Number System

1. Number System Conversion

- **$N = A_{n-1} \cdot r^{n-1} + A_{n-2} \cdot r^{n-2} + \ldots + A_1 \cdot r + A_0 \cdot r^0 + \ldots + A_{-m} \cdot r^{-m}$***

  n -> Most Significant Bit (MSB)

  m -> Least Sinificant Bit (LSB)

  r -> radix

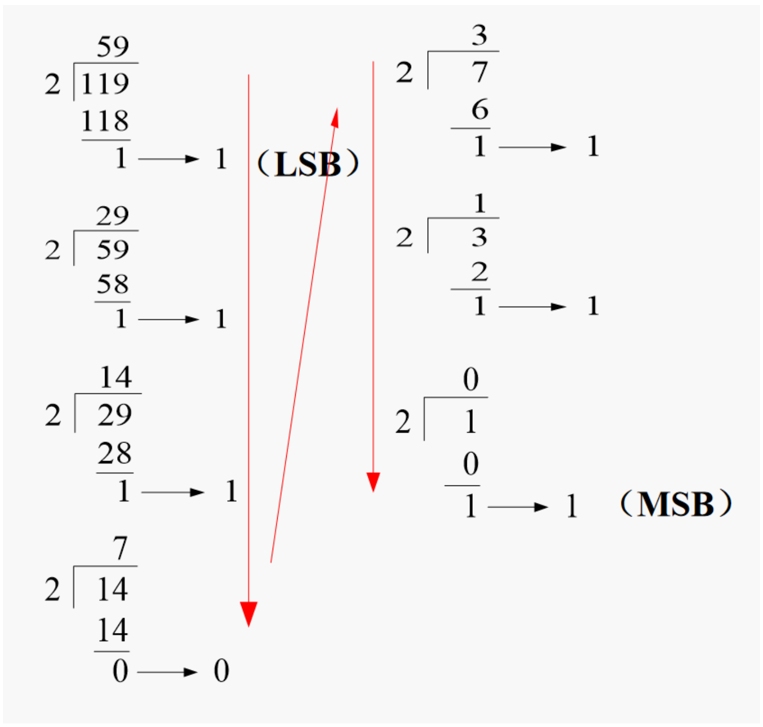
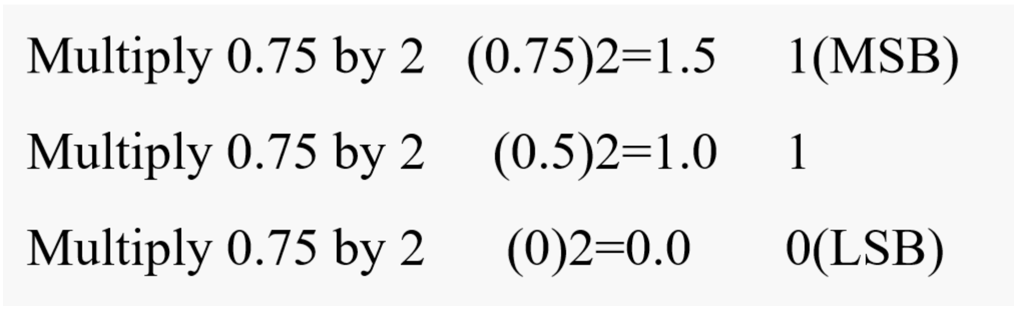
- Binary (r = 2)

  Octal (r = 8)

  Decimal (r = 10)

  Hexadecimal (r = 16)

- Decimal to Binary Conversion

> **E.g.** Convert 119(10) to binary
>
> 
>
>
> **Solution:** 1110111(2)
>
> **E.g.** Convert 0.75(10) to binary
>
> 
>
> | | | |
> |---|---|---|
> | Multiply 0.75 by 2 | (0.75)2=1.5 | 1(MSB) |
> | Multiply 0.75 by 2 | (0.5)2=1.0 | 1 |
> | Multiply 0.75 by 2 | (0)2=0.0 | 0(LSB) |
>
> **Solution** 0.110(2)

- Hexadecimal And Octal to Binary Conversion

> **E.g.** 3 A (16)= 0011 1010 (2)
>
> **E.g.** 2 7 (8) = 010 111 (2)

2. Binary Codes

- Signed Number Binary

Decimal	3	2	1	0	-1	-2	-3
Sign magnitude	0011	0010	0001	0000	1001	1010	1011
2s complement	0011	0010	0001	0000	1111	1110	1101
1s complement	0011	0010	0001	0000	1110	1101	1100

  MSB为符号位；

  原码->反码(1s):除符号位外全部取反；

  反码->补码(2s):反码+1

- Gray code

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3-bit	000	001	011	010	110	111	101	100								
4-bit	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000

  二进制码→格雷码（编码）：

  此方法从对应的n位二进制码字中直接得到n位格雷码码字，步骤如下：

  1. 对n位二进制的码字，从右到左，以0到n-1编号
  2. 如果二进制码字的第i位和i+1位相同，则对应的格雷码的第i位为0，否则为1（当i+1=n时，二进制码字的第n位被认为是0，即第n-1位不变）

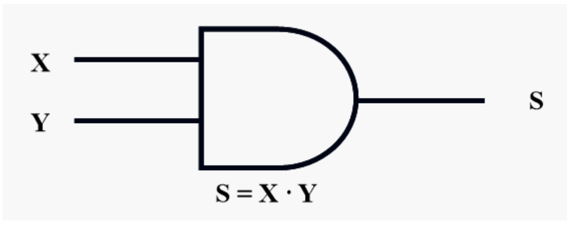
- Parity Method
  - The parity method is a method of error detection for simple transmission errors involving one bit (or an odd number of bits).
  - A parity bit is an “extra” bit attached to a group of bits to force the number of 1’s to be either even (even parity偶校验) or odd (odd parity奇校验).

    > Example: The ASCII character for “a” is 1100001 and for “A” is 1000001. What is the correct bit to append to make both of these have odd parity?
    >
    > The ASCII “a” has an odd number of bits that are equal to 1; therefore the parity bit is 0. The ASCII “A” has an even number of bits that are equal to 1; therefore the parity bit is 1.

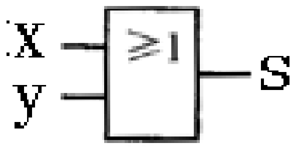
Boolean Switching Algebra

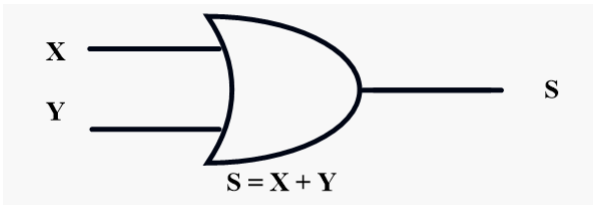
1. Logic Gates

- Tree Pirmitive Logic Gates: AND、OR、NOT
  - AND

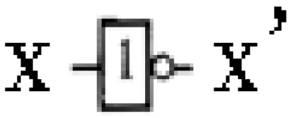
    

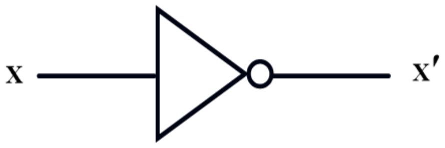
  - OR

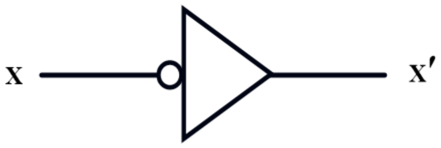
    

  - NOT

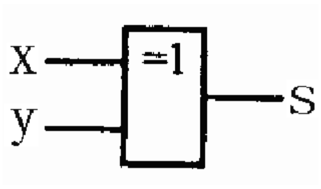
    

- Derived Logic Gates: NAND 、NOR、EX-OR、EX-NOR
  - XOR: s= x ⊕ y

    S = $\overline{X}$ **Y + X** $\overline{Y}$

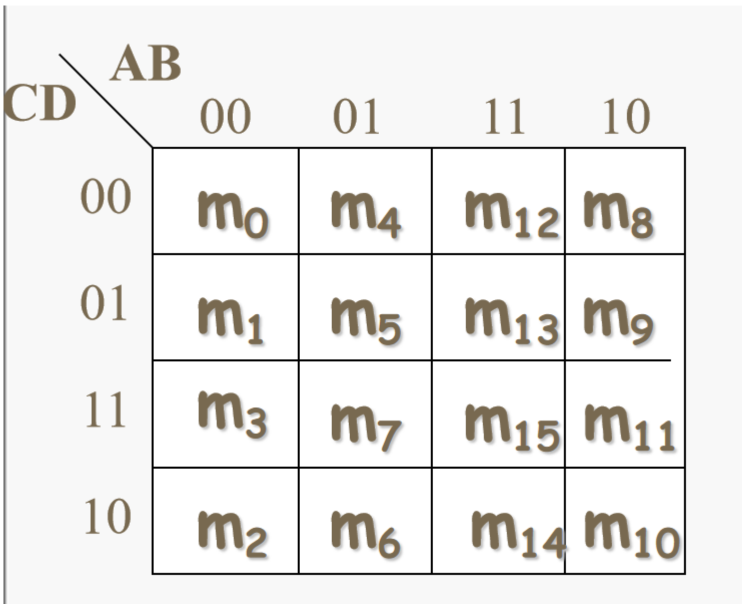
  - NXOR: s = (x ⊕ y)'

