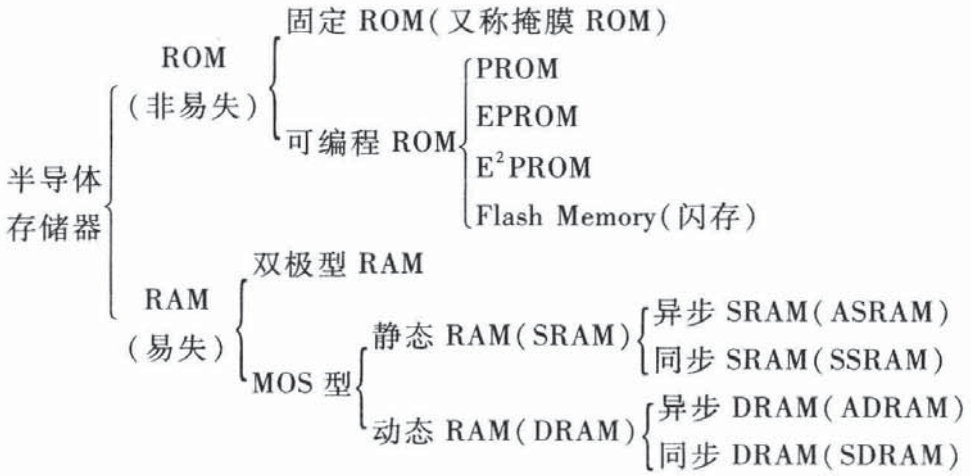
7　半导体存储器

一、内容提要及重点

本章的内容提要：

1. 半导体存储器的分类

（1）按数据易失性与非易失性分为两大类：



易失性存储器断电后，其所存的数据将全部丢失，而非易失性存储器断电后数据仍可长久保存。

SRAM 用触发器记忆数据，DRAM 靠 MOS 管栅极电容存储数据。因此，在不停电的情况下，SRAM 的数据可以长久保持，而 DRAM 则必须定期刷新。

无论是 SRAM 还是 DRAM，目前都有在时钟脉冲作用下工作的同步 RAM（SSRAM 和 SDRAM），且已成为主流存储器。在此基础上发展起来的 DDR、DDR Ⅱ 和 QDR 等 RAM 也已愈来愈多地应用于计算机内存、显存和通讯设备中。

（2）按正常工作时读写特性分为：

(3) 按输入输出方式不同还可分为并行存储器和串行存储器。

2. 存储器的一般构成

半导体存储器一般由三部分构成:存储阵列、地址译码器和输入输出控制电路见图 7.1 所示。

3. 存储器的容量和存取时间是反映存储器性能好坏的重要技术指标。存储器的容量一般是用存储的**字数**和每个字所含**位数**的乘积表示。存储器的容量越大,意味着能存储的数据越多。存取时间是反映存储器工作速度快慢的指标。

4. 存储器的读写定时关系非常重要,各种信号必须满足规定的时序要求,才能完成数据的正确读写。相同类型的存储器,其读写控制大致相同,只是具体定时参数有所差异,使用时须参考相应的数据手册。

5. 当只用单个芯片不能满足存储容量要求时,可通过简单方法进行容量扩展。扩展既可以是位数(字长)的扩展,又可以是字数的扩展,也可以是两者同时扩展。

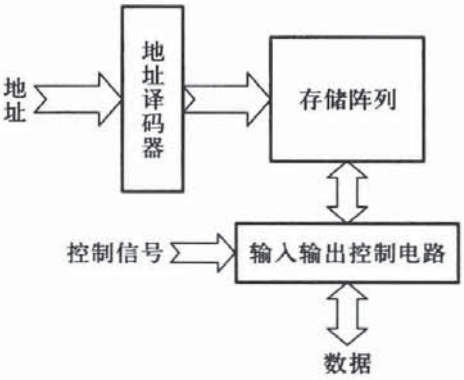


图 7.1 存储器的基本结构框图

本章的重点:

各种半导体存储器的基本结构、工作原理和使用方法;存储器容量的扩展。

二、典型例题解析

例 7.1 指出容量为 64K×4 的存储器有多少个存储单元,其地址码至少需要几位?数据位是几位?

解题思路:求解本题时,只要弄清以下几个关系就能很容易得到结果:

存储单元数 = 字数×位数

地址码的位数 n 与字数 N 的关系为:$N=2^n$

数据位数 = 位数

解:存储单元 = 64K×4 = 256K 个(注:1K = 1024);

因为,64K $=2^{16}$,即 $n=16$,所以地址码位数为 16;

数据位数等于位数,此处为 4 位。

所以,64K×4 的存储器系统具有 256K 个存储单元,至少需要 16 位地址码,数据位是 4 位。

例 7.2 试用一个具有片选使能$\overline{CE}$、输出使能$\overline{OE}$、读写控制$\overline{WE}$、容量为 4K×4 位的 SRAM 芯片和必要的逻辑门,扩展成为 8K×8 位的存储器系统,试画出其逻辑图。

解题思路:字数和位数同时扩展是扩展存储容量的常用方法,通常先进行位数扩展,然后用扩展的高位地址进行字数的扩展。

用并联方式实现位扩展,即将存储器的地址线、读/写控制线和片选信号对应地并联在一起,而各个芯片的数据输入/输出端作为字的各个位线。

通常用外加译码电路控制存储器芯片的片选使能输入端，并将对应的数据线并接，来实现字数的扩展。

解：采用 4K×4 位的 SRAM 构成 8K×8 位的存储器系统，必须同时进行字扩展和位扩展。

首先进行位扩展：用 2 片 4K×4 位的芯片，并接它们的地址线、读/写控制线和片选信号，扩展成 4K×8 位系统。用同样的方法再扩展一个 4K×8 位系统，然后进行字扩展。将位扩展得到的两个 4K×8 位系统，通过片选信号组合起来实现 8K×8 位的存储器系统，此时需增加 1 根地址线 A_{12}，通过一个**非**门控制两组 4K×8 位系统的片选使能 $\overline{CE}$，扩展后的存储系统如图 7.2 所示。其中(1)和(2)、(3)和(4)分别构成两个 4K×8 位存储器系统；**非**门将 A_{12} 反相，并将 A_{12} 和 $\overline{A}_{12}$ 分别连接到两组 4K×8 的片选使能端 $\overline{CE}$，实现字扩展。系统共用了 4 片 4K×4 位芯片，一个**非**门。

