doi:10.3772/j.issn.1002-0470.2015.01.007

数据触发的基本块间弹性控制电路综合方法①

黄元杰②*,**,*** 陈云霁*,** 吴承勇*,**

(*中国科学院计算机体系结构国家重点实验室 北京 100190)

(**中国科学院计算技术研究所 北京 100190)

(***中国科学院大学 北京 100049)

摘 要 研究了面向弹性粗粒度可重构阵列(CGRA)的高级语言综合方法,针对现有方法由于采用逐个执行基本块的方式限制了循环代码性能的问题,提出了一种在内层循环基本块间局部地采用数据触发的新型控制方式。这一新的综合方式在保证正确性的同时可缩短循环迭代间隔。实验表明,该方法平均只需引入25.4%的面积开销即可缩减50%的执行时间,并在5/6的测试程序上节约了执行能耗。

关键词 可重构处理器,弹性电路,动态调度,数据流控制

0 引 言

粗粒度可重构阵列(coarse-grained reconfigurable array, CGRA)[1]是一种新兴的加速器件,它类似现场可编程逻辑门阵列(field-programmable gate array, FPGA),可提供丰富的计算资源,另外与FPGA相比,基于字的粒度使其在实现原软件算法时需要的面积更少、配置加载时间更短、能效更高。尽管有这样的潜力,CGRA却因为编程方法上的问题尚未得到广泛的应用。传统的CGRA使用类似超长指令字(very-long instruction word, VLIW)处理器的静态调度和编程方式,因而工具复杂,难以处理较大规模代码,而且难以和不确定延迟的内存系统交互。针对这一问题,Huang等人[2]提出了引入弹性电路实现动态调度的CGRA结构和综合方法,然而其采用的逐个基本块执行的基本块间控制流综合方式会限制CGRA性能的发挥。本文提出了一种改进的从高级语言代码生成弹性CGRA配置的控制流综合方法。这一方法采用数据触发的方式实现循环的内层基本块间的控制,而在其他基本块间沿用基本块出口处同步并逐个基本块执行的控制方式。这一新的控制流综合方法,在保证生成控制电路正确性的同时,可显著缩短对性能最为关键的内层循环的迭代间隔。我们采用TSMC 65nm标准单元库实现基于弹性电路控制的CGRA,并为其开发了支持不同基本块间控制流综合方法的工具链,而且对比了前人方法和本文方法所综合成的电路的执行时间、面积和执行能耗。

1 弹性控制及其综合方法

弹性控制是一种数据流式的控制:数据和控制信号成对在电路中传播,运算和存储在获得数据有效的控制信号才会进行,而操作后有效信号会继续传递给后继。同时后继的停止信号可以阻塞前驱。这种握手传递的方式使得电路生成时可以含有不定时长的操作,如访问外部存储器。数据流式控制并不新颖,但以往粗粒度加速器的研究中数据流控制多以异步电路实现。异步电路流水线以Muller C

① 国家自然科学基金(61003064,61221062,61303158)和中科院先导专项(XDA06010403)资助项目。

② 男,1985年生,博士生;研究方向:计算机体系结构;联系人,E-mail: hungyuanjie@ict.ac.cn

(收稿日期:2014-04-14)

门[3]为核心实现流水级间的控制。这种门电路的行为和 SR 锁存器类似,当且仅当前驱输出为高,后继反馈为低时,Muller C 门才会锁存并输出高电平。异步电路的流水线控制需要对控制信号进行如图 1 所示的计算延迟匹配,若将其用于可重构阵列,匹配延迟的要求会大大增加布线复杂度。

Lewis 等人[4]总结并提出了在同步电路中实现握手控制的方式,其后 Cotadella 等人[5]提出了使用称为弹性电路同步互锁结构并将其用于综合生成数据流控制器。弹性电路中流水级由弹性缓冲(elastic buffer, EB)组成。EB 可基于锁存器实现亦可基于寄存器实现。

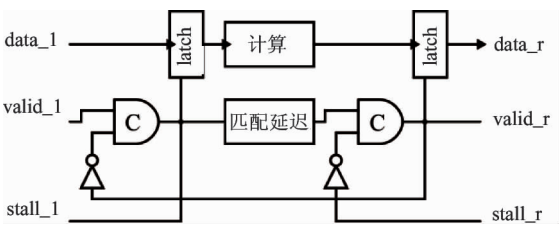


图 1　异步电路流水线

将高级语言综合到弹性电路需要将数据流和控制流转换为算术逻辑和弹性电路元素构成的电路。弹性电路综合步骤分为基本块内数据流综合和基本块间控制流综合两部分。基本块在编译中指拥有单个入口和单个出口的代码片段,每个基本块可以有多个前驱,但入口和出口都是唯一的,基本块出口处可以有最多两个后继。

