Microelectronics Journal 61 (2017) 57–66

Contents lists available at ScienceDirect

Microelectronics Journal

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

Efficient utilization of imprecise computational blocks for hardware implementation of imprecision tolerant applications

Hamid Reza Mahdiani[a,*], Mohammad Haji Seyed Javadi[b], Sied Mehdi Fakhraie[c]

[a] *Computer Science and Engineering Department, Shahid Beheshti University, G.C. Tehran, Iran*
[b] *Faculty of Electrical Enginering, Shahid Beheshti University, G.C. Tehran, Iran*
[c] *School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran*

ARTICLE INFO

Keywords:
Bio-inspired Imprecise Computational (BIC) block
Error-behavior compatibility matching
Imprecision Tolerant (IT) application
Multiply-accumulate
VLSI design flow

ABSTRACT

Recently, it has been reported that exploiting imprecise arithmetic building blocks such as adders and multipliers significantly improves digital implementation costs as well as performance of an important category of systems named as Imprecision Tolerant (IT) applications. We have categorized this new type of functional units as Bio-inspired Imprecise Computational (BIC) blocks. To efficiently exploit BICs in an IT application, however, the traditional hardware design flow should also be customized based on unique features of this new type of computing blocks. The most significant modification on traditional design flow to maximize the cost-performance of BIC implementations is to verify that the application is capable of tolerating those types of errors which are inherently introduced by the selected BIC based on its internal structure. We call this “error-behavior compatibility matching” between the system and the selected BIC building blocks. In this paper, we introduce and explain a customized hardware design flow for BICs with the main focus on error-behavior compatibility matching process as the main difference between traditional and BIC design flows. Two different error-behavior compatibility matching strategies are also introduced and their applicability is verified by applying them to exploit BICs for hardware implementation of some significant case studies including a general multiply-accumulate (MAC) block as the basic building block of many signal processing applications as well as an Artificial Neural Network (ANN) as a critical instance of MAC-based IT applications.

1. Introduction

Many real-world problems (such as classification, recognition, image and voice processing, etc.) are pervasively imprecise and uncertain [1]. Therefore, most of the existing algorithms and methodologies which are mainly developed to deal with these real-world problems are expected to inherently be capable of tolerating or even exploiting this existing unavoidable imprecision and uncertainty [2,3]. Otherwise, they will work inefficiently at elevated costs [4]. We categorized this type of applications and named them as Imprecision Tolerant (IT) applications and also clarified their distinct border with respect to other syntactically similar concepts such as error or fault tolerance [5]. An application is Imprecision-Tolerant if it tolerates some degrees of imprecision when realized using an implementation bed. In other words, the implemented system produces acceptable results to the end user despite existing sources of imprecision [5]. The main importance of IT applications lies in this fact that their hidden property of tolerance against imprecision can be explicitly traded to gain considerable and various benefits in their realization process according to their implementation bed [5]. As an instance in digital VLSI territory, a designer might utilize some irregular adders and multipliers that are intentionally relaxed to provide an approximation of the computation result instead of its precise value. This relaxation also might imply decreased implementation costs of these Bio-inspired Imprecise Computational blocks (BICs) in terms of area, delay, and power consumption with respect to the traditional precise components [6]. Although some other names such as approximate [7] or sloppy [8] are later proposed for this type of blocks, the term imprecise is carefully chosen to show tight correspondence of the BICs with Imprecision-Tolerant applications. Besides, the term imprecision covers much broader area with respect to approximation or sloppy. As a very significant instance, imprecision might be introduced due to unwanted soft-errors which frequently occur in a nano-level CMOS circuit. We have shown that in this case, the imprecision tolerance of an application might be exploited to reduce system fault-tolerance cost when realizing it in a faulty implementation bed [5].

The important note about IT applications is that different IT applications provide intrinsic susceptibility against different impreci-

* Corresponding author.
E-mail addresses: mahdiani@sbu.ac.ir, mahdiani@gmail.com (H.R. Mahdiani).

http://dx.doi.org/10.1016/j.mejo.2017.01.002
Received 20 April 2016; Received in revised form 24 November 2016; Accepted 4 January 2017

