Cybersecurity Education: Bridging the Gap Between Hardware and Software Domains

Marcin Lukowiak*, Stanisław Radziszowski†, Jim Vallino‡,
and Christopher Wood†‡
*Department of Computer Engineering, †Department of Computer Science, ‡Department of Software Engineering, Rochester Institute of Technology, Rochester, NY 14623
Email: Marcin.Lukowiak@rit.edu

***Abstract*—With the continuous growth of cyberinfrastructure throughout modern society, the need for secure computing and communication is more important than ever before. As a result, there is also an increasing need for entry-level developers who are capable of designing and building practical solutions for systems with stringent security requirements. This calls for careful attention to algorithm choice and implementation method, as well as trade-offs between hardware and software implementations. This paper describes motivation and efforts taken by three departments at Rochester Institute of Technology (Computer Engineering, Computer Science and Software Engineering) that were focused on creating a multi-disciplinary course that integrates the algorithmic, engineering, and practical aspects of security as exemplified by applied cryptography. In particular, the paper presents the structure of this new course, covered topics, lab tools and results from the first two spring quarter offerings in 2011 and 2012.**

***Index Terms*—Security-Oriented Curriculum, Cybersecurity Education, Multi-Disciplinary Applied Cryptography, Hardware and Software Design**

I. Introduction

With the growth and pervasiveness of cyberinfrastructure in modern society, secure computing and communication have become critically important. Sample applications with important security requirements include e-commerce, voice/video communications, secure databases, and financial market transactions. In addition, there is a growing trend toward integrating security into many different points of the information technology infrastructure, such as embedding in disk drives, processors (e.g., built-in encryption/decryption), trusted system boards, network switching elements, mobile devices, and sensors. All of these points require careful attention to algorithm choice and implementation method, and trade-offs between software and hardware. The development of such systems requires a population of entry-level developers who have the knowledge and skills needed to design them. One of the key elements of this is an understanding of cryptographic algorithms and their implementation.

In a typical approach, the study of cryptography and security in undergraduate computing programs focuses primarily on the mathematical aspects of cryptography, which unfortunately gives students fundamental theory but not practical implementation experience. To gain the necessary knowledge and skills, a student must learn concepts from the multiple disciplines of computer engineering, computer science, and software engineering, as each of them play a significant role in different aspects of secure computing and communications, which is schematically presented in Figure 1. This is not done in standard undergraduate computing curricula. Computer engineering students usually learn how to design general purpose digital systems, but they lack the knowledge related to the design of specialized cryptographic circuits and optimizations of hardware-software co-designs. Those students who may do some cryptographic design might construct a hardware implementation of a particular algorithm without knowing the fundamental theory on which it is based. Computer science and software engineering students study cryptographic algorithms mostly as a mathematical exercise with some software implementation. They rarely investigate the performance of their implementations and are usually not familiar with software optimizations or hardware implementations. In many applications, implementation aspects are crucial because of the complexity of cryptographic algorithms. This is especially true since they may operate upon data from a streaming

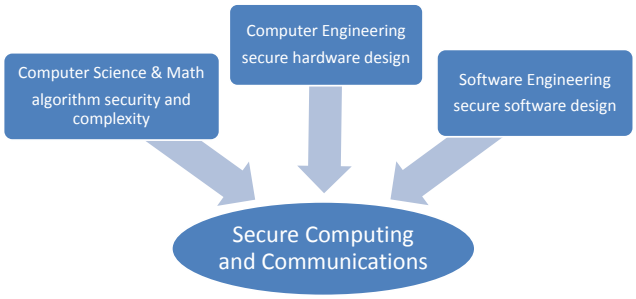

Fig. 1. The multi-disciplinary relationship that served as the basis for the new course draws from the fields of computer engineering, computer science, and software engineering.

communications channel, with a constrained power budget, or in an environment where side-channel attacks can be exploited to compromise the system.

Our review of the coursework available at other institutions found only a few examples of courses that have content which crosses disciplines, and none have the comprehensive, integrated approach. Two examples would be graduate-level courses in the Electrical and Computer Engineering Departments at Worcester Polytechnic Institute [1] and at Virginia Tech [2]. Realizing this, the authors of this paper developed and introduced a new multi-disciplinary course entitled "Hardware and Software Design for Cryptographic Applications".

The design of this course, focused on undergraduate students, emphasizes active learning pedagogy [3] [4] [5]. This pedagogy strives to immediately engage students with the material being covered through hands-on exercises during class time rather than being primarily lecture-based with other learning activities occurring only outside of the class. Within this course, team assignments and projects require implementations of selected cryptographic primitives. Such implementations are then followed by an in-depth comparison and contrast of various implementation alternatives, including software, custom Field Programmable Gate Array (FPGA) hardware, and hybrid hardware-software. This crosses three disciplines to span the range of skills from fundamental theory to practical software and hardware implementations.

