类比推理协处理器的实现*

沈旭昆 王双全 王军玲 李 波 赵沁平

(北京航空航天大学计算机系 北京 100083)

摘要 类比推理技术对于开发高智能的AI系统十分重要,普通的面向数值处理的计算机进行类比推理效率往往很低,严重影响其研究与应用.本文介绍了一种高效的嵌入式类比推理协处理器系统ARC(analogical reasoning coprocessor).由于ARC采用了符合类比推理机制的处理器体系结构和加速策略,并采用先进的器件实现,其推理速度比普通的PC机有大幅度的提高.

关键词 类比推理,协处理器,体系结构,加速策略.

中图法分类号 TP302

使计算机系统具有更高的智能水平一直是计算机科学研究领域中的重大课题,也是众多计算机科学工作者不懈追求的目标.要进一步从实质上提高计算机的智能水平,就应当使计算机系统具有人类所经常使用的另一类典型的逻辑思维能力,即类比联想.类比推理是由于认识到新情况(New Situation)和已知情况(Know Situation)在某些方面相似,从而推出它们在其它相关方面也相似的推理形式.针对当前类比推理研究中存在的问题,我们已提出了一种比较完整的类比推理理论,给出了相应的计算模型和实现算法,并构造了实验性类比推理系统BHARS.[1]

通过对类比推理计算模型的分析,可以得出类比推理过程具有以下特点:

(1) 大量的信息查找和类型判断:在类比推理的联想和求精阶段,需要经常查询对象、谓词、命题和情况的基本描述及某种特性.此外,类比推理过程常常需要判断某个符号是对象、谓词、命题还是函数.

(2) 结构型数据多:类比推理以情况为基础,情况通过命题来描述,类比过程的中间和最终结果常以表或集合形式表示.因此类比推理的数据大多数是结构型的.一般来讲命题和函数的长度变化范围不大,而表的长度变化范围很大.

(3) 过程调用和返回频繁:由于类比推理分为联想、求精、匹配和转换等相对独立的不同阶段,每个阶段又分为不同的过程,因此在类比推理时过程调用和返回十分频繁.

在实验测试和应用中我们体会到:由于类比模型固有的计算复杂性,仅靠改进模型很难大幅度提高处理速度;使用面向数值计算的传统计算机来进行类比推理效率很低,制约了类比推理技术的应用.因此迫切需要采用专用部件和硬件实现来提高处理效率.我们在"863"智能计算机主题的支持下研制了适应类比推理机制的协处理器ARC(analogical reasoning coprocessor),大幅度提高了类比推理的执行效率.

1 指令系统和数据结构

为了提高数据处理速度和节约存储空间,ARC处理的数据采用了带有标记的数据,即每个数据字包括类型、结构型数据压缩存储信息和数据3个字段.这种自定义表示方法符合类比推理机制,为类比推理的高效执行提供了良好的支持.[2]

* 本文研究得到国家863高科技项目基金资助.作者沈旭昆,1965年生,讲师,主要研究领域为类比推理,虚拟现实,中文信息处理.王双全,1974年生,硕士生,主要研究领域为类比推理.王军玲,女,1960年生,博士生,工程师,主要研究领域为类比推理.李波,1966年生,博士,副教授,主要研究领域为类比推理,图象处理.赵沁平,1948年生,博士,教授,博士导师,主要研究领域为类比推理,虚拟现实.

本文通讯联系人:沈旭昆,北京100083,北京航空航天大学计算机系

本文1997-04-18收到原稿,1997-07-25收到修改稿

1.1 带标记的的数据表示

ARC采用了带标记的数据表示形式,数据字长为24位,其中高8位为标记字段,低16位为数据字段.如下所示:

GC(1-bit)	CC(2-bit)	TYPE(5-bit)	Data(16-bit)

GC:废料收集时使用.

CC:采用CDR编码支持结构型数据的压缩存储,指明其存储信息.

TYPE:指明该数据字的数据类型,共有NIL(空表)、INT(整数)、SYM(符号)、VEC(向量)和PTR*(指针)5种.

DATA:数据值,目前支持16位的数据字.

