Statistical Model Checking of Approximate Circuits: Challenges and Opportunities

The author's details are intentionally left blind due to the paper submission rules available at
https://www.date-conference.com/submission-instructions
"*Double blind review process: manuscripts should not include the author names nor affiliations.*"

***Abstract*—Many works have shown that approximate circuits may play an important role in the development of resource-efficient electronic systems. This motivates many researchers to propose new approaches for finding an optimal trade-off between the approximation error and resource savings for predefined applications of approximate circuits. The works and approaches, however, focus mainly on design aspects regarding relaxed functional requirements while neglecting further aspects such as signal and parameter dynamics/stochasticity, relaxed/non-functional equivalence, testing or formal verification. This paper aims to take a step ahead by moving towards the formal verification of time-dependent properties of systems based on approximate circuits. Firstly, it presents our approach to modeling such systems by means of stochastic timed automata whereas our approach goes beyond digital, combinational and/or synchronous circuits and is applicable in the area of sequential, analog and/or asynchronous circuits as well. Secondly, the paper shows the principle and advantage of verifying properties of modeled approximate systems by the statistical model checking technique. Finally, the paper evaluates our approach and outlines future research perspectives.**

***Index Terms*—approximate circuit, error, trade-off, relaxed equivalence, formal verification, timed automaton, stochastic automaton, modeling, simulation, statistical model checking**

I. Introduction

Approximate computing (AxC) builds on the observation that in some applications, errors can be tolerated if they provide an acceptable trade-off between the output quality and computation effort [1]. Designers use this characteristic to apply selective approximations or occasional relaxations of the specifications [2].

As a result, many approximate concepts, methods and designs have been proposed in many works to take advantage of AxC, representatives of which are detailed in II-A. Existing works show that great efforts have been made regarding the design and automated generation of approximate combinational digital circuits, few of them have dealt with sequential AxC circuits, their verification or testing. However, many problems regarding, e.g., reconfigurable AxC circuits, parameter variability of AxC circuits due to aging/stress, signal instability or equivalence in the time domain have not been opened yet in the works. This paper proposes to solve the problems using the so-called statistical model checking (SMC) technique detailed in II-B. Our approach and results we have achieved so far are summarized in III. Sect. IV concludes the paper.

II. Research Background

A. Representative Approaches

An overview regarding AxC systems can be found in the broad-range book [1] and in the design-oriented book [3]. Further works cover more specific areas. For example, [4] presents an evaluation framework. other works deal with designing AxC circuits, typically adders or multipliers [5]–[7], but also sensors, processors, memories, neural networks or compilers [3]. Further works focus on formal verification, particularly on the accumulated error [8], relaxed equivalence checking [9] or sequential circuits [1]. Few works, such as [2], [10], [11], deal with testing of AxC circuits.

B. Reasoning and Instruments of our Research

Our research activity concentrates on modeling and verification of AxC systems in their application-specific input/output scope ("context" in brief). Apparently, contexts for various applications may differ whereas the context for a simpler application (e.g., using an adder to sum n consecutive integers starting by 0) is more predictable than the context for a complex application (e.g., using an adder to sum the partial CPU loads of dynamically scheduled real-time tasks). To succeed in our research effort, we must be able to reconstruct contexts first – using a computational model in our case. For simpler applications, a precise model can be constructed easily. For complex applications, however, it makes more sense to construct a stochastic rather than a precise model.

In our approach, we use the means [12] of stochastic timed automata (STA) to construct a model of an AxC system and its context. To formally verify properties of such model(s), we use the statistical model checking (SMC) technique [13]. Simply said, SMC technique conducts simulations over a stochastic model, monitors them and processes them statistically to infer, with a predefined degree of confidence, whether they provide a statistical evidence for the satisfaction/violation of a property. SMC techniques are advantageous due to the following facts. Firstly, they replace the binarity (regarding the satisfaction) by the ability to quantify the impact of a change in a system with a given degree of uncertainty [14]. Practically, this allows one to get estimates of the probability measure on the satisfaction of a property, not just producing a simple "Yes"/"No" answer. Secondly, they easily scale to industrial size systems [15]. SMC has already been successfully applied to many circuit related problems [16]–[19], but an application in the AxC domain is missing.

