ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 7, Issue 5, May 2018

DESIGN OF LOW LATENCY VEDIC MULTIPLIER ARCHITECTURE WITH ADAPTIVE HOLD LOGIC

[1]PREETHI.S, Assistant professor, Dept. of ECE,
Pollachi Institute Of Engineering and Technology- Pollachi, Tamilnadu, India,

***ABSTRACT*-- The effect of reducing the aging problem in the multiplier deign plays a major role in improving the system performance. The overall performance of the any systems depends on the throughput of multiplier design. The aging problem in the multiplier significant affects the overall performance of the system. Therefore, it is important to design reliable, high-performance multipliers. This paper proposes an aging-aware variable latency multiplier design with adaptive hold logic (AHL) circuit. Use of the variable latency multiplier in re-execution of the clock cycle eliminates the occurrence of delays and timing violations. The variable latency multiplier also provides higher throughput and can adjust the adaptive hold logic(AHL) circuit to reduce the performance degradation that is caused due to aging effects of transistors. In this paper, the variable latency design with adaptive hold logic applied to existing and proposed method. The proposed method of a Vedic multiplier with AHL circuit helps to increase the performance of the multiplier. It can be able to reduce timing violations and also provide low power consumption through AHL circuit. Finally, the experimental results, analyze that 4x4 Vedic multiplier with AHL can achieve low power and delay as compared to the array, row by-passing multiplier with AHL and Vedic multiplier without AHL.**

***Key words*--Vedic multipliers, Razor flip flop, Adaptive Hold Logic, Variable latency.**

I. INTRODUCTION:

Since the era of integrated circuit, digital multipliers have been applied for many applications because of its most critical arithmetic functional system. Inherently, it is adopted in day to day applications through some platforms, especially Fourier Transforms, Discrete Cosine Transforms. Compared to any other digital systems it too has some precise error tradeoff between accuracy and hardware complexity. Moreover, if our chosen multiplier design is too slow, then the performance of entire circuits will be reduced and there will be also computation errors. To overcome this many compensation techniques will be presented for array multipliers. Along with other considerations, truncation error in multipliers was too considered as a hassle nowadays. As if it is being used in Digital Signal Processing (DSP) and Application Specific Integrated Circuits (ASICs), many engineering computations, mainly in computer machines, it has received a greater attestation to utilize it more effectively and efficiently. In the conventional part of Digital Multipliers, the basic errors were considered an issue, but the legacy of multiplier continued as

such and their applications caused a major paradigm shift in day to day applications because, the then called cons were today advanced and supplemented with other parameters like Testability, Low Power, Flexibility, Reliability. Even power dissipation has considered an important constraint in the design of digital multipliers.

Here by on observing many combinations in digital systems, **a Vedic multiplier using Razor Flip-Flop and Adaptive Hold Logic (AHL) is chosen, for reducing the delay.** The architecture of Vedic multiplier has many improved segments on comparison with Array, Row by-pass multipliers, Column by-pass multipliers and then any other multipliers. My preference towards Vedic multiplier is that it has an architecture in such a way that the multiplier's speed is increased, as it requires lesser gate counts compared to Array and Row by-pass multipliers. Moreover, this proposed system uses less number of full adders that increases the computational speed and too. The proposed highly efficient method of multiplication – **"Urdhva Tiryakbhyam Sutra"** based on Vedic mathematics with Adaptive Hold Logic. It is a method for hierarchical multiplier design which clearly indicates the computational advantages offered by Vedic methods. Vedic multiplier with AHL implemented the Verilog code on Xilinx. The computational path delay for proposed 4x4 bit Vedic multiplier is found to be 10.789 ns. It is observed that the Vedic multiplier is much more efficient than Array multiplier with AHL and Row bypassing multiplier with AHL in terms of execution time (speed). Accordingly, this system is supplemented with Razor Flip-Flop and Adaptive Hold Logic (AHL). The tilt towards Razor Flip-Flop is because of its legendary removal of timing violations and re-executing the operation.

II. PRELIMINARIES

A) ARRAY MULTIPLIER

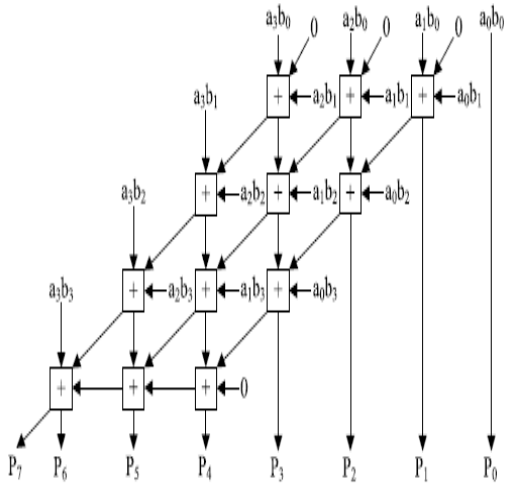

Fig 1. 4X4 Array multiplier

The array multiplier is a fast parallel multiplier and is shown in Fig.1 and it consists of (n-1) rows of carry save adder, in which each row contains (n-1) full adders. Each full adder in the carry save adder array has two outputs, they are the sum bit goes down and the carry bit goes to the lower left full adder. The ripple adder for carry propagation. The FAs in the AM are always active regardless of input states and it can be found in [5].