其中带*标记的为结构型数据类型,ARC用一片连续存放的数据字表示它们.实际上结构型数据的值是指针,指向该数据在Heap区的地址,如图1所示.

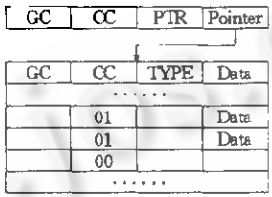

图1　结构型数据在ARC上的表示

1.2 结构型数据的压缩表示

类比推理以处理结构型数据为主,因此,它们的存储方式合理与否直接关系到能否有效利用和管理存储空间.结构型数据的一般表示方法有3种:连续存储方式、链接存储方式和CDR编码方式.

链接存储方式的基本单位是由相邻的两个单元构成.第1个单元存放表的元素,第2个单元存放指向表的剩余部分的指针,结构型数据可表示成由其函数和变元构成的表.一般情况下,表的物理表示中有近一半的空间用于存放链接指针,不但浪费了存储空间,而且对于非表元素的存取,有近一半的存取时间用在链接指针访问上.

在连续存储方式中,结构型数据表示成由一片连续单元构成的块.这种方式虽然节省了存储空间,但对其操作的指令序列要比链接方式的指令序列复杂得多,相应地也就需要更多的程序空间和执行时间.由此看来,这两种方式都有较大的缺点.

目前大多数处理知识的专用机都采用CDR编码方式,它将结构型数据顺序存放在连续的存储空间中,通过使用特殊的标记字段来指示CDR域,ARC采用的正是这种方式.ARC把对象、谓词、命题、函数等结构型数据和其参量存放在连续单元中,尽可能将表结构中的表元素连续存储,只有在必要时才分散存放,并且用CDR编码连接.这种方式的操作指令序列与链接方式的一样简单,而它表示结构数据所需的空间与连续存储方式相同,只有在最坏情况下才与链接方式所需空间一样.由于结构型数据元素可能连续存放,也可能分散存放,下一个元素是在下一个相邻单元还是在另一个非相邻的单元中,需要通过动态测试CDR位决定,这样必然降低执行速度.但ARC通过专用的字段抽取和判断硬件,明显地缩短测试时间,从而提高结构型数据的处理效率.

关于CDR编码,目前已出现多种,较著名的有MIT Lisp机的编码方式、Symbolics 3600的编码方式和Xerox Lisp机的编码方式等.在分析这些编码方案的基础上,我们提出了一种适合于类比推理过程的有效编码方案,具体编码如下:

CC=00(CC_NIL):该数据字是表或其它结构型数据的尾项.

CC=01(CC_NEXT):该数据字是表或其它结构型数据的项,并且下一个数据字是该表的下一项.

CC=10(CC_INDIR):该数据字是表或其它结构型数据的项的间接指针,并且下一个数据字是该表的下一项.

CC=11(CC_CDR):该数据字是表或其它结构型数据的项的间接指针,并且该数据字是表的尾项.

1.3 指令系统

ARC共有46条指令,采用32位固定字长,分为4个字段.bit23～bit31为操作码字段,其余3个字段为操作数字段.每条指令可以有0～3个操作数,只有寄存器寻址和立即数两种寻址方式.采用固定长度指令及减少寻址方式有利于简化硬件逻辑.在ARC指令系统中除了通用的运算、数据传送指令外,还有许多适应类比推理机制的表处理类、联

想类和匹配类指令.这主要有以下3方面考虑:

(1) 在整个类比推理过程中存在大量的表操作,例如取表头和进行表间运算等复杂操作.在指令集中精心地设计它们,使其功能强、速度快,可以大大加快整个类比推理的执行.

例1:list_def $R1$, $R2$

指令格式:BiChar

指令说明:该指令完成寄存器$R1$中表与寄存器$R2$中表的差运算,结果元素依次压入临时栈中,元素个数放入$R0$中.

(2) 在类比推理的联想和求精阶段,需要经常查询对象、谓词、命题和情况的基本描述以及它们的某种特性.增加一些特殊的简单查询指令是很有必要的.

例2:type R

指令格式:UniChar

指令说明:本指令用于查询寄存器R中数据的类型,将值放入$R0$.

