Part Ⅳ 可编程逻辑器件 PLD

Lecture 14 可编程逻辑器件*OK*!

一、概述

1、可编程逻辑器件的概念

传统数字系统的设计方法，一种是使用**标准芯片**进行搭建，如前面介绍的各种中、小规模集成电路；另一种是向厂家定制面向特定用途的**专用**

集成电路 ASIC。

可编程逻辑器件简称为 PLD，本身作为一种通用集成电路芯片进行生产，用户可以根据需要编程配置其内部用于连接逻辑门的可编程开关，从而实现满足特定应用需求的逻辑功能。

PLD 的基本结构如图 6.1.1 所示，由输入电路、与阵列、或阵列和输出电路构成。

输入电路主要由缓冲器和反相器构成，能够产生每个输入变量的原和反变量形式的信号，这些信号被送入由多个与门构成的与阵列，由**与阵列**形成输入信号的乘积项输出。与阵列产生的乘积项又被送入由多个或门构成的或阵列，由**或阵列**形成乘积项之和经输出电路输

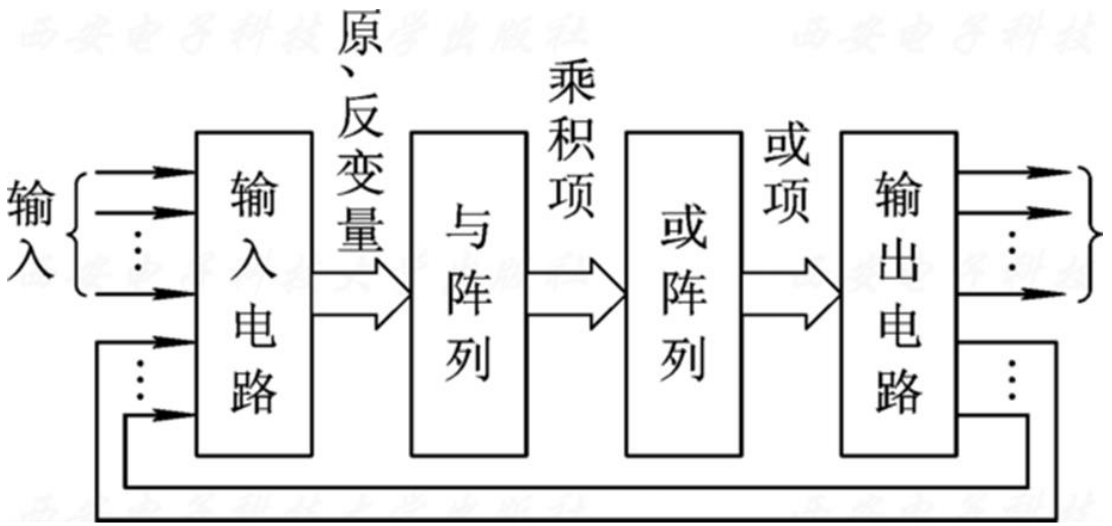


图6.1.1 PLD的基本结构

出。

可编程逻辑器件中，连接到与、或阵列的信号都要经过一个开关，用户可在应用时通过一定的方法改变这些开关的状态，从而改变与、或阵列的连接方式，这一过程就称为编程，相应的开关称为可编程开关。

由于逻辑函数总能表示为乘积项之和的形式，因此通过编程改变与、或阵列的连接方式，就能输

出不同的乘积项之和，从而实现不同的逻辑函数。

2、可编程逻辑器件的发展历程

最早的可编程逻辑器件是出现于 20 世纪 70 年代初的**只读存储器** PROM 和**可编程逻辑阵列** PLA，随后经历了 PAL、GAL、EPLD 到 CPLD 和 FPGA 的演变历程，在**结构、工艺、功能、速度、功耗、集成度和灵活性**等方面都有很大的改进和提高。

只读存储器PROM内部的地址译码器形成固定的与阵列，存储单元构成可编程的或阵列，可对其编程来实现组合逻辑函数，是可编程逻辑器件的雏形。

可编程逻辑阵列PLA是在PROM的基础上形成的一种与、或阵列都可编程的PLD器件，但由于开发工具等原因而没有获得什么实际的应用。

可编程阵列逻辑PAL出现于20世纪70年代

末期，是继 PLA 之后的第一个具有典型实用意义的可编程逻辑器件。PAL **在工艺上采用一次性可编程开关，在结构上由可编程的与阵列和固定的或阵列组成**，工作速度和输出结构的种类较之前的可编程逻辑器件有了很大的进步。

通用阵列逻辑 GAL 是继 PAL 之后，于 20 世纪 80 年代初期，由 Lattice 公司推出的一种比 PAL 更为灵活的可编程逻辑器件。

GAL 在工艺上采用了 EEPROM，具有可擦除、可重新编程的特点；在结构上采用了输出逻辑宏单元的结构形式，能够灵活配置为多种不同的输出结构。因此，GAL 比 PAL 器件更灵活，功能更全面，能够取代大部分的 PAL 器件。

20 世纪 80 年代中期，Xilinx 公司提出了现场可编程的概念，并同时推出了现场可编程门阵列 FPGA 器件。FPGA 在工艺上采用了 SRAM 存储元，

可无限次反复编程；在结构上采用逻辑单元阵列结构，使得器件集成度大为提高，设计更加灵活。目前，多个厂家推出的 FPGA 芯片可使用的门达到千万门级。

与 FPGA 出现的同期，在 1985 年，Altera 公司在 EPROM 和 GAL 的基础上，推出了**可擦除可编程的逻辑器件 EPLD**。EPLD **在工艺上采用 EPROM 或 EEPROM，具有可擦除、可重新编程的特点；基本结**

构和 PAL 及 GAL 相仿，都采用与、或阵列结构，但集成度要比 PAL 和 GAL 高很多。

随着集成电路技术的发展，生产工艺不断改进，Altera、Xilinx 等多家公司都不断推出新的 EPLD 器件，EPLD 器件规模不断扩大，集成度不断提高，逻辑功能也不断增强，由此，**复杂可编程逻辑器件 CPLD** 应运而生。CPLD 在 EPLD 的基础上发展起来，并在结构和工艺上都有所改进，其集成度

更高、性能更好、设计也更灵活。

20 世纪 80 年代末，Lattice 公司又提出了**在系统可编程 ISP** 的概念，并随后推出了**在系统可编程大规模集成电路 ispLSI**，使得基于可编程逻辑器件的数字系统的设计、生产过程更为简化，不需要专门的编程器，也不需要移动器件，就能够对可编程逻辑器件进行在板或在系统的编程，可方便地进行系统调试、测试、更新、维护。在系统可

编程技术已成为当前可编程逻辑器件普遍支持的技术。

3、可编程逻辑器件的分类

可编程逻辑器件的分类方法较多，也不统一。常见的有：按照器件集成度、按照编程元件及编程技术、按照器件的结构特点等分类方法。

按照集成度，PLD 可分为低密度 PLD（LDPLD）

和高密度 PLD（HDPLD）两类。

低密度 PLD 包括前面介绍的 PROM、PLA、PAL 和 GAL，通常也将这些 PLD 称为简单可编程逻辑器件 SPLD。

高密度 PLD 主要包括 EPLD、CPLD 和 FPGA。相对于低密度 PLD，高密度 PLD 在结构方面有了较大的改进，集成度更高、功能更为强大。

目前，新型的高密度PLD器件已开始采用65nm工艺制造，并正向45nm工艺发展，可使用的门达到千万门级，能够嵌入CPU等知识产权核(IP)。

按照器件所采用的编程元件及编程技术，可将PLD分为一次性编程器件和可多次编程的PLD。

一次性编程的PLD通常采用熔丝型的编程开关，使用专门的编程器进行编程，编程过程就是对不需要连接的线路，烧断其可编程开关的熔丝。由

于熔丝的熔断过程不可逆，因此一旦编程就不能再改写， 不适合在数字系统的研制、开发和实验阶段使用。

可多次编程的 PLD 按照其使用的编程元件的结构，又可以分为**浮栅型**编程器件和使用**静态存储元 SRAM** 的编程器件。

浮栅型编程器件使用 EPROM、EEPROM 或闪存作为开关元件，可多次编程，且编程后信息可长期

保存。

采用静态存储元 SRAM 的 PLD 器件利用 SRAM 存储元存储编程信息，不需要专门的编程器，可以直接在电路板上对器件进行编程，能进行无限次反复编程，但其缺点是一旦掉电，信息即丢失(易失性)

EEPROM、闪存和 SRAM 是目前 PLD 应用最多的编程元件。

3. 按结构特点分类

按照器件的结构特点，可将 PLD 分为**乘积项结构**的 PLD 和**查找表结构**的 PLD 两类。

基于乘积项结构 PLD 的主要构成部分是与、或阵列，PROM、PLA、PAL、GAL、EPLD 和大部分的 CPLD **都属于这种结构**。器件通常采用 EEPROM 或闪存作为存储元，集成度一般小于 5000 门/片。

基于查找表结构的 PLD 通常采用 SRAM 存储元存储逻辑函数的真值表信息，使用查找表(LUT)实现基于真值表的逻辑函数。目前，绝大多数 FPGA 都属于这种结构。这种器件设计灵活，集成度高，但由于 SRAM 的易失性，实际应用时通常还需要额外配置一块非易失的存储器件。

二、PLD 的编程元件*

1、熔丝型开关

熔丝型开关是早期 PROM、PLA 和 PAL 等可编程逻辑器件所采用的编程元件。出厂时，所有的熔丝开关都是连通的。用户可以根据需要，借助一定的编程工具熔断选定开关的熔丝，来改变开关状态。

下面以可编程只读存储器 PROM 为例，来介绍熔丝型开关的基本工作原理及与或阵列的构成。

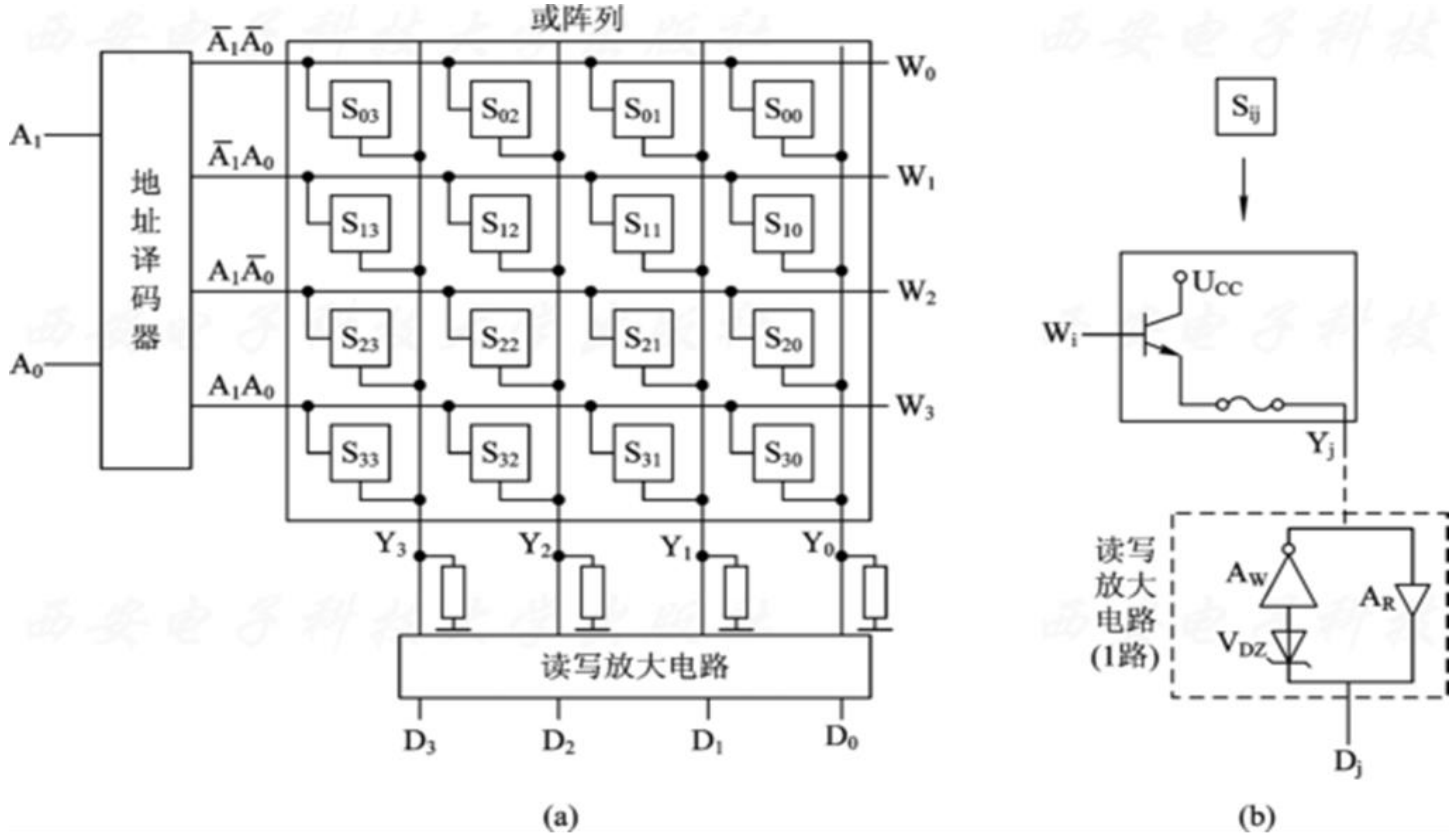


图6.2.1 4×4位PROM

(a) 结构图； (b) 熔丝型开关和读写放大电路

PROM 由**三部分组成：地址译码器、存储矩阵和读写电路**。

图 6.2.1(a)给出了一个 4×4 位 PROM 的结构图。其中，地址译码器对外部送入的地址信号进行译码，输出字选择线 W_0、W_1、W_2、W_3；下方的 4 个下拉电阻和读写放大电路构成读写电路，形成 4 个输出信号(位线)D_3、D_2、D_1、D_0；字线和位线横竖交叉，形成 16 个交叉点，在每个交叉点处接有一个熔丝型开关，构成 4×4 位的存储矩阵。