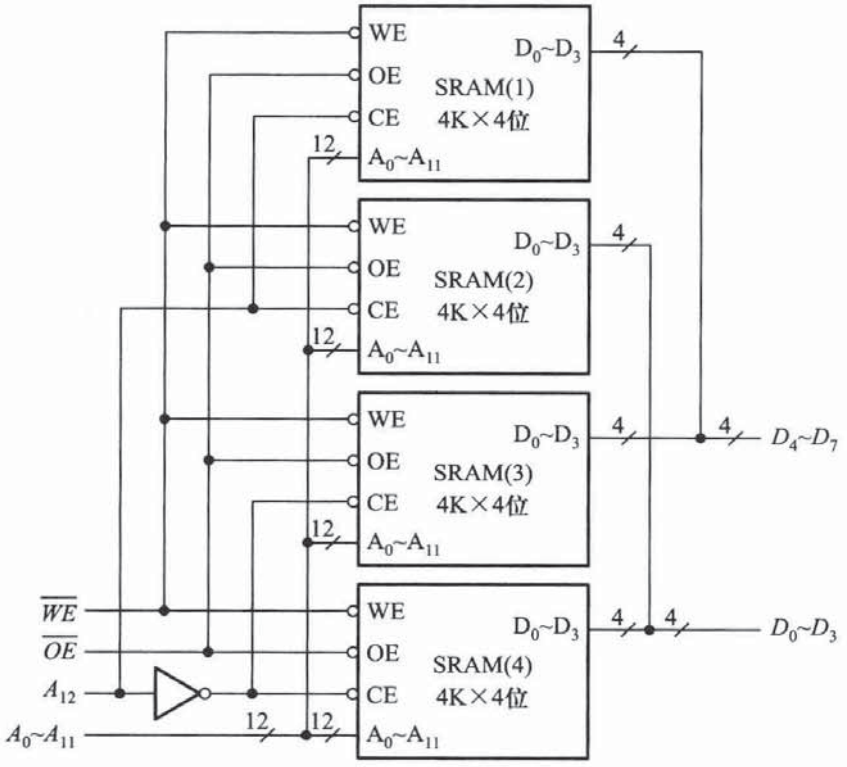


图 7.2 用 4K×4 位 SRAM 芯片构成 8K×8 位的存储器系统

三、习题全解

7.1 只读存储器

7.1.1 指出下列存储系统各具有多少个存储单元，至少需要几根地址线和数据线。

(1) 64K×1 (2) 256K×4 (3) 1M×1 (4) 128K×8

解：(1) 存储单元 = 64K×1 = 64K 个(注：1K = 1024 = 2^{10})；

因为，64K = 64×1024 = 2^{16}，即 n = 16，所以地址线为 16 根；

数据线根数等于位数，此处为 1 根。

同理得：

（2）$4\times256\times1024=2^{20}=1M$ 个存储单元；地址单元 $256K=2^8\times2^{10}=2^{18}$，即 18 根地址线；数据线为 4 根

（3）存储单元 $=1M\times1=1M$，地址单元 $1M=2^{20}$，即 20 根地址线；1 根数据线

（4）存储单元 $=128K\times8=1M$，地址单元 $128K=2^{17}$，即 17 根地址线；数据有 8 位，需要 8 根数据线

7.1.2 设存储器的起始地址为全 0，试指出下列存储系统的最高地址为多少？

（1）2K×1　　（2）16K×4　　（3）256K×32

解：因为存储系统的最高地址 = 字数 + 起始地址 −1，（1）、（2）、（3）题给出的字数分别为 2K、16K 和 256K，而起始地址均为 0。所以它们的十六进制地址码是：

（1）最高地址 $=2K+0-1=7FFH$　　同理：　　（2）3FFFH　　（3）3FFFFH

7.1.3 试确定用 ROM 实现下列逻辑函数时所需的容量：（1）实现两个 3 位二进制数相乘的乘法器；（2）将 8 位二进制数转换成十进制数（用 BCD 码表示）的转换电路。

解：用 ROM 实现逻辑函数时，逻辑函数的输入变量由 ROM 地址线输入，逻辑函数值由 ROM 数据线输出。

（1）两个 3 位二进制数相乘，共有 6 位输入，即需要 6 根地址线；而两个 3 位二进制数相乘的最大值是 49，即 **111×111**=110001，共需要 6 位输出，所以 ROM 的容量应为 $2^6\times6$ 位。

（2）8 位二进制数的最大值为 **11111111**，转换成十进制数表示为 255，用 BCD 码表示为 **1001010101**，即输入 8 位，输出 10 位，所以 ROM 的容量应为 $2^8\times10$ 位。

7.1.4 用一片 128×8 位的 ROM 实现各种码制之间的转换。要求用从全 **0** 地址开始的前 16 个地址单元实现 8421BCD 码到余 3 码的转换；接下来的 16 个地址单元实现余 3 码到 8421BCD 码的转换。试求：（1）列出 ROM 的地址与内容对应关系的真值表；（2）确定输入变量和输出变量与 ROM 地址线和数据线的对应关系；（3）简要说明将 8421BCD 码的 0101 转换成余 3 码和将余 3 码的 1001 转换成 8421BCD 码的过程。

解：（1）设 128×8 位 ROM 的 7 根地址线分别为 $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$，8 根数据线为 $D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$。将输入码作为地址码，数据输出作为转换后的输出码，根据 8421BCD 码和余 3 码的关系，实现码制转换的 ROM 地址与内容对应关系的真值表如表题解 7.1.4 所示。

表题解 7.1.4

地址（8421BCD 码输入）	内容（余 3 码输出）	地址（余 3 码输入）	内容（8421BCD 码输出）
$A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	$A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$
0 0 0 0 0 0 0	0 0 0 0 0 0 1 1	0 0 1 0 0 0 0	× × × × × × × ×
0 0 0 0 0 0 1	0 0 0 0 0 1 0 0	0 0 1 0 0 0 1	× × × × × × × ×
0 0 0 0 0 1 0	0 0 0 0 0 1 0 1	0 0 1 0 0 1 0	× × × × × × × ×
0 0 0 0 0 1 1	0 0 0 0 0 1 1 0	0 0 1 0 0 1 1	0 0 0 0 0 0 0 0
0 0 0 0 1 0 0	0 0 0 0 0 1 1 1	0 0 1 0 1 0 0	0 0 0 0 0 0 0 1
0 0 0 0 1 0 1	0 0 0 0 1 0 0 0	0 0 1 0 1 0 1	0 0 0 0 0 0 1 0

续表

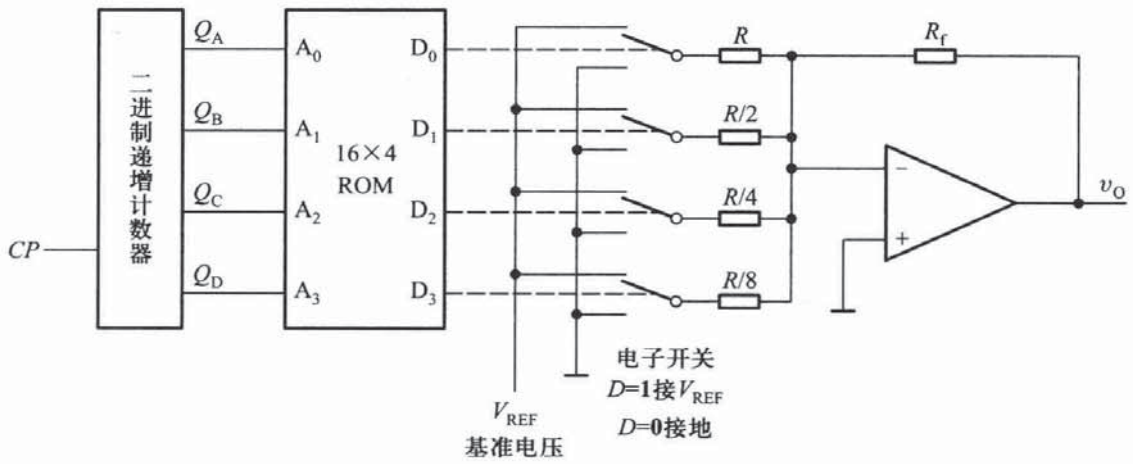
地址 （8421BCD 码输入）	内容 （余 3 码输出）	地址 （余 3 码输入）	内容 （8421BCD 码输出）
$A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	$A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$	$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$
0 0 0 0 1 1 0	0 0 0 0 1 0 0 1	0 0 1 0 1 1 0	0 0 0 0 0 0 1 1
0 0 0 0 1 1 1	0 0 0 0 1 0 1 0	0 0 1 0 1 1 1	0 0 0 0 0 1 0 0
0 0 0 1 0 0 0	0 0 0 0 1 0 1 1	0 0 1 1 0 0 0	0 0 0 0 0 1 0 1
0 0 0 1 0 0 1	0 0 0 0 1 1 0 0	0 0 1 1 0 0 1	0 0 0 0 0 1 1 0
0 0 0 1 0 1 0	× × × × × × × ×	0 0 1 1 0 1 0	0 0 0 0 0 1 1 1
0 0 0 1 0 1 1	× × × × × × × ×	0 0 1 1 0 1 1	0 0 0 0 1 0 0 0
0 0 0 1 1 0 0	× × × × × × × ×	0 0 1 1 1 0 0	0 0 0 0 1 0 0 1
0 0 0 1 1 0 1	× × × × × × × ×	0 0 1 1 1 0 1	× × × × × × × ×
0 0 0 1 1 1 0	× × × × × × × ×	0 0 1 1 1 1 0	× × × × × × × ×
0 0 0 1 1 1 1	× × × × × × × ×	0 0 1 1 1 1 1	× × × × × × × ×

