Combating Aging with the Colt Duty Cycle Equalizer

Erika Gunadi, Abhisek A. Sinkar, Nam Sung Kim, Mikko H. Lipasti
Department of Electrical and Computer Engineering
University of Wisconsin - Madison
erika.gunadi@gmail.com, sinkar@wisc.edu, nskim3@wisc.edu, mikko@engr.wisc.edu

***Abstract*--Bias temperature instability, hot-carrier injection, and gate-oxide wearout will cause severe lifetime degradation in the performance and the reliability of future CMOS devices. The design guardband to counter these negative effects will be too expensive, largely due to the worst-case behavior induced by the uneven utilization of devices on the chip. To mitigate these effects over a chip's lifetime, this paper proposes Colt, a simple yet holistic scheme to balance the utilization of devices in a processor by equalizing the duty cycle ratio of circuits' internal nodes and the usage frequency of devices. Colt relies on alternating true- and complement-mode operations to equalize the duty cycle ratio of signals (thus the utilization of devices) in most data path and storage devices. Colt also employs a pseudo-random indexing scheme to balance the usage of entries in storage structures that often exhibit highly uneven utilization of entries. Finally, an operand-swapping scheme equalizes utilization of the left and right operand data paths. The proposed mechanisms impose trivial overhead in area, complexity, power, and performance, while recapturing 27% of aging-induced performance degradation and improving mean time to failure by an estimated 40%.**

Keywords-design; reliability; BTI; HCI; gate oxide wearout

I. INTRODUCTION

The reliability and performance of CMOS devices in future process technologies are projected to degrade substantially over their lifetime due to multiple device-level aging mechanisms. While not a new problem for CMOS devices, the aging has been modest and design-time guardband has sufficed for guaranteeing chip- and system-level performance over a component's expected lifetime in previous technology generations. In future process technologies, however, this will no longer be true, and both architecture- and circuit-level approaches for mitigating and tolerating the negative effects of device aging will be critical.

The primary CMOS aging mechanisms are *bias temperature instability* (BTI) [1] and *hot-carrier injection* (HCI) [2], both of which can affect device performance and lead to timing violations; as well as gate-oxide wearout [3] which can cause hard failures due to breakdown of device's gate oxide and result in short circuits. For example, BTI for PMOS (i.e. NBTI) can increase the threshold voltage of devices by more than 50mV [4][5], degrading the circuit speed by 20% or more under realistic operating conditions over a processor's expected lifetime. In other words, manufactured processors must be set to operate at 20% slower *maximum frequency* (F_{max}) than they can during manufacturing test to guard against subsequent timing failure. HCI also has a similar impact on device delay (thus F_{max} of processors) although the stress mechanism is slightly different from that of BTI. Furthermore, as a device's gate oxide becomes thinner, there is a much higher probability of oxide breakdown, leading to a hard failure of a device that exceeds its intended (or targeted) stress time.

To first order, device aging under these mechanisms is proportional to device stress time (i.e. the average duty cycle or on/off ratio) and switching frequency of the circuit's internal nodes. Thus, devices with highly biased duty cycle (i.e., close to 0 for PMOS devices) and slow switching frequency are stressed more heavily, suffering the consequences of aging more quickly and more severely. This effect is compounded by thermal feedback, since active devices located at die hot spots operate at an elevated temperature, which further accelerates device aging. Meanwhile, devices with a balanced duty cycle ratio (i.e., balanced device on/off-time) and reasonable switching frequency will maintain their nominal device characteristics for much longer due to associated recovery mechanisms during device off-time.

It is well known that the duty cycles of internal circuit nodes and the switching frequencies of elements in microarchitectural components vary broadly, even for otherwise homogeneous structures like cache arrays. Unfortunately, failure estimation and design-time guardbands have to be set based on the worst-case behavior, derived from the devices with the highest utilization. This leads to dramatic increase in estimated device failures and the corresponding guardband. For example, our detailed analysis of a modern 32-bit adder block predicts ~20% delay degradation due to NBTI, along with a 40% higher device failure probability due to gate-oxide wearout within the same *mean-time-to-failure* (MTTF) target. However, we expect that microarchitectural solutions that balance the utilization of devices throughout the entire processor would recapture a substantial fraction of the penalty associated with device aging.

This paper proposes Colt, a simple yet holistic approach to equalize the duty cycle ratio and the usage frequency of devices to balance the utilization of devices in a modern microprocessor. Colt employs three simple and low-cost approaches that collectively mitigate the detrimental effects of aging.

- **Complement Mode Execution:** Colt relies on the complementary structure of CMOS circuits, and specifies a modestly-revised control path, data path, and storage hierarchy (including tag arrays) capable of

processing both true and one's-complement forms of program state. True and complement forms are employed in alternate coarse-grained epochs in the control path, data path, and storage structures, leading to nearly-balanced duty cycles and utilization of the PMOS and NMOS devices in these circuits, hence minimizing the negative effects due to aging at the circuit level.

- **Cache Set Rotation:** Colt distributes usage frequencies of entries in a modern processor's caches through careful microarchitectural resource management using LFSR hashing.
- **Operand Identifier Swapping:** Colt equalizes the utilization of the left and right operand data paths.

These approaches guarantee more uniform, balanced usage distribution of references to all microarchitectural structures, and shift worst-case behavior closer to the average case, recapturing much of the aging penalty. Collectively, the techniques employed in Colt recapture 27% of the delay degradation and 40% of the reduction in MTTF of a baseline 32-bit adder design, and impose only a trivial cost in area, power, and per-cycle performance. Colt is the first comprehensive proposal to achieve these benefits with minimal effect on performance or power consumption.

II. RELATED WORK

As mentioned in Section I, recent trends in technology scaling have exacerbated reliability concerns such as BTI, HCI, and gate oxide wearout. Aging is driven by switching activity, so prior proposals for power savings, e.g. operand swapping [6], can provide benefits similar to Colt. Recently, microarchitectural solutions have been proposed to reduce or hide the negative effects of device aging. Kumar [7] analyzed the impact of NBTI on SRAM cells and proposed periodic cell flipping in SRAM structures, while Recovery Boosting changes the SRAM cell to periodically enable a recovery mode [8]. Abella et al. proposed Penelope [9], a collection of strategies to mitigate NBTI degradation in both combinational and storage blocks. Shin [10] proposed to proactively use sub-array redundancy in caches to allow the relaxation of non-faulty components. Each sub-array in a cache is periodically put into recovery mode by charging the gate and discharging the source, and using redundant sub-arrays to help migrate the data around. Colt obtains similar or better benefits with a simpler, lower overhead approach.

Other work [11]-[13] have also been proposed to reactively improve lifetime reliability by using microarchitectural redundancy. In [14] a dynamic reliability management technique that explores adaptive techniques to adjust processor temperature, current density, voltage, and/or frequency as needed at runtime was also presented. Mitigating NBTI in a chip multiprocessor environment has been proposed by Olay [15] and Facelift [16]. Olay uses low level sensors to monitor the condition of each core, and it assigns jobs in a manner that minimizes the impact on lifetime reliability. Facelift proposed a similar age-driven job scheduling technique to hide the aging. Additionally, adaptive supply voltage and adaptive body biasing can be applied at different stages of the chip lifetime to reduce the aging. DeBole [17] proposed New-Age, an aging assessment framework capable of examining arbitrary microarchitectural components and evaluating them for performance degradation. All of these techniques are orthogonal to Colt and could be used in conjunction with our scheme to obtain further benefits.

