Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Design of NBTI-resilient extensible processors

Mehdi Kamal [a], Ali Afzali-Kusha [a,b,*], Saeed Safari [a], Massoud Pedram [c]

[a] School of Electrical and Computer Engineering, University of Tehran, Iran
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Iran
[c] Department of Electrical Engineering-Systems, University of Southern California, USA

ARTICLE INFO

Article history:
Received 8 July 2014
Received in revised form
29 November 2014
Accepted 1 December 2014
Available online 10 December 2014

Keywords:
Extensible processors
Custom instruction
NBTI
Speedup
Lifetime

ABSTRACT

In this paper, we present techniques for mitigating the Negative Bias Temperature Instability (NBTI) effect on extensible processors. Firstly, the effect of NBTI on the extended instruction set architecture and the arithmetic logic unit of extensible processors is studied. The study includes modeling the circuit delay increase due to the NBTI and analyzing its impact on the processor lifetime. The study shows that in some cases, the lifetime is decreased while in other cases, it is increased compared to that of a baseline processor. Next, to lower the impact of the NBTI on the extensible processor lifetimes, we present four different techniques. The first technique is based on injecting proper input vectors during the idle time of the custom instructions such that the NBTI effect is reduced. The lifetime improvement, however, is limited by the delay increase reduction of the applied input vectors. In the second technique, the guard band delay of extensible processor is extended. This is effective because the rate of lifetime increase due to extending the guard band delay is much higher than the rate of speedup reduction. For the third technique, we duplicate the critical custom instructions to increase the processor lifetime without losing any speedup. The last technique estimates the delay increase of candidate custom instructions (CIs) and selects those that do not reduce the lifetime. The efficacies of the aforesaid techniques are evaluated and compared against each other by using benchmarks from different application domains.

1. Introduction

In digital systems, speed and power consumption are two critical design parameters [1],[2]. Although application specific ICs (ASICs) offer high performance, they are expensive and lack programmability. On the other hand, general purpose embedded processors are fully flexible and cheaper compared to ASICs. They, however, tend to fail to meet the required performance and power consumption of embedded applications. Extensible processors have emerged in the field of embedded computing as a promising approach to remedy the shortcomings of ASICs and general-purpose processors [1]. This approach exploits a simple general-purpose processor and extends its instruction set architecture with appropriate custom instructions (CIs) to provide flexibility and high performance. Hence, the methodology to design this kind of processors is called Application Specific Instruction set Processor (ASIP) methodology.

In the ASIP design approach, the instruction set of a general purpose processor is extended through ASIC design based on the features of the given application. The augmented instructions are determined such that the desired speed, power, and cost requirements are fulfilled. The main idea behind using an ASIP design approach is to run the hotspot parts of an application using custom instructions (CIs) and run the other parts of the application on the processor Arithmetic Logic Unit (ALU). The CIs, which are executed using a hardware block called Custom Functional Unit (CFU), improve the processor speed (performance and computing speed are used interchangeably in this paper) by increasing the instruction level parallelism, and reducing the register file accesses. Similarly, by decreasing the accesses to the cache and register file, the CIs reduce the power consumption [3].

When realizing digital systems with transistors provided by a state-of-the-art CMOS fabrication facility, designers encounter some critical issues including reliability concerns such as soft errors, hard errors, and process variations [4]. One type of hard errors is caused by the Bias Temperature Instability (BTI) phenomenon, which is becoming more crucial by shrinking the technology size. The phenomenon which is a transistor-aging effect is induced for the PMOS (NMOS) transistor under negative (positive) bias, and hence, is called Negative (Positive) BTI or NBTI (PBTI) [4]. The phenomenon, which alters the threshold voltage of transistors, affects both propagation delay and stability of the digital circuits. It is a more pronounced effect for the PMOS transistors suffering from extended periods of negative bias (recall that the main aging effect for NMOS transistors is the hot electron effect) [5].

* Corresponding author. Tel.: +98 2182084920; fax: +98 2188778690.
E-mail addresses: mehdikamal@ut.ac.ir (M. Kamal), afzali@ut.ac.ir (A. Afzali-Kusha), saeed@ut.ac.ir (S. Safari), pedram@usc.edu (M. Pedram).

http://dx.doi.org/10.1016/j.vlsi.2014.12.001
0167-9260/© 2014 Elsevier B.V. All rights reserved.

The simplest and most common solution to BTI is to do extra guard banding [4]. In this solution, one increases the clock period in order to tackle the propagation delay increase caused by BTI and raises the minimum voltage of storage elements to solve the instability problem. The former degrades the performance while the latter increases the power consumption of the digital circuits. When other BTI mitigating techniques can be invoked, the amount of the guard band may be reduced. It should be mentioned that the CIs are generated by combining some basic primitives (*e.g.*, ADD and SUB) of the baseline processor. Hence, the CFU contains more critical paths compared to those of the baseline processor ALU. Therefore, the probability of the maximum delay violation due to the aging effects in the CFU may become higher than that of the ALU. On the other hand, the execution load is divided between the CFU and the ALU. For a given application, the chip temperatures may be different in the case of the extensible processor and baseline processor [6]. Since BTI effect is a strong function of the temperature and load (stress), the lifetime change of the extensible processor compared to a baseline processor should be studied.

In this paper, first, the impact of the NBTI effect on the lifetime of the extensible processor by considering the processor temperature is studied. Then, we propose some techniques to increase the lifetimes of extensible processors in the cases where their lifetimes are shorter than that of the baseline processor. Some of these techniques may have been proposed in the previous works to be used in the general purpose processors. However, as the knowledge of the authors goes, in this work, these techniques, for the first time, are used in the extensible processors and their effectiveness for these types of processors is studied. It should be noted that in the case of ASIPs, the objective is to increase the speedup by adding the CFU. Since the impact of the NBTI on the CFU may lead to the lifetime reduction due to different temperatures and critical paths compared to those of the baseline processor, the different lifetime improvement techniques may have different impacts on the speedup and lifetime of the ASIPs. Hence, one of our contributions in this work is to study these impacts.

The rest of the paper is organized as follows. A brief review in the field of the instruction set architecture (ISA) extension and also NBTI effect on the processors is given in Section II. The impact of the NBTI and its modeling are described in Section III. In Section IV, the ISA extension design flow is explained. The proposed methods to decrease the NBTI delay increase are described in Section V while the results are discussed in Section VI. Finally, Section VII concludes the paper.

2. Related works

Considering the effect of NBTI on the design of processors has attracted a great deal of attention (see e.g., [4,7–9]). In these works, several aging-resilient and NBTI mitigating methods for processors have been proposed. In [4], a multi-level optimization approach for reducing the impact of NBTI on the functional units (FUs) was proposed. The technique combines methods at the circuit and micro-architecture levels. In [7], a simple holistic scheme to balance the utilization of devices in a processor is proposed. In this approach, the duty cycle ratio of circuit internal nodes and the usage frequency of devices were equalized. The impact of the NBTI on the SRAM-based register-files was thoroughly investigated in [8]. The authors showed that the periodic bit inversion was not a good solution. Instead they proposed two micro-architecture techniques based on the register rotation (RR) and bit level rotation (BR) to reduce the delay increase due to the NBTI effect. In [9], the functional units of the out-of-order processors were studied and some methods based on the power gating and bit flipping were proposed to mitigate the delay impact of the NBTI on the functional units.

In in-order processors, the execution unit determines the maximum clock frequency of the processor. Hence, the BTI-induced delay increase of this part leads to some performance loss. This delay increase may also deteriorate the speedup of the ASIPs based on in-order processors. In [10], an ISA extension design flow in the presence of the process variation and aging effects is proposed. The proposed design flow makes use of some models for the process variation and the impact of NBTI on delays of the CIs before the selection phase. The main problem with the proposed method is that, for a proper modeling of NBTI, the duty cycle of the identified CIs as well as the CFU temperature are required. These parameters are determined after the runtime of the application which depends on the speedup of the extensible processor determined by the selected CIs.

In [11], the impact of the NBTI on the processor under different task allocation policy has been studied. Also, a novel dynamic instruction allocation strategy has been proposed. The efficacy of the proposed method has been assessed under the process variation and technology scaling. To partially recover the NBTI effect, a method based on the NOP operation has been proposed in [12]. In this work, by using the ILP technique, a method to find the best NOP instruction is proposed. Using the NOP operation, proper input vectors are applied to the execution unit. To apply this method, the execution unit should be modified. Additionally, by using the complex gates (e.g., AOI gates) and also, increasing the number of the inputs of the gates, the number of the variables in the ILP formulation increases, which results in increasing the runtime of the ILP solver. In [13], a method based on the instruction classification has been proposed. Based on their worst-case delays, the instructions are classified into critical and non-critical instructions. Hence, for each class of instructions, a specific ALU is considered. Specific ALU for the critical instructions leads to more idle time for this type of instructions, and hence, smaller delay increases.

Finally, we briefly review the works in the field of ISA extension. There are many works dealing with both the CI identification (see, e.g., [1,2]) and selection (see, e.g., [1,14]). In all of the proposed identification methods, increasing the speedup of the selected CIs has been the goal while observing the I/O constraint and convexity of the selected CI. Also, in the field of the ISA selection, the main goal has been to select the best CIs in an acceptable time. The two main objectives in the selection CIs are cycle saving and area overhead. Additionally, to tackle the issues arising in nano-scales technology, there are some works which consider the impact of the process variation on the ASIP design. In [15,3], the impact of the process variation on the ASIP design was studied and some design time techniques were proposed to increase the yield and also the speedup of the extensible processor.

In this paper, we present a study of the NBTI aging impact on the extensible processor and propose four techniques for lowering the impact of the NBTI effect on the extensible processors. Note that among the four techniques, one of them is a new system-level approach which is proposed in this work. In the cases of the other three approaches, we should mention that, although, they have been studied for the general purpose processors, their efficiencies for improving the lifetimes and speedups of the extensible processors have not been investigated. In short, the originalities of this work include proposing a system-level technique as well as studying the efficiencies of three conventional approaches (previously used in the general purpose processors) for the extensible processors.

