第四章

4.1

简单可编程逻辑器件（SPLD）包括可编程只读存储器(PROM)、可编程逻辑阵列器件（PLA）、可编程阵列逻辑器件（PAL），以及通用阵列逻辑器件（GAL）。

它们的共同点是以与阵列和或阵列作为核心逻辑资源，能够以“积之和”的形式实现布尔逻辑函数。

PROM 的特点是:与阵列固定，或阵列可编程。

PLA 的特点是：与阵列和或阵列都可编程。

PAL 的特点是：与阵列固定，或阵列可编程。

GAL 的特点是：它是在 PAL 的基础上，吸收了先进的浮栅技术，输出端都集成着一个可编程的输出逻辑宏单元（OLMC），通用性更强。

4.2

（1）8421 码至余三码

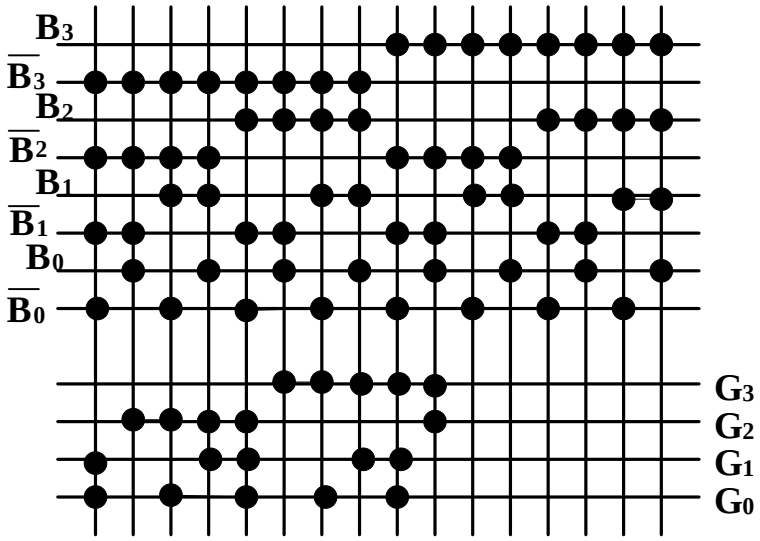
真值表如下：

8421 码				余三码			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	1	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1

最小项表达式为：

$G^3 = \sum(5,6,7,8,9)$ $G^2 = \sum(1,2,3,4,9)$ $G^1 = \sum(0,3,4,7,8)$ $G^0 = \sum(0,2,4,6,8)$

阵列图为：



（2）二进制码转至 2421

真值表如下：B 为二进制码，G 为 2421 码

B3	B2	B1	B0	G10	G3	S2	S1	S0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	1	0	1	1
0	1	1	0	0	1	1	0	0
0	1	1	1	0	1	1	0	1
1	0	0	0	0	1	1	1	0
1	0	0	1	0	1	1	1	1
1	0	1	0	1	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1
1	1	1	0	1	0	1	0	0
1	1	1	1	1	1	0	0	0

最小项表达式为：

$G_{10} = \sum(10,11,12,13,14,15)$ $G_3 = \sum(5,6,7,8,9,15)$ $G_2 = \sum(4,6,7,8,9,14)$

$G_1 = \sum(2,3,5,8,9,12,13,15)$ $G_0 = \sum(1,3,5,7,9,11,13,15)$

阵列图为：略

(3)典型 Gray 码至 8421

真值表如下：B 为典型 Grey 码，G 为 8421 码

B3	B2	B1	B0	G10	G3	S2	S1	S0
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	1	0
0	0	1	0	0	0	0	1	1
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	1	0	1
0	1	0	1	0	0	1	1	0
0	1	0	0	0	0	1	1	1
1	1	0	0	0	1	0	0	0
1	1	0	1	0	1	0	0	1
1	1	1	1	1	0	0	0	0
1	1	1	0	1	0	0	0	1
1	0	1	0	1	0	0	1	0
1	0	1	1	1	0	0	1	1
1	0	0	1	1	0	1	0	0
1	0	0	0	1	1	0	0	0

