Safety Analysis for Integrated Circuits in the Context of Hybrid Systems

V Prasanth[1], Rubin Parekhji[1] and Bharadwaj Amrutur[2]
[1] *Texas Instruments (India) Pvt. Ltd., Bangalore, India.*
[2] *Indian Institute of Science, Bangalore, India.*
prasanth.v@ti.com, parekhji@ti.com, amrutur@ece.iisc.ernet.in

Abstract

Many real-life systems have integrated circuits interacting with physical systems in safety critical applications. These systems are called hybrid systems. The safety analysis of integrated circuits used in such systems is typically done in isolation of the end application and associated physical system, and hence results in the need to take recourse to conservative design techniques utilizing costly redundancy. We are gradually moving away from the paradigm of independently designing the digital and physical parts of hybrid systems towards simultaneous considerations for both. These systems have an acceptable tolerance determined by the application due to the inertial nature of the physical system, error tolerance capability in closed loop applications, built-in hardware and software functionality, etc. In this paper, we perform a comparative study of integrated circuit safety analysis as practiced today and system level application specific safety analysis that incorporates a physical system. We propose an improved method based upon the divide and conquer approach for such co-analysis to address practical limitations associated with adopting system level analysis techniques during integrated circuit design. Experimental results for a representative motor control system indicate that the application has an error tolerance of 92-160 cycles of closed loop operation for worst case errors and a control value error tolerance in the range of 5-7% at different operating conditions. Incorporation of application tolerance results in up to 4.3X reduction in the number of hardware elements which need to be protected.

Keywords: Reliability evaluation, safety analysis, error tolerance, closed loop system robustness.

I. INTRODUCTION

The count of electronic components in modern automobiles and other functional safety systems is on the rise. For automobiles, this ranges from the Antilock Braking System (ABS) and power steering to brake assist and collision avoidance, and further to autopilot systems. Similar examples exist in other functional safety systems as well. Many existing functions are modified from mechanical (manual) to electronic (automatic) implementations using integrated circuits (ICs). The increase in the computation power, miniaturization and power efficiency of ICs has helped in this endeavor [1, 2].

Along with the benefits that the use of ICs offer, it also leads to an increased set of design challenges caused by systematic failures due to the ageing effects of silicon structures (e.g. NBTI and HCI failures) [3] and random failures due to atmospheric particle strikes (alpha particles and neutrons) [4] termed as soft errors. Of the different failure mechanisms, random failures due to atmospheric particle strikes pose the biggest threat to the reliable operation of ICs. These challenges are well illustrated in some of the recent publications [5, 6].

In addition to the increased failure rate, ICs offer additional challenges due to complexity in analyzing the potential failure modes [7]. The failure modes of the mechanical systems (which the modern electronic components are replacing) are predictable and render themselves to an easy analysis. But due to the inherent complexity in the design, manufacturing and functionality of ICs today, a much more rigorous approach is required. For example, if we compare a mechanical steering to a steer-by-wire system [2], a failure in mechanical steering will lead to predictable failure modes of loss of steering or insufficient steering. But for a steer-by-wire system, a bit-flip in the IC can cause the steer-by-wire system to even steer in the opposite direction, which is not a potential failure mode in the case of mechanical steering.

The problems of functional safety of electric and electronic systems are addressed in the avionics and industrial domains by having redundancy based control architectures [8, 9, 10] like *1oo2* (1-out-of-2 also known as Dual Modular Redundancy), *2oo3* (2-out-of-3 also known as Triple Modular Redundancy), etc. Redundancy based architectures lead to significant increase in the implementation cost. Since initially those systems were not mass produced, the additional cost incurred for functional safety was not as much a concern. This situation has changed in consumer, automobile, industrial and many of the modern systems wherein there is a requirement to keep the cost to a minimum while re-purposing an IC across multiple applications. This is forcing us to think along new vectors for IC safety analysis and use of cost effective hardened circuit components [11], improved design techniques [12] and improved architectural methods [13].

Not every random fault occurring in an IC will result in application failure. Different masking effects (e.g. logical masking, electrical masking, latching window masking and application level masking) [14] can prevent the fault from propagating to the output and causing the application to fail. Safety analysis without considering these masking effects will result in over-design leading to an increased hardware overhead. While there is reported work on effects of logical, electrical and timing masking, those due to application masking have not been as well studied. Analysis at higher abstraction levels helps to incorporate additional masking effects and these can be beneficially employed to reduce the hardware overhead. On the other hand, analysis at higher abstraction levels significantly

978-1-5386-3413-4/17/$31.00 ©2017 IEEE

increases the analysis complexity. As an illustration, an application level analysis for functional safety of a motor control system will require consideration of motor and its different operating conditions involving motor speed, load, etc. This requires inclusion of the motor models and makes the analysis more complex, as compared to that of the IC considered standalone. Figure 1 indicates the hardware overhead and analysis complexity trade-offs associated with functional safety analysis when carried out at different abstraction levels.

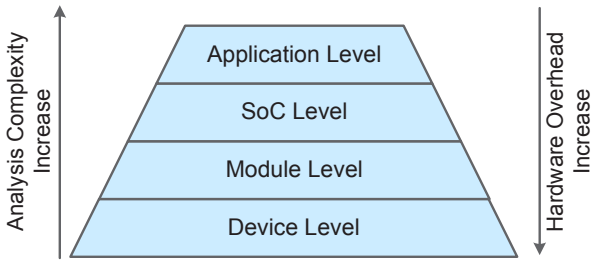

Figure 1. Safety analysis complexity and hardware overhead tradeoffs.

Many of today's real life safety critical systems have high accuracy and self-correction requirements to adjust for the physical system's non-linearity and noise effects. These systems are designed as closed loop control systems and typically have an associated acceptable tolerance in the extent to which the behaviour can deviate from the ideal or centre position. The control algorithms [15, 16] for these systems are also designed to accommodate the variability in physical system. The design of ICs used in such systems can beneficially employ these system level tolerances to identify the minimum set of components that are required to be protected, thereby reducing the hardware overhead. However, this significantly increases the safety analysis complexity.