2、浮栅型编程元件

EPROM： 可擦除可编程只读存储器 EPROM

EEPROM：也可记为 E^2PROM，是电可擦除可编程只读存储器的简称，可以进行电擦除、电编程以及擦除部分信息，比 EPROM 的使用更为方便灵活。

闪存：综合了 EPROM 和 EEPROM 的优点，不但具有 EPROM 高密度、低成本的优点，而且具有 EEPROM 电擦除及快速的优点。

3、SRAM 编程元件

静态随机读写存储器 SRAM 是一种可随机读写的存储器，内部包含大量的 SRAM 存储元，每个存储元可存储 1 位二进制信息。给定一组地址信号，就能够选中其中的若干个存储元(构成存储字)，可读出其中存储的各位二进制信息，或向其写入二进制信息。

SRAM 存储元是 FPGA 通常采用的编程元件，可

用来存储查找表中的真值表(参见 6.5.1 节)，也可用来配置互连线路的连接状态。

三、简单 PLD 的原理与结构

1、PLD 的阵列图符号

1）PLD 的互补输入

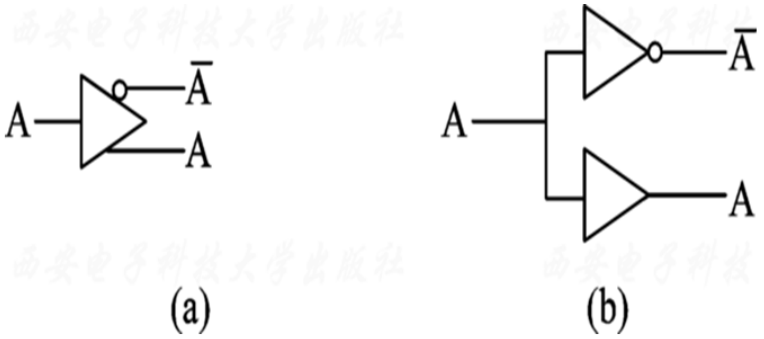


图6.3.1 互补输入

(a) PLD表示； (b) 一般表示

PLD 输入电路形成输入信号的原、反变量互补输入。

PLD 中通常使用的互补输入表示与一般的互补输入表示分别如图 6.3.1(a)和(b)所示。

2）阵列线连接的表示

PLD 中的阵列线连接状态可分为**未连接**、**固定连接**和**编程连接**三种。

图6.3.2 阵列线连接

未连接指水平和垂直线之间没有连接；固定连接指出厂时已连接；编程连接指通过编程可编程开关使水平和垂直线相互连接导通。

3）与、或阵列的表示

图 6.3.3(a)给出了一个传统的多输入与门的逻辑符号表示。

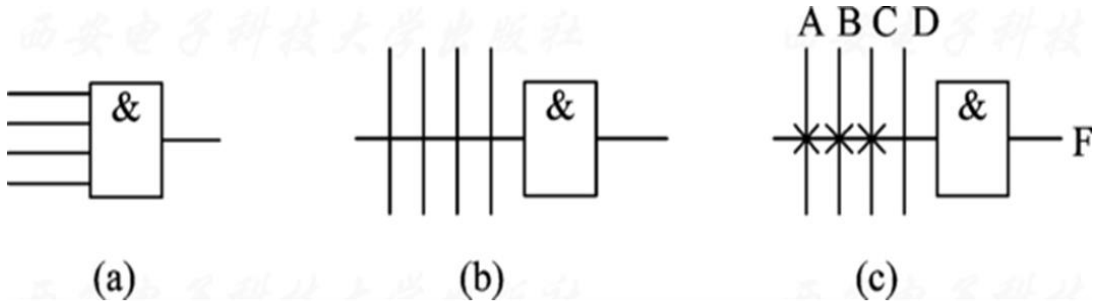


图6.3.3 多输入与门

(a) 一般表示；(b) 阵列表示；(c) F=ABC

为便于 PLD 阵列结构的表示，对应于这种传统表示方法，PLD 中的与阵列通常采用如图6.3.3(b)所示的逻辑符号表示，水平信号线表示与阵列的输入，四根垂直的信号线表示可连接到

