doi:10.11959/j.issn.1000-436x.2015132

# 用于二维 RCA 跨层数据传输的旁节点无冗余添加算法

陈乃金[1,2]，冯志勇[1]，江建慧[3]

（1. 天津大学 计算机科学与技术学院，天津 300072；
2. 安徽工程大学 计算机与信息学院，安徽 芜湖 241000；3. 同济大学 软件学院，上海 201804）

**摘 要**：针对二维可重构单元阵列(RCA)硬件任务的跨层数据传输问题，提出了一种前序遍历回溯旁节点添加算法。该算法针对跨层输入树、跨层输出树 2 种类型的数据流图，保持了原有运算节点之间的逻辑关系，实现了旁节点的无冗余添加。给出了动态可重构系统划分映射的量化评估指标体系和流水化模型，给出了添加旁节点映射的临界条件。实验结果表明，基于相同的系统结构和划分映射算法，在满足临界条件的情况下，与不加旁节点映射算法相比，加旁节点映射在划分模块数，非原始输入输出次数、配置时间、总执行周期、功耗等方面均获得了较好的改进；与已有的先进算法相比，文中算法平均执行总周期降低了 23.3%($RCA_{5\times5}$)和 30.5%($RCA_{8\times8}$)，平均消耗功耗降低了 15.7%($RCA_{5\times5}$)和 18.6%($RCA_{8\times8}$)，从而验证了所提方法的合理性和有效性。

**关键词**：可重构单元阵列；数据流图；旁节点；临界条件；时域划分与映射

**中图分类号**：TP302 **文献标识码**：A

# Bypass node non-redundant adding algorithm for crossing-level data transmission in two-dimension reconfigurable cell array

CHEN Nai-jin[1,2], FENG Zhi-yong[1], JIANG Jian-hui[3]

(1. School of Computer Science and Technology, Tianjin University, Tianjin 300072, China;
2. School of Computer and Information Engineering, Anhui Polytechnic University, Wuhu 241000, China;
3. School of Software Engineering, Tongji University, Shanghai 201804, China)

**Abstract:** As for the problem of hardware task crossing-level data transmission, a preorder traversing backtracking adding_bypass_node (PTBA) algorithm is presented which maintains logic relation among original computing nodes and does not add redundancy nodes based on data flow graph with crossing-level-in-tree(CLIT) and crossing-level-out- tree (CLOT).The pipelined model of partitioning mapping and the quantitative evaluation indexes are presented for the dynamic reconfigurable system. The critical condition of PTBA mapping is proposed. Compared with preorder traversing backtracking no adding_bypass_node (PTBNA) mapping, and under the premise of critical condition,experimental results show PTBA mapping can improve the number of modules, the number of non-original input times and non-original output times, the total execution delay and powers of all partitioning based on the same system architecture and partitioning mapping algorithm. The proposed algorithm obtains the less average execution total cycles by 23.3%($RCA_{5\times5}$), 30.5%($RCA_{8\times8}$), and the less average power consumption by 15.7%($RCA_{5\times5}$),18.6%($RCA_{8\times8}$) than previous advanced split-push kernel mapping(SPKM). PTBA has rationality and effectiveness.

**Key words:** RCA; data flow graph; bypass node; critical condition; temporal partitioning and mapping

**收稿日期**：2014-06-10；**修回日期**：2014-12-19

**基金项目**：国家高技术研究发展计划（“863”计划）基金资助项目（2009AA011705,2013AA013204）；安徽省自然科学基金资助项目(1408085MF124)；国家自然科学基金资助重点项目（61432017）；芜湖市科技计划自然科学基金资助项目(芜科计字[2012]95 号)

**Foundation Items:** The National High Technology Research and Development Program of China (863 Program) (2009AA011705, 2013AA013204); The Natural Science Foundation of Anhui Province(1408085MF124); The National Natural Foundation of China (61432017); The Natural Science Foundation of Wuhu([2012]95)

## 1 引言

可重构处理器在硬件自重构图像处理、高清视频编解码加速等方面已经展示了巨大的优势和潜力[1~3]。以现场可编程门阵列(FPGA, field programmable gate array)为代表的细粒度可重构体系结构(FGRA, fine grained reconfigurable architecture)在进行位级运算时优势明显，但在处理规则的、位数宽的字级逻辑运算时效率较低。为了克服FGRA的缺点，研究人员相继提出了各种二维粗粒度可重构体系结构(CGRA, coarse grained reconfigurable architecture)[4~8]，包括可重构单元阵列(RCA, reconfigurable cell array)或称为处理单元阵列(PEA, processing element array)。如何进行多约束的时域划分与映射仍然是一个难题。目前，国际上针对不同架构的CGRA平台已经提出了多种不同的手动或自动数据流图(DFG, data flow graph)的划分与映射算法，但是已有工作对RCA块内跨层数据传输延迟等问题考虑不够[1,3~16]，文献[11]提出了加旁节点(BN, bypass node)进行映射的方法，但是该方法没有考虑添加BN的临界条件等问题，该临界条件是添加BN映射的主要瓶颈之一。

针对上述已有工作的不足，本文对此进行了深入的研究和探讨，提出了一种用于二维RCA跨层数据传输的旁节点添加方法，该方法的主要贡献如下。

1) 考虑了跨层数据传输互连延迟，并进行了最小化映射，建立了CGRA硬件任务编译器划分与映射的流水化模型，给出了较为合理的量化评估指标体系，并对BN添加等问题进行了形式化定义。

2) 针对跨层输入树CLIT (crossing-level-in-tree)、跨层输出树CLOT(crossing-level-out-tree)2种类型的DFG，提出了一种满足临界条件且追求互连延迟最小化的BN无冗余添加算法，在满足临界条件下，通过与不加BN映射算法相比较，加BN映射在非原始输入(出)次数、配置时间、划分块数等方面均获得了优化。

## 2 相关工作和跨层数据传输互连延迟

### 2.1 相关工作

CGRA可以获得好的加速性能，消耗的功耗也较少，具有一定的运算灵活性，所以国际上多个研究机构对其展开了大量研究，特别是2000年左右，CGRA的研究出现了一个高峰，二维CGRA的典型代表有PipeRench[4]、REMARC[5]、ADRES[6]、Morphosys[7]等，与其相应的片上系统(SOC, system on chip)的任务编译器研发极其重要，任务编译器是SOC高层逻辑综合与优化的关键，对其后期设计流片具有直接指导作用，而流水化时域划分与映射是实现任务编译器的重要一环，相关研究较多。

2000年及以后，文献[7]提出了一种手动或半自动的方法实现了多个基准的Morphosys映射；文献[9]设计了DRESC(dynamically reconfigurable embedded system compiler)编译器，目的是通过IPC(instructions per cycle)值来获得优化的编译性能；文献[10]设计的XPP-VC编译器实现了运算节点到RCA上映射，实验结果表明相对于Pentium Ⅲ处理器，映射到RCA后可获得两位数的加速比。

2009年至今，文献[11]基于RSPA(resource sharing and pipelining architecture)架构，并采用列展开加点，后行展开方式来进行映射，在硬件资源的约束下，一遇到块内跨层或不跨层的数据过渡就添加BN，且允许RC间数据通过互连线跨一层行或列传输，从而导致RCA内的数据传输延迟等指标的增大。文献[12]对浮点数映射进行了初步探讨。文献[13]基于ADRES架构，实现了H.264等多媒体应用的映射，该方法采用循环合并的方式来减少块间通信成本。文献[14]提出首先定位并映射好DFG的一个源节点，然后以源节点为起点，根据CGRA路由互连方式，依次尽可能地把与源节点有依赖的其他节点映射到RCA上去，避免了无效的路径搜索，从而获得了编译时间的改进。文献[15]提出了一种基于LEAP（loop engine on array processors）结构模型的循环映射算法，获得了较高的资源利用率和吞吐量。文献[16]研究了RCA互连定制和不精确图匹配映射等问题。但是上述工作对DFG的在RCA块内跨层数据传输产生的时延考虑不足，并且没有较明确给出CGRA时域划分与映射的量化指标体系。

### 2.2 跨层数据传输互连延迟

**定义1** （跨层数据传输互连延迟）就RCA而言，层的物理含义为其行号$Row_u$(或列号$Col_v$)，如图1(a)所示，其中，$0\leqslant u\leqslant Row-1$, $0\leqslant v\leqslant Col-1$，本文约定层表示RCA的行号；在满足RCA互连、面积等约束下，一DFG被划分映射到RCA后，每个划分内所有的边都能在硬件上实现正确连接，块

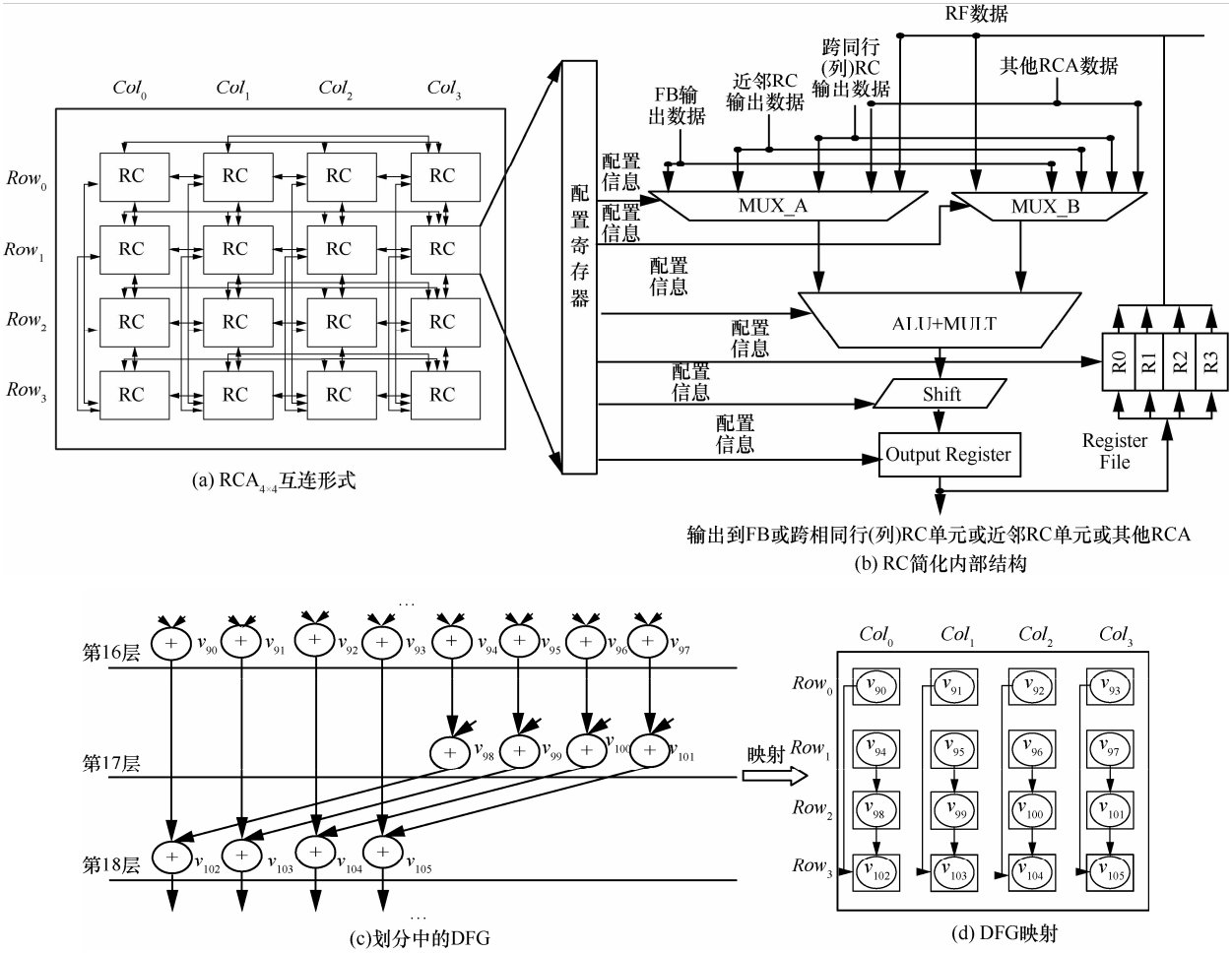



图 1　Morphosys 的 $RCA_{4\times4}$ 互连及 DFG 划分映射说明

内运算节点间的层差大于 1 称为跨层(行)；若 RCA 块内节点间存在跨层，则通过配置和计算获得的一个节点运算结果首先被存储在该点 RC 的寄存器中，等到同块该节点的后继要使用该运算结果时，就需要从寄存器中取出数据，然后通过 RC 之间的跨层互连线(路由器或总线等)传递到需要该数据的 RC 单元。这种由于 RCA 块内跨层数据传输而产生的延迟，称为跨层数据传输互连延迟，简称互连延迟(interconnection delay)，记为 $I_{ID}$。若 RCA 块内节点之间的层差为 1，则可进行点到点的直接数据流水传递，其 $I_{ID}$ 近似为 0。举例说明如下。

