Algorithms and Structures for Reconfigurable Multiplication Units

Simon D. Haynes*, António B. Ferrari**, Peter Y. Cheung*
*Department of Electrical and Electronic Engineering, Imperial College, Exhibition Road, London, SW7 2AZ, England
** Departamento de Electrónica e Telecomunicações, Universidade de Aveiro, 3810 Aveiro, Portugal
email: s.d.haynes@ic.ac.uk - ferrari@inesca.pt – p.cheung@ic.ac.uk

Abstract

This paper evaluates alternative multiplication algorithms and parallel multiplier structures for the design of expandable multiplication units to be incorporated into hardware components for reconfigurable computing. It shows how the well-known methodologies used for fixed-length parallel multipliers can be extended to deal with expandable units. A novel design for such units is shown to offer a significant speed advantage over existing schemes.

1. Introduction

Traditionally arithmetic units of digital computers had to deal with one or at most two different data representation formats, and the same applied to most application-specific processors. The increasing importance that new media applications – image and audio processing, 3D graphics, MPEG video playback, videoconferencing, etc. - have been gaining, led to the extension of microprocessor instruction sets with multimedia-oriented instructions. At present the dominant processor architectures, Intel's Pentium [19], Sun's Ultrasparc [21], HP's PA 8000 [17], and lately PowerPC, all feature multimedia extensions.

Although the multimedia instruction subsets differ from one architecture to the other, they have in common the support for parallel short length arithmetic, the so-called "subword parallelism" [17], where binary arithmetic operations are performed simultaneously over multiple shortlength operand pairs. Hence the need to incorporate on such processors ALUs capable of operating both on word-long operands, typically 32 or 64-bit long, and on several short operands in parallel. The length of the short operands is typically the 8-bit data formats used in graphics and video processing, 16-bit in audio applications and to keep full precision when processing 8-bit data, and 32-bits for some 3D graphics algorithms.

On the other hand, in the domain of field-programmable gate arrays their increase in complexity of to tens and even hundreds of thousands of equivalent gates on a single chip, makes possible their use as the basic building block of complete systems on a chip. An array of general-purpose logic cells, the structure of current FPGAs, is however quite inefficient in handling the requirements of most systems. As the real estate on the chips continues to grow, the inclusion of specialized functional blocks becomes viable without affecting the application-generic nature of the component.

Fast arithmetic figures high on the list of requirements, being a critical performance parameter in a number of application domains where the use of FPGA-based systems has been investigated. Suppliers have somehow started to address the problem by providing FPGAs with some resources dedicated to speed up addition. However, configuring existing FPGAs to perform multiplication results in slow and expensive solutions. Hence the embedding of dedicated multiplier blocks into the FPGA structure appears as an attractive proposition, capable of offering good performance with an efficient use of silicon area. To keep however the FPGA structure application-generic, these new blocks should be able to handle both signed and unsigned multiplication and to accommodate the different operand wordlengths typical of the various application domains.

Hence a 8X8 bits multiplier is a sensible choice for the basic block. It has however to be expandable, such that longer wordlength multipliers can be built simply by connecting basic blocks, and to be able to deal with both two's complement and unsigned operands. These are also the requirements for the multiplication units of the current microprocessor architectures that feature multiply instructions in their multimedia extensions, as is the case with Pentium MMX and Ultrasparc VIS. Hence the topic of this paper is of interest to the apparently disjoint domains of reconfigurable systems hardware and arithmetic units for general-purpose computation.

This paper proceeds with a review of multiplication algorithms and parallel multiplier structures, followed by a novel approach to the design of expandable multi-mode multipliers. Finally some results, obtained in the context of the design of reconfigurable systems, are presented.

2. Multiplication algorithms and structures

2.1. Multiplication Algorithms

Schemes to achieve fast multiplication times have been pursued since the early days of electronic computers. They have addressed both the multiplication of unsigned binary and two's complement numbers.

Two fundamental questions have been addressed – on the one hand how to improve on the basic add-shift algorithm, the binary analogue of manual multiplication, and on the other hand how to deal with the multiplication of two's complement numbers. Two classes of algorithms have evolved, one class grouping solutions derived from the basic add-shift algorithm, optimized to deal with two's complement operands [1], along with unsigned multiplication for which it has been originally designed [16], the other class including solutions based on multiplier recoding, originally proposed by A. Booth [3] for two's complement multiplication and that has since been extended to deal with unsigned operands [18], [8]. These are generally known as Modified Booth, Overlapped Multiple-bit Scanning or Higher-Radix multiplication algorithms.

