Invited talk: Automated synthesis of approximate circuits

Zdenek Vasicek
Brno University of Technology, Faculty of Information Technology,
IT4Innovations Centre of Excellence
Brno, Czech Republic
Email: vasicek@fit.vutbr.cz

***Abstract*—Many fundamentally new and different approaches have recently been introduced under the term of approximate computing. The concept of approximate computing investigates the idea of accepting a certain level of inaccuracy in computations to reduce complexity and other non-functional properties of digital systems. Among others, technology-independent functional approximation can be employed. The idea of functional approximation is to implement a slightly different function to the original one provided that the accuracy is kept at desired level and the power consumption or other design parameters are reduced adequately. Functional approximation can be performed manually but the current trend is to develop fully automated functional approximation methods that can be integrated into computer aided design tools for digital circuits.**

In this talk, we start with an overview of the methods for functional approximation. Then, we focus on the search-based functional approximation. The search-based synthesis is based on a heuristic procedure (e.g. an evolutionary algorithm) that gradually modifies the original implementation. This procedure is typically repeated iteratively in order to improve the current approximate implementation in the subsequent steps. The advantage of this approach is the ability to produce high-quality approximate circuits. Unfortunately, the search-based synthesis is, in general, computationally expensive (hundred thousands of iterations are typically evaluated). Hence, we will discuss various approaches for improving the scalability of this method and reducing the computational requirements. In particular, advanced methods of formal error analysis using binary decision diagrams and satisfiability problem solving will be described. Finally, recent results obtained using the search-based synthesis in various problem domains (design of power-efficient approximate arithmetic components and their usage in the area of machine learning) will be discussed.

I. Approximate computing

To address the relentless rise in demand for energy-efficient systems, many fundamentally new and different approaches have recently been introduced under the term of approximate computing. A good survey of existing techniques can be found for example in [1], [2]. The concept of approximate computing investigates the idea of accepting a certain level of inaccuracy in computations to reduce complexity and other non-functional properties of digital systems. The key motivation behind the approximate computing is the inherent error resilience of many real-world applications. For example, Chippa et al. reported that about more than 83% of runtime is spent in computations that can be approximated [3]. The inherent error resilience means that it is not always necessary to implement precise and usually area expensive circuits. Instead, much simpler, approximate, circuits may be used to solve a given problem without any significant degradation in the output quality.

II. Functional approximation

Among others, technology-independent *functional approximation* can be employed. The idea of functional approximation is to implement a slightly different function to the original one provided that the accuracy is kept at desired level and the power consumption or other electrical parameters are reduced adequately. Functional approximation can be performed manually but the current trend is to develop fully automated functional approximation methods that can be integrated into computer aided design tools for digital circuits.

Approximation can be done at various levels. One of the possibilities how to approximate a given circuit is to make a change at the level of behavioral description, e.g. in the truth table. This idea was used, for example, by Kulkarni et al. who manually designed a small 2-bit approximate multiplier consisting of five gates [4]. Kulkarni modified a single row of the truth-table of 2-bit multiplication. He obtained an approximate multiplier that outputs 7 for 3x3 instead of 3x3=9. This form of inaccuracy enabled not only to improve area, but also to reduce the number of output signals and improve delay when applied in a large multiplier where long carry chains are typically present. Unfortunately, the approach based on truth table modification is hardly applicable in practice. Firstly, it does not scale well since the number of rows grows exponentially with the number of inputs. Secondly, the modification needs to be performed in such a way that it leads to a more efficient implementation compared to the original implementation. A smaller, faster or power-efficient variant is typically requested. Unfortunately, we can not easily predict how a given modification projects to the considered electrical parameters such as area, delay or power consumption.

To overcome limitations of the truth-based approach, an analytical approach to approximate design can be employed. Many analytical approaches have been proposed in the literature to approximate key arithmetic circuits such as adders and multipliers [5], [6]. In the case of multipliers, generation of partial products, the summation tree, counters, or compressors are approximated. The approximation is typically conducted either at the level of common gates or at the level of more complex building blocks such as full adder cells. Among others, truncation (bit-width reduction) represents a straightforward and efficient approach to perform approximation on the partial product tree. The key idea is to remove k least significant bits of the inputs operands. It means that a smaller $(n-k)$-bit multiplier is utilized instead of an accurate n-bit multiplier. The $2k$ least significant bits of the final product are always zero. Currently, a wide range of more or less efficient analytical approaches is available in literature. The advantage of the analytical approach is the possibility to exactly formulate the error parameters as a function of the number of data bits. Despite of that, there is no general approach that could be applicable to arbitrary circuit because each method is tightly connected with a particular architecture that is approximated.

