IMPLEMENTING MULTIPLY-ACCUMULATE OPERATION IN MULTIPLICATION TIME

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

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

Abstract

Multiply-Accumulate is an important and expensive operation. It is frequently used in Digital Signal Processing and video/graphics applications. As a result, any improvement in the delay for performing this operation could have a positive impact on clock speed, instruction time, and processor performance. In this paper we show how by extending our view of a parallel multiplier we can apply recent innovations in parallel multiplier design to multiply-accumulators. This application results in multiply-accumulators that are as fast as multipliers of the same size. (These multipliers have been shown to result in provably optimal delays faster than current designs.) This allows a single (optimal multiply-accumulate) circuit to be used for both operations without delay penalty. As a result, multiply-accumulate can be efficiently and effectively implemented as an instruction in RISC CPUs. Additionally, the circuit design reduces the number of devices needed over current fast multiplier designs, so that real estate and power savings also result.

Keywords: *Multiply-accumulate, Parallel Multiplier, Partial Product Reduction Tree, Final Adder, Algorithms, VLSI circuits.*

1. INTRODUCTION

The problem of designing improved multipliers is one that has gained a lot of attention recently. A related problem that has not received as much attention is that of designing improved multiply-accumulators. Here we apply lessons learned from analyzing parallel multiplier design to the design of multiply-accumulators. The resulting multiply-accumulate circuit is as fast as the optimal multiplier of the same size [14], so that the same circuit can be used for both multiplication and multiply-accumulate, allowing the multiply-accumulate instruction to be included in RISC CPUs with minimal impact on clock speeds, pipelines, chip real estate usage, and power requirements.

Improvement in parallel multiplier design has been in two areas: Partial Product Reduction Tree Design and Final Adder Design. 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 [9,10], and optimal TDM PPRT designs for reducing the partial products to two rows typified. Following their usage, 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 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 were of the following, stricter, pattern:

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

$t_0 \le t_1 \le \ldots < t_k = t_{k+1} = \ldots = t_p > t_{p+1} \ge \ldots \ge t_{2m-2}$. These designs were faster than current competing designs.

In [7],[12],[13],[14] Stelling and Oklobdzija showed that the traditional adder designs for uniform inputs can be improved on significantly by using adder designs that are tailored to the input delay profile. Specifically, they showed how to design adders that decreased the incremental delay of the Final Adder by over 20% compared to traditional Conditional Sum designs. They further showed that their Final Adder designs are delay-optimal, i.e., no other design based on Carry-Select and Conditional Sum logic can have smaller incremental delay.

In this paper we will extend those results to the design of a multiply-accumulator that takes as input two n -bit factors A and B and a $2n$ -bit addend C and computes $((A \times B)+C)$ in the same delay as an optimal n -bit multiplier.

2. APPROACH

Our approach utilizes both of the results mentioned above. First, we recognize that we can combine the $2n$ -bit addend with the partial products in the TDM PPRT, and then we show that an optimal Hybrid Adder for combining the resulting two ``numbers'' can be designed which is as fast as the Hybrid Adder which would be used for multiplication.

2.1. Using the TDM PPRT

In an n -bit parallel multiplier, the partial products are first generated in the manner of long multiplication, and then reduced using a partial product reduction tree to two bits per column. We use the convention of numbering the columns in increasing order of significance from 0 to $(2n-1)$, where the i^{th} column has the bits of value 2^i. Initially there are $(k+1)$ bits input to the PPRT for each of the columns $k = 0,\ldots,(n-1)$, and $(2n-k)$ bits input to the PPRT for each of the columns $k = n,\ldots,(2n-1)$. The PPRT combines these bits until there remain 2 bits per column for columns 1 through $(2n-1)$ (and the original bit for column 0). In TDM PPRTs columns with odd numbers of input bits (≥ 3) have their parity *adjusted* through the use of a single half-adder on the two earliest inputs.