(3) 在类比推理的匹配阶段,因为要合并命题映射形成极大映射,从而需要多次判断命题映射与映射间的一致关系.为加快匹配速度,我们用位向量表示映射,其第i位为1表示该映射包括第i个命题映射.因而匹配类指令包括位向量的处理.

例3:vec_and $R1$,$R2$,$R3$

指令格式:TriChar

指令说明:对$R1$与$R2$指示的位向量进行对应位的与操作,结果放在$R3$指示的位向量中.$R1$,$R2$和$R3$指示的3个位向量长度相同.

这些指令一般都很复杂,在每条指令内包含有大量的数据访问和处理条件判断等操作,它们的执行时间占整个处理时间的60%以上,因此,如何处理表处理类指令、联想类指令和匹配类指令是提高处理器性能的关键之一.

2 存储结构及系统构成

类比推理在执行过程中对存储器的访问十分频繁,因此数据的访问速度(吞吐率)是影响推理速度的一个重要因素,为突破这一瓶颈,ARC采取了以下措施:

(1) 数据与程序存储器相互独立.当代通用计算机的程序和数据一般拥有共同统一的主存储器,但采用相互独立的指令Cache和数据Cache,这样有助于消除取指令与数据访问引起的存储器资源冲突.出于成本和实现环境限制的考虑,ARC的存储体系中没有设置Cache,而采用数据与程序存储器相互独立的策略来保证ARC的性能不受损害.

(2) 设置高速临时栈.经过分析,在根据类比推理模型建立的模拟系统上各种典型实例的运行结果,我们发现:在许多关键指令中需要将几个表拆开并重新组合.为此,ARC设置了一个临时栈Temp,用于存放操作的中间结果.对临时栈和数据区Heap的访问可以同时进行,容易实现指令内部的并行及流水操作.

(3) 面向寄存器操作.ARC的指令系统只有寄存器寻址和立即数两种寻址方式,有近半数的指令为双寄存器或三寄存器操作指令.为了能对这些寄存器并行访问,我们采用双端口高速静态RAM来实现寄存器堆.

(4) 参数寄存器采用寄存器窗口技术.参数寄存器区分成64组,每次调用函数后,将使用新的寄存器组,而不含覆盖上一层函数的参数寄存器,这样就避免了频繁的压栈、弹栈操作,节省了大量时间.

(5) 基于以上理由,同时考虑到性能/成本比,我们将Heap,Symbol区物理地合并在一个存储体中,宽度为24位;将寄存器堆、临时栈Temp、返回栈Ret和参数寄存器区(ARA)物理地合并在一个高速双端口SRAM中;程序存储器独立于以上各存储器.

(6) 为了充分发挥多存储体的作用,系统采用了多总线结构,其中包括两条数据地址总线,两条数据总线,一条指令地址总线(IA-Bus),一条指令总线(I-Bus).

ARC由指令执行单元(IEU)、数据存储器(Heap)、符号表区(Symbol Area)、通用寄存器(General Registers)、参数寄存器区(ARA)、临时栈(Temp-Stack)、函数返回地址栈(Return-Stack)、程序存储器区(Code Area)、中断机构和ARC-Host接口单元(Interface Unit)等组成.

ARC通过接口单元(IU)和主机的系统扩展总线ISA连接,ARC的微程序、程序加载、运行状态监测和运行结果均通过接口实现.

3 控制方式

ARC的指令系统比较复杂,所能使用的设计工具和器件性能有限,因此,我们采用微程序控制方式.ARC采用水平编码方式的微指令,每条微指令由多个控制字段组成,每个字段直接控制某个功能部件,完成一个微操作.基于微程序的指令控制器在一个周期内只需发出一条微指令,而我们可以通过将尽可能多的微操作组合在一条微指令中来提高该指令的执行速度,进而提高程序的执行速度.这和VLIW技术非常相似,因此,从某种意义上来说,VLIW是微程序技术的一种延伸.[3]ARC微操作的最大可并行度由以下设计策略得以保证:

(1) 采用多存储器多总线结构,最多允许在一个周期内有3个访问数据存储单元的微操作.

