

可重构处理器阵列的系统级建模研究

潘 鹏,王 鹏,林水生

(电子科技大学 通信与信息工程学院,四川 成都 611731)

摘 要: 由于粗粒度可重构体系结构设计空间复杂,设计满足应用需求的 CGRA 需要建立系统级仿真模型进行性能评估.文中提出一种可重构处理器阵列的系统级模型,使用 SystemC 事务级语言实现建模.模型采用多层互连网络结构实现任意 2 个处理器间的通信,并且处理器的资源能够通过参数快速地进行配置.仿真实验表明,模型适用于应用算法到粗粒度可重构体系结构映射的模拟仿真.

关键词: 粗粒度可重构体系结构;处理器阵列;SystemC;事务级建模

中图分类号: TP 391.9

文献标识码: A

文章编号: 1000-7180(2011)11-0085-04

Research on System-Level Modeling of Reconfigurable Processor Array

PAN Peng, WANG Peng, LIN Shui-sheng

(School of Communication and Information Engineering,

University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract: Due to the complexity of coarse-grained reconfigurable architectures, design of CGRA require system-level modeling and simulation-based performance exploration. This paper proposes a system-level model for reconfigurable processor array, which is constructed with SystemC transaction level language. The model is based on a hierarchical interconnect network, in which any two processors can initiate communication and resources in processors can be quickly configured by parameters. The result of the simulation shows that the model can be used in the simulation of application algorithms mapping onto CGRA.

Key words: coarse-grained reconfigurable architectures; processor array; SystemC; transaction level modeling

1 引言

可重构体系结构作为一种并行体系结构,由于同时兼具通用处理器的灵活性和专用集成电路的高效性,被视为解决应用加速的高性能方案而备受关注^[1].传统的细粒度可重构体系结构(Fine-Grained Reconfigurable Architectures, FGRA)如 FPGA,用于实现字节级操作的算法应用时存在着资源开销大、功耗高的缺点.相比 FGRAs,采用粗粒度可重构体系结构(Coarse-Grained Reconfigurable Architectures, CGRA)实现字节级操作的算法可获得更快的运算速度以及更少的功耗和资源开

销,更能满足数据密集型应用算法的实现需求.

在 CGRA 设计过程中,由于需要实现应用领域中的一组算法,增加了设计空间搜索的复杂度.针对特定应用领域设计满足性能/开销的 CGRA,需要搭建仿真平台进行设计空间搜索^[2],系统地评估设计方案.处理单元阵列作为 CGRA 的运算核心部分,其资源的数量、支持的操作类型和实现应用算法采用的映射策略等是影响系统性能和开销主要考虑的因素.文中利用 SystemC 搭建一种可重构处理器阵列系统级模型,该模型能够实现处理单元参数化的配置,采用多层互连网络结构实现任意 2 个处理单元间的相互通信,能够用于可重构计算仿真平台

的搭建以及体系结构和映射策略的研究^[3-6].

2 可重构处理器阵列的 SystemC 建模

2.1 模型整体结构

文中模型采用一种多层的混合互连网络结构,支持任意 2 个处理单元间的直接通信,如图 1 所示. PE 是支持执行流水的处理器,为可重构处理器阵列的主要计算资源. 而互连通信资源包括 PE 间的连接通道和路由单元,路由单元与路由单元间以及路由单元与 PE 单元间按层次互连,底层的路由单元与 2×2 的 PE 单元相连,上一层的路由单元与下一层的 4 个路由单元相连,相邻的 PE 之间可以直接通信,距离较远的 PE 通过路由单元实现通信. 路由单元采用静态路由表实现路由,PE 间的路径具有唯一性. 模型的计算资源不局限于 16 个 PE,如果应用需要更多的计算资源,可以相应地增加路由单元来实现更多 PE 间的通信,例如模型计算资源为 64 个 PE,则相应的路由单元需要 21 个.

