BTI-Aware Design Using Variable Latency Units

Saket Gupta and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455, USA.

***Abstract*—Circuit degradation due to bias temperature instability (BTI) can lead to timing failures in digital circuits. We develop variable latency unit (VLU) based BTI-aware designs, with a novel scheme for multioutput hold logic implementation for VLUs. A key observation is the identification and exploitation of specific supersetting patterns in the two-dimensional space of frequency and aging of the circuit. The multioutput hold logic scheme is used in conjunction with an adaptive body bias framework to achieve high performance, allowing the design to be easily incorporated in traditional synthesis flows. As compared to conventional combinational BTI-resilience scheme, our design achieves an area reduction of 9.2%, with a significant throughput enhancement of 30.0%.**

§1. INTRODUCTION

Bias Temperature Instability (BTI) [1], in the form of negative BTI (NBTI) in PMOS and positive BTI (PBTI) in NMOS transistors, is a significant concern in nanoscale circuits. BTI causes the transistor threshold voltage to shift over time, and the resulting increases in delay could cause a circuit to fail timing specifications as it ages.

Published approaches for enhancing BTI-resiliency include transistor sizing, logic resynthesis, or postsilicon tuning. These methods are built for conventional synchronous designs, where the worst-case delay determines the clock period. This work addresses the case where the clock period is based on the notion of average-case computations rather than the worst-case computations in a circuit, an approach that leads to improved data throughput.

Within the synchronous paradigm, two classes of techniques have been proposed for exploiting the average-case computations: variable-latency units [2]–[4], and error detection-correction units [5]. Our work focuses on the design of BTI-resilient circuits using variable latency units (VLUs). Unlike conventional combinational circuits that complete operations within one clock cycle, VLUs allow the computation of the combinational circuit to be completed in a variable, integer, number of clock cycles. By allowing high-probability operations to complete in a single cycle, but allowing rarer events to use multiple (typically two) cycles, the average cycle time may be shorter than that of the conventional implementation, implying that the circuit throughput for a VLU may be significantly larger.

As an illustration of a VLU, consider the 6-bit ripple carry adder (RCA) shown in Figure 1, with six full adders. Assuming unit gate delays, the conventional single-cycle fixed-latency combinational circuit has a cycle time, $T_{clk} = 13$ units, equal to the delay of its longest path, corresponding to a throughput, $\eta_1 = 1/13$. The VLU implementation of this adder operates at a reduced cycle time, $T_{clk} < 13$. For $T_{clk} = 9$, assuming that all primary input signals are mutually independent and have signal probabilities of 50%, 18.75% of the input patterns violate T_{clk}, and the VLU allows these to complete execution in two cycles. Under the 50% assumption above, each pattern is equiprobable, so that the average VLU delay is $0.8125 \times 9 + 0.1875 \times 18 = 10.69$ units, and the corresponding throughput $\eta_2 = 1/10.69$ is 21.6% better.

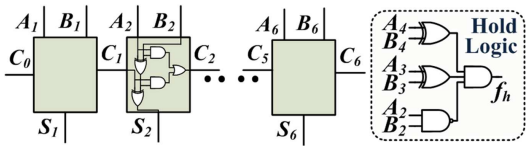


Fig. 1: A VLU implementation of a 6-bit ripple carry adder.

VLUs require dedicated combinational circuitry for identifying the input patterns that require two cycles for completion, to prompt each output flip-flop to hold its current value at the next clock transition (rather than clocking in a new value). This is referred to as "hold logic" and its output is called the "hold signal." Techniques for constructing the hold logic have been proposed in [2], [4]. The hold logic for the RCA here is small and is shown in Figure 1.

Therefore, exploiting the average-case computation can result in higher performance (i.e., throughput) than the worst-case. However, the performance of a VLU can degrade or even become incorrect in the presence of BTI, potentially leading to circuit failure as the circuit degrades temporally. Few efforts have been in the direction of constructing BTI-resilient circuits using the variable latency paradigm. One such technique has been explored in [6] for specific adders, but this work does not extend to general circuits.

Our contribution is to develop novel methods for building VLU-based BTI-resilient designs for a general circuit, such that the performance over its lifetime is maximized, and relies on two ideas: (a) using a novel scheme that uses *multioutput hold logic* (MOHL) to alter the appropriate hold logic over time, in conjunction with (b) using adaptive body biases [7] to maximize circuit performance.

The contents of the paper are organized as follows: §2 describes methods for building VLUs and overviews our BTI model. The concept of MOHL is then described in §3, followed by the details of our approach in §4. Next, §5 shows how our schemes can be used even with general BTI models, and §6 discusses circuit performance optimization using MOHL as well as body biasing. Finally, we experimentally validate our method in §7.

§2. PRELIMINARIES

A. Hold Logic Generation

Our VLU scheme assumes that an operation completes in either one or two clock cycles. Given a timing constraint T_{clk}, each path whose (delay + setup time) is larger than or equal to T_{clk} is termed as a critical path. If a set of input assignments I_P sensitizes a set of critical paths C_P, it must also evaluate the hold signal to 1. Paths excited by these input patterns are allowed two cycles for completion.

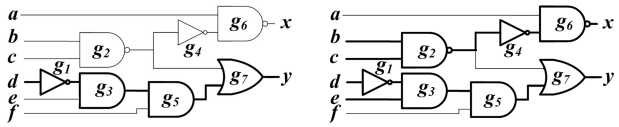


Fig. 2: An example of a circuit at various timing specifications so that it has (a) one critical path and (b) four critical paths.

