

TIMBER: Time borrowing and error relaying for online timing error resilience

Mihir Choudhury[†] Vikas Chandra[‡] Kartik Mohanram[†] Robert Aitken[‡]

[†]Department of Electrical and Computer Engineering, Rice University, Houston

[‡]ARM R&D, San Jose

Email: [†]{mihir, kmram}@rice.edu [‡]{vikas.chandra, rob.aitken}@arm.com

Abstract

Increasing dynamic variability with technology scaling has made it essential to incorporate large design-time timing margins to ensure yield and reliable operation. Online techniques for timing error resilience help recover timing margins, improving performance and/or power consumption. This paper presents TIMBER, a technique for online timing error resilience that masks timing errors by borrowing time from successive pipeline stages. TIMBER-based error masking can recover timing margins without instruction replay or roll-back support. Two sequential circuit elements — TIMBER flip-flop and TIMBER latch — that implement error masking based on time-borrowing are described. Both circuit elements are validated using corner-case circuit simulations, and the overhead and trade-offs of TIMBER-based error masking are evaluated on an industrial processor.

1. Introduction

As static and dynamic variability effects increase with technology scaling, the timing margins that are added to compensate for worst-case static and dynamic variability effects are also increasing [1]. Design-time techniques to address static variability based on clock skew adjustment [2], soft-edge flip-flops [3], and latches [4] have been proposed in literature. Post-manufacturing techniques like speed-binning can also reduce the timing margins necessary to offset static variability sources like process variations by assigning a different voltage/frequency to each chip during manufacturing test. This is possible because static variability sources do not change with time and are also workload independent. However, dynamic variability is both time and workload dependent. As a result, there is significant interest in solutions that provide runtime resilience to timing errors, thereby recovering the timing margins necessary to offset dynamic variability effects.

Online techniques for timing error resilience provide necessary hardware support for robustness to timing errors. Techniques for online timing error resilience proposed in literature can be broadly classified into three categories: error detection, error prediction, and error masking. Error detection techniques (e.g., [5–9]) are based on monitoring data-path signals for transitions arriving *after* the clock edge. Although the dynamic variability timing margin is recovered completely, the error is detected after the state of the system has been corrupted. Hence, error correction overhead is incurred either by a roll-back or a local instruction replay. Error prediction techniques (e.g., [10–12]) are based on monitoring the outputs for a specified time period *before* the clock edge in order to predict a potential timing error. Although the state of the system is always correct, the dynamic variability timing margin cannot be recovered because it is necessary to add an error prediction guard-

band before the clock edge. Error masking techniques (e.g., [13]) logically mask timing errors by adding extra logic. Although the dynamic variability timing margin is recovered completely and the system state is always correct, error masking incurs a modest combinational area and power overhead. Further details and comparison of existing techniques for online timing error resilience are provided in Sec. 2.

This paper proposes TIMBER, a technique for online timing error resilience that masks timing errors by borrowing time from successive pipeline stages, without requiring hardware support for roll-back or instruction replay. TIMBER was motivated by an analysis of the distribution of the critical paths in an industrial processor, where it was found that only a small fraction of flip-flops serve as *both* start and end-points of critical paths. Thus, timing errors arising from local dynamic variations and fast changing global dynamic variations that frequently span only a single pipeline stage can be masked by borrowing time from the successive pipeline stage. Further, TIMBER can also handle timing errors due to slow changing global variations that may result in multi-stage timing errors. A multi-stage timing error occurs when two or more critical paths are affected by dynamic variability on successive clock cycles across multiple stages. In such cases, TIMBER enables the system to run error-free for multiple cycles during which it initiates a temporary reduction of the clock frequency at the system level to mitigate the occurrence of timing errors. Since the probability of a multi-stage timing error is very small, the loss in performance due to a temporary reduction in clock frequency is negligible.

Two sequential elements are proposed to implement the TIMBER architecture: TIMBER flip-flop and TIMBER latch. TIMBER flip-flop implements time-borrowing in discrete units, thus preserving the edge-sampling property of a conventional master-slave flip-flop. On the first pipeline stage with a timing error, TIMBER flip-flop masks the error by borrowing one time unit. An error relay logic alerts the next stage to borrow an additional time unit, if a timing error propagates to that stage. On the other hand, TIMBER latch implements continuous time-borrowing, i.e., TIMBER latch is transparent for the entire checking period (equal to multiple discrete time units), and hence any late arriving transition in the checking period is masked by borrowing time. Thus, TIMBER latch does not require error relay logic. Although the edge-sampling property of TIMBER flip-flop is lost and TIMBER latch propagates glitches and spurious transitions in the checking period, our implementation guarantees that TIMBER latch does not signal a false timing error.

