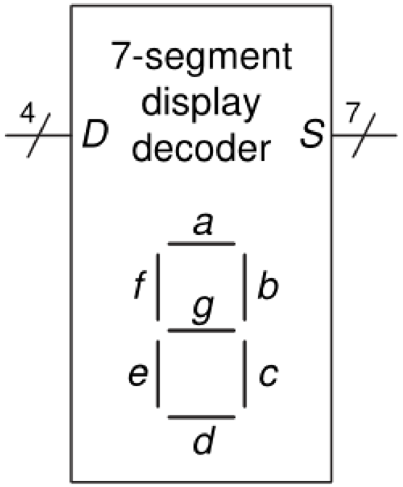
		BC			
		00	01	11	10
A	0	1	0	1	1
	1	1	0	0	1

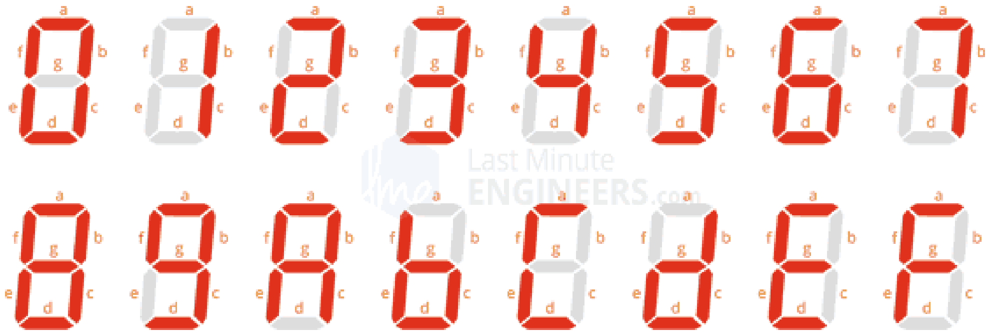
- Rules for finding a minimized equation:
  - Use the fewest circles necessary to cover all the 1's.
  - All the squares in each circle must contain 1's.
  - Each circle must span a rectangular block that is a power of 2 (i.e., 1, 2, or 4) squares in each direction.
  - Each circle should be as large as possible.
  - A circle may wrap around the edges of the K-map.
  - A 1 in a K-map may be circled multiple times if doing so allows fewer circles to be used.

Example: 7-segment decoder

- 7-segment display







(Pictures are from the Internet)

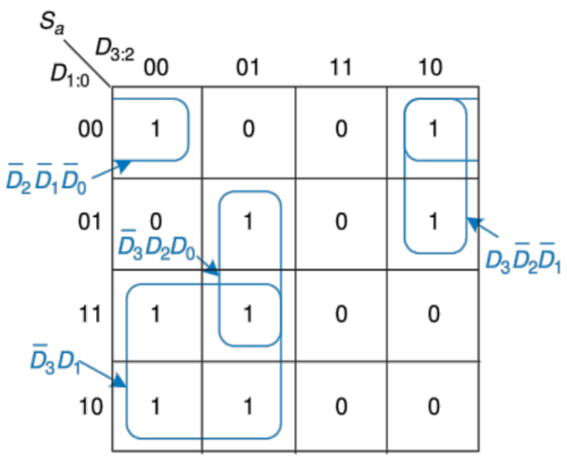
- Truth table

binary-coded decimal (BCD)

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

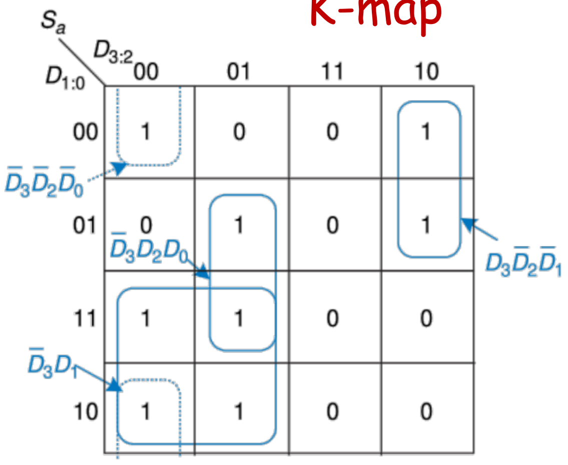
Example: 7-segment decoder

- K-map solution of Sa



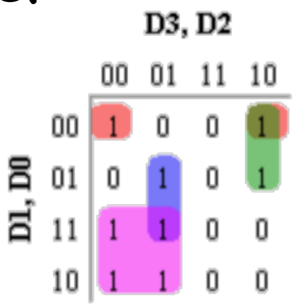
$S_a = \bar D_3 D_1 + \bar D_3 D_2 D_0 + D_3 \bar D_2 \bar D_1 + \bar D_2 \bar D_1 \bar D_0$

Alternative K-map

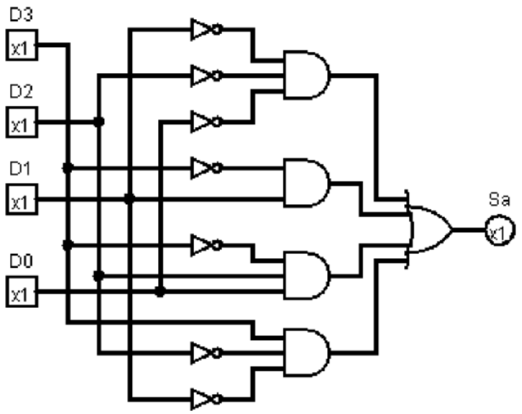


$S_a = \bar D_3 D_1 + \bar D_3 D_2 D_0 + D_3 \bar D_2 \bar D_1 + \bar D_3 \bar D_2 \bar D_0$

- Result & circuit



$\overline{D1}\,\overline{D0}\,\overline{D2} + \overline{D1}\,D3\,\overline{D2}$
$+ D0\,\overline{D3}\,D2 + D1\,\overline{D3}$



- Truth table

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0