Table 1 indicates the number of partial products generated by the radix r algorithm, together with the minimum number of bits necessary for the representation of the partial products to handle unsigned, two's complement and mixed mode multiplication.

No. of partial products	$(n/\log_2 r) + 1$
Length of partial products	$m + \log_2 r$
Multiplicand multiples	$-r/2, \ldots, r/2$

Table 1 – Radix r multiplication parameters
n – multiplier length; m – multiplicand length

Only the radix-2, the original Booth's algorithm, and radix-4, known as 2-bits-at-a-time Modified Booth's algorithm, or just Modified Booth's algorithm, do not require multiples of the multiplicand not obtainable by simple shifts. Hence the radix-4 algorithm is by far the most widely used, since it halves the number of partial products generated for only a small increase in complexity of the partial product generation logic.

2.2. Parallel multiplier structures

In parallel multipliers the partial products are generated simultaneously and the product is calculated using a combinatorial scheme, hence the designation "non-iterative multipliers" that distinguish such architectures from sequential ones, where registers are used to store partial results and the combinatorial part is used iteratively.

The addition of the partial products could be done using multiple standard adders. Since adders reduce the two operands at their inputs to one output, a maximally-parallel tree of adders scheme can be used (a "Par-Mult" structure [22]), to achieve multiplication times of $\log_2(\Delta_{CPA})$, where Δ_{CPA} is the delay of a carry-propagate adder with the same number of bits as the partial products. The speed of such a structure is limited by the propagation of the carries, the major factor contributing to Δ_{CPA}. Hence parallel multiplier structures are based usually based on a different, and cheaper, approach, where the set of generated partial products is first reduced to two numbers whose sum produces the final result, leading to the structure depicted in Fig.1.

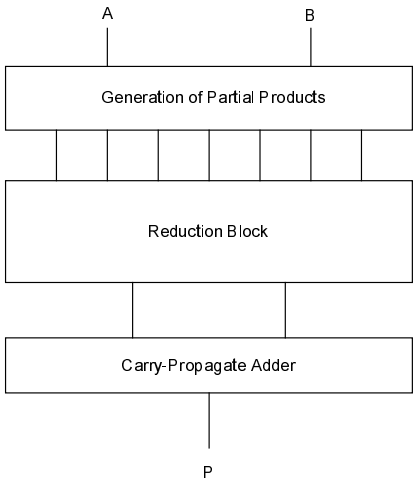

Fig.1 – Parallel multiplier block structure

The design of the reduction block depends on the choice of the reduction strategy and the type of components to be used.

The set of partial products can be viewed as a bit-matrix, where all bits in a column have the same binary weight. The reduction process is then a way of producing the binary count of the number of bits in each column, the procedure being iterated until a matrix with columns of height two is obtained. Hence the components to be used are called counters [6], since they produce the binary count of the number of ones at their inputs. If the counters operate on single columns, i.e. if their inputs have identical weight, they are called *column-compressors*, *unary-to-binary converters* [4] or (m,n)-counters, where m is the number of inputs and n the number of outputs. If they are capable of producing a weighted count, i.e. if they admit at their inputs bits from different columns, they are known as *generalized counters* [20]. Full-adders, i.e. (3,2)-counters, and half-adders, i.e. (2,2)-counters, are the simplest counters. The

use of high-order counters looked attractive when the scale of integration did not allow for single-chip fully parallel multipliers, each counter being then realized as a single chip, and allowing for a reduction in the number of components in the design and hence faster operation through less off-chip delays. However such counters, when implemented with gates, are basically cascades of full and half-adders, offering no significant reduction in the number of gates and making the layout more complex because of the heterogeneity of the structure. Hence in this paper FAs and HAs are the only components to be considered for the reduction block in Fig.1.