We now briefly review an algorithm built on a corner based methodology for generating the hold logic; details are provided in [2], [4]. Consider the circuit in Figure 2(a). Assuming unit gate delays, the longest path has a delay of 4 units. With a required time of 3 units on x, y, the circuit has one critical path P, and the sensitization condition for this path constitutes the hold logic expression. Here, the critical path can be sensitized if every gate g on the path has noncontrolling values on its noncritical inputs. Let $S(g)$ represent this condition for gate g. The condition for sensitization for P, which activates the hold logic, is then given by $f_h = \prod_{\forall g \in P} S(g) = b \cdot c \cdot e \cdot f$.

For a more stringent required time specification of 2 units on x, y, the circuit has four critical paths, as shown in Figure 2(b). The sensitization conditions for multiple critical paths can be recursively computed using a method described in [2] to obtain the hold logic expression, $f_h = a + b \cdot c \cdot f$. Note that the algorithm *does not* work with paths, but involves a single topological traversal and processing of all the gates in the circuit, incurring *an O(n) complexity*, where n = number of gates in the circuit. Heuristic techniques to control the size of the hold logic have been presented in [2], [4].

Given the signal probabilities at the primary inputs (PIs) of the circuit, the average throughput η of the VLU is evaluated as the inverse of the average cycle time [4]:

$$\eta = \frac{1}{P_{hold} \cdot 2T_{clk} + (1 - P_{hold}) \cdot T_{clk}} = \frac{1}{(1 + P_{hold}) \cdot T_{clk}} \quad (1)$$

where P_{hold} is the hold logic activation probability.

B. VLUs at the Architectural-Level

Variable latency operation can be easily incorporated in a processor architecture. Figure 3 shows a typical five stage instruction pipeline in a microprocessor: variable latency designs may be employed at the EX stage. When the EX stage requires a two-cycle operation, it generates a stall for one cycle, preventing data from the IF and ID stage from moving forward to the EX stage. Such a stall can be implemented through a simple extension of a conventional *hazard detection unit* (HDU), which is a standard feature that stalls such pipelines in the presence of data hazard. A modified design shown in Figure 3, which now allows both the hold signal and the HDU output to activate and control the stalling mechanism in the pipeline, shows a low-overhead implementation that facilitates the use of the VLU.

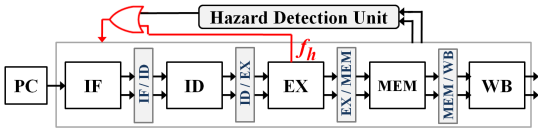


Fig. 3: Variable latency operation at the architectural level: the output of the HDU is appended to the hold signal to stall the pipeline for a two-cycle operation.

C. BTI Degradation Model and Delay Monotonicity

Given a BTI model, we use HSPICE to precharacterize the impact of changes in V_{th} on the delay D_g of a gate with n transistors as [8]:

$$D_g(t) = D_{g,0} + \sum_{i=1}^{n} \frac{\partial D_g}{dV_{th_i}} \Delta V_{th_i}(t) \quad (2)$$

where $D_{g,0}$ is the nominal delay, and the partial derivative represents the sensitivity of the delay to the threshold voltage, V_{th_i}, of device i. Such characterizations are reasonably inexpensive and have been widely deployed, e.g., in statistical static timing analysis (SSTA) methodologies.

This computation requires the determination of ΔV_{th} for each device in the gate. For both NBTI and PBTI, the effect of aging on the threshold voltage degradation for a stressed device increases as:

$$\Delta V_{th}(t) \propto t^{1/6} \quad (3)$$

Clearly, this is a monotonically increasing curve, and captures the effect of applying constant stress to a gate. Although BTI is known to show some recovery when the stress is removed, modeling this recovery requires precise knowledge of the input pattern distributions for each specific gate.

One way to achieve this is through signal probability information: however, there are two drawbacks to this. First, such information may not be available. Second, the available information can be quite inaccurate; it may predict the average behavior over all users and programs for a system, but the actual aging depends on the behavior of a specific user, which may not be predictable. Another potential approach is through the use of on-chip sensors, but these are of limited utility since they do not experience the same signal patterns as the circuits whose aging is to be measured. In our implementation, we employ the worst-case pattern for aging for BTI analysis, at the worst-case temperature corner, as a guaranteed pessimistic estimate of usage; this approach is consistent with widely-used methods for handling BTI.

For well-characterized circuits, specific information that is available about the signal probabilities at each node can easily be incorporated into our framework. Such circuits undergo stress/recovery cycles, and the delay model may work with the stress/recovery envelope [9], [10] of the V_{th} vs. t curve. By definition, such an envelope is always monotonically increasing.

One case that deserves special mention is when a functional unit is turned off for a long period of time, which is detectable by a sensor (e.g., software timers, sensors, or separate antifuses may keep track of the on- and off-times). In this case, due to recovery effects, the circuit undergoes "rejuvenation" as the threshold voltage recovers and the delay degradation is eased, and the use of an envelope waveform may be far too pessimistic. This case is addressed in §5.

§3. MULTIOUTPUT HOLD LOGIC: CONCEPT

We now introduce the concept of *multioutput hold logic* (MOHL). We begin with the idea that the task of building BTI-resilient VLUs is, in essence, one of partitioning the set of circuit paths into one-cycle and two-cycle paths[1]. Intuitively, for high throughput, it is important to keep the most frequently-excited paths in the one-cycle set (or more quantitatively, to keep the value of P_{hold} high). However, under BTI, path delays may change with time: under the monotone delay model, as path delays increase, more paths may move from the one-cycle set to the two-cycle set. We develop a systematic framework for choosing an appropriate partition of one-cycle/two-cycle paths with the option of changing this partition over time as delays degrade due to BTI.

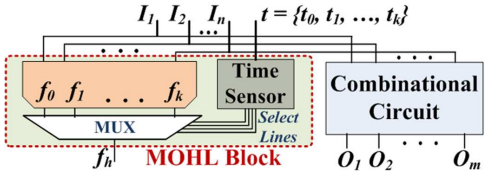


Fig. 4: The concept of MOHL VLU design. A time sensor selects the hold logic to be triggered at time t.

