一种基于控制流解耦的可重构阵列动态调度方法

尹 琛[1],彭 飞[2],景乃锋[1]
(1.上海交通大学 电子信息与电气工程学院,上海 200240;2.上海航天电子技术研究所,上海 201109)

摘 要: 可重构阵列依靠数据流驱动带来的能效优势,被广泛运用在特定领域的运算加速中。随着应用范围的增大,当应用中存在不同控制流区域时,采用传统的空间调度方案同时执行整个数据流图,会由于非一致性控制流的存在,造成严重的性能损失。本文提出一种基于控制流解耦的调度方法,通过将处于不同控制边界的数据流解耦成若干个相互独立的子图交替执行,同时将每个子图进行充分并行展开以提高阵列的计算资源利用率。实验结果表明:在相同面积开销的约束下,利用本文提出的调度方法分别在执行性能和执行能效上,相比于一种典型的静态调度可重构阵列(Plasticine),分别提高了 35% 和 18%;相比于一种典型的指令调度的可重构阵列(TIA),分别提高了 27% 和 45%。

关键词: 可重构阵列; 控制流解耦; 子图调度; 阵列利用率; 高性能计算
中图分类号: TP 303 **文献标志码:** A **DOI:** 10.19328/j.cnki.2096-8655.2021.04.010

A Dynamic Scheduling Method for Reconfigurable Arrays Based on Control Flow Decoupling

YIN Chen[1], PENG Fei[2], JING Naifeng[1]
(1.School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; 2.Shanghai Aerospace Electronic Technology Institute, Shanghai 201109, China)

Abstract: Reconfigurable arrays have been widely used in computing acceleration in specific fields owing to their energy efficiency advantages brought by data stream driving. With the increase in the application range, when there are different control flow areas in the application, using the traditional spatial scheduling scheme to execute the entire data flow graph at the same time will cause serious performance loss due to the existence of inconsistent control flow. In this paper, a scheduling method is proposed based on control flow decoupling, in which the data flows at different control boundaries are decoupled into several independent subgraphs. Subgraphs are executed alternately with fully unrolling to increase the utilization of computing resource. The experimental results show that under the same computing resource constraints, the proposed scheduling method can improve the performance and energy efficiency by 35% and 18% over a static-mapped coarse-grained reconfigurable array (CGRA) (Plasticine) and by 27% and 45% over an instruction-driven CGRA (TIA).

Key words: reconfigurable array; control flow decoupling; subgraph scheduling; array utilization; high-performance computing

0 引言

粗粒度可重构阵列(Coarse-Grained Reconfigurable Array, CGRA)由于兼具通用计算的灵活性和专用计算的高效性,被广泛运用在特定领

收稿日期: 2021-03-16; **修回日期:** 2021-04-23
基金项目: 上海航天技术研究院-上海交大航天先进技术联合研究基金
作者简介: 尹 琛(1995—),男,博士研究生,主要研究方向为计算机体系结构与可重构计算。
通信作者: 景乃锋(1982—),男,博士,副研究员,主要研究方向为计算机体系结构。

域中。整个阵列由大量执行单元(Processing Element, PE)以数据流驱动的模式执行操作。PE之间通过软流水的执行模式实现操作，一旦输入数据有效，PE便可以并行执行，这种计算方式极大地提高了阵列的计算吞吐率。

随着可重构阵列的应用范围扩大，当采用传统的直接调度方案，将整个数据流图(Data Flow Graph, DFG)同时映射时，会因控制流的非一致性导致处于不同控制体的数据流无法同时并行执行，进而引起阵列计算单元利用率下降的问题。常见的非一致性控制流有非完美循环、分支以及循环依赖等。针对非完美循环，可以通过循环交换和循环展开，对内层循环体和外层循环体进行重新组织，但并没有解决执行外层循环PE利用率低的问题[1-4]；针对分支，通过将不同分支上的指令合并映射到同一PE上以提高PE利用率，但这种方法不适用于路径长度差异较大的非平衡分支[5-6]；针对循环依赖，通过利用线程间通信挖掘细粒度的数据并行，但是会引入额外的硬件开销[7-8]。此外，现有的数据流调度方法都只针对其中某一种非一致性控制流进行调度优化，无法同时支持上述3种情形。

