Abstract

. . . ... Short summary of the thesis ...

Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

1 Introduction

To be updated

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel ?? – ??: Hier werden werden die Grundlagen dieser Arbeit beschrieben.

Kapitel ?? – ?? fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

2 Faults in VLSI Systems

Reliability is always a cause of concern during chip manufacturing. A manufactured chip needs to function correctly not just during post-manufacturing tests but during the complete lifespan of the chip. Typical lifespan for a chip designed for commercial purpose is defined as 11.4 years or 100,000 hours [KP09]. Failure rate of ICs with respect to time is shown in figure 2.1, typically known as *bathtub curve*.

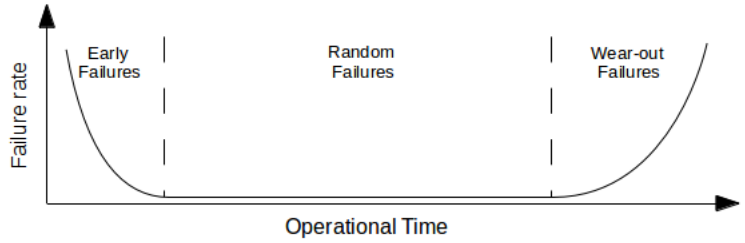

Figure 2.1: Bathtub Curve

The first region of the graph is called early failures or *infant mortality region*. The second region is the lifetime of the device when random failures occur. The error rate in this region is low and constant. Third region of the graph is wear-out and is caused by failures at the end of the useful life [KP09]. It can be expected that the ICs will not enter this region due to technology advances and obsolescence. This makes the first region important from the view of product quality and thus it is important to detect faults at manufacturing level.

Incorrectness of a VLSI system can be described as a fault, defect or an error. A *defect* in an electronic system is an unintended difference between implemented hardware and its design [Agr00]. Defects can occur either during the manufacturing process or lifetime of the device. An *error* is said to have occurred when an unintended output signal is produced by the system. An error is essentially manifestation of a defect. For the purpose of analysis, a defect is modeled as a *fault*, which is simply representation of defect at abstracted function level.

The figure 2.2 shows how faulty chips are identified. To decide whether the Device under Test (DUT) is working properly, a set of input stimuli called *test pattern set* is applied to the DUT. *output response* is observed and is then compared standard output. If these outputs do not match then the chip is said to be faulty.

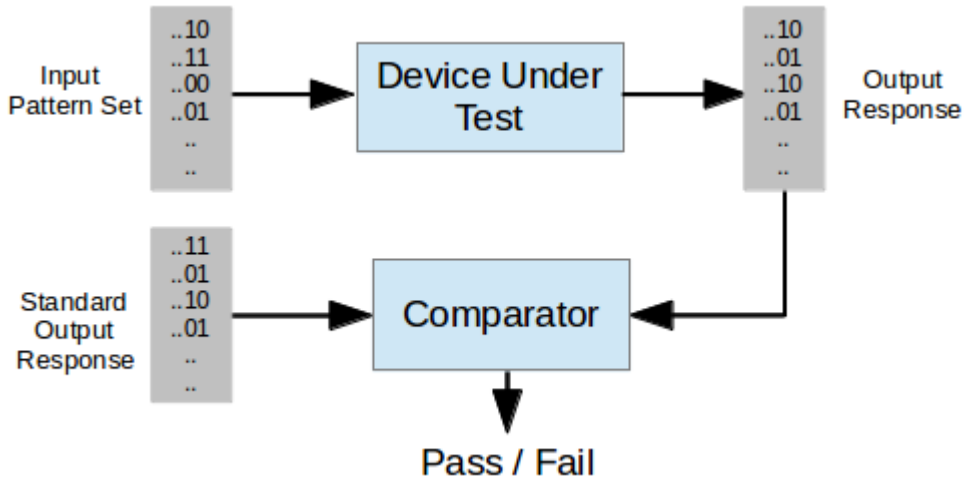

Figure 2.2: Typical test flow for VLSI Chips

The sources of the fault can be internal or external. When healthy chips fail due external mechanisms like α-particle strikes, they are thrown away which contributes to yield loss. Thus to maintain product quality and to reduce yield loss, it is important classify the faults. Section 2 of this chapter describes a taxonomy for such classification according to their sources and characteristic. It also focuses on various fault models that can be used to analyze these faults. The existing techniques for such fault classification are explained in section 3. The last section of this chapter explains diagnostic techniques and how they can used to classify faults.

