**Electro-Migration Aware Architecture for Modern Microprocessors**

|  |
| --- |
|  |

**Abstract**

Reliability is a fundamental requirement in any microprocessor design to guarantee correct execution over its lifetime. The design rules related to reliability depends on the process technology being used and the expected operational conditions of the device. The use of advanced process technologies (28nm and below) imposes highly challenging design rules in order to meet reliability requirements. Such design-for-reliability rules have become a major burden on the VLSI implementation flow due to their severe physical constraints.

The focus of this paper is on one of the major critical factors that affect semiconductor reliability, Electro-migration (EM). EM is the aging process of on-die wires and vias. It is induced by an excessive current flow that can potentially result in damaged wires and may also have a significant impact on the IC clock frequency. EM has a comprehensive global effect since it impacts wires that may reside inside the standard or custom logical cell, between logical cells, inside memory elements, and within wires that interconnect functional blocks.

So far, design implementation flows (synthesis and place-and route) detect the EM reliability rules violations and attempted to solve them; this paper proposes a new approach that suggests to enhance these flows with an EM-aware architecture. This study shows that our solution can relax EM design efforts in microprocessors and extend their lifetime by x2 or higher. This work demonstrates this new approach for modern micro-processors, though the principals and ideas can be adopted to other use-cases as well.

# Introduction

Chip reliability is an essential design requirement that is crucial to assure the correct functionality of a semiconductor integrated circuit (IC). Chip vendors are required to provide for every product, e.g., processor, a guaranteed minimum lifetime, that depends on its reliability prediction. To meet these reliability requirements, a design-for-reliability methodology was developed. Unfortunately, it is a highly complicated process since it depends on the expected workload, the process technology, the operating voltage, and temperature (PVT) conditions. As part of the design-for-reliability methodology of modern processors, a workflow is defined [15, 16, 20] that aims to guarantee a minimum lifetime of the product under a specified workload, aka mission profile. The need for high reliability is becoming more crucial recently as a result of the use of new advanced process technologies and the use of new applications such as compute-intensive infrastructures, e.g. autonomous cars, data center computing, cloud computing, life support systems etc.

Electro-migration (EM) has become one of the most influential factors on modern systems reliability. This is due to the shrinking VLSI technology dimensions, the increasing density of logical elements and the challenging operating conditions of voltage and temperature. EM is a phenomenon related to wires and vias reliability in integrated circuits. It is induced by an excessive current flow that can potentially damage a physical device. Such a damage may cause either reduction of wires conductivity or a wire disconnect. Both can result in reliability concerns. In this work, we focus on the impact of the EM on on-chip wires vias that reside inside logical cells or memory elements; or used as interconnects between logical cells or functional units.

So far, the design community has focused on enhancing the chip design implementation flow ([1], [8], [10], [13], [14], [15], [16], [17], [18]) to solve EM issues and there were limited works that proposed architectural solutions. In this study, we present a novel architecture that significantly improves reliability by reducing EM impact while relaxing the physical design efforts and significantly extending microprocessors' lifetime. The groundwork of this study is based on the observation that many of the reliability concerns are a result of excessive write activities (or change of logical state) spread across the processing elements of the same type (gates, logical units, or memories elements) in a non-uniform manner. This observation leads to the development of enhanced new resource allocation mechanisms that uniformly distribute write operations workload across all resources. As a result, the maximum EM stress induced by singular elements is minimized, and the overall IC reliability is extended in up to several orders of magnitude. In this work we present the microprocessor as a case-study, though the concepts can be applied to other ICs and applications as well.

The rest of this paper is organized as follows: Section 2 introduced EM reliability challenges and provide overview on EM and previous works to deal with EM effects, Section 3 introduces the limitations of modern microprocessor architecture to deal with EM effects, Section 4 describes our novel EM-aware architectural enhancements, Section 5 presents simulation results of the proposed EM-aware architecture and Section 6 summarizes this study and suggests future research works.

# IC reliability

IC reliability has become a crucial discipline in VLSI chip design. The need for highly reliable systems exists from the early days of modern computing. In. the past such a demand was mainly driven by “special-systems” such as mission-critical embedded systems. However, today reliability is a fundamental requirement from most systems due to the vulnerability of the new process technology and the appearance of new applications that require highly safe and reliable processing such as autonomous cars, large-scale compute-intensive systems (e.g., HPC, cloud computing, data centers) and life-supporting systems. The product specification of such systems enforces strict requirements on reliability through the lifetime and operating conditions. For example, the automotive industry expects an IC to correctly function for 10-15 years under given temperature (usually range of up to 125C [22], [23]) and under various workloads. In datacenter computing, the requirements are little relaxed but still challenging: the lifetime requirements is for at least 10 years while temperature can range between 105-110C with arbitrary workloads. All these reliability sensitive applications cannot afford any microprocessor fault induced by reliability causes.

In the recent decade, as advanced process technologies were introduced, the susceptibility to reliability related issues has dramatically grown. Starting at process technology of 28nm and below (16, 7, 5 and 3nm), the design efforts dedicated to reliability have substantially increased. The design community has mainly invested efforts to enhance the synthesis and place-and-route flows to minimize and solve reliability related issues. Such flows involve substantial design efforts and, in many cases, required multiple iterations to converge the IC compliance with the design rules (also know as the sign-off process). It should also be noted that limited number prior works exists that addressed these physical reliability challenges from the architecture point of view ([8], [10], [13], [14]). The rest of this section provides an overview is provided on the EM phenomenon and the prior works to handle EM impact.

