基于 JTAG 的 Flash 在线编程系统实现

吴玉香,周建香
(华南理工大学 自动化学院,广东 广州 510641)

摘 要:在研究了 JTAG 调试原理和 ARM920T 调试模型的基础上,提出了 Linux 系统下 NAND Flash 在线烧写系统的软硬件实现方案。硬件采用了简易并口 JTAG,软件分为 4 个层次。程序在 Linux 系统下成功编译并运行,实现了 Flash 的在线编程。

关键词:Linux;JTAG;Flash;烧写系统

中图分类号:TP316 **文献标识码**:A

In system Flash programming system based on JTAG

WU Yu Xiang, ZHOU Jian Xiang
(College of Automation, South China University of Technology, Guangzhou 510641, China)

Abstract: Based on the research of JTAG standard and ARM920T debugging model, this paper proposes a hardware and software solution of in system programming system for NAND Flash in Linux operation system. The hardware uses the simple parallel port JTAG and the software includes four layers. This program can be successfully compiled and run in the Linux operation system, achieving the goal of Flash programming.

Key words: Linux; JTAG; Flash; programming system

随着电子技术的迅速发展,芯片以及系统越来越复杂,体积越来越小,系统测试、故障排除的难度和成本不断增加,边界扫描技术为以上问题提供了一个行之有效的解决途径。IEEE 1149.1 标准俗称 JTAG 调试标准,最初由 JTAG(Joint Test Action Group)小组提出,最终由 IEEE 批准并标准化。人们一般用 JTAG 代表 IEEE 1149.1 规范。JTAG 调试标准极大地推动了边界扫描技术的发展,在电子产品设计及调试的各个阶段得到广泛应用。

1 JTAG 调试原理

1.1 边界扫描技术

边界扫描是边界扫描技术的核心概念,其基本思想是在芯片的输入输出管脚上增加一些边界扫描寄存器单元(boundary-scan register cell),边界扫描寄存器单元其实是移位寄存器单元。芯片有两种工作状态:调试状态和正常运行状态。调试状态下,边界扫描寄存器单元将芯片与外围的输入输出隔离开,通过这些边界扫描寄存器单元可以实现芯片输入输出信号的观察与控制。对于芯片输入管脚,通过与之相连的边界扫描寄存器单元可把信号加载到该管脚中去;对芯片输出管脚,通过与之相连的边界扫描寄存器单元可实现对该管脚上的输出信号的捕获(capture)。正常运行状态下边界扫描寄存器单元对芯片来说是透明的,对芯片的正常工作不会造成任何影响[1]。这样边界扫描寄存器就提供了一个便捷的方式实现芯片输入输出信号的观察和控制。此外,芯片输入输出引脚上的边界扫描寄存器单元可以相互串起来在芯片周围形成一个边界扫描链。一般芯片中会提供几条边界扫描链,实现数据的串行输入和输出,在时钟信号和控制信号的作用下,方便地观察和控制调试状态下的芯片。

芯片在调试状态与正常运行状态由不同的时钟信号驱动:正常运行时由系统主时钟(MCLK)驱动,调试状态下由调试时钟(DCLK)驱动[2]。调试时钟 DCLK 一般要比系统主时钟 MCLK 慢。

1.2 TAP(Test Access Port)

边界扫描链可实现数据的输入输出,从而实现对芯片的观测与控制。TAP 是一个通用端口,并在 IEEE1149.1 标准中定义,实现对边界扫描链的控制。

IEEE1149.1 标准里，寄存器分为数据寄存器(DR)和指令寄存器(IR)。边界扫描链只是数据寄存器中的一种。TAP 提供了 4 个强制信号 TDI、TDO、TMS、TCK 和一个可选信号 TRST。通过这些控制信号实现对数据寄存器(DR)和指令寄存器(IR)的访问。JTAG 结构示意图如图 1 所示。

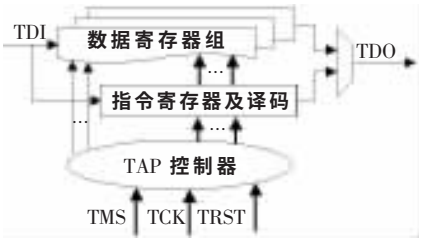

图 1 JTAG 结构示意图

(1)TCLK(Test Clock Input)：TAP 时钟驱动信号。

(2)TMS(Test Mode Select)：TAP 的模式选择信号，用来控制状态机的转换。

(3)TDI(Test Data Input)：数据串行输入接口。

(4)TDO(Test Data Output)：数据串行输出端口。

(5)TRST(Test Reset Input)：TAP Controller 复位信号。

TAP 是芯片与仿真器的接口，对芯片的任何访问都是通过 TAP 来实现。TAP Controller 通过 TMS 控制信号和 TCLK 时钟驱动信号实现状态转换，其状态转换机如图 2 所示[1]。状态转换机共有 16 个状态，每一个状态在 TCLK 上升沿根据 TMS 信号的高低电平来决定是否进入下一个状态。

