MERGED ARITHMETIC FOR SIGNAL PROCESSING

by

Earl. E. Swartzlander, Jr.

TRW Defense and Space Systems Group
One Space Park
Redondo Beach, California 90278

Abstract

The concept of merged arithmetic is introduced and applied to signal processing. The basic idea involves synthesizing a composite arithmetic function (e.g., a complex multiply) directly instead of decomposing it into multiply and add operations as is conventional practice. This approach results in a simpler design which is also faster.

Introduction

In the implementation of digital signal processing systems, conventional practice involves interconnecting arithmetic elements (i.e., adders, subtractors, and multipliers) and memories to perform the desired algorithms.[1] Consideration of array multiplication algorithms such as the Wallace tree[2] and its successor, Dadda's method,[3] which has been shown to be optimum,[4] suggests that when sums of products are to be formed, a saving in hardware and improvement in speed results if the multiplication computations are not completed and added as is conventional. The alternative approach, termed merged arithmetic since it dissolves the boundaries and merges the multipliers and adders into a single functional circuit, uses Dadda's matrix reduction scheme to produce the sum of products with a single reduction process. Before presenting this new approach, Dadda's multiplier algorithm, upon which it is based, will be reviewed.

Dadda's Multiplier

The sequence of operations required to perform parallel multiplication is shown in Figure 1. The first step is to generate the bit product matrix with an array of N^2 AND gates. During the second step, the matrix is reduced by counting the ones in each column and performing carry processing which results in a two-row matrix. The two rows are summed in a carry lookahead adder to generate the product of A and B.

The technique used to reduce the bit product matrix to a two-row matrix is illustrated for an 8 x 8 multiplier in Figure 2. At the top of the figure, the bit product matrix is shown as a skewed array of 8 x 8 dots (each dot represents one of the N^2 bit products). A counter[5] is a circuit with many inputs that generates a binary count of the number of inputs which are ones is provided for each column of the bit product matrix which has more than two elements. In this drawing, the inputs to each counter are in a rectangle which has a line extending down to the ouputs in the next matrix. Matrix II of Figure 2 shows the outputs of the counters (and those bit products which were not reduced with counters). Each of the counters has a sufficient number of outputs to indicate how many of the inputs are active (an i input counter has $1 + [\log_2 i]$ outputs*). All counter outputs except the least significant are carried up to more significant columns of matrix II. Since the two rightmost columns of matrix I have fewer than three entries, they are transmitted directly to matrix II. Column 3 of matrix I has three entries, so a counter is used to reduce it to an entry in column 3 and a carry to column 4 of matrix II. The next column of matrix I has four entries so its counter will have three outputs (corresponding to counts of 1, 2, and

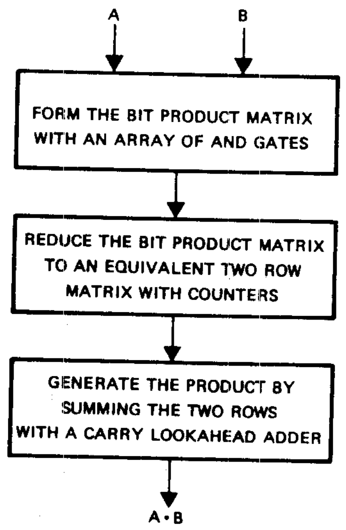
A
B
FORM THE BIT PRODUCT MATRIX WITH AN ARRAY OF AND GATES
REDUCE THE BIT PRODUCT MATRIX TO AN EQUIVALENT TWO ROW MATRIX WITH COUNTERS
GENERATE THE PRODUCT BY SUMMING THE TWO ROWS WITH A CARRY LOOKAHEAD ADDER
A•B

Figure 1. Flow Chart for Multiplication Using Dadda's Method

*The notation [X] indicates the largest integer not greater than X, i.e., the Entier or Floor function.