## 2.1 Electro-Migration

Electro-Migration (EM) is a physical phenomenon related to excessive current density within wires and vias. EM had become a major concern in advanced process technologies where the geometrical dimension of wires and vias shrank into very small dimensions such that they became highly susceptible to the negative effect of electrical current stress. This stress is induced by the force of conduction electrons and metal ions. When the force of conduction electron reaches a certain strength level, it may tear atoms from boundary of the metal and transport them in the direction of the current flow. When such current force is kept constant for a long time or when there are frequent current flows, the wire may become malformed. One may consider to switch the current directions in order to ease the problem, but experiments indicate that it has only minor impact on the overall reliability issues; e.g. wire disconnect or significant change in the wire resistance. When such an issue occurs even on a single wire, it may result in the overall chip failure. It should also be noted that geometrical granularity of wires play a major role in susceptibility to EM where smaller wire granularity encourages higher EM forces. Therefore, we expect that EM will continue to be a major challenge in semiconductors as we leverage new advanced process technologies ([18]).

In EM study done by Black ([6]), a formula for the EM mean time to failure, MTTF, was introduced:

Equation 1 - EM MTTF

Where A is a constant, J is the current density, is the activation energy, n is a scaling factor, K is the Boltzmann constant and T is the temperature. It can be observed that the MTTF is exponentially dependent on temperature. In fact, higher temperature accelerates the EM negative effect since it weakens the wire’s atom connections by making them even more sensitive to EM forces. Since many new applications, and in particular control systems such as in automotive or robotics are required to operate at high thermal conditions of 105-125C; this induces even much higher EM susceptibility that will be highly challenging to mitigate during the IC implementation and signoff.

In addition to the temperature effect, [6] and [14] formulated the current density in metals J, as:

Equation 2 - Current Density

When C is the wire capacitance, W/H are the metal width/height, VDD is the operating voltage, f is the clock frequency, and p the switching probability, also known as the toggle rate. In order to meet the reliability requirements, 2 additional design rules constraints are usually enforced by advanced process technology design rules ([24]):

1. The current applied on every wire should be less or equal to the peak current allowed by the process technology.
2. The current flow over a wire needs to be calculated using RMS (root mean square). It should be notes that using an average current will not be useful for EM analysis since usually the average current is 0. This is due to the fact that the number of charge carries is equal when charging and discharging an electrical junction. Further study on RMS current can be found in [24].

For advanced process technologies, RMS current has become a very significant cause for EM reliability concerns due to the incredibly shrinking dimension of metals and vias.

Handling both max current and RMS current design rules is highly challenging. Max current constraint is mainly enforced by the physical design implementation tools that assure that driving gates will not exceed max current limitation and by other physical design means ([24]). With respect to the RMS current, the situation is even more complicated. It can be observed in Equation 2, that the RMS current flow within wires is proportional to both toggle rate and the clock frequency. This means that the higher the toggle rate of logical elements the higher susceptibility to EM stress. Therefore, the MTTF of wires and vias can be increased by either increasing their physical with, *W*, or by minimizing their switching rate, *p*. Increasing the metal wires width has, of course, a negative impact on die area and the number of available routing resources which can effectively lead to a degradation in performance and overall power increase. Minimization of the switching probability is both workload and architecture dependent. In many cases the switching probability is a result in the change of logical state due to write operation or utilization of logical elements for different computations.

Further studies on EM and its failure effects can be found in [1], [2], [3], [4], [5] and [19]. In Section 4 we present a novel architectural solution to relax EM stress effect that takes advantage of the EM and the toggle rate relationship.

## 2.2 EM Related Prior Works

This sub-section summarizes previous EM related studies. Our overview will distinguish works that attempt to find EM solutions through the physical design flow from studies attempting to find relief through micro-architecture or architecture solutions.

### 2.2.1 Physical design based Prior Works

EM phenomena were broadly studies from the physical design point of view. Various studies ([6], [15], [20]) examined different interconnect such as copper or aluminum and how they are affected by EM under different PVT (process, voltage, and temperature conditions). The most common solution for EM, from a physical point of view, is to widen the interconnect wires. As Equation 2 indicates, widening a wire reduces the current density and eventually degrade the EM impact, but from the physical design point of view it is not always the preferable solution since it may introduce several overheads such as increase of the interconnect and the overall die area, a creation of more suspectable to crosstalk delays and hence, it may degrade the device frequency. In addition, the potential increase of die size may also create timing and power challenges as signals need to travel to longer distances.

Modern EDA tool vendors in conjunction with process foundries enforce EM related design rules that must be met as part of the IC sign-off process. Such tools validate that interconnects and vias meet the EM design rules and identify all EM related violations that require design fixes. EM analysis tools are even able to simulate switching activity patterns that are extracted from functional simulations representing real applications and take it into account in the EM analysis process. When the worst-case switching patterns cannot be determined, designers often use a statistical analysis by the EDA EM sign-off tool. In this case the design is analyzed under a given switch probability numbers - This may of course lead to an over-design process. The EM sign-off process is a very painful activity that involves many fix iterations and trials. Some of the trials involve usage of wider metals and via and in several cases, it may even end up in limiting clock frequency, the switching activity rate and the computational workload. All these limitations may result in degraded IC performance.

Additional study by Dasgupta *et al.* introduced in [13] a methodology for synthesizing the design and scheduling data transfers of the control data flow graph onto the hardware buses in EM aware manner. Their algorithm requires determining the activity in advanced and as a result it becomes tightly coupled to a specific computational use-case.

