文章编号：0253-374X(2017)08-1218-09 DOI:10.11908/j.issn.0253-374x.2017.08.017

行并行可重构单元阵列流水映射性能评估

陈乃金[1,2]，冯志勇[1]，江建慧[3]，何瑞祥[2]，王　真[4]

(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；4.上海电力学院 计算机科学与技术学院，上海 200090)

摘要：针对粗粒度单元阵列流水映射问题，设计了三种行流水结构阵列，并分析了其执行步骤，提出了一种基于行流水阵列通用的流水映射算法.该算法综合考虑混合多层迭代启动间距、块间流水通信成本、块配置成本等多个因素，一组测试基准程序实验结果表明了文中算法的合理性，与多目标优化映射算法相比，该算法消耗总时延平均节省了 4.0%(可重构单元阵列 $RCA_{4\times4}$)和 4.3%(可重构单元阵列 $RCA_{8\times8}$)；与满射映射相比，该算法消耗总时延平均节省了 52.1%($RCA_{4\times4}$)和 56.2%($RCA_{8\times8}$).

关键词：行流水；映射；多约束；流水段；启动间距

中图分类号：TP302　　**文献标志码**：A

Pipeline Mapping Performance Evaluation for Row Parallel Reconfigurable Cell Array

CHEN Naijin[1,2], *FENG Zhiyong*[1], *JIANG Jianhui*[3], *HE Ruixiang*[2], *WANG Zhen*[4]

(1. School of Computer Science and Technology, Tianjin University, Tianjin 300072, China; 2. College of Computer and Information Science, Anhui Polytechnic University, Wuhu, Anhui 241000, China; 3. School of Software Engineering, Tongji University, Shanghai 201804, China; 4. School of Computer Science and Technology, Shanghai University of Electric Power, Shanghai 200090, China)

Abstract: As for the problem of coarse-grained cell array pipeline mapping, this paper designed three row pipeline architecture array, analyzed their execution step, and presented a universal pipeline mapping (PM) algorithm for row pipeline array. This algorithm had comprehensive considered multi-level iteration initiation interval, communication costs between blocks, block reconfigurable costs and etc. The experimental results of a set of benchmark programs show the rationality of the algorithm. Comparing with multi-objective optimization map (MOM), the average execution total cycles of PM saved by 4.0% (reconfigurable cell array, $RCA_{4\times4}$) and 4.3% (reconfigurable cell array, $RCA_{8\times8}$). Comparing with epimorphism map (EPIMap) algorithm, the average execution total cycles of PM saved by 52.1% ($RCA_{4\times4}$) and 56.2% ($RCA_{8\times8}$).

Key words: row pipeline; mapping; multiple constraints; pipeline segment; initiation interval

粗粒度可重构体系结构(coarse-grained reconfigurable architecture，CGRA)可以获得高的计算性能和低的功耗消费[1].针对不同互连结构运算阵列，已有多种映射方法被提出[2-8]，但是存在以下缺陷：一是对数据流图(data flow graph，DFG)进行流水时域映射时，没有对可重构单元阵列(reconfigurable cell array，RCA)量化评估指标进行细化；二是没有给出或者说明 RCA 块流水执行流程.

针对已有研究的缺陷，本文进行了探索，其主要贡献如下所述：

(1)设计了三种行流水结构及映射算法，可重构多媒体系统(reconfigurable multimedia system，REMUS[7])具有近邻型粗粒度可重构单元阵列(neighbor coarse grained reconfigurable cell array，NCGRCA)的特征，在此基础上，设计了 NCGRCA 架构，同时设计了路由型粗粒度可重构单元阵列(router coarse grained reconfigurable cell array，RCGRCA)架构、总线型粗粒度可重构单元阵列(bus coarse grained reconfigurable cell array，BCGRCA)架构(图 1)；同时设计了一种面向上述架构流水映射(pipeline mapping，PM)算法，该算法综合考虑了流水线架构下的混合多层迭代启动间距、块间流水通

收稿日期：2016-10-23
基金项目：国家“八六三”高技术研究发展计划(2013AA013204)；国家自然科学基金(61432017，61572036)；安徽省自然科学基金(1408085MF124)；安徽省高校自然科学研究基金重点项目(KJ2015A003，KJ2013A001)；安徽省高校优秀中青年骨干人才国内外访学研修重点项目(gxfxZD2016102)
第一作者：陈乃金(1972—)，男，工学博士，副教授，主要研究方向为可重构计算、VLSI/SoC 测试与容错等.
E-mail：86naijinchen@tongji.edu.cn

信成本等多个要素.

(2) 将两个循环体的启动间距(initiation interval，II)推广到运算节点层面，考虑了循环 DFG 多个运算节点并行执行启动间距，同时给出了流水映射的形式化定义及流水执行成本的计算公式.

1　相关工作

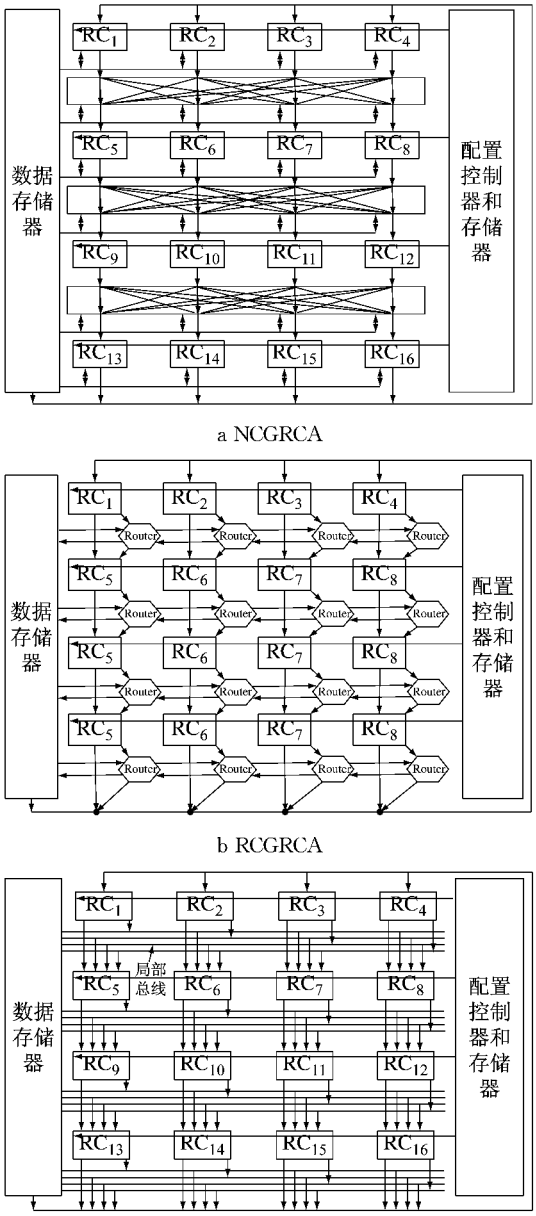
相关典型研究阐述如下：文献[2]提出了一种满射映射(epimorphism mapping，EPIMap)算法，该算法通过加旁节点过渡块间数据和重复计算来获得启动间距的较小化，但是没有给出启动间距的说明. 文献[3]从操作级层面对不跨层 RCA 互连时延评估进行了研究. 文献[4]基于统一 CGRA 行并行架构，设计实现了 RCA 块内跨行旁节点添加算法，并给出了其临界条件. 文献[5]考虑行并行粗粒度 RCA 执行的多个硬件资源约束，提出了一种粗粒度可重构体系结构多目标优化映射算法. 文献[6]设计了 CGRA 多线程编译框架，并对多线程 CGRA 的性能和功耗进行了分析. 但是上述文献对行并行和配置流水 RCA 执行定量分析考虑不足，没有给出 RCA 块执行的流水分段，对流水块间通信成本考虑不足.

