Hardware Implementation of a High Speed Self-Synchronizing Cipher Mode

Yuanchi Tian and Howard M. Heys

Abstract— **Pipelined statistical cipher feedback (PSCFB) mode is a new mode of operation for block cipher encryption. It is an improved version of conventional SCFB mode with higher throughput. SCFB mode has the mechanism of self synchronization to recover from bit slips during transmission in a communication channel. The mechanism of SCFB mode resembles output feedback (OFB) mode and cipher feedback (CFB) mode. However it has self synchronization that OFB mode does not and has higher efficiency than CFB mode. To improve the throughput, PSCFB is a modified version of SCFB that allows for the pipelining of the underlying block cipher while still preserving the efficiency and self-synchronizing capabilities. In this paper, the Advanced Encryption Standard (AES) with a pipeline architecture is used as the block cipher in PSCFB. The PSCFB system is designed, simulated and synthesized targeted to an Altera Cyclone IV FPGA. The structures and processes of both the encryption and decryption are presented. The system performance is analyzed based on the simulation and synthesis results.**

Index Terms— **Block Cipher, Advanced Encryption Standard (AES), Digital Hardware, PSCFB, Mode of Operation**

I. INTRODUCTION

Statistical cipher feedback (SCFB) mode of operation [1] allows a block cipher to work as a self-synchronizing stream cipher. Pipelined statistical cipher feedback (PSCF-B) [2] mode uses a pipeline architecture in the block cipher based on counter (CTR) mode. In this paper, AES [3] in PSCFB mode with a 128 bit key is used as the block cipher to generate keystream. Both the encryption and decryption of PSCFB is implemented targeted to Altera Cyclone IV FPGA [4]. Considering the trade-off between system speed and hardware resource complexity, the hardware is designed to minimize complexity, while still operating at a high frequency. The design of PSCFB is illustrated in the following sections.

II. BACKGROUND

In this section, we discuss the necessary background selected to PSCFB mode.

A. Classical Cipher Modes

In cryptography, a mode of operation uses a block cipher to encrypt data which is longer than one block. Block ciphers operate on a single block of data, which contains a fixed number of bits. In this paper, we will use B as the size of a single block in bits. For AES, $B = 128$ bits. On the contrary, stream ciphers operate on a single bit. However, some modes of operation can make the block cipher work as a stream cipher. Output Feedback (OFB) mode, Cipher Feedback (CFB) mode and Counter (CTR) mode [5] are typical modes of operation that operate like stream ciphers.

OFB mode feeds the B bit output back to the input. This mechanism continuously generates the keystream for XORing with plaintext. However, this mode requires precise synchronization between transmitter and receiver, that is, encryption and decryption. Since OFB mode cannot self synchronize, an extra module is needed for synchronization. Under this circumstance, some additional resources are required.

CFB mode feeds the ciphertext, which is the result from XORing plaintext and output of block cipher, back to the input. CFB is a mode of self-synchronization which means recovering from bit slips (i.e., when one or more bits eliminated from the received ciphertext stream). In CFB mode, in order to recover from any number of lost bits, only one bit can be XORed with one bit of plaintext. Hence, from every block of B bits produced by the block cipher CFB is very inefficient.

In CTR mode, a counter with B bits is typically used as the input of a block cipher. The counter is loaded with an Initialization Vector (IV) first and then increases by 1 for each block encryption, and, hence, there is not any type of feedback in CTR mode. The key stream produced by the block cipher can be XORed in groups of B bits with plaintext in encryption or ciphertext in decryption. As well, in addition to being efficient, pipelining can be used in CTR mode, thus improving the throughput substantially. CTR mode is not self-synchronizing.