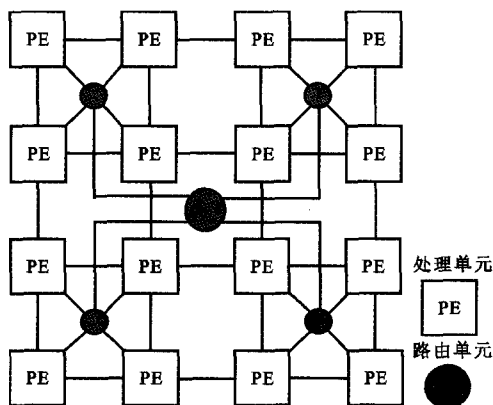


图 1 4×4 的可重构处理器阵列模型结构

和常见可重构阵列的互连网络模型相比较,如基于总线的多级网络和 2D-mesh 的片上网络,前者由于总线带宽的限制,难以实现任意 PE 之间的通信,对应用的映射约束大;后者虽然灵活性高,但需要较多的路由资源,同时也可能需要较复杂的路由算法支持,导致模型实现复杂. 文中的多层互连网络模型,不但能够方便实现任意 PE 间的通信,同时由于相邻的 PE 直接相连通信,不用经过路由单元,使得阵列整体的吞吐量能够得到保证. 模型整体扩展性较好,实现简单.

2.2 处理单元的实现

处理单元是可重构阵列的运算核心,具体的结构和资源大小要根据应用的需求设计和优化. 文中实现的 PE 模型结构与传统的 RISC 处理器核相似,如图 2 所示. PE 主要由程序计数器、指令存储器、通

用寄存器、ALU、流水寄存器以及输入和输出寄存器,支持取指(IF)、译码(ID)、执行(EX)、写回(WB)4 级流水. 其中输入和输出寄存器分别于输入和输出端口连接,起到缓存数据的作用. ALU 支持 32bit 粗粒度运算操作,可以完成加、减、移位等算术逻辑操作和存储器访问的读写操作. 应用映射到处理单元阵列后,指令存储中的指令(含有配置信息)经过取指译码后实现 I/O 端口的选择.

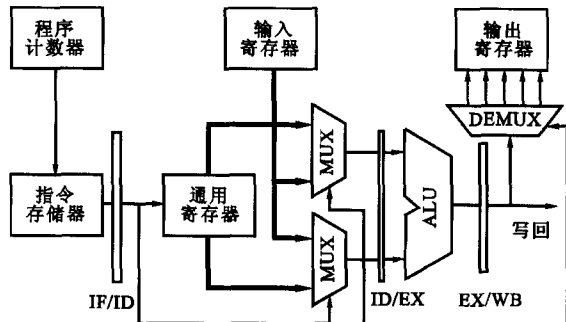


图 2 处理单元的结构框图

为了便于实现模型资源配置的参数化和指令集的可定制,文中采用基于 SystemC 的结构描述语言 ArchC^[7]实现 PE 的模型. ArchC 用两个组成部分描述一个设计的结构:指令集结构描述(AC_ISA)和结构资源描述(AC_ARCH),在 AC_ARCH 中可以对 PE 单元中的资源数量进行参数化的设置,PE 支持的操作指令用 AC_ISA 进行制定.

```
AC_ARCH(PE){
    ac_wordsize 32; // 设置操作字宽为 32bit
    ac_mem IM;24K; // 设置指令存储器大小为 24K
    // 设置通用寄存器的深度为 64,字宽为 32bit
    ac_regbank GPR<32>; 64;
    ac_reg IPR, OPR; // 输入输出寄存器申明
    ac_pipe pipe={ IF, ID, EX, WB }; // 设定 4 级流水
    ac_format Fmt_IF_ID="%npc;32"; // 定义标识符
    .....
    ac_reg<Fmt_IF_ID> IF_ID; // 各级流水寄存器申明
    ac_reg<Fmt_ID_EX> ID_EX;
    ac_reg<Fmt_EX_WB> EX_WB;
    ARCH_CTOR(PE){
        /* 设置 PE 模型使用的操作指令结构描述文件 */
        ac_isa("PE_isa.ac");
    };
};
```

图 3 PE 资源的 ArchC 描述

图 3 所示代码为 PE 模型的资源设置的顶层描述示例. 通过顶层的 AC_ARCH 可以方便的设置 PE 中各个模块资源的大小和流水的级数,在 PE_

isa. ac 文件可以对 ALU 支持的操纵指令的格式和行为等信息进行定制,文中模型主要设置了常用的算术逻辑运算指令和访问存储器操作的指令。

2.3 路由单元的实现

路由单元主要包含有路由控制器、交换开关、输入和输出缓存模块,如图 4 所示。作为路由单元中的核心控制部分,路由控制器负责控制交换开关,根据输入和输出缓存队列的状态和路由表信息来实时配置交换开关。交换开关完成配置后,输入缓存中的数据将被发送到目的端口的输出缓存。为了使模型能够方便连接阵列外的局部存储器,在底层路由单元中提供额外的 I/O 端口连接局部存储器,让连接路由单元的 4 个 PE 共享使用一个局部存储器。