（2）根据表题解 7.1.4 的关系可知，输入变量对应地址线的 $A_3\ A_2\ A_1\ A_0$，输出变量对应数据线的 $D_3\ D_2\ D_1\ D_0$（也可用 $D_7\ D_6\ D_5\ D_4$，但表中的内容也要做相应调整）。A_4作为转换控制位。当 $A_4=\mathbf{0}$ 时，实现 8421BCD 码到余 3 码的转换；当 $A_4=\mathbf{1}$ 时，实现余 3 码到 8421BCD 码的转换；

（3）要将 8421BCD 码的 **0101** 转换成余 3 码时，输入地址码应为 $A_6A_5A_4A_3A_2A_1A_0=\mathbf{0000101}$，此时 $D_3\ D_2\ D_1\ D_0$ 的数据输出 **1000** 即为余 3 码；

而将余 3 码的 **1001** 转换成 8421BCD 码时，输入地址码应为 $A_6A_5A_4A_3A_2A_1A_0=\mathbf{0011001}$，此时 $D_3\ D_2\ D_1\ D_0$ 的数据输出 **0110** 即为 8421BCD 码。注意，此时 A_4 必须为 **1**。

7.1.5 利用 ROM 构成的任意波形发生器如图题 7.1.5 所示，改变 ROM 的内容，即可改变输出波形。当 ROM 的内容如表题 7.1.5 所示时，画出输出端随 CP 变化的波形。



图题 7.1.5

表题 7.1.5

$A_3A_2A_1A_0$	$D_3D_2D_1D_0$	$A_3A_2A_1A_0$	$D_3D_2D_1D_0$
0 0 0 0	0 1 0 0	1 0 0 0	0 1 0 0
0 0 0 1	0 1 0 1	1 0 0 1	0 0 1 1
0 0 1 0	0 1 1 0	1 0 1 0	0 0 1 0
0 0 1 1	0 1 1 1	1 0 1 1	0 0 0 1
0 1 0 0	1 0 0 0	1 1 0 0	0 0 0 0
0 1 0 1	0 1 1 1	1 1 0 1	0 0 0 1
0 1 1 0	0 1 1 0	1 1 1 0	0 0 1 0
0 1 1 1	0 1 0 1	1 1 1 1	0 0 1 1

解:由图看出,二进制递增计数器在 CP 作用下不断生成 ROM 的地址码。ROM 的数据输出控制电子开关,当 ROM 的数据输出为 **1** 时,电子开关接 V_{REF};ROM 的输出为 **0** 时,电子开关接地。运放和相关电阻构成反相比例加法电路,其输出电压与 $D_3 \sim D_0$ 的关系为

$$v_o = -R_f\left(\frac{D_0V_{REF}}{R}+\frac{2D_1V_{REF}}{R}+\frac{4D_2V_{REF}}{R}+\frac{8D_3V_{REF}}{R}\right)$$

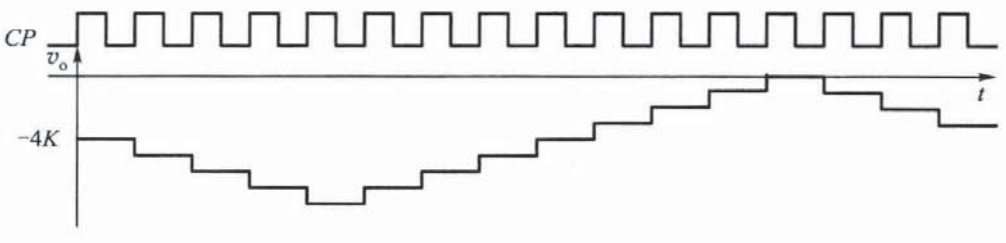
$$= -\frac{V_{REF}R_f}{R}(D_0+2D_1+4D_2+8D_3)$$

设 $K=\dfrac{V_{REF}R_f}{R}$,则 v_o 与 $D_3 \sim D_0$ 的对应关系如表题解 7.1.5 所示。

v_o 一个周期的输出波形如图题解 7.1.5 所示。

表题解 7.1.5

$A_3\ A_2\ A_1\ A_0$	$D_3\ D_2\ D_1\ D_0$	v_o	$A_3\ A_2\ A_1\ A_0$	$D_3\ D_2\ D_1\ D_0$	v_o
0 0 0 0	0 1 0 0	$-4K$	1 0 0 0	0 1 0 0	$-4K$
0 0 0 1	0 1 0 1	$-5K$	1 0 0 1	0 0 1 1	$-3K$
0 0 1 0	0 1 1 0	$-6K$	1 0 1 0	0 0 1 0	$-2K$
0 0 1 1	0 1 1 1	$-7K$	1 0 1 1	0 0 0 1	$-1K$
0 1 0 0	1 0 0 0	$-8K$	1 1 0 0	0 0 0 0	0
0 1 0 1	0 1 1 1	$-7K$	1 1 0 1	0 0 0 1	$-1K$
0 1 1 0	0 1 1 0	$-6K$	1 1 1 0	0 0 1 0	$-2K$
0 1 1 1	0 1 0 1	$-5K$	1 1 1 1	0 0 1 1	$-3K$

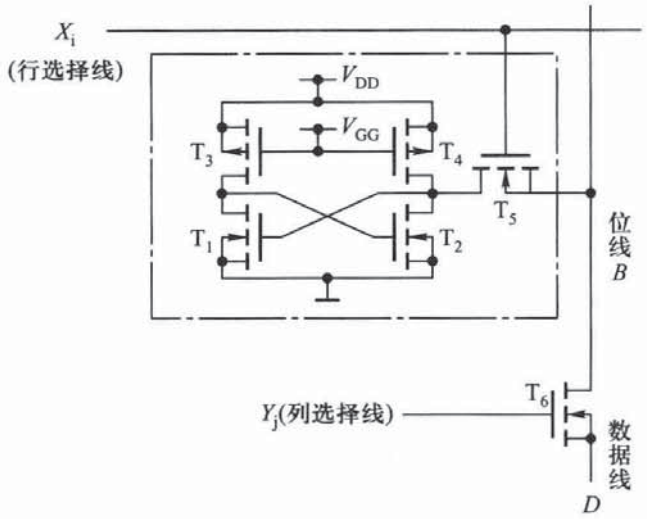


图题解 7.1.5

7.2 随机存取存储器

7.2.1 一个 CMOS 存储单元如图题 7.2.1 所示，试分析其工作原理。

解：此电路为五管静态存储单元电路。T_1 ~ T_4 为两个反相器交叉连接而成的双稳态电路，是存储单元的主体。T_5 和 T_6 分别是由行选线和列选线控制的传输门，当行选线和列选线都为高电平时，T_5、T_6 导通，可以对该存储单元读出或存入数据。



