International Journal of Science Engineering and Advance Technology

Implementation of Area Efficient and High Bit Rate Serial-Serial Multiplier With On-the-Fly Accumulation by Asynchronous Counters

Suda Sriram
M.Tech, Embedded Systems.
Joggaih Institute of Tech. & Engg. College of Engineering.
Kalagampudi, West Godavari District.
Andhra Pradesh, India.
sriramsuda@gmail.com

Ch. Kanakalingeswara Rao
Assistant Professor in ECE dept.
Joggaih Institute of Tech. & Engg. College of Engineering. Kalagampudi, West Godavari District. Andhra Pradesh, India.
klr_mani786@yahoo.co.in

Abstract—**Traditional Serial-Serial multiplier addresses the high data sampling rate. It is effectively considered as the entire partial product matrix with n data sampling cycle for n×n multiplication function instead of 2n cycles in the conventional multipliers. This multiplication of partial products by considering two series inputs among which one is starting from LSB the other from MSB. Using this feed sequence and accumulation technique it takes only n cycle to complete the partial products. It achieves high bit sampling rate by replacing conventional full adder and highest 5:3 counters. Here asynchronous 1's counter is presented. This counter takes critical path is limited to only an AND gate and D flip-flops. Accumulation is integral part of serial multiplier design. 1's counter is used to count the number of ones at the end of the nth iteration in each counter produces. The implemented multipliers consist of a serial-serial data accumulator module and carry save adder that occupies less silicon area than the full carry save adder. In this paper we implemented model address for the 8bit 2's complement implementing the Baugh-wooley algorithm and unsigned multiplication implementing the architecture for 8×8 Serial-Serial unsigned multiplication.**

Keywords—Binary multiplication, on-the-fly accumulation, parallel multipliers, serial and serial-parallel multiplier.

I. INTRODUCTION

Multipliers are essential building blocks in VLSI circuits. Hardware implementation of a multiplication operation consists of three stages, specifically the generation of partial products (PPs), the reduction of partial products (PPs), and the final carry propagation addition. The partial products can be generated either in parallel or serially, depending on the target application and the availability of input data. The partial products are reduced by carry-save adders (CSAS)[5]using an array or tree structure. Carry propagation addition is inevitable when the number of partial products is reduced to two rows. This final adder can be a simple ripple carry adder (RCA) for low power or a carry look-ahead adder (CLA) for high speed. As the height of PP tree increases linearly with the word length of the multiplier, it aggravates the area, delay and power dissipation of the two subsequent stages.

Therefore, it is highly desirable to reduce the number of partial products before the CSA Stage. In the implemented method the partial product formation is revamped using an algorithm named as serial-serial algorithm which is explained in the following sections. The generated partial products are passed to a group of asynchronous 1's counters for accumulation. The counters will count the number of ones in the partial products which is used for addition. In the following sections an approach to the design of serial multiplier that is capable of processing input data without input buffering and with reduced total number of computational cycles.

A new approach to form the entire partial product matrix in just n sampling cycles for an N x N multiplication instead of at least 2n cycles in the conventional serial-serial multipliers. A new approach to serial accumulation of data by using asynchronous counters is suggested here which essentially count the number of 1s in respective input sequences. The counters effectively replace the full adders in the accumulation circuit. In this project the partial products are formed in serial manner and they are accumulated by the counters and finally the product can be obtained by adding the accumulated output by the ripple carry adder. Serial multipliers are popular for their low area and power. They are broadly classified into two categories, namely serial-serial and serial-parallel multiplier. In a serial-serial multiplier both the operands are loaded in a bit-serial fashion, reducing the data input pads. To reduce the number of computational cycles from 2n to n in an N

x N serial multiplier, several serial-parallel multipliers.

II. SERIAL MULTIPLIERS

In a serial-serial multiplier both the operands are loaded in a bit-serial fashion, reducing the data input pads to two serial multipliers are popular for their low area and power. Bit-serial processing can result in efficient communications, both within and between VLSI chips, because of the reduced number of interconnections required. Serial multiplier designs which are particularly suitable for applications where input data are sequentially presented .The operating speeds are determined mainly by the propagation delays along the critical path within the processing elements. A major advantage offered by bit serial processors is when the operands are available only one bit at a time, the processing speed can be improved using bit-serial arithmetic elements. The structures that use this approach can achieve moderate speeds with comparatively small area.

