Transactions Briefs

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

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

***Abstract*—In this paper, 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 a 64-bit carry-select adder design, more than 40% energy saving is obtained when a similar throughput is maintained.**

***Index Terms*—Digital arithmetic, IC design, logic design.**

I. Introduction

Power supply voltage ($V_{\rm DD}$) scaling can effectively reduce both leakage and dynamic power of a VLSI circuit [1]. The value of $V_{\rm DD}$ is mainly determined by the desired operation frequency (or clock cycle period $T_{\rm clk}$) and the ***critical*** path delay, denoted by $T_{\rm crit}$. $V_{\rm DD}$ is selected such that $T_{\rm clk} \geq T_{\rm crit} + \delta$, 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 ($C_{\rm O}$) generated by the adder of the least significant bit (LSB) propagates through the adder of the most significant bit (MSB) [2]. The operands $\mathrm{A}\langle 31:0\rangle = 0\times\mathrm{FFFFFFFF}$ and $\mathrm{B}\langle 31:0\rangle = 0\times 00000001$ 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 are shifted by one more clock cycle so that the adder output can be latched on correctly. Compared to a conventional adder design, a VL-adder can work at a much lower $V_{\rm DD}$ 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 when a similar throughput is maintained.

Moreover, we proposed a modification of the 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 recategorized as long-latency operation, which is processed within two clock cycles.

Manuscript received July 03, 2008; revised January 30, 2009 and May 05, 2009. First published August 18, 2009; current version published October 27, 2010.

Y. Chen is with Seagate Technology, Bloomington, MN 55435 USA (e-mail: yiran.chen@seagate.com).

H. Li is with the Department of Electrical and Computer Engineering, Polytechnic Institute of NYU, Brooklyn, NY 11201 USA (e-mail: hai.helen.li@ieee.org).

C.-K. Koh, J. Li, and K. Roy are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: chengkok@purdue.edu; jingli@purdue.edu; kaushik@purdue.edu).

G. Sun and Y. Xie are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 USA (e-mail: gsun@cse.psu.edu; yuanxie@cse.psu.edu).

Digital Object Identifier 10.1109/TVLSI.2009.2026280

II. Concept of VL-Adder

A. Carry Propagation in a Carry-Select Adder

In a conventional design of M-bit square-root carry-select adder (CSA), the input bits are divided into $\approx \sqrt{2M}$ carry-select stages (CSSs), as shown in Fig. 1. One CSS includes carry generation blocks, a multiplexer (MUX) for carry selection, and a sum generation block. In CSS_i, two carry-out signals $\mathrm{C_{o0}}$ and $\mathrm{C_{o1}}$ of every bit are precalculated by assuming that $\mathrm{C}_{\mathrm{in},i}$, the carry-in signal of CSS_i, is 0 or 1, respectively. $\mathrm{C_{o0}}$ or $\mathrm{C_{o1}}$ is then selected by $\mathrm{C}_{\mathrm{in},i}$, the actual carry-out signal $\mathrm{C}_{\mathrm{out},i-1}$ of CSS_{i-1}, the previous CSS. The carry-out signal of the MSB adder in CSS_i is sent out as the carry-out signal $\mathrm{C}_{\mathrm{out},i}$ of CSS_i, which is also the carry-in signal $\mathrm{C}_{\mathrm{in},i+1}$ 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 CSSs, n is chosen as $(3+n)n/2 \approx M$. 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 $D_{\rm CSA}$ can be calculated by [3]

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

where $D_{\rm carry}$, $D_{\rm mux}$, and $D_{\rm sum}$ 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 CSSs; and $D_{\rm setup}$ is the setup time of CSA, which is the time taken to create the intermediate signals G (generation) and P (propagation) [3].

The delay of a CSA heavily depends on the input vectors, e.g., 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 $k = 1, 2, \ldots, m_i$ in CSS_i, the carry-out signal $\mathrm{C}_{\mathrm{out},i}$ of CSS_i is determined by the input bit pairs $(A_k, B_k, k = 1, 2, \ldots, m_i)$ of CSS_i only, regardless of the carry-in signal $\mathrm{C}_{\mathrm{in},i}(\mathrm{C}_{\mathrm{out},i-1})$ from CSS_{i-1}. Hence, the carry propagation through the MUX chain is *killed* at the MUX of CSS_i.

