

ECE746 Project

AES as A Stream Cipher

Project Specification Draft V1.1

George Mason University

Bin Zhou

G00368680

1 Introduction and Motivation

The main project goal is to implement AES using compact mode for comparison with eSTREAM ciphers. Different parameters will be investigated with trading speed/throughput vs. area.

One purpose for this work is to try to implement AES efficiently as a stream cipher with a nice speed and area, compared to the other eSTREAM ciphers; the other is to try to implement the compact architecture, which could extend the AES usage to a broader area, such as cell phones, wireless networks, video stream systems, etc., which is of significant importance for resource-limited environments.

Some background:

The AES is usually considered as a block cipher. But by using block cipher mode, it could easily be turned to stream cipher. And there is some work done on the AES as a stream cipher. In [1], the AES was implemented on a small FPGA using an application specific instruction processor; in [2] and [5], a compact architecture is introduced, using the data path widths equal to 64-bit, 32-bit, and 8-bit.

The Mode Selection:

The AES Counter (CRT) mode, Output Feedback (OFB) mode and Cipher Feedback (CFB) mode are suitable for stream cipher. However, when using as a stream cipher, the resources are more restricted, which requires the working mode should be as small as possible and better pipelined to gain speedup.

First, we consider the same block for both encryption and decryption. The electrical codebook mode requires separate encryption and decryption block (figure 1), which is not good. This will consume more resources for only one block. So here we eliminate ECB mode.

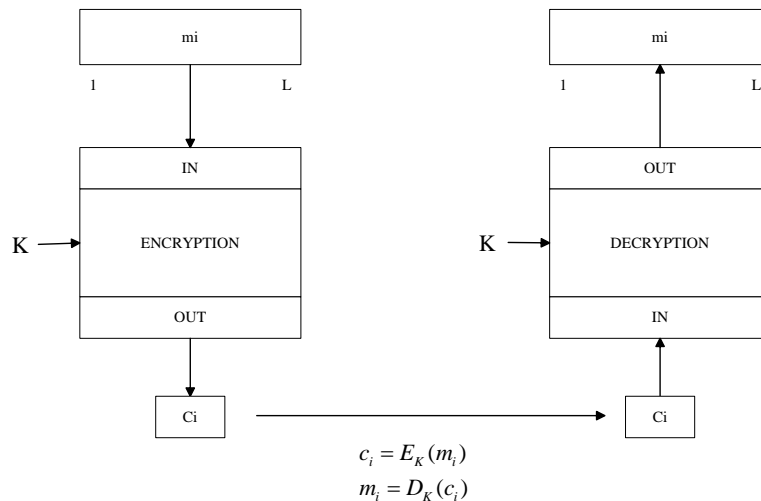


Fig 1 ECB mode, separate encryption and decryption

Then we consider the pipeline requirement for the stream cipher. It seems that intermediate stages could be added to CRT, OFB and CFB modes. The difference between CRT with the other two modes is that CRT has no feed back to the state part. As shown in Figure 2, it is easy to add pipeline stages to CRT mode. The next stage $IS_{i+1} = IS_i + 1$ is predictable and the pipeline could run without waiting for the intermediate states.