**例 1**　以 Morphosys 为例($RCA_{4\times4}$)，其 RC 简化内部结构如图 1(b)所示，一个 DFG 被划分映射后的情况如图 1(d)所示，在按行执行的过程中，$Row_0$ 的 $v_{90}$、$v_{91}$、$v_{92}$、$v_{93}$ 运算结束后，其结果被暂存在各自 RC 的寄存器文件(RF, registers file)中，$Row_3$ 的 $v_{102}$、$v_{103}$、$v_{104}$、$v_{105}$ 的一个操作数直接来自 $Row_2$ 相应节点的运算结果，另外一个操作数来自 $Row_0$ 的相应节点，则 $v_{90}$、$v_{91}$、$v_{92}$、$v_{93}$ 运算结果又要从各自 RC 单元的 RF 中被取出，各自消耗的 RF 存取延迟 $delay_{RF}$=2 cycle，然后进行跨层传输，各自消耗的 RC 单元间接传输延迟 $delay_{RC\,间接}$= RC 控制步长×跨层传输长度=0.5×3=1.5 cycle。综上，图 1(d)的累加 $I_{ID}$=4×2+4×1.5=14 cycle。

本文研究基于目标架构 REMUS[17,18]，在映射时考虑了 $I_{ID}$，表 1 给出了几种 CGRA 的比较。

上述 CGRA 均包含多路选择器 MUX、配置寄存器组、局部数据存储器等单元，在此没有给出。表 1 中相关符号全称列举如下：PR(pass registers)，ALU(arithmetic logical unit)，NN(nearest neighbor)，FU (functional unit)，MULT(multiplier unit)，MPE(memory processing element)，CPE(computing processing element)。为了说明 $I_{ID}$ 对 CGRA 加速性能的影响，给出如下约定：1) 每块 RCA 映射成功后，每一行节点均可并行执行，执行顺序是从 RCA 的第一行开始，依次进行；2) 采用贪婪深度优先搜

**表 1** **几种 CGRA 的比较**

| CGRA 名称 | RC 互连方式 | 行流水 | RC 间数据跨层传输 | RCA 块内数据通信大致模型 |
|---|---|---|---|---|
| RipeRench | 一维总线+RC 上下 NN 行路由互连 | 支持 | 允许跨层 | 总线+PR+ALU+NN 行互连 |
| REMARC | 两维总线+RC 上下左右 NN 互连 | 支持 | 允许跨层 | 总线+数据 RAM+ALU+NN |
| ADRES | RC 上下左右 NN+跨行（列）全互连 | 支持 | 允许跨层 | FU+RF+NN |
| Morphosys | RC 上下左右 NN+跨行（列）全互连 | 支持 | 允许跨层 | ALU+MULT+Shift+RF |
| LEAP | RC 上下左右 NN+开关路由器互连 | 支持 | 允许跨层 | MPE+CPE+节点路由 |
| REMUS | RC 上下 NN 行路由互连 | 支持 | 不允许跨层 | ALU+NN 行路由 |

**表 2** **4 种 CGRA 跨层传输延迟值(单位：cycle，$\delta=I_{ID}-S_{SD}$)**

| 划分基准 | CGRA | $I_{ID}$ | $S_{SD}$ | $\delta$ | 划分基准 | CGRA | $I_{ID}$ | $S_{SD}$ | $\delta$ |
|---|---|---|---|---|---|---|---|---|---|
| SODE | PipeRench | 34 | 6 | 28 | EWF6 | PipeRench | 1 342 | 39 | 1 303 |
| | REMARC | 32 | 6 | 26 | | REMARC | 1 272 | 39 | 1 233 |
| | ADRES | 3 | 6 | −3 | | ADRES | 134 | 39 | 95 |
| | LEAP | 12 | 6 | 6 | | LEAP | 536 | 39 | 497 |
| FEAL | PipeRench | 208 | 21 | 187 | MATRIX4 | PipeRench | 638 | 25 | 613 |
| | REMARC | 200 | 21 | 179 | | REMARC | 604 | 25 | 579 |
| | ADRES | 18 | 21 | −3 | | ADRES | 56 | 25 | 31 |
| | LEAP | 72 | 21 | 51 | | LEAP | 224 | 25 | 199 |
| FFT4 | PipeRench | 68 | 6 | 62 | FDCT6 | PipeRench | 240 | 50 | 190 |
| | REMARC | 64 | 6 | 58 | | REMARC | 228 | 50 | 178 |
| | ADRES | 8 | 6 | 2 | | ADRES | 30 | 50 | −20 |
| | LEAP | 24 | 6 | 18 | | LEAP | 120 | 50 | 70 |
| FFT8 | PipeRench | 272 | 9 | 263 | MEDIAN | PipeRench | 318 | 9 | 309 |
| | REMARC | 256 | 9 | 247 | | REMARC | 300 | 9 | 291 |
| | ADRES | 24 | 9 | 15 | | ADRES | 29 | 9 | 20 |
| | LEAP | 96 | 9 | 87 | | LEAP | 124 | 9 | 115 |

索且允许可跨层的映射方案，只考虑定点数跨层或数据过渡添加 BN 的问题；3) 只统计 RCA 块内节点的 $I_{ID}$。

一般而言，RCA 互连形式大致分为总线、路由、RC 直(或间)接互连等方式，其 $I_{ID}$ 计算方法分别为 $delay_{\text{总线}}$=总线基本传输延迟(20 cycle)+总线控制步长(6 cycle)×跨层传输长度[19]；$delay_{\text{路由}}$=路由基本传输延迟(8cycle)+路由控制步长(2 cycle)×跨层传输长度[19]；$delay_{RF(\text{或 }PR)}$ = 2 cycle (基本存取时间)；$delay_{RC\text{ 间接}}=delay_{RC\text{ 直接}}$(不跨层近似为 0) +RC 控制步长(0.5 cycle )×跨层传输长度。具体架构 $I_{ID}$ 计算方法分别为： PipeRench 的 $I_{ID}=delay_{\text{总线}}+delay_{RF}$；REMARC 的 $I_{ID}=delay_{\text{总线}}$；ADRES 和 Morphosys(仅考虑一块 $RCA_{4\times4}$) 的 $I_{ID}$ 近似相等，即 $I_{ID}=delay_{RF}+delay_{RC\text{ 间接}}$；LEAP 的 $I_{ID}=delay_{\text{路由}}$。

采用存在跨层现象的 8 个基准程序：SODE 、FEAL 、FFT4、FFT8、EWF6、FDCT6、MEDIAN、MATRIX4，RCA 的规模为 8×8，通过实验获得的 $I_{ID}$ 见表 2，其中，$S_{SD}$ 为一个 DFG 在 RCA 上执行所需的计算延迟。由表 2 可知，除了 ADRES(SODE，FEAL，FDCT6)，$I_{ID}$ 均大于 $S_{SD}$，最大为 PipeRench (EWF6)，其倍数为 1 342/39≈34 倍，所以 $I_{ID}$ 是制约系统加速性能关键，在进行 RCA 映射应给予考虑。

## 3 问题定义和评估指标体系

### 3.1 问题定义

为了便于说明问题，下面给出了可重构单元阵列模型(RCAM, reconfigurable cell array model)等定义。

**定义 2** RCAM 模型：一个二维 $Row \times Col$ 的

RCAM 是一个 4 元组，RCAM=($RC$,$I$,$O$,$E_L$)。其中，$RC$={$RC_{0,0}$, $RC_{0,1}$,…, $RC_{u,v}$}为一有限集合，每个 $RC_{u,v}$($0\leqslant u\leqslant Row-1$, $0\leqslant v\leqslant Col-1$)可进行算术逻辑运算，也可进行较为复杂运算，如移位加减法运算等；$I=I(RC_{0,0})\cup I(RC_{0,1})\cup\cdots\cup I(RC_{u,v})$，$I(RC_{u,v})$为 $RC_{u,v}$ 输入端口集合；$O= O(RC_{0,0})\cup O(RC_{0,1})\cup\cdots\cup O(RC_{u,v})$, $O(RC_{u,v})$为 $RC_{u,v}$ 输出端口集合；$E_L\subseteq O\times I=\{<o, i>|\ o\in O, i\in I\}$是一有限集合,其中每一个元素代表一 $RC$ 输出口到另一 $RC$ 输入口存在连接关系。

一般而言，图 $G$=($V$, $E$, $W$, $D$)经时域划分后[20]，其到 RCAM=($RC$, $I$,$O$, $E_L$)的映射是一对一的单射过程。一个 RCA 是由若干个 RC(或称 PE，processing element)通过各种不同互连方式构成的阵列，每个 RC 具有运算、路由等功能。每个运算符所占的硬件资源面积可用 RC 的个数来表示，因本文的研究对象是定点数运算，故一个运算符占用的资源为一个 RC，每个 RC 的高度和宽度均为 1。为了便于说明问题，本文后续研究约定 CGRA 包含一块 RCA，其面积为一定值且表示为 $A_{RPU}$。

**定义 3**　CLOT：如果给定 $G'$=($V'$, $E'$,$W'$,$D'$)$\subseteq$ $G$=($V$,$E$,$W$,$D$)，其中$|V'|\geqslant 2$，$\forall\ v_i,v_j,v_k\in V'$，$i\neq j\neq k$，$i,j,k\leqslant n-1$，$v_i$ 为 $v_j$,$v_k$ 共同的父节点，即 level($v_j$)−level ($v_i$)>1 且 level ($v_k$)−level ($v_i$)>1，level($v_i$)表示 $v_i$ 所在的层，若 2 个子节点在不同层，即|level($v_k$)−level ($v_j$)|≥1，或者 2 个子节点在同一层，即 level($v_k$)= level($v_j$)，则这样的一对多跨层子图称为 CLOT。CLOT 的例子如图 2(a)所示。有序运算符一对一跨层是 CLOT 的线性结构。

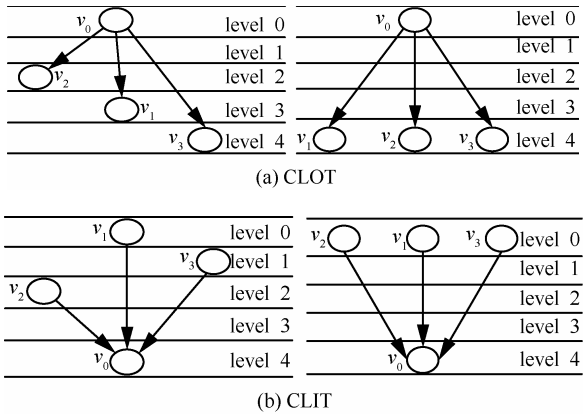



图 2　解释 DFG 跨层类型的例子

**定义 4**　CLIT：如果给定 $G'$=($V'$,$E'$,$W'$,$D'$)$\subseteq$ $G$=($V$,$E$,$W$,$D$)，其中$|V'|\geqslant 2$，$\forall\ v_i,v_j,v_k\in V'$，$i\neq j\neq k$，$i,j,k\leqslant n-1$，$v_i$ 为 $v_j$,$v_k$ 共同的子节点，即 level($v_i$)−level($v_j$)>1 且 level($v_i$)−level ($v_k$)>1，level($v_i$)表示 $v_i$ 所在的层，若 2 个父节点在不同层，即|level($v_k$)−level($v_j$)|≥1，或者 2 个父节点在同一层，即 level ($v_k$)=level ($v_j$)，则这样的多对一跨层子图称为 CLIT。CLIT 的例子如图 2(b)所示。

**定义 5**　BN：BN 是图 $G$ 中的运算符节点 $V$={$v_0$, $v_1$,…, $v_{n-1}$}映射到 RCA 上时，因硬件互连等约束，为了满足 RC 间的跨层数据传输或不跨层数据过渡而添加的数据存储转发的过渡点。它具有如下性质：1) 可以被配置寄存器组所配置；2) 配置时延或过渡传输时延为 1 cycle，运算时间忽略不计；3) 可以正确转发跨层或不跨层顶点对间的过渡传输数据。

