

Research in Digital Design and Test at Tallinn University of Technology

Raimund Ubar, Gert Jervan, Artur Jutman, Jaan Raik, Peeter Ellervee, Margus Kruus

Abstract — An overview about the recent research results at the Tallinn University of Technology in the field of digital design and test is presented. The main topics discussed in the paper cover digital design, verification, emulation, dependability, fault simulation, and test generation. An experimental research environment is described which consists of prototype tools developed as a side-effect of our research activities. This environment together with a set of dedicated e-learning tools serves also for teaching purposes for the disciplines of design and test of embedded systems.

1. INTRODUCTION

INCREASING complexity of electronic systems has made testing and verification one of the most complicated and time-consuming problems in system design and production. The importance of design for testability is growing because the expenses of testing are becoming the major components of the design and manufacturing costs of new products. It is estimated that more than 70% of the design cycle for systems is spent on test and verification [1]. Nanometer technologies are introducing new challenges making test quality and dependability of systems a very fast moving target [2]. Enhancing productivity and quality of test related solutions is thus a key competitive aspect, both in terms of time-to-market and end-product quality.

In this paper an overview about the recent results in the field of digital test at Tallinn University of Technology (TUT) is presented. One of the most important research areas has been multi-level diagnostic modeling of digital systems by Decision Diagrams (DD) [3]. Using DDs, a hierarchical automated test program generator DECIDER was developed which outpaces similar known academic systems in the speed of test generation [4]. Commercial tools of this type are missing today. A special class of Binary DDs (BDD) called structurally synthesized BDDs (SSBDD) has been developed [3] which allowed to implement ultra-fast fault simulator for combinational circuits [5]. Based on SSBDDs a defect-oriented test generator DOT was developed which is unic with its ability to prove redundancy of physical defects in digital circuits [6].

Recent results of research in the field of reconfigurable logic allowed to create a hardware accelerator to replace traditional software simulators, which allowed to increase the speed of fault simulation in digital circuits about 200

times [7]. Most of our current research is concentrated in the hot problems of testing Network-on-Chips (NoC) [8].

A set of prototype tools, developed as a side-effect of our research, together with dedicated set of tools targeted for e-learning and created in frames of several EU projects, serve now at TUT for teaching design and test, design for testability and fault tolerance. The tools support lecture courses by hands-on training opportunity.

In the following several most important research results obtained in the recent years at TUT are presented. In Section 2 the results in design verification are presented. Section 3 describes simulation speed-up possibilities by hardware emulation, whereas Section 4 presents new software algorithmic possibilities to increase the speed of fault simulation. In Section 5 a novel approach to defect-oriented test generation is presented, and in Section 6 our research on dependability issues is described. Finally, in Section 7 an overview is given about the prototype tool environment developed as a side-effect of our research.

2. HIGH-LEVEL DD-BASED VERIFICATION

With the increase in size and complexity of modern ICs, it has become imperative to address critical verification issues in the design cycle. The process of verifying correctness of designs consumes between 60% and 80% of design effort [9]. For every designer the number of verification engineers may vary from 2 to 4 depending on the design complexity. Moreover, validation is so complex that, even though it consumes most of the computational resources and time, it is still the weakest link in the design process. Ensuring functional correctness is the most difficult part of designing a hardware system [10].

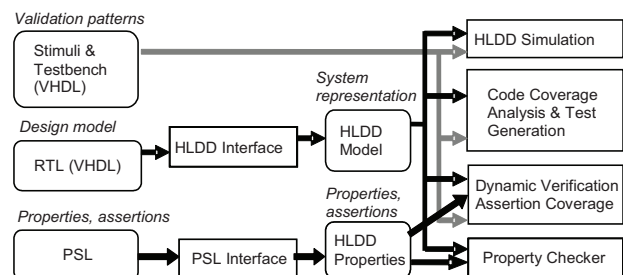


Fig. 1. APRICOT verification flow

We have developed a new framework for digital systems verification, called APRICOT (Assertions, PProperties, COde coverage and Test generation) [11]. It includes different tasks, such as assertion checking, code coverage analysis, simulation, test generation and property checking. APRICOT is easy to set up and use. The novelty of APRICOT lies in a system representation model called High-Level Decision Diagrams (HLDD). The framework has interfaces to commonly used design formats such as VHDL, SystemC, PSL and EDIF. Fig. 1 presents the general structure of the APRICOT framework.

Decision Diagrams have been used in verification for about two decades. Reduced Ordered BDDs [12] as canonical forms of Boolean functions have their application in equivalence checking and in symbolic model checking. Additionally, a higher abstraction level DD representation, called Assignment Decision Diagrams (ADD) [13], have been successfully applied to, both, register-transfer level (RTL) verification and test.

In this paper we consider a different decision diagram representation, High-Level Decision Diagrams (HLDD) that, unlike ADDs can be viewed as a generalization of BDDs. HLDDs can be used for representing different abstraction levels from RTL to behavioral. HLDDs have proven to be an efficient model for simulation and diagnosis since they provide for a fast evaluation by graph traversal and for easy identification of cause-effect relationships [14].

2.1. Code Coverage Analysis

Code coverage provides insight into how thoroughly the code of a design is exercised by a suite of simulations. Code coverage analysis is a well-defined, well-scalable procedure and, thus, applicable to large designs. The main limitation of code coverage metrics lies in the fact that they only measure the quality of the test case in stimulating the implementation and do not necessarily prove its correctness with respect to the specification.

