Making It Real: Complex Specification-Based Projects for VLSI, VHDL, and Capstone Design

Charles E. Stroud [1]

Abstract

The development specifications for complex, group-oriented projects can provide students with realistic, industry-like experience in design, design verification, and system integration. Projects of sufficient complexity facilitate partitioning a large group of students into smaller groups responsible for various tasks including hardware design, software development, physical design, and prototype implementation. The development and verification of these specifications can require considerable time, particularly in the case of hardware/software co-design projects. However, with minor modifications, the same or similar projects can be subsequently used in multiple courses such as VLSI design and testing courses, VHDL modeling, simulation and synthesis courses, as well as senior capstone design project courses. This paper gives an overview of hardware/software co-design projects, most of which resulted directly from industry suggestions, that have been developed and modified for use in multiple courses.

Introduction

Industrial product design and development is intensively group-oriented due to the size and complexity of most current products and systems. This requires engineers to meet specifications and schedules with a high probability that designs will not only work but will also smoothly integrate into the overall system. For example, until the late 1980s, Application Specific Integrated Circuits (ASICs) were typically designed by a single designer per chip with integration at the printed circuit board (PCB) level representing group-oriented design. Similarly, PCBs were integrated at the unit and/or system-level by various groups of designers followed by integration with system software. The failure to meet design specifications requires redesign of one or more ASICs or PCBs with resultant delays to final system integration along with the associated time-to-market delay and loss of revenue for the product. Many, if not most, redesigns result from lack of attention to specifications with simple errors like reversed bus order, incorrect signal names, incorrect active signal levels, etc. As the complexity of VLSI devices grew with the capabilities of hardware description languages (HDLs), such as VHDL and Verilog, and their associated Computer-Aided Design (CAD) tools for automated synthesis, groups of designers were required for a single ASIC design. While the features and capabilities of HDLs and CAD tools facilitate higher levels of modeling and design verification, the need to meet system specifications and requirements has not diminished. In fact, the need for paying attention to the details of specifications has grown due to shortened design cycles and schedules. This requires more collaborative group effort, particularly between hardware and software designers, where each designer is relied upon to design their individual component to work properly with the rest of the system in order for the project to succeed.

Gaining experience in this type of design environment can be difficult for students in the individual or small group performance-based educational system. Complex specification-based projects provide this design experience for students, particularly when the project is implemented in an integrated hardware/software environment and when the project is of sufficient complexity to require multiple students to work together to successfully complete the project such that the design fully meets the specifications. In order to maximize the value added to the students' design experience, the requirements and specifications for the project must be extensive and well defined, usually to

[1] Dept. of Electrical and Computer Engineering, 200 Broun Hall, Auburn University, AL 36849-5201.

a greater degree than those typically encountered in industry where requirements and specifications are often developed as the project progresses. Developing this level of detail and accuracy in the specifications for a project can be time consuming, but this investment can yield greater dividends when the project is used in multiple courses. In order to maximize the effectiveness of a project in a given course, the requirements and specifications may need to be modified to fit the framework of that specific course. This paper gives an overview and assessment of specifications-based hardware/software co-design projects that have been developed by the author and modified for use in multiple courses. Many of the project ideas resulted directly from industry suggestions. The paper begins with a discussion of five types of project-oriented design courses and the level of detail for the specifications needed in each of these courses. This is followed by an overview of specific projects that have been developed by the author and used in these various courses over the past ten years. Next, the specifications for one of these projects are provided to give a detailed example of the level of detail in the specifications and to provide these specifications for use (with or without modification) in similar courses at other universities.

Design Project Courses

The type of course can dictate the level of detail needed in specifications for the project. Five types of courses in which a single project can be used include capstone design courses, VLSI design courses, VLSI testing courses, HDL courses, and programmable logic courses. HDL courses typically focus on either VHDL or Verilog. Programmable logic courses typically focus on Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). Often, HDLs and programmable logic are combined in a single course due to the natural interface between these two subjects as a result of current CAD tools where an HDL is easily synthesized in to FPGAs or CPLDs for a fast and effective hardware implementation. As a result, these courses will be assumed to be combined in the following discussion. Similarly, VLSI design and testing courses are often associated with each other as will be assumed in the following discussion of specifications for projects related to these courses.