最小项表达式为：

$G_{10} = \sum(8,9,10,11,14,15)$ $G_3 = \sum(12,13)$ $G_2 = \sum(4,5,6,7,8,9)$

$G_1 = \sum(2,3,4,5,10,11)$ $G_0 = \sum(1,2,4,7,8,11,13,14)$

阵列图为：略

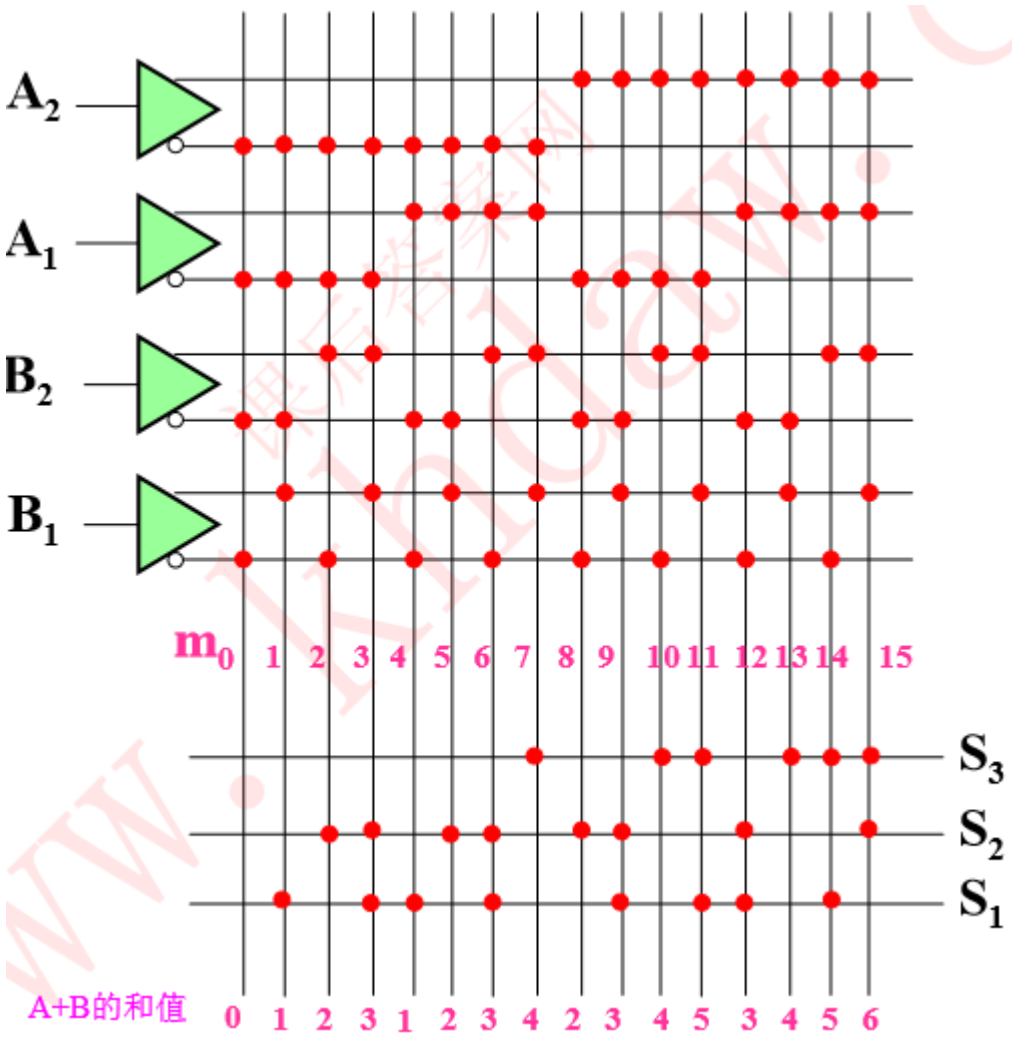
4.3 二位加法器

设 A，B 为两个二位二进制数，S 为相加结果，真值表如下：

A2	A1	B2	B1	S3	S2	S1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0

1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

$S_3 = \sum(1,3,4,6,9,11,12,14)$ $S_2 = \sum(2,3,5,6,8,9,12,15)$ $S_1 = \sum(7,10,11,13,14,15,)$

