Design Strategies for the Final Adder in a Parallel Multiplier

Paul F. Stelling*
Dept. of Computer Science
University of California at Davis
Davis, CA 95616
Email: stelling@cs.ucdavis.edu

Vojin Oklobdzija
Dept. of Electrical and Computer Engineering
University of California at Davis
Davis, CA 95616
Email: vojin@ece.ucdavis.edu

Abstract

In this paper we address the problem of adding two n-bit numbers when the bit arrival times are arbitrary (but known in advance). In particular we address a simplified version of the problem where the input arrival times for the i^{th} significant bits of both addends are the same, and the arrival times t_i have a profile of the form:
$t_0 \leq t_1 \leq \ldots < t_k = t_{k+1} = \ldots = t_p > t_{p+1} \geq \ldots \geq t_{n-1}$
This profile is important because it matches the signal arrival time profile of the reduced partial products in a parallel multiplier before they are summed in the final adder.

In this paper we present a design strategy specific to arrival time profiles generated by partial product reduction trees constructed by optimal application of the Three Dimensional Method presented by Oklobdzija, Villeger, and Lui and subsequently analyzed by Martel, Oklobdzija, Ravi, and Stelling. This strategy can be used to obtain adders for any arrival time profile that matches the above form, as well as a broad class of arrival time profiles where even greater variation in the input times is allowed.

Finally, we show that our designs significantly outperform the standard adder designs for the uniform signal arrival profile, yielding faster adders that (for these profiles) are also simpler and use fewer gates.

Keywords: *Parallel Multiplier, Final Adder, Algorithms, VLSI circuits.*

1 Introduction

The problem of constructing fast and efficient adders when all input bits arrive at the same time is one that has been well studied [3]. A related problem that is also important in the construction of high performace machines is the design and implementation of efficient adder circuits when the bit arrival times are arbitrary (but known in advance). One situation of this type is the final adder for a parallel multiplier. The design of the final adder for a parallel multiplier is important because any improvements in final adder performance directly impact multiplication time, and multiplication is a commonly used and expensive operation.

In [6], Oklobdzija, Villeger, and Liu suggested a new approach, the Three Dimensional Method (TDM), for Partial Product Reduction Tree (PPRT) design that produces PPRTs that outperform the current best designs. In the TDM the PPRT is designed by interconnecting (3,2)-adders (full adders) in a globally optimal way based on careful modelling of input-to-output delays. Specifically, delays are measured in equivalent XOR delays. If $a \leq b \leq d$ are the inputs to a (3,2)-adder then the sum output is generated at time $s = \max(b+2, d+1)$ and the carry at time $c = d+1$. The TDM approach was subsequently analyzed by Martel, Oklobdzija, Ravi, and Stelling in [10], and optimal TDM PPRT designs for reducing the partial products to two rows typified.

The usual practice has been to add the last two rows of partial products using as the Final Adder (FA) one of the fast schemes such as Carry Lookahead (CLA). This concept was first challenged by Oklobdzija in [7], where it was shown that under the non-equal signal arrival profile some of the commonly known fast schemes for addition do not perform well. For example, if the LSB arrives first and MSB last, with the signal delay increasing by $\frac{1}{2}$ equivalent XOR delay per bit, then the CLA adder will be slower than a Ripple Carry Adder (RCA). Given that the signal arrival profile to the final adder is more complex, the problem of constructing the FA is becomes more complicated. This is augmented by the fact that the optimal FA is really a "hybird" consisting of a number of blocks that can encompass several different types of adders. In this paper we will examine the design of optimal adders that may contain Ripple-Carry, Carry-Skip, and Carry-Select blocks. The extension of our ideas to Carry-Lookahead adders is straight-forward, but we will not address it here.