This course uses the multi-disciplinary approach which Vallino and Czernikowski [6] [7] showed was effective for teaching courses in real-time and embedded systems. To the extent possible, student teams are composed of one computer engineering student and one software engineering or computer science student. Computer engineering students lead the hardware design portions of each project, while software engineering and computer science students lead the software development portions. The breadth of material dispersed throughout the course covers many different aspects of software and hardware development and revisits the mathematical theory behind their designs. Such structure provides students with ample opportunity to reinforce the fundamental concepts taught in lectures with hands-on exercises conducted in a studio setting. The course is organized to fit a quarter system with 10 weeks of classes.

This course emphasizes learning through hands-on exercises, with team assignments and projects requiring implementations of selected cryptographic primitives. Such implementations are then followed by an in-depth comparison and contrast of various implementation alternatives, including software, custom Field Programmable Gate Array (FPGA) hardware, and hybrid hardware-software. This allows to cross three disciplines to span the range of skills from fundamental theory to practical software and hardware implementations. Student teams in our course are ideally composed of one computer engineering student and one software engineering or computer science student. Computer engineering students lead the hardware design portions of each project, while software engineering and computer science students lead the software development portions.

Although many different cryptographic primitives were incorporated into the lectures, the primary focus was on the Advanced Encryption Standard (AES) and the SHA-3 hash function candidates. The AES was a clear candidate for study due to its status as the leading block cipher after its selection by NIST in 2001 [8]. Similarly, the SHA-3 hash function finalist candidates were investigated for the course project because of their importance for the cybersecurity community, and the large amount of readily available material associated with the NIST competition [9] [10]. The different trade-

offs that exist between the candidate functions provided students with a realistic perspective on the challenges faced when designing and implementing these algorithms.

The development platform was the Xilinx ML-507 evaluation board. It comes equipped with the Virtex-5 FXT FPGA device, which combines an embedded PowerPC 440 processor and reconfigurable fabric on a single chip. Like most modern FPGAs, this device is also capable of running soft-core processors within the FPGA resources. The whole platform was well-suited for all practical assignments due to the large number of available logic cells and external memory, both of which gave the students more freedom when experimenting with various software and hardware designs.

This paper first describes the structure of the new course. It then enumerates the sequence of in-class lectures and laboratory exercises, including selection criteria for course topics, tools, techniques, and development methodologies used to illustrate these concepts. Lastly, potential modifications are identified from the experiences developing the course, presenting the material in class, and the feedback received from the students who participated. This feedback is also presented and analyzed as a means of quantifying the course learning objectives and outcomes.

II. Course Structure

At a high level, there are four different areas of material covered in this course: 1) cryptographic foundations, 2) FPGA-based embedded systems, 3) embedded software development and optimization, and 4) hardware and hardware-software co-design and development.

The complete progression of the course for 10 weeks of classes with 4 contact hours per week is shown in Figure 2, where each week typically consists of one lecture and one lab assignment. This can be varied based on student prerequisites for the course and the outcome learning objectives.

A. Layered Progression

The ultimate goal of the course is to teach students about various aspects of applied cryptography using a layered approach, in which each layer contains more implementation abstractions and less algorithmic details than its successor. The base for this layered structure was defined by taking the different backgrounds of the target students into consideration. In our case, this meant that the course started with software-oriented exercises and gradually moved elements of these solutions into hardware. At each step, more algorithmic details are unveiled to the students as they experimented with different implementation techniques. This created a natural progression for the material taught in the course where each specific implementation drew from the design of its predecessor.

The following list presents the consecutive implementations of the core AES algorithm as the course progressed through the various layers (see also Figure 5):

- Software-only implementation based on a direct translation of the mathematical operations in the cipher algorithm [8]. The focus was on code readability and maintainability.
- Optimized version of the AES based on the T-box design [11] with further performance improvements based on loop manipulation, data size adjustment, and function inlining.
- Hardware-software implementation using the Impulse C development tool. This version was a direct translation of the unoptimized version of the AES with no focus on performance degradation due to pointer manipulation, memory usage, loop sizes, and data types.
- Optimized version of hardware-software implementation of the AES developed with Impulse C, where the details of the underlying communication buses and available memory resources were taken into account to reduce the overall cycle count.
- All hardware implementation of the AES with a folded register architecture (amounting in a 32-bit data path) [12].

All software was written using the C programming language and hardware was modeled using Very-high-speed integrated circuits Hardware Description Language (VHDL). In addition, a hardware-software co-design methodology was practiced using Impulse C, which is a specialized subset of the C programming language intended for rapid hardware and hardware-software development using a C-to-FPGA compilation engine [13].

In both offerings of this course, improving student understanding of the cryptographic algorithms and different implementation techniques by removing

high-level abstractions paved the way for each new implementation of the AES. As part of each implementation exercise, students were required to analyze their work from performance gain and associated cost perspectives. This was a necessary step to understand the balancing act that exists between development time and effort, design complexity and code readability, implementation cost (source code size and circuit area), and performance improvements.

B. Breadth of Material

1) Cryptographic Foundations: From a cryptographic standpoint, the focus of the course was on block ciphers and hash function implementations in both software and on FPGA-based embedded systems. The AES has been, and continues to be, the subject of intense research in large part focusing on software and hardware optimization efforts, and resistance to side-channel attacks, all of which tie in with the underlying theme of this course.

