Efficient Adaptive Hold Logic Aging-Aware Reliable Multiplier Design using Verilog HDL

P. Raviteja
M.Tech (DECS)
Department of ECE
Vishnu Institute of Technology
(VIT), Bhimavaram
Andhra Pradesh, India

B. V. V. Satyanarayana
Assistant Professor
Department of ECE
Vishnu Institute of Technology
(VIT), Bhimavaram
Andhra Pradesh, India

D. Durga Prasad
Assistant Professor
Department of ECE
Vishnu Institute of Technology
(VIT), Bhimavaram
Andhra Pradesh, India

ABSTRACT
Digital multipliers are among the maximum essential arithmetic purposeful devices. The average performance of these systems relies upon at the throughput of the multiplier. In the meantime, the negative bias temperature instability impact occurs while a pMOS transistor is underneath negative bias (Vgs = −Vdd), increasing the threshold voltage of the pMOS transistor, and reducing multiplier pace. A similar phenomenon, positive bias temperature instability, happens when an nMOS transistor is underneath positive bias. Each effect degrade transistor pace and in the long term, the device may also fail due to timing violations. Therefore, it is essential to design dependable high-overall performance multipliers. In this paper, suggest an aging-aware multiplier model with a novel adaptive hold logic (AHL) circuit. The multiplier is able to provide higher throughput through the variable latency and may modify the AHL circuit to mitigate overall performance degradation this is because of the aging effect. Furthermore, the proposed structure can be applied to a Pre-Encoded NR4SD Multiplier.

Keywords
Adaptive hold logic (AHL), Positive bias temperature instability (PBTI), Negative bias temperature instability (NBTI), Reliable multiplier

1. INTRODUCTION
Digital multipliers are amongst the maximum essential arithmetic purposeful units in many applications, together with the discrete cosine transforms, Fourier transform, and digital filtering. The throughput of those applications depends on multipliers, and if the multipliers are too gradual, the performance of complete circuits will be reduced. Moreover, negative bias temperature instability (NBTI) happens while a pMOS transistor is beneath negative bias (Vgs = −Vdd). On this situation, the interaction between inversion layer holes and hydrogen-passivated Si atoms breaks the Si–H bond generated at some stage in the oxidation process, producing H or H2 molecules. When those molecules diffuse away, interface traps are left. The accrued interface traps between silicon and the gate oxide interface result in improved threshold voltage (Vth), decreasing the circuit switching speed. When the biased voltage is eliminated, the opposite response occurs, reducing the NBTI impact. However, the reverse response does not get rid of all the interface traps generated for the duration of the strain segment, and Vth is extended inside the long term. Therefore, it is important to design a dependable high-performance multiplier. The corresponding effect on an nMOS transistor is positive bias temperature instability (PBTI), which happens whilst an nMOS transistor is beneath positive bias. As compared with the NBTI impact, the PBTI impact is much smaller on oxide/polygate transistors, and consequently is normally left out. However, for high-*k*/metal-gate nMOS transistors with big rate trapping, the PBTI effect cannot be neglected. In reality, it has been shown that the PBTI impact is extra enormous than the NBTI impact on 32-nm high-*k*/metal-gate processes [1] – [4].

A traditional method to mitigate the aging effect is overdesign [5], [6], including such matters as guard-banding and gate oversizing; however, this approach can be very pessimistic and area and power inefficient. To keep away from this problem, many NBTI-aware methodologies were proposed. An NBTI-aware generation mapping technique changed into proposed in [7] to guarantee the overall performance of the circuit during its lifetime.

In [8], an NBTI-aware sleep transistor became designed to lessen the ageing results on pMOS sleep-transistors, and the lifetime balance of the power-gated circuits under consideration changed into improved. Wu and Marculescu [9] proposed a joint logic restructuring and pin reordering approach, that's based totally on detecting functional symmetries and transistor stacking effects. Additionally, they proposed an NBTI optimization technique that taken into consideration path sensitization [12]. In [10] and [11], dynamic voltage scaling and body-basing techniques were proposed to reduce power or extend circuit life. Those techniques, however, require circuit change or do not offer optimization of unique circuits.

Traditional circuits use important path delay as the overall circuit clock cycle with the intention to perform effectively. But, the probability that the critical paths are activated is low. In most cases, the path delay is shorter than the critical route. For these noncritical paths, the use of the crucial path delay as the general cycle length will bring about good sized timing waste. Therefore, the variable-latency design becomes 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 effectively in one cycle, while longer paths need two cycles to execute. Whilst shorter paths are activated often, the common latency of variable-latency designs is higher than that of traditional designs. As an instance, numerous variable-latency adders were proposed using the hypothesis approach with blunders detection and healing [13] – [15]. A short path activation function set of rules become proposed in [16] to enhance the accuracy of the hold logic and to optimize the performance of the variable-latency circuit. A training scheduling set of rules changed into proposed in [17] to schedule the operations on non-uniform latency practical

devices and enhance the overall performance of Very long instruction word processors. In [18], variable latency pipelined multiplier architecture with a booth algorithm became proposed. In [19], process-variant tolerant structure for mathematics units was proposed, where the impact of process variation is considered to growth the circuit yield. Similarly, the crucial paths are divided into two shorter paths that could be unequal and the clock cycle is about to the delay of the longer one. Those research designs have been capable of lessen the timing waste of conventional circuits to improve performance, but they did not take into account the aging effect and couldn't adjust themselves for the duration of the runtime. A variable-latency adder layout that considers the aging effect turned into proposed in [20] and [21]. However, no variable-latency multiplier design that considers the aging effect and can alter dynamically has been accomplished.

