

基于微码控制器的 NoC 网络接口设计

周 帅, 李 丽, 潘红兵, 李 伟, 何书专

(南京大学 微电子设计研究所, 江苏 南京 210093)

摘 要: 本文提出了一种适用于双通道路由器构建的片上网络系统的网络接口. 该网络接口集成 IP 核一侧采用 AMBA 协议, 网络一侧同时兼容 3 种片上通信协议以适应不同的 NoC 应用环境. 在该设计中微码控制器通过加载并执行微程序控制整个接口的工作流程, 这种可编程的设计使得该网络接口具有一定的灵活性. 独立双通道的设计可以同时进行网络包的发送和接收, 保证了片上通信效率. 实验结果表明, 该网络接口资源消耗少, 响应延迟低.

关键词: 网络接口; 片上网络; 微码控制器; 多协议

中图分类号: TN47

文献标识码: A

文章编号: 1000-7180(2012)12-0007-05

Design of Network Interface Based on Mirco-code Controller

ZHOU Shuai, LI Li, PAN Hong-bing, LI Wei, HE Shu-zhuan

(Institute of VLSI Design, NanjingUniversity, Nanjing 210093, China)

Abstract: In this paper, a network interface is proposed for Network-on-Chip built based on dual-channel routers. It supports AMBA protocol on the side of IP core and 3 different communication protocols on the side of network to accommodate different applications. Micro-code controller controls the work process of this interface by loading and executing micro-programs. Designed with dual-channel port, this network interface can send and receive packets independently. According to the experimental results, this design consumes relatively few hardware resources and keeps low latency.

Key words: network interface; network on chip; micro-code controller; multi-protocol

1 引言

随着集成电路工艺的不断发展,多核体系成为管理多资源的有效途径,如何协同多核之间的工作则是整个系统的关键. 传统的基于总线的片上系统(System-on-Chip, SoC)集成方案有其内在的不足,在核的数目逐渐增多时,通信延迟、噪声、功耗、信号完整性等一系列问题将更加严重. 基于包交换的片上网络(Network-on-Chip, NoC)正是为了解决这些问题而被提出来的^[1]. 片上网络结构通信带宽高,可扩展性好,同时有利于 IP 核的重用性,在当前和未来的片上多核系统设计中具有良好的应用前景.

片上网络一般由片上路由器(Router, R)、路由器之间的互连网络、网络接口(Network Interface,

NI)和 IP 核组成. 网络接口则负责将多个 IP 核集成到路由器上,对整个系统通信质量有重大影响,设计出高效、低消耗的网络接口对提升 NoC 系统的性能十分重要. 一般而言,网络接口的设计是基于有限状态机(Finite State Machine, FSM),不仅设计复杂,逻辑资源的消耗也较大. 本文提出一种基于微码控制器、同时兼容 3 种链路通信协议的网络接口设计,并对其进行分析和验证.

2 协议分层

从 ISO/OSI 参考模型^[2]的角度来看,网络接口工作在传输层^[3],它为 NoC 网络中的 IP 核提供数据交换服务,使得 IP 核与底层网络分离,从而保证 IP 核的独立性. 由于与路由器直接相连,网络接口

还同时提供数据链路层和物理层服务。

2.1 传输层

本协议层规定了包的种类、发送数据的分组方法以及数据流的控制机制。包的种类主要有数据和控制,其中数据包用于发送一定量的数据,数据多少不确定,而控制包仅仅发送很少量的控制信息,包的长度一般很短。根据以上要求,本文设计提出的网络接口采用双通道的结构,同时支持控制和数据两种类型的包,其中控制包长度固定为 4 个数据微片,而数据包的长度可配置。

2.2 链路层

本协议层定义了每类包的详细格式,包括构成形式、标志位、数据和配置信息的组织方式以及链路建立的机制。本文设计中,每个包至少包括包头、配置信息、包尾三段结构。包头由一个数据微片构成,包含发送方的节点位置、接收方的节点位置等关键的路由信息;配置信息由两个数据微片构成,用于描述包的类型、数据格式、数据长度、控制信息等;包尾由一个数据微片构成,用于表征包结束,不含其它信息。控制包仅含以上三段结构,而数据包则在配置信息和包尾之间包含一段数据,其长度不定。

不同的 NoC 通信协议适用于不同的应用场合。为保证一定的适应性,本文提出的网络接口同时支持包—电路连接交换(Packet Connected Circuit, PCC)^[4]、信元流控(Credit based Flow Control, CBFC)^[5]和 4 步握手流控(4-phase Handshake Flow Control, 4PHFC)^[6]三种链路协议。

