

Variable-Latency Adder (VL-Adder) Designs for Low Power and NBTI Tolerance

Yiran Chen, Hai Li, Cheng-Kok Koh, Kaushik Roy,
Guangyu Sun and Jing Li

Abstract — we proposed a new adder design, called **Variable-Latency Adder (VL-adder)**. This technique allows the adder to work at a lower supply voltage than that required by a conventional adder, while maintaining the same throughput. The VL-adder design can be further modified to overcome the effects of Negative Bias Temperature Instability (NBTI) on circuit delay. By applying VL-adder concept to 64-bit carry-select adder design, more than 40% energy saving is obtained while a similar throughput is maintained.

Index Terms— Integrated circuit design, Logic design, Digital arithmetic.

I. INTRODUCTION

Power supply voltage (V_{DD}) scaling can effectively reduce both leakage and dynamic power of a VLSI circuit [1]. The value of V_{DD} is mainly determined by the desired operation frequency (or clock cycle period T_{clk}) and the **critical** path delay, denoted as T_{crit} . V_{DD} is selected such that $T_{clk} \geq T_{crit} + \delta$, where $\delta > 0$ is a margin added to take into account the uncertainties of circuit/device parameters, environmental factors, as well as the expected lifetime of the circuit.

The critical timing path of a VLSI circuit is rarely activated or sensitized. In a 32-bit unsigned ripple-carry adder (RCA) for example, the longest carry propagation delay occurs only when the carry-out signal (C_O) generated by the adder of the least-significant bit (LSB) propagates through the adder of the most-significant bit (MSB) [2]. The operands $A <31:0> = 0xFFFFFFFF$ and $B <31:0> = 0x00000001$ would activate such a critical delay path. The occurrence probability of operands that result in the longest carry propagation delay is very low, i.e., $2^{31}/2^{64} \approx 1.2 \times 10^{-10}$, assuming that operands are random.

In this paper, we propose a *variable-latency adder (VL-adder)* concept that exploits the differences in the latency required by an adder to process different operands. The VL-adder predicts the operation latency on the fly. When a long-latency operation occurs, the data capturing clock edge is shifted by one more clock cycle so that the adder output can be latched in correctly. Compared to a conventional adder design, a VL-adder can work at a much lower V_{DD} such that the majority of operations can be processed within one clock cycle. Very few operations with long latency take two cycles to complete. Significant power savings can be achieved while a similar throughput is maintained.

Yiran Chen and Hai Li are with Seagate Technology, Bloomington, MN 55435 USA. (e-mail: {yiran.chen, helen.li}@seagate.com).

Cheng-Kok Koh, Kaushik Roy, and Jing Li are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: {chengkok, kaushik, jingli}@purdue.edu).

Guangyu Sun is with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 (e-mail: gsun@cse.psu.edu).

Moreover, we proposed a modification of VL-adder design for the Negative Bias Temperature Instability (NBTI) tolerance: when a chip is aging, the detection threshold of long-latency operation is automatically adjusted. Hence, the operation that incurs the timing violation due to NBTI is automatically re-categorized as long-latency operation, which is processed within two clock cycles.

II. CONCEPT OF VARIABLE-LATENCY ADDER (VL-ADDER)

A. Carry Propagation in Carry-Select Adder

In a conventional design of M -bit square-root CSA, the input bits are divided into $\approx \sqrt{2M}$ carry-select stages, as shown in Fig. 1. One carry-select stage (CSS) includes carry generation blocks, a multiplexer (MUX) for carry selection, and a sum generation block. In CSS_i , two carry-out signals C_{00} and C_{01} of every bit are pre-calculated by assuming that $C_{in,i}$, the carry-in signal of CSS_i , is 0 or 1, respectively. C_{00} or C_{01} is then selected by $C_{in,i}$, the actual carry-out signal $C_{out,i-1}$ of CSS_{i-1} , the previous CSS. The carry-out signal of the most significant bit (MSB) adder in CSS_i is sent out as the carry out signal $C_{out,i}$ of CSS_i , which is also the carry-in signal $C_{in,i+1}$ of CSS_{i+1} , the next CSS.

Usually the number of input bits for each CSS increases linearly from the least significant stage to the most significant stage and the first CSS includes two single-bit adders, one full adder (FA) and one half adder (HA). Hence, in a conventional square-root CSA with n CSS's, n is chosen as $(3+n) \cdot n/2 \approx M$. In particular, when $M = 64$, $n \approx 10$. The design of a conventional 64-bit CSA is shown in the row "Conv. CSA" in TABLE I [4].

