A Comparative Study of Design and Analysis of 4 Bit Multipliers.

S. Subbaraman and Pravinkumar G. Parate

Abstract: **Recently, several experimental systems based on programmable logic have been designed and implemented which are programmed using a hardware design methodology. One necessary component of the software environment will be a library of standard macrocells corresponding to commonly used arithmetic and logical operations. In this paper Array multiplier is designed specially for programmable logic. This multiplier is cellular, highly pipelined and uses only of local interconnections. In the later part of this paper exposure to Booth multiplier and Wallace tree multiplier also has been given which is one of the reduction techniques for multipliers. The design is particularly carried out for a 4-bit multiplier**

***Index Terms*: adders, algorithm, array, binary arithmetic, Boolean algebra, logic design, circuit simulation, delay estimation, field programmable gate array, multiplication**

I. INTRODUCTION

Today, complex circuits are described in high-level description languages, like VHDL or Verilog, and synthesized to gate-level. A core operation in actual circuits, especially in digital signal processing such as Filtering, Modulation, or Video Processing or Neural Networks or Satellite Communication or Graphics or Control systems etc, is multiplication. Often, the computational performance of a DSP system is limited by its multiplication performance. This paper presents fundamental of some multiplication algorithm including signed and unsigned multiplication and there implementation details at CMOS level and the results thereof. Hardware multiplier implementation will have better speed than implementing the same using sequential statements in any higher level language. Traditionally shift and add algorithm has been implemented to design however this is not suitable for VLSI implementation and also from delay point of view. Some of the important algorithm proposed in literature for VLSI implimentable fast multiplication is Booth multiplier, array multiplier and Wallace tree multiplier. This paper presents the fundamental technical aspects behind these approaches, the details of implementation at CMOS level and the simulation results thereof. HDL to GDS flow of Mentor Graphics ASIC tool has been used to implement these multipliers.

II. MULTIPLICATION OPERATION

The most basic form of multiplication consists of forming the product of two unsigned binary numbers. This can be accomplished through the traditional technique thought in primary school, simplified to base 2.

(m×n) bit multiplication can be viewed as forming n partial product of m bits each, and then summing appropriately shifted partial products to produce an (m+n) bit result P. Binary multiplication is equivalent to a logical AND operation. Therefore, generating a partial product consist of[1] logical ANDing of the appropriate bit of multiplier and multiplicand. In the multiplication algorithm shown in Fig. 3 all n^2 combinations of the bits representing the input operands A and B are ANDed together. Perhaps the most obvious method of producing these partial products is an array of n^2 AND gates [7], as shown in Fig. 1. This circuit is cellular and may be implemented using only local interconnections. In this representation, however, the circuit is combinational. All n^2 partial products are generated in parallel. Another approach is to produce combinations of a_i and b_j systolically. By moving the bits of one operand across the other in the manner of a convolution, between 1 and n partial products are generated per cycle.

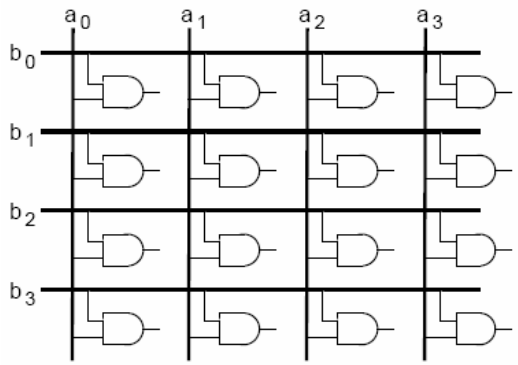

Fig. 1: Generation of Partial Products.

Prof Dr. (Mrs) S. Subbaraman is faculty & Dean(Academic) at Walchand College of Engineering, Sangli, India (416415)
(shailasubbaraman@yahoo.com)
Pravinkumar G Parate is ME Student at Walchand College of Engineering, Sangli, India 416415. (pgparate@rediffmail.com

Let A and B be the operands with m and n bits respectively. Using shift and add type of approach the product P of these two operands can be represented as shown in (1).

