Microelectronics Reliability 63 (2016) 267–277

Contents lists available at ScienceDirect

Microelectronics Reliability

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

PVTA-aware approximate custom instruction extension technique: A cross-layer approach

Bahar Farahani [a,*], Saeed Safari [a], Nader Sehatbakhsh [b]

[a] School of Electrical and Computer Engineering, University of Tehran, Iran
[b] Georgia Institute of Technology, Atlanta, GA, United States

ARTICLE INFO

Article history:
Received 17 August 2015
Received in revised form 15 May 2016
Accepted 18 May 2016
Available online 18 June 2016

Keywords:
PVT variations
Transistor aging
Approximate computing
Cross-layer
Fault tolerance

ABSTRACT

Process, Voltage, and Temperature variations together with transistor Aging (PVTA) can result in significant number of timing errors in Custom Instructions (CIs) manufactured at nano-scaled silicon nodes. The state-of-the-art approach to tackle this concern is to use guard-band. However, this policy can adversely decrease the performance gain obtained by CIs as the gap between worst-case delay and true delay due to PVTA variations is increased. This paper proposes a novel approximate CI selection technique to address this issue. This technique allows the applications which do not require perfect accuracy to experience a tolerable amount of timing errors imposed by PVTA variations in favor of significantly improving the performance of the extensible processor. To achieve this, the proposed CI selection technique not only considers those CIs which their PVTA-aware delay is less than the given timing constraint, but also it takes into account the approximate CIs (i.e., those CIs that cannot strictly meet the timing constraint resulting in noisy/approximate computations). First, a timing analysis is performed to precisely compute the delay distribution of CIs in the presence of workload- and circuit-dependent PVTA variations. Then, based on the obtained distribution for each CI, a fault-map (i.e., timing error locations) is extracted. Using the fault-map, each circuit-level timing error is propagated to application-level to evaluate the quality/accuracy of the application output in the presence of PVTA-induced errors in approximate CIs. Finally, based on this cross-layer information, an optimal set of CIs is selected. This set results in maximum performance per silicon area under the given constraints on the power consumption and the errors which can be tolerated by the user. The simulations for various benchmark applications show that the proposed cross-layer technique results in up to 2.7× speedup increase compared to the existing techniques, which comes at the expense of 6% more error.

1. Introduction

The evolution from desktop computing to mobile computing creates relentless demands on designing low power and high performance embedded processors with a very short time-to-market window [1–5]. *General Purpose Processors* (GPPs) are not very desirable for embedded systems, since they are *one-size-fits-all* solutions to provide high average efficiency for a wide spectrum of applications. However, workloads come in a variety of shapes and characteristics, and as a result the ultimate flexibility of GPP leads to significant overheads in power consumption and non-optimal performance speedup for very certain applications. On the other hand, *Application Specific Integrated Circuits* (ASICs) provide both high speedup and low power consumption, but they are usually costly and less flexible in comparison to GPPs. Moreover, ASICs' long time-to-market may not be tolerable by market [4]. The emerging as a tradeoff between GPPs and ASICs us *Application Specific Instruction Set Processors* (ASIPs) [6]. In this methodology, hotspot regions (i.e., frequently used sequences of instructions) of the applications are accelerated by adding specific *Custom Instructions* (CIs) to the instruction sets of GPPs in order to improve the speedup [3,4,7,8].

As outlined by *International Technology Roadmap for Semiconductors (ITRS)* [9], susceptibility to timing errors induced by *process, voltage, temperature, and transistor aging* (PVTA) variations has become one of the major designing challenges for processors [4,10–12]. Indeed, due to PVTA variations, the delay of CIs increase over time and thus, they may result in timing errors when the timing constraint (i.e., clock period) is violated [3,13,14]. The most common approaches for tolerating PVTA variations in order to prevent timing errors are guardbanding and adding redundancies at different layers of design hierarchy [15].

Selecting a CI that fails to meet timing constraint of the processor may increase the speedup [16]. This is at the expense of timing violations of critical paths, resulting in timing errors. The rate of timing errors in a CI is a statistical parameter which depends on PVTA-dependent delay distribution of CIs, the operating frequency of the processor, workload, and input patterns. However, a timing error that occurs in the circuit-level either may be masked by micro-architecture or by application. Even, an altered application output still might be tolerated

* Corresponding author.
E-mail address: bahar.farahani@ut.ac.ir (B. Farahani).

http://dx.doi.org/10.1016/j.microrel.2016.05.008
0026-2714/

by user when the precise value of the output is not a necessity (i.e., *acceptable* instead of precise output). Based on this observation, we propose an *approximate CI selection* technique. Indeed, *Approximate computing* is emerged as a new promising source of efficiency. This approach is driven by the fact that the today's computing demand is almost overwhelmed by a growing number of applications (such as media applications for mobile devices) which are intrinsically tolerant to *noisy/approximate* calculations. Approximate computing can significantly increase the speedup of calculations by adjusting the degree of accuracy needed for the given tasks [17]. The key idea behind our proposed approximate CI selection technique is to let computation precision of the extensible processor be slightly reduced in favor of gaining more speedup. This is achieved by pushing the limit of CIs timing and selecting those CIs that do not strictly meet the clock period.