A broad survey of additional physical design-based techniques to mitigate EM impact can be found in [18].

### 2.2.1 Architecture-based Prior Works

Only limited number of prior works that suggested architecture-based solution to EM problem. Srinivasan *et al.* in [10] suggested structural duplications (SD) and graceful performance degradation (GPD) techniques to handle EM effect. SD uses spare design structures which are added to the IC and are turned on when the original structures fail. GPD however, shutdowns failing structures but keeps the IC functional while degrading its performance. This approach seems to have a major hardware area overhead related to the dedicated mechanisms to detect EM degradation through the IC normal operation and the need for special circuits to switch on the redundant logic. In addition, it introduced extra power and performance overhead due the redundant hardware being added.

Abella *et al.* suggested in [14] a novel architectural approach for “refueling” bi-directional busses by monitoring the current flow direction every time data was transferred on the bus and suggested a mechanism that will trigger a current compensation whenever there is an imbalance in the amount of current flowing in both directions. Such a scheme could indeed relief EM stress in older technologies, however it has limited impact on advanced process node technologies since the healing effect of RMS current is less effective and its negative impact on wires and vias conductivity and reliability is more significant. In addition, modern VLSI design circuits do not commonly use bidirectional buses due to their design complexity. The refueling mechanism also disrupts the bus operation and may introduce a dynamic power overhead due to the reversal current.

Srinivasan *et al.* in Suggested in [8] and [9] a dynamic reliability management (DRM) approach where the processor dynamically maintain its lifetime reliability target by responding to changing application behavior. As a result, it allows a processor with lower reliability to operate while compromising performance or operating conditions.

As indicated by this section, applying physical design-based solutions only is not sufficient due to the growing challenges involved by EM. The rest of this paper describes our comprehensive architectural solution for EM handling.

# EM Stress Distribution in Modern Microprocessors

Since EM design rules are limited by the weakest-link; i.e., the wire, which is most likely to be damaged, we start this study by looking at the distribution of the EM stress effect over the entire design of a modern microprocessor (it should be noted that the same concept could be applied to other ICs and applications as well). In this paper we choose to focus on subsystems that expect to show an intensive toggling rate of wires that in return, result in hotspots of EM stress. The next 2 sub-sections are organized as follows: In sub-section 3.1 we describe our experimental environment and in sub-section 3.2 we present comprehensive observations on EM stress in micro-processors.

## 3.1 Experimental Environment

For this study we use the sniper x86-64 simulator [21]. We modified the simulation platform and added the needed mechanisms to model the behavior and measure the characteristics required for our experiments. The simulation environment includes both a detailed cycle-level x86 core model and a memory system. The configuration of the simulation environment is summarized in the following Table 1 (based on Intel Gainestown core [25]).

|  |  |  |
| --- | --- | --- |
| Core model | Frequency | 2.66 GHz |
| Execution units | 3 ALUs, 1 FP add / sub, 1 FP mul /div  1 Branch, 1 Load unit, 1 Store unit |
| Dispatch width | 4 |
| Execution order | Out-of-order (instruction window: 128) |
| Memory system model | L1-D Cache | 32KB, 8-Way, 64B block size, LRU, 4 clock cycles access time and a throuput period of one cycle. |
| L1-I Cache | 32KB, 4-Way, 64B block size, LRU, 4 clock cycles access time with instruction prefetching and instruction queueu of 16-byte per cycle throughput |
| L2 Cache | 256KB, 8-Way, 64B block size, LRU, 8 clock cycles access time. |
| L3 Cache | 8MB, 16-Way, 64B block size, LRU, 30 clock cycles access time. |
| D-TLB | 64 entries, 4-Way |
| I-TLB | 128 entries, 4-Way |
| S-TLB (2nd level) | 512 entries, 4-Way |

Table 1 – Baseline Simulation Model Configuration

Our simulation benchmarks are Spec2017 ([11], [12]) with *ref* inputs. Every benchmark was run as a single core workload in 2 different regions of interest: Initialization phase and main execution phase (denoted as Init and Main respectively). Each experiment uses 10 billion instructions (for both initialization and main execution phased).

## 3.2 EM Stress Experimental Observations

This section examines the EM stress induced by 3 different regions of the microarchitecture: ALU execution units, architecture register files and memory hierarchy sub-system. We believe that these areas involve the most intensive EM activities when running these workloads and hence, will induce heavy EM stress.

**ALUs**: Figure 1 depicts the distribution of write operations among different ALUs, when FIFO selection mechanism, among all ready instructions is used. It can be observed that ALU0 is the most utilized ALU among all the 3 available ALUs, while ALU2 is the least utilized. This can be explained due to the fixed allocation policy of the available ALUs where a higher priority is given to a lower ALU index. Since ALU execution time is 1 clock cycle, all ALUs become available every cycle. For example, for a program that has exactly one instruction ready every cycle, we can expect that only ALU0 will be used. Figure 1 supports this claim and shows that ALU0 is utilized more than twice in respect to ALU1, and nearly 10 times more frequent than ALU2 in most benchmarks. In such a logical implementation, the worst-case switching factor of ALU0 dictates the worst-case EM scenario that will need to be taken into account and be applied to all ALUs.



