Transactions Briefs

Bit-Level Pipelined Digit-Serial Array Processors

A. Aggoun, M. K. Ibrahim, and A. Ashur

***Abstract*—A new architecture for high performance digit-serial vector inner product (VIP) which can be pipelined to the bit-level is introduced. The design of the digit-serial vector inner product is based on a new systematic design methodology using radix-2^n arithmetic. The proposed architecture allows a high level of bit-level pipelining to increase the throughput rate with minimum initial delay and minimum area. This will give designers greater flexibility in finding the best tradeoff between hardware cost and throughput rate. It is shown that sub-digit pipelined digit-serial structure can achieve a higher throughput rate with much less area consumption than an equivalent bit-parallel structure.**

A twin-pipe architecture to double the throughput rate of digit-serial multipliers and consequently that of the digit-serial vector inner product is also presented. The effect of the number of pipelining levels and the twin-pipe architecture on the throughput rate and hardware cost are discussed. A two's complement digit-serial architecture which can operate on both negative and positive numbers is also presented.

I. Introduction

Bit-serial computations have been studied extensively during the last decade. Complete very large scale integration (VLSI) design methodologies based on bit-serial arithmetic have been built, particularly for the VLSI implementation of digital signal processing (DSP) algorithms [1], [2]. However, the disadvantage of the bit-serial systems is that they are considered to be very slow for many application such as radar, sonar, and video and image processing. For higher throughput DSP applications, however, it is clear that a move toward higher bit-width operations is necessary, where the solution often used is bit-parallel computation. However, the bit-parallel systems require a very large amount of silicon area, communication overhead, and pin-out.

To avoid the disadvantages of the bit-serial and the bit-parallel computation, the concept of digit-serial implementations has been proposed in recent years [3]–[5]. The digit-serial systems are ideal for moderate speed applications, for which the bit-serial style is too slow, and the bit-parallel style is faster than necessary.

Digit-serial algorithms have been proposed using on-line arithmetic (most significant digit first) based on signed-digit number representation [6]. The drawbacks of this approach are the increased size of the computational elements and the overhead of data conversion [5].

Also, several approaches have been proposed to design digit-serial architectures based on two's complement number representation, which are summarized in Fig. 1. Most of the proposed design methodologies use the *bit-level cellular arrays* as their *starting* point [3]–[5]. The approach proposed by Wu and Cappello [13] is based on partitioning a bit-parallel multiplier into K blocks, then applying the space-time transformation proposed by Chen [12] to each block. A similar architecture has been previously proposed by Smith *et al.* [3]. Although the resulting architectures are K times faster than the bit-serial structures, the initial delay is still that of the bit-serial (i.e., $2N$, where N is the wordlength). Another drawback is the additional cost needed for the switching circuitry. Two other methods for the design of digit-serial structures were proposed later, which help overcome the problem of the initial delay. The first is to start with a bit-parallel structure and then use folding to obtain the digit-serial architecture [5]. The second approach is to start with a bit-serial architecture and then use unfolding to obtain the digit-serial one [4]. The major drawback of the architectures based on these approaches is that they cannot be pipelined at a sub-digit level, which has severely limited their throughput. This could be a major obstacle for high-speed applications. In digit-serial systems, the throughput rate often represents the overriding factor dictating system performance. The main reason why these structures cannot be pipelined is due to the existence of carry feedback loops, which are impossible to pipeline. Modification of these designs is very complex and could ultimately mean a redesign because they are specified in terms of gates and full adders by the initial bit-parallel (bit-serial) architecture and the folding (unfolding) process.

Recently, a new approach to the design of digit-serial structures has been proposed based on radix-2^n arithmetic [8], [9]. The advantages of the radix approach are: 1) it has enabled for the first time the design of functionally correct digit-serial structures that are based on two's complement representation, directly and in a hierarchical way without the need for bit-parallel or bit-serial designs as an initial starting point; 2) it is more general than the bit-level cellular arrays approach due to the fact that more designs can be derived from the radix approach; 3) it allows for *the first time* the direct application of all *the existing synthesis* methods in designing digit-serial structures; and 4) it only specifies the functionality of the basic cell and hence any internal architecture can be used so long as it satisfies the functionality specification of the cell.

Recently, the first digit serial structures that can be pipelined at sub-digit level based on radix-2^n arithmetic has been reported by the authors [7]. It involves a novel carry feedforward technique which was proved functionally correct using radix-2^n arithmetic. The use of carry feed-forward has solved a major bottleneck of the carry feedback loops of existing designs. Also, the flexibility offered by the radix-2^n approach in choosing the cell architecture has enabled the design of the basic cells using carry save arithmetic, which is faster and requires less area.

The possibility of high degree of pipelining offered by the structure in [7] will increase the throughput rate of the digit-serial systems. However, because of the interconnections between adjacent cells, the number of pipelining levels is limited by the area constraint as well as the initial delay. The reason is that the initial delay of this structure increases rapidly with the degree of pipelining and the number of digits per word.

Some DSP problems require a rapid response from a system and cannot tolerate significant initial delay. In this paper, a new radix-2^n algorithm for VIP is proposed. Also, a systematic design methodology is proposed to design bit-level pipelined digit-serial VIP directly from the radix-2^n algorithm. Furthermore, a new cell architecture is designed, which allows bit-level pipelining in such a way that the initial delay becomes independent of the number of

Manuscript received July 21, 1996; revised July 21, 1996. This work was supported by EPSRC, U.K. This paper was recommended by Associate Editor F. J. Kurdahi.

A. Aggoun and M, K. Ibrahim are with the Faculty of Computing Sciences and Engineering, De Montfort University, Leicester LE1 9BH, U.K.

A. Ashur is with the Department of Electrical and Electronic Engineering, University of Nottingham, Nottingham NG7 2RD, U.K.

Publisher Item Identifier S 1057-7130(98)05060-5.

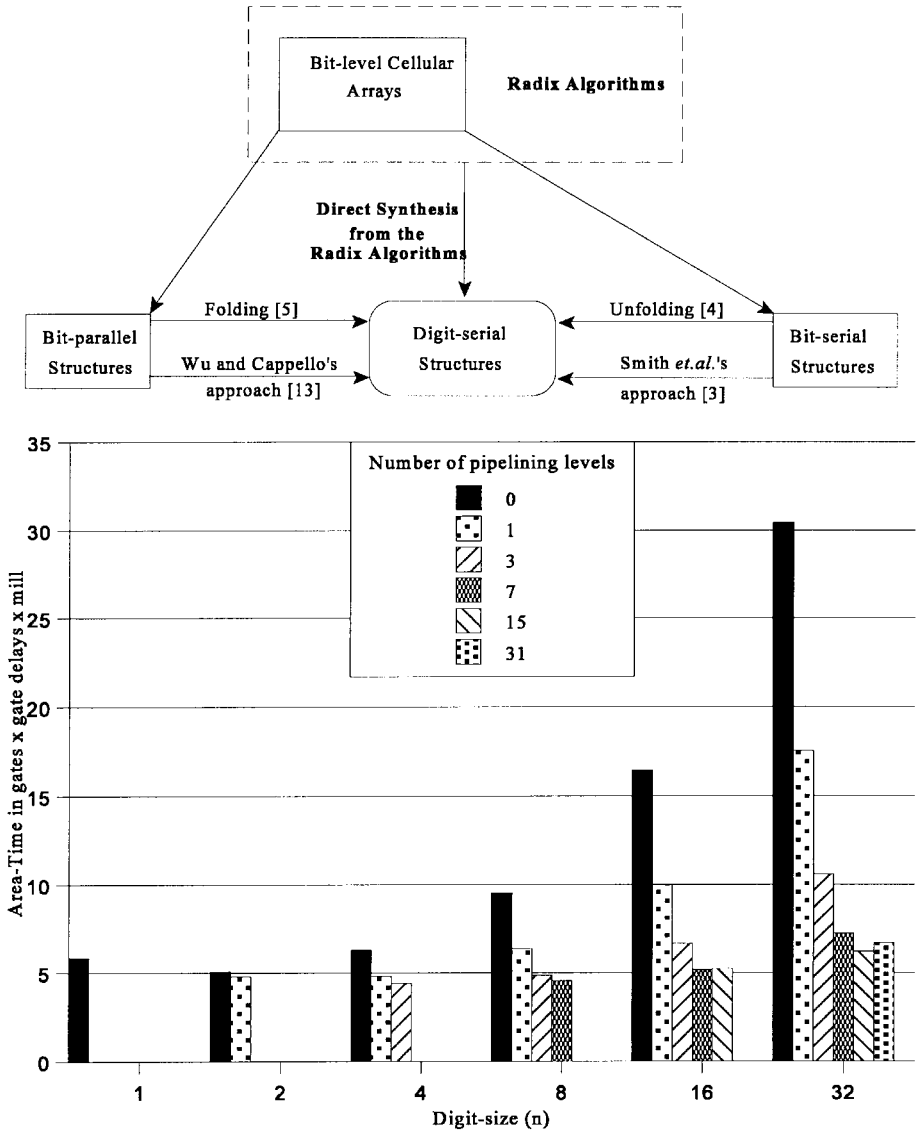


Fig. 1. Different approaches used to design digit-serial structures.

digits. In this case, the degree of pipelining is only limited by the area constraint.