3. NBTI Effects and modeling

The BTI effect increases the absolute value of the threshold voltage of transistors, thereby reducing the transistor ON currents, I_{ON}. This lower current results in an increase in the propagation delay of the logic cells inside a VLSI circuit. This may cause a violation of the maximum propagation delay bound of the critical paths. Additionally, the noise margin may be reduced due to the threshold voltage increase, and hence, malfunctioning of digital circuits may occur. It should be noted that higher temperatures and larger electrical fields make the NBTI effect more pronounced.

The NBTI phenomenon is caused by interface states existing at the Si and gate dielectric interface of PMOS transistors [4,6]. Under negative gate bias conditions where the absolute value of the gate to source voltage (V_{gs}) is equal to the supply voltage (V_{dd}), some chemical reactions occur. The reactions, which are caused by the holes in the channel, break the weak Si–H bonds generating some traps near the gate oxide. These traps increase the threshold voltage. When the negative bias is removed, some interface traps are eliminated, which implies that the NBTI is a partially reversible process, and hence some of the threshold voltage change may be restored. Hence, we can distinguish two phases: a stress phase where the delay is increased, and a recovery phase where the delay is decreased (compared to the stress phase). Note that the increase in the delay cannot be fully recovered in the recovery phase. Hence, the delay of the gates gradually increases over time. Beside physical parameters of the transistors, the variation of the threshold voltage, V_{th}, depends on the stress and recovery times.

Based on the above discussion, the variation may be modeled as [16]

$$\Delta V_{th} = AY^n t^n \tag{1}$$

where A is a technology dependent factor which is a function of temperature, n is a constant which depends on the fabrication process, and Y is the duty cycle (also called the *stress parameter*) which is ratio of the stress time to the total time (t). To find the value of the parameter Y for each gate, we use the signal probability (SP) propagation method which has been proposed in [17]. The SP parameter for each input signal shows the ratio of the time that the signal is 1 to the total time (t). The duty cycle for PMOS transistors is equal to $1-SP$. Note that using the SP parameters of the input signals of a circuit, one can calculate the SP values of its internal gates [17]. Finally, note that the parameter A depends on several factors including the temperature. The relation between A and temperature (T) is expressed by [16]

$$A \propto e^{(-E_a/kT)} \tag{2}$$

where k is the Boltzmann constant and E_a is the activation energy of hydrogen species.

4. ISA extension flow

The ISA extension starts by extracting the data flow graph (DFG) of an application. The DFG shows the flow of the instructions which should be executed by the processor. Custom (extended) instructions are subgraphs of this DFG that meet some predefined constraints such as convexity, I/O ports, and propagation delay [1]. Among the identified sub-graphs (CIs), there may be similar sub-graphs based on the functional and structural isomorphism. These CIs can be executed on one CFU, which results in less area overhead and better area utilization. Such similar subgraphs are identified and placed into CI groups. If a DFG node is included in two CIs, these two CIs are considered to have overlap with each other. Because each node in the DFG should not be executed by more than one CI, all the CIs that have an overlap with a selected CI are removed from the list of the identified CIs in the selection phase. Based on the overlap between the CIs, a conflict graph is constructed. In the last phase (selection phase), the best CI groups are selected among all identified CIs.

There are two general approaches for the CI selection which are optimal and non-optimal approaches [1]. The former are exact methods which finds the best (optimal) CIs set by searching all the search space. This technique may become intractable for real applications where the number of nodes in the DFG of the application is too large. Hence, in this approach, we should prune the search space to make this algorithm practical. Non-optimal approaches are much faster than the optimal techniques. In the selection phase, the candidates are selected based on their merit value. Normally, cycle saving is the main objective in the ISA extension. The cycle saving may be computed as [3]

$$CS_i = \#Itr \times \left(\#CI_i.SW - CI_i.IOPenalty - ceil\left(\frac{CI_i.CriticalPathDelay}{Clock\ Period}\right)\right) \tag{3}$$

where CS_i is the cycle saving of the ith CI (CI_i), $\#Itr$ is the execution frequency of the basic block to which CI$_i$ belongs, and $\#CI_i.SW$ is the number of clocks that the CI needs to run by the base processor. Also, $CI_i.IOPenalty$ is the number of extra accesses to the register file for reading data from or writing data to (when the number of CI I/O ports is more than the number of register file read/write ports) and the fraction is the number of clocks for executing CI_i on the CFU (we assume that CIs can be multi-cycle or single cycle.) In the fraction, $CI_i.CriticalPathDelay$ is the propagation delay of the CI critical path and Clock Period is the desired clock period for the extended processor [3]. Note that, in the selection phase the CI groups are selected, hence, the merit value of a CI group is

$$Merit_{CI\ Group} = \sum_{i=1}^{|CI\ group|} CS_i \tag{4}$$

In the rest of the paper, we use the term ‘CI’ to refer to ‘CI group’.

5. Proposed liftetime improvement techniques

As mentioned in Section 3, the stress parameter (Y) is determined by the ratio of the stress time to total time. It is one of the main parameters determining the NBTI effect. The CIs selected for an extensible processor impacts both of these time durations (i.e., stress time and total time) in the cases of the CFU and the ALU. Additionally, as mentioned before, the voltage threshold degradation is an exponential function of the temperature. The selected CIs also affect the chip temperature profile, which in turn influences the impact of NBTI in different parts of the chip (see, e.g., [6]). Therefore, one should find an approach for estimating the impact of NBTI on delays of the CFU and ALU.

The ISA extension design flow augmented with a part for estimating the delay impact of NBTI (called *Calc_BTI_Effect*) is shown in Fig. 1. In this design flow, after selecting the CIs in the selection phase, they are synthesized by a gate-level synthesis tool. The gate level model is used to propagate the signal probability inside the CIs. In addition, temperatures of different parts of the processor including the selected CIs are modeled based on the application profile. Finally the impact of the BTI on the ALU and CFU are calculated by considering these temperatures. Using this information (temperature and signal probability), we calculate the threshold voltage change due to the NBTI aging effect which may be used to determine the delay increase over time. Based on this information, one can estimate the processor lifetime. Next, we present four techniques to increase the lifetime of the extensible processors in the presence of the NBTI effect.

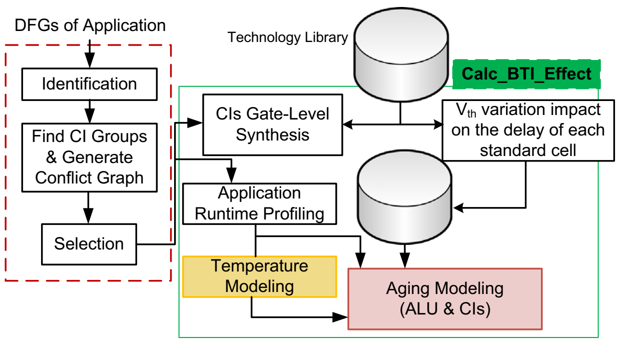
DFGs of Application
Identification
Find CI Groups & Generate Conflict Graph
Selection
Technology Library
Calc_BTI_Effect
CIs Gate-Level Synthesis
V_{th} variation impact on the delay of each standard cell
Application Runtime Profiling
Temperature Modeling
Aging Modeling (ALU & CIs)

Fig. 1. The ISA extension design flow augmented with a part for estimating the delay impact of NBTI (called *Calc_BTI_Effect*).

5.1. Input vector control technique

In this technique, we apply proper input vectors to the idle parts of the CFU and the ALU to counter the NBTI effect. This selective application of the input vectors requires some modifications of the architectures. Next, we describe the method in detail.

5.1.1. Finding proper input vectors

To obtain the input vectors, any optimization method may be utilized. In this work, we present a genetic algorithm (GA)-based method to find the (near-optimal) input vector for each *critical* custom instruction as well as the *critical* primitives of the ALU (e.g., ADD and SUB). It should be mentioned that we use the term *critical* for a path, primitive, or a CI whose delay can be degraded by the NBTI effect to a value larger than the clock period of the baseline (or extensible) processor. Note that the proposed optimization method may be applied to any digital circuits. The input vector is generated for each *critical* custom instruction as well as the *critical* primitives. Hence, each critical CI and each critical primitive have their own input vectors which are injected only during their idle times.

The proposed GA flow is depicted in Fig. 2. GA is a general-purpose powerful search algorithm used to solve hard optimization problems by simulating natural evolution over populations of candidate solutions. While the algorithm does not guarantee finding the optimal input vector, it is fast technique which can find close to optimum solutions. The algorithm has been used in many areas of the hardware design. For more details about this algorithm, one can refer to [18]. To find the best input vector, we use the binary representation for our genetic algorithm approach where each chromosome is an input vector whose length is equal to number of input bits of the circuit (see Fig. 3). In the initialization step, a binary random number is assigned to the each gene (input bit) of the chromosomes. Next, each input vector (chromosome) is applied to the circuit. Based on this input vector, the inputs to each gate of the circuit during the idle times is determined. Also, the activity of critical CIs and primitives during their active modes to determine the stress on their gates is considered. Finding the activities of the critical CIs and primitives helps finding the amount of the stress on the internal gates during the active mode leading us to finding efficient input vectors for minimizing the NBTI effect on both the active and idle modes. Using this information, we determine the impact of the NBTI effect in circuit delay increase (fitness value). The calculation is performed by using a library of pre-calculated delay increase for each standard cell after a predefined time (e.g., ten years in this work). The standard cells delay increase library was generated by using HSPICE simulation and (1). In this algorithm, in each iteration, first the termination condition (which, in this work, is the maximum

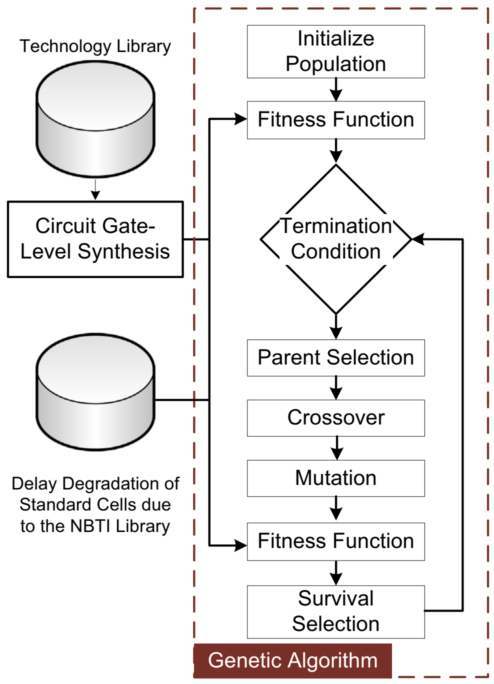

Fig. 2. The Genetic Algorithm Flow for extracting proper input vectors.