Larger multiplication can be more conveniently illustrated using dot diagram [1]. Fig. 2 shows a dot diagram for a simple 8×8 multiplier. Each dot represents a placeholder for a single bit that can be a 0 or 1. The partial product is represented by a horizontal boxed row of dots, shifted according to their weight. The multiplier bits used to generate the partial product are shown on the right.

$$A=[\sum_{j=0}^{m-1} a_j\, 2^J\ ] \qquad B=[\sum_{i=0}^{n-1} b_i\, 2^i\ ]$$

$$P=[\sum_{j=0}^{m-1} a_j\, 2^j\ ][\sum_{i=0}^{n-1} b_i\, 2^i\ ]$$

$$=\sum_{j=0}^{m-1}\ \sum_{i=0}^{n-1} a_j\, b_i\, 2^{i+j} \qquad (1)$$

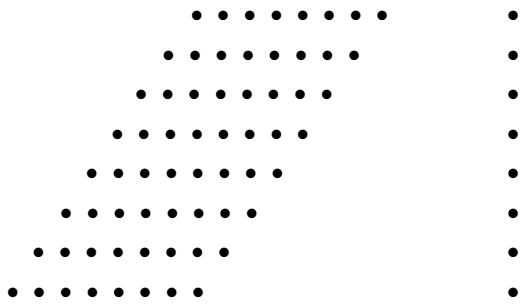

Fig 2- Dot Diagram

There are number of techniques that can be used to perform multiplication. In general, the choice is based upon factors such as latency, throughput, area, and design complexity. An obvious approach is to use an (m+1) bit carry propagate adder (CPA) to add the first two partial product, then other CPA to add the third partial product to the running sum, and so forth. Such an approach requires (n-1) CPAs and is slow even if a faster CPA is employed. More efficient parallel approach uses some sort of array or tree of full adders to sum partial products. Array multiplier, Booth Multiplier and Wallace Tree multipliers are some of the standard approaches to have hardware implementation of binary multiplier which are suitable for VLSI implementation at CMOS level. Beginning with a simple array for unsigned multipliers, and then modify the array to handle signed 2's compliment number using the Baugh-Wooley algorithm. The number of partial products to sum can be reduced using Booth encoding and the number of logic levels require to perform the summation can be reduced with Wallace tree. Unfortunately, Wallace tree are complex to layout and have long irregular wires, so hybrid array structure may be more attractive.

```
Multiplicand                 a3   a2   a1   a0
Multiplier                   b3   b2   b1   b0
-----------------------------------------------
                          b0a3 b0a2 b0a1 b0a0
                     b1a3 b1a2 b1a1 b1a0
                b2a3 b2a2 b2a1 b2a0
           b3a3 b3a2 b3a1 b3a0
-----------------------------------------------
      P7   P6   P5   P4   P3   P2   P1   P0
```

Fig 3 : Multiplication of two 4 bit numbers

III. MULTIPLICATION ALGORITHMS

Parallel multiplier uses combinational circuits only and thus operates much faster than serial multipliers. 4 such a algorithms are explained below.

A. Unsigned Array Multiplier

Implementation of array multipliers to multiply two 4 bit unsigned binary numbers $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$ is shown in Fig. 4. Here the basic building block of array is full adder block (FA) and the total numbers of FA required is 4 × 3 = 12. The FA block generates output as

(1) SUM = A ExOR B ExOR CI
(2) $C_O = A.B + B.C_I + A.C_I$.

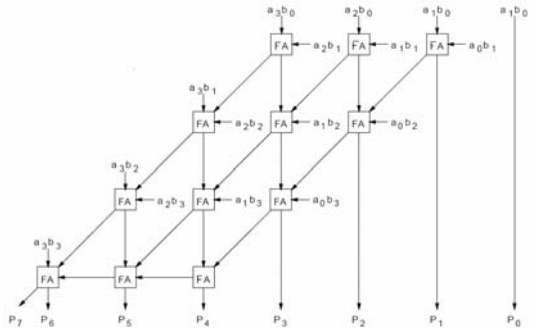

Fig 4: Array Multiplier

In general for two bit unsigned array multiplier the number of full adder required is n(n-1). Each AiBj is realized using an "AND" gate. Each output bit is computed by adding the appropriate AiBj in respective column and carry from previous column. To get the final 8 bit output we have to wait for maximum combinatorial delay of sum generation of two full adder blocks plus carry propagation time of 4 full adder block of last but one column. Thus it is fast multipliers but hardware complexity is high [4].

B. 2's Compliment Array Multiplication: Baugh-Wooley Algorithm.

Multiplication of a 2's Compliment number at first might seem more difficult because some partial product are negative and must be subtracted. Two signed n bits and m bit binary number can be represented as shown in (2) and (3) and the product P of these two signed number is represented in (5). In (5) two of the partial product have negative weight and thus should be subtracted rather than added.