2.2.1 PCC 协议

该协议规定了 7 组信号,分别为: data、stb、fwd、ack、fail、suspend、cancel。一次成功的发包过程包括以下几个阶段:请求阶段,发送方拉高 stb 信号,并在 data 上输出包头,如果该次发包请求被允许,则接收方会将 ack 信号拉高一个周期;传输阶段,请求被允许后,发送方依次发送数个数据微片,用 fwd 信号与 suspend 信号进行同步,并以包尾结束;释放阶段,接收方检测到包尾,将 cancel 信号拉高一个周期,然后发送方拉低 stb、fwd 信号,释放链路。该协议下,一旦发送请求被允许,对相同链路的发送请求会被屏蔽,直到当前包发送完毕,因此该协议比较适合包个数不多但长度较长的应用。

2.1.2 CBFC 协议

该协议共有 3 组信号,分别为 credit、request 和 data。credit 信号位宽为 1,表征包接收方当前的状态,若为 1 则表明可以接收数据微片,为 0 则反之;

request 信号位宽为 1,用于表征发送方的传输请求;data 为要传输的数据。包发送过程相对简单,只有在 credit 为 1 时,才能发送传输请求。该协议信号较少,并且会根据接收方的状态来选择是否发送下一个数据微片,有利于缓解网络拥塞,比较适合包个数多的应用。

2.2.3 4PHFC 协议

该协议有 3 个信号,分别为 request、data 和 ack。request 位宽为 1,用于表征发送方的传输请求;ack 信号位宽为 1,用于接收方响应传输请求。每发送一个微片,request 和 ack 都要完成一次 4 步握手,以保证数据传输的正确性。该协议适合异步通信的应用。

2.3 物理层

本协议层由基本的逻辑门及连线构成,为链路层提供物理链接。由于所提出的设计支持三种协议,并且三种协议应用在不同场合,它们可以共用一组物理连线,如图 1 所示。为了支持 PCC 协议,至少要有 7 根连线,其他协议则选用其中的部分连线。

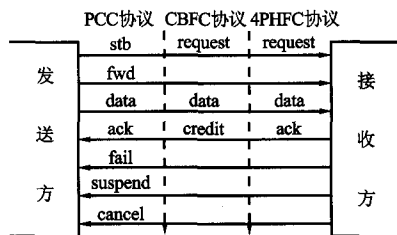


图 1 3 种协议共用链路示意图

3 网络接口的设计实现

本文提出的网络接口在 IP 核一侧支持 AMBA 总线协议^[7],发送包的过程由 IP 核通过 AMBA 总线配置寄存器启动。首先 IP 核将待发送包的节点、包的类型等关键信息写入网络接口中的寄存器组,然后再配置 DMA,最后将发送信号置 1,启动发送。发送数据的过程则由 DMA 和网络接口协同完成。一个包发送完成后,网络接口会产生中断信号通知 IP 核。接收包的过程则是开始于网络接口响应片上路由器的传输请求并产生中断信号通知 IP 核;IP 核在收到中断后会读取网络接口寄存器、配置 DMA,接收数据的工作由网络接口和 DMA 协同完成;在一个包接收完成后,网络接口会以中断的形式通知 IP 核。

本文中的网络接口设计由输出通道和输入通道两个部分组成。两个通道各包含一组寄存器组、一个数据缓冲 buffer、一个微码控制器和一个协议实现

层;另外输出通道还有一个计数器和一个负责生成包头的模块,对应的输入通道有一个解包头的模块.输出通道和输入通道的结构分别如图 2 和图 3 所示.微码控制器和协议实现层是本设计的核心,下面将进一步做详细介绍.