与阵列的输入信号。类似的表示方法也可以用于PLD 的或阵列。

例如，图 6.3.3(c)所示的阵列线连接状态，表示与阵列输出 F=ABC。

4）数据选择器的表示

PLD 的宏单元以及 I/O 模块等单元中通常会设置一些数据选择器，以增加器件的灵活性。

下图给出了一个 4 选 1 数据选择器在 PLD 中的一般表示方法。

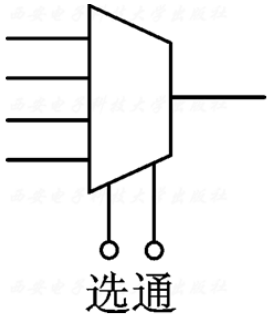


图6.3.4 PLD中的数据选择器表示

数据选择器下方的端子表示由 EPROM 或 SRAM 编程开关控制的数据选择器的选通(地址)信号，在不强调选通信号的情况下，也可以省去该信号。

【例 6.3.1】用 16×4 位的 PROM 实现 4 位二进制码 B_3、B_2、B_1、B_0 到循环码 G_3、G_2、G_1、G_0 的转

换，画出阵列图。

解答：列出真值表，并根据真值表写出各输出变量的标准与或式：

表6.3.1 二进制码转换为循环码的真值表

二进制码				循环码			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

$$G_3(B_3,B_2,B_1,B_0)=\sum m(8,9,10,11,12,13,14,15)$$

$$G_2(B_3,B_2,B_1,B_0)=\sum m(4,5,6,7,8,9,10,11)$$

$$G_1(B_3,B_2,B_1,B_0)=\sum m(2,3,4,5,10,11,12,13)$$

$$G_0(B_3,B_2,B_1,B_0)=\sum m(1,2,5,6,9,10,13,14)$$

最后**根据标准表达式画出阵列图**：用 PROM 实现的二进制码到循环码转换的阵列图如图 6.3.5

所示。

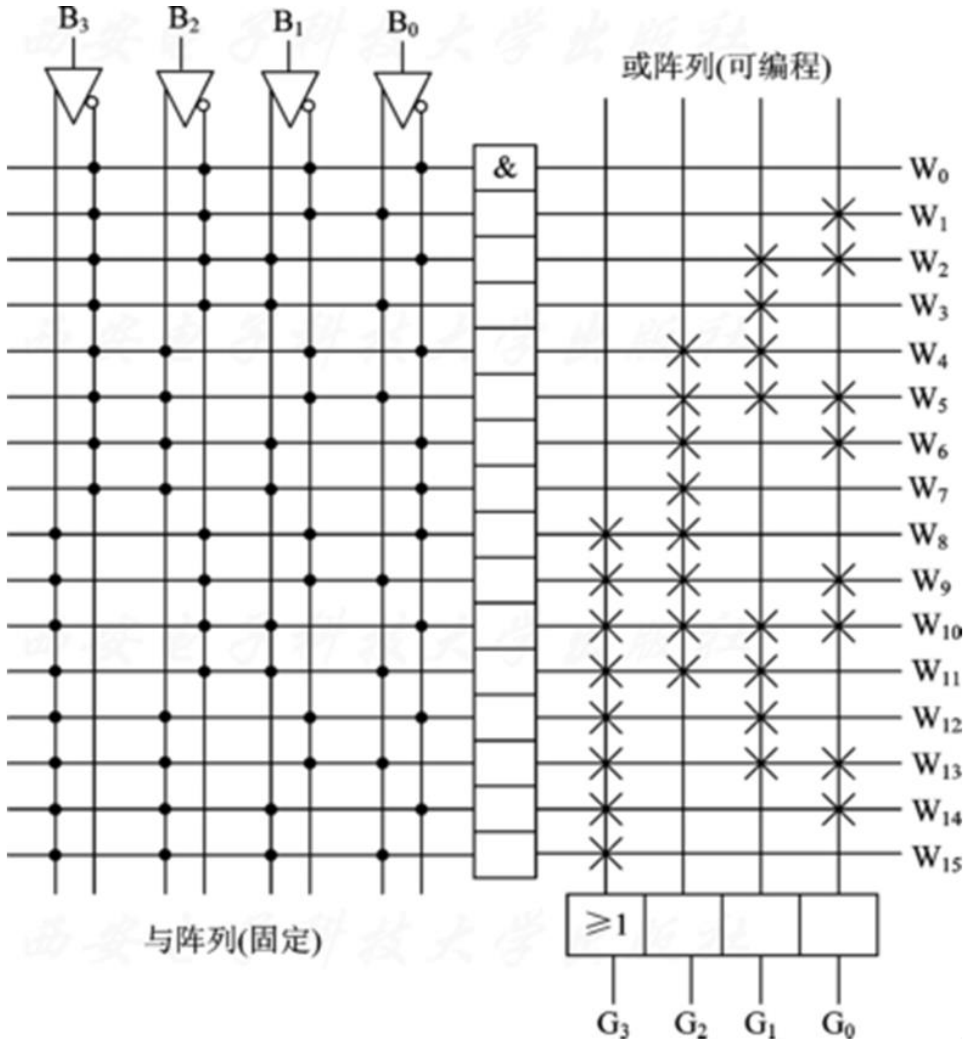


图6.3.5 二进制码到循环码转换的PROM阵列图

PROM 的地址译码器构成与阵列，是固定不可编程的，连接方法固定，用·表示连接状态。或阵列是可编程的，在需要连接的位置画上×表示编程后保持连接导通。

2、可编程逻辑阵列 PLA

用 PROM 实现逻辑函数时，由于其与阵列固定，形成输入信号的全部最小项，用到的单元个数多，不利于实现较复杂的逻辑电路。

PLA 在 PROM 的基础上进行改进，形成一种与、或阵列两级都可编程的器件。

PLA 的基本结构如图 6.3.6 所示。输入电路产

生的互补输入送入可编程的与阵列，与阵列编程产生乘积项，送入可编程的或阵列，或阵列编程形成乘积项之和输出。

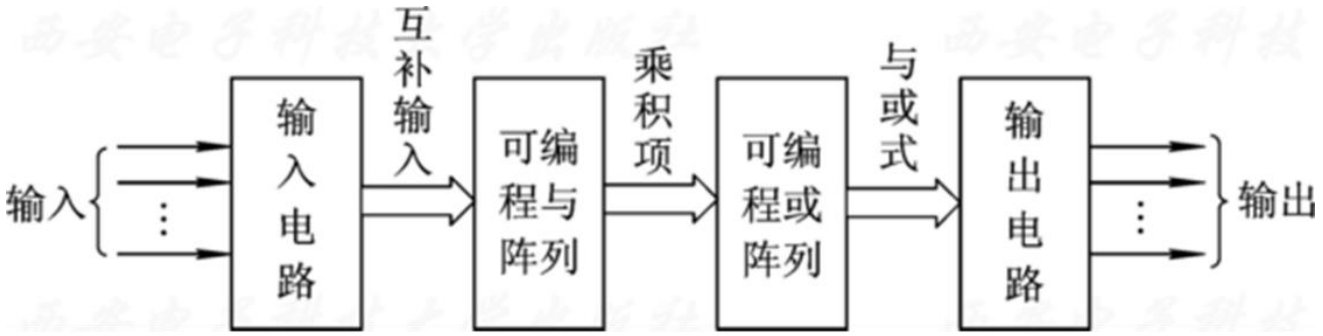


图6.3.6 PLA的基本结构

图 6.3.7 给出了一个实现 3 输入 2 输出逻辑函数的 PLA 内部阵列逻辑示意，用来说明 PLA 器

件内部的逻辑结构，实际应用中的 PLA 器件规模要比图示的逻辑阵列大许多。

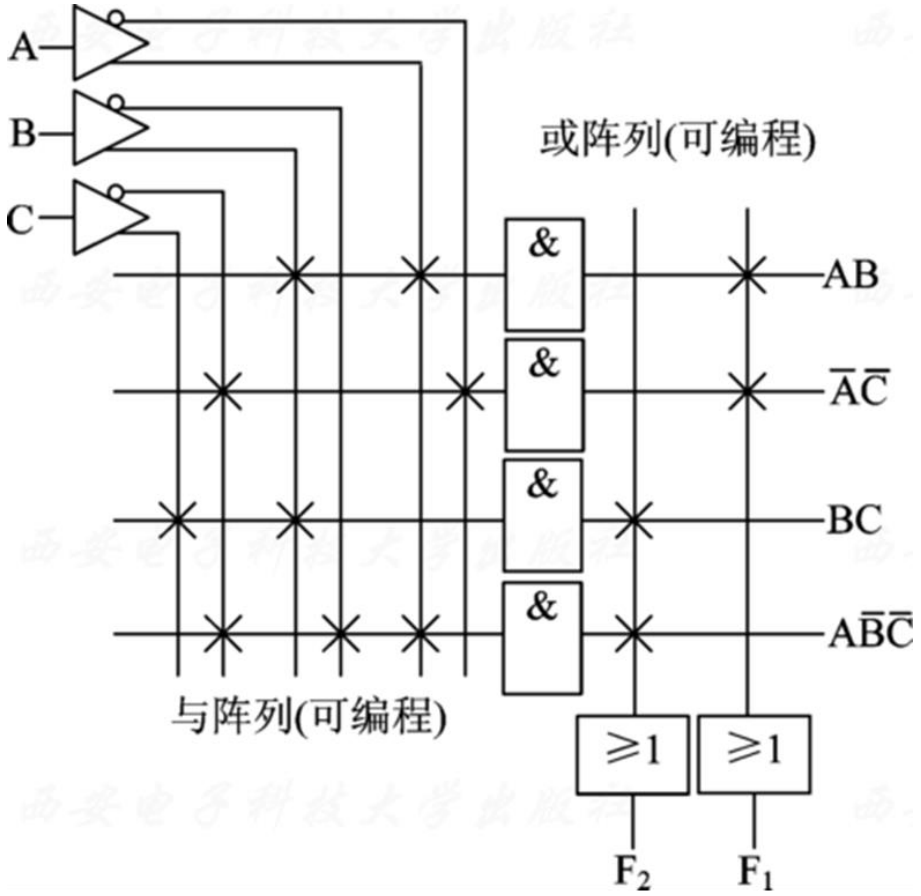


图6.3.7 一个3输入2输出的PLA

图中，PLA 的与阵列的编程状态形成了四个乘积项，这些乘积项被送入或阵列，由或阵列编程后形成了两个输出函数：

$$F_1 = AB + \overline{A}\overline{C}，\quad F_2 = BC + A\overline{B}\overline{C}$$

PLA 器件的规模通常用输入端个数、乘积项个数和输出端个数来定义。由于 PLA 集成度较低，乘积项个数有限，因此在用 PLA 设计逻辑电路时需要尽可能地化简逻辑函数，减少乘积项的数目。对多输出逻辑函数，则要尽量提取、利用公共乘积项。

使用 PLA 设计实际的逻辑电路时，可以借助计算机辅助设计工具生成 PLA 器件的**编程表**或**编**

程文件，由编程表或编程文件来描述将哪些输入信号编程连接到 PLA 的与门，以及将与阵列产生的哪些乘积项编程连接到 PLA 的或门，也就是 PLA 的熔丝映像，而不需要给出类似于图 6.3.7 所示的阵列连接图。

设计好的编程表或编程文件可以提交给制造厂商，由厂商根据用户的编程需求生产定制的 PLA，这样的 PLA 称为**掩膜 PLA**。另一种 PLA 则允许用户

使用专门的编程器，直接对 PLA 器件进行现场编程，来实现需求的逻辑功能，这样的 PLA 称为**现场可编程逻辑阵列 FPLA**。图 6.3.7 所示的阵列图，实际上是基于 FPLA 的实现，但在不特别强调哪种 PLA 的情况下，许多时候都简单称为 PLA。

3、可编程阵列逻辑 PAL

可编程的阵列逻辑 PAL 是一种具有**可编程与阵列**和**固定或阵列**的可编程逻辑器件。

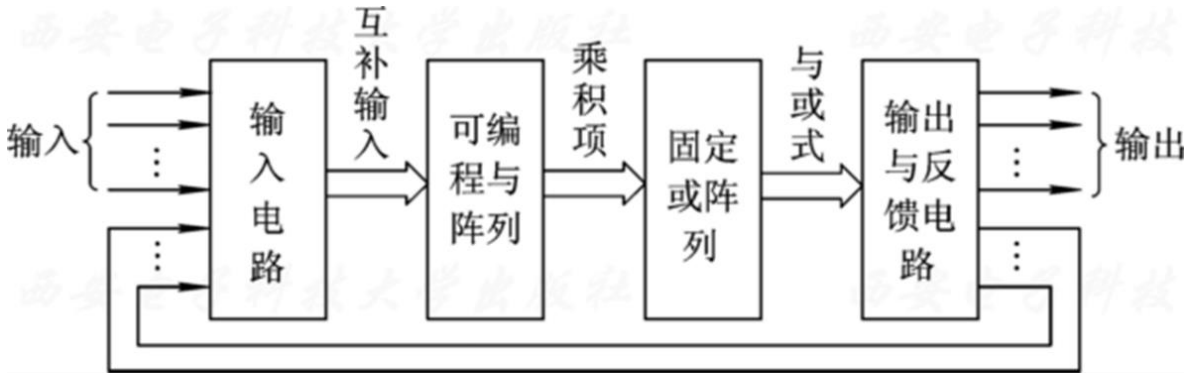


图6.3.8 PAL的基本结构

PAL 的基本结构如图 6.3.8 所示，输入电路产生输入信号的互补输入送入可编程的与阵列，与阵列编程形成乘积项，送入固定的或阵列形成乘

积项之和输出，输出信号也可以被反馈送入输入电路。

一个实现 4 输入 4 输出逻辑函数的 PAL 内部阵列结构如图 6.3.9 所示。

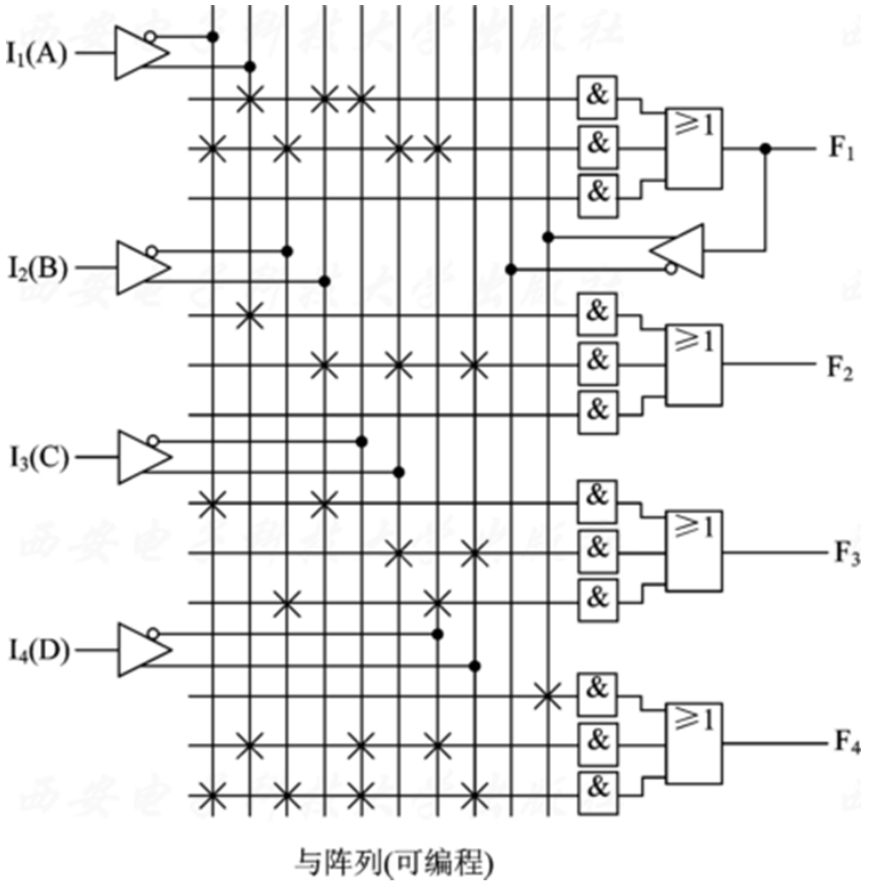


图6.3.9 一个4输入4输出的PAL

图中，PAL 有 4 个输入端 I_1、I_2、I_3 和 I_4，4 个输出端 F_1、F_2、F_3 和 F_4，输出 F_1 又可反馈输入到与阵列。

内部阵列分为4个**单元**，每个单元包含 3 个与门和 1 个或门，每个与门可编程形成关于 5 个输入信号(4 个外部输入和 1 个来自 F_1 的反馈输入)的任意乘积项，而每个或门的输入则是固定的，只能将 3 个与门形成的最多 3 个乘积项相或后输出，即与阵列是可编程

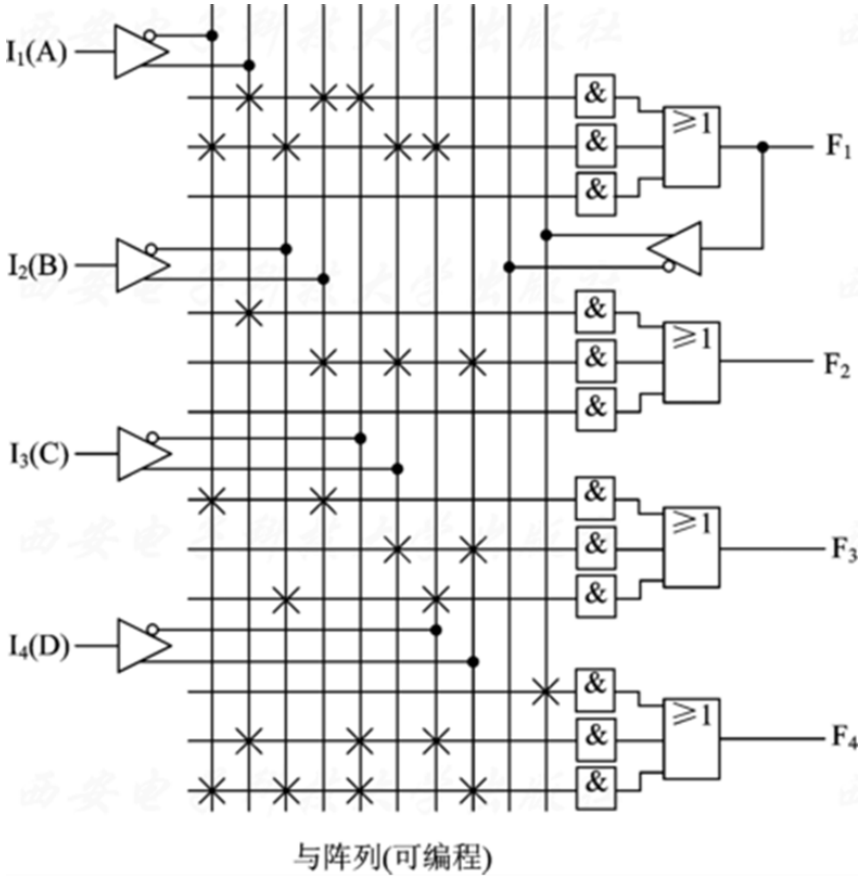


图6.3.9 一个4输入4输出的PAL

的，而或阵列是固定的。

在图示的编程状态下，输出端实现的逻辑函数是：

$$F_1 = AB\bar{C} + \bar{A}\bar{B}C\bar{D} \quad ;$$

$$F_2 = A + BCD \quad ;$$

$$F_3 = \bar{A}B + CD + \bar{B}\bar{D};$$

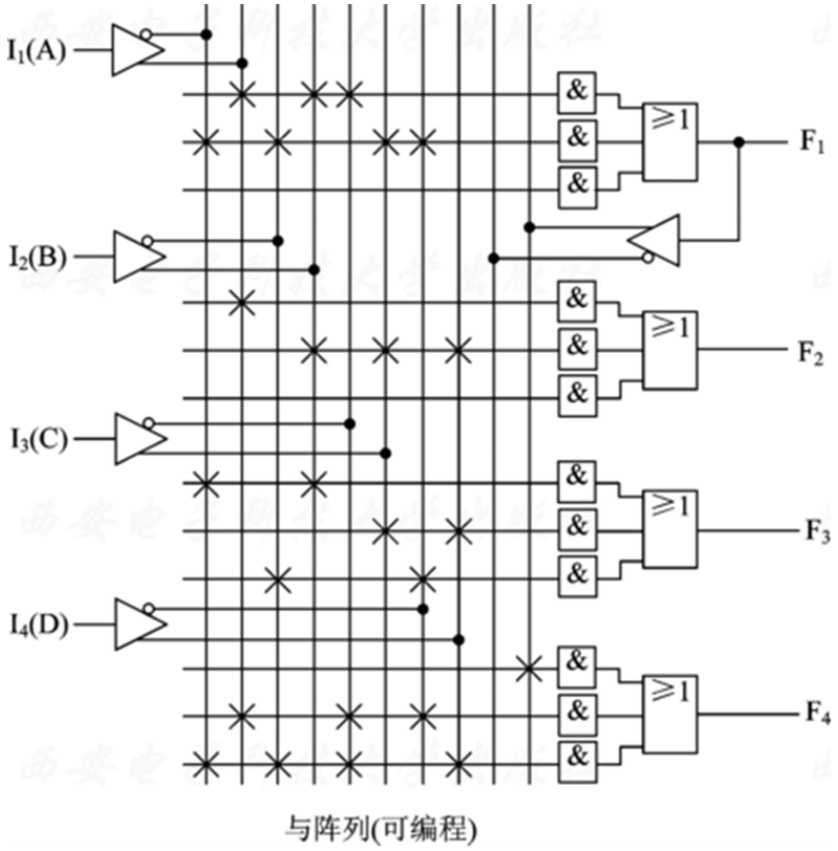


图6.3.9 一个4输入4输出的PAL

$$F_4 = F_1 + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D = AB\bar{C} + \bar{A}\bar{B}C\bar{D} + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$$

