第34卷 第15期 计 算 机 工 程 2008年8月
Vol.34 No.15 Computer Engineering August 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)15—0252—03 文献标识码：A 中图分类号：TP311

基于JTAG的ARM7TDMI调试系统

许 琼
(西安科技大学电气学院，西安 710054)

摘 要：在研究JTAG标准和ARM7TDMI处理器调试模块的基础上，提出调试ARM7TDMI处理器的软硬件实现方案。采用简易JTAG接口，通过计算机并口控制测试访问端口。该软件采用3层结构，共有7个模块。该文分析层次、模块的划分及接口的定义和实现过程。实验测试结果表明，该软件具有良好的实用性、可调试性。

关键词：边界扫描；测试访问端口；嵌入式系统

ARM7TDMI Debugging System Based on JTAG

XU Qiong
(Electric College, Xi'an University of Science & Technology, Xi'an 710054)

【Abstract】 Based on research of JTAG standard and the ARM7TDMI debugging model, this paper proposes the hardware and software solution of debugging ARM7TDMI. The easy JTAG is introduced to control Test Access Port(TAP) by computer parallel port. The software is three-level structure and contains seven models. It discusses the level, model partition, and the definition and implementation of interfaces. Experimental results demonstrate that the software has good practicability and debuggability.

【Key words】 boundary-scan; Test Access Port(TAP); embedded system

IEEE 1149.1最初是由JTAG(Joint Test Action Group)提出的，1990年被IEEE批准并对其进行了标准化。人们通常采用JTAG来表示IEEE1149.1规范，满足IEEE1149.1规范的接口、测试。JTAG极大地推动了边界扫描技术的发展，在电子产品设计调试的各个阶段都有着广泛的应用：在单片机、ARM、DSP等处理器中，其主要用于软件调试；在CPLD、FPGA开发中，其主要用来配置PROM。

1 JTAG标准简介

1.1 JTAG基本原理

边界扫描技术(boundary-scan)的基本思想是在芯片的管脚上增加一些边界扫描寄存器单元(boundary-scan register cell)。芯片有2个状态：调试状态和运行状态。处于调试状态时，这些边界扫描寄存器将芯片和外围器件的输入/输出隔离。并且通过边界扫描寄存器，可以实现对芯片输入/输出信号的观察和控制。因为在调试状态，边界扫描寄存器将芯片和外围器件的输入/输出进行隔离。本文所述的输入/输出管脚是指靠近边界扫描积存器芯片内部里面的管脚，而不是所能看到的管脚。对于输入管脚，可以通过与之相连的边界扫描寄存器把信号加载到输入管脚上。对于输出管脚，可以通过与之相连的边界扫描寄存器“捕获”(CAPTURE)该输出管脚上的输出信号。边界扫描寄存器提供了一个便捷的方式用以观测和控制所需要调试的芯片。芯片输入/输出管脚上的边界扫描(移位)寄存器单元可以相互连接起来，在芯片的周围形成一个边界扫描链(boundary-scan chain)。一般的芯片会提供几条独立的边界扫描链，用来实现完整的测试功能。边界扫描链可以进行串行的输入/输出，通过相应的时钟信号和控制信号，就可以方便地观察和控制处在调试状态下的芯片[1]。在正常的运行状态下，这些边界扫描寄存器对芯片来说是透明的。

调试状态和运行状态的CLOCK不同。以ARM7TDMI为例，在调试状态下，ARM7TDMI由内部的调试时钟DCLK(Debug Clock)驱动。在运行状态下，ARM7TDMI由MCLK(Memory Clock)驱动[2]。DCLK比MCLK慢。

1.2 Test Access Port

IEEE1149.1定义了TAP。TAP是一个通用的端口，在芯片上可以通过TAP访问芯片提供的所有数据寄存器(DR)和指令寄存器(IR)。TAP包括4个强制信号TCK、TMS、TDI、TDO和1个可选信号TRST。

(1)TCK(Test Clock Input)，为TAP的操作提供了一个独立的、基本的时钟信号，TAP的所有操作都是通过这个时钟信号来驱动的。

(2)TMS(Test Mode Selection Input)信号，用来控制TAP状态机的转换。TMS在TCK的上升沿有效。

(3)TDI(Test Data Input)，数据输入的接口。

(4)TDO(Test Data Output)，数据输出的接口。

(5)TRST(Test Reset Input)，对TAP Controller进行复位(初始化)。

TAP是芯片与仿真器之间的接口。TAP通过TMS和TCK驱动TAP Controller，从而控制芯片内部扫描寄存器单元。TAP Controller的状态机如图1所示[1]，共有16个状态。状态的转换是由TMS信号控制。箭头上1或0表示在TCK的上升沿采集TMS的信号高低，其决定下一步转换到哪个状态。假

