GNOMO: Greater-than-NOMinal V_{dd} Operation for BTI Mitigation

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

***Abstract*—This paper presents a novel scheme for mitigating delay degradations in digital circuits due to bias temperature instability (BTI). The method works in two alternating phases. In the first, a greater-than-nominal supply voltage, $V_{dd,g}$ is used, which causes a task to complete more quickly but causes greater aging than the nominal supply voltage, $V_{dd,n}$. In the second, the circuit is power-gated, enabling the BTI recovery phase. We demonstrate, both at the circuit and the architectural levels, that this approach can significantly mitigate aging for a small performance penalty.**

I. Introduction

A major component of run-time delay changes in digital circuits is attributable to negative/positive bias temperature instability (NBTI/PBTI) in PMOS/NMOS devices; collectively, these effects are referred to as BTI. In a CMOS gate, when an NMOS (PMOS) device is stressed under BTI, typically by applying a logic 1 (logic 0) at its gate input, its threshold voltage degrades, resulting in an increase in the gate delay. When the stress is removed, there is partial (but not complete) recovery in the threshold voltage, and hence the delay.

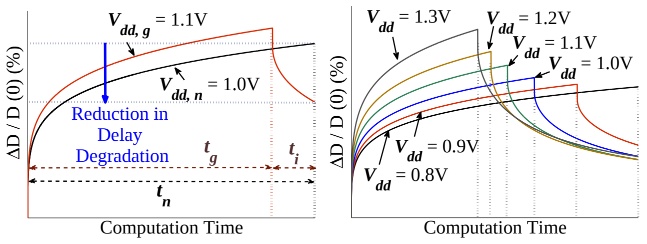
Various approaches have been proposed to overcome this degradation. Some methods introduce delay guardbands using sizing or resynthesis [1] to add a delay margin to the nominal ($t = 0$) design. At the circuit level, adaptive body bias/adaptive supply voltage schemes [2], [3] compensate for BTI by dynamically increasing the values of V_{dd} and V_{bb} voltages to speed up the circuit, compensating for aging-related degradation. Since the optimum for each circuit block may be different, this could involve the generation of a large set of V_{dd} and V_{bb} values, which poses a significant challenge.

Chip-level dynamic voltage scaling (DVS) schemes [4]–[6] to recapture lost performance overcome this problem by dynamically varying the supply voltage at the processor level. These methods also mitigate BTI by managing the workload amongst multiple cores. DVS schemes, however, have benefits only in the early lifetime [7] of the chip as the BTI degradation occurs rapidly in the first few months.

State-based schemes detect the idle states of the circuit during computation [8], [9], and apply a suitable recovery mechanism to lower the degradation. Other methods in this class distribute tasks over partitioned functional units to balance aging [10] and perform node vector control [11] or power gating [12] during idle times.

Such idle-state approaches have some common limitations. First, the idle states are workload/circuit configuration dependent: the precise idle times tend to be unpredictable, or difficult to predict dynamically. Hence, the schemes require a complex hardware/software control mechanism that can (a) dynamically detect the idle times during execution, (b) apply the appropriate recovery mechanism, and (c) keep track of which parts of the circuit have partially recovered after the idle time, and by how much. Second, it may not be easy to exploit such idle times fully, for modern out-of-order execution and multi-threading endeavor to hide idle periods. A better approach would be to have predictable idle times of fixed durations, requiring a potentially much simpler control mechanism.

We propose GNOMO, a novel and superficially counterintuitive scheme for mitigating BTI that works both at the circuit and the architecture-level by introducing *predictable idle times*. Going against the conventional wisdom that operation at a higher V_{dd} will result in a higher delay degradation, we show that by elevating V_{dd} to an optimal, greater-than-nominal value, we can achieve a lower delay degradation than that incurred at the nominal V_{dd}.



(a) (b)

Fig. 1: The delay degradation patterns of MCNC benchmark alu4 at (a) nominal supply voltage $V_{dd,n} = 1$V and greater-than-nominal supply voltage $V_{dd,g} = 1.1$V, and (b) $V_{dd} \in [0.8\text{V}, 1.3\text{V}]$ values.

We illustrate this idea in Fig. 1(a) through the example of an ALU (MCNC benchmark alu4). Note that monotone degradation under stress shown here captures the effect of alternate stress/recovery cycles and plots the *envelope* of BTI degradation [13].

With a nominal supply voltage value $V_{dd,n} = 1.0$V, the ALU requires a computation time t_n for a given workload. In GNOMO (greater-than-nominal operation), at a higher supply voltage $V_{dd,g}$, the ALU has a lower delay and works at a higher clock frequency, requiring only $t_g < t_n$ time for completing the whole computation. As $V_{dd,g} > V_{dd,n}$, the degradation rate while in GNOMO is higher during $t \in [0, t_g]$. However, since $t_g < t_n$, additional idle time $t_i = t_n - t_g$ is generated during which the ALU can be power-gated while maintaining the same throughput. This enables recovery, which may lower the overall delay degradation of the ALU at $t = t_n$, as compared to the nominal operation. Fig. 1(b) illustrates this for a different baseline voltage, $V_{dd,n} = 0.8$V, and several $V_{dd,g}$ values. Note that the idle times show diminishing returns as $V_{dd,g}$ increases.

