第 30 卷 第 2 期
2007 年 4 月

电 子 器 件
Chinese Journal Of Electron Devices

Vol. 30 No. 2
Apr. 2007

Interface Circuit Design and Programming for Storing the Data with TMS320VC5509 by Using Flash Memory

WU Mei-jun, WU Nai-ling
(Electronic Science and Engineering College, SEU, Nanjing 210096, China)

Abstract: The main features of the TMS320VC5509 DSP are introduced briefly. Several aspects which should be paid attention to when storing the data with TMS320VC5509 by using flash memory are indicated in detail, such as interface circuit design, addressing the external data space, initializing the registers and detecting the program or erase operation to flash memory using DSP. Furthermore, the solution of accessing the flash memory AM29LV800 correctly at high frequency by appropriate delay is proposed, the program of which is given. The validity of solution is verified in practice.

Key words: DSP; flash memory; addressing data space; delay
EEACC: 1265

TMS320VC5509 应用 flash 存储数据的接口电路设计与编程

伍美俊, 吴迺陵
(东南大学电子科学与工程学院, 南京 210096)

摘 要: 简要介绍了 DSP 芯片 TMS320VC5509 的主要特点, 指出了 TMS320VC5509 在使用 flash 存储器 AM29LV800 存储数据的应用中, 在电路接口设计、数据空间寻址、寄存器初始化设置及 DSP 对 flash 的写或擦除操作的检测等方面应该注意的一些问题. 提出了 DSP 工作在较高的时钟频率时通过适当延时来正确访问 AM29LV800 的解决方案, 并给出了主要的编程范例. 在实践中证明了方案的有效性.

关键词: DSP; flash 存储器; 数据空间寻址; 延时
中图分类号: TP274 文献标识码: A 文章编号: 1005-9490(2007)02-0675-04

在数据采集系统中, 往往有大量的数据要求进行非易失性存储. 采用 flash 存储器对这些大量数据进行存储是一种性价比较高的选择. 在笔者设计的以 TI 公司的 DSP 芯片 TMS320VC5509 为主处理器的数据采集及处理系统中, 使用了 AMD 公司的 flash 存储器 AM29LV800 作为数据存储芯片. 使用 C5509 对 AM29LV800 进行控制, 在硬件接口设计及软件编程方面都存在值得我们注意的地方.

1 TMS320VC 5509 简介

TMS320VC5509 是 TI 公司的一款 16 位高速低功耗的 DSP 芯片. 主要应用于对音频、图像的数字信号处理, 是设计便携设备的较佳解决方案. 该 DSP 芯片在核心电压为 1.35 V 时, 最大工作频率为 144 MHz. 由于地址线为 24 位宽, 因此对地址的寻址范围为 16 M ×8-Bit 或 8 M ×16-Bit. C5509 采用统一的编址方式, 即存储空间地址没有重叠. 但是寻址方式却有两种不同的方式 ——字节寻址和字寻址. 当 DSP 中的 CPU 访问程序存储空间时, 采用字节寻址方式, 而 CPU 访问数据存储空间时, 采用的是字寻址方式. C5509 通过外部存储器接口(EMIF)这一片内外设对外部存储空间进行管理. 如图 1 所示, 外部存储空间被分成 CE0～CE3 四个空间, 分别由 EMIF 的 CE0#～CE3# 引脚管理. 而 CEn# (n = 0～3)一般与外设的片选端连接. 例如: 当 CE1# 引脚为低电平时, CE1# 引脚选中的存储器, 其存

收稿日期: 2006-08-17
作者简介: 伍美俊(1975-), 男, 工作于桂林空军学院, 讲师, 在读研究生, 主要研究方向为电路与系统设计, wumeijun@163.com.

储内容就会映射到 CE1 空间. AM29LV800 等外部存储设备通过 EMIF 与 C5509 进行无缝链接.