Our solution is based on the concept of an MOHL, introduced in Figure 4. Under this scheme, the VLU is controlled by temporally-varying hold logic circuitry. As the circuit ages, by the monotonicity argument presented in §2-C, an increasing number of single-cycle paths may require two cycles for completion, and the set of two-cycle input patterns changes. To account for this, the MOHL circuit has multiple hold signal outputs: depending on the age of the circuit, one is chosen. A hardware or software time sensor can capture the system operational time, and this information can be fed to a multiplexer that selects the proper hold signal to be triggered at time t.

§4. MULTIOUTPUT HOLD LOGIC: THEORY

We will now outline some useful properties of the MOHL design problem. In particular, we will describe some specific supersetting patterns in the two-dimensional space of frequency and circuit aging that can be used to build compact implementations of the hold logic. These patterns are based on the assumption of monotonic aging, described in §2-C; this assumption is removed in §5. We use standard synthesis tools for synthesizing MOHL; our contribution is

[1]As stated earlier, references to enumerated paths are only an aid to explanation; our actual implementations *do not* perform path enumeration.

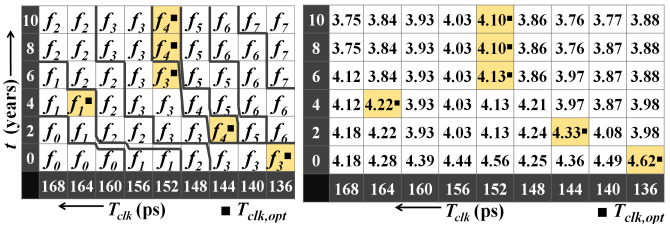
in identifying these subsetting patterns that reduce the implementation overhead in terms of the number of outputs, area, etc.

A. Tabulating the Effects of Aging on VLUs

As a circuit ages, the distribution of its path delays changes, and therefore, the hold logic that is required to operate the circuit at a specific value of T_{clk} changes. Conversely, at any time t, the hold logic required to ensure timing correctness is a function of T_{clk}. We capture these relationships in a *frequency/aging (F/A) grid*, a table whose columns are the possible values of T_{clk} and whose rows correspond to t, the age of the circuit. In the discussion to follow,

- $\mathcal{H}(T_{clk}, t)$ denotes the ON-set (or the $\mathcal{H}$-set) of the hold logic required to achieve a clock period of T_{clk} at a given time t.
- $\eta(T_{clk}, t)$ represents the corresponding value of η.

We use the terms "hold logic" and "$\mathcal{H}$-set" interchangeably.



(a) (b)

Fig. 5: The F/A grid for circuit apex7 showing (a) the hold logics and (b) the corresponding η values (shown on a 10^{-3} scale). The patterns in the grid correspond to supersetting structures and result as a consequence of the application of Theorems 1 and 2, and Corollaries 1 and 2.

The entry at each (T_{clk}, t) point in the grid represents the hold logic function (with T_{clk} as the one-cycle time), which is associated with a specific value of P_{hold} and η, the probability of hold logic activation and the throughput, respectively. In principle, the F/A grid is a discretization on the continuous space of (T_{clk}, t) values. A representative example of an F/A grid for benchmark apex7 is shown in Figure 5(a), and the corresponding η values are displayed in Figure 5(b).

The throughput for each hold logic function is computed using equation (1). We may perform a linear search on the values of T_{clk} in a row of the F/A grid to determine the point at which η is maximized. In each row in Figure 5(b), we highlight the maximum-throughput entry and denote the corresponding value of T_{clk} by $T_{clk,opt}(t)$. It is easily seen that $T_{clk,opt}(t)$ is not, in general, monotone with t.

B. Supersetting Trends

We now present some supersetting trends that help in minimizing the circuitry required to implement MOHL as a multioutput circuit.

Theorem 1 *At a given time t, for two clock period constraints applied to a VLU, $\mathcal{H}(T_{clk,2}, t) \supseteq \mathcal{H}(T_{clk,1}, t)$ and $P_{hold}(T_{clk,2}, t) \geq P_{hold}(T_{clk,1}, t)$ value if $T_{clk,1} > T_{clk,2}$.*

Proof: At any time t during the life of the chip, as we decrease the value of T_{clk}, the timing constraints on the circuit are tightened and more paths require two cycles for completion. Since gate delays increase monotonically with time, for any $T_{clk,2} < T_{clk,1}$, all input patterns that excite a path with a delay larger than $T_{clk,1}$ will clearly excite a path with delay larger than $T_{clk,2}$ (in addition to these, other patterns may also excite two-cycle paths). Since such input patterns constitute the ON-set of the hold logic of the circuit, the ON-set of hold logic at $T_{clk,1}$ is wholly contained within the ON-set of hold logic at $T_{clk,2}$, i.e., $\mathcal{H}(T_{clk,2}, t) \supseteq \mathcal{H}(T_{clk,1}, t)$. The corresponding result about P_{hold} follows trivially from this. □.

Example: In Figure 5, $f_0 \subset f_1 \subset f_2 \subset f_3$ in the row $t = 0$.

Theorem 2 *Under a monotonic delay model, at a given clock period T_{clk} applied to a VLU, for two time points $t_1 > t_2$, $\mathcal{H}(T_{clk}, t_1) \supseteq \mathcal{H}(T_{clk}, t_2)$ and $P_{hold}(T_{clk}, t_1) \geq P_{hold}(T_{clk}, t_2)$.*

Proof: By the monotonicity of the BTI degradation model described in §2-C, the path delays in a circuit slow down with time. Therefore, the same clock period T_{clk} constitutes a tighter requirement at time t_1 than at t_2. Therefore, more input patterns are included in the ON-set of the hold logic at t_1, i.e., $\mathcal{H}(T_{clk}, t_1) \supseteq \mathcal{H}(T_{clk}, t_2)$. □

Example: In Figure 5, by Theorem 2, $f_1 \subset f_3 \subset f_4$ in the column $T_{clk} = 152$.

