计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20150889
54(5):1121-1129, 2017

粗粒度可重构 SoC 层次化配置存储器设计

沈剑良[1,2] 李思昆[2] 刘 磊[2] 王观武[2] 汪 欣[1] 刘勤让[1]
[1](国家数字交换系统工程技术研究中心 郑州 450002)
[2](国防科学技术大学计算机学院 长沙 410073)
(shenjianliang@outlook.com)

Hierarchical Configuration Memory Design for Coarse-Grained Reconfigurable SoC

Shen Jianliang[1,2], Li Sikun[2], Liu Lei[2], Wang Guanwu[2], Wang Xin[1], and Liu Qinrang[1]
[1](*National Digital Switching System Engineering & Technological R&D Center*, *Zhengzhou* 450002)
[2](*College of Computer*, *National University of Defense Technology*, *Changsha* 410073)

Abstract The generate efficiency and quality of configuration information directly affect the operation effect of the coarse grained reconfigurable SoC. Since the traditional approach treats the configuration memory as a whole, and each processing unit needs to read configuration information from the memory, the operation efficiency is low and the power consumption is large. In this paper, a low power hierarchical configuration information storage architecture is designed, which divides configuration information into separate operating configuration information and interconnect configuration information, and then generates the configuration information based on the context. Experimental results show that the configuration information generation method proposed in this paper can reduce power consumption of 23.7%-32.6% while keeping the same performance. At the same time, because of the separation of the operation and the configuration information, the configuration information capacity is small, so it has a great advantage in configuration speed and performance.

Key words coarse grained reconfigurable SoC; configuration information memory; hierarchical; low power; configuration information generation method

摘 要 配置信息的生成效率与质量直接影响着粗粒度可重构 SoC 结构的运行效果.传统的方法将配置信息作为一个整体存储器,每个处理单元在需要配置信息时都要从该存储器读取配置信息,运行效率低下且功耗较大.为降低配置信息生成方法的功耗,设计了一种低功耗层次式的配置信息存储器结构,将配置信息分为相互独立的操作配置信息和互连配置信息存储器两部分,实现了不同层次上的重构,最后根据上下文优化配置信息生成.实验结果表明:在运行性能不变的情况下,提出的配置信息生成方法功耗可以减少 23.7%~32.6%.同时,由于操作和互连配置信息相分离,使得每次需要配置的存储器容量较小,在配置速度和性能上也有很大的优势.

收稿日期:2015-09-30;**修回日期**:2016-08-08
基金项目:国家"八六三"高技术研究发展计划基金项目(2014AA01A704);国家自然科学基金创新群体项目(61521003);国家自然科学基金面上项目(61572520)
This work was supported by the National High Technology Research and Development Program of China (863 program) (2014AA01A704), the Innovation Group Program of the National Natural Science Foundation of China (61521003), and the General Program of the National Natural Science Foundation of China (61572520).

关键词 粗粒度可重构 SoC;配置信息存储体;层次化;低功耗;配置信息生成方法

中图法分类号 TP391

粗粒度可重构多核 SoC 系统架构集成了多个处理器核,极大地增强了系统的处理能力,但这也对整个 SoC 系统的片上访存与核间通信提出了更高要求.在粗粒度可重构 SoC 系统设计中,配置信息极其关键,它的生成效率与质量直接影响着可重构结构的运行效果.配置信息从硬件上看是一个存储器,可重构结构中的每个处理单元从该存储器读取配置信息,通过该配置信息控制处理单元的操作和数据流向.配置存储器的结构性能在很大程度上决定了整个可重构 SoC 系统实现应用的性能,同时也影响着系统编译器的复杂度设计,而功耗则已成为当前粗粒度可重构 SoC 系统架构设计继性能、灵活性之后又一重要指标[1],低功耗的配置存储器设计是当前研究的热点之一.

配置信息生成方法研究国内外并不多,一般将其作为编译映射的一个子问题进行描述.配置信息往往较多,一方面对静态存储时要求存储空间较大;另一方面在可重构时每次要提供的数据较多,对配置数据带宽的要求较大.在配置信息存储器设计上,通常大多数的粗粒度可重构系统通过每个周期向处理单元发送配置信息,如图 1 所示[1]:

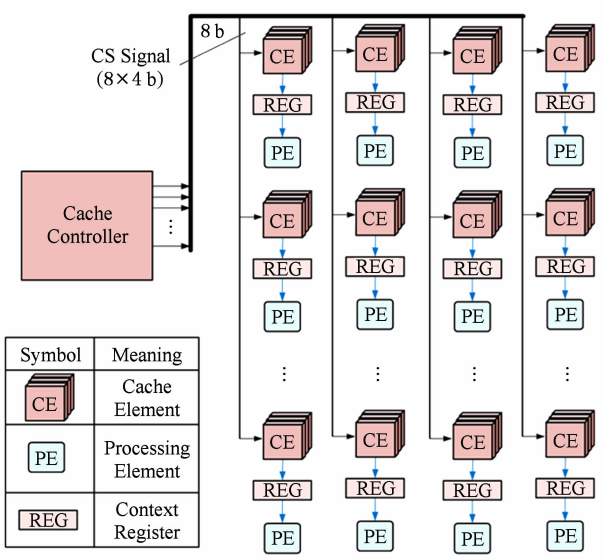


Fig. 1 Distributed configuration memory architecture

图 1 分布式配置存储器结构[1]