Figure 1 - ALU Execution Count distribution

**Register-file:** Our next set of experiments examine the EM stress on architectural registers. Figure 2 illustrates the distribution of write operations on GPR registers (integer general purpose) for the Spec2017 benchmarks. It can be clearly observed that the distribution is not uniform, for example, the RAX register is the most stressed register in terms of write operation while non-legacy registers hardly being used, and so, are significantly less stressed than the x86 legacy registers. The root cause of these differences is the nature of register allocation algorithms of compilers. Figure 2 also shows that the ratio of average number of write operations to the maximum number of writes varies between nearly 7% to 33%. This measurement provides another indication that EM stress is not equally balanced between registers; thus, the register with the worst-case number of writes will dictate the overall switching ratio for EM.



Figure 2- General Purpose Register Writes Distribution

Figure 3 presents the number of write operation on FP registers only for the Spec2017 benchmarks that involve FP operations. The results presented in this case are similar to results presented in Figure 1. For FP registers, the number of writes is significantly higher in the registers with lower indexes, i.e., ZMM0, ZMM1 and ZMM2 are the registers with highest write count relative to all others. Similar to integer registers, this can be also explained by the nature of the register allocation algorithm of common compilers. In this case the ratio of average number of write operations to the maximum number of writes is even smaller which indicated of even a bigger variance relative to integer registers.



Figure 3 - Floating Point Registers Write Distribution

**Memory hierarchy**: Memories are highly suspectable to EM since they employ high density bitcells with narrow and long metal wires that toggle upon every change of logical state. In addition, physical design tools lack the ability to handle every bitcell in an individual manner, therefore the worst-case scenario is commonly applied to all bitcells. Since write operations are not uniformly distributed across all memory bitcells, the worst-case scenario is determined by the bitcell with the biggest number of writes.

Please note that the granularity of the EM stress differs from one level of the memory hierarchy to another’ e.g., at the L1 cache, a single byte can be written, but at all other levels of the cache hierarchy, a minimum granularity of a cache line is imposed (assuming line fill mechanism). Since all bits within that write granularity has the same EM stress, we need to assume that all of them has the same probability for failure and conventional error correction mechanisms, are not effective at that granularity.

Due to the importance of the memory hierarchy to the reliability of the entire system, Figure 4, put together the write statistics of the components that assemble modern memory hierarchy; L1 instruction and data caches, L2 cache, L3 cache and the instruction TLB (ITLB), data TLB (DTLB) and secondary TLB (STLB). This figureFigure 4 depicts the ratios of the average number of write operations (as a result of TLB entry allocation) per entry. It shows that DTLB involves a significantly higher number of writes of operations than the ITLB. DTLB also involves nearly x10 writes operations relative to STLB. Similar observation is reported when examining the ratio of write access of the L1-D cache to the L1-I cache. L1-I cache involves write-operations only upon cache lines replacement, while L1-D maintains much higher rate of write operations due to block replacement and each time that a memory location is the target of an instruction

It should be noted that although the initial observations indicate that the L1-D cache and the D-TLB are the elements with the highest rate of writes. we still need to continue carefully watching the write distributions in the rest of the memory hierarchy. In particular, it is important to watch after the write distribution to L2 and L3 caches. Although our experimental results show these caches maintain lower rates of write operations, they may be much more suspectable to EM than the L1 caches due to physical design considerations. Since both L2 and L3 are significantly bigger than the L1 cache, they involve higher density memory bitcells and significantly longer and narrower interconnect metal. Equation 2 supports this argument since it indicates that the current density is inverse proportional to the metal width while proportional to the wire capacitance. The interconnect metals in both L2 and L3, that uses long wires, introduce a much higher interconnect capacitance relative to L1 caches.



Figure 4 – Write ratios in memory hierarchy

Based on this observation, the next few graphs will focus at the EM impact on L1-D cache, L2 cache, L3 cache and D-TLB. In the next figures we present histograms of write operations partitioned into 5 histogram bins: 0-25%, 26-50%, 51-75%, 76-90% and 91-100%. Each bin shows the number of cache entries with the ratio of write distribution relative to the cache entry with maximum number of write operations. E.g., if the value of bin 26-50% is 20% then it means that 20% of the cache entries experienced (each) write operations in the ratio of 26-50% relative the cache entry with the maximum number of write operation. The cache entry with the maximum number of writes is the one that dictates eventually the EM toggle rate assumption for the entire cache and therefore illustrating the distribution of all cache entries relative to the one with the maximum number of writes can help understanding how the EM stress is distributed among all cache entries and explore for new architecture to relief the EM stress.

Figure 5 shows the write histogram of D-TLB entries and their tags. It can be observed that for all Spec2017 benchmarks, only small number of entries experience a heavy ratio (above 90% relative to the entry with the maximum number of writes) and dictate the overall switching rate of the D-TLB, while the majority of entries experience much smaller rates of write operations. This figure also illustrates the ratio of the average number of writes per entry to the entry with the maximum number of writes which varies from 2% up to 100% while the average is 55%.



Figure 5 - DTLB writes distribution

Figure 6 illustrates the writes histogram of L1-D cache data lines. We observe similar phenomenon to the one observed in the D-TLB. Only small number of cache lines have a heavy stress of writes (above 90% relative to the maximal data cache line) while the majority of cache lines experience much lower write stress. In most of the benchmarks the ratio of average to max number of writes is less that 30% while the average ratio is 33%.

