Error Rate Estimation of a Design Implemented in an FPGA based on the Operating Conditions

Marc Alexandre Kacou*†, Fakhreddine Ghaffari†, Olivier Romain†, Bruno Condamin*
*Valeo Siemens eAutomotive, Cergy-Pontoise, France
{marc.kacou, bruno.condamin}.jv@valeo-siemens.com
†ETIS, UMR 8051,Univ Paris Seine, Univ Cergy-Pontoise, ENSEA, CNRS, France
{assi-marc.kacou, fakhreddine.ghaffari, olivier.romain}@ensea.fr

Abstract

This paper presents a framework to evaluate the error rate of a design implemented in an FPGA, depending on the operating conditions. The effects of the operating temperature as well as voltage variation are taken into account. For example, in applications such as the electric vehicle motor control, the non-negligible electromagnetic field can result in FPGA supply voltage variation. It is shown that successfully passing the standard static timing analysis for such an application is not always sufficient to ensure the reliability of the application. Thus, it is needed to assess the operation of the design in the real operating conditions, to ensure the correct execution of the application. The error rate results obtained by measurements show good correlation with the ones obtained from the proposed framework.

1. Introduction

FPGAs are more and more used in the industry thanks to several advantages such as tasks parallelization and reprogrammability. Hence, they are used in several industrial domains such as aerospace and automotive. In contrast, they are very sensitive to their operating conditions which can induce runtime errors. When they are used to implement safety-critical applications running in an aggressive environment, it is of paramount importance to ensure their correct execution.

Internal FPGA parameters such as the propagation delay are directly dependent to the junction temperature as well as the supply voltage [1]. Thus, once the operating conditions change, the propagation delay also changes. The static timing analysis (STA) of the FPGA design is a mandatory step to ensure that a design will run correctly on the FPGA at the expected frequency. STA tools are generally corner-based and guarantee the correct execution only for some operating conditions. For example, regarding the temperature, automotive grade FPGAs are guaranteed for the range from -40°C to +125°C while commercial grade are insured in the range from 0°C to 85°C. Regarding the voltage, STA is generally performed for the device recommended voltage range, generally $\pm 5\%$ around the nominal voltage [2] [3] [4] [5] [6]. With technology downscaling, it is now in the range of tens of millivolts.

It has been presented in [7] that an electromagnetic field can induce a parasitic voltage onto the FPGA power supply lines and cause the FPGA supply voltage to vary. As previously explained, that voltage variation results in a variation of the propagation delay inside the FPGA, inversely to the supply voltage. Hence, when the induced voltage makes the FPGA voltage outside the range covered by STA tools, successfully passing STA doesn't guarantee the correct execution of the design. The variation of the circuit propagation delay can thus induce under certain conditions, timing errors in the application which could then lead to the functional failure of the application.

Timing errors can happen at the synchronization points in the design, generally D flip-flops (DFF). A lot of techniques exist to detect and correct timing errors in a given design [8] [9] [10] [11]. However, they require to monitor either all the DFFs of a design or some DFFs at the end of critical paths. In the case all DFFs have to be monitored, the detection/correction scheme could add much overhead depending on the number of DFF in the design. Regarding the monitoring of critical paths, it could be inefficient as, depending on the application, some of those paths should be never or less sensitized than other paths or simply not functionally critical for that application. In addition, it is not explained how many critical paths should be monitored or why the selected number of critical paths has been chosen.

In this paper, we address the selection of the DFFs to be monitored by presenting an error rate estimation method. It evaluates the error rate of each endpoint in a design for a given operating condition. The idea behind is to allow the identification of the really critical DFFs of the application. In fact, depending on the error rate

978-1-5386-3299-4/17/$31.00

of each DFF and their criticality with respect to the functionality of the application, it is possible to determine a set of application-critical DFFs, allowing the designers to only focus on those DFFs. This reduces the detection/correction overhead.

Although the method tackles the case of environmentally induced voltage and temperature variation, it can also be applied to any problem linked with voltage and temperature variation such as voltage scaling [12]. It can also be used to study the effect of a drift on the FPGA voltage regulator output over time for safety calculations. For voltage scaling, knowing in advance the error rate of an application for a given operating condition could help reducing the overhead of the detection/correction scheme as aforementioned. Regarding the drift of the FPGA regulator voltage over time, for a given assumed drift, the method can be used to estimate the associated error rate of the design endpoints, thus analyzing its effects on the implemented design.