文献[1]在这种通用分布式配置信存储器结构中按空间映射和时间映射将配置信息存储器分成时间映射配置信息存储器和空间映射配置信息存储器.前者由于流水线作用,只需要第 1 列使用配置信息配置功能,前一列运行结果后将配置信息和结果打包发送到下一列中;后者则每个处理单元都需要配置信息,但是一般而言其总的配置信息层数较少,因此可以将存储器设计得层数较少.通过与优化的映射方法配合取得了一定的功耗减低和配置信息总量减少.文献[2]中 Morphosys 采用图 1 中的结构,由于采用 SIMD 形式挖掘数据并行性,每列的 8 个处理单元所执行的功能相同,因此同列 8 个处理单元共用一条配置信息,每条配置信息 32 b,总共每个周期需要 32×8 b 配置信息.

文献[3]提出一种一般的粗粒度可重构阵列模板,这种阵列结构不仅包括一个通常的配置信息存储器,而且在每个处理器单元中还包括一个配置信息 Caches.只有当 Caches 中的配置信息不足时,才从处理单元外的配置存储器调取配置信息,这种结构大大减少了配置信息的访存频率.尽管如此,它也带来了配置空间这个新的问题,最好的情况是一次程序映射的所有配置信息都可以存储在 Caches 中,在一些简单循环映射中这种情况是存在的.图 2 为其提出的一般粗粒度可重构阵列模板.

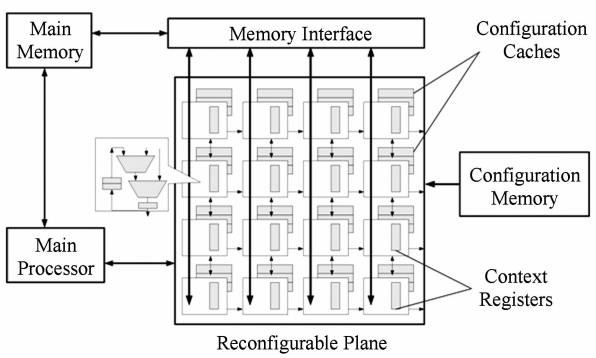


Fig. 2 General dynamic reconfigurable architecture template

图 2 一般动态可重构体系结构模板

文献[4]在 Montium 项目中提出一种层次化的配置信息存储器模型.Montium 项目中可重构处理器由 5 个功能强大的处理单元组成流水线结构.它的配置信息存储器包括单个算术逻辑单元(arithmetic logical unit, ALU)配置存储器(配置单个 ALU 在一次程序映射中执行 8 个不同的操作)和 5 段 ALU 配置存储器(配置 5 段可重构流水线在一次程序映射中可以执行 32 个不同的数据子图).

然而配置信息对可重构结构的整个性能和功耗

非常重要,如图 3 所示为文献[1]分析的传统的可重构阵列在运行 2D-FDCT 程序时,可重构阵列各个部分消耗的功耗占总功耗的饼状图.从图 3 中可以看出,功能处理单元阵列所占功耗最大比例最大为 48%,其次则为配置信息存储器占了 43%,由于 CGRA 广泛挖掘数据级并行的局部性,尽管配置信息存储器和帧缓存使用的是相同的存储器技术,但帧缓存(frame buffer, FB)功耗只有 8%左右.

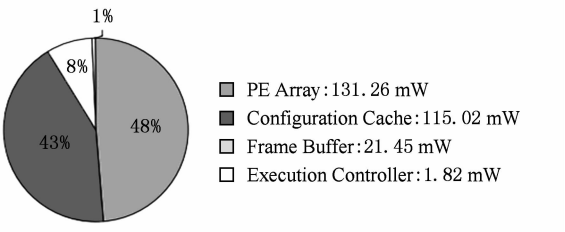


Fig. 3 CGRA running 2D-FDCT power consumption pie chart

图 3 CGRA 运行 2D-FDCT 时功耗饼状图[1]

由于对配置信息存储器频繁的读操作,导致了配置信息存储器所消耗的功耗巨大,为降低整个 SoC 系统的配置代价,在有效地提升所生成的配置信息的配置性能的同时,本文研究了低功耗的层次化的配置存储器设计及其配置信息生成方法.

