The degradation of CMOS devices over the lifetime can cause severe threat to the system performance and reliability at deep submicron semiconductor technologies. The negative bias temperature instability (NBTI) is among the most important sources of the aging mechanisms. Applying the traditional guardbanding technique to address the decreased speed of devices is too costly. On-chip memory structures, such as register files and on-chip caches, suffer a very high NBTI stress. In this paper, we propose the aging-aware design to combat the NBTI-induced aging in integer register files, data caches and instruction caches in high-performance microprocessors. The proposed aging-aware design can mitigate the negative aging effects by balancing the duty cycle ratio of the internal bits in on-chip memory structures. Besides the aging problem, the power consumption is also one of the most prominent issues in microprocessor design. Therefore, we further propose to apply the low power schemes to different memory structures under aging-aware design. The proposed low power aging-aware design can also achieve a significant power reduction, which will further reduce the temperature and NBTI degradation of the on-chip memory structures. Our experimental results show that our aging-aware design can effectively reduce the NBTI stress with 30.8%, 64.5% and 72.0% power saving for the integer register file, data cache and instruction cache, respectively.

Keywords: Negative bias temperature instability; register file; cache.

1. Introduction

With continuous scaling of the semiconductor technology, the degradation of the performance and reliability of the CMOS devices over the lifetime due to aging mechanisms has become a major concern. The increased current density and temperature in future devices will further accelerate the degradation. Bias temperature instability (BTI), hot-carrier injection and gate-oxide wearout are the primary aging
mechanisms for CMOS devices.\textsuperscript{2–4} The negative bias temperature instability (NBTI) for pMOS devices are one of the most prominent and persistent threats for future technologies. NBTI will cause an increase in the threshold voltage ($V_{th}$) of the pMOS devices when negative voltage is applied at the gate (logic “0”). The threshold voltage can be increased by 50 mV, which may result in a degradation of the circuit speed by 20\% or cause functional failure during the expected lifetime.\textsuperscript{5–8}

Besides the NBTI, the time-dependent dielectric breakdown (TDDB), electromigration (EM), stress migration (SM) and thermal cycling (TC) are among the major reasons of permanent failures and short lifetime in integrated circuit (IC).\textsuperscript{9,10} TDDB is caused by the trapping of the charges in the oxide and the consequent charge flows that break down the gate oxide. EM is caused by the high current densities that form the voids in the metal line or hillocks inducing short circuits. SM is caused by the mechanical stress gradients that form voids in IC metallization and TC caused damage is mainly due to the uneven heating and cooling in the system that might be induced by aggressive power management. Among these, the NBTI-induced aging is one of the most dominant failure mechanisms in the chips.\textsuperscript{10} Therefore, we propose to target at mitigating the NBTI-induced wearout of the on-chip memory structures in this work.

The conventional methodology to address the decreased speed of devices due to NBTI is guardbanding. The guardbanding is a technique where the operating frequency is reduced in order to overcome the degradation that may be incurred over the lifetime of the devices. For example, a large guardband of 20\% in cycle time may be required, given that the circuit speed may be reduced by 20\% due to NBTI. The conventional guardbanding technique is too expensive because of the worst-case behavior caused by the uneven utilization of different devices on the chip. Moreover, in future technologies, the guardbanding technique may not be suitable to guarantee the performance and reliability requirements for future devices.\textsuperscript{1}

In general, the aging of devices is proportional to the device stress time and the switching frequency of the internal nodes. Therefore, if a device has a highly biased duty cycle ratio, i.e., logic “0” for the pMOS device, it will have a heavy stress and the aging of the device will be accelerated. Since the register file is holding the current processor context, as well as intermediate computation results, the performance and reliability of the register file is very important for high-performance and reliable microprocessor design. However, due to the presence of the narrow-width values (the data with many leading 0s/1s can be represented by fewer bits than the full data width), integer register files suffer a very highly biased duty cycle ratio, thus a heavy NBTI stress, especially for these leading 0s nodes in the register entries. Besides the register files, the on-chip data and instruction caches also suffer a heavy NBTI stress due to the uneven use of the cachelines and the presence of the narrow-width values.\textsuperscript{11} However, the NBTI-induced degradation of device reliability cannot be mitigated simply by adopting some traditional techniques, such as guardbanding, which may incur significant reduction in circuit speed. Therefore, the focus of this
work is the microarchitectural solution to balance the duty cycle ratio and mitigate the aging stress.

