An FPGA-based AES-CCM Crypto Core For IEEE 802.11i Architecture

Arshad Aziz and Nassar Ikram
(Corresponding author: Arshad Aziz)

National University of Sciences & Technology, Habib Rehmantullah Road, Karachi 75350, Pakistan
(Email: {arshad, nassar}@nust.edu.pk)

(Received Dec. 17, 2005; revised and accepted Jan. 27 & Apr. 24, 2006)

Abstract

The widespread adoption of IEEE 802.11 wireless networks has brought its security paradigm under active research. One of the important research areas in this field is the realization of fast and secure implementations of cryptographic algorithms. Under this work, such an implementation has been done for Advanced Encryption Standard (AES) on fast, efficient and low power Field Programmable Gate Arrays (FPGAs) whereby computational intensive cryptographic processes are offloaded from the main processor thus results in achieving high-speed secure wireless connectivity. The dedicated resources of Spartan-3 FPGAs have been effectively utilized to develop wider logic function which minimizes the critical paths by confining logic to single Configurable Logic Block (CLB), thus improving the performance, density and power consumption of the design. The resultant design consumes only 4 Block RAMs and 487 Slices to fit both AES cores and its key scheduling.

Keywords: AES, CCMP, cryptography, FPGA, verilog

1 Introduction

Cryptography is a fundamental component of any secure system enabling protection of sensitive information. However, the performance of a secure system is affected due to computationally intensive cryptographic transformations. Efficient hardware implementation was therefore, one of the evaluating criteria for AES [11]. The choice of development platforms for embedding cryptographic applications is made considering many factors like the processing power and speed besides, obviously the cost. With the ever increasing computational power vis-a-vis decreasing costs, reconfigurable devices like FPGAs have become attractive platforms for embedding cryptographic applications. Because of outperforming merits, which distinguish FPGAs from other development platforms, there is an increasing trend of their deployment in a number of cryptographic applications. Under our current research work, an efficient FPGA based implementation of AES, especially for the modes involving feedbacks, has been realized. In our sequential design, the functionality for a single round of AES was developed and implemented in hardware, which was then iteratively used to execute all the rounds. For the last round, however, MixColumns function has been bypassed. The sequential designs are compact and best suited for small form factor with low power applications.

We present an outline of our paper first in Section 2, we define new security paradigm for wireless networks. Section 3 gives a brief summary of AES and Section 4 deals with Counter Mode with Cipher Block Chaining Message Authentication Code (CCM). Section 5 presents the system architecture adopted in our implementation. Comparison of our implementation with those done earlier has been presented in Section 6. Section 7 concludes the paper setting directions for further work.

2 New Security Paradigm for Wireless Networks

The IEEE standard 802.11 introduced Wired Equivalent Privacy (WEP) for Wireless LANs [9]. The prime objective of WEP was to defend confidentiality of data from eavesdroppers. Other objectives were to guard against covert modification (i.e., integrity) and provision of access control. WEP utilized the RC-4 encryption algorithm. However, certain flaws were exposed in WEP's intended security goals by researchers from the University of California at Berkeley and Zero Knowledge Systems [12] after which it was realized that a strong security mechanism was needed. The RC-4 encryption algorithm was not necessarily weak; it was the flawed key exchange that compromised the WEP. The new security mechanism thus defined thereafter [7], uses two main developments; Wi-Fi Protected Access (WPA) and Robust

Security Network (RSN). WPA provides important data encryption enhancements to address WEP vulnerabilities, which include a per-packet key mixing function, a Message Integrity Check (MIC) to prevent packet forgeries, an extended initialization vector with sequencing rules, and finally, a re-keying mechanism. The other development i.e. RSN uses dynamic negotiation of authentication and encryption algorithms. Together, these implementations provide a framework for strong user authentication. Deployment of IEEE 802.11i on legacy WLAN client devices and Access Points, however, remains a challenge as the existing hardware does not offer the required computational power required to support the new security mechanisms. In IEEE 802.11i, the security protocol built around AES is called Counter Mode with Cipher Block Chaining-Message Authentication Code (CBC-MAC) Protocol or CCMP. CCMP defines a set of rules that use the AES block cipher to enable the encryption and protection of IEEE 802.11 frames of data. CCMP encrypts/decrypts data at the MPDU (MAC Protocol Data Unit) level as is shown in Figure 1.

