AC 2008-2502: UNIFYING LABORATORY CONTENT OF A DIGITAL SYSTEMS AND COMPUTER ARCHITECTURE CURRICULUM THROUGH HORIZONTAL AND VERTICAL INTEGRATION

Steve Naumov, Purdue University Calumet

Steve Naumov graduated in 2007 with highest distinction from Purdue University Calumet with a B.S. in Computer Engineering and minor in applied mathematics. He intends on pursuing a Ph.D. in electrical engineering from the University of Wisconsin – Madison. His research interests include high performance computer architecture, digital system verification, and computer architecture education. Along with initiating the accomplishments described in this paper, he has held two consecutive internships at Intel Corp. as a validation engineer. Contact him at naumov82@gmail.com.

William Obermeyer, Purdue University Calumet

William J. Obermeyer is an undergraduate computer science student at Purdue University Calumet and anticipates graduating with highest distinction in May 2008 with a Bachelor's of Science in Computer Science, Associates of Arts in History, and minor in applied mathematics. He intends on obtaining a graduate degree in mathematics from Purdue University – Calumet to pursue his passions of education with a career in academia. Additionally, William has over ten years software development and engineering experience. Contact him at obermeyer@calumet.purdue.edu.

Rahul Singhal, Purdue University Calumet

Rahul Singhal is pursuing a Master's of Science in Electrical Engineering at Purdue University Calumet and intends on continuing studies at the University of Wisconsin – Madison. His research interests include high performance microprocessors and energy-efficient digital systems. Rahul has held two internships, one with Freescale Semiconductor and the other with Intel Corporation. Rahul graduated with highest distinction from Purdue University Calumet with a Bachelor's of Science in Electrical Engineering in December 2006. Contact him at r.singhal17@gmail.com.

Eduardo Garcia, Purdue University Calumet

Eduardo Garcia is employed as controls and automation engineer for ArcelorMittal Steel Corporation in Portage, Indiana. He was a recipient of the Indiana Louis Stokes Alliance for Minority Participation (LSAMP) Fellowship award in the summer of 2003. Eduardo graduated from Purdue University Calumet with a degree in electrical engineering with a minor in applied mathematics. Contact him at garcia.eddie@gmail.com.

Nasser Houshangi, Purdue University Calumet

Dr. Nasser Houshangi is professor of electrical and computer engineering at Purdue University Calumet. His research interests include robotics, intelligent control, and multi-sensor integration. His teaching interests include microprocessor and computer architecture, industrial automation, adaptive control, and robotics. Nasser received a doctoral degree in electrical engineering from Purdue University in 1990. Contact him at hnasser@calumet.purdue.edu.

Unifying Laboratory Content of a Digital Systems and Computer Architecture Curriculum through Horizontal and Vertical Integration

Abstract

This paper describes the application of horizontal and vertical integration to unify the digital systems and computer architecture curriculum for the Department of Electrical and Computer Engineering at Purdue University Calumet. An enhanced set of twelve laboratory assignments and five design projects resulted from performing the integration. Horizontal integration was achieved by providing a consistent presentation of concepts across two computer architecture laboratory courses while simultaneously providing students the necessary skill-set for developing a successful career as a computer engineer. Vertical integration was achieved by interweaving common technical theories and skills to establish interdependence among all digital system and computer architecture laboratory coursework. The restructured laboratory sequence provides a cohesive educational experience and significant exposure to concepts, design methodologies, and software tools ubiquitous in the semiconductor and computer industry.

1. Introduction

Three digital systems and computer architecture courses are administered sequentially by the Department of Electrical and Computer Engineering at Purdue University - Calumet. The first course, ECE 370: Digital Systems – Logic Design, is a three credit hour course with a two hour lecture component and a three-hour laboratory component. The course introduces students to combinational and sequential logic design principles through the use of a hardware description language (HDL) and reconfigurable hardware. The second course, ECE 371: Microprocessor Systems, is a three credit-hour course organized into a two-hour lecture and one, three-hour laboratory session. This course introduces students to the fundamentals of computer organization and design. The third course, ECE 464: Computer Architecture and Organization, is a four credit-hour course with a three-hour lecture and one, three-hour laboratory section. This course builds on the fundamental computer organization and design concepts taught in ECE 371 by examining advanced concepts in computer architecture.

A proactive undergraduate student who completed the three course sequence observed a stable platform provided by the ECE 370 laboratory course, however, he noticed a disjoint learning experience for the two computer architecture laboratory courses. Examining the structure, approach, concepts, and tools used to administer the laboratory sections for both courses indicated a need for improvement. A more effective ECE 371 laboratory would incorporate the use of an architectural simulator and reconfigurable hardware, establish an increased emphasis on dataflow and structural digital system modeling, and expand the instruction set support of the RISC microprocessor designed in the laboratory course. The observations revealed greater concerns with the goals of the ECE 464 laboratory content. Deviating from lecture, the requirement of designing an IEEE 802.3 network repeater created a difficult learning experience for students enrolled in ECE 464. Through the Department of Electrical and Computer Engineering senior design course, a team was independently assembled to formulate and implement solutions to improve the two computer architecture laboratory sections of the three course computer engineering sequence focusing on hardware.