No. of P.P.s (NP)	No. of reduction stages
3	1
4	2
5 <= NP <= 6	3
7 <= NP <= 9	4
10 <= NP <= 13	5
14 <= NP <= 19	6
20 <= NP <= 28	7
29 <= NP <= 42	8
43 <= NP <= 63	9

Table 3 – Minimum number of carry-save adder stages as a function of the number of partial products

Two main reduction strategies are available: parallel (or tree) reduction and iterative (linear) reduction. Parallel reduction strategies have been proposed in [23] and [6]. Following the nomenclature proposed in [1], parallel reduction can be done either on a word-basis ("Wallace multipliers") or on a column-basis ("Dadda multipliers"). The minimum number of reduction stages as a function of the number of partial products to be reduced (NP) is indicated in Table 3 when just (3,2) and (2,2) counters are used. This number is $O(\log_{3/2} NP)$, NP being the number of partial products. Both Wallace and Dadda multipliers achieve the reduction in the minimum number of stages and are in that sense optimal.

In a Wallace structure [23] partial products are grouped into sets of three. Within each set FAs and HAs are employed to reduce the 3 partial products to two words. Extra rows, i.e. those that are not part of a three partial product set, are transferred to the next stage without modification, to be reduced in a later stage. The procedure is iterated until a 2-row bit matrix is obtained. Silicon implementations of Wallace multipliers have been reported, one notably using the Modified Booth algorithm [9].

Dadda's scheme [6] views the partial products generated as a bit matrix, rather than as a set of multiple-bit entities. Partial product bits are taken individually and the reduction is viewed as a sequence of column-compression operations whose aim is to obtain a matrix with columns of height 2, hence the designation "column-compression multipliers" given to these structures. Different reduction procedures have been proposed for column-compression multipliers, all of them requiring the minimum number of reduction stages and an almost identical number of gates. Dadda's procedure [6] is to use at each stage in the reduction the minimum number of FAs and HAs required to compress the matrix columns to the maximum height that can be reduced in one less stage (as given by Table 3). Hence, for instance, if the initial column height is 8, requiring 4 reduction stages, the output of the first reduction stage is a matrix with a maximum column height of 6, which can then be reduced in 3 further stages. The resulting structure is highly irregular – partial product bits have to be routed in a highly irregular pattern and some counter interconnections have to cross several reduction stages. However some VLSI implementations claiming reasonable area efficiency have been reported (e.g. [5]).

An alternative column-reduction strategy, leading to the so-called Reduced Area (RA) multiplier [2], is to apply Wallace procedure on a bit, rather than a word, basis. Hence at each stage (3,2) counters are used to achieve the maximum compression possible. If as a result column-height still exceeds the maximum that can be reduced in one less stage, (2,2)-counters are used. A (2,2)-counter is also used at the rightmost column containing 2 bits, in order that the final output of the reduction process requires a minimum length carry-propagate adder. The advantage of such a scheme is to minimize the number of bits passing between successive stages in the reduction, hence reducing the interconnection overhead, while using the same number of counters as Dadda's scheme. The number of latches required in highly pipelined implementations is also minimized in this scheme.

Other schemes aiming at improving the mapping to silicon have been proposed [24], in what became known as "Windsor multipliers" [4]. The criteria used in order to minimize silicon area are:

1. To distribute the adders as evenly as possible among the reduction stages in order to achieve a rectangular form factor.
2. To minimize the number of cross-stage interconnections
3. To reduce the maximum length of interconnections (favour nearest-neighbour interconnections).

Different structures result depending on which criteria, 1 or 2, is given priority, but in both cases the number of counters is the same as for the previous two schemes. That number has been shown to be minimum [10], whereas Wallace multipliers a larger number. One of the schemes leads also to a shorter carry-propagate adder, similarly to [2].

A question that must be addressed is the use of these tree-reduction schemes when the set of partial products is generated by a Booth class algorithm. Then the partial products generated are signed two's complement numbers and hence their summation must be done mod(2^{m+n}). Given that the reduction is being done in a maximally parallel way, the conventional sign-extension scheme used to prevent the need for double-length addition does not work, and the sign-extension bits must figure explicitly in the bit matrix. In order to get around the problem a strategy similar to the one used in Baugh-Wooley to deal with the partial product sign bits can be used – the partial products are stripped of their sign bits, and these are collected in a single word whose two's complement is then calculated. In order to simplify the logic this extra word can be made to correspond to the situation where all partial products are negative, a +1 being added at the most significant end of all non-negative partial products [7]. This is in fact equivalent to generate (NP/2+2), instead of (NP/2+1) partial products.

Linear reduction translates into an array-like structure as first proposed in [18] (albeit for an iterative scheme). In these structures delay is O(NP), i.e. a linear function of the number of partial products, (NP-2) reduction stages being required. The regularity of the structure leads to very compact chip layouts, making it by large the most used in actual designs.