An additional consideration in GNOMO is the increased power consumption at the higher $V_{dd,g}$ value. The generated idle times, however, serve to lower these overheads. Thus, there is a trade-off with an optimal operating point that can appreciably reduce the degradation while incurring small power/performance overheads (and as we show, even some power gains). Our paper presents the approach for exploring such an operating point, and its practical adoption at the circuit/architectural level. The salient features and contributions of our work can be summarized as follows:

First, given a nominal V_{dd}, GNOMO statically determines the optimal greater-than-nominal V_{dd} for each functional block. We then present a simple architectural-level control framework for operating the entire processor at a common greater-than-nominal V_{dd}, thus bridging the gap between circuits and architecture.

Second, we show that GNOMO enables a reduction of about 20%-45% in delay degradation. For the same lifetime, this reduction in degradation implies that reduced guardbands are necessary as compared to the nominal voltage case. This yields a reduction of up to about 2x in the area and power overheads.

Third, GNOMO does not require fine-grained voltage supplies/control, nor does it require the detection of idle times (or the potentially complex associated circuitry) for idle times are *generated*, and not detected, and are hence predictable-by-construction.

We first present the preliminaries for this work in Section II. Section II-B then presents the control framework for GNOMO, followed by our methodology for finding the optimal GNOMO V_{dd} for a particular circuit in Section IV. We then present our results and conclusion in Sections V and VI.

II. Preliminaries

A. BTI Modeling

We work with a widely adopted model [13] for predicting delay degradation due to BTI. We present an expression for PMOS NBTI under alternate stress/relax cycles, for a given V_{dd} and signal probability α at the input of the PMOS (for PBTI, similar equations may be used since the mechanism of NMOS delay degradation is similar to that of PMOS, albeit with a lower degradation magnitude [3]):

$$\textit{Stress: } \Delta V_{th}(t) = \left(K_v\sqrt{t-t_0} + \sqrt[2n]{C(t-t_0)}\right)^{2n} \tag{1}$$

$$\textit{Recovery: } \Delta V_{th}(t) = \Delta V_{th}(t_0)\Big(1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C(t-t_0)}}{2t_{ox} + \sqrt{Ct}}\Big) \tag{2}$$

Long-term model:

$$\Delta V_{th}(t) = \frac{(K_v^2 \alpha T_{clk})^n}{(1-\beta_t^{1/2n})^{2n}};\ \beta_t = 1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C(1-\alpha)T_{clk}}}{2t_{ox} + \sqrt{Ct}} \tag{3}$$

where equations (1) and (2) model all-stress and all-recovery. The long-term model in equation (3) predicts the envelope of the BTI degradation pattern with alternating stress and recovery. The precise definitions of the symbols above may be found in [13], but it is important to note that:

- the exponent n = 1/6.
- K_v (and hence $\Delta V_{th}(t)$) is a superlinear function of V_{dd}.

B. Delay and Power Modeling

As in past research, we use compact sensitivity-based performance models for the delay (D) and the logarithm of the leakage power ($\log L$) in terms of V_{th} [3]. For $\mathcal{X} \in \{D, \log L\}$, we characterize

$$\mathcal{X}(t) = \mathcal{X}_0 + \sum_{i=1}^{n} \frac{\partial \mathcal{X}}{\partial V_{th_i}} \Delta V_{th_i}(t) \tag{4}$$

where $\partial\mathcal{X}/\partial V_{th_i}$ denotes the sensitivity of the quantity $\mathcal{X}$ with respect to the V_{th} of the i^{th} transistor along the input-output path.

III. **GNOMO**: **G**reater-Than-**NOM**inal V_{dd} **O**peration

We now employ architectural-level analysis to present the GNOMO framework for BTI mitigation. We use the term $V_{dd,n}$ to refer to the nominal supply voltage and $V_{dd,g}$ to the GNOMO value.

A. Voltage Supplies and Operational Frequencies

It is important to emphasize that GNOMO is *not* an adaptive supply voltage scheme (ASV) for BTI mitigation: under GNOMO, the processor operates at a constant voltage and frequency (however, it is possible to apply the GNOMO framework when ASV is required for power management). In our implementation, the elevated V_{dd} may take one of several values; each such value corresponds to a different frequency of operation for the processor. We assume processor operation with realistic discrete supply voltage/frequency (V_{dd}/f) pairs, adopted from Intel's recent 48-core IA-32 Processor [14] as shown in Table I, where V_{dd} lies in the range [0.7V, 1.3V] (this choice is only for illustration purposes; any other V_{dd}/f framework can be used instead). This allows us to operate within the framework of existing technologies to illustrate the principles of GNOMO.

TABLE I: Operational V_{dd}/f pairs adopted from Intel's IA-32 Processor [14]

V_{dd} (Volts)	0.7	0.8	0.9	1.0	1.1	1.2	1.3
Frequency (GHz)	0.25	0.47	0.68	0.86	1.03	1.17	1.30
T_{clk} (ns)	4.00	2.13	1.47	1.16	0.97	0.85	0.77

It is worth noting that Table I shows that with a linear increase in the value of V_{dd}, the increase in the clock frequency is only sublinear. The effect of this will be discussed further in Section IV-A.

B. Circuit Recovery through Power Gating

1) Motivating Intuition: The essential idea of GNOMO is inspired by the intuition depicted through Fig. 1. In principle, for a fixed clock period, GNOMO finishes the computations early in every clock cycle. We can then potentially "switch-off" the circuits during the fraction of the clock cycle when they do not function (i.e., they remain idle), so that the circuits can recover sufficiently to gain a lower delay degradation after every clock cycle. This recovery can then be accumulated over all the clock cycles as the circuit functions, and lower its overall delay at the end-of-lifetime.

