The Development of a Course in Programmable Digital Devices

Rosida Coowar

University of Central Florida

Abstract

In industry the requirement of short-time-to-market is playing an increasingly important role in product design. In the case of electronic equipment and devices this can be achieved by the use of programmable devices. It is therefore important for students, upon entry into the industry, to be conversant with the various architectures, development tools, trade offs and limitations of these devices. To this end a course in programmable digital devices (PDDs), a senior technical elective for electrical engineering technology majors was developed by the author at the University of Central Florida.

1. Introduction

The course in Programmable Digital Devices covers devices from simple PALs™ (programmable array logic)/GALs™ structures to CPLDs(complex PLDs) and FPGAs(field programmable gate arrays). The course outline is found in Appendix A. The architecture of the most commonly used chips in each category is covered so that the student can have a good understanding of the resources available inside of those chips, their limitations, and the need for a more complex architecture as the design to be implemented grows in complexity.

The prerequisites to the PDD course are:(1) Digital Fundamentals

(2) Digital Systems

since the student enrolled in this course needs a good, solid background in the digital area.

Most of the students who join the electrical engineering technology program at UCF are transfer students and would already have taken the digital fundamentals course at a community college. Although the more recently appearing introductory logic texts include discussions on PLDs, the presently used majority still is TTL-based and deals at most with SSI and MSI logic implementation. It is also a fact of life that due to time constraints the really interesting items are usually barely discussed in class. To add yet another item to this already full task seems impractical, however, the low-end PLD devices can easily be incorporated in basic digital courses. Examples of the latter are PLAs (Programmable Logic Arrays) and PALs (Programmable Array Logic). The difference between these two types is minor so far as the user is concerned and for all practical purposes, the PLA can just be explained briefly and the PAL architecture, which for manufacturing convenience is the actually implemented structure, can be used from then on. PLDs(Programmable Logic Devices) come in a great variety of architectures, degrees of complexity and programming methods. The low-end devices can implement small amounts of "random" logic and require little in the way of programming tools. The high-end PLDs on the other hand, can implement complex logic or even small systems, but require sophisticated development tools, which have long learning curves if one needs to become familiar with all their options and capabilities. The use of PLDs also requires that a choice be made between reprogrammable and non-reprogrammable devices, and additionally for the former, whether in-system or externally programmable devices are preferable. The latter choice, however, often boils down to whether or not volatile or non-volatile devices are needed. The above mentioned differences are primarily important for deciding which devices should be used in the student laboratories.

II. The Digital Systems Course

In the digital systems course in the Electrical Engineering Technology curriculum at UCF, PALs are revisited. This discussion of the PALs can be very brief since their basic structure, just AND and OR-gates, should be familiar to the student very early in the introductory course. It is then only a small step to the low-end programmable device and at that point actual existing device architectures can be introduced. Although the programming of these devices is of such simplicity that it could actually be done by hand, a good paper exercise for a few examples incidentally, a programming tool is needed to be able to do the physical part of the programming. On that level the tools are simple and easy to learn, intuitive almost, so the learning curve is acceptable. There is actually a bonus to using these tools since they usually have a built-in simulator. Since the latter may show actual time delays it is valuable for the students to see real gate behavior which probably is in contrast to the ideal waveform and gate behavior that texts tend to show.

Since it is the usual practice to implement several digital functions in a single device, a brief discussion of minimization techniques, other than Karnaugh maps and the Quine-McClusky method, which seem to rate less and less attention in the curriculum anyway, is appropriate at this time. It is interesting to note that the design entry tends to be of the behavioral type rather than schematic capture, since it fits the tools much better. Most of them allow entry in the form of Boolean equations, truth-tables and state machines and the students should at that time in the course be quite familiar with at least the first two methods. The third one can be dealt with later.

An example of a very easy- to- use tool, available free from the manufacturer, is PLDShell from Intel (this branch of the business was recently sold to Altera and that company, rather than Intel, should be contacted for the tools and the devices). Fully functional PALs are also available in reasonable quantities either free of charge or for a nominal fee. The only “capital” outlay would be for the actual programmer, but any general programmer, such as from Data/IO, if already available, would be more than adequate in this case. The programmed device can then easily be incorporated in an overall design in a way not different from any other SS1 or MSI chip. Testing of the actual hardware can then be carried out in the usual manner and these results compared with the ones obtained from functional and/or full-timing testing.

The students are also introduced Workview from Viewlogic for schematic capture. During the course of the semester they are required to do a couple of projects , one using Workview for the schematic capture and simulation of their design before actually building it, and the other using the PALasm tools and actually programming a chip to implement the design.

By the time they take the PDD course they are already familiar with two development tools : one of the entry is of the behavioral type and the other one requires some type of schematic entry. At this point they are also aware of the different types of PLDs available on the market.