III. Automated synthesis of approximate circuits

In contrast to the analytical approaches, various general-purpose approximation methods have been proposed. The goal is to obtain an approach performing automatic approximation of digital circuits independently of their structure. These methods typically employ various heuristics to identify circuit parts suitable for approximation. The Systematic methodology for Automatic Logic Synthesis of Approximate circuits (SALSA) is one of the first approaches that addresses the problem of approximate synthesis [7]. In order to be able use the existing synthesis tools, the authors mapped the problem of approximate synthesis into an equivalent problem of traditional logic synthesis – don't care based optimization. SALSA starts with a description of the exact version of the circuit and an error constraint that specifies the type and amount of error that the approximate implementation can exhibit. The methodology introduces the so-called Q-function which takes the outputs from both the original circuit and approximate circuit and decides if the quality constraints are satisfied. The Q-function outputs a single Boolean value. The SALSA algorithm attempts to modify the approximate circuit with the goal of keeping the output of the Q-function unchanged. The modification is driven by the concept of the observability don't cares. Another systematic approach, Substitute-And-SIMplIfy (SASIMI), tries to identify signal pairs in the circuit that exhibit the same value with a high probability, and substitutes one for the other [8]. These substitutions introduce functional approximations. Unused logic can be eliminated from the circuit which results in area and power savings. A different approach was proposed by Lingamneni et. al that employed a probabilistic pruning, a design technique that consists of removing circuit blocks and their associated wires to trade exactness of computation against power, area, and delay saving [9].

IV. Search-based functional approximation

Compared to the approximate design based on truth-table modification, the pruning-like techniques are limited considering the ability to generate novel circuit structures. In fact, they identify the largest sub-circuit that can be discarded from the original circuit without violating a user-specified error constraint. Neither the probabilistic pruning nor probabilistic substitutions allow to replace a part of the original circuit with a sub-circuit that does not form a part of the original circuit.

In order to address this issue, a more advanced approaches have been introduced recently. Various evolutionary algorithms have been applied to accomplish approximations [10], [11], [12]. The problem of approximate synthesis is mapped to a search-based design problem. An automated circuit approximation procedure is seen as a multi-objective search process in which a circuit satisfying user-defined constraints showing desired trade-off between the quality and other electrical parameters is sought within the space of all possible implementations. A heuristic procedure (e.g. an evolutionary algorithm) that gradually modifies the original implementation is typically utilized. The modification procedure can affect either the node function (e.g. AND node can be modified to inverter or vice versa), node input connection, or primary output connection. It is thus able to not only disconnect gates but also to introduce new gates (by activating redundant gates). Both can happen by changing either primary output connection or node input connection or by changing node function resulting in increasing/decreasing of its arity.

Among others, the following approaches have been proposed. Nepal et al. introduced a technique for automated behavioral synthesis of approximate computing circuits (ABACUS) [13]. The method first creates an abstract synthesis tree (AST) from the input behavioral description, and then applies variant operators to the AST. ABACUS uses a simple greedy search algorithm to modify AST. In order to approximate gate-level digital circuits, Sekanina and Vasicek introduced an evolutionary approach based on Cartesian Genetic Programming (GP) [10], [11]. The reasons for using the advanced evolutionary approach was that the population-based approach suits well in finding multiple solutions and its niche-preservation methods can be exploited to discover diverse solutions. As shown in [14], this approach is able to produce high-quality approximate circuits that are unreachable by traditional approximate techniques. In the context of FPGAs, circuit approximation has been introduced and evaluated by means of the GRATER tool [12]. GRATER uses a genetic algorithm to determine the precision of variables within an OpenCL kernel. By selectively reducing the precision, the number of parallel approximate kernels that can be mapped in the fixed area budget of an FPGA can be increased with respect to the original kernel implementations.

V. Current issues

One of the open problems of search-based approximate methods is the limited scalability caused by various factors. The search-based approximate methods are typically constructed as iterative methods in which several candidate approximate circuits have to be generated and evaluated in terms of functional (quality) and non-functional requirements (electrical parameters). The functionality is expressed using one or several error metrics such as error probability, average-case error, or worst-case error. Unfortunately, the search-based synthesis is, in general, computationally expensive (hundred thousands of iterations are typically evaluated). Hence, the evaluation needs to be fast as it has a great impact on the scalability of the whole design process.