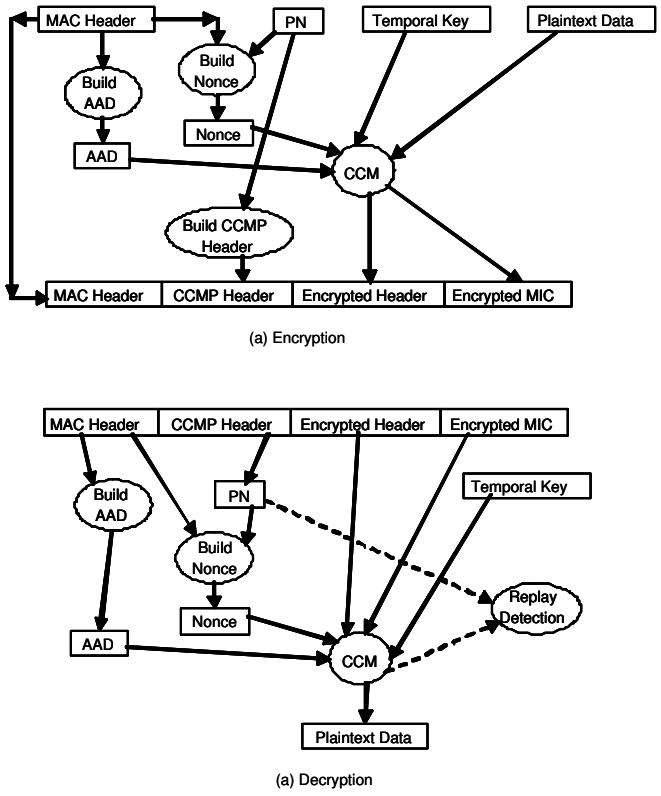

Figure 1: IEEE 802.11i encryption decryption

3 AES Encryption Algorithm

Rijndael algorithm [4] has been selected as the new Advanced Encryption Standard (AES) algorithm [5] by the National Institute of Standards and Technology (NIST) [13]. AES is a symmetric block cipher having variable key and fixed data length. The key lengths can be independently chosen as 128, 192 or 256 bits, which result in 10, 12 and 14 rounds of operation respectively. The data length is however fixed to 128 bits. The input as well as intermediate data can be considered as a matrix with four rows and four columns called state. Each element of the matrix is composed of eight bits, therefore enabling efficient implementation of AES on 8 bit platforms also. The AES algorithm has four basic transformations.

1) SubByte Transformation - a nonlinear transformation applied to the elements of the matrix. This first step in each round is a simple substitution, when implemented as a Look Up Table (LUT). It operates independently on each byte of state using S-box. The byte, $s[i, j]$ become $s`[i, j]$ through a defined substitution table.

2) ShiftRows Transformation - a cyclical shift operation with constant offsets, applied to the rows of the matrix. This second step in each round is permutation of rows by left circular shift; the first (leftmost, high order) i elements of row i are shifted around to the end (rightmost, low order).

3) MixColumns Transformation - the third step is a resource intensive transformation on the columns of state under which the four elements of each column are multiplied by a polynomial, essentially diffusing each element of the column over all four elements of that column.

4) AddRoundKey Transformation - performs modulo 2 (XOR) operation with the round key, which is obtained from the initial key by a key expansion procedure.

The encryption flow starts with the addition of the initial key to the plaintext. Then the iteration continues for (Nr - 1) rounds (Nr being the total number of rounds). In last round the MixColumn step is bypassed as shown in Figure 2.

4 CCM Protocol

CCM is a new mode of operation of a block cipher that combines the existing Counter (CTR) and CBC-MAC modes. It uses encryption algorithm to generate encrypted and authenticated data at the same time [6]. CCM mode was created especially for use in IEEE 802.11i, which is 128 bit AES based. It is intended for packet environments with no attempt to accommodate streams. The AES CCM process is shown in Figure 3.

