Efficient Hardware Accelerator for IPSec based on Partial Reconfiguration on Xilinx FPGAs

Ahmad Salman, Marcin Rogawski and Jens-Peter Kaps
Volgenau School of Engineering
George Mason University
Fairfax, Virginia 22030
email: {asalman, mrogawsk, jkaps}@gmu.edu

***Abstract*—In this paper we present a practical low-end embedded system solution for Internet Protocol Security (IPSec) implemented on the smallest Xilinx Field Programmable Gate Array (FPGA) device in the Virtex 4 family. The proposed solution supports the three main IPSec protocols: Encapsulating Security Payload (ESP), Authentication Header (AH) and Internet Key Exchange (IKE). This system uses efficiently hardware-software co-design and partial reconfiguration techniques. Thanks to utilization of both methods we were able to save a significant portion of hardware resources with a relatively small penalty in terms of performance. In this work we propose a division of the basic mechanisms of IPSec protocols, namely cryptographic algorithms and their modes of operation to be implemented either in software or hardware. Through this, we were able to combine the high performance offered by a hardware solution with the flexibility of a software implementation. We show that a typical IPSec protocol configuration can be combined with Partial Reconfiguration techniques in order to efficiently utilize hardware resources.**

***Index Terms*—Partial reconfiguration; IPSec; Xilinx FPGA**

I. Introduction

Internet Protocol Security (IPSec) [1]–[3] provides security against attacks on data transmitted over the Internet through security services facilitated by a set of protocols. It was designed to operate at the level of the Internet layer according to the OSI network model. This makes it completely transparent to applications and users.

The security services provided by the Internet Protocol Security (IPSec) include:

- Confidentiality - Prevents unauthorized access to the transmitted data.
- Data integrity - Ensures data was not altered during transmission.
- Authentication - Enables the identification of the information source.

The IPSec series of protocols makes use of various cryptographic algorithms such as encryption modules, hash functions and modular arithmetic in order to provide security services. The Internet Key Exchange (IKEv2) protocol in version two has to be used to establish secure connections, so called Security Associations (SAs). IKEv2 uses cryptographic algorithms: key exchange algorithm (Diffie-Hellman) and pseudo random function based on the Advanced Encryption Standard (AES) in XCBC mode(AES-XCBC-PRF-128). The Encapsulating Security Payload (ESP) protocol provides mechanisms for confidentiality and data integrity services. It uses AES cipher in Cipher-Block-Chaining (CBC) and Counter (CTR) modes of operation. The Authentication Header (AH) protocol provides connectionless integrity and data origin authentication. AH uses Hashed Message Authentication Code (HMAC) with Secure Hash Algorithm (SHA). To assure protection and standardization, the minimum set of cryptographic algorithms that must be supported by an implementation of IPSec for ESP, AH and IKEv2 protocols as stated in [2] is illustrated in Table I.

TABLE I
IPSec Supported Protocols and Algorithms

Protocol	Security Service Provided	Supported Algorithm
ESP	confidentiality through encryption and optional data integrity	AES in CBC or CTR mode
AH	connectionless integrity and data origin authentication	HMAC-SHA1-96, AES-XCBC-MAC-96, HMAC-SHA-256
IKE	negotiates connection parameters	Diffie-Hellman scheme in 1024 or 2048 bits groups and AES in PRNG mode

Due to the broad use of IPSec, it has been implemented in hardware and software with various designs and parameters to suit different platforms and provide better solutions. Among popular implementations of IPSec in hardware are those that target FPGA platforms because of the flexibility they offer the designer, ease of programming and high speeds that cannot be achieved through software. Due to the fact that FPGAs are resource limited devices, even efficient implementations of IPSec with all the services it provides might not fit on low cost devices or low area devices that are meant for light weight implementations [4]. A solution to this problem can be Partial Reconfiguration which allows some IPSec services to be available in the system and the remaining services can be recalled when needed by an application.

