Design and VLSI Implementation of Low Latency IEEE 802.11i Cryptography Processing Unit

Jun-Dian Li and Chih-Peng Fan

***Abstract*—IEEE 802.11i is the important security standard for wireless local area network, and it includes three security functions, which are WEP, TKIP, and CCMP, to provide the data confidentiality. In this paper, the effective cipher architecture of IEEE 802.11i is developed to achieve the low-latency application. For the cryptography processing functions, the cipher core of WEP and TKIP is the RC4 algorithm, and that of the CCMP is the AES algorithm. For a ciphered packet by WEP and TKIP, the RC4 operations need a constant latency, which generates the excessively low throughput when the packet length is too short. For the low-latency design, the 16-bit packed memory algorithm is applied to reduce the constant latency in the RC4 computations. To reduce the hardware cost of CCMP for the byte-wise data transmission in IEEE 802.11, the 32-bit AES architecture is used in place of the conventional 128-bit AES design. For VLSI implementation, the proposed low-latency IEEE 802.11i cryptography processing architecture is synthesized by Synopsys Design Compiler with TSMC 0.18um technology. Excluding the cost of memory module, the proposed design for cipher computations requires about 44,300 gate counts, and the maximum operational frequency is 51MHz. Besides, the power consumption of the processing unit at 50MHz is 12.61mW.**

***Index Terms*—IEEE 802.11i, cryptography, low latency, VLSI implementation.**

I. Introduction

IEEE 802.11i is the security standard and specification of wireless local area network, and it defines three algorithms which are related to the data confidentiality, that is, WEP, TKIP, and CCMP. The cipher core of the WEP and TKIP is the RC4 algorithm, and that of the CCMP is the AES algorithm. Both of the RC4 and AES algorithms are the symmetric ciphers, whose feature is that the transmitter and the receiver must share one same secret key to achieve the data confidentiality. Fig. 1 depicts the simplified model of conventional symmetric encryption in [1]. IEEE 802.11 [2] is a standard for Wireless Local Area Network (WLAN), and it defines a medium access control (MAC) layer and a physical (PHY) layer. Since the data transmission of wireless local area network is through the air, the transmitted data is easily eavesdropped. Thus, IEEE 802.11 defines a data security mechanism, i.e. Wired Equivalent Privacy (WEP), which aims to provide an equivalent data transmission security, which is also provided by traditional wired networks. However, in recent years, many cryptanalysts have found that WEP has many weaknesses. Therefore, in 2004, IEEE 802.11i [3] emerged to enhance the security of wireless local area networks.

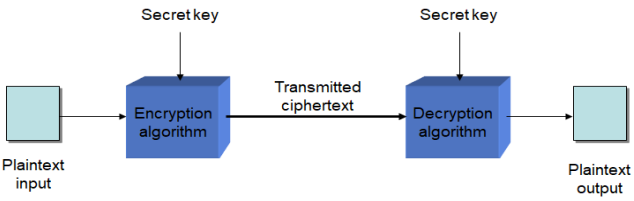

Fig. 1. Simplified model of conventional encryption [1].

The IEEE 802.11i network defines two types of security algorithms, which are the RSNA (Robust Security Network Association) algorithm and the Pre-RSNA algorithm. The Pre-RSNA security mechanism includes two algorithms, i.e. WEP and IEEE 802.11 entity authentication. On the other hand, the RSNA security mechanism includes the following algorithms, which are (1) Temporal Key Integrity Protocol (TKIP), (2) CTR with CBC-MAC Protocol (CCMP) [4], (3) RSNA establishment and termination procedures including the use of IEEE 802.1X acknowledgments, and (4) key management procedures. In addition to the open system authentication, all Pre-RSNA security mechanisms are not suggested because they do not meet the expected security goals. Although the new implementations need to support the Pre-RSNA approach, the use of Pre-RSNA just helps to migrate the system to the RSNA method. In the RSNA security mechanism, TKIP and CCMP are main protocols and functions for data confidentiality and integrity. In all IEEE 802.11 devices that claim to comply with RSNA, the implementation of CCMP is mandatory, and TKIP is not necessary because the confidentiality and integrity of TKIP are not as strong as those of CCMP. In the previous works [5], and [6], the efficient hardware of the RC4 stream cipher were implemented for low-latency applications. To realize the CCMP mode in IEEE 802.11i, the effective AES-based ciphering architectures were designed in [7]-[9]. For the low-power and high-throughput implementation of the IEEE 802.11i security functions, the performance analysis, the hardware-software co-design, or the FPGA/VLSI-based design were revealed in [10]-[14].