In order to address the complexity issues associated with system level analysis, a new divide and conquer approach is proposed in this paper. The approach involves a two-step process. The first step uses the system level physical system tolerance information to apportion the system level tolerance to the individual modules in the system, including the IC (or ICs). The apportioned tolerance information is then used to identify the components which need to be protected. The proposed approach is illustrated using a representative system, i.e. a three phase Brushless DC (BLDC) motor control application. The hardware overhead associated with three different approaches is evaluated and compared, namely, (i) existing approach as practiced today during IC design, (ii) ideal approach of system level evaluation using the physical system information, and (iii) proposed divide and conquer approach.

The main contributions of this paper are: (i) proposing an integrated approach for the functional safety analysis of hybrid systems, (ii) apportioning of system level tolerances as value tolerance and time tolerance to various modules in the system, (iii) divide and conquer approach where the value and time tolerances derived from the system level analysis is used to perform module level analysis, including that of the ICs, and (iv) study of the benefits and limitations of the proposed approach.

The rest of the paper is organized as follows. Section II gives a brief overview of related work in this area. Section III describes the improved safety analysis technique. The experimental setup used to demonstrate the benefits of the proposed approach is explained in Section IV. Some analysis of the results is presented in Section V and Section VI concludes the paper.

II. Background and Related Work

Many real-life systems have ICs interacting with physical systems in safety critical applications. These systems are typically designed as closed loop control systems. A closed loop system, by its very nature, can correct certain errors since such a system can build its resilience across subsequent iterations of the control loop. In addition, the interacting physical system has latency and tolerance. A change in the control value driving the physical system may not reflect immediately on its behaviour due to the inherent inertia. As an example, in a BLDC motor, a stuck-fault on the control input takes about 92 cycles of closed loop operation (4.6 ms) to result in a 5% variation in the motor speed. In this paper, we propose improved safety analysis techniques to consider these effects while designing an optimised system which meets functional safety requirements.

In order to ascertain the suitability of an IC to be used in functional safety applications, it goes through a safety analysis phase. Fault injection [17] is the preferred technique used to ascertain the safety worthiness (robustness) of the circuit. Fault injection based robustness evaluation requires us to ascertain the impact of random faults in every circuit element in each operating cycle of the work-load. Comprehensive evaluation of robustness is a computationally intensive problem and several techniques have been suggested to speed-up this evaluation [18, 19]. As safety analysis becomes a mandatory step in the IC design process, EDA vendors have also started supporting fault injection and associated data analysis [20].

Several analytical and statistical methods have also been proposed to speed-up the functional safety analysis of ICs. In RAVEN (RApid Vulnerability EstimatioN) [21], the authors propose partitioning the IC into smaller sub-blocks followed by separate analysis of each sub-block. The sub-block analysis results are then used to estimate the system level error probability and identification of critical components. [22] proposes an approach for computation of output error probability using individual signal probability and fault propagation probability. The authors introduce the concept of using time tolerance for soft error evaluation and propose the use of Markov models to estimate error

probability after the time tolerance window. [23] proposes a correlation based approach to identify dependencies of signal probabilities of error free signals and error probability of erroneous signals to provide higher SER estimation accuracy.

These methods, however, treat IC safety as a standalone problem without considering its interaction with the surrounding physical system. Even when system level effects are considered, they are dependent on a particular state and not on the closed loop behaviour [21, 22]. Hybrid systems, on the other hand, have multiple such states which are acceptable across different control loop iterations. In the absence of hybrid system consideration, the analysis will be pessimistic [24].

Multiple research works have focused on implementing robust control in the presence of errors [25] for generic control systems as well as for specific motor control systems [26]. These works have been limited to robustness to sensor and actuator errors within certain parametric bounds and do not consider interactions between the closed loop modules. There are several observer based approaches employed for fault detection. VDA consortium (Verband Der Automobilindustrie - German Association of the Automotive Industry) came up with a monitoring concept to ensure functional safety of gasoline and diesel engines known as VDA E-Gas architecture [27]. This approach is widely used in the engine control system of many automobiles [28]. In [29], the use of residual generator and residual evaluation module for fault detection purposes is proposed. [30] introduces the concept of using mapped predictive check states for the detection of transient faults in the controller. These approaches help in online fault detection, but do not help to localize the fault or take the corrective action. As fail operational system requirements, (i.e. systems which continue to operate even after a fault is detected), are becoming more prominent in automotive applications [31], these approaches have some limitations which restrict their applicability.

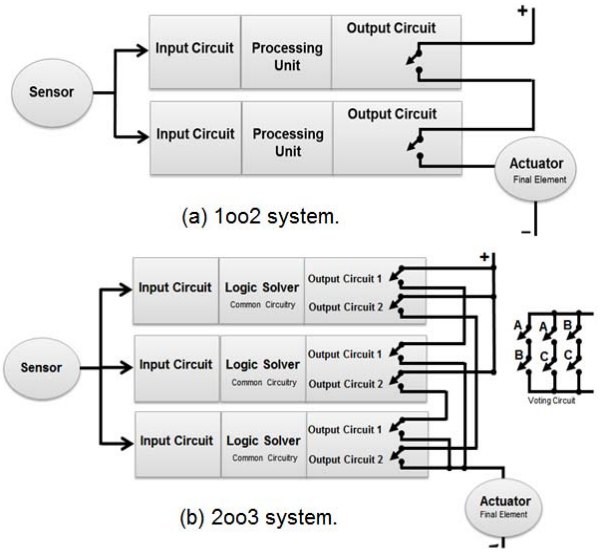

Figure 2. Representative fault tolerant systems.

Due to the complexity associated with the functional safety analysis for applications requiring high reliability, many safety critical systems employ redundancy at the system level, namely *1oo2* (DMR), *2oo3* (TMR). These are shown in Figure 2. The higher cost incurred in designing such redundancy systems is driving the need to impart functional safety with reduced design overheads, and hence improved safety analysis techniques.

The methodology in this paper addresses some limitations of earlier work reported in the literature by including some new artefacts into the analysis. (i) Safety analysis for the IC is performed together with the interacting physical system making the analysis more reliable. (ii) Application level attributes like closed loop operation and acceptable error are considered for the analysis making it less pessimistic. (iii) In order to reduce the analysis complexity due to the additional system information being considered, a divide and conquer approach is used. (iv) Critical flip-flops required to be protected are directly identified using this analysis, as against restricting it to just identifying the system drifting into an unsafe state, thereby enabling the implementation of low cost monitoring and correction mechanisms. Low overhead design robustness (either through component hardening or fault tolerance) techniques can then be used for protecting the right sub-set of critical components (e.g. flip-flops) as against the entire design.

