多核DSP软件调试环境研究与设计

王 品[1],于 莹[2],苗政民[1],贺红卫[1]
(1.中国兵器科学研究院,北京 100089;
2.中国航天科工集团第二研究院706所,北京 100039)

摘 要:针对武器装备嵌入式系统向自主化、智能化、小型化、低功耗快速发展的趋势,介绍了装备研制对自主多核处理器及其软件调试环境的迫切需求,分析了嵌入式系统远程调试的基本原理和特点、JTAG标准和边界扫描技术。以自主同构8核数字信号处理器为目标平台,基于目标平台JTAG控制器之间的菊花链连接方式,提出了面向该目标平台的软件调试环境设计方案,讨论了USB接口仿真器软硬件设计和多线程调试代理软件设计等关键技术。实现的软件调试环境能够在调试主机上对目标平台进行指令级和源码级交叉调试,解决了目标平台缺乏配套软件调试手段的实际问题,为目标平台在武器装备上的推广应用提供了有力支撑,对其他面向多核处理器的调试环境设计具有参考价值。

关键词:多核处理器;仿真器;嵌入式系统;远程调试;软件调试环境;JTAG
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2021)01-0110-06
doi:10.3969/j.issn.1673-629X.2021.01.020

Research and Design of Software Debugging Environment for Multi-core DSP

WANG Pin[1],YU Ying[2],MIAO Zheng-min[1],HE Hong-wei[1]
(1.Academy of Chinese Weapon Science,Beijing 100089,China;
2.Institute 706,Second Academy of China Aerospace Science and Industry Corporation,Beijing 100039,China)

Abstract:In view of the trend of rapid development toward autonomy,intelligence,miniaturization and low power consumption for embedded system of weapons and equipment,the urgent demand for autonomous multi-core processor and its software debugging environment in equipment development is introduced,and the basic principle and characteristics of remote debugging of embedded system,JTAG standard and boundary scanning technology are analyzed. With an autonomous homogeneous eight-core DSP as the target platform,based on the daisy-chain connection between JTAG controllers of the target platform,we propose the design scheme of software debugging environment for the target platform and discuss the key technologies such as software and hardware design of USB interface emulator and software design of multi-thread debugging agent. The software debugging environment is able to debug the target platform at the instruction level and source level on the host,solving the problem that the target platform lacks the means of supporting software debugging,which provides a strong support for application and promotion of the target platform in weapons and equipment and has reference for other debugging environment design oriented to multi-core processors.

Key words:multi-core processor;emulator;embedded system;remote debug;software debugging environment;JTAG

0 引 言

数字信号处理器(digital signal processor,DSP)具有灵活、稳定、重复性好、可大规模集成和易于实时实现等优点,被广泛应用于语音、通信、图像处理、声纳、生物医学仪器等诸多领域,在防空反导、机动突击、火力打击、光电侦察、指挥控制等各类武器装备中也发挥着重要作用[1]。

多核技术是提升处理器计算能力的重要途径,随着武器装备嵌入式系统向自主化、智能化、小型化、低功耗方向发展,对高精度、高性能和安全可控计算需求不断提高,自主多核处理器在装备研制中得到越来越广泛的应用。

目前,国内在核心电子器件领域的技术水平有了长足进步,已经研制出多款多核处理器,对相应的软件开发调试环境提出了迫切需求。提供包括仿真器在内的、与处理器相配套的软件调试环境,是自主多核处理

收稿日期:2020-03-08 **修回日期**:2020-07-10
基金项目:"核高基"国家科技重大专项(2015ZX01020301)
作者简介:王 品(1984-),男,硕士,副研究员,研究方向为嵌入式系统。

器能够推广应用与深入发展的前提[2]。

1　远程调试原理与远程串行协议

1.1　嵌入式系统远程调试原理

相对于通用计算机,嵌入式系统的软硬件资源有限,通常无法完成本地自主调试,需要借助软硬件资源丰富的通用计算机,使用远程调试的方式进行软件调试[3]。

一个典型的嵌入式软件远程调试系统包括三个部分:调试主机、仿真器(调试协议转换器)和目标平台。调试主机一般为通用计算机,通过某种接口(如 USB 接口、网口、并口)与仿真器相连,仿真器通过 JTAG 接口连接目标平台,以此达到调试主机与目标平台进行调试信息交互的目的。调试主机运行调试器(如 GDB)和调试代理软件,并为开发调试人员提供图形化操作和观察界面[4]。

