Test, Reliability and Functional Safety Trends for Automotive System-on-Chip

Test, Reliability and Functional Safety Trends for Automotive System-on-Chip

F. Angione[§], D. Appello[‡], J. Aribido[*], J. Athavale[*], N. Bellarmino[§], P. Bernardi[§], R. Cantoro[§], C. De Sio[§], T. Foscale[§], G. Gavarini[§], J. Guerrero[§], M. Huch[†], G. Iaria[§], T. Kilian[†¶], R. Mariani[*], R. Martone[§], A. Ruospo[§], E. Sanchez[§], U. Schlichtmann[¶], G. Squillero[§], M. Sonza Reorda[§], L. Sterpone[§], V. Tancorre[‡], R. Ugioli[‡]
[*]NVIDIA, US,[†]Infineon Technologies, DE, [‡]STMicroelectronics, IT, [§]Politecnico di Torino, IT,
[¶]Technical University of Munich, DE

***Abstract*—This paper encompasses three contributions by industry professionals and university researchers. The contributions describe different trends in automotive products, including both manufacturing test and run-time reliability strategies. The subjects considered in this session deal with critical factors, from optimizing the final test before shipment to market to in-field reliability during operative life.**

***Index Terms*—DNNs reliability, Inter-wafer performance variation estimation, SLT-BI automatic test equipment.**

I. Introduction

Nowadays, major trends for Automotive Systems-on-Chip are the extreme complexity and the massive usage in safety-critical environments. Such a combination of leading factors generates very strong quality requirements and it triggers crucial objectives to reach ultra-dependable vehicle electronics.

Mitigation of potentially harmful behavior deviations due to transient or permanent events is a must-have to ensure the functional safety. A recurrent scenario concerns the reliability of neural networks and how they could self-mitigate the effects of a circuit misbehaviour. Section II investigates on Deep Neural Networks intrinsic robustness against permanent faults.

To stably reach the expected performance from manufactured chips is another consequence of the major trends. This is not easy to ensure because nanotechnology suffers from process variation. Section III illustrates how to use Machine Learning applications to predict wafer to wafer performance variation.

Section IV concludes the paper topics looking into test equipment that could revolve Automotive tester scenario. A promising solution permits to merge System-level-Test and Burn-In steps into a single one, by enabling a micro-controller to control the Design-for-Testability features of the device under test other than establishing a functional cooperation.

II. Self-mitigation properties on Image segmentation Neural Networks

The adoption of artificial intelligence (AI)-based algorithms is gaining increasing attention in the automotive area, particularly for autonomous vehicles. Indeed, due to their near-human computational capabilities in tasks like image recognition, Deep Neural Networks (DNNs) are considered appealing solutions in many applications, also in safety-critical areas. However, to safely deploy them in human contexts, it is crucial to evaluate and ensure their functional safety and reliability, such as compliance with the ISO 26262 auto safety standard. To handle the complexity of the driving environment, autonomous vehicles are powered with deep learning algorithms which have learned how to manage situations, how to make decisions based on input stimuli, how to recognize pedestrians or traffic signs. In the last few decades, the research community has invested a lot of effort in assessing and guaranteeing the reliability of DNN models and DNN-based systems. Particularly, understanding their vulnerability to random hardware faults has become of paramount importance. However, recently proposed neural networks feature novel architectural details, such as batch normalization or skip connections, which require deepening the reliability studies done on older neural networks. The principal intent of this work is to study the reliability of a neural network used for the image segmentation task: its vulnerabilities and strengths are highlighted by means of fault injection campaigns. Experimental results demonstrate that the presence of the Rectified Linear Unit (ReLU) activation function halves the percentage of critical faults in layers. It means that this activation function is not only important for its computational role of the nonlinearity (the ReLU supports accurate context-dependent transformations and its removal substantially impairs model performance [1]) but also for the fault tolerance perspective. In this light, the results of further reliability assessment studies on neural networks used for the classification task are shown to confirm the self-mitigation property of the ReLU activation function. To conclude, the fault tolerance of skip connections is investigated: indeed, they introduce a double path for faults, which might be concerning for the resilience of the neural network.

A. Deep Neural Networks for Image Segmentation

Image segmentation neural networks are used to partition an image into different segments (or classes). They can be distinguished in two main categories, depending on the number of segments they have been purposely trained to segment. We refer to *semantic* segmentation if there are only two possible segments to partition, otherwise, when multiple regions must be identified, we refer to *instance* segmentation. A DNN used for this task takes as input an image and returns as output a segmentation mask, where each pixel of the input is assigned to a given segment. U-Net [2] is a semantic segmentation DNN developed in the field of biomedical image segmentation. It features an encoder-decoder architecture, and it is composed of four consecutive encoder blocks, a bottleneck and four

consecutive decoders blocks. The encoders, each containing two convolutional layers and a max-pooling, extract the latent representation from the input image. This information is then used by the decoders, each one composed of a transpose convolution and two convolutional layers, as a starting point to build the segmentation mask. Noteworthy, the decoders receive high-level information of the input image though direct connections with the encoders, called *skip connections*.

B. Experimental Analysis

Reliability assessments on DNNs are performed through software fault injection campaigns. Under the single fault assumption, permanent faults are injected in static parameters, i.e., weights, and the faulty results are compared with the golden ones. To measure the effects of the injected stuck-at faults, a common evaluation metric for semantic image segmentation is the mean Intersection over Union (m-IoU). For an individual class, the IoU is defined as the intersection between the ground truth area of that class and the segmentation mask area of the same class, over the union of the twos. For a segmentation task, the m-IoU is the defined as the mean IoU for all the possible classes. Comparing the m-IoU of a run affected by fault with the same metric of the equivalent golden run, we can measure the impact of the fault. Therefore, we can classify a fault based on the variation in m-IoU (expressed as a percentage) between the faulty and the golden run. A fault can be:

- **Masked**: No difference is observed between the faulty DNN and the golden one.
- **Observed**: The m-IoU of the golden network and the faulty DNN are different.

Observed Faults can be further classified as **Accept** if the m-IoU varies less than 5 %, **Warning** if there is a difference in m-IoU between 5 % and 10 %, and **Critical** if the variation is more than 10 %. The segmentation neural network under assessment is U-Net, a 27-layer convolutional neural network for biomedical image segmentation [2]. In more details, we used a freely available pre-trained version from PyTorch. Static parameters are internally represented as 32-bit single precision floating point numbers and account for a total of 7,757,153 weights. Evaluating exhaustively all the possible stuck-at faults would have been out of the timing and computational possibilities. To deal with this complexity, typically a statistical fault injection is performed. Initially, we set the number of fault injections by following the methodology proposed in [3] with an error margin of 1% and a confidence level of 99.80%: a total of 23,870 fault injections have been performed, corresponding to the 0.004% of the total possible stuck-at faults. However, the very low number of injected faults in some layers was a limiting factor to be able to draw meaningful conclusions. As a consequence, we performed a further fault injection campaign by injecting the same amount of faults in every layer (excluding the max pooling ones), for a total of 168,935 injected stuck-at faults.

