

VARIABLE LATENCY MULTIPLIER WITH AHL

Anthonyammal S

Department of ECE, M.E.VLSI Design
Sasurie Academy of Engineering
Coimbatore, India
dheepa.ct@gmail.com

Poonkothai S

Department of ECE, Assistant Professor
Sasurie Academy of Engineering
Coimbatore, India
poonkothais@gmail.com

Abstract—Digital multipliers are the most critical arithmetic functional units. The overall performances of the systems are depends on the throughput of the multiplier. The negative bias temperature effect occurs when an pMOS transistors are under negative bias, increasing the threshold voltage of the pMOS transistor, and reducing multiplier speed. Similarly Positive bias temperature instability, occur when an nMOS transistor is under positive bias. These two effects degrade the transistor speed, so the system may fail due to timing violations. The particular effect of an nMOS transistor is positive bias temperature instability, which occurs when an nMOS transistor is under positive bias. It is important to design reliable high performance multiplier. In this paper, propose a variable latency multiplier design with an novel adaptive hold logic (AHL) circuit. The multiplier is able to provide that higher throughput through the variable latency and can adjust the AHL circuits to mitigate performance degradation. Moreover, the proposed architecture can be applied to a column- or row-bypassing multiplier. This proposed architecture can achieve performance improvement as compared with fixed-latency row-bypassing multipliers. The experimental results show that the proposed architecture with 16×16 and 32×32 column-bypassing multiplier can attain up to 68.76% and 82.28% performance improvement, respectively, compared with 16bit and 32bit fixed latency column-bypassing multipliers

Keywords— AHL; NBTI; PBTI; Reliable Multiplier; Latency; Precision Multiplier.

I. INTRODUCTION

Multipliers are among the critical arithmetic functional units in many applications, such as the Fourier transform, discrete cosine transforms, and digital filtering. Multipliers play an important role in today's digital signal processing and various other applications. With advance in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets are high speed, low power consumption, regularity of layout and hence less area or combination of them in one multiplier it making them suitable for various high speed, low power and compact VLSI implementation.

Multiplication involves two operations: The generation of the partial products and their accumulations. There is a common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added, it is the main parameter that determines the performance of the multiplier. Two possible ways are there. Speeds up the multiplication: Reduce the number of partial products or accelerate their accumulation. A smaller number of partial products are also reduces the complexity and time. The throughput of these applications are depends on multipliers, and if the multipliers are slow, the performance of entire circuits will be reduced.

A. Algorithm

- The LSB of Multiplier is '1', then add the multiplicand into an accumulator.
- Shift the multiplier one bit to the right and the multiplicand one bit to the left.
- Stop when all bits of the multiplier are zero.

B. Temperature Instability

A traditional method to mitigate the, including such things as guard-banding and gate oversizing; however, this approach can be very cynical and area and power inefficient. To avoid this problem, many NBTI methodologies have been proposed. An NBTI sleep transistor was designed to reduce the effects on pMOS sleep-transistors, and the lifetime constancy of the circuits under consideration was improved.

Wu and Marculescu also proposed an NBTI optimization method. In dynamic voltage scaling, body-biasing techniques were proposed to reduce power or extend circuit life. The variable-latency design was projected to reduce the timing waste of traditional circuits. The variable-latency design divides the circuits into two parts: 1) Shorter path and 2) Longer path. Shorter paths can execute correctly in one cycle, where-as longer paths needs two cycles to execute.

C. Paper Contribution

In this paper proposes a reliable multiplier design with a novel adaptive hold logic (AHL) circuit. The multipliers are

based on the variable-latency technique and can adjust the AHL circuit to achieve reliable operation under the effects of NBTI and PBTI.

To be specific, the contributions of this paper are summarized as follows:

- 1) Novel variable-latency multipliers with an AHL circuit. The AHL circuit can decide whether the input patterns need one or two cycles and can adjust the judging criteria to make certain that there is minimum performance degradation after substantial aging occurs.
- 2) All-inclusive analysis and comparison of the multiplier's performance under different cycle periods to show the efficacy of our proposed architecture.
- 3) An aging-aware consistent multiplier design method that is suitable for great multipliers. Although the experiment is performed in 16- and 32-bit multipliers, our proposed architecture can be easily protracted to large designs.

