Architectural Design Features of a Programmable High Throughput AES Coprocessor

Alireza Hodjat, Patrick Schaumont, Ingrid Verbauwhede
Electrical Engineering Department
University of California, Los Angeles
{ahodjat, schaum, ingrid} @ ee.ucla.edu

Abstract

Programmable, high throughput domain specific crypto processors are required for different networking applications. This paper presents the architectural design features that lead to a multiple Gbits/s rate AES coprocessor, which is programmable with domain specific instructions for Gbit throughput IPSec and other applications. Our design is a loosely coupled, independently working crypto-coprocessor that runs AES in ECB, CBC-MAC, Counter, and CCM modes of operation at a maximum throughput of 3.43 Gbits/s in a 0.18-µm CMOS technology without any penalty in throughput for any of the above modes.

1. Introduction

High throughput security is a significant issue in a large number of applications. High-speed IPSec applications like VPN and Giga-bit Ethernet are examples of such applications that require high performance and flexible security engines. VPN applications that use IPSec require a throughput of over 2 Gbits/s and Giga-bit Ethernet might require a throughput of over 1 Gbit/s. A high-speed CPU does not usually produce multi-gigabits/s throughput because of different factors such as memory bandwidth, cache misses, etc. Table 1 shows the throughput of the AES algorithm on different CPUs clocked at over one GHz. It is observed that the achieved throughput is around or less than one Gbit/s in the case of high-speed CPUs. These throughputs are obtained in ideal circumstances: the AES is the only algorithm running on the CPU and there is no overhead associated with other tasks. Only assembly code optimizations are able to produce a throughput over 1 Gbit/s.

Therefore, multi-gigahertz CPUs are not the solution for achieving multi-gigabits/s security. Instead, we propose the design of programmable, multi-gigabits/s domain specific coprocessors to obtain the required throughput.

IPSec uses the AES (Advanced Encryption Standard [1]) encryption algorithm in different modes of operations [2]. These modes are Counter mode, CBC-MAC, and CCM. Recently some IETF efforts propose the AES encryption algorithm in combination with different modes of operation. Reference [3] describes the use of AES in Counter mode of operation with IPSec and [4] addresses the use of AES in XCBC-MAC with IPSec. Similarly, [5] presents the use of AES in CCM mode (Counter and CBC-MAC) for IPSec. The standard proposals have a tendency to change. Usually these changes are limited to initialization, setup, key management, etc. Therefore, programmability is an important factor that provides support for a wide range of current and future standards for these applications. This imposes design challenges for high throughput, programmable security processors. Domain specialization is the main solution that helps to cover the gap between performance and programmability.

This paper presents the architectural design features of a high throughput, programmable crypto coprocessor that runs the AES algorithm in different modes of operation for IPSec applications. A maximum throughput of 3.43 Gbits/s is achieved at a 295 MHz clock frequency on the crypto-coprocessor using a 0.18 µm CMOS technology. The instruction set includes initialization, key-setup, and AES encryption for different modes of operation. It also includes pipeline instructions that allow AES to run in ECB, CBC-MAC, Counter, and CCM modes of operation in 11 clock cycles per block of 128 bits without any loss in throughput compared to a plain AES without modes of operation.

The rest of this paper is organized as follows: In section 2 the crypto coprocessor architecture and the design principles for high throughput encryption is presented. Section 3 includes the programming interface and section 4 provides the performance results. Section 5 is the conclusion.

Table 1. Throughput comparison for the AES algorithm

Multi-Giga Hertz Machines (from [6])			
Compiler	CPU	Clock freq	Throughput
Assembly	Pentium 4	3.06 GHz	1436.7 Mbits/s
gcc 3.0.2	AMD Athlon	2.25 GHz	861.0 Mbits/s
Assembly	Pentium III	1.33 GHz	718.4 Mbits/s
gcc 3.0.2	Pentium III	1.33 GHz	466.5 Mbits/s

2. Crypto Coprocessor Design Features

This section presents the architectural design features of our programmable, high throughput crypto co-processor. This coprocessor can perform the AES algorithm for ECB, CBC-MAC, Counter, and CCM modes of operation without any loss in throughput as a result of the feedback in the modes (CBC-MAC) or due to other overhead. It is highly programmable with domain specific instructions and is compatible to the IPSec standard proposals.

