ArISE: Aging-aware Instruction Set Encoding for Lifetime Improvement

Fabian Oboril Mehdi Tahoori
Karlsruhe Institute of Technology (KIT)
Chair of Dependable Nano Computing (CDNC)
Karlsruhe, Germany
e-mails: {Fabian.Oboril,Mehdi.Tahoori}@kit.edu

Abstract—Microprocessors fabricated at nanoscale nodes are exposed to accelerated transistor aging due to Bias Temperature Instability and Hot Carrier Injection. As a result, device delays increase over time reducing the Mean Time To Failure (MTTF) of the processor. To address this challenge, many (micro)-architectural techniques target the execution stage of the instruction pipeline, as this one is typically most critical. However, also the decoding stages can become aging-critical and limit the microprocessor lifetime, as we will show in this work. In this paper, we propose a novel aging-aware instruction set encoding methodology (ArISE), that improves the instruction encoding iteratively using a heuristic algorithm. Our experimental results show that MTTF of the decoding stages can be improved by 1.93x with negligible implementation costs.

I. Introduction

With the continuous downscaling of CMOS technology into nanoscale dimensions, reliability has emerged as an important design constraint besides performance, power and cost [3]. Among various reliability issues, accelerated transistor aging mainly caused by *Bias Temperature Instability (BTI)* [17, 22] and *Hot Carrier Injection (HCI)* [19] is a major challenge. Both phenomena manifest themselves in increasing switching and path delays and eventually cause faster wearout of the system. As a result *Mean Time To Failure (MTTF)* is reduced and timing violations can occur in the field. To avoid these, designers add safety margins to their designs to ensure a certain operational lifetime. However, such an overdesign is very expensive [13]. Hence, new aging-aware design techniques are necessary to take further advantage of scaled technology nodes in terms of performance, power, area and reliability.

As the microprocessor lifetime is mainly determined by the execution units [8], various aging-aware solutions have been proposed to extend the lifetime of these units. However, many of these techniques such as aging-aware instruction scheduling [15] or aging-aware NOP instructions [10] are hardly applicable to other stages of the instruction pipeline. Hence, by using such techniques and extending the lifetime of execution stages, the impact of other near-aging-critical stages on the overall microprocessor lifetime becomes more pronounced. As a result, MTTF improvements for the entire microprocessor become much smaller than the individual benefits for the execution units. Therefore, additional approaches are necessary to also improve MTTF of other pipeline stages. In particular the *decoding stages* have to be targeted, since these suffer from high wearout rates and are almost as critical as or even more critical than the execution stages, as we will show in this work. However, such approaches to improve lifetime of pipeline stages in the frontend are still very rare.

To close this gap, we propose a novel aging-aware instruction set encoding technique[1] *called ArISE.* This technique exploits the fact that the instruction set encoding (ISE) has a considerable impact on wearout of the decoding stages, since it affects the input patterns (via the applied opcodes) and the gate-level implementation of these stages, which both influence wearout. To find a good instruction set encoding in terms of MTTF, our approach uses a hierarchical optimization algorithm based on simulated annealing to obtain an aging-aware opcode for each instruction in such a way, that the overall lifetime of the decoding stages is improved. Thereby, only the representing bit patterns are modified, while the opcode length remains unchanged. Together with existing aging mitigation techniques for other stages, this approach can help to significantly improve the overall microprocessor lifetime.

The results show that the proposed technique can improve MTTF[2] of the decoding stages of the FabScalar microprocessor [7] by 1.93x, while other pipeline stages MTTF are not (negatively) affected. In addition, performance is not impaired and the area costs are negligible.

In summary, the key contributions of this work are:

- We propose and evaluate an aging-aware ISE to increase MTTF of the decoding stages.
- We present a generic flow to obtain an aging-aware ISE

The rest of this paper is organized as follows. In Section II the considered transistor aging phenomena, BTI and HCI, are introduced followed by a discussion of related work. The novel ArISE approach is motivated in Section III and the methodology to obtain such an encoding is detailed in Section IV. Afterwards, we present in Section V our experimental results. Finally, Section VI concludes the paper.

II. Preliminaries

BTI and HCI impair the threshold voltage of transistors, which results in longer path delays. This effect is explained in this section, followed by a discussion of related work.

A. Transistor Aging

A.1 Hot Carrier Injection (HCI)

HCI is a wearout mechanism of NMOS transistors. When accelerated channel-electrons collide with the gate oxide interface electron-hole pairs are created and free electrons get trapped inside the oxide layer. As a result, the threshold volt-

[1] The mapping of instructions (e.g. add) to their binary representation (e.g. 1101), i.e. opcode, is called Instruction Set Encoding.

