Digital LC-2
From Bits & Gates to a Little Computer

Albert Cohen
A3 group, INRIA Rocquencourt

Olivier Temam
LRI, Université Paris-Sud

May 26, 2002

Abstract

This paper describes *DigLC2*, a gate-level simulator for the *Little Computer 2* architecture (LC-2) [3] which serves to strengthen the bottom-up approach to teach computer architecture. *DigLC2* is based on *Chipmunk*'s digital circuit simulator [1]; the circuit is freely available on the web and ready to use.

1 Context and Presentation

The principle of our approach is to combine a bottom-up presentation of computer architecture (from digital gates to processor and system) with an intuitive graphical gate-level design tool. This combination enables students to truly understand the logic behind processor design and internal processor workings, and simultaneously to gain confidence in the acquired knowledge thanks to experimental validation of concepts with a gate-level processor simulator (*DigLC2*). Based on this solid knowledge, we believe students are much more likely to quickly grasp and master new information about the evolution of processor design.

DigLC2 [2] is a gate-level simulator for the *Little Computer 2* (LC-2), as described by Patt and Patel in their introductory textbook on architecture and programming [3]. Unlike the existing LC-2 functional simulator [4, 5], it provides a detailed description of all processor components at the gate-level, so that students can themselves build a full processor using only elementary gates (AND, OR, NOT and Tri-State), thereby demystifying processor architecture.

The *DigLC2* simulator started as a support tool for a course at École Polytechnique (France) [6]. Designed to cooperate with the LC-2 functional simulator and assembler environment [4, 5], we wanted it robust and modular for practical lectures, as intuitive as possible to serve as a basis for student projects, and versatile enough to explore fundamental architecture and programming concepts. *DigLC2* contributed to our teaching experience in the following ways:

- to understand the detailed sub-cycle behaviour of a realistic 16-bit processor;
- to experiment custom processor components in the context of a whole processor;
- to compare multiple data-flow and control models;
- to execute sample LC-2 programs, displaying processing stages from instruction-fetch to write-back;
- to play with basic input/output and interrupt mechanisms (they were not supported in the functional simulator [4]).
- to understand simple operating systems concepts;
- to extend the processor with hardware devices and off-chip controlers;
- to design and implement architecture enhancements for performance.

We followed the bottom-up approach advocated by Patt and Patel: students have been directly involved in the design of each processor component exploring multiple design issues. They achieved a finer understanding of the data-path and control structures, with a broader view of processor and system construction. Based on these fundamental concepts, the course diverted towards high-performance designs, program optimization techniques, and the forseeable future of micro-architectures.

The students were already familiar with C, object-oriented and functional programming (OCaml) on one side, and analog electrical engineering on the other, but they had no experience in digital systems. Our intent was neither to bridge the gap between assembly and high-level languages nor to describe the mapping of ideal transistors to silicon wafers — both topics being taught in the following semesters. We focused instead at the intermediate levels of the design, demystifying the building blocks of a microprocessor: from gates to combinatorial and sequential logics to data-paths and microprogrammed control to the instruction set architecture to assembler programming [6].

2 Technical Overview

The LC-2 system [3, 5] comprises a simple 16-bit microprocessor and a basic terminal emulation hardware. The instruction set is load/store[1] with 8 registers and 3 operands; it appears as a tradeoff between control-friendly and education-friendly features. The data-path is based on a 16-bit bus to connect almost all components *and* to communicate outside of the chip. Control is microprogrammed (fifty 39-bits wide microinstructions) and relies on a dedicated microsequencer for fast instruction selection and compaction. The LC-2 instruction set is very sketchy but supports a universal machine (e.g., no subtract, no OR operator, no shift...), forgetting about efficiency considerations. In comparison, system and device interaction is rather realistic and complete for such an educational architecture: both polling and interrupt-driven mechanisms are supported, and system calls (TRAPs) are clearly distinct from subroutine calls (yet the system does not address memory protection and address translation). Thanks to the original and efficient teaching model proposed by Patt and Patel, more and more introductory architecture courses are being build on the LC-2; the clean educational design of this processor is obviously a major incentive to do so.

The *DigLC2* simulator is free software (GPL), available online at
http://www-rocq.inria.fr/~acohen/teach/diglc2.html.
It is fully reusable, adaptable, and ready to use. Installation and usage documentation is available. The user should be familiar with the LC-2 specification, signal names and processor structures, as defined in Patt and Patel's textbook [3] (along with its appendices). *DigLC2* still lacks a technical manual, but the circuit is simple and most of the design is a straightforward implementation of the LC-2 specification. It runs over *DigLog*, Chipmunk's digital circuit simulator (GPL) [1]. We implemented the complete LC-2 architecture, including I/O terminal-emulation devices, interrupt vectors and memory (with customizable latency). Except for the SRAM memory chips and terminal device, every component of the LC-2 is built of elementary gates. The data-path and microsequencer are identical to the LC-2 specification.