2.1 The AES Core

Figure 1 shows the inside of the AES core that is used in our crypto coprocessor. It uses 128-bit key and data and performs the encryption in 11 cycles. One round of the algorithm is executed in 1 clock cycle and there are 10 rounds in the128 bit key, 128 bit data AES version. There is one additional clock cycle for the initial key addition phase and after that there is one clock cycle for each round. Therefore, the encryption of one 128-bit block takes only 11 cycles.
The AES core is optimized for critical path and has the least possible delay for one round. The S-boxes of the substitution phase are designed using look-up tables and all the other steps of each round are chains of XORs that are optimized for minimum delay. For the implementation of S-boxes there are other alternatives. We have evaluated some options other than look-up table implementation and found out that the straightforward implementation is the fastest one. An experiment like that is presented in [7].

2.2 Modes of Operation

In almost every security application, AES is used in a different mode of operation. The straightforward AES mode (also called ECB), is not considered secure, because it is vulnerable to statistical attacks [8]. Reference [9] explains the NIST's recommendation for block cipher modes of operation. These modes are ECB, CBC, Counter, CFB, and OFB. There is also a new mode of operation called CCM that is the combination of the Counter and CBC-MAC modes. It only requires the encryption algorithm and is used to generate the encrypted and authenticated data at the same time [10].
Figure 2 shows the datapath of the encryption module of the proposed crypto coprocessor. This figure shows how the AES core is used in the ECB, CBC-MAC, Counter, and CCM modes of operation. The AES core is based on the architecture that was presented in section 2.1. The most important feature is to run the AES core in the above modes of operation without any loss in throughput. There are several enabling factors that provide this capability, explained next.

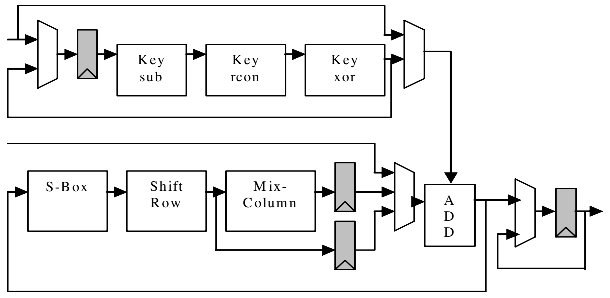

Figure 1: Internal architecture of the AES core

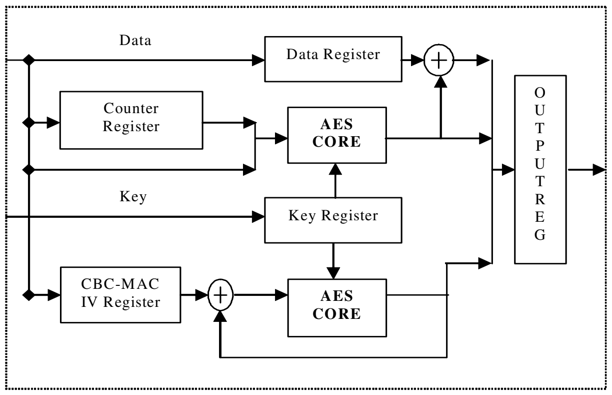

Figure 2: The datapath of the encryption module.

2.3 Separate Data and Control Stream

Figure 3 shows the block diagram of the crypto coprocessor. It includes the input module, encryption module, output module, and the top controller. Encryption module is based on the datapath presented in figure 2. The input and output modules perform the handshaking to read the input and write the encrypted data. Their main task is to read or write the 128-bit block of data from the 32-bit interface. Figure 3 shows how control and data streams are separated in the coprocessor. The flow of data through the co-processor is through the input module, then the encryption module, and then the output module datapaths while the top controller takes care of reading the instructions. The input and output controllers perform their task without any interference of the top controller. Following this methodology, the coprocessor can be programmed to encrypt the stream of input data and produce the output continuously while the top control interface can process new instructions.

2.4 Hierarchy of Control

Design with multiple controllers brings the issue of partitioning the control to different modules. This is important when there are multiple modules that communicate together using a handshake protocol.

