A Fast Serial Multiplier Design using Ripple Counters

Vishnupriya.A, AlthafKhan.J, Gopalakrishnan.R

Abstract: ***Multipliers are the fundamental components of many digital systems. Low power and high speed multiplier circuits are highly demanded. It is not possible to achieve both the criteria simultaneously.Therefore, there exists some trade-off between speed and power consumption in the design of a binary multiplier.The partial product (PP) formation and reduction of the partial product tree height in serial multipliers is our primary concern. In existing CSAS architecture of partial product formation is carried out in 2n clock cycles, where n is the number of operand bits. Here, the critical path is along the full adders and AND gates. The disadvantage of this method is increased delay as it involves 2n computational clock cycles. This disadvantage is overcome by using the concept of data accumulation by ripple counters. Column compression technique is used for the reduction of partial product tree height. Unlike the CSAS architecture, the critical path in this technique is found only along the AND gates. The partial product formation is done in n clock cycles instead of 2n clock cycles and hence delay is reduced. Moreover, the counters change states only when input is '1', which leads to low switching power.The final product is computed by the Ripple carry adder (RCA) in both the above architectures. RCA is replaced by KSA for improved performance. The average connection delay in the multiplier architecture due to KSA is reduced and is expressed in terms of nano seconds.***

Keywords— Binary multiplication, Partial product, Ripple counters, Serial multiplier.

I. INTRODUCTION

Multipliers are generally the slowest element in the system and the performance of the system or a chip is generally determined by the performance the multiplier. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors. The array multiplier [2] originates from the multiplication parallelogram. Each stage of the parallel adders should receive some partial product inputs. In the multiplier based on Wallace tree [12], the multiplicand-multiples are summed up in parallel by means of a tree of carry save adders. Booth–Mac Sorely algorithm is simply called the Booth algorithm [7].

Manuscript published on 30 February 2013.
***** Correspondence Author (s)
Vishnupriya.A, Electronics and Communication Engineering, Avinashilingam Institute for Home Science and Higher Education for Women University, Coimbatore , India.
AlthafKhan.J, Electronics and Communication Engineering, V.S.B Engineering College (Anna University Chennai), Coimbatore, India,.
GopalaKrishnan.R, Electrical and Electronics Engineering, Muthayammal Engineering College (Anna University Chennai), Namakkal, India.

The Booth algorithm is implemented into two steps: Booth encoding and Booth selecting. The main disadvantage of Booth multiplier is the complexity of the circuit to generate a partial product bit in the Booth encoding.A binary multiplier [16] is build using binary adders. It is an electronic hardware device used in digital electronics to perform rapid multiplication.

Parallel multipliers [11], [19] are popular for their high speed of operation. The cost of hardware is the drawback in long word length multiplication using parallel multipliers. It also consumes more power in such a case. In public key cryptography like RSA encryption and decryption, integer multiplications of 1024 bits are specific [1]. A full precision bit-serial multiplier was introduced by Strader et al [2] for unsigned numbers. The multiplier accepts binary operands supplied in a serial fashion, least significant bits first. A full-precision scheme for 2's complement numbers has been presented in [3]. This is the first bit-serial multiplier that directly handles the negative weight of the most significant bit (MSB) in 2's complement representation. This method needs only n cells for an n * n bit multiplication but it introduces an XOR gate in the critical path, which results in a more complicated overall design.A serial-parallel multiplier loads one operand in a bit-serial manner and the other is always available for parallel operation.2n clock cycles are required to complete the process of multiplication. n clock cycles are used for addition of partial products and another n clock cycles for the propagation of carry [4]. The delay due to storage elements is eliminated.

A serial multiplier and a squarer with no latency cycles are presented in [5]. Algorithms for serial squarer's and serial/serial multipliers were derived in [6] and Error propagation in two's complement operation has been investigated to minimize the number of D flip-flops that need to be cleared between operations.

Serial multipliers [12]-[17] 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 [17] multiplier both the operands are loaded in a bit-serial fashion, reducing the data input pads to two. The Modified Baugh-Wooley Two's Complement Signed Multiplier [8]-[10] is the best known algorithm for signed multiplication because it maximizes the regularity of the multiplier logic and allows all the partial products to have positive sign bits. According to the Baugh-Wooley approach, an efficient method of adding extra entries to the bit matrix is suggested to avoid having to deal with the negatively weighted bits in the partial product matrix.