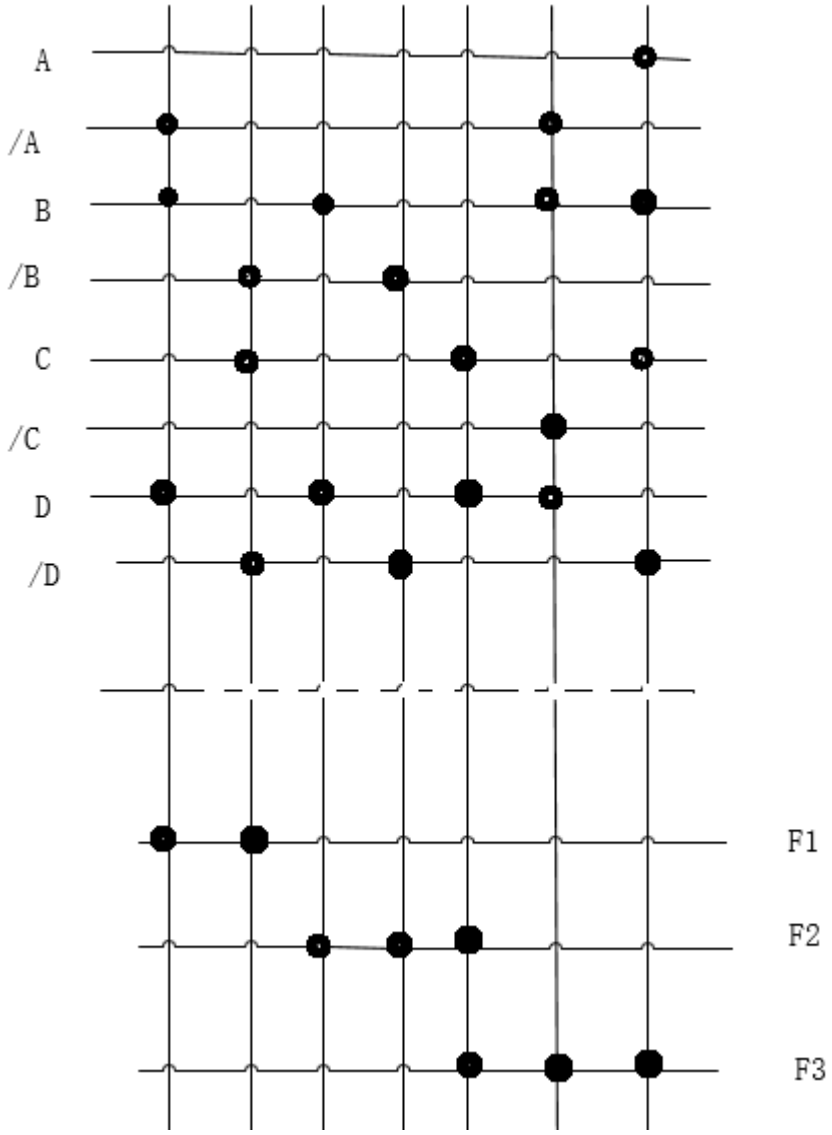


4.4

$L = \overline{A}\overline{B}C\overline{D} + AB\overline{C}\overline{D} + \overline{B}CD + \overline{A}\overline{B}CD + A\overline{B}\overline{C}\overline{D}$

4.5

可以利用 PLA 阵列图表示所要求实现的逻辑函数，如图：



4.6

组合 PAL 器件输出不带寄存器，每组与阵列的输出被固定连接到或阵列的输入端。输出引脚分为 O 输出和 IO 输出。

而时序 PAL 器件输出部分具有输出寄存器，输出引脚分为 IO 输出和寄存器输出，都具有三态输出的功能，且都为低有效。

R 系列不带异或门，触发器的值来自与或阵列，触发器的输出通过三态缓冲器连到输出引脚，而且寄存器输出引脚不能当成输入引脚使用。

X 系列带有异或门，寄存器中每个触发器的输入信号来自一个异或门的输出，而每个异或门的两个输入分别来自两个较小的与或阵列的输出，与 R 系列的寄存器输出类似，但产生次态逻辑的电路结构不同。X 系列的输出结构特别适用于完成算术运算的场合。