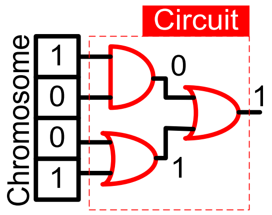

Fig. 3. An example of a circuit and its corresponding chromosome.

iteration count) is checked. Next, a set of the parents are selected by using the linear ranking method. Then, by utilizing one-point crossover and multi-point mutation methods, new offspring is generated. Next, the fitness values of the offspring are calculated. Finally, the parents and offspring with the minimum fitness values are selected as a new population which is used in the next iteration. Note that the number of the chromosomes in the new population is equal to the number of the chromosomes in the initial step. This process is iteratively run till the termination condition is satisfied.

5.1.2. Architecture modifications

In the baseline processor considered in this paper (in-order MIPS architecture), one of the inputs of the ALU (input A) is connected to three different data signals through a multiplexer. Two of these inputs belong to the forwarding path (from execution and memory stages), and the other one is from the decoder stage (from the register file). The other input of the ALU (input B) is connected to the four different data signals through a multiplexer. Three of them are similar to those of the other input while the forth data signal belongs to the immediate input data. Note that this architecture is a common architecture for in-order processors [19]. In the case of extensible processors, this input paths and the corresponding multiplexer exist for each input of the ALU and CFU. Also, in each cycle, only the proper output from either the CFU or ALU is selected using a tri-state buffer (see Fig. 4(c)). We could use

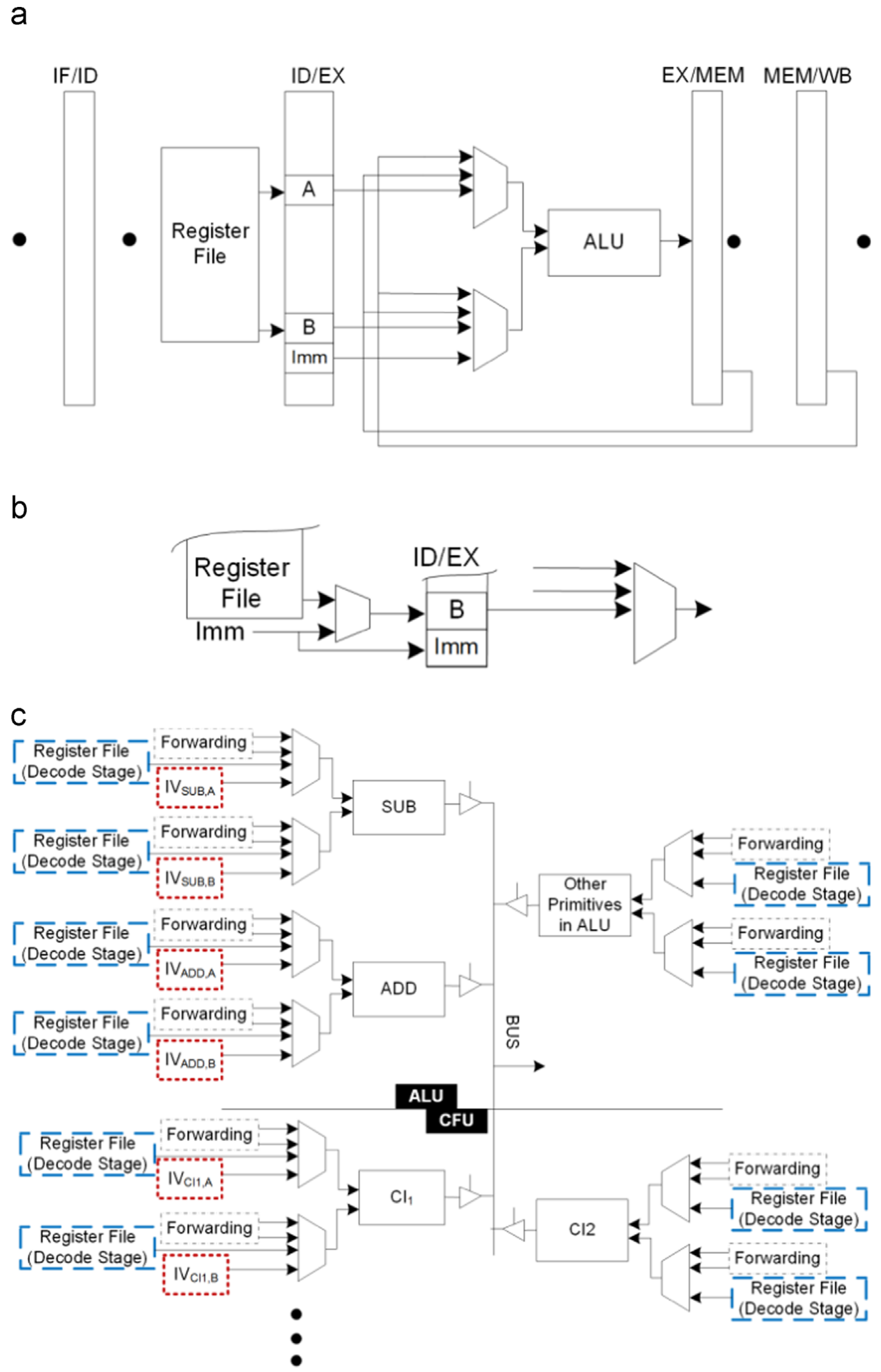

Fig. 4. (a) The architecture of the conventional in-order processors. (b) The modified path of selecting the data input of the input B. (c) The internal architecture of the execution unit of the extensible processor in the case of using ideal input vector. ADD and SUB are critical primitives and hence are separated from other primitives.

multiplexers which have higher delays than the tri-state buffers when the number of CIs is small.

In the case that the input vector control (IVC) method is used, each ideal input vector should be stored in a register to be invoked during the idle time. Hence, we should add another input to the input multiplexers of the critical primitives in the ALU and CFU. As shown in Fig. 4(a), the multiplexer of the input A has one free input. Hence, we can add another input to it without any timing overhead. On the other hand, the multiplexer of the input B has no free input, and to add extra input, we should use a larger multiplexer which has a larger delay. To prevent enlarging the multiplexer, we added a 2-input multiplexer for selecting between the immediate input data and the register file in the Instruction Decoder (i.e., ID) stage (see Fig. 4(b)). This multiplexer increases the delay of the ID stage while the maximum delay propagation of the processor is not changed. This originates from the fact that the delays of the critical paths in the execution unit are larger than those in the instruction decoder stage even after adding this multiplexer. Finally, Fig. 4(c) shows the modified architecture of the execution unit where the registers of the proper input vectors are added to the CFU and the ALU. Note that this method can be used in the baseline processor to increase its lifetime.

Finally, we should mention that, in the processor architecture, a pipelined multiplier has been used, and hence, it does not contain any critical path. If the multiplier had some critical paths, the proposed IVC method would be applicable.

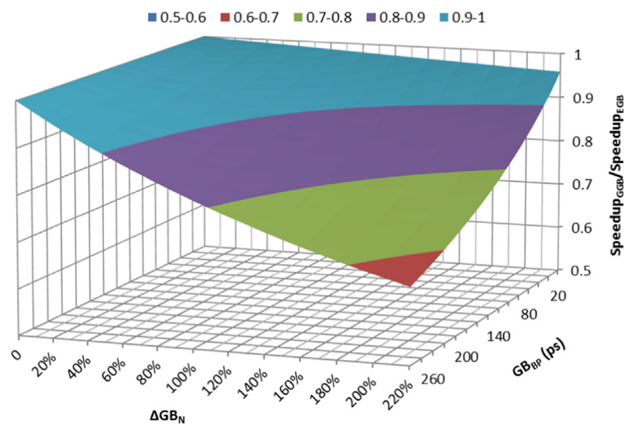

Fig. 5. Impact of the increasing guard band on the speedup reduction.

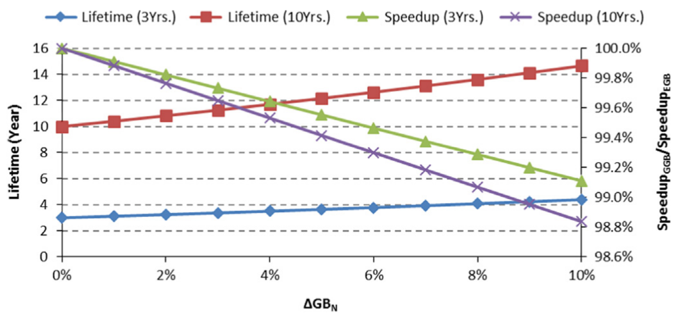

Fig. 6. Impact of the increasing guard band on the lifetime improvement and speedup reduction.