The performance of the proposed designs are evaluated with respect to area, time, and area-time complexity. It is shown that sub-digit pipelined digit-serial architectures designed using the proposed radix-2^n algorithm can achieve a *better* throughput rate than the bit-parallel structure with much lower hardware cost. As a consequence, the digit-serial structures can no longer be considered as a compromise approach between the bit-serial and the bit-parallel structures but as one of the main contenders in the design of array architectures for high-speed applications.

This paper is organized as follows. Section II describes the radix-2^n approach, where a design of the digit-serial multiplier reported in [7] is described. In Section III, a systematic design methodology of digit-serial VIP based on the proposed radix-2^n algorithms is presented. In this section, new radix-2^n algorithms are presented. The design and optimization of the basic cell architecture based on an array multiplier and the design of the first bit-level pipelined digit-serial adder are also described. Evaluation of the hardware cost and the throughput rate of the proposed digit-serial VIP with respect to the digit size and the number of pipelining levels are presented in Section IV. A twin-pipe and a two's complement digit-serial multipliers are presented in Sections V and VI, respectively. Conclusions are drawn in Section VII.

II. The Radix-2^n Approach

The radix-2^n algorithm can be considered as a generalization of the binary arithmetic where the simplest building block is an AND gated full adder (FA) [8]. The basic cell of the radix-2^n multiplication algorithm, for instance, multiplies two n-bit digits u_i and v_j and adds the product to two other n-bit digits $s_{\rm in}$ and $c_{\rm in}$ viz.,

$$\{c_{\rm out}, s_{\rm out}\} = u_i v_j + s_{\rm in} + c_{\rm in} \tag{1}$$

The result will always be a two digit number, the most significant digit (MSD) is the carry digit, $c_{\rm out}$, and the least significant digit (LSD) is the sum digit, $s_{\rm out}$. This implies that any bit-level architecture can be generalized to perform a particular radix-2^n algorithm by simply replacing the bit-level basic block with a radix-2^n basic cell. Furthermore, the bit-level DG's which have been proposed for multiplication and VIP operations [10]–[12] can all be generalized for radix-2^n multiplication and VIP operations. One of the most significant advantages of the radix-2^n approach which was indicated

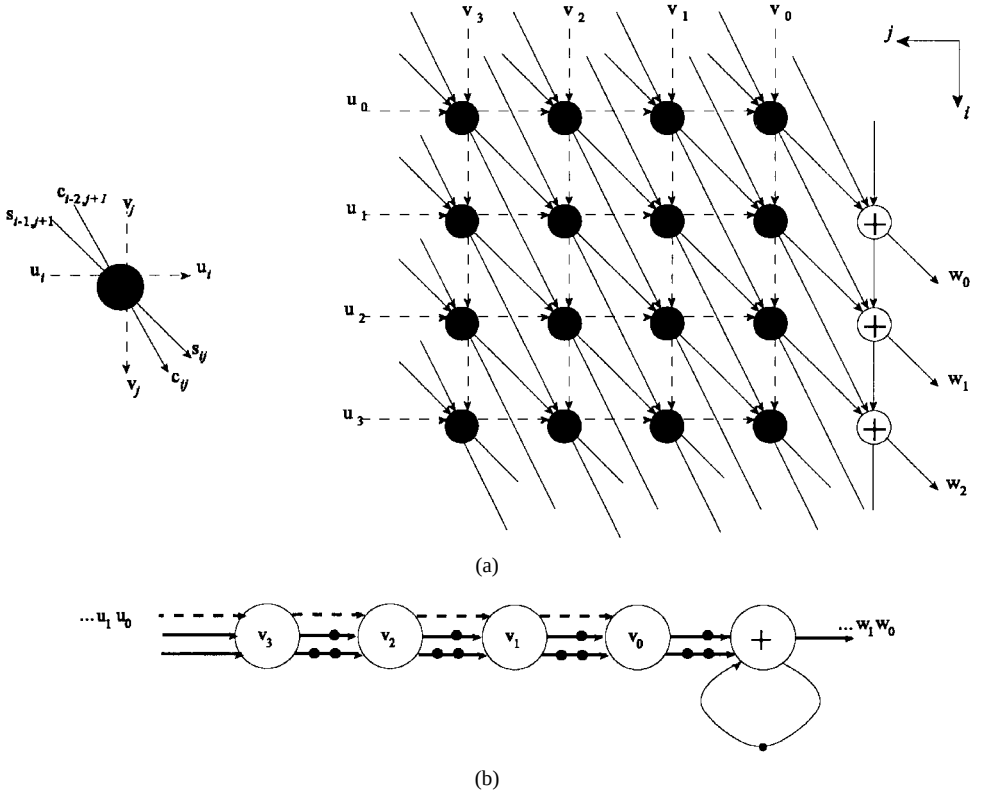


Fig. 2. Design of the digit-serial multiplier reported in [7]. (a) Dependency graph. (b) Digit-serial multiplier derived by projecting the graph onto vertical axis.

earlier is that the digit-serial structures can be, for the first time, derived *directly* from such radix *dependency* graphs.

The design of the digit-serial multiplier can be derived by linear projection of the DG's of the radix-2^n multiplication algorithm according to the classical theoretical framework of regular array architectures synthesis [2]. Depending on the way carry digits are transmitted to the next most significant digit position, there are several possible graphs, one of which is given in Fig. 2. In this graph, the carry digits originating in node (i,j) are transmitted forward to node $(i+2, j-1)$, where i and j are the horizontal coordinate axes. This radix-2^n DG can be seen as the radix-2^n generalization of the bit-level DG's (graph 3) presented by Privat [11]. A bit-level pipelined digit-serial multiplier can be derived by projecting this graph in the direction of $[1,0]^T$ (Fig. 2(b)). It is similar to the digit-serial multiplier reported by the authors [7], where the basic cell based on the carry save array multiplier is shown in Fig. 3. The main disadvantage of this cell architecture is that the initial delay of the digit-serial multiplier increases rapidly with the number of pipelining levels and the number of digits per word. In what follows, new radix-2^n algorithms are presented. In this paper, the proposed radix-2^n algorithms are used to design bit-level pipelined digit-serial vector inner product with minimal initial delay computation.

III. Design Methodology

As mentioned earlier, all of the reported design methodologies can be adopted to design digit-serial structures directly from the radix-2^n algorithms. The design methodology adopted in this paper is based on the one proposed by Kung [2]. It can be described by three main steps, namely: 1) writing the algorithm using the radix-2^n arithmetic; 2) generating the dependency graph and selecting the projection direction; and 3) design and optimization of the radix-2^n cell.

A. Writing the Algorithm Using the Radix-2^n Arithmetic

To illustrate the concept, the example of the VIP algorithm will be adopted. Here, new radix-2^n algorithms are developed based on splitting the digit partial products prior to accumulation, as opposed to the multiply accumulate operation used in Section II.

New Radix-2^n Vector Inner Product Algorithms: The evaluation of the inner product of two vectors is one of the most important arithmetic computations in the field of digital signal processing. In fact, both digital filters and DCT processors require the computation of VIP's. In the last decade, several bit-level VIP architectures have been reported in the literature [10].