A. Implementation Stages for Serial- Serial Multiplier

This operation consists of three stages

i) The generation of partial products.
ii) The reduction of partial products.
iii) Final carry-propagation addition.

The partial products can be generated either in parallel or serially, depending on the target application and the availability of input data. The partial products are generally reduced by carry-save adders using an array or a tree structure. Carry propagation addition is inevitable when the number of partial products is reduced to two rows. This final adder can be a simple ripple carry adder for low power or a carry look-ahead adder for high speed. It is highly desirable to reduce the number of partial products before the carry-save adder's stage. The drawback is that and higher order compressors are slower and consumes more power than the full adders. An accumulator is an adder which successively adds the current input with the value stored in its internal register.

B.Serial Multiplier with Counter Based Accumulator

In this type of multipliers the partial products are formed in a serial manner. Instead of using full adders counters are used to count the number of 1"s in a column in this structure. This method provides power minimization from the fact that the counter output will not toggle unless a „1" is present at its input. The counter output is applied to a ripple carry adder to obtain the final product. By using this counter based accumulator the partial products are formed in N cycles for an NXN multiplication. A new approach to serial/parallel multiplier design by using parallel 1s counters to accumulate the binary partial product bits. The 1s in each column of the partial product matrix due to the serially input operands are accumulated using a serial D-flip flop (DFF) counter.

III. IMPLEMENTED SERIAL-SERIAL MULTIPLIER

Accumulation is an integral part of serial multiplier design. A typical accumulator is simply an added that successively adds the current input with the value stored in its internal register. Generally, the adder can be a simple RCA but the speed of accumulation is limited by the carry propagation chain. The accumulation can be speed up by using a CSA with two registers to store the intermediate sum and carry vectors, but a more complex fast vector merged adder is needed to add the final outputs of these registers. A new approach to serial accumulation of data by using asynchronous counters is suggested here which essentially count the number of 1's in respective input sequences.

A.8 Bit Word Length for Unsigned Multiplier

A new technique of generating the individual row of partial products by considering two serial inputs, one starting from the LSB and the other from MSB. It takes only n cycles to complete the entire partial product generation.

The product of two unsigned numbers X and Y can be written as

$$P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_i y_j 2^{i+j} \qquad \text{------ (1)}$$

Where xi and yj are the ith and jth bits of X and Y with bit 0 being the LSB.

Reversing the sequence of index I and rearranging the above equation can be written as

$$P = \Sigma_r PP_r \qquad \text{---------- (2)}$$

Where

$$PP_r^L = \{0 \quad r=0$$

$$\Sigma_k x_{n-k-1} . y_r . 2^{n+r-k-1}$$

$$r=1,2,...n-1\} \qquad PP_r^C = \{0$$

$$r=0$$

$$\Sigma x_{n-r-1} . y_r . 2^{n-1} \quad ,$$

$$r=1,2,...n-1\} \qquad PP_r^R = \{0$$

$$r=0$$

$$\Sigma k x_{n-r-1} . y_{r-k-1} . 2^{n-k-2},$$

$$r=1,2,....n-1\}$$

The partial product row PP rcan be generated in r cycles if X is fed from MSB(bitn-1) first and Y is fed from LSB first (bit 0), then in the r^{th}cycle ,PP_r^C is a partial product bit generated by the current input bits x_{n-r-1} and y_r,PP_r^L are partial product bits of the current input bit, y_r and each of the preceding input bits of X, i.e x_{n-k-1},for k=0,1,….r-1.and$PP_r^R$are the partial product bits of the inputbitxn-r-1,andeachofthe preceding input bits of Y, i.e Y_{r-k-1} for k=0,1,2,….r-1.By appropriately sequencing the input bits of X and Y into a shift register, one PP(PP_r)in each cycle can be generated. As a result P can be obtained in n cycles.