**定义 6**　冗余 BN(RBN, redundant BN)：针对计算密集型任务关键循环提取出的包含一个或多个跨层输出(入)树的 DFG，为了实现一个 RCA 的块内跨层数据传输，而多加的 BN 点。

文中后续映射方法统一采用了最大图覆盖(MGC, max graph covering)，该方法是按照不同 RCA 的规模、互连等约束，对一 DFG 按不同时域划分算法进行无死锁的分割，每次提取 DFG 子图包含的节点尽可能多，因 RCA 面积等约束，虽然每次提取的子图较大但是不能完全被映射而会裁剪一部分，这时若 RCA 还有空闲资源，再次按约束映射下一个子图，由于每次分割的子图较大，且子图有时不能完全被映射，所以这种映射方法可称之为 MGC。

### 3.2　CGRA 划分映射的量化评估指标体系

文献[21]概括说明了可重构计算的任务执行时间，但没有给出具体分析且不完备,本文在此基础添加了 $I_{ID}$ 部分，执行一个的 DFG 运算所需的总周期数 $T_{TOTAL}$ 大致包括以下 5 个部分(如图 3 所示)，同时应该考虑功耗，这样评价 CGRA 划分映射量化评估指标体系为：$M$、$C_{CON}$、$N_1$、$N_2$、$S_{SD}$、$I_{ID}$、$T_{TOTAL}$ 和 $P_{POWER}$。其中，$M$ 表示一个 DFG 所用的划分块数(即一块 RCA 重复使用次数)；$C_{CON}$ 表示完成一个 DFG 运算所用的配置时间；$N_1$ 表示所有划分块间的非原始输入次数；$N_2$ 表示所有划分块间的非原始输出次数；$N_{org1}$ (或 $N_{org2}$) 表示一个 DFG 原始的输入(出)次数；$S_{SD}$ 和 $I_{ID}$ 的含义具体见 2.2 节相关部分；$T_{TOTAL}$ 包括数据输入时间(即 $N_1$+$N_{org1}$)、数据输出时间(即 $N_2$+$N_{org2}$)、$S_{SD}$、$C_{CON}$、$I_{ID}$；$P_{POWER}$ 表示完成一 DFG 消耗功耗(单位：mW)。

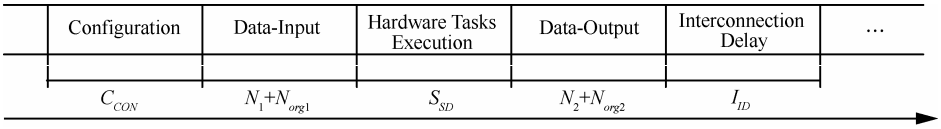



图 3 在 RCA 上执行一个 DFG 的时间说明

这些指标之间的关联性简述如下：$M$ 增大会导致 $C_{CON}$、$P_{POWER}$、$T_{TOTAL}$ 等的增大；$I_{ID}$、$S_{SD}$、$N_1$、$N_2$ 增大会导致 $T_{TOTAL}$ 增大；优化 $S_{SD}$，会导致 $N_1$、$N_2$ 的增大，反之亦然；若追求 $M$ 较小化，即进行贪婪放置会导致 $N_1$、$N_2$、$S_{SD}$ 的增大。若不考虑 $I_{ID}$ 最小化映射并且 $I_{ID}$ 较大时，影响 $T_{TOTAL}$ 从大到小排序大致为 $M$、$C_{CON}$、$I_{ID}$、$N_1+N_2$、$S_{SD}$ 等。影响 $P_{POWER}$ 的重要性排序大致为 $M$、$C_{CON}$。综上，一个好的 CGRA 划分映射算法要折中考虑 $M$、$C_{CON}$、$N_1$、$N_2$、$S_{SD}$、$I_{ID}$、$P_{POWER}$ 等指标。

### 3.3 CGRA 任务编译器划分与映射模型

本文提出的 CGRA 任务编译器划分与映射流水化模型如图 4 所示，划分器将一个 DFG 经过多个约束划分后，形成装载队列；映射装载器按节点划分的调度顺序依次将运算节点放入 RCA 阵列；在线添加器针对含有跨层子图或不跨层数据过渡子图在映射时进行 BN 在线动态添加(对不加 BN 映射，可省略)；反馈裁剪器配合映射放置器裁去不能放置的点，放入到待划分队列，如 RCA 还有空闲，再次启动划分器贪婪搜索满足要求的放置点进行流水化放置，每次按“该放就放尽可能填满整个 RCA 阵列”的原则进行，在硬件资源分配完成后，生成相应的配置字，控制可重构硬件形成所需的运算功能。

## 4 实验目标和 DFG 添加旁节点算法

### 4.1 实验目标

本文实验目标的定义和说明包括以下 2 点：1)满足 2-D REMUS 以及通过总线、路由、行列路由全互连等较为通用 CGRA 的运算阵列块内节点跨层或过渡数据传输、$I_{ID}$ 最小化、行操作节点并行执行等要求，针对 CLOT 和 CLIT 树 2 种类型的数据流，设计实现一个 BN 无冗余添加算法；2)获得 BN 无冗余添加的临界条件，同时与国际先进的 SPKM 算法进行比较。

### 4.2 DFG 添加旁节点算法

本文提出了前序遍历回溯添加旁节点(PTBA, preorder traversing backtracking adding_bypass_node)算法，通过调用 CLOT 和 CLIT 树的添加 BN 函数 *addnode_CLOT*( )和 *addnode_CLIT*( )来实现对跨层子图的流水化映射，如果不调用此函数，映射算法变为前序遍历回溯不添加旁节点(PTBNA, pre-order traversing backtracking no adding_bypass_node)算法。

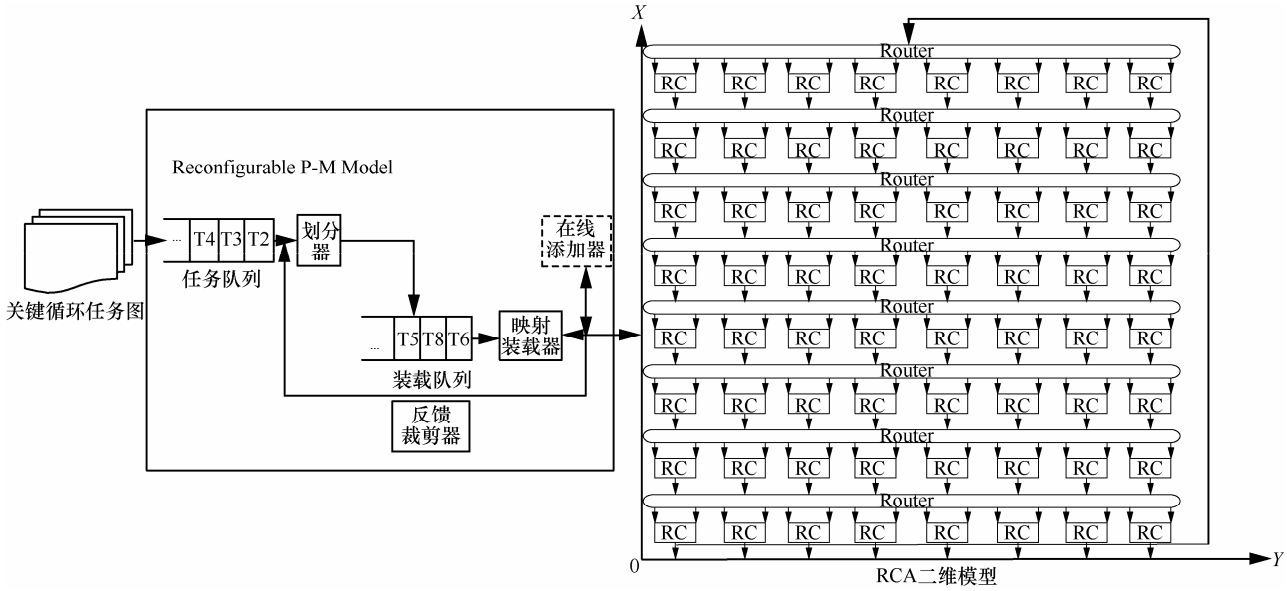



图 4 CGRA 任务编译器划分映射流水化模型

PTBA 的算法流程描述如下。

输入：一个 DFG；

输出：配置信息；DFG 所有节点映射到 RCA 的坐标及其相应的 $M$、$C_{CON}$、$N_1$、$N_2$、$S_{SD}$、$I_{ID}$、$P_{POWER}$、$T_{TOTAL}$；

约束：可重构单元面积 $A_{RPU}$=$Row$×$Col$；DFG 所有划分块之间均不产生非法依赖关系；每行(列)运算节点的个数小于等于 $Row$($Col$)；$I_{ID}$ 近似为 0；映射成功后的 RCA，每行节点均可并行执行。

映射方法：MGC。

适合架构：2-D REMUS 及其通过总线、路由、行列全互连的 CGRA 等架构。

**step1**　初始化，读入数据表，初始化变量；

**step2**　访问根节点，扫描数据表，获取当前划分子图的根节点作为起始点；

**step3**　访问左子树

(a) 左子树根节点，如果该点前驱没有划分，转 step3(b)，否则放入该点，转 step3(c)；

(b) 左子树前驱：

If（左子树前驱与根节点的层次之差等于 1&&有足够的硬件空间）

{满足约束，划入所有的合法点，回溯到左子树根节点；}

Else if(左子树前驱与根节点的层次之差大于 1&&有足够的硬件空间)/*PTBNA 此步骤省略*/

{调用 addnode_CLIT()，实现 CLIT 映射，如不满足约束，回溯到左子树根节点；}

(c) 左子树后继：

If（左子树后继与根节点的层次之差等于 1&&有足够的硬件空间）

{满足约束，划入所有的合法点，回溯到左子树根节点；}

Else if(左子树后继与根节点的层次之差大于 1&&有足够的硬件空间)/*PTBNA 此步骤省略*/

{调用 addnode_CLOT()，实现 CLOT 映射，如不满足约束，回溯到左子树根节点；}

**step 4**　访问右子树

(a) 右子树根节点，如果该点前驱没有划分，转 step4(b)，否则放入该点，转 step4(c)；

(b) 右子树前驱，方法同 step3(b)；

(c) 右子树后继，方法同 step3(c)；

**step5**　若当前 RCA 还有可以放入的点，贪婪放入，否则切换到下一块 RCA，转 step2，继续；

若一个 DFG 的全部映射完毕，转 step6；

**step6**　END PTBA。

采用以下 2 个策略并遵循了 2 个准则来进行 PTBA 任务映射。

**策略 1**　CLOT 和 CLIT 添加 BN 前的预处理，过滤 RCA 添加前的块间 RBN。

以 $A_{RPU}$=16(即 $RCA_{4\times4}$)为例，如图 5 所示，$v_0$、$v_1$ 等相关节点已经映射完毕，$v_0$ 点为块入口点，深色点表示合理 BN，浅色点表示 RBN，现在来考察节点 $v_2$(为了便于说明问题，以三目条件运算符?:为例)，其预处理实现方案如下所述。

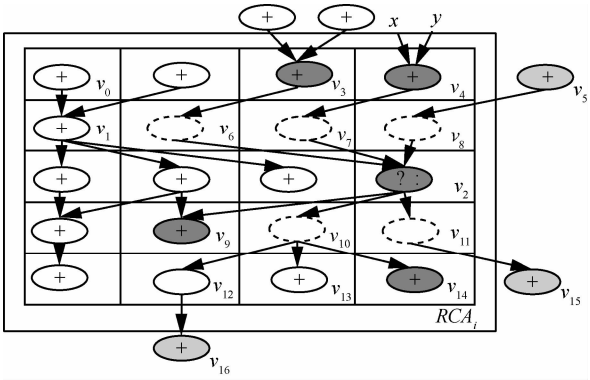



图 5　解释预处理去除 RCA 块间 RBN 的例子

1) 方案 1：$v_2$ 可放入当前 RCA 的前驱节点集的预处理(即 CLIT)。

可分为 3 种情况：a) $v_2$ 直接前驱不跨层且可放，直接映射；若跨层且可放，可分为 2 种情况：一是 $v_2$ 前驱 $v_3$ 的前驱已经划入其他块；二是 $v_2$ 的前驱 $v_4$ 无前驱。预处理方法是在就绪调度队列保留 $v_3$，$v_4$；b) $v_2$ 前驱 $v_5$ 已经划入其他块，预处理方法是在就绪调度队列过滤 $v_5$，避免产生块间 RBN 点 $v_8$。c)若 $v_2$ 直接前驱的前驱存在跨层且可映射当前块，通过递归实现上述 a)和 b)。