In addition to the AES algorithm itself, students were also introduced to various NIST-recommended modes of operation for block ciphers. The following list enumerates these modes and how they contribute to the security properties of block ciphers [14], [15], [16], [17], [18], [19].

- Confidentiality - Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), Counter (CTR), Ciphertext Stealing for CBC mode.
- Authentication - Cipher Block Chaining Message Authentication Code (CBC-MAC).
- Confidentiality and Authentication - CBC-MAC with Counter, or Galois/Counter Mode (GCM).
- Confidentiality on Storage Devices - Tweakable encryption XOR-Encrypt-XOR (XEX), XEX-based tweaked Code Book with Ciphertext Stealing (XTS).

For a practical assignment, the focus was mainly on the Galois/Counter Mode mode of operation in order to study both data confidentiality and authentication. Given that the students were already familiar with the mathematical and algorithm design background concepts from previous lectures and assignments, the integration of this mode of operation was relatively seamless. The culmination of this material resulted in a software implementation of the core AES-GCM.

Following a similar mindset, the course project was based upon the SHA-3 hash function competition. This exposed students to hash functions as another class of cryptographic primitives. It also prepared them for the undoubtedly large integration effort in which existing hash functions will be replaced by the soon-to-be-selected SHA-3 winner.

Various topics on public-key cryptography, including RSA, ElGamal, and elliptic curve cryptosystems [20], were also integrated into the course schedule. These served to emphasize the need for security through cryptography and the applicability of cryptographic primitives in the real world. It was also necessary to cover the basic mathematical properties and operations that are utilized in such designs in order for the students to grasp the specific details of these algorithms. This part of material was largely focused on binary field arithmetic (specifically, Galois fields GF(2^8) and GF(2^{128})). The properties of finite fields and related operations were discussed in class. The students were left to explore these ideas in more detail as part of an exercise where they had to implement a library of finite field functions. In addition to basic addition, subtraction, and multiplication operations, the students were asked to implement functions to find the multiplicative inverses using the extended Euclidean Algorithm [20], and generators of the field.

2) FPGA-Based Embedded Systems: A portion of the course was devoted to the discussion of FPGA-based embedded systems. Due to the inherent co-existence of hardware and software in these systems, as well as the numerous performance, power, and size constraints they face. They are particularly useful for cryptographic applications because they provide the following:

- Algorithm acceleration with spatial (parallel) computing in FPGA fabric.
- Flexibility of processor-based computing using high-level software tools.
- Tightly coupled hardware and software domains for performance and flexibility benefits. A more in depth treatment of this benefit was described by [21] [22].

In order to familiarize the students with this platform the first two lab exercises served to introduce the basic design flow and techniques for using

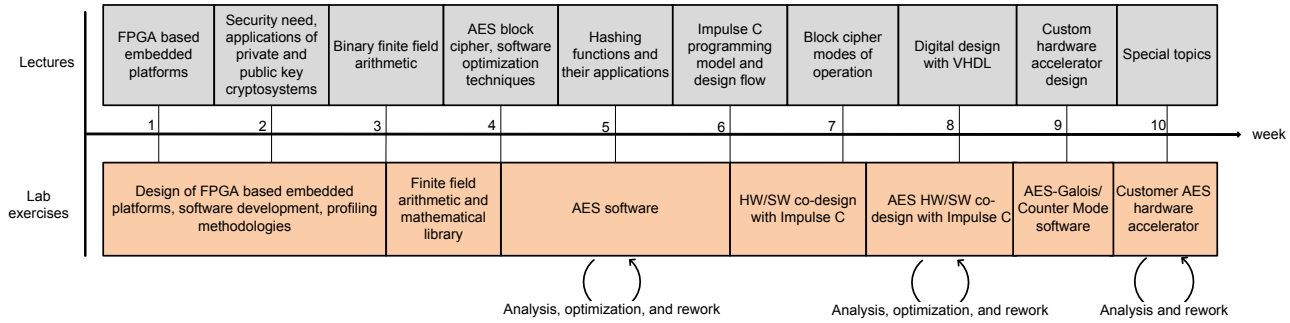

Fig. 2. Weekly progression of the course lecture material and lab exercises. Special topics include an overview of public-key cryptosystems, side channel attacks, and memory encryption. These can vary based on active research in the area, and are meant to expose students to current challenges being addressed by engineers and researchers.

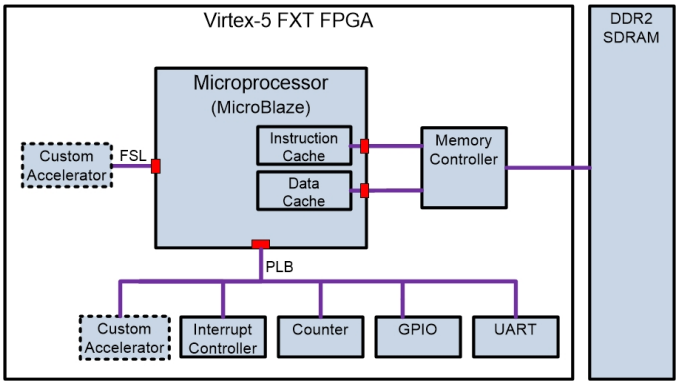

Fig. 3. General structure of the embedded system developed by the students. Components surrounded by dashed lines indicate that they are optional and are not required for normal system operation.