B) ROW BY-PASSING MULTIPLIER

A low-power row-bypassing multiplier [2] is also proposed to reduce the activity power of the AM. The operation of the low-power row-bypassing multiplier is similar to that of the low-power column-bypassing multiplier, but the selector of the multiplexers and the tristate gates use the multiplicator.

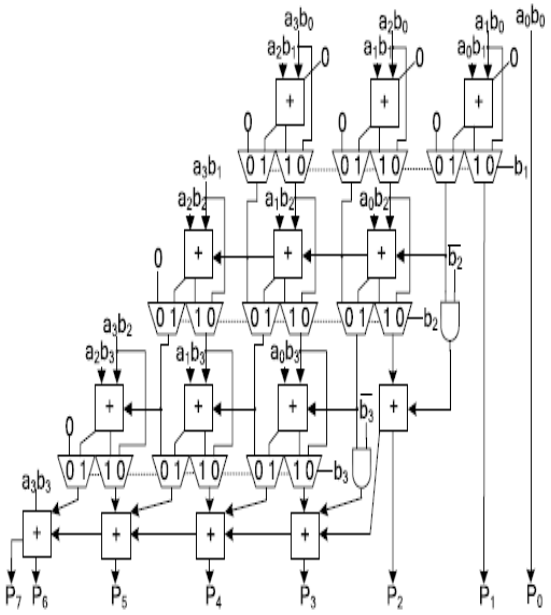

Fig 2. 4x4 Row by-passing multiplier

Fig 2. shows a 4 × 4 row-bypassing multiplier. Each input is connected to an FA through a tristate gate. When the inputs are 1111 * 1001, the two inputs in the first and second rows are 0 for FAs. Because *b*1 is 0, the multiplexers in the first row select *aib*0 as the sum bit and select 0 as the carry bit. The inputs are bypassed to FAs in the second rows, and the tristate gates turn off the input paths to the FAs. Therefore, no switching activities occur in the first-row FAs; in return, power consumption is reduced. Similarly, because *b*2 is 0, no switching activities will occur in the second-row FAs. However, the FAs must be active in the third row because the *b*3 is not zero. More details of the row-bypassing multiplier can also be found in [2].

C) VARIABLE-LATENCY DESIGN

Traditional circuits use critical path delay as the overall circuit clock cycle in order to perform correctly.

However, the probability that the critical paths are activated is low. In most cases, the path delay is shorter than the critical path. For these noncritical paths, using the critical path delay as the overall cycle period will result in significant timing waste. Hence, the variable-latency design was proposed to reduce the timing waste of traditional circuits. The variable-latency design divides the circuit into two parts: 1) shorter paths and 2) longer paths. Shorter paths can execute correctly in one cycle, whereas longer paths need two cycles to execute. When shorter paths are activated frequently, the average latency of variable-latency designs is better than that of traditional designs. For example, several variable-latency adders were proposed using the speculation technique with error detection and recovery [13] and [8]. A short path activation function algorithm was proposed in [11] to improve the accuracy of the hold logic and to optimize the performance of the variable-latency circuit. An instruction scheduling algorithm was proposed in [12] to schedule the operations on nonuniform latency functional units and improve the performance of Very Long Instruction Word processors.

In [6], process-variation tolerant architecture for arithmetic units was proposed, where the effect of process-variation is considered to increase the circuit yield. In addition, the critical paths are divided into two shorter paths that could be unequal and the clock cycle is set to the delay of the longer one. These research designs were able to reduce the timing waste of traditional circuits to improve performance, but they did not consider the aging effect and could not adjust themselves during the runtime. A variable latency adder design that considers the aging effect was proposed in [7]. However, no variable-latency

multiplier design that considers the aging effect and can adjust dynamically has been done.

III. ALGORITHMS OF PROPOSED VEDIC MATHEMATICS

A) VEDIC MULTIPLICATION

The proposed Vedic multiplier is based on the Vedic multiplications formula9sutras). Vedic multiplier can be deals with more basic operations as well as not Simple mathematical operations. In this Vedic multiplier the basic arithmetic operation is done in a simple and powerful manner. It is a unique method of calculation and done by simple principles and rules. Vedic mathematics is mainly depends on 16 sutras. These sutras have been conventionally used for the multiplications of two numbers in decimal number system and it deals with several applications in mathematics like arithmetic, trigonometry, algebra etc.

B) URDHAVA MULTIPLIER

Urdhava Tiryakbhyam [14] [15] (Vertically and Crosswise), is one of Sixteen Vedic Sutras and deals with the multiplication of numbers. The sutra is illustrated in Example 1 and the hardware architecture is depicted in Fig.7. In this example, two decimal numbers (522 x 715) are multiplied. Fig 4 shows the Multiplication of two 4 bit numbers using the Urdhava Tiryakbhyam method. The digits on the two ends of the line are multiplied and the result is added to the previous carry. When three or more lines are present, all the results are added to the previous carry. The least significant digit of the number, thus obtained acts as one of the result digits and the rest act as the carry for the next step. Initially the carry is taken to be zero.