远程调试系统组成如图 1 所示。

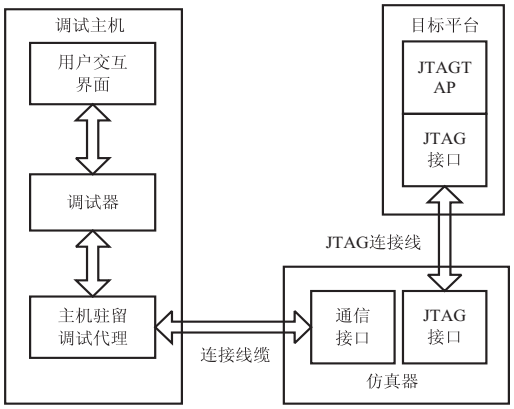

图 1　远程调试系统组成

与本地调试相比,远程调试具有以下特点[5]:

(1)调试环境和被调试程序运行在不同的计算机上,通过某种通信方式将调试主机和目标平台相连,避开了目标平台上软硬件资源不足的限制,为用户提供强大的调试功能。

(2)目标平台无需操作系统支持。目标平台上即使运行操作系统,也主要是用于嵌入式应用的支撑,而不是用于目标平台的软件开发调试环境。

(3)本地调试只能调试相同指令集的程序,而远程调试可以调试与主机不同指令集的程序,这一特点也被称为交叉调试。

(4)由于调试环境和被调试程序的分离,不可避免地引入通信开销,对调试效率产生一定影响。

1.2　GDB 远程串行协议

GDB 是 GNU 开源组织发布的源码级调试器,支持多种处理器架构和编程语言,在调试代理的配合下,实现断点、单步执行、查看变量、查看寄存器、查看内存、查看堆栈等远程调试功能[6]。

GDB 与调试代理通过远程串行协议(remote serial protocol,RSP)实现调试信息交互。RSP 是一种基于 ASCII 码字符的通信协议,包括客户端(RSP client)和服务器端(RSP server),通常以 GDB 作为客户端,以调试代理作为服务器端,两者可以位于相同或不同的计算机上,通过网络 Socket 或 RS232 串口等方式通信。

RSP 以数据包的形式传输数据,一个数据包包含数据信息和校验码两个部分。数据信息是一串 ASCII 码字符,以“$”字符作为起始标志,以“#”字符作为结束标志;数据信息后面跟随长度为 16 位的校验码,该校验码的值是数据信息中所有字符的 ASCII 码数值相加后除以 256 的余数,用 2 个十六进制字符表示。

接收方接收到完整的数据包后,使用校验码对数据信息进行校验,如果接收到的数据信息校验计算无误,返回“+”字符通知发送方数据包接收完成;如果校验出错,则返回“-”字符,通知发送方重新发送。发送方通过接收方返回的响应字符判断数据包发送是否成功。

如果调试代理暂不支持某个 RSP 调试指令,可以返回字符串“$#00”通知 GDB,GDB 会尝试使用其他 RSP 调试指令来代替不支持的调试指令完成同样的功能。然而,要实现 GDB 的调试功能,调试代理必须支持读寄存器的值(g)、修改寄存器的值(G)、读内存的值(m)、修改内存的值(M)、恢复运行(c)、单步执行(s)等 RSP 调试指令[7]。

2　JTAG 标准

IEEE 1149.1 标准由联合测试行动组(joint test action group,JTAG)提出,因此也被称为 JTAG 标准。JTAG 标准最初用于芯片测试,现在也常用于对芯片上的软件进行调试。

JTAG 标准中提出了边界扫描(boundary-scan)概念,其基本思想是在芯片的输入输出管脚上放置一个移位寄存器,称为边界扫描寄存器单元(boundary-scan register cell)。芯片正常运行时,这些边界扫描寄存器单元对芯片来说是透明的,不产生任何影响;当芯片处于测试状态时,边界扫描寄存器单元将芯片和外围的输入输出隔离,芯片的输入管脚可以通过与之相连的边界扫描寄存器单元把数据加载到该管脚中去,输出管脚可以通过与之相连的边界扫描寄存器单元“捕获(capture)”该管脚上的输出信号。