This paper describes content developed for the laboratory which requires students to complete an enhanced set of twelve laboratory assignments and five design projects. Figure 1 illustrates the application of a horizontal and vertical integration philosophy to unify the digital systems and computer architecture laboratory curriculum.

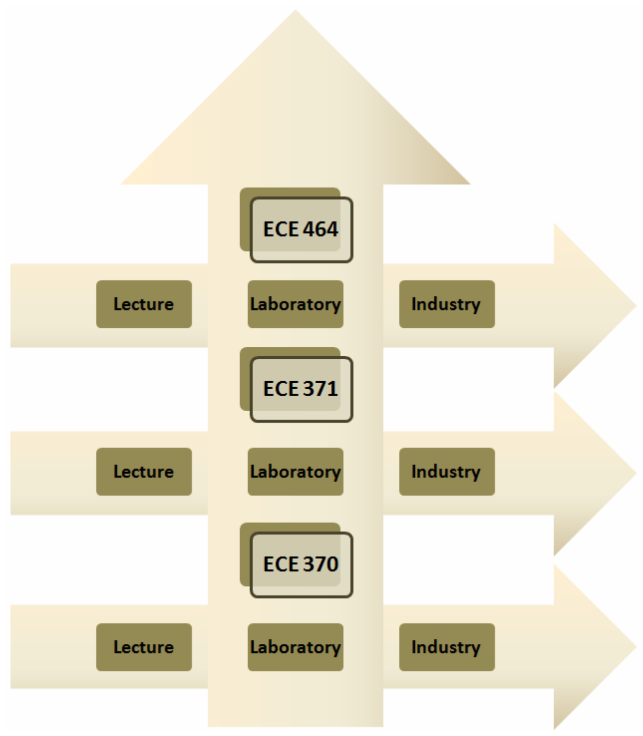

Figure 1: Horizontal and Vertical Integration Applied to the Digital Design and Computer Architecture Computer Engineering Curriculum

In broadening the customary interpretation, horizontal integration is achieved by a consistent presentation of topics across lecture and laboratory sessions and an alignment with industry expectations. A consistent lecture and laboratory presentation allows for a focus on strengthening the skill-set of the student in preparation for a successful engineering career. This can be accomplished through significant exposure to concepts, design methodologies, and software tools that are ubiquitous in the semiconductor and computer industry. Vertical integration is accomplished through interweaving common technical theories and skills and coordinating ECE 371 and ECE 464 laboratory sessions with ECE 370. The ECE 370 laboratory pedagogy provides an ideal platform to establish interdependency among the three courses.

The sections that follow describe the tools employed to complete the prescribed laboratories. Additionally, a philosophy, schedule, brief overview, and evaluation of both laboratory courses are provided. However, the scope of this paper does not provide the proper forum to deliver the details of each laboratory experiment or project. Thus, for comprehensive insight into the details of the laboratory courses, please contact the primary author of this paper or visit http://www.calumet.purdue.edu/ece/Houshangi-new.html for a link to the web pages for both ECE 371 and ECE 464.

2. Tools Employed in the Laboratories

The MIPS R2000 ISA was chosen to administer the laboratory assignments to complement the material presented in the required textbooks for ECE 371[1] and ECE 464[2]. The list of hardware and software tools chosen to facilitate ECE 371 and ECE 464 based on the MIPS R2000 ISA selection consist of an architectural simulator, hardware description language (HDL), HDL development environment, FPGA development platform, and software tools applied for verification. Table 1 indicates the specific materials and tools selected to facilitate the laboratory experiments and projects for both courses based on the categories mentioned above.

Category	**Tools Selected**
Instruction Set Architecture(ISA)	MIPS R2000
Architectural Simulator	MIPS Assembler & Runtime Simulator (MARS)
Hardware Description Language (HDL)	VHDL
HDL Development Environment	Altera Quartus II
FPGA Development Platform	Altera DE2 Development and Education Board
Verification Software Tools	SPIM Simulator and *mifWrite*

Table 1: Materials and Tools Selected to Facilitate the Laboratory Courses

The MIPS Assembler and Runtime Simulator (MARS)[3] was chosen to explore the MIPS R2000 ISA. MARS possesses a functional user interface, instruction-set extensibility, and is available through Internet download as regularly maintained, open-source software.

