Teaching Computer Architecture with FPGA Soft Processors

***Dr. Andrew Strelzoff*[1]**

Abstract – Computer Architecture has traditionally been taught to Computer Science students using simulation. Students develop and test small scale processors using a hardware description language [HDL]. Building a processor in simulation can be interesting and instructive. However, the HDL experience is generally not useful for other Computer Science classes or senior projects. Complex circuits built entirely in simulation usually do not function properly if translated to real hardware. A great deal of additional effort and a much deeper understanding of transistor level and gate level hardware are required to develop a functioning integrated circuit purely from simulation. Simulated processors also lack high level language compilers, loaders or linkers. Testing is usually done with hand-written assembly code. This is reasonable for Computer Architecture but makes the simulated processor unsuitable as an embedded processor. The ideal situation would be a way that students could study Computer Architecture with processors which would actually be useful for future projects.

Field Programmable Gate Arrays [FPGA] are integrated circuits which are imprinted with arrays of logic units connected with switches. By programming the switches a logical circuit is formed. FPGA were originally developed to prototype other types of printed circuits, but in recent years these devices have developed very rapidly in capabilities and capacity. Prices for FPGA mounted on supporting boards have fallen dramatically. In addition, a wide variety of open source soft processors have become available.

A soft processor is a HDL description of a processor which may be implemented on a variety of different FPGA. For students this offers the opportunity to use real functioning computer hardware with processors that range from simple micro-controllers to complex multi-threaded superscalar machines. Compilers are available which translate various high level languages to the assembly languages of each processor. In addition, a wide variety of peripheral cores such as interface protocols and common arithmetic tasks are available and may be easily integrated into a student's design. Students using FPGA rather than simulation enjoy using actual computer hardware and they gain a valuable skill which is applicable to wide variety of other student projects.

This article discusses the use of FPGA with soft processors to teach Computer Architecture. The use of FPGA in the classroom, the various methods for programming the FPGA, and the development of a workbook and curriculum which features the Altium LiveDesign FPGA kit and the TSK family of soft processors are discussed.

Keywords: Computer Science Education, FPGA, Soft Processors

---

[1] Department of Computer Science, University of Southern Mississippi , 118 College Blvd. Hattiesburg MS. 39406, and strelz@orca.st.usm.edu

INTRODUCTION

Computer Architecture classes have primarily been taught using HDL simulation. The advantage of this approach is that simulation takes place on a PC without requiring specialized hardware. The primary disadvantage is that although building a processor in simulation is interesting and instructive students are developing a skill which they cannot easily apply to other classes or projects. Hardware built purely in simulation usually does not usually function properly if implemented in actual circuitry. In addition, most students do not build compilers for their simulated processors. Compiler design is well beyond the scope of a Computer Architecture class and most students test their simulated processors with hand written assembly code. As a result processor designs which require entire semester to complete are abandoned. The ideal situation would be for students to study real working processors which would then be useful for a variety of further educational purposes.

FIELD PROGRAMMABLE GATE ARRAYS (FPGA)

FPGA were originally developed to prototype integrated circuits but have become both much less expensive and much more capable in the last few years. An FPGA, as shown in Figure 1, is an integrated circuit imprinted with thousands of logic cells or "gate arrays" which are "field programmable". Each logic cell is basically a multiplexer attached to the rest of the array through programmable switches. Pathways through the gate array are set by the bit pattern of the switches. This bit pattern is created from a schematic or HDL specification by a device-specific optimizer which seeks to find a bit pattern which implements the desired logical circuit as efficiently as possible. The design process for an FPGA is similar to that for a hardware simulation environment except that real functioning hardware circuits are created rather than simulation tracings.

The Altium LiveDesign evaluation board is a $99 kit containing a board with either a Xilinx or Altera FPGA with around 12,000 logical units. The LiveDesign software provides an integrated development environment with a schematic design editor. In schematic design logical elements such as gates and more complex structures such as counters and arithmetic operators are placed in a graphical environment and connected by dragging wire or bus lines. Computer Science students who generally have minimal hardware backgrounds usually find schematic design more intuitive than pure HDL code. A simple schematic design in the LiveDesign environment is shown in figure 2.

In addition to combinatorial and sequential logic the LiveDesign library of elements includes several soft processors. This allows students to drag and drop an entire processor into their schematic diagram. The example shown in figure 3 features a TSK51, which is functionally equivalent to an 8051 microcontroller, and requires about 12 percent of board resources. This microcontroller is easily programmed through LiveDesign by adding and compiling C code using the built in Tasker compiler. A sample of this code is shown in figure 4. Processor ports may be referenced in the embedded code by using their schematic designations. Thus, the example uses the variable 'B1" which is the same name as a wire attached to the board push button. This greatly simplifies the design of processor I/O.