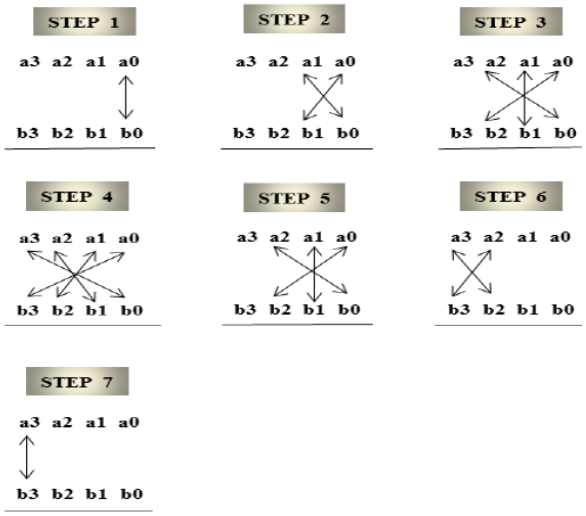

Fig 3. Multiplication of two 4 bit numbers using Urdhava Tiryakbhyam method

Example 1:

Multiplication of two decimal numbers- 522*715

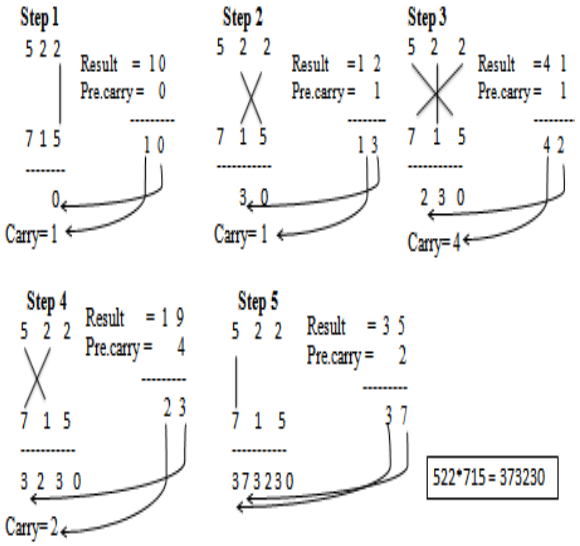

Fig 4. Multiplication of two numbers

From the Example 1, it is observed that all the partial products are generated in parallel. So the speed of the multiplier is higher compared to array multiplier.

The above discussions can now be extended to a multiplication of binary number system with the preliminary knowledge that the multiplication of two bits a0 and b0 is just an AND operation and can be implemented using simple AND gate. To illustrate this multiplication scheme in the binary number system, consider the multiplication of two binary numbers a3a2ala0 and b3b2b1b0. As the result of this multiplication would be more than 4 bits, the product is expressed as r7r6rSr4r3r2rlr0. Least significant bit r0 is obtained by multiplying the least significant bits of the multiplicand and the multiplier as shown in the Fig 3. The digits on both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result (rn) and a carry (Cn). This carry is added in the next step and thus the process goes on. If more than one line is there in one step, all the results are added to the previous carry. In each step, least significant bit acts as the result bit and the other entire bits act as carry. For example, if in some intermediate step, we get 110, then 0 will act as result bit and 11 as the carry (referred to as Cn in this text). It should be clearly noted that Cn may be a multi-bit number. Thus the following expressions (1) to (7) are derived:

r0=a0b0……...(1)
c1r1=a1b0+a0b1……..…(2)
c2r2=c1+a2b0+a1b1 + a0b2 ……..…(3)
c3r3=c2+a3b0+a2b1 + a1b2 + a0b3…..…..(4)
c4r4=c3+a3b1+a2b2 + a1b3……...(5)
c5r5=c4+a3b2+a2b3……..……(6)
c6r6=c5+a3b3 ……..……...(7)

with c6r6rSr4r3r2r1r0 being the final product. Partial products are calculated in parallel and hence the delay involved is just the time it takes for the signal to propagate through the gates.

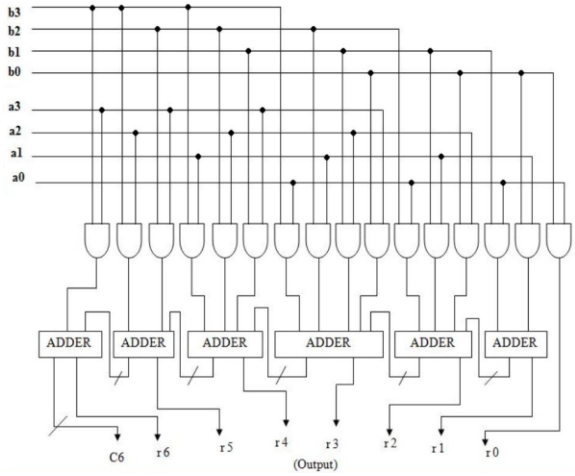

Fig 5. Architecture of the Urdhva tiryakbhyam Multiplier

The main advantage of the Vedic Multiplication algorithm (Urdhava Tiryakbhyam Sutra) stems from the fact that it can be easily implemented in FPGA due to its simplicity and regularity [3]. The digital hardware realization of a 4-bit multiplier using this Sutra is shown in Fig. 5. This hardware design is very similar to that of the array multiplier where an array of adders is required to arrive at the final product. Here in Urdhava, all the partial products are calculated in parallel and the delay associated is mainly the time taken by the carry to propagate through the adders.