The inner product of vector $U = (U_0, U_1, \ldots, U_{M-1})$ and $V = (V_0, V_1, \ldots, V_{M-1})$ can be obtained by multiplying one pair of numbers (U_m, V_m) and adding their product to an accumulating result. This is described at the word level by a simple recursion of the form $W_m \leftarrow W_m + U_m V_m$. Assume that the elements U_m and V_m are unsigned numbers and can be divided into K digits of n-bit each. Let u_{im} and v_{jm} represent the ith and jth digits of U_m and V_m, respectively, viz.,

$$U_m = \sum_{i=0}^{K-1} u_{im} 2^{in} \quad \text{and} \quad V_m = \sum_{j=0}^{K-1} v_{jm} 2^{jn} \tag{2}$$

The product W_m of the two numbers U_m and V_m can be computed according to the following equation:

$$W_m = \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} 2^{(i+j)n} u_{im} v_{jm}. \tag{3}$$

Equation (3) can be written in a recursive manner using (1), from which the DG shown in Fig. 2 can be derived (m is kept constant). Since the partial products $u_{im} v_{jm}$ are $2n$-bit number, $2n$-bit adders are required for the accumulation in each radix-2^n cell. In what follows, a whole new set of radix-2^n algorithms are derived using

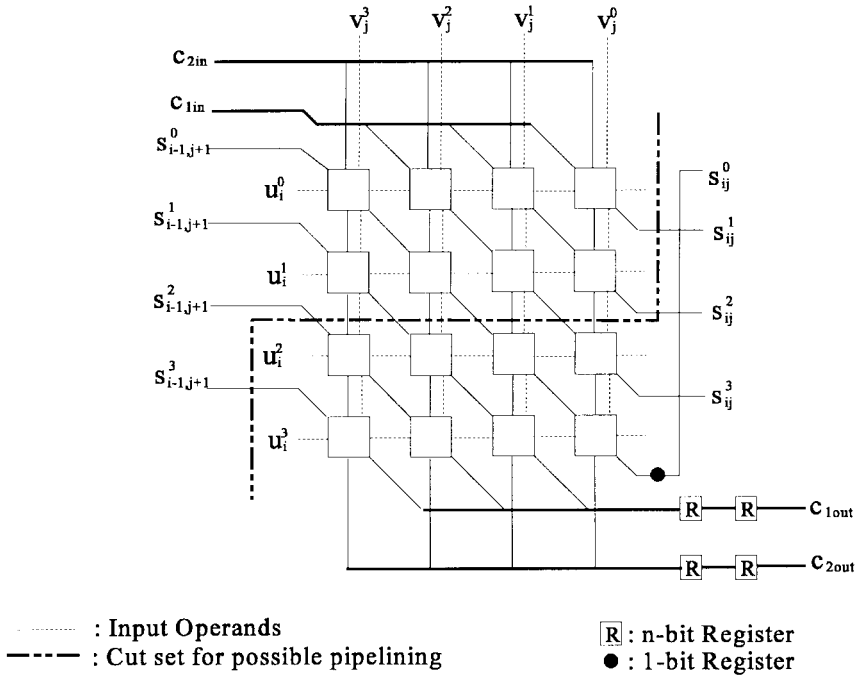


Fig. 3. Cell architecture of the digit-serial multiplier reported in [7].

the fact that these partial products can be partitioned into the most significant digit (MSD), $(u_{im}v_{jm})_{\rm MSD}$, and the least significant digit (LSD), $(u_{im}v_{jm})_{\rm LSD}$. The product W_m can be rewritten as

$$W_m = \sum_{i=0}^{K-1}\sum_{j=0}^{K-1} 2^{(i+j)n}((u_{im}v_{jm})_{\rm MSD}2^n + (u_{im}v_{jm})_{\rm LSD}). \quad (4)$$

The new radix-2^n algorithms are shown to result in a more efficient implementation of the radix-2^n cell by reducing the length of the adders required for the accumulation.

In the index space (i,j,m), $(i+j)$ represents a family of vertical planes and determines the significance. The most and least significant digits of the partial products $(u_{im}v_{jm})_{\rm MSD}$ and $(u_{im}v_{jm})_{\rm LSD}$ are computed at the plane $(i+j)$. Since the term $(u_{im}v_{jm})_{\rm MSD}$ is on the next highest significance with respect to $(u_{im}v_{jm})_{\rm LSD}$, these two terms cannot be added within the same plane $(i+j)$. This implies that in order to add the terms with the same significance, the MSD generated at the plane $(i+j)$ must be added to any of the LSD's on the next highest significance at the $(i+j+1)$ plane. If (4) is to be implemented in a recursive manner, the MSD's and the LSD's with the same significance must be made to accumulate along the same path, which implies that one of these terms must be transferred to the path of the other. Hence, there are two different approaches to implement (4) depending on the transfer of either the MSD's or the LSD's of the partial products. It should be noted that these transfers can be performed on the same horizontal plane m or from one horizontal plane to another. Each transfer represents one algorithm which can be formulated in a recursive form using space-time indices. However, in this paper only one of these algorithms is considered for the design of a digit-serial VIP that can be pipelined at the bit-level. It is shown that using the approach developed by Kung [2], a systematic design methodology of digit-serial vector inner products can be developed. Other digit-serial vector inner product can be design following the same methodology.

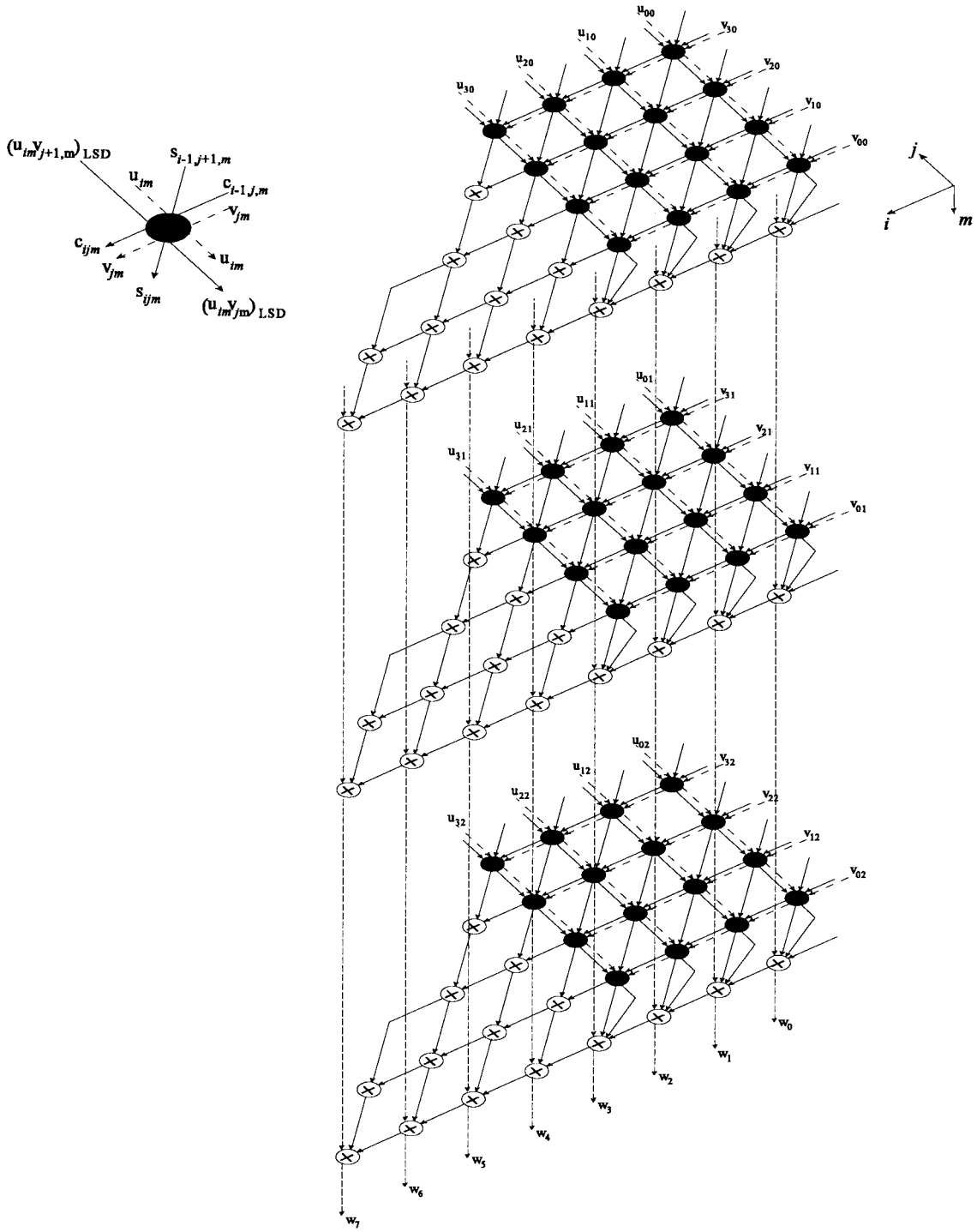
The Radix-2^n Approach to the Design of Digit-Serial Vector Inner Products The proposed bit-level pipelined digit-serial VIP is designed using the algorithm which considers the transfer of the LSD's of the partial products on the same horizontal plane m, from spatial coordinates $(i,j+1,m)$ to spatial coordinates (i,j,m). It can be easily shown that (4) can be computed by the following recurrence:

$$\{c_{ijm}, s_{ijm}\} = (u_{im}v_{j+1,m})_{\rm LSD} + (u_{im}v_{jm})_{\rm MSD} + s_{i-1,j+1,m} + c_{i-1,j,m} \quad (5)$$

where s_{ijm} is the accumulating partial sum along the vertical planes $(i+j)$ and c_{ijm} represents the partial carry digit generated at the spatial coordinates (i,j,m). In the notation used in (1) and (5), $\{c_{ijm}, s_{ijm}\}$ denote an r-bit binary number $(r>n)$ of which the n least significant bits are s_{ijm} and the remaining bits are c_{ijm}. Since $(u_{im}v_{jm})_{\rm MSD}$, $(u_{im}v_{jm})_{\rm LSD}$, and s_{ijm}, are n-bit numbers, the total sum of these three numbers is $(2^{n+1}+2^n-3)$ which is $(n+2)$-bit wide. As a result, c_{ijm} is at least 2-bit wide. If $c_{i-1,j,m}$ is 2-bit wide (i.e., its maximum value is 3), the maximum value of the sum, i.e., $\{c_{ijm}, s_{ijm}\}$ in (5) is $(2^{n+1}+2^n)$, which can be represented using $(n+2)$ bits. Since s_{ijm} is n-bits, the output carry digit c_{ijm} remains to be 2-bit This implies that two n-bit adders are required for the accumulation in (5) compared to two $2n$-bit adders in (1).

