On Simulation-based Metrics that Characterize the Behavior of RTL Errors

Zissis Poulos
University of Toronto
10 King's College Road,
Toronto, Canada
zpoulos@eecg.toronto.edu

Ryan Berryhill
University of Toronto
10 King's College Road,
Toronto, Canada
ryan@eecg.toronto.edu

John Adler
University of Toronto
10 King's College Road,
Toronto, Canada
adler@eecg.toronto.edu

Andreas Veneris
University of Toronto
10 King's College Road,
Toronto, Canada
veneris@eecg.toronto.edu

ABSTRACT

Recent advances in automated debugging offer significant reductions in the manual effort required to localize RTL errors. These tools return relatively compact sets of RTL locations that can be potential error sources. However, once these locations are returned, the engineer still has to perform detailed analysis to discard irrelevant locations and identify the culprit. This process happens without any further guidance from the debugger. In this work, we perform a statistical analysis that exposes a significant discrepancy between RTL errors and other unrelated locations returned by these tools. The analysis is conducted on industrial designs and is based on metrics extracted from simulation. Our methodology determines that specific continuous distributions can effectively characterize the behavior of RTL errors. Using these well-defined metrics one can automate the process of further pruning the RTL locations returned by debuggers, effectively accelerating the localization of error sources.

Author Keywords

Failure Triage; Design Debugging; Satisfiability; Model Fitting

1. INTRODUCTION

Functional verification has grown to become the primary bottleneck in the modern design cycle [5]. Design debugging, the task of localizing and correcting an error once it has been revealed, can account for the majority of time spent in verification [4]. These tasks can be made even more complex when applied during regression testing, where thousands of input vectors are used to exercise a large portion of the design functionality. Traditionally, functional verification reveals an error when an observation value differs from the expected response. This provides an error trace that is then used in design debugging. Due to the large number of vectors and the potential co-existence of multiple design errors at this stage, hundreds of error traces may be exposed. Performing design debugging with hundreds of error traces is a daunting task.

Automated debugging tools [12, 7, 1, 2] exist to somewhat mitigate the substantial engineering resources required by these tasks. Given an error trace, a debugging tool returns a set of possible error sources in the RTL, known as the *suspect set*. It is then the task of an engineer to determine which of the suspects is the actual source of the error and implement the fix in a process known as detailed debug. These tools are exhaustive and, as such, the suspect set is guaranteed to contain the actual error source. However, it may include several other locations that are not responsible for the failure, but merely represent noise in the suspect set. This stifles attempts at automation, as it then becomes the responsibility of an engineer to investigate each suspect location and determine which one is truly the source of the error.

Toward the goal of easing this challenge, this paper presents a statistical analysis of the suspect sets returned by an automated debugging tool. Various features of the returned suspect locations are measured and compared in order to identify those that distinguish the actual error source from other locations. In practice, these results can then be used as the basis for a post-processing step that filters out noise from the suspect sets returned by an automated debugging tool. Doing so reduces the amount of time engineers must spend investigating irrelevant suspect locations.

Aside from automated design debugging, this filtering system has applications in automated failure triage as well. Automated failure triage seeks to group a set of error traces into bins, where all of the error traces in a given bin have the same root cause. Recent work in the field of failure triage applies similar feature-based techniques exclusively to the problem

SummerSim-SPECTS 2016 July 24-27 Montreal, Quebec, Canada

of failure binning [10]. Naturally, it is expected that such approaches can achieve more accurate results if the suspect sets contain less noise.

In greater detail, the presented analysis is as follows. Two specific simulation-based metrics are extracted for each of the suspects that are identified by automated debugging. These metrics, which we refer to as *temporal distance* and *excitation resistance*, quantify key behaviors observed by suspect locations. Particularly, temporal distance expresses the amount of time it takes for an error to propagate to an observation point, while excitation resistance offers a measure of difficulty in exciting the error. Given this empirical data, we then perform model fitting to explore known distributions that can explain the observations. The distributions are fitted across two disjoint sets of suspects. The first is the set of suspect locations that represent the actual error source, while the second is the set of suspect locations that are unrelated. It is found that these sets follow different distributions with respect to the metrics at hand. Effectively this discrepancy renders these metrics a proper means of distinguishing error sources from noise in the suspect set where they belong. Our findings can stimulate further use of these and similar other simulation metrics towards pruning a large portion of unrelated suspects, thereby saving substantial engineering resources for design debugging and improving the accuracy of failure triage.

The remainder of this paper is organized as follows. Section 2 presents background information relevant to the contributions and the simulation metrics at hand. Section 3 describes our methodology for evaluating the usefulness of these metrics for debugging and failure triage. Finally Section 4 presents the results of this evaluation and Section 5 concludes the work.