There are multiple critical carry propagation paths in CSA, each of which starts from the first bit adder of every CSS, crosses the MUXes of the sequential stages, and ends at the sum generation block of the last stage (see Fig. 1). The longest delay of CSA D_{CSA} can be calculated by [3]:

$$D_{CSA} = D_{setup} + q_0 D_{carry} + n D_{mux} + D_{sum}, \quad (1)$$

where D_{carry} , D_{mux} and D_{sum} are the delays of carry generation circuitry, MUX, and sum generation circuitry, respectively; q_0 is the number of input bits in the first CSS; n is the number of CSS's; and D_{setup} is the setup time of CSA, which is the time

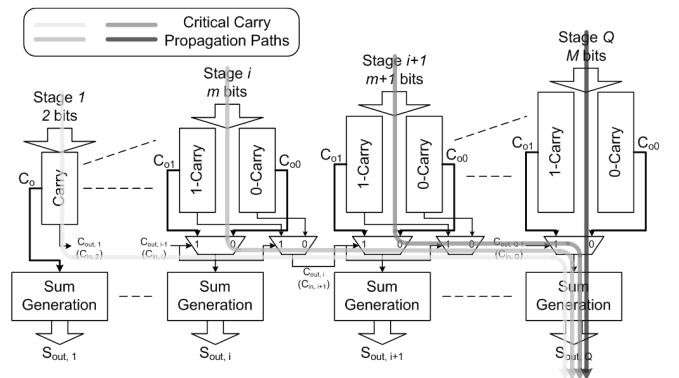


Fig. 1. Carry propagation in CSA.

TABLE I
THE NUMBERS OF INPUT BITS OF CARRY-SELECT STAGES IN 64-BIT STANDARD CSA AND VL-CSA (AND A MODIFIED VL-CSA)

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13
Conv. CSA	2	2	3	4	6	7	8	9	11	12	/	/	/
VL-CSA	2	2	3	4	6	7	7	<u>4</u>	<u>3</u>	5	<u>6</u>	<u>7</u>	<u>8</u>

taken to create the intermediate signals G (Generation) and P (Propagation) [3].

The delay of a CSA heavily depends on the input vectors for example, the carry propagation through the MUX chain of CSA. Let m_i denote the number of bits in CSS_i . When the input bits $A_k = B_k$ for any $k = 1, 2, \dots, m_i$ in CSS_i , the carry-out signal $C_{out,i}$ of CSS_i is determined by only the input bit pairs $(A_k, B_k, k = 1, 2, \dots, m_i)$ of CSS_i , regardless of the carry-in signal $C_{in,i}$ ($C_{out,i-1}$) from CSS_{i-1} . Hence, the carry propagation through the MUX chain is *killed* at the MUX of CSS_i .

B. Variable-Latency CSA – VL-CSA

A new variable-latency CSA architecture called “VL-CSA” is shown in the row “VL-CSA” in TABLE I. The CSS ’s of VL-CSA are divided into two separate sequences, sequence I and sequence II. Sequence I and sequence II include k ($=7$) and m ($=6$) CSS ’s, respectively. In each sequence, the first stage CSS starts with a relatively small number of input bits and the number of input bits to a CSS increases with the stage number of the CSS , as in the case of a conventional CSA.

Now, we combine the two sequences of CSS ’s and label them from 1 through $m+k$. Observe that by design, there is only one critical carry propagation path, which starts from the first bit adder of CSS_1 , crosses the MUXes of all CSS ’s, and ends at the sum generation block of CSS_{m+k} . Obviously, the longest delay of VL-CSA L_{Long} is longer than that of conventional CSA, D_{CSA} .

However, the carry can propagate through the first N_C CSS ’s in sequence II, i.e., $CSS_{k+1}, \dots, CSS_{k+N_C}$, only if

$$P_{N_C} = \prod_{i \in \{CSS_{k+1}, \dots, CSS_{k+N_C}\}} P_i = 1, \quad (2)$$

where $P_i = A_i \oplus B_i$ is the propagation signal of the adder of bit i in $CSS_{k+1}, \dots, CSS_{k+N_C}$. A long-latency operation, of which the longest delay is L_{Long} , occurs only when the carry can propagate through first N_C CSS ’s in sequence II. For random inputs, the probability of long-latency operations, denoted by Pr_{Long} , is:

$$Pr_{Long} = \prod_{i=1}^{N_C} \frac{1}{2^{m_i}}, \quad (3)$$

where m_i is the total number of input bits of CSS_{k+i} .

