4.1、有PROM，PAL，PLA和GAL

相同点：都以与阵列或或阵列为核心逻辑资源。

不同点：PROM：与阵列固定，或阵列可编程 PAL：或阵列固定，与阵列可编程

PLA：与、或阵列都可编程 GAL在PAL基础上，输出端集成个可编程的输出逻辑宏单元

4.2：(2)、BCD 2 4 2 1

0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
~~5~~	~~0100~~	~~1011~~
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110
9	1001	1111
10	1010	0000
11	1011	0001
12	1100	0010
13	1101	0011
14	1110	0100
15	1111	1011

(3)、Gray码 8421码

0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

$G_{10} = m_{10} + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$

$G_3 = m_5 + m_6 + m_7 + m_8 + m_9 + m_{15}$

$G_2 = m_4 + m_6 + m_7 + m_8 + m_9 + m_{14}$

$G_1 = m_2 + m_3 + m_5 + m_8 + m_9 + m_{12} + m_{13} + m_{15}$

$G_0 = m_1 + m_3 + m_5 + m_7 + m_9 + m_{11} + m_{13} + m_{15}$

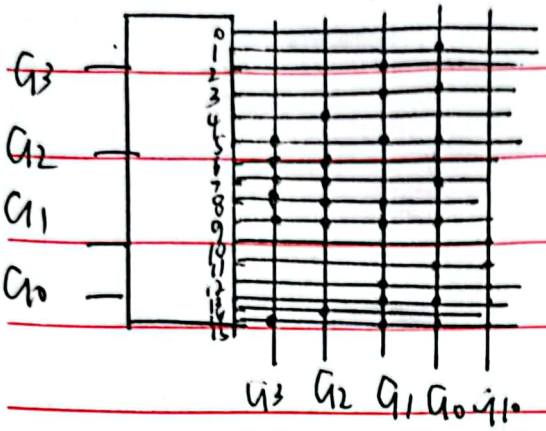
$G_{10} = m_8 + m_9 + m_{10} + m_{11} + m_{14} + m_{15}$

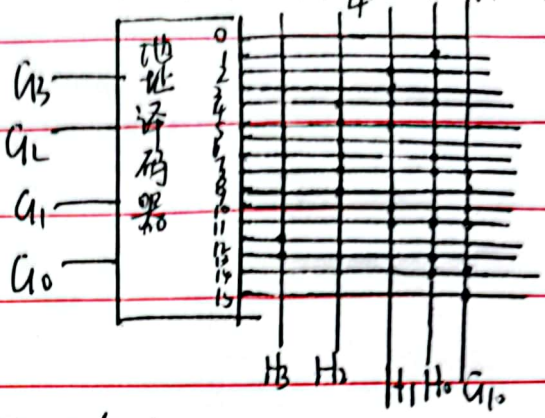
$H_3 = m_{12} + m_{13}$

$H_2 = m_4 + m_5 + m_6 + m_7 + m_8 + m_9$

$H_1 = m_2 + m_3 + m_4 + m_5 + m_{10} + m_{11}$

$H_0 = m_1 + m_2 + m_6 + m_7 + m_8 + m_9 + m_{12} + m_{13} + m_{14}$





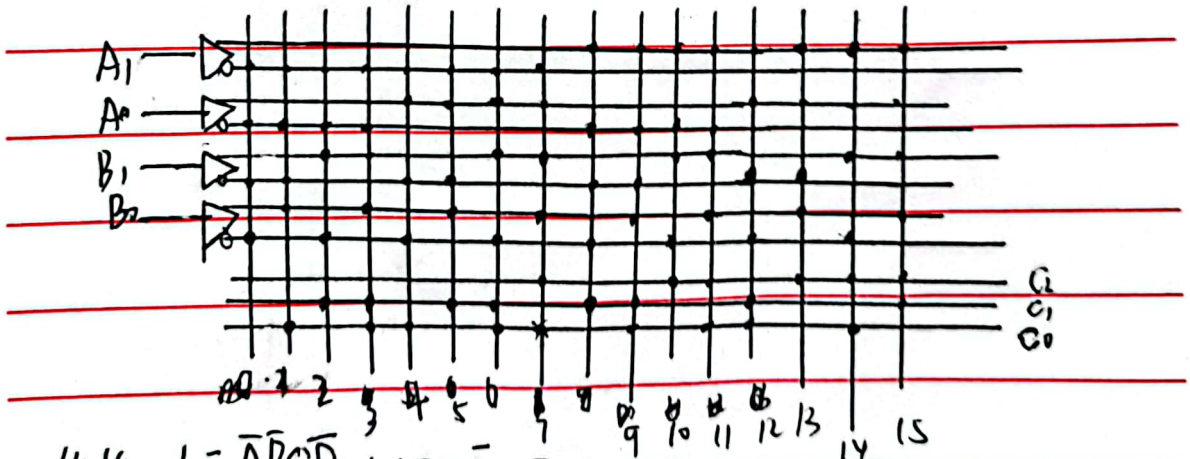
4.3、已知输入两位 A_1A_0，输出 B_1B_0，输出 $C_2C_1C_0$

A_1A_0	B_1B_0	$C_2C_1C_0$
00	00	000
00	01	001
...	...	...