Another, extremely useful feature is virtual instrumentation. The LiveDesign object library includes several virtual instruments which may be added to a schematic design, as shown in figure 5. When the schematic design is loaded onto the FPGA the virtual instrument pops up in the LiveDesign control environment. The instrument panels may be used to send input to the schematic design or may be used to monitor the activity of the board while it is running. The LiveDesign environment was developed for professional embedded device development with FPGA. In order to make this technology accessible to students a workbook is being developed.

COMPUTER ARCHITECTURE WITH FPGA WORKBOOK

The Prentice Hall FPGA-Based Computer Architecture Workbook (to Appear Sept. 2007) will be bundled with an Altium LiveDesign Kit board which hosts an Altera Cyclone II FPGA. The workbook has the following practical and educational goals:

Quick Start: Students need to learn the basic functionality of the environment quickly so that they can move on to more serious exercises. This is accomplished through a series of basic combinatorial and sequential logic exercises leading up to an exercise which guides students through several steps to build a basic simple 8-bit 3-phase processor with a simplified instruction set. A completed section of this practice processor is shown in figure 6.

Introduction to embedded processing: Students are then introduced to the steps needed to build a schematic design with an embedded processor. A sample of an embedded processor design featuring the TSK51 is shown in figure 3. Students can then compare the performance of sample C code for various processors and see the effect that various processor design features such as pipelining have on actual performance.

Extending processor designs: For some subjects, such as Reduced Instruction Set architecture [RISC], there are several processors available in the Altium environment for students to study. For other subjects Altium or Open Source [4] processors are not available because they are currently out of vogue (Complex Instruction Set architecture [CISC] for example) or are too commercially valuable (Super-Scalar for example). This is solved by having students build versions of these architectures using other processors as building blocks. For example, to approach a fundamental problem in Computer Architecture "CISC versus RISC" students need a CISC processor. The Altium processor sets include several RISC machines including the TSK80 which is analogous to an Intel 8080 processor. Students build a CISC processor by adding a TSK51 coprocessor to the TSK80. The coprocessor is programmed with one or more computational intensive tasks such as random number generation. Students can generate two versions the TSK80 assembly code, one using the normal RISC approach and one using a single powerful assembly instruction to perform a complex task, the essence of CISC.

Processor internals: Pipelining is one of the most important topics in Computer Architecture but is a difficult and unintuitive subject for students. Pipeline architecture is studied by using LiveDesign virtual instrumentation to examine various pipeline events such as stalls and hazards. In addition exercises lead students to delve in to the HDL code underlying the schematic processor blocks. This code is quite complex but very small changes can be made such as commenting out a hazard detecting condition. Students can than watch the effect on actual processor operation illuminating the function and effect of each policy.

The processor as part of a central processing unit (CPU): The processor is served by a variety of devices such as memory, bus controllers and I/O devices. The same approach as in "CISC vs. RISC" above is used to develop exercises which illustrate the workings of these devices. For example, a TSK51 is used as a memory and cache controller mediating internal FPGA memory (essentially banks of un-used switches), external on-board RAM and through the serial port data stored on the hard drive of the host computer. Students can than explore various caching paradigms using an actual working system with memory timings analogous to a real-world CPU.

Modern processors: One of the drawbacks of simulation and one of the weakest areas in the current state of Computer Architecture class materials is that most current processor issues are examined in only a cursory manner and only at a very high abstract level. An example is Super Scalar architecture. Most machine since the mid-90's are Super Scalar, meaning they can issue instructions to multiple resources such as ALUs, yet most architecture classes skim over the topic because such machines are too complex to be built in simulation in a single semester. Using simpler processors as building blocks several more advanced processors can be built. For example, students build a small parallel video card with four TSK51s. This composite processor can execute single instruction multiple data instructions (SIMD) driving output to the Altium board's VGA output port. Time Sharing, Superscalar and Super Multi-Threading (SMT) CPUs are all built by students with the same approach.

FPGA-based Computer Architecture lab exercises provide Computer Science students with an interesting and educational experience. In addition students learn to implement embedded processors in FPGA, a skill which will be useful in other classes and in their work life.