Capstone Design

Capstone design project courses offer an excellent opportunity to provide students with an industrial-type design environment. This is particularly true with hardware/software co-design projects of sufficient complexity to warrant partitioning the class into multiple groups (of three to four students each) responsible for hardware design, software design, hardware/software interface design, physical design and prototype implementation. In this type of course, it is important for the instructor to have a detailed set of specifications that would represent customer requirements for the final system. However, it is also essential to give the students experience with partitioning the system and developing requirements and specifications for each partition as part of the project. Therefore, with the "customer" requirements in hand, the instructor should provide general architectural guidance to assist the complete group of students in developing detailed and realistic requirements, specifications, and schedules during the first few weeks of project while background lectures are given. The instructor then acts as the project manager to oversee and monitor the progress of each group while weekly class meetings focus on status reports presented by each group along with technical discussion and communication between groups. Sufficient time should be given for system level integration and debugging at the end of the course to facilitate a successful completion of the project since students are typically optimistic (due to their inexperience) in regard to how much time and effort is involved in this phase of a project. The industrial-like environment is further enhanced with performance reviews by students at the end of the semester ranking group members, groups, and the entire class in terms of their contributions to the project successes and/or failures. Since the detailed specifications are developed as part of the project for this type course, there is not as much investment on the part of the instructor before the project (course) begins. However, it is important that the project be successful for the students and, hence, it is important that sufficient time and effort goes into defining the project at all levels of detail to help ensure that the project has a high probability of success.

VLSI Design and Testing

VLSI courses are focused by nature on hardware projects while in practice there are often software interfaces and implications associated with the system-level operation of many VLSI designs. As a result, VLSI design projects

that include processor interfaces and timing requirements provide students with a better understanding of hardware/software interfaces. The overall operation of the design in the intended system is also an important aspect for students in understanding the importance of design specifications as they relate to proper system operation. Since the focus of the project is typically a single VLSI device, a project of sufficient complexity to warrant partitioning of the design into 3 to 4 subcircuits with each student designing a particular subcircuit while forming groups to complete the chip design provides the students with a more realistic experience in current VLSI design methodologies. Therefore, detailed specifications should be developed prior to the project (course) where the chip architecture is partitioned into subcircuits of approximately equal complexity. The class can then be broken into groups of 3 to 4 students for each chip with each student in the group responsible for the design of a subcircuit. Initially each student will design their subcircuit at the logic gate level that corresponds to the implementation technology (for example, NAND, NOR, AND-OR-Invert, and OR-AND-Invert gates with appropriate fan-in limitations for CMOS technology) and verify their design via logic simulation. To ensure that each design works, the instructor should independently develop design verification vectors that thoroughly test each subcircuit against the specifications and requirements. Each group would then combine their subcircuit gate-level models to simulate, debug as needed, and verify their chip meets the chip-level specifications via logic simulation. Once again, each chip should be tested by a set of vectors developed by the instructor to thoroughly verify the operation of the chip according to the specifications. It is important that the students are not given access to these independent test vectors; otherwise they will not develop the skills needed for developing design verification vectors in industry. At this point in the project, each student is responsible for the physical design and verification of their subcircuit, followed by independent verification by the instructor. The group then puts their chip together near the end of the semester and verifies operation with independent verification by the instructor prior to submission to MOSIS.

In a subsequent VLSI testing course, students can develop test vectors to achieve some specified fault coverage (for example, greater than 95% using the single stuck-at gate fault model) and test the fabricated chips when they come back from MOSIS. In this case, the specifications developed for the VLSI design course are used by the students in the testing course to understand the architecture and operation of the chip in order to develop test vectors as well as to understand the architectural and design issues that affect the testability of the device. Students that take both the VLSI design course followed by the testing course test their own chip and, aside from a greater feeling of ownership, see first hand how their design could be made more testable. On the other hand, students that did not take the VLSI testing course gain experience in understanding specifications as well as understanding another person's design as they relate to testing and test development.

HDL and Programmable Logic

VHDL and Verilog with subsequent synthesis into an FPGA or CPLD allow students to manage more complex designs than in the case of the gate and transistor level VLSI design. The same specifications used in the VLSI class can be used for the VHDL/Verilog course. However, when synthesizing into FPGAs or CPLDs, the specifications may need to be modified to reflect features and/or limitations of the implementation technology. In this case, each student can easily model the entire chip that required several students in the VLSI design course. Using the same specifications and partitioning of subcircuits as was used in the VLSI design course helps students in the VHDL/Verilog course to gain experience with hierarchical modeling and design verification. Students who take the VLSI design course and then the VHDL/Verilog course obtain an appreciation not only for the efficiency of designing with HDLs in conjunction with synthesis CAD tools but also for techniques for writing VHDL models for optimized synthesis results as a result of their intimate knowledge of the gate level design of the project.

