A Low Complexity and Highly Robust Multiplier Design using Adaptive Hold Logic

Vaishak Narayanan[1] Mr.G.RajeshBabu[2]
[1]PG Scholar [2]Assistant Professor
[1,2]Department of Electronics & Communication Engineering
[1,2]SVS College of Engineering

Abstract— This project deals with Aging Aware Reliable multiplier design with adaptive hold logic, Digital multipliers is one of the major unit in arithmetic functional units. The overall performance of the AHL systems depends on the throughput of the multiplier. Even though the negative bias temperature instability effect occurs when a pMOS transistor is under negative bias voltage, it leads in the increasing threshold voltage of the pMOS transistor and it will reduce the speed of the multiplier design. Hence like this the positive bias temperature instability occurs when an nMOS transistor is under positive bias. These two effects degrade transistor speed, and in the long term the system may fail due to timing violations. Hence it is important to design reliable high performance multipliers. In this project we propose an aging aware multiplier design with an adaptive hold logic circuit. By this the multiplier is able to provide higher throughput through the variable latency and can adjust adaptive hold logic circuit to mitigate performance degradation that is due to aging effect.

Key words: Multiplier, Adaptive Hold Logic, Razor Flipflop, Xilinx

I. INTRODUCTION

The multipliers are among the most critical arithmetic functional units in many applications, such as the digital filtering, Fourier transforms and discrete cosine transforms. The throughput of these applications indicated here, depends on multiplier. Hence if the multipliers are too slow, then the performance of entire circuits will be reduced. The negative bias temperature instability (NBTI) occurs when pMOS transistor is under negative bias condition which is (Vgs= -Vdd). In this case, the interaction between inversion layer holes and hydrogen- passivated Si atoms breaks the Si–H bond generated during the oxidation process, generating H or H2 molecules. The accumulated interface traps between silicon and the gate oxide interface result in increased threshold voltage (Vth), reducing the circuit switching speed.

The corresponding effect on an nMOS transistor is positive bias temperature instability (PBTI), which occurs when an nMOS transistor is under positive bias. A traditional method to mitigate the aging effect is over the design including such things as guard banding and gate over sizing however, this approach can be very pessimistic and area and power inefficient. To overcome the problem, many NBTI-aware methodologies have been proposed.

The mapping technique was proposed in to guarantee the performance of the circuit for the whole lifetime. In an NBTI-aware sleep transistor was designed to decrease the aging effects on pMOS sleep-transistors, and the lifetime stability of the power-gated circuits under consideration was improved.

The traditional circuits mostly uses the critical path delay as the overall circuit clock cycle in order to perform the operation 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 such as shorter paths and longer paths.

II. LITERATURE SURVEY

A. Aging-Aware Reliable Multiplier Design with Adaptive Hold Logic

In this paper, we propose an aging aware reliable multiplier design with novel adaptive hold logic (AHL) circuit. The multiplier is based on the variable-latency technique and can adjust the AHL circuit to achieve reliable operation under the influence of NBTI and PBTI effects.

- Novel variable-latency multiplier architecture with an AHL circuit. The AHL circuit can decide whether the input patterns require one or two cycles and can adjust the judging criteria to ensure that there is minimum performance degradation after considerable aging occurs;
- An aging-aware reliable multiplier design method that is suitable for large multipliers. Although the experiment is performed in 16- and 32-bit multipliers, our proposed architecture can be easily extended to large designs.

B. Low-Cost Low-Power Bypassing-Based Multiplier Design

The low power multiplier is based on the reducing the addition operations in a low-power bypassing-based multiplier. A low-cost and low-power bypassing-based multiplier is proposed in this design. Compared with row-bypassing multiplier, column-by passing multiplier and2-dimensional bypassing-based multiplier for 20 tested examples, the experimental results show that our proposed low-cost low power multiplier saves 15.1% of hardware cost and reduces29.6% of the power dissipation on the average for 4x4, 8x8 and16x16 multipliers.

