第 33 卷 第 3 期 计 算 机 工 程 2007 年 2 月
Vol.33 No.3 Computer Engineering February 2007
·工程应用技术与实现· 文章编号：1000—3428(2007)03—0253—03 文献标识码：A 中图分类号：TP18

模糊 CMAC 的硬件结构分析及其 FPGA 实现

沈宪明，白瑞林，章智慧
（江南大学控制科学与工程研究中心，无锡 214122）

摘 要：提出了模糊 CMAC 的一种基于 FPGA 的硬件实现方法。与其它 FPGA 实现的神经网络相比，包含了可以用于在线学习的权学习算法。分析了模糊 CMAC 的模型结构及其相应的硬件模块；用 VHDL 实现基于上述模块的模糊 CMAC；对该模糊 CMAC 进行硬件综合与测试。测试结果表明：该模糊 CMAC 的 FPGA 实现方法是可行的，硬件化后的网络具有速度快、精度高、占用器件资源少的特点，是在 SOPC 中实现模糊 CMAC 模块的一种有效方法。

关键词：模糊 CMAC；FPGA；VHDL；函数学习

Hardware Structure Analysis of Fuzzy CMAC and Its FPGA Implementation

SHEN Xianming, BAI Ruilin, ZHANG Zhihui
(Research Center of Control Science and Engineering, Southern Yangtze University, Wuxi 214122)

【Abstract】 This paper proposes a FPGA implementation structure of a fuzzy CMAC. Compared with other neural networks implemented by FPGA, it contains the learning algorithm which can be employed to realize the on-line learning. The model and the relevant hardware modules of fuzzy CMAC are analyzed. It implements the fuzzy CMAC based on the above hardware modules with VHDL. The design is synthesized and tested. The test result shows that the method of the hardware implementation of the fuzzy CMAC is feasible. The implemented network comprises the characteristics of high speed, good precision and little chip resource. It is an effective method in implementing the module of fuzzy CMAC in SOPC.

【Key words】 Fuzzy CMAC(cerebellar model articulation controller); FPGA; VHDL; Function identification

1 概述

在非线性系统的在线辨识中，常规的串行辨识算法在实时性方面并不能满足某些场合的要求；而一些智能辨识算法，如BP算法，虽然在处理数据时，采用并行方式，提高了辨识速度，但它本质上属于全局优化算法，在一定程度上阻碍了速度的进一步提高。Albus提出的基于局部学习的CMAC (Cerebellar Model Articulation Controller)网络[1,2]，具有很快的收敛速度和可观的精度，已广泛应用于机器人控制、信号处理和模式识别。但是，它的内在缺陷影响了它的进一步应用：(1)随着维数的增加，权空间大小将呈几何级数增长，使用哈希影射在一定程度上降低了权空间大小，伴随而来的权碰撞，将会影响到权值的收敛，因此难以解决；(2)在一些精度要求比较高的场合，CMAC的学习精度难以满足要求。为提高精度，已有了一些改进型算法[3,4]，但并不能从根本上解决问题：由于CMAC的基等于 1，当CMAC的不同输入向量经量化后，可能映射到同一个box函数中，输出值将相同，这样输出就不平滑，精度也就难以提高。

模糊CMAC的出现解决了以上两个问题。它将模糊逻辑引入CMAC，结合了CMAC具有学习能力和模糊逻辑善长表达近似与定性知识的优点[5]，使不仅具有CMAC训练速度快的优点，而且存储空间小、可以连续量输入、无须量化、避免了CMAC的哈希映射对权值收敛的影响。由于模糊CMAC使用的box函数是隶属度函数，因此不同的输入映射到同一box函数后，其输出值是不同的，消除了CMAC输出不平滑的缺陷；此外，模糊CMAC可以构造模糊推理规则，加入了人的经验，从根本上解决了CMAC训练精度不高的问题。这样，模糊CMAC的实现具有了更好的实用价值。与软件实现相比，硬件化模糊CMAC可以保持网络自身的并行计算能力，提高处理速度。

FPGA 的结构特性适合实现并行运算，与 ASIC 实现相比，FPGA 的可重复编程性更加灵活，而且 FPGA 实现更加便宜。同时，FPGA 可以实现模糊 CMAC 的学习算法，使得硬件化结构可以在线学习，这是 BP 网络等不能比拟的。

本文提出了一种基于 FPGA 实现的模糊 CMAC 结构。首先分析了模糊 CMAC 的结构算法及相应的硬件模块，然后基于 VHDL 语言实现了各硬件模块的功能描述，最后由 Altera 公司的设计工具进行综合与测试。