We have shown how classical coverage metrics map to HLDD constructs [15]. Covering all nodes in the HLDD model corresponds to covering all statements in the respective HDL. However, the opposite is not true. We showed that HLDD node coverage is more stringent than HDL statement coverage [15]. This is due to the fact that in HLDDs diagrams are generated to each data variable separately. Such partition on variables includes an additional context to statement coverage. Similar to the statement coverage, branch coverage has also very clear representation in HLDD simulation. The ratio of every edge activated in the HLDD simulation process constitutes to branch coverage.

2.2. Assertion-Based Verification

Assertions have been found to be beneficial for solving a wide range of tasks in systems design ranging from modelling, verification to manufacturing test. In this paper,

we present an approach to checking PSL assertions using HLDDs. Property Specification Language (PSL) is a recently accepted IEEE standard language [16] that is commonly used to express the assertions. Here, the assertions are translated to HLDD graphs and integrated into fast HLDD-based simulation. The structure of HLDD design representation with a temporal extension proposed in [17] allows straightforward and lossless translation of PSL properties.

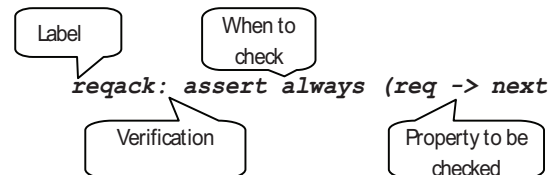


Fig. 2. PSL property reqack

An example PSL property reqack structure is shown in Figure 2. Its possible timing diagram is also illustrated by Figure 3a. It states that ack must become high next after req being high. A system behaviour that activates reqack property however obviously violating it is demonstrated in Figure 3b. Figure 3c shows the case when the property was not activated.

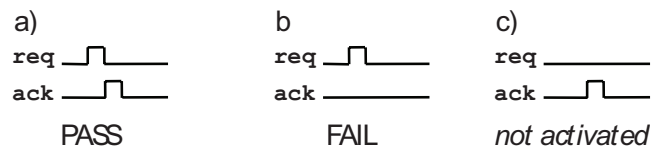


Fig. 3. Timing diagrams for the property reqack

Let us consider an example PSL property P1.

$P1: \text{assert always } !\text{ready and } (a=b) \rightarrow \text{next_e}[1:3]\text{ready}$

Assertion P1 states that whenever 'ready' is low and 'a' is equal to 'b' then during the next three cycles ready must become true. The resulting HLDD graph describing this property is shown in Fig. 4.

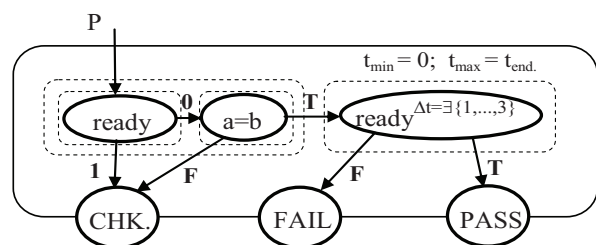


Fig. 4. HLDD for property P1

Note that a HLDD representing assertions has always exactly three terminal nodes labeled by constants:

- FAIL – assertion P has been simulated and does not hold;
- PASS – the assertion has been simulated and holds;
- CHECKING – P has been simulated and it does not fail, nor does it pass non-vacuously.

It has been shown by experiments that HLDDs is an efficient model for performing assertion checking [17].

2.3. Formal Verification

The formal methods to be included to the APRICOT framework include high-level Automated Test Pattern Generation (ATPG) [18] and formal property checking. The latter is under development and will be reduced to using the first one as a model-checking engine.

3. USING EMULATION FOR SIMULATION SPEED-UP

One of the main problems when designing modern on-chip systems is to make sure that already the first version of the chip is “alive”. That is, (1) all essential hardware components are working, and (2) application software and drivers are ready when the chip arrives from the factory (see e.g. [19]). Software simulation is the simplest way to check the functionality of a system and is typically the first choice. Unfortunately, because of its slowness, simulation does not guarantee that the results are available when needed and different approaches are under development to find possibilities for accelerating the simulation process. A possible solution is to use emulation (simulation in hardware) using reconfigurable logic like FPGA-s (see e.g. [20]). To model a hardware module, hardware description language (HDL) based simulation is the choice of most engineers. Taking into account that a model of the system may consist not only of HDL modules at various levels of abstraction but also of modules written in different HDL-s, a complicated simulation environment is required to make sure that the system is working as expected. [21].

Fault simulation is another widely used procedure in the digital circuit design flow. Test generation, fault diagnosis, test set compaction serve as examples of application of fault-free/fault simulators. Accelerating fault simulation would improve all the above-mentioned applications.

The maximum gain in performance could be achieved by moving all the required modules into hard-ware, i.e., emulating the test-bench in hardware. There exist several approaches that confirm the usefulness of replacing simulation with emulation (see, e.g., [19,20]). Difficulties arise when the test-bench is so complex that major modifications are needed for implementing in hardware – test-benches are only models and are not meant to be implemented in hardware. To test the idea of replacing simulation with emulation, an environment was created

with the purpose to evaluate the feasibility of replacing fault simulation with FPGA based emulation [7].