其中，F_1 和 F_2 只用到了两个与门形成的两个乘积项，第三个与门输入端没有编程输入，即与门输出总为 0；F_4 由 4 个乘积项构成，利用 F_1 的输出反馈作为其输出，解决了 PAL 的一个单元不能满足乘积项个数需求的问题。

PAL 器件产品按多种不同的规格进行生产制造，包括不同的输入端个数、输出端个数以及每个或门的不同输入端个数等。同时，为适应不同的应

用环境，许多 PAL 器件在或门输出端增加了一些额外的电路，形成具有不同输出和反馈结构的 PAL。

例如，一种带**反馈寄存器**的 PAL 输出端电路结构如图 6.3.10 所示，在 PAL 或门的输出端增加了一个 D 触发器。

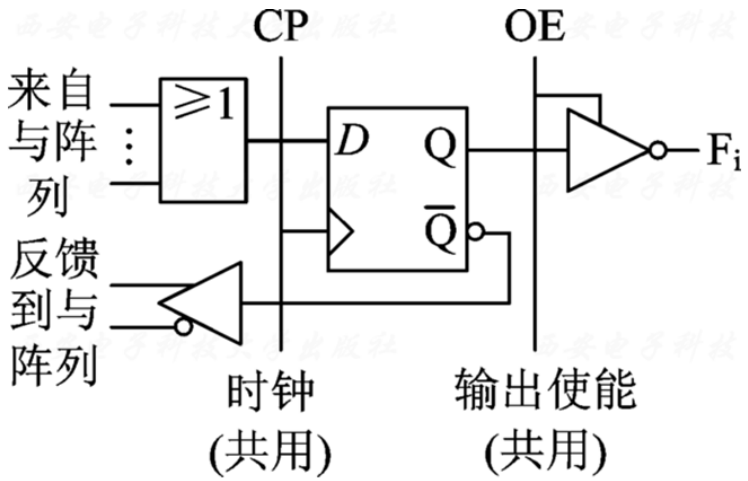


图6.3.10 带反馈寄存器的PAL输出端

PAL 的输出与反馈结构除了上面介绍的基本与或阵列型和带反馈的寄存器型结构外，还有可

编程输入/输出型、算术选通反馈型、异步可编程寄存器输出型、乘积项公用输出、宏单元输出结构等。

4、通用阵列逻辑 GAL

通用阵列逻辑器件在组成结构中采用了宏单元 MC 的结构形式，允许用户通过编程将其配置为不同的结构形式，因而功能更加全面，结构更为灵活。

GAL 的基本结构可类似于 PAL，具有可编程的与阵列和固定的或阵列，称为 PAL 型 GAL；也可类似于 PLA，与、或阵列两级都可编程，称为 PLA 型 GAL。

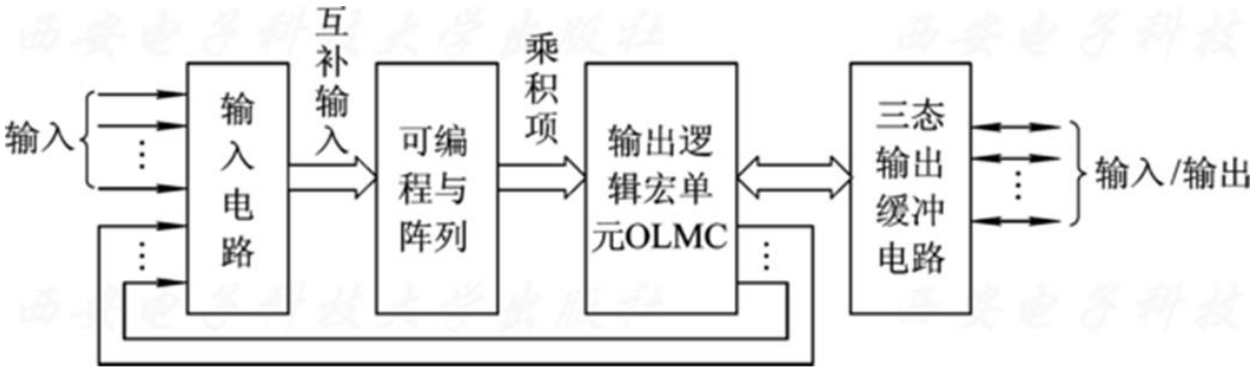


图6.3.11 PAL型GAL的基本结构

PAL 型 GAL 的基本结构如图 6.3.11 所示，由**输入电路**、**可编程与阵列**、**输出逻辑宏单元 OLMC** 和**三态输出缓冲电路**组成。