B. Generation and Mapping of the DG into Digit-Serial VIP

The inner product W of vector U and vector V can be obtained by accumulating w_m following the m axis. The DG, which represents the above computation, is shown in Fig. 4. This DG is embedded in a three-dimensional index space represented by (i,j,m). The multiplication of two elements U_m and V_m is performed on the horizontal planes m. Each node in this graph shows the interaction of the digits within each word u_{im} and v_{jm}. The accumulating sum digits s_{ijm} are connected in the diagonal direction so that the products of the form $(u_{im}v_{jm})_{\rm MSD} + (u_{im}v_{j+1,m})_{\rm LSD}$ are accumulated as indicated by (5). The LSD's are transferred horizontally from left to right while the carry digits c_{ijm} are transferred horizontally following the j direction. In order to form the final digits of the vector inner product W, the partial sum at the node $(i,0,m)$, s_{i0m} is added to



(a)

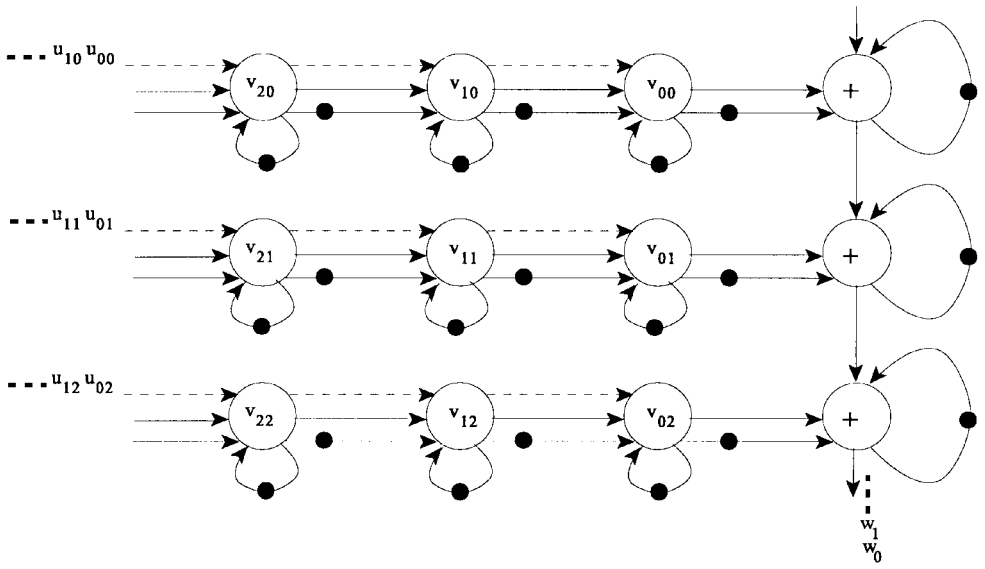
Fig. 4. Design of a digit-serial vector inner product based on the proposed radix-2^n algorithms. (a) Dependency graph.

the LSD of the partial product $(u_{im}v_{0m})_{\mathrm{LSD}}$ and to the partial result w_{m-1} from plane $(m-1)$ as shown in Fig. 4. This is represented by the vertical plane on the right in the DG.

This DG can be projected in several directions to obtain different digit-serial VIP. Considering the projection in the direction of $[1,0,0]^T$, the digit-serial VIP shown in Fig. 4(b) will be obtained. Each row of the VIP shown in Fig. 4(b) performs a digit-serial multiply-add operation. The multiplication of two terms in the inner product is performed using a digit-serial multiplier where the design of the basic cell is the subject of the next section.

C. Design and Optimization of the Basic Cell

The basic cell of the digit-serial multiplier can be implemented using a carry-save array multiplier. The basic cell which computes (5) is shown in Fig. 5 for $n=4$, where the upper indices are used to represent the bit significance. It uses the fact that the full adders at the boundaries of the carry-save array multiplier have free bit positions. The n sum bits produced on the right-hand side of the array multiplier which form the LSD $(u_{im}v_{jm})_{\mathrm{LSD}}$ are transferred to the next cell on the right and added with the MSD $(u_{im}v_{j-1,m})_{\mathrm{MSD}}$. This addition



(b)

Fig. 4. (*Continued.*) Design of a digit-serial vector inner product based on the proposed radix-2^n algorithms. (b) Digit-serial vector inner product derived by projecting the graph onto the direction $[1, 0, 0]^T$.

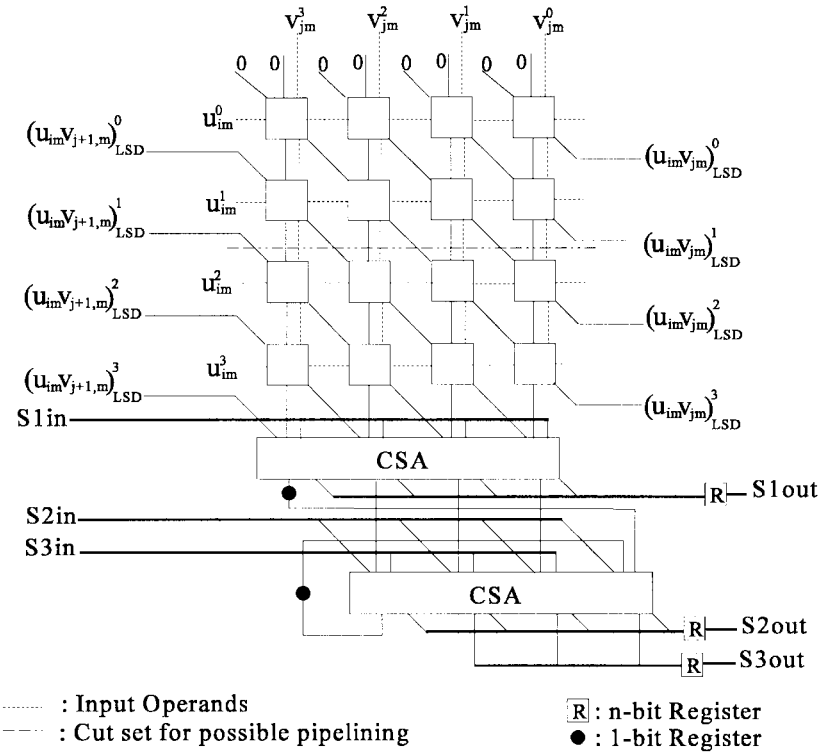


Fig. 5. Cell architecture using array multiplier.

is performed using the empty bit positions on the left-hand side of the array multiplier.