The rest of this paper is organized as follows: the related work is discussed in Section 2. Section 3 compares the conventional selection techniques of CIs with the proposed technique using a motivational example. Section 4 explains the sources of approximation in this paper. Section 5 discusses the problem statement and overviews the proposed cross-layer CI selection technique. Section 6 explains the cross-layer PVTA profiling flow to extract candidate CIs. Section 7 presents the application error estimation technique in the presence of PVTA-aware approximate CIs. Section 8 discusses the CI selection method and its merit function. Section 10 shows the results. Finally, Section 11 is the conclusion of this paper.

2. Related work

In the field of Instruction Set Architecture (ISA) extension, various techniques have been proposed for CI selection that can be categorized to deterministic CI selection and approximate CI selection groups [5,3,20,21,22,18,23,16,24,25,26,27]. In particular, the impact of process variations on deterministic CI selection is studied in [18]. The authors discussed that to improve the speedup of extensible processors while maintaining the desired timing yield, it is a necessity to obtain the delay of CIs using accurate statistical timing analysis rather than using conventional corner-based timing analysis. The importance of growing problem of transistor aging in deterministic CI selection is discussed in [3,19]. These old-fashioned single-layer CI selection techniques neither take into account any approximation models, nor capture the impacts of runtime voltage and temperature variations. In addition, since their aging model is based on old reaction–diffusion model, they cannot capture the impacts of trapping-detrapping and random telegraph noise in the nano-scale technology nodes.

Approximate CIs can be classified into two groups. The first group is based on imprecise hardware implementation [17], while the second group relies on aggressive clocking [16]. A new class of CIs based on imprecise hardwares is introduced in [17]. The authors proposed replacing the precise arithmetic CI hardware with inaccurate neural networks for better performance and energy efficiency. On the other hand, the authors of [16] proposed to include those CIs that do not satisfy the timing constraint of the processor in order to gain speedup. This technique results in timing errors and in turn imprecise computation. This work suffers from two shortcomings. First, it only considers static process variations, while the impacts of workload-dependent PVTA variations on the generation of timing errors are ignored. Second, the focus of this work is only on the generation of errors in the output of CIs at circuit-level. In this paper, the propagation of errors from circuit-level to application-level is neglected. Our approach falls into the second category (i.e., aggressive clocking). It is also orthogonal to imprecise CI hardware approach. Therefore they can complement each other. Table 1 briefly compares the state-of-the-art CI selection techniques with our proposed approach.

3. A motivational example

Let us compare *Approximate CI selection* with the traditional CI selection algorithms using a motivational example. To do so, we selected two CIs (i.e., CI_1 and CI_2) from mibench benchmark suite [28]. Both CI_1 and CI_2 implement the same computation, but with different hardwares. CI_1 has a low clock saving (low speedup) and low circuit-level delay. CI_2 offers high clock saving (high speedup) and it has a high circuit-level delay. Since CI_2 cannot meet the timing constraint (due to its high delay), in order to guarantee the timing yield of the processor, traditional CI selection algorithms tend to select CI_1 instead of CI_2 [29,3]. Fig. 1 shows the timing error rate of CI_1 and CI_2 in the presence of delay variations. According to the results, although CI_1 has less delay, CI_2 results a graceful degradation in output quality (lower error rate) as delay is increased. On the other hand, CI_1 is significantly sensitive to delay variations and its error rates explode once the hardware experiences small changes in the circuit delay/clocking. Therefore, depending on the requirements, one of the CIs is selected. Assume the delay constraint for CI selection is 0.8 units of time. With this assumption, traditional techniques tend to select CI_1 to guarantee the reliability of the system. On the other hand, in approximate computing that higher errors can be tolerated, CI_2 will be selected. In our example, this decision enables us to perform PVTA-aware over clocking at runtime (i.e., reduce the delay beyond 0.8 units of time) in order to improve the performance.

Note that CIs not only show different behavior in terms of error generation at circuit-level, but also they tend to have different error masking properties (i.e., error propagation from circuit to application). Consider CDFG of Fig. 2 as an example. In this figure, CI_3 provides exactly the same speedup as CI_2, but with higher error rate generation at circuit-level. Regarding error propagation, an error in CI_2 should pass through basic block BB_1, BB_2, and BB_3 to corrupt the output of the application. On the other hand, an error in CI_3 should go through BB_4, BB_5, and BB_6. In order to fairly compare CI_2 and CI_3, we have to compute the corresponding error masking capability of each basic block. According to the figure, BB_2 and BB_3 are entirely identical to BB_5 and BB_6, respectively. Therefore, the only difference between CI_2 and CI_3 in terms of error propagation is rooted in BB_1 and BB_4. Comparing these two basic blocks reveals that BB_4 has much more error masking capability than BB_1. The reason stems from the fact that *logic operations* in BB_4 have more

Table 1
Comparison of the proposed technique with state-of-the-art methods.

Method	Cross-layer	Deterministic CI selection	Approximate CI selection			
				Aggressive clocking		
				Error generation		
			Imprecise hardware	Variations	Input statistics	Error propagation
[18]		P				
[3,19]		PA				
[16]				P		
[17]			ANN			
Proposed	✓			PVTA	✓	✓