Journal of VLSI Signal Processing 14, 321–331 (1996)
© 1996 Kluwer Academic Publishers. Manufactured in The Netherlands.

Design Strategies for Optimal Hybrid Final Adders in a Parallel Multiplier

PAUL F. STELLING*
Department of Computer Science, University of California at Davis, Davis, CA 95616

VOJIN G. OKLOBDZIJA
Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA 95616

Received March 17, 1996; Revised July 18, 1996

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 ith significant bits of both addends are the same, and the arrival times t_i have a profile of the form:

$$t_0 \leq t_1 \leq \cdots < t_k = t_{k+1} = \cdots = t_p > t_{p+1} \geq \cdots \geq t_{n-1}$$

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 Liu 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 out-perform the standard adder designs for the uniform signal arrival profile, yielding faster adders that (for these profiles) are also simpler and use fewer gates.

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 [1]. 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 [2] Oklobdzija et al., 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 et al. in [3], and optimal TDM PPRT

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

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) or Conditional Sum (CSA). This concept was first challenged by Oklobdzija in [4], 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 becomes more complicated. This is augmented by the fact that the optimal FA is really a "hybrid" consisting of a number of blocks that can encompass several different types of adders. Such adders were introduced and analyzed by Stelling and Oklobdzija in [5], where they developed optimal adders using Ripple-Carry, Carry-Skip, and Carry-Select blocks under a simple timing model. Here we extend that model to achieve more realistic results. In this paper we will examine the design of optimal adders that may contain Ripple-Carry, Carry-Skip, and Carry-Select blocks.

1.1. *Adder Goals*

In [3] 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-Earliest" heuristic was introduced for evaluating 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 are of the following, stricter, pattern: $t_0 \leq t_1 \leq \cdots < t_k = t_{k+1} = \cdots = t_p > t_{p+1} \geq \cdots \geq t_{2m-2}$.

In this paper we 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 [2, 3], 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-Earliest TDM PPRT output profile (and hence final adder input profile) for 32-bit multiplication is given in Fig. 1.

As can be seen from the figure, starting with the output bits for column 1 (the PPRT generates a single bit for column 0), 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

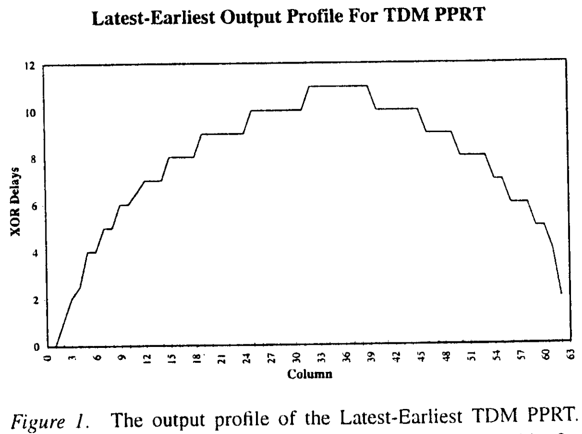


Figure 1. The output profile of the Latest-Earliest TDM PPRT. There is only one output bit for position 0, and two output bits for positions 1 through 62.

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 as input a carry-in bit (except possibly the first block) and some input bits corresponding to the n columns in the block, and generate sum bits for the block and a carry-out value that is propagated in some fashion to appropriate blocks with more significant bits. For such designs as Ripple-Carry, Carry-Skip, and Carry-Lookahead the column input bits are the initial inputs and/or signals generated from them (e.g., carry generate and carry propagate signals). Carry-select blocks take as input a carry-in bit, alternative sum bits, and alternative carry-out bits and generate sum bits and a carry-out bit. 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). In this paper we will only consider designs where the smallest block is a (3,2)-adder. (Our work in progress includes Conditional-Sum Adders which use alternative (faster) circuits for generating sum and carry values). We will, however, 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 the 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. Some of these definitions are generalizations of definitions that have been used elsewhere [6] in relation to adders where all input signals arrive at the same time.

Definition. Given an input profile $T = (t_0, \dots, t_n)$, and blocks $B_0, \dots, B_k$ of sizes $b_0, \dots, b_k$ respectively, we define for bit b_h in block B_i:

- $I(b_h)$ is the internal-carry delay of bit b_h in block B_i, i.e., it is the maximum delay at which a carry generated entirely within B_i (or one of its sub-blocks) can arrive at position b_h. Similarly, 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) is denoted $I(B_i) = \max_{b_h \in B_i} I(b_h)$;
- $G(B_i, B_j)$ 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;
- $A(b_h)$ is the carry-assimilate delay of bit b_h in block B_i, the maximum delay associated with propagating a carry within B_i to position b_h block B_i when that carry arrives at B_i from any previous block. Analogously to $I(B_i)$, we define the carry-assimilate delay of B_i as $A(B_i) = \max_{b_h \in B_i} A(b_h)$;
- $L_D(B_i)$ is the latest delay possible that a carry signal can depart B_i to a later (or enclosing) block; and
- $L_A(B_i)$ is the latest delay possible that a carry generated in any block before (or enclosing) B_i can arrive at B_i.