The availability of large FPGA-s doesn’t allow merely implementation of the circuit under test along with fault models but, additionally, to include test vector generation and output response analysis circuits, which in this case correspond to the test-bench, into a single reconfigurable device. Here we relied on a well-known solution for BIST – Linear Feedback Shift Register (LFSR) is used both for input vector generation and output correctness analysis. Automation of emulation environment generation was rather easy because of the modular structure of the hardware part. All commonly used modules are written in VHDL that allows to parameterize design units (see Fig.5). The abstraction level of VHDL modules corresponds to register-transfer level thus allowing the use of basically any FPGA mapping tool.

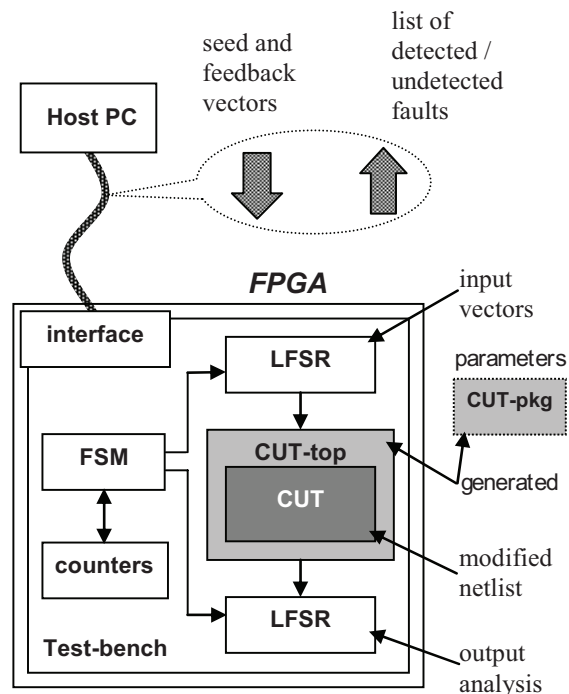


Fig. 5. Fault emulation environment structure

The proposed approach allows simulation speed-up of 40-500 times as compared to the software-based fault simulation [7]. It should be noted that when taking into account also synthesis time, the speed-up is much smaller and therefore the most beneficial is to use scenarios where the number of simulation runs is large, e.g., evaluation of generator/analyzer structures for BIST.

Based on experiences with fault emulation, a more elaborated emulation environment is under development. The need for such an environment is based on the fact that the whole description of a system almost always contains modules described at different abstraction levels. Some of these parts are never meant to be implemented in hardware,

e.g., test-benches and application software. The three following levels can be outlined in the first order:

- Register-transfer level that is synthesizable and therefore directly implementable on FPGA.
- Behavioral (functional) level that is synthesizable by high-level synthesis tools under certain circumstances.
- The rest, essentially software, has lost hardware related issues from its abstraction and is therefore compilable for the used processor (core).

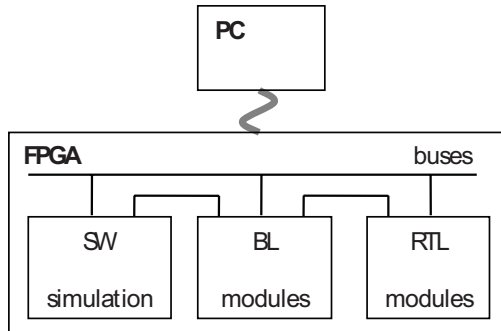


Fig. 6. Multi-module emulation environment

As a result, the needed complexity and performance of the simulator will be reduced. Fig. 6 depicts the structure of such emulation environment, consisting of multiple parallel modules. Additional information about research challenges and potential solutions, especially synchronization related, can be found in [22,23].

4. SYSTEM LEVEL DESIGN OF DEPENDABLE REAL-TIME SYSTEMS-ON-CHIP

The future on chip systems will resemble more computer networks than traditional chips and the Network-on-Chip (NoC) paradigm has been proposed. In addition, new integration methodologies have enabled new 3D architectures, where the dies are stacked into 3-dimensional structures, thus providing even higher densities and complexity.

As technologies advance and semiconductor process dimensions shrink into the nanometer and subnanometer range, a high degree of sensitivity to defects begins to impact overall yield and quality [3]. It becomes very expensive to obtain perfectly operational hardware and the design processes have to be changed.

4.1. Our NoC platform

Our NoC platform is scalable packet switched communication platform for single chip heterogeneous systems. The hard guarantees are provided in Time Division Multiple Access (TDMA) way.

The NoC topology is $m \times n$ (2D) mesh with bi-directional links between the switches. Each switch is connected to 4 switches and to 1 resource. Every resource is connected to

switch via resource network interface (RNI). The NoC platform uses a subset of OSI Reference Model layers: physical, data link, network layer, transport layer and application layer. A resource operates on all 5 layers while switches operate on 4 lower layers. We use wormhole switching with virtual channels and deterministic dimension-ordered (XY) routing.

We concentrate on hard real-time data dominated event triggered NoC systems. However, not all of the traffic must be real-time.

4.2. System Specification & Design

In our approach the application is specified using C code. Based on the input description we extract the Extended Conditional Task Graph (ECTG). In NoC the communication platform introduces communication latency which depends not only on message size but also on resource mapping and needs to be taken into account. Fig. 7.a depicts an example task graph which describes an extract of a GSM decoder.

