TU Delft

Delft University of Technology

Hardware-based aging mitigation scheme for memory address decoder

Kraak, Daniel; Agbo, Innocent; Taouil, Mottaqiallah; Hamdioui, Said; Weckx, Pieter; Cosemans, Stefan; Catthoor, Francky

DOI
10.1109/ETS.2019.8791536

Publication date
2019

Document Version
Accepted author manuscript

Published in
2019 IEEE European Test Symposium (ETS)

Citation (APA)
Kraak, D., Agbo, I., Taouil, M., Hamdioui, S., Weckx, P., Cosemans, S., & Catthoor, F. (2019). Hardware-based aging mitigation scheme for memory address decoder. In *2019 IEEE European Test Symposium (ETS)* (pp. 1-6). IEEE . https://doi.org/10.1109/ETS.2019.8791536

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Hardware-Based Aging Mitigation Scheme for Memory Address Decoder

Daniël Kraak Innocent Agbo Mottaqiallah Taouil Pieter Weckx[1] Stefan Cosemans[1] Francky Catthoor[1,2]
Said Hamdioui
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands
{D.H.P.Kraak, I.O.Agbo, M.Taouil, S.Hamdioui}@tudelft.nl

[1]imec vzw., Kapeldreef 75, B-3001, Leuven, Belgium
[2]Katholieke Universiteit Leuven, ESAT, Belgium
{Pieter.Weckx, Francky.Catthoor}@imec.be

***Abstract*—Designers typically add design margins to memories to compensate for their aging. As the aging impact increases with technology scaling, bigger margins become necessary. However, this negatively impacts area, yield, performance, and power consumption. Alternatively, mitigation schemes can be used to reduce the impact of aging. This paper proposes a hardware-based mitigation scheme for the memory's address decoder logic. The scheme is based on adapting the decoder's workload during idle cycles by stressing the short paths and putting long paths into relaxation. Thanks to the adapted workload, the impact of aging on the address decoder is reduced, resulting in a more reliable memory. To validate the benefit of the mitigation scheme, the decoder's degradation of the L1 data and instruction caches for an ARM v8-a processor is analyzed. The experimental results show that the proposed mitigation scheme reduces the degradation of the decoder's timing margin with up to 4.1x at negligible area and no more than 3% power overhead.**

***Index Terms*—memory, address decoder, aging, mitigation**

I. Introduction

The aggressive downscaling of CMOS technology has been the main driver of the improved performance and functionality of Integrated Circuits (ICs) over the past decades. However, due to several challenges, the rate of downscaling and its benefits have started to decrease [1]. One of the drawbacks of downscaling is that it worsens the reliability of ICs due to increased time-dependent variability [2]; this consists of variations that occur during the operational lifetime of the IC. It includes environmental variations, such as supply voltage and temperature fluctuations, and *aging* variations due to, for example, Bias Temperature Instability (BTI) [3]. Traditionally, designers use guard-banding to tolerate this variability, meaning extra design margins are added to guarantee a reliable operation of the circuit. However, these extra margins negatively impact the area, performance, and power consumption of the design. Alternatively, designers can embed mitigation schemes into the design that reduce the impact of time-dependent variability. In this work, we investigate an aging mitigation scheme for the *address decoder* logic of memories. Several works report that *delay faults* in the decoder logic are a major contributor to the total amount of customer returns [4, 5]. These delay faults (e.g., due to aging) may cause *read* or *write failures*. Hence, understanding the impact of aging on the address decoder logic and providing appropriate mitigation schemes is an important part of designing reliable memories.

This work was supported through the projects TRACE (CATRENE, Grant 16ES0488K-16ES0502, 16ES0737) and PRYSTINE (ECSEL, Grant 783190).

Previous works have mainly focused on estimating and mitigating the impact of aging on the memory cell array [6–10]. Most of these works aim at balancing the probability of writing zeroes and ones to the memory cells, as this minimizes their degradation. In contrast to the memory cells, the peripheral circuitry has received significantly less attention. Most of the works have focused on estimating the impact of aging; examples are the sense amplifier [11] or the address decoder [12]. To our best knowledge, only one work exists on mitigation schemes for the peripheral circuitry [13]; it proposes a scheme for the sense amplifier that is based on periodically switching its inputs to create a balanced workload. Mitigation schemes for the other peripheral circuitry have not yet been researched. With respect to the address decoder, several hardware solutions have been proposed for detecting delay faults (e.g., due to aging) during run-time, such as [14, 15]. A shortcoming of these solutions is that they only detect faults, but do not mitigate them.