1.1 Adder goals

In [10] the PPRT circuits for m-by-m bit multiplication were evaluated based on the corresponding vectors $(t_0, t_1, \ldots, t_{2m-2})$ of the output times of the latest output signal for each column. (All inputs to the PPRT were assumed to be available at time 0, and t_i is the time at which the last output bit for column i was generated.) Following their useage, we say that a TDM PPRT circuit with output time vector $V = (v_0, v_1, \ldots, v_{2m-2})$ is **undominated** in its class if there is no other TDM PPRT circuit which takes the same inputs and generates an output time vector $U = (u_0, u_1, \ldots, u_{2m-2})$ such that $u_i \leq v_i \ \forall i \in \{0, 1, \ldots, (2m-2)\}$, and $V \neq U$. A "Latest-fewest" heuristic was introduced for evaluat-

*Research supported by NSF grants CCR-94-03651 and CCR-91-03937.

1058-6393/96 $5.00 © 1996 IEEE
Proceedings of ASILOMAR-29

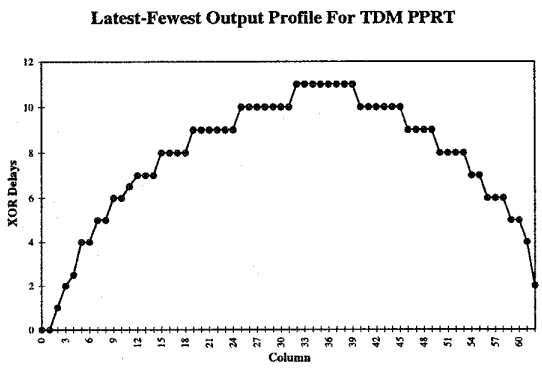

Figure 1: *The output profile of the Latest-Fewest TDM PPRT. There is only one output bit for position 0, and two output bits for positions 1 through 62.*

ing the output vectors of the undominated circuits in a class. By this heuristic V comes before U if the largest value which does not appear in exactly the same positions in V and U either first appears later in U, or first appears in the same position of both V and U, but last appears later in U. The optimal vectors by this heuristic both have the minimum maximum delay value and follow the well known profile pattern for other PPRT designs whereby the signals for the least and most significant bits are generated earliest, with the middle signals appearing later. In fact, they were of the following, stricter, pattern: $t_0 \leq t_1 \leq \ldots < t_k = t_{k+1} = \ldots = t_p > t_{p+1} \geq \ldots \geq t_{2m-2}$.

In this paper will will address the design of efficient adders for such a profile. By efficient we mean that the adder design should be fast, of small area, and use low power. When alternative designs have the same speed, we prefer the simpler design (with fewer gates).

2 Adders for TDM PPRT Profiles

In [6, 10], TDM PPRT designs were analyzed based on the time delay of an XOR gate, with the delay of NAND and NOR gates being approximately 0.5 XOR delays. In this paper we will also use this convention, although our results do not rely on it. By this convention, a carry ripples through a full adder ((3,2) adder) with 1 XOR delay (1 NAND delay plus 1 NOR delay). Based on this convention, the latest-fewest TDM PPRT output profile (and hence final adder input profile) for 32-bit multiplication is given in Figure 1.

As can be seen from the figure, starting with the output bits for column 1, the delays first increase with the column numbers and then decrease. Also, after column 4, the number of columns with integer delay i is increasing with i. As the multiplication size m increases, the changes to the delay profile consist generally of the insertion of columns of higher delay near the middle of the profile. In the following sections we will first show how to construct an adder for profiles of this form, and then we will generalize the approach for a more general (less restricted) class of profiles.

All n-bit adder schemes can be characterized as being made up of blocks that each take a carry-in bit (except possibly the first block), add some portion of the n bits, and generate a carry-out value that is propagated in some fashion to appropriate blocks with more significant bits. These blocks are typically (but need not be) subdivided into smaller blocks, and so on, until some smallest sub-block (generally a full adder, sometimes called a (3,2)-adder). We will only consider designs where the smallest block is a (3,2)-adder, but will allow the basic blocks (and even sub-blocks within a block) to be of various types. I.e., some blocks may be Ripple-carry blocks, some Carry-skip blocks, and others Carry-lookahead or Carry-Select blocks. By using a **Hybrid adder** in this fashion we will achieve significantly faster addition while maintaining overall simplicity and regularity within blocks required for compact and power-efficient circuits.