The ECTG describes the application tasks, their dependencies and task parameters – for example Worst Case Execution Time (WCET), task size etc. Additionally, we need to have the system reliability requirements, how many faults need to be tolerated, and describe the available hardware resources. All the information above and the ECTG itself are captured in XML file. An example of captured information for a part of GSM Decoder can be seen on Fig. 7b.

Once we have the refined task graph, system architecture and dependability description we need to produce a schedule which meets the application deadlines and dependability requirements. During the whole process we take into consideration also possible task mapping. For example – data intensive tasks could be mapped to the same resource or nearby resources to compensate network latency. The exact scheduling algorithms are known to be NP-hard problems. Therefore, different heuristics are used for calculating near-optimal schedules with reasonable time. Fig. 7.c depicts an example of scheduling Figure 1a task graph on multi-processor system.

4.3. Dependability analysis

Reliability improvement techniques have been extensively studied in various systems, either in bus based embedded, macro distributed systems or cover lower layers of NoC. Our objective is to extend those techniques to the system level, to provide design support at early stages of the design flow. The application should be able to tolerate transient or intermittent faults. Permanent faults can be handled by re-scheduling and re-mapping the application on a NoC. During the scheduling process we have from one hand the list of tasks with Worst Case Execution Times and on another hand the dependability requirements.

This is a design area where do not exist any integrated system level design methodology with dependability

requirements. One of the objectives is also to extend the existing system level design tasks into the new design

paradigms, such as NoC-based systems and 3D architectures.

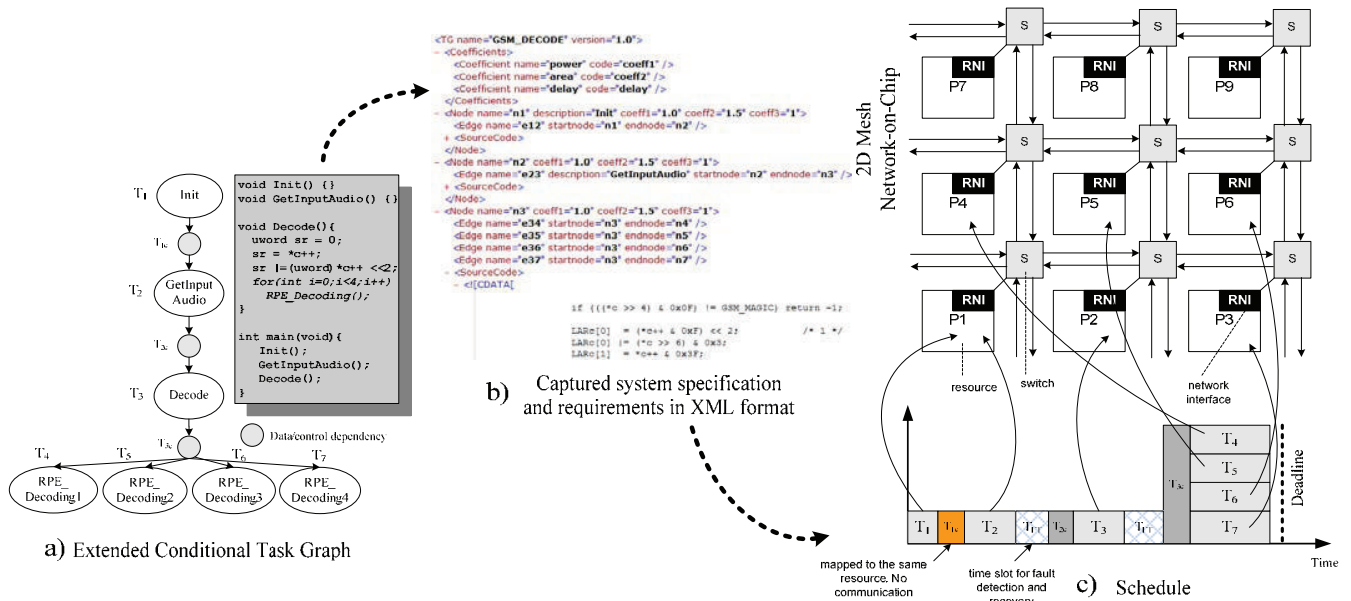


Fig. 7. Scheduling and mapping

5. ULTRA-FAST FAULT SIMULATION

Fault simulation is a well investigated research field, and a lot of methods have been proposed during the last decades, like parallel fault simulation, deductive fault analysis, critical path tracing. The main problem of very powerful critical path method is related to handling of reconvergent fan-outs. It can “process” by a single simulation run all the faults in the circuit, however it works exactly only in the fan-out free circuits. A modified rule based critical path technique that is linear time, exact, and complete was proposed in [24]. However, the rule based strategy does not allow simultaneous parallel analysis of many patterns beyond the fan-out free regions.

In [25] we proposed a new concept of parallel critical path tracing throughout the whole circuit.

Differently from the known critical path tracing approaches, a method is proposed to create ordered topological model for parallel fault backtracing. The model is based on the full Boolean differential, which allows generalization of the parallel critical path fault tracing beyond the reconvergent fan-out stems (see Fig.8). The method is based on the following theorem [25].