针对上述问题，本文提出了一种基于控制流解耦的可重构阵列映射方法。如图1所示，通过将原始DFG中的数据流按照不同的控制流边界解耦成若干个相互独立的数据流子图(Sub-DFG)，各个解耦后的子图可以独立执行，因此，可以通过循环完全展开的方式尽可能地利用计算资源。同时通过交替执行不同的子图，以完成整个应用的操作。借助这种控制流解耦的调度方法，在阵列总面积开销相同的条件下，相比于静态调度的CGRA[9]和完全指令驱动的CGRA[10]，本方法可以分别提高35%和18%的执行性能，同时提升27%和45%的计算能效比。

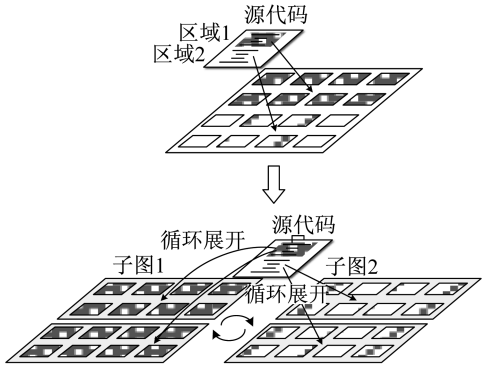

图1　控制流解耦调度方法的基本示意图

Fig.1　Basic schematic diagram of the control flow decoupling and scheduling method