1 层次化配置信息存储器设计

传统可重构处理器配置信息从主处理器的外存通过直接内存存取(direct memory access, DMA)载入粗粒度可重构处理阵列(coarse grained reconfigurable array, CGRA)的配置信息存储器,这部分一般可以与主处理器的计算叠加隐藏,因此一般不考虑其性能影响.当 CGRA 运行起来后,配置信息依旧存储在配置信息存储器中,传统的配置信息发射方法是每个处理单元通过存储读操作访问存储器,同样这部分访存操作也可以与 CGRA 的计算叠加,但是频繁的读操作会带来 3 个问题[5-6]:

1) 增大的数据带宽要求.由于处理单元个数众多,如果每个处理单元都需要一次访存操作的话,整个 CGRA 阵列要求的数据带宽就非常大了.

2) 面积.由于访存单元往往占用较大面积,多个访存单元将导致阵列面积快速增加.

3) 功耗.存储器读操作虽然与计算重叠,可以认为其不对性能产生影响,但是读操作功耗较大,频繁的存储器读操作必将带来较大的功耗.

映射到 CGRA 可重构阵列的循环核心天然地具有循环重复特性,由于 CGRA 中配置信息描述系统操作和互连特性,因此配置信息也天然地具有这种循环性和重复特性.使用数据局部性是获得性能和功耗降低的重要方法.如在 CPU(central processing unit)中利用数据局部性使用高速缓存保存当前数据,因为后续操作有非常大的可能性还会用到当前数据,所以取得性能提升和功耗降低.层次化的配置信息存储器设计思想正是使用这一方法,在配置信息生成方面获得改进.通过局部性原则,在 CGRA 配置信息生成方面获得功耗改进的原理[5]是:

1) 使用局部寄存器中数据比使用局部存储器中数据功耗更小;

2) 使用局部寄存器中数据比使用非局部寄存器中数据功耗更小;

3) 使用局部存储器中数据比使用非局部存储器中数据功耗更小;

因此在设计中分层次使用局部寄存器、局部存储器和全局存储器 3 个层次组建配置信息存储器.当配置信息位于局部寄存器中时,尽量使用局部寄存器配置信息;当配置信息不在局部寄存器中时,尽量使用局部存储器中配置信息;只有配置信息不在前两处时,才从全局存储器中读取配置信息;当全局存储器中也没有配置信息,就必须从 SoC 外存中调入配置信息.在设计中,期望信息位置越靠近局部寄存器越好.局部寄存器性能较好,但造价更为昂贵容量较小,全局寄存器虽然性能较差但造价便宜容量更大.对于领域应用满足特定领域程序映射要求的配置空间大小情况下,减少局部寄存器容量可以大大减少造价而不影响性能.图 4 给出了传统可重构阵列结构示意图[1]:

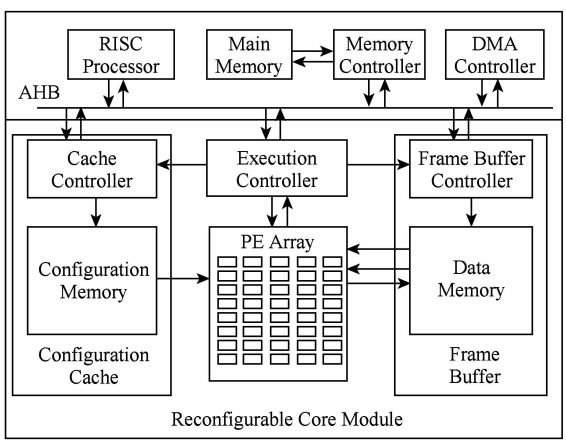


Fig. 4 Traditional reconfigurable system

图 4 传统可重构系统[1]

从图 4 可以看出配置信息作为一个整体存储器,虽然实现上可能是分布,是多 bank 存储器,但是其设计思想仍是采用一个全局外部存储器,每个处理单元在需要配置信息时,都是从该存储器读取配置信息.

本文研究的配置存储器设计主要分 3 个步骤[5]:

1) 设计一个高效层次化的配置信息存储器. 以往在可重构结构的配置存储器设计上研究较少,一般都将其设计为一个简单的存储器,但这种设计方法忽略了配置信息广泛存在的局部性,效率低下从而导致配置时间开销较长. 因而本文引入了层次化的配置存储器设计,将其设计为 3 个层次:局部配置寄存器、局部配置存储器和全局配置存储器. 这种设计充分利用了配置信息的局部性,可以带来功耗的较大改善.

2) 设计互连配置信息存储器和运算操作配置信息存储器,这其实是对运行时系统的运行模式进行了设计,这种设计需要与编译流程中映射结果进行衔接,与可重构 SoC 系统架构设计紧密相关. 从可重构流水线中提取体系结构模式集,并以该模式集为输入,以循环核心为另一输入进行匹配. 配置信息存储器主要分成 2 个存储器,即互连配置信息存储器和运算操作配置信息存储器,利用这 2 种配置信息重构的频率和开销,结合层次化配置信息存储器,设计了各层次上不同容量的设计方案. 这种设计承接了映射的结果,很好地反映了系统运行时的真实信息,同时也带来功耗和面积的改善.