1.1. Paper Contribution

In this paper, advise an aging aware reliable multiplier design with novel adaptive hold logic (AHL) circuit. The multiplier is based at the variable-latency method and might alter the AHL circuit to gain reliable operation below the have an impact on of NBTI and PBTI results. To be precise, the contributions of this paper are summarized as follows:

1) Novel variable-latency multiplier structure with an AHL circuit. The AHL circuit can determine whether or not the enter styles require one or two cycles and can regulate the judging criteria to make certain that there is minimum according performance degradation after considerable aging occurs;

2) Complete analysis and contrast of the multiplier's overall performance under different skip numbers to reveal the effectiveness of our proposed structure;

3) An aging-aware reliable multiplier method that is appropriate for huge multipliers. Despite the fact that the test is accomplished in 4-, 8-, 16- and 32-bit multipliers, our proposed architecture can be effortlessly prolonged to big designs;

4) The experimental outcomes display that our proposed structure with the 16×16 and 32×32 Non-Redundant radix-4 Signed Digit (NR4SD) multipliers can acquire exceptional overall performance development in comparison with the 16×16 and 32×32 Non-Redundant radix-four Signed-Digit (NR4SD) multipliers.

The paper is prepared as follows. Segment 2 introduces the overture of the Non-Redundant radix-four Signed Digit (NR4SD) multiplier and NBTI/PBTI models. Segment 3 details the aging-aware reliable multiplier primarily based at the Non-Redundant radix-four Signed-Digit (NR4SD) multiplier. The experimental results and comparisons are supplied in Segment 4. Segment 5 concludes this paper.

2. OVERTURE

2.1. Modified Booth Algorithm

Modified Booth (MB) could be a redundant radix-4 coding technique [22], [23]. Considering the multiplication of the 2's complement numbers A, B, all consisting of n=2k bits, B is painted in MB type as:

$$B = < b_{n-1} \ldots b_0 >2's = -b_{2k-1}2^{2k-1} + \sum_{i=0}^{2k-2} b_i 2^i$$

$$= <b_{k-1}^{MB} \ldots b_0^{MB}>_{MB} = \sum_{j=0}^{k-1} b_j^{MB} 2^{2j} \quad (1)$$

Digits $b_j^{MB} \in \{-2,-1, 0, +1, +2\}$, $0 \le j \le k-1$, are formed as follows:

$$b_j^{MB} = -2b_{2j+1} + b_{2j} + b_{2j-1} \quad (2)$$

In which $b_{-1} = 0$. Every MB digit is represented by using the bits s, one and two (table 1). The bit s suggests if the digit is negative (s=1) or positive (s=0). One indicates if the absolute fee of a digit equals 1 (one=1) or now not (one=0). Two suggests if absolutely the fee of a digit equals 2 (two=1) or now not (two=0). The use of these bits, to calculate the MB digits b_j^{MB} as follows:

$$b_j^{MB} = (-1)^{sj}.(one_j + 2two_j). \quad (3)$$

Equations (4) form the MB encoding signals.

$s_j = b_{2j+1}$; $one_j = b_{2j-1} \oplus b_{2j}$;

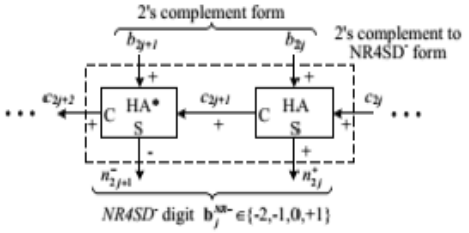
$two_j = (b_{2j+1} \oplus b_{2j}) \wedge \sim(one_j)$:

Table 1. Modified Booth Encoding

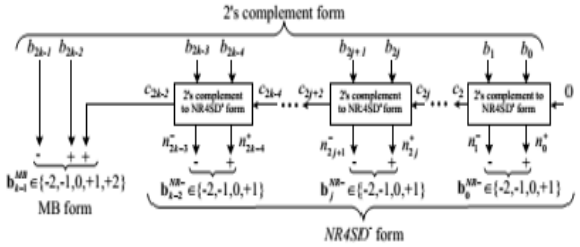
b_{2j+1}	b_{2j}	b_{2j-1}	b_j^{MB}	s_j	one_j	Two_j
0	0	0	0	0	0	0
0	0	1	+1	0	1	0
0	1	0	+1	0	1	0
0	1	1	+2	0	0	1
1	0	0	-2	1	0	1
1	0	1	-1	1	1	0
1	1	0	-1	1	1	0
1	1	1	0	1	0	0