[2] In this work MTTF is the time until first timing violation occurs

age V_{th} increases irreversibly [22]. The magnitude of the V_{th}-shift has an exponential relation with temperature [5] and depends also on the number of transitions [19] (i.e. clock frequency, runtime and switching activity), as energetic electrons are only generated during transitions. To estimate the resulting V_{th}-shift, the model from [19] is used.

A.2 Bias Temperature Instability (BTI)

BTI consists of *positive* and *negative* BTI, affecting NMOS and PMOS transistors, respectively [17]. When the gate-source of a PMOS (NMOS) transistor is negatively (positively) biased, $|V_{th}|$ increases due to the creation of traps at the interface between gate oxide and channel as well as inside the gate oxide (*stress* phase). When the gate bias is removed, some of the previously generated traps are filled, which results in a decreasing $|V_{th}|$ (*recovery* phase). However, the initial shift cannot be entirely compensated leading to a gradual increase of $|V_{th}|$ over time. The wearout rate depends on several aspects, such as temperature and duty cycle, i.e the ratio of stress to total time. To estimate the overall V_{th}-shift, the model from [22] is used.

B. Related Work

B.1 Aging Mitigation

To alleviate transistor aging several design techniques have been already proposed of which we name just a few ones. Device and circuit-level techniques such as gate sizing [24], V_{th}-tuning [20], input vector control [23], path balancing [9] and stacking-based pin reordering [20] are orthogonal to our work and can be used together with our method.

At (micro)-architecture-level most techniques address transistor aging for the execution units of a microprocessor, as these are typically the lifetime-limiting factors [8]. In [15, 18] aging-aware instruction scheduling techniques are evaluated and in [10] a special NOP-instruction is used to alleviate the impact of NBTI on an ALU. Also some techniques address wearout in memories. [14] uses cell-flipping to make duty cycles close to 0.5 and in [21] BTI-induced aging in a register-file is mitigated by flipping the leading bits of narrow-width values periodically.

In summary, although these (micro)-architecture-level techniques can increase MTTF of memory elements and execution units considerably, since other stages of the instruction pipeline do not benefit from these approaches, the lifetime of the entire microprocessor will be limited. Hence, aging mitigation techniques targeting other stages, particularly the decoding stages, are necessary. Besides this work, this issue is also targeted in [11]. The authors proposed to periodically invert the instruction opcode to make the signal duty cycles close to 0.5, which should mitigate transistor aging due to BTI. However, HCI is not addressed and the overall MTTF improvements are much lower than those of our proposed approach (see Section V).

B.2 Instruction Set Encoding (ISE)

Special ISEs are often applied to reduce the dynamic power consumption of instruction buffers and registers [1, 6] by minimizing the input switching activity. Although we also propose ISE modifications, our approach is orthogonal to these techniques, since we target logic circuits instead of memory, and reliability (lifetime) instead of power. As a consequence, power can be impacted. For example we observed a 14 % higher switching activity at the inputs of the instruction buffer (SRAM-based) for the aging-aware ISE (see Sec. V). This shows that, although power and wearout are coupled (via temperature), they require different optimization strategies. Hence, both approaches can be combined to achieve a good trade-off between power and MTTF.

III. Motivation and Main Idea

As mentioned earlier pipeline stage delays increase considerably during runtime due to BTI and HCI. Thereby, the input stream of each stage has a significant influence on the aging rates, as it affects duty cycle and switching activity of internal signals. Hence, also the instruction opcodes as part of this stream contribute to the wearout of the affected stages. To underline this fact and motivate this work, we use the FabScalar[3] microprocessor as our case study [7].

To evaluate wearout of different stages for different ISEs, we extracted the delay at design time and after 3 years for each stage using the accurate gate-level aging-estimation flow detailed in Section IV.E in combination with the setup given in Section V and SPEC2000 benchmarks.

The results of this analysis are depicted in Fig. 1. Obviously ISE has a significant influence on the wearout of frontend stages, especially the decoding stages (Predecode and Decode). The effect on other stages is in contrast negligible. This is because the instruction opcode forms only a small part of the inputs of the other stages (or is no input at all). This difference in delay degradation rates for the Predecode stage translates to more than 2x difference in MTTF. Moreover, the results also show that the decoding stages can even become lifetime-limiting, if the ISE is not designed aging-aware, as shown in Fig. 1 (see Encoding 3). This underlines the need for using an aging-aware instruction encoding.