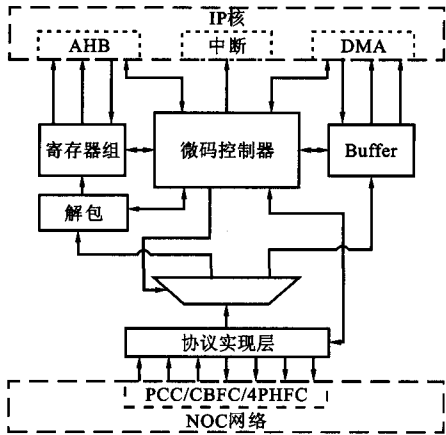


图 2 NI 输出通道结构示意图

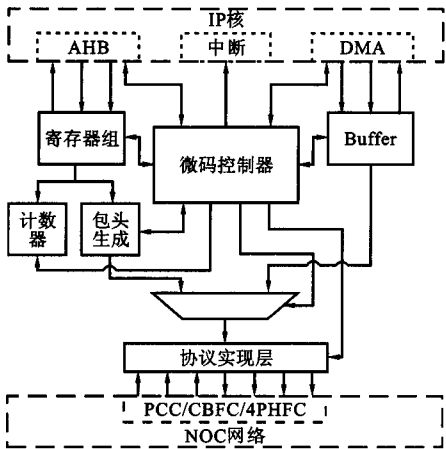


图 3 NI 输入通道结构示意图

3.1 微码控制器

微码控制是设计 CISC 处理器控制器的流行技术,采用微程序语言实现,设计简单灵活.在本设计中,微码控制器取代了传统的 FSM,负责整个通道所有部件的控制.它具有可编程性,可以通过编写不同的微程序来支持多种协议.每段微程序都是由多条微指令组成,而每条微指令又由多个域组成.本设计中的微指令按照图 4 所示的结构进行定义:

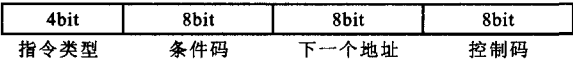


图 4 水平微指令结构示意图

其中指令类型位宽是 4 bit,该域定义微指令的类型;条件码位宽 8 bit,用于条件跳转指令,该域的值经过特定的编码解码能完整反映外部组件的工作状态;下一个地址位宽 8 bit,用于跳转指令;控制码

位宽 8 bit,用于对外部组件的控制,经过特定的编码解码,该域能够对外部的每个组件生成控制信号.

由于指令类型位宽为 4 bit,最多可以定义出 16 种不同的微指令,目前已定义的有 5 条,其编码及功能如表 1 所示.

表 1 微指令定义

指令类型	功 能
4'b0000	维持当前地址
4'b0001	当前地址加 1
4'b0010	跳转至指定地址
4'b0100	条件码符合则跳转,否则维持当前地址
4'b1000	条件码符合则跳转,否则当前地址加 1
其余	未定义

本设计中两个通道的微码控制器都采用了如图 5 所示的结构.

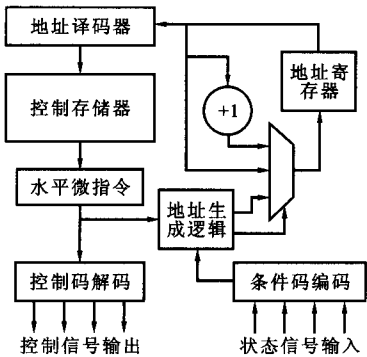


图 5 微码控制器结构示意图

控制存储器根据当前地址译码器的结果输出一条水平微指令.该指令的控制码域经过解码生成一组控制信号输出,其余的各个域发送到地址生成逻辑单元,该单元根据指令的类型与外部输入的状态信号计算出下一条微指令的地址,并写入地址寄存器中,地址寄存器中的值经过地址译码后输入控制存储器,读取下一条微指令.

PCC、CBFC 和 4PHFC 三种协议各对应于一微程序,存放在控制存储器中.不同协议之间的切换通过以下方式实现:在寄存器组中,有一个寄存器专门用于表征当前所选择的协议,该寄存器的值作为状态信号的组成部分,通过条件码编码被地址生成逻辑识别,从而选择出对应于当前协议的微程序.

3.2 协议实现层

协议实现层根据当前选择的协议,控制所有的输入输出信号.在这里,采用了冗余逻辑的设计方式,为每种协议单独设计一个实现模块,并通过协议选择信号进行选择.整个协议实现层由两个多路选择器包裹一组协议实现模块构成,在与片上路由器

相连的一侧,设置有 7 路信号,在与内部组件相连的一侧,有多组信号分别与对应的组件相连.图 6 所示是输入通道协议实现层的结构.输出通道的协议实现层的结构与之类似.

