An Efficient Fault Injection Algorithm for Identifying Unimportant FFs in Approximate Computing Circuits

Jiaxuan Lu
Nagoya University
jiaxn_lu@ertl.jp

Yutaka Masuda
Nagoya University
masuda@ertl.jp

Tohru Ishihara
Nagoya University
ishihara@ertl.jp

***Abstract*—Approximate Computing (AC) saves energy and improves performance by introducing approximation into computation in error-torrent applications. This work focuses on an AC strategy that accurately performs important computations and approximates others. In order to determine which calculations are unimportant, we propose a novel importance evaluation algorithm, in which the key idea is a two-step fault injection to extract the near-optimal set of unimportant flip-flops in the circuit. The proposed algorithm reduces the complexity of architecture exploration from an exponential order to a linear order without understanding the functionality and behavior of the target application program.**

***Index Terms*—approximate computing, importance evaluation, fault injection**

I. Introduction

Over the last decade, approximate computing (AC) [1]–[4] has attracted much attention as a post-Moore computing paradigm, enabling further power saving, area reduction, and performance enhancement of integrated circuits. AC relaxes the conventional policy that all computations are performed accurately and introduces approximation into the computation. This work focuses on an AC strategy that accurately performs important computations and approximates others. To make AC circuits practical, we need to determine which computation is how important carefully and thus need to appropriately approximate the unimportant computation for maintaining the required quality.

Here, as one of the most popular techniques in the dependable computing field, fault injection (FI) has been widely studied [5]–[9]. FI refers to the operation of introducing fault information into a circuit. The algorithm, computational mechanism, and circuit component vulnerable to faults can be evaluated by observing whether the fault information induces abnormal behaviors. Originating from its simplicity and flexibility, FI can be widely deployed for importance evaluation. For example, Constantin et al. proposed a method using dynamic timing analysis and instruction set level FI to assess the impact of delay faults on AC circuits [10].

Motivated by the achievements of FI for importance evaluation, this paper tackles to evaluate the importance of computation in AC circuits using FI. This work focuses on the importance at the flip-flop (FF) level, which is defined as "how much the computational quality degrades when the approximation is applied for the FF." One naive approach is to perform FI for all combinations of FFs, estimate the importance of each combination, and derive unimportant FFs. However, this is not feasible since the number of combinations explodes exponentially for the number of FFs.

In this paper, we propose a novel importance evaluation algorithm in which the key idea is a two-step FI. In the first step, we perform the FI simulation for each FF and extract the candidates of unimportant FFs, which reduces the complexity of architecture exploration from an exponential order to a linear order. Considering the case that several FFs are simultaneously approximated, the second step in the proposed algorithm further explores unimportant FFs in a binary search manner. The main contributions of this work include (1) the importance evaluation methodology and (2) quantitative evaluation of area reduction and power saving effects thanks to the design optimization for extracted unimportant FFs.

II. Proposed Algorithm

Figure 1 shows an overview of the proposed algorithm. We formulate the problem of extracting unimportant FFs as follows: The inputs for this problem are one pre-AC circuit having N_{FF} FFs and the constraint of computational quality ($\mathrm{Quality}^{\mathrm{min}}$). The output is a list of unimportant $\mathrm{N}_{\mathrm{unimp}}$ FFs. The objective of this problem is to maximize $\mathrm{N}_{\mathrm{unimp}}$ in order to amplify the benefit of AC. Note that this work assumes the FI simulation that the value of selected FFs is always fixed with user-defined constant values.

The proposed algorithm derives unimportant FFs with the two-step FI. In the first step, candidates of unimportant FFs are extracted by performing the FI simulation on each FF. After each FI simulation, the computational quality is derived and compared with the constraint of $\mathrm{Quality}^{\mathrm{min}}$. If the quality satisfies the constraint, we regard the FF as an unimportant candidate. Thus, the first step extracts $\mathrm{N}_{\mathrm{unimp}}^{\mathrm{cand}}$ candidates of unimportant FFs with N_{FF} times FI simulation.

