文章编号：1671-1742(2008)03-0300-04

基于DSP TMS320VC5509A图像采集的脱机系统设计与实现

蔡 成， 罗代升， 卿粼波
(四川大学电子信息学院图像信息研究所，四川 成都 610064)

摘要：首先介绍了脱机系统的结构，并给出系统中核心芯片DSP TMS320VC5509A与存储芯片EEPROM AT25640的硬件电路设计方案，从该DSP的各种BOOTLOADER方式中分析研究，考虑到烧写速度及电路设计简洁性等因素，采用了SPI EEPROM模式。随后介绍了加载过程中的数据表——BOOT表的概念及生成方式，详细介绍了程序烧写至存储器的过程及软件实现的流程图，最终用一个实例实现系统的脱机运行。

关 键 词：BOOTLOADER；EEPROM；烧写
中图分类号：TP919.81 **文献标识码**：A

1 引言

目前图像采集系统的实现有多种方式，基于工控的系统实现较简单，然而成本高，携带不便，基于FPGA的系统，压缩算法用硬件实现，并行处理度高，但由于压缩算法较复杂，开发难度比较大，基于DSP的系统，灵活性强，能满足对各种视频格式处理的要求，并具有很好的扩展性、升级性和维护性，系统采用基于DSP的实现方式，对比综合性能后，选择TI公司的TMS320VC5509A作为系统的主处理器，选用ATMEL公司的AT25640做为外部存储器。在硬件平台搭建好以后，将介绍如何利用固化在DSP芯片ROM区中的BOOTLAODER程序实现系统脱机运行，成为便携式的独立应用系统。

2 系统框架图

系统框架如图1所示，当系统上电复位以后，DSP TMS320VC5509A调用固化在ROM区的BOOTLOADER程序从EEPROM-AT25640存储器中自动调用已经烧制好的图像采集程序(程序在经过转化以后以数组数据的形式存放在EEPROM中)，通过COMS进行图像采集，FPGA用于CMOS、FIFO的片选控制和图像采集过程中的时序控制，CMOS数字图像采集头将所采集的数据传送至FPGA辅助系统的FIFO中，然后通过DMA传输方式，将图像信息数据传送到DSP内部RAM或外部SDRAM里，再由DSP做相应的处理，最后送至VGA接口或标准复合视频信号接口。

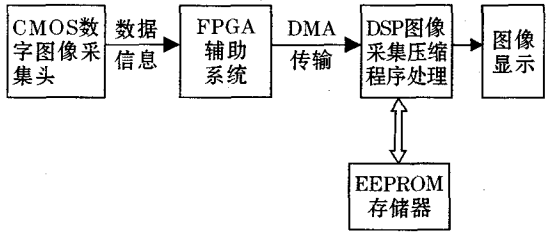

图1 系统框架图

3 TMS320VC5509A与AT25640的硬件电路连接

对数据进行非易失性存储一般使用EEPROM或Flash存储器来完成。相对于Flash存储器，EEPROM不需要擦除时间，缩短了烧制程序的时间，需要对中等大小数据量数据进行存储时，EEPROM是性价比较高的选择。在对比多种EEPROM后，选择ATMEL公司的AT25640EEPROM存储器作为DSP TMS320VC5509A的外部存储器。因为它不仅具有上述的优点，还具有体积小引脚数少的优点，这样为整个系统的电路设计带来了更大的灵

收稿日期：2007-10-30；修订日期：2007-11-15

活性。

如图 2 所示,DSP 作为 SPI 的主设备(Master),EEPROM 作为从设备(Slave),$\overline{WP}$,$\overline{HOLD}$引脚拉高,$\overline{WP}$引脚——写保护引脚,防止外部设备改动 EEPROM 内部存储器或寄存器,由于 BOOTLOADER 只是对 EEPROM 执行读操作,故需把该引脚拉高,$\overline{HOLD}$引脚——延缓串行输入引脚,用来延迟串行数据的输入,如果是选通状态,将会阻止 BOOTLOADER 的正常工作,故需将该引脚拉高。

DSP 通过 DR0 引脚从 EEPROM 的 SO 引脚接收数据,DX0 传输数据给 SI,CLKX0 为 DSP 工作时钟,这时候,EEPROM 的工作时钟频率为 DSP 工作时钟的 244 分频,例如 DSP 工作频率 144MHz,这时候加载的频率为,144/244 = 604kHz。IO4 引脚与 CS 相连作为 EEPROM 的片选信号,当程序加载做好准备后,IO4 自动变低选通 EEPROM,当程序加载完毕后,IO4 自动变高,禁止选通 EEPROM。