图题 7.2.1

7.2.2 设同步 SRAM 工作在从发模式下，若数据写入的首地址为 0094H，问接下来的 3 个数据将被写入的存储单元的地址分别是多少？

解：接下来的 3 个地址分别是 0095H，0096H，0097H。

7.2.3 一个有 4096 位的 DRAM，如果存储矩阵为 64×64 结构形式，且每个存储单元刷新时间为 100ns，则存储单元全部刷新一遍最快需要多长时间？如果刷新每行的最长间隔时间为15.6 μs，则该 DRAM 的刷新周期最长为多少？刷新操作所用时间占刷新周期的百分比是多少？

解：DRAM 一次可刷新一行，所以将其全部刷新一遍最少需要的刷新时间为

$$64\times100\ \text{ns}=6400\ \text{ns}=6.4\ \mu\text{s}$$

64 行的总间隔时间为

$$64\times15.6\ \mu\text{s}=998.4\ \mu\text{s}\approx1\ \text{ms}$$

这就是此 DRAM 的最长刷新周期。可见刷新操作所用时间（刷新时间）占刷新周期的百分比为

$$6.4\ \mu\text{s}/1\text{ms}=0.64\%\ 。$$

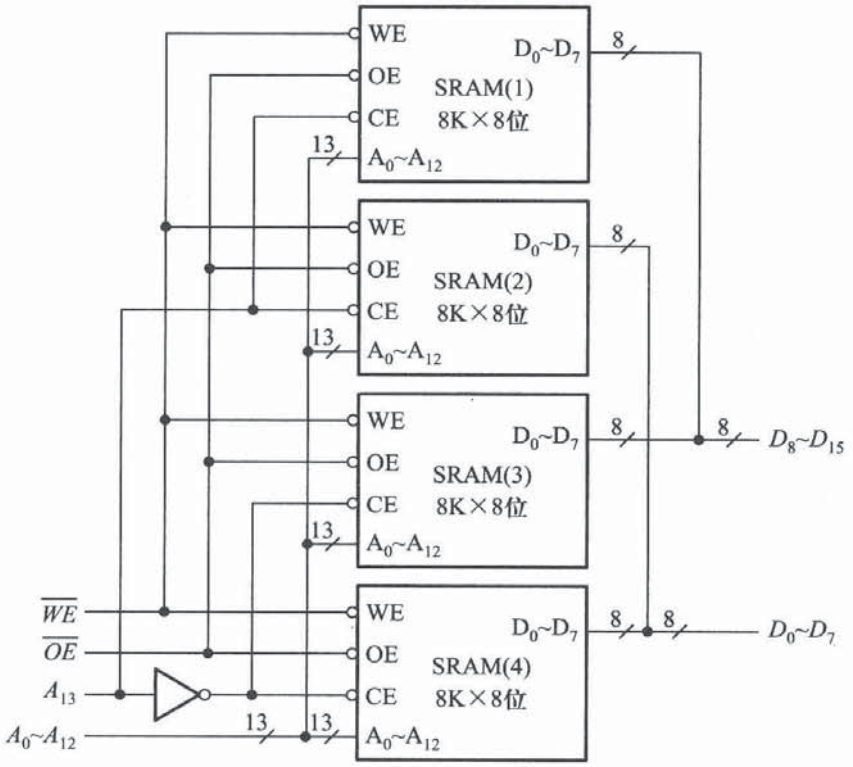
7.2.4 一个有 1M×1 位的 DRAM，采用地址分时送入的方法，芯片应具有几根地址线？

解：由于 $1\text{M}=2^{10}\times2^{10}$，共需 20 根地址线，即行地址线和列地址线各 10 根。所以，采用地址分时送入的方法，芯片应具有 10 根地址线。

7.2.5 试用一个具有片选使能$\overline{CE}$、输出使能$\overline{OE}$、读写控制$\overline{WE}$、容量为 8K×8 位的 SRAM 芯片和必要的逻辑门，设计一个 16K×16 位的存储器系统，试画出其逻辑图。

解:采用 8K×8 位的 SRAM 构成 16K×16 位的存储器系统,必须同时进行字扩展和位扩展。用 2 片 8K×8 位的芯片,通过位扩展构成 8K×16 位系统,此时需要增加 8 根数据线。要将 8K×16 位扩展成 16K×16 位的存储器系统,还必须进行字扩展。因此还需 2 片 8K×8 位的芯片通过同样的位扩展,构成 8K×16 位的存储系统,再与另一个 8K×16 位存储系统进行字扩展,从而实现16K×16 位的存储器系统,此时还需增加 1 根地址线。系统共需要 4 片 8K×8 位的 SRAM 芯片。

用增加的地址线 A_{13}控制片选使能$\overline{CE}$便可实现字扩展,两片相同地址的 SRAM 可构成 16 位数据线。其逻辑图如图题解 7.2.5 所示。其中(1)和(2)、(3)和(4)分别构成两个 8K×16 位存储系统;非门将 A_{13}反相,并将 A_{13}和$\overline{A}_{13}$分别连接到两组 8K×16 的片选使能端$\overline{CE}$上,实现字扩展。



图题解 7.2.5

7.2.6 图 7.2.11 的 LED 点阵列字符动态显示电路中,(1) 若人的视觉暂留时间为 0.05 秒,在满足 LED 阵列图像稳定不闪烁的情况下,试计算 CP 脉冲的最低工作频率。(2) 若要在显示屏上显示“HELP”,试确定 RAM 中存放的点阵数据。(3) 若 LED 阵列改为 16×128(行×列),则需要 RAM 的位数为多少? 容量是多少?

解:(1) 由于 LED 阵列扫描一次需要 32 个 CP 脉冲,所以 CP 脉冲的最长周期应不超过 0.05 s/32,所以最低工作频率为 32/(0.05 s)= 640 Hz。