The rest of the paper is organized as follow : Section 2 presents in details, the proposed method to estimate the error rate of an endpoint. In Section 3, the method is used to estimate the error rate of several FPGA designs and the results are compared with the ones obtained from measurements. Finally, section 4 concludes the paper.

2. Error Rate Quantification

To evaluate the error rate of each endpoint of an FPGA design, that design is decomposed into logic cones. A logic cone, presented in Fig.1, is a single-output synchronous sub-circuit concentrating a number of inputs through a logical function f(I), whose output is sampled by a clock signal. Throughout the rest of this paper, the inputs of a logic cone will be noted as the source points of that logic cone and its output will be referred to as the endpoint of that logic cone. Each FPGA synchronous design can be decomposed into a set of logic cones. The evaluation of the error rate of an endpoint boils down to the analysis of the logic cone associated to that endpoint. Hence, the analysis of an entire FPGA design is equivalent to the analysis of each logic cone of this design.

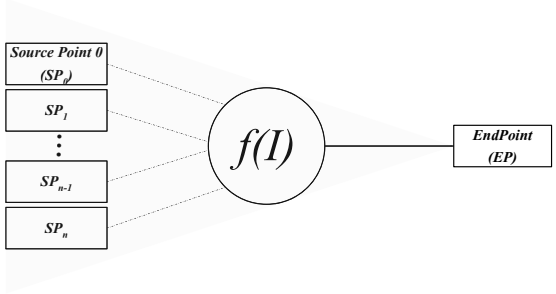


Figure 1: Graphical representation of a logic cone.

For a timing error to occur, three requirements are to be met :

1) A logic cone should have at least one faulty path;
2) A transition on the input of a faulty path should lead to a transition on the output of the function;
3) In addition, the sampled value should be different from the expected value.

From these requirements, we identify three steps for our proposed method.

2.1. Requirement 1 : Determination of the faulty paths

Requirement 1 is met when the delay of a path is greater than its slack. Designers perform STA to ensure that all circuit paths are correctly sampled at the required frequency before the design is implemented. But as previously explained, standard FPGA STA tools perform corner-based analysis, limiting the set of covered operating conditions. Hence, the correct execution of the design is only guaranteed for a maximum and minimum temperature as well as for a maximum and minimum voltage. When the operating conditions could be outside these ranges, whether intentionally by performing voltage scaling, or unintentionally by the effect of an electromagnetic field, the correct execution of the design is no more guaranteed and timing errors could occur.

To determine the faulty paths, a variation model noted $\Delta T_p(V,T)$ is required. It links the added propagation delay inside the FPGA to a given voltage and temperature. Such models have been studied in [7] [13]. From this model, it is possible to determine the associated faulty paths for a given design and a given operating condition. A path is considered faulty as soon as the added delay obtained from the aforementioned variation model is greater than the slack of that path. In [7], the presented methodology allows to determine the induced voltage in function of the magnitude of a magnetic field. Then, from the variation model, once can associate a given magnetic field level to its corresponding added delay for a given FPGA. The same methodology can be adapted to associate any voltage variation resulting from an electromagnetic field coupling or voltage scaling, to its corresponding delay variation, thus identifying the faulty paths if they exist.

Hence, from the first step, a set of faulty paths is obtained for each logic cone for a given operating condition. It should be noted that a circuit with no faulty path at a given temperature and voltage won't face timing error and thus is guaranteed fault-free for that operating condition. From that, it is possible to establish a fault-free region even outside the corners of

standard STA tools, representing another contribution of our method.

2.2. Requirement 2 : Identification of input combinations responsible of output transition

From the first step, a set of faulty paths associated to an operating condition has been identified. The requirement 2 can only be evaluated when a logic cone has at least one faulty path. Thus, the analysis presented hereafter is only done for such logic cones of the design.

For requirement 2 to be satisfied, a transition on the input of a faulty path should lead to a transition on the output of the function. Identifying the input combinations satisfying this requirement is equivalent to determine for which input combinations, the logic cone function is directly dependent on the input of the faulty path. This is obtained by calculating the Boolean difference [14] [10] of the function with respect to the input of the faulty path.