Partial Reconfiguration [5] is a configuration method for FPGAs that allows certain portions of the device to be reconfigured during run-time without affecting other portions in the system or their functionality. In this paper we will investigate the effect of implementing IPSec services using Partial

Reconfiguration in terms of speed, area and reconfiguration time. For that, we built an embedded system controlled through an embedded processor to provide self reconfiguration of the system through a software application. We implemented the embedded system using the Microblaze soft-core processor targeting a low area Virtex-4 device to perform thorough testing on the proposed design and analyze the results.

The rest of this paper is organized as follows: In Sect. II we discuss previous work, Sect. III is devoted to the description of our proposed system, and Sect. IV describes the experiment methodology we followed to evaluate the system. Sect. V discusses and analyze the results and we draw conclusions in Sect. VI.

II. Previous work

One important software implementation of IPSec was developed within the KAME project [6]. Racoon software (which is part of the KAME project) is a tool for handling Internet Key Exchange (IKE) in IPSec and almost all of its source code has been merged into FreeBSD and NetBSD. Hardware implementations of IPSec basic components were investigated in several publications. In [7] a hardware co-processor was proposed based on AES and HMAC-SHA1 cores. This solution was implemented on XCV1000E Xilinx Virtex device. An implementation on Xilinx Virtex-II Pro FPGA was presented in [8]. The key management and negotiation functions were moved into software. The encryption and data integrity support was based on AES, HMAC-MD5 and HMAC-SHA1. Those algorithms were implemented on a single device and all hardware cores achieved over 1 Gb/s throughput.

A typical node connected to the Internet has to process multiple different streams of data. The efficient handling, of those streams of data, was investigated in [9] which proposed pipeline techniques on HMAC-SHA1.

A very complex system such as an IPSec embedded solution requires finding the correct balance between flexibility and performance in a design. Principles of the hardware/software co-design techniques are described in [10]. A proposed embedded solution for only the IPSec AH protocol can be found in [11]. There are also hardware accelerators of the IPSec protocol suite available as commercial products: [12], [13] and [14].

A Partial Reconfiguration [5] technique is relatively new in the area of SRAM based Altera devices [15] and it is only supported in the newest 28nm Stratix V. In the case of Xilinx devices [16] this method has been known for almost a decade, but due to the complex nature it has never been very popular in both commercial and academic applications.

To the best of authors knowledge, the first utilization of partial reconfiguration on Xilinx FPGA devices for IPSec was proposed in [17].

III. System Description

A. Overview

Our proposed IPSec embedded system's structure is summarized in Table II. We implemented three cryptographic transformations in hardware where at any given point in time, only one is available for its utilization. In order to use a different algorithm than the one currently available, partial reconfiguration operation is performed. This process is controlled by a modified Round Robin with time sharing scheduling algorithm [18] implemented on the embedded processor. Packets are sent from and received by the system through input and output queues. The input queues send packets to be processed by the corresponding cryptographic algorithm and the output queue is devoted for storing the output results as shown in Fig. 1.

The Round Robin scheduling algorithm is used for switching the control between queues when packets are ready to be processed. In this system, we use a configurable time slot value which specifies for how much time packets from one queue are being processed before switching to another queue. Even if the current queue still contains data the processor switches to a different queue to make sure that no specific type of packets monopolize the co-processor. We will show in Sect V that the time slot parameter can be considered as a trade-off between total throughput of the proposed system and latency of packet transition through the IPSec co-processor.

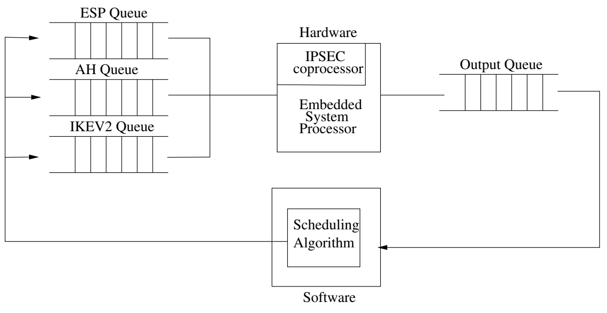


Fig. 1. Synchronization Circuit Between Hardware and Software

For example, if the scheduler assigned a time slot for the packets that require AES operation (ESP packets), the CPU makes sure that the AES module is currently running on the co-processor otherwise it performers partial reconfiguration to make it available for the assigned packets. If the time slot expires and there are new packets in the SHA-2 (AH Packets) related queue then another partial reconfiguration operation is required to load the SHA-2 module to the co-processor. After this operation is completed, the packets from SHA-2 related queue can be transmitted to IPSec co-processor. The transmission between the embedded processor and co-processor requires hardware-software synchronization which is described in the synchronization circuit subsection.