(2) 在ARC内部尽量减少关键路径的延迟,采取直接连接方式,并设置若干缓存器,支持微指令流水.

(3) 除了通用运算部件以外,还根据数据结构特点设置了专门的处理标志字段的功能部件.

(4) 针对关键指令中有大量条件判断,设置了专门的快速条件处理部件.

ARC有24类共90余条微操作.每类微操作占用一个微指令字段,即一条微指令最多可以有24个微操作.我们对所有指令编制了微程序,统计表明,每条微指令平均有6.17个微操作,而对于关键微指令可以达到16个微操作并行.如此高的并行度是ARC执行类比推理速度比通用处理机要快得多的主要因素.

4 流水线

流水线是当代处理器采用的主要技术,其基本思路是通过使多条指令重叠执行来提高处理程序的速度.[4~8]ARC中也采用了流水线技术.由于ARC采用了微程序控制方式,所以不仅有指令一级的流水线,也有微指令(微操作)一级的流水线.

4.1 指令及微指令流水线

在处理器的设计中,复杂指令的处理对指令流水线的高效运转是至关重要的.为了满足复杂指令的要求,一般有两种方法.一种是对复杂指令采用专门的复杂组合逻辑,并降低时钟频率,使复杂指令在一个周期内可以完成.显然,这样对其它简单指令是不公平的,处理器的性能下降较大.另一种方法是增加流水线的级数,即将复杂指令分为几个简单的阶段完成,这样,时钟频率可以不降低,但可能存在以下问题:

(1) 如何从不同的复杂指令中分解出公共的小阶段以组成不同的流水级.

(2) 如何处理因每次处理数据不同而执行周期不同的指令,例如串操作指令.

(3) 如何处理对于简单指令来说多余的流水级.

为了解决这些问题,以组合逻辑控制方式实现的处理器的成本和复杂度必将大大增加.我们采取的策略是,在ARC指令一级只使用三级简单流水线结构,分别是取指令(IF)、指令译码(ID)、执行指令(IE).而在执行指令阶段(IE)转入微指令流水线MEX-Pipelinc.此后,指令流水线继续流经IF,ID级,等待上一条指令的微指令流水线完成后立刻进入当前指令的微指令流水线,如图2所示.

图2 ARC的指令流水线

从图2看,似乎微指令的执行是一种顺序执行方式.实际上,同指令流水线一样,微指令的执行也是一种重叠并行的方式.在这个阶段,功能部件(即处理数据的部件)和存储资源得到了充分的利用.

4.2 指令预取及其流水化

为了维持指令流水线的高效运转,需要在一个周期内将一条指令读入指令寄存器.由于ARC采用微程序控制方式,在取指令的同时还需形成微程序的入口地址.我们采用的程序存储器是访问周期为45ns的静态存储器,宽度为32位.ARC的工作时钟为16MHz.如果执行机构直接从程序存储器中读取指令,从程序地址形成到微程序的入口地址产生的时间大于一个时钟周期,因此有必要在程序存储器与指令寄存器之间增加一级缓存机构.为此,我们在ARC中设置长度为3的指令预取FIFO队列,在执行指令的同时,可以从程序存储器中读出指令进行预处理,并在必要时将它放入预取队列中.预取队列的访问时间为20ns左右,从队首读出一条指令到微程序入口地址形成只需45ns的时间,小于一个时钟周期,满足要求.

5 对转移指令的处理策略

ARC 指令系统中包括 4 条转移指令：TRUE_JUMP，JUMP，CALL 和 RET. TRUE_JUMP 为条件转移指令，TRUE_JUMP，JUMP 和 CALL 指令是绝对地址转移指令，目标地址从指令本身就可以得到. 通过在模拟系统上对 11 个典型事例进行分析，发现各转移指令执行的平均频率如下：

指令	CALL/RET	JUMP	TRUE_JUMP
频率(%)	5.72	8.35	22.10

所有转移指令平均执行频率之和高达 36.17%. 通过模拟还发现，其它指令的平均执行周期(CPI)为 3.3 左右，转移指令造成的周期损失为 3. 这样，在处理器运行的整个过程中，功能部件的空闲时间与有效工作时间之比为 3×36.17%：3.3×63.83%≈0.52：1.00，浪费是非常惊人的，因此必须采取措施消除或减少这种浪费.

