

# A Second Undergraduate Course in Digital Logic Design: The Datapath+Controller-Based Approach

Mitchell A. Thornton<sup>1</sup> and Aaron S. Collins<sup>2</sup>

## Abstract

A second undergraduate course in digital logic design is described based on the philosophy of separating a design into a synchronous sequential controller and a datapath circuit. The datapath portion of the circuit is the portion that transforms the input signals into the corresponding output signals. The controller is the portion that governs the flow of the input signals through the various components comprising the datapath. We believe that this is a very effective way of teaching advanced digital systems design since it is an easy mechanism for allowing concepts to be described that are not commonly included in introductory digital design courses and is also a very good preparatory course for computer organization. Some of these topics include introductions to: design specification/synthesis using a *Hardware Description Language* (HDL), the role and pitfalls of asynchronous circuitry, state assignment, and behavioral synthesis issues such as resource estimation and scheduling. The use of *Field Programmable Gate Arrays* (FPGAs) and the corresponding *Computer Aided Design* (CAD) tools provided by FPGA vendors allow for an “easy-to-learn” and effective laboratory environment for such course. In addition, many of these tools are freely available in student editions to support this type of course. This means the professional design tools may be used in a standard course at nominal or zero direct cost. The remainder of the paper will focus on experiences the authors have had and descriptions of currently offered courses.

## Introduction

The rapid increase in device count on current digital *Integrated Circuits* (ICs) is in part responsible for the information age that we currently enjoy. Technology advances have allowed for increasingly smaller transistors to be fabricated and have resulted in increased density on an IC [SRC:01]. In addition to these tremendous advances in fabrication technology, *Computer Automated Design* (CAD) tools have also necessarily been required to allow designers to manipulate and use millions of transistors in a single device. For this reason many engineering schools with B.S programs in electrical engineering, computer engineering, and computer science have instituted a second undergraduate course in digital design. Such a course is justified since practicing engineers routinely deal with circuits that contain millions of transistors in a single digital IC and designs of this scope cannot be easily included in a first undergraduate course.

The first course in digital design typically covers material that begins with an introduction to two-valued Boolean algebra and a discussion of how this algebra can be used to model digital circuitry. Next, combinational logic minimization techniques are covered including Karnaugh maps [Veitch:52,

---

<sup>1</sup> Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX,  
mitch@engr.smu.edu.

<sup>2</sup> Department of Electrical and Computer Engineering, Mercer University, Macon, GA,  
collins\_as@mercer.edu.

Karnaugh:53] and possibly the tabulation method sometimes referred to as the Quine-McCluskey technique [Quine:55 ,McCluskey:56]. Usually a very brief discussion of asynchronous combinational logic circuits follows that gives just enough information to describe a latch and ultimately a flip-flop. Generally, the first course ends with discussions of techniques to design very simple synchronous deterministic finite state machines although the techniques described are generally applicable only to relatively small circuits. If time permits, additional topics are often included such as state reduction or encoding techniques, introductions to asynchronous circuit design, or introductions to the use of *Hardware Description Languages* (HDLs). Many excellent textbooks that cover these topics are available such as those by Mano, Roth and Fletcher, etc.

The authors of this paper agree that such a first course is absolutely necessary and very pertinent; however, this first course is not sufficient to prepare a B.S. level engineer for entry into a digital design job. Because of the need for using CAD tools and for designing circuits at a higher level of abstraction so that large numbers of available transistors may be used, a second course is mandatory. In the following text we make the case that the *Field Programmable Gate Array* (FPGA) is a good choice for an implementation target since it offers thousands of gate-equivalent circuits per device, has free (for educational purposes) CAD tools, and does not require knowledge of custom *Very Large Scale Integration* (VLSI) layout. Furthermore, we suggest that an effective way of teaching the design of moderately sized circuits is to separate them into a datapath portion which is the part of the circuit that transforms the input data to the output data and a controller which is the part of the circuit that is implemented as a synchronous sequential circuit and is responsible for generating the signals that control the flow of data signals throughout the datapath. While we are aware that custom digital circuit designs require unique approaches for implementation, this more rigorous approach has proved to be an excellent vehicle for introduction to digital circuit design of moderately sized circuits and is general enough to handle almost any design.

The organization of the remainder of this paper is as follows. The concept of separating moderately sized digital circuits into disjoint datapath and controller portions is provided. This approach illustrates the rigor of the method and shows how other important topics such as state encoding, resource estimation and scheduling can be introduced in the course. Next, a sample design project is proposed and various CAD tools used to realize it are described such as the use of datapath diagrams, *Algorithmic State Machine* (ASM) charts, and the *VHSIC Hardware Description Language* (VHDL). Finally we provide a sample course outline and close with conclusions.