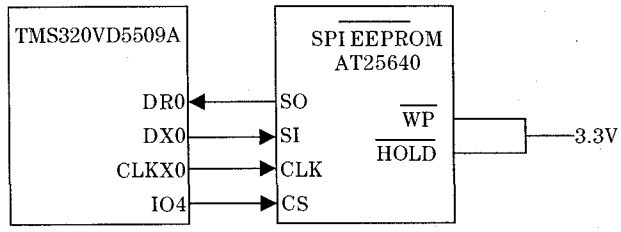

图 2　TMS320VC5509A 与 AT25640 的硬件电路连接

TMS320VC5509A 的自举程序支持两种 SPI EEPROM BOOTLOADER 模式,一种是基于 16 位字节地址的 SPI EEPROM,最大可达到 64K 寻址空间,另一种是基于 24 位字节地址的 SPI EEPROM,最大可达到 16M 寻址空间,利用硬件设置即可实现两种模式的选择,当 DSP 的 IO.0 - IO.3 设置为 1100 时,选择第一种自举模式,当 DSP 的 IO.0 - IO.3 设置为 0100 时,选择第二种自举模式,系统电路板上设置了对应的跳线连接,方便系统以后扩展不同的自举加载模式的选择。系统采用的是第一种模式即把对应的跳线调整为 1100。当程序在后期改进,数据量变大的时候,可以扩展最大 16M 的 EEPROM,硬件实现与系统一致,只需对相应的软件的写程序部份做一些修改,将 16 位地址改为 24 位地址操作。

4　BOOTLOADER

在 DSP 系统引导装载(BOOTLOADER)时,系统上电复位以后,DSP 将一段存储在外部的非易失性存储器的代码(BOOT 表)搬移到内部的高速存储器 RAM 单元中执行,这样既利用了外部的存储单元扩展 DSP 本身有限的 ROM 资源,又充分发挥了 DSP 内部资源的效能,还实现了整个系统独立于计算机,实现真正的脱机系统。

TMS320VC5509A 从上电复位到进入工作状态前一瞬间的这个阶段称为 Boot 阶段,在上电复位以后,DSP 的程序指针自动指到一个固定的入口地址,必须将程序可执行的代码首地址放在这个入口地址处,所谓的 BOOTLOADER 就是 DSP 上电复位以后将固化在外部存储器中的程序读入到 DSP 的片上 RAM 或者片外 RAM 映射成的存储区间的一个过程。其加载的程序,是由一组数据表构成,称之为 BOOT 表。

TMS320VC5509A 有多种 BOOTLOADER 模式,直接从外部存储器执行模式、并行 EMIF 模式、EHPI 模式、标准串行模式、SPI EEPROM 模式、IIC EEPROM 模式、USB 模式等,在系统中,采用 SPI EEPROM 模式。在该模式下首先必须要对 TMS320VC5509A 时钟停止模式下的 McBSP SPI 操作有一定的了解,在 SPI EEPROM 模式中,DSP 作为 SPI 主设备,存储器作为从设备。

32bit 程序入口地址			
32bit 寄存器配置个数			
16bit 寄存器地址		16bit 寄存器内容	
16bit 的延迟说明		16bit 的延迟时间	
32bit 代码段长度			
32bit 代码段起始地址			
数据	数据	数据	数据
数据	数据	数据	数据
32bit 的 0,boot 表结束的标志			

图 3　BOOT 表的结构

4.1　BOOT 表的概念及生成方式

BOOT 表实际上是一组数据,其中包含了要加载到 DSP 中的系列代数据代码如:程序入口地址、寄存器配置信息、程序延迟信息、数据段等,它是由 CCStudio 提供的 hex55.exe 工具配合编写的 CMD 文件生成,如图 3 所示。

BOOT 表的结构:首先是 32bit 的程序入口地址,接下来是寄存器的配置个数用 32bit 表示,再接下来是寄存器配置的详细说明,前 16bit 为所配置的寄存器的地址,后 16bit 是配置寄存器的内容,再下来是 16bit 的延迟指示,一般来说为 0xFFFF 即延时寄存器的地址。接下来是 16bit 的延迟的大小,再接下来的 32bit 表示代码段长