These two theorems can be combined to define a corollary that describes a broader supersetting relationship between the hold logics.

Corollary 1: If $t_2 > t_1$ and $T_{clk,2} < T_{clk,1}$, then $\mathcal{H}(T_{clk,1}, t_1) \subseteq \mathcal{H}(T_{clk,2}, t_2)$ and $P_{hold}(T_{clk,1}, t_1) \leq P_{hold}(T_{clk,2}, t_2)$.

As a result of these supersetting trends, it is possible to detect patterns in the F/A grid. In Figure 5(a), the F/A grid for apex7 is divided into regions such that a single hold logic represents each region. Note that by Corollary 1, a region that is to the north or east of another region bears a subset relationship for the corresponding hold logic. The subsetting relationships for the $T_{clk,opt}$ entries in the table are:

$$\mathcal{H}(164, 4) = f_1 \subseteq \mathcal{H}(152, 6) = f_3 \subseteq \mathcal{H}(152, 8) = f_4$$
$$\mathcal{H}(136, 0) = \mathcal{H}(152, 6) = f_3$$
$$\mathcal{H}(144, 2) = \mathcal{H}(152, 8) = \mathcal{H}(152, 10) = f_4$$

In other words, all of the hold logic in this example follows a supersetting trend; however, this is not necessarily true in general. Even so, there can be substantial overlap in the minterms of the optimal hold logic at various times, a result that is formalized below.

Corollary 2: If $t_2 > t_1$ and $T_{clk,2} > T_{clk,1}$, then, $\exists\ \mathcal{H}(T_{clk,2}, t_1)$ that is a subset of both $\mathcal{H}(T_{clk,1}, t_1)$ and $\mathcal{H}(T_{clk,2}, t_2)$.

The "common ancestor," $\mathcal{H}(T_{clk,2}, t_1)$, allows sharing between the circuitry needed to implement $\mathcal{H}(T_{clk,1}, t_1)$ and $\mathcal{H}(T_{clk,2}, t_2)$, and implies that the two share a set of minterms. This indicates the likelihood that the area required to implement the hold logic as a multioutput circuit is less that the sum of separate implementations.

§5. REJUVENATION: NONMONOTONE BTI MODELS

Due to recovery effects, BTI aging may be nonmonotone. In practical workload scenarios, a gate may suffer both stresses and relaxations (removal of stress, causing delay recovery) and the monotonic delay degradation assumption breaks down. As discussed in Section §2-C, we focus on the more predictable case when a circuit is power-gated. During this period, the threshold voltage of all transistors recovers from BTI stress in a predictable way. This case is also practical because online sensing/detection techniques can easily be leveraged to determine the period of time when the circuit is in recovery mode.

We show that recovery has the effect of *rejuvenation*, effectively making a circuit "younger," and we may utilize the F/A table, except that an "effective age" of the circuit is used along the t axis.

Theorem 3 *Delay recovery in a circuit, when BTI stress is removed, can be captured by moving backward in time t along the F/A grid.*

Proof sketch: (Details omitted due to space limitations) The change in the delay is the product of the delay sensitivity to V_{th}, multiplied by ΔV_{th}. Since ΔV_{th} is reduced during recovery, the delay degradation is reduced, and aging is effectively (partially) reversed, corresponding to moving backward in time along the F/A grid.

It is easy to extend this analysis to multiple stress/recovery cycles.

§6. BTI-RESILIENT VLUs

At a fixed value of T_{clk}, as the VLU ages and more paths require two-cycle operation, the concept of MOHL, as outlined in §3, may be used to implement BTI-resilience. We classify the techniques for MOHL as being either *static*, when T_{clk} is constant through the lifetime of the circuit, or *dynamic*, when its value is tuned for maximal throughput during the circuit lifetime. Since the optimally-tuned T_{clk} may vary from one functional unit to another, dynamic MOHLs are

not realistic in mainstream systems, where T_{clk} is set through the use of system-level considerations, but the corresponding ideas will be used to obtain a better implementation of a static MOHL circuit.

A. Static MOHL VLU Implementation

This approach chooses a fixed T_{clk} through the life of the circuit. At $t = 0$, the hold logic corresponds to an initial set of two-cycle paths. As the circuit ages, under monotonicity, more paths require two cycles, and the MOHL is updated appropriately.

This scheme corresponds to moving along a single column of the F/A grid, corresponding to the $\{(T_{clk}, t), \forall t\}$, where T_{clk} is the specified period. By Theorem 2, the hold logic at each successive point in time is a superset of that before it. Therefore, there are substantial containment relationships between the $\mathcal{H}$-sets, which can be leveraged to build minimal multioutput functions.

The static MOHL VLU has two limitations: 1) the value of P_{hold} also increases monotonically with time; from equation (1), its throughput reduces monotonically as a function of time; 2) it cannot benefit from optimizations (§4-A) that adjust the clock period.

B. Adaptive MOHL VLU Implementation using Body Biases

To overcome these limitations of static MOHL VLUs, we consider a scenario where we build a dynamic VLU, for which it is permissible to change the clock period of the VLU as a function of time. Under this scheme, at each value of t, the MOHL VLU is operated at the optimal clock period, $T_{clk,opt}(t)$, that maximizes the throughput at time t. To enable this, the optimal hold logic is selected from different columns of the F/A grid (unlike the case for static VLUs); from Figure 5, $T_{clk,opt}(t)$ is not a constant or monotone with t.