Before describing our approach in detail, we make the following definitions, which we will use extensively. These definitions are generalizations of definitions that have been used elsewhere [5] in relation to adders where all input signals arrive at the same time.

Definition: Given an input profile $T = (t_0, \ldots, t_n)$, and blocks $B_0, \ldots, B_k$ of sizes $b_0, \ldots, b_k$ respectively, we define:

- $I(B_i)$ is the internal-carry delay of block B_i, the maximum delay associated with generating a carry within B_i and propagating it within the same block;
- $G(B_i, B_j)$ is the carry-generate delay of block B_i and B_j, the maximum delay associated with generating a carry within B_i and propagating it to block B_j;
- $P(B_i, B_j)$ is the carry-propagate delay for blocks B_i and B_j $(j > i)$, the maximum delay associated with propagating a carry that arrives at B_i from any previous block to B_j;
- $A(B_i)$ is the carry-assimilate delay of block B_i, the maximum delay associated with propagating a carry within B_i when that carry arrives at B_i from any previous block; and
- $L_A(B_i)$ is the latest delay possible that a carry generated in any block before B_i can arrive at B_i.

Note that these definitions all pertain to the time at which the carry-in signals become available for the relevant column or block. The sum signal for each column is then generated between 1 and 2 XOR delays later. Our goal in designing a Hybrid Adder is to minimize the maximum of these delays over all blocks, subject to any power, space, and complexity constraints we wish to impose. Depending on the block structures used, the various delays above may or may not be relevant to a given design. For example, $P(B_i, B_j)$ may not be an appropriate measure for two blocks in a Carry-Lookahead Adder, since for three blocks B_h, B_i, and B_j $(h < i < j)$, B_i is not on the critical path for propagating carries from B_h to B_j. In our description of the Hybrid Adder design we must be careful to identify which delays are relevant to each block or combination

of blocks. Note that the definitions above make no assumptions about the structure of the blocks, but that $L_A(B_i)$ depends on $G(B_h, B_i)$ for all blocks B_h, $h < i$, and could also depend on $P(B_k, B_i)$ for some blocks B_k, $k < i$.

We show how we apply these definitions for some standard (and well-known) types of adders. First we apply it to a Ripple-Carry Adder block B_i containing bits $t_r, \ldots, t_s$. Using 1 XOR delay as our time unit, we then have that for each block (column) B_i:

- $I(B_i) = \max_{r \le j \le s-1}(t_j + s - j)$;
- $G(B_i, B_j)$ is relevant only for $j = i + 1$, and $G(B_i, B_{i+1}) = \max_{r \le j \le s}(t_j + s - j + 1)$;
- $P(B_i, B_j)$ is relevant only for $j = i + 1$, and $P(B_i, B_{i+1}) = \max(L_A(B_i) + s - r + 1, \max_{r \le j \le s}(t_j + s - j + 1))$;
- $A(B_i) = \max(L_A(B_i) + s - r, \max_{r \le j \le s}(t_j + s - j))$;
- $L_A(B_0) = 0$ (assuming no carry in to the adder, otherwise the latest time at which it can arrive), and
- $L_A(B_i) = \max(G(B_{i-1}, B_i), P(B_{i-1}, B_i))$ for $i \neq 0$.