We can observe that the system latency consists of the latency of data preparation for computation and latency of the computation process itself. The latency of the data preparation includes the time needed to perform partial reconfiguration. The latency of computation consists of latency of input transmission to the hardware core, latency of cryptographic transformation and finally the time for sending and storing the result. It is clear that in order to utilize the co-processor more efficiently, the time for computations should be maximized compared to the time needed for data preparation. On the other hand, if both latencies are too high, the responsiveness of system will degrade. Finally, in order to maximize the efficiency of computation time, the interface latency needs to be minimal.

TABLE II
HARDWARE-SOFTWARE CO-DESIGN IMPLEMENTATION DETAILS OF PROPOSED IPSEC SYSTEM

Implementation		
In Hardware	In Software	Application
AES	CBC, CTR modes	ESP
	MAC-XCBC-96	AH
	XCBC-PRF-128	IKEv2
SHA-256	HMAC	AH
MODEXP	Montgomery domain transformations	IKEv2
-	Round Robin scheduling algorithm	PR trigger

B. Partial Reconfiguration

Partial Reconfiguration (PR) [5] is the process of configuring a portion of a FPGA while the other part is still running [5]. The PR method is independent of its implementation method, meaning that although the idea of creating a partially reconfigurable design is the same between different companies and PLD manufacturers like Xilinx and Altera, each has their own tools and implementation methods.

A typical PR system is composed of static regions known as Base Region (BR) and a dynamic region known as Partial Reconfigurable Region (PRR). The BR holds the portion of the design that does not get affected by partial reconfiguration while the PRR holds the portion of the design that gets swapped during partial reconfiguration process which is known as Reconfigurable Modules (RM). A PRR is composed of at least a single RM and usually multiple ones as shown in Fig. 2.

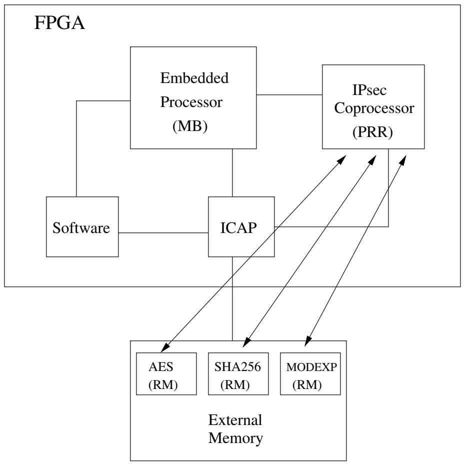


Fig. 2. A Partial Reconfigurable Region and Associated Reconfigurable Modules

In our design, the BR includes an embedded processor that controls the PR process and some supporting peripherals while the PRR includes hardware accelerators for IPSec protocols. Initially the system is configured with the BR and the PRR is loaded with one of the RMs or left blank with no RM loaded. The remaining RMs are stored on an external memory and are swapped with other RMs by the scheduler.

During PR, the embedded processor communicates with the Internal Configuration Access Port (ICAP) which loads the partial bitstream that holds the information of the requested RM from the external memory and replaces the currently running RM or the blank space in the PRR with it during run-time as shown in Fig. 2.

