第 34 卷 第 21 期 计 算 机 工 程 2008 年 11 月
Vol.34 No.21 Computer Engineering November 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)21—0219—03 文献标识码：A 中图分类号：TP31

基于整数线性规划的 TTA 代码优化

胡 维，祝永新，姜 雷
(上海交通大学微电子学院，上海 200240)

摘 要：针对传输触发结构代码生成中的指令调度、多寄存器堆分配、全局寄存器分配和软件旁路等优化问题，给出一个整数线性规划形式化模型，并实现了一个软件架构来验证该模型的正确性。试验结果表明该方法可以有效地应用到 40 条传输指令以内的基本块，并生成高质量的代码。

关键词：整数线性规划；指令调度；寄存器分配；传输触发结构；软件旁路

ILP-based TTA Code Optimization

HU Wei, ZHU Yong-xin, JIANG Lei
(School of Microelectronics, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 This paper provides Integer Linear Programming(ILP) formulation of optimal code generation for Transport-Triggered Architecture (TTA) architectures, which integrates instruction scheduling, multiplies register files allocation, global register allocation and software bypass. A framework has been implemented to testify this model and the experimental results show the algorithm is applicable up to 40 transport instructions per basic block and generates optimal code.

【Key words】 Integer Linear Programming(ILP); instructions scheduling; register allocation; Transport-Triggered Architecture(TTA); software bypass

1 概述

随着各种可定制处理器在嵌入式设计中的广泛应用，针对不同的目标处理器结构生成优化的代码成了编译器的一个艰巨而重要的工作。为了得到优化的代码，编译器需要考虑比以往通用处理器更多的优化可能性：指令集的选择，指令的调度和寄存器分配等，以及它们之间的顺序和整合问题。当先进行指令调度时，变量的生命周期也会发生变化，这可能会造成寄存器压力，从而引入过多的存储器访问指令；而当先进行寄存器分配时，可能会由于使用同一寄存器而引入一些假的依赖关系，影响调度结果。对这些问题的不同考虑可能会极大地影响到代码的效率以及对硬件资源的利用率。

整数线性规划(Integer Linear Programming, ILP)[1]算法为处理这些问题提供了一种统一的处理方式，可以将指令选择、指令调度和寄存器分配整合进一个优化问题。同时，还可以将功耗优化等问题也考虑进去。OPTIMIST是一个编译优化架构，它使用整数线性规划的方法[2]整合考虑了前面提出的 3 个问题。但它们针对超常指令字结构(Very Long Instruction Word, VLIW)，同时整数线性规划的方法也只是考虑了单一寄存器堆的情况，而且没有考虑全局寄存器的分配问题。它们能优化大约 15 条指令以内的基本块。

当前一个新的可定制处理器结构传输触发结构(Transport-Triggered Architecture, TTA)[3]正在逐渐被工业界使用。它的结构类似于VLIW，但是不像VLIW，它的每次操作都由一次或几次传输操作来完成。比如完成一次加法操作，TTA需要进行 3 次传输操作。2 次传输操作数到加法功能单元的操作数寄存器中，其中第 2 次传输触发加法单元的执行。在一定延时之后从加法单元的结果寄存器中再取回结果数。这使得TTA的各功能单元之间可以更好地共用传输带宽，但也加大了编译器调度的难度。然而拆分操作为代码优化带来了更多的可能性，比如软件旁路等，同时也为低功耗带来了好处。文献[4]针对TTA结构的指令调度和寄存器分配的顺序和整合问题进行了探讨。主要是使用一些启发式的算法，但这些算法并没有完全地解决代码优化问题。

本文使用整数线性规划算法针对 TTA 结构进行代码优化。给出了一个形式化的优化模型，局限在基本块内整合代码调度、多寄存器堆分配和全局寄存器分配等问题。同时还将 TTA 结构特有的软件旁路的优化可能性也考虑进去，并且实现了一个软件架构来验证模型的正确性。

2 代码优化问题的整数线性规划描述

2.1 代码和硬件描述

优化的范围是基本块，所以被优化的代码可以被描述成一个有向无环图(DAG)。形式化的描述如下：

一个基本块的 DAG 是一个六元组

$(T, T_{type}, R, E, E_{dep}, E_{type})$

其中，T 是操作节点集合；T_{type} 是操作类型集合；R 是操作数和结果节点集合；关系集合 $E \subset (T \times R \times E_{type}) \cup (R \times T \times E_{type})$ 表示操作和它使用的操作数及生成的结果之间的直接依赖关系；关系集合 $E_{dep} \subset E \times E$ 表示触发传输操作和操作数传输操作之间的先后关系以及存储器读写的依赖关系；$E_{type} = \{0,1\}$ 表示传输操作是触发操作还是一般的操作数传输操作，其中，0 表示是触发操作。对于 $\forall r \in R$，r 的入度 $IN(r)$=1，它们组