1.1　基本块内的数据流控制和所用弹性元素

基本块内的数据流控制所用弹性元素包括弹性缓冲(EB)元素、JO(join)元素和 EF(eager fork)元素,其结构如图 2 所示。

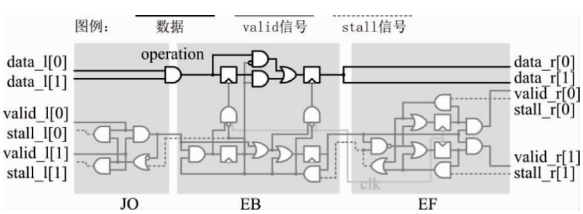


图 2　JO、EB 和 EF 弹性元素结构

每个基本块内的数据流都可以表达为一个无环的依赖图,该图以基本块入口处的活跃变量为头,以基本块出口处的活跃变量为尾。基本块的综合即为将这个依赖图表达为弹性电路的过程。

EB 元素在综合中用于存储基本块出口处的活跃变量,供后继基本块使用。为了优化电路时钟周期,EB 亦被用于算术操作之间,切分关键路径。弹性电路中控制和数据部分总是匹配地出现,如图 2 中的 EB 元素,就由上半部分的数据寄存器和下半部分的弹性控制组成,所有弹性元素都使用具有同样语义的有效(valid)和停止(stall)信号对。

JO 元素[4]用于从多个前驱生成一个有效信号,表示所有前驱的输出都可用。在控制由基本块内操作组成的电路时,JO 被用于多元运算,连接提供数据的元素和计算后存储数据的 EB。如图 2 中的 JO 即用于保证 EB 输入的数据由两个有效的数据计算而来。

EF 元素[4]用于确认所有后继都已成功接收数据。当基本块内的一个操作的结果需要被多个操作使用时,便采用 EF 元素确保操作的数值在全部后继都成功获得之前不会被改写。如图 2 中的 EF 可确保 EB 在两个后继出现不能接收数据时不会更新输出数据。

除上述三种弹性元素外,综合基本块还需要在粗粒度可重构阵列(CGRA)中设置弹性访存接口[2]来一一映射代码中的访存操作。这些接口和多 bank 的内存或缓存连接,实现 CGRA 对系统主内存的访问。当源码的访存间有依赖关系时,则通过生成额外的弹性控制连接确保顺序。这一设计使得弹性访存接口和内存或缓存之间无需完整的读写序列去保证访存操作返回的顺序性,从而大大降低了其复杂性。同时因为现代编译技术仍无法完全分析所有的指针依赖,所以这类情况我们暂时采取用手工在源码中标注的方式为工具提供辅助信息。

1.2　基本块间的控制流综合所用的弹性元素

如前小节所述,每个基本块电路的输入都是其活跃变量的值和相应的弹性信号,输出都是出口处的活跃变量值和每个值的弹性信号。基本块之间没有运算,但是需要根据程序运行中计算的条件选择将数据在基本块对应的电路之间传递并激活基本块转换成的弹性电路。控制流的表达需要 BR

(branch)元素和 MG(merge)两个弹性元素,它们的结构如图 3 所示。

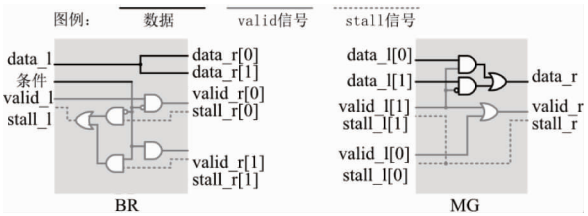


图 3　BR 和 MG 元素结构

BR 元素[4]根据输入的数据将弹性控制信号对与其两个后继中的一个连接,完成根据动态的条件选择数据或纯弹性控制使能信号目的地的功能。

MG 元素[2]可从多个前驱获得数据有效信号并将其传递给后继。与同样拥有多个前驱的 JO 元素不同,MG 元素没有同步功能,不会阻塞前驱,它采用先到先传的方式处理多个输入。因此 MG 元素可以表达基本块入口处的活跃变量具有多个来源的语义。

