LogiFlash – A Flash-based Logic-Simulator
for educational Purposes

Proposal for an interactive session at the Ed-Media 2003

Markus Damm, Bernd Klauser, KlausWaldschmidt
damm | klauser | waldsch @ti.informatik.uni-frankfurt.de

Abstract: A graphical logic-simulator implemented as a Macromedia Flash animation is presented. It can be used to construct interactive logical circuits, which can be embedded into (X)HTML pages or other Flash-Animations. There is also the possibility to create interactive exercises with these circuits, whereas the solutions can be checked automatically.

1. Introduction
LogiFlash is a simple graphical logic-simulator implemented as a Macromedia-Flash-Animation. The advantage over other existing, highly sophisticated graphical Logic-Simulators ([1][2][3], to name a few) is that they are embeddable into (X)HTML-Pages. In learning objects on logical circuits, LogiFlash can be used to present circuits and to demonstrate their behaviour. It also supports interactive exercises of the form “Wire the given nand-gates to a xor-gate.”

LogiFlash has a focus on visualisation, e.g. by highlighting those wires which carry a logical one. Therefore it is also convenient to be used in lectures to demonstrate how certain logical circuits work, while the lecturer can make the same animations available via Internet to the students for further studies. This is actually being done in the ongoing 2002/2003 winter semester at the universities of Karlsruhe, Lübeck and Kaiserslautern in Germany, and in the 2002 summer semester at the university of Frankfurt am Main. The experiences so far are very positive; especially the students appreciate the interactivity within a simple framework. There is also the possibility to integrate LogiFlash within other Macromedia-Flash animations.

The development of this simulator was made within the scope of the project Wissenswerkstatt Rechensysteme (“WWR”, Knowledge Factory for Computing Systems), founded by the German Ministry of Education and Research. It is a compound project of twelve German universities, including those mentioned above, with the goal of implementing XML-based multimedia learning objects for technical computer science ([4][5]).

2. Objectives of the interactive session
The intended audience of this session are people engaged in education of technical computer science, but also people who are interested in basics about logical circuits. It could also attract those who are involved in programming simulations and learning objects of various kinds using Macromedia Flash or comparable platforms.

After an overview of the intentions of LogiFlash, its concept and architecture, the different uses described above will be demonstrated. Afterwards, the participants get the opportunity to try out LogiFlash in various forms: trying out interactive circuits, using animations with integrated LogiFlash-circuits, working on exercises and finally building their own circuits. Since everything will be used within Internet-browsers, the technical requirements are minimal.

3. The Concept
LogiFlash consists of two parts: The editor and the viewer. Both are pure Macromedia Flash-animations, which can be used in any browser with integrated FlashPlayer 6 plugin.

With the editor, we can construct arbitrary logical circuits with the most common gate types. As signal sources, there are switches and oscillators. As signal-drains, there are lamps and seven-segment-displays. The signals in LogiFlash are logical ones or zeros. If an input of a gate is not connected to a wire, this will be interpreted as a logical zero.

In LogiFlash, everything is simulated during construction time, so students can learn about the functioning of the gates they use. For example, if a wire is connected to the output of a NAND-gate, whose inputs carry a logical zero, this wire will immediately be illuminated, since it then carries a logical one. This immediate simulation can be turned of if the construction process is focused.
We can also add text to the circuit, and especially text-labels to the signal sources and drains. Currently supported are different sizes, bold and italic, over- and underscores, as well as sub- and superscripts.

LogiFlash-circuits can be saved and reloaded in a specially developed XML-format. Due to the security restrictions, saving can only be done so far using a copy & paste solution out of a dynamic Flash-textfield, while loading the circuits is unproblematic. For the future, a solution using a web-server application will be developed.

The editor also allows the creation of interactive exercises. For this, a correctly working circuit (the sample solution) is constructed at first. Then a test pattern is entered by pressing the appropriate signal-source-switches. The corresponding outputs are stored together with the test patterns. Afterwards, all (or a subset of the given) wires can be deleted, or errors can be built in by changing the wire connections. The test-patterns are then stored together with the (incomplete or defective) circuit in XML. Exercises such constructed can be edited by the student, and the solution is then checkable automatically with these test patterns.

The viewer offers the functionality to use circuits constructed as above. It has two different uses:

- It can be embedded into a (X)HTML-Page, whereas the name of the circuit (or a list of names) is passed as a parameter. The viewer then loads and displays the circuit. If no test pattern is present, the user can only handle the signal-sources and watch, how the circuit is working. In the other case, it is possible to edit the circuit by adding and deleting wires, moving and connecting the given components, and finally checking the solution. If a list of circuits was passed, they can be browsed through.
- It can be loaded by another Flash-Animation, whereas it offers functions for the encapsulating animation to control it and to synchronize it with its own content.

4. The architecture and simulation concept

Both the editor and the viewer are based on a Sub-MovieClip (the “LogiFlash-engine”), which offers the full functionality described. The editor contains the appropriate controls and menus to use all of the engines features. The viewer contains a reduced menu set sufficient for its purposes (see screenshots).

The graphical components (i.e. the gates, sources, drains, wires, nodes, in- and outputs) are implemented as ActionScript-classes. These classes are registered to the corresponding Movie-Clips, which define their graphical appearance. The picture to the left shows the inheritance-chain of the Gate class. The inheritance at instantiation time in the first level is achieved by setting the __proto__ property to the desired “gate-function-class” in the Gate constructor.
Every class also contributes to the simulation process. This is depicted below, where we see the flow of a signal through a gate (the in- and outputs are subclips of the gate). The flow goes as follows:

1. The node passes the signal to its adjacent wires.
2. The wire passes the signal to its other end-node.
3. The node passes the signal to the gate-input it is connected to.
4. If the input is inverted, the signal is negated. If that signal is different from its current state, the input informs its gate about a new signal and stores it as new state.
5. The gate informs the simulation-controller, that it has gotten a new signal and needs to propagate new output signals.
6. The simulation controller schedules the gates waiting to propagate their output.
7. The simulation controller tells the gate to propagate its output.
8. The gate computes new output signals according to the states of its inputs and passes them to its outputs.
9. If the output is inverted, the signal is negated. If that signal is different from its current state, the output passes the signal to the node it is connected to and stores it as new state.
10. The node passes the signal to its adjacent wires.

During this process, the wires and nodes change their appearance according to the signals they carry. The simulation-controller schedules the gates waiting to propagate. As of now, this is done such that signals are propagated through the circuit in a breath-first-search manner, while feedback signals are delayed. Other simulation concepts are also realizable within this framework.

5. Conclusion and Future Work

LogiFlash is a simple, but for the educational needs in the field of basic technical computer science sufficient, presentation-oriented logic simulator. It enables teachers to demonstrate the principles of logical circuits interactively, and to create interactive exercises concerning logical circuits. It is not intended for designing and simulating sophisticated circuits or even processors. LogiFlash is currently in a trial phase. Despite the usual beta-testing it is also checked if there is a need for more circuit components (which could be implemented easily), or alternative simulation concepts. Another task is due to the use of LogiFlash within the XML-based WWR learning objects. Those modules are designed such that they can be converted not only to XHTML, but also to PDF, and therefore it is also necessary to have printable versions of the LogiFlash-circuits. Since these are stored as XML, it is straightforward to convert them via XSLT into SVG. This work is currently underway together with our WWR-partners. A similar tool for register-transfer-logic is planned.

So far, the gates are displayed according to the German DIN-standard. For the future, we are planning to include an option to switch between DIN and the American standard.

References