C. Analyzing Application Contexts

Despite general applicability of our approach, we limited this paper just to application-specific problems that need a multiplier to be solved. From the function viewpoint, a multiplier has two inputs (x, y) and one output (z). Typically, for n-bit wide x, y the bit-width of z is $2 \times n$, but this may differ in the AxC domain [5]. Tab. I illustrates the truth-table for $n = 2$: the accurate multiplier needs 4 output bits (green background) while the approximate multiplier needs just 3 output bits (red background) as the most significant bit is 0. Binary values in the table do not differ except the value for $x_1x_0 = y_1y_0 = 11$ whereas the outputs 1001 and (0)111 belongs to the accurate and approximate multiplier, resp.

TABLE I
TRUTH TABLE FOR 2×2-BIT ACCURATE/APPROXIMATE MULTIPLIER

x_1x_0 \ y_1y_0	00	01	10	11
00	0000	0000	0000	0000
01	0000	0001	0010	0011
10	0000	0010	0100	0110
11	0000	0011	0110	**1001** **0111**
	$z_3z_2z_1z_0$			

For selected multiplier applications, we gathered statistics about data being propagated to x, y. Fig. 1 illustrates that the statistics differ across various applications like a) searching the rank of 100 8-character strings, b) computing the binom. coef. $C(i, j)$ for integers i, j ranging from 0 to 16 or c) computing the Catalan number for $i = 0, 1, \ldots, 9$.

Fig. 1 shows that some (x, y) pairs occur at multiplier inputs much frequently (such a region is marked by a red rectangle) while some pairs occur rarely or never (yellow rectangle). If a set of applications is known a designer can use the statistics to optimize his general-purpose design for applications from the set. For example, as the yellow rectangle is large and many input values are not likely to appear regarding Fig. 1b, the corresponding design can be approximated well; trivially, "yellow" inputs may be ignored, processed by a hash function etc. Hopefully, this will lead to a better trade-off among parameters such as the accuracy, speed or energy consumption.

D. Challenges and Opportunities

We have decided to initiate our research because we detected multiple challenging tasks to be solved in the AxC area. Especially, we miss a scalable framework capable to i) intuitively describe and then formally evaluate discrete/-continuous/hybrid AxC systems in their application-specific contexts, ii) reflect non-functional (e.g., dynamic) aspects of real systems, such as time/value jitters, concurrency, sequentiality, (a)synchronicity and/or reconfigurability, iii) be open to changes, automated and be easily incorporated/interfaced (in)to commonly used tool-chains. We think that STA/SMC means are capable enough to complete tasks like that. Before we show (by our approach) an idea of how to construct such a framework, let us support this opinion by the following facts. Firstly, STA means allows one to create a model of a dynamic

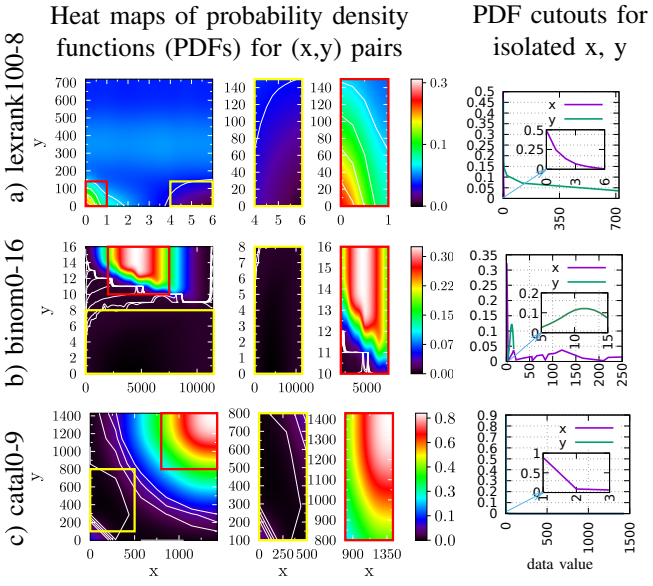

Fig. 1. Statistics of contexts for selected multiplier applications

system using a network of priced timed automata. The automata may be parameterized, are driven by the progression of time, may communicate (be synchronized) and model the price of fired transitions. Secondly, properties (such as the maximum accumulated error of an AxC system) of the network may be checked (using probability estimation, probability comparison, hypothesis testing etc.) by the SMC engine able to verify a property with a predefined degree of confidence.