III. Improved Safety Analysis Technique

A representative hybrid system is shown in Figure 3. It consists of a physical system comprising of the motor, power stage providing current to the motor and Hall sensor determining the motor position. This physical system is controlled by a digital controller whose main function is to maintain the speed of the motor within an acceptable range as set using the communication interface.

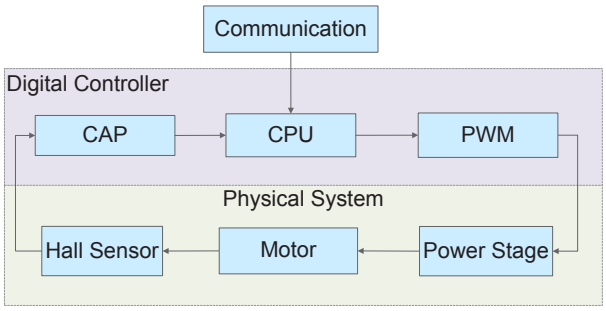

Figure 3. Illustration of a hybrid system.

The digital controller consists of: (i) A CAP (CAPture) module which decodes the Hall sensor output and provides rotor position information to CPU. (ii) The CPU which receives the speed set-point information and rotor position

information of the motor. It executes the control algorithm to determine the actuation required to maintain the desired motor speed. (iii) A PWM (Pulse Width Modulator) which provides pulsed inputs to the power stage which drives the motor. The motor speed is proportional to the PWM duty cycle. (The feedback control loop adjusts the motor speed under different operating conditions).

For the purpose of discussion in this paper, only random faults due to single event upsets (SEUs) are considered. (Permanent faults are covered at the time of power on, using self-test methods (functional or structural). This form of self-test cannot take advantage of tolerance during the real-time execution of the application and is not the focus of this paper).

A. *Value and Time Tolerance*

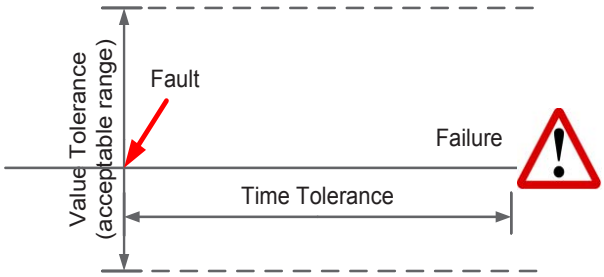

Figure 4. Tolerance in value and time over the control input range.

Hybrid systems are typically associated with a tolerance range around the control point wherein they can operate in an acceptably correct manner and any minor perturbation around this point can be corrected during the course of time, (i.e. in the subsequent operating loops), without the application being perceptibly impacted. We refer to this as *value tolerance*. This depicts the set of control values around the correct control value using which the system can operate in an acceptably correct manner. This can also be visualized as the maximum steady state error which can be tolerated by the system.

Additionally, these systems can be slow to react (with respect to the frequency of the clock used in the digital control). It will take some finite time, (i.e. a few operating loops), termed as *time tolerance*, for the system to respond to a change in the control input, even for real-time applications. Time tolerance depicts the time duration for which the system is able to function in an acceptably correct manner for a maximum deviation in the control values.

The tolerance in value and time is depicted in Figure 4. Beyond this tolerance range, i.e. beyond a range of control values and / or beyond a given time duration, the system can drift and go outside the acceptable operating range. The closed loop operation can either contain or correct many of the errors within an IC without causing the system to fail.

The combined impact due to value tolerance and time tolerance can be complex since not all value tolerances may be applicable across the time tolerance windows. In this paper, therefore, we calculate the time tolerance across the entire operating range for the worst-case variation in the values of control inputs.

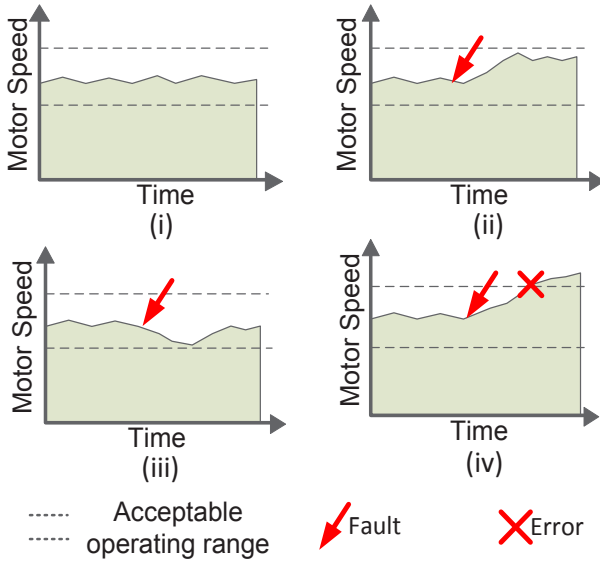

Figure 5. Motor speed variation for various injected errors.

To understand the tolerance in a motor control system, we performed random fault injections (explained further in Section IV) and observed the controlled parameter (speed) variation with time after the faults are injected. The various profiles thus obtained are illustrated in Figure 5. Case (i) indicates the fault free scenario. Certain injected faults depicted in Case (ii), (e.g. those causing minor perturbation in closed loop algorithm co-efficients which are not corrected across iterations), cause perturbation in the motor speed; however the motor settles to a new speed which is within the acceptable range. Other faults depicted in Case (iii), (e.g. those in the data computation logic which get corrected in closed loop iterations), cause perturbation in speed which also gets corrected within an acceptable time For faults depicted in Case (iv), (e.g. those in the program counter which changes the control flow causing unrecoverable error), cause the motor speed to drift out of the acceptable range resulting in application failure. Safety analysis techniques employed to identify the critical elements within the IC should consider these system level tolerance effects. In its absence, the analysis will be pessimistic.