Fig 5 shows the partial product that must be summed. The upper parallelogram represents the unsigned multiplication of all but the most significant bit of input. The next two pair of row is the inversion of the term to be subtracted. Each has implicit leading and trailing 0's, which are inverted to leading and trailing 1's. Extra ones must be added in the least significant column when taking 2's Compliment. The multiplier delay depends on number of partial product to be summed. The modified Baugh-Wooley multiplier reduces this number of partial product by recomputing the sum of the constant 1's and pushing some of the terms upward to extra column. Fig. 6 shows such an arrangement.

$$Y = -y_{m-1}2^{m-1} + \sum_{j=0}^{m-2} y_j \ 2^j \qquad (2)$$

$$X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i \ 2^i \qquad (3)$$

$$P = [-y_{m-1}2^{m-1} + \sum_{j=0}^{m-2} y_j \ 2^j][-x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i \ 2^i]$$

$$= \sum_{i=0}^{n-2} \sum_{j=0}^{m-2} x_i y_j \ 2^{i+j} + x_{n-1}y_{m-1}2^{m+n-2}$$

$$-[\sum_{i=0}^{n-2} x_i \ y_{m-1}2^{i+m-1} + \sum_{j=0}^{m-2} y_j \ x_{n-1}2^{i+n-1}] \qquad (4)$$

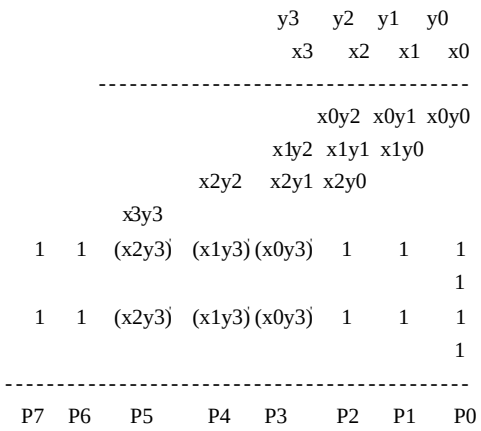

Fig.5. Partial Product for 2's Compliment Multiplication (terms in bracket indicate compliment)

Implementation of signed multiplier two signed 5 bit binary is shown in Fig. 7. Here two set of partial products $(A_4B_3)(A_4B_2)(A_4B_1)(A_4B_0)$ & $(A_3B_4)(A_2B_4)(A_1B_4)(A_0B_4)$ are to be subtracted from other partial product. So in the last two rows there are 4 full subtractor cell in each row. Full subtractor cell subtracts from it's A input the other two input in B and C_I and computes difference (DIF) and output borrow (C_O) as per following logic.

(1) DIF=A xor B xor C_I
(2) C_O=A'B+A'.C_I+B.C_I.

So these two partial products are input to B and CI terminal of FS block. Other partial product resultant sum is fed to the A input of the FS block. The addition is done by first 4 rows of FA blocks, each now containing three blocks. thus for signed multiplication of two 5 bit binary number, full adder blocks required is 3 × 4 = 12 numbers and fill subtractor blocks required are 2 × 4 = 8.

In general for signed multiplication of two n bit binary numbers full adder blocks required are (n-2)(n-1) and full subtractor block required is 2(n-1). To get the final 9 bit output we have to wait for maximum combinational delay of sum generation of three full adder blocks plus carry propagation time of 5 FA block of last and last but one column. The final carry output is ignored since it does not affect the result. So it is a fast multiplier but hardware complexity is also high.

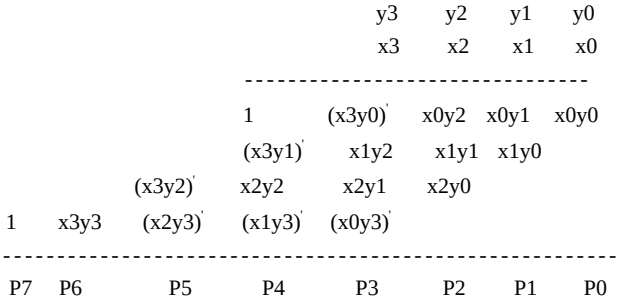

Fig. 6 : Simplified Partial Product for 2's Compliment Multiplication (terms in bracket indicate compliment)

