一种面向多核处理器的通用可调试性架构

杨 旭[1,2]，刘 江[3]，钱 诚[1]，苏孟豪[1]，吴瑞阳[1]，陈云霁[1]，胡伟武[1]

[1]（中国科学院计算技术研究所计算机系统结构重点实验室 北京 100190）

[2]（中国科学院研究生院 北京 100049）

[3]（北京航天自动控制研究所四室 北京 100854）

（yangxu@ict.ac.cn）

摘 要：硅后调试对于当代集成电路设计变得日益重要，用于辅助硅后调试的可调式性设计（DFD）应运而生. 由于多核处理器往往包含多种不同类型的部件，每个部件都有各自的调试功能需求，极大地提高了可调式性设计的复杂度. 针对上述问题，提出一种面向片上多核处理器的通用可调试性架构. 该架构使用简单的监测器来监测和控制处理器中用于互连的片上网络，通过专门的调试总线将各个监测器与调试总控模块连接在一起，并使用 EJTAG 通用调试接口与外部调试主机传递信息. 与传统的可调试性架构相比，该架构无需片上 RAM，硬件代价低，具有模块化的特性. 此外，文中提出的架构采用了工业界通用的 EJTAG 调试接口，因此通用性较高，已经被应用于龙芯-3B 多核处理器中. 实验结果显示，该架构可以在高频高数据带宽的环境下工作.

关键词：多核；片上网络；可调试性设计；硅后调试

中图法分类号：TP391

A General DFD Infrastructure for Chip Multiprocessor

Yang Xu[1,2], Liu Jiang[3], Qian Cheng[1], Su Menghao[1], Wu Ruiyang[1], Chen Yunji[1], and Hu Weiwu[1]

[1]（*Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190）

[2]（*Graduate University of Chinese Academy of Sciences, Beijing* 100049）

[3]（*The Fourth Laboratory, Beijing Aerospace Automatic Control Institute, Beijing* 100854）

Abstract: Post-silicon debugging is becoming more and more important in modern integrated circuit design. Hence, design for debug (DFD) is proposed to facilitate post-silicon debugging. The DFD in multi-core processors are very complicate, as there are lots of different components contained in a multi-core processor, and each of them have its own debugging requirements. In this paper, we propose a general DFD infrastructure for chip multiprocessor (CMP) that contains several simple monitors to watch and control the network on chip (NoC). All monitors are connected to a centralized controller, which gathers information and communicates with debugging host using EJTAG port. Compared with traditional DFD infrastructures, the proposed infrastructure is modularized, low cost, and general. This DFD infrastructure has been adopted by a state-of-art industry CMP Godson-3B. Experimental results demonstrated that the DFD infrastructure can work under high frequency and high data bandwidth environments.

Key words: multi-core; network on chip; design for debug; post-silicon debug

---

收稿日期：2011-02-27；**修回日期**：2011-07-15. **基金项目**：国家科技重大专项（2009ZX01028-002-003，2009ZX01029-001-003，2010ZX01036-001-002）；国家自然科学基金（60921002，61050002，61003064，60736012）.**杨 旭**（1977—），男，博士研究生，高级工程师，主要研究方向为高性能计算机体系结构、VLSI 设计；**刘 江**（1979—），男，硕士，主要研究方向为自动控制；**钱 诚**（1985—），男，博士，助理研究员，主要研究方向为微体系结构；**苏孟豪**（1982—），男，博士，助理研究员，主要研究方向为高性能计算机体系结构；**吴瑞阳**（1991—），男，博士研究生，主要研究方向为微体系结构；**陈云霁**（1983—），男，博士，副研究员，主要研究方向为高性能计算机体系结构、硬件验证；**胡伟武**（1968—），男，博士，研究员，博士生导师，主要研究方向为高性能计算机体系结构、VLSI 设计.

1　背景介绍

处理器技术的飞速发展使得芯片内部结构越来越复杂,单个芯片可以提供更多的功能并包含更多的模块.但与此同时,随着芯片结构的复杂化,硅后调试代价越来越大,目前硅后调试占用整个芯片设计流程的时间已经上升到 35%左右,大量的人力和物力开销被消耗在硅后调试之中[1].为了加速硅后调试进程,工业界和学术界开始广泛关注在芯片中采用可调试性架构.可调试性架构可以帮助调试工程师在硅后调试过程中显著地提高对芯片内部信号的控制和观测能力[2].基于这一点,目前主流的集成电路设计公司都在反映其当前水平的工艺产品中采用了可调试性架构,例如用于记录程序执行流的 Intel® Pentium® M SV platform, ARM CoreSight™ Program Flow Trace™ architecture.

一般而言,一个硅后可调试性架构需要包含监控器(获取或改变目标位置的信号),压缩器(用来压缩得到的信号),存储器(用来存储压缩后的数据),传输网络(用来输入和输出信号),接口(可调试性提供给外部的接口),外部调试主机和调试软件(分析信号、定位和修正错误)等部分,每个部分的设计和技术指标需要综合考虑调试需求和工程实际之间的平衡.可调试性架构的设计师必须考虑增加的面积、传输带宽、结构复杂度变化、功耗和功能要求等各方面的需求.所以,在片上多核处理器(on-chip multiprocessors, CMP)中采用可调试性架构需要面对 2 个主要的挑战:

1) 额外的硬件开销.通常,更完备的可调试性支持功能会带来更多的硬件开销.CMP 的片上网络开销控制十分敏感,如果可调试性架构需要增加太多面积将导致整个二维芯片难以布线等困难.