图 1 外部存储空间的划分

2 硬件电路设计

2.1 C5509 与 flash 的硬件连接

由于我们普遍使用 LQFP 封装的 C5509,它对外只提供 14 条地址线引脚,因此只能寻址 8k ×16 bit 空间范围. 而要对 AM29LV800 的 512 k ×16 bit 存储空间寻址,需要 19 条地址线. 解决的办法是通过译码、锁存电路使数据总线中的 D[7:0]具有数据/地址复用功能. 其中只用 D[5:0]为 flash 存储器 AM29LV800 提供高 6 位地址. C5509 与 AM29LV800 的硬件连接图如图 2.

图 2 TMS320VC5509 与 AM29LV800 的硬件连接示意图

由于 TMS320VC5509 不支持对 8 位宽的异步存储器进行访问,因此在图中,AM29LV800B 的 BYTE # 位置 1,从而定义其成为 16 位宽模式. 译码器使用 74HC138,其 CS # 引脚接 CE3 # 使锁存器 74HC245 的寻址地址为字地址 600007h (按图中电路所接),向该地址写数据就可以通过数据总线中的 D[5:0]向 flash 存储器 AM29LV800 提供高 6 位地址.

2.2 C5509 对 16 位宽 flash 存储器的寻址

由于 DSP 中在访问程序存储空间时,采用字节寻址方式,而访问数据存储空间时,采用的是字寻址方式. 因此,在程序编写时对数据空间的访问都应该定义字地址. 例如由图 2 可知,AM29LV800 的 CE # 引脚与 C5509 的 CE1 # 引脚相连,则编写程序时 flash 的起始地址应定义为字地址 200000h 而不是字节地址 400000h.

C5509 的字地址宽度为 23 bit,而其内部地址总线为 24 bit,字地址要加载到内部地址总线上进行传输时,内部地址总线就在 23 bit 地址的后面自动再添上一个 0. 例如:一条指令要读取字地址 (23 bit) 为 000013h 处的一个字,则读数据地址总线传输的 24 bit 实际值为 000026h.

字地址:000 0000 0000 0000 0001 0011

数据地址总线:0000 0000 0000 0000 0010 0110

因此在图 2 中,C5509 的地址线 A0 不与 AM29LV800 相连. 如果我们将 EMIF 的 A[13:0]与 flash 存储器的 A[13:0]连接,则当编程对字地址 200000h 寻址时,被设置成 16 位宽模式的 AM29LV800 获得的地址是 400000h,并且因获得的总是偶数地址而出现错误.

3 C5509 对 flash 读写操作的编程

3.1 相关寄存器的初始化设置

在 DSP 对 AM29LV800 进行读写访问之前,要对 DSP 内相关寄存器进行正确的设置. 相关的寄存器设置有:外部总线选择寄存器 EBSR、EMIF 外设中的 EMIF 全局控制寄存器 EGCR 及 CE1 空间控制寄存器 CE11、CE12 和 CE13.

我们应该注意 CE 寄存器中的三个时间位域的设置即:读或写的建立时间 (t_{setup})、选通时间 (t_{strobe}) 和保持时间 (t_{hold}). 三个时间的定义如图 3 所示. 图 3 是以写时序为例. t_{WPH} 为两个写脉冲的间隔时间. t_{WC} 为写周期时间. 表 1 给出了 AM29LV800 的这几项参数的最小值.

图 3 TMS320VC5509 中三个时间的定义

表 1 AM29LV800 写操作各时间参数

Parameter/min	Speed Option/ns		
	-70	-90	-120
t_{WC}	70	90	120
t_{setup}		0	
t_{strobe}	35	35	50
t_{hold}		0	
t_{WPH}		30	

在表 1 中,虽然 t_{setup} 和 t_{hold} 的最小值为 0 ns,但在实际运用中,根据 t_{strobe} 与 t_{WC} 的具体情况,我们一般会赋予它们一个大于 0 的数值. 以 70 ns 的

AM29LV800B 为例,如果 C5509 的工作频率设为 144 MHz,则其一个时钟周期约为 7 ns,按照 $t_{strobe} \geq 40$ ns, $t_{setup} \geq 20$ ns, $t_{hold} \geq 10$ ns 来取值,则在 CE12 寄存器中的三个位域 WRSTROBE、WRSETUP 和 WRHOLD 应分别取 6、3、2 个时钟周期数.