Maintaining the use of VHDL and the Altera Quartus II CAD toolset were natural choices to facilitate both laboratory courses because they are the primary tools utilized in ECE 370. Students have regular access to four computer laboratories equipped with licensed versions of the Altera Quartus II software package. Alternatively, students can download a six-month free license of Altera's Quartus II Web-Edition[4] software. Additionally, the department upgraded their use of the Altera UP-2 Development Board to Altera's latest educational FPGA platform, the DE2 Development and Education Board. The Cyclone II FPGA along with the DE2 board provides additional hardware resources and functionality.

To facilitate system level verification across both laboratory sections, the SPIM[5] architectural simulator (another MIPS simulator) was utilized in conjunction with a locally modified version of the *mifwrite*[6] software tool

3. Microprocessor Systems Laboratory (ECE 371)

The new ECE 371 laboratory schedule is comprised of twelve enhanced laboratory assignments that span fifteen weeks. While the ECE 371 lecture content represents a first exposure to design concepts, the laboratory experiments place a higher degree of emphasis on implementation to incrementally enhance students' design capabilities. Therefore, students are provided the majority of schematic diagrams and explanations necessary to complete the required tasks. In applying this pedagogical approach, the academic goals for ECE 371 have been reformulated as indicated below.

- Utilize an architectural simulator to design, implement, and debug small assembly language programs to gain understanding of the MIPS ISA
- Define and apply dataflow, structural, and behavioral modeling to implement fundamental logic circuits utilized in microprocessor microarchitectures
- Implement, verify, synthesize, and program complex digital systems utilized in microprocessor microarchitectures onto reconfigurable hardware
- Implement and verify a five-stage scalar pipelined RISC microprocessor microarchitecture utilizing a bottom-up engineering design methodology

To support these goals, the laboratory schedule shown in Table 2 for ECE 371 was developed.

Lab	**Duration**	**Objectives**
1	1 Week	*introduce MARS and MIPS instruction encoding, format, types, registers and conventions, and arithmetic/logic related programs*
2	1 Week	*design small programs using memory reference, control flow, pseudo-instructions, and subroutines*
3	1 Week	*introduce VHDL modeling techniques via a 2-to-1 mux, 2-to-4 decoder, and modified full adder*
4	2 Weeks	*implement generic muxes, decoders, and a look-ahead carry unit*
5	1 Week	*implement generic comparator and sign-extension unit. Also, implement an 8-bit carry look-ahead adder and hex-to-7-segment decoder*
6	2 Weeks	*design 32-bit arithmetic logic unit*
7	1 Week	*synthesize an 8-bit arithmetic logic unit on Altera DE2 board*
8	1 Week	*implement D flip-flop, generic register, register file, ROM and RAM*
9	1 Week	*synthesize 8x8-bit register file on DE2 board*
10	1 Week	*implement fetch, decode, and execute stages*
11	1 Week	*implement memory and write-back stages, and pipelined control unit*
12	2 Weeks	*integrate pipe stages and perform microprocessor verification*

Table 2: The ECE 371 Laboratory Schedule and Objectives

Laboratory 1 – Laboratory 2

Laboratory assignments one and two utilize a workbook developed by the authors of this paper to familiarize students with the MIPS R2000 ISA using MARS. This workbook-based approach allows students to explore MARS and MIPS throughout both assignments. Students investigate MIPS instruction encodings, formats, data types, the register set, register naming conventions, addressing modes, subroutines, and pseudo-instructions. Additionally, students write small programs that exercise instructions which span the three major classifications: arithmetic/logical, memory reference, and control flow. Overall emphasis is placed on the subset of instructions supported by the microprocessor designed throughout both laboratory courses.

Laboratory 3 – Laboratory 5

Laboratory assignments three, four, and five accomplish two goals. First, the experiments offer a comprehensive review of digital systems modeling in VHDL. Initially, dataflow, structural, and behavioral architectural models are examined. However, since behavioral modeling hides crucial gate-level implementation details that facilitate learning, students are required to use dataflow and structural modeling when implementing their designs throughout the semester. The second goal of these experiments is to enable students to implement and unit-test the fundamental combinational logic circuits required to construct their microprocessor.

To accomplish these goals, students implement a 2-to-1 multiplexer, full adder, and 2-to-4 binary decoder using a dataflow architecture model for laboratory assignment three. Assignment four builds on these three components to hierarchically and parametrically implement larger multiplexers and decoders. Additionally, students use the full adder as the basis to hierarchically implement an 8-bit carry look-ahead addition subtraction unit. Finally, laboratory five requires the implementation of a parametric signed comparator, sign/zero extension unit, and seven-segment decoder/driver. These implementation goals allow for a focus on hierarchically integrating and verifying the complete microprocessor datapath and control during the later part of the semester, facilitating a bottom-up engineering design flow.

