# CPU Design Kit: An Instructional Prototyping Platform for Teaching Processor Design

**Anujan Varma, Lampros Kalampoukas, Dimitrios Stiliadis, and Quinn Jacobson**

**Computer Engineering Department**
**University of California**
**Santa Cruz, CA 95064**
E-mail: varma@cse.ucsc.edu
World-Wide Web: http://www.cse.ucsc.edu/research/hsnlab/projects/CPUkit.html

*Abstract*

The CPU Design Kit is a prototyping platform designed at University of California, Santa Cruz, for teaching the Processor Design class. The prototyping platform allows the design and implementation of a 32-bit pipelined CPU. The prototyping hardware consists of an ISA-bus-based printed-circuit board containing six Altera FLEX EPF81500 programmable logic chips providing a total of 80,000 usable gates, static RAM chips to implement register file and caches, and hardware support for monitoring and debugging. A Windows-based user interface provides access to the necessary software tools for downloading designs into the board, debugging, and control of the CPU.

## I. Introduction

Many universities now offer a course on *processor design* or *computer design* as part of the computer engineering curriculum. The objective of this course is to teach the fundamental concepts involved in the design of a Central Processing Unit (CPU) by means of a hands-on project. Typically, the project involves the design, fabrication, and testing of a small CPU within a semester- or quarter-long class. Until a few years ago, these CPU designs were prototyped using bit-slice ALU chips, in combination with microprogrammed control units. This approach limited the scope and flexibility of the project; key concepts such as pipelining, common among modern RISC processors, could not be explored.

With the availability of dense programmable logic chips such as the Altera FLEX series, it has now become possible to construct more complex designs reflecting the state-of-the-art in CPU design within a short time. Because of the complexity of the chips, however, it is still difficult to prototype and debug a CPU design employing multiple FPGA chips within the classroom environment. The CPU Design Kit is a prototyping platform, consisting of both hardware and software, designed to facilitate the construction of such projects within a limited period of time, such as an academic quarter or semester.

The CPU Design Kit is designed to allow implementation of a pipelined 32-bit processor with minimal effort on fabrication and testing. The prototyping hardware consists of an ISA-bus-based printed-circuit board containing six Altera FLEX EPF81500 programmable logic chips, static RAM chips to implement register file and caches, and hardware support for monitoring and debugging. Software available with the platform include tools to load compiled designs into the FLEX devices, set and display contents of the register file and caches, monitor the state of signals on the board, and control execution of the CPU.

## II. Board Architecture and Hardware

Figure 1 shows the devices on the prototyping board and Figure 2 illustrates the data and control paths on the board while implementing a 32-bit pipelined processor similar to the DLX described by Hennessy and Patterson [1]. The hardware consists of five Altera FLEX EPF81500 chips, each intended for implementing specific functional blocks of the CPU design. A sixth FLEX device is used to implement the ISA bus interface of the board to the host PC and to control and monitor the execution of a program on the prototype CPU.

Each of the FLEX devices provides approximately 16,000 usable gates and 200 I/O pins. Thus, the board provides a total of 80,000 usable gates for implementing the CPU design. The architecture of the board has been designed to allow implementation of a 32-bit pipelined RISC CPU similar to the DLX, but the board may also be used for other styles of instruction sets.

The functional blocks of the CPU design are intended to be partitioned among the five FLEX devices as follows:

1. *Instruction Fetch, Decode, and Branch Unit:* These functions are implemented by FPGA-1 in Figure 2. This module maintains the program counter, handles fetching of instructions from the instruction cache, and implements conditional branches and jumps.
2. *Arithmetic Logic Unit:* The ALU module is implemented by two FLEX devices, FPGAs 2 and 3 in Figure 2. FPGA-2 is meant to be the primary ALU chip, with FPGA-3 providing additional logic for implementation of logic-intensive functions such as a barrel shifter.
3. *Data Memory Interface:* FPGA-5 provides the interface to the data cache, and allows routing of data between the ALU/register file and the data cache.
4. *Control/Forwarding Unit:* The logic needed for forwarding and control of the pipeline is intended to be placed in FPGA-4.

Program execution and debugging of the CPU are controlled by logic within the sixth FPGA, which also implements the ISA bus interface. This FPGA can be programmed to control execution of programs by single-stepping, setting up breakpoints, etc. This FPGA also generates the clocks that control all system timing from a 40 MHz master clock fed to it. The clocks are distributed to all the FPGAs via four dedicated lines.