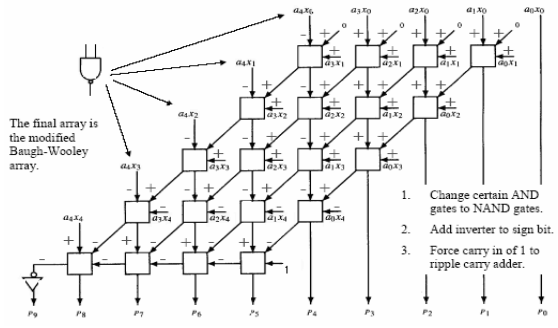

Fig.7. Baugh Wooley signed multiplier

C. Booth Multiplication

One way to speed up the multiplication is Booth encoding [3], which performs several steps of multiplication at once. Booth's algorithm takes advantage of the fact that an adder subtractor is nearly as fast and small as a simple adder [6][8]. In the elementary school algorithm, we shift the multiplicand X, then use one bit of multiplier Y if that shifted value is to be added into the partial product. The most common form of booth's algorithm looks at three bit of multiplier at a time to perform two stages of multiplication.

Consider once again the two's compliment representation of multiplier Y.

$$Y = -2^{n} Y_{n} + 2^{n-1} Y_{n-1} + 2^{n-2} Y_{n-2} + \ldots\ldots\ldots$$

Taking advantage of the fact that $2^a = 2^{a+1} - 2^a$ above equation can be written as

$$Y = 2^{n}\left(Y_{n-1} - Y_{n}\right) + 2^{n-1}\left(Y_{n-2} - Y_{n-1}\right) + 2^{n-2}\left(Y_{n-3} - Y_{n-2}\right) + \ldots\ldots\ldots$$

Now, extract the first two terms.

$$2^{n}\left(Y_{n-1} - Y_{n}\right) + 2^{n-1}\left(Y_{n-2} - Y_{n-1}\right)$$

Each term contributes to one step of the elementary school algorithm: the right hand term can be used to add x to partial product, while the left hand term can add 2x.(in fact, since y_{n-2} also appears in another term, no pair of terms exactly corresponds to a step in elementary school algorithm. But if we assume that the y bit to the right of the decimal point are 0 all the required terms are included in the multiplication.) if for example, $y_{n-1} = y_n$, the left hand term does not contribute to the partial product. By picking three bits of y at a time, we can determine whether to add or subtract x or 2x (shifted by proper amount, two bits per step) to the partial product. Each three bit value overlaps with its neighbour by 1 bit. Table I shows the contributing term for each three bit code from y.

TABLE I
CONTRIBUTING TERM FOR EACH 3 BIT CODE FROM Y

$Y_i Y_{i-1} Y_{i-2}$	Increment
000	0
001	X
010	X
011	2X
100	-2X
101	-X
110	-X
111	0

The guidelines for performing Booth Multiplication are given in Table 1. From the basics of Booth Multiplication it can be proved that the addition/subtraction operation can be skipped if the successive bits in the multiplicand are same. If 3 consecutive bits are same la LSB then addition/subtraction operation can be skipped. This fact is indicated in the above table.

Thus in most of the cases the delay associated with Booth Multiplication are smaller than that with Array Multiplier. However the performance of Booth Multiplier for delay is input data dependant. In the worst case the delay with booth multiplier is on per with Array Multiplier.

The delay is confidently reduced in Wallace Tree Multiplier which is explained in the following section.

D. Wallace Tree Multiplier

A fast process for multiplication of two numbers was developed by Wallace [5]. Using this method, a three step process is used to multiply two numbers; the bit products are formed, the bit product matrix is reduced to a two row matrix where sum of the row equals the sum of bit products, and the two resulting rows are summed with a fast adder to produce a final product.

Consider an example of multiplication of two four bit numbers. In the first step the partial products from MSB position are rearranged as shown in Fig. 8(b) and the partial product from the middle column are added using half adder as grouped in Fig. 8(b). in stage two the resultant arrangement with sum and carry bit from addition of first step being properly placed and bits in column two to five are added using half adder or full adder as applicable as shown in Fig. 8(c). This will results into the arrangement of bits in two rows which are again added using half adder or full adder as applicable as shown in Fig. 8(d). Thus there is a delay of only three stages as compared to six in array multiplier. The schematic of Wallace Tree Multiplier is shown in Fig. 8.

Table II shows the number of stages required for Wallace Tree Multiplication. If the multiplier has just one more bit than any of the "thresholds", one more stage is needed in the reduction process. For example, 5 stages are needed if n is equal to 13, but 6 stages are needed if n equals to 14.

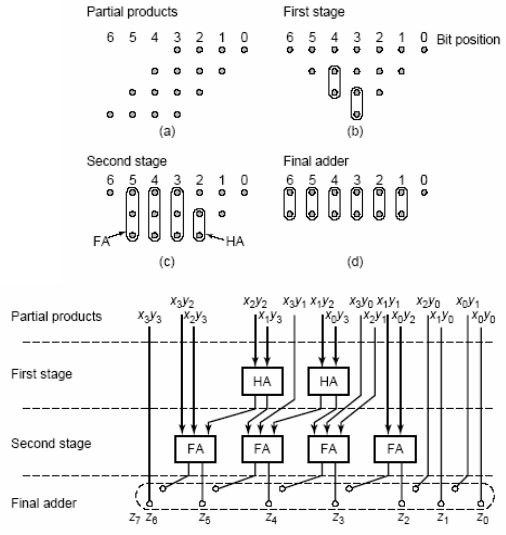

Fig. 8 Wallace Tree Multiplier

TABLE II
STAGES FOR WALLACE TREE MULTIPLIER

Number of bit In Multiplier	No of reduction stages
3	1
4	2
4<n<=6	3
6<n<=9	4
9<n<=13	5
13<n<=19	6
19<n<=28	7
28<n<=42	8
42<n<=63	9

IV. IMPLIMENTATION DETAILS

The above three multipliers are implemented in CMOS using HDL to GDS flow of Mentor Graphics ASIC tools. the design entry was using VHDL which was converted into Verilog and Synthesized using Leonardo Spectrum. The verilog file from synthesizer is inputted to IC Station to generate Layout and Netlist and simulated for functionality using ELDO simulator. The layouts generated by IC Station are shown in Fig. 9, 10, 11 and 12 for unsigned Array Multiplier, signed Array Multiplier, Booth Multiplier and Wallace Tree Multiplier respectively. All the tools used are from Mentor Graphics ASIC Suite. The results of simulation for each multiplier are explained in the next section.

V. RESULTS

Table III shows the simulation results of all the multipliers mentioned above w.r.t. Power at Vdd = 5V and Vdd = 3.3V, number of components required, memory consumed and delay. A power delay product has been computed for comparison.

TABLE III
SIMULATION RESULTS W.R..T. POWER, NUMBER OF COMPONENTS, MEMORY & DELAY

Multiplier	Power (nW)		No of components	Memory (Kbytes)	Delay	Power Delay Product
	Vdd =5V	Vdd = 3..3 V				
Unsigned Array Multiplier	4.447	1.857	490	2336	14.4 ns	64.03
Signed Array Multiplier	3.889	1.625	476	2315	14.4 ns	56.00
Booth Multiplier	11.42	4.723	1278	5265	16.8 ns	191.8
Wallace Tree Multiplier	5.019	2.146	562	2506	13.7 ns	68.76

VI. CONCLUSION

This article presents a basic multiplier algorithm for multiplication of 4 bit binary numbers which is suitable for CMOS implementation with partial product generation technique. From the results of simulation it can be concluded that Booth Multiplier is inferior in all respect and hence should be avoided. From power delay product Array Multiplier turns out to be better than Wallace Tree Multiplier. However Array Multiplier gives optimum power consumption as well as number of components required, but delay for this multiplier is larger than Wallace Tree Multiplier. Hence for low power requirement Array multiplier is suggested and for less delay requirement Wallace Tree Multiplier is suggested.

VII. REFERENCES

Books:

[1] Neil H.E. Weste, Devid Harris Ayan Banerjee, *Principles of CMOS VLSI Design* , third edition.

[2] Etienne Sicard, Sonia Delmas Bendhia, *Basic of CMOS Cell Designs*,. New Delhi: McGraw-Hill, , pp. 267-271.

[3] Wayne Wolf, *Modern VLSI Design, Systems on Silicon*. (2nd ed.), Pearson Education Asia, pp. 297-305

Papers from Conference Proceedings (Published):

[4] Jorn Stohmann Erich Barke, "A Universal Pezaris Array MultiplierGenerator for SRAM-Based FPGAs" IMS – Institute of Microelectronics System, University of Hanover Callinstr, 34, D-30167 Hanover, Genmany.