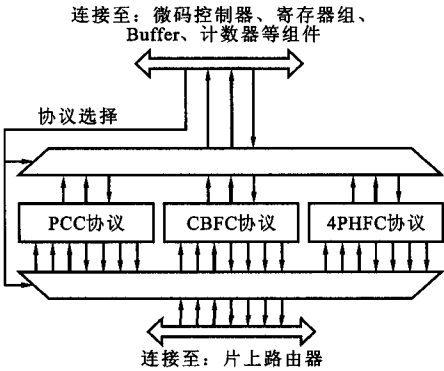


图 6 NI 输入通道协议实现层结构示意图

3.3 解包和包头生成

本设计采用读写寄存器的方式进行解包和打包.任何一个在片上网络中传输的包,都带有一些关键信息.在图 3 所示的 NI 输入通道中包含一个解包模块,负责从收到的包头中提取该包的信息,并写入寄存器组中的不同寄存器,然后 IP 核通过总线读取寄存器,根据相应的值做出响应.在图 2 所示的 NI 输出通道中包含一个包头生成模块,IP 核将包头信息写入寄存器组之后,该模块会根据寄存器的值,生成一个包头.

3.4 寄存器组

寄存器组包含多个寄存器,IP 核通过总线可以访问其中任何一个.寄存器组内一部分用于暂存包头信息,另一部分用于暂存数据.这就使得网络包内数据来源不仅仅是数据缓冲 buffer,也可以是寄存器组,相应的网络包内数据也可以写入寄存器.

3.5 数据缓冲 Buffer

数据缓冲 Buffer 采用 FIFO 实现,通过 DMA 访问 IP 核内的存储单元,可以较快速读取和写入数据.为满足高速传输需求,数据缓冲 Buffer 采用片内存储实现.该 Buffer 的深度取决于 DMA 和片上路由器吞吐率的差值和二者之间的响应延迟,在此不再做详细的讨论.本文中数据缓冲 Buffer 的深度设置为 16,即最多能缓冲 16 个数据微片.

4 实验结果

本文使用 Verilog HDL 语言描述,采用 ISE12.2 综合工具,基于 Xilinx VC5VLX30 FPGA 器件,同时设计实现了以下 5 种类型 NoC 网络接口:

(1)NI_FSM_PCC,基于有限状态机实现仅支

持 PCC 协议的网络接口;

(2)NI_MC_PCC,基于微码控制器实现仅支持 PCC 协议的网络接口;

(3)NI_MC_CBFC,基于微码控制器实现仅支持 CBFC 协议的网络接口;

(4)NI_MC_4PHFC,基于微码控制器实现仅支持 4PHFC 协议的网络接口;

(5)NI_MC_3PRO,基于微码控制器实现同时兼容 3 种协议的网络接口.

在所有上述设计中,网络包的数据位宽为 32 bit,DMA 接口数据位宽为 32 bit,AHB 接口数据位宽为 32 bit.网络包的数据微片位宽为 34 bit,其中 2 bit 作为类型标识.在此将给出设计实现结果,并从响应延迟和资源消耗两个方面评估本文所提出的网络接口性能.

4.1 片内存储的消耗统计

片内存储的消耗由两部分组成:数据缓冲 Buffer 内的存储和微码控制器内的存储.其中数据缓冲 Buffer 内的存储深度固定为 16,位宽为 32 bit;微码控制器内的存储深度等于微程序的长度,位宽为 28 bit.影响微程序长度的主要因素包括所实现的协议和微指令集.对于相同的微指令集,一般协议越复杂,需要的微程序也就越长;而实现相同协议时,微指令集越复杂,需要的微程序也就越短.表 2 中给出了 5 种不同设计对片内存储资源的需求.

表 2 不同设计消耗的片上存储统计

设计类型	Buffer		微码存储		总计 (bit)
	输入	输出	输入	输出	
NI_FSM_PCC	16	16	0	0	1024
NI_MC_PCC	16	16	8	7	1444
NI_MC_CBFC	16	16	6	7	1388
NI_MC_4PHFC	16	16	13	14	1780
NI_MC_3PRO	16	16	27	28	2564

4.2 逻辑资源的消耗统计

为了便于从整体上进行对比,所有设计的片内存储均采用 FPGA 内的 LUT 实现,具体的结果在表 3 第二列给出.表 3 右侧两列分别给出了各个设计总的逻辑资源消耗(含片上存储)情况.

参考文献[8]设计了 MNA 和 SNA 两种接口,分别实现 4PHFC 协议和 CBFC 协议,并选用与本文相同的 FPGA 器件进行综合,给出了这 4 种设计的逻辑资源的消耗情况.作为对比,该文献的具体实验结果在表 3 的最后 4 行列出.