Unlike the CSAS architecture, the critical path is found only along the AND gates [17]. The partial product formation is done in n clock cycles instead of 2n clock cycles and hence delay is reduced. Thus, the numbers of computational cycles are reduced in the proposed method. Moreover, the counters change states only when input is ‘1’, which leads to low switching power.

Y.Kim and L.S.Kim [18] proposed a carry select adder which is implemented by using single ripple adder instead of using dual ripple carry adder so that it could reduce the area with negligible speed penalty by add one circuit based on multiplier. The Brent-Kung adder [19] tree computes prefixes for 2-bit groups. The drawback of this adder is its low performance.Kogge-stone adder is called as parallel prefix adder. It is the common design for high-performance adders in industry. It is considered as the fastest adder as the computation time taken by this adder is O(log n) time[21].The logic-level implementation of the basic cells used in parallel-prefix adders is described in [22].

The rest of this paper is organized as follows Section II describes the background information about CSAS multiplier. Section III reviews the description of the counter – based serial multiplier. Section IV explains the implementation of Kogge-stone adder in the Counter – based multiplier. Section V illustrates the implementation and simulation results that have been arrived. Section VI gives the conclusion of the work done so far.

II. THE CSAS MULTIPLIER

Data accumulation is carried out using carry save adder (CSA). These multipliers are based on carry-save add-and-shift (CSAS) architecture.

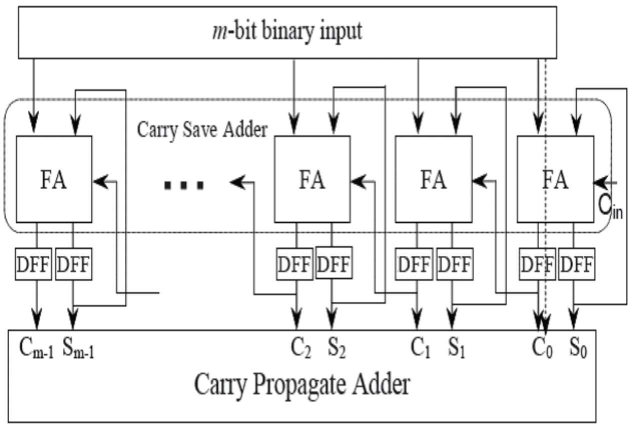

Fig 1Carry Save Adder based accumulator [13]

m-bit binary input is the product bit which is the output from the AND gate. A series of full adders is the critical path. The carry propagate adder (CPA) can be replaced by any other adder block.

Both signed and unsigned multiplication involves AND gates, full adders and D flip flops. . AND gates are used to multiply the serially loaded data. Full adders are employed for addition of PP. The critical path is along the D flip flop and the AND gates. DFFs below the FA are used for the storage of carry that is generated by the FA. The adjacent DFFs are used for shifting of the sum which is the result of the FA.

A. *Unsigned multiplication*

At the first clock cycle, y_0 is loaded to all the AND gates and $x_0,..,x_{n-1}$ are loaded to their respective AND gates. Addition is performed simultaneously by FA. The sum is stored in the adjacent DFF and in the next clock cycle, carry is saved in the DFF below. The sane is repeated for all operand bits in the following clock cycles. After 2n clock cycles, the product P is obtained. A ‘0’ at the MSB side is used for reset, before loading a new set of operands.

B. *Signed multiplication*

The operation is similar to unsigned multiplication except that it involves a bit Q and another EX-OR gate.Q stores the MSB of the operands, which is the signed bit that is EX-ORed with the multiplicand. The result of EX-OR is then ANDed with y_0. Then the product P is obtained as in unsigned multiplication.

