Hardware Design and Analysis of Statistical Cipher Feedback Mode Using Serial Transfer

Liang Zhang, Howard M. Heys
Electrical and Computer Engineering
Memorial University of Newfoundland
St. John's, Newfoundland
e-mail: {lzhang, howard}@engr.mun.ca

Abstract

In this paper, the hardware design of a recently proposed mode of operation for a block cipher, referred to as Statistical Cipher Feedback (SCFB), is investigated. Specifically, we examine a structure which employs serial transfer from the Plaintext Queue to the Ciphertext Queue. SCFB mode is the hybrid of output feedback (OFB) mode and cipher feedback (CFB) mode that allows a block cipher to be configured as a self-synchronizing stream cipher. Consequently, SCFB mode feeds back ciphertext to the input of the block cipher similar to the conventional CFB mode, except that the feedback only occurs when the n bit sync-pattern is recognized thus making SCFB more efficient in its implementation than conventional CFB mode. An iterative based implementation of the Advanced Encryption Strandard (AES) is investigated and the relationship among three different clock domains associated with a serial transfer implementation is studied based on the synthesis results for various components of the system, as is the system efficiency. From simulations, an appropriate buffer size which minimizes queue overflow is selected for the design. The design is synthesized as an ASIC targeted to 0.18 *CMOS standard cell technology. From the synthesis result, the throughput of the SCFB system is determined to be 100 Mbps. The total area of the SCFB system is approximately* 41600 *gates, of which 16900 is for AES.*

1 Introduction and Background

A stream cipher is an important method of encryption in which the plaintext is encrypted bit-by-bit or symbol-by-symbol to produce the corresponding ciphertext. A stream cipher can be constructed by generating a pseudo-random keystream using a block cipher output to exclusive-or (XOR) with the plaintext to produce ciphertext at the transmitter. At the receiver, the plaintext is recovered by generating the identical keystream which is then XORed with the ciphertext. Stream ciphers can be used for high-speed networks at the physical layer in a communication system.

In a typical stream cipher configuration, a single bit of ciphertext error only results in a single bit of recovered plaintext error. However, a stream cipher will cause complete nonsense data for the rest of the recovered plaintext if bit slips or insertions happen in the communication channel. Hence, it is important to keep the keystream of both the transmitter and receiver synchronized. Output feedback (OFB) mode and cipher feedback (CFB) mode are two conventional modes of operation of block ciphers that allow their use as stream ciphers. In this work, we are concerned with statistical cipher feedback (SCFB) mode, proposed in [1] and investigated in [2], which is a hybrid of CFB and OFB. This SCFB mode configures block ciphers, such as the Advanced Encryption Standard (AES) [3], as stream ciphers capable of self-synchronization. SCFB mode has been proposed to provide physical layer security for a SONET/SDH environment and is suitable for many other applications as well. In this paper, the hardware structure for SCFB mode using a serial transfer implementation is thoroughly investigated.

The sketch of SCFB mode is shown in Figure 1. The Sync Pattern Recognition block is used to scan ciphertext to find an n bit sync pattern and collect an initialization vector(IV) after the sync pattern is found. The input of the block cipher can be either the output of the block cipher (OFB mode) or the IV from the ciphertext (CFB mode) depending on whether the n bit sync pattern is recognized.

2 Hardware Design of SCFB Using Serial Transfer

2.1 AES with Key-Scheduling

The AES algorithm [3] is a symmetric block cipher developed for the National Institute of Standards and Technology (NIST) to replace DES. In this work, AES with a key length of 128 bits is adopted for the block cipher to generate the key stream block. The AES algorithm repeats a series of operations for 10 rounds. Figure 2 shows the steps of the AES algorithm. In a hardware implementation, the round function is performed iteratively 10 times and the data path is shared for different rounds of the algorithm. The S-box of AES is based on the composite field based on $GF(2^4)$ implementation [5].

2.2 Hardware Implementation Details

The hardware implementation of SCFB mode using serial transfer from the plaintext queue to the ciphertext queue is illustrated in Figure 3. The plaintext queue is needed to store the incoming bits and transfer them out to XOR with the keystream bit by bit. The ciphertext queue is needed to store the ciphertext bits and send them out of the SCFB system bit by bit.