2.1 Fault Taxonomy

According to the sources of faults, they can be classified into three types: *permanent*, *intermittent*, and *transient*. Permanent faults reflect irreversible physical changes. Intermittent faults occur because of unstable or marginal hardware and can be activated by environmental changes, like higher or lower temperature and voltage. Transients occur because of temporary environmental conditions [Con03]. The likelihood of these faults is expected to be higher as increasingly greater amount of transistors are integrated in smaller area [Con07a].

2.1.1 Permanent Faults

Permanent faults are those which occur due to permanent physical changes on chip. These faults generally occur due to issues in manufacturing process, however they can also occur during operational lifetime of the circuit, especially when circuit is old and starts to wear-out [L$^+$09].

Sources of Permanent Faults:

Manufacturing process: So-called *spot defects* can occur during manufacturing of a VLSI chip, and take form of either missing or extra material. Such a defect can cause an unwanted short or open between nodes or make an unintended multi-terminal transistor,leading to changed circuit topology. These defects mainly arise from some contamination, usually in form of dust particles or liquid droplets deposited on the wafer surface during some fabrication step [KM96]. Also missing or excess metal may cause unwanted capacitance and resistance respectively resulting in delay lines [WK95].

Operational failures: *Electromigration* (EM) is defined as mass transport of metal atoms created by collision of electrons [Gha82]. This movement of material will result in voids or hillock growth as in figure 2.3 [L+09], which can result in open circuit or short between adjacent tracks [Ana00]. With lower technology nodes, the wire widths are also getting smaller making EM a serious problem.

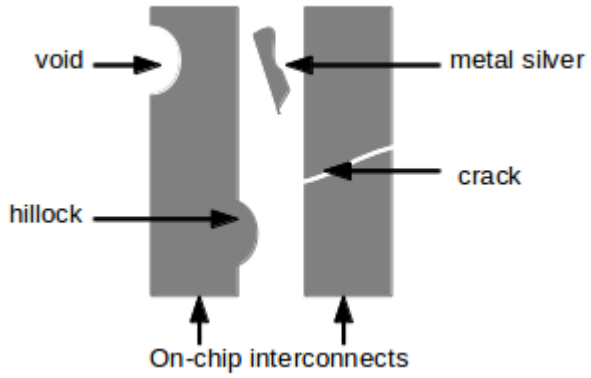

Figure 2.3: Manufacturing defects as sources for intermittent and permanent faults

Characteristics of permanent faults:

- Once permanent fault appears in the system, it does not go away.
- The faults are localized on the chip. They will affect the same set of primary outputs (POs).
- Permanent faults are reproducible and provide a predictable output response.
- Permanent faults only go away once the offending component is replaced [Con07a].

Fault models for permanent faults:

Stuck-at fault model: Stuck-at fault models are the most simplest fault models. Due to its simplicity and ability to explain many defects accurately, they are widely used [Lar06]. It assumes that the fault location has a fixed logical value, either stuck-at 0 or stuck-at 1. These can be seen as short to ground or short to power supply respectively. When it is assumed that there is only one fault in the circuit at a time then *single stuck-at* (SSA) model is used, otherwise in case of multiple defects *multiple stuck-at* model is used. In this thesis we have assumed that there is only one fault active in the circuit at a given time; hence we have used SSA model for our analysis.

Wired AND/OR fault model: Unlike stuck-at, *bridge fault* models a short between signal lines. *Wired AND/OR* fault model is type of bridging fault model. These models are used to describe logic behavior of two nodes that are shorted in the circuit. Wired AND model assumes that the faulty node of the bridge always has value 0, whereas wired OR model assumes faulty node has value 1.

Delay fault model: *Delay fault* model is used to model timing related faults. Delay testing is required for modern VLSI systems running at high frequencies, as even minor timing violations can lead to system performing out of specifications [Lar06]. There are two was to realize delay viz. *Gate delay* and *Path delay* models. Gate delay model assumes that the delay is only between input and output of individual logical gates on chip. In contrast, path delay models assume that the delay is spread over complete path from input to output.

2.1.2 Intermittent faults

Intermittent faults are those caused by marginal or unstable hardware and are activated when certain conditions like voltage, temperature or frequency are met [Con03, L$^+$09]. Intermittent faults often precede occurrence of permanent failures [L$^+$09].