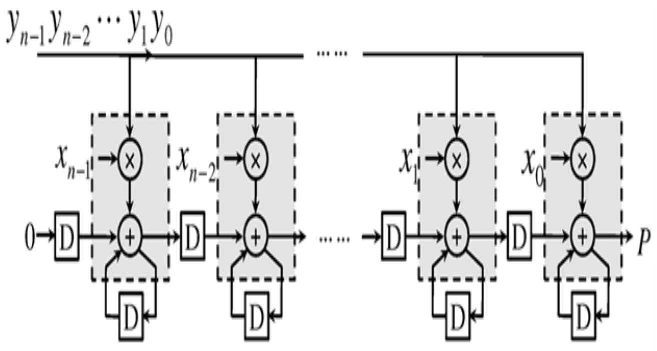

Fig 2 Unsigned CSAS serial-parallel multiplier [4]

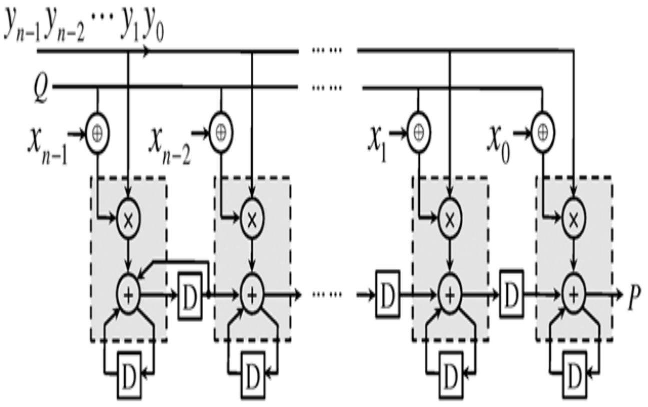

Fig 3Signed CSAS serial-parallel multiplier [4]

In the existing method, data accumulation is done using CSA. The critical path is along the AND gates, FA’s and DFF’s. The disadvantage of this CSA based accumulation is it takes 2n computational cycles to compute the product.

III. THE RIPPLE COUNTER-BASED MULTIPLIER

Asynchronous counter-based data accumulation is introduced in our method to overcome the drawbacks of CSA based accumulation.

A. *Ripple counter-based accumulation*

Accumulator is an adder that adds the current input with the value that is already stored in its internal register. Mathematically, accumulation of n integers x_i for i=0,1,…n-1, can be given by where S is the sum.

$$S = \sum_{i=0}^{n-1} x(i).$$

Asynchronous counters are also called as ripple counters. Here, the DFFs are used for storage. The clock input to the first DFF in a 3 bit 1's counter is the ANDed output of the clock signal and input. The clock for the successive DFF's is driven by the output of its preceding DFF.

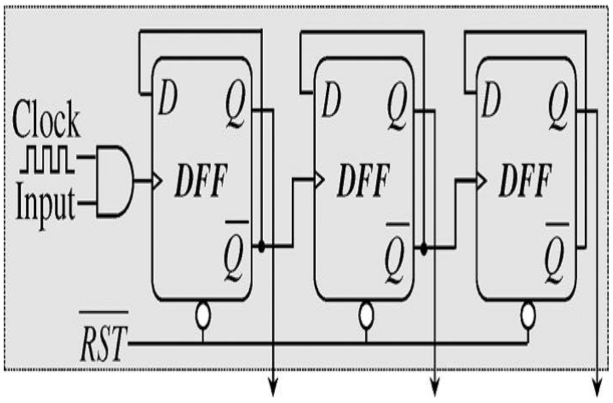

Fig 4 Architecture of 3-bit 1's counter

When a new operand is loaded, the counter gets incremented if a '1' is present in the bit position, corresponding to that column. At the end of n clock cycles, the counters outputs are accumulated in the latching register. These 1's counters are called as accumulators.

B. Ripple Counter-based multiplication scheme

Two serial inputs are considered, one starting from the LSB and the other from the MSB, for generating the individual row of partial products. The product P of two n-bit unsigned binary numbers X and Y is obtained as follows:

$$P = X \cdot Y = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_i y_j 2^{i+j}$$

where$x_i$and $y_i$are the *i*th and*j*th bits of X and Y respectively, with 0 being the LSB. Fig 5 shows the conventional partial product formation of an 8*8 multiplier.