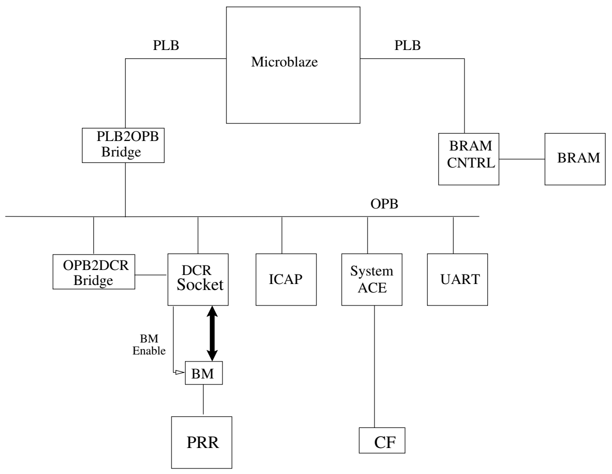


Fig. 3. Embedded System Processor and Peripherals

C. Hardware

Our design targets the ML403 board with a Virtex-4 device described in [19]. Although the target device includes a PowerPC embedded processor, the soft-core Microblaze processor was used as the embedded processor in the design to insure compatibility with other devices that do not include a PowerPC. Figure 3 shows our embedded system block diagram. The internal data bus for the Microblaze processor is 32-bit wide. It is connected to the BRAM-block peripheral through the Processor Local Bus (PLB) BRAM Interface Controller (BRAM_IF_Ctrl) which is interfaced to the PLB Bus. The BRAM-block peripheral gives the processor access to the BRAM components which constitute the memory of the system.

The peripherals in the system are interfaced with each other and the Microblaze through the On-chip Peripheral Bus (OPB). Peripherals in the system include a Universal Asynchronous Receiver/Transmitter (UART) for debugging and output display, a System ACE to interface a FAT32 Compact Flash (CF) memory card used as the non-volatile memory that holds the partial bitstreams of the RMs and the Hardware Internal Configuration Access Port (HWICAP) which is the hardware peripheral that enables the Microblaze to access and modify the configuration memory while the circuit is operational through the ICAP. There is a custom peripheral which represents the PRR in the system which is the target for the RMs. This custom peripheral is interfaced to the system through the Device Control Register (DCR) bus. There are also two bus bridges used, the plb2Opb bridge and opb2dcr bridge, to allow the communication between different buses in the system and the peripherals interfaced to them.

Bus Macros (BMs) are used to provide a means of locking the routing between RMs and the BR, making the RMs pin compatible with the base design. With the exception of global clock signals, all other signals including reset signals must pass through BMs. During PR process, the Microblaze disables BMs using a Bus Macro Enable/Disable signal as shown in Fig. 3 to prevent data from being sent to or received from the target PRR until the PR process is completed then BMs are enabled again.

The PRR in the system is composed of three RMs, Advanced Encryption Standard (AES), Secure Hash Algorithm (SHA)-256 and a modular exponentiation module (MODEXP) representing the hardware accelerators for the IPSec ESP, AH and IKEv2 protocols respectively.

The AES hardware core is based on a 128-bit datapath and the latency is 11, 13 and 15 clock cycles for 128, 192 and 256-bit main key length, respectively. This architecture is described in [20]. The SHA-256 hardware implementation is based on the architecture with the best throughput/area ratio proposed in [21]. The modular arithmetic is based on the high 2^{16} radix multiplication Algorithm 3 proposed in [22] and on the exponentiation algorithm proposed in [23]. In order to save area, we decided to restrict the number of processing elements to one, which can be described by the parameters: word size = 16 bits, number of scanned bit = 16. Thanks to this assumption our basic processing element could be efficiently implemented on just 3 DSP blocks. We decided to exploit special shift register mode (SRL-16) to organize storage system for arithmetic arguments and intermediate results. Our very restricted area budget did not allow us to use directly FPGA optimized architectures proposed in [24] or [25]. All three cores support a simple FIFO based interface described in [26] where the input and output data width is 32 bit wide.

D. Software

The software portion of the system includes the software drivers for the hardware peripherals in the system, some basic C libraries as well as Initialization functions for the HWICAP and the ICAP API. In addition to that, all modes of operations (e.g. CBC and XCBC) and data preparation (e.g. HMAC calculations) are being done in software to give the system flexibility as these operations can be used with different cryptographic algorithms by applying minimal software-only changes to the system.