B. *Proposed Approach*

We have seen in Section 2 that fault injection is the preferred technique to identify the set of critical elements (e.g. flip-flops) which must be protected during the IC design phase. Fault injection driven simulation is performed wherein the faults are injected one at a time on

every flip-flop inside the circuit, and the corresponding outputs are observed. If the injected fault in a flip-flop results in an unacceptable change in at least one output (across value and time) which is not detected by any safety mechanism within the circuit, then this flip-flop must be protected.

In an ideal scenario, the physical system can be modelled and the key parameters of this system can also be observed during the fault injection experiments. Such co-simulations environments have been demonstrated for smaller logic [32] but have not been adopted for larger ICs due to limitations of slow progression of digital simulation, longer simulation time due to slower response of the physical system in response to any change in control input, etc. The increase in the simulation time along with the requirement to inject a fault in every operating cycle and on every flip-flop of the circuit significantly increases the computational complexity of such evaluations.

In order to address the simulation complexity issue, we propose a divide and conquer approach. This is carried out in two steps: (i) Application tolerance is mapped (apportioned) to individual module outputs as value and time tolerance. (ii) Safety evaluation is carried for the individual modules using these tolerance values. As an illustration for the mapping of application tolerance, consider the case of the closed loop motor control system in Figure 6. A ±5% variation is considered the acceptable tolerance for the motor speed. (This was determined based upon the peak overshoot for which the control system is designed). In this system, the application tolerance δ_{OT} (±5% variation in speed) has to be mapped to individual module outputs as value and time tolerance. In the figure, δ_{O1}, δ_{O2} and δ_{O3} denote the value and time tolerance (can be represented as a tuple $\{Vt, Tt\}$) at the output of the CAP, CPU and PWM modules.

The probability of soft errors is considered low. Recent works have indicated at least four orders of magnitude difference between SEU (single event upset) and MBU (multiple bit upset) rate [33]. As a result, in the fault injection campaigns, only single faults modelling SEUs are considered.

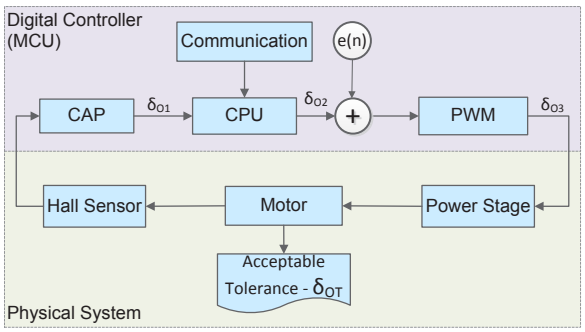

Figure 6. Computation of value and time tolerance.

Consider the case of estimating the value and time tolerance at the output of the CPU module. In order to estimate the time tolerance, values corresponding to maximum error, (i.e. the CPU outputs generating the maximum and minimum possible actuator input values in this case), are forced and the time required for the motor to go beyond the ±5% tolerable range of speed variation is obtained. i.e. for a motor operating speed of 900 RPM, the lower and upper speed limits are set at 855 and 945 RPM respectively. Similarly, to ascertain the value tolerance, the maximum and minimum permissible CPU output values during the steady state operating condition, are ascertained. This process can be repeated at the output of each module to determine the value and time tolerance of individual modules.

We enumerate the steps to compute the time tolerance and value tolerance of CPU below.

Time tolerance:

1) Force the maximum value at the CPU output (data bus interface of the module forced to 0xFF…) during application execution.
2) Determine the time taken by the application to go out of acceptable application tolerance (*Tt-max).*
3) Force minimum value at the CPU output (data bus interface of the module forced to 0x0…) during application execution.
4) Determine the time taken by the application to go out of acceptable application tolerance (*Tt-min).*
5) Time tolerance *Tt = min (Tt-max, Tt-min).*

Value tolerance:

1) Strobe the CPU output during application execution for the entire operating speed of the motor (corresponding to the full range of outputs of the PWM module). The module output changes within a particular range due to steady state error of the control system.
2) Determine the maximum (*Vt-max*) and minimum (*Vt-min*) values at the module output.
3) Value tolerance Vt = (*Vt-max* - *Vt-min*).

During the closed loop operation, there can be change in operating conditions (e.g. load), change in commanded speed, error in the sampling of input values (e.g. sensor linearity error), etc. The control algorithm execution is dependent upon these input parameters, which are captured into an associated memory and associated set of states. (The integral part of Proportional Integral Derivative (PID) control keeps a memory of the input values). Their combined effect will create multiple different application threads for the single application. Hence, it may be noted that the parameters δ_{O1}, δ_{O2} and δ_{O3} themselves can drive multiple {Vt, Tt} tuples since different variations in the

value (and time) tolerances can affect the individual modules differently. As a result, the complexity of the analysis can increase requiring multiple threads of execution across the fault injection experiments. In our work, we have considered the worst case value tolerance and estimated the time tolerance accordingly.

The process of mapping of the application tolerance to individual circuit modules can be considered similar to the problem of timing budgeting performed during the SoC design phase, wherein the timing slack available at the SoC level (due to the clock frequency as well as the I/Os interacting with the external system) is apportioned to individual IP modules therein. Timing closure for these modules is carried out individually using these apportioned budgets and integrated into the SoC. Such a methodology may not lead to an optimal design as the combined effect of interacting signals is not considered. However, such a divide and conquer approach is widely adopted due to practical limitations, (e.g. design and analysis complexity, EDA tool run-times, etc.), associated a flat timing analysis approach. This has also motivated the divide and conquer methodology presented in this paper.

Once the individual module tolerances are identified, they are used for standalone fault injection driven module analysis. (A simulation model, (e.g. Verilog netlist), or an emulation platform can be used). If the injected fault causes the output of the flip-flop to deviate beyond the value tolerance and beyond the time tolerance, then this flip-flop is marked for protection.

IV. Experimental Results

In order to evaluate the effectiveness of the proposed method, three experiments on the closed loop motor control system have been performed and their results compared.