FPGA-based embedded systems. These exercises included working with the Xilinx Platform Studio (XPS) and Embedded Development Kit (EDK) to create multiple configurations with both the hard-core PowerPC and soft-core MicroBlaze processors and run sample test applications. Additionally, the students were shown how to add custom IP (Intellectual Property) components to their configurations, including timers, counters, and an interrupt controller, which were later used to profile software applications using a simple cycle-based measure.

3) Embedded Software Development and Optimization: From an implementation standpoint, the focus was on translating cryptographic algorithms and designs into running software and hardware, profiling and analyzing these implementations, and taking further actions to improve their performance.

The optimization efforts were centered around the standard closed loop optimization cycle, as depicted in Figure 4. To identify the performance bottlenecks in the original AES implementation, the students used a counter for simple cycle-based measurements and the GNU profiling tool gprof to analyze the source code at runtime, with and without the instruction and data caches enabled.

This analysis exposed students to the performance critical sections of the core AES algorithm. As part of unveiling the next implementation layer in the course the students exercised this knowledge by conducting both code-level refactoring and design-level modifications to optimize the software. These optimizations were primarily focused on basic code restructuring techniques with emphasis on the target platform. These implementation-specific techniques included:

- loop manipulation,
- replacing finite field arithmetic operations with lookup tables (LUTs),
- function inlining,
- data type resizing and data structure refactoring.

In addition to these optimizations, the students also explored the T-table design for AES, which exploits the independence between the SubBytes, ShiftRows, and MixColumns operations in the cipher. This allowed them to be combined into four different 1024 byte lookup tables, each one repre-

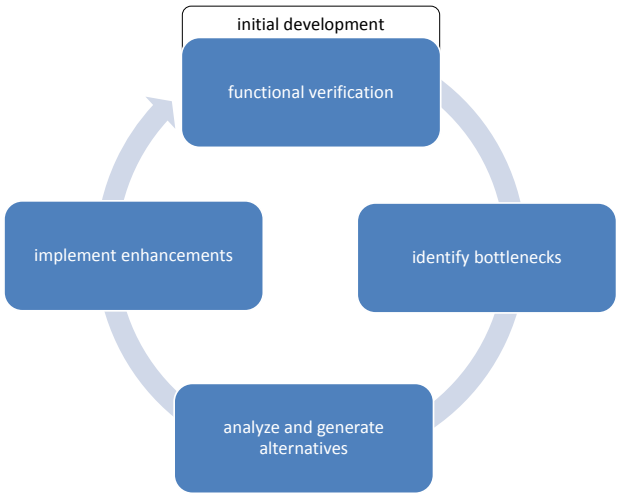

Fig. 4. The standard closed loop optimization cycle that is followed to incrementally improve performance of software and hardware applications.

senting the result of the operations on a 32-bit word in the internal state array. Using these tables the students reduced an entire round of encryption (and decryption) to a series of table lookups and XOR operations, which greatly improved the performance of their software implementations at the cost of application size.

4) Hardware and Hardware-Software Design and Development: The next phase was to port the existing software implementations to a hardware-software project. This served as the students' first exposure to custom hardware accelerator versions of the core AES algorithm. To facilitate the initial hardware and hardware-software development, Impulse C, a C-to-FPGA programming model and compilation engine that promotes rapid prototyping of digital hardware using a subset of the C programming language was introduced. Using Impulse C as an efficient tool in the development of hardware systems by software oriented students was studied by [22]. In this course students used it to transform their existing software designs for AES to hybrid hardware-software solutions that performed at significantly higher levels. This was a two-fold process that first consisted of abstracting the AES encryption routine as a single hardware process that was driven by two software processes, as shown in Figure 5. The producer software process was responsible for feeding the encryption process with plaintext and key data. Once encrypted, the ciphertext was streamed to the consumer process.

TABLE I
AES IMPLEMENTATION PERFORMANCE GAINS.

Implementation type	Relative performance
Unoptimized software	1.0
Optimized software	5.6
Unoptimized Impulse C	128.6
Optimized Impulse C	137.2

After the students became familiar with the Impulse C programming model and the C-to-VHDL compilation process, the next step was to analyze and optimize the existing implementation for improved throughput. This included using a variety of techniques, shown below, that exploit the code generation stages of the Impulse C compiler.

- Change the size of the streaming interfaces to match the width of the data transfer bus to improve consumption rates.
- Modify the size of the data types in arrays to make full use of the PLB for memory fetches.
- Replace arrays with local variables where possible to reduce external memory reads.
- Split up arrays into smaller constituent arrays to parallelize memory accesses to make use of multiple memory channels.
- Modify loops to experiment with both improved parallelism and reduced code size.
- Pipeline blocks of code where possible to improve throughput.