此外,通过边界扫描寄存器单元的相互连接,在芯片的周围形成一条边界扫描链(boundary-scan chain),在适当的时钟信号和控制信号驱动下,测试数

据可以在边界扫描链上串行输入和输出。通过边界扫描寄存器单元和边界扫描链,能够对芯片的输入输出信号进行观察和控制,实现芯片测试和软件调试功能。一种芯片可以同时提供几条独立的边界扫描链,以满足不同的测试要求。

在 JTAG 标准中,定义了两类寄存器:数据寄存器(data register,DR)和指令寄存器(instruction register,IR)。边界扫描链即是一种重要的数据寄存器,用于控制和观察芯片的输入输出;IR 用来实现对 DR 的控制,例如在芯片可供选择的全部边界扫描链中,通过 IR 指定一条边界扫描链作为访问对象[8]。

JTAG 设备内部定义了一个状态机,称为 TAP 控制器。TAP 控制器的状态通过 TCK 和 TMS 两个输入引脚进行控制,通过 TAP 控制器的状态变化,完成 JTAG 指令寄存器的输入和数据寄存器的输入输出,从而对芯片内部的处理器和外设进行控制和状态获取,在调试主机和仿真器软件的配合下,实现调试功能。标准的 JTAG 接口必须至少具有 TCK、TDI、TDO 和 TMS 四个引脚,其中 TCK 为 TAP 控制器提供时钟信号驱动,TMS 控制 TAP 控制器的状态转换,TDI 为数据输入引脚,TDO 为数据输出引脚。此外,还有一个可选引脚 RST,用于 JTAG 设备的状态复位[9]。

3 多核 DSP 调试环境设计方案

调试环境的目标平台为“核高基”国家科技重大专项支持的自主同构 8 核 DSP。为了节约片上资源,将 8 个 DSP 内核中的 JTAG 控制器用菊花链(daisy chain)的方式连接起来,通过一个 JTAG 接口进行控制,如图 2 所示。

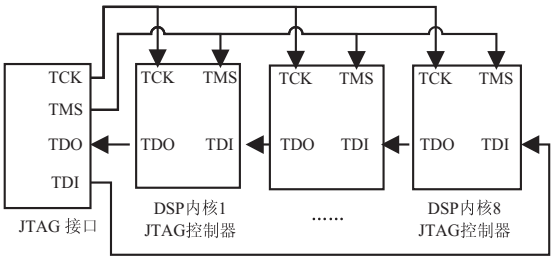

图 2 目标平台 JTAG 连接

为了对这一设计进行支持,每个 DSP 内核的 JTAG 控制器都支持 bypass 指令。当某个 DSP 内核不是要调试的目标内核时,通过 bypass 指令,将位数为 1 的 bypass 寄存器连接在数据移入接口 TDI 和数据移出接口 TDO 之间,为数据移入移出提供一条长度最短的通过路径,以便调试数据快速地通过该 DSP 内核,到达目标 DSP 内核或调试主机[10]。

目标平台设计了自主的调试协议,无法使用现有的任何一种软件开发环境进行软件调试。该文针对目标平台的特点和调试协议,提出了一种调试环境设计方案。调试环境交互界面选择安装 CDT 插件的 ECLIPSE 平台,主要功能是提供友好的开发调试界面,接收用户调试命令输入,并显示调试结果,如目标机寄存器内容、内存内容、堆栈内容及断点信息等;采用 GDB 作为调试器,ECLIPSE 平台集成针对 DSP 编译形成的 GDB,ECLIPSE 平台与 GDB 通过标准的 Machine Interface(MI)协议通讯[11];调试主机驻留唯一的调试代理软件,调试代理通过 8 个线程使用 8 个端口分别与 8 个调试器建立连接,利用信号量解决通过单一 JTAG 接口访问 8 核 DSP 的互斥问题;调试代理利用调试主机的 USB 接口和仿真器进行通信。仿真器驻留 8 核 DSP 调试协议解析程序,该程序作为宿主机调试代理和被调试程序之间的媒介存在[12]。

