Web Based Tools for Synthesis and Testing of Digital Devices

Sergei Devadze, Artur Jutman, Margus Kruus, Alexander Sudnitson, and Raimund Ubar

Abstract: *This paper describes a design system for distance-learning based on using of so called "living pictures". While describing the approach, it addresses relevant questions about the design and test of control intensive digital systems. The system is implemented in a form of Java applets and can be freely accessed over Internet. The Java applets has built-in multilingual support to ensure easy integration into teaching courses of universities over the world.*

Key words: *Distance-learning, Applets, Automata Decomposition, Register Transfer Level, Test*

INTRODUCTION

Entering the modern concepts of digital system design means teaching more material on high-level system design together with the same amount of essential basics, that must not be forgotten. In its turn, this means that more information volumes must be fitted into the same time frame. At the same time, teaching the basics of digital design and test means teaching a lot of complex connections. At first, those connected topics usually explained one by one. Then their dynamic interactions come [1, 2].

After the lecture this dynamic part is lost and the students only have a static part of the whole scenario in their notes. After the lecture they can only consult these notes trying to solve some problems by using a newly learned method as good as they remember. Other accompanying materials students use in most cases are books, scripts etc. which is definitely insufficient to restore the dynamic part of the lecture.

In the following, a conception and tools are presented to increase the teaching quality in the field of electronics design and test. To illustrate both design and test problems together, we use the same system that supports the possibility of distance learning as well as a web-based computer-aided teaching. The system is designed mainly to illustrate Register Transfer Level (RTL) problems in control intensive digital systems. Such topics as investigation of tradeoffs between speed and hardware cost in digital design, decomposition, register transfer level simulation, fault simulation, test generation, built-in self-test and other similar are covered by the training system. The system consists of several interactive modules focused on easy action and reaction, learning by doing, game-like use to encourage students for critical thinking, problem solving, and creativity.

RTL DESIGN FRAMEWORK

Typically, the synthesized design from high level steps of synthesis (scheduling and allocation) consists of two modules: a control part (or controller) and an operative part (or datapath). The formal description of control unit is a finite state machine which generates control signals to activate the different operations in specific clock cycles. The datapath unit consists of instantiation of datapath components such as multipliers, adders, incrementers and multiplexors [3].

The controller implements the FSM. It computes the next state and the signals controlling the transfers in datapath according to primary control input lines, status lines and the present state. The extracted FSM is described as unit with the set of binary inputs (channels and the set of binary outputs (channels).

The teaching system is implemented as a Java applet, which consists of the following parts:

- System model subpanel
- Data path
- Control path
- Simulation module

- Fault simulation module
- Test generation module
- BIST module

The conception we describe in the following allows to solve and illustrate many problems related to RT-level control intensive digital design together with test. The range of problems includes (but is not limited to):

- Design of data path and a microprogram (control path) on RT level
- Investigation of tradeoffs between speed & HW cost in the system
- RT level simulation
- Fault simulation
- Test generation
- Design for testability and BIST

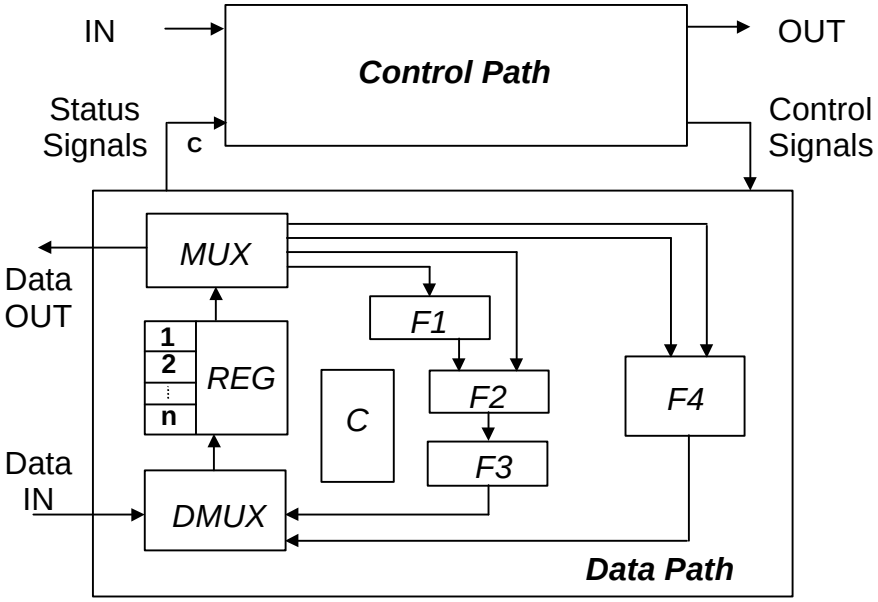


Fig. 1. System model

Each functional unit *F1..F4*, *MUX*, and *DMUX* has a list of micro-operations unary or binary. It is supported by an RT-level and gate-level models of these microoperations. The RT-level model is needed for high-level simulation by JAVA library subprograms, while the gate-level model is used for gate-level logic and fault simulation. All the microoperations are labeled by a control signal which activates the microoperation.

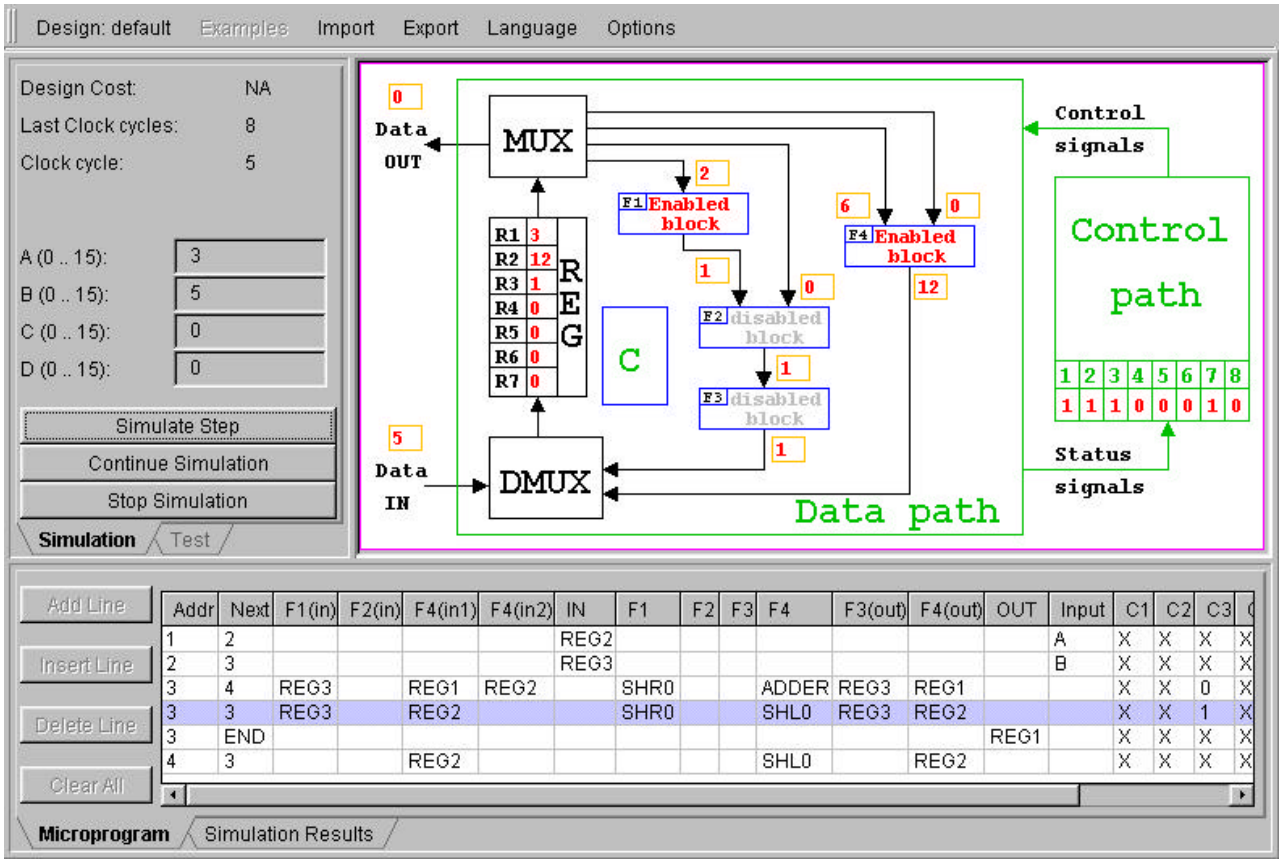
While designing own device (implementing a given algorithm or a function like multiplication, division etc.), a student can select needed microoperations for each unit of data path from the whole set of possible pre-designed microoperations. Each microoperation has a gate-level implementation, and the number of gates determines the cost of the microoperation. By selecting a set of microoperations for the whole data path the student will get also the cost of the data path in the number of gates.

Different architectures can be chosen for implementation of a given function. Students can compare them and find the tradeoffs:

- *M-automaton* (*F1* and *F3* are selected to be transparent, *F4* is disabled, all microoperations are selected only from *F2*). In this case it is possible to carry out a single microoperation in one clock cycle, the speed is low, but the cost of hardware is saved.
- *Sequential IM-automaton* (*F1*, *F2*, *F3* are enabled, *F4* is disabled). In this case the system can carry out maximum 3 microoperations during a single clock cycle if the

algorithm gives such a possibility. For example, in division *F1* is used for inversion to allow subtraction by the adder in *F2*, and *F3* can be used for shifting of data.

- *Parallel IM-automaton* (all blocks are selected). In this case the system can carry out maximum 4 microoperations during a single clock cycle if the algorithm allows to. For instance, while the three functional units described above perform division, F4 can count clock cycles.



Addr	Next	F1(in)	F2(in)	F4(in1)	F4(in2)	IN	F1	F2	F3	F4	F3(out)	F4(out)	OUT	Input	C1	C2	C3
1	2					REG2								A	X	X	X
2	3					REG3								B	X	X	X
3	4	REG3		REG1	REG2		SHR0			ADDER	REG3	REG1			X	X	0
3	3	REG3		REG2			SHR0			SHL0	REG3	REG2			X	X	1
3	END												REG1		X	X	X
4	3			REG2						SHL0		REG2			X	X	X

Fig. 2. RTL design applet window

For every chosen architecture, the system calculates the cost of HW. The speed (the number of clock cycles the microprogram needs) can be measured by simulation.

THE IMPLEMENTATION OF RTL DESIGN SYSTEM

The RTL design teaching system is implemented in a form of Java applet. It can be accessed through URL [4]. The Java 1.1 is chosen as the platform for the applet because at the moment it is a de-facto standard and it is included in installation packages of most popular browsers.

Applet interface (Fig. 2) consists of following parts:

- *Schematic View* - panel which contain schematic representation of a design. This panel allows user to define or change some properties of data path of the design. Some components can be enabled/disabled or their functionality can be changed. In the step-by-step simulation mode the results of the simulation are visually demonstrated on the *Schematic View* after each step of the simulation.
- *Microprogram tab-panel* is used to define control path of the system. In the simulation mode this panel shows which part of the microprogram is currently executed.
- *Simulation tab-panel* and *Test tab-panel* are used to simulate and test design correspondingly.

The simulation can be carried out in two ways:

- *Step-by-step simulation mode*. In this case each row of the microprogram is executed separately and all the results of this execution (including state of registers and functional blocks, inputs and outputs, status signals, etc) can be viewed directly on *Schematics View* sub-panel. This mode of simulation is useful for illustrating how the design works or for debugging.
- *Test mode*. This mode is used to test the design repeatedly with some set of input data. User fills in the *Test data table* in the *Test tab-panel* with the data that will be used in testing. Then the design simulation can be executed for a particular row of test table or for all the rows at once.

Applet has a built-in collection of examples (they also can be extended or modified) that implement different algorithms. They will help users to understand principles of functioning of the system. For connecting the system to other applications as well as providing users with possibility to save the results of their work for further use applet has a capability of exporting results and importing source data.

The steps to create own design are as folllows:

Step 1 - defining data path. By clicking on highlighted blocks of the *Schematic View* some properties of the design data path part can be defined or changed. For example clicking on functional blocks allows to enable/disable them or change their functionality. In a same way status signals signal from data path to control path (for realizing conditions) can be defined.

Step 2 - defining control path. Control path can be defined using *Microprogram tab-panel*. Buttons *Add Line*, *Insert Line*, *Delete Line* and *Clear All* (on the right side of the *Microprogram tab-panel*) allow to manipulate with microprogram rows. All values in a some row of a microprogram (including *Address* and *Next address*, *control signals* and *conditions*) can be changed.

Column *Input* allows to specify an value that will be set on *Data In* channel when some address is reached during microprogram execution. This value can be defined in a hard way by specifying some concrete value or by specifying one of indexes to allow to substitute different values for design testing.

The applet has a flexible design. Should any other RT-level model be used instead of the described one, it must be only specified in a form of text-files. Then it can be loaded just as simple as the original one. The Microprogram table and the Simulation Results table will be automatically reconfigured as well.

Applet has a built-in multilingual support: its interface can be easily translated to other languages if needed (current language is English).

APPLETS FOR CONTROLLER DECOMPOSITION

This part of work focuses on particular but comprehensive problem of decomposition of finite automata (finite state machine). Theoretical background of our system is the automata decomposition theory, which uses partition pair algebra proposed in [6]. The importance of this theory lies in the fact that it provides a direct link between algebraic relationships and physical realizations of finite state machines. The mathematical foundation of this theory rest on an algebraization of the concept of "information" in a machine and supply the algebraic formalism necessary to study problems about the flow of this information in machines as they operate. It falls squarely in the interdisciplinary area of applied algebra, which is a part of engineering mathematics. We are concerned with solving complex combinatorial tasks arising from the process of design.

In this part of the distance-learning tool set, different applets for studying the basics of the decomposition theory of automata have been developed [4]. The applet on construction of an automata network allows experimenting with decomposition of automata. Different partitions can be chosen to decompose the given sequential machine to meet different design restrictions.

The developed set of applets can be used for teaching the basics of automata theory. The teacher can use the applet during the lecture explaining the basics of the topic. On the other hand, the applet can be used during the exam for giving some tasks to students. Students can use the same applet for training purposes.

APPLET FOR LEARNING LOGIC LEVEL TEST

This applet was developed to study the basic problems in the field of logic level testing, especially, in fault simulation, test generation, and fault location [7].

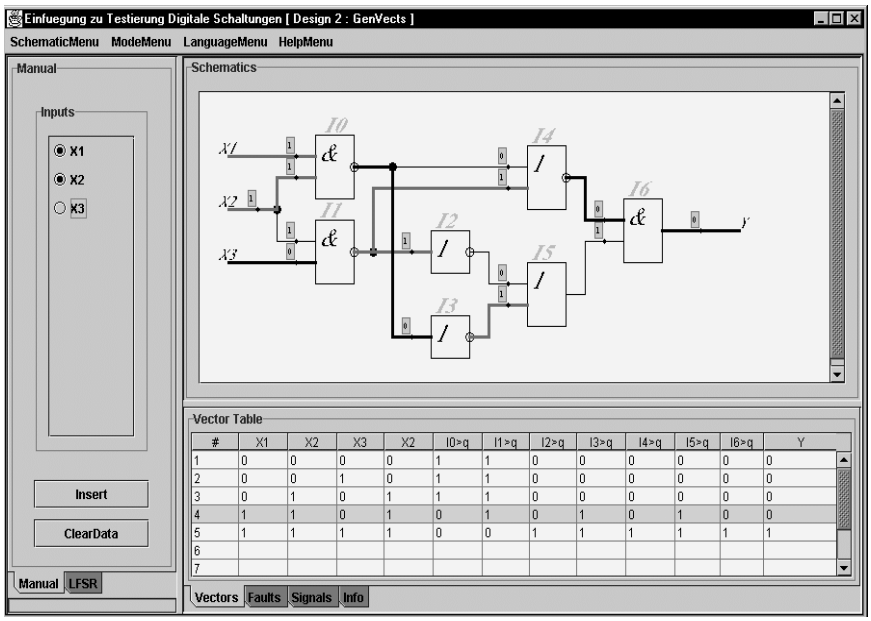


Fig. 3. "Living picture" as a Java applet for testing

From the list of predefined circuits by *schematics menu*, a simple circuit under test (CUT) can be chosen. The *mode menu* tells the applet what is to be done – automated or manual test vector insertion, manual test vector generation, fault simulation or fault diagnosis (two possible modes: sequential and combinational diagnosis).

In *test generation mode* we choose a target fault in the circuit, and create manually step by step proper activated paths in the circuit with the goal to find a pattern that tests the fault. We have to activate the fault at its site, and to propagate the error signal to the output by clicking the needed signal values on the lines in the CUT. The colours on lines help to understand the current status of the task: activated faults and activated paths are marked by red and green lines, the inconsistencies of the signal values are highlighted by blue colour. The faults detected by the generated test pattern are displayed in the form of fault table.

In fault simulation mode, a fault table is generated and shown for all the test patterns generated by the given moment.

In fault diagnosis mode we need first, to create a fault table by running the fault simulator for a set of previously generated test patterns. Entering into the diagnosis mode will insert a random fault into the circuit. The following diagnosis strategies chosen from menu can be investigated: combinational and sequential diagnosis. For learning the combinational diagnostic strategy, a single vector or a subset of vectors can be selected and applied to the erroneous circuit (by imitating test experiments). The applet shows the results of testing, and displays also the subset of suspected faults. To improve the diagnostic resolution, additional test vector(s) may be generated and used in the repeated test experiment. Sequential diagnosis is based on the guided probing strategy. A test

pattern is applied and the expected behavior of the circuit is displayed. By clicking on the connection boxes the real values of signals of the faulty circuit can be measured.

CONCLUSIONS AND FUTURE WORK

The pedagogical basis on which our system is built is that students learn most quickly when they are presented with material that they can quickly use to solve design problems. By the use of web-based media we achieve: presentation of course material independent of place and time, individual learning according to the students' own needs, new forms of communication between teachers and students etc. The conception presented allows to improve the skills of students to be educated for electronics design and test related topics. The principal mission of the conception is to inspire students to learn, and to prepare them for developing problem-solving strategies.

In the future we plan to update the system to carry on an interactive session on Computer Aided Design tools across the Web using nothing but a Web Browser.

ACKNOWLEDGEMENT

This work is partially supported by the Ministry of Education in Thüringen, Germany (DILDIS project), by EU V Framework project REASON, and by the Estonian Science Foundation (grants No 3658, 4876 and 4300).

REFERENCES

[1] Wuttke, H.-D., Henke, K., R.Peukert. Internet Based Education - An Experimental Environment for Various Educational Purposes. Proc. of the IASTED Int. Conf. on Computers and Advanced Technology in Education, Philadelphia, PA USA. IASTED/Acta Press No. 292, May 6-8, 1999, pp. 50-54.

[2] Ubar, R., H.-D.Wuttke. Action Based Learning System For Teaching Digital Electronics And Test. 3rd European Workshop on Microelectronics Education" (EWME 2000), Aix-en-Provence, Kluwer Academic Publishers, May 18-19, 2000, pp. 107-110.

[3] Armstrong, J.R., F.G.Gray. Structured logic design with VHDL. Prentice-Hall, Engl. Cliffs, 1993.

[4] Available: http://www.pld.ttu.ee/dildis/automata/applets/

[5] M.Abramovici et al. Digital Systems Testing and Testable Design. IEEE Press, 1999.

[6] Hartmanis, J., R.E.Stearns. Algebraic Structure Theory of Sequential Machines. Prentice-Hall, Engl. Cliffs, 1966.

[7] Available: http://www.pld.ttu.ee/tt/

ABOUT THE AUTHORS

Student Sergei Devadze,
Ph.D. Student Artur Jutman,
Assoc. Prof. Margus Kruus, Ph.D.,
Assoc. Prof. Alexander Sudnitson, Ph.D.,
Prof. Raimund Ubar, Dr. Sc.,
Department of Computer Engineering, Tallinn Technical University, Raja 15, 12617 Tallinn, Estonia, Phone: +372 620 2251, E-mail: alsu@cc.ttu.ee