Laboratory 6

With comprehensive VHDL experience, students are ready to progress to laboratory assignments six and seven. Laboratory assignment six requires the design of the first major microprocessor datapath component, the arithmetic logic unit (ALU). The ALU supports ten, 32-bit operations, outputs three status flags (`V, C, Z`), and is decomposed into three major sub-components. The Boolean Logic Unit (BLU) is parametrically designed and provides support for `AND`, `OR`, `XOR`, and `NOR` bitwise logical operations. The Add/Subtract Unit (ASU) implements a carry look-ahead architecture and facilitates the required addition and subtraction operations of the ALU. Finally, the Logarithmic Shift Unit (LSU) supports all logical and arithmetic shift operations and has a logarithmic architecture.

Laboratory 7

Laboratory assignment seven is the first of two assignments that provide students tangible experience in hardware prototyping using a modern reconfigurable hardware platform. Using the Altera DE2 Board, students synthesize and program an 8-bit ALU, as shown in Figure 2, onto the Cyclone II FPGA.

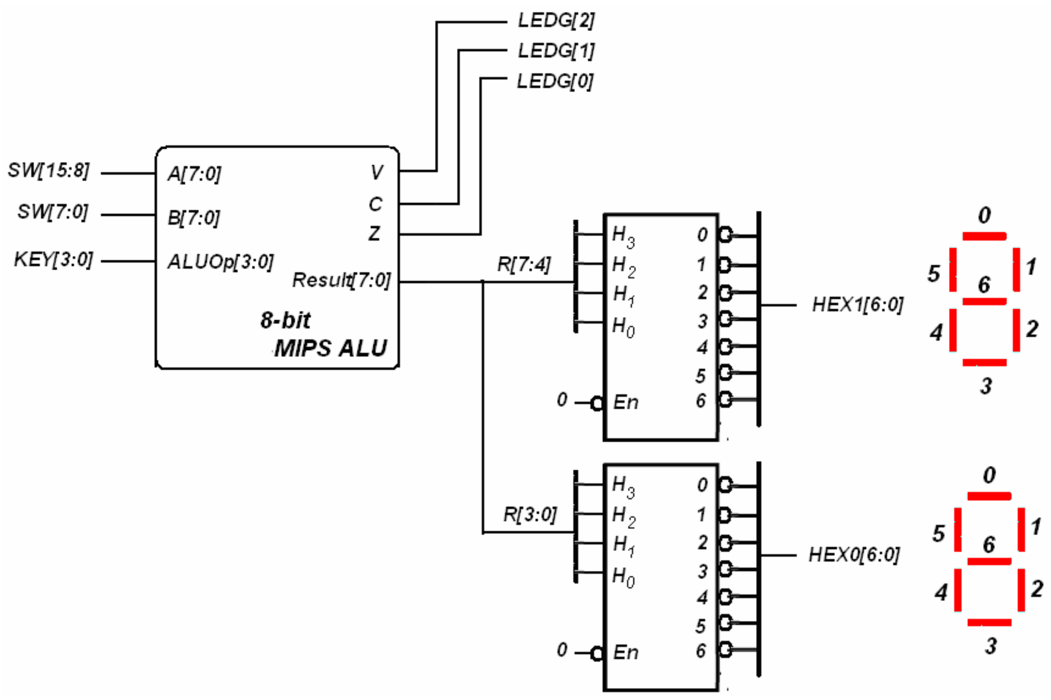

Figure 2: The 8-bit Arithmetic Logic Unit (ALU) FPGA System

The 8-bit ALU maintains the organization and functionality of the 32-bit ALU implemented in assignment six, however, its datapath is scaled down to provide an easier I/O interfacing experience with the Altera DE2 board. The ALU interfaces with the user through the DIP switches, debounced push buttons, and seven-segment display units on the DE2 Board. In addition to the interfacing and device programming, students evaluate and optimize the ALU's area and timing characteristics.

Laboratory 8

Laboratory assignment eight changes the focus from combinational to sequential circuit design, primarily emphasizing the implementation of sequential circuits and storage elements necessary to construct the microprocessor datapath. Students begin by implementing a D flip-flop, use the D flip-flop to implement a generic, parallel-load register, and then utilize both components along with several multiplexers and decoders to implement the architectural register file (ARF). The ARF is similar to the one described by Patterson & Hennessy[1] and supports thirty-two, 32-bit registers that are read asynchronously and written synchronously on the positive edge.

After implementing the ARF, students use Altera's altsyncram mega-function in association with the Quartus II Mega-Function Wizard Tool to implement the instruction ROM (I-ROM) and data RAM (D-RAM). The altsyncram mega-function provides students with parameterized true dual-port read/write RAM and ROM components for the Cyclone II FPGA. The properties of the I-

ROM and D-RAM components implemented by the students and utilized in the microprocessor datapath are shown in Table 3.