3) 配置信息的生成. 虽然可重构结构灵活性较高,本文在应用算法分析和映射时采用体系结构模式集为基础进行匹配,因而其可重构流水线并不复杂. 互连配置信息和运算操作配置信息分离的设计,减少了配置信息的传输时间(配置开销),从而提升了配置信息生成的整体性能.

图 5 给出了本文设计的层次化配置信息存储器[5]:

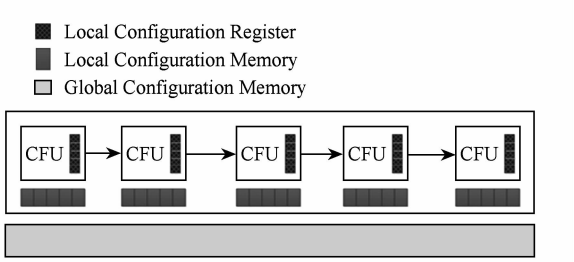


Fig. 5 Hierarchical configuration information memory for reconfigurable pipeline

图 5 可重构流水线层次化配置信息存储器[5]

图 5 中可以看出,本文的可重构流水线采用层次化配置信息存储器,3 个层次由内到外分别为局部配置寄存器、局部配置存储器和全局配置存储器. 由内到外 3 个层次速度越来越慢,容量越来越大,造价越来越高. 靠内的存储器均可向靠外的存储器调入新的配置信息用来替换当前配置信息.

2 互连配置及运算操作配置信息存储器设计

编译映射的最终结果是可重构流水线可以识别和执行的配置信息,因此在配置信息生成时也应适应编译映射方法. 基于模式的可重构编译方法,以体系结构模式集为被匹配对象,以循环核心为待匹配对象,最终循环核心被一系列体系结构模式集覆盖,每个体系结构模式集匹配成功后还有操作模式补充说明此前节拍各个处理单元执行的操作. 之所以选择使用互连模式作为被匹配对象而不是操作模式主要基于 3 个原因[5,7].

1) 对于可重构流水线定制完成以后,有限的处理单元之间的互连情况可以枚举出来,而操作模式枚举的个数要远远大于互连模式的个数,被匹配集如果过大,会急剧增加程序映射时间,增加编译映射难度.

2) 一般而言处理单元可执行的为几类简单操作和特定的定制指令,操作个数有限且较少,在配置信息中用于表示操作的一般为 6～7 b,而用于表示互连的往往达到 20 几位. 一个互连模式可以对应多个操作模式,当重构操作时,需要的配置信息则较少;反过来,如果一个操作模式对应多个互连模式,当重构互连时,需要的配置信息则较多. 配置信息的多少直接影响到功耗.

3) 简化编译映射. 由于图同构是一典型的 NP 难问题,目前还没有有效的方法在多项式时间内解决该问题. 图同构问题会随着被匹配集的个数、被匹配图中节点的个数急剧增加. 从互连模式出发,一方面匹配集的个数较少,另一方面每个匹配图中最大节点数为处理单元个数.

综上所述,良好的编译映射方法可以减少配置信息总量,减少因配置信息重构而造成的功耗;同时一个良好的配置信息生成方法也可以简化编译映射的难度. 两者需要同时考虑,互相支持才能使整个可重构编译流程高效[8-9].

为了支持互连模式和操作模式,本文在设计配置信息存储器时将其分开设计为互连配置信息存储器和运算操作配置信息存储器,分别对应互连模式

和操作模式,具体设计又有所不同.一个互连模式可以对应多个操作模式,但是在某一节拍,处理单元执行的功能是确定的.操作模式即为当前互连模式中使用的处理节点的当前节拍操作.正是因为一个互连模式对应多个操作模式,所以操作模式重构的可能性更大,而互连模式相对比较固定,互连模式相对比较稳定,因此操作配置信息存储器应该设计得更靠内.另一方面操作模式使用的位数较少,而互连模式描述的信息较多使用的位数较长,需要的互连配置存储器容量较大,因此在设计互连配置信息存储器时应该设计得靠外.

综上分析,本文计划将配置信息存储器在上小节的分层配置信息存储器基础上作如下改进,如图 6 所示[5]:

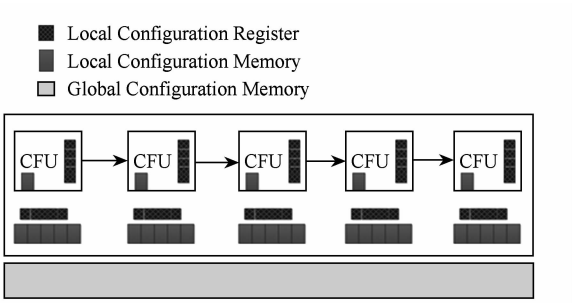


Fig. 6 Operating configuration information and interconnect configuration information memory