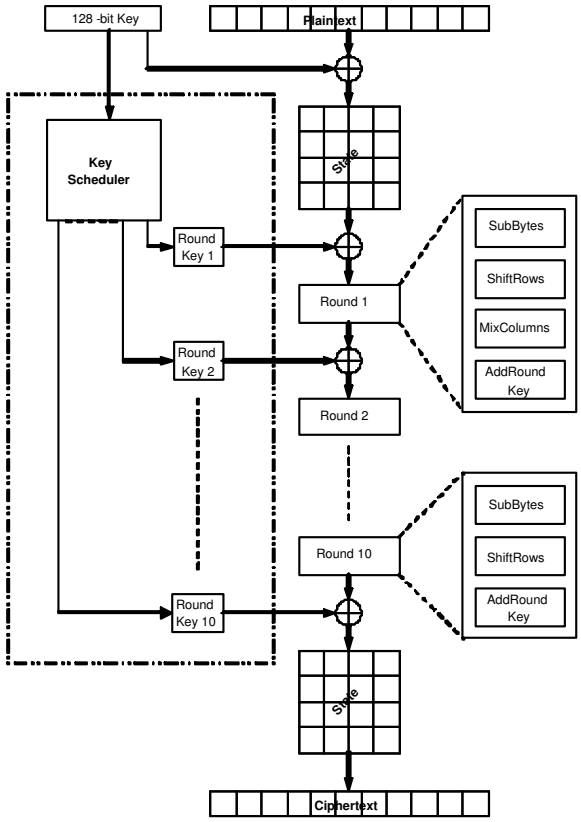

Figure 2: Block diagram of AES encryption process

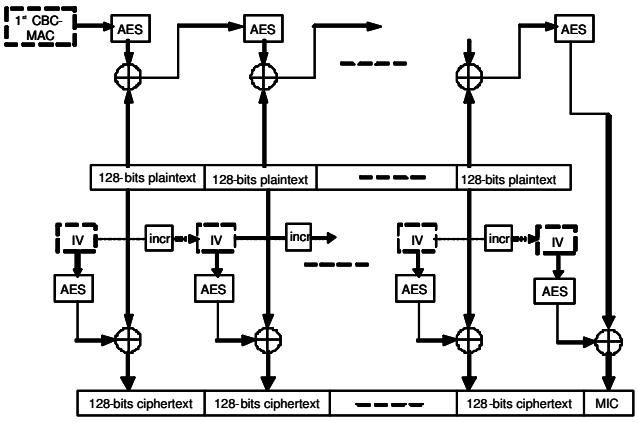

Figure 3: AES-CCM process

ments of IEEE 802.11i. All data paths are 32-bit wide. A 128-bit register is used to store data and each 32-bit is processed at every clock cycle.

5.1 AES Architecture

In an effort to achieve higher throughput, various architectures have been adopted for AES. A few recent efficient implementations are based on pipelined AES architecture [8, 21], offering throughput greater than 30 Gbps. However, these are not viable candidates for applications involving feedbacks in their mode of operations and is therefore meant for applications with non-feedback modes i.e. Electronic Code Book (ECB) and Counter mode only. The latency generated by pipelined systems cannot be tolerated by feedback mode of operation. To provide support for the feedback modes, sequential implementation of AES algorithm has been used in our work. By reducing the critical paths and applying efficient design methodologies, improvement in the performance has been achieved. The design architecture is shown in Figure 4 with four AES transformations duly indicated.

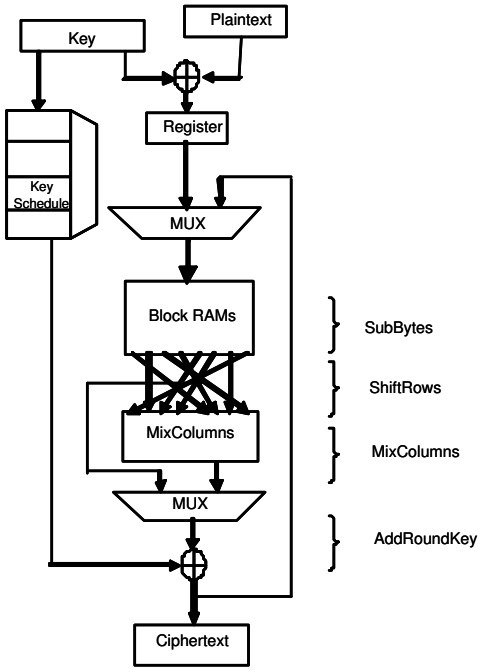

Figure 4: AES datapath

5 System Architecture and Implementation

The AES-CCM process requires two AES cores. In order to achieve higher throughput, two separate AES cores, one for CBC-MAC and the other for Counter Mode are developed. The design is continuation of our previous work [2], but has been optimized to quarter of round approach for meeting the reduced area and power require-

Our design is characterized by adoption of a scaleable architecture, programming the cryptographic functions in Verilog and utilization of reusable hardware functions to avoid duplications. Implementation of byte permutation, used in ShiftRows operation and bitwise addition modulo 2 (XOR) functions, on FPGA are computationally easy. However, the MixColumns function that involves matrix multiplication in GF(2^8) and Byte Substitution are resource intensive when implemented as com-

binational logic; the former generates critical paths and the latter consumes more than 75% of utilized FPGA resources. Implementations of four AES transformations are explained in succeeding subparagraphs.