5.1 零周期处理策略

转移指令的执行并不需要处理数据的功能部件直接参与，可以考虑在执行其它指令的同时对转移指令进行处理. 这样看起来，似乎执行转移指令是不花时间的，所以称之为“零周期转移”策略.[4]图 3 描述了在 ARC 中“零周期转移”的具体实现策略.

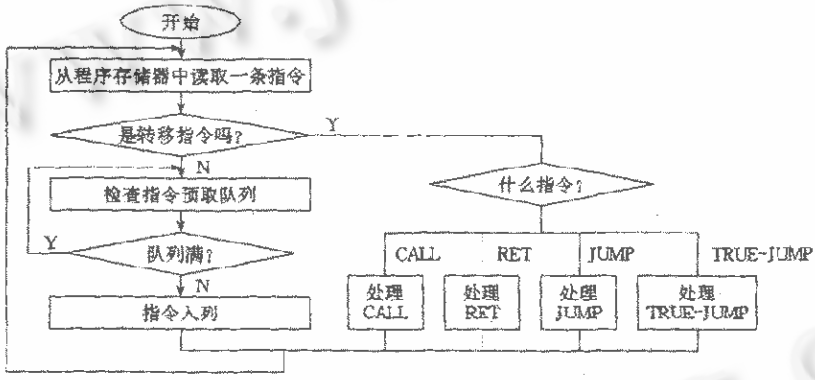

图3 “零周期转移”实现流程图

当从程序存储器中读出 JUMP 指令时，将绝对目标地址写入程序地址寄存器，继续读取下一条指令进行处理. JUMP 指令本身不放入预取队列.

对于 RET 指令，其目标地址在函数返回栈栈顶，由于返回栈与数据存储器是公共的，因此如果此时从返回栈中读出目标地址，就有可能发生资源冲突. 对于 CALL 指令，要向返回栈中写入函数返回地址同样会引起资源冲突，为此我们采用以下方案：

在 ARC 芯片内设置一个长度为 6 的返回栈顶队列，它的队尾是返回栈的实际栈顶，与之相区别的在存储器中的返回栈的栈顶称为伪栈顶. 当从程序存储器中读出 CALL 或 RET 指令时，将它们放入指令预取队列，并将返回地址暂时保存于栈顶队列或从中取出返回地址，以便继续读取下一条指令. CALL 指令或 RET 指令将被当作普通指令一样执行，但由于其后续指令已从程序存储器中读出，因此执行 CALL 或 RET 指令时只要花一个周期的时间在返回栈与栈顶队列之间进行数据交换及修改栈指针就行了. 这样可以将 CALL 或 RET 指令引起的周期损失减小至原来的三分之一.

5.2 推测式处理策略

通过在模拟系统上处理典型事例，我们发现，对于条件转移指令，转移不发生的概率高达 85.53%，因此在 ARC 中采用了“预测转移不发生”的这种简单有效的推测式处理策略. 在预取过程中遇到条件转移指令时，假定转移不发生，选择顺序执行的分支读入指令，并保存转移分支的地址. 随着预取队列的推进，当其前一条指令执行完时，若转移条件不成立，说明预测是正确的，继续执行指令；否则，清除该预测及之后的所有队列单元(置为空操作指令)，并选择转移分支重新建立指令队列.

6 专用芯片设计与系统实现技术

由于现成的芯片难以满足ARC板设计要求，因此，ALU指令执行单元、程序及堆栈控制器等逻辑使用美国ALERA公司的Flex 8000系列芯片实现. Flex 81188系列芯片属新一代可编程逻辑器件，采用SRAM技术实现，有24 000个门电路，其中可用门12 000. 其连续式互连结构利用同样长度的金属丝实现逻辑单元的互联，消除了延时上的差异，同时提供大量快速通道和级联链进位链. 其优越的性能和功能可以很方便、快速地实现系统所需要的算术、逻辑、运算及控制功能. ARC板的ALU、各种堆栈控制器、指令预取队列控制器等绝大部分逻辑功能仅采用两块Flex81188芯片实现，使得印刷板的布局、走线、调试容易，大大缩短了设计周期. Flex81188芯片的在线配置和重配置使得ARC板的升级成为可能. 绝大部分功能集中在两块芯片中实现，这是ARC板的一大特色.

