ADAC: Automated Design of Approximate Circuits*

Milan Češka, Jiří Matyáš, Vojtech Mrazek,
Lukas Sekanina, Zdenek Vasicek, Tomáš Vojnar

Faculty of Information Technology,
Centre of Excellence IT4Innovations
Brno University of Technology, Czech Republic

Abstract. Approximate circuits with relaxed requirements on functional correctness play an important role in the development of resource-efficient computer systems. Designing approximate circuits is a very complex and time-demanding process trying to find optimal trade-offs between the approximation error and resource savings. In this paper, we present ADAC—a novel framework for automated design of approximate arithmetic circuits. ADAC integrates in a unique way efficient simulation and formal methods for approximate equivalence checking into a search-based circuit optimisation. To make ADAC easily accessible, it is implemented as a module of the ABC tool: a state-of-the-art system for circuit synthesis and verification. Within several hours, ADAC is able to construct high-quality Pareto sets of complex circuits (including even 32-bit multipliers), providing useful trade-offs between the resource consumption and the error that is formally guaranteed. This demonstrates outstanding performance and scalability compared with other existing approaches.

1 Introduction

In the recent years, reduction of power consumption of computer systems and mobile devices has become one of the biggest challenges in the computer industry. *Approximate computing* has been established as a new research field aiming at reducing system resource demands (and, in particular, power demands) by relaxing the requirement that all computations are always performed correctly. Approximate computing exploits the fact that many applications, including image and multimedia processing, signal processing, data mining, machine learning, neural networks, and scientific computations, are *error resilient*, i.e. produce acceptable results even though the underlying computations are performed with a certain error. Therefore, the error can be used as a design metric and traded for chip area, power consumption, or runtime. Chippa et al. [7] claims that almost 80 % of runtime is spent in procedures that could be approximated.

Approximate computing can be conducted at different system levels with arithmetic circuit approximation being one of the most popular as such circuits are frequently used in the core computations. In our work, we focus on functional approximation where the original circuit is replaced by a less complex one which exhibits some errors but improves non-functional circuit parameters such as power consumption or chip area. Circuit approximation can be formulated as an optimisation problem where the error and non-functional circuit parameters are conflicting design objectives. Designing complex approximate circuits is a time-demanding and error-prone process. Moreover, its automation is challenging too since the design space including candidate solutions is huge and checking that a candidate solution has the required error is itself a computationally demanding task, especially if formal guarantees on the error have to be ensured.

* This work was supported by the IT4Innovations excellence in science project No. LQ1602.

In this tool paper, we present *ADAC*[1]—a novel framework for automated design of approximate circuits. The framework implements a design loop including (i) a *generator* of candidate solutions employing genetic search algorithms, (ii) an *evaluator* estimating non-functional parameters of a candidate solution, and (iii) a *verifier* checking that the candidate solution does not exceed the permissible error. ADAC is integrated as a new module into the ABC tool—a state-of-the-art and widely used system for circuit synthesis and verification [1]. The framework takes as the inputs:

- a golden combinational circuit in Verilog implementing the correct functionality,
- an error metric (such as the worst-case error, mean error, Hamming distance, etc.),
- a threshold on the error metric representing the maximal permissible error,
- a time limit on the overall design process, and
- a file specifying sizes of gates available to the design process.

With these inputs, ADAC searches for an approximate circuit satisfying the error threshold and having the minimal estimated chip area. Previous works [14, 22, 20, 3] confirmed that the chip area is a good optimization objective as it highly correlates with power consumption, which is a crucial target in approximate computing.

The results of [21] clearly demonstrate that search algorithms based on *Cartesian Genetic Programming* (CGP) [12] are well capable of generating high-quality approximate circuits. For complex circuits, however, a high number of candidate solutions has to be generated and evaluated, which significantly limits the scalability of the design process. Our framework implements several approaches for error evaluation suitable for different error metrics and application domains. They include both *SAT and BDD-based techniques* for approximate equivalence checking providing *formal error guarantees* as well as a *bit-parallel circuit simulation* utilising the computing power of modern processors. We also implement a novel search strategy that drives the search towards *promptly verifiable approximate circuits*, which significantly accelerates the design process in many cases [3]. As such, the framework offers a unique integration of techniques based on simulation, formal reasoning, and evolutionary circuit optimisation. Our extensive experimental evaluation demonstrates that ADAC offers outstanding performance and scalability compared with existing methods and tools and paves a way towards an automated design process of complex provably-correct circuit approximations.