Observation 2.1 If $I(B_i)$, $G(B_i, B_{i+1})$, $P(B_i, B_{i+1})$, and $A(B_i)$ are all $\le t_{s+1}$, then a Ripple Carry Adder will suffice for the block, because those values will all be less than the corresponding values for block B_{i+1}. I.e., if $L_A(B_i)$, $t_r, t_{r+1}, \ldots, t_s$ are all on or below the line $t = c + (t_{(s+1)} - (s + 1))$ (the line of slope 1 that passes through the point $((s + 1), t_{(s+1)})$), then all of the delays associated with B_i will be $\le t_{(s+1)}$.

We now denote by $t_r, \ldots, t_s$ the delays of the input bits in B_i. Also, we assume that for Carry-Skip skips can always be completed (for any size block) by time $1 + \max(L_A(B_i), \max_{r \le j \le s}(t_j))$. (This assumption is reasonable for small blocks and blocks where $L_A(B_i) >> \max_{r \le j \le s}(t_j)$). Then for Carry-Skip blocks we have that:

- $I(B_i) = \max_{r \le j \le s}(t_j + s - j)$;
- $G(B_i, B_j)$ is relevant only for $j = i + 1$, and $G(B_i, B_{i+1}) = \max_{r \le j \le s}(t_j + s - j + 1) + 0.75$ (the 0.75 is for an OR or NOT/NOR to combine generated and skipped carries between blocks);
- $P(B_i, B_j)$ is relevant only for $j = i + 1$, and $P(B_i, B_{i+1}) = L_A(B_i) + 1$ (since blocks are of size at least 1);
- $A(B_i) = \max(L_A(B_i) + s - r, \max_{r \le j \le s}(t_j + s - j))$;
- $L_A(B_0) = 0$ (assuming no carry in to the adder, otherwise the latest time at which it can arrive); and
- $L_A(B_i) = \max(G(B_{i-1}, B_i), P(B_{i-1}, B_i))$ for $i \neq 0$.

Observation 2.2 For a Carry-Skip Block B_i over columns $r, \ldots, s$ as defined above, a necessary condition to minimize the maximum of $I(B_i)$, $G(B_i, B_{i+1})$, $P(B_i, B_{i+1})$, and $A(B_i)$ is that b_i (given r and the value of $L_A(B_i)$) be such that:

- $P(B_i, B_{i+1}) = L_A(B_i) + 1$, and
- $G(B_i, B_{i+1}) < P(B_i, B_{i+1}) + 1 = L_A(B_i) + 2$.

Otherwise either B_i could be a Ripple Carry block or B_i could be split into two blocks $B_{i'}$ and B_i'' that combined have simpler structure than B_i (since the skip trees are smaller) and achieve delays:

- $G(B_{i'}, B_{i''}) < G(B_i, B_{i+1})$ and $G(B_{i''}, B_{i+1}) < G(B_i, B_{i+1})$ (because $B_{i'}$ and $B_{i''}$ include (disjoint) subranges of B_i;
- $P(B_{i'}, B_{i''}) = P(B_i, B_{i+1})$; and
- $P(B_{i''}, B_{i+1}) = P(B_i, B_{i+1}) + 1 < G(B_i, B_{i+1})$ (by assumption).

(Note that one of $B_{i'}$ and B_i'' could be a Ripple Carry block).

Observation 2.3 For a Hybrid Adder with Ripple-Carry and Carry-Skip blocks as defined above, a sufficient condition for the maximum of $I(B_i)$, $G(B_i, B_{i+1})$, $P(B_i, B_{i+1})$, and $A(B_i)$ over all blocks to be less than some limit d is that the size b_i of any Carry-Skip block B_i be such that $G(B_i, B_{i+1}) > P(B_i, B_{i+1}) - 1 = L_A(B_i)$ unless it would require that $I(B_i) > d$ or $A(B_i) > d$, in which case b_i is set as large as possible subject to those constraints.