Improving the ISE is a challenging task, since many instruction encodings have to be modified to become aging-aware. For example, in case of the FabScalar microprocessor the instruction set architecture contains 131 instructions that are encoded using 8 bits. This means that there are $(2^8)!/(2^8 - 131)! \approx 10^{297}$ encoding possibilities. Since most encodings infer modifications in the gate-level implementation of the stages and affect signal duty cycles as well as switching activities, each encoding requires a new synthesis and simulation flow for aging analysis. Moreover, instructions cannot be considered in-

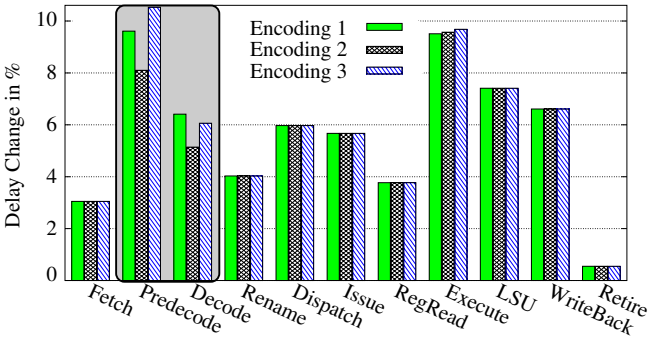

Fig. 1. Worst-case delay change after 3 years for different pipeline stages for 3 different ISEs (obtained with setup detailed in Section V)

[3] FabScalar is an out-of-order, 11-stage, superscalar processor

dependently, since duty cycles and switching activities depend not only on the actual instruction, but also on the preceding and subsequent instructions. Therefore, to see the effect of a particular encoding, or even optimizing the encoding for just one instruction, one has to simulate instruction streams. Due to this complexity an exhaustive method is infeasible. Hence, we use a hierarchical heuristic methodology based on simulated annealing to find an aging-aware ISE as described next.

IV. ARISE: AGING-AWARE INSTRUCTION SET ENCODING

In this section the flow to create an aging-aware ISE considering the overall MTTF of the decoding stages is explained (IV.A) and a hierarchical approach to improve its efficiency is discussed (IV.B). Furthermore, a runtime analysis and further improvements are presented in part IV.C. The application of a modified ISE to a real system is described in IV.D and the aging estimation flow is detailed in Section IV.E.

A. Optimization Algorithm

The starting point of the optimization algorithm, which is shown in Fig. 2, is a random ISE (here: FabScalar's standard ISE). For this ISE all pipeline stages have to be synthesized and their MTTF values need to be extracted according to the flow presented in Section IV.E (Steps 1+2). Afterwards, a simulated annealing (SA) algorithm is invoked (Step 3). As a first step it generates a "neighbor" ISE for the current one (Step 3.1). For this new ISE the overall MTTF considering the decoding stages is estimated (Step 3.2). As neighbor definition we use in this work the following principal. Two ISEs are neighbors if and only if a) both ISEs differ only in one instruction opcode (only possible if there are more binary opcodes than instructions), or b) the second ISE can be derived from the first one by exchanging the opcodes of two instructions. Using this definition it is guaranteed that every possible ISE can be evaluated and that the difference between the new and the old ISE is not that huge that the optimization process is too random.

The next step is to evaluate if the new ISE will replace the old one (Step 3.3). Therefore we applied the exponential function detailed in Equation (1) as cost function:

$$P(\text{accept ISE}) = \exp\left(-(D_{new} - D_{old})/T\right) \overset{?}{>} P_{reject}. \quad (1)$$

The inputs for the cost function are the worst case delay (which can be used to represent MTTF) after 3 years for the new and old ISE. P_{reject} is the reject probability and T the annealing temperature, which can be iteratively reduced (Step 3.0) according to the SA principal. Since SA tries to find the global optimum, it does not only accept solutions with lower costs (i.e. better MTTF) but also intermediate solutions with higher costs to avoid local optima. In both cases a neighbor for the new ISE is created (Step 3.3.2) and the evaluation continues with this new neighbor. Otherwise, if the costs for the new ISE are too high, a new neighbor for the old ISE is generated (Step 3.3.1).