2.2. Non-Redundant Radix-4 Signed Digit Algorithm

In this section have a tendency to gift the Non-Redundant radix-4 Signed-Digit (NR4SD) encoding technique. As in MB shape, the range of partial products is reduced to half of. While encoding the 2's complement quantity B, digits b_j^{NR-} take considered one of 4 values:$\in$ {-2, -1, 0, +1} or b_j^{NR+} take one of four values: $\in$ {-1, 0, +1, +2} at the NR4SD$^-$ or NR4SD$^+$ algorithmic program, severally. Only 4 specific values are used and no longer five as in MB set of rules, which results in $0 \le j \le k-2$. As need to cowl the dynamic range of the 2's complement form, the maximum huge digit is MB encoded (i.e.,$b_{k-1}^{MB} \in$ {-2,-1, 0, +1, +2).The NR4SD$^-$ and NR4SD$^+$ encoding algorithms are illustrated in detail in Fig. 1 and 2, respectively.

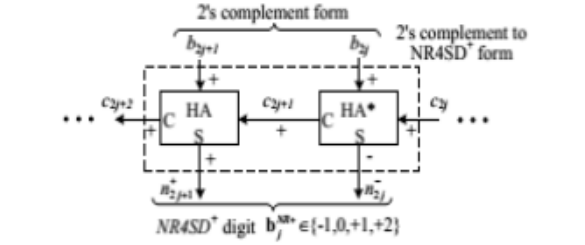


(a)

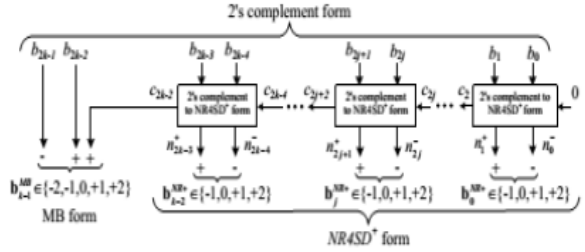


(b)

Fig 1: Block Diagram of the NR4SD⁻ Encoding Scheme at The (a) Digit and (b) Word Level.



(a)



(b)

Fig 2: Block Diagram of the NR4SD⁺ Encoding Scheme at the Word Level.

2.2.1 NR4SD⁻ Algorithm

Step 1: contemplate the initial values j = 0 and c_0=0.

Step 2: Calculate the convey c_{2j+1} and the sum n^+_{2j} of a half of Adder (HA) with inputs b_{2j} and c_{2j} (Fig. 1a).

$$c_{2j+1} = b_{2j} \wedge c_{2j};\ n^+_{2j} = b_{2j} \oplus c_{2j} \quad (4)$$

Step3: Calculate the positively signed carry c_{2j+2} (+) and therefore the negatively signed sum n^-_{2j+1} (-) of a Half Adder* (HA*) with inputs b_{2j+1} (+) and c_{2j+1} (+) (Fig.1a). The outputs c_{2j+2} and n^-_{2j+1} of the HA* relate to its inputs as follows:

$2c_{2j+2} - n^-_{2j+1} = b_{2j+1} + c_{2j+1}$:

The following Boolean equations summarize the HA* operation:

$c_{2j+2} = b_{2j+1} \wedge c^-_{2j+1}$, $n^-_{2j+1} = b_{2j+1} \oplus c_{2j+1}$.

Step 4: Calculate the value of the b_j^{NR-} digit.

$$b_j^{NR-} = -2n^-_{2j+1} + n^+_{2j}. \quad (5)$$

Equation (5) shows results from the n^-_{2j+1} is negatively signed and n^+_{2j} is positively signed.

Step 5: j: = j + 1.

Step 6: If (j < k-1), then forward to Step 2. If (j = k-1), encode that the most significant value based on the MB technique and considering the three consecutive bits to be b_{2k-1}, b_{2k-2} and c_{2k-2} (Fig. 1b). If (j = k), stop.

Table 2 shows how the NR4SD⁻ digits are formed. Equations (6) show how the NR4SD⁻ encoding signals one^+_j, one^-_j and two^-_j of Table 2 are generated.

$$one^+_j = \sim (n^-_{2j+1}) \wedge n^+_{2j},\ one^-_j = n^-_{2j+1} \wedge n^+_{2j},$$
$$two^-_j = n^-_{2j+1} \wedge n^+_{2j} \quad (6)$$

The minimum and maximum limits of the dynamic range in the NR4SD⁻ form are $-2^{n-1} - 2^{n-3} - 2^{n-5} - \ldots - 2 < -2^{n-1}$ and $2^{n-1} + 2^{n-4} + 2^{n-6} + \ldots\ldots + 1 > 2^{n-1}-1$. The NR4SD⁻ form has larger dynamic range than the 2's complement form.

2.2.2 NR4SD⁺ Algorithm

Step 1: contemplate the initial values j = 0 and c_0=0.