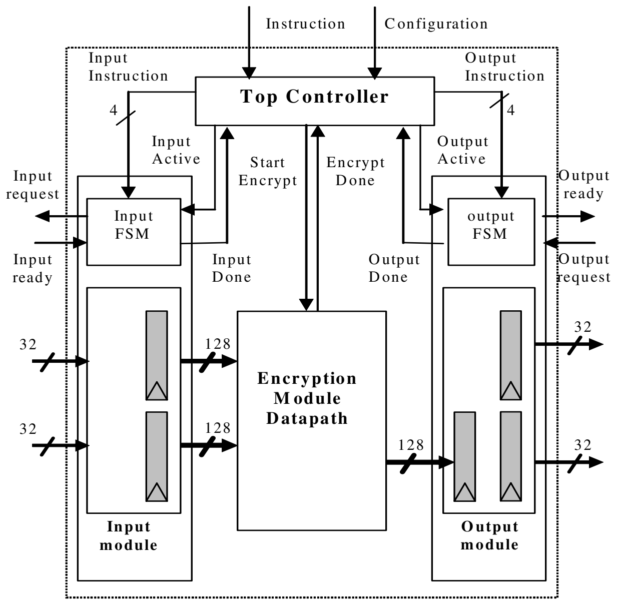

Figure 3: Block diagram of the crypto coprocessor

Hierarchical design of the control is a solution that simplifies the controllers' communications and allows combining high performance and programmability. The top-level control in the system architecture is implemented in the main processor core. The instructions from the main embedded CPU bring the commands to the main controller of the coprocessor. In the next level, the top controller controls the lower level modules.

Figure 4 shows that for the crypto coprocessor of figure 3 there are four control blocks that are controlled by the main control unit. These are the Input FSM, CBC FSM, CNT FSM, and the Output FSM. The Input and Output FSMs perform the handshaking sequence for reading and writing of the data blocks. The CBC FSM controls the sequence to generate a CBC-MAC and CNT FSM controls the encryption sequence for the Counter mode of operation. Depending on the instruction that is read by the main control, it will assert the start signal for one of the sub-controllers. Then the controller (sub-module), starts its operation and will assert the done signal when its operation is finished. This done signal will enable the main controller to reassert the start signal for the following instructions.

2.5 Block Pipelining

Since there is feedback in the CBC-MAC and CCM modes of operation, the AES core cannot be pipelined. On the other hand, a throughput of multiple Gbits/s is required for high throughput applications. This performance is achieved using the block pipelining technique. It is enabled by the hierarchical control design and the handshaking interface between all the modules shown in figures 3 and 4. Figure 5 shows how this pipeline can be designed for AES in the ECB mode of operation. In steady state, encryption is performed on the data while new data is read from the input and the previous encryption result is written to the output. Based on this method, the slowest block decides the cycle time.

In the execution of the AES algorithm, first the input data and the key are added and then there are 10 additional rounds where a new key is generated and added with the byte permuted value of data (see figure 1). Therefore, in the best case, it takes 11 cycles to encrypt one block of data. Using the block level pipelining approach, it is possible to keep this minimum of 11 cycles if the input and output modules consume less than 11 cycles to finish their job. The input and output FSMs are made such that they take less than 11 cycles. Therefore, each pipe stage will take only 11 cycles and one block of output will be ready every 11 cycles.