Static power dissipation from the leakage current and the consumption is proportional to the number of the used transistors. On the other hand, dynamic power dissipation is from the switching transient current and the consumption is obtained from the charging and discharging of the load capacitances. In general, the average dynamic power dissipation for a CMOS.

III. MULTIPLIER DESIGN

First, a low-power row-bypassing multiplier is taken as a design and its waveforms are verified. It 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. Figure.2.1 is a 4 × 4 row-bypassing multiplier. Each input is connected to an FA through a tristate gate.

A. 4X4 Braun Mulitplier

Fig 1 indicates the braun multiplier design with row bypassing. Based on the simplification of the incremental adders and half adders instead of full adders in an array multiplier, a low-power multiplier design with row and column bypassing is designed .

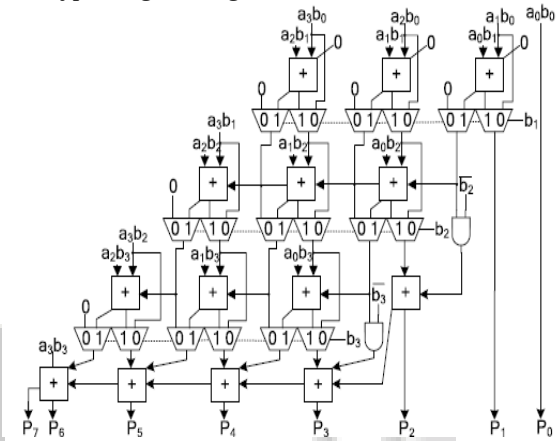

Fig. 1: 4X4 Braun Multiplier with row bypassing

B. Column Bypassing Mulitplier

A column-bypassing multiplier is an improvement on the normal array multiplier (AM). The AM is a fast parallel AM and is shown in Figure.2.2 The multiplier array consists of (n−1) rows of carry save adder (CSA), in which each row contains (n− 1) full adder (FA) cells. Each FA in the CSA array has two outputs: 1) the sum bit goes down and 2) the carry bit goes to the lower left FA.

The last row is a ripple adder for carry propagation. The FAs in the AM are always active regardless of input states. In a low-power column-bypassing multiplier design is proposed in which the FA operations are disabled if the corresponding bit in the multiplicand is 0. Figure.2.2 shows a 4×4 column-bypassing multiplier. Suppose the inputs are 10102 * 11112, 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 *aibi.*

Therefore, the output of the adders in both diagonals is 0, and the output sum bit is simply equal to the third bit, which is the sum output of its upper FA. Hence, the FA is modified to add two tristate gates and one multiplexer. The multiplicand bit *ai* can be used as the selector of the multiplexer to decide the output of the FA, and ai can also be used as the selector of the tristate gate to turn off the input path of the FA. If ai is 0, the inputs of FA are disabled, and the sum bit of the current FA is equal to the sum bit from its upper FA, thus reducing the power consumption of the multiplier. If ai is 1, the normal sum result is selected. More details for the column-bypassing multiplier can be found in.

C. Variable Latency Design

Here 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 variable-latency design has smaller average latency. For example, Figure. 3.1 is an 8-bit variable-latency ripple carry adder (RCA). *A*8–*A*1, *B*8–*B*1 is 8-bit inputs, and *S*8–*S*1 are the outputs. Supposing the delay for each FA is one, and the maximum delay for the adder is 8.

Through simulation, it can be determined that the possibility of the carry propagation delay being longer than 5 is low. Hence, the cycle period is set to 5, and hold logic is added to notify the system whether the adder can complete the operation within a cycle period. Figure.3.1 also shows the hold logic that is used in this circuit. The function of the hold logic is (*A*4 XOR *B*4*)* (*A*5 XOR *B*5*)*. If the output of the hold logic is 0, i.e., *A*4 = *B*4 or *A*5 = *B*5, either the fourth or the fifth adder will not produce a carryout. Hence, the maximum delay will be less than one cycle period. When the hold logic output is 1, this means that the input can activate paths longer than 5, so the hold logic notifies the system that the current operation requires two cycles to complete.

