

引用格式:钱恺宸,尹琛,景乃锋.一种基于数据流解耦的可重构阵列映射方法[J].微电子学与计算机,2021,38(8):28-33.
[QIAN K C, YIN C, JING N F. A mapping method for reconfigurable array based on data flow decoupling[J]. Microelectron-
ics & Computer, 2021, 38(8): 28-33.]

一种基于数据流解耦的可重构阵列映射方法

钱恺宸,尹琛,景乃锋

(上海交通大学 电子信息与电气工程学院, 上海 200240)

摘 要:可重构阵列依靠数据流驱动带来的能效优势被作为加速器广泛运用在特定领域之中.随着应用范围的增大,当应用中存在不同执行速率的区域时,采用传统的空间映射方案将整个数据流图进行直接映射会造成严重的性能损失.提出了一种基于数据流解耦的映射方法,通过在执行速率不同的区域之间加入解耦单元以解决执行速率失配的问题.同时利用一种分布式多阶段的布局算法,将解耦后的数据流图映射在分簇式的互连结构中.与传统映射方案相比,对数据流进行解耦后的映射方案在保持较高的互连布通率的基础上可以平均提升 57.68% 的执行性能,同时降低 32%~43% 的互连开销.

关键词:可重构阵列;数据流解耦;分簇式互连;布局布线

中图分类号: TP303

文献标识码: A

文章编号: 1000-7180(2021)08-0028-06

A mapping method for reconfigurable array based on data flow decoupling

QIAN Kaichen, YIN Chen, JING Naifeng

(School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Reconfigurable arrays are widely used as accelerators in specific fields relying on the energy efficiency advantages brought by the driving of data flow. With the increase of application range, when there are regions with different execution rates in the application, it will cause serious performance loss by adopting the traditional spatial mapping scheme to map the whole data flow graph directly. A mapping method based on data flow decoupling is proposed to solve the mismatch of execution rate by adding decoupling units between regions with different execution rates. At the same time, a distributed multi-stage placement algorithm is used to map the decoupled dataflow graph into a clustered interconnection structure. Compared with the traditional mapping scheme, the mapping scheme after decoupling the data flow can improve the execution performance by 57.68% on average while maintaining a high interconnection routability and reduce interconnection costs by 32% to 43%.

Key words: reconfigurable array; data flow decoupling; clustered interconnection; placement and routing

1 引言

可重构阵列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域中.整个阵列由大量执行单元(Processing Element, PE)组成,每个 PE 以数据流驱动的模式执行操作.通过空间映

射的方式,一旦输入数据有效,大量 PE 便可以同时并行执行.这极大地提高了阵列的计算吞吐率.

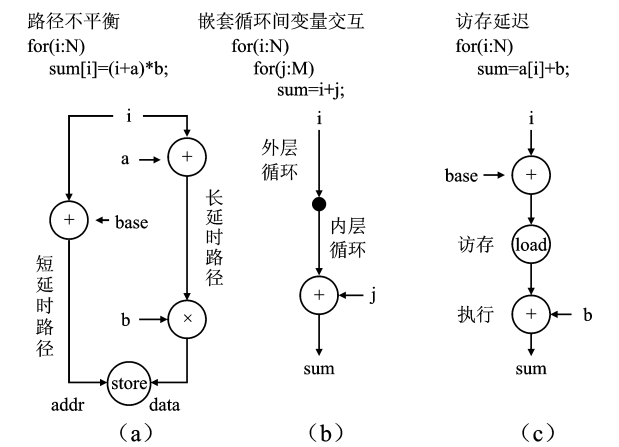
随着可重构阵列的应用范围扩大,当采用传统的映射方案对数据流图(Data Flow Graph, DFG)进行直接映射时,会面临因数据流执行速率不一致而引起的执行效率下降的问题,例如图 1 中所示的

非平衡路径,嵌套循环间变量交互以及访存延迟等.传统的可重构阵列映射技术^[1~3]只考虑了 DFG 映射的功能正确性,而并未对 DFG 中无法直接体现出的数据流执行速率的差异性进行优化.

针对上述问题,本文提出了一种基于数据流解耦的可重构阵列映射方法.首先,在原始 DFG 中数据流执行速率不一致的区域之间插入缓存进行解耦.之后,利用一种分布式多阶段的布局算法将解耦后的区域分布映射到分簇式的互连结构上.借助这种数据流解耦的映射方案,通过在原始 PE 阵列中增加约 5%面积开销的解耦单元,即可在保持较高布通率的基础上,相比于传统的映射方案平均提升 57.68%的执行性能,同时降低 32%~43%的互连开销.