However, it is completely impractical to implement such a scheme, where circuits must be put to sleep and woken up within a single clock cycle. However, the idea can be extended to a realistic architectural framework: instead of switching-off/waking-up the circuit within each cycle, we run the circuit for a large number of cycles and then introduce a predetermined amount of idle time (also corresponding to a large enough number of cycles so that the overheads of sleep/wake-up are amortized). This idea effectively provides the same sleep/wakeup "duty cycle" as in the concept above. By the concept of frequency independence of BTI, the degradation/recovery depends on the duty cycle rather than the precise distribution of on/off periods, and therefore this alternative, more practical formulation results in the same amount of recovery as the conceptual idea presented earlier. In such a scenario, it is easy to switch-off the circuit (e.g., by power gating) as both computational and idle times are significantly larger than the switch-off time.

2) Mechanism: The practical implementation of GNOMO works as follows: the processor runs at GNOMO supply voltage, $V_{dd,g}$, for a certain number of cycles, and then sleeps for some cycles. It continues execution in this intermittent way throughout its lifetime.

For a given workload, consider the operation of the processor during one of these periods, corresponding to a fixed number of clock cycles, c_f. Let the number of instructions executed, while operating at $V_{dd,n}$ ($V_{dd,g}$), be I_n (I_g), and let the corresponding execution time be t_n (t_g) time units. Clearly,

$$t_n = c_f \cdot T_{clk,n} \text{ and } t_g = c_f \cdot T_{clk,g} \tag{5}$$

These durations are termed as the *compute-phases*. Since the computation is completed earlier under GNOMO, an additional *idle phase* of time duration t_i (c_i cycles) is generated after the compute phase, during which the circuits do not perform any computation. Note that this idle phase is deliberately inserted and therefore easily predictable, and is thus different from the idle periods that may occur within the compute phase due to cache misses, TLB misses, branch mispredictions, etc. Fig. 2 depicts the GNOMO scheme, showing the time (in cycles as well as seconds) along the x-axis, and V_{dd} along the y-axis. During the compute phase, the processor is active with $V_{dd} = V_{dd,g}$, and transitions to the idle state with $V_{dd} = 0$V when the computation is finished. Upon the completion of the idle phase, the processor then enters the next compute phase.

For circuit recovery during this idle phase, power gating (using the existing on-chip power gating framework) is applied to all the circuits, incurring an overhead of t_s time units (c_s cycles) for the circuits to

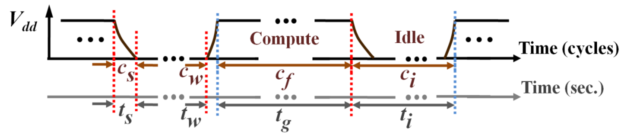


Fig. 2: The compute and idle phases in GNOMO in the practical implementation.

transition to the sleep state. This operation is applied to computational units rather than caches (which may need to preserve state): the internal nodes of computational units do not need to save their logical states since the computation is complete; therefore, during the idle phase, these circuits undergo recovery. The circuits are then woken up again before the idle phase completes (to execute the instructions in the next c_f cycles), incurring an overhead of t_w time units (c_w cycles) for wakeup. The sleep/wake-up transitions are deliberately designed to occur inside the idle phase, ensuring that the execution of instructions is not affected by the GNOMO scheme.

3) Overheads: Existing power gating frameworks offer sleep transition times (c_s) of about 10 to 50 cycles for various circuits in a processor [12]. The wakeup time (c_w) is typically about 5-10 cycles. Since these transitions are designed to occur within the idle phase, the effective idle time may decrease significantly if c_s and c_w are comparable to c_i. Moreover, the sleep/wake-up transitions consume power, which incur a power overhead. These overheads, however, are easily compensated for and become negligible by addressing some of the architectural-level issues that arise in implementing GNOMO, as discussed shortly in Section V-B.

C. Idle Time Generation – Practical Considerations

Recall that the number of instructions executed in c_f cycles at $V_{dd,n}$ and $V_{dd,g}$ are I_n and I_g, respectively.

"Ideal" Case: If the frequencies of all the components in a CPU (both on-chip and off-chip components) were to scale at the same rate as V_{dd} is changed, as dictated by Table I, the number of instructions executed in c_f cycles would be the same, i.e., $I_n = I_g$. The idle time $t_{i,1}$ may be computed as:

$$t_{i,1} = t_n - t_g = c_f \cdot (T_{clk,n} - T_{clk,g}) \tag{6}$$

Idle time $t_{i,1}$ is shown in Fig. 3(a), which presents both the nominal operation and GNOMO, over one set of c_f cycles, with time along the x-axis and percentage delay degradation along the y-axis. Fig. 3(a) shows that during the nominal operation ($V_{dd} = V_{dd,n}$ and $T_{clk} = T_{clk,n}$), c_f cycles take t_n time to execute. At GNOMO ($V_{dd} = V_{dd,g}$ and $T_{clk} = T_{clk,g}$), the same c_f cycles take less amount of time, t_g, to execute, generating $t_{i,1}$ idle time.

Note that the extra computation depicted by t_o in Fig. 3(a) does not occur in this ideal case, but is seen in a more realistic case as discussed next.