5.1.1 SubBytes Step

This is implemented as a simple Look Up Table (LUT), which obviously uses a large number of gates. Efficiency entails deployment of large embedded memories for S-box. Each S-box uses 2048 bits of memory and allows 8 bit processes. Implementation of 16 S-boxes for AES requires 32kb memory.

Dedicated embedded memory blocks are ideal for implementing S-box. Our design uses the special feature of Xilinx Spartan-3 FPGA [18] offering multiple block RAMs, organized in columns. Each block RAM contains 18kb of fast static RAM [19]. The Xilinx Spartan-3 xc3s50pq208-5 has 4 block RAMs. These block RAMs can either be used as Single or Dual port RAM. The single Dual port can also be configured as per block RAM only as shown in Figure 5, therefore utilizing each Dual port RAM block as two single port RAMs. With quarter of round approach, we are processing 32 bits at a time which require 4 S-boxes, realizable in only 2 block RAMs.

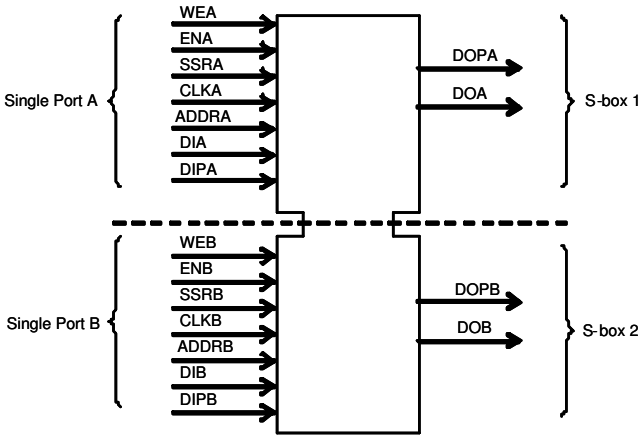

Figure 5: One block RAM becomes two independent single-port RAMs

5.1.2 ShiftRows Step

The ShiftRows transformation can be expressed as an arrangement of the matrix using an address expression for each element. The address expression calculates row dependant circular shift of rows. The circular shift operation uses routing only, which entails some routing delays but no dedicated hardware resources in term of gates are required.

5.1.3 MixColumns Step

The MixColumns transformation maps one column of the input state to a new column state. The transformation is based on a four-byte input. Multiplication is costly both in terms of area and delay. To architecturally optimize the multiplication, our design uses the mathematical properties of multiplication by a fixed constant in GF(2^8) optimally using combinational logic circuit. Multiplication used in MixColumn transformation is given as:

$$\{03\} \bullet B(x) = (x+1)B(x) \text{ and } \{02\} \bullet B(x) = xB(x),$$

where

$$B(x) = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + b_6x^6 + b_7x^7.$$

The resultant multiplications are

$$\begin{aligned} \{03\} \bullet B(x) \quad = \quad & (b_0 \oplus b_7) + (b_0 \oplus b_1 \oplus b_7)x + (b_1 \oplus b_2)x^2 \\ & + (b_2 \oplus b_3 \oplus b_7)x^3 + (b_3 \oplus b_4 \oplus b_7)x^4 \\ & + (b_4 \oplus b_5)x^5 + (b_5 \oplus b_6)x^6 + (b_6 \oplus b_7)x^7 \end{aligned} \tag{1}$$

and

$$\begin{aligned} \{02\} \bullet B(x) = b_7 \quad + \quad & (b_0 \oplus b_7)x + b_1x^2 + (b_2 \oplus b_7)x^3 \\ + \quad & (b_3 \oplus b_7)x^4 + b_4x^5 + b_5x^6 + b_6x^7 \end{aligned} \tag{2}$$

Implementations of above equations are simple being XOR operations. The optimized hardware implementation considers its four byte input as a polynomial over GF(2^8) and is capable of performing a multiplication of the input with the constant polynomial. This zero clock cycle parallelism reduces the entire MixColumns to combinational logic.

5.1.4 AddRoundKey Step

The key scheduling, presented in Figure 6 uses on the fly key generation. The initial key is 128-bit and the round keys are expanded from the initial key by XORing of previous column. For columns that are in multiple of four, the process involves Shift operation (Rot), byte substitution (S-box) and a round constant (Rcon) addition. On the fly key generation method produces the keys needed for the round at every clock cycle and does not store all the keys in the register, thus decreasing the resources needed for overall design.