Figure 2 shows an optimal Hybrid Adder design for using 1 level of Carry-Skip blocks and a single Carry Select block That adds two 62-bit numbers whose signals are all available at time 0 in total time equivalent to 12.5 XOR delays. Figure 3 shows the output delays of the same adder when applied to the profile of Figure 1, giving a maximum output delay of 23.5 XOR delays. Thus when appled to the profile the latest output is later by the delay of the latest input signal, even though many of the signals arrive significantly earlier.

Now we use the formulas above to construct a Hybrid Adder for the profile in Figure 1 using Ripple Carry and Carry Skip blocks. First we note that based on the input signal profile vector, Observation 2.1 applies to columns $1, \ldots, 5$ and nowhere else (column 0 has only one bit as input). Thus, we can set B_0 as a Ripple-Carry block on columns $1, \ldots, 5$. Given that $L_A(B_0) = 0$, we then have that $I(B_0) = 5$, $G(B_0, B_1) = 5$, $P(B_0, B_1)$ is not applicable, $A(B_0)$ is not applicable, and $L_A(B_1) = G(B_0, B_1) = 5$.

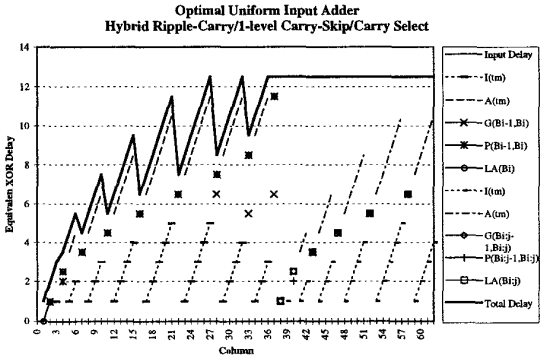

Figure 2: *The delays of the optimal 62-bit Hybrid Adder for uniform inputs using 1-level Carry-Skip and one Carry-Select block.*

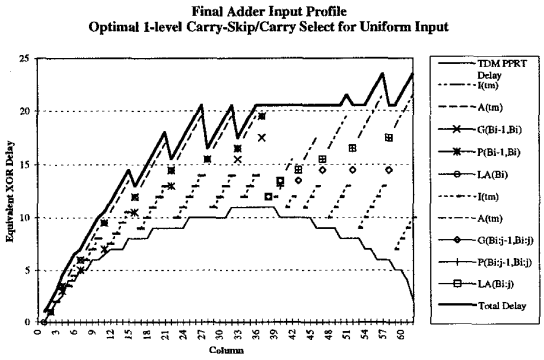

Figure 3: *The delays of the optimal 62-bit Hybrid Adder for uniform inputs using 1-level Carry-Skip and one Carry-Select block when applied to the 32-bit Latest-Fewest TDM PPRT output profile.*

Since Observation 2.1 does not apply past column 5, we will use Carry-Skip blocks for the later columns. Our approach is to first construct a preliminary design using block sizes based exclusively on the values $G(B_i, B_{i+1})$ and $P(B_i, B_{i+1})$. The maximum of those values will give us a lower bound on the delay d of the latest output signal of the optimal Hybrid Adder that uses Ripple-Carry and Carry-Skip blocks. An upper bound will be provided by the maximum $I(B_i)$ and $A(B_i)$ values for that same design. We can then use binary search on the possible values of d to find the smallest achievable value.

For B_1, we have already seen that $L_A(B_1) = 5$. Using the formulas above, we have that the minimum possible value of $P(B_1, B_2)$ is $L_A(B_1) + 1 = 6$. By Observations 2.2 and 2.3 we use block size b_1 such that $L_A(B_1) < G(B_i, B_{i+1}) < L_A(B_1) + 2 = 7$, i.e., $b_1 = 2$. By repeated application of the principles in the observations we get the preliminary design and delays depicted in Figure 4. By using binary search on the values between the lower bound of 19.5 ($P(B_{12}, B_{13})+1$) and the upper bound of 25.5 ($A(B_{12})$ we obtain the design in Figure 5, with latest output at 23.5 XOR delays. Thus we have matched the performance of the adder for uniform inputs that used a Carry-Select block with a much simpler adder.