Realistic Case: In practice, the voltages and frequencies are scaled only for on-chip components (processor, cache, on-chip buses, etc.), but remains the same for off-chip components (memory, memory buses, and memory controllers). Hence, the access *time* for off-chip memory (upon a cache miss) remains the same at both $V_{dd,n}$ and $V_{dd,g}$, but this time corresponds to a larger number of cycles under the faster clock at $V_{dd,g}$. Therefore, during the fixed number of c_f

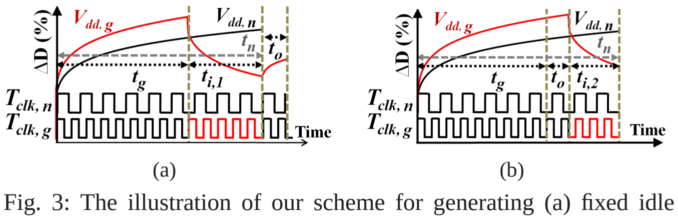


Fig. 3: The illustration of our scheme for generating (a) fixed idle time, $t_{i,1}$, and (b) variable idle time, $t_{i,2}$.

clock cycles, the number of instructions executed under GNOMO will be smaller: $I_g = I_n - I_o$, where I_o is the number of overhead instructions that still need to be executed. Let these I_o instructions require an overhead of c_o number of clock cycles (t_o time units) for completion.

This overhead can be accommodated in two ways: first by keeping the duration of idle phase fixed (= $t_{i,1}$) and deferring the execution of I_o instructions *after* the idle phase, as shown in Fig. 3(a). This incurs a performance penalty of t_o time units (we show that this performance penalty is small in Section V-B). The alternative way is to execute the I_o instructions *within* the idle phase, as shown in Fig. 3(b) (which shows the same operations as in Fig. 3(a), except for the placement of the execution overhead). This reduces idle time from $t_{i,1}$ to $t_{i,2}$:

$$t_{i,2} = t_n - (t_g + t_o) = t_{i,1} - t_o \tag{7}$$

This reduction in idle time also reduces the overall recovery possible, albeit without a performance penalty. Further, for a specific value of $V_{dd,n}$, the value of $t_{i,1}$ is *fixed* as it depends only on the fixed number of cycles c_f and the frequency corresponding to $V_{dd,n}$, which is also fixed according to Table I. On the other hand, the value of $t_{i,2}$ *varies* with the number of off-chip accesses during execution.

Based on these observations, we now outline a mechanism for generating idle times: after completing c_f cycles, this scheme uses $t_{i,1}$ time units ($c_{i,1}$ cycles) as the fixed idle time, for gaining recovery for BTI mitigation.

D. Idle Time Generation – Framework

In this scheme, we generate a fixed amount of idle time $t_{i,1}$, given by equation (6), by applying power gating after completing the c_f number of clock cycles. Since the idle phase duration is fixed, this ensures a fixed and predictable amount of recovery.

As seen in Section III-C, in practice, an overhead of c_o cycles is required to complete the remaining I_o instructions at $V_{dd,g}$. Since the idle times are fixed, the completion time of $I_g + I_o$ instructions is delayed by c_o cycles, i.e., there is a performance penalty involved, but the amount of recovery time is guaranteed. The total performance penalty for a particular workload is given by:

$$\text{Performance Penalty} = \frac{t_o}{t_n} = \frac{c_o \cdot T_{clk,g}}{c_f \cdot T_{clk,n}} \tag{8}$$

Our scheme relies on the assumption that given a $V_{dd,n}$ for nominal operation, all the circuits in a chip are operated at the same value of $V_{dd,g}$ for GNOMO (as the V_{dd}/f pairs and power gating framework work for *all* circuits in the chip). We show in Section IV-C that this is indeed true for our scheme.

IV. Optimal Choice for GNOMO V_{dd}

The framework of idle time generation was presented through architectural-level considerations in the previous section. We now examine the implications at the circuit level and determine the optimal GNOMO V_{dd} value based on circuit-level considerations.

A. Delay Degradation as a Function of $V_{dd,g}$

The fraction of the required idle time at the GNOMO $V_{dd,g}$ value in the "ideal" case, to the execution time for c_f cycles at $V_{dd,n}$, can be computed as valid combinations of ($V_{dd,n}$, $V_{dd,g}$). These are computed as follows:

$$\frac{t_{i,1}}{t_n} = \frac{c_f \cdot (T_{clk,n} - T_{clk,g})}{c_f \cdot T_{clk,n}} = 1 - \frac{T_{clk,g}}{T_{clk,n}} \tag{9}$$

Table II shows this percentage: note that it is a reasonable approximation of the realistic time, $t_{i,2}$, since our experiments show that the performance penalty is never more than 5.5%, and often much less. We observe the following diminishing returns in idle times:

- For a particular value of the nominal supply voltage $V_{dd,n}$ (say 0.8V), a linear increase in the value of $V_{dd,g}$ (along the row from 0.9V to 1.3V) increases the idle time durations only in a sublinear fashion. The corresponding increase in the degradation rate, however, is almost quadratic [13]). This implies that the time for recovery at higher values of $V_{dd,g}$ will not sufficiently reduce the additional degradation that occurs with GNOMO.
- At higher values of $V_{dd,n}$ ($\geq$ 1.1V), the available idle time is low.