Sources of intermittent faults:

Manufacturing defects: As illustrated in figure 2.3 *metal silvers* are stray pieces of metal on die due to some process imperfections. In certain conditions like increase in temperature, the metal may expand and touch the interconnects creating a short. In some cases the short might cause a current surge, damaging the circuit and can manifest into a permanent fault [HKS03]. Similarly cracks, as shown in same figure can continue to work normally at design temperature but at low temperatures can cause open circuits.

Technology scaling: With technology scaling, the supply voltages are also lowered down, which results in degraded noise tolerance [L+09]. Also the reduced thickness of oxide layers may result in current leakage, with a mechanism known as soft breakdown (SBD) [Sta01]. In such a breakdown, current fluctuates creating intermittent fault, without causing thermal damage [Sta01, Con07b, Con07a].

Characteristics of intermittent faults:

- Once intermittent fault appears in the system, its probability of recurrence increases [BCDGG00].
- The faults are localized on the chip. They will affect the same set of primary outputs (POs).
- Intermittent faults not reproducible every time however they provide a predictable output response.
- Intermittent faults only go away once the offending component is replaced [Con07a].
- Intermittent faults have tendency to occur in bursts [Con07a, Con03].

Fault models for intermittent faults

Intermittent faults can be modeled as conditional stuck-at faults, which are traditional stuck-at faults activated by trigger condition. The activation condition can be expressed as boolean function and may depend on timing or environmental conditions [HW09].

High frequency power droop: A high frequency power droop occurs when multiple cells on a chip connected to same power grid segment switch in same direction, increasing their current demand cause power starvation in some other part of chip. This fault is modeled as a set of aggressor lines $a_1, a_2, ...$ and a victim line v [PCKB07]. The fault occurs on victim line, due to presence of aggressor lines.

2.1.3 Transient faults

Transient faults are temporary deviations of normal circuit function caused by some temporary environmental factors or some external phenomenon. They are called soft errors as they do not do any permanent damage to the chip. A *single-event upset* (SEU), which is change in value of single bit, is the most common manifestation of transient faults.

Sources of transient faults

Particle strikes: When an α-particle, proton or a neutron passes through a semiconductor material and starts to loose energy, it frees electron-hole pairs along its path [DM03]. If this material happens to be a reversed biased p-n junction, it can result in significant transient currents to bring about an SEU. Hence with scaling to lower technology nodes, it is very likely that probability of such SEUs will increase.

Other sources: *Electromagnetic interference* caused by sources emitting high energy signal may interfere with working chip to bring about SEUs. *Electrostatic discharge* due to users releasing static charge can also affect chips to cause transient faults.

Characteristics

- Transient faults are non-deterministic faults.
- These faults are not localized hence can affect any of the POs.
- Transient faults are not reproducible.
- Replacing offending component may not make transient faults to go away [Con07a].
- Transient faults are isolated incidences of error occurrence, they usually do not occur in bursts like intermittent faults [Con07a, Con03].

Fault models for transient faults

One of the ways to model transient fault is to implement a conditional stuck-at fault, at multiple fault location and to use a deterministic function to trigger the fault [HW09]

2.2 Fault Classification

Test flows like one shown in figure 2.2 are able to distinguish between faulty and healthy chips. A better approach in [DK09] is able to distinguish permanent faults from faults with non-deterministic behavior. However the traditional post-manufacturing tests still work same way as shown in figure 2.4 [WE85].

With advances in manufacturing processes number of permanent faults are reducing [KP09], while on the other hand the impact of soft-errors and other non critical failures is increasing [Con03]. Some studies have indicated that up to 80% failures can be attributed to SEUs