A critical limitation of such a design is that varying T_{clk} over time can create synchronization problems in pipelines. Since T_{clk} is typically set by global considerations, and its optimal value may vary from one unit to another in a system, dynamic changes in T_{clk} may only be possible under restricted scenarios such as asynchronous systems and not under mainstream applications.

Therefore, we modify this scheme to build a new approach, based on the observation that the variations in $T_{clk,opt}(t)$ in the dynamic method above are typically very small over all values of t in the lifetime of the circuit. Examining equation (1), the primary gains in the throughput come about due to large discrete changes in P_{hold} (e.g., between $\mathcal{H}(152, 8)$ and $\mathcal{H}(164, 4)$ in Figure 5) and the contribution of T_{clk} variations is small. This allows the possibility of operating under a more practical paradigm, with a constant T_{clk} over all time. Figure 6(a) shows the working of this scheme for the F/A grid for the apex7 benchmark. Using $\mathcal{H}(T_{clk,opt}(t), t)$, ABB allows the circuit to be clocked at $T_{clk,opt}(0)$ at all t. Since $T_{clk,opt}(t)$ changes with t, the applied body bias $V_{bb}(t)$ also changes with t.

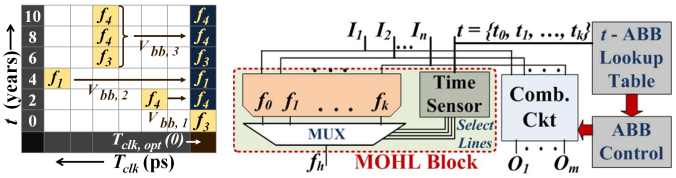
We consider two cases and use the concept of adaptive body bias ABB) [7] to preserve T_{clk}:

Case I: If $T_{clk,opt}(t) \leq T_{clk}$, we may simply operate the circuit at T_{clk} with hold logic corresponding to this T_{clk} (i.e., $\mathcal{H}(T_{clk}, t)$). This is functionally correct and represents an identical P_{hold} and a small shift in T_{clk} from $T_{clk,opt}$; therefore, the change in the throughput from the $t = 0$, as predicted by equation (1), is very small.

Case II: If $T_{clk,opt}(t) > T_{clk}$, then the optimal circuit clearly violates T_{clk}. In this case, we use the hold circuitry (and hence P_{hold}) from the $T_{clk,opt}(t)$ point and employ forward body bias (FBB) by applying a positive body bias voltage, $V_{bb}(t)$ to speed up the path delays and reduce the clock period to T_{clk}. Since the optimal hold logic does not change and $T_{clk,opt}(t)$ is close to T_{clk}, as predicted by equation (1), the throughput remains almost the same as that at $T_{clk,opt}$.

In Case II, we denote the set of one-cycle (two-cycle) paths at $T_{clk,opt}$ by $P_{1,opt}$ ($P_{2,opt}$), and the corresponding hold logic as $\mathcal{H}_{\text{opt}}$. The application of FBB speeds up all paths and the value of V_{bb} is chosen to guarantee that all paths in $P_{1,opt}$ meet the constraint, T_{clk}. It is theoretically possible that some paths in $P_{2,opt}$ may be sped up faster than those in $P_{1,opt}$ if they have vastly different sensitivities to V_{bb}, to the point that they become one-cycle paths. We refer to this set of paths as $P_{2\rightarrow1,opt}$: in practice it is unlikely that this set will have any members. Even if it were to, using the hold logic $\mathcal{H}_{\text{opt}}$ is functionally correct, but pessimistic (since it may allocate two cycles to the paths in $P_{2\rightarrow1,opt}$ instead of one).

The ABB scheme for an adaptive MOHL VLU operating at a fixed T_{clk} is illustrated through Figure 6(b). To the original MOHL scheme described in Figure 4, we add a time-based t-ABB lookup table, which saves the value of $V_{bb}(t)$ that must be applied to the circuit at time t to achieve the best throughput. On a practical level, our implementation applies the same body biases to all transistors in the functional block under consideration.



(a) (b)

Fig. 6: (a) Block description of the MOHL VLU design incorporating ABB, and (b) the ABB scheme corresponding to the F/A grid for apex7. Here, V_{bb} is the applied body bias to the circuit.

Over all benchmark circuits, we have found that $|V_{bb}(t)| \leq 0.25$V. In this range, we have verified that leakage power (including junction leakage) is negligible using SPICE simulations. This was also confirmed through a detailed leakage analysis presented in [8]. Therefore, we explore the range, in steps of 0.05V.

§7. EXPERIMENTAL RESULTS

In this section we present results on various ISCAS85, ISCAS89, MCNC, LGSYNTH93 and ITC99 benchmark circuits, synthesized using ABC [11] on the 45nm PTM [12] based library. Our .genlib library for ABC used for mapping circuits consists of INVs; BUFs; 2-4 input NANDs and NORs; 2 input XORs and XNORs; all with different sizes. We choose $t_{life} = 10$ years, and our experiments are based on the monotone BTI degradation model.

A. Evaluation Methodology

In order to evaluate the effectiveness of our MOHL VLU schemes as presented in §4 and §6, we compare the results of these designs, in terms of area overhead and throughput enhancements achieved, with other VLU-extensions of existing BTI-resilient designs for combinational circuits, such as the *delay padding* schemes as described below. A design may be made BTI-resilient by padding the timing specification using a margin to ensure that the circuit meets its timing requirements through its lifetime, t_{life}.

1) ***VLU Sizing-Based Padding*****:** A VLU can be synthesized using library delay models that predict the circuit delay at t_{life}. A padding strategy ensures that the circuit meets specifications throughout its lifetime by adding a safety margin to the timing specification. The circuits are sized to the knee of the area vs. delay curve.