Step 2: Calculate the carry positively signed value c_{2j+1} (+) and the negatively signed value sum n^-_{2j} (-) of a HA* with inputs b_{2j} (+) and c_{2j} (+) (Fig. 2a). The carry c_{2j+1} and the sum n^-_{2j} of the HA* relate to its inputs as follows:

$2c_{2j+1} - n^-_{2j} = b_{2j} + c_{2j}$,

The outputs of the HA* are can calculate at gate level in the following equations as:

$c_{2j+1} = b_{2j} \vee c_{2j}$, $n^-_{2j} = b_{2j} \oplus$ c2j ,

Step 3: Calculate the carry c_{2j+2} and the sum n^+_{2j+1} of a HA with inputs b_{2j+1} and c_{2j+1}

$c_{2j+2} = b_{2j+1} \wedge c_{2j+1}$, $n^+_{2j+1} = b_{2j+1} \vee c_{2j+1}$,

Step 4: Calculate the value of the b_j^{NR+} digit.

$$b_j^{NR+} = 2n^+_{2j+1} - n^-_{2j} \quad (7)$$

Equation (7) results from the n^+_{2j+1} is positively signed and n^-_{2j} is negatively signed.

Step 5: j := j + 1.

Step 6: If (j < k-1), go to Step 2. If (j = k-1), encode the most significant value according to MB technique and considering the three consecutive bits to be b_{2k-1}, b_{2k-2} and c_{2k-2} (Fig. 2b). If (j = k), stop.

Table 3 shows how the NR4SD⁺ digits are formed. Equations (8) show how the NR4SD⁺ encoding signals one^+_j, one^-_j and two^+_j of Table 4 are generated.

$$one^+_j = n^+_{2j+1} \wedge n^-_{2j};$$
$$one^-_j = n^+_{2j+1} \wedge n^-_{2j};$$
$$two^+_j = n^+_{2j+1} \wedge n^-_{2j}: \quad (8)$$

The minimum and maximum values of the dynamic range in the NR4SD⁺ form are $-2^{n-1} - 2^{n-4} - 2^{n-6} - \ldots\ldots -1 < -2^{n-1}$ and $2^{n-1}+2^{n-3}+2^{n-5}+\ldots +2 > 2^{n-1}-1$.

Table 2. Numerical Examples of the Encoding Techniques

2's Complemen	10000000	10011010	01011001	01111111
Integer	-128	-102	+89	+127
NR4SD⁻	$\bar{2}000$	$\bar{1}2\bar{1}2$	$22\bar{2}1$	$200\bar{1}$
NR4SD⁺	$\bar{2}000$	$\bar{2}122$	1121	$200\bar{1}$

As determined in the NR4SD$^-$ encoding technique, the NR4SD$^+$ type has large dynamic variety than the two's complement form. Thinking about the eight-bit 2's complement variety N, Table 2 shows the restriction values -28 = -128, 28 - 1 =127, and two ordinary values of N, and presents the MB, NR4SD$^-$ and NR4SD$^+$ digits that end result when making use of the corresponding encoding strategies to every cost of N taken into consideration. A bar above the negatively signed digits in order to distinguish them from the positively signed ones.

2.3. Pre-Encoded NR4SD Multipliers Design

The device design for the pre-encoded NR4SD multipliers is bestowed in Fig. 6. Two bits at the moment are stored in ROM: n^-_{2j+1}, n^+_{2j} (Table 3) for the NR4SD$^-$ or n^+_{2j+1}, n^-_{2j} (Table 4) for the NR4SD$^+$ kind. On this manner, can reduce the storage requirement to n+1 bits consistent with coefficient even as the corresponding memory required for the pre-encoded MB scheme is 3n/2 bits per coefficient. Consequently, the quantity of saved bits is identical to that of the conventional MB design, besides for the maximum widespread digit that wishes a further bit as its far MB encoded. Compared to the pre-encoded MB multiplier, in which the MB encoding blocks are ignored, the pre-encoded NR4SD multipliers need additional hardware to generate the values of (6) and (8) for the NR4SD$^-$ and NR4SD$^+$ form, respectively. The NR4SD encoding blocks of Fig. 4 put into effect the circuitry of Fig. 5.

Partial product is now given by the relation:

$$P = A.B = COR + \sum_{j=1}^{k-1} PP_j 2^{2J} \quad (9)$$

$$COR = \sum_{j=0}^{k-1} C_{in,j} 2^{2J} + 2^n (1 + \sum_{j=0}^{k-1} 2^{2j+1}) \quad (10)$$

Every partial product of the pre-encoded NR4SD$^-$ and NR4SD$^+$ multipliers is carried out primarily based on Fig. 3b and 3c, respectively, except for the PPk-1 that corresponds to the vastest digit. As this digit is in MB form, can use the PPG of Fig.3a making use of the s_j bit. The partial products, well weighted, and the correction time period (COR) of (10) are fed right into a CSA tree. The input carry cin_j of (10) is calculated as $cin_j = two^-_j \wedge one^-_j$ and $cin_j = one^-_j$ for the NR4SD$^-$ and NR4SD$^+$ pre-encoded multipliers, respectively, primarily based on Tables 2 and 3. The carry save output of the CSA tree is eventually summed the usage of a quick CLA adder.