[1]Plus indirect load and store operations — for programming convenience — that we personally would not have provided and that we intentionally avoided in the course and application exercises.

We rewrote the microprogram from scratch — see the *DigLC2* documentation — and applied large-scale tests on sample codes and student projects. The "boot-time" memory structure (vector table, operating system, boot ROM and memory-mapped I/O) is almost identical to the functional simulator's model [5], except that the initial PC is 0x0000 and that some I/O routines have been optimized.

Concerning I/O operations, the LC-2 description is not complete and we had to make a few implementation choices: the interrupt vectors for keyboard input and CRT output (0x0010 and 0x0011, respectively) and the detailed implementation of I/O registers (interrupt control bits, strobe signals, device operation latency).

Figure 1 shows the control panel of the LC-2 simulator. It displays every addressable and internal register, the full microinstruction, and many other signals. It also provides keyboard and screen emulations (standard *DigLog* components) for interactive terminal operations.

As one may expect, performance is much lower than Postiff's functional simulator: approximately 20 cycles per second on a $500\,MHz$ pentium III (interactive run, maximum details displayed): gate-level simulation of big programs is not realistic. However, we found these performances quite reasonable for the educational purposes of the LC-2 architecture:

- target codes implement short-lived classroom algorithms, toy programs and simple I/O operations;
- the most tedious part is linked with string processing and printing, e.g., the full CRT synchronization protocol proposed by Patt and Patel leads to a very slow implementation; still, choosing pragmatic parameters (short strings) and optimizing the code of display-oriented subroutines is usually satisfactory;
- in many cases, the user may even want to watch the real-time execution of the program, looking for errors in the assembly code, in a processor component, or in some custom additional circuit.

Eventually, we found only two architecture faults during circuit implementation: the first one is about choosing latches or flipflops and has been (arguably) corrected in recent online errata, the second one is a tricky page/PC-incrementation bug in conditional branch instructions. Considering the overall design, the detailed implementation choices and our teaching experience, we believe that the LC-2 architecture is a significant progress over previous educational systems; but we also hope that feed-back from professors and students around the world will be taken into account in future versions of the Little Computer and contribute to further improvements.