2) 方案 2：$v_2$ 可放入当前 RCA 的后继节点集的预处理(即 CLOT)

可分为 3 种情况：a) $v_2$ 直接后继 $v_9$ 不跨层且可放，直接映射；若跨层且可放，预处理方法是在就绪调度队列保留 $v_{13}$，$v_{14}$；b) $v_2$ 到其后继 $v_{15}$、$v_{16}$ 也存在跨层：可分为 2 种情况，一是 $v_2$ 后继 $v_{16}$ 映射不到当前块(行数不满足)，预处理方法是在就绪调度队列过滤 $v_{16}$，避免产生块间 RBN 点 $v_{12}$；二是 $v_2$ 后继 $v_{15}$ 映射不到当前块(列数不满足)，预处理方法是在就绪调度队列过滤 $v_{15}$，避免产生块间 RBN 点 $v_{11}$；c)若 $v_2$ 直接后继的后继存在跨层且可映射当前

块，通过递归实现上述 a)和 b)。

**策略 2** 在策略 1 的基础上，分别实现 CLOT 和 CLIT 块内无 RBN 添加映射，具体包括如下 2 种情形。

**情形 1** CLOT 直接后继索引下标 *index* 穿通映射

设 $v_0$ 有多个后继(如图 6(a)所示)，若采用直接加点映射会出现 RBN 点，由图 6(b)可知 RBN=3(即 $v_4$、$v_5$、$v_6$)，但采用 CLOT 直接后继索引下标 *index* 穿通映射可使 RBN=0，如图 6(e)所示，具体叙述如下。

1) 当前点后继索引下标的建立：在 RCA 动态映射过程中，当前点的后继下标编号乱序现象常有发生(如图 6(a)中 $v_0$ 的后继 $v_2$、$v_1$、$v_3$ 的下标)，首先应处理与 $v_0$ 层次差最小的点 $v_2$，但是无法通过下标来动态更新 BN 的后继个数，所以应建立后继下标索引 *index* 数组。

2) 通过索引 *index* 动态更新 BN 入出度及前驱和后继逻辑关系：

第一次从 PTBA 算法接口传入的参数是可以映射到当前块的 CLOT 树的一对跨层父子对(如当前点有多个后继，则选择当前点与其直接后继层差大于 1 且最小的点作为穿通顶点对(如图 6(a)中的<$v_0$,$v_2$>)，扫描数据表求解获得当前点与其后继层次差等于 1 的点的索引数组下标值(这样的点如有多个，取最大下标值)，若找到，返回 *index* 值，否则 *index*=0；然后进行跨层父子对的 BN 动态添加，同时根据当前父节点的出度和 *index* 的值计算新加 BN 的出度。如图 6 (a)所示，因为 $v_0$ 的后继层次差等于 1 的节点不存在，故 *index*=0，新加 BN1 的出度为 outdegree(BN1)=outdegree($v_0$)-*index*=3−0=3，因新加点的前驱是当前父节点，故 BN1 的入度更新为 indegree(BN1)=1，BN1 前驱更新为父节点 $v_0$，BN1 后继变为 $v_2$、$v_1$、$v_3$，依次类推，后面添加及结果如图 6(d)和图 6(e)所示。特别地，若当前根节点只有一后继，则非线性结构 CLOT 就退化为线性结构。

**情形 2** CLIT 穿通映射

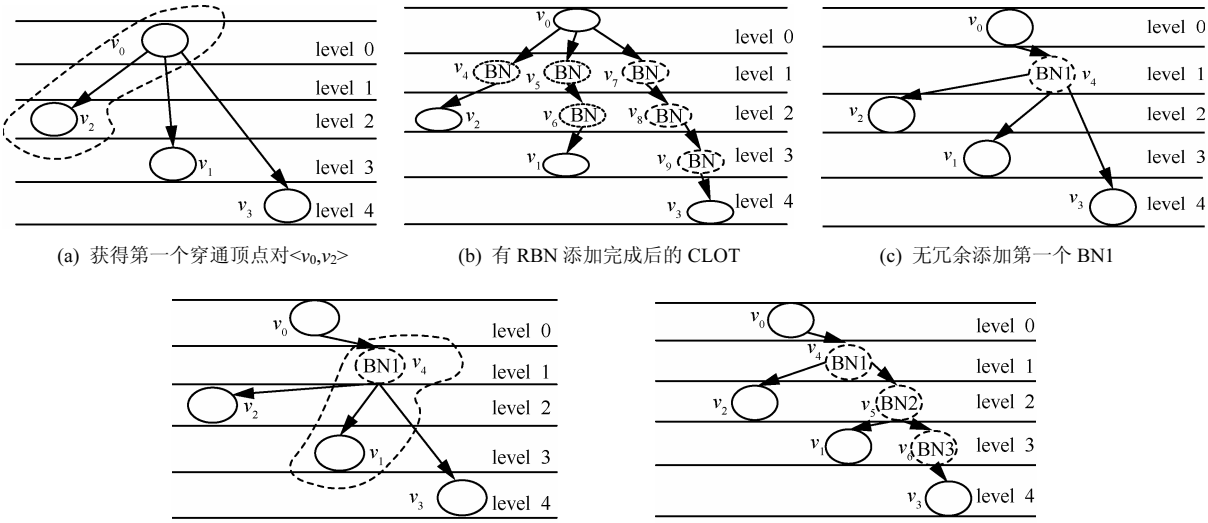
对于图 7(a)，首先从 PTBA 算法接口传入是可映射到当前块的 CLIT 树的一对跨层子父对(如当前子节点有多个前驱，则选择当前子节点与其前驱层差大于 1 且层差最大的前驱点作为线性穿通的入口点，然后从该点开始从上向下依次添加若干个 BN，直到 BN 与子节点的层次差等于 1 为止(如图 7(b)所示)。若该子节点还有一个前驱(双目运算)，则再从子节点开始从下向上依次添加 BN(如图 7(c)所示)。若该子节点有 2 个前驱(三目运算)，则先选择层差大于 1 且较小的跨层子父对，且从子节点开始从下向上依次添加 BN，直到新加 BN 与父节点的层次差为 1 时再回溯到子节点，然后再找层差大于 1 且较大的跨层前驱添加，具体如图 7(d)所示。

PTBA 在添加 BN 时的 2 个准则。

**准则 1** 添加BN后的DFG运算关系应保持不变。

**准则 2** 当前入口点与其直接跨层后继应保持有序。

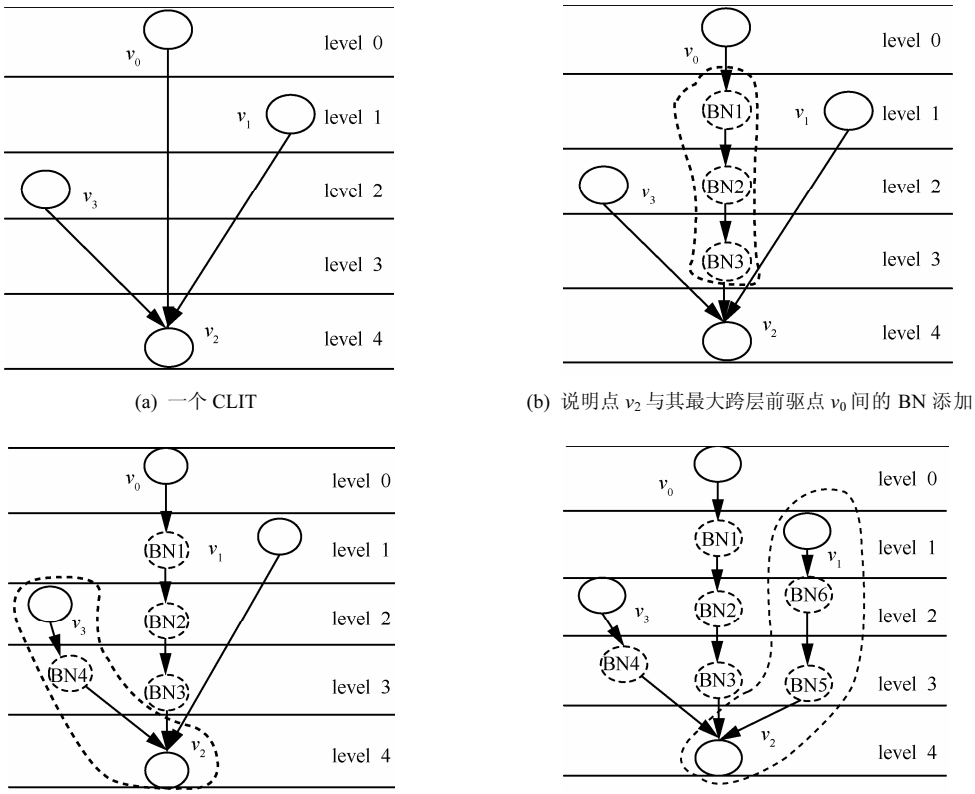
策略 2 对当前入口点的后继建立索引，为了方




(a) 获得第一个穿通顶点对<$v_0$,$v_2$>　(b) 有 RBN 添加完成后的 CLOT　(c) 无冗余添加第一个 BN1

(d) 获得第二个穿通顶点对<$v_4$,$v_1$>　(e) 无 RBN 添加完成后的 CLOT

图 6 解释 CLDT 直接后继索引非线性穿通的例子




(a) 一个 CLIT

(b) 说明点 $v_2$ 与其最大跨层前驱点 $v_0$ 间的 BN 添加

(c) 说明点 $v_2$ 与其最小跨层前驱点 $v_3$ 间的 BN 添加

(d) 说明点 $v_2$ 与其较大跨层前驱点 $v_1$ 间的 BN 添加

图 7　解释 CLIT 线性穿通的例子

便 CLOT 和 CLIT 动态添加 BN，还应对具有索引下标的节点按层次从小到大进行排序，保证当前入口点与其直接跨层后继有序，目的是便于添加 BN。

遵循上述策略和准则的 PTBA 调用函数 *addnode_CLOT*( )和 *addnode_CLIT*( )的设计思路分述如下。

1) *addnode_CLOT*( )的实现

输入：一个 CLOT 树

输出：加了 BN 的 CLOT 树被成功映射到当前块及其配置信息

约束条件：RBN=0；加了 BN 后 CLOT 保持原运算逻辑关系不变；映射完成后的 CLOT 树中无跨层顶点对。

**step1**　变量初始化；建立节点 $v_i$ 后继节点的下标索引；扫描数据表，获取 CLOT 父节点和层次差大于 1 且层次最小的后继；扫描父节点的后继，动态获取当前父节点与其后期层次等于 1 的最大下标值 *index*，若找到 return（*index*），否则 return（0）。

**step2**　动态申请内存空间，并设置新加 BN 的大小等属性；采用当前点直接后继索引下标 *index* 非线性穿通法进行无冗余 BN 的添加映射；每加一个 BN 节点总数 nodenumber++；每添加一个 BN 均及时更新其前驱和后继。

若 CLOT 树通过添加 BN 被全部映射后，转 step3。

**step3**　End *addnode_CLOT*()。

*addnode_CLOT*( )的平均时间复杂度为 $O(suc_1 \times suc_2 \times pre_1)$。其中，$suc_1$ 为新加 BN 的后继数，$suc_2$ 为 CLOT 父节点跨层后继数，$pre_1$ 表示 CLOT 父节点跨层前驱数。

