# From CMOS-Gates to Computer Architecture: Lessons Learned From Five Years of Java-Applets

Norman Hendrich

Dept. of Computer Science, University of Hamburg
Vogt-Koelln-Str 30, D 22 527 Hamburg
Email: hendrich@informatik.uni-hamburg.de

**Abstract** Static images in textbooks are only poorly suited to illustrate complex non-static processes. A typical example is the task to explain and visualize the parallel data-transfers in a microprocessor pipeline or cache. One candidate solution to this dilemma is the use of dynamic media in teaching, e.g. videos or interactive simulations. Shortly after the first version of Java became available, we started to build a library of Java-applets to visualize several topics in logic design and computer architecture. In this paper, we will present the rationale behind our applet collection. Based on user feedback and our experiences in teaching, we derive some guidelines concerning the design and use of Java applets in teaching.

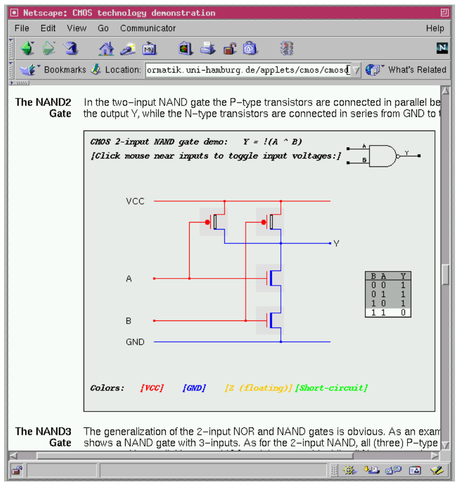
## 1 A Library of Java Applets

Many topics in microelectronics education are concerned with complex sequences of events, often occurring in parallel and even asynchronously. It is quite a challenge to explain these processes just with the help of static images. An excellent example is the description of the MIPS pipeline in [1], where a dozen pages are allocated to color graphics, each of them showing one step of pipelined instruction execution. However, despite this effort, still only a few of the pipeline states and hazards are covered.

The use of interactive simulations to illustrate and visualize such processes suggests itself. As Java-applets are portable and require no installation, they seem an ideal candidate to implement such interactive simulators. Also, the applets are embedded into HTML pages which can directly provide the applet user-guide and additional documentation about the applets' context. Finally, the performance problems often attributed to Java are usually not a concern for teaching.

Shortly after Sun Microsystems announced Java [2], we therefore started to build a collection of applets [3], each one of them illustrating a topic which had proved difficult for our students. Currently, public access is possible for the following applets via our applet homepage, tech-www.informatik.uni-hamburg.de/applets:

- demonstration of static CMOS-gates
- Karnaugh-Veitch diagrams and logic minimization
- binary decision diagrams
- an interactive finite-state machine editor and simulator
- an highly-animated simulator for a simple von-Neumann processor
- a complete applet-based course in computer architecture, from basic RT-level components to a whole computer system including user-configurable caches
- the HADES digital system design and co-simulation framework [4]


Netscape: CMOS technology demonstration
File Edit View Go Communicator Help
Bookmarks Location: ormatik.uni-hamburg.de/applets/cmos/cmosd
What's Related
VCC
A
B
GND
Y

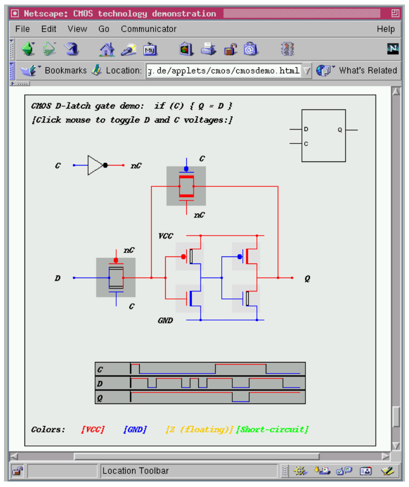
The NAND2 Gate In the two–input NAND gate the P–type transistors are connected in parallel be the output Y, while the N–type transistors are connected in series from GND to

*CMOS 2-input NAND gate demo: Y = !(A ^ B)*
*[Click mouse near inputs to toggle input voltages:]*

| B A | Y |
|---|---|
| 0 0 | 1 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

*Colors: [VCC] [GND] [Z (floating)] [Short-circuit]*

The NAND3 Gate The generalization of the 2–input NOR and NAND gates is obvious. As an exam shows a NAND gate with 3–inputs. As for the 2–input NAND, all (three) P–type