E. Synchronization Circuit

As the speed of hardware is significantly higher than software, any hardware/software co-design should include a synchronization method between the hardware and software to assure correctness of communication and prevent data loss. We created a hardware/software synchronization circuit with a FIFO interface shown in Fig. 4. Sending and receiving data is controlled by control signals from the embedded processor. When data is being sent from software to the hardware core, the processor sets the read_ack signal to high enabling hardware to read data in 32 bit at a time according to the core settings. After the core performs calculations, data is sent from hardware to software in the same manner with the processor setting the write_ack signal to high to let the software know that the output is ready.

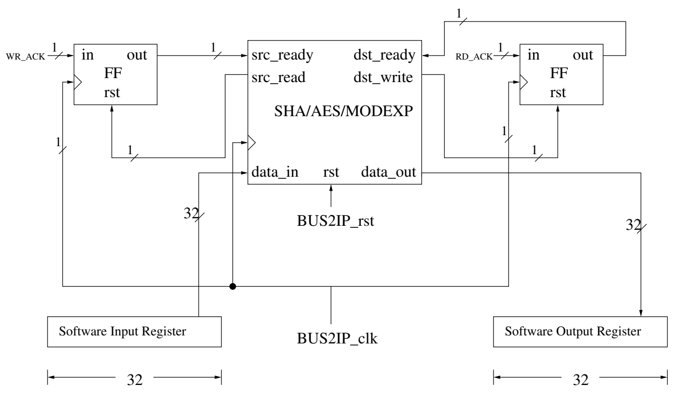


Fig. 4. Synchronization Circuit Between Hardware and Software

IV. Methodology

A. Overview

Before implementing the proposed design in a PR system, we first implemented it in a non-PR system. The main reason for the non-PR implementation was to make sure that the AES, SHA-256 and MODEXP cores perform as expected and producing the correct results at run-time. We also wanted to test the synchronization circuit and to find out the amount of area consumed by each core implemented separately to compare it to that of the PR design.

The bottleneck in the design is the PR process, i.e. the process of replacing one module with another. PR is a time consuming process which increases the system latency hence decreasing performance. This could make the design impractical. Therefore, we used the scheduling algorithm described in the Sect. III to increase the throughput.

To test the design, we emulated IPSec traffic through a sequence as input to the system. The sequence starts with an IKEv2 packet for handshaking, algorithm selection and sharing of keys. The reset of the sequence is composed of string of bits representing the data portion in the ESP and AH packets. Packets are identified and assigned to the queues corresponding to their type. The scheduler always starts from the IKEv2 queue when a new SA is established. For this reason, the MODEXP module should be loaded to the PRR during the initial FPGA configuration. Following SA establishment, the scheduler assigns packets to the Microblaze to be processed by the hardware.

B. *IKEv2 vs AH and IKEv2 vs ESP*

In a typical IPSec Security Policy Database (SPD) tunnel configuration, a connection parameter renegotiation is established no more than once per hour. Even though there are some applications that require more exotic configurations. AH and ESP protocol handling will take more than 99% of any practical IPSec co-processor. Due to that fact, the IKEv2 is rarely used compared to AH/ESP. The MODEXP, which is a basic module for this protocol, can be swapped with AH/ESP accelerators when needed. The AH and ESP hardware accelerators are also being used interchangeably on the chip using PR but due to the fact that they are both used excessively, techniques like using input queues or on board memory as well as packets scheduling algorithms as we described in the previous section should be used to allow for high throughput.

C. *AH vs ESP*

In the case of ESP and AH protocols the situation with efficient Partial Reconfiguration is more complicated. First of all, the number of tasks related to AH vs ESP is more balanced than in case of IKEv2. This means that in order to decrease the influence of relatively expensive, in terms of latency, partial reconfiguration operation, we have to use scheduling of tasks. We selected Round Robin with time sharing scheduling algorithm, because it is a very simple and well known scheduling algorithm. One very important property of this algorithm is its security against the starvation problem [18].

