ARITHMETIC FOR A HIGH-SPEED ADAPTIVE LEARNING NETWORK ELEMENT

Hideaki Kobayashi* and Ronald D. Bonnell

Department of Electrical and Computer Engineering
University of South Carolina
Columbia, SC 29208

ABSTRACT

This paper presents a novel arithmetic scheme for a high-speed adaptive learning network (ALN) element. An ALN is a self-organizing scheme for implementing the Kolmogorov-Gabor (K-G) polynomial which maps an input vector X into an output scalar Y. In the first layer of an ALN there are n(n-1) / 2 elements. In the next layer the number of elements needed depends upon the number of outputs that are propagated from the first layer. In this paper only the design of a single element is considered. An array of memories (RAMs) and a parallel adder are used to perform multinomial arithmetic for the element. The memory array contains subfunction values which are calculated by an external host computer and downloaded to the memory array. All the memories operate on the input variables concurrently via a common address bus. The subfunction values from the memory array are then summed by a parallel adder to obtain the output of the element. A complete ALN implemented with the proposed ALN elements has advantages in operation speed and less hardware.

INTRODUCTION

In many applications, the model dynamics are difficult to describe analytically. Therefore, it is desirable to have a dynamic model that can be adjusted to represent the input-output database of the application dynamics. An ALN [1,2] is a self-organizing nonlinear multilayer network that can be "trained" to determine the terms in a general K-G polynomial for many variables:

$$Y = W0 + \sum_I WI\ XI + \sum_I \sum_J WIJ\ XI\ XJ + \sum_I \sum_J \sum_K WIJK\ XI\ XJ\ XK + \ldots$$

where Xs and Ws are input variables and coefficients, respectively, and Y is the output variable. The key in building an ALN is to use layers of high speed elements [3] that perform six terms in the complete multinomial for two input variables:

$$Y = W0 + W1\ X1 + W2\ X2 + W3\ X1\ X2 + W4\ X1^2 + W5\ X2^2$$

This paper describes an arithmetic scheme for a high-speed ALN element which uses a memory array and a parallel adder. The complete multinomial will be partitioned into a set of subfunctions each represented in a sum-of-product form. The proposed ALN element will be compared with a typical element using multipliers, memories, and a parallel adder.

MULTINOMIAL PARTITIONING

Figure 1 shows an implementation of the complete multinomial for two n-bit input variables and six n-bit coefficients. The intermediate products are generated by n x n-bit parallel multipliers (MULs 1-3). Memories (MEMs 0-5) containining the respective coefficients (W0-W5) are used to generate the six terms of the complete multinomial. The terms from the memory array are then summed by a parallel adder to obtain the element output Y. The above implementation is not practical for a large input or coefficient length n since large multipliers (2n x n) and memories (2^{2n} x 3n) are needed.

* This work was supported in part by the National Science Foundation under Grant ECS82-04987 and by the University of South Carolina under a Research and Productive Scholarship Fund.