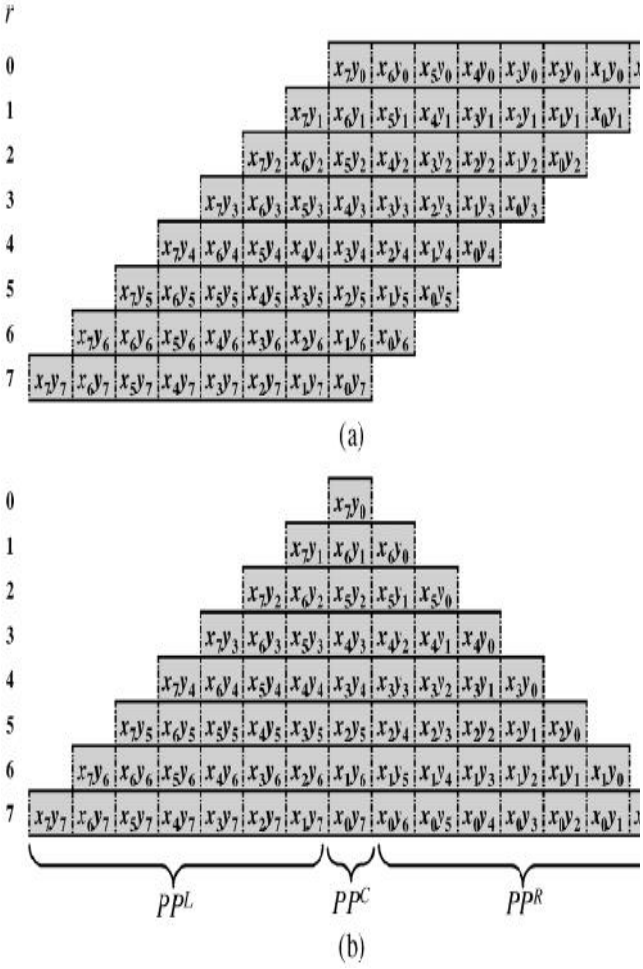

Figure.1 Partial product generation schemes for an 8 X 8 serial-serial multiplication :(a) conventional and (b) proposed

The PP generation of an 8×8 multiplier for two unsigned numbers X and Y. "Fig.1 (a)", shows the conventional partial product formation and "Fig.1 (b)", shows the generation sequence of the PPs. Row r generated in cycle r, for r=0,1……..n-1.The PPs in "Fig.2", are generated in such an unconventional way in order to facilitate their accumulation on-the-fly by the Implemented counter-based accumulation technique. A PP bit corresponding to the middle column of the PP is produced by the centered AND gate.

The latched outputs are wired to the correct FAs and HAs (half adders) according to the positional weights of the output bits produced by the counters. From "Fig.2", it is observed that the column height has been reduced from 8 to 4 and the final product, can be obtained with two stages of CSA tree and a final RCA. Similarly, for 16×16, 32×32 and 64× 64 multipliers the column heights are reduced logarithmically from 16, 32, and 64 to 5, 6, and 7, respectively. This drastic reduction in column height leads to a much simpler CSA tree, and hence reducing the overall hardware complexity and power consumption.

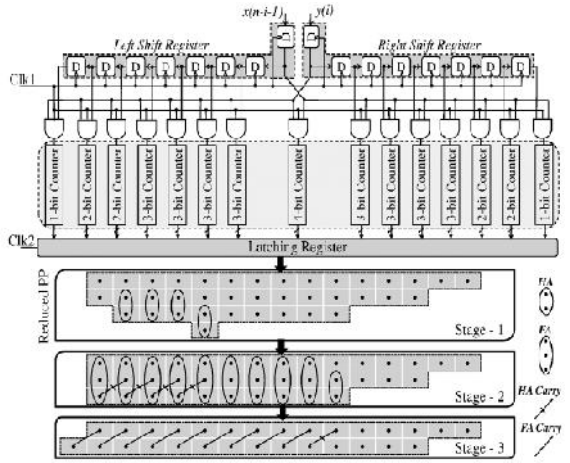

Figure.2 Implemented architecture for 8 × 8 serial-serial unsigned multiplication

The latching register between the counter and the adder stages not only makes it possible to pipeline the serial data accumulation and the CSA tree reduction, but also prevents the spurious transitions from propagating into the adder tree.