Netscape: CMOS technology demonstration
File Edit View Go Communicator Help
Bookmarks Location: g.de/applets/cmos/cmosdemo.html
What's Related
D
C
Q
C
nC
VCC
GND
C
D
Q
Location Toolbar

*CMOS D-latch gate demo: if (C) { Q = D }*
*[Click mouse to toggle D and C voltages:]*

*Colors: [VCC] [GND] [Z (floating)] [Short-circuit]*

Figure 1: The CMOS-gate demonstration applets use colors to indicate voltage levels and allow the user to interactively play with static CMOS gates and flipflops. Two examples are shown: 2-input NAND-gate (left), D-LATCH with waveforms (right).

Written at the end of 1995, our CMOS-gate demonstration turned out to use some of the very first Java-applets in Europe, and remains one of the most frequently accessed pages on our website. Based on a 0/1/X/Z switch-level model, each applet allows the student to toggle the input voltages for the transistors and calculates the corresponding output values. Different colors are used to visualize the logical values on the wires, and the transistor symbols are updated to show which transistors are conducting (figure 1). Usually, just a few minutes of playing with the applets allow the students to just “see” the complementary nature of the p- and n-paths in the static CMOS-gates.

Another example applet, the interactive Karnaugh-Veitch map editor, is shown in figure 2. It supports both sum-of-products and product-of-sums minimization of user-defined logical functions (with 2..6 inputs and 1..8 outputs). Unlike a textbook description which can only show a few steps of the minimization process, the applet instantly updates the circuit schematic corresponding to the terms selected in the K-map. Again, colors are used to indicate the user-selected terms (cubes) and the corresponding gates in the logic realization. In “student mode”, the circuit schematic highlights the gates corresponding to a cube in the K-map, before collapsing them to show the minimized schematic.

## 2 An Applet-based Course in Computer Architecture

Based on the experiences with the simple applets, we proceeded to create a set of applets as interactive illustrations for a tutorial on computer architecture [5]. All applets use the same conventions for visualization (like highlighting active registers or memory cells in red) and include context-sensitive textual explanations. To save valuable screen space, a very simple 16-bit accumulator machine with a basic instruction set is used. This allowed us to display all machine registers together with a sizeable part of the memory on each screen. The course starts with simple applets to illustrate the basic building blocks like registers, memory, and ALUs. Next, the 16-bit microprocessor is introduced and

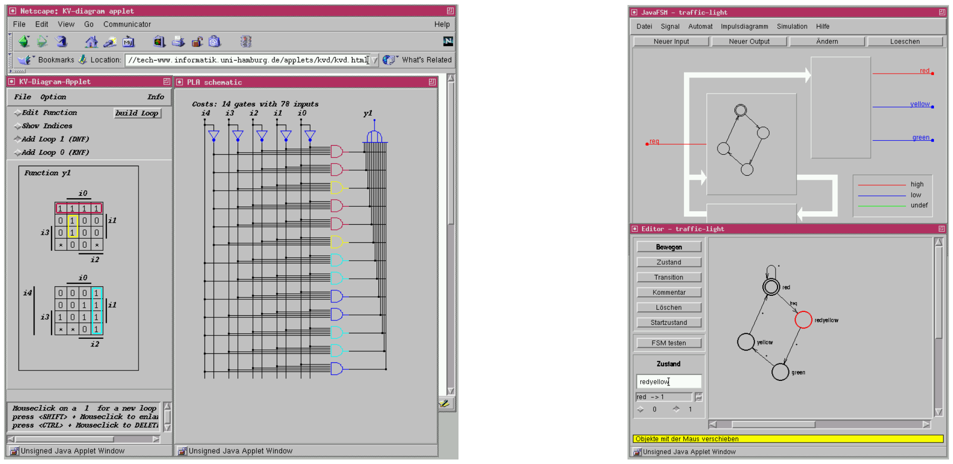

Netscape: KV-diagram applet
File Edit View Go Communicator Help
Bookmarks Location: //tech-www.informatik.uni-hamburg.de/applets/kvd/kvd.html What's Related
KV-Diagram-Applet
File Option Info
Edit Function
build Loop
Show Indices
Add Loop 1 (DNF)
Add Loop 0 (KNF)
Function y1
PLA schematic
Costs: 14 gates with 78 inputs
Unsigned Java Applet Window
JavaFSM - traffic-light
Datei Signal Automat Impulsdiagramm Simulation Hilfe
Neuer Input Neuer Output Ändern Loeschen
red yellow green
high low undef
Editor - traffic-light
Bewegen Zustand Transition Kommentar Löschen Startzustand FSM testen
Zustand
redyellow
Objekte mit der Maus verschieben

Figure 2: Our interactive KV-diagram editor (left) and a simple state-machine editor with integrated simulator (right).