B. SCFB Mode

Statistical self-synchronization is proposed in [6] and Statistical Cipher Feedback (SCFB) mode is analyzed in [1] as a way to solve the efficiency and synchronization problem. This mode works as OFB or CFB under different conditions. When the system is scanning for n-bit sync pattern in ciphertext, it works as OFB mode. In this paper, the size of sync pattern is $n = 8$ bits. When the sync pattern is found, the B bit ciphertext following the sync pattern will be loaded as a new IV as in CFB mode. While collecting the B bit IV, any newly recognized sync pattern will be ignored. After the new IV has been loaded, a new

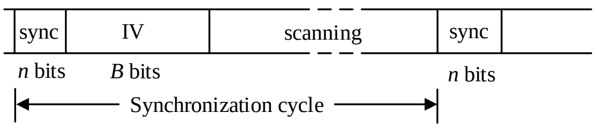

Fig. 1. Synchronization cycle of SCFB [1].

Y. Tian is with Department of Electrical and Computer Engineering, Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada yuanchi.tian@mun.ca

H. M. Heys is with Department of Electrical and Computer Engineering, Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada hheys@mun.ca

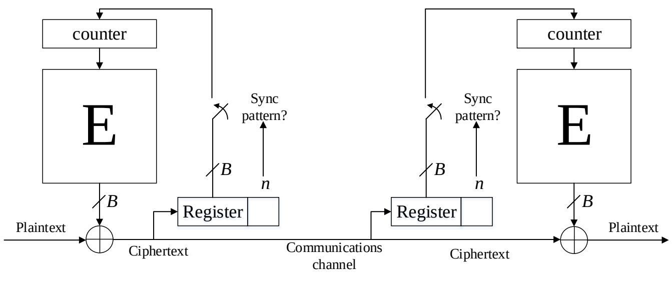

Fig. 2. SCFB mode with counter [2].

round of scanning will start again and the system will return to OFB mode. Fig. 1 shows the whole process of a synchronization cycle.

As in Fig. 2, OFB mode in the system can also be replaced by counter mode. If a sync pattern is found, the next B bits will be loaded to the counter as a new IV block. During the scanning period, the counter increases by one for each block encryption.

C. PSCFB Mode

Using counter mode instead of OFB mode makes SCFB suitable for an improved version, which is called Pipelined Statistical Cipher Feedback (PSCFB) mode shown in Fig. 3. Pipelining leads to high speed, which means high throughput in the network communication. In PSCFB mode, the length of the synchronization cycle is extended due to pipelining. For example, if we assume AES with $L = 10$ pipeline stages, the output can be expected 10 block encryptions after a sync pattern is detected. As a result, the period when the scanning is disabled, referred to as the blackout period, will be extended to 10 blocks of B bits. This is illustrated in Fig. 4.

III. Design

When designing the structure of hardware, speed and area are two major factors needed to be considered. Trade-off between how much data can be transferred in a unit time and how many resources it costs is important. Since the PSCFB system is designed for use in high speed networks, the goal is to reduce the resource usage as much as possible without affecting throughput.

For the following descriptions of the system components, our discussion focuses on the encryption process. For the decryption process, the roles of the plaintext and ciphertext

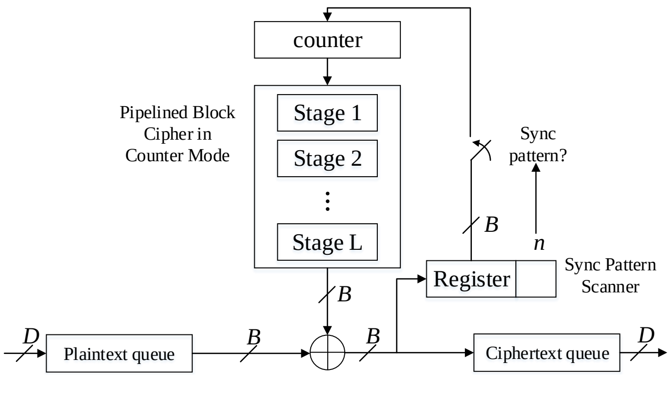

Fig. 3. PSCFB (encryption) [2].

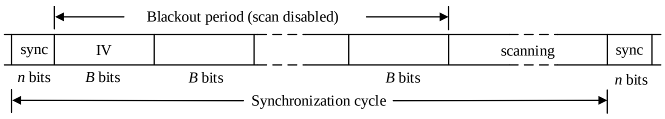

Fig. 4. Synchronization cycle of PSCFB [2].

queues get reversed as will be discussed at the end of this section.