II. EXISTING METHOD

A. Array Multiplier

Array multiplier is well notorious due to its regular structure. Multiplier circuit is based on repeated addition and shifting procedure. Each partial product is engendered by the multiplication of the multiplicand with one multiplier digit. The summation can be performed with typical carry propagation adder. N-1 adders are required where N is the number of multiplier bits.

The delay of the array multiplier is given by

$$(T_{critical}) = [(N-1) + (N-2)] * T(Carry) + (N-1) * T(Sum) + T(AND) \quad (1)$$

Where,

$T(Carry)$ is the propagation delay between input and output carry,

$T(Sum)$ is the delay between the input carry, sum bit of the full adder,

$T(AND)$ is the delay of AND gate,

N is the length of multiplier operand.

The adjournment associated with the array multiplier is the time taken by the signals to propagate through the AND gates and adders that form the multiplication array.

Delay of an array multiplier rest on only upon the depth of the array not on the partial product width.

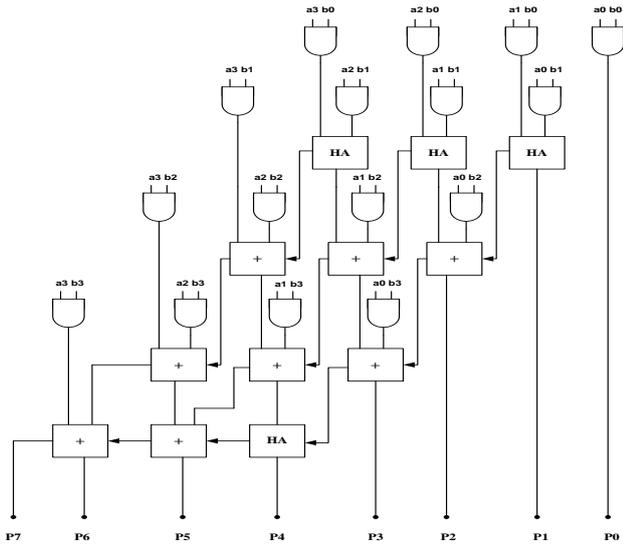


Fig.1. Architecture of Array Multiplier

B. Normal Operation

The AM is a fast parallel AM. The multiplier array consists of $(n - 1)$ rows of carry save adder (CSA), in which each row comprises $(n - 1)$ full adder (FA) cells. Each FA in the CSA array has two outputs: 1) the sum bit goes down and 2) the carry 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. It uses short wires that go from one full adder to adjacent full adders horizontally, vertically or diagonally. An $n \times n$ array of AND gates can compute all the terms simultaneously. The bits are summed by an array of „ $n [n - 2]$ “ full adders and „ n “ half adders.

The shifting of partial products for their correct alignment is performed by simple routing and does not require any logic. The number of rows in array multiplier signifies length of the multiplier and width of each row denotes width of multiplicand. The output of each row of adders are acts as input to the next row of adders. Each row of full adders or 3:2 compressors adds a partial product to the partial sum, generating a new-fangled partial sum and a sequence of carries.

C. Disadvantages

- The speed will be slow for a very eclectic multiplier.
- Array multipliers tend to consume most of the power in DSP computations.
- It also requires larger number of gates because of which area is also increased.
- The worst-case delay of the array multiplier is proportional to the width of the multiplier and delay for this multiplier is larger.