1　非一致性控制流分析

本章如图2所示，将对3种常见的非一致性控制流的执行特点进行分析，并阐述这种非一致性在软流水执行模式下对阵列性能的影响。

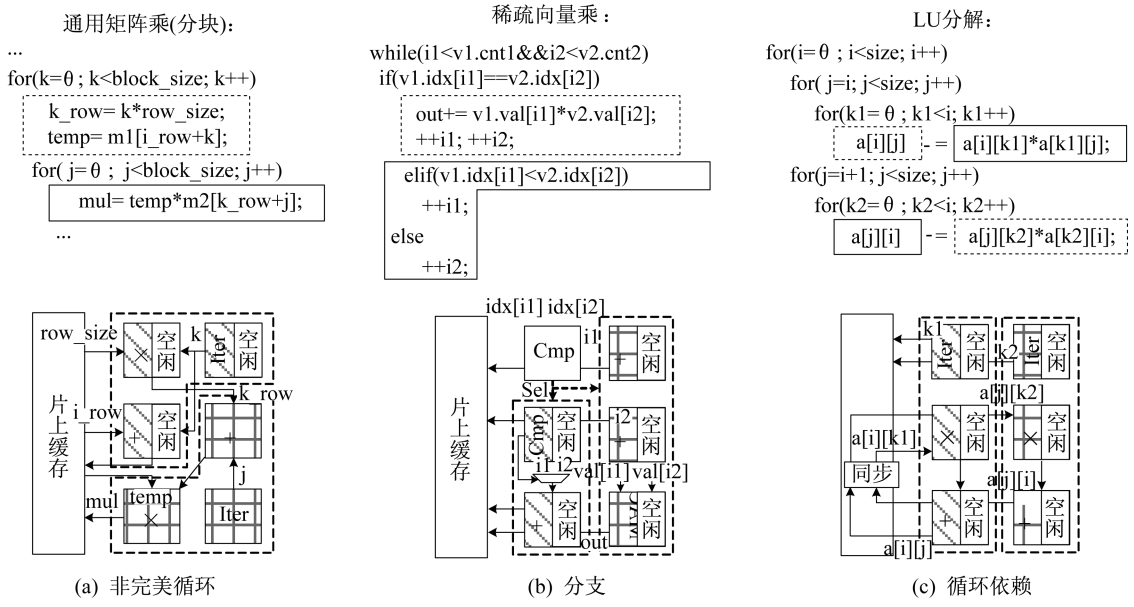

图2　3种典型的非一致性控制流

Fig.2　Three typical inconsistent control flows

图2(a)以通用矩阵乘该应用为例,对非完美循环进行阐述。图2(a)上方代码,内层循环体(实线框所示)每执行block_size次,外层循环体(虚线框所示)才执行一次;由于非完美循环中内外层循环的触发频率不同,导致执行外层循环的PE(斜线阴影所示)大部分时间处于空闲状态,因此,会降低PE利用率。

图2(b)以稀疏向量乘该应用为例,对分支执行进行阐述。图2(b)上方代码,if分支和else分支会根据分支的判断条件选择性执行,不会同时执行。图2(b)下图,执行if分支的PE(斜线阴影所示)和执行else分支的PE(格子阴影所示)在任意时刻都必定有一方处于空闲状态。因此,同样会降低PE利用率。

图2(c)以LU分解为例,对循环依赖进行阐述。图2(c)上方代码,具有数据依赖的执行体(虚线框和实线框所示)无法同时并行执行。图2(c)下图,执行这2块区域的PE(斜线阴影和格子阴影)会交替处于空闲状态,进而会降低PE的利用率。

以上3种非一致性控制流都会将原始的DFG分割成具有不同控制语义的区域。在软流水的执行模式下,这些区域内的数据流无法并行执行,因此,会显著降低阵列中计算单元的利用效率,进而影响阵列的执行性能。

2 基于控制流解耦的调度方案及硬件架构设计

针对以上问题,本文提出了一种基于控制流解耦的调度方案。在软件调度上,将原始的DFG按照不同的控制流边界分割成若干个相互独立的子图,子图间可以并行展开执行。在硬件结构上,本文提出了一种动态子图调度器,可以根据数据流的执行情况,切换阵列当前所执行的子图配置。同时,本文在传统的PE内部增加了一个配置切换单元,以支持软流水执行模式下子图配置的动态切换。

2.1 基于子图分割的控制流解耦

调度方法的整体示意图如图1所示,相比于现有的调度方法,将程序中所有的控制流区域同时映射在整个阵列上,将不同的控制流区域解耦成若干个子图,并将每个子图经过充分循环展开后独立映射在整个阵列上。通过交替执行所有的子图,以实现子图间的解耦调度,消除子图间由于非一致性控制流造成的性能损失。如图3所示,将对图1中的3个例子分别进行详细阐述。