	Width	Width	Address	Capacity	Clocks	Inputs	Outputs
I-ROM	32-bits	32-bits	8-bits	256 Words	1 (Read)	Registered	Registered
D-RAM	32-bits	32-bits	8-bits	256 Words	1 (R/W)	Unregistered	Unregistered

Table 3: Altera Megafunction Properties for the Instruction and Data Memories

<u>Laboratory 9</u>

Providing students additional experience in synthesis and FPGA programming, laboratory assignment nine specifies an ARF, as shown in Figure 3, suitable to program and interface onto the Cyclone II device.

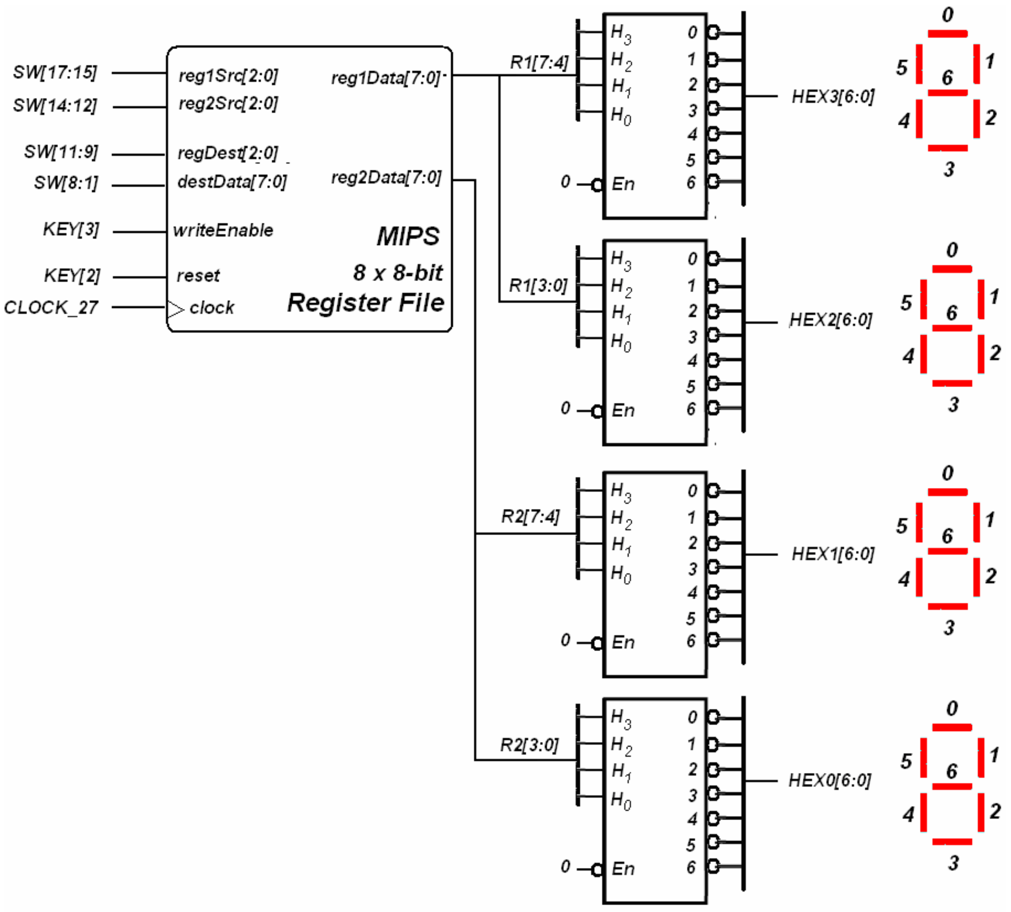

Figure 3: The Architectural Register File (ARF) FPGA System

Like the 8-bit ALU FPGA system of assignment seven, this ARF is a scaled down replica of the 32 x 32-bit ARF designed in laboratory eight that also interfaces to the debounced push buttons, DIP switches, and seven-segment displays. Unlike the ALU, synthesizing and programming the ARF onto the FPGA gives students exposure to programming a synchronous digital system.

This laboratory specifies that students use the on-board 27MHz crystal oscillator for the system clock. Similar to laboratory seven, students evaluate and optimize the ARF's area and timing characteristics. Unlike laboratory seven, however, students focus on maximizing the clock frequency of the ARF system since it is a sequential digital system.

Laboratory 10 – Laboratory 12

Students conclude the fifteen week ECE 371 laboratory course by completing assignments ten through twelve. These assignments require students to design, implement and verify the datapath and control units for a five-stage (instruction fetch, instruction decode, execution, memory, and write-back) scalar pipelined MIPS microprocessor similar to the one described in the Patterson & Hennessey[1] textbook. To assemble the datapath, students utilize all of the designs implemented throughout the semester. The resulting microarchitecture of the microprocessor implements a scalar five-stage pipeline, a Harvard style idealized (no caches) memory organization, a hardwired (combinational) control unit, and a one-cycle delay slot for all branch and jump instructions. Additionally, the microprocessor supports an instruction set comprising twenty integer instructions from the MIPS R2000 ISA as indicated in Table 4.

add	and	beq	j	nor	ori	slt	srav	sub	xor
addi	andi	bne	lw	or	sllv	slti	srlv	sw	xori

Table 4: The MIPS R2000 ISA Subset Supported by the ECE 371 Microprocessor Datapath

Two modifications to the standard MIPS R2000 ISA were made to reduce the complexity of the hardware and facilitate synthesis on the Cyclone II device. The first modification reduced the virtual address size of the MIPS R2000 ISA from 32-bits to 8-bits. The second modification converted all byte offsets to word offsets to complement the word addressability of the I-ROM and D-RAM components.