本文研究的两个条件：① 面向行流水 RCA，提出将一条完整指令拆分 6 个流水段. 乘法运算设为 2 时钟周期(cycle)，其他算术逻辑运算时延设为 1 cycle，另外 5 个流水段执行时间均为 1 cycle. ② 一块 RCA 块内和块间数据传输和执行按行流水进行，配置成本包括 RCA 全局互连控制、重复单元(reconfi-gurable cell，RC)逻辑算术运算和路由控制等.

2　映射流水架构和 RCA 流水执行分析

2.1　映射流水架构

图 1 给出了行并行 NCGRCA、RCGRCA、BCGRCA 流水架构. 传统经典的 CGRA 架构主要有有 Morphosys[8]、REMARC[9] (reconfigurable multimedia array coprocessor)、LEAP[10] (loop engine on array processors). 相比现有架构，NCGRCA、BCGRCA、RCGRCA 三种结构具有的特点是：① RCA 块内和块间映射成功节点可按行流水执行和配置；② 上下行的可重构单元点点互连流水传递数据消除了 RCA 块内运算节点跨层数据传输的互连时延；③ 三种流水架构可以实现 RCA 块间流水. 图 1 中，Router 表示路由器.



a NCGRCA

b RCGRCA

c BCGRCA

图 1　三种 RCA 流水线结构

Fig.1　Three RCA pipeline structures

2.2　RCA 流水执行分析

定义 1　RCA 重复使用数 M：一个循环 DFG 可以表示为 $G=(V,E,W,D)$[11]，基于 CGRA，V 表示运算节点的集合，E 表示运算节点对之间边的集合，W 表示运算节点占用的可重构单元(reconfigurable cell，RC)个数，D 表示运算节点执行时延的集合. 本文研究的对象为定点数，一个节点占用一个 RC. 若 DFG 规模大于一块 RCA 面积，该 DFG 按硬件约束被划分映射到 RCA 上，并按时间域重复使用，这个重复使用数被记为 M.

定义 2 RCA 非原始输入(出)N_1 和 N_2 的细分:$G=(V,E,W,D)$,设 RCA_k 和其紧邻 RCA_{k+1} 分别表示第 k 和 $k+1$ 块 RCA,$R\{k\}-n$ 表示 RCA_k 的最后一行,$R\{k+1\}-1$ 表示 RCA_{k+1} 的第一行,若运算节点 $v_a,v_b\in V$ 已经被映射 RCA_k 的 $R\{k\}-n$ 行,若 v_a,v_b 的直接不跨层后继 v_c,v_d 被映射到紧邻的 RCA_{k+1} 的 $R\{k+1\}-1$,由于 RCA 为行流水结构,这样 RCA_{k+1} 块的运算节点 v_c,v_d 的非原始输入可以通过块间流水数据线直接从 RCA_k 块 $R\{k\}-n$ 行获得,这样的非原始输入不需要存储到数据存储器,则称为非原始输入 N_{11},否则需要存储到数据存储器的非原始输入称为 N_{12}. 同理,v_a,v_b 的非原始输出可以通过块间流水数据线直接传递到 RCA_{k+1} 块,这样的非原始输出不需要存储到数据存储器,则称为非原始输出 N_{21},否则需要存储到数据存储器的非原始输出称为 N_{22}. N_{11} 和 N_{12} 称为 RCA 非原始输入 N_1 的细分;N_{21} 和 N_{22} 称为 RCA 非原始输出 N_2 的细分. N_1 和 N_{12} 的关系是 $N_{12}=N_1-N_{11}$,N_2 和 N_{22} 的关系是 $N_{22}=N_2-N_{21}$,N_{11} 和 N_{21} 不需要存储,可作为行流水执行的一个段.

定义 3 RCA 配置时间 C_{CON} 的细分:非流水或流水 RCA 架构,完成一个任务 DFG 的配置时间 C_{CON} 大致包括两个部分:一个是完成 RCA 每次重复执行的全局运行控制字等配置成本,记为 C_{CON1};一个是完成 RC 逻辑算术运算、路由等的动态配置时间记为 C_{CON2}. C_{CON1} 和 C_{CON2} 称为流水 RCA 配置时间 C_{CON} 的细分.

定义 4 行指令级并行和行操作级并行:循环 DFG 中的任务节点按约束被映射到一块 RCA 上后,若运算节点按行进行指令级分段并发执行,称为运算节点的行指令级并行. 若按运算操作节点的类型并发执行,称为运算节点的行操作级并行.

分析 1 RCA 流水执行分析:RCA 按行操作并行执行,可以获得好的计算性能,基于流水 RCA 定量评估仍然是个难题,原因如下:①RCA 块内已映射节点的数据通路和 RCA 整体运行控制方式等均有所不同,RCA 每次运行均要消耗 C_{CON1};②行流水 RCA 结构,相邻行的节点可以点点流水快速执行,但是不相邻行的节点或位于不同 RCA 块的节点的存储成本也需要考虑.