Figure 7 shows the cache writes histogram of L2 cache data lines. The observations in this case are similar to the L1-D cache. For both data blocks and tags we observe that only small portion of cache entries (data and tags) experience the highest ratio of write (above 90% relative to the entry with the maximum number of writes) and as a result, they dictate severe EM conditions to all cache entries. On average we observe that the ratio of average number of writes per entry to the entry with maximum number of writes is approximately 50%. Similar behavior of write operations on cache lines was also observed by Valero *et al* in their study on different aspects of cache reliability ([19]).



Figure 6 - L1-D cache block writes distribution

When examining Figure 6 and Figure 7Figure 7, we observe that 2 benchmarks, 623.deepsjeng-init and 649.fotonik3d-main, show a different behavior relative to all other benchmarks. This is explained by the fact that the initialization phase of 623.deepsjeng and the main execution phase of 649.fotonik3d has write distribution that is spread uniformly through most of all cache lines.



Figure 7 - L2 cache block writes distribution

Figure 8 illustrates L3 writes histogram for cache data lines. It can be observed in most of the benchmarks that the number of writes is very small for the majority of cache data lines where almost all of them experience 25% or less write operation relative to a very small portion of cache lines with the maximum number of writes. Overall, the ratio of average number of write operations to the maximum number of writes in cache data lines is 8%. 631.deepsjeng-init exhibits a similar behavior of writes that are spread uniformly across all cache lines. This is a similar behavior to the L2 cache behavior due the relatively high store instruction count that peculates to the L3 cache as well.

Figure 9, Figure 10 and Figure 11 illustrates the write histograms of L1-D, L2 and L3 tag writes respectively. It is observed that writes spread more uniformly relative data lines, and that the majority of cache tags experience smaller variance in the number of writes. The ratio of average number of tag writes to the maximum number of tag write is nearly 70% on average (over all benchmarks) for L1-D cache and approximately 50% for L2 and L3 tags.



Figure 8 - L3 cache block writes distribution

To conclude the discussion on the impact of EM on the memory hierarchy, we can determine that cache data lines experience high variance distribution of writes accesses where a minority of cache data lines experience high stress of write in the maximum number of write operations and as a result dictate much more severe conditions of EM for the entire cache. Similar conclusions are obtained from our observation on register file write access and ALUs utilization where in both case the EM stress that induced by the workload is not uniformly distributed. Such behavior leads to an over-design EM conditions that can result in overall performance degradation and IC area increase. In next section we will propose novel architectural mechanisms that take EM considerations into account and smoothen EM stress uniformly. As a result, the overall EM sign-off design conditions are relaxed dramatically.



Figure 9 - L1-D cache tag writes distribution



Figure 10 - L2 cache tag writes distribution



Figure 11 - L3 cache tag writes distribution

# Our New EM Aware Resource Allocation Mechanism

In this section, we introduce our novel architecture solutions to reduce EM stress. The principal of all solutions is based on an EM aware resource allocation that smoothens write operation and computational elements utilization over all available resources As a result, it reduces the EM stress significantly. This section is divides into 3 sub-sections that introduce EM aware architectures for dealing with EM stress on: ALU execution units, register files and cache memories.

## 4.1 EM-Aware ALU Allocation

In the previous section, we observed that ALUs are not utilized in an EM-aware manner, and as a result, the maximum EM stress is dictated by a small subset of ALUs that are over-utilized. Our new EM-aware scheme, assumes that all pending ALU instructions are allocated to a centralized instruction queue, and each cycle a scheduler allocates ALUs to instructions which are ready for execution. The proposed scheme is presented for ALUs, but it can also be applied to any type of multi execution units being employed by the microprocessor.

We present two alternatives that implement the same basic principle in different ways. The aim of both solutions to start allocating the resources from a different leading point each time. The first simple solution is to have a counter that will be incremented every clock cycle and wrap around when expired so that a leading resource number to be used will be calcudlated as:

Resource# = counter mod N

Equation 3 - Leading resource allocation

Where N is the number of the physical resources. Thus, for our simulated environment we assume N=3. When the counter expired, we stop allocating resources for that cycle, reset its content and continue with the allocation next cycle.

The second solution is illustrated in Figure 12; here we extend each resource with a single bit and add a single global bit for the overall management of the allocation. All counters are initialized to 0 upon reset.



Figure 12 - EM-aware ALU allocation scheme

The allocation algorithm is specified as follows:

|  |
| --- |
| **Algorithm 1** – EM-Aware Execution-units allocation: |
| **Input**: k<N number of execution units to be allocated. |
| **Output**: Vector E=(e0, e1, …, en-1), for every 0≤ i ≤ n-1, if ei=1 execution unit i to be allocates, otherwise execution unit i will not be allocated. |
| **Initialization:** Ex\_counter[i]=0 for every 0≤ i ≤ n-1, Global\_counter=0 |
| 1. M = {0≤ i ≤ n-1 | Ex\_counter[i]= Global\_counter}  2. **if** k< |M| **then** |
| 3. let P⊂M such that |P| = k |
| 4. ei=1 for every i∈P, otherwise ei=0 |
| 5. Ex\_counter[i]++ for every i∈P |
| 6. **end if** |
| 7. **else** // k≥ |M| |
| 8. let P ⊆ U\M such that |P|= k-|M| |
| 9. ei=1 for every i∈P∪M, otherwise ei=0 |
| 10. Ex\_counter[i]++ for every i∈ P∪M |
| 11. Global\_counter++ |
| 12. **end else** |
| 13. **return** E |