Overview of Previous Projects

Overviews of various projects that have been developed by the author and used in multiple classes are listed below. A more detailed overview of the logic analyzer projects is given in the next section along with an example of detailed specifications used in a VLSI design, VLSI testing, and VHDL courses. A controller for an electrostatic particle separator was suggested as a project by the University of Kentucky Center for Applied Energy Research. The electrostatic particle separator is used to remove carbon from fly ash at coal burning power plants so that the fly ash can be sold as a concrete additive; otherwise, ash with high carbon content must be discarded as waste material at the expense of the power plant. The Built-In Self-Test (BIST) project was suggested by the US Air Force for

testing mixed-signal systems. This BIST project was later funded by the US Air Force and DARPA for graduate and undergraduate research with the VLSI devices designed in the VLSI class as well as the VHDL models developed in the VHDL class used to construct hardware/software units for evaluation of the BIST approach on various mixed-signal benchmark circuits. The test machine for VLSI devices was a hardware/software co-design project suggested by Oak Ridge National Laboratory as a low cost solution to testing digital devices being designed at their facility and fabricated through MOSIS. The only hardware projects that did not have a software counterpart were the FPGA, CPLD, and microprocessor projects; yet, compiler and programming software for all three of these devices could have been incorporated to create a more complex system for a capstone design project, for example. As can be seen from the summary below, most of these projects were used in multiple courses.

Logic Analyzers and Data Acquisition Systems

1. *PC-Based Logic Analyzer*: Prototype hardware/software co-design in Senior Capstone Design (Fall 1994)
2. *Embedded Logic Analyzer for FPGAs and CPLDs*: Parameterized VHDL with prototype hardware/software co-design in Senior Capstone Design (Fall 1999) (for more details see reference 1)
3. *Logic Analyzer on a Chip*: Design and fabrication via MOSIS in VLSI Design (Fall 1999), test development and testing of MOSIS TinyChips in Digital System Testing (Spring 2000), modeling and synthesis in FPGAs in Intro to VHDL (Spring 2000) (detailed specifications for this project are provided in the next section)

Micro-Controllers

1. *8-bit Microprocessor*: Design and fabrication via MOSIS in VLSI Design (Fall 2000), test development and testing of fabricated MOSIS TinyChips in Digital System Testing (Spring 2001), modeling and synthesis in FPGAs in Intro to VHDL (Spring 1999)
2. *Controller for Electrostatic Particle Separator*: Design and fabrication via MOSIS in VLSI Design (Fall 1998), test development and testing of fabricated MOSIS TinyChips in Digital System Testing (Spring 1999), modeling and synthesis in FPGAs in Intro to VHDL (Fall 1997)

Built-In Self-Test and Test Machines

1. *Mixed-Signal Built-In Self-Test*, Version 1: Design and fabrication via MOSIS of Test Pattern Generator and Output Response Analyzer TinyChips in VLSI Design (Fall 1997), test development and testing of fabricated MOSIS TinyChips in Digital System Testing (Spring 1998) (for more details see references 2, 3, and 4)
2. *Mixed-Signal Built-In Self-Test*, Version 2: Design and fabrication via MOSIS of Test Pattern Generator and Output Response Analyzer TinyChips in VLSI Design (Fall 2001), test development and testing of fabricated MOSIS TinyChips in Digital System Testing (Spring 2002), modeling and synthesis in FPGAs in Intro to VHDL (Spring 2002) (for more details see reference 5)
3. *Test Machine for VLSI Devices Fabricated through MOSIS*, Version 1: prototype TTL-based hardware/software co-design in Senior Capstone Design (Spring 1994) (for more details see reference 6)
4. *Test Machine for VLSI Devices Fabricated through MOSIS*, Version 2: Design and fabrication via MOSIS in VLSI Design (Fall 1995), printed circuit board design and fabrication for MOSIS TinyChip-based test machine and software for bi-directional I/O control of test machine in Independent Study (Spring 1996)
5. *Test Machine for VLSI Devices Fabricated through MOSIS*, Version 3: FPGA-based hardware/software co-design in Senior Capstone Design (Fall 1996)

Programmable Logic

1. *Look-Up Table Based FPGA*: Design and fabrication via MOSIS in VLSI Design (Fall 2002), test development and testing fabricated MOSIS TinyChips in Digital System Testing (Spring 2003)
2. *Re-programmable PLA-Based CPLD*: Design and fabrication in VLSI Design (Fall 1996), test development and testing fabricated MOSIS TinyChips in Digital System Testing (Spring 1997)

Specifications Example: Logic Analyzer

The logic analyzer was first implemented as a TTL SSI/MSI-based wire-wrap board that interfaced to a PC for graphical control and display in a capstone design course in fall of 1994. During the summer of 1999, Cypress Semiconductor inquired about the effort required to implement an embedded logic analyzer for their Delta 39K® series CPLDs. Their inquiry lead to an second capstone design course in fall of 1999 where a parameterized VHDL model was developed for inclusion with a user's system function VHDL and subsequent synthesis into a Cypress 39K CPLD. This project resulted in the development of an actual product for Cypress Semiconductor (similar to

SignalTap® from Altera) and was a predecessor to ChipScope® from Xilinx. Since the capstone design project took place before the introduction of the Cypress 39K CPLD, the project included the development of a wire-wrap board that emulated a 39K CPLD using multiple Cypress 37K CPLDs and Random Access Memories (RAMs). The same basic project was modified such that the design would fit into a MOSIS TinyChip and used in the VLSI design and testing courses and again in the Intro to VHDL course. The following subsections give the specifications as given to students in the VLSI design and VHDL courses.