III. OUR APPROACH

During our research, we have utilized STA/SMC means being implemented in the publicly available UPPAAL SMC tool [20]. Availability of the tool allows one to test our approach, evaluate it and check whether it is applicable to desired areas of interest. This section is organized as follows. Firstly, it gives a top-level view of our approach (III-A) and presents our approach to modeling AxC systems and their contexts by means of STA (III-B). Secondly, it sketches our SMC approach to formal verification of properties of AxC systems and presents results we have achieved (III-C).

A. Top Level View of our Approach

Our approach builds on STA means, allowing one to describe a (stochastic) model of a timed system using a network of (potentially) synchronized automata. Consequently, properties of such a model may be checked by SMC instruments.

To check properties of an AxC system using SMC, we created a framework (Fig. 2) built of blocks detailed in III-B. Let all concepts be explained over Tab. I and Fig. 3.

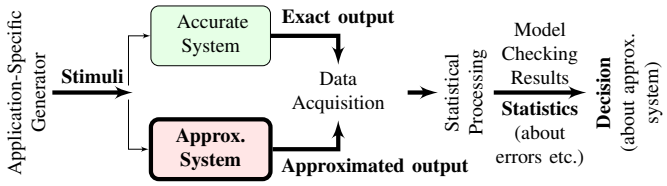

Fig. 2. Block schema of our statistical model checking framework

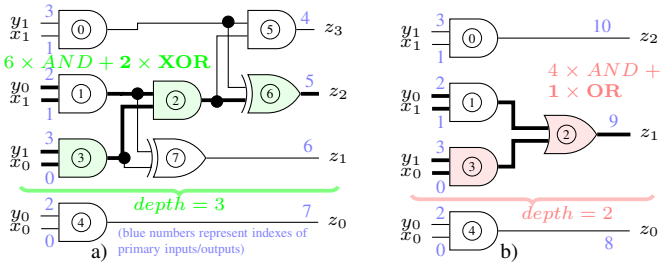

Fig. 3. Gate-level circuits for (a) accurate and (b) approximate multipliers with two 2-bit inputs, one 4-bit and 3-bit output, resp.; redrawn from [5]

B. Aspects of Modeling AxC Systems and their Contexts

Firstly, we need an interface for accurate (Fig. 3a) and approximate (Fig. 3b) systems. In our model, this includes (Listing 1) the amount of primary inputs (NPI) and outputs (NPO), indexes of primary inputs shared by acc. and approx. systems (PI), separate indexes of primary outputs for acc. and approx. systems (POac and POap, resp.).

Listing 1. Top-level interface setup of our framework from Fig. 2+3

```
1 const int NPI = 4;    // # primary input bits
2 const int NPO = 4;    // # primary output bits
3 const int PI[NPI] = {0,1,2,3};  // shared PI indexes
4 const int POac[NPO] = {4,5,6,7};   // accurate PO indexes
5 const int POap[NPO] = {8,9,10,-1}; // approx. PO indexes
```

Secondly, we modeled (Fig. 4a–e) all blocks from Fig. 2 by means of STA; initial states of all models are colored light-green. Particularly, Fig. 4a models a generator able to produce stimuli for x, y according to the corresponding application-specific context. After it is initialized, the model gets a new stimulus, produced by $f()$, and synchronizes waiting models by sending a message via the channel *update* while applying the stimulus to primary inputs. Stimuli are generated until the stop condition is true (e.g., error/coverage is achieved) whereas a new stimulus is produced after the equivalence checking for the previous one ends (waiting for a message via *eqChkDone*).