作者简介：胡 维(1982－)，男，硕士研究生，主研方向：应用可定制处理器的编译器技术，设计空间探索技术；祝永新，副教授、博士后；姜 雷，硕士

收稿日期：2007-12-26 **E-mail：**huwei@ic.sjtu.edu.cn

成一个有向无环图。为描述方便，定义 $E_{rt} \subset R \times T \times E_{type}$，$E_{tr} \subset T \times R \times E_{type}$，$E = E_{tr} \cup E_{rt}$；$GR$ 表示全局操作数或者结果数；LR 表示局部操作数或者结果数；$CTR \subset R$ 表示跳转指令使用的操作数，$R = GR \cup LR$。

而笔者针对的 TTA 结构的互联网络为全连接，并可以挂接多个寄存器堆，所以需要考虑的结构参数包括总线的个数、各个功能单元的个数和每个寄存器堆的大小和它们的读写接口的个数。定义 F 为功能单元类型集合，RF 为寄存器堆集合。

因为在 DAG 中每条边表示一条传输操作，所以以 TTA 结构为目标的指令调度实际上是调度边 $e \in E$。

2.2 优化问题的变量定义

$St_{t,i}$ 是布尔变量，当 $St_{t,i}=1$ 时，表示操作 $i \in T$ 在 t 时刻处于被执行状态。

$Se_{t,i,j,p,k}$ 是布尔变量，当 $Se_{t,i,j}=1$ 时，表示边 $(i,j,p) \in E$ 在 t 时刻被调度，同时这个传输操作使用的是寄存器堆 $k \in RF$。

$Sr_{t,i,j}$ 是布尔变量，当 $Sr_{t,i,j}=1$ 时，表示操作数或者结果数 $i \in R$ 在 t 时刻被分配了寄存器堆 $j \in RF$ 中的某个寄存器。

$Ser_{t,i,j,p,k}$ 是布尔变量，当 $Ser_{t,i,j,p,k}=1$ 时，表示在 t 时刻边 $(i,j,p) \in E$ 被调度，同时使用的操作数或者结果数被分配了寄存器堆 $k \in RF$ 中的某个寄存器。这个变量用来表示调度边时是否使用了寄存器堆。

λ 是一个整数变量，表示代码最后执行完的时间。

2.3 优化问题的参数定义

L_i 是非负整数参数，表示操作 $i \in T$ 的延时。

$Lp_{i,j,p,k,l,q}$ 是非负整数参数，表示依赖关系 $(i,j,p,k,l,q) \in E_{dep}$ 的延时要求。

Ot_i 表示操作 $i \in T$ 的操作类型，$Ot_i \in T_{type}$。

$Up_{i,j}$ 是布尔变量，当 $Up_{i,j}=1$ 时，表示操作类型 $i \in T_{type}$ 的操作应该由类型为 $j \in F$ 的功能单元来执行完成。

M_f 是非负整数参数，表示类型为 $f \in F$ 的功能单元的个数。

B 是非负整数参数，表示总线的个数。

Rp_i 是非负整数参数，表示寄存器堆 $i \in RF$ 的读接口个数。

Wp_i 是非负整数参数，表示寄存器堆 $i \in RF$ 的写接口个数。

RFs_i 是非负整数参数，表示寄存器堆 $i \in RF$ 的大小。

$T\max$ 是非负整数参数，表示使用启发式算法得到的可能的最大执行时间。

2.4 优化问题的约束条件

2.4.1 指令调度

对 TTA 而言，基本块中每条边 $(i,j,p) \in E$ 表示一条需要调度的传输操作。下式限制了每条边必须被调度 1 次，且仅被调度 1 次。

$$\forall (i,j,p) \in E, \sum_{t \in \{1,2,\cdots,T\max\}} \sum_{k \in RF} Se_{t,i,j,p,k} = 1 \tag{1}$$

而对于每个操作节点的入边都需要在该边依赖的前一个操作被完成后才能被调度。下式描述了这个约束。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall (l,i,o) \in Ert,$$
$$\forall (i,j,p) \in Etr, \forall (j,k,p) \in Ert, \tag{2}$$
$$\sum_{r1 \in RF} Se_{t,l,i,o,r1} + \sum_{\substack{tt \in \{1,2,\cdots,t+L_i-1\} \\ tt \le T\max}} \sum_{r2 \in RF} Se_{tt,j,k,q,r2} \le 1$$