2) *addnode_CLIT*( )的实现

输入：一个 CLIT 树

输出：加了 BN 的 CLIT 树被成功映射到当前块及其配置信息

约束条件：RBN=0；加了 BN 后 CLIT 保持原运算逻辑关系不变；映射完成后的 CLIT 树中无跨层顶点对。

**step1**　变量初始化；扫描数据表，获取 CLIT 当前子节点与其直接前驱的层次差大于 1 且层次差是最大的前驱（即当前点的父节点）。

**step2**　动态申请内存空间，并设置新加 BN 的

**表 3** 划分基准集

| 划分基准 | 加法 | 减法 | 乘法 | 取模 | 逻辑比较 | 异或 | 左移 | 右移 | 等于 | 小于 | 小于等于 | 总数 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SODE | 2 | 2 | 6 | – | 1 | – | – | – | – | – | – | 11 |
| FEAL | 6 | – | – | 4 | – | 20 | 4 | – | – | – | – | 34 |
| FFT4 | 4 | 4 | 4 | – | – | – | – | – | – | – | – | 12 |
| FFT8 | 12 | 12 | 12 | – | – | – | – | – | – | – | – | 36 |
| EWF6 | 168 | – | 36 | – | – | – | – | – | – | – | – | 204 |
| FDCT6 | 78 | 78 | 96 | – | – | – | – | – | – | – | – | 252 |
| MEDIAN | – | – | – | – | 19 | – | – | – | – | – | – | 19 |
| MATRIX4 | 48 | – | 64 | – | – | – | – | – | – | – | – | 112 |
| MATRIX8 | 512 | – | 512 | – | – | – | – | – | – | – | – | 1 024 |
| IDCT | 18 | 14 | 11 | – | – | – | 1 | 10 | – | – | – | 54 |
| DCT32 | 222 | 128 | 129 | – | – | – | – | 83 | – | – | – | 562 |
| H.264 | 141 | 181 | 180 | – | – | – | 4 | 27 | 4 | 76 | 51 | 664 |
| HHLFLIC | 172 | 132 | 212 | – | – | – | 36 | 24 | – | – | 20 | 596 |
| HHLFLC | 152 | 212 | 136 | – | – | – | 12 | 60 | – | 52 | 24 | 648 |

大小等属性；从当前点的父节点开始依次从上向下添加若干个 BN，直到 BN 与子节点的层次差等于 1 为止，每加一个 BN 节点总数 nodenumber++；每添加一个 BN 均及时更新其前驱和后继。然后转 step1，寻找层次差最小跨层子父对进行映射；最后寻找层次差较小跨层子父对进行映射。若 CLIT 树通过添加 BN 被全部映射后，转 step3。

**step3** End *addnode_CLIT*()。

*addnode_CLIT*()的平均时间复杂度为 $O(n)$, 其中，$n$ 表示一个 DFG 的节点数。

**定理 1** PTBA 算法在 RCA 块内在线添加节点时具有以下特性：动态更新 CLOT 和 CLIT 运算节点之间的前驱后继逻辑关系，并且不会添加 RBN。

**证明** 对于任一个 CGRA 系统添加 BN 问题可描述为一个通用的参数化函数形式，即 $\mathrm{F}(\{A_{hw}^{x}, B_{hw}^{y}, A_{hw}^{x} \cap B_{hw}^{y}\}, C^{v_i})$，PTBA 主要考虑 $A_{hw}^{x}$、$B_{hw}^{y}$、$A_{hw}^{x} \cap B_{hw}^{y}$ 等 3 种结构。其中，$A_{hw}^{x}$ 表示可放入当前块且仅含 CLOT 树 $x$；$B_{hw}^{y}$ 表示可放入当前块且仅含 CLIT 树 $y$；$A_{hw}^{x} \cap B_{hw}^{y}$ 表示 可放入当前块且含有公共边的 CLOT 和 CLIT 混合树；$C^{v_i}$ 表示 $v_i$ 是添加的某个 BN 点，对于 $A_{hw}^{x}$、$B_{hw}^{y}$、$A_{hw}^{x} \cap B_{hw}^{y}$ 而言，若满足最小化 $C^{v_i}$，则表示 F 函数取得了最优解，即表明添加的 BN 是无冗余的。

1) $A_{hw}^{x}$ 结构：设 CLOT 树的节点个数为 $q$，其全部节点构成集合$\{v_0, v_1, \cdots, v_{q-1}\}$。

**step1** 有序运算符的空间 $\Omega$ 赋初值为$\{v_0, v_1, \cdots, v_{q-1}\}$。

**step2** 记 $v_\alpha$ 为 $\Omega$ 中层最小点，即 $\mathrm{level}(v_\alpha)=\min\limits_{0\leqslant i\leqslant q-1}\{\mathrm{level}(v_i)\}$，$S$ 为与 $v_\alpha$ 的层差为 1 的点的集合即 $S=\{v\in\Omega \mid \mathrm{level}(v)-\mathrm{level}(v_\alpha)=1\}$，$v_\gamma$ 为与 $v_\alpha$ 的层差最小且大于 1 的任一点即 $\mathrm{level}(v_\gamma)-\mathrm{level}(v_\alpha)=\min\limits_{v\in\Omega-\{v_\alpha\}}\{\mathrm{level}(v)-\mathrm{level}(v_\alpha)\}$，且 $\mathrm{level}(v_\gamma)-\mathrm{level}(v_\alpha)>1$。*index* 为 $v_\alpha$ 后继列表数组下标最大值，即为$|S|$。若 $S=\Phi$，则转 step3，否则在 $v_\alpha$ 的后继添加节点 BN，$v_\alpha$ 的前驱不变，后继列表更新为 $S\cup\{\mathrm{BN}\}$，后继个数为 *index*+1，BN 的前驱为 $v_\alpha$，后继列表更新为 $\Omega-\{v_\alpha\}-S$，后继个数为 $|\Omega|-1-index$；

**step3** $\Omega\leftarrow\Omega-\{v_\alpha\}-S$;

**step4** 若 $\Omega\neq\Phi$，则转 step2，否则结束。由上，对于 CLOT 而言，没有添加 RBN。

2) $B_{hw}^{y}$ 结构：设 CLIT 树的节点个数为 $p$，有序运算符的空间 $\Omega$ 赋初值为$\{v_0, v_1, \cdots, v_{p-1}\}$。$v_\alpha$ 的含义同 1), $v_\beta$ 为 $\Omega$ 中层最大点。

**step1** 用上述的线性结构 CLOT 添加 BN 的算法在 $v_\alpha$ 和 $v_\beta$ 之间从上到下添加 $h$ 个 BN，不妨记集合为 $H=\{v_p, \cdots, v_{p+h-1}\}$，$h=\mathrm{level}(v_\beta)-\mathrm{level}(v_\alpha)-1$，直到 BN 与 $v_\beta$ 层差等于 1 为止；

**step2**　$V_\delta=\min_{v\in\Omega-\{v_\beta,v_\alpha\}}\{\text{level}(v)\}$，若 $\text{level}(v_\beta)-\text{level}(v_\delta)=1$，则结束；否则，在 $v_\delta$ 与 $v_\beta$ 间添加 $r$ 个点，记集合为 $R=\{v_{p+h}, v_{p+h+1}, \cdots, v_{p+h+r-1}\}$，其中 $r=\text{level}(v_\beta)-\text{level}(v_\delta)-1$，添加方法在 $\text{level}(v_\beta)-1$ 层上添加 $v_{p+h}$；$\text{level}(v_\beta)-2$ 层上添加 $v_{p+h+1}$,…,$\text{level}(v_\beta)$-$r$ 层上添加 $v_{p+h+r-1}$。相邻两点连一条边得到长度为 $q$ 的路径：$v_{p+h}\rightarrow\cdots\rightarrow v_{p+h+r-1}$，每添加一个点动态更新该点的前驱和后继关系，其中，$v_{p+h}$ 的后继为 $v_\beta$，$v_{p+h+r-1}$ 的前驱为 $v_\delta$。

**step3**　$\Omega\leftarrow(\Omega\cup H\cup R)-(\{v_\delta\}\cup\{v_\alpha\}\cup H\cup R)$，即 $\Omega\leftarrow\Omega-(v_\delta\cup v_\alpha)$；

**step4**　若 $\Omega\neq\Phi$，则转 step2，否则结束。显然，对于 CLIT 而言，没有添加 RBN。

3) $A_{hw}^x\cap B_{hw}^y$ 结构：它包含前面 2 种情况，因为 $A_{hw}^x$ 容易产生冗余点，故添加顺序是先 $A_{hw}^x$ 后 $B_{hw}^y$，因此，这样做可避免在公共边上多添加 RBN，结合 1)和 2)可知 $A_{hw}^x\cap B_{hw}^y$ 没有添加 RBN，且可动态更新节点间的前驱后继关系。 综上，命题得证。

## 5　实验及其分析

### 5.1　划分基准程序设计

除了采用的 2.2 节中的 8 个基准程序外，还增加了 MATRIX8、IDCT、DCT32、H.264- deblocking (为了简洁，用 H.264 来表示) 、h264_h_loop_ filter_luma_intra_c (HHLFLIC)、h264_h_loop_filter_ luma_c(HHLFLC)6 个程序，各基准程序的操作单元的数量如表 3 所示。

### 5.2　实验结果分析及临界条件

#### 5.2.1　PTBA 增大 $T_{TOTAL}$ 和 $P_{POWER}$ 的实验结果

采用 C 语言实现了 PTBA 和 PTBNA 这 2 种算法。文中运算的执行时延乘法为 2 cycle，取模(或除法)为 4 cycle，其他 ALU 为 1 cycle，BN 点为 0 cycle。基于 REMUS 架构(点到点的映射，$I_{ID}$ 近似为 0，所以本文后续讨论基于 REMUS 架构映射时均没有考虑 $I_{ID}$)，为了对 PTBA 和 PTBNA 算法进行较为全面的比较，综合考虑了 $M$、$C_{CON}$、$N_1$、$N_2$、$S_{SD}$、$T_{TOTAL}$ 等指标，在 3.2 节研究的基础上，得出执行总时间计算方法为

$T_{TOTAL}=\alpha$[数据输入时间(即 $N_1+N_{org1}$)+

数据输出时间(即 $N_2+N_{org2}$)]+ $S_{SD}+C_{CON}$　(1)

其中，$\alpha$ 为修正系数，它由不同可重构系统所决定，通过对 REMUS 编译器的实际测试，得出 $\alpha$= 0.5；一个配置字所需的时间为 1cycle，则 $C_{CON}=M\times N_{con}+$运算 RC 数(即待映射 DFG 的节点数)+路由 RC 数(即 BN 数，若不加 BN 则为 0)，$N_{con}$ 表示配置硬件功能和连接关系的控制寄存器数(REMUS 为 17)。

一般而言，CGRA 功耗可大致包括运算 RC、路由 RC(即 BN)、空载 RC、配置字消耗的功耗、剩余硬件整体功耗(一般为一个常数)等，因文献[11]考虑的硬件功耗较为全面，故本文采用了该文的功耗估计值，即一个运算 RC 的功耗 $P_{computing\_RC}$ =2.542 93 mW，一个配置字的功耗 $P_{context}$ =2.721 675 mW，一个空载 RC 功耗 $P_{frag\_RC}$ =0.254 293 mW，一个路由 RC(即 BN)的功耗 $P_{router}$=0.847 321 mW，剩余硬件整体功耗为一常数，其值 $P_{hardware}$ = 64.970 43 mW。这样执行总功耗计算方法为

$$P_{POWER}=P_{\text{运算}RC}+P_{\text{路由}RC}+P_{\text{空载}RC}+P_{\text{配置}}+P_{\text{剩余硬件}}\quad(2)$$

其中，$P_{\text{运算}RC}=nP_{computing\_RC}$；$P_{\text{路由}RC}=$ BN 数$\times P_{router}$；$P_{\text{空载}RC}=$空载 RC 的个数$\times P_{frag_RC}$；$P_{\text{配置}}=P_{context}\times(N_{con}M+$运算 RC 数+路由 RC 数)。$A_{RPU}$ 随机选取 25($RCA_{5\times5}$)和 64($RCA_{8\times8}$)2 个值。对于不同的 $A_{RPU}$，具体的 $P_{POWER}$、$M$、$N_1$、$N_2$、$S_{SD}$、$C_{CON}$、$T_{TOTAL}$ 的比较结果如表 4~表 6 所示，表 5 和表 6 分别给出了 $A_{RPU}$=25、64 时采用 PTBNA 和 PTBA 得到的划分映射指标体系的结果，以表 5 的 $M$ 列 EWF6 为例，当 $A_{RPU}$=25 时，采用 PTBNA 算法所获得的 $M$=9 而采用 PTBA 算法所获得的 $M$=10，Δ%=+11.1%(正值表示没有改进)。