In this work, we propose a *hardware-based* mitigation scheme to reduce the impact of aging on the memory's address decoder; as already mentioned the address decoder is one of the critical components of the memory [4, 5]. The scheme is based on adapting the decoder's workload during idle cycles by stressing its short paths and putting long paths into relaxation. As a result, the impact of aging on the address decoder is reduced, resulting in a prolonged lifetime and improved reliability of the memory. In short, the contributions of this work are as follows:

1) It proposes a hardware-based aging mitigation scheme for the memory's address decoder that is based on adapting the workload during idle cycles.
2) It validates the superiority of the proposed scheme by investigating, as a case study, the aging impact of real applications on the L1 data and instruction caches of an ARM v8-a processor with and without mitigation.
3) It uses an *industrial-strength* $14\,\mathrm{nm}$ FinFET address decoder design and a *calibrated* aging model for the circuit simulations; hence, it provides accurate results.

The rest of this paper is organized as follows. Section II provides the background. Section III discusses the impact of aging on the address decoder. Section IV presents the proposed mitigation scheme. Section V presents a case-study to validate the mitigation scheme. Section VI provides a brief discussion. Finally, Section VII concludes this work.

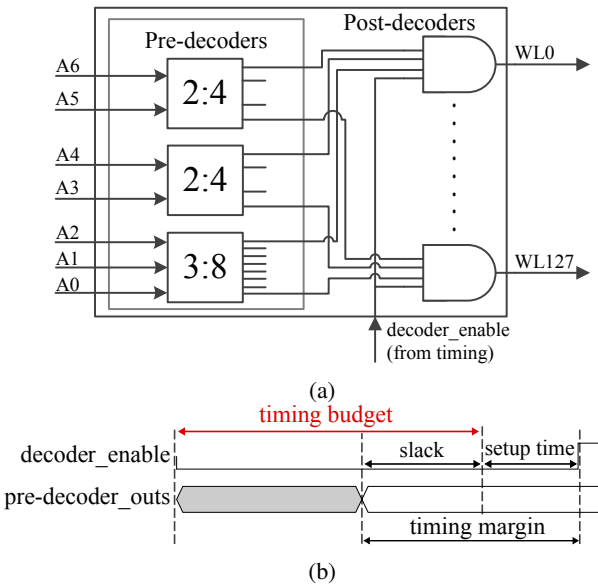


Fig. 1: (a) Schematic and (b) Slack metric of the decoder.

II. Background

We first discuss the memory's address decoder and its reliability metric and, subsequently, the Bias Temperature Instability aging mechanism (BTI); it is considered to be the most important aging mechanism in deeply scaled CMOS technologies [3] and, therefore, it is the focus of this work.

A. Address Decoder

Memories typically contain logic to access certain rows in the memory cell array, referred to as the *wordline decoder*, and logic to select certain columns, referred to as the *column decoder*. In general, the wordline decoder is more critical, as memories typically have more rows than (selectable) columns. Therefore, we limit our analysis to the wordline decoder for this study.

In this work, we consider a 7-to-128 wordline decoder design at *industrial strength*. Fig. 1a shows its simplified diagram. It is implemented using a hierarchical architecture that consists of a *pre-decoder* stage and a *post-decoder* stage. The pre-decoder stage is implemented using one 3-to-8 decoder and two 2-to-4 decoders. They have the address bits as their inputs. Fig. 2 shows the design of the 2-to-4 decoder. It is implemented using two inverters and four AND-gates. The inverters create inverted signals of the input address bits. Unique combinations of the original and inverted address bits are fed to the inputs of the AND-gates. This way, each input combination to the decoder results in one of its outputs becoming high. It is worth noting that the 3-to-8 decoder is implemented in a similar way. It uses, however, three inverters and eight AND-gates.