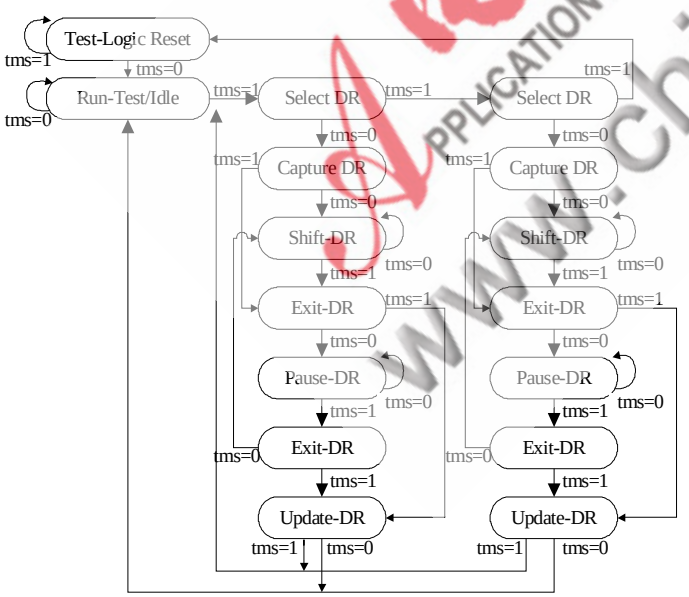

图 2 TAP 状态机

通过 TAP 访问数据寄存器(DR)的步骤为：(1)通过指令寄存器(IR)选择待访问的数据寄存器；(2)指定的数据寄存器连接在 TDI 和 TDO 之间；(3)在时钟信号 TCLK 的驱动下，由 TDI 实现新数据输入，由 TDO 实现数据输出[3]。

2 ARM920T 调试系统

ARM920T 处理器与调试相关的模块有 ARM CPU core(提供调试的硬件支持)、Embedded ICE(产生调试中断，设置断点和观察点)和 TAP Controller 等。

ARM920T 常用扫描链[2]有：

Scan chain 0，长度为 184 bit，可实现芯片连接检查和芯片内部逻辑测试。

Scan chain 1，长度为 67 bit，其中包括 32 bit 数据位、32 bit 指令位和 3 bit 控制信号。

Scan chain 2，可访问 EmbeddedICE 中的硬件寄存器。

Scan chain 3，长度用户自定义，可以访问外部边界扫描链。默认使用的扫描链。

Scan chain 6，包括 32 bit 数据位、7 bit 地址位、1 bit 读写控制位，可对 ETM9 中的寄存器编程。

ARM920T 中常用指令有：

IDCODE(b1110)：主要用来读取 CPU ID 号。

SCAN_N(b0010)：主要用来实现不同扫描链的选择，ARM920T 默认选择扫描链 3。

EXTEST(b0000)：将扫描链置于外部测试模式。

INTEST(b1100)：将扫描链置于内部测试模式。

RESTART(0100)：使 ARM920T 处理器由调试态返回正常运行态。

3 烧写系统实现

NAND Flash 烧写系统分为硬件系统和软件系统。硬件系统负责 JTAG 协议转换，实现对 TAP 的硬件控制；软件系统负责 JTAG 工作时序的模拟以及 TAP 的软件控制，是烧写系统的核心。

3.1 硬件实现

一般 JTAG 仿真器并不具有 Flash 烧写功能，且其价格比较昂贵，因此文中采用了目前较为流行且比较简单的 WIGGLER 小板，实现 JTAG Flash 在线烧写的硬件支持。这种 WIGGLER 小板是一种简易并口 JTAG，可方便地实现并口对 TAP 的直接控制。硬件原理图如图 3 所示。其中 74HC244 是一款三态缓冲器，其作用是实现电平转换。

