一种模糊推理机的VLSI实现：单片专家系统

Masaki Togai, Hiroyuki Watanabe

摘 要

我们给出了一种能处理不确定性和近似推理的推理机制的VLSI实现。为了有效和实时应用，设计是以模糊集合论中的"极大极小运算"为基础的，这种推理机制能处理不精确和不确定的知识，从而能获取专家知识并模拟其推理过程。我们采用定制 CMOS 技术实现了这种推理机制，并突出了简洁性、可扩充性和效率。定时模拟表明这种推理机每秒能完成约80,000次模糊逻辑推理。这类推理机的潜在应用包括指挥与控制领域中的实时决策和机器人系统的自适应命令生成等。

决策和推理过程中用到的信息可能是不确定、不精确，或甚至是含混、不完整的。由于人类专家给出的知识往往是不确定或不精确的，在基于规则的专家系统中，不确定性推理方法就越来越显得重要了。

本文将报告一种能处理不确定性、完成近似推理的推理机结构的的设计和相应的VLSI芯片设计。为了处理不确定性，我们采用了基于模糊集合论的模糊逻辑[12]；我们提出了一种适于硬件实现的推理结构，并在一个定制VLSI芯片上予以实现(该片子包含两个简单的单元：计算极大元素和极小元素的线路)。这种设计可以扩充以处理规则数目很大的情况，而且推理的速度与规则数量无关。我们将阐述这种设计的简洁性、可扩充性和效率。

模糊逻辑已在几个专家系统中[5,13]得到成功运用，例子之一就是 CATS。这是一个内燃机车诊断系统[2]，目前约有530条规则，很快就将增加到1200条左右。在指挥与控制领域[6]的实时决策中，模糊推理也被用于为截击导弹选择最佳导航算法。这些例子表明需要一个高效推理机以处理大的规则集和实时应用。

一、模糊逻辑导论

为了方便读者，在这一节中将模糊集合论及语言变量方法的有关方面简介如下，更详细的内容见参考文献[12]。

1.1 符号和术语

符号U代表论域空间，它是由任意一组对象或数学结构形成的集合。如果A是U的一个有穷子集，其元素为$u_1, u_2, \cdots, u_n$，则A表示为

$$A=\{u_1, \cdots, u_n\} \qquad (1)$$

U的一个有穷模糊子集A是一个序对集：

nguage", Addison-Wesely, 1986.
16. D. A. Moon, "Object-Oriented Programming With Flavors", OOPSLA'86 Proceedings, Sept. 1986.
17. A. Yonezawa等, "Object-Oriented Concurrent Programming in ABCL/1", OOPSLA'86 Proceedings, Sept. 1986.
18. W. G. olthoff, "Augmentation of Object-Oriented Programming by Concepts of Abstract Data Type Theory: The ModPascal Experience", OOPSLA'86 Proceedings, Sept. 1986.
19. J. A. Goguen等, "Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types", In Current Trends in Programming Methodology IV, Prentice-Hall, 1987, pp. 80-144.

$$A=\{(u_i,\ \mu_A(u_i))\},\quad u_i\in U,\qquad (2)$$

其中$\mu_A(u_i)$代表隶属度(或隶属函数)，如果$\mu_A(u_i)\in\{0,1\}$，则模糊子集理解为非模糊子集或普通子集，函数$\mu_A(u_i)$也就成了二值布尔函数。但是，除非特别声明，我们假定$\mu_A(u_i)$的值位于[0、1]区间，并且值0表示无隶属关系，值1表示完全隶属。

1.2 语言变量和模糊子集

我们用模糊子集表示语言变量。非形式化地说，语言变量L是一种取值为自然语言或其子集中的词或句子的变量。如果"年龄"作为一个语言变量，则它的词集T(年龄)可以是：

$$T(\text{年龄})=\{\text{年轻，老，非常年轻，不年轻，非常老，非常非常年轻，多少有点年轻，}\cdots\}\qquad (3)$$

其中每个词都表达为论域空间U，比方说$U=[0,\ 100]$，上的一个模糊子集。例如，给定T(年龄)中的"年轻"和"老"两个模糊子集，我们就可以构造其它模糊子集，比如"非常年轻"、"多少有点年轻"以及"不非常年轻"，如图1所示。