## **Datapaths and Controllers**

For most digital design projects, there are multiple correct answers. The iteration between datapath and controller design requires that choices be made in each step in the process, and sometimes the best choice is selected based only on the judgment and experience of the designer. By presenting the process of design of circuits of this complexity, we feel that discussing a clear separation between the datapath and a single synchronous controller for the realization a target circuit aids novice designers to later decide upon the use of multiple controllers and datapath components and helps to accelerate the maturity needed to tradeoff the relative merits of each for the production of competitive designs.

### **Datapath Descriptions**

The datapath can include busses, multiplexers, tri-state logic gates, counters, shift registers, FIFOs, memories, and other digital devices. For some designs, the input data and its characteristics are known, and the output data and its characteristics are also known. The engineer must create a datapath that connects the input data to the output data, and must design a controller to co-ordinate the transfer. For instance, if a 16-bit analog-to-digital converter chip is to be the source of the data, and an 8-bit microprocessor bus is to be the destination, the designer must find a way to route the most-significant

eight bits to the bus, then to route the least significant eight bits, or vice-versa. To select first eight of 16 bits, then the other eight of 16 bits, the engineer has two obvious choices: either eight 2-to-1 multiplexers controlled by the same select signal, or to sets of eight tri-state buffers controlled by the same select signal. Either will work, and both require control signals to and from the controller to co-ordinate the process. Typically, the description of this transformation of input and output data are all that are specified by a customer along with performance constraints such as overall throughput in terms of time, cost (area or device count), and available power to be expended and perhaps other constraints such as mechanical and size requirements. The portion of the circuit that requires the control of the flow of the input data through the datapath offers the portion of the circuitry that the designer must utilize for design tradeoffs in order to meet these design specifications.

## **Controller Descriptions**

Many people are familiar with terms like a “floppy disk” or a “CPU” controller. It may not be clear that the circuits implementing these functions are designed by similar processes and that they involve both the development of a controller and a datapath circuit. The controller circuit is at the heart of every digital system, but its’ design cannot be separated from the datapath design. Indeed, in the majority of designs, complexity can be traded from either the controller or the datapath; the true art in digital design is to reach the best compromise that best satisfies customer constraints.

For the example started in the datapath description above, it is clear that the controller needs to receive some input signals and generate appropriate controller output signals. For instance, the controller (state machine) will potentially send a Start-of-Conversion signal to the *Analog to Digital* (A/D) converter, then wait for an End -of-Conversion signal. Upon receipt of the End-of-Conversion signal, the Controller will assume that a 16-bit digital word is available at the output of the A/D converter, and will proceed through some internal controller states in order to transfer the first 8-bits to the microprocessor bus, then to send the second 8-bits. To do this, select signals must be sent to the multiplexers or to the tri-states and appropriate control signals must be sent to the microprocessor bus.

Controller design is somewhat similar to programming in that it involves sequential logic development. Datapath design, on the other hand, is more open-ended with the designer reaching into his or her well of experience and knowledge to choose just the right devices to use as the data path. The ability to formulate an appropriate datapath at the level of this proposed second digital design course rests heavily upon the experiences the student received in the introductory digital design course. The ability to remember the functionality of basic gate-level and medium-scale level combinational devices is crucial for success at this stage of the digital design education process.

## **Sample Course Outline**

The course needs to briefly review the first course on digital design to assure that students have a common language. The first course needs to be extended to give students ownership of the medium scale devices commonly used to construct data paths. They see and probably understand these devices in the prerequisite course, but do not "own" the devices sufficiently to creatively design with them. In addition, timing diagrams must be fully understood at this time, prior to moving into large system (controller/data-path) design.

With full ownership of the materials briefly seen in the first course, controller/data-path design may be presented. The controller design process appears to be just the design of a state machine, and that was seen in the prerequisite course. However, the controller is a state machine with potentially a huge number of inputs and outputs, and the engineer must create both the state machine inside the controller and the data-path outside the controller. The design process involves iteration between "design of the controller" and

"design of the data path." An organized approach to the design of digital hardware is necessary in order to avoid "hardware hacking." This term is intended to carry all the negative connotations with which "software hacking" carries in the software engineering world.

Finally, once the traditional controller design process is mastered, the more modern approach of designing with VHDL can be better understood. The same process discussed above is usually used with the difference that the datapath devices are not necessarily explicitly specified by the engineer; rather they are "suggested" to the VHDL synthesizer through the choice of the VHDL code constructs that are used. The best VHDL designer will already know what he or she wants the synthesizer to create before writing the code.