4.7

4.8

4.9

4.10

GAL 器件在 PAL 器件的基础上做了一些改进。在工艺方面，GAL 器件采用高速电可擦除除 CMOS 的 E^2CMOS 工艺制作，使其具有了可测试性、低功耗、高集成性、高速率和可重复编程性。在结构方面，GAL 器件的结构是通用的，为设计者提供了最大的灵活性。器件内增加了可被编程的保密位，以防对逻辑的复制。

4.11

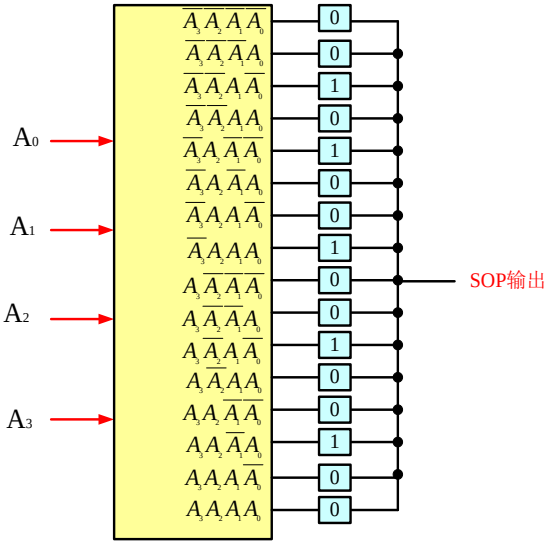
在结构方面，CPLD(Complex Programmable Logic Device)，即复杂可编程逻辑器件，是在 SPLD 的基础上发展起来的，是利用可编程的互连总线连接起来的多路 SPLD。CPLD 为逻辑板块编程，即以逻辑宏单元为基础，加上内部的与或阵列和外围的输入输出模块，不但实现了除简单逻辑控制之外的时序控制，又扩大了整个系统的应用范围。

FPGA（Field－Programmable Gate Array），即现场可编程门阵列，它的结构比早期的 CPLD 更复杂，它采用类似掩模可编程门阵列结构，结合了可编程逻辑器件的特性，继承了门阵列逻辑器件密度高和通用性强的优点。

在编程方式上，FPGA 比 CPLD 具有更大的灵活性。CPLD 通过修改具有固定内连电路的逻辑功能来编程，FPGA 主要通过改变内部连线的布线来编程；FPGA 可在逻辑门下编程，而 CPLD 是在逻辑块下编程。CPLD 主要是基于 E^2PROM 或 FLASH 存储器编程，编程次数可达 1 万次，优点是系统断电时编程信息也不丢失。CPLD 又可分为在编程器上编程和在系统编程两类。FPGA 大部分是基于 SRAM 编程，编程信息在系统断电时丢失，每次上电时，需从器件外部将编程数据重新写入 SRAM 中。其优点是可以编程任意次，可在工作中快速编程，从而实现板级和系统级的动态配置。

4.12

4 变量 LUT 编程



4.13

MAX7000S 是在系统可编程的 CPLD 器件，内部结构是基于乘积项，也就是与或阵列结构。采用 E^2PROM 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，又能和 3.3V 的器件兼容。MAX7000 是高成本 CPLD，规模比较大。

4.14

EPM7128S 包括逻辑阵列 LAB、宏单元、I/O 控制块以及可编程连接阵列 PIA。

逻辑阵列块包含 16 个宏单元，每个宏单元能单独配置为组合逻辑或者时序逻辑。多个逻辑阵列块通过可编程互连阵列 PIA 连接。PIA 接受来自专用输入引脚、I/O 引脚和宏单元的信号。I/O 控制块允许各 I/O 引脚配置为输入、输出或双向 I/O。