CH1892-9/83/0000/0164$01.00 © 1983 IEEE

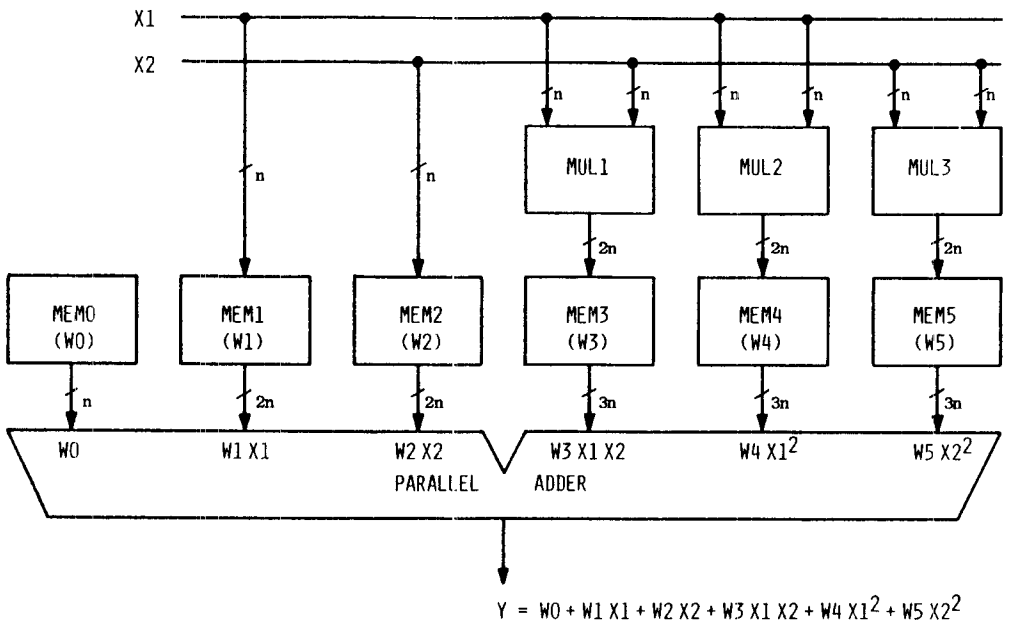

Figure 1. A typical implementation of an ALN element using multipliers, memories, and a parallel adder.

The above approach can be improved by eliminating the multipliers. This is accomplished by programming the subfunction values into the memory array. Since the total number of memory bits increases exponentially with the address length, it is necessary to partition large operands into several suboperands. Let n-bit input variables be partitioned into d number of r-bit suboperands (where d = 2, 3, 4, ...). For example, the input variables are each partitioned into the following suboperands:

$$X1 = X1H\ 2^{r} + X1L\ 2^{0}$$

$$X2 = X2H\ 2^{r} + X2L\ 2^{0}$$

where XH's and XL's represent the most and least significant suboperands. Then the complete multinomial for two input variables can be partitioned into the following subfunctions:

$$Y = Y1\ 2^{2r} + Y2\ 2^{r} + Y3\ 2^{r} + Y4\ 2^{r} + Y5\ 2^{r} + Y6\ 2^{0}$$

where

$$Y1 = W3\ X1H\ X2H + W4\ X1H^{2} + W5\ X2H^{2}$$

$$Y2 = W3\ X1H\ X2L$$

$$Y3 = W3\ X1L\ X2H$$

$$Y4 = W1\ X1H + 2\ W4\ X1H\ X1L$$

$$Y5 = W2\ X2H + 2\ W5\ X2H\ X2L$$

$$Y6 = W0 + W1\ X1L + W2\ X2L + W3\ X1L\ X2L + W4\ X1L^{2} + W5\ X2L^{2}$$

Note that each subfunction includes a unique pair of input suboperands. The number of subfunctions, q, is given by

$$q = d^{2} + 2(d - 1)!$$

where d is the number of input suboperands.

A memory of only 2^{2r} x m_i bits is needed to contain each value of subfunction YI (I = 1, 2, ..., q), Figure 2. The range for the number of memory output bits, m_i, required to represent the value of subfunction YI is given by

$$(2r + n) \leq m_i \leq (2r + n + 2)$$

where n is the number of bits in the coefficient. The two additional bits are generated by multi-term addition.

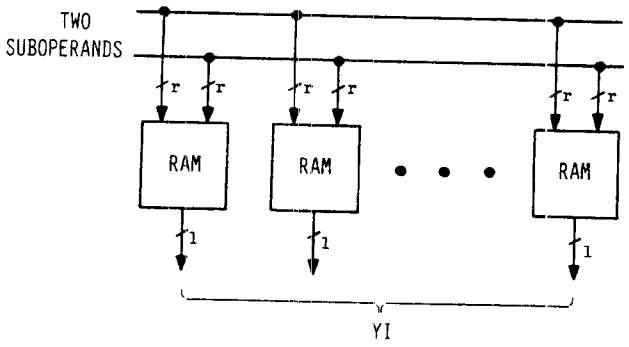

Figure 2. Generation of a subfunction value using RAMs.