In this paper, the proposed hardware design implements the related data encryption and decryption functions, i.e. WEP, TKIP and CCMP. TKIP is mainly used as a short-term mechanism to improve the deficiency of WEP. For the system that only supports WEP, the data security can be

Manuscript received August 19, 2019; revised May 13, 2020. This work was financially supported in part by the Ministry of Science and Technology (MOST) under Grant No. MOST 108-2634-F-005-002 and by the "Innovation and Development Center of Sustainable Agriculture" from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

Jun-Dian Li and Chih-Peng Fan are with the Department of Electrical Engineering, National Chung Hsing University, 145 Xingda Rd., South Dist., Taichung City 402, Taiwan (e-mail: jundian@gmail.com, cpfan@dragon.nchu.edu.tw).

doi: 10.18178/jacn.2020.8.1.274

enhanced through the firmware update on hardware. CCMP is a new security mechanism proposed by IEEE 802.11i, which serves as a security mechanism for long-term wireless local area networks. The well-known Advanced Encryption Standard (AES), adopted by the encryption and decryption core of CCMP, is the popular-used and efficient symmetric block cipher scheme [1], [15]. The rest of the paper is described as follows. In Section II, the previous IEEE 802.11i hardware-based design is briefly reviewed. In Section III, the low-latency design of the proposed IEEE 802.11i cryptography processing unit is discussed. The VLSI implementation results and discussions are described in Section IV. Finally, a conclusion is stated.

II. Brief Review of Previous IEEE 802.11i Hardware Design

The functions of data encryption and decryption are achieved through WEP, TKIP, or AES-CCM. However, the computational cost of these algorithms is too high, and it is difficult to achieve sufficient data throughput by the implementation with embedded software. Therefore, in [10], the hardware-based IEEE 802.11i encryption with low-cost and low-power consumption is designed, and the ciphering data can achieve the maximum transmission rate for the application of IEEE 802.11a/g.

To achieve the maximum transmission rate of IEEE 802.11a/g when the data encryption and decryption is active, the computations of WEP, TKIP, and AES-CCM need at least 220MHz, 440MHz and 3.5GHz clock frequency for the ARM9 processor, respectively [10]. However, the maximum operational frequency of the ARM9 processor is 250MHz. Obviously, it is very difficult that only pure software-based implementation is utilized for the real-time IEEE 802.11i encryptions and decryptions in the IEEE 802.11a/g wireless network. Besides, at the WEP and TKIP modes, when the packet data is shorter, the computational cost becomes higher. The reason is that because the shuffling process of the cored RC4 algorithm performs initialization for each packet key, and it takes up most of the processing cycles of encryptions, as shown in Table I. According to the analysis in [10], the hardware based architecture design for the IEEE 802.11i encryption algorithms are needed.

TABLE I: Computational Cost (Mcycles/sec) of WEP [10]

packet data length (Byte)	computational cost（Mcycles/sec）			required clock freq.（MHz）
	shuffling	substitution	others	
100	583	54.1	13.5	651
500	218	109	27.4	354
1,000	126	120	30.0	276
2,000	68.4	126	32.4	227

Since RC4 is a stream encryption method, the unit of processing data is 1 byte, and the RC4 key is used for the shuffling process and substitution. However, as mentioned above, for shorter packet data lengths, the RC4 shuffling process occupies most of the encryption cycles. Thus, the designers in [10] proposed a way to reduce the encryption cycles of RC4 shuffling operations. Reducing the shuffling cycles of RC4 means that the system reduces the number of accesses to the memory. However, the data dependence makes the parallel processing impossible. In [10], a 16-bit based memory access method is developed to complete the shuffling process, which reduces the number of memory accesses while avoiding the dependency of intermediate data. In Table II, the 32-bit memory access scheme is only 20% less than the 16-bit memory access scheme does; however, the 32-bit based memory access design requires more complex control, and it results in more hardware cost, compared with the 16-bit memory access scheme. Therefore, the 16-bit packed memory access method is adopted in [10], and its memory access number is 30% less than the number of the 8-bit memory access scheme.

TABLE II: Comparison of Memory Access Numbers [10]

Bus width	Memory access numbers
8 bits	1,282
16 bits	896
32 bits	704

On the other hand, for the encryption and decryption process of AES-CCM in [4], [10], [15], and [16], two AES processing hardware are required, where one is as the processing of the CBC-MAC mode, and the other is as the processing of the counter mode. To achieve higher data throughput, the overlapped pipelined processing for the CBC-MAC and counter modes is used in [10]. The AES processing unit in [10] is based on the 32-bit data-path and multi-cycle byte-by-byte shifters, and it can support the maximum data throughput of IEEE 802.11a/g at the 40 MHz clock frequency. Finally, the WEP/TKIP and AES-CCM operations require 2,048 bits of memory to store the expanded key and the round key. Since the IEEE 802.11i encryption algorithm is used alternately, the WEP/TKIP and AES-CCM operations shares a single 128×16-bit memory.