Theorem: If a stuck-at fault is detected by a test pattern at the output y of a subcircuit (see in Fig. 8) represented by a Boolean function $y = F(x_1, \dots, x_i, x_j, \dots, x_n)$, then the fault at the fan-out stem x which converges in y at the inputs x_1, \dots, x_i , is also detected iff

$$\frac{\partial y}{\partial x} = y \oplus F((x_1 \oplus \frac{\partial x_1}{\partial x}), \dots, (x_i \oplus \frac{\partial x_i}{\partial x}), x_j, \dots, x_n) = 1 \quad (1)$$

From the formula (1), a method results for generalizing the parallel exact critical path tracing beyond the fan-out free regions. All the calculations in (1) can be carried out in parallel because they are Boolean operations. Further details about the solution for nested reconvergencies can be found in [15].

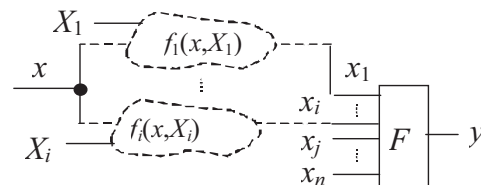


Fig. 8. Reconvergent FFR in a circuit

A topological pre-analysis is carried out to generate an efficient optimized model for backtracing of faults to minimize the repeated calculations because of the reconvergent fan-outs. The algorithm is equivalent to exact critical path tracing, while the backtracing is organized in parallel for groups of test patterns. To achieve high simulation speed, the network of macros rather than gates is used. To make it possible to rise from the lower gate level to the higher macro level, the macros are modeled by structurally synthesized BDDs. A special calculation method was developed to handle the SSBDDs in parallel for groups of test patterns [25].

The proposed exact parallel path tracing fault analysis is carried out in the following sessions:

- topological pre-analysis of the circuit to create a model for fault tracing along the critical paths;
- parallel simulation of a given set of test patterns to calculate the values of all variables of the circuit;
- parallel fault backtracing on the topological model created during the first session.

The topological pre-analysis to create a model for fault backtracing is carried out only once to serve all the next sessions of the procedure. It consists of the following procedures:

- creation of the Reconvergency Graph (RG) of the circuit,
- creation of the whole calculation model of the circuit.

Because of the parallelism, higher abstraction level modeling, and optimization of the topological model, the speed of fault simulation was considerably increased.

Table 1 presents the fault simulation results for the circuits of ISCAS'85 and ISCAS'89 families (column 1) to compare different fault simulators: exact critical path tracing [24] (column 2), two state-of-the-art commercial fault simulators from major CAD vendors (columns 3 and 4), and the proposed new method (column 5). The simulation times were calculated for the sets of random 10000 patterns. The time needed for topology analysis is included and is negligible compared to the gain in speed compared to the previous best method.

TABLE 1. COMPARISON OF TOOLS FOR FAULT SIMULATION

Circuit	Fault simulation time, s			
	[24]	C1	C2	New
c432	70	13.0	3.8	0.64
c499	190	3.0	2.8	0.98
c880	140	26.0	4.0	0.65
c1355	640	44.0	9.0	1.33
c1908	640	53.0	15.6	1.61
c2670	560	104.0	11.0	1.99
c3540	770	191.0	37.4	4.43
c5315	1270	246.0	28.6	3.41
c6288	4280	1159.0	139.2	46.39
c7552	1480	378.0	40.5	5.44
s4863_C	N/A	353.0	30.0	5.10
s5378_C	N/A	170.0	15.9	4.17
s6669_C	N/A	416.0	40.8	7.94
s9234_C	N/A	248.0	26.7	6.72
s13207_C	N/A	332.0	27.2	10.18
s15850_C	N/A	470.0	57.8	13.75
s35932_C	N/A	1751.0	111.6	36.22
s38417_C	N/A	1351.0	157.0	39.05
s38584_C	N/A	1399.0	115.3	34.97
Average speed gain	258.9	41.1	5.8	1

Compared to the commercial tools C1 and C2, the average gain in speed is 41.1 and 5.8 times, respectively. All the experiments were run on a 366 MHz SUN Ultra60 server using SunOS 5.8 operating system except the experiments for the known exact critical path fault simulator where the data are taken from [24]. The experiments in [24] were run on a 2.8 GHz Pentium 4 computer with Windows XP.

6. DEFECT-ORIENTED TEST GENERATION

The logical stuck-at fault (SAF) model has been a long time the prevalent technique to handle formally the real physical defects in electronic systems. In today's systems, however, we have two difficulties when using this model: it is too complex because of the huge number of faults to be handled in systems, and it is inaccurate to represent real physical defects which are taking place in today's nanoelectronic circuits. The paradox is that the two difficulties are working against each other: when trying to represent the defects with less complex and higher level fault models the accuracy will even decrease, and vice versa, when trying to increase the accuracy of defect modeling, the complexity of the fault model will increase. To get out from the deadlock, the two opposite trends – high-level modeling and defect-orientation – should be combined into hierarchical approach.

Another problem is that the know-how about defects is quickly getting obsolete. New semiconductor processes will introduce new failure mechanisms, defects, and fault effects. This makes defect-based testing difficult, and all the needed changes in defect modeling should be taken into account and introduced continuously into the database of test generation and fault simulation tools.

We have developed a new approach for hierarchical defect simulation based on defect preanalysis for the components included into the libraries, and using the results of preanalysis in higher level fault modeling. The cornerstone of the new approach is - the functional fault model as a method for mapping faults from one hierarchical level to another. Based on this approach, a hierarchical algorithm for defect-oriented deterministic test generation was developed and implemented [6].

A methodology was developed which allows to find the types of defects that may occur in a real circuit, to determine their probabilities of occurrence, and to find the input test patterns (logical constraints) that allow to activate and detect these defects. This set of constraints which allows to detect all defects in a given component is called functional fault model of the component.

According to this model, each library component is represented by a set of logical constraints needed for activating the defects in the component. Simulations for finding the constraints are carried out on the layout level of components. The set of logical constraints can be regarded as a method for mapping physical defects to the logic level.

During higher (logic) level test generation and fault simulation the physical defects are modeled only by logical constraints without referring back to the layout details.

The proposed functional fault model allows to represent and handle arbitrary physical defects not only in the library components, but also the physical defects in the communication network of components by the same technique.

There is a class of physical defects which increase the number of states in the circuit. To activate these defects, sequences of patterns (sequential constraints) are needed. Simulation based technique to find sequential constraints is not the best solution. For this purpose analytical approach was developed. A physical defect in the component is modeled as a defect variable in a generic Boolean differential equation which includes both the correct and faulty behavior of the component. Solutions of these

equations give the logical constraints (or sequential constraints) for activating defects locally. In such a way, for example, the bridging faults that cause a feedback and transform a combinational circuit into a sequential one, can be modeled for test generation purposes.

A defect-oriented deterministic test generation tool (DOT) was developed [6]. The experimental data obtained by the tool for ISCAS'85 benchmark circuits are presented in Table 2. It was shown that 100% stuck-at fault tests covered only about 75-82% physical defects (column 5 in Table 2). The main feature of the new tool is its ability to reach 100% defect testing efficiency (percentage of covering the non-redundant defects) for the given set of defects by proving the redundancy of not detected defects. The tool allows to prove the redundancy of physical defects in relation to the logic behavior of a circuit.

TABLE 2. EXPERIMENTAL DATA OF DEFECT-ORIENTED TEST GENERATION

Circuit	Number of defects			Defect coverage			
	All defects	Redundant defects		100% stuck-at fault ATPG			DOT
		Gates	System				
1	2	3	4	5	6	7	8
c432	1519	226	0	78,6	99,05	99,05	100,00
c880	3380	499	5	75,0	99,50	99,66	100,00
c2670	6090	703	61	79,1	98,29	98,29	100,00
c3540	7660	985	74	80,1	98,52	99,76	99,97
c5315	14794	1546	260	82,4	97,73	99,93	100,00
c6288	24433	4005	41	77,0	99,81	100,00	100,00

Column 6 in the Table 2 shows the defect testing efficiency after proving the redundancy of defects inside the library cells, and column 7 shows the defect testing efficiency after proving the redundancy for the whole set of defects. The column 8 shows the defect testing efficiency reached by the test generation tool DOT.

7. DESIGN AND TEST RESEARCH ENVIRONMENT

The experimental tools developed as a side effect of the research carried out at TUT during the recent 5-6 years are organized as an experimental R&D environment for investigating a broad set of design and test problems (Fig. 9). The environment consists of the following parts:

- Synthesis tools (high-level and logic level synthesis).
- Test generation and fault simulation tools (hierarchical, logic and defect level test sequence generators).
- Converters (interfaces between tools).
- Other (university) tools linked to the environment.

Design information can be created in different ways: (1) VHDL files to be processed by commercial or experimental high-level or logic synthesis systems, (2) manually by schematic editors. The gate-level design is presented in the

EDIF format. In university research practice, ISCAS benchmark families which have their own presentation format (ISCAS format) are widely used. In order to link test generation and fault simulation tools with all the needed formats, different converters are developed. EDIF netlists can be converted into ISCAS'85 or ISCAS'89 formats. Necessary technology library files to support such conversion have been created for the research environment.

The Turbo-Tester tools are based on SSBDDs, they have EDIF-SSBDD converters to link the tools with commercial CAD systems. Hierarchical ATPG DECIDER uses two inputs – higher level (RTL) descriptions in VHDL and low gate-level descriptions in EDIF. For importing VHDL descriptions to DECIDER which uses high-level DDs as input, a converter VHDL-DD is available.

As a set of examples, the following design flows can be exercised in this environment.

- Design and hierarchical ATPG. RTL VHDL design is synthesized by high-level synthesis tool. A logic level synthesis for the high-level blocks follows. For these designs DD and SSBDD models are generated. Using DDs and SSBDDs, hierarchical ATPG DECIDER generates test sequences.

- Logic level ATPG. Using SSBDDs, Turbo Tester ATPG generates logic level test patterns targeted to detect logic level stuck-at faults.
- Defect-oriented ATPG. Using SSBDDs and the defect library, the defect-oriented test generator DOT generates test patterns targeted to defect

physical defects. The defect libraries available are created in cooperation with Warsaw University of Technology.

- University tools that traditionally use ISCAS benchmarks can be linked via EDIF-ISCAS converter to commercial design tools.

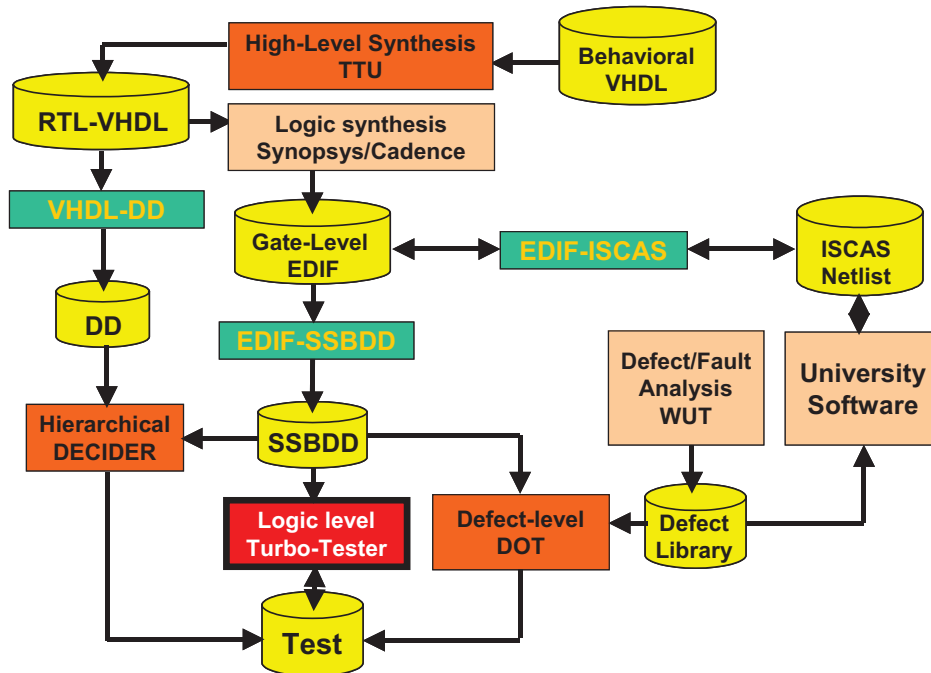


Fig. 9. Hierarchical design and test research environment

Turbo Tester tool set represents an independent logic level test research environment. It consists of a set of tools for solving different test related tasks by different methods and algorithms:

- Test pattern generation by deterministic, random and genetic algorithms.
- Test optimization (test compaction).
- Fault simulation and fault grading for combinational and sequential circuits by different algorithms.
- Defect-oriented fault simulation and test generation.
- Multi-valued simulation for detecting hazards and analyzing dynamic behavior of circuits.
- BIST analysis and quality evaluation for different BIST architectures.

All the Turbo Tester tools operate on the model of SSBDD. The tools run on the structural level whereas two possibilities are available – gate-level and macro-level modeling. In the latter case, the gate network is transformed into macro network where each macro represents a tree-like sub-network. Using the macro-level helps to reduce the complexity of the model and to improve the performance of tools. The fault model used in the Turbo Tester is the traditional stuck-at one. However, the fault simulator and test generator can be run also in the defect-oriented mode, where defects in the library

components can be taken into account. In this case, additional input information is needed about defects in the form of defect tables for the library components.

A selection of the prototype tools described above together with a set of separate tools (Java applets) developed specially for teaching purposes are integrated into e-learning environment to support university courses by providing opportunity for the students for hands-on training [26]. This environment consists of toolsets: (1) Turbo Tester - CAD Software for Digital Test, (2) xTractor - CAD Software for High-Level Synthesis, (3) DefSim – HW/SW environment for experimental study of CMOS defects, (4) BIST Analyzer - a training system for learning self-testing issues of modern multi-core electronic systems, (5) Trainer 1149 - a multi-functional SW system, which provides a simulation, demonstration, and CAD environment for learning, research, and development related to IEEE 1149.1 Boundary Scan (BS) standard, (6) Applets for training and teaching logic synthesis and test at gate- and register transfer levels, (7) Applets for FSM Decomposition and Synthesis, (8) Deterministic traffic generator for NoC simulator, and (9) Test Time Calculator (Simple NoC simulator, based on XY-routing).

The laboratory tasks developed for this environment represent simultaneously real research problems, which

allow to foster in students critical thinking, problem solving skills and creativity in a real research environment and atmosphere.

CONCLUSIONS

An overview was given about the recent research results at TUT in the field of design and test of dependable embedded systems. These results have been obtained thanks to the broad international cooperation during the last decade in frame of several EU projects like SYTIC, VILAB, REASON, eVIKINGS II, VERTIGO [27]. As a result of these projects, two new competence centres were established – Estonian Research Centre for Dependable Computing and Estonian Development Centre of Mission Critical Embedded Systems (ELIKO). ELIKO contracts between 7 private SMEs in Estonia under the leadership of TUT. Both centres are working on transfer of technology to local industry. Through ELIKO very tight links have been established now between the Academia and the industry of Estonia.

As a side-effect of the research carried out during recent years, an experimental research environment has been developed to support in the future both, research and teaching. The originality of the environment is in multi-functionality of the system (important for research and training), low-cost and ease of use. The multi-functionality means that different abstraction level models can be easily synthesized (to analyze the influence of the complexity of the model to the efficiency of methods); different methods of the same task are implemented (to analyze the efficiency of different approaches), the fault models can be easily changed and updated (to analyze the adequacy and accuracy of testing). The multi-functionality allows to set up and modify easily different experimental schemes and scenarios for investigating new ideas and methods. This multi-functionality gives an excellent opportunity for students working in this environment to understand the ideas, advantages and drawbacks of different methods at changeable conditions. In traditional commercial design tools these purely research oriented possibilities are missing.

ACKNOWLEDGMENT

The work has been supported by Estonian Science Foundation grants 5910, 6717, 6829, 7068, EC 6th FP IST project VERTIGO, Estonian IT Foundation (EITSA) and Enterprise Estonia.

REFERENCES

- [1] R.Klein, T.Piekarz. Accelerating Functional Simulation for Processor Based Designs. Mentor Graphics Corporation. White paper, 2005.
- [2] K.Roy, T.M.Mak, K.-T.T.Cheng. Test consideration for nanometer-scale CMOS circuits. *IEEE Design and Test of Computers*, vol.23, no 2, pp.128-136, 2006.
- [3] R.Ubar. Test Synthesis with Alternative Graphs. *IEEE Design and Test of Computers*. Spring, 1996, pp.48-59.
- [4] J.Raik, R.Ubar. Fast Test Pattern Generation for Sequential Circuits Using Decision Diagram Representations. *JETTA*. Kluwer Acad Publishers, Vol. 16, No. 3, pp. 213-226, 2000.
- [5] R.Ubar, S.Devadze, J.Raik, A.Jutman. Fast Fault Simulation in Digital Circuits with Scan Path. 13th Asia and South Pacific Design Automation Conference – ASP-DAC 2008, Seoul, Korea, Jan. 21-24, 2008, pp. 667-672.
- [6] J.Raik, R.Ubar, J.Sudbrock, W.Kuzmicz, W.Pleskacz. DOT: New Deterministic Defect-Oriented ATPG Tool. *Proc. of 10th IEEE European Test Symposium*, May 22-25, 2005, Tallinn, pp.96-101.
- [7] P.Ellervee, J.Raik, R.Ubar, K.Tammemäe. FPGA-Based Fault Emulation of Synchronous Sequential Circuits. *IEE Proc. on Computers & Digital Techniques*. Vol.1, Issue 2, pp.70-76, March 2007.
- [8] J.Raik, R.Ubar, V.Govind. Test Configurations for Diagnosing Faulty Links in NoC Switches. 12th IEEE ETS 2007, Freiburg, Germany, May 20-24, 2007, pp.29-34.
- [9] International Technology Roadmap for Semiconductors 2006 report, [URL] www.itrs.net, 2006.
- [10] S.Tasiran, K.Keutzer, Coverage metrics for functional validation of hardware designs. *Design&Test of Computers*, IEEE, Vol 18, Issue 4, Jul-Aug. 2001, Pages 36-45.
- [11] URL: <http://www.vertigo-project.eu>
- [12] R.Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Comp.*C-35, 8:677-691, 1986
- [13] V.Chayakul, D.D.Gajski, L.Ramachandran, “High-Level Transformations for Minimizing Syntactic Variances”, *Proc. of ACM/IEEE DAC*, pp. 413-418, June 1993.
- [14] R.Ubar, J.Raik, A.Morawiec, Back-tracing and Event-driven Techniques in High-level Simulation with Decision Diagrams. *ISCAS 2000*, Vol. 1, pp. 208-211.
- [15] K.Minakova, U.Reinsalu, A.Chepurov, J.Raik, M.Jenihhin, R.Ubar, P.Ellervee. High-Level DD Manipulations for Code Coverage Analysis, *Baltic Electronics Conf.*, IEEE, 2008.
- [16] IEEE-Commission, “IEEE Standard for Property Specification Language (PSL),” 2005, IEEE Std 1850-2005.
- [17] M.Jenihhin, et al. Temporally Extended High-Level Decision Diagrams for PSL Assertions Simulation. *Proc. of the 13th IEEE European Test Symposium*, 2008.
- [18] J.Raik, R.Ubar, T.Viilukas, M.Jenihhin. Mixed Hierarchical-Functional Fault Models for Targeting Sequential Cores. *Elsevier Journal of Systems Architecture*.
- [19] A.Bigot et al. Deploying Hardware Platforms for SoC Validation: An Industrial Case Study. *The International Conference on Field Programmable Logic and Applications (FPL’04)*, Antwerp, Belgium, pp. 64-73, Aug. 2004.
- [20] N.Genko et al. A Complete Network-On-Chip Emulation Framework. *Design Automation & Test in Europe (DATE’05)*, Munich, Germany, pp. 246-251, March 2005.
- [21] K.Morris. Debug Dilemma. Simulate or Emulate? *FPGA and Structured ASIC J.*, <http://fpgajournal.com>, Jan. 2005.
- [22] P.Ellervee, A.Arhipov, K.Tammemäe. Clock Manipul. for Heterogenous Emulation Environment. *The 24th NORCHIP Conference*, Linköping, Sweden, pp. 213-216, Nov. 2006.
- [23] P.Ellervee, U.Reinsalu, A.Arhipov. Translating Behavioral VHDL for Emulation. *The 25th NORCHIP Conference*, Aalborg, Denmark, Nov. 2007.
- [24] L.Wu, D.M.H.Walker. A Fast Algorithm for Critical Path Tracing in VLSI. *Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, Oct. 2005, pp.178-186.
- [25] R.Ubar, S.Devadze, J.Raik, A.Jutman. Ultra Fast Parallel Fault Analysis on Structural BDDs. 12th IEEE ETS, Freiburg, Germany, May 20-24, 2007, pp.131-136.
- [26] <http://ati.ttu.ee/projects/tools.html>
- [27] <http://ati.ttu.ee/index.php?page=800>