5.2 Efficient FPGA Resources Utilization

The architectural optimizations discussed in Section 5.1 reduced the hardware overheads both in S-box and MixColumn, but the critical paths were still high and reduced the over all performance and increased the power consumption of the design. Modern generation of FPGAs

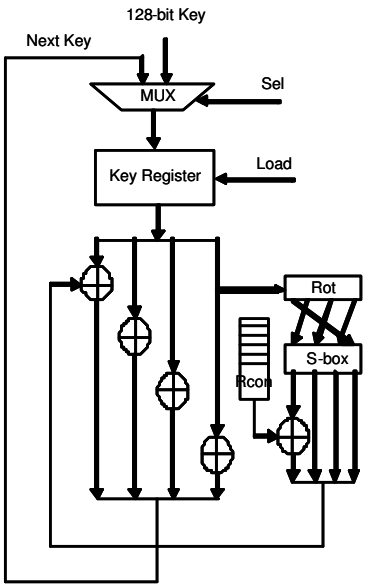

Figure 6: Key scheduling

haves several interesting features [1, 16], which can be utilized to improve the performance, density and power consumption of the design by reducing the critical paths. A further improvement is made by confining the logic within single CLB, where possible. Xilinx specialized tools like CORE Generator can also be effectively utilized to further improve the performance.

5.2.1 Wider Logic Function

The Spartan-3 FPGAs are based on a 4-bit input LUTs. One possible way to implement functions that are wider than 4-bit inputs is to cascade the multiple LUTs. For example, a 16-bit input function could be built by combining the outputs of two LUTs into a third LUT. Without special resources and implementation techniques, logic functions would consume seven LUTs as well as add three levels of logic delays plus two levels of routing delays. Unfortunately this method adds logic delays plus an additional routing delay between the LUTs.

The Spartan-3 FPGA architecture includes dedicated multiplexers (MUX) within the CLBs. These specialized multiplexers are effectively utilized in our implementation for deploying wide input functions to improve the performance and density of design.The critical paths in the design are minimized by using logic wider than 4-bit input and confining it to a single CLB. To increase design speed and density, we use Spartan-3 FPGAs dedicated MUXs named F5MUX, F6MUX, F7MUX and F8MUX in CLB [18]. These MUXs are used to replace additional levels of LUT-based logic and produce the function wider than 4-bit without cascading the LUTs. Each slice is identical with respect to logic and MUX resources.

In our design the F5MUX in Slice S0 to S3 is used to develop a 9-bit wide logic function within each slice. The F5MUX combines the two LUTs in a slice. If the two LUTs contain independent functions, the select input of F5MUX becomes the ninth input of the function (four inputs from each LUT plus one select input). The slice S0 and S2 have F6MUX. The output of each previously F5MUX connected to input of F6MUX which result in a 19-bit wide logic function. To further widen the logic, the output of two F6MUX is combined using F7MUX present in slice S1, the resultant is a 39-bit input wide logic function as shown in Figure 7.

A significant benefit of the dedicated multiplexers is the dedicated routing that connects between levels, the connections from the LUTs to the MUXs and between the MUXs is dedicated and has zero routing delay. The combination of LUTs and dedicated multiplexers results very efficient implementation wider logic without any delay.Thus by using the wider input logic functions the logic is confined within single CLB, which minimize critical paths of the design to great extent.

5.2.2 Logic Optimization with Core Generator

The CORE Generator tool [20] is a component of the Xilinx Foundation series software used in our design. It enables us to configure a parameterized design for Xilinx devices. It produces an optimized, predefined set of building blocks for common functions and enables faster design completion. The CORE Generator takes full advantage of Xilinx's core friendly FPGA architecture such as LUTs and other resources already available on FPGA. It also enables relative location constraints and expert logic mapping and floor planning to optimize performance of a design and delivers a high level of performance and area efficiency.

For our S-box, it generates an Electronic Data Interchange Format (EDIF) netlist (EDN file), a Verilog template (VEO) file with a Verilog (V) wrapper file, a netlist file (NGC), and an output file (NDF). The CORE generator EDIF file provided during the synthesis, contains all the path timings. Based on the surrounding logic, it efficiently alters the timing constrains. The critical paths confined with in design and results in better performance. Without adding the CORE Generator EDN file to the design Xilinx post PAR (place and route) result in lower clock frequency. However, when we added the CORE Generator EDN file to the synthesis process, the clock frequency increase significantly because of additional path optimization performed by it. The resultant optimized design is compact with minimize interconnects, improved performance and reduction in power requirements.