Using the optimized Impulse C tool students were able to achieve a speedup of 137.2 from the original software implementation using the PLB for communication, as shown in Table I. The overhead of software and hardware process communication was the ultimate bottleneck in the solution, as the bandwidth of the PLB only allowed for so much data to be transferred every clock cycle. Future course offerings will transition from a dependency on the PLB to the Fast Simplex Link (FSL) for communication with custom hardare accelerators. As part of this shift, students were asked to experiment with both the PLB and FSL in their lab exercises. Preliminary performance results from these are detailed in Table I.

It is important to recognize that the C-to-FPGA translation process was also a limiting factor in the performance of the solution. Since Impulse C is simply another abstraction layer on top of a custom HDL implementation, the compilation process generates code according to a generic template.

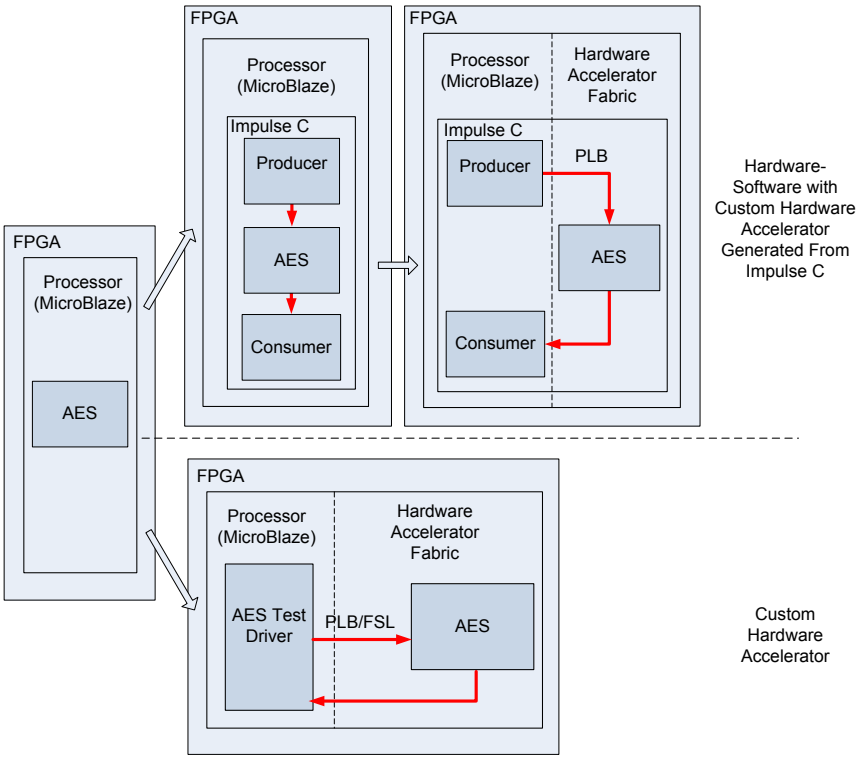

Fig. 5. The incremental development versions of AES that were studied throughout the course. The left box depicts the original software version of AES as it was run on the ML507 development board. The top two boxes show the incremental development cycle used to translate the software version into a hardware-software co-designed application using Impulse C. The bottom box depicts the design of the custom hardware accelerated version of AES, where the software was only responsible for driving the internal logic of the hardware component. The FSL was added to the custom hardware accelerator implementations to expose students to the different interfaces available on the target platform.

Typically, the resulting code will not be as concise or efficient as if it were manually modeled and then synthesized.

The next and final stage in the development cycle focused on customized hardware accelerators for the AES encryption engine. This required an understanding of basic hardware modelling and digital design concepts. The difficulty of this part of the course is that it had to be tailored towards software-oriented students, who might lack the computer and electrical engineering background that other students possessed. To account for such students, a single class was dedicated to VHDL syntax, development practices, and modelling techniques. This was done in order to provide them with enough information to at least interpret and understand VHDL code.

Once the students became acquainted with hardware modelling techniques and design flows, they were provided with a working implementation of the AES engine. A high-level design of this module is shown in Figure 6. After verifying the functionality of this model using a provided testbench, the students were asked to modify redesign the architecture using a folded register scheme in order to shorten the width of the data path.

Putting all of these stages together, the overall flow of the AES implementations throughout the course is depicted in Figure 5.

C. Course Project

The SHA-3 hash function candidates were chosen as the basis for the course project because of their importance in the study and practice of cybersecurity [9].