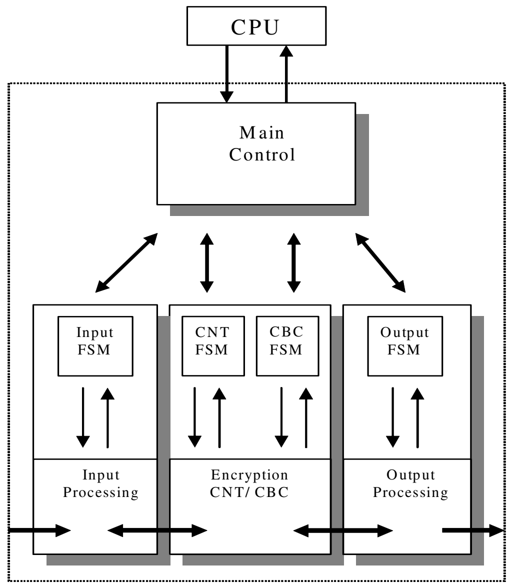

Figure 4: Control Hierarchy

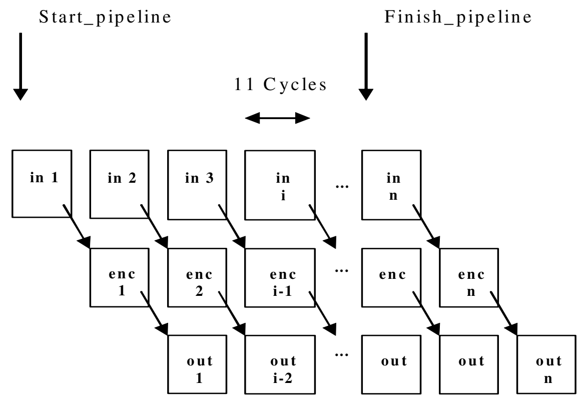

Figure 4: Block Pipeline Model

3. Programmability

This section presents the programming interface and the instruction set of the proposed AES-based domain specific crypto coprocessor. Two different categories of instructions are introduced to provide a flexible instruction set. They are called single and continuous instructions. Single instructions are those that perform one task. After they are done, a new instruction is required for further processing. These instructions can take either one cycle or multiple cycles to execute. One-cycle single-instructions are those that move data from a specific module or register to another module or register. These are required for key set-up and initialization. Examples of multiple-cycle single-instructions are single-block encryption, reading one block of data and/or key, and writing one block of output. Continuous instructions provide high throughput encryption streams using block pipelining. The most important continuous instruction is the *Start_pipeline* instruction that performs block pipeline encryption in any mode of operation. Using these types of instructions, the input stream of data can be encrypted and written to the output continuously, while the control interface (Top Controller) is ready to read a new instruction.

As an example consider the CCM mode of operation. This mode is a complex mode that includes the AES in both Counter mode and CBC-MAC. As shown in figure 2, in this mode each input block goes to both AES CBC-MAC and AES-Counter modules. The CBC-MAC module generates the MAC value using AES in the feedback mode of operation. This value is used for authentication. In parallel, the Counter module encrypts the input payload for confidentiality purposes. Figure 8 shows the pipeline model of this mode. The *Start_pipeline* instruction along with CCM mode configuration starts the pipeline. In steady state, a new block of data is read [I], and a new counter value is encrypted [C]. The previous encrypted counter value is XORed with the previous input, and is written to the output [O]. The previous input block is XORed with the last CBC-MAC value and is encrypted [B]. This continues until the *Finish_pipeline* instruction is inserted. In the last pipe stage, the encrypted MAC value is calculated and is written to the output.

Table 2 presents the list of all instructions that are supported by this coprocessor. There are different types of read input instructions that are defined to load the data and/or key from the input. *Read_32bit_key* reads 32 bits of the key from the 32-bit input port that is connected to the main CPU core. *Read_32bit_data* does the same thing for data. *Write_32bit_out* writes 32 bits of output result into the port connected to the main CPU. On the other hand *Read_block_data* reads a complete 128-bit block of data from the other 32-bit port asynchronously. As shown in figure 3, there are handshaking signals that make it feasible to read the whole block of data asynchronously. Also, *Write_block_out* writes the 128-bit encryption result to the second 32-bit output port asynchronously.

Encrypt_once instruction performs one encryption of the 128-bit input block. This instruction takes 11 cycles. *Start_Pipeline* and *Finish_Pipeline* are the instructions that perform high throughput AES encryption continuously in different modes of operation. *Move_input_2reg* is a special instruction for moving the input data or key to the specific internal registers. *Move_encrypt_2reg* places the encryption result to a specific internal register and *Move_outreg_2out* moves the data stored in the output register to the output module. For most of the instructions listed in table 2, there are special settings through the configuration register. For example, when the *Start_pipeline* instruction is used, the configuration register specifies which mode of operation must be performed-- ECB, CBC-MAC, Counter, or CCM.

Table 2. Instruction Set

Mnemonic	Single or Continuous	Cycles
reset	Single	1
Read_32bit_key	Single	1
Read_32bit_data	Single	1
Write_32bit_out	Single	1
Read_block_data	Single	10
Read_block_key	Single	10
Write_block_out	Single	10
Encrypt_once	Single	11
Move_input_2reg	Single	1
Move_encrypt_2reg	Single	1
Move_outreg_2out	Single	1
Increament_counter	Single	1
Start_pipeline	Continuous	11
Finish_pipeline	Continuous	11