Two cycles are sufficient for the longest path to complete (5 * 2 is larger than 8). The performance improvement of the variable-latency design can be calculated as follows: if the possibility of each input being 1 is 0.5, the possibility of (*A*4 XOR *B*4*)*(*A*5 XOR *B*5*)* being 1 is 0.25.

The average latency for the variable-latency design is 0.75∗5+0.25∗10 = 6.25. Compared with the simple fixed-latency RCA, which has an average latency of 8, the variable-latency design can achieve a 28% performance improvement. Figure.3.1 shows the path delay distribution of a 16 × 16 AM and for both a traditional column bypassing and traditional row-bypassing multiplier with 65 536 randomly chosen input patterns. All multipliers execute operations on a fixed cycle.

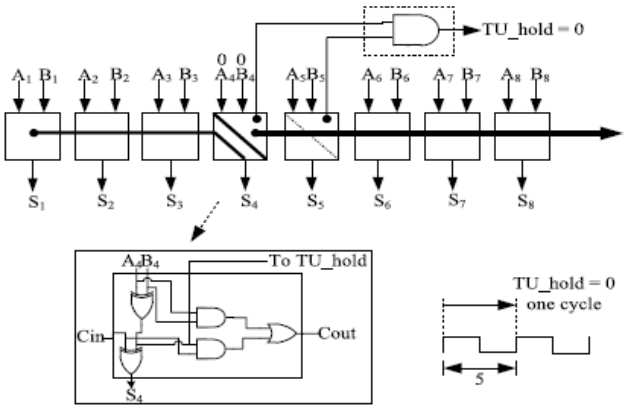

Fig. 2: bit RCA with a hold logic circuit

The maximum path delay is 1.32 ns for the AM,1.88 ns for the column-bypassing multiplier, and 1.82 ns for the row-bypassing multiplier. It can be seen that for the AM, more than 98% of the paths have a delay of <0.7 ns. Moreover, more than 93% and 98% of the paths in the FLCB and row-bypassing multipliers present a delay of

<0.9 ns, respectively. Hence, using the maximum path delay for all paths will cause significant timing waste for shorter paths, and redesigning the multiplier with variable latency can improve their performance. Another key observation is that the path delay for an operation is strongly tied to the number of zero's in the multiplicands in the column-bypassing multiplier.

Figure.3.2 shows the delay distribution of the 16×16 column-bypassing multiplier under three different numbers of zero's in the multiplicands: 1) 6, 2) 8, and 3) 10. Three thousand randomly selected patterns are used in each experiment. It can be seen as the number of zeros in the multiplicands increases, delay distribution is left shifted, and average delay is reduced. The reason for this is the multiplicand is used as the select line for column-bypassing multipliers, and if more zeros exist in the multiplicand, more FAs will be skipped, and the sum bit from the upper FA is passed to the lower FA, reducing the path delay. Row-bypassing multipliers .However, because the results are similar, they are not shown to avoid duplications.

For a row-bypassing multiplier, the multiplications are used to determine whether a pattern needs one cycle or two cycles to complete an operation because the multiplication is used as the select line. This makes the column-bypassing multiplicand and row-bypassing multiplier excellent candidates for the variable latency design since we can simply examine the number of zeros in the multiplicand or multiplication to predict whether the operation requires one cycle or two cycles to complete.

IV. PROPOSED DESIGN

A. *Proposed Architecture*

The proposed aging-aware reliable multiplier design is considered in row and column bypassing architecture.. It introduces the overall system architecture and the functions of each component and also describes how to design Adaptive Hold Logic that adjusts the circuit when significant aging occurs. Figure 4.1 shows our proposed aging-aware multiplier architecture, which includes two m-bit inputs (m is a positive number), one 2m-bit output, one column- or row-bypassing multiplier, 2m 1-bit Razor flip-flops and an Adaptive Hold Logic circuit.