由分析 1 可以得出:DFG 经过时域划分与映射后,RCA 流水执行步骤:

(1)固定配置 :即 C_{CON1},其值为一个常数 θ,不同的 CGRA 有所不同,REMUS 的 $\theta=17$cycle;

(2)预取数据:即不相邻块的数据提前预取非原始输入 N_{12} 或原始输入 N_{org1} 的成本,其表示一块 RCA 流水执行前从数据存储器预取数据并存入待执行 RC 的自身寄存器堆文件,目的是防止流水执行 RC 的读数据冲突;

(3)流水计算:即 $S_{pipeline}$,其表示一个 RCA 块内操作按行流水并行执行的成本;

(4)数据存储:即非原始数据不相邻块输出 N_{22} 或计算结果原始输出 N_{org2} 的成本.

基于流水执行步骤,给出如下定义:

定义 5 RCA 六段流水:流水线 RCA 一条完整的指令执行可划分为取指令(instruction fetch,IF)、指令译码(instruction decode,ID)、取数据(data fetch,DF)、硬件配置(hardware configuration,HC)、执行(execute,EX)、写回(write back,WB)6 段流水. 各段流水完成的功能为:IF 的功能是从指令存储器取出一条指令;ID 的功能是确定运算类型;DF 的功能是从 RC 寄存器堆或从 RCA 块紧邻上一行 RC 输入数据(即 N_{11});HC 的功能是通过配置寄存器来实现 RC 计算、路由等功能;EX 的功能是完成算术逻辑或复杂混合运算;WB 把 RC 计算结果写入紧邻下一行 RC 寄存器堆作为直接数据来源(即 N_{21})或写入数据存储器中. 通过流水获得一个 DFG 执行计算延迟可表示为 $S_{pipeline}$. $S_{pipeline}$ 包含了 N_{11}、N_{21} 等计算成本. 表 1 给出重复使用一块 RCA 的不跨 RCA 行操作级并行和指令级并行执行评估指标说明.

表 1 CGRA 操作级和指令级执行评估指标

Tab. 1 CGRA Operational level and instruction level execution evaluation indicators

含义	执行类型	
	行操作级并行	行指令级并行
一个 DFG 执行计算延迟	S_{SD}	$S_{pipeline}$
RCA 间的非原始输入次数	N_1	N_{12}
RCA 间的非原始输出次数	N_2	N_{22}
一个 DFG 所用的配置时间	C_{CON}	C_{CON}
一个 DFG 原始输入次数	N_{org1}	N_{org1}
一个 DFG 原始输出次数	N_{org2}	N_{org2}
一块 RCA 重复使用数	M	M

一块 RCA 流水执行流程如图 2 所示,循环 DFG 在一块 RCA 上流水执行总时延 $T_{TOTAL}=\alpha\cdot$(固定配置成本)$+\beta\cdot$(通信成本)$+\gamma\cdot$(流水执行成本)$=\alpha\cdot T_{固定配置}+\beta\cdot(T_{预取}+T_{存储})+\gamma\cdot T_{流水}=\alpha\cdot C_{CON1}+\beta\cdot[(N_{12}+N_{org1})+(N_{22}+N_{org2})]+\gamma\cdot S_{pipeline}$,其中,$\alpha$、$\beta$、$\gamma$ 为流水线 RCA 成本修正系数,

其值由具体可重构计算体系结构决定，本文设定评估系数的值为 $\alpha=\beta=\gamma=1$.

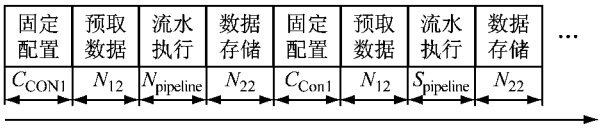


图 2　一块 RCA 流水执行流程

Fig.2　A RCA pipeline execution process

定义 6　RCA 混合多层迭代启动间距(multi-level iteration initiation interval，MIII)：传统的循环流水线启动间距是指相邻两个循环体的启动时间间隔，本文将其推广到计算任务节点启动时间间隔，有依赖的运算任务节点 DFG 被划分映射到 RCA 后，由于 RCA 的互连方式、面积等多个约束，会导致 RCA 块间或块内有的运算节点进行有依赖流水运行，有的运算节点进行无依赖流水运行，RCA 按段流水执行，每个节点的启动间距为一个固定值，将其称为 RCA 混合多层迭代启动间距 MIII. MIII 按映射到 RCA 块内后，运算节点有无依赖可分为图 3a～3c 三种情形. 无依赖流水 MIII=0；6 段 RCA 流水相邻行点之间存在依赖或混合依赖，下一行节点取数据段必须在上一行相关节点写回段后执行，故 MIII=3，RCA 按行流水执行三种情况如图 3d～3f 所示. 由图 3 可以得出流水段数为 m(本文 $m=6$)，每段流水线时间 Δt(本文 $\Delta t=1$cycle)，可以得出运算节点执行时间相等时，运算节点无依赖 $S_{pipeline}=(m-1)\Delta t$，运算节点有或混合依赖时 $S_{pipeline}=2m\Delta t-2\Delta t$. 将其推广，得到定理 1：