III. PROPOSED METHOD

A. Block Diagram

Block Contains

- a) Column Row Bypass Multiplier
- b) Razor Flip Flop
- c) AHL & Aging Indicator

These are the main blocks of the proposed blocks which are explained below,

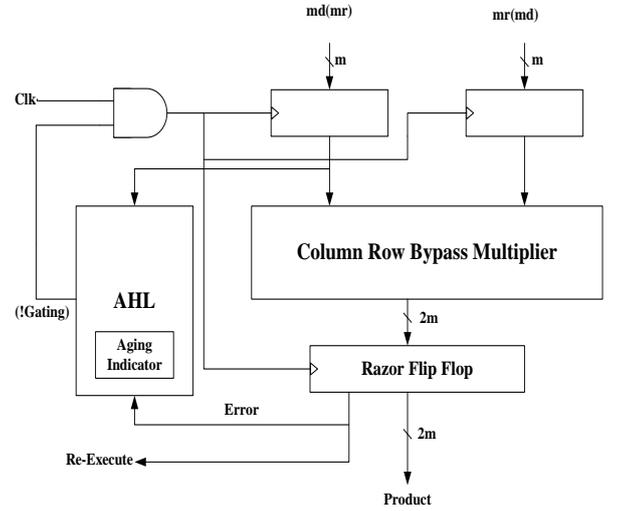


Fig.2. Main Block Diagram

B. Bypassing Technique

Bypassing with references to multiplier means turning off some columns or rows or both in the multiplier array whenever firm multiplier or multiplicand or both bits are zero.

Two types of Bypassing Technique is used

- 1) Column Bypassing Technique
- 2) Row Bypassing Technique

a. Column & Row Bypass Multiplier

This paper focuses on the column- and row-bypassing multipliers can be examined by the number of zeros in either the multiplicand or multiplier to envisage whether the operation requires one cycle or two cycles to complete. When input patterns are accidental, the number of zeros and ones in the multiplier and multiplicand follows a normal distribution. Therefore, the number of zeros or ones as the judging measures results in similar outcomes. Hence, the two multipliers can be implemented using analogous architecture, and the difference between the two bypassing multipliers lies in the input signals of the AHL.

b. Column Bypass Multiplier

A column-bypassing multiplier is an upgrading on the normal array multiplier (AM). The AM is a fast parallel AM.

The multiplier array comprises of $(n - 1)$ rows of carry save adder (CSA), in which each row contains $(n - 1)$ full adder (FA) cells. Every 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 swell adder for carry propagation. The FAs in the AM are always active regardless of input positions. 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. For a low-power column-bypassing multiplier, the addition operations in the $(i+1)^{th}$ column can be sidestepped if the bit, a_i , in the multiplicand is 0. In the multiplier design, the modified FA is modest than that in the row bypassing multiplier.

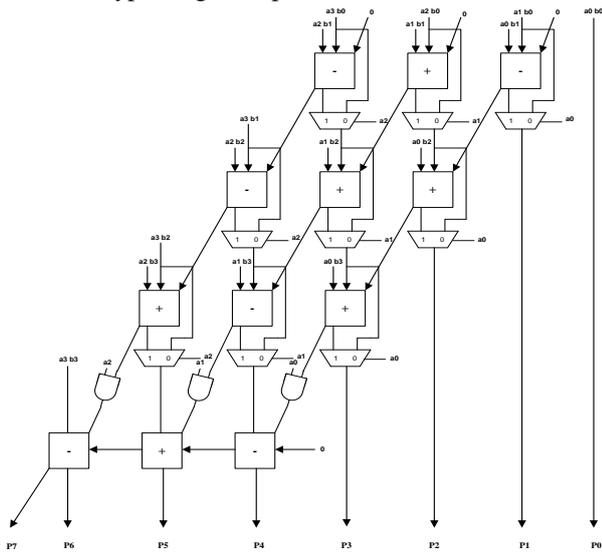


Fig.3. Column Bypass Multiplier

The column bypassing multiplier is constructed by using two three-state gates and one multiplexor. The modified HA cell is shown in Figure. If $a_j = 0$, the HA will be disabled.