下面是 C5509 的 EMIF 外设中相应寄存器的初始化设置程序举例:

```
//* * * * * * * * * * *flash_ini.h * * * * * * * * * * * * * * * *
ioport unsigned int  *ebsr  = (unsigned int  *) 0x6c00;
ioport unsigned int  *egcr  = (unsigned int  *) 0x800;
ioport unsigned int  *ce11  = (unsigned int  *) 0x806;
ioport unsigned int  *ce12  = (unsigned int  *) 0x807;
ioport unsigned int  *ce13  = (unsigned int  *) 0x808;
 *ebsr = 0x01;      //选定完全 EMIF模式
 *egcr = 0x0a10;    //关闭 ARDY 控制
//在 144MHz DSP 工作频率下,设置 CE1
 *ce11 = 0x131a     //设置存储器模式为异步 16
                    //位,读操作的 setup、strobe、
                    //hold 与写操作的相同.
 *ce12 = 0xf31a;//写操作的 setup、strobe、hold
 *ce13 = 0;
```

在 DSP 复位后,CE 寄存器的以上三个时间位域默认值都为最大值. 这当然可以保证所用闪速存储器可以可靠工作,但闪速存储器优异的快速读写性能就会因此不能得到充分发挥. 所以,我们还是应该精心的设置这些时间参数.

3.2　读写操作的编程及读写状态的判断

对 AM29LV800 的读写过程必须严格按照 AM29LV800 提供的命令时序来完成. 表 2 给出了 AM29LV800 对数据字进行复位、擦除、读、写的命令时序,更多命令时序参看 AM29LV800 使用手册.

表 2　AM29LV800 的命令时序

当我们把擦除或编程的命令字按照其时序写入 flash 时,可以通过嵌入在 AM29LV800 的内部算法来判断读写操作的完成情况. 在写数据字命令时序或擦除命令时序的最后一个 WE 上升沿到来之后,AM29LV800 会自动运行一个嵌入的内部算法来判断编程或擦除操作是否结束. 如图 4 所示,图中

图 4　AM29LV800 写操作时序图

t_{WHWH1} 为内部算法需经历的时间,其典型值为 9 μs (字节)和 11 μs(字). 内部算法通过 flash 数据总线中的 DQ7、DQ6、DQ5 等引脚向外输出的状态来反映其写或擦除操作目前的情况. DQ7 为数据检测位 (Data # Polling bit),在写操作的内部算法执行过程中,如果输出的 DQ7 位为输入数据的 DQ7 位的补码,则说明写操作正在执行过程中;如果输出的 DQ7 位和输入数据的 DQ7 位状态相同,则说明写一个字(字节)的操作结束,我们即可以进行下一个操作. 擦除程序时,如果输出的 DQ7 为 0,则说明擦除未完成,DQ7 为 1 则擦除操作结束.

写或擦除操作的内部算法必须在输出的 DQ5 状态为 0 时完成. 若 DQ5 为 1 后,DQ7 仍为数据相应位的补码,则写或擦除操作失败. 图 5 为用 DQ7 判断写操作完成的流程图.

图 5　Data # polling 位的算法

3.3　在较高的 CPU 时钟频率下对 AM29LV800 写操作的延时

经过实践发现,按照图 5 编写的写操作结束判断程序在 C5509 工作于 48 MHz 频率以下时,可以正确执行. 但工作频率大于 48 MHz 后,判断程序必须加一定延时才能正常工作. 实验结果为:工作在 96 MHz 是,需延时 3.75 μs,延时 5.225 μs 时,工作频率在 144 MHz 程序仍可正常执行. 分析结果,我认为这是因为 DQ5 的定时时间发生了变化. 从使用手册可知,当 AM29LV800 的内部算法启动后,DQ5 的状态由一个内部定时器来决定. 当内部计数