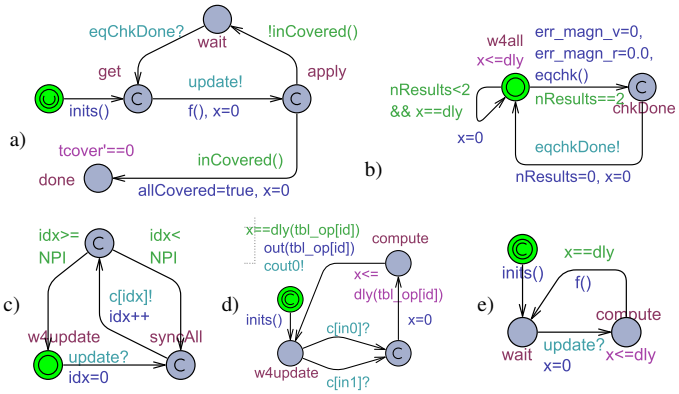

Fig. 4. STA models of key components in our approach: a) stimuli generator, b) equiv. checker, c) primary input sync., d) log. gate, e) approx. multiplier

Fig. 4b depicts our equivalence checker model. Basically, it checks just the functional equivalence, i.e. whether accurate and approximate outputs match or not. In its initial state, the model waits (delay *dly*) until 2 outputs are available ($nResuls == 2$): accurate and approximate. Then, it checks the outputs by *eqchk()* and sends a message via *eqChkDone*. Fig. 4c waits until the stimulus updates (*update?*). Then, it uses the channel c to initiate the update of each primary input (identified by *idx*) according to the stimulus.

Fig. 4d represents our model of a two-input, one-output logical gate. After it is initialized, the model enters *w4update*. Here, it waits until its input *in0* or *in1* changes (*c[in0]?*, *c[in1]?*). If it changes, the model waits (in *compute*) until its propagation delay (*dly(tbl_op[id])*) is over and then, it produces its output (*out(tbl_op[id])*. *id* identifies a gate and *tbl_op[id]* returns a logical function associated with the gate. An approximate multiplier is constructed as a network of logical gates whereas gates, interconnections etc. are characterized by unique properties (such as the signal propagation delay), potentially changing in time.

Fig. 4e depicts our model of the accurate circuit; here, it is a combinational circuit defined by a truth-table. After the model is initialized, it waits until the stimulus updates (*update?*). Then, it waits *dly* units of time and uses *f()* to produce the output for the stimulus and the truth-table. To model a more complex (e.g., sequential) system, the automaton can be simply extended to describe the corresponding behavior.

Finally, all blocks must be instantiated and interconnected via their interfaces, according to Fig. 2 (see 2).

Listing 2. Making block instantiation/interconnections based on Fig. 2+3

```
1 sgen=tStimuliGen(PI[0],...,PI[3],dg,covR);
2 ac=tAc(PI[0],...,PI[3],POac[0],...,POac[3],tblac,dac);
3 gap4=gateAp(4,PI[0],PI[2],POap[0],c[0],c[2],c[13]);
4 gap3=gateAp(3,PI[0],PI[3],12,c[0],c[3],c[12]);
5 gap2=gateAp(2,11,12,POap[1],c[11],c[12],c[14]);
6 gap1=gateAp(1,PI[1],PI[2],11,c[1],c[2],c[11]);
7 gap0=gateAp(0,PI[1],PI[3],POap[2],c[1],c[3],c[15]);
8 spi = syncPI();
9 echk = equivChk(de);
```

C. Formal Verification of AxC Systems

For this paper, we decided to present just few representative SMC results (Fig. 5, Fig. 6) we achieved so far. Each of them was produced from a SMC query specifying the property to be checked. Fig. 5 presents simulation results based on the query (Q1) `simulate [<=`t_{max}`;`n`]` $\{\phi\}$, where ϕ is a list of properties to be monitored in n simulation runs, each taking t_{max} units. It can be seen that our approach covers input/output jitters (a, b), evaluates/compares output values (b) and finally, evaluates parameters such as absolute/accumulated error/sum and coverage in various contexts (c, d).