2. PRELIMINARIES

2.1 SAT-based Automated Debugging

The analysis presented here is based on suspect locations obtained using the Boolean Satisfiability (SAT)-based debugging algorithm of [12]. Given an error trace exposing some failure, the algorithm returns the set of all design locations where a fix can be made to correct the error (the *suspect set*). Traditionally, SAT-based debugging operates at the gate-level, returning each logic gate where a fix can be implemented. In this paper, it is assumed that the debugger operates at the RTL-level, so that suspects can be *e.g.,* modules, always-blocks, conditions, expressions, etc. Before introducing the algorithm itself, it is necessary to first define the notation, which will be used throughout this paper.

Given a sequential circuit C and an error trace of k clock cycles, the set of primary input, observation points, and state variables (flip flops) of C are denoted by $X = \{x_1, x_2, ..., x_{|X|}\}$, $Y = \{y_1, y_2, ..., y_{|Y|}\}$, and $V = \{v_1, v_2, ..., v_{|V|}\}$, respectively. Similarly, let V' denote the next-state variables (inputs to flip flops). The set of initial states for C is denoted as I. The transition relation of C is denoted as T. It is represented as a Boolean formula over the variables of X, Y, and V, such that $x \wedge y \wedge v \wedge T \wedge v'$ evaluates to 1 if and only if applying x to the primary input during state v causes the circuit to transition to state v' and produce output y. Let $\mathcal{X}^i$ denote the primary input values from the error trace in cycle i, and allow $\mathcal{Y}^i$ to denote the expected response at the observation points in cycle i. Let $B = \{b_1, ..., b_{|B|}\}$ denote the set of RTL blocks that may be returned as suspects, where b^i_j is the output of block j in cycle i. In an Iterative Logic Array (ILA) unrolling of the transition relation T, let T^i denote T with its primary input, state elements, and primary output indexed with i. This represents the i-th copy of the transition relation, or in other words, the i-th time-frame of the ILA.

The algorithm begins by constructing an enhanced transition relation T_{en} from the original transition relation of the circuit. It is constructed by the addition of a set of error-select lines $E = \{e_1, ..., e_{|B|}\}$. One error-select line is added per RTL block in B. If $e_i = 0$ then the behavior of block b_i is unchanged. Setting $e_i = 1$ replaces block output b_i with an unconstrained free variable. Subsequently, the enhanced transition relation is unrolled into an ILA representation with k time-frames. Additional constraints are constructed from the error trace to force the primary input to $\mathcal{X}^i$ in each time-frame i. Similar constraints are added to force the primary output to the expected response values $\mathcal{Y}^i$. The state elements in the first time-frame are constrained to a particular initial state using the formula for I. Finally, the number of simultaneously active error select lines is constrained to n using a cardinality constraint ϕ_n.

As such, for an error trace consisting of k time-frames, the problem is encoded as the following Boolean formula:

$$I \wedge \bigwedge_{i=1}^{k} \left(T^i_{en} \wedge \mathcal{X}^i \wedge \mathcal{Y}^i \right) \wedge \phi_n \tag{1}$$

Each satisfying assignment to the formula of Eq. 1 corresponds to an n-tuple of suspect locations where a fix can be implemented to make the circuit output the expected response for the particular error trace. An all-solutions SAT solver is used to find every such n-tuple of suspect locations. For the remainder of this paper, it is assumed that $n = 1$ and therefore the locations returned by the automated debugger are individual RTL blocks.

This approach is exhaustive, in that it finds every design location where a change can be implemented to correct the erroneous behavior. In this paper we assume that a single error is responsible for the behavior exposed in the error trace. Due to the exhaustive nature of the approach and this assumption, the returned suspect set is guaranteed to contain the actual design error. However, in practice it tends to include many other locations, as there are often many unrelated blocks where a change can be made to mask the failure. A *spurious suspect* refers to such a result *i.e.,* a suspect location returned by the debugger that is not the actual error source.

We refer to the set of suspect RTL blocks as suspect set $S = \{s_1, ..., s_{|S|}\}$, with $S \subseteq B$. Furthermore, SAT-based debugging allows us to retrive the exact time-frame in which

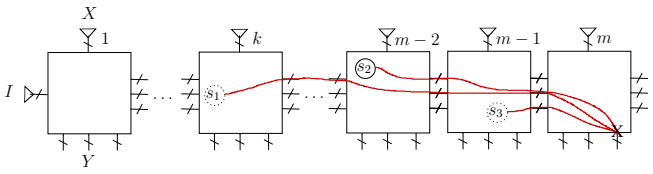


Figure 1: Error trace and suspect blocks

a block's error-select line is set to 1 for the first time. We refer to this time-frame as the excitation time t_i of suspect block s_i.