We suggest the EM-aware allocation algorithm to select execution units in which their corresponding counter state is equal to the global counter. If the number of available execution units that satisfy this condition is greater than the required number of instructions to be issued, then a subset (based on the required number of instructions to be issued) of those execution units will be selected and all their corresponding counters will be switched (between 0’s and 1’s). Otherwise all execution units which their counter state equal to the global counter will be the selected while the rest of the execution units needed to satisfy the required instruction to be issued will be selected from the from the other pool of ALUs which their counter value is not equal to the global counter. Only in the latter case the global counter is incremented. In addition, the counters corresponding to the selected execution units will be incremented. The following table illustrates an example of the algorithm operation for 3 ALUs:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Clock cycle | ALU instructions to be issued | ALU  2, 1, 0  counters | Global counter | Selected ALU |
| 0 | 0 | 0, 0, 0 | 0 | None |
| 1 | 2 | 0, 1, 1 | 0 | 0, 1 |
| 2 | 2 | 1, 1, 0 | 1 | 2, 0 |
| 3 | 3 | 0, 0, 1 | 0 | 1, 2, 0 |

Table 2 - EM-Aware ALU scheduling example

As it can be observed, upon every instruction issue, both algorithms balance the utilization of all execution units and so, prevent any execution unit to be over utilized. The implementation of the first solution is straight forward and may well perform when there is a large number of execution unit resources. The implementation of the second solution can more complicated, but our implementation trial indicates that can be achieved with a negligible overhead in terms of logical area, and computation time for both ALUs selection logic and counter incrementation logic.

## 4.2 EM-Aware Register Allocation

Our experimental measurements presented in previous section clearly indicated that writes operation to registers are not uniformly distributed. Moreover, specific registers, e.g., RAX, exhibited an excessive number of writes. Such behavior by a small number of registers will dictate tough EM conditions on all registers and may result in reliability concerns. It should be noted that this section mainly deals with archtitectural registers which are assigned by the compiler rather than the physical registers as implemented by the out-of-order (OoO) microprocessors. For the latter, physical registers (implemented within the reorder-buffer) are usually implemented as a cyclic buffer and as a result all writes are spread uniformly over time.

Our new architectural solution, illustrated in Figure 13, can overcome the register writes hotspot by periodically changing the mapping of registers to their corresponding architectural hosting locations. The principal of the schemes is based on modulo rotation of the mapping between the architectural register identifier to their physical locations. As illustrated in Figure 13, a pulse trigger is asserted to shift the mapping of registers either periodically, or each time when we change CR3 or as part of the ROI (return form Interrupt) procedure before saving back the values of the user level process. A modulo-counter (RF rotator) is employed to map the architectural register number to the physical register location by modulo addition. Upon every rotation trigger assertion (in any arbitrary time point), the counter is incremented, and the register values needs to be shifted between registers as illustrated in Figure 13.



Figure 13 – EM-Aware RF mapping Scheme

## 4.3 EM Aware Cache

EM in cache structures involves hot sports in various cache lines that are spread in a non-uniform manner. It should be noted that in this subsection the term cache will refer to any architectural structure that employs a cache organization, e.g., TLBs, L1 cache etc. As a result, a small portion of cache lines will enforce the worst EM scenario on the entire cache. The principal of our new EM-aware cache memory scheme, illustrated in Figure 14, is similar to the register file solution. It can overcome the cache writes hotspots by periodically changing the cache set mapping of memory addresses to their corresponding physical cache lines. Similar to the RF solution, the principal of this schemes is based on modulo rotation of the mapping between the set field (taken from the memory address) to its physical set location. A pulse trigger is periodically asserted to shift the mapping of set. A modulo-counter (cache rotator) is employed to map the address set field to the physical set location by modulo addition. Upon every rotation trigger assertion, the counter is incremented, and all cache lines are invalidated as illustrated in Figure 14. In order to avoid the potential overhead incur as a result of flushing the content of the caches (and write back all of its dirty lines), we suggest doing the operation either in a very infrequent way or taking advantage of events that requites to flush these structures; e.g, after a sleep mode where all caches were cleaned.

****

Figure 14 - EM-aware Cache Memory Mapping

# 5. EM-Aware Architecture Experimental Study

In this section we present out experimental results for our novel architecture solution, that were presented in previous section, to relax EM impact. It should be noted that for all proposed techniques, no performance overhead was reported, so this section will focus on the impact of the new algorithms over the EM stress. We first examine EM-aware solution for ALU execution units. Figure 15 show the impact of the second method solution presented in previous section (refer to Algorithm 1) on the EM stress over the SPEC2017 benchmarks. We examined the 2 solutions and our observations indicate that they exhibit very similar behavior with negligible differences. As it can be observed, the new algorithm achieves a significant EM stress reduction of 50% over all benchmarks. The results vary from nearly 25% reduction up to 65% reduction. This is accomplished due to the fact that our scheme distributes the ALU utilization uniformly and as a result the maximum EM stress is smoothen over all ALUs.

As part of our study, we also compare the IPC (instruction per clock) versus EM stress reduction as illustrated in Figure 16. It can be observed that benchmarks with small IPC has a greater potential for EM stress utilization and this is due to the underutilized ALUs that could potentially help relaxing the maximum EM stress and distribute it uniformly.