(2) 根据“HELP”的字形,可确定 RAM 中的点阵数据如表题解 7.2.6。

(3) RAM 的位数为 16,容量是 128×16 位。

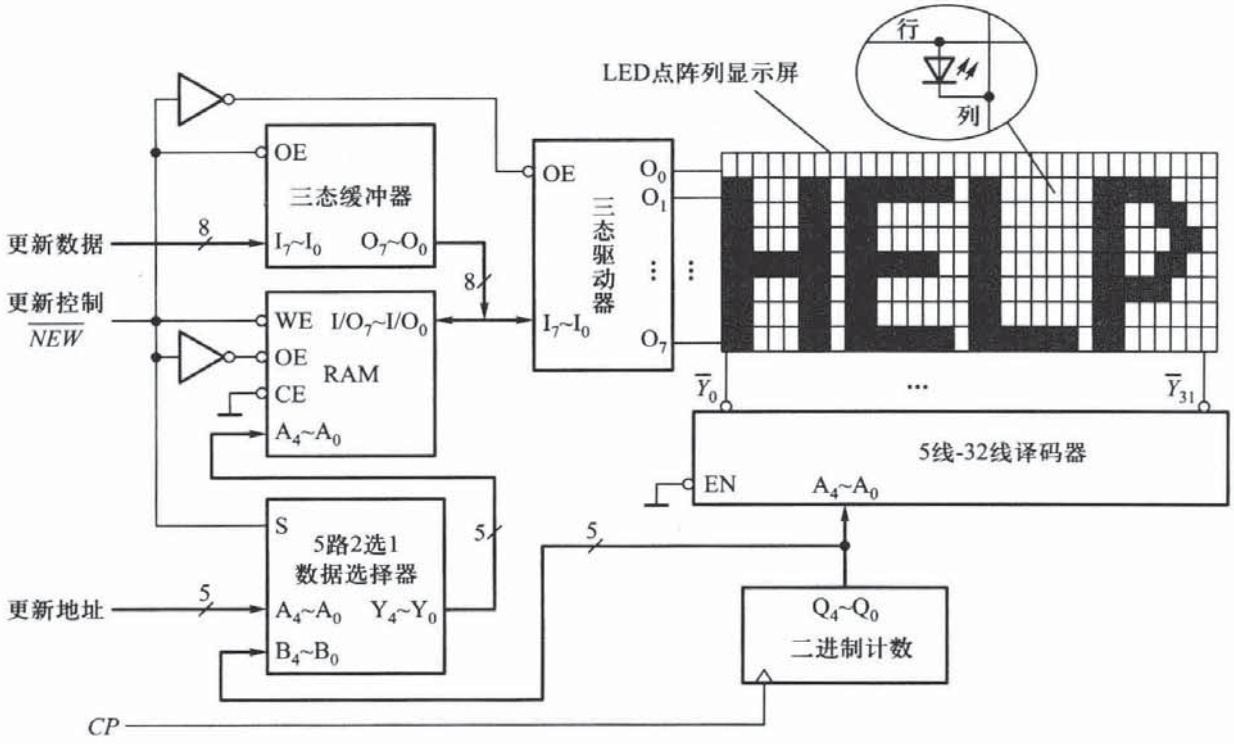


图 7.2.11 8×32 LED 点阵列动态显示原理

表题解 7.2.6 显示“HELP”时 RAM 中的点阵数据

地址 $A_4A_3A_2A_1A_0$	内容 $D_7D_6D_5D_4D_3D_2D_1D_0$	地址 $A_4A_3A_2A_1A_0$	内容 $D_7D_6D_5D_4D_3D_2D_1D_0$
00000	11111110	10000	11111110
00001	11111110	10001	11111110
00010	01010000	10010	10000000
00011	00010000	10011	10000000
00100	00010000	10100	10000000
00101	11111110	10101	10000000
00110	11111110	10110	10000000
00111	00000000	10111	00000000
01000	11111110	11000	11111110
01001	11111110	11001	11111110
01010	10010010	11010	00100010
01011	10010010	11011	00100010
01100	10010010	11100	00110110
01101	10000010	11101	00011100
01110	10000010	11110	00001000
01111	00000000	11111	00000000

8 CPLD 和 FPGA

一、内容提要及重点

本章的内容提要：

1. 可编程逻辑器件(PLD)可以由用户自行设计逻辑功能。它们具有集成度高、可靠性高、处理速度快和保密性好等特点。高密度的可编程逻辑器件一般包括三个基本部分：逻辑块、内部连线资源和 I/O 控制块。

2. 逻辑块是可编程逻辑器件实现各种组合和时序逻辑功能的核心模块。

3. 通过对可编程连线资源的编程，可以实现各逻辑块之间的相互连接，也可以与 I/O 模块相连，实现与芯片外部的信息交换。

4. I/O 块是可编程逻辑器件封装引脚和内部逻辑间的接口。每个 I/O 单元对应一个封装引脚，通过对 I/O 单元编程，可将引脚分别定义为输入、输出和双向功能。

5. CPLD 是在 GAL 的基础上发展起来的复杂可编程逻辑器件，多数 CPLD 逻辑块的核心是**与-或**阵列和触发器。

6. CPLD 器件通常采用 CMOS E^2PROM 工艺制造，对器件编程后，即使切断电源，其逻辑也不会消失，且可以在系统编程(ISP 特性)。部分 CPLD 内部还集成了 E^2ROM、FIFO 或双口 RAM，以适应不同功能的数字系统设计。

7. FPGA 中的逻辑块不像 CPLD 那样采用可编程的“与-或”阵列来实现逻辑函数，而是采用查找表(LUT)实现逻辑函数。逻辑块中包含若干 LUT 和触发器，这种结构避免了“与-或”阵列结构上的限制，且功能更强。一片 FPGA 可以包含数量众多的逻辑块，能够实现更大规模、更复杂的逻辑电路。FPGA 是目前规模最大、密度最高的可编程器件。

8. 大部分 FPGA 的 LUT 由数据选择器和 SRAM 构成，切断电源后，其逻辑会消失。所以 FPGA 一般需要一个外部的 PROM 保存编程数据。上电后，FPGA 首先从 PROM 中读入编程数据进行初始化，然后才开始正常工作。

9. 无论是 CPLD 还是 FPGA，其开发过程必须借助相关开发软件和编程设备才能完成。

本章的重点：

可编程逻辑器件的结构特点；用可编程逻辑器件设计数字电路的方法。

二、典型例题解析

例 8.1 已知逻辑函数 $L=A\overline{B}\,\overline{C}+\overline{A}C$，若用 2 输入的 LUT 实现该逻辑函数，需要几个 LUT？确定 LUT 中的内容，并以主教材图 8.2.2 所示的形式画出。

解题思路：由于采用 2 输入的 LUT，所以输入变量个数不能超过 2 个，需要先将逻辑函数进行拆分，然后确定 LUT 的内容。

解：通过增加中间变量，将逻辑函数拆分为能用 2 输入变量表达的逻辑函数：

$$L=A\overline{B}\,\overline{C}+\overline{A}C=(A\overline{B})\overline{C}+\overline{A}C=F_1\overline{C}+F_2=F_3+F_2$$

其中：$F_1=A\overline{B}$，$F_2=\overline{A}C$，$F_3=F_1\overline{C}$。这样 F_1、F_2、F_3 和 L 都是 2 输入逻辑函数，由此可知共需 4 个 2 输入的 LUT，LUT 的内容及逻辑函数实现如图 8.1 所示。