Verification of the entire microprocessor system was accomplished through the use of an assembly language test-bench suite. The test-bench suite is comprised of several MIPS programs that fully stress the microprocessor microarchitecture. The programs attempt to read and write to all thirty-two registers and exercise the twenty supported instructions. The goal of the testbench suite was not to perform a logical task, but rather to perform a series of operations whose result can be observed in D-RAM memory. To apply the test-suite to the simulation, the required Altera memory initialization files (.MIF) for the I-ROM and I-RAM memories of the microprocessor were produced by SPIM and *mifWrite*.

4. Computer Architecture and Organization Laboratory (ECE 464)

The new laboratory curriculum for ECE 464 consists of five design projects that build upon the ECE 371 laboratory goals. Contrasting ECE 371's laboratory teaching philosophy, the ECE 464 laboratory provides students the forum to assume a higher degree of design responsibility. Having acquired the necessary implementation skills, students are expected to traverse the entire digital systems design flow in order to successfully complete each project. Using this designdirected teaching style, the newly defined academic goals for the ECE 464 laboratory section are shown below.

- Apply behavioral and register-transfer level (RTL) digital system modeling in the context of larger, more complex digital systems found on modern microprocessors
- Expose students to the design complexities that arise when implementing a comprehensive instruction set
- Provide practical experience with the implementation details associated in designing digital systems that implement advanced microarchitectural algorithms
- Expose students to the logic required to implement fundamental microarchitectural concepts that inhibit pipelined processor performance

To support these goals, the laboratory schedule shown in Table 5 was developed for ECE 464.

Project	**Duration**	**Objective**
1	3 Weeks	*increase instruction support to thirty-one instructions*
2	3 Weeks	*forwarding and pipeline interlocks*
3	3 Weeks	*implementation, verification, and synthesis of final microprocessor*
4	3 Weeks	*2-way set-associative, pipelined, least-recently-used instruction cache*
5	3 Weeks	*generic 2–level adaptive branch predictor/branch target buffer*

Table 5: The ECE 464 Laboratory Schedule and Objectives

Design Project 1

Design project one provides students the opportunity to review the microprocessor implementation completed in the ECE 371 laboratory. This project requires students to include support for an additional eleven MIPS R2000 integer instructions into the base ECE 371 datapath implementation. The results of this project increase the total number of instructions supported by the microprocessor to thirty one, as indicated in Table 7.

add	beq	bltz*	jalr*	nor	sllv	srav	sw
addi	bgez*	bne	jr*	or	slt	srl*	xor
and	bgtz*	j	lui*	ori	slti	srlv	xori
andi	blez*	jal*	lw	sll*	sra*	sub	

* Indicates additional instructions incorporated into ECE 464 microprocessor datapath

Table 6: Thirty-One MIPS R2000 Integer Instructions as the ISA Subset

To accomplish project one, students must incorporate additional components into the datapath as well as modify the main control unit (MCU) and the pipelined control unit (PCU), two components initially implemented in laboratory assignment ten of ECE 371. The PCU synchronizes the control signals generated by the MCU and delivers them to each pipeline stage in lockstep with each instruction. The MCU and PCU generate control signals that are routed to the datapath based on the current state of the pipeline.

Design Project 2

Design project two provides students with a practical understanding of the fundamental techniques utilized to improve microprocessor performance. Students design, implement, and verify the forwarding logic unit (FLU) and hazard detection unit (HDU). The FLU monitors the state of the instructions within each pipeline stage, detects read-after write (RAW) data dependencies, and generates the select signals of the two bypass multiplexers shown in Figure 4.