0840-7789/07/$25.00 ©2007 IEEE

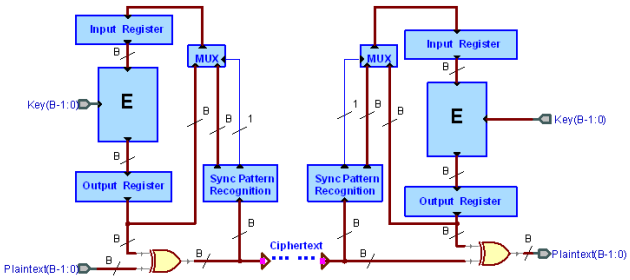

Figure 1: SCFB system

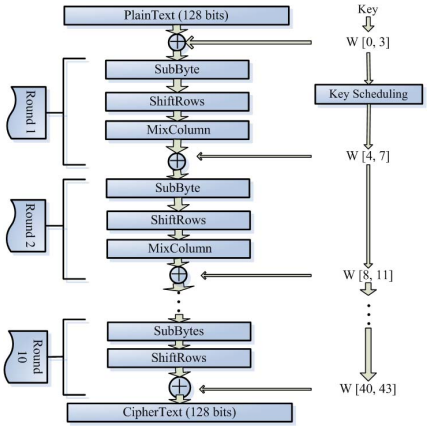

Figure 2: AES using key-scheduling

The queuing system is necessary to accomodate periods during which the keystream is not available due to resynchronization. A previous implementation of SCFB mode transfered data between queues in blocks of 128 bits [4]. However, the resulting design required a large amount of hardware. In the serial design, there are three clocks, $clk1$, $clk2$ and $clk3$, to control the running speeds of the data transfer and the block cipher: $clk1$ is used to clock the transfer of data out of the plaintext queue and into the ciphertext queue, $clk2$ is used to clock data into and out of the SCFB system, and $clk3$ is used to clock a round of the block cipher. The plaintext queue and the ciphertext queue are initialized to be empty and full, respectively. While the plaintext data is being collected bit by bit in the plaintext queue, a keystream block of 128 bits is generated by the block cipher. If a block of keystream is ready and the sync pattern is not recognized, the $B = 128$ bits of the keystream will be loaded into Block Register. Also the same keystream will be loaded into the block cipher as the new input data. Then, Shift Register (SR) will load in this block of keystream if it is empty and then begin to shift bits out one by one. At the same time, the plaintext queue will shift out the data bit by bit to XOR with the keystream coming from Shift Register. When the sync pattern is recognized, the system will continue working in the OFB mode for at least 128 $clk1$ cycles to collect the complete IV into IV_Shift_Register. When the 128 bits of IV are ready in IV_Shift_Register, Shift Register, Plaintext Queue and Ciphertext Queue will be held. That is to say, Shift Register and the plaintext queue will not shift out bits any more, and the ciphertext queue will not have any incoming data until the new IV is used to create a new keystream block. However, the plaintext queue will continue to accept incoming data and the ciphertext queue will continue to transmit outgoing data. The new IV block is sent into the block cipher as the new "data_in", and the next block of key stream will be generated by the block cipher. After this new keystream is ready, the controller will provide it to Shift Register and simultaneously unhold Shift Register, Plaintext Queue, Ciphertext Queue and IV_Shift_Register.

3 Synthesis Results, Analysis and Comments on the Design

As we mentioned before, there are three clock domains in this system. Among these clocks, $clk1$ is the fastest clock and it can be the base system clock in the implementation. The clocks $clk2$ and $clk3$ can be derived from $clk1$. As shown in Figure 3, the rate of incoming plaintext data to Plaintext Queue, R, is directly equal to the frequency of $clk2$, since the data collection of Plaintext Queue is based on $clk2$. The system efficiency can be controlled by adjustment of these three clock frequencies. Plaintext Queue collects incoming data at the rate R and outputs the data at the rate of $clk1$. Ciphertext Queue has the reverse situation. The interfaces (Block Register, Shift Register, etc.) of the block cipher also use $clk1$ to keep the same pace with the two queues. The block cipher, which is clocked at a per-round rate of $clk3$, has to run as fast as possible in order to reduce the idle time that stalls the queue bit transfer due to generating the keystream when resynchronization occurs.