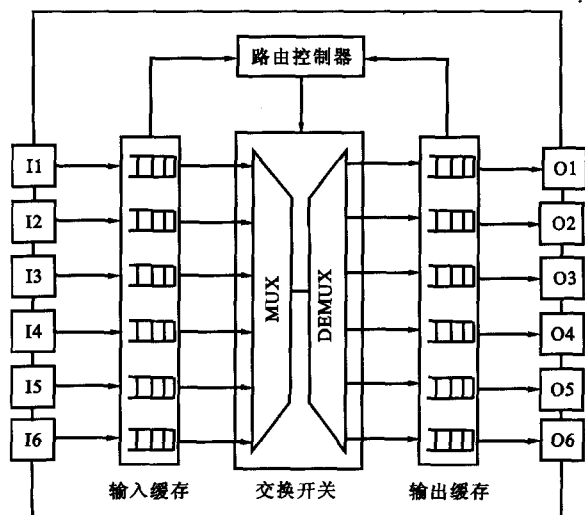


图 4 路由单元的结构

路由单元模型采用静态路由表的方法实现路由。在路由表中,用不同的 ID 号表示各个处理单元和路由单元端口,如图 5 所示。在 4×4 的 PE 阵列中,用 2 位数字标示不同位置的 PE,用 3 位数字标示各个路由单元的端口,最低位数字表示端口号,0 号端口提供给局部存储器使用,5 号端口连接上层路由单元,高 2 位数字为路由单元编号。路由单元收到数据包时,先查找路由表上是否有数据包的目的地的 ID,如果有就将该数据包向对应的端口发送,否则就向上层路由单元路由的端口发送。后继路由单元一直重复上述工作,直到数据包到达目的资源节点。

路由单元的 SystemC 模型的实现形式和处理单元不同,由于在 SystemC 中通道是实现通信的载体,结构层次明显的通信模块如总线可以设计为分层通道^[8],因而作为通信资源的路由单元可以搭建为分层通道的模型。在路由单元模型内,输入、输出

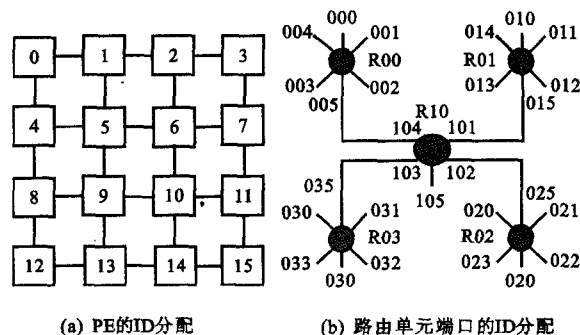


图 5 4×4 可重构阵列的资源单元 ID 分配

缓存为深度可设置的 FIFO 通道队列,PE 模块要实现路由模块的读写操作,需要设置导出端口 `sc_export` 才能调用 FIFO 队列通道中定义的读写方法,因而输入、输出缓存模块中每个 FIFO 通道的接口都相应地绑定了一个导出端口。

2.4 互连通信的实现

文中的多层混合互连网络,采用 2 种通道模型实现。PE 间的全局互连通信通过 2.2 节的路由单元的分层通道模型实现,每个 PE 模块的全局通信端口与路由单元通道的接口绑定,通过端口调用通道的接口方法完成数据的收发。相邻 PE 间的局部通信通过 SystemC TLM 库定义的 `tlm_fifo` 通道实现,由于 `tlm_fifo` 通道为单向的通道,所以 PE 单元的输入输出端口分别和不同的 `tlm_fifo` 通道关联。`tlm_fifo` 通道除了实现 PE 间通信外还要负责运算的同步,发送方的 PE 需等待接收方 PE 的寄存器有可用空间才发送数据,接收方的 PE 也需等待接收到要处理的数据才执行操作,为了准确模拟 PE 之间握手通道的机制,每条 `tlm_fifo` 通道的队列深度设为 1。

3 仿真实验

3.1 仿真方法

文中的可重构处理器阵列模型主要用于 CGRA 设计初期的结构探索和应用映射策略评估,需要将具体应用算法映射到模型上仿真,根据应用需求评估和优化设计方案,仿真实验将利用 3 种常见的数据密集型算法在模型上做实际映射分析。

仿真流程如图 6 所示。由于实现具体的算法还需要存储资源,因而在处理器阵列模型的基础上增加存储资源模型,在每个底层路由单元上连接上一个双端口的局部存储器相连,由路由单元连接的 4 个 PE 共享。在仿真开始前,配置各个资源模型的个数和参数,本仿真实验采用 8×8 个 PE 的阵列模

型,每个 PE 的资源按照图 3 配置,3 层路由单元的互连网络结构,对应的路由单元和局部存储器数目分别为 21 和 16. 由于存储器对性能的影响不是本仿真实验研究重点,因而不限制局部存储器的大小,能够存放实验算法所需的所有数据,存储器的访问延迟为 1 周期.