When a short-latency operation is detected, say $P_{N_C} = 0$, the carry propagation in VL-CSA is divided into two parts: one from CSS_1 through CSS_{k+N_C} and the other one from CSS_{k+1} through the last stage CSS_{k+m} . The longest latency of all short-latency operations, denoted by L_{Short} , is:

$$L_{Short} = \max(D_{max1}, D_{max2}), \quad (4)$$

where D_{max1} and D_{max2} are the longest delays of the first and the second parts, respectively. We have:

$$D_{max1} = D_{setup} + q_{0,I} D_{carry} + (k + N_C) D_{mux} + D_{sum}, \quad (5)$$

$$D_{max2} = D_{setup} + q_{0,II} D_{carry} + m D_{mux} + D_{sum}.$$

Here $q_{0,I}$ and $q_{0,II}$ are the numbers of input bits in the first CSS ’s in sequence I and sequence II, respectively.

Now, let V_{CSA} be the supply voltage at which the conventional CSA operates correctly. To achieve better power-delay product (PDP), the VL-CSA should satisfy the following conditions:

1) At V_{CSA} , $L_{Short} < D_{CSA}$, where D_{CSA} is the longest delay of conventional CSA at V_{CSA} and L_{Short} is the longest delay of all short operations of the VL-CSA at V_{CSA} ;

2) At a scaled supply voltage V_{VL-CSA} ($< V_{CSA}$), at which the longest latency of all short-latency operations of the VL-CSA $L_{Short-VL}$ is exactly D_{CSA} , the PDP of VL-CSA is better than that of the conventional CSA at V_{CSA} :

$$P_{VL-CSA} \cdot (1 + Pr_{Long}) < P_{CSA}, \quad (6)$$

where P_{VL-CSA} and P_{CSA} are power consumptions of the VL-CSA (at V_{VL-CSA}) and the conventional CSA (at V_{CSA}), respectively.

Based on the CSA structure, the numbers of CSS ’s in the two sequences of VL-CSA, m and k , satisfy:

$$(2q_{0,I} + k - 1) \cdot k / 2 + (2q_{0,II} + m - 1) \cdot m / 2 \approx M = 64, \quad (7)$$

By combining Eq. (1)-(7), a 64-bit VL-CSA with 13 CSS ’s is designed, as shown in the row “VL-CSA” of TABLE I. In each sequence, the number of input bits to a CSS roughly increases with the stage number. The number of input bits of CSS_8 and CSS_9 are 4 and 3, respectively. Pr_{Long} is only $1/(2^4 \cdot 2^3) \approx 0.78\%$ while $N_C = 2$. If we assume that $D_{carry} = D_{mux}$, $L_{Short} = D_{setup} + 11D_{carry} + D_{sum}$. The longest carry propagation paths of sequence I and II start from the first bit adder of CSS_1 or CSS_8 , cross the MUXes of sequential CSS ’s, and ends at the sum generation block of CSS_9 or CSS_{13} , respectively. $L_{Long} = D_{setup} + 15D_{carry} + D_{sum}$. For comparison, $D_{CSA} = D_{setup} + 13D_{carry} + D_{sum}$.

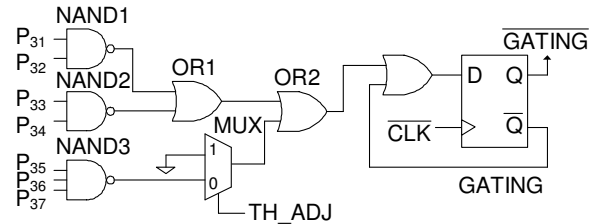


Fig. 2. CLDC design for VL-CSA.

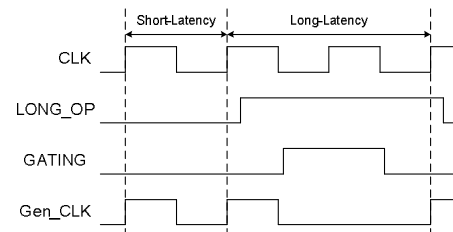


Fig. 3. Timing diagram of VL-adder operation.