输出逻辑宏单元 OLMC 包括**固定的或阵列**以及一些

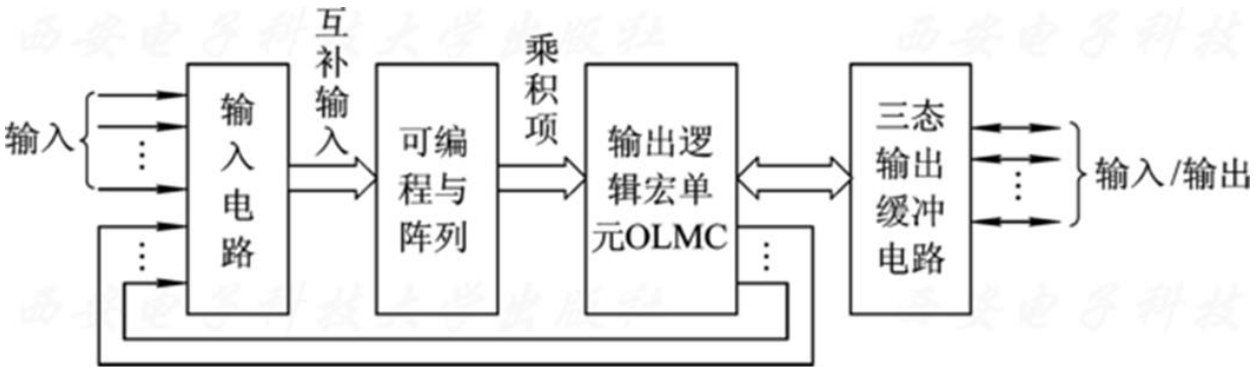


图6.3.11 PAL型GAL的基本结构

可编程的配置电路，这些电路可通过编程配置形成不同的输出与反馈结构，从而以灵活多样的方式满足不同的应用需求。

GAL 的结构示意如图 6.3.12 所示。图中，I_1、

I_2为输入端，I/O_1、I/O_2为可编程输入/输出端，2 个虚线框内为 2 个输出逻辑宏单元。

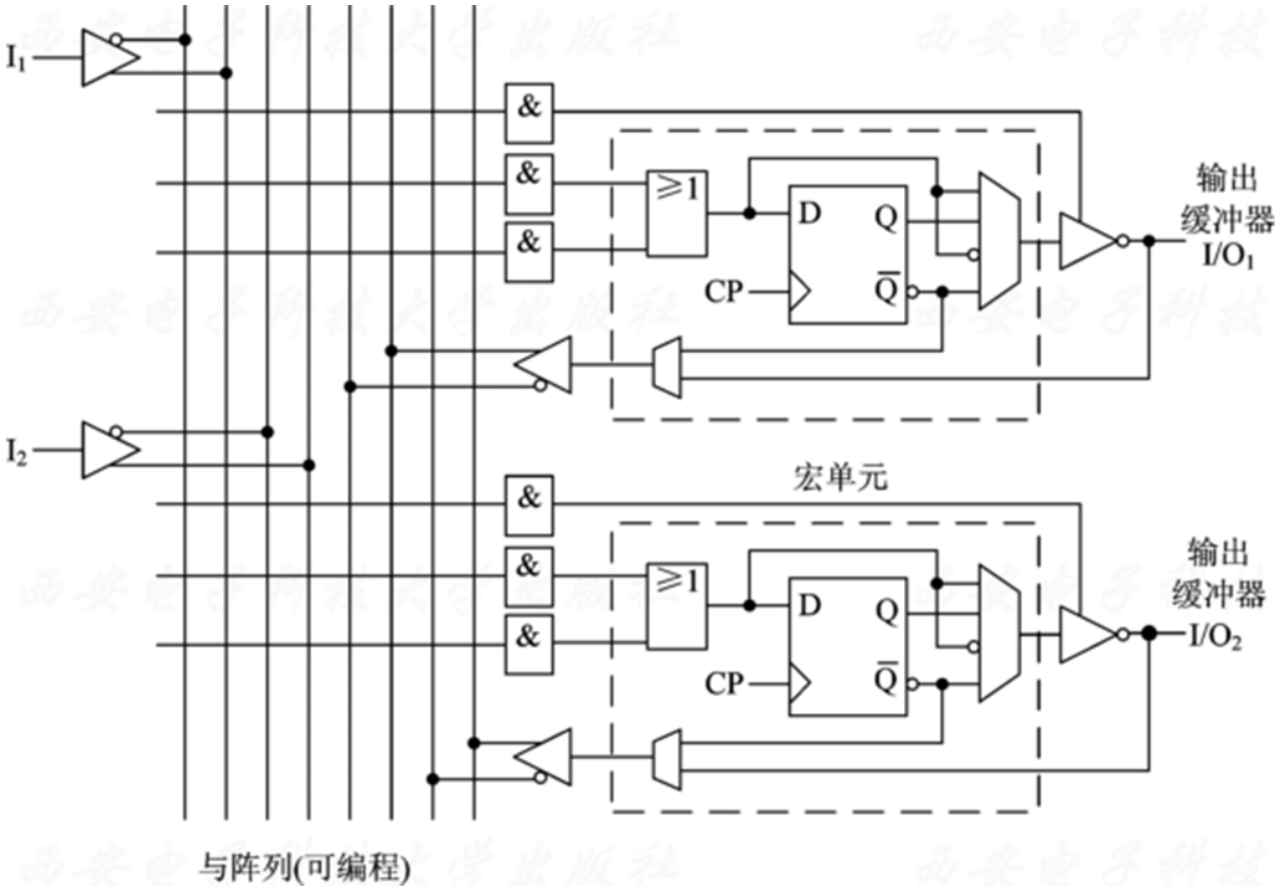


图6.3.12 GAL的结构示意

输出逻辑宏单元由具有固定输入端个数的或门、D 触发器和两个数据选择器构成，所有宏单元中的或门构成了 GAL

的固定或阵列。

图示结构按宏单元可划分为两个部分，每个部分分配有 **3 个乘积项**(与门)，**每个乘积项都可通过对与阵列编程形成。**

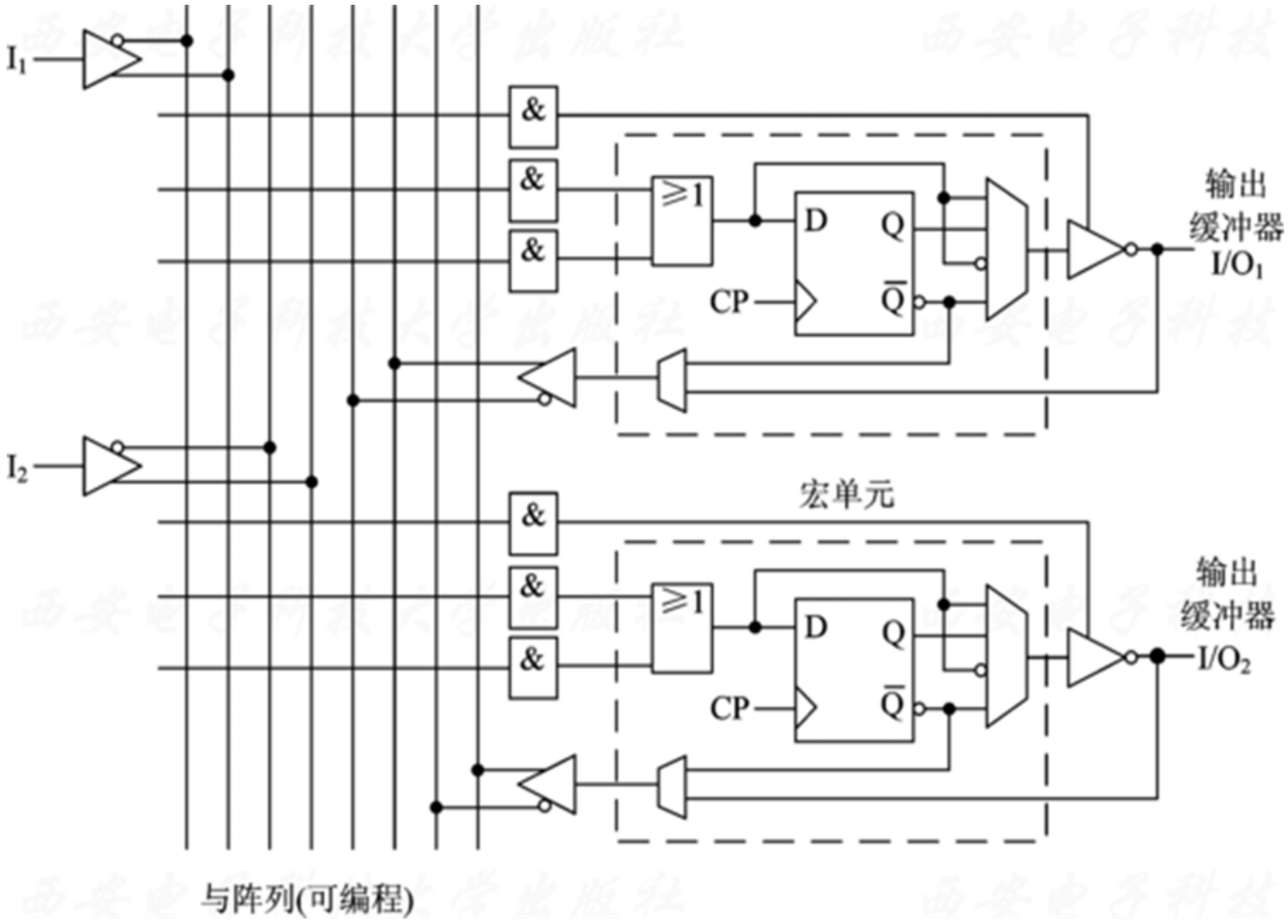


图6.3.12 GAL的结构示意

第一个乘积项作为输出三态缓冲器的输出使能控制信号，若为高电平有效，则宏单元输出信号经输出三态缓冲器输出；若为低电平无效，则三态缓冲器呈现高阻状态，此时，与该三态缓冲器相连的 I/O 端即可成为输入端，能够将输入信号送至与阵列。两个 I/O 端是作为输出端，还是作为输入端，由每部分的第一乘积项编程控制，因此称为**可编程输入/输出端**。

每个部分的其他两个乘积项则送入宏单元中的或门(固定的或阵列)，形成乘积项之和。

宏单元中的D触发器可在时钟脉冲CP作用下寄存或门的输出。

两个数据选择器则可以分别选择要输出的信号及反馈到与阵列的反馈信号。

右边的四选一数据选择器是输出信号选择器，

可选择旁路 D 触发器，直接将或门的输出经输出三态缓冲器缓冲输出，形成组合型输出；或选择 D 触发器的输出，形成时序型输出。

不论是组合型输出还是时序型输出，都**可以选择输出信号的极性**，即以原或反变量的形式输出。

下方的二选一数据选择器是反馈信号选择器，用来选择反馈至与阵列的信号。若选择 D 触发器

的$\bar{Q}$端反馈，则形成器件内部的寄存器反馈；若选择输出缓冲器 I/O 端反馈，则根据第一乘积项控制的输出三态缓冲器的状态，该信号可以是外部输入，也可以是宏单元输出的反馈。

宏单元中数据选择器的选通信号(图中未画出)由 EEPROM 构成的可编程开关控制，编程配置这些开关的状态即能实现不同的输出与反馈结构。

由于可编程的宏单元结构可被配置成多种工

作模式，因此 GAL 器件功能更加全面，结构灵活，通用性强，少数几种 GAL 器件就几乎可取代大多数的中小规模数字集成电路和 PAL 器件。目前，在一些小规模、低成本的场合，GAL 器件仍然有着重要的应用。

四、复杂可编程逻辑器件 CPLD

目前，大多数的 CPLD 都采用**乘积项**和**宏单元**结构，使用 EPROM、EEPROM 或 FLASH 制造工艺。

CPLD 的一般结构如图6.4.1所示，多个较小规模的**PLD 模块**集成在一个芯片上，各个模块间通过全局布线的**可编程开**

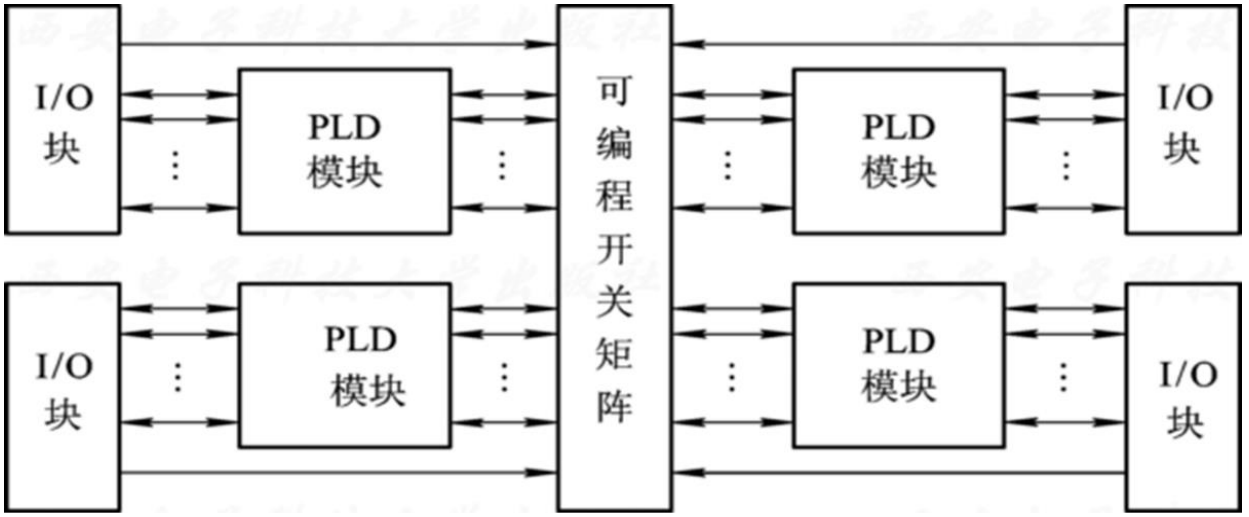


图6.4.1 CPLD的一般结构

关矩阵相互连接，同时由 I/O 模块提供芯片内部逻辑与外部引脚间的连接与封装。PLD 模块用于实现目标电路中相对独立的功能模块，多个 PLD 模块通过开关矩阵相互连接，可实现大规模的复杂逻辑电路。

PLD 模块可以类似于与、或阵列都可编程的 PLA，也可以类似于具有可编程与阵列和固定或阵列的 PAL。

图6.4.2给出了一个类似于PAL的PLD模块以及它通过可编程开关矩阵与其他模块互连的示意图。图中，**PLD模块由3个宏单元组成，每个宏单元包含一个4输入或门、一个异或门、一个触发器、一个二选一数据选择器和一个三态缓冲器。**

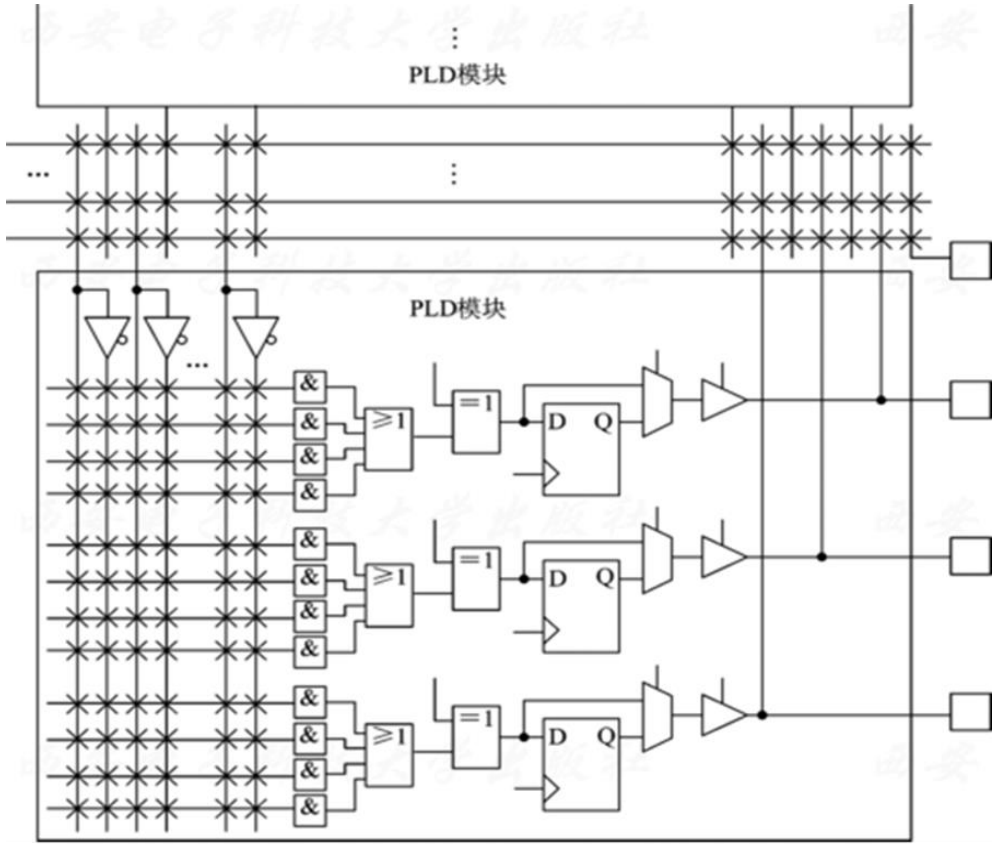


图6.4.2 PLD模块及其互连示意

对应于模块的**可编程与阵列**，只对输入至该模块的信号形成乘积项。4 个乘积项送入一个宏单元中的**或门**(各个宏单元内的或门构成固定的或阵列)。

异或门用于编程配置与或阵列输出信号的极性，若编程端(图中异或门悬空的引脚)编程为 1，则输出信号变反。对应于该引脚的内部电路该怎么利用也是需要考虑的问题。

触发器用来寄存或门的输出(实际器件中的触发器通常是可编程的，可以对类型、置/复位、时钟等进行编程)。

数据选择器用来编程选择输出是组合型输出(异或门直接输出)还是时序型输出(寄存器输出)。

输出三态缓冲器用来编程配置宏单元输出信号是否使能，若输出禁止，则对应的 I/O 引脚可作为外部输入引脚。输出三态缓冲器使外部引脚成

为可编程的输入/输出引脚。在实际器件中，如果引脚编程作为输入引脚，则对应于该引脚的内部电路该怎么利用也是需要考虑的问题。

图 6.4.2 中，**PLD 模块外部的互连线路**通过可编程开关可以被配置为不同的连接模式，它们决定了各 PLD 模块之间以及 PLD 模块与 I/O 模块间的连接，包括将 PLD 模块中各宏单元的输出反馈至其他 PLD 模块以及将外部的输入信号引入 PLD

模块。

CPLD 通常采用 EPROM、EEPROM 或闪存作为编程元件，因此具有非易失特性，一旦编程，信息就可长期保存，可用于电路规模不是很大，而且要求上电就立即执行功能的应用中。作为全局布线的可编程开关矩阵，通常具有传输延迟可预测的典型特性。