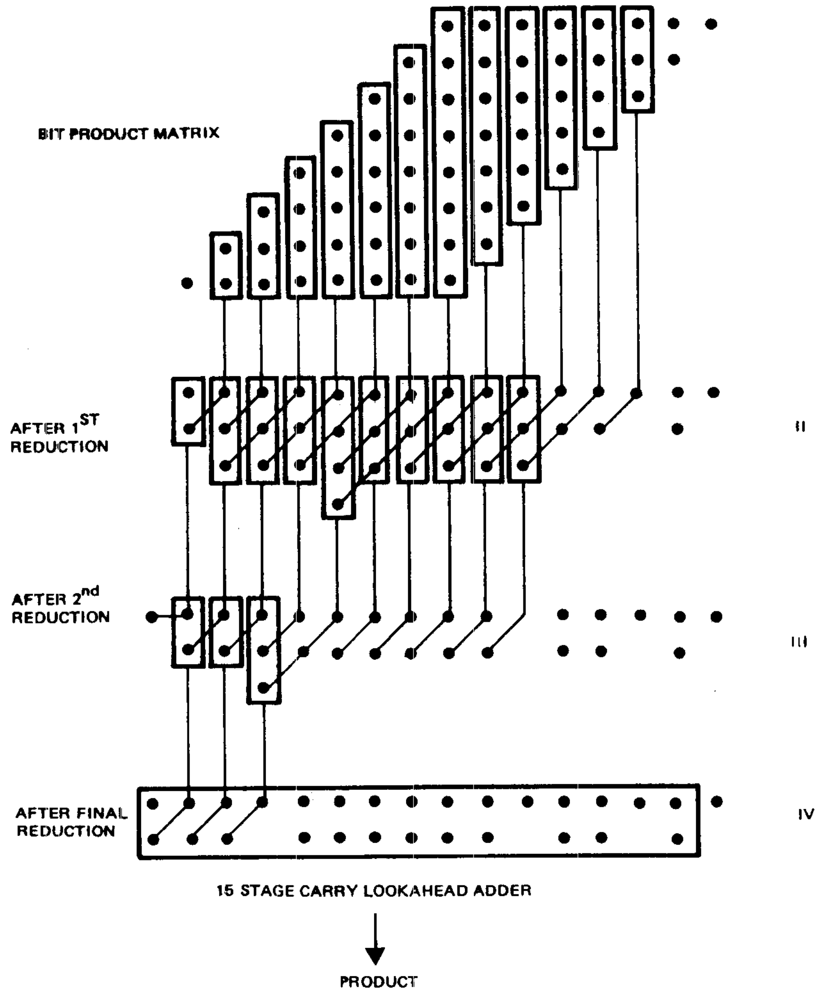

Figure 2. Reduction of Bit Product Matrix to an Equivalent Two Row Matrix

4) which go to columns 4, 5, and 6 respectively, of matrix II. The second matrix is reduced similarly, yielding matrix III, which is reduced to yield matrix IV which has two rows. A 15 bit wide carry lookahead adder completes the multiplication process.

The approach up to this point assumes the use of counters of all sizes between 3 and i in the implementation of an i bit multiplier. Implementation of counters of arbitrarily large size have been considered elsewhere.[5] Counters with more than three inputs are generally constructed with a number of adder modules (full and half adders). However, it is more efficient to implement the multiplier directly with adder modules. A procedure similar to the one which was shown in Figure 2 is used. The difference is that if any column of a matrix contains more than three entries, a multiplicity of three input counters (full adders) and two input counters (half adders) is used to perform the reduction. Suppose a column has eight entries: two full adders and one half adder are used; as a result, the next matrix contains three entries in that column and three carries to the next most significant column. Dadda has developed a technique[3] which is a slight variation on this approach that achieves the minimum delay with minimum complexity. Instead of fully reducing each column in the initial reduction, only a partial reduction is performed. This scheme is illustrated for an 8 bit multiplier in Figure 3. The outputs of full and half adders are indicated by two dots connected with dashes and crossed dashes, respectively. The procedure involves four distinct steps:

1) Let $d_1 = 2$ and $d_j + (3d_{j-1}/2)$. Find the largest j such that at least one column of the bit matrix has more than d_j elements.