This paper is organized as follows. Section 2 describes existing literature on online techniques for timing error resilience. Section 3 motivates TIMBER and Section 4 describes the time-borrowing mechanisms in TIMBER. Section 5 describes the TIMBER circuits. Section 6 presents a case-study for TIMBER on an industrial processor. Section 7 is a conclusion.

Table 1: Comparison of various techniques for online timing error resilience.

Feature	Error detection	Error prediction	Error masking	
			Logical	Temporal
Error detection mechanism	Duplicate latch/FFs [14, 15] Transition detectors [15]	Duplicate latch/FFs [11] Sensors [10] Duplicate paths [12]	Redundant logic [13]	Duplicate latch/FFs [16] Edge detectors [17] TIMBER
When? (Relative to clock edge)	After	Before	–	After
Error recovery mechanism	Rollback or instruction replay	No error	No error	No error
Clock-tree loading	Yes	Yes	No	Yes
Short-path padding	Yes	Yes	No	Yes
Sequential overhead	Large	Large	None	Large
Combinational overhead	Small	None	Moderate	Small
Timing margin recovery	Full	Partial	Full	Full
Variability source targeted	All dynamic	Gradual dynamic	All dynamic	All dynamic
Techniques	RAZOR [14] TDTB and DSTB [15]	Canary FFs [11] Sensors [10] TRC [12]	Approximate circuits [13]	PEDFF [16] DCFE [17] TIMBER

2. Prior work

Several techniques have been proposed in literature for online timing error resilience. Broadly, these techniques can be classified into three categories based on error detection, error prediction, and error masking as summarized and compared in Table 1.

Error detection: Error detection techniques are based on monitoring data-path signals for transitions arriving *after* the clock edge. In [5], one of the earliest circuits for online timing error detection using an online stability checker that monitored late transitions arriving in a stability checking period after the clock edge was described. In [6, 7], a sensing circuit for delay faults for self-checking applications was described. In [8], error detection based on re-sampling data-path signals after a delay, and then comparing the resampled value to the value stored in the data-path flip-flop was proposed. RAZOR [14] proposed the application of this online timing error detection scheme to reduce power or increase performance using runtime voltage/frequency tuning. A variant of RAZOR that replaces the data-path flip-flop by a latch to avoid metastability issues was proposed in [15]. However, the duty cycle of the clock has to be adjusted to avoid severe hold-time constraints introduced by the latch. In [9], an error detection circuit based on a sense amplifier that can detect both timing errors and soft errors was described. Logic-based techniques for concurrent delay testing (e.g., [18]) based on circuit duplication have also been proposed.

Error prediction: Error prediction techniques are based on monitoring data-path signals for transitions for a specified time period *before* the clock edge. In [10], a stability checker design that predicts timing errors due to a gradual increase in delay due to wearout and aging effects was described. Another error prediction technique that pads the data-path with a delay element and samples the delayed data-path signal in another flip-flop, called the canary flip-flop, was described in [11]. A timing error is predicted when the value in the data-path flip-flop differs from the value in the canary flip-flop. Error prediction based on duplicating critical paths and using timing errors on the duplicated paths to predict a timing error on the original paths was described in [12]. This approach is limited in its effectiveness since (i) the duplicated and critical paths in the design may experience different workloads and variability and (ii) the critical paths may change over time [19].

Error masking: Error masking techniques proposed in literature can be classified into two categories: logical and temporal. Logical error masking techniques (e.g., [13]) use redundant logic to compute the correct value of the output with a smaller delay when critical paths are exercised. Temporal error masking techniques

mask errors by time-borrowing, i.e., delaying the arrival time of the correct data to the next pipeline stage. In [16], a temporal error masking technique based on stalling the clock for one cycle after detecting a timing error to correct the state of the system was proposed. This technique assumes that the latency for consolidating errors from various flip-flops in the design is less than a clock cycle to stall the clock before the state is corrupted. However, in practice, this may be difficult to achieve in high performance designs due to (i) a small cycle time and (ii) long latency involved in consolidating error signals from a large number of flip-flops susceptible to timing errors. In [17], an edge detector detects timing violations near the clock edge, and a delayed clock is subsequently used to re-sample and correct the data-path value by borrowing time from the next pipeline stage. This technique assumes that the time borrowed from the next pipeline stage is absorbed by a non-critical path being sensitized in the next stage. This may not be a valid assumption and may lead to timing errors, especially in high performance designs. Further, the edge detector circuit depends on accurate delay values and margining may be needed in the presence of process variations. Finally, a mathematical formulation for time-borrowing in a linear pipeline using a soft-edge flip-flop was described in [20].