[5] Moises E. Robinson and Earl Swartzlander, Jr. *"A Reduction Schem to Optimize the Wallace Multiplier"* Department of Electrical and computer Engineering , University of Texas at Austin, Austin, USA.

[6] R. Lyon, *"2's Compliment Pipelined Multipliers"*, IEEE Transactions on Communication, April, 1976.

[7] Steven A. Guccione MARIO j. Gonzalez *"A Cellular Multiplier for Programmable Logic"* Computer Engineering Research Center, " Department of Electrical and computer Engineering , University of Texas at Austin, Austin, USA, febuary, 1994.

[8] Kai Wang, *"Global and Modular two's Compliment Array Multipliers"* IEEE Transactions on Computers, Vol- C-28, issue 4, Apr 79.

[9] Shivaling S. Mahant-Shetti, Poras T. Balsara, and Carl Lemonds " High Performance Low Power Array Multiplier Using Temporal Tilting" IEEE Transaction on VLSI systems, vol. 7, No 1, Mar 1999,

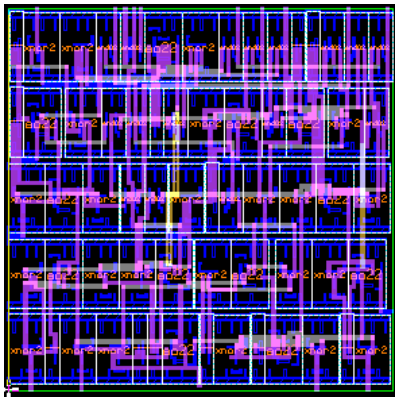

Fig. 9. Layout of Unsigned Array Multiplier

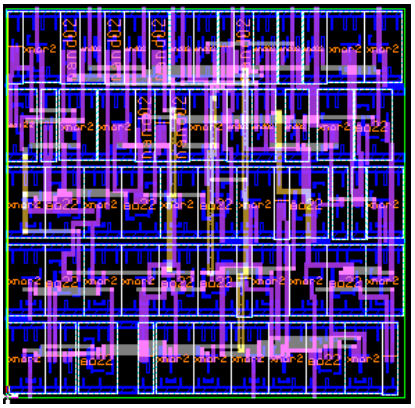

Fig. 10. Layout of Signed Array Multiplier

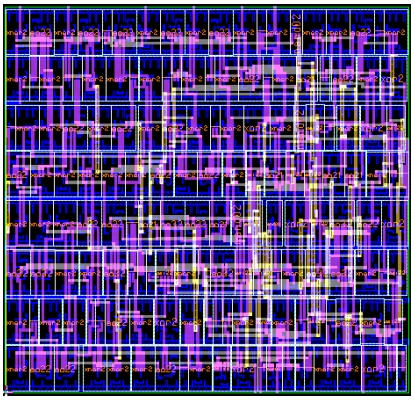

Fig. 11. Layout of Booth Multiplier

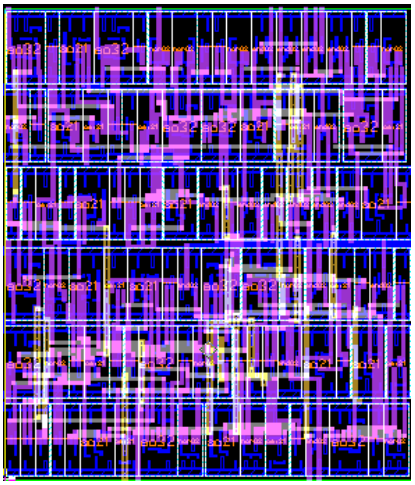

Fig. 12. Layout of Wallace Tree Multiplier

VIII. BIOGRAPHIES

Shaila Subbaraman received her M-Tech degree from IISc Bangalore in 1975 and Ph.D. from IIT Bombay in 1999. she worked in Semiconductor Device Manufacturing company from 1975 to 1989. Currently she is Professor in Department of Electronics in Walchand College of Engineering, Sangli. she has keen interest in the field of Microelectronics and VLSI Design

Pravinkumar G. Parate received the B.E. degree in Electronics Engineering in 2002 from BDCoE Sevagram (Wardha), Nagpur University, Nagpur, India and perusing the M.E. degree in Electronics Engineering from Walchand College of Engineering, Sangli, India. He has put in three years teaching experience and worked in RAICSIT, Wardha and G.H. Raisoni College of Engineering, Nagpur.