2) 存储和传输观测到的信号.CMP 的结构变得越来越复杂,以往的单核处理器和复杂片上系统(system on chip, SoC)只有几十位数量级的总线宽度,以及接近树形拓扑结构的片上网络,而现在的 CMP 和高集成的 SoC 的片上网络往往具有几百位的总线带宽和复杂的拓扑结构,这将带来控制和观测处理器中信号的数量呈现爆炸式增长.随着信息宽度的增长,其中包含的信息含量也急速增加,传统的一些方法,例如压缩 trace(被记录下的信息)方法很难在不增加大量硬件开销的情况下达到理想的效果.如果仍然沿用一些传统的思路来设计 CMP 的可调试性架构,不仅会造成设计和验证整个设计的困难,更可能会导致整个设计失去物理实现的可能性.

受限于上述片上网络监测和调试的困难,目前还没有被公认的优秀片上网络可调式性架构设计.

2　相关研究

在用来组成可调试性架构的基本单元中,可测性设计往往被直接采用[3-4].可测性设计单元已经发展得比较成熟,它们可以很方便地被用来抓取信号,缺点是导出信号的速度比较慢(例如扫描链)并缺乏针对性的优化.

确定性重放方法是一种新的可调试性设计技术[5],该方法通过在程序运行时录制必要信息,在必要的时候进行重放.多核并行环境十分复杂,不确定因素很多,经常导致并行程序运行出现致命错误.为了重现不确定因素,保证程序的正确性,确定性重放已经成为一种重要方法.但是确定性重放方法主要解决如何精确地获取待观测信号的问题,而对于设计一款可调试性架构需要解决的存储和导出问题则需要结合另外的方法来解决,该方法一般仍然需要大量的 RAM 和高宽度的总线来进行辅助.

已经有很多可调试性架构被用来调试处理器核[6-7]和 SoC 芯片[8-9],很多公司和研究小组也推出了各种可调试性架构和模型[10].这些架构和模型对于提高调试的自动化做出了巨大的贡献,但是如果应用到 CMP 中,所有这些架构和模型都面临着一个额外开销太大的问题,如大量的额外 RAM,复杂的智能分析逻辑,物理实现的低频率.

调试多核芯片一般包括 2 个部分:调试片上网络中存在的问题和调试处理器核中存在的问题,其中主要的挑战是调试片上网络中存在的问题.因为处理器核内部的调试可以通过现有成熟的单核调试手段来进行,而多核环境带来的问题主要集中在片上网络中(如存储一致性验证等问题),但目前并没有通用的调试手段.当调试工程师工作时,可调试性架构需要向其提供高度可视化的片上网络信息以协助其高效的解决问题.利用多核处理器片上网络中的传输信息进行调试的相关研究还比较少,最近几年人们提出了一些相应的模型[11-13].

片上网络中的信号非常多,很难用小的开销达到完全获取所需调试信息的目的.为了解决上面提到的存储和传输问题,很多压缩算法被用来压缩获取

的片上网络信息，但极少有算法可以适应 CMP 系统的环境，即使是一些算法上可行的基于数据挖掘和机器学习的方法，运用到现实中也需要大量的额外开销. 根据香农的信息论，压缩后的数据量取决于信息本身的自信息量[14]. CMP 片上网络包含大量复杂来源的信息和状态，其压缩后的数据量常常会超出可调试性架构设计时允许的存储容量要求，仅仅使用压缩并不能从根本上解决可调试性架构的存储和导出信息问题.

进行硅后调试时，常常需要将芯片的频率降低并保持在一个适合进行调试的频率上，这样也可以起到调节可调试性架构获取的数据带宽和调试接口带宽之间平衡的作用. 但问题是如何自动调节带宽才合适呢？由于可调试性架构中调试网络和片上网络之间的独立性，处理调试网络和片上网络之间的握手信息以精确智能地控制时钟信号并不是一件容易的事情. 而实践中，如果不能精确控制调试频率则往往使得这种调整没有实际意义. 因此如何处理好调试网络和片上网络之间的交互，使得可调试性架构高效工作，也是一个设计中面临的问题.

龙芯-3B 八核芯片的设计过程中对可调试性设计架构的需求是：尽可能地用最小的面积、最小的功耗、最小的结构改动来达到加入可调试性架构的目的. 像龙芯-3B 这样包含 6 000 万门以上的大规模芯片设计中往往需要尽量复用已有的高可靠结构，过于频繁的结构更改会导致即使某些组成部件在性能上产生浪费，也会带来物理设计上的困难，从而导致需要重新设计、验证、布线等一系列繁重而又必要的劳动. 为了使得紧张的设计周期不至于因此而推迟，并可以减少验证整个系统的时间、尽量使用已有的空闲面积，需要有针对性地设计一种可调试性架构. 为了满足 CMP 的硅后调试需要，以及在未来新型号的芯片中仍然能够得到升级甚至更改调试网络的算法模型，设计的可调试性架构必须模块化、可重构以及功能齐全. 例如，AXI 总线是一种处理器核之间的互连方法，其有 AR，AW，R，W 和 B 共计 5 条通路，在目前的龙芯-3B 多核处理器中，其宽度分别从 16～159 位不等，而且会随着未来的处理器设计改变而变化. 用于互连的总线协议也有可能会有所变更，因此需要一个相比目前需求略有冗余的模块化实现来存储和导出这些信息.