B. Hierarchical Optimization

To improve the efficiency of the optimization process, we try to minimize the number of steps until an acceptable solution is found. Therefore, we applied the *hierarchical optimization approach* shown in Fig. 3. First the instructions are classified into different (sub)groups based on their characteristics, e.g. all ALU-instructions are put into the same group (Step 1). Afterwards, at each hierarchy-level, the (sub)groups are ranked according to their aging impact (Step 2). However, as the aging rate strongly depends on the encoding, the real aging rates cannot be used for this ranking. As a matter of fact, instructions affecting the hardware-implementation of the decoder have the biggest aging impact. Therefore, the impact on hardware modifications due to a particular coding scheme is used to form the ranking. If there are several (sub)groups affecting the hardware-implementation, they are ranked depending on their occurrence frequency. The next step is to find the best encoding for each (sub)group, for all hierarchy-levels starting at the coarsest hierarchy-level (Step 3.1). Thereby, at each level, the group ranking determines the optimization order. As a result, an exhaustive technique can be applied if there are only a few (sub)groups at the same hierarchy level with considerable aging impact. Otherwise, the SA process of Section IV.A can be applied for the (sub)groups.

Using this approach, at each hierarchy-level, only the opcode bits corresponding to that level are modified. For example, if there are 16 groups at the coarsest level, only the four most significant opcode bits are modified (i.e. 16! configurations). As we used just 16 instruction groups, each of which contained at most 16 instructions the search space for the entire optimization process became much smaller than the original space ($16!^2 \approx 4 \cdot 10^{26}$ vs. 10^{297}), leading to fewer optimization steps.

We compared this approach with the optimization method described in the previous subsection. To limit the runtime for the latter, we enforced that one of the modified instructions in each iteration had to be one of the 10 most frequent instructions

1. Select a starting instruction set encoding (ISE): ISE_old
2. Get overall MTTF and worst-case delay after X years
   for decode stages (see IV.E): MTTF_old & D_old
3. **While** solution is not good enough or number of steps $<$ limit **do**
   3.0. Adjust temperature T
   3.1. Generate "neighbor" ISE of ISE_old: ISE_new
   3.2. Get overall MTTF and worst-case delay after X years
      for decode stages (see IV.E): MTTF_new & D_new
   3.3. **If** ISE_new is not acceptable according to Equation (1)
      3.3.1. **then** GoTo 3.1.
      3.3.2. **else** ISE_old = ISE_new
            ISE_best = ISE_old */* memorize best ISE */*
            GoTo 3.1.
      EndIf
   End

Fig. 2. Algorithm to generate an aging-aware ISE

1. Partition instructions into groups and subgroups
   */*Instruction groups, subgroups inside groups, */*
   */*instructions insides subgroups, etc.*/*
2. Rank each group (and subgroups subsequently)
   2.1 Based on their hardware-impact
   2.2 **If** there are groups/subgroups with same ranking
      then use occurrence frequency to rank these
3. **For** the coarsest down to the finest hierarchy-level **do**
   For the highest ranked group down to the lowest **do**
      3.1 Find the best encoding for the elements within that group
         */*Either exhaustive or with simulated annealing*/*
      3.2 Stop as soon as MTTF is satisfactory
   Endfor
Endfor

Fig. 3. Hierarchical approach to obtain an aging-aware ISE

to avoid modifying only the encoding of infrequent instructions, which have negligible effect on the overall aging rate. While this version usually finds a feasible ISE within the first 80 to 90 iterations, the hierarchical approach needs usually at most 30 iterations (i.e. at most 2 hours) to obtain an acceptable ISE, i.e 3x improvement in runtime.

Please note that this hierarchical approach can be divided into fewer or more optimization levels, depending on the size of the instruction set and maximum runtime. Fewer levels allows to investigate more ISEs and hence could find a better solution than an approach with more levels. However, more hierarchy depth results in a much shorter runtime.

Furthermore, please note that also other grouping schemes can be used for this purpose. In this work, we considered a two-level categorization and grouped the instructions based on their type, e.g. all branch, arithmetic, load, store and logic instructions had their own group, which is also the highest level of the hierarchy. The next hierarchy-level is the lowest level, i.e. all instructions inside a group are optimized independently. Other schemes, for example based on the occurrence frequency, are also possible and can affect the runtime as well as MTTF.

C. Runtime Analysis and Further Improvements

The runtime for a single simulated annealing step depends mainly on the time required for synthesizing the decoding stages, as the time for simulation and aging-estimation can be reduced to a negligible fraction (see Sec. IV.E). Doing so, a single step takes in our case less than 4 minutes, which means that 100 steps can be performed within 6 hours.

In addition, we avoid re-evaluating the same ISE in multiple simulated annealing steps, to reduce the runtime further. Also, the best solution found during simulated annealing steps is memorized, such that we can always apply the best ISE in terms of MTTF that was found and not just the last accepted one, which may not necessarily be the best one.

D. Applying Modified Instruction Encoding

An ISE modification affects the usability of existing software binaries since software compiled for the original ISE can no longer be executed. To address this issue a software- or a hardware-based technique can be used.