用户通过 ECLIPSE 平台输入调试命令,ECLIPSE 平台根据当前处于活动状态的工程,区分输入命令的目标平台,将命令转换为若干 MI 协议指令,发送给相应 GDB,GDB 将 MI 协议指令转换为若干 RSP 指令,通过 Socket 连接发送给调试代理,调试代理根据接收到指令的线程区分指令的目标内核,将目标内核信息添加到指令中,封装成为多核调试协议指令,通过 USB 接口发送给仿真器软件;仿真器软件接收多核调试指令,根据指令中的目标内核信息,区分指令对应的内核,将指令解析为指定 DSP 内核的调试指令+其余 7 个 DSP 内核的 bypass 指令,按照 JTAG 状态机的时序要求,通过 GPIO 端口将解析后的指令数据扫描到目标平台中,同时以移位方式获取返回结果;最后仿真器将调试结果通过 USB 接口发送给调试主机,经 ECLIPSE 解析后以图形化方式呈现给用户[13]。整个远程调试系统结构如图 3 所示。

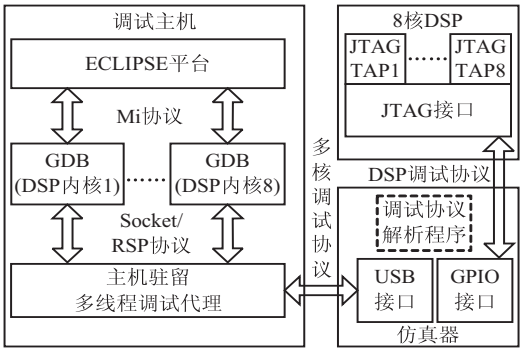

图 3 多核 DSP 远程调试系统结构

4 调试代理软件设计

根据多核 DSP 调试环境设计方案,调试代理创建并启动 8 个线程,使用 8 个端口分别与面向每个 DSP 内核的 GDB 建立 Socket 连接。其中一个连接接收到完整的 RSP 调试指令后,将其转换为相应的多核调试

协议指令。为避免与其余内核的调试相冲突,需申请全局信号量,获取到全局信号量后,通过 USB 接口向仿真器发送多核调试协议指令,等待仿真器软件返回执行结果;接收到执行结果后,将结果封装为 RSP 数据包,通过 Socket 连接返回给相应的 GDB;最后释放全局信号量,一条 RSP 指令执行完毕,等待接收下一条 GDB 发送的 RSP 指令。

DSP 内核可能由于断点、异常等原因,由运行状态转变为挂起状态。调试代理软件通过内部变量记录每个内核的状态,如果一个连接没有接收到 RSP 指令,还应判断相应内核是否处于运行状态,如果内核处于运行状态,则需获取全局信号量,通过 USB 接口向仿真器发送内核状态查询指令,如果该内核已经转变为挂起状态,调试代理主动向 GDB 发送内核已挂起信号,最后释放全局信号量。