The Boolean difference of a function $f(I)$ with respect to one of its input I_j is given by [14]:

$$\frac{\partial f(I)}{\partial I_j} = f(I_0, ..., I_j = 1, ..., I_n) \oplus f(I_0, ..., I_j = 0, ..., I_n) \quad (1)$$

where $\oplus$ denotes the exclusive-or Boolean operator.

The result of the Boolean difference is independent of the considered input I_j. It gives a set of combinations representing the states the other inputs should have for the function to be directly dependent on input I_j. When a combination is evaluated to 1, a transition from either 0→1 or 1→0 on the considered input I_j results in a transition on the output of the function, otherwise, the transition doesn't affect the output of the function for that combination. Those combinations evaluated to 1 will be referred to as the faulty input combinations.

Equation 1 is used to identify the faulty input combinations when the logic cone has a single faulty path. In the case of a logic cone with multiple faulty paths, the faulty input combinations of a function f(I) with respect to the subset of inputs I_S representing the source points of each faulty path, are given by [14]:

$$\frac{\partial f(I)}{\partial I_S} = \frac{\partial f(I)}{\partial(\prod_{j \in S} I_j)} + \sum_{j \in S} \frac{\partial f(I)}{\partial I_j} \quad (2)$$

where the second term represents the individual application of equation 1 to each input of the subset. The first term represents the vertical Boolean difference of the function with respect to all inputs of the subset collectively. It is given, when considering a subset of m inputs, by [10] :

$$\frac{\partial f(I)}{\partial(\prod_{j \in S} I_j)} = \begin{cases} f(I_{S0}, I_{S1}, ..., I_{Sm-1}) \oplus f(\overline{I_{S0}}, \overline{I_{S1}}, ..., \overline{I_{Sm-1}}) \\ f(\overline{I_{S0}}, I_{S1}, ..., I_{Sm-1}) \oplus f(I_{S0}, \overline{I_{S1}}, ..., \overline{I_{Sm-1}}) \\ \cdot \\ \cdot \\ \cdot \\ f(I_{S0}, I_{S1} ..., \overline{I_{Sm-1}}) \oplus f(\overline{I_{S0}}, \overline{I_{S1}}, ..., I_{Sm-1}) \end{cases}$$

Each combination obtained from the above equations has the same properties as the ones from equation 1.

As an example, let's consider the circuit presented in Fig.2 and its associated truth table. Assuming the path P1 from A to S is the only faulty path, the combinations leading to a transition on the output of the function are given by the equation 1 as follow :

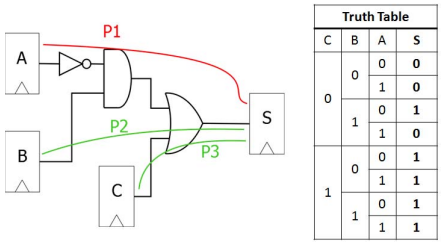
$$\begin{aligned} \frac{\partial f(A,B,C)}{\partial A} &= f(A=0,B,C) \oplus f(A=1,B,C) \\ &= \overline{C}.B \end{aligned} \quad (3)$$

It means that, if the path from A to S is a faulty path, each time input C is low while input B is high and a transition from either 0→1 or 1→0 occurs on input A, that transition will be propagated to the output. These combinations will be noted LHR and LHF where L stands for low, H for high, R for rise and F for fall.

Hence, from this step, the faulty input combinations are obtained for each logic cone containing at least one faulty path.

2.3. Requirement 3 : Calculation of the occurrence probability

After obtaining the different faulty input combinations, it is now possible to estimate the error rate of the logic cone endpoint by calculating the occurrence probability of these combinations. For that, the signal probabilities of the source points of the logic cones are used. The signal probabilities of an input are the probabilities for that input to be at logic level low or high. They are noted respectively, for a given input I_j, $P_L(I_j)$ and $P_H(I_j)$. From these signal probabili-



Truth Table			
C	B	A	S
0	0	0	0
		1	0
	1	0	1
		1	0
1	0	0	1
		1	1
	1	0	1
		1	1

Figure 2: An example of circuit with its truth table.

ties, the following four transition probabilities can be derived :

$$P_0(I_j) = P_L(I_j) * P_L(I_j) \quad (4)$$
$$P_R(I_j) = P_L(I_j) * P_H(I_j) \quad (5)$$
$$P_F(I_j) = P_H(I_j) * P_L(I_j) \quad (6)$$
$$P_1(I_j) = P_H(I_j) * P_H(I_j) \quad (7)$$

where $P_0(I_j)$ (resp. $P_1(I_j)$) is the probability associated to a transition 0→0 (resp. 1→1) and $P_R(I_j)$ (resp. $P_F(I_j)$) is the probability associated to a transition 0→1 (resp. 1→0).