The software-based solution re-compiles applications using a compiler based on the modified ISE either on-the-fly (runtime compilation) or when the application is started for the first time. In case of application-specific processors re-compilation is not necessary, since hardware and compiler are typically designed in close interaction and backwards compatibility is seldom required. In general, the advantage of this approach is that it infers no additional hardware costs and is easy to implement.

The hardware-based approach is intended for processors that have to be backward compatible to old software, or when re-compilation is not a feasible solution. Therefore, a mapper is used, which translates the standard ISE into the aging-aware ISE during runtime by using a lookup-table or logic-statements (if-else). This mapping can be done while the instructions are written to or read from the instruction cache. Our analysis shows that, the overhead of such a solution is negligible (less than 0.2 % area overhead for the FabScalar microprocessor). Besides area overhead this mapper can potentially impact performance, since it could increase the critical path length. However, in case of FabScalar using a mapper did not negatively affect the critical path, i.e there was no performance penalty.

E. Aging-Estimation Flow

The flow to calculate the aging rate (MTTF) of a pipeline stage is based on the one presented in [16]. In this flow, to accurately estimate the aging rate (MTTF) of a pipeline stage, the properties (duty cycle, switching activity) for all internal signals of this stage, its gate-level implementation and temperature behavior are analyzed. Hence, the first step is to generate a gate-level netlist of this stage. Afterwards gate-level simulations using real-world applications are performed to obtain the properties of all internal signals for the evaluated stage under realistic scenarios. This data is used to extract power and then temperature based on the floorplan and layout. Please note that since two neighboring stages affect the temperature behavior of each other, all stages have to be considered during the temper-

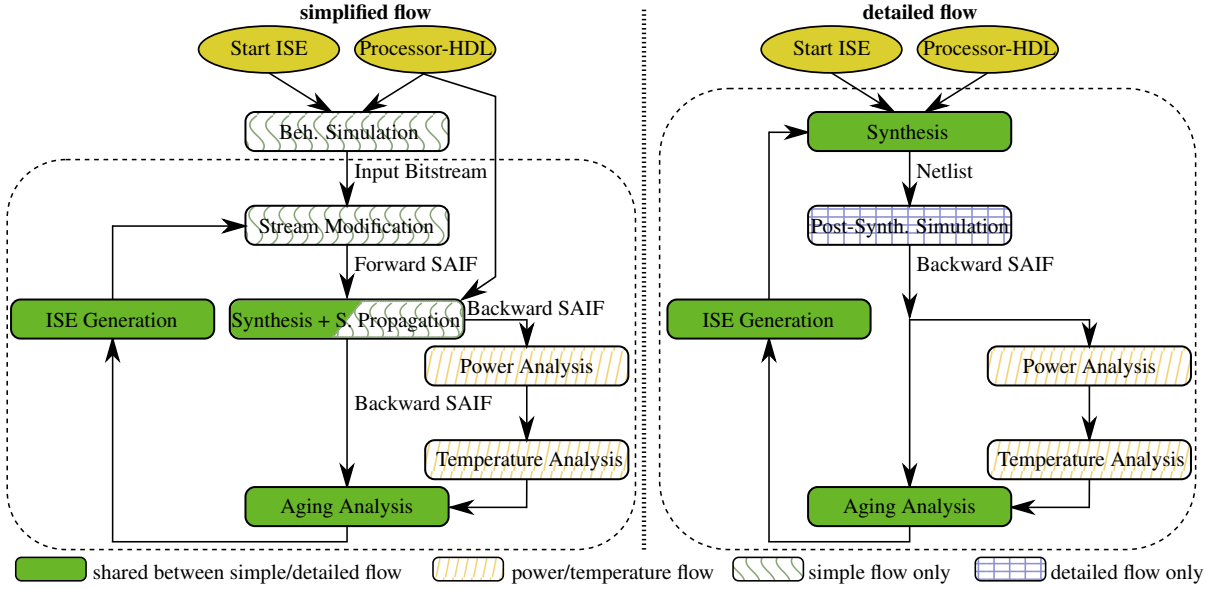

Fig. 4. ISE Optimization flows including aging analysis: Left: fast version without post-synthesis simulation, Right: slow and accurate version

Synthesis+Path Extraction	Synopsys Design Compiler D-2010.03
Simulation+VCD-Generation	Cadence NCSim 12.10 + SPEC2000
Power Extraction	Synopsys PrimeTime F-2011.06
Temperature Extraction	HotSpot 5.02 [12]
Aging Analysis	Inhouse C++-Tool