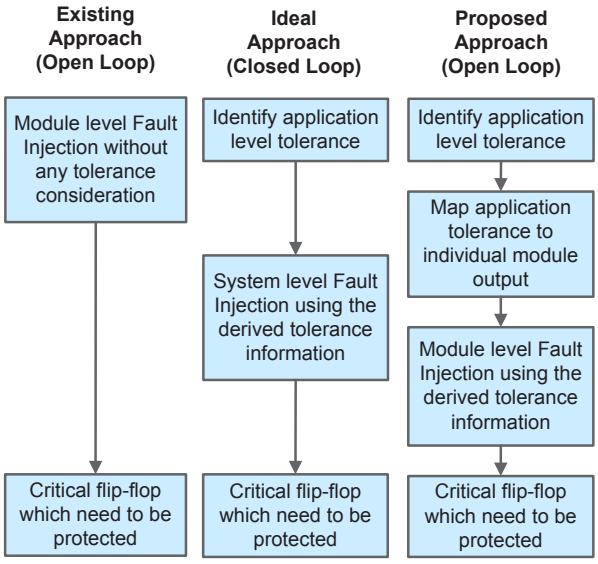

Figure 7. Safety analysis approaches.

These are indicated in Figure 7 and are explained below:

1) Existing approach: This consists of analysing individual modules in a standalone way without any tolerance consideration, as is the practice today

2) Ideal approach: This consists of identifying critical flip-flops at the hybrid system level using the system level tolerance information. No approximation or tolerance budgeting is employed here.

3) Proposed approach: This consists of performing module level analysis with tolerance allocation, i.e. the divide and conquer method described in Section III.

A hardware emulation setup was chosen as the platform for performing this comparative study. This has enabled a faster evaluation over simulation methods, and avoids the requirement for having a co-simulation environment with the physical model of the motor and the Verilog netlist of the controller. For a circuit in design stage, evaluating application level effects may be infeasible due to the absence of the physical system model and the complexity it adds during fault injection. In such cases, higher level system models (e.g. MATLAB Simulink [34]), can be used for deriving the individual module tolerances.

A. Experimental Setup

The experimental setup consists of a three phase eight pole BLDC motor driven by a voltage source inverter (DRV8312) [35] and Microcontroller (F2805x) [36] shown in Figure 8. The power stage of the inverter is actuated by a Pulse Width Modulator (PWM) [37] which is part of the Microcontroller (MCU). The MCU has a CPU which executes the closed loop control algorithm to drive the BLDC motor to rotate at the commanded speed. The BLDC motor is equipped with three Hall sensors which produce voltage outputs depending on the rotor position. The CAPture (CAP) module [38] of the MCU uses these outputs to estimate the rotor position. The motor speed is estimated based on the rate of change of position.

The control flow of the application is shown in Figure 9. The CPU executes the PID (Proportional Integral Derivative) control algorithm every time it receives an interrupt from the timer. For every Interrupt Service

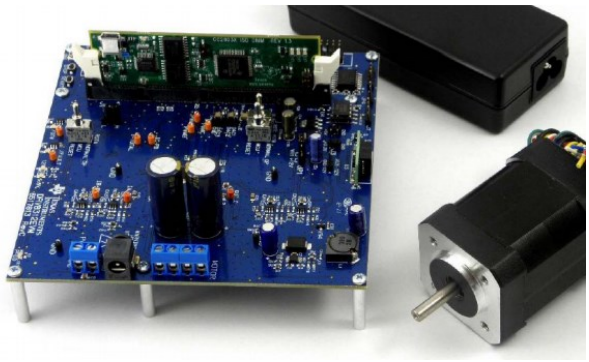

Figure 8. DRV8312, F2805x and BLDC motor.

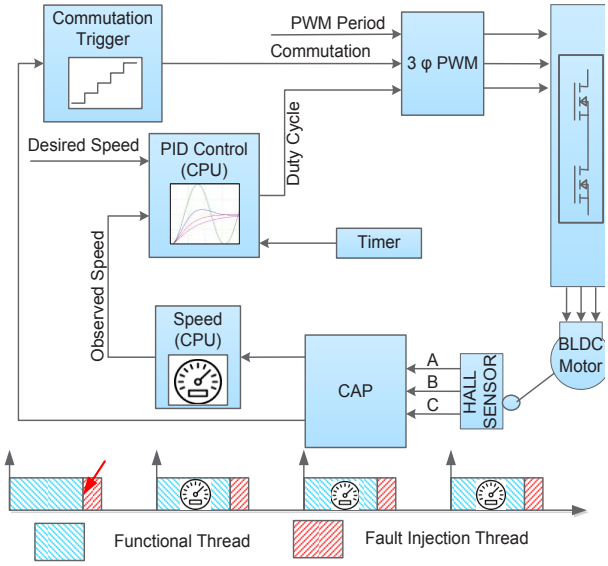

Figure 9. Dataflow of closed loop motor control application.

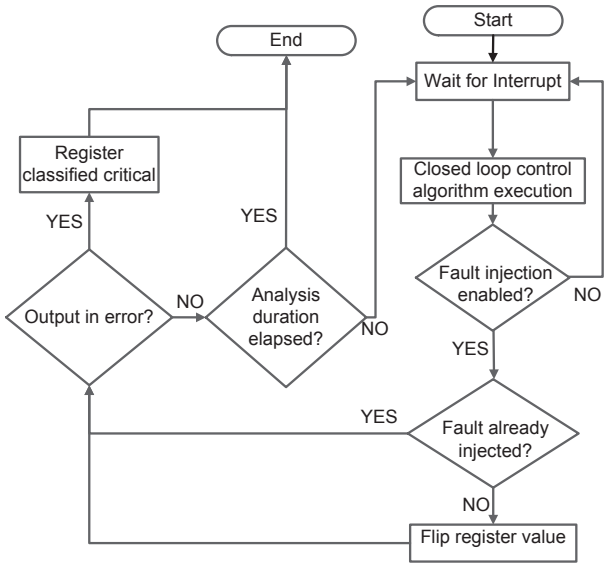

Figure 10. Software fault injection flow.

Routine (ISR), the CPU samples the CAP input and estimates the motor speed and position. This information along with the desired speed input received from communication interface is used to determine the magnitude of current to be applied on the three phase windings for torque generation. For this experiment, the CPU frequency is 80 MHz and the control loop frequency is 20 KHz. For demonstration of application tolerance, we have selected speed tolerance of ±5%, which corresponds to the peak overshoot for which the control system is designed.