In order to make the hardware size as small as possible, design simulations for buffer sizes ranging from 48 to 256 bits and different clock frequencies for the block cipher are undertaken. From the simulations, for clock frequencies of $clk1 = 1/10$ ns, $clk2 = 1/5$ ns and $clk3 = 1/25$ ns, an appropriate buffer size of 64 bits which was found to have no queue overflow is selected. The distribution of number of bits in the plaintext queue is shown in Figure 4 for varying sync-pattern sizes. The simulation results are based on 4000 cycles of $clk3$. In general, with high probability there will be fewer than 6 bits in the queue. At times, with

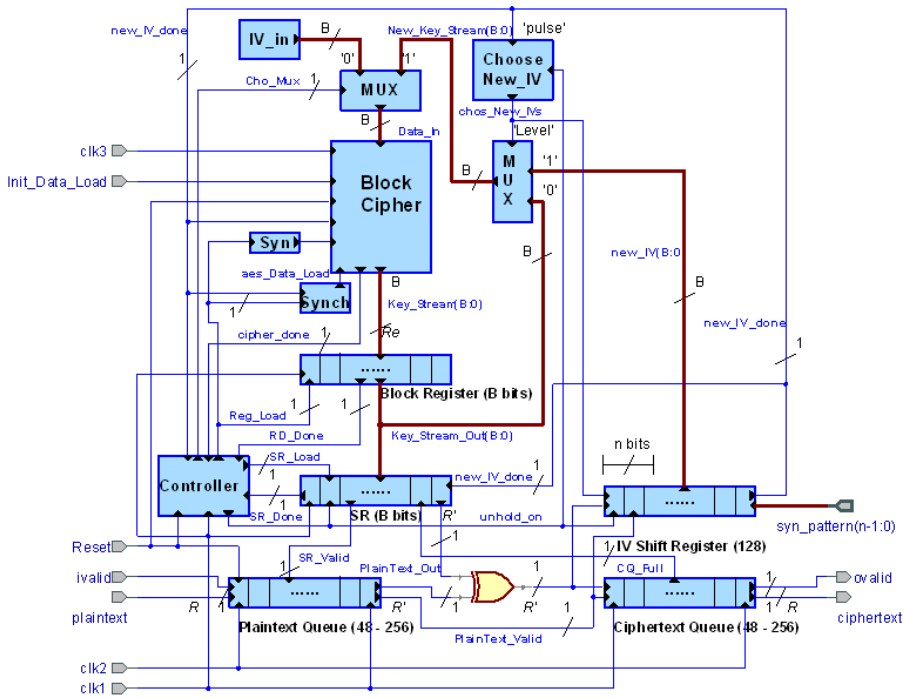

Figure 3: Hardware implementation of SCFB using serial transfer

non-zero probability, as many as 45 bits were found in the queue. This results from the resynchronization of the SCFB system. The number of stored bits continuously increases without any outgoing bits in the plaintext queue when the new IV is used to generate a keystream block. The resynchronization happens more frequently for the smaller size of sync-pattern. So the queue would have more chances to be filled with incoming bits without any outgoing bits during the resynchronization for the smaller size of sync-pattern. The same queue would have less time for the normal operation where the resynchronization does not happen. This is why the peak for the smaller size sync-pattern is lower than that for the larger size sync-pattern. From simulations, we also get the average number of bits = 7.99, 11.88 and 14.13 in queue for sync-pattern size = 8, 6 and 4, respectively.

An ASIC synthesis with 0.18 micron CMOS standard cell technology using Synopsys tools supported by Canadian Microelectronics Corporations (CMC) was completed. We use the number of equivalent 2-input NAND gates for the total area as a metric of circuit size. The synthesis results of the block cipher, Plaintext Queue and Ciphertext Queue are shown in Table I. The speed of the block cipher is set to $128/12 \times 25$ ns $\approx$ 426.67 Mbps using $clk3$ to be 1/5 of $clk2$. The throughput of the SCFB system is 1/10 ns = 100 Mbps. Hence, the efficiency is $100/426.67 \approx 23.4\%$. Thus, the throughput of Plaintext Queue becomes the bottleneck of the system. To improve the efficiency and speed of this system the structure must be changed from serial to parallel transfer, when bits can be transfered between queues in blocks that are much less than 128 bits in size. This modification is left for future work to be done.