5.2.3 Manual Place & Route

However, for routing we have specified area constraints so that the design placement and routing is performed with minimal lengths for routes. By doing so, we restrict Xilinx Synthesis tool from automatically routing design components using long wires. This way we reduce the

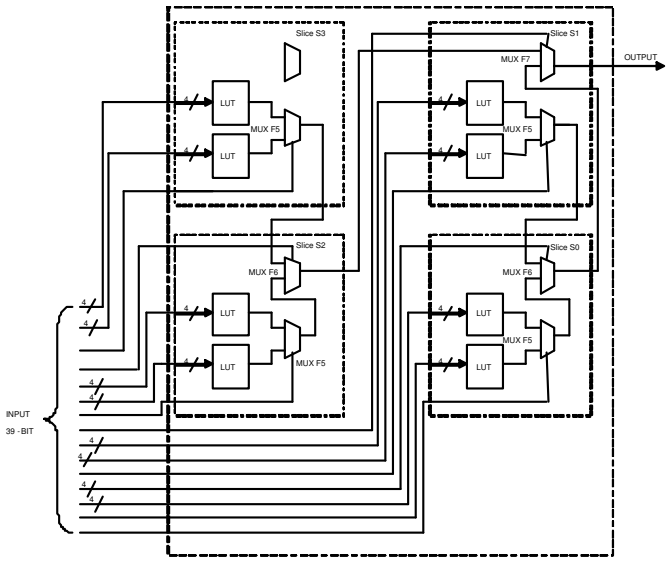

Figure 7: Wide input logic function

delay contributed by routing interconnects.

5.3 AES CBC-MAC Core

In AES CBC-MAC, the first AES core is working in cipher feedback mode. It XORs the new input to previously encrypted data. The core is used to calculate the MIC for the authenticity of data. The process starts with encrypting the first block and then successively XORs subsequent blocks and encrypts the result. The final MIC is one 128-bits block. The Header Analyzer supplies the input values Flag, Data Length (DLen) and nonce required to calculate the first blocks for CBC-MAC. The Nonce is the combination of Packet Number (PN) and Media Access Control (MAC) address as shown in Figures 8 & 9. This value is stored in register and then each 32-bit from this register is processed by the core using AES transforms. Once the first block has been prepared, XORing the current block with previously encrypted block computes the MIC one block at a time. If the last block is not exactly 128-bit, it is padded with zeros. The final output is one 128-bit block, but the CCM requires only a 64-bit MIC, so the low order 64-bit of final output is discarded.

5.4 AES Counter Mode Core

In AES Counter Mode, the second core is used to generate the Ciphertext for the IEEE 802.11i architecture. Once the MIC is calculated by AES CBC-MAC core, it is appended to the plaintext data. The Header Analyzer supplies inputs values Flag, DLen, nonce and counter, required to calculate the Initial Value (IV) for AES CTR. The Nonce is the combination of PN and MAC address, the counter starts at 1 and counts up as counter mode proceeds, as shown in Figures 8 & 9 it encrypts the initial value (IV) and then XORs this data with the next plaintext block and so on to produce encrypted data. This value is stored in register and then each 32-bit from this register is processed by the core using AES transforms.

5.5 Core Encryption and Decryption Process

The Encryption and the Decryption processes of the crypto core are shown in Figures 8 & 9, respectively. The Encryption process uses AES CBC-MAC core to generate the MIC and the AES Counter core for the encryption of data. The Key Scheduler provides the fresh Temporal Key (TK) for every new session. In AES-CCM, decryption is almost identical to encryption. It first decrypts the data using AES Counter Mode core and calculates the MIC using AES CBC-MAC to compare both the results for data integrity.

6 Implementation Results and Comparison

The design has been authored in Verilog HDL and its Synthesis done with Xilinx XST. Xilinx ISE Foundation 6.3i has been used for performing mapping, placing and routing. For functional simulation and testing, ModelSim 6.0 has been used. The Synthesis tool was configured to optimized for the area and high effort consideration. The targeted device was Spartan-3 xc3s50pq208-5 with