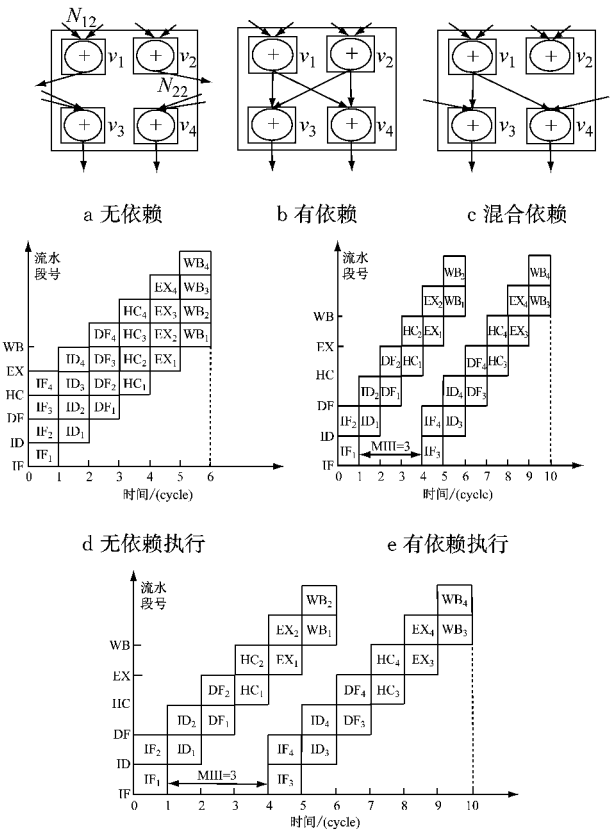


图 3　三种依赖 RCA 时空图

Fig.3　Three dependence RCA space-time diagrams

定理 1　一块行可流水执行 $RCA_{k\times b}$，有 m 段，n 个计算任务，Δt 为每段流水线的执行时间，$S_{pipeline}$ 为一块 $RCA_{k\times b}$ 流水线所消费的时间，level_max 为一块 RCA 已经成功映射的最长依赖子图长度，EX_1，EX_2，…，EX_m 为最长依赖子图 m 个节点的执行时延，$EX_{max\text{-}node_i}$ 为一块 RCA 无依赖节点的最大执行时延，且执行时间不等或相等. 则按块流水执行 $S_{pipeline}$ 为：

(1)无依赖执行：

$$S_{pipeline}=m\cdot\Delta t+\theta,\ \theta=EX_{max\text{-}node_i}-\Delta t$$

(2)有依赖或混合依赖：

$$S_{pipeline}=level\_max\cdot m\cdot\Delta t-2\cdot(level\_max-1)\cdot\Delta t=\sum_{level\_i=1}^{k}\{\max\{EX_1,EX_2,\cdots,EX_m\}-\Delta t\}$$

证明：(1)无依赖：设有 n 个节点被同时映射到一块 RCA，节点间无块内数据输入输出依赖. 流水线段数为 m，且时间为 Δt，则累加和为 $m\cdot\Delta t$，节点执行时间 EX_i($i\in[1,n]$)属于流水线中的一段，且 $EX_i\geqslant\Delta t$. (a)若一块 RCA 中 n 个无依赖节点 EX_i 均相等，且 $EX_i=EX_{max\text{-}node_i}=\Delta t$，则 $\theta=0$；$EX_i=EX_{max\text{-}node_i}\neq\Delta t$，则 $\theta=EX_i-\Delta t$；(b)若一块 RCA 中 n 个无依赖节点 EX_i 不相等，由于某个节点的 EX_i 决定了 m 段流水线执行段的延迟，所以 $\theta=EX_{max\text{-}node_i}-\Delta t$.

(2)有依赖或者混合依赖：设 RCA 具有 k 行，并且行与行之间的数据存在依赖，设有若干个子图被成功映射到一块 RCA，则最长子图的执行时延决定了该块 RCA 的 $S_{pipeline}$，由于每行节点执行 DF 段依赖于上一行的 WB 段，故当前行的 ID 和上一行的 EX 和 WB 可以并发执行. 则按硬件约束，RCA 第 1 行成功映射的节点数为 e 个，第 1 行 RCA 的基本执行时间 $T_1=m\cdot\Delta t+\max\{EX_1,\cdots,EX_e\}$；第 2 行流水执行时间依赖于第 1 行的数据写回，第 2 行成功映射的节点数为 f 个，则第 2 行可并发节点的取指和译码段可与上一行的执行和写回段重叠，第 2 行节点数据获取段须在第 1 行的运算数据写回段后进行，故实际流水段少了 $2\Delta t$，又因为第 2 行节点的执行时间段不等，需单独计算，第 2 行的执行时间为

$T_2=m\cdot\Delta t-2\cdot\Delta t+\max\{EX_{e+1},\cdots,EX_{e+f}\}$;依次递推,若第 i 行映射的节点集合为$\{v_k,\cdots,v_n\}$,则 $T_i=m\cdot\Delta t-2\cdot\Delta t+\max\{EX_k,\cdots,EX_n\}$.

综上所述:

$$\begin{aligned}S_{pipeline}=&T_1+T_2+\cdots T_k=m\cdot\Delta t+\max\{EX_1,EX_2,\cdots,EX_e\}+m\cdot\Delta t-2\cdot\Delta t+\max\{EX_{e+1},EX_{e+2},\cdots,EX_{e+f}\}+\cdots+m\cdot\Delta t-2\cdot\Delta t+\max\{EX_k\cdots,EX_n\}=\text{level\_max}\cdot m\cdot\Delta t-2\cdot(\text{level\_max}-1)\Delta t+\max\{EX_{e+1},EX_{e+2},\cdots,EX_{e+f}\}+\cdots=\text{level\_max}\cdot m\cdot\Delta t-2\cdot(\text{level\_max}-1)\cdot\Delta t+\sum_{\text{level\_}i=1}^{k}\{\max\{EX_1,EX_2,\cdots,EX_m\}-\Delta t\}\end{aligned}$$

证毕.

3 实验动机和流水映射算法设计

3.1 实验动机

本文实验动机包括以下两点:① 针对已有算法块间通信成本等指标仍较大的缺陷,设计一个考虑RCA块间按行流水线并行执行的流水映射(pipeline mapping,PM)算法. ② 考虑流水线背景下的 $S_{pipeline}$、N_{12}、N_{22}、C_{CON1}等指标的统计.

3.2 PM算法设计

多目标优化映射(multi-objective optimization map, MOM)算法在行可并行操作级CGRA获得了较好计算性能[5],但是通过深入的研究,在RCA流水线执行的背景下,发现MOM仍然存在 N_{12}、N_{22}等均较大的缺陷.为了克服这些不足,本文设计了流水映射PM算法,该算法考虑RCA块内和块间流水成本 N_{12}、N_{22}等的优化,采用的方法说明如下:

方法1 最小化RCA块内和块间流水数据输入和输出成本.

