Java Technology Based Training System for Teaching Digital Design and Test

S. Devadze, A. Jutman, A. Sudnitson, R. Ubar

Tallinn Technical University,
Raja 15, 12618 Tallinn, Estonia
d990888@ttu.ee | artur@pld.ttu.ee | alsu@cc.ttu.ee | raiub@pld.ttu.ee

H.-D.Wuttke

Ilmenau Technical University
Ilmenau, Germany
Dieter.Wuttke@theoinf.tu-ilmenau.de

ABSTRACT: A conception of training system for teaching design and test of digital devices is presented. The system is designed mainly to illustrate RT-level (Register Transfer Level) problems in control intensive digital systems including: investigation of tradeoffs between system's speed and HW cost, RT-level simulation, fault simulation, test generation, built-in self-test (BIST) and others. The system is implemented in a form of Java applet and can be freely accessed over Internet. The latter makes it easy for students even from foreign universities to use this system any time and in any place. The Java applet has built-in multilingual support to ensure easy integration into teaching courses of universities over the world.

1 Introduction

Rapid advances of electronic technologies and design automation enabled engineers to design larger, more complex, integrated circuits. At the same time the problems of test and design for testability become more important, as costs of verification and testing are getting the major component of design and manufacturing. Today, design and test are no longer separate issues. The companies frequently hire test experts to advise their designers on test problems, and they even pay a higher salary to the test experts than to their VLSI designers. This reflects today's university education: everyone learns about design, but only truly dedicated students learn test [1-3].

Entering the SoC (System-on-Chip) era with its new concepts 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.

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 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.

2 System Overview

The core of the teaching concept presented here is a Java-applet of a special type, which we call "Living Pictures" [4]. Those applets simulate tricky, quite complicated situations of the learning subject in a graphical form on the computer screen. The graphic is self-explanatory and provides interaction possibilities. By using these possibilities the students can generate examples that are interesting enough to encourage their own experiments but not too complicated for learning.

The following figure shows the four phases of the

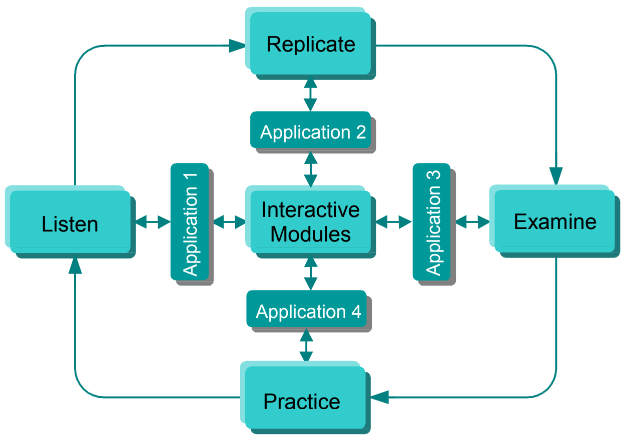
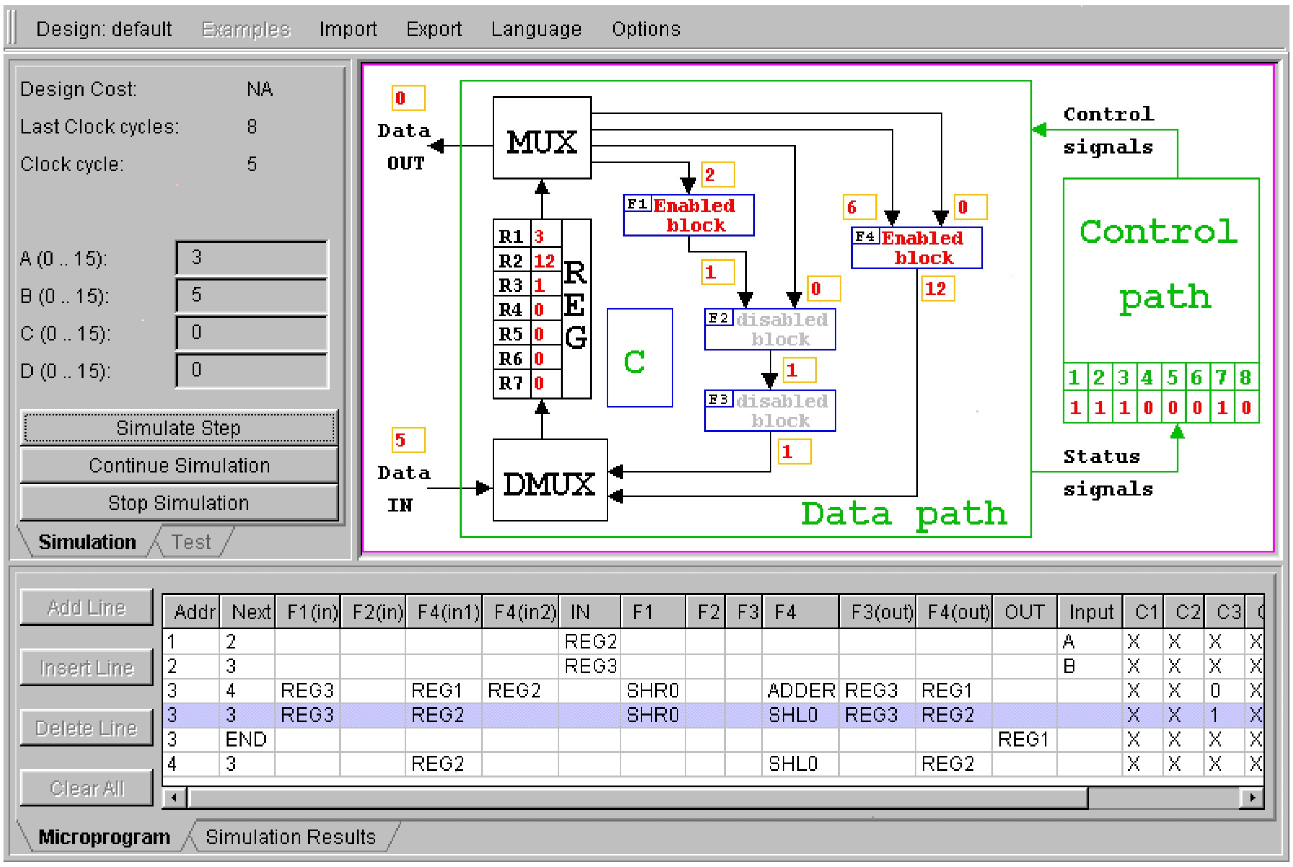


Fig. 1 The four phases of learning process



Design Cost: NA
Last Clock cycles: 8
Clock cycle: 5

A (0 .. 15): 3
B (0 .. 15): 5
C (0 .. 15): 0
D (0 .. 15): 0

R1	R2	R3	R4	R5	R6	R7
3	12	1	0	0	0	0

1	2	3	4	5	6	7	8
1	1	1	0	0	0	1	0

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

Fig. 2 Training system

learning process supported by the education system: listening, replication, examination and practice phase. The system supports the action based training since for each phase there exists a special application service adapted to the learning process which allows different views and actions using the same interactive module.

In our teaching system [5] we succeeded to combine and illustrate many different problems related to both RT-level control intensive digital design and test. This gives a unique possibility to teach all of them in a consecutive iterative approach. The range of problems includes:

- Design of a data path and control path
- Investigation of tradeoffs between speed & HW cost
- RT level simulation
- Fault simulation
- Test generation
- Design for testability and BIST

The system (Fig. 2) consists of following parts:

- *Schematic View* panel provides the schematic representation of the design and the graphical simulation data. The structure of the *data path* is reflected there
- *Microprogram table* is used to define the *control path* of the system. During the simulation this panel shows which part of the microprogram is currently executed.
- *Simulation* and *Test tab-panels*. The simulation can be carried out in two different modes:
  – In the *step-by-step mode* or each row of the microprogram is executed separately and results are constantly updated in schematic view panel. This mode is useful for illustration of the design work and for debugging.
  – In the *test mode* one is testing the design repeatedly with some set of input data.
- *Simulation Results* tab-panel is the place where the results of simulation or test are stored.
- *Fault simulation* module provides fault simulation for the data path and its units.
- *BIST module* provides the basis to experiment with embedded self-test facilities.

The applet has a flexible design. The RT-level system model, shown in Fig 2 is not mandatory. Should any other model be used, 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 extendable collection of examples implementing different algorithms. They help users to understand principles the system operation. For connecting the system to other applications as well as for providing users with a possibility to save the results of their work for further use applet has a data import/export capability. It also has a built-in multilingual support.

3 Teaching RT-Level Design

The following modules allow to introduce basics of RT-level design to students.

Data Path. 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 two models are needed for high-level simulation and gate-level logic and fault simulation respectively. All the microoperations are labeled by a control signal which activates the microoperation. The description of the data path functionality in format "unit control signal: microoperation" is presented in Fig. 3. There is an overlay between functions of F4 and of F1, F2 and F3 to allow a parallelization of the given algorithm

MUX	m_{ij}: $B_j = R_i$	i = 1,…,n – Register number; j = 0,1,2,4 – Bus number where B_0 is Data OUT bus, B_1 is the Bus to F1 etc.
DMUX	d_{ij}: $R_j = B_i$	i = 0,3,4 – Bus number where B_0 is Data IN bus, B_3 is the Bus from F3 etc. j = 1,…,n – Register number
F1 (F3)	f_{1j} (f_{3j})	unary microoperations like: various shiftings, inverting, counting (+1, -1) etc.
F2	f_{2j}	various binary microoperations (with 2 operands)
F4	f_{4j}	various unary and binary microoperations (with 2 operands)
C	c_i	a list of Boolean conditions

Fig. 3 Description of data path functionality

Students can select needed microoperations for each unit of data path from the whole set of pre-designed microoperations when implementing a given algorithm or a function (like multiplication, division etc.). Each micro-operation has a gate-level implementation, and the number of gates determines the cost of the microoperation. The student can select thus a particular implementation of his algorithm (like *I* or *M-automata*, sequential or parallel *IM-automata*) meeting either the cost or timing requirement. 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.

Control Path. The control path is a microprogrammed controller [6], which implements Mealy FSM (Final State Machine). The controller consists of a microprogram table and an interpreter. The microprogram is developed by the user to realize a given algorithm based on the selected in prior resources of the data path. The user fills in the microprogram table as described below.

The first two columns of the microprogram table (Fig. 2) represent the address of current microinstruction and the address of the next microinstruction correspondingly. In case if its operation depends on the set of conditions C, the current microinstruction can be split into several rows. Only the proper row will be selected then according to the conditions. Columns 3 to 6 correspond to MUX and indicate which register (REG1…REGn) contents will be multiplexed into which functional unit (F1, F2, F4). Registers where the input data from Data IN (column "Input") will be written are specified in column "IN". The input data are the operands of the implemented algorithm (this will be further discussed in the next subsection).

Columns F1 to F4 stand for a certain microoperation selected in a corresponding functional unit (F1 to F4) in a certain clock cycle. The DMUX section is specified in next two columns. It shows to which register the data from functional units F3 and F4 will be written. Column "OUT" indicates the register, which content will be redirected to Data OUT. The last columns (C1, C2, …) stand for conditions. They are specified by 0, 1, X (don't care).

In Fig. 2 an example of algorithm of multiplication of two operands A and B is presented. The result of multiplication is stored in REG1 and fed to the data output.

Simulation Module. Simulation is carried out at the higher level by using Java subroutines (corresponding to functional units) which are activated by the control signals in the order given in the microprogram table according to condition values. The overall algorithm is the following:

```
Current State = 1
While Current State ≠ END
{
 Read Status Signals
 If (Current State = ADDR) and (Status Signals ⊂ C)
   Select the Row from microprogram table
 Else
   Error: "Simulation Error"
 Assign function to each enabled Functional Unit
 $R_{IN}$ = Data IN
 $R_{F3(out)} = F3(F2(F1(R_{F1(in)}), R_{F2(in)}))$
 $R_{F4(out)} = F4(R_{F4(in1)}, R_{F4(in2)})$
 Data OUT = $R_{OUT}$
 Current State = NEXT
}
```

Simulation can be carried out also at the gate level by using Structural BDDs. The process is also controlled by the data in microprogram table. The simulation data is stored in a subpanel Simulation Results. This data reflects the states of all the registers, outputs of all the functional blocks, data input and output of the device, current states at each clock cycle and condition signals. The simulation data can be used later by the student as a debugging info.

4 Teaching RT-Level Test

The following modules are intended to illustrate the principles of RT-level test.

For investigation of test problems (like test generation, fault simulation, etc.) we use the same microprogram repeatedly for several input data.

Test Generation. For test generation no special automatic means are provided. Either manually generated functional patterns or randomly chosen patterns (test data) can be used. A microoperation of one of the units F1, .., F4 can be chosen as a target for testing. It is assumed that the same microprogram will be repeated for a set of operands for the chosen microoperation. Then each repetition of the microprogram will be regarded as a test with a corresponding number. The operands must be written into the *Test Data Table* (Table 1). The efficiency (quality) of these tests is estimated by fault simulation.

Table 1 The test data

No	DA = 1	DA = 2	DA = 3	DA = 4
1	Data_11	Data_21	Data_31	Data_41
…	…	…	…	…
n	Data_1n	Data_2n	Data_3n	Data_4n

Fault Simulation. Fault simulation is carried out at the gate level by using Structural BDD model. Faults for the given block are inserted into BDDs. The simulation process is controlled by the data in microprogram table. The target of the fault simulation (a unit, and a microoperation in the unit) are selected by a student and then highlighted. The fault simulation data is reported by the applet in the *Fault Coverage Table*.

BIST Module. Usually functional test patterns do not provide a good fault coverage. Therefore scan-path with random test pattern generator is introduced. Special BIST subpanel with BIST results subpanel can be opened for this operation mode. By Scan-Path technology the inputs and the outputs of the combinational blocks in data path are directly accessible by scan-path registers TPG (random test pattern generator), SA (signature analyzer), and TPG/SA (combined TPG and SA) [7]. See Fig. 4.

Two modes are possible: BILBO (Built-In Logic Block Observer) or CSTP (Circular Self-Test Path) mode based on using TPG and SA, or a combined TPG/SA scan-path register correspondingly.

Both modes can be implemented in two ways: different settings for each combinational circuit to be tested, or the same setting for all circuits. The aim of the student's work is to find best settings. Again, the targets for testing are microoperations in blocks F1, … F4. Random test patterns generated by the TPG are saved, then fault simulated, and finally the fault coverage is displayed.

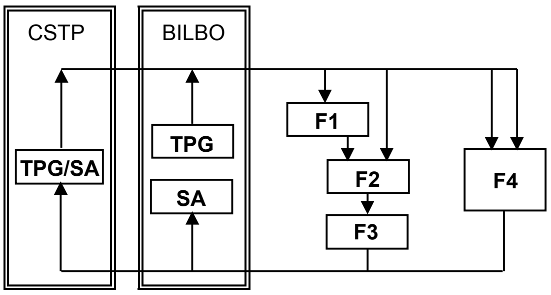


Fig. 4 Scan-path design

5 Conclusions

The conception presented allows to improve the skills of students in the area of digital hardware and SOC design connected with testing. The free-access basis and self-contained nature makes it easy for students even from foreign universities to use this system independently of time and place, and learn individually according to their own needs. This concept brings new forms of communication between teachers and students and up-to-date course material. The system's built-in multilingual support ensures easy integration into teaching courses of universities over the world.

The applet fully reflects the "easy action and reaction" conception which was taken as the major target for its creation. Each field, each functional unit, and other modules are click-able. Their functions can be changed or further adjusted. The reaction on each action is instantly reflected by highlighting and changed colors of selected modules.

On the other hand the tasks chosen for training represent simultaneously real research problems. This provides students with dynamic environment to experiment with and to find interesting solutions for stated problems.

At the moment the applet is still in its beta version stage. However the most of the functionality is already implemented. The fault simulation and BIST are the only missed interactive modules planned for future work.

Acknowledgements: This work is partially supported by the Thuringian Ministry of Science, Research and Art (Germany), by EU V Framework project REASON, and by the Estonian Science Foundation (grants No 3658, 4876 and 4300).

References:

[1] M.L. Bushnell. Increasing Test Coverage in a VLSI Design Course. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1133.

[2] J. Harrington. VLSI Design 101 – the Test Module. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1134.

[3] V.D. Agrawal. Increasing Test Coverage in a VLSI Design Course. International Test Conference, Atlantic City, NJ, USA, 1999, p. 1131.

[4] H.-D. Wuttke, K. Henke, R. Peukert. Internet Based Education - An Experimental Environment for Educational Purposes. Proc. of IASTED, May 6-8, Philadelphia, PA USA, pp. 50-54, 1999.

[5] Teaching system URL: http://www.pld.ttu.ee/dildis/automata/applets/9/

[6] Armstrong J. R., Gray F. G. Structured logic design with VHDL. *Prentice-Hall*, Englewood Cliffs, 1993, 482 p.

[7] Abramovici M., Breuer M.A., and Friedman A.D. Digital systems testing and testable design. *IEEE Press,* New York, 1999, 652 p.