Architectural Overview

The Logic Analyzer (LA) chip is a device intended for inclusion on a Printed Circuit Board (PCB) such that system signals (data and/or control) on the PCB can be sampled on-line, in real time, for off-line display of the sampled signal waveforms on a PC. When the LA is incorporated onto a PCB, 8 candidate signals for sampling along with a sampling clock hardwired to the LA chip. Then the LA chip can be programmed for the desired triggering and sampling operation during the operation of the system. The LA chip is partitioned into five basic components: the Micro-Processor Interface (MPI), the Trigger circuit (TRIG), the Sample Memory (SMEM), the Address controller (ADD), and the Programmable Interconnect Module (PIM), as illustrated in Figure 1. The PIM allows the user to select up to 4 signals to sampled from a total of 8 input signals with the same 4 signals selected for establishing the triggering condition. The TRIG allows the user to define the triggering conditions in a sum-of-products (or product-of-sums) form with up to 2 product terms for the 4 signals selected by the PIM. The SMEM is the sample memory where signal values are stored in sequence once the LA has been triggered; the SMEM can store up to 16 consecutive samples of the 4 selected signals to be sampled. The triggering process consists of programming the PIM and TRIG and then arming the LA; after which, once the triggering condition has been observed, the sampling process begins. Alternatively, the LA can be manually triggered. The ADD is used to control the sequencing of the SMEM sample storage locations for writing (once the LA has been triggered) and for reading (for data transfer to the display software running on a PC). Finally, the MPI provides the user with an interface for programming the PIM and TRIG, arming the LA, and reading the SMEM once the LA has been triggered and the sampling process is complete.

Primary Input/Output Pins:

SIGNAL NAME	I/O	DESCRIPTION
DIN3-0	I	4-bit input write data for the MPI, PIM, and TRIG registers
ADD2-0	I	3-bit address for selecting register where DIN is to be written or DOUT is to be read from
STROBE	I	Active low enable input for MPI operations
CLK	I	Sample clock input
INP7-0	I	Data input channels for sampling and triggering source selection
DOUT3-0	O	4-bit output read data from SMEM3-0 (CNT3-0 from ADD when TestReg0=1 & TestReg1=0, PIM3-0 when TestReg1=1 & TestReg0=0)
BUSY		Active high output indicating active MPI operation or LA is armed but not done sampling BUSY returns to 0 after MPI operation or when sampling is finished
DONE		Active high output indicating LA has been armed, triggered, and has finished sampling
TRIG		Active high output indicating LA has been armed and triggered (PTRG from TRIG if TestReg0 does not equal TestReg1)

Register Address Map:

ADD2	ADD1	ADD0	MPI Operation (active STROBE)	DIN/DOUT3	DIN/DOUT2	DIN/DOUT1	DIN/DOUT0
0	0	0	Control Register (DIN bits)	SOP/POS *	RESET **	TESTREG1	TESTREG0
0	0	1	Manually Trigger Logic Analyzer	X	X	X	X
0	1	0	Arm Logic Analyzer	X	X	X	X
0	1	1	Advance SMEM Add (DOUT bits)	SMEM BIT3	SMEM BIT2	SMEM BIT1	SMEM BIT0
1	0	0	Write PIM0/TRIG Bit PT1 (DIN bits)	BIT3	BIT2	BIT1	BIT0
1	0	1	Write PIM1/TRIG Bit-Bar 1 (DIN bits)	BIT3	BIT2	BIT1	BIT0
1	1	0	Write PIM2/TRIG Bit 2 (DIN bits)	BIT3	BIT2	BIT1	BIT0
1	1	1	Write PIM3/TRIG Bit-Bar 2 (DIN bits)	BIT3	BIT2	BIT1	BIT0

Notes: * SOP/POS implements product-of-sums in TRIG when this bit=1, otherwise sum-of-products
****** RESET resets TRIG, DONE, BUSY output flags and ADD to all 0s (when this bit=1)

Control Bits:

TESTREG1	TESTREG0	Test Mode/Register Selection
0	0	DOUT3-0 <= SMEM3-0/Select PIM Registers
0	1	DOUT3-0 <= CNT3-0/Select PIM Registers
1	0	DOUT3-0 <= PIM3-0/Select TRIG Registers
1	1	DOUT3-0 <= SMEM3-0/Select TRIG Registers

