International Journal on Applications in Electrical and Electronics Engineering
Volume 2: Issue 5:May 2016, pp 1- 5. www.aetsjournal.com ISSN (Online) : 2395 - 3527

Realization of Multiplier Design with Novel Adaptive Hold Logic Using Verilog HDL

G.S Sankari , G.Sharmila

Abstract— Digital multipliers are the most critical arithmetic functional units. The overall performance of the systems depends on the throughput of the multiplier. At the same time, the negative bias temperature instability effect occurs when a transistor is under negative bias, increasing the threshold voltage of the transistor and reducing multiplier speed. A similar phenomenon, positive bias temperature instability occurs when a transistor is under positive bias. Therefore, it is important to design high-performance reliable multipliers. In this paper, aging-aware multiplier design with novel adaptive hold logic (AHL) circuit is proposed. The proposed architecture can be applied to a column or row bypassing multiplier. The experimental results show that, the proposed architecture with 4 × 4 column-bypassing multipliers and 4 × 4 row-bypassing multipliers. This Adaptive Hold Logic Multiplier is implemented using Verilog HDL and Simulated by Modelsim 6.4c and Synthesized by Xilinx 9.1 or Xilinx 13.2. Finally, the Proposed Design is burned in to FPGA Spartan 3.

Index terms — Adaptive hold logic, razor flip flop, reliable multiplier, variable latency.

I. INTRODUCTION

Digital multipliers are the more critical arithmetic functional units in many applications, such as Discrete cosine transform, Fourier transform, and digital filtering. The throughput of these applications depends on multipliers, if the multipliers are too slow, the entire circuit performance will be reduced. In Proposed System an aging-aware reliable multiplier design with novel adaptive hold logic (AHL) circuit is used. The multiplier is based on the variable-latency technique. The AHL circuit is used to achieve reliable operation under the influence of NBTI and PBTI effects. In proposed architecture, 4×4 column-bypassing multipliers and 4×4 row-bypassing multipliers is used.

Traditional circuits use critical path delay as the overall clock cycle of circuit 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, the critical path delay is used as the overall cycle period will result in significant timing waste. Hence, the variable-latency design is proposed to reduce the timing waste of traditional circuits.

The variable-latency design divides the circuit into two parts such as shorter paths and 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 the traditional designs. For example, several variable-latency adders were proposed using the speculation technique with error detection and error recovery. A short path activation function algorithm was proposed to improve the accuracy of the hold logic and optimize the performance of the variable-latency circuit. An instruction scheduling algorithm was proposed to schedule the operations on non uniform latency functional units and improve the performance of Very Long Instruction Word Processors. In addition, the critical paths are divided into two shorter paths that could be unequal and clock cycle is set to the delay of the longer one. These designs were able to reduce timing waste of traditional circuits to improve performance, but they did not consider the aging effect and could not adjust themselves during the runtime.

G.S Sankari , Assistant Professor , Department of ECE , M.A.M College of Engineering & Technology , Trichy, India
G.Sharmila , PG Scholar, Department of ECE , M.A.M College of Engineering & Technology , Trichy, India

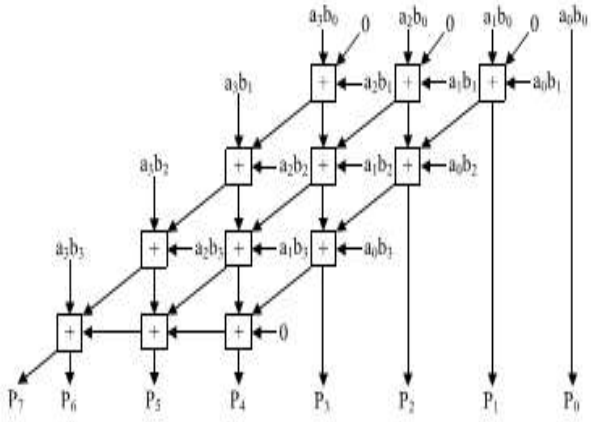

Fig - 1 4×4 normal AM

II. PRELIMINARIES

A. Column-Bypassing Multiplier

A column-bypassing multiplier is an improvement on normal array multiplier (AM). The array multiplier is a fast parallel array multiplier and is shown in Fig.1. The multiplier array consists of (n−1) rows of carry save adder (CSA), in this each row contains (n − 1) full adder (FA) cells. Each full adder in the carry save adder array has two outputs. First, the sum bit goes down. Second, the carry bit goes to the lower left full adder. The last row is a ripple adder for carry propagation.