Example 1: *To demonstrate the above concepts consider an error trace, as depicted in Figure 1, where we show the sequential behavior of the circuit for that trace using its Iterative Logic Array (ILA) representation. In more detail, an error at block s_2 is excited in cycle $m-2$ and propagates to cause a mismatch at an observation point in cycle m. The generated error trace of length m is then passed to an automated debugger. The result is a suspect set $\mathcal{S}_1 = \{s_1, s_2, s_3\}$ of design components that can explain the wrong output. Suspects s_1, s_2 and s_3, excited in time-frame k, $m-2$ and $m-1$ respectively, along with their propagation paths are illustrated in Fig. 1. Note that the erroneous component is included in the set S as suspect s_2, thus suspects s_1 and s_3 are considered spurious.*

For various reasons, it is desirable to limit the number of spurious suspects returned. During detailed debug, the engineer must spend time investigating each of the suspects. Therefore fewer spurious suspects results in less wasted time. Additionally, having fewer of these spurious suspects is expected to be a major benefit to automated failure triage approaches as developed in [10]. The presence of fewer spurious suspects would accelerate the failure binning procedure, as there are fewer suspects to consider. Perhaps more importantly, it is expected to increase the accuracy of that method, as an indication of how likely a suspect is to be spurious can be used as an additional feature when prioritizing the failures.

2.2 Simulation Metrics

Of course there is no known method to determine which suspects are spurious with perfect accuracy. This subsection introduces particular simulation-based metrics that provide a starting point for a statistical analysis of suspects towards the goal of identifying features that distinguish spurious suspects from the actual error source. Similar metrics have also been proposed in [6] to enhance state-space search in verification. As shown in [10], metrics computed from simulation during regression runs can be used to rank suspect components. This suggests that these metrics follow different distribution models for spurious suspects and the actual error source, and can thus be used as a means of filtering out the former. The two main metrics that are the focus of this paper are, namely, *temporal distance* and *excitation resistance*.

Excitation resistance relates to the number of times signals toggle in a circuit or a portion of the circuit (*e.g.,* an RTL block). Given an error trace with k clock cycles, a suspect block s_i, and a clock cycle $0 < j < k$, we define the *toggle rate* $f_j(s_i)$ of s_i in cycle j with respect to an error trace as follows:

$$f_j(s_i) = \frac{< \text{\# of inputs to } s_i \text{ that toggled in cycle j} >}{< \text{\# of inputs to } s_i >} \quad (2)$$

That is, the toggle rate is the fraction of the inputs that toggled in the relevant clock cycle. Then, the excitation resistance $r(s_i)$ of a suspect RTL block s_i is defined as the average of the toggle rate for the cycles between the initial states and the suspect's excitation time t_i:

$$r(s_i) = \frac{1}{t_i} \sum_{j=1}^{j=t_i} f_j(s_i) \quad (3)$$

Note that excitation resistance takes real values in the range $[0 \ldots 1]$. Intuitively, the smaller the excitation resistance is the easier it is for an error in the relevant RTL block to be excited by the test vectors and eventually propagate to the observation point.

Temporal distance relates to the amount of time between the excitation of the suspect and the observation of the failure. Specifically, it refers to the number of cycles between the excitation of the suspect and the observation of the failure divided by the length of the error trace. As such, the temporal distance $d(s_i)$ of suspect s_i with excitation time t_i in an error trace of length k cycles is given as follows:

$$d(s_i) = \frac{k - t_i}{k} \quad (4)$$

Just like excitation resistance, temporal distance also takes real values in the range $[0 \ldots 1]$. Here, the smaller the distance is, the easier it is for the error excited at the relevant RTL block to propagate at the observation point and justify the mismatch.

The goal of this paper henceforth is to perform a probabilistic analysis that will reveal to what extent spurious suspects and actual error sources exhibit different behavior with respect to the above metrics. If an apparent separability is justified probabilistically, then these metrics can be used to rank suspects by their likelihood of being the real error source, thereby reducing debug time. They additionally could provide guidance to automated failure triage tools and other approaches that use the results of a debugging tool in a similar manner. All discussions in prior art regarding these metrics are purely intuition-driven. Particularly, the existence of underlying distribution models for these metrics has not been explored, although it is hinted to some extent by experimental results.

3. ERROR BEHAVIOR MODELS

From a probabilistic view, classical SAT-based debugging assigns zero probability of being a design error to those RTL blocks that do not appear in some suspect set. This effectively reduces the search space that the engineer needs to analyze in

order to come up with a rectification. However, for those RTL blocks that appear as suspects, the debugger has no mechanics available to filter them based on how likely they are to be design errors or merely equivalent explanations of the erroneous circuit behavior. This task is manual and left as a whole to the engineer.