5.2. Guard band extension

In the case of baseline processors, one can add a guard band of GB_{BP} to the clock cycle period (CP) to mitigate the lifetime degradation due to the NBTI effect. Thus, we can use $CP+GB_{BP}$ as the clock period where GB_{BP} is the guard band for the baseline processor. The increase, however, degrades the performance. The same increase in the clock period does not necessarily provide the same lifetime improvement in the case of extensible processors. To obtain the same lifetime improvement, we can increase the guard band to GB_{EP} (guard band for the extensible processor). This extra delay leads to a speedup reduction. The speedup is obtained from

$$Speedup_{GGB} = \frac{(CP+GB_{BP})CC_{BP}}{(CP+GB_{EP})CC_{EP}} = \frac{(CP+GB_{BP})}{(CP+GB_{EP})} Speedup_{EGB}$$

where the $Speedup_{GGB}$ ($Speedup_{EGB}$) is the speedup of the extensible processor in the case of $GB_{EP} > GB_{BP}$ ($GB_{EP}=GB_{BP}$). Also, CC_{BP} (CC_{EP}) shows the cycle counts of running the application on baseline (extensible) processor. Now, we study the speedup reduction and lifetime improvement versus GB_{EP}. A 3D plot of $Speedup_{GGB}/Speedup_{EGB}$ versus GB_{BP} and the normalized guard band increase ($\Delta GB_N = (GB_{EP} - GB_{BP})/GB_{BP}$) is plotted in Fig. 5. As expected, for a given GB_{BP}, increasing ΔGB_N decreases the speedup. The rate of decrease is higher for greater GB_{BP} which leads to larger GB_{EP}. Also, Fig. 6 indicates the rate of lifetime change (the left vertical axis) versus ΔGB_N. The two rates shown in this plot differs in GB_{BP} values. The GB_{BP} of the lower (upper) one corresponds to the lifetime increase of three (ten) years (see the points at $\Delta GB_N = 0$). This figure shows that increasing the guard band by 10%, the lifetime increases by about 4.5 (1.2) years when for the upper (lower) rate. In addition, this figure shows the speedup reduction versus ΔGB_N (the right vertical axis). As is evident from the results, the lifetime increase rate is much more than the speedup decrease rate. Depending on the relative importance of these two parameters, the designer may decide on using the proper value for ΔGB_N.

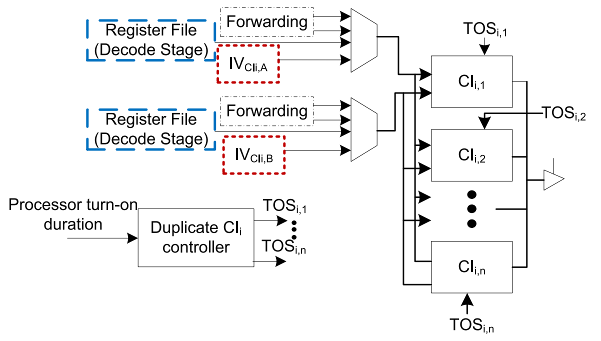
Register File (Decode Stage)
Forwarding
IV_CIi,A
Register File (Decode Stage)
Forwarding
IV_CIi,B
Processor turn-on duration
Duplicate CI_i controller
TOS_i,1
TOS_i,n
TOS_i,1
CI_i,1
TOS_i,2
CI_i,2
CI_i,n
TOS_i,n

Fig. 7. Hardware structure of the critical CI_i, while enhanced with the IVC and duplication method.

5.3. Hardware duplication

In this subsection, we propose to duplicate the hardware of critical CIs to increase the lifetime of the extensible processors. The number of duplications is a function of the target lifetime. In this method, the system starts with the first instance of the CI and when its delay increase exceeds the maximum degradation constraint, this instance becomes deactivated and the next one becomes activated. The activation and deactivation processes are realized using the power gating technique. Therefore, in this method, based on the lifetime of the ith critical CI ($CI_i.lifetime$), the number of its duplications (N_i) is determined from

$$N_i = \frac{CI_i.lifetime}{Target\ Lifetime} \tag{5}$$

While this method guarantees achieving the target lifetime without any speedup reduction, it imposes some area overhead. This technique may be combined with the input vector control method to reduce the number of required duplications. To use the power gating technique, the extensible processor should be equipped with a timer controlling the activation time of the next instance. Note that the turn-on time of the next instance must be prior to the turn-off time of the current instance. Fig. 7 shows the duplication of the critical CI_i while the IVC method is used too. Each duplicated CI is turn on and off by a turn-on signal (TOS), hence, each CI has different power domain. Also, only one set of register is needed to store the proper input vector for all duplicated CIs in a duplicated set.

5.4. NBTI aware CI selection

This technique is based on selecting CIs that are less vulnerable to the NBTI effect. This requires modifying the design flow to determine the critical CIs before the selection phase and estimate the runtime delay increase of the CIs in the selection phase. Fig. 8 shows an overview of the proposed ISA extension design flow, which is called *NBTI-Aware ISA Extension* (NAIE) design flow. In the proposed design flow, after enumerating the CIs and indicating the isomorphic CIs, the critical CIs are determined based on a rough NBTI delay increase estimation in the NBTI aware pre-selection phase. In this phase, firstly, the CIs which have the potential to become critical CIs are predicted. To choose potentially critical CIs, the delay increases of all the identified CIs are estimated based on the delay increases of the basic primitives (e.g., ADD, SUB, and SHR) of the CIs while the input stress parameter is set to 50%. The delay increases of the basic primitives are generated before calling the design flow.

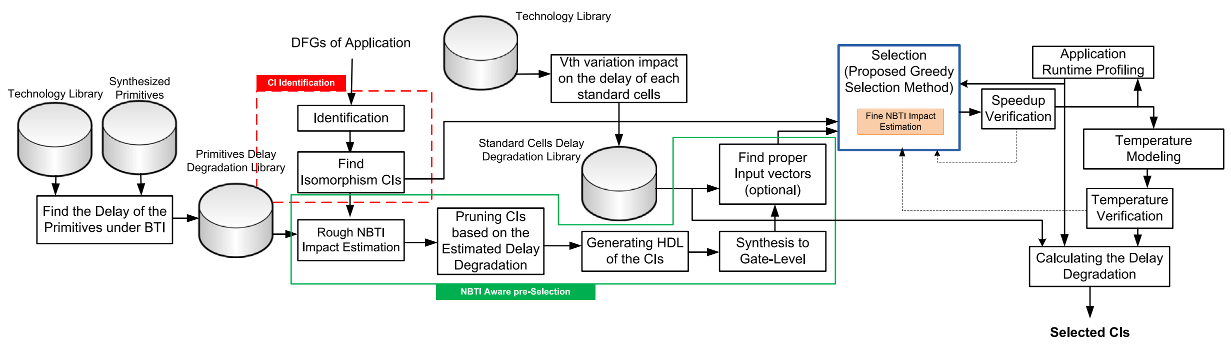
Technology Library
Synthesized Primitives
Find the Delay of the Primitives under BTI
DFGs of Application
CI Identification
Identification
Find Isomorphism CIs
Primitives Delay Degradation Library
Rough NBTI Impact Estimation
Pruning CIs based on the Estimated Delay Degradation
NBTI Aware pre-Selection
Technology Library
Vth variation impact on the delay of each standard cells
Standard Cells Delay Degradation Library
Generating HDL of the CIs
Synthesis to Gate-Level
Find proper Input vectors (optional)
Selection (Proposed Greedy Selection Method)
Fine NBTI Impact Estimation
Speedup Verification
Application Runtime Profiling
Temperature Modeling
Temperature Verification
Calculating the Delay Degradation
Selected CIs

Fig. 8. NAIE ISA extension design flow.

To estimate the delay increase of the primitives, firstly, they were synthesized as gate-level netlists. Next, the probability of the primary inputs to be 0 (or 1) was assumed to be 0.5. By propagating this probability to the internal nodes, the probabilities (and stresses) of the internal nodes are calculated [20]. Finally, by using the stress values and a pre-assumed temperature (denoted by T_{pa}), the delay increases of the primitives for a predefined number of years (e.g., 10 years) were calculated. To estimate the delay increase, the sum of the delay values of the primitives in different paths from inputs to outputs is calculated. The delay of the ith CI under the NBTI effect, which is denoted by $CI_i.Delay$, is considered as the maximum delay of the paths. The effect of temperature difference between T_{pa} and estimated runtime temperature on $CI_i.Delay$ is considered by multiplying it by the coefficient ζ. Assuming a linear relation between the delay and the voltage threshold change and considering (1) and (2), we can obtain the coefficient ζ from

$$\xi = \frac{e^{-(E_a/kT_{BP})}}{e^{-(E_a/kT_{pa})}} \tag{6}$$

where T_{BP} is the average baseline processor temperature when running the corresponding application. Next, the CIs, whose delays after the degradation are higher than $CP + \alpha GB_{ref}$, are chosen as critical CIs in this stage. Here, GB_{ref} is the reference guard band. Since the NBTI's delay impact is calculated based on the runtime temperature and stress estimations which may have some errors, we use a fudge factor of α which is between zero and one to guarantee that all CIs that may violate the selected guard band delay are considered as critical.

It should be noted that the CIs which have delays larger than $CP + GB_{ref}$ violate the timing constraints of the ASIP and may not be selected. However, the delays were obtained based on the estimated temperature and activity. When running the corresponding application, the changes in the actual activity and temperature, may lower the delay making these CIs eligible for being selected. Hence, in the proposed approach, the impact of the NBTI effect on these CIs is studied during the selection to determine if it can be selected for the CFU. Also, in the case that the estimated delay was smaller than $CP + GB_{ref}$ while the actual delay may be larger than $CP + GB_{ref}$, to prevent wrong pruning of the CI (not being selected for the studying the impact of the NBTI effect), we propose to use smaller delay constraint of $CP + \alpha GB_{ref}$. Smaller values for α yield more CIs for NBTI impact consideration and increase the running time of the ISA extension design flow. Our experiments show that the value of α should be about 0.5. In the last step of this phase, the critical CIs are synthesized to gate-level netlist which will be used to more accurately estimate the NBTI degradation of the delay during the selection phase. As an option, the netlist may be used to lower the delay increase of the critical CIs using the IVC method.

```
1:  GreedySelection(CI Candidate Set)
2:    While (The CI canidate set is not empty)
3:      Select the CI group with highest CS
4:      Remove Local Conflicts
5:      Estimate the Delay Increase of the CI group
6:      If (Delay Increase > Predefined Guard Band)
7:        Remove the CI members of the CI group from the CI candidate set
8:        GOTO Line 2
9:      END If
10:     Select the CI Group
11:     Remove CIs overlapping with the members of the selected CI group
                                                   from the CI candidate
12:   END DO
13: END
```

Fig. 9. The pseudo-code of the proposed greedy selection method.