需要特别注意的是 MG 和其他弹性元素不同,其输出受到其来自前驱的数据到达顺序影响。而在弹性电路中到达顺序可能因为不确定的延迟而改变,所以为确保执行的正确性,需要在综合控制流时加以限制,使得 MG 元素的两个前驱严格互斥。

本节之后将采用图 4 中的示例代码对比不同的综合基本块间控制流的方法。

```
intcond; inti=j=w=0;
do{
do{
cond=i<=20;i++; *out=w;
p=in+i;
w=*p;
}while(cond);
w=j;
}while(j++<5);
```

图 4　控制流综合示例代码

1.3　逐基本块执行的控制方式

Huang 等人在综合控制流时,采用了逐基本块的执行方式,以一个基本块级弹性有效信号表达基本块的入口或出口处的全部活跃变量都有效[2]。多前驱基本块入口处的每个活跃变量会用一个 MG 元素和其控制的多路选择器(Multiplexer, MUX)表达。这些 MUX 以相应前驱基本块的活跃变量的数据输出为数据前驱。同时入口处会为每个前驱基本块生成一个 EF 元素,该 EF 将弹性信号分发至每个活跃变量对应的 MG 元素,作为其弹性控制前驱。而在基本块电路的出口处,会生成一个 JO 元素,接收并同步所有活跃变量的弹性控制信号对,生成该基本块的出口的基本块级弹性有效信号。如果该基本块以分支结尾,则这一信号对经过 BR 元素传递给两个后继。这一基本块间控制方式如图 5 示意。

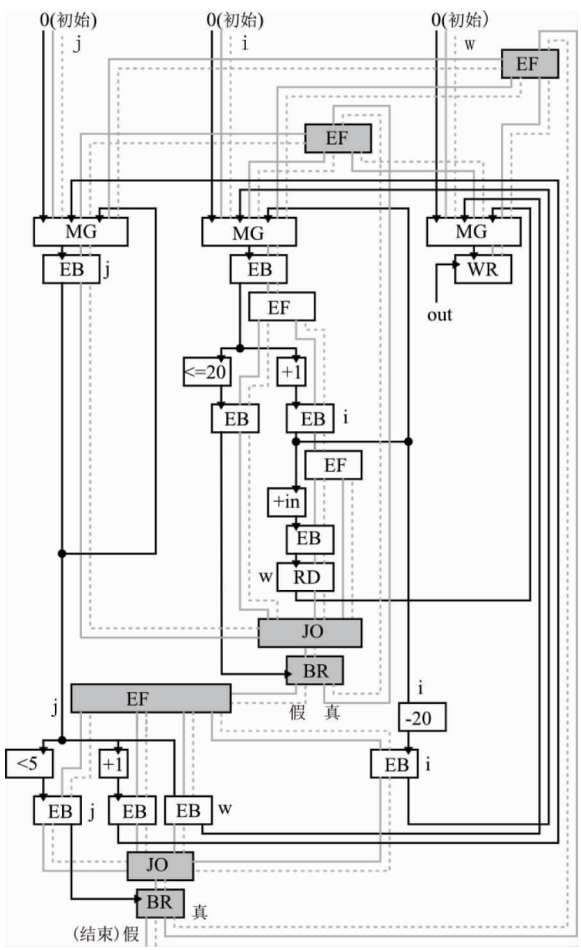


图 5　基本块级同步的控制方式

因为基本块级的同步,所以任意时刻都只可能有一个基本块属于完成并输出基本块级有效信号的状态,这种方法可以确保 MG 元素弹性控制输入间不会存在竞争。这种方式的显著缺点是在执行循环时,只能等待一次迭代的全部操作结束才可以发起下一次迭代。而在高级语言编写的代码中,来自循环不同迭代的操作往往可以同时执行。如例子中的

代码,如果形成流水线,则来自不同迭代的 $i <= 20$; $i++$; $p = in + i$; $w = *p$; $*out = w$; 可以同时执行,加上 MG 后的 EB,迭代间隔仍可短至 2 周期,而非图中的 4 周期(假设读内存需要 1 周期)。

1.4 基本块间使用数据驱动的控制方式

相较基本块间同步,一种更直接的控制是在基本块之间使用各个活跃变量的有效信号对分别触发后继基本块中依赖这一数据的计算。实现这种方式只需为基本块出口处的每个活跃变量生成一个 BR 元素,使之根据分支条件直接将这一变量的数据和弹性控制信号组合送至后继基本块入口的对应该变量 MG 元素。这种方式生成的电路如图 6 所示。