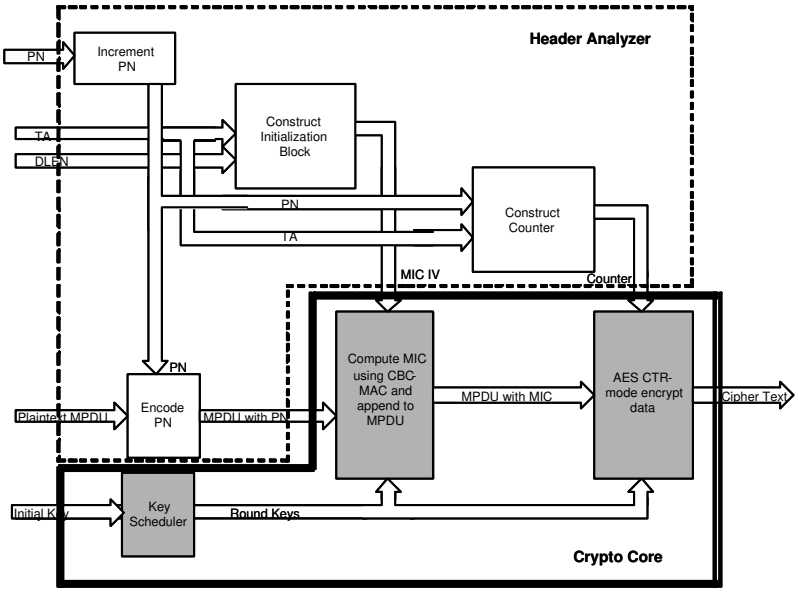

Figure 8: CCM encryption process

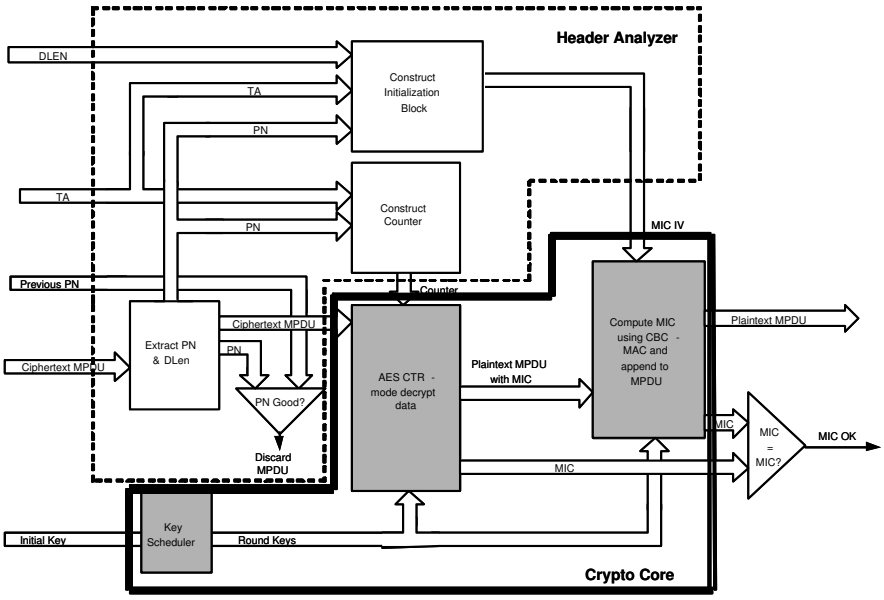

Figure 9: CCM decryption process

Table 1: Implementation result

CORE	Slices	BRAM	Frequency	Throughput
AES-CTR	143	2	247MHz	687.30Mbps
AES-CBC	151	2		
Key Scheduler	193	0		
Total	487	4		

detailed specifications at [18]. Our design occupies 487 Slices for both AES cores and key scheduler (63nd 4 Block RAMs. Operating at frequency of 247 MHz, throughput of 687.30 Mbps has been achieved. The detail device utilization is shown in Table 1. The core has a latency of 46-clock cycles.

There exist few reported implementations of AES-CCM both in academia and industry. The two commercially available compact cores, one by CAST [3] utilizes Spartan-3 x3s200-5 FPGA using 414 slices (support one mode of operation at a time) working at a frequency of 147MHz , the other one by SiWork [14] utilizes 835 Slices of Virtex2 FPGA, working at a frequency of 100 MHz produces a throughput of 582 Mpbs. The implementation by Y. Mitsuyama et al. [10] is on ASIC, working at a frequency of 85 MHz and has a throughput of 54 Mbps. The implementation by K. Vu et al. [17] and N. Sklavos et al. [15] are using Spartan-II and Virtex devices and has a throughput of 258.8 Mbps and 177 Mps respectively. The comparison with previously discussed implementations clearly indicates that our implementation of AES-CCM is more efficient both in term of speed, throughput and device utilization on Spartan-3 device.

7 Conclusion

The field of cryptography is perpetually changing; new algorithms, new techniques of implementation are constantly being derived. Keeping in view the present scenario of FPGA implementation of AES-CCM, the paper gives great insight into an efficient and low power implementation of AES block cipher to satisfy the security requirement of IEEE 802.11i Architecture. Our implementation makes headway in the midrange bandwidth applications as it provides the flexibility of lower power consumption and cost, employing AES in feedback modes. Offering encryption rate of 687.30Mbps for CCM, it not only meets current IEEE 802.11a/b/g operating data requirements, but also high speed of emerging wireless standard like IEEE 802.11n. The core provides privacy, integrity, authenticity and replay protection of data and is also applicable to other IEEE wireless standard like 802.15 and 802.16 and can also be adopted for IPSec implementation.