III. Proposed Architecture Design for Low Latency IEEE 802.11i Processing Unit

Fig. 2 illustrates the architecture design of the proposed IEEE 802.11i processing unit (i.e. core11i). For the low-latency 802.11i cryptography processing circuit, firstly the overall architecture is described. Next, the architecture design of the WEP/TKIP modes is introduced. Finally, the applied CCMP architecture CCMP is discussed. Table III lists the input/output signal descriptions of the proposed core11i module.

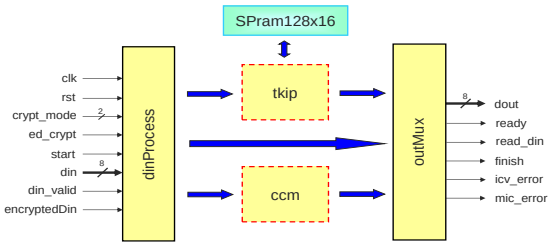

Fig. 2. Architecture design of the proposed IEEE 802.11i processing unit (core11i).

In Fig. 2, the “dinProcess” module is the input data processing unit, and the “tkip” module is the WEP/TKIP encryption and decryption unit, and then the “SPram128x16” module is a single-port 128×16-bit memory. Moreover, the

"ccm" module is designed for the CCMP encryption and decryption unit, and the "outMux" module is designed for the output multiplexer between the modules. The "dinProcess" module determines which module the data should be sent to and generates the corresponding control signals based on the order of the input data. Table IV defines the operational modes and order of input data for the encryption and decryption of the WEP, TKIP, and CCMP modes.

TABLE III: INPUT / OUTPUT SIGNAL DESCRIPTIONS OF THE PROPOSED CORE11I MODULE

Name	Type	Descriptions
clk	Input	System clock signal (positive edge trigger)
rst	Input	System reset signal (synchronous reset)
crypt_mode[1:0]	Input	0: Chip Disable, 1: WEP, 2: TKIP, 3: CCMP
ed_crypt	Input	0: encryption, 1: decryption
start	Input	This signal is 1 to start to process the input data
din[7:0]	Input	Input data
din_valid	Input	If this signal is 1, there is a valid data on the input
encryptedDin	Input	This signal is 1 for the input data as the encrypted MIC value and the encrypted ICV value. This signal is only used in the WEP and TKIP decryptions
dout[7:0]	Output	Output data
ready	Output	This signal is 1 when the output data is valid.
read_din	Output	Read request for Input data
finish	Output	This signal is 1 means that all data has been processed.
icv_error	Output	If this signal is 1, the ICV check result is different for decrypting. The signal is only used in the WEP and TKIP decryptions
mic_error	Output	If this signal is 1, the MIC check result is different for decrypting. This signal is only used in the TKIP and CCMP decryptions

TABLE IV: OPERATIONAL MODES AND ORDER OF INPUT DATA OF THE PROPOSED IEEE 802.11I PROCESSING UNIT

Mode	Order of input data
WEP encryption	5-byte WEP key, 4-byte IV field, N-byte Plaintext
WEP decryption	5-byte WEP key, 4-byte IV field, N-byte Ciphertext, 4-byte E(ICV)
TKIP encryption	16-byte TK, 8-byte MIC key, 6-byte TA, 6-byte DA, 6-byte SA, 4-byte IV field, 4-byte extended-IV field, N-byte Plaintext
TKIP decryption	16-byte TK, 8-byte MIC key, 6-byte TA, 6-byte DA, 6-byte SA, 4-byte IV field, 4-byte extended-IV field, N-byte Ciphertext, 8-byte E(MIC), 4-byte E(ICV)
CCMP encryption	16-byte TK, 2-byte MLng, 24- or 30-byte MAC header, 8-byte CCMP header, N-byte Plaintext
CCMP decryption	16-byte TK, 2-byte MLng, 24- or 30-byte MAC header, 8-byte CCMP header, N-byte Ciphertext, 8-byte E(MIC)

In the proposed hardware architecture, the encryption and decryption functions of TKIP and WEP are integrated into a single module. In Fig. 3, the "michael", "mic2crc" and "micCpr" modules are only used in the TKIP mode. The "mic2crc" and "p2s" modules are used for the WEP and TKIP encryptions, while the "micCpr" and "icvCpr" modules are used for the WEP and TKIP decryptions. Except the above mentioned modules, all of the rest modules will be used whether the WEP or TKIP mode is active or not. Fig. 4 shows the applied architecture for the RC4 operations, and it is based on the architecture design in [10]. The design goal of the RC4 ciphering module is to generate an 8-bit key stream by using a single-port 128×16-bit memory. The RC4 stream encryption is a core encryption and decryption method in the WEP and TKIP modes, and it is one of symmetric encryption methodologies. Although RC4 is a symmetric encryption method, the key it uses does not directly encrypt or decrypt data as AES does. The key used by RC4 is simply used to shuffle a 256-byte data. Once the shuffling process is completed, the key is no longer used. A 1-byte key stream is systematically selected from the 256-byte shuffled array to encrypt or decrypt the data.