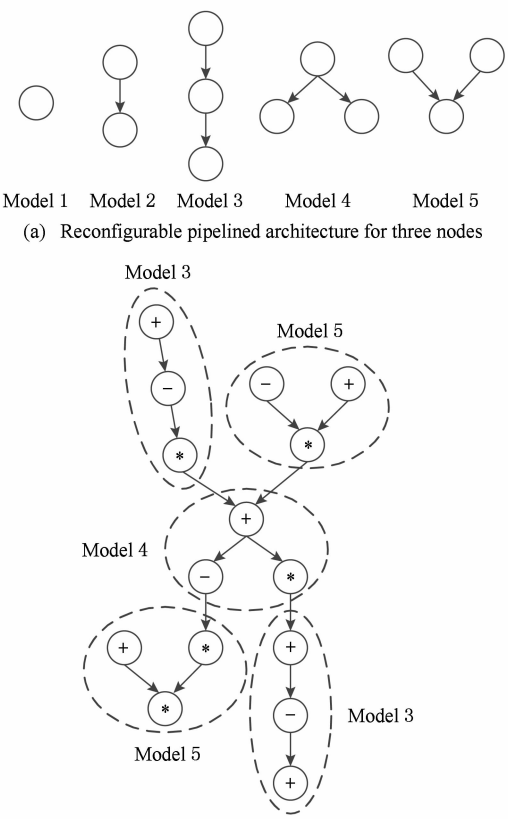
图 6 互连配置信息存储器 & 操作配置信息存储器[5]

从图 6 中可以看出,存储器分为操作配置信息存储器、互连配置信息存储器,以及既包含操作配置信息又包含互连配置信息的全局存储器.各个存储器的大小相对地表现了存储器的容量.从图 6 中可以看出在处理器内部的局部寄存器中,有大量的操作配置信息存储器和少量的互连配置信息存储器,这是因为操作往往重构的比较频繁,而互连模式相对比较稳定.在局部存储器这一层次都存在较多的配置信息存储器,互连配置信息更多一点,这是因为互连信息往往较大.图 6 中虽然是分布式的存储器布局,但这些分布式存储器共同决定了一个模式.对于全局配置存储器既包含了互连配置信息,又包含了操作配置信息,当前两层配置空间不足时即从该处调取.

3 配置信息生成方法

依据第 1,2 节的研究方案设计好配置信息以后,配置信息的生成方法是本文重点研究的问题.良好配置信息生成方案的关键在于充分利用编译映射方法中的隐含信息和发挥配置信息存储器的结构特点.配置信息生成的技术指标主要是配置信息总的位数,位数越多说明需要的存储空间更大,重构开销更大,表现为系统指标就是面积更大和功耗更大.

基于模式的可重构编译映射方法有许多隐含信息,可以使用这些隐含信息最大程度地减少配置信息的位数.基于模式的可重构编译映射方法使用首先互连映射,以体系结构互连模式集为被匹配对象,以循环核心为待匹配对象[10-11].图 7(a)为 3 个节点的可重构流水线体系结构互连模式集[5]:



(a) Reconfigurable pipelined architecture for three nodes

(b) Interconnection mapping diagram

Fig. 7 Reconfigurable pipelined architecture and interconnection mapping diagram

图 7 可重构流水线体系结构互连模式及映射示意图

3 个节点的可重构流水线体系结构互连模式集只有 5 个,使用 3 b 就可以标识这 5 种不同情况,对于每一个模式,如果匹配成功,还需再加上当前时刻的操作模式.而对于某一特定循环核心的映射,可能只使用了其中的某几个模式,这样互连模式个数就更少了.一个互连模式隐式地就包含了体系结构的互连信息,在生成配置信息时,可以不用再生成全部的互连配置信息,而是通过模式标识号就可以表示当前互连情况,这样大大减少了生成的配置信息数量.而具体的互连信息可以在编译映射时,将所有使用到的模式及其互连信息建立联系,将需要使用的

具体的互连信息初始化到全局配置信息存储器中，配置信息则用相应的模式标识表示[10-11].

前面的步骤中模式，程序都以图的形式表示，在配置信息中不能直接表示图. 一般都是通过多少位二进制数据表示，在读入配置信息后进行解码. 在本文设计中同样使用这种形式，但不同的是本文的每个互连模式都包含了确定的隐含信息，这些信息虽同样采用二进制数据表示，通过预载入的方式载入，但以标号标识，以便下次在进行重构时只需通过标号进行载入即可. 这种设计在重构空间满足程序的情况下，将大大减少功耗.

4 可重构配置信息的设计与实现

为了验证层次化配置信息存储器，本文使用Verilog语言实现了图8所示的可重构密码处理SoC的可重构部分. 该结构包括16个应用定制的处理单元，每个处理单元根据密码应用领域定制特定的功能[5].