TABLE I
SYNTHESIS RESULT USING 0.18 MICRON CMOS

	Total Area (# gates)
Plaintext Queue	1232
Ciphertext Queue	2291
PQ_CQ_Integrated	3525
AES	16919
SCFB System	41600

4 Conclusion

In this paper, the hardware implementation of Statistical Cipher Feedback (SCFB) using serial transfer from the plaintext queue to the ciphertext queue was investigated. An iterative implementation of the Advanced Encryption Strandard (AES) was adopted as the block cipher in this SCFB system. Although the throughput of the block cipher is high, the throughput of the plaintext queue can only reach 100 Mbps, which becomes the bottleneck of the system efficiency and throughput. By doing the functional simulations for different buffer sizes, we have selected an appropriate buffer size of 64 bits which minimizes queue overflow. We have also investigated how the various sync-pattern sizes affect the probability distribution of the current

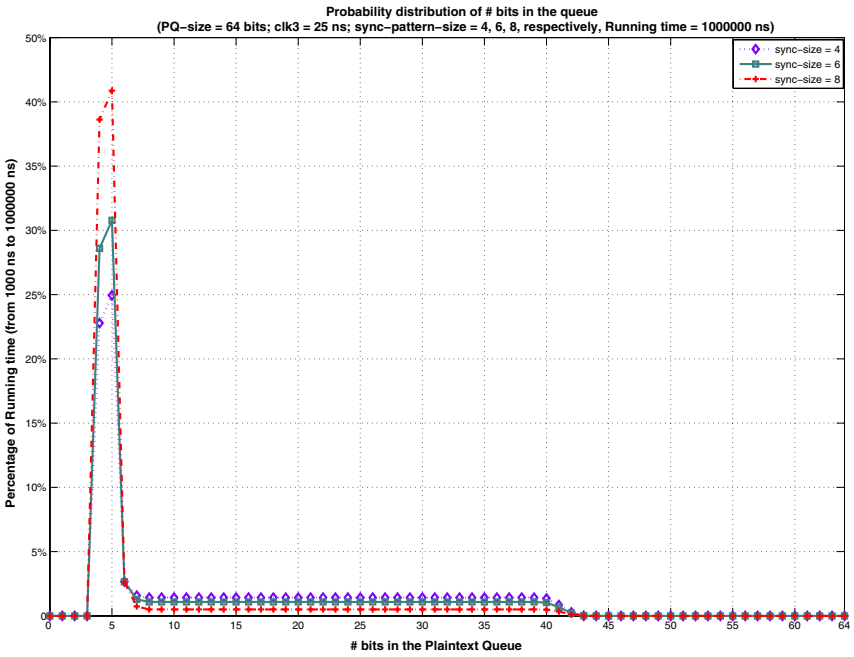

Figure 4: Probability Distribution of # bits in the plaintext queue

number of bits and average number of bits in the plaintext queue.

References

[1] O. Jung and C. Ruland, "Encryption with Statistical Self-Synchronization in Synchronious Broadband Networks",*Cryptographic Hardware and Embedded Systems - CHES'99,* Lecture Notes in Computer Science 1717, Springer-Verlag, pp. 340-352, 1999.

[2] Howard M. Heys, "Analysis of the Statistical Cipher Feedback Mode of Block Ciphers",*IEEE Transactions on Computers,* vol. 52, issue 1, pp. 77-92, Jan. 2003.

[3] National Institute of Standards and Technology, AES web site: csrc.nist.gov/encrytion/aes.

[4] Fang Yang, "Analysis and Implementation of Statistical Cipher Feedback Mode and Optimized Cipher Feedback Mode," Master Thesis, Memorial University of Newfoundland, Jan. 2004.

[5] J. Wolkerstorfer, E. Oswald, M. Lamberger, "An ASIC implementation of the AES SBoxes," *The Cryptographer's Track at the RSA Conference 2002,* pp. 67, Feb. 2002.