#### 5.2.2　加和不加 BN 映射的实验结果分析

本文研究以 REMUS 为例($RCA_{4\times4}$)，其 RCA 互连模式及每个 RC 内部结构如文献[17,18]所述。由表 4~表 6 可知，发现部分基准程序加 BN 映射后其获得的 $P_{POWER}$ 和 $T_{TOTAL}$ 反而增大了很多，不加 BN 映射反而效果好。经过深入分析，发现加 BN 映射可以减少 $N_1$ 和 $N_2$，加少量 BN 映射可以减少 $M$，但是如果增加了较多的 BN，使 $M$ 得不到减少，甚至增大，这样会带来 $C_{CON}$、$S_{SD}$、$P_{POWER}$、$T_{TOTAL}$ 等的增大。

**例 2**　下面举一例说明上述情况，图 8(a)为一划分中的循环 DFG 子图，原始输入为 24，原始输出为 3，有 21 个加法运算，2 种映射方法结果如图 8(b)和图 8(c)所示。表 7 给出了 PTBA 和 PTNBA 映射结果比较，PTBA 在映射时加了 6 个 BN，$N_1$，$N_2$ 减少了，$C_{CON}$、$S_{SD}$、$P_{POWER}$ 等指标增大了，$T_{TOTAL}$ 增大了，为了解决这个问题，必须给出 PTBA 映射平衡的临界条件。

**表 4** **5.2 节实验 $P_{POWER}$ 比较**

| 划分基准 | $RCA_{5\times5}$ | | | $RCA_{8\times8}$ | | |
|---|---|---|---|---|---|---|
| | PTBNA | PTBA | △% | PTBNA | PTBA | △% |
| FEAL | 523.139 282 | 529.768 677 | +1.3 | 425.377 899 | 465.154 358 | +9.4 |
| EWF6 | 2 080.469 727 | 2 297.506 836 | +10.4 | 1 659.671 875 | 1 871.812 866 | +12.8 |
| FDCT6 | 2 556.156 982 | 2 622.451 172 | +2.6 | 1 900.166 870 | 1 933.313 843 | +1.7 |
| MATRIX4 | 1 149.136 108 | 1 333.026 367 | +16.0 | 943.695 923 | 1 033.192 871 | +9.5 |
| MATRIX8 | 9 952.004 883 | 11 321.160 156 | +13.8 | 7 170.777 832 | 9 722.280 273 | +35.6 |
| HHLFLIC | 5 808.455 078 | 6 784.256 348 | +16.8 | 4 388.795 898 | 4 897.500 488 | +11.6 |
| HHLFLC | 6 539.376 465 | 7 253.467 773 | +10.9 | 5 031.873 047 | 5 575.484 375 | +10.8 |
| 平均△% | – | – | +10.3 | – | – | +13.1 |

**表 5** **5.2 节实验 $M$、$N_1$、$N_2$ 比较**

| 划分基准 | $M$ | | | | | | $N_1$ | | | | | | $N_2$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | |
| FEAL | 3 | 2 | 3 | 2 | – | – | 13 | 10 | 12 | 4 | −7.7 | −60 | 12 | 10 | 11 | 4 | −8.3 | −60 |
| EWF6 | 9 | 5 | 10 | 5 | +11.1 | – | 123 | 102 | 117 | 83 | −4.9 | −18.6 | 101 | 93 | 95 | 67 | −5.9 | −28 |
| FDCT6 | 11 | 5 | 11 | 5 | – | – | 102 | 70 | 101 | 64 | −1.0 | −8.6 | 98 | 68 | 97 | 62 | −1.0 | −8.8 |
| MATRIX4 | 5 | 3 | 6 | 3 | +20 | – | 53 | 52 | 53 | 40 | – | −23.1 | 53 | 52 | 53 | 40 | – | −23.1 |
| MATRIX8 | 41 | 16 | 48 | 24 | +17.1 | +50 | 596 | 567 | 502 | 419 | −15.8 | −26.1 | 596 | 567 | 502 | 419 | −15.8 | −26.1 |
| HHLFLIC | 24 | 11 | 29 | 12 | +20.8 | +9.1 | 348 | 302 | 264 | 213 | −24.1 | −29.5 | 326 | 296 | 241 | 206 | −26.1 | −30.4 |
| HHLFLC | 28 | 14 | 31 | 14 | +10.7 | – | 359 | 334 | 289 | 210 | −19.5 | −37.1 | 353 | 333 | 284 | 203 | −19.5 | −39.0 |
| 平均△% | – | – | – | – | +11.4 | +8.4 | – | – | – | – | −10.4 | −29.0 | – | – | – | – | −10.9 | −30.8 |

**表 6** **5.2 节实验 $S_{SD}$、$C_{CON}$、$T_{TOTAL}$ 比较**

| 划分基准 | $S_{SD}$ | | | | | | $C_{CON}$ | | | | | | $T_{TOTAL}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | |
| FEAL | 25 | 24 | 25 | 21 | – | −12.5 | 85 | 68 | 87 | 80 | +2.4 | +17.6 | 140 | 120 | 141 | 123 | +0.7 | +2.5 |
| EWF6 | 63 | 38 | 71 | 56 | +12.7 | +47.4 | 357 | 289 | 404 | 353 | +13.2 | +22.1 | 607 | 500 | 656 | 559 | +8.1 | +11.8 |
| FDCT6 | 81 | 57 | 85 | 57 | +4.9 | – | 439 | 337 | 459 | 347 | +4.6 | +3 | 740 | 583 | 763 | 587 | +3.1 | +0.7 |
| MATRIX4 | 39 | 27 | 45 | 32 | +15.4 | +18.5 | 197 | 163 | 234 | 190 | +18.8 | +16.6 | 361 | 314 | 404 | 334 | +11.9 | +6.4 |
| MATRIX8 | 318 | 199 | 399 | 322 | +25.5 | +61.8 | 1 721 | 1 296 | 2 002 | 1 894 | +16.3 | +46.1 | 3 179 | 2 606 | 3 447 | 3 179 | +8.4 | +22.0 |
| HHLFLIC | 205 | 131 | 234 | 157 | +14.1 | +19.8 | 1 004 | 783 | 1 206 | 915 | +20.1 | +16.9 | 1 807 | 1 474 | 1 953 | 1 542 | +8.1 | +4.6 |
| HHLFLC | 198 | 139 | 235 | 168 | +18.7 | +20.9 | 1 124 | 886 | 1 284 | 1 050 | +14.2 | +18.5 | 1 972 | 1 653 | 2 100 | 1 719 | +6.5 | +4.0 |
| 平均△% | – | – | – | – | +13.0 | +22.3 | – | – | – | – | +12.8 | +20.1 | – | – | – | – | +6.7 | +7.4 |

**表 7** **图 8 的 PTBA 和 PTNBA 映射比较**

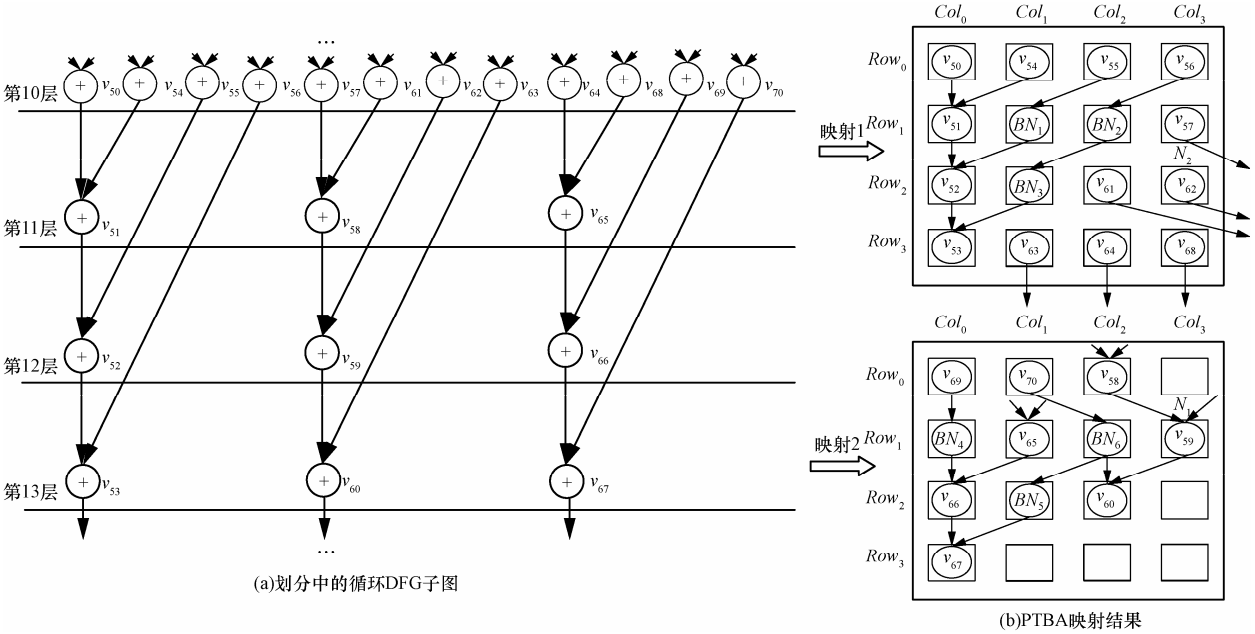
| 算法 | $M$ | $N_1$ | $N_2$ | $S_{SD}$ | $C_{CON}$ | $T_{TOTAL}$ | $P_{POWER}$ |
|---|---|---|---|---|---|---|---|
| PTNBA | 2 | 9 | 9 | 6 | 21 | 49.5 | 335.831 738 |
| PTBA | 2 | 6 | 6 | 8 | 27 | 54.5 | 355.719 956 |

### 5.3 临界条件

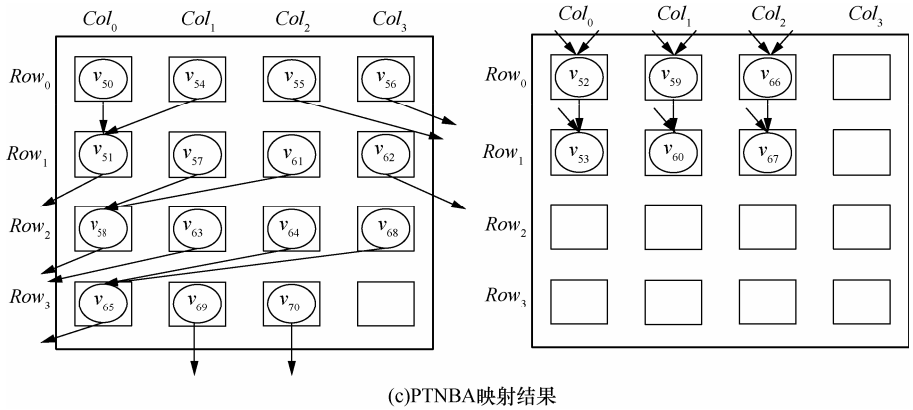
下面给出了 PTBA 的临界条件，具体见准则 3 和准则 4。

**准则 3** $T_{TOTAL}$ 临界条件

设 $M_1(M_2)$、$N_{11}(N_{12})$、$N_{21}(N_{22})$、$S_{SD1}(S_{SD2})$为 PTBA (PTBNA)所用的块数、非原始输入次数、非原始输出次数、计算延迟；PTBA 配置时间 $C_{CON1}=M_1N_{con}+n+\sum_{k=1}^{M_1}BN(k)$，PTBNA 配置时间 $C_{CON2}=M_2N_{con}+n$；$BN(k)$表示第 $k$ 块 RCA 添加的 $BN$ 数，$n$ 表示一 DFG 的节点数。PTBA 和 PTBNA 所花费总时间分别为




(a)划分中的循环DFG子图　　(b)PTBA映射结果




(c)PTNBA映射结果

图 8　PTBA 和 PTNBA 映射说明

$$T_{TOTAL1}=\alpha[\ (N_{11}+N_{org1}\ )+\ (\ N_{21}+N_{org2}\ )]+\ S_{SD1}+C_{CON1}$$
$$=\alpha N_{11}+\alpha N_{org1}\ +\alpha N_{21}+\alpha N_{org2}+\ S_{SD1}+\ M_1N_{con}+n+\sum_{k=1}^{M_1}BN(k) \tag{3}$$

$$T_{TOTAL2}=\alpha[\ (N_{12}+N_{org1})\ +\ (N_{22}+N_{org2})]+\ S_{SD2}+\ C_{CON2}$$
$$=\alpha N_{12}+\alpha N_{org1}+\alpha N_{22}+\alpha N_{org2}+\ S_{SD2}+\ M_2N_{con}+n \tag{4}$$