Fig 6 shows the proposed partial product formation of an 8*8 multiplier. To form the PP matrix, the multiplier and the multiplicand are ANDed.To reduce the height of the PP matrix, column to row transformation is performed using 1's counters.The length of the counters varies according to the column height of the PP matrix which gets incremented from 1 to n and then reduces to 1. One row of PPs is generated in each cycle. In reversing the sequence of index *i* in the general equation,

$$P = \sum_{i=n-1}^{0} \sum_{j=0}^{n-1} x_i y_j 2^{i+j}.$$

By rearranging the above equation we get,

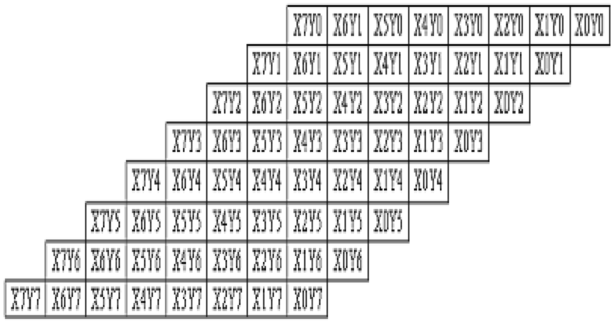
X7Y0 X6Y1 X5Y0 X4Y0 X3Y0 X2Y0 X1Y0 X0Y0
X7Y1 X6Y1 X5Y1 X4Y1 X3Y1 X2Y1 X1Y1 X0Y1
X7Y2 X6Y2 X5Y2 X4Y2 X3Y2 X2Y2 X1Y2 X0Y2
X7Y3 X6Y3 X5Y3 X4Y3 X3Y3 X2Y3 X1Y3 X0Y3
X7Y4 X6Y4 X5Y4 X4Y4 X3Y4 X2Y4 X1Y4 X0Y4
X7Y5 X6Y5 X5Y5 X4Y5 X3Y5 X2Y5 X1Y5 X0Y5
X7Y6 X6Y6 X5Y6 X4Y6 X3Y6 X2Y6 X1Y6 X0Y6
X7Y7 X6Y7 X5Y7 X4Y7 X3Y7 X2Y7 X1Y7 X0Y7

Fig 5 Conventional PP Formation

$$P = \sum_{r=0}^{n-1} PP_r$$

where,

$$PP_r = PP_r^L + PP_r^C + PP_r^R$$

$$PP_r^L = \begin{cases} 0, & r=0 \\ \sum_{k=0}^{r-1} x_{n-k-1} . y_r . 2^{n+r-k-1}, & r=1,2,\ldots,n-1 \end{cases}$$

$$PP_r^C = x_{n-r-1} . y_r . 2^{n-1}, \quad r=0,1,\ldots,n-1$$

$$PP_r^R = \begin{cases} 0, & r=0 \\ \sum_{k=0}^{r-1} x_{n-r-1} . y_{r-k-1} . 2^{n-k-2}, & r=1,2,\ldots,n-1. \end{cases}$$

The partial products so formed are counted column wise for the number of ones. Multiplication is performed in the following manner:

- ANDing of the operands
- Counting the number of 1's
- Adding

x and y are the multiplier and multiplicand bits respectively that are loaded serially.Two clocks, clk1 and clk2 are employed to synchronize the data flow.Clk2 is derived from clk1 to drive the latching register.Clk1 is given to AND gates in order to make sure that their outputs have no glitches.The latching register is used in between the counter and the adder stages to prevent spurious transitions.There are 3 stages in addition: half adder (HA), full adder (FA) and ripple carry adder (RCA).

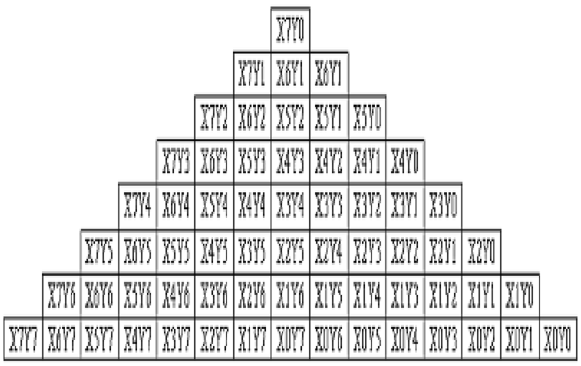
X7Y0
X7Y1 X6Y1 X6Y1
X7Y2 X6Y2 X5Y2 X5Y1 X5Y0
X7Y3 X6Y3 X5Y3 X4Y3 X4Y2 X4Y1 X4Y0
X7Y4 X6Y4 X5Y4 X4Y4 X3Y4 X3Y3 X3Y2 X3Y1 X3Y0
X7Y5 X6Y5 X5Y5 X4Y5 X3Y5 X2Y5 X2Y4 X2Y3 X2Y2 X2Y1 X2Y0
X7Y6 X6Y6 X5Y6 X4Y6 X3Y6 X2Y6 X1Y6 X1Y5 X1Y4 X1Y3 X1Y2 X1Y1 X1Y0
X7Y7 X6Y7 X5Y7 X4Y7 X3Y7 X2Y7 X1Y7 X0Y7 X0Y6 X0Y5 X0Y4 X0Y3 X0Y2 X0Y1 X0Y0