TABLE II: Percentage idle time $t_{i,1}$ for various $(V_{dd,n}, V_{dd,g})$

$V_{dd,g} \rightarrow$ / $V_{dd,n} \downarrow$	0.8	0.9	1.0	1.1	1.2	1.3
0.7	46.8%	63.2%	70.9%	75.7%	78.6%	80.8%
0.8	–	30.8%	45.3%	54.3%	59.8%	63.9%
0.9	–	–	20.9%	34.0%	41.9%	47.7%
1.0	–	–	–	16.5%	26.5%	33.6%
1.1	–	–	–	–	12.0%	20.8%
1.2	–	–	–	–	–	10.0%

These idle times correspond to available time for BTI recovery, and therefore the delay degradation improvements also show diminishing returns. Fig. 4(a) shows the variation in percentage delay degradation at the end-of-lifetime for benchmark alu4 with $V_{dd,n}$ = 0.8V and for various values of $V_{dd,g}$ (similar trends are seen for other values of $V_{dd,n}$). The data in the figure assumes a random workload for the ALU (a random distribution of signal probabilities at the PIs), which mimics the variation in workload on the ALU at the architectural level. As $V_{dd,g}$ is increased, percentage ΔD first decreases, reaches a minimum at 1.1V, and then increases again. The initial trend can be attributed to the large recovery times; for higher $V_{dd,g}$ values, this is counteracted by the increased BTI effects. Therefore, there is the notion of an optimal value of $V_{dd,g}$ for a given value of $V_{dd,n}$.

B. Power Dissipation as a Function of $V_{dd,g}$

GNOMO impacts the power dissipation as follows:

- With the supply voltage increased to $V_{dd,g}$, dynamic power increases quadratically and leakage increases exponentially [15].
- Even though the V_{th} increase due to BTI degradation is small, the exponential relationship in leakage power leads to a significant reduction over the lifetime of the chip [3]. This effect is more pronounced at higher V_{dd} values (due to increased BTI degradation).
- A further reduction in the average dynamic and leakage power consumption occurs due to generation of idle time, since power dissipation now occurs only in the compute-phase and not in the idle-phase (except during sleep/wake-up cycles), which is a fraction of the total nominal computation time.

Fig. 4(b) illustrates the variations of the normalized (by the nominal value) average dynamic, leakage and total (dynamic + leakage) power consumption for a typical case with $V_{dd,n}$ = 0.8V and $V_{dd,g}$ = 0.9V to 1.3V (plotted under the same workload conditions as Fig. 4(a)). Similar trends are seen for other values of $V_{dd,n}$. It can be seen that the dynamic power increases subquadratically with V_{dd} due to the

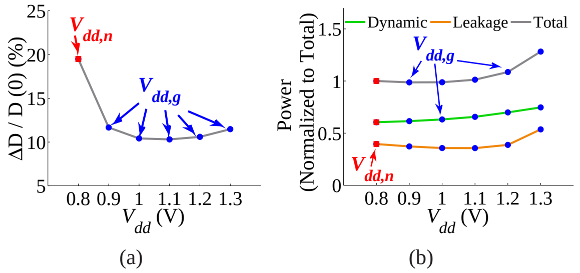


Fig. 4: (a) Delay degradation and (b) normalized dynamic, leakage and total power for for alu4 as a function of $V_{dd,g}$; $V_{dd,n} = 0.8$V.

effect of idle times, but the leakage increases significantly, especially at higher values of V_{dd}. The total power consumption remains about flat until $V_{dd,g}$ = 1.0V and then begins to increase beyond this point.

C. Choosing the Optimal GNOMO Supply Voltage

The discussion above is summarized as follows: for $V_{dd,n}$ = 0.8V, the delay degradation, dynamic power, leakage power, and total power are minimized at $V_{dd,g}$ = 1.1V, 0.8V, 1.1V, and 1.0V, respectively. An optimal choice of $V_{dd,g}$ must balance these individual optima.

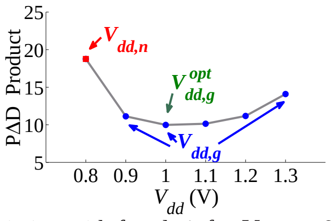


Fig. 5: PΔDP variation with for alu4, for $V_{dd,n}$ = 0.8V and for $V_{dd,g}$ = 0.9V to 1.3V, illustrating the optimality criterion for choosing the optimal $V_{dd,g}$ for GNOMO.

Fig. 5 captures this by plotting the Power-ΔD-product (PΔDP), the product of the percentage delay degradation and the normalized total power. We choose an optimal point on this plot, subject to the requirements that the power overhead should be within 10% of the optimal value. From simulations, we find that the for $V_{dd,n} \leq$ 1.1V, the total power overhead is less than 7% with GNOMO. For the data shown in the figure, the optimal GNOMO supply voltage is $V_{dd,g}^{opt}$ = 1.0V. This value corresponds to end-of-lifetime delay degradation improvement from 19.5% to 10.4%.

TABLE III: The optimal $V_{dd,g}$ values for various values of $V_{dd,n}$

$V_{dd,n}$ (V)	0.7	0.8	0.9	1.0	1.1	1.2
$V_{dd,g}^{opt}$ (V)	1.0	1.0	1.1	1.1	1.2	1.2

The above arguments can be used to find the best choice of $V_{dd,g}$ for each value of $V_{dd,n}$ for the circuit alu4. We have performed the same analysis for a large number of ISCAS85, MCNC, and ITC99 benchmarks: their details are described in Section V.

Interestingly, the optimal $(V_{dd,n}, V_{dd,g})$ pairs for each circuit are identical, and the results are recorded in Table III. Intuitively, this is because all circuits use the same V_{dd}/f pairs, which yields the same idle time, as shown in equation (9). This can be formally proved, and the proof is omitted due to space limitations. Loosely, it is based on the idea that, using the models in Section II-A, the delay degradation at time t_n is proportional to the delay degradation at time t_g, where the proportionality constant depends on the t_i/t_n ratio. Since the t_i/t_n ratio is strictly determined by the values of $V_{dd,n}$ and $V_{dd,g}$, and not by the circuit, the result follows.