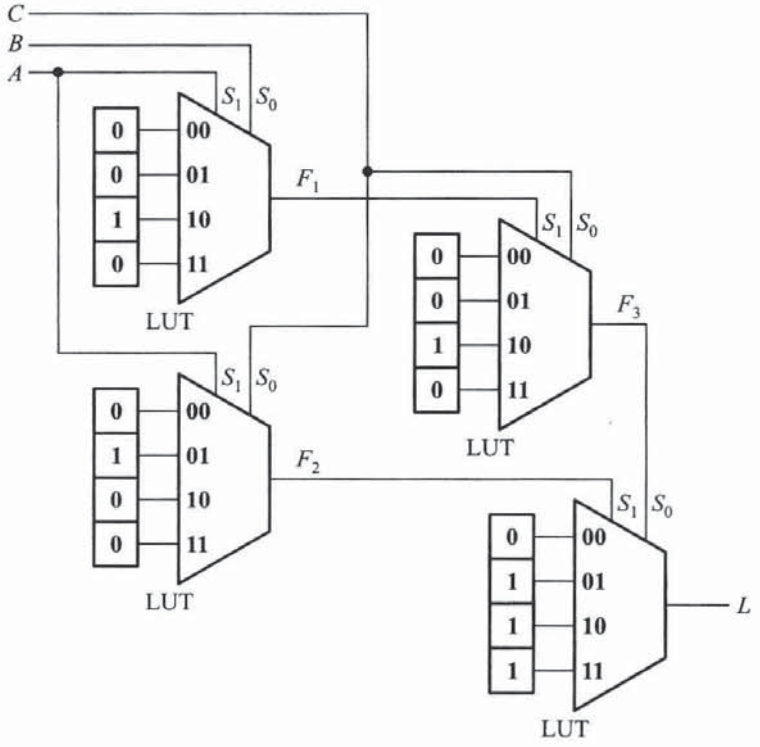


图 8.1

三、习 题 全 解

8.1 复杂可编程逻辑器件(CPLD)简介

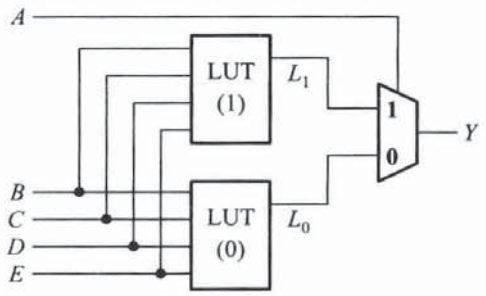
8.1.1 若某 CPLD 中的逻辑块有 36 个输入(不含全局时钟、全局使能控制等)，16 个宏单

元。理论上,该逻辑块可以实现多少个逻辑函数?每个逻辑函数最多可有多少个变量?如果每个宏单元包含 5 个乘积项,通过乘积项扩展,逻辑函数中所能包含的乘积项数目最多是多少?

解:每个宏单元可以产生一个逻辑函数,所以可以实现 16 个逻辑函数。每个逻辑函数可以有 36 个变量。通过乘积项扩展,逻辑函数中包含的乘积项最多为 16×5=80 个。

8.2 现场可编程门阵列(FPGA)

8.2.1 电路如图题 8.2.1 所示,LUT 的内容如表题 8.2.1 所示。试写出 Y 的逻辑函数表达式。



图题 8.2.1

表题 8.2.1

$B\ C\ D\ E$	$L_1\ L_0$	$B\ C\ D\ E$	$L_1\ L_0$
0 0 0 0	0 1	1 0 0 0	0 1
0 0 0 1	0 0	1 0 0 1	1 0
0 0 1 0	1 0	1 0 1 0	0 0
0 0 1 1	0 0	1 0 1 1	0 1
0 1 0 0	1 0	1 1 0 0	0 1
0 1 0 1	0 0	1 1 0 1	1 0
0 1 1 0	0 1	1 1 1 0	1 0
0 1 1 1	0 1	1 1 1 1	0 1

解:由表题 8.2.1 可写出:

$$L_0=\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}+\overline{B}CD\,\overline{E}+\overline{B}CDE+B\,\overline{C}\,\overline{D}\,\overline{E}+B\,\overline{C}DE+BC\,\overline{D}\,\overline{E}+BCDE$$

$$L_1=\overline{B}\,\overline{C}D\,\overline{E}+\overline{B}C\,\overline{D}\,\overline{E}+B\,\overline{C}\,\overline{D}E+BC\,\overline{D}E+BCD\,\overline{E}$$

由图可知

$$\begin{aligned}Y&=AL_1+\overline{A}L_0\\&=A(\overline{B}\,\overline{C}D\,\overline{E}+\overline{B}C\,\overline{D}\,\overline{E}+B\,\overline{C}\,\overline{D}E+BC\,\overline{D}E+BCD\,\overline{E})\\&\quad+\overline{A}(\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}+\overline{B}CD\,\overline{E}+\overline{B}CDE+B\,\overline{C}\,\overline{D}\,\overline{E}+B\,\overline{C}DE+BC\,\overline{D}\,\overline{E}+BCDE)\end{aligned}$$

8.2.2 用 5 个 2 输入 LUT 实现逻辑函数:$f(x_1,x_2,x_3)=x_1\bar{x}_2+x_1x_3+x_2\bar{x}_3$。用主教材中图 8.2.3 的风格标出每个 LUT 的真值表,不必画 FPGA 中的无关连线和编程节点。

解:$F=f(x_1,x_2,x_3)=x_1\bar{x}_2+x_1x_3+x_2\bar{x}_3=F_1+F_2+F_3=F_4+F_3$

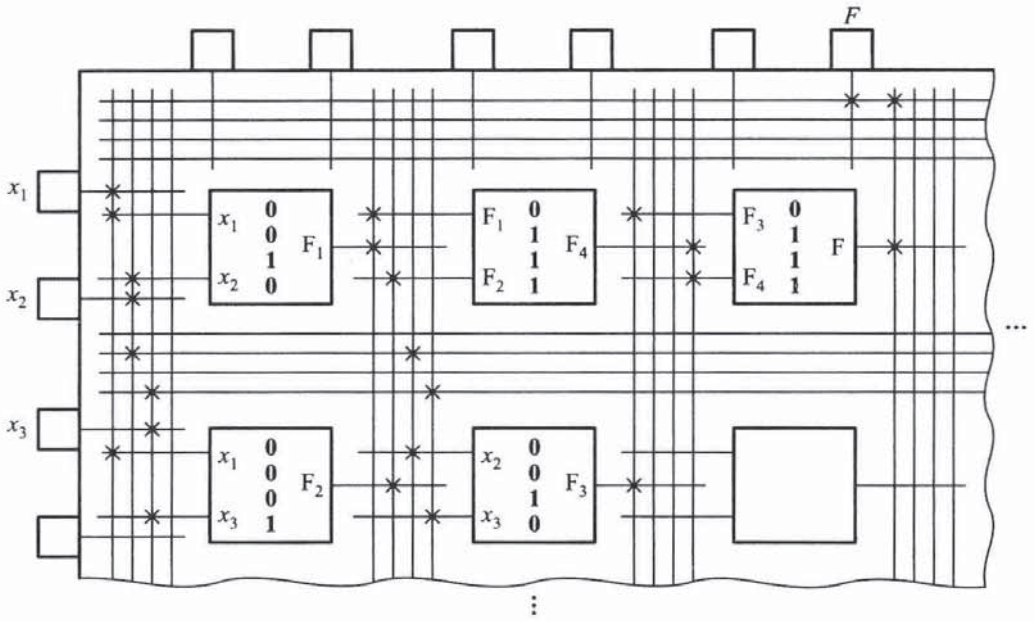
其中:$F_1=x_1\bar{x}_2$,$F_2=x_1x_3$,$F_3=x_2\bar{x}_3$,$F_4=F_1+F_2$。由此可知逻辑函数实现如图题解 8.2.2 所示。

8.2.3 用 2 个 2 输入 LUT 实现逻辑函数:$f(x_1,x_2,x_3)=\sum m(2,3,4,6,7)$。用主教材中图 8.2.3 的风格标出每个 LUT 的真值表,不必画 FPGA 中的无关连线和编程节点。