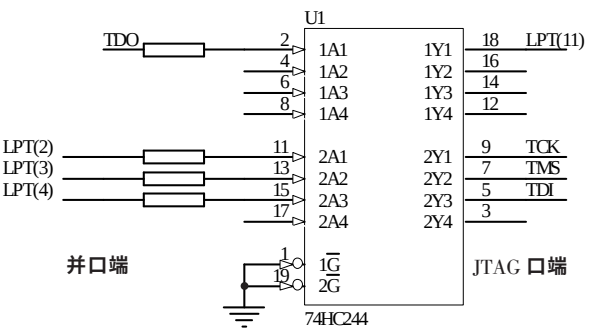

图 3 JTAG 硬件原理图

3.2 软件实现

软件系统总体上分为4个层次：并口驱动层、JTAG控制层、数据处理层以及应用程序层。软件的层次结构如图4所示。

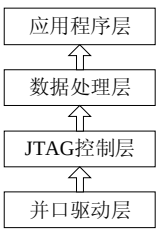

图4 软件层次结构

并口驱动层实现软件最底层的操作，本软件基本的思想就是通过对PC机上标准并口的直接操作实现对TAP的控制，从而达到观测和控制芯片的目的。JTAG控制层是整个软件的关键部分，它利用并口驱动层底层操作接口，实现TAP操作和状态机不同状态的循环控制。数据处理层相对于底层和上层的功能，可视为数据处理缓冲层，这一层并没有牵扯到任何底层的操作，仅是为了方便应用程序的实现，定义了一些关键的数据结构和数据处理函数。应用程序层是软件核心功能实现层，主要实现了NAND Flash工作时序的软件模拟以及有关的读、写及擦除等操作。

3.2.1 Linux下并口操作

Linux程序运行在保护模式下，不能直接对并口进行操作，可通过函数调用ioperm(unsigned long port,unsigned long num, bool on_off)来获得并口的访问权。参数port代表要访问并口的地址，在程序中共定义了三个并口地址：#define LPT1 0x378、#define LPT2 0x278和#define LPT3 0x3bc；参数num代表连续的端口数目，一般包括数据寄存器端口、控制寄存器端口和状态寄存器端口；逻辑变量on_off代表对端口操作方式，1代表打开0代表关闭。并口可用性可通过向端口写入数据再读回数据的方式来检查，读回的数据如果和写入的数据相同则端口可用。并口驱动层提供的访问接口有：

```
int Getvalidppt(void);                    //取得可用并口地址
void Setpptcompmode(void);                //设置并口工作模式
```

此外还有两个宏定义：

```
#define Outputppt (value) outb (unsigned long validPort , value)    //并口数据输出
#define Inputppt () inb ((unsigned long) (validPpt+0x1))            //并口数据读入
```

3.2.2 JTAG控制层

JTAG控制层主要实现TAP CONTROLLER控制，其中涉及TCK、TMS、TDI、TDO 4个控制信号和状态机的实现。输出信号控制接口由如下宏实现：

```
#define JTAG_SET(value) Outputppt(value)
```

其中value为输出数据，组合模式为TDI|TMS|TCK，TDI、TMS、TCK分别有两种状态，如：TDI_H、TDI_L，TMS_H、TMS_L，TCK_H、TCK_L，分别代表三种信号的高低电平。

输入信号(TDO)接口由如下宏实现：

```
#define JTAG_GET_TDO () ((Inputppt ()& (1 <<7)) ? LOW:HIGH )
```

TDO输出信号与状态寄存器第7位相连，此位使用了反相器，故在读入数据时需要取反。

JTAG控制层利用TAP Controller状态控制机主要实现数据的输出与输入、指令的输入、CPU ID号的读取等功能。主要的接口函数有：

```
void JTAG_Shiftdrstate(char *wrDR, char *rdDR);
                                          //同时实现数据输出与读入
void JTAG_Shiftdrstatenotdo(char *wrDR);
void JTAG_Shiftirstate(char *wrIR);       //指令输出
void JTAG_Readid(void);                   //读取CPU ID
```

访问指令寄存器的状态转换流程为：

Run-Test/Idle->Select-DR->Scan->Select-IR->Scan->Capture-IR->Shift-IR->Exit-IR->Update-IR->Run-Test/Idle

数据寄存器由指令寄存器中的当前指令决定，访问数据寄存器的状态转换流程为：

Run-Test/Idle->Select-DR->Scan->Capture-DR->Shift-DR->Exit-DR->Update-DR-> Run-Test/Idle

函数JTAG_ShiftDRState()同时实现数据读入、读出，其程序流程图如图5所示。

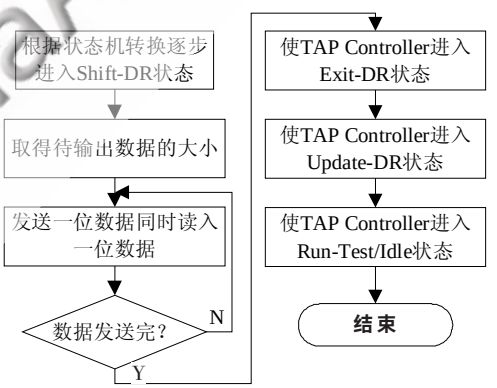

图5 数据输出与读入

3.2.3 数据处理层

边界扫描单元在使用前需要初始化，边界扫描单元的数目即为边界扫描链的长度，s3c2410处理器的边界扫描链的长度为426。处理器的每个引脚都对应一个边界扫描单元，每个引脚可视为边界扫描单元的索引，s3c2410处理器有272个引脚。对边界扫描单元的初始化即是对处理器引脚赋初值。初始化数据放在边界扫描链数组中，有如下定义：

```
char outcelldata[SC2410_MAX_CELL_INDEX+2];
char incelldata[SC2410_MAX_CELL_INDEX+2];
```