3. TIMBER: Motivation

In this section, we study the distribution of critical paths between flip-flops in an industrial processor to motivate the potential for error masking based on time-borrowing. We argue that timing errors caused by dynamic variations frequently span only one pipeline stage on successive clock cycles, and thus can be masked by time-borrowing. For a critical path p , we define a single-stage timing error as the event that dynamic variability causes the delay of p to exceed the clock period. Consider multiple critical paths, p_1, p_2, \dots, p_k such that the terminal flip-flop of p_{i-1} is the starting flip-flop of p_i , $1 < i \leq k$. For $k > 1$, we define a k -stage (i.e., multi-stage) timing error as the event that over k successive clock cycles, dynamic variability causes a $(k-1)$ -stage timing error on paths p_1, \dots, p_{k-1} and the sum of the delays on p_1, \dots, p_k exceeds k times the clock period.

Fig. 1(a) summarizes the critical path distribution between flip-flops in an industrial processor. Three performance points — low, medium, and high — were considered. There are four bars for each performance point corresponding to the percentage of flip-flops that have a path in the top 10%, 20%, 30%, and 40% critical paths terminating at them. The shaded portion of each bar indicates the percentage of flip-flops that have critical paths starting and terminating at them. Consider the top 20% paths in the medium performance processor. Although nearly 50% of the flip-flops have critical paths

terminating at them, 70% of these flip-flops do not have any top 20% critical path originating from them in the next pipeline stage. Hence, 70% of the flip-flops have at least 20% timing slack on all paths in the successive pipeline stage, and are only susceptible to single-stage timing errors.

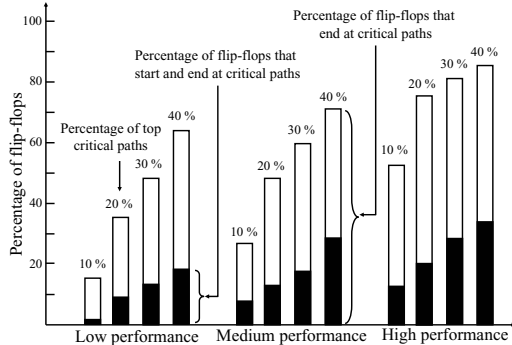


Figure 1: Critical path distribution between flip-flops.

The remaining 30% flip-flops have critical paths starting and terminating at them, and thus are susceptible to multi-stage timing errors. For a multi-stage timing error to occur, multiple critical paths connected end-to-end will have to be sensitized on successive clock cycles. The critical path sensitization probability for the top 10% critical paths is of the order of 10^{-4} – 10^{-8} [13]. Hence, the probability of a multi-stage timing error resulting from sensitization of multiple critical paths on successive clock cycles is negligibly small. To summarize, the single-stage timing error rate due to dynamic variability effects is much higher than the multi-stage timing error rate in modern processors. In TIMBER, multi-stage timing errors are masked by time-borrowing up to two or three stages. Multi-stage timing errors spanning more stages are avoided by reducing the clock frequency at the system level. Based on the latency incurred for error consolidation, the clock frequency can either be reduced immediately after the first timing error occurs or it can be deferred until the first multi-stage timing error occurs. Both approaches have their merits and the trade-offs are discussed in detail in the next section.

4. TIMBER: Overview

The core principle of TIMBER is to detect a timing error after the clock edge and to mask the timing error by borrowing time from the next pipeline stage. By masking timing errors, TIMBER can recover timing margins required to offset dynamic variability effects without roll-back or instruction replay. The largest timing violation that can occur due to dynamic variability effects in a single pipeline stage is equal to the timing margin that we seek to recover using TIMBER, henceforth termed the recovered timing margin. The period of time after the clock edge reserved for error detection and masking is referred to as the checking period. The duration of the checking period and the recovered timing margin are fixed during design, and they are related as follows. When the first timing error, i.e., a single-stage timing error occurs, the late arriving data signal can cause a worst-case timing violation equal to the recovered timing margin. TIMBER can mask this single-stage timing error by borrowing a time interval of duration equal to the recovered timing margin. If the next pipeline stage is also affected by dynamic variability, then the late arriving data signal can cause a worst-case timing violation equal to twice the recovered timing margin. TIMBER can mask a two-stage timing error by borrowing a time interval of duration equal to twice the recovered timing

margin. In general, for a given checking period, c , and a recovered timing margin, t , TIMBER can mask up to k -stage timing errors such that $c = k \times t$. Thus, the checking period can be divided into k intervals, each of duration t . Note that the checking period determines the hold-time constraints for the design, i.e., the short paths in the design must be padded to have a delay greater than the sum of hold time and the checking period.

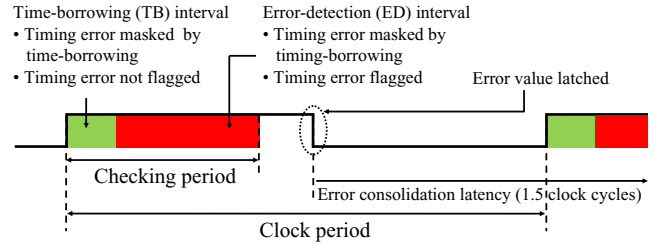


Figure 2: TIMBER-based error detection and error masking.

The time intervals in the checking period are classified into two types: time-borrowing (TB) and error-detection (ED), such that the first $k_{TB} \geq 0$ intervals are of type TB and the rest of the $k_{ED} = k - k_{TB}$ intervals are of type ED. TIMBER is designed to mask up to k -stage timing errors, and to avoid a $(k+1)$ -stage timing error by reducing the clock frequency at the system level. Timing errors up to k_{TB} stages can be masked by borrowing only the TB time intervals in the checking period. These timing errors are not flagged to the central error control unit, i.e., flagging of timing errors to the central error control unit are deferred to the first occurrence of a $(k_{TB} + 1)$ -stage timing error. The advantage of deferring error flagging is that timing errors can be masked without reducing the clock frequency. However, the TB intervals required for deferring error flagging result in lesser recovered timing margin since the checking period has to be divided into more intervals of shorter duration. Hence, the recovered timing margin can be increased for the same checking period by eliminating the TB interval, i.e., by flagging single-stage timing errors to the central error control unit.

On the other hand, masking a $(k_{TB} + 1)$ -stage timing error requires borrowing an ED interval (in addition to all k_{TB} time intervals) in the checking period. A $(k_{TB} + 1)$ -stage timing error is the first timing error that is flagged to the central error control unit. The remaining $(k_{ED} - 1)$ ED intervals ensure that all timing errors are masked for $(k_{ED} - 1)$ clock cycles after the first timing error is flagged to the central error control unit. The error signal is latched on the falling edge of the clock cycle on which the first ED time interval is borrowed, i.e., when the first $(k_{TB} + 1)$ -stage timing error occurs. This provides an extra half clock cycle for error consolidation. The error signals from all TIMBER sequential elements is consolidated using an OR-tree at central error control unit. After an error consolidation latency, attributed mainly to the latency of the OR-tree, the central error control unit reduces the clock frequency temporarily at the system level to mitigate the timing error rate.

Fig. 2 shows a checking period that is divided into three intervals, with one TB interval and two ED intervals. Based on this, all single-stage timing errors are masked but not flagged to the central error control unit. Two-stage timing errors are masked by borrowing a TB and an ED time interval and the timing error is also flagged to the central error control unit. Since there are two ED intervals, the second ED interval ensures that all timing errors are masked for $(k_{ED} - 1)$ clock cycles after the first timing error is flagged to the central error control unit. Hence, the error consolidation latency must be less than 1.5 clock cycles (half a clock cycle is added because the error signal is latched on the falling clock edge).

5. TIMBER: Circuit design

This section describes two sequential circuit elements — TIMBER flip-flop and TIMBER latch — that implement the TIMBER architecture for a checking period with one TB and two ED intervals. TIMBER flip-flop preserves the edge-sampling property of a master-slave flip-flop because error masking is performed by borrowing discrete time intervals. As a result, the TIMBER flip-flop-based design requires error relay logic to determine the number of time intervals required to mask errors in successive pipeline stages. TIMBER latch eliminates the need for error relay logic by implementing time-borrowing using a level-sampling latch. However, TIMBER latch propagates glitches and spurious transitions during the checking period. An important feature of both TIMBER flip-flop and TIMBER latch is that there is an enable signal that allows the time-borrowing mechanism to be disabled for operation as a normal master-slave flip-flop.

5.1 TIMBER flip-flop

A TIMBER flip-flop consists of two master latches, M0 and M1, and a common slave latch as shown in Fig. 3(a). The clock control logic for the TIMBER flip-flop is shown in Fig. 3(b). The signal EN denotes the system reset signal and the signal EN is the enable signal. Time-borrowing in a TIMBER flip-flop can be turned off by setting EN to zero. When EN is low, P0 is \overline{CK} , and P1 is high. Thus, M0 and the slave latch together function as a conventional master-slave flip-flop and M1 is blocked because the transmission gate P1 is open. In a conventional master-slave flip-flop, M0 samples the value of the data signal D at the rising edge of the CK and drives the slave latch and the output Q to the sampled value when CK is high. When CK goes low, the transmission gate P0 is open and the slave latch drives the output Q.