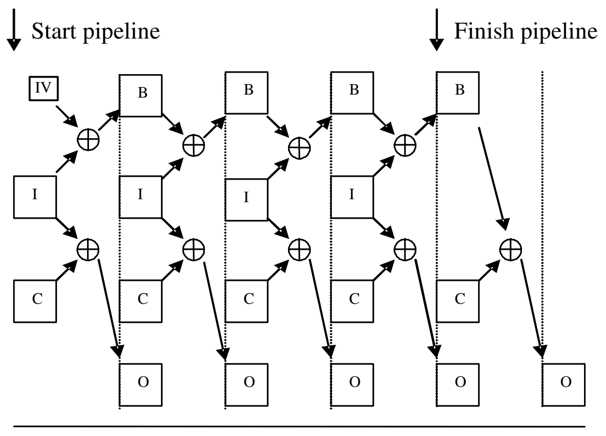

Figure 8: Block pipeline for the CCM mode

4. Performance Results

Table 3 shows the synthesis result of our AES-based coprocessor using a 0.18-µm CMOS standard cell library. Synthesis is done using typical UMC 0.18 library with Synopsys synthesis tool. The conservative model of the wire load is used. Since the core produces a 128-bit output every 11 cycles, the throughput is calculated by multiplying the frequency with 128 divided by 11 which results in the number of bits produced per second. The maximum throughput of 3.43 Gbits/s is achieved at 295 MHz clock frequency. This satisfies the required performance of the high throughput IPSec applications. The power consumption is 86 *mW* at 1.8V and 295 MHz.
Table 4 compares the performance of our design with some other security processors. Although our design is synthesized using a 0.18 µm CMOS technology, its throughput is higher than other security processors that are designed using 0.13 µm or 0.11 µm CMOS technologies. Unique to our design is that it supports a variety of modes of operation for the AES algorithm at 3.43 Gbits/s. None of the other cases support all four modes of ECB, CBC, Counter, and CCM. Also our coprocessor is highly programmable with a set of domain specific instructions

Table 3. Synthesis Results of the crypto coprocessor using UMC 0.18 µm CMOS standard cell library

Optimized for Speed	Critical path	3.38 nsec
	Clock frequency	295 MHz
	Total area	0.732 mm^2
	Gate count	73.2 kgates
	Throughput	3.43 Gbits/s
	Power Estimation	86 mWatts

Table 4. Performance Comparison with other AES based Security Processors

Design	Clock Freq.	Modes of operation	CMOS Technology	Throu-ghput
Our Design	295 MHz	ECB, CBC Counter, CCM	0.18 µm	3.43 Gbits/s
Satoh [11]	224 MHz	ECB CBC	0.11 µm	2.60 Gbits/s
Cavium Networks [12]	500 MHz	ECB only	0.13 µm	2.18 Gbits/s
Hifn HIPPIII 4300[13]	133 MHz	CBC Counter	0.13 µm	2 Gbits/s

5. Conclusion

Separation of control and data streams, hierarchical design of control, pipelining the block modules, designing single and continuous instructions, and considering different modes of operations are the design techniques that are presented in this paper. These techniques help us to design a high-throughput crypto coprocessor, which is programmable with a domain specific instruction set. A loosely coupled, independently working AES-based co-processor is presented that runs in ECB, CBC-MAC, Counter, and CCM modes of operation at a maximum throughput of 3.43 Gbits/s at a 295 MHz clock frequency.

6. Acknowledgment

This material is based upon work supported by the Space and Naval Warfare Systems Center - San Diego under contract No.N66001-02-1-8938. The authors would like to acknowledge the funding of this project.

7. References

[1] National Institute of Standards and Technology (U.S.), Advanced Encryption Standard. Available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] S. Frankel, S. Kelly, R. Glenn, "The AES Cipher Algorithm and Its Use with IPsec", May 2003.

[3] R. Housley, "Using AES Counter Mode with IPSec ESP", Internet Draft, July 2003.

[4] S. Frankel, H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use with IPSEC", March 2003.

[5] R. Hously, "Using AES CCM Mode with IPsec ESP", Internet Draft, July 2003.

[6] http://www.tcs.hut.fi/~helger/aes/rijndael.html

[7] I. Verbauwhede, P. Schaumont, H. Kuo, "Design and performance testing of a 2.29 Gb/s Rijndael processor", IEEE Journal of Solid-State Circuits, March 2003.

[8] A. Menezes, P. Oorschot, S. Vanstine, "Handbook of Applied Cryptography", CRC Press, October 1996.

[9] M. Dworkin, SP 800-38A 2001, "Recommendation for Block Cipher Modes of Operations", Dec. 01.

[10] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: The CCM Mode For Authentication and Confidentiality",NIST special Publication 800-38c, September 2003.

[11] A. Satoh, S. Morioka, K. Takano, S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization", ASIACRYPT 2001, LNCS 2248, 2001.

[12] D. Carlson, D. Brasili, A. Hughes, A. Jain, T. Kisezly, P. Kodandapani, A. Vardharajan, T. Xanthopoulos, V. Yalala, "A High Performance SSL IPSEC Protocol Security Processor", ISSCC 2003, Feb. 03.

[13] http://www.hifn.com/products/4300.html