对于与操作节点相邻的边，需要满足它们之间的先后关系。下式给出了这种关系的约束条件。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall (i,j,p) \in Ert, \forall (j,k,q) \in Etr,$$
$$\sum_{r1 \in RF} Se_{t,i,j,p,r1} + \sum_{\substack{tt \in \{1,2,\cdots,t+L_j-1\} \\ tt \le T\max}} \sum_{r2 \in RF} Se_{tt,j,k,q,r2} \le 1 \tag{3}$$

而对于 TTA 结构特殊要求的触发传输操作和操作数传输操作之间的先后关系，以及存储器访问的约束关系，由下式来统一给出。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall (a,b,p,c,d,q) \in E_{dep},$$
$$\sum_{r1 \in RF} Se_{t,a,b,p,r1} + \sum_{\substack{tt \in \{1,2,\cdots,t+Lp_{a,b,p,c,d,q}-1\} \\ tt \le T\max}} \sum_{r2 \in RF} Se_{tt,c,d,q,r2} \le 1 \tag{4}$$

2.4.2 寄存器分配

每条被调度的边都和一个操作数或者一个结果数有关。如果某个操作数或者结果数被分配到了某个寄存器堆，那么与之相关的边必须使用被分配的寄存器堆。式(5)~式(8)给出了这些限制。

$$\forall l \in RF, \forall (k,j,q) \in E_{tr},$$
$$\sum_{t1 \in \{1,2,\cdots,T\max\}} \sum_{(i,j,p) \in E_{tr}} Se_{t1,i,j,p}, l + \sum_{t2 \in \{1,2,\cdots,T\max\}} \sum_{(j,m,o) \in E_{rt}} Se_{t2,j,m,o,l} \le N \times \sum_{t3 \in \{1,2,\cdots,T\max\}} Se_{t3,k,j,q,l} \tag{5}$$

$$\forall l \in RF, \forall (j,k,q) \in E_{rt},$$
$$\sum_{t1 \in \{1,2,\cdots,T\max\}} \sum_{(i,j,p) \in E_{tr}} Se_{t1,i,j,p}, l + \sum_{t2 \in \{1,2,\cdots,T\max\}} \sum_{(j,m,o) \in E_{rt}} Se_{t2,j,m,o,l} \le N \times \sum_{t3 \in \{1,2,\cdots,T\max\}} Se_{t3,j,k,q,l} \tag{6}$$

$$\forall (i,j,p) \in Etr, \forall l \in RF,$$
$$\sum_{t1 \in \{1,2,\cdots,T\max\}} Sr_{t1,j,l} \le N \times \sum_{t2 \in \{1,2,\cdots,T\max\}} Se_{t2,i,j,p,l} \tag{7}$$

$$\forall (i,j,p) \in Ert, \forall l \in RF,$$
$$\sum_{t1 \in \{1,2,\cdots,T\max\}} Sr_{t1,i,l} \le N \times \sum_{t2 \in \{1,2,\cdots,T\max\}} Se_{t2,i,j,p,l} \tag{8}$$

每个节点 $j \in R$ 在被第 1 条与之相关的边调度之后被分配到某个寄存器堆，在与之相关的所有边被调度之后释放被分配的寄存器。下式描述了这种约束。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall (j,k,p) \in E_{rt},$$
$$\forall (i,j,q) \in E_{tr}, \forall l \in RF, \tag{9}$$
$$\sum_{t1 \in \{1,2,\cdots,t\}} Se_{t1,i,j,q,l} - \sum_{t2 \in \{1,2,\cdots,t\}} Se_{t2,j,k,p,l} \le Sr_{t,j,l}$$

每个节点 $j \in R$ 只有在它的入边被调度了之后，才能被分配寄存器。下式描述了这个约束。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall j \in LR, \forall l \in RF,$$
$$\sum_{tt \in \{1,2,\cdots,t\}} \sum_{(i,j,p) \in E_{tr}} Se_{tt,i,j,p,l} \ge Sr_{t,j,l} \tag{10}$$

同时如果一个节点 $j \in CTR$，那么认为它需要在第 1 条与之相关的边被调度之后被分配到某个寄存器堆直到基本块的代码执行完为止。下式给出了这个约束。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall j \in CTR, \forall l \in RF,$$
$$\sum_{tt \in \{1,2,\cdots,t\}} \sum_{(i,j,q) \in E_{tr}} Se_{tt,i,j,q,l} \le N \times Sr_{t,j,l} \tag{11}$$