A. Counter

The general counter increments by 1 in each clock cycle. For CTR mode in cryptography, the counter is loaded with an IV and then incremented by 1 for each block encryption. In our system, a linear feedback shift register (LFSR) is used to replace the general counter. Several bits are XORed to generate a feedback bit to be shifted into register. For a 128 bit register, a feedback expression with only 4 bits is used: $r'_{127} = r_{29} \oplus r_{17} \oplus r_2 \oplus r_0$ [7]. In this expression, r_i, $0 \leq i \leq 127$, represents a register bit, with the shift moving from higher number bits to lower number bits and r'_{127} representing the next value of bit to be shifted in. The 128 bit LFSR covers $2^{128} - 1$ states except for the all zero state. Since there is the possibility that all zero IV is applied, an LFSR with all zero state [7] is used here. The feedback expression is modified to be $r'_{127} = r_{29} \oplus r_{17} \oplus r_2 \oplus r_0 \oplus z$ and $z = 1$ when the 127 register bits from r_{127} to r_1 are all zero.

B. Pipelined AES and Modified Blackout Period

For AES with 128 bit key, there are 10 rounds in total [3]. There are four operations in each round. SubBytes uses a S-box byte-by-byte mapping to perform substitution. ShiftRows is a simple shift operation among different rows. MixColumn is a 32 bit linear transformation operated over $GF(2^8)$. AddRoundKey performs bit-by-bit XOR between the block and round key. In this paper, the AES is implemented as a pipelined structure based on the example from [8]. No memory is used in our implementation with only combinational gates and registers being used.

As shown in Fig. 5, 10 pipeline registers are inserted after AddRoundKey in each round. Each stage of the pipeline is processed in less than a clock period. The initial AddRoundKey and four transformations in the round 1 together form the first stage of pipeline. For the rounds 2 to 9, each stage contains SubBytes, ShiftRows, MixColumns and AddRoundKey. For the round 10 in the last pipeline stage, only three transformations are included as shown in Fig. 5. When a new IV is loaded to AES, it takes 10 clock cycles to generate the corresponding output. However, it takes 11 clock cycles to generate the output since there is one clock cycle to load IV on the LFSR counter. Hence, the blackout period is extended to $L = 11$ blocks of B bits.

C. Sync Pattern Scanner

The sync pattern scanner scans the ciphertext for the $n = 8$ bit sync pattern with the format of 10000110. A block diagram of this component is presented in Fig. 6. The bottom 128 bit register receives 128 bit ciphertext and the top register receives the data from the bottom one, thus making up a 256 bit register with 128 bit input width. In

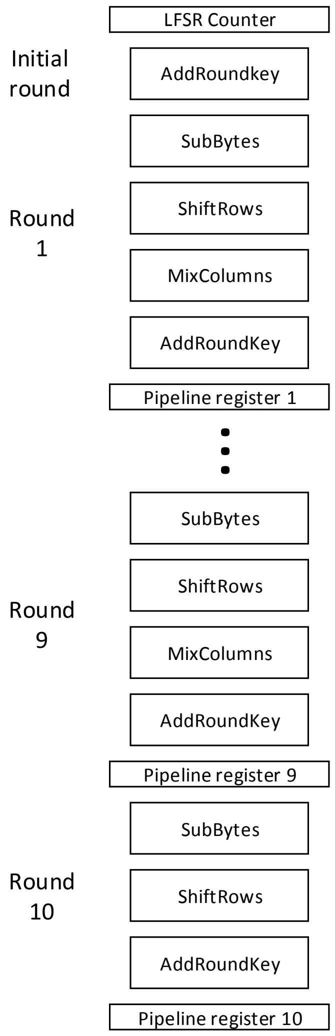

Fig. 5. Pipelined AES in PSCFB.

most cases it takes 2 blocks of ciphertext to collect a new IV after the sync pattern. Hence, it is a necessary to have the double registers.