sion types. For example, although a binary ANN is not sensitive against number or even amplitude of the imprecision sources, it is highly sensitive about average of the imprecision sources due to its accumulation-based structure. On the other hand, an image or voice processing system does not directly care about mean imprecision value. While in contrast, it is highly susceptible against both number and maximum absolute value of the imprecision source(s) that distinguishably alter the values of pixels or voice samples, so that the changes can be seen or heard by human eyes and ears. A very similar concern exists about the BICs and they also can introduce different imprecision types. In contrast to the traditional precise computational blocks of the same type (i.e. adder or multiplier) which have a similar functionality (based on binary addition and multiplication rules) and differ only in their physical properties [9], the BICs of the same type provide different physical properties as well as different error behaviors which describes their result-approximation characteristics [6] according to their internal structure. Based on this important similar issue in both IT applications and BICs, there is a very important conceptual difference when using BICs to implement an IT application instead of precise components. In contrast to all precise blocks of the same type which can be functionally exploited in place of each other to implement an application, different BICs of the same type which propose different error-behaviors does not provide similar effectiveness if they are utilized to implement a specific IT application. In a more technical way and in contrast to precise blocks, choosing a suitable BIC is an application-oriented task and significantly affects the performance and physical properties of the final implementation. Therefore, it is mandatory to investigate the error behavior of BICs to check their “error behavior compatibility” with respect to necessities of a specific IT application. A BIC error behavior should be described in terms of a spanning set of error criteria to correctly demonstrate its error-behavior from different aspects. There are many different well-known and standard error criteria which can be simultaneously exploited to express a BIC error behavior [10]: Probability of Error (PE) which indicates the probability of producing an erroneous result by BIC, Mean output Error (ME), Mean Squared output Error (MSE), or maximum (MAX) and minimum (MIN) output errors. The error factors of a BIC are computed with respect to the output results of a corresponding precise component. As an example, based on error sensitivity of an ANN, an imprecise adder which mostly provides erroneous results (high PE) while its average error is very low due to its small positive and negative errors (low MAX and ME) [6], is more error behavior compatible with an ANN with respect to another imprecise adder which mostly leads to the precise result (low PE) while also provides rare but large and positive errors [11] (high MAX and ME). Both of these two adder types will be discussed in more detail in the following sections.

Although the experimental results show that using a BIC for implementing an application can be generally feasible [6,12–15], it is very important to determine a customized VLSI design flow for BICs to efficiently exploit the tolerance of the IT applications and gain the best performance in case of BIC implementation. The next section introduces the problem and deals with the main drawbacks of the traditional design flow when exploiting BICs for system implementation. Section 3 reviews the most important recently published BIC related works and investigates them from design flow aspects to demonstrate their common pitfalls due to lack of an convenient design flow for BICs. Section 4 proposes an efficient design flow for BICs while it also provides a detailed comparative study between (traditional) precise and (introduced) BIC design flows. Section 5 specifically focuses on a major difference between traditional and BIC VLSI design flows named as error behavior compatibility matching process. Two different error-behavior compatibility matching approaches are introduced in this section. These two methods can be used either individually or in a combined manner to determine whether a BIC (with a specific error-behavior) is suitable for handling an application or not. Application of these two methods for a MAC (as the basic underlying component of many DSP systems) is also demonstrated in this section. Section 6 presents experimental results of a pure MAC as well as an ANN (as a critical and practical MAC based application) to support the results of the previous section and prove the efficiency of the proposed methods. The last section concludes the paper.

2. Problem description

To gain the best performance when implementing an IT application using BICs, the design flow should be customized to both quantify and qualify a BIC component in terms of amount and type of its errors respectively to guarantee its efficiency (not only its feasibility) for implementation of a specific IT application. As each BIC non-uniformly introduces some types of errors based on its internal architecture (or result approximation method), the quantification process deals with tuning the total amount of errors inside a BIC (and letting it to increase) up to maximum acceptable error which can be tolerated by each application. This implies that in addition to Word-Length (WL) which uniformly affects all types of errors in a block, the BIC components should be equipped with one or even more extra parameters named as imprecision parameters. These parameters enable the designer to fine-tune the levels of the BIC specific imprecision types in order to maximally utilize the inherent tolerance of the specific IT application against that type of imprecision and gain the maximum benefit [16].

The qualification process on the other hand is a higher level and more important concept which investigates whether a BIC is basically compatible with an application or not, before any further effort for fine-tuning of its imprecision parameters. For normal implementation of a system, the designer picks some suitable components based only on their physical properties. For a BIC implementation, however, the error-behavior of the BIC components is the dominant factor in choosing suitable components. Because as an early stage in BIC design process, it is very important to basically determine whether the specific types of errors (PE, ME, MSE, MAX, etc.) which are introduced by a BIC due to its internal structure, suitably match with the types of errors which can be tolerated by the specific IT application or not. We name this qualification process as error-behavior compatibility matching which determines if the specific application inherently tolerates the same type of imprecision which are introduced by a specific BIC or not. The results of the error behavior compatibility matching process might also be utilized to guess the promising applications of a previously designed BIC, or to determine the suitable error-behavior of a BIC that should be specifically designed for a certain application. The error behavior compatibility matching concept, clearly emphasizes on application-oriented nature of the suitable BIC selection.

A customized design flow for BICs including quantification (fine tuning) and qualification (error behavior compatibility matching) steps is presented in the following sections of the paper to resolve the addressed inefficiencies.

3. Related works

Working on imprecise building blocks and their applicability in realization of IT applications has been seriously started around 2010 [6]. Beside some minor weaknesses, lack of exploiting even an elementary but clear and convenient design flow should be considered as the major drawback of all previously published works in this area. As we will briefly review, the researchers in this field mostly have focused only on one or more introductory sub-domains including: (1) exploiting different techniques for development of new BICs (such as adders and multipliers), (2) using analytical or simulation approaches for comparing the proposed BIC with other existing BICs of the same type, and finally (3) proving only the feasibility (and not even the efficiency) of utilizing the proposed block in at most one IT application. Due to lack of a customized and well-defined design flow for BICs, all these works suffer from important weaknesses even in their activities in each of