MTTF-Balanced Pipeline Design

Fabian Oboril and Mehdi B. Tahoori
Chair of Dependable Nano Computing (CDNC), Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Email: {fabian.oboril, mehdi.tahoori}@kit.edu

***Abstract*—As CMOS technologies enter nanometer scales, microprocessors become more vulnerable to transistor aging mainly due to Bias Temperature Instability and Hot Carrier Injection. These phenomena lead to increasing device delays during the operational lifetime, which results in increasing pipeline stage delays. However, the aging rates of different stages are different. Hence, a previously delay-balanced pipeline becomes increasingly imbalanced resulting in a non-optimized design in terms of Mean Time to Failure (MTTF), frequency, area and power consumption. In this paper, we propose an MTTF-balanced pipeline design, in which the pipeline stage delays are balanced after the desired lifetime rather than at design time. This can lead to significant MTTF (lifetime) improvements as well as additional performance, area, and power benefits. Our experimental results show that MTTF of the FabScalar microprocessor can be improved by 2x (or frequency by 3 %) while achieving an additional 4 % power, and 1 % area optimization.**

I. Introduction

Nowadays almost all microprocessors ranging from low-power embedded parts to high-performance processors use a pipelined architecture to increase the instruction throughput and by that means the performance. To maximize the performance, designers follow the same paradigm since the dawn of the first pipelined microprocessors: They try to balance all pipeline stage delays at design-time (called: *delay-balanced pipeline*). The advantage of this approach was the combination of high throughput together with an efficient energy and area usage. This was due to the fact, that as long as a pipeline stage is faster than the slowest one (which determines the frequency), it can be often made slower using gate sizing or higher threshold voltage to save energy and die area [8], [11].

However, with the ongoing aggressive transistor scaling of CMOS technology, reliability expressed in *Mean Time to Failure (MTTF)* is becoming an important design constraint, together with performance, power and area [2], [3], [13]. Transistor aging due to *Bias Temperature Instability* (BTI) [15], [20] and *Hot Carrier Injection* (HCI) [17] leads to increasing path delays and so increases pipeline stage delays during runtime. Hence, nowadays the clock frequency of the shipped parts can no longer be set according to the worst-case delay at design time (t_{design}). Instead, manufacturers have to add safety margins to their delay-balanced designs, to ensure that the chips will be functional for a certain lifetime (t_{target}).

As we will show in this paper, the wearout rates (i.e. delay increase due to BTI and HCI) vary widely among pipeline stages, due to different temperature and usage rates. For example, our experimental results show that the execution stage of the FabScalar microprocessor [7] has a 3x higher delay increase than the retire stage[1] within the first 3 years. Hence, although the original pipeline was delay-balanced, after some operational runtime the stage delays become highly imbalanced. This also affects MTTF[2] of different pipeline stages, which varies tremendously (more than 20x). Thus, one stage can fail due to timing violation while others are still executing correctly. Obviously such a design can be further improved. Slow-aging stages should have less slack to save area and energy, while fast-aging stages should have more slack, to improve the overall MTTF.

In this paper we proposed a radically new MTTF-balanced pipeline design scheme to replace the traditional delay-balanced paradigm. Using this paradigm the stage delays are balanced at t_{target} and not at t_{design}. Hence, also all stage MTTFs are equal to t_{target}. By that means the full optimization potential for MTTF, area, power and performance can be exploited.

We demonstrate these benefits using the FabScalar microprocessor. While the delay-balanced design results in 20x variation in MTTF of different stages, our MTTF-balanced approach resolves this problem and yields a more than 2x longer MTTF for the entire microprocessor, while achieving the same performance (i.e. frequency) as the delay-balanced design. In addition, the power consumption can be reduced by 4 % and the area by 1 %. If an improved MTTF is of secondary interest, the gained headroom can be used to increase the clock frequency by 3 % for a target lifetime of 3 years.

In summary, the key contributions of this work are:

- We present a generic flow based on standard commercial tools to estimate the aging rate of a pipeline stage at transistor-level as well as its MTTF considering detailed temperature and signal information (delay, switching activity, signal probabilities) for real-world applications.
- To avoid imbalanced pipeline stage MTTFs and improve the microprocessor design in four dimensions[3], including reliability, we propose a novel, generic pipeline design paradigm applicable to any in-order and out-of-order processor: MTTF-balanced pipeline design. Furthermore, we provide a design-flow that describes the design process for such a pipeline.