The most important process of the RC4 algorithm is to continuously shuffle the memory. In our design, the 16-bit packed memory method is applied to design the RC4 pseudo random number generator (PRNG) by using a single-port 128×16-bit memory. In Fig. 4, the "rc4_key" is the lowest 8 bits from the 128-bit keyReg. The address values of the memory, i.e. "addr", are assigned from three address generation sources, and the input data values of the memory, i.e. "data", are allocated from five data sources. The output data, i.e. "q_out", is loaded into the register "q_outReg" before it is transferred to different target registers. Fig. 5, 6, and 7 show the architecture designs of the TKIP Michael, CRC-32, and Cipher functions with surrounding circuits, respectively. In addition, Fig. 8 also depicts the block diagram of the ICV and MIC check circuits.

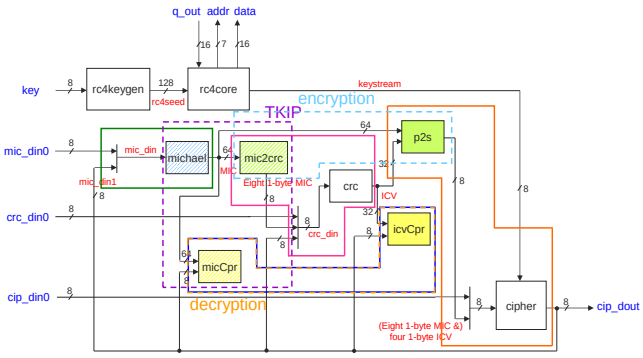

Fig. 3. Internal Structure of TKIP.

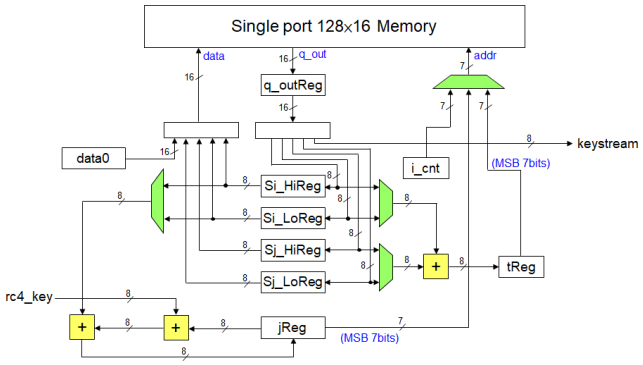

Fig. 4. The architecture of the RC4 PRNG module with a single-port 128×16-bit memory.

The developed RC4 PRNG module has three operational phases, which are described as follows:

A. *The Memory Initialization Phase*

The value of the write address is from "i_cnt", and it will be increased from 0 to 127. The data value is in the "data0" counter, and the initial value is 0x0001, and then each increment is set to 0x0202. Finally, its value is 0xFEFF. In this phase, the operations are completed in 128 cycles.

B. *The Memory Shuffling Phase*

(1) The data is read from the address i (i.e., i_cnt), and it is stored in Si_HiReg and Si_LoReg. The address j/2 is calculated from the value of Si_HiReg.

(2) Read the data from the address j/2, and store it in Sj_HiReg and Sj_LoReg.

(3) Write Si_HiReg to the appropriate location of the address j/2.

(4) Calculate the address j/2 based on the value of Si_LoReg. The data is read from the address j/2, and stored in Sj_HiReg and Sj_LoReg. If the address j/2 equals the address i, the high-order portion of the read data is in Si_HiReg.
(5) Write the value in Si_LoReg to the appropriate location of the address j/2. If the location written to the address j/2 exactly equals the high-order part of the address i, the system updates the value stored in Si_HiReg.
(6) Write the data stored in Si_HiReg back to the address i.
(7) Repeat Step (1) to Step (6), and the value of i_cnt is increased from 0 to 127.