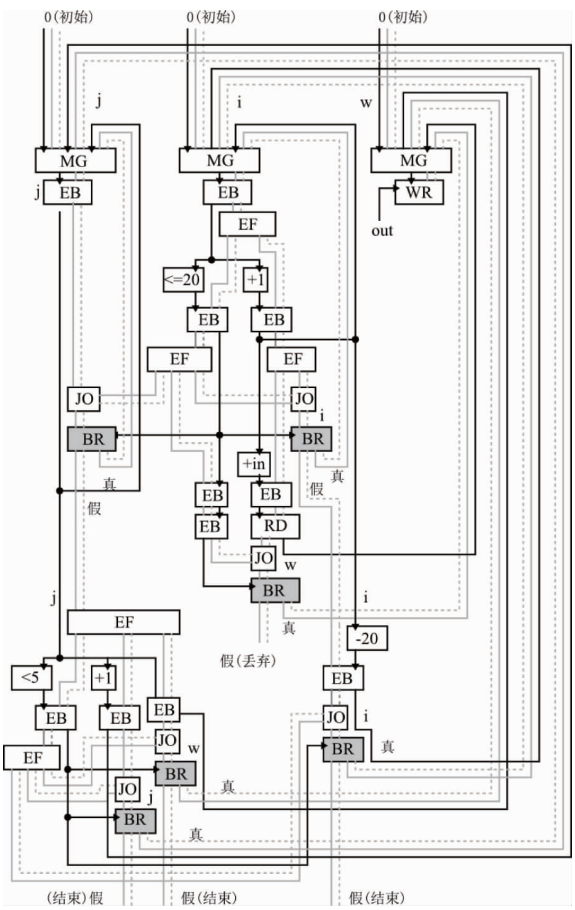


图 6 无基本块级同步的纯数据流控制方式

这种方式虽然直观,但无法保证程序的正确性。假如在循环结束时可能前一迭代的活跃变量仍未计算完成,则可能在 MG 处出现输入竞争。假设图 6 中的弹性电路可能出现内层循环中读操作可能需要多个周期,则倒数第二次迭代的 w 值可能落后于最后一次迭代的 j 和 i 的值就绪。此时通过外层循环变量 j 会生成一个新的 w 值,与内层循环产生的 w 值构成 MG 入口处的竞争。

为了解决同步方式的性能问题和无同步数据流方式的正确性问题,我们提出在对性能最为关键的循环最内层基本块间的连接采用无同步的数据触发,而其它基本块间仍同步所有活跃变量的混合控制方式。这样在退出内层循环时需要经过基本块级别的同步,确保基本块级的有效信号离开循环时全部迭代都已完成,因此不会在 MG 元素上出现输入竞争。上述例子使用本文提出的这种方式综合得到的电路如图 7 所示。

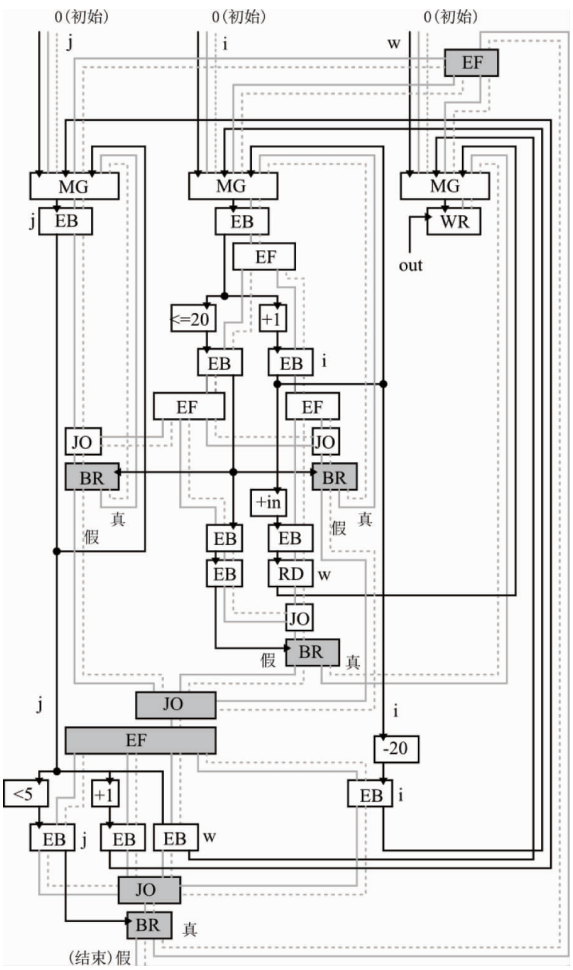


图 7 本文提出的内层循环局部数据驱动的控制方式

可以看到,图 7 所示的电路每次只需两周期即可计算 i 和 j 的值,之后就可以开启下一次迭代,因此可达到 2 周期的循环间隔,性能相比图 6 中电路提高了 2 倍。