The scan logic contains the combinational logic shown in Fig. 7 replicated 128 times. When the 8 bit sync pattern is matched to a single scan logic part, the output is 1. For 128 scan logic parts, the scanner needs $128 + n - 1 = 135$ bits for scanning. So it takes full block data from the first register and lower 7 bits from the second register. The scan logic sends out 128 bit result to indicate if the sync pattern is detected. However, it is likely to have more than one sync pattern found in a block, that is, not only one 1 may appear in the output of the scan logic. The matched sequence from the upper bits has higher priority so that the first matched pattern should be accepted. A priority encoder receives all 128 bits but takes the uppermost high level bit and then encodes it. Unlike the general priority encoder, the priority encoder in our implementation is reversed. For example, when only $in[0] = 1$, which means input sequence is $in[127:0] = 0...0001$, the output of the priority encoder is $out[6:0] = 1111111$. If MSB $in[127] = 1$, the output of the priority encoder will be $out[6:0] = 0000000$. This encoding will make it easier for other components in the system. The number of bits d to be transferred at the end of blackout period exactly equals to the output code plus 1, that is, $d = code + 1$. Hence, the code is directly sent to control unit of the queues discussed in sections D and E.

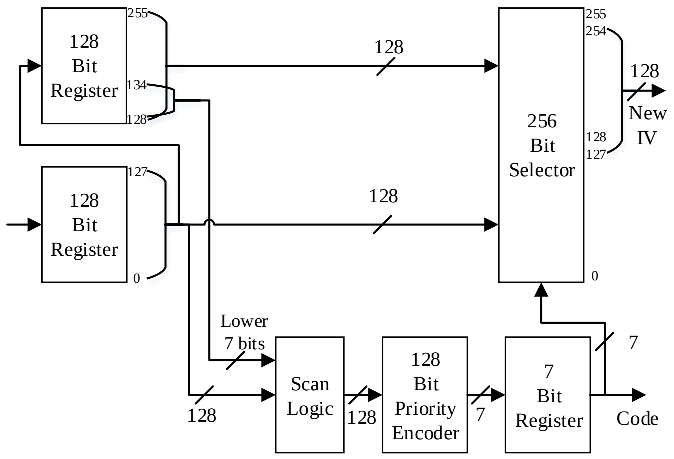

Fig. 6. Sync pattern scanner (data path).

In order to output the correct IV, the code will also be sent to the selector as 7 bit selection signal. The selector is based on barrel shifter architecture. The barrel shifter will be described in the later part. It receives 256 bits data from the registers and shifts the selected 128 bits to fixed positions from 254 to 127. For example, if a sync pattern is found as $register[7:0] = 10000110$, the input of the priority encoder will be $in[127:0] = 0...0001$ and the output will be $out[6:0] = 1111111 = code$. Since the 128 bit new IV is right after the sync pattern, in the next clock cycle, the incoming block at the bottom register is the new IV. The selector will left shift data $register[127:0]$ by 127 positions, which will be presented at the output as $selector\_output[254:127]$.

D. Plaintext Queue in Encryption

Based on the proposed structure in [2] and illustrated in Fig. 3, a plaintext queue and ciphertext queue are two significant parts which can affect the speed of the PSCFB system. They are required to ensure that while the data processing can happen at irregular rates, the input and output to the system maintains a constant rate. The output of the block cipher requires the plaintext queue to transfer B bits data to the ciphertext queue except at the end of the blackout period where it is possible to transfer d bits with $(1 \leq d < B)$. For the plaintext queue, the rate D means that there are D bits data to be enqueued every clock cycle. To avoid overflow in plaintext queue, D must be less than B. Queue size M should also be large in order to guarantee no overflow, and the minimum value is $M \geq B + 3D - 2$

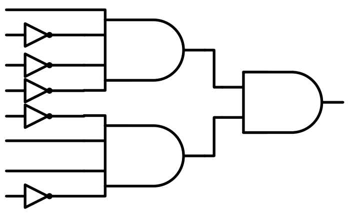

Fig. 7. A single scan logic for 8 bit sync pattern 10000110.