C. The Key Stream Generation Phase

The processing steps are listed as follows:
(1L) Read the data from the address I, and store it in Si_HiReg and Si_LoReg. The address j/2 is calculated from the value of Si_LoReg.
(2L) Read the data from the address j/2, and store it in Sj_HiReg and Sj_LoReg.
(3L) After writing Si_LoReg to the position of the address j/2, the value of Si_LoReg is added to the value of Sj_HiReg (or Sj_LoReg) to calculate the address t/2, where the key stream is located.
(4L) Write Sj_HiReg or Sj_LoReg back to the lower byte part of the address i.
(5L) Read the data from the address t/2, and select the appropriate partial output, which is the key stream.
(6L) Determine if the value of "din_valid" is 1. If it is 1, continue to generate the key stream, go to Step (1H), and the value of i_cnt is increased by one; if it is 0, stop to generate the key stream until the value of "din_valid" is 1, and then continue to generate the key stream.
(1H) Read the data from the address I, and store it in Si_HiReg and Si_LoReg. The address j/2 is calculated from the value of Si_HiReg.
(2H) Read the data from the address j/2, and store it in Sj_HiReg and Sj_LoReg.
(3H) After writing Si_HiReg to the position of the address j/2, the value of Si_HiReg is added to the value of Sj_HiReg (or Sj_LoReg) to calculate the address t/2, where the key stream is located.
(4H) Write Sj_HiReg or Sj_LoReg back to the higher byte part of the address i.
(5H) Read the data from the address t/2, and select the appropriate partial output, which is the key stream.
(6H) Determine if the value of "din_valid" is 1. If it is 1, continue to generate the key stream, go to Step (1L); if it is 0, stop to generate the key stream until the value of "din_valid" is 1, and continue to generate the key stream.

In Step (1L) to Step (6L), the procedures of low-byte portion processing are revealed, and the procedures of the high-byte portion processing are indicated in Step (1H) to Step (6H). In our design, the AES hardware used in CCMP is based on the 32-bit architecture, and the applied S-Box part is implemented by referring the low-cost S-Box architecture in [16]. The AES function defined in CCMP uses a 128-bit encryption key and processes 128-bit data. In the proposed design, the designed AES hardware is based on 32-bit architecture, so the key, data input, and data output need to be processed in 32-bit units. Fig. 9 depicts the internal block diagram of the ccm module. In Fig. 9, the "CCMcore" module is the main control circuit, which controls the AES modules to encrypt and decrypt data and calculate the MIC value. Two AES hardware is implemented in our design, and their module names are "aes4cbc" and "aes4ctr". The "aes4cbc" module is designed for the CBC-MAC mode, and calculates the MIC value of the plaintext. The "aes4ctr" module performs the CTR mode to encrypt or decrypt the data. Besides, the entire "ccm" module is active only when the "crypt_mode" signal is set to the value, i.e. 2.

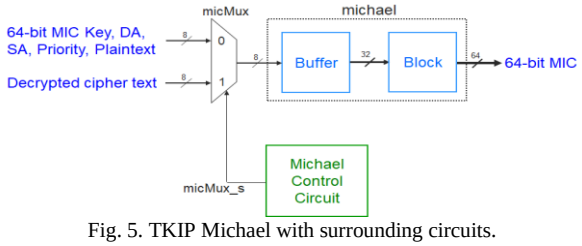

Fig. 5. TKIP Michael with surrounding circuits.

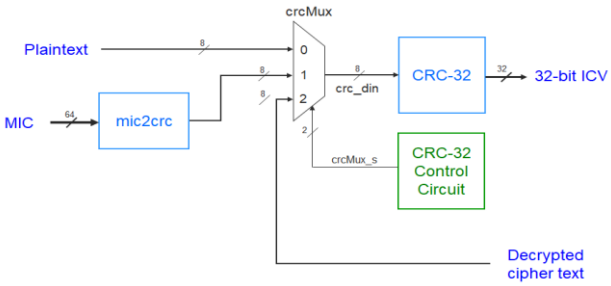

Fig. 6. CRC-32 with surrounding circuits.

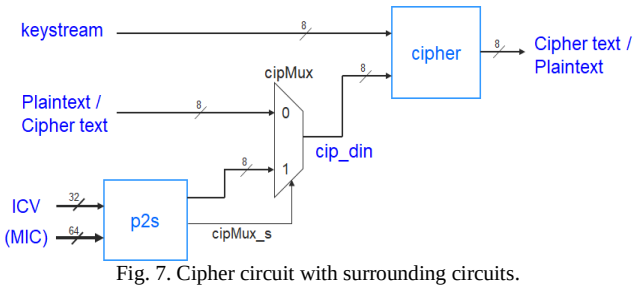

Fig. 7. Cipher circuit with surrounding circuits.

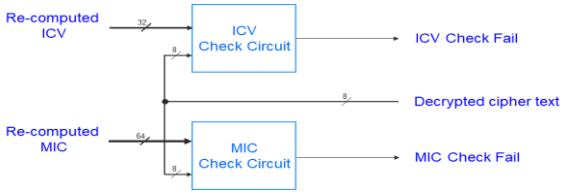

Fig. 8. ICV and MIC check circuits.

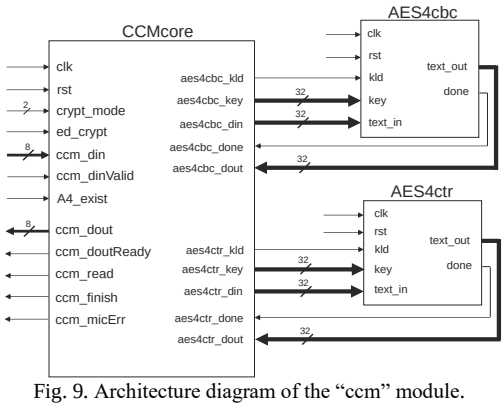

Fig. 9. Architecture diagram of the "ccm" module.