When EN is high, the TIMBER flip-flop operates in the time-borrowing mode. The three intervals in the checking period are encoded using the select input signals, S_1S_0 . $S_1S_0 = 00$ is the TB interval and $S_1S_0 = 01, 10$ are the ED intervals. On system reset, S_1S_0 is set to 00. Error masking based on time-borrowing happens as follows. The master latch M0 samples the value of the data signal, D, on the rising edge of clock and drives the slave latch and the output, Q, to the sampled value. The master latch M1 samples the data signal, D, on the rising edge of the delayed clock, DCK, after a delay δ determined by the value of the select inputs S_1S_0 . On the rising edge of the delayed clock, DCK, the transmission gate P0 opens and the transmission gate P1 closes. Thus, after delay δ , for the rest of the clock period when CK is high, the master latch M1 drives the slave latch and the output Q to the new value sampled by M1. If no timing error has occurred, the master latches M0 and M1 would sample the same value. Hence, M0 drives the slave latch and the output to the correct value on the rising edge of CK, and no time-borrowing occurs.

If a timing error occurs at the flip-flop, the master latches M0 and M1 sample different values, and M1 masks the timing error after delay δ as follows. Recall that error masking in a TIMBER flip-flop occurs by borrowing discrete time units. Suppose each interval in the checking period has a duration of 100ps, and S_1S_0 is 00. If a timing error occurs due to a 80ps timing violation on the data input, then the error is masked by the master latch M1 after a 100ps delay, i.e., 100ps is borrowed from the next stage. Note that TIMBER flip-flop does not suffer from data-path metastability issues because a data-path signal violating setup time on the rising edge of clock is masked by the delayed sampling of the data-path signal by master latch M1. To mask multi-stage timing errors, error relay logic configures the select inputs of TIMBER flip-flops in successive pipeline stages as follows.

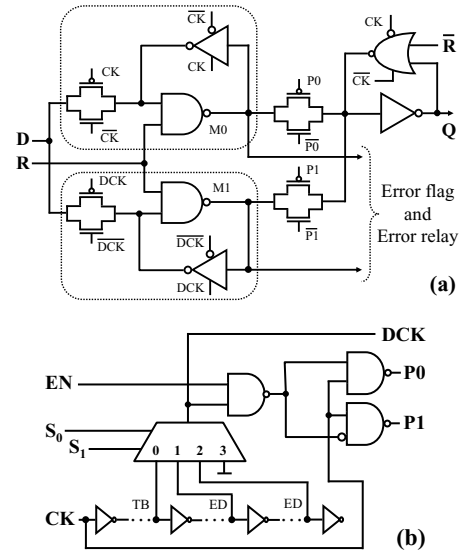


Figure 3: TIMBER flip-flop (a) design and (b) clock control.

Error relay: Consider a TIMBER flip-flop, f , with m TIMBER flip-flops g_1, g_2, \dots, g_m in the fanin cone of f . Denote $S(g_i)$ as the select input to g_i . If no error occurs at g_i , then the select output of g_i is set to 00. If an error occurs at g_i , then the select input $S(g_i)$ is incremented by 1 to obtain the select output for g_i . Incrementing $S(g_i)$ by 1 ensures that the TIMBER flip-flop f can borrow an additional time interval if a multi-stage timing error occurs at f . The select input for f is obtained as the maximum over all the select outputs from g_1, g_2, \dots, g_m . The logic for generating the select outputs at each TIMBER flip-flop using its select inputs is omitted from Fig. 3(a) due to space constraints. Fig. 4 is the block diagram for the error relay logic. Note that the error relay logic is different from the error consolidation logic to the central error control unit. Recall that the error signal is latched on the falling edge of the clock. Since the error relay logic must set the select inputs before the next rising clock edge, the error relay logic can have a maximum delay of half of the clock period. In Sec. 6, a case-study for an industrial processor shows that the delay of the error relay logic is much smaller than half a clock period. This is because the error relay for a TIMBER flip-flop must occur only from a small number of TIMBER flip-flops in its fanin cone that are both start and endpoints of critical paths (refer Fig. 1).

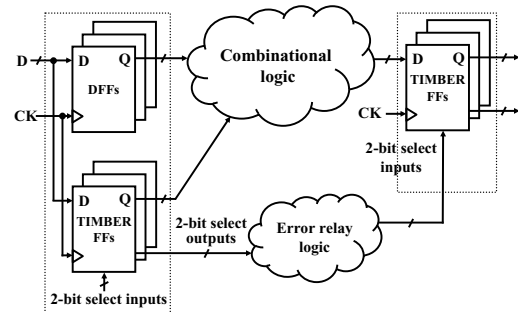


Figure 4: TIMBER flip-flop error relay logic.

Fig. 5 shows SPICE waveforms for error masking when a two-stage timing error occurs on two TIMBER flip-flops, f_1 and f_2 , on successive pipeline stages. The signals D_1 (D_2), Q_1 (Q_2), and Err_1 (Err_2) are the data, output, and error signals for flip-flop f_1 (f_2). The first timing error, occurring at flip-flop f_1 , is masked by

borrowing one TB time interval at f_1 . Although the timing error is not flagged to the central error control unit (Err_1 signal is 0), the error relay logic configures the select inputs of flip-flop f_2 to 01. Thus, when a two-stage timing error occurs at flip-flop f_2 , the error is masked by borrowing a TB and an ED time interval at f_2 . The timing error at f_2 is flagged to the central error control unit by latching the error signal (Err_2 signal goes high) on the subsequent falling edge of CK.