B.8 bit word length for 2's complement numbers

The Most digital systems operate on signed numbers commonly represented in 2's complement. 2's complement numbers using the Baugh–Wooley algorithm. The architecture of the Implemented 2's complement serial-serial multiplier is depicted below structure.

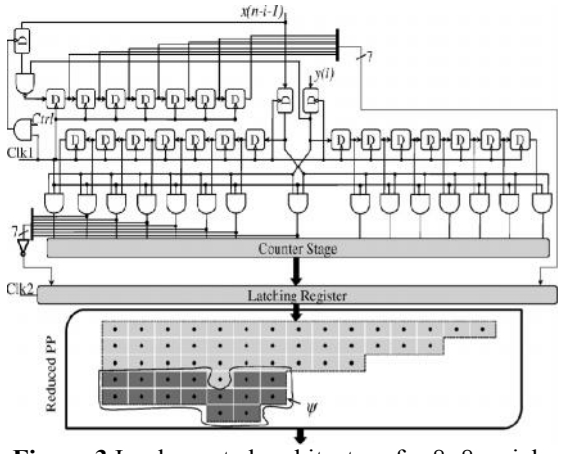

Figure.3 Implemented architecture for 8×8 serial-serial 2's complement multiplication

The addition of the term raises the height of CSA tree by only two bits regardless of the word length of the operands.

$$P^I = X^I Y^I = \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} x_i y_j 2^{i+j} + 2^{n-1}\overline{\sum_{i=0}^{n-2} x_i y_{n-1} 2^i} + \overline{\sum_{j=0}^{n-2} x_{n-1} y_i 2^j}$$

$$+ x_{n-1}y_{n-1}2^{2n-2} + 2^n - 2^{2n-1} \quad \text{-------3}$$

The multiplication of two unsigned numbers by the multiplier can be expressed as,

$$P^I = X\,Y = \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} x_i y_j 2^{i+j} + 2^{n-1}\overline{\sum_{i=0}^{n-2} x_i y_{n-1} 2^i} + \overline{\sum_{j=0}^{n-2} x_{n-1} y_i 2^j}$$

$$+ x_{n-1}y_{n-1}2^{2n-2} \quad \text{----------4}$$

The difference between (3) and (4) is given by

$$\Psi = P^I - P$$

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

$$= 2^{n-1}\sum_{i=0}^{n-2} (x_i y_{n-1} + x_i y_{n-1} - 2x_i y_{n-1})\, 2^i + \sum_{j=0}^{n-2} (x_{n-1} y_i + x_{n-1} y_j - 2x_{n-1} y_j)\, 2^j + 2^n - 2^{2n-1} \quad \text{---------- (5)}$$

As $x_i y_{n-1} + x_i y_{n-1} = 1$ and $x_{n-1}\overline{y_i + x_{n-1}} y_j = 1$, (5) can be simplified to

$$\Psi = 2^{n-1}\sum_{i=0}^{n-2} (1 - 2x_i y_{n-1})\, 2^i + \sum_{j=0}^{n-2} (1 - 2x_{n-1} y_j)\, 2^j + 2^n - 2^{2n-1} \quad \text{---(6)}$$

To extend the unsigned multiplier architecture for signed multiplication without introducing a high over-head,thedifferenceexpressedin6mustbesimplified.Since $\Sigma_i^{n-2} 2^i = \Sigma_j^{n-2} 2^j$,the following summation terms embedded in(6) can be simplified by the closed form expression of a geometric progression, i.e., $2^0 + 2^1 + 2^2 + \ldots + 2^{n-2} = 2^{n-1} - 1$.

$$2^{n-1}\sum_{i=0}^{n-2} 2^i + \sum_{j=0}^{n-2} 2^j = 2^{\Psi-1}(2^{n-1} - 1 + 2^{n-1} - 1)$$

$$= 2^{2n-1} - 2^n \quad \text{---------- (7)}$$

By substituting (7) into (6), the constant terms $2^{2n-1} - 2^n$ cancel out and the difference, Ψ is reduced to

$$\Psi = -2^n \sum_{i=0}^{n-2} x_i y_{n-1} 2^i + \sum_{j=0}^{n-2} x_{n-1} y_j 2^j \quad \text{----------(8)}$$