IV PROPOSED AGING-AWARE VEDIC MULTIPLIER WITH AHL

A) PROPOSED ARCHITECTURE

The proposed aging-aware multiplier architecture, which includes two m-bit inputs (m is a positive number), one 2m-bit output, one array, row/column-

bypassing and Vedic multiplier, 2m1-bit Razor flip-flops, and an AHL circuit. In [17], the proposed architecture, the Vedic multipliers can be examined by the number of zeros in either the multiplicand or multiplicator to predict whether the operation requires one cycle or two cycles to complete. When input patterns are random, the number of zeros and ones in the multiplicator and multiplicand follows a normal distribution.

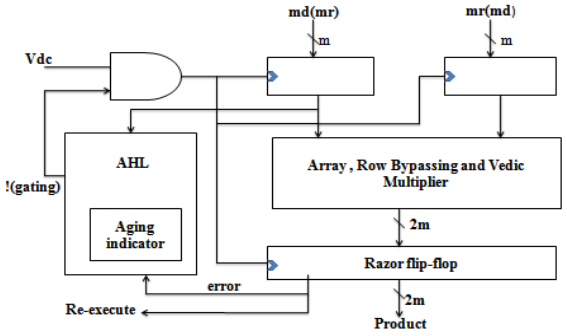

Fig 6. Proposed architecture (md- multiplicand; mr - Multiplicator)

Hence, the three aging-aware multipliers can be implemented using similar architecture, and the difference between the array and row by-passing multipliers lies in the input signals of the AHL.

B) VEDIC 4 X 4 BIT MULTIPLIER

In this section 4 X 4 Vedic multiplier architecture is discussed. Consider two 4-bit numbers such as A and B where A = A3A2A1A0 and B = B3B2B1B0. The LSBs of two numbers (A0 X B0) are multiplied to generate LSB S0 of the final result. The same procedure is followed here as 2 X 2 Vedic multiplication procedure discussed earlier. The initially pre - carry is set to zero. In each and every step generated query is forwarded to the next step and the process goes on. At the end C6 and S6 are obtained after performing last step. Finally generated result is given as C6S6S5S4S3S2S1S0. The 4 X 4 Vedic multiplication diagram is shown in fig 7.

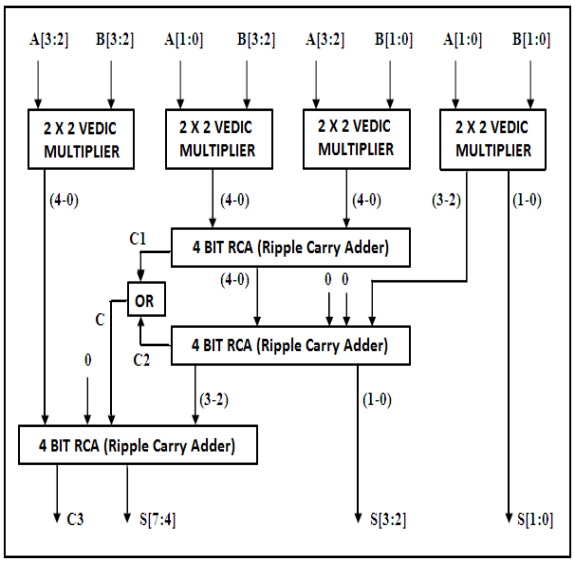

Fig 7. 4 X 4 Vedic multiplier architecture

Here, 2 X 2 Vedic multipliers are used to implement 4 X 4 Vedic multiplier to generate the partial product. Three ripple carry adders of 4 bits each are used for addition of generating partial products. The carry output of first two ripple carry adders are ORed and output of this OR gate is given to the next ripple carry adder. Zero inputs are given to some of the ripple carry adders wherever required. The arrangement of ripple carry adders is made in such way that computation time required for the whole multiplication process is reduced and speed of working is increased.

C) RAZOR FLIP FLOPS

In [3], A 1-bit Razor flip-flop contains a main flip-flop, shadow latch, XOR gate, and MUX. The main flip-flop catches the execution result of the combination circuit using a normal clock signal, and the shadow latch catches the execution result using a delayed clock signal, which is slower than the normal

clock signal. If the latched bit of the shadow latch is different from that of the main flip-flop, this means the path delay of the current operation exceeds the cycle period, and the main flip-flop catches an incorrect result. If errors occur, the Razor flip flop will set the error signal to 1 to notify the system to re-execute the operation and notify the AHL circuit that an error has occurred. We use Razor flip-flops to detect whether an operation that is considered to be a one-cycle pattern can really finish in a cycle. If not, the operation is re-executed with two cycles.