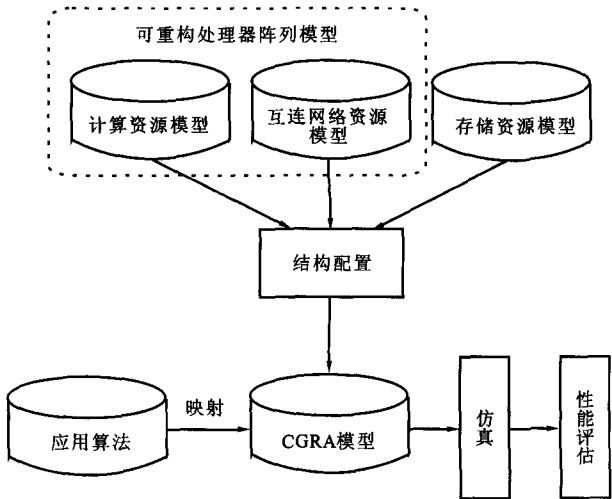


图 6 实验仿真流程

实验采用的 3 种算法分别为: 8×8 2D-DCT 行列分解算法、FFT 算法和 Sobel 边缘检测算法. 在保证循环体能够流水执行的情况下,对这 3 种算法采用 2 种映射策略手动映射到模型上:①核心循环体直接映射;②核心循环分解映射.

3.2 仿真结果

3 种典型算法在 2 种映射策略下,文中的仿真模型执行的性能如表 1 所示. 仿真记录下每个算法涉及到的操作 OP、占用的计算资源数 UPEs、所需执行乘法操作的计算资源数 MULTs、计算资源利用率 PEU 以及执行时间 ET.

利用仿真结果可以对应用算法的映射策略进行性能分析. 从表 1 中可知,采用核心循环体直接映射的策略(MP1)资源利用率较低,可以使用较少的资源实现算法,在速度满足应用要求的情况下,适合多个应用算法映射到同一处理器阵列情况. 相比于 MP1,核心循环分解映射策略(MP2)使用更多的资源提高计算并行性,循环体分解 1 次使用的资源增加 1 倍,速度也将近提高 1 倍. MP2 资源利用率高,如 2D-DCT 算法 PEU 达到了 96.8%,适用于处理器阵列上只映射了一个或较少的算法的情况.

通过仿真结果可以针对应用算法的需求优化处理器阵列的资源. 为了减少面积的开销,根据仿真记录下算法涉及的操作,精简 PE 支持的操作集合,同时限制乘法器的数量. 对于实验的 3 种算法,除了访问存储器的存取操作外,PE 只需执行表 1 中的 5 种操作类型. 在采用的 2 种映射策略中,参与乘法操作的 PE 数目最多需要 16 个,因而最多只需要 16 个乘法器便能可以满足算法的资源需求.

表 1 典型算法在仿真模型的性能列表

算法	OP	UPE		MULT		PEU		ET(cycle)	
		MP1	MP2	MP1	MP2	MP1	MP2	MP1	MP2
2D-DCT(8×8)	add/sub/mul	31	62	5	10	48.4%	96.8%	88	48
512 点 FFT	add/sub/mul	10	40	4	16	15.6%	62.5%	4617	1161
Sobel ED(64×128)	add/sub/abs/gte	18	36	0	0	28.1%	56.3%	16396	8204

注: add:加; sub:减; mul:乘; abs:取绝对值; gte:大于或等于.

4 结束语

文中提出了一种由处理单元和路由单元组成的可重构处理器阵列模型,利用 SystemC 实现了处理单元模块、路由单元模块和多层互连网络的系统级建模. 处理单元模块具有资源配置参数化和指令集可定制的特点,因而能够根据应用的需求方便地配置模型进行仿真. 同时采用路由单元搭建的多层互连网络实现任意 2 个处理单元间的通信,为算法的映射提供更高的灵活性. 仿真实验表明,仿真模型能够用于评估可重构处理器阵列的应用算法的映射策

略和资源设计方案,为粗粒度可重构计算平台的仿真建立了基础.