Fault injection to identify the critical flip-flops is performed using software mutation. The approach involves creating a fault injection thread in parallel with the application thread. The modified application thread is shown in Figure 10. The fault injection thread inverts the values stored in the flip-flops, one at a time, at every cycle of the application execution, across multiple iterations corresponding to the workload. In this particular evaluation we have restricted the fault injection to memory mapped registers i.e. flip-flops which are directly accessible using software code. (This was the level of granularity feasible in the emulation setup used in this experiment). The output, (module output in case of existing approach and proposed approach and system output in the case of ideal approach), is observed for each fault. If the output drifts beyond the expected value, (golden output in case of existing approach and acceptable output with tolerance considerations in case of ideal approach and proposed approach), the flip-flop in which fault is injected is classified as critical and must be protected.

B. Module Interface Tolerance Evaluation

As illustrated in Section III.B, in order to evaluate the time tolerance of CPU input (CAP module output), we inject faults leading to worst case behavior at the CPU input and determine the number of cycles required for the motor speed to move out of the application tolerance range. The CAP module provides a 16 bit binary output value which is input to the CPU. Values of 0x0 (denoting zero speed i.e. stationary motor) and 0xFFFF (denoting the maximum speed i.e. motor spinning at full rated speed) are forced on the CAP module output.

Similarly, to evaluate time tolerance at the CPU output, we inject faults leading to worst case variation at the CPU output and determine the number of cycles required for the motor speed to move out of the application tolerance range. The CPU provides a 32 bit binary value to the PWM module. To compute the time tolerance at the CPU output, values of 0x0 (denoting zero duty cycle for PWM i.e. zero electrical energy applied to the motor) and 0xFFFFFFFF (denoting 100% duty cycle i.e. maximum electrical energy applied to the motor) are forced on the CPU outputs.

The results are plotted in Figure 11. The X-axis denotes the motor speed (RPM – revolutions per minute) under different operating conditions and the Y-axis denotes the number of closed loop cycles required for the motor to go out of the tolerable speed range. Four different values of time tolerance, corresponding to minimum and maximum values of CAP module output (CPU input) and PWM module input (CPU output), are calculated at different speed settings of the motor. These results indicate that the closed loop motor control application has a time tolerance of 92-160 cycles of closed loop operation (which is equivalent to 276K to 480K cycles of CPU operation, translating to 4.6 to 8.0 ms), which corresponds to +/-5% speed variation for different operating speed settings. The application can operate in an acceptably correct manner for at least 92 cycles across the full RPM range in the presence

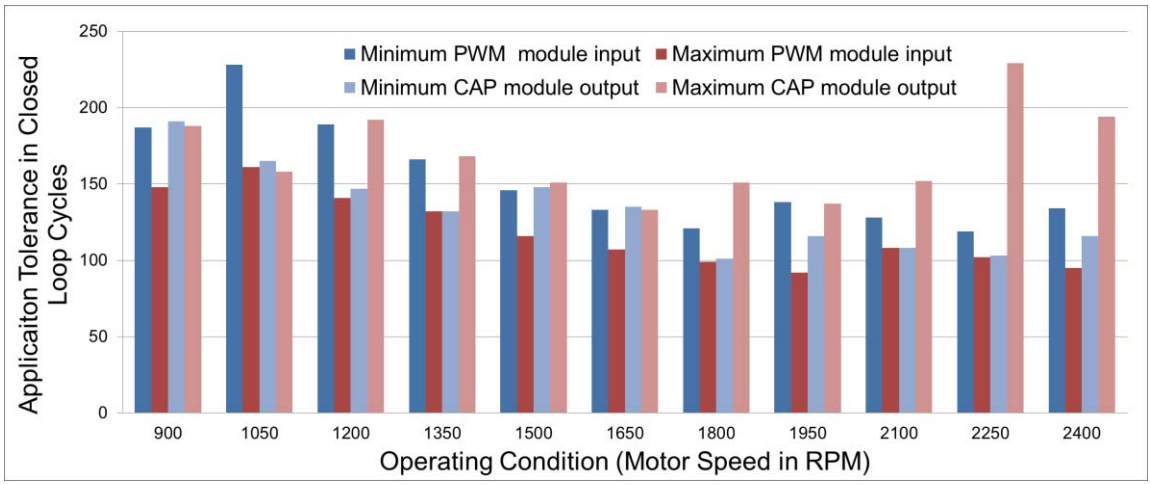

Figure 11. Application time tolerance in the presence of worst case errors.

of an error, across the worst-case variation in the CAP module output and PWM module input.

In order to compute value tolerance, the CPU output (PWM input) is monitored during application execution. Due to variations in the operating conditions and error in position / speed determination, there is a slight variation in the sensed speed and the CPU output will change based on the algorithm being executed. The tolerable variation (steady state error) in CPU output thus obtained for different speed settings is tabulated and its variation from the value corresponding to desired speed is noted. The minimum (maximum) tolerance is observed at an RPM of 1650 (2400). The tolerable CPU output variation at the RPM of 1650 (2400) is 0.54-0.568 (0.785-0.842) which equals a variation of 5% around 0.55 (7% around 0.8). The CPU output tolerance expressed as a percentage of CPU output value is shown in Figure 12. Here the X-axis denotes the different motor speeds, and the Y-axis denotes the CPU output value tolerance.

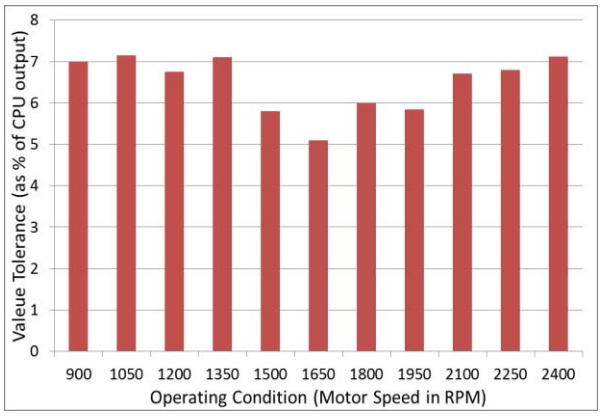

Figure 12. Application value tolerence as percentage of CPU output value.