The difference Ψ of (8) is added to P so that the structure can be modified to obtain P^I, the product of the 2's complement multiplication

$$P^I = P + \Psi$$

$$= P - 2^n \sum_{i=0}^{n-2} x_i y_{n-1} 2^i + \sum_{j=0}^{n-2} x_{n-1} y_j 2^j \quad \text{---- ----(9)}$$

In the implemented PP generation method, Y_{n-1} arrives only in cycle n-1.The generation of $\Sigma_i^{n-2} X_i Y_{n-1} 2^i$ has to be delayed until cycle n-1.There main in g terms $\Sigma_j\, n^{-2}\, X_{n-1}\, Y_j\, 2^J$ can be computed during the initial n-1cycles. Hence, The difference can be corrected in the CSA tree .It is trivial that a n-1 bit shift register, a NAND gate and several FAs are required for adding Ψ.

IV. PERFORMANCE COMPARISON AND IMPLEMENTATION RESULTS

VHDL implementation results shows that Implemented multiplier by using Unsigned and Signed methods based on higher order column compressor by using Asynchronous Counters and final adder is Carry Propagation Adder and Carry Look ahead Adder for multiplication of Two n*n binary numbers is faster than multipliers compared to Array and Booth and Vedic multiplier[10] shown in Table I It also proves that as the number of bits increases to N, where N can be any number, the delay time is greatly reduced in Implemented Multiplier as compared to other multipliers. Implemented Multiplier has the advantages as over other

multipliers also for Delay and area [11] shown in Table II.

TABLE I
DELAY COMPARISON OF 8x8 MULTIPLIERS [10]

S.No	Name of the Multiplier		Delay(nsec)
1	Array Multiplier		47
2	Booth Multiplier		117
3	Vedic Multiplier		27
4	Implemented	Signed	4.36
		Unsigned	4.28

TABLE II
AREA AND DELAY COMPARISON OF 8x8 MULTIPLIERS [11]

S. No	Multiplier Algorithm for 8x8		NO of Slices/CLB'S	Cycles	Max Operating Frequency(MHZ)	Critical Path Delay (nsec)
1	Carry ripple Multiplier		60	2n	14.97	66.8
2	Wallace tree multiplier		79	2n	19.23	52.00
3	Braun Multiplier		79	2n	21.82	45.82
4	Pipelined constant coefficient Multiplier		26	2n	70.42	14.20
5	Pipelined Serial parallel with converters		21	2n	88.9	11.24
6	Shift and add Multiplier		33	2n	93.98	10.64
7	Pipelined Guild Multiplier		199	2n	98.75	10.12
8	Pipelined Carry Save Multiplier		187	2n	100.09	9.99
9	Implemented	Signed	177	n	227.99	4.38
		Unsigned	96	n	233.31	4.28

VHDL simulation

The VHDL simulation of the two multiplier is presented in this section. The VHDL code for an unsigned and signed multiplier using a fast carry propagation adder and fast carry look-ahead adder (CLA). The VHDL model has been developed using the Direct VHDL simulator. The multipliers use 8-bit values and simulation results of two multipliers are shown below waveforms. This code was synthesized using Xilinx Spartan 3E family. This chapter explains about the simulation of Area Efficient and A High Bit Rate Serial-Serial Multiplier with On-the-Fly Accumulation by Asynchronous Counters Synthesis report shown TABLE III.