With the results of the SHA-3 competition still pending, hash function research efforts continued with the five finalist functions (BLAKE, Grøstl, JH, Keccak, and Skein [10]). Given the importance of selecting the candidate that provides the best combination of security, performance, simplicity,

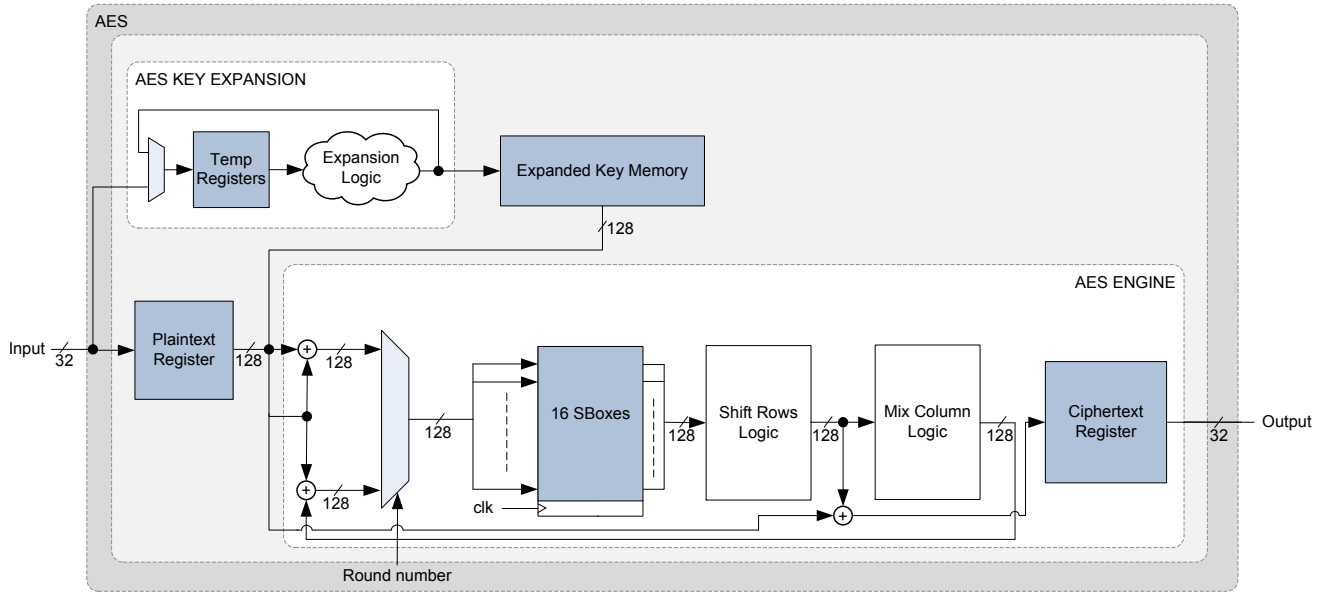

Fig. 6. High-level design of AES custom hardware accelerator. Input and output to the module consists of streams of 32-bit words that contain the key and plaintext data on the input channel and the ciphertext on the output channel.

and modularity, many teams across the world have focused on the mathematical and statistical properties, and the implementation efficiency of these five hash functions.

For the purpose of the project, the students were divided into five teams; each one focusing on a single SHA-3 finalist. The term-long project had four major parts culminating with a research paper and presentation by each team. The highlights of each part were:

1) Researching the history of hash functions, the current standard hash functions, and recent advancements in cryptanalysis efforts that target hash functions.
2) Discussing the internal algorithmic details of the teams hash function, which served as a basis for the remaining parts of the project.
3) Analyzing published research efforts on the design, implementation and performance of the hash function.
4) Implementing the hash function in software using the publicly available resources provided by each SHA-3 candidate team.

In order for students to have a common baseline for comparison of their results, all teams implemented the 512-bit digest versions of their hash functions. All teams quantified the performance of their implementations by collecting cycles/byte measurements on fixed sized messages.

Now, with Keccak announced as the winner [23], we anticipate more attention in our courses will be devoted to this particular algorithm.

III. Student Impact

In order to assess achievement of the course learning objectives and outcomes, an anonymous survey was administrated to the students at the end of the course, to which 21 answers were received. The main results are summarized in Tables II and III, which indicate an overwhelming agreement that this new course successfully completed all specified goals. The following are some exemplary quotes from students that illustrate their support and approval of this new course.

The course material was diverse enough for the various disciplines and covered a great deal of material.

It has opened my eyes to more topics that are commonly discussed in industry. I am interested in obtaining a Ph.D. in the field of computer security, and this course plays right into the research and materials I am interested in.

I think demonstrating hardware/software co-design is very valuable to students since this is such a common team configuration in the industry. This will help to develop practical skills.

I think the most difficult thing will be taking all the prerequisites - the value of this course could be immense if everyone came into the class with a strong background in crypto, and both hardware and software design.

A few auxiliary questions were asked in the survey in order to measure student's general interests in applied cryptography, their self-assessment on background preparation, and achievements in this course. Selected results referring to this part of the survey are presented in Table IV.

In the second offering of the course in the spring quarter of 2012, we also included a longitudinal test that quantitatively assessed general knowledge acquired throughout the term. Questions were drawn from the areas of cryptographic foundations, FPGA-based embedded systems, and high-level synthesis. This test was administered at both the begining and end of the course. Tables V and VI present the results that were collected.

We performed a 1-tailed paired difference t-test analysis on the sample of 13 students. Our analysis indicated that there was a statistically significant increase in knowledge acquisition for all question groups, with a p-value less than 0.005, based on the pre- test and post-test mean scores derived from the data in Table V. In calculating the mean, unanswered questions were treated as incorrect answers. Our result can easily be seen in the raw data shown in Table VI by observing the three-fold decrease in unanswered and incorrect questions from 141 to 45, and approximately a doubling of correct answers from 68 to 122. Also, there was a clear lack of knowledge in high-level synthesis and cryptography, as shown by the large cluster of unanswered questions from these two categories in the perliminary test. However, as the results show, knowledge in these areas showed a three-fold decrease from 96 unanswered and incorrect answers to 34 unanswered and incorrect answers.