Figure 15 - ALU Execution count distribution with EM-aware allocation



Figure 16 - ALU EM stress reduction vs. IPC

Our next results present the EM stress reduction that is accomplished by our architectural solution for both GPR register file and FP register file. These results are illustrated in Figure 17 and Figure 18 - FP Register writes distribution with EM-aware allocationFigure 18Figure 17 respectively. For both register files it can be observed that the number of writes is distributed uniformly over all registers and there are no longer hotspots observed (e.g. RAX or ZMM0). In addition, we observed a dramatic reduction in the write stress of nearly 80% on average for the GPR register file and 90% for the FP register file. The rotation trigger in our simulation was asserted every 10,000,000 clock cycles. In our experiment, we examined different rotation trigger rates value and found that this value yields an unnoticeable performance impact.



Figure 17 – General purpose register writes distribution with EM-aware allocation



Figure 18 - FP Register writes distribution with EM-aware allocation

As part of our EM study, we also observed that the flags and stack pointer registers experienced excessive stress of write operations and therefore they can be highly susceptible to EM. Figure 19 illustrates the number of write operations to the flags register, stack-pointer register and compares them to the maximum number of writes per register in the GPR register file. It can be observed that almost in all benchmarks the flags register has a significantly greater number of writes in respect to the GPR and stack-pointer registers. This is due to the fact that almost every computation instruction involves implicit write operation to the flag register. This observation motivates us to extend the EM-aware scheme that was proposed for the GPR register file and also include both the flag and stack-pointer registers. It can be observed in this figure that in this case we can accomplish even a higher reduction in the maximum number of write operations (EM stress) which varies from 80% till 90% and above.



Figure 19 - General purpose registers, flags and stack pointer write distribution with EM aware allocation

The last part of this section is devoted to examining the write stress reduction obtained for the TLBs and cache memories data lines and tags. Our experimental results are illustrated in Figure 20, Figure 21 and Figure 22, respectively. In most cases, we observe a significant reduction in the EM write stress. This reduction is obtained as a result of the repetitive rotation of the set mapping and the cache invalidation. Such rotation and invalidation actions contribute to distributing write operations uniformly over all sets and ways. For the D-TLB we suggest triggerring the rotation either when the TLB is flushed by the system, or by performing a period rotation; e.g., every 10M TLB accesses. As for and L1-D cache we suggest a similar periodic rotation trigger of 10M accesses. For all these options, we found that the performance overhead is minimal while the reduction of the EM stress is as indicated by Figure 20. As previously discussed, for both L2 and L3 we suggest triggering the set rotation upon every system wakeup from sleep mode. In this case, no performance overhead will be required. In our simulation we use an interval of 10M cache accesses, the same trigger duration of the L1-D cache for both L2 and L3 caches.

Figure 20 illustrates the writes stress reduction for DTLB. On average, over all benchmarks the write stress is reduced by 44%. Figure 21 summarizes EM write stress reduction for L1-D, L2 and L3 caches. For L1-D, L2 and L3 caches an average reduction in the maximum number of writes is 69%, 46% and 92% respectively. Cache tags EM stress reduction is summarized in Figure 22. In this case, the EM stress reduction is 28%, 46%, and 46% for L1-D, L2, and L3 caches respectively. It should be noted that the experimental results of our EM-aware architectural solution are correlated to the results presented in sub-section 4.3. These figures suggest that the lower the ratio of average number of write operations to the maximum of writes the higher EM stress reduction. This is explained by the fact that when the ratio of average number of write operations to the cache entry with the maximum number of writes is low, then the potential of EM stress reduction is higher.



Figure 20 - DTLB EM stress reduction



Figure 21 - Cache Lines EM stress reduction



Figure 22 - Cache tags EM stress reduction

# 6. Conclusions

Based on our experimental results, we conclude that a significant reduction can be accomplished various microprocess building blocks that are highly susceptible to EM: execution units, register files and the memory hierarchy. Our observations exhibit an average reduction in EM stress of 50% for ALUs, 80-90% for the register files and 46-92% in the data blocks of cache memories. These results indicate that with our proposed EM-aware solution, microprocessors designers may significantly relax the sign- toggling rate, and as a result, a significant number of potential EM violations are avoided. Alternatively, since EM is an accumulative phenomenon, the reduction in the total number of switching be translated to a lifetime extension of the device. As it was indicated in section 2, the MTTF is proportional to the switching rate and therefore a reduction of 50% in the switching rate will extend the lifetime by a factor of 2. These numbers, of course, depends on the workload being run by the microprocessor, and as it can be observed there are benchmarks where the reduction of EM stress is even much higher, e.g. in 600.perlbench the write reduction in the memory hierarchy is more that 70% where this may extend the overall lifetime by more than 3x. Still there are small number of benchmarks, e.g. 628.pop2, which exhibit smaller potential EM stress reduction in the range of 5-25% and as a results the overall lifetime extension that is gained is 5-33%.

Microprocessors reliability is a crucial requirement that introduces major micro-architectural and design challenges in advanced process nodes. In this study, we observed that microprocessors are highly suspectable to EM due to their nature of processing highly variable dynamic workloads while employing non-EM-aware micro-architectures. We introduced a novel architectural solution that takes into account EM effect and reduced excessing utilization of execution units and write operations to register files and the memory hierarchy elements. The principal of our solution is based on EM aware resource allocation mechanisms that smoothen write operation and computational elements utilization over all available resources. Our experimental result indicate that our novel architecture solution can significantly relax the EM sign-off conditions by 50% for ALUs, 80-90% for the register files and 46-92% in the data blocks of cache memories. In addition, we indicated since MTTF is proportional to the switching rate, we can translate these number to at least x2 lifetime extension. This, of course, depends on the behavior of the workload and there are benchmarks where the lifetime extension can be in a factor of 3 or even higher.