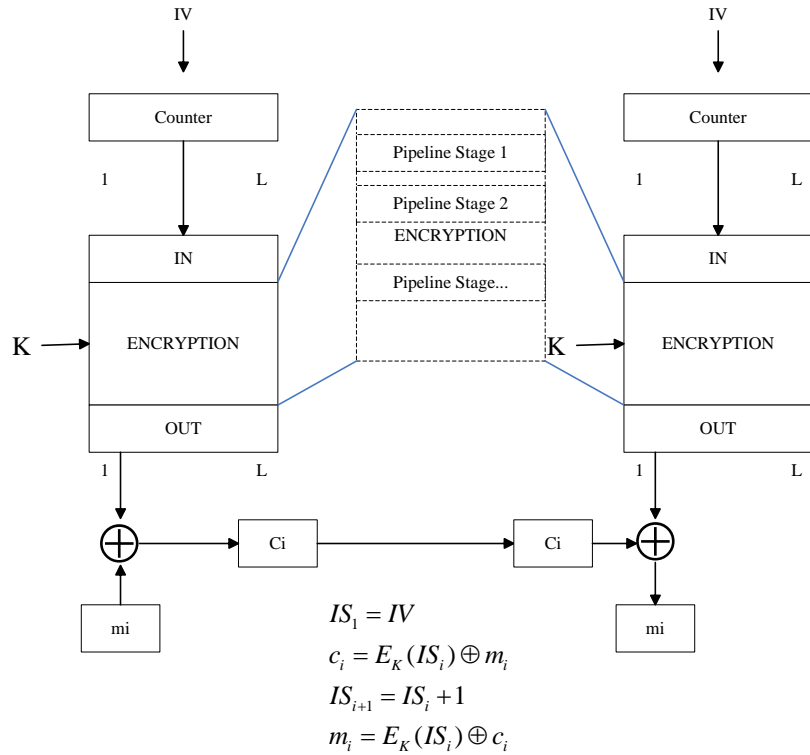


Figure 2 Adding Pipeline stages to CRT mode

However, things are different for OFB and CFB modes when adding stages. Figure 3 shows why this is not good for pipeline.

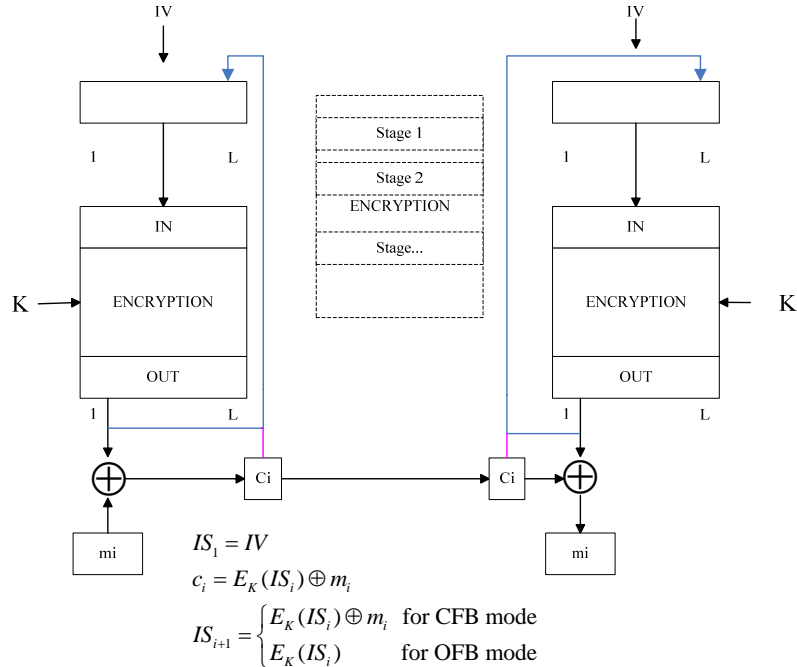


Figure 3 Adding intermediate stages to Feedback modes

When adding stages, the next state value IS_{i+1} will depend on the last stage value IS_i in both modes. So the next iteration could not start until the last stage completes. This will slowdown the whole process rather than speed it up.

So we choose CRT mode as our implementation subject.

2 Design Related Options

(1) Design entry method

VHDL as the main method.

(2) Target implementation

Xilinx Spartan 3 XC3S50-5, the minimum device first.

(3) CAD tools used to specify, implement, and verify the design

Xilinx ISE 8.2 and Modelsim6.0

3 Additional libraries of basic digital circuits required

Currently None.

4 Detailed Specification of the IO of the circuit(s).

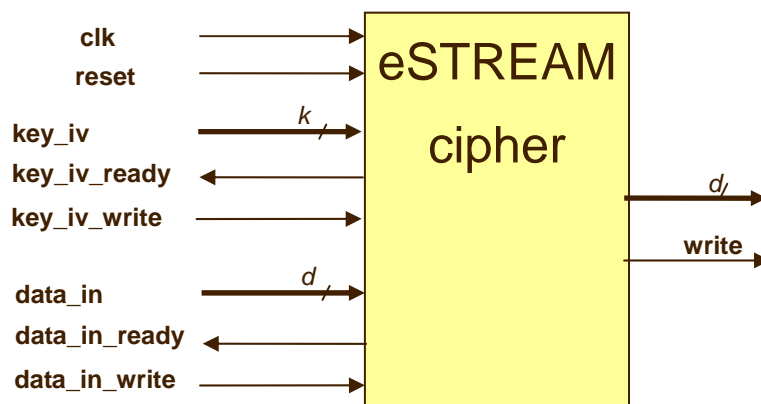


Fig 1 Interface (David Huang, ECE545 Project Slides,GMU)