The sign-extension problem is also easier to deal with. For these reasons, array schemes have so far been the almost universal choice in actual chip designs. The resulting structure is called a *carry-save array*.

3. Expandable Multiplier Array Blocks

To perform multiple-length multiplication using nXn multipliers a number of extra adders is required, as illustrated in Fig.2 for 2nX2n multiplication. The use of Programmable Additive Multiplier Modules (PAM), where each module performs the function

$P = A*B + C + D$

has been proposed in [16]. The module design used is a Baugh-Wooley design modified to perform not only two's complement but also unsigned and mixed mode multiplication, with the final carry-propagate addition performed by a ripple-carry adder. For MXN multiplication, where:

$M = p\,X\,n$ *and* $N = q\,X\,n,$

the overall delay Δ_T is:

$\Delta_T = [(p + q) - 1]\,\Delta_{PAM}$

Δ_{PAM} being the delay of the nXn PAM block.

However, if the multiplier blocks are integrated on a single chip, it becomes possible to merge the outputs of the reduction stages of each block, instead of adding the short length results. In such a scheme the inputs to the reduction stage of each block are not only the short-length partial products but also the outputs of the reduction stages of preceding blocks.

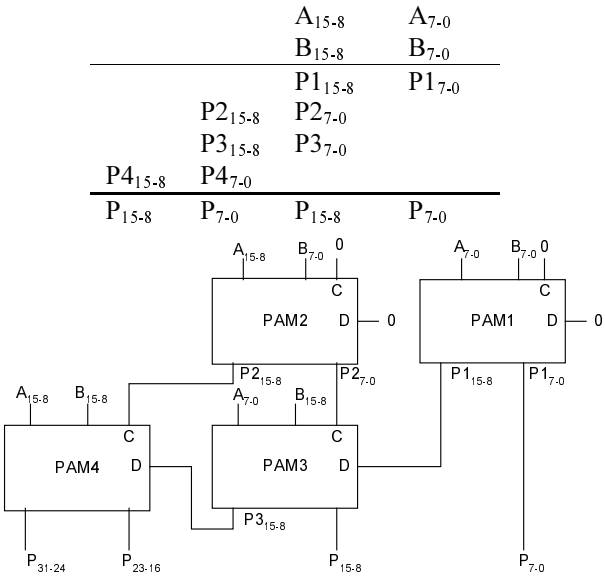

Fig. 2 – Doing 16X16 multiplication with 8X8 multipliers (4 PAM modules, P1 – P4 required)

Fig.3 illustrates the scheme for 16X16 multiplication based on 8X8 multiplier blocks. The partial products have been produced by Baugh-Wooley's algorithm. Fig.4 represents the resulting bit matrix to be reduced by each block. Figures 5 and 6 do the same as figures 3 and 4 but for the case when the radix-4 multiplication algorithm is used. Overall delay is now: $\Delta_T = (p - 1)\,\Delta_R$, where Δ_R being the delay of the reduction block of a nXn multiplier.

Fig.3 – 16X16 multiplier built from 8X8 Baugh-Wooley blocks.
** - block o/ps that are fed to the i/ps of connected blocks*
! – sign-extensions of the o/ps of intermediate blocks

For the reasons already explained 8X8 has been chosen as the dimension for the basic multiplier block. Hence the parallel and iterative reduction of the bit matrices depicted in figures 4 and 6 must be compared in terms of performance and numbers of components (FAs and HAs) required. Tables 4 and 5 summarize the results. Since a main factor in determining the achievable speed with a reduction scheme is the number of reduction stages, the data shows that the use of parallel reduction on 8X8 expandable multiplier blocks can provide at most a 20% increase over an iterative scheme when partial products are generated by the radix-4 Modified Booth algorithm. The potential nominal speed advantage of parallel over iterative reduction when Baugh-Wooley's is used is almost double that at 37.5%. Since an identical number of reduction stages (5) is required for the radix-4 Modified Booth array structure and the Baugh-Wooley tree structure, we can conclude that the nominal speed of these two alternative solutions is similar.

It must be stressed however that these comparisons are only based on logic delays, ignoring connection delays, which are longer in tree than in array structures.