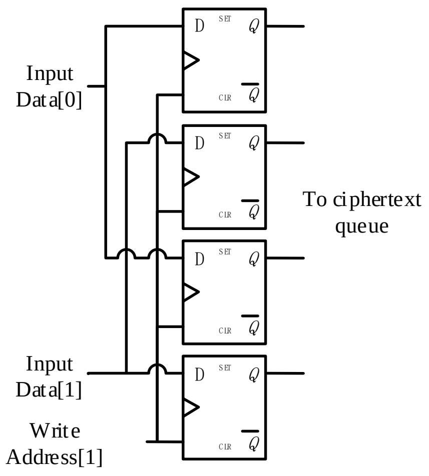

Fig. 8. Structure of plaintext queue for small scale system.

[1]. However, it is inevitable that the queue will not always have enough data to be transferred, thus causing plaintext transfer to pause during some clock cycles.

Also, the plaintext queue and the ciphertext queue work in a complementary manner. If the plaintext queue is being filled up, the ciphertext queue is being evacuated at the same time. When plaintext stops transferring data, the ciphertext queue only sends out data. The sum of the number of bits in the plaintext queue and the number of bits in the ciphertext queue equals M.

The minimum size of plaintext and ciphertext queue is $M = B + 3D - 2 = 444$, with $B = 128$ and input rate $D = 116$ [2]. This provides the maximum efficiency of $D/B = 90.625\%$ without any overflow in the plaintext queue or underflow in the ciphertext queue.

The plaintext queue has fixed width input, which is D bits. However, the output width is variable with 3 different cases: (1) B bits during normal counter mode of operation, (2) zero bits when the transfer is paused and (3) d bits with $1 \leq d < B$ because of the end of the blackout period.

For the input part, one way to simplify the hardware structure is to make the number of queue bits a multiple of input width. For illustration, consider a small scale system with $B = 3$, $M = 4$ and $D = 2$. As is shown in Fig. 8, the input width is $D = 2$ bits and there are $M = 4$ register bits. With a 2 bit write address, there are only two cases: (1) the write address starts at 00 and data is sent to 00 and 01, and (2) the write address starts at 10 and data is sent to 10 and 11. For $B = 128$, compared to the best case scenario (where $D = 116$), using a similar structure to Fig. 8 results in a smaller D and reduced throughput. For example, we can select $M = 320$ bits with the input set to be $D = 64$ bit width, thus reducing efficiency to $D/B = 50\%$. Although efficiency is lower than the 90.625% with $D = 116$, it is still substantially higher than that of stream cipher using bit by bit transfer or CFB mode and because M is a multiple of D, it is very efficient to implement the enqueuing process in the plaintext queue.

In Fig. 8, nothing is needed in the structure of output part of the plaintext queue. Each register output Q is connected to the D input of register bits in the ciphertext queue.

[1]Note that this is a correction to the constraint $M \geq B + 2D - 2$ given in [2].

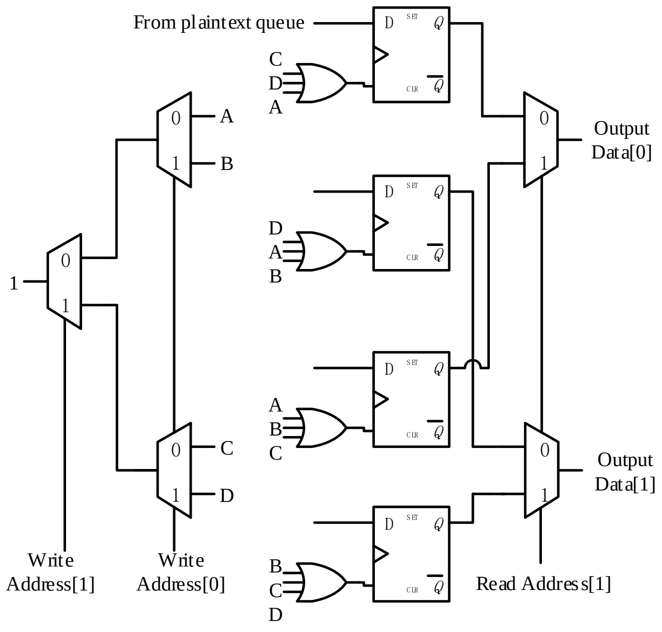

Fig. 9. Structure of ciphertext queue for small scale system.