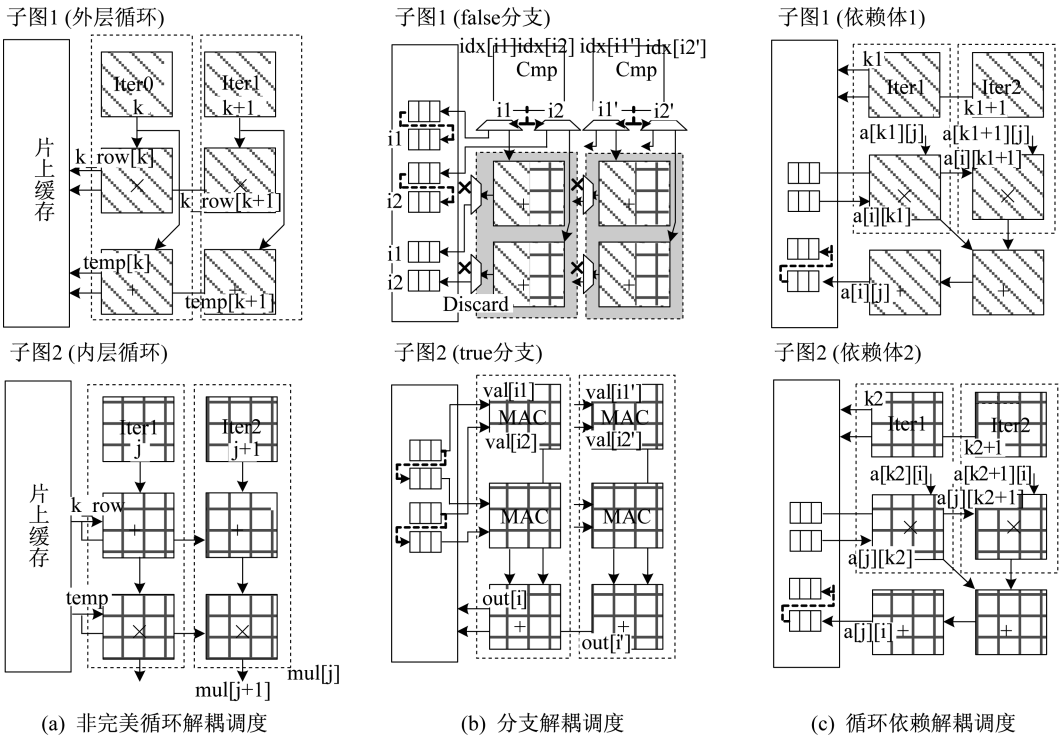

图3 基于控制流解耦的调度方法

Fig.3 Mapping with control flow decoupling

图 3(a)阐述的是针对非完美循环的控制流解耦调度。首先，将外层循环（子图 1）在整个阵列上循环展开 2 次并行执行，如图 3(a)两个虚线框所示，并将中间数据 k_row[ ]和 temp[ ]存入片上缓存的相应区域。当片上缓存的相应区域存满之后，再切换阵列配置执行内层循环（子图 2），并从缓存中读出之前存入的中间数据 k_row[ ]和 temp[ ]。通过交替执行子图 1 和子图 2，实现控制流解耦。

图 3(b)阐述的是针对分支的控制流解耦调度。首先，执行所有的分支判断条件，将执行 false 分支的中间变量（i1，i2）和执行 true 分支的中间变量（i1，i2）分别存在片上缓存的相应区域。之后将 false 分支（子图 1）在阵列上并行展开 2 次同时执行，如图 3(b)中 2 个虚线框所示；当 false 分支全部执行完后，或片上缓存中存放 false 分支中间数据的相应区域满后，再切换阵列配置执行 true 分支（子图 2）。同样，2 个子图交替执行，实现控制流解耦。

图 3(c)阐述的是循环依赖解耦调度。与上述 2 种解耦类似，通过分别执行具有数据依赖关系的 2 个子图，实现控制流解耦。同样，在执行单个子图时，对其进行充分的循环展开以尽可能利用片上资源。

2.2　子图动态调度器

为支持上述子图间的动态调度，本文设计了一个子图调度器，如图 4 所示。由于每个子图执行所需要的数据和产生的中间数据都存放在片上缓存相应的子存储器（bank）内。因此，如果当前子图已经消耗完对应 bank 中所有的数据或者产生的中间数据已经将对应 bank 放满，则需要结束当前子图，切换为下一个子图。此时，子图调度器会根据当前存储器中 bank 的状态，通过子图检测电路判断出当前每个子图的状态，然后，通过优先编码器从已经准备就绪的子图中，动态选出一个作为接下来要执行的子图，并将该子图的 ID 绑定在数据流上，发送给阵列。