根据上述需求，本文提出了一种着眼于片上多核处理器片上网络的通用硅后可调试性架构，该架构可以用来观测整个 CMP 片上网络中的调试需求信息，并具有很低的硬件开销. 图 1 所示为该可调试性架构在多核处理器中的位置，其中，DDR 控制器、PCI 是处理器的对外接口，超传输(hyper transport，HT)是片上互连总线，$C0$～$C7$ 表示 8 个处理器核；图中带圈的 M 标志是本文方法加入的检测器，每个检测器通过总线连接至调试总控模块，并通过各自的地址和总共模块通信. 图 1 中每个监视器 M 和调试总控模块之间为总线连接，但为了简洁明晰，在图中未全部标出. 已收集的信息和控制的命令通过已有的 EJTAG 接口与外部调试主机进行传送. 图 1 中低成本的监测器被添加到片上网络中每个路由节点的边界上. 本文使用一种开销极小的不需要 RAM 的监测器设计，每个检测器使用一个 192 位的数据寄存器(目前绝大多数的片上网络信息都不超过

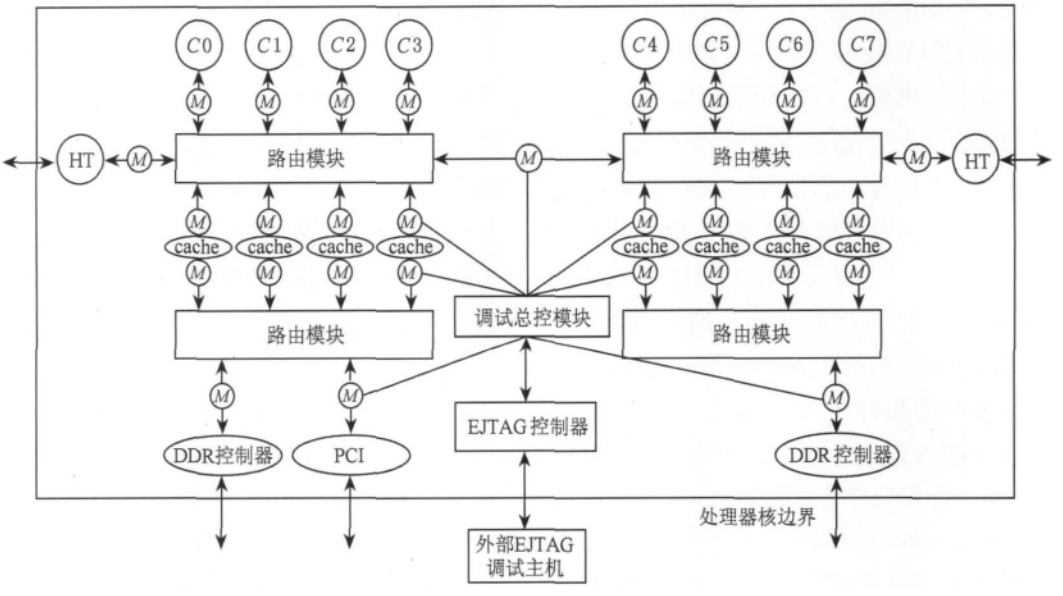

图 1 CMP 硅后可调试性架构

192 位)和一个 64 位的控制寄存器,采用单步的方法将数据导出.每个监测器包含读、写、断开和连接 4 个基本的功能,这些基本的功能可以组合成不同的操作来对单个监测器负责的片上网络总线信息进行获取和改变.调试网络的总线连接每个监测器和总控模块,总控模块再连接调试网络和调试接口.为了节省在每个监测器中使用压缩所需要的 RAM,采用单步调试的方法.单步调试方法将片上网络悬挂起来,直到需要的信息被采集到总控模块中.总控模块中的压缩器集中压缩各监测器送来的信息,这需要一些缓冲区,不过开销仍然不超过任何一个监测器中实现压缩算法所需要的 RAM 大小.总控模块将调试接口传来的调试控制信息和调试网络传来的 trace 信息进行交互.外部调试主机上的软件很容易通过调试接口来控制整个调试网络的运行.这样,通过添加低成本的监测器到片上网络中的每个关键节点,达到覆盖采集并控制片上网络的目的.整个架构的开销很小,并可以根据不同系统的实际需要对开销进行缩减和增加,以削弱或增加调试功能.

该可调试性架构具有如下优点:

1) 单步调试方法对片上网络中的信号使用悬挂机制,节省了在每个监测器中增加额外 RAM 的开销.只有软件当前需要的信号才会在几个时钟周期的间隔内被抓取,极大降低了可调试性架构中产生的 trace 带宽,从而使得调试网络的传输总线宽度变得很窄.这样物理设计的复杂度会大大降低,整个设计的验证和物理实现时间会大大缩短.

2) 所有复杂的逻辑都被分解成一系列简单的逻辑,这些简单的逻辑实现各种基本的功能,通过调试总线发送过来的一串控制指令将基本功能按不同次序组合成不同的高层抽象操作,既简化了维护、验证、升级的工作,也由此带来了可调试性架构的工作频率提升.即使单步调试方法的悬挂特性需要将片上网络中的信号传输停止,但是由于可调试性架构的工作频率可以达到很高,整体的工作效率仍然相当可观.