In the next phase, we perform the CI selection phase based on our proposed greedy selection method shown in Fig. 9. It is the modified version of the greedy method proposed in [14]. The selection is started by choosing the CI group with the highest cycle saving. It is possible that the CIs in a CI group have overlap with each other. Hence, before estimating the NBTI's delay impact, the local conflicts between the CIs in the suggested CI group are removed (Line 4). To remove the conflicts, the CIs with the lowest CS and also large number of the local conflicts are eliminated from the CI group using a merit value of $CS/(Local\ Conflict\ \#)$. This process is continued until there is no overlap between any two CIs in the CI group.

Next, the delay increase of the critical CIs due to the NBTI is estimated by using the method which will be described later in this part. If the delay increase of the selected CI group (in the case where the CI is critical) is higher than the $\mathrm{CP} + \mathrm{GB}_{\mathrm{ref}}$, then all CIs from the group will be removed from the candidate set (Lines 6–9). Otherwise, all the CIs in the group are marked as selected and removed from the candidate set. In addition, all the CIs in the candidate set which have overlaps with the selected CIs are removed from the set (Line 11). The selection process is terminated when the candidate CI set becomes empty.

Now, we explain how the delay increases of the critical CIs are estimated. The ΔV_{th} for each gate in the netlist of the CI is calculated using (1) by using the gate activity and temperature. It is obvious that during the design time these values are not known a priori. To estimate the stress, we propagate the activities of the primary inputs to estimate the stresses of the internal nodes. To find the stress of a gate, we need to determine the probability that each input signal is zero. The weighted mean that a primary input signal is zero is obtained from

$$Y = \frac{T_{Idle} \times Y_{Idle} + T_{EXE} \times Y_{EXE}}{T} \tag{7}$$

where T_{Idle} (T_{EXE}) is the duration that the CI is in the idle time (being executed) and Y_{Idle} (Y_{EXE}) is the zero probability of the signal in the idle time (during the execution). Note that the summation of the T_{Idle} and T_{EXE} is equal to application runtime which is denoted by T. For the primary inputs, if the input vector control technique is used, we calculate Y_{Idle} based on the optimized input vectors which should be applied when the CI is not utilize, else, it is assumed to be 0.5. The variable T_{EXE} is considered to be equal to the number of times that the CI is called. This is calculated based on the number of times that the basic blocks to which the CI belongs are called. In this phase, Y_{EXE} is calculated based on the application profile. The variables T_{Idle} and T depend on the runtime of the application on the extensible processor. Hence, the exact values of these parameters depend on the speedup of the extensible processor. Hence, we propose to use the speedup of the extensible processor in the case where no techniques is used to reduce the NBTI effect on the CFU (denoted by SpU_{NT}). It means that we need to run the design flow in Fig. 1 one time before estimating the NBTI effect on the CIs. The reason for this suggestion is that we do expect that SpU_{NT} is about the same as the speedup of the selected CFU when the NAIE method is exploited. Hence, T and T_{Idle} are obtained from

$$T = SpU_{NT} \times ACC \tag{8}$$

$$T_{Idle} = SpU_{NT} \times ACC - T_{EXE} \tag{9}$$

where the ACC is the total application cycle count. Also, for the temperature, we use the value for the case where no NBTI effect reduction technique is used (denoted by $Temp_{NT}$). Now, using (1), estimated Y, and $Temp_{NT}$, one can calculate ΔV_{th}. Having found the voltage threshold change, the delay increases of the gates are modeled by making use of a library which contains the delay variation of the standard cells as a function of V_{th}. This library is generated by HSPICE simulation of all standard cells.

As shown in Fig. 8, there is a speedup verification process for the selection phase. After we reach the end of the selection phase, we compare the speedup of the selected CFU (SpU) with SpU_{NT}. If there is a non-negligible difference (more than 1–2%) between the two speedups, we repeat the selection phase with SpU instead of SpU_{NT}. The process is repeated until the difference becomes negligible. This process is important because the speedup impacts the stress parameter, and hence, the delay of the CIs increases. Similarly, the temperature may need to be corrected. The more accurate temperature may be obtained using the Temperature Modeling module (see Fig. 8). In this phase, firstly, the temperature of the extensible processor is extracted. If the temperature difference is non-negligible (i.e., it is more than1–2 °C), then the temperature will be updated and the selection phase will be repeated. For the benchmarks studied in this work, the temperature and activity (speedup) errors were reduced as the iteration number increased. This may be justified by noting that the critical CIs formed about, on average, 25% of the total selected CIs in the case of the NAIE technique. Since the fraction is small, its effect on the temperature and speedup variations is not large. For the cases that the errors do not reduce as the iteration number increases, one may use the new temperature of the next iteration from the relations

$$T_{pa,i+1} = T_{pa,i} + \gamma \Delta T_{pa} \tag{10}$$

where $T_{pa,i+1}$ is the temperature which should be used for the $(i+1)^{th}$ iteration, $T_{pa,i}$ is the temperature was used in the ith iteration, and γ is a number between 0 and 1 determined for the benchmark. A similar approach may be used for the speedup. The approach provides the convergence for the NAIE technique. Note that the speedup convergence loop is nested inside the temperature convergence loop.

As mentioned before, the temperature of CFU may increase the chip temperature and consequently the ALU temperature. This temperature increase could enhance the NBTI's delay impact reducing the lifetime. These cases happen when the temperature effect is more than the ALU stress reduction due to the use of the CFU. In our processor, the critical paths in the ALU belong to the ADD and SUB primitives. We call these primitives as critical primitives. To prevent these cases, we can reduce the ALU stress by selecting CIs with more critical primitives. This may be achieved by modifying the merit function in the selection phase. In this work, instead of using merely CS as the merit function (the conventional approach), we include the number of the critical primitives in the merit function as follows

$$Merit_{CI\ Group} = \left(\frac{|CriticalPrimitives_i|}{\max(\{\forall j \in \{CI\ Groups\},\ |CriticalPrimitives_j|\})} \right) \times \sum_{i=1}^{|CI\ group|} CS_i \tag{11}$$

where the $|CriticalPrimitives_i|$ shows the number of the critical primitives in the ith CI group. In this function, we normalize the critical primitive number to the maximum number of the critical primitives in whole CI groups. While this merit function could reduce the stress on the ALU critical paths, the speedup of the selected CFU may also decrease. Therefore, the use of this merit function is recommended when the reference guard band is violated by the ALU delay increase.

6. Results and discussion

In this section, we evaluate the efficacies of the proposed techniques using several benchmarks from different application domains. The selected benchmarks included *IPsec* and *MD5* from PacketBench [21], *lms* from the SNU-RT benchmark suits [22], *rijndael*, *sha*, *bitcounter*, and *adpcm* from MiBench [23], and *G721 decoder/encoder* from Mediabench [24].

6.1. Simulation setup

The proposed design flows and algorithms were implemented in the C# language. Using GCC (GNU Complier Collection), DFGs and hotspots of these applications were generated and fed to the ISA extension design flow. In the identification phase of the ISA extension, we used the exact method proposed in [2], and for the selection phase we used the greedy selection method described in [14]. The stress and activity time of the ALU were extracted by profiling the running applications on the baseline (a 32-bit in-order MIPS) processor. The applications were run on the simulator model of the processor where the activity of the ALU was profiled. By using the DFG of the application and the profiled data of the baseline processor ALU, the activities of the ALU and CFU of the extended processor were calculated. We used Synopsys design compiler to extract the power consumption of the baseline processor and the CIs. Also, the temperature of the processors was modeled by employing the Hotspot tool [25]. Note that the Hotspot tool considers the material boundary conditions.

All studies were performed using the nangate standard CMOS 45 nm technology [26]. To find the gate-level of the ALU and CFUs, we used Synopsys Design Complier where the maximum propagation delay constraint of 1 ns was assumed. Note that, in the identification phase, also the delay constraint of 1 ns was considered where single cycle CIs were identified. To find the NBTI impact, for each standard cell of the technology library, we performed HSPICE simulations to obtain the delay as a function of the threshold voltage. To find the delay degradation of each component of the processor (i.e., CIs and ALU), first, we calculated the voltage threshold increase of the gates of that component due to the NBTI effect using (1) after Y years and found their corresponding delay degradations. Next, the delay degradation of the component was extracted using the static timing analysis.

For the NBTI effect study, we supposed a lifetime of 10 years for the baseline processor, and hence, the value of the guard band of the processor was set accordingly for each application. Also, we set GB_{ref} of the extensible processor equal to the guard band of the processor. Hence, the aging effect was studied for each benchmark separately. It should be noted that the proposed lifetime improvement techniques were only applied to the benchmarks where the extensible processor lifetime became smaller than that of the baseline processor. Also, the target lifetime of 10 years is used as one of optimization constraints in the proposed techniques.

6.2. NBTI's delay impact on extensible processors

To study the impact of NBTI on performance of the extensible processors, first, we ran the proposed design flow shown in Fig. 1 for all the benchmarks. The speedup of the extensible processor in each case is shown in Fig. 10. The maximum speedup belongs to the *IPsec* (~4.28) and the minimum belongs to the *rijndael* (~1.3). Table 1 shows the number of candidate CIs, number of CI groups, number of selected CIs and finally the area usage of the selected CIs for each benchmarks. Note that the area usage of the baseline processor was 35,000 μm^2 with the ALU area usage of 9400 μm^2.

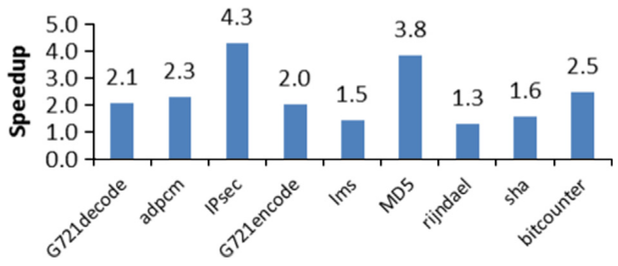

Fig. 10. Speedup of the extensible processor.