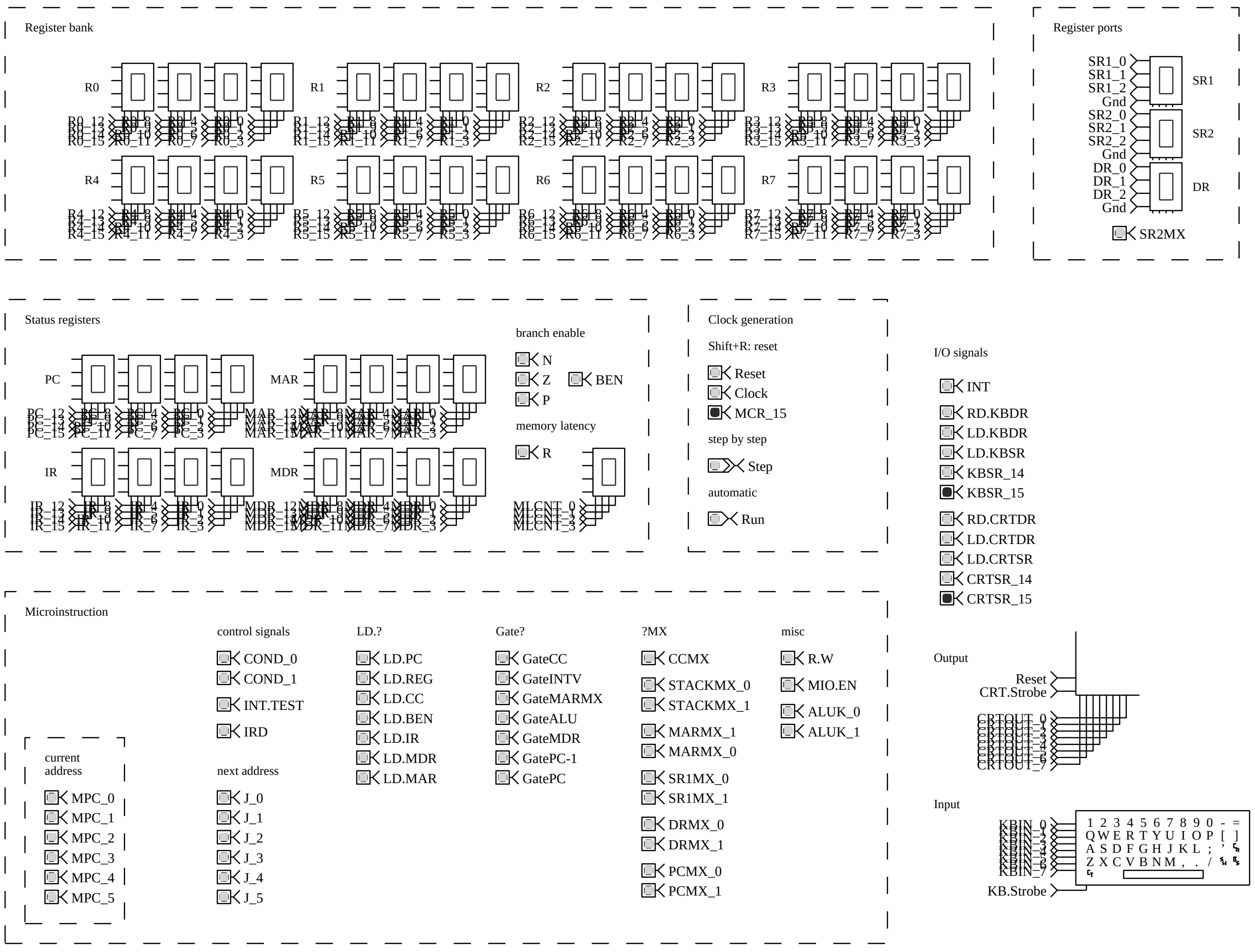

Figure 1: *DigLC2* control panel.

3 Student Projects

Three application projects have been proposed based on this digital simulator.

- Pipelining the LC-2, with a simple hazard detection and branch prediction mechanism. One student implemented a prototype version of a pipelined LC-2 (hardwired control, no indirect memory instructions). As a side-effect, simulator performance was significantly improved...

- Implementing a DMA controller for video output and experimenting a few bus protocols. This kind of extension is greatly simplified by the modular structuire of *DigLC2*. For example, every memory control signal has a LC-2 side and a SRAM-chip side, and the LC-2 is designed to cope with an arbitrary/unknown memory latency.

- Adding an instruction cache and/or a data cache to the LC-2; trying various associativity and replacement policies.

We believe that many existing student projects could benefit from *DigLC2*, focusing on the most interesting part of the project without the overhead of building a full processor or the complexity of a real-world processor. It can also be used to investigate the detailed implementation of processor performance enhancements — such as pipelining, superscalar and out-of-order execution — in the context of interrupts, and to interact with an existing assembler and legacy source code.

4 Conclusion and Future Work

DigLC2 is an interesting compromise between high-level structural modeling of digital circuits and expensive hardware test-beds. It is a useful tool for architecture courses, practical lectures, student projects and tutorials.

DigLC2 is an intuitive and modular implementation of the complete LC-2 system; it does not intend to be a fully realistic view of the actual silicon mapping, but provides a full gate-level simulation. By combining the bottom-up approach with *DigLC2* within the course and classes, students were able to progressively build their own full processor, using components they themselves designed session after session, and then they were able to visualize the execution of simple assembly programs at the gate-level.

Still, we would like to emphasize on the preliminary nature of this work. We believe that the tool might become even more beneficial if provided with multiple alternative implementations of each component, variations on the instruction-set architecture, and performance enhancements. We are not acquainted with processor verification techniques and did not address the testing and/or formal validation issues. Thanks to the wide distribution of Patt and Patel's textbook, we strongly encourage a community effort to contribute to the *DigLC2* project, as well to the underlying *DigLog* simulator [1].

References

[1] The Chipmunk system (specifically *DigLog*, the digital part of the *Log* simulator). Available online at http://www.cs.berkeley.edu/~lazzaro/chipmunk.

[2] A. Cohen. DigLC2: a gate-level simulator for the little computer 2. Available online at http://www-rocq.inria.fr/~acohen/teach/diglc2.html.

[3] Y. N. Patt and S. J. Patel. *Introduction to computing systems: from bits & gates to C & beyond*. McGraw-Hill, 2001. http://www.mhhe.com/engcs/compsci/patt.

[4] M. Postiff. LC-2 simulator (and assembler). Available online at http://www.mhhe.com/engcs/compsci/patt/lc2unix.mhtml.

[5] M. Postiff. *LC-2 Programmer's Reference and User Guide*. University of Michigan (EECS 100), 1999. http://www.mhhe.com/engcs/compsci/patt/lc2labstud.mhtml.

[6] O. Temam and A. Cohen. Cours d'architecture. École Polytechnique 3$^{\text{ème}}$ année majeure 1, 2001–2002. http://www.lri.fr/~temam/X/index.html (in French).