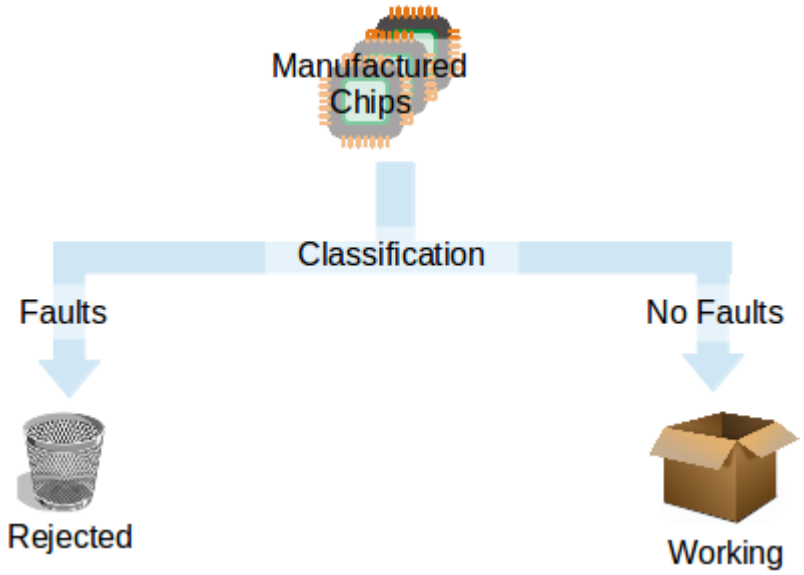

Figure 2.4: Traditional flow for fault classification

[IR86, DKCP94, KP09], which suggest that these cases will result in unnecessary yield loss. This section describes a few techniques to discriminate between different error categories described earlier in the chapter. As the error categories are similar in characteristics to those observed on PCs or workstations, the approaches used for their classification also provide a few pointers towards fault classification in VLSI systems.

In [LS90] a mechanism to classify between transient and intermittent faults is explained for error log analysis. In a technique called *Dispersion Frame Technique* (DFT), inter-arrival time in between successive error events of same error types is used to determine type of fault in the system. Heuristics are applied to determine closeness in time and affected area which are then considered as parameters to decide whether the error is of the same type.

Authors in [IYI90] use a similar technique to identify persistent failures in the system. Here they have used error rates to build up correlation using simple probabilistic techniques between error records, leading to a set of symptoms which may suggest a common cause (permanent errors).

A probabilistic approach is considered in [PSBDG98], which updates probability of module being affected by permanent fault. It then weighs the consequences of actions performed by a faulty module vs. fault-free module uses Bayesian inference to discriminate between permanent and transient errors

2.2.1 Fault Classification Techniques for VLSI Systems

Historically a lot of work has been done to analyze impact of different types of faults on VLSI systems [Con03, Con07a, DM03] and to classify them [Sav80, EARG13, BCDGG00, DK09]. The most popular techniques to classify transients from other types of faults are grouped under a family called α-count techniques [BCDGG00]. In a scheme called*single threshold α-count techniques*, a single threshold is established and if error count exceeds this threshold then fault is classified as permanent or intermittent, whereas a smaller non-zero value indicates presence of transient faults. In an other variant of the same called *dual threshold α-count techniques*, two thresholds are established. If error count exceeds first threshold then component is assigned a restricted functionality and when it exceeds the second threshold it is taken out of service, like single threshold. However a component in between thresholds can be taken in full service once its error count is lowered than first threshold.

2.3 Fault Diagnosis

Diagnosis is the process of locating faults in a physical chip at the various levels down to real defects. In traditional fault-dictionary based diagnosis, we are given two sets of data, a *predicted output* P which is set of outputs when fault a particular fault is active in the system, a *measured set* M, which is observed fault behavior and corresponding fault $f_i \in \{f_1, f_2, ..., f_n\}$. When the two sets match i.e. $P = M$ the corresponding fault is diagnosed to be active . When $P \neq M$ then logic diagnosis tries to find best fitting explanation. However it is practically infeasible to construct such fault dictionaries for modern circuits, as the fault dictionary should consist of all possible faults and their combinations [Wan10].

An adaptive approach which does not use fault dictionaries called *Partially Overlapping Impact couNTER* (POINTER) for diagnosis is described in [HW09]. This approach uses test pattern sets with *Single Location At a Time* (SLAT) property [BSHH01] to diagnose faults present in the circuit. The author in [HW09] defines a *(*Fault Machine) (FM),i.e. a circuit with stuck-at fault injected. As shown in figure 2.5 a set of tuple parameters called *evidance* is defined as,

$e(f, T) = (\sigma_T, \iota_T, \tau_T, \gamma_T)$, where for all patterns $t \in T$

- σ_T is sum of number of failing outputs where Device Under Diagnosis (DUD) and FM match
- ι_T is sum of number of outputs which fail in FM but are correct in DUD.
- τ_T is sum of number of outputs which fail in DUD but are correct in FM.
- γ_T is the sum of maximum of ι_t and τ_t for every pattern $t \in T$.