IV. Experimental Results and Discussions

In experiments, firstly the functional model of IEEE 802.11i is built with the C/C++ codes, and then the hardware model is constructed by the synthesizable Verilog HDL. Next, the cell-based design flow is used for the following VLSI implementation. The "ModelSim" HDL simulator is used for functional simulation and verification, and then the hardware-based Verilog HDL model is synthesized by TSMC 0.18um process through Synopsys Design Compiler. Besides, the Artisan Standard Library SRAM Generator [17] is applied to generate the single-port 128×16-bit memory. To compare between two different RC4 PRNG hardware architectures, but do not include the memory cost, only the core control logic is considered, where the "rc4core" module is illustrated in Fig. 3. Table V lists the comparison of the "rc4core" module by using different memories after logic syntheses.

TABLE V: Comparison of the "rc4core" Module by Different Memories

Applied memory types	Gate counts excluding memory	Maximum operational frequency	Power consumption
Single port 256×8-bit	2,315	117.65MHz	1.1661mW
Single port 128×16-bit	2,969	85.76MHz	1.3716mW

By referring to the specification in [17], when the "rc4core" module accesses the memory, it takes 2 cycles to read the data, and only 1 cycle is required to write the data. In Table VI, the initialization phase only performs the write operations, and the number of required cycles is not changed. The shuffling phase includes the read and write operations, and then the number of required cycles is increased. Our design goal is to improve the data throughput of the IEEE 802.11i encryption and decryption processing unit by reducing the latency delay. According to the analysis in Table VI, the proposed design uses a single-port 128×16-bit memory to design the "rc4core" module.

TABLE VI: The Required Memory Access Cycles for the "rc4core" Module by Using Different Memories

Memory types	Cycles for initialization	Cycles for shuffling process
Single port 256x8-bit	256	256 x 6 = 1,536
Dual port 256x8-bit	128	256 x 5 = 1,280
Single port 128x16-bit	128	128 x 9 = 1,152

By VLSI implementation, the cipher computational functions of the low-latency IEEE 802.11i cryptography architecture need 44,300 gate counts, and the maximum operational frequency is 51MHz. At 50MHz operational frequency, the power consumption of the processing unit is 12.61mW. Table VII describes the VLSI implementation comparison between two different IEEE 802.11i hardware architectures, where the gate counts excludes the cost of the single-port 128x16-bit memory module. Table VIII also lists the gate counts of each module for the proposed VLSI design.

TABLE VII: VLSI Implementation Comparison between two IEEE 802.11i Hardware Architectures

	Proposed (core11i)	Design in [10]
Gate counts	44,275	17,832
Memory size	2K bits	2K bits
Maximum frequency	50.99MHz	72MHz
Power consumption	12.61mW@50MHz	14.5mW@60MHz
Supporting ciphering functions	Full functions	Full functions*

* Some functions may be not completely supported in [10].

TABLE VIII: The Gate Counts of Each Module for the Proposed VLSI Design

Module name	Gate counts
dinProcess	901
tkip	16,486
ccm	26,845
outMux	43
Total	44,275

The comparisons between the proposed VLSI design and the previous design in [10] are discussed as follows:

- The function of the key mixing unit in [10] is equivalent to that of the proposed "rc4keygen" module in Fig. 3, which is designed to generate the RC4 key. However, for the TKIP hybrid function phase, the required S-box seems not be implemented in [10], and it causes that the calculated RC4 key does not meet the expected value. In our design, the hardware includes the S-box, which requires about 2,000 gate counts.
- After calculating the MIC value, TKIP will attach the MIC to the plaintext, and will send it to the WEP encryption for processes. The MIC value calculated by TKIP must be sent to CRC-32 to calculate the ICV value. However, in [8], such the data path seems to be lost, which may cause the incorrect ICV value. In the proposed hardware, the "mic2crc" module handles the data path, and the required number of gate counts for this module is about 790.
- In [10], only one AES hardware is used, but the proposed design needs two AES hardware for pipeline processing. The hardware cost of the applied AES hardware requires 6,160 gate counts. Since the proposed hardware uses a synchronous design, all of the module outputs are register-based outputs, which result in the additional hardware cost of output registers.
- Compared with the design in [10], the proposed hardware needs a lower maximum operating frequency. In the proposed design, the critical path is existed during the operation of the "rc4keygen" module, and the processing time of the critical path is the sum of delay time of the 16-bit XOR operation, the lookup table process of S-box, and the 16-bit addition. In the proposed architecture, the critical path is processed in one cycle for low latency design.