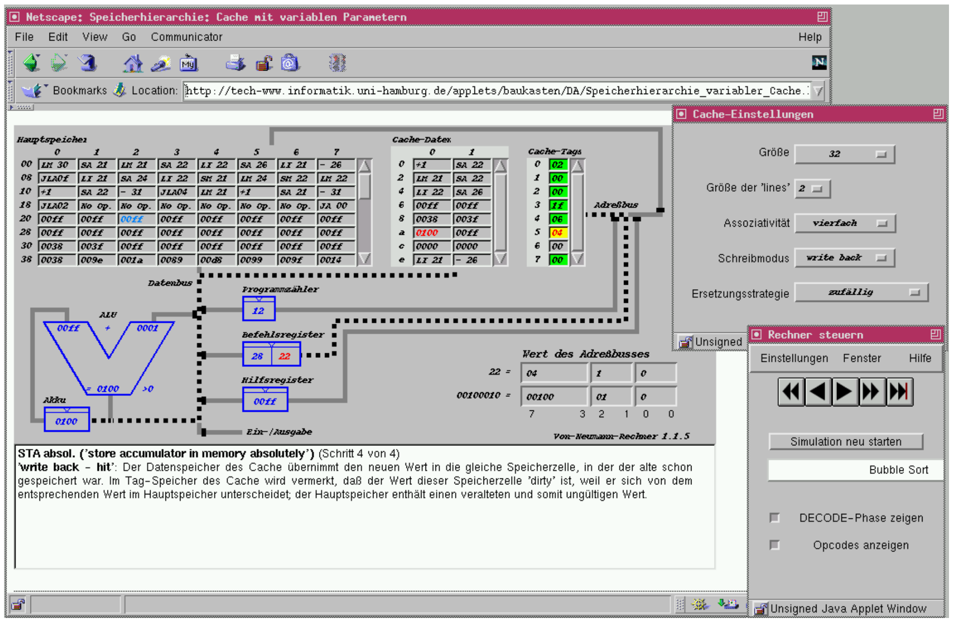
several applets are used to demonstrate first a few elementary instructions and then the full instruction set. A separate applet, based on the 68000-series architecture, is used to illustrate typical addressing modes.

Perhaps the most complex applet is shown in figure 3. It combines the simple microprocessor with a fully-configurable cache. The user can select many parameters of the cache architecture (including cache size and line size, direct-mapped, 2-way or 4-way associative organization, write-through or write-back, random or LRU replacement). For each of the user selections, the applet generates the corresponding simulator, including context-sensitive help-texts, and allows to run programs on the selected architecture while gathering miss-rate statistics. As shown in figure 3, data transfers are animated and several colors are used to indicate active data elements and the state of cache lines and tags. Almost none of the visualization aids would be possible on paper.

## 3 Experiences and Applet Guidelines

While eye-catching animations easily impress one-time users, it is much more difficult to make them really helpful for teaching. In fact, one of our highly animated applets (not shown here) proved much less useful than an older text-mode simulator. Besides the standard simulation controls like run or single-step, we always found it necessary to allow the user to take steps back, in order to repeat difficult parts or to explore different approaches. Unfortunately, this seemingly simple requirement often results in a very complex implementation, like the use of saving and restoring simulator checkpoints.

Apart from some annoying (and platform-dependent) bugs in all major browsers, we found Java a good environment to build complex simulators. For example, the cache applet shown above may easily exceed the memory limit enforced by some browsers. Other problems are the limited support for audio and the access to user data, restricted by applet security. While Java performance has been a major obstacle, it is no longer a problem on modern hardware. Full screen animations are possible in all current browsers, and simulation speed may surpass one million events or instructions per second.