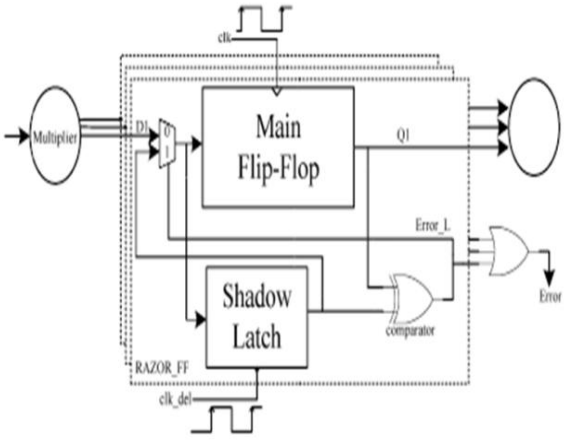

Fig 8. Razor Flip Flop.

D) ADAPTIVE HOLD LOGIC (AHL)

The Adaptive Hold Logic circuit is the key component of variable-latency multiplier. The AHL circuit contains decision block, MUX and a D flip-flop. If the cycle period is too short, the column-bypassing multiplier is not able to complete these operations successfully, causing timing violations. These timing violations will be caught by the Razor flip-flops, which generate error signals. If errors happen frequently, it means the circuit has suffered significant timing degradation due to the aging effect.

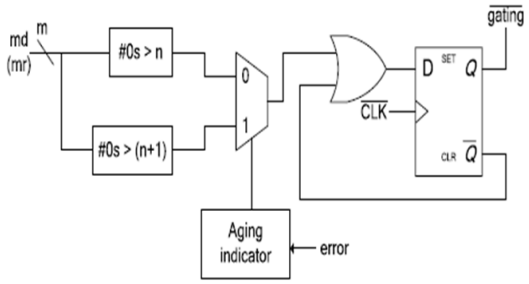

Fig 9. Adaptive Hold Logic

When an input pattern arrives, decision block will decide whether the pattern requires one cycle or two cycles to complete and pass both results to the multiplexer. The multiplexer selects one of either result based on the output of the razor flip-flop. Then an OR operation is performed between the result of the multiplexer, and the Q bar signal is used to determine the input of the D flip-flop. When the pattern requires one cycle, the output of the multiplexer is 1. The (gating) signal will become 1, and the input flip-flops will latch new data in the next cycle. On the other hand, when the output of the multiplexer is 0, which means the input pattern requires two cycles to complete, the OR gate will output 0 to the D flip-flop. Therefore, the (gating) signal will be 0 to disable the clock signal of the input flip-flops in the next cycle.

E) OPERATION OF VEDIC MULTIPLIER USING AHL

When an input pattern arrives, the Vedic multiplier and AHL perform their operation at a same time. The multiplicand have number of zeros, the adaptive hold circuit decides it takes 1 or 2 cycles. After finishing the operation of Vedic multiplier and it goes to razor flip-flop.

The razor flip-flop checks whether there have any path delay timing violation. So, if any timing

violation occurs, it executes the operation by using two cycle's patterns and it indicates to the AHL. Finally, our architecture minimizes the timing violation of the circuit.

V SIMULATION RESULTS AND TABLES

The result shows the comparison of Vedic multiplier against array and row bypassing multipliers ,where AHL is used in all the three multipliers. These multipliers can be implemented using VHDL coding. The delay report is obtained by synthesizing the multipliers with Xilinx and Modelsim. The architectural design, power and area calculations of the proposed multiplier is carried out with the help of a Cadence digital tool(Cadence Encounter ® RTL Complier).

A) ARCHITECTURE DESIGN AND SIMULATION RESULTS

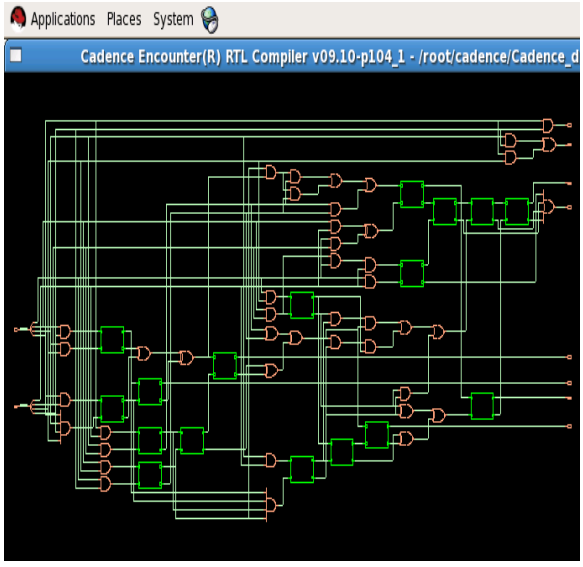

Fig 10. RTL schematic view of 4X4 Vedic multiplier without AHL.