7 结 论

在ARC中，我们精心设计指令系统和数据结构，采用了针对类比推理机制的各种加速策略，实现了以下目标：

(1) 在硬件支持下大幅度提高类比推理效率.

(2) 有效利用和管理存储空间.

(3) 简化硬件逻辑，降低成本，提高硬件资源的利用率，这与当代通用处理器设计的基本原则是一致的.

(4) 增加指令系统灵活性，采用最新硬件技术，使ARC板的升级成为可能.

测试表明，ARC在16Mhz时钟下运行时，进行类比推理的速度是486DX2-80的39.3倍，是Pentium 90的10.9倍(在PC机上运行的都是经过速度优化编译的32位Windows 95应用程序). 随着时钟频率的进一步提高，系统效率还可得到进一步改善.

ARC为类比推理的应用奠定了良好的基础，它既可作为独立的类比问题求解系统，又可嵌入其它决策支持系统、专家系统、数据库、知识库与事实库等，作为知识获取与问题求解系统. 目前我们正在开展类比推理的应用研究，相信随着计算机应用领域的进一步扩大和担负更具智能性的任务，对其要求将会更高，需求趋势将更为显著.

参考文献

1 李波. 类比推理理论及实现研究[博士论文]. 北京航空航天大学，1993
(Li Bo. Research on theory and implementation of analogical reasoning[Ph. D. Thesis]. Beijing University of Aeronautics and Astronautis, 1993)

2 王军玲. 类比推理抽象机的设计和模拟[硕士论文]. 北京航空航天大学，1995
(Wang Jun-ling. Design and implementation of analogical reasoning virtual machine [M. S. Thesis]. Beijing University of Aeronautics and Astronautics, 1995)

3 Moreno J H. Dynamic translation of tree-instruction into VLIWs. IBM Research Report, July 1996
http://domino.watson.ibm.com/library/CyberDig.nsf/8193.ps.gz

4 李晓明，程旭. 现代处理器的核心技术与基本结构. 北京：电子工业出版社，1995
(Li Xiao-ming, Cheng Xu. Key techniques and fundamental architecture of modern processor. Beijing: Electronic Industry Press, 1995)

5 沈绪榜. RISC及后编译技术，北京：清华大学出版社，1994
(Shen Xu-bang. RISC and post-compilation. Beijing: Tsinghua University Press, 1994)

6 何德书等. RISC结构——当代计算机发展的最新技术. 北京：电子工业出版社，1992
(He De-shu *et al*. Architecture of RISC—— up-to-data technique of modern computer. Beijing: Electronic Industry Press, 1992)

7 Kai Hwang 著，王鼎兴等译. 高等计算机系统结构. 北京：清华大学出版社，1995
(Kai Hwang. Advanced computer architecture. Beijing: Tsinghua University Press, 1995)

8 Bayko J. Great microprocessor of the past and present. University of Regina, Canada, Oct. 1996
http://infopad.eecs.berkeley.edu/CIC/archive/cpu_history.html

Implementation of an Analogical Reasoning Coprocessor

SHEN Xu-kun WANG Shuang-quan WANG Jun-ling LI Bo ZHAO Qin-ping

(*Department of Computer Science Beijing University of Aeronautics and Astronautics Beijing* 100083)

Abstract Analogical reasoning technique is significant to the development of AI systems. Computing systems based on conventional architecture are very inefficient in doing analogical reasoning and this obstruct the research and application of analogical reasoning. To solve this problem, a high-performance embedded analogical reasoning systems called ARC(analogical reasoning coprocessor) is presented in this paper. Since ARC has a specialized architecture with special speedup strategy to support analogical reasoning mechanism, and is implemented with advanced integrated devices, it's much faster than conventional computing systems in doing analogical reasoning.

Key words Analogical reasoning, coprocessor, architecture, speedup strategy.