作者简介：许 琼(1980－)，女，助教，主研方向：嵌入式系统，电路与系统

收稿日期：2007-12-14 **E-mail：**joanrun_xu@hotmail.com

设状态是 Select-IR-Scan，如果 TMS 为 0，转到 Capture-DR；如果 TMS 为 1，则转到 Test-logic-Reset 状态。系统上电后，TAP Controller 自动进入 Test-Logic Reset 状态。

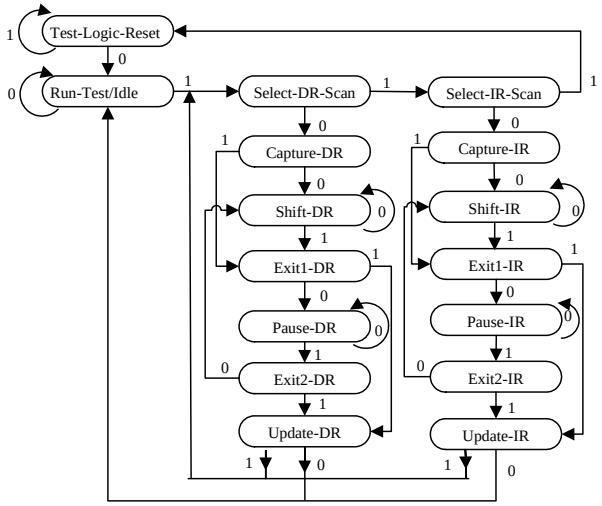


图 1　TAP Controller 状态机

2　ARM7TDMI

ARM7TDMI 处理器与调试相关的模块有：ARM CPU Main Processor Logic，包括了对调试的硬件支持；Embedded ICE-RT Logic 用来产生调试异常、设置断点和观察点；TAP Controller。

ARM7TDMI 提供了 4 条扫描链[2]：

(1)扫描链 0

长度为 113 bit，包括：32 bit 数据总线，32 bit 地址总线，内核控制信号。

(2)扫描链 1

是扫描链 0 的子集，长度为 33 bit，包括：32 bit 数据总线，BREAKPT 信号。

(3)扫描链 2

专门用来访问 EmbeddedICE-RT 内部的寄存器，长度为 38 bit。让 ARM7TDMI 进入调试状态，并设置断点、观察点。

(4)扫描链 3

可以访问外部的边界扫描链。

ARM7TDMI 常用指令：IDCODE(b1110)指令用于读取 CPU 的 ID；SCAN_N(b0010)指令的用于选择扫描链，在 TAP Controller 复位以后，默认状态下选择的是扫描链 3；BYPASS(b1111)指令将 1-BIT 长的 BYPASS 寄存器连接到 TDI 和 TDO 之间；INTEST(b1100)指令将扫描链置于内部测试模式；RESTART(b0100)指令使 ARM7TDMI 处理器从调试状态退回到正常的运行状态。

3　ARM7TDMI 调试系统

调试系统分为硬件和软件 2 个部分。硬件控制 TAP 端口。软件主要实现寄存器、内存、断点的设置和访问。

3.1　硬件实现

JTAG 仿真器的硬件主要采用 USB 或并口 2 种方案。USB 的优点是速度快，缺点是要写 firmware 和操作系统的驱动程序，调试软件调用驱动程序，间接控制 TAP。并口简易 JTAG 的优点是可以直接通过并口控制 TAP，便于定位软硬件错误，缺点是因为要通过软件来产生 DCLK，所以，速度较慢。本文采用简易 JTAG，硬件原理图见图 2。其中，LPT/2 指并口的第 2 个引脚，其他以此类推。SN74LS244 是三态缓冲器。

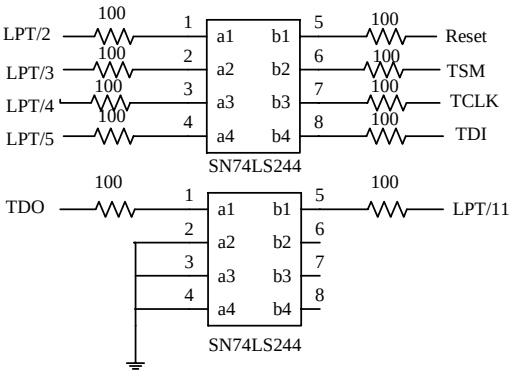


图 2　JTAG 仿真器原理

3.2　软件实现

软件架构如图 3 所示，分为 3 层。最低层是并口操作，实现对并口的读写访问。中间是 TAP 控制和状态机：TAP 控制模块实现对 TAP 端口的控制，从而改变引脚的电平变化；状态机记录当前状态和给定输入后的下一个状态。最上层是 Memory 读写(图中用 MEM 表示)、Register 读写(图中用 REG 表示)、选择扫描链(图中用 Select-SC 表示)、断点设置(图中用 Break 表示)。