TABLE I
TOOLS USED FOR RESULT EXTRACTION

ature analysis. Then, the signal properties, temperature information and the netlist are used to estimate the V_{th}-shift of each transistor using accurate aging models based on [22] for BTI and [19] for HCI. Afterwards, the delay degradation of each gate is estimated based on an alpha-power delay model [4]. Finally, this information is given to the synthesis tool (in form of a Standard Delay Format, SDF, file) together with the netlist. The synthesis tool annotates the circuit with the degraded gate delays and afterwards the aged stage delay is extracted. As the synthesis tool considers all possible paths during the critical path extraction process, it is guaranteed that no path is excluded. Having the delay knowledge the stage's MTTF can be calculated. The tools used for these steps are given in Table I.

While the runtime of the aging estimation itself is negligible, the runtime of the post-synthesis simulations significantly affects the runtime of the optimization process. For this reason we propose to replace the post-synthesis simulations during the optimization phase as shown in Fig. 4. Instead prior to the optimization process a behavioral simulation is performed and the input-stream for the decoding stages is stored in a file. Then, during the optimization process, this input-stream is modified according to the ISE changes, i.e. old opcodes are replaced with modified ones. The resulting input signal properties are then given to the synthesis tool (in form of a Switching Activity Interchange Format, SAIF, file). The synthesis tool propagates these properties through the entire design and calculates the signal properties for all (internal) signals. By this means extracting the internal signal properties takes a negligible fraction of time compared to post-synthesis simulation (a few seconds vs. $\approx$ 30 minutes for 10^6 clock cycles). However, the aging estimation accuracy will be impacted, as signal correlations are not taken into account. Nevertheless, we observed that the inaccuracy in terms of delta delay is less than 0.5 % and hence accurate enough to be used in the optimization process.

Please note that for the final results again gate-level simulations are used to extract the signal properties. This way it is ensured that the presented results are accurate and that the optimization process was successful.

V. EXPERIMENTAL RESULTS

In this section the impact of an aging-aware ISE on the FabScalar microprocessor [7] is presented (Section V.A). The aging-aware ISE was generated using the hierarchical flow presented in Section IV. For the evaluation we used six SPEC2000 benchmarks (bzip, gap, gzip, mcf, parser, vortex) and simulated 10^6 cycles after a warmup. Furthermore, in Section V.B we compare our technique to the inversion method from [11].

A. Evaluation of Aging-aware Instruction Set Encoding

In case of FabScalar, the Predecode stage is more critical than the Decode stage, which means that first the instructions with considerable aging impact on the Predecode stage have to be optimized to extend the overall MTTF. As wearout of the Predecode stage is mainly affected by the branch instruction group, the best encoding for the corresponding instruction group was exhaustively determined using 16 iterations. Afterwards 100 simulated annealing iterations were performed to optimize the encoding for each branch instruction inside the group. Already after 25 iterations, i.e. after not even 100 minutes, the best ISE was obtained for which the delay degradation within the first 3 years dropped from 9.5 % to 8.1 % for the Predecode stage and from 6.4 % to 6.1 % for the Decode stage. Hence, the overall MTTF of the decoding stages can be improved by 1.93x from 3 years to 5.8 years as shown in Table II. This considerable improvement comes from the fact that the relation between runtime and delay degradation follows a root-like function. For example, in case of BTI the following relation can be used to estimate the delay degradation [2]:

$$\Delta d(t) \sim \delta^n \cdot t^n, \tag{2}$$

where, d is the delay, δ the transistor's duty cycle, t the runtime and n is a technology constant equal to 0.25. Hence, a delta delay reduction from 9.5 %, to 8.1 % corresponds to a duty cycle, which is roughly 1.9x smaller than the original one. As a result, the lifetime improves by 1.9x. Since the behavior for HCI is very similar, however with $n = 0.5$, the real results shown in Table II slightly differ from this estimation.

Stage	Standard Encoding			Best Encoding		
	Delay [ns] (0y)	Delay [ns] (3y)	MTTF [years]	Delay [ns] (0y)	Delay [ns] (3y)	MTTF [years]
Predecode	1.35	1.48	3.0	1.35	1.46	5.8
Decode	1.34	1.43	15.9	1.34	1.42	19.1
Overall	1.35	1.48	3.0	1.35	1.46	5.8 **+1.93x**

TABLE II
IMPROVEMENTS OF THE BEST ISE IN TERMS OF MTTF AND DELAY (BOTH WORST-CASE OVER THE USED SPEC2000 BENCHMARKS) FOR THE EVALUATED PREDECODE AND DECODE STAGE

Please note that the best ISE also improves power and area of the decoding stages. However, these are just positive side-effects and are negligible considering the entire processor. Furthermore, the switching activity at the inputs of the instruction buffer, that stores decoded instructions, increased in average by 14 %. Hence, depending on the memory technology also the power consumption of the memory elements can increase.