其执行流程如图 4 所示。

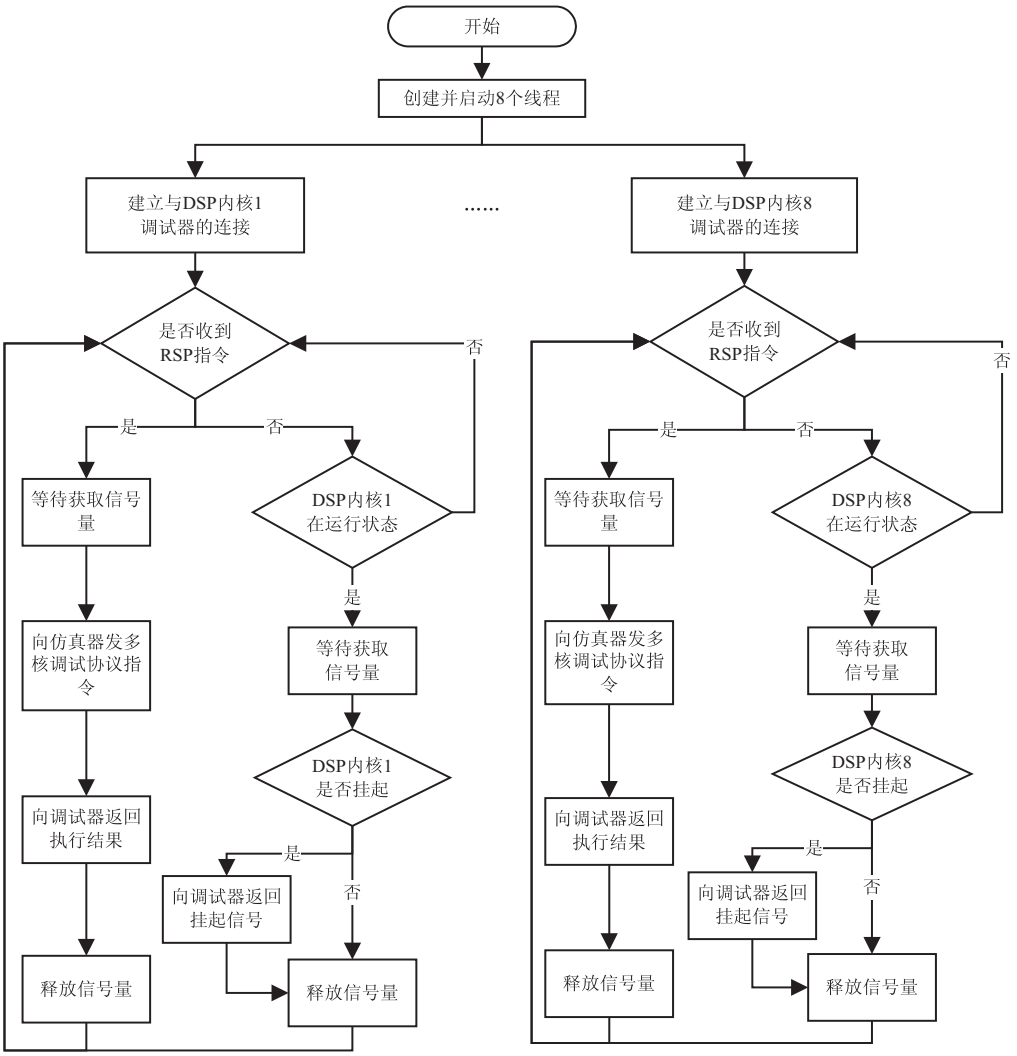

图 4　调试代理执行流程

5　仿真器设计

5.1　硬件设计

选用 ARM 架构 AT91SAM7X512 处理器作为仿真器的主控芯片,利用其内部集成的 USB 2.0 全速设备端口作为与调试主机连接的 USB 接口。主控芯片引出 5 个 GPIO 引脚,与 IEEE1149.1 标准中的 TDI、TDO、TMS、TCK、RST 五个接口信号相对应,通过缓冲处理芯片 HC244SJ 与目标平台对外引出的标准 JTAG 接口连接。控制芯片外接 3 个 LED 指示灯,分别与 TDI、TDO 及 RST 接口对应的 GPIO 引脚串联,当 JTAG 接口有信号传输时,相应的 LED 灯就会闪烁,起到指示作用,以便实时观察调试过程中仿真器与目标平台之间的通信过程。仿真器硬件设计如图 5 所示。

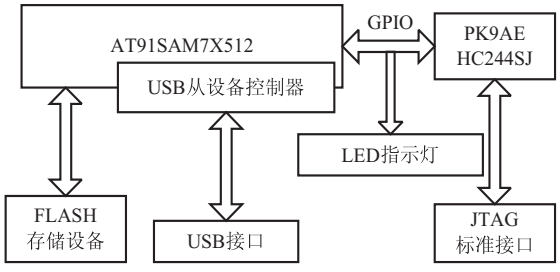

图 5　仿真器的硬件设计

仿真器软件经烧写后存储在 FLASH 中。为了提高仿真器的调试速度,将仿真器软件搬移到 SRAM 中运行,设计了单独的启动引导程序(BootLoader)。BootLoader 同样烧写在 FLASH 中,是仿真器加电后执行的第一段代码,在完成 CPU 和相关硬件的初始化之后,再将仿真器软件所在的 FLASH 地址的内容,复制到 SRAM 中,然后跳转到仿真器软件程序的入口点地址,将仿真器控制权交给仿真器软件[14]。

5.2 软件设计

仿真器软件主要包括 USB 端点中断处理和调试协议解析处理两个功能模块。

USB 端点中断处理模块通过对端点中断的处理,将发送缓冲区的数据发送到调试主机,将从调试主机收到的数据写入接收缓冲区;调试协议解析处理模块完成多核调试指令的接收、解析和处理,JTAG 状态机的驱动,以及 DSP 调试协议的实现等仿真器的核心功能。