II. The Programmable Digital Devices Course

The simple PAL-like structures and devices covered before, need not be abandoned at this point, in fact they should now be considered simply another option for at least part of more complex designs. For larger circuits or those which, due to complexity, are beyond the capabilities of the PALs, the CPLD (Complex PLDs) or EPLD (Extended PLDs) types will be more appropriate. Structurally their architectures are just more elaborate and larger and thus it takes very little time to understand them. The tools are not much more elaborate either, so it should be a smooth transition to these more capable devices. The main difference is the inclusion on the device of some kind of interconnect matrix, essentially allowing the interconnecting of four or more PALS on the same chip instead of four individual PALs on a PCB (Printed Circuit Board). Conceptually this is inconsequential, but the increased versatility of this type of device is such that more powerful tools are needed to make efficient use of the device, particularly the interconnect matrix. The “place and route” issues which therefore crop up can, however, be totally ignored in the logic course, if considered inappropriate. It would suffice to point out that they form an integral part of these more elaborate structures and then leave it to the tools to take care of it.

The apparent loss of control over the actual implementation of the circuit may be disconcerting to the student designer, but it should be treated as an example of the inevitable trade-offs that constantly have to be made in design methodology. At this point, it becomes necessary to consider reprogrammable versus non-reprogrammable types of devices. A further distinction among the former is on-the-fly or in-system reprogrammability versus out-of-system reprogrammability. On-the-fly reprogrammability requires an SRAM (Static Ram) based architecture, which has as concomitant characteristic volatility, in other words, loss of power to the device means loss of function. This is probably the one overriding characteristic when it comes to deciding to use this type of device instead of a non-volatile type. It should be pointed out, however, that for the much more complex devices such as SRAM-based FPGAs (Field Programmable Gate Arrays, which will be mentioned later) there is an option for downloading from an on-board PROM, which eliminates that problem. The other major reason for choosing an SRAM-based device is that if there is a need to be able to do in-system reprogramming in such a short time that the IC's functionality is adaptable to system requirements on the fly.

In the student environment this characteristic is probably of no consequence. The non-SRAM based devices have programming times which are so many orders of magnitude larger, minutes versus mini-seconds, that once they are inserted in the system they have essentially just a single functionality. These devices are either UV-erasable or Electrically Alterable and require additional hardware for the programming and they must be removed from the circuit for reprogramming. The latter devices are reprogrammable but not volatile, i.e. power loss does not mean loss of function . To complicate the choice there are also the non-reprogrammable devices, which are of course non-volatile, but programmable only once. Typically these are fuse-link or anti-fuse™ devices. That may seem like a serious disadvantage, but it should be considered yet another trade-off, since these devices, due to their significantly lower interconnect resistance, can be used at higher clock speeds. This advantage is, however, probably of no importance in junior and senior courses and design projects, but it may be a very significant characteristic in a more demanding environment.

111. Tools and Development Systems

The devices discussed above are readily available from Xilinx (SRAM) and Altera and Actel (both non-volatile) by contacting their University Program Managers.. For instance, the SRAM-based devices, due to the space limitations on the chip have fewer interconnects and routing resources per logic entity than the fuse-link or anti-fuse™ based devices from Actel and Alters, Thus the Xilinx tools for instance, depend more heavily on the sophistication of the place-and-route routines to achieve acceptable delays in the programmed device, the more so if the device is heavily populated. To make optimum use of these devices the designer also must also work more interactively with the tools or at least be able to select judiciously from the rich set of options. This requires knowledge of the architecture of the various families of devices. Also the Xilinx tools themselves do not include any front-end directly, but several well-established and powerful front-ends, such as VHDL and schematic capture, such as available from ViewLogic, Orcad, Mentor Graphics, Cadence and Synopsys, can be easily integrated in the Xilinx tools, providing an almost seamless interface. These are very powerful and large tool-suites, which have in themselves long learning curves. So, unless they have been introduced and used in other courses and have become a staple diet for the student already, it may seem that they cannot be casually added into just the digital logic courses. However, with some ground work on the part of the instructor or lab or teaching assistant, it is relatively easy to provide the students with a handout or "walk-through" of a simple design covering a small subset of the commands, sufficient to do the necessary design work. It should be mentioned here that tools of this nature usually provide means to almost completely ignore the more subtle options and by just accepting the default options allow a "one-button" approach which in most cases gives a perfectly acceptable result. Exploring the full power of the tools can then be left to the curious and investigative student.

The Altera tools, as another example, provide a completely integrated set, i.e. schematic capture, implementation and simulation, and due to their much simpler architecture and elaborate interconnect structure, do not put such a heavy burden on the place and route routine, requiring less intervention by the designer. time period. One advantage is obviously that the student has to deal with only a single interface even though many different tools are used.

Also, the more powerful the tool suite and the better integrated, the more likely it is to be found in an industrial environment where considerable design is done.