Stage	Standard Encoding			Best Encoding			Changes
	Delay [ns]	Power [mW]	Area [μm^2]	Delay [ns]	Power [mW]	Area [μm^2]	
Fetch	1.35	8.18	23262	1.35	8.18	23262	no
Predecode	1.35	11.2	35030	1.35	10.8	33459	yes
Decode	1.34	4.15	23616	1.34	3.60	23625	yes
Rename	1.33	0.91	4050	1.33	0.91	4050	no
Dispatch	1.33	0.12	1867	1.33	0.12	1867	no
Issue	1.35	9.59	30719	1.35	9.59	30719	no
RegRead	1.33	1.44	12061	1.33	1.44	12061	no
Execute	1.35	4.28	27529	1.35	4.06	27341	yes
LSU	1.35	34.1	107664	1.35	34.1	107664	yes
WriteBack	1.34	2.69	3183	1.34	2.69	3183	no
Retire	1.35	1.11	3201	1.35	1.11	3201	no
Overall	1.35	77.71	273133	1.35	76.54 -1.5 %	271383 -0.6 %	

TABLE III
COMPARISON OF THE FABSCALAR'S STANDARD ISE AND THE BEST OBTAINED ONE IN TERMS OF DESIGN-TIME DELAY, AVG. POWER (W/O MEMORY) AND AREA (W/O MEMORY) CONSIDERING ALL BENCHMARKS

	Our Technique	Periodical Inversion [11]		
		never	always	every 10^3 cyc
Δ-Delay @ 3y	8.1 %	9.1 %	9.0 %	9.1 %
MTTF	5.8 years	4.0 years	4.1 years	4 years

TABLE IV
COMPARISON BETWEEN OUR PROPOSED TECHNIQUE AND PERIODICAL ISE INVERSION [11] IN TERMS OF DELAY DEGRADATION AND MTTF (NEVER = ISE IS NEVER INVERTED, ALWAYS = ISE IS ALWAYS INVERTED, EVERY 10^3 CYC = INVERSION PERIOD =10^3 CYCLES)

Therefore, if power is another optimization objective our technique has to be combined with power reduction techniques.

In Table III the effect of ISE modification on the entire microprocessor is shown. As it can be seen, ISE changes also result in modifications of the Execution stage as well as Load-Store-Unit (LSU). However, in terms of delay, wearout or MTTF these changes are negligible. This is because the instruction opcode forms only a small part of the inputs.

B. Aging-aware ISE vs. Periodical Inversion

As mentioned in Section II.B.1, ISE can be periodically inverted to reduce BTI-induced wearout [11]. To compare this approach with our aging-aware ISE, we implemented that technique in the Predecode stage of FabScalar.

However, the improvements in terms of MTTF are much smaller than that of our proposed technique as summarized in Table IV. ISE inversion every 1000 cycles increases MTTF by just 1 year. Moreover, if ISE is inverted permanently, the improvements become even slightly better. Hence, our proposed technique significantly outperforms the periodic inversion technique of [11]. This is due to two reasons. First, HCI-induced wearout is not considered in [11]. Second, the periodical inversion is intended to balance wearout (duty cycles ≈ 0.5). However, this does not necessarily yield the best MTTF. The reason is that it is often better to reduce wearout of the most critical paths as much as possible (duty cycles $<< 0.5$) at the expense of faster wearout of non-critical paths (duty cycles $>> 0.5$). Overall this way a much better MTTF can be achieved.

Please note that the delay degradation with periodical inversion is not directly comparable with the degradation of the standard design, as some additional circuitry is necessary to re-invert the opcode everywhere it is used. Hence, also the overall circuit wearout is slightly different.

VI. CONCLUSION

Nanoscale microprocessors are exposed to accelerated transistor aging due to Bias Temperature Instability and Hot Carrier Injection. While various techniques exist to mitigate transistor aging in the execution units, only very few approaches target the frontend of the instruction pipeline. However, as we have shown, the decoding stages can become aging-critical and limit the microprocessor lifetime. To alleviate this problem we proposed a novel aging-aware instruction set encoding methodology (ArISE), which increases MTTF of the decoding stages. The ArISE technique exploits the fact that the instruction set encoding significantly affects the gate-level implementation as well as the signal behavior (switching activity, duty cycle) which all affect aging. To find an aging-aware instruction set encoding we presented a hierarchical heuristic algorithm that improves the binary instruction opcode with respect to MTTF. Our experimental results show that ArISE can improve MTTF of the decoding stages of the FabScalar microprocessor by 1.93x with negligible implementation costs.