5.2.1 USB 端点中断处理

仿真器作为 USB 设备端,与调试主机通过 USB 接口连接,遵循 USB 主从通信协议。每一个 USB 设备在正常工作前必须完成主机对它的配置过程,即总线枚举。USB 设备在总线上共有六种状态:接入态、加电态、默认态、地址态、配置态和挂起态。当调试主机与仿真器连接后,开始通过端点中断进行设备的总线枚举过程,仿真器完成总线枚举后,从配置态进入挂起态,等待来自调试主机的多核调试协议指令。

在 USB 设备端存在以下几类中断:帧起始中断、设备恢复中断、设备挂起中断和端点中断,来自调试主机的指令引起的中断是端点中断。端点是 USB 主机和设备的数据流终点,每个端点仅支持单一方向的数据流,按照数据流方向分为输入端点和输出端点,来自主机的数据流止于设备上的输出端点,发往主机的数据流则始于设备上的输入端点[15-16]。

图 6 为端点中断处理流程。

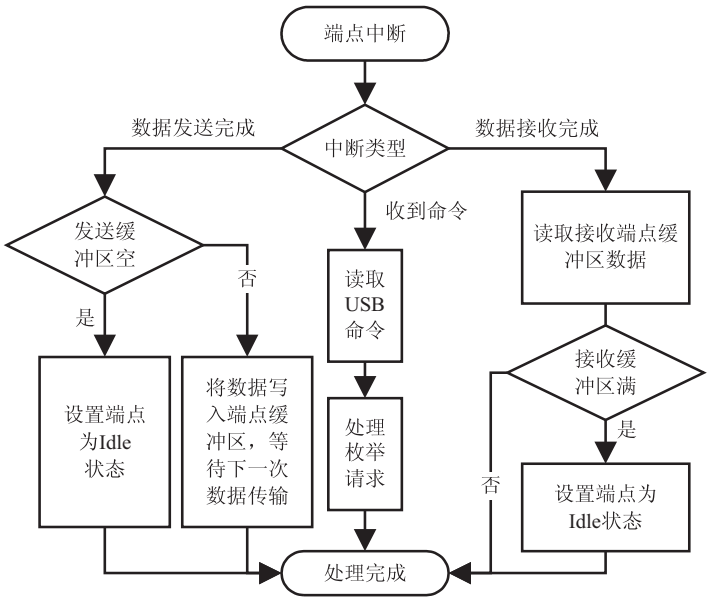

图 6 仿真器端点中断处理流程

根据引起中断的原因,端点中断又分为数据包接收中断、数据包发送完成中断和 SETUP 命令包接收中断,通过 USB 设备控制器的控制状态寄存器来区分中断类型。如果中断类型为数据包接收中断,说明从调试主机收到了一个数据包,此时应将数据包放入接收缓冲区,判断是否还有要接收的数据,如果已接收数据的数量不小于预期接收数据数量,说明本次数据接收任务已完成,将输出端点中断禁用,设置数据接收完成标识变量,通知主程序数据接收完成。如果中断类型为数据包发送中断,说明上一个数据包发送已完成,此时应判断发送缓冲区中是否还有数据需要发送,如果有,则将数据加入到发送队列;如果没有数据要发送,说明本次数据发送任务已完成,将输入端点中断禁用,设置数据发送完成标识变量,通知主程序数据发送完成。如果中断类型为 SETUP 命令包接收中断,说明接收到主机的 SETUP 命令,SETUP 命令主要用于 USB 主机对设备进行配置和控制,例如总线枚举等,不涉及仿真器核心功能的实现。

5.2.2 调试协议解析处理

调试协议解析处理模块是仿真器软件的主程序,其处理流程如图 7 所示。在完成 USB 和 GPIO 等设备初始化与总线枚举后,仿真器与调试代理进行通信,等待接收并执行多核调试协议指令。调试代理先发送 2 个字节的指令长度,然后发送指令的内容。仿真器通过端点中断接收到指令长度后开始等待接收此长度的指令内容。接收完毕后对指令进行解析,识别指令的

目标内核,将多核调试协议指令转化为目标 DSP 内核调试指令,与其余内核的 bypass 指令进行组合,通过 TDI 引脚按照 TAP 状态机的时序要求移入内核执行。DSP 内核调试指令执行完成后,仿真器将从 TDO 引脚移出的处理结果返回给调试代理,一条多核调试协议指令执行完毕,等待接收下一条调试代理发送的指令。