The full adders in the AM are always active regardless of input states. A low-power column-bypassing multiplier design is proposed in which, the full adder operations are disabled if the corresponding bit in the multiplicand is 0. Fig.2 shows a

4×4 column-bypassing multiplier. Suppose the inputs are 1010_2 * 1111_2, it can be seen that for the FAs in the first and third diagonals, two of the three input bits are 0: the carry bit from its upper right FA and the partial product a_ibi. Therefore, output of the adders in both diagonals is 0, and the output sum bit is simply equal to the third bit, which is sum output of its upper FA.

Hence, the FA is modified to add two tristate gates and one multiplexer. The multiplicand bit a_i can be used as the selector of the multiplexer to decide the output of the FA, and a_i can also be used as selector of the tristate gate to turn off the input path of the FA. If a_i is 0, the inputs of FA are disabled, and sum bit of the current FA is equal to the sum bit from its upper FA, thus the power consumption of the multiplier is reduced. If a_i is 1, the normal sum result is selected.

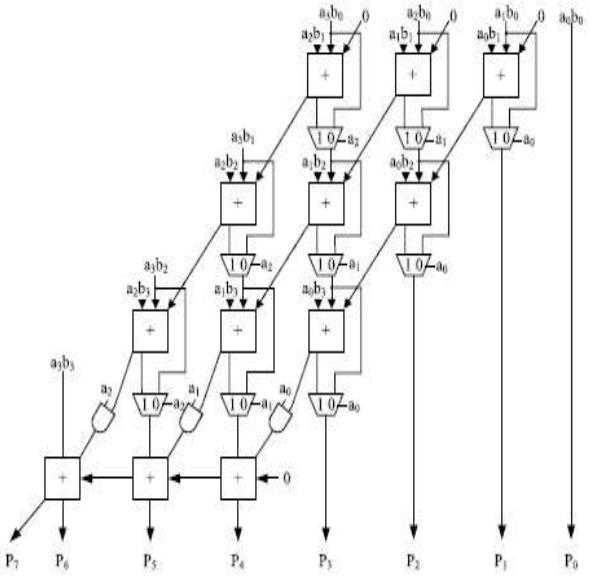

Fig. 2 4×4 column bypassing multiplier

B. Row-Bypassing Multiplier

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

Fig. 3 is a 4 × 4 row-bypassing multiplier. Each input is connected to a FA through a tristate gate. Suppose the inputs are 1111_2 * 1001_2, 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 a_ib_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. Hence, no switching activities occur in the first-row FAs; in return, power consumption is reduced. Similarly, b_2 is 0, and there is 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.

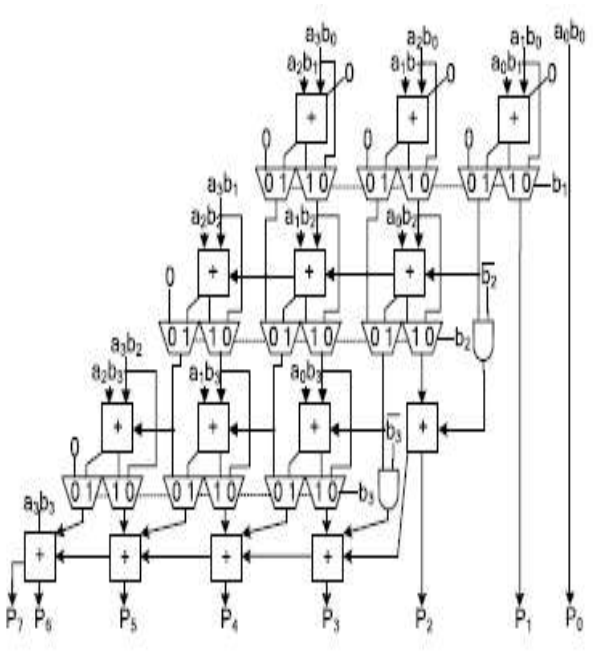

Fig.3 4×4 row bypassing multiplier

C. Variable-Latency Design

Section I mentioned that the variable-latency design was proposed to reduce the timing waste occurring in traditional circuits that use the critical path cycle as an execution cycle period. The basic concept is to execute a shorter path using a shorter cycle and longer path using two cycles. Since most paths execute in a cycle period that is much smaller than the critical path delay, the variablelatency design has smaller average latency.

