Functional Analysis of Circuits Under Timing Variations

Mehdi Dehbashi, Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
Görschwin Fey, Institute of Space Systems, German Aerospace Center, 28359 Bremen, Germany
Kaushik Roy, School of Electrical and Computer Eng., Purdue University, West Lafayette, IN 47907
Anand Raghunathan, School of Electrical and Computer Eng., Purdue University, West Lafayette, IN 47907

Abstract

This paper presents a methodology to model and analyze the functional behavior of logic circuits under timing variations. In the framework, first a *Time Accurate Model* (TAM) of the circuit is constructed. The TAM represents the behavior of the circuit in the functional domain under a discrete time model. Afterwards, *Variation Logic* is inserted to apply the timing variation delays. Moreover, the circuit TAM is enhanced by *Time Control* (TC) logic to model the circuit frequency. We apply the proposed methodology to analyze a circuit or an approximated circuit under timing variations as well as to analyze a circuit under timing-induced errors for approximate computing.

1 Introduction

As *Integrated Circuit* (IC) technology continues to scale down, variability is recognized to be a major challenge in analyzing the circuits. In this case, delay deviations are imposed by process variations such as uncertainty in the parameters of fabricated devices and interconnects, and by environmental variations such as temperature and voltage [3] [2] [13].

Recently, there is a range of works that considers timing analysis of circuits under variations. A survey of the works focusing on *Statistical Static Timing Analysis* (SSTA) is given in [3]. The statistically-critical paths under process variations are extracted by a bound-based method in [22]. The extracted paths have the highest probability to fail the timing constraint. The effects of process variations on the delays of logic gates and timing errors are analyzed in [2] [9] [13]. Timing error detection and correction has been proposed as an approach to bridge the gap between typical case and worst case design, by allowing circuits to operate without any margins [8] [19]. The work in [14] presents a framework to evaluate how microarchitectural techniques can trade off variation-induced errors for power and processor frequency.

On the other hand, in the recent years, significant progress in the areas such as approximate and probabilistic computing has been achieved. Much of the computations addressed in these areas focus on good enough or bounded results but not necessarily exact results [4]. In these techniques, the requirement of exact numerical or Boolean equivalence is relaxed to yield performance or energy efficiency [15] [5] [20].

An approximate implementation of a circuit does not exactly match the specification because of timing-induced errors or functional approximations [4]. Timing-induced errors can be produced by voltage over-scaling or overclocking.

This work has been funded in part by the German Research Foundation (DFG, grant no. FE 797/6-1).

Systematic synthesis of approximate circuits is exploited in [16] [17] [12] to reduce circuit area and delay as well as to increase yield. The work in [21] develops a logic optimization procedure that utilizes multi-V_t (threshold voltage) libraries to optimize a circuit for higher frequency and throughput under timing error detection and correction. The work in [10] uses a power-aware slack redistribution that shifts the timing slack of frequently-excercised, near-critical timing paths in a power- and area-efficient manner. The work in [11] presents an *Error-Resilient System Architecture* (ERSA) which combines unreliable cores with a small fraction of reliable processor cores for running system software, controlling application flow, and recovering from errors generated on unreliable cores. Scalable effort hardware design is proposed in [6] to identify mechanisms at each level of design abstraction (circuit, architecture, and algorithm) which can be used to vary the computational effort expended for generating the exact results. These scaling mechanisms are utilized to improve energy efficiency while maintaining an acceptable result.

A systematic methodology for the modeling and analysis of circuits for approximate computing is proposed in [20]. The methodology is utilized to analyze a circuit under timing-induced approximations as well as functional approximations using multiple metrics. However, timing variations are not considered. Since variations can significantly perturb the timing of various paths in a circuit, it is natural to expect that they will also significantly impact which paths fail under timing-induced approximations, and therefore the functional behavior of approximate circuits. Therefore, it is essential to consider the impact of variations during the analysis of approximate circuits.