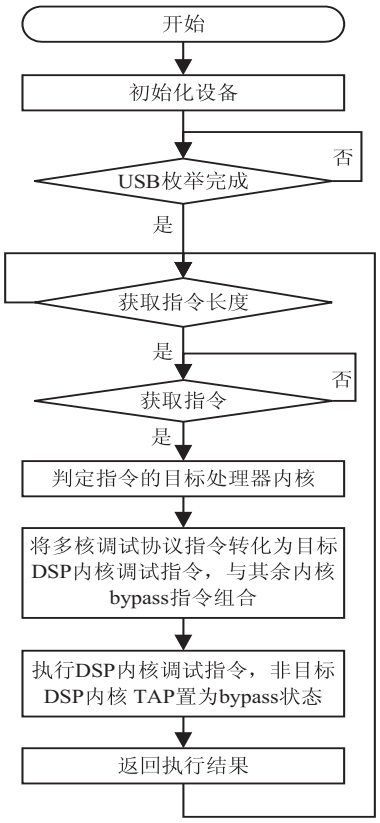

图 7　仿真器软件流程

6　结束语

该文提出的面向自主 8 核 DSP 调试环境的设计方案,解决了目标平台缺乏配套软件调试手段的实际问题。方案采用自主研发的软硬件和开源软件,具有功能完备、性能良好、安全可控、灵活易扩展等优点。该方案能够集成 GCC 等编译链接工具,结合 ECLIPSE 平台自身的功能,即可构成一个图形化的软件集成开发环境,实现一体化的工程管理、编译链接和远程调试功能,对其他自主同构或异构多核处理器软件开发环境的设计具有一定的参考价值。

参考文献:

[1] 马晓东,李冰琪,魏　鹏,等. DSP 技术发展与应用研究综述[J]. 电子世界,2018(24):46-47.

[2] 周　楠,胡　娟,胡海明. 多核处理器发展趋势及关键技术[J]. 计算机工程与设计,2018,39(2):393-399.

[3] 李志丹. 嵌入式软件调试方法研究[J]. 计算机与数字工程,2012,40(7):157-159.

[4] 章　辉,桥　井,高　桥,等. 基于仿真器的源码级调试器设计与实现[J]. 计算机工程与设计,2010,31(8):1685-1688.

[5] 殷绍剑,雷　航,詹瑾瑜. 嵌入式远程调试原理研究与实现[J]. 计算机应用与软件,2014,31(6):240-243.

[6] 钱思园,王　琼,李　暾. 嵌入式软件跟踪调试技术的研究与设计[J]. 单片机与嵌入式系统应用,2012,12(2):1-4.

[7] 胡先浪,张培培. 基于调试代理的远程协同调试模型[J]. 计算机技术与发展,2011,21(5):29-32.

[8] IEEE STD 1149.1-2001,IEEE standard test access port and boundary-scan architecture[S]. USA: Institute of Electrical and Electronics Engineers,2001.

[9] FOGARTY P,HEFFERNAN D,MACNAMEE C. On-chip support for software verification and debug in multi-core embedded systems[J]. IET Software,2013,7(1):56-64.

[10] PARK H,XU Jingzhe,KIM K H,et al. On-chip debug architecture for multicore processor[J]. ETRI Journal,2012,34(1):44-54.

[11] 田　丹,李运喜,胡　宁,等. 基于 Eclipse 的嵌入式软件交叉调试[J]. 现代电子技术,2015(6):86-89.

[12] HE Huiqin. Application research of JTAG standard based on ARM debugging system[J]. Applied Mechanics and Materials,2015,719-720:522-526.

[13] PARK H,XU Jingzhe,JI J H,et al. Design methodology for on-chip-based processor debugger[J]. Design Automation for Embedded Systems,2015,19(1/2):35-57.

[14] 黎　君. 基于 ARM9 嵌入式系统的 Bootloader 移植[J]. 科学技术与工程,2011,11(32):8061-8064.

[15] 王　品,尚利宏. RTEMS 管理机制与 USB 驱动程序设计[J]. 单片机与嵌入式系统应用,2009(1):27-30.

[16] 康云川, 代　彦. 恶意 USB 设备原理及防护措施研究[J]. 计算机技术与发展,2020, 30(1):112-117.