Future work includes further resource optimization to achieve lower power and higher throughput.

References

[1] A. Amara, F. Amiel, and T. Ea, "FPGA vs. ASIC for low power applications," *Microelectronics Journal*, in Press, Corrected Proof, Jan. 2006.

[2] A. Aziz and N. Ikram, "An efficient FPGA based sequential implementation of advanced encryption standard," in *proceedings of IEEE ITI 3rd International Conference on Information and Communication Technology (ICICT 2005)*, pp. 875-882, Cairo, Egypt, Dec. 2005.

[3] *CAST AES-CCM Core Product Brief.* (http://www.cast-inc.com/co res/aes_e/cast_aes_e-x.pdf)

[4] J. Daemen and V. Rijmen, *AES Proposal: Rijndael, AES algorithm submission*, Sept. 1999. (http://www.nist.gov/CryptoToolkit)

[5] *Draft FIPS for the AES*, Feb. 2001. (http://csrc.nist.gov/encryption.aes)

[6] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CCM Mode For Authentication and Confidentiality*, NIST special Publication 800-38C, May 2004.

[7] J. Edney and W. A. Arbaugh, *Real 802.11i Security Wi-Fi Protected Access and 802.11i*, Addison-Wesley, Aug. 2003.

[8] M. Jing, Z. Chen, J. Chen, and Y. Chen *Reconfigurable System for High-Speed and Diversified AES using FPGA*, *Microprocessors and Microsystems*, in Press, Uncorrected Proof, Mar. 2006.

[9] LAN MAN Standards Committee of the IEEE Computer Society, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, ANSI/IEEE Std 802.11, 1999.

[10] Y. Mitsuyama, M. Kimura, T. Onoye, and I. Shirakawa, "Architecture of IEEE802.11i cipher algorithms for embedded systems," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol E88-A. no. 4, pp. 899-906, 2005.

[11] J. Nechvatal, E. Basker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, *Report on the Development of the Advanced Encryption Standard (AES)*, NIST, Computer Security Division, Information Technology Laboratory, Oct. 2000.

[12] B. Nikita, G. Ian, and W. David, *Security of the WEP Algorithm*, University of California at Berkeley, Feb. 2001.

[13] National Institute of Standards and Technology. (http://csrc.nist.gov)

[14] *SiWork AES-CCM Core Product Brief.* (http://www.siworks.com)

[15] N. Sklavos, G. Selimis, and O. Koufopavlou, "FPGA implementation cost & performance evaluation of IEEE 802.11 protocol encryption security Schemes," in *Second Conference on Microelectronics, Microsystems and Nanotechnology, (MMN'04)*, pp. 361-364, Athens, Greece, Nov. 2004.

[16] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90nm low-power FPGA for battery-powered applications," in *International Symposium on Field Programmable Gate Arrays*, pp. 3-11, Monterey, California, USA, Feb. 2006.

[17] K. Vu and D. Zier, *FPGA Implementation AES for CCM Mode Encryption Using Xilinx Spartan-II*, ECE-679, Oregon State University, Spring 2003.

[18] Xilinx, *Spartan-3 Field Programmable Gate Array data sheets.* (http://www.xilinx.com/spartan3)

[19] Xilinx, *Using Block RAM in Spartan-3 FPGAs.* (http://www.xilinx.com/xapp/xapp463.pdf)
[20] Xilinx CORE Generator. (http://www.xilinx.com/products/design_tools/logic_design/design_entry/coregenerator.htm)
[21] S. Yoo, D. Kotturi, D. Pan, and J. Blizzard "An AES crypto chip using a high-speed parallel pipelined architecture," *Microprocessors and Microsystems*, vol. 29, no. 7, pp. 317-326, Sep. 2005.

Arshad Aziz is a Ph.D. candidate in electrical engineering at National University of Sciences & Technology, Karachi, Pakistan. He received his M.S. and B.S. in Computer Engineering in the year 2002 and 1998 respectively from Sir Syed University of Engineering & Technology, Karachi, Pakistan. His research interests include Network Security, Cryptography and Digital Design.

Nassar Ikram graduated in Electrcial Engineering from NED University, Karachi, Pakistan in 1986. He completed his MSc from Cranfield University in 1995. In 1999, he attained his PhD from Bradford University in the field of Cryptography. Since then, he is serving at National University of Sciences and Technology, Pakistan