III. PROPOSED SYSTEM

The Fig. 4 shows our proposed aging-aware multiplier architecture, which includes two *m*-bit inputs (*m* is a positive number), one 2*m*-bit output, one column- or row-bypassing multiplier, 2*m* 1-bit Razor flip-flops, and an AHL circuit.

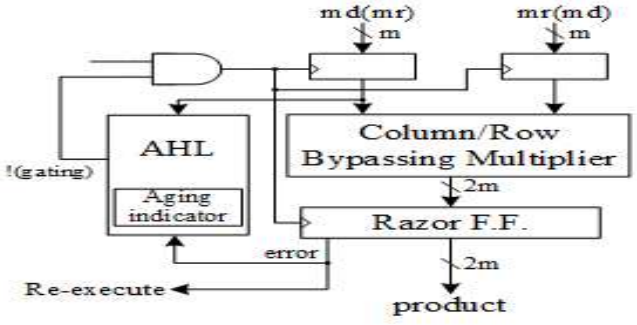

Fig.4 Proposed architecture

In the proposed architecture, the column- and row-bypassing 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. According to the bypassing selection in the

column or row-bypassing multiplier, the input signal of the AHL in the architecture with the column-bypassing multiplier is the multiplicand, whereas that of the row bypassing multiplier is the multiplicator. Razor flip-flops can be used to detect whether timing violations occur before the next input pattern arrives.

Fig.5 shows the details of Razor flip-flops. 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 for 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. Although the re-execution may seem costly, the overall cost is low because the reexecution frequency is low.

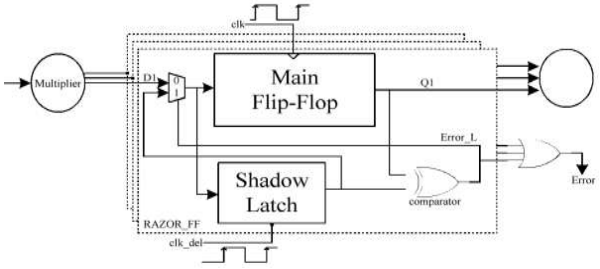

Fig. 5 Razor Flip Flop

The AHL circuit is the key component in the aging ware variable-latency multiplier. Fig. 6 shows the details of the AHL circuit. The AHL circuit contains an aging indicator, two judging blocks, one mux, and one D flip-flop. The aging indicator indicates whether the circuit has suffered significant performance degradation due to the aging effect. The aging indicator is implemented in a simple counter that counts the number of errors over a certain amount of operations and is reset to zero at the end of those operations. If the cycle period is too short, the column- or row-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 and exceed a predefined threshold, it means the circuit has suffered significant timing degradation due to the aging effect, and the aging indicator will output signal 1; otherwise, it will output 0 to indicate the aging effect is still not significant, and no actions are needed.

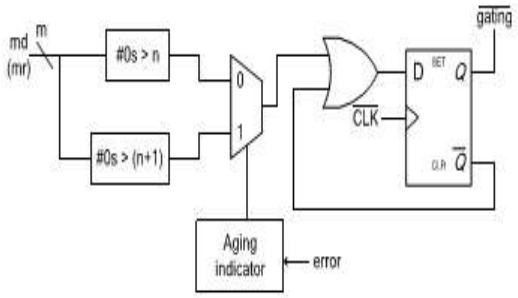

Fig. 6 AHL circuit

The details of the operation of the AHL circuit are as follows: when an input pattern arrives, both judging blocks 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 aging indicator. Then an OR operation is performed between the result of the multiplexer, and the .*Q* signal is used to determine the input of the D flipflop.

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. Note that only a cycle of the input flip-flop will be disabled because the D flip-flop will latch 1 in the next cycle.