NAME	PORT	BITS	DESCRIPTION
clk	IN	1	System clock
reset	IN	1	Active-high reset
key_iv_ready	OUT	1	When cipher is ready to receive programming of key and initialization vector (IV), cipher sets key_iv_ready=1. When not ready, cipher sets to 0.
key_iv_write	IN	1	After cipher sets key_iv_ready=1, external user sets key_iv_write=1 to begin programming the key and IV.
key_iv	IN	k	After setting key_iv_write=1, user puts the key onto the key_iv bus k-bits at a time. Immediately after the key is programmed, user puts the initialization vector (IV) onto this bus k-bits at a time. When the cipher recognizes all key and IV bits have been programmed, it sets key_iv_ready=0.
data_in_ready	OUT	1	After key and IV are programmed, the cipher initializes itself. After initialization is done, the cipher sets data_in_ready=1 to indicate it is ready to receive input data.
data_in_write	IN	1	After cipher sets data_in_ready=1, user sets data_write=1 to begin sending input data to the cipher.
data_in	IN	d	After setting data_write=1, user puts the input data onto this bus at d-bits per clock cycle. There should be a "natural" setting of d for any given cipher; many ciphers have d=1 as the natural size.

write	OUT	1	When data_out is valid, cipher sets write=1. If data_out is not valid, cipher sets write=0.
data_out	OUT	d	Output data of the stream cipher, which comes out d-bits per clock cycle, just as the input comes in d-bits per clock cycle.

(David Huang, ECE545 Project Slides,GMU)

5 Brief Description of the Function

- (1) First, AES in counter mode will be first implemented in an iterative architecture of word width = 128bits. Second, the compact architecture [2] will be implemented in 64, 32, and 8 bits. Both pipelined and non-pipelined implementation will be experimented. The optimization criteria include minimum area, maximum throughput and throughput/area ratio.
- (2) The S-box in the AES SubByte operation will be implemented in lookup table first and then using pure logic to take place of it. This is of benefit to the speed. And pipeline stages could be inserted to the logics. The logic will be implemented as instructed in Chapter 6.1 of [2]
- (3) The key scheduling will be implemented first as storing the round keys in the memory. This requires the 11*128 bits. Then the on-the-fly method will be implemented, which generates the round key with each round runs. It follows the architecture in Fig. 55 of [2].
- (4) The compact mode will be of data path 8, 32, and 64 bits. The basic architecture follows the one in chapter 7.3 of [2].

6 Procedures for Testing

a. The simulator in use.

C/C++ Source code for debugging (intermediate results), HW3 for partial function results

b. The source of test vectors.

(1) From the document: NIST, “Announcing the ADVANCED ENCRYPTION STANDARD (AES)”, Federal Information Processing Standards Publication 197, November 26, 2001

(2) NIST website for basic ECB mode results

c. The format of input stimuli.

VHDL test bench. The test vector is in a file called “vectorfile.txt”

The basic input format is like:

```

0
ffffffffffffffffffffffffffffffff
0123456789abcdef0123456789abcdef
0123456789abcdef0123456789abcdef
4a8a9840b5d683d8b8cbe3f19a12b024
    
```

The first line is the encryption (0) or decryption (1) bit. The second line is the key and the third line IV. The fourth line is the plaintext and the last line is the expected cipher text. The values are in hexadecimal format.

The output format is like:

```

Key=FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
IV=0123456789ABCDEF0123456789ABCDEF
plaintext=0123456789ABCDEF0123456789ABCDEF
cipher=4A8A9840B5D683D8B8CBE3F19A12B024
expectedcipher=4A8A9840B5D683D8B8CBE3F19A12B024
    
```

e. Parameters to be determined using the implementation tools.

Parameters	Tool
number of CLBs	ISE XST and Synplify
LUTs, FFs, BRAMs,	ISE XST and Synplify
minimum size FPGA device able to hold	ISE XST and Synplify
Maximum Clock Speed	ISE&Modelsim(post-route simulation)

7 Plan of Simulation Experiments

The simulation experiment will cover the matrix as the following table:

	S-Box	Data path width	Key Scheduling
Pipeline	Pure logic	8(compact)	On the fly
		32(compact)	On the fly
		64(compact)	On the fly
Non-pipeline		8(compact)	On the fly
		32(compact)	On the fly
		