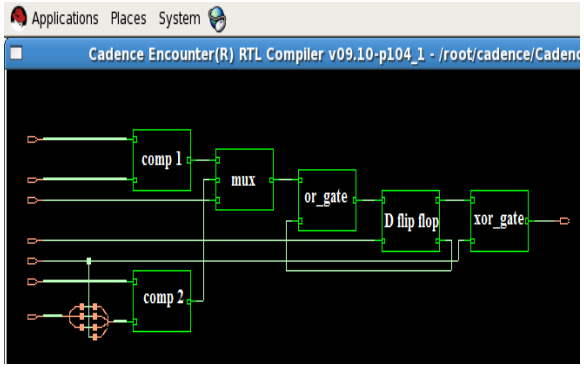

Fig 11. RTL schematic view of AHL.

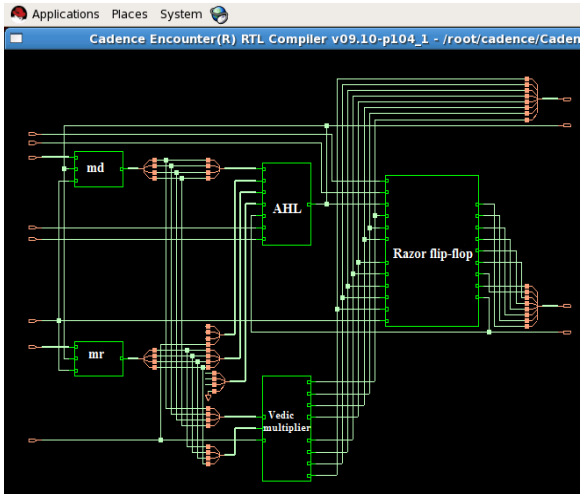

Fig 12. Architecture of aging aware Vedic multiplier with AHL

The architecture design of proposed Vedic multiplier with AHL are designed by using the Cadence digital tool_(Cadence Encounter(® RTL Compiler) and it is shown in Fig 12.

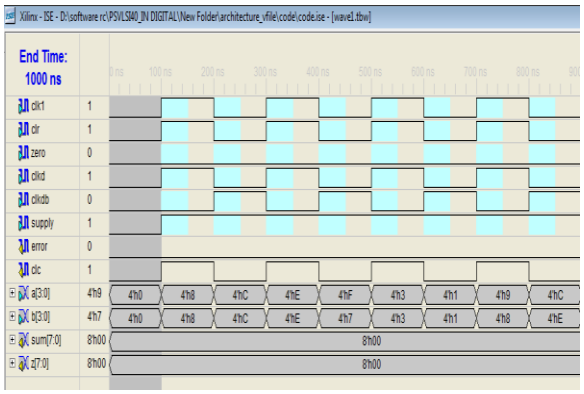

Fig 13. Input wave form

The Fig 13. shows the various for Input combinations, supply, Reset and clock of the array, row bypassing and Vedic multiplier with AHL. Initial error is set as '0' after that it is again changing to '1' state.

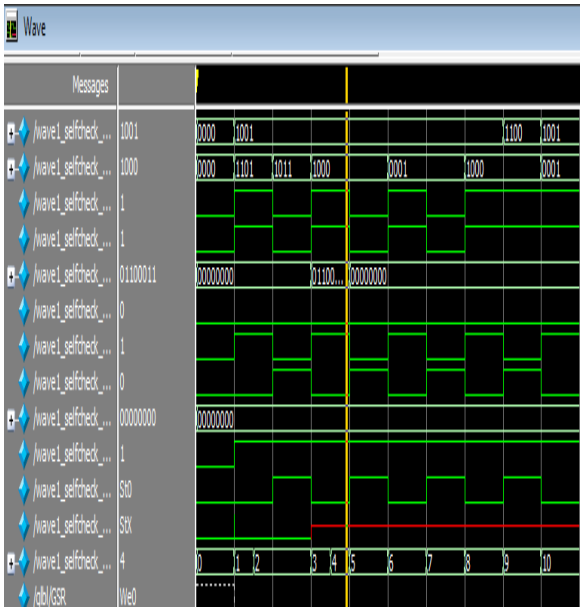

Fig 14. 4 X 4 Array multiplier with AHL.

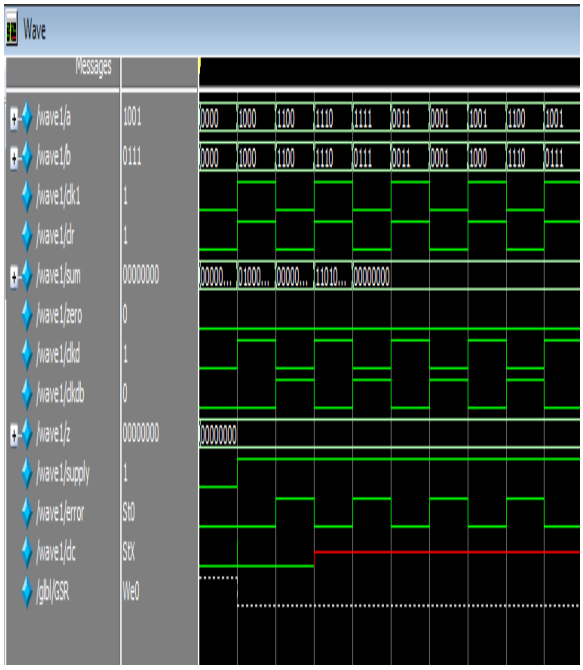

Fig 15. 4X4 Row bypassing multiplier with AHL.