由2.2节可知,数据输入和输出成本是 T_{TOTAL} 的重要组成部分,所以对此优化具有一定意义.方法1具体包括两个方面:一方面是RCA块内流水数据输入和输出次数,由于本文是针对行并行流水RCA,且是点点行流水执行,对于跨层循环DFG通过加过多旁节点来带来配置成本的大量增加[4],通过对并行CGRA编译器的实际测试,每条DFG路径加旁节点(bypass node, BN)的个数约为小于等于2时,在不增加配置成本的前提下,可以减少RCA块内流水数据输入和输出次数;另一方面是充分考虑相邻块行流水的断流,策略是将当前RCA最后一行的节点直接后继映射到紧邻RCA的第一行,这样可减少RCA块间行流水执行的数据输入和输出次数.综上所述,方法1目的是优化 N_{12}、N_{22}.

方法2 尽可能减少行流水RCA混合多层迭代启动间距MIII.

采用按行左右运算节点无依赖映射、紧邻行上下有依赖点点映射方法.循环有依赖DFG被映射到6段流水RCA后,当前行节点的IF和ID可以与上一行节点的EX和WB段重叠执行,一方面保证了当前行节点的取数据DF(即在上一行节点的写回WB后执行)得以顺利执行;另一方面减少了MIII.同时进行节点贪婪映射,对一块RCA上的RC进行最大化利用,从而减少了RCA块使用次数的配置成本.方法2目的是优化 $S_{pipeline}$、C_{CON1}和 M.

由上述方法构造调度函数 $D(v_i)=\text{depth}(v_i)+\text{delay}(v_i)$,函数值越大优先级越高,其中,$\text{depth}(v_i)$表示DFG中点 v_i 的层号,其目的使层号大的节点具有高的优先级,体现 N_{12}、N_{22}的优化;$\text{delay}(v_i)$表示DFG中点 v_i 的执行时延,其目的尽可能把执行时延相同的点放在同一行,体现 $S_{pipeline}$的优化;并且采用节点贪婪映射,尽可能塞满每块RCA,体现 C_{CON1}和 M 优化.根据上述方法设计的PM算法流程见图4.设Max-level和 n 分别表示一个DFG最大层次号和节点总数,由PM算法流程图可知,按DFG层次和节点号扫描DFG消耗的时间复杂度为 $O(\text{Max-level}\cdot n)$,每次扫描DFG计算就绪节点 v_i 的调度函数 $D(v_i)$的值消耗的时间复杂度为 $O(n)$,综上所述,PM算法平均复杂度为 $O(\text{Max-level}\cdot n^2)$.

例1 为了说明PM算法的映射效果,给出如下例子,一段循环代码及DFG(其包括24个原始输入和4个原始输出)如图5a~5b所示,相关的映射结果如图5c~5d所示. 图5中,$i_1\sim i_4$ 为4个输入变量,$BN_1\sim BN_8$ 表示添加8个旁节点,$v_1\sim v_{32}$ 为32个顶点.表2给出了PM和MOM算法比较,结果显示PM具有较好的优化结果.

4 实验结果及分析讨论

为了便于比较,本文采用了IDCT、BTREE32、FDCT2、DCT8、FFT16等基准程序(表3),随机选定RCA面积(可表示为 A_{RPU})为16RC($RCA_{4\times4}$)、64RC($RCA_{8\times8}$),对于不同的 A_{RPU}值,采用一组基准程序集对PM、MOM、EPIMap等三种时域映射算法进行了测试.

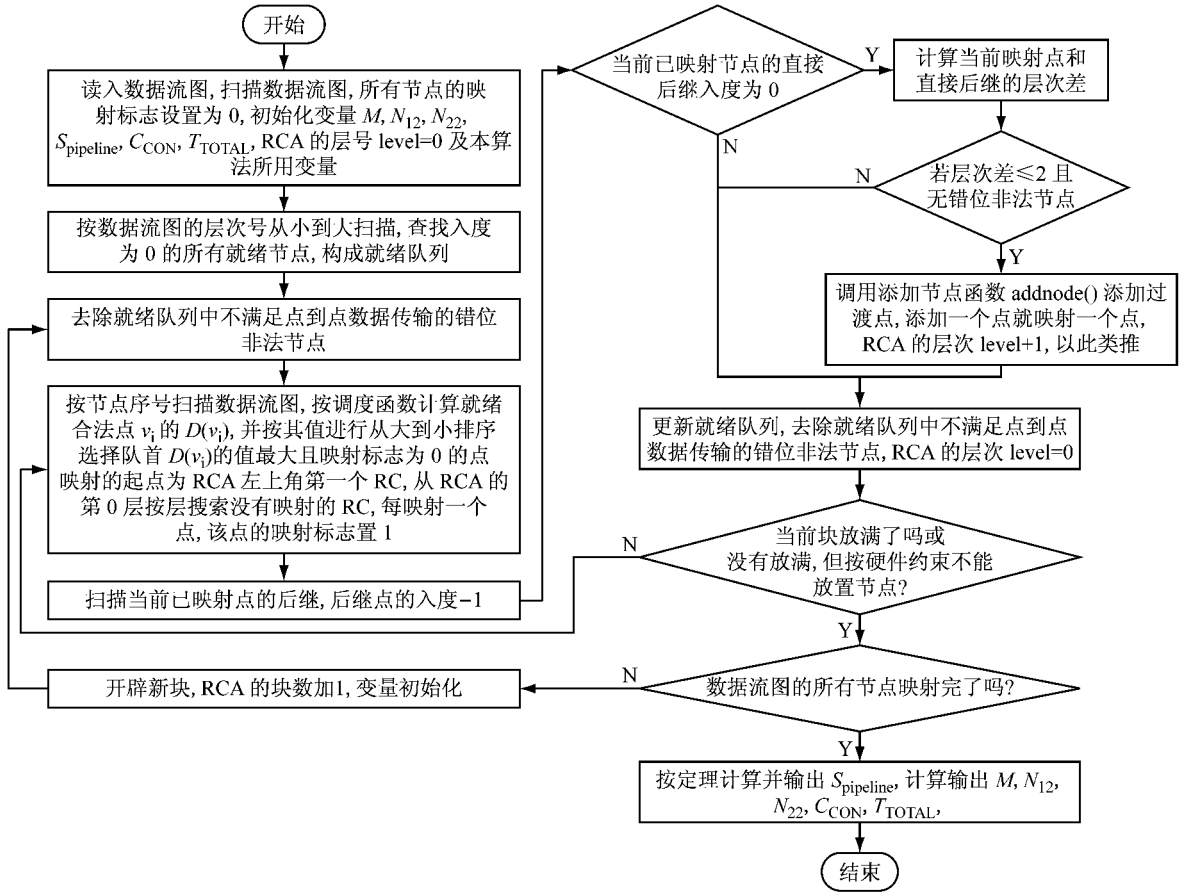


图 4　PM 算法流程图

Fig. 4　PM algorithm flow chart

表 2　MOM 和 PM 映射比较

Tab. 2　Comparison of MOM and PM mapping

参数指标	类型	
	MOM	PM
M	4	3
N_{12}	21	2
N_{22}	19	2
$S_{pipeline}$	48	46
C_{CON}	100	91
T_{TOTAL}	216	169

4.1　MOM、EPIMap 和 PM 比较

4.1.1　M 比较

由图 6 可以看出，当 $A_{RPU}=16$ 和 $A_{RPU}=64$ 时，相比较 MOM，随着 A_{RPU}的增大，除了 FDCT2，PM 算法均获得了较少的 M 值；相比较 EPIMap，PM 算法获得了 M 全部优化.