We make the following observations about the table data:

- For $V_{dd,n}$ = 1.2V, $V_{dd,g}^{opt}$ = $V_{dd,n}$ and no gain is possible. This is attributed to the fact that the total power increases by 15.5% for $V_{dd,g}$ = 1.3V candidate value, exceeding the 10% threshold: this is primarily attributed to a steep increase in the leakage power due to the higher voltage value and also due to the low idle time.
- GNOMO scheme works best when the value of $V_{dd,n}$ is lower, but it provides significant improvements for all $V_{dd,n} \leq$ 1.1V.

V. Results

We now present the results of applying GNOMO at the circuit and architectural levels.

A. Circuit-level Results

At the circuit level, we examine the application of GNOMO on various ISCAS85, MCNC and ITC99 benchmarks, synthesized using ABC [16] on the 32nm PTM [17] based library. Our library consists

TABLE IV: Delay degradation and area, power overhead results for end-of-lifetime BTI compensation, for 3 sets of ($V_{dd,n}$, $V_{dd,g}^{opt}$)

Circuit	ΔD (%)						ΔA (%)						ΔP (%)					
	$V_{dd,n}$ 0.8V	$V_{dd,g}^{opt}$ 1.0V	$V_{dd,n}$ 0.9V	$V_{dd,g}^{opt}$ 1.1V	$V_{dd,n}$ 1.0V	$V_{dd,g}^{opt}$ 1.1V	$V_{dd,n}$ 0.8V	$V_{dd,g}^{opt}$ 1.0V	$V_{dd,n}$ 0.9V	$V_{dd,g}^{opt}$ 1.1V	$V_{dd,n}$ 1.0V	$V_{dd,g}^{opt}$ 1.1V	$V_{dd,n}$ 0.8V	$V_{dd,g}^{opt}$ 1.0V	$V_{dd,n}$ 0.9V	$V_{dd,g}^{opt}$ 1.1V	$V_{dd,n}$ 1.0V	$V_{dd,g}^{opt}$ 1.1V
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19
alu4	19.5	10.4	20.9	12.6	23.1	15.9	16.8	4.5	19.9	6.3	25.0	10.4	15.1	5.5	17.9	8.6	22.5	13.1
b12	25.0	13.7	27.3	17.1	28.9	20.9	28.3	8.0	33.1	12.1	35.9	19.1	29.2	9.8	34.2	15.4	37.0	23.4
b15	23.9	12.6	25.4	15.4	27.9	19.7	36.0	8.8	40.7	12.5	46.7	22.6	43.4	12.1	49.1	18.0	56.2	31.0
c1908	19.9	10.3	22.0	12.6	24.5	16.8	13.7	3.6	17.2	5.4	23.0	9.5	15.1	5.5	18.9	8.8	25.3	14.2
c2670	20.0	10.6	21.3	12.7	23.1	16.1	23.7	7.5	27.9	9.0	33.6	13.7	25.7	9.7	30.3	12.6	36.5	18.6
c5315	20.8	11.4	22.6	14.0	25.0	18.0	18.9	6.3	22.9	8.3	28.4	13.6	23.0	9.2	27.8	12.9	34.5	20.2
c6288	21.1	11.8	22.8	14.3	24.9	18.0	27.2	8.2	32.5	10.7	39.4	18.1	30.7	10.7	36.7	15.0	44.5	24.1
dalu	23.8	12.8	25.3	15.5	27.7	19.6	26.7	6.5	30.6	9.8	36.4	16.9	28.1	8.4	32.3	13.2	38.4	21.5
des	20.4	11.2	22.0	13.7	24.2	17.6	24.8	7.8	29.9	10.0	37.3	17.1	29.9	10.9	36.1	15.0	44.9	24.3
i10	20.9	11.6	22.3	14.0	24.3	17.7	15.2	4.6	17.8	6.6	22.5	10.5	12.3	5.2	14.4	8.3	18.1	12.2
Avg.	21.8	11.8	23.4	14.4	25.6	18.3	23.8	6.8	28.1	9.4	33.7	15.7	26.4	9.1	31.1	13.2	37.3	21.1

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. We optimize the circuits by introducing delay margins to compensate for BTI aging, using the algorithms in [3].

1) Degradation Reduction and Area Savings: To begin with, consider the application of GNOMO to the circuit alu4 with ($V_{dd,n}$, $V_{dd,g}^{opt}$) = (0.9V, 1.1V). The area vs. delay curve for this circuit, for various target delay specifications, is shown in Fig. 6. The area values are normalized to point A, which corresponds to the uncompensated circuit for which no delay margins are added. The uncompensated circuit is designed to have a delay specification D_{spec}^{uc} = 722 ps near the "knee" of the area-delay curve since efficient area-delay trade-offs can be achieved in this region. We compare optimizations using the nominal and the GNOMO supply voltages:

- At $V_{dd,n}$, the ALU incurs a 20.9% delay degradation which is compensated by mapping the circuit with a tighter specification, $D_{spec}^{c,n}$, using the delay margin algorithm in [3]. This corresponds to point B on the curve, which incurs an additional area overhead of 19.9% over point A.
- At the GNOMO voltage, $V_{dd,g}$, the BTI degradation is reduced to 12.6%, and hence the delay margin is relaxed, corresponding to a delay specification of $D_{spec}^{c,g}$ at Point C. This reduces the area overhead to 6.3%. Thus, the area overhead for BTI compensation is reduced by 3× for GNOMO as compared to the $V_{dd,n}$ case.