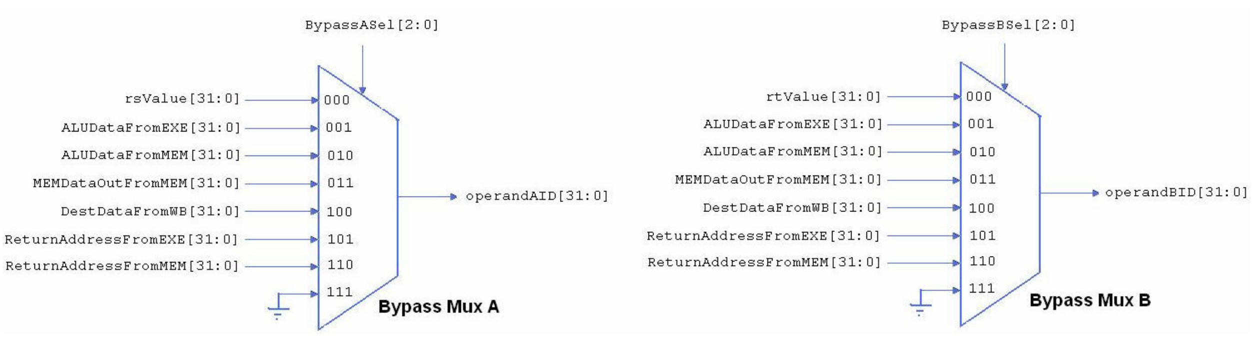

Figure 4: Datapath of the Bypass Multiplexers in the Instruction Decode Pipeline Stage

The FLU, in combination with the bypass multiplexers in the instruction decode pipeline stage, selects the youngest operands provided to the ALU, thus eliminating pipeline stalls and therefore increasing performance. Similar to the FLU, the HDU detects and resolves RAW hazards in the microprocessor pipeline. The HDU detects a load instruction in the execution pipeline stage and a consumer instruction in the instruction-decode stage. When this condition occurs, the HDU disables the instruction-fetch pipeline register for one cycle and deactivates all control signals that modify microprocessor state.

Design Project 3

The third design project requires students to integrate, implement, and verify the enhancements made in the first two design projects. Completing the three design projects through the first nine weeks of the semester results in a microprocessor microarchitecture that supports thirty-one integer MIPS R2000 instructions, implements a scalar five-stage pipeline, contains idealized (no caches) Harvard style memories, a hardwired (combinational) control unit, a one-cycle delay slot for all branch and jump instructions, supports subroutine calls, incorporates full operand bypassing, and implements pipeline interlocks. Overall, students gain a more intimate understanding of the intricacies in the microprocessor pipeline while incorporating performance-enhancing features. Additionally, after completing the first three design projects, students broaden their design experience by relying on their increased implementation skills gained from completing ECE 370 and ECE 371.

Design Project 4

Design project four requires students to design a blocking two-way set associative pipelined instruction cache with a least-recently-used (LRU) replacement strategy. Each cache way contains three arrays: the tag array, the valid-bit array, and the data array. Additionally, an LRU array tracks the least-recently accessed way within a particular set. All the arrays are implemented using the Altera altsyncram Mega-Function and are configured as dual-port RAMs. The read/write sequencing is done by a simple idealized cache controller module implemented as a Mealy finite state machine. Table 7 shows the instruction cache parameters specified for design project four.

Instruction Address Width	16 bits
Number of Sets	32
Cache Line Size	64 bits (2, 32-bit MIPS Instructions)
Word Offset Size	1 bit
Tag Size	10 bits
Index Size	5 bits
LRU Bits	1
Total Cache Size (only data)	4,096 bits (512 Bytes)
Total Cache Size	4,832 bits (604 Bytes)

Table 7: Pipelined Instruction Cache Parameters

This project provides students with an opportunity to practically extend the cache memory concepts which were introduced in ECE 371 and further investigated in ECE 464 lecture. Enhancing the performance of the cache through pipelining stresses that this technique is not only used to improve performance of microprocessors but generally applicable when designing high performance digital systems. Additionally, pipelining the cache gives students insight into possible methods that enhance instruction fetch throughput when increasing the depth of the pipeline. Lastly, the pipelined cache provides a serious exposure to signal synchronization issues in designing systems that are heavily pipelined.

Design Project 5

The last design project specifies that students design a parameterized two-level adaptive branch prediction system[7] utilizing a pattern history table (PHT), a branch history shift register (BHSR), saturating counters, and a direct-mapped branch target buffer[8] (BTB). The system parameters and default values are summarized in Table 8.

Parameter	**Default**	**Description**
`INSTR_ADDRESS_WIDTH`	8	Number of bits used for the PC
`BTB_PHT_ADDRESS_WIDTH`	4	Number of bits used to access the BTB and PHTs
`NUM_PREDICTION_BITS`	2	Number of bits for the *n-bit* saturating counters
`NUM_GLOBAL_HISTORY_BITS`	2	Number of global history bits for the BHSR

Table 8: Branch Predictor Generic Parameters and Default Values

The branch predictor design project provides a good balance of design complexity for the undergraduate level student. Similar to cache design concepts, dynamic branch prediction forms a major part of the ECE 464 lecture content. Designing a branch predictor exposes students to advanced techniques utilized in modern microprocessors for increasing performance.

Parametrically defining the branch predictor allows students to analyze the prediction algorithm through instantiating multiple instances of the system and investigating the prediction accuracy and performance tradeoffs.