The post-decoder stage consists of 128 post-decoders that are implemented using AND-gates. Each post-decoder has as inputs a unique combination of the outputs from the pre-decoders and activates one of the wordlines of the memory cell array. An additional *enable_decoder* signal is connected

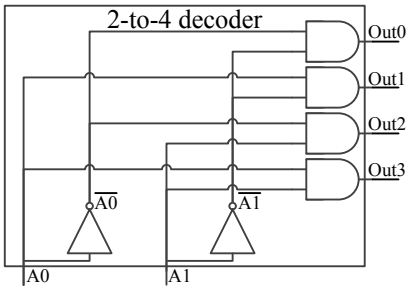


Fig. 2: Schematic of a 2-to-4 decoder.

to the input of these AND-gates to control the activation of the wordlines. In general, it is driven by the timing circuitry of the memory.

An important reliability metric of the wordline decoder is its *slack*; it is illustrated in Fig. 1b for our wordline decoder design. It is defined as the time between the outputs of the pre-decoders being stable and the *setup time* of the post-decoder's AND-gates. In case the delay of the pre-decoders is too high and the post-decoders' setup time is violated (i.e., a negative slack), a correct functionality of the post-decoders cannot be guaranteed; this may result in a delayed activation of the wordline, the selection of a wrong wordline, or the selection of multiple wordlines, potentially causing a *read failure* or a *write failure*. Hence, a higher slack corresponds to a more reliable operation of the memory.

B. Bias Temperature Instability

The BTI failure mechanism takes place inside the MOS transistors and causes an increment in their absolute threshold voltage (V_{th}). This happens under *negative gate stress* for PMOS transistors, referred to as Negative BTI (NBTI). For NMOS transistors this happens under *positive gate stress*, which is referred to as Positive BTI (PBTI).

BTI is a threat for the reliability of digital circuits, such as sequential logic or (memory) address decoders. This is due to the fact that the increase of the transistor's threshold voltage results in increased delays of their logic gates. This causes in turn increased path delays. In case BTI is not taken into account during design or not mitigated during operation, logic paths may exceed their timing budget (e.g., determined by the clock period for sequential logic or an enable signal in memories). This results in *timing violations*, which may cause *field failures*.

This paper uses the atomistic model presented in [16] to model BTI. It is based on the capture and emission of traps during stress and relaxation periods, respectively. Each occupied trap contributes to the total threshold voltage shift ΔV_{th}. The probabilities of the capture P_C and emission P_E of traps are defined by [17] as follows:

$$P_C(t_{STRESS}) = \frac{\tau_e}{\tau_c+\tau_e}\left\{1 - exp\left[-(\frac{1}{\tau_e}+\frac{1}{\tau_c})t_{STRESS}\right]\right\} \quad (1)$$

$$P_E(t_{RELAX}) = \frac{\tau_c}{\tau_c+\tau_e}\left\{1 - exp\left[-(\frac{1}{\tau_e}+\frac{1}{\tau_c})t_{RELAX}\right]\right\} \quad (2)$$

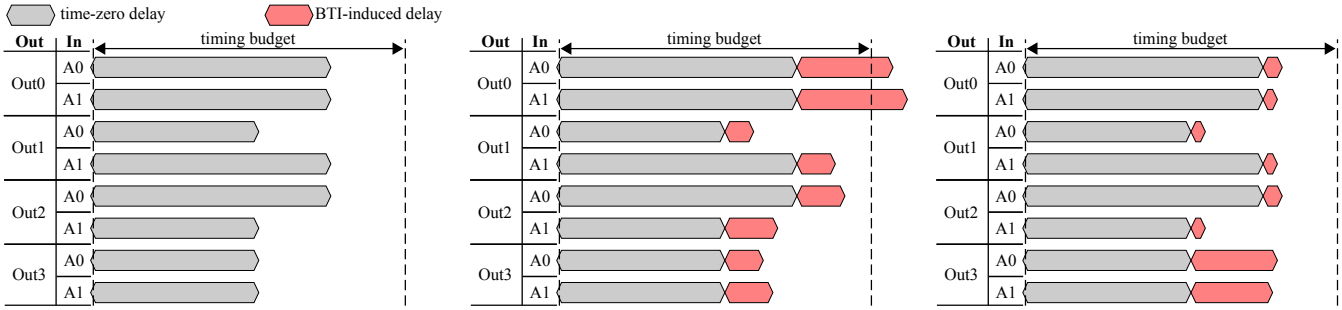


Fig. 3: (a) Activation path delays of 2-to-4 decoder. (b) Impact of aging. (c) Projected degradation with mitigation.