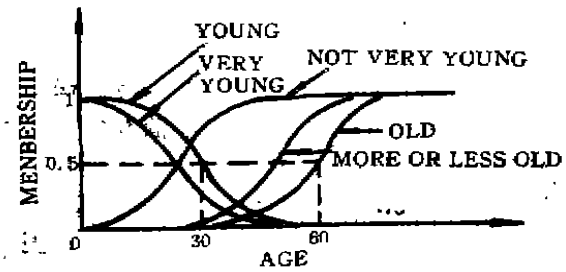


图1 由模糊集表示的各种语言变量值

MORE OR LESS OLD：多少有点老
NOT VERY YOUNG：不非常年轻
VERY YOUNG：非常年轻
MEMBERSHIP：隶属度
YOUNG：年轻
OLD：老
AGE：年龄

1.3 模糊关系

在传统的命题演算中，表达式"如果A则B"被写成$A\to B$，其中A及B是命题变元，$\to$是一个连词，定义为：

$$A\to B\equiv\overline{A}\cup B,\qquad (4)$$

其中$\overline{A}$表示非A。

在我们的方法中起重要作用的一个更一般的概念是所谓模糊条件语句：如果A则B，或简写为$A\to B$，其中A(前件)和B(结论)是模糊子集而不是命题变元。如果$A\in U$且$B\in V$，则从A到B的一个关系R就是笛卡尔积集$U\times V$上的一个模糊子集，于是，可用模糊关系R代表形如"如果U是A，则V是B"的条件语句，且定义如下：

$$\mu_R(u,v)=\min(\mu_A(u),\mu_B(v)),\quad u\in U,\ v\in V.\qquad (5)$$

定义模糊关系的方法有很多，这里只介绍最常用的一种。

1.4 推论组合规则

如果R是U到V的一个模糊关系，x是U上的模糊子集，则由x导出的、V上的模糊子集y表示如下：

$$y\equiv x\circ R\qquad (6)$$

且定义为

$$\mu_y(v)=\max_{u\in U}\min(\mu_x(u),\mu_R(u,v)).\qquad (7)$$

值得注意的是，当$R=A\to B$且$x=A$时，可得到如下的恒等式：

$$y=x\circ(A\to B)=B,\qquad (8)$$

此式可看作是假言推理的推广。

二、模糊推理机的逻辑结构

在这一节，我们提出一种便于硬件实现的推理机制。这一机制建立在模糊蕴含，或模糊规则，以及推论组合规则等概念之上。一条模糊规则定义为观察到的事实(前件)与动作(结论)之间的一个关系。对于一个给定的模糊规则集，动作是从一些观察到的事实和由规则构成的模糊关系推导出来的。

如果A和B是代表变量的模糊子集，它们分别定义在论述某方面问题的空间U和V上，则形如"如果A则B"的判定规则可用关于A和B的二元隶属函数定义如下：

$$\mu_{A\to B}(u,v)=f_{\to}(\mu_A(u),\mu_B(v)),\quad u\in U,\ v\in V\qquad (9)$$

更准确地说，设$A_1,A_2,\cdots A_N$是U上的模糊子集，$B_1,\ B_2,\ \cdots,\ B_N$是V上的模糊子集，则

模糊关系可由如下的一些规则定义：

规则1：　　如果A_1则B_1

规则2：否则如果A_2则B_2

⋮

规则N：否则如果A_N则B_N

这样，所有规则都通过连词“否则”组合在一起，从而得到一个总的模糊关系R。

R_i是从规则i及语言变量A_i、B_i构造出来的模糊关系，连词“否则”用函数f_{ELSE}代表，则总的关系R定义为，

$$\mu_R(u,v)=f_{ELSE}\{\mu_{Ri}(u,v)\}$$
$$=f_{ELSE}[f_{\rightarrow}(\mu_{A_i}(u),\mu_{B_i}(v))]$$
$$\text{for}\quad i=1,\cdots,N. \qquad (10)$$