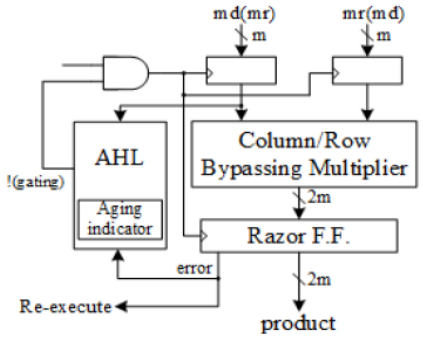

Fig. 3: Proposed Architecture

The inputs of the row-bypassing multiplier are the symbols in the parentheses. 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 zero's and ones in the multiplicator and multiplicand follows a normal distribution, as shown in Figures. 4.2 And 4.3 Therefore, using the number of zeros or ones as the judging criteria results in similar outcomes. Hence, the two aging-aware multipliers can be implemented using similar architecture, and the difference between the two bypassing multipliers lies in the input signals of the AHL.

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.

B. *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 execution frequency is very low. More details for the Razor flip-flop can be found in.

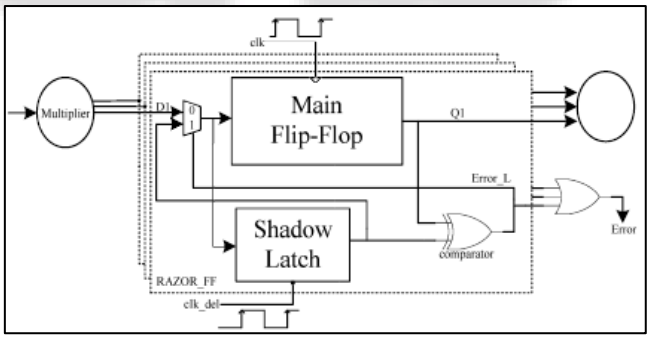

Fig. 4: Razor Flip flops

The AHL circuit is the key component in the aging-ware variable- latency multiplier. Figure.4.5 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.

Otherwise, it will output 0 to indicate the aging effect is still not significant, and no actions are needed. The first judging block in the AHL circuit will output 1 if the number of zeros in the multiplicand (multiplicator for the row-bypassing multiplier) is larger than n (n is a positive number, and the second judging block in the AHL circuit will output 1 if the number of zero's in the multiplicand (multiplicator) is larger than $n + 1$.

They are both employed to decide whether an input pattern requires one or two cycles, but only one of them will be chosen at a time. In the beginning, the aging effect is not significant, and the aging indicator produces 0, so the first judging block is used. After a period of time when the aging effect becomes significant, the second judging block is chosen. Compared with the first judging block, the second judging block allows a smaller number of patterns to become one-cycle patterns because it requires more zeros in the multiplicand (multiplicator).

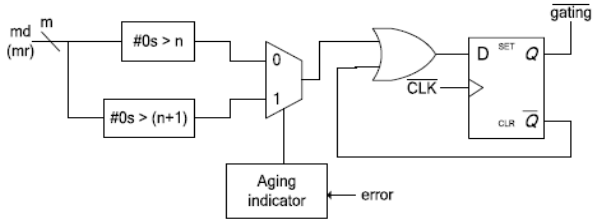

Fig. 5: AHL diagram

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 flip-flop.

When the pattern requires one cycle, the output of the multiplexer is 1the 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 1in 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 (multiplication), 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.

V. SIMULATION RESULTS

Existing	Minimum period	Minimum input arrival time before clock	Maximum output required time after clock	Maximum combinational path delay
Row	10.001ns	2.671ns	34.564ns	12.823ns
Column	10.001ns	2.671ns	27.850ns	12.593ns

Table 1: Existing Delay Analysis

Proposed	Minimum period	Minimum input arrival time before clock	Maximum output required time after clock	Maximum combinational path delay
Row	6.237ns	2.671ns	27.850ns	10.333ns
Column	6.237ns	2.841ns	34.567ns	11.883ns

Table 2: Proposed Delay Analysis

The multiplier is able to adjust the Adaptive Hold Logic and to mitigate performance degradation due to increased delay. The proposed architecture with 16×16 and 32×32 column-bypassing multipliers can attain up to 21.5063% and 5.803% performance improvement compared with the 16×16 and 32×32 FLCB multipliers, respectively. Furthermore, our proposed architecture with the 16×16 and 32×32 row-bypassing multipliers can achieve up to 80.17% and 69.40% performance improvement compared with the 16 × 16 and 32 × 32 FLRB multipliers.