[1] In out-of-order processors the retire stage restores the original instruction order after the out-of-order execution

[2] In this work MTTF is equal to the time until first timing violation due to aging occurs

[3] performance, area, power and MTTF

978-3-9815370-0-0/DATE13/©2013 EDAA

The rest of this paper is organized as follows. In Section II the BTI and HCI phenomena are introduced. The new design paradigm is motivated in Section III, followed by the presentation of the proposed MTTF-balanced design paradigm itself in Section IV. In Section V, the flow to extract MTTF for each stage is explained. Afterwards we present in Section VI our experimental results followed by a discussion of related work in Section VII. Finally, Section VIII concludes the paper.

II. Transistor Aging

Transistors degrade mainly due to Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) [2]. Both effects lead to a threshold voltage shift of the impaired transistors, which manifests in increasing gate and path delays. Hence, these effects increase the pipeline stage delay during runtime. In this section the impact on threshold voltage is explained.

A. BTI

BTI appears in two difference types: Negative BTI (NBTI) and Positive BTI (PBTI). While NBTI is affecting PMOS transistors, PBTI degrades NMOS transistors and emerged as a reliability issue with the introduction of high-k gate oxides [15]. In both variants, BTI consists of two different phases. When a logic '0' (logic '1') is applied at the gate of a PMOS (NMOS) transistor, this transistor is under (NBTI/PBTI)-*stress*. During that phase, traps are generated in the interface between gate oxide and channel, which increases $|V_{th}|$. In contrast, when a logic '1' (logic '0') is applied at the gate of the same transistor, some traps are filled, which leads to a decreasing $|V_{th}|$ (*recovery* phase). However, the initial shift cannot be entirely compensated leading to an overall V_{th} drift over time. Thereby, the shift depends on several different aspects, e.g. temperature and the ratio between the time a transistor is under stress and total time (duty cycle). For estimating the V_{th} shift the model presented in [20] is used.

B. HCI

HCI is mainly affecting NMOS transistors, where accelerated electrons inside the channel collide with the gate oxide interface and thereby create electron-hole pairs. Thus, free electrons get trapped in the gate oxide layer, which leads to an increasing V_{th}. In contrast to BTI, the V_{th} shift due to HCI is irreversible [20]. The V_{th} shift has an exponential relation with temperature [4] and since "hot" energetic electrons are generated when the NMOS transistor is making a transition, the V_{th} shift is also very sensitive to the number of transitions [17], i.e. clock frequency, runtime and switching activity. Putting all this together leads to the model detailed in [14], which we use in this work for estimating the V_{th} shift.

III. Motivation and Main Idea

As mentioned in Section II, BTI and HCI can significantly increase pipeline stage delays during runtime. To illustrate this circumstance and motivate our work, we use as an example FabScalar, an out-of-order, 11-stage, superscalar processor [7], which was synthesized with Synopsys Design Compiler and the TSCM 65 nm library. However, our idea is not limited to this specific processor or library and can be applied to every in-order or out-of-order pipelined microprocessor.

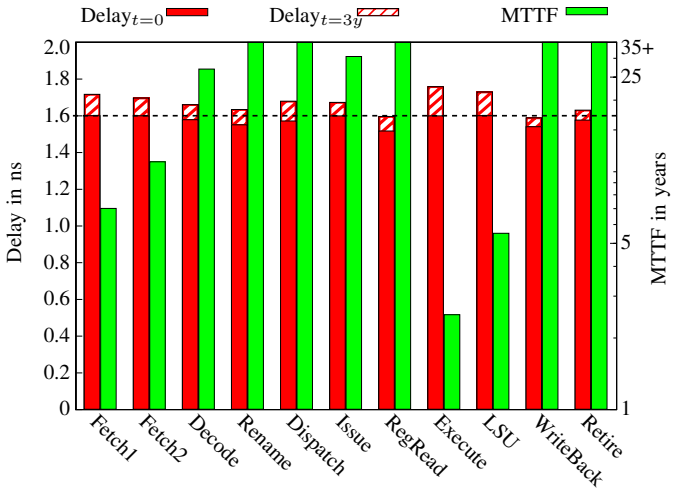

Fig. 1. Delay after 0 and 3 years and MTTF for different pipeline stages of the FabScalar microprocessor