2 多速率数据流执行特性分析

本节将会对图 1 中三种常见的数据流执行速率不一致现象进行分析,并阐述数据流执行速率失配对阵列性能的影响.



(a) 路径不平衡; (b) 嵌套循环间变量交互; (c) 访存延迟

图 1 三种常见的数据流执行速率失配现象

图 1(a)展现了一个路径不平衡的例子.其中,两条路径共享相同的数据生产节点和数据汇聚节点,但是却拥有不同的执行延迟.假设短路径的执行周期为 S ,长路径的执行周期为 L .由于汇聚节点(此例中为 store 节点)需要匹配来自两条路径的操作数,因此来自于短路径的数据将会填满汇聚节点的输入缓存,并最终通过背压(backpressure)机制阻塞数据生产节点.这会导致数据流的吞吐率缩减为 S/L .传统的数据映射方案^[4~5]通常通过插入冗余 PE 的方式对短路径进行填充,但这会浪费用于计算的 PE 资源.

图 1(b)展现了在嵌套循环中,内外层循环体迭

代次数不一致的例子.由于外层循环在内层循环迭代执行 M 次后才会执行一次,因此循环变量 i 在内外层循环边界上会拥有不同的执行速率.这会导致内层循环的流水线停滞,直到外层循环产生下一轮迭代 $i+1$ 的数据.流水线停滞引起的填充流水开销会显著影响可重构阵列的执行性能,尤其对于内层循环体较小的应用.已有的映射方案^[6]通过循环交换的方式减少内层循环的迭代次数,但未能完全消除流水线停顿引起的填充流水开销.

图 1(c)展现了访存操作在数据流执行中引起的执行速率不一致的情况.访存单元从片外存储请求数据时存在较长的访存延迟.当访存请求响应时,大量的数据会以猝发(burst)的方式在很短的时间内传输给 PE.然而在软流水的执行模式下,PE 会以恒定的速率进行计算.因此,数据访存和数据执行的速率出现不匹配,这会引起流水线阻塞并最终导致阵列吞吐率下降.若下一轮循环迭代中的访存操作需要等待当前循环中的计算结束后再触发,即使在内存带宽足够的情况下仍会造成性能损失.文献^[7]通过动态数据流乱序执行的方式隐藏访存延迟,但是会额外引入不同轮次间数据匹配的开销.

以上三种数据流执行速率失配现象都会将原始的 DFG 分割成具有不同执行速率的区域.在软流水的执行模式下,这些区域间的数据传递都会引起流水线停顿问题,进而显著影响阵列的执行效率.

3 数据流解耦的映射方案设计

针对以上问题,本文提出了一种数据流解耦的映射方案,整个流程如图 2 所示.首先,在原始 DFG 中执行速率失配的区域间加入解耦单元(Decouple Element, DE),对数据流进行解耦.其次,利用一种“局部紧密,全局稀疏”的分簇式互连结构,将解耦后的数据流分隔在若干个规则的互连区域内.最后,通过一种分布式多阶段的布局算法,对每块互连区域内的数据流进行物理布局.

3.1 基于解耦的数据流优化

针对图 1 中的示例,通过加入解耦单元对执行速率失配的数据流进行解耦,具体细节如图 3 所示.

对于图 1(a)中所示的路径不平衡,可以在短路径上加入解耦单元作为缓存进行延时,如图 3(a)所示.由于 DE 中缓存大小固定,为了应对不同应用中路径长短的差异,DE 之间可以通过灵活地串接组成不同大小的缓存.因此,通过对不同路径的执行周期进行分析,在短路径上串接合适数量的 DE 进行