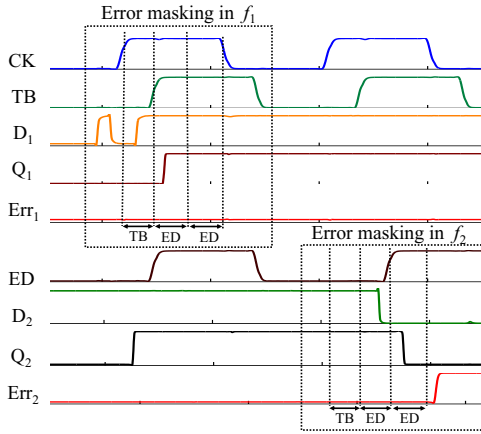


Figure 5: Two-stage timing error in a TIMBER flip-flop design.

5.2 TIMBER latch

TIMBER latch implements time-borrowing in continuous units using a level-based sampling of the data using a pulse-gated latch. A TIMBER latch consists of a master and a slave latch as shown in the circuit schematic in Fig. 6(a). The clock control logic for a TIMBER latch is shown in Fig. 6(b). The signal R denotes the system reset signal and the signal EN is the enable signal. Time-borrowing in a TIMBER latch can be turned off by setting EN to zero. When EN is low, the transmission gate L is open and the TIMBER latch operates as a conventional master-slave flip-flop.

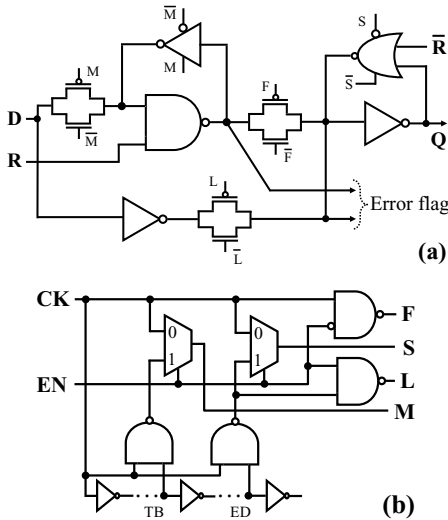


Figure 6: TIMBER latch (a) design and (b) clock control.

When EN is high, the TIMBER latch operates in the time-borrowing mode. In this mode, the transmission gate F is open and the master latch and slave latch operate independently as pulse-gated latches. The checking period is divided into one TB and one ED

interval. Note that this ED interval is equivalent to the sum of the ED intervals in the TIMBER flip-flop. The master latch is transparent during the TB interval and the slave latch is transparent for the entire checking period. A timing error is detected by comparing the values stored in the master latch and the slave latch on the falling edge of the clock. When a single-stage timing error occurs, the timing violation of the late arriving data signal lies within the TB time interval. The timing error is masked because the slave latch is transparent for the entire checking period. Since the master is also transparent for the TB interval, both the master latch and slave latch hold the same value and hence, a timing error is not flagged. However, if a two-stage timing error occurs such that the timing violation of the late arriving data signal is greater than the TB interval, then the master and slave latches sample different values, and a timing error is detected and flagged to the central error control unit. Recall that a TIMBER latch masks timing errors by borrowing continuous time units. Suppose the TB interval is 100ps and a timing violation of 80ps occurs at a TIMBER latch, then the error is masked by borrowing 80ps from the next stage. Since the slave latch is transparent for the entire checking period, error relay logic is not required. However, TIMBER latch propagates glitches and spurious transitions during the checking period. Note that TIMBER latch does not have metastability issues because level-sensitive sampling is used for time-borrowing.

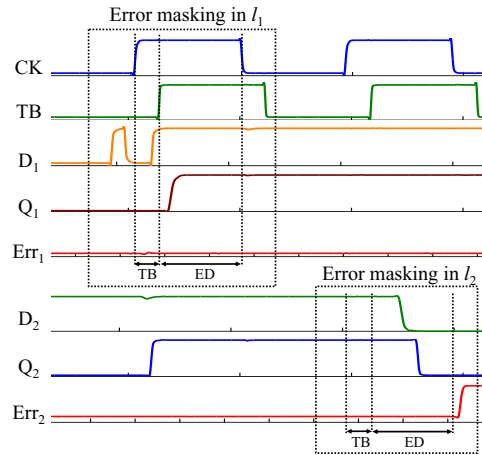


Figure 7: Two-stage timing error in a TIMBER latch design.