五、现场可编程门阵列 FPGA

与 SPLD、CPLD 的结构不同，FPGA 内部并不包含与或阵列，而是通过可配置的逻辑块实现逻辑功能，其一般结构如图 6.5.1 所示：

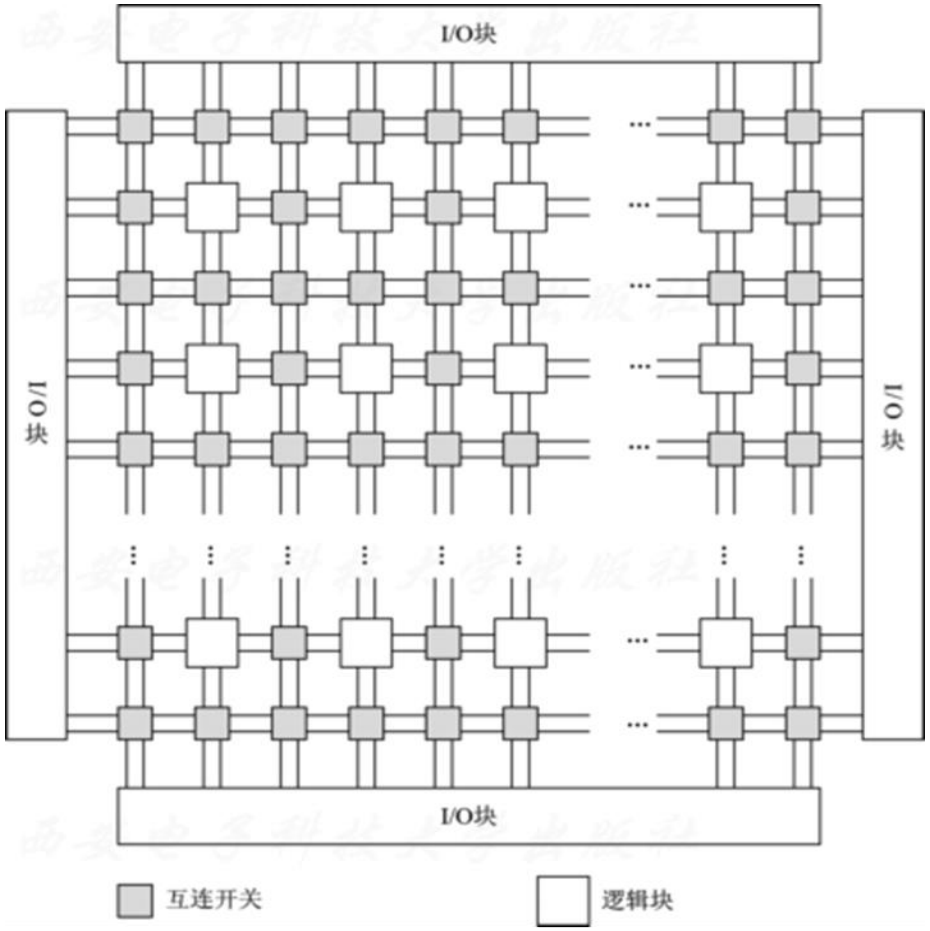


图6.5.1 FPGA的一般结构

主要包括三个组成部分：逻辑块、I/O 块和互连资源。

逻辑块编程配置实现用户要求的逻辑功能，**I/O 块**提供逻辑块到器件外部封装引脚的可编程接口，**互连资源**在各块之间传递信号。

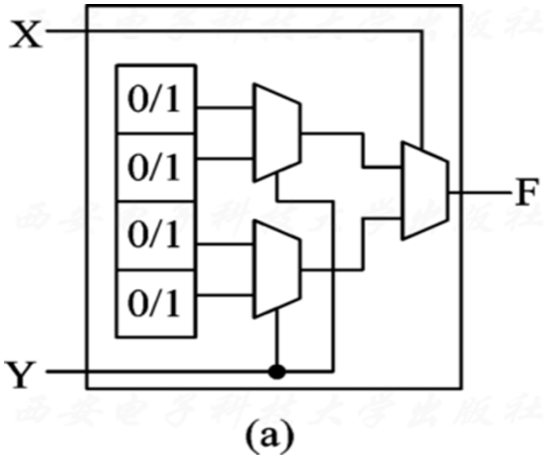
FPGA 的逻辑块通常只有少量的输入、输出。**不同的 FPGA 器件具有不同类型的逻辑块，最常用的一种逻辑块结构就是查找表 LUT。**

查找表实际上是一个**逻辑函数发生器**，其实现逻辑函数的机制类似于用 ROM 实现组合逻辑函

数。查找表内部有存储元，能够存储逻辑函数在不同输入情况下对应的输出信息，也即相当于存储了函数的真值表。输入变量作为地址信号，用不同取值选中不同的存储元，从而输出函数功能对应的函数值。改变存储元的存储内容，就能够实现不同的逻辑函数。

图 6.5.2(a)给出了一个 2 输入查找表的电路结构。

图中，每个存储元可存储 1 位“0”或“1”，四个存储元可存储任意一个 2 变量逻辑函数的真值表信息，输入变量 X、Y 用于控制三个数据选择器，最终选中不同的存储元输出。若将图 6.5.2(a)中的四个存储元从上到下依次配置为 0、1、1 和 0，则能够产生真值表如图 6.5.2(b)所示



(a)

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

(b)

图6.5.2 输入查找表

(a) 电路结构； (b) 实现函数

的逻辑函数。类似于图 6.5.2(a)的电路结构，也可以构造出 3 输入、4 输入的查找表。

除了查找表,为了能够实现时序逻辑,FPGA 的逻辑块中通常也配置有可编程的触发器。查找表的输出可配置为触发器旁路,形成组合逻辑;或配置为触发器输出，形成时序逻辑。

FPGA 的逻辑块通常组织为两维的行列阵列，相应的,互连资源也被组织为逻辑块行、列之间的

水平和垂直布线通道以及逻辑块和 I/O 块之间的布线通道，如图 6.5.1 所示。布线通道包括连线和可编程开关，可由软件根据电路的定时等要求自动配置，形成多种不同的、灵活的连接方式。

I/O 块提供逻辑块到 I/O 引脚的可编程接口。不同的器件有不同的 I/O 块结构，但一般都具有可编程的输入输出寄存器来寄存输入输出信号以及可编程的输出缓冲器，可通过编程控制实现输

入、输出或双向输入输出。

与大多数 CPLD 基于 EPROM 的编程不同，FPGA 的编程是基于 SRAM 静态存储元的。将配置数据编程写入器件内部 SRAM 存储元，即实现了电路的定制,存储的值决定了 FPGA 的逻辑功能和互连实现。

使用 SRAM 的优点是可以在线写入，可在器件驻留于系统时，现场重新配置来改变系统功能。这使得设计修改和更新变得非常容易，一片 FPGA 甚

至可以在不同的时候被重新动态配置以执行不同的功能。

FPGA 的可重配置能力能够简化硬件设计和调试过程，缩短产品投放市场的时间。但**基于 SRAM 编程的缺点则是其易失特性，每次上电时，都需要重新加载配置**。应用时，通常要在电路板上增加一块可编程只读存储器 PROM 芯片，保存应用于 FPGA 的配置信息，上电时写入 SRAM。同时，FPGA 器件

内部信号间灵活多样的连接方式使FPGA信号的传递延迟具有不确定性。

六、CPLD 和 FPGA 的编程*

（一）在系统可编程技术

1、在系统可编程的概念

在系统可编程技术简称为 ISP 技术，是指对用户设计的目标电路或印刷电路板的逻辑和功能的可重新配置能力。这种重新配置，或称为重构，可以在产品设计、制造过程中的任一环节进行，甚

至可以在产品交付用户之后再进行。支持 ISP 的可编程逻辑器件，称为“在系统可编程逻辑器件”。

ISP 技术解决 PLD 的编程问题，消除了传统 PLD 编程的局限性，具体表现在：

(1)在板或在系统级的编程。传统的电路板设计，需要先对 PLD 编程，然后再将 PLD 装配到电路板中，而 ISP 技术则允许直接将未编程的 PLD 装配到电路板中，然后根据电路或系统的功能要求，

进行在板或在系统的编程。

(2)简单方便的修改和升级。对电路板上的ISP 器件，可以像对待其他器件一样对待，不需要额外的生产流程，不需要移动器件，可以在系统不掉电的情况下，使用标准逻辑电平信号实现器件的编程或重新配置。这使得硬件可以像软件一样简单方便的修改，系统升级也变得非常容易。

2、在系统可编程逻辑器件的优势

与传统的可编程逻辑器件相比，使用在系统可编程逻辑器件，在系统设计、制造、编程、缩短产品投放市场时间、降低成本等方面都有着明显的优势。

(1)允许采用原型板设计简化设计流程。大多数系统的设计中，像微处理器、RAM 等电路板主要构建块，在系统逻辑设计之前就已决定。使用 ISP

器件，完全可以采用带有这些主要构建块的原型板进行系统设计，由 ISP 器件完成各功能块的互连，不需要更换电路板上的模块或改变电路板的布线，只需要下载信息到板上的 ISP 器件中，就能实现新的设计。

(2)可用于板级调试和系统测试。在系统设计期间，一旦 ISP 器件能够在电路板上稳定工作，就可用来调试电路板的其他部分。例如，可以重新配

置 ISP 器件中，使它能够将激励信号强制送到电路板的其他部分，以调试这部分电路板的工作情况。而当设计完成后，ISP 器件又可用来测试系统整体功能。例如，将临时的诊断测试模式写入 ISP 器件，来验证各个系统功能。

(3)**实现多功能硬件**。所谓多功能硬件，就是通过在系统编程，使同一个硬件设计具有不同的系统级功能。例如，一个需要支持两种不同总线接

口的电路板设计，传统的解决方法是为每种总线接口设计一套单独的电路，或是在一种设计的基础上增加一些额外的电路，而使用 ISP 器件则可以采用通用的总线接口设计，通过在系统配置来支持不同的接口标准。ISP 器件的这一特性能够减少电路板使用的模块数量，从而降低生产成本。

(4)简化生产流程并减少损伤。在生产时使用非 ISP 器件的电路系统，需要先对各类不同的 PLD

进行单独编程，然后贴上标签入库保存，在装配时，又要从库房领出器件安装在电路板上，而使用 ISP 器件时，可以直接将未编程的器件安装在电路板上，然后进行在板的编程、测试、定型，从而简化了生产流程。随着 PLD 器件集成度的提高，器件引脚也迅速增加。将引脚多而密的 PLD 器件安装在编程器插座上，本身就是一件有难度的工作，一旦弄弯或折断引脚，就会造成永久性的损伤，而使用

ISP 器件则能够减少这种损伤，降低生产成本。

(5)便于系统升级和维护。传统技术使得产品一旦交付用户，就难以升级，维护困难且费用昂贵，而 ISP 器件的在系统可重配置能力则使这一过程简单易行。

3、ISP 器件编程

ISP 技术最初由 Lattice 公司在 20 世纪 80 年代末提出，并推出了一系列 ISP 可编程逻辑器件。由于 ISP 器件在产品开发、调试、现场升级、简化生产流程、降低成本等方面的优势，因而迅速获得了广泛的应用。在 1990 年，大约只有 8%的系统设计人员承认 ISP 会影响他们对高密度 PLD 的选择，但到了今天，ISP 已成为 PLD 的必需要求。

ISP 器件编程有三种方式，分别是：ISP 编程方式、JTAG 编程方式和 IEEE 1532 编程方式。

1）ISP 编程方式

ISP 编程方式是 Lattice 公司的 ISP 器件独有的编程方式，使用“ISP 状态机”实现编程，支持的平台包括 PC 机、工作站、ATE(Automated Test Equipment)或其他通用编程器。其中，PC 机是应用最普遍的，编程时，需要在 PC 机上安装 ISP 编

程软件，并用 ISP 编程电缆连接 PC 机并行口和 ISP 器件的 ISP 接口，此时即可对一个或多个 ISP 器件实现快速、简单、便宜的编程。

2）JTAG 编程方式

JTAG 编程方式采用 IEEE 1149.1 标准的“边界扫描测试存取口” TAP(Boundary Scan Test Access Port)和“TAP 状态机”实现编程。

IEEE 1149.1 标准最初针对板级测试问题而提出，但相同的边界扫描串行路径和控制信号也可被用于器件编程，从而将 ISP 融入到板级测试系统的体系中，能够使用同一个接口实现测试及系统编程。

3）IEEE 1532 编程方式

大多数集成电路生产厂商多年来一直遵循着 IEEE 1149.1 规范，也即支持符合 IEEE1149.1 规

范的 PLD 板级测试，并可通过边界扫描测试系统进行系统编程，但不同厂商、不同类型芯片的编程算法和数据格式一直没有统一的标准。为解决这一问题，约在 2000 年，JTAG 对 1149.1 标准进行了增强，目标是标准化芯片模块级的编程算法，并定义描述编程算法和相关数据格式的软件需求，从而形成了 IEEE 1532 标准。使用 IEEE 1532 标准，用户可以将不同厂商的器件连在一起，使用相

同的编程软件进行编程，而不需要了解特定厂商编程方面的知识。

（二）JTAG 边界扫描测试技术

1、边界扫描测试的基本原理

JTAG 边界扫描测试 BST 技术将测试电路设置在集成电路芯片内部，通过内置的测试电路，可以

完成**三类测试**：测试集成电路芯片的内部功能；测试不同集成电路芯片间的连线故障；在电路正常工作模式下，观察或修改电路的工作数据。

边界扫描测试的**基本工作原理**就是将**边界扫描单元 BSC** 配置到集成电路芯片的所有引脚，串行连接构成边界扫描通道。**边界扫描通道按照移位寄存器的方式工作**，通过给集成电路引脚施加测试信号，并观测其响应，从而实现对组装在一块