In the second step, we extracted $\mathrm{N}_{\mathrm{unimp}}$ FFs from $\mathrm{N}_{\mathrm{unimp}}^{\mathrm{cand}}$ FFs in a binary search manner. Specifically, each FI simulation in the second step performs the FI to $\mathrm{N}_{\mathrm{unimp}}$ FFs, not one FF in the first step. Note that $\mathrm{N}_{\mathrm{unimp}}$ is swept in binary search manner between 0 and $\mathrm{N}_{\mathrm{unimp}}^{\mathrm{cand}}$. Therefore, in the second step, the required number of FI simulations is $(\lfloor log_2(\mathrm{N}_{\mathrm{unimp}}^{\mathrm{cand}}) \rfloor + 1)$ times at most, whose overhead for the first step is quite small. In summary, the proposed algorithm reduces the complexity of

978-3-9819263-7-8/DATE23/© 2023 EDAA

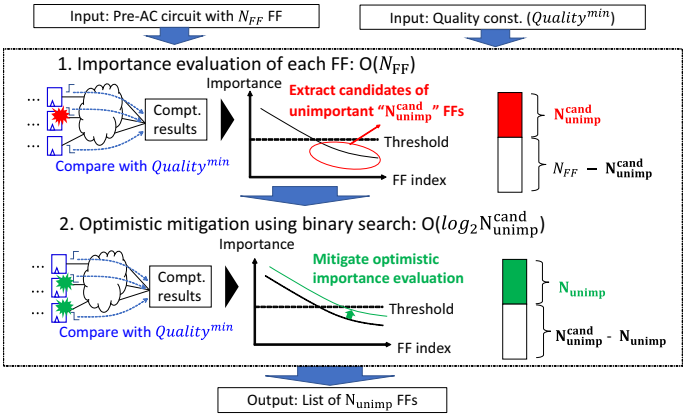

Fig. 1. Overview of the proposed algorithm with the two-step FI.

architecture exploration without understanding the functionality and behavior of the target application program.

III. Experimental Evaluation

This section evaluates the importance of FFs in a case study of an image processing accelerator and discusses the area reduction and power saving effects thanks to the approximation for extracted unimportant FFs.

As the target circuit, we used the Sobel Filter in S2CBench [11] and designed a 16-way Sobel accelerator. We selected 9 images from SIDBA [12] as the workload, chose the Peak Signal-to-Noise Ratio (PSNR) as the computational quality, and prepared 3 PSNR constraints of 20 dB, 30 dB, and 50 dB. The PSNR of the output image can be calculated by referring to the golden outputs from the circuit. The lower PSNR indicates that inserted errors highly degrade the output image quality, i.e., the inserted FF has higher importance.

Firstly, the pre-AC circuit and 3 images are given to the proposed algorithm. We used Verilator as the RTL FI simulator and utilized FI operation that always fixes the input to target FF to "0". Then, in each of the 16 Sobel instances, the first step identifies 11 FFs and 19 FFs as candidates of unimportant FFs for the cases with constraints of 30 dB and 20 dB, respectively. With the most strict constraint, i.e., 50 dB PSNR, the algorithm concludes that no unimportant FFs exist. This is an interesting observation since it can suggest the designer to give up the approximation if necessary. Through the second step, 7 FFs and 13 FFs in each Sobel instance are finally extracted as unimportant FFs for the cases with constraints of 30 dB and 20 dB, respectively. For extracted FFs, we applied the simple AC technique that truncates the input to unimportant FFs to "0".

Next, we perform the quality validation with different 6 input images to discuss the importance of the second step of the proposed algorithm. Table I shows the summary of PSNR comparison results. We can see that the comparative approach violates the PSNR constraint, whereas the proposed algorithm satisfies it. From these results, we experimentally confirmed that the second step contributes to mitigating the optimistic property in the importance evaluation.