To investigate the aging rates of different pipeline stages, we extracted the delay at design time and after 3 years ($= t_{\text{target}}$) for each stage using the flow described later in Section V. The results of this analysis, illustrated in Figure 1, clearly show that different pipeline stages have different wearout rates. While the delay of the execution stage increases by 9 % within 3 years, the delay of the issue stage increases just by 4 %, although their delays at design time were similar ($\approx$ 1.60 ns). These differences are due to the fact that the parameters influencing aging, i.e. temperature and usage (duty cycle, switching activity) are different for different pipeline stages. Also the imbalance in terms of MTTF (given a timing slack of 10 %) can be huge. Between the execution stage, which starts to fail first, and the issue stage there is a factor of more than 10x difference. In the worst case (RegRead) the imbalance is even more than 20x. This means that one pipeline stage already produces timing failures, while other stages are still running correctly. Hence, the latter are overdesigned. Furthermore, we observed that the timing critical stage changes over runtime. At the beginning it is Fetch1, which changes to Execute after less than 2 years.

Please note that for other microprocessor designs or other technology libraries Figure 1 might look different, i.e. other stages age faster, have different MTTF values, etc. However, the overall observation of delay and MTTF imbalance after a certain runtime remains valid (e.g. in [9] similar results are reported for the IVM microprocessor).

Since MTTF of the microprocessor is determined by the smallest MTTF of all pipeline stages, and the clock frequency is mandated by the slowest stage after the target lifetime t_{target}, it is obvious that the described imbalance leads to a sub-optimal design. Looking at the criticality of different pipeline stages at t_{target}, there are two possible optimization strategies:

- First, stages that are faster (i.e. have more slack) than the critical stage after t_{target} can be designed slower

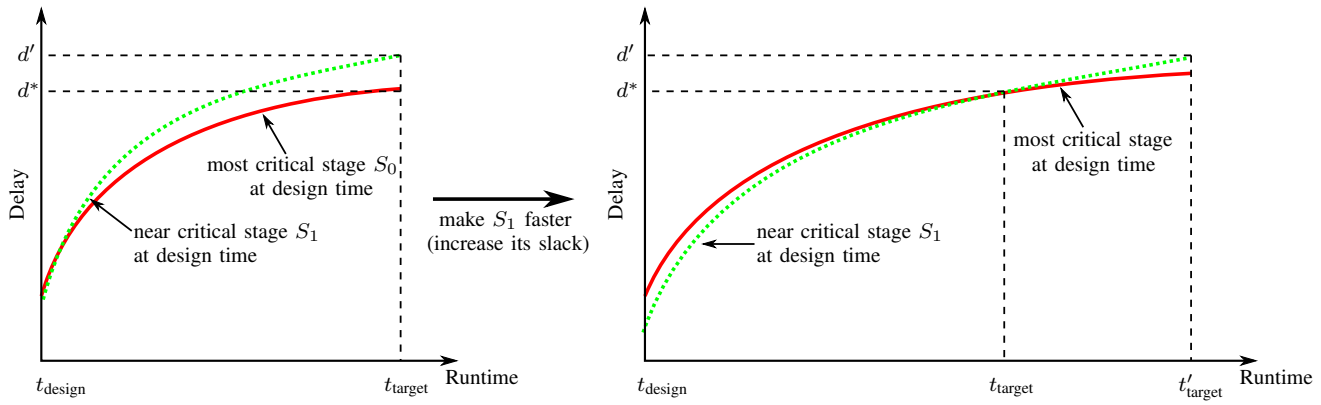

Fig. 2. Abstracted graphic to illustrate the effect of delay reduction on MTTF and performance
Left (Delay-Balanced): At t_{design} stage S_1 has less delay than stage S_0, but ages faster $\Rightarrow$ For clock period $= d'$ the target lifetime $= t_{\text{target}}$
Right (MTTF-Balanced): S_1 is accelerated $\Rightarrow$ After t_{target} stages S_0 and S_1 have same delay (d^*) $\Rightarrow$ smaller clock period possible (d^* instead of d')
Alternatively: For same clock period (d') a longer lifetime is possible (t'_{target})

(i.e. with less slack), by applying appropriate gate-sizing techniques, using a higher threshold voltage, etc., leading to extra energy and area savings [8], [11].

- Second, if a stage S_1 has more delay after t_{target} than the design-time-critical stage S_0 after t_{target}, the first one can be designed faster (i.e. with more slack). This can be used in two different ways, both shown in Figure 2. First, the clock frequency (i.e. clock period) can be kept constant, so that a higher MTTF can be achieved (that means that t_{target} can be increased). In Figure 2 that means that the clock period remains at d' and that the target lifetime increases to t'_{target}. In the second case, MTTF is kept constant (i.e. equal to t_{target}), so that the clock period can be reduced (i.e. higher frequency, meaning higher performance). Using the annotations from Figure 2 that means that the new clock period is $d^* < d'$.

