基于集束式整数线性规划模型的专用指令集自动定制

赵 康 边计年 董社勤
(清华大学计算机科学与技术系 北京 100084)
(zhao-k04@mails.tsinghua.edu.cn)

摘 要 提出集束式整数线性规划形式化模型,利用指令间的功能依赖性解决专用指令集处理器中指令集自动定制的指数性空间问题. 在此基础上,针对其前端和后端分别提出了相应的指令定制实现策略. 实验结果表明,该指令定制方法可以有效地实现专用指令集的自动设计,并使最终处理器的运算性能得到优化.

关键词 专用指令定制;设计自动化;集束式整数线性规划模型;专用指令集处理器
中图法分类号 TP302.1

Specific Instruction-Set Automated Customization Based on Clustering ILP Model

Zhao Kang Bian Jinian Dong Sheqin
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract To solve the problem of exponential space in the instruction-set automated design for the application specific instruction-set processor (ASIP), a formular clustering integer linear programming model (CIM) is proposed, which can decrease the exploration space effectively utilizing function dependencies between instructions. In addition, implemental strategies are presented for the front-end and back-end in the whole design. The experimental results show that this instruction customization technique can effectively achieve the specific instruction-set automated design, and it can also increase the performance of the final synthesized processor.

Key words specific instruction customization; design automation; clustering integer linear programming model; application specific instruction-set processor

专用指令集处理器(application specific instruction-set processor, ASIP)是系统级芯片(system-on-a-chip, SoC)的重要组成部分,它同时兼有专用集成电路的高效性和通用处理器的灵活性. 面对当今对 SoC 上系统的高性能、低功耗、低成本等诸多要求以及缩短产品上市时间的双重压力,根据用户的特定应用需求来定制专用的 ASIP 已成为最佳解决方案[1]. 然而,由于专用微处理器本身设计的复杂性,完全实现其设计的自动化是一项极具挑战性的工作.其中专用指令集不仅决定着最终处理器的功能,对其性能也有影响,因此,如何定制一个符合用户需求的专用指令集对实现 ASIP 的自动生成至关重要[2].

近年来,针对 ASIP 专用指令集综合的研究非常活跃,根据指令集的生成方式可将其分为 3 类:Holmer 首次将指令集的自动生成定义为一个优化问题,并对整个指令集的自动生成方法进行了详细归类[3];此外,指令选择作为一种有效的技术被提出,其主要手段是根据特定的功能需求从指令池中选择已有的指令构成指令集[4];指令集扩展(instruction

收稿日期:2007-01-24;修回日期:2007-03-20. **基金项目**:国家自然科学基金(90207017, 60236020);国家"九七三"重点基础研究发展规划项目(2005CB321605). **赵 康**,男,1982 年生,博士研究生,主要研究方向为专用处理器的综合与定制、VLSI 设计自动化. **边计年**,男,1945 年生,博士,副教授,博士生导师,主要研究方向为数字系统设计自动化(bianjn@tsinghua.edu.cn). **董社勤**,男,1964 年生,博士,副教授,主要研究方向为 VLSI CAD 算法和系统.

set extension，ISE)也逐渐成为研究的热点[5]，其主要思想是基于已有的基本指令集，根据应用需求通过增加新指令来实现指令集的定制，该方法的局限性是它必须依赖于某一特定的处理器. 国内对专用指令定制的研究还处于初步阶段[6]，仅局限于处理器的前端编译和后端模拟.

除了本身固有的局限性外，以上 3 类方式都不利于实现专用指令定制的自动化，都需要人为地强行干预. 如完全的指令集自动生成很明显有过多的重复性操作，而指令集的选择则提前需要人为地构造可供选择的指令池或指令库，ISE 则要求对输入的应用程序做大量的分析、统计和优化；同时，专用指令的定制不可避免地会牵涉到编译优化与可配置的体系结构等方面. 本文研究的重点正是如何降低该问题的搜索空间，并实现快速而且准确地自动定制满足用户需求的专用指令.