To improve scalability, many authors simplify the problem and evaluate the functionality of approximate circuits by applying a set of input vectors. Monte Carlo simulation is typically utilized to measure the error of the output vectors with respect to the original solution [8], [13], [6]. Unfortunately, this approach provides no guarantee on the error and make it difficult to predict the behavior of an approximate circuit under different conditions. For example, a completely different output value may be produced when a nasty approximate multiplier is employed in a neural network. There is currently a clear need to come up with a new approach to the problem of evaluating the quality of approximate complex digital systems. If the exact error of the approximation has to be determined, *formal relaxed equivalence checking* is requested, stressing the fact that the considered systems will be checked to be equal up to some bound with respect to a suitably chosen error metric [15]. This research area is rather unexplored because almost all formal approaches have been developed for exact equivalence checking.

Currently, the SAT solver based equivalence checking represents a method of the first choice. The SAT solvers can be used also for the quality analysis of approximate circuits. A common approach is to construct an auxiliary circuit referred to as approximation miter. This circuit instantiates both the candidate approximate circuit and the accurate (reference) circuit and compares their outputs to quantify the error. The comparison is typically ensured by means of an error computation block (e.g. subtracter followed by a circuit which determines absolute value is used for arithmetic quality metrics). For computing the worst-case error, the approximation miter is converted to a CNF formula and the resulting formula is used together with an objective function as input of the SAT solver. Usually a variant of binary search is applied to determine the wost-case error. A much simpler task is to check whether a predefined worst-case error is violated by the candidate approximate circuit. In this case, the error computation block is followed by a comparator that has a binary output. Unfortunately, no practically useful method capable of establishing the average-case error, error rate and total Hamming distance using a SAT-based solver has been proposed up to now. In addition to that, there are some pathological cases of circuits for which SAT does not scale well. Sadly, not only multipliers but also integer division, remainder, square root and reciprocal exhibit in general exponential run-time. Despite of that, Ceska et. al. was able to approximate various 32-bit multipliers and 128-bit adders using a conflict-driven SAT solver [16]. Similarly to Sekanina and Vasicek, an evolutionary approach based on GP was used. In order to avoid calculation of the exact value of WCE, the predefined error level is used as constrain. To compute whether the chosen level is violated, the concept of approximation miter was utilized. The SAT solver is given a predefined amount of time that can be used to decide whether the resulting CNF is satisfiable or not. When the runtime exceeds this limit, the SAT solver is terminated and the corresponding candidate solution is discarded. This strategy drives the search towards promptly verifiable candidate solutions and thus provides scalable approximation of complex circuits.

Acknowledgment

This work was supported by the Czech Science Foundation project 16-17538S (Relaxed equivalence checking for approximate computing).

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, 2016.

[2] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, 2016.

[3] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *The 50th Annual Design Automation Conference 2013, DAC'13*. ACM, 2013, pp. 1–9.

[4] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *J. Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.

[5] H. Jiang, J. Han, and F. Lombardi, "A comparative review and evaluation of approximate adders," in *Proc. of GLVLSI'15*. ACM, 2015, pp. 343–348.

[6] H. Jiang, C. Liu, N. Maheshwari, F. Lombardi, and J. Han, "A comparative evaluation of approximate multipliers," in *IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2016, Beijing, China, July 18-20, 2016*, 2016, pp. 191–196.

[7] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC '12*. ACM, 2012, pp. 796–801.

[8] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *Design, Automation and Test in Europe, DATE'13*. EDA Consortium San Jose, CA, USA, 2013, pp. 1–6.

[9] A. Lingamneni, C. Enz, J. L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[10] Z. Vasicek and L. Sekanina, "Evolutionary design of approximate multipliers under different error metrics," in *IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems 2013*. IEEE, 2014, pp. 135–140.

[11] ——, "Evolutionary approach to approximate digital circuits design," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 432–444, 2015.

[12] A. Lotfi, A. Rahimi *et al.*, "Grater: An approximation workflow for exploiting data-level parallelism in FPGA acceleration," in *2016 Design, Automation Test in Europe Conf. Exhibition*, ser. DATE '16. EDA Consortium, March 2016, pp. 1279–1284.

[13] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "Abacus: A technique for automated behavioral synthesis of approximate computing circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '14. EDA Consortium, 2014, pp. 1–6.

[14] 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.

[15] Z. Vasicek, "Relaxed equivalence checking: a new challenge in logic synthesis," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 1–6.

[16] 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.