C. Detection of Long Latency for VL-CSA

In our 64-bit VL-CSA design, the input bits of CSS_8 and CSS_9 (see TABLE I) are used to detect the long-latency operations. The detection logic can be expressed as:

$$LONG_OP = (A_{31} \oplus B_{31}) \cdot (A_{32} \oplus B_{32}) \cdot \dots \cdot (A_{37} \oplus B_{37}). \quad (8)$$

A long-latency operation occurs only when $LONG_OP = 1$. The corresponding circuit, called carry length detection circuit (CLDC), is shown in Fig. 2. When $LONG_OP$ signal is pulled up to ‘high’, a GATING signal is triggered to disable the clock (Gen_CLK) switching in the following clock cycle, as shown in Fig. 3. Here signal TH_ADJ is fixed at ‘low’.

D. NBTI-tolerable VL-CSA

Due to the NBTI effect, the circuit delay continuously increases with the degradation of PMOS threshold voltage [5]. We propose a modification of VL-adder of which the detection threshold of long/short-latency operations can be adaptively adjusted to account for NBTI-degraded delay: Operations with a NBTI-degraded delay that is longer than one clock cycle are automatically categorized as long-latency operations and are executed within two clock cycles.

Fig. 4 shows the structure of a modified 64-bit VL-CSA for NBTI-tolerance. A NBTI sensor (or called ‘Guardband Violation Sensor’ [6]) is added to the output flip-flops. ‘Guardband’ is defined as the reserved time period between the last possible data switching and the capturing clock edge, as shown in Fig. 5. When the circuit performance is degraded by NBTI effect, a guardband violation may occur as an output switch within the guardband.

The CLDC design is the same as that of the 64-bit VL-CSA. Before a guardband violation is caught, signal TH_ADJ is fixed at ‘low’. L_{Long} and L_{Short} equal $D_{setup} + 11D_{carry} + D_{sum}$ or

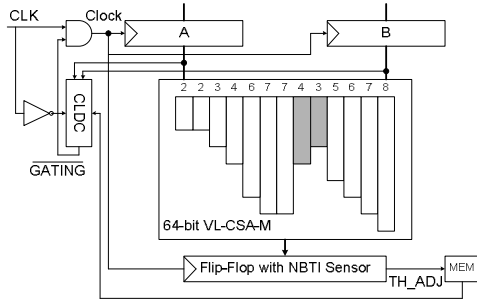


Fig. 4. A 64-bit VL-CSA for NBTI-tolerance.

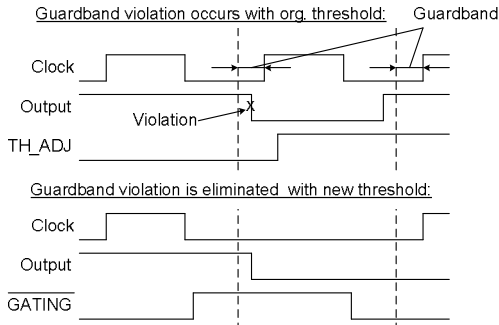


Fig. 5. Timing diagrams of VL-adder before and after detection threshold of long-latency operation changes.

$D_{setup} + 15D_{carry} + D_{sum}$, respectively, by assuming $D_{carry} = D_{mux} \cdot Pr_{Long}$, which equals the possibility of $LONG_OP = 1$, is 0.78% when the operands are random.

When a NBTI sensor detects a violation, signal TH_ADJ is raised to ‘high’. Pr_{Long} increases to $1/2^4 = 6.25\%$. The critical timing path of short-latency operation now starts from the input of CSS_1 to the output of CSS_8 , or from the input of CSS_8 to CSS_{13} . Hence, L_{Short} changes from $D_{setup} + 11D_{carry} + D_{sum}$ to $D_{setup} + 10D_{carry} + D_{sum}$ accordingly.

We note that only three guardband violation sensors, which are located at the outputs of CSS_9 , are required at the three output bits of CSS_9 .

III. EXPERIMENTAL RESULTS AND ANALYSIS

A. VL-Adder Implementation

To evaluate the area overhead of VL-adder design, a 64-bit conventional CSA and a 64-bit VL-CSA were implemented with TSMC 180nm technology. Every CSS is designed as a ripple-carry adder. After including NBTI sensor and CLDC (91 and 64 transistors, respectively), VL-CSA introduces a 3.97% area overhead, compared to the conventional CSA.