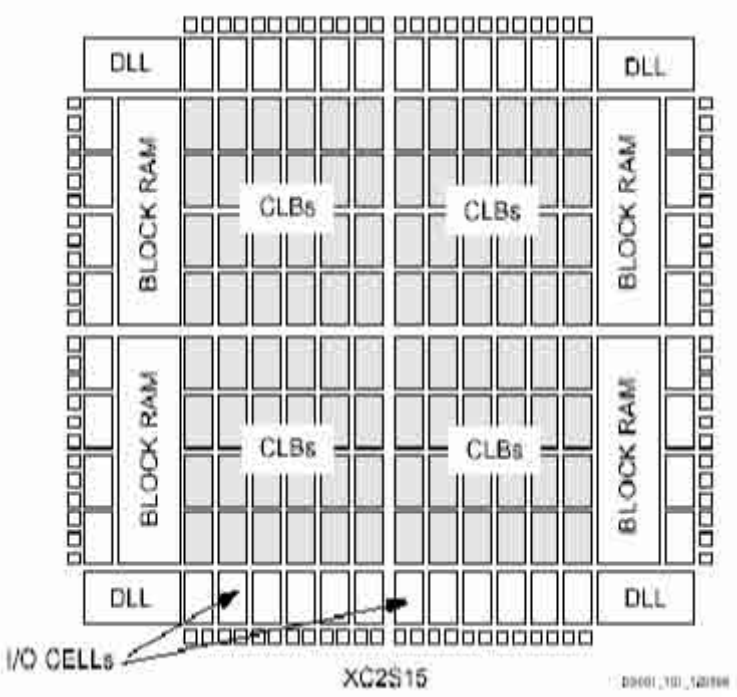

Figure 1 – Field Gate Programmable Array (FPGA) fabric showing banks of logical units connected with switches and surrounded by I/O cells [2].

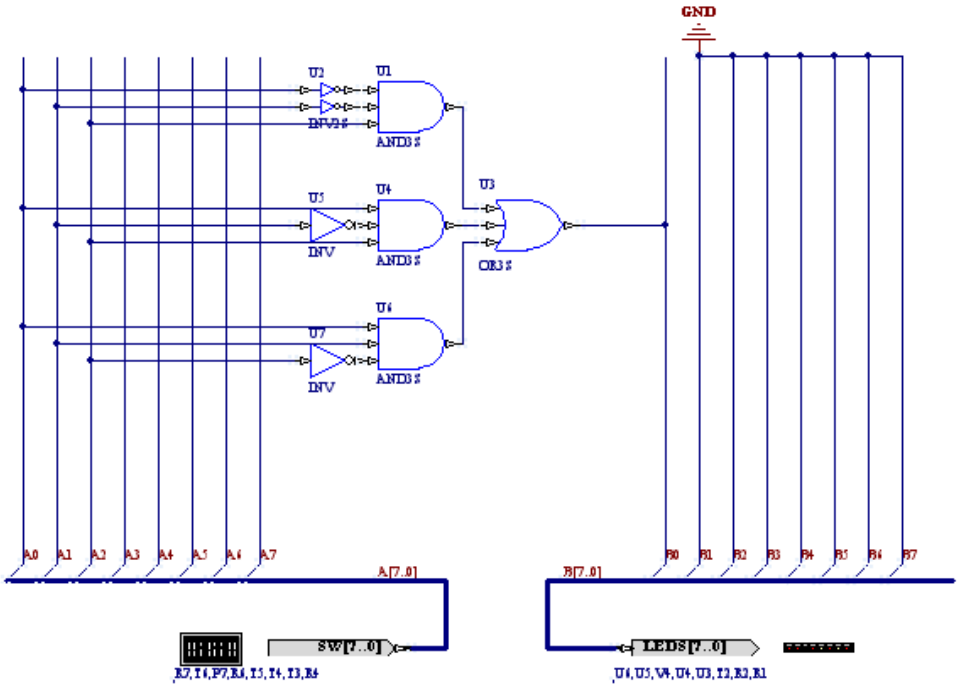

Figure 2 – A simple schematic logical design in the Altium LiveDesign [3] environment.