2 弹性 CGRA 结构和工具

为验证本文提出的综合方法,我们实现了一种如图 8 所示的 CGRA 结构。这一结构主要由同构的拼接块组成,拼接块是完全同构的矩形,相邻两块的连线端点完全对应,可以无缝地拼接在一起。在拼接块组成的阵列的一侧,分布着弹性虚拟内存接口。CGRA 结构中灰色背景标示出的即为一个拼接块。每个拼接快包含一个算术逻辑单元(arithmetic logic unit,ALU)可重构单元、一个多路选择器(MUX)可重构单元和连接这些单元与周边的可重构连接网络。

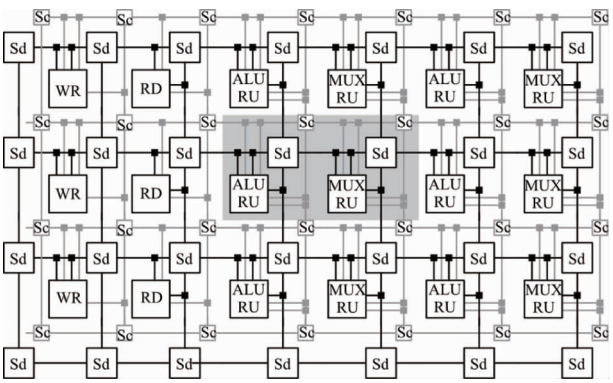


图 8 CGRA 结构图

2.1 弹性可重构单元

弹性可重构单元是弹性 CGRA 的基本功能元素,每个单元都包括一个弹性控制元素的集合,用于实现对本单元中数据计算部分的控制功能。我们根据 Huang 等人的论文[2]实现了其提出的由 ALU、MUX 和 Sync 三种可重构单元组成的阵列结构。我们发现在映射本文方法优化后的测试程序弹性电路时,其结构中的 MUX 单元和 Sync 单元在启用的拼接块中的平均利用率分别只有10.6%和5.2%,而为连接其 Sync 单元所需的可重构连线和配置存储就占到综合后拼接块面积的 11%。因此我们提出将可重构单元减为两种,其中 ALU 单元主要用于执行计算,而 MUX 单元则处理无需 ALU 的多种功能。

ALU 单元:我们的 CGRA 中的 ALU 可以执行以下操作:加、减、乘、左右移位、比较。因为本文中的 CGRA 基于 32 位字长,所以在计算访存地址时常用到逻辑左移 2 的操作,因此 ALU 具有计算前将操作数左移 2 位的功能。ALU 单元中只设置存储数据或有效信号的 EB 和常伴随算术操作的 JO、EF 控制元素。

MUX 单元:这种单元含有一个由弹性元素控制的数据 MUX 和全部的 5 种弹性控制元素:EB、MG、JO、BR 和 EF。

以上两种单元的结构如图 9 所示。本文提出的 MUX 单元在各测试程序所启用的拼接块中利用率平均达到了 95%。同时在相同时序约束条件下,综合后拼接块面积也由 59319μm^2 缩减到了 49209μm^2。

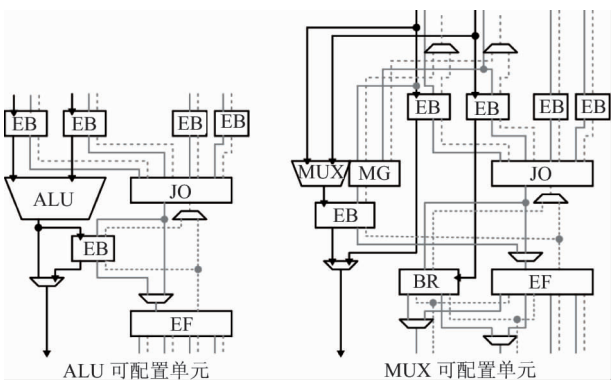


图 9 ALU 单元和 MUX 单元

2.2 连接网络

本文中的弹性 CGRA 的连接网络由 32 位宽和 1 位宽的两个可重构连接网络组成,分别用于数据连接和控制信号连接。网络结构与 FPGA 类似:S-block 纵横连接彼此,连线上的 C-block 将可重构单元连入网络。因为数据网络中数据以 32 位字整体传输,所以其 S-block 和 C-block 的每个 32 位通道的全部连线可以共用配置存储。在图 8 中,Sd 和 Sc 分别表示数据网络和控制网络中的 S-block,而黑色和灰色的方点则表示相应网络中的连接盒 C-block。