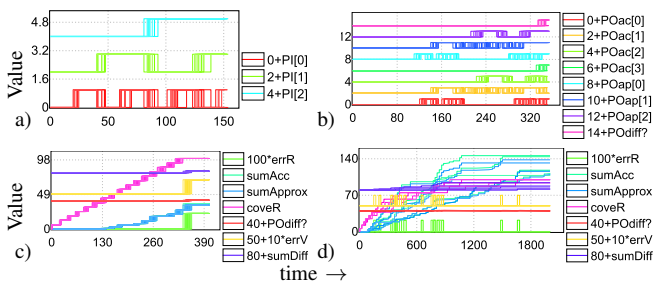

Fig. 5. Results from multiple (5–10) simulation runs of Q1 aiming to show the evolution of: a) jitters of values 0–7 at PI, b) PO values and difference, c,d) accumulated sums, errors etc. for two different application-specific contexts

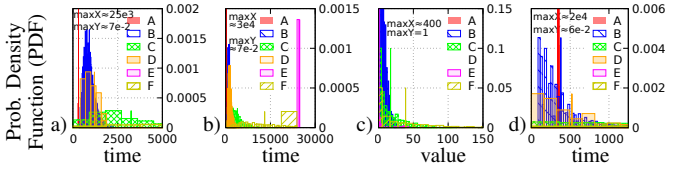

Fig. 6. Results of a) Q2, b) Q3, c) Q4, d) Q5 for the probability uncertainty (ε) set to 0.05 % and various application-specific scenarios (A–F). Visualization is incomplete – just cutouts of the area defined by $(x = 0, y = 0)$ to $(maxX, maxY)$ are visualized to show the most interesting results

Fig. 6 presents (cutouts of) results of the following queries: (Q2) `Pr[<=`t_{max}`] (<>coveR>87.5%)` for checking the probability that outputs for more than 87.5 % inputs will be found equivalent within the given time bound (t_{max}), (Q3) `E[<=`t_{max}`; n] (max:tcover)` for estimating the maximum amount of time for checking the equivalence during t_{max} when repeated n times, (Q4) `E[<=`t_{max}`; n] (max:sumDiff)` for estimating the maximum accumulated error of the approximate output during t_{max} when repeated n times, (Q5) `Pr[<=`t_{max}`] (<>errR > 5%)` for checking the probability that the accumulated error exceeds 5 % within t_{max}, where $t_{max} = 25e3$, $n = 1e3$.

From the scalability viewpoint, our results shows that the checking time grows linearly and inversely proportionally to the probability uncertainty (ε). For example, for $\varepsilon = .1\%$, the checking time ranges from 2 s to 286 s, depending on a query. The memory consumption is almost independent of ε, but (actually) it grows exponentially with the number of bits needed to represent the interface of a circuit (≈45 MB for a 2–8bit multiplier, growing exponentially to ≈110-1500 MB for a 10–15 bit multiplier).

IV. Conclusion

In this paper, we presented novel approach to modeling and verification of AxC systems regarding their application-specific contexts. Our experiments show that the STA/SMC approach presented in this paper has a potential to contribute to the area of AxC systems. The approach covers a wide range of systems and is able to formally evaluate and compare various static as well as dynamic properties of both accurate and approximate variants of systems.

Our future research plans regarding SMC of AxC systems will concentrate on more complex, sequential systems (e.g., DRAMs or CPUs) in real operating conditions (e.g., fault or aging/stress scenarios). Also, we plan to apply the STA/SMC instruments in the area of approximate control of systems.

References

[1] A. Chandrasekharan, D. Große, and R. Drechsler, *Design Automation Techniques for Approximation Circuits: Verification, Synthesis and Test*. Springer, 2019, ISBN 978-3-319-98964-8, DOI 10.1007/978-3-319-98965-5.

[2] A. Chandrasekharan, S. Eggersglüß, D. Große, and R. Drechsler, "Approximation-Aware Testing for Approximate Circuits," in *23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2018, pp. 239–244, DOI 10.1109/ASPDAC.2018.8297312.