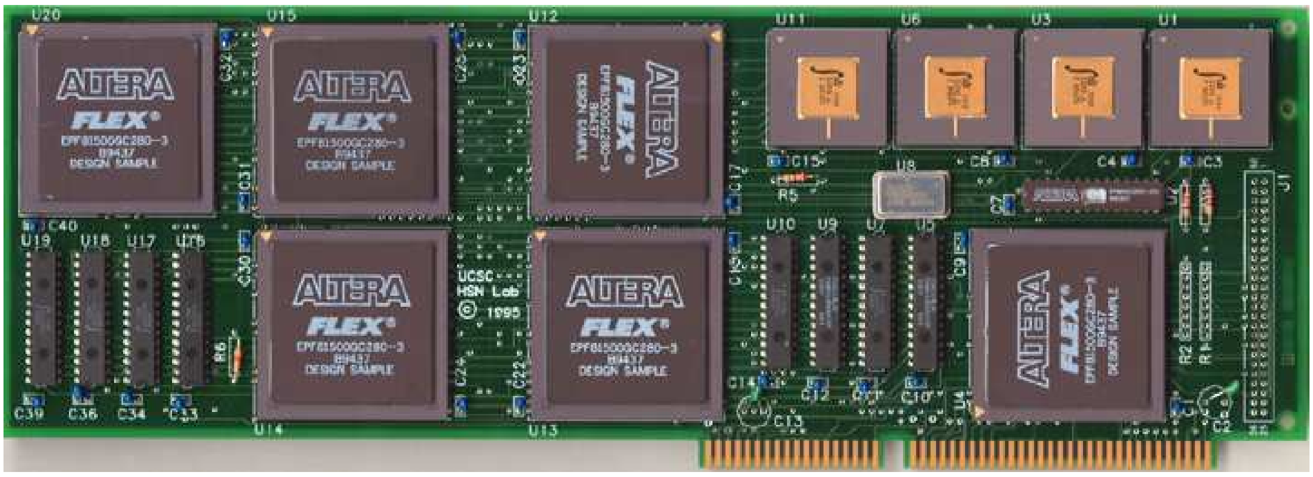


Fig. 1. The CPU Design Kit prototyping board.