可得：$C_2 = \sum m(7,10,11,13,14,15)$

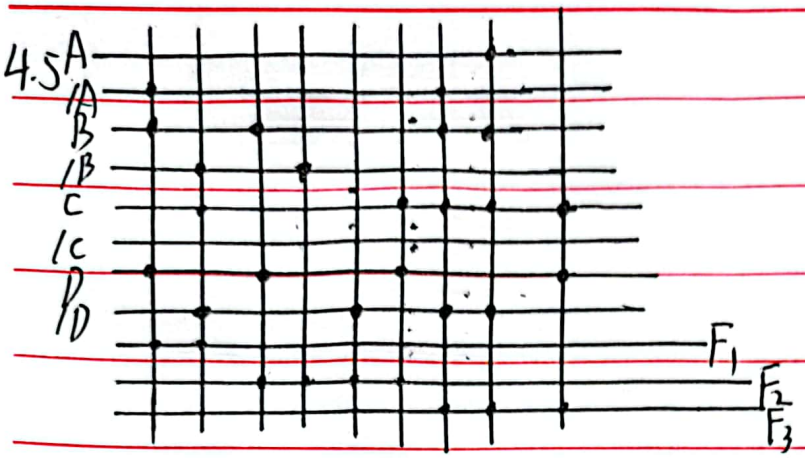
$C_1 = \sum m(2,3,5,6,8,9,12,15)$

$C_0 = \sum m(1,3,4,6,9,11,12,14)$



4.4、$L = \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D} + \bar{B}CD + \bar{A}BCD + A\bar{B}\bar{C}\bar{D}$

4.5

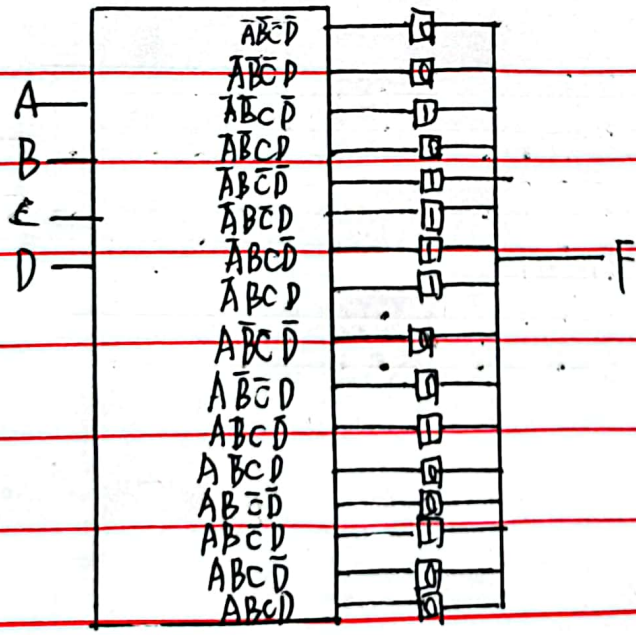


4.8. 每一级寄存器的输入与上一级的反馈相连，此外最高一级与最低一级相连。上一级输出0时，反馈/Q=1(Q=1)，此时下个时钟信号会使这个寄存器的存储值变为1，输出值为0，即传送到高位；上一级输出1时，反馈为/Q=0(Q=0)下个时钟信号使这个寄存器值变为0，输出为1，体现了循环计数的功能

4.10. 工艺：采用ECMOS工艺，使其具有可测试性，低功耗，高集成性，高速率和可编程性。

结构：通用结构，为设计提供了最大的灵活性，器件可加密

4.11 4.12



4.13. MAX7000是系统可编程的CPLD器件，内部是结构是基于乘积项，也就是与或阵列结构，采用E²PROM工艺，采用多电压的I/O接口，其I/O既能够和5V的器件兼容，Max 7000是高成本CPLD，规模较大。

4.15. 可编程逻辑块CLB，可内部配置有存储器SRAM阵列，可编程输入输出块，可编程内部连线PI

4.17.
```
Library IEEE;
    use IEEE.std_logic_1164.All
Entity sop IS
port(X:out std_logic;A,B,C,D,E,F:IN std_logic);
End ENTITY SOP;
Architecture fun OF sop IS
Begin
    X<=(A AND B) OR(C AND D) OR (E AND F);
End architecture fun;
```

4.13.
```
Library IEEE
    Entity Boo Is
port(F:OUT std_logic;A,B,C:IN std_logic);
END ENTITY Boo;
Architecture fun OF Boo IS
Begin
F<=(A OR (NOT B) OR C) AND (A OR B OR (NOT C)) AND (Not(C AND B AND C)
End Architecture Fun;
```

4.19.
```
Library IEEE;
    use IEEE,...;
Port(A,B,C,G1,G2,G3:IN std_logic;Y0,Y1...Y7:OUT std_logic)
```

```
END ENTITY GTE;
ARCHITECTURE FUN OF TE IS
PROCESS(G1, G2, G3)
Begin
  If G1='1' and G2='0' and G3='0' Then
    Y0<=NOT C (NotA) AND (NOT B) AND (NOT C));
    Y1<=NOT C (NOT A) AND (NOT B) AND C );
    - - -
    Y7<=NOT(A AND B AND C);
  Else
    Y0<='1';
    - - -
    Y7<='1';
  END IF
END Process
END Architecture FUN.
```

4.20.

```
Architecture FUN OF GD IS
  Process (CLK);
Begin
IF CLK' event and CLK=1 Then
  If y='1' Then
    Case F Is
    when "000"=>F<="001";
    - - -
    when "111"=>F<="000";
    END CASE;
```

```
elsif  y='0' Then
  case F IS
    when "000" => F<="111";
    when "111" => F<="110";
  END CASE
END IF
END IF
END Process
END Architecture 加FUN;
```