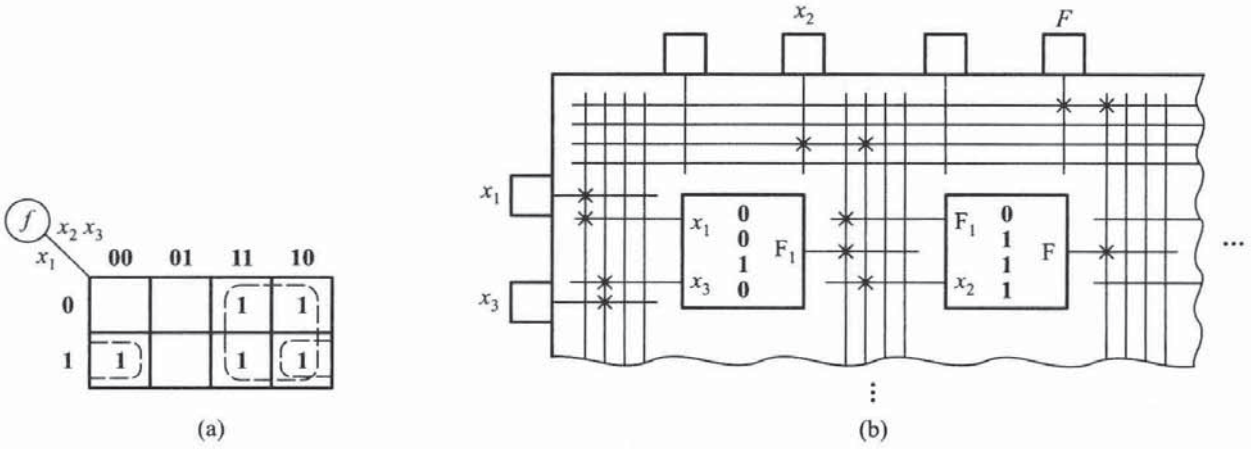
解:先化简逻辑函数。由图题解 8.2.3(a)卡诺图化简得:

$$F=f(x_1,x_2,x_3)=x_1\bar{x}_3+x_2=F_1+x_2$$

其中:$F_1=x_1\bar{x}_3$,由此可知逻辑函数实现如图题解 8.2.3(b)所示。



图题解 8.2.2



图题解 8.2.3

8.4 用 EDA 技术和可编程器件的设计例题

8.4.1 试用 Verilog 语言设计一个 1 位十进制数(用 8421BCD 码表示)加法器,用 Quartus Ⅱ软件对电路进行逻辑功能仿真,并给出仿真波形。

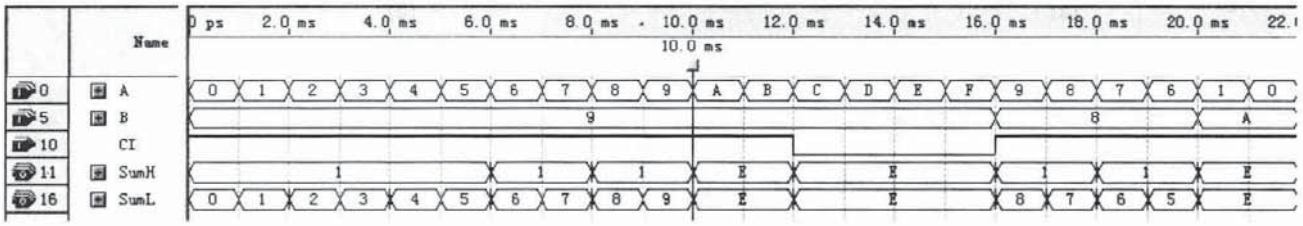
解:由于 **Verilog HDL** 语言的算术加法指令只适用于二进制数的加法,所以在源程序中对输入的数据按二进制数的加法进行计算,然后对相加的结果再进行调整,即将相加的二进制数结果转换为 8421BCD 码表示。

程序第一个 **always** 块完成二进制数相加,第二个 **always** 块完成二进制数到 **BCD** 码转换。程序中,A、B 为两个用 8421BCD 码表示的数,分别代表被加数和加数,**S** 代表和数,为 5 位二进制数,SumH、SumL 为 8421BCD 码表示的和数,**ERR** 表示输入数据不为 **BCD** 码时的出错信号。

1 位十进制数加法器的源程序如下:

```
module BCD_ADDER (SumH, SumL, A, B, CI);   //模块开始
    input [3:0] A, B;                      //用 8421 BCD 码表示的被加数、加数
    input  CI;                             //低位来的进位输入
    output reg [3:0] SumH, SumL;           //用 8421 BCD 码表示的和数
    reg ERR;                               //Input Error Flag
    reg [4:0] S;                           //Binary result
    always @ (A or B or CI)
      if ((A>9)|(B>9))                     //输入数据出错
        begin S=5'd0; ERR=1'b1; end
      else                                 //完成二进制加法运算
        begin
          S=A+B+CI;
          ERR=1'b0;
        end
    always @ (S or ERR)                    //完成 4 位二进制数到两位 BCD 码转换
      case (ERR)
        1'b0:if (S<4'd10)
                begin SumH=4'd0; SumL=S[3:0]; end
             else
                begin SumH=4'd1; SumL=S-5'd10; end
        1'b1:  {SumH,SumL}={4'b1110,4'b1110}; //输入数据错误时,输出代码 EE
      endcase
endmodule
```

仿真波形如图题解 8.4.1 所示。



图题解 8.4.1

8.4.2 设计一个 4 位数字显示的简易频率计。要求：

(1) 能够测试 100 Hz ~ 9999 Hz 正方波信号(幅度为 3 ~ 5 V)的频率。

(2) 电路输入的基准时钟为 1 Hz,要求测量值以 8421BCD 码形式输出。

(3) 系统有复位按键。

(4) 采用分层次、分模块的方法,用 Verilog 语言进行设计。

(5) 最后用 Quartus Ⅱ 软件对电路进行逻辑功能仿真,并给出仿真波形。

解:数字频率计的测频原理如图题解 8.4.2(a)所示,在确定的闸门时间 T_w 内,记录被测信

号的脉冲个数 N_X，则被测信号的频率为：$f_x = N_X / T_W$。若 T_W 为 1 秒，被测信号被送到测量计数器进行计数，当 1 秒钟结束时，闸门关闭，计数器停止计数，则信号的频率就是计数器的计数值 N_X。这种方法的计数值会产生±1 个字误差，测试精度与计数器中记录的数值 N_X 有关。

数字频率计组成框图如图题解 8.4.2(b)所示，系统包括逻辑控制模块、测量计数器、锁存器和译码显示电路。由 FPGA 实现前三个模块，如图中虚线框内所示。逻辑控制模块的输入为 1Hz 标准时钟信号和系统复位信号，输出为三个控制信号，分别为：Cnt_CR、Cnt_EN 和 Latch_Sig。Cnt_CR 信号用于在每一次测量开始时，对测量计数器进行清零，以清除上次测量结果，该复位信号高电平有效，持续半个时钟周期的时间。Cnt_EN 信号为允许计数信号，即闸门控制信号，用于控制测量计数器的计数时间。在 Cnt_EN 信号上升沿到来时，测量计数器开始对被测脉冲信号进行计数，计数时间达到 1 秒钟时，停止计数，此时计数器的计数值即为被测信号的频率。然后将该值锁存，并送译码显示电路显示出来。Latch_Sig 为锁存信号，在其上升沿到来时，将测量计数器的值锁存到寄存器中。设置锁存器的好处是使显示的数据稳定，因为计数器在 1 秒钟内要计成千上万个输入脉冲，若不加锁存器，显示器上的数字将随计数器的值而变化，不便于读数。每一次测量开始时，都要先发一个清零脉冲将计数器清零。系统复位信号用于对整个系统进行复位，低电平有效。