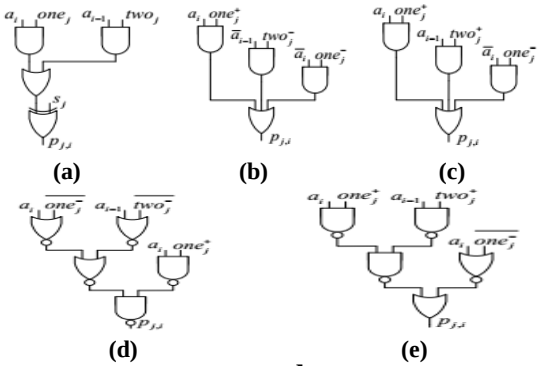


Fig 3: Generation of the i^{th} Bit $p_{j,i}$ of PP_j for a) Pre-Encoded MB Multipliers, b) NR4SD$^-$, c) NR4SD$^+$Pre-Encoded Multipliers, and d) NR4SD$^-$, e) NR4SD$^+$Pre-Encoded Multipliers after reconstruction.

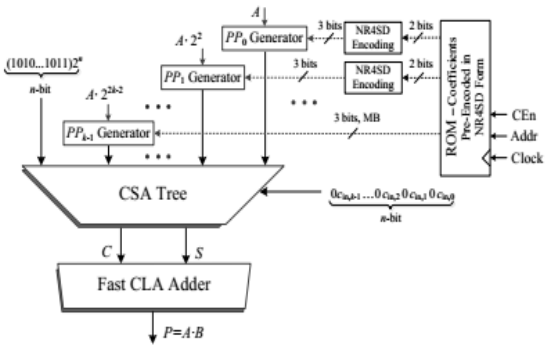
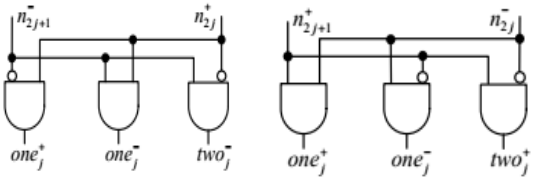


Fig 4: System Architecture of the NR4SD Multipliers.



(a) (b)

Fig 5: Extra Circuit Needed in the NR4SD$^-$ Multipliers to Complete the (a) NR4SD$^-$ and (b) NR4SD$^+$ Encoding.

Table 3. NR4SD$^-$ Encoding

2's complement			NR4SD$^-$ form			Digit	NR4SD$^-$ Encoding		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^-_{2j+1}	n^+_{2j}	b_j^{NR-}	one^+_j	one^-_j	two^-_j
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	+1	1	0	0
0	1	0	0	0	1	+1	1	0	0
0	1	1	1	1	0	-2	0	0	1
1	0	0	1	1	0	-2	0	0	1
1	0	1	1	1	1	-1	0	1	0
1	1	0	1	1	1	-1	0	1	0
1	1	1	1	0	0	0	0	0	0

Table 4. NR4SD$^+$ Encoding

2's complement			NR4SD$^+$ form			Digit	NR4SD$^+$ Encoding		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^+_{2j+1}	n^-_{2j}	b_j^{NR+}	one^+_j	one^-_j	Two^+_j
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	+1	1	0	0
0	1	0	0	1	1	+1	1	0	0
0	1	1	0	1	0	+2	0	0	1
1	0	0	0	1	0	+2	0	0	1
1	0	1	1	0	1	-1	0	1	0
1	1	0	1	0	1	-1	0	1	0
1	1	1	1	0	0	0	0	0	0

3. PROPOSED AGING-AWARE MULTIPLIER

Here recommend an aging-aware reliable multiplier layout with a novel adaptive hold logic (AHL) circuit. The multiplier is based totally at the variable-latency approach and may

regulate the AHL circuit to reap reliable operation underneath the have an impact on of NBTI and PBTI effects.

To be unique, the contributions of this undertaking are summarized as follows:

1) Novel variable-latency multiplier architecture with an AHL circuit. The AHL circuit can determine whether the input styles require one or two cycles and might modify the judging criteria to ensure that there's minimum performance degradation after great getting aging happens;

2) Comprehensive evaluation and comparison of the multiplier's performance underneath one-of-a-kind cycle intervals to reveal the effectiveness of our proposed structure;

3) An aging-aware dependable multiplier design method this is appropriate for huge multipliers. Despite the fact that the experiment is completed in 16- and 32-bit multipliers, our proposed structure can be effortlessly extended to large designs;

3.1. Proposed Architecture

Fig.6 indicates our proposed aging-aware multiplier architecture, which incorporates two m-bit inputs (m is a positive number), one 2m-bit output, one NR4SD^{-} or NR4SD^{+} multiplier, 2m 1-bit Razor flip-flops [22], and an AHL circuit. Inside the proposed architecture, the NR4SD multipliers may be tested by the range of zero's in either the multiplicand or multiplicator to expect whether the operation requires one cycle or two cycles to finish. While enter patterns are random, the number of zero's and one's inside the multiplicator and multiplicand follows a normal distribution. Consequently, the use of the quantity of zero's or one's as the judging criteria results in similar results. Subsequently, the two aging-aware multipliers may be applied the use of comparable structure, and the distinction between the 2 multipliers lies within the enter signals of the AHL. Razor flip-flops can be used to locate whether or not timing violations arise before the next input pattern arrives.