2.3 工具链

本研究中的弹性 CGRA 工具链以 GCC 编译器为前端,我们实现的弹性 CGRA 综合器为后端。前端输出代码的中间级表示,后端首先将中间级表示综合成弹性元素和算术逻辑操作组成的弹性电路,然后再打包成 CGRA 的可重构单元网表。之后通过经多次调用开源 FPGA 布局布线工具 VPR 将可

重构单元网表映射到 CGRA 上。

3 实验评估

我们将第 1.3 节介绍的逐基本块执行的控制方式[2]作为基准点,和本文提出的方式分别综合同样的测试程序代码到同样的 CGRA 结构上,来验证我们方法的效果。实验中的 CGRA 以 TSMC 65nm LP 标准单元工艺库在 Synopsys Design Ultra 2009, IC Compiler 2009 中进行设计、综合和布局布线。CGRA 的能耗计算工具为 Synopsys PrimeTime 2009,估计功耗时采用了布局布线后的仿真波形。我们计算的能耗不包含 CGRA 上未使用的拼接块,因为它们的供电可以关闭。

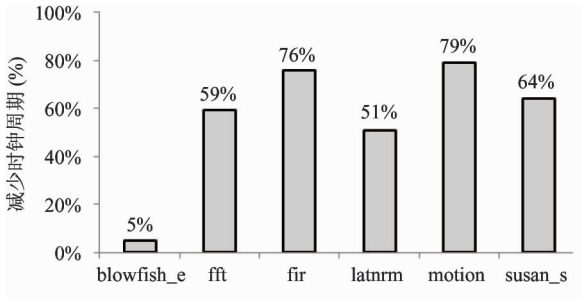
仿真中的访存行为由测试驱动中的 Verilog HDL 代码模拟,因为访存行为并非本文关注的重点,仿真中我们假设访存可以在 CGRA 的 cache 中进行,读写操作均只需要一个周期即可完成。

我们选取了共计 6 个测试程序,这些程序分别来自自动化、安全、通信和图像领域。通过 gprof 我们选取了其中的热循环作为面向 CGRA 综合的测试代码。

3.1 代码执行时间

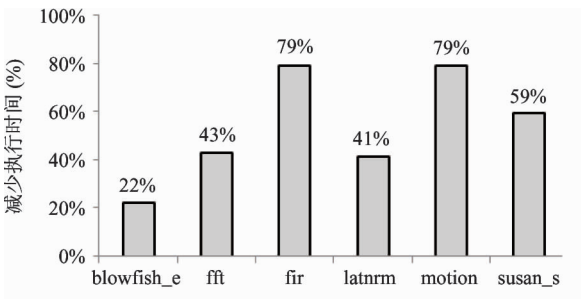
我们的局部数据触发的控制方式允许来内层循环的不同迭代的操作在时间上重叠,辅以管道平衡优化理论上可最大程度地发挥内层循环的指令级并行度。如图 10 所示,该方法确实在多数测试程序上显著降低了代码执行的周期数据。其中 blowfish _ e 只取得了很小的性能提升,这是因为其代码固有的数据间依赖限制了不同迭代同时执行的可能性。

因为 CGRA 在执行一个配置(程序)时的最高频率受布局布线影响,因此还需要考察总的实行时间。如图 11 所示,程序的执行时间同样减少显著。其中 blowfish _ e 和 fir 更是取得了较时钟周期数减少更大幅度的执行时间减少。这是因为使用我们的内层循环控制方式,在循环体基本块到自身的迭代时,可避免基本块入口和出口的较多元素参与的 EF 和 JO,因而使得 CGRA 上的布线不易拥塞,最长路径更短。



	blowfish_e	fft	fir	latnrm	motion	susan_s
优化前周期数	1606	21113	278725	26375	5908	124530
优化后周期数	1531	8562	66565	12999	1243	44610