The overall flow of our proposed architecture is as follows: when input patterns arrive, the column- or row bypassing multiplier, and the AHL circuit execute simultaneously. According to the number of zeros in the multiplicand (multiplicator), the AHL circuit decides if the input patterns require one or two cycles. If the input pattern requires two cycles to complete, the AHL will output 0 to disable the clock signal of the flip-flops. Otherwise, the AHL will output 1 for normal operations. When the column or row-bypassing multiplier finishes the operation, the result will be passed to the Razor flip-flops. The Razor flip-flops check whether there is the path delay timing violation. If timing violations occur, it means the cycle period is not long enough for the current operation to complete and that the execution result of the multiplier is incorrect. Thus, the Razor flip-flops will output an error to inform the system that the current operation needs to be re-executed using two cycles to ensure the operation is correct. In this situation, the extra re-execution cycles caused by timing violation incurs a penalty to overall average latency. However, our proposed AHL circuit can accurately predict whether the input patterns require one or two cycles in most cases. Only a few input patterns may cause a timing variation when the AHL circuit judges incorrectly. In this case, the extra re-execution cycles did not produce significant timing degradation.

IV. SIMULATION RESULT

The Fig. 7 shows the Block diagram of the aging-aware multiplier.

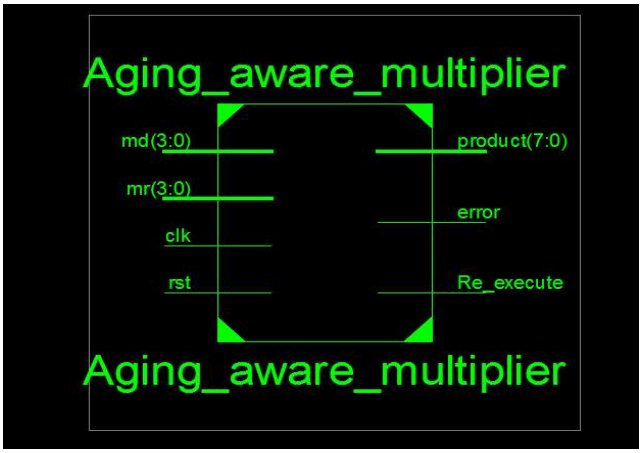

Fig. 7 Block diagram

The Fig. 8 shows the Simulated output waveform of the 4 x 4 aging-aware multiplier using column bypassing multiplier**.**