本文借鉴已有的工作，通过对比以上 3 类生成方式的优缺点，确定了采取指令选择与指令扩展相结合的策略. 同时，由于新定制的复杂指令与基本指令之间必然出现功能重叠或功能依赖，本文提出了一种集束式整数线性规划(integer linear programming，ILP)模型——CIM(clustering ILP model)，用于对指令的选取进行抽象化描述，并围绕着指令的 ILP 选择，详细介绍了其前端和后端所采取的指令定制策略，以有效地解决专用指令定制过程中的降低搜索空间问题.

1 专用指令定制流程

为了将指令选择和指令扩展 2 种策略的优势进行结合，本文给出图 1 所示的专用指令定制流程，该流程的核心是专用指令的选取. 首先通过指令模板鉴定得出应用程序中可定制的候选指令集；之后根据系统性能约束条件对指令池中的候选指令进行选取，以确定最终指令集；选取结果通过指令的添加后，提交处理器模拟工具进行性能的分析和比较，最终评估结果将返回指令的选取过程作为启发式的决策信息，如此循环直到得到一个较优的定制结果.

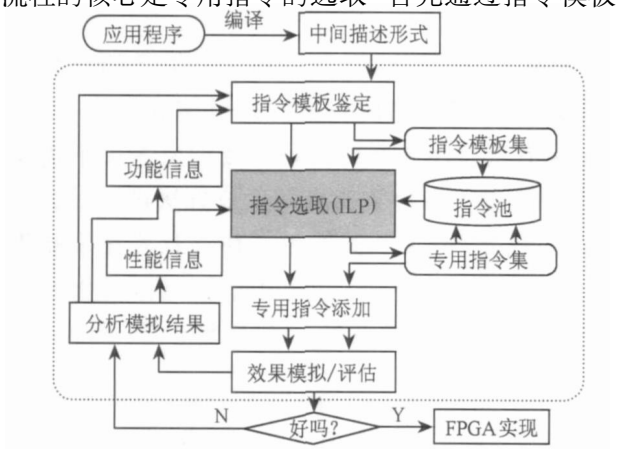

图 1 专用指令定制设计流程

在该流程中，数据流图(data flow graph，DFG)被用作系统功能的中间表示格式，由系统描述(C/C++等高级语言)直接转换得到. 整个指令定制的流程均利用 DFG 上的操作来完成，包括图的搜索、合并、抽取等.

2 CIM

在整个专用指令定制流程中，专用指令选择属于优化问题，其输入是指令池中的候选指令集，输出是满足用户功能与性能双重需求的最终指令集. 由于复杂指令由基本指令组成，指令之间必然存在一定的功能相关性或依赖性，问题的关键就在于如何利用这种功能相关性来有效地降低整个指令定制问题的搜索空间.

对于优化问题，首先面临的任务是如何利用数学模型对它进行形式化和抽象化. 一般情况下，优化问题包括一个设计空间以及一系列约束条件和解决方法. 针对指令的选取问题，本文将其搜索空间定义在运行时间、综合的面积和功耗 3 个约束条件下.定义 $I=\{u_1, u_2, \cdots, u_n\}$ 为原始指令集，$I^*=\{v_1, v_2, \cdots, v_t\}$ 为选取优化后的最终指令集. 于是得到集合元素个数 $|I|$ 和 $|I^*|$ 的关系式

$$\begin{cases} |I| = n \\ |I^*| = t \\ |I^*| \leqslant |I| \Rightarrow t \leqslant \mathrm{n} \end{cases} \tag{1}$$

设最大元件占用面积为 $A_{\max}$，对指令 u_i 进行元件面积计算的函数为 $area(u_i)$，则可得到

$$\sum_{i=1}^{n} w_i \times area(u_i) \leqslant \mathrm{A}_{\max},\ w_i = \{0,1\} \tag{2}$$

其中 w_i 是布尔变量，$w_i=1$ 代表指令 u_i 被选中，$w_i=0$ 代表 u_i 未被选中.