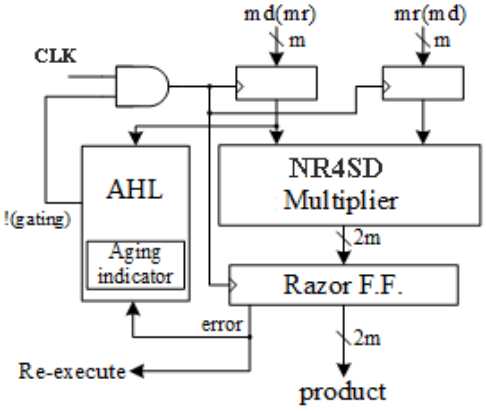


Fig 6: Proposed architecture (md means multiplicand; mr means multiplicator)

Fig. 7 suggests the information of Razor flip-flops. A 1-bit Razor flip-flop includes a main flip-flop, shadow latch, XOR gate, and mux. The main flip-flop catches the execution end result for the combination circuit using a ordinary clock signal, and the shadow latch catches the execution result the usage of a delayed clock signal, that is slower than the regular clock signal. If the latched little bit of the shadow latch isn't the same as that of the main flip-flop, this indicates the course delay of the current operation exceeds the cycle period, and the main flip-flop catches an incorrect result. If errors arise, the Razor flip-flop will set the error signal to at least one to notify the system to re-execute the operation and notify the AHL circuit that an error has befell. Here use Razor flip-flops to locate whether an operation this is considered to be a one cycle sample can surely end in a cycle. If now not, the operation is re-executed with 2 cycles. Despite the fact that the re-execution may seem steeply-priced, the overall price is low because the re-execution frequency is low. Extra info for the Razor flip-flop may be discovered in [23].

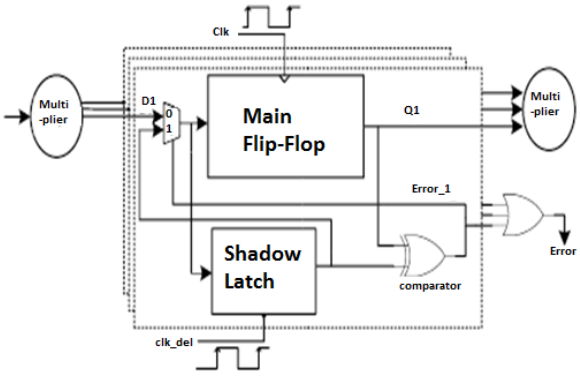


Fig 7: Razor flip flops.

The AHL circuit is the key aspect in the ageing-aware variable-latency multiplier. Fig.8 shows the details of the AHL circuit. The AHL circuit incorporates an aging indicator, judging blocks, one mux, and one D turn-flop. The aging indicator suggests whether or not the circuit has suffered tremendous overall performance degradation because of the aging effect. The aging indicator is carried out in a easy counter that counts the number of errors over a certain amount of operations and is reset to zero at the give up of those operations. If the cycle period is too low, the NR4SD multiplier isn't able to finish those operations effectively, causing timing violations. These timing violations could be stuck by the Razor turn-flops, which generate error alerts. If errors happen regularly and exceed a predefined threshold, it way the circuit has suffered significant timing degradation due to the aging impact, and the aging indicator will output sign 1; in any other case, it's going to output 0 to indicate the aging impact remains no longer extensive, and no moves are wished.

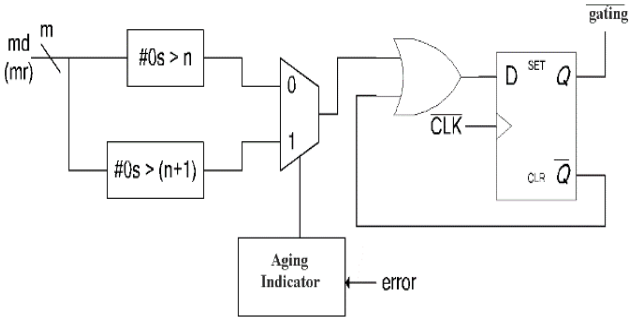


Fig 8: Diagram of AHL (md means multiplicand; mr means multiplicator)

The primary judging block inside the AHL circuit will output 1 if the number of zero's in the multiplicand (multiplicator) is bigger than n, and the second judging block in the AHL circuit will output 1 if the quantity of zero's inside the multiplicand (multiplicator) is greater than n+1. They're each hired to decide whether or not an enter sample requires one or two cycles, but most effective one among them can be chosen at a time. In the beginning, the ageing impact isn't significant, and the aging indicator produces 0, so the first judging block

is used. After a time, frame while the aging effect will become significant, the second one judging block is chosen. compared with the primary judging block, the second one judging block allows a smaller number of patterns to turn out to be one-cycle styles because it calls for more zero's within the multiplicand (multiplicator) The info of the operation of the AHL circuit are as follows: while an input pattern arrives, both judging blocks will determine whether or not the sample requires for one cycle or two cycles to complete and pass both outcomes to the multiplexer.