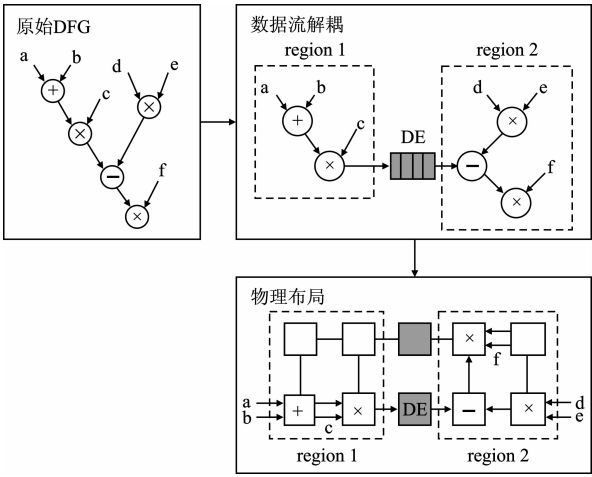


图 2 基于数据流解耦的映射流程

延时,可以实现长短路径间的数据流解耦。

对于图 1(b)所示的嵌套循环间变量交互,可以在内外层循环间加入 DE 对循环变量进行缓存,这样内层循环可以在执行结束时立刻从 DE 获取下一轮循环所需的外层数据,从而实现无停顿的流水化执行,如图 3(b)所示. 同样,外层循环可以利用内层循环的计算时间进行下一轮迭代变量的计算,使外层循环也可以无停顿地流水化执行完 N 次迭代。

对于图 1(c)中所示的访存延迟,可以在访存单元和执行单元间插入一个具有灵活流水线级数的 DE 作为解耦合空间,有效缓解访存延迟引起的流水线停滞,如图 3(c)所示. 和访存计算解耦合(DAE)类似,访存响应通过突发机制预取的数据可以缓存在 DE 中,并在发生访存延迟时将数据持续发送给执行端,避免访存端和执行端数据流速率不匹配造成的流水线停滞。

DE 解耦单元的设计参考了文献[8],其内部结构如图 4 所示,其输入输出可以根据配置设定为 PE 或者其它 DE. 当用作解耦访存延迟时,DE 从 PE 接收访存地址并发送给存储系统. 为了维持访存顺序,DE 中通过设置一个范围和内部缓存大小相等的自加器,为每一个访存请求标记一个 tag,并将访存得到的数据按照访存 tag 存入内部缓存中相应的条目(entry). 此外,可以通过配置 DE 的输入/输出为其它 DE,将多个 DE 串接起来以满足不同应用场景下对缓存容量的需求。

3.2 分簇式互连结构

为了保证布局的成功率,需要为每个解耦后的数据流内部提供规则且丰富的互连资源. 而不同数

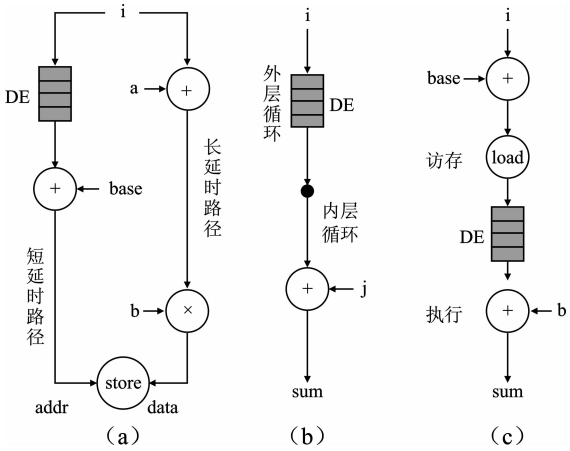


图 3 通过加入解耦单元对(a)路径不平衡;(b)嵌套循环间变量交互;(c)访存延迟进行数据流解耦

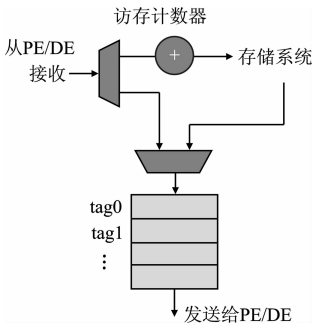


图 4 DE 解耦单元内部结构图

据流之间为了满足和 DE 的互连需求,则需要更为灵活稀疏的互连结构,以降低全局的互连开销。

针对这种情况,设计了一种“局部紧密,全局稀疏”分簇式互连结构,如图 5 中所示. 对于一个 8×8 结构的 PE 阵列,将其分割成 16 个 2×2 的簇,每个簇内的 PE 都和其它 3 个 PE 全互连(如虚线框内所示);而簇间只需提供少量的互连线,即为相邻簇同位置的 PE(如斜线部分所示)提供直连功能,以满足簇间的互连需求. 对于需要长距离互连而又没有提供直连线的 PE,本文采取文献[3]中提出的方案,通过在 PE 上增加 bypass 端口来实现数据的路由转发功能. 即 PE 可以同时执行正常端口的 ALU 操作和路由端口的数据转发,并且两者可以根据需要独立配置. 因此通过上述将整个阵列的互连网络划分成簇,并在簇内和簇间提供不同互连结构的形式,可以在保证阵列的互连布通率的基础上降低全局的互连开销。