As such, classical debugging treats all suspects in a suspect set as equiprobable in being the culprit of a failure. One can therefore say that for any suspect $s_i \in S$ the probability of being a design error is $\frac{1}{|S|}$. This implied uniform distribution may in fact be a sufficient assumption, if we would generally expect design errors to have a random nature (random source code mutations, random stack-at-faults etc). In reality, though, design errors are human-introduced. This intuitively implies that there must exist some bias in the behavior of such design errors that separates them from random ones. If that is the case, then the uniform distribution assumed in classical debugging fails.

Identifying whether such bias exists reduces to the task of determining suspect features that can be expected to be seen in a design error but not in spurious suspects or vice-versa. Again, from a probabilistic standpoint, this means that these features must follow significantly different distribution models over design errors and spurious suspects. This Section describes our methodology of learning the underlying distribution models of temporal distance and excitation resistance for design errors and contrasting them with those of spurious suspects.

3.1 Theoretical Models

In [11] the authors offer a probabilistic model for design error behavior, an extended version of which is forms the basis of failure triage methods in [10]. The model is built in a manner that embeds the concepts of excitation probability and propagation probability into a single expression for the probability of a design location causing a failure at some observation point. We restate it below.

Assuming that an error exists in the design and that simulation starts at cycle 1, let ex be the probability that the error is excited at cycle i. Also, let pr be the probability of the error propagating from cycle i to cycle $i+1$, and ob be the probability of observing a failure at some observation point at cycle i given that the error has propagated to that cycle. Also assume that the input vector sequences are temporally independent and stationary random sequences. Then, the probability $p_{m|n}$ of observing the first failure at cycle m given that the error is excited at cycle n is:

$$p_{m|n} = (1-ex)^{n-1} \times ex \times pr^{m-n} \times (1-ob)^{m-n} \times ob \quad (5)$$

From the above expression, one can make two important observations: (a) the absolute distance $m-n$ from the excitation cycle n to the observation cycle m has a decaying exponential effect on $p_{m|n}$, and (b) for a fixed distance $m-n$, $p_{m|n}$ decays towards 0 as the excitation probability ex tends to its limits 0 and 1. Further for a varying distance $m-n$ we are interested in values for ex where $p_{m|n}$ obtains its maxima.

Since the absolute distance $m-n$ is directly proportional to temporal distance in Eq.(4) and the excitation probability ex can characterize the excitation resistance in Eq.(3), we can hypothesize that the distributions of these two metrics will be similar to the probability density function of pr and ex in Eq.(5). Therefore our hypothesis is that temporal distance distributes exponentially across design errors, while the Weibull distribution arises as a continuous characterization of excitation resistance. The latter is justified by extreme value theory, as we are interested in maxima of $p_{m|n}$ and $p_{m|n}$ has a finite lower limit [3]. Broadly speaking, our expectation is that, compared to spurious suspects, human introduced design errors are easier to excite (a low mean excitation resistance) and easier to propagate to observation points (low mean temporal distance), with the latter also being the motivating factor behind Bounded Model Debugging [11]. This is not to say that all human-introduced errors are easy to excite and propagate. Rather, in early regression verification stages we expect that a significant fraction of human-introduced errors are of such nature, due to fast prototyping. We discuss the special case of deep bugs later in the empirical evaluation Section.

Testing our hypothesis involves the following steps: (a) generate proper datasets, that is, sets of suspect locations, both actual design errors and spurious suspects and measure the relevant metrics presented in Section 2, (b) aggregate this information and measure the frequency in which design errors and spurious suspects appear to have specific ranges of values for the aforementioned metrics, and finally (c) conduct a model fitting (parameter learning) process across various candidate distributions to test whether the expected distributions explain the data well.

3.2 Data Generation

In order to learn models that will be representative of ground truth with confidence, we need relatively large datasets. Our method is to manually inject multiple design errors in a set of Verilog/VHDL designs, perform regression testing/simulation that will expose multiple failures and then run multiple SAT-based debugging sessions to generate enough suspects for our empirical evaluation.

For each design, a set of different errors is manually injected by modifying the RTL description. The injected errors resemble typical human-introduced errors that are observed in the industry. Examples of such design errors include missing pipeline stages, incorrect operators in expressions, bad stimulus, complemented conditions in if-statements, incorrect state transitions, etc.

For each simulation run that exposes one or more failures, SAT-based debugging is performed at the RTL block level, and suspect sets are collected. A suspect component can potentially be returned by various debugging runs and belong to suspect sets that justify various failures. Each occurrence of a suspect is accompanied by potentially different values for temporal distance and excitation resistance. As such, even if

a suspect's occurrence corresponds to the same RTL block, we treat it as a separate suspect-datapoint for our empirical evaluation. This is also necessary because across different design failures the same suspect RTL block can be the actual design error or a spurious suspect.