2 Architecture and Implementation

The ADAC framework has a modular architecture illustrated in Figure 1.

The setup phase is responsible mainly for preparing a chromosome representation of the golden circuit. The circuit is given in a high-level Verilog format, which is first translated to a gate-level representation using the tool Yosys [25], and then the chromosome representation is obtained using our V2CH script. The setup phase is also responsible for generating a configuration file controlling the main design loop. It is generated from the user inputs and optional parameters for CGP and search strategies.

The design loop consists of three components: (i) a generator of candidate designs, (ii) an evaluator of non-functional parameters of the candidate circuit (currently estimating the chip area), and (iii) a verifier evaluating the candidate error. The chip area and the error form a basis of the *fitness function*, whose value is minimised via our search strategy. In particular, the fitness is infinity if the circuit error exceeds the given

[1] https://github.com/imatyas/ADAC

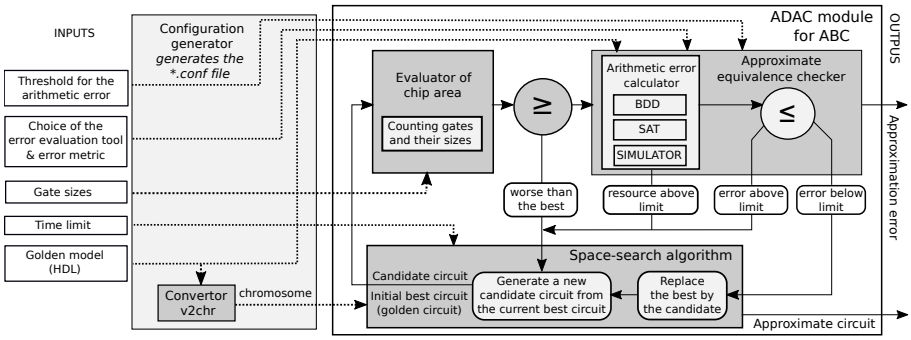

Fig. 1: A scheme of the ADAC architecture.

threshold, and the chip area otherwise. In the future, we plan to support a more general specification of the fitness. As an additional feature, ADAC can also quantify the difference (in the given metric) between two given circuits.

The real values of non-functional parameters, such as the chip area or the power-delay product (PDP), depend on the target technology, and the synthesis of an optimal implementation of the given circuit using the target technology is highly time-consuming. Therefore, our design loop currently uses the *chip area* as the sole non-functional parameter. The chip area is estimated as the sum of the sizes of the gates of the circuit, which are given as one of the inputs of ADAC. The chip area is typically a good estimate of the power consumption [14, 22, 20, 3]. The output of ADAC (in the gate-level Verilog format) can be passed to industrial circuit design tools to obtain accurate circuit parameters for the target technology. In our experiments, we report PDP for the 45 nm technology synthesised by the Synopsys Design Compiler [19].

We now briefly describe the candidate circuit generator and three methods for error evaluation that are currently supported in ADAC.

The *candidate circuit generator* is based on CGP where a candidate solution is encoded as a chromosome describing an oriented acyclic graph, given as a 2-dimensional array of 2-input nodes. Every node is numbered and is encoded by 3 integers where the first two numbers denote the inputs and the third represents the function of the node. New candidate circuits are obtained using a mutation operator that performs random changes in the chromosome. The mutations can either modify the node interconnection or functionality. The area of candidate circuits is reduced by making some nodes unreachable (such nodes, however, are removed only at the very end, and so they can still be mutated and even become reachable again). The candidates are evaluated, and the one with the best one is used in the next iteration of the design loop. The whole loop starts with the golden circuit and iteratively generates approximate solutions with better fitness values until a termination criterion (typically a given time limit) is met. Optionally, user can provide approximate circuit satisfying the threshold on the error as a seed to start with.

The *bit-parallel circuit simulation* supports all common error metrics, including the worst-case error (WCE), the mean error, the error rate representing the number of inputs leading to an incorrect output, and the Hamming distance. It utilises the power of modern processors by simulating the circuit on multiple inputs vectors (e.g. 64 inputs for 64-bit processors) in a single pass through the circuit [24]. However, despite the parallel processing that significantly accelerates the simulation, for circuits with arguments

of larger bit-widths (beyond 12 bits), it is not feasible to simulate the circuits on all possible inputs, and so statistical guarantees on the approximation error are provided only.