Simulation results of 8x8 Unsigned Serial_Serial_Multiplier_CPA

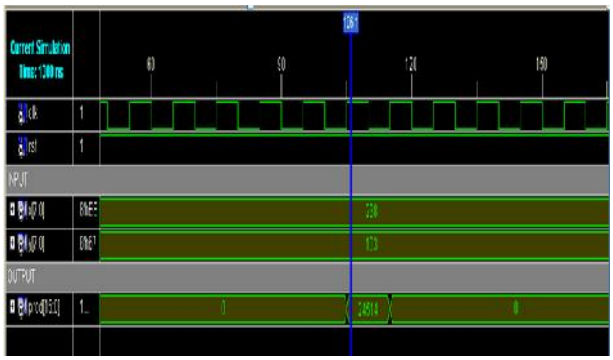

Figure.4 Unsigned 8x8 Serial_Serial_Multiplier_CPA

Simulation results of 8x8 Signed Serial_Serial_Multiplier_CPA

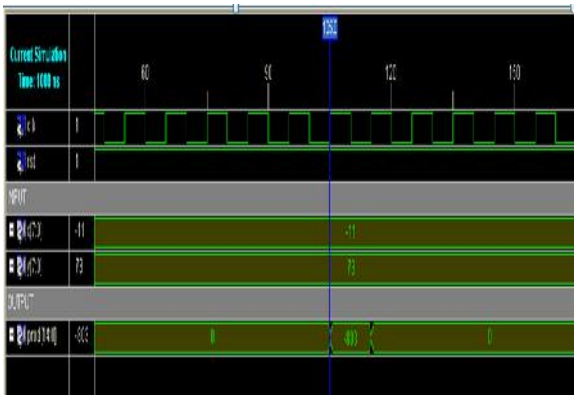

Figure.5 Signed 8x8 Serial_Serial_Multiplier_CPA

Simulation results of 8x8 Unsigned Serial_Serial_Multiplier_CLA

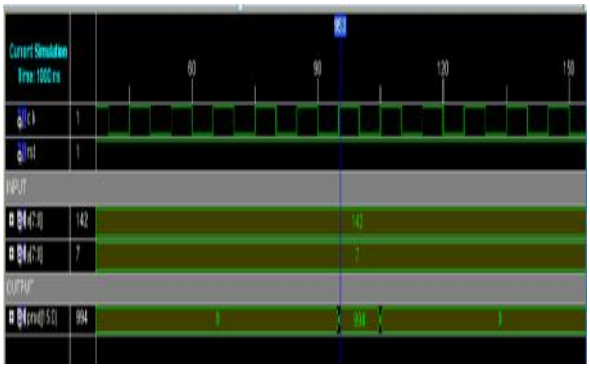

Figure.6 Unsigned 8x8 Serial_Serial_Multiplier_CLA

Simulation results of 8x8 Signed Serial_Serial_Multiplier_CLA

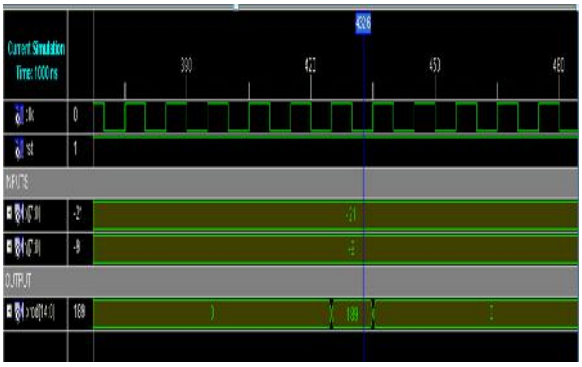

Figure.7 Signed 8x8 Serial_Serial_Multiplier_CLA

**Synthesis Report*

The Synthesis report for both an unsigned multiplier using a fast carry propagation adder and fast carry look-ahead adder and an signed multiplier using a fast carry propagation adder and fast carry look-ahead adder shown Table III.

TABLE III
AREA AND DELAY RESULTS OF IMPLEMENTED 8x8 MULTIPLIERS

Description	Serial_Serial_Multiplier Carry Propagation adder		Serial_Serial_Multiplier Carry look-ahead adder	
	No. of slices utilization out of 4656 (Area)	Critical Path Delay (nsec)	No. of slices utilization out of 4656 (Area)	Critical Path Delay (nsec)
Unsigned	96 (2%)	4.283	94 (2%)	4.283
Signed	177(3%)	4.368	179 (3%)	4.386