A similar procedure was used to perform reliability assessments on DNNs performing image classification. Three different CNNs performing image classification were analysed: LeNet, AlexNet, and Darknet19. The LeNet model classifies images taken from the MNIST dataset, while AlexNet and Darknet19 from the ImageNet one. LeNet, AlexNet, and Darknet19 include about 3M, 62M, and 20M parameters, respectively, represented in a 32-bit floating point representation. As in the U-Net case, running exhaustive fault simulations was unfeasible. Therefore, a statistical fault injection was performed by considering a confidence level of 99% and an error margin of 1%, for each layer.

C. Experimental Results

The outcomes of the permanent fault injection on U-Net show that, overall, the 89.87 % of fault is classified as Masked and the 10.13 % as Observed, of which the 6.18 % is classified as Accept, the 0.05 % as Warning, and the 3.90 % as Critical. Going in more details, it turned out that the 99.21 % of critical faults were due to stuck-at-1s affecting the 30th bit of the U-Net weights (i.e., the most significant bit of the exponent part of the 32-bit floating point data representation). This is in line with the literature [4]. In Table I we report the U-Net experimental results divided by layers (the same number of faults is injected in each of them). The m-IoU percentage is given for every above-mentioned category. The first observation that can be drawn is related to the different per-layer resilience: the first and the last layers seem to be less robust with respect to the middle ones. As stated, the U-Net features an encoder-decoder architecture: layers from 0 to 7 belong to the encoder; layers from 8 to 10 hold the latent information of the DNN; layers from 11 to 22 constitute the decoder block. The percentage of *Masked* faults increases as we get deeper into the intermediate layers.

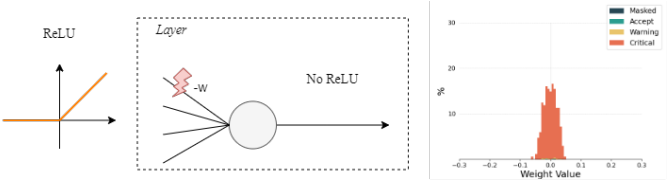


Fig. 1: First Scenario

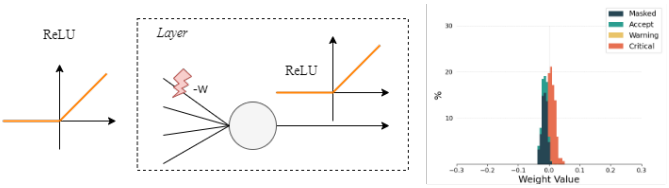


Fig. 2: Second Scenario

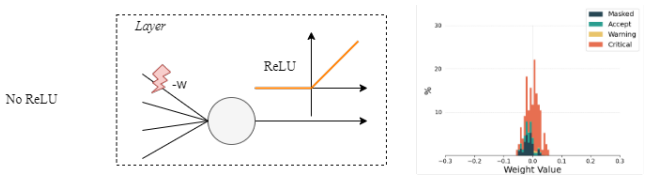


Fig. 3: Third Scenario

The second important observation refers to the class of *Critical* faults: the highest percentage of critical faults is

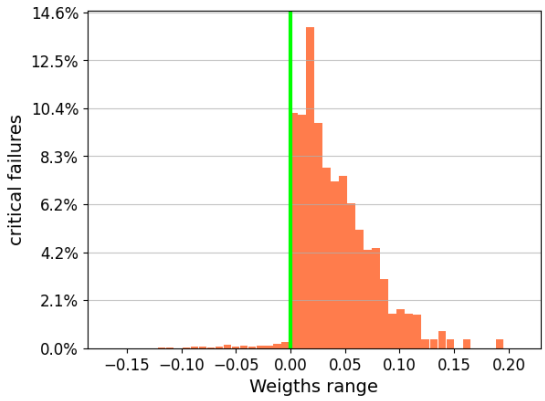
evidenced in layers *L0, L10, L13, L16*, and *L19*. They exhibit double as many critical failures as the others on average. The investigation of those scenarios led us to an interesting consideration about the role of the ReLU activation function (Eq. 1) which is applied after the neuronal computations, almost after every convolutional layers.

$$f(x) = max(0, x) \tag{1}$$

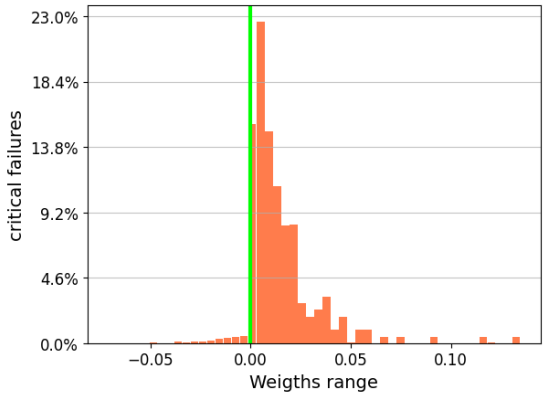
Nevertheless, particularly in modern DNN architectures, there are specific layers applying deconvolution or transpose convolution that are used to perform upsampling. In many cases, they do not apply the ReLU activation function after the deconvolution operations. They are particularly useful to decode the latent information stored in the bottleneck sections; and therefore are part of the decoder section in U-Net. In our U-Net architecture, layers *L10, L13, L16*, and *L19* are upsampling layers. The lack of the ReLU after their deconvolution operations do not prevent the propagation of stuck-at faults on negative weights to the output, resulting in an increased rate of critical faults. Indeed, negative values are cut away from the ReLU (Eq. 1), and consequently also faults on negative weights, leading to a negative value at the output of the neuron. We summarize and clarify the observed scenarios in Fig. 1, 2, and 3. Hereinafter, motivated by the above-mentioned source of critical faults, the three graphs shown in Fig. 1, 2, and 3 plot the effect of the most critical one: a stuck-at-1 affecting bit 30th of the weight.