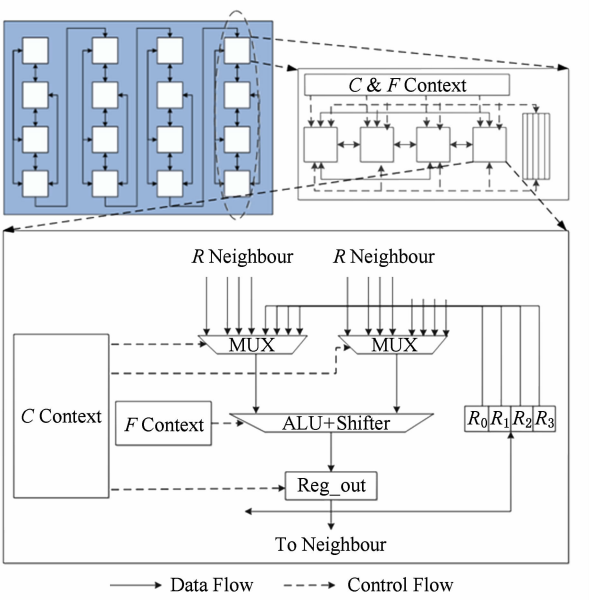


Fig. 8 Reconfigurable cipher processing SoC

图8 可重构密码处理SoC可重构部分[5]

从图8中可以看到本文将可重构配置信息存储器按层次分为3层：第1层为整个可重构协处理器部分的全局配置信息存储器，第2层为每列的列配置信息存储器，第3层为处理单元内部的局部配置信息寄存器. 同时每一层的配置信息存储器又分为了操作配置信息存储器部分和互连配置信息部分.

可重构结构可以动态改变硬件的操作和连接关系. 传统的可重构结构都是同时重构操作或者互连，配置信息量由配置字位数乘以需要配置的处理单元个数. 单元个数多导致配置信息数据量大，导致功耗高、带宽要求高. 实际上应用中大量存在单独重构操作或者单独改变互连的情况，如图9所示[5]：

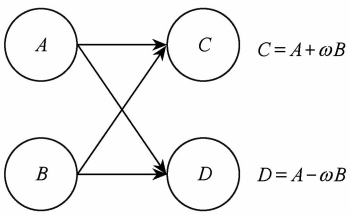


Fig. 9 Examples of operations and interconnections change independent

图9 操作与互连单独改变实例[5]

MD5包括4轮处理，每轮处理由16个迭代步组成，而这64步操作都非常相似. 每一步的运算形式为

$$A<<-B+(A+F(B,C,D)+M[k]+T[i])<<S$$

如图10所示：

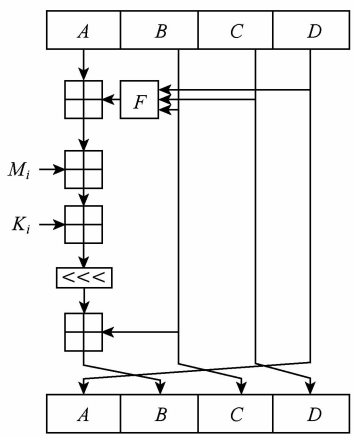


Fig. 10 Each step of the iterative calculation for MD5

图10 MD5中每一步迭代计算示意图[5]

应用定制时将该迭代计算定制为一个指令MD5round. 每个处理单元都可以在一个周期内完成一步迭代计算. MD5的64迭代计算过程构成完美的流水线，这64步的数据流向完全相同；每16步的操作也相同. 也就是说在重构的过程中每16个迭代需要重构操作，整个64个迭代都不需要改变互连. 综合时序报告显示其关键路径为4.635 ns，利用该定制功能部件最大可以运行在215 MHz.

5 配置信息生成实验结果

5.1 可重构密码处理SoC配置信息生成方法功耗分析

传统的可重构架构在可重构配置信息设计方面有3方面的考虑.

1) 传统可重构配置信息为了配合硬件设计,往往设计成 32 b 或 32 的整数倍. 对于较简单的可重构体系结构(例如可重构流水线),有可能不需要使用到如此多的位数,多余的位数则浪费. 一方面使芯片面积增大,另一方面增加重构的功耗. 而面向密码处理领域的可重构 SoC 配置信息生成方法设计了分开的操作配置信息存储器和互连配置信息存储器,不存在不使用的配置信息位数,从而减少了整个存储器容量,减少系统的静态功耗[5,12-13].

2) 传统可重构配置信息都是同时重构操作和互连,也就是一次重构的位数为处理单元乘以配置信息位数. 配置信息数据量大,同时又是存储器读操作,大大增加了系统的功耗. 而面向密码处理领域的可重构 SoC 配置信息生成方法可以单独地重构操作或互连,需要重构的次数明显减少,需要传递的配置信息总量明显减少,减少系统的动态功耗[5,12-13].

3) 传统可重构结构为了保证系统功能完备性,往往要设计相当容量的配置信息存储器,而面向密码处理领域的可重构 SoC 配置信息生成方法挖掘了重构过程中的不变性,因此可以适当地减少配置信息存储器的容量,减少了系统的静态功耗[5,14].

为了验证面向密码处理领域的可重构 SoC 可重构部分的功耗特点,本文选取了 LiverMore loop benchmark,DSPstone 和密码处理领域的关键算法[15-16].