5. Evaluation and Future Considerations

The computer architecture laboratory sequence discussed in this paper has been accepted as the official curriculum by the Department of Electrical and Computer Engineering and has been administered for one academic year. Unifying the digital systems and computer architecture laboratory courses with the approach described in this paper has been proven successful[9], which validates the positive feedback provided by students that have completed ECE 371 and ECE 464. Though based on a limited sample space, student commentary indicated a higher appreciation of lecture topics derived from the comprehensive laboratory schedule. A more accurate quality assessment will be provided through student surveys and focus groups as of the spring 2008 semester. Data will be gathered and analyzed to assess the effectiveness of the curriculum as a much larger and more diverse set of students complete the new laboratory curriculum and provide their feedback.

The experiments and projects for the ECE 371 and ECE 464 laboratories should be considered dynamic in order to maintain a contemporary computer architecture laboratory curriculum. This philosophy suggests implementing technological enhancements such as migrating to the use of the more industry-prevalent Verilog as opposed to VHDL for administering the projects in both laboratory courses.

The first year of teaching experience provided insight into possible scheduling options. One option is to reposition the first two experiments of the ECE 371 laboratory into the lecture course as homework assignments. This would lighten the ECE 371 laboratory schedule of twelve projects in fifteen weeks and provide three alternatives to occupy the newly available time. The first alternative could allow a review of additional digital systems design concepts like the modeling of finite state machines in the chosen HDL. Secondly, these adjustments could allow time to incorporate logic into the ECE 371 microprocessor to interface to the Altera DE2 board, thus offering students additional device programming and synthesis experience. Finally, the first three design projects of ECE 464 could be moved into ECE 371 to provide for further laboratory curriculum flexibility.

The alterations mentioned above regarding the ECE 464 laboratory schedule allow for the exploration of more advanced microarchitectural concepts and algorithms. Areas of focus can include a study of deep, superscalar, and out-of-order pipelines and their associated hardware structures. Projects can be developed that require students to design and implement components such as a register renaming unit, reservation stations, or even a reorder buffer. Additionally, ECE 464 can more comprehensively incorporate the use of the Altera DE2 board by having students experiment with the interfacing of their microprocessor with more elaborate I/O devices such as VGA, keyboard, and mouse as a means for exploring interrupts and exception handling.

6. Conclusion

A disjoint learning experience was observed in the two course computer architecture laboratory curriculum offered at Purdue University - Calumet. The efforts of the undergraduate senior design team represented in this paper resulted in an enhanced set of twelve laboratory assignments and five new design projects. These projects comprehensively reinforce digital systems and computer architecture concepts through use of industry-standard software tools in the design, implementation, verification, and synthesis of a RISC microprocessor, pipelined instruction cache, and parametric branch predictor. Utilizing horizontal and vertical integration, the resulting curriculum collectively unifies lecture and laboratory content of the three course digital design and computer architecture sequence, creates interdependency through the computer engineering coursework, and increases student employability within the semiconductor and computer industry. In preserving the intent of this paper, the technical nature of this curricula indicates the content be continuously reviewed for effectiveness with an open mind toward revision in the future.

Bibliography

1. Patterson, David A., Hennessy, John L. *Computer Organization and Design: The Hardware/Software Interface*. 3rd Ed (Revised). Morgan Kaufmann, Inc., San Francisco, CA, 2007.
2. Patterson, David A., Hennessy, John L. *Computer Architecture: A Quantitative Approach*. 4th Edition. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2007.
3. Vollmar, K., Sanderson, P: *MARS: An Education-Oriented MIPS Assembly Language Simulator*, ACM SIGCSE Bulletin, v.38 n.1, March 2006.
4. *Download Center*. Altera Corp. available at: https://www.altera.com/support/software/download/sof-download_center.html. [accessed: 01/11/2008].
5. Larus, James. *SPIM: A MIPS32 Simulator*. Computer Science Department, University of Wisconsin –Madison. available at: http://www.cs.wisc.edu/~larus /spim.html [accessed: 11/25/2006].
6. Loomis, John. *ECE 595c Notes*. Electrical and Computer Engineering Department, University of Dayton. available at: http://www.johnloomis.org/ece595c/notes/notes.html. [accessed: 01/11/2008].
7. Yeh, T.Y., Y.N. Patt: *Two-Level Adaptive Training Branch Prediction*. Proceedings, 24th Annual International Symposium on Microarchitecture, 1991, Pages. 51- 61.
8. Perleberg, C.H.; Smith, A.J.; *Branch Target Buffer Design and Optimization*, *IEEE Transactions on Computers*, Volume 42, Issue 4, April 1993. Pages: 396 – 412.
9. Thompson, T., Herr, D., Brown, S., Traylor, R., Fiez, T.; *Educational Design, Evaluation, & Development of Platforms for Learning*, 34th ASEE/IEEE Frontiers In Education, 2004.