Fig 6 Proposed PP formation [17]

The first 2 adders form the carry save adder (CSA) unit. The product of multiplication is produced in parallel by the final RCA.

The architecture of Counter-based architecture for serial unsigned multiplication is shown in Fig 7. x and y bits are loaded serially and are shifted left and right respectively at the successive clock cycles using DFFs. The ANDed output is then given as input to its corresponding 1's counter. If a '1' is at the output of the AND gate, the counter changes its state at the rising edge of the clock. The counter outputs are accumulated in the latching register at the end of n clock cycles. Now, the PP matrix should further be reduced which is performed by the adder blocks present below the latching register to obtain the product.

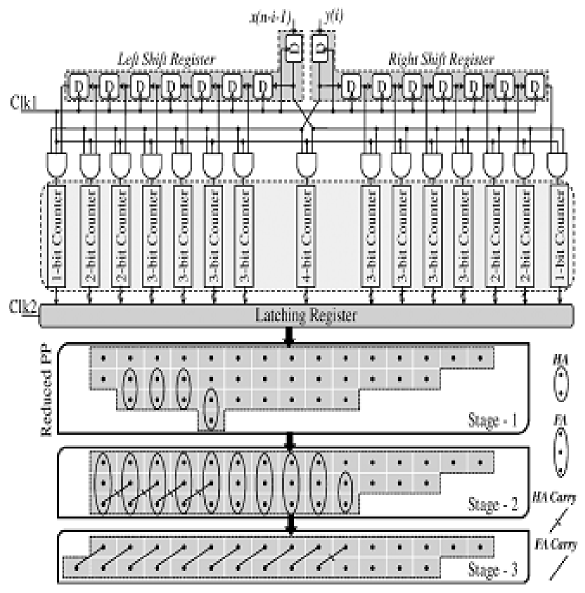

Fig7 Counter-based architecture for serial unsigned multiplication [17]

Baugh-Wooley algorithm is followed for signed multiplication. In this algorithm, the MSB of each row and the last row in PP matrix are complemented, except the MSB of the last row. The architecture of Counter-based architecture for serial signed multiplication is shown in Fig 8. A '1' is added to the immediateleft column of middle column. An inverter for complementing some PP is present in the multiplier block. Now the PPs are added as usual and the other operations are same as unsigned multiplier.

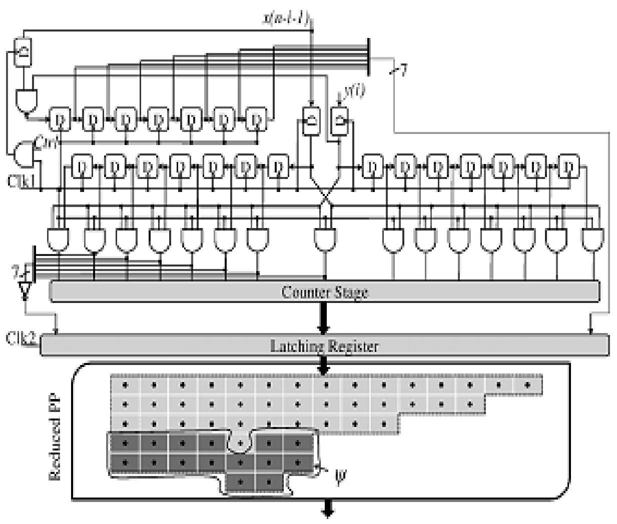

Fig 8 Counter-based architecture for serial signed multiplication [17]

IV. THE RIPPLE COUNTER-BASED MULTIPLIER USING KOGGE-STONE ADDER

The final product is computed by the Ripple carry adder (RCA) in both the above architectures. A number of fast adders like carry-skip adder, carry-select adder and carry look-ahead adders have been proposed in the past [24]-[29].Kogge-Stone adders (KSA) are popular choice in high speedALU design due to its faster operation, regular structure and balanced loading in internal nodes compared to other sparse tree adders.

