Communications

Comments on "Residue Arithmetic VLSI Array Architecture for Manipulator Pseudo-Inverse Jacobian Computation"

Çetin K. Koç

Abstract—**The pipelined array architecture given by Chang and Lee for the mixed-radix conversion problem is not as efficient and suitable for VLSI implementation as the authors claim. We identify the shortcomings of the design and then give an efficient systolic/wavefront array that requires fewer hardware resources.**

I. Introduction

The pipelined array architecture given in [1, Section VI-C] for the mixed-radix conversion (MRC) problem is not as efficient and VLSI implementable as the authors claim. The first weak point of the design is that it requires *broadcasting*. Second, the computation scheme proposed is not *regular*, and the design becomes more complicated and less efficient as the number of inputs grows.

The broadcasting technique is one of the most obvious ways to make multiple copies of an element. Furthermore, broadcasting may also provide schedules with fewer time steps. For these reasons, VLSI computing systems are allowed to have broadcast in some cases (see *semisystolic arrays* in [4]). However, regularity is the most important characteristic of VLSI architectures. The design given by Chang and Lee does not take into account or exploit the regularity of the computations required by the MRC algorithm. We show that the triangular array of buffers and the binary tree structure for the final summation are unnecessary, and we give a computational scheme that simplifies the array by pipelining the computation of the mixed-radix coefficient and the final summation.

Koç and Cappello give several time-optimal and space-time-optimal systolic arrays for computing a process dependence graph corresponding to the MRC algorithm [3]. The process dependence graph is easily obtained by considering the steps of the algorithm. The original graph requires broadcasting; however, with small modifications, one can easily obtain a computational graph without broadcasting. The nodes of the resulting graph become dependent only on the neighboring nodes. The arrays and their corresponding schedules are then found by embedding the process dependence graph in spacetime. The arrays given in [3] compute the mixed-radix coefficients of the residue number. In the following, we give a two-dimensional systolic array implementing the MRC algorithm to compute the binary number X given its residue representation.

II. Systolic Mixed-Radix Conversion

Let $\{m_1, m_2, \ldots, m_L\}$ be the set of prime (or relatively pairwise prime) moduli. We are given the remainders x_i modulo m_i of a weighted number X, i.e.,

$$X \equiv x_i \pmod{m_i} \quad \text{for} \quad 1 \le i \le L.$$

The MRC algorithm computes the mixed-radix coefficients u_i of X such that

$$X = u_1 + u_2 m_1 + u_3 m_1 m_2 + \cdots + u_L m_1 m_2 \cdots m_{L-1}. \quad (1)$$

We define the constants c_{ij} as the multiplicative inverse of m_i modulo m_j, i.e.,

$$c_{ij} m_i \equiv 1 \pmod{m_j} \quad (2)$$

for all $i = 1, 2, \ldots, L-1$ and $j = i, i+1, \ldots, L$. The MRC algorithm computes a triangular table (multiplied difference table [5]) of values where the diagonal entries are the mixed-radix coefficients. The details of the MRC algorithm can be found in [6], [2]. The computation of the weighted number X is then achieved using (1). Let $w_1 = 1$ and

$$w_i = m_1 m_2 \cdots m_{i-1} \quad (3)$$

for $i = 2, 3, \ldots, L$. We can then write (1) as

$$X = u_1 w_1 + u_2 w_2 + u_3 w_3 + \cdots + u_L w_L. \quad (4)$$

The computation of X using (4) can be included in the computation of the entries of the multiplied difference table. These two steps are separated in Chang and Lee's design, which computes the mixed-radix coefficients on a triangular array of cells and the final summation using a binary tree of cells. This approach yields an unnecessarily complicated array since the interface between these two parts requires as many as $L(L-1)/2$ buffers.

