Teaching a Modern Digital Systems Design Course: How to Select the Appropriate Programmable Devices and Software?

Steve Menhart
Department of Engineering Technology
University of Arkansas at Little Rock

Abstract

Faculty face many tradeoffs and choices when they are called upon to select the programmable logic and associated software that they will use in their digital courses and that is the primary focus of this paper. The integrated lecture/laboratory digital systems design course in the Department of Engineering Technology at the University of Arkansas at Little Rock is no exception. Programmable logic was first introduced to this course in the late 1990's, with the hardware consisting of simple Programmable Logic Devices (PLDs) and Complex PLDs (CPLDs). VHDL (Very high speed integrated circuit Hardware Description Language) was selected as the programming language. This worked well, with students programming their chips and incorporating them in circuits. The course was upgraded several years ago to use Altera chips and the Quartus II development platform, because the software from Cypress semiconductor, which was used previously, no longer had the desired level of support. Microcontrollers are typically not included in a digital systems design course, however for many digital applications microcontrollers provide a very cost effective solution in a compact package. This paper examines the tradeoffs and suitability of CPLDs, Field Programmable Gate Arrays (FPGAs), microcontrollers, and their associated software for digital applications typically taught in a digital design course in an engineering technology program.

Introduction

In typical electronics and computer engineering technology curriculums (and similarly for engineering) programmable devices fall into either of two tracks. The first would include CPLDs and FPGAs, and would be used to implement combinational logic, sequential logic, and state machines, etc. The second would be comprised of microcontrollers (embedded systems) and would be used to implement a variety of control-type applications. In addition to hardware, these two tracks use different development environments and programming languages. Therefore, in selecting hardware the user has most often also selected the programming language, by default.

VHDL is generally considered the preeminent hardware description languages. It was developed to provide a standardized means to describe ASICs (Application Specific Integrated Circuits). It is a completely defined, standardized high-level language and may be used to describe and simulate the behavior of a wide range of digital devices including CPLDs and FPGAs. VHDL essentially configures a digital device to implement a described application. Although VDHL may contain conditional statements such as "if-else" it does not result in a program executing line-by-line in a sequential way, but rather programs hardware that is

concurrent and executing simultaneously[1]. Microcontrollers implement a design by executing a program line-by-line at a rate determined by a clock frequency. The program may be written in a high-level language, such as C, or in assembly language. C is standard high-level language and is not tied to any one operating system or device. Assembly language is device specific, with different mnemonics and syntax for different microprocessor families, e.g., Intel and Microchip (PICs). Regardless of whether C or assembly language is used the microcontroller's program ultimately consists of op-code (strings of hex digits), which is executed byte by byte (or larger) when the program is run.

In some instances a particular application may be implemented using either programmable logic or microcontrollers. In such a case factors such as ease of implementation and cost may be the most pertinent. In other applications this is not the case and either programmable logic or a microcontroller must be used. The following examines a typical digital application, and discusses the suitability of the various digital device families and the software used for implementation.

Programmable Logic versus Microcontrollers

The following will compare and contrast the differences between an application implemented using VHDL and a CPLD, or C programming and a microcontroller[2, 3]. A simple Boolean expression will be used for this purpose: $f = a \bullet b + c$.

Implementation using VHDL

Simple Boolean expressions may be programmed using a schematic interface in which a logic circuit is drawn. For the purpose of contrasting a VHDL implementation to C, the code will be used. Figure 1 shows the architecture block of a VHDL program used to implement: $f = a \bullet b + c$.

```
architecture boolean of bool is
begin
        f <= (a and b) or c;
end boolean;
```

Figure 1. Implementation of Boolean Logic using VHDL.

The name of the architecture block is "boolean" and it has an associated entity block called "bool", which is not shown. An entity block is used to define inputs (a and b) and outputs (c). The symbols "<=" constitute the assignment operator and are read as: is assigned. When compiled this code would generate a file to program the selected device. This would result in a programmed logic circuit. Any change in an input would result in the output being immediately updated, subject only to propagation delays.

Implementation using C and a Microcontroller

Most microcontrollers may now be programmed using a high level language or assembly language. Ultimately, whichever language is used the code is converted to machine language or op-code. Figure 2 shows the main function of a C program used to implement: $f = a \bullet b + c$.

```
main()
{
while(1)
{
if (((PORTA&0x03) == 0x03) || ((PORTA&0x04) == 0x04)) PORTB|=0x01;
else PORTB &= 0xFE;
}
Return 0;
}
```

Figure 2. Implementation of Boolean Logic using C.

In the code of figure 2 the Boolean inputs a, b, and c are connected to the three least significant bits of PORTA (port A), which is comprised of eight bits. That is PORTA= ? ? ? ? ? c b a. "0x" is the hex prefix used for hexadecimal numbers. Therefore, 0x03 is the hex equivalent of binary 0000 0011. The bitwise AND operator (&) is used to mask out unwanted bits. For example, PORTA&0x03 returns 0000 00**, where each * will be either a logic 1 or 0, depending on PORTA. The logical condition "((PORTA&0x03) == 0x03)" is true if both of the two least significant bits of PORTA are logic 1 (a • b). "((PORTA&0x04) == 0x04)" tests input c and will be true if input c is logic 1. The logical OR operator is "||". Therefore, the "if" statement represents the desired Boolean function, and if true turns on the output f, which is connected to the least significant bit of port B. Note, the "|=" in the assignment results in output f being turned on, while all other bits of port B are unchanged. If the "if" statement is not true, output c is off, or logic 0. The while(1) condition sets up a perpetual loop, causing the logical "if-else" statement to be executed indefinitely.