3.3 分布式多阶段布局算法

针对上述分簇式的互连结构,本文提出一种分布式多阶段映射算法. 具体算法如下: 首先,将解耦

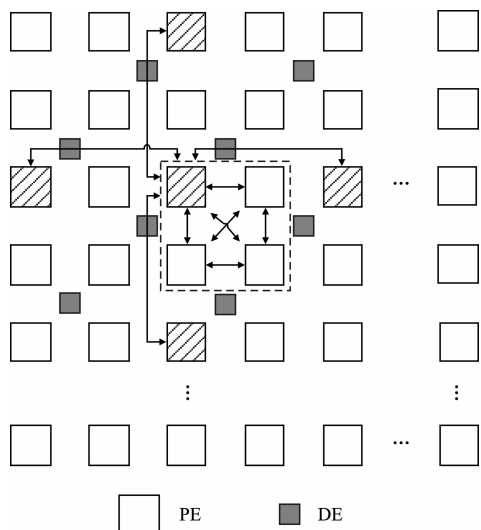


图 5 分簇式互连结构

后的数据流图根据节点数目分别确定所需互连簇的数量。其次,将解耦后的数据流图按照簇的大小进行拆分,并通过第一阶段布局算法确定簇内 PE 间的互连。之后,通过第二阶段布局算法确定各个簇在阵列上的实际布局以及簇间 PE 和 DE 之间的互连,并最终得出完整的 DFG 映射方案。

其中,存在数据传递的节点间的最大传输延时限制在一个时钟周期之内。为了避免不同路径的互连线传输延时的差异,在布局算法中我们加入了节点间最大路径长度的约束,以保证最长路径的传输延时也能满足时序要求。

两个阶段的布局算法都采用模拟退火算法,其中的主要参数分别是模拟退火参数和代价函数。每个温度点移动数目参数 L_k 和退火终止条件均采取文献^[9]中的公式,如下所示:

$$L_k = \mu N_{\text{blocks}}^{\frac{4}{3}}, \quad \mu = 10$$

$$T < \epsilon \frac{\text{Cost}}{N_{\text{nets}}}, \quad \epsilon = 0.005$$

温度衰减函数仿照热力学公式和线性衰减函数设计,使整体接受度曲线近似于文献^[9]中提出的曲线,如下所示:

$$T_{k+1} = \begin{cases} T_k e^{\frac{-0.045T_k}{\sigma}}, & I < 11 \text{ and } \delta > 0 \\ T_k e^{\frac{-0.03T_k}{\sigma}}, & I < 22 \text{ and } \delta > 0 \\ T_k e^{\frac{-0.0001T_k}{\sigma}}, & I < 97 \text{ and } \delta > 0 \\ 0.85T_k, & \text{otherwise} \end{cases}$$

代价函数综合考虑了连接单元在阵列中的棋盘距离(互连最短线长)和图距离(互连最少线数),以及 bypass 端口使用的冲突情况。参考 FPGA 里对互

连拥塞情况的分析^[10],新型代价函数如下所示:

$$\text{Cost} = \text{WireNum} + \lambda_1 \text{WireLength} + \lambda_2 \text{Conflifx}$$

$$\text{其中 } \lambda_1 = \frac{1}{\text{MaxWireLength}}, \text{MaxWireLength 表}$$

示阵列上最远的两个单元之间的棋盘距离。这样当图距离更短时,选择最少线数的互连路线;当图距离长度相同时,选择最短线长的互连路线。 λ_2 取经验值 0.01。

4 实验结果与分析

测试应用集取自 MachSuite^[11],这是一个针对通用计算的应用程序样本集。实验从 MachSuite 中挑选了六个适合阵列加速的应用进行测试。

实验中采用的阵列结构如图 3 所示,其中 PE 规模为 8×8 ,DE 数量为 24 个,其中每个 DE 内部缓存大小为 4。

4.1 性能对比

本文采用的试验平台是基于 C++ 编写的系统级模拟器。模拟器包含周期精确的 PE 阵列(包含 DE)和 32KB 片上缓存,同时使用了周期精确的 DRAMSim2 仿真器^[12]对内存延迟和带宽进行评估。此外,为了评估单个 DE 的面积开销,本文用 Verilog 实现了 DE 的 RTL 模型,并用 Synopsys Design Compiler 在 40nm 工艺库下进行了综合。