C. *Comparison of Different Approaches*

Fault injection is performed to identify critical flip-flops using the three approaches illustrated in Figure 7. The number of flip-flops thus identified is shown in Figure 13. For the motor speed setting of 900 RPM, 190 flip-flops were identified as dangerous in the existing approach, while the numbers were 48 and 50 for the ideal approach and the proposed approach respectively. Similar results were observed for other speed settings as well.

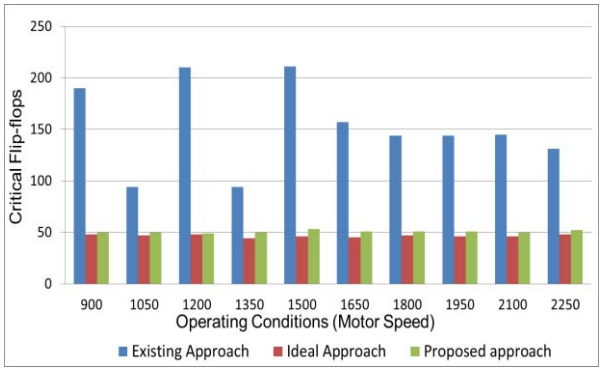

Figure 13. Critical elements identifed at different operating conditions.

Since the control must be robust for any operating speed of the motor, the union of flip-flops identified as critical at different speed settings is considered. 239 flip-flops were accordingly identified as critical using the existing analysis technique, 54 using the ideal approach and 55 using the proposed approach. The results thus indicate a 4.3x reduction in the number of critical flip-flops using the proposed approach.

V. Analysis of Results

The above experiments were performed using the same closed loop control code for all the three approaches. We further analyse the results and make a few observations:

1) In the absence of any tolerance considerations, the existing approach (most pessimistic) must identify the maximum number of flip-flops as dangerous. A subset of these will be identified as critical using the proposed approach with the mapped application tolerances applied at the module level. This may not be the minimal set since interactions between modules are not considered. A smaller subset will be identified as critical using the ideal approach with closed loop analysis performed using system model together with application tolerances. The flip-flops identified as critical are mapped into the three main groups in Figure 14.

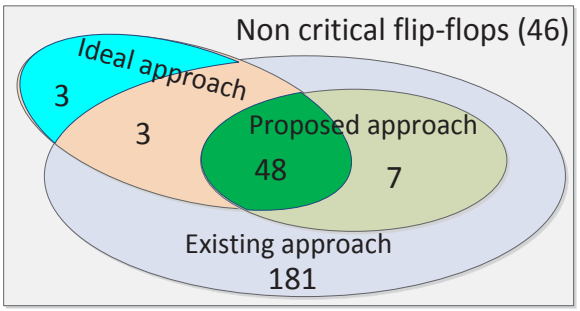

Figure 14. Critical flip-flops identified using each approach.

2) Contrary to expectations, the following discrepancies were observed. Six flip-flops identified as critical in the ideal approach were not identified using the proposed approach. Three flip-flops identified as critical in the ideal approach were not so identified even using the existing (most pessimistic) approach.

3) The ideal approach with the closed loop model represents the real life situation and is therefore more accurate. In the closed loop approach, slight differences in the measured speed due to the behaviour of the Hall sensor and PID control algorithm are also considered. Such variations across different closed loop iterations cause different application threads to be executed as against a single thread with constant reference speed executed in the other two approaches. (This was also indicated in Section III.B, wherein the parameters δO1, δO2 and δO3 were indicated to drive multiple {*Vt, Tt*} tuples). These multiple threads have resulted in the identification of three additional flip-flops as critical using the ideal approach as compared to the existing approach.

We have not focused on exactly matching the results from the different approaches given the low margin of error (6/288 = 2.1%, which is within the acceptable margin of error typically used to derive the sample size for the fault injection campaigns [39]). (Even the existing approach suffers from such errors due to workload restrictions, i.e. lack of comprehensive set of workloads, associated with fault injection campaign).

VI. Conclusion

In this paper, we have demonstrated how the available tolerances in a closed loop system can be utilized to reduce the pessimism in the safety analysis of integrated circuits, thereby reducing the number of flip-flops which are identified as critical, (i.e. state corrupting, leading to intolerable excursions in the system performance), thus saving the area overhead required to make the circuit safe. We have also proposed a divide and conquer approach to address the practical limitations in performing such an evaluation during the IC design phase. The presented results demonstrate the effectiveness of the proposed approach in identifying the critical flip-flops. Experiments on a brushless DC motor control system indicate a 4.3x reduction in the number of flip-flops which must be protected (from 239 to 55).

Further investigations are planned towards identifying pessimism in the proposed approach (flip flops wrongly being identified as critical) and vice-versa, efficient methods to analyze multiple threads during iterations, partitioning and tolerance allocation methods within modules, and embedded check mechanisms for identification and protection of critical flip-flops based upon application requirements.

Appendix: Algorithm for Identification of Critical Flip-flops

The module under evaluation can be considered as generating an output $\{O\}$ based on the input $\{I\}$ and the state in the set of module flip-flops $\{S\}$ holding a specific value. In the presence of a fault in one of the module flip-flops $s \in S$ and for the same input data stream $\{I\}$, the output changes to $\{O'\}$. The golden output $\{O\}$ and output under fault injection $\{O'\}$ are compared. If the change in flip-flop value causes the module output to deviate more than permissible value tolerance Vt after the time tolerance Tt, then the flip-flops in which the fault is injected is added to the set of critical flip-flops $\{C\}$ and the process is repeated for each flip-flop of the module.

Identification of critical flip-flops

```
Critical_flip-flop_Identification (S)
    C = φ;
    O = f(S, I)

for each s ∈ S do
    s = s'
    O' = f(S', I)
    If f(O, O') > Vt after Tt
      C = C U s
    end for
```

References

1.L. Cripps, "Electronics in cars," *IET Journal for Electronics and Power*, 1970.

2.R. Isermann, R. Schwarz, and S. Stolzl, "Fault-tolerant drive-by-wire systems," *IEEE Control Systems*, 2002.