The prior proposals most similar to Colt are Kumar et al.'s periodic cache flipping [7] and Abella et al.'s Penelope [9]. While similar in that it periodically flips the cache contents, Kumar et al. require XOR gates to flip the cache contents back into normal mode upon leaving the cache. In contrast, Colt uses the flipped cache content as is without the need to flip them back. Penelope relies on the availability of idle pipelines, cache blocks, registers, ports to storage structures, and other resources, and inserts targeted values into such elements in order to mitigate uneven duty cycles. When idle resources are available less than 50% of the time and when the bits are highly biased, Penelope's benefit is not optimal. Moreover, dynamic power is consumed to exercise those resources with the targeted values and to do value sampling to update their RINV registers. Penelope also requires an increase of area to store special inputs as well as for RINV registers. In contrast, Colt does not rely on the availability of idle resources, nor does it impose more than a trivial power or performance overhead. Instead, duty cycles are naturally balanced throughout the microarchitecture even as useful computation is performed, with only minor overhead to manage infrequent mode changes and to perform complement conversions (where needed). In short, in contrast to Penelope, Colt provides robust resilience to BTI aging throughout the microarchitecture, independent of the availability of idle resources, and at very low cost in power or performance.

III. DEVICE STRESS AND AGING IN MICROARCHITECTURE COMPONENTS

A. *Device Aging Mechanisms in CMOS Technology*

In nanoscale CMOS devices, there are three major device aging mechanisms: 1) *bias temperature instability* (BTI), 2) gate-oxide wearout, and 3) hot carriers. First, negative (positive) bias to the gate of PMOS (NMOS) devices increases *threshold voltage* (V_{TH}) of the device over time, climbing by as much as 50mV in 45nm technology due to NBTI [1]. Further, it is expected that PBTI will become comparable to NBTI in near future CMOS technology. Second, the gate-oxide (i.e., insulator) wearout starts with the formation of defects (traps) in the oxide when the devices are on. As more traps are formed over time, the gate oxide of devices loses its property as insulator and it begins to conduct current. As a result, the devices do not function correctly. Third, hot carriers are electrons in the channel of devices which have energy high enough to overcome the potential barrier at the Si-SiO2 interface. These hot carriers traverse

through the gate oxide and constitute gate current. A small percentage of these carriers get trapped in the oxide and cause measurable shifts in V_{TH}. Both the degradation of devices due to BTI and gate-oxide wearout is proportional to their "on" time while that due to hot carriers is proportional to the number of device switching events.

B. *Case Studies for Data Path Components in Microprocessors*

It is well known that there is substantial non-uniformity in the utilization of various data-path components in microprocessors. This creates uneven stress (thus aging) among different components or even parts of components in microprocessors. In order to compensate for this uneven aging, either design or manufacturing-time guardbands are added to the cycle time of the control and data paths and the minimum operating voltage of memory structures, accommodating the worst-case aging of the components. Hence, it is also important to achieve more uniform aging among different parts in data-path components, minimizing the overall aging effects.

Adder-Subtractor: It is well-known that the adders in the integer execution unit often add small values for incrementing loop indices etc. For example, in a 32-bit adder, 8 or 16-bit small values only exercise lower 8- or 16-bit portion of the adder circuit for ADD instructions, while many circuit nodes in the upper 24-or 16-bit part of the adder are stuck at one value for a long time. This increases the stress time of either NMOS or PMOS devices in the circuit, causing substantially more BTI degradation or oxide wearout for the stressed devices than the ones experiencing more frequent circuit node switching. Figure 1 shows the average duty cycle ratio of each internal node in a 32-bit Kogge-Stone adder-subtractor. The input operands for ADD and SUB are collected from a 4-wide out-of-order machine running bzip2 for 100milion cycles (cf. Table I in Section V for more detailed simulation parameters) and they are applied to a custom switch-level simulator that models a 32-bit Kogge-Stone adder-subtractor to collect the average duty cycle of individual nodes.

As shown in Figure 1, we observed highly-biased duty cycle ratios in many internal nodes associated with high-order operand bits. For instance, if the average duty cycle ratio of a signal is far less (more) than 50%, PMOS (NMOS) will degrade more substantially. Unless the duty cycle ratio is not close to 50%, either NMOS or PMOS will degrade more substantially in terms of both BTI and gate-oxide wearout. Moreover, we also observed much longer average signal switching cycle time in the internal nodes related to high-order operand bits. This means that a signal stays high or low for a longer time with a biased duty cycle ratio, the device will be more impacted by aging since it becomes harder to recover from relatively longer-term degradation.

I-Cache and D-Cache: As shown in prior work, data caches often contain significantly more '0' than '1' [7][9]. This will stress devices in SRAM cells that have symmetric

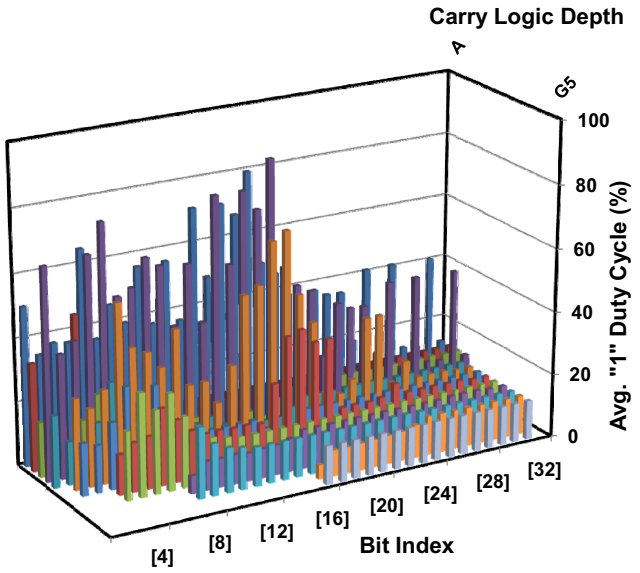

Figure 1. Average duty cycle ratio (%) of each internal node of a 32-bit Kogge-Stone adder-subtractor running bzip2.

circuit topology unevenly, exacerbating SRAM failures at low supply voltage (V_{DDMIN}) and limiting supply voltage scaling for power-efficient computing. The extreme bit bias in SRAM cells requires an additional SRAM V_{DDMIN} guardband of more than 50~100mV [9]. This is substantial, considering the nominal supply voltage (V_{DD}) is 1V and V_{DDMIN} is around 750mV; the amount of potential processor power reduction with extra 50~100mV lower supply voltage can be more than 13~25% of the total power at V_{DD} equal to 750mV. To mitigate this problem, periodic inversions of data cache contents has been proposed, reducing the guardband [7][9].

In addition to cache content bias, caches also show strong bias in access distribution. Some sets are an order of magnitude more frequently accessed than other sets due to temporal and spatial locality. This implies that the corresponding wordlines and their drivers have to switch far more frequently than others. This leads to more HCI-induced degradation, impacting SRAM array access time and device failure rates sooner than expected due to over-utilization. Meanwhile, the biased set access pattern also impacts BTI of the wordline drivers of rarely accessed sets; usually, two inverters form a wordline driver (or buffer) and the PMOS and NMOS devices of the first and the second stage inverters will be stressed for a long time with biased duty cycle ratio and low switching frequency.