The *BDD-based evaluation* also supports all common error metrics, and, unlike simulation, it is able to provide formal error guarantees for circuits with larger input bit-widths. For the purpose of the evaluation, the original correct circuit and its approximation are interconnected into an auxiliary circuit called a *miter* such that the error can be deduced from its output (e.g. to compute the error rate, the outputs of the golden and candidate circuits are subtracted, and the result is compared with 0). The miter is encoded as a BDD on which the circuit error is evaluated using BDD operations [22, 23]. However, this technique does not scale well with the complexity of the circuits in terms of the number of their gates as the resulting BDD representation becomes prohibitively huge. Hence, this approach works well for large adders and similar circuits, but, it fails, e.g., for multipliers beyond 12-bits.

The *SAT-based evaluation* currently supports WCE only, but it provides formal guarantees and a superior performance to the BDD-based technique. ADAC implements a novel miter construction based on subtracting the output of the golden and approximate circuit, followed by a comparison with the error threshold [3]. The construction is optimised for SAT-based evaluation by avoiding long XOR chains known to cause poor performance of state-of-the-art SAT solvers [5, 9]. This allows us to exploit the ABC engine `iprove`, designed originally for miter-based exact circuit equivalence checking, to quickly evaluate WCE.

The final ingredient of the design process is the *search strategy*. Apart from the standard evolutionary strategies based solely on the fitness function, ADAC also implements a novel verifiability-driven approach [3] combined with the SAT-based evaluation.

The *verifiability-driven search strategy* uses a limit L on the resources available to the underlying SAT decision procedure. The limit effectively controls the time the SAT solver can use. We require that every improving candidate has to be verifiable using the resource limit L. Therefore the strategy drives the search towards candidates that improve the fitness and can be promptly evaluated. As the result, we can evaluate in the given time a much larger set of candidate circuits. Our experiments indicate that this strategy often leads to a higher number of improving solutions and thus finds circuits having a smaller chip area meeting the permissible error. On the other hand, it can happen that, for a limit L, no improving sequence exists, while it exists for a slightly greater resource limit. We are currently implementing auto-adaptive techniques that should automatically select the adequate resource limit for the given circuit.

Integration to the ABC tool. To make ADAC easily accessible, it is implemented as a new module for the ABC tool. ABC allows us to support an important subset of the Verilog specification and implementation language. We also utilize ABC to translate the circuits among different intermediate representations used for constructing miters. As mentioned before, we employ the `iprove` engine in our SAT-based method for evaluating the WCE. Note that `iprove` uses MiniSat [18] as the SAT solver. Despite the fact that ABC supports a BDD-based circuit representation and manipulation, we implemented our own BDD component (based on the BuDDy library [2]) that is tailored for evolutionary circuit approximation.

	Bit-width of the arguments		
	w = 6	w = 10	w = 14
Simulation	**210μs**	76 ms	31.23 s
BDD ϵ_{wce}	350 μs	12 ms	0.38 s
speedup	0.59×	6.04×	80.74×
BDD ϵ_{me}	370 μs	**13ms**	**0.79s**
speedup	0.59×	5.72×	38.94×
SAT ϵ_{wce}	920 μs	**1.4ms**	**1.7ms**
speedup	0.23×	53.7×	18468×

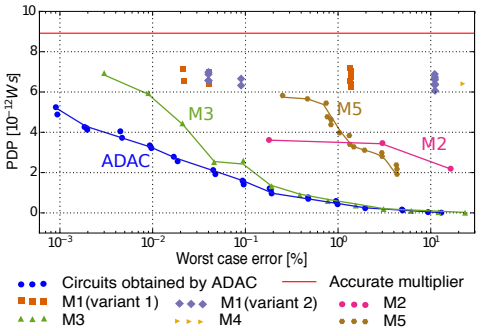

Fig. 2: (Left) Performance of error evaluation methods for adders. (Right) A comparison of 16-bit approximate multipliers designed by ADAC vs. the best known solutions.

Extensibility. Due to its modular architecture, ADAC can be easily extended. Apart from the extensions mentioned above, we are working on a new component for error evaluation based on SAT counting methods (e.g. #SAT [4]) that could offer formal guarantees and a better scalability for the mean error and error-rate metrics, and on new candidate circuit generators counter-examples produced during the verification of candidate circuits. In a long term perspective, we plan to generalise the underlying methods and support also design of approximate sequential circuits.