In conventional carry-save array multipliers, a carry propagate adder is required to add the carry and sum bits from the lower row of full adders to form the carry digit. To allow bit-level pipelining, these carry and sum bits from the lower boundary of the array multiplier are summed with the partial sums from the previous cell using two carry save adders as shown in Fig. 5. The partial result from previous cell on the left consists of three digits, $S1_{\rm in}$, $S2_{\rm in}$ and $S3_{\rm in}$. The digit $S1_{\rm in}$ is summed to the two digits from the lower boundary of the array multiplier using the first CSA to produce two 1-digit outputs. One of the two digits, $S1_{\rm out}$, obtained from the first CSA is fed to the next cell on the right. While the second digit is fed down into a second CSA to be summed with the two digits, $S2_{\rm in}$ and $S3_{\rm in}$, coming from the cell on the left, the two carry bits produced by the two CSA's are summed to form the carry digit c_{ijm}. The carry digit c_{ijm} is fed back into the same cell to be added as shown by (5). This is performed using the empty least significant bit position available in the second CSA as shown in Fig. 5. As it can be seen from Fig. 5, the feedback of c_{ijm} will not affect the pipelining, since the delay

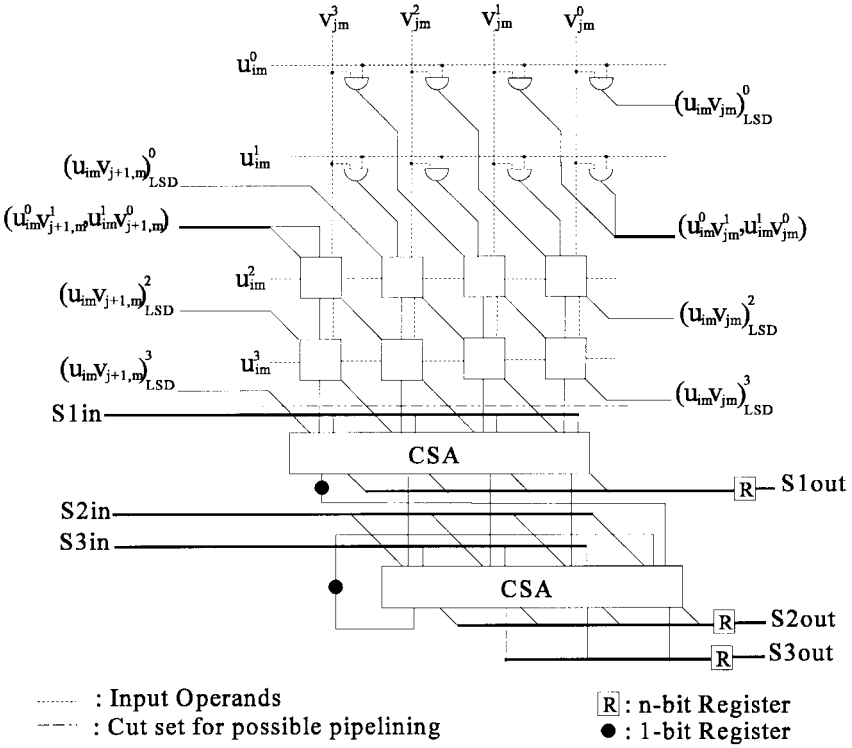


Fig. 6. Optimized cell architecture using array multiplier.

within the loop is one FA delay. It should be noted that the DG can be modified to allow c_{ijm} to be fed forward to the next cell.

The cell architecture shown in Fig. 5 can be made more efficient by removing the top two CSA, since they have empty bit positions. The optimized cell is illustrated in Fig. 6. This does not affect the final result as explained below. It should be pointed out that this description must be seen in conjunction with Figs. 5 and 6. Let k represent the bit significance of $(u_{im}v_{jm})_{\rm LSD}$. From Fig. 5 it can be seen that there are no carries from significance $k=0$ to significance $k=1$. If the bits $u_{im}^0 v_{jm}^1$ and $u_{im}^1 v_{jm}^0$ are not added within the cell (j,m), but fed to be added in the cell $(j-1,m)$ as shown in Fig. 6, the carries to the significance $k=2$ are also equal to 0. The two bits $u_{im}^0 v_{jm}^1$ and $u_{im}^1 v_{jm}^0$ are added to the bit $u_{im}^2 v_{j-1,m}^{n-1}$ ($n=4$ in Fig. 6) using one FA. This implies that the three bits $u_{im}^0 v_{jm}^k$, $u_{im}^1 v_{jm}^{k-1}$, and $u_{im}^2 v_{jm}^{k-2}$ for $k=2$ to $n-1$ can be added using one single FA as shown in Fig. 6.

As shown in Fig. 4(b), a column of adders is required to add the results of the multiplication of each pair of elements U_m and V_m in the inner product to obtain the final result. The addition is perform in two stages. First, the digits obtained from the last cell on the right of the digit-serial multiplier are added to the digits of the partial result, w_{m-1}, obtained from the digit-serial multiplier-accumulator on the top, using an array of CSA's to produce two 1-digit outputs which are propagated down to the next digit-serial multiply-accumulator. The two digits from the last digit-serial multiply-accumulator are added using a digit-serial adder.

D. Bit-Level Pipelined Digit-Serial Adder

The conventional digit-serial adder uses a carry propagate adder (CPA), where the last carry bit is fed back to the same adder. To take full advantage of the high degree of pipelining offered by the proposed cell architectures, the CPA is implemented using a digit-serial pipelined adder. This is achieved by using two CPA's, where the carry bit produced by the first CPA is propagated forward and added to the next digit output using the second CPA as shown in Fig. 7. Note that the only case where carry bit obtained from the second CPA is equal to one is when the current output of the first CPA is 2^n-1, where n is the digit size, and the carry from the previous digit is one. In this case, the carry to the next significant digit should be set to one. A simple circuit that will propagate the correct carry to the second CPA is shown in Fig. 7. The digit from the first CPA is fed to a n-input AND gate to check whether it is equal to 2^n-1. This is performed by using an array of 2-input AND gates arranged in a tree structure as shown in Fig. 7. The resulting bit and the carry bit obtained in the previous cycle are fed to a 2-input AND gate to check whether the current output of the first CPA is 2^n-1 and the carry from the previous digit is one simultaneously. The output bit and the carry of the current digit are fed to a 2-input OR gate to calculate the correct carry bit to be fed to the second CPA.

As can be seen from Fig. 7, there is only one feedback loop within the digit-serial adder. However, the propagation delay within the feedback loop of the carry bit is equal to one AND plus one OR gate delay, and hence the pipelining within the loop is not necessary. The remaining data paths in the digit-serial adder can all be pipelined to the bit-level since they are moving in the same direction.

IV. Pipelining of the Digit-Serial Vector Inner Product

In this section, the performance of the proposed digit-serial inner product is evaluated with respect to the number of pipelining levels for different digit sizes. Prior to comparing the time taken, T, area consumed, A, and area-time measure, AT, for the proposed digit-serial VIP, first the time and area for each component are given. It should be pointed out that both unit time (Δ) and unit area (A_T) considered here are, respectively, the time taken and area required in an inverter circuit [14] which depends on the technology implemented.

1) The time and area taken by a full adder are $T_{\rm FA}=6\Delta$ and $A_{\rm FA}=10A_T$.

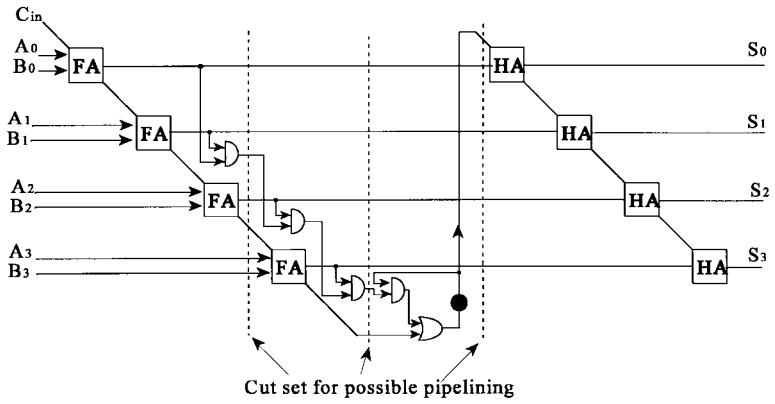


Fig. 7. 4-bit bit-level pipelined digit-serial adder.

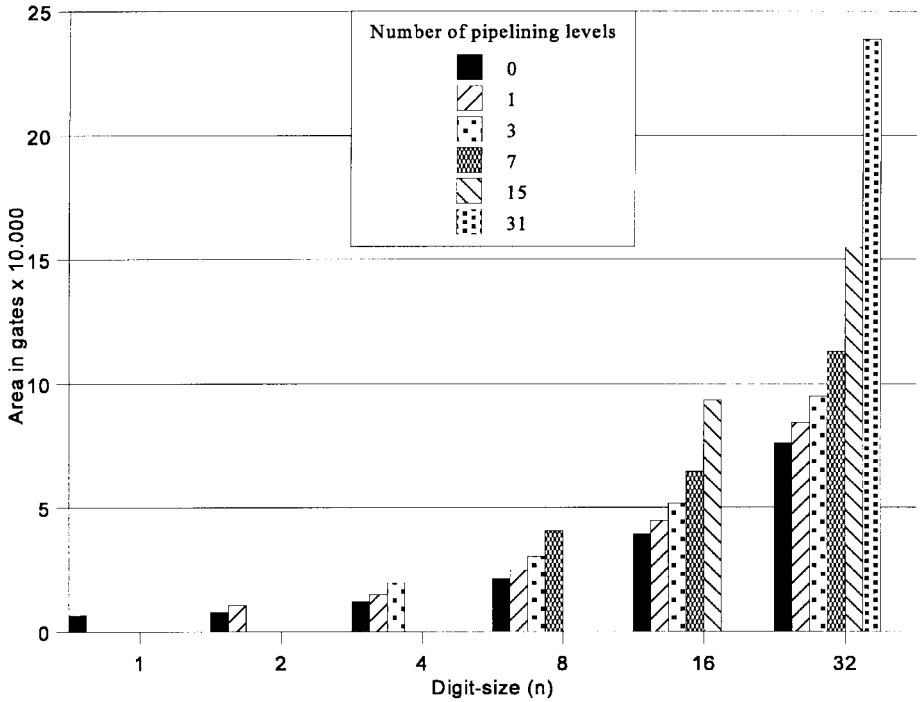


Fig. 8. Area of a vector inner product as a function of the digit-size and the degree of pipelining ($N = 32$ bits and $M = 5$).