IV. Results and Observations

The first time this course was offered it had an enrollment of 28 students, fourteen of which were electrical engineering majors and students who had already graduated with a degree in computer engineering. Some students were exposed for the first time in their career to PLDs, FPGAs and the development tools, starting with PLDAsm and then “graduating” to ViewLogic and Xilinx.. All students were able to finish their projects, download and test the actual hardware. This was a very compressed and very lab-intensive course. The evaluations from the students for the course were 4.9 out of 5.0. The majority of the technology students who took that course got jobs in industry through their exposure and hands-on experience with the development tools used.

The student, conversant with these tools and not requiring a substantial learning time upon entry into that industry obviously has quite an advantage. There are in fact several examples where a job was obtained due to such hands-on experience on the part of the job applicant. This is not to say that only sophisticated and powerful toolsuites should be used, as the design principles and the use of programmable devices still remain the main issue here and they can be learned quite effectively with the smaller sets. An issue which need to be brought up at this point, is that the more powerful the toolsuite, the more likely it is that networked workstations are required. This may be an unattainable goal for the smaller colleges and thus it is good to know that many of the issues mentioned in this article so far, can be effectively taught on platforms such as PCs. In fact, ViewLogic's Workview+, the window version of Workview, has an GUI (Graphical User Interface) which is so similar to Powerview, the workstation version, that the transition from one to the other is virtually effortless. It should be obvious that toolsuites of this type are very expensive, many ten of thousands of dollars, but usually available to universities at a small fraction of their cost. This carries with it one important restriction, however, the tools may not be used for industrial projects or other application where they would essentially compete with environments where the full price is levied.

V. Conclusions

Developing a course in programmable digital devices is a lot of work. It requires that the instructor be very familiar with the tools and this in itself is very time consuming. Also there are not many text books available on the subject and the development of the course notes as well as tutorials for the labs can be quite a task. However it is well worth it if it makes the engineering technology graduate more marketable which it does. Feedback from industry about the graduates employed in that area have been extremely positive. Moreover industry provides a lot of support for individuals who are serious about either including PPDs in a course e.g. senior design or developing a whole course on the subject.

BIBLIOGRAPHY

1. David E. Van den Bout, "FPGA Workout, Beginning Exercises with Intel FLEXlogic FPGA" X Engineering Software Systems corporation, 1994
2. The Programmable Logic Data Book, Xilinx, 1994
3. Jesse Jenkins, "Designing with FPGAs and CPLDS", Prentice Hall, 1994
4. Stephen M. Trimberger, "Field-Programmable Gate Array Technology", Kluwer Academic Publishers, 1995
5. John V. Oldfield and Richard C. Dorf, "Field Programmable Gate Arrays, Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems", John Wiley and Sons, 1995
6. Altera Data Book, 1994
7. Actel Data Book, 1994
8. Pak K. Chan and Samira Mourad, "Digital Design Using Field Programmable Gate Arrays", Prentice Hall, 1994

APPENDIX A

COURSE: **CET 4138C - DIGITAL PROGRAMMABLE DEVICES**
PREREQUISITE: **CET 3198C DIGITAL SYSTEMS**
GRADING PLAN: Exams - 40%
Lab Assignments - 30%
Final Project - 30%

COURSE DESCRIPTION

This course exposes the students to the various programmable devices that are commercially available. It covers the architectures and the programming of those devices. Students will be required to design and implement digital systems with a variety of programmable devices.
Development systems from Viewlogic, Xilinx, Intel ,Altera and Lattice will be used.
A formal lab report is expected with each lab assignment. The format for the reports will be given to the students. All reports are to be individual efforts.

TOPIC:

INTRODUCTION
- **Overview** of programmable devices
- Reasons for using Programmable Devices
- Trade-offs versus ASIC

PALS
- Architecture ,
- Types of circuitry suitable for implementation in PALs
- Schematics of some commercially available PALs
- Programming Examples
- Compilation and source file format

SIMPLE PLDs AND COMPLEX PLDs
- Architecture
- Types of circuitry suitable for implementation in PLDs
- Schematics of some commercially available PLDs
- Programming Examples
- Compilation and source file format

EXAM #1

FPGAs
- Architecture etc.
- Alters FPGAs — comparison and trade-offs
- Actel FPGAs
- Xilinx FPGAs

DESIGN ENTRY METHODS AND DEVELOPMENT TOOLS
- Schematic Capture
- HL Schematic Capture - XBLOX
- Boolean - XABEL - VHDL
- FSMS
- Truth Table
- Combination
- Minimization and Optimization
- Hierarchical Design
- Examples of Implementations

EXAM #2

FINAL PROJECT DUE

ROSIDA COOWAR is an Assistant Professor in the Department of Engineering Technology at the University of Central Florida. She holds a Diploma in Electronics and Telecommunications from the U.K. and a M.S.E.E. from the University of Massachusetts, Dartmouth.