In this paper, we propose a unified framework that can be used to analyze *how a circuit behaves under timing variations*, *how a circuit behaves under timing-induced approximations*, and *how an approximate circuit beha-*

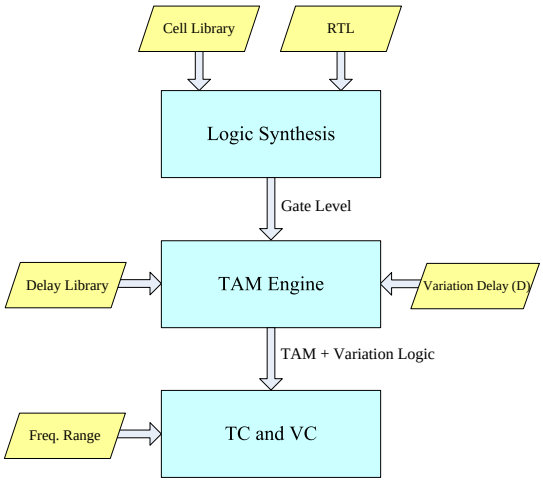

Fig. 1 Overview of proposed methodology

ves under timing variations. By considering the functional domain, our approach is complementary to SSTA. In the approach, we first convert the timing behavior of a circuit into the functional domain according to a time unit model. The newly constructed circuit is called the *Time Accurate Model* (TAM) of the circuit. The TAM represents the functional behavior of the circuit with respect to the circuit delay and a precision of an arbitrarily fine-grained but discrete time unit. Afterwards, *Variation Logic* is inserted in the TAM to apply the timing variation delays. The variation logic is applied at each gate. The cumulative delay normalized to a time unit may affect the correct behavior of the circuit. The behavior of the variation logic is determined by *Variation Control* (VC) inputs. Moreover, the circuit is also enhanced by *Time Control* (TC) logic. TC is a flexible logic which controls the frequency at the inputs. TC models timing-induced approximations like overclocking. We use *Boolean Satisfiability* (SAT) as an underlying reasoning engine to analyze the circuits.

The rest of this paper is organized as follows. Section 2 introduces preliminary information. Section 3 describes our approach to construct the TAM, to insert variation logic, and to enhance a circuit by TC and VC. Then, experimental results on arithmetic units are presented in Section 4. Section 5 concludes the work.

2 Preliminaries

2.1 Timing Parameters

We consider a *time unit* which is an arbitrarily fine-grained but discrete unit of delay. The delays of gates and interconnects are assumed to be an integer multiple of one time unit. In a circuit where the shortest path delay is D_s time units, and the longest path delay is D_l time units, the current output O_t depends on the inputs of I_{t-D_s}, I_{t-D_s-1}, ..., I_{t-D_l}. Indices denote the times of input with a step of one time unit. Each index is also called *time step*.

A *clock period* is defined as T time units. In synchronous circuits, the input to the combinational logic changes only once every clock period. If the circuit has a clock period of T, the output at time step t depends on the inputs of the following clock periods:

$$\forall i, a \leq i \leq b : [I_{t-iT-1}, \ldots, I_{t-(i+1)T}] \qquad (1)$$

$$a = \lceil D_s/T \rceil - 1\,, \quad b = \lceil D_l/T \rceil - 1$$

This formula partitions the times of input according to the clock period T such that in each clock period, the inputs are assumed to be fixed. For example, when $D_s = 1$, $D_l = 5$, $T = 5$, O_t depends on input values from time steps that fall within the previous clock period $[I_{t-1}, I_{t-2}, I_{t-3}, I_{t-4}, I_{t-5}]$, and in this clock period, the inputs do not change: $I_{t-1} = I_{t-2} = I_{t-3} = I_{t-4} = I_{t-5}$. When $T = 2$, O_t depends on input values from time steps that fall within the following clock periods: $[I_{t-1}, I_{t-2}]$, $[I_{t-3}, I_{t-4}]$, $[I_{t-5}, I_{t-6}]$. Overall, when $T < D_l$, the clock is overscaled, i.e., the current output depends on the inputs of multiple previous clock periods. This case is also called overclocking. We note that, for our purpose, voltage overscaling has the same effect as overclocking, since the delays of the gates will be scaled up based on the lower voltage, while the clock period remains the same.

When the clock is overscaled, the longer paths fail because the input does not have enough time to propagate to the output. In this case, the current output result depends not only on the input of one previous clock period but also on the inputs of multiple previous clock periods. The older inputs (the inputs of the clock periods more distant from current time t) influence the output through longer paths and the newer inputs (the inputs of the clock periods closer to the current time t) affect the output through shorter paths.