Both figures 1 and 2 will implement f = a • b + c, however there are significant differences in the way they accomplish that. Figure 1, whether implemented using a CPLD or FPGA will result in a programmed logic circuit. The output f will reflect changes in the inputs a, b, and c instantaneously, except for a small propagation delay of a few ns. The C code of figure 2 will be converted to machine language when built. Assume the code is to be used to program a PIC 18F1320, which is an 8-bit microcontroller[4]. Assume that the PIC is being clocked from its internal 8 MHz oscillator. When built, the while(1) loop of figure 2 generates 14 bytes of op-code. Therefore, the while(1) loop will be reevaluated every 14 periods of the 8 MHz clock, or every 1.75 μs. This means that a change in input conditions might not be reflected in a change in the output for up to 1.75 μs. This is the major difference (speed) between the two implementations, and may or may not be important, depending upon the application.

Microcontrollers do possess some advantages over programmable logic. For example, even low cost microcontrollers such as the PIC 18F1320 (less than $3 each) have on-board, multi-channel, 10-bit analog-to-digital converters (ADCs). The ability to interface directly with the analog world enables the design of more comprehensive control applications. In addition, microcontrollers usually have on-board pulse width modulation units and various communication modules. Other applications can be implemented using either programmable logic or microcontrollers in much the same way, with the same tradeoffs.

Software

Programmable logic software and high-level language compilers for microcontrollers are supplied by hardware vendors. They are usually packaged as a part of an Integrated Development Environment (IDE), which enables a project to be created and simulated, and finally the hardware programmed. In many cases these are usually available free of charge, especially for educational use. A very significant factor for faculty is academic support for the software in regard to the course textbook. A textbook that supports the chosen software by way of examples and tutorials is viewed very positively by the students. In fact, given that the software is often free, this may be the deciding factor for software selection.

In terms of programmable logic, the most widely supported software (via available textbooks) is probably the Quartus II software from Altera[5], which is available free of charge (web edition). Numerous digital textbooks support this software. Of course, this software will only program Altera's CPLDs and FPGAs. This software has been adopted for the Digital Systems Design course in the Department of Engineering Technology at the University of Arkansas at Little Rock. Unfortunately, Altera no longer manufactures simple PLDs, listing these parts as obsolete. Consequently, students must implement even very basic designs using either CPLDs or FPGAs. Neither of these solutions may be optimum, because Altera CPLDs have a large minimum package size of 44 pins and FPGAs are volatile, and so require a development board. This is an example of software selection limiting hardware choices, and is a trade-off that a faculty member must evaluate carefully.

In the area of microcontrollers, Microchip's PIC processors are widely used in academia. Their entry level microcontrollers are inexpensive, yet feature packed. Microchip offers a free IDE and C-compiler for most of their microcontrollers. The microcontroller's course in this department uses a Technological Arts board[6] with a 9S12 microcontroller (Freescale[7]) and Imagecraft's C-compiler[8]. This works well for a laboratory setting, but has some drawbacks from a student's perspective. The Technological Arts board is expensive (over $100) and the 9S12 cannot be simply pulled from the board and placed in a circuit without additional circuitry. It has been observed, that although not taught as a part of our curriculum, students usually select PIC processors for their senior design projects. Students simply program their PICs and place them in their circuits without need of additional circuitry. It is expected that within the next couple of years our microcontroller's course will convert to using PIC processors exclusively.

Summary and Conclusions

Microcontrollers can implement almost any application that can be programmed into a CPLD or an FPGA. Most PIC microcontrollers have an available internal oscillator, so that the program will run on power-up, without additional circuitry. This makes PICs very easy to use. The one caveat is that a microcontroller, by virtue of it executing a sequential program, may be slower than programmed logic. That is, there could be a time lag between a change in inputs and a resulting change in an output, of several clock cycles. In the case of all inputs being controlled by human interaction these types of delays may not matter, because they are much shorter than human reaction times. Unless delays of no greater than typical propagation delays are required,

most applications should probably be implemented using microcontrollers because they are inexpensive and feature packed. Any project, which requires information to be displayed to a user via a liquid crystal display (LCD), would probably be most effectively implemented using a microcontroller. This task is made easier because C contains support for floating point numbers and functions to enable data strings to be outputted one character at a time. Therefore, microcontrollers are naturally selected when a certain level of complexity in the design is encountered, although they may still be the most appropriate device for other, simpler applications. In fact, it may be advocated that microcontrollers should be introduced at an earlier point in the curriculum and that greater emphasis be given to them.

Selection of software is largely a predetermined by the device being programmed. Cost is usually not an issue as most vendors of programmable devices and microcontrollers provide development software that is free of charge. Determining factors include ease of use, support for desired hardware, the ability to program hardware cost effectively, and academic support in the form of textbooks.

References

1. Skahill, K.., 1996, VHDL for Programmable Logic, Addison Wesley, California.
2. URL: http://www.altera.com/support/examples/vhdl/vhdl.html
3. URL: http://www.microchipc.com/
4. URL: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=64
5. URL: http://www.altera.com/
6. URL: http://www.technologicalarts.com/
7. URL: http://www.freescale.com/
8. URL: http://www.imagecraft.com/

STEVE MENHART

Dr. Menhart currently serves as a Professor of Electronics and Computer Engineering Technology at the University of Arkansas at Little Rock. He teaches courses primarily in digital systems design (VHDL) and microcontrollers. His current research interests include digital control and energy efficiency related issues.