指令选取问题可以被抽象描述如下：

1) 标识符

I 表示原始指令集合，I^* 表示选中的指令组成的集合，P 表示功耗，T 表示总的运行时间，G 表示由原始指令集到新指令集的映射.

2) 问题描述

给定原始指令集 I,在式(1),(2)约束条件下,由 I 到最终指令集 I^* 之间产生映射关系 G,使得 I 中的每一项功能操作都会在 I^* 中被覆盖,且实现运行时间 T 与功耗 P 的最小化.

然而,由于所有指令形成的设计空间非常巨大,很难达成统一的描述或定义,因此选择的范围往往根据具体指令实现的功能差别来进行划分. 基于这种考虑,本文提出了一种功能集束化的思路,并在 ILP 问题描述的基础上进行了改进,使具有功能依赖特性的指令集取代了单条指令并作为运算的最小单位.

假定 $F=\{F^1, F^2, \cdots, F^m\}$ 代表不同功能的指令集的集合,其中任意 2 个元素不存在交集,即 $F^i \cap F^j=\varnothing$($i\neq j$ 且 $1\leqslant i, j\leqslant m$). 设 u_j^i 为 F^i 中的第 j 条指令,有

$$F=\left\{F^1 \cup F^2 \cup \cdots \cup F^m\right\},\ |F|=m;$$

$$F^1=\left\{u_1^1, u_2^1, u_3^1, \cdots, u_{n1}^1\right\},\ |F^1|=n_1;$$

$$F^2=\left\{u_1^2, u_2^2, u_3^2, \cdots, u_{n2}^2\right\},\ |F^2|=n_2;$$

$$F^m=\left\{u_1^m, u_2^m, u_3^m, \cdots, u_{nm}^m\right\},\ |F^m|=n_m;$$

$$F^i \cap F^j=\varnothing,\ i\neq j,\ i, j\in[1, m].$$

最初指令集中指令之间存在某种功能上的依赖性,尤其在原子指令与复杂指令之间. 以图 2 为例,MUL 操作既可利用 ADD 指令,通过软件控制的累加方式来达到,也可以通过由乘法器和加法器组合而成的乘加运算来完成,其差别仅在于运算方式的不同. 不管利用硬件的方式(直接用复杂指令)还是用软件的方式(间接利用另一条简单指令),都取决于对程序的分析结果. 基于这种考虑,本文假定函数 $U(F)$代表指令集 F 功能映射与预处理操作的结果.

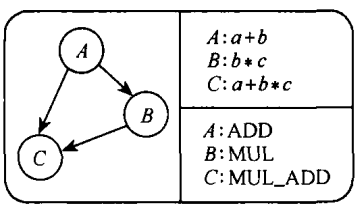

图 2　指令间功能依赖性示例

本文采用布尔变量 x_j^i 来表示指令 u_j^i 是否选中,当 u_j^i 被选中时,$x_j^i=1$;否则,$x_j^i=0$. 于是有

$$t=\sum_{i=1}^{|U(F)|}\sum_{j=1}^{n_i} x_j^i,\ x_j^i=0 \text{ or } 1,\ |U(F)|\leqslant m.$$

由于每个 F^i 代表某一种操作运算,若缺少某种运算则处理器不能正确执行相应的应用程序,因而每个 F^i 中至少要有一条指令被选中. 在该约束下,利用集束型 ILP 的思想将指令选取问题描述为

$$\min:\sum_{i=1}^{|U(F)|}\sum_{j=1}^{n_i} x_j^i\times f_j^i\times\{q\cdot time(u_j^i)+(1-q)power(u_j^i)\},$$

$$\text{Subject}:\begin{cases}\sum_{i=1}^{|U(F)|}\sum_{j=1}^{n_i} x_j^i\times area(u_j^i)\leqslant A_{max}\\ t\leqslant n\Rightarrow\sum_{i=1}^{|U(F)|}\sum_{j=1}^{n_i} x_j^i\leqslant n\\ \forall i\left(1\leqslant i\leqslant|U(F)|\wedge\sum_{j=1}^{n_i} x_j^i\geqslant 1\right)\end{cases},$$

$$x_j^i=0 \text{ or } 1;$$

其中,f_j^i 代表目标应用程序中指令 u_j^i 的利用次数,q 表示与功耗相比运行时间的因素所占的比率,$time(u_j^i)$和 $power(u_j^i)$分别代表针对指令 u_j^i 的运行时间和功耗.