在任何时刻 $t \in \{1,2,\cdots,T\max\}$ 被分配了的寄存器数目都不能超过该寄存器堆的大小。下式给出了这个约束。

$$\forall t \in \{1,2,\cdots,T\max\}, \forall i \in RF, \sum_{j \in R} Sr_{t,j,i} \le RFs_i \tag{12}$$

在笔者的这个最优化模型中，还给出了全局寄存器的分配约束条件，假定堆栈寄存器 sp 使用寄存器堆 0，并在整个代码的执行过程中都保持被分配的状态。用式(13)和式(14)来描述。

$$\sum_{i \in GR} \sum_{\substack{j \in RF \\ j=0}} \sum_{t \in \{1,2,\cdots,T\max\}} Sr_{t,i,j} = T\max \tag{13}$$

$$\sum_{i\in GR}\sum_{\substack{j\in RF\\ j\neq 0}}\sum_{t\in\{1,2,\cdots,T\max\}} Sr_{t,i,j}=0 \tag{14}$$

2.4.3 资源分配

每条被调度的边都和某个操作节点 $j\in T$ 相关，如果操作节点 j 的某个相关的边被调度，那么该节点被分配一个相关的功能单元，且直到与该节点相关的所有边被调度之后，被分配的相关功能单元被释放。同时，如果操作节点 j 的出度为 0，那么该节点必须在所有入边被调度后经过 L_j-1 后才能释放功能单元。式(15)~式(17)描述了该约束。

$$\forall t\in\{1,2,\cdots,T\max\},\forall j\in T,$$
$$\sum_{tt\in\{1,2,\cdots,t\}}\sum_{(j,k,p)\in E_{tr}}\sum_{(i,j,q)\in E_{rt}}\sum_{l\in RF}(Se_{tt,i,j,q,l}-Se_{tt,j,k,p,l})\leq N\times St_{t,j} \tag{15}$$

$$\forall t\in\{1,2,\cdots,T\max\},\forall j\in T,$$
$$\sum_{tt\in\{1,2,\cdots,t\}}\sum_{(k,j,p)\in E_{tr}}\sum_{(i,j,q)\in E_{tr}}\sum_{l\in RF}(Se_{tt,i,j,q,l}-Se_{tt,k,j,p,l})\leq N\times St_{t,j} \tag{16}$$

$$\forall t\in\{1,2,\cdots,T\max\},\forall(i,j,p)\in E_{rt},$$
$$L_j\times\sum_{k\in RF}Se_{t,i,j,p,k}\leq\sum_{\substack{tt\in\{t,t+1,\cdots,t+L_j-1\}\\ tt\leq T\max}}St_{tt,j} \tag{17}$$

在任何时刻 $t\in\{1,2,\cdots,T\max\}$ 只有当与节点 $j\in R$ 有关的边被调用同时节点 j 也被分配了寄存器，才表示这次传输操作与寄存器堆有关。式(18)和式(19)描述了这个限制。

$$\forall t\in\{1,2,\cdots,T\max\},\forall(i,j,p)\in E_{tr},\forall k\in RF,$$
$$Se_{t,i,j,p,k}+Sr_{t,j,k}\leq 1+Ser_{t,i,j,p,k} \tag{18}$$

$$\forall t\in\{1,2,\cdots,T\max\},\forall(i,j,p)\in E_{rt},\forall k\in RF,$$
$$Se_{t,i,j,p,k}+Sr_{t,i,k}\leq 1+Ser_{t,i,j,p,k} \tag{19}$$

在任何时刻 $t\in\{1,2,\cdots,T\max\}$ 被使用的总线不能超过总线的数目，也不能超过每种功能单元的个数，以及每个寄存器的读接口和写接口的个数。式(20)~式(23)给出了约束条件。

$$\forall t\in\{1,2,\cdots,T\max\},\sum_{(a,b,c)\in E}\sum_{i\in RF}Se_{t,a,b,c,i}\leq B \tag{20}$$

$$\forall t\in\{1,2,\cdots,T\max\},\forall f\in F,\sum_{\substack{i\in T\\ Up_{O_{i,}f}=1}}St_{t,i}\leq M_f \tag{21}$$

$$\forall t\in\{1,2,\cdots,T\max\},\forall k\in RF,$$
$$\sum_{(i,j,p)\in E_{tr}}Ser_{t,i,j,p,k}\leq Rp_k \tag{22}$$

$$\forall t\in\{1,2,\cdots,T\max\},\forall k\in RF,$$
$$\sum_{(i,j,p)\in E_{rt}}Ser_{t,i,j,p,k}\leq Wp_k \tag{23}$$