由于关系R必须由规则1或规则2或…或规则N组成，所以为了导出一个总的模糊关系R，连词f_{ELSE}必须解释为“或”连词，这样，总的关系R表达为：

$$R=\bigcup_i R_i=\max_i f_{\rightarrow}(\mu_{A_i}(u),\mu_{B_i}(v))$$
$$\text{for}\quad i=1,\cdots,N. \qquad (11)$$

假设我们得到一个模糊观察值A'及一个总关系R，则合成的动作B'可由推论组合规则导出，即为：

$$B'=A'\circ R \qquad (12)$$

B'的隶属函数值通过(7)式中定义的所谓“极大极小操作”计算而得：

$$\mu_{B'}(v)=\max_{u\in U}\min(\mu_{A'}(u),$$
$$\mu_R(u,v)). \qquad (13)$$

下面我们再继续前面的讨论，提出一种适合硬件实现的模糊推理机结构。

让我们考虑一组N条规则中的第i条规则。给定一个观察事实A'及一条规则R_i，动作B'_i被推导和定义如下：

$$B'_i=A'\circ R_i;\quad A'\in U,\quad B_i\in V,$$
$$\text{and}\ R_i\subset U\times V; \qquad (14)$$

$$\mu_{B'_i}(v)=\max_{u\in U}\min(\mu_{A'}(u),\mu_{R_i}(u,v))$$
$$=\min\max_{u\in U}[\min(\mu_{A'}(u)\mu_{A_i}(u)),\mu_{B_i}(v)]$$
$$=\min(\alpha_i,\mu_{B_i}(v)), \qquad (15)$$

其中

$$\alpha_i=\max_{u\in U}\min(\mu_{A'}(u),\mu_{A_i}(u)). \qquad (16)$$

由公式(14)和(15)给出的操作在图2中说明。B_1，B_2，…，B_N中的最大值(即“取极大”操作的结果)决定了总的结果判定(或动作)B'，即

$$B'=\bigcup_i B'_i \qquad (17)$$

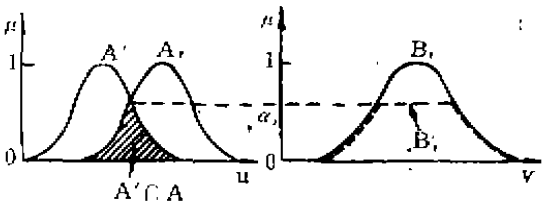
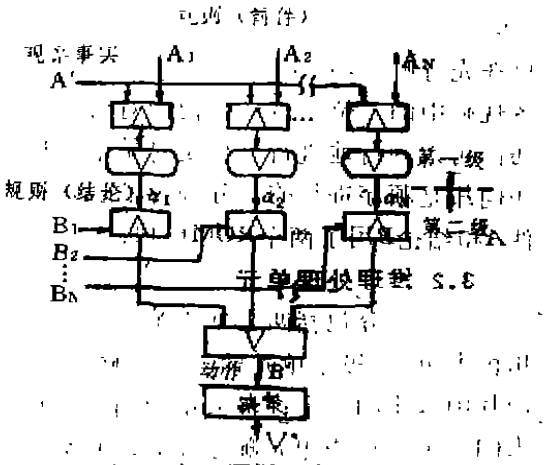


图2　$B'=A'\circ(A\rightarrow B)$的“极大极小运算”机制

上述的推理机制可以用图3所示具有两层分级结构的逻辑结构实现。诸α_i的适当选择是第一层(或级)判定，B'的适当选择是第二层(或级)判定。在这结构每列中串行执行的操作与图2所示的操作等价。文献[10]表明，无论是对单一观察、单一动作的情形，还是对多个观察、多个动作的情形，模糊推理机制的功能结构基本上都与图3所描述的相同。本文提出的结构的优点是：最大和最小操作分别由或门和与门来实现。