器计数值溢出时,DQ5 的状态由 0 变为 1. 在频率为 48M Hz 以下时,决定 DQ5 状态的内部定时器计时的最大值大于内部算法所需的时间,所以内部算法可以正常执行,当频率升高,时钟周期变小,导致内部计数器计时的最大值小于内部算法所需的时间后,写操作就会出现异常. 解决这一问题的最简办法自然是加延时. 由实验结果可知,我们只要延时 6 μs,即可让 AM29L V800 的内部算法在 C5509 的任何工作频率中正常执行. 以下为 C5509 对 AM29LV800 进行写操作的部分程序,共大家参考:

```
# include flash_ini.h
# define flash_ba 0x200000 //定义 AM29L V800 的
                           //起始地址
unsigned int PA ,PD;
/ * * * * *一个周期命令字的子程序 * * * * */
void write_se(unsigned int se_addr,unsigned int se_data )
{
   unsigned int  *flash_adr;
flash_adr = (unsigned int  *) (flash_ba + se_addr);
    *flash_adr = se_data;
}
/ * * *判断写操作是否完成的子程序 * * */
void program_over(void)
{
   unsigned int algorith_out;
   unsigned int datapolling_bit;
   unsigned int exceed_time;
   Delay(144,6);// 频率 144M,延时 6 μs
   algorith_out = read_arry(PA);
   datapolling_bit = intalgorith_out &0x80;
   exceed_time = algorith_out &0x20;
while((datapolling_bit! = PD &0x0080) &&( exceed_time!
= 0x20))
   {
     algorith_out = read_arry(PA);
     datapolling_bit = algorith_out &0x80;
     exceed_time = algorith_out &0x20;
   }
If(exceed_time = =0x20)
   {
   datapolling_bit = read_arry(PA) &0x80;
   if(datapolling_bit! = PD &0x0080)
     {
     reset_flash();
     alarm();
     }
   }
}
/ * * * * * *写操作的命令时序 * * * * * */
void program_command(void)
{
   write_se(0x555,0x0aa);
   write_se(0x2aa,0x55 );
   write_se(0x555,0x0a0);
   write_se( PA ,PD);
}
/ * * * * *写一个字到 AM29LV800 * * * * */
void program_operation(void)
{
  program_command();
  program_over();
}
……
```

4 结束语

对数据进行非易失性存储一般使用 EEPROM 或 flash 存储器来完成. EEPROM 不需要擦除时间,而 flash 存储器虽然有较长的擦除时间,但读写速度快,且容量较大时, flash 存储器的价格也比相同容量的 EEPROM 低很多,因此 EEPROM 通常使用在存储数据量较小的场合,而需要对大量的数据进行非易失性存储时,flash 存储器相对有很高的性价比. 本文叙述了 DSP 芯片 TMS320VC5509 使用 AM29LV800 存储数据时,接口设计原理和程序编写的过程以及在硬件和软件设计过程中需要注意的问题. TMS320VC5509 在编程访问其它 flash 存储器时,同样也需要注意这些问题.

参考文献:

[1] Texas Instruments. TMS320VC5509 Fixed- PointDigital Signal Processor Data Manual[S]. Literature Number: SPRS163E, April 2001-Revised July 2003.

[2] Texas Instruments. TMS320VC5509 DSPExternal Memory Interface (EMIF) Reference Guide[S]. Literature Number: SPRU670,October 2003.

[3] Advanced Micro Devices, am29lv800b[R]. AMD Technical specification, publication # 21490,August 2000.

[4] Texas Instruments. TMS320C55x DSP Programmer's Guide [S]. Literature Number: SPRU376a,August 2001.

[5] Texas Instruments. TMS320C55x DSP CPU Reference Guide [S]. Literature Number: SPRU371F,February 2004.

[6] 彭启琮,武乐琴等. TMS320VC55X系列 DSP 的 CPU 与外设[M]. 北京:清华大学出版社. 2005.

[7] 汪春梅,孙洪波等. TMS320VC5000 系列 DSP 系统设计与开发实例[M]. 北京:电子工业出版社. 2004.

[8] 胡庆钟,李小刚等. TMS320VC55X DSP 原理、应用和设计[M]. 北京:清华大学出版社. 2005.