In this paper, we propose aging-aware designs to combat the lifetime degradation in the performance and reliability of the integer register file, data cache and instruction cache, by duty cycle balancing. For the integer register file, based on the fact that the leading bits (leading 30 bits for 64-bit data) of narrow-width register values are not needed during register accesses, we propose to bit-flip/complement these leading bits periodically. Further, to reduce the power consumption of the register file, the leading bits of the narrow-width values are gated during the register access for power saving. For the data and instruction caches, we first conduct the detailed analysis on their lifetime behaviors to categorize the cachelines into different groups. For the clean cachelines in both caches, we propose to do the idletime-based cacheline invalidation (CI) first and then bit-flip these invalidated cachelines periodically. For the dirty cachelines in the data cache, we propose to do the idle-time-based early write-back first, and then do the invalidation and apply the bit-flipping. For the invalid cachelines in both caches, we can just do the bit-flipping periodically. By carefully choosing the idle-time and bit-flipping time intervals, the average duty cycle ratio of the static random access memory (SRAM) cells can be well balanced with negligible performance and energy overheads, and thus the NBTI degradation will be significantly mitigated. Further, to reduce the power consumption of the data and instruction caches, we adopt the drowsy scheme for the invalidated cachelines in our design. Previous research\textsuperscript{12,13} shows that the increasing device operating temperature will accelerate the NBTI degradation. Therefore, our low-power design can further reduce the temperature (power density) and mitigate the NBTI degradation.

The rest of the paper is organized as follows. In the next section, we discuss related work in aging-aware/NBTI-aware designs. In Secs. 3–5, we provide detailed designs of our proposed low-power aging-aware register file (AARF) data cache and instruction cache, separately. The experimental setup and results are presented and discussed in Sec. 6. Section 7 draws the conclusion.

2. Related Work

As the technology is continuously scaling down, the exacerbated performance and reliability concerns caused by the lifetime degradation of complementary metal oxide semiconductor (CMOS) devices have drawn a wealth of research. To mitigate the NBTI-induced aging on the SRAM cells, there are mainly three types of solutions: (a) design customized NBTI-resilient SRAM cells\textsuperscript{14,15} (b) exploit low-energy states of the SRAM cells for alleviating the aging effect\textsuperscript{16–18} and (c) balance the duty cycle ratio of the SRAM cells.\textsuperscript{19–21} In Kumar et al.'s work,\textsuperscript{19} the impact of the NBTI on the SRAM cells was studied and an NBTI-aware SRAM structure operating in the inverted mode during half of the time was proposed. Abella et al. proposed and
evaluated the design of Penelope, an NBTI-aware processor. Penelope consists of generic strategies to mitigate the degradation in both combinational and storage blocks. It has global strategies as well as specific mechanisms to protect all types of structures, such as memory-like blocks, in the processor. A microarchitecture redundancy scheme was proposed by Shin et al. for combating NBTI-induced wearout failure in on-chip cache SRAM. In Gunadi et al.’s work, a holistic approach (Colt) to equalize the duty cycle ratio and the usage frequency of the devices in modern microprocessor was proposed. Colt employed the complement mode execution, cache set rotation and operand identifier swapping schemes to mitigate the detrimental effects of aging. Oboril et al. proposed the aging-ware designs for instruction coding and instruction pipelines. The aging-ware designs for the general-purpose graphics processing unit (GPUs) and video memories were also proposed and studied. Yang et al. proposed the techniques of sensing the NBTI degradation in the register files and it could be foundational for the reliability management schemes.

Compared to the data flipping technique proposed for SRAM cells in Kumar et al.’s work, which requires extra XOR gates to invert the data back to the normal mode during the SRAM cell access, our aging-aware design does not need to do the bit flipping during the access, thus it has no impact in cycle time. Colt uses the flipped SRAM cell content without the need to flip them back. However, the complement mode applied to the whole data path, control path and storage hierarchy (including register files) is too complicated for management. Moreover, extra XOR gates are still needed in Colt to do the bit complementing. In our design, no extra XOR gates are needed for bit-flipping/complementing since the bit-flipping/complementing operation in our design is just writing ones or zeros to the leading bits according to their current status. Penelope relies on the idle time of the processor resources, such as pipelines, cache blocks, registers, etc., to balance the duty cycle ratios of the devices. The power consumption will increase due to their value sampling and the updates to the inverted register (RINV) registers. Our design does not rely on the availability of the idle times. Even under the situation that the data contents are heavily in use, our design can still mitigate the NBTI stress effectively. Moreover, compared to all previous work, our design can also reduce the power consumption, thus the temperature of the on-chip memory structures. Therefore, it will further mitigate the aging effect.

3. Low Power Aging-Aware Register File

3.1. Narrow-width values

The narrow-width values in high-performance microprocessors have been well studied and exploited for performance and power optimizations. In a 64-bit microprocessor, values that can be represented by less than 64 bits are generally referred to as narrow-width values. In our simulated microprocessor, the experimental results
show that on average 96% of the produced integer register values can be represented by no more than 34 bits and most of them have 30 bits leading zeros.

The presence of narrow-width values is a significant contributor to the NBTI stress of integer register files. For example, in a 64-bit register file, the higher (leading) 30 bits of the register entry stay “0” most of the time, which will accelerate the aging effect. Our experimental results shows that on average the leading 30 bits of the register entry will stay logic “0” in 97.5% of the cycle time during the execution, which produces an extremely heavy NBTI stress on the device.

3.2. Low-power value-aware register file

In Wang et al.’s work,35 a thermal-aware register file (TARF) was proposed by exploiting the narrow-width values. For the low power design, we propose a simplified value-aware register file (VARF) with low hardware overheads. Our VARF can significantly reduce the power consumption of the register file by avoiding/disabling accesses to the leading zero bits. For register read, the original 64-bit value can be restored by using the exiting sign extension logic at the inputs of the ALUs. Instead of controlling/activating the bitlines according to the bit width of the narrow-width values, we divides the integer register values into two categories: 34-bit narrow-width values and 64-bit regular values.

For the narrow-width detection, we utilize the existing leading-0/1 detection logic within the functional units to overlap the timing overhead in deeply pipelined designs.36 Figure 1 shows the schematic diagram of the proposed low power VARF. In the low power VARF, the register file is partitioned into two halves: a lower 34-bit half and an upper 30-bit half. One narrow-width flag bit is added to each register

Fig. 1. The schematic diagram of the low power VARF.
entry for bit control. In most cases, the narrow-width values are stored in the lower 34 bits and the upper 30 bits are gated for power saving. For instance, during the register file read, after precharging the bitlines, the wordline of the upper half is gated by the narrow-width flag bit, which means that the power consumption is reduced by only accessing the lower half of the entire register file for narrow-width values and the upper half is rarely accessed. The multiplex is placed in Execute stage to minimize the performance overhead. Note that compared to the TARF proposed in the previous work, our low power VARF has a much simpler design, which does not need the support of the value swapping/interleaving between two halves. The narrow-width values are always stored in the lower half of our VARF. The TARF needs more complicated control logics to maintain their values. In addition, the upper half of our VARF is only 30-bit compared to the 34-bit in TARF. Therefore, the space overhead of our VARF is 1 bit (narrow-width flag bit) out of the 64-bit register entry (1/64 = 1.6%), while TARF needs 6 additional bits (6/64 = 9.4%).

### 3.3. Duty cycle balancing in VARF

For the original register file design, as we discussed above, the leading 30 bits of the register entries are all zeros most of the time due to the dominant narrow-width values. The unbalanced duty cycles for these bits will significantly increase the NBTI degradation. In our low power VARF file design, the leading 30 bits (upper half) of the narrow-width values are gated during the register read, which means that these upper bits are not used and can be treated as “idle”. To balance the duty cycles of these “idle” bits, we propose an AARF design based on the low power VARF. In our AARF, we propose to periodically flip/complement these upper idle bits at a predefined time interval. For example, at the very beginning, these upper 30 bits are all zeros. After a certain time interval, we flip them to all ones. Then after another time interval, we do the complementing again to bring them back to all zeros. Therefore, the duty cycle ratio of these upper idle bits can be perfectly balanced to \( \frac{50}{50} \). Note that the bit-flipping/complementing in our AARF is just writing all zeros or ones to the upper 30 bits. Therefore, no extra XOR gates are needed to do the flipping, which means that our AARF design has much lower overheads compared to schemes in the previous work.

Since the upper idle bits are not accessed during the register read, the bit-flipping/complementing operation is not in the critical path and has no impact on the performance. However, in order to reduce the power overhead due to the bit-flipping/complementing operation, we can choose a large time interval. The narrow-width flag bit is utilized to control whether the bit flipping/complementing should be performed to upper half or not. Therefore, no additional hardware overheads are needed for each register entry. Overall, our AARF design not only can significantly reduce the NBTI stress to register file (upper half) by effectively balancing its duty cycles, but also can achieve the power saving of the register accesses compared to the
original register file design. In addition, the reduced power density in the register file will also result in the reduction in device temperature, which will further mitigate the negative aging effect.

4. Low Power Aging-Aware Data Cache

4.1. Motivation

Based on the observation from the previous work, caches often contain more “0” than “1”. In our simulated microprocessor, the experimental results also show that the duty cycle ratio in caches is not balanced to 50% (the best case), which means the pMOS device stays logic “0” at most of the time. Therefore, the stress on the SRAM cells will be uneven and further accelerate the failures in the SRAM cells especially when applying some low power computing strategies. If the conventional guardbanding technique is used, previous work showed that it would require more than 50–100 mV guardband in SRAM $V_{DDMIN}$. The high guardband will limit the supply voltage scaling and thus needs to be mitigated.

4.2. Lifetime behavior of the data cache

The lifetime behaviors of L1 caches have been broadly studied in prior work, especially for the reliability enhancement against soft errors. Due to the variety of access patterns in the L1 data cache, such as read, write, replace and write-back, the lifetime model of the L1 data cache in their studies is quite complicated, which makes the data cache difficult to be analyzed and optimized. Therefore, we simplify the lifetime model of cachelines in the data cache and divide their lifetime into the following three phases, Live, Dead and Invalid, similar to the analysis in the previous work.

- **Live**: lifetime phase between first access and last access of a data item,
- **Dead**: lifetime phase between the last access and the replacement of a data item,
- **Invalid**: lifetime phase when the data item is in the invalid state.

Figure 2 shows the correlation among three lifetime phases for typical data cache activities, and the access (A) can be a cache read (R) or a cache write (W). Notice that the data item in the data cache can be a cacheline, a word, a byte or a single bit. Although previous work claims that a byte-level analysis is accurate for the lifetime

```
Cache Miss

Replace

Access | Access | Access

Invalid | Live | Dead

Fig. 2. The lifetime of a data item in the data cache.
```
characterization for the data cache, we choose the a cacheline level model in our following study for two reasons: (a) the control of a byte-level bit-flipping/complementing is too costly, so we do the bit-flipping/complementing for each cacheline and (b) the target of our work is to mitigate the NBTI-induced aging in the data cache, we do not need an accurate model to characterize the lifetime behavior of the data cache.

4.3. Aging-aware design for different lifetime phases

Based on the lifetime categorization of the data cache, different strategies can be adopted to different lifetime phases in order to reduce the NBTI stress of the SRAM cells while maintaining the minimum overheads. For the cachelines in the invalid state, we propose to simply bit-flip/complement these cachelines periodically. Since the invalid data in these cachelines will not be needed in the future, we do not need to flip them back even if they are in the complemented mode when the cachelines are becoming valid due to the update from the L2 cache. Notice that our bit-flipping/complementing is just writing all zeros or ones to these cachelines. Therefore, no extra XOR gates are needed to do the flipping, which means that our design has much lower overheads compared to the previous schemes.19,21

For the cachelines in the valid states, we cannot simply do the bit-flipping/complementing since the data may be needed in the future during the cache access. For the cachelines in the Live phase, if we do the similar flipping scheme (writing zeros or ones) to the cachelines, the data will be erased. If we adopt the inverting scheme proposed in previous work,19,21 extra XOR gates are needed to do the inverting for each bit. For the clean cachelines in the Dead phase, they are actually not needed in the future. Since the data in the clean cacheline is read-only, the data will be just discarded at the replacement. It seems that we can do the similar flipping scheme for these cachelines. However, the problem is that we cannot know which read operation to the clean cacheline is the last read during the program execution. Therefore, we cannot determine when our bit-flipping/complementing can be applied.

Based on the observation that most read–read (RR) instances have small intervals (less than 1K cycles) and these RR instances with small intervals only contribute a small percent of the overall RR time, Wang et al.39 proposed a clean cacheline invalidation (CCI) scheme to reduce the vulnerability factor of the clean cachelines in the data cache by invalidating the cachelines after being idle for some predefined intervals. Different from their scheme, we adopt the CI scheme to do bit-flipping/complementing and reduce the NBTI stress on the cachelines. After the clean cacheline in the data cache remains idle for a certain predefined interval, we propose to invalidate it and then do the bit-flipping/complementing similar to these invalid cachelines. By applying our CI and flipping (CIF) scheme, most of Dead phase in the clean cacheline will be converted into the invalid phase, therefore the NBTI stress can be mitigated by the bit-flipping/complementing. Moreover, the Live phase in
the clean cacheline will be reduced if a small invalidation interval is chosen. Therefore, part of the Live phase will also be converted into the invalid phase and its aging effects can be mitigated. The remaining Live phase is not optimized in terms of the NBTTI stress. However, since the remaining Live phase only contributes a small percentage (less than 10%) to the cacheline lifetime, the overall duty cycle ratio of the SRAM cells in the clean cachelines will be well balanced.

For the dirty cachelines in a write-back data cache, the data in these cachelines are still needed and will be written back into the L2 cache at the replacement. Therefore, we cannot apply our CIF scheme to the dirty cachelines directly. Instead, we propose to do the idle-time-based early write-back (EWR) first, and then do the invalidation and bit-flipping/complementing. Similar to the clean cachelines, due to the small percentage of the Live phase after the early write-back and invalidation, the overall duty cycle ratio of the SRAM cells in the dirty cachelines will also be well balanced. Notice that for a write-through data cache, the situation is much simple. Since all cachelines are clean in the write-through data cache, we can just apply our CIF scheme to balance the duty cycle ratio.

4.4. Microarchitecture of the AADC

The key issues in the aging-aware data cache (AADC) design are how to do the early write-back (EWR) for the dirty cachelines, the CI for the clean cachelines, and the bit-flipping/complementing for the invalid cachelines. Figure 3 shows the block diagram of our AADC design. We use the valid bit \( V \) in the tag array to control whether the EWR/CI or the bit-flipping/complementing scheme should be applied to each cacheline. For the valid cacheline \( V = 1 \), an \( N \)-bit global counter (IT for idle-time based) ticked by the clock signal and a per cacheline two-bit local counter ticked by the global counter every \( 2^N \) cycles are introduced. The local counter is reset to zeros once the cacheline is accessed. If the local counter saturates, we use the dirty
bit \((D)\) to control either EWR+CI is performed for the dirty cacheline \((D=1)\), or only CI is performed for the clean cacheline \((D=0)\). After that, the valid bit \(V\) is set to zero, and the local counter is also reset to zero. For the invalid cacheline \((V=0)\), a global counter (BF) is used for the bit-flipping/complementing. The BF counter and the cacheline state zero bit \((Z)\) work together to determine whether all zeros or ones should be written into the entire cacheline. If the BF counter saturates and the \(Z\) bit is equal to one, which means that currently the data in the cacheline are all zeros, the cacheline will be updated with all ones in order to balance the duty cycle ratios of the SRAM cells in the cacheline, and the \(Z\) bit will be set to zero. If the BF counter saturates and the \(Z\) bit is equal to zero, all zeros should be written into the cacheline and the \(Z\) bit will be set to one. Note that in order to minimize the area overhead of our AADC design, we choose the same idle time interval for EWR and CI. Therefore, only one two-bit local counter is needed for each cacheline and it can be shared by using \(D\) bit for dirty and clean cachelines.

### 4.5. Power optimization

Some previous aging reduction solutions explore the aging benefits provided by the low-energy states\(^{16–18}\). If power saving or leakage control schemes\(^{40,41}\) are applied to data caches, the aging effect will be mitigated. In our AADC design, the cachelines will not be needed after the invalidation, which makes them very suitable for applying the power saving schemes, such as the drowsy scheme. Therefore, we propose to adopt the drowsy scheme to further reduce the aging effect of the data cache, i.e., applying the drowsy scheme to these invalidated cachelines. Moreover, the leakage control and power saving schemes will also result in the temperature reduction in the data cache, which can further mitigate the aging.

### 4.6. Area, performance and power overheads of the AADC

As we discussed above, no extra XOR gates or inverting operation are needed in our AADC design. The space overhead of our AADC is mainly from one extra \(Z\) bit indicating the current state (all zeros or ones) for the invalid cachelines, and the two-bit local counter to support the CI for each cacheline. The space overheads of the global counter BF and IT are negligible since they are shared by the entire data cache. The space overheads of the \(Z\) bit and the two-bit local counter is also very low. For example, in a microprocessor with a cacheline size of 64-byte in the data cache, the space overhead of our AADC compared to the data array is only 3 bits out of 64 bytes \(3/(64 \times 8) = 0.6\%\).

For the performance overhead, since the data in the invalid cacheline will not be needed in the future and the bit-flipping/complementing operation is not in the critical path, there is no impact on the performance. However, the early write-back and CI schemes do have the impact on the performance, because the invalidation operations may cause additional cache misses, if the invalidated cachelines need to
be accessed in the near future. Therefore, we need to carefully choose the proper idle interval in order to maximize the lifetime aging mitigation and minimize the performance degradation. Note that the drowsy scheme in our AADC has no performance impact, since all the cachelines in drowsy modes are invalid and will not need to be wakened up during accesses.

The major contribution of the power overhead in our AADC scheme is the bit-flipping/complementing operation. In general, a large time interval for bit-flipping/complementing should be used in order to reduce the power overhead. However, if the time interval is too large, the effectiveness of the duty cycle balancing will be hurt. Therefore, a proper bit-flipping/complementing interval needs to be chosen.

5. Low Power Aging Aware Instruction Cache

5.1. Lifetime behavior of the instruction cache

Due to variety of access patterns in the L1 data cache, such as read, write, replace and write-back, the lifetime behavior of the L1 data cache is much more complicated than that of the instruction cache, which makes the data cache more difficult to be analyzed and optimized. On the other hand, due to read-only property of the instruction cache, the operations to the instruction cache are just read, replace and invalidate (during the cache flush). Therefore, the lifetime phases of the cachelines in the instruction cache are easy to be categorized.

According to the previous work, the lifetime of the instruction cache can be divided into the following three phases: $\text{RR}$, $\text{RPL}$ and $\text{Invalid}$, based on the previous activity and the current one.

- $\text{RR}$: lifetime phase between two consecutive reads of a data item,
- $\text{RPL}$: lifetime phase between the last read and the replacement of a data item,
- $\text{Invalid}$: lifetime phase when the data item is in the invalid state.

Figure 4 shows the correlation among three lifetime phases for typical instruction cache activities. Similar to the data cache, we choose a cacheline level model for the data item in our following study.

5.2. Aging-aware design for different lifetime phases

For the instruction cache, we also adopt different strategies to different lifetime phases in order to reduce the NBTI stress of the SRAM cells, based on the lifetime

\[
\begin{array}{cccccc}
\text{Read Miss} & \text{Read} & \text{Read} & \text{Read} & \text{Read} & \text{Replace} \\
\uparrow & \text{RR} & \uparrow & \text{RPL} & \text{Invalid} & \text{RR} \\
\end{array}
\]

Fig. 4. The lifetime of a data item in the instruction cache.
categorization. For the cachelines in the invalid state, we propose to simply bit-flip/complement these cachelines periodically. For the cachelines in the valid states, we use the similar CIF scheme which is proposed for the valid clean cachelines in the data cache. Therefore, we also need to choose a proper invalidation interval for the valid cachelines in the instruction cache.

5.3. Microarchitecture of the AAIC

Figure 5 shows the block diagram of our aging aware instruction cache (AAIC) design. The control mechanism is very similar to the AADC proposed above. The N-bit global counter here is CI only for CI. There is no need to write back the dirty data for the idle-time-based EWR scheme, since all the data in the instruction cache are clean.

5.4. Area, performance and power overheads of the AAIC

Similar to the AADC design, the space overhead of our AAIC is mainly from one extra Z bit indicating the current state (all zeros or ones) for the invalid cachelines, and the two-bit local counter to support the CI for each cacheline. The space overheads of the global counter BF and CI are negligible since they are shared by the entire instruction cache. The space overheads of the Z bit and the two-bit local counter is also very low. For example, in a microprocessor with a cacheline size of 64-byte in the instruction cache, the space overhead of our AAIC compared to the data array is only 3 bits out of 64 bytes (3/(64 × 8) = 0.6%). The performance and power overheads in our AAIC scheme are also very similar to those in the AADC scheme, which will be evaluated in the following study.

For the power reduction, we also adopt the drowsy scheme for the invalid cachelines in the instruction cache.
6. Experimental Evaluation

6.1. Experimental setup

We derive our simulators from SimpleScalar V3.0\textsuperscript{42} to model a high-performance microprocessor similar to Alpha 21364. In the new simulator, the original register update unit (RUU) structure is replaced by a separated integer issue queue, a floating-point issue queue, an integer register file, a floating-point register file and the active list (a.k.a. the re-order buffer). Table 1 gives the detailed configuration of the simulated microprocessor. To evaluate the power efficiency of our design, the McPAT\textsuperscript{43} is used for power profiling (at 22 nm technology).

For experimental evaluation, we use the SPEC CPU benchmark suite compiled for the Alpha Instruction Set Architecture (ISA) using the “-arch ev6-non\_shared” option with “peak” tuning. For integer register file, we use 12 integer benchmarks for our experimental evaluation. For data and instruction caches, 10 benchmarks are randomly selected. We use the reference input sets for this study. Each benchmark is first fast-forwarded to its early single simulation point (\textit{gap} uses the standard single simulation point instead of the very large early single simulation point) specified by SimPoint.\textsuperscript{44} We use the last 100 million instructions during the fast-forwarding phase to warm-up if the number of skipped instructions is more than 100 million. Then, we simulate the next 100 million instructions in detail.

<table>
<thead>
<tr>
<th>Processor core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datapath Width</td>
</tr>
<tr>
<td>Int Issue Queue</td>
</tr>
<tr>
<td>FP Issue Queue</td>
</tr>
<tr>
<td>Load/Store Queue</td>
</tr>
<tr>
<td>Active list (ACL)</td>
</tr>
<tr>
<td>Int Register File</td>
</tr>
<tr>
<td>FP Register File</td>
</tr>
<tr>
<td>Function Units</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Branch predictor</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Branch Predictor</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>BTB</td>
</tr>
<tr>
<td>Memory hierarchy</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>L1 I/DCache</td>
</tr>
<tr>
<td>L2 UCache</td>
</tr>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>TLB</td>
</tr>
</tbody>
</table>
6.2. Experimental results and analysis for VARF

To study the NBTI stress of the original register file design, especially the upper 30 bits for the narrow-width values, we profile the stress duty cycle ratio for the original register file by dividing it into two halves: the lower 34-bit half and the upper 30-bit half. According to our profiling results, around 96% of the integer register values can be presented by no more than 34 bits. Therefore, the NBTI stress of the leading (upper) 30 bits should be very high. Figure 6 shows that the lower 34-bit half (Lower_Half) has a stress duty cycle ratio of 68.5%, while the upper 30-bit half has much higher (Upper_Half) stress duty cycle ratio of 97.5%. If we consider the NBTI stress for the entire register file (Entire_Reg), the average stress duty cycle ratio is 82.1%. The results confirm us that we need aging-aware design to reduce the NBTI stress of the register file, especially for the upper 30 bits.

To implement our low power AARF, first we need to decide time interval for bit-flipping/complementing. If we use a small interval, the power overhead will increase, but the duty cycle will be more perfectly balanced. If a large interval is adopted, the power consumption will be reduced, but the effectiveness of the duty cycle balancing will be hurt. Based on our experimental results, a 40K-cycle bit-flipping/complementing interval has negligible power and performance overheads with nearly perfect duty cycle balancing capability. Therefore, we choose the 40K-cycle bit-flipping/complementing interval for our low power AARF design.

Figure 7 shows the average stress duty cycle ratio for the low power AARF design. For the upper 30-bit half (Upper_Half) in the AARF, the stress duty cycle ratio is reduced to 51.8%, which is very close to the ideal stress duty cycle ratio of 50%. If we consider the entire register file, the average stress duty cycle ratio is also reduced to 60.7%. Previous study has shown that the gate-oxide failure probability is

![Graph](image-url)

Fig. 6. The average stress duty cycle (zero) ratio for original integer register files.
proportional to the device stress time. Therefore, we can expect a similar MTTF (mean time to failure) improvement for the register file.

Compared to other aging-aware designs, our AARF can also reduce the power consumption of the register file significantly. As we discussed in Sec. 3, the power consumption of the register file will be reduced because only lower half of the entire register file is accessed for narrow-width values. The bit flipping/complementing operation has negligible power overhead due to the large (40K-cycle) time interval. Figure 8 shows that our AARF design can achieve a 30.8% power reduction for the integer register files. These power reduction in AARF can result in on average 5-degree temperature reduction in the register file, which can further mitigate the aging effect.

Fig. 7. The average stress duty cycle (zero) ratio for the low power AARF.

Fig. 8. The power consumption reduction rate for the low power AARF.
6.3. Experimental results and analysis for AADC

Before applying our AADC design, we first conduct the detailed lifetime behavior analysis on the data cache in our simulated microprocessor and this characterization is performed at the cacheline level. Our experimental results show that most of the cachelines in the data cache are valid (in-use) during the execution. As shown in Fig. 9, 99.5% of the cachelines in the data cache are valid on the average. The Live and Dead phase are the lifetime phases when the cachelines are in the valid state. Figure 9 shows that the Live phase accounts for about 24.6% of a cacheline’s lifetime and the Dead phase contributes about 74.9% on the average. Therefore, in order to apply different effective aging mitigation schemes according to the different lifetime behaviors of the cachelines, we first divide the cachelines into two groups in our AADC study: valid and invalid cachelines.

For the invalid cachelines, we propose to bit-flip/complement these cachelines periodically. However, as we discussed in Sec. 4, we need to choose the bit-flipping/complementing time interval carefully in order to balance the average duty cycle ratio of the invalid cachelines and minimize the overheads. If we use a small interval, the power overhead will increase, but the duty cycle ratio will be more perfectly balanced. If a large interval is adopted, the power consumption will be reduced, but the effectiveness of the duty cycle balancing will be hurt. Based on our experimental results, a 40K-cycle interval for bit-flipping/complementing has negligible power and performance overheads with nearly perfect duty cycle balancing capability. Therefore, we choose the 40K-cycle bit-flipping/complementing interval for our AADC design.

For the valid cachelines, our experimental results in Fig. 10 show that the average stress duty cycle (zero) ratio is 84.0%, which needs to be further reduced. For clean cachelines, based on the observation that most of the RR instances have small...
intervals (less than 1K cycles), we propose to use an idle-time-based CI scheme to invalidate the valid cachelines after being idle for some predefined intervals in Sec. 4. By applying the CI scheme, most of the duty cycles of clean cachelines will be converted into the duty cycles of invalid cachelines, and thus can be further reduced by adopting the bit-flipping/complementing. However, similar to the bit-flipping/complementing scheme, the problem is how to choose the proper invalidation interval that can reduce the RR phase significantly with negligible performance loss. Our experimental results show that if a small 500-cycle interval is chosen, the RR phase can be significantly reduced to 0.5% from the original 13.7%, but the performance loss is also high, 5.4% on the average. This high performance loss is mainly caused by the high pipeline stall penalty due to the increased data cache misses incurred by the CI scheme, which is not affordable in high-performance designs. On the other hand, if a large 64K-cycle interval is used, the performance degradation is less than 0.3%, while the RR phase will increase to 6.3%. Based on our experimental results, 4K-cycle is a good choice for the CCI. The performance loss is under 0.7% and the RR phase is reduced from 13.7% to 2.4%.

For dirty cachelines, we proposed to adopt the idle-time-based EWR scheme first, and then apply the invalidation and bit-flipping/complementing. Similar to the idle-time chosen for CCI, we conduct a study based on different idle times and the experimental results show that 4K-cycle is also a good choice for the EWR, which can effectively reduce the Live phase in dirty cachelines with negligible performance overheads.

Therefore, as we discussed in Sec. 4, we choose a 4K-cycle interval for both idle-time-based CI and early write-back to minimize the area overhead of our AADC design. After the CI, we use the same 40K-cycle interval for bit-flipping/complementing in order to achieve duty cycle balancing. Our experimental results in Fig. 11 show that our AADC design can reduce the average stress duty cycle ratio to

![Fig. 10. The average stress duty cycle (zero) ratio for valid cachelines in the data cache.](image-url)
54.1% for all cachelines in the data cache with the performance loss under 0.8%. Previous study has shown that the gate-oxide failure probability is proportional to the device stress time.\textsuperscript{4} Therefore, we can expect a similar mean time to failure (MTTF) improvement for the data cache, which is 48% in our study.

For further power saving, we propose to adopt the drowsy scheme to these invalidated cachelines. We scale the power numbers provided in Flautner et al.’s work\textsuperscript{41} for this study. Since the data in invalidated cachelines of our AADC design will not be needed during the drowsy mode, the performance overhead due to the wake-up operations for drowsy scheme can be ignored. Figure 12 shows that our

![Fig. 11. The average stress duty cycle (zero) ratio for all cachelines after applying the proposed AADC scheme.](image)

![Fig. 12. The power reduction rate by applying the drowsy scheme in the data cache.](image)
AADC design can achieve a 64.5% power reduction for the data cache, which can further mitigate the aging effect.