A novel implementation of the complete multinomial for two n-bit input variables and six n-bit coefficients is shown in Figure 3. Note, the absence of multipliers. An array of 2^{2r} x m_i-bit memories (MEMs 1-6) is configured in parallel to generate the m_i-bit values of subfunctions which are calculated by an external host computer and downloaded to the memory array. Next, the values of subfunctions from the memory array are summed by a parallel adder to yield the element output Y. The approach based on partitioning multinomial arithmetic can easily be extended to implement ALN elements for large operand lengths. For example, the first design requires memory bits in the order of 2^{2n} and this memory requirment can be reduced into memory bits in the order of 2^{2r} by partitioning input variables.

SUMMATION OF PARTITIONED MULTINOMIALS

A carry-save adder network [4] for the summation of the values of subfunctions is shown in Figure 4, where "asterisks" represent binary digits. A subfunction matrix of the values of six subfunctions is reduced by carry save adders (full adders) into a two-row matrix. The two rows are then summed by a carry lookahead (CLA) adder to yield the element output Y. To minimize the number of stages required for summation, it is important to reduce the height of each matrix according to a specific threshold

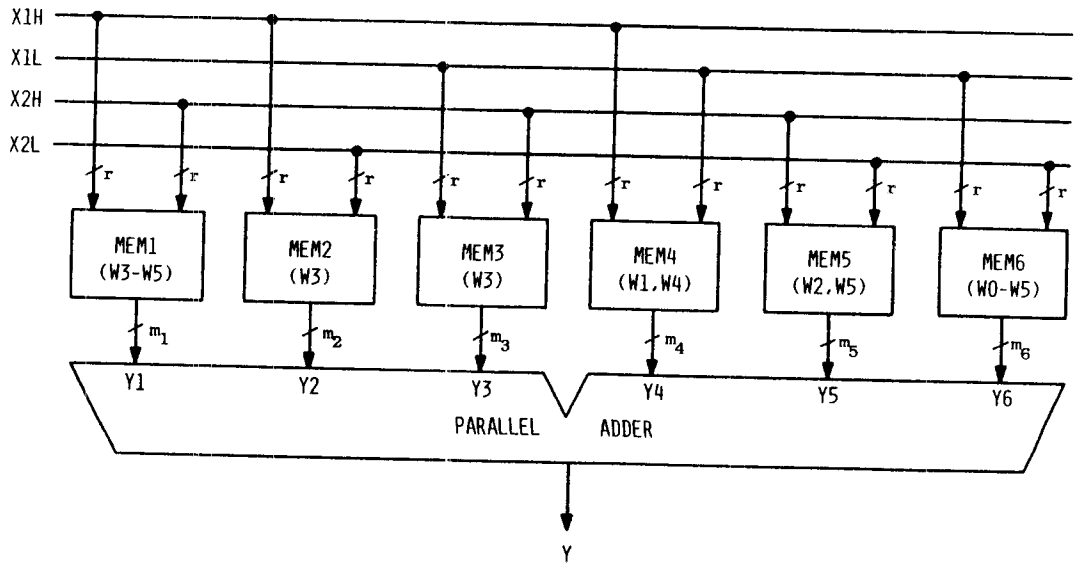

Figure 3. A novel implementation of an ALN element using a memory array and a parallel adder.

value. The ideal height of the i-th matrix, h_i, is given by

$$h_i = \left\lfloor \frac{3}{2} h_{i+1} \right\rfloor$$

where [X] represents the greatest integer equal to or less than X. The carry-save adder network in Figure 4 has the ideal height sequence of 6, 4, 3, and 2.

Since large random logic CLA adders are extremely difficult to implement in integrated circuits, the subfunction matrix needs to be reduced into a smaller two-row matrix. Note that the CLA adder length is reduced at the expense of adders which are used in the rightmost portions of the matrices, Figure 4.