IV. Discussion

Our students have particularly varied backgrounds because they are coming from different departments. The balance of different components of our course can be adjusted to take into account specific needs of enrolled students. For example, computer engineering students may need more material on cryptography whereas software engineering and computer science students almost certainly will require more extensive lectures and practice on hardware implementations. Additional material on cryptography could include more thorough treatment of number theoretic foundations or complexity theory considerations in relation to the security cryptographic primitives. Extended hardware perspectives for software-oriented students might encompass more time dedicated to high-level synthesis tools and methodologies, custom digital logic design with hardware description languages, and target implementation platforms.

The project activity of the initiative described in this paper is based on the finalist SHA-3 hash function candidates. With the announcement of Keccak [24] as the winner of the SHA-3 competition on October 2, 2012 [23], it will be natural to shift more of the focus of the material of this course towards this algorithm. The current Digital Signature Standard (DSS) uses as its component the SHA-1 hashing, whose security has been questioned [9] [10] and has different message digest length from SHA-3. Thus, we expect that in the near future there will be intense discussion and perhaps even similar competition organized to propose and introduce new standards for cryptographic digital signatures. In such a case we would like to follow such developments in our course by refocusing the project activity towards signatures.

V. Conclusion

This course was developed as part of a new security-oriented curriculum that draws from the fields of computer science, computer engineering, and software engineering. The main goal was to accommodate a mixture of computer engineering, software engineering, and computer science students by teaching them about the theoretical and practical aspects of developing implementations of cryptographic primitives, such as block ciphers and hash functions. The focus was on the standardized AES

TABLE II
COURSE LEARNING OBJECTIVES QUANTIFIED FROM STUDENT SURVEY.

Learning objective	Strongly agree	Agree	Undecided	Disagree	Strongly disagree
Provide knowledge and understanding for design and implementation of cryptographic primitives on FPGA-based embedded systems.	52%	48%	0%	0%	0%
Provide knowledge and understanding of hardware/software co-design methodologies and techniques.	48%	52%	0%	0%	0%

TABLE III
MEASURABLE LEARNING OUTCOMES QUANTIFIED FROM STUDENT SURVEY.

Learning outcome	Strongly agree	Agree	Undecided	Disagree	Strongly disagree
Students have successfully customized and implemented an FPGA based embedded processor system; students understand how to configure linker scripts and build software projects for cryptographic primitives.	67.0%	33.0%	0.0%	0.0%	0.0%
Students know how to profile software applications to identify performance bottlenecks. Students have successfully optimized a software application to improve performance; students have analyzed cost in terms of the application size.	47.6%	47.6%	4.8%	0.0%	0.0%
Students have successfully performed high level synthesis from C programs to FPGA hardware; students have analyzed performance improvement in a hardware/software system.	57.1%	38.1%	0.0%	0.0%	4.8%
Students have successfully implemented a custom hardware accelerator for selected architecture to achieve desired performance in cost and area.	52.4%	47.6%	0.0%	0.0%	0.0%

TABLE IV
SELECTED AUXILIARY QUESTIONS. THE ENTRY MARKED * INDICATES NO REPLIES WERE GIVEN FOR THE CORRESPONDING QUESTION, NOT EXPLICIT DISAGREEMENT.

Question	Strongly agree	Agree	Undecided	Disagree	Strongly disagree
The amount I learned was worth the time invested in this course.	52.4%	28.6%	19.0%	0.0%	0.0%
My preparation was adequate for taking this course.	52.4%	38.1%	4.8%	4.8%	0.0%
I would recommend to a friend to take this course as an elective.	52.4%	38.1%	9.5%	0.0%	0.0%
This course increased my interest in computer security.	23.8%	47.6%	23.8%	0.0%	4.8%*
I plan to seek employment in the computer security area.	9.5%	4.8%	52.4%	23.8%	9.5%

TABLE V
INDIVIDUAL STUDENT SCORES GATHERED FROM THE PRE-TEST AND POST-TEST. OUT OF 15 TOTAL STUDENTS IN THE CLASS, 12 PROVIDED BOTH SETS OF DATA.

Pre-Test			Post-Test			Score Data		
Correct	Incorrect	Unanswered	Correct	Incorrect	Unanswered	Pre-Test Scores	Post-Test Scores	Score Differences
4	2	8	12	0	2	28.57	85.71	57.14
7	2	5	11	2	1	50.00	78.57	28.57
5	2	7	12	1	1	35.71	85.71	50.00
3	1	10	12	0	2	21.43	85.71	64.28
7	2	5	11	1	2	50.00	78.57	28.57
6	0	8	12	0	2	42.86	85.71	42.85
3	2	9	6	5	3	21.43	42.86	21.43
1	4	9	3	2	8	7.14	21.43	14.29
6	6	2	12	2	0	42.86	85.71	42.85
4	0	10	9	0	5	28.57	64.29	35.72
6	2	6	11	2	1	42.86	78.57	35.71
6	3	5	11	3	0	42.86	78.57	35.71

TABLE VI
QUANTITATIVE RESULTS FROM PRE- AND POST-TESTS ADMINISTERED IN THE SECOND OFFERING OF THE COURSE. THE TABLE ENTRIES ARE AGGREGATE SUMS OVER ALL COLLECTED TESTS.