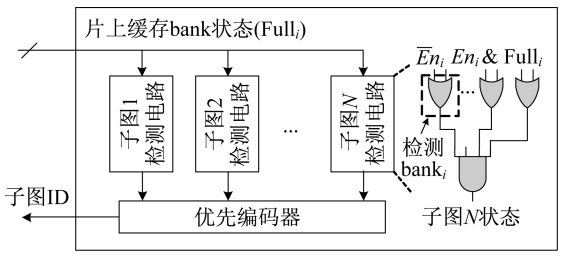

图 4　子图动态调度器硬件结构

Fig.4　Structure of the dynamical scheduler of a subgraph

2.3　支持子图切换的 PE 单元设计

为支持上述子图配置的动态切换，在现有的 PE 结构中增加一个配置切换单元，如图 5 所示。配置切换单元中会记录当前子图 ID，并同时监测每个周期携带在输入数据上的子图 ID。当输入数据上的子图 ID 和记录的子图 ID 不同时，将触发配置切换，同时更新当前记录的子图 ID。配置单元会根据新子图 ID 从配置缓存器中读出相应的一个新配置，并通过配置更新电路逐级改变 PE 的配置信息，其中包括输入数据选择器（Mux）、输出数据选择器（Demux）和算术逻辑单元（Arithmetic and Logic Unit，ALU）配置。

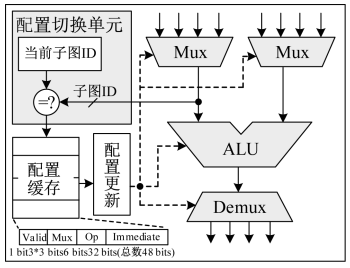

图 5　支持子图切换的 PE 内部结构

Fig.5　Inner structure of PE with subgraph switching

按上述方式，PE 内部的配置信息可以随数据流逐级切换。对于 PE 内部的某一流水级而言，在当前周期完成子图配置更新后，在下一周期即可执行更新后的子图所对应的数据流。因此，可以避免现有的动态配置切换技术中，由于在流水线中引入气泡所导致的流水线停顿现象[11]。

3　实验结果与分析

3.1　实验设置

从 3 种专用加速器领域广泛使用的测试集中选择了多个含非一致性控制流的代表性应用，见表 1。

表 1　用于进行评估对比的测试应用

Tab.1　Workloads for evaluation

应用名称	非一致性控制流类型	所属测试应用集
通用矩阵乘，Viterbi 动态规划	非完美循环	MachSuite[11]
Sort 排序，FFT	分支执行	
流体动力学	非完美循环	Rodinia[12]
HotSpot 检测	分支执行	
LU 分解，GE 消元	循环依赖	
Gesummv	非完美循环	PolyBench[13]
Cholesky 分解	循环依赖	

为了对比分析,本文选用了2种典型的CGRA结构,2种结构分别是完全静态配置的Plasticine[9]和指令调度的TIA[10]。为了公平比较,本文将通过对3种架构设定不同的阵列大小,使其具有相同的阵列面积,其中,3种架构的具体配置参数见表2。

表2 架构参数设定

Tab.2 Architecture parameter setting

项目	参数	文献[9]	文献[10]	本文
PE	指令/子图缓存大小/bit		16×128	8×48
	面积/μm² @800 MHz	10 327	31 090	13 745
片上缓存大小	子存储器端口数量	16		
	缓存容量/kB	64		
	面积/μm²	266 160		
系统参数	阵列大小	8×8	4×5	6×8
	面积/mm²	0.927	0.887	0.933

性能评估方面,采用的试验平台是基于C++编写的系统级模拟器。模拟器含周期精确的PE阵列和64 kB片上缓存,同时使用了周期精确的DRAMSim2仿真器[15]对内存延迟和带宽进行评估。对于Plasticine[9]和TIA[10]2种结构,通过提供的开源实例进行了行为级建模。此外,选用了一种灵活的片上存储模型Buffets[16]作为3种架构的片上访存接口。面积评估方面,本文通过Synopsys的Design Compiler在40 nm工艺库下对3种架构的硬件实现进行综合,设定时钟约束都为800 MHz。同时,通过Synopsys PrimeTime对3种架构的功耗进行评估。