2) The time (including setup and hold time) and area taken by a 1-bit latch are $T_{\mathrm{Latch}} = 6\Delta$ and $A_{\mathrm{Latch}} = 7A_T$.
3) The time and area taken by an AND gate are $T_{\mathrm{AND}} = 2\Delta$ and $A_{\mathrm{AND}} = 2A_T$.
4) The time and area taken by a 2-bit multiplexer are $T_{\mathrm{MUX}} = 2\Delta$ and $A_{\mathrm{MUX}} = 7A_T$.

Pipelining of the digit-serial VIP to the bit level will certainly increase the throughput rate, but at the same time increase the number of latches which are only used to temporarily hold the data. As a result, the number of pipelining levels is only limited by the area constraint. The time in terms of gate delays, the area in terms of gates, and area-time (AT) complexity of a 32-bit digit-serial VIP with length 5 as a function of the digit-size and the degree of pipelining, are shown in Figs. 8, 9 and 10, respectively. Figures 8 and 9 show that as the number of pipelining levels increases, the throughput rate as well as the area increase. For low pipelining levels, the percentage increase in hardware is less than the percentage increase in time, whereas for high pipelining levels, the percentage increase in hardware is much more than the percentage decrease in time. This can be shown clearly in Fig. 10, where it is shown that for each digit size, there is a transitional number of pipelining levels which results in the best area-time tradeoff for that particular digit size. It is also shown that for each number of pipelining levels, there is an intermediate digit size value which produces the minimum area-time complexity. However, the most significant result shown by these figures is that sub-digit pipelined digit-serial VIP can achieve higher throughput rates than the bit-parallel VIP with less silicon area consumption. As an example from Figs. 8 and 9, it can be seen that bit-level pipelined digit-serial VIP with digit size 4 is twice as fast as the equivalent bit-parallel VIP and uses a quarter of the hardware cost required by the bit-parallel VIP. This is very significant because the digit-serial structure is no longer seen as a compromise between bit-serial and bit-parallel structures, but can perform better than the bit-parallel in high-speed applications such as video signal processing and radar.

Fig. 11 shows the percentage of improvement in area-time obtained with the 32-bit digit-serial VIP based on the proposed radix-2^n algorithm when compared to the one derived from the generalization

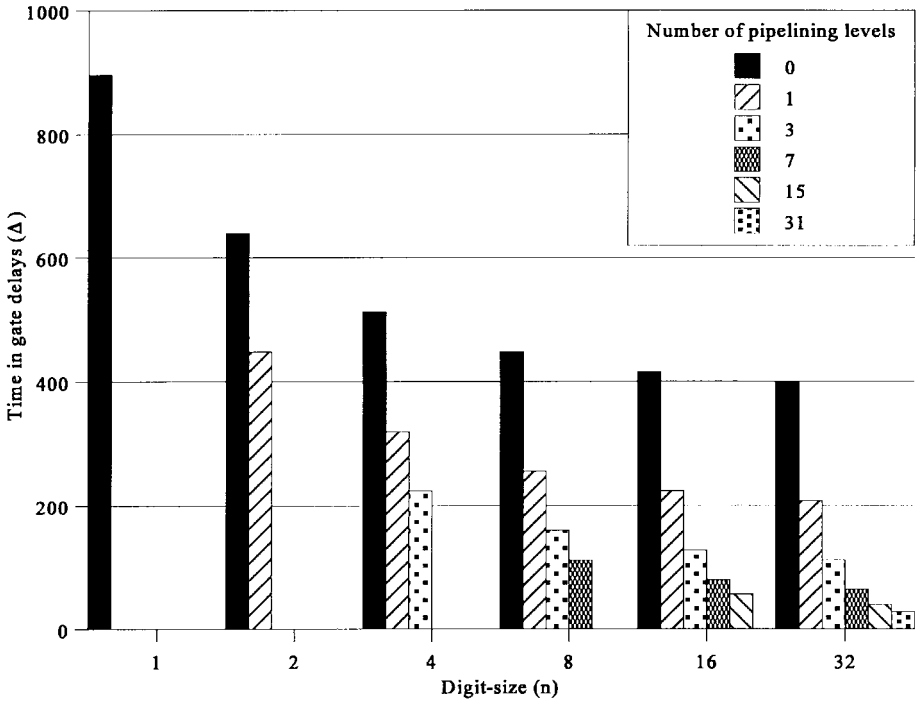


Fig. 9. Time of a vector inner product as a function of the digit-size and the degree of pipelining ($N = 32$ bits and $M = 5$).

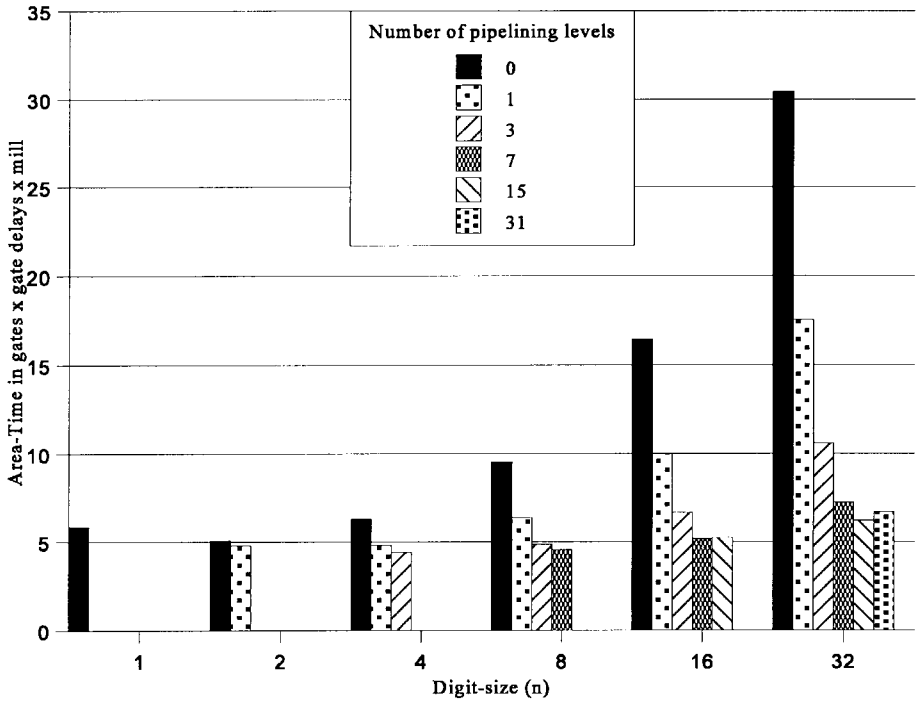


Fig. 10. Area-time of a vector inner product as a function of the digit-size and the degree of pipelining ($N = 32$ bits and $M = 5$).

of the binary arithmetic. The comparison is performed for optimum pipelining. The optimum pipelining is defined by the number of pipelining levels which produce the lowest area-time complexity for each digit size. From Fig. 11, it can be seen that the proposed digit-serial VIP has a far better performance than the one based on the previously reported radix-2^n algorithm [7], [8]. It shows that for $n = 8$, an improvement of 31% in area-time complexity is obtained. Without pipelining, both digit-serial VIP structures have similar performances, although the proposed digit-serial VIP reduces the number of latches by n per basic cell. This implies that the advantage of the proposed digit-serial VIP is that it can be pipelined more efficiently than the one based on the existing radix-2^n algorithm.