3) 分析. 前人工作中针对指令选取所采用的 ILP 模型的搜索空间为 2^n. 由于指令之间功能集束的限制,本文提出的 CIM 将指令选取问题的搜索空间明显降低了,因为 $n=n_1+n_2+\cdots+n_m$,所以

$$C_{n1}^1 2^{n1-1}\times C_{n2}^1 2^{n2-1}\times\cdots\times C_{n|U(F)|}^1 2^{n|U(F)|-1}=$$
$$(n_1\cdot n_2\cdots\cdot n_{|U(F)|})2^{n1-1+n2-1+\cdots+n|U(F)|-1}\leqslant$$
$$(n_1\cdot n_2\cdots\cdot n_m)2^{n1-1+n_2-1+\cdots+n_m-1}=$$
$$(n_1\cdot n_2\cdots\cdot n_m)2^{n-m},$$

故$[2^n-(n_1\cdot n_2\cdots\cdot n_m)2^{n-m}]/2^n=1-(n_1\cdot n_2\cdots\cdot n_m)/2^m$.

分析结果表明,与一般的 ILP 描述相比,采用 CIM 可以使专用指令定制问题的搜索空间降低 $(1-(n_1 n_2\cdots n_m)/2^m)$.

3　专用指令定制自动化的实现与分析

对于实现整个指令定制过程的自动化,仅解决了专用指令的选取还不够,还需要对其前端和后端提出相应的实现策略,才能最终生成合适的指令集. 因此,实验中需要重点解决 2 个问题:1) 如何进行指令模板的抽取,才能得到适合指令选取所需要用到的指令池;2) 如何将新定制的指令添加到应用程序中,并在汇编码中反映出新指令的存在.

3.1　前端:指令模板的鉴定

图 1 中,指令选取的任务是针对已有的指令池,根据功能与性能上的需求进行筛选,得到最终的指令集. 为了从最初的应用程序中得到指令池中的候选指令,需要在其前端部分完成指令模板的鉴定.

定义 1. 用于构造指令池中候选指令的由基本操作组合而成的复杂操作运算称为指令模板.

为了挖掘应用程序中的指令模板信息,首先将应用程序转换为 DFG,在此基础上进行频度分析,并通过基本操作的组合得出候选指令模板. 在 DFG 中,基本操作映射为单个节点,而每个指令模板对应一组原子节点,即一个子图. 因此,指令模板的鉴定本质上是有效子图的抽取问题. 如图 3 所示,DFG 中多次用到了 2 种复杂操作运算——乘加操作和与移位或操作. 为了加快运算速度,将该 2 种运算抽取出来作为指令模板进行综合,并相应加入硬件运算单元,即可提升处理器的运算性能.