Issue Queue, Bypass Network, and Register File: In general, most structures in the microprocessor are designed to accommodate two operand identifiers: 1) the operand identifiers are used to wake-up instructions in the issue queue, 2) they will then used to access register file entries, and 3) they are used to capture bypassed operands from the bypass network. The path for the first and the second operands is usually hard-wired from the issue queue to the

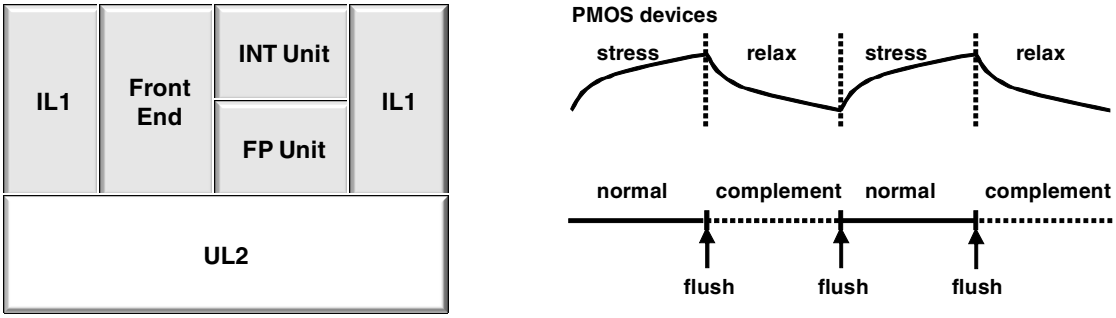

Figure 2. Colt complement architecture: the components in the shaded areas are complemented to relax PMOS devices periodically.

register file, to the bypass logic, and to the execution unit. All these structures are often in the critical path of the processor. Our data shows that the first operand identifier is used far more frequently than the second one since many instructions rely on immediate value or offset rather than register values, creating an unbalanced utilization between the first and second operand identifiers. Furthermore, considering that the high-order bits coming from adder (or ALU) are highly biased to "0" as shown in Figure 1, the buffers driving the high-order bits of forwarding values will experience more aging. As a result, uneven usage between the first and the second identifier paths in conjunction with biased high-order bits will require a substantial amount of aging guardbands, impacting the cycle time and the device failure rate of processors.

IV. COLT DUTY CYCLE EQUALIZER

In order to equalize the device duty cycle ratio and the usage frequency of devices throughout all the structures in a modern microprocessor, we propose Colt, a conceptually simple yet holistic approach that adopts three simple and low-cost techniques that collectively mitigate the detrimental effects of aging on device performance and MTTF. Colt revises the control path, data path, and storage hierarchy so they are capable of processing both true and complement forms of program state. True and complement forms are employed in alternate coarse-grained time slices in the data path. This leads to near-uniform duty cycle ratios and usage frequencies for the PMOS and NMOS devices in these CMOS circuits, hence minimizing the aging effects of these devices. Colt also distributes the utilization of entries across caches by time-sliced pseudo-random indexing the structures (caches, translation buffers). Last, Colt swaps left and right operands in alternating time slices to equalize utilization of data-path for the left and right operands [6].

A. Complement Mode Execution

As shown in Section III, very low signal switching frequencies as well as low duty cycle ratios are observed in the circuit nodes associated with high-order operand bits. Both will lead to more BTI and oxide wearout degradations on the devices connected to these signals. To mitigate this problem, we propose to complement the entire data path periodically to get a more balanced duty cycle over time. During the complement mode, all data in the execution core are in complement mode (one's complement). Thus, the problematic high-order operand bits that were originally biased towards '0' are now biased towards '1'. This complement mode provides relaxation (or recovery) periods for PMOS devices that process higher-order bits. Figure 2 shows the basic block diagram of the complement mode architecture. The blocks covered by the complement architecture are the front-end, integer execution, floating-point units, and L1 caches, as shown in shaded portion in Figure 2. All data crossing the shaded area boundary (between L1 and L2 caches) have to be transformed into the true or complementary form. Execution is divided into alternating epochs, normal and complement mode. Pipelines and caches are flushed between epochs. Since epochs of 1 to 1000 ms are sufficient, pipeline and cache flush have little effect on performance and power consumption. To assure that all computation results are correct while in complement mode, a number of issues to be addressed are discussed in the following sections.

L1 Caches: In complement mode, both tags and data are stored in complement form. Both instruction and data caches are flushed between epochs. The allocated instructions and data from L2 caches have to be complemented using XOR gates before being stored in both the IL1 and DL1 caches in complement mode. Fetched instruction words and load data are received in complemented form. Since it takes several cycles (e.g., 8 cycles in this study) for each L2 access, the added XOR gate delay in the data bus between L1 and L2 caches can be easily absorbed or hidden in the L2 cache data path. Hence, we expect that the L2 cache cycle time impact is negligible for supporting both the true and complementary modes.

Integer ALU: ALUs are the heart of a microprocessor, thus it is important to assure that all operations are done correctly when the data is in complement mode. Figure 3-(a) shows all the possible inputs for a full adder. The rows with the same fill pattern are complement to each other. As illustrated, each addition maintains the characteristic that the outputs (SUM and COUT) are also complemented when all the inputs (IN1, IN2, and CIN) are complemented. Note that this is valid for any N-bit adder (and carry generation logic).

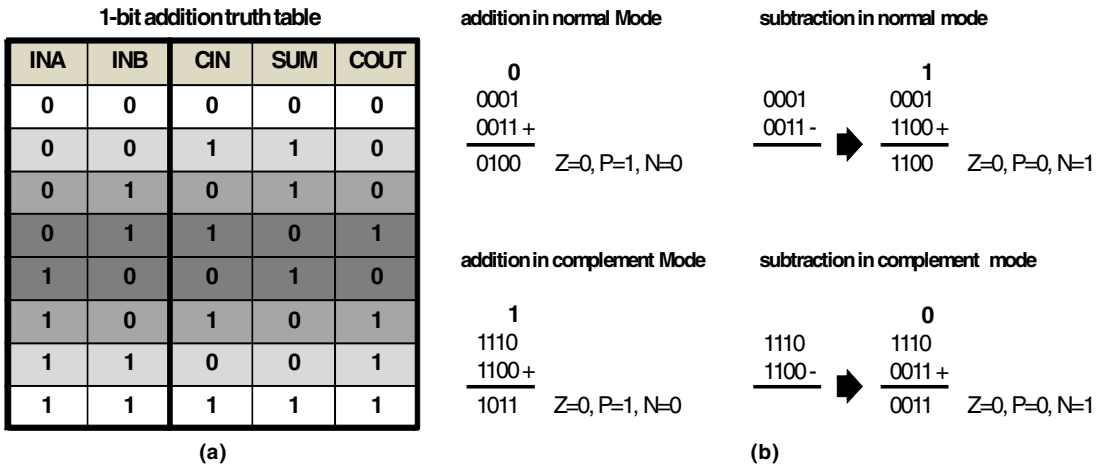

Figure 3. (a) 1-bit addition truth table. (b) Example of arithmetic operation in normal and complement mode.

Since complementing inputs always results in complementing the outputs, the correctness of an arithmetic operation is not affected with complemented inputs. However, we should produce correct flag values such as overflow ($COUT_N$ XOR $COUT_{N-1}$), negative (SUM_N = 1), and zero. Overflow detection using existing logic remains valid even if $COUT_N$ and $COUT_{N-1}$ are inverted ($\overline{COUT}_N$ XOR $\overline{COUT}_{N-1}$ = $COUT_N$ XOR $COUT_{N-1}$). The negative flag requires an additional XOR with the mode bit while the zero flag requires additional logic to detect all ones in complement mode. Figure 3-(b) shows an example of doing arithmetic operations in complement mode. All the inputs — IN1, IN2, and CIN have to be inverted in order to get the correct result in complement mode. As shown, the flags are always maintained in true form since they used in control path. They are also often used in branch instructions to determine the direction, thus must always be maintained in true form.

For AND and OR operation, the complement mode operation follows DeMorgan's Law where ~(A OR B) is equal to ~A AND ~B. In complement mode, the operation has to be changed from AND to OR and vice versa. Since the ALU already supports both AND and OR, only the ALU control signals need to be altered based on the mode bit in complement mode. For shift and rotate operations, the value specifying the shift/rotate amount must be in true form. Additional XOR gates are needed for inversion during complement mode. However, the shifted/rotated register value can be in true or complement form. The bits inserted for logical right shifts have to be 1 (rather than 0) in complement mode.