参考文献:

[1] Katherine Compton, Scott Hauck. Reconfigurable computing: a survey of systems and software[J]. ACM Computing Surveys, 2002, 34(2):171-210.
[2] Hauser J, Wawrzynek J. Garp: a MIPS processor with a reconfigurable coprocessor[C]// IEEE Symposium on FPGAs for Custom Computing Machines. USA: Napa Valley, 1997:12-21.

(下转第 93 页)

传统时钟设计方案,顶层进行时序收敛后,对于子设计的影响较大,误差为 $54.41914 - 1.79679 = 52.62235\text{ps}$ 。

以上数据均在 Cadence 的 SocEncounter 平台下运行获取的。

6 结束语

这种时钟设计方法已成功应用在某大规模通信 SOC 芯片的设计中^[8-9]。实践证明,这种层次化模式下的新型时钟方案有如下优点:

(1) 分频电路通过同步器产生,无需在 SDn 内部实现二分频电路,实现简单;

(2) 进行顶层集成时,只需对单比特的 phase_sync 信号进行时序收敛,降低了时序收敛的难度;

(3) 顶层无需进行异步复位设计,降低了顶层复位设计给时序造成的难度;

(4) 每个子设计只有一个时钟输入端口,因此当 SDn 单独完成时序收敛后,顶层时序收敛对 SDn 影响很小,大大减少了迭代次数。

参考文献:

[1] Synopsys. PrimeTime fundamentals user guide[S]. Version C-2009. 6.

[2] Synopsys. Advance timing analysis user guide[S]. Version C-2009. 6.

[3] Michael D Ciletti. Advanced digital design with the verilog HDL[M]. 北京:电子工业出版社,2010.

[4] Clifford E Cummings, Don Mills. Synchronous resets? asynchronous resets? [EB/OL]. [2001-03-12]. <http://www.sunburst-design.com/papers/>.

[5] Clifford E. Cummings. synthesis and Scripting techniques for designing multi-asynchronous clock designs [EB/OL]. [2001-03-12]. <http://www.sunburst-design.com/papers/>.

[6] 千路,林平分. ASIC 后端设计中的时钟偏移以及时钟树综合[J]. 半导体技术,2008(6):33.

[7] 刘辉华,刘振,李蜀霞,等. 层次化时钟网络设计研究[J]. 微电子学与计算机,2008,25(11):52-54.

[8] 李蜀霞. 基于 ASIC 实现雷达信号处理芯片的后端设计[D]. 合肥:电子科技大学,2008.

[9] 于倩. ASIC 芯片的层次化后端实现[D]. 上海:复旦大学,2008.

作者简介:

王丹丹 女,(1977-),硕士研究生,讲师. 研究方向为检测技术与信号处理、图像处理技术

(上接第 88 页)

[3] Mei B, Vernalde S, Verkest D, et al. ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix [C]// International Conference on Field-programmable Logic and Applications. Los Angeles, 2003:61-70.

[4] Panesar G, Towner D, Duller A, et al. Deterministic parallel processing[J]. International Journal of Parallel Programming, 2006, 34(4):323-341.

[5] Kissler D, Kupriyanov A, Hannig F, et al. A highly parameterizable parallel processor array architecture [C]// IEEE International Conference on Field Programmable Technology. Thailand, 2006:105-112.

[6] 李焯挺,梁利平. 一种基于 SystemC 的片上网络建模与仿真方法[J]. 微电子学与计算机,2010,27(3):78-82.

[7] 奚杰,陈杰,朱玥. 利用 SystemC 实现多核系统的快速建模[J]. 微电子学与计算机,2010,27(7):214-217.

[8] Rigo S, Araujo G, Bartholomeu M, et al. Archc: a systemC-based architecture description language [C]// 16th Symposium on Computer Architecture and High Performance Computing SBAC-PAD. Brazil, 2004:66-73.

作者简介:

潘 鹏 男,(1986-),硕士研究生. 研究方向为可重构计算、事务级建模等。

王 鹏 男,(1985-),硕士研究生. 研究方向为嵌入式系统设计、可重构计算等。

林水生 男,教授. 研究方向为嵌入式系统、通信专用集成电路与 SoC 设计。