B. V_{DD} Scaling and Energy Savings

To evaluate the performance and power of VL-adder design, we generated the SPICE netlists of a 64-bit VL-CSA with NBTI sensor and CLDC, as well as the corresponding conventional CSA with PTM 70nm technology [7]. Three corners, ‘TT’, ‘SS’ and ‘FF’ are simulated. Fig. 6 shows the L_{Long} and L_{Short} of VL-CSA under different V_{DD} 's. The conventional CSA works at a V_{DD} of 1.0V. At ‘SS’ corner, L_{Short} equals $D_{CSA-70nm}$ when the V_{DD} of VL-CSA is 0.77V at ‘SS’ corner. Simulation results also show that $L_{Short} < L_{Long} < 2 \cdot L_{Short}$ under the simulated V_{DD} range. It means that the long-latency operations can always complete within two clock cycles.

We use NanoSim to simulate the energy consumptions of a conventional CSA and a VL-CSA with $2^{14} = 16384$ random inputs as shown in Fig. 7. Bar ‘Conv.’ denotes the energy consumption of the conventional CSA's at 1.0V. Compared to the conventional CSA, 44.4% total energy savings is achieved by VL-CSA with ~0.78% performance loss at ‘TT’ corner.

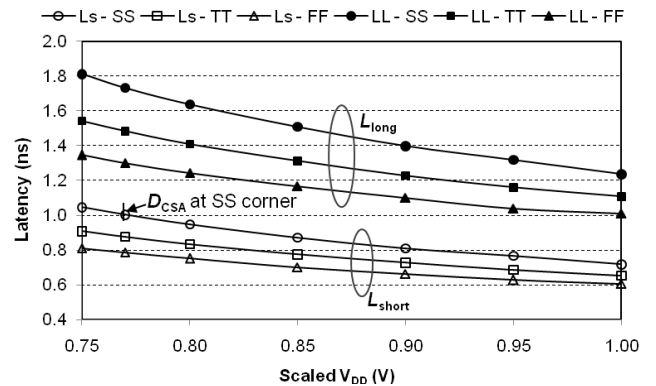


Fig. 6. 70nm Performance of VL-CSA under scaled V_{DD} 's.

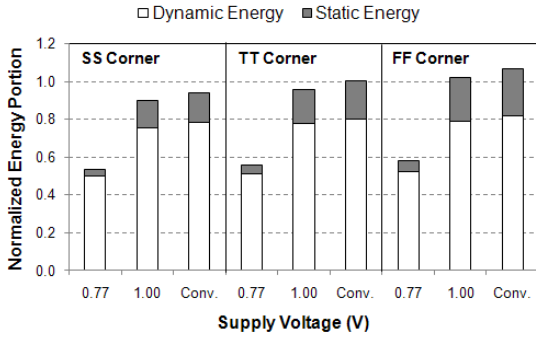


Fig. 7. Energy consumption of conventional CSA vs. VL-CSA.

C. VL-CSA for NBTI-tolerance

Our simulations used the NBTI model proposed in [8] and assumed a working temperature of 125 °C. The PMOS transistor threshold voltage V_{TH} at the beginning of the chip lifetime was 0.21V. Corner “TT” is selected to obtain the typical power saving results. The longest delays of both long- and short-latency operations of VL-CSA over the chip lifetime are shown in Fig. 8. Duty cycles (the portion of time when the PMOS transistor is on) of 0.25, 0.5 and 0.75 were simulated. To ensure the correct timing for a 7-year lifetime and a duty cycle of 0.75, the clock cycle T_{clk} should be at least 0.617ns.

We choose a lower V_{DD} of 0.94V than the original V_{DD} (1.0V) in our VL-CSA design for NBTI tolerance. The degradation of the corresponding L_{Short} is shown in Fig. 9. For a duty cycle of 0.75, signal TH_ADJ is raised to ‘high’ after the chip is in about 8 months (2×10^7 seconds) of operation. In practice, designers can increase the supply voltage (and hence, power) to delay the adjustment of the signal TH_ADJ.

Because the modification of VL-CSA for NBTI tolerance allows the adder to work at a lower V_{DD} , a higher energy-efficiency can be achieved even taking into account the increased Pr_{Long} after signal TH_ADJ is raised to ‘high’: The normalized PDP of a 64-bit VL-CSA for NBTI tolerance is always better than that of a 64-bit VL-CSA over a 7-year chip lifetime, as shown in Fig. 10. Here a total of $2^{14} = 16384$ random inputs were simulated. The power dissipation of CLDC adjustment circuitry has been accounted for.