Here, τ_c is the mean capture time constant, τ_e the mean emission time constant, t_{STRESS} the stress period, and t_{RELAX} the relaxation period. The model also incorporates the impact of voltage and temperature [16].

III. Aging in Address Decoders

As discussed in the previous section, an important reliability metric of the memory address decoder is its *slack*. It is the time between the outputs of the pre-decoders being stable and the *setup time* of the post-decoder's AND-gates. Hence, the delay of the pre-decoders is crucial for this metric. One of the main challenges for the pre-decoders is that they need to drive long wires due to the fact that the post-decoders are divided across the whole memory array. Hence, they need to drive a high parasitic capacitance due to this wiring. Driving this capacitance takes a considerate portion of the total path delays of the pre-decoders. For our design, the delays to activate the pre-decoder outputs (i.e., drive them high) are the highest. This is due to the fact that PMOS transistors are weaker than NMOS transistors. Moreover, the PMOS transistors are more sensitive to BTI than the NMOS transistors [3]. Therefore, the paths that activate the pre-decoder outputs (in contrast to the deactivation paths) are the most likely to cause delay faults due to aging.

To illustrate potential problems due to the activation delay of the pre-decoders' outputs, we take the 2-to-4 decoder from Fig. 2 as an example. The activation path delays at time-zero are illustrated for each output in Fig 3a. Note that these are based on SPICE simulations. Each output can be activated through a path that starts either at input bit $A0$ or $A1$. Depending on the responsible input, this may result in a different delay. The figure reveals that outputs $Out0$, $Out1$, and $Out2$ each have at least one path with a high delay and, thus, a low slack. Hence, workloads that favor the activation of one of these outputs are the most likely to cause aging-induced delay faults, as these will stress paths with the low slack the most. An example of such a case is depicted in Fig. 3b; here, the BTI-induced delays are illustrated for the case in which a workload favors the activation of output $Out0$. As can be seen, the paths to activate this output have the biggest BTI-induced delays, which causes timing violations (i.e., a negative slack).

Outputs $Out0$, $Out1$, and $Out2$ each have a high maximum path delay due to the fact that at least one of the inputs propagates through an inverter. For example, this is the case for output $Out1$, when it is activated through input $A1$. On the contrary, the last output $Out3$ has lower delays for both inputs. This is due to the fact that none of the inputs need to propagate through an inverter to activate this output.

It is worth noting that this observed trend also applies for pre-decoders with different sizes. For example, the first seven outputs of a 3-to-8 decoder have a high maximum activation path delay, as at least one of the inputs needs to propagate through an inverter. The last output, however, has a low delay, as none of its inputs need to propagate through an inverter.

IV. Proposed Mitigation Scheme

Typically, in case a memory is idle, the input address of the address decoder is kept unchanged compared to the applied address during the last operation. Hence, in case an application frequently keeps pre-decoder outputs with long paths activated during idle cycles, this may significantly contribute to the decoder's degradation. Note that although the address decoder's enable signal is not activated during idle cycles, the pre-decoders are, nevertheless, still active and stressed. Hence, to mitigate this aging, we propose to utilize these idle cycles by applying addresses that activate pre-decoder outputs with short paths. This ensures that short paths are stressed during idle cycles and, hence, the long paths are put into relaxation. As we observed previously, the last output of a pre-decoder has the lowest path delays and, therefore, this is a good candidate for mitigation. This observation applies to all the pre-decoders (e.g., also the 3-to-8 decoder). The effect of this mitigation scheme for the 2-to-4 decoder is illustrated in Fig. 3c; here, the paths that activate $Out3$ have the highest BTI-induced delays, as they are stressed the most. However, due to the fact that its time-zero delays are the lowest, the BTI-induced delays are *masked*. As a result, the aging has a lower impact on the decoder's slack.

The proposed mitigation scheme can be implemented, for example, with an additional multiplexer that is placed before the address flip-flops. This multiplexer is used to select between the original address coming from the host (e.g., a CPU) and a *mitigation address* that stresses the short paths (and, hence, puts the long paths into relaxation). All bits of this mitigation address are high (hardwired), as this ensures that the last outputs of all pre-decoders are activated. The select

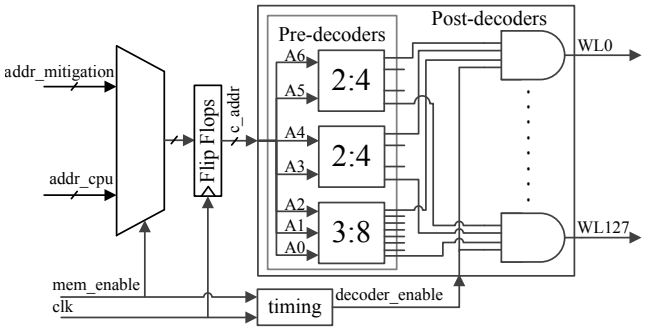


Fig. 4: Address decoder mitigation scheme.