It is one of the parallel prefix adder structures .In this section, first we briefly discuss the design, operation and general properties of KSA. In KSA, the carries are computed fast by computing them in parallel.

KSA belongs to the family of fastest parallel prefix adders with complexity of $\log_2 N$ (where Nis the width of the adder) meaning thereby that the addition can be done in $\log_2 N$stages. The basic structure of 8-bit radix-2 Kogge-Stone adder [22] is illustrated in Fig 9. It operates on the principle of blockpropagate (p) andblock generate (g) [23]. The block propagate determines whether the input carry can propagate through the block of bits or not. The block generate determines if the block of bits can generate a carry or not.

If aand bare input operands of the adder, the propagate/generate and carry in (Ci)/carry out (Ci+1) are related as follows

$$p_i = a_i \oplus b_i; \quad g_i = a_i \bullet b_i; \quad C_{i+1} = g_i + p_i C_i$$

In absence of carry input to the adder core (i.e., C-1= 0), the generate signal become the carry inputs for the intermediate carry computations. In 8-bit KSA, the carry iscomputed ink=3stages. The logic-level implementation of thebasic cells used in parallel-prefix adders are shown in Fig 10.

RCA is replaced by KSA for improved performance. The average connection delay in the multiplier architecture due to KSA is reduced and is expressed in terms of Nano seconds.

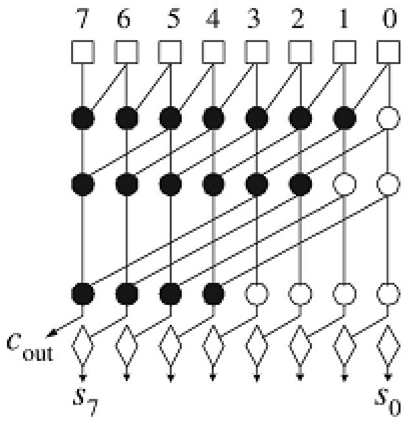

Fig 9 The basic structure of 8-bit radix-2 Kogge-Stone adder [22]

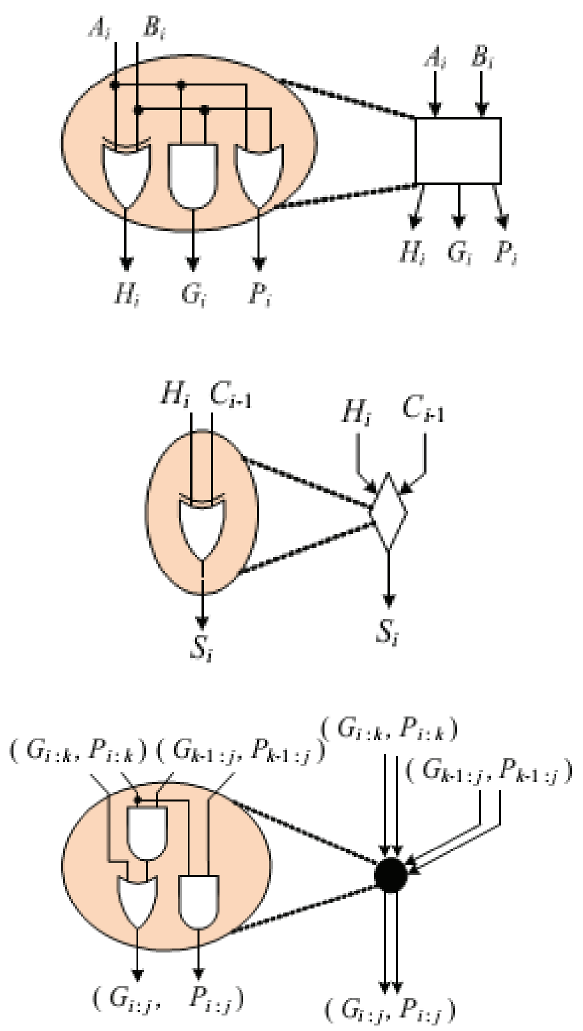

Fig 10 The logic-level implementation of the basic cells used in parallel-prefix adders

