EVENT-DRIVEN SIMULATOR FOR E-LEARNING

Nikolay Georgiev Chillev, Tania Krumova Vassileva

Departament of Computer Science, *Departament of Electronics, Technical University of Sofia,
Kliment Ohridsky 8, 1756, Sofia, Bulgaria, phone: +359 2 965 3731, e-mail: nikic@lark.tu-sofia.bg

Keywords: Schematic editor, logic simulator, e-learning

The paper outlines the architecture and implementation of a web based utility, which integrates an interactive graphical design tool with a module for schematics comparison and a discrete event-driven simulation engine. Signal editor, which allows user-defined signal waveforms to be assigned to input ports, is also provided.

The interactive schematic editor/logic simulator is designed as a Java applet and allows user to create, edit, save and simulate digital circuits. The event-driven logic simulator keeps track on current time, current time step and the event list that holds future events. Delays are also considered, permitting simulation of glitches and hazards in real digital circuits. The simulator provides feed back to the user and can communicate with Learning Management systems, being this way a useful tool in any e-learning environment.

1. INTRODUCTION

The ability to simulate a system, prior to its physical implementation is very essential both for students and for developers, who design new products. Simulators provide an economical means of understanding and evaluating the performance of both abstract and real world systems. Logic simulators are essential tools for design verification in the field of digital electronics. They can reduce the cost and required time for education and increase the quality of newly created products.

Using simulators for educational purposes in e-learning environment face many problems.

First of all, most of web-based learning resources, wide spreading today are static without ability for interactions. They just accept input data, no matter correct or wrong, and outputs result, without any comments or guidance. From didactical point of view such simulators don't serve as a useful tool especially for open, distant or self-learning, where nobody can help in errors detecting and result analyzing.

In addition to being efficient and easy to use, modern day simulators should be easily extensible so that the behavior and performance of a wide variety of systems may be studied. In the field of digital simulation most of the web based simulators are just functional without tracking gate delays. Normally these simulators use only gate level primitives and flip-flops without ability to provide custom components, custom signals as well as hierarchical design.

Other problem is that every simulator has its own user interface. Engineers regularly use Computer Aided Design (CAD) software products in their everyday work, so the training with CAD tools is an essential part in technical education. The front-end part of most electronic commercial CAD systems is the schematic editor.

After several decades of evolution commercial schematic editors offer intuitive and almost similar user interface. An educational schematic editor, generalizing the features of popular commercial products can facilitate adaptation to such products and can be used as a portal to them.

Learning Management Systems (LMS) provide a sophisticated methodology to track student's progress in online courses. Commercial simulation tools and many online courses are not designed to communicate with LMS. This is the reason most of the courses to consider mainly theoretical aspects without training with simulators.

Globalization of teaching market and world wide distribution of existing courses imposes requirements for internationalization of education materials and associated simulation tools.

Increasing demands of today education for intensive use of simulation requires a simulator available at any time and any place, which can easily be incorporated in e-learning courses, can ensure guidance during tasks execution, save results for further use, provide an opportunity to change tasks by the user, communicate with LMSs and can easily be adapted to any language for wide dissemination.

To address all these issues a web based event driven logic simulator is developed, capable to simulate real digital circuits and to cope with increasing requirements for interactions, guidance, feedback, internationalization and communication with LMSs.

2. ARCHITECTURE OF THE SYSTEM

Developed web-base system for digital circuit simulation consists of three basic modules – schematic editor, netlist comparison module and event driven simulator.

2.1 Sophisticated Schematic Editor

The schematic editor provides all common functions, supported by commercial schematic editors. It is an upgraded version of the web-based tool, reported in [1].

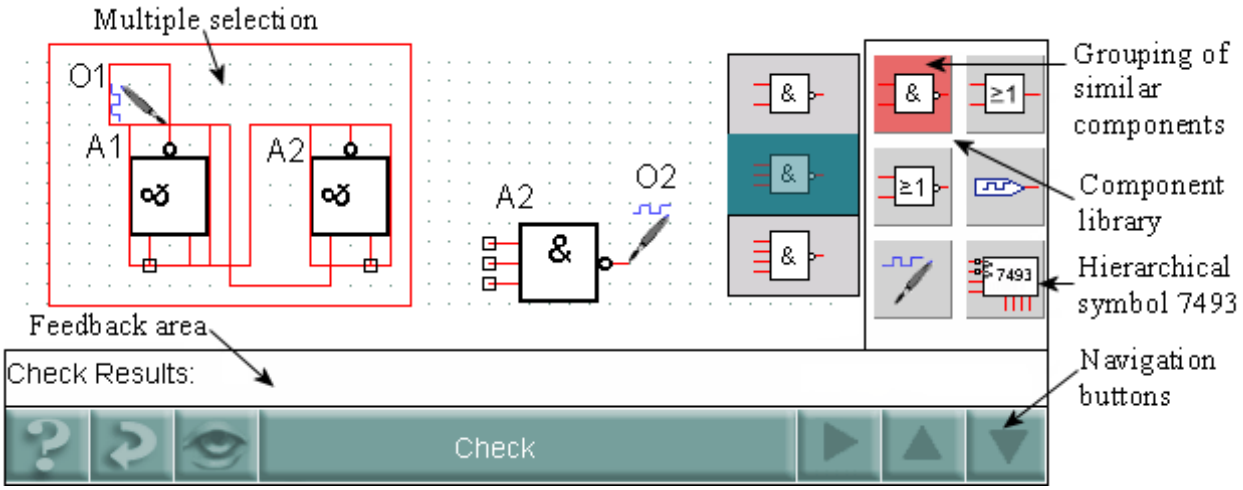

Fig. 1 User interface of the schematic editor

Multiple components selection is provided (Fig.1), which allows components to be grouped for moving, dragging, or deleting this way increasing user productivity.

A simulator library supports a wide range of logic elements and user defined blocks for *hierarchical* design. Component library is flexible and can be easily configured with different components. Gates with the same logic function, but with different number of inputs like 2-, 3- or 4-input NAND for example are grouped in lists, which are popped up from a button with a common logic function, as shown in Fig. 1. After selecting a component as an instance from the library it can be moved, rotated, mirrored and finally placed in the sheet.

Any component or input pin can be associated with appropriate *attributes* to add performances to the item. Logic components have an associated delay, which in fact is performance setting and determines the time, needed from signal arrival at the input till the output reaction is available at the output. If the signal is shorter than the gate delay, it is ignored by the simulator. Attributes can be entered or edited through a popup dialog box. Attributes are labels and gate delays for components, signal types for input stimulus etc.

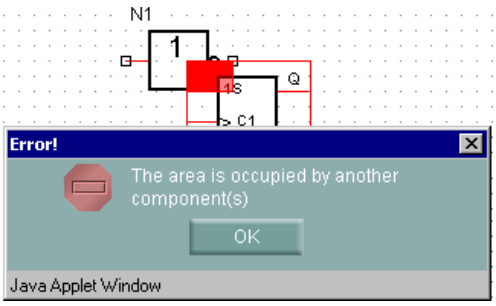

Fig. 2 Components' overlapping is disabled

Common problem for many schematic editors, including commercial, are multiple instances of the same component placed at one place. Such error creates a problem during the schematic check phase, when overlapping is reported. The error correction is difficult since it is not obvious on the screen where a multiple instant component exists. To prevent such situation, components' *overlapping is disabled* at the moment of placement by checking corresponding coordinates. If a current problem is detected a warning message is displayed in a popup dialog box, as shown in Fig. 2.

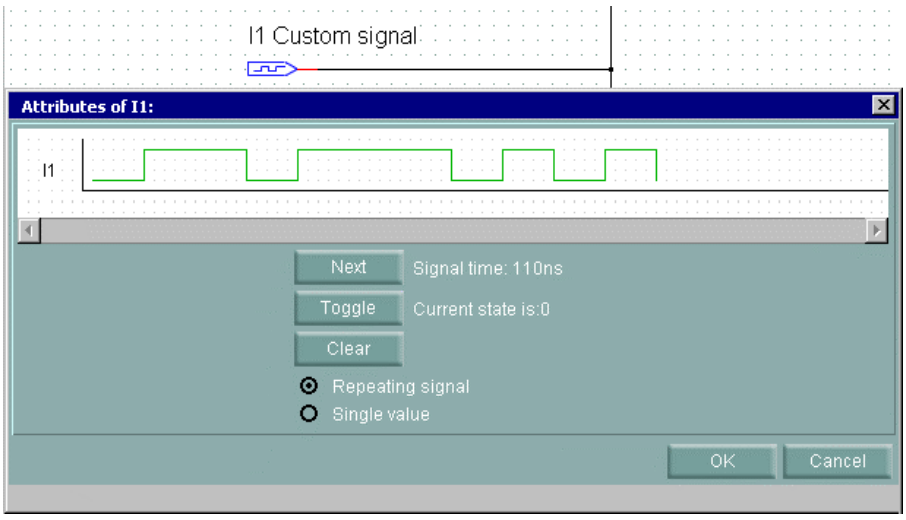

Fig. 3 Signal editor window

Input stimulus can generate waveforms with fixed logic levels – logic 0 (Low) or logic 1 (High), clock signals with different periods or custom signals with single or repeating sequences. Custom signals can be configured by developed *graphic signal editor*. When custom signal is selected, a signal editor window (Fig. 3) is displayed. A string of zeroes and ones can interactively be created or modified with Next and Toggle buttons. This sequence is used during the simulation, as a stimulus passed to the input of corresponding logic element.

The schematic editor is implemented as a Java applet. Due to a security reasons Java applets are restricted to access files on the client's computer. For educational purposes it is important to read and write schematics files on the local file system. The learner should be able to save files for further use. To overcome security issues a signed applet is used, which enable *saving/loading* files on a client computer.

2.2 Netlists Comparison Module

Once the schematic has been completed, the user should compare it with the pre-defined assignment. Both netlists, one of created schematic and the other of correct circuit, are compared and appropriate messages are given for any obtained difference. At first the presence of all required components is verified. After, a check for extra components is performed, followed by an inspection for unconnected pins. If all these checks are successfully passed, circuit connectivity is examined by a topological comparison between the graph extracted from drawn schematic and the initial graph of assignment. Electrical Rules Check (ERC) is performed before simulation. Outputs fixed to grown or bias, unconnected inputs, two or more outputs connected together are among common electrical rules under control to prevent incorrect logic function, self oscillation or damaging current through the circuit.

Once the circuit has been successfully checked, the user is allowed to simulate the schematic.

2.3 Event-Driven Simulator

The event-driven simulator is a powerful tool for verifying and understanding operation of digital circuits. The simulator is capable to simulate any digital circuits, containing logic gates, flip-flops and hierarchical components. Logic gates and flip-flops are considered as basic primitives and are modeled with their truth tables. Hierarchical components consist of basic primitives, connected in appropriate way. Signal's propagation starts from input ports, goes thought the components, finishing at the outputs. The waveforms, generated by input stimulus generators and obtained at outputs, marked with probes are included in the result waveform sheet for displaying. Arbitrarily long simulation can be displayed, allowing user to observe all interesting events, arising in the schematic. Event-driven simulator tracks the gate delay and can simulate situation occurring in real digital circuit. Visualization module displays glitches, hazards due to signals race and self oscillations. Time settings for displaying are set interactively by the user. A time marker is provided, which permits time measurements between any two points of the waveforms. Simulation window with simulated waveforms is shown in Fig. 5.

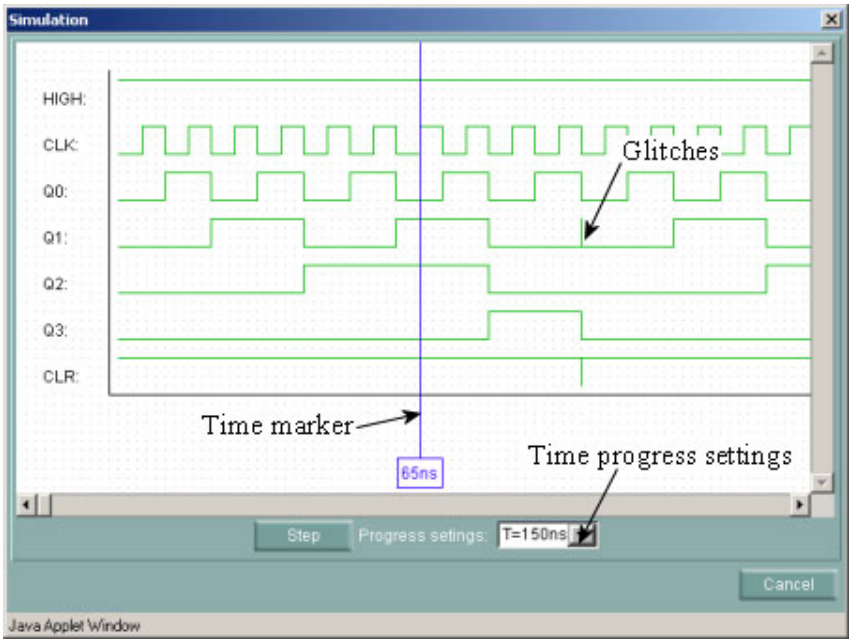

Fig. 5 Simulation window

3. IMPLEMENTATION

Developed event-driven simulator and associated modules for schematic entry and netlists comparison are implemented as a Java applet. Java is object oriented platform independent language. Allthough it is "greedy" for memory and CPU cycles it is proper for the development of interactive learning materials. Due to security reasons applets are restricted to access files on the client's computer. With the introduction of Java 1.2 and JRE programmers one can sign applet's jar file. This way the user is assured for the programmer's identity. With the signed version of the applet, user can load and save schematics. Allthough Java has advanced APIs for user interfaces (Swing) and parsing XML files (JDOM), older libraries like AWT are used to be compatible with popular browsers [2]. AWT provides a framework for custom components, which can be internationalized. All components are released as custom components, extending base Component class. The layout is flexible, so the applet can fit into random layout and size. Configuration is organized in XML files, because of their structure and readability. Microstar's Aelfred, open source parser, is used for XML parsing. All resources of the applet, zipped in a jar file, do not exceed 200kb, a satisfactory size concerning slow Internet connections. Schematics are saved as a layout description in XML file, by the signed version of the applet. The assignment does not contain any layout information. It contains only topological information, thus reducing approximately twice the file size. Internationalization is assured by separating language dependent texts in external file for easy translation.

The comparison between assignment and drawn schematic is in fact a topological comparation of the schemes graphs. The idea of "graph polymorphism" algorithm [3] has been used. Each component has a weight. It is formed as a hash function of it's type and properties. The developped algorithm finds equal subgraphs in the two schemes and then tries to append more components to them. At first subraphs consist

of single components. The algorithm appends new vertices to the two subgraph only if they have same weights and links between equivelent pins. Two graphs are equal, only if all components are appended to equivenent subgraphs. If the algorihtm can not make a step ahead, it returns one step behind and tries combinations with other components [4]. In fact this is a backtracking algorithm, which complexity in the worst case is O(N!), where N is the number of components in the schematic, but the implemented restrictions limit the number of combinations and avoid the "combinatorial explosion" [6]. The algorithm is fastest when the components of the user's schematic are entered in the same order as the components of the assignment, it is slowest when the components in the two schemes are in completely reverse order. For the schematic with about 20 components, execution of the algorithm takes 300 ms on a PIII/600MHz, which is an interactive speed for mostly spreaded computer classes.

The simulator runs at gate level. All output ports are modelled with three states – zero, one and undefined (X). If some input port of a component is connected to an undefined output, the outputs of this component remain undefined. Each change of state (e.g. an event) at the component output port is appended to a queue. First change of states occurs at the input ports when stimuli are applied. The last change of states occurs at the schematic outputs. An algorithm, known as "heap sort" schedules the earliest events for propagation [5]. The complexity of event insertion and searching in the queue is Olog(N), where N is the number of pending events in the queue. This is enough fast to produce animation of waveforms in real time.

4. CONCLUSION

The developped web based interactive simulator performs event-driven simulation of digital circuits considering gate delays. It provides guidance at any step of task execution giving appropriate feedback to the user. The simulator is implemented as a Java applet. It is platform independent, compatible with LMSs, and can easily be adapted to different languages. These features make them widely accessible and proper for integration in any e-learning environment. The simulator is an essential part of many commercial e-learning products in the field of digital electronics [7]. As a further work it can be extended to simulate all kinds of circuits, which is modeled with discrete-events, as Local Area Networks for example.

5. REFERENCES

[1] Chilev N., T.Vassileva, V. Thchoumatchenko, *Schematic editor for Educational Purposes on the World Wide Web*, The eleventh International Scientific and Applied Science Conference, Electronics ET'2002, Book 2, pp.154-159

[2] Eckel B., *Thinking in Java*,Prentice Hall, New Jersey 1998

[3] Sendov B., *Dynamic programing,* Prosveta, Sofia 1985

[4] Manev K., *Discrete mathematics,* Prosveta, Sofia 1997

[5] Nakov P., *Introduction to computer algorithms,* Top team books, Sofia 1998

[6] Elinger T., *VLSI Design*, Addison Wessley 1988

[7] http://www.ingenatic.com/site/en/home/content/html/prod_demos.htm