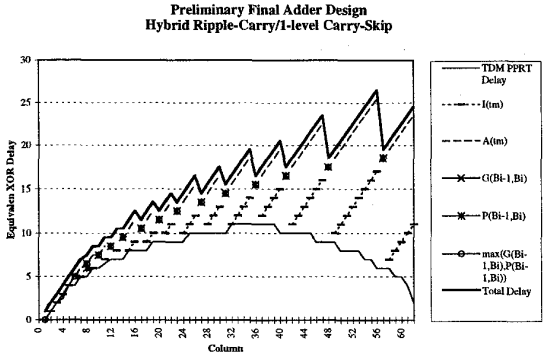

Figure 4: *The delays of the preliminary 62-bit Hybrid Adder using Ripple-Carry and 1-level Carry-Skip blocks for the 32-bit Latest-Fewest TDM PPRT output profile.*

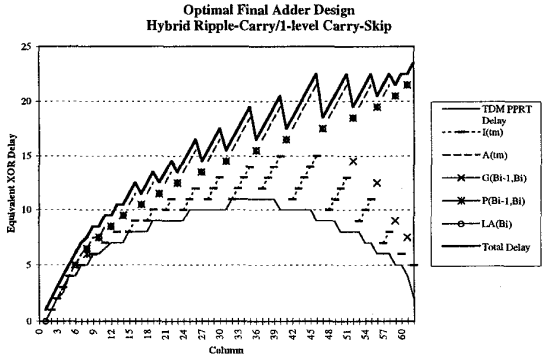

Figure 5: *The delays of the optimal 62-bit Hybrid Adder using Ripple-Carry and 1-level Carry-Skip blocks for the 32-bit Latest-Fewest TDM PPRT output profile.*

This design is optimal among all Hybrid Adders that use only Ripple-Carry and 1-level Carry-Skip blocks, but can be easily improved. One simple change would be to add a second skip block over columns 59 through 62. This would reduce the maximum output delay to 22.5. Similarly, the principles described here can be appled to multi-level Carry-Skip blocks, Carry-Lookahead blocks, and Carry Select blocks. It is this last possibility that we will now examine.

The terms defined above can be easily applied to Carry-Select blocks. For this analysis we increase the $A()$ and $G()$ values of the previous block by 2 to allow for the increased delay caused by the large fan-out of the last NOR gate[11]. The delays within the Carry Select block are based on the underlying Carry-Skip sub-blocks.

The design approach is similar to the one used previously. We know that the Carry-Select block will be-

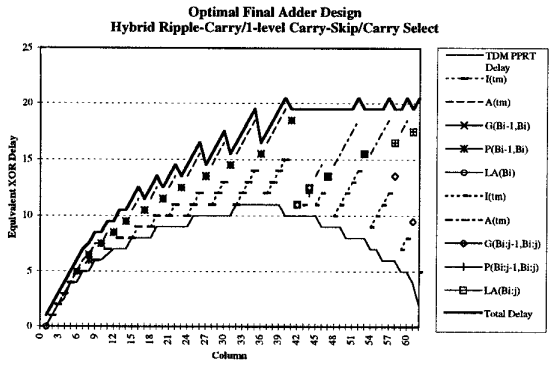

Figure 6: *The delays of the optimal 62-bit Hybrid Adder using Ripple-Carry, 1-level Carry-Skip, and one Carry Select blocks for the 32-bit Latest-Fewest TDM PPRT output profile.*

gin immediately following one of the Carry-Skip blocks, and that any optimal Hybrid Adder that uses a Carry-Select block must perform at least as well as the optimal Ripple-Carry/1-level Select solution. Thus we try to achive progressively smaller delay bounds beginning with the optimal result from the design of Figure 5. As thebound is decreased some of the Carry-Skip blocks may need to be changed to remain in compliance with it, bu the Carry-Select block will always begin immediately following an exisiting block, so only a limited search is necessary. The optimal design is shown in Figure 6, and achieves maximum delay of 20.5 equivalent XORs. Thus we have achieved better than 12.5% improvement over the original design for uniform signal profile.