Our approach is to include the bits from the addend with the partial products as inputs to the PPRT. This results in there being $(k+2)$ inputs to each column $k = 0,\ldots,(n-1)$, and $(2n-k+1)$ inputs to each column n through $2n$. Thus, for $i = 0,\ldots,2n$, the i^{th} column now has the same number of inputs as the $(i+1)^{th}$ column of a n -bit parallel multiplier. Therefore the optimal PPRT for an n -bit multiply-accumulator is exactly the same as the optimal PPRT for an n -bit multiplier. The number of Partial Product bits and adders in a TDM PPRT for the columns are shown in Table 1. Thus for an n -bit multiplier the number of Half Adders used is $(n-1)$ and the number of Full Adders is $(n-1)(n-3)$, so that the TDM PPRT for a multiply-accumulator has an additional 1 Half Adder and $(2n-3)$ Full Adders (1 AND, $(4n-5)$ XORs, and $(8n-12)$ NANDs).

Fig. 1 shows the output delay profiles of the Latest-Fewest TDM PPRTs for a 32-bit parallel multiplier and for a 32-bit parallel multiply-accumulator.

Table 1: *Number of Input Bits and Gates in a PPRT for n-bit Multiplication* $(2 \le k \le 2n-2)$

Column k :	$k \le n-1$	$k = n$	$k \ge n+1$
# Partial Product Bits	$k+1$	$k-1$	$2n-k-1$
# Carry-In Bits	$k-2$	$k-2$	$2n-k-1$
Total Input Bits	$2k-1$	$2k-3$	$4n-2k-2$
# Half Adders	1	1	0
#Full Adders	$k-2$	$k-3$	$2n-k-2$
Total Half Adders	$n-1$		
Total Full Adders	$(n-1)(n-3)$		

2.2. Using the Optimal Final Adder Design Strategies

Next, we design a Final Adder for the multiply-accumulator using the strategies presented by Stelling and Oklobdzija in [14]. That analysis follows the convention in [9],[10],[12],[13] of measuring delays in equivalent (fan-out 1) XOR delays. The equivalent XOR delays used were computed from the actual delays given in [11] and are shown in Table 2. (The delays in [11] are provided only for output line counts that are powers

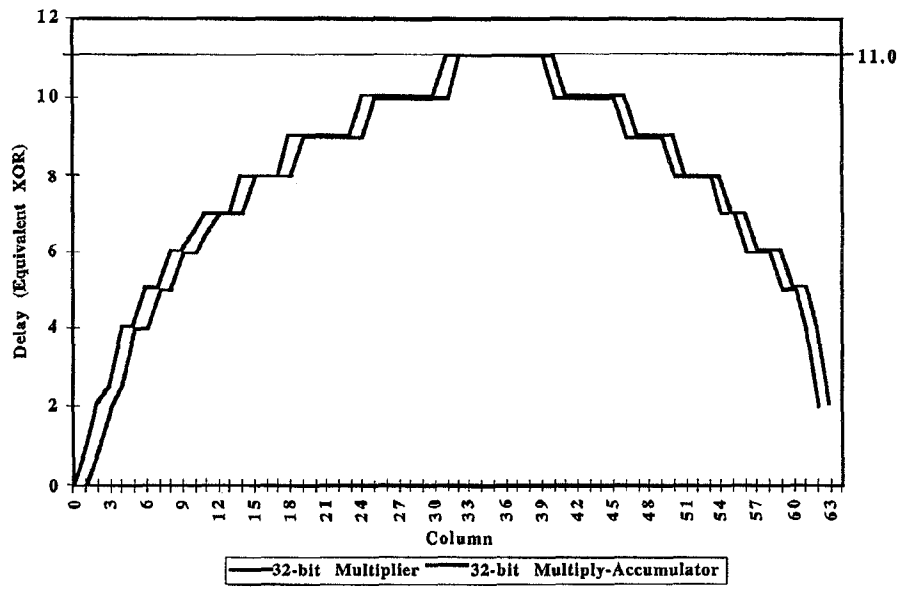


Fig. 1: *The Latest-Fewest TDM PPRT output profiles of a 32-bit Parallel Multiplier and a 32-bit Multiply-Accumulator*

of 2, the numbers in parentheses represent the numbers of output lines for which we used the given values.)

In [14] two distinct designs are given for Hybrid Adders whose input arrival profiles are the output delay profiles for the 32-bit Latest-Fewest TDM PPRT. These designs are shown not only to be 20% faster than traditionally used Conditional Sum designs, but also to be delay optimal over all designs based on Conditional Sum or nested Carry-Select blocks (they are also simpler, due to decreased nesting). The structure and delays of one of those adders are depicted in Fig. 2. Thus the optimal 32-bit parallel multiplier based on the new TDM PPRT and Hybrid Adder technology finishes 16.0 XOR delays after the generation of the partial products. We will now show that with only minor modification the Hybrid Adder used in this design can be used on multiply accumulate TDM PPRT output described above to achieve the same lower bound. In [14] a lower bound on the delay is derived based on the

Table 2: *Input to Output (Equivalent XOR) Delays for Typical Logic Elements by Number of Output Lines (Fan-Out)*

Device	Input Line(s)	Number of Output Lines					
		1	2	4 (3-5)	8 (6-10)	16 (11-18)	32 (19+)
Inv Mux	S	0.5	0.5	0.5	0.75	1	1.25
	A,B	0.875	0.875	1	1	1	1.125
Mux	S	1	1	1	1	1.125	1.5
	A,B	0.75	0.875	0.875	1	1	1.125
NAND		0.5	0.5	0.5	0.625	0.875	1.125
NOR		0.5	0.5	0.625	0.75	1	1.75
NOT		0.25					

delay necessary to use a Conditional Sum adder to add the columns of latest input and also to combine the results for those columns

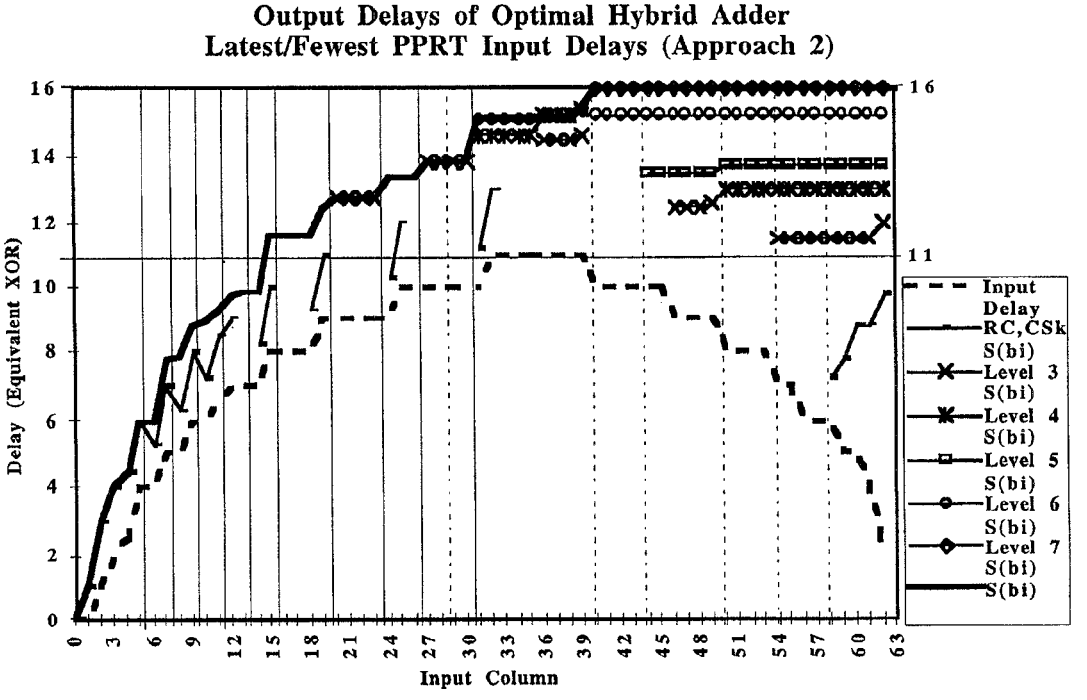


Fig. 2. *The delays of an optimal Hybrid Adder for the Latest-Fewest TDM PPRT output profile of a 32-bit Parallel Multiplier.*

with the results for all other columns, assuming that the results for the other columns can be derived as early as necessary. The specifics of that analysis follow. In the case of the Final Adder for a 32-bit multiplier, there are eight columns (numbers 32 through 39) with the maximum delay of 11 XORs. By using a Symmetric Conditional Sum block on those positions we can generate the (alternative) outputs for those columns at time 14.625, and the carry-out signals for propagation at time 14.375. If we let B_0 be a block (or blocks) containing columns 0 through 31, B_1 be a block containing columns 32 through 39, and B_2 be a block (or blocks) containing columns 40 through 62, then it remains to combine the Conditional Sum block for B_1 with adder blocks for B_0 and B_2. The fastest way these could be combined is by Carry Selection. This could be done in either of two ways: by using cascading carry selection from B_0 to B_1 to B_2; or by cascading carry selection from B_0 to a larger block consisting of B_1 and B_2, with those two combined by cascading carry select. Suppose for the moment that the carry-out signal from B_0 can be generated arbitrarily early (at time t_0), as can the alternative sum and carry-out output signals for B_2 (at time t_2). Then under both approaches the final output sum bits for columns 32 through 39 will be available at time $\max(t_0+1.0, 14.625+0.75)=15.375$. Under the first approach the cascading carry-out bit from B_1 would be available at time $\max(t_0+1.25, 14.375+1.125)=15.5$, and the sum bits and carry-out bit from the columns in B_2 would be available at time $\max(15.5+0.5, t_2+0.875)=16.0$. Note that this bound is achieved so long as $t_0 \le 14.25$ and $t_2 \le 15.125$.

Under the second approach the number of output lines from the carry-out of B_1 is 24, increasing the output delay from 3.375 to 3.75, so that the carry-out signal is generated at time 14.75. The sum and carry-out outputs of B_2 would thus be available at $\max(14.75+0.5, t_2+0.875)=15.25$, and the final output delays after the cascading carry select from B_0 would be $\max(t_0+1, 15.25+0.75)=16.0$. Note that this bound is achieved so long as $t_0 \le 15.0$ and $t_2 \le 14.375$. Thus we have a lower bound on the total delay of 16.0, achievable by both approaches, but with differing bounds on the delays of the output signals from B_0 and B_2. To achieve the lower bound established above we

must generate the final sum bits for the columns in B_0 by time 16.0.

Also, to use the first approach we must generate the cascading carry-out signal from B_0 by time 14.25 and

the sum and carry-out signals for the columns in B_2 by time 15.125. To use the second approach we must generate the cascading carry-out signal from B_0 by time 15.0 and the sum and carry-out signals for the columns in B_2 by time 14.375.

In [14] designs are presented that achieve the lower bound using both of the two approaches. We now show that with minor modification to the design of the second approach (represented in Fig. 2) we can convert that adder to an Hybrid Adder that takes 64-column input with the delay of the 32-bit multiply-accumulator TDM PPRT from above, and generates the final result in the same delay.

To do this, we note that the input profile for the multiply-accumulate final adder is the same as for the multiplier, with the exception that there are two additional columns of delay 11. We will divide the columns into three blocks, B_0' with columns 0 through 32, B_1' with columns 33 through 40, and B_2' with columns 41 through 63. First we note that B_1' and B_2' will have precisely the same input delays in the multiply-accumulator as B_1 and B_2 have in the multiplier (the design in [14] actually uses other results to include column 31, of delay 10, but that enhancement is not needed for delay optimization in this case). Thus by using the same design for B_1' as for B_1, and for B_2' as for B_2, we achieve performance that will allow us to achieve the lower bound.

It remains to show that we can add the columns in B_0' to get the sum bits by time 16.0 and the carry-out signal from B_0 by time 14.25. We do this by using the same design for the first 30 columns of B_0' as was used for B_0, generating the sum signals for those columns by time 13.875, and also the carry-out from column 29 by time 13.375. If we use a carry-select block for columns 30 through 32, then we can use any sub-block design we desire, so long as the sum signals from the select are available by time 16.0, and the carry-out is available by time 15.125. This would require that the sub-block sum signals be available by time 15.125 and the carry-out be available by time 14.0. But all of those signals will be availble by time 13.0 if a simple Ripple-Carry Adder is used over those three columns (in fact two Ripple-Carry Adders are needed -- one assuming a carry-in value of 0

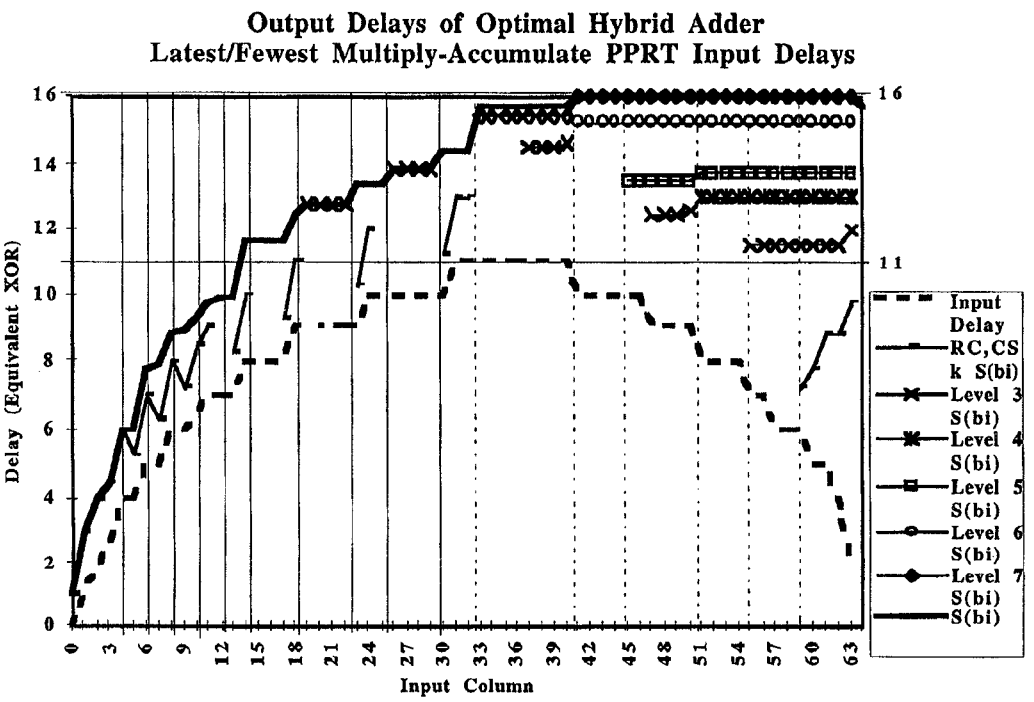


Fig. 3. *The delays of the optimal Hybrid Adder for the Latest-Fewest TDM PPRT output profile of a 32-bit Multiply-Accumulator.*

and the other a carry-in value of 1). Therefore by inserting the required sub-blocks for those columns we achieve the desired performance with the addition of very few logic gates (and by implication, circuit devices). The total number of devices thus used is considerably fewer than the number used by the Final Adder for a multiplier

if implemented as a symmetric Conditional Sum Adder. Fig. 3. shows the delays of the resultant adder, as well as its block and major sub-block divisions. The block diagram of resulting Final Adder is shown in Fig. 4.a.b.c.d.

By combining the two innovations we have a multiply-accumulator design that is as fast as a delay-optimal multiplier of the same size. Our research shows that this result can be achieved for most sizes of multiply-accumulators, including the common case of 16-bit factors and a 32-bit addend.

3 CONCLUSION

We extended previous results in the analysis and design of optimal TDM PPRTs and Final Adders for parallel multipliers to the problem of designing fast Multiply-Accumulators. We showed that for commonly used sizes multiply-accumulators can be designed that are as fast as parallel multipliers of the same size. Thus we developed a design strategy for designing multiply-accumulate circuits which models predict will be as fast as new (optimal) multiplier designs, which are in turn predicted to be faster than current designs. These results allow the use of the same (multiply-accumulate) circuits for both multiplication and multiply-accumulate operations with no penalty in the delay of the operations, and with reductions in the number of devices used compared to current fastest designs. As a result, RISC processors using the new circuits can include Multiply-Accumulate in their instruction set while reducing the delay, real-estate, and power requirements associated with the Multiply instruction.

We are presently working on extending our results to find the simplest (fewest gates) circuits with the optimal delay. With our approach we have determined that we can build faster adders using simpler circuitry, but we have not yet completed optimal designs. Additional future work is the simulation and implementation of these circuits.

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 and 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," IEEE Transaction on Computers, March 1996.

[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] 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).

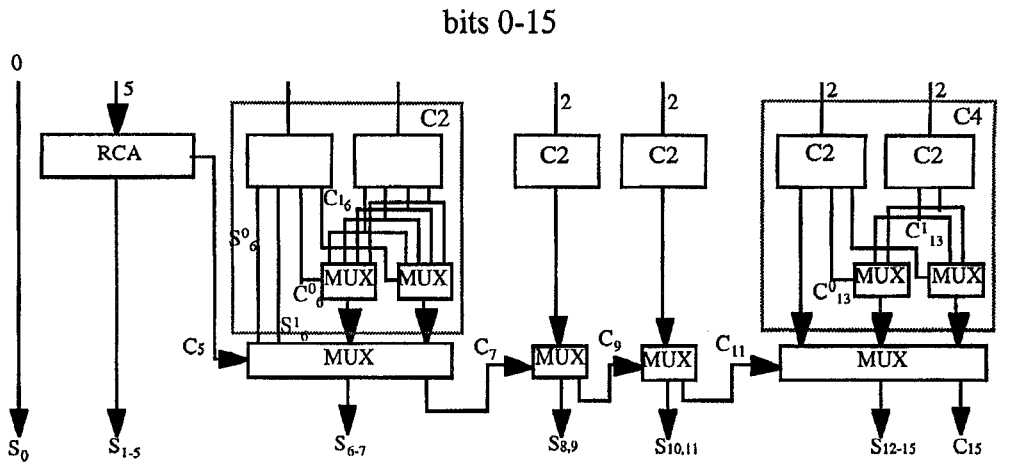
[10] Charles Martel, Paul F. Stelling, Vojin Oklobdzija, and R. Ravi, "Optimal Circuits for Parallel Multipliers", in submission IEEE Transacton on Computers 1996.

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

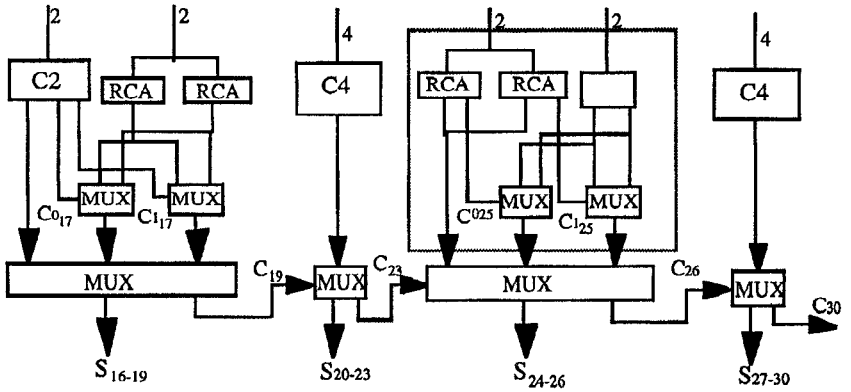
[12] Paul F. Stelling and Vojin G. Oklobdzija, "Design Strategies for the Final Adder in a Parallel Multiplier", Conference Record of the 29th Asilomar Conference on Signals, Systems, and Computers , pp.591-595, 1995.

[13] Paul F. Stelling and Vojin G. Oklobdzija, "Design Strategies for Optimal Hybrid Final Adders in a Parallel Multiplier", Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology, December 1996.

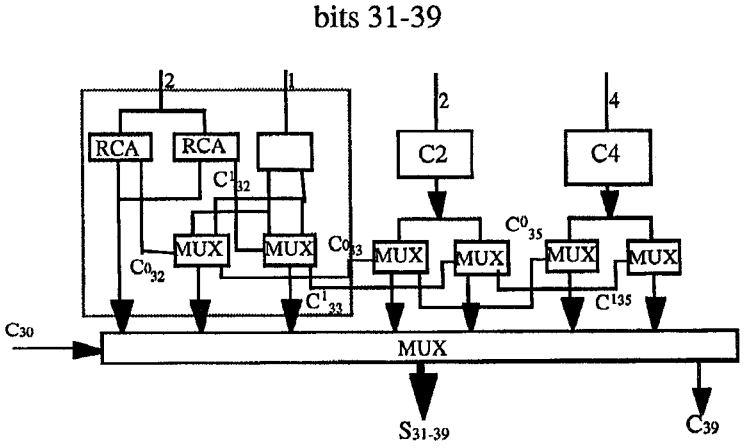
[14] Paul F. Stelling and Vojin G. Oklobdzija, "Designing Optimal Hybrid Final Adders in a Parallel Multiplier Using Conditional Sum Blocks", 15th IMACS World Congress 1997 on Scientific Computation, Modeling and Applied Mathematics, August 24-29, 1997.



(a)

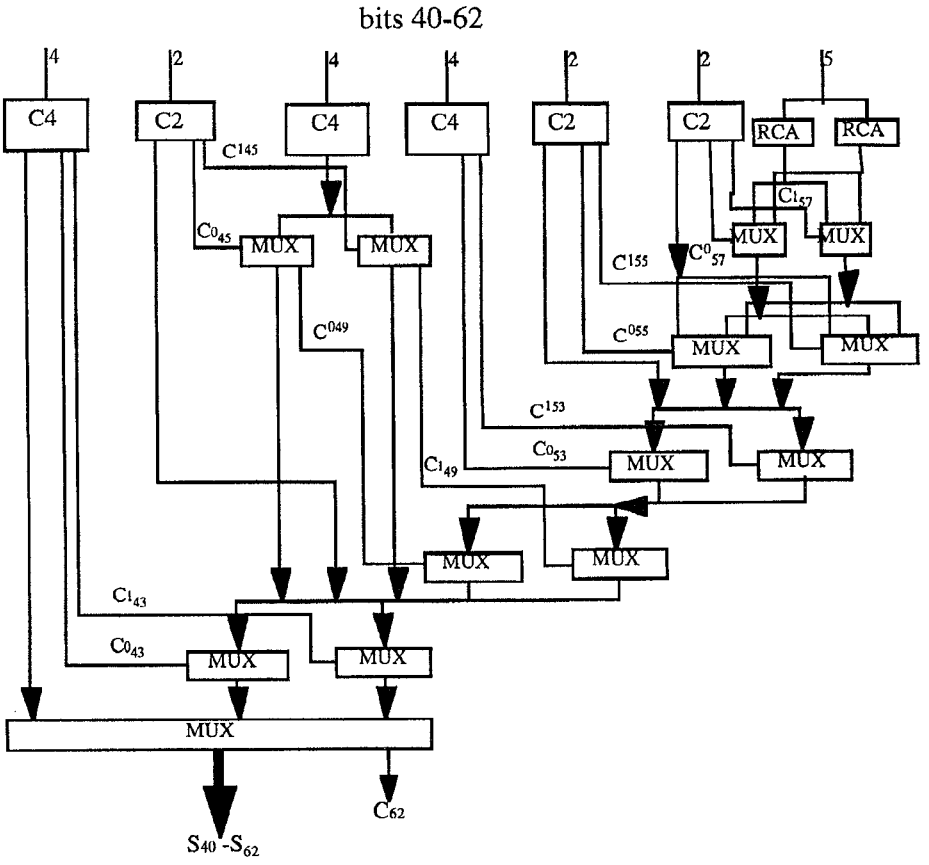


(b)



(c)

Figure 4: *The block diagram of the optimal Hybrid Adder for the Latest-Fewest TDM PPRT output profile of a 32-bit Multiply-Accumulator*



(d)

Figure 4: *The block diagram of the optimal Hybrid Adder for the Latest-Fewest TDM PPRT output profile of a 32-bit Multiply-Accumulator*