The total number of adder input bits, k, is given by

$$k = \sum_{i=1}^{q} m_i$$

HARDWARE REQUIREMENT AND TIME DELAY

The total hardware requirement for the first approach using multipliers, memories, and a parallel adder are:

- Three n x n-bit multipliers
- Two 2^n x 2n-bit memories
- Three 2^{2n} x 3n-bit memories
- One 14n-bit parallel adder

The total time delay ,Tl_{tot}, for the typical ALN element is:

$$Tl_{tot} = Tl_{mul} + Tl_{mem} + Tl_{add}$$

where Tl_{mul}, Tl_{mem}, and Tl_{add} represent multiplier delay, memory access time, and adder delay, respectively.

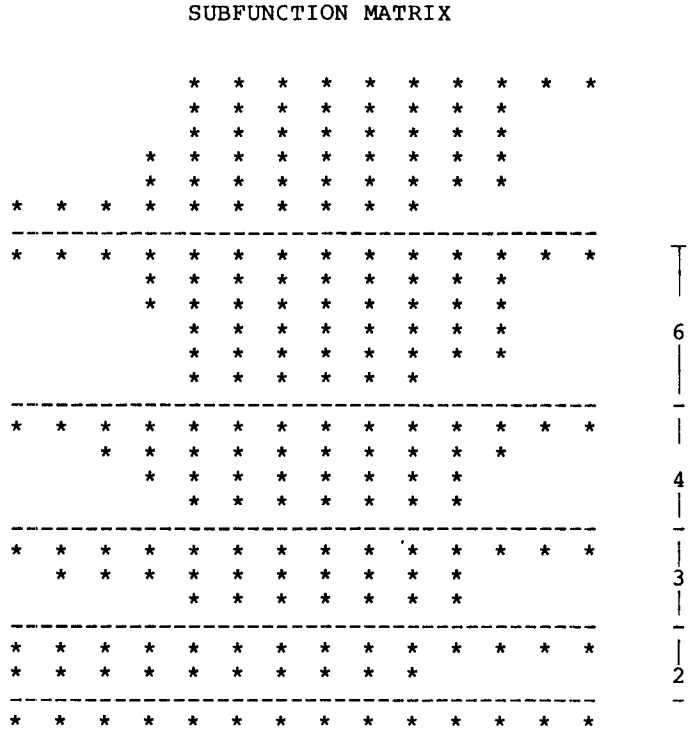

Figure 4. Summation of the subfunction values using a carry-save adder network and a CLA adder.

The total hardware requirement for the proposed ALN element using a memory array and a parallel adder are:

. k 2^{2r}-bit memories

. One k-bit parallel adder

The total time delay, $T2_{tot}$, for the novel ALN element is:

$$T2_{tot} = T2_{mem} + T2_{add}$$

Compared with the former approach, the latter is clearly advantageous for faster operation and less hardware.

The regular and iterative structure of the memory array offers an efficient VLSI implementation for large ALN elements. The use of a memory array also meets the requirement of minimizing the number of possible cell types in a VLSI implementation.

CONCLUSION

In this paper several new approaches to implement a high-speed ALN element using a memory array and a parallel adder have been presented. The table lookup technique has been extended to eliminate multipliers by simply partitioning the input variables into several suboperands. It also has been shown that both time delay and hardware requirments are reduced significantly.

ACKNOWLEDGEMENT

The authors wish to thank Y. P. Foo of University of South Carolina for his invaluable assistance.

REFERENCES

1. A. G. Ivakhnenko, "Polynomial Theory of Complex Systems," IEEE Trans. Systems, Man and Cybernetics, vol. SMC-1, pp. 364-378, Oct. 1971.

2. R. D. Bonnell and P. A. Karnazes, "System Identification Techniques Using the Group Method of Data Handling," 6-th IFAC Symp. on System Identification and Parameter Estimation, Washington, DC, June 1982.

3. H. Kobayashi, Y. P. Foo, and R. D. Bonnell, "Multinomial Arithmetic for a High-Speed Adaptive Learning Network Element," IEEE Southeastcon'82, Destine, FL, Apr. 1982.

4. H. Kobayashi and P. H. Chen, "Automating Design of Carry-Save Adder Networks," IEEE CICC'83, Rochester, NY, May 1983.