The first scenario covers the U-Net upsampling layers, where the deconvolution operations do not go through the ReLU activation function, neither other types. As shown in Fig. 1, stuck-at-1s affecting both negative or positive weights on bit 30th are propagated to the output as critical faults. In this figure, the upsampling layer *L10* is illustrated. The same reasoning applies to the other U-Net upsampling layers. On the contrary, Fig. 2 shows the self-mitigating effects of the ReLU. In layers adopting such activation function at their output, and (importantly) preceded by layers using ReLU as well, all stuck-at-1s affecting negative weights on bit 30th are masked (they are cut away from the computation). The illustrated weight distribution is related to the layer *L8*, but the same trend is observed for similar-feature layers in U-Net.

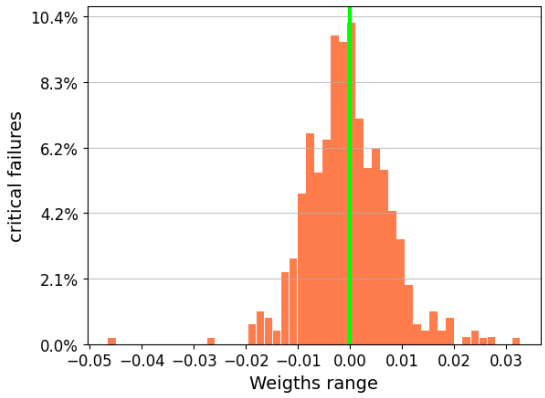
Worthy of note is the last scenario (Fig. 3). As evident, the target layer is provided with the ReLU activation function. One could think that all faults on negative weights are masked. However, the weight distribution shows that also stuck-at-1s on negative weights are propagated to the output as critical faults (orange bars in the negative x-axis). The reason is straightforward: the previous layer is devoid of the ReLU. It means that the target layers receives negative inputs as well, which, multiplied by negative weights (faulty in our case) become positive (faulty) values, and are propagated to the output. Values illustrated in the graph of Fig. 3 refer to layer *L17* and, generally, correspond to all the U-Net layers following the upsampling ones. As also shown in Table I, all those layers (*L11, L14, L17*), and *L20* have a m-IoU% of critical faults higher than average (but lower than upsampling layers). This third scenario is also applicable to *L0*, where inputs are standardized around the mean, and thus can be both negative and positive.



(a) LeNet



(b) AlexNet



(c) Darknet19

Fig. 4: Impact of the ReLU activation function on the propagation of critical faults in DNN classifiers.

Regarding the classification DNNs, similar outcomes are observed: the 30th bit is the most critical one, producing an accuracy degradation of up to 70% for Darknet19 and up to 25% for AlexNet. Noteworthy, it is important to mention the ReLU role also in these DNNs: faults affecting the positive weights generate a significant amount of wrong classification results than the ones affecting the negative parameters. This can be explained by the ReLU (Eq. 1), which blocks any negative operation result in the neurons but allows the propagation of positive values. Fig. 4 shows the percentage of wrong classification results produced by critical faults in the weights of different neural networks. Clearly, the CNNs that

TABLE I: U-Net fault injection results

IoU [%]	Layer																						
	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20	L21	L22
Masked	69.68	84.28	83.15	86.36	87.20	89.01	93.20	94.66	95.19	95.76	92.31	94.21	95.25	91.43	91.44	92.30	89.63	90.27	92.84	89.25	92.12	93.70	76.58
Accept	24.06	12.44	13.13	10.68	9.40	7.55	3.88	2.54	1.62	0.91	0.78	1.21	1.92	2.60	4.99	4.64	4.89	5.43	4.64	5.04	3.82	4.14	20.75
Warning	0.25	0.07	0.04	0.04	0.03	0.02	0.00	0.01	0.01	0.01	0.07	0.03	0.02	0.07	0.04	0.04	0.12	0.06	0.04	0.06	0.10	0.04	0.07
Critical	**6.01**	3.21	3.68	2.92	3.37	3.42	2.92	2.80	3.18	3.31	**6.85**	4.54	2.81	**5.90**	3.54	3.02	**5.36**	4.24	2.48	**5.64**	3.96	2.12	2.60

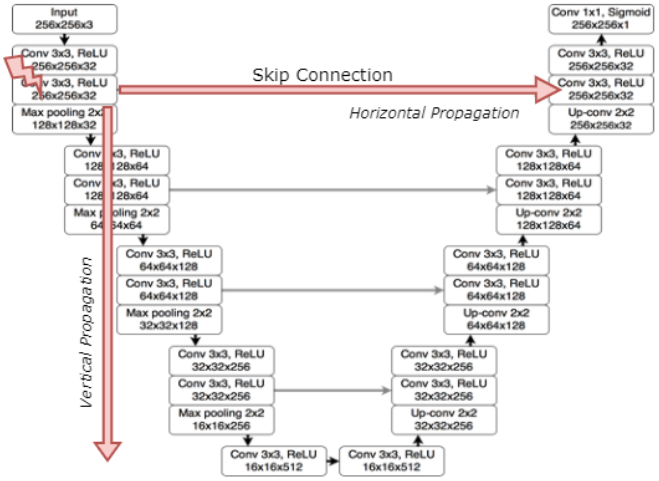


Fig. 5: U-Net Skip Connections and Fault Propagation.

incorporate the ReLU activation function (Fig. 4a, Fig. 4b) generate a high rate of wrong predictions for faults affecting positive weights and are resilient to faults that cause large negative parameters. In both cases, more than 97% of the incorrect classification results are produced by faults impacting positive weights. In contrast, the Darknet19 neural network implements a different activation function, named Leaky ReLU, where negative parameters are attenuated instead of being set to zero. According to the results, this activation function makes Darknet19 vulnerable to faults in the whole range of weight values producing wrong classification results when faults yield large values for positive and negative parameters (see Fig. 4c).

To investigate how hardware faults propagate in DNNs having skip connections, a further resilience analysis was performed. We generated four different fault lists, one for each skip connection. For each fault lists, three distinct FI campaigns were performed: it was injected (one fault at a time) in the corresponding encoder and propagated through the original network. Then, we injected the same faults only through the skip connection (horizontal propagation, Fig. 5), and finally only in the deep part of the network (vertical propagation, Fig. 5). In Table II, we can observe that the number of critical faults tends to be the same for the three scenarios for deeper encoders. Only in the first encoder we can observe a marginal variation in critical faults: 3.16% for the original scenario, 2.98% for the horizontal scenario and 3.26% for the vertical scenario. Since there is no substantial difference, we can claim that skip connections do not amplify critical faults, rather they might slightly mitigate them.