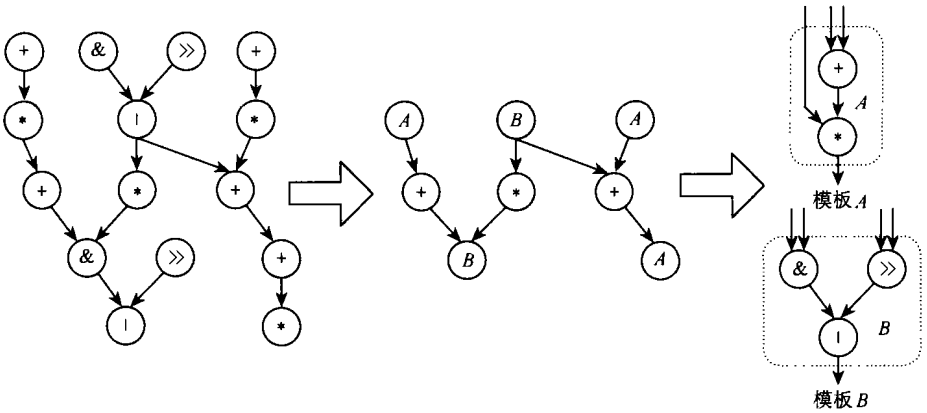

图 3 指令模板的鉴定与抽取

定义 2. $G(V,E)$为一有向无环图,其中 V 是节点的集合,代表应用程序中基本的原子操作,E 是边的集合,代表原子操作之间的数据依赖关系.

定义 3. 设 $G'(V',E')$是 $G(V,E)$的子图,其始于 V'之外节点终止于 V'之内节点的边的数目称为子图 G'的入度;始于 V'之内节点终止于 V'之外节点的边的数目称为子图 G'的出度.

定义 4. 设 $G'(V',E')$是 $G(V,E)$的子图,对于 V'内的任意节点 v_1 与 v_2,如果 v_1 与 v_2 之间的所有路径上的节点全部包含在 V'中,则称 G'为 G 的凸子图;否则,称 G'为 G 的非凸子图.

如图 4 所示,对于同一有向无环图 G,图 4a 中虚线框内的子图是 G 的凸子图,图 4b 中是非凸子图,这是由于图 4b 中 E 节点被排斥在外,且处于节点 B 与 F 之间的路径上,而节点 B 和 F 都在子图范围之内,因此图 4b 中子图是非凸的. 基于以上定义,可以看出指令模板对应的有效子图的特性:

1) 由于指令的输入/输出操作数的数目有限,有效子图的入度与出度受到限制.

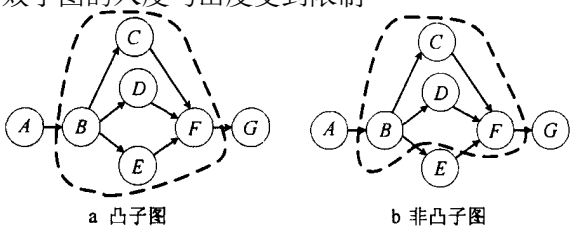

图 4 凸子图与非凸子图

2) 有效子图必须是凸子图. 这是由于对于非凸子图,必然会出现与另一子图之间同时存在输入边与输出边的情况,即形成一条环路,不利于实现专用指令定制的自动化.

同时满足以上 2 个条件的子图才可以抽取出来并存入指令池,为专用指令的选取提供候选指令集.

3.2 后端:专用指令的添加

通过专用指令的选取得到最终指令集后,如何将新指令添加到应用程序中,并保证交叉编译器和模拟器能够识别出?

由于每条新指令对应于 DFG 中一个有效子图,因此可以通过子图的合并完成 DFG 的简化. 如图 3 中经过新指令的添加,其节点数目由 15 减为 8. 而由于 DFG 本身与应用程序之间一一对应,因此,本文在源码层和汇编层同时对应用程序进行修改,用指令模板对原程序中的原子操作集合进行替换,并修改编译器自带的指令集,使之能够在汇编层识别出新指令. 如图 5 所示,在代码层将指令模板 A 所对应的代码部分转换成为"函数调用"的格式,接口

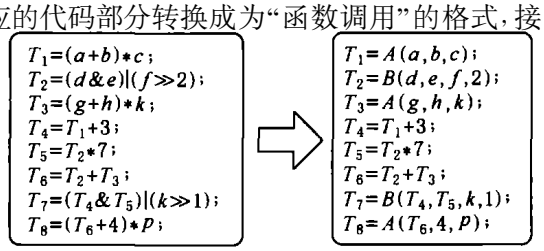

图 5 专用指令在代码层的添加

的参数均按照操作数顺序进行读写操作;而在汇编层,每条新定制的汇编指令则对应一组原子指令,如图 6 所示.