B. VL-CSA

A new VL CSA architecture called "VL-CSA" is shown in the row "VL-CSA" in Table I. The CSSs of VL-CSA are divided into two separate sequences: sequence I and sequence II. Sequence I and sequence II include $k\ (=7)$ and $m\ (=6)$ CSSs, respectively. In each sequence,

TABLE I
NUMBERS OF INPUT BITS OF CSSs IN A 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	***4***	***3***	5	*6*	*7*	*8*

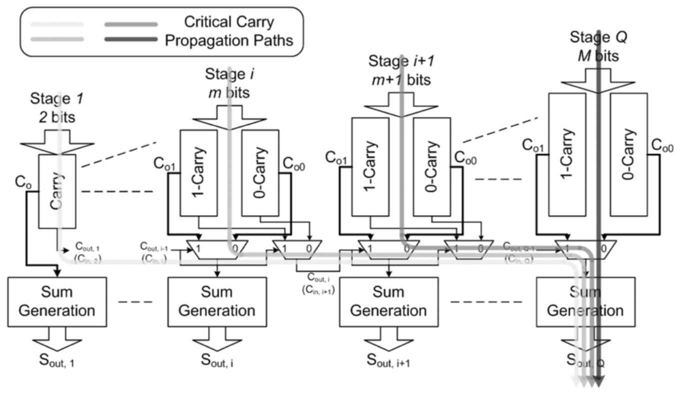

Fig. 1. Carry propagation in CSA.

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 CSSs and label them from 1 to $m+k$. 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 CSSs, 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 the conventional CSA, D_{CSA}.

However, the carry can propagate through the first N_C CSSs in sequence II, i.e., $\mathrm{CSS}_{k+1},\ldots,\mathrm{CSS}_{k+N_C}$, only if

$$P_{N_C} = \prod_{i \subset \{\mathrm{CCS}_{k+1},\ldots,\mathrm{CCS}_{k+N_C}\}} P_i = 1 \tag{2}$$

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

$$\mathrm{Pr}_{\mathrm{Long}} = \prod_{i=1}^{N_C} \frac{1}{2^{m_{\mathrm{i}}}} \tag{3}$$

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

$$L_{\mathrm{Short}} = \max(D_{\max 1}, D_{\max 2}) \tag{4}$$

where $D_{\max 1}$ and $D_{\max 2}$ are the longest delays of the first and the second parts, respectively. We have

$$\begin{aligned} D_{\max 1} &= D_{\mathrm{setup}} + q_{0,\mathrm{I}} D_{\mathrm{carry}} + (k+N_{\mathrm{C}}) D_{\mathrm{mux}} + D_{\mathrm{sum}} \\ D_{\max 2} &= D_{\mathrm{setup}} + q_{0,\mathrm{II}} D_{\mathrm{carry}} + m D_{\mathrm{mux}} + D_{\mathrm{sum}}. \end{aligned} \tag{5}$$

Here $q_{0,\mathrm{I}}$ and $q_{0,\mathrm{II}}$ are the numbers of input bits in the first CSSs 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 $V_{\mathrm{CSA}}, L_{\mathrm{Short}} < D_{\mathrm{CSA}}$, 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 $V_{\mathrm{VL-CSA}} (< V_{\mathrm{CSA}})$, at which the longest latency of all short-latency operations of the VL-CSA $L_{\mathrm{Short-VL}}$ is exactly D_{CSA}, the PDP of VL-CSA is better than that of the conventional CSA at V_{CSA}

$$P_{\mathrm{VL-CSA}}(1+\mathrm{Pr}_{\mathrm{Long}}) < P_{\mathrm{CSA}} \tag{6}$$

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

Based on the CSA structure, the numbers of CSSs in the two sequences of VL-CSA, m and k, satisfy

$$(2q_{0,\mathrm{I}} + k - 1)\frac{k}{2} + (2q_{0,\mathrm{II}} + m - 1)\frac{m}{2} \approx M = 64. \tag{7}$$

By combining (1)–(7), a 64-bit VL-CSA with 13 CSSs is designed, as shown in the row "VL-CSA" of Table I. In each sequence, the number of input bits of a CSS roughly increases with the stage number. The number of input bits of CSS_8 and CSS_9 are 4 and 3, respectively. $\mathrm{Pr}_{\mathrm{Long}}$ is only $1/(2^4 \times 2^3) \approx 0.78\%$ while $N_C = 2$. If we assume that $D_{\mathrm{carry}} = D_{\mathrm{mux}}, \mathrm{then} L_{\mathrm{short}} = D_{\mathrm{setup}} + 11 D_{\mathrm{carry}} + D_{\mathrm{sum}}$. The longest carry propagation paths of sequences I and II start from the first bit adder of CSS_1 or CCS_8, cross the MUXes of sequential CSSs, and ends at the sum generation block of CSS_9 or CSS_{13}, respectively. $L_{\mathrm{Long}} = D_{\mathrm{setup}} + 15 D_{\mathrm{carry}} + D_{\mathrm{sum}}$. For comparison, $D_{\mathrm{CSA}} = D_{\mathrm{setup}} + 13 D_{\mathrm{carry}} + D_{\mathrm{sum}}$.