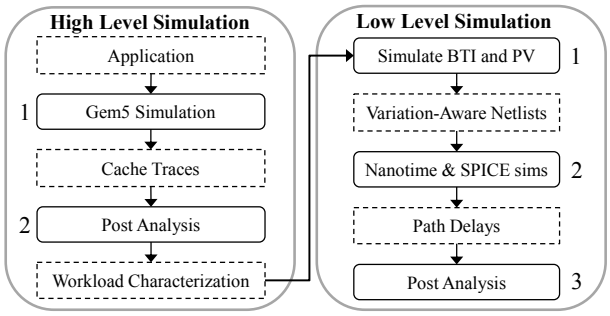


Fig. 5: Used Simulation Flow [19].

signal of the multiplexer is connected to the memory's enable signal, so the mitigation address is applied to the address decoder during idle cycles. Note that in this implementation, the address decoder itself is not modified and only surrounding logic is added.

V. Validation via a Case Study

In this section, we perform a case-study to validate the mitigation scheme.

A. Experimental Setup

We implemented the mitigation scheme for the decoder design from Fig. 1a. Its structure is depicted in Fig. 4. A multiplexer has been placed in front of the address flip-flops. This multiplexer is controlled by *mem_enable* (the memory's enable signal). When *mem_enable* is low, the mitigation address is stored into the flip-flops and when it is high, the address from the CPU is stored. Hence, the mitigation address is applied to the decoder during idle cycles. For a fair analysis, the part of the memory's timing circuit that generates the enable signal for the decoder is also included. The reason for this is that aging of the timing circuit delays this enable signal and, thus, it partially compensates for the decoder's aging [18].

To investigate the impact of aging on the decoder, we use the methodology presented in [19]. It is able to accurately and efficiently analyze the impact of aging on the memory's digital logic (e.g., address decoder and timing circuit) for real applications. Its flow is depicted in Fig. 5. As the figure shows, it consists of a high level and a low level simulation part. In the high level simulation part, the workload of the input application is characterized. In the first step, a CPU architecture is simulated using gem5 simulator [20]. During this simulation, traces of the caches (e.g., L1 data and instruction caches) are created that contain all memory operations for each cycle. Next, post analysis is performed on these traces to generate a workload characterization [19]. This workload characterization contains the duty factors of the gate signals for each transistor in the memory logic. In the low level simulation part, this workload characterization is used to evaluate the impact of aging on the circuit under analysis. In the first step, variation-aware netlists are generated by performing Monte Carlo simulations in which BTI (using the model discussed in Section II) and process variation (using Pelgrom's model [21]) are simulated. Subsequently, the paths of these variation-aware netlists are extracted into SPICE netlists using Nanotime [22], which are then simulated in SPICE to measure their delays. Finally, post analysis is performed on the measured path delays. In our analysis, the worst-case path delay of the pre-decoders (i.e., the critical path delay) is determined for each Monte Carlo instance.

TABLE I: Gem5 configuration.

Processor	ARM v8-a, single-core, out-of-order @ 1.8GHz
L1 Data & Instruction Cache	32 kB, 4-way set associative, 32 B linesize
L2 Cache	1 MB, 8-way set associative, 32 B linesize

Using the flow from Fig. 5, we investigate the impact of aging on the decoder for the L1 data and instruction caches with and without mitigation. In the high level simulation part, we simulate six different applications from the SPEC2006 Benchmark suite [23] on an ARM v-8a processor in gem5. Details of the gem5 configuration can be found in Table I. In order to make the simulations feasible, we simulate a sample of one billion instructions per application. In the gem5 simulations, we assume that the data and tag sections of each cache set are implemented using separate memories. Hence, two memory traces are generated per cache set (one for the data section and one for the tag section). Due to similar accesses to the data and tag sections within a set and also to the different sets within the cache, we limit our analysis to the first set's data section. In the low level simulations, we assume that the wordline decoder and timing circuit of this data section are implemented using the circuit from Fig. 4. Note that for the analysis without mitigation, the multiplexer is omitted and, hence, the last requested memory address stays applied to the decoder during idle cycles.