临界条件为 $T_{TOTAL1}\leqslant T_{TOTAL2}$，即

$$\alpha[N_{11}+N_{21}]+\ S_{SD1}+\ M_1N_{con}+\sum_{k=1}^{M_1}BN(k)$$
$$\leqslant\alpha[N_{12}+N_{22}]+\ S_{SD2}\ +M_2\ N_{con} \tag{5}$$

由表 5 和表 6 可知，若 $M_1>M_2$，如 BN 添加过多，不能保证 $T_{TOTAL1}\leqslant T_{TOTAL2}$，故 $T_{TOTAL}$ 临界条件可细分为 $M_1=M_2$ 和 $M_1<M_2$ 的 2 种情形。

1) 情形 1 ($M_1=M_2$)

$$\sum_{k=1}^{M_1}BN(k)\leqslant deta_1+deta_2 \tag{6}$$

2) 情形 2 ($M_1<M_2$)

$$\sum_{k=1}^{M_1}BN(k)\leqslant deta_1+deta_2+deta_3 \tag{7}$$

其中，$deta_1=\alpha|[(N_{12}+N_{22})-(\ N_{11}+N_{21})]\ |$，$deta_2=|S_{SD1}-S_{SD2}|$，$deta_3=N_{con}(M_2-M_1)$。

**准则 4**　$P_{POWER}$ 临界条件

$$P_{POWER1}=P_{运算RC}+\ P_{路由RC}+\ P_{空载RC1}+\ P_{配置1}+P_{剩余硬件1}$$
$$=nP_{computing\_RC}+\sum_{k=1}^{M_1}BN(k)\ P_{router}+(M_1A_{RPU}-\sum_{k=1}^{M_1}BN(k)-n)\ P_{frag\_RC}+\ (M_1N_{con}+n+$$

$$\sum_{k=1}^{M_1} BN(k))P_{context} + M_1P_{hardware} \quad (8)$$

$$P_{POWER2} = P_{运算RC} + P_{空载RC2} + P_{配置2} + P_{剩余硬件2}$$
$$= nP_{computing\_RC} + (M_2A_{RPU} - n)P_{frag\_RC} + (M_2N_{con} + n)P_{context} + M_2P_{hardware} \quad (9)$$

其中，$P_{POWER1}(P_{POWER2})$为 PTBA(PTBNA)的功耗，若 $P_{POWER1} > P_{POWER2}$，则加 BN 映射是不合理的。故功耗临界条件为 $P_{POWER1} \leqslant P_{POWER2}$，化简后可得

$$\sum_{k=1}^{M_1} BN(k)\,P_{router} + (M_1A_{RPU} - \sum_{k=1}^{M_1} BN(k))P_{frag\_RC} + (M_1N_{con} + \sum_{k=1}^{M_1} BN(k))P_{context} + M_1P_{hardware}$$
$$\leqslant (M_2A_{RPU})P_{frag\_RC} + (M_2N_{con})P_{context} + M_2P_{hardware} \quad (10)$$

若 $M_1 > M_2$，则配置等功耗全部增加，很难获得平衡，不给予考虑，所以把 $P_{POWER}$ 临界条件细分为 $M_1=M_2$ 和 $M_1<M_2$ 的 2 种情形。

1) 情形 1($M_1=M_2$)

$$P_{router} - P_{frag\_RC} + P_{context} \leqslant 0 \quad (11)$$

2) 情形 2 ($M_1<M_2$)

$$\sum_{k=1}^{M_1} BN(k)\ \ P_{router} \leqslant deta_4 + deta_5 + \sum_{k=1}^{M_1} BN(k)\,P_{context} + deta_6 \quad (12)$$

记 $deta_4 = P_{frag\_RC}\ [(M_2 - M_1)A_{RPU} - \sum_{k=1}^{M_1} BN(k))]$，$deta_5 = P_{context}[(M_2 - M_1)N_{con})]$，$deta_6 = P_{hardware}(M_1 - M_2)$。

### 5.4 PTBA 满足准则 3 和准则 4 的实验结果

在满足准则 3 和准则 4 临界条件下，对其进行了验证。由实验结果可知，PTBA 相比于 PTBNA，就 $N_1$、$N_2$、$M$、$C_{CON}$、$P_{POWER}$、$T_{TOTAL}$ 等指标而言，PTBA 获得了较为全面的优化，表 8~表 10 具体给出了 $P_{POWER}$、$M$、$N_1$、$N_2$、$S_{SD}$、$C_{CON}$、$T_{TOTAL}$ 等指标的比较结果。结论是加 BN 映射只有在满足上述临界条件下，PTBA 才能获得好效果。

### 5.5 与 SPKM 映射算法的比较

本文选取了国际上先进的 SPKM 方法进行了实验比较，SPKM 面向 CGRA 在硬件资源的约束下，采用添加 BN 的方法来解决 RCA 块内跨层或块内数据过渡等问题，这与本文研究的问题基本一致，

**表 8　　5.4 节实验 $P_{POWER}$ 比较**

| 划分基准 | $RCA_{5\times5}$ | | | $RCA_{8\times8}$ | | |
|---|---|---|---|---|---|---|
| | PTBNA | PTBA | △% | PTBNA | PTBA | △% |
| SODE | 290.305 878 | 176.024 368 | −39.4 | 310.130 747 | 185.941 788 | −40.0 |
| FFT4 | 295.316 193 | 184.349 380 | −37.6 | 315.151 062 | 194.266 800 | −38.4 |
| FFT8 | 650.756 165 | 556.362 854 | −14.5 | 562.912 170 | 334.402 496 | −40.6 |
| MEDIAN | 565.580 811 | 477.816 925 | −15.5 | 605.250 549 | 389.999 664 | −35.6 |
| IDCT | 740.941 772 | 643.233 765 | −13.2 | 653.097 839 | 538.842 957 | −17.5 |
| DCT32 | 5 755.701 172 | 5 674.566 406 | −1.4 | 4 345.958 984 | 4 341.089 355 | −0.1 |
| H.264 | 7 795.503 418 | 7 684.688 477 | −1.4 | 7 024.743 164 | 5 925.490 723 | −15.6 |
| 平均△% | – | – | −17.6 | – | – | −26.8 |

**表 9　　5.4 节实验 $M$、$N_1$、$N_2$ 比较**

| 划分基准 | $M$ | | | | | | $N_1$ | | | | | | $N_2$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | |
| SODE | 2 | 2 | 1 | 1 | −50 | −50 | 2 | 2 | 0 | 0 | −100 | −100 | 2 | 2 | 0 | 0 | −100 | −100 |
| FFT4 | 2 | 2 | 1 | 1 | −50 | −50 | 4 | 4 | 0 | 0 | −100 | −100 | 4 | 4 | 0 | 0 | −100 | −100 |
| FFT8 | 4 | 3 | 3 | 1 | −25 | −66.7 | 18 | 16 | 10 | 0 | −44.4 | −100 | 18 | 16 | 9 | 0 | −50 | −100 |
| MEDIAN | 4 | 4 | 3 | 2 | −25 | −50 | 12 | 12 | 7 | 2 | −41.7 | −83.3 | 12 | 12 | 7 | 2 | −41.7 | −83.3 |
| IDCT | 4 | 3 | 3 | 2 | −25 | −33.3 | 20 | 15 | 14 | 8 | −30 | −46.7 | 20 | 15 | 14 | 8 | −30 | −46.7 |
| DCT32 | 25 | 12 | 24 | 11 | −4 | −8.3 | 218 | 176 | 209 | 151 | −4.1 | −14.2 | 193 | 163 | 187 | 140 | −3.1 | −14.1 |
| H.264 | 38 | 29 | 35 | 17 | −7.9 | −41.4 | 419 | 377 | 346 | 239 | −17.4 | −36.6 | 367 | 340 | 290 | 211 | −21 | −37.9 |
| 平均△% | – | – | – | – | −26.7 | −42.8 | – | – | – | – | −48.2 | −68.7 | – | – | – | – | −49.4 | −68.9 |

但 PTBA 有以下 3 点与 SPKM 的约束有所不同[11]：

RC(即 PE)互连：在 SPKM 中，假设 RC 可以与其近邻或跨一层水平或垂直的 RC 直接连接，文中 PTBA，RC 间采用左右不可通信，RC 通过行路由与下一层 RC 点到点行流水直接互连。

块间数据存储：在 SPKM 中，假设某行最多有 2 个输入 (即 load)和一个输出(即 store)操作，文中 PTBA 不存在此约束。

优化目标方面：在 SPKM 中，优化目标定义为 CGRA 行数利用的最小化，即$|UR|_{\min}$；本文统一了量化评估的映射指标体系，即 $M$、$C_{CON}$、$N_1$、$N_2$、$S_{SD}$、$I_{ID}$、$P_{POWER}$、$T_{TOTAL}$。

采用了与 5.4 节相同的一组划分基准程序集，表 11~表 14 给出了 SPKM 和 PTBA 算法的比较结果，由比较结果可知，SPKM 采用了 RC 间跨一层和加 BN 过渡块内数据等方法进行映射，导致 RCA

**表 10　　5.4 节实验 $S_{SD}$、$C_{CON}$、$T_{TOTAL}$ 比较**

| 划分基准 | $S_{SD}$ | | | | | | $C_{CON}$ | | | | | | $T_{TOTAL}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | | PTBNA | | PTBA | | △% | |
| SODE | 6 | 6 | 6 | 6 | − | − | 45 | 45 | 29 | 29 | −35.6 | −35.6 | 62 | 62 | 44 | 44 | −29 | −29 |
| FFT4 | 6 | 6 | 6 | 6 | − | − | 46 | 46 | 31 | 31 | −32.6 | −32.6 | 63 | 63 | 44 | 44 | −30.2 | −30.2 |
| FFT8 | 13 | 9 | 14 | 9 | +7.7 | − | 104 | 87 | 94 | 61 | −9.6 | −30 | 149 | 126 | 132 | 84 | −11.4 | −33.3 |
| MEDIAN | 9 | 9 | 12 | 9 | +33.3 | − | 87 | 87 | 79 | 65 | −9.2 | −25.3 | 117 | 117 | 107 | 85 | −8.5 | −27.4 |
| IDCT | 20 | 14 | 18 | 18 | −10 | +28.6 | 122 | 105 | 111 | 92 | −9 | −12.4 | 200 | 172 | 180 | 156 | −10 | −9.3 |
| DCT32 | 175 | 132 | 179 | 133 | +2.3 | +0.8 | 987 | 766 | 981 | 786 | −0.6 | +2.6 | 1 681 | 1 381 | 1 671 | 1 378 | −0.6 | −0.2 |
| H.264 | 270 | 243 | 265 | 200 | −1.9 | −17.7 | 1 310 | 1 157 | 1 332 | 1 083 | +1.7 | −6.4 | 2 150 | 1 936 | 2 092 | 1 685 | −2.7 | −13.0 |
| 平均△% | − | − | − | − | +4.5 | +1.7 | − | − | − | − | -13.6 | -20.0 | − | − | − | − | -13.2 | -20.3 |

**表 11　　5.5 节实验 $P_{POWER}$ 比较**

| 划分基准 | $RCA_{5\times5}$ | | | $RCA_{8\times8}$ | | |
|---|---|---|---|---|---|---|
| | SPKM | PTBA | △% | SPKM | PTBA | △% |
| SODE | 176.024 368 | 176.024 368 | − | 185.941 788 | 185.941 788 | − |
| FFT4 | 301.945 61 | 184.349 38 | −38.9 | 321.780 464 | 194.266 8 | −39.6 |
| FFT8 | 562.991 349 | 556.362 854 | −1.2 | 455.286 764 | 334.402 496 | −26.6 |
| MEDIAN | 501.718 992 | 477.816 925 | −4.8 | 376.740 836 | 389.999 664 | +3.5 |
| IDCT | 881.740 791 | 643.233 765 | −27 | 702.818 364 | 538.842 957 | −23.3 |
| DCT32 | 6 852.261 98 | 5 674.566 406 | −17.2 | 5 419.308 55 | 4 341.089 355 | −19.9 |
| H.264 | 9 753.433 21 | 7 684.688 477 | −21.2 | 7 818.512 181 | 5 925.490 723 | −24.2 |
| 平均△% | − | − | −15.8 | − | − | −18.6 |