Hence, in summary, slow-aging stages should be designed with less slack (i.e. slower) to save area and energy, while the timing slack for fast-aging stages should be increased (i.e. speed-up), to improve their MTTF and in turn the MTTF of the entire microprocessor or the overall performance (i.e. clock frequency). Thereby. the key design aspect is that the pipeline stage delays should be balanced after the target lifetime and not at design time. Since this is not achievable using the traditional delay-balanced design approach, we propose a new MTTF-balanced pipeline design. We will explain this paradigm in detail in the following section.

IV. MTTF-balanced Pipeline Design

The key idea of the MTTF-balanced pipeline design is that the pipeline stage delays are balanced after the desired lifetime t_{target} rather than at design time t_{design}. Hence, also MTTFs of all stages are equal to t_{target}. In the following we will explain the flow to generate an MTTF-balanced pipeline. This flow, detailed in Figure 3 leads to a design that is as fast as possible for a given target lifetime t_{target}.

The starting point of the transformation process is a delay-balanced design, as it is used nowadays (Step 1). Next, the delay, d^*, after the given target lifetime t_{target} of the critical stage at design time is extracted using the flow presented later in Section V (Step 2). Since this stage cannot be designed any faster (otherwise it would not be critical at design time), the clock period of the final MTTF-balanced pipeline cannot be smaller than this delay. Since the final design should be as fast as possible, d^* will work as a reference for the clock period.

The next step (Step 3) is to extract the delay d_i of each pipeline stage after t_{target} and to compare it with d^*. If

1. Generate a delay-balanced design
2. d^* = delay at t_{target} of stage, that is critical at t_{design}
   d' = delay at t_{target} of stage, that is critical at t_{target}
/* $\Rightarrow$ *clock period of delay-balanced design* $= d' > d^*$ */
3. **Forall** stages $i = 0, \ldots, n$ **do**
   Extract d_i = delay at t_{target} of stage i
   stage$^i_{old}$ = current version of stage i
   /* *if stage is faster than necessary* */
   If ($d_i < d^*$) **then**
   stage$^i_{new}$ = new version of stage$^i_{old}$ with more delay at t_{design}
   /* *if stage is slower than necessary* */
   Elseif ($d_i > d^*$) **then**
   stage$^i_{new}$ = new version of stage$^i_{old}$ with less delay at t_{design}
   /* *if no speedup is possible adjust* d^* */
   If (stage$^i_{new}$ == stage$^i_{old}$) **then**
   $d^* = d_i$
   Goto 3. /**restart with new* d^* */
   Endif
   Else
   stage$^i_{new}$ = stage$^i_{old}$
   Endif
   End
/* *Extract new delay information at* t_{target} */
4. **Forall** stages $i = 0, \ldots, n$ **do**
   Extract $d_{i,new}$ = delay at t_{target} of stage$^i_{new}$
   /* *if old version was faster and new version is slower take old one* */
   If ($d_{i,new} > d^*$) and ($d_i < d^*$) **then**
   stage$^i_{new}$ = stage$^i_{old}$
   Endif
   End
/* *If no more modifications are possible, transformation is done* */
5. **Forall** stages $i = 0, \ldots, n$ **do**
   If (stage$^i_{old} \neq$ stage$^i_{new}$) **then**
   Goto 3. /* *restart* */
   Endif
   End
6. **Done.** /* *generated MTTF-balanced design* */

Fig. 3. Algorithm to transform a delay-balanced design into an MTTF-balanced design

the delay is smaller than d^* (i.e. the stage is faster than necessary), a new, slower version of this pipeline stage will be generated using gate sizing, higher threshold voltage, and so on [8], [11]. Therefore, we adjust the timing constraints for this pipeline stage and re-synthesize it. This means that also paths are reorganized and optimized for the new constraints, which further enhances energy and area efficiency. In case re-synthesis is not feasible, it is also possible to modify only small sub-circuits or gates [6].

If the delay d_i is greater than d^* (i.e. stage is slower than necessary), a new, faster version (using gate sizing, etc.) will be generated. If this is not possible, the final design has to use a clock frequency of at least d_i. Hence, d^* will be increased and set to d_i. In that case Step 3 has to be restarted.