c. Modified HA and FA cell

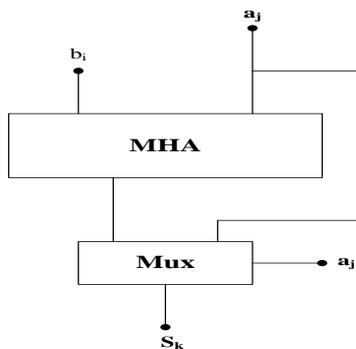


Fig.4. Modified HA cell for column bypassing multiplier

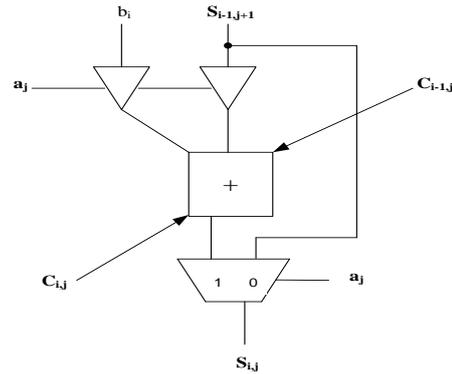


Fig.5. Modified FA cell for column bypassing multiplier

As a result, when $a_j = 0$, the two input of $FA_{0,j}$ are disabled, and thus its output carry bit will not be changed. The total number of full adders required to design column bypass multiplier are $n*(n-2)$. The total number of half adders are obligatory to design column bypass multiplier are n . The total number of tri-state buffers required to design column bypass multiplier are $2*n*(n-1)$. The total number of 2 to 1 Multiplexers required to design column bypass multiplier are $(n-1)*(n-1)$.

d. Row Bypass Multiplier

Row bypassing technique is based on number of zeros in the multiplier bits. In this multiplier operation of some of the rows of adders in the basic multiplier array are disabled during operation. A low-power row-bypassing multiplier is also projected to reduce the activity power of the AM. The operation of the low-power row-bypassing multiplier is parallel to that of the low-power column-bypassing multiplier, but the selector of the multiplexers and the tri-state gates use the multiplier.

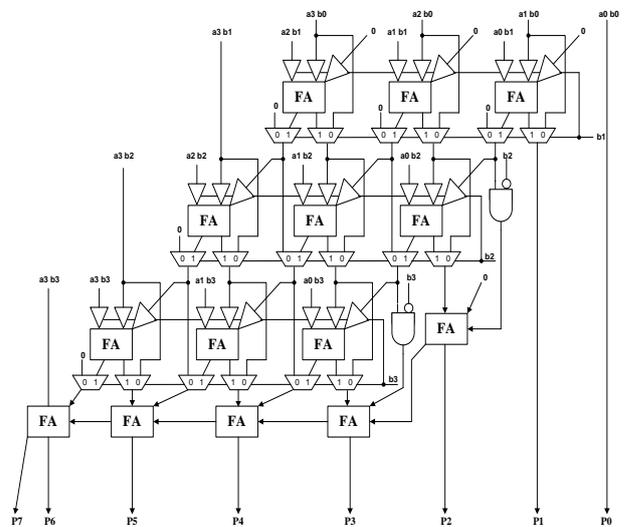


Fig.6. Row bypassing Multiplier

The tri-state buffers placed at the input of the adder cells disable signal transitions in the adders which are by-passed, and the input-carry bit and input-sum bit are passed to downwards. Thus, the power depletion can be reduced if one can condense the switching activity of a given logic circuit without changing its function. An evident method to condense the switching activity is to shut down the idle part of the circuit which is not in operating condition.

e. Row Bypassing Adder Cell

The Row Bypassing Adder Cell (RA) when the consistent partial product is zero, the RA disabled unnecessary transitions and bypassed the inputs to outputs. But, the limitation of this technique is that it cannot stop the switching activity even if the bit coefficient is zero that eventually results in unnecessary power dissipation.

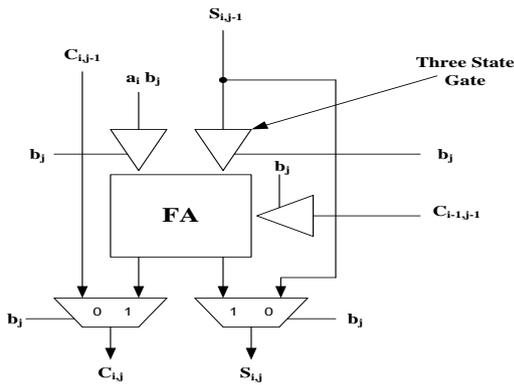


Fig.7. Row Bypassing Adder Cell (RA)

C. Variable Latency Design

Variable-latency design was projected to reduce the timing waste occurring in traditional circuits that use the critical path cycle as an effecting cycle period. The basic concept is to execute a shorter path using a shorter cycle and longer path using two cycles.