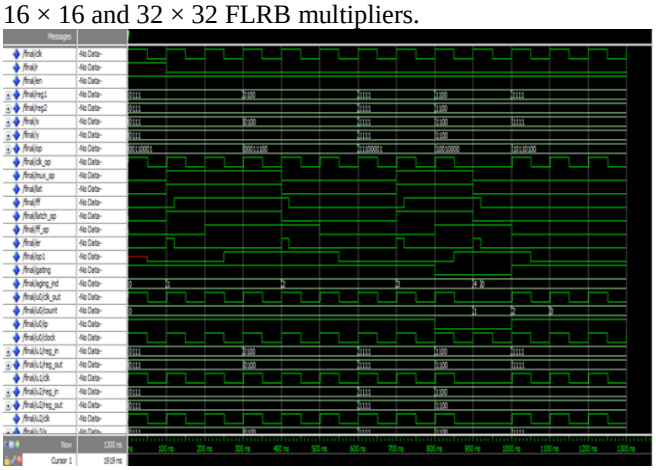

Fig. 6: Simulated output wave form 16 and 32 column bypassing

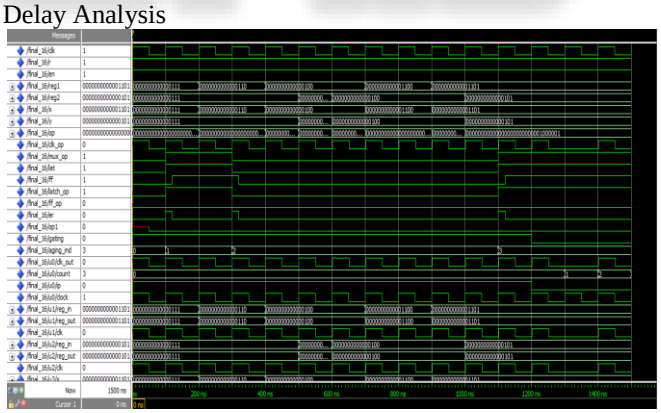

Fig. 7: Simulated out put wave form 16 and 32 row bypassing

	Existing	Proposed
Power (mW)	204	180

Table 3: Power Comparison

VI. CONCLUSION

The proposed design is an aging aware variable latency multiplier design with the Adaptive Hold Logicwith minimum power consumption. The conmparison table shows the power 180mW for proposed design when compared with the existing 12.5% difference in th overall system performance in power level. The multiplier is able to adjust the AHL to mitigate performance degradation due to

increased delay The future work can be implemented in adaptive filters and in the controller unit.

REFERENCES

[1] S. Zafar et al., "A comparative study of NBTI and PBTI (charge trapping) in SiO2/HfO2 stacks with FUSI, TiN, Re gates," in Proc. IEEE Symp. VLSI Technol. Dig. Tech. Papers, 2006.
[2] S. Zafar, A. Kumar, E. Gusev, and E. Cartier, "Threshold voltage instabilities in high-k gate dielectric stacks," IEEE Trans. Device Mater. Rel., vol. 5, no. 1, pp. 45–64, Mar. 2005.
[3] 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, Jun. 2011.
[4] R. Vattikonda, W. Wang, and Y. Cao, "Modelling and minimization of pMOS NBTI effect for robust nanometre design," in Proc. ACM/IEEE DAC, Jun. 2004.
[5] 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
[6] 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.
[7] Calimera, E. Macii, and M. Poncino, "Design techniques for NBTI Tolerant power-gating architecture," IEEE Trans. Circuits Syst., Exp. Apr. 2012.
[8] K.-C. Wu and D. Marculescu, "Joint logic restructuring and pin reordering Against NBTI-induced performance degradation," in Proc. DATE, 2009, pp. 75–80.
[9] Y. Lee and T. Kim, "A fine-grained technique of NBTI-aware voltage scaling and body biasing for standard cell based designs," in Proc. ASPDAC, 2011, pp. 603–608.