3 层的关系是：上层模块调用下层模块的接口，下层模块不可以调用上层模块，同一层模块之间可以互相调用。软件运行流程如下：(1)初始化并口模块；(2)初始化 TAP Controller；(3)进入调试状态；(4)写二进制代码到内存；(5)设置断点；(6)运行程序直到下一个断点。

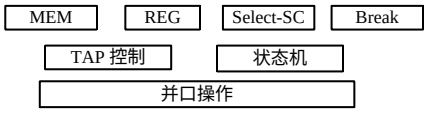


图 3　软件架构

3.2.1　Windows 下的并口操作

Windows NT 以上系统对 I/O 操作进行了保护。通过调用 WinIo 驱动程序，可以绕过 Windows 的保护机制在应用程序中直接调用 in, out 指令。WinIo 提供了用于二次开发头文件和库文件 winio.h 和 winio.lib。在调用 WinIo 的端口访问 API 之前要初始化 WinIo。并口访问接口函数如下：

```
void WriteParallPort( unsigned char data);
unsigned char ReadParallPort();
```

3.2.2　TAP 引脚的控制

对 TAP 引脚的控制通过 3.2.1 节的 2 个函数来实现。分别实现如下接口：

```
void    WriteTDI(unsigned char data);
void    WriteTMS(unsigned char data);
void    WriteTCK(unsigned char data);
void    WriteTRST (unsigned char data);
unsigned char ReadTDO();
```

TDI,TMS,TCK,TRST 对 PC 而言为输出信号，当参数 *data* 为 0 时，置信号为低电平，当参数 *data* 为 1 时，置信号为高电平。TDO 为输入信号，ReadTDO 读取 TDO 引脚信号电平高低。

3.2.3　状态机的实现

本节实现了图 1 的状态机。根据当前状态和输入决定下一个状态。有限状态机的实现方式主要有：

(1)switch/case 或 if/else

对简单的状态机，用这种方法最直接。

(2)状态表

维护一个二维状态表，横坐标表示当前状态，纵坐标表示输入，表中，一个元素存储下一个状态和对应的操作。二维表与状态机一一对应，易于维护，因此，笔者选择了二维状态表。

二维状态表用二维数组为

{{2,5},{3,5},{2,1},{3,0},{14,15},{12,7},{2,1},{6,4},{10,13},{11,13},{10,9},{11,8},{12,7},{12,7},{10,9},{12,15}}

如第 0 个元素{2,5}对应图 1 的状态 EXIT2_DR，TMS 输入分别为 0 和 1 时，下一个状态分别为 2(表示状态 SHIFT_DR)和 5(表示状态 UPDATE_DR)[3]。

实现接口函数为 int NextState(unsigned char TMS)。

3.2.4 Register 读写

本节实现寄存器的读/写。寄存器的读/写是调试软件必要模块之一。从图 1 可以看出，访问数据寄存器的状态转换流程为

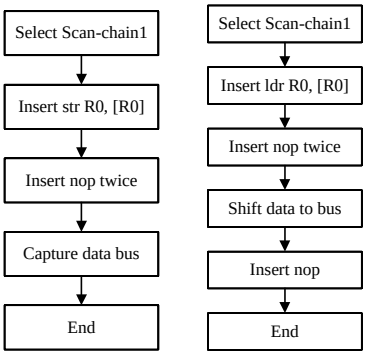
Run-Test/Idle->Select-DR-Scan->Select-IR->Scan-> Capture-IR->Shift-IR->Exit1-IR->Update-IR->Run-Test/Idle

可以访问的数据寄存器由指令寄存器中的当前指令决定。访问指令寄存器的流程为

Run-Test/Idle->Select-DR-Scan->Capture-DR->Shift-DR->Exit1-DR->Update-DR->Run-Test/Idle

ARM7TDMI 可以通过扫描链 1 访问通用寄存器。如用指令 STR R0, [R0]将寄存器 R0 值存储到地址为 R0 的内存中去。

在 ARM7TDMI 处于调试状态时，ARM7TDMI 和内存是隔离的，该指令访问是的扫描寄存器单元而不是内存，流程见图 4(a)。类似可以用 LDR R0, [R0]写 R0 寄存器，其流程见图 4(b)。