3.2 性能和PE利用率评估

3种架构的性能和PE利用率如图6所示。其中,当子图个数为m,第i个子图执行的周期数为T_i,有效映射PE数量为N_i,周期t时刻处于激活状态的PE数量为n_t时,PE利用率为

$$r=\frac{\sum_{t=1}^{T}n_t}{\sum_{i=1}^{m}N_i\times T_i}\times 100\% \tag{1}$$

式中:r为PE利用率。

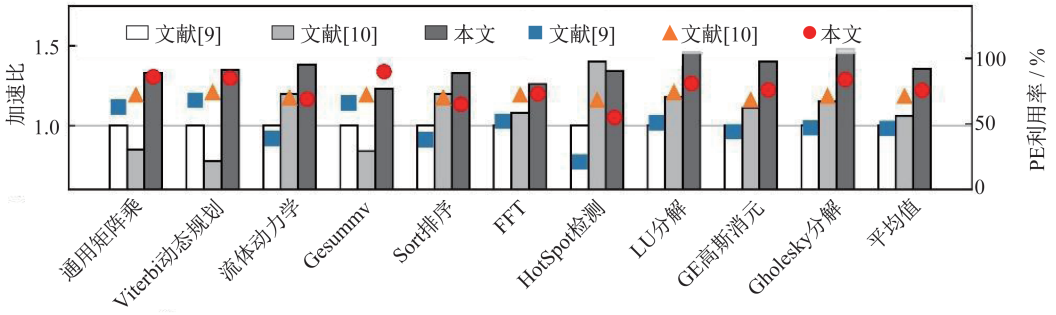

图6 3种架构间PE利用率(散点图)和性能加速比(柱状图)的对比(以文献[9]为基准)

Fig.6 Comparison of PE utilization ratio (in marker) and performance acceleration ratio (in bar) among three architectures normalized to Plasticine[9]

对于包含非完美循环的应用,如GEMM通用矩阵乘、Viterbi动态规划和Gesummv而言,外层循环所占比例和内层循环相当,完全静态调度的Plasticine中几乎有50%的PE处于闲置状态。TIA和本文提出的设计可以分别通过灵活的指令和子图级别的调度,将PE利用率提升到80%左右。本文的设计相比Plasticine取得了约为1.3倍的加速比,而TIA由于指令缓存的面积消耗,因此,阵列规模要小得多,平均性能只有Plasticine的0.8倍。

对于包含分支的应用,如Sort排序算法、FFT

傅里叶变换和 HotSpot 检测算法，由于有大量的 elseif 分支，因此，Plasticine 中的 PE 利用率大大降低。而 TIA 通过位于不同分支路径的指令合并映射到同一个 PE 上，因此，提高了 PE 利用率，并相比 Plasticine 性能上取得了 1.2 倍的加速比。而本文的设计通过控制流解耦，取得了 1.3 倍的加速比。

对于存在循环依赖的应用，如 LU 分解、GE 高斯消元和 Cholesky 分解，Plasticine 中显式的数据同步降低 PE 利用率到约 45% 左右。而 TIA 和本文的设计通过指令和子图级的调度方式，取得了 75% 的 PE 利用率。