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 follows:

$$\mathrm{LONG\_OP} = (A_{31} \oplus B_{31})(A_{32} \oplus B_{32}) \cdots (A_{37} \oplus B_{37}). \tag{8}$$

A long-latency operation occurs only when $\mathrm{LONG\_OP} = 1$. The corresponding circuit, called carry length detection circuit (CLDC), is shown in Fig. 2. When an LONG_OP signal is pulled up ''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 in which the detection threshold of long/short-latency operations can be adaptively adjusted to account for NBTI-degraded delay: operations with an NBTI-degraded delay that is longer than one clock cycle are automatically categorized as long-latency operations and executed within two clock cycles.

Fig. 4 shows the structure of a modified 64-bit VL-CSA for NBTI tolerance. An NBTI sensor (or called "guard band violation sensor"

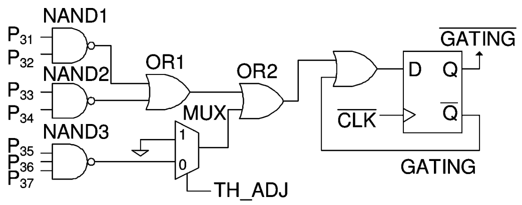

Fig. 2. CLDC design for VL-CSA.

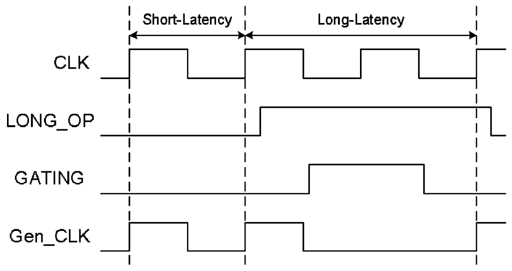

Fig. 3. Timing diagram of VL-adder operation.

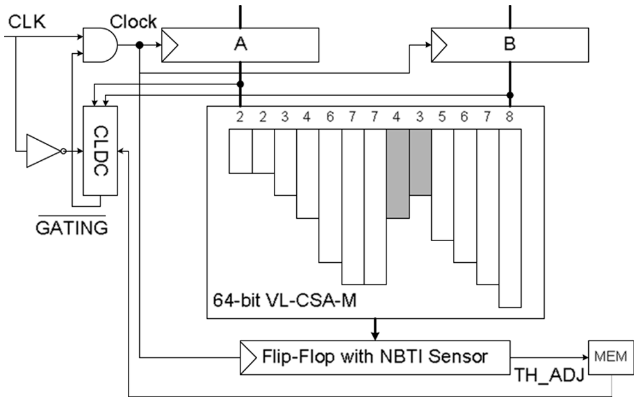

Fig. 4. 64-bit VL-CSA for NBTI tolerance.

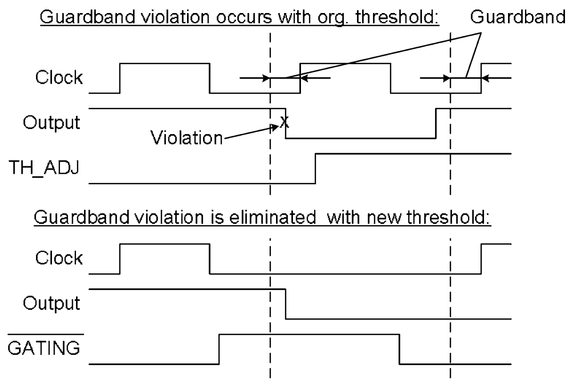
Guardband violation occurs with org. threshold:
Guardband
Clock
Output
Violation
TH_ADJ
Guardband violation is eliminated with new threshold:
Clock
Output
GATING

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

[6]) is added to the output flip-flops. "Guard band" 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 guard band violation may occur as an output switch within the guard band.