The data throughputs of the proposed hardware design are discussed with the WEP, TKIP and CCMP modes as follows:

In the WEP operational mode, the "dinProcess" module has one latency cycle, and the "rc4keygen" module has nine latency cycles. The "rc4core" module has three operational phases. The delay period in the memory initialization phase is 128 cycles, and the delay period in the memory shuffling

phase is 1,152 cycles. In the keystream generation phase for generating 1-byte key stream, the delay period for operations needs 9 cycles. For encrypting the N-byte plaintext, the delay period of the phase is 9×*N*. If both the plaintext and the key stream are ready, the cipher module can complete the encryption in only one cycle, and the encryption is performed during generating the next key stream by the "rc4core" module. Therefore, the required duty cycles in the WEP mode are (1 + 9 + 128 + 1,152) + (9×*N*), where *N* is the total number of bytes in plaintext. Table IX shows the required duty cycles of the WEP mode at different plaintext lengths, where the data throughput is calculated when the clock frequency is set to 50MHz.

TABLE IX: THROUGHPUT OF WEP FOR THE PROPOSED DESIGN

Data length (N)	# Clock cycles	Throughput (@50MHz)
100-byte	2,190	18.26Mbps
500-byte	5,790	34.54Mbps
1,000-byte	10,290	38.87Mbps
2,000-byte	19,290	41.47Mbps

When the processing unit operates in the TKIP mode, the delay period of the "dinProcess" module is 23 cycles, and the delay period of the "rc4keygen" module is 84 cycles. The delay period of the "rc4core" module is the same as that in the WEP mode. Therefore, the required duty cycles in the TKIP mode is (23 + 84 + 128 + 1,152) + (9×*N*), where *N* is the total number of bytes in plaintext. Table X lists the required duty cycles of the TKIP mode at different plaintext lengths. In Table X, the data throughput is also calculated at the 50MHz operational frequency.

TABLE X: THROUGHPUT OF TKIP FOR THE PROPOSED DESIGN

Data length (N)	# Clock cycles	Throughput (@50MHz)
100-byte	2,287	17.49Mbps
500-byte	5,887	33.97Mbps
1,000-byte	10,387	38.51Mbps
2,000-byte	19,387	41.26Mbps

TABLE XI: THROUGHPUT OF CCMP FOR THE PROPOSED DESIGN

Data length (N)	# Clock cycles	Throughput (@50MHz)
100-byte	524	82.44Mbps
500-byte	1,699	119.60Mbps
1,000-byte	3,156	127.76Mbps
2,000-byte	6,070	132.32Mbps

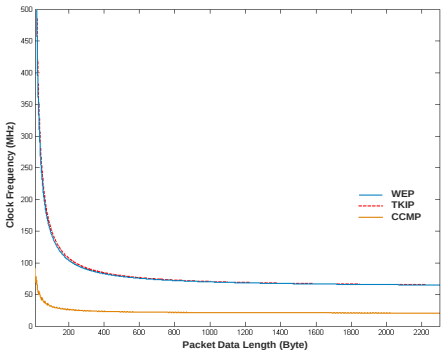

Fig. 10. Required clock frequency to achieve the transmission rate of 54Mbps for the proposed design.

Table XI reveals the required duty cycles of the CCMP mode at different plaintext lengths. The data throughput is also estimated when the operational frequency is set to 50MHz. Fig. 10 illustrates the required clock frequency to achieve the transmission rate of 54Mbps for the proposed architecture design at the WEP, TKIP, and CCMP modes, respectively.

V. CONCLUSION

IEEE 802.11i has three security mechanisms, which include WEP, TKIP and CCMP. In this paper, the three major security functions are implemented with hardware design due to their enormous amount of computational costs. The 16-bit packed memory algorithm in [10], the non-pipelined 32-bit AES architecture, and the TKIP mixing function are applied to achieve the goal of low latency. By VLSI implementation with TSMC 0.18um technology, the cipher computational functions of the low-latency IEEE 802.11i cryptography architecture need 44,300 gate counts, and the maximum operational frequency is 51MHz. At 50MHz operational frequency, the power consumption of the processing unit is 12.61mW. In future works, the low-latency IEEE 802.11i security processing circuit will be efficiently developed to support higher throughput for the next-generation IEEE 802.11 communications, e.g. IEEE 802.11n/ac.

CONFLICT OF INTEREST

To the best of our knowledge, the named authors have no conflict of interest, financial or otherwise.

AUTHOR CONTRIBUTIONS

Jun-Dian Li conducted the architecture research and VLSI design; Chih-Peng Fan wrote the paper; all authors had approved the final version.

ACKNOWLEDGMENT

This work was financially supported in part by the Ministry of Science and Technology (MOST) under Grant No. MOST 108-2634-F-005-002 and by the "Innovation and Development Center of Sustainable Agriculture" from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan (R.O.C.). The authors would like to thank Taiwan Semiconductor Research Institute (Former National Chip Implementation Center) in Taiwan for EDA supports.