    S = X Y + $\overline{XY}$

2. Switching Algebra Theorems

- Idempotency Property (幂等性): $X+X=X \qquad X\cdot X=X$
- Absorption Property(吸收): $X+X\cdot Y=X \qquad X\cdot(X+Y)=X$
  $X+\overline{X}\cdot Y=X+Y \qquad X\cdot(\overline{X}+Y)=X\cdot Y$
- De Morgan Properties(反演，狄·摩根): $\overline{X+Y}=\overline{X}\cdot\overline{Y} \qquad \overline{X\cdot Y}=\overline{X}+\overline{Y}$
- Adjacency Properties: $X\cdot Y+X\cdot\overline{Y}=X \qquad (X+Y)\cdot(X+\overline{Y})=X$

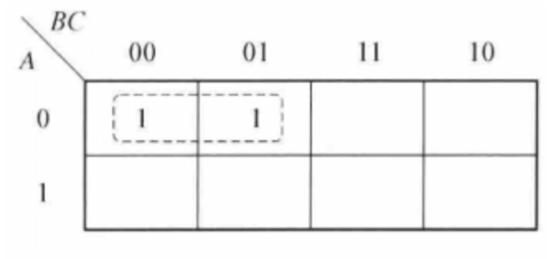
3. Karnaugh Maps



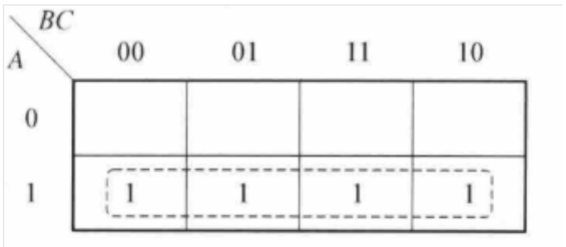
CD \ AB	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

- Simplification using K-map2、3、4 varibles

  卡诺图化简逻辑函数的基本原理，是依据关系式$A\cdot B+A\cdot\overline{B}=A$，即两个“与”项中，如果只有一个变量相反，其余变量均相同，则这两个“与”项可以合并成一项，消去其中互反的变量。
- 画卡诺圈所遵循的规则：
  - 必须包含所有的最小项；
  - 按照“从小到大”顺序，先圈孤立的“1”．再圈只能两个组合的，再圈四个组合的……
  - 圈的圈数要尽可能少(乘积项总数要少)；
  - 圈要尽可能大(乘积项中含的因子最少)。



(a) $F=\overline{A}\,\overline{B}\,\overline{C}+\overline{A}\,\overline{B}\,C=\overline{A}\,\overline{B}$



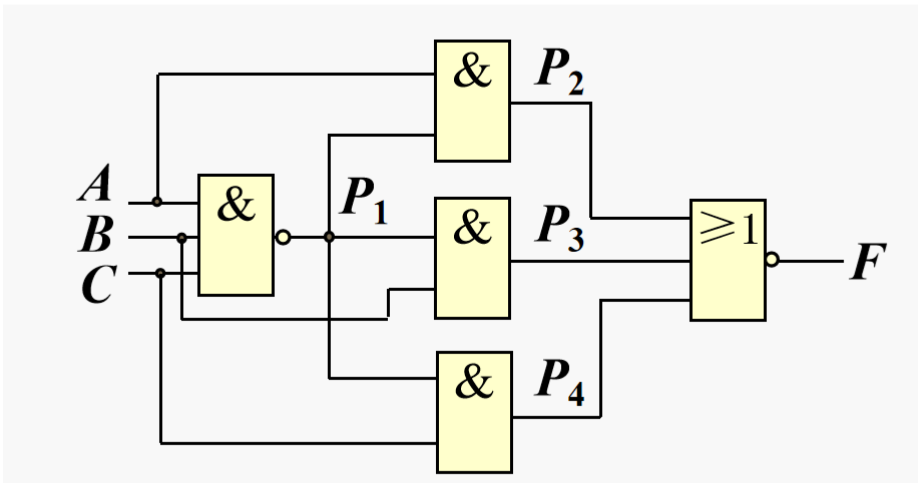
A \ BC	00	01	11	10
0				
1	1	1	1	1

(b) $F = A\overline{B}\,\overline{C} + A\overline{B}C + ABC + AB\overline{C} = A$

Combinational Circuit Design

1. Analysis of Combinational Logic

- Deriving Switching Equations
- Simplifying the switching equations
- Giving truth table
- Logic function conclussion



Swtiching Equations:

$P_1 = \overline{A \cdot B \cdot C}$

$P_2 = A \cdot P_1$

$P_3 = B \cdot P_1$

$P_4 = C \cdot P_1$

$F = \overline{P_2 + P_3 + P_4} = \overline{(A + B + C) \cdot \overline{A \cdot B \cdot C}}$

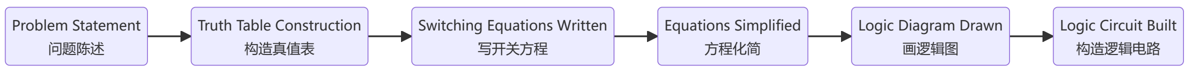
Equations Simplified:

$F = \overline{A} \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C$

Truth Table:

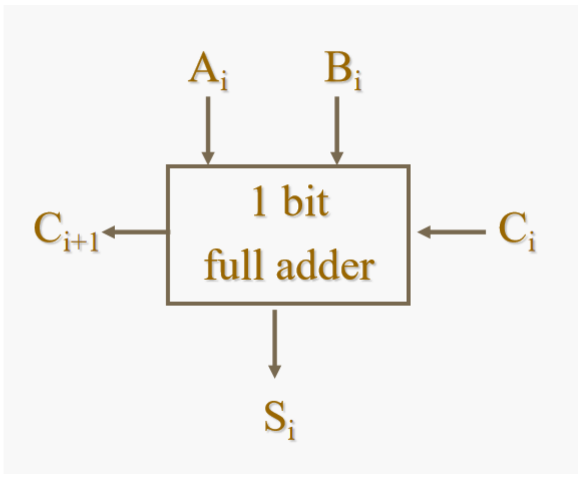
> | A | B | C | F |
> |---|---|---|---|
> | 0 | 0 | 0 | 1 |
> | ... | ... | ... | 0 |
> | 1 | 1 | 1 | 1 |
>
> **Logic Function:**
>
> agreement circuit

2.Combinational Circuit Design

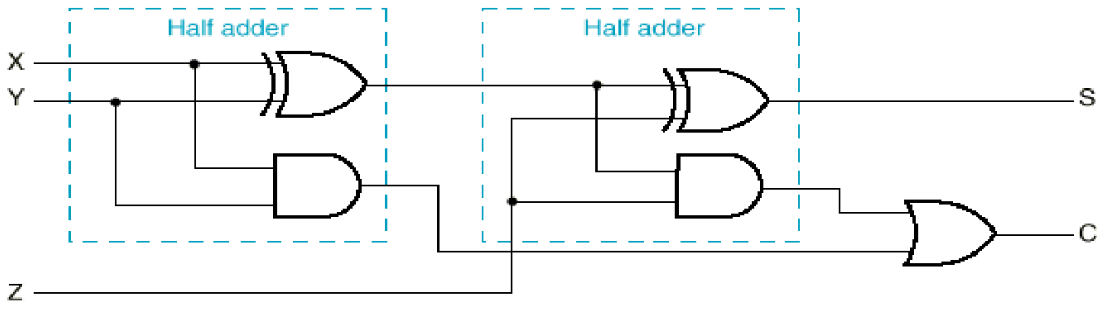

Problem Statement 问题陈述
Truth Table Construction 构造真值表
Switching Equations Written 写开关方程
Equations Simplified 方程化简
Logic Diagram Drawn 画逻辑图
Logic Circuit Built 构造逻辑电路

3. Full Adder

- Combinational circuit that performs the additions of 3 bits (two bits and a carry-in bit)