表 1 给出了可重构密码处理 SoC(reconfigurable cryptographic processing SoC,RCPSoC)与传统可重构处理结构(traditional reconfigurable processing architecture,TRPA)不同配置信息生成方法功耗对比.

Table 1 Power Consumption Comparison Between RCPSoC and TRPA Configuration Information Generation Method

表 1 RCPSoC 与 TRPA 配置信息生成方法功耗对比[5]

Benchmark	Power Consumption/mW		Promotion/%
	TRPA	RCPSoC	
First_Diff	320.65	240.17	25.1
MUL Vector	243.88	186.08	23.7
2D-FDCT	223.90	168.15	24.9
MD5round	318.82	214.88	32.6
SHA-1round	342.85	234.85	31.5

从表 1 可以看出,由于可重构密码 SoC 配置信息存储器层次化设计和拆分为操作、互连 2 个存储器,取得了功耗上较大的改善,改善幅度从 23.7%~32.6%不等. 在 MD5 和 SHA-1 这些散列加密算法中大量的迭代计算间存在较大的相似性,从而在可重构密码 SoC 中减少了重构的次数和配置信息总量. 与其他的测试程序相比,MD5 和 SHA-1 的重构不变的是互连,在配置信息中互连配置信息占主要部分,而操作配置信息占次要部分,因此这 2 种算法的功耗提升更为明显[5].

5.2 可重构密码处理 SoC 配置速度及性能分析

在层次化配置信息存储器设计时,将配置存储器分为互连配置信息存储器和操作配置信息存储器. 在这种设计方法下,某一种模式的重构不需要更改所有的配置信息,而只需修改相应的配置信息即可,不仅配置的信息量减小了,而且重构的开销也减少了. 互连模式是多个处理单元组合的结果,互连配置信息存储器可作用于整条可重构流水线;操作模式作用于每个处理单元,操作配置信息存储器既可以作用于整条流水线也可以作用于单个处理单元. 将互连配置信息和操作配置信息分离的方法极大地减少了配置空间的大小,使用少数几个模式就可以描述大量的数据流图,在大大减少配置信息存储器读操作的同时又解决了配置空间问题,使得每次需要配置的存储器容量变小,因而性能更优,速度也更快[5].

在密码处理算法中,在一个确定的互连模式下,可以包含可重构流水线功能单元的多种操作模式,很多时候不需要更改互连模式就能完成密码算法的处理. 同时,由于流水线的作用,只需要使用第一条流水线使用的配置信息,前一条流水线运行后将配置信息和运算结果打包发送到下一条流水线中,总的配置信息层数也较少,层次化配置存储器的配置信息局部性比较好,有利于降低功耗和提高性能. 只有当局部配置寄存器中的配置信息不足时,才从处理单元外的局部配置存储器调取配置信息,大大减少了配置信息的访存频率[5].

基于上述 2 点分析,从直观上看,层次化的配置信息存储器设计方法在获得功耗减小的同时,配置的速度和性能也有很大的优势. 为了验证面向密码处理领域的可重构 SoC 配置信息生成方法的性能,与传统的可重构结构进行了比较,配置时间用时钟周期数来表示,比较结果如表 2 所示[5].

从表 2 的数据可以看出,在完成相同密码处理程序的时候,可重构密码 SoC 配置信息存储器层次化设计由于配置信息容量的减少以及较好的局部性特点,获得了 35.3%~53.5%的性能提升[5].

Table 2 Performance Comparison Between RCPSoC and TRPA Configuration Information Generation Method

表 2 RCPSoC 与 TRPA 配置信息生成方法性能比较[5]

Benchmark	TRPA		RCPSoC		Promotion/%
	Template Number	Configuration Cycle	Template Number	Configuration Cycle	
First_Diff	4	17	2	11	35.30
MUL Vector	16	32	16	15	53.20
2D-FDCT	8	16	9	9	43.75
MD5round	16	28	13	13	53.50
SHA-1round	4	23	11	11	52.20

6 结束语

本文研究了配置信息存储器的设计方法,与大部分传统可重构体系结构使用分布式的配置信息存储器进行动态重构相比,本文所采用的方法带来较大的功耗提升.本文首先设计了层次式的配置信息存储器结构;其次将配置信息分为操作配置信息和互连配置信息存储器 2 部分,即可以在不同层次上进行重构,也可以单独重构某一部分;最后,以面向密码处理领域的可重构 SoC 的配置信息生成方法为实例进行了验证.实验表明:在运行性能不变的情况下,本文的配置信息生成方法减少了 23.7%～32.6%的功耗.对于不同的应用,当应用中互连状态可以不需要重构时,由于配置信息中互连信息所占比例较大,因此带来更大的功耗改善[5].