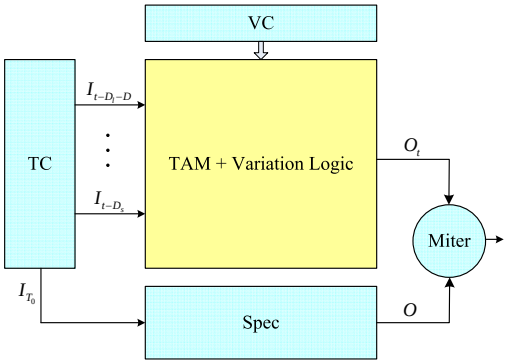

Fig. 2 Overall model created by the framework

3 Methodology

The fine-grained timing behavior of a circuit is converted into the functional domain. Having the behavior of a circuit according to a fine-grained time unit allows us to utilize it for modeling races, glitches, etc. The fine-grained timing model is also utilized to control and to

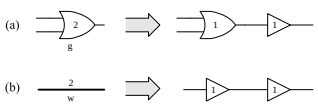

Fig. 3 Converting original gates and wires to untimed gates and wires

```
1  function TAM(In : untimed circuit, Out : TAM circuit)
2  time = 0
3  SIG = PO
4  while SIG ≠ ∅ do
5  {
6     SIG_temp = ∅
7     foreach sig ∈ SIG do
8     {
9        gate = predecessor(sig)
10       copy(gate, i_{t-time-1}, o_{t-time})
11
12       foreach input ∈ I(gate) do
13          if input ∉ SIG_temp and input ∉ PI then
14             SIG_temp = SIG_temp ∪ input
15    }
16    SIG = SIG_temp
17    time++
18 }
19 end function
```

Fig. 5 Creation of Time Accurate Model

modify the frequency of a circuit during our analysis. This fine-grained timing model of the circuit is called *Time Accurate Model* (TAM). When the timing behavior of a circuit is available in the functional domain, formal verification methods can comprehensively analyze the timing effects of the circuit.

Figure 1 shows the overview of our approach. Firstly, the gate level circuit (synthesized netlist) is generated according to a cell library. Afterwards, the TAM engine creates the time accurate model of the circuit which models the fine-grained timing behavior of a circuit in the functional domain. The TAM is generated according to a fine-grained time unit. The time unit specifies the granularity of analysis and hence controls the accuracy of frequencies and our evaluations. The delays of all gates in the circuit are normalized according to the time unit. Variation logic is inserted in the TAM according to the maximum delay (D) induced under timing variations.

In the final step, *Time Control* (TC) and *Variation Control* (VC) are added. TC includes some constraints on the inputs to control the clock period according to Formula 1. VC is a constraint to control the delays induced under timing variations. By this, the model can be used to analyze a circuit with respect to different frequencies.

Figure 2 shows the overall model created by our framework for the analysis. The three main components of the model are: TAM and variation logic, TC, VC. Also the model includes two side components: *spec* and *miter*. These two components serve different tasks in different applications. Spec can be a golden specification or golden properties of the ideal circuit behavior. Miter measures the deviation of the circuit output result against its specification. Here, we use a SAT solver as an underlying engine to measure the deviations.

In the following, Section 3.1 describes how the TAM of a circuit and its variation logic are created. The TC and VC components are explained in Section 3.2.

3.1 TAM Engine

The TAM algorithm was inspired from the algorithm presented in [20]. However, there are key differences necessitated by the need to handle variations - we consider a fine-grained time unit and also the variation logic is added into the model.

The underlying idea is that a signal s_t represents a signal s of the original circuit at time step t. In the TAM engine, first the delays of the original gates and wires are converted into the functional domain according to the chosen time unit. An *original gate* g with delay n is converted to n successive *untimed gates*: $(g, Buf_{n-1}, \ldots, Buf_1)$ (Figure 3(a)). The equivalent untimed gates show the behavior of the original gate with an accuracy of one time unit. Also a wire w with delay n is converted to n successive buffers: $(Buf_n, \ldots, Buf_1)$ (Figure 3(b)). The circuit with the untimed gates and wires is called the *untimed circuit*.