REFERENCES

[1] W. Stallings, *Cryptography and Network Security, Principles and Practices*, 3rd ed. Prentice Hall, 2003.

[2] Information technology — Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements— Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ANSI/IEEE Std. 802.11, 1999 Edition (R2003).

[3] IEEE Standard for Information technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications

Amendment 6: Medium Access Control (MAC) Security Enhancements, IEEE Std. 802.11i, 2004.
[4] IETF RFC 3610, Counter with CBC-MAC (CCM), 2003.
[5] P. Kitsos, G. Kostopoulos, N. Sklavos, and O. Koufopavlou, "Hardware implementation of the RC4 stream cipher," in *Proc. 46th IEEE Midwest Symposium on Circuits & Systems*, Cairo, Egypt, 2003.
[6] J. D. Lee and C. P. Fan, "Efficient low-latency RC4 architecture C. sivakumar and A. velmurugan, "High speed VLSI design CCMP AES cipher for WLAN (IEEE 802.11i)," in *Proc. 2007 International Conf. on Signal Processing, Communications and Networking*, Chennai, India, Feb. 2007.
[7] C. Sivakumar and A. Velmurugan, "High speed VLSI design CCMP AES Cipher for WLAN (IEEE 802.11i)," in *Proc. 2007 International Conf. on Signal Processing, Communications and Networking*, Chennai, India, Feb. 2007.
[8] I. Algredo-Badillo, C. Feregrino-Uribe, R. Cumplido, and M. Morales-Sandoval, "FPGA implementation and performance evaluation of AES-CCM cores for wireless networks," in *Proc. International Conference on Reconfigurable Computing and FPGAs*, 2008, pp. 421-426.
[9] S. A. Hoseini, B. Khodabandeloo, M. J. Mamaghani, P. Teymoorim, and N. Yazdani, "High throughput low power CCMP architecture for very high speed wireless LANs," in *Proc. 15th CSI International Symposium on Computer Architecture and Digital Systems*, Tehran, Iran, 2010.
[10] Y. Mitsuyama, M. Kimura, T. Onoye, and I. Shirakawa, "Architecture of IEEE802.11i cipher algorithms for embedded systems," *IEICE Trans. on Fundamentals*, vol. E88-A, no. 4, pp. 899-905, April 2005.
[11] O. Song and J. Kim, "Hardware-software co-design of secure WLAN system for high throughput, *2nd IFIP Wireless Days (WD)*, Paris, France, 2009.
[12] Y. Li, J. Han, S. Wang, J. Liu, and X. Zeng, "A NoC-based multi-core architecture for IEEE 802.lli CCMP," in *Proc. 9th IEEE International Conf. on ASIC*, Xiamen, China, 2011.
[13] A. Kumar and P. Paul, "Security analysis and implementation of a simple method for prevention and detection against evil twin attack in IEEE 802.11 wireless LAN," in *Proc. International Conf. on Computational Techniques in Information and Communication Technologies (ICCTICT)*, New Delhi, India, 2016.
[14] T. Hayajneh, S. Ullah, B. J. Mohd, and K. S. Balagani, "An enhanced WLAN security system With FPGA implementation for multimedia applications," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2536-2545, 2017.
[15] National *Institute of Standards and Technology (NIST): Advanced* Encryption *Standard (AES). Federal Information Processing Standards (FIPS) Publication 197*, November 2001.
[16] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-Box optimization," in *Proc. the 7th International Conf. on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, Lecture Notes In Computer Science*, 2001, vol. 2248, pp. 239-254.
[17] Artisan *Standard Library SRAM Generator User Manual, ug_2004q1v0 and ug_2003q2v0,* Artisan Components, Inc., 2003.

Jun-Dian Lee was born in Taoyuan, Taiwan in 1982. He received his B.S. degree in computer and communication engineering from National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan, in 2005, and the M.S. degree in electrical engineering from National Chung Hsing University, Taichung, Taiwan, in 2007. His research interests include internet encryption and VLSI designs.

Chih-Peng Fan was born in Miaoli, Taiwan in 1969. He received the B.S., M.S., and Ph.D degrees in electrical engineering from the National Cheng Kung University, Taiwan in 1991, 1993 and 1998, respectively. From October 1998 to January 2003, he was a design engineer with N100, Computer and Communications Research Laboratories (CCL), Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan. In 2003, he joined the faculty of the Department of Electrical Engineering, National Chung Hsing University, Taichung city, Taiwan, where he is currently a full professor. He has more than 100 publications, which include technical journals, book chapters, conference papers, and technical reports. His teaching and research interests include digital image processing and pattern recognition, digital video processing, baseband transceiver design, and VLSI design/FPGA prototype of DSP systems.