The multiplexer selects considered one of both result based on the output of the getting older indicator. Then an OR operation is accomplished between the end result of the multiplexer, and the Q^{-} signal is used to decide the input of the D flip-flop. When the pattern calls for one cycle, the output of the multiplexer is 1. The !(gating) sign turns into 1, and the input flip flops will latch new data within the next cycle. on the other hand, while the output of the multiplexer is 0, which means that the input sample requires for 2 cycles to finish, the OR gate will output zero to the D flip-flop. Consequently, the !(gating) signal can be 0 to disable the clock signal of the input flip-flops inside the subsequent cycle. Be aware that best a cycle of the input flip-flop may be disabled because the D flip-flop will latch 1 in the subsequent cycle.

The overall flow of our proposed architecture is as follows: when input patterns arrive, the NR4SD multiplier, and the AHL circuit execute simultaneously. In line with the number of zero's within the multiplicand (multiplicator), the AHL circuit comes to a decision if the enter patterns require one or two cycles. If the input pattern requires two cycles to finish, the AHL will output 0 to disable the clock signal of the flip-flops. Otherwise, the AHL will output 1 for regular operations. When the NR4SD multiplier finishes the operation, the result may be passed to the Razor flip-flops. The Razor flip-flops test whether or not there may be the course put off timing violation. If timing violations occur, it approaches the cycle period isn't always lengthy enough for the current operation to finish and that the execution end result of the multiplier is incorrect. Accordingly, the Razor flip-flops will output a blunders to tell the device that the modern operation desires to here execute the use of cycles to make certain the operation is accurate. In this case, the extra re-execution cycles as a result of timing violation incurs a penalty to universal average latency.

But, our proposed AHL circuit can accurately expect whether or not the input patterns require one or two cycles in most instances. Only a few input styles may additionally purpose timing variations when the AHL circuit judges incorrectly. In this situation, the more re-execution cycles did no longer produce good sized timing degradation.

In précis, our proposed multiplier design has 3 key capabilities. First, its miles a variable-latency design that minimize the timing waste of the noncritical paths. 2nd, it is able to offer dependable operations even after the aging impact happens. The Razor flip-flops stumble on the timing violations and re-execute the operations using two cycles. Ultimately, our architecture can modify the share of 1-cycle patterns to reduce performance degradation due to the aging impact. While 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 2 cycles.

4. RESULTS

A simulation end result for NR4SD multiplier is simulated in a Xilinx ISE 14.1. These tools will help to research its performance and calculate the power, delay and area .Snapshot is nothing but each and every moment of the application while running shown in Figs. 9 to 10. These snapshots gives the clear view of application developed. It will be most useful to the new peoples to understand for the future steps.

In the below Table 5-6 4*4,8*8,16*16,32*32 NR4SD multipliers are compared for area of NR4SD Multiplier. By observing the Table-5-6 it shows the clear view regarding number of components required to develop particular NR4SD multiplier based on input range with and without AHL circuit.

Table 5. Device Utilization (area) Summary of NR4SD Multiplier with AHL circuit

Logic Utilization	32-bit	16-bit	8-bit	4-bit
Number of Slice Flip Flops	226	130	87	58
Number of 4 input LUTs	4608	1,050	272	106
Number of occupied Slices	2,304	607	180	72
Average Fan-out of Non-Clock Nets	3.97	3.41	3.15	2.86

Table 6. Device Utilization (area) Summary of NR4SD Multiplier without AHL circuit

Logic Utilization	32-bit	16-bit	8-bit	4-bit
Number of 4 input LUTs	2094	553	155	36
Number of occupied Slices	1246	315	84	19
Number of bonded IOBs	128	64	32	16
Average Fan-out of Non-Clock Nets	4.44	4.04	3.60	3.26

Here in Table-7 it shows the time delay consuming of NR4SD multiplier with and without AHL circuit. By observing this it shows the range of delay increasing due to increasing number of input ranges. Here the time consuming is in the ranges of Nano seconds.

Table 7. Time Delay consuming of NR4SD Multiplier

Input range	Delay with AHL circuit	Delay without AHL circuit
4-bit	8.068ns	12.063ns
8-bit	13.379ns	22.326ns
16-bit	26.755ns	43.790ns
32-bit	40.537ns	77.345ns

Table 8. Power analysis of NR4SD Multiplier

Input range	Power(W) without AHL circuit	Power(W) with AHL circuit
4-bit	0.271	0.209
8-bit	0.343	0.219
16-bit	0.508	0.233
32-bit	0.883	0.306

Fig 9 shows the simulation result of NR4SD multiplier with AHL circuit. Here Figs9 (a) to (d) shows 4*4, 8*8, 16*16, 32*32 bit NR4SD multiplier response and Figs 10 shows the simulation result of NR4SD multiplier without AHL circuit. Here Figs 10 (a) to (d) shows 4*4, 8*8, 16*16, 32*32 bit NR4SD multiplier response. In these figures shows the input and output responses. Table 8 shows the 4*4, 8*8, 16*16, 32*32 NR4SD multipliers power analysis with and without AHL circuit.