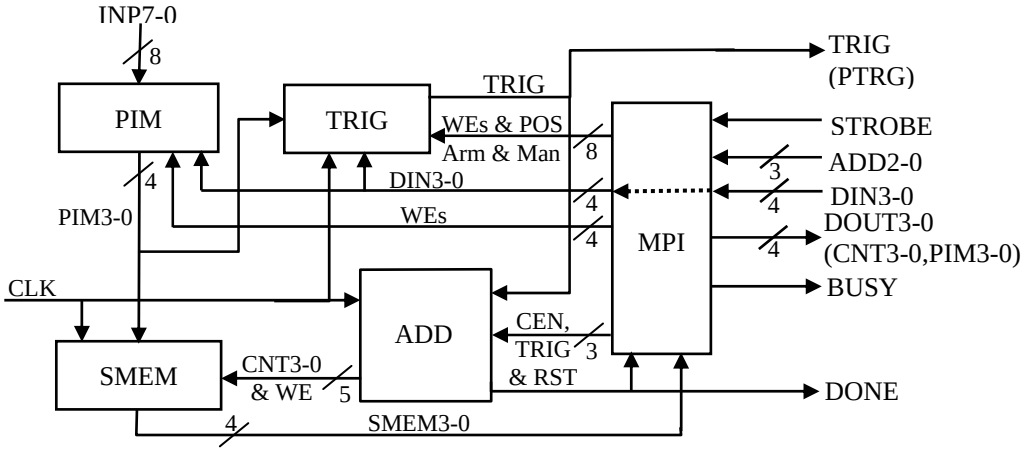

Figure 1. Block Diagram of Logic Analyzer Chip

General Operation

LA Initialization: The LA can be initialized by writing address 000 with data X1XX (note address and data bit ordering is from MSB to LSB) followed by a rising edge of CLK. This resets the output flags (BUSY, TRIG, and DONE) to 0, resets the ADD circuit to SMEM address location 0, and disarms the LA by setting internal signals WR and CEN to logic 0.

PIM Programming: The input channels to be sampled can be selected by writing address 000 with XX00 to select access to the PIM. Then each of the sample PIM registers can be written via addresses 100 through 111 for registers 0 through 3, respectively, which in turn control which input channel will be sampled and stored in SMEM bits 0 through 3, respectively, as well as then inputs to the TRIG inputs 0 through 3, respectively. The 3-bits of data written to each of these registers indicates which of the 8 input channels will be selected for sampling with data 000 selecting IN0, and so on, up to data 111 selecting IN7.

TRIG Programming: The triggering conditions that will initiate the sampling process can be established by programming sum-of-product equations into the 2 product term bit and bit-bar registers of the TRIG. These can be accessed by writing address 000 with XX11 to select the 2 product term bit registers and the 2 product term bit-bar registers. Note that since both product terms are ORed together, then all 4 registers must be written with valid data to ensure proper operation of the triggering condition. A logic 1 written into a given bit or bit-bar register will activate that input or inverted input, respectively, as a literal in that respective product term. Note that a logic 0 written into both bit and bit-bar registers will disable that input from activating the product term, while a logic 1 written into both the bit and bit-bar registers will mean that the literal is always active. In order to maximize the effectiveness of the 2 product term limit, product-of-sum expressions can be programmed by writing address 000 with 1XXX which, in turn, inverts the output of the PLA.

Trigger Activation: Once programmed, the LA can be armed by writing address 010 with XXXX. When the trigger condition specified by the product terms has been satisfied, the sampling sequence will begin. Alternatively, the LA can be manually triggered (to force the sampling sequence to begin independent of the programmed trigger conditions) by writing address 001 with XXXX. Note that to re-arm the LA, reset the LA and then address 010 or 001 must be re-written to XXXX (depending on the desired triggering) once the sampling sequence is completed.

Sampling Sequence: Once the LA has been triggered, the logic values of the 4 selected inputs to be sampled will be stored in the SMEM memory locations during each clock cycle of the selected input sampling clock. The ADD

controls sequencing through the address locations of the SMEM until all memory 12 locations in the SMEM have been written with sampled data, at which time the ADD discontinues operation in order to avoid overwriting the stored samples (the ADD should return to the first SMEM location that was written in order to prepare for the data retrieval process).

Sampled Data Retrieval: Once the sampling sequence is completed, the sampled data stored in the first SMEM location will be present at the DOUT3-0 outputs (as long a TESTREG0=TESTREG1, indicating a non-test mode). The SMEM location output to DOUT3-0 can be advanced by writing address 011 with XXXX. This process (writing to address 011) is repeated until the entire SMEM has been read (all 12 memory locations). At this point, the LA can be reprogrammed and/or reset then re-armed so that a new set of data samples is taken.