3.3 Data Aggregation

Our goal is to create a histogram $\mathbf{h^d_{err}}$ that captures the number of suspects-errors that have temporal distance falling within specific intervals of length 0.1 from $[0.0 \dots 0.1)$ to $[0.9 \dots 1.0]$. Similarly, we create histogram $\mathbf{h^d_{spur}}$ for the temporal distance of spurious suspects, $\mathbf{h^r_{err}}$ for the excitation resistance of suspects-errors, and $\mathbf{h^r_{spur}}$ for the excitation resistance of spurious suspects. As a representative sample of each interval we take the mean of the interval itself (i.e 0.05 represents interval $[0.0 \dots 0.1)$).

Measurement of the relevant simulation data can be done via commercial simulators, and the computation of the metrics per suspect component is polynomial in the length of the error trace.

3.4 Model Fitting

As mentioned previously, our expectation is that Weibull explains well our empirical data when it comes to excitation resistance. The pdf of Weibull is given below:

$$f_{WB}(x;\lambda,\kappa) = \frac{\kappa}{\lambda}(\frac{x}{\lambda})^{\kappa-1}e^{-(\frac{x}{\lambda})^{\kappa}} \tag{6}$$

where $\lambda > 0$ and $\kappa > 0$ are scale and shape parameters, respectively.

However, for the sake of rigor it is important to further test the applicability of alternative distributions that also offer at least exponentially decreasing tails. Our other candidate distributions are the Log-normal and Gamma, whose pdfs are respectively given below:

$$f_{LN}(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}}e^{-\frac{(\log x-\mu)^2}{2\sigma^2}} \tag{7}$$

$$f_{G}(x;\theta,\eta) = \frac{1}{\theta^{\eta}}\frac{1}{\Gamma(\eta)}x^{\eta-1}e^{\frac{-x}{\theta}} \tag{8}$$

where $\Gamma(\cdot)$ is the gamma function and $\theta > 0$ and $\eta > 0$ are scale and shape parameters, respectively.

When it comes to temporal distance we solely fit the exponential distribution, as it naturally arises from Eq.(5).

In order to fit the above continuous distributions, we apply multinomial maximum likelihood [8] on the produced histograms. To evaluate the quality of fit we use the Kullback-Leibler (KL) divergence, as opposed to the χ^2 test which underestimates goodness of fit when it comes to histograms of non-categorial data. The KL divergence between empirical data $\mathbf{h} \in \{\mathbf{h^r_{err}}, \mathbf{h^d_{err}}\}$ and fitted model $\mathbf{f} \in \{f_{WB}, f_{LN}, f_G, f_{exp}\}$ is given below:

Table 1: Dataset Statistics

Ckt.	# gates	# vectors	# failures	# errors	# spurious
vga	72292	25206	361	42	561
fpu	83303	20094	312	37	477
spi	1724	5019	103	21	231
mem_ctrl	46767	13370	248	34	360

$$D_{KL}(\mathbf{h}|\mathbf{f}) = \sum_d h_d \log \frac{h_d}{f_d} \tag{9}$$

where f_{exp} is the exponential distribution, and d refers to sampling points used for evaluating the goodness of fit. KL divergence measures information loss when $\mathbf{h}$ is represented by $\mathbf{f}$. Thus, the lower the KL divergence is, the better the model explains the data.

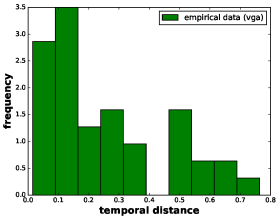
Finally, once we find which distribution offers the lowest KL divergence for suspects-errors we use that same model to fit our empirical data for spurious suspects. From that fitting process we obtain the KL divergence as well. Our goal is to show that the model that explains well the metrics for suspects-errors does not offer a good fit for spurious suspects. One can also view this process as having suspect-errors forming the training set, spurious suspects forming the validation set, and demonstrating that the characterization of the training set is unfit for the validation set. This, in turn, exposes a separability between these two entities, which renders the metrics at hand useful for filtering suspects in debugging and failure triage.

4. EMPIRICAL EVALUATION

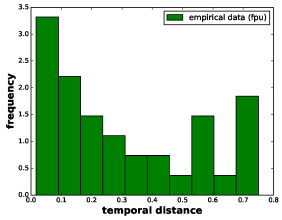
This Section presents our experimental results. The SAT-based automated debugger used for data generation is implemented in C++ based on [12]. A platform coded in Python is developed to parse simulation logs, collect and aggregate all relevant data, and perform model fitting as described in Section 3. Four *OpenCores* [9] designs are used for the evaluation (`vga`, `fpu`, `spi` and `mem_ctrl`).