The input combinations need to be expanded to input transitions in order to calculate the error rate using the transition probabilities. Considering once again, the example circuit in Fig. 2, it has been determined from step 2 that when path A is faulty, a transition on its input propagates to the output for the combinations LHR and LHF. For C to be at logic level low, it can be the result of a transition from 0→0 or 1→0. Likewise, for B to be at logic level high, it can be the result of a transition from 1→1 or 0→1. Thus, the set of input combinations LHR, LHF can be expanded to the set of input transitions $01R, 0RR, F1R, FRR, 01F, 0RF, F1F, FRF$.

It should be noted that not all the obtained inputs transition will lead to a transition on the output. This can be seen when analyzing the truth table for the transitions $0RR$, $F1F$ and FRF. Thus, the input transitions can be separated into two groups, one group for the ones leading to a transition on the output and a second one for the others. The transitions from the first group are the one for which each occurrence will lead certainly to a timing error. We will refer to these transitions as the "certain" input transitions. In contrast, the transitions from the second group will only cause timing errors in the case the implementation of the function lead to the occurrence of glitches in the function. A glitch can occur due to the difference in the arrival times of the inputs to the logic gates. These transitions will be referred to as the "likely" input transitions.

From this observation, the method will give an upper and a lower bound of the error rate for a given operating condition. The lower bound is given by the occurrence probability of the "certain" input transitions while we assume a pessimistic approach for the upper bound by calculating the occurrence probability of both the "certain" and "likely" input transitions.

2.4. Summary

Our overall methodology is presented in Fig.3 and can be summarized by the following four phases:

- Phase 0 : Preliminary step where the design is split into logic cones. In the same time, each logic cone is parsed to extract the different signal probabilities of each endpoint. This is done by propagating the primary inputs signal probabilities throughout the design.
- Phase 1 : Each logic cone is then analyzed separately to determine the set of faulty paths for a given operating condition. At this step, the delay variation model $\Delta T_p(V,T)$ of the FPGA is needed to quantify the operating condition added delay to each path. A path is considered faulty when that added delay is greater than its slack.
- Phase 2 : Each logic cone containing at least one faulty path is analyzed to determine its faulty input combinations. Logic cones without any faulty path have an error rate equal to 0 for the considered operating condition.
- Phase 3 : The previously obtained input combinations are expanded into input transitions, which are then separated into two groups; one for the ones definitely causing timing errors and the second for the other ones likely to cause timing errors depending on the circuit implementation. The error rate of the logic cone endpoint is then obtained by computing the probability of occurrence of these combinations.

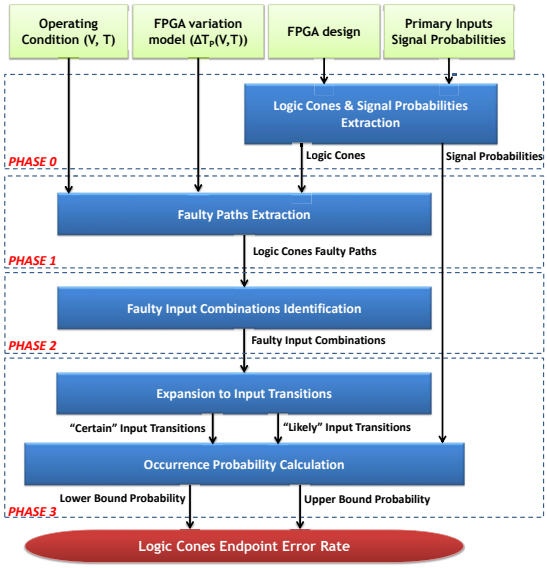


Figure 3: Methodology to quantify the error rate of a design for a given operating condition.