3) 采用模块化设计.每个组成部分都是独立的,监测器的工作状态只受其自身的控制寄存器影响,调试网络只根据地址信息来读写各监测器中的寄存器,总控模块负责将调试需求信号和抓取的 trace 信号在调试网络和 EJTAG 控制器之间进行转换.所有部件都可以单独更换和根据需要进行升级,从而使得整个可调试性架构实现不同侧重的功能以及不同的工作方式.绝大多数 CMP 都可以不受约束地采用这种可调试性架构.

4) 可以很方便地和外部调试主机进行对接(包含各种软硬件工具的调试系统,如 ARM CoreSight™ Program Flow Trace™ platforms).整个架构采用通用 EJTAG 接口,该接口已经被主流外部调试主机广泛采用.

本文提出的可调试性架构已经被工业化的 CMP 龙芯-3B 所采用.实践结果证明,该可调试性架构是一种实用的获取 CMP 状态的结构,其调试网络只需要 18 位的宽度,对于整体物理设计非常有利;由于工作频率可以达到高频,采集数据的带宽可以达到 500 MB/s;每个监测器的额外开销很小,只需要 256 个触发器(还可以根据需要进行相当程度的缩减).

3　CMP 可调试性架构设计分析

3.1　需要抓取的信号分析

多核调试主要包含调试片上网络和处理器核 2 个部分.对于单个处理器核的调试和验证方法已经比较成熟,但处理器核的调试很少涉及存储一致性或复杂的 IO 问题,所以工业界和学术界都很关注如何调试用于多个处理器核之间互连的片上网络.当调试片上网络问题时,可以把处理器核看成与缓存、I/O,DDR 控制器等一样的设备,这样调试多核处理器实际上是一个调试片上网络并监测其中传输信息的过程.

片上网络的总线协议中一般都包含地址信息、握手信息和数据信息.在调试多核处理器时,因为片上网络中的地址映射非常频繁,所以一般必须有地址信息和握手信息,有时也需要数据信息.

在多核调试中,时间也是一个重要的信息[15],信息的时间顺序错误也会导致 CMP 执行出错.如果不使用 RAM 记录信息的时间信息,那么需要控制总线保持悬挂,以使得抓取的信息保持时间顺序.当使用监测器的断开功能切断片上网络中相应总线的握手信号时,该总线上的所有信号就被保持在片上网络中了;然后可以使用单步调试功能一步步地把该信号抓取出来,通过连接功能使总线恢复运行再抓取下个信号.

一般需要抓取片上网络中信号源、信号目的地、中间路由模块中的信息,不仅这样,在通过调试网络将事先准备好的信息导入到片上网络中时,这些导入的信息也需要能被抓取,这是因为需要判断该信息

是否被正确地导入了片上网络,以及在片上网络中是否产生了想要的结果.

3.2 信息记录方法

如果采用大量的 RAM 来记录片上网络中的信息,就需要采用压缩算法. 根据分析,对 CMP 的片上网络中信息进行压缩需要采用分组的方法才能达到比较好的压缩效果,即根据局部性原理将易于压缩的一些信息放到一起进行压缩. 一般情况下,需要采用 8 组才能达到较好的压缩效果,对于一次需要获取 128 位信号的部件,至少需要超过 1024 个触发器才能实现这个功能. 在很多低频率系统中,结合压缩逻辑可以降低使用的触发器数目. 但是这在高频率的系统中很难实现,因为压缩逻辑越复杂触发器之间的延迟越大,越无法在高频率下实现. 一个多核系统的片上网络往往有大量的路由部件、数据源和目的部件,每个部件的边界都需要加入一个监测器. 如果单个监测器的开销太大,则整体的开销往往难以承受.

压缩算法需要记录信息的初始信息(往往和原始信息长度一致)和偏移量信息,一些算法还需要字典信息(一张映射表),例如 LZ 压缩算法[16-17]. 如果使用这些压缩算法去处理这些信息,每个监测器所需的额外开销将十分巨大,因为每个监测器连接的总线信号宽度超过 500 个,采集的信号每组在 192 个.

为了取代存储-压缩算法,不使用 RAM,本文从另一个方向去思考. 如果每次只输出所采集信息的一部分,分多次输出信息,那么只需要使用一个有限大小的寄存器就可以达到要求. 但是随之而来的代价是采集的速度要低于数据的自然产生速度,由于目前 EJTAG 调试接口的频率远小于芯片的工作频率,因此不会形成性能瓶颈,而且受各种分析算法的时间复杂度制约,外部调试主机的分析软件目前所需要的信息带宽也不高.

为了达到分多次采集一组信息的目的,把监测器设计成可以根据需要将 192 位的数据寄存器分别接到不同组总线中的某些信号上. 由于总线根据握手信号来判断数据是否成功传输,因此监测器还可以同时将多组总线的握手信号断开,这样当一组总线的信号被连接到寄存器时,其他相关组总线的信号也可以被保持在总线上,直到监控器对相应握手信号进行了操作,可以避免总线之间出现的相关错误以及信号的时间信息被打乱.

本文使用一个集中式的压缩器来对各监测器汇总到总控模块中的信息进行压缩. 为了保证压缩的高效率,充分利用外部调试主机中软件的程序局部性,各监测器汇总的信息都是当前软件需要分析的信息,其余的信息处于不导出的状态(不需观测或者悬挂在总线上). 一个很小的循环缓冲用来保存压缩后的信息,并被调试接口实时导出.