Since the plaintext queue and ciphertext queue work in a complementary way, the read address of plaintext queue and write address of ciphertext queue are always the same. Only the read pointer in the control unit changes.

For the small scale system, although 3 bits data are always provided at the output, the increment of the read address is not always the same. Assume only 1 bit is needed at the end of blackout period. The output port provides 3 bits of data but the read address will only increase by 1. This means the downstream ciphertext queue will only take the first bit from the plaintext queue. The remaining 2 bits will be kept in plaintext queue. Similarly in the full scale system, if d bits ($1 \leq d < B$) are to be transferred, as discussed above in case 3, the read address will only increase by d regardless of B bits always being available from the plaintext queue.

E. Ciphertext Queue in Encryption

As can be seen from Fig. 9, the ciphertext queue can be functionally regarded as the reverse of the plaintext queue. The input width is variable and output width is $D = 2$ for the small scale system. All the data inputs are directly from the outputs of the plaintext queue. One difference is that enable signal must be considered. There are three 2-to-1 demuxes piled up as a pyramid. Demuxes in different layers are controlled by different bits in the read address, that is, the read pointer in the control unit. Each D flip-flop receives the enable signal from a 3 input OR gate. Although the OR gate receives enable signals from different outputs, there is only one high level signal among the three. This structure guarantees that each three consecutive D flip flops can be selected and enabled. Although there are 3 bits data to be sent to the input and saved, the write address may not increase by 3 if case 3 with $d < B$ occurs.

Because the output width is $D = 2$ bits, the output part contains two 2-to-1 muxes. One mux has data from bit registers 0 and 2, and another mux takes data from bit registers 1 and 3. As stated above, there are only two cases

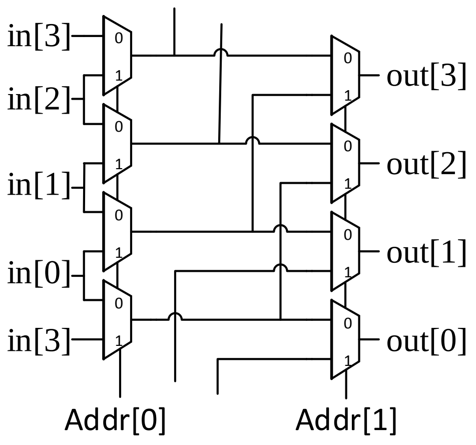

Fig. 10. Structure of barrel shifter.

for output: (1) the first mux will output bit 0 and the second mux will output bit 1, which is $00/01$ in address, and (2) the first mux gives bit 2 and the second mux gives bit 3, which is $10/11$ in address. Hence, only 1 bit of address is enough. Similar principles can be applied in scaling up the design to a full scale system.

F. Barrel Shifter

The barrel shifter can accomplish a cyclic shift using minimum hardware resources. A data sequence can be shifted by specified number of positions in one clock cycle. Fig. 10 shows a simple 4 bit barrel shifter, which can shift the data by 0 to 3 positions. In the first layer, the $address[0]$ can shift data by 1 bit. In the second layer, the $address[1]$ can shift data by 2 positions. Hence, for a 2 bit address of 00, 01, 10 or 11, the barrel shifter can shift input data by 0, 1, 2 or 3 positions.

For a barrel shifter operating on M bits, we would need $\lceil \log_2 M \rceil$ layers of muxes. In the full scale system with $B = 128$, $D = 64$ and $M = 320$, the size of the barrel shifter is 320 bits. It uses 9 layers and each layer can shift the data by 1, 2, 4, 8, 16, 32, 64, 128 and 256 positions.

In the PSCFB system, there are 3 barrel shifters. One is at the output part of sync pattern scanner and the other two are on the transfer path between the plaintext queue and ciphertext queue. In Fig. 6, the first barrel shifter at the output part of sync patten scanner receives data from two 128 bit registers and shifts it to the specified position of output port, which is from 254 down to 127. The second barrel shifter (labelled ”A” in Fig. 11) takes 128 bit data from pipelined AES and shifts it to the same position as the data to be transmitted from the plaintext queue. Then the output of plaintext queue and barrel shifter are XORed and received by ciphertext queue. The third barrel shifter (labelled ”B” in Fig. 11) is necessary because sync pattern scanner and ciphertext queue have different input width. The only difference is that this barrel shifter rotates data in the opposite way so that output data is at the fixed position. Fig. 11 and 12 illustrates the second and third barrel shifter, A and B, in the PSCFB encryption and decryption systems.