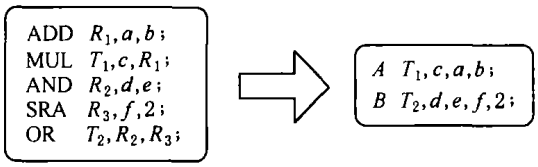

图 6　专用指令在汇编层的添加

该方法的优势是可以清晰地将新指令反映到应用程序中,而且不会改变其功能的实现,有利于编译器和模拟器的识别;其不足之处是不可避免地会引起寄存器分配情况的紊乱,需要重新进行寄存器分配的调整. 例如图 6,原程序需要用到 3 个寄存器 R_1, R_2 和 R_3,但经过新指令的添加后则不再需要. 因此,若前后的指令间存在数据依赖的话,必须重新进行寄存器的调整才不致使程序运行出现错误.

将新指令添加到应用程序中后,则可对应改动交叉编译器和体系结构模拟器自带的指令集,并对应用程序进行重新编译、模拟,最终通过比较可以模拟出新指令对处理器运算性能的提升效果.

3.3　分　析

指令定制的目的在于如何将复杂运算定制为专用指令,从而提高处理器的运算性能. 因此,实现指令定制自动化的关键在于 3 点:1) 如何从应用程序中抽取出需要定制的指令;2) 抽取出的专用指令集如何在功能和性能上同时满足需求;3) 如何使新指令能够为交叉编译器与模拟器识别.

本文所提出的定制策略在以上 3 个关键点都有优势. 首先,利用指令模板的鉴定,从 DFG 中抽取出满足约束条件的候选指令集,为指令的选择提供了可操作的对象;其次,根据专用指令选择的抽象化描述与定义,按照 ILP 的规则选取出功能与性能上同时满足用户需求的专用指令集;最后通过"函数调用"形式的转换,完成新指令在代码层与汇编层的添加,使编译器和模拟器能够识别出新指令.

基于以上分析,该方法具有如下优点:1) 层次分明. 分别有针对性地解决了专用指令定制的几个关键问题;2) 将复杂问题简化. 从应用程序中抽取出满足功能与性能双重需求的指令本身就是很复杂的问题,本文将其分解为模板鉴定和指令选取两步,有效地降低了问题的复杂性;3) 围绕专用指令的 ILP 选取,根据用户在功能与性能上的需求,用有向无环图的操作代替直接对程序代码的剖析.

4　专用指令对处理器性能影响的模拟验证

利用本文方法进行了初步的实验,用于检验专用指令集对最终综合出的处理器性能的影响. 由于指令系统的好坏直接影响各方面的性能,因此实验中采用与 X86 兼容的指令系统,并在此基础上增加了 2 条专用指令:乘法指令 MUL 和乘加指令 MA. 其中 MA 指令的一个乘数默认读取寄存器 R_{10} 中的数值,如表 1 所示.

表 1　乘法指令与乘加指令的设计

指令名	含义
MUL *DR*, *SR*	$DR\times SR\rightarrow DR$
MA *DR*, *SR*	$DR+SR\times R_{10}\rightarrow DR$

由于增加了乘法和乘加 2 种运算元件,因此必须修改基准处理器的体系结构. 实验中所设计的运算器 ALU 包括 4 个部分:1)16 位的算术逻辑部件;2)FLAG 寄存器,由 SST 控制选择标志运算状态的信号来源;3)16 个 16 位的通用寄存器组;4)乘法器和乘加器. 乘法器和乘加器 2 种特殊的操作都设定为三周期指令,其中前 2 个周期用于取指,第 3 个周期用于计算. 此外,由于默认规定 MA 指令涉及到寄存器 R_{10} 内的数值,因此实验要求修改体系结构,并加入寄存器阵列到 MUX 的信号输入,用于传递 R_{10} 寄存器的数据.

实验采用的测试程序为求 $\sum n^2$ 的值. 对于该应用有 3 种不同的实现方案:1)不用乘法指令;2)只用乘法指令;3)用乘加指令.