Along with the basic skills needed for designing controllers and data path systems in an organized manner, many alternative design architecture choices need to be presented and compared. Finally, the timing and signal issues that can cripple an otherwise excellent design must be addressed. Asynchronous inputs, interfacing between different logic families, buffering fast data flows, clock skew, and several other topics must be fitted into the framework of controller/data-path design. The goal is to track students through an incremental design skill improvement process while fitting in all these signal and timing issues as the course progresses.

### **Course Description**

The following illustrates a course description used at one of the authors' institutions. This course has been offered and refined for the last three years. We note that although the Altera® devices and tools are used in this description, the other author uses the Xilinx® tools and devices. It is our belief that the particular FPGAs and supporting tools that are used are not important; only that students are exposed to a current toolset and industry-popular device set. We acknowledge that several other vendors offer appropriate tools and university support to utilize the ideas presented here.

#### **Catalog Description:**

Modern topics in digital systems design including the use of HDLs for circuit specification and automated synthesis tools for realization. Programmable logic devices are emphasized and used throughout the course. This course has a heavy laboratory assignment content and a design project.

#### **Textbook:**

**Fundamentals of Digital Logic with VHDL Design**, Stephen Brown and Zvonko Vranesic, McGraw-Hill, 2000, ISBN 0-07-235596-4.

#### **Course Goals:**

Equip students with the skills and modern techniques needed for the design of moderately complex digital systems. Emphasis is placed on the use of CAD tools.

#### **Prerequisites by Topic:**

Proficiency in the use of a high-level programming language and a grade of C or better in a basic digital logic design course.

#### **Major Topics Covered in the Course:**

##### **Review Topics**

Digital Logic Design and Modern Design Techniques (HDLs, ASM)  
Programmable Logic Architectures  
Programmable Logic Device Timing Characteristics

Pipelining for Increased Throughput

Latency and Initiation Rate  
Datapath versus Controller pipelining

Sequential Logic Synthesis using VHDL

ASM Charts  
FSM State Assignment

High Level Synthesis

dataflow graphs  
Resource estimation  
Scheduling  
Register Allocation  
System level pipelining  
Overlapped computation  
Limits based on I/O bus width

Partial Design Validation

Simulation  
Emulation

Introduction to Formal Design Verification

Equivalence Checking  
Model Checking

**Laboratory Content:**

This class has a required laboratory component that consists of 8/9 laboratory assignments and a final design project. Students will be required to use the student version of the Altera<sup>®</sup> MaxPlus2<sup>®</sup> software to complete all labs. A CD containing this software is supplied in the required textbook and may be installed on the student's PC.

**Project Description**

You are implement a circuit that will multiply a  $4 \times 3$  matrix with it's transpose, that is, the circuit will compute  $\mathbf{B}$  where  $\mathbf{B} = \mathbf{A}^T \mathbf{A}$ . Each matrix element,  $a_{ij}$ , is in 8-bit, 2's complement form.

Mathematically the function of this circuit is as follows:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

If the above equations don't make sense to you, please consult a textbook on linear algebra - the library has a lot of them! This design should be called *trnsmult*.

### **Project Description and Design**

There are no resource limitations for this project other than it must fit into the EPF10K20RC240-4 (Altera© flex 10k©, note the speed grade). The interface to the design is given as follows:

#### **Inputs**

- *clk, reset* - clock and asynchronous reset
- *din[7..0]* - data bus for the 8-bit matrix coefficients,  $a_{ij}$
- *cf\_load* - used to load the 12 matrix coefficients it is a signal that will go high the clock cycle before the first coefficient is loaded and will remain high while the first coefficient is present on *din*.

#### **Outputs**

- *dout[15..0]* - output bus for transformed coordinates
- *output\_rdy* - asserted when the outputs, the  $b_{ij}$  values are present on *dout[15..0]*

This project has constraints on the I/O resources since there is only a single 8-bit input and a single 16-bit output bus. The order in which the matrix coefficients are provided to the input bus, *din[7..0]* is as follows (NOTE: column NOT row order):

$$a_{11}, a_{21}, a_{31}, a_{41}, a_{12}, a_{22}, a_{32}, a_{42}, a_{13}, a_{23}, a_{33}, a_{43}$$

Where  $a_{11}$  is input initially, followed by  $a_{21}$ , etc.