3.3　执行能效评估

上述 3 种架构的执行能效比如图 7 所示。

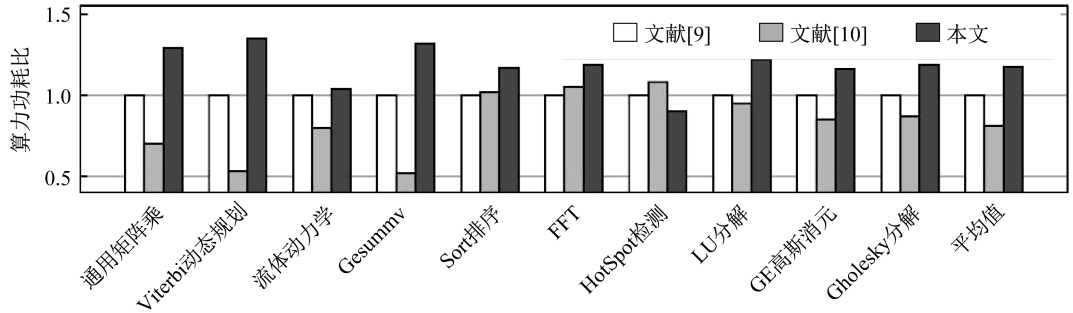

图 7　3 种架构间的执行能效对比[9]

Fig.7　Comparison of energy efficiency among three architectures normalized to Plasticine[9]

图中，尽管 TIA 取得了相对较高的 PE 利用率和性能，但是细粒度的指令调度显著增大了执行的能耗。因此，对于大部分应用，其在能效比方面均要低于 Plasticine。而本文的设计，通过高效的子图粒度的解耦和重调度，在大多数应用中取得了较高的能效比。而对于 HotSpot 检测和 CFD 流体动力学这类应用，由于频繁的子图切换，各个子图执行过程中的中间数据需要暂存在片上缓存中，因此，会增加片上缓存的访问次数，进而导致额外的能耗增加。

4　结束语

针对应用中出现的非一致性控制流，本文提出了一种基于控制流解耦的调度方案。通过将处于不同控制流边界下的数据流解耦成多个可独立执行的子图，每个子图可以在阵列上充分展开以提高阵列的计算利用率。实验结果表明：相比于传统的直接映射方法，本方案可以通过在典型的 PE 内部增加轻量化的调度单元，进而提高可重构阵列的资源利用率和执行性能。

参考文献

[1] LIN X, YIN S, LIU L, et al. Exploiting parallelism of imperfect nested loops with sibling inner loops on coarse-grained reconfigurable architectures[C]// IEEE Asia and South Pacific Design Automation Conference. 2016: 456-461.

[2] RONG H, TANG Z, GOVINDARAJAN R, et al. Single-dimension software pipelining for multidimensional loops [C]// IEEE International Symposium on Code Generation and Optimization. 2004:163-174.

[3] LEE J, SEO S, LEE H, et al. Flattening-based mapping of imperfect loop nests for CGRAs [C]// ACM International Conference on Hardware/Software Code Sign and System Synthesis. 2014: 1-10.

[4] 俞磊，景乃锋，王琴 . 一种基于数据流驱动的混合粒度可重构阵列架构[J]. 微电子学与计算机，2019，36(5)：19-22，28.

[5] KARUNARATNE M, WIJERATHNE D, MITRA T, et al. 4D-CGRA: Introducing branch dimension to station-temporal application mapping on CGRAs[C]// IEEE/ACM International Conference on Computer-Aided Design (ICCAD). 2019: 1-8.

[6] BALASUBRAMANIAN M, DAVE S, SHRIVASTAVA A, et al. LASER: a hardware/software approach to accelerate complicated loops on CGRAs [C]// IEEE Automation & Test in Europe Conference & Exhibition (DATE). 2018: 1069-1074.

[7] VOITSECHOV D, ETSION Y. Inter-thread communication in multithreaded, reconfigurable coarse-

grain arrays [C]// IEEE/ACM International Symposium on Microarchitecture (MICRO). 2018: 42-54.

[8] WENG J, LIU S, WANG Z, et al. A hybrid systolic-dataflow architecture for inductive matrix algorithms [C]// IEEE International Symposium on High Performance Computer Architecture (HPCA). 2020: 703-716.

[9] PRABHAKAR R, ZHANG Y, KOEPLINGER D, et al. Plasticine: a reconfigurable architecture for parallel patterns [C]// ACM/IEEE Annual International Symposium on Computer Architecture (ISCA). 2017: 389-402.