3.3 数据传输分析

为了取代存储-压缩算法,本文使用有限的触发器来记录信息,并通过分步的方法来导出. 使用监测器的切断功能采集特定组的信号,相关的信号都被悬停在总线上,以保证时间序和相关性不受破坏.

保持从监测器到 EJTAG 调试接口的数据高效传输十分重要. 可调试性架构的最大带宽受 EJTAG 调试接口的限制,可调试性架构的总线频率和片上网络频率一致,远大于 EJTAG 调试接口. 尽管采用单步的方法,但是由于频率很高,调试总线能达到的带宽仍然大于调试接口,所以可调试性架构的带宽只受调试接口的制约.

在调试网络和调试接口之间的总控模块中放入压缩器和缓冲可以充分利用片上的空闲面积,还可以调节调试网络和接口带宽的不匹配. 低成本的监测器由于面积小可以很方便地插入片上网络中原有小空隙面积中,集中的总控模块面积稍大,可以放到较外围的空置面积中.

4 可调试性架构

图 2 所示为采用先进工艺的龙芯-3B 多核芯片,其具有 8 个处理器核、8 块共享缓存、连接了 2 个 HT 端口和 2 个 DDR 控制器的可扩展片上网络. 该片上网络有 4 个路由器,并且还可继续扩展. 可调试性架构需要探测通过路由器的信号,检查信号是否正确,是否违反给定的协议. 所有到路由器的源和目标总线都应添加一个监测器,调试时,外部调试主机上的软件会判断需要获取信息的目标区域(例如片上网络中的设备),该目标区域的输出通道应首先断

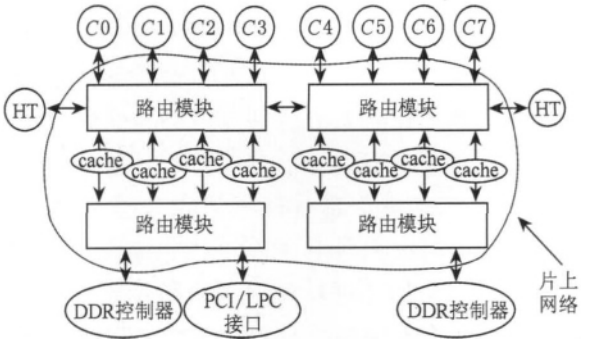

图 2 龙芯-3B 多核处理器的结构图

开,那么所有的信号会被保持在片上网络的这一区域中,这样可以利用芯片中已有的硬件电路来保存要导出的信号.使用逐步抓取的方法将不同来源的信号抓取出来,然后软件根据这些信号分析产生错误的地点——来源或路由器.

4.1　模块化的结构设计和架构的拓扑结构

本文使用模块化的架构和模块化设计降低成本和复杂性,方便更新可调试性架构.图 3 所示为模块化的架构中不同的组成模块,如总控模块、调试网络和监测器.其中总控模块转换 EJTAG 接口和调试网络之间的信号,调试网络传输总控模块和监测器之间的交互信息.根据地址,不同的调试请求将被发送到不同的监测器.监测器接收调试请求,监测被调试的总线,并将相应的数据存储在图 3 中 $R0\sim R5$ 等寄存器中,通过寄存器与调试网络交互.当更新可调试性架构时,仅需要改变架构中的子系统以及子系统到其他部分的映射关系.映射关系由有限状态机来确定(例如总控模块和监测模块中的有限状态机),当总控模块和监测器中的有限状态机被配置好后,这些组成部分就可以具有特定的行为,从而整个架构的运行体现出特定的工作模式(本文中即单步调试模式,通过改变各状态机整个架构可以以其他模式工作).例如,需要监测不同种类的片上网络时,只需要修改监测器的实现,其他部分不需要改变.模块化的结构设计大大提高了本文方法的通用性.

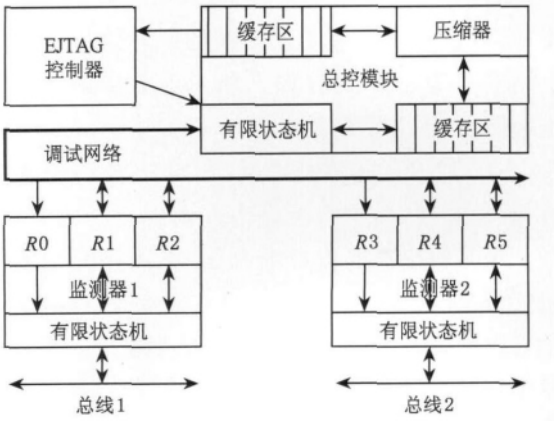

图 3　模块化架构的示意图

多核处理器的片上网络可以作为有向图来进行分析.每一个目标和源设备可以被看作是一个节点,它们之间的每一条单向总线为一条边.如图 4 所示,如果要调试有向图的一个区域,应该将监测器加入这一区域的割边,这些监测器再由调试网络连接.从 EJTAG 接口得到的调试请求消息转化为调试网络上的信息;信息被广播到监测器,从调试网络另一方向的总线上得到监测器返回的应答信息.