Note that these definitions all pertain to the time at which carry-in signals become available to a column or block. We also define:

- $C(b_h)$ is the latest delay possible at which the carry-out bit for position b_h within block B_i can be generated; and
- $S(b_h)$ is the latest delay possible at which the sum bit for position b_h within block B_i can be generated.

The sum signal for each column is generated at some delay after the arrival of the carry-in signal for the column or block. Our goal in designing a Hybrid Adder is to minimize the maximum of the $S(b_h)$ by minimizing the maximum of the carry 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 may not be 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, $k < i$.

2.1. *Hybrid Adders using Ripple-Carry and Carry-Skip Blocks*

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 $r, \ldots, s$ available at times $t_r, \ldots, t_s$. Using 1 XOR delay as our time unit, we then have that:

- $I(b_r) = 0$, and $I(b_j) = \max(I(b_{j-1}), t_{j-1}) + 1$ for $r < j \le s$. Thus, $I(B_i) = I(b_s) = \max_{r \le j \le s-1}(t_j + s - j)$;
- $G(B_i, B_j)$ is relevant only for $j = i + 1$, and $G(B_i, B_{i+1}) = \max(I(b_s), t_s) + 1 = \max_{r \le j \le s}(t_j + s - j + 1) = \max(I(B_i), t_s) + 1$;
- $P(B_i, B_j)$ is relevant only for $j = i + 1$, and $P(B_i, B_{i+1}) = \max(L_A(B_i) + s - r + 1, \max_{r \le j \le s}(t_j + s - j + 1))$;
- $A(b_r) = L_A(B_i)$, and $A(b_j) = \max(A(b_{j-1}), t_{j-1}) + 1$ for $r + 1 \le j \le s$. Thus, $A(B_i) = A(b_s) = \max(L_A(B_i) + s - r, \max_{r \le j \le s}(t_j + s - j))$;
- $L_A(B_0) = 0$ (assuming no carry in to the adder, otherwise the latest time at which it can arrive);
- $L_D(B_i) = L_A(B_{i+1}) = \max(G(B_i, B_{i+1}), P(B_i, B_{i+1}))$ for $i \ge 0$; and
- $S(b_j) = \max(\min(t_j, \max(I(b_j), A(b_j))) + 2, \max(t_j, I(b_j), A(b_j)) + 1)$.

Observation 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}, and $L_D(B_i) = L_A(B_{i+1})$ will also be $\le t_{s+1}$. 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 examine Carry-Skip blocks. First, we assume that in Carry-Skip blocks the skips can always be completed (for any size block) by time $\max(\max_{r \le j \le s}(t_j) + 2, \max(L_A(B_i) + 1)$. The delay of 1 corresponds to a NAND with the propagate signal to get the complemented skip signal followed by a NAND with the complemented generated carry-out to give a single carry-out signal. The delay of 2 corresponds to the delays of two NOR gates to combine the input bit signals to get the propagate signal plus the 1 delay previously described. (This assumption is reasonable for small blocks and blocks where $L_A(B_i) \gg \max_{r \le j \le s}(t_j)$. In situations where the assumption does not hold functions based on the circuit that minimizes the maximum delays for the block can be easily derived. The optimal circuits for those situations vary greatly with the input profile, and will not be discussed here other than to say that the basic case above suffices for final adders which take their inputs from PPRTs based on current designs and technology.) For Carry-Skip blocks we have that:

- $I(b_r) = 0$, and $I(b_j) = \max(I(b_{j-1}), t_{j-1}) + 1$ for $r < j \le s$. Thus, $I(B_i) = I(b_s) = \max_{r \le j \le s-1}(t_j + s - j)$;
- $G(B_i, B_j)$ is relevant only for $j = i + 1$, and $G(B_i, B_{i+1}) = \max(I(b_s), t_s) + 1 + 0.75 = \max_{r \le j \le s}(t_j + s - j + 1) + 0.75$ (the 0.75 is for NOT and NAND gates to combine the generated and skipped carries between blocks);
- $P(B_i, B_j)$ is relevant only for $j = i + 1$, and $P(B_i, B_{i+1}) = \max(\max_{r \le j \le s}(t_j) + 2, L_A(B_i) + 1)$ as described above;
- $A(b_r) = L_A(B_i)$, and $A(b_j) = \max(A(b_{j-1}), t_{j-1}) + 1$ for $r + 1 \le j \le s$. Thus, $A(B_i) = A(b_s) = \max(L_A(B_i) + s - r, \max_{r \le j \le s}(t_j + s - j))$;
- $L_A(B_0) = 0$ (assuming no carry in to the adder, otherwise the latest time at which it can arrive); and
- $L_D(B_i) = L_A(B_{i+1}) = \max(G(B_i, B_{i+1}), P(B_1, B_{i+1}))$ for $i >= 0$; and
- $S(b_j) = \max(\min(t_j, \max(I(b_j), A(b_j))) + 2, \max(t_j, I(b_j), A(b_j)) + 1)$.

Observation 2. If B_i is a Carry-Skip block over positions $b_r, \ldots, b_s$ in an optimal Hybrid Ripple-Carry/1-Level Carry-Skip Adder, it must be that $L_A(B_i) \ge \max_{r \le j \ge s}(t_i) + 1$. Otherwise, if we let b_k be the least significant bit in B_i such that $t_k > L_A(B_i) - 1$. Then (by assumption) $P(B_i, B_{i+1}) = \max(\max_{r \le j \le s}(t_j) + 2, L_A(B_i) + 1) = \max_{r \le j \le s}(t_j) + 2 \ge t_k + 2$. But we can split B_i into two blocks $B_{i'}$ (over bits b_r through b_{k-1}) and $B_{i''}$ (over bits b_k through b_s) 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 $G(B_{i''}, B_{i+1}) \le G(B_i, B_{i+1})$ (because $B_{i'}$ and $B_{i''}$ include (disjoint) subranges of B_i;
- $P(B_{i'}, B_{i''}) < P(B_i, B_{i+1})$ (because $P(B_{i'}, B_{i''}) = \max(\max_{r \le j \le k-1}(t_j) + 2, L_A(B_i) + 1) = L_A(B_i) + 1 < t_k + 2 \le P(B_i, B_{i+1})$; and
- $P(B_{i''}, B_{i+1}) \le P(B_i, B_{i+1})$ (because $P(B_{i''}, B_{i+1}) = \max(\max_{k \le j \le s}(t_j) + 2, L_A(B_{i'}) + 1) = \max$

$(\max_{k \leq j \leq s}(t_j) + 2,\ L_A(B_i) + 2) = \max_{k \leq j \leq s}(t_j) + 2 \leq P(B_i, B_{i+1}))$.

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

Observation 3. 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:

- $P(B_i, B_{i+1}) < \max(L_A(B_i), t_{s+1}) + 2$, and
- $G(B_i, B_{i+1}) < P(B_i, B_{i+1}) + 1$.

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

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

Observation 4. 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(a) shows the delays associated with an optimal adder design for using one level of Carry-Skip blocks to add two 62-bit numbers whose signals are all available at time 0 in total time equivalent to 15.75 XOR delays. Figure 2(b) shows the output delays of the same adder when applied to the profile of Fig. 1, giving a maximum output delay of 25.75 XOR delays. Thus when applied to the profile the latest output is later by one less than the delay the latest input signal, even though many of the signals arrive significantly earlier.

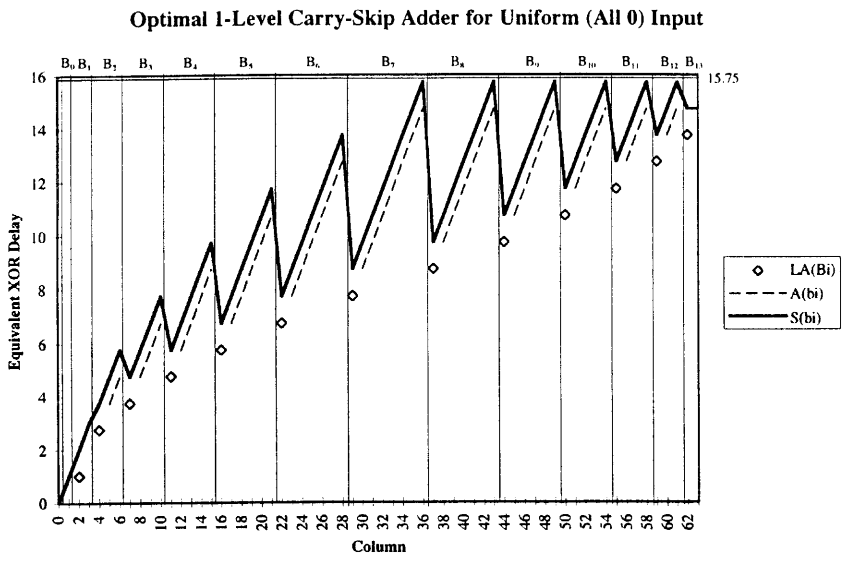
Now we use the formulas above to construct a Hybrid Adder for the profile in Fig. 1 using Ripple-Carry and Carry-Skip blocks. First we note that based on the input signal profile vector, Observation 1 applies to columns $1, \ldots, 5$ and nowhere else (Column 0 has only one bit as input). Thus, we can set B_0 as a Ripple-Carry block including at minimum columns $1, \ldots, 5$. If B_0 included only those bits then we would have that $L_D(B_0) = 5 = t_6$. However, we would then have by Observation 2 that B_1 would include only one position (b_6) (because $t_7 = 5$). The resulting maximum carry-in delay to the block starting with position b_7 is the same (6) in both cases, so we use the simpler design whereby B_0 covers $b_1, \ldots, b_6$. Observation 1 does not apply past column 6, so 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 $L_A(B_1) = L_D(B_0) = 5$. 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 3 and 4 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 Fig. 3(a). By searching on the values between the lower bound of 20.75 ($P(B_{13}, B_{14})$) and the upper bound of 27.25 (the maximum value of $S(b_i)$) we obtain the design in Fig. 3(b), with latest output at 23.75 XOR delays (versus 25.75 for the optimal Uniform Input design). Thus we have achieved almost 8% improvement using an adder of comparable complexity.

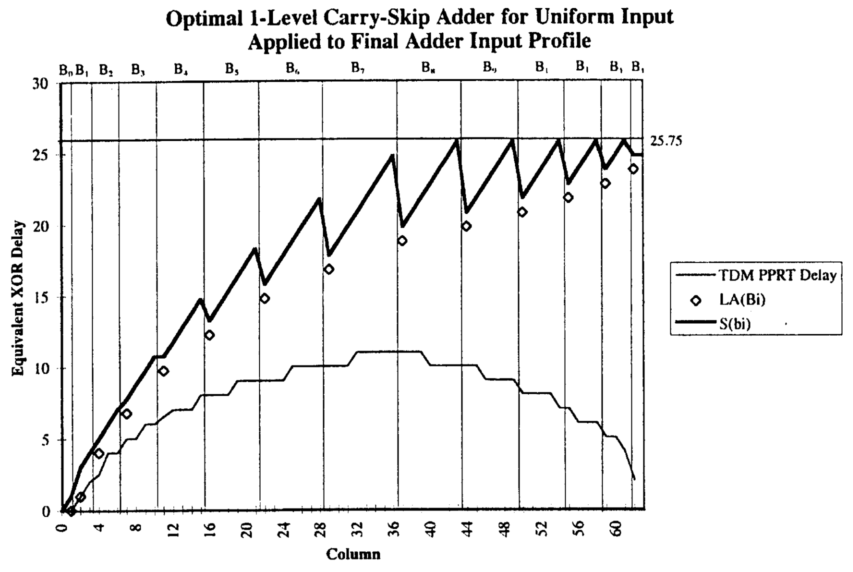
This design is optimal among all Hybrid Adders that use only Ripple-Carry and 1-level Carry-Skip blocks. We now consider the situation where we also allow designs that include one Carry-Select block.

2.2. *Using Carry-Select Blocks*

The principles described above can be appled to multi-level Carry-Skip blocks, Carry-Lookahead blocks, and



(a)

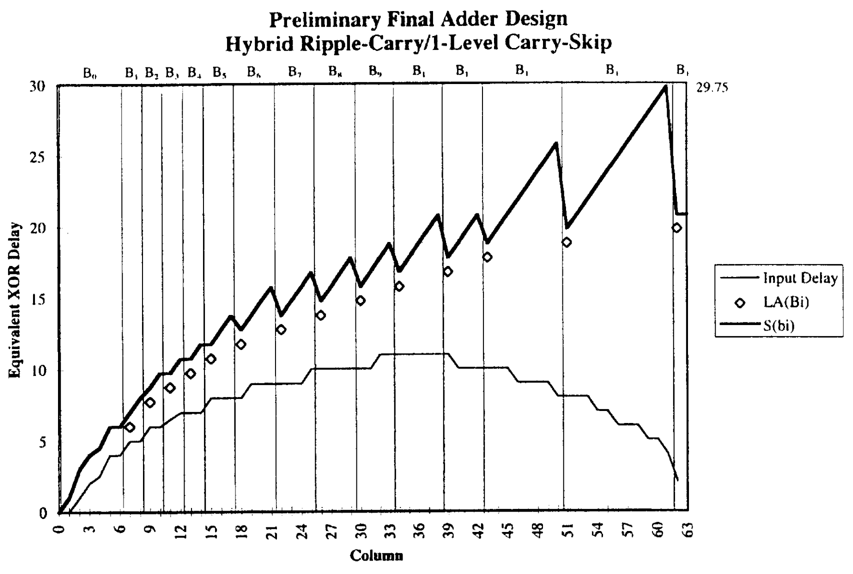


(b)

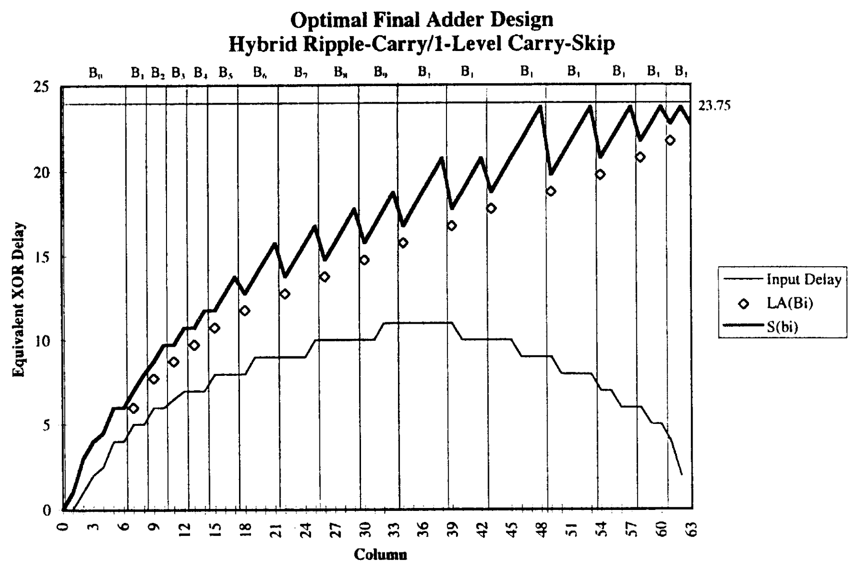
Figure 2. The delays of the optimal 62-bit 1-Level Carry-Skip Adder for uniform inputs when applied to (a) uniform inputs and (b) the 32-bit Latest-Earliest TDM PPRT output profile.

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 the delay functions are based on the values of the delay functions for the underlying sub-blocks of the Carry-Select block, the time of the carry-in $L_A()$, additional delays corresponding to the increased load (due to the number of output lines [7]) on the gate that generates $L_A()$, and the delay of the multiplexors used. (Multiplexors have approximate delay of 1 XOR from the select line and 0.75 XOR delays from the data lines, and inverting multiplexors have approximate delay of 0.5 XOR delay from the select line and 0.875 XOR delays from the data lines [7].) Due to the nature of Carry-Select blocks, we can limit our analysis of a Carry-Select block B_i containing bits $b_r, \ldots, b_s$ to finding $S(b_j)$ for each bit in B_i and $C(B_i)$ (and by implication $L_D(B_i)$).



(a)



(b)

Figure 3. The delays of the (a) preliminary and (b) optimal 62-bit Hybrid Adder using Ripple-Carry and 1-level Carry-Skip blocks for the 32-bit Latest-Earliest TDM PPRT output profile.

When there can be confusion regarding which block a signal corresponds to, we will use the identifying subscript of the appropriate block. For example, the sum signal for b_j out of Carry-Select block B_i will be referred to as $S_i(b_j)$, and the sum signal for b_j out of sub-block $B_{i:k}$ (and input to B_i) will be referred to as $S_{i:k}(b_j)$.

For example, if a Carry-Select block B_i has (immediate) sub-blocks $B_{i:1}, \ldots, B_{i:h}$, and if we let d be the additional delay associated with the gate that generates $L_D(B_i)$ due to the number of output lines, then we have that:

- $S_i(b_j) = \max(L_A(B_i) + 1, S_{i:k}(b_j) + .875)$;
- $C_i(b_j) = \max(L_A(B_i) + 1, C_{i:h}(B_{i:h}) + .875) + d$ if B_i is the last sub-block of an enclosing Carry-Select block; and
- $L_D(B_i) = \max(L_A(B_i) + 1, C_{i:h}(B_{i:h}) + .875) + d$.

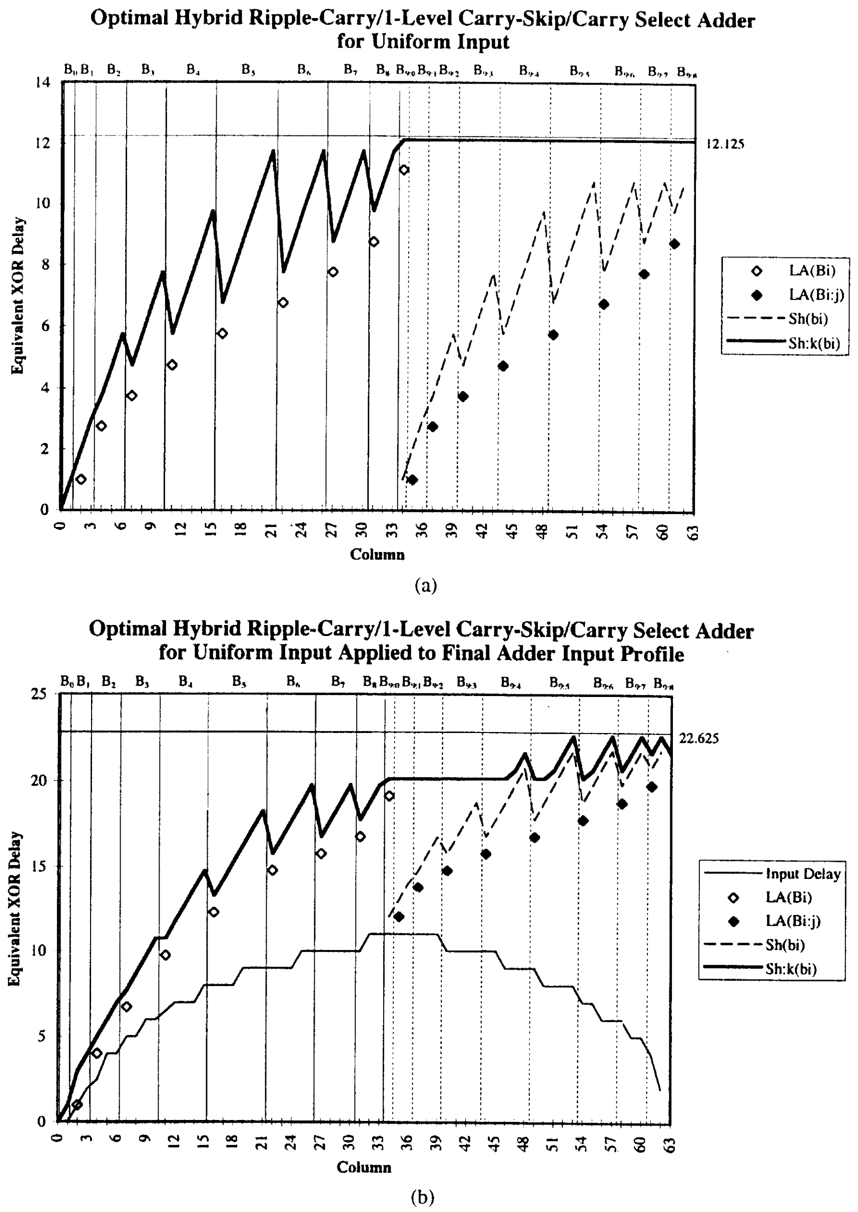


Figure 4. The delays of the optimal 62-bit Hybrid Adder for uniform inputs using 1-level Carry-Skip and one Carry-Select block when applied to (a) uniform inputs and (b) the 32-bit Latest-Earliest TDM PPRT output profile.

The delay from a NAND gate (which generates the carry-out from a Carry-Skip block) is approximately (.35 ns + .04 ns × (# output lines). For a Carry-Select block over 30 columns there are 31 output lines (1 for the carry-out) for a total delay of 1.59 ns, or approximately 1.875 XOR delays, so that $d = 1.375$. If the Carry-Select block instead is over only 21 columns, then the 22 output lines imply a delay of 1.23 ns, so that $d = 1$ XOR delay.

The design approach is similar to the one used previously. We know that the Carry-Select block will begin 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 achieve progressively smaller delay bounds beginning with the optimal result from the design of Fig. 3(b). As the bound is decreased some of the Carry-Skip

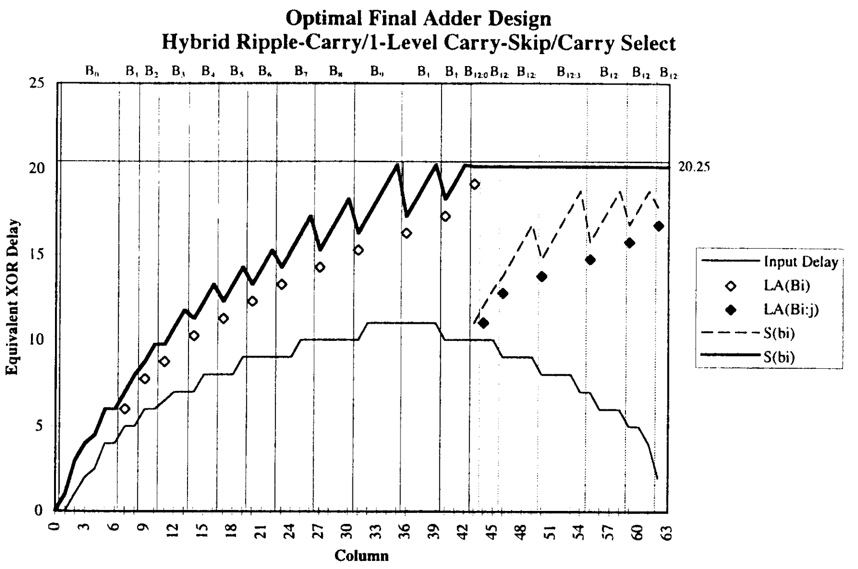


Figure 5. The delays of the optimal 62-bit Hybrid Adder using Ripple-Carry, and 1-level Carry-Skip, and one Carry-Select blocks for the 32-bit Latest-Earliest TDM PPRT output profile.

blocks may need to be changed to remain in compliance with it, but the Carry-Select block will always begin immediately following an existing block, so only a limited search is necessary. Figure 4(a) shows the delays associated with an optimal Hybrid Adder with Ripple-Carry, Carry-Skip, and one Carry-Select block, with maximum sum output delay of 12.125 XOR delays. Figure 4(b) shows the delays that result if that design is applied to the Latest-Earliest TDM PPRT output profile, yielding a maximum delay of 22.625 equivalent XORs. An optimal design for the Latest-Earliest TDM PPRT output profile is shown in Fig. 5. This design has latest maximum delay of 20.25 XOR delays. Thus we have achieved better than 10% improvement over the original design for a uniform signal profile. The output profiles for all four of the designs discussed are given together in Fig. 6 for comparison.

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 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. In our examples we have normalized our results to equivalent XOR delays, but our approach works equally well with the raw timing delays of the gates used in the designs.

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 32×32-bit multiplier. The improvement in speed is estimated to be 10% over commonly used CLA scheme. This improvement almost directly translates to total multiplication time since our time measures are relative to the time at which the Partial Product signals become available to the PPRT in the 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 only slightly better than 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.

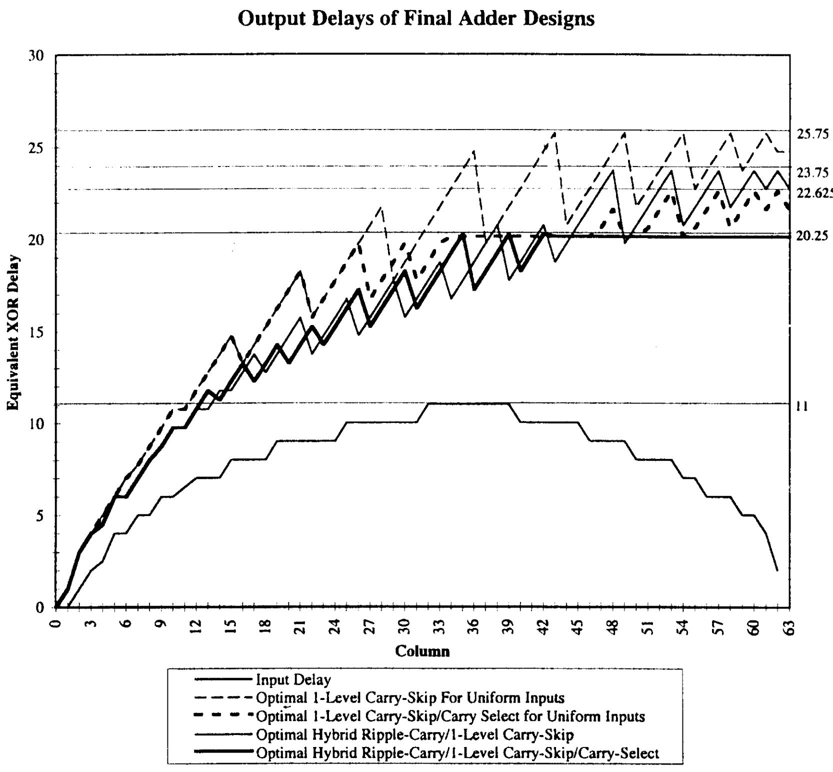


Figure 6. The output delay profiles of the four adder designs discussed when applied to the Latest-Earliest TDM PPRT output profile for a 32-bit multiplier.

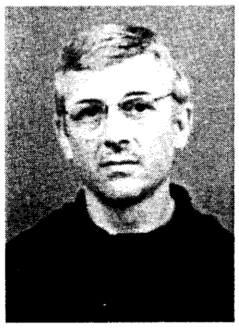
References

1. E.E. Swartzlander (Ed.), *Computer Arithmetic*, Los Alamitos, CA: IEEE Computer Society Press, Vols. 1 & 2, 1990.
2. V.G. Oklobdzija, D. Villeger, and S.S. Liu, "A Method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Transactions on Computers*, Vol. 45, No. 3, pp. 294–306. March 1996.
3. C. Martels, V.G. Oklobdzija, R. Ravi, and P.F. Stelling, "Design strategies for optimal multiplier circuits," *Proceedings of the 12th Symposium on Computer Arithmetic*, pp. 42–49, 1995.
4. V.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.
5. P.F. Stelling and V.G. Oklobdzija, "Design strategies for the final adder in a parallel multiplier," *Proceedings of the 29th Asilomar Conference on Signals, Systems, and Computers*, pp. 591–595, 1995.
6. P.K. Chan, M.D.F. Schlag, C.D. Thomborson, and V.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, Aug. 1992.
7. *1.0-Micron Array-Based Products Databook*, LSI Logic Corporation, Sep. 1991.
8. V.G. Oklobdzija and E.R. Barnes, "Some optimal schemes for ALU implementation in VLSI technology," *Proceedings of the 7th Symposium on Computer Arithmetic*, 1985.
9. V.G. Oklobdzija and E.R. Barnes, "On implementing addition in VLSI technology," *Journal of Parallel and Distributed Computing*, Vol. 5, pp. 716–728, 1988.
10. B.D. Lee and V.G. Oklobdzija, "Improved CLA scheme with optimized delay," *Journal of VLSI Signal Processing*, Vol. 3, pp. 265–274, 1991.
11. V.G. Oklobdzija and D. Villeger, "Improving multiplier design by using improved column compression tree and optimized final adder in CMOS technology," *IEEE Transactions on VLSI Systems*, Vol. 3, No. 2, pp. 292–301, June 1995.
12. V.G. Oklobdzija and D. Villeger, "Optimization and analysis of a carry-propagate adder under the non-uniform signal arrival profile," (in preparation).

Paul F. Stelling received the Ph.D. degree from the University of California, Davis (1995), the M.S. degree from the University of

Nebraska (1982), and the B.S. degree from Claremont McKenna College (1978).

He is currently at the University of California, Davis as a researcher and lecturer. His research interests include the application of computing theory to such areas as circuit design, computational biology, and computer security, and parallel processing and algorithms.
stelling@cs.ucdavis.edu



Vojin G. Oklobdzija obtained the Dipl. Ing. (M.Sc.E.E.) degree in electronics and telecommunications from the Electrical Engineering Department of the University of Belgrade, Yugoslavia, in 1971, and stayed on the faculty there until 1976 when he became a Fullbright Scholar at the University of California at Los Angeles. He obtained his M.Sc. and Ph.D. in computer science from UCLA in 1978 and 1982, respectively. After receiving his Ph.D. he became a research staff member at the IBM T.J. Watson Research Center in New York where he spent eight years making contributions to development of RISC and super-scalar RISC architecture and processors. He left IBM in 1991 and today is with Integration Berkeley, California, and the Electrical and Computer Engineering Department of the University of California at Davis. As a visiting faculty member from IBM, Professor Oklobdzija taught courses in computer architecture, computer arithmetic, and computer design at the University of California at Berkeley from 1988 to 1990. His industrial experience includes positions at the Microelectronics Center of the Xerox Corporation, and consulting positions at Sun Microsystems Laboratories, AT&T Bell Laboratories, and various others.

Dr. Oklobdzija has published more than 80 papers in the areas of circuits and technology, computer arithmetic, and computer architecture. He has presented many invited talks in the U.S., Europe, Latin America, Australia, China, and Japan. He holds one of the patents on the IBMRS/6000—"PowerPC" architecture, and four U.S. and four European patents in the area of circuits and computer design. His current research interests are in VLSI and fast circuits, efficient implementations of algorithms, and computation.

He is a fellow of the IEEE, Member of the American Association of the University Professors and New York Academy of Science. He serves on the editorial boards of the *Journal of VLSI Signal Processing* and *IEEE Transactions on VLSI Systems*, and on the program committees of the International Solid-State Circuits Symposium, the International Symposium on VLSI Technology, and the International Conference on Computer Design.
vojin@ece.ucdavis.edu