For each design, we use a pre-generated set of test sequences from the test bench that accompanies each design. Each regression run involves hundreds to thousands of diagnose input vectors. For the purpose of capturing failures we use end-to-end "golden model" checkers that compare the expected value for various operations, exception checkers and various existing and newly written assertions throughout the designs.

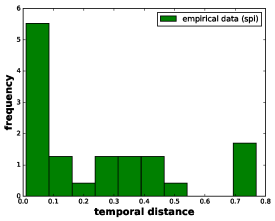
Table 1 summarizes dataset information and statistics for all simulation and debugging sessions per design. From left to right, columns show the circuit name and number of gates, the number of diagnose vectors, the number of injected RTL errors (which are also returned as suspects by SAT-based debugging), the number of observed failures, which is also the number of failing diagnose vectors, and the number of distinct spurious suspects generated by SAT-based debugging.



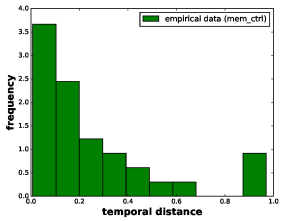
(a) vga histogram



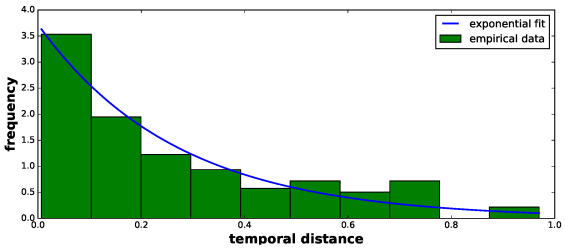
(b) fpu histogram



(c) spi histogram

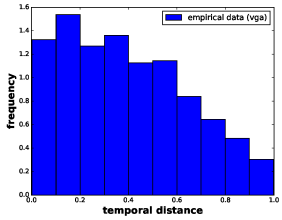


(d) mem_ctrl histogram

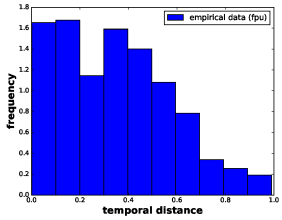


(e) Exponential fit on aggregate histogram $\mathbf{h^d_{err}}$

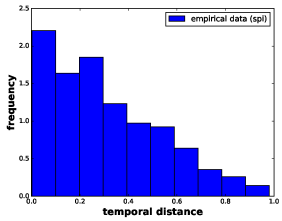
Figure 2: Temporal distance histograms for suspects-errors



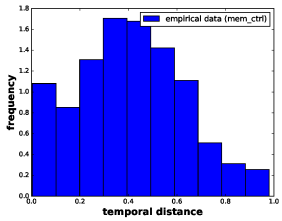
(a) vga histogram



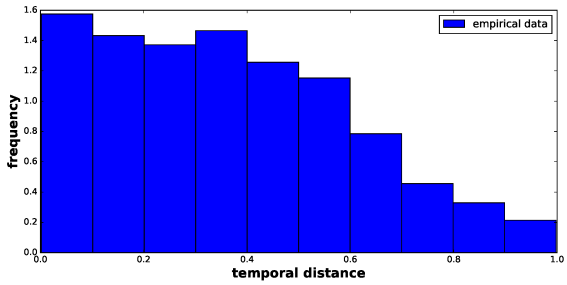
(b) fpu histogram



(c) spi histogram



(d) mem_ctrl histogram



(e) Aggregate histogram $\mathbf{h^d_{spur}}$

Figure 3: Temporal distance histograms for spurious suspects

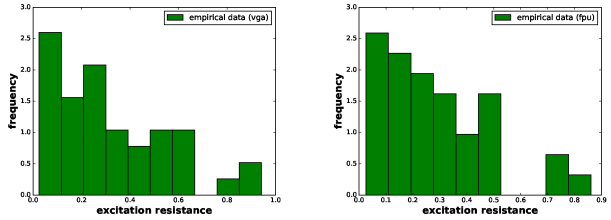
Finally, note that in our experimental setting a single injected design error may be the root cause of multiple design failures.

Figure 2 shows the histograms of temporal distance that we obtain for suspects-errors. Particularly, Figures 2.(a), 2.(b), 2.(c) and 2.(d) illustrate the normalized frequencies of suspects-errors with temporal distances that fall within bins of length 0.1, for the circuits `vga`, `fpu`, `spi` and `mem_ctrl`, respectively. Figure 2.(e) shows the aggregate histogram across all circuits ($\mathbf{h^d_{err}}$) and the exponential fit that is produced by multinomial maximum likelihood. Visually it appears that the distribution of temporal distance over suspects-errors follows closely the exponential model as expected, with `fpu` being the only histogram with some discrepancy, mainly due to variable latency.