电路板上的多个集成电路芯片进行测试与诊断。系统正常工作时，边界扫描电路不会影响系统的正常操作。

2、JTAG 的基本结构

符合 JTAG 标准的集成电路芯片结构如图 6.6.1 所示。

1）测试存取口 TAP

为支持边界扫描测试，集成电路芯片需要增设 TDI、TDO、TMS 和 TCK 四个引脚，可以选择设置$\overline{TRST}$引脚，这些引脚构成了 JTAG 标准的测试存取口 TAP。

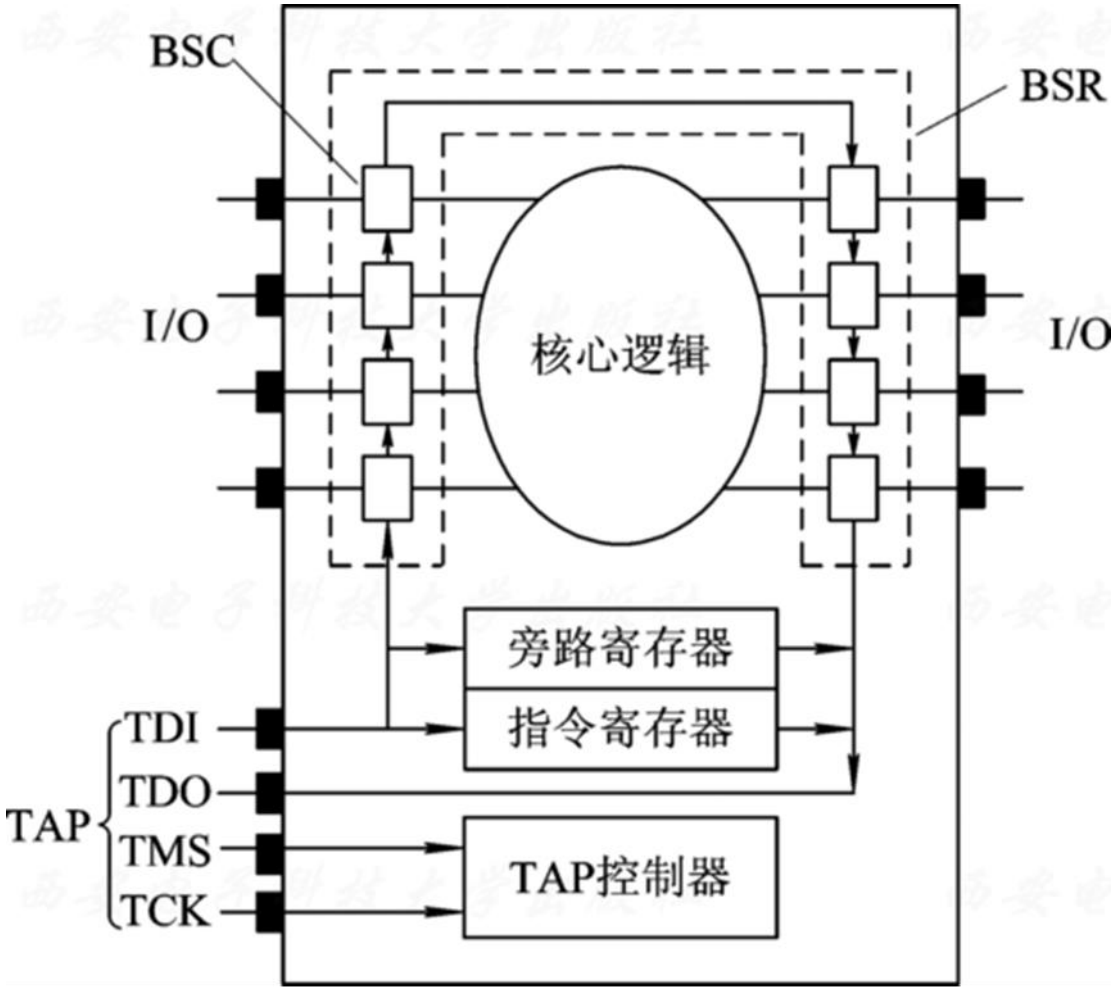


图6.6.1 支持IEEE 1149.1标准的芯片结构

各引脚功能如下：

•TDI：测试数据或指令输入引脚，是扫描数据或指令的串行移入端。

•TDO：测试数据或指令输出引脚，是扫描数据或指令的串行移出端。

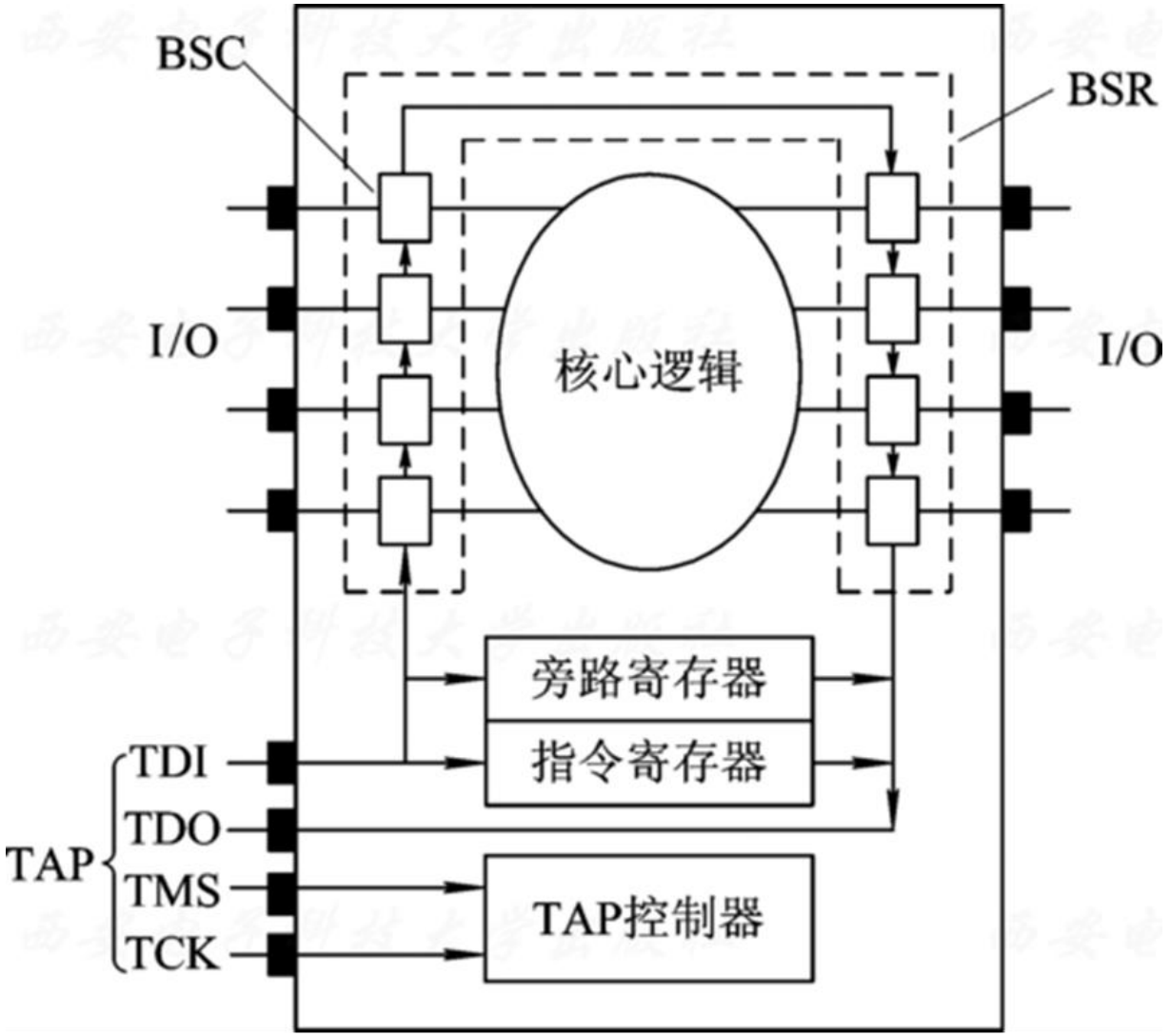


图6.6.1 支持IEEE 1149.1标准的芯片结构

•TCK：测试时钟。

•TMS：测试模式选择，决定扫描通道的工作模式。

•$\overline{TRST}$：测试电路复位，低电平有效，可选。

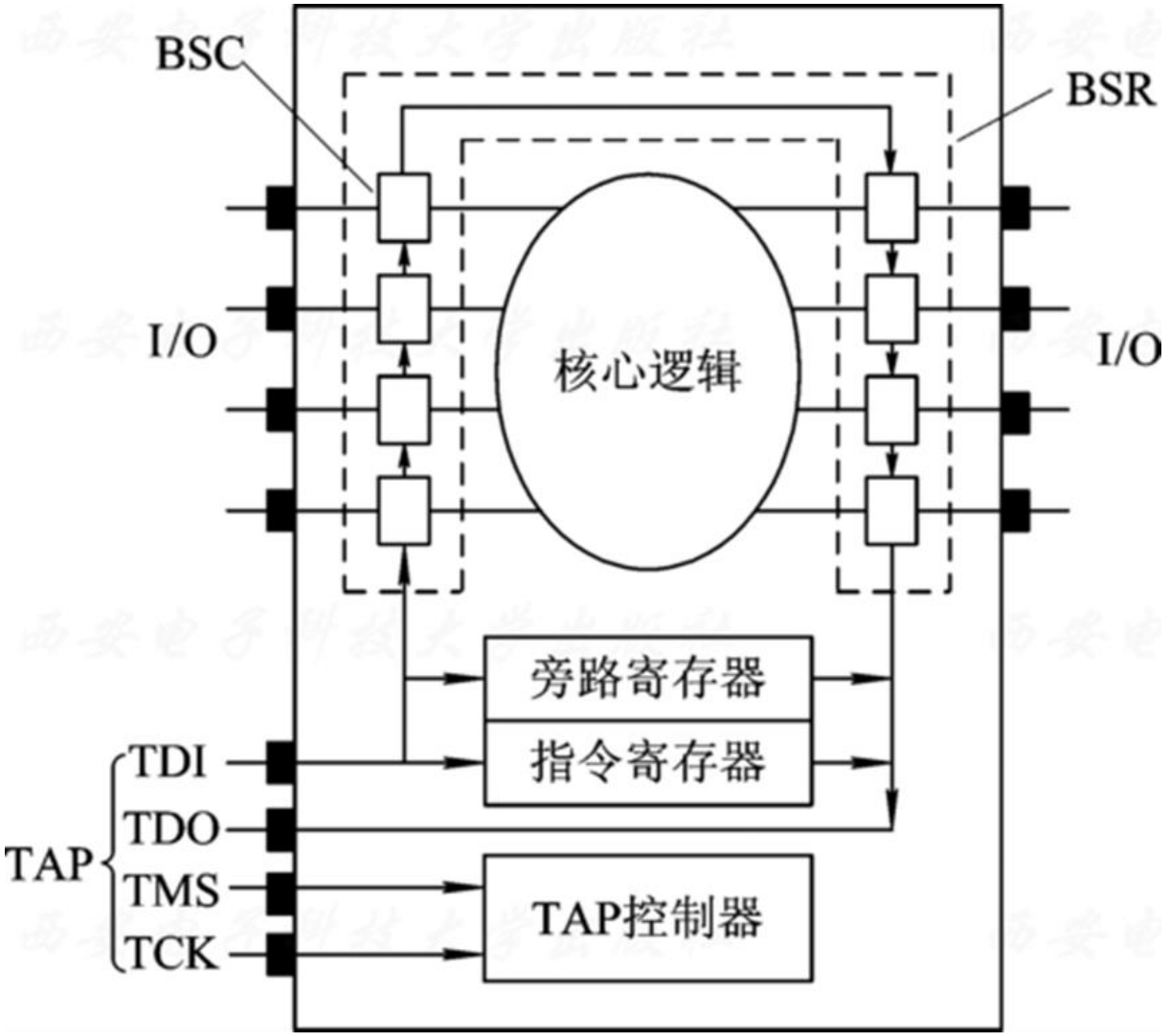


图6.6.1 支持IEEE 1149.1标准的芯片结构

2）边界扫描寄存器 BSR

边界扫描寄存器 BSR 是由芯片内所有边界扫描单元 BSC 组成的一个**串行数据寄存器**，是芯片内的串行数据扫描通道，始端与 TDI 相连，末端与 TDO 相连。