2) ***Hybrid Padding*****:** As a circuit ages, an increasing number of paths violate the clock period. A conservative approach is to identify the paths that violate T_{clk} at the end-of-life: such paths are allowed two cycles throughout the entire lifetime of the circuit. This however results in significant throughput penalties, and some of these paths may well work within one cycle for part of the circuit lifetime.

We therefore combine the sizing and conservative approaches by introducing sizing-based partial padding into the combinational circuit. This approach ensures that it meets T_{clk} up to time $t = t_p$, where $t_p < t_{life}$. For $t \geq t_p$, we simply move the paths to a second cycle. The benefit achieved is that we incur lower sizing overhead as

compared to the sizing method, and also lower throughput degradation as compared to the conservative method.

The area overhead for this method arises from the extra hold logic required, and from sizing costs. The choice of t_p is important in ensuring low sizing overhead. Our implementation uses the point where the circuit suffers nearly 50% of the total degradation that it suffers over its entire lifetime. We can deduce, both through analytical algebraic techniques and through simulations, that for t_{life} =10 years, t_p = 2 years meets this criterion.

B. Area Overhead and Throughput Enhancements

1) ***Tabulation Details****:* The results of the area overhead and throughput enhancements as achieved by various schemes are summarized in Table I. Here, ΔA denotes the percentage area overhead (padding overhead and/or the hold logic area), and $\Delta\eta$ denotes the average change in throughput from $t = 0$ to t_{life}, incurred in each of the respective designs. Note that the circuit apex7 has a different mapping from that used in Figure 5, and therefore shows different numbers in our results table below.

TABLE I: Area Overhead and Throughput Comparisons of Various Designs for Overcoming BTI Degradation

Circuit	Sizing/ Padding		VLU/ Hybrid		Static MOHL VLU			Adaptive MOHL VLU				
	ΔA (%)	$\Delta\eta$ (%)	ΔA (%)	$\Delta\eta$ (%)	$\lvert\mathcal{H}\rvert$	ΔA (%)	$\Delta\eta$ (%)	$\lvert\mathcal{H}\rvert$	ΔA (%)	$\Delta\eta$ (%)	Max V_{bb}	CPU (min.)
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
Set1 Benchmarks												
c1908	0.4	36.6	-5.3	-3.8	4	-7.1	9.2	2	-13.1	37.5	0.25	8.83
c2670	7.8	18.1	2.2	-4.3	4	7.8	-34.1	1	-6.8	19.9	0.25	10.28
c5315	1.7	22.0	-5.9	-3.4	3	-2.1	16.5	3	-3.7	25.7	0.20	8.16
c7552	6.7	15.3	1.8	-9.3	4	-3.7	-6.0	2	-5.1	10.2	0.25	24.62
s344	-3.5	38.0	-9.0	-0.4	2	-10.3	10.8	2	-12.2	39.5	0.25	0.02
s635	0.8	61.6	-4.9	0.0	4	-1.2	30.0	1	-14.2	61.2	0.25	0.18
s1269	-0.6	14.3	-7.0	-7.1	2	-16.1	9.3	1	-16.1	13.7	0.05	0.21
s1512	-4.2	22.2	-9.9	-6.8	2	-17.2	1.2	1	-18.2	22.3	0.25	0.12
s3271	-2.6	13.5	-9.0	-1.5	2	-15.5	-0.6	3	-16.3	13.5	0.25	0.21
s3384	-1.3	61.0	-7.3	-1.6	4	-10.3	21.4	1	-15.8	59.5	0.25	0.87
cmb	4.6	34.7	-0.3	-0.2	4	7.9	-16.3	1	-6.1	31.8	0.20	0.03
lal	7.4	43.0	3.9	-0.1	4	15.2	-3.8	1	-0.4	37.4	0.25	0.05
ttt2	7.6	10.9	2.8	0.0	4	1.6	-9.1	1	-9.6	11.1	0.25	0.04
apex7	2.6	42.4	-4.6	-3.3	3	-10.2	35.3	3	-7.5	39.9	0.25	0.04
alu2	-1.1	40.2	-7.0	-0.3	3	-12.6	19.8	2	-14.6	36.5	0.15	0.08
i5	7.7	7.8	1.6	-5.8	3	-5.8	-13.9	2	-6.1	9.8	0.20	0.13
b11	9.3	26.1	2.3	-0.6	4	1.1	-22.7	3	-4.6	30.6	0.25	0.11
apex6	3.6	81.5	-1.9	0.0	2	-2.2	20.0	3	-0.8	75.0	0.20	0.11
alu4	-2.7	48.1	-8.5	-1.7	2	-15.7	12.4	2	-15.4	48.1	0.25	0.20
x3	0.5	85.1	-4.8	0.0	2	-7.8	12.7	3	-5.5	78.7	0.20	0.12
dalu	3.6	42.9	-0.6	0.0	3	5.8	7.2	3	-0.6	44.8	0.25	2.99
Avg.	2.3	36.4	-3.4	-2.4		-4.7	4.7		-9.1	35.6		2.73
Set2 Benchmarks												
s6669	-4.5	1.8	-10.4	-40.6	4.0	6.5	-43.6	3.0	-1.6	1.8	0.25	1.88
vda	-2.6	8.2	-6.7	-40.3	4.0	-2.9	-45.4	2.0	-11.9	0.6	0.25	0.04
des	-3.8	0.0	-8.7	-37.6	2.0	-15.6	-43.5	1.0	-15.6	0.0	0.25	0.06
t481	0.7	0.0	-3.7	-23.4	3.0	-2.6	-39.6	1.0	-14.2	0.0	0.25	0.10
Avg.	-2.6	2.5	-7.4	-35.4		-3.7	-43.1		-10.0	0.6		0.52
Overall	1.5	31.0	-4.0	-7.7		-4.5	-2.9		-9.2	30.0		2.38

Baseline case: The baseline corresponds to a conventional one-cycle combinational implementation, where worst-case sizing is used, where all paths are required to meet T_{clk} specification throughout the circuit lifetime, using library delay models that predict the circuit delay at t_{life}. VLUs with positive (negative) overhead values incur larger (smaller) overhead as compared to this design and imply that the corresponding VLU circuit is better (worse) than its single-cycle counterpart[2].