度,然后是 32bit 的代码段的起始地址,后面是数据区,最末尾是结束标志,为 32bit 的全 0。

为直观,在调试板上的 GOIO6、GPIO7 端分别接了 2 个 LED(LED6、LED7),现编写一程序,功能是实现 LED6、LED7 先同时闪烁 3 次,然后交替闪烁,现在把该程序在 CCStudio 编译生成 gpiotest.out 文件,通过仿真器在线下载到 DSP 中,运行成功。然而由于 CCStudio 生成的 .out 可执行文件是 AT&T 的模块化的 COFF 代码格式,这个格式与实际的外部存储器的存储区间不匹配,所以不能直接写入到外部存储器上,因此必须将之转化为外部存储器能匹配的格式,CCStudio 提供了代码格式转化方法来完成这种匹配。

生成 BOOT 表的实现,首先制作一个 CMD 文件(hex.cmd)。

4.2 CMD 文件的实现

```
gpiotest.out          //需要被写到 EEPROM 中去的程序文件
-a                    //输入 ASCII 的十六进制格式
-map gpiotest.mxp     //生成一名为 gpiotest.mxp 的 MAP 表,从这张表里可以找到程序的入口地址
-o cc.hex             //指定输出的文件名
-serial8              //采用串行模式
-e 0x100d6            //修改程序入口地址,
-boot                 //生成一张 boot 表
-v5510:2              //版本匹配
```

把 CCStudio 提供的 hex55.exe 文件与该 CMD 文件放在同一个目录下,在 DOS 环境下,输入命令行:hex55.exe hex.cmd(必须加上后缀名,否则导致运行失败)。运行生成一个名为 cc.hex 的文件。最后,还需编写一个 C 程序将之转化为 .h 的数组头文件,最后将之包含入烧写程序的工程文件中。

5 程序烧写至存储器过程的实现

AT25640 的写时序图如图 4 所示。

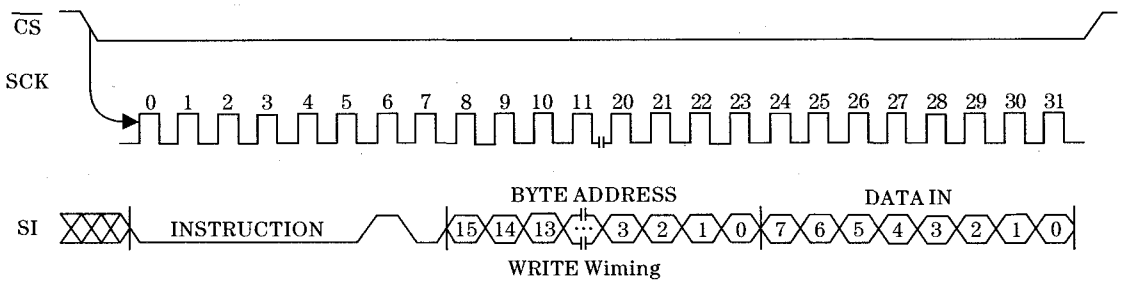

图 4 AT25640 的写时序图

写指令需要按照以下步骤,SCK 为时钟频率,首先是 DSP 使 IO4 口变低,当 CS 被拉低后选通 EEPROM,然后 DSP 向 EEPROM 的 SI 引脚发送一个 8bit 的写操作指令(02h),紧接着是 16bit 的字节地址(A15 - A0)以及 8bit 的数据(D7 - D0)所有的数据在一个时钟上升沿的时候被写入,编程将在 CS 引脚拉高以后开始,最后一个数据 D0 传输完以后 CS 立即发生一个上升沿。在下一个指令到来前,DSP 会先使 IO4 再次变低,以后步骤重复操作,直到所有的数据全部写完为止。

AT25640 的读时序图如图 5 所示。

首先,DSP 使 IO4 为低,选通 EEPROM,接下来 DSP 发出读指令(03h),接着是 16bit 的字节地址,一般来说是全零,EEPROM 响应 DSP,通过 SO 引脚开始传送数据到 DSP 中,一次传 8bit 数据,此时 DSP 不用发送每个数据的地址,而是由 SPI eeprom 内部自动将地址增加,直到所有的数据传完,boot 过程完毕,IO4 变高,禁止选通 EEPROM,然后 DSP 程序指针自动跳转到程序入口地址,开始执行烧写好的程序。

写程序过程如图 6 所示。

编译烧写程序工程,生成 .out 文件,利用仿真器将该文件下载到 DSP 中并运行,将要实现的程序全部烧入至 EEPROM 中,程序运行完毕以后,断开与 CCStudio 的连接,再断开调试板与仿真器的连接,然后再对调试板重新上电复位,LED6、LED7 如前述正常工作,加载成功。