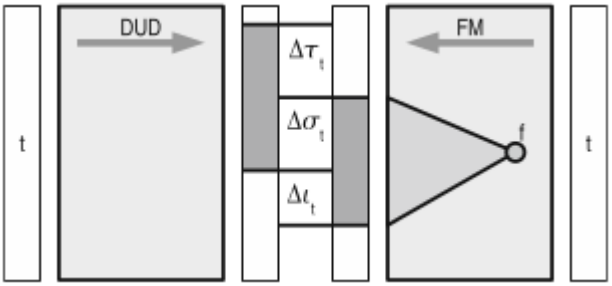

Figure 2.5: Definition of evidance

These parameters vary depending on the type of fault present in the circuit. The use of these parameters to classify faults is explained in section <section reference: feature selection> of this report.

3 Machine Learning

An *algorithm* is set of instructions used to convert input values to output, based on certain rules. Consider an example where we need to find all even numbers from dataset. Here, we can set up a *rule* that if number is completely divisible by two then it should be included in the output dataset, otherwise not. Naturally, as there can be more than one way to solve a problem, there can be more than one algorithm to solve it. However there are certain examples where formation of set of rule is practically infeasible. For example, consider handwriting recognition software used to scan handwritten forms. Figure illustrates problem at hand, where a simple character can be written in a number of ways. It is interesting to note that humans are able to read this data without trouble, but it is really difficult express a certain rules which will result in accurate recognition with help of an algorithm. Machine learning is employed in such cases. Specifically *Machine Learning* (ML) is programming computers to optimize a performance criterion (e.g. character recognition) using example data or past experience [Alp04].

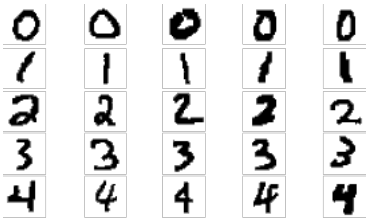

Figure 3.1: Example of Machine Learning: Character recognition

The "example data" with its *label* is collectively called as *training set*, and it is used to teach machine learning how the character with given label looks like, so that ML can recognize when it encounters similar data in future. Machine learning can be applied in wide range of applications where it is not possible to express expertise but a large amount of sample data is available. Typical applications of machine learning include computer vision, pattern recognition, spam filtering, search result optimization etc.

3.1 Basics of Machine Learning

3.1.1 Types of learning algorithms

Based on application, ML algorithms can be can be classified in two major categories *viz.* supervised learning and unsupervised learning.

Supervised learning algorithms are used when labels of the data to be are known. A spam filter is a good example where supervised learning can be used for *classification*. Here we know an email received is either "spam" or "not-spam", these categories can be used as labels for the sample population and learning algorithm can classify within these two type. One more application of supervised learning is to predict a numerical value in *regression*. Consider a problem to predict value of a used property, the input parameters in this case are initial value, year of construction, size of property, locality and so on, whereas output is current resale value. one can construct a training set of known resale values and receptive values of input parameters and train leaning algorithm to predict other inputs. To generalize, aim in supervised learning is to learn mapping from input to output whose correct vales are provided by supervisor [Alp04].

Unsupervised learning or *clustering* is used in classification problems where the labels for the data are not known. An example of such problem is document clustering [Alp04]. One of applications of document clustering is to cluster news reports which belong to same category like sport, science, art and so on. The number of such categories is not clear, and the machine learning application in such case needs to cluster articles based on some common words, and provide the supervisor data, which he may use to label clustered groups.

In case of fault classification, we have clearly defined taxonomy in earlier chapter, making our case as supervised classification problem. In following subsection, we define basic terms as applied to case of supervised learning.

3.1.2 Features and Feature Selection

A *feature* (x_i) is a result of measurement made on a unit input data. Generally, a set of features $(\boldsymbol{x}^t)$ is needed to characterize a unit of input data and is expressed as,

$$\boldsymbol{x}^t = [x_1, x_2, \ldots x_m]^T$$

Its label r denotes the class $C_i \in \{C_1, C_2 \ldots C_k\}$ it belongs to and is denoted as,

$$r_i^t = \begin{cases} 1 & \text{if } \boldsymbol{x}^t \in C_i; \\ 0 & \text{if } \boldsymbol{x}^t \in C_j and j \neq i \end{cases}$$

The training set X is then defined as ordered set containing N values of such examples,

$$X = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^N$$

The aim for machine learning algorithm is to learn values in training set and then classify new examples $\boldsymbol{x}$ by estimating value of $C(\boldsymbol{x})$. To achieve this algorithm tries to find out a hypotheses $h_i, i \in \{1, 2, \dots k\}$ from a set of all possible hypotheses such that,

$$h_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}^t \in C_i; \\ 0 & \boldsymbol{x}^t \in C_j and j \neq i \end{cases}$$