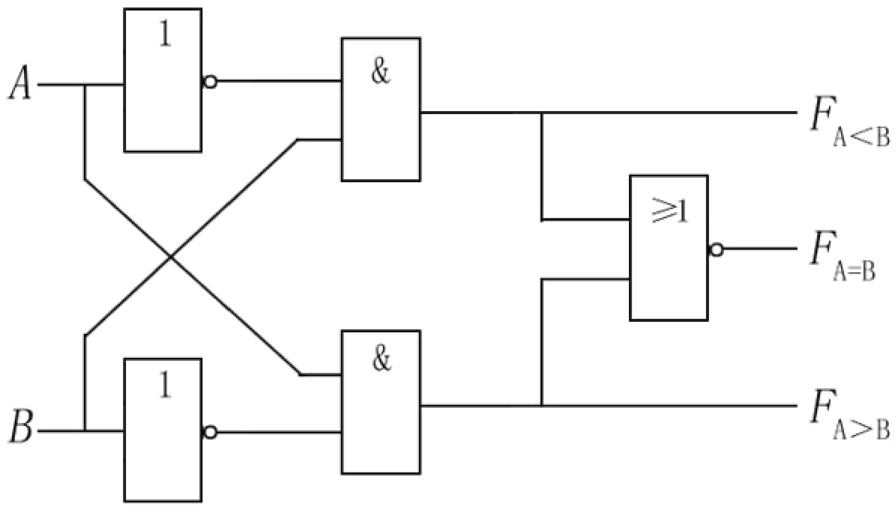


- A full adder can also be realized with two half adders and an OR gate

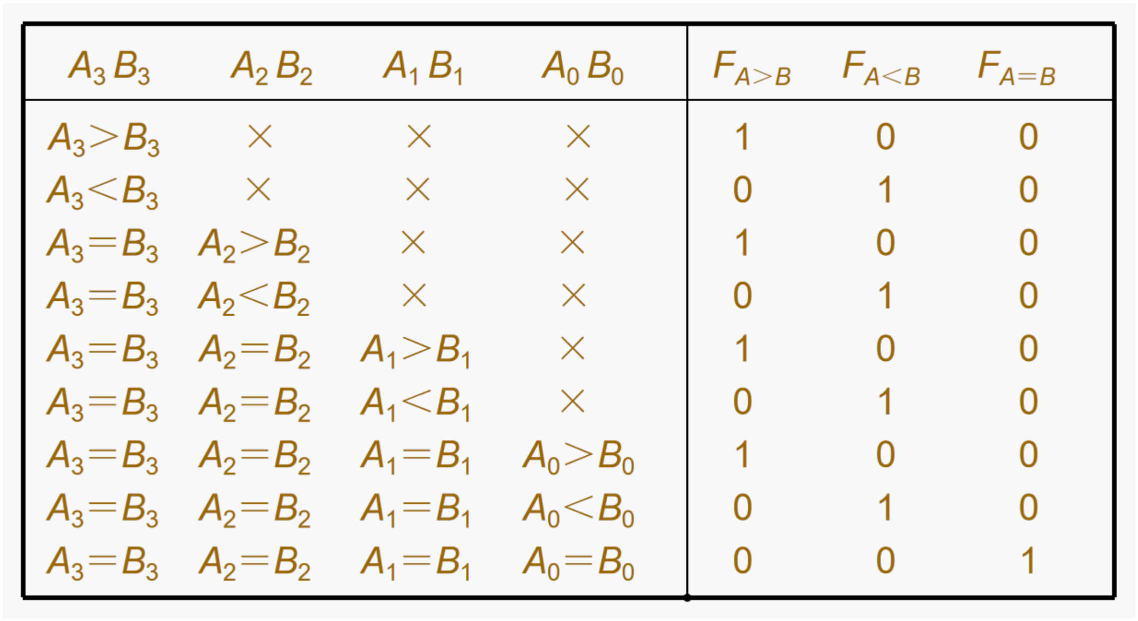


4. Binary Comparators

- Two-bit comparator

> **True Table:**
>
> | A | B | $F_{A>B}$ | $F_{A<B}$ | $F_{A=B}$ |
> |---|---|---|---|---|
> | 0 | 0 | 0 | 0 | 1 |
> | 0 | 1 | 0 | 1 | 0 |
> | 1 | 0 | 1 | 0 | 0 |
> | 1 | 1 | 0 | 0 | 1 |
>
> $$F_{A>B} = A \cdot \overline{B} = A \cdot \overline{AB}$$
> $$F_{A<B} = \overline{A} \cdot B = B \cdot \overline{AB}$$
> $$F_{A=B} = \overline{A} \cdot \overline{B} + A \cdot B = \overline{B \cdot \overline{AB} + A \cdot \overline{AB}}$$
>
> 
>

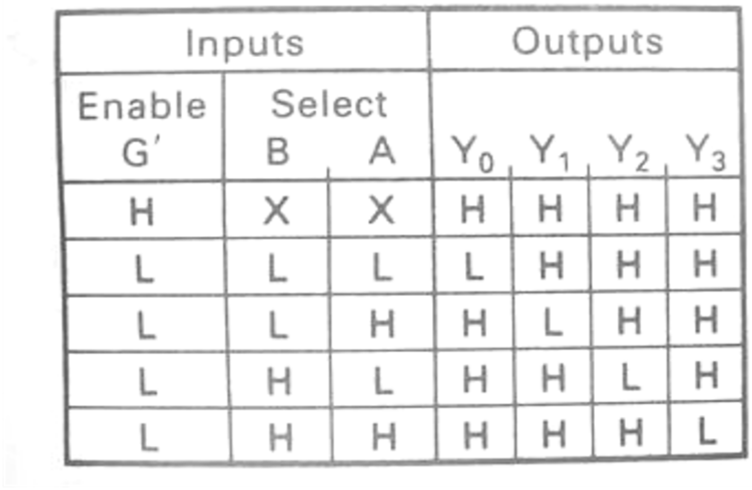
- Multiple-bit comparators



$A_3\,B_3$	$A_2\,B_2$	$A_1\,B_1$	$A_0\,B_0$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_3>B_3$	×	×	×	1	0	0
$A_3<B_3$	×	×	×	0	1	0
$A_3=B_3$	$A_2>B_2$	×	×	1	0	0
$A_3=B_3$	$A_2<B_2$	×	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	×	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	×	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	1	0	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	0	1	0
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	0	0	1

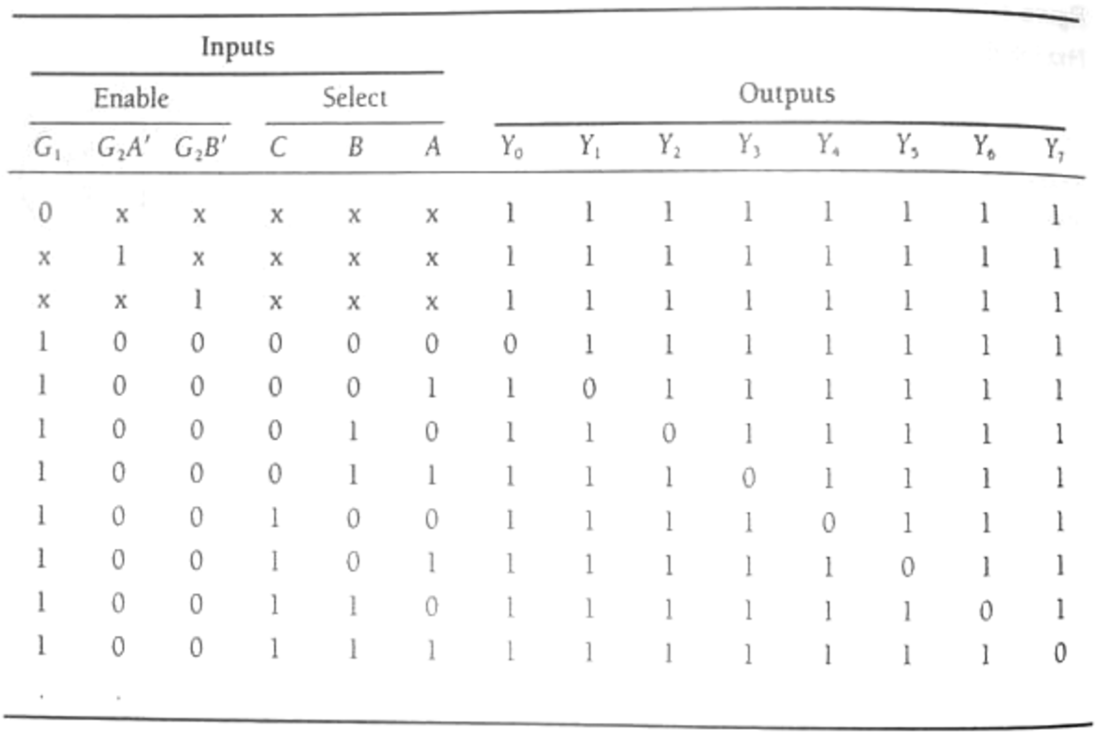
5. Decoders

- 2 to 4 Decoders



Inputs			Outputs			
Enable	Select					
G′	B	A	Y_0	Y_1	Y_2	Y_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

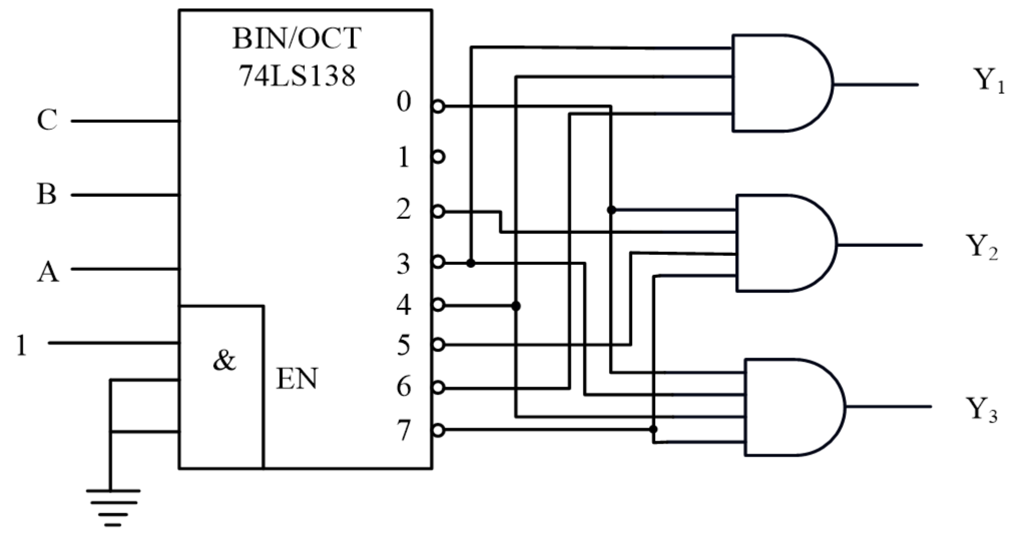
- 3 to 8 Decoders



Inputs						Outputs							
Enable			Select										
G_1	G_2A'	G_2B'	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

- Implementing Boolean functions using decoders

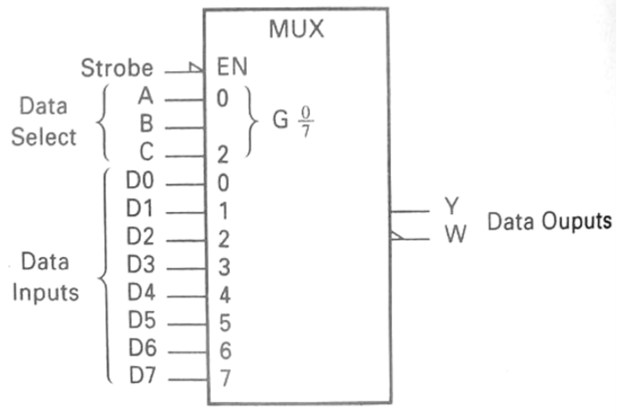
> Use the decoder (74LS138) to implement the function
>
> Y_1=A′B′+AC+A′C′
>
> Y_2=A′C+AC′
>
> Y_3=B′C+BC′
>
> Y1= A′B′+AC+A′C′ = m0+m1+m2+m5+m7 = ∑(0,1,2,5,7) = ∏(3,4,6) = m3′m4′m6′
>
> Y2= A′C+AC′ = m1+m3+m4+m6 = ∑(1,3,4,6) = ∏(0,2,5,7) = m0′m2′m5′m7′
>
> Y3= B′C+BC′ = m1+m2+m5+m6 = ∑(1,2,5,6) = ∏(0,3,4,7) = m0′m3′m4′m7′



6. Endcoders

- MUX -> Multiplexers
- 8 to 1 MUX

  Y = C’B’A’D_0 + C’B’AD_1 + C’BA’D_2 + C’BAD_3 + CB’A’D_4 + CB’AD_5+ CBA’D_6+ CBAD_7 = $\sum_{i=0}^{2^n-1} m_i D_i$

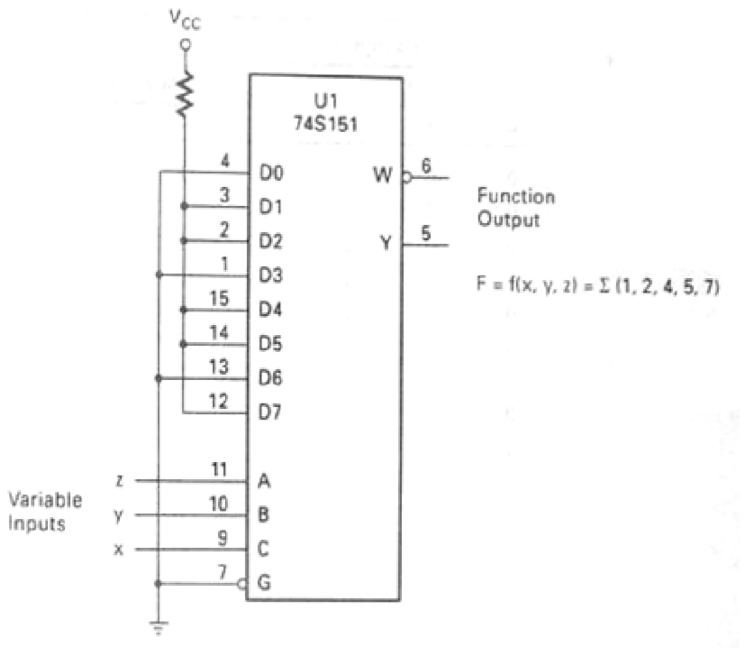


(b) IEEE logic symbol

- Implementing Boolean functions with Multiplexers

  Realize the Boolean function F=f(x,y,z)=∑(1,2,4,5,7)

  - **Using 8 to 1 MUX**

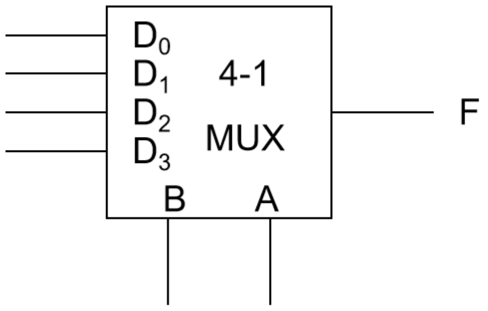


$D_0 = D_3 = D_6 = 0$

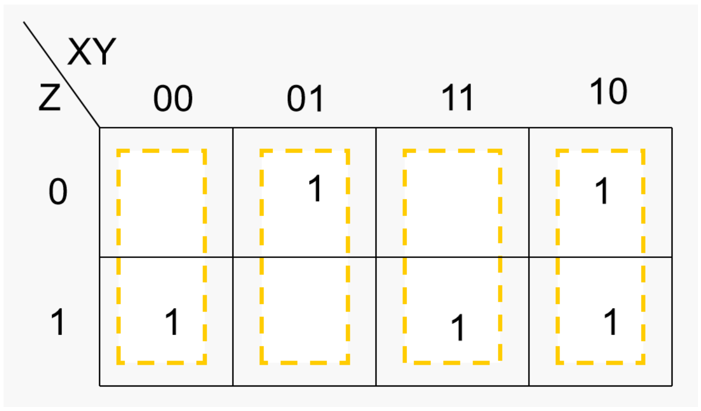
$D_1 = D_2 = D_4 = D_5 = D_7 = 1$

C = x, B = y, A = z

- **Using 4 to 1 MUX**



若A = Y, B = X, 则F = $X'Y'D_0 + X'YD_1 + XY'D_2 + XYD_3$

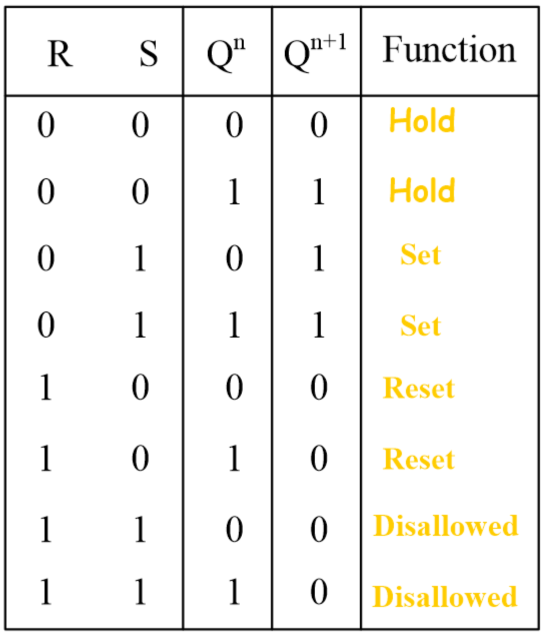


其中，根据卡诺图可得，$D_0 = Z$, $D_1 = Z'$, $D_2 = 1$, $D_3 = Z$

Sequential Logic

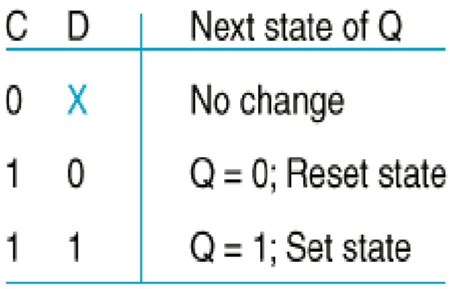
1. Flip-Flops(FF)

- Type of FF:
  - SR
  - D
  - JK
- Type of Triggering
  - Untriggered (asynchronous 异步)
  - Level-triggered (C = 1)
  - Edge-triggered (rising or falling edge of C)
- SR latch
  - Characteristic Equation: $Q^{n+1} = S + R' \cdot Q^n$ && $S{\cdot}R = 0$
  - True Table:



R	S	Q^n	Q^{n+1}	Function
0	0	0	0	Hold
0	0	1	1	Hold
0	1	0	1	Set
0	1	1	1	Set
1	0	0	0	Reset
1	0	1	0	Reset
1	1	0	0	Disallowed
1	1	1	0	Disallowed

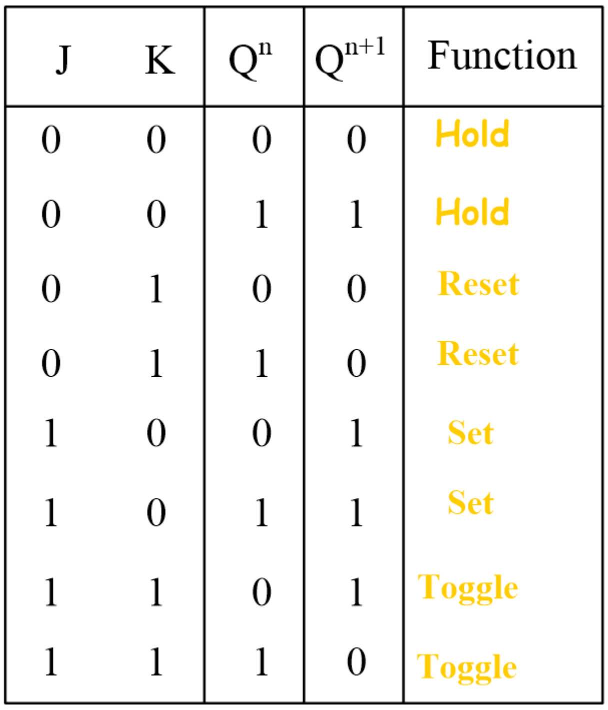
- D latch
  - Characteristic Equation: $Q^{n+1} = D$
  - True Table:



C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

- JK latch
  - Characteristic Equation: $Q^{n+1}=JQ^{n\prime}+K'Q^n$

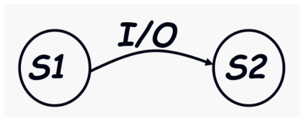
- True Table:



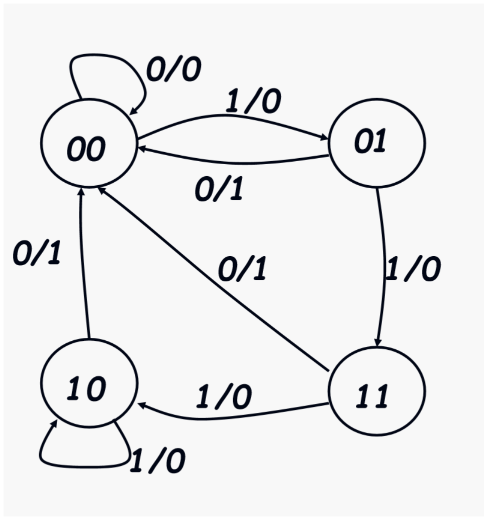
J	K	Q^n	Q^{n+1}	Function
0	0	0	0	Hold
0	0	1	1	Hold
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	Set
1	0	1	1	Set
1	1	0	1	Toggle
1	1	1	0	Toggle

2. State Diagram

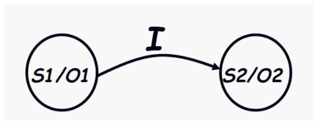
- Mealy model



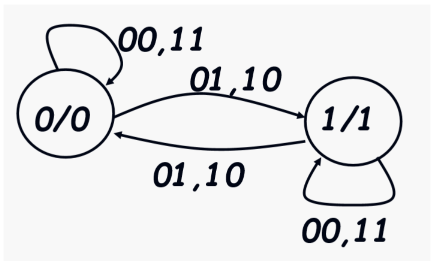
When at state s1 and apply input I, we get output O and proceed to state s2.



- Moore model

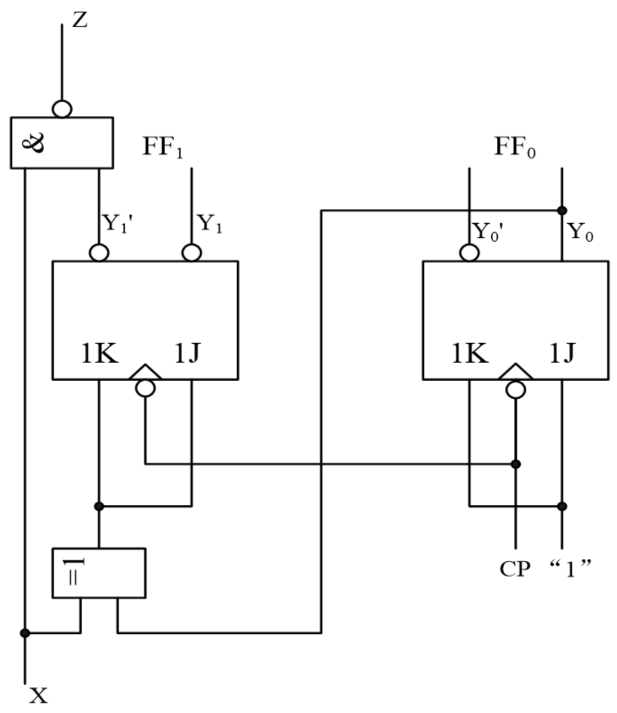


When at state s1 with outputO1 and apply input I, we proceed to state s2 with Output O2.



3. Synchronous Sequential Circuit Analysis

- 1.Determine the system variables: input, state, and output
- 2.Determine the flip-flop type. Write the characteristic equations
- 3.write the excitation equations
- 4.write the next state equations
- 5.Write the output variable equations
- 6.Construct a transition table
- 7.Assign symbols to the states and construct a table or state diagram
- 8.When possible, construct a timing diagram
- 9.Functionality analysis)



Analysis the following synchronous sequential circuit, suppose the present state is 00, the input sequence is 0000011111, give the timing diagram.

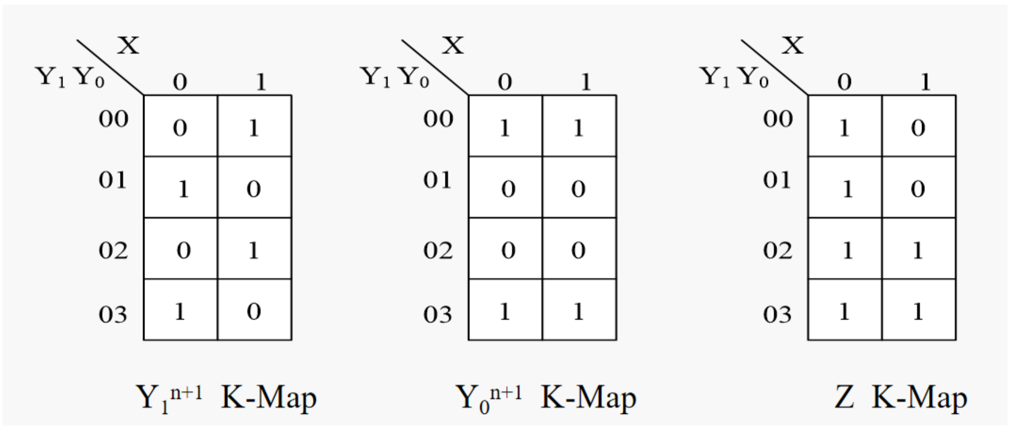
- input = X

  output = Z

  state variables = Y_1 and Y_2
- $Y^{n+1} = JY^{n'}+K'Y^n$

- $K_0 = J_0 = 1$

  $K_1 = J_1 = X \oplus Y_0$
- $Y_0^{n+1} = Y_0'$

  $Y_1^{n+1} = X'Y_1'Y_0 + X'Y_1Y_0' + XY_1'Y_0' + XY_1Y_0$
- $Z = (XY_1')' = X' + Y_1$
- 

Y_1Y_0 \ X	0	1
00	0	1
01	1	0
02	0	1
03	1	0

Y_1^{n+1} K-Map

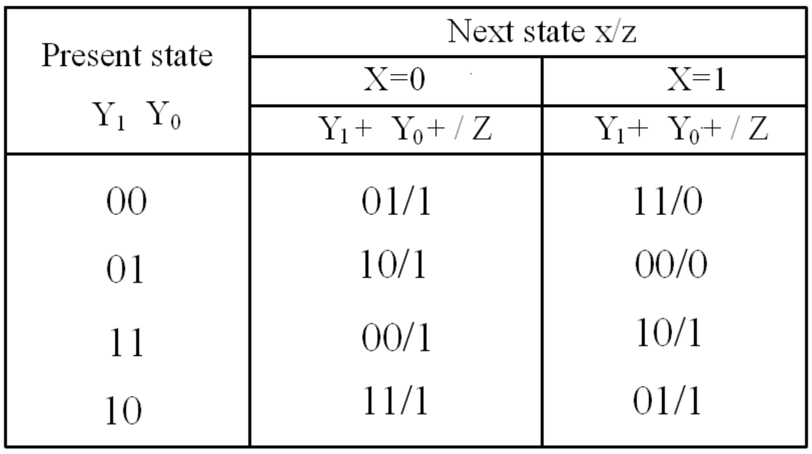
Y_1Y_0 \ X	0	1
00	1	1
01	0	0
02	0	0
03	1	1

Y_0^{n+1} K-Map

Y_1Y_0 \ X	0	1
00	1	0
01	1	0
02	1	1
03	1	1

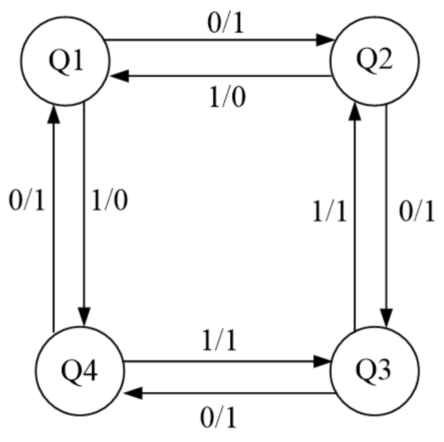
Z K-Map

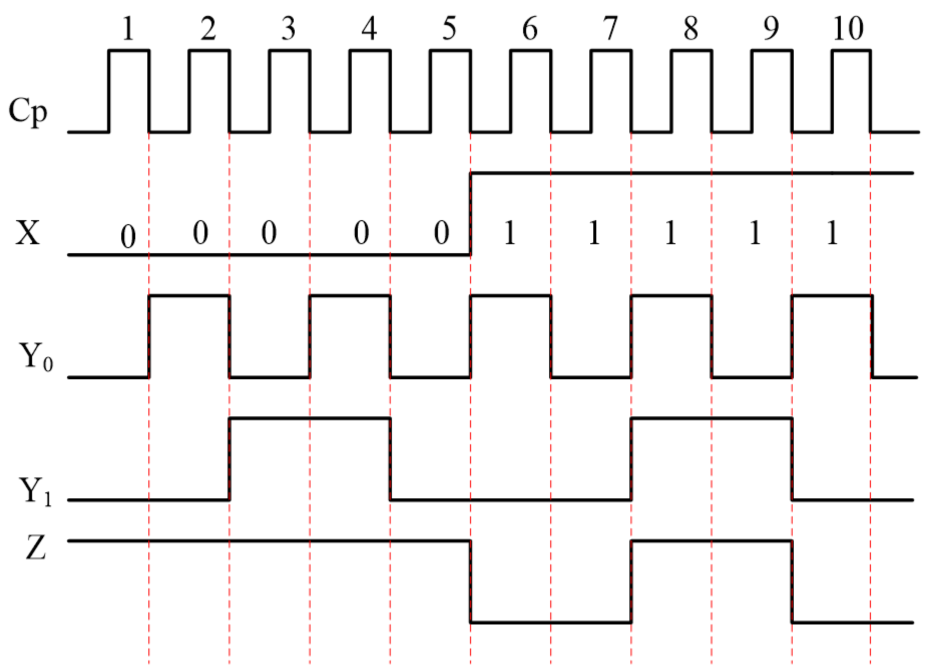
  Transition table:



Present state	Next state x/z	
	X=0	X=1
Y_1 Y_0	Y_1+ Y_0+ / Z	Y_1+ Y_0+ / Z
00	01/1	11/0
01	10/1	00/0
11	00/1	10/1
10	11/1	01/1

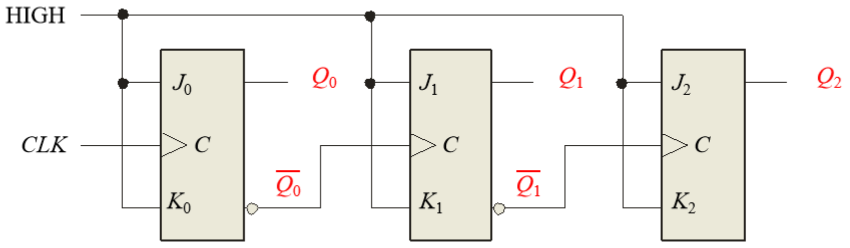
-
  - Q1 - 00
  - Q2 - 01
  - Q3 - 11
  - Q4 - 10

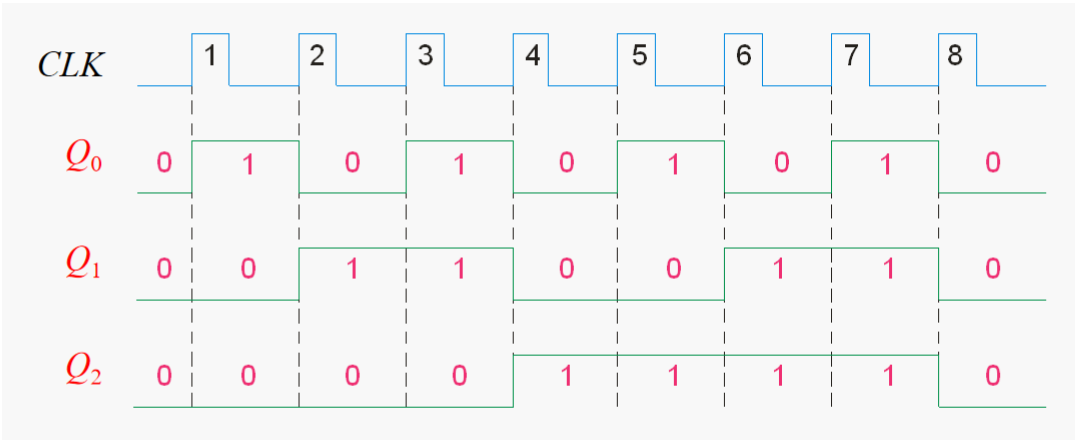




4. Asynchronous Counters

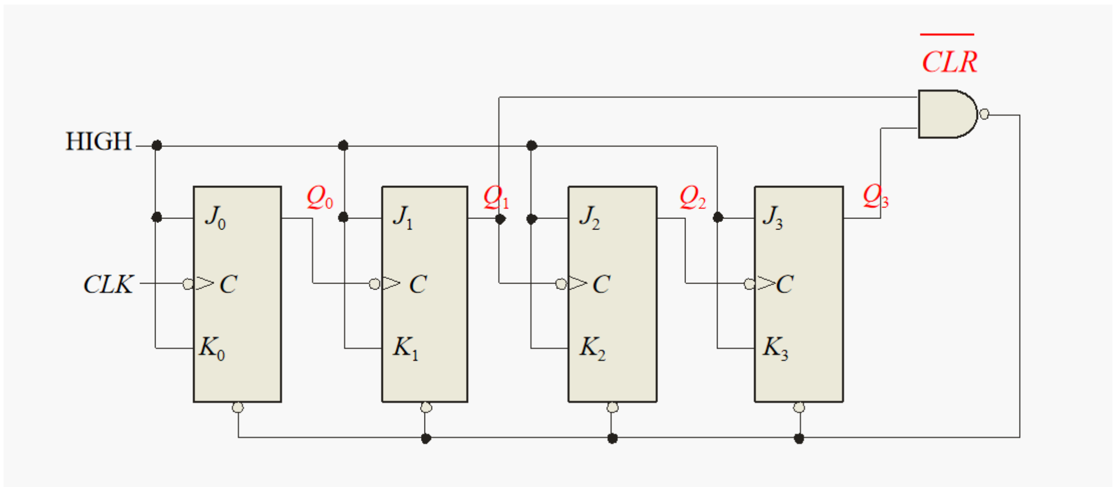
- Three bit Asynchronous Counter

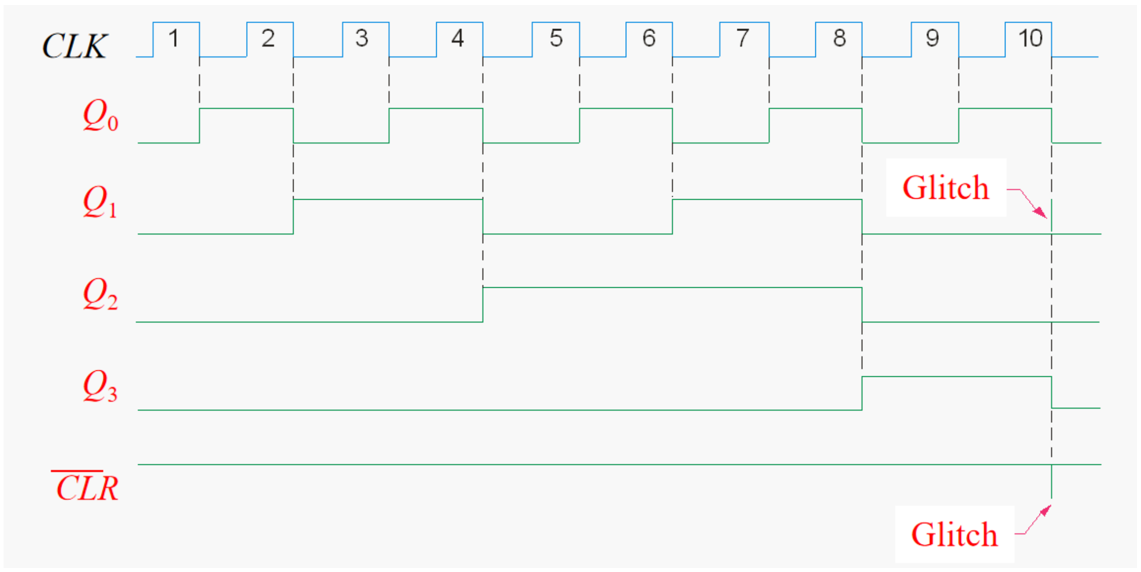




- Asynchronous Decade Counter(异步十进制计数器)

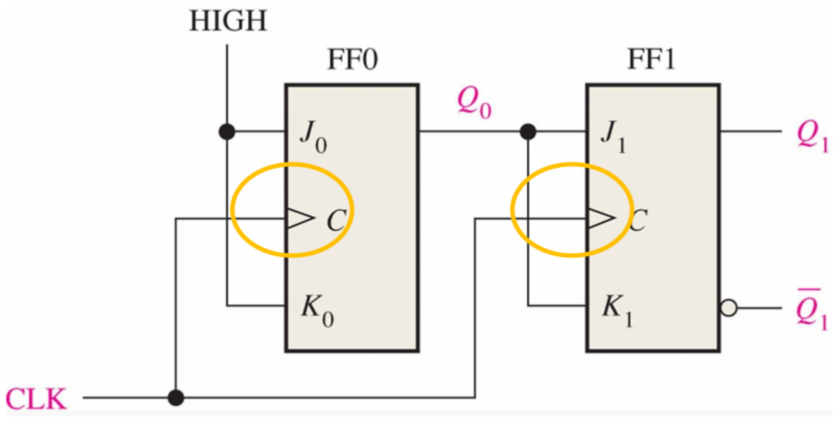
  This counter uses partial decoding to recycle the count sequence to zero after the 1001 state.



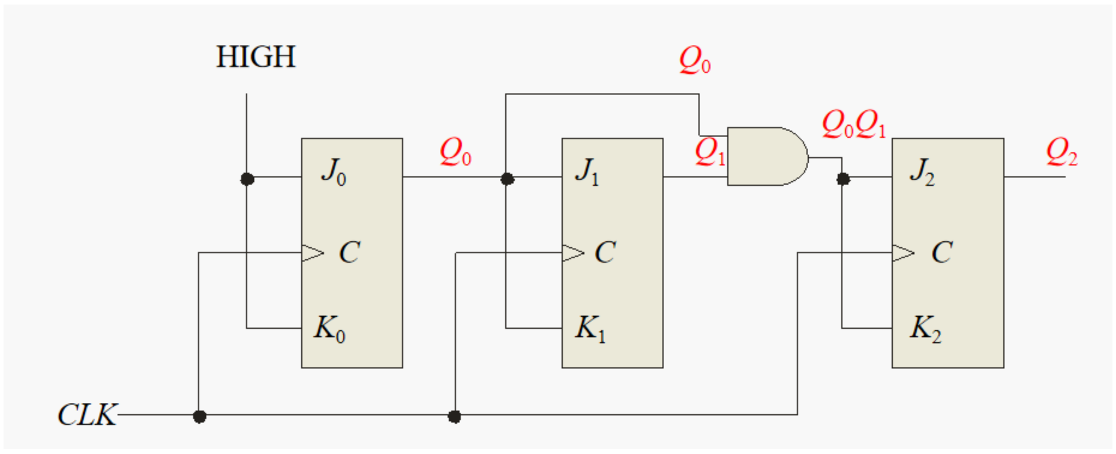


- Quick design a asynchronous module-K counter: Counting from 0 to K
    1. Choose the type of flip-flops.
    2. Decide the number of flip-flops: $2^{n-1} < K \le 2^n$
    3. Let all the flip-flops recycle back to 0 after the K-1 state.

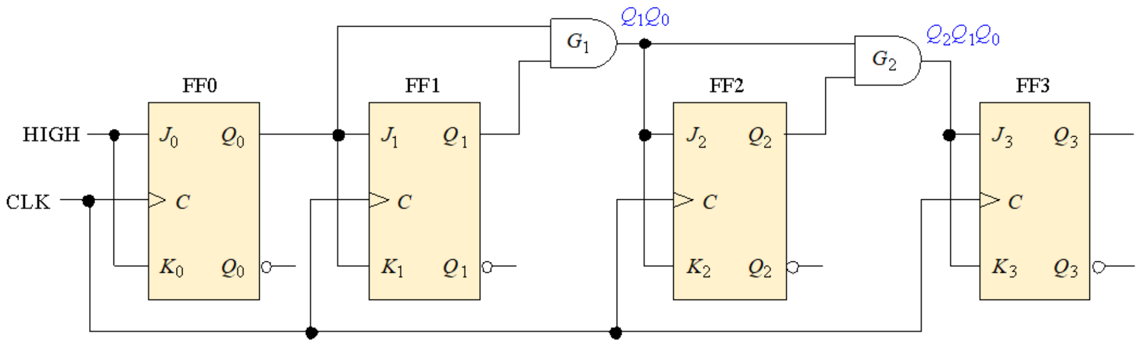
5. Synchronous Counters



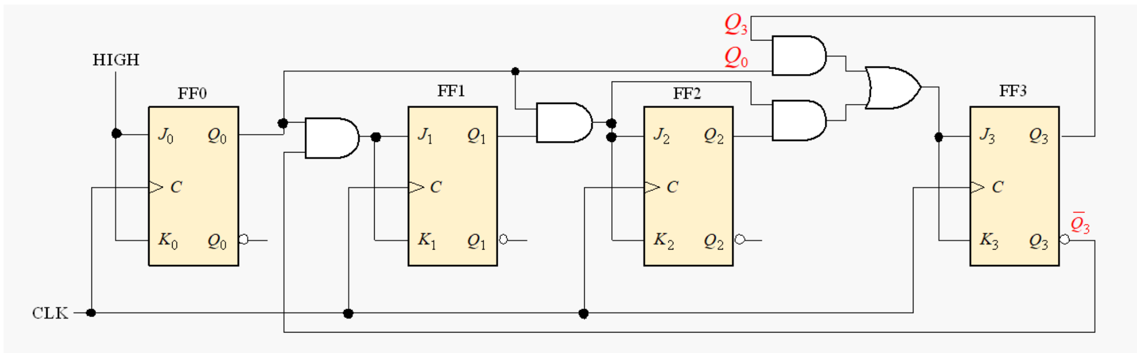
- A 3-bit binary Synchronous Counters(三位同步计数器)



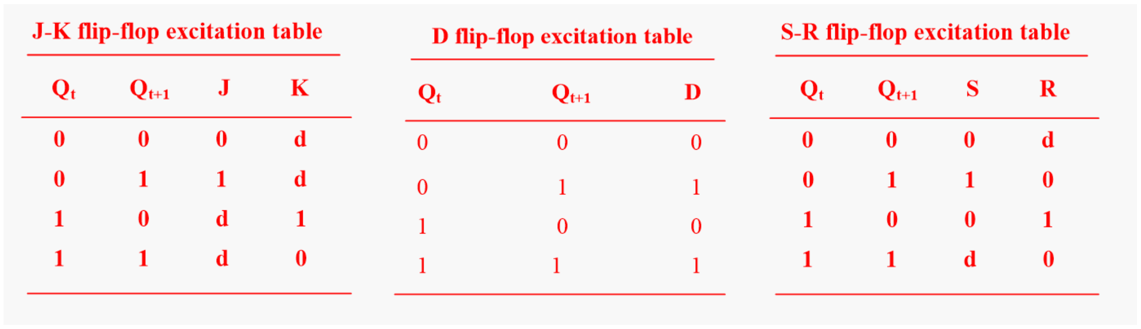
- A 4-bit Synchronous Binary Counter



- BCD Decade Counter



- Excitation Table and Equations



J-K flip-flop excitation table

Q_t	Q_{t+1}	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

D flip-flop excitation table

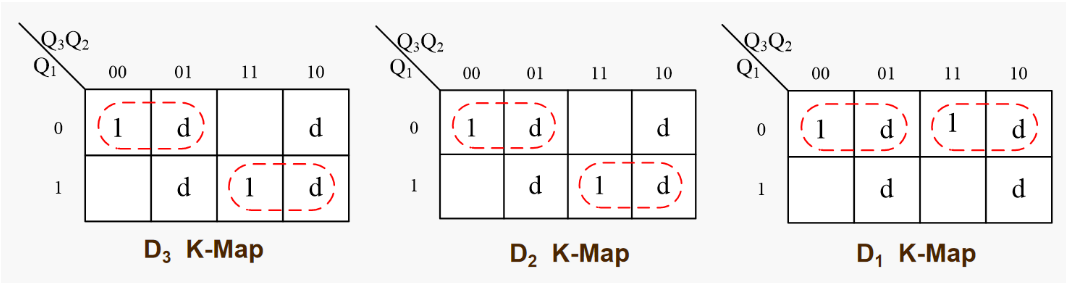
Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

S-R flip-flop excitation table

Q_t	Q_{t+1}	S	R
0	0	0	d
0	1	1	0
1	0	0	1
1	1	d	0

> Example1: Use D-FFs to draw the logic diagram for sequence generator (counter) for: 0 -> 7 -> 6 -> 1 -> 0 (000 -> 111 -> 110 -> 001 -> 000)

$Q_3Q_2Q_1$	$Q_3+Q_2+Q_1+$	$D_3D_2D_1$	Z
000	111	111	0
001	000	000	1
010	ddd	ddd	d
011	ddd	ddd	d
100	ddd	ddd	d
101	ddd	ddd	d
110	001	001	0
111	110	110	0

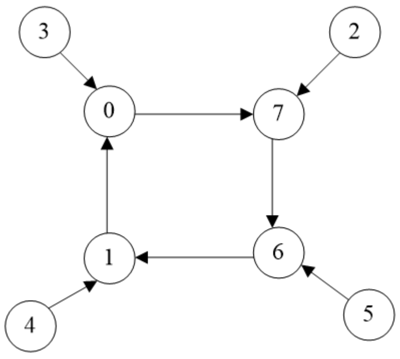


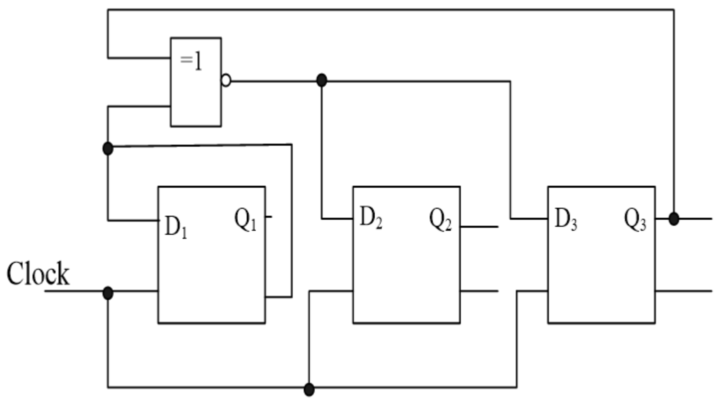
D_3 K-Map D_2 K-Map D_1 K-Map

$D_3 = Q_3' Q_1' + Q_3 Q_1$

$D_2 = Q_3' Q_1' + Q_3 Q_1$

$D_1 = Q_1'$



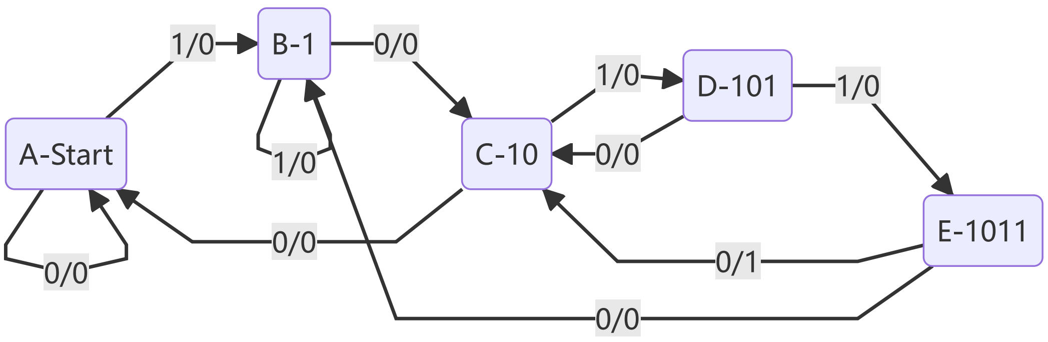


Example2:Construct a Mealy state diagram that will detect a serial input sequence of 10110. The detection of the required bit pattern can occur in a longer data string and the correct pattern can overlap with another pattern. When the input pattern has been detected, cause an output z to be asserted high.

For example, let the input string be

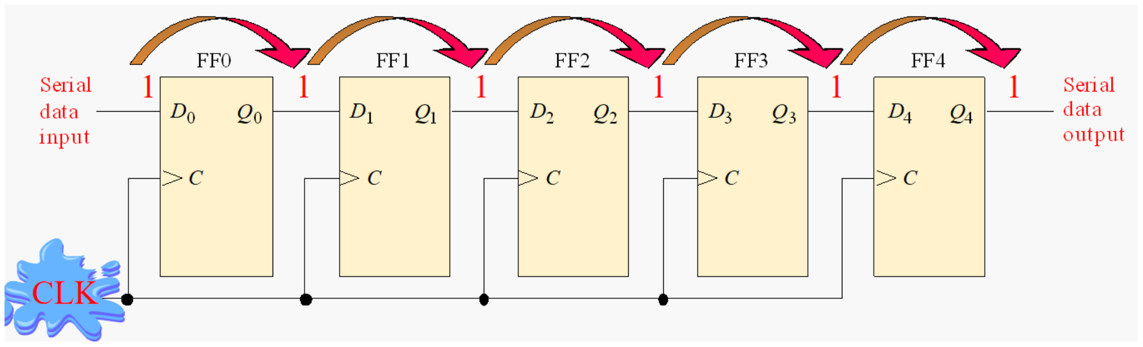
x = 10110110110

z = 00001001001

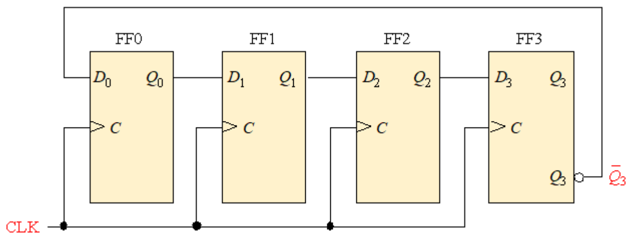


6. Registers

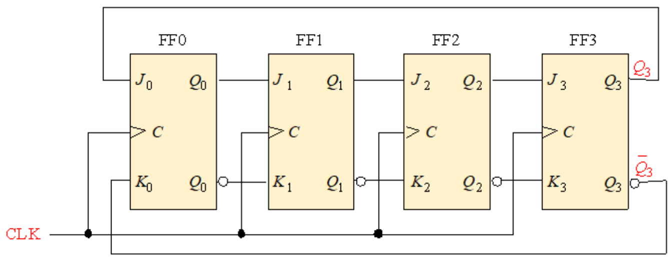
- Serial-in/Serial-out Shift Register



- Shift Registers Counters
  - Johnson Register



  - Ring Register


FF0
FF1
FF2
FF3
J_0 Q_0
J_1 Q_1
J_2 Q_2
J_3 Q_3
C
K_0 Q_0
K_1 Q_1
K_2 Q_2
K_3 Q_3
Q_3
$\bar{Q}_3$
CLK