VII. ACKNOWLEDGEMENT

This work was partly supported by the German Research Foundation (DFG) as part of the national focal program "Dependable Embedded Systems" (SPP-1500, `http://spp1500.ira.uka.de`).

REFERENCES

[1] L. Benini *et al.*, "Reducing Power Consumption of Dedicated Processors Through Instruction Set Encoding," in *GLSVLSI*, Feb. 1998, pp. 8–12.
[2] S. Bhardwaj *et al.*, "Predictive modeling of the nbti effect for reliable design," in *CICC*, Sep. 2006, pp. 189–192.
[3] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, pp. 10–16, Nov. 2005.
[4] K. A. Bowman *et al.*, "A physical alpha-power law MOSFET model," in *ISLPED*, 1999, pp. 218–222.
[5] A. Bravaix *et al.*, "Hot-Carrier Acceleration Factors for Low Power Management in DC-AC stressed 40nm NMOS node at High Temperature," in *IRPS*, Apr. 2009, pp. 531–548.
[6] A. Chattopadhyay *et al.*, "Power-efficient Instruction Encoding Optimization for Embedded Processors," in *VLSID*, Jan. 2007, pp. 595–600.
[7] N. Choudhary *et al.*, "FabScalar: Automating Superscalar Core Design," *IEEE Micro*, pp. 48–59, May 2012.
[8] M. DeBole *et al.*, "New-Age: A Negative Bias Temperature Instability-Estimation Framework for Microarchitectural Components," *Int'l J. of Parallel Prog.*, pp. 417–431, Aug. 2009.
[9] M. Ebrahimi *et al.*, "Aging-aware Logic Synthesis," in *Proc. of the Int'l Conf. on Computer-Aided Design*, Nov. 2013.
[10] F. Firouzi *et al.*, "NBTI Mitigation by NOP Assignment and Insertion," in *DATE*, Mar. 2012, pp. 218–223.
[11] E. Gunadi *et al.*, "Combating Aging with the Colt Duty Cycle Equalizer," in *MICRO*, Dec. 2010, pp. 103–114.
[12] W. Huang *et al.*, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design," *IEEE Trans. on VLSI Systems*, pp. 501–513, May 2006.
[13] K. Kang *et al.*, "Estimation of Statistical Variation in Temporal NBTI Degradation and its Impact on Lifetime Circuit Performance," in *ICCAD*, Nov. 2007, pp. 730–734.
[14] Y. Kunitake *et al.*, "Signal Probability Control for Relieving NBTI in SRAM Cells," in *ISQED*, Mar. 2010, pp. 660–666.
[15] F. Oboril *et al.*, "Reducing NBTI-induced Processor Wearout by Exploiting the Timing Slack of Instructions," in *CODES+ISSS*, Oct. 2012, pp. 443–452.
[16] F. Oboril and M. B. Tahoori, "MTTF-Balanced Pipeline Design," in *DATE*, Mar. 2013, pp. 270–275.
[17] S. Pae *et al.*, "BTI Reliability of 45 nm High-K + Metal-Gate Process Technology," in *IRPS*, May 2008, pp. 352–357.
[18] T. Siddiqua and S. Gurumurthi, "A Multi-Level Approach to Reduce the Impact of NBTI on Processor Functional Units," in *GLSVLSI*, May 2010, pp. 67–72.
[19] E. Takeda *et al.*, "New hot-carrier injection and device degradation in submicron MOSFETs," *IEEE Proc. I, Solid-State and Electron Devices*, pp. 144–150, Jun. 1983.
[20] R. Vattikonda *et al.*, "Modeling and Minimization of PMOS NBTI effect for Robust Nanometer Design," in *DAC*, Jun. 2006, pp. 1047–1052.
[21] S. Wang *et al.*, "Low Power Aging-Aware Register File Design by Duty Cycle Balancing," in *DATE*, Mar. 2012, pp. 546–549.
[22] W. Wang *et al.*, "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology," *IEEE Trans. on Device and Materials Reliability*, pp. 509–517, Dec. 2007.
[23] Y. Wang *et al.*, "On the efficiancy of Input Vector Control to mitigate NBTI effects and leakage power," in *ISQED*, Mar. 2009, pp. 19–26.
[24] X. Yang and K. Saluja, "Combating NBTI Degradation via Gate Sizing," in *ISQED*, 2007, pp. 47–52.