边界扫描单元 BSC 的结构如图 6.6.2 所示，由两个数据选择器 MUX1、MUX2 和两个触发器组成，相同结构的 BSC 既可用于器件的输入引脚，也可

用于器件的输出引脚。

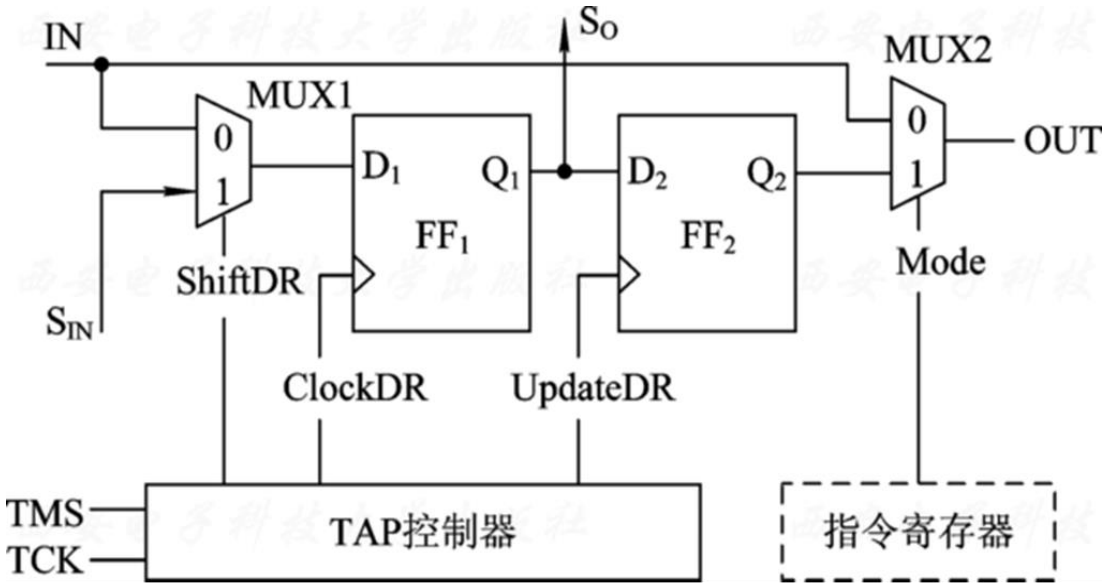


图6.6.2 边界扫描单元BSC的结构

BSC 有**四种工作模式**：

(1)正常模式。芯片处于正常工作状态时，扫描通道不影响芯片的工作，由 Mode 信号控制 MUX2 选择 IN 信号输出。若 BSC 用于芯片的输入引脚，则 IN 信号为芯片的外部引脚输入，OUT 信号为芯片的内部核心逻辑输

入；若 BSC 用于芯片的输出引脚，则 IN 信号为芯片的内部核心逻辑输出，OUT 信号为芯片的外部引脚输出。此时，IN 和 OUT 应该是相同的。

(2)扫描模式。所有 BSC 组成串联的扫描通道，串行移位测试数据，由 ShiftDR 信号控制 MUX1 选择串行信号 S_{IN} 输入，ClockDR 持续运作逐个锁存各个串行位，然后移出。BSC 的输入为 S_{IN}，可来自于前一个BSC的S_0，或外部输入TDI(第一个BSC)；

输出为 S_0，作为下一个 BSC 的 S_{IN}，或外部输出 TDO(最末一个 BSC)。

(3)捕获模式。若要用 BSC 输出测试结果，则需要在 ClockDR 信号的作用下，将 IN 信号暂存于 FF_1，然后适时将它输送出去。

(4)更新模式。在 UpdateDR 信号的作用下，可以将暂存于 FF_1 中的数据更新到 FF_2，然后由 Mode 信号选择 FF_2，从 OUT 输出。比如要将测试样

本施加到特定引脚，需要先用扫描通道将数据串行移位到相应位置，然后捕获至 FF_1，再将 FF_1 的数据更新到 FF_2，最后由 Mode 信号选择 FF_2 的信息输出。

3）指令寄存器 IR

指令寄存器对扫描指令进行译码，确定边界扫描测试的工作方式。 边界扫描寄存器 BSR 中的测试数据和指令寄存器 IR 中的 JTAG 指令都通过

TDI 引脚串行移入，TAP 控制器可选择对哪个寄存器进行移位扫描。

IEEE 1149.1 标准定义了多条指令，并允许器件厂商定义特定设计需求的指令来扩展测试逻辑的功能。这里只对执行内测试的 INTEST 和执行外测试的 EXTEST 两条指令进行简单介绍。

(1) INTEST 指令。INTEST 指令执行内部测试。内部测试时，测试向量从 TDI 移入，通过扫描通道

移位施加到芯片核心逻辑的内部输入端，芯片核心逻辑的输出，也即响应向量， 被捕获至芯片输出引脚的 BSC，并从 TDO 移出。将移出的响应向量与预期输出进行比较，即可分析检测集成芯片内部的工作情况。

(2)EXTEST 指令。执行 EXTEST 指令可以进行外部测试。外部测试时，电路板上不同芯片的 BSR

相互串联构成一个更大的扫描通道，图6.6.3给出了两个芯片的外测试连接方式。测试向量通过芯片1的扫描通道移位施加到其输出引脚，由于芯片2的输入引脚与芯片1的输出引脚对应相连，因此芯片2的输入引脚状态即成为响应向量，捕获该响

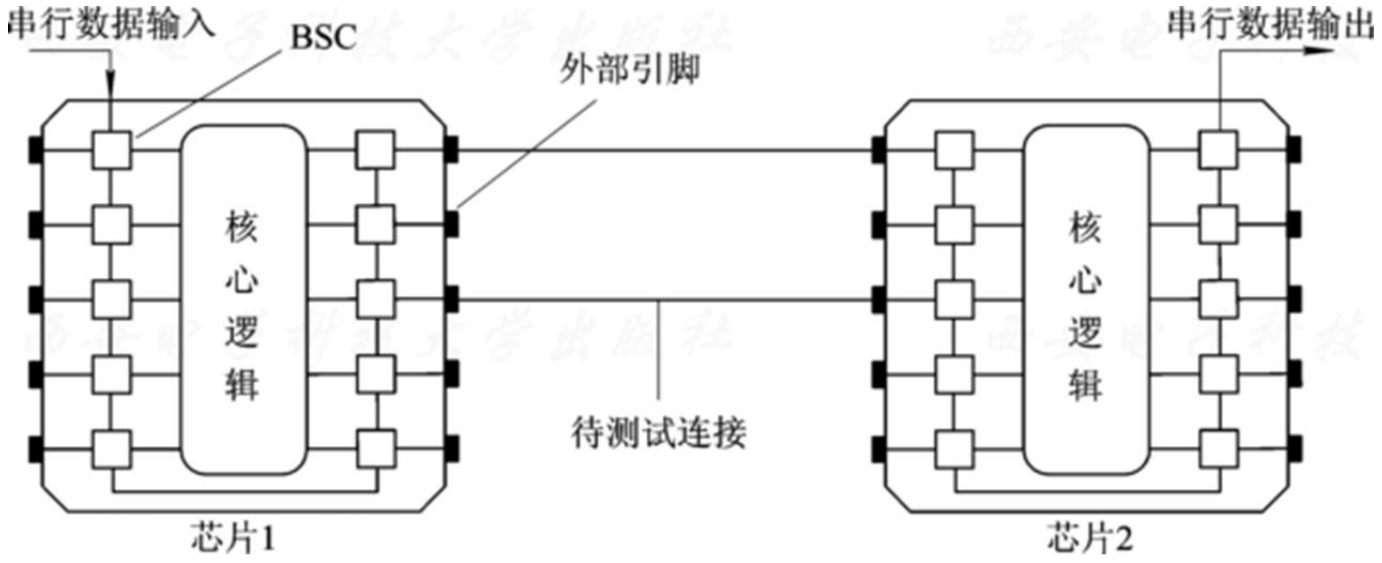


图6.6.3 JTAG测试

应向量并从芯片 2 的 TDO 移出，与预期值进行比较即可测试集成电路芯片间的连线故障。

4）旁路寄存器和 TAP 控制器

旁路寄存器是一个移位寄存器，提供芯片内 TDI 到 TDO 的最短通路。当多个芯片的扫描通道相互串联构成更大的扫描通道时，若某个芯片的数据寄存器不参与扫描，可由旁路寄存器旁路掉，以减少不必要的扫描时间。

TAP 控制器是一个有 16 种状态的状态机，用来决定芯片扫描电路所要进行的动作。

TAP 控制器的一个特殊设计是，当 TMS 输入连续四个以上的逻辑 0 时，TAP 控制器就会处于闲置状态。这样的设计是为了不论 TAP 控制器处于哪一个状态，连续输入四个以上的逻辑 0，TAP 控制器都会回到闲置状态，从而不影响系统的正常工作。

3、JTAG 编程

JTAG 的边界扫描测试体系结构能够在芯片正常工作的情况下，测试芯片引脚间的连通性，或者是芯片内部的工作情况。同时，使用 JTAG 电路，将编程数据移入可编程逻辑器件，也能够实现可编程逻辑器件的在系统编程。

基于 JTAG 接口的编程是目前可编程逻辑器件普遍支持的编程方式。JTAG 编程使用测试存取口

TAP 所定义的 TDI、TDO、TMS 和 TCK 四个引脚完成。TDI 用来将编程数据移入可编程逻辑器件;TDO 用于将编程数据移出器件，主要用在多个器件构成 JTAG 编程链时,可将编程数据移到下一个器件;TMS 选择 TAP 控制器的工作状态；TCK 提供编程时钟。

使用 PC 机进行编程时，需要用下载电缆连接电路板的 JTAG 接口和 PC 机的并/串行口。PC 机通过安装的 **EDA 软件**生成可下载的**编程文件**，然后将编程文件下载至电路板上的 PLD 器件内，从而实现编程。多个 PLD 器件也可以构成一个**编程链**，通过同一个 JTAG 接口实现编程。图 6.6.4

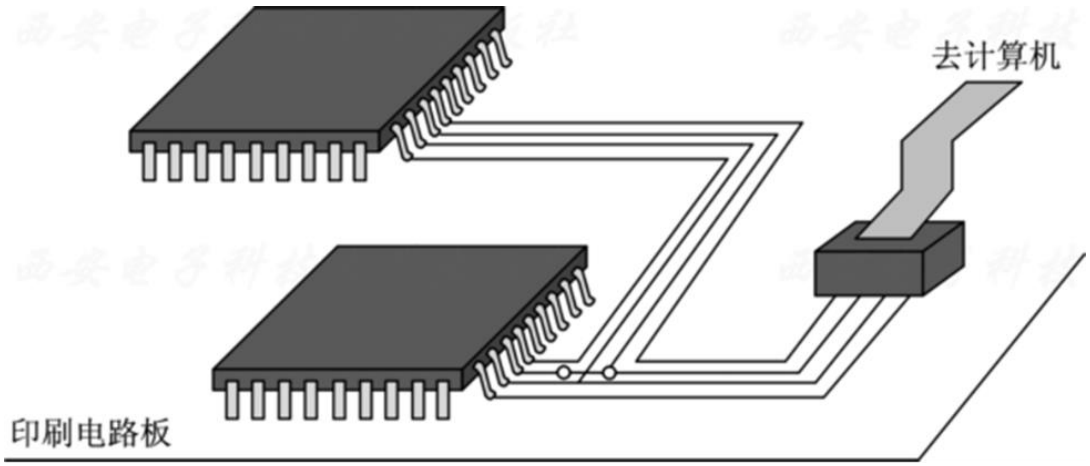


图6.6.4 JTAG编程连接示意图

给出了使用一个 JTAG 接口编程两个 PLD 器件的示意图。

值得注意的是，对于 FPGA 器件，其配置数据存储于 SRAM 单元中，由于 SRAM 的易失特性，因此必须在每次上电时都重新装载配置数据。配置数据写入 FPGA 的方式称为 FPGA 的**配置模式**。在实验环境或电路板的设计和调试阶段，通常都可以使用 PC 机，通过 FPGA 器件的 JTAG 接口对器件

进行在系统的编程配置。而在设计完成之后的现场环境，则必须为 FPGA 器件配置一个非易失的配置存储器。

用户设计并经过开发系统编译后产生的数据配置文件，必须事先写入配置存储器，才能在系统上电后写入 FPGA 的存储元。不同厂家、不同系列芯片支持的配置模式也有所不同，但总体上可分为被动模式和主动模式两大类。

被动模式由配置存储器和 FPGA 之外的器件提供控制信号，控制配置过程。被动模式常用于带有智能主机(如 PC 机、嵌入式微处理器等)的系统中，由智能主机控制配置过程，从存储设备(如闪存、硬盘、RAM 等)读出配置数据并写入 FPGA。

主动模式由目标 FPGA 控制配置过程，系统上电后，FPGA 产生同步信号，主动从配置存储器中读取配置数据。配合不同 FPGA 芯片的主动配置需

求，各厂家也发布有相应的配置存储器件。配置存储器件如果支持 JTAG 接口，则可以通过 JTAG 接口进行在系统编程，否则就需要采用其他的方法，比如在安装于系统板之前使用专门的编程器进行预编程等。