In the below Table 9-12 the 4*4 ,8*8,16*16,32*32 NR4SD multipliers are compared for Error count based on no.of i/p and no.of Zero's in multiplicand. Here applying the different number of inputs which are randomly generated and shows the number of error counts based on number of zero's present in the randomly generated input bits.

Table 9. Error count based on no.of i/p and no.of Zero's in multiplicand of NR4SD 4 bit multiplier

No of i/p	No of 0's-2	No of 0's-1
13000	5237	2891
11000	4429	2440
9000	3625	2007
5000	2000	1131

Table 10. Error count based on no.of i/p and no.of Zero's in multiplicand NR4SD 8 bit multiplier

No.of i/p	0's-5	0's-3	0's-7
5000	2337	1333	3461
9000	4185	2374	4436
11000	5113	2896	5421
13000	6047	3436	6406

Table 11. Error count based on no.of i/p and no.of Zero's in multiplicand NR4SD 16 bit multiplier

No.of i/p	0's-3	0's-5	0's-7	0's-9	0's-11	0's-13
5000	40	467	1432	2162	2416	2422
9000	78	835	2569	3999	4379	4420
11000	102	1030	3151	4761	5359	5420
13000	119	1220	3723	5631	6339	6419

Table 12. Error count based on no.of i/p and no.of Zero's in multiplicand of NR4SD 32 bit multiplier

No.of i/p	0's-10	0's-15	0's-20	0's-25
5000	189	1490	2402	2382
9000	334	2730	4322	4381
11000	413	3339	5284	5381
13000	494	3952	6246	6380

5. CONCLUSIONS

This paper proposed an aging-aware reliable multiplier design with the AHL. The multiplier is able to modify the AHL to mitigate overall performance degradation due to increased delay. Be aware that in addition to the BTI impact that increases transistor delay, interconnect additionally has its aging issue, that is referred to as electromigration. Electromigration happens whilst the current density is high enough to cause the drift of metal ions along the direction of electron flow. The metal atoms can be regularly displaced after a time period, and the geometry of the wires will change. If a wire becomes narrower, the resistance and delay of the wire will be expanded, and in the end, electromigration might also cause open circuits. This problem is also more severe in advanced manner technology because metal wires are narrower, and changes in the wire width will cause larger resistance differences. If the aging effects as a result of the BTI effect and electromigration are considered collectively, the delay and overall performance degradation will be more significant. Fortunately, our proposed variable latency multipliers can be used under the influence of both the BTI effect and electromigration. Similarly, our proposed variable latency multipliers have much less performance degradation because variable latency multipliers have much less timing waste, but conventional multipliers need to consider the degradation resulting from both the BTI effect and electromigration and use the worst case delay as the cycle period.

6. REFERENCES

[1] Ing-Chao Lin, Member, IEEE, Yu-Hung Cho, and YiMing Yang, "Aging-Aware Reliable Multiplier Design With Adaptive Hold Logic" IEEE Transactions On Very Large Scale Integration (VLSI) Systems.

[2] 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, pp. 23–25.

[3] S. Zafar, A. Kumar, E. Gusev, and E. Cartier, "Threshold voltage instabilities in high-k gate dielectric stacks," *IEEE Trans. Device Mater.Rel.*, vol5, no.1, pp.45–64, Mar.2005.

[4] 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.

[5] R.Vattikonda, W.Wang, and Y. Cao, "Modeling and minimization of pMOS NBTI effect for robust naometer design," in *Proc. ACM/IEEE DAC*, Jun. 2004, pp. 1047–1052.

[6] 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.

[7] 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.

[8] A. Calimera, E. Macii, and M. Poncino, "Design techniqures for NBTItolerant power-gating architecture," *IEEE Trans. Circuits Syst., Exp. Briefs*, vol. 59, no. 4, pp. 249–253, Apr. 2012.

[9] K.-C. Wu and D. Marculescu, "Joint logic restructuring and pin reordering against NBTI-induced performance degradation," in *Proc. DATE*, 2009, pp. 75–80.

[10] 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.

[11] 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.

[12] K.-C. Wu and D. Marculescu, "Aging-aware timing

analysis and optimization considering path sensitization," in *Proc. DATE*, 2011, pp. 1–6.

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

[14] 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.

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

[16] 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.

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

[18] M. Olivieri, "Design of synchronous and asynchronous variable-latency pipelined multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 4, pp. 365–376, Aug. 2001.

[19] 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.

[20] Y. Chen, H. Li, J. Li, and C.-K. Koh, "Variable-latency adder (VL-Adder): New arithmetic circuit design practice to overcome NBTI," in *Proc. ACM/IEEE ISLPED*, Aug. 2007, pp. 195–200.

[21] 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.

[22] Z. Huang, "High-level optimization techniques for low-power multiplier design," Ph.D. dissertation, Department of Computer Science, University of California, Los Angeles, CA, 2003.

[23] Z. Huang and M. Ercegovac, "High-performance low-power left-to-right array multiplier design," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 272–283, Mar. 2005.