Figure 4 shows the average duty cycle ratio of each output node of all the major gates in a 32-bit Kogge-Stone adder with the proposed complemented architecture; the true and complementary-mode operands were applied during the first and the second half-million-cycles of the simulation, respectively. As shown in Figure 4, unlike the average duty cycle ratio presented in Figure 1 that shows low 10% duty cycle ratio for many output nodes related to high-order bits, the average duty cycle ratio of most output nodes approaches 50%, minimizing the delay degradation as well as the oxide wearout of the devices in the adder; as we use a longer period for inverting input operands, the average duty cycle ratio approaches 50% more closely for more nodes. Applying complement mode also creates a more balanced average signal switching time, compared to the original long signal switching time that implies that they are stuck at a particular value for a long time period. It is important to understand that the critical path of the adder is formed by nodes from index 1 to 32. Although there are some nodes that do not have 50% duty cycle, most nodes in the critical path have 50% duty cycle. Thus the overall guardbands to apply will be much

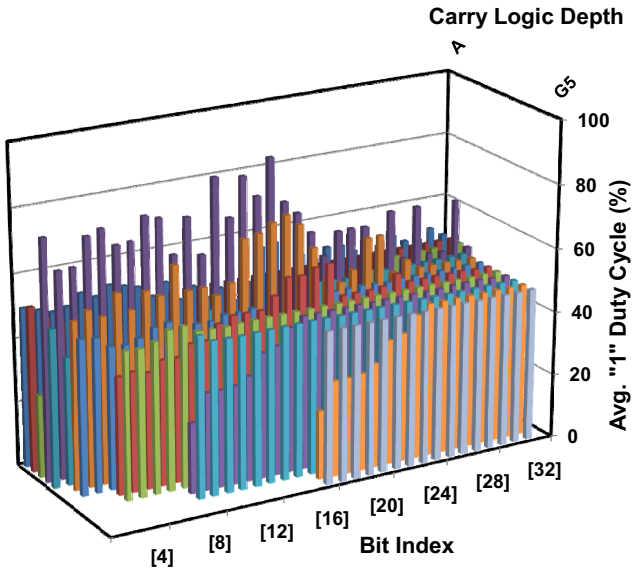

Figure 4. Average duty cycle ratio (%) of each internal node of a 32-bit Kogge-Stone adder-subtractor running bzip2 after the normal and complementary-mode operands were applied during the first and the second half-million-cycle simulations.

lower since only a few nodes on the critical path slow down due to NBTI aging.

Integer Multiplication and Division: Combinational integer multipliers perform two operations: partial product generation and summing of the partial products. The latter is usually realized with a Wallace Tree of carry-save adders, and will work correctly using complemented inputs, following the reasoning employed for integer addition. Partial products are generated as the logical AND operation of all pairs of input bits. Partial products of complemented inputs will have to be generated with an OR operation instead. Since the multiplier area is dominated by the summing logic, the area overhead due to the duplicated OR partial product generators should be minimal. Iterative multipliers rely on sequences of additions and subtractions, and will work with minor changes to the control logic. The analysis of high-performance dividers is beyond the scope of this paper, but iterative dividers rely on sequences of subtractions and comparisons, which will work with complemented operands, requiring only minor changes to the control logic.

Floating-Point Multiplication and Addition: Floating-point multiplication consists of exponent addition with a correction for the bias, which will work with complemented inputs but will require a complemented bias for the correction; significand multiplication which works with the modifications described above for integer multiplication; and normalization and rounding. The last two steps will require minor changes to the control logic, including proper generation and interpretation of the rounding bits. A floating-point addition consists of subtracting exponents to determine the shift amount required for radix alignment, which will work based on the reasoning above for integer arithmetic; radix alignment, which will require modifications to the barrel shifter control inputs to account for complemented outputs from the exponent subtraction; significand addition, which will work without modification; and normalization and rounding, which again will require minor changes to control logic and generation and interpretation of the rounding bits. Both types of floating-point operations also require sign bit generation, support for various rounding modes and special representations (NaN, Inf), as well as generation of flags and exceptions.

Front-End Structures: The program counter is maintained in complement form during complement mode. Instructions are fetched from the IL1 cache in complement form. Branch target computations are done in complement form since both the program counter and the offset fetched from the IL1 are in complement form. Only the OPCODE bits of the instructions need to be inverted back into true form upon entering the decoder. Thus, normalizing XOR logic needs to be added for the OPCODE bits upon entering the decoder as shown in Figure 5. In complement mode, the rename table is also maintained in complement form. This means that e.g. R0 is now located in the location of R31. The contents of the rename table are swapped between

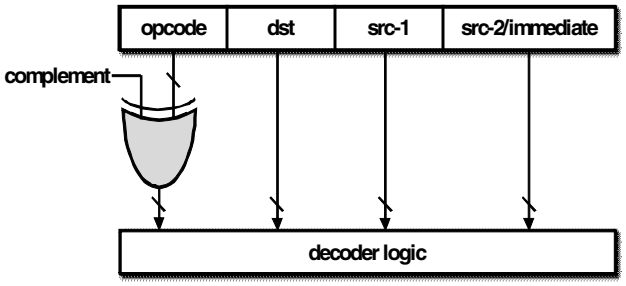

Figure 5. Decoder to handle instructions in complement mode.

complementary registers between modes. Maintaining the rename table in complement form helps to create a more uniform access distribution for rename table. Some registers tend to be used more than others creating a non-uniform access distribution of the rename table. The branch predictor is maintained in normal mode, requiring XOR gates to invert addresses upon accessing the branch predictor. Including the branch predictor in the complement architecture is left for future work.

Execution Core: In addition to the ALUs discussed earlier, the execution core consists of the issue queue, physical register file, payload RAM, load/store queue, and reorder buffer. The issue queue contains physical register file identifiers of input and output operands, valid bits, and status bits. With FIFO management of the physical register free list, we expect a fairly uniform distribution of register specifiers. Valid and ready bits are inputs to control logic and cannot be easily complemented, so are left unchanged. Physical register file contents are complemented during complement mode, and contents of the physical registers with valid mappings have to be complemented between epochs. The physical register identifiers can be kept in true mode since they are allocated in FIFO order. Immediate values in the payload RAM must be stored in complement form during complement epochs; since they are typically derived from the complemented instruction words, the only complication is to properly zero-extend (one-extend in complement mode) or sign-extend the values, depending on the instruction's semantics. The contents of the load/store queue and reorder buffer are in true and complement form in alternating time slices. The load/store queue identifiers can also be kept in true mode since they are allocated in circular manner.

In summary, virtually all data-path storage and computational elements operate on complement values in alternating epochs, providing uniform duty cycles and mitigating the effects of aging.

B. Cache Set Rotation

Cache accesses are often badly skewed toward a small number of cache sets. This non-uniform access distribution causes uneven aging across cache lines. To create a more uniform cache access distribution, we propose altering the cache index function for every epoch to spread the 'hot' set around. We use an LFSR to seed the hash index function as shown in Figure 6. Every change of epoch, the L1 caches are

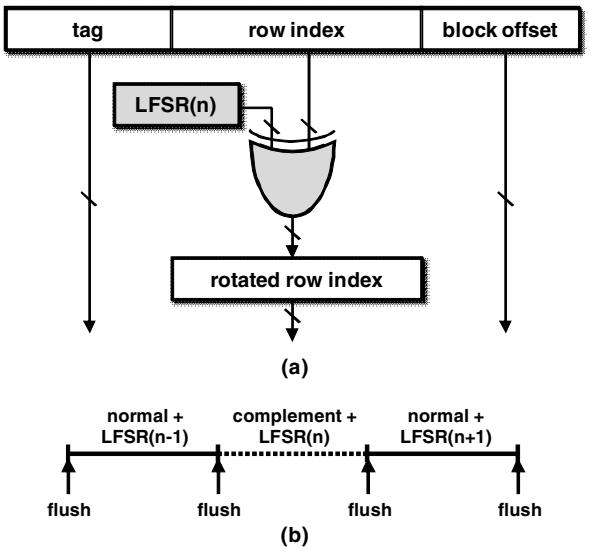

Figure 6. (a) Colt cache set rotation logic. (b) Colt operations over epochs with cache set rotation.

flushed, and the LFSR is updated to create a new hashing function. This hashing function is then XOR-ed with the original index bits to create new index bits. With a well-designed LFSR, the variation in index spreads out accesses across all the cache sets. Within a set, references are distributed across lines by initializing the head of the LRU list to a different entry at the beginning of each epoch. Since the pre-decoding to determine the bank(s) to access is often performed before the set (or row) address decoding, the delay due to the XOR gate shown can be hidden without impacting the overall delay of L1 caches. Note that we apply this set rotation technique to the L1 data and instruction caches only, but it could also be used for translation buffers (TLBs and segment registers) as well as the rename table.