As EM has become a major challenge in advanced technologies, we further studies to continue exploring new architectures and studies to identify potential EM reduction and lifetime extension. In this study we examined the EM stress effect on modern micro-processors, though this work should be further extended to other processing elements like VLIW machines, DSPs, Network processors, Security engines, GPUs, TPUs and others.

**References**

[1] Jens Lienig. Electromigration and Its Impact on Physical Design in Future Technologies. Proceedings of the 2013 ACM International symposium on Physical Design, March 2013.

[2] I. A. Blech, Electromigration in thin aluminum films on titanium nitride, J. Appl. Phys., vol. 47 (1976), 1203–1208. <http://dx.doi.org/10.1063/1.322842>

[3] C. S. Hau-Riege, An introduction to Cu electromigration, Microel. Reliab., vol. 44 (2004), 195–205. DOI= <http://dx.doi.org/10.1016/j.microrel.2003.10.020>

[4] A. Scorzoni, B. Neri, C. Caprile, F. Fantini, Electromigration in thin- film inter-connection lines: models, methods and results, Material Science Reports, New York: Elsevier, vol. 7 (1991), 143–219. <http://dx.doi.org/10.1016/0920-2307(91)90005-8>

[5] D. Young, A. Christou, Failure mechanism models for electromigration, IEEE Trans. on Reliability, vol. 43(2) (June 1994), 186–192. DOI= <http://dx.doi.org/10.1109/24.294986>

[6] J. R. Black, “Electromigration – A brief survey and some recent results,” IEEE Trans. on Electronic Devices (April 1969), 338-347. DOI= <http://dx.doi.org/10.1109/T-ED.1969.16754>

[7] F.M. D’Heurle, Electromigration and Failure in Electronics: An Introduction, Proceedings of the IEEE, vol. 59, issue. 10, Oct. 1971, pp. 1409-1418.

[8] J. Srinivasan, S. V. Adve, P. Bose and J. A. Rivers. Lifetime Reliability: Toward an Architectural Solution. IEEE Micro, special issue on Emerging Trends, vol. 25, issue 3, May-June 2005, 2-12.

[9] J. Srinivasan, S. V. Adve, P. Bose and J. A. Rivers, The Case for Lifetime Reliability-Aware Microprocessors, Proceedings of 31st International Symposium on Computer Architecture (ISCA '04) June 2004.

[10] J. Srinivasan, S. V. Adve, P. Bose and J. A. Rivers. Exploiting Structural Duplication for Lifetime Reliability Enhancement. the Proceedings of the 32nd International Symposium on Computer Architecture (ISCA'05) June 2005.

[11] A. Limaye and T. Adegbija, “A workload characterization of the spec cpu2017 benchmark suite,” in 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 149–158, April 2018

[12] Qinzhe Wu, Steven Flolid, Shuang Song, Junyong Deng, Lizy K. John. Hot Regions in SPEC CPU2017. 2018 IEEE International Symposium on Workload Characterization (IISWC).

[13] A. Dasgupta and R. Karri, Electromigration Reliability Enhancement Via Bus Activity Distribution, Proc. 33rd Ann. Conf. Design Automation (DAC 96), ACM Press, 1996, pp. 353-356.

[14] Jaume Abella, Xavier Vera, Osman S. Unsal Oguz Ergin, Antonio Gonza ́lez and James W. Tschanz. Refueling: Preventing Wire Degradation due to Electromigration. IEEE Micro (Volume: 28 , Issue: 6 , Nov.-Dec. 2008 ).

[15] X. Xuan, Analysis and Design of Reliable Mixed-Signal CMOS Circuits, PhD thesis, Georgia Inst. of Technology, Dept. of Electrical and Computer Engineering, 2004.

[16] J. Lienig and G. Jerke, Embedded Tutorial: Electromigration-Aware Physical Design of Integrated Circuits, Proc. 18th Int’l Conf. VLSI Design (VLSID 05), IEEE Press, 2005, pp. 77-82.

[17] 4. J. Tao et al., Modeling and Characterization of Electromigration Failures under Bidirectional Current Stress, IEEE Trans. Electron Devices, vol. 43, no. 5, May 1996, pp. 800-808.

[18] J. Abella and X. Vera, Electromigration for Microarchitects. ACM Computing Surveys (CSUR)March 2010 Article No.: 9

[19] A. Valero, N. Miralaei, S. Petit, J. Sahuquillo, and T. M. Jones. On Microarchitectural Mechanisms for Cache Wearout Reduction. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 25, No. 3, March 2017.

[20] J. Lienig, Introduction to electromigration-aware physical design. In Proceedings of the International Symposium on Physical Design (ISPD’06). ACM, New York, 39–46.

[21] T. E. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. In Proceedings of the International Conference for High Performance Computing, Net- working, Storage and Analysis (SC), Nov. 2011.

[22] Operating Temperature, Wikipedia - <https://en.wikipedia.org/wiki/Operating_temperature>.

[23] Failure Mechanism based Stress test Qualification for Integrated Circuit. Automotive Electronics Council, Component Technical Committee - AEC - Q100 - REV-G standard. .

[24] Andrew B. Kahng, Siddhartha Nath and Tajana S. Rosing, On Potential Design Impacts of Electromigration Awareness. 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)

[25] Michael E. Thomadakis. The architecture of the Nehalem processor and Nehalem-EP smp platforms. Technical report, December 2010. http://sc.tamu.edu/systems/eos/nehalem.pdf.