A New CMAC Neural Network Architecture and Its ASIC Realization

Yuan-Pao Hsu, Kao-Shing Hwang, Chien-Yuan Pao, and *Jinn-Shyan Wang

Electrical Engineering Dept.,
National Chung Cheng University
Chia-Yi, Taiwan
*ieegsw@ccunix.ccu.edu.tw

***Abstract*--A new VLSI architecture of the CMAC neural network, called CCMAC, which is composed of an embedded *content addressable memory* (CAM) is proposed. The CCMAC can determine whether the addressing content resides in the memory or not faster. If a match doesn't occur, the activated address is stored in a vacant cell of the CAM. With this mechanism, memory utilization rate can be improved to 100 % and control noise can be suppressed. Moreover, the associated mask function, which is triggered while the CAM is full, can improve control performance. Comparison between CCMAC and a conventional CMAC is examined on the truck backer-upper control simulation. A CMAC ASIC chip has been implemented using a 0.6-μm CMOS technology under 3.3- or 2.5-V supply voltages after obtaining design trade-off. The chip contains 0.33-million transistors and operates at 20 MH for a 2.5-V supply with only 10–mW power dissipation.**

I. Introduction

The Cerebella Model Articulation Controller (CMAC), originally proposed by James Albus [1], is the architecture of a neural network. Fundamentally, a CMAC estimates the desired output by taking input states as an index to refer to a look-up table where the synaptic weights are addressed and stored. The relationship between input and output can be represented approximately by a CAMC if addressable weights are properly updated. Functionally, a CMAC is defined by a series of mappings as shown in Fig. 1. Due to the properties of local generalization, rapid computation, function approximation and output superposition, it has been widely used in robot control, pattern recognition, and signal processing.

Since the conceptual memory of a CMAC theoretically needs a huge space to address the encoded input information, its hardware implementation is hard to be realized. In general, hash coding algorithms [1] are always applied to reduce the space into a more reasonable scale. However, this approach has some drawbacks [4]. First, collisions always occur. For instance, as shown in Fig. 1, the cells of c_3 and a_2 project onto a memory cell P_1 simultaneously. Although some algorithms [2] have been introduced to achieve low collision rate, some cells are still never visited thoroughly. Second, while collisions occur, from the viewpoint of system control, it could cause CMACs interfered by so-called control noises.

The objective of this paper is to propose a new VLSI architecture of a CMAC, which is composed of an embedded content addressable memory (CAM). The CAM is to replace the hash coding for address mapping used in conventional CMACs and can achieve the goals of good memory utilization and alleviation of the control noise resulting from hash algorithms.

The paper is organized as follows. Section II describes the VLSI architecture of a conventional CMAC. Section III introduces the new architecture of the CCMAC. Meanwhile, in Sections II and III, a truck backer-upper control example is demonstrated. Section IV shows the design considerations and the implementation of the CCMAC ASIC chip. Finally, conclusion is given in the last section.

II. The Conventional CMAC

Fig. 2 shows the functional block diagram of a conventional CMAC, called HCMAC in this paper. The input vector is fed to the Conceptual Mapper *M* is from the plant or environment. Eq. (1) describes the conceptual mapping from *S* to a conceptual memory *C* if the input vector is one dimension:

$$f_c(S_j) = X_{ji} = [\text{int}(\frac{S_j + R - i - 1}{R})] \times R + i, \qquad (1)$$

Sj is the input, and *R* is the receptive field (*R* = 4 in the case of Fig. 2). *i* is an index number of the activated address, and *Xji* is the activated address mapped from *S* to *C* . And a Hash Coder further maps the activated cells into a physical memory. The output of the controller is derived by the summation of the values(weights) stored in the indexed cells. Weights are adjusted based on the errors evaluated from the difference between the desired and actual outputs.

The following is a control example of truck backer-upper simulation [4]. Fig. 3 is the configuration of the workspace. ϕ and x are the input variables to the neural network. θ is the output variable. To control the truck in a

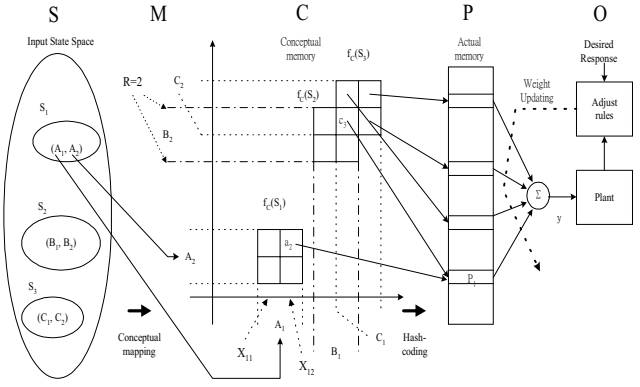

Fig. 1 Principle of CMAC

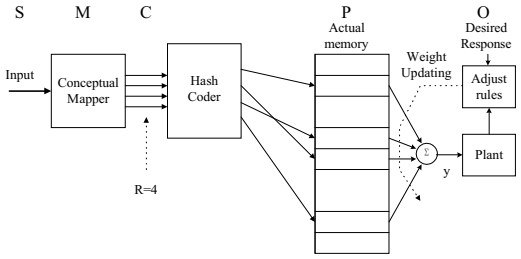

Fig. 2 HCMAC architecture

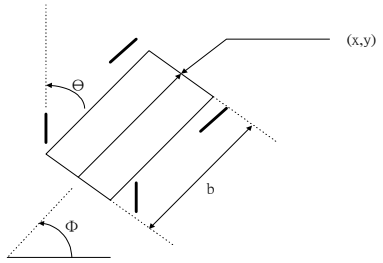

Fig. 3 Truck angle and position definitions

manner of backer-upper going toward the middle line until $\phi = 90°$ and $x = 0$ is the goal of the control.

There are 36 active addresses. The size of the actual memory P is set to 1024. A weight-updating algorithm similar to the Hebbian learning rule is applied to adjust weight incrementally. Fig. 4 shows the trajectory along which the truck, controlled by HCMAC, reached the desired position at the 129th sampling time from the starting point.

III. The Proposed CCMAC

The functional architecture of a CCMAC is shown in Fig. 5. The CAM stores the activated addresses and their corresponding weights. The input of each control cycle introduces some activated addresses through a conceptual mapping function similar to Eq. (1). These addresses are compared with the ones stored in the CAM immediately. If a match occurs, the corresponding weight is read out, otherwise, these activated addresses is stored in vacant cells of the CAM indexed by an circular incremental pointer. If

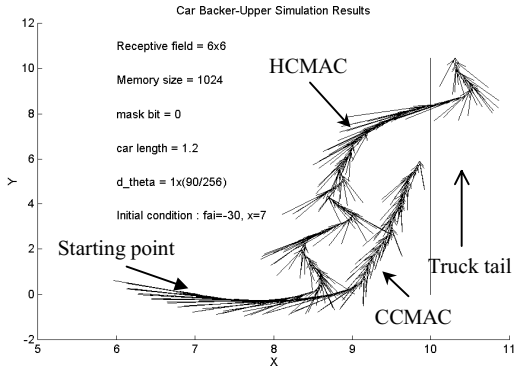

Fig. 4 Simulation results for the truck backer-upper application using CCMAC and HCMAC, respectively.

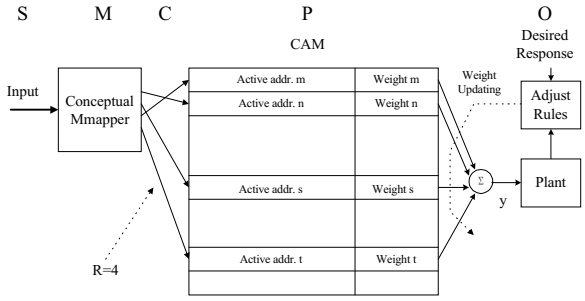

Fig. 5 CCMAC architecture

there is vacancy no more, the mask function is invoked alternatively. In the worst case, where there might be no match throughout the masking-matching process, the address and its context is pushed in an occupied cell. With this mechanism, memory utilization rate can be improved to 100 % without the problem of control noise.

The simulation results of CCMAC are also illustrated in Fig. 4. Obviously, CCMAC is superior to HCMAC. In fact, the truck reached the desired situation at the 80th sampling time from the starting point. A brief comparison between both neural networks in various aspects is shown in Table 1.

IV. Design Of The CCMAC Chip

The excellent performance of the new architecture of the CCMAC neural network motivates its VLSI implementation. An application-specific VLSI chip also enhances the control speed in real applications. The design flow of the CCMAC ASIC for the truck backer-upper control is shown in Fig. 6.

Some important VLSI design parameters should be discussed here after the architecture designed.

(a) The size of the receptive field

The larger the receptive field is, the more number of address mappings should be manipulated. Conversely, If the field has a tiny size, the neural network may lose memory plasticity. The best way is to adjust the size dynamically

TABLE 1
Comparisons Between CCMAC And HCMAC

	CCMAC	HCMAC
Active address mapping	Via index pointer	Hash-coding
Hardware requirement	8k-bit SRAM storing weight, 20k-bit CAM storing active address	8k-bit SRAM storing weight, Multiplier (20 x 8*), Adder
Mapping latency	Depend on CAM matching time (fast)	Depend on multiplier operation time (slow)
Memory usability	100%	According to hash-coding algorithm (<100%)
Control noise	Hardly observed [4]	High
Learning ability	Yes	Yes
Applications	Jobs need that low control noise	Can not handle jobs that need low control noise (e.g. truck backer-upper control)

*At least 8-bit

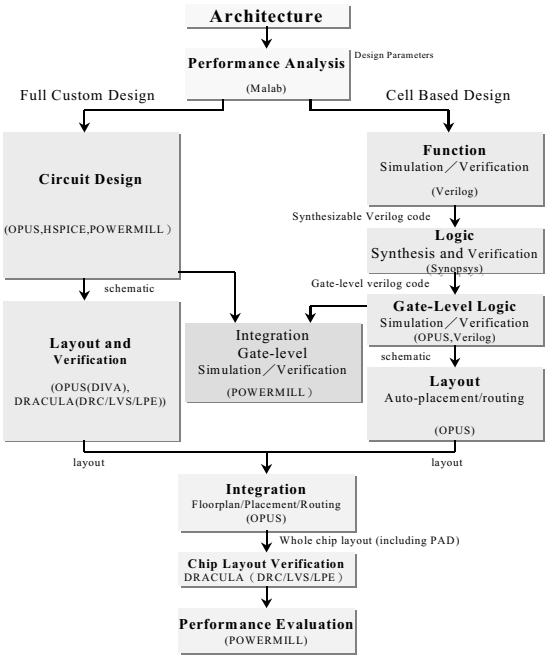

Fig. 6 Design flow of the CCMAC VLSI

by means of the clustering distribution of the input if possible. However, it is very difficult to realize in hardware. Therefore, the trade-off size is 36 in the design.

(b) The size of CAM

The simulation results as shown in Fig. 7, when the size of the receptive field is 6 × 6, CCMAC with only 36 cells of the CAM can drive the truck to the target. But the controller actually has no learning ability. To make CCMAC is capable of control learning, a sufficient capacity of the CAM must be provided. For instance, in a simulation similar to the one in Fig. 7, the CAM only with 1024 cells became full on the 47^{th} sampling time and the truck reached the target at 122^{th} sampling time. Therefore, the size of the CAM needed to record all the addresses in the simulation is 1796. Where the simulation shown in Fig. 7 is under the case while the incremental adjustment of weight updating is chosen as $\Delta\theta = 1$. If the quantity is set to $\Delta\theta = 3$, then the truck reaches the target at the 56^{th} sampling time and the needed size of the memory is reduced to 836. So, the size of the CAM eventually depends on the considerations of hardware realization. The reasonable size of the CAM for this neural network chip is chosen as 1024 × 20-bits.

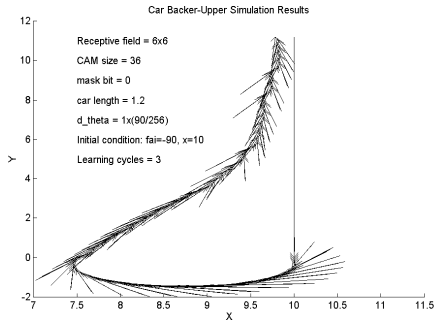

Fig. 7 Simulation result of CCMAC truck backer-upper control under very low CAM size

(c) The length of maskable bits

The CAM possibly overflows in a learning epoch. A special exceptional handling is designed to tackle this problem. When the CAM is all occupied, the masking function of CCMAC is then triggered. Sequentially masking a bit at each matching cycle, CCMAC adjusts its searching range dynamically until a match is found or the mask bit upper bound is reached. The masking function would be limited if too few maskable bits were assigned. Conversely, many maskable bits may introduce unexpected control noise. Therefore an adaptive mechanism to adjust maskable length is the best design.

After the important parameters are determined, the design was proceeded in two directions. Cell based approach, that starts from the verilog coding through logic synthesis to automatic placement-and-routing, is taken for random logic circuits. Whereas, full-custom approach is for the memory parts. A low-power CMOS cell library [6] is used in the technology-mapping step that translates the verilog code into the gate-level design. The full-custom blocks are composed of 1k × 8-bits SRAM, 1k × 20-bits CAM, and a 1024-bit priority encoder. Current-mode technique [7][8] is used in the design of the SRAM and the CAM to help reduce the power consumption of the chip.

In the design of the priority encoder, a self-controlled technique is proposed to enhance the circuit's reliability. Part of the 1024-bit priority circuit is shown in Fig. 8. Without the shaded NMOS's (Node A and node B are shorted.), the circuit is identical to the conventional circuit [9]. In the conventional design, we need to generate the current-priority-clear signal *Current_clear* to be a short-pulsed train, as shown in Fig. 9(a). In our design, the pulse width should be smaller than 2 ns. It is hard to design such a signal without being affected by process variation. Without a precise short pulse width, the cleared data may induce a race problem that will clear the next-priority data in the same cycle. In order to remove this problem, those shaded NMOS transistors are added to the conventional circuit. These transistors will be turned off when *Current_clear* goes high, and the clear signal of the flip-flop of the next priority will be isolated. This self-controlled mechanism can thus prevent from the race problem. The most convenient manner to design the *Current_clear* signal is to make it a gated clock as shown in Fig. 9(b).

The whole-chip layout is shown in Fig. 10, and the summary of the chip features is listed in Table 2.

V. CONCLUSIONS

A new VLSI architecture of CMAC neural network is

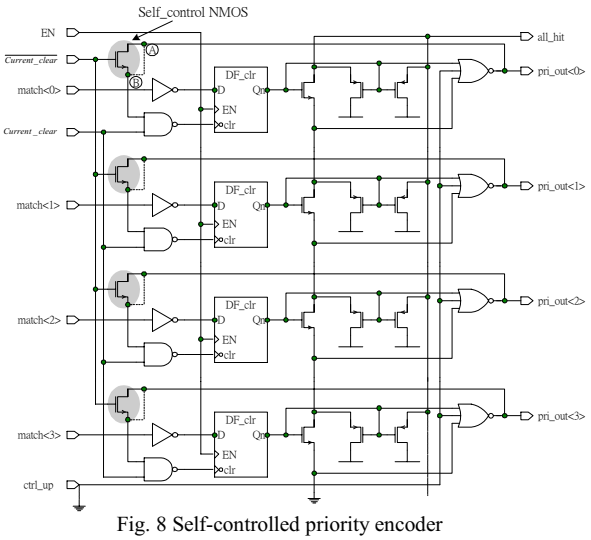

Fig. 8 Self-controlled priority encoder

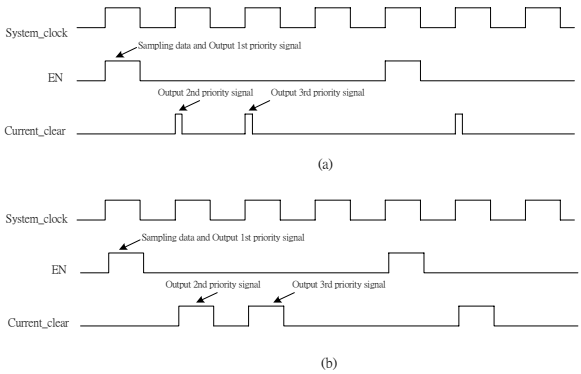

Fig. 9 Control signals (a) without (b) with self-control

TABLE 2
SUMMARY OF THE CCMAC ASIC CHIP FEATURES

Technology	0.6-μm SPTM CMOS
Current-Mode SRAM	1k × 8 bits
Current-Mode CAM	1k × 20-bits and 36 × 11-bits
Self-Controlled Priority Encoder	1024 bits
No. of transistors	366,324
Chip Size	4225.725 × 4845.325 $(\mu m)^2$
Core Size	3245.925 × 3865.625 $(\mu m)^2$
Clock Rate	25 MHz @ 3.3 V or 20 MHz @ 2.5V
Average power dissipation	10 mW @ 2.5V

proposed. CCMAC uses an embedded CAM to replace the hash coding used in the conventional CMAC architecture. The new architecture achieves faster control, higher memory utilization rate, and much lower control noise than the conventional CMAC architecture.

A CMAC ASIC chip for the truck backer-upper control application has been implemented using a 0.6-μm CMOS technology under 3.3- or 2.5-V supply voltages. A low-power cell library for random circuits and the current-mode technique for SRAM and CAM are utilized to reduce the power consumption. The chip contains 0.33-million transistors and can operate at 20 MH @ 2.5 V for only 10–mW power dissipation.

VI. REFERENCES

[1] J. S. Albus, "A new approach to manipulator control: The cerebella model articulation controller (CMAC)," Trans. ASME, J. Dynamic Syst. Meas., Contr., vol. 97, pp. 220-227, Sept. 1975.

[2] Z. –Q. Wang, J. Schiano and M. Ginsberg, "Hash-coding in CMAC Neural Networks," proceedings of International Conference on Neural Networks, pp. 1698-1703, Washington, D. C., June 3-6, 1996.

[3] Yuan-Bao Hsu, Kao-Shing Hwang, and Jinn-Shyan Wang," A CMAC neural controller with embedded content addressable memory (in Chinese)," 1999 Automatic Control Conference, pp.463-469, Taichung, Taiwan, 1999.

[4] L. –X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," IEEE Transactions on systems, man, and cybernetics, vol. 22, no. 6, Nov./Dec. 1992.

[5] B. Kosko, Neural Networks and Fuzzy Systems: A dynamical systems approach to machine intelligence, Prentice Hall, Englewood Cliffs, N. J., 1991.

[6] Jinn-Shyan Wang, Shang-Jyh Shieh, J.-C. Wang, and Chingwei Yeh, "Design of standard cells used in low power ASIC's exploiting the multiple-supply-voltage scheme," in Proceeding of 11th Annual IEEE International ASIC Conference, pp. 119-123, 1998.

[7] Jinn-Shyan Wang and Hong-Yu Lee, "A new current-mode sense amplifier for low-voltage low-power SRAM design," in Proceeding of 11th Annual IEEE International ASIC Conference, pp. 163-167, 1998.

[8] Jinn-Shyan Wang, Po-Hui Yang, and Wayne Tseng, "Low-power embedded SRAM macros with current-mode read/write operations," in Proceedings of International Symposium on Low Power Electronics and Design, pp.282-287, 1998.

[9] H. Kadota, J. Miyake, Y. Nishimichi, H. Kudoh, and K. Kagawa, "An 8-kbit content addressable and reentrant memory," IEEE J. Solid-State Circuits, vol. SC-20, no.5, pp. 951-957, Oct. 1985.

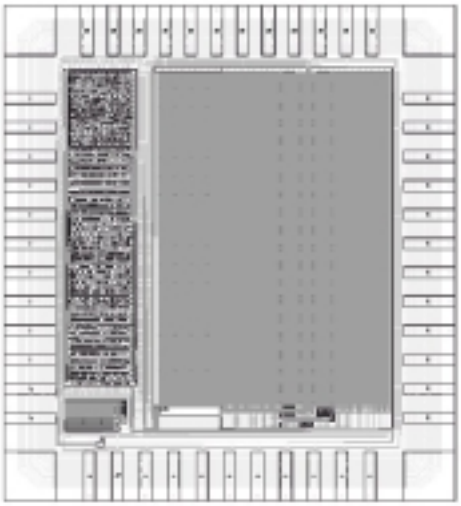

Fig. 10 Whole-chip layout of the CCMAC ASIC chip