C. Operand Identifier Swapping

Due to unary operations and immediate operands, the left operand is much more active than the right operand, creating an uneven aging between the two paths. Thus, Colt swaps left and right operands in alternating epochs to create a more balanced utilization of the two operand data paths. Commutative logical and arithmetic operations can safely swap operands, but non-commutative operations (e.g. shift, rotate, and divide) will not perform the swap. To make the adder/subtractor fully commutative, the left operand must be made invertible via XOR gates. The baseline adder/subtractor already has XOR gates for the right operand for subtraction (A-B), creating a pre-existing imbalance in the delay path. Thus, adding XOR gates for the left side will not increase the delay. Additional logic is shown as shaded in Figure 7. Swapping left and right operands has a beneficial effect on uneven utilization in the issue queue's tag matching logic, the physical register file read access path, the bypass network, as well as any pipeline registers along the way. For non-commutative operations, two-to-one muxes are embedded in the functional units to swap the source operands back to their true position during 'swap' epochs. Since the shifter and divider are not in the critical path, additional muxes should not affect cycle time.

D. Coverage

As shown in Figure 2, our coverage includes L1 caches, front-end as well as integer and floating-point unit. The L2 cache is not included in our coverage as it runs cooler than the main execution core, most subarrays are in low-voltage sleep mode, and it is much less vulnerable to aging. Moreover, the L2 cache is not in the critical path, so relying on a small amount of V_{DDMIN} and cycle-time guardbands will be sufficient to combat any aging effect. Within the execution core, the branch predictor PHT (which is inherently fault-tolerant) is not covered by Colt. While flushing the BTB and RAS can easily be done on every time epoch, as we do for the L1 caches, the branch direction predictor cannot be flushed easily because it takes a long time to train a branch predictor. Thus flushing it every epoch will result in poor predictions.

Parts of the control path are also not covered. The OPCODE and various flags and status bits are always represented in true form anywhere in the pipeline. However, the control path is generally not on the critical path and represents a small percentage of the core area. Thus, guardbanding and/or increasing the size of control path circuitry to mitigate aging can be employed. While logical register identifiers are inverted, physical register ones are not. This is because the physical registers do not suffer from access bias, as long as they are allocated in a circular manner. Similarly, ROB and load/store queue entries are allocated in a circular manner. Hence, each entry has an equal percentage of utilization. The contents of the ROB, physical register file, and issue queue are covered by our technique.

Colt also operates during idle periods. Even if the timer

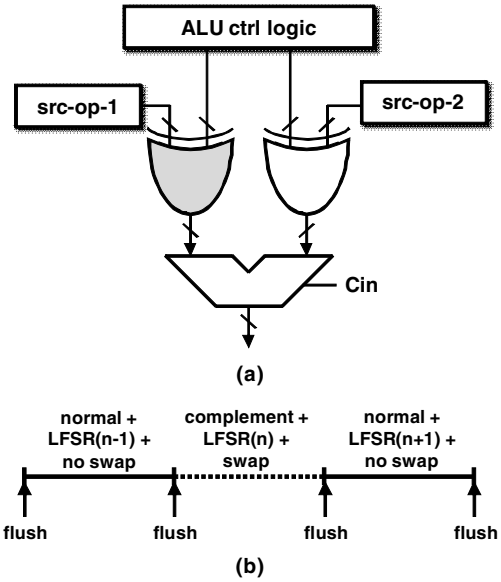

Figure 7. (a) Colt operand identifier swapping logic. (b) Colt operations over epochs with operand identifier swapping.

interrupt is the only code that runs, the pipelines are flushed and any values in all pipeline latches are complemented on every change of phase. During idle periods, Colt creates a virtually perfect balanced duty cycle with minimal power impact.

Lastly, Colt is necessary for both single-threaded and multi-threaded environment, such as SMT. The bias toward small value operands are found across benchmarks. Thus, although two threads from different programs are running together, it is likely that the high significant bits are still biased toward zeroes. In multi-processor environment, Colt can also be employed easily without much added complexity. Since Colt techniques are confined within core and L1 caches, it should not complicate communication between cores.

E. Overhead

Cycle Time and IPC: Although Colt requires XOR gates to be added in several paths, the latency of these gates is mostly hidden. The ones added in the adder/subtractor do not add any delay, as they simply balance the existing XORs in the baseline design. Similarly, XOR gates added to hash row index bits can also evaluate in parallel with the subarray/subbanking computation, so the delay is hidden. While there are XOR gates with exposed latency, those are not on the critical path, such as the ones added in the decode stage to invert the opcode, and the one added between L1-L2 caches. Since the inversion is only required for the OPCODE, we expect that it will not impact logic delay. In the worst case, XOR delays can be compensated by redesign and device resizing that will impose a slight power penalty.

Colt requires pipeline and local cache flushes every epoch change, thus possibly impacting the IPC of the program. However, epoch transition is done at second to millisecond granularity, diluting the IPC impact of the flush significantly. Our results in Section B show that there is at most 1% IPC impact with 1 ms epoch transition, and virtually no IPC impact at more realistic epoch granularities. These results match intuition, as the epoch transition overhead is amortized over millions (1ms epochs) to billions (1s epochs) of committed instructions at GHz clock rates.

Power: The average power (thus thermal) impact of Colt is minimal since the overall switching activities remain almost the same. Colt only inverts the polarity of signals; overall signal switching activities do not change significantly. However, we suffer a slight power increase due to mode transitions since we flush the pipelines and flash-invalidate write-through level-1 caches every epoch (once amortized over millions to billions of instructions, this is not significant).

Area: Colt requires the addition of LFSR generator and XOR gates to be added in various paths. However, it does not add any major structures to the core. Thus the area impact of COLT is minimal.

F. Relationship to Prior Work

While it is difficult to do quantitative comparisons with prior work, we would like to qualitatively compare our technique with relevant work. Prior work focused on scheduling and adaptive voltage control [15][16] and Colt are orthogonal to each other. While focusing on slowing down aging on the core as a whole, scheduling and adaptive voltage control do not stop the imbalanced aging of different structures within a core. Thus combining Colt with such techniques can maximize the benefits.

Among the related work discussed in Section II, Penelope [9] is the most similar to our work since it focuses on a single core and provides broad coverage for many structures. Penelope uses special values inserted during idle periods to un-stress PMOS transistors. One major difference is that Penelope relies entirely on the availability of idle periods to function optimally. Besides the overhead that the technique entails, relying on idle periods might not be the most reliable approach. While they show that idle periods for some structure are close to 50% while actively running benchmarks, in real environments CPUs rarely run at such levels of utilization, and are often mostly idle. For Penelope to work effectively under varying utilization levels, it must carefully rate-match the insertion of targeted values with the actual work being performed. Furthermore, the targeted values must be carefully chosen to match all possible workloads, which is not a trivial design task. In contrast, Colt works equally well under any utilization, and since no special target vectors are needed, substantially reduces the burden on the designer.