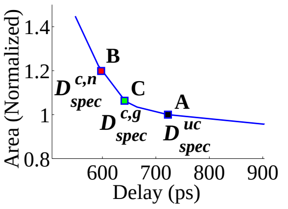


Fig. 6: The normalized-area vs. delay curve for alu4, with area normalized by the area of the uncompensated circuit.

Similar results are presented for other benchmark circuits in Table IV, which lists the end-of-lifetime percentage delay degradation (ΔD), the corresponding percentage area overheads (ΔA) and the power overheads (ΔP, as discussed next in Section V-A2) incurred for achieving BTI-compensation against this degradation, for three different ($V_{dd,n}$, $V_{dd,g}^{opt}$) pairs listed in Table III: (0.8V, 1.0V), (0.9V, 1.1V) and (1.0V, 1.1V).

For the purposes of our discussion, we use the notation Cm to denote Column m. For various benchmarks listed in C1, the percentage ΔD incurred with nominal V_{dd} operation are listed in C2, C4 and C6. The corresponding percentage ΔD values at the optimal GNOMO are shown in C3, C5 and C7. We then show the percentage ΔA incurred with nominal design in C8, C10 and C12, with the corresponding percentage ΔA at optimal GNOMO listed in C9, C11 and C13.

It can easily be seen from this data that GNOMO achieves significant reductions in delay degradation. This impacts the reduction in area overheads significantly. Further, our gains are higher when starting with lower values of $V_{dd,n}$, which corresponds with the trend of diminishing returns in idle times.

2) Power Savings: In Section IV-B, we had analyzed and shown the increment in power due to GNOMO, considering the uncompensated design (point A in Fig. 6) as the baseline for both the $V_{dd,n}$ and $V_{dd,g}$ circuits[1]. We did not, however, consider the change in power consumption due to compensation. As discussed in Section V-A1, if we consider the uncompensated design at point A in Fig. 6 as the baseline, the $V_{dd,g}$ circuit at Point C has much lower area overheads as compared to the $V_{dd,n}$ circuit at Point B[2]. Reductions in area overheads imply that the power overheads are also reduced further than our previous analysis. In this section, we conduct a more thorough analysis to determine the precise power overheads. Our analysis proceeds as follows:

- The $V_{dd,g}$ circuit corresponds to a delay-margined circuit at a supply voltage of $V_{dd,g}$, and this circuit is guaranteed to be functional throughout the projected chip lifetime. The power overheads in this circuit come from two sources: operation at the GNOMO V_{dd}, and from compensation.
- Similarly, the $V_{dd,n}$ circuit is delay-margined at a supply voltage of $V_{dd,n}$, and is guaranteed-functional throughout the projected chip lifetime. The power overheads in this circuit come only from compensation.
- As indicated by comparing the pairs of columns (C14, C15), (C16, C17), and (C18, C19) in Table IV, the $V_{dd,g}$ circuit has a significantly lower power overhead than the $V_{dd,n}$ circuit. This shows that the total power overheads of the $V_{dd,g}$ circuit are reduced by the decrease in power due to lower area requirements.

This reduction can be further elaborated upon as follows. From Table IV, the power overhead for $V_{dd,n}$ circuit is in a range from 26.4% to 37.3%. When GNOMO is used, the corresponding power overhead ranges from 9.4% to 21.1%. This includes the power overhead in Section IV-B, which never exceeds 5% (upto $V_{dd,n}$ = 1.0V as shown in the Table IV). This range is significantly below the range at $V_{dd,n}$, implying that overall power savings are achieved. Note that these ΔP values exclude the power associated with the caches and cache-like structures, which may increase at the higher $V_{dd,g}$ value. When we take this into account, *GNOMO not only reduces aging, but remains approximately neutral in terms of power dissipation, as compared to the nominal case.*

[1]For brevity, we will refer to the compensated circuit at point B in Fig. 6 as the "$V_{dd,n}$ circuit" and that at point C as the "$V_{dd,g}$ circuit."

[2]It should be noted that the uncompensated baseline does not meet the specifications over the life of the circuit, and hence is nonfunctional. The comparison should be made between the two functional versions of the circuit.

B. Analyzing the Architectural Performance Penalty

To determine the architectural performance penalty of the GNOMO scheme, SPEC 2000 benchmark suite was simulated using SimpleScalar on an out-of-order MIPS-like processor, described in Table V under the MinneSPEC input set [18]. The cycles in the workload execution were divided into sets (in the order of execution), each of size c_f cycles. We recorded the values of t_o required by every set of I_o instructions. Fig. 7 shows the average of the performance penalty (from equation (8)), over all the sets of c_f cycles for different workloads and a choice of $V_{dd,n}$ = 0.7V to 1.1V and the corresponding $V_{dd,g}^{opt}$ value from Table III.

We find that choosing c_f = 10 million ensures that at $V_{dd,n}$ = 0.7 (which shows the largest penalty), the performance penalty for GNOMO is under 2% for most of the workloads, and is 3.2% and 5.5% for the remaining two. Further, our simulations show that for the workloads with the largest overheads, such cases are rare: over 90% of the c_f sets for these workloads have $<$ 2% overheads.