6.4. Experimental results and analysis for AAIC

Before applying our AAIC design, we first conduct the detailed lifetime behavior analysis on the instruction cache at the cacheline level. Different from the data cache, our experimental results show that not most of the cachelines in the instruction cache are valid (in-use) during the execution. As shown in Fig. 13, only 33.3% of the cachelines in the instruction cache are valid on the average. Some applications, such as vpr and bzip2, have a very low cacheline-in-use ratio (less than 10%), while some applications like gcc and crafty have a high cacheline-in-use ratio (more than 90%). For the processor with a small instruction cache compared to our simulated one, the cacheline-in-use ratio may increase. However, the performance will be degraded for the benchmarks with high demand in instruction cache size, such as gcc and crafty. Therefore, normally we will not adopt a small instruction cache in the processor in order to increase the cacheline-in-use ratio.

The RR and RPL phase are the lifetime phases when the cachelines are in the valid state. Figure 13 shows that the RR phase accounts for about 21.5% of a cachelines lifetime and the RPL phase contributes about 11.8% on the average. The RR phases in gcc and crafty are also very high (more than 50%) due to their high utilization of the cachelines. Similarly, in order to apply different effective aging mitigation schemes according to the different lifetime behaviors of the cachelines, we divide the cachelines into two groups in our AAIC study: valid and invalid cachelines.

For the invalid cachelines, we propose to bit-flip/complement these cachelines periodically. Similar to the data cache, we need to choose the bit-flipping/complementing time interval carefully. Based on our experimental results, an 80K-cycle

![Fig. 13. The lifetime distribution of the cachelines in the instruction cache.](image-url)
interval for bit-flipping/complementing has negligible power and performance overheads with nearly perfect duty cycle balancing capability. Therefore, we choose the 80K-cycle bit-flipping/complementing interval for our AAIC design.

For the valid cachelines, our experimental results in Fig. 14 show that the average stress duty cycle (zero) ratio is 70.5%, which needs to be further reduced. For adopting the CIF scheme, our experimental results show that if a small 1K-cycle interval is chosen, the RR phase can be significantly reduced to 3.0% from the original 21.5%, but the performance loss is also tremendous, 19.3% on the average. This high performance loss is mainly caused by the high pipeline stall penalty due to the increased instruction cache misses incurred by the CI scheme, which is not affordable.

Fig. 14. The average stress duty cycle (zero) ratio for valid cachelines in the instruction cache.

Fig. 15. The average stress duty cycle (zero) ratio for valid cachelines after applying the proposed AAIC scheme.
in high-performance designs. On the other hand, if a large 64K-cycle interval is used, the performance degradation is less than 0.5%, while the RR phase will increase to 16.3%. Based on our experimental results, 16K-cycle is a good choice for the CI. The performance loss is under 0.9% and the RR phase is reduced from 21.5% to 8.7%.

Therefore, for the valid cachelines, we choose a 16K-cycle interval for idle-time-based CI, and after the CI, we use the same 80K-cycle interval for bit-flipping/complementing in order to achieve duty cycle balancing. Our experimental results in Fig. 15 show that idle-time-based CI with the bit-flipping/complementing can reduce the stress duty cycle ratio to 56.2% for the valid cachelines with the performance loss.
under 0.9%. By further combining the bit-flipping/complementing scheme for the invalid cachelines, our AAIC design can reduce the average stress duty cycle ratio to 51.7% for the entire instruction cache, as shown in Fig. 16.

For power reduction, we also adopt drowsy scheme to these invalidated cachelines in the instruction cache. Figure 17 shows that our AAIC design can achieve a 72.0% power reduction for the instruction cache.

7. Conclusion

The performance and reliability degradation due to the aging effect are becoming substantial for CMOS devices in future technologies. In the high-performance microprocessors, on-chip memory structures, such as register files and on-chip caches, suffer an extremely high NBTI stress, which will accelerate their lifetime degradation. In this paper, we propose low power aging-aware designs to combat the aging effect in integer register files, data caches and instruction caches. For the integer register file, we propose to periodically bit-flip/complement the leading bits of the narrow-width values in registers. For the data and instruction caches, based on our detailed study on the lifetime behaviors of the cachelines, different aging reduction schemes, such as idle-time-based invalidation for clean cachelines, EWR and invalidation for dirty cachelines, and bit-flipping scheme for invalid cachelines, are proposed. Experimental results show that by applying our aging-aware design, the duty cycle ratio of these on-chip memory structures can be reduced to ~ 50% and the device stress will be significantly mitigated. In addition, our low power aging-aware design can also achieve a 30.8%, 64.5%, 72.0% power reduction in the integer register, data cache and instruction cache, respectively, which will further mitigate the aging effect.

Acknowledgment

This work was supported in part by a grant from National Science Foundation of China under Grant No. 61321491.

References