Coverage Comparison: While Penelope covers major structures in the execution core, it does not cover latches and front-end structures. On the other hand, by inverting all values, Colt covers the front-end and latches, as well as major structures. Unlike Colt, Penelope also does not handle uneven aging across different sets in memory structures and uneven datapath utilization.

Overhead Comparison: Although Penelope does not severely impact the peak or thermal design power, it requires significant energy overhead on affected structures. For example, in order to un-stress transistors in ALUs, it has to issue special values during idle slots. Similarly, it has to read, invert, and re-write entries in various storage structures. It also requires that there are significant idle cycles while running a program to insert those values. While minimally increasing peak power consumption, Penelope increases the activity counts for each of the affected structures, thus increasing the total energy consumed when running a program. In contrast, Colt does not increase the activity counts, and thus does not have a significant impact on the energy consumption. While Colt imposes energy overhead for refilling caches at each epoch transition, the epochs are rare enough that the overhead is not significant once amortized over the millions to billions of instructions executed in each epoch. In contrast, Penelope invalidates and inverts roughly 50% of the cache at any time. Again, the inversion process increases activity. Additionally, there will be L2 misses caused by the cache inversion/invalidation.

Finally, Penelope requires area overhead to store special target values as well as RINV sampling registers in various structures. On the other hand, Colt requires minimal area overhead for XOR gates in various datapath locations.

V. METHODOLOGY

NBTI, HCI degradation, and gate-oxide wearout probability are estimated by obtaining duty cycle and average switching cycle times through microarchitectural simulations of real programs. These also provide input streams for a functional block and/or logic simulations to obtain internal signal statistics of the functional block. For device and circuit-level analysis, we used a 32nm predictive technology model [18] and HSPICE. To obtain long-term NBTI degradations, we used the analytical models presented in [19] where the major circuit and application-specific parameters are 1) device stress time in seconds, 2) average switching cycle time of a signal applied to devices, and 3) signal duty cycle ratio representing signal stress time ratio during average signal switching cycle time. To model the effects of aging on complex datapath logic, we performed a detailed circuit-level study of a modern Kogge-Stone adder and a simpler datapath structures modeled using a representative 32F04-delay combinational circuit. We omit detailed results on SRAM structures, since our findings closely match related prior work [7].

For microarchitectural study, we use a modified *Simplescalar / Alpha 3.0* tool set for the Alpha AXP ISA[20]. The machine configuration used is shown in Table I. A frequency of 2GHz is assumed. The SPEC2000 INT and FP benchmark suites are used for case studies and results. All benchmarks were compiled with the DEC C and C++ compilers under the OSF/1 V4.0 operating system using -O4 optimization. SPEC2000 FP benchmarks that are in Fortran are omitted since they are incompatible with our simulator. However we do not expect that the result will differ much. Reference input sets, fast-forwarded to the first Simpoint [21][22] with cache and branch predictor warm-up, and a detailed run of 200M are used. Detailed runs of 2B cycles after fast-forwarding are used for the IPC impact study.

TABLE I. Machine Configuration

Execution Pipeline	4-wide fetch/issue/commit, 128 ROB, 96 PRF, 32, 24SQ, 32-entry scheduler, 11-stage pipeline
Branch Prediction	Combined bimodal (16k) / gshare (16k) with a selector (16k), 16-entry RAS, 4-way 1k-entry BTB
Functional Units	2 integer ALU (1-cycle), 2 integer mult/div (3/20-cycle), 2 general memory ports (1+3)
Memory System	L1 I-Cache: 64KB, direct-mapped, 64B line size (2-cycle) L1 D-Cache: 64KB, 4-way, 64B line size (2-cycle), write-through policy L2 Unified: 2MB, 8-way, 128Bline size (8-cycle) Off-chip memory: 350-cycle latency

VI. RESULTS

A. Circuit Level Analysis

To model the effects of aging on complex datapath logic, we performed a detailed circuit-level study of a modern Kogge-Stone adder. Figure 8-(a) shows the average of stress duty cycle, which is when the input is '0' for each bit of the adder. The data is averaged over all benchmarks in SPEC 2000 benchmark suite. On average, the conventional machine has a stress duty cycle of 71% on high index bits. Colt is able to reduce the stress duty cycle to roughly 51%, very close to the ideal stress duty cycle of 50%.

Figure 8-(b) shows the average device stress level that is proportional to gate-oxide failure probability in for SPEC 2000 benchmark suite. The result presented in Figure 8-(b) shows the average of relative stress levels of all the internal nodes in the 32-bit Kogge-Stone adder-subtractor for each benchmark program; a higher stress level of a node is

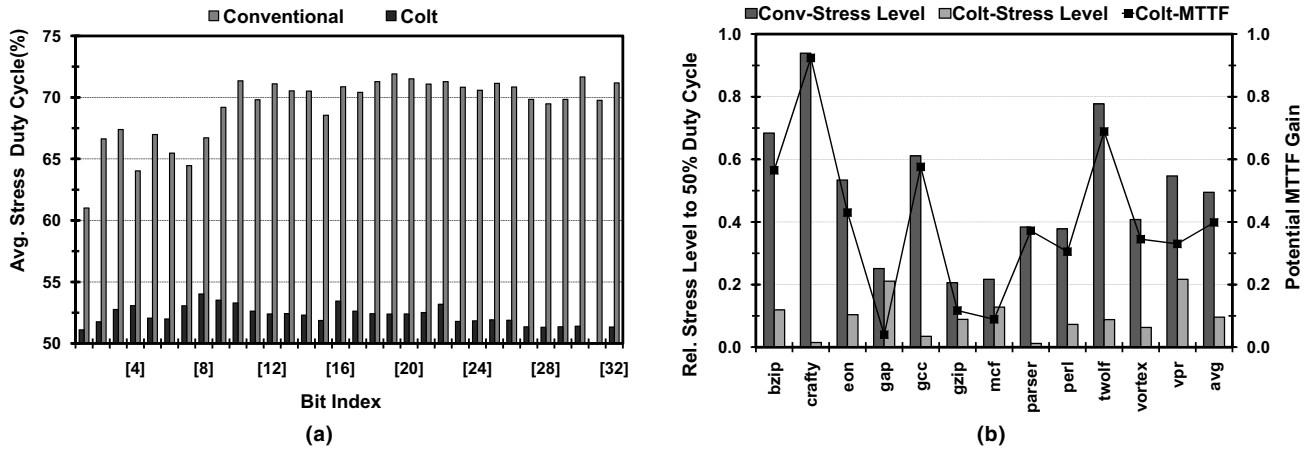

Figure 8. 32-bit Kogge-Stone adder-subtractor: (a) average stress duty cycle and (b) stress level and potential MTTF improvement. Comapring stress level between Colt and conventional machine. Second y axis in (b) shows percentage MTTF improvement normalized to conventional machine.