The *empirical error* after training is calculated as,

$$E(\{h\}_{i=1}^k | X) = \sum_{t=1}^{N} \sum_{i=1}^{k} |h_i(\boldsymbol{x}) \neq r_i^t)|$$

Figure 3.2 shows two possible hypotheses h_1 and h_2 for a simple 2-class classification problem, both with same value of empirical error and also actual boundary of classification C. If we choose hypothesis h_1 then the examples which lie in region between h_1 and C will get incorrectly classified and this is called as *overfitting*. On the other hand, if we choose h_2 then same will happen for examples in region between C and h_2, called *underfitting*. To avoid this and get a hypothesis which is as close to C as possible, one more labeled dataset with examples other than training set called as *cross-validation set* is picked. The empirical error is then calculated over this set and hypotheses obtained during training, and the hypothesis with least value of error is selected.

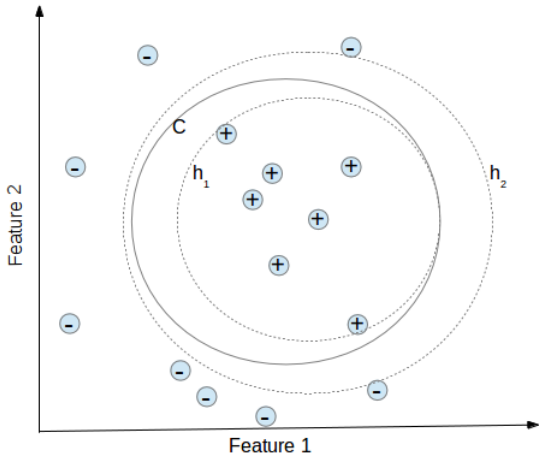

Figure 3.2: Example of overfitting and underfitting

We use the term *sample population* collectively for training set and cross-validation set.

3.1.3 Sample population

Hi hrello bol do

3.2 Machine Learning Algorithms for Classification

3.2.1 Bayesian networks

3.2.2 Decision Trees

3.2.3 Multilayered Perceptrons

3.2.4 Support Vector Machines

3.3 Criterion To Choose Suitable ML Algorithm

Bibliography

[Agr00] V. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Frontiers in Electronic Testing. Springer, 2000. (Cited on page 7)

[Alp04] E. Alpaydin. *Introduction to machine learning*. MIT press, 2004. (Cited on pages 17 and 18)

[Ana00] Analog Devices Inc. *ADI Reliablity Handbook*, 2000. (Cited on page 9)

[BCDGG00] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, F. Grandoni. Threshold-based mechanisms to discriminate transient from intermittent faults. *Computers, IEEE Transactions on*, 49(3):230–245, 2000. (Cited on pages 11 and 14)

[BSHH01] T. Bartenstein, D. Sliwinski, D. Heaberlin, L. Huisman. Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm. In *2013 IEEE International Test Conference (ITC)*, pp. 287–287. International Test Conference, 2001. (Cited on page 14)

[Con03] C. Constantinescu. Trends and challenges in VLSI circuit reliability. *IEEE micro*, 23(4):14–19, 2003. (Cited on pages 8, 10, 11, 12 and 14)

[Con07a] C. Constantinescu. Impact of intermittent faults on nanocomputing devices. *Proceedings of IEEE/IFIP DSN-2007 (Supplemental Volume), Edinburgh, UK*, pp. 238–241, 2007. (Cited on pages 8, 9, 11, 12 and 14)

[Con07b] C. Constantinescu. Intermittent faults in VLSI circuits. In *Proceedings of the IEEE Workshop on Silicon Errors in Logic-System Effects*. Citeseer, 2007. (Cited on page 11)

[DK09] J. De Kleer. Diagnosing Multiple Persistent and Intermittent Faults. In *IJCAI*, pp. 733–738. 2009. (Cited on pages 12 and 14)

[DKCP94] A. Dharchoudhury, S.-M. Kang, H. Cha, J. H. Patel. Fast timing simulation of transient faults in digital circuits. In *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design*, pp. 719–722. IEEE Computer Society Press, 1994. (Cited on page 13)