B. Performed Experiments and Used Metrics

Using the experimental setup described above, the following experiments are performed to investigate the impact of aging on the decoder with and without the mitigation scheme:

1) **Application dependency**: we investigate the impact of aging on the decoder of the L1 data cache for different applications.
2) **Cache dependency:** we investigate the impact of aging on the decoder of the L1 instruction cache for different applications as well and compare it with the degradation of the L1 data cache.

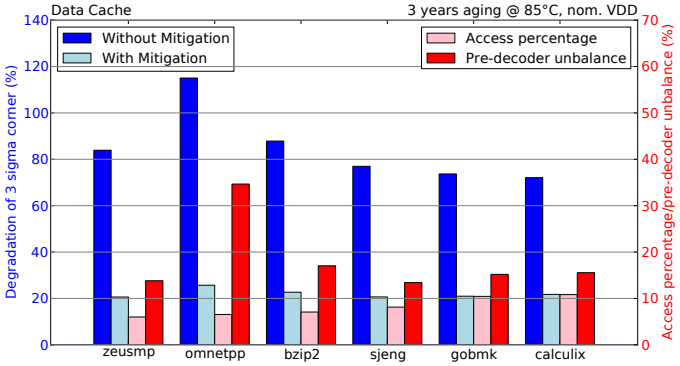


Fig. 6: Degradation of the data cache's timing margin.

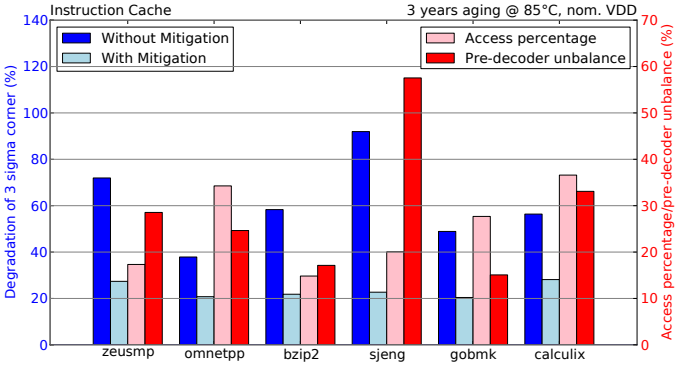


Fig. 7: Degradation of the instruction cache's timing margin.

For each application, the following metrics are measured:

- **Timing Margin**: it is the sum of the slack and setup time of the worldline decoder, as illustrated in Fig. 1. Note that the setup time is typically significantly lower than the slack. Hence, the timing margin is a representation of the slack.
- **Access Percentage**: it is the average percentage of performed memory operations (read and write) for the whole application.
- **Pre-decoder unbalance**: it is the percentage of time that a critical path of the 3-to-8 decoder (i.e., a path activated through an inverter) is stressed by the application; here, we consider the most stressed path. The 3-to-8 decoder is used, as it has longer paths than the 2-to-4 decoders and, thus, it is the dominating circuit for the decoder's degradation.

For each experiment, 1,000 Monte Carlo simulations are performed per application/workload. We assume that the decoder is aged for three years at 85°C (junction temperature) with a nominal supply voltage of $0.8\,\mathrm{V}$. For each set of Monte Carlo simulations, the $3\,\sigma$ corner of the timing margin is calculated based on the average and spread of the measured timing margins. These are then compared with the $3\,\sigma$ corner at time-zero (which is only affected by process variation).

C. Experimental Results

Application Dependency

Fig. 6 shows the degradation of the $3\,\sigma$ corner of the timing margin for the L1 data cache for different applications. The degradation is shown without and with mitigation. In addition, the *access percentage* and *pre-decoder unbalance* metrics are shown. The scale of these two metrics is shown on the right axis. The following observations can be made from the figure:

- The decoder's degradation is strongly application-dependent (without mitigation). The lowest degradation of the timing margin is ∼72% (for 'calculix'), while the highest degradation is ∼115% (for 'omnetpp').
- The strong application-dependency of the decoder's degradation (without mitigation) is mainly caused by the different amount of accesses between applications. Generally, a higher *access percentage* corresponds to a lower degradation. For example, 'zeusmp' (access percentage of ∼6%) has a degradation of ∼84%, while 'calculix' (access percentage of ∼11%) has a degradation of ∼72%. A higher *access percentage* results in a lower degradation, because the timing circuit is activated more frequently and, thus, stressed more. As a result, the *decoder_enable* signal from the timing circuit is delayed, which compensates for the decoder's degradation.
  In addition, applications with a higher *pre-decoder unbalance* have a higher degradation. For example, 'omnetpp' has the highest degradation (∼115%), while it does not have the lowest amount of accesses. A higher *pre-decoder unbalance* means that at least one of the 3-to-8 pre-decoder's critical activation paths is stressed more and, thus, the timing margin has a higher degradation.
- The mitigation scheme significantly reduces the decoder's degradation (up to 4.5x). The highest observed degradation is only ∼26% with mitigation, while this is ∼115% without mitigation. This shows the *superiority* of the mitigation scheme.
- The decoder degradation mainly depends on the *pre-decoder unbalance* instead of the *access percentage* when mitigation is applied. For instance, 'omnetpp' (pre-decoder unbalance of ∼35%) has a degradation of ∼26%, while 'sjeng' (pre-decoder unbalance of ∼13.41%) has a degradation of ∼21%. The *access percentage* does not have a high impact anymore due to the fact that the critical paths of the pre-decoders are only stressed during memory operations, similarly to the timing circuit, with the mitigation scheme. Hence, a higher *access percentage* no longer compensates the aging of the decoder as much as without mitigation.

Cache Dependency

Fig. 7 shows the degradation of the $3\,\sigma$ corner of the timing margin for the L1 instruction cache for the six applications. Comparing the degradation of the instruction cache with that of the data cache (i.e., Fig. 6) reveals the following:

- The data cache, typically, has a higher degradation than the instruction cache (without mitigation). The degradation is between ∼72% and ∼115% for the data cache, while it is between ∼38% and ∼92% for the instruction cache. The data cache has a higher degradation, because it has a lower *access percentage* (between ∼6% and ∼11%) than the instruction cache (between ∼15% and ∼37%). Therefore, the timing circuit receives a lower stress and, hence, the decoder's degradation is compensated less.

- Similarly to the data cache, the mitigation scheme also significantly reduces the degradation of the instruction cache (up to 4x). After applying the mitigation scheme, the instruction cache, however, has a slightly higher degradation than the data cache. The highest degradation is ∼28% for the instruction cache, while it is ∼26% for the data cache. This is because most applications have a higher *pre-decoder unbalance* for the instruction cache. Therefore, critical paths of the instruction cache's pre-decoders are stressed more.

VI. Discussion

This work proposes a hardware-based aging mitigation scheme for the memory's address decoder logic. Based on our case-study, we conclude the following:

Improved reliability: our case study reveals that the mitigation scheme is able to significantly reduce the degradation of the decoder's timing margin. The degradation is up to 115% without mitigation, while it is up to 28% with mitigation; a reduction of ∼4.1x. Thus, the mitigation scheme achieves a more reliable address decoder and, hence, it leads to a memory with a *higher reliability* and *prolonged lifetime*.

Applicability of the scheme: since the proposed scheme is applied during idle memory cycles, it is mainly useful for applications with a relatively low amount of memory operations, such as CPU caches. Our simulation results show, however, that the decoder has a lower degradation for applications with a high number of memory operations due to the fact that the increased aging of the timing circuit compensates the decoder's aging. Hence, these applications also have a lower need for the proposed mitigation scheme.

Costs of the scheme: the cost of our scheme in terms of area, performance (i.e., delay), and power should be considered. First, the area overhead is negligible; the scheme uses only one additional multiplexer, which has a negligible area overhead compared to the cell array. Second, the performance overhead is also negligible; it does not affect the memory's speed, as the multiplexer is placed outside the memory. It does add a marginal delay to the logic that interfaces with the memory. However, this delay can most likely be masked by retiming the pipeline stages. Finally, the scheme causes an increased power consumption, because it induces extra switching in the decoder logic: when the memory starts an idle cycle, the decoder switches, and when it starts a new operation, it may switch again depending on the applied address. To evaluate the power overhead, we implemented the scheme for a memory with the same size as the cache set considered in the case-study. Subsequently, we measured the energy of the worst-case switching of the decoder (all address bits need to flip when the idle cycle starts and they flip again when the next operation starts) and compared it to the energy of a read and write operation. Our results show that this worst-case decoder switching energy is only ∼3% and ∼2.5% of the energy required for the read and write operations, respectively. It is low due to the fact that the wordline is not activated and only the pre-decoders switch. Hence, the power overhead of our scheme is acceptable.