V. Twin-Pipe Digit-Serial Architectures

A digit-serial multiplier requires $2K$ clock cycles to produce the $2K$-digit product of a K-digit multiplicand (data word) and a K-digit multiplier (coefficient). In this section, a twin-pipe architecture

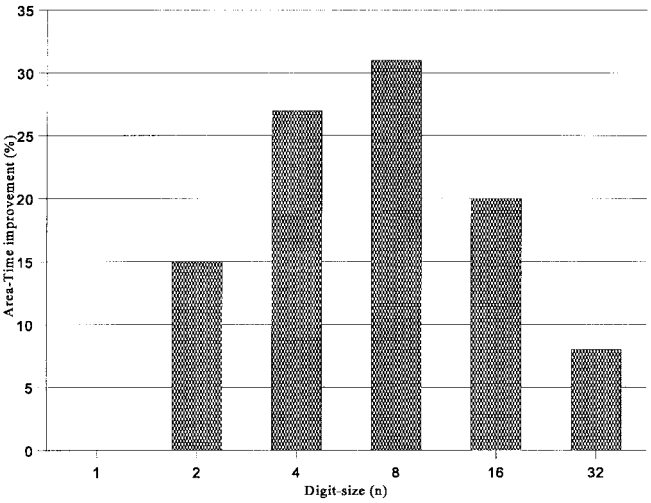


Fig. 11. Comparison of the area-time of the vector inner product based on the proposed radix-2^n algorithms and the existing radix-2^n algorithms as a function of the digit-size for optimum pipelining.

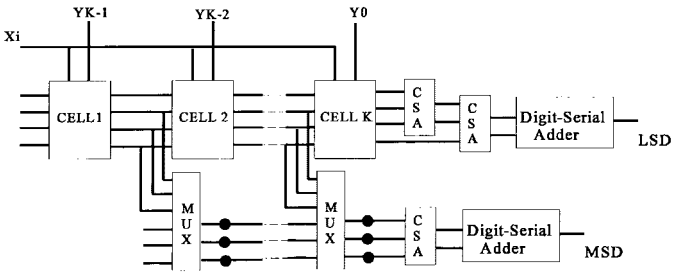


Fig. 12. Twin-pipe digit-serial multiplier.

which reduces the multiplication time to K cycles without truncating the results is presented. To be able to reduce the multiplication time to K cycles without truncation of the results, a second pipe is required for the computation of the K most significant digits (MSD) as shown in Fig. 12. After K cycles, the K least significant digits (LSD) of the result appear in the top accumulator. During the next K cycles, the K MSD of the result are shifted out using registers in the lower pipe, while the array multiplier computes the partial product of the next data whose the K LSD are computed in the top accumulator. A twin-pipe digit-serial VIP can be obtained by accumulating the K LSD and the K MSD of each digit-serial multiplier separately.

The twin-pipe architecture doubles the throughput rate of the digit-serial VIP's at the expense of some extra hardware. Fig. 13 shows the area-time complexity of the single and twin pipe digit-serial VIP. This figure shows the area-time of the twin pipe architecture is much lower than the single pipe. The reason is that the twin pipe doubles the throughput rate; however, the increase in hardware due to the second pipe is considered insignificant with respect to the total area and the improvement in the throughput rate. The improvement in area-time complexity varies between 11% for $n = 2$ to 42% for $n = 16$.

VI. Two's Complement Digit-Serial Multiplication

So far, it has been assumed that the vector elements U_m and V_m to be multiplied are both unsigned numbers. Most real applications require the processing of negative numbers. In this section, a two's complement digit-serial multiplier which can operate on both negative and positive numbers is presented.

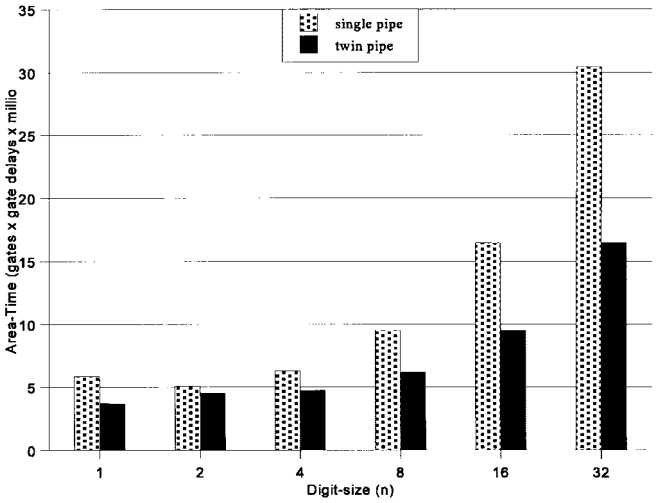


Fig. 13. Area-time of the single and twin pipe digit-serial vector inner product as a function of the digit-size.

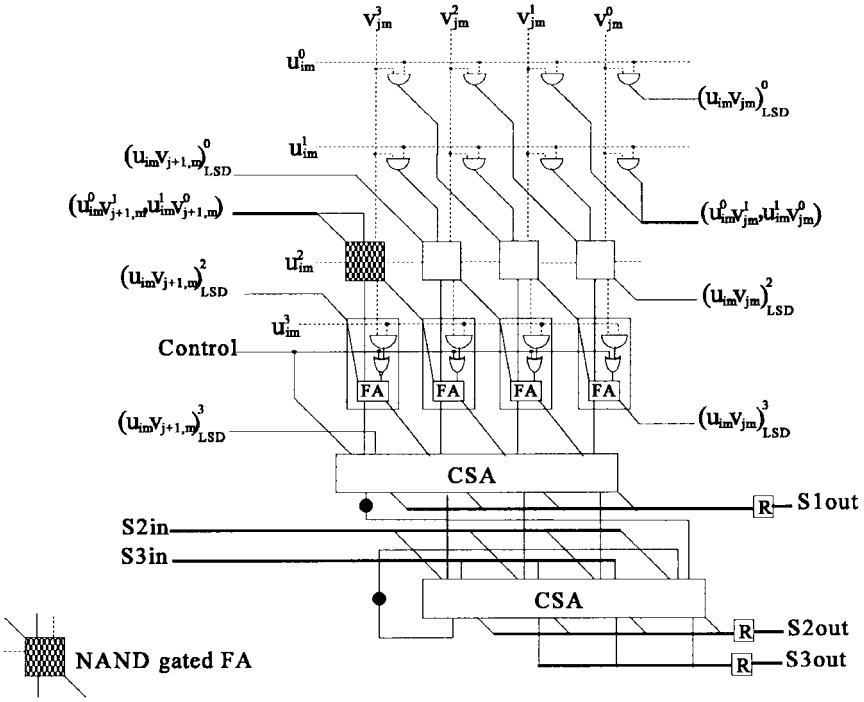
The conventional design of the two's complement digit-serial multipliers requires the extension of the sign bit. The proposed design is based on the algorithm for direct two's complement array multiplication proposed by Baugh and Wooley in 1983 [15]. The principle advantage of this algorithm is that the signs of all the summands are positive, thus allowing the array to be constructed entirely with conventional full adders. The two's complement digit-serial multiplier cells are shown in Fig. 14. To show how two's complement digit-serial multiplication is performed, we will first consider the multiplication of two N-bit two's complement numbers which may be written in the form

$$\begin{aligned} U_m &= -u_m^{(N-1)}2^{N-1} + \tilde{u}_m \\ V_m &= -v_m^{(N-1)}2^{N-1} + \tilde{v}_m \end{aligned} \tag{6}$$

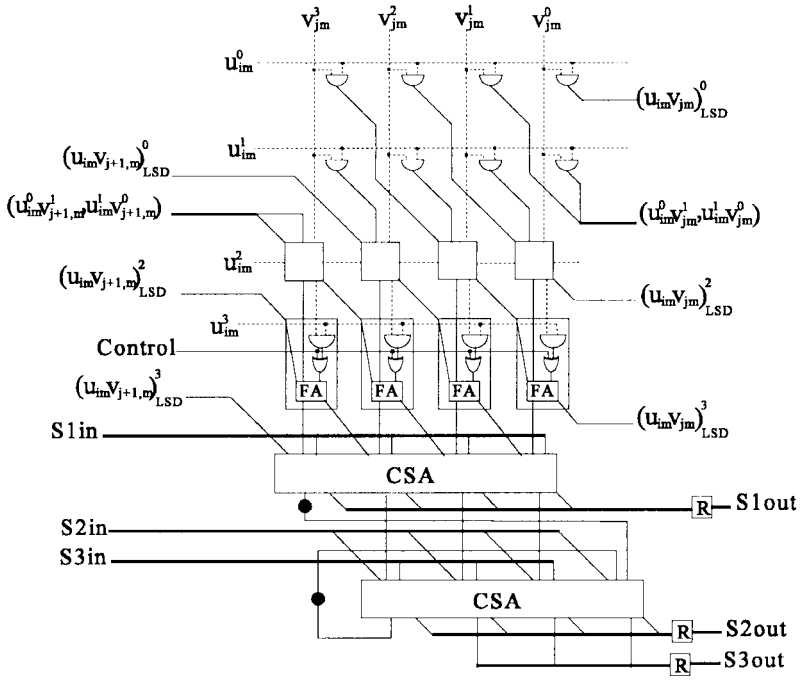
where $\tilde{u}_m$ and $\tilde{v}_m$ are $N-1$ bit positive numbers comprising the $N-1$ least significant bits of U_m and V_m, and N is the wordlength. Following the same procedure as in [16], the product of the two numbers, U_m and V_m, is given by

$$\begin{aligned} U_m \cdot V_m = \left(u_m^{(N-1)} v_m^{(N-1)}\right)2^{2N-2} + \tilde{u}_m \tilde{v}_m + \Big(\overline{\tilde{u}_m v_m^{(N-1)}} \\ + \overline{\tilde{v}_m u_m^{(N-1)}}\Big)2^{N-1} + 2^N - 2^{2N-1} \end{aligned} \tag{7}$$