After all pipeline stages are analyzed and eventually modified, their new delay information is extracted (Step 4). Here it is extremely important to investigate all stages in one step and not only those that have been changed in Step 3. This is due to the fact that as long as one stage is modified, the power consumption and hence the temperature distribution will change, which can affect also the wearout and hence the delay of other stages. If it is detected in Step 4 that a stage, which was previously faster than necessary, is now slower than necessary, the changes leading to this situation will be reverted and the previous implementation will be used. Since these situations are undesired, the delay differences between the new and the old implementation should be very small. Therefore, we use a resolution of 0.01 ns for our experiments.

If there is at least one modified stage remaining after Step 4, again Step 3 followed by Step 4 will be executed until no pipeline stage is modified anymore, i.e. until no stage can be tuned further. When this saturation state is reached, the transformation to the MTTF-balanced design is finished.

Please note that in some application areas it might be more important to minimize the die area or power consumption, instead of performance (clock frequency). In that case, the transformation procedure is very similar to the one explained before. The only difference is, that d' is used as reference delay in place of d^*. Hence, no stage will be accelerated. Instead all stages, beside the one that is critical at t_{target}, will be designed slower, hence with less power and area consumption.

The runtime for the transformation process depends mainly on the number of iterations, i.e. the number of synthesis steps and delay/MTTF estimation steps. In our case the latter is dominant (hours vs. minutes). Hence, the transformation time is proportional to the number of pipeline stages multiplied with the average time needed for the MTTF estimation.

V. MTTF-Estimation Flow

To accurately evaluate the aging rates (delay changes) and MTTF values for each pipeline stage, a suitable analysis flow is necessary. In this section a generic flow is described, which is based on standard industrial design tools (see Table I).

As shown in the Section II, the aging rate of a transistor strongly depends on its duty cycle and its switching activity. Moreover, these values influence the power consumption of the microprocessor and hence the temperature distribution, which in turn affects the aging rate of the transistors. Hence, it is very important to accurately calculate these values. Therefore, a gate-level description of the pipeline stage under investigation is necessary, to monitor all signals of this stage, while the processor executes representative workloads (applications).

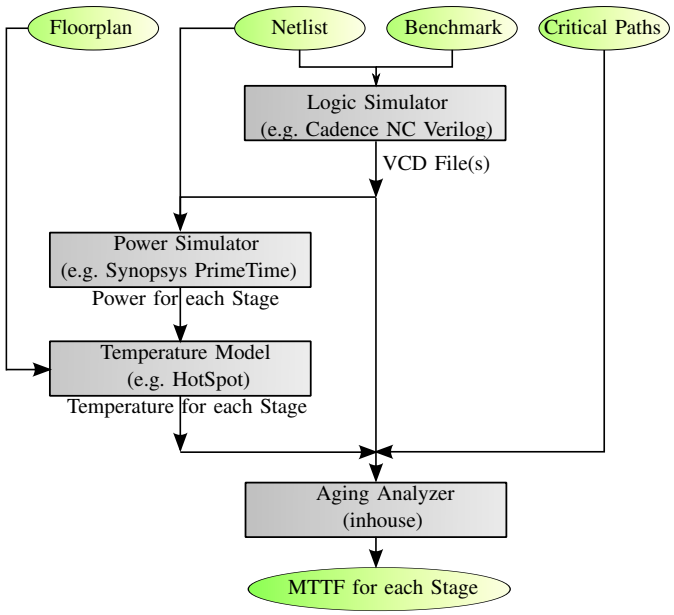

Fig. 4. Flow for extracting MTTF for each pipeline stage

Hence, the first step of the estimation flow, depicted in Figure 4, is to generate a gate-level description (netlist) of the pipeline stage under investigation. Afterwards this gate-level description is used to simulate the pipeline stage behavior, when an application (here: SPEC2000 benchmarks) is executed by the processor. During the simulation, the behavior of all signals is stored in a Value Change Dump (VCD) file. This VCD file is then used by a power simulator to extract the power consumption for each (sub)-block of the investigated pipeline stage. The results are given to a temperature model (e.g. HotSpot [10]), which calculates the temperature for each block. Since, the temperature of a block strongly depends on the temperature of the surrounding blocks, the temperature model has to consider the floorplan of the entire microprocessor as well as the power information for all other pipeline stages. The last step is to extract signal probabilities (probability that signal value is '1') and switching activity from the VCD file. This data is used together with the temperature information to calculate the threshold voltage shift for every transistor in the design under investigation using transistor-level aging models (here: model from [20] for BTI, model from [14] for HCI). By that means the delay degradation for each gate, each path and hence, for the entire pipeline