实验方案 1 的汇编码如下:

```
MOVI R1,0      ;结果
MOVI R0,5      ;n=5
MOVI R6,15
L1:
MOV  R2,R0     ;被乘数
MOV  R3,R0     ;乘数
MOVI R4,0      ;积
L3:
MOVI R5,1
AND  R5,R3
JRZ  L2
ADD  R4,R2
L2:
SHL  R2
SHR  R3
```

```
MOVI R6,15
AND R6, R3
JRNZ L3
ADD R1, R4
DEC R0
JRNZ L1
HALT
```

实验方案 2 的汇编码如下：

```
MOVI R1,0      ;结果
MOVI R0,5      ;n=5
L1:
MOV R2, R0     ;被乘数
MUL R2, R0     ;积
ADD R1, R2
DEC R0
JRNZ L1
HALT
```

实验方案 3 的汇编码如下：

```
MOVI R0,0      ;结果
MOVI R10,5     ;n=5
L1:
MOV R1, R10    ;被乘数
MA R0, R1      ;积
DEC R10
JRNZ L1
HALT
```

最终模拟的实验结果如表 2 所示. 可以看出，用乘加指令的 CPU 运算效率最高，只用乘法的其次，不用乘法指令的最差，其中使用乘加指令可以减少约 20%的运行时间. 这是因为如果不用乘加指令，则需要用移位并累加的方法模拟乘法并耗费大量的时间，而且其复杂度和 n 的长度有关系. 如果用乘法或者乘加指令的运算时间是 $O(n)$的话，那么不用乘法的运算时间就需要再乘以 $\mathrm{lb}\,n$，其复杂度是 $O(n\log n)$.

表 2 3 种实验方案的运行时间比较

实验方案	运行时间/ns	运行周期	比率
1	40 850	409	6.50
2	7 750	78	1.24
3	6 250	63	1

因此对于特定的应用需求，定制专用指令对处理器计算性能的提高是明显的，这对于那些单纯依靠算法设计很难提高运算效率的应用问题具有一定的现实意义.

5 结 论

本文针对专用指令集的定制方法，提出了一种围绕 CIM 进行设计的指令定制方法. 该流程以专用指令的 ILP 选取为中心，结合前端的指令模板鉴定与后端的专用指令添加，既保证了最终指令集满足功能上的一致性，又兼顾性能上的系统约束. 分析及实验结果表明，该定制方法层次分明，有效地降低了专用指令定制问题的复杂性，最终对处理器的运算效率有着明显的提高.

今后工作重点将集中在指令枚举与指令选择相结合的算法设计，以及面向多核处理器的专用指令定制研究方面.

参 考 文 献

[1] Cong J, Fan Y, Han G, *et al*. Application-specific instruction generation for configurable processor architectures [C] //Proceedings of the 12th International Symposium on Field Programmable Gate Arrays, Monterey, California, 2004: 183—189

[2] Jain M K, Balakrishnan M, Kumar A. ASIP design methodologies: survey and issues [C] //Proceedings of the 14th International Conference on VLSI Design, Bangalore, 2001: 76—81

[3] Holmer B K. Automatic design of computer instruction sets [D]. Berkeley: University of California, 1993

[4] Gschwind M. Instruction set selection for ASIP design [C] //Proceedings of the 7th International Workshop on Hardware/Software Co-Design, Rome, 1999: 7—11

[5] Biswas P, Choudhary V, Atasu K, *et al*. Introduction of local memory elements in instruction set extensions [C] //Proceedings of the 41st Design Automation Conference, San Diego, California, 2004: 726—734

[6] Zhao Xuemi, Wang Zhiying, Yue Hong, *et al*. Automatic generation of application specific instruction-set processors directed by transport triggered architecture [J]. Journal of Computer-Aided Design & Computer Graphics, 2006, 18(10): 1491—1496 (in Chinese)
(赵学秘, 王志英, 岳 虹, 等. 传输触发体系结构指导下的 ASIP 自动生成[J]. 计算机辅助设计与图形学学报, 2006, 18(10): 1491—1496)