[DM03] P. E. Dodd, L. W. Massengill. Basic mechanisms and modeling of single-event upset in digital microelectronics. *IEEE Transactions on Nuclear Science*, 50(3):583–602, 2003. (Cited on pages 12 and 14)

[EARG13] J. Espinosa, D. Andrés, J.-C. Ruiz, P. Gil. The Challenge of Detection and Diagnosis of Fugacious Hardware Faults in VLSI Designs. In M. Vieira, J. Cunha, editors, *Dependable Computing*, volume 7869 of *Lecture Notes in Computer Science*, pp. 76–87. Springer Berlin Heidelberg, 2013. (Cited on page 14)

[Gha82] P. Ghate. Electromigration-induced failures in VLSI interconnects. In *20th Annual Reliability Physics Symposium, 1982.*, pp. 292–299. IEEE, 1982. (Cited on page 9)

[HKS03] C. Hawkins, A. Keshavarzi, J. Segura. View from the bottom: nanometer technology AC parametric failures-why, where, and how to detect. In *Proceedings of 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2003*, pp. 267–276. IEEE, 2003. (Cited on page 10)

[HW09] S. Holst, H.-J. Wunderlich. Adaptive debug and diagnosis without fault dictionaries. *Journal of Electronic Testing*, 25(4-5):259–268, 2009. (Cited on pages 11, 12 and 14)

[IR86] R. K. Iyer, D. J. Rossetti. A measurement-based model for workload dependence of CPU errors. *IEEE Transactions on Computers*, 100(6):511–519, 1986. (Cited on page 13)

[IYI90] R. K. Iyer, L. T. Young, P. V. K. Iyer. Automatic recognition of intermittent failures: An experimental study of field data. *Computers, IEEE Transactions on*, 39(4):525–537, 1990. (Cited on page 13)

[KM96] J. Khare, W. Maly. *From Contamination to Defects, Faults and Yield Loss: Simulation and Applications*. Frontiers in Electronic Testing. Springer, 1996. (Cited on page 9)

[KP09] K. Kishore, V. Prabhakar. *VLSI Design*. I.K. International Publishing House Pvt. Limited, 2009. (Cited on pages 7, 12 and 13)

[L+09] T. Lehtonen, et al. *On fault tolerance methods for networks-on-chip*. Ph.D. thesis, 2009. (Cited on pages 8, 9, 10 and 11)

[Lar06] E. Larsson. *Introduction to Advanced System-on-Chip Test Design and Optimization*. Frontiers in Electronic Testing. Springer, 2006. (Cited on page 10)

[LS90] T.-T. Y. Lin, D. P. Siewiorek. Error log analysis: statistical modeling and heuristic trend analysis. *IEEE Transactions on Reliability*, 39(4):419–432, 1990. (Cited on page 13)

[PCKB07] I. Polian, A. Czutro, S. Kundu, B. Becker. Power droop testing. In *IEEE International Conference on Computer Design, 2006*, pp. 243–250. IEEE, 2007. (Cited on page 11)

[PSBDG98] M. Pizza, L. Strigini, A. Bondavalli, F. Di Giandomenico. Optimal discrimination between transient and permanent faults. In *Proceedings of Third IEEE International High-Assurance Systems Engineering Symposium, 1998*, pp. 214–223. IEEE, 1998. (Cited on page 13)

[Sav80] J. Savir. length M. Each set of curves approaches limiting curve for large. *IEEE Transactions on Computers*, 100(29), 1980. (Cited on page 14)

[Sta01] J. H. Stathis. Physical and predictive models of ultrathin oxide reliability in CMOS devices and circuits. *IEEE Transactions on Device and Materials Reliability*, 1(1):43–59, 2001. (Cited on page 11)

[Wan10] L.-C. Wang. Data learning based diagnosis. In *15th Asia and South Pacific Design Automation Conference (ASP-DAC), 2010*, pp. 247–254. IEEE, 2010. (Cited on page 14)

[WE85] N. H. Weste, K. Eshraghian. Principles of CMOS VLSI design: a systems perspective. *NASA STI/Recon Technical Report A*, 85:47028, 1985. (Cited on page 12)

[WK95] I. A. Wagner, I. Koren. The effect of spot defects on the parametric yield of long interconnection lines. In *Proceedings of 1995 IEEE International Workshop onDefect and Fault Tolerance in VLSI Systems, 1995*, pp. 46–54. IEEE, 1995. (Cited on page 9)

All links were last followed on March 17, 2014.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature