designs

MDPI

Article

Automated Test Case Generation for Digital System Designs: A Mapping Study on VHDL, Verilog, and SystemVerilog Description Languages

Ashish Alape Vivekananda and Eduard Enoiu *

Department of Networked and Embedded Systems, Mälardalen University, 722 20 Västerås, Sweden; mr.ashishvivek@gmail.com

* Correspondence: eduard.paul.enoiu@mdh.se

Received: 30 May 2020; Accepted: 24 July 2020; Published: 5 August 2020

Abstract: Researchers have proposed different methods for testing digital systems and circuits in the last couple of decades. The need for testing digital logic circuits has become more important than ever due to the growing complexity of such systems. During the design phase, testing is focusing on design defects, as well as manufacturing and wear out type of defects. Failures in digital systems could be caused, for example, by design errors, the use of inherently probabilistic devices, and manufacturing variability. As a way to test digital systems in a more efficient way, automated test generation has been proposed to automatically create tests that can quickly and accurately identify faulty components. Examples of such techniques are the sequential test generation, the scan path testing, and the random test generation techniques. With the research domain becoming more mature and growing, it is essential to systematically identify, analyze, and classify these contributions. We performed a systematic mapping study of automated test generation for digital circuits aimed at providing an overview of the application of these techniques. We focused on three of the most widely-used and well-supported hardware description languages (HDLs) for digital systems: Verilog, SystemVerilog, and VHDL. Our results suggest that the majority of the test generation methods for digital circuits are focused on the behavioral and register-transfer design levels. Fault-independent and fault-oriented test generation are the most frequently reported types of test generation methods, while HDL model simulation is the most common test generation technology used to search for test cases in these academic studies. While the results are suggesting a growing interest in this area, the majority of articles are published as conferences papers. Our results show that only 31% of the methods are implemented as software tools and only 63% of all contributions are actually generating executable test cases. This study makes three important contributions, (i) a state-of-the-art of test generation for digital system designs research is provided, (ii) the reported characteristics are identified in both the primary papers and experimental reports, (iii) gaps and opportunities for future test generation for digital system designs research are identified.

Keywords: testing; test generation; digital systems; digital circuits; Verilog; VHDL

1. Introduction

Digital systems are electronic system that are used in every industrial domain from railway, automotive, process, and automation to communication technologies. Hardware description languages (HDL) are used to describe the structure and behavior of digital systems for complex embedded systems, such as application-specific integrated circuits, microprocessors, and programmable logic devices. Testing of these digital circuits is used to check if the whole system or a sub-component deviates from its design specification [1]. Testing of circuits designed using HDLs has been an active

research area for the last couple of decades. A fault is said to have occurred in a digital system if the behavior of the circuit deviates from its nominal behavior [2]. A digital system with a fault will therefore not exhibit similar functionality as a circuit without a fault. Fault models [1,3] are used at different levels of design abstractions for modeling the logical correspondence to a physical defect (e.g., behavioral, functional, structural fault models). For example, at the behavioral design level, an *if* construct can fail when one set of statements is always executed and this logical behavior relates to an actual defect in the actual hardware. Certain test inputs can be generated to distinguish these logical differences between the correct and the faulty circuit.

The advancements of VLSI (Very Large-Scale Integrated Circuit) [4] technology has enabled the manufacturing of complex digital logic circuits on a single chip. This poses many challenges in terms of both functional and non-functional aspects that need to be considered when testing such systems using HDLs. Producing a digital system begins with the specification of its design using an HDL and ends with manufacturing and shipping the overall system. This process involves simulation, synthesis, testing, and maintenance of such a digital system. Many of the approaches proposed in academia for test generation have been adopted in different Electronic Design Automation (EDA) tools [5] during the last couple of decades. In this paper, we undertake a systematic mapping study to present a broad review of the academic studies on test generation for Verilog, VHDL, and SystemVerilog, three standard HDLs that are widely supported. The goal of this study is to identify, gather the available evidence, and to identify possible gaps in test generation of digital systems designed using HDLs, including but not limited to the following aspects: test generation of HDL-based designs, testing of performance and timing issues, and testability. We conclude that test generation for digital systems needs broader studies, and the results of this mapping study can directly help in this effort.

2. Background

In this section, we introduce digital circuit testing and hardware description languages.

2.1. Digital Circuits

Digital circuits can be designed using interconnected logical elements. A few examples of logical elements are AND/OR gates, registers, inverters, flip flops, and others [6]. A digital circuit must be able to process finite-valued discrete signals. A well-known digital circuit is a microprocessor used in digital computers. Digital systems are part of a wider category of discrete systems. Stefan [7] defined digital systems in a taxonomy of orders based on feedback loops, as follows:, zero-order systems containing the combinational circuits, 1-order systems containing the memory circuits, 2-order systems containing the automata, 3-order systems containing the processors, and n-order systems containing self-organizing circuit behavior. In Figure 1, we show an example of a selection circuit, an elementary multiplexer for two inputs in different forms from the logical schematic to the structural description in a hardware description language. There are many other circuits that can be composed for programmable circuits. For example, latches are memory elements that are triggered by a level sensitive enable signal. Designing a digital system means the use of such representations and specifications to engineer a complex system that takes into account the internal state and the evolution of outputs in the whole system.

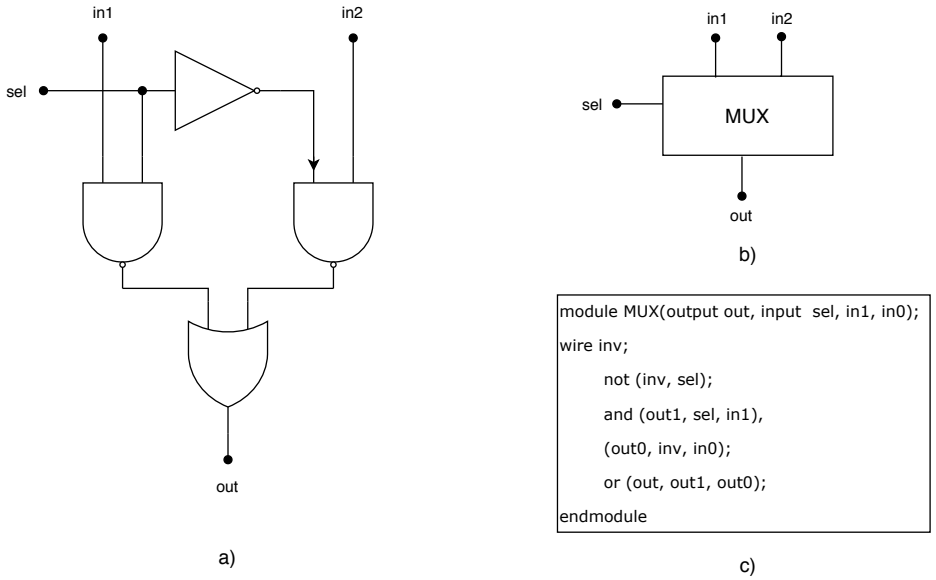


Figure 1. An example of a selection circuit showing the logic schematic for the elementary selector (**a**), the logic component for MUX (**b**), and the structural description in Verilog (**c**).

HDLs [8] are languages used to describe the hardware function of a digital system. HDLs can be used to implement and describe the function of a hardware at varying level of abstractions (as shown in Figure 1). HDLs are widely used languages to manage complex digital systems and can be used to design and simulate a circuit before implementation. Simulation aids in discovering design errors. VHDL, Verilog, and SystemVerilog are three examples of hardware description languages and are considered in this mapping study due to their industrial applicability and popularity [9]. The sample code given in Figure 1 describes the structural description that specifies all the details at the level of elementary gates.

2.2. Testing of Digital Systems

Developing a digital circuit starts with a design specification in a high-level design language (e.g., Verilog) and ends with manufacturing the system. Different testing processes are involved in each development stage. Testing and debugging of the design using the HDL model as input in a simulator can assure the designer that the description of the design meets the specification. The primary goal of this step is related to design functionality, and detailed timing and physical faults are not addressed. After obtaining a simulated and tested layout, manufacturing testing is done on the physical circuit against manufacturing defects (e.g., transistor defects, broken wires). In this paper, we focus on testing digital systems represented in an HDL (i.e., HDL-based Testing).

To verify a digital system design, a test module needs to be designed to generate stimuli for the input of the System Under Test (SUT), execute the test cases, and monitor the inputs and the outputs of the circuit. Wang [6] defines testing of digital circuits as the process of detecting faults in a system and locating such faults to facilitate their repair. A set of test stimuli called test vectors or patterns are applied to the inputs of the SUT. The outputs are analyzed by collecting the responses and comparing it to the expected responses. If the actual behavior deviates from the expected behavior, then a fault has been detected. In Figure 2, we show the general principles behind test generation for digital systems. Testing of digital circuits can be classified based on the type of device being tested, the testing equipment used, and the purpose of testing. Different testing tasks can be performed using the HDL testing constructs and capabilities. An example of a test generation method is the random generation of test vectors [10], applying them to the SUT, and selecting these based on their effectiveness in detecting faults. In some cases, a specialized test device can be used to apply test vectors on the actual test interface [1]. Virtual test platforms are usually implemented as HDL testbenches that are capable

of manipulating the SUT in terms of observability. A main characteristic of these test generation approaches is the HDL model coverage criteria used to guide the search for test cases. Most coverage metrics for HDLs are based on syntactic properties of a functional design. For example, branch and path coverage are used for finding test sequences covering all branches and paths in a HDL design. HDL coverage criteria is also a measure of register-transfer level adequacy that can also be applied at gate-level for measuring test adequacy at the implementation level.

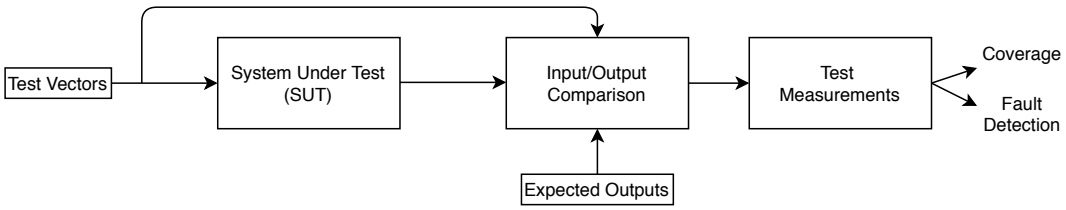


Figure 2. An overview of the test generation process for a SUT.

Many different test generation approaches for HDLs have been proposed at different levels of design abstractions: system, behavioral, register transfer, logical and physical levels. In this study, we aim to map this area in terms of the methods used to generate test cases for certain HDLs.

3. Mapping Studies

Different methods have been proposed for verification and validation of circuits designed in VHDL or Verilog HDLs. To the best of our knowledge, this is the first mapping study on the topic of test generation for digital systems. A mapping study allows researchers to collect data for providing an overview of the test generation research area to assess the quantity of evidence existing on this topic [11]. This research method has been used in the software engineering and testing community to identify the evidence available and to identify gaps in the application of test generation. For example, Divya et al. [12] conducted a mapping study on built-in self testing in component-based software.

4. Conducting the Mapping Study

A mapping study is conducted to organize and classify the literature in a specific domain [13]. In our case, the mapping study is performed with the goal of organizing and classifying the literature in the domain of test generation for digital circuit designs. We have adapted the mapping study method proposed by Petersen et al. [14]. We have used a five-step method depicted in Figure 3. Each step has a certain outcome associated with it and these are represented in Figure 3. Each of the steps are described in the following subsections. These five steps are covering the definition of research question (shown as (1) in Figure 3), conducting a search for articles ((2) in Figure 3), screening the papers based on an inclusion and exclusion criterion ((3) in Figure 3), classification of the articles ((4) in Figure 3), and data extraction and analysis ((5) in Figure 3).

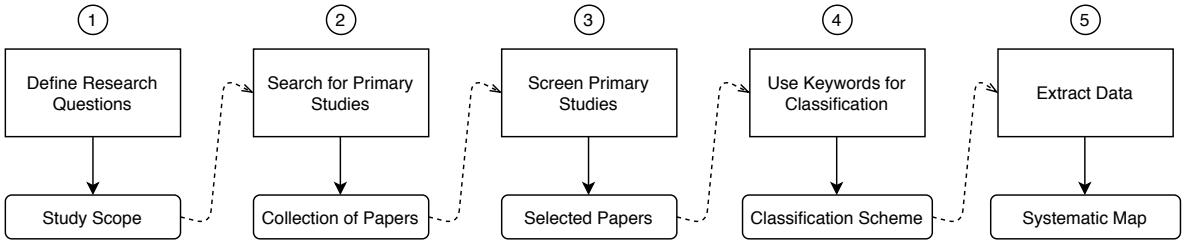


Figure 3. The mapping study steps performed in this study.

We start by defining the research question in order to clearly define the scope of this study. A search string is created in order to search for the bulk of papers in the test generation research area. Terms related to this area are used when forming a search string so that a representative set of papers

gets through the selection phase. These papers are now filtered by applying the defined inclusion and exclusion criteria. Keywording is the process of analyzing the abstracts and keywords of the papers collected with the goal of investigating if a paper fits into the mapping study. Papers where the inclusion decision is based on abstracts or the full text are read thoroughly. A final set of primary studies are collected at the end of this step. The primary studies in the final set of papers are reviewed with the intention to seek answers for the research questions defined. The collected data is presented in a quantitative and qualitative manner.

4.1. Definition of Research Questions

The objective of this mapping study is to identify, analyze, and classify the methods for test generation on digital circuits associated with Verilog, VHDL, and SystemVerilog HDLs. The following research questions are derived from this objective.

- **RQ 1.** What are the publication trends in the domain of test generation for digital circuits represented using Verilog, VHDL, and SystemVerilog?

  Goal. The goal with this research question is to evaluate the interest, publication types, and venues targeted by researchers over the years.
- **RQ 2.** What are the different abstraction levels at which the test generation methods are proposed?
  Goal. The goal with this research question is to identify and classify the research based on abstraction levels for testing digital circuits associated with VHDL, Verilog, and SystemVerilog HDLs.
- **RQ 3.** What are the test methods used to generate tests for digital circuits and implemented as software tools?
  Goal. The goal with this research question is to map the methods of testing digital circuits proposed in terms of the available tooling.
- **RQ 4.** Which are the different test purposes and goals used in test generation for digital circuits?
  Goal. The goal with this research question is to identify the test goals in each of the primary studies.
- **RQ 5.** What are the test technologies used in test generation for digital logic circuits?
  Goal. The goal with this research question is to identify the technologies on which the test generation techniques are based on.
- **RQ 6.** Are the test generation methods for digital circuits creating and executing test cases on the SUT?
  Goal. The goal with this research question is to identify the extent of practical realization of the test cases and their execution on the digital system.
- **RQ 7.** What are the metrics used for evaluating the test generation techniques used for digital systems?
  Goal. The goal with this research question is to identify the metrics used to evaluate the efficiency and effectiveness of the proposed test generation techniques.

4.2. Search Method

Searching for primary studies is the first step in our mapping study. It is an iterative process and that is carried out on different databases and indexing systems. The objective of this step is to collect all relevant articles for the study. The selected databases are focusing on this area of research contributions as these return the most relevant publications [15]. We used the following publication databases: ACM digital library, IEEE Xplore digital library, and the Scopus scientific database. Following this step, we framed the search strings based on the research questions. In this study, the search string was developed based on the scope of the research questions, including the search terms, population (i.e., application area), and intervention (i.e., techniques used) [14]. VHDL and Verilog are the expected keywords that will include all documents with these HDLs in title, abstract, or keywords. The rationale

for using the terms VHDL and Verilog is that this study will cover studies that discuss these HDLs upfront. A threat to the validity of this search can relate to the HDL language not necessarily being highlighted in a paper. To restrict the focus of this mapping study, we considered only those methods that support Verilog and VHDL as their primary HDL input and mention these languages in their title or abstract. The term "test generation" has been used to include studies focusing on automated test generation. This is a generic term used in both hardware and software testing. These keywords are combined using the logical operators AND and OR. Each chosen database has its own syntax for the search strings, and we adapted these to exercise each particular database. The search strings applied to each of the three databases are shown as follows:

- Search String IEEE: *((((VHDL) OR (verilog)) AND ("test generation")))*;
- Search String ACM: *("VHDL" "verilog" +"test generation")*;
- Search String Scopus: *TITLE-ABS-KEY ("VHDL" OR "verilog" AND "test generation")*.

We mention here that these search strings were subject to multiple iterations. While other search keywords can be used to refer to Verilog designs (e.g., netlists) or to automated test pattern generation techniques (e.g., ATPG), we argue that relevant studies on test generation of VHDL and Verilog designs will contain these generic keywords in their data fields. To obtain evidence on this matter, we analyzed the results obtained using these keywords (i.e., ATPG and netlists) and crosschecked them against our original results and randomly selected papers. While searching for articles on the IEEE Xplore digital library, the search was restricted to the available metadata, and, while performing this for the Scopus scientific database, the search was restricted to titles, keywords, and abstracts. The search process was performed for the time period until 2018. This resulted in a total number of 1131 scientific publications. Out of these, 932 publications were obtained from the ACM digital library, 105 publication from Scopus scientific database, and 94 from the IEEE Xplore digital library. In Figure 4, we show the different steps taken from the initial search to the selection of a final set of papers. In the following subsections, we detail each of these steps.

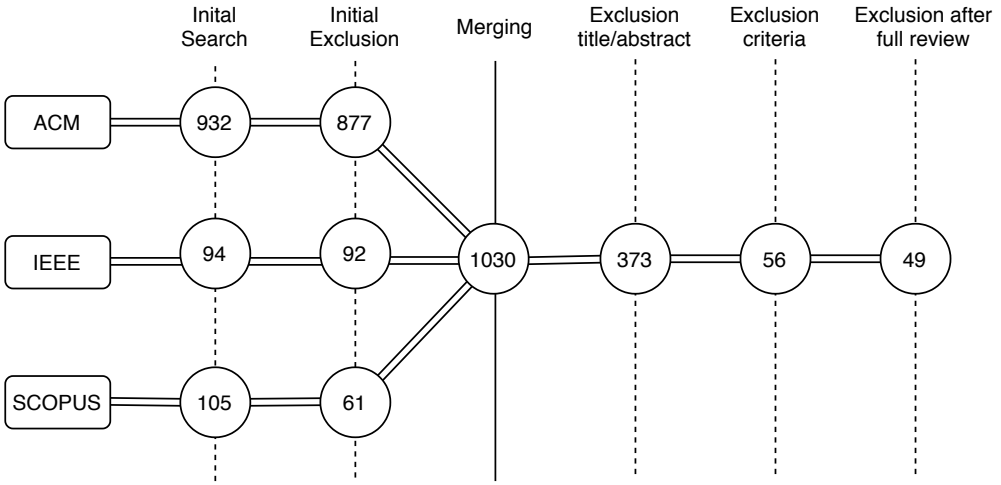


Figure 4. Mapping study steps for searching and selecting the final set of papers.

4.3. Screening the Articles

In this step, we filter the primary studies based on the inclusion/exclusion criteria. The following criteria were applied for the mapping study:

- The paper is a scientific publication and not an expert opinion or a summary.
- The paper has been written in English.
- The paper could be accessed as full-text.
- The paper is related to digital system testing.
- The paper presents test generation methods focusing on VHDL, Verilog, or SystemVerilog HDLs.

Figure 4 describes the selection step in detail. From the ACM, IEEE, and Scopus databases, we obtained 877, 92, and 61 publications, respectively, after excluding the duplicates. In the succeeding steps, we merged these 1030 resulting publications. Excluding the publications based on titles and abstracts led to the elimination of 657 publications, and this resulted in 373 papers (out of the 373 extracted papers, 29 were from IEEE, 35 were from Scopus, and 309 were from ACM). This step was complicated by the fact that, for ACM, the abstract could not be downloaded in any format while exporting the articles from this database. To address this issue, we extracted these abstracts from ACM separately using our own filters. It has to be noted that ACM was the database that contributed with the largest number of publications. Papers were read, and the abstract and titles that clearly indicated that the contributions were outside the focus of this mapping study were excluded. For example, the search term "verilog" retrieved studies about verilog in simulation and synthesis related to testing. If a title or abstract did not understandably uncover the test generation focus of the paper, it was included for review in the subsequent stages. In the next step, by reviewing the title and abstract using the inclusion and exclusion criteria above, we obtained 56 publications. Out of the 373 extracted papers from the previous stage, 294 papers were not directly related to digital-system/circuit testing and 23 were not directly focused on VHDL, Verilog, or SystemVerilog, resulting in the remaining 56 publications. In this stage, one author read the titles and abstracts of the papers for each criterion, and another one examined their relevance to this mapping study. In the next step, we excluded 7 articles since some of these contributions were not available in full-text or the results were already contained in other extended papers already included. In the end, we obtained 49 primary studies that matched our research questions.

4.4. Classification

To extract information from all primary studies and answer our research questions, a classification needs to be defined. To answer our research questions, the primary studies were classified using the following six different facets based on the automated test generation categorization for digital circuits proposed by Zander et al. [16] and Utting et al. [17] for model-based test generation approaches:

1. *Design Level*. In order to manage complexity, testing activities can be considered at different levels of design abstractions.
2. *Test Generator Tooling*. Implementation details of the test generator in terms of the architectural and tooling details used to develop and run the test generator.
3. *Test Goal*. This dimension defines the goal used to control the generation of test cases. Test generators can be classified according to which kinds of goals and test generation criteria they support.
4. *Test Generator Technology*. This dimension refers to the test generator capability to derive test cases using different search technologies, such as: graph search, random, search-based techniques, model checking, and symbolic execution.
5. *Test Execution*. This dimension is concerned with test execution and the actual implementation of test cases in a testbench.
6. *Test Evaluation*. The last dimension refers to the type of test oracle (comparison with the expected behavior and logic) used and in which conditions the evaluation takes place.

The classification scheme was developed so that the primary studies could be categorized into these facets and data extraction could be carried out. A taxonomy is the main classification mechanism used during data extraction. Taxonomies can be enumerative or faceted [18]; in enumerative, the classification is fixed, and it is not ideal for an evolving research area, such as the automated test generation for digital systems area. Hence, we used the faceted classification scheme shown above where the aspects of categories can be combined or extended and are generic enough. The facets are drawn from the research questions mentioned in Section 4.1 and adapted from both the digital system testing literature, as well as the model-based testing community. Apart from the above-mentioned facets, we considered basic information from each of the primary studies to answer RQ1. The titles and publication details were collected as basic information from each of the primary studies. The facets contained basic classification attributes and evolved throughout the process of the mapping study. In Section 4, we discuss each of the facets and show examples.

4.5. Data Extraction and Analysis

The objective of this step is to analyze the data extracted from the primary studies. In addition, the direct connections between each of the primary studies and the research questions are derived in this step. The results of the analysis are presented in qualitative and quantitative form (e.g., graphical or tabular form). The outcomes of the mapping study are discussed in Section 5. The possible threats to validity are also considered. We describe in Table 1 the details of the data extraction and analysis.

Table 1. Information about data extraction and analysis.

No.	Description	Details
1	Bibliographic Info	Title, author, publication venue and type (i.e., a conference paper or journal)
2	Test Goal	Stated test purpose, including the stopping criteria.
3	Tooling	Test generator implementation details.
4	Test Level	Abstraction level at which the test generator works.
5	Technology	Test generation technology characteristics.
6	Test Execution	Information about the execution of generated test cases
7	Test Evaluation	Information about the evaluation of the generated test cases
8	Classification	Relevance to the predefined categories

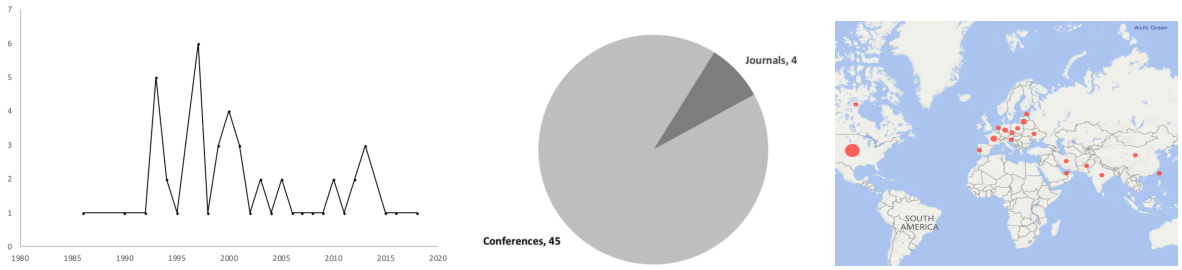
The data extraction for this mapping study is defined in Table 1 (i.e., 2 to 7, and the synthesis is carried out in 8). In the final step, we kept track of the data extraction from each of the primary studies. The main focus of the analysis is to answer the predefined research questions.

5. Results

We performed the mapping study based on the procedure described in Section 3. In this section, we present the results. Each of the subsections aim at answering the research questions mentioned in Section 4.1.

5.1. Publication Trends

We focused on the publication trends by collecting data, such as publication year, type, and venue for each of the primary studies. The trends are visualized in Figure 5a as a temporal distribution during the years. In Figure 5b, we show a pie chart depicting the publication types, while, in Figure 5c, we show a map chart by conference venues.



(**a**) Distribution of publications by Year (**b**) Type of Publication (**c**) Conference Venues

Figure 5. Distribution of publications by year, type, and conference venues.

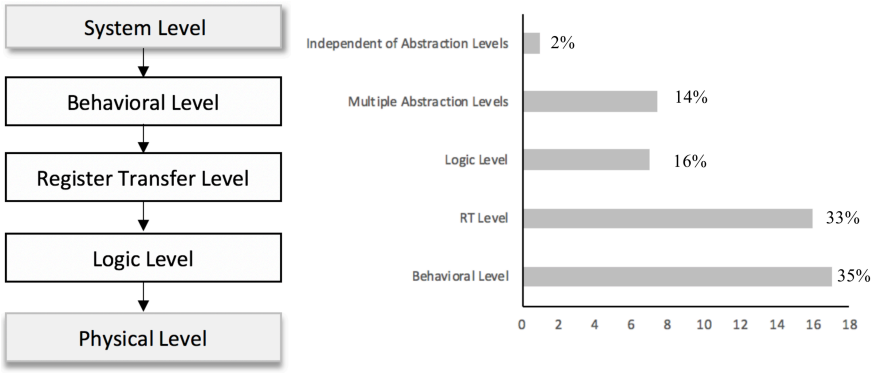
The distribution of the publications in Figure 5a spans the time between 1986 to 2018. The first study on test generation for VHDL- and Verilog-designed digital systems was published in 1986. The interval between 1993 and 2002 saw a growing interest in the subject among researchers, with an average of 5 papers published. Until 1993, we see a decreasing number of papers published on test generation for digital circuits. The most recent upsurge in publications has started on 2013. As an example, these recent publications mostly focused on fault independent test generation and coverage directed test generation techniques [19,20]. We can observe a decrease in the number of publications from 2015 onward. We also collected data about the publication type. The results clearly suggest that the contributions in test generation for Verilog- and VHDL-designed digital systems are mainly published in conference venues (i.e., 92% of the publications are conference papers, and only 8% are journal publications). We also looked at the publication trends by venues. We categorized the data using the countries in which these conferences were organized. The USA was found to be the country with the most conference venues in which test generation for digital systems papers have been published (i.e., 21 publications). France is following by six publications, Germany and Lithuania have three publications, India has two publications, and the rest have one publication each.

In addition, to assess the impact of the collected data in this research area, we collected the citation count for each paper. We used Google Scholar citation count based on an automatic search on 6.07.2020. Our results suggest that papers dealing with test generation for Verilog, VHDL, and SystemVerilog designs have a citation count of 21 on average. The paper with the highest citation count (i.e., 131) by Duale and Uyar [21] is targeting VHDL designs and has been published in the IEEE Transactions on Computers in 2004. These results shows that the impact of this research area in terms of scientific citations is moderately high for some papers. Nevertheless, several papers have low citation counts, and this shows that the relative academic impact of certain publications in this area is rather low. More studies are needed to investigate the specific publication trends and reasons behind these results.

5.2. Test Level

The design and testing of digital systems is performed at several abstraction levels. The abstraction levels for digital systems are shown in Figure 6 (as adopted from Bengtsson et al. [22]) and Table 2. System level is the highest abstraction level at which the physical circuit is realized. CAD tools are develop to aid in the transformation of a circuit from a higher abstraction level to a lower level. The system level is sometimes called a functional level since the system constraints and functions are specified (e.g., performance and cost). The behavioral level or the algorithmic level is where the digital systems are designed in terms of the algorithms and detailed functions used. At this level, there is no time measurement in terms of clock cycles. In addition, at the RT level, the algorithms are described in more detail, including the data-paths and controllers. A data-path consist of functional units, vector multiplexers and registers. The functional units that require several clock cycles are described at this level. The controller, as the name suggests, generates load signals for registers and controls the functional units in each data-path. The RT-level is the abstraction level at which the functions are defined at each clock cycle. Digital systems are designed at the logical level in terms of their logical synthesis. At this level (also known as the Gate level), the RTL descriptions are

transformed into logic gates and flip flops. After this synthesis, the digital system is designed at the physical level using a technology independent description in the logic level that is transformed into a physical implementation.



(**a**) (**b**)

Figure 6. Overall results based on abstraction levels. (**a**) Abstraction levels; (**b**) Distribution of papers based on abstraction levels.

To answer RQ2, we categorized each of the primary studies by the test level these test generation methods are focusing on. During the data collection phase, we observed that just a few studies explicitly mentioned the targeted test level. One author read the papers in full-text to identify this information. We show the overall results in Figure 6b. We notice that the primary studies fall under all the five levels. The answer to RQ2 was found with the categorization of the studies into these levels. We discovered that there is a sizeable gap between studies on behavioral level and logic level test generation methods. Our results suggest that the behavioral level with 17 studies and the RT level with 16 studies are the most targeted levels by test generation researchers. One possible reason for this could be explained by the fact that these techniques yield better results in terms of efficiency and effectiveness early in the development process at higher levels of abstraction. To better understand the categorization of these techniques, we present examples for each of the test level.

Hayne et al. [23] proposed a test method focused on targeting the coverage of faults at the gate level. The method devised in this paper applies the test vectors derived from the behavioral level fault models to gate level realizations of a range of circuits, including arithmetic and logical functions. The results of this study suggest that using test generation for VHDL can yield better gate level fault coverage.

Gent et al. [20] proposed a method for test generation at the RT level. This method uses a combination of a stochastic search technique and a deterministic search aimed for RTL design validation. The effectiveness of this techniques is measured using branch coverage.

Gulbins et al. [24] proposed a novel approach to functional test generation applied at the behavioral level using the knowledge of links between inputs and outputs. The behavioral level fault model that is considered by the authors can handle faults in both the data flow and the control parts of the digital system. The test generation method is implemented in a tool called GESTE and applied to generate symbolic tests at the behavioral level. Another study from Pla et al. [4] proposed a method to generate test from behavioral circuit descriptions that can be used in other level of abstractions. This method is based on formal modeling of a VHDL description using two level-independent models: the input/output model and the activation model.

Since not all faults can be modeled at the behavioral level, some methods focused on modeling only after the synthesis to a lower level [25]. In this way, faults are modeled on one level and after the synthesis the faults are mapped back. Such a hierarchical test generation method is proposed by Tupuri et al. [26]. Test cases are generated for faults using commercial ATPGs for individual modules. Functional constraints extracted for each module are described in Verilog or VHDL and applied to

the synthesized gate level. As a last step, the module level vectors are translated to processor level functional vectors. Hence, the process of test generation is performed at multiple levels.

Table 2. Categorization of papers into abstract levels.

Papers	Level
[19,24–40]	Behavioural
[20,21,34,41–53]	RT
[23,54–58]	Logic
[52,59–65]	Mixed
[4]	Independent of any abstraction level

5.3. Tool Implementation

In order to answer RQ3, we collected data from the primary studies related to the tool implementation details for each test generation method. We intended to map the methods implemented as tools that can be used by both practitioners and researchers. This dimension is used to evaluate the level of applicability and practicality.

Our results suggest that 31% of the studies implemented their methods in a tool for testing digital circuits. Table 3 provides an overview of the tools implemented for test generation for VHDL and Verilog. One example of such a tool is proposed by Murtza et al. [25] and called VertGen. We considered other methods, such as the ones proposed by Lu et al. and Rao et al. [30,48], which propose and show the applicability of these tools for the method applied to test digital circuits.

Table 3. List of method for test generation implemented as software tools.

Paper	Tool Name	Tool Description
[60]	FACTOR	Functional constraint extractor is implemented for hierarchical functional test generation for complex chips
[57]	Test Synthesis Tool	The tool employs an RTL model written in Verilog or VHDL to drive the test generation process.
[25]	VertGen	VertGen is an automatic testbench generation tool for Verilog.
[31]	Remote Test Generation Tool	The tool uses a fault simulation to generate tests based on transmission models. A transmission model represents the functions carried out by a system and is not considering clock and reset information.
[34]	A Constraint Logic Programming (CLP) Tool	The tool generates computation constraints, fault constraints for a VHDL model. Together these constraints form the ATPG constraints whose solution produces test sequences.
[61]	Gate Level Test Enhancement Tool	The tool combines software testing-based methods at high level with test enhancement methods at gate level. The tool maximizes coverage of single stuck-at faults.
[42]	Prototype Test Processor for Functional Testing	The tool can generate mixed-mode test vectors.

Table 3. *Cont.*

Paper	Tool Name	Tool Description
[24]	The GESTE tool	The tool supports the generation of symbolic tests by constructing a path through the graph model.
[38]	AgenMix	The tool automatically generates test mixes.
[47]	FSM from VHDL descriptions.	Verification test generation using a circuit model.
[48]	Virtual Test Environment (VTE)	An open architecture VTE is presented that can be integrated into various automatic test systems.
[64]	VPI based tool	The tool is based on Verilog Procedural Interface that performs module based mixed level fault simulation and test generation.
[52]	Behavioral Test Generator	The tool uses path enumeration and constraint programming techniques to generate design verification tests.
[53]	ATKET	Automatic Test Knowledge Extraction Tool synthesizes test knowledge using structural and behavioral information available in VHDL descriptions of a design.
[65]	Logic Development Tool	A system that provides a link between design and test and synthesizes the test vectors from VHDL.

5.4. Test Goal

The test goals used by test generation methods vary depending on the level on which testing is performed. The primary level used for simulation is the Register Transfer (RT) level. Testing at this level is focusing on detecting only functional defects [1]. In some approaches, test generation involves generating test vectors that can quickly and accurately identify defects. Navabi el al. [1] provides an overview on how test generation can be used in several ways for a given circuit. There are just a few methods that use circuit topologies and functional models of a circuit.

In this mapping study, we were interested at reviewing the different objectives used by test generation methods for digital circuits. We based this dimension on the categories shown by Navabi et al. [1]. We collected data regarding the test goals in the data extraction phase and categorized them into one of the following subdimensions:

1. Fault Oriented Test Generation. This is a method suitable for detecting certain defects. Fin et al. [54] proposed a fault-oriented test generation method that uses an efficient error simulator that is able to analyze functional VHDL descriptions.
2. Fault Independent Test Generation. This method automatically creates tests for a large class of faults. A test is generated and then it is evaluated if it can detect certain injected faults. This method of test generation is useful when the SUT contains many possible defects [1]. Vedula et al. [60] describe a functional test generation technique for a module embedded in a large design.
3. Random Test Generation (RTG). Random test generation is a method in which the test vectors are selected using a random strategy. RTG techniques are complemented with evaluation procedures that aid in the selection of test vectors. Often, RTG programs target an area of a SUT that contains faults that are hard to detect [1]. For example, Shahhoseini et al. [50] provided evidence that their algorithms are simple and useful for efficient test generation.
4. Coverage Directed Generation. This method of generating tests targets a specific coverage metric, such as statement, branch, or condition coverage of the HDL representation. For example, Ferrandi et al. [59] developed a test generation algorithm that targets a coverage metric named bit

coverage. The bit coverage includes full statement coverage, branch coverage, condition coverage, and partial path coverage for behaviorally sequential models.

In Figure 7, we show the distribution of the primary studies under the four test goal categories. Our results suggest that 25 publications proposed methods for test generation that are targeting a fault independent test goal, while 15 other studies focused on generating test cases for a fault-dependent test goal. It seems that 10% of the papers focused on coverage directed test generation methods, 6% used random test generation, and 31% of the methods generated tests using a fault-oriented goal, while 51% of the papers focused on fault independent test generation. It seems that random test generation is only used in a handful of studies, while half of the methods focus on test generation methods for fault-independent detection as a test goal.

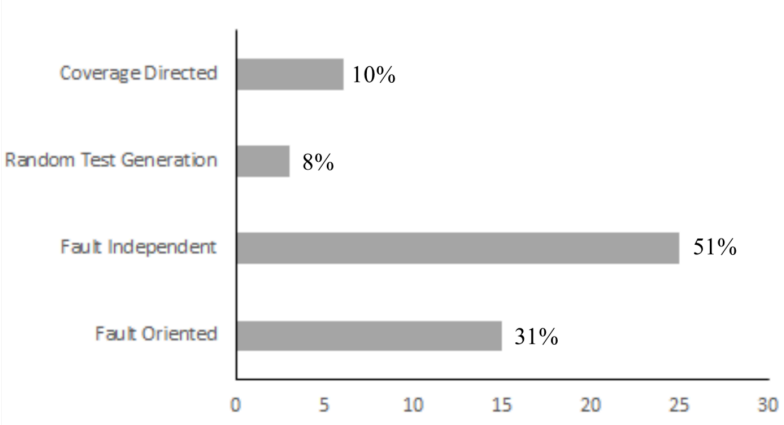


Figure 7. Distribution of studies based on the test goal.

5.5. Technology

In this subsection, we aimed at answering the RQ5 related to the search engine technology behind the test generation methods. Different technologies are used in these tools, such as simulation, constraint-based test methods, path-based test methods, and model checking, just to name a few. In the process of conducting the mapping study, we considered first the information from all primary studies and established these categories a-posteriori in broader terms. This was done to identify the number of publications under each of the technology facets and to check if there is an emphasis on one particular technology. The technology used in the primary studies was categorized as follows:

- Model Simulation—26 primary studies—54%
- Model Checking—6 primary studies—13%
- Automatic test pattern generation—3 primary studies—6%
- Constraint-based test generation—4 primary studies—9%
- Behavioral test generation—1 primary study—2%
- Path-based test generation—2 primary studies—4%
- Test-Bench Generation—2 primary studies—4%
- Test-Sequence Compaction—3 primary studies—6%
- Pseudo-Exhaustive Execution—1 primary study—2%

Overall, our results suggest that model simulation is the most widely used test generation technology with 54% of primary studies focusing on it. In Table 4, we show a classification of the studies based on the different technologies identified. In more than half of the approaches, a model is developed for the SUT and then this model is executed in simulation to find possible faults in the system by test generation. In Reference [54], the authors present an error simulator.

Model checking is another popular method for exhaustive verification. Murugesan et al. [43] proposed a satisfiability-based test generation for stuck-at faults. Automatic test pattern generation operates by injecting fault into a circuit and then activates the fault followed by propagation to the output. As a result, the output flips from expected to a faulty signal.

Constraint-based test generation involves conversion of a circuit model into a set of constraints and developing constraint solvers to generate tests. Constraint-based test generation has been mainly used for behavioral level functional test generation. Cho et al. [29] describe a behavioral test generation algorithm that generates tests from the behavioral VHDL descriptions using stuck at open faults and operation faults.

Path-based test generation is a method dependent on the control flow graph of a program that is using the selection of control flow paths using a certain test criterion. Other techniques are focusing on test-benches by generating stimulus waveform and comparing the reference output with that given by the design under test. In other approaches, Test sequence compaction is used to remove test vectors that are not necessary in the test sequence, thereby reducing the length of the test sequence. In addition, at the higher level of abstractions, exhaustive functional testing can be used to detect faults due to the relative simplicity of the state space. Given that a huge number of test patterns are required, it is not suitable to be used for industrial applications. One approach focused on reducing the number of test patterns needed while maintaining a high fault coverage using pseudo-exhaustive executions.

Table 4. Classification of the studies under the different technologies identified.

Papers	Technologies
[19,21,23,24,28,30,31,33–35,39,41,44,45,48–51,54,55,57–59,64,65]	Model Simulation
[4,20,36,37,43,47]	Model Checking
[28,38,53]	Automatic Test Pattern Generation
[26,34,52,60]	Constraint Based Test Generation
[29]	Behavioral Test Generation
[32,56]	Path Based Test Generation
[25,40]	Test Bench Generation
[42,61,63]	Test Sequence Compaction
[46]	Psuedo Exhaustive Execution

5.6. Test Execution

Test execution is another part of the classification scheme used in our mapping study. We consider test execution as an aspect of applicability of test generation techniques. Navabi [1] considers two of the most common methods of testing: scan testing and boundary scan testing. Test cases can be executed using a Built In Self Test (BIST) in which a hardware structure applies the test vectors to the SUT. Fault coverage is used as a measure when evaluating the effectiveness of BIST.

The results suggest that 63% of the studies execute the generated tests. The other approaches which are generating abstract test cases are just proposing certain algorithms and provide preliminary results on test effectiveness.

5.7. Experimental Evaluation

Evaluation is the process of measuring the efficiency and effectiveness of the generated test cases with the aid of metrics, such as coverage, generation time, number of test cases, and fault detection. We collected data regarding the evaluation of the proposed methods and the metrics used to evaluate these. The results suggest that 37% of the primary studies do not evaluate the proposed methods in an experimental evaluation. A total of 31 primary studies, out of 49 considered for this mapping study, experimentally evaluated their methods in terms of efficiency or effectiveness.

As shown in Figure 8, 18 studies use test coverage as a proxy metric for effectiveness. Other four studies used both generation and execution time as a measure of test efficiency. Fault detection capability has been used in three studies as a proxy measure of effectiveness. It seems that coverage criteria is the most common metric used for test suite evaluation. Rao et al. [30] propose a hierarchical test generation algorithm (HBTG) and evaluate their technique using only statement coverage. Riahi et al. [66] used fault coverage to measure the effectiveness of a random test vector and simulation run time as a measure of efficiency. Lynch et al. [44] proposed an automatic test pattern generation (ATPG) system and evaluated in terms of faults detected and isolated.

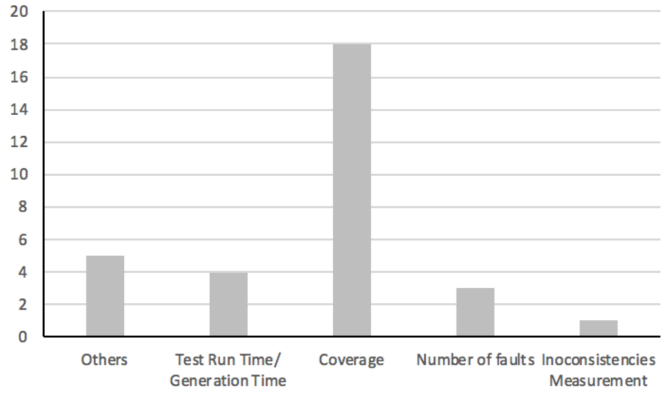


Figure 8. Distribution of studies under different evaluation metrics.

When a data-flow graph representation is used to generate test sequences, some of the test may not be feasible. Inconsistencies may be present due to the conditional statements or variables used in actions [33]. A measure of detecting such inconsistencies is an evaluation metric [33].

Overall, 58% of the primary studies evaluate the test generation methods using the following types of test coverage criteria: combined coverage, code coverage (e.g., statement coverage, branch coverage), and path coverage. Some studies are using statement coverage for measuring test quality, which has been shown to be a rather weak metric [30].

6. Limitations and Threats to Validity

This section discusses the threats and outlines some strategies used to mitigate these (c.f. Reference [67]). Construction validity relates to the design of the research method and the search string. In our mapping study, we adopted the research method of Petersen et al. [14]. The search string was formed to retrieve studies for different databases, and we used well-known test characteristics and categories. External validity relates to the degree to which the results can be generalized. We minimize this threat by following a rigorous research methodology and involving an experienced researcher in data analysis. Internal validity relates to the measured or collected relationships and factors. As the aim of the study was not to establish a causal relationship on test generation methods, it is not considered a threat to this study. Conclusion validity relates to researcher bias in the interpretation of the data. We reduced the risk by involving two authors in the final analysis. Although the results are limited by the studies included in the used databases, they covered a wide range digital system literature.

7. Conclusions

We presented the results from a systematic mapping study to map the landscape of test generation methods for digital systems designed in Verilog, VHDL, and SystemVerilog HDLs. The results of our systematic mapping study focused on seven different facets: the publication trends, test level, tool implementation, test goal, test generation technology, test execution, and the experimental evaluation. We considered 49 primary studies to conduct this mapping study. Our results suggest a considerable growth in interest in the domain of test generation for digital circuits. The mapping study revealed that there is an emphasis on test generation for digital circuits at behavioral and RT Level. A majority of test methods focus on fault independent test generation as their test goal and depend on model simulation as a technology for experimental test execution.

Author Contributions: The first two authors contributed equally to the research goals definition, approach, study design, and reporting of the research work. In addition, the first author contributed with data collection and data analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No. 737494 and the Swedish Innovation Agency (Vinnova) through the MegaM@Rt2 and XIVT projects.

Acknowledgments: We would like to thank the anonymous reviewers for their useful suggestions and comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Navabi, Z. *Digital System Test and Testable Design*; Springer: Berlin, Germany, 2011; pp. 97814419–97875485.
2. Lala, P.K. An introduction to logic circuit testing. *Synth. Lect. Digit. Circuits Syst.* **2008**, *3*, 1–100. [CrossRef]
3. Jha, N.K.; Gupta, S. *Testing of Digital Systems*; Cambridge University Press: Cambridge, UK, 2003.
4. Pla, V.; Santucci, J.F.; Giambiasi, N. On the modeling and testing of VHDL behavioral descriptions of sequential circuits. In Proceedings of the EURO-DAC 93 and EURO-VHDL 93-European Design Automation Conference, Cch Hamburg, Germany, 20–24 September 1993; pp. 440–445.
5. MacMillen, D.; Camposano, R.; Hill, D.; Williams, T.W. An industrial view of electronic design automation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2000**, *19*, 1428–1448. [CrossRef]
6. Wang, F.C. *Digital Circuit Testing*; Academic Press: Cambridge, MA, USA, 1991.
7. Stefan, G. *Loops & Complexity in Digital Systems*; Lecture Notes on Digital Design; 2016. Available online: http://users.dcae.pub.ro/~gstefan/2ndLevel/teachingMaterials/0-BOOK.pdf (accessed on 30 May 2020).
8. Ghosh, S.K. *Hardware Description Languages: Concepts and Principles*; Wiley-IEEE Press: Hoboken, NJ, USA, 1999.
9. Golson, S.; Clark, L. *Language Warsin the 21st Century: VerilogversusVHDL–Revisited*; Synopsys Users Group (SNUG): Beijing, China, 2016.
10. Haghbayan, M.; Karamati, S.; Javaheri, F.; Navabi, Z. Test pattern selection and compaction for sequential circuits in an HDL environment. In Proceedings of the 2010 19th IEEE Asian Test Symposium, Shanghai, China, 1–4 December 2010; pp. 53–56.
11. Ahmad, A.A.S.; Brereton, P.; Andras, P. A systematic mapping study of empirical studies on software cloud testing methods. In Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), Prague, Czech Republic, 25–29 July 2017; pp. 555–562.
12. Gill, N.S.; Singh, L.; others. Built-in testing in component-based software-a mapping study. In Proceedings of the 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 11–13 March 2015; pp. 159–168.
13. Keele, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering, Ver. 2.3*; EBSE Technical Report EBSE-2007-01; University of Durham: Durham, UK, 2007.
14. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic mapping studies in software engineering. *Ease* **2008**, *8*, 68–77.
15. Brereton, P.; Kitchenham, B.A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.* **2007**, *80*, 571–583. [CrossRef]

16. Zander, J.; Schieferdecker, I.; Mosterman, P.J. *Model-Based Testing for Embedded Systems*; CRC Press: Boca Raton, FL, USA, 2011.
17. Utting, M.; Pretschner, A.; Legeard, B. A taxonomy of model-based testing approaches. *Softw. Testing Verif. Reliab.* **2012**, *22*, 297–312. [CrossRef]
18. Usman, M.; Britto, R.; Börstler, J.; Mendes, E. Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method. *Inf. Softw. Technol.* **2017**, *85*, 43–59. [CrossRef]
19. Klimenko, A.; Gorelova, G.; Korobkin, V.; Bibilo, P. The Cognitive Approach to the Coverage-Directed Test Generation. In *Proceedings of the Computational Methods in Systems and Software*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 372–380.
20. Gent, K.; Hsiao, M.S. Functional test generation at the RTL using swarm intelligence and bounded model checking. In Proceedings of the 2013 22nd Asian Test Symposium, Yilan, Taiwan, 18–21 November 2013; pp. 233–238.
21. Duale, A.Y.; Uyar, M.U. A method enabling feasible conformance test sequence generation for EFSM models. *IEEE Trans. Comput.* **2004**, *53*, 614–627. [CrossRef]
22. Bengtsson, T.; Kumar, S. *A Survey of High Level Test Generation: Methodologies and Fault Models*; Ingenjörshögskolan, Högskolan i Jönköping: Jönköping, Switzerland, 2004.
23. Hayne, R.J. Presynthesis test generation using VHDL behavioral fault models. In Proceedings of the 2011 Proceedings of IEEE Southeastcon, Nashville, TN, USA, 17–20 March 2011; pp. 264–267.
24. Gulbins, M.; Straube, B.; Elst, G. A model for functional test generation and testability analysis. In Proceedings of the ETC 93 Third European Test Conference, Rotterdam, The Netherlands, 19–22 April 1993; pp. 515–516.
25. Murtza, S.A.; Hasan, O.; Saghar, K. Vertgen: An automatic verilog testbench generator for generic circuits. In Proceedings of the 2016 International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 18–19 October 2016; pp. 1–5.
26. Tupuri, R.S.; Abraham, J.A. A novel hierarchical test generation method for processors. In Proceedings of the Tenth International Conference on VLSI Design, Hyderabad, India, 4–7 January 1997; pp. 540–541.
27. Pikula, T.; Gramatova, E.; Fischerova, M. Deterministic test generation for digital circuits by cellular automata in a Java applet. In Proceedings of the IEEE Region 8 EUROCON 2003. Computer as a Tool, Ljubljana, Slovenia, 22–24 September 2003, Volume 2, pp. 40–43.
28. Gulbins, M.; Straube, B. Applying behavioural level test generation to high-level design validation. In Proceedings of the 1996 European Conference on Design and Test. IEEE Computer Society, Washington, DC, USA, March 11–14 1996; p. 613.
29. Cho, C.H.; Armstrong, J.R. B-algorithm: A behavioral test generation algorithm. In Proceedings of the International Test Conference, Washington, DC, USA, 2–6 October 1994; pp. 968–979.
30. Rao, S.R.; Pan, B.Y.; Armstrong, J. Hierarchical test generation for VHDL behavioral models. In Proceedings of the 1993 European Conference on Design Automation with the European Event in ASIC Design, Paris, France, 22–26 February 1993; pp. 175–182.
31. Bareisa, E.; Jusas, V.; Motiejunas, K.; Seinauskas, R. Functional test generation remote tool. In Proceedings of the 8th Euromicro Conference on Digital System Design (DSD'05), Porto, Portugal, 30 August–3 September; pp. 192–195.
32. Gharehbaghi, A.M.; Navabi, Z. Behavioral test generation for VHDL processes. In Proceedings of the 12th International Conference on Microelectronics (ICM 2000) (IEEE Cat. No. 00EX453), Tehran, Iran, 31 October–2 November 2000; pp. 123–126.
33. Uyar, M.U.; Duale, A.Y. Modeling VHDL specifications as consistent EFSMs. In Proceedings of the MILCOM 97 Proceedings, Monterey, CA, USA, 3–5 November 1997; Volume 2, pp. 740–744.
34. Xin, F.; Ciesielski, M.; Harris, I.G. Design validation of behavioral VHDL descriptions for arbitrary fault models. In Proceedings of the European Test Symposium (ETS'05), Tallinn, Estonia, 22–25 May 2005; pp. 156–161.
35. Ferrandi, F.; Fummi, F.; Sciuto, D. Implicit test generation for behavioral VHDL models. In Proceedings of the International Test Conference 1998 (IEEE Cat. No. 98CH36270), Washington, DC, USA, 18–23 October 1998; pp. 587–596.

36. Deng, S.; Cheng, K.T.; Bian, J.; Kong, Z. Mutation-based diagnostic test generation for hardware design error diagnosis. In Proceedings of the 2010 IEEE International Test Conference, Austin, TX, USA, 2–4 November 2010; p. 1.
37. Fecko, M.A.; Uyar, M.U.; Duale, A.Y.; Amer, P.D. Efficient test generation for Army network protocols with conflicting timers. In Proceedings of the 21st Century Military Communications. Architectures and Technologies for Information Superiority (MILCOM2000) (Cat. No. 00CH37155), Los Angeles, CA, USA, 22–25 October 2000; Volume 1, pp. 133–138.
38. Hudec, J. An Efficient Adaptive Method of Software-Based Self Test Generation for RISC Processors. In Proceedings of the 2015 4th Eastern European Regional Conference on the Engineering of Computer Based Systems, Brno, Czech Republic, 27–28 August 2015; pp. 119–121.
39. Jusas, V.; Neverdauskas, T. Combining software and hardware test generation methods to verify vhdl models. *Inf. Technol. Control* **2013**, *42*, 362–368. [CrossRef]
40. Jusas, V.; Neverdauskas, T. Novel Method to Generate Tests for VHDL. In Proceedings of the International Conference on Information and Software Technologies, Kaunas, Lithuania, 24 October 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 365–375.
41. Fummi, F.; Rovati, U.; Sciuto, D. Functional design for testability of control-dominated architectures. *ACM Trans. Des. Autom. Electron. Syst. TODAES* **1997**, *2*, 98–122. [CrossRef]
42. Altaf-Ul-Amin, M.; Darus, Z.M. VHDL design of a test processor based on mixed-mode test generation. In Proceedings of the Ninth Great Lakes Symposium on VLSI, Ypsilanti, MI, USA, 4–6 March 1999; pp. 244–245.
43. Murugesan, S.; Ranjithkumar, P. Satisfiability based test generation for stuck-at fault coverage in RTL circuits using VHDL. In Proceedings of the 2010 Second International conference on Computing, Communication and Networking Technologies, Bangkok, Thailand, 23–25 April 2010; pp. 1–6.
44. Lynch, M.L.; Singer, S.M. A next generation diagnostic ATPG system using the Verilog HDL. In Proceedings of the Meeting on Verilog HDL (IVC/VIUF'97), Santa Clare, CA, USA; pp. 56–63.
45. Uyar, M.U.; Duale, A.Y. Resolving inconsistencies in EFSM-modeled specifications. In Proceedings of the IEEE Military Communications. Conference Proceedings (MILCOM 1999) (Cat. No. 99CH36341), Atlantic City, NJ, USA, 31 October–3 November 1999; Volume 1, pp. 135–139.
46. Tang, R.; Si, P.; Huang, W.; Lombardi, F. Testing IP cores with pseudo exhaustive test sets. In Proceedings of the 2001 4th International Conference on ASIC Proceedings (ASICON 2001) (Cat. No. 01TH8549), Shanghai, China, 23–25 October 2001; pp. 740–743.
47. Jusas, V.; Neverdauskas, T. FSM based functional test generation framework for VHDL. In Proceedings of the International Conference on Information and Software Technologies, Kaunas, Lithuania, 13–14 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 138–148.
48. Lu, P.; Glaser, D.; Uygur, G.; Helmreich, K. A novel approach to entirely integrate virtual test into test development flow. In Proceedings of the Conference on Design, Automation and Test in Europe. European Design and Automation Association, Nice, France, 20–24 April 2009; pp. 797–802.
49. Sunkari, S.; Chakraborty, S.; Vedula, V.; Maneparambil, K. A scalable symbolic simulator for Verilog RTL. In Proceedings of the 2007 Eighth International Workshop on Microprocessor Test and Verification, Austin, TX, USA, 5–6 December 2007; pp. 51–59.
50. Shahhoseini, H.S.; Kazerouni, B.H. Semi-Algorithmic Test Pattern Generation. In Proceedings of the IEEE International Conference on Computer Systems and Applications, Washington, DC, USA, 8–11 March 2006; pp. 429–432.
51. Fummi, F.; Sciuto, D. A complete test strategy based on interacting and hierarchical FSMs. In Proceedings of 1997 IEEE International Symposium on Circuits and Systems. Circuits and Systems in the Information Age (ISCAS'97), Hong Kong, China, 9–12 June 1997;Volume 4, pp. 2709–2712.
52. Vemuri, R.; Kalyanaraman, R. Generation of design verification tests from behavioral VHDL programs using path enumeration and constraint programming. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **1995**, *3*, 201–214. [CrossRef]
53. Vishakantaiah, P.; Abraham, J.; Abadir, M. Automatic test knowledge extraction from VHDL (ATKET). In Proceedings of the 29th ACM/IEEE Design Automation Conference, Anaheim, CA, USA, 8–12 June 1992; pp. 273–278.

54. Fin, A.; Fummi, F. A VHDL error simulator for functional test generation. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537), Paris, France, 27–30 March 2000; pp. 390–395.
55. Barclay, D.S.; Armstrong, J.R. A heuristic chip-level test generation algorithm. In Proceedings of the the 23rd ACM/IEEE Design Automation Conference, Las Vegas, NV, USA, 29 June–2 July 1986; pp. 257–262.
56. Shkil, A.; Skvortsova, O.; Mehedy, M.M.; Jahirul, H.H. Test generation for digital device on FPGA, CPLD. In *Experience of Designing and Applications of CAD Systems in Microelectronics, Proceedings of the VI-th International Conference. CADSM 2001 (IEEE Cat. No.01 EX473), Lviv-Slavsko, Ukraine, 12–17 February 2001*; IEEE: Piscataway, NJ, USA, 2001; pp. 83–84.
57. Masud, M.; Karunaratne, M. Test generation based on synthesizable VHDL descriptions. In Proceedings of the EURO-DAC 93 and EURO-VHDL 93-European Design Automation Conference, CCH Hamburg, Germany, 20–24 September 1993; pp. 446–451.
58. Navabi, Z.; Shadfar, M. A VHDL based test environment including models for equivalence fault collapsing. In Proceedings of the VHDL International Users Forum, Oakland, CA, USA, 1–4 May 1994; pp. 18–25.
59. Ferrandi, F.; Ferrara, G.; Sciuto, D.; Fin, A.; Fummi, F. Functional test generation for behaviorally sequential models. In Proceedings of the Design, Automation and Test in Europe. Conference and Exhibition 2001, Munich, Germany, 13–16 March 2001; pp. 403–410.
60. Vedula, V.M.; Abraham, J.A. FACTOR: A hierarchical methodology for functional test generation and testability analysis. In Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 4–8 March 2002; pp. 730–734.
61. Rudnick, E.M.; Vietti, R.; Ellis, A.; Corno, F.; Prinetto, P.; Reorda, M.S. Fast sequential circuit test generation using high-level and gate-level techniques. In Proceedings of the Conference on Design, Automation and Test in Europe, Paris, France, 23–26 February 1998; IEEE Computer Society: Washington, DC, USA, 1998; pp. 570–576.
62. Brera, M.; Ferrandi, F.; Sciuto, D.; Fummi, F. Increase the behavioral fault model accuracy using high-level synthesis information. In Proceedings of the 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (EFT'99), Albuquerque, NM, USA, 1–3 November 1999; pp. 174–180.
63. Ahmady, M.; Sayedi, S.M. Fault coverage improvement and test vector generation for combinational circuits using spectral analysis. In Proceedings of the 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Montreal, QC, Canada, 29 April–2 May 2012; pp. 1–5.
64. Riahi, P.A.; Navabi, Z.; Lombardi, F. Using Verilog VPI for Mixed Level Serial Fault Simulation in a Test Generation Environment. In Proceedings of the Embedded Systems and Applications, Las Vegas, Nevada, USA, 23–26 June 2003; pp. 139–143.
65. Nurie, G.M. Linking VHDL and test. In *Wescon Conference Record*; Wescon: Wichita, KS, USA, 1990; pp. 400–403.
66. Riahi, P.A.; Navabi, Z.; Lombardi, F. The VPI-Based Combinational IP Core Module-Based Mixed Level Serial Fault Simulation and Test Generation Methodology. In Proceedings of the 2003 Test Symposium, Munich, Germany, 7 March 2003; p. 274.
67. Runeson, P.; Host, M.; Rainer, A.; Regnell, B. *Case Study Research in Software Engineering: Guidelines and Examples*; John Wiley & Sons: Hoboken, NJ, USA, 2012.