3. Verification and Accuracy

To verify our method, we randomly generated several 4-input logic cones presented in Table 1. We then measured their error rates and compared them with the ones obtained from our methodology. We first present

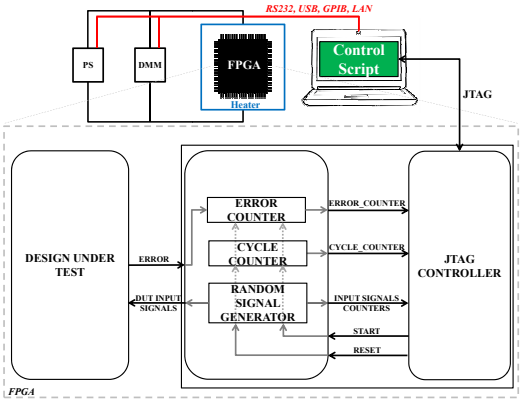


Figure 4: Measurement platform with the embedded instrumentation to measure the error rate.

the error rate measurement setup before discussing the results.

3.1. Error rate measurement setup

To measure the error rate, we developed a measurement platform consisting in a programmable power supply used to set the FPGA voltage, a digital multimeter to verify the set voltage and a heater used to control the FPGA temperature. To compute the error rate, an embedded instrumentation, running at the same frequency as the design under test, is implemented in the FPGA. It consists in an error counter to count the number of errors that occurred during the test and a cycle counter to count the number of clock cycles that the test lasts. The error rate is then equal to the ratio of the number of errors over the number of cycles. These counters can be accessed from a computer through a JTAG connection. A script is used to control the entire measurement sequence, from applying the required voltage on the FPGA to reading back the content of error and cycle counters from the embedded instrumentation to compute the error rate. It is not possible to directly access the JTAG connection so a JTAG server is connected to the FPGA embedded instrumentation, offering the control script an access to the counters via TCP/IP on a client/server model. To

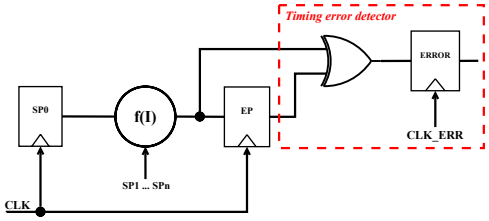


Figure 5: Timing error detection mechanism from [8].

Table 1: Randomly generated logic cones used to verify the proposed methodology.

Name	Truth Table	Equation
LC1	0x3F65	$\overline{C}.A + D.\overline{C} + D.\overline{B} + \overline{D}.B.\overline{A} + C.\overline{B}.A$
LC2	0x4F76	$\overline{B}.A + D.\overline{C} + \overline{D}.A.\overline{B} + \overline{D}.C.\overline{B}$
LC3	0x5A7A	$(C \oplus A) + \overline{D}.A.\overline{B}$
LC4	0xC55F	$\overline{D}.\overline{C} + \overline{D}.\overline{A} + \overline{C}.\overline{A} + D.C.B$
LC5	0xDC9E	$\overline{C}.B + B.A + D.B + \overline{D}.\overline{C}.A + \overline{B}.C.\overline{A}$

count the number of errors that occurred, the design under test provides an error output indicating that a timing error occurred in the design under test. That error signal is then sampled by the error counter which is incremented each time an error is detected. The control script can also start, stop and reset the counters as presented in Fig.4.

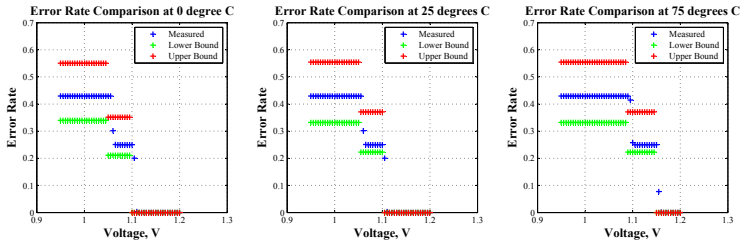
Errors are detected using the timing error detector described in [8] and presented in Fig.5. To detect a timing error, the value on the input and output of the capture DFF are compared using an XOR gate which will have its output at logic level high when its input values differ. The XOR gate output is then latched on the rising edge of CLK_ERR which has the same frequency as CLK but shifted by an amount of time T_{shift}, calibrated to detect the timing errors. Hence, in the case of a timing error, the erroneous value sampled by the capture flip-flop at the rising edge of CLK will differ from the valid data that will arrive after the CLK edge and before T_{shift}, leading to the detection of the timing error.