3.D. Lorenz, G. Georgakos, and U. Schlichtmann, "Aging analysis of circuit timing considering NBTI and HCI," in *International On-Line Testing Symposium*, 2009.
4.R. C. Baumann, "Radiation induced soft errors in advanced semiconductor technologies," *IEEE Tran. on Device and Materials Reliability*, 2005.
5.A. Geist, "Supercomputing's monster in the closet," *IEEE Spectrum*, 2016.
6.G. Hubert, L. Artola, and D. Regis, "Impact of scaling on the soft error sensitivity of bulk, fdsoi and finfet technologies due to atmospheric radiation," *Integration, the VLSI journal*, 2015.
7.R. Mariani and G. Boschi, "A systematic approach for failure modes and effects analysis of system-on-chips," in *International On-Line Testing Symposium*, 2007.
8.A. Hayek and J. Börcsök, "Safety chips in light of the standard IEC 61508: survey and analysis," in *International Symposium on Fundamentals of Electrical Engineering*, 2014.
9.E. Ugljesa and J. Börcsök, "Evaluation of sophisticated hardware architectures for safety applications," in *International Symposium on Information, Communication and Automation Technologies*, 2009.
10. W. M. Goble and H. Cheddie, *Safety Instrumented Systems verification: practical probabilistic calculations*. ISA, 2004.
11. T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron cmos technology," *IEEE Tran. on Nuclear Science*, 1996.
12. V. Prasanth, V. Singh, and R. Parekhji, "Derating based hardware optimizations in soft error tolerant designs," in *VLSI Test Symposium*, 2012.
13. P. Subramanyan, V. Singh, K. K. Saluja, and E. Larsson, "Multiplexed redundant execution: A technique for efficient fault tolerance in chip multiprocessors," in *Design, Automation & Test in Europe*, 2010.
14. M. Ebrahimi, A. Evans, M. B. Tahoori, R. Seyyedi, E. Costenaro, and D. Alexandrescu, "Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales," in *Design, Automation & Test in Europe*, 2014.
15. Q. Zhao and J. Jiang, "Reliable state feedback control system design against actuator failures," *Automatica*, 1998.
16. G.-H. Yang, J. L. Wang, and Y. C. Soh, "Reliable h-infinity controller design for linear systems," *Automatica*, 2001.
17. A. Benso and P. Prinetto, *Fault injection techniques and tools for embedded systems reliability evaluation*. Springer Science & Business Media, 2003.
18. F. Kriebel, S. Rehman, D. Sun, P. V. Aceituno, M. Shafique, and J. Henkel, "Acsem: Accuracy-configurable fast soft error masking analysis in combinatorial circuits," in *Design, Automation & Test in Europe*, 2015.
19. S. Mirkhani, S. Mitra, C.-Y. Cher, and J. Abraham, "Efficient soft error vulnerability estimation of complex designs," in *Design, Automation & Test in Europe*, 2015.
20. A. Sherer, J. Rose, and R. Oddone, "Ensuring functional safety compliance for iso 26262," in *Design Automation Conference*, 2015.
21. H. Cho, S. Mirkhani, C.-Y. Cher, J. A. Abraham, and S. Mitra, "Quantitative evaluation of soft error injection techniques for robust system design," in *Design Automation Conference*, 2013.
22. I. Polian, J. P. Hayes, S. M. Reddy, and B. Becker, "Modeling and mitigating transient errors in logic circuits," *IEEE Tran. on Dependable and Secure Computing*, 2011.
23. L. Chen, M. Ebrahimi, and M. B. Tahoori, "Cep: correlated error propagation for hierarchical soft error analysis," *Journal of Electronic Testing*, 2013.
24. V. Prasanth, R. Parekhji, and B. Amrutur, "Improved methods for accurate safety analysis of real-life systems," in *Asian Test Symposium*, 2015.
25. T. Maeba, M. Deng, A. Yanou, and T. Henmi, "Swing-up controller design for inverted pendulum by using energy control method based on lyapunov function," in *IEEE International Conference on Modelling, Identification and Control*, 2010.
26. M. I. Momtaz, S. Banerjee, and A. Chatterjee, "Real-time dc motor error detection and control compensation using linear checksums," in *VLSI Test Symposium*, 2016.
27. Standardized e-gas monitoring concept for gasoline and diesel engine control units. [Online]. Available: https://www.iav.com/-sites/default/files/attachments/seite/ak-egas-v6-0-en-150922_1.pdf
28. D. Geyer, M. Kick, and M. Kraus, "Monitoring the functional reliability of an internal combustion engine," Patent 8,392,046, 2013.
29. P. Pisu, *Fault Detection and Isolation with Applications to Vehicle Systems*. Springer, 2016.
30. S. Banerjee, A. Chatterjee, and J. A. Abraham, "Efficient cross-layer concurrent error detection in nonlinear control systems using mapped predictive check states," in *International Test Conference*. IEEE, 2016.
31. A. Kohn, R. Schneider, A. Vilela, U. Dannebaum, and A. Herkersdorf, "Markov chain-based reliability analysis for automotive fail-operational systems," *SAE International Journal of Transportation Safety*, 2017.
32. Y.-S. Kung, N. V. Quynh, N. T. Hieu, C.-C. Huang, and L.-C. Huang, "Simulink/modelsim co-simulation and fpga realization of speed control ic for pmsm drive," *Procedia Engineering*, vol. 23, pp. 718–727, 2011.
33. C. Bottoni, M. Glorieux, J. Daveau, G. Gasiot, F. Abouzeid, S. Clerc, L. Naviner, and P. Roche, "Heavy ions test result on a 65nm sparc-v8 radiation-hard microprocessor," in *Reliability Physics Symposium, 2014 IEEE International*, 2014.
34. S. A. R. Kashif. (2013) Speed control of brush-less dc motor. [Online]. Available: https://in.mathworks.com/matlabcentral/-fileexchange/42060-speed-control-of-brush-less-dc-motor
35. DRV8312 - Three Phase Brushless DC Motor Driver IC. [Online]. Available: http://www.ti.com/product/DRV8312
36. F2805x - Real time control MCU. [Online]. Available: http://-www.ti.com/product/TMS320F28055
37. Enhanced Pulse Width Modulator (ePWM) Reference Guide. [Online]. Available: http://www.ti.com/lit/ug/spruge9e/-spruge9e.pdf
38. Enhanced Capture Module (eCAP) Reference Guide. [Online]. Available: http://www.ti.com/lit/ug/sprufz8a/-sprufz8a.pdf
39. R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *Design, Automation and Test in Europe*, 2009.