A Pipeline Architecture with 1-Cycle Timing Error Correction for Low Voltage Operations

Insup Shin[1], Jae-Joon Kim[2], Yu-Shiang Lin[3], and Youngsoo Shin[1]
[1]Department of EE, KAIST, Daejeon 305-701, Korea
[2]Department of Creative IT Engineering, POSTECH, Pohang 790-784, Korea
[3]IBM T. J. Watson Research Center, Yorktown Height, NY 10598, USA

***Abstract*— We present a new timing error correction scheme which allows each pipeline stage to halt for one cycle only. The small timing penalty for the error correction operation in the proposed scheme makes it possible to eliminate the extra timing guardband that was needed to accommodate timing uncertainty due to process variations. As a result, lower supply voltage can be used with the proposed scheme for low power operations. Compared to the previous 1-cycle error correction scheme which uses two-phase transparent latch based pipeline [1], the proposed scheme can be applied to the pipeline based on more popular clocking elements such as flip-flop or pulsed latch.**

I. Introduction

Traditionally, delay increase due to the variability in digital circuits has been handled by addition of timing guardband to the original cycle time. As technology scales down, however, the magnitude of variations increases significantly so that the amount of required timing guardband becomes substantial. As a result, the supply voltage needs to be set higher to accommodate the excessive timing guardband, and the higher supply voltage causes the larger power consumption. To reduce the extra power consumption, adaptive techniques have been gaining attention as they can reduce timing guardband by dynamically changing supply voltage or clock frequency.

Adaptive designs are typically composed of the sensors which monitor the magnitude of the variation and control logics which dynamically adjust supply voltage and clock frequency using the sensor outputs. Several methods have been proposed for this purpose: on-chip variation sensors, canary-circuits, and error detection sequential (EDS) circuits. On-chip sensors [2], [3] measure supply voltage droop or temperature of the chip. Canary-circuits [4], [5] measure delay increase in replica of critical path of the chip. While these schemes are effective to measure global and static variations, they cannot detect temporal variations due to dynamic noise and/or local variations. In contrast, EDS circuits [1], [6]–[9] detect variations on actual critical path so that timing guardband against temporal variations and local variations can be eliminated.

In the embedded EDS design, the data is captured by shadow latch with the delayed clock as well as by the main latch with the nominal clock [7]. If the shadow latch data is different from the main latch when compared, an error is flagged and the error is corrected by error correction circuits. Since EDS-style systems can detect and correct timing errors, they can run below the minimum voltage necessary for error-free operation at given frequency, which allows significant power savings.

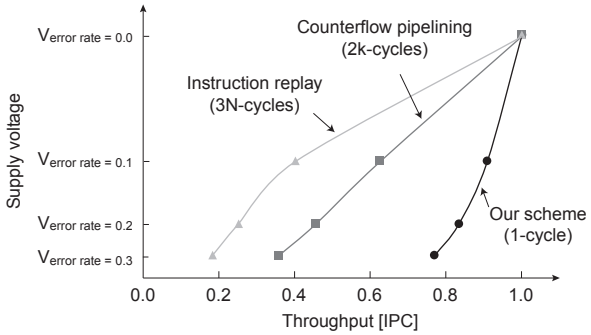

Fig. 1. Relationship between supply voltage and throughput for our scheme, counterflow pipelining, and instruction replay when a pipeline design has 5 stages ($N = 5$, $k = 3$).

Since the overall timing penalty depends on the number of cycles for error correction in EDS-style schemes, there is a great need to minimize the number of error correction cycles. Several error correction schemes have been proposed to correct timing errors. Global clock gating [6] has only 1-cycle timing penalty but its applicability is limited because clock gating signal must be propagated to all stages in a cycle. Counterflow pipelining [6] does not need global clock gating signal but it takes multiple clock cycles for error correction. Its timing penalty per error correction is $2k$-cycles where k is the order of the stage, which detects an error, in the pipeline. Thus, the completion time of the next instruction following the failed instruction is delayed by $2k$-cycles.

Meanwhile, the instruction replay scheme [8] has the smallest design overhead among error correction schemes but it needs more clock cycles for error correction than counterflow pipelining. Its timing penalty is $3N$-cycles where N is the number of pipeline stages.

In order to see how important reducing the timing penalty of error correction schemes is for low voltage design, we plot the relationship between supply voltage and pipeline throughput (IPC) for three different schemes (instruction replay, counterflow pipelining, and our proposed scheme with 1-cycle penalty) in Fig. 1. We assume that the design has 5-stage ($N = 5$) and errors always occur at stage 3 ($k = 3$). Error correction scheme with smaller timing penalty has larger

maximum tolerable error rate under the same throughput, and thus can be made to operate with lower supply voltage. If target throughput is 0.8, the maximum tolerable error rates for counterflow pipelining and instruction replay are 0.04 and 0.016 respectively. In contrast, the maximum torelable error rate for the 1-cycle error correction is 0.25. Therefore, we can obtain more energy savings with 1-cycle error correction scheme by using lower supply voltage.

In addition, the timing penalties per error correction of counterflow pipelining and instruction replay are dependent on the number of pipeline stages so that throughput degradation becomes larger as the number of pipeline stages increases.

The error correction mechanism proposed in [1] made a breakthrough by reducing the penalty down to one cycle. However, it can only be used for two-phase transparent latch based designs. Since flip-flop and pulsed-latch are more popular choices for clocking elements in current digital circuit design methodologies than level sensitive latch, there is a great need for 1-cycle error correction in the flip-flop and/or pulsed-latch based EDS-style schemes.

In this paper, we propose new error correction scheme which has only 1-cycle penalty for flip-flop or pulsed-latch based design. Since it is based on local stalling, it can be used for high-frequency or complex designs unlike global clock gating.

The remainder of this paper is organized as follows. We review previous error correction schemes in Section II. Our 1-cycle error correction scheme is presented in Section III. Experimental results are provided in Section IV and Section V concludes the paper.

II. Previous Error Correction Schemes

Conceptually, the simplest error correction scheme is global clock gating [6]. When a stage detects an error, all pipeline stages are stalled for one cycle and shadow latch data is restored into main flip-flop. However the clock gating signal may take multiple cycles to be propagated to all stages in the high-frequency or complex designs. Several approaches have been introduced to overcome the drawback of global clock gating, e.g. counterflow pipelining [6], instruction replay [8], [10], and Bubble Razor [1].

Counterflow pipelining [6] performs local stalling of the stage which detects the error for one cycle, and thus does not need global clock gating signal. The stage which detected an error sends clock gating signals to its output stages and flush signals to its input stages. These signals are propagated to other stages in a wave-like fashion. When the flush signal reaches the first stage, the next-cycle instruction to the failed instruction is issued again. If an error is detected at k-th stage, the completion time of the next-cycle instruction is delayed by $2k$. In an example shown in Fig. 2(a), instruction $i3$ could have completed in cycle 7 if there was no error. When an error occurs at stage C in cycle 4, $i3$ is reissued in cycle 9 and is completed in cycle 13 (completion time of $i3$ is delayed by $2k = 6$ cycles).

Instruction replay [8], [10] has the smallest design overhead. When an error occurs, the error signal is propagated to the last

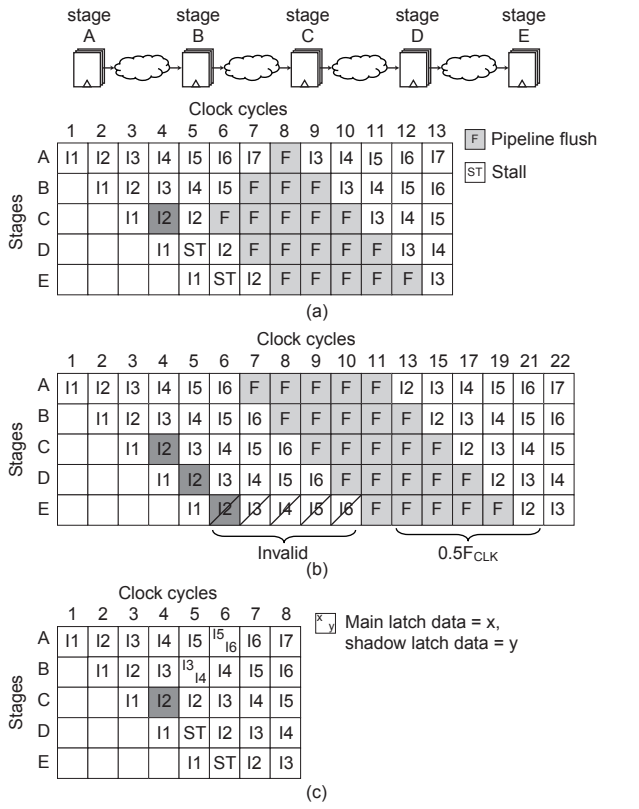

Fig. 2. Error correction examples for (a) counterflow pipelining, (b) instruction replay at half the clock frequency, and (c) our scheme when an error occurs at stage C in cycle 4.

pipeline stage. When the error signal reaches the last stage, flush signal is issued for flushing entire pipeline stages. After the pipeline flushing, the failing instruction is replayed at half the clock frequency. The number of cycles needed for pipeline flushing is N and the number of cycles needed for the replay of the failing instruction is $2N$. Thus, the completion time of the next-cycle instruction is delayed by $3N$. Error correction example of instruction replay is illustrated in Fig. 2(b). When the error signal reaches the last stage, instruction replay begins with flushing the pipeline. During the replay of instruction $i2$, clock frequency is reduced by half so that $i2$ is completed in cycle 21. The completion time of instruction $i3$ is therefore delayed from cycle 7 to cycle 22.

III. 1-Cycle Error Correction Scheme

Since the data in the shadow latch is a correct one even in the failing stage, the simplest method for error correction is restoring the data in the shadow latch into the main latch. The only problem with this method is that the data coming from the input stage will be lost during the restore cycle. That is the reason why the counterflow pipelining reissues the next instruction to the failed instruction after the error correction [6]. Our main idea to solve this problem is to gate the main latch only for the previous stages to the failed stage without gating its shadow latch. If the main latch of a stage is gated while shadow latch is being clocked, the stage can capture input data (at the shadow latch) and retain the previous

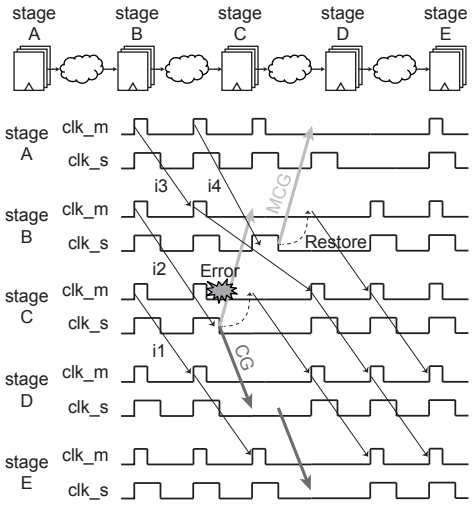

Fig. 3. Error correction example when proposed scheme is used.

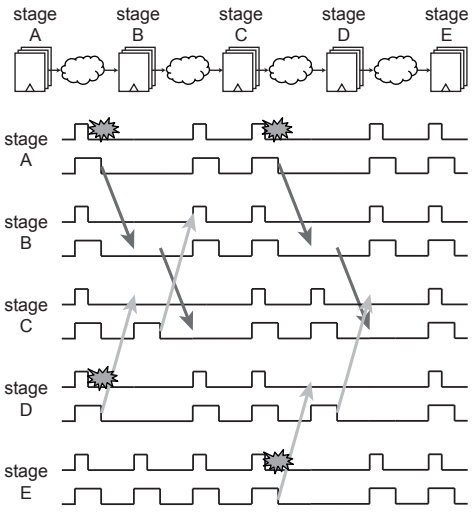

Fig. 4. Propagations of CG and MCG are stopped when they are met or crossed.

data (at the main latch) at the same time. Thus, the stage which detects error can receive the correct data in the next cycle after error correction. As illustrated in the example in Fig. 2(c), the main latch in the stage B retains instruction $i3$ and the shadow latch in the stage B stores instruction $i4$ in cycle 5. As a result, $i3$ can be propagated to the stage C in the next cycle without causing error.

A. Gating Signal Propagation

In order to maintain correctness of the data, each previous stage to the one where the error happened, eventually goes through 2-cycle process in which the main latch is gated in the first cycle and data in its shadow latch is restored into its main latch in the second cycle. In addition, we should prevent the propagation of incorrect data from the stage in which timing error happened. To accomplish these, we introduce two types of clock gating control signals: CG and MCG. When a stage receives CG signal, the clock for its main latch, which is denoted by clk_m, and the clock for its shadow latch, which is denoted by clk_s, are gated for one cycle. CG signal is propagated from the stage where error occurs to its output stages in a wave-like fashion. When a stage receives MCG signal, its clk_m is gated for one cycle and then both clk_m and clk_s are gated in the next cycle. Similar to the way the CG signals are propagated, MCG signals are propagated to the previous input stages in a wave-like fashion.

An example of propagation of CG and MCG is shown in Fig. 3. Assume that an error occurs at stage C in cycle 2. In cycle 3, instruction $i2$ is restored at the stage C by passing the correct data in the shadow latch to the main latch. In the same cycle, the main latch in the stage B is gated to prevent data loss while its shadow latch still receives the data from the stage A. The stage D must be stalled in cycle 3 because its input data from the stage C is incorrect. In cycle 4, the instruction $i3$ which has already arrived at cycle 3 is captured into stage C. The instruction $i4$ in the shadow latch in the stage B is restored into its main latch and the stage E is stalled to prevent double sampling of the instruction $i1$.

When multiple errors occur, CG and MCG can meet or cross each other. In these cases, propagation of CG and MCG must be stopped to maintain proper operation. We, therefore, have two stop conditions to take care of those cases. First, if a clock gating signal is propagated to the stage which received the clock gating signal in the previous cycle, clock gating signal stops propagation at the stage. In Fig. 4, stage C receives CG signal from stage B in cycle 3, but since main latch of stage C was gated in cycle 2, this CG signal is nullified and it is not propagated to stage D. The second condition is that, if CG and MCG are propagated to same stage, main latch and shadow latch are gated but propagation of CG and MCG are both stopped. In Fig. 4, stage C receives CG and MCG in cycle 6. Thus, main latch and shadow latch are gated in cycle 6, and propagations of two signals are stopped.

B. Extension to General Cases

Our error correction scheme can be extended to more general cases in which there are loops or multiple fan-out and multiple fan-in in the pipeline. In case of multiple fan-out and multiple fan-in, there are two problems that need to be addressed.

The first one is data loss at a multiple fan-in stage when not all the input stages have sent CG signal. Consider a pipeline architecture that has multiple fan-in stage and multiple fan-out stage shown in Fig. 5. Assume that an error occurs at stage A as shown in Fig. 5(a). Stage B sends CG signals to its output stages, stage C and stage E. Stage E is therefore stalled in cycle 3; this results in loss of instruction $i2$ sent from stage D. This problem can be solved by modifying the propagation algorithm as follows: if a stage receives CG signal from any of its input stages, it sends MCG signals to all of its input stages in the same cycle as shown in Fig. 5(b). Now each input stage of the multiple fan-in stage will stall for a cycle and propagate the data in the next cycle, and hence the pipeline can maintain proper data synchronization.

The second problem occurs at multiple fan-out stage when

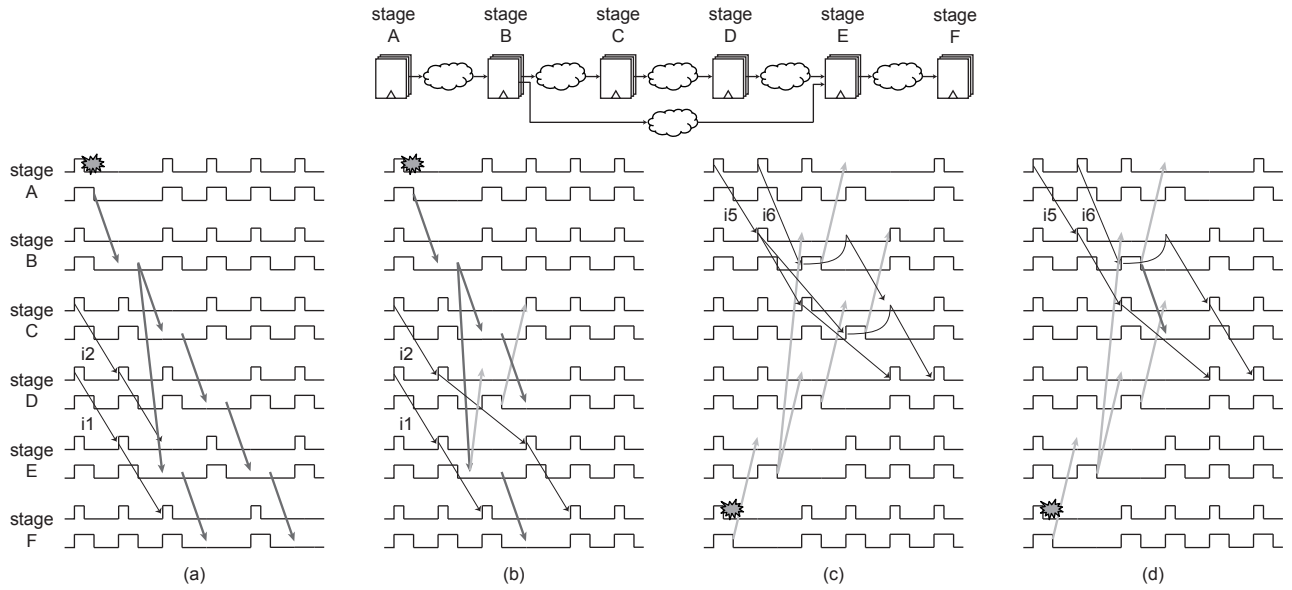

Fig. 5. A pipeline circuit which has multiple fan-out stage and multiple fan-in stage. (a) When a data lost problem occurs at stage E, (b) the result when our propagation algorithm is applied, (c) when double sampling problem occurs at stage C, and (d) the result when our propagation algorithm is applied.

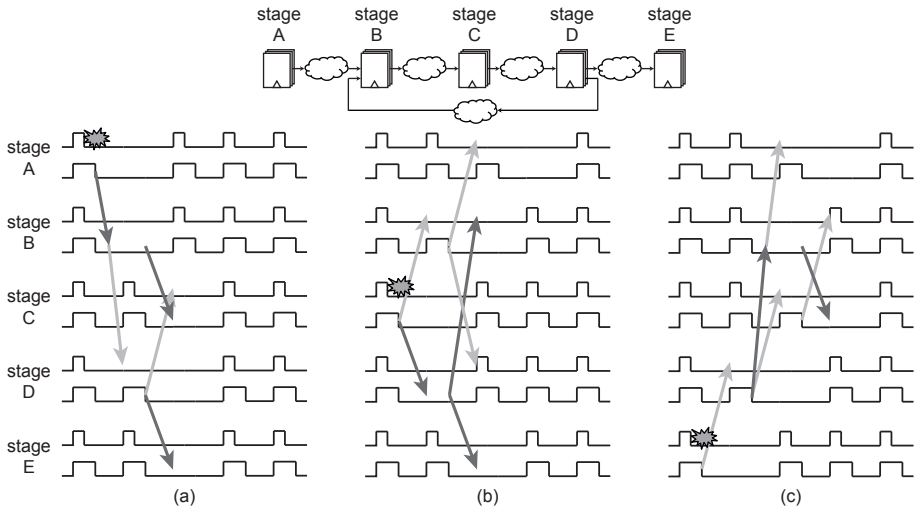

Fig. 6. Loop case. (a) Error occcurs before the loop, (b) error occurs in the loop, and (c) error occurs after the loop.

not all the output stages have sent MCG signal. Assume that an error occurs at stage F as shown in Fig. 5(c). In cycle 3, stage B receives MCG signal from stage E and then clk_m of stage B is gated to retain the previous data $i5$. Therefore $i5$ is double sampled and $i6$ is lost at stage C. This problem can be also solved by extending the propagation algorithm as follows: if a stage receives MCG signal from any of its output stages, it sends CG signal to all of its output stages in the next cycle.

Our error correction scheme can also handle loop condition. The main challenge is to prevent indefinite looping. Since CG and MCG are propagated in opposite directions and they always meet each other within the loop, propagation of CG and MCG stops and indefinite looping does not occur. To verify this, we show three examples for the loop conditions in Fig. 6: 1) an error occurs before the loop, 2) an error occurs in the loop, and 3) an error occurs after the loop.

The first case is illustrated in Fig. 6(a). Assume that an error occurs at stage A. CG signal is inserted into the loop. CG signal and MCG signal meet each other at stage C and therefore, propagation of CG and MCG are stopped at stage C. In addition, CG signal is propagated to stage E which is outside of the loop and then the signal is propagated to the upstream stages in a wave-like fashion.

The second case is illustrated in Fig. 6(b); an error occurs at stage C. CG signal and MCG signal are propagated in opposite directions in the loop and they cross each other at stage B and stage D. Following the stop condition of clock gating signal propagations, the propagation of CG and MCG stops at the stages B and D.

The last case is illustrated in Fig. 6(c); an error occurs at stage E. MCG signal is propagated back to the loop. Stage D

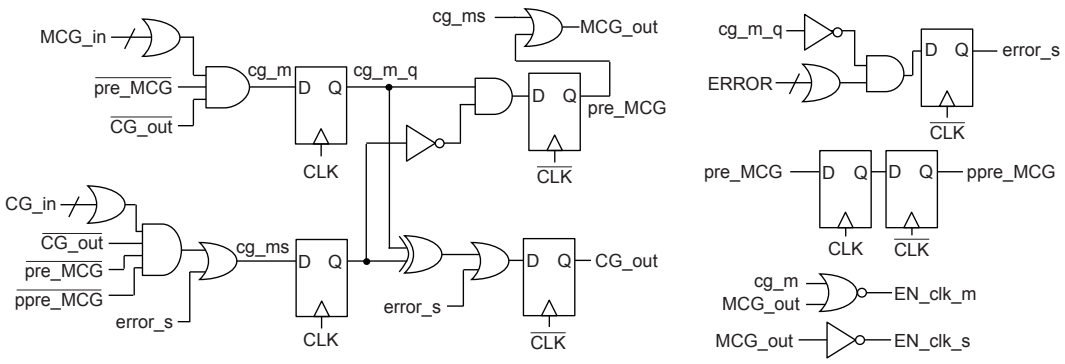

Fig. 7. Schematic of control logic.

receives the signal and sends CG signal to stage B and MCG signal to stage C. Those signals cross each other at stage B and stage C so that propagation of CG and MCG stops at these stages.

IV. Experimental Results

In our experiment, pulsed-latch based Razor circuits are used. We generated two 32-bit pipeline circuits by using ISCAS benckmark circuits. The first one has 5-stages based on c1908 circuits. The second one has 10-stages based on c3540 circuits. Each circuit was synthesized with commercial logic synthesis tool [11] using the 45-nm Open Cell Library [12]. The pulse width of main latch is 230 ps, which is the minimum pulse width, and that of shadow latch is 480 ps. Thus, the width of error detection window is 250 ps. Area increase from the original design without Razor latch due to the increased number of the buffer to account for the larger hold time violations caused by the wider clock pulse width for shadow latch is 31% for the 5-stage design and 24% for the 10-stage design. Energy consumption of 5-stage circuit and 10-stage circuit were measured with a clock period of 1.6 ns and 2.1 ns respectively. We used a fast SPICE simulator [13] to measure the energy consumption.

The schematic of control logic for our error correction scheme is illustrated in Fig. 7. All sequential elements in the figure are transparent latches. When a stage receives CG signal from any of its input stages, node cg_ms becomes high and then MCG_out is high. As a result, MCG signals are propagated back to its output stages in the same cycle. When a stage receives MCG signal from any of its output stages, the outputs of XOR2 gate and AND2 gate become high. Therefore, both MCG signal and CG signal are propagated to its neighbor stages in the next cycle. Nodes $\overline{pre\_MCG}$, $\overline{CG\_out}$, and $\overline{ppre\_MCG}$ are for the stop conditions.

We compared our error correction scheme with counterflow pipelining [6] and instruction replay at half the clock frequency [10]. Fig. 8 shows the throughput and energy per instruction as a function of the supply voltage. The results for 5-stage circuit are illustrated in Fig. 8(a) and Fig. 8(b). When error rate is 0 (when supply voltage is larger than 0.95 V), the instruction replay technique has the smallest energy consumption because it has the simplest control logic.

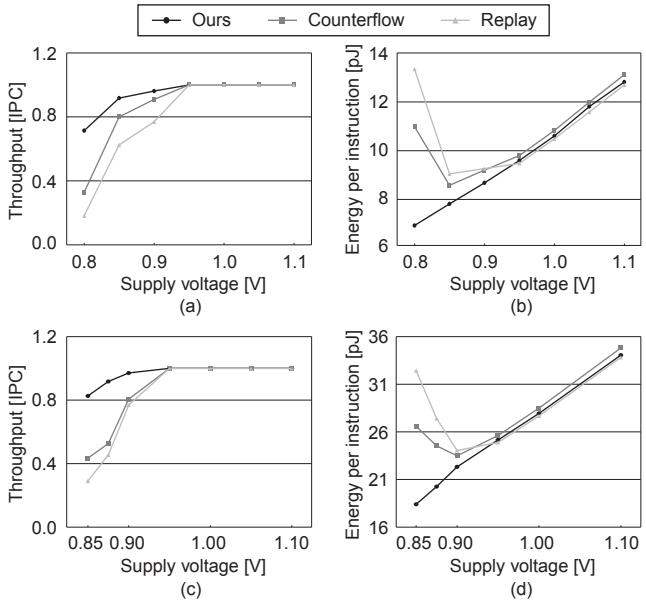

Fig. 8. (a) Throughput and (b) energy consumption per instruction for 5-stage circuit versus supply voltage and (c) throughput and (d) energy consumption per instruction for 10-stage circuit versus supply voltage.

However, since the instruction replay has the largest timing penalties for error correction, its throughput is significantly degraded as the error rate increases. Since dynamic power is still consumed during error correction cycle, energy per instruction increases as shown in Fig. 8(b).

Counterflow has the largest design overhead because each stage has to have its own flush logic. Thus, when error rate is 0, it has the largest energy per instruction as shown in Fig. 8(b). However, since it has smaller timing penalty than instruction replay, additional energy consumption during error correction is smaller than that of instruction replay.

Note that the energy overhead in our scheme is significantly lower than that of the other techniques. The optimal voltage in our scheme, which corresponds to the minimum total energy consumption, is lower than that of the other techniques as shown in Fig. 8(b).

The results for 10-stage circuit are illustrated in Fig. 8(c) and Fig. 8(d). As the number of pipeline stage increases, the timing penalty for error correction of counterflow pipelining and instruction replay techniques also increases. Thus, overall

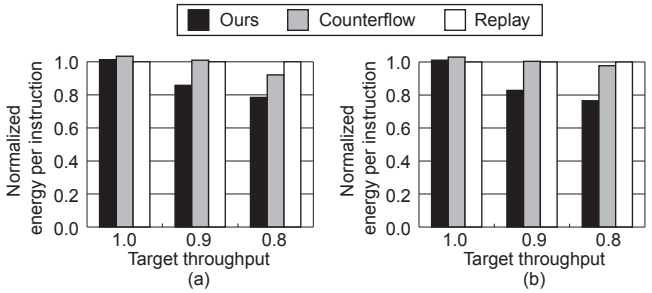

Fig. 9. Comparison of normalized energy consumption per instruction of our scheme (left bars), counterflow pipelining (middle bars), and instruction replay (right bars) for (a) 5-stage circuit and (b) 10-stage circuit.

timing penalty and energy overhead for those two techniques increases much faster in Fig. 8(c) and Fig. 8(d) than those in Fig. 8(a) and Fig. 8(b). The optimal voltages for these two techniques are, therefore, slightly higher than those in 5-stage circuit.

Fig. 9 shows the normalized energy per instruction for 5- and 10-stage circuits. Target throughputs are set to 1.0, 0.9, and 0.8. For each case, energy per instruction of our scheme and counterflow pipelining are normalized by that of instruction replay. In case of 5-stage circuit, the energy consumption is reduced by 14% and 22% (compared with instruction replay) when target throughputs are 0.9 and 0.8 respectively. In case of 10-stage circuit, the energy consumption is reduced by 17% and 24% (compared with instruction replay) when target throughputs are 0.9 and 0.8 respectively. When error rate is 0 (target throughput is 1), the energy consumption of our scheme is larger than that of instruction replay by only 1%.

It is worthwhile to note that all the latches must be replaced with Razor latches in our proposed scheme because the shadow latch is used for error correction as well as the error detection. On the other hand, the counterflow and replay scheme can use Razor latches for part of the pipeline only because shadow latch is used for error detection only. Therefore, for a fair comparison, we replaced only 50% and 25% of latches with Razor latches in the counterflow scheme and replay scheme and performed the same experiment again. For this experiment, we generated a 32-bit 5-stage pipeline circuit based on c6288 circuit. Area increase from the original design without Razor latch is 16%.

The results are illustrated in Fig. 10. Target throughputs are set to 1.0, 0.9, and 0.8. For each case, energy per instruction of our scheme, counterflow, and instruction replay are normalized by that of instruction replay when all latches are replaced with Razor latches. It can be seen that energy consumption in our proposed scheme is larger than the counterflow and replay scheme when there is no timing error (target throughput = 1) due to the larger number of shadow latches. However, our proposed scheme consumes less power than the other two schemes as the target throughput decreases as shown in Fig. 10. Note that we intend to run the circuits at a supply voltage which does not account for guardband so target throughputs 0.9 and 0.8 are more realistic than 1.0.

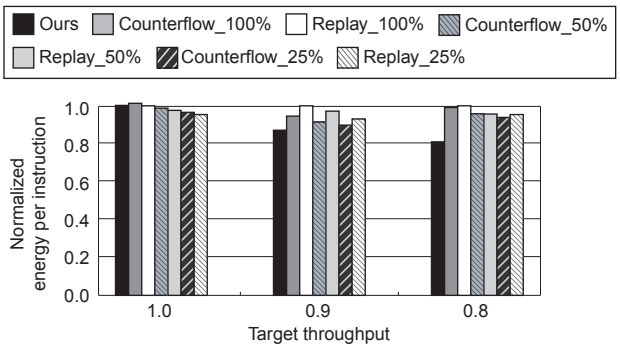

Fig. 10. Comparison of normalized energy consumption per instruction of our scheme, counterflow pipelining, and instruction replay. The ratios of Razor latches in counterflow and replay schemes are set to 1.0, 0.5, and 0.25.

V. Conclusion

It is important to reduce the timing penalty in the error correction operations to make the practical error detection/correction scheme for the pipeline. [1] made a breakthrough by proposing 1-cycle error correction but the scheme can be applied to two phase transparent latch design only. In this work, we present a new 1-cycle error correction scheme which can be applied to more popular clocking elements such as flip-flop or pulsed latch. The circuit simulation results with a 5-state pipeline and a 10-stage pipeline show that the proposed scheme consumes 14%-24% lower power than the previous error correction schemes such as instruction replay and counterflow pipelining for the same throughput.

References

[1] M. Fojtik *et al.*, "Bubble Razor: An architecture-independent approach to timing-error detection and correction," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 488–490.

[2] A. Muhtaroglu, G. Talyor, and T. Rahal-Arabi, "On-die droop detector for analog sensing of power supply noise," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 4, pp. 651–660, Apr. 2004.

[3] J. Tschanz *et al.*, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2007, pp. 292–293.

[4] A. Drake *et al.*, "A distributed critical-path timing monitor for a 65 nm high-performance microprocessor," in *Proc. Int. Solid-State Circuits Conf.*, Feb. 2007, pp. 398–399.

[5] J. Tschanz *et al.*, "Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance," in *Proc. Symp. on VLSI Circuits*, June 2009, pp. 112–113.

[6] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. Symp. on Microarchitecture*, Dec. 2003, pp. 7–18.

[7] D. Ernst *et al.*, "Razor: Circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, Nov. 2004.

[8] S. Das *et al.*, "Razor II: In situ error detection and correction for PVT and SER tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[9] K. A. Bowman *et al.*, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 49–63, Jan. 2009.

[10] K. A. Bowman *et al.*, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 194–208, Jan. 2011.

[11] *Design Compiler User Guide*, Synopsys, Sept. 2011.

[12] Nangate, Available: http://www.nangate.com/.

[13] *NanoSim User Guide*, Synopsys, Sept. 2011.