4.15

XC400 主要由可编程逻辑块 CLB、可配置存储器 SRAM 阵列、可编程输出输出块以及可编程内部连线 PI 组成。

CLB 给予查找表结构，还包括 2 个触发器，用来实现时序逻辑。

可配置存储器是一种静态存储器，由两个 CMOS 反相器和一个用来控制读写的 MOS 传输开关构成。

可编程输入输出块 IOB 可以灵活编程，以实现不同逻辑功能，满足不同逻辑的需求。

可编程内部连线 PI 是 XC4000 系列中特殊的内部连线，连线资源由水平和垂直的布线通道构成。

4.16

XC4000 的一个 CLB 包括三个查找表，其中两个为 4 输入查找表，第三个查找表的输入由前两个查找表的输出提供，从而实现 9 输入的逻辑功能。

4.17

源代码：

```
        LIBRARY   IEEE；
        USE   IEEE.STD_LOGIC_1164.ALL；

ENTITY   sop   IS
            PORT (a，b，c，d，e，f：IN STD_LOGIC；
                      x：OUT STD_LOGIC）；
        END sop；

        ARCHITECTURE sop_arc OF sop IS
            BEGIN
            x＜＝(a AND b) OR (c AND d) OR (e AND f)；
        END sop_arc；
```

4.18

源代码：

```
        LIBRARY   IEEE；
        USE   IEEE.STD_LOGIC_1164.ALL；

ENTITY   boolean   IS
            PORT (a，b，c：IN STD_LOGIC；
                          f：OUT STD_LOGIC）；
```

```
END boolean ;

ARCHITECTURE boolean_arc OF boolean IS
    BEGIN
    f<=(a OR (NOT b) OR c) AND (a OR b OR (NOT c)) AND ((NOT a) OR (NOT
b) OR (NOT c));
END boolean_arc;
```

4.19

源代码：

```
LIBRARY  IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;

ENTITY  decoder_3_to_8  IS
    PORT （a，b，c，g1，g2a，g2b：IN STD_LOGIC;
            y：OUT  STD_LOGIC _VECTOR（7 downto 0））;
  END  decoder_3_to_8;

ARCHITECTURE  rt1  OF  decoder_3_to_8  IS
     SIGNAL  indata：STD_LOGIC _VECTOR（2 downto 0）;
  BEGIN
    indata<=c & b & a;
    PROCESS（indata，g1，g2a，g2b）
      BEGIN
        IF（g1='1' AND g2a='0' AND g2b='0'）THEN
CASE  indata  IS
                WHEN "000"=>y<="11111110";
                WHEN "001"=>y<="11111101";
                WHEN "010"=>y<="11111011";
                WHEN "011"=>y<="11110111";
                WHEN "100"=>y<="11101111";
                WHEN "101"=>y<="11011111";
                WHEN "110"=>y<="10111111";
                WHEN others=>y<="01111111";
            END CASE;
          ELSE
            y<="11111111";
        END IF;
    END PROCESS;
  END  rt1;
```

4.20

```
LIBRARY  IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;
```

```
ENTITY  gray_count  IS
   PORT （clk，y：IN STD_LOGIC；
              qout：OUT STD_LOGIC _VECTOR（2 downto 0））；
END  gray_count；

ARCHITECTURE  arch_gray  OF  gray_count  IS
SIGNAL  iq：  STD_LOGIC _VECTOR（2 downto 0）；
BEGIN
      PROCESS（clk）
          BEGIN
 IF  (clk′event  AND  clk＝′1′)  THEN
IF（y＝′1′）THEN
CASE  iq  IS
                              WHEN  "000"＝＞iq＜＝"001"；
                              WHEN  "001"＝＞iq＜＝"011"；
                              WHEN  "011"＝＞iq＜＝"010"；
                              WHEN  "010"＝＞iq＜＝"110"；
                              WHEN  "110"＝＞iq＜＝"111"；
                              WHEN  "111"＝＞iq＜＝"101"；
                              WHEN  "101"＝＞iq＜＝"100"；
                              WHEN  others＝＞iq＜＝"000"；
                         END CASE；
                    END IF；
IF（y＝′0′）THEN
CASE  iq  IS
                              WHEN  "000"＝＞iq＜＝"100"；
                              WHEN  "100"＝＞iq＜＝"101"；
                              WHEN  "101"＝＞iq＜＝"111"；
                              WHEN  "111"＝＞iq＜＝"110"；
                              WHEN  "110"＝＞iq＜＝"010"；
                              WHEN  "010"＝＞iq＜＝"011"；
                              WHEN  "011"＝＞iq＜＝"001"；
                              WHEN  others＝＞iq＜＝"000"；
                         END CASE；
                     END IF；
               END IF；
       END PROCESS；
qout＜＝iq；
END  arch_gray；
```