TABLE II: Skip Connections Experimental Results

Encoder	Layer	Critical Faults [%]		
		Original	Vertical	Horizontal
1	1	3.16	3.26	2.98
2	3	1.48	1.51	1.45
3	5	1.73	1.76	1.74
4	7	1.41	1.41	1.41

D. Conclusions and Future Works

In this work, the main outcomes of reliability assessments on DNNs performing segmentation and classification tasks are reported. Experimental results on U-Net show that the ReLU activation function reduces by half the propagation of critical faults thought the neural network. This self-mitigation effect is also verified in CNN used for image classifications (i.e., LeNet, AlexNet, and Darknet19). Finally, experiments on the U-Net skip connections demonstrate that the effect of faults is not amplified by the presence of double connections.

III. A Machine Learning Framework to Catch Inter-Wafer Process Variations in Microcontrollers' Performance Prediction

In safety-critical applications, such as automotive or avionics, microcontrollers (MCs) must be tested to satisfy strict quality constraints and performance in terms of maximum operating frequency ($F_{\max}$). Several works in literature have demonstrated that on-chip ring-oscillator (the speed monitors, or SMONs) measurements can be related to the performance of integrated circuits. SMONs have been used in machine learning (ML) models as *features* for $F_{\max}$ prediction for performance screening and speed binning during the production [5]. However, the trained ML models can become significantly inaccurate over time due to shifts in the data distribution caused by variations between wafers or changes in industrial manufacturing processes. Previous research has demonstrated how to predict, using regression, the MC performance from SMON data using corner-lot wafers [5], [6]. This work aims to show how to extend this approach to cope with inter-wafer process variations. We exploit a novelty detection procedure that is only based on wafer-level information derived by SMON measurements on single dice. Test engineers can use this method to evaluate new production lots and understand whether to retrain the model to achieve a better generalization.

A. Process Variations in IC Manufacturing

The CMOS manufacturing process is continuously affected by process variation. Generally speaking, such variations can

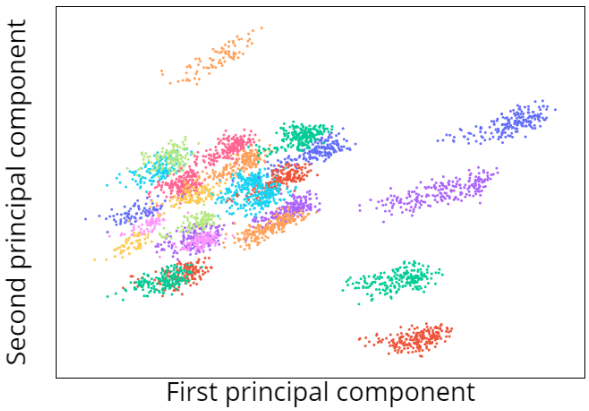


Fig. 6: A 2D PCA representation of 26 corner-lots wafers. Each colored-blob is a wafer

be *local* (i.e., intra-die) or *global* (i.e., inter-die). Local process variation has an impact inside each die. It is not the focus of this work. Global process variation can occur from *die-to-die*, *wafer-to-wafer*, and *lot-to-lot* [7]. In this work, we focus on wafer-to-wafer, also known as *inter-wafer*, process variation.

Design *corner analysis* is a standard method to represent the process variation. The corner analysis represents the device behavior under different process, voltage, and temperature (PVT) conditions. Especially for the device performance, the worst-case corner has to be considered. Voltage and temperature are deterministic values aside from potential noise and will be set to the most critical values. However, process variation has more parameters that need to be considered, and this is done with the help of corner-lot wafers. Corner-lot wafers are manufactured to observe the corners of the process parameter space. Those wafers mimic these process variations to understand parameter shifts and their influence on the performance. The corner-lot wafers distinguish between nMOS and pMOS transistor shifts in the process parameters. The nMOS transistor respective the pMOS transistor can be designed fast (F), typical (T), or slow (S). Each permutation of the transistor designs is applied in one lot with the corner-lot wafer [7]. With those variabilities in the process parameters, devices can be artificially changed, and plenty of combinations can be used to cover as many process shifts as possible.

The process parameters of the entire manufactured population is usually modeled as a normal distribution. The delay and the performance are also assumed as a normal distribution. The observation of the device performance of the left or right tail devices of this distribution is challenging because those *tail devices* are seldom in the production flow — especially when they deviate several sigmas from the mean. The corner-lot wafers have exactly this desired property: they differ significantly from the typical devices. A substantial number can be collected and characterized by the slow and fast tail of the distribution using the manufactured corner-lot wafer. Hence, the initial training set for performance screening contains various corner-lot wafers that imitate the whole manufacturing distribution.

However, as corner-lots are manufactured in the engineering phase to understand and improve process shifts for the product ramp-up, their characteristics might be intrinsically different from the production, and therefore be deleterious for the final model accuracy.

In Figure 6, it is possible to see a 2D representation of the devices that comes from the 26 corner-lots wafers used for our analysis. This was created using a PCA representation able to extract the 2 principal components of the SMONs in the dataset.

B. Machine Learning Background

1) Dataset Shift: Dataset shift is a common problem in predictive modeling that occurs when the joint distribution of inputs and outputs differs between training and test stages. This problem can be a consequence of data sparsity or how training and test sets are built, in the sense of the presence of bias in experimental design, due to the non reproducibility of the testing conditions at training time, or because of the intrinsic nature of the problem that has changed over time. The hypothesis of i.i.d (independent and identically distributed) data between training and test set often does not hold in real-world application, and this can lead to a machine learning model that is suboptimal for a part of the data. Dataset shifting occurs within supervised learning paradigm or semi-supervised learning, often involving the label of the task we are facing. Dataset shift manifests in different forms:

- Covariate shift
- Prior probability shift
- Concept shift
- Internal covariate shift (a sub-type of covariate shift)

Covariate Shift occurs when only the input distribution changes among training and test set. This is normally due to changes in state of latent variables, which could be temporal or spatial, facing non-stationary environment. Naming $\mathcal{D}(x)$ the distribution of the input feature, covariate shift is in formula:

$$\mathcal{D}(x)_{train} \neq \mathcal{D}(x)_{test} \tag{2}$$

Prior Probability Shift occurs when the dependent variable distributions (i.e. the labels or class distributions, $\mathcal{D}(y)$) between test and training set are different:

$$\mathcal{D}(x|y)_{train} = \mathcal{D}(x|y)_{test},\ \mathcal{D}(y)_{train} \neq \mathcal{D}(y)_{test} \tag{3}$$

An intuitive way to think about it might be to consider an unbalanced dataset. Concept drift means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This causes problems because the predictions become less accurate as time passes. Concept shift is defined as:

$$\mathcal{D}(x)_{train} = \mathcal{D}(x)_{test},\ \mathcal{D}(y|x)_{train} \neq \mathcal{D}(y|x)_{test} \tag{4}$$