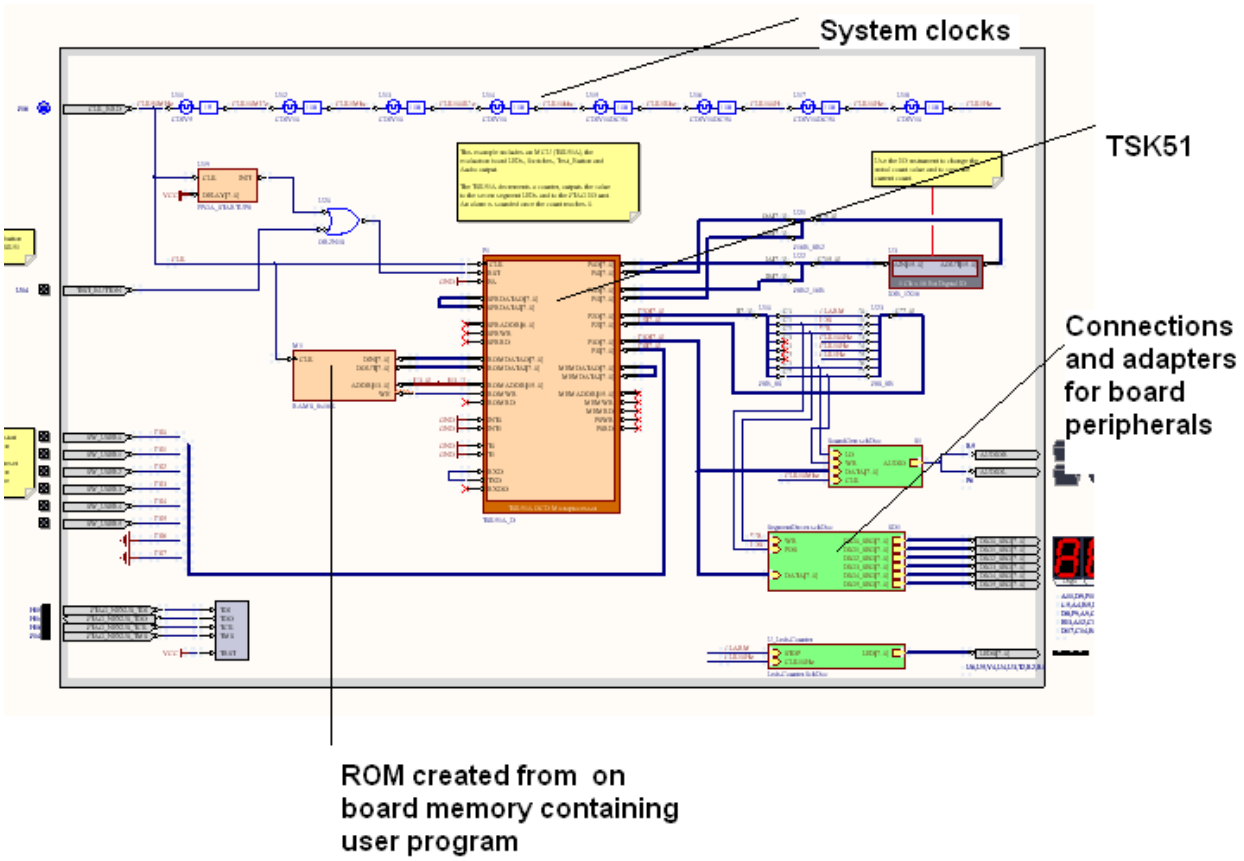

Figure 3 – A simple embedded design using TSK51 in the Altium LiveDesign environment.

```
//-----------------------------------------
// Read start count value from JTAG I/O
//-----------------------------------------

unsigned int getStartCount(void)
{
   const unsigned int DEFAULT_COUNT = 20;
   unsigned int count;

   //Read count high and low value from ports
   count = (P1<<8)+P0;

   if (count==0)
      count = DEFAULT_COUNT;
   return count;
}
```

Figure 4 – A section of embedded C code in the Altium LiveDesign environment. P0 and P1 are ports on the TSK51.