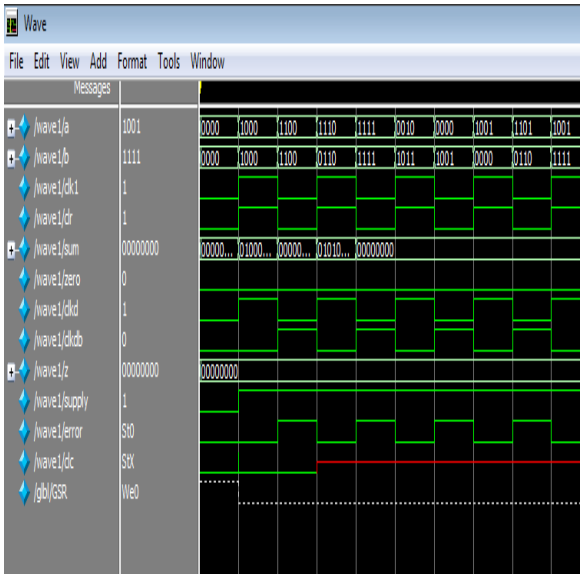

Fig 16.Vedic multiplier with AHL.

The fig 14,15 and 16 shows the waveform of aging aware array multiplier, row bypassing multiplier and Vedic multiplier with AHL. Here we have two inputs A and B, with input values 1110 and 1001 respectively. The output produced is given as sum, which is 01001000. The clock signal/gating signal is provided at the bottom of the waveform next to the error signal. The product of multiplier and multiplicand is not predictable exactly since AHL is used in the aging aware column bypass multiplier.

This is because the exact output is provided only after the error is detected by AHL. The gating signal is represented as green line and red line, which denotes the dormant ant active conditions of the multiplier respectively. The multiplier though remains on, but inactive, it works exactly only when the gate signal is applied. The error signal will be low when the number of zeros is less and is represented by '1' in the sum. Lower the error, the higher is the number of '1' in the sum. The error signal increases with an increase in the number of zeros in the sum. This error is detected by the razor flip-flop and is sent to the aging indicator.

B) TABLE

1) PERFORMANCE COMPARISSION TABLE

PARAMETERS	4*4 ARRAY MULTIPLIER WITH AHL	4*4 Row Bypassing MULTIPLIER WITH AHL	4*4 Vedic MULTIPLIER WITH AHL
TOTAL DELAY	18.371ns	18.085ns	10.789ns
TOTAL MEMORY USAGE	188616 Kilobytes	188040 Kilobytes	187976 Kilobytes
INTERNAL POWER	11131.92nW	16618.07nW	11006.00nW

DELAY COMPARISION CHART

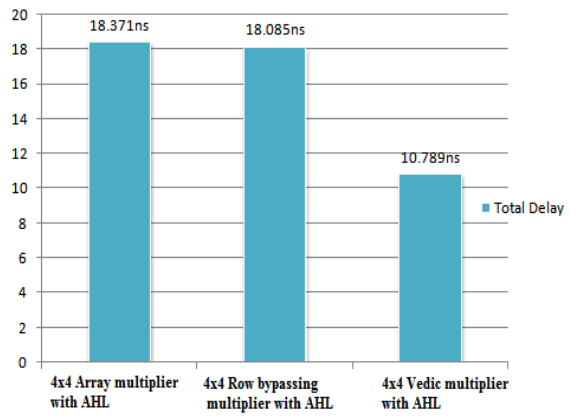

Figure 17. Delay estimation of various 4x4 Multipliers with AHL

2) DELAY COMPARISSION TABLE BETWEEN VEDIC MULTIPLIER WITHOUT AND WITH AHL

PARAMETERS	4*4 VEDIC MULTIPLIER WITHOUT AHL	8*8 VEDIC MULTIPLIER WITHOUT AHL	4*4 VEDIC MULTIPLIER WITH AHL
DELAY	15.778ns	25.853ns	10.789ns

DELAY COMPARISION CHART

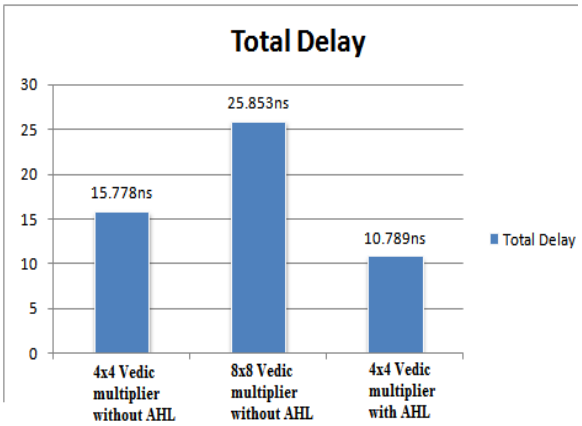

Figure 18. Delay estimation of various 4x4 Vedic Multipliers

The above tabulation, I and II depicts that the power and delay analyses for 4X4 Vedic multiplier with AHL compared with the array with AHL and row bypassing multiplier with AHL and also done with 4x4 Vedic multiplier with and without AHL by simulation. The comparison proves that by the use of 4x4 Vedic multiplier with AHL power and delay got reduced.