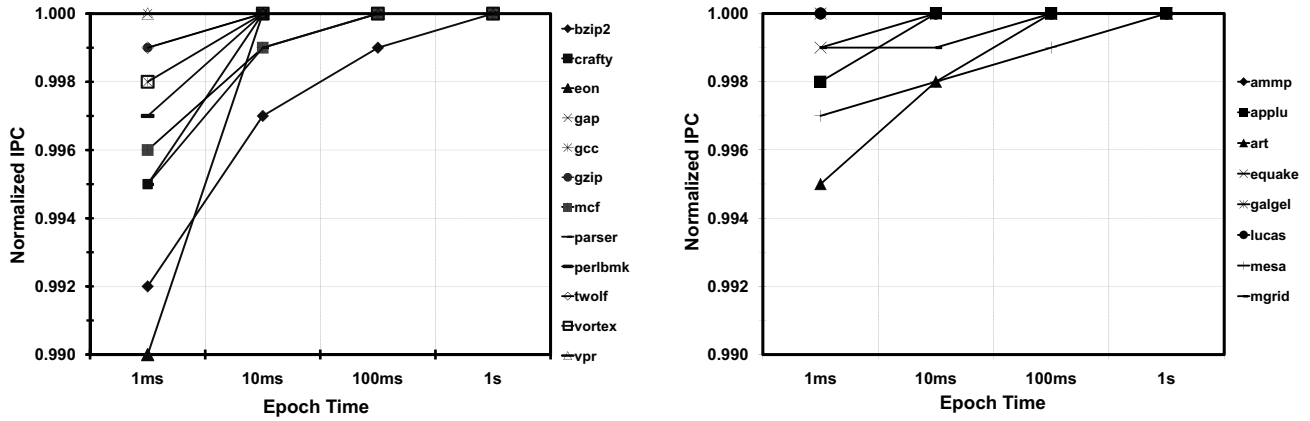

Figure 10. IPC impact of Colt: (a) SPEC2K INT (a) and (b) SPEC2K FP.

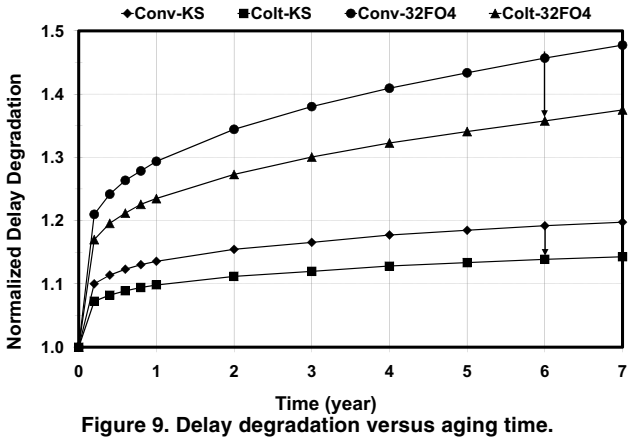

Figure 9. Delay degradation versus aging time.

resulted from a highly-biased duty cycle ratio of the signal; either NMOS or PMOS device failure probability increases, as the signal duty cycle ratio becomes highly biased to either 1 or 0 (or far from the 50% duty cycle ratio). The gate-oxide failure probability (or time to failure) has been shown to be directly proportional to device stress time [3], so the average relative stress level of all the internal nodes can be interpreted as relative MTTF as shown in the secondary y-axis of Figure 8-(b). We can improve the relative MTTF by ~40% on average. However, more recent studies show that the gate-oxide failure probability increases superlinearly in proportion to device stress time in nanoscale technologies with ultra-thin gate oxide [23]. Thus, our projection for the MTTF shown Figure 8-(b) is very conservative and more substantial improvement for MTTF is expected.

Figure 9 shows delay degradation versus aging time in a Kogge-Stone adder-subtractor and 32 FO4 combinational circuit. To model the NBTI degradation of PMOS devices, we used the model presented in [1] for a 32nm predictive technology model [18]; the average duty cycle ratio and signal switching cycle time of the entire SPEC 2000 benchmark suite were applied to the NBTI model. If we balance each signal's duty cycle ratio close to 50% using our proposed Colt architecture, we can reduce F_{max} guardband for Kogge-Stone adder-subtractor by 27% as shown in Figure 9-(a).

Figure 9-(b) shows delay degradation of a 32 FO4 combinational circuit. The circuit modeled is composed of inverters, NAND gates, and NOR gates, and is representative of other datapath components in the microprocessor (e.g., bypass networks, wordline drivers, etc.). Figure 9-(b) shows Colt achieves 28% F_{max} guardband reduction, similar to the result of adder-subtractor in Figure 9-(a). Lastly, we found that enforcing periodic data flipping can recapture more than 30% of V_{DDMIN} degradation [7] for SRAM memory elements.

B. Microarchitectural Analysis

Figure 10 shows the IPC impact of Colt using varying epoch intervals. Assuming a 2GHz frequency, a 1ms epoch time translates to 2M cycles. Beyond flash-invalidating the L1 caches, we also impose a 100 cycle penalty at each epoch transition. This penalty is for flushing the pipeline, calculating a new LFSR value, moving the rename table mapping and complementing architected register values. As seen in Figure 10, the worst-case IPC impact is less than 1% for the smallest epoch time, 1ms. There is virtually no IPC impact as we increase the epoch time to 100ms. Note that the aging process occurs slowly on a much larger time scale than the epoch intervals shown in Figure 10.

Figure 11 shows L1 data cache before and after cache set rotation is applied, including per-benchmark access patterns and an approximation of time sharing, with all benchmarks run consecutively. An ideal 0.4% access distribution is shown in both figures; the line represents the ideal case where all sets are accessed uniformly. Figure 11-(a) shows the original cache set access distribution where some sets are never touched, thus having a zero percentage of accesses while the hot sets can have as high as 20-30% of accesses. Even with time sharing, the lowest percentage of access is 0.15% while the highest percentage of access is 2%. Figure 11-(b) presents the distribution after applying cache

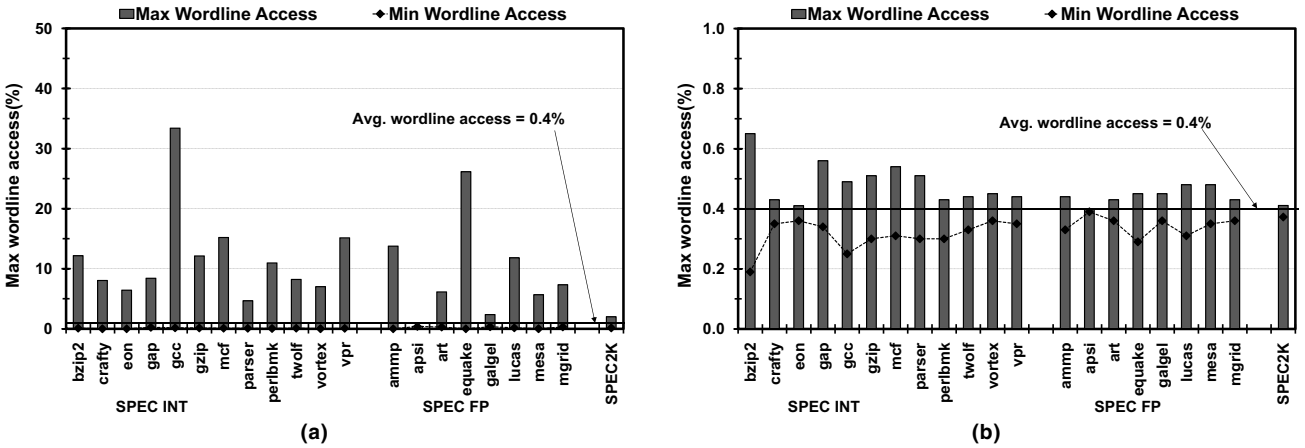

Figure 11. DL1 cache set access distribution: (a) before and (b) after Colt is applied.