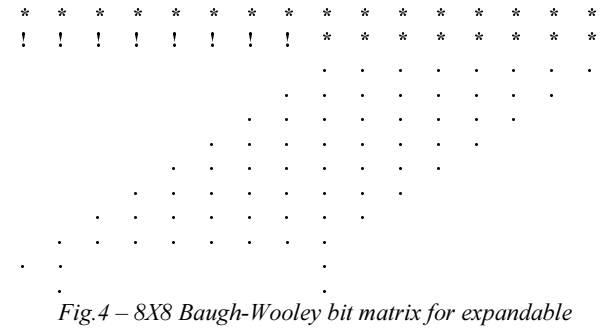

Fig.4 – 8X8 Baugh-Wooley bit matrix for expandable multipliers

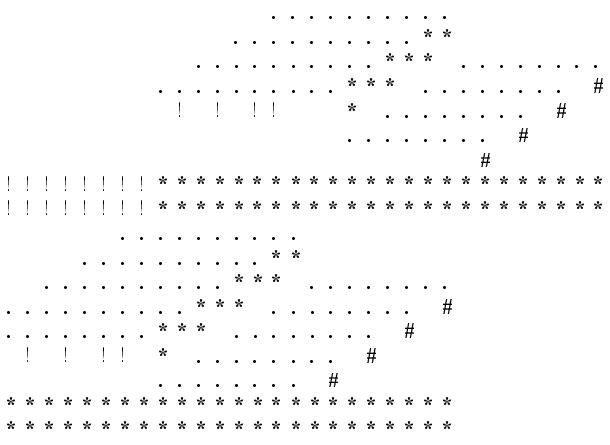

Fig.5 – 16X16 multiplier built from 8X8 Modified Booth blocks using sign-extension correction bits
 # - "hot 1s"

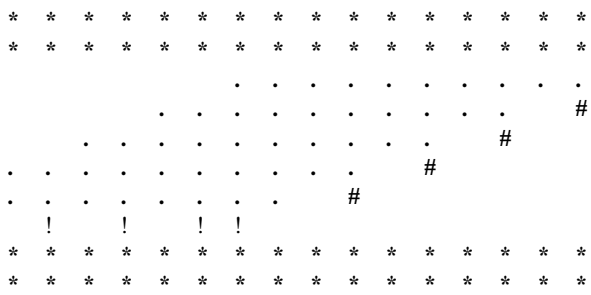

Fig.6 - 8X8 radix-4 M.Booth partial product matrix for expandable multipliers

	Tree	Array
Baugh-Wooley	5	8
Radix-4 M.B.	4	5

Table 4 – No. of Reduction stages for 8X8 expandable multiplier blocks

	Tree	Array
Baugh-Wooley	65 FAs + 6 HAs	64 FAs
Radix-4 M.B.	54 FAs + 6 HAs	48 FAs

Table 5 – No. of 1-bit full and half adders in the reduction stages of 8X8 expandable multiplier blocks

In what concerns the number of components required in the reduction stage, the fact that in the iterative (array) structure it is possible to sign-extend the words, dispensing the need to explicitly include the sign-extension bits in the original matrix to be reduced (i.e. the **!** bits in the figures), causes the array schemes to require less logic. The economy is bigger for the radix-4 M.B. array (roughly 15%, assuming that a FA requires around twice the area of a HA).

The design of a expandable multiplier block based on Baugh-Wooley's algorithm has been reported in [14] and [15], for a block of dimensions 4X4.

Fig.7 shows that cascading radix-4 Modified-Booth array blocks leads to a quite straightforward design. The logic blocks to be bypassed are shown shaded. The connections between blocks are also represented. The multiplier decoders are not shown. Only the decoders of the blocks dealing with the most significant part of the partial products are active; the others should be disabled.

From an i/o perspective only some extra control bits indicating the relative position for each multiplier block (most significant end or not, top, middle or bottom of the reduction array) are required.

Given that the top carry-save adder row has some of its inputs free, it is possible to use the block as a multiplier-accumulator performing the function

P = A*B + C C - n-bit operand

When calculating an inner product, if A and B are integers, C is the least significant half of the output and attention must be paid that intermediate results do not overflow. However when A and B are two's complement

numbers the most significant half of the result can be fed into the last row of the carry-save array that is not used when doing two's complement multiplication. Hence a full-precision accumulation can be performed with the array of Fig.7.