3 Evaluation, Related Works, and Applications

We first compare the performance of the different methods of circuit error evaluation supported in ADAC. For that, we use results from adder approximation obtained from 10 runs, each for 5 minutes. The table in Fig. 2 shows average runtimes of a single error evaluation using the bit-parallel simulation, the BDD-based approach, and the SAT-based approach. The reported speedups are with respect to the simulation. We can see that the simulation provides the best performance for small bit-widths only, but it does not scale well The SAT-based method offers the best scalability and dominates for larger circuits, but it supports the WCE evaluation only. The BDD-based method, like simulation, supports all metrics and significantly outperforms the simulation for larger circuits. Note that, for more complex circuits such as multipliers, we would observe similar results with a worse relative performance of the BDD-based approach.

There indeed exist also other known methods for computing approximation errors for arithmetic circuits, including methods based on BDDs [6] or a SAT-based miter solution [5]. Comparing to ADAC, these methods are less scalable, which is demonstrated by the fact that they have been used for approximating multipliers limited to 8-bit operands and adders limited to 16-bit operands only. Apart from that, there are efficient methods for *exact* equivalence checking based on algebraic computations [8, 16]. However, they are so far not known for approximate equivalence checking.

Next, we compare the quality of approximate circuits obtained using ADAC with circuits that appeared in the literature. We consider 16-bit multipliers since existing approaches are not able to handle larger and more complex circuits. The different points in Fig. 2 correspond to circuits with different trade-offs between WCE in % and the power-delay product (PDP[2]), which is a key non-functional circuit characteristic. These

[2] PDP characterises both the speed and energy efficiency of the circuit.

circuits were obtained using various existing approaches including: (M1) configurable circuits from the lpACLib library [17], (M2) the bit-significance-driven logic compression [15], (M3) the bit-width truncation [10], (M4) compositional techniques [11], and (M5) circuits from the EvoApproxLib library [13]. We can see that just the bit-width truncation can provide a quality of results comparable with ADAC (in terms of the PDP reduction for the given WCE), but for large target errors (20 % WCE or more) only. For small target errors, ADAC clearly dominates.

Note that, for each target WCE, we performed 30 independent runs of CGP to obtain statistically significant results. For each run, ADAC was executed for 2 hours on an Intel Xeon X5670 2.4 GHz processor using a single core. Also note that the individual runs are independent and thus can be easily parallelised.

Further, Fig. 3 presents approximate multipliers up to 32 bits obtained by ADAC. It shows Pareto fronts representing circuits with different compromises between WCE in % and PDP, and demonstrates that ADAC goes beyond capabilities of existing methods and tools. For each target WCE, ADAC was executed for 4 hours in the case of the 24-bit instances and for 6 hours in the case of the larger instances. Note that a 32-bit exact multiplier requires over 6,300 gates, and, to the best of our knowledge, ADAC is the first tool that is able to approximate such complex circuits with formal error guarantees.

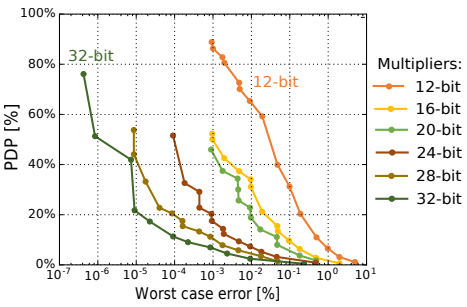

Fig. 3: Approximate multipliers designed by ADAC. 100 % refers to PDP of the accurate circuits for the given bit-width.

Besides the approaches mentioned above, there also exist general-purpose methods, such as SALSA [14] or SASIMI [15], approximating circuits independently of their structure. We were unable to perform a direct comparison with them due to their implementation is not available, but based on the published results, ADAC is able to provide a significantly better scalability.

Practical impacts. The following list briefly characterises several resource-aware applications that build on approximate circuits. The circuits were obtained using prototype implementations of the above mentioned approaches that are now integrated in ADAC.
Approximate multipliers for convolutional neural networks [14]. In such networks, millions of multiplications have to be performed. The usage of application-specific approximate multipliers led to 90 % savings in terms of power consumption of the data path for a negligible drop in classification accuracy.
Approximate adders and subtractors for a discrete convolutional transformation [22]. These adders and subtractors were designed to reduce the power consumption in video compression for the High Efficiency Video Coding (HEVC) standard. They show better quality/power trade-offs than implementations available in the literature. For example, a 25 % power reduction for the same error was obtained in comparison with a recent highly-optimised implementation.
Approximate adders and multipliers for image processing [20]. These circuits were used in the development of efficient hardware implementations of filters and edge detectors. A 50 % reduction was observed in the number of look-up tables used in a field programmable gate array for a negligible drop in the image visual quality.