For each design, the error rate is measured in several operating conditions by varying the FPGA temperature as well as its voltage. Specifically, a temperature is first applied, and the FPGA voltage is set to its nominal voltage. The error and cycle counters are then started at nominal voltage when no error could happen. The test required voltage is then applied and after a dwell time, the error and cycle counters are stopped and the FPGA voltage is set back to the nominal voltage. The counters are then read from the FPGA, the error rate is calculated and the counters are reset. The same operation takes place for each couple of voltage and temperature.

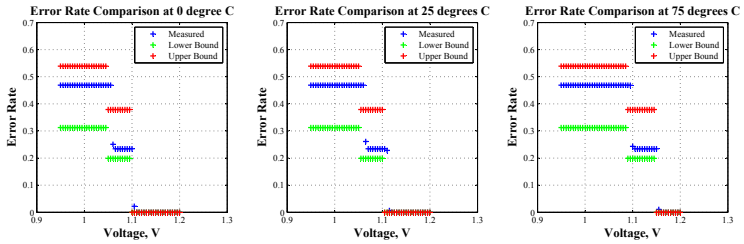
Independent random generated signals are connected to the inputs of the logic cone and are associated to a counter which is incremented each time the generated signal is at logic level high. Thus, by reading these counters, the associated signal probabilities can be calculated and are used to estimate the error rate using the proposed method.

3.2. Results and comparison

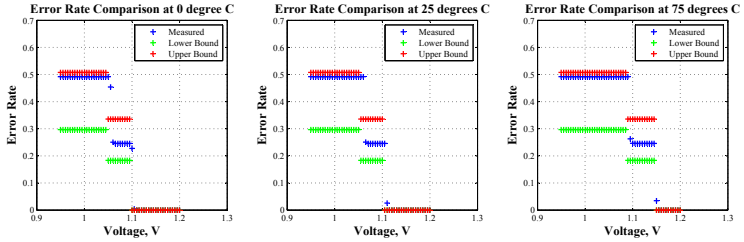
The logic cones presented in Table 1 have been individually implemented in the FPGA and tested according to the protocol previously described. For the FPGA implementation, each gate of the logic cone



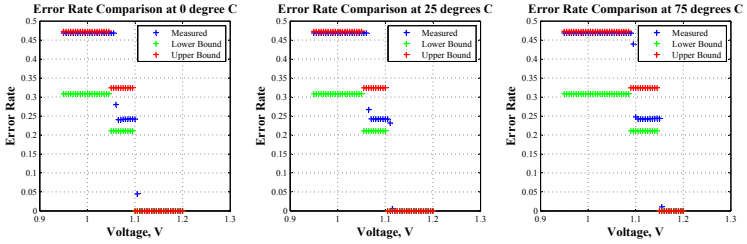
(a) Comparison between measurements and prediction for logic cone LC1.



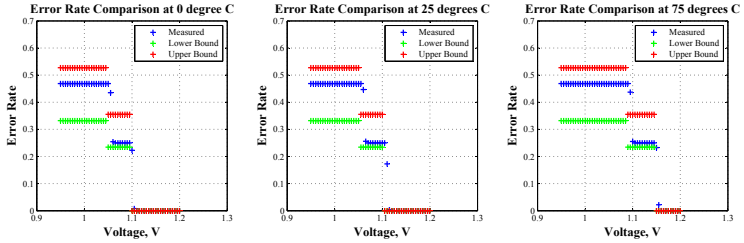
(b) Comparison between measurements and prediction for logic cone LC2.



(c) Comparison between measurements and prediction for logic cone LC3.



(d) Comparison between measurements and prediction for logic cone LC4.



(e) Comparison between measurements and prediction for logic cone LC5.

Figure 6: Validation of the method : Comparison between measured and estimated error rates for each randomly generated logic cone and for several operating conditions. The measured error rate is always between our two theoretical bounds.

have been assigned to an FPGA logic element. The voltage has been varied from the nominal $1.2V$ down to $0.95V$ by steps of 5 millivolts and for three values of temperature, $0°C$, $25°C$ and $75°C$. Hence, for each logic cone, 153 different operating conditions have been tested.