After untiming the original gates and wires, the second step of the TAM engine starts to convert the timing behavior of the overall circuit into the functional domain. Figure 5 describes the algorithm in pseudo code. The input data of this algorithm is an untimed circuit.

The algorithm starts from *Primary Outputs* (PO) and traverses the untimed circuit graph backward (line 3). By considering the precision of one time unit, the output of an untimed gate depends on its corresponding inputs one time step ago. These inputs are simply given by the predecessor node of the output in the untimed circuit graph (line 9). Given the output and its driving inputs, a new gate is created in which the output and the inputs have the timing difference of one time unit (line 10). The newly created gates are called *TAM gates* and constitute a new circuit called the *TAM circuit*. The inputs of the current untimed gates are collected in the set SIG_temp to be used for the next backward traversal step (lines 12-14). If the input is a *Primary Input* (PI) or already exists in the set SIG_temp (fanout case), it will not be added to the set SIG_temp (line 13).

We explain the algorithm by the example original circuit of Figure 4(a). For sake of simplicity, the delays of original gates are considered to be 1. Therefore, the untimed circuit (Figure 4(b)) is the same as the original circuit. In the first step, the algorithm starts from primary output e and copies the OR gate: $OR(c_{t-1},\ d_{t-1},\ e_t)$. The copied gates (TAM gates) are shown in Figure 4(c). The dotted lines visualize the time steps. There, $SIG\_temp = \{c, d\}$ is created. The second step is backward traversal of the untimed circuit from c and d. The NOT and AND gates are copied: $NOT(b_{t-2},\ c_{t-1})$, $AND(a_{t-2},\ c_{t-2},\ d_{t-1})$. Having $SIG\_temp = \{c\}$, the third step starts and copies the NOT gate: $NOT(b_{t-3},\ c_{t-2})$. As explained in Section 2.1, O_t depends on the inputs of $I_{t-D_s}, I_{t-D_s-1}, \ldots, I_{t-D_l}$ where $D_s = 2$ and $D_l = 3$ in this example.

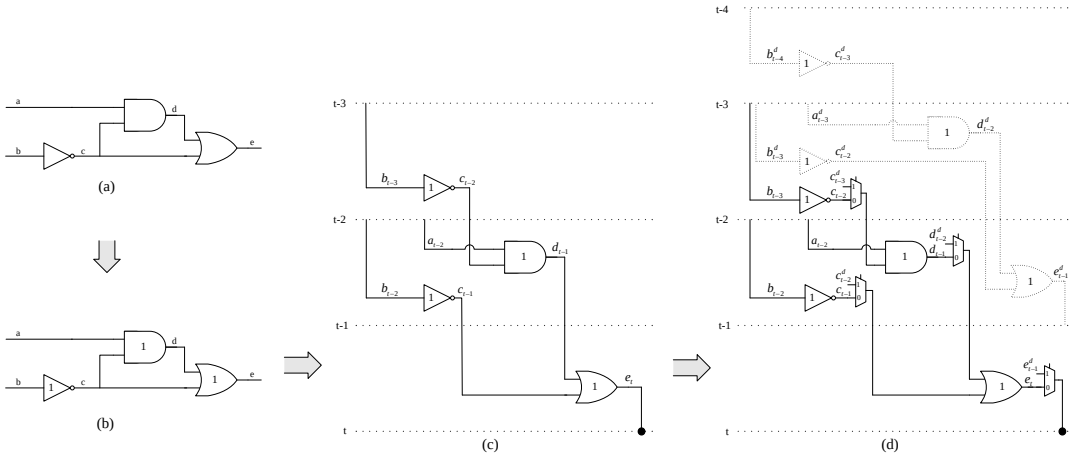

Fig. 4 (a) Original circuit and original gates (b) Untimed circuit and untimed gates (c) TAM circuit and TAM gates (d) TAM and variation logic

```
1  function TAM + VariationLogic()
2  time = 0
3  SIG = PO
4  while SIG ≠ ∅ do
5  {
6     SIG_temp = ∅
7     foreach sig ∈ SIG do
8     {
9        gate = predecessor(sig)
10       for 0 ≤ d ≤ D do
11          copy(gate_d, i_{t-time-1-d}, o_{t-time-d})
12
13       if sig is Original_Gate_Output then
14          for 0 ≤ d < D do
15             Insert_Variation_Mux_d(in : gate_d, ··· , gate_D)
16
17       foreach input ∈ I(gate) do
18          if input ∉ SIG_temp and input ∉ PI then
19             SIG_temp = SIG_temp ∪ input
20    }
21    SIG = SIG_temp
22    time + +
23 }
24 end function
```

Fig. 6 TAM + Variation Logic

For investigating the behavior of a circuit under timing variations, the *Variation Logic* is inserted in the TAM. The variation logic skews a signal by a delay of d time units. The delay is activated for each gate independently by activating the select line of the variation multiplexer of the corresponding gate. Therefore, in addition to the value of a signal in the corresponding time, the value of the signal d time steps ago is also needed. Figure 6 describes this algorithm which extends the algorithm of Figure 5.

In lines 10-11 of Figure 6, when $d = 0$, the gate related to the corresponding time is copied. This gate is called TAM gate. After that, the behavior of the gate up to d time steps ago is also copied (lines 10-11, when $d \neq 0$). These gates are called *delay gates*. Parameter d is limited by a maximum variation delay D specified by the user. Given a TAM gate ($gate_0$) and its corresponding delay gates ($gate_1, \cdots, gate_D$), a variation multiplexer is added (lines 14-15). The variation multiplexer can skew the normal behavior of a gate by a delay up to D time units. The timing variation of each original gate is modeled by its corresponding variation multiplexers (line 13). Delay skews can also be applied on the delay gates such that the total delay is less equal than D (lines 14-15, when $d \neq 0$).

Figure 4(d) shows the circuit generated by the algorithm where $D = 1$. The dashed gates are the delay gates. For each original gate, one variation multiplexer is inserted at the output of its corresponding TAM gate. A variation multiplexer either selects the normal behavior of a signal or the signal value one time step ago.

The size of the TAM depends on the topology and the delay of reconverging paths of the original circuit. In the worst case, the size of the TAM may be exponentially larger than the original circuit. However, in our experiments the increase in size was moderate.

3.2 Time Control (TC) and Variation Control (VC)

Frequency and clock period are denoted by f and T, respectively ($f = 1/T$). The task of the TC is applying a clock period T on the inputs with the accuracy of one time unit. According to Formula 1, inputs are constrained to have a constant value throughout each clock period.

The task of the VC is controlling the select lines of the variation multiplexers. The VC applies the maximum variation delay by the following constraint:

$$\sum_{i=1}^{n} s_i \leq D \tag{2}$$

where s_i denotes the integer value of the select lines related to one variation multiplexer.

Alternative constraints can also be added to model more complex variations. For example to apply and to control block-based variation models [3], variations of each region can be controlled by a constraint. In a hierarchical manner, the variations of different regions in each level of hierarchy (like quadtree partitioning [1]) can be correlated by having additional constraints in each level.

Our approach can also be extended to consider clock skew. This can be done by adding some units for the delay of the clock network and the sequential elements.

Our model is a conservative model in the sense that it overapproximates delays induced under timing variations. The model allows delays to be activated independently in every location of a circuit. The model can be used to evaluate the worst case of a circuit functional deviation under timing variations. Here, a SAT solver is used to compute the maximum or the worst-case error under induced variations.

4 Experimental Results

We apply the proposed approach to analyze arithmetic circuits under timing variations. The experiments are carried out on a Quad-Core AMD Phenom(tm) II X4 965 Processor (3.4 GHz, 8 GB main memory) running Linux. The techniques described in this paper are implemented using C++ in the WoLFram environment [18]. MiniSAT is used as underlying SAT solver [7].

We evaluate Ripple Carry Adder (RCA) and Carry Look-ahead Adder (CLA) benchmarks. For the experiments, the delays of NOT and BUF gates are considered to be 1. AND and OR gates with n inputs have a delay of $\lceil n/2 \rceil + 1$. The delay of XOR gates is 5.

As Figure 2 showed, our framework includes two side components: *spec* and *miter*. In the experiments, the original circuit is considered as a specification. The inputs of the most recent clock period are applied to the specification. For arithmetic circuits, we use a miter on the outputs of the specification and the TAM to measure the output deviation as the numerical difference. This miter is an integer subtractor followed by a comparison operation. In order to measure the *maximum positive error*, the miter subtracts the specification output (O) from the TAM output (O_t): ($O_t - O \geq L$). To compute the *maximum negative error*, the following miter is used: ($O - O_t \geq L$). Our approach increases the parameter L by a step k to calculate the maximum error ($L = L + k$).