**表 12　　5.5 节实验 $I_{ID}$ 比较**

| 划分基准 | $RCA_{5\times5}$ | | | $RCA_{8\times8}$ | | |
|---|---|---|---|---|---|---|
| | SPKM | PTBA | △% | SPKM | PTBA | △% |
| SODE | 3 | 0 | −100 | 3 | 0 | −100 |
| FFT4 | 12 | 0 | −100 | 12 | 0 | −100 |
| FFT8 | 25 | 0 | −100 | 21 | 0 | −100 |
| MEDIAN | 11 | 0 | −100 | 11 | 0 | −100 |
| IDCT | 24 | 0 | −100 | 25 | 0 | −100 |
| DCT32 | 85 | 0 | −100 | 242 | 0 | −100 |
| H.264 | 366 | 0 | −100 | 539 | 0 | −100 |
| 平均△% | − | − | −100 | − | − | −100 |

**表 13** **5.5 节实验 $M$、$N_1$、$N_2$ 比较**

| 划分基准 | $M$ | | | | | | $N_1$ | | | | | | $N_2$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPKM | | PTBA | | △% | | SPKM | | PTBA | | △% | | SPKM | | PTBA | | △% | |
| SODE | 1 | 1 | 1 | 1 | – | – | 0 | 0 | 0 | 0 | – | – | 0 | 0 | 0 | 0 | – | – |
| FFT4 | 2 | 2 | 1 | 1 | −50 | −50 | 4 | 4 | 0 | 0 | −100 | −100 | 4 | 4 | 0 | 0 | −100 | −100 |
| FFT8 | 3 | 2 | 3 | 1 | – | −50 | 11 | 8 | 10 | 0 | −9.1 | −100 | 9 | 7 | 9 | 0 | – | −100 |
| MEDIAN | 3 | 2 | 3 | 2 | – | – | 8 | 8 | 7 | 2 | −12.5 | −75 | 7 | 6 | 7 | 2 | – | −66.7 |
| IDCT | 5 | 3 | 3 | 2 | −40 | −33.3 | 23 | 12 | 14 | 8 | −39.1 | −33.3 | 20 | 14 | 13 | 8 | −35 | −42.9 |
| DCT32 | 33 | 18 | 24 | 11 | −27.3 | −38.9 | 193 | 214 | 209 | 151 | +8.3 | −29.4 | 171 | 176 | 187 | 140 | +9.4 | −20.5 |
| H.264 | 52 | 30 | 35 | 17 | −32.7 | −43.3 | 337 | 288 | 346 | 239 | +2.7 | −17 | 217 | 195 | 290 | 211 | +33.6 | +8.2 |
| 平均△% | – | – | – | – | −21.4 | −30.8 | – | – | – | – | −21.4 | −50.7 | – | – | – | – | −13.1 | −46 |

**表 14** **5.5 节实验 $S_{SD}$、$C_{CON}$、$T_{TOTAL}$ 比较**

| 划分基准 | $S_{SD}$ | | | | | | $C_{CON}$ | | | | | | $T_{TOTAL}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SPKM | | PTBA | | △% | | SPKM | | PTBA | | △% | | SPKM | | PTBA | | △% | |
| SODE | 7 | 7 | 6 | 6 | −14.3 | −14.3 | 29 | 29 | 29 | 29 | – | – | 47.5 | 47.5 | 44 | 44 | −7.4 | −7.4 |
| FFT4 | 9 | 9 | 6 | 6 | −33.3 | −33.3 | 48 | 48 | 31 | 31 | −35.4 | −35.4 | 83 | 83 | 44 | 44 | −47 | −47 |
| FFT8 | 16 | 9 | 14 | 9 | −12.5 | – | 96 | 76 | 94 | 61 | −2.1 | −19.7 | 159 | 125.5 | 132 | 84 | −17 | −33.1 |
| MEDIAN | 24 | 24 | 12 | 9 | −50 | −62.5 | 78 | 61 | 79 | 65 | +1.3 | +6.6 | 129.5 | 112 | 107 | 85 | −17.4 | −24.1 |
| IDCT | 37 | 48 | 18 | 18 | −51.4 | −62.5 | 149 | 120 | 111 | 92 | −25.5 | −23.3 | 269.5 | 243.5 | 180 | 156 | −33.2 | −36 |
| DCT32 | 201 | 168 | 179 | 133 | −10.9 | −20.8 | 1 170 | 961 | 981 | 786 | −16.2 | −18.2 | 1 951 | 1 879 | 1 671 | 1 378 | −14.4 | −26.7 |
| H.264 | 301 | 266 | 265 | 200 | −12 | −24.8 | 1 642 | 1 375 | 1 332 | 1 083 | −18.9 | −21.2 | 2 842.5 | 2 756.5 | 2 092 | 1 685 | −26.4 | −38.9 |
| 平均△% | – | – | – | – | −26.3 | −31.2 | – | – | – | – | −13.8 | −15.9 | – | – | – | – | −23.3 | −30.5 |

块内跨层较多，块内跨层或数据过渡存储转发的 BN 点较多，导致 SPKM 的 $I_{ID}$ 和 $S_{SD}$ 等较大，PTBA 采用是 RC 上下行点的行流水映射，其 $I_{ID}$ 近似为 0，$S_{SD}$ 较少；SPKM 没有进行贪婪映射且加的点较多，RC 碎片较多，从而导致 $M$、$C_{CON}$、$N_1$、$N_2$、$P_{POWER}$、$T_{TOTAL}$ 等指标增大，PTBA 进行了贪婪映射，每次尽可能填满每一块 RCA，所以 $M$、$C_{CON}$、$N_1$、$N_2$、$P_{POWER}$、$T_{TOTAL}$ 等指标相对较少一点。综上所述，PTBA 在满足临界条件等约束下的映射较具优势。

## 6 结束语

本文给出了面向统一 CGRA 样机平台时域划分映射统一评价指标体系及其编译器流水线模型，同时提出了一种追求 $I_{ID}$ 最小化的二维 RCA 旁节点添加算法，采用了提取跨层节点和无冗余添加同时进行的策略，避免了多次重复遍历 DFG。本文算法同时能够自适应地选择跨层节点对进行无冗余节点添加，对于同一流水线 RCA 而言，在满足 $T_{TOTAL}$ 和 $P_{POWER}$ 临界条件下，对含有较多 CLOT 和少量 CLIT 的 DFG，通过加少量 BN 映射可以获得优化的 $M$、$N_1$、$N_2$、$C_{CON}$、$T_{TOTAL}$、$P_{POWER}$ 等指标，通过比较，PTBA 优于国际先进算法 SPKM，从而表明文中提出方案合理性。

## 参考文献：


[1] CARDOSO J M P, DINIZ P C, WEINHARDT M. Compiling for reconfigurable computing: a survey[J]. ACM Computing Surveys, 2010，42(4): 1301-1365.

[2] SALVADOR R, OTERO A, MORA J, *et al*. Self-reconfigurable evolvable hardware system for adaptive image processing[J]. IEEE Transactions on Computers, 2013, 62(8): 1481-1493.

[3] AHN Y,HAN K, LEE G, *et al*. SoCDAL: system-on-chip design accelerator[J]. ACM Transactions on Design Automation of Electronic Systems, 2008, 13(1): 171-176.

[4] GOLDSTEIN S C, SCHMIT H, BUDIU M, *et al*. PipeRench: a reconfigurable architecture and compiler[J]. Computer, 2000,33(4): 70-77.

[5] MIYAMORI T, OLUKOTUN K,BUDIU M, *et al*. REMARC: reconfigurable multimedia array coprocessor[J]. IEICE Transactions on Information and Systems,1999, E82-D(2): 389-397.

[6] MEI B,VERNALDE S,VERKEST D, *et al*. ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix[A]. Proceedings of 13th International Conference on Field Programmable Logic and Application[C]. Lisbon Portugal, Springer Press, 2003.61-70.

[7] SINGH H, LEE M H, LU G M, *et al*. MorphoSys: an integrated reconfigureable system for data parallel and computation intensive applications[J]. IEEE Transactions on Computers, 2000, 49(5): 465-481.

[8] 窦勇，邬贵明，徐进辉等．支持循环自流水的粗粒度可重构阵列体系结构[J].中国科学:信息科学,2008,38(4):579-591.
DOU Y, WU G M, XU J H, *et al*. A coarse-grained reconfigurable computing architecture with loop self-pipelining[J]. Science China Information Sciences , 2008, 38(4): 579-591.

[9] MEI B,VERNALDE S,VERKEST D, *et al*. DRESC: a retargetable compiler for coarse-grained reconfigurable architectures[A]. Proceedings of 2002 IEEE International Conference on Field- Programmable Technology (FPT)[C]. Hong Kong, China,2002. 166-173.

[10] CARDOSO J M P, WEINHARDT M. XPP-VC: AC compiler with temporal partitioning for the PACT-XPP architecture[A]. Proceedings of 12th International Conference on Field-Programmable Logic and Applications (FPT)[C]. Berlin, 2002.864-874.

[11] YOON J W, SHRIVASTAVA A, PARK S,*et al*. A graph drawing based spatial mapping algorithm for coarse-grained reconfigurable architectures[J]. IEEE Transactions on Very Large Scale Integration Systems, 2009, 17(11): 1565-1578.

[12] LEE G,CHOI K, DUTT N D. Mapping multi-domain applications onto coarse-grained reconfigurable architectures[J].IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2011, 5(30): 637-650.

[13] BEREKOVIC M, KANSTEIN A, MEI B, *et al*. Mapping of nomadic multimedia applications on the ADRES reconfigurable array processor[J]. Microprocessors and Microsystems, 2009, 33(4): 290-294.

[14] KIM W,CHOI Y,PARK H. Fast modulo scheduler utilizing patternized routes for coarse-grained reconfigurable architectures[J]. ACM Transactions on Architecture and Code Optimization, 2013, 10(4): 1-58.

[15] 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[J].计算机学报, 2009, 32(6): 1089-1099.
WANG D W, DOU Y, LI S K. Loop kernel pipelining mapping onto coarse-grained reconfigurable architectures[J]. Chinese Journal of Computers, 2009, 32(6): 1089-1099.

[16] YOON J W, LEE J, PARK S, *et al*.Architecture customization of on-chip reconfigurable accelerators[J]. ACM Transactions on Design Automation of Electronic Systems,2013,18(4):58:1-58:22.

[17] 于苏东.可重构处理器的软硬件协同设计研究[D].北京:清华大学, 2009.
YU S D. Research on the Software/Hardware Co-design for Reconfigurable Processor[D]. Beijing: Tsinghua University, 2009.

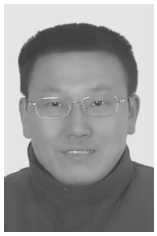
[18] 魏少军, 刘雷波, 尹首一. 可重构计算处理器技术[J].中国科学:信息科学, 2012, 42(12): 1559-1576.
WEI S J, LIU L B, YIN S Y. Key techniques of reconfigurable computing processor[J]. Science China Information Sciences, 2012, 42(12): 1559-1576.

[19] 孙康.可重构计算相关技术研究[D]. 杭州: 浙江大学, 2007.
SUN K. Research on Reconfigurable Computing Technologies[D]. Hangzhou: Zhejiang University, 2007.

[20] 陈乃金,江建慧.融合面积估算和多目标优化的硬件任务划分算法[J]. 通信学报, 2013, 34(2): 40-55.
CHEN N J, JIANG J H. Hardware-task partitioning algorithm merged area estimation with multi-objective optimization[J]. Journal on Communications, 2013, 34(2): 40-55.

[21] EL-ARABY E, GONZALEZ I, EL-GHAZAWI T. Virtualizing and sharing reconfigurable resources in high-performance reconfigurable computing systems[A]. Proceedings of the Second International Workshop on High-Performance Reconfigurable Computing Technology and Applications[C]. Austin, USA, 2008.1-8.

**作者简介：**



**陈乃金**（1972-），男，安徽合肥人，天津大学博士后，安徽工程大学副教授、硕士生导师，主要研究方向为可重构计算、时域划分与映射、VLSI/SoC 测试与容错等。

**冯志勇**（1965-），男，内蒙古呼和浩特人，博士，天津大学教授、博士生导师，主要研究方向为语义大数据表示与存储等。

**江建慧**（1964-），男，浙江淳安人，博士，同济大学教授、博士生导师，主要研究方向为可信系统与网络、软件可靠性工程、VLSI/SoC 测试与容错等。