Fig. 7 shows SPICE waveforms for error masking when a two-stage timing error occurs on two TIMBER latches, l_1 and l_2 , on successive pipeline stages. The signals D_1 (D_2), Q_1 (Q_2), and Err_1 (Err_2) are the data, output, and error signals for latch l_1 (l_2). The first timing error, occurring at latch l_1 , can be masked by borrowing the time unit TB. Hence, the timing error is not flagged (Err_1 signal is 0). When a two-stage timing error occurs at latch l_2 , the error is masked by borrowing a TB and an ED time interval. The timing error at latch l_2 is flagged by latching the error signal (Err_2 signal goes high) on the subsequent falling edge of clock CK.

6. TIMBER case study

We present results from a case-study when TIMBER is integrated into an industrial processor. Three processor performance points — low, medium, and high — each with four checking periods of 10%, 20%, 30%, and 40% of the clock period are considered. For a checking period equal to $c\%$ of the clock period, all flip-flops terminating at the top $c\%$ critical paths are replaced by a TIMBER sequential circuit element (TIMBER flip-flop or TIMBER latch).

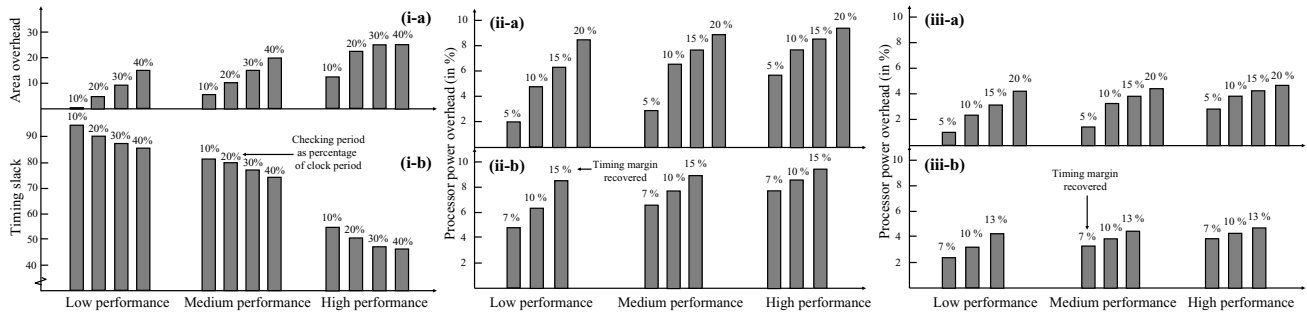


Figure 8: (i) TIMBER flip-flop error relay logic: (a) area overhead and (b) timing slack, (ii) Power overhead for TIMBER flip-flop: (a) without TB and (b) with TB interval, and (iii) Power overhead for TIMBER latch: (a) without TB and (b) with TB interval.

As described in Sec. 4, larger dynamic variability timing margins can be recovered for the same checking period by eliminating the TB interval, i.e., by flagging a single-stage timing error to the central error control unit. We present results for both cases in this paper. When the TB interval is not considered, the checking period is divided into two ED intervals. Hence, for a checking period equal to $c\%$ of the clock period, the timing margin recovered is equal to $c/2\%$ of the clock period. When the TB interval is considered, the checking period is divided into one TB and two ED intervals. Thus, the timing margin recovered is equal to $c/3\%$ of the clock period. Results for both cases (without and with the TB interval) for the processor based on TIMBER flip-flop and TIMBER latch is reported in Fig. 8(i)-(ii) and Fig. 8(iii), respectively.

TIMBER flip-flop: The overhead for a design based on TIMBER flip-flop includes (i) overhead of a TIMBER flip-flop over a conventional flip-flop and (ii) overhead of the error relay logic. Fig. 8(i-a) shows the area overhead for the error relay logic used in TIMBER flip-flop architecture for the four checking periods at each performance point. Recall from Sec. 5 that the error relay logic can have a maximum latency of half a clock cycle. Fig. 8(i-b) presents the timing slack as the percentage of half the clock period for the error relay logic in the TIMBER flip-flop architecture. A large timing slack is available because error relay has to be performed only from a small number of TIMBER flip-flops that are the start and end-points of critical paths (refer Fig. 1). The total power consumption of a TIMBER flip-flop is about two times that of a conventional master-slave flip-flop. The switching activity in the error relay logic is small because under normal operation, the inputs to the error relay logic are all zeros and change only when a timing error occurs. Hence, the error relay logic mainly contributes to the static power overhead. Fig. 8(ii-a) and (ii-b) present the total power overhead for TIMBER flip-flop architecture over the base design without and with the TB interval, respectively.

TIMBER latch: The overhead for a design based on TIMBER latch can be attributed to the overhead of a TIMBER latch over a conventional master-slave flip-flop. The total power consumption for a TIMBER latch is about 1.5 times that of a conventional master-slave flip-flop. Fig. 8(iii-a) and (iii-b) present the total power overhead for TIMBER latch architecture over the base design without and with the TB interval, respectively.