The average run times required to compute the maximum error at one frequency with step $k = 1$ for 4-bit RCA and 4-bit CLA are 7 and 9 seconds, respectively. The number of gates in the untimed circuit and the TAM for 4-bit RCA and 4-bit CLA are (64, 150) and (89, 167), respectively.

Figure 7 and Figure 8 show the maximum positive and negative error computed for a 4-bit RCA and a 4-bit CLA under overclocking and in the presence of timing variations. The X-axis indicates the frequency f as the inverse of the clock period T ($f = 1/T$). The minimum frequency specified by the X-axis is related to the maximum delay of the circuit considering the maximum delay variation D. The Y-axis indicates the maximum error as a result of the computed error divided by the maximum output value. When $D = 0$, no timing variation is activated. In this case, the diagram shows the maximum error caused by clock overscaling. While the frequency increases, the maximum error sometimes decreases. This is because the failing output bits

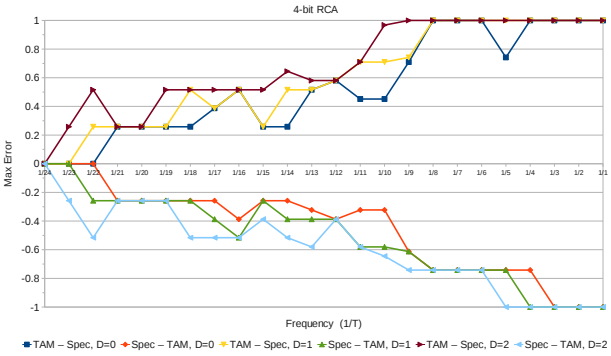

Fig. 7 Maximum error for 4-bit RCA

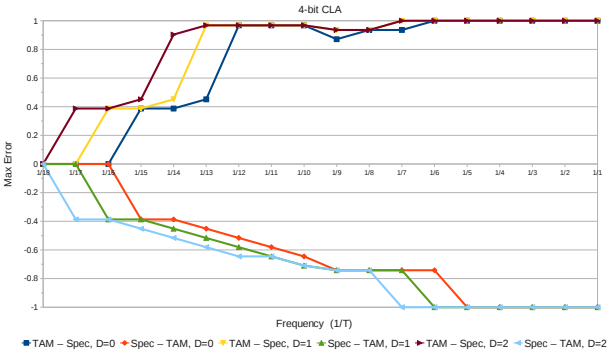

Fig. 8 Maximum error for 4-bit CLA

are functionally correlated as they may share certain paths. Also, the maximum positive error and the maximum negative error may have different values at the same frequency because overscaling modifies the functionality and induces asymmetry in the circuit. When the frequency increases for the CLA, the maximum error increases faster (Figure 8). This is due to the fact that more MSBs have a closer delay. Thus, multiple MSBs may fail together by overclocking. Therefore, the CLA is more sensitive to overclocking. But for the RCA, the maximum error increases slower than for the CLA because the MSBs have different delays. Therefore, they fail gradually along overclocking. In Figure 7, the delay of the circuit is 22 time units. Therefore when $D = 0$ and $f = 1/22$, there is no error at the output of the circuit. When the clock is overscaled, for example the frequency increases from $f = 1/22$ to $f = 1/21$, then the maximum error 0.26 is observed on the outputs.

When $D = 1$, the constraint of Formula 2 is applied. In this case, the maximum variation delay is considered to be 1. The diagram shows the maximum error computed along the frequency axis while the variation delay is activated. As the diagram shows, the maximum error starts earlier. Also the maximum error has a different progress in comparison to a non-varied circuit. In Figure 7, when $D = 1$, at the frequency $f = 1/22$, an error is observed on outputs. The sum_3 bit fails as the delay of the sum_3 critical path is 22 time units. When $D = 2$, at the frequency $f = 1/22$, the $cout$ bit fails as

the delay of $cout$ is 21 time units. Therefore the maximum error observed for $D = 2$ is more than $D = 1$ at the frequency $f = 1/22$.