V. CONCLUSION

In this paper, design of multiplier by using carry propagate adder (CPA) and Carry look-ahead adder for addition of final partial product terms and by introducing Carry Save Adder (CSA) for reducing the partial product lines. Our implementation technique effectively forms the entire partial product matrix in just n sampling cycles for an n x n multiplication instead of at least 2n cycles in the conventional serial-serial multipliers. It achieves a high bit sampling rate by replacing conventional full adders and 5:3 counters with asynchronous 1's counters so that the critical path is limited to only an AND gate and a D flip-flop (DFF). The use of 1's counter to column compress the partial products preliminarily reduces the height of the partial product matrix from n to [log2 n] +1, resulting in a significant complexity reduction of the resultant adder tree. The multipliers presented in this paper were all modeled using VHDL (Very High Speed Integration Hardware Description Language) for 8-bit unsigned and signed data. The comparison is done on the basis performance parameters i.e. Area (2% & 3%) and Speed (4.283ns & 4.368ns) previously in the literature.

VI .SCOPE OF EXPANSION

The further extension of this 8 x 8 serial-serial unsigned multiplication and 8 x 8 serial-serial 2's compliment multiplication can be implemented by using 16 x 16 –bit, 32 x 32 -bit are more than but architecture includes an area overhead and design complexity.

REFERENCES

[1] Meher MR,Ching Chuen Jong; Chip Hong Cheng , "*A High bit rate Serial-Serial multiplier*",IEEE Trans.VLSI,vol.19,pp.1733-1745,Sept 2010

[2] Essam Elsayed and Hatem M. El-Boghdadi "*Area-Time Efficient Digit-Serial-Serial Two's Complement Multiplier*" Vol. 2, Special Issue 1, Part 2, February 2009.

[3] OHSANG KWON and KEVIN NOWKA "*A 16-Bit by 16-Bit MAC Design Using Fast 5:3 Compressor Cells*" Journal of VLSI Signal Processing 31, 77–89, 2009.

[4] P. Assady "*A New Multiplication Algorithm Using High-Speed Counters*" European Journal of Scientific Research ISSN 1450-216X Vol.26 No.3 (2009), pp.362-368.

[5] Harpreet Singh Dhillon and Abhijit Mitra "*A Reduced-Bit Multiplication Algorithm for Digital Arithmetic*" International Journal of Computational and Mathematical Sciences 2:2 2008

[6] Ravi Nirlakalla, Thota Subba Rao, Talari Jayachandra Prasad, "*Performance Evaluation of*

High Speed Compressors for High Speed Multipliers" Vol. 8, No. 3, November 2006, 293-306

[7] Nibouche.O, Bouridarie.A, and Nibouche.M (2001), '*New architectures for serial-serial multiplication*' in Proc. IEEE Conf. Circuits Syst. (ISCAS), Sydney, Australia, vol. 2, pp. 705–708.

[8] H. K. Mecklai and A. L.Webb, "*Compact buffer design for serial I/O*"U.S. Patent 6 167 109, Dec. 26, 2000.

[9] Sungwook Kim and Gerald E. Sobelman, "*Digit-Serial Multiplier Design Using Skew-Tolerant Domino Circuits*" University of Minnesota Minneapolis, MN 55454, USA (2000).

[10] Sumit Vaidya and Deepak Dandekar "*Delay-Power Performance Comparison Of Multipliers In Vlsi Circuit Design*" International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4,
(2010)

[11] J.Senthil Kumar G.Lakshminarayanan "*Design and Implementation of FPGA based Fast Multipliers with Optimum Placement and Routing using Structure Organizer*" Department of ECE, Regional Engineering College, Tiruchirapalli.

[12] 0. Nibouche.O, A. Bouridarie, M. Nibouche.O "*New Architectures for Serial-Serial Multiplication*" 0-7803-6685-9101/$10.0002000 IEEE.

[13] A AGGOUN, A ASHUR and M K IBRAHIM "*Area-Time Efficient Serial-Serial Multipliers*" ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland.

[14] Paolo Ienne and Marc A. Viredaz "*Bit-Serial Multipliers and Squarers*" IEEE Transactions on Computers. vol. 43, no. 12, December 1994.

[15] R. GNANASEKARAN "*On a Bit-Serial Input and Bit-Serial Output Multiplier*" IEEE Transactions on Computers, VOL. C-32, NO. 9, September 1983.