To adapt (7) for digit-serial operation, the two's complement format is accounted for in two ways. First, in the first radix cell on the left of the digit-serial multiplier, the partial products involving the sign bit v_m^{N-1} are inverted using a NAND gate rather than an AND gate (see Fig. 14(a)). Second, at the Kth cycle, where K is the number of digits, the final row of each radix cell must subtract rather than add the final partial product (which involve the sign bit u_m^{N-1}) using two's complement addition. During the previous cycles, the last row of each radix cell performs an addition. This is achieved by using a control bit to define the Kth cycle and EXOR gates for the inversion as shown in Fig. 14. The answer is then obtained by adding a fixed correction term (given by the last two terms in (8)) to the final result. The addition of the term -2^{2N-1} is taken care of by adding a one to the most significant bit of the result and the term 2^N by adding a one to the $(N+1)$th column of the final result [16]. This is achieved by feeding the control bit into the most significant bit position of the first CSA of the first cell for the -2^{2N-1} term and delaying the control



(a)



(b)

Fig. 14. Two's complement cell architectures using array multiplier. (a) First cell. (b) Intermediate cell.

bit by one cycle and feed it into the least significant bit position of digit-serial adder for the 2^N term (see Fig. 14).

VII. Conclusion

A whole new set of radix-2^n algorithms are proposed. A design methodology of digit-serial VIP is also presented. The design of digit-serial VIP is based on the proposed radix-2^n algorithms. It was shown that the proposed digit-serial VIP outperforms the one based existing radix-2^n algorithms. Also, a new cell architecture is proposed for high-performance digit-serial vector inner product. It allows pipelining to the bit-level for high throughput rate with minimum initial delay and minimum area. The initial delay is made dependent only on the number of the pipelining levels. The number of pipelining levels can be varied to find the best compromise between the throughput rate and the hardware cost.

A twin-pipe architecture is presented which doubles the throughput rate of the digit-serial VIP. The increase in hardware due to the second pipe is considered insignificant compared to the total hardware cost and the improvement in the throughput rate. Also, a two's complement digit-serial multiplier which can operate on both positive and negative and numbers is presented.

REFERENCES

[1] P. B. Denyer and D. Renshaw, *VLSI Signal Processor—A Bit-Serial Approach*. Reading, MA: Addison-Wesley, 1985.

[2] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice–Hall, 1988.

[3] S. G. Smith *et al.*, "Techniques to increase the computational throughput of bit-serial architectures," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, ICASSP, Apr. 1987, pp. 543–546.

[4] K. K. Parhi, "A systematic approach for design of digit-serial signal processing architectures," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 358–375, Apr. 1991.

[5] R. Hatley and P. Corbett, "Digit-serial processing techniques," *IEEE Trans. Circuits and Systems*, vol. 37, pp. 707–719, June 1990.

[6] M. D. Ercegovac, "On-line arithmetic: An overview," *SPIE*, vol. 495, pp. 86–93, 1984.

[7] A. Aggoun, M. K. Ibrahim, and A. Ashur, "A novel cell architecture for high performance digit-serial computation," *Electron. Lett.*, vol. 29, no. 11, May 1993.

[8] M. K. Ibrahim, "Radix-2^n multiplier structures: A structured design methodology," *Inst. Elect. Eng. Proc.*, July 1993, vol. 140, Pt. E, no. 4, pp. 185–190.

[9] A. E. Bashagha and M. K. Ibrahim, "Radix digit-serial pipelined divider/square-root architecture," *Inst. Elec. Eng. Proc. Comput. Digit. Tech.*, Nov. 1994, Vol. 141, no. 6, pp. 375–380.

[10] J. V. McCanny *et al.*, "The use of data dependence graphs in the design of bit-level systolic arrays," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. 38, no. 5, pp. 787–793, May 1990.

[11] G. Privat, "A novel class of serial-parallel redundant signed-digit multipliers," in *IEEE Int. Symposium on Circuit and Systems*, ISCAS, 1990, pp. 2116–2119.

[12] M. C. Chen, "The generation of a class of multipliers: Synthesizing highly parallel algorithms in VLSI," *IEEE Trans. Comp.*, vol. 37, no. 3, Mar. 1988.

[13] C. W. Wu and P. R. Cappello, "Block multipliers unify bit-level cellular multiplications," *Int. J. Comp.-Aided VLSI Design*, pp. 113–124, 1989.

[14] K. Hwang, *Computer Arithmetic Principle, Architecture, and Design*, New York: Wiley, 1979.

[15] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Comp.*, vol. c–33, pp. 1045–1047, 1983.

[16] J. V. McCanny *et al.*, "Optimized bit level systolic array for convolution," *Inst. Elec. Eng. Proc.*, Vol. 131, pt. F, no. 6, pp. 632–637, Oct. 1984.

Pipelined DFE Architectures Using Delayed Coefficient Adaptation

R. Perry, David R. Bull, and A. Nix

***Abstract*— In this paper the delayed least-mean-square (DLMS) algorithm is proposed for training a transversal filter-based decision feedback equalizer (DFE). Delays in the filter coefficient update process are used to pipeline the DFE, thereby increasing the throughput rate, for a given speed of hardware. The filter structures selected for the feedforward and feedback section of the DFE facilitate the use of a shared error signal, thereby reducing communication costs. The new resulting structure is highly modular and is very suitable for very large scale integration (VLSI) implementation. A pipelined form for the normalized least-mean-square algorithm (NMLS) is also obtained which removes the dependency of the convergence speed on the input signal power. The convergence and residual mean-square-error characteristics of the different pipelined filters are compared.**

***Index Terms*— Adaptive filters, delayed least-mean-squares, equalization, pipelined.**

I. INTRODUCTION

Although numerous adaptive equalizers have been reported in the literature, decision feedback equalization (DFE) remains a popular choice of equalization technique, especially for high-data-rate (e.g., HIPERLAN) applications, where alternatives, such as Viterbi equalizers, are precluded due to their complexity [1]. The use of long training sequences enables the selection of relatively simple gradient-based adaptive algorithms for equalizer training. Despite this concession, the realization of a high-sampling-rate DFE still remains challenging. This is due to the inherent sampling rate limitation of any adaptive filter associated with the generation and feedback of the error signal required for the filter coefficient updates. Exploitation of the natural parallelism in the least-mean-square (LMS) algorithm can help to reduce the algorithm iteration period. However, the regularity of the filtering structure is limited [2] and global broadcasting of the error signal to all coefficient update modules is still required. This has motivated the use of approximations to the LMS algorithm which sacrifice performance to increase the level of pipelining in the adaptation feedback loop. The delayed LMS (DLMS) algorithm is an example of an approximate form of the LMS algorithm, where the coefficient updates are delayed by an arbitrary number of sample periods [3].

In this brief a DFE architecture is derived, comprising a cascade of identical processing modules (PM's). The *order recursive filter* (ORF), described in [4], is used for the feedforward section but cannot be used for the feedback filter (FBF) because of the latency in the output. A transposed direct-form transversal filter (TF) together with a modified coefficient update is used to realize a pipelined FBF. By exploiting shared input parameters between the feedforward filter (FFF) and FBF stages, a new modular DFE structure is obtained (Section II-A), requiring minimal global communication.

Despite the attraction of its relative simplicity, a particular problem of implementing the LMS algorithm is the dependency of the algo-

Manuscript received August 1, 1996; revised April 1, 1997. This work was supported by Hewlett Packard Laboratories, Bristol, and by the U.K. Engineering and Physical Sciences Research Council (EPSRC). This paper was recommended by Associate Editor K. K. Parhi.

The authors are with the Centre for Communications Research, University of Bristol, Bristol BS8 1TR, U.K. (e-mail: Dave.Bull@bristol.ac.uk).

Publisher Item Identifier S 1057-7130(98)05061-7.