We propose an alternative scheme that computes the binary number X using (4) as soon as the first mixed-radix coefficient u_1 becomes available. The steps of the algorithm are illustrated below, where $[m_i]$ stands for (mod m_i). For $L = 5$, we start with $u_i = x_i$ for $1 \le i \le L$, and compute the binary number X as follows:

$$\begin{array}{ll}
u_1 := x_1 & u_4 := x_4 \\
X := u_1 w_1 & u_4 := (u_4 - u_1) c_{14} \; [m_4] \\
u_2 := x_2 & u_4 := (u_4 - u_2) c_{24} \; [m_4] \\
u_2 := (u_2 - u_1) c_{12} \; [m_2] & u_4 := (u_4 - u_3) c_{34} \; [m_4] \\
X := X + u_2 w_2 & X := X + u_4 w_4 \\
u_3 := x_3 & u_5 := x_5 \\
u_3 := (u_3 - u_1) c_{13} \; [m_3] & u_5 := (u_5 - u_1) c_{15} \; [m_5] \\
u_3 := (u_3 - u_2) c_{23} \; [m_3] & u_5 := (u_5 - u_2) c_{25} \; [m_5] \\
X := X + u_3 w_3 & u_5 := (u_5 - u_3) c_{35} \; [m_5] \\
 & u_5 := (u_5 - u_4) c_{45} \; [m_5] \\
 & X := X + u_5 w_5.
\end{array}$$

The data dependencies among the entries above easily lend themselves to systolic implementation. This observation was made in [3], and several time-optimal and space-time-optimal linear and two-dimensional systolic arrays were introduced. We will not elaborate on the properties and the variations of the embedding technique. The interested reader is referred to the aforementioned paper. However, by noticing the data dependence properties of the multiplied difference table, a two-dimensional systolic array can be given for the mixed-radix conversion problem (see *Embedding* E_5 in [3]). The design is more efficient in terms of time and area (the amount of hardware resources and ROM space required for modular arithmetic computations). The systolic array and the cell definitions are given in Figs. 1 and 2, respectively. The cells are designed in the same way as in Chang and Lee's design. The circular cells compute the mixed-radix coefficients, and the square cells accumulate the binary

Manuscript received April 10, 1990. This work was supported in part by the U.S. Army Research Office under Grant DAAL03-91-G-0106.

The author is with the Department of Electrical Engineering, University of Houston, Houston, TX 77204.

IEEE Log Number 9100989.

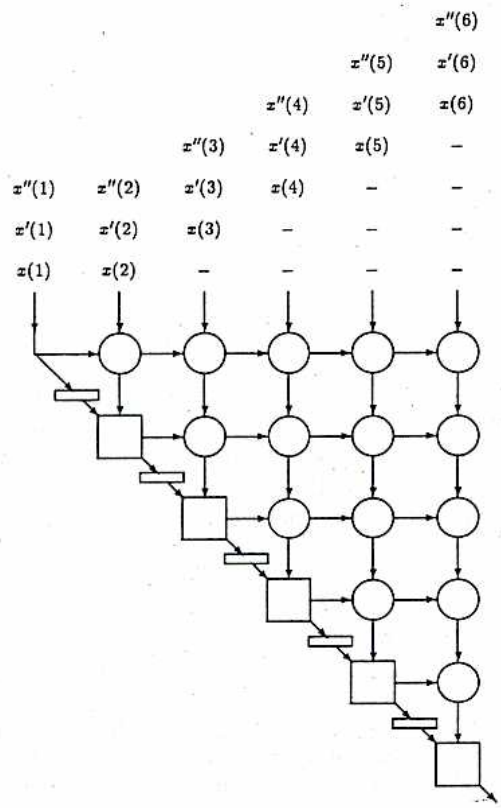

Fig. 1. Systolic array for mixed-radix conversion.

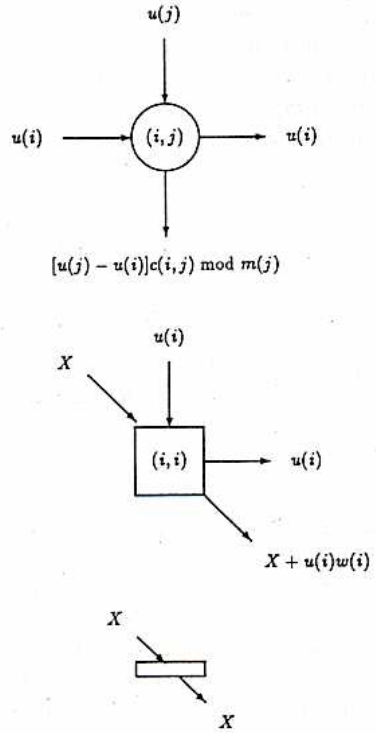

Fig. 2. Cell definitions.

value X starting with the initial value $X = u_1 w_1 = x_1$. The constants c_{ij} [see (2)] and w_i [see (3)] are precomputed and saved in the circular and the square cells, respectively.