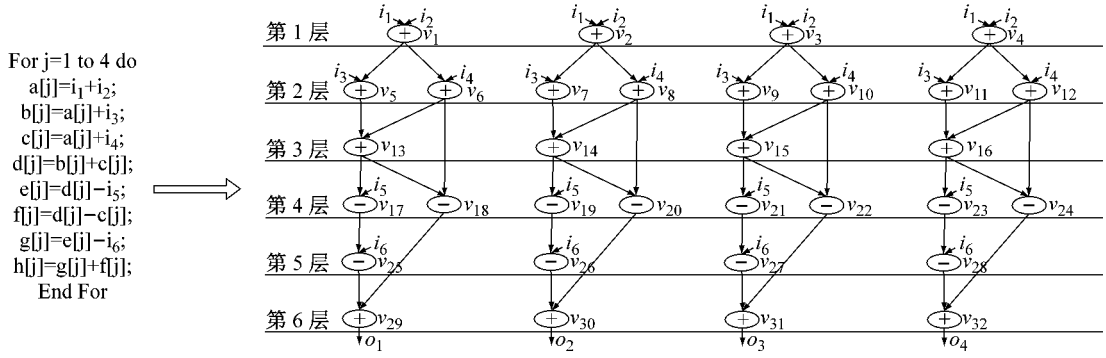
4.1.2　N_{11}和 N_{22}比较

由图 7 和图 8 可以看出，当 $A_{RPU}=16$ 和 $A_{RPU}=64$ 时，由于 MOM 对块间通信成本考虑不够，所以 PM 算法在指标 N_{12}方面获得了全面的优化；在指标 N_{22}方面，除了基准 FDCT2 相当外，PM 算法也是获得了较好的优化. 因为 EPIMap 以增大 M 作为代价来减少 N_{11}和 N_{22}，所以 EPIMap 获得的 N_{12}和 N_{22}值要优于 PM 算法.

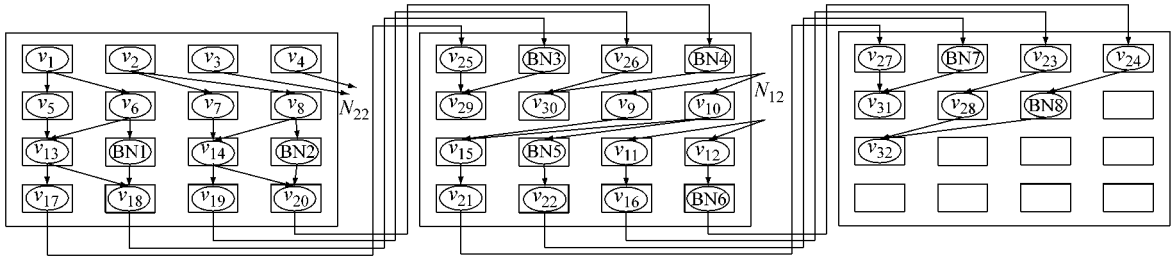
表 3　基准程序集

Tab. 3　Benchmark program sets

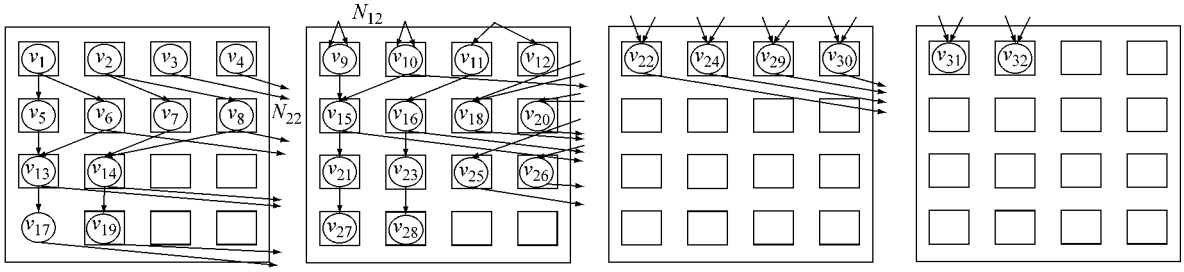
基准程序	原始输入次数	原始输出次数	边数	节点数	加法	减法	左移	右移	乘法	逻辑比较
IDCT	40	8	68	54	18	14	1	10		
BTREE32	32	1	30	31						31
EWF2	40	10	94	68	56				12	
FDCT2	64	16	104	84	26	26			32	
DCT8	98	8	82	90	40	16			34	
FFT16	45	37	121	84	31	42			11	