2) Use adder modules as required to achieve a reduced matrix with no column containing more than d_j elements. Note that only columns with more than d_j elements (or

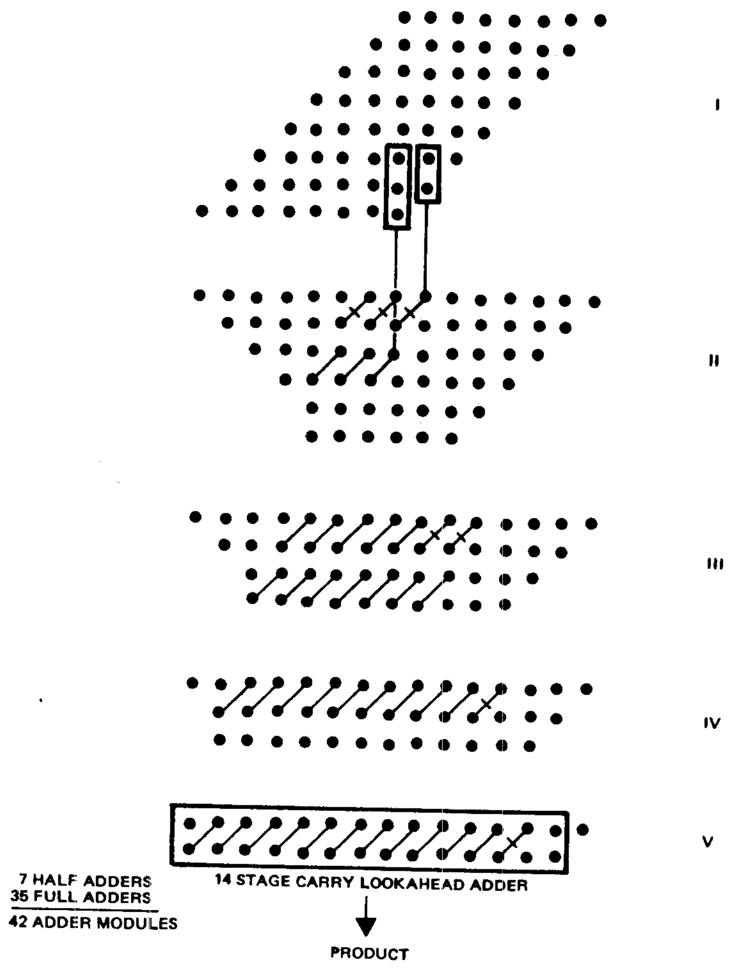

Figure 3. An 8 Bit Parallel Multiplier Implemented with Adder Modules

those which receive carries from less significant columns) are reduced.

3) Repeat step 2) with j = j - 1 until a matrix with only two rows is generated.

4) Use a carry lookahead adder to compute the product.

The total delay for computation of the product of two N bit operands includes the delay of the gates that form the bit product array, delay of the network of adder modules, and carry lookahead array. The delay of the network of adder modules is equal to the largest j such that

$$N > d_j$$

where

$$d_1 = 2 \text{ and } d_j = [3d_{j-1}/2]$$

The total delay is approximately

$$T_{mult} = T_{gate} + T_{add}\,[2 \log_2(N) - 2] + T_{cla} \quad (1)$$

The number of adder modules (most of which will be full adders) is

$$N_{add} = (N-1)\,(N-2)$$

These techniques are illustrated in Figure 3 which shows the implementation of an 8 bit parallel multiplier. A total of 42 adder modules (seven half adders and 35 full adders) are used to effect the reduction.

Conventional Complex Multiplier

The basic notion of the merged arithmetic element is that the four multipliers and two adders which comprise a complex multiplier can be coalesced into a simpler overall structure. To evaluate the improvement resulting from merged arithmetic consider the conventional approach.