Internal covariate shift is problem related specifically to the deployment of a Deep Neural Network. This happen due to the variation in the distribution of activations from the output of a given hidden layer. Since this is used as the input to a subsequent layer, the network layers can suffer from covariate shift which can impede the training of deep neural networks. The most common causes of dataset shift are bias in sample selection and non-stationary environment. Bias in sample selection can occur also in Cross Validation split schemes without noticing. Non-stationary environments

occurs when training and test sets environments are different due to temporal or spatial change. In a conventional Machine Learning problem, we assume that the input data distribution does not vary over time. But if the input data distribution changes over time during the testing/operating phase, because of the presence of non-stationarities, covariate-shift happens. This requires the developments of a method that should be computationally efficient and that has to be able to detect the shift-point in the underlying distributions of the data stream. Online or real time classification may seen as problems in which dataset shift is present and for which exist several methods presented in the literature to deal with, such as completely retraining the classifier on the new received data in an offline mode or retraining the classifier (sample-by-sample) in an online mode fashion. Active learning approaches can be helpful to solve these problems, with the aim of finding an optimal training set able to generalize well on new unseen data. Also Novelty Detection techniques can be applied to model the data source distribution.

2) Novelty detection: Novelty detection has the aim of classifying data that differ from known data available during the training. This may be seen as "one-class classification" problem, which goal is to build a model able to describe "normal" training data and identify data that are different in some way.

In our case, the "novelty" can be seen as the inter-wafers variations: without special measures, a model built using only the information contained in the corner wafers cannot always be applied in the prediction of the maximum operating frequency of devices in the production phase because, due to the dataset shift, it would not be able to generalize. Novelty detection techniques allow us to monitor the model as soon as it encounters new data, estimating the accuracy of the prediction.

The novelty detection is performed by measuring the *Local Outlier Factor* (LOF) among the wafers. The LOF algorithm [8] measures the local deviation of density of a given sample with respect to its neighbors. It is "local" in the sense that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood, indicating the degrees of outlierness for each point in the dataset, not relying on a binary classification problem. Local outlier factor is based on "local density", where locality is given by the k-neighbors of a point. By comparing the local density of a sample to the local densities of its neighbors, one can identify samples that have a substantially lower density than their neighbors. In our case, the anomaly score LOF(w_n) is computed as the mean value of all anomaly scores LOF($x_i \in w_n$) of the samples in the new wafer w_n. LOF model is trained on the wafer inserted in the training set, while it is tested on new wafers that comes from production. Outliers have high LOF values (greater than 1), thus the wafer with the highest LOF is marked as novel/outlier. The features of the LOF model are the SMONs value only. Since this approach do not require the labels (i.e. $F_{\max}$) to mark a point as outlier, it can be identified as an unsupervised method.

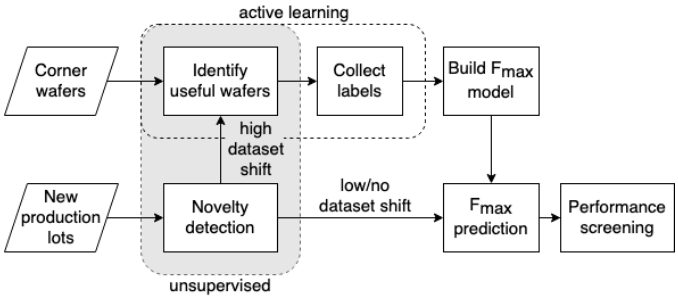


Fig. 7: Proposed performance screening flow.

C. Proposed Approach

The initial $F_{\max}$ prediction model is trained with corner-lot wafers during the engineering phase. All corner wafers are characterized, and the regression model is finalized. Some parameters are tuned or shifted during production ramp-up and high volume production due to the changes in the manufacturing process. Hence, the initial data on which the model was trained might considerably differ from the data in production, and this novelty must be identified and eventually used to give feedback to the training process. Novelty detection can be used for this purpose, to check if the new wafers are statistically different from the ones available at training time.

Once the problem is identified, the challenge is how to achieve high generalization capabilities in the model. One opportunity is retrain the model adding the novel-detected wafers, or by using different methods. Active learning is one of these: it can be used to minimize the amount of new wafers to characterize. This is crucial for two reasons: the labeling process of new data is time-consuming and, even worse, stopping the production and accessing productive devices is costly and should be done only when strictly necessary. Thus, only a small number of wafers from the production can be labeled with a feasible effort. The use of active learning techniques was discussed in [6], while in this work we are interested in the novelty detection phase, to identify unpredictable wafer that comes from production lines.

Apart from the novelty detection aspects during production, the initial training set can be validated and improved. For a new product, corner-lot wafers are used for an initial model buildup. Then, novelty detection is performed and novel wafer are selected. Active Learning can now be used to rank the wafers on the basis of how much information they would add to the model and on their impact on the regression model. This rank can be used possibly to retrain the model if a shift in the production has been detected. Figure 7 graphically depicts the above flow.

D. Experimental Evaluation

Our dataset is composed by about 2,400 labelled devices that from come from 26 corner-lots wafers. Each device is a state-of-the-art automotive microcontroller. For each wafer, the number of devices ranges from 46 to 204 with an average of 139. For each devices, the measurements of 27 SMONs are collected, and these are used as "features" by our algorithms. A second set of data was used for the novelty detection, composed of 44,476 samples from 125 production wafers.

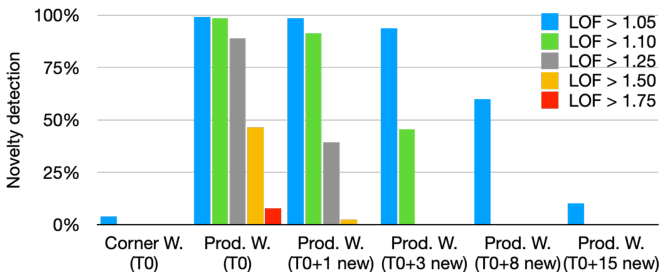


Fig. 8: Quality assessment in new production wafers using LOF novelty detection and active learning.

We tested the novelty detection mechanism on the dataset of production wafers. We used the LOF model trained with all 26 corner wafers as the baseline — however, the same experiment can be repeated using k steps of the active learning approach; we reference this model as T_0. The novelty of a dataset can be calculated as the amount of devices where the model cannot be used because the value of LOF is higher than the threshold τ. The threshold therefore must be set according to the degree of accuracy required, and it influences the total number of retrains. Five values are reported, namely 1.05, 1.1, 1.25, 1.5, and 1.75.