(a)读 R0 流程　　(b)写 R0 流程

图 4　读、写 R0 流程

其他寄存器的读/写在 R0 基础上，实现各自的访问。读流程如下：(1)选择扫描链 1；(2)读 R0 并且备份；(3)将要读的寄存器存入 R0；(4)读 R0 就是要读寄存器的值；(5)恢复 R0 寄存器。写流程类似。

ARM7TDMI 寄存器的访问接口函数如下：

int ReadReg(int index, unsigned int *pData);

int WriteReg(int index, unsigned int nData)

3.2.5 选择扫描链

ARM7TDMI 有 4 条扫描链。选择扫描链的流程：(1)装载 SCAN_N 指令 b0010 到指令寄存器。(2)在 CAPTURE-DR 状态，b1000 将被捕获到扫描链选择寄存器中。(3)在 SHIFT-DR 状态，将扫描链号码(0-3)通过 TDI 输入到扫描链选择寄存器中。(4)在 UPDATE-DR 状态，被选择的扫描链将被连接到 TDI 和 TDO 之间[3]。

实现接口函数为 int SelectScanChain(int cNum)。

3.2.6 Memory 读/写

内存读/写是调试软件的基本功能。调试软件一般将二进制代码先写到内存，然后运行到下一个断点。通过扫描链 1 插入一条指令的时候，如果将 BREAKPT 置 1，意味这条指令的下面一条指令将在 MCLK 的驱动下执行，执行完后，自动返回调试状态。内存访问要在 MCLK 下运行，利用 BREAKPT 置 1 的特点，实现内存访问。读内存的流程为：将要访问的地址写到寄存器 R0，用指令 LDR R1, [R0]将该地址的值装载到 R1，再读出 R1。写内存的流程：将要访问的地址写到 R0，数据写到 R1，执行 STR R1, [R0]指令[4]。

接口函数如下：

int ReadMem(unsigned int nAddr, unsigned char *pBuf, int len);

int WriteMem(unsigned int nAddr, unsigned char *pBuf, int len);

3.2.7 断点设置

在调试软件时，调试状态和运行状态之间不断切换。ARM7TDMI 从运行状态切换到调试状态的方式——控制 DBGRQ 信号、断点、观察点。ARM7TDMI 退出调试状态，使用 RESTART JTAG 指令。断点分为硬件断点和软件断点。硬件断点监视地址总线，一旦匹配则强制 ARM 核进入调试模式，硬件断点可以用于任意地址。软件断点则识别从任何地址取出的数据，一旦匹配才会进入调试模式。可以设置任意多个，但是一般只能设置在可读写的区间[5]。硬件断点设置比较简单。软件断点设置的流程见图 5。

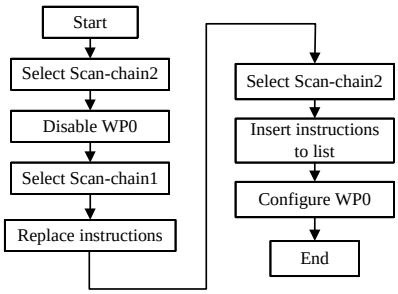


图 5　软件断点设置流程

接口函数如下：

int SetSoftBreakPoint(unsigned int nAddr);

int SetHardBreakPoint(unsigned int nAddr);

int ClearBreakPoint(unsigned int nAddr);

4　结束语

本文研究了 JTAG 标准和 ARM7TDMI，提出了调试系统的软硬件解决方案。硬件采用简易 JTAG 接口，给出了原理图。设计了软件架构，讨论了各模块，对接口进行了设计和实现。实现了寄存器的读/写、内存读/写、设置断点等功能。本软件架构有良好的移植性和扩展性，可以快速地移植、应用到其他处理器平台中。

参考文献

[1] IEEE. IEEE1149.1-2001 IEEE Standard Test Access Port and Boundary-scan Architecture[S]. 2001-09: 3-32.

[2] ARM Ltd.. The ARM7TDMI Debug Architecture[Z]. 2004.

[3] ARM Ltd.. ARM7TDMI Technical Reference Manual[Z]. 2004.

[4] OPEN-JTAG. ARM JTAG 调试原理[EB/OL]. (2004-09-02). http://www.embedworld.com.

[5] 陆　晗, 潘雪增, 平玲娣. 基于 JTAG 的 ARM 调试实现[J]. 计算机应用与软件, 2007, 24(2): 137-139.