Output Flags: BUSY will normally be 0 but will go to a 1 during any MPI operation (anytime STROBE=0, BUSY=1, until STROBE goes back to 1). BUSY will also go to a 1 once the LA has been armed (or manually triggered) and will stay at logic 1 until the sampling sequence has been completed, then BUSY will go back to 0. TRIG will also normally be 0 but will go to a 1 once the LA has been triggered and will remain 1 until the LA is reset (disarmed) by writing address 000 with X1XX, at which time TRIG returns to 0. DONE will normally be 0 but will go to 1 once the sampling sequence is complete and will remain 1 until the LA is reset (disarmed) by writing address 000 with X1XX, at which time DONE returns to 0.

Subcircuit Specifications

PIM – Programmable Interconnect Module:
Inputs (16 inputs): INP7-0, DIN2-0, WEP3-0, STROBE
Outputs (4 outputs): PIM3-0
Description: The PIM consists of 4 identical subcircuits with each of these consisting of a 4-bit register and a 16-to-1 multiplexer. The data inputs to the multiplexer are the 8 input channels (IN7-0). The data inputs to the PIM registers are the 4-bit data bus (DIN3-0). WEP3-0 & are 4 active high write enables for the PIM registers. The PIM registers consist of flip-flops (clocked on the rising edge of STROBE) with the outputs of each register driving its respective multiplexer in order to select 1 of the 8 input channels.

TRIG – Trigger Circuit:
Inputs (21 inputs): PIM3-0, DIN3-0, WEB1-0, WEBB1-0, POS, ARM, MAN, RST, TESTREG1-0, STROBE, CLK
Outputs (2 output): TRIG(PTRG),TRIG (to ADD)
Description: The TRIG circuit is a Programmable Logic Array (PLA) with a fully programmable AND-plane and a fixed OR-plane. The AND-plane consists of 2 product terms of bit and bit-bar for the 4 inputs (PIM3-0). The literal selection (bit, bit-bar, or neither) for each product term is determined based on the contents of two 4-bit registers (one for bits and one for bit-bars) where a logic 1 will active the associated literal. The data inputs to the product term registers is the 4-bit data bus (DIN3-0) and the data is written into the registers based on the active high write enables (WEB1-0 and WEBB1-0). The outputs of the PLA (PTRG) will be inverted for product-of-sums implementations when POS=1. The product term registers consist of flip-flops (clocked on the rising edge of STROBE) with the outputs of these registers driving the programmable AND-plane. The principle output of the TRIG circuit is the active high signal TRIG which drives the ADD circuit and indicates that a trigger condition has been encountered (indicated by a one on the PTRG output of the PLA). Either when ARM is high and PTRG goes high or when MAN goes high, TRIG goes high on the rising edge of CLK and remains high until RST is activated, at which time TRIG resets. The TRIG signal is also a primary output of the LA, however, during test mode (when TESTREG0 is not equal to TESTREG1) the PLA output PTRG is sent out on the primary output (but not to the counter).

SMEM – Sample Memory:
Inputs (10 inputs): PIM3-0, CNT3-0, WE, CLK
Outputs (4 outputs): DO3-0
Description: The SMEM is a Random Access Memory (RAM) with 16 address locations and 4-bit data words per address location. The input data to be written into the RAM comes in on PIM3-0. The output data read from the RAM (as specified by CNT3-0) goes out on SMEM3-0 to the MPI where they are read out on DOUT3-0. The write enable for the RAM (WE) is also active high such that a given address location *i* (as specified by CNT3-0) is written

with the input data only when WE=1 on the rising edge of CLK. As a result, each single bit memory location consists of a rising edge triggered flip-flop.

ADD – Address Controller:
Inputs (4 inputs): CLK, RST, CEN, TRIG
Outputs (6 outputs): CNT3-0, DONE, WE
Description: The ADD is a 4-bit synchronous, rising edge-triggered, binary up-counter (clocked by CLK) with active high count enable (CEN) and active high synchronous reset (RST). The outputs of the counter are CNT3-0. Note that RST takes precedence over all other functions. A valid sampling sequence begins when TRIG goes high and the sampling sequence continues until all 12 memory locations have been written. At that point the counter produces the active high DONE when all counter bits have returned to logic 0 after the sampling sequence and counting discontinues. Once DONE goes high, it remains high until RST is activated. The count enable (CEN) from the MPI is used to advance the counter during the read cycle; the counter advances once and only once during the time that CEN is high and will not advance again until CEN goes low and then back high. The address circuit also produces the active high write enable (WE) to control writing the SMEM during the sampling sequence.