References

1. Brayton, R., Mishchenko, A.: ABC: An academic industrial-strength verification tool. In: Proc. of CAV'10. Springer (2010)
2. BuDDy: A BDD package. http://buddy.sourceforge.net/manual/main.html, online; January 2018
3. Češka, M., Matyáš, J., Mrazek, V., Sekanina, L., Vasicek, Z., Vojnar, T.: Approximating complex arithmetic circuits with formal error guarantees: 32-bit multipliers accomplished. In: Proc. of ICCAD'17. pp. 416–423. IEEE (2017)
4. Chakraborty, S., Meel, K.S., Mistry, R., Vardi, M.Y.: Approximate probabilistic inference via word-level counting. In: Proc. of AAAI'16. pp. 3218–3224. AAAI Press (2016)
5. Chandrasekharan, A., Soeken, M., Große, D., Drechsler, R.: Precise error determination of approximated components in sequential circuits with model checking. In: Proc. of DAC'16. pp. 129:1–129:6. ACM (2016)
6. Chandrasekharan, A., Soeken, M., et al.: Approximation-aware rewriting of AIGs for error tolerant applications. In: Proc. of ICCAD'16. pp. 83:1–83:8. ACM (2016)
7. Chippa, V.K., Chakradhar, S.T., Roy, K., Raghunathan, A.: Analysis and characterization of inherent application resilience for approximate computing. In: Proc. of DAC'13. pp. 1–9. IEEE (2013)
8. Ciesielski, M., Yu, C., Brown, W., Liu, D., Rossi, A.: Verification of gate-level arithmetic circuits by function extraction. In: Proc. of DAC '15. ACM (2015)
9. Han, C.S., Jiang, J.H.R.: When boolean satisfiability meets gaussian elimination in a simplex way. In: Proc. of CAV'12. pp. 410–426. Springer (2012)
10. Jiang, H., Liu, C., Liu, L., Lombardi, F., Han, J.: A review, classification, and comparative evaluation of approximate arithmetic circuits. J. Emerg. Technol. Comput. Syst. 13(4), 60:1–60:34 (2017)
11. Kulkarni, P., Gupta, P., Ercegovac, M.D.: Trading accuracy for power in a multiplier architecture. J. Low Power Electronics 7(4), 490–501 (2011)
12. Miller, J.F.: Cartesian Genetic Programming. Springer-Verlag (2011)
13. Mrazek, V., Hrbacek, R., et al.: Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In: Proc. of DATE'17. pp. 258–261. EDAA (2017)
14. Mrazek, V., Sarwar, S.S., Sekanina, L., Vasicek, Z., Roy, K.: Design of power-efficient approximate multipliers for approximate artificial neural networks. In: Proc. of ICCAD'16. pp. 81:1–81:7. ACM (2016)
15. Qiqieh, I., Shafik, R., et al.: Energy-efficient approximate multiplier design using bit significance-driven logic compression. In: Proc. of DATE'17. EDAA (2017)
16. Sayed-Ahmed, A., Große, D., et al.: Formal verification of integer multipliers by combining Gröbner basis with logic reduction. In: Proc. of DATE'16. pp. 1048–1053. IEEE (2016)
17. Shafique, M., Ahmad, W., et al.: A low latency generic accuracy configurable adder. In: Proc. of DAC'15. pp. 86:1–86:6. ACM (2015)
18. Sorensson, N., Een, N.: MiniSat v1.13 – a sat solver with conflict-clause minimization. SAT 2005(53), 1–2 (2005)
19. Synopsys design compiler. https://www.synopsys.com/, online; January 2018
20. Vasicek, Z., Mrazek, V., Sekanina, L.: Evolutionary functional approximation of circuits implemented into fpgas. In: Proc. of SSCI'16. pp. 1–8. IEEE (2016)
21. Vasicek, Z., Sekanina, L.: Evolutionary approach to approximate digital circuits design. Transactions on Evolutionary Computation pp. 432–444 (2015)
22. Vasicek, Z., Mrazek, V., Sekanina, L.: Towards low power approximate DCT architecture for HEVC standard. In: Proc. of DATE'17. pp. 1576–1581. EDAA (2017)

23. Vasicek, Z., Sekanina, L.: Evolutionary design of complex approximate combinational circuits. Genetic Programming and Evolvable Machines 17(2), 169–192 (2016)
24. Vašíček, Z., Slaný, K.: Efficient phenotype evaluation in cartesian genetic programming. In: Proc. of EuroGP'12. pp. 266–278. Springer (2012)
25. Wolf, C.: Yosys open synthesis suite. http://www.clifford.at/yosys/, online; January 2018