Similarly, Figure 3, shows plots of histograms for spurious suspects per circuit, as well as the aggregate histogram $\mathbf{h^d_{spur}}$. From these histograms one can see that temporal distance distributes in a different manner, with a tail that does not decrease exponentially and a large portion of the frequencies almost evenly distributed within the range $[0.0 \ldots 0.6]$. Interestingly, however, the majority of spurious suspects have relatively low temporal distance. To some extent this bias can be justified by the fact that many spurious suspects are temporally related to the actual design error when they reside in its fan-out or fan-in cone. Nevertheless, the divergence between $\mathbf{h^d_{err}}$ and $\mathbf{h^d_{spur}}$ is clear, as it will also be shown later in this Section.

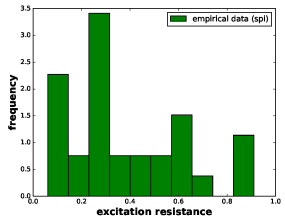
With respect to excitation resistance, the histograms for suspects-errors and spurious suspects are shown in Figure 4 and Figure 5, respectively. The aggregate histograms $\mathbf{h^r_{err}}$ and $\mathbf{h^r_{spur}}$ can be seen in Figure 4.(e) and Figure 5.(e). Figure 4.(e) also illustrates the Weibull fit on our empirical data for suspects-errors. The only discrepancy was observed in Figure 4.(c) for the `spi` histogram, mainly due to the small size of the design in terms of the number of gates and flip-flops. In such "shallow" designs the error trace length is relatively small and excitation resistance may not follow a clear pattern.

In this experimental setting, one can see an apparent divergence between the distributions. Human-introduced errors appear to follow a Weibull model when it comes to excitation resistance, and in fact with a relatively low mean. On the other hand, spurious suspects seem to concentrate around arbitrary modes, and not necessarily close to small excitation resistance values. This structured concentration can potentially be explained by the fact that specific modules in the design under test are rigorously exercised by test vectors and probably by the fact that some of the tests are tailored to exercise specific functionality.

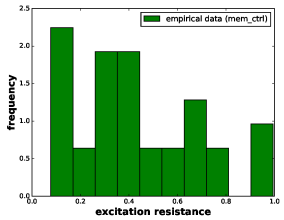


(a) vga histogram

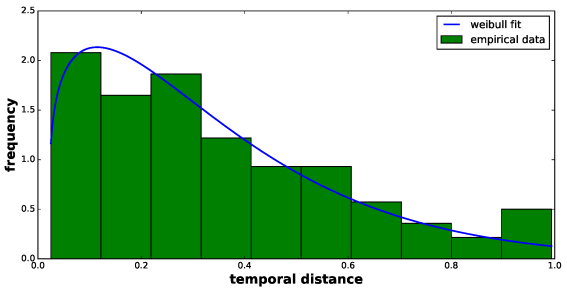
(b) fpu histogram



(c) spi histogram

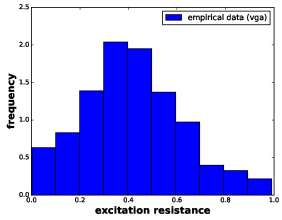


(d) mem_ctrl histogram

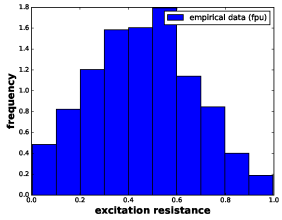


(e) Weibull fit on aggregate histogram $\mathbf{h^r_{err}}$

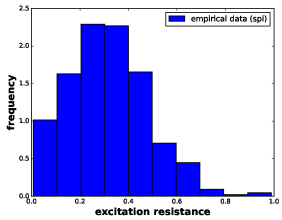
Figure 4: Excitation resistance histograms for suspects-errors