The conventional approach for implementing a complex multiplier is to compute the real and imaginary components of the product (i.e., P real and P imag) directly:

$$P\ real = X\ real \cdot Y\ real - X\ imag \cdot Y\ imag$$

$$P\ imag = X\ real \cdot Y\ imag + X\ imag \cdot Y\ real$$

Clearly this requires four multipliers and two adders as shown in Figure 4. The time required to perform the complex multiplication, T_{cmult}, is simply

$$T_{cmult} = T_{mult} + T_{add}$$

where T_{mult} and T_{add} are the time to perform a real multiplication and addition, respectively. Using the multiplier delay from equation (1) and assuming that the real adder is realized with a carry lookahead adder:

$$T_{cmult} = T_{gate} + T_{add}\ [2 \log_2(N) - 2] + 2\ T_{cla} \quad (2)$$

Since each multiplier requires a carry lookahead adder, the total hardware requirement is

$$N_{gates} = 4\ N^2$$

$$N_{add} = 4\ (N-1)\ (N-2)$$

$$N_{cla} = 6 \text{ (2N bits wide).}$$

where the inputs are N bit numbers and a 2N + 1 bit result is produced at the multiplier outputs. A major fraction of the multiplier complexity is the need to provide six carry lookahead adders of 2N bits each.

Merged Arithmetic Complex Multiplier

Direct application of the merged arithmetic concept to the complex multiplier of Figure 4 is illustrated by considering a two-term multiplier/adder. Such a device forms the sum of two products directly. Two two-term multiplier/adders are used to realize the complex multiplier (one forms the real part of the product while the other forms the imaginary part). An implementation of the two-term multiplier/adder is shown in Figure 5. The two bit-product matrices are reduced through six stages of adders to produce a pair of numbers which are summed with a carry lookahead adder to produce the value of P imag.

This example (for 8 bit operands) requires the following components:

97 full adders
7 half adders
1 16 bit carry lookahead adder

Here the array of 128 2-input AND gates has not been included since both the conventional and merged arithmetic approaches require the same number of AND gates. The total delay for this circuit, T_{merged}, is given by

$$T_{merged} = T_{gate} + 6\ T_{add} + t_{cla} \quad (3)$$

Emprical experience indicates that the number of adder modules required to implement a two-term multiplier adder for N bit operands is:

$$N_{add} = N^2 - 3N$$

A conventional implementation requires two 8 bit multipliers and a 16 bit carry lookahead adder for a total complexity of:

70 full adders
14 half adders
2 15 bit carry lookahead adders
1 16 bit carry lookahead adder

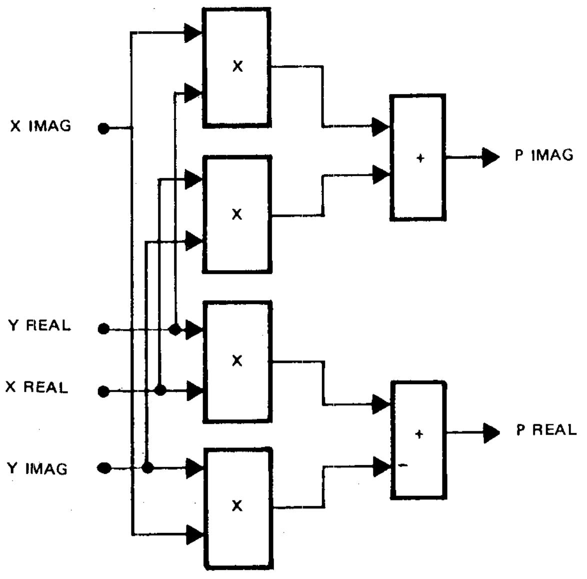

Figure 4. Complex Multiplier

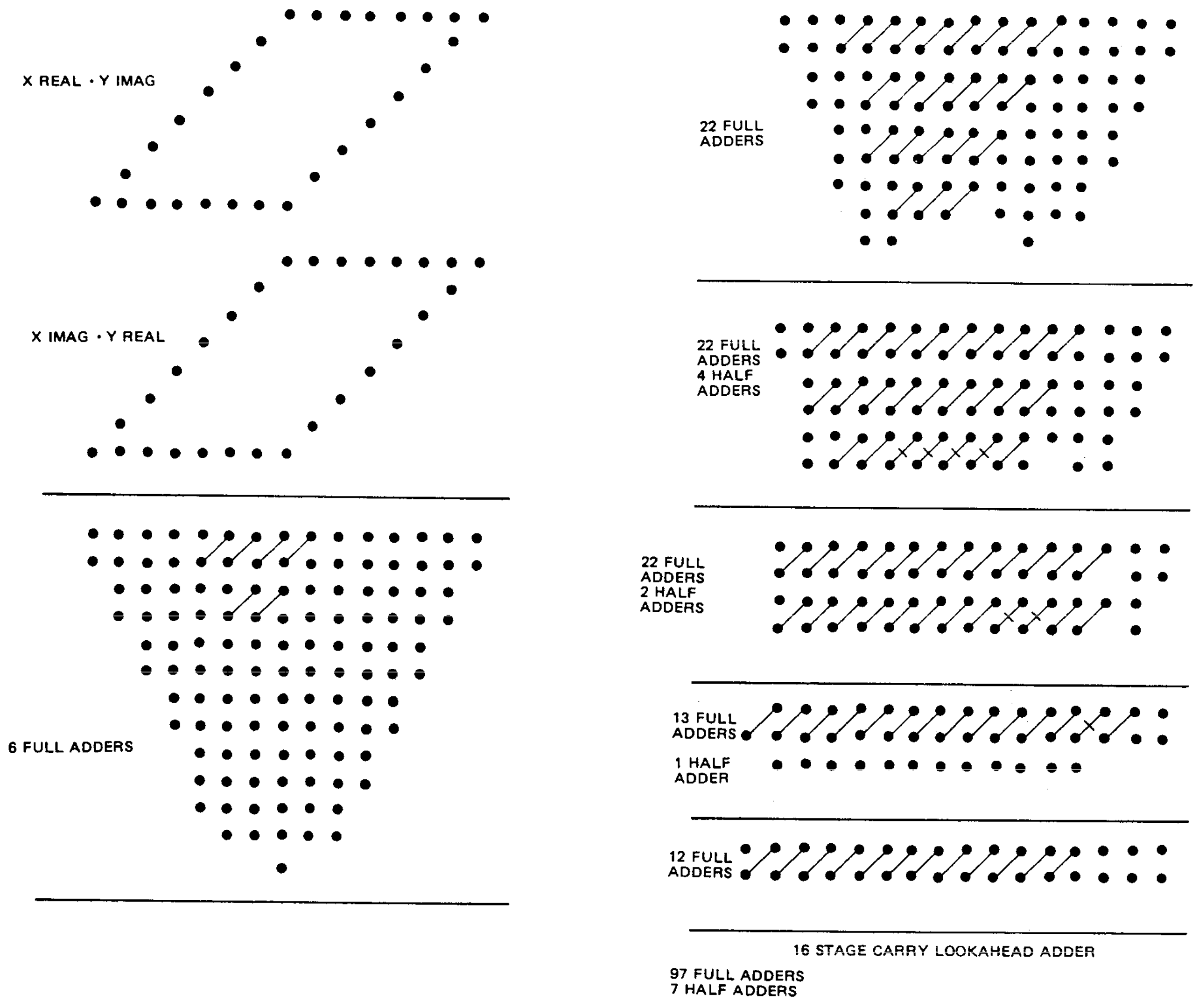

Figure 5. Two-Term Merged Multiplier/Adder

The total delay for this circuit, T_{conv}, is found from equation (2):

$$T_{conv} = T_{gate} + 4\ T_{add} + 2\ T_{cla} \qquad (4)$$

Comparing the two approaches, the merged approach requires 27 more full adders, seven fewer half adders, and two fewer 15 bit carry lookahead adders. To simplify the comparison, note that a 15 bit carry lookahead adder (realized with carry lookahead across 5 bit blocks) is implemented with 280 2-input gates, a full adder requires eight gates, and a half adder requires four gates.[6] The net savings for a two-term multiplier/adder is 372 gates. In terms of speed, the merged approach incurs two more full adder delays, but saves a 15 bit carry lookahead adder delay. The gate delay timing is equivalent since two levels of carry lookahead are required.