三、VLSI设计

相应的VLSI推理机由三个主要部分构成：规则集存贮器，推理处理单元及控制器。前节所述的推理机制并行地执行所有的规



A…实现min的逻辑元素
V…实现max的逻辑元素

图3　模糊推理机制的功能结构

则，这一高度并行性要求在规则集存贮单元与推理处理单元之间用宽总线联接。由于在这两个单元之间数据通讯的速率很高，我们决定把规则存贮在一个芯片上，否则，对管脚数目的限制将不利于充分利用并行性。

因为这是设计的第一个版本，我们更强调简洁性，为此而做的重要决策之一就是顺序地处理各条规则，这一点简化了设计，增强了可扩展性。模糊推理机制的逻辑结构被很好地映射到了相应的 VLSI 结构上。在模糊逻辑的基本运算与 VLSI 推理处理单元的组成部件之间，有一一对应的关系。在下一节，我们将详细讨论推理机的三个主要部分。

3.1 规则集的存贮和格式

规则集既可用随机存取存贮器（RAM）也可用只读存贮器(ROM)来存贮。使用RAM的好处在于灵活性。根据不同的应用，规则集可以从芯片外加载上去。另一方面，在存贮同样数量的数据时，使用 ROM 占的空间小，操作速度快。推理机的控制单元可以很简单，因为不必从芯片之外加载规则集。而且，我们已经有了一个精心设计的 ROM 生成器，所以我们采用ROM存贮推理规则。

我们考虑实用中模糊子集的大小以及模糊性的等级。在多数情况下，模糊子集的大小选择在3到16之间，模糊性的等级则在3到12之间[7,8]。在实现中，我们限定模糊子集的论域空间为一个具有31个元素的有限集。隶属函数则离散化为16个等级(即4位二进制数)。也就是说，0表示无隶属关系，15表示完全隶属，其它数则代表单位区间〔0,1〕上的点。共有124位用于隶属函数的数字化*。规则的表达格式如下：

Rule i: $A_i \rightarrow B_i$,

其中，

	u_1	u_2	u_3	…	u_j	…	u_{31}
A_i	0010	0100	1111	…	$\mu_{A_i}(u_j)$	…	0000

	v_1	v_2	v_3	…	v_j	…	v_{31}
B_i	0000	0001	0011	…	$\mu_{B_i}(u_j)$	…	1100

这里，每 4 位代表论域空间中一个元素的隶属度。例如，在子集A_i中，u_1的隶属度为$\frac{2}{15}$，而u_3则完全隶属。每个 4 位整数按最高有效位在先存贮。ROM 中的每个词都存贮所有各规则中的一位，这样就使得对所有规则的访问可以并行地进行。但是对单个的规则，访问却是顺序进行的。分别存贮规则集的前件A和结论B用了两个ROM模块。

3.2 推理处理单元

这一部分包括两个基本单元：极小单元和极大单元。极小单元输入两个整数，然后得出其中较小的一个；极大单元则输出其中较大的一个。这些单元顺序地处理整数。极小单元的状态图示于图4中。极小单元和极大单元是执行模糊“与”和“或”运算的基本单元。

推理机处理单个规则时的基本数据通路示于图5(a)中，它直接对应于图3所示推理机中的一条数据通路。移位寄存器用于保持第一级中”与“运算完成之后所得的极大值，这之所以必要是因为单条规则的处理是顺序进行的；当第一级上的运算完成之后，这一寄存器记录下极大点的值α_i。第二级上的最后一次运算需要在所有数据通路（也即所有规则）中得出最大隶属函数，完成这一运算的方法就是将所有极大单元联成如图 6 所示的二叉树结构。

*)因为生成器要求ROM的列长是2的幂，所以对规则集中的每一模糊子集我们事实上用了 128 位 ROM空间。其中的第一位总是0，当推理处理器的IF部分或THEN部分不被计算时，该位用于使它们钝化。另外还有3个哑位，因此仅仅有124位（31个元素)的有效数据。