V. Results Discussion

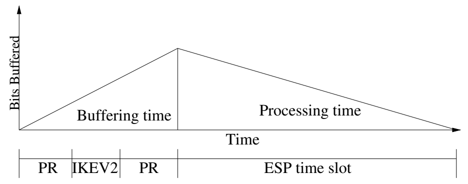
The design was synthesized and implemented using Xilinx Design Suite 9.1 as it supports PR design implementations as well as non-PR designs. The target device is the XC4VFX12 Virtex-4 FPGA on the ML403 board. We choose the XC4VFX12 FPGA because it is a low cost high speed device and has enough area to implement the proposed design. The SHA-256 and AES cores were verified individually using test vectors provided by FIPS standards [27] and [28] respectively. The MODEXP arithmetic core was verified using test vectors generated by the OpenSSL protocol suite [29].

The implementation results are summarized in Table III. The first two columns summarize the resources of the static and dynamic portions of the system. The other three columns are implementation results for each of the three cores (AES, SHA-256 and MODEXP) implemented independently in non-PR designs. When comparing the dynamic portion of the design to the non-PR implementations, it can be noticed that the PR design uses 2148 slices compared to 3285 slices used by the three non-PR designs combined together. This is an area improvement of more than 34%. These area savings are open to further improvements if more RMs are available for the same PRR. Which means that other cryptographic algorithms supported by IPSec can be added without requiring additional logic resources.

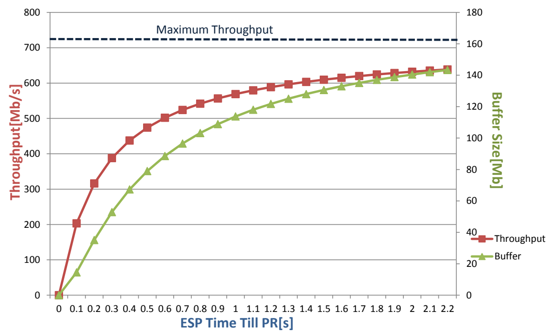
Not only is the PR design more resource efficient, it also makes implementing an embedded processor with the IPSec co-processor feasible on the target platform. When Implementing all three cores as parallel independent IP cores in a single non-PR processor system, The Microblaze adds an overhead to the area of each core. This area increase in addition to routing issues makes the target device fails to accommodate a fully parallel implementation of the design.

A. *Latency vs Throughput*

As a system's performance is measured by area and throughput together. We wanted to test how the overhead time caused by the PR process degrades performance. The most common scenario is when the system processes IKEv2 and ESP packets. Initially the system has to perform PR to load Modxp RM to process IKEv2, perform PR again to load AES RM and process ESP packets until the time slot for ESP expires as shown in Fig. 5a. Then this scenario repeats. During PR and IKEv2 processing, ESP packets are being received and buffered in the ESP queue. The amount of bits buffered is also shown in Fig. 5a. This buffer size defines the size of the queue for each protocol depending on the time slot assigned as well as the network traffic. Once the AES RM is loaded, processing of ESP packets starts. ESP has to process the packets faster than the arrival rate such that the ESP queue is empty when the ESP time slot expires. Hence, the ESP throughput is depending on the ratio of ESP time slot versus the time required for PR and IKEv2 processing as shown in Fig. 5b.



(a) Buffer level depending on operation



(b) AES hardware core performance depending on size of time slot

Fig. 5. AES hardware core performance dependency on ESP time slot for the 32-bit internal interface width

It can be shown that in order to reduce the effect of PR on the total time (time needed from packet received by the system to sent back after being processed), the amount of time assigned for computations (encryption/hashing) should be increased. For example, if the PRR is loaded with the AES core module, then the scheduling algorithm should direct the Microblaze to fetch tasks from the ESP queue as long as it

TABLE III
RESOURCES SUMMARY FOR IMPLEMENTATIONS ON XC4VFX12 VIRTEX-4 FPGA

Device Utilization Summary	PR Design				Non-PR Implementation					
	Static		Dynamic		AES core		SHA-256 core		MODEXP core	
Resource Logic	Used	Utilization	Used	Utilization	Used	Utilization	Used	Utilization	Used	Utilization
Number of Slices	1588	29%	2148	39%	1862	34%	924	16%	499	9%
Number of Slice Flip Flops	1566	14%	1008	9%	807	7%	1008	9%	421	4%
Number of 4 input LUTs	2059	18%	3600	32%	3600	32%	1620	14%	861	9%
Number of DSP48	0	0%	3	9%	0	0%	0	0%	3	9%
Number of FIFO16/RAMB16s	33	91%	0	0%	1	2%	0	0%	0	0%

is not empty or the other queues are not full. The less PR is triggered, the higher the throughput.