(a) 循环代码 (b) 循环 DFG



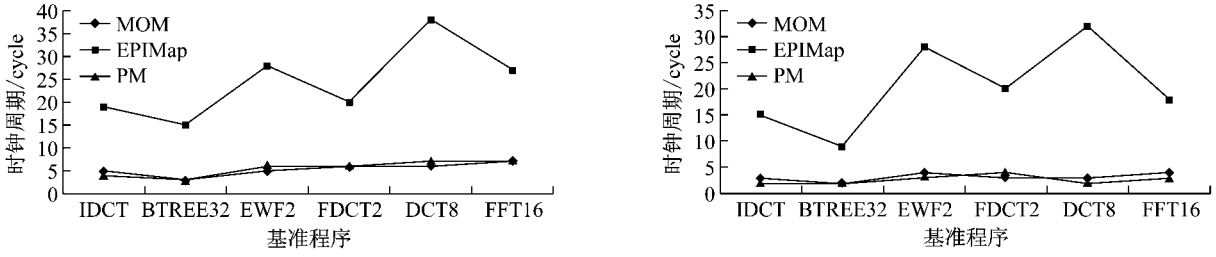
c 循环 DFG 的 PM 映射结果



d 循环 DFG 的 MOM 映射结果

图5 MOM和PM映射结果比较

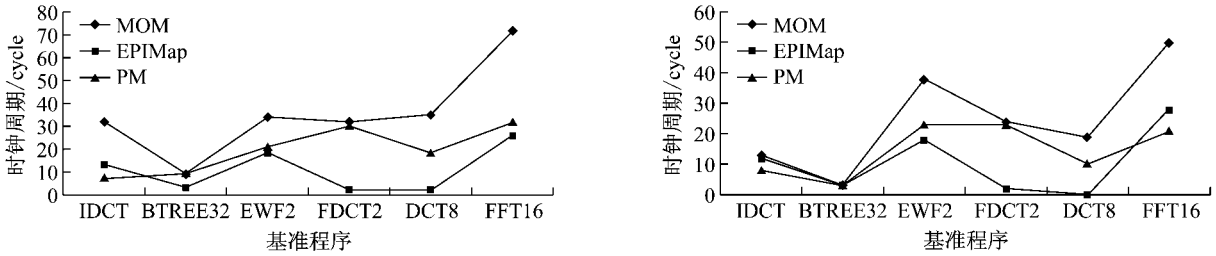
Fig.5 Comparison of MOM and PM mapping result



a $A_{RPU}=16$ b $A_{RPU}=64$

图6 MOM、PM、EPIMap映射 M 比较

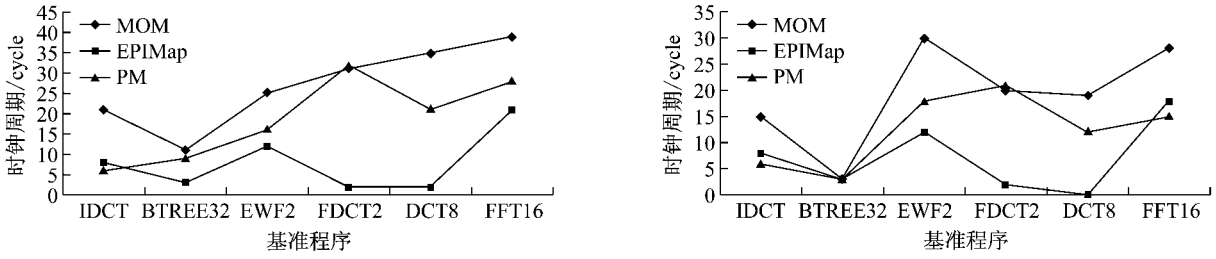
Fig.6 Comparison of M for MOM, PM, EPIMap mapping



a $A_{RPU}=16$ b $A_{RPU}=64$

图7 MOM、PM、EPIMap映射 N_{12} 比较

Fig.7 Comparison for N_{12} of MOM, PM, EPIMap mapping



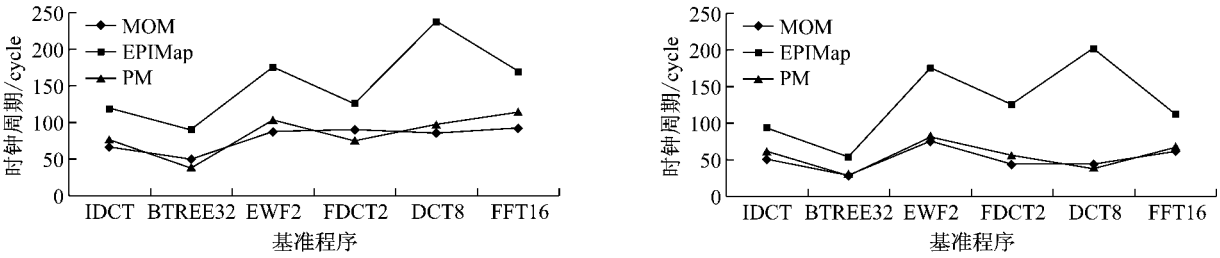
a $A_{RPU}=16$　　b $A_{RPU}=64$

图 8　MOM、PM、EPIMap 映射 N_{22} 比较

Fig. 8　Comparison of N_{22} for MOM, PM, EPIMap mapping

4.1.3　$S_{pipeline}$ 比较

由图 9 可以看出，当 $A_{RPU}=16$ 和 $A_{RPU}=64$ 时，由于 EPIMap 的 M 大，导致 $S_{pipeline}$ 较大，相比较 EPIMap，PM 算法在指标 $S_{pipeline}$ 方面获得了全面优化. MOM 在优化 $S_{pipeline}$ 方面取得了较好的效果，PM 在 $S_{pipeline}$ 优化方面不及 MOM.



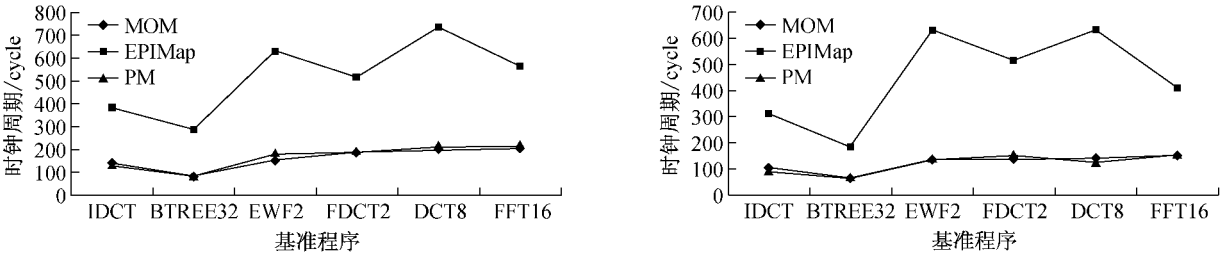
a $A_{RPU}=16$　　b $A_{RPU}=64$

图 9　MOM、PM、EPIMap 映射 $S_{pipeline}$ 比较

Fig. 9　Comparison of $S_{pipeline}$ for MOM, PM, EPIMap mapping

4.1.4　C_{CON} 比较

由图 10 可以看出，当 $A_{RPU}=16$ 和 $A_{RPU}=64$ 时，由于 EPIMap 的 M 大，导致 RCA 固定配置成本的增大，所以相比较 EPIMap，PM 算法在指标 C_{CON} 方面获得了全面优化. 随着 A_{RPU} 的增大，除了 FDCT2，PM 获得了较少的 M 值，所以相比 MOM，PM 在指标 C_{CON} 也获得了一定的优化.