Table 1
Number of the candidate CIs, CI Groups, and selected CIs along with area of the Selected CIs.

benchmark	|Candidate CIs|	|CI Groups|	|Selected CIs|	Area (μm^2)
lms	42	33	13	4151
adpcm	62	44	13	4358
G721encode	1042	686	36	14,400
G721decode	1077	609	32	14,390
sha	1519	777	21	10,680
IPsec	19,927	14,953	28	14,244
bitcounter	39,832	34,153	36	8480
MD5	48,676	29,366	35	17,835
rijndael	425,260	32,721	23	9212

This baseline processor was used in the cases of the extensible processors which had different CFUs.

The delay increases of the ALU of the baseline processor and the ALU and CFU of the extensible processor after 10 years are reported in Fig. 11. The results show in four cases, the delay increases of the ALU and CFU of the extensible processor are smaller than that of the baseline processor ALU. In the cases of *G721decode*, *G721encode*, *IPsec*, *MD5 and sha*, the delay increase of the extensible processor CFU is higher than the baseline processor ALU delay increase. Therefore, it can be concluded that the extended ISA can both reduce and enlarge the NBTI delay effect and increase and decrease the processor lifetime. The lifetime of the extensible processor for each application is shown in Fig. 12. As mentioned before, the guard band for each application is considered equal to the delay increase of the corresponding baseline processor after 10 years. The highest lifetime decrease with respect to the baseline processor belongs to *MD5* (−50%) while the highest lifetime increase is for *rijndael* (+45%). We attribute these different behaviors for different benchmarks to different selected CFUs for these benchmarks.

Also, as mentioned before, the NBTI effect is a strong function of the temperature. In Table 2, we have reported the temperatures of the ALU in the baseline processor, and ALU and CFU in the extensible processor. The results show that the temperatures of the CFU and ALU may be higher in the extensible processor compared to that of the baseline processor ALU. This originates from the fact that the power density of the CFU may be larger yielding a higher temperature. Also, since the CFU and ALU placed next to each other, the increase in the temperature of the CFU, gives rise to the temperature increase of the ALU too.

6.3. Input vector control technique

As mentioned before, one solution to reduce the NBTI delay increase is to apply proper input vector to the critical CIs and critical primitives of the ALU during the idle time. As an example, in Fig. 13, the delay increase of a CI after 10 years under different input vector obtained in the different iterations of the proposed genetic algorithm approach is shown. For this specific CI with the

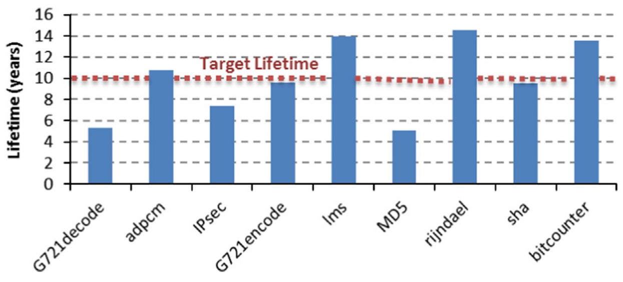

Fig. 12. Lifetime of the extensible processor.

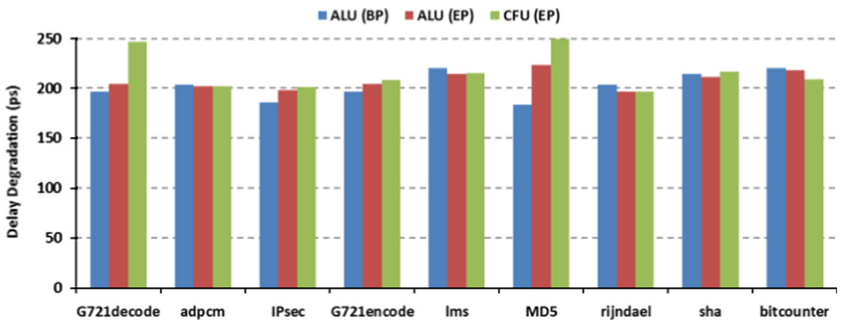

Fig. 11. NBTI induced delay increase of the ALU and the CFU in the case of baseline (BP) and extensible (EP).

initial delay of 0.88 ns, the proposed approach finds an input vector which is able to reduce the delay increase of the CI from 26.9% to 17%.

To study the efficacy of the IVC technique, we have applied the technique to both CFU and ALU of the extensible processor. Note that the CFU may have both critical and non-critical CIs. Firstly, let us consider the case that we apply the technique to all the CIs. In Fig. 14, we compare the delay increases with and without using the techniques. The results show that for all the benchmarks, the delay increases of the ALU and CFU are reduced when the IVC method is used. The highest (lowest) reduction in the ALU delay increase belongs to the *IPsec* (*G721decode*) benchmark with the value of 35% (19%). In the case of the CFU, the largest (smallest) reduction in the CFU delay increase belongs to the *lms* (*MD5*) benchmark with the value of 27.7% (21.7%). The average values for the ALU and CFU cases are about 25.2% and 25%, respectively.

Table 2
Temperature of the ALU in baseline processor, and ALU and CFU in the extensible processor

Benchmark	Temperature (°C)		
	Baseline processor ALU	Extensible processor	
		ALU	CFU
adpcm	69.37	68.27	67.85
G721decode	66.65	69.76	70.05
G721encode	66.6	69.68	69.98
IPsec	61.91	66.92	67.85
lms	75.85	73.85	72.85
MD5	61.12	77.44	79.33
rijndael	69.43	66.09	65.18
sha	73.99	72.66	71.97
bitcounter	75.56	75.2	75

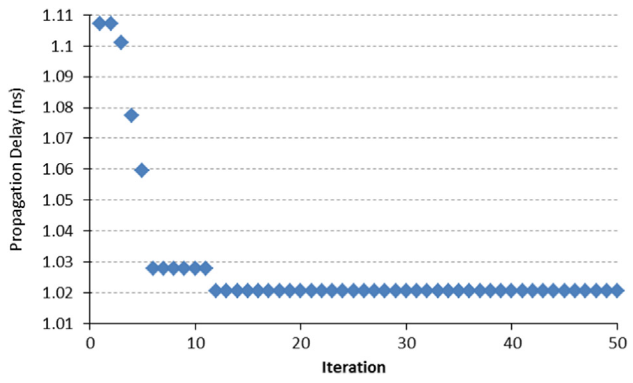

Fig. 13. Propagation delay of a CI after 10 years. The initial delay for this CI was 0.88 ns. Different input vectors are obtained during the GA algorithm iterations.

These delay increase reductions resulted in lifetime improvements which are demonstrated in Fig. 15 show the lifetime of the extensible processor. The results show that in all the cases, except for the *MD5* benchmark, the lifetimes of the extensible processors become larger than the baseline processor (more than 10 years). For the four cases of the *adpcm*, *lms*, *rijndael* and *bitcounter*, the lifetime improvements are larger than 200%. In the case of the *MD5* benchmark, the IVC technique improves the lifetime of the extensible processor from 5 years to 9.5 years which is still shorter than the target lifetime of 10 years. This indicates that the IVC method may not able to provide us with the target lifetime for all the benchmarks. The average lifetime for the benchmarks considered in this work is 25 years. While the technique does not have any speedup reduction, it needs some registers to store proper input vectors. The area overhead of these registers compared to the area usage of the CFU is shown in Fig. 16. The worst (best) case belongs to the *adpcm* (*MD5*) where the area overhead is 91% (57%). The average overhead is about 69%.

Now, we consider the case where the IVC technique is applied to the critical CIs only. This reduces the area overhead. For this study, the extensible processors with lifetime of more than 10 years were excluded. We show the fraction of the critical CIs to the total selected CIs in Fig. 17(a) where the worst (best) case belongs

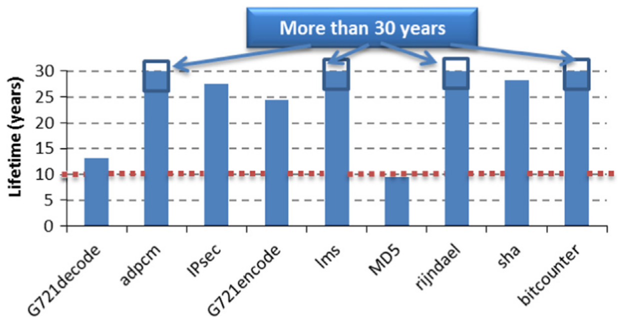

Fig. 15. Lifetime improvement of the extensible processor enhanced with the IVC method compared to the baseline processor.

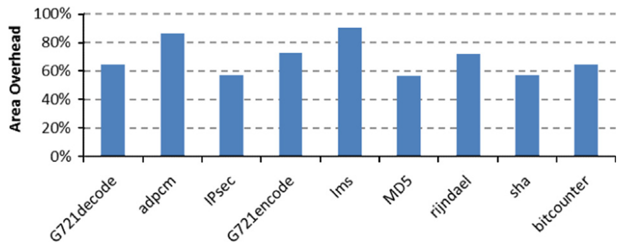

Fig. 16. Area overhead of the IVC technique while it is used for all selected CIs.

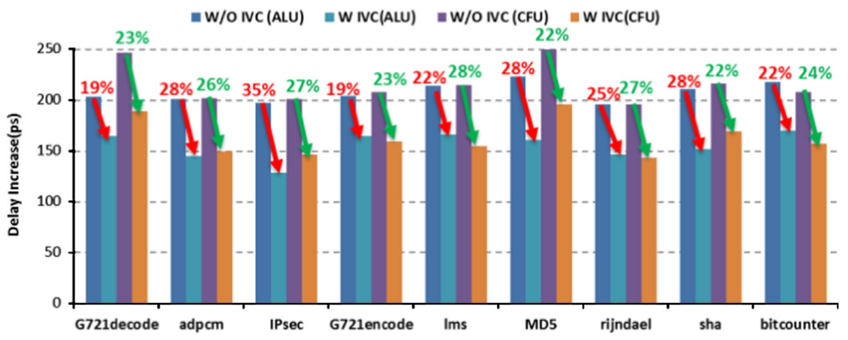

Fig. 14. Delay increases of the ALU and CFU of the extensible processor with and without the IVC method.

to the *MD5* (*sha*) benchmark with a fraction of 74% (14%). On average, 44% of the selected CIs were critical. The area overheads in this case are shown in Fig. 17(b). In the worst case, the area overhead is about 42.2% which belongs to the *MD5* benchmark. The average overhead reduction is about 27%.