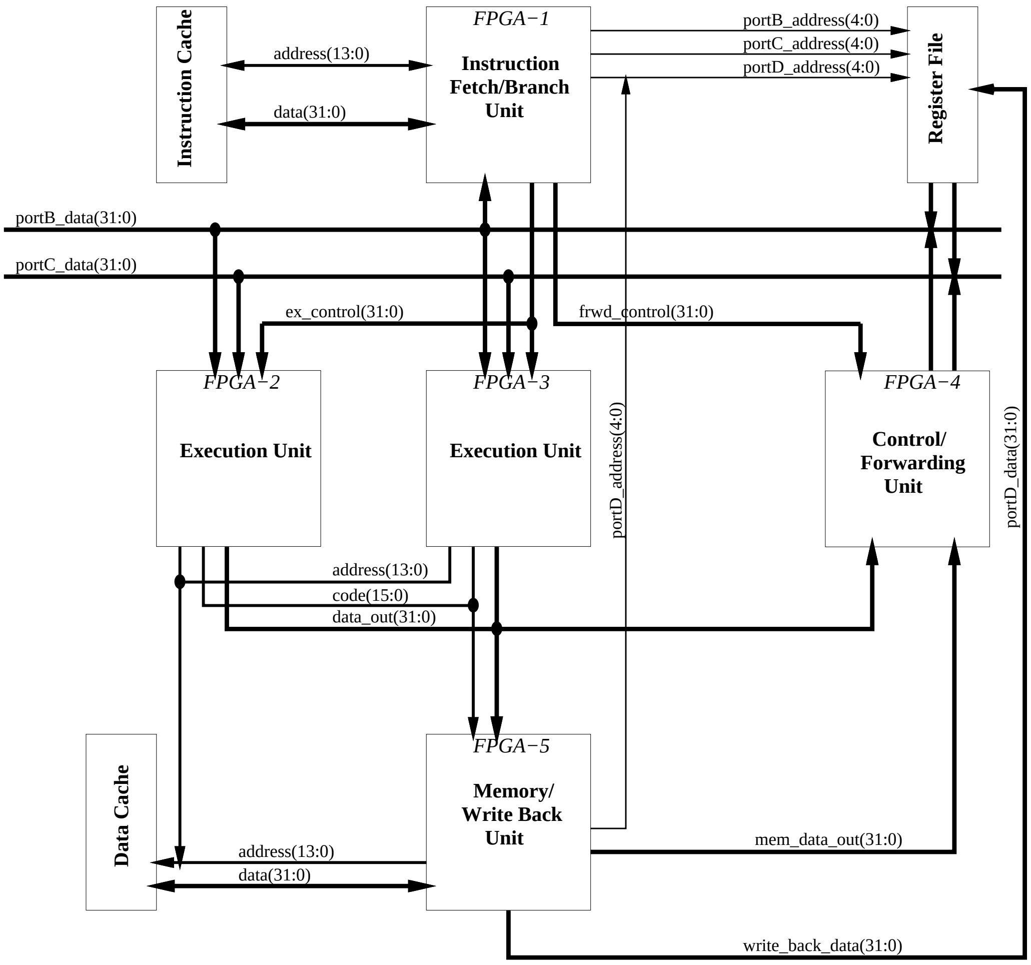
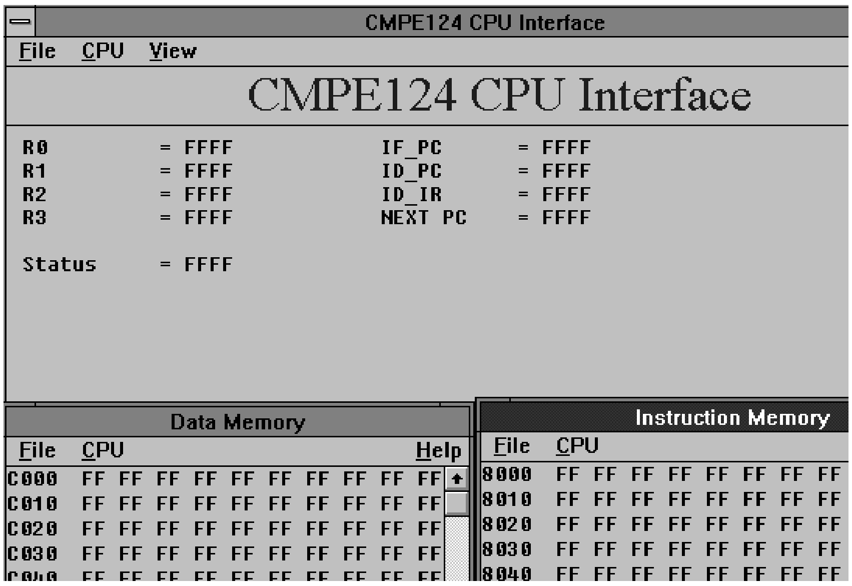


Fig. 2. Block diagram of the prototyping board. The five FPGAs shown are used to implement the CPU design. The sixth FPGA, used to implement the ISA-bus interface, as well as control and debugging of the CPU design, is not shown.



```
CMPE124 CPU Interface

R0        = FFFF          IF_PC      = FFFF
R1        = FFFF          ID_PC      = FFFF
R2        = FFFF          ID_IR      = FFFF
R3        = FFFF          NEXT PC    = FFFF

Status    = FFFF
```

```
C000  FF FF FF FF FF FF FF FF FF FF
C010  FF FF FF FF FF FF FF FF FF FF
C020  FF FF FF FF FF FF FF FF FF FF
C030  FF FF FF FF FF FF FF FF FF FF
C040  FF FF FF FF FF FF FF FF FF FF
```

```
8000  FF FF FF FF FF FF FF FF
8010  FF FF FF FF FF FF FF FF
8020  FF FF FF FF FF FF FF FF
8030  FF FF FF FF FF FF FF FF
8040  FF FF FF FF FF FF FF FF
```

Fig. 3. Windows-based user interface.

In addition to the FPGAs, the board also contains a register file. The register file is a 4-port 4K $\times$32-bit static RAM module. Three of the four ports of the register file are connected to the ALU; in a three-address register architecture such as the DLX, two of the ports are used as read ports and the third as a write port. The unused fourth port is used by the system monitor software to access the register file so that register contents can be set and displayed during debugging.