The trend of the detection using those thresholds is shown graphically in Figure 8, starting from T_0 and adding new wafers from the production. Data with $\tau = 1.05$ is reported as an extreme case: we observe about 3.8% of novelty even in the corner-lot devices used for the training. When evaluating the LOF of production devices, we observed a high-percentage of novelty even with $\tau = 1.50$, while only 8% of devices were above $\tau = 1.75$. By adding a new wafer in the training set of the novelty detection model, all wafers get below the threshold $\tau = 1.75$, and less than 4% remain above 1.5. With 3 new wafers added to the set, no novelty is recorded with $\tau = 1.25$; with 8, the threshold can be reduced to $\tau = 1.10$; finally, 10% of novelty is still observed in the extreme case of $\tau = 1.05$ adding 15 new wafers.

Such an analysis can be done during production to monitor the novelty detection given by process variations: test engineers can evaluate new production lots and understand whether to add generalization to the $F_{\max}$ model to cope with new process variations; alternatively, they can decide to relax the novelty detection threshold and lose some accuracy in the $F_{\max}$ model and consequently tweak the performance screening threshold as discussed in [5]. After the assessment of the $F_{\max}$ model with new data, they can repeat the same analysis using T_1 (i.e., using the wafers in the new model) and new production lots.

E. Conclusions and Future Works

We presented an innovate work based on active learning and novelty detection for the performance screening of microcontrollers. We demonstrated that process variations introduced during production can be identified with unsupervised novelty detection algorithms using only speed monitors' readouts during the wafer sort. Novelty detection methods helps test engineer in finding the shift-point during the production, supporting the decision of retrain or not a machine learning model. Experiments showed that LOF model is an appropriate novelty detection method and can be easily applied to new wafers that come from production lines. Anomaly detection can be further improved. Also model accuracy can be improved, to achieve a better yield during the production. Active Learning techniques to create an optimized training set was discussed in [6], but also Transfer Learning techniques can be useful to reuse the knowledge acquired by the models trained on corner-lots and to adapt it on new wafers. This may come from an additional outlier detection step that takes into account measurement error that may happen during the Fmax acquisition phase. Future works can be done in this area by introducing new algorithms for novelty detection.

IV. System-level Test and Burn-In oriented Test Equipment

In the last decades, the importance and the use of electronic devices have grown day by day. Today, we have microcontrollers in almost every everyday device, from a simple household appliance to the high-end, latest trending automobile. Those embedded devices have grown in complexity in such a way that never before happened, thanks to the possibility of miniaturizing those components, the appearance of more cores inside the microcontroller, and, last but not least, some ad-hoc design plan to achieve optimal performance in minimum space.

The massive presence of electronic devices, achieved only with mass production, led us to the necessity of finding a new suitable way to test those devices [9]. Testing a device is a crucial, and essential way to guarantee that a product will work as intended when it was conceived, but not only. When a device is tested, it also reduces infant mortality. This is done by highlighting the devices that present some defects and by discarding them. Identifying those faulty devices leads to a reduction of errors, inoperability, or malfunctioning of the complex systems, precedently mentioned.

When a testing procedure is in effect, primary importance is taken by the time invested in the ongoing testing program's execution [10]. Since modern electronic devices have grown in complexity, also the number of internal components and peripherals has grown. Consequently, the time required to drive effective results has increased with the complexity of modern devices.

To achieve those goals, the testing procedure must be carried out with mastery because the identification of a fault could be very difficult. But, with the appropriate methodology, tools, and design the result will be a successfully tested device.

One of the most important and widely used testing techniques is the so-called "Burn-In". To use the burn-in technique is a major requirement that the device to test is designed for testability, this means that the flip-flop inside the device must be of type mux-scan. In such a way it's possible to insert inside the scan-chain, which is the chain of all the flip-flops of the device, a specific pattern (more details about the insertion and the pattern in the next section). After that, all the flip-flops that compose the scan-chain are forced to a predetermined value. The system status will be evolved to the next one. This passage could be performed during stressful conditions like higher or lower voltage than the nominal one, warmer

or colder temperature than operative one, etc. After that, the values inside the flip-flops are extracted and matched with a "golden pattern", which represents the expected status after the system's evolution.

An innovative testing technique, today more and more widespread, is the "system-level test" [11]. This particular technique is used to test devices as close as possible to operational conditions, verify the correct behavior of the system, and identify possible defects. This is possible by testing the whole device or only part of it. One of the most common procedures makes use of an operative system to simulate as close as possible to the real stress condition the normal workflow that the device should do if it will pass the test phase.

It's a good common practice when a test is performed to use multiple testing techniques in different production phases. This should be done to identify as soon as possible defective components and to avoid spending resources on costly manufacturing processes.

In these few pages will be exposed the proposed architecture (section IV-A), the experimental results (section IV-B), and in the end (section IV-C) together with the conclusion of this article what will be our future works on this topic.

A. *Proposed architecture for SLT and BI ATE*

In this section, you will be informed about the background knowledge used to implement the testing functionality of the tester (that will be discussed in section IV-B).

To better comprehend how the tester works it's necessary to start from the different types of testing techniques. There is a profusion of approaches that could be used to perform a test. Our choice fell back to the burn-in, this technique, as was previously mentioned works with patterns and scan-chain. Before treating those arguments, it's better to explain what is a "design for testability", in which category the burn-in belongs.

Design for testability means that the circuit was designed, as the definition, to facilitate the testing phase, and allow a more punctual control of the components of the system during the testing phase. To achieve this result, a possibility is to modify the circuit's flip-flops in such a way that from standard flip-flops it is composed only with mux-scan flip-flops. The so-called scan-chain of flip-flops allows one to deal with a sequential circuit as a combinational one. Usually, when a device is designed for testability, it also has some dedicated port that could be used to guarantee easier access to the pin used for the testing purpose. One of the most common standards widely used is the JTAG (Joint Test Action Group), which takes its name from the homonym consortium. By using the FSM (Finite State Machine) logic implemented inside the TAP (Test Access Port) Controller it is possible to set the device under test (also known as DUT) in test mode. Those concepts are now the state of the art, and more relevant information about those standards can be found here: [12].

The procedure to set the DUT in test mode takes the name of "test mode entry". It's a device-dependent mechanism that has the necessity of setting in specific registers inside the DUT, specific values. This is done to configure the DUT properly.

After the "test mode entry", if everything goes as planned, the DUT has entered the "test mode", from this moment it is possible to force the flip-flops by shifting inside the scan-chain

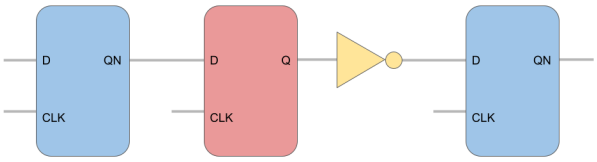


Fig. 9: close-up on a real scan-chain. Notice the NOT port present on the Q output of the second flip-flop

a pattern in case of a burn-in test, simulate the circuit of the DUT or stimulate a component by the SLT testing procedure. A pattern could be described as the state of the system at a specific time. In fact, after the shifting procedure, the system will have all the flip-flops set in a state determined by the pattern.

At this point, by applying a capture beat, it's possible to make the system evolve like one clock stroke at a time. The circuit will work as a sequential device, which it is. According to the test that wants to be performed and the "golden pattern" which could be described as the desired output after a specific number of clock strokes, the value of the flip-flops are extracted by another shifting procedure.

The way a pattern is shifted inside the scan-chain could be multiple, is it possible to use GPIO (General Purpose Input/Output) or the SPI (Serial Peripheral Interface) protocol. The shift-in (that's what is generally called the insertion shifting part of the scan test) is the most time-demanding part of the burn-in test. It follows that to optimize the test procedure a good practice is to shift data as quickly as possible (an above limit is introduced by the maximum shifting speed of the scan-chain which is device-dependent).

The last step that must be performed is to check if the extracted output from the scan-chain is compatible with the expected pattern. The patterns that must shift inside the scan-chain and the expected patterns are usually provided by the device's producer, as a result of a simulation of the circuit or in some cases as a result of some pseudo-random generation process. Every time a pattern is generated it is fundamental to consider that the scan chain, differently from what one might think, is not only a sequence of identical flip-flops. The scan-chain could contain some flip-flops which have the output as Q and could contain other flip-flops which have the output as QN. Moreover, the scan-chain is not only composed of flip-flops but we could find inside also some NOT logic ports. Those are present downstream every flip-flop with the Q output, to have the input of every flip-flop coherent (figure 9). Those considerations must be taken into account by modifying the pattern according to the design of the scan-chain. Unfortunately also this procedure is device-dependent and could be performed by using some offline parsing tool. Which will modify the patterns as requested by the position of the NOT port in the scan-chain.

Furthermore, the verification of the correctness is a bottleneck of the entire test process. So, it's crucial to deal with it as quickly as possible. There are several ways to accelerate the verification phase: instead of checking one single flip-flop value to the expected ones, it's possible to make a signature that will be used to compare the obtained signature to the desired one or implement the comparison in hardware using,

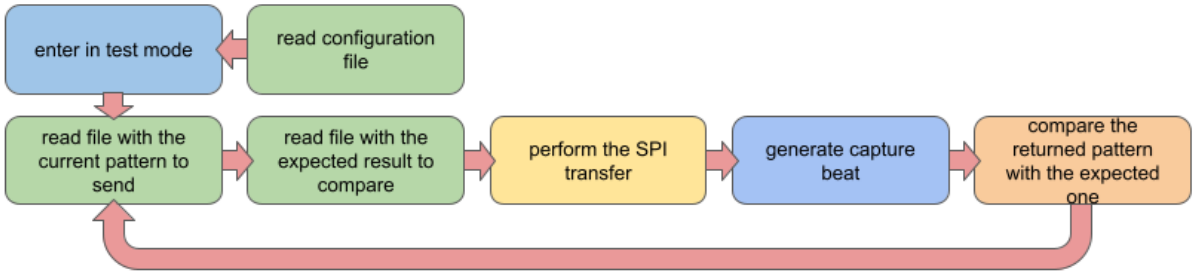


Fig. 10: flow-chart of a burn-in test

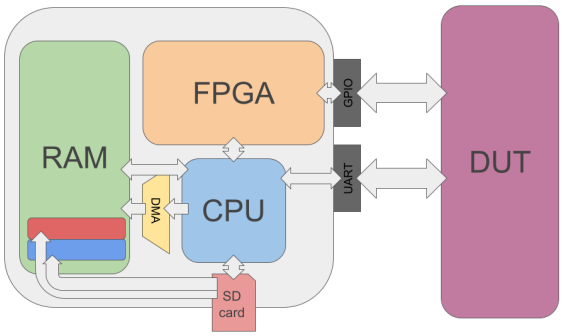


Fig. 11: proposed architecture of the tester and the DUT

for example, an FPGA (Field Programmable Gate Array) or some dedicated hardware. If the use of dedicated hardware is unapproachable, a fine alternative should be the use of parallel programming to reduce as much as possible the time required to perform the comparison task.

Another salient factor to keep in mind is the availability of the patterns. Normally an embedded testing device doesn't have much memory to store those patterns. This problem leads to the poor possibility to perform multiple testing procedures, inject specifically designed patterns, or, even else, rely only on pseudo-random pattern generation which could focus the test, not on critical parts of the DUT. A solution, adopted in this proposed architecture, plans to use an SD (Secure Digital) card to store those patterns, by using a large amount of memory available on those memory peripherals. This allows not only to store all the patterns that the tester needs to operate independently but also to store eventually log file reports of error that could be analyzed offline by using ad-hoc tools. In such a way, a device is tested but in case of a detected defect, the error found is listed to ulterior and more detailed investigation.

The tasks prior described , which represent a common burn-in test procedure using the proposed approach are represented in figure 10.

The proposed architecture of the tester (represented in figure 11) dispose of:

- PS (Processing System, also known as CPU): implements the logic of the tester, interfaces with the FPGA's IP core (Semiconductor Intellectual Property Core), interfaces with the memory and handles read and write operation with the SD card. A major duty under its responsibility is to handle the test-mode entry to put the DUT in test mode. Also, the comparison is handled by this component but could be moved inside the FPGA in case a speed-up is necessary.
- FPGA: implements every IP core necessary for every function of the testing procedure, in particular through the FPGA the tester will send and receive data with the DUT
- SD card: external memory used to store patterns and results from the testing phase. The patterns, with the auxiliary of the CPU, are read from the SD card and temporarily stored inside the RAM. Here for every burn-in test cycle (see the second line of figure 10), two different patterns are extracted from the SD card and placed in memory (red and blue part inside RAM, figure 11). Those patterns represent for every cycle the pattern to shift inside the scan chain and the expected result of the previously inserted pattern

B. *Experimental results*

In this section, the reader will be made aware of the real cause of the study, enriched with some experimental data (table III) about the tester implementation.

For the implementation of a tester based on programmable architecture, case of study of this paper, it was used the Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit, and for the DUT the chip SPC58x produced by STMicroelectronics. The SPC58x is mounted inside a daughterboard which is placed above the motherboard. The motherboard brings the power supply to the SPC58x chip, while the daughterboard brings the JTAG port used to put the chip in "test mode" and to shift in the pattern to carry out the scan test. The working set-up could be seen in figure 12, in particular, on the left, the DUT with the LFBGA 292 housing for the SPC58x chip, on the right, the Xilinx Zynq UltraScale+ MPSoC ZCU104 who represent the tester.

In compliance with the background discussed in section IV-A, the Xilinx Zynq UltraScale+ MPSoC ZCU104 implements most of the features described before. Its duty begins with sending the data required to the test mode entry for the DUT, this is done by using the AXI GPIO implemented in the PL (Programmable Logic, also known as FPGA). The logic to entry in the test mode, since it is device-dependent, is directly implemented in software, in such a way that follows the TAP Controller FSM. After that, the tester provides a CLI (Command Line Interface), by using it, the tester can

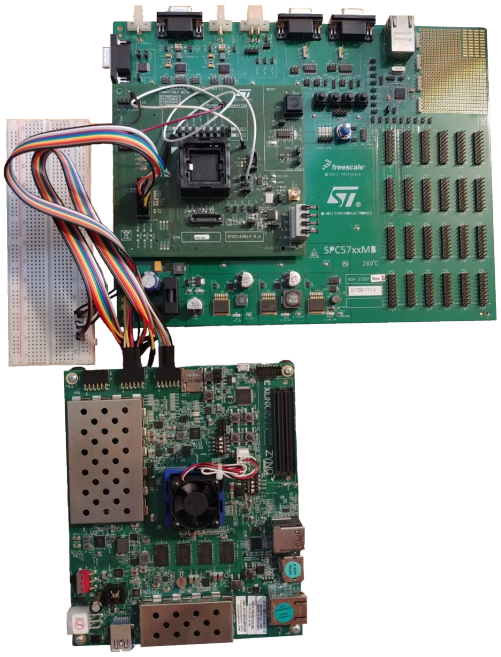


Fig. 12: operative set-up

TABLE III: execution time of a single burn-in test

Read configuration file	3.95 *ms*
Enter in test mode	612.93 *ms*
Read file with pattern to be sent	90.28 *ms*
Read file with expected result	90.28 *ms*
SPI transfer	125.91 *ms*
Capture beat	16.25 μs
Result comparison	2.35 *ms*
Total time for a burn-in pattern (excluding reading the configuration file and test mode ingress)	311.84 *ms*

communicate with a host PC using serial communication. Through the CLI it is possible to send commands to the tester to configure the execution of the various command and burn-in test methods present inside the firmware of the tester. When the burn-in test command is sent to the tester, the software starts to read, from the SD card present on-board, files previously parsed by external programs. Those files contain the pattern that must be sent into the scan-chain and the expected output. Since the scan-chain works as a queue, every time a new value is inserted a value is extracted from the end of the scan-chain. The shift-in and shift-out are performed using the SPI protocol at a speed of around 6.25MHz, this is due to an upper limit of the scan-chain shifting speed. To avoid unnecessary waste of time for every injection in the scan-chain the current value of the scan-chain is extracted and compared to the expected one. Moreover, the tester gave the user the possibility to save the result into the SD card, to have feedback, and to log errors that could be studied offline by using appropriate tools. Those tools just mentioned, are necessary because to fully exploit the SD card memory the patterns are coded in a binary format and saved in binary files. The tools will encode the patterns used by the tester, or decode the pattern received by the tester to provide human-readable files to the user.

C. Conclusions and Future Works

This paper presented a way to implement a testing procedure using jointly user-defined hardware design, FPGA, and software firmware to speed up the so-called testing phase. Especially by using the FPGA, the design is very scalable and easily customizable as needed, but keeping optimum performance.

Our aims for the future are to extend the testing capability for the previously described tester. This extension foresees adding the burn-in delay test, which should make the tester able to detect more and more defective devices. Furthermore, we intend to start to test in an improved stressful condition the SPC58x chip, regulating the supply tension by using a controlled board, capable of a voltage regulator.

V. Remarks

This paper touches extremely relevant topics and it shows solutions addressing crucial issues in the production of today's System-on-Chip designed for the Automotive segment.

The three contributions from major players in the Automotive arena cover self-mitigation abilities estimation for DNNs, AI-based analysis of inter-wafer performance variation, and effective test equipment to minimize the cost of Burn-In and System-Level-Test.

References

[1] T. Ito, G. Yang, P. Laurent, D. Schultz, and M. Cole, "Constructing neural network models from brain data reveals representational transformations linked to adaptive behavior," in *Nature Communications*, vol. 13, 2022.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[3] R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009, pp. 502–506.

[4] A. Bosio, P. Bernardi, A. Ruospo, and E. Sanchez, "A reliability analysis of a deep neural network," in *2019 IEEE Latin American Test Symposium (LATS)*, 2019, pp. 1–6.

[5] R. Cantoro, M. Huch, T. Kilian, R. Martone, U. Schlichtmann, and G. Squillero, "Machine learning based performance prediction of microcontrollers using speed monitors," in *2020 IEEE International Test Conference (ITC)*, 2020, pp. 1–5.

[6] N. Bellarmino, R. Cantoro, M. Huch, T. Kilian, R. Martone, U. Schlichtmann, and G. Squillero, "Exploiting active learning for microcontroller performance prediction," in *2021 IEEE European Test Symposium (ETS)*, 2021, pp. 1–4.

[7] V. Champac and J. G. Gervacio, *Timing Performance of Nanometer Digital Circuits Under Process Variations*. Springer International Publishing, 2018.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *2000 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2000, p. 93–104. [Online]. Available: https://doi.org/10.1145/342009.335388

[9] D. Appello, C. Bugeja, G. Pollaccia, P. Bernardi, R. Cantoro, M. Restifo, E. Sanchez, and F. Venini, "An optimized test during burn-in for automotive soc," *IEEE Design Test*, vol. 35, no. 3, pp. 46–53, 2018.

[10] N. Sumikawa, L.-C. Wang, and M. S. Abadir, "An experiment of burn-in time reduction based on parametric test analysis," in *2012 IEEE International Test Conference*, 2012, pp. 1–10.

[11] F. Almeida, P. Bernardi, D. Calabrese, M. Restifo, M. S. Reorda, D. Appello, G. Pollaccia, V. Tancorre, R. Ugioli, and G. Zoppi, "Effective screening of automotive socs by combining burn-in and system level test," in *2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2019, pp. 1–6.

[12] W. Lu, R. Wang, C. Zeng, C. Liu, and X. Wang, "A general fault injection method based on jtag," in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, 2018, pp. 604–608.