The extension of this approach to 12 x 12 arithmetic results in the comparison shown in Table 1. This comparison is based on element complexity figures from Table 2. Note that the merged approach replaces two 23 bit carry look-ahead adders (a total of 900 2-input gates) with 32 adders (a total of 256 2-input gates). This savings of 644 2-input gates for each multiplier/adder reduces the total complexity of the complex multiplier by over 1000 gates.

Table 1. 12 Bit Two-Term Multiplier/ Adder Complexity

FUNCTION	CONVENTIONAL		MERGED	
	USAGE	GATE COUNT	USAGE	GATE COUNT
FULL ADDER	220	1760	252	2016
23 BIT CARRY LOOKAHEAD ADDER	2	900		
24 BIT CARRY LOOKAHEAD ADDER	1	485	1	485
TOTAL GATE COUNT		3145		2501

Table 2. Element Complexity

FUNCTION	WIDTH	2-INPUT GATE COUNT
FULL ADDER	1 BIT	8
CARRY LOOKAHEAD ADDER	23 BITS	450
CARRY LOOKAHEAD ADDER	24 BITS	485

In fact, the complexity is reduced enough that a complete complex multiplier becomes feasible as a single very large scale integrated (VLSI) circuit. Initial assessment suggests that use of the triple diffused silicon VLSI process will result in a multiplier operating at a 10 MHz rate. As finer geometries are employed over the next few years, it should be possible to increase the speed to the 30 MHz level, with ultimate speeds on the order of 50 to 100 MHz.

The merged arithmetic concept offers a significant advantage in the implementation of a single-chip complex multiplier as only two carry lookahead adders are required compared to six carry lookahead adders for the conventional approach. Since large carry lookahead adders are difficult to realize on single VLSI circuits, the complex multiplier becomes practical only with the advent of merged arithmetic.

Although the circuits shown here are for positive numbers only, correction terms may be added to the bit product matrix[7] to accommodate two's complement operands.

Conclusion

The merged arithmetic concept which is similar in many respects to the quasi serial inner product computer,[8] facilities the development of a complex multiplier, which is a key function for signal processing system. As has been shown by examples, the complexity of complex multipliers is reduced sufficiently through the merged arithmetic approach that realization on single VLSI circuits is now practical.

References

1. B. Gold and C.M. Radar, Digital Processing of Signals (New York: McGraw-Hill Book Co., 1969) Chapter 5.

2. C.S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Trans. Electronic Computers, Vol. EC-13, 1964, pp. 14-17.

3. L. Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, Vol. 34, 1965, pp. 349-356 (Reprinted in Ref. 9).

4. A. Habibi and P. Wintz, "Fast Multipliers," IEEE Trans. Computers, (Short Notes), Vol. C-19, 1970, pp. 153-157.

5. E.E. Swartzlander, Jr., "Parallel Counters," IEEE Trans. Computers, Vol. C-22, 1973, pp. 1021-1024.

6. O.L. MacSorley, "High-Speed Arithmetic in Binary Computers," Proc. IRE, Vol. 49, 1961, pp. 67-91. (Reprinted in Ref. 9.)

7. R. Baugh and A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," IEEE Trans. Computers, Vol. C-22, 1973, pp. 1045-1047.

8. E. Swartzlander, Jr., B.K. Gilbert, and I.S. Reed, "Inner Product Computers," IEEE Trans. Computers, Vol. C-27, 1978, pp. 21-31.

9. E. Swartzlander, Jr., Computer Design Development: Principal Papers, (Rochelle Park, New Jersey: Hayden Book Co., Inc., 1976).