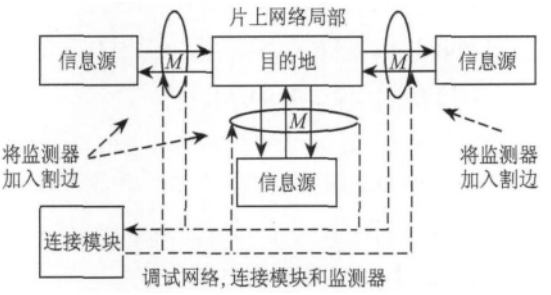

图 4　插入可调试性电路的示意图

4.2　监测器

每个监测器连接着几组不同的总线,根据调试需求每次采集其中的一组信号,图 5 所示为监测器的示意图.图 5 中,总线上的信号通过数据寄存器记录下来,数据寄存器每次连接到一组信号上,配置寄存器控制具体哪组信号连接到数据寄存器上以及哪些信号需要被断开.

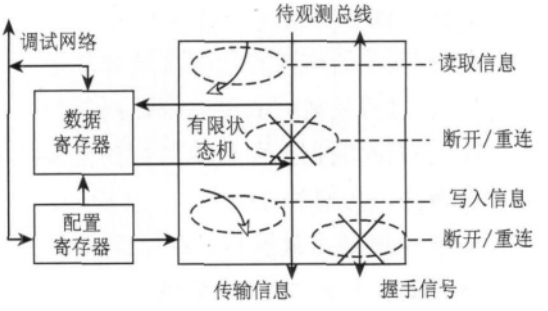

图 5　监测器示意图

监测器有断开连接、重新连接、读寄存器和写寄存器 4 个基本功能.当用数据寄存器和相应总线连接后,连接到监测器的输入信号可以被采集,输出信号可以被改变.断开总线上的握手信号则将信号保持在片上网络的总线中.当总线和数据寄存器连接起来后,配置寄存器可以控制总线上的信号被采集多少次或者从寄存器写多少次数据到总线上,每次数据的交互通过重置总线的握手信号来实现.当需要总线悬停时,握手信号被断开;当需要交互数据时,握手信号被连接到配置寄存器并赋予具体含义的值,完成一次交互后握手信号被重置等待下次握手.

如果想对总线实现某种操作,例如读取总线上的信号,则需要按一定顺序调用监测器的基本功能;监测器必须支持所有必要的顺序操作.图 6 显示了监测器状态机工作时在基本功能状态下转换所遵循的顺序关系,状态机转换一系列的状态,监测器完成一种特定的操作.

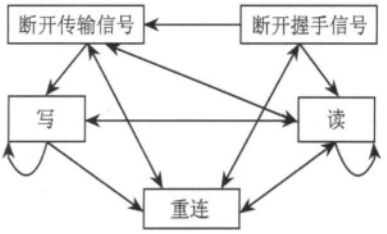

图 6 对信息进行原子操作的组合顺序关系

断开连接功能是单步可调试性架构的一个重要功能：

1）片上网络中的时钟频率远远超过了单步抓取信息的频率(虽然调试网络频率可以保持和片上网络一致，但是片上网络中某一周期所传输的信号在单步调试方法中需要多个周期才能全部抓取出来)，如果不对这种频率差异进行处理就不可能正确获取信息. 为了避免添加额外的握手逻辑、异步缓冲以及时钟侦测器等同步时钟所需要的结构，本文使用断开连接的方法来减缓片上网络的信息传输速度. 当单步可调试性架构工作时，外部调试主机上的软件判断片上网络中的哪一片区域需要被隔离出来；然后命令调试网络断开这一区域和其他区域之间的连接，通过一系列的调试命令控制该片区域的数据传输，从而保证片上网络中的数据传输和调试网络中的数据传输速率同步.

2）每个监测器需要控制几组总线，这些总线之间往往具有依赖关系，例如当总线上传输的是一次写访存请求时，传输数据的总线和传输地址的总线中的内容可能需要保持同步. 如果要抓取其中一组总线上的信息，在断开该组总线并单步抓取数据的同时，其他相关组总线也应该被断开，而不能任由其传输，避免片上网络中的目的部件对传输出现错误的解读.

3）断开功能可以使得信息被保持在片上网络中. 如果握手没有完成，片上网络中的信息会被保持在上一级寄存器中，这样就可以利用已有的硬件来保存暂时不能导出的数据.

读写数据寄存器和配置寄存器是调试网络和监测器交互的方式. 配置寄存器收到调试网络传来的数据后，状态机会相应地转换状态，根据状态的不同，相应总线和数据寄存器发生连接/断开操作，同时握手信号根据对总线的不同操作得到不同的值. 通过总线和寄存器之间的交互、寄存器和调试网络之间的交互，实现调试网络和片上网络总线之间的信息交互. 调试网络既可以获取片上网络总线中的数据，也可以将数据导入到片上网络的总线中. 每个监测器只需要 256 个触发器，其中 192 个是数据寄存器，其余的是配置寄存器.

4.3 单步调试方法

监测器分布在由路由节点和设备组成的拓扑网络中，调试网络连接这些监测器和总控模块，总控模块和一些调试设备连接. 这些调试设备接收外部调试主机的请求，将调试请求发送给调试网络，并从调试网络中获取得到的信息. 由于 EJTAG 接口在当前的商业处理器中已经非常普及，因此龙芯-3B 多核处理器选择 EJTAG 作为可调式性架构的调试设备. 如上文所述，本文中的设计具有很好的模块化特性，因此只需要简单地更换调试设备的实现方式，就可以换用其他种类的调试设备.