Finally, the power dissipation and circuit area are compared to the baseline circuit under the ASIC-based and FPGA-based

TABLE I
PSNR comparison results. The comparative approach truncates FFs extracted with the first step of proposed algorithm.

image name	PSNR 30 dB const.		PSNR 20 dB const.	
	PSNR[dB] (Proposed)	PSNR[dB] (Comparative)	PSNR[dB] (Proposed)	PSNR[dB] (Comparative)
Aerial	32.2	25.2	22.7	12.1
Boat	31.4	24.1	21.8	10.6
Clock	31.1	23.4	21.4	11.2
House1	31.2	24.1	22.2	9.0
Jelly beans	30.0	22.8	20.8	10.5
Text	30.7	23.9	22.1	11.3

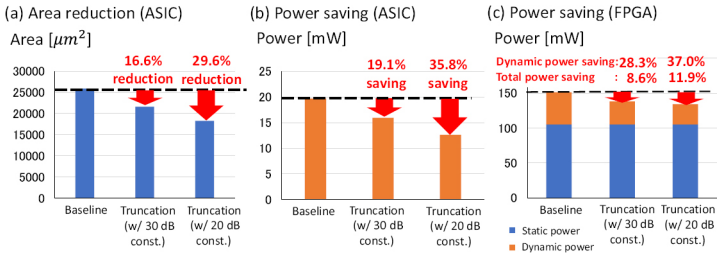

Fig. 2. (a) Area reduction and (b) power saving effects under the ASIC implementation and (c) power saving effects under the FPGA implementation thanks to the AC for unimportant FFs extracted with the proposed algorithm.

implementation using logic synthesis tools (Synopsys Design Compiler and Xilinx Vivado). Figure 2 shows the comparison results. From Fig. 2, we can see the significant area reduction and power saving effects both under the constraint of 20 dB and 30 dB. For example, from Fig. 2(a)(b), the circuit area and power dissipation are saved by 29.6% and 35.8% under the ASIC-based implementation with the PSNR constraint of 20 dB. Similarly, from Fig. 2(c), we can see that the dynamic power and total power dissipation under the FPGA-based implementation are saved by 37.0% and 11.9%. Therefore, we experimentally confirmed that the proposed algorithm contributes to enhancing the area and power efficiency while satisfying the constraint. When we target FPGA to implement the circuits found by our algorithm, the proposed algorithm can be more general, contributing to reducing the power dissipation of the different applications run on FPGA.

Acknowledgment

This work was partially supported by JSPS KAKENHI Grant Number JP20K19767 and JST, PRESTO Grant Number JPMJPR20M9, Japan.

References

[1] J. Han et al., *Proc. ETS*, pp. 1-6, 2013.
[2] H. Esmaeilzadeh et al.,*Proc. ASPLOS*, pp. 301-312, 2012.
[3] R. Hegde et al., *IEEE TVLSI*, vol. 9, no. 6, pp. 813-823, 2001.
[4] A. B. Kahng et al., *Proc. ASP-DAC*, pp. 825-831, 2010.
[5] N. J. Wang et al., *Proc. DSN*, pp. 61-70, 2004.
[6] F. F. D. Santos et al., *Proc. DSN*, pp. 292-304, 2021.
[7] N. J. Wang et al., *IEEE TDSC*, vol. 3, no. 3, pp. 188-201, 2006.
[8] R. Venkatagiri et al., *Proc. MICRO*, pp. 1-14, 2016.
[9] Y. Zhang et al., *Proc. DATE*, pp. 60-63, 2022.
[10] J. Constantin et al., *Proc. DAC*, pp. 1-6, 2016.
[11] B. C. Schafer et al., *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 53-56, 2014.
[12] Signal and Image Processing Institute, https://sipi.usc.edu/database/.