7. Conclusions

This paper described TIMBER, an online technique for timing error resilience to recover dynamic variability timing margins. TIMBER masks timing errors by borrowing time from successive pipeline stages. Two sequential circuit elements are described: TIMBER flip-flop and TIMBER latch. TIMBER flip-flop preserves the edge-sampling property of a master-slave flip-flop by borrowing

discrete time units from the successive pipeline stage using error relay logic. TIMBER latch implements time-borrowing using a level-sensitive latch to eliminate the error relay logic. Although it does not possess the edge-sampling property of a flip-flop, and is transparent to glitches and spurious transitions, it does not flag false errors. Results of a case study of TIMBER in a modern processor at several performance points indicate that TIMBER can recover significant dynamic variability margin for very low overhead and negligible loss in performance.

References

- [1] A. Wang *et al.*, *Adaptive Techniques for Dynamic Processor Optimization*. Springer, 2008.
- [2] A. Tiwari *et al.*, "ReCycle: Pipeline adaptation to tolerate process variation," in *Proc. Intl. Symposium on Computer Architecture*, pp. 323–334, 2007.
- [3] M. Wiecekowsky *et al.*, "Timing yield enhancement through soft edge flip-flop based design," in *Proc. Custom Integrated Circuits Conference*, pp. 543–546, 2008.
- [4] L. X. Liang *et al.*, "Revival: A variation-tolerant architecture using voltage interpolation and variable latency," *Proc. Intl. Symposium on Microarchitecture*, vol. 29, pp. 127–138, 2009.
- [5] P. Franco *et al.*, "On-line delay testing of digital circuits," in *Proc. VLSI Test Symposium*, pp. 167–173, 1994.
- [6] M. Favalli *et al.*, "Sensing circuit for on-line detection of delay faults," *IEEE Trans. VLSI Systems*, vol. 4, pp. 130–133, 1996.
- [7] C. Metra *et al.*, "On-line detection of logic errors due to crosstalk, delay, and transient faults," in *Proc. Intl. Test Conference*, pp. 524–533, 1998.
- [8] M. Nicolaidis, "Time redundancy based soft error tolerance to rescue nanometer technologies," in *Proc. VLSI Test Symposium*, pp. 86–94, 1999.
- [9] Y. Tsiatouhas *et al.*, "A sense amplifier based circuit for concurrent detection of soft and timing errors in CMOS ICs," in *Proc. Intl. On-line Testing Symposium*, pp. 12–16, 2003.
- [10] M. Agarwal *et al.*, "Circuit failure prediction and its application to transistor aging," in *Proc. VLSI Test Symposium*, pp. 277–286, 2007.
- [11] T. Sato *et al.*, "A simple flip-flop circuit for typical-case designs for DFM," in *Proc. Intl. Symposium on Quality Electronic Design*, pp. 539–544, 2007.
- [12] K. Bowman *et al.*, "Circuit techniques for dynamic variation tolerance," in *Proc. Design Automation Conference*, pp. 4–7, 2009.
- [13] M. R. Choudhury *et al.*, "Masking timing errors on speed-paths in logic circuits," in *Proc. Design Automation and Test in Europe*, pp. 87–92, 2009.
- [14] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. Intl. Symposium on Microarchitecture*, pp. 7–18, 2003.
- [15] K. Bowman *et al.*, "Energy-efficient and metastability-immune timing-error detection and recovery circuits for dynamic variation tolerance," in *Proc. Intl. Conference on Integrated Circuit Design and Technology*, pp. 155–158, 2008.
- [16] M. Kurimoto *et al.*, "Phase-adjustable error detection flip-flops with 2-stage hold driven optimization and slack based grouping scheme for dynamic voltage scaling," in *Proc. Design Automation Conference*, pp. 884–889, 2008.
- [17] K. Hirose *et al.*, "Delay-compensation flip-flop with *in-situ* error monitoring for low-power and timing-error-tolerant circuit design," *Japanese Journal of Applied Physics*, vol. 47, pp. 2779–2787, 2008.
- [18] A. Paschalis *et al.*, "Concurrent delay testing in totally self-checking systems," in *Journal of Electronic Testing: Theory and Applications*, vol. 12, pp. 55–61, 1998.
- [19] B. Paul *et al.*, "Impact of NBTI on the temporal performance degradation of digital circuits," *IEEE Electron Device Letters*, vol. 26, pp. 560–562, 2005.
- [20] M. Ghasemazar *et al.*, "A mathematical solution to power optimal pipeline design by utilizing soft-edge flip-flops," in *Proc. Intl. Symposium on Low Power Electronics and Design*, pp. 33–38, 2008.