Pre-Test			Post-Test			Question Subject
Correct	Incorrect	Unanswered	Correct	Incorrect	Unanswered	
44	14	31	61	3	8	FPGA-based embedded systems
6	11	28	20	8	7	High-level synthesis
18	7	50	41	7	12	Cryptography

algorithm for the majority of the lecture and lab material, and the SHA-3 hash function competition for the course project.

The material provided to students, both in class and in the lab exercises, covers a number of different topics, including theoretical background for the chosen cryptographic primitives, FPGA-based embedded system platforms and components, software implementation and optimization, and hardware accelerators design and modelling. Several engineering tools have been introduced for use throughout the course, including the Xilinx Platform Studio and Embedded Development Kit, gprof, and Impulse C.

Organizing the course material to represent an incremental structure proved to be an effective method for reinforcing the content provided in class lectures. This was reflected in the quantitative results taken from the student impact survey. The course learning outcomes and objectives were achieved while covering a wide breadth of material within the time constraints of a 10-week term.

Future offerings of this course will more emphasize on aspects of custom and secure hardware design, including side-channel attacks and countermeasure methods.

VI. ACKNOWLEDGEMENT

This work was supported in part by the NSF CCLI grant award 08-546 DUE-0837656. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Worcester Polytechnic Institute, "ECE 673 Advanced Cryptography." [Online]. Available: http://www.ece.wpi.edu/Graduate/13509.htm

[2] Virginia Tech, "ECE 5520 Secure Hardware Design." [Online]. Available: http://www.ece.vt.edu/schaum/securehw.html

[3] R. Hake, "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses," *American journal of Physics*, vol. 66, p. 64, 1998.

[4] M. Prince, "Does active learning work? a review of the research," *JOURNAL OF ENGINEERING EDUCATION-WASHINGTON-*, vol. 93, pp. 223–232, 2004.

[5] H. Hadim and S. Esche, "Enhancing the engineering curriculum through project-based learning," in *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, vol. 2. IEEE, 2002, pp. F3F–1.

[6] J. Vallino and R. Czernikowski, "Interdisciplinary teaming as an effective method to teach real-time and embedded systems courses," in *Proceedings of the Workshop on Embedded Systems Education 2008*, October 2008.

[7] ——, "Thinking inside the box: a multi-disciplinary real-time and embedded systems course sequence," in *Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference*, oct. 2005, pp. T3G –12.

[8] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)," Federal Information Processing Standards Publication 197, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[9] R. Kayser, "Announcing request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family," *Federal Register*, vol. 72, no. 212, November 2, 2007. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf

[10] National Institute of Standards and Technology, Computer Security Division, Cryptographic Hash Project Website. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/index.html

[11] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," March 1999.

[12] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," in *Cryptographic Hardware and Embedded Systems - CHES 2003*, ser. Lecture Notes in Computer Science, vol. 2779. Springer Berlin / Heidelberg, 2003, pp. 319–333.

[13] D. Pellerin and S. Thibault, *Practical FPGA Programming in C*. Prentice Hall, 2005.

[14] M. Dworkin, "Recommendation for Block Cipher Modes of Operation, Methods and Techniques," National Institute of Standards and Technology, Tech. Rep. 800-38A, 2001. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[15] ——, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," National Institute of Standards and Technology, Tech. Rep. 800-38D, 2007. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[16] ——, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality," National Institute of Standards and Technology, Tech. Rep. 800-38C, 2007. [Online]. Avail-

able: http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf
[17] ——, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication," National Institute of Standards and Technology, Tech. Rep. 800-38B, 2005. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
[18] ——, "Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices," National Institute of Standards and Technology, Tech. Rep. 800-38E, 2010. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
[19] ——, "Recommendation for Block Cipher Modes of Operation, Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode," National Institute of Standards and Technology, Tech. Rep. Addendum to 800-38A, 2010. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38a/addendum-to-nist_sp800-38A.pdf
[20] D. Stinson, *Cryptography: Theory and Practice*. CRC Press, third edition, 2006.
[21] P. Schaumont, "A Senior-Level Course in Hardware-Software Codesign," in *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, june 2007, pp. 7 –8.
[22] Y. Dandass, "Teaching Application Implementation on FPGAs to Computer Science and Software Engineering Students," *Computers in Education Journal*, vol. 18, no. 1, January 2008.
[23] "NIST selects winner of secure hash algorithm (SHA-3) competition," http://www.nist.gov/itl/csd/sha-100212.cfm. [Online]. Available: http://www.nist.gov/itl/csd/sha-100212.cfm
[24] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "Keccak specifications," 2009.