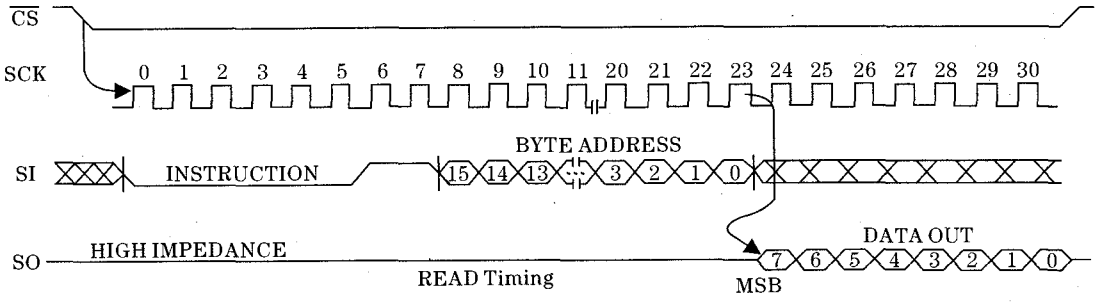

图 5　AT25640 的读时序图

最后连接 FPGA 板与 CMOS 视频采集头,将图像采集程序按照上述方法进行转换并烧写入 EEPROM 中,系统上电复位以后,自动采集图像,实现脱机运行。

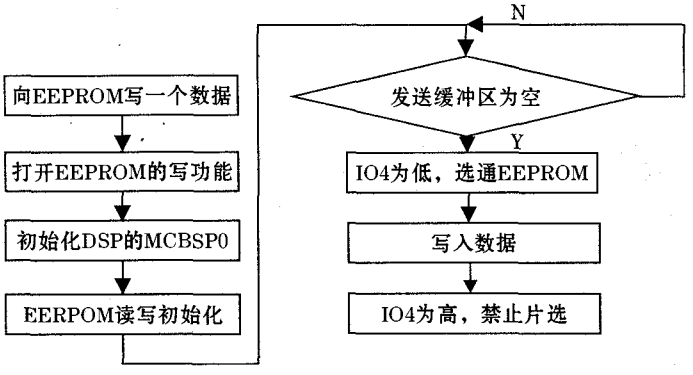

图 6　写 EEPROM 程序流程

6　结论

系统需要注意的几个地方:(1)程序入口地址的确定,该程序入口地址为 _c_int00 的地址,该地址可以在生成的 gpiotest.map 文件中查找。否则将导致程序加载能成功,然而却不自动运行的情况。(2)CMD 文件必须加上 -v5510:2,否则将导致生成不正确的 BOOT 表。(3)在烧写程序的延迟函数里面,延迟设置必须恰当,否则将导致 MCBSP0 初始化失败。

设计最终使整个系统实现真正意义上的脱机,选用的主要的芯片具有低功耗和高效的性能,延长了系统的独立工作时间以及增强系统稳定性,同样,在对其他的 TMSVC32055X 系列 DSP 也可以按照上述的硬件电路设计来实现 BOOTLOADER,具有非常好的通用性,设计具有较高的使用价值及应用前景。

参考文献:

[1]　张勇.C/C++ 语言硬件程序设计[M].西安:西安电子科技大学出版社,2006.
[2]　彭启琮,武乐琴,张舰.TMS320VC55X 系列 DSP 的 CPU 与外设[M].北京:清华大学出版社,2005.
[3]　汪春梅,孙洪波,任治刚.TMS320C5000 系列 DSP 系统设计与开发实例[M].北京:电子工业出版社,2004.

Design and implementation of offline system of image acquisition process based on DSP TMS320VC5509A

CAI Cheng,　LUO Dai-sheng,　QING Lin-bo
(College of Electronic Information, SCU, Chengdu 610064, China)

Abstract: The offline system is introduced. The circuit design of the DSP TMS320VC5509A as the kernel chip and the EEPROM AT25640 as the memory chip in this system is given. The feasible approaches of the DSP BOOTLOADER and the speed and complexity of the circuit are studied and the SPI EEPROM load is adopted. The data table in the loading process-the concept and formation of the BOOT table is then introduced. The complete procedure of the programming memory chip and the flow chart of the source code are described in detail. An example of the realization of the system offline operation is given.

Key words: BOOTLOADER; EEPROM; burn