Fig. 6 is a 3D representation of the system's total throughput calculated from the time slots assigned for AH and ESP protocols. The graph shows that the maximum throughput is achieved when more ESP packets are processed within a time slot as the AES core has the highest throughput among all three cores. As more time is assigned to AH packets processing, the overall throughput decreases. If the work is evenly divided among both protocols, then the system will achieve a total throughput higher than the maximum throughput of the SHA-256 core but lower than that of the AES core.

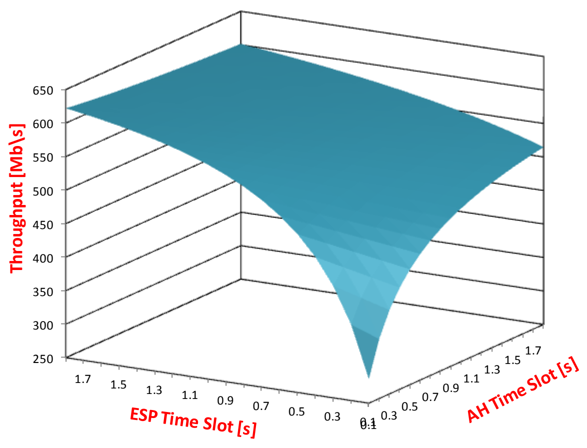


Fig. 6. 3D representation of the overall system performance dependency on processing time assigned to ESP and AH protocols

These results indicates that it is practical to use our design for higher traffic networks where the flow of packets to the input queues allows the scheduler to assign tasks that will minimize PR which will increase the throughput.

VI. CONCLUSION

We implemented a PR design to perform IPSec protocol operations in hardware. The design is divided into a static region representing a Microblaze embedded processor with some supporting peripherals and the dynamic region representing an IPSec co-processor to perform AH, ESP and IKEv2 calculations using hardware accelerators. The results indicate that the PR design shows significant improvements in terms of area savings compared to non-PR designs. These savings in area can be further improved if more cryptographic algorithms supported by IPSec are implemented as RMs in the co-processor. A scheduling algorithm was used to handle task assignments to minimize the effect of additional latency caused by the PR process. Another advantage of this solution can be observed when comparing utilization time of IKEv2 functions against ESP and AH functions. Implementing a hardware accelerator for IKEv2 as a RM not only saves area with almost no time penalty, but it also protects the module from attacks especially if it was preloaded with keys. In the case of ESP and AH hardware accelerators, the traffic flow should be high to minimize PR and increase throughput.

VII. FUTURE WORK

We would like to investigate our system on a Virtex-6 platform and compare its results to this work to see what the impact of the new bitstream authentication is on the overall performance of the Virtex-6. We would also like to investigate how implementing the AES core as part of the static region and only implementing the SHA-256 and MODXP cores as RMs affects the throughput/area of the system. This analysis is interesting because our AES core is comparatively large and AES is used by ESP, AH, and IKEv2.