V. COMPARISON RESULTS

In the CSAS multiplier, the inputs has to pass through AND gate, Full adder and DFF. The critical path delay exists along these units. This technique requires 2n clock cycles for computation. In the counter-based multiplier, the critical path delay exists in only one AND gate of the counter. Hence it requires n clock cycles for computation [17]. 8*8 multiplication has been performed for both the techniques. The simulation results have been compared using Modelsim 10.0b. The average connection delay in the multiplier architecture due to KSA is reduced and is expressed in terms of Nano seconds. The delay report is obtained by simulating the code in Xilinx ISE simulator.

Table I

Parameter	CSAS multiplier		Ripple counter-based multiplier	
Number of clock cycles	Unsigned	Signed	Unsigned	Signed
	16	16	8	8

Table II

Parameter	Ripple counter based multiplier with RCA		Ripple counter based multiplier with KSA	
Average	Unsigned	Signed	Unsigned	Signed
connection delay(nano seconds)	1.541	1.417	1.339	1.395

VI. CONCLUSION

In our paper, a new method for computing serial multiplication by using ripple countersis introduced. It is possible to generate all the rows serially in just n cycles for an n* n multiplication. The counters are used for counting the number of 1's in each column, which allows the partial product bits to be generated and partially accumulated in place with a critical path delay of only an AND gate and a D Flip-Flop(DFF) . Initially the design is implemented in Model Sim 10.0a using VHDL code and it is observed from the above comparison *Table I* that the number of computational clock cycles has been reduced from 16 to 8 for an 8*8 multiplication. Then the code is simulated using Xilinx ISE Simulator and the delay report is extracted. Now from *Table II*, it is clear that the average connection delay is reduced to 1.339ns and 1.395ns from 1.541ns and 1.417ns for unsigned and signed operations of ripple counter-based multiplier with RCA and KSA respectively. n * n multiplication can also be performed.

REFERENCES

[1]. A. K. Lenstra and E. Verheul, "Selecting cryptographic key sizes,"J.Cryptology, vol. 14, no. 4, pp. 255–293, 2001.
[2]. N. R. Strader and V. T. Rhyne, "A canonical bit-sequential multiplier," IEEE Trans. Comput., vol. C-31, no. 8, pp. 791–795, Aug. 1982.
[3]. R. Gnanasekaran, "On a bit-serial input and bit-serial output multiplier,"IEEE Trans. Comput., vol. C-32, no. 9, pp. 878–880, 1983.
[4]. R. Gnanasekaran, *"A fast serial-parallel binary multiplier,"* IEEE Trans. Comput., vol. C-34, no. 8, pp. 741–744,1985.
[5]. P. Ienne and M. A Viredaz, "*Bit-serial multipliers and squarers*," IEEE Trans. Comput., vol. 43, no. 12, pp. 1445–1450,1994.
[6]. Mark Vesterbacka, Kent Palmkvist, and Lars Wanhammar, *"Serial squarers and serial/serial multipliers".*
[7]. A. D. Booth, *"A signed binary multiplication technique,"* Quarterly J.Mechan.Appl. Math., vol. 4, no. 2, pp. 236–240, 1951.
[8]. C.P. Lerogue, P.Gerard, and J.S. Colardelle, " A fast 16- bit NMOS parallel multiplier", IEEE Journal of Solid-state circuits, Vol. 19, no.3, pp. 338-342, Mar 1984.
[9]. Jin-HaoTu and Lan-Da Van, "Power-Efficient PipelinedReconfigurable Fixed-Width Baugh-Wooley Multipliers" IEEE Transactions on computers, vol. 58, No. 10, October 2009.
[10]. A. Dandapat, S. Ghosal, P. Sarkar, D. Mukhopadhyay (2009), "A 1.2- ns16×16-Bit Binary Multiplier Using. High Speed Compressors", International Journal of Electrical, Computer, and Systems Engineering, 2009, 234-239.
[11]. William J. Stenzel, William J. Kumitz,Member IEEE, GillesH.Garcia, "Compact *High- Speed Parallel Multiplication Scheme*" IEEE Trans. Comput. C-26:948-957(1977).
[12]. S. Haynal and B. Parhami, *"Arithmetic structures for inner-product and other computations based on a latency-free bit-serial multiplierdesign,"*presented at the 13th AsilomarConf. Signals, Syst. Comput., Geneva, Switzerland, 1996.
[13]. ManasRanjanMeher, Ching-ChuenJong,Chip-Hong Chang "High-Speed and Low-Power Serial Accumulator for Serial/Parallel Multiplier" IEEE Xplore,2010.
[14]. B. Parhami, Computer Arithmetic: Algorithms and Hardware De-signs. New York: Oxford Univ. Press, 2009.
[15]. R. Menon and D. Radhakrishnan, "High performance 5:2 compressor architectures,"IEEProc-Circuits Devices Syst., vol. 153, no. 5, pp.447–452, Oct. 2006.
[16]. G. Bi and E. V. Jones, "High-performance bit-serial adders andmultipliers,"IEEProc G-Circuits Devices Systs, vol. 139, no. 1, pp.109–113, Feb. 1992.