We note that in reality, the probability of long-latency operation Pr_{Long} is much less than that of random inputs. Our simulation conducted by SimpleScalar shows that on average,

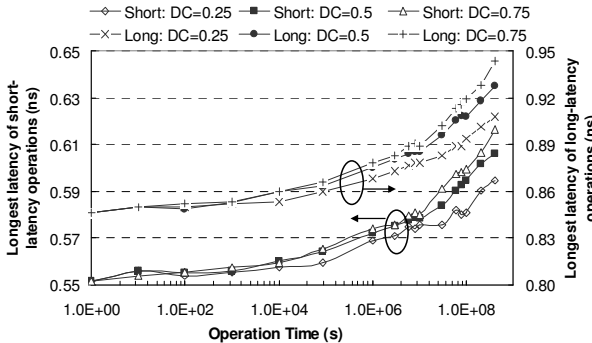


Fig. 8. Longest delays of long-/short -latency operations of a 64-bit VL-CSA.

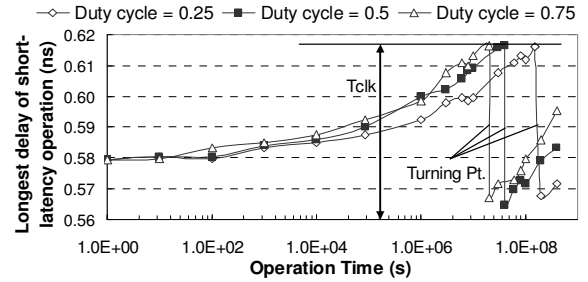


Fig. 9. Longest delay of short-latency operations of a 64-bit VL-CSA for NBTI tolerance.

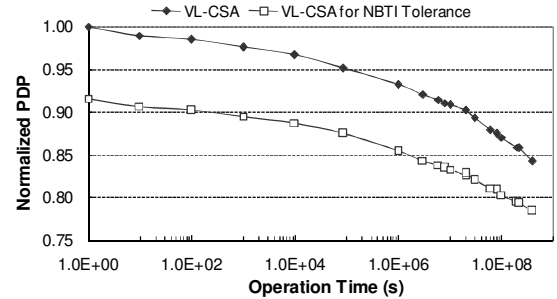


Fig. 10. PDPs of 64-bit VL-CSA and 64-bit VL-CSA for NBTI tolerance.

Pr_{Long} is respectively only 0.27% or 0.34%, before and after signal TH_ADJ is raised to ‘high’.

IV. CONCLUSION

We proposed a new adder design concept called “Variable Latency-Adder” (VL-adder) for low power and NBTI tolerance. When an operation with long-latency occurs, the data-capturing clock edge is postponed by one more cycle to allow for more computation time. Therefore, a VL-adder can work at a lower supply voltage while maintaining the similar throughput compared to a conventional adder. With a slight modification, VL-adder design can be used to overcome the NBTI-induced circuit performance degradation with further energy saving.

REFERENCES

- [1] J. Pouwelse, K. Langendoen, and H. Sips, “Dynamic voltage scaling on a low-power microprocessor,” in *Proc. the 7th Int’l Conf. on Mobile Computing and Networking*, Jul. 2001, pp. 251-259.
- [2] H. Suzuki, et al, “Low Power Adder with Adaptive Supply Voltage”, in *Proc. the 21st Int’l Conf. on Computer Design*, Oct. 2003, pp. 103-106.
- [3] J. M. Rabaey, *Digital Integrated Circuits: a design perspective*, Englewood Cliffs, NJ: Prentice Hall, 1996.
- [4] T.-Y. Chang and M.-J. Hsiao, “Carry-select Adder Using Single Ripple-carry Adder,” *IEE Electronics letters*, Vol. 34-22, pp. 2101-2103, Oct. 1998.
- [5] N. Kimizuka, et al, “The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on MOSFET scaling,” *VLSI Symp. on Tech.*, 1999, pp. 73-74.
- [6] M. Agarwal, et al, “Circuit Failure Prediction and Its Application to Transistor Aging,” in *Proc. 25th IEEE VLSI Test Symposium (VTS’07)*, May 2007, pp.277-286.
- [7] Y. Cao, et al, “New paradigm of predictive MOSFET and interconnect modeling for early circuit design,” in *Proc. IEEE Custom Integrated Circuits Conf.*, Jun. 2000, pp. 201-204.
- [8] K. Kang, et al, “Efficient Transistor-Level Sizing Technique under Temporal Performance Degradation due to NBTI,” in *Proc. IEEE Int’l Conf. on Computer Design*, 2006, pp. 216-221.