a $A_{RPU}=16$　　b $A_{RPU}=64$

图 10　MOM、PM、EPIMap 映射 C_{CON} 比较

Fig. 10　Comparison of C_{CON} for MOM, PM, EPIMap mapping

4.1.5　T_{TOTAL} 比较

由表 4 可以看出，当 $A_{RPU}=16$ 和 $A_{RPU}=64$ 时，相比 MOM 算法，由于 MOM 的通信成本 N_{12} 和 N_{22} 考虑不够，导致 T_{TOTAL} 较大，PM 分别获得了 T_{TOTAL} 平均 4.0% 和 4.3% 的总体优化. 相比 EPIMap 算法，由于 EPIMap 算法的 M、$S_{pipeline}$、C_{CON} 等均较大，虽然通信成本 N_{12} 和 N_{22} 较小，但是不足以平衡 T_{TOTAL}，导致 PM 的总体性能优于 EPIMap，PM 获得了 T_{TOTAL} 改进百分比分别为 52.1%（$A_{RPU}=16$）和 56.2%（$A_{RPU}=64$）.

4.2　小结

从 4.1 节的实验结果可知，相比于 MOM 和 EPIMap 算法，随着硬件面积的增大，PM 算法在 M、C_{CON}、T_{TOTAL} 等较具优势. 但是PM算法的 $S_{pipeline}$ 不

表4 $A_{RPU}=16$ 和 $A_{RPU}=64$ MOM、PM、EPIMap映射 T_{TOTAL} 比较

Tab.4 Comparison of T_{TOTAL} for MOM, PM, EPIMap mapping Based on $A_{RPU}=16$ and $A_{RPU}=64$

划分基准	T_{TOTAL}						基准	T_{TOTAL}					
	MOM		PM		Δ/%			EPIMap		PM		Δ/%	
IDCT	307	232	266	216	−13	−7	IDCT	570	475	266	216	−53	−55
BTREE32	185	132	171	132	−8	0	BTREE32	415	277	171	132	−59	−52
EWF2	350	330	370	308	+6	−7	EWF2	888	888	370	308	−59	−65
FDCT2	420	304	403	332	−4	+9	FDCT2	726	726	403	332	−45	−54
DCT8	454	331	450	290	−1	−12	DCT8	944	778	450	290	−52	−63
FFT16	489	374	472	340	−4	−9	FFT16	862	651	472	340	−45	−48
Δ/%					−4.0	−4.3	Δ/%					−52.1	−56.2

如MOM算法，N_{12}和N_{22}不如EPIMap算法. PM算法适用场合：行流水RCA且要求较少 T_{TOTAL}.

5 结语

本文基于REMUS设计了NCGRCA架构，同时设计了RCGRCA、BCGRCA行流水并行执行架构，并分析了三种行流水结构阵列执行步骤，给出了流水线RCA性能评估新的细化指标 $S_{pipeline}$、N_{12}、N_{22}等，提出了一种流水映射PM算法，基于一组映射基准程序集，与MOM、EPIMap两种算法进行了比较，结果表明，PM算法在减少 M、T_{TOTAL} 等方面具有可行性.

参考文献：

[1] Kim C, Chung M, Cho Y, *et al*. ULP-SRP: Ultra low-power Samsung reconfigurable processor for biomedical applications [J]. ACM Transactions on Reconfigurable Technology and Systems, 2014, 7(3): 1.

[2] Hamzeh M, Shrivastava A, Vrudhula S. EPIMap: using epimorphism to map applications on CGRAs [C]// Proceedings of International Conference on 49th Design Automation Conference. Austin; IEEE Computer Society Press, 2012: 1284-1291.

[3] 陈乃金，冯志勇. 不跨层行操作并行RCA互连时延性能评估[J]. 天津大学学报(自然科学与工程技术版)，2017，50(4)：429.
CHEN Naijin, FENG Zhiyong. Interconnect delay perfor-mance evaluation for non-crossing level and row operands parallel RCA [J]. Journal of Tianjin University(Science and Technology), 2017, 50(4):429.

[4] 陈乃金，冯志勇，江建慧. 用于二维RCA跨层数据传输的旁节点无冗余添加算法[J]. 通信学报，2015，36(4)：1.
CHEN Naijin, FENG Zhiyong, JIANG Jianhui. Bypass node non-redundant adding algorithm for crossing-level data transmission in two-dimension reconfigurable cell array[J]. Journal on Communications, 2015, 36(4):1.

[5] 陈乃金，江建慧. 一种粗粒度可重构体系结构多目标优化映射算法[J]. 电子学报，2015，43(11)：2151.
CHEN Naijin, JIANG Jianhui. A Multi-objective Optimization Mapping Algorithm for Coarse Grained Reconfigurable Architectures [J]. *Acta Electronica Sinica*, 2015, 43(11): 2151.

[6] PAGER J, Jhrieyapaul R, Shrivastava A. A software scheme for multithreading on CGRAs [J]. ACM Transactions on Embedded Computing Systems, 2015, 14(1): 1.

[7] 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.
WEI Shaojun, LIU Leibo, YIN Shouyi. Key techniques of reconfigurable computing processor [J]. Science China Information Sciences, 2012, 42(12): 1559.

[8] 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.

[9] 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.

[10] 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[J]. 中国科学：信息科学，2008，38(4)：579.
DOU Yong, WU Guiming, XU Jinhui, *et al*. A coarse-grained reconfigurable computing architecture with loop self-pipelining [J]. Science China Information Sciences, 2008, 38(4): 579.

[11] 陈乃金，江建慧，陈昕，等. 一种考虑执行延迟最小化和资源约束的改进层划分算法[J]. 电子学报，2012，40(5)：1055.
CHEN Naijin, JIANG Jianhui, Chen Xin, *et al*. An Improved Level Partitioning Algorithm Considering Minimum Execution Delay and Resource Restraints [J]. *Acta Electronica Sinica*, 2012, 40(5): 1055.