The memory system of the board consists of separate instruction and data caches. Each cache has a capacity of 16K words (64K bytes). The instruction cache is directly connected to FPGA-1 implementing the instruction-fetch function, and the data cache is interfaced to the ALU/register-file via FPGA-5. The address for data cache is driven by the ALU output; data may be stored into the cache from any of the two register ports feeding the ALU, or loaded from the cache into the third (write) port of the register file.

In addition to the logic resources needed to prototype the CPU, the board also contains hardware to facilitate debugging and benchmarking. The sixth EPF81500 FLEX device forms the core of the monitoring hardware and provides the following facilities:

1. Facility to access internal registers and signals within the FPGAs through a dedicated debug bus. The user may define accessible registers within the design for debugging, which can then be monitored or set via the debug bus. Software tools allow symbolic names to be attached to these registers for convenience.
2. Facility to allow monitoring the state of signals on the board from the PC. This is achieved by implementing a multiplexing function that receives some of the on-board signals on its input and multiplexes them to the ISA-bus interface. This allows “probing” all signals on the board during debugging without accessing them physically.
3. Monitoring of logic events, similar to a logic analyzer, is also achieved by programming the FLEX device to watch for the signal states of interest. Events can be signaled to the interface FPGA by any of the other devices via the global bus.
4. Performance monitoring can also be implemented within this FPGA by implementing counters to keep track of performance metrics such as total cycles of program execution, number of pipeline stall cycles, etc.

## III. Software Tools

Using the CPU Design Kit to prototype a CPU design efficiently requires a variety of software tools. As part of this project, we are currently developing software to provide a complete environment for processor design. Altera’s MaxPlus tools already provide an intergated environment for design entry, synthesis, and simulation of the system from a high-level language description of the target hardware. We are supplementing them with tools that are specific to the CPU Design Kit environment. The following are the key tools currently under development:

1. Tools to download the compiled designs into the FLEX devices on the board.
2. Tools to control clocking — enable single-stepping, breakpoint execution, etc.
3. Tools to monitor on-board signals and set up event monitors. This provides a logic analyzer-like function. Software will allow attaching labels to signals and displaying them in a variety of formats.
4. Software to read and modify the register file, instruction cache, and data cache.
5. Performance-monitoring tools to set up counters and generate statistics of interest to the user.
6. A Meta-Assembler that allows generation of code for custom instruction sets with minimal effort. Many public-domain tools are already available for this purpose.

The user-interface of the software is currently based on Microsoft Windows. Figure 3 shows a snapshot of the user interface being developed. The tools may be ported to the Unix/X Windows platform in the future.

## IV. Course Organization

Because of the tight time constraints in the quarter system at UCSC, we have found it important to organize the project so that students can complete the design task in a sequence of well-defined stages. The CPU Design Kit allows the processor design project to be carried out in phases, each phase implementing a part of the design. For example, assuming a 5-stage pipeline as in the example implementation of Hennessy and Patterson [1], the entire design of the pipelined CPU can be structured into four phases as follows: The instruction fetch and branch logic is implemented and tested in the first phase. This requires design and programming of only one of the five FPGAs. Only the first two stages of the pipeline are implemented in this phase. The second phase adds the ALU and register file to the design and expands the pipeline to five stages. All the ALU instructions are implemented and tested in this phase. Note that all the data hazards introduced by the pipeline are ignored in this phase. Phase 3 involves adding the data memory interface to the design so that load and store instructions can be tested. Finally, the logic necessary to implement all the forwarding requirements is incorporated in Phase 4, thus completing the design. We have found that this organization allows students to accomplish the entire design and implementation within the limited time of one academic quarter.

## V. Conclusions and Status

The CPU Design Kit was first introduced in the Processor Design class at UCSC in the Fall of 1995, where it was used to design a complete 5-stage pipelined RISC processor similar to the DLX as described by Hennessy and Patterson [1]. The entire instruction set of DLX was implemented, except for instructions dealing with floating point and exceptions. The logic needed for the design was well below the capacity of the FPGAs. Thus, the board leaves room for attempting substantially more complex designs in the future.

We are currently working on the next generation of CPU Design Kit, based on Altera's FLEX 10K family. This family includes single-chip devices with up to 100,000 usable gates and clock speeds as high as 70 MHz, allowing much more complex CPU designs to be attempted. An additional advantage of the FLEX 10K family is the availability of on-chip SRAM, thus allowing multiport register files to be implemented on-chip.

## References

[1] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach," Morgan Kaufmann, 1995.

[2] Altera Corporation, FLEX 8000 Handbook, 1995.