MPI – Micro-Processor Interface:
Inputs (22 inputs): STROBE, ADD2-0, DIN3-0, CNT3-0, SMEM3-0, PIM3-0, DONE
Outputs (20 outputs): DOUT3-0, BUSY, WEB1-0, WEBB1-0, WEP3-0, POS, RST, CEN, ARM, MAN, TESTREG1-0
Description: The MPI performs all interface functions between the LA and the PC. All MPI operations to the LA via the PC are performed according to the timing diagram of Figure 2. There is one control register (described above) which is written with the input data (DIN3-0) on the rising edge of STROBE based on the 3-bit address (ADD2-0) specified above. Similarly, write operations to the PIM and TRIG registers are based on the 3-bit address (ADD2-0) and the values of TESTREG1-0 (as specified above) with the MPI supplying the active high write enable to the appropriate register. As a result, this function can be performed using a decoder with inputs comprised of ADD2-0 and TestREG1-0. Internal signals RST, POS, and TESTREG1-0 are simply the contents of the control register. However, when RST=1, the output flag BUSY as well as the internal flag ARM should also be reset to 0 such that the state of the LA is not armed, not triggered, and not waiting to complete the sampling sequence. The BUSY output is a logic 1 whenever STROBE=0, also BUSY=1 when the LA is armed (or manually triggered) and remains a logic 1 until DONE goes active; otherwise BUSY=0. The internal signal to the ADD circuit CEN is set to a logic 1 during the data retrieval process when STROBE=0 and ADD2-0=011 and this input condition must remain active for sufficient time for the ADD to one-shot off of CEN using the sample clock CLK (here we'll assume that

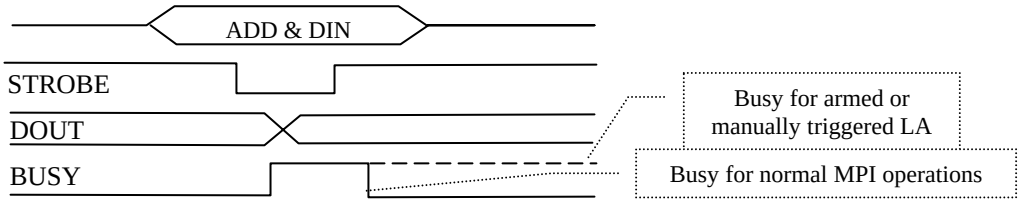

Figure 2. Timing Diagram for Logic Analyzer MPI Operation

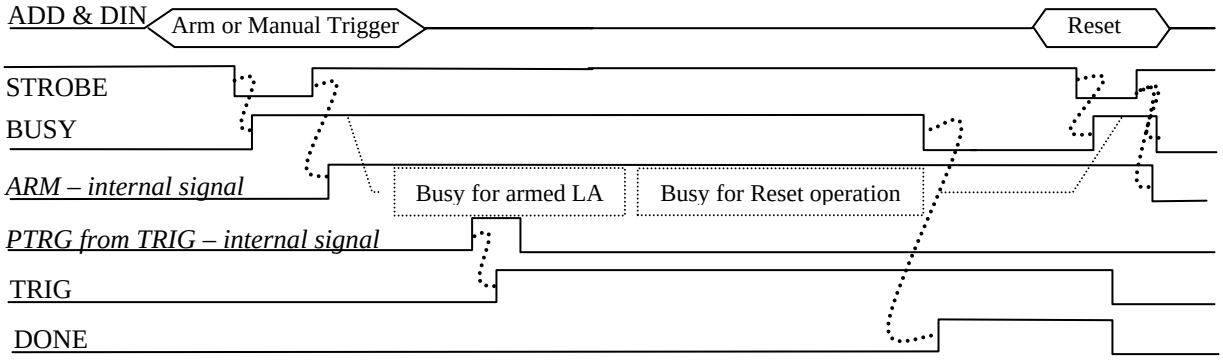

Figure 3. Timing Diagram for Logic Analyzer Output and Internal Flags

time STROBE is low is longer than the period of CLK). Normal operation at the primary outputs is achieved when TESTREG0=TESTREG1, otherwise when TESTREG0=1 and TESTREG1=0, internal signals CNT3-0 are brought out on the primary outputs (DOUT3-0) to facilitate improved design verification and testability, or when TESTREG1=1 and TESTREG0=0, internal signals PIM3-0 are brought to the primary outputs DOUT3-0. Figure 3 illustrates the operation of the various output and internal flags for an arming of the LA. The only difference in the case of a manual trigger is that PTRG is ignored and TRIG immediately goes high (on the next active edge of CLK) such that the sampling sequence begins.

Summary and Conclusions

Complex specification-based projects for capstone design, VLSI design and testing, and HDL and programmable logic courses provide students with experience that is closely related to the industry environments in which they will be working. These projects expose students to the need to understand and pay careful attention to the details of specifications associated with the design to minimize design errors and ensure smooth system-level integration. However, to maximize the effectiveness of these projects, it is important that the specifications be detailed and accurate. This requires considerable time on the part of the instructor to verify the specifications as well as the design prior to the beginning of the course in order to minimize student frustrations associated with changing requirements (they will get plenty of that in industry). Based on the author's experience, a capstone design project requires at least two weeks of concentrated effort to develop specifications for the project. While these specifications are not generally given to the students in their entirety since the students help to develop the system requirements and specifications as part of the project, the independent development of specifications by the instructor helps to ensure that the project is challenging but achievable with a high probability for successful completion. The development of a good set specifications for a VLSI design class usually requires about four weeks which includes gate-level design and design verification, a by-product of which is the design verification vectors that would be used to independently verify that student designs meet the specifications for the subcircuits as well as the complete chip. Less time is required to develop specifications for projects for VHDL/Verilog courses since the HDLs deal with a higher level of design abstraction. However, the specifications developed for the VLSI design course can be used with little or no modifications in VHDL/Verilog courses. Therefore, an initial investment in the development of a good set of specifications for a complex project can yield dividends through use in multiple courses as well as benefit many students with real industry-like experience in those courses.

References

1. Bailey, J. and Stroud, C. (2002) "Embedded Logic Analyzers for On-Line System Analysis," *Proceedings International Conference on Information Systems Analysis and Synthesis*, Orlando, Florida.

2. Stroud, C., Karunaratna, P. and Bradley, E. (1997) "Digital Components for Built-In Self-Test of Analog Circuits," *Proceedings IEEE International Application Specific Integrated Circuits Conference*, Portland, Oregon.

3. Lewis, B., Lim, S., Puckett, R. and Stroud, C. (1999) "A Prototype Unit for Built-In Self-Test of Analog Circuits," Proceedings IEEE Southeast Regional Conference, Lexington, Kentucky.

4. Maggard, K. and Stroud, C. (1999) "Built-In Self-Test for Analog Circuits in Mixed-Signal Systems," Proceedings IEEE Southeast Regional Conference, Lexington, Kentucky.

5. Stroud, C., Morton, J., Islam, A., and Alassaly, H. (2003) "A Mixed-Signal Built-In Self-Test Approach for Analog Circuits," *Proceedings IEEE Southwest Symposium on Mixed-Signal Design*, Las Vegas, Nevada.

6. Fields, T. and Stroud, C. (1995) "A Test Machine for Digital VLSI Circuits Fabricated Through MOSIS," *Proceedings IEEE Automatic Test Conference*, Atlanta, Georgia.

Charles E. Stroud

Charles E. Stroud received his B.S. and M.S. degrees in Electrical Engineering from the Univ. of Kentucky in 1976 and 1977, respectively. He then joined AT&T Bell Laboratories in Naperville, IL, where he worked for fifteen years in digital printed circuit board and VLSI design, Computer-Aided Design (CAD) tool development, and digital VLSI testing and diagnosis. He designed twenty-one VLSI devices and three printed circuit boards for telephone switching systems and computers. He developed CAD tools for behavioral model synthesis of VLSI devices and CPLD/FPGA based circuit boards, automatic implementation of Built-In Self-Test approaches in VLSI devices, and timing and fault simulation and analysis. He received the AT&T Bell Laboratories Distinguished Member of Technical Staff Award in 1987. While working at Bell Labs, he received his Ph.D. in Electrical Engineering & Computer Science from the Univ. of Illinois at Chicago in 1991. He left Bell Labs in 1993 to begin an academic career where he served for seven years as an Associate Professor in the Dept. of Electrical Engineering at the University of Kentucky, three years as a Professor in the Dept. of Electrical and Computer Engineering at the University of North Carolina at Charlotte, and is currently a Professor in the Dept. of Electrical and Computer Engineering at Auburn University. Charles is a member of Eta Kappa Nu, Tau Beta Pi, and a Senior Member of IEEE. He has served on the Technical Program Committees for IEEE International Test Conference (three years as BIST topic coordinator), ACM/IEEE International Workshop on Hardware-Software Co-Design (two years as Publications Chair), IEEE International ASIC Conference (three years), IEEE International On-Line Test Symposium (five years), and the ACM International Symposium on FPGAs (one year). He has served as the Design for Testability Editor for IEEE Design & Test of Computers (four years) and as Associate Editor for IEEE Transactions on VLSI Systems (two years). He is author of the recent book entitled "A Designer's Guide to Built-In Self-Test" and has authored/co-authored over 100 journal and conference papers, primarily in the area of design and test of VLSI and FPGA devices and systems. He has been awarded sixteen research grants (totaling $3.5 million) in the areas of design and test of VLSI and FPGA devices and systems from DARPA, NSA, NSF, US Air Force, US Army, AT&T, Lucent Technologies, Agere Systems, and Cypress Semiconductor. He has received six teaching awards (five department level awards and one college level award) and has received Best Paper Awards at the 1988 ACM/IEEE Design Automation Conference, the 2001 IEEE Automatic Test Conference, and the 2002 International Systemics, Cybernetics, and Informatics Conference. He holds thirteen U.S. patents for various BIST approaches for VLSI and FPGAs with five more pending.