参 考 文 献

[1] Kim Y, Park I, Choi K, et al. Power-conscious configuration cache structure and code mapping for coarse-grained reconfigurable architecture [C] //Proc of the 2006 Int Symp on Low Power Electronics and Design. New York: ACM, 2006: 310-315

[2] Ye Z A, Moshovos A, Hauck S, et al. CHIMAERA: A high-performance architecture with a tightly-coupled reconfigurable functional unit [C] //Proc of Annual Int Symp on Computer Architecture. New York: ACM, 2000: 225-235

[3] Lee J E, Choi K, Dutt N D. Compilation approach for coarse-grained reconfigurable architectures [J]. IEEE Design & Test of Computers, 2003, 20(1): 26-33

[4] Guo Y. Mapping applications to a coarse-grained reconfigurable architecture [D]. Enschede, Overijssel, Netherlands: University of Twente, 2006

[5] Shen Jianliang. Research on the design methodology of application specific coarse grained reconfigurable system on chip [D]. Changsha: National University of Defense Technology, 2014 (in Chinese)
(沈剑良. 应用定制的粗粒度可重构 SoC 设计方法研究[D]. 长沙: 国防科学技术大学, 2014)

[6] Rauwerda G K, Heysters P M, Smit G J M. Towards software defined radios using coarse-grained reconfigurable hardware [J]. IEEE Trans on Very Large Scale Integration Systems, 2008, 16(1): 3-13

[7] Venkataramani G, Najjar W, Kurdahi F, et al. A compiler framework for mapping applications to a coarse-grained reconfigurable computer architecture [C] //Proc of Int Conf on Compilers, Architecture, and Synthesis for Embedded Systems. New York: ACM, 2001: 116-125

[8] Lee J, Choi K, Dutt N D. An algorithm for mapping loops onto coarse-grained reconfigurable architectures [J]. ACM Sigplan Notices, 2003, 38(7): 183-188

[9] Dmuroulakos G, Galanis M H D, Goutis C E. Optimized back-end compiler techniques for mapping applications on coarse-grained reconfigurable matrices [J]. World Scientific and Engineering Academy and Society Trans on Computers, 2007, 6(1): 181-188

[10] Zuo Yanhui. Research on compiler for coarse grained reconfigurable array processor [D]. Changsha: National University of Defense Technology, 2008 (in Chinese)
(左艳辉. 粗粒度可重构阵列处理器编译工具研究[D]. 长沙: 国防科学技术大学, 2008)

[11] Zhu Min, Liu Leibo, Yin Shouyi. Timing parameter analysis of critical loop to reconfigurable array mapping [J]. Journal of Computer Engineering, 2012, 38(22): 260-262 (in Chinese)
(朱敏, 刘雷波, 尹首一. 关键循环到可重构阵列映射中的时序参数分析[J]. 计算机工程, 2012, 38(22): 260-262)

[12] Yang Xiaohui, Dai Zibin, Zhang Yongfu. Research and design of reconfigurable computing targeted at block CipherProcessing [J]. Journal of Computer Research and Development, 2009, 46(6): 962-967 (in Chinese)
(杨晓辉, 戴紫彬, 张永福. 可重构分组密码处理结构模型研究与设计[J]. 计算机研究与发展, 2009, 46(6): 962-967)

[13] Garcia A, Berekovic M, Aa T V. Mapping of the AES cryptographic algorithm on a coarse-grain reconfigurable array processor [C] //Proc of Int Conf on Application-Specific Systems, Architectures and Processors. Los Alamitos, CA: IEEE Computer Society, 2008: 245-250

[14] Veredas F J, Scheppler M, Moffat W, et al. Custom implementation of the coarse-grained reconfigurable ADRES architecture for multimedia purposes [C] //Proc of the 15th Int Conf on Field Programmable Logic and Applications. Piscataway, NJ: IEEE, 2005: 106-111

[15] Yoon J W, Shrivastava A, Park S, et al. A graph drawing based spatial mapping algorithm for coarse-grained reconfigurable architectures [J]. IEEE Trans on Very Large Scale Integration Systems, 2009, 17(11): 1565-1578

[16] Lee G, Lee S, Choi K, et al. Routing-aware application mapping considering steiner points for coarse-grained reconfigurable architecture [C] //Proc of Int Symp on Applied Reconfigurable Computing. Berlin: Springer, 2010: 231-243

Shen Jianliang, born in 1982. PhD and lecturer. Member of CCF. His main research interests include reconfigurable computing and embedded SoC design.

Li Sikun, born in 1941. Professor and PhD supervisor. Senior member of CCF. His main research interests include VLSI design methodology and reconfigurable computing (sikunli@126.com).

Liu Lei, born in 1984. PhD candidate. His main research interest is reconfigurable computing (liulei@nudt.edu.cn).

Wang Guanwu, born in 1986. PhD candidate. His main research interest is reconfigurable computing (guanwuwang1986@126.com).

Wang Xin, born in 1986. Master. His main research interests include reconfigurable computing and SoC design (wx@ndsc.com.cn).

Liu Qinrang, born in 1975. PhD and researcher. His main research interests include reconfigurable computing and network architecture (lqr@ndsc.com.cn).