如图 1，2 所示，整个片上网络可以看作一个有向图，该有向图可以被视作由多个子图构成. 当调试多核处理器时，主要是判断信号在片上网络中的传输是否正确，以及不同的传输之间是否有冲突. 传输的目的设备往往通过有向图的割边和其余部分相连，或者通过有向图子图的割边和一个子图的其余部分相连，在有向图或其子图的割边中加入监测器就可以切断目的设备和其余部分的连续. 在切断目的设备的信号传输后，进一步逐级切断通向源设备的路由节点等的割边，根据调试需求隔离出一片区域；然后就可以导出需要观察的数据并判断是否正确. 如果该区域中的所有信号都是正确的，则转向其他区域或者重连该区域的总线进行下一步的观测；如果发现有错误信号，则可以通过分析来定位错误，当然也可以人为注入一些特定的信号来观察片上网络中是否会出错. 图 7 所示为单步调试方法的流程图.

在龙芯-3B 多核处理器中，调试网络的总线具有 18 位宽度，其中一半是输入调试命令的总线，另外一半是输出抓取信号的总线. 在网络上传输的每组信号都有 1 位握手信号、几位地址信号和一些表示操作的信号，所有命令信号通过输入总线广播给监测器，相应的应答则通过输出总线返回. 总控模块会处理命令和返回之间的对应.

由于 EJTAG 控制器的时钟和调试网络中的不同，因此在总控模块中需要集中处理时钟同步，这些开销都集中于总控模块中，避免了重复的消耗. 由于调试网络和监测器的硬件成本非常低，有限的开销又集中合并于总控模块，因此单步可调试性架构增加的整个额外开销非常有限. 下面列出几种可调试性架构的开销对比.

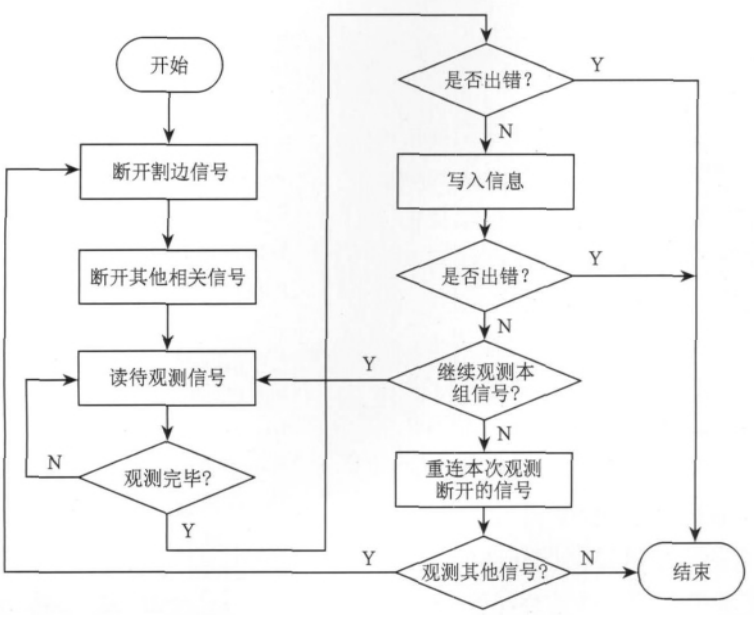

图 7　单步调试方法的流程图

4.4　开销和性能对比

表 1 所示为 4 种最新的片上网络可调试性架构之间的成本和性能对比. 其中,指令足迹记录和分析(instruction footprint recording and analysis, IFRA)[6]可调试性架构用于调试处理器流水线中的错误,其具有经验智能的分析逻辑并能覆盖绝大多数种类的错误,它使用 28 个足迹记录构件(footprint recording structures, FRS),每个时钟周期可以记录 4 条 128 位指令的信息;另外 2 种可调试性架构用于验证 SoC 芯片,其中一个采用实时压缩导出方法,另一个采用高压缩率的存储-导出算法. 这 3 种都是当前技术领先的架构.

表 1　4 种可调试性架构的对比

架构名称	监测器名称	触发器数	trace 产生带宽/位/时钟周期	待测芯片总线宽度/位/频率/MHz(极值)	信息获取带宽/MB/s
IFRA[6]	FRS	2 K(最少)	35(最大)		342(平均)
Real-time compresser[8]	Segment buffer	8 K	23	32/500	1 450
Infrastructure of DSP debug[9]	Trace interface	32	32	32/24	96
本文提出的架构	monitor	256	3.9	192/1 000	499

相对于表 1 中前 3 种可调试性架构,本文提出的单步可调试性架构具有非常高的性价比并在多核处理器的调试中有不可替代的作用. 单步可调试性架构记录每一组 192 位信号需要 50 时钟周期(每写一次配置寄存器并分 3 次读 192 位数据寄存器可以采集到 192 位信号,每个读写操作需要对调试网络耗费 10 个时钟周期. 再加上若干次采集中间的断开和重新操作所需要的配置开销,平均耗费 50 个时钟周期的开销). 由于操作指令很简单,并且没有使用 RAM 和相应的压缩逻辑,调试网络的频率可以和片上网络的一致,达到 1 GHz. 这样每个时钟周期生成的抓取信号带宽只有 3.9 位/时钟周期(这便是调试网络可以保持很小宽度的原因),但采集到的信息带宽却可以达到 499 MB/s,每个监测器只需要 256 个触发器,比其他 3 种可调试性架构性价比高. 更重要的是,本文的单步可调试性架构是一种通用的架构,可以很方便地应用于任何有片上网络的多核处理器中. 其贡献包括高调试频率、模块化特性、利用调试软件具有的局部性特征避免抓取信息爆炸增长、集中压缩、监测器不需要 RAM 等,这些贡献解决了多核处理器可调试性架构面临的基本问题,即控制硬件开销和处理抓取信息的存储导出问题,而这些通用性的解决方法是其他 3 种可调试性架构所不能替代的.