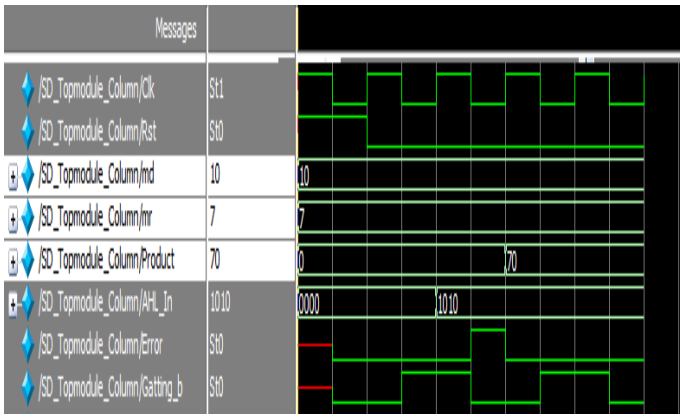

Fig. 8 Simulated output waveform using column bypassing multiplier

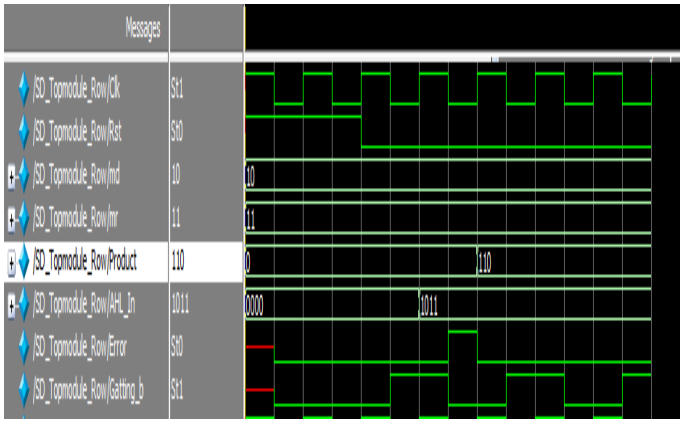

Fig. 9 Simulated output waveform using row bypassing multiplier

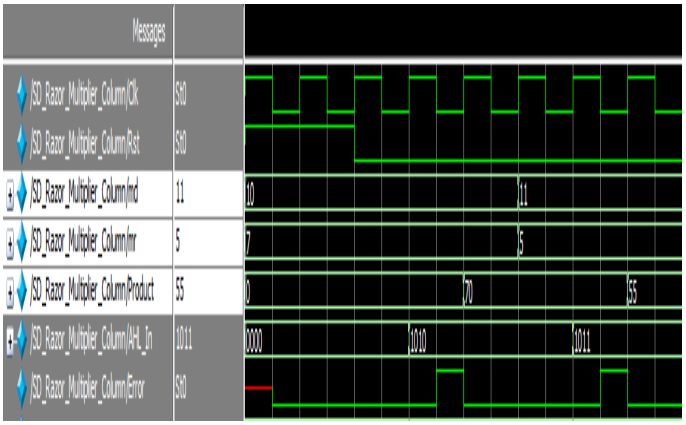

Fig. 10 Output waveform of Razor flipflop with column bypassing multiplier

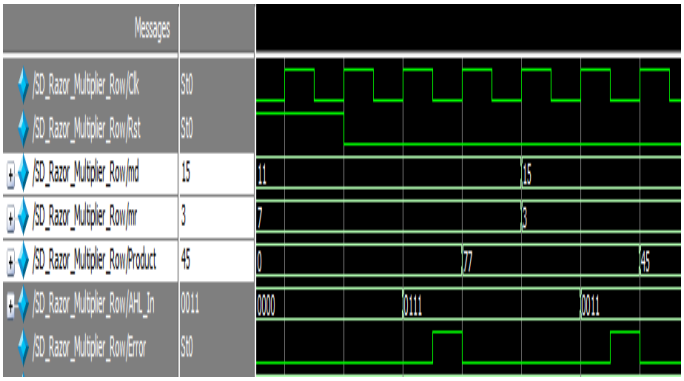

Fig. 11 Output waveform of Razor flips flop with row bypassing multiplier

V. CONCLUSION AND FUTURE WORK

This paper proposed an aging-aware variable-latency multiplier design with the AHL. The multiplier is able to adjust the AHL to mitigate performance degradation due to increased delay. The experimental results show that our proposed architecture with 4×4 column-bypassing multipliers can attain up to good performance improvement compared with the Existing multipliers, respectively. Furthermore, the proposed architecture row-bypassing multipliers can achieve performance improvement compared with Existing multipliers. This AHL Multiplier is done by Verilog Code and simulated using Modelsim 6.4 c and synthesized by Xilinx 9.1. The Hardware Implementation is done in FPGA Spartan Family. The future scope of the paper is implementing the Application Part FIR Filter Design based on the Proposed Multiplier.

REFERENCES

[1] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in Proc. DATE, 2008, pp. 1250–1255.
[2] Calimera, E. Macii, and M. Poncino, "Design techniques for NBTI tolerant power-gating architecture," IEEE Trans. Circuits Syst., Exp. Briefs, vol. 59, no. 4, pp. 249–253, Apr. 2012.
[3] D. Baneres, J. Cortadella, and M. Kishinevsky, "Variable-latency design by function speculation," in Proc. DATE, 2009, pp. 1704–1709.

[4] H. Abrishami, S. Hatami, B. Amelifard, and M. Pedram, "NBTI-aware flip-flop characterization and design," in Proc. 44th ACM GLSVLSI, 2008, pp. 29–34
[5] H.-I. Yang, S.-C. Yang, W. Hwang, and C.-T. Chuang, "Impacts of NBTI/PBTI on timing control circuits and degradation tolerant design in nanoscale CMOS SRAM," IEEE Trans. Circuit Syst., vol. 58, no. 6, pp. 1239–1251, Jun. 2011.
[6] K.-C. Wu and D. Marculescu, "Aging-aware timing analysis and optimization considering path sensitization," in Proc. DATE, 2011, pp. 1–6.
[7] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in Proc. DATE, 2012, pp. 1257–1262.
[8] M. Basoglu, M. Orshansky, and M. Erez, "NBTI-aware DVFS: A new approach to saving energy and increasing processor lifetime," in Proc. ACM/IEEE ISLPED, Aug. 2010, pp. 253–258.
[9] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in Proc. ACM/IEEE DAC, Jun. 2007, pp. 370–375.
[10] S. Zafar, A. Kumar, E. Gusev, and E. Cartier, "Threshold voltage instabilities in high-k gate dielectric stack," IEEE Trans. Device Mater. Rel., vol. 5, no. 1, pp. 45–64, Mar. 2005.