5 Conclusion

This paper introduced a methodology to model and to analyze the functional behavior of circuits under timing variations. The framework includes three main components: *Time Accurate Model* (TAM) and variation logic, *Time Control* (TC), and *Variation Control* (VC). Our framework is utilized to analyze a circuit under timing variations, a circuit under approximation (called approximated circuit), as well as an approximated circuit under timing variations.

6 Literatur

[1] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Int'l Conf. on CAD*, pages 900–907, 2003.

[2] M. Alioto, G. Palumbo, and M. Pennisi. Understanding the effect of process variations on the delay of static and domino logic. *IEEE Trans. VLSI Syst*, 18(5):697–710, 2010.

[3] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. Statistical timing analysis: From basic principles to state of the art. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(4):589–607, 2008.

[4] M. A. Breuer. Hardware that produces bounded rather than exact results. In *Design Automation Conf.*, pages 871–876, 2010.

[5] S. T. Chakradhar and A. Raghunathan. Best-effort computing: re-thinking parallel software and hardware. In *Design Automation Conf.*, pages 865–870, 2010.

[6] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. Scalable effort hardware design: exploiting algorithmic resilience for energy efficiency. In *Design Automation Conf.*, pages 555–560, 2010.

[7] N. Eén and N. Sörensson. An extensible SAT solver. In *SAT 2003*, volume 2919 of *LNCS*, pages 502–518, 2004.

[8] D. Ernst, N. S. Kim, S. Das, S. Pant, R. R. Rao, T. Pham, C. H. Ziesler, D. Blaauw, T. M. Austin, K. Flautner, and T. N. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *Int'l Symposium on Microarchitecture*, pages 7–18, 2003.

[9] M. Gao, Z. Ye, Y. Peng, Y. Wang, and Z. Yu. A comprehensive model for gate delay under process variation and different driving and loading conditions. In *Int'l Symp. on Quality Electronic Design*, pages 406–412, 2010.

[10] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori. Slack redistribution for graceful degradation under voltage overscaling. In *ASP Design Automation Conf.*, pages 825–831, 2010.

[11] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra. ERSA: Error resilient system architecture for probabilistic applications. In *Design, Automation and Test in Europe*, pages 1560–1565, 2010.

[12] A. Lingamneni, C. Enz, J.-L. Nagel, K. Palem, and C. Piguet. Energy parsimonious circuit design through probabilistic pruning. In *Design, Automation and Test in Europe*, pages 764–769, 2011.

[13] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. VARIUS: A model of process variation and resulting timing errors for microarchitects. *IEEE Trans. Semiconductor Manufacturing*, 21(1):3–13, 2008.

[14] S. R. Sarangi, B. Greskamp, A. Tiwari, and J. Torrellas. EVAL: Utilizing processors with variation-induced timing errors. In *Int'l Symposium on Microarchitecture*, pages 423–434, 2008.

[15] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones. Stochastic computation. In *Design Automation Conf.*, pages 859–864, 2010.

[16] D. Shin and S. K. Gupta. Approximate logic synthesis for error tolerant applications. In *Design, Automation and Test in Europe*, pages 957–960, 2010.

[17] D. Shin and S. K. Gupta. A new circuit simplification method for error tolerant applications. In *Design, Automation and Test in Europe*, pages 1566–1571, 2011.

[18] A. Sülflow, U. Kühne, G. Fey, D. Große, and R. Drechsler. WoLFram – a word level framework for formal verification. In *IEEE/IFIP Int'l Symposium on Rapid System Prototyping*, pages 11–17, 2009.

[19] J. Tschanz, K. A. Bowman, C. Wilkerson, S.-L. Lu, and T. Karnik. Resilient circuits - enabling energy-efficient performance and reliability. In *Int'l Conf. on CAD*, pages 71–73, 2009.

[20] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan. MACACO: Modeling and analysis of circuits for approximate computing. In *Int'l Conf. on CAD*, 2011.

[21] L. Wan and D. Chen. Dynatune: Circuit-level optimization for timing speculation considering dynamic path behavior. In *Int'l Conf. on CAD*, pages 172–179, 2009.

[22] L. Xie and A. Davoodi. Bound-based statistically-critical path extraction under process variations. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 30(1):59–71, 2011.