VI CONCLUSION

This work proposed an aging-aware variable-latency Vedic multiplier design with AHL along with total gate delay, memory usage and estimated power consumption measurements to

achieve reliable high performance. Previously Array multipliers and Row by-pass multipliers with AHL are designed with disadvantage like more delay time and very less speed. The proposed Vedic multiplier with AHL overcomes the above mentioned disadvantage and the proposed variable latency multiplier is able to adjust the AHL to mitigate performance degradation due to increased delay and it has three key features. First, it is a variable-latency design that minimizes the timing waste of the noncritical paths. Second, it can provide reliable operations after the aging effect occurs. The Razor flip-flops detects the timing violations and re-execute the operations using two cycles. Finally, our architecture can adjust the percentage of one-cycle patterns to minimize performance degradation due to the aging effect. When the circuit is aged, and many errors occur, the AHL circuit uses the second judging block to decide if an input is one cycle or two cycles.

VII REFERENCE

[1] A.D. Booth, "A Signed Binary Multiplication Technique", Qrt. J. Mech. App. Math.,, vol. 4, pp. 236–240, 1951.

[2] J. Ohban, V. G. Moshnyaga, and K. Inoue, "Multiplier energy reduction through bypassing of partial products," in *Proc. APCCAS*,2002, pp. 13–17.

[3] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 36th Annu. IEEE/ACM MICRO*, Dec. 2003, pp. 7–18.

[4] M.-C. Wen, S.-J. Wang, and Y.-N. Lin, "Low power parallel multiplier with column bypassing," in *Proc. IEEE ISCAS*, May 2005, pp. 1638–1641.

[5] Jung-Yup Kang, Member, IEEE, and Jean-Luc Gaudiot,,"A Simple High-Speed Multiplier Design", IEEE Transactions on Computers, Vol. 55, No. 10, October 2006, pp.1253-1258.

[6] D. Mohapatra, G. Karakonstantis, and K. Roy, "Low-power processvariation tolerant arithmetic units using input-based elastic clocking," in *Proc. ACM/IEEE ISLPED*, Aug. 2007, pp. 74–79.

[7] D. Baneres, J. Cortadella, and M. Kishinevsky, "Variable-latency design by function speculation," in *Proc. DATE*, 2009, pp. 1704–1709.

[8] Y. Chen *et al.*, "Variable-latency adder (VL-Adder) designs for low power and NBTI tolerance," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 11, pp. 1621–1624, Nov. 2010.

[9] Anvesh kumar,Ashish raman,"Low Power, High Speed ALU Design by Vedic Mathematics"publish in National conference organized by NIT,hamirpur,2009.

[10] Prabir Saha, Arindam Banerjee, Partha Bhattacharyya, Anup Dandapat ""High Speed ASIC Design of Complex Multiplier Using Vedic Mathematics" Proceeding of the 2011 IEEE Students' Technology Symposium 14-16 January, 20 II, lIT Kharagpur .

[11] Y.-S. Su, D.-C. Wang, S.-C. Chang, and M. Marek-Sadowska, "Performance" optimization using variable-latency design style," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 10, pp. 1874–1883, Oct. 2011.

[12] N. V. Mujadiya, "Instruction scheduling on variable latency functional units of VLIW processors," in *Proc. ACM/IEEE ISED*, Dec. 2011, pp. 307–312.

[13] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in Proc. DATE, 2012,pp. 1257–1262.

[14] V S Kumar Chunduri, G.Sree Lakshmi, "Designand Implementation of Multiplier Using Kcm and Vedic Mathematics by Using Reversible Adder", International Journal of Modern Engineering Research (IJMER) Vol. 3, Issue. 5, Sep - Oct. 2013 pp-3230-3141.

[15] Poornima M, Shivaraj Kumar Patil, Shivu kumar, Shridhar KP,Sanjay H, "Implementation of Multiplier Using Vedic Algorithm", IJITEE, ISSN:-2278-3075,Volume-2,Issue-6,May-2013.

[16] G.vaithiyanathan, s.sivaramakrishnan, k.venkatesan, and s. Jayakumar, "simulation and implementation of vedic multiplier using vhdl code", international journal of scientific & engineering research volume 4, issue 1, january-2013.

[17] Ing-Chao Lin, *Member, IEEE*, Yu-Hung Cho, and Yi-Ming Yang,"Aging-Aware Reliable Multiplier Design With Adaptive Hold Logic", VOL. 23, NO. 3, MARCH 2015.