Fig. 17(c) shows the lifetime of the extensible processors in this case. Note that most of the lifetimes are shorter than those of the case where the IVC method is applied to all the CIs. This means that the non-critical CIs whose delay increases are not mitigated by the IVC technique determine the lifetime of the extensible processor. In the case of the *MD5* benchmark, the lifetime is specified by the critical CIs. The average lifetime is about 47% lower than that of the case of Fig. 15.

6.4. Guard band extension

As mentioned before, by using this approach, the performance of the system is reduced. The performance loss rate versus the lifetime improvement is considerably lower. As an example, we have plotted the lifetime improvement and speedup reduction of the *IPsec* benchmark versus the guard band increase in Fig. 18. For this benchmark, by increasing the guard band about 14 ps, the lifetime increases by about 35% while the speedup decreases almost −1.4%.

Fig. 19 shows the speedup reduction of the five benchmarks whose lifetimes were smaller than the 10 years when their guard bands were equal to their corresponding GB_{BP}. By increasing the guard bands, we made their lifetimes equal to 10 years. The worst case belongs to *MD5* benchmark where the speedup reduction is about –5.6%. The average of the speedup reduction for the benchmarks is about –2.4% while, on average, the lifetimes are increased about 26%.

6.5. Hardware duplication

In this subsection, the efficacy of the CI duplication method is studied. Table 3 shows the number of the critical CIs for each benchmark. Our study showed that we only need an additional instance of the hardware for each critical CI to achieve the target lifetime. The area overhead corresponding to these additional instances are also listed in the table. The percentage of the area overhead compared to that of the CFU area is also given in the table. The average area overhead is about 48% of the CFU area usage. The largest overhead corresponds to the case of the *MD5* benchmark (about 73%) while the smallest belongs to the *sha* benchmark (about 21%). Although this method does not suffer from the speedup reduction or failure in reaching the target lifetime, it suffers from a large area overhead.

It should be noted that this technique may help the IVC method which was not able to provide the target lifetime for the *MD5* benchmark. In this case, the CIs, which are remained critical after the application of the IVC method, may be duplicated where in this case the area overhead becomes 60%.

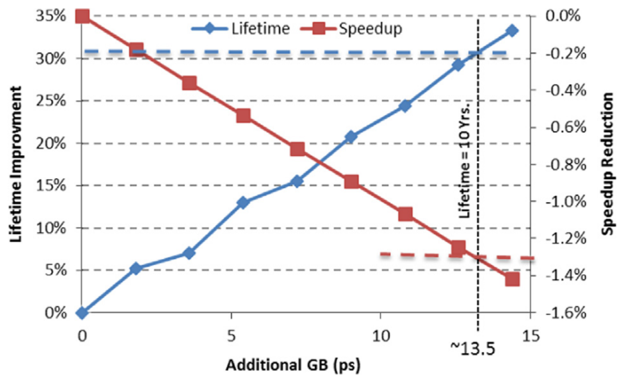

Fig. 18. Impact of increasing the Guard band on the speedup reduction and lifetime improvement.

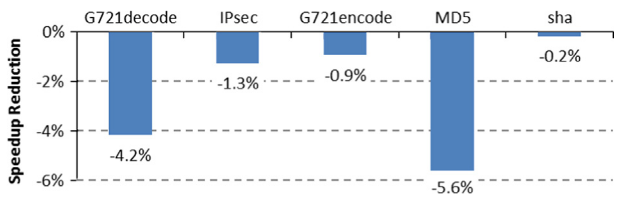

Fig. 19. Speedup reduction due to the using guard banding technique

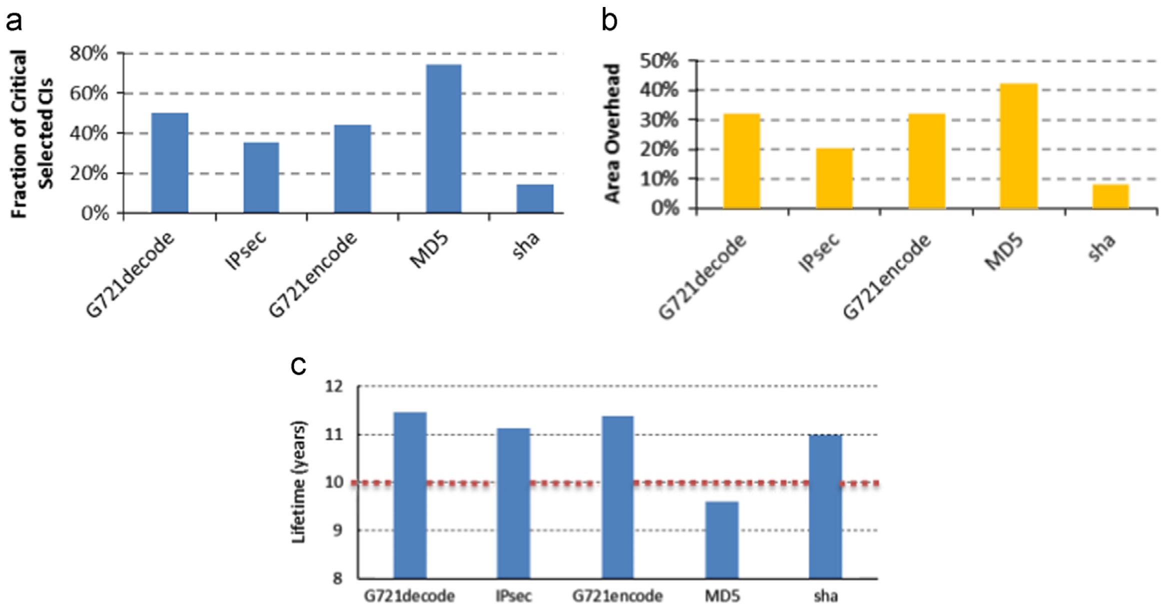

Fig. 17. (a) Fraction of the critical selected CIs, (b) the area overhead of the input vector control method and (c) the lifetime of the extensible processor when the IVC method is applied to the critical CIs.

Table 3
Number of critical CIs, and area overhead of the duplicated CIs for each benchmark

Benchmark	No. of Critical CIs	Area overhead (μm^2)	Area overhead (%)	Lifetime (years)
G721decode	16	8696	60	11.45
G721encode	16	6926	48	11.45
IPsec	9	5461	38	11.12
MD5	26	12,944	73	12.2
sha	3	2289	21	11

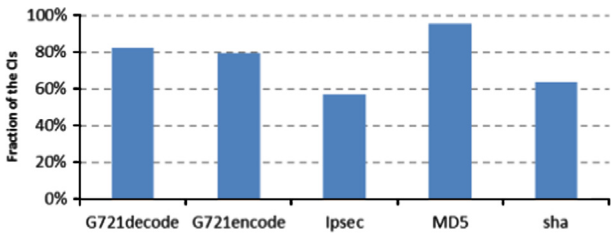

Fig. 20. Fraction of the identified CIs that recognized as critical in NBTI aware pre-selection phase.

Table 4
The speedup, delay increase, and lifetime of the benchmarks while using the Naie Isa extension design flow.

Benchmark	Speedup	Speedup reduction[a] (%)	Delay increase Compared to GB_{BP} (%)		Lifetime (year)
			ALU	CFU	
G721decode	1.56	24.8	−0.37	−5.10	13.3
G721encode	1.53	25.0	−0.18	0.00	10
IPsec	3.78	11.8	0.11	−15.68	9.5
IPsec (PMF)	3.36	21.6	−6.7	−3.0	16.6
MD5	3.77	2.0	0.00	0.00	10
sha	1.52	2.5	−1	−1.2	11.8

[a] Speedup reduction compared to conventional ISA extension design flow.

6.6. NBTI aware CI selection

In this subsection, the efficacy of the NAIE technique in increasing the lifetime of the extensible processor is evaluated. In this study, we only included the benchmarks whose delay increases were more than the reference guard band. As shown in Fig. 8, in the pre-selection phase, the potentially critical CIs are determined. The fraction of these CIs to the total identified CIs are drawn in Fig. 20. The results show that, in the worst case of the *MD5* benchmark, more than 95% of the CIs are potentially critical while in the best case of the *IPsec* benchmark, only 57% of the identified CIs are potentially critical. On average about 75% of the identified CIs are critical.

Having found the potentially critical CIs, the last phase of the NAIE ISA extension design flow was performed to select the better CIs. In Table 4, we have reported the speedups and lifetimes of the benchmarks obtained from the NAIE flow. Note that the results were obtained using the conventional merit function. Except for the *IPsec* benchmark, in all other benchmarks, the delay increase of the CFU became smaller or equal to the GB_{ref} proving us with the lifetime of equal or greater than 10 years. In the case of the *IPsec* benchmark, the target lifetime was not achieved because of larger delay increase of the ALU. While the stress on the ALU in the extensible processor was smaller than that in the baseline processor, the ALU temperature increase in the extensible processor gave rise to a higher delay increase. In the case, we ran the NAIE method again with the merit function proposed (PMF) given in (11). This improved the lifetime to more than 10 years (see *IPsec* (*PMF*) in Table 4) by forcing the selection of the CIs with more critical primitives. This discriminative selection yields some speedup reduction for the *IPsec* benchmark. As observed from the results, the use of the NAIE ISA extension design flow reduces the speedups in all the cases. The worst case is for *G721decode* (25%) while the best case is for *MD5* (2%). The use of the IVC technique (optional in the NAIE flow), which enlarges the CI exploration space, may improve the speedup (and lifetime) at the expense of some area overhead. For example, in the cases of *G721decode* and *MD5*, the speedup reductions decrease to 0% and 0.3%, respectively. In the former case, the application of the technique makes all the potentially critical CIs to non-critical CIs.