The CLDC design is the same as that of the 64-bit VL-CSA. Before a guard band violation is caught, signal TH_ADJ is fixed at "low." L_{Long} and L_{Short} equal $D_{\mathrm{setup}} + 11D_{\mathrm{carry}} + D_{\mathrm{sum}}$ or $D_{\mathrm{setup}} + 15D_{\mathrm{carry}} + D_{\mathrm{sum}}$, respectively, by assuming $D_{\mathrm{carry}} = D_{\mathrm{mux}}$. $\mathrm{Pr}_{\mathrm{Long}}$, which equals the possibility of $\mathrm{LONG\_OP} = 1$, is 0.78% when the operands are random.

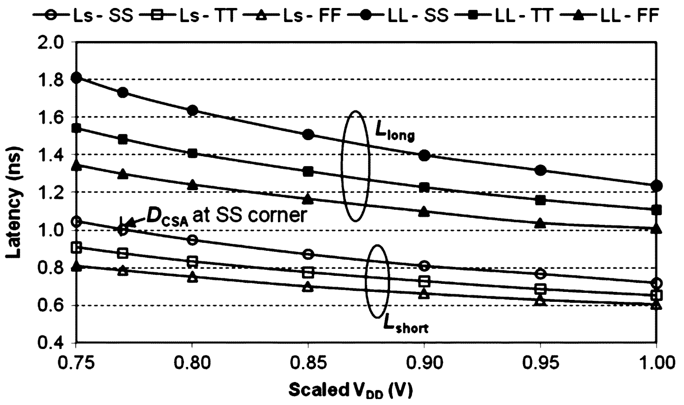

Fig. 6. 70-nm performance of VL-CSA under scaled V_{DD}'s.

When an NBTI sensor detects a violation, signal TH_ADJ is raised to "high." $\mathrm{Pr}_{\mathrm{Long}}$ increases to $1/2^4 = 6.25\%$. The critical timing path of the 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 $D_{\mathrm{setup}} + 11D_{\mathrm{carry}} + D_{\mathrm{sum}}$ to $D_{\mathrm{setup}} + 10D_{\mathrm{carry}} + D_{\mathrm{sum}}$ accordingly.

We note that only three guard band 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 180-nm 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 predictive technology model (PTM) 70-nm technology [7]. Three corners, "TT," "SS," and "FF" are simulated. Fig. 6 shows the L_{Long} and the L_{Short} of VL-CSA under different V_{DD}'s. The conventional CSA works at a V_{DD} of 1.0 V. At the "SS" corner, L_{Short} equals $D_{\mathrm{CSA}} - 70$ nm when the V_{DD} of VL-CSA is 0.77 V at "SS" corner. Simulation results also show that $L_{\mathrm{Short}} < L_{\mathrm{Long}} < 2L_{\mathrm{Short}}$ 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 CSAs at 1.0 V. Compared to the conventional CSA, 44.4% total energy savings is achieved by VL-CSA with $\sim 0.78\%$ performance loss at the "TT" corner.

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 0.21 V. Corner

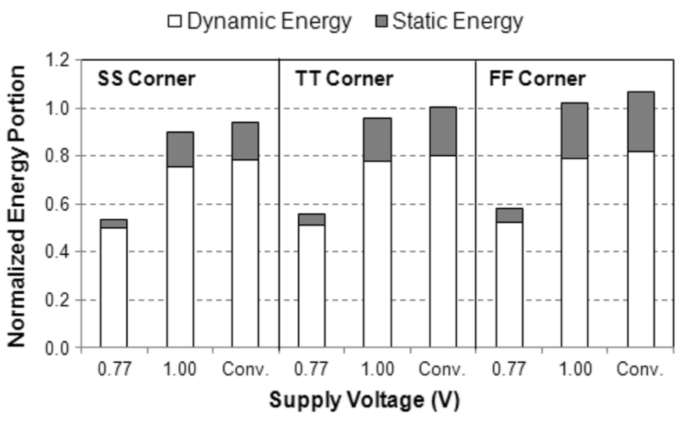

Fig. 7. Energy consumption of conventional CSA versus VL-CSA.

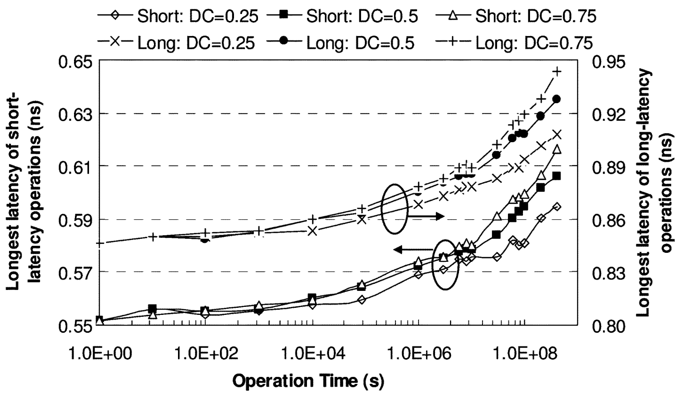

Fig. 8. Longest delays of long-/short-latency operations of a 64-bit VL-CSA.