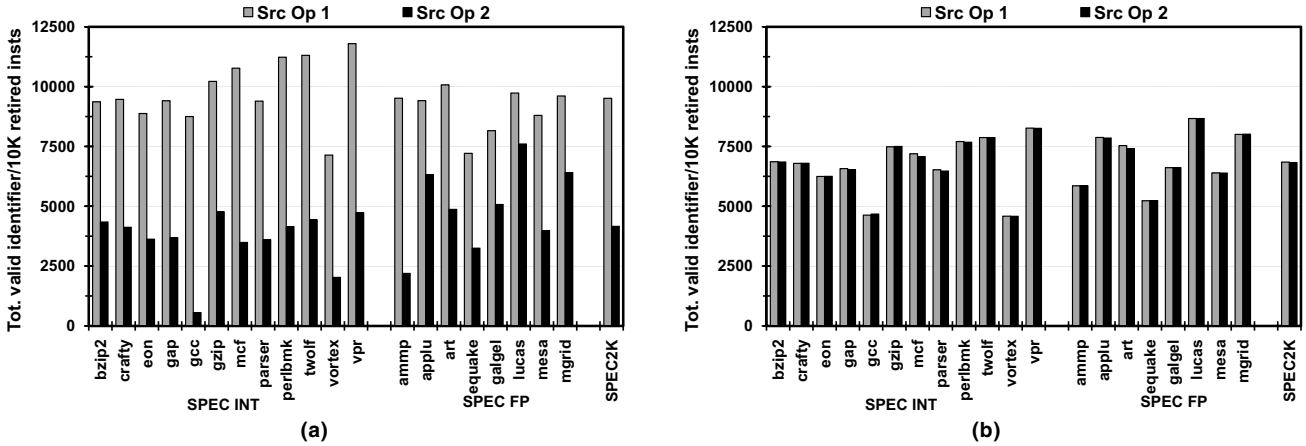

Figure 12. Total valid operand identifier reads per 10,000 retired instructions: (a) before and (b) after Colt is applied.

set rotation. Figure 11 shows that cache set rotation succeeds in bringing the highly accessed rarely accessed sets to have a distribution close to the ideal average distribution of 0.4%, implying a uniform aging across sets. The L1 instruction cache has similar behavior.

Figure 12 shows the count of valid left (Operand 1) and right (operand 2) (a) before and (b) after applying operand identifier swapping. As seen in Figure 12-(b), both operand data paths are now equally used compared to the original state shown in Figure 12-(a) where one operand path is used twice or even three times more often than the other operand path.

In summary, these results demonstrate that the proposed Colt techniques — complement mode execution, cache set rotation, and operand swapping--are quite effective at equalizing device duty cycles across these critical microprocessor components. Extrapolating from these results, it is clear that a comprehensive and holistic approach like Colt will mitigate the effects of BTI, HCI, and gate oxide wearout throughout all components of a microprocessor core.

VII. CONCLUSION AND FUTURE WORK

The performance and reliability effects of aging on future CMOS devices are substantial, and can no longer be addressed solely with design-time guardbands. Our detailed analysis of a modern microprocessor shows that there exists a significant amount of unbalance in device utilizations in virtually every data path component and its sub-components, requiring unacceptably high guardbands. For example, 1) the frequency of a 32-bit adder in the integer execution unit is expected to degrade by 20% over a ten-year lifetime while failure probability increases by an estimated 40%; 2) certain sets in IL1 and DL1 caches are accessed 70 times more frequently than the average one, accelerating the wearout of the associated structures like wordline drivers, and their metal lines. To combat this problem, we propose Colt, a comprehensive approach to balance the utilization of devices in a processor by equalizing the duty cycle ratio of circuit's internal nodes and the usage frequency of devices. This allows the affected devices to recover and distributes stress across different devices.

Colt consists of three techniques: 1) complement-mode execution, which uses true- and complement-mode representations in alternating epochs to smooth out uneven distributions of values in the processor's data-path and storage structures; 2) cache set rotation, which distributes

uneven access patterns to architected storage structures so that all elements are stressed uniformly; and 3) operand swapping, which balances the stress on left and right operand delivery lanes in the execution core. These techniques impose negligible overhead in terms of area, complexity, and power, but are able to recapture 27% of aging-induced frequency degradation for various representative data path circuits, while delivering an estimated 40% improvement in reliability for our case study for a 32-bit adder circuit. In future work, we plan to examine the circuit-level effects of aging in additional components, focusing initially on front-end structures like branch predictors, as well as elements of the memory hierarchy beyond L1 caches.

ACKNOWLEDGEMENT

We want to thank Kewal Saluja for his contributions in the early phases of this work, as well as the anonymous reviewers for their helpful comments. This work was supported in part by an IBM Fellowship, a gift from Microsoft Research, and National Science Foundation awards (CCF-0702272 and CCF-0953603).

REFERENCES

[1] W. Wang et al. The impact of NBTI on the performance of combinational and sequential circuits. *In Proc. Design Automation Conf.*, 2007.

[2] E. Rosenbaum et al. Effect of hot-carrier injection on n- and pMOSFET gate oxide integrity. *IEEE Electron Device Letters*, 12:(11), Nov. 1991.

[3] E. Minami et al. Circuit-level simulation of TDDB failure in digital CMOS circuit. *IEEE Trans. on Semiconductor Manufacturing*, 8:(3), Aug. 1995.

[4] S. Borkar. Electronics beyond nano-scale CMOS. *In Proc. Design Automation Conf.*, 2006.

[5] D. Schroder and J Babcock. Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *Journal of Applied Physics*, 94(1), Jul. 2003.

[6] W. Li and L. Wanhammar. Low power design for data dependence. *In Proc. Swedish System-on-Chip Conf.*, 2003.

[7] S. Kumar et al. Impact of NBTI on SRAM read stability and design for reliability. *In Proc. Int. Sym. on Quality Electronic Design*, 2006.

[8] T. Siddiqua and S. Gurumurthi. Recovery boosting: A technique to enhance NBTI recovery in SRAM arrays. *In Proc. Computer Society Ann. Symp. on VLSI*, 2010.

[9] J. Abella et al. Penelope: The NBTI-aware processor. *In Proc. of Int. Symp. on Microarch.*, 2007.

[10] J. Shin et al. A proactive wearout recovery for exploiting microarchitectural redundancy to extend cache SRAM lifetime. *In Proc. Int. Symp. on Comp. Arch.*, 2008.

[11] F. Bower et.al. Tolerating hard faults in microprocessor array structures. *In Proc. Int.Conf. on Dependable Systems and Networks*, 2004.

[12] J. Srinivasan et.al. Exploiting structural duplication for lifetime reliability enhancement. *In Proc. Int. Symp. on Com. Arch.*, 2005.

[13] M. Lucente et al. Memory system reliability improcement through associative cache redundancy. *IEEE J. Solid-State Circuits*, 26:(3), Mar. 1991.

[14] J. Srinivasan et al. Lifetime reliability: Toward an architectural solution. *IEEE Micro*, 25:(3), May-Jun. 2005.

[15] S. Feng et. al. Olay: Combat the signs of aging with introspectice reliability management. *In Proc. Workshop on Quality Aware Design*, 2008.

[16] A.Tiwari and J.Torrellas. Facelift: Hiding and slowing down aging in multicores. *In Proc. Int. Symp. on Microarch.*, 2008.

[17] M. DeBole et al. A framework for estimating NBTI degradation of microarchitectural components. *In Proc. Asia & South Pacific Design Automation Conf.,* 2009.

[18] http://www.eas.asu.edu/~ptm.

[19] S. Bhardwaj et al. Predictive modeling of the NBTI effect for reliable design. *In Proc. Custom Integrated Circuit Conf.*, 2007.

[20] D. Burger and T. Austin. The Simplescalar toolset. *Technical Report TR-97-1342*, Univ. of Wisconsin-Madison, 1997.

[21] T. Sherwood et al. Basic block distribution analysis to find periodic behavior and simulation points in applications. *In Proc. Parallel Arch. and Compilation Ttechniques*, 2001.

[22] T. Sherwood et al. Automatically characterizing large scale program behavior. *In Proc. Arch. Support for Programming Lang. and Operating Systems*, 2002.

[23] J. Chen et al. Thin oxide breakdown mechanism of constant voltage stress on MOSFET. *Physica Scripta, T101, 10-13*, 2002.