G. Decryption vs. Encryption

As shown in Fig. 11 and Fig. 12, the structural difference between encryption and decryption is that decryption does not have the 128 bit XOR before the input of sync pattern scanner. The positions of two queues are exchanged but the ciphertext queue in decryption is the same as the plaintext queue in encryption. Similarly, the plaintext queue in decryption and ciphertext queue in encryption are equivalent. For the control unit, there are only some small differences.

IV. Implementation and analysis

Both the PSCFB encryption and decryption systems are simulated and synthesized using Modelsim [9] and Quartus II [10] targeted to Altera Cyclone IV FPGA [4].

Table 1 is a summary of resource usage by component from the synthesis tool Quartus II. The FPGA is based on logic elements (LEs), and a logic element contains a look-up table (LUT) for combinational logic, and a register [4]. The encryption and decryption system cost 47% and 46% of the resource, repectively.

In encryption, the pipelined AES component costs 85% of the combinational logic and 54% of the registers. There are 11 AddRoundKeys, 10 SubBytes, 10 ShiftRows and 9 MixColumns because of pipelining, thus greatly increasing the resource usage. Since no memory is applied, key expansion and the SubBytes operation cost the most combinational logic elements in AES. There are ten SubBytes operations with each costing 3328 combinational logic elements. Key expansion costs 8861 combinational logic

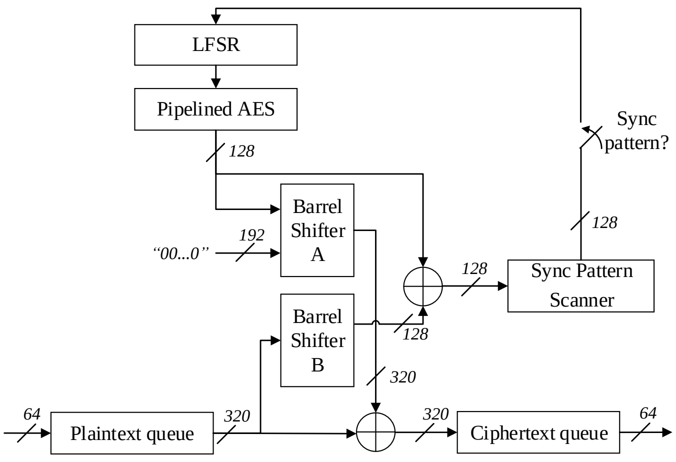

Fig. 11. Structure of encryption system (data path).

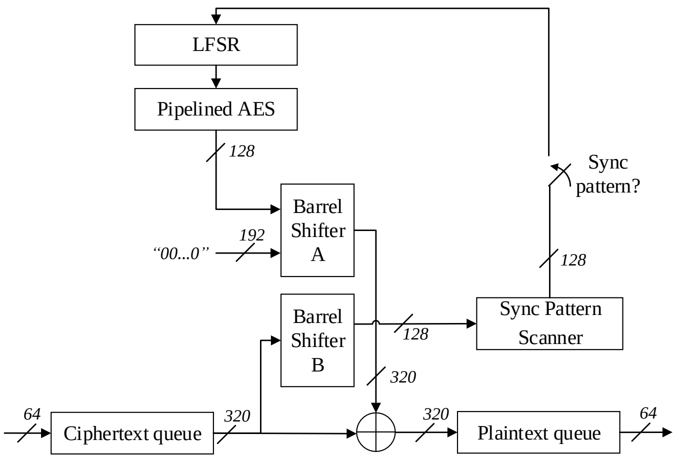

Fig. 12. Structure of decryption system (data path).