4. Conclusions

A new methodology for the design of expandable multiplier blocks has been presented. It was shown that this methodology leads to designs enjoying a significant speed advantage over previous schemes. Alternative algorithms and structures were evaluated for the design of 8X8 blocks, considered to be the most adequate dimension to build variable-length multiplication units for reconfigurable systems. A design using the radix-4 Modified Booth's algorithm and an array reduction structure was shown to hold significant speed advantages over the alternatives. Such a design offers also the advantage of being able to execute multiply-accumulate operations at no extra hardware or speed cost.

References

[1] – A.R.Baugh, B.A.Wooley, "A two's complement parallel array multiplication algorithm", *IEEE Transactions on Computers*, vol.C-22, no.12, December 1973, pp.1045-1047

[2] - K.C.Bircken staff, M.Schulte, E.E.Swartzlander, "Reduced Area Multipliers", Proc. ASAP'93, pp.478-489

[3] - A.D. Booth, "A signed binary multiplication technique", *Quarterly J. Mechan. Appl. Math.*, vol.IV, part 2, 1951, pp.236-240

[4] - P.R.Capello, K.Steiglitz, "A VLSI layout for a pipelined Dadda multiplier", *ACM Transactions on Computer Systems*, vol.1, no. 2, May 1983, pp.157-174

[5] - D.G.Crawley, G.A.J.Amaratunga, "8X8 bit pipelined Dadda multiplier in CMOS", *IEEE Proceedings on Part G*, vol.135, 1983, pp.231-240

[6] - L.Dadda, "Some schemes for parallel multipliers", *Alta Frequenza*, vol.34, pp.349-356, May 1965

[7] – J.Fadavi-Ardekani, "MxN Booth Encoded Multiplier Generator using Optimized Wallace Trees", *IEEE Transactions on VLSI*, vol.1, no.2, June 1993, pp.120-125

[8] - A.B. Ferrari Almeida, "A unified approach to the selection of parallel multiplier algorithms and structures in a LSI/VLSI environment", Ph.D. Thesis, Brunel University, 1982

[9] - A.E.Gamal et al., "A CMOS 32b Wallace Tree Multiplier-Accumulator", *Proceedings ISSCC 86*, p.194-…

[10] – A.Habibbi, P.A.Wintz, "Fast Multipliers", *IEEE Transactions on Computers*, vol.C-19, 1970, pp.153-157

[14] - S.D.Haynes, P.Y.K.Cheung, "Configurable multiplier blocks for embedding in an FPGAs", *Electronic Letters*, vol.34, no.7, 2nd April 1998, pp.638-639

[15] - S.D.Haynes, P.Y.K.Cheung, "A reconfigurable multiplier array for video image processing tasks, suitable for embedding in an FPGA structure", *Proceedings of FCCM'98*, IEEE Computer Society Press

[16] - K.Hwang, Computer Arithmetic – principles, architecture and design, John Wiley, 1979

[17]– R.B.Lee, "Subword Parallelism with MAX-2", IEEE Micro, Aug.96, pp.51-59

[18] - O.L.MacSorley, "High-speed arithmetic in binary computers", *Proceedings of the IRE*, January 1961, pp.67-91

[19] – A.Peleg, S.Wilkie, U.Reiser, "Intel MMX for Multimedia PCs", CACM, v.40, no.1, Jan.97, pp.25-38

[20] - W.J.Stenzel, W.J.Kubitz, G.H.Garcia, "Compact high-speed parallel multiplication scheme", *IEEE Transactions on Computers*, vol.C-29, no.10, October 1977, pp.948-957

[21] – M.Tremblay, J.M.O'Connor, V.Narayanan, L.He, "VIS Speeds New Media Processing", ", IEEE Micro, Aug.96, pp.10-20

[22] – J.Vuillemin, "A very fast multiplication algorithm for VLSI implementation", *Integration, the VLSI Journal*, 1, 1983, pp.39-52

[23] - C.S.Wallace, "A suggestion for a fast multiplier", *IEEE Transactions on Electronic Computers*, vol.EC-13, Feb. 1964, pp.14-17

[24] - Z.Wang,G.A.Julien,W.C.Miller, "A new design technique for column compression multipliers",*IEEE Transactions on Computers*, vol.44, no.8, Aug. 1995, pp.962-970.

Fig. 7 – Reduction part of a 16X16 built from a 8X8 Radix-4 Modified Booth Expandable Multiplier Block