[3] S. Reda and M. Shafique, Eds., *Approximate Circuits: Methodologies and CAD*. Cham: Springer International Publishing, 2019, DOI 10.1007/978-3-319-99322-5.

[4] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and Analysis of Circuits for Approximate Computing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2011, pp. 667–673, DOI 10.1109/ICCAD.2011.6105401.

[5] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," in *24th Internatioal Conference on VLSI Design*. Los Alamitos: IEEE, 2011, pp. 346–351, DOI 10.1109/VLSID.2011.51.

[6] X. Jiao, V. Camus, M. Cacciotti, Y. Jiang, C. Enz, and R. K. Gupta, "Combining Structural and Timing Errors in Overclocked Inexact Speculative Adders," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 482–487, DOI 10.23919/DATE.2017.7927037.

[7] V. Mrazek, Z. Vasicek, and L. Sekanina, "Design of Quality-Configurable Approximate Multipliers Suitable for Dynamic Environment," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Aug 2018, pp. 264–271, DOI 10.1109/AHS.2018.8541479.

[8] A. Chandrasekharan, M. Soeken, D. Große, and R. Drechsler, "Precise Error Determination of Approximated Components in Sequential Circuits with Model Checking," in *53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6, DOI 10.1145/2897937.2898069.

[9] Z. Vasicek, "Relaxed Equivalence Checking: a New Challenge in Logic Synthesis," in *20th IEEE International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 1–6, DOI 10.1109/DDECS.2017.7968435.

[10] A. Gebregiorgis and M. B. Tahoori, "Test Pattern Generation for Approximate Circuits Based on Boolean Satisfiability," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 1028–1033, DOI 10.23919/DATE.2019.8714898.

[11] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, "Testing Approximate Digital circuits: Challenges and Opportunities," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, March 2018, pp. 1–6, DOI 10.1109/LATW.2018.8349681.

[12] N. Bertrand, P. Bouyer, T. Brihaye, Q. Menet, C. Baier, M. Größer, and M. Jurdzinski, "Stochastic Timed Automata," *Logical Methods in Computer Science*, vol. 10, no. 4, 2014, DOI 10.2168/LMCS-10(4:6)2014.

[13] G. Agha and K. Palmskog, "A Survey of Statistical Model Checking," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, no. 1, pp. 6:1–6:39, Jan. 2018, DOI 10.1145/3158668.

[14] K. G. Larsen and A. Legay, "Statistical model checking past, present, and future," in *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, T. Margaria and B. Steffen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 135–142, DOI 10.1007/978-3-662-45231-8_10.

[15] J. H. Kim, A. Boudjadar, U. Nyman, M. Mikucionis, K. G. Larsen, and I. Lee, "Quantitative Schedulability Analysis of Continuous Probability Tasks in a Hierarchical Context," in *2015 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*, May 2015, pp. 91–100, DOI 10.1145/2737166.2737170.

[16] Y. Wang, A. Komuravelli, P. Zuliani, and E. M. Clarke, "Analog Circuit Verification by Statistical Model Checking," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*. IEEE, Jan 2011, pp. 1–6, DOI 10.1109/ASPDAC.2011.5722168.

[17] Y. Zhang, S. Sankaranarayanan, F. Somenzi, X. Chen, and E. Ábraham, "From Statistical Model Checking to Statistical Model Inference: Characterizing the Effect of Process variations in Analog Circuits," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 662–669, DOI 10.1109/ICCAD.2013.6691186.

[18] J. A. Kumar, S. N. Ahmadyan, and S. Vasudevan, "Efficient Statistical Model Checking of Hardware Circuits With Multiple Failure Regions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 6, pp. 945–958, June 2014, DOI 10.1109/TCAD.2014.2299957.

[19] G. Gielen, N. Xama, K. Ganesan, and S. Mitra, "Review of Methodologies for Pre- and Post-Silicon Analog Verification in Mixed-Signal SOCs," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 1006–1009, DOI 10.23919/DATE.2019.8714828.

[20] A. David, K. Larsen, A. Legay, M. Mikučionis, and D. Poulsen, "Uppaal SMC Tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015, DOI 10.1007/s10009-014-0361-y.