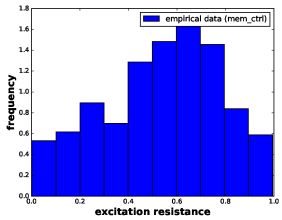
(a) vga histogram



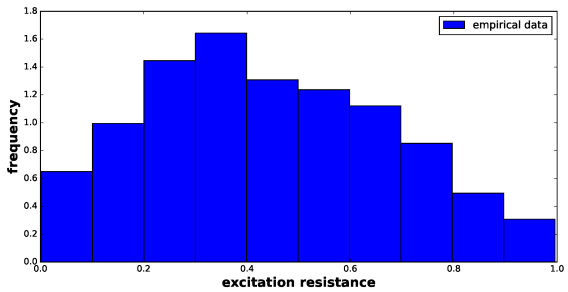
(b) fpu histogram



(c) spi histogram



(d) mem_ctrl histogram



(e) Aggregate histogram $\mathbf{h^r_{spur}}$

Figure 5: Excitation resistance histograms for spurious suspects

It should also be stressed that in our analysis we also introduce RTL errors that are relatively obscure and thus harder to excite and propagate to observation points. These often appear in the long tail of the shown distributions. This fact implies an inherent uncertainty when using the proposed metrics to identify such errors as non-spurious. Nevertheless, the majority of design errors in regression verification can be accurately distinguished, something that can lead to effective automated pruning methodologies preceding a deeper detailed analysis from engineers to locate and fix deeper bugs.

Goodness of fit results are shown in Table 2, where we report the D_{KL} divergence of the model fitting process for the histograms that are produced. Recall that lower divergence means better fit. Column 1 corresponds to the model to be fitted each time, columns 2 to 5 show divergence values for the aggregate histogrms $\mathbf{h^d_{err}}$, $\mathbf{h^d_{spur}}$, $\mathbf{h^r_{err}}$ and $\mathbf{h^r_{spur}}$, respectively.

Table 2: Goodness of fit

model	D_{KL} divergence			
	$\mathbf{h^d_{err}}$	$\mathbf{h^d_{spur}}$	$\mathbf{h^r_{err}}$	$\mathbf{h^r_{spur}}$
f_{exp}	**0.0623**	0.0978	N/A	N/A
f_{WB}	N/A	N/A	**0.0439**	0.2634
f_G	N/A	N/A	0.0638	0.1047
f_{LN}	N/A	N/A	0.0496	0.1992

These results imply that the two metrics described in this work offer a robust model upon which effective suspect filtering systems can be built for the purposes of functional debugging and failure triage. Moreover, they provide an empirical explanation of why recent failure triage frameworks that use these (or a variation of these) metrics accomplish high levels of accuracy.

5. CONCLUSION AND FUTURE DIRECTIONS

This work provides a probabilistic and statistical analysis of simulation metrics that can be used in automated RTL debugging and triage. It exposes a separability in the behavior of design errors compared to this of spurious suspects that are returned by formal debugging tools. This establishes a potential for additional use of these metrics that offers an extra layer of refining once formal tools identify potential error locations; an advance that can significantly boost the accuracy of these tools. A study on a much larger sample of designs

will also verify the applicability of the proposed metrics on a broader level.

Finally, additional metrics should also be explored in the future, especially for larger designs (i.e in microprocessor and System-on-Chip verification). At a larger design scale the underlying models may differ and may need to be learned using a similar method to the one we propose in this work.

REFERENCES

1. Chang, K.-H., Markov, I., and Bertacco, V. Automating post-silicon debugging and repair. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on* (Nov 2007), 91–98.
2. Fey, G., Staber, S., Bloem, R., and Drechsler, R. Automatic fault localization for property checking. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 27*, 6 (June 2008), 1138–1149.
3. Fisher, R. A., and Tippett, L. H. C. Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Mathematical Proceedings of the Cambridge Philosophical Society 24* (4 1928), 180–190.
4. Foster, H. Assertion-based verification: Industry myths to realities (invited tutorial). In *Intl Conference on Computer-Aided Verification (CAV)* (2008), 5–10.
5. Foster, H. From volume to velocity: The transforming landscape in function verification. In *Design Verification Conference* (2011).
6. Ganai, M. K., and Aziz, A. Rarity based guided state space search. In *Proceedings of the 11th Great Lakes Symposium on VLSI*, GLSVLSI '01, ACM (New York, NY, USA, 2001), 97–102.
7. Huang, S.-Y., and Cheng, K.-T. *Formal Equivalence Checking and Design DeBugging*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
8. Jennrich, R., and Moore, R. Maximum likelihood estimation by means of nonlinear least squares. *ETS Research Bulletin Series i*, 36 (1975).
9. OpenCores.org. http://www.opencores.org, 2007.
10. Poulos, Z., and Veneris, A. Clustering-based failure triage for rtl regression debugging. In *Proc. IEEE International Test Conference* (2014), 1–10.
11. Safarpour, S., Veneris, A., and Najm, F. Managing verification error traces with bounded model debugging. In *ASP Design Automation Conf.* (2010), 601—606.
12. Smith, A., Veneris, A., Ali, M. F., and Viglas, A. Fault diagnosis and logic debugging using boolean satisfiability. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst 24*, 10 (Oct. 2005), 1606–1621.