Hence, overall, the proposed scheme comes at very marginal overheads.

VII. Conclusion

This work presented a mitigation scheme to reduce the impact of aging on memory address decoders. The scheme is based on adapting the decoder's workload during idle cycles. Hence, it may be possible to apply the same concept to other electronic components with idle cycles as well in order to mitigate their aging.

References

[1] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, July 1999.

[2] J. Srinivasan, S.V. Adve *et al.*, "The impact of technology scaling on lifetime reliability," in *International Conference on Dependable Systems and Networks, 2004*, June 2004, pp. 177–186.

[3] S. Bhardwaj, W. Wang *et al.*, "Predictive modeling of the nbti effect for reliable design," in *CICC*, Sept 2006, pp. 189–192.

[4] W. Needham, C. Prunty, and E.H. Yeoh, "High volume microprocessor test escapes, an analysis of defects our tests are missing," in *Proceedings International Test Conference 1998*, Oct 1998, pp. 25–34.

[5] A.J. van de Goor, S. Hamdioui, and R. Wadsworth, "Detecting faults in the peripheral circuits and an evaluation of sram tests," in *2004 International Conference on Test*, Oct 2004, pp. 114–123.

[6] S.V. Kumar, K.H. Kim, and S.S. Sapatnekar, "Impact of nbti on sram read stability and design for reliability," in *7th International Symposium on Quality Electronic Design (ISQED'06)*, March 2006, pp. 6 pp.–218.

[7] A. Carlson, "Mechanism of increase in sram v_{min} due to negative-bias temperature instability," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 3, pp. 473–478, Sept 2007.

[8] S. Khan, I. Agbo *et al.*, "Bias temperature instability analysis of finfet based sram cells," in *DATE*, March 2014, pp. 1–6.

[9] A. Gebregiorgis, M. Ebrahimi *et al.*, "Aging mitigation in memory arrays using self-controlled bit-flipping technique," in *The 20th Asia and South Pacific Design Automation Conference*, Jan 2015, pp. 231–236.

[10] A. Valero, N. Miralaei *et al.*, "On microarchitectural mechanisms for cache wearout reduction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 857–871, March 2017.

[11] I. Agbo, M. Taouil *et al.*, "Integral impact of bti, pvt variation, and workload on sram sense amplifier," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1444–1454, April 2017.

[12] S. Khan, M. Taouil *et al.*, "Impact of partial resistive defects and bias temperature instability on sram decoder reliablity," in *2013 8th IEEE Design and Test Symposium*, Dec 2013, pp. 1–6.

[13] D. Kraak, I. Agbo *et al.*, "Mitigation of sense amplifier degradation using input switching," in *DATE*, March 2017, pp. 858–863.

[14] P.S. Hughes, "Detection of address decoder faults," U.S. Patent US20 090 037 782A1, Feb. 05, 2009.

[15] R. Ramaraju and A.B. Hoefler, "Word line fault detection," U.S. Patent US8 379 468B2, Feb. 19, 2013.

[16] B. Kaczer, T. Grasser *et al.*, "Origin of nbti variability in deeply scaled pfets," in *IRPS*, May 2010, pp. 26–32.

[17] M. Toledano-Luque, B. Kaczer *et al.*, "Response of a single trap to ac negative bias temperature stress," in *IRPS*, April 2011, pp. 4A.2.1–8.

[18] D. Kraak, I. Agbo *et al.*, "Degradation analysis of high performance 14nm finfet sram," in *DATE*, March 2018, pp. 201–206.

[19] ——, "Methodology for application-dependent degradation analysis of memory timing," in *DATE*, March 2019, In press.

[20] N. Binkert, B. Beckmann *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, pp. 1–7, 08 2011.

[21] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers, "Matching properties of mos transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, Oct 1989.

[22] Synopsys, "Nanotime - transistor-level static timing analysis solution for custom designs," https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/nanotime-ds.pdf.

[23] J.L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.