[10] PARASHAR A, PELLAUER M, ADLER M, et al. Triggered instructions: a control paradigm for spatially-programmed architectures [J]. ACM SIGARCH Computer Architecture News, 2013, 41(3): 142-153.

[11] VOITSECHOV D, ETSION Y. Single-graph multiple flows: energy efficient design alternative for GPGPUs [J]. ACM SIGARCH Computer Architecture News, 2014, 42(3): 205-216.

[12] REAGEN B, ADOLF R, SHAO Y S, et al. Mach suite: benchmarks for accelerator design and customized architectures [C]// IEEE International Symposium on Workload Characterization (IISWC). 2014: 110-119.

[13] CHE S, BOYER M, MENG J, et al. Rodina: A benchmark suite for heterogeneous computing [C]// IEEE International Symposium on Workload Characterization (IISWC). 2009: 44-54.

[14] POUCHET L N. Polybench: the polyhedral benchmark suite [EB/OL]. (2012-03-19) [2021-02-27]. http://web. cse. ohio-state. edu/~pouchet. 2/ software/polybench/.

[15] ROSENFELD P, COOPERBALIS E, JACOB B. DRAMSim2: a cycle accurate memory system simulator [J]. IEEE Computer Society, 2011, 10(1): 16-19.

[16] PELLAUER M, SHAO Y S, CLEMONS J, et al. Buffets: an efficient and compostable storage idiom for explicit decoupled data orchestration [C]// ACM The Twenty-Fourth International Conference. 2019: 137-151.

(上接第 67 页)

[4] TI. TMS570LS3137-EP 16- and 32-Bit RISC Flash Microcontroller [EB/OL]. (2015-02-20) [2021-03-01]. https://www.ti.com/lit/ds/symlink/tms570ls3137-ep. pdf? ts=1626068581351&ref_url=https%253A%252F%252Fwww. ti. com%252Fproduct%252FTMS570LS3137-EP.

[5] ITURBE X. A triple core lock-step (TCLS) ARM® Cortex®-R5 processor for safety-critical and ultra-reliable applications [C]// 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops. Washington D. C., USA: IEEE Press, 2016: 246-249.

[6] 陶飞,马昕,胡天亮,等.数字孪生标准体系[J].计算机集成制造系统,2019,25(10): 2405-2418.

[7] 韩凤宇,林益明,范海涛.基于模型的系统工程在航天器研制中的研究与实践[J].航天器工程,2014,23(3): 119-125.

[8] 卢志昂,刘霞,毛寅轩,等.基于模型的系统工程方法在卫星总体设计中的应用实践[J].航天器工程,2018,27(3):7-16.

[9] 孙红俊,张文杰,张利艳.美欧先进军工企业航天制造智能化发展分析[J].智能制造,2019(6):26-31.

[10] 谢浩,张健.基于SPARC V8的嵌入式星载计算机[J].微计算机信息,2008(5):39-38.

[11] BERGER R, ARTZ D, KAPCIO P. RAD750 ™radiation hardened PowerPC™ microprocessor [EB/OL]. (2018-07-08) [2021-03-01]. http://caxapa. ru/thumbs/440955/download.pdf.

[12] 胡振波.手把手教你设计CPU:RISC-V处理器篇[M].北京:人民邮电出版社,2018.

[13] PATTERSON D A, HENNESSY J.计算机组成与设计:硬件/软件接口[M].5版.北京:机械工业出版社,2015.

[14] PATTERSON D A, HENNESSY J.计算机体系结构:量化研究方法[M].5版.北京:人民邮电出版社,2013.

[15] AMBA: The standard for on-chip communication [EB/OL]. (1999-05-07) [2021-03-01]. https://www.arm.com/ products/ silicon-ip-system/ embedded-system-design/ amba-specifications.