数组outcelldata[]存放待输出数据，incelldata[]存放读

入数据,数组的每个单元对应一个边界扫描单元。其中SC2410_MAX_CELL_INDEX为s3c2410处理器边界扫描单元的数目426。

s3c2410处理器数据宽度为32位,地址线27位,为便于数据、地址的统一处理定义如下3个数组:

```
int   dataoutcellindex[32];
int   dataincellindex[32];
int   addrcellindex[27];
```

数据输出与读入对应不同边界扫描单元,如:DATA0读入对应的扫描单元索引为100,输出对应的扫描单元的索引为99。将数据输出扫描单元的索引组合到具有32个元素的数组(dataoutcellindex)中,便于数据输出扫描单元的引用;将数据读入扫描单元的索引组合到具有32个元素的数组(dataincellIndex)中,便于数据读入扫描单元的引用;将地址输出扫描单元的索引组合到具有27个元素的数组(addrcellindex)中,便于地址输出扫描单元的引用。比如数据位DATA0要输出低电平,数组引用方式如下:

```
outcelldata [dataOutCellIndex[0]]=LOW;
```

从DATA0读入一位数据,数组的引用方式如下:

```
incelldata[dataInCellIndex[0]]=JTAG_GET_TDO();
```

数据处理层主要接口函数有:

```
void  SC2410_Initcell(void);     //边界扫描单元初始化
void  SC2410_Setpin(int  index,  char  value);
                                 //处理器引脚电平的设置
char  SC2410_Getpin(int  index);   //引脚信号的读入
void  SC2410_Setaddr(U32  addr);     //设置地址数据
void  SC2410_Setdatabyte(U8  data);    //写字节数据
U8  SC2410_Getdatabyte(void);          //读字节数据
```

3.2.4 应用程序层

应用程序层主要实现NAND Flash的读、写及擦除等上层操作。以K9F1208为例,NAND Flash一般的操作流程是:先向Flash芯片发操作命令,再发操作地址,如果Flash芯片准备就绪再进行数据的读/写或芯片的擦除等操作。K9F1208主要控制信号有:CLE(芯片命令锁存,高电平有效)、ALE(地址锁存,高电平有效)、WE(芯片写操作,低电平有效)、RE(芯片读操作,低电平有效)、CE(芯片使能)、R/B(芯片状态指示,高电平代表芯片就绪,低电平代表芯片忙)、IO(0~7)数据输入/输出端口。程序主要接口函数有:

NF_CMD():实现Flash写命令操作。

NF_ADDR():实现Flash地址输出。

NF_WRDATA():实现数据写。

NF_RDDATA():实现数据读。

参考K9F1208芯片写命令操作时序,NF_CMD()Flash写命令函数实现为:设CE片选信号有效;命令锁存信号CLE有效同时无效地址锁存信号ALE;写信号WE有效同时无效读信号RE;输出命令;最后无效WE信号实现命令锁存。其他相关函数的实现都是以软件的方式模拟NAND Flash的硬件工作时序,其实现方法与Flash写命令函数NF_CMD()相似。

3.3 测试及实验

烧写软件在Linux系统下编译成功,在命令行输入"./zjx_sjf_linux /f:interrupt.bin",烧写程序开始运行,运行界面如图6所示。其中zjx_sjf_linux是应用程序名,/f:为命令行参数,interrupt.bin为待烧写程序。

图6 程序运行界面

从图6可以看出程序能够成功运行且能够实现程序在Linux系统下的烧写。

本文研究了JTAG标准和ARM920T,介绍了NAND Flash在Linux系统下烧写系统的软硬件实现方案。硬件采用了简易并口JTAG,软件部分给出了系统的设计架构、功能模块和实现接口。并口JTAG烧写Flash,速度有较大的限制,进一步的工作就是改善Flash的烧写速度,提高Flash烧写效率。

参考文献

[1] IEEE1149.1. IEEE standard test access port and boundary-scan architecture [S]. 2001.

[2] ARM Corp. ARM920T Technical Reference Manual. http://www.arm.com.

[3] OPEN-JTAG开发小组.ARM JTAG调试原理[Z].2007.

[4] 陆晗,潘雪增.基于ARM的JTAG调试器[J].计算应用与软件,2007,24(2).

(收稿日期:2009-04-02)