This choice of a large value of c_f has other benefits. The repeated compute-standby operation in our scheme may seem to create regular interruptions in workload execution. Since c_f = 10 million, these occur much less frequently (and also predictably) as compared to the unpredictable interruptions and pipeline flushes caused by cache read/write misses, branch mispredictions, etc., which occur much more frequently and cause performance bottlenecks.

Further, the power gating overheads of 10 to 50 cycles, discussed in Section III-B, become completely negligible.

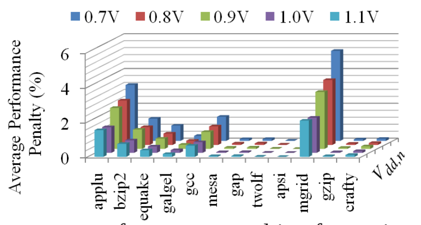


Fig. 7: The average performance penalties for various SPEC CPU 2000 workloads.

We noted that as c_f is increased from 100,000 to 10 million cycles, the maximum performance penalty over the execution of a benchmark decreases. This can be explained by the fact that a larger value of c_f corresponds to a larger number of on-chip operations, offering a greater potential for hiding latencies for off-chip operations through out-of-order execution. Thus it is better to choose a higher value of c_f. The average penalty, however, remains approximately the same.

With an increase in $V_{dd,n}$, the penalty decreases sublinearly. This is because the off-chip latency in cycles is directly related to the clock frequency shown in Table I, which also decreases sublinearly.

TABLE V: Configuration of the processor

Fetch/Decode/ Issue/Commit width	4/4/4/4 (instructions/cycle)
RUU size	64 entries
LSQ size	32 entries
Private L1 Data cache	16KB, 4-way set associative, 32B block size
Private L1 Instruction cache	16KB, 4-way set associative, 32B block size
Private L2 Unified Data and Instruction cache	512KB, 8-way set associative, 64B block size
Memory access bus width	8 bytes
Data Translation Lookaside Buffer	512KB, 4-way set associative, 4KB block size
Instruction Translation Lookaside Buffer	256KB, 4-way set associative, 4KB block size
Number of integer ALUs	4
Number of integer multiplier/dividers	4
Number of floating point ALUs	2
Number of floating point multipliers/dividers	2
Number of memory system ports available to CPU	2 (1 read, 1 write)

VI. Conclusion

This paper introduces the idea of GNOMO, where a processor is operated at a higher-than-nominal V_{dd} value with interspersed idle periods. We demonstrate at the architectural and circuit levels that this scheme is viable, and that it provides significant gains in aging with low performance overheads. The current implementation focuses on a constant nominal V_{dd}; however, in principle, the idea can be extended when the nominal case uses dynamic voltage and frequency scaling.

Acknowledgment

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

References

[1] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in *Proceedings of the Design Automation Conference*, pp. 370–375, 2007.
[2] X. Chen, Y. Wang, Y. Cao, Y. Ma, and H. Yang, "Variation-aware supply voltage assignment for minimizing circuit degradation and leakage," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 39–44, 2009.
[3] 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.
[4] U. R. Karpuzcu, B. Greskamp, and J. Torrellas, "The bubblewrap many-core: popping cores for sequential acceleration," in *Proceedings of the International Symposium on Microarchitecture*, pp. 447–458, 2009.
[5] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-aware processor," in *Proceedings of the International Symposium on Microarchitecture*, pp. 85–96, 2007.
[6] L. Zhang and R. P. Dick, "Scheduled voltage scaling for increasing lifetime in the presence of NBTI," in *Proceedings of the Asia and South Pacific Design Automation Conference*, pp. 492–497, 2009.
[7] T.-B. Chan, J. Sartori, P. Gupta, and R. Kumar, "On the efficacy of NBTI mitigation techniques," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1–6, 2011.
[8] J. Shin, V. Zyuban, P. Bose, and T. M. Pinkston, "A proactive wearout recovery approach for exploiting microarchitectural redundancy to extend cache SRAM lifetime," in *Proceedings of the International Symposium on Computer Architecture*, pp. 353–362, 2008.
[9] L. Li, Y. Zhang, J. Yang, and J. Zhao, "Proactive NBTI mitigation for busy functional units in out-of-order microprocessors," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 411–416, 2010.
[10] T. Siddiqua and S. Gurumurthi, "A multi-level approach to reduce the impact of NBTI on processor functional units," in *Proceedings of the Great Lakes Symposium Symposium on VLSI*, pp. 67–72, 2010.
[11] D. R. Bild, G. E. Bok, and R. P. Dick, "Minimization of NBTI performance degradation using internal node control," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 148–153, 2009.
[12] A. Calimera, E. Macii, and M. Poncino, "NBTI-aware clustered power gating," *ACM Transactions on Design and Automation of Electronic Systems*, vol. 16, pp. 1–25, November 2010.
[13] S. Bhardwaj, W. Wenping, R. Vattikonda, C. Yu, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," in *Proceedings of the Custom Integrated Circuits Conference*, pp. 189 –192, 2006.
[14] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart, "A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 173–183, January 2011.
[15] S. Bhunia and S. Mukhopadhyay, *Low-power variation-tolerant design in nanometer silicon*. Springer, 2010.
[16] Berkeley Logic Synthesis and Verification Group, ABC: A System for Sequential Synthesis and Verification, Release 70930.
[17] Predictive Technology Model. http://www.eas.asu.edu/∼ptm.
[18] A. J. KleinOsowski and D. J. Lilja, "MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research," *Computer Architecture Letters*, vol. 1, pp. 7–7, 2002.