[17]. “*A High Bit Rate Serial-Serial Multiplier With On-the-Fly ccumulation by Asynchronous Counters*”ManasRanjanMeher, *Student Member, IEEE*, ChingChuen Jong, and Chip-Hong Chang, *Senior Member, IEEE,* IEEE transactions on very large scale integration (VLSI) systems, 2011.
[18]. Y.Kim and L.S .Kim, “64-bit carry-select adder with reduced area,” Electron. Lett. vol.37, no.10, pp.614-615, May 2001.
[19]. R.P.BrentandH.T.Kung,“ARegular Layout forParallel Adders,” IEEE Transactions on Computers,vol. C-31,no.3,pp.260–264, 1982.
[20]. J. Sklansky, “Conditional-sum addition logic,” IRE Transactions on Electronic Computers, vol. 9, pp. 226–231, 1960.
[21]. http://en.wikipedia.org/wiki/Kogge%E2%80%93Stone_adder
[22]. P.M. Kogge and H.S. Stone, “A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,”IEEE Trans. Computers, vol. C-22, no. 8, pp. 786-792, Aug. 1973.
[23]. J.Rabaey, “Digital Integrated Circuits: A Design Perspective”, Prentice Hill, Second Edition, 2003.
[24] S. Ghosh et al., “Tolerance to small delay defects by adaptive clock stretching,” IOLTS, 2007.
[25] P. Ndai et al., "Fine-grained redundancy in adders,” ISQED, 2007.
[26] W. J. Townsend et al., "Quadruple time redundancy adders," DFT, 2003.
[27] B. W. Johnson, “Design and analysis of fault tolerant digital systems,” Addison Wesley Publishing Company, 1989.
[28] J. H. Patel et al., “Concurrent error detection in ALUs by recomputing with shifted operands,” Trans. Comput., 1982.
[29] K. Wu et al., “Algorithm level RE-computing with shifted operands - a register transfer level concurrent error detection technique,” ITC, 2000.

Vishnupriya.A, received the B.E. degree in Electronics and Communication Engineering, from V.S.B. Engineering College, Karur, affiliated to Anna University Chennai, India, in the year 2010. Her current technical interests include Hardware Description Language (HDL) and low power VLSI Design.She is currently pursuing final year M.E. in VLSI Design at Avinashilingam Institute for Home Science and Higher Education for Women University, Coimbatore, India.

AlthafKhan.J,received the B.E. degree in Electronics and Communication Engineering, from V.S.B. Engineering College, Karur, affiliated to Anna University Chennai, India in the year 2010. He has got close to two years of experience in theInformation Technology and is currently working as Programmer Analyst at Cognizant Technology Solutions (CTS), Coimbatore, India.

Gopalakrishnan.R,completed the M.E degree in Power System Engineering at Anna University, Coimbatore and received the B.E. degree in Electrical and Electronics Engineering at Muthayammal Engineering College, Namakkal, affiliated to Anna University Chennai, India, in the year 2006. His current technical interests are Power Electronics, Circuit Theory and Electrical Machines. He has got two years of teaching experience. He is currently working as Assistant Professor in the department of Electrical and Electronics Engineering at P.A. College of Engineering and Technology, Coimbatore, India.