图 10 减少的时钟周期百分比

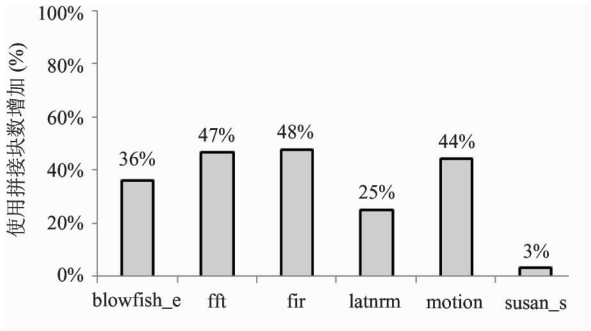


	blowfish_e	fft	fir	latnrm	motion	susan_s
优化前时间(s)	9.93E-06	9.00E-05	1.17E-03	1.10E-04	2.35E-05	7.31E-04
优化后时间(s)	7.75E-06	5.12E-05	2.44E-04	6.48E-05	4.95E-06	2.98E-04

图 11 减少的代码执行时间百分比

3.2 拼接块数目(面积)

图 12 显示了使用本文中控制方式所生成的电路在 CGRA 上利用到的拼接块数目的变化。因为 CGRA 上每个拼接块的大小是固定的,因此这一数字反映了电路所占的芯片面积。可以看到我们方法增加的面积开销较低。这里需要指出的是,因为我们对电路进行了流水线平衡,会仅为使用 EB 元素而启用较多的可重构单元。因此存在通过改进 CGRA 结构,增加单元中 EB 数目来优化面积开销的机会,只是这一优化偏离了本文主旨,尚未实现。susan _ s 的面积增量少于其他测试程序是因为其内层循环的循环间活跃变量少,且循环体更小。

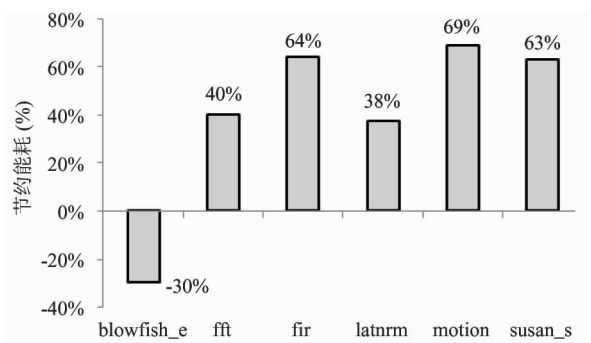


	blowfish_e	fft	fir	latnrm	motion	susan_s
优化前拼接块数	97	77	23	36	45	128
优化后拼接块数	132	113	34	45	65	132

图 12 增加的拼接块数目百分比

3.3 能耗

图 13 总结了我们方法在能耗上带来的影响,可以看到多数程序都获得了较大的能耗削减。这一削减主要来自于更少的执行周期数目带来的时钟网络和寄存器上的能耗降低。唯一产生更高能耗的 blowfish _ e 的执行周期减少的较少而面积增加的较多。后续的研究中我们计划以代码静态分析判断内层循环的优化可能再进行变换,避免这样的开销。



	blowfish_e	fft	fir	latnrm	motion	susan_s
优化前能耗(J)	2.66E-06	2.77E-05	1.09E-04	1.64E-05	4.54E-06	2.71E-04
优化后能耗(J)	3.44E-06	1.65E-05	3.90E-05	1.02E-05	1.40E-06	1.00E-04

图 13 节约的能耗百分比

4 粗粒度可重构处理器的相关研究

除前文提到的 Huang 等人[2]设计的弹性控制的 CGRA 之外,早期的 CGRA 如 ADRES[6,7]、MATRIX[8]、Tartan[9]、MorphoSys[10]均相似地采用了复杂的编译技术来生成静态调度。这些粗粒度处理器为了缩短迭代间隔,需要计算复杂的静态 modulo 调度[11,12]。本文采用的方法使得 CGRA 在具备适应内存延迟的动态调度能力的基础上取得了与这些静态调度方法类似的短迭代间隔。

周学海[13]等人提出了一种流式处理的编程模型,而许牧[14]等人提出了一种数据驱动的可重构多核处理器,这些工作均未涉及从高级语言代码向数据流控制结构综合的方法并深入讨论控制流的实现方式和效果,与本文探讨的问题不同。