Netscape: Speicherhierarchie: Cache mit variablen Parametern

Location: http://tech-www.informatik.uni-hamburg.de/applets/baukasten/DA/Speicherhierarchie_variabler_Cache

Hauptspeicher

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 00 | LM 30 | SA 21 | LM 21 | SA 22 | LI 22 | SA 26 | LI 21 | - 26 |
| 08 | JLA0f | LI 21 | SA 24 | LI 22 | SM 21 | LM 24 | SM 22 | LM 22 |
| 10 | +1 | SA 22 | - 31 | JLA04 | LM 21 | +1 | SA 21 | - 31 |
| 18 | JLA02 | No Op. | No Op. | No Op. | No Op. | No Op. | No Op. | JA 00 |
| 20 | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff |
| 28 | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff |
| 30 | 0038 | 003f | 00ff | 00ff | 00ff | 00ff | 00ff | 00ff |
| 38 | 0038 | 009e | 001a | 0089 | 00d8 | 0099 | 009f | 0014 |

Cache-Daten

| | 0 | 1 |
|---|---|---|
| 0 | +1 | SA 22 |
| 2 | LM 21 | SA 22 |
| 4 | LI 22 | SA 26 |
| 6 | 00ff | 00ff |
| 8 | 0038 | 003f |
| a | 0100 | 00ff |
| c | 0000 | 0000 |
| e | LI 21 | - 26 |

Cache-Tags

| | |
|---|---|
| 0 | 02 |
| 1 | 00 |
| 2 | 00 |
| 3 | 1f |
| 4 | 06 |
| 5 | 04 |
| 6 | 00 |
| 7 | 00 |

Programmzähler: 12
Befehlsregister: 28 22
Hilfsregister: 00ff
Akku: 0100

Wert des Adreßbusses

| | | | |
|---|---|---|---|
| 22 = | 04 | 1 | 0 |
| 00100010 = | 00100 | 01 | 0 |
| | 7 3 | 2 1 | 0 0 |

Von-Neumann-Rechner 1.1.5

**STA absol. ('store accumulator in memory absolutely')** (Schritt 4 von 4)
**'write back – hit'**: Der Datenspeicher des Cache übernimmt den neuen Wert in die gleiche Speicherzelle, in der der alte schon gespeichert war. Im Tag-Speicher des Cache wird vermerkt, daß der Wert dieser Speicherzelle 'dirty' ist, weil er sich von dem entsprechenden Wert im Hauptspeicher unterscheidet; der Hauptspeicher enthält einen veralteten und somit ungültigen Wert.

Cache-Einstellungen

Größe: 32
Größe der 'lines': 2
Assoziativität: vierfach
Schreibmodus: write back
Ersetzungsstrategie: zufällig

Rechner steuern

Simulation neu starten
Bubble Sort
DECODE-Phase zeigen
Opcodes anzeigen

Figure 3: The *cache applet* from our applet series on computer architecture. It demonstrates and visualizes the function of both direct-mapped and associative caches. After selection of the cache architecture, the user can load and run example programs (or single-step through the programs). The screenshot shows the applet and the configuration and simulation control dialog windows. Main memory, cache data and tags, and the CPU registers are all visible at once. Note the color encoding used to highlight valid, invalid, and dirty cache entries, the (context-sensitive) help text, and the visualization (animated dotted lines) of the data transfers.

All applets are currently used in lectures and lab courses at our department, despite the fact that the required infrastructure (data projector etc.) is not always readily available. While we do not monitor external access to the applets, the usage statistics for our webserver show that the applet pages are among the most active. Overall, the students' feedback to the applets has been quite positive. This is especially true for the simple applets, which can be used right-away without studying user manuals. Unfortunately, the computer architecture tutorial is still only available in the original German version.

## References

1. Hennessy, J.L. and Patterson, D.A., *Computer organization and design: the hardware/software interface*, Morgan Kaufmann Publishers, 1998
2. Gosling, J. and McGilton, H., *The Java Language Environment*, Sun Microsystems, 1995
3. Hendrich, N., The Hamburg computer architecture applet gallery homepage, http://tech-www.informatik.uni-hamburg.de/applets/
4. Hendrich, N., A Java-based framework for simulation and teaching: Hades, in *Microelectronis Education, Proc. EWME-2000*, 285-288, 2000
5. Kelling, C., *Ein Rechnerbaukasten: Simulation und Visualisierung der Zeitabläufe in Rechnerarchitekturen mit Java-Applets*, MSc. thesis, Dept. Computer Science, Univ. Hamburg, 1998