Unlike the design by Chang and Lee, the systolic array is more suitable for VLSI implementation since it requires *only local communication*. Their array computes the mixed-radix coefficient u_i at level i and then broadcasts this element to every cell in the next level. The new array eliminates the broadcasting by passing u_i to the neighboring cell and skewing the input stream as shown in Fig. 1. Furthermore, the new systolic array requires fewer buffers than Chang and Lee's array. We eliminate most of the buffers by coupling the computation of X using (1) with the computation of the entries of the multiplied difference table.

There is a penalty paid for the systolic design: the latency of the systolic array (the number of time steps required to produce the first result) is now equal to $2(L-1)$. Even though the latency of the array in [1] is only $L + \lceil \log_2 L \rceil$, the duration of the clock cycle has to be long enough to allow signal propagation required for broadcasting. In effect, due to its broadcasting requirements, the total time required by Chang and Lee's design is much longer. Elimination of broadcasting in order to reduce the total time is one of the basic principles of VLSI design philosophy. This rule is apparently ignored in the design of Chang and Lee.

In Table I, we compare the area and time requirements of the systolic design to those of Chang and Lee's design. In this table, we include the semisystolic design, i.e., the systolic design that allows broadcasting. The semisystolic array is easily obtained from the systolic array in Fig. 1 by removing the data dependencies between the cells in the same level and eliminating the buffers between the square cells. The input for the semisystolic array is not skewed and enters the array in parallel. The area and time requirements for the semisystolic design are given for comparison proposes, and also to illustrate the merits of the computational scheme proposed in this comment. The semisystolic array can be useful in case the penalty for broadcasting is insignificant, e.g., when L is small.

TABLE I
COMPARISON OF THE DESIGNS

	Chang & Lee	Systolic	Semisystolic
Mod m Subtractors	$\frac{L}{2}(L-1)$	$\frac{L}{2}(L-1)$	$\frac{L}{2}(L-1)$
Mod m Multipliers	$\frac{L}{2}(L-1)$	$\frac{L}{2}(L-1)$	$\frac{L}{2}(L-1)$
Adders	$L-1$	$L-1$	$L-1$
Multipliers	L	$L-1$	$L-1$
Buffers	$\frac{L}{2}(L-1)$	$L-1$	–
Latency	$L + \lceil \log_2 L \rceil$	$2L-1$	L
Period	1	1	1

REFERENCES

[1] P. R. Chang and C. S. G. Lee, "Residue arithmetic VLSI array architecture for manipulator pseudo-inverse Jacobian computation," *IEEE Trans. Robotics Automat.*, vol. 5, no. 5, pp. 569–582, Oct. 1989.

[2] D. E. Knuth, *The Art of Computer Programming*, vol. 2, *Seminumerical Algorithms*, 2nd ed. Reading, MA: Addison-Wesley, 1981.

[3] Ç. K. Koç and P. R. Cappello, "Systolic arrays for integer Chinese remaindering," in M. D. Ercegovac and E. Swartzlander, Eds., *Proc. 9th Symp. Comput. Arithmetic* (Santa Monica, CA, Sept. 6–8 1989). Los Alamitos, CA: IEEE Computer Society, 1989, pp. 216–223.

[4] H. T. Kung, "Why systolic architectures?," *IEEE Comput.*, vol. 15, no. 1, pp. 37–45, Jan. 1982.

[5] J. D. Lipson, *Elements of Algebra and Algebraic Computing*. Reading, MA: Addison-Wesley, 1981.

[6] N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*. New York: McGraw-Hill, 1967.