3 Conclusions

We have shown that fast adder designs based on uniform signal delay profiles can give poor results when used as the final adder in a parallel multiplier. The optimal final adder is instead a complex hybrid structure containing blocks that may consist of a variety of different adder designs. We have given a generalized model for evaluating the delays associated with each block of such a Hybrid Adder and applied it to signal profiles corresponding to the final adder inputs of a parallel multiplier using optimal TDM PPRT circuits. We have shown how to design an optimal Hybrid Adder made up of blocks of Ripple-Carry, Carry-Skip (1-level), and Carry-Select Adders using an approach that easily extends to blocks made up of other adders, such as multi-level Carry-Skip, Carry-Lookahead, and Conditional Sum. Our optimization method has produced an optimal structure for the case of a 32X32-bit multiplier. The improvement in speed is estimated to be 12.5% over commonly used CLA scheme. This contributes roughly to 4% of the total speed given that the final adder uses about one third of the time used for multiplication in a parallel multiplier.

We are presently analyzing other adder designs that are commonly used in Final Adders, namely Carry-Lookahead and Conditional Sum Adders. Our preliminary results show that for the final adder problem the standard designs yield latest output bits of delay approximately equal to the sum of the delay of the latest input bit and the delay of the adder on a uniform input profile. With our approach we have determined that we can build faster adders using simpler circuitry, but we have not yet completed optimal designs.

References

[1] Vojin G. Oklobdzija and Earl R. Barnes, "Some Optimal Schemes for ALU Implementation in VLSI Technology", *Proceedings of the 7th Symposium on Computer Arithmetic*, (1985).

[2] Vojin G. Oklobdzija and Earl R. Barnes, "On Implementing Addition in VLSI Technology", *Journal of Parallel and Distributed Computing*, 5, pp. 716-728 (1988).

[3] Earl E. Swartzlander, ed., *Computer Arithmetic Vol. 1 & 2*, IEEE Computer Society Press, (1990).

[4] Brian D. Lee and Vojin G. Oklobdzija, "Improved CLA Scheme with Optimized Delay", *Journal of VLSI Signal Processing*, 3, pp. 265-274 (1991).

[5] Pak K. Chan, Martine D. F. Schlag, Clark D. Thomborson, and Vojin G. Oklobdzija, "Delay Optimization of Carry-Skip Adders and Block Carry-Lookahead Adders Using Multidimensional Dynamic Programming", *IEEE Transactions on Computers*, Vol. 41, No. 8, pp.920-930 (August 1992).

[6] Vojin G. Oklobdzija, David Villeger, Simon S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," in press, *IEEE Transaction on Computers*, (1995).

[7] Vojin G. Oklobdzija, "Design and Analysis of Fast Carry-Propagate Adder Under Non-Equal Input Signal Arrival Profile", *Proceedings of the 28th Asilomar Conference on Signals, Systems, and Computers*, (1994).

[8] Vojin G. Oklobdzija and David Villeger, "Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology", in press, *IEEE Transactions on VLSI*, (1995).

[9] V. G. Oklobdzija and David Villeger, "Optimization and Analysis of a Carry-propagate Adder Under the Non-uniform Signal Arrival Profile," in preparation.

[10] Charles Martel, Vojin Oklobdzija, R. Ravi, and Paul F. Stelling, "Design Strategies for Optimal Multiplier Circuits", *Proceedings of the 12th Symposium on Computer Arithmetic*, pp. 42-49 (1995).

[11] *1.0-Micron Array-Based Products Databook*, LSI Logic Corporation, (September 1991).