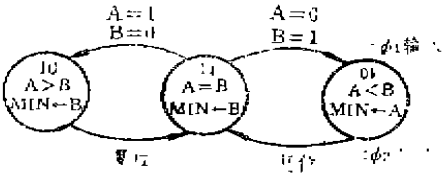


图4 极小单元的状态图

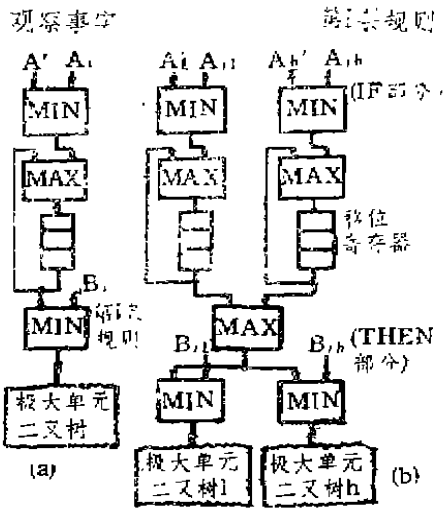


图5 (a)推理处理器的单数据通路；(b)一条规则的二数据通路

3.3 控制器

由于体系结构的简单性，所以推理机的控制器也是简单明了的。控制器由两个计数器组成，它们顺序访问两个ROM模块。控制器每隔四个循环就向极小、极大单元发一次复位信号。当前件部分的处理完成时，控制器就立即访问规则集的结论部分。控制器还能通知用户有效输出开始。

3.4 线路细节

第一次实现的方案能存储和处理16条规则，每条规则包含124位的前件和124位的结论。一条观察事实和一个动作各占124位，其加载和产生都是顺序进行的。系统使用了一个片外(即不做在系统芯片上的)非重叠两相时钟。系统的操作由一个复位信号初始化，该信号必须延续一个时钟周期并且使整个电路复位。观察事实必须在复位后两个时钟周期，也即在第三个时钟周期内输入。推理机则在复位信号后第133个时钟周期开始输出结果。由控制器发出有效输出开始的信号。

芯片的有效面积是2.99×3.48 mm，使用68-脚封装。但只有8个管脚用于芯片的运算，这些是：VDD，VSS，phi-1，phi-2时钟信号，顺序加载的观察事实，动作，复位信号，以及指示有效输出开始的信号。30支管脚用于输出处理器中关键节点的值，以便调试。

3.5 定时模拟

在MULGA系统[1]上的定时模拟，提供了一个20.8兆赫(即周期为48微秒)的操作速率。在目前每条规则占用124位的数据格式的情况下，一个推理过程要占用256个时钟周期。这样，推理机每秒能完成大约80,000次模糊逻辑推理(即80,000 FLIPS)。如果规则的分辨率(即刻划模糊子集的详细程度

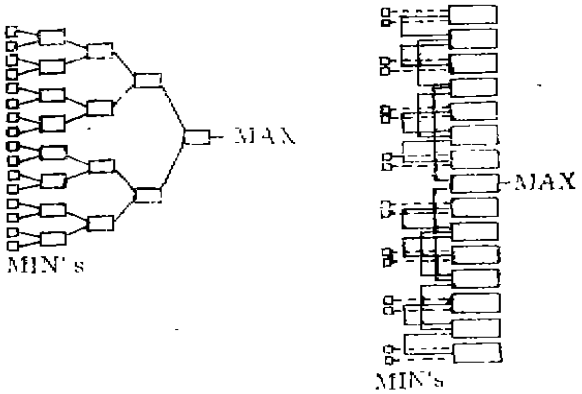


图6 极大单元二叉树

——译者)加倍，则推理机的速度将减半。但是，如果为每条规则分配两条数据通路，就可以使每次推理中速度只减少几个时钟周期，其细节在下节中介绍。

四、设计的易扩充性

前述推理机结构的优点在于简洁性和易扩充性。只需在设计上作很小的变化，就可以将规则表示格式，即31个整数(每个占4个二进位)，加以扩展。例如，论域空间的元素数目(即数据点数目)可以多于31，而只需对控制器作不大的改进。通过增加移位寄存器的长度，修改控制器，我们可以使隶属函数的数字化达到更高的分辨率(用4-位以上的二进制数)；通过增加数据通路并相应改变

二叉树的设计可以使规则的数目增加。

我们可以增加推理处理器的吞吐率（即单位时间内能处理的规则数目——译者），这只需对结构作不大的改动。增加一个移位寄存器，可使我们得到一个流水线处理器。前件(第一级)和结论（第二级）可以同时运算。第二个移位寄存器紧放在第一个之后，这样，第一个移位寄存器用于保存处理前件时得到的最大值α_1，而第二个移位寄存器将计算得到的α_2值保存起来供计算结论部分用。通过这种流水作业，处理器的吞吐量可增加一倍。

另一种可行的结构是每条规则都采用多条数据通路。这种情况下，存在面积-速度之间的权衡与折衷问题。例如，通过为每条规则增加一个数据通路，在被开销占用掉一个时钟周期的情况下，可以使处理速度几乎增加一倍。我们将规则分为两半，独立存储。观察事实的相应两半也分别从两支管脚上装入；在处理器的第一、第二级上，规则和观察事实的每一半的处理几乎是独立的。在第一级的最后，为了组合α_1的两部分，我们需要一次额外的极大运算。在第二级上，这时需要两个独立的极大单元二叉树。这一结构的略图示于图5(*b*)。

注意在第二级的最后一步之前，每一条规则都是由一条数据通路独立执行的（指的是图8所示的设计——译者）。在第二级的最后一步，等选出一个最大的元素（从诸规则的结论中——译者）。为了能处理大量的规则，可按照规则分片方式设计一组芯片，每个芯片上执行规则集的一个子集（例如每片32条规则）。把这些芯片并联起来(原文为串联(cascading)，疑误——译者），就可以并行地执行大量规则。为了选择最终结果，我们可以把极大数选择器设计成一个独立的芯片。例如，使用两层32输入的极大数选择器(图6)，我们就可使用1024条数据通路。

我们提出了一个基于模糊集合论"极大极小运算"的，用于实时目的的推理机制。它能处理不精确、不确定性的知识，能处理不确定性的条件，并能获取人类专家的知识与推理过程。我们给出了这种推理机的结构，一个实际的推理机已采用CMOS技术设计成功，并已投入生产。VLSI设计强调简洁性、可扩充性以及高效率。广泛的定时模拟表明主要的设计目标都已经达到。这种推理机在实时方面的可能应用，不仅包括指挥与控制领域中的决策制定，而且还包括智能机器人系统中的实时控制等。

参 考 文 献 （略）

〔**石桥林** 译自 "Information Sciences, An International Journal", Vol.38, No. 2, 1986, p.147-163 **刘大有**校〕

（上接44页）

·用于演化式原型方法的设计技术是决定性的因素。使用开式系统结构是最有效的、最恰当的。

·文档可以并且应该保持最新版本，而且，人们应该在决定记录什么时要恰当地作选择，文档工具的使用也是很重要的。

·抛弃代码和设计并不是不可取的，只要对目标有好处，在许多情况下，它对缩短将来的工时是一种投资。

·保持设计的清晰和结构化是重要的，在这方面需不断地努力以确保灵活性。特别是在原型设计过程中，不应进行优化设计，这反会使设计变坏，并使进一步开发变得困难。

·虽然原型方法支撑环境很重要，但是目前许多可用的工具和程序设计环境仍是有用的，Lisp程序设计环境的使用可能尤为有效。

·演化式原型方法可能涉及一些关键性的决策活动，通过合适的审查，有助于保证不忽视这个问题。

我们的结果仅仅建立在单一实例研究基础之上。毫无疑问，需要更进一步的资料，才能得出有关演化式原型方法可能适用于大型软件项目的更全面结论。我们希望本文报告的项目在这一方面已经迈出了一步，也许会激励对这一领域的更深入研究。

参考文献（略）

〔**王曼玲**译自《ACM SIGSOFT SOFTWARE ENGINEERING NOTES》1987, Vol.12 No.1, PP38—41, **郭浩志**校〕