分析本文给出 5 种设计的综合结果可以发现:

同样支持 PCC 协议,基于微码控制器的设计比基于 FSM 的设计多占用了 48 个 LUT 用于实现片内存储,但总的逻辑资源消耗却节省了约 10%;基于微码控制器单独实现某种协议的设计消耗的逻辑资源量相差不多;同时实现 3 种协议所需的逻辑资源与单独实现某种协议相当,仅比 PCC 协议的设计多了约 2.3%。

表 3 不同设计消耗的逻辑资源统计

设计类型	片内存储 LUT	总的 Registers	总的 LUTs
NL_FSM_PCC	48	748	699
NL_MC_PCC	96	622	684
NL_MC_4PHFC	96	624	667
NL_MC_CBFC	96	621	653
NL_MC_3PRO	96	631	705
MNA_4PHFC[8]	N/A	714	579
SNA_4PHFC[8]	N/A	1324	1246
MNA_CBFC[8]	N/A	897	698
SNA_CBFC[8]	N/A	1078	960

同样支持 4PHFC 协议的设计,本文中的网络接口消耗的逻辑资源比 MNA 和 SNA 的平均值降低约 33%;而支持 CBFC 协议时,本文中的网络接口消耗的逻辑资源比 MNA 和 SNA 的平均值降低约 30%。

4.3 响应延迟统计

网络接口的响应延迟是网络通信延迟的组成部分,是衡量网络接口服务质量的重要参量. 响应延迟越大,则意味着包传输的网络延迟越大,同时链路的使用率越低. 本设计将发包的响应延迟定义为从 IP 核发起请求到包的第一个微片出现在输出端口的周期数,收包的响应延迟定义为从包的第一个微片出现在输入端口到 IP 核接收到收包请求的周期数。

在基于微码控制器的设计中,不同协议的响应延迟统计结果如表 4 所示。

表 4 不同协议的响应延迟统计

所用协议	发包响应延迟	收包响应延迟
PCC	2	2
CBFC	2	2
4PHFC	2	5
CBFC[8]	3.3	4
4PHFC[8]	2.7	8.7

表 4 的最后两行是文献[8]中的实验结果,它所提出的两种设计具有不同的响应延迟,这里取其平均值作为参考. 可以看到,本文提出的设计平均响应

延迟为 2.5,相比于文献[8]有很大改进. 在包长 1000 以上时,响应延迟的影响几乎可以忽略不计,链路带宽的利用率达到 99%以上。

5 结束语

文中提出了一种新颖的网络接口设计,该设计用微码控制器取代了传统的有限状态机,对片上存储资源的需求略有增长,而总的逻辑资源的消耗却大幅度降低. 基于微程序的设计使得多协议兼容更加容易实现,可以更好地适应多种应用环境. 实验结果表明,基于微码控制器设计的网络接口易于实现,耗用资源少,具有灵活性高、响应延迟低的特点. 该设计为今后 NoC 与实际应用相结合提供了有益的参考。

参考文献:

[1] Benini L, De Micheli G. Networks on chips: A new So-C paradigm[J]. IEEE Computer, 2002, 35(1): 70-78.

[2] M T Rose. The open book: a practical perspective on OSI [M]. Boston: Prentice Hall, 1990.

[3] Yong-Long Lai, Shyue-Wen Yang, Ming-Hwa Sheu, et al. A high-speed network interface design for packet-based noC[C]//International Conference on Communications, Circuits and Systems Proceedings. USA: Alexandria, 2006: 2667-2671.

[4] Sumant Sat he, Daniel Wiklund, Dake Liu. Design of a switching node(Router) for on-chip networks[C]//Proceedings of the 5th International Conference on ASIC. China: Beijing, 2003: 75-78.

[5] Seongmin N, Daehyun K, Vu-Duc N, et al. Performance and complexity analysis of credit-based end-to-end flow control in network-on-chip[C]//Parallel and Distributed Processing and Applications. Berlin: Springer, Heidelberg, 2007: 268-277.

[6] Beigné E, Vivet P. Design of on-chip and off-chip interfaces for a GALS NoC architecture[C]//IEEE International Symposium on Asynchronous Circuits and Systems. Queba, Montreal, 2006: 183.

[7] AMBATM Specification Revision 2. 0[S]. 1999.

[8] Zitouni B A A, Tourki R. Design and implementation of network interface compatible OCP for packet based NOC[C]//2010 5th International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS). Tunisia, 2010: 1-8.

作者简介:

周 帅 男, (1986—), 硕士. 研究方向为片上网络关键技术及其应用。