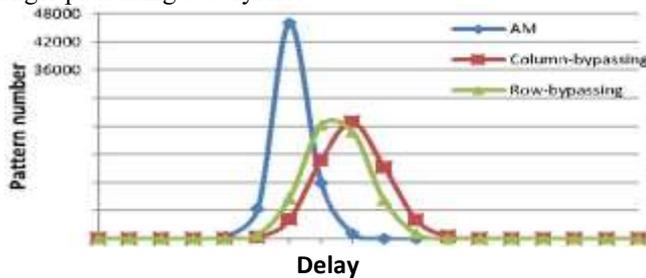


Fig.8. Path delay distribution

D. Razor Flip Flop

- A 1-bit Razor flip-flop contains,
- 1. Main flip-flop
 - 2. Shadow latch
 - 3. XOR gate
 - 4. Multiplexer

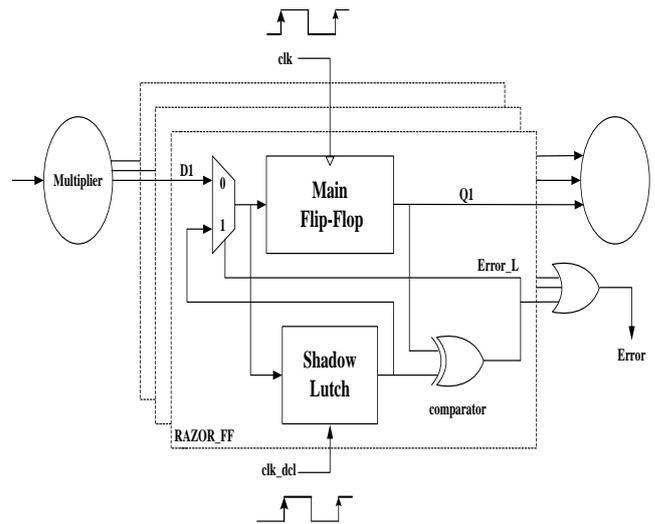


Fig.10. Razor Flip Flop

The main flip-flop catches the execution result for the amalgamation circuit using a normal clock signal, and the shadow latch catches the execution result using a deferred clock signal, which is slower than the normal clock indication. If the latched bit of the shadow latch is altered 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 claps an incorrect result. If errors occur, the Razor flip-flop will set the blunder signal to 1 to notify the system to re-execute the operation and notify the AHL circuit that an error has occurred.

Razor flip-flops used to detect whether an operation that is considered to be a one-cycle patterns can really finish in a cycle. If not, the maneuver is re-executed with two cycles.

E. AHL and Aging Indicator

The AHL circuit is the key component in the variable-latency multiplier. The AHL circuit contains,

- 1. Aging indicator
- 2. Multiplexer
- 3. Two judging blocks
- 4. D flip-flop

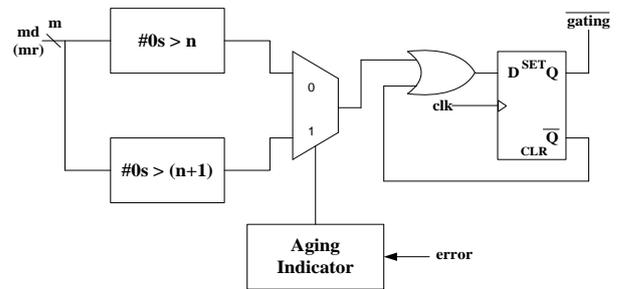


Fig.11. AHL

The aging indicator indicates whether the circuit has suffered significant performance degradation due to the effect of aging. The aging indicator is implemented in a simple

counter that counts the number of errors over a convinced amount of operations and is reset to zero at the end of those operations. The first arbitrating block in the AHL circuit will output 1 if the number of zeros in the multiplicand is larger than n , and the second arbitrating block in the AHL circuit will output 1 if the number of zeros in the multiplicand is larger than $n + 1$.