Synthesis + Path Extraction	Synopsys Design Compiler D-2010.03-SP4
Simulation + VCD-Generation	Cadence NC Sim 10.20-s073
Power Extraction	Synopsys PrimeTime F-2011.06-SP3-2
Temperature Extraction	HotSpot 5.02 [10]
Aging Analysis	Inhouse C++-Tool

TABLE I
Tools used for Result Extraction

stage can be calculated. Together with the information about the clock period the time until timing violations occur, i.e. MTTF, can be finally extracted. Please note that it is practically impossible to investigate all paths for large pipeline stages. Therefore, we only analyze the 10 % most critical paths.

The runtime of this flow is mainly determined by the runtime of the logic simulator and the power simulator, since the temperature model and the aging analyzer are heavily parallelized. For example, PrimeTime needs more than 6 hours to calculate the power consumption for 10^6 clock cycles for the Load-Store-Unit (more than 60,000 gates). A possible simplification is to use Design Compiler to extract the power consumption. However, Design Compiler does not consider the actual application behavior, but assumes switching activities and signal probabilities. Hence, the power information is not that accurate compared to PrimeTime, which considers the running application. Nevertheless this assumed data can be used, if for example the time available for MTTF estimation is very limited. Furthermore, also the assumed values for switching activities and signal probabilities can be used for the aging estimation, instead of analyzing a VCD file. Of course, accuracy will be impaired, but the time for the MTTF estimation will be reduced further. By this means it would be also possible, to extend Design Compiler in such a way, that beside timing, area and power constrained synthesis also an MTTF-constrained synthesis is available.

VI. Experimental Results

In this section a comparison using the FabScalar microprocessor [7] between the proposed MTTF-balanced design paradigm and the classical delay-balanced one is presented. The MTTF-balanced design was generated using the flow presented in Section IV for $t_{\text{target}} = 3$ years. For the evaluation we use 6 SPEC2000 benchmarks (bzip, gap, gzip, mcf, parser, vortex) and simulated the processor behavior for 10^6 cycles after a warmup. Table II summarizes the main results.

For both designs, the (worst-case) delay at design time and after 3 years was extracted (for each of the six SPEC benchmarks). The worst-case delay values over all benchmarks are reported in Table II, since these values determine MTTF as well as the clock period of the microprocessor. For this reason, the delay-balanced design has to use a clock period of at least 1.76 ns (i.e. 10 % timing margin) to achieve a lifetime (MTTF) of at least 3 years. In contrast the MTTF-balanced design needs just a clock period of 1.71 ns. Hence, the frequency can be increased by 3 %, yielding a better performance, while MTTF remains the same. This is mainly due to the fact, that the execution units and the Load-Store-Unit (LSU) can be designed faster using gate sizing, to increase their timing slack. All other stages could be designed with less slack, which made it possible for some stages to use a higher threshold voltage (20 % increase) in the (near)-critical paths of the stages. By this means the average power consumption over all benchmarks (extracted with PrimeTime) of the MTTF-balanced design is 4 % lower than the one of the traditional delay-balanced pipeline for a clock period of 1.76 ns. In addition also the area for the MTTF-balanced pipeline is slightly smaller (- 1 %).

If one does not want to increase the clock frequency (i.e. clock period remains with 1.76 ns), the MTTF-balanced design can achieve a lifetime (MTTF) of at least 7 years, which is an improvement of more than 2x. If the clock period should be equal to 1.71 ns, the MTTF of the delay-balanced design would reduce to just 1 year due to the front-loaded nature (delay change is a "sqrt-like" function of time) of the investigated transistor aging phenomena, while the MTTF-balanced approach can achieve a lifetime of 3 years. This means an improvement of more than 3x. Since, also a middle way is possible, the MTTF-balanced design can allow higher clock frequencies along with higher MTTF.

VII. Related Work

A lot of research is done at various design levels to alleviate the effects of BTI and HCI. To name just a few ones, special NBTI-resilient processors are proposed in [1] and specific input patterns at the primary inputs of a subcircuit can mitigate the NBTI effect during idle periods [21]. Furthermore, power gating [5], adaptive body biasing and enhanced instruction

Stage	Delay-Balanced						MTTF-Balanced						Changes
	delay		MTTF^1	MTTF^2	Power	Area	delay		MTTF^1	MTTF^2	Power	Area	
	$@t_{\text{design}}$ [ns]	$@t_{3\text{ years}}$ [ns]	(1.71 ns) [years]	(1.76 ns) [years]	[mW]	[μm^2]	$@t_{\text{design}}$ [ns]	$@t_{3\text{ years}}$ [ns]	(1.71 ns) [years]	(1.76 ns) [years]	[mW]	[μm^2]	
Fetch1	1.60	1.71	3.0	7.0	6.38	22822	1.60	1.71	3.0	7.0	6.28	22822	untouched
Fetch2	1.60	1.69	4.0	11.0	8.40	35549	1.62	1.71	3.0	8.0	8.38	35533	GS
Decode	1.58	1.65	12.0	27.0	1.57	22524	1.66	1.71	3.0	22.0	1.12	22988	GS, HVT
Rename	1.55	1.62	50+	50+	0.60	3992	1.65	1.70	3.0	50+	0.44	3992	GS, HVT
Dispatch	1.57	1.59	50+	50+	0.13	1880	1.65	1.71	3.0	50+	0.09	1574	GS, HVT
Issue	1.60	1.67	10.5	30.5	5.97	32259	1.64	1.71	3.0	8.5	5.96	32236	GS
RegRead	1.51	1.59	50+	50+	1.24	13820	1.64	1.68	3.0	50+	0.86	13524	GS, HVT
Execute	1.60	1.76	1.0	3.0	1.79	29103	1.55	1.69	3.0	7.5	1.81	29071	GS
LSU	1.60	1.72	1.5	5.5	27.2	117781	1.58	1.69	3.0	8.0	27.3	116508	GS
WriteBack	1.53	1.59	50+	50+	2.19	2696	1.67	1.70	3.0	50+	1.40	2667	GS, HVT
Retire	1.57	1.61	50+	50+	0.89	3133	1.67	1.71	3.0	50+	0.58	3119	GS, HVT
Entire CPU	1.60	1.76	1.0	3.0	56.4	285559	1.67 + 4 %	1.71 -3 %	3.0 +300 %	7.0 +233 %	54.2 - 4 %	284034 -1 %	

TABLE II

Comparison of a delay-balanced and MTTF-balanced design for the FabScalar microprocessor in terms of worst-case delay, worst case MTTF, avg. power (without SRAM) and area (without SRAM) considering all benchmarks (GS = gate sizing, HVT = higher threshold voltage in the critical path, MTTF^1 is MTTF for clock period of 1.71 ns, MTTF^2 is MTTF for 1.76 ns)

as well as application scheduling techniques [19] have been proposed to mitigate the effect of NBTI and HCI. All these techniques are orthogonal to our work and can be used in combination with our proposed design paradigm.

In the context of pipeline delay re-balancing, a famous technique is cycle-time stealing/borrowing. For example in [12], [18] such approaches are proposed to re-balance the pipeline delay due to process variation. Cycle time is "stolen" from fast stages and given to slow stages, so that the pipeline can operate at a clock period closer to the average stage delay. Potentially this idea can be used similarly to our MTTF-balanced design paradigm. Stages that have high aging rates take some cycle time from stages with lower aging rates, to increase their MTTF. However, using these techniques, cycle time has to be redistributed, which is a complex task and not always possible. Using our design paradigm, no redistribution is necessary, which makes our technique suitable for almost every design. In addition, our approach can improve power and area efficiency, which is not possible if cycle-time stealing is applied for a delay-balance pipeline.

Another re-balancing technique using cycle-time borrowing is presented in [16], which is intended to balanced the power consumption of different pipeline stages. By that means the problem that some pipeline stages consume much more energy than others is reduced. Potentially this can also help to avoid hotspots, which can slow down transistor aging. However, since the purpose is to minimize the overall power consumption, the timing slack for each pipeline stage is minimized after applying cycle-time borrowing to the pipeline. This slack reduction can negatively affect MTTF. In contrast, our technique tries to increase the timing slack of some stages, to improve their MTTF and so MTTF of the entire processor.

In [9] a flow to estimate the delay degradation of a pipeline stage is introduced, which is similar to the flow presented in Section V. However, this flow can just extract lower and upper bounds for aging induced delay-degradation, but not the real value, which our flow can.

VIII. Conclusion

Microprocessors at nano-scale are exposed to various reliability issues, which include a more rapid aging of all components. This leads to increasing pipeline stage delays during the operational lifetime, resulting in imbalanced designs in terms of delay and MTTF, if the delays are balanced at design time. In this paper we have shown that this imbalance hides a lot of optimization potential for higher clock frequencies, longer lifetimes (i.e. higher MTTF) as well as reduced power and area consumption.

Therefore, we proposed a radically new MTTF-balanced pipeline design scheme to replace the traditional delay-balanced paradigm. Using the new approach, the imbalance during runtime is minimized, allowing "better" designs. Our experimental results show that for the FabScalar microprocessor the MTTF-balanced design yields a more than 2x longer MTTF, while the same performance (i.e. frequency) as for the delay-balanced design can be maintained. In addition, the power consumption can be reduced by 4 % and the area by 1 %. If an improved MTTF is of secondary interest, the gained headroom can be used to increase the clock frequency by 3 % for a target lifetime of 3 years.

References

[1] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-Aware Processor," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec. 2007, pp. 85–96.

[2] K. Bernstein *et al.*, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM Journal of Research and Development - Advanced silicon technology*, pp. 433–449, Jul. 2006.

[3] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, pp. 10–16, Nov. 2005.

[4] A. Bravaix *et al.*, "Hot-Carrier Acceleration Factors for Low Power Management in DC-AC stressed 40nm NMOS node at High Temperature," in *IEEE International Reliability Physics Symposium*, Apr. 2009, pp. 531–548.

[5] A. Calimera, E. Macii, and M. Poncino, "NBTI-Aware Power Gating for Concurrent Leakage and Aging Optimization," in *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design*, Aug. 2009, pp. 127–132.

[6] J. Chen, S. Wang, and M. Tehranipoor, "Efficient Selection and Analysis of Critical-Reliability Paths and Gates," in *Proceedings of the Great Lakes Symposium on VLSI*, May 2012, pp. 45–50.

[7] N. Choudhary *et al.*, "FabScalar: Automating Superscalar Core Design," *IEEE Micro*, pp. 48–59, May 2012.

[8] O. Coudert, "Gate Sizing for Constrained Delay/Power/Area Optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 465–472, Dec. 1997.

[9] M. DeBole *et al.*, "New-Age: A Negative Bias Temperature Instability-Estimation Framework for Microarchitectural Components," *International Journal of Parallel Programming*, pp. 417–431, Aug. 2009.

[10] W. Huang *et al.*, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 501–513, May 2006.

[11] T. Karnik *et al.*, "Total Power Optimization by Simultaneous Dual-Vt Allocation and Device Sizing in High Performance Microprocessors," in *Proceedings of the 39th Annual Design Automation Conference*, Jun. 2002, pp. 486–491.

[12] X. Liang, G. Wei, and D. Brooks, "ReVIVal: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency," *IEEE Micro*, pp. 127–138, Jan. 2009.

[13] V. Narayanan and Y. Xie, "Reliability Concerns in Embedded System Designs," *Computer*, pp. 118–120, Jan. 2006.

[14] F. Oboril and M. B. Tahoori, "ExtraTime: Modeling and Analysis of Wearout due to Transistor Aging at Microarchitecture-Level," in *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Jun. 2012.

[15] S. Pae *et al.*, "BTI Reliability of 45 nm High-K + Metal-Gate Process Technology," in *IEEE International Reliability Physics Symposium*, May 2008, pp. 352–357.

[16] J. Sartori, B. Ahrens, and R. Kumar, "Power Balanced Pipelines," in *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture*, Feb. 2012, pp. 1–12.

[17] E. Takeda *et al.*, "New hot-carrier injection and device degradation in submicron MOSFETs," *IEEE Proceedings I, Solid-State and Electron Devices*, pp. 144–150, Jun. 1983.

[18] A. Tiwari, S. R. Sarangi, and J. Torrellas, "ReCycle: Pipeline Adaptation to Tolerate Process Variation," *SIGARCH Computer Architecture News*, pp. 323–334, Jun. 2007.

[19] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores," in *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture*, Nov. 2008, pp. 129–140.

[20] W. Wang *et al.*, "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology," *IEEE Transactions on Device and Materials Reliability*, pp. 509–517, Dec. 2007.

[21] Y. Wang *et al.*, "On the efficiancy of Input Vector Control to mitigate NBTI effects and leakage power," in *Proceedings of the International Symposium on Quality of Electronic Design*, Mar. 2009, pp. 19–26.