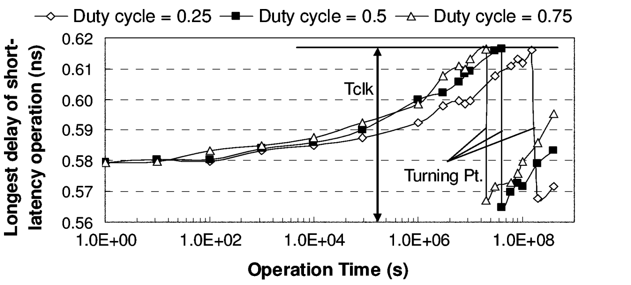

Fig. 9. Longest delay of short-latency operations of a 64-bit VL-CSA for NBTI tolerance.

"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 $T_{\rm clk}$ should be at least 0.617 ns.

We choose a lower $V_{\rm DD}$ of 0.94 V than the original $V_{\rm DD}$ (1.0 V) in our VL-CSA design for NBTI tolerance. The degradation of the corresponding $L_{\rm Short}$ 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 eight months (2×10^7 s) of operation. In practice, designers can increase the supply voltage (and hence, power) to delay the adjustment of the signal TH_ADJ.

Since the modification of VL-CSA for NBTI tolerance allows the adder to work at a lower $V_{\rm DD}$, a higher energy efficiency can be achieved even taking into account the increased $\Pr_{\rm Long}$ after signal TH_ADJ is raised to ''high'': The normalized programmed data processor (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.

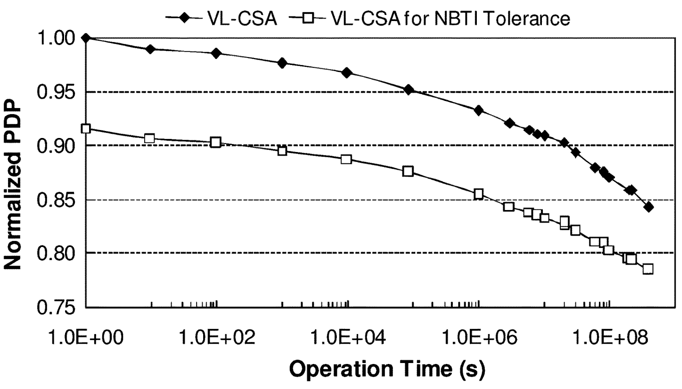

Fig. 10. PDPs of a 64-bit VL-CSA and a 64-bit VL-CSA for NBTI tolerance.

We note that in reality, the probability of the long-latency operation $\Pr_{\rm Long}$ is much less than that of random inputs. Our simulation conducted by SimpleScalar shows that on average, $\Pr_{\rm Long}$ 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 "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 when maintaining the similar throughput compared to a conventional adder. With a slight modification, the 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. 7th Int. Conf. Mobile Comput. Netw.*, Jul. 2001, pp. 251–259.
[2] H. Suzuki, W. Jeong, and K. Roy, "Low power adder with adaptive supply voltage," in *Proc. 21st Int. Conf. Comput. Des.*, 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," *Inst. Electr. Eng. Electron. Lett.*, vol. 34–22, pp. 2101–2103, Oct. 1998.
[5] N. Kimizuka, T. Yamamoto, T. Mogami, K. Yamaguchi, K. Imai, and T. Horiuchi, "The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on MOSFET scaling," in *Proc. Very Large Scale Integr. Symp. Techol.*, 1999, pp. 73–74.
[6] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *Proc. 25th IEEE Very Large Scale Integr. Test Symp. (VTS)*, May 2007, pp. 277–286.
[7] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. IEEE Custom Integr. Circuits Conf.*, Jun. 2000, pp. 201–204.
[8] K. Kang, H. Kufluoglu, M. A. Alain, and K. Roy, "Efficient transistor-level sizing technique under temporal performance degradation due to NBTI," in *Proc. IEEE Int. Conf. Comput. Design*, 2006, pp. 216–221.