2 模糊 CMAC 的模型、学习算法及其仿真

2.1 模糊 CMAC 的模型结构

模糊 CMAC 本质上是一种类似于 CMAC 的查找表结构模型，其隶属度函数保证了可以连续量输入，它通过查询权值表和借助模糊推理，进行权值的存储与学习，以获得神经网络的输出。模糊 CMAC 的结构如图 1 所示，分为 5 层：输入层，模糊化层，模糊相联层，模糊后相联层和输出层。

(1)输入层：该层的各节点与输入向量的各分量x_i相连，将输入向量$x=(x_1, x_2, \ldots, x_n)^T$传递到下一层。输入状态空间的划分以模糊化层的语言变量为依据，不同的输入对应不同的语言变量，即被感知的神经元节点不同。

(2)模糊化层：模糊化层分别对应于模糊逻辑中的模糊化和

作者简介：沈宪明(1981－)，男，硕士生，主研方向：嵌入式系统研究；白瑞林，教授；章智慧，硕士生
收稿日期：2006-03-29 **E-mail：**ofa-shenxianming@163.com

CMAC 中的感受域，它继承了 CMAC 模型有限宽的特点，即每次只有几个模糊节点被激活，从而保证了学习速度快的特点。该层中能被同一输入分量激活的所有节点组成一个模糊子集，而其中的每个节点分别对应一个语言变量，不同的输入对应不同的语言变量，完成上一层输入的隶属函数运算。常用的隶属函数有铃型函数、三角形函数和梯形函数等。

(3)模糊相联层：对应于模糊逻辑中的模糊推理。它的各个节点进行模糊运算，如模糊与、模糊乘等，以得到相应的规则适用度。

(4)模糊后相联层：模糊后相联层和输出层组合成模糊逻辑中的解模糊运算，以得到清晰化的网络输出。此层完成规则适用度的归一化运算，具有与模糊相联层一样的节点数，为网络输出运算做准备，其计算如式(1)所示。

$$\overline{\alpha_j} = \alpha_j \Big/ \sum_{i=1}^{N_A} \alpha_i , i = 1,2,...,N_A \tag{1}$$

(5)输出层：该层完成归一化规则适用度与网络权值的加权线性和运算，得出清晰化的输出值。如(2)式所示。

$$y = \sum_{i=1}^{N_A} w_i \overline{\alpha_i} \tag{2}$$

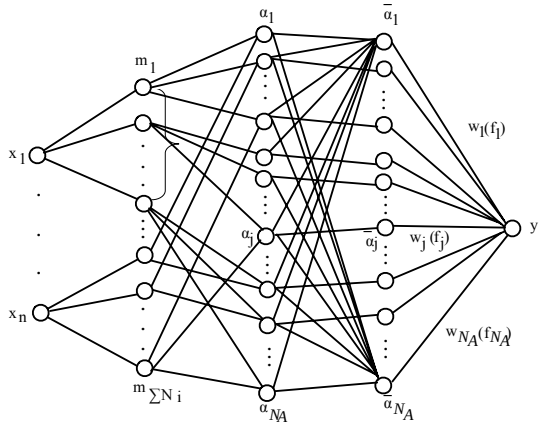

图 1 模糊 CMAC 的结构

2.2 模糊 CMAC 的学习算法

模糊CMAC的学习算法一般沿用CMAC的算法，如C-L、LMS、GA 和变学习率等。由于输出层神经元是线性的，根据 2.1 节的模糊 CMAC 结构，本文采用 LMS 算法，如式(3)所示。此学习算法简单，易于硬件实现。

$$w_j = w_j + \overline{a_j}\beta\frac{(y_d - y)}{m} \tag{3}$$

其中，y_d为期望输出，y为实际输出，β为学习率，m为模糊规则适用度不为零的个数。

3 模糊 CMAC 的硬件结构分析及 FPGA 实现

图 2 为 包含控制单元的模糊 CMAC 顶层模块图，包括模糊化模块、模糊运算模块、归一化模块、地址生成模块、解模糊化模块、权学习算法模块和控制单元模块。

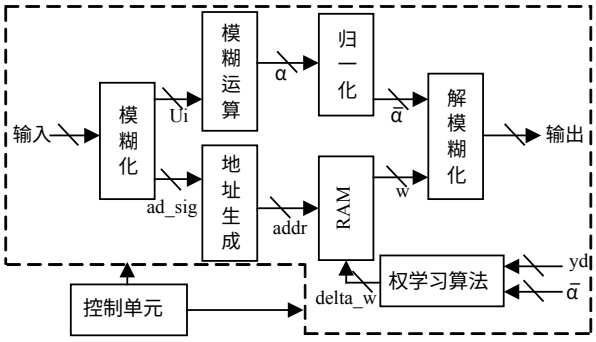

图 2 模糊 CMAC 顶层模块

(1)模糊化模块

在模糊化模块中，我们根据输入状态空间来定义模糊节点的隶属度函数。本处产生 8 个模糊节点，如图 3 所示，相应用信号 ad_sig 来表示节点的位置。对于某维输入，它所激活的模糊节点进行模糊运算，输出隶属度值 Ui(i=1, 2 , …, m。m 为模糊化层节点的个数)和 ad_sig 值；对于没有激活的节点，则输出的隶属度值保持为 0。在实际应用中，输入一般是多维的，权地址需要精确运算或编码以避免地址碰撞。

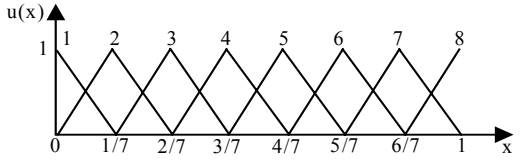

图 3 隶属度函数

(2)模糊运算模块

此处选择模糊乘作为模糊推理算法，以各隶属度值 Ui(i=1, 2 , …, m。m 为模糊化层节点的个数)作为模块输入，输出为模糊规则适用度。对于硬件乘法器，必须在算法的精度和资源利用率上做一个折中，如果芯片的乘法单元(如 DSP block)比较丰富，则可以适当地增加字长，否则只能减少，因为用 LEs 实现乘法器会花费较大的芯片资源。

(3)归一化模块

如果模糊 CMAC 的输入向量是二维的，隶属度函数如图 3 所示，则每维每次有 2 个模糊节点被激活，所得的隶属度值之和将是 1，此时，本模块可以省略。但大多数情况下，如输入为多维或隶属度函数较复杂，那隶属度值之和就不为 1，需要用到除法器，这时也需要考虑到算法的精度与硬件规模的关系。在选择多维输入的模糊子集时，应尽量使隶属度值之和为 2 的幂次，因为在综合过程中，软件会自动地将除数为 2 的幂次的运算转化为移位运算，而移位运算不占用器件资源。

(4)地址生成模块

模糊化模块不仅产生隶属度值，而且也产生 ad_sig，ad_sig 用于生成权地址。每次所需要的权地址由 ad_sig 通过运算产生，而 ad_sig 的值由被激活的模糊节点位置决定。权空间大小由各维的模糊子集决定，如输入为三维，各模糊子集所含语言变量的个数分别为 a、b 和 c，则权地址空间的大小为 a×b×c。需要注意的是，每次得到的权地址应不存在碰撞的情况，否则会发生错误，甚至使整个网络失效。当然，如果输入是一维的，地址生成模块可以省略，实际的权地址就是模糊节点序号。

(5)解模糊化模块

此模块的任务是完成式(2)，即获得网络的实际输出，其结构如图 4 所示，包含加法器和乘法器。

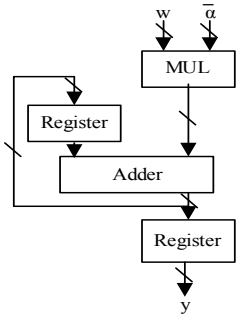

图 4 解模糊化模块

RAM 中的权与相应的归一化模糊规则适用度的加权线性和即为模糊 CMAC 的实际输出。可用寄存器来累加这些乘积。当然，如果器件资源允许的话，我们可以不用累加，而并行处理这些乘积，这样可以提高速度。

根据 FPGA 的内部结构特点，可以用 3 种资源来实现数据的动态存储：Block RAM，LUT 和 Register。此处选择 Block RAM 来存储权值，因为使用 Block RAM 可以节省 LEs 资源，是最大程度发挥所选器件效能、节约成本的一种体现；而且，Block RAM 是一种可配置的硬件结构，其可靠性和速度与 LUT 和 Register 构建的存储器相比更有优势。

(6)权学习算法模块

不同的权学习算法对应不同的硬件结构，本文使用式(3)。y 与 yd 的差乘以学习率和归一化模糊规则适用度，所得的乘积除以 m，得到 delta_w，即权值修改量。通过 delta_w 使权值得到更新。

这部分包含乘法和除法，将花费较大的器件资源。因此在运算过程中，要及时地截位，以降低芯片的使用。

(7)控制单元

用于控制各模块的执行顺序，一般可以遵循 MATLAB 仿真时的顺序。

4 模糊 CMAC 的 FPGA 实现测试

根据以上提出的结构，用函数逼近来测试所提出的模糊 CMAC 的硬件结构，选择非线性函数：F=cos(x2)*sin(x1)，x1,x2∈[0,1]作为测试对象。x1,x2 在[0,1]中各取 9 个点，每个点的间距为 0.125，这样得到 81 对样本用于训练网络。网络训练 30 次后，得到 81 对样本的累加误差为 207(16 位表示)，由于值代表的是 2 位整数 14 位小数，所以实际值为 207/2^14=0.012 6，与 MATLAB 仿真结果 0.012 2 基本相符。由此，可以证明所设计的硬件结构是可行的，而且，模糊 CMAC 具有很好的学习精度。相应器件利用率如表 1 所示。

从表 1 中可以看到，器件内部的资源分配很合理。由于 DSP block 的存在，LEs 占用得不多，内部存储单元占用得很少，使得在 SOPC 中嵌入此模糊 CMAC 模块成为了可能。

表 1 模糊 CMAC 占用器件资源情况

Device	EP1S10F780C6
Total logic element	1 340/10 570(12%)
Total memory bits	4 096/920 448(<1%)
DSP block 9-bit elements	20/48(41%)
Maximum clock rate	15.95MHz

5 结论

本文提出了模糊 CMAC 的一种硬件结构，并在 FPGA 上实现，经实验测试证明了该方案的可行性。

模糊 CMAC 降低了权空间的大小，解决了因权空间大而硬件难以实现的问题；模糊推理和隶属度函数的存在，使网络输出更加平滑，从根本上解决了 CMAC 的精度问题，使其具有更好的学习精度；在实现乘法和除法运算时，要在算法的精度和资源利用上找一个折中；在实现权存储单元时，以 Block RAM 为首选，因为它可以节省 LEs，可靠性和速度与 LUT 和 Register 相比更有优势。

本文提出的模糊 CMAC 结构可以作为 SOPC 的一个用户自定义 IP 模块，用于系统的在线辨识。

参考文献

1 Albus J S. A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller[J]. Trans. of the ASME: Journal of Dynamic Systems, Measurement and Control, 1975, 97(3): 220-227.

2 Albus J S. Data Storage in the Cerebellar Model Articulation Controller[J]. Journal of Dynamic Systems, Measurement and Control, 1975, 97(3): 228-233.

3 Luo Jianxu, SHAO Huihe. Improved Learning Algorithm of Hyperball CMAC and Its Convergence Analysis[J]. Journal of Shanghai Jiaotong University(Science), 2004, E-9(3): 21-24.

4 Wang Yanpin, Su Shunfeng, Lee Znejung. Robust Credit Assigned CMAC[J]. IEEE Trans. on Systems, Man and Cybernetics, 2003, 5(5-8): 4457-4462.

5 邓志东，孙增圻，张再兴. 一种模糊 CMAC 神经网络[J]. 自动化学报, 1995, 21(3): 288-294.

（上接第 252 页）

时，要求得到的程序的每条指令束的指令组成、指令束的数量、执行顺序和执行周期数均与按照变长指令编码方法得到的相同，如图 2 所示。

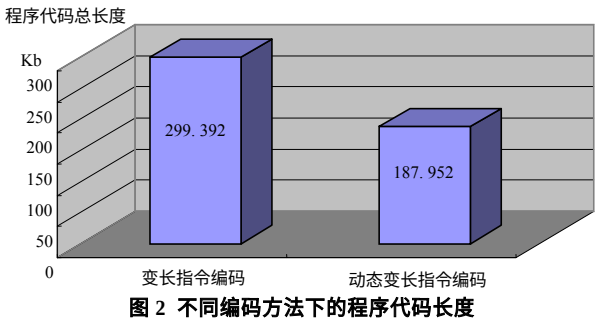

图 2 不同编码方法下的程序代码长度

相对于变长指令编码，动态变长指令编码的压缩率为 187 952/299 392=62.8%(压缩率=压缩后的指令长度/压缩前的指令长度)。

4 结论

对于低编码率的语音算法的专用指令集处理器(ASIPs)，动态变长指令编码可以将代码长度压缩率提高到 62.8%。这样大大减少了程序代码长度，也就减少了程序代码存储空间，因而可以减少 ASIPs 设计的总体成本。

参考文献

1 Pan H, Asanovic K. Heads and Tails: A Variable-length Instruction Format Supporting Parallel Fetch and Decode[C]//Proc. of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, Atlanta, GA, 2001-11.

2 Xie Y, Wolf W, Lekatsas H. A Code Decompression Architecture for VLIW Processors[C]//Proceedings of the 34th ACM/IEEE International Symposium on Microarchitecture, CA, USA, 2001: 66-75.

3 Rau B R, Fisher J A. Instruction-level Parallel Processing: History, Overview, and Perspective[J]. The Journal of Supercomputing, 1993, 7(1/2).

4 Smotherman M. Understanding EPIC Architectures and Implementations[C]//Proc. of the the 40th Annual Southeast ACM Con-ference, Raleigh, North Carolina, USA, 2002-04-26.

5 Segars S, Clarke K, Goudge L. Embedded Control Problems, Thumb, and the ARMT7TDMI[J]. IEEE Micro, 1995, 15(5): 22-30.

6 Nam S. Improving Dictionary-based Code Compression in VLIW Architectures[J]. IEICE Trans. on Fundamentals, 1999, E82-A(11): 2318-2324.