The results are presented in Fig.6. When the error rate is equal to 0, the logic cone has no faulty path at the corresponding operating conditions. The other levels correspond to more paths becoming faulty. The gap between the measured value and the lower and upper bounds is due to the pessimistic approach adopted and previously explained. In fact, all the "likely" transitions are also considered to cause timing errors, which is not the case in reality. It can be seen that only a fraction more or less important of these transitions cause timing errors. We are working on this in order to give more accurate error rates. It can be seen that the measured error rate is always between the lower and the upper bound from our proposed method.

4. Conclusion and Future Works

In this paper, we presented a methodology to estimate the error rate of each endpoint of a design based on the operating conditions. It has been shown that successfully passing static timing analysis is not always sufficient in the case of applications subject to voltage and temperature variation. Hence, the proposed methodology represents a step towards the identification of the endpoints to be monitored in such an application to ensure its correct execution. Being able to identify the application-critical endpoints helps to reduce the overhead of the detection/correction scheme when used to ensure the correct execution. The results of our methodology when applied to estimate the error rate of some FPGA designs have shown a good correlation with the measured error rates.

There is still room for improvement and we are currently working on adding one last step to the methodology aiming at analyzing in depth glitch-induced errors to improve the accuracy of the estimated error rate. A CAD tool will be developed in order to integrate this methodology into the FPGA development process of safety-critical applications.

Acknowledgment

This work is supported by General Council of Val dOise (CGVO) and the urban community (CA) of Cergy-Pontoise, under Grant Agreement number 14RETID382 (EXELA-VO project).

References

[1] A. Sassone, A. Calimera, A. Macii, E. Macii, M. Poncino, R. Goldman, V. Melikyan, E. Babayan, and S. Rinaudo, "Investigating the effects of inverted temperature dependence (itd) on clock distribution networks," in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012, pp. 165–166.

[2] *Guaranteeing Silicon Performance with FPGA Timing Models*, Wp-01139-1.0 ed., Altera, August 2010.

[3] *System Design with Advance FPGA Timing Models*, Wp-01213-1.0 ed., Altera, February 2014.

[4] *Xilinx Constraints Guide*, Ug625 (v. 14.5) ed., Xilinx, April 2013.

[5] *Vivado Design Suite User Guide - Design Analysis and Closure Techniques*, Ug906 (v2017.1) ed., Xilinx, April 2017.

[6] *Timing Closure*, Lattice Semiconductor, October 2013.

[7] M. A. Kacou, F. Ghaffari, O. Romain, and B. Condamin, "Influence of high-power electric motor on an fpga used in the drive system of electric car," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 4796–4801.

[8] S. Valadimas, A. Floros, Y. Tsiatouhas, A. Arapoyanni, and X. Kavousianos, "The time dilation technique for timing error tolerance," *IEEE Transactions on Computers*, vol. 63, no. 5, pp. 1277–1286, May 2014.

[9] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner, "Razor: circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, Nov 2004.

[10] J. Zhang, F. Yuan, R. Ye, and Q. Xu, "Forter: A forward error correction scheme for timing error resilience," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2013, pp. 55–60.

[11] M. R. Choudhury, V. Chandra, R. C. Aitken, and K. Mohanram, "Time-borrowing circuit designs and hardware prototyping for timing error resilience," *IEEE Transactions on Computers*, vol. 63, no. 2, pp. 497–509, 2014.

[12] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Dynamic voltage scaling for commercial fpgas," in *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005.*, Dec 2005, pp. 173–180.

[13] J. F. Freijedo, J. Semião, J. J. Rodriguez-Andina, F. Vargas, I. C. Teixeira, and J. P. Teixeira, "Modeling the effect of process, power-supply voltage and temperature variations on the timing response of nanometer digital circuits," *J. Electron. Test.*, vol. 28, no. 4, pp. 421–434, Aug. 2012.

[14] F. F. Sellers, M. Y. Hsiao, and L. W. Bearnson, "Analyzing errors with the boolean difference," *IEEE Transactions on Computers*, vol. C-17, no. 7, pp. 676–683, July 1968.