通过对每个应用原始的执行周期数和加入 DE 进行数据流解耦优化后的执行周期数进行评估,实验结果如表 1 所示。

对于应用 NW 和 SPMV,由于内层循环迭代次数较少,循环间交互的流水线停顿成为了性能的主要瓶颈。同时由于内外层交互的循环变量数量较少,因此只需要加入少量 DE 进行解耦就可以分别缩短 44% 和 57% 的执行周期数。应用 MD 中虽然访存指令占比最少,但是路径不平衡现象最为严重,因此需要 11 个 DE 对短路径的数据流进行缓存延时,优化后的执行周期减少了 66%。而对于只存在少量路径不平衡现象的 BFS,只需要 4 个 DE 即可缩短 58% 的执行周期。双调排序和脉动矩阵乘中因为分别需要频繁地将比较的中间结果和累加的部分和在片上缓存上进行存取,因此访存指令占比较多,所以需要 18 个 DE 作为 DAE fifo 对访存操作和计算操作进行解耦,优化后的执行周期分别缩短了 57% 和 65%。根据 DC 综合的结果,在 40 nm 工艺下单个 DE 的面积为 $1184.8 \mu\text{m}^2$ 。通过将文献^[13]中公布的 PE 面积折算到相同工艺,可以得出在 64 个 PE

的阵列中加入 24 个 DE 后,增加了 4.3%的面积开销.

综合来看,通过插入 DE 进行数据流解耦,可以在增加不到 5%的面积开销的前提下,使阵列的平均执行周期缩短 57.68%.

4.2 布局资源开销对比

我们选取了三种传统的可重构阵列互连结构作为对比,分别是 Mesh, ADRES 中的 4Way1Hop 和 MorphoSys 中的 4Way2Hop,并且加入了相同数量的 DE. 我们以相邻 PE 的距离为单位,用每种互连结构提供的互连线总长度评估互连开销,如表 2 所示. 其中,传统的互连结构采用单次模拟退火布局算法,分簇式互连结构采取分布式多阶段布局算法.

表 1 流水周期数优化率

应用	NW 序列检测	BFS	MD 分子动力学	SPMV	双调 排序	脉动 矩阵乘	平均值
原始执行周期数	162 537	62 212	31 211	32 075	33 692	105 007	/
解耦后执行周期数	91 732	26 212	10 700	13 784	14 436	37 025	/
所需 DE 数量	3	4	11	7	18	18	10
性能提升(%)	43.56	57.87	65.72	57.03	57.15	64.74	57.68

表 2 不同互连结构下的布局资源开销对比

应用	互连开销 DFG 节点数	Mesh		4Way1Hop ^[14]		4Way2Hop ^[15]		分簇式	
		1 792		4 864		8 704		4 928	
		总线数	总线长	总线数	总线长	总线数	总线长	总线数	总线长
NW 序列检测	35	177	483	135	397	109	387	86	299
BFS	28	201	908	184	790	167	744	109	494
MD 分子动力学	43	/	/	255	872	214	788	144	488
SPMV	45	/	/	273	907	254	947	164	617
双调排序	51	/	/	/	/	273	871	179	563
脉动矩阵乘	48	/	/	/	/	271	938	207	706

5 结束语

针对可重构阵列中数据流执行速率不一致的情况,本文提出了一种基于数据流解耦的映射方案. 通过插入解耦单元解耦后的数据流图可以保持原有的执行速率以软流水的方式高效执行. 分簇式的互连结构可以在保证布局成功率的前提下,减小全局的互连开销. 实验结果表明,相比于传统的映射方案,本方案在保持较高的互连布通率的基础上,可以同时大幅提高执行性能和减少互连资源开销.

参考文献:

[1] YIN S Y, YAO X Q, LU T Y, et al. Conflict-free loop mapping for coarse-grained reconfigurable architecture with

表 2 评估了在不同的互连结构下,不同应用所需的最少互连总线数和最短总线长. 每种互连结构都重复布局 50 次,取最少开销结果记录.