In presenting the data in the table, we refer to column number m as Cm, as marked in the table. For various benchmark circuits listed in C1, the delay degradation $\frac{\Delta D(t_{life})}{D(t=0)}$ over the entire lifetime lies between 5.6% and 15.6%, with an average of 11.4%. To be more realistic, our results choose the starting point $t = 0$ to be three months after the circuit is manufactured to model burn-in test procedures. (Note that starting at the real $t = 0$ would improve slightly the numbers shown in our results, but should leave the relative comparisons between various methods unchanged.)

In C2–C5, we present the results for the two methods described in §7-A: VLU sizing-based padding and hybrid padding. We then show the results for two MOHL VLU schemes (§6): the static (C6–C8) and the ABB-based (C9–C13) designs. Columns C6 and C9 show the number of outputs, $|\mathcal{H}|$ (number of different $\mathcal{H}$-sets), present in the MOHL circuit, and C12 shows the maximum magnitude of $V_{bb}(t)$ required at any time point in the life of the adaptive MOHL VLU.

Run times for the adaptive MOHL VLU method for various circuits are shown in C13, and are seen to be reasonable. The CPU times are presented only for the adaptive MOHL VLU scheme, since it involves relatively more runs of the hold logic computation algorithm (§2-A) for the generation of all points for the F/A grid. In other designs, we either simply perform a remapping (for sizing), or run the hold logic generation algorithm for one value of T_{clk} (for static MOHL VLU).

These runtimes depend on multiple factors: the size of the circuit, the logic functionality (which changes the size of BDD's used) and the distribution of paths in the circuit and thus do not show a monotone trend with any one of these factors: e.g., two circuits with comparable size (s3384 and c2670) have very different runtimes. Similar observations have also been made in [2], [4].

We categorize the circuits in Table I into two sets, named Set1 and Set2, based on the throughput improvements ($\Delta\eta$) obtained. Our method is generally successful on circuits in Set1 and not so on those in Set2. We will analyze this further, discuss the root causes, and present a technique in §7-C for predetermining whether a circuit can benefit from the use of our methods (and in general, from VLUs).

2) ***Area and Throughput Analysis****:* All comparisons shown here are with respect to the padded baseline one-cycle circuit.

VLU Sizing-Based Padding: An area overhead is induced due to (a) sizing and (b) additional hold logic. Although this method yields the highest throughput over all designs, the area overhead induced is also the highest, similar to or even greater than the baseline. Since the baseline is a combinational design, T_{clk} for the baseline is always greater than that of VLUs, as demonstrated for the RCA in §1.

Hybrid Padding: The sizing overhead of this method is due to the insertion of hold logic (§7-A): in some circuits, this is large enough that the net area overhead is positive.

Static MOHL VLU: The static MOHL incurs an average savings in area as compared to the worst-case design. In considering the throughput overhead, it is important to note that $\Delta\eta$ for this method provably decreases monotonically with time, as more paths move to two-cycle operation with time. This is illustrated in Figure 7.

Let us examine the $\Delta\eta$ values for the hybrid padding method and the static MOHL VLU design. For the Set1 benchmarks, the hybrid method shows only a small throughput degradation while the static method shows a positive or negative change. For the Set2 benchmarks, both methods show large overhead.

On analyzing this further, we determined that this is caused because the circuits in Set2 have either a zero or small margin between the delays of the near-critical paths and other paths in the circuit. This observation was also made for such circuits in [13] for MCNC circuits that proved to be hard to optimize for average-case operation. Thus, as we move along a particular column of the F/A grid of Figure 5, the value of P_{hold}) changes rapidly from a small value (such as 0.05) to a large value (such as 0.99), implying that upon degradation, almost all paths are moved to second cycle, resulting in large negative values of $\Delta\eta$. This is also confirmed by the $\Delta\eta$ results of the adaptive MOHL VLU design (discussed next), where such circuits yield only a very

[2]It is possible to use a baseline corresponding to a nominal unaged circuit without padding: in such a case, the overhead would always be positive. However, such an uncompensated circuit does not meet performance specifications, and we prefer a comparison with a functionally correct circuit. Such a comparison also shows the advantages of BTI-resilient VLUs over single-cycle implementations.

small enhancement in throughput.

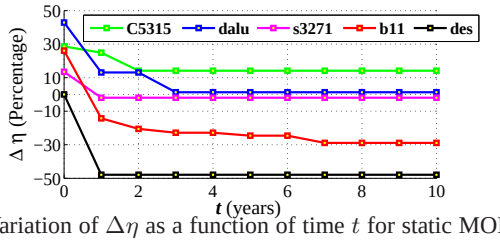


Fig. 7: Variation of $\Delta\eta$ as a function of time t for static MOHL VLU design for a subset of the benchmark circuits.

Adaptive MOHL VLU: Empirically, we see that allowing the choice of $\mathcal{H}$-sets from different columns of the F/A grid, not only allows for throughput enhancement *throughout* the lifetime, but also requires fewer hold logics for correct functionality, and hence the largest area savings. We also see that only a small amount of V_{bb} (maximum 0.25V) is necessary for all the circuits. We also note that amongst all designs, adaptive MOHL achieves lowest area overhead (-9.1%), with throughput enhancements quite close to the highest throughput enhancements of the sizing-based padded VLU. These negative overhead numbers are significant, for a lower area also requires lower power requirements.