REFERENCES

[1] RFC-4301, "http://www.ietf.org/rfc/rfc4301.txt," 2005.
[2] RFC-4308, "http://www.ietf.org/rfc/rfc4308.txt," 2005.
[3] RFC-4309, "http://www.ietf.org/rfc/rfc4309.txt," 2005.
[4] J.-P. Kaps, "Cryptography for ultra-low power devices," Ph.D. Dissertation, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2006.
[5] *Early Acess Partial Reconfiguration, User Guide*, Ug208 (v1.1) ed., Xilinx, Inc., Mar 2006.
[6] K. Project, "http://www.kame.net/project-overview.html," 2006.
[7] M. McLoone and J. McCanny, "A single-chip IPSEC cryptographic processor," in *Signal Processing Systems, 2002. (SIPS '02). IEEE Workshop on*, Oct 2002, pp. 133–138.
[8] J. Lu and J. Lockwood, "Ipsec implementation on xilinx virtex-ii pro fpga and its application," in *Reconfigurable Architecture Workshop, RAW*, 2005.
[9] A. P. Kakarountas, H. Michail, A. Milidonis, C. E. Goutis, and G. Theodoridis, "High-speed fpga implementation of secure hash algorithm for ipsec and vpn applications," *The Journal of Supercomputing*, vol. 37, no. 21, pp. 179–195, Aug 2006.
[10] P. R. Schaumont, *A Practical Introduction to Hardware/Software Codesign*. Springer, 2010.

[11] H. Michail, G. Athanasiou, A. Gregoriades, C. L. Panagiotou, and S. Goutis, “High throughput hardware/software co-design approach for sha-256 hashing cryptographic module in ipsec/ipv6,” *Global Journal of Computer Science and Technology*, vol. 10, no. 4, pp. 54–59, June 2010.

[12] Fortinet, “http://www.fortinet.com/products/fortigate/,” 2011.

[13] *An Introduction to the Helion IPsec ESP Engine*, v. 1.0.0 ed., Helion Technology Limited, 2006.

[14] *Sun Crypto Accelerator 4000 Board*, v. 1.1 ed., Sun Microsystems, Inc., 2004.

[15] *Increasing Design Functionality with Partial and Dynamic Reconfiguration in 28-nm FPGAs*, v. 1.0 ed., Altera Corporation, 2010.

[16] *Partial Reconfiguration, User Guide*, Ug702 (v12.1) ed., Xilinx, Inc., May 2010.

[17] A. A. Salman, “IPSec implementation in embedded systems for partial reconfigurable platforms,” Masters Thesis, ECE Department, George Mason University, Fairfax, Virginia, USA, May 2011.

[18] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*. Wiley, 2008.

[19] *ML401/ML402/ML403 Evaluation Platform, User Guide*, Ug080 (v2.5) ed., Xilinx, Inc., May 2006.

[20] K. Gaj and P. Chodowiec, *Cryptographic Engineering*. Springer, 2009, ch. FPGA and ASIC Implementations of AES, pp. 235–294.

[21] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, “Improving sha-2 hardware implementations,” in *Cryptographic Hardware and Embedded Systems - CHES 2006*, Oct 2006, pp. 298–310.

[22] H. Orup, “Simplifying quotient determination in high-radix modular multiplication,” in *Proceedings of the 12th Symposium on Computer Arithmetic*, Jul 1995, pp. 193–199.

[23] M. Joye and S.-M. Yen, “The montgomery powering ladder,” in *Cryptographic Hardware and Embedded Systems CHES 2002,*, ser. Lecture Notes in Computer Science, B. Kaliski, Çetin K.. Koç, and C. Paar, Eds., vol. 2523. Springer-Verlag, 2002, pp. 291–302.

[24] D. Suzuki, “How to maximize the potential of fpga resources for modular exponentiation,” in *Workshop on Cryptographic Hardware and Embedded Systems—CHES 2007*. Berlin: Springer-Verlag, 2007.

[25] E. Öksüzoğlu and E. Savaş, “Parametric, secure and compact implementation of RSA on FPGA,” in *Reconfigurable Computing and FPGAs, 2008. ReConFig '08. International Conference on*, Dec. 2008, pp. 391–396.

[26] *Hardware Interface of a Secure Hash Algorithm (SHA)*, v. 1.4 ed., Cryptographic Engineering Research Group, George Mason University, Jan 2010.

[27] *Secure Hash Standard (SHS)*, National Institute of Standards and Technology (NIST), Oct. 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

[28] *Advanced Encryption Standard (AES)*, National Institute of Standards and Technology (NIST), FIPS Publication 197, Nov 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[29] Openssl, “http://www.openssl.org/docs/apps/openssl.html,” 2009.