F. Proposed System Applications

DSP-Digital Signal Processing is the precise manipulation of an information signal to modify or improve it in some way.

ALU-ALUs of various bit-widths are frequently required in very large-scale integrated circuits (VLSI) from processors to application specific integrated circuits (ASICs).

IV. COMPARISON

Table.1 Utilization Comparison

Comparison	Utilization Comparison			
	Existing Method		Proposed Method	
1	INV	3	INV	1
2	LUT	15	LUT	8
3	FF	32	FF	25
4	REG	4	REG	2
5	IO BUF	21	IO BUF	19

Table.1 Utilization Comparison

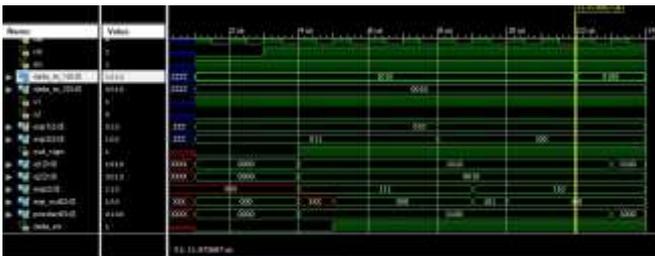
S.No	Performance Analysis		
	Features	Array	Bypass
1	Power(w)	0.089	0.043
2	Time(ns)	3.326	2.225
3	Gate delay	0.113	0.097
4	Net delay	0.381	0.279
5	Efficiency	86.3	95.8
6	Performance	Low	High

V. SOFTWARE REQUIREMENT

A. Software

- **Verification Tool** - Modelsim 6.4c
- **Synthesis Tool** - Xilinx ISE 9.1

B. Simulation Result



VI. CONCLUSION

This proposed architecture is design with the AHL. The multiplier is able to adjust the AHL to mitigate performance degradation due to increased delay. In addition, our proposed variable latency multipliers have less performance degradation because variable latency multipliers have less timing waste, but traditional multipliers need to consider the degradation caused by both the BTI effect and electro-migration and use the worst case delay as the cycle period. In future it will implemented in Precision Multiplier.

ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if I did not have a support of many individuals. Therefore I would like to extend our sincere gratitude to all of them.

VII. REFERENCES

- [1] Ing-Chao Lin, Member, IEEE, Yu-Hung Cho, and Yi-Ming Yang “Aging-aware reliable multiplier design with adaptive hold logic,” IEEE Transl. Grant NSC 100-2221-E-006-177 and Grant NSC 102-2221-E-006-281. Feb 2014.
- [2] A.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] 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, pp. 253–258. Aug. 2010.
- [4] K. Du, P. Varman, and K. Mohanram, “High performance reliable variable latency carry select addition,” in Proc. pp. 1257–1262. 2012.
- [5] Y. Lee and T. Kim, “A fine-grained technique of NBTI aware voltage scaling and body biasing for standard cell based designs,” in Proc. ASP-DAC, pp. 603–608. 2011.
- [6] N. V. Mujadiya, “Instruction scheduling on variable latency functional units of VLIW processors,” in Proc. ACM/IEEE ISE, a.307–312. Dec. 2011.
- [7] K.-C. Wu and D. Marculescu, “Aging-aware timing analysis and optimization considering path sensitization,” in Proc., pp. 1–6. 2011.
- [8] 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.
- [9] 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.
- [10] R. Vattikonda, W. Wang, and Y. Cao, “Modeling and minimization of pMOS NBTI effect for robust nanometer design,” in Proc. 43rd ACM/IEEE DAC, Aug. 2006, pp. 1047–1052.
- [11] 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.
- [12] D. Mohapatra, G. Karakonstantis, and K. Roy, “Low-power processvariation tolerant arithmetic units using input-based elastic clocking,” in Proc. ACM/IEEE ISLPED, pp. 74–79. Aug. 2007.
-