窦勇[15]、王大伟[16]和徐进辉[17]等人在其设计的一种 CGRA 上实现了循环的自流水控制,然而其针对的代码仍旧具有规范性,依靠对 FOR 循环等模板进行映射的方式进行控制。这一方式相比我们的可正确综合任意弹性电路的方式缺乏灵活性。

5 结 论

本文提出了新型的综合高级语言代码控制流到弹性电路的方式,创新性地提出了通过在内层循环基本块到自身的连接上使用数据触发,而在其他基本块间连接进行基本块级同步的方式。这一方式在保证正确性前提下可以较低的面积开销显著提高弹性 CGRA 执行循环代码的效率,缩短程序运行时间,并随之降低执行程序所需能耗。

参考文献

[1] De Sutter B, Raghavan P, Lambrechts A. Coarse-grained reconfigurable array architectures. In: Handbook of Signal Processing Systems. USA: Springer US, 2010. 449-484

[2] Huang Y, Ienne P, Temam O, et al. Elastic CGRAs. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, USA, 2013. 171-180

[3] Muller E, Bartky S. A Theory of Asynchronous Circuits

I:[Technical Report] University of Illinois, Graduate College, Digital Computer Laboratory, 1957. 1-32

[4] Lewis M, Brackenbury L. Synchronous handshake circuits. In:Proceedings of the 7th International Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, USA, 2001. 86-95

[5] Cortadella J, Kishinevsky M, Grundmann B. Synthesis of synchronous elastic architectures. In: Proceedings of the 43rd Annual Conference on Design Automation, San Francisco, USA, 2006. 657-662

[6] Mei B, Vernalde S, Verkest D, et al. DRESC: a retargetable compiler for coarse-grained reconfigurable architectures. In:Proceedings of 2002 IEEE International Conference on Field-Programmable Technology, Hongkong, China, 2002. 166-173

[7] Mei B, Vernalde S, Verkest D, et al. ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix. In: Proceedings of 13th International Conference on Field Programmable Logic and Application, Lisbon, Portugal, 2003. 61-70

[8] Mirsky E, DeHon A. MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources. In: Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, Monterey, USA,1996. 157-166

[9] Mishra M, Timothy J, Callahan C, et al. Tartan: evaluating spatial computation for whole program execution. In: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, USA, 2006. 163-174

[10] Singh H, Kurdahi F, Bagherzadeh N, et al. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications. *IEEE Transactions on Computers*, 2002. 49(5): 465-481

[11] Park H, Fan K, Kudlur M, et al. Modulo graph embedding: mapping applications onto coarse-grained reconfigurable architectures. In: Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, New York, USA, 2006. 136-146

[12] Friedman S, Carroll A. SPR: an architecture-adaptive CGRA mapping tool. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, USA, 2009. 191-200

[13] 周学海, 罗赛, 王峰等. 一种数据驱动的可重构计算统一编程模型. 电子学报, 2007, 35(11): 2123-2128

[14] 许牧, 安虹, 汤旭龙等. 一种类数据流驱动的可重构众核流处理器设计. 小型微型计算机系统, 2013, 34(06): 1359-1364

[15] 窦勇, 邬贵明, 徐进辉等. 支持循环自动流水线的粗粒度可重构阵列体系结构. 中国科学 E 辑:信息科学, 2008, 38(4): 579-591

[16] 王大伟, 窦勇, 李思昆. 核心循环到粗粒度可重构体系结构的流水化映射. 计算机学报, 2009,32(06): 1089-1099

[17] 徐进辉, 杨梦梦, 窦勇等. 粗粒度可重构平台中循环自流水硬件实现. 计算机学报, 2009, 32(06): 1080-1088

A method for elastic controller synthesis using data-triggered execution across basic blocks

Huang Yuanjie[*,**,***], Chen Yunji[*,**], Wu Chengyong[*,**]

([*]State Key Laboratory of Computer Architecture, Chinese Academy of Sciences, Beijing 100190)

([**]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

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

Abstract

The controller synthesis for an elastic coarse-grained reconfigurable array (CGRA) was studied, and an existing synthesis method's performance limitation introduced by the basic-block-by-basic-block execution was addressed. Then, a novel elastic controller synthesis method using data-triggered execution within the inner-most-loop was proposed. This new method can shorten the iteration interval of inner-loops without breaking correctness constraints. The experimental results show that with a low extra-area overhead of 25.4%, the execution time can be reduced by 50% on average, and the execution energy can be saved on five over six of benchmark programs.

Key words: reconfigurable processor, elastic circuit, dynamic controller, data-triggered execution