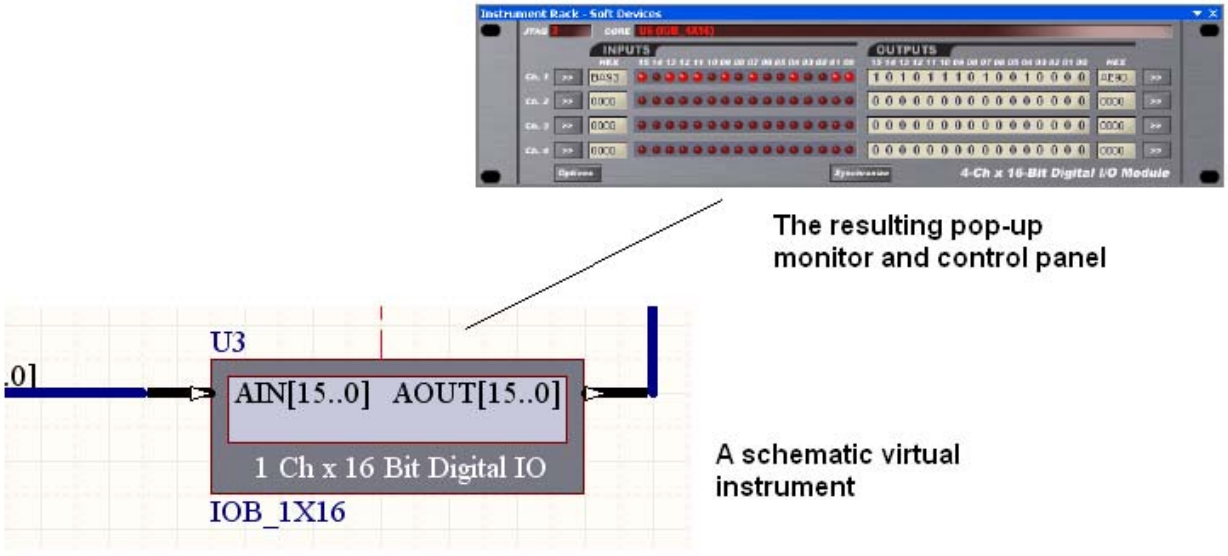

Figure 5 – An Altium LiveDesign virtual instrument. Traffic on the bus shown can be viewed in real time through a virtual instrument panel.

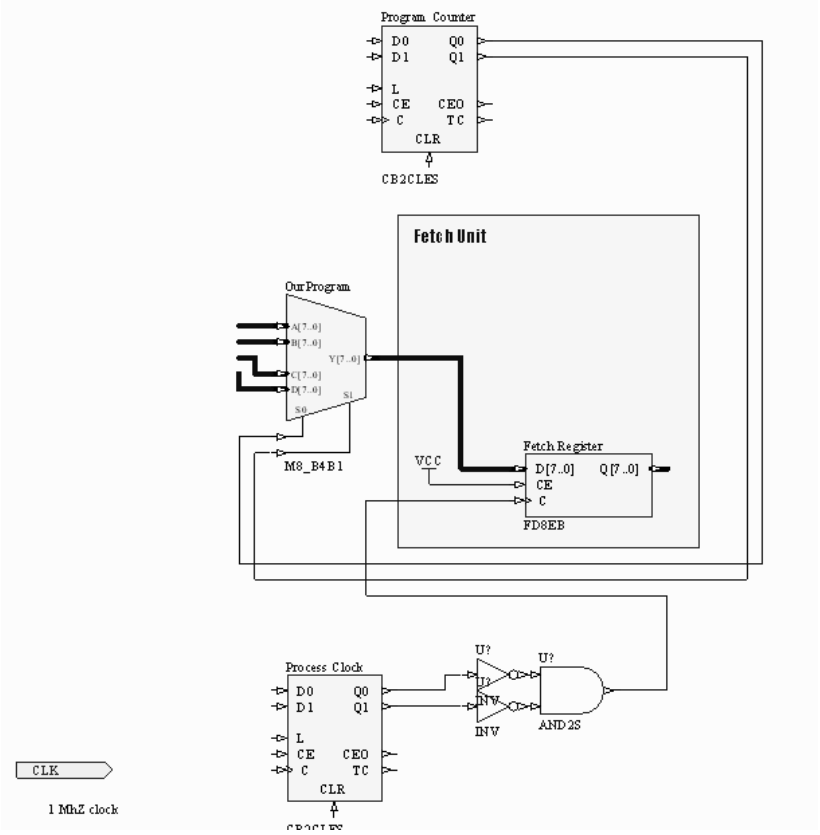

Figure 6 – An early student exercise in the Prentice Hall FPGA-based Computer Architecture workbook

REFERENCES

[1] Mentor Graphics, **ModelSim**, www.model.com.
[2] Xilinx, **Spartan II users manual**, www.xilinx.com.
[3] Altium, LiveDesign , www.altium.com
[4] Open Cores repository, open.cores.org

Dr. Andrew Strelzoff
2004 Ph.D. Computer Science, University of California, Santa Barbara
2004 Assistant Professor, Computer Science, University of Southern Mississippi
Areas of Research: FPGA computing, Radio Frequency Identification (RFID), Condition Based Maintenance (CBM)