5 总 结

本文从架构设计的角度探讨了面临许多实际限制的多核处理器可调试性架构设计. 功能的强大和灵活意味着更复杂的逻辑、更多的触发器和功耗的增加. 更重要的是,在工业实际中,需要在设计需求和设计时间之间寻找一个平衡点.

本文提出一种面向多核处理器的普适性单步可调试性架构,其使用单步调试方法,分 2 步解决了设计面临的存储和导出数据问题:首先利用片上网络的特性实现将调试信息保持在片上网络中,从而避免了在监测器中加入复杂的压缩逻辑和大量的 RAM;然后利用集中式的压缩以及调试软件的局部性,将只和当前调试需求相关的信息进行高效压缩,避免了观测信息的爆炸问题,提高了存储导出的效率. 本文架构具有不需要 RAM、只需要很窄的调试总线、低成本的硬件开销等优点,被先进工艺多核处理器龙芯-3B 所采用,在实践中很好地满足了芯片开发的条件约束和功能需求;用较低的开销支持调试需求,并避免了在实现和验证整个系统中增加过多时间消耗. 更值得指出的是,该架构对于多核处理器具有通用性,可以很方便地移植于任何有片上网络的多核处理器中.

参考文献(References):

[1] Josephson D. The good, the bad, and the ugly of silicon debug [C] //Proceedings of the 43rd Annual Design Automation Conference. New York: ACM Press, 2006: 3-6

[2] Pyron C, Bangalore R, Belete D, *et al*. Silicon symptoms to solutions: applying design for debug techniques [C] // Proceedings of IEEE International Test Conference. Washington D C: IEEE Computer Society Press, 2002: 664-672

[3] Carbine A, Feltham D. Pentium pro processor design for test and debug [J]. IEEE Design and Test, 1998, 15(3): 77-82

[4] Pyron C. Scan and BIST can almost achieve test quality levels [C] //Proceedings of the 2002 IEEE International Test Conference. Washington D C: IEEE Computer Society Press, 2002: 1196

[5] Xu M, Bodik R, Hill M D. A "flight data recorder" for enabling full-system multiprocessor deterministic replay [C] //Proceedings of the 30th Annual International Symposium on Computer Architecture. New York: ACM Press, 2003: 122-133

[6] Park S B, Mitra S. IFRA: instruction footprint recording and analysis for post-silicon bug localization in processors [C] // Proceedings of the 45th Annual Design Automation Conference. New York: ACM Press, 2008: 373-378

[7] Sarangi S, Narayanasamy S, Carneal B, *et al*. Patching processor design errors with programmable hardware [J]. IEEE Micro, 2007, 27(1): 12-25

[8] Lin Y T, Shiue W C, Huang I J. A multi-resolution AHB bus tracer for real-time compression of forward/backward traces in a circular buffer [C] //Proceedings of the 45th Annual Design Automation Conference. New York: ACM Press, 2008: 862-865

[9] Hsieh M C, Huang C T. An embedded infrastructure of debug and trace interface for the DSP platform [C] // Proceedings of the 45th Annual Design Automation Conference. New York: ACM Press, 2008: 866-871

[10] Abramovici M, Bradley P, Dwarakanath K, *et al*. A reconfigurable design-for-debug infrastructure for SoCs [C] // Proceedings of the 43rd Annual Design Automation Conference. New York: ACM Press, 2006: 7-12

[11] Goossens K, Vermeulen B, van Steeden R, *et al*. Transaction-based communication-centric debug [C] // Proceedings of the 1st International Symposium on Networks-on-Chip. Washington D C: IEEE Computer Society Press, 2007: 95-106

[12] Vermeulen B, Goossens K, van Steeden R, *et al*. Communication-centric SoC debug using transactions [C] // Proceedings of the 12th IEEE European Test Symposium. Washington D C: IEEE Computer Society Press, 2007: 69-76

[13] Abramovici M, Goossens K, Vermeulen B, *et al*. You can catch more bugs with transaction level honey [C] // Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. New York: ACM Press, 2008: 121-124

[14] Chen Y J, Chen T S, Hu W W. Global clock, physical time order and pending period analysis in multiprocessor systems [OL]. [2011-02-27]. http://arxiv.org/abs/0903.4961v2

[15] Shannon C E. A mathematical theory of communication [J]. ACM SIGMOBILE Mobile Computing and Communications Review, 2001, 5(1): 3-55

[16] Ziv J, Lempel A. A universal algorithm for sequential data compression [J]. IEEE Transactions on Information Theory, 1977, 23(3): 337-343

[17] Welch T A. A technique for high-performance data compression [J]. Computer, 1984, 17(6): 8-19