分析图题解 8.4.2(c)所示的波形可知，该频率计完成一次频率测量所需时间包括计数、锁存和清零的时间，总共为 2 秒。如果提高逻辑控制电路输入的基准时钟信号是频率，使锁存数据和清零的时间缩短，则可以缩短测量时间。

频率计的 Verilog 程序如下：

```
//频率计的顶层模块
module Fre_Measure (BCD, _1HzIN, SigIN, nRST_Key,Cnt_EN, Cnt_CR,Latch_Sig,Cnt);
    input  _1HzIN, SigIN, nRST_Key;
     output wire [15:0] BCD;                    //锁存器的输出信号,给译码显示电路
//内部节点信号的定义
    output wire Cnt_EN, Cnt_CR;//测量计数器的控制信号,为了便于仿真,将该信号引出
    output wire Latch_Sig;                      //锁存信号,为了便于仿真,将该信号引出
    output wire [15:0] Cnt;                     //测量计数器 8421 BCD 码输出信号
//===============  Call Control Block ==========================
control U0(Cnt_EN, Cnt_CR, Latch_Sig, nRST_Key, _1HzIN);
//===============  Measure counter=============================
counter10 U1(Cnt[3:0], Cnt_CR, Cnt_EN,  SigIN); // ~nRST_Key
counter10 U2(Cnt[7:4], Cnt_CR, Cnt_EN, ~(Cnt[3:0]==4'h9)); //异步计数器
counter10 U3(Cnt[11:8], Cnt_CR, Cnt_EN, ~(Cnt[7:0]==8'h99));
counter10 U4(Cnt[15:12], Cnt_CR, Cnt_EN, ~(Cnt[11:0]==12'h999));
//===============  Call Latch Block ==========================
Latch_16 U5(BCD, Cnt, Latch_Sig, ~nRST_Key);
endmodule
```

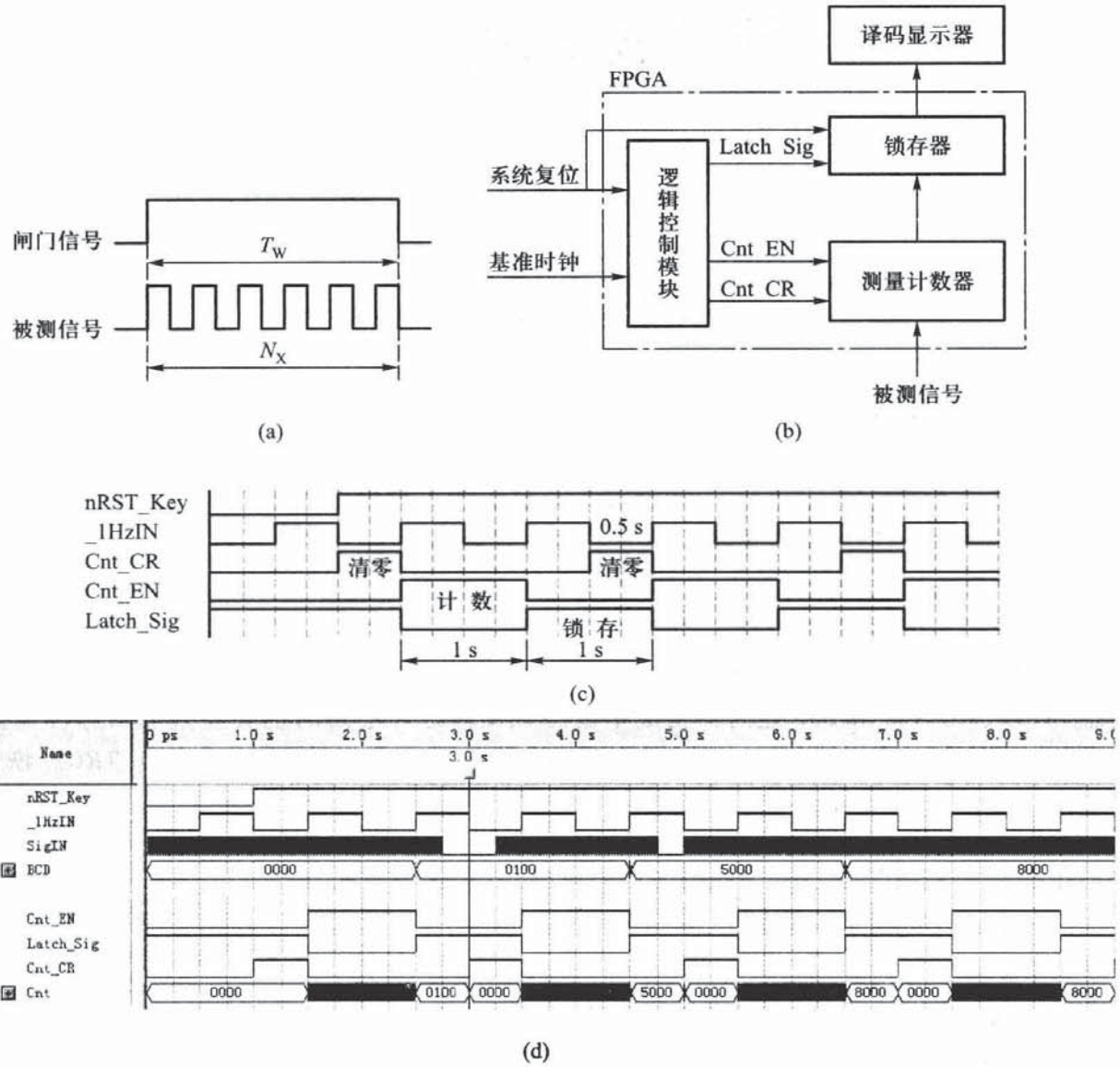
```
//以下是频率计的各个下层子模块
// ********** Control Block **********
module control(Cnt_EN, Cnt_CR, Latch_Sig, nRST, CP);
    input CP, nRST;
    output reg  Cnt_EN;
    output wire  Cnt_CR, Latch_Sig;
    always @(posedge CP or negedge nRST)
    begin
    if(~nRST)                                    //产生测量计数器的使能信号
        Cnt_EN=1'b0;                             //不允许计数
    else  Cnt_EN=~Cnt_EN;                        //对 CP 进行 2 分频
    end
    assign Latch_Sig=~Cnt_EN;                    //产生锁存信号
    assign Cnt_CR=nRST & (~CP & Latch_Sig);      //产生测量计数器的清零信号
endmodule

// ********** Latch Block **********
module Latch_16(Qout, Din, Load, CR);
    input Load,CR;
    input [15:0] Din;
    output reg [15:0] Qout;
    always @(posedge Load or posedge CR)
  if (CR) Qout<=16'h0000;
  else  Qout<=Din;
endmodule

// ********** counter10.v (BCD: 0~9) **********
module counter10(Q,CR,EN,CP);
    input CP,CR,EN;
    output reg [3:0]Q;
    always @(posedge CP or posedge CR)
    begin
      if(CR)      Q<=4'b0000;                    //CR=1,计数器被异步清零
      else if(~EN) Q<=Q;                         //EN=0,暂停计数
      else if(Q==4'b1001)
          Q<=4'b0000;
      else Q<=Q+1'b1;                            //计数器增 1 计数
    end
```

endmodule

仿真波形如图题解 8.4.2(d)所示。



图题解 8.4.2

(a) 测频原理 (b) 频率计组成框图 (c) 控制信号的时序关系 (d) 仿真波形图