2.4.4 TTA 软件旁路优化

每个操作节点的入边在它依赖的操作节点完成操作之后和依赖的操作节点的出边被调度之前有软件旁路优化的可能性。式(2)、式(3)、式(18)、式(19)、式(22)、式(23)联合起来隐式地给出了 TTA 软件旁路优化的可能性。

2.4.5 优化目标

代码最后执行完成的时间 λ 应该大于或等于所有边 $(a,b,c)\in E$ 被调用的时间，同时如果操作节点 $i\in T$ 的出度为 0 则代码最后执行完成的时间 λ 也必须大于或等于操作节点 i 被执行的时间。下面的不等式给出了约束。

$$\forall(a,b,c)\in E,\sum_{t\in\{1,2,\cdots,T\max\}}\sum_{i\in RF}(t\times Se_{t,a,b,c,i})\leq\lambda \tag{24}$$

$$\forall i\in T,\forall t\in\{1,2,\cdots,T\max\},$$
$$(t+L_i-1)\times(St_{t,i}-\sum_{(i,j,p)\in E_{tr}}\sum_{tt\in\{1,2,\cdots,T\max\}}\sum_{k\in RF}Se_{tt,i,j,p,k})\leq\lambda \tag{25}$$

优化目标是使 λ 最小。

3 优化模型实现架构

笔者实现了一个架构来验证优化模型的正确性。在实现的架构中，使用 MACHSUIF 编译器架构生成中间代码，并使用 XML 描述硬件结构。

XML 描述的硬件结构使用 Xerces C++ parser 解析，然后输入到 MACHSUIF 中。MACHSUIF 根据得到的硬件信息使用 GNU 数学建模语言 GMPL 输出源程序每个基本块的数据描述，再使用 GNU 整数线性规划软件包中的运算器 gplsol 计算得到最后的 TTA 汇编代码。整个流程如图 1 所示。

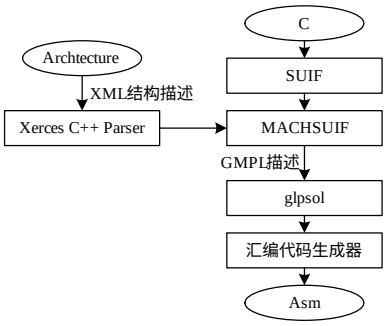

图 1 优化代码生成流程

4 初步试验结果

选择 2 个目标处理器结构：

(1)结构 a：4 条总线；1 个存储器存取单元；1 个整数算术运算单元；1 个整数乘法单元；1 个逻辑运算单元；1 个有 2 个读接口 2 个写接口大小为 8 个字的寄存器堆以及 1 个有 1 个读接口 1 个写接口大小为 16 个字的寄存器堆。

(2)结构 b：在结构 a 基础上增加 2 条总线和 1 个整数算术运算单元。

笔者编译了一些随机生成的基本块，得到的结果如图 2 所示。使用的主机是 Althon 1 GHz, 512 MB 内存。运行环境是在 VMware Workstation 虚拟环境，192 MB 内存。试验结果表明在典型处理器结构 a 下算法能解决大约 40 条传输指令以内的基本块，而在结构 b 下优化速度会更快，同时能处理的指令也更多。

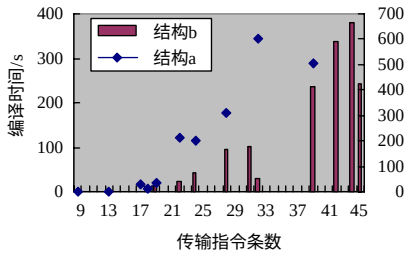

图 2 优化编译结果

5 结束语

本文实现了一个使用整数线性规划算法来优化TTA代码的软件架构，将指令调度、多个寄存器堆分配、全局寄存器分配和TTA软件旁路等问题整合到一个整数线性规划模型中，同时也以此验证了优化模型的正确性。得到的试验结果表明在典型处理器结构下算法能解决大约 40 条传输指令以内的基本块，同时随着资源的增多能解决的指令也会增多。笔者将在未来的工作中针对TTA代码优化问题进行改进，包括DAG的预处理和分析、冗余约束的去除、整数线性规划模型的改善和分支定界的应用定制[5]等，整数规划算法将能很好地解决TTA的代码优化问题。

（下转第 224 页）