For some cases in Set1 (C2670, cmb, i5, b11), although the hybrid scheme results in only a small throughput degradation, and the adaptive scheme results in good throughput enhancement, the static MOHL VLU gives a relatively higher throughput degradation. We observe that such circuits have a large number of paths with delays close to the maximum one-cycle path delay in the nominal VLU, and only a few paths with delays close to that of the critical-path delay of the circuit. With delay-degradation, static MOHL moves all such paths to second cycle, experiencing a higher throughput degradation. Since such circuits also have only a few paths with delays close to the critical-path delay of the circuit, they do not suffer much throughput degradation with hybrid design, and show significant throughput gain with adaptive MOHL design.

We can conclude from the above analysis, that for benchmarks in Set1, the static MOHL VLU scheme with ABB proves to be most suitable in ensuring BTI resilience with large savings in area, and significant gains in throughputs throughout lifetime. For benchmarks in Set2, combinational or VLU sizing schemes may perform as well as adaptive MOHL but much better than the static MOHL scheme.

C. Benchmark Categorization

Although our results have categorized benchmarks into Set1 and Set2 after analysis, it would be useful for a designer to be able to do so *a priori*, without having the need of constructing the F/A grid and of the subsequent analysis. We present a method for such categorization. For this method, we only need to work with the nominal combinational design of the circuit.

As highlighted in §7-B, at a given T_{clk}, for circuits that belong to Set2, numerous paths delays are close to the longest path delay, and these are moved to the second cycle as the circuit degrades with aging. We identify ϵ-critical paths in the nominal design: paths whose delay values lie within a fraction ϵ of becoming critical, i.e., delays in the interval $[\epsilon D_c, D_c]$, where $D_c = D(t = 0)$ is the critical path delay. We choose $\epsilon = (1 - \frac{\Delta D(t_{life})}{D_c})$, since paths that are within ϵ of D_c are likely to age so that their delays increase and violate T_{clk}. If the number of such ϵ-critical paths is large, we will likely incur low gains, or losses, in throughput in the VLU-based designs.

A measure of this change in throughput can be computed as follows. Let η_1 be the throughput of the nominal circuit ($T_{clk} = D_c$), and let η_2 be the throughput if we set $T_{clk} = \epsilon D_c$. Using equation (1):

$$\eta_1 = \frac{1}{D_c} \text{ and } \eta_2 = \frac{1}{(1 + P_{hold}) \cdot \epsilon D_c} \tag{4}$$

where the P_{hold} value is obtained by the hold logic generation algorithm in §2-A. The throughput change is then computed as:

$$\Delta\eta = \frac{\eta_2 - \eta_1}{\eta_1} = \frac{1}{\epsilon \cdot (1 + P_{hold})} - 1 \tag{5}$$

We choose $\Delta\eta_{tol}$ = -25% as the tolerance on the estimated percentage change in the throughput incurred (compared to nominal design) when the ϵ-critical paths are moved to the second cycle. If $\Delta\eta$ (from equation (5)) $\geq \Delta\eta_{tol}$, the design is categorized in Set2. For such designs, the P_{hold} value, as expected, is seen to be quite large. We have found this choice to work well for all the benchmarks tested.

Note that percentage of ϵ-critical paths, with respect to the total number of paths in the circuit, is equal to $(1 - P_{hold}) \times 100$ (P_{hold} generated with $T_{clk} = \epsilon D_c$), for P_{hold} essentially is the fraction of total number of paths that have delays less than ϵD_c. This can be computed *in linear time* since the computational cost for determining such paths is the same as the hold logic generation algorithm: O(n).

§8. CONCLUSION

We have presented a novel BTI-resilience scheme that exploits the average-case performance of the circuit, through an efficient MOHL scheme. We have augmented the MOHL VLU with an adaptive body bias framework to achieve maximal throughputs throughout the lifetime, and demonstrated the efficacy of the method.

ACKNOWLEDGMENT

This work was supported in part by the NSF under award CCF-1017778.

REFERENCES

[1] M. A. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Reliability*, vol. 45, pp. 71–81, January 2005.

[2] L. Benini, G. D. Micheli, A. Lioy, E. Macii, G. Odasso, and M. Poncino, "Automatic synthesis of large telescopic units based on near-minimum timed supersetting," *IEEE Transactions on Computers*, vol. 48, pp. 769–779, August 1999.

[3] S. Ghosh, S. Bhunia, and K. Roy, "CRISTA: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 1947–1956, November 2007.

[4] Y. S. Su, D. C. Wang, S. C. Chang, and M. S. Malgorzata, "Performance optimization using variable-latency design style," *IEEE Transactions on Very Large Scale Integration Systems*, vol. PP, no. 99, pp. 1 –10, 2010.

[5] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for better than worst-case design," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 2–7, 2005.

[6] Y. Chen, H. Li, J. Li, and C. K. Koh, "Variable-latency adder (VL-adder): new arithmetic circuit design practice to overcome NBTI," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 195–200, 2007.

[7] J. W. Tschanz, J. Kao, S. G. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1396–1402, November 2002.

[8] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Adaptive techniques for overcoming performance degradation due to aging in CMOS circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 19, pp. 603–614, April 2011.

[9] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negative bias temperature instability," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 493–496, 2006.

[10] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, pp. 509–517, December 2007.

[11] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification, Release 70930.

[12] Predictive Technology Model. http://www.eas.asu.edu/~ptm.

[13] J. Cong and K. Minkovich, "Mapping for better than worst-case delays in LUT-based FPGA designs," in *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pp. 56–64, 2008.