TABLE I

RESOURCE USAGE

Entity	Logic Element	Component	Combinational Logic Elements	Registers
Encryption	53298 LEs 47%	AES	45042	1280
		LSFR	378	129
		Plaintext queue	58	329
		Ciphertext queue	2258	332
		Sync pattern scanner	1632	276
		Barrel shifter A	1259	0
		Barrel shifter B	2241	0
Decryption	53185 LEs 46%	AES	45048	1280
		LSFR	378	129
		Ciphertext queue	58	329
		Plaintext queue	2367	332
		Sync pattern scanner	1646	276
		Barrel shifter A	1235	0
		Barrel shifter B	2370	0

TABLE II

TIMING ANALYSIS

Entity	Maximum Frequency	Throughput
Encryption	87.43 MHz	5.60 Gbps
Decryption	88.08 MHz	5.64 Gbps

elements. Also, 10 stages of registers cost 1280 registers. The plaintext queue with 320 bits of register costs only 58 combinational logic elements. However, the ciphertext queue costs more resources. The ciphertext queue with 320 bits of register costs 2258 combinational logic elements. From the design described above, it is obvious that demultiplexers and OR gates for the enable signal are a significant cost. The sync pattern scanner requires 1632 combinational logic elements and 276 registers. The second and third barrel shifters need 1259 and 2241 combinational logic elements respectively. The decryption part has similar resource usage.

Timing analysis is also conducted using TimeQuest Timing Analyzer [11] in Quartus II. The result is based on the worst case operating condition, slow 85°C, which provides slow silicon, low voltage and high temperature [12] and results in the slowest speed for the FPGA. For the encryption part, the maximum speed is 87.43 MHz and the throughput is 5.60 Gbps. For the decryption part, it can reach the maximum frequency at 88.08 MHz and the throughput is 5.64 Gbps. The throughput is calculated based on an efficiency of $D/B = 50\%$.

V. CONCLUSION

The design, implementation, simulation and synthesis of full scale PSCFB encryption system has been investigated. The simulation results shows that the system works successfully. In the system with 320 bit queues and 64 bit input/output width, the efficiency is down to 50%, which is still much higher than the efficiency of bit by bit stream ciphers or a self-synchronizing mode like CFB. Our synthesis results indicate that in the full system, only about 15% of the combinational logic and less than 50% of the registers are due to the PSCFB mode. The major resource cost comes from the pipeline structure of AES. In the future work, a PSCFB system with input/output width higher than 64 and queue size larger than 320 bits will be implemented and tested to reach higher throughput.

REFERENCES

[1] H.M. Heys, ”Analysis of the statistical cipher feedback mode of block ciphers,” *IEEE Transactions on Computers*, vol. 52, no. 1, pp. 77-92, Jan. 2003.

[2] H.M. Heys and L. Zhang, ”Pipelined Statistical Cipher Feedback: A New Mode for High-Speed Self-Synchronizing Stream Encryption,” *IEEE Transactions on Computers*, vol. 60, no. 11, pp. 1581-1595, Nov. 2011.

[3] NIST, *Advanced Encryption Standard (AES)*. FIPS PUB 197, 2001.

[4] Altera Cyclone IV Device Handbook, available at http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf.

[5] A.J. Menezes, P. van Pprschot, and S.A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.

[6] O. Jung and C. Ruland, Encryption with Statistical Self-Synchronization in Synchronous Broadband Networks, *Proc. Cryptographic Hardware and Embedded Systems(CHES 99)*, pp. 340-352, 1999.

[7] P.P. Chu, *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*. Wiley, 2006.

[8] M.C. McCoy, *Basic AES 128 Cipher*, available at https://code.google.com/p/inmcm-hdl/source/browse/trunk/AES.

[9] Modelsim-Altera Starter Edition, available at http://www.altera.com/products/software/quartus-ii/modelsim/qts-modelsim-index.html.

[10] Altera Quartus II Web Edition, available at http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html.

[11] TimeQuest Timing Analyzer Documentation, availavle at http://www.altera.com/support/software/timequest/sof-qts-timequest.html.

[12] Altera, Guaranteeing Silicon Performance with FPGA Timing Models, available at http://www.altera.com/literature/wp/wp-01139-timing-model.pdf.