根据实验结果可以得出,Mesh 和 4Way1Hop 的互连能力较弱,在需要大量的线作转接的前提下也只能完成简单应用(如 NW, BFS 等)的布局. 4Way2Hop 的互连能力相对较强,借助于更长的直连线可以成功布通互连复杂的应用,但丰富的互连资源也导致互连开销增大. 相比而言,分簇式的互连结构可以在布通率和互连开销之间进行很好的折中,因此相比于三种传统的互连结构,在保证较高布通率的前提下,可以平均减少 43%, 36%和 32%的互连总线长开销.

multi-bank memory[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(9): 2471-2485. DOI: 10.1109/TPDS.2017.2682241.

[2] ZHAO Z Y, LIU Y T, SHENG W G, et al. Optimizing the data placement and transformation for multi-bank CGRA computing system[C]//2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). Dresden, Germany; IEEE, 2018: 1087-1092. DOI: 10.23919/DAT.2018.8342173.

[3] 行华斌, 景乃锋. 一种基于多阶段模拟退火的异构可重构阵列布局算法[J]. 微电子学与计算机, 2020, 37(6): 1-5.

XING H Y, JING N F. A placement algorithm for HGRA based on multi-stage simulatedanneal[J]. Microelectronics & Computer, 2020, 37(6): 1-5.

- [4] CONG J, HUANG H, MA C Y, et al. A fully pipelined and dynamically composable architecture of CGRA[C]//2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines. Boston, MA, USA: IEEE, 2014: 9-16. DOI: 10.1109/FCCM.2014.12.
- [5] DE P, MANDAL C, PRAMPALLI U. Path-balanced logic design to realize block ciphers resistant to power and timing attacks[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 27 (5): 1080-1092. DOI: 10.1109/TVLSI.2019.2896377.
- [6] LIN X H, YIN S Y, LIU L B, et al. Exploiting parallelism of imperfect nested loops with sibling inner loops on coarse-grained reconfigurable architectures[C]//2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC). Macao, China: IEEE, 2016: 456-461. DOI: 10.1109/ASPDAC.2016.7428054.
- [7] 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. DOI: 10.1145/2678373.2665703.
- [8] HONG T, GUAN N, YIN C, et al. Decoupling the multi-rate dataflow execution in coarse-grained reconfigurable array[C]//2020 IEEE International Symposium on Circuits and Systems (ISCAS). Seville, Spain: IEEE, 2020: 1-5. DOI: 10.1109/ISCAS45731.2020.9181275.
- [9] BETZ V, ROSE J, MARQUARDT A. Architecture and CAD for deep-submicron FPGAs[M]. New York: Springer, 1999: 51-55.
- [10] MIRHOSEINI A, GOLDIE A, YAZGAN M, et al. Chip placement with deep reinforcement learning[J]. arXiv e-prints, 2020: arXiv:2004.10746.
- [11] REAGEN B, ADOLF R, SHAO Y S, et al. MachSuite: Benchmarks for accelerator design and customized architectures[C]//2014 IEEE International Symposium on Workload Characterization (IISWC). Raleigh, NC, USA: IEEE, 2014: 110-119. DOI: 10.1109/IISWC.2014.6983050.
- [12] ROSENFELD P, COOPER-BALIS E, JACOB B. DRAMSim2: A cycle accurate memory system simulator[J]. IEEE Computer Architecture Letters, 2011, 10(1): 16-19. DOI: 10.1109/L-CA.2011.4.
- [13] PRABHAKAR R, ZHANG Y Q, KOEPLINGER D, et al. Plasticine: A reconfigurable architecture for parallel patterns[C]//2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA). Toronto, ON, Canada: IEEE, 2017: 389-402. DOI: 10.1145/3079856.3080256.
- [14] SINGH H, LEE M H, LU G M, et al. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications [J]. IEEE Transactions on Computers, 2000, 49 (5): 465-481. DOI: 10.1109/12.859540.
- [15] MEI B F, 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. Berlin, Heidelberg: Springer, 2003: 61-70. DOI: 10.1007/978-3-540-45234-8_7.

作者简介:

钱恺宸 男,(1996-),硕士研究生.研究方向为计算机体系结构与可重构计算. E-mail: mikeqian@sjtu.edu.cn.

尹琛 男,(1995-),博士研究生.研究方向为计算机体系结构与可重构计算.

景乃锋 男,(1982-),博士,副研究员.研究方向为计算机体系结构.