Likewise, the output values should appear in the following order with  $b_{11}$  first and  $b_{33}$  last in row order (NOT column order):

$$b_{11}, b_{12}, b_{13}, b_{21}, b_{22}, b_{23}, b_{31}, b_{32}, b_{33}$$

#### **Timing Constraints**

Your design should have an initiation rate of 15 clock cycles with a latency of 23 clock cycles. The clock frequency must be greater than 10 MHz.

## **Project Implementation**

This project has several interesting design challenges. Careful examination of the specifications and I/O bus sizes reveal that students may not simply read in all 12, 8-bit data values, perform the calculations and then output the results. Rather, it is necessary that computations begin before all values are read in, thus enhancing the concept of pipelining. Furthermore, in order to preserve the specified throughput, computation on the following set of input data must begin before the former set is complete. The designation of a specific target FPGA ensures that the student have a reasonable set of design area constraints allowing them to “flex” their newly found design muscles in terms of the classic speed versus area tradeoffs. While the class focuses more on design techniques than mere tool usage, students are encouraged to experiment with features not discussed in the lab or class in order to meet the specifications given..

While such a design would appear to be quite daunting for entering students to the class, careful construction of prior experiments and course lectures have proven to show that this project can be quite satisfying to most students and gives them a sense of confidence since they are able to produce designs that apparently perform much more relevant work than that typically encountered in introductory digital design courses that may perhaps culminate in the design of a simple traffic controller. Furthermore, incentives such as giving additional test points for every 5MHz of clock speed over the specified amount foster a sense of competition and fun in completing this project.

## **Conclusions**

We have argued for the need of a second course in digital design in B.S. level programs in electrical engineering, computer engineering, and computer science that wish to educate students for entry-level jobs in the digital design area. Furthermore we described a method for introducing design projects as being composed of two distinct parts; the datapath and the controller circuitry. Such an organization provides a convenient mechanism for introducing relevant design concerns such as area versus speed tradeoffs, controller versus datapath complexity, state encoding, resource estimation, pipelining, and many other issues. We remark that a computer (arguably one of the most complex digital circuits currently being implemented) is just a controller and a datapath designed to meet the specification stated for the target device. The ability to design a controller and a datapath system is therefore perhaps the most fundamental level of digital hardware design. With the knowledge and use of this skill set, an entry-level engineering graduate can create any digital system. A sample course outline and design project has also been given that illustrate the approach we describe.

## **References**

- [SRC:01] Semiconductor Research Corporation, (2001), "International Technology Roadmap for Semiconductors", <http://public.itrs.net/Files/2001ITRS/Home.htm>.
- [Veitch:52], Veitch, E. W., (1952), "A Chart Method for Simplifying Truth Functions", *Proc. Assoc. Comp. Mach.*, May:127-133.
- [Karnaugh:53], Karnaugh, M., (1953), "The Map Method for Synthesis of Combinational Logic Circuits", *Trans. of A.I.E.E.*, 72(pt. 1):593-599.
- [Quine:55], Quine, W., (1955), "A Way to Simplify Truth Functions", *Amer. Math. Monthly*, 62:627-631.
- [McCluskey:56], McCluskey, E.J., "Minimization of Boolean Functions", *Bell System Tech. Journ.*, 35.
- .

### **Mitchell A. Thornton**

Mitchell Thornton is currently an associate professor of computer science and engineering at Southern Methodist University. He received the Ph.D. in computer engineering and the M.S. in computer science from Southern Methodist University, the M.S. in electrical engineering from the University of Texas at Arlington and the B.S. in electrical engineering from Oklahoma State University. Previously, Mitch served on the engineering faculties at the University of Arkansas and Mississippi State University. His industry experience includes five years at E-Systems (now L3-Com) in Greenville, Texas and one year at Cyrix Corporation (now Via Technologies) in Richardson, Texas. His research interests focus on electronic design automation algorithms and he has published over 50 refereed research papers in this area. Mitch is a senior member of the IEEE and is a registered professional engineer in the states of Texas, Mississippi, and Arkansas.

### **Aaron S. Collins**

Aaron Collins is currently professor and chair of electrical and computer engineering at Mercer University. He received his B.S., M.S., and Ph.D. degrees from Clemson University in electrical engineering. Previously, Aaron served on the electrical engineering faculty at Tennessee Tech University and at Clemson University. His industry experience includes two years with the Lockheed-Georgia Corporation, three years with Combustion Engineering, and five summers with NASA Goddard SFC and Computer Sciences Corporation. His most active current interest is digital system design. He is a senior member of IEEE and is a registered engineer in South Carolina.