6.7. Pros and cons comparison of the techniques

The area overhead of the increasing the guard band is zero while it leads to some speedup reduction. On the other hand, the area overhead of the IVC is small while the amount of its lifetime improvement is limited. In other words, the IVC technique may not improve the lifetime as much as is desired. In the case of the duplicated method, the area overhead is large while one can increase the lifetime to any target lifetime. In the cases of the latter two, there is no speedup reduction. Finally, *NAIE* technique does not impose any area overhead while its lifetime improvement may not as much as is desired. It should be noted that all of these techniques may be combined with each other. The designer should invoke any of these techniques or a combination of them considering the target lifetime, speedup, and area overhead.

7. Conclusion

In this work, we investigated the impact of the NBTI effect on CFU and ALU of the extensible processor. Based on the NBTI model, the lifetimes of the extensible processors were estimated. The results showed that in some cases the NBTI effect on the extensible processor was smaller/larger than that of the baseline processor increasing/decreasing the lifetime compared to that of the baseline processor. To tackle the delay increase in extensible processors, we proposed four different techniques. The first one was based on the input vector control technique during the idle time of the CIs. The results showed this technique improved the lifetimes of the benchmarks. In the second approach, we proposed to increase the delay guard band to increase the lifetime. We showed that the rate of increasing the lifetime was much more than the rate of decreasing the speedup. In the third technique, we proposed to duplicate the critical CIs along with power gating to increase the lifetime. This method was able to increase the lifetime without decreasing the speedup at the expense of some area overhead. Finally, an ISA extension design flow which selected CIs based on the estimation of the delay increase of the CIs was proposed. Finally, note that one may use a combination of these techniques to further improve the efficiency.

Acknowledgment

This research was in part supported by a grant from IPM (No. CS1392-4-27).

References

[1] C. Galluzi, K. Bertels, The instruction-set extension problem: a survey, ACM Trans. Reconfig. Technol. Syst. 4 (2) (2011) 18-1–18-28.

[2] L. Pozzi, K. Atasu, P. Ienne, Exact and approximate algorithms for the extension of embedded processor instruction sets, IEEE Trans. Comput.-Aided Des. 25 (7) (2006) 1209–1229.
[3] M. Kamal, A. Afzali-Kusha, S. Safari, M. Pedram, An architecture-level approach for mitigating the impact of process variations on extensible processors, in: Proceedings of the Design, Automation and Test in Europe (DATE), 2012, pp. 467–472.
[4] T. Siddiqua, and S. Gurmurthi, A Multi-Level Approach to Reduce the Impact of NBTI on Processor Functional Units, in: Proceedings of the 20th symposium on Great lakes symposium on VLSI(GLVLSI), 2010, pp. 67–72.
[5] W. Wang, V. Reddy, A.T. Krishnan, R. Vattikonda, S. Krishnan, Y. Cao, Comapct modeling and simulation of circuit reliability for 65-nm CMOS technology, IEEE Trans. Device Mater. Reliab. 7 (2007) 509–517.
[6] H. Lin, et al., Thermal-aware design considerations for application-specific instruction set processor, in: Proceedings of Symposium on Application Specific Processors (SASP), 2008, pp. 63–68.
[7] E. Gunadi, A.A. Sinker, N.S. Kim, M.H. Lipasti, Combating aging with the colt duty cycle equalizer, in: Proceedings of the 43rd Annual IEEE/ACM Internation Symposium on Microarchitecture (MICRO), 2010, pp. 103–114.
[8] S. Kothawade, K. Chakraborty, S. Roy, Analysis and mitigation of the nbti aging in register file: an end-to-end approach, in: Proceedings of the 12th International Symposium on Quality Electronic Design, 2011, pp. 1–7.
[9] L. Li, Y. Zhang, J. Yang, J. Zhao, Proactive NBTI mitigation for busy functional units in out-of-order microprocessors, in: Proceedings of the Design, Automation and Test in Europe (DATE), 2010, pp. 411–416.
[10] Y. Hara-Azumi, F. Firouzi, S. Kiamehr, M. Tahoori, Instruction-set extension under process variation and aging effects, in: Proceedings of the Design, Automation and Test in Europe (DATE), 2013, pp. 182-187.
[11] S. Corbetta, and W. Fornaciari, NBTI mitigation in microprocessor designs, in: Proceedings of Symposium on Great lakes symposium on VLSI(GLVLSI), 2012, pp. 33–38.
[12] F. Firouzi, S. Kiamehr, and M.B. Tahoori, NBTI mitigation by optimized NOP assignment and insertion, Proceedings of the Design, Automation and Test in Europe (DATE), 2012, pp. 218–223.
[13] F. Oboril, F. Firouzi, S. Kiamehr, M.B. Tahoori, Reducing NBTI-induced processor wearout by exploiting the timing slack of instructions, in: Proceedings of the CODES+ISSS, 2012, pp. 443–451.
[14] P. Bozini, L. Pozzi, Recurrence-aware instruction set selection for extensible embedded processors, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 16 (2008) 1259–1267.
[15] M. Kamal, A. Afzali-Kusha, M. Pedram, Timing variation-aware custom instruction extension technique, in: Proceedings of the Design, Automation and Test in Europe (DATE), 2011, pp. 1517–1520.
[16] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, Y. Cao, The impact of NBTI effect on combinational circuit: modeling, simulation, and analysis, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 18 (2) (2010) 173–183.
[17] S.V. Kumar, C.H. Kim, and S.S. Sapatnekar, NBTI-aware synthesis of digital circuits, in: Proceedings of 44th Design Automation Conference (DAC), 2007, pp. 370–375.
[18] J.M. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[19] D.A. Patterson, J.L. Hennessy, Computer Organization and Design the Hardware/Software Interface, Morgan Kufmann, Canada, 2011.
[20] M. Pedram, Power minimization in IC design: principles and applications, ACM Trans. Des. Autom. Electron. Syst. 1 (1) (1996) 3–56.
[21] R. Ramaswamy, T. Wolf, PacketBench: a tool for workload characterization of network processing, in: Proceedings of the IEEE International Workshop on Workload Characterization, October 2003, pp. 42–50.
[22] SNU-RT Real Time Benchmarks. Available: 〈http://archi.snu.ac.kr/realtime/benchmark/〉.
[23] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, R.B. Brown, MiBench: a free, commercially representative embedded benchmark suite, Proceedings of the International Workshop on Workload Characterization, 2001, pp. 3–14.
[24] C. Lee, M. Potkonjak, and W.H. Mangione-Smith,MediaBench: a tool for evaluating and synthesizing multimedia and communications systems, in: Proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture, 1997, pp. 330–335.
[25] W. Huang, K. Sankaranarayanan, R.J. Ribando, M.R. Stan, K. Skadron, Accurate, pre-RTL temperature-aware processor design sing a parameterized geometric thermal model considerations, IEEE Trans. Comput. 57 (9) (2008) 1277–1288.
[26] FreePDK, AFree OpenAccess 45 nm PDK and Cell Library for University, 〈http://www.eda.ncsu.edu〉.

Mehdi Kamal received his B.Sc., M.Sc. and Ph.D. in Computer Engineering from Iran University of Science and Technology, Sharif University of Technology, and University of Tehran in 2005, 2007, and 2013, respectively. Currently, Dr. Kamal is a research associate of the Low-Power High-Performance Nanosystem Laboratory at school of Electrical and Computer Engineering at the University of Tehran. His research interests include Reliability in nano-scale design, ASIP design, HW/SW co-design, and Low power design.

Ali Afzali-Kusha received his B.Sc., M.Sc., and Ph.D. degrees all in Electrical Engineering from Sharif University of Technology, University of Pittsburgh, and University of Michigan in 1988, 1991, and 1994, respectively. From 1994 to 1995, he was a Post-Doctoral Fellow at The University of Michigan. Since 1995, he has joined The University of Tehran, where he is currently a Professor of the School of Electrical and Computer Engineering and the Director of Low-Power High-Performance Nanosystems Laboratory. Also, on a research leave from the University of Tehran, he has been a Research Fellow at University of Toronto and University of Waterloo in 1998 and 1999, respectively. He is a senior member of IEEE, and his current research interests include low-power high-performance design methodologies from the physical design level to the system level for nanoelectronics era.

Saeed Safari received his Ph.D. degree in Computer Architecture from Computer Engineering Department, Sharif University of Technology, Tehran, IRAN, in 2005. Since then, he has been a faculty member of Electrical & Computer Engineering Department, University of Tehran, Tehran, Iran. From May 2009 to September 2010, he collaborated with TeleRobotics and Applications (TERA) Lab., IIT, Genoa, Italy, working on different aspects of low-power parallel implementation of machine vision applications. His research interests are Fault Tolerant System Design, High Performance Computing, Test and Design for Test, On-chip Interconnection Networks, and Computer Architecture.

Massoud Pedram, who is the Stephen and Etta Varra Professor in the Ming Hsieh department of Electrical Engineering at University of Southern California, received a Ph.D. in Electrical Engineering and Computer Sciences from the University of California, Berkeley in 1991. He holds 10 U.S. patents and has published four books, 12 book chapters, and more than 140 archival and 350 conference papers. His research ranges from low power electronics, energy-efficient processing, and cloud computing to photovoltaic cell power generation, energy storage, and power conversion, and from RT level optimization of VLSI circuits to synthesis and physical design of quantum circuits. For this research, he and his students have received six conference and two IEEE Transactions Best Paper Awards. Dr. Pedram is a recipient of the 1996 Presidential Early Career Award for Scientists and Engineers, a Fellow of the IEEE, an ACM Distinguished Scientist, and currently serves as the Editor-in-Chief of the ACM Transactions on Design Automation of Electronic Systems. He has also served on the technical program committee of a number of premiere conferences in his field and was the founding Technical Program Co-chair of the 1996 International Symposium on Low Power Electronics and Design and the Technical Program Chair of the 2002 International Symposium on Physical Design.