Design Space Exploration for Approximate Implementations of Arithmetic Data Path Primitives

Lukas Sekanina, Vojtech Mrazek and Zdenek Vasicek
Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: {sekanina, imrazek, vasicek}@fit.vutbr.cz

***Abstract*—While a detailed analysis of various approximation strategies for elementary arithmetic circuits (such as adders and multipliers) is widely covered in the literature, much less is known about approximate arithmetic data path primitives (such as Dot Product (DP) to name one example) because it is difficult to exactly analyze their error and other parameters. We provide a detailed analysis of approximation options if a two-dimensional DP circuit is implemented using pre-characterized approximate arithmetic circuits available in existing libraries of approximate circuits. We propose a method capable to cheaply estimate properties of candidate approximate implementations of DP circuit. By a careful selection of approximate components, hundreds of approximate DP circuits showing excellent tradeoffs between key design objectives were obtained.**

***Index Terms*—approximate computing, automated design, digital circuit, dot product, multiplier**

I. Introduction

The current renewed interest in approximate implementations of circuits and systems is primarily caused by the urgent need to reduce power consumption of computer systems ranging from low power nodes of Internet of Things (IoT), via mobile devices to supercomputers.

This paper deals with *functional approximation* of arithmetic circuits in which the accurate (original) circuit implementation is simplified in order to optimize power consumption and other parameters. This topic has already been intensively studied in different contexts: (i) For elementary circuits, such as adders and multipliers, various ad hoc approximation strategies have been proposed and compared; see a detailed survey and classification in [1]. (ii) More complex arithmetic circuits—in this paper, collectively referred to as the *arithmetic data path primitives*—such as Dot Product (DP), Discrete Cosine Block (DCT) and Multiply-and-Accumulate (MAC) structures were approximated by replacing their accurate components (adders and multipliers) with pre-designed approximate implementations, while keeping the structure of the original circuit unchanged [2]. (iii) Both the elementary circuits and the data path primitives were approximated using general-purpose automated approximation methods, typically performing approximate gate-level resynthesis and optimization [3], [4].

All these approaches have to evaluate the approximation error of the resulting circuits. As we deal with arithmetic circuits, it is natural to analyze their error by means of arithmetic error metrics such as the *error rate*, the *mean absolute error* (MAE) and the *worst case error* (WCE). The approximation error can exactly be calculated (if responses for all possible inputs are checked or formal verification methods are applied [3]) or estimated (by circuit simulation or statistical analysis [2], [5], [6]). Obviously, the error estimation methods are applicable to more complex circuits than the exact error calculation methods. On the other hand, the error estimation does not provide strong guarantees in terms of error, especially when one has to deal with corner cases which are hard to discover. It turns out that the tradeoffs that can be obtained between the error and power consumption (or other relevant circuit parameters) are better understood for elementary circuits (because of many case studies were conducted and formal guarantees were given) than for more complex arithmetic circuits (only a few studies were conducted and circuit parameters were estimated).

This paper focuses on a detailed analysis of approximation options for combinational circuits that are composed of several elementary arithmetic circuits (components)—*data path primitives*. In particular, we will thoroughly investigate and compare approximate implementations of a DP circuit computing $y = a * b + c * d$, where all operands are 8 bit unsigned numbers and the conventional (exact) implementation would consists of two 8 bit multipliers and a single 16 bit adder. DP has a number of applications, for example, it is a building block of parallel implementations of the convolution operation in configurable image filters or in the convolutional layers of convolutional neural networks (CNN). As DP circuit has 32 inputs and contains two multiplication blocks, it would be a hard target for automated circuit approximation methods that provide formal guarantees in terms of the approximation error. On the other hand, it is still possible to analyze the DP response for all 2^{32} input combinations in a reasonable time (see Section IV). It means that thousands of approximate implementations of DP can be characterized in terms of the (exact) error and other circuit parameters (that can be assessed by means of standard circuit design tools) using a common personal computer.

In order to create these approximate implementations of DP, we will employ approximate components that are available in the EvoApprox8b library [4]. Every component of EvoApprox8b is fully characterized in terms of the error (seven error metrics provided), area, delay, power consumption and other parameters for selected fabrication processes.

978-1-5386-9562-3/18/$31.00 ©2018 IEEE

The goal of the paper is to identify approximate implementations of DP showing the best tradeoffs between the error and other parameters. Furthermore, the proposed methodology should enable us to analyze to what extent it is reliable to estimate the error (resp. power consumption) of approximate DPs solely on the basis of the parameters of its components.

II. Approximate Circuits

Two major classes of circuit approximation methods are distinguished in the literature: *timing induced approximation* and *functional approximation*. As it is difficult to systematically adopt voltage over-scaling and predict its impact on timing and approximation errors, most approaches currently use the principles of functional approximation.

Various *ad hoc* approaches to functional approximation have been developed for specific classes of circuits. For example, the design of approximate adders and multipliers based on various simplifications, precision scaling and pruning of circuit structures is quite well covered in the literature (see a comprehensive survey in [1]). Our study showed that out of these approaches a simple truncation and the so-called broken-array multiplier (BAM) [7] provide the best tradeoffs between WCE (MAE) and the power delay product (PDP) [3].

Because the manual ad hoc approach typically requires a lot of human effort, fully automated functional approximation methods, applicable to broad classes of circuits or even arbitrary circuits, have been proposed. Starting with a fully functional circuit, an error metrics and a list of constraints, the objective of the automated functional approximation is to provide circuit implementations showing good trade-offs between key design objectives in an automated fashion. These methods typically employ truncation, pruning, component replacement, and approximate logic re-synthesis [2]–[4], [8].

The error calculation is often based on *circuit simulation* using a randomly generated subset of all input vectors. However, this approach does not provide any guarantees in terms of the result. The exact error can be obtained by *formal methods* based on, e.g., Binary Decision Diagram analysis and satisfiability (SAT) solvers [3], [8]. While the (exact) WCE calculation scales up to 32-bit multipliers, it is an open problem how to approximate circuits such as 12-bit multipliers under the MAE metric [3]. It has to be noted that multipliers are hard problem instances from the verification point of view and, hence, they usually serve as good benchmarks.

Other error modeling approaches are based on *probabilistic and statistical analysis*. They assume that (i) the circuit under analysis (e.g. an approximate adder or a multiplier) has a specific structure, (ii) it is possible to reliably model the error of its sub-circuits and (iii) combine these error models using statistical methods. For example, Mazahir et al. introduced the error probability analysis for recursive approximate multipliers with approximate partial products [6] and generic configurable adders [5]. However, this approach cannot be applied if the approximate implementation is obtained by an automated gate-level approximation method because there are no common sub-circuits preserved which an error model can be built for. Furthermore, approximate circuits containing many regular sub-structures are known to be far from the optimum in terms of the error-power tradeoffs [3].

In some cases, this type of statistical error modelling can be generalized for complex arithmetic approximate structures. For example, a variance error model capable of capturing the error propagation along the circuit structure, considering also structural correlations, was introduced in the context of high level synthesis [2].

III. Efficient Design Space Exploration

Conducting the design space exploration for approximate circuits is a highly non-trivial problem because, in addition to traditional metrics (power, delay and area), the error metric introduces a new search dimension. As quantifying the error is usually computationally expensive, it determines how much and in which quality the search can be performed. We deal with approximate arithmetic circuits composed of several elementary approximate components. It is assumed that these approximate components are fully characterized in terms of electrical parameters, their error is pre-calculated with respect to several error metrics and their implementations are available in a library. In our case, we will use EvoApprox8b library containing hundreds of approximate 8 bit multipliers and adders (in the fixed point number representation) that were automatically created by means of genetic programming in our previous work [4]. It means that nothing can be supposed about their internal structure and hence the methodology developed in [5], [6] is not applicable to model the error distribution. The objective is to find the combination(s) of approximate components showing the best tradeoffs among design objectives at the level of the whole circuit.

A. Approximate Dot Product

Detailed design space exploration will be performed for the approximate DP circuit, whose exact implementation consists of two 8 bit multipliers and a 16 bit adder. In order to create reference (exact) implementations of DP, we employed common accurate multipliers and an adder synthesized using the Verilog + operator. All results will be reported for 45 nm technology (PDK) with $V_{cc} = 1$ V. As only 17% of power consumption is due to the adder, we will keep it exact and focus on the approximation process of the multipliers. Three types of 8-bit multipliers will be considered: (i) exact multipliers, (ii) approximate BAM and truncate multipliers taken from the literature (these multipliers belong to the best-performing approximate multipliers [1], [3]) and (iii) approximate multipliers from EvoApprox8b library.

Let us suppose that k different 8-bit multipliers are available. Under this setup, an optimal solution is obtained if the whole search space containing $k(k+1)/2$ design points is analyzed. We will synthesize all DP circuits to obtain their electrical parameters. The (exact) approximation error will be determined by the bit-wise parallel simulation (Section III-B). However, the question is if we can skip this very time consuming process and obtain the optimum combination(s)

of approximate multipliers in another way, for example, by exploiting pre-computed parameters of the circuits available in EvoApprox8b library.

B. Error Analysis

The (exact) error of a combinational gate-level circuit can be obtained by bitwise parallel simulation conducted for all input combinations. The idea of this kind of simulation is to utilize bitwise operators operating on multiple bits in a high-level language (such as C) to perform more than one evaluation of a gate in a single step. The widely available Advanced Vector Extension (AVX) instruction set allows us to operate with 256-bit operands. It means that every circuit with eight inputs can completely be simulated in one pass by applying a single 256-bit test vector at each input. When a more complex circuit has to be evaluated, multiple 256-bit vectors are applied sequentially. The obtained speedup is w on a w-bit processor (assuming $2^{inputs} \geq w$). Although this technique does not scale well, it is feasible to exactly evaluate the chosen 32-input DP circuit.

In general, we try to avoid this expensive approach because it makes the design space exploration inefficient. One of the possibilities is to replace the exact error by a suitable error estimation. We can estimate the error of DP in a constant time using the known errors of the multipliers (precomputed in the library); the adder can be omitted from the analysis because its error is 0.

The worst-case error of an approximate multiplier $\widetilde{\mathrm{M}}(a,b)$ can be calculated as

$$\mathrm{WCE}_{a\widetilde{*}b} = \max_{\forall a,b} |a\widetilde{*}b - a*b|, \tag{1}$$

where $\widetilde{\mathrm{M}}(a,b) = a\widetilde{*}b$ denotes the approximate multiplication, $a,b \in \{0, 2^w - 1\}$, and w denotes the bit-width of the multiplier. This equation can be rewritten to the following expression

$$\mathrm{WCE}_{a\widetilde{*}b} = \max\{\mathrm{WCE}^{+}_{a\widetilde{*}b}, -\mathrm{WCE}^{-}_{a\widetilde{*}b}\}, \tag{2}$$

where $\mathrm{WCE}^{+}_{a\widetilde{*}b}$ denotes the maximum error magnitude and $\mathrm{WCE}^{-}_{a\widetilde{*}b}$ the minimum error magnitude defined as follows:

$$\begin{aligned} \mathrm{WCE}^{+}_{a\widetilde{*}b} &= \max_{\forall a,b}(a\widetilde{*}b - a*b) \\ \mathrm{WCE}^{-}_{a\widetilde{*}b} &= \min_{\forall a,b}(a\widetilde{*}b - a*b) \end{aligned} \tag{3}$$

The *worst-case error* of a DP circuit consisting of two approximate multipliers $\widetilde{\mathrm{M}}_1(a,b)$ and $\widetilde{\mathrm{M}}_2(c,d)$ and one accurate adder is defined as

$$\mathrm{WCE}_{DP} = \max_{\forall a,b,c,d} |(a\widetilde{*}b + c\widetilde{*}d) - (a*b + c*d)|. \tag{4}$$

We can rearrange Eq. 4 and express the worst-case error using the notation introduced in Eq. 3 as

$$\begin{aligned} \mathrm{WCE}_{DP} &= \max_{\forall a,b,c,d} |(a\widetilde{*}b - a*b) + (c\widetilde{*}d - c*d)| \\ &= \max\{\mathrm{WCE}^{+}_{a\widetilde{*}b} + \mathrm{WCE}^{+}_{c\widetilde{*}d}, -\mathrm{WCE}^{-}_{a\widetilde{*}b} - \mathrm{WCE}^{-}_{c\widetilde{*}d}\} \end{aligned} \tag{5}$$

In order to obtain the exact value of WCE for DP, it is thus necessary to know WCE^{+} and WCE^{-} errors of the approximate multipliers. However, they are easy to calculate in the 8-bit case.

The *mean error* (ME) of DP is defined as

$$\mathrm{ME}_{DP} = \frac{1}{2^{4w}} \sum_{\forall a,b,c,d} [(a\widetilde{*}b + c\widetilde{*}d) - (a*b + c*d)] \tag{6}$$

and it can also be exactly determined because $\widetilde{\mathrm{M}}(a,b)$ and $\widetilde{\mathrm{M}}(c,d)$ are independent random variables produced by multipliers M_1 and M_2. In this case, we obtain the expression

$$\mathrm{ME}_{DP} = \mathrm{ME}_{M_1} + \mathrm{ME}_{M_2}. \tag{7}$$

For the *mean absolute error* of DP, however, the situation gets complicated. In this case, we will simply estimate the error as

$$\mathrm{MAE}_{DP} \approx \mathrm{MAE}_{M_1} + \mathrm{MAE}_{M_2} \tag{8}$$

because nothing is, in general, known about the distribution of MAE_{M_1} and MAE_{M_2}.

C. Electrical Parameters

The electrical parameters of approximate DPs can be obtained in a standard way, e.g. with Synopsys Design Compiler. However, as this is a very time consuming procedure in our case, it is preferred to call this tool only for the most promising implementations. Hence, the electrical parameters of DPs will be estimated using electrical parameters of the components available in the library. The most straightforward estimate of the area (resp. power consumption) lies in summing the areas (resp. power consumptions) of all components involved in DP. Delay d_{DP} of an approximate DP is estimated as the delay along the longest path

$$d_{DP} = \max\{d_{\widetilde{\mathrm{M}}(a,b)}, d_{\widetilde{\mathrm{M}}(c,d)}\} + \mu.d_{adder}, \tag{9}$$

where $d_{(.)}$ denotes component's delay and μ ($\mu \in < 0, 1 >$) is a coefficient reflecting the adder's contribution to the delay.

IV. Results

The following characterization of the approximate DP circuit is based $k = 174$ multipliers: 128 approximate multipliers from the EvoApprox8b library (only non-dominated multipliers were selected considering power and MAE as the objectives), 36 approximate BAM multipliers, 7 truncated multipliers and 3 exact multipliers. The design space then contains 15 225 DP circuits whose parameters were estimated using methods presented in Section III-B and III-C.

Then, we filtered out all dominated solutions with respect to estimated values of WCE, MAE, area, delay and power consumption. The remaining 1082 DP circuits were synthesized with Synopsys Design Compiler to obtain their 'exact' electrical parameters and exhaustively simulated to get the 'exact' approximation error. Fig. 1 shows how PDP (the area, power consumption and other parameters are omitted because of limited space) depends on MAE and WCE for all 1 082 DP circuits (only the 'exact' parameters were used to construct

Fig. 1). An important observation is that it is crucial to employ approximate multipliers from EvoApprox8b because their usage has led to much better tradeoffs than the traditional well structurally-understood approximate multipliers (such as BAM and truncated multipliers) can provide. The design points forming the Pareto front provide a wide spectrum of different approximate implementations of DP from which the designer can pick the right solution satisfying his/her requirements.

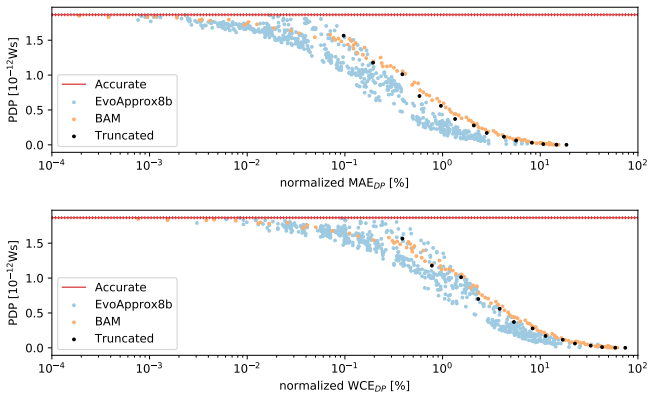

Fig. 1. Power delay product vs. MAE and WCE for approximate DP circuits. All parameters were obtained by 'exact' methods.

In order to assess the quality of our parameter estimation methods, we compared the 'exact' parameters with their estimated values for all 15 225 DP circuits (their synthesis took 13 hours and the 'exact' error analysis took 380 hours when re-calculated to a single core processor time).

In the case of MAE estimation, Fig. 2 (top-left) shows that our simple estimation method is too pessimistic, i.e. approximate DP circuits usually exhibit lower MAE than we estimated. However, Pearson correlation coefficient (PCC = 0.994) still indicates a quite good estimate. It has to be noted that one blue point represents one DP circuit and the red line represents the optimal solution.

While WCE of approximate DPs was estimated almost perfectly (Fig. 2, top-right), the proposed power estimation method shows some imperfections in the corner cases (Fig. 2, left-down). We recognized that this phenomenon is caused by our incorrect assumption that the adder is in the estimation methods always considered as an exactly computing circuit with two 16 bit operands (it is implemented with the $+$ operator in our Verilog models). In reality, when more aggressively approximated multipliers are created, some of their outputs are becoming unused, the bit width of the adder is adequately reduced by the Synopsis Design Compiler and power consumption of DP is lowered. This simplification of the adder, which is hard to predict, is not reflected in our estimate (PCC = 0.989).

Delay estimation conducted for $\mu = 0.5$ shows severe issues as seen in Fig. 2, right-down. This is again caused by pruning the 16 bit adder and our very simplified estimation of this process (PCC = 0.764).

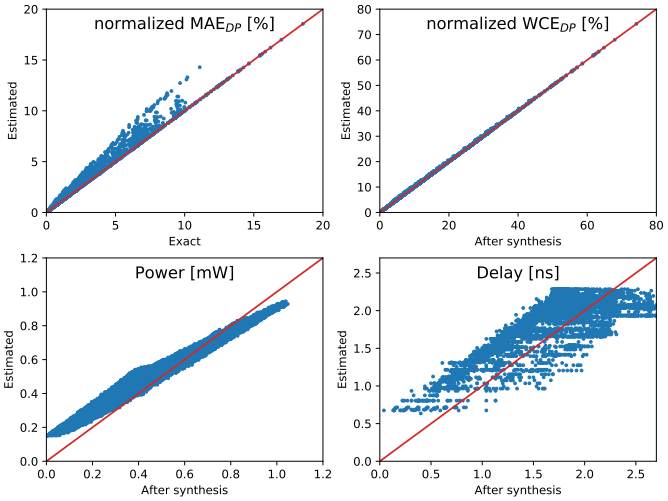

Fig. 2. Relations between estimated and exactly obtained parameters for all 15225 DP circuits

V. Conclusions

We provided a detailed analysis of approximation options if a two-dimensional DP circuit is implemented using one common adder and two approximate multipliers whose implementations are taken from a library of approximate circuits. By means of the proposed estimation methods, we reduced the number of design alternatives (that have to be exactly evaluated) by 92% with respect to all possible designs.

Our future work will be devoted to improving our error estimation methods and applying the proposed methodology to the design of other approximate data path primitives. We will apply advanced search methods such as evolutionary algorithms to even reduce the number of candidate designs.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports, within the INTER-COST project LTC18053.

References

[1] H. Jiang, C. Liu *et al.*, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, Aug. 2017.

[2] C. Li, W. Luo *et al.*, "Joint precision optimization and high level synthesis for approximate computing," in *Proc. of DAC'15*. ACM, 2015, pp. 1–6.

[3] M. Ceska, J. Matyas, V. Mrazek, L. Sekanina, Z. Vasicek, and T. Vojnar, "Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished," in *Proc. of 36th IEEE/ACM Int. Conf. On Computer Aided Design*. IEEE, 2017, pp. 416–423.

[4] V. Mrazek, R. Hrbacek *et al.*, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Proc. of DATE'17*, 2017, pp. 258–261.

[5] S. Mazahir, O. Hasan, R. Hafiz, and M. Shafique, "Probabilistic error analysis of approximate recursive multipliers," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1982–1990, 2017.

[6] ——, "Probabilistic error analysis of approximate recursive multipliers," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1982–1990, 2017.

[7] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, April 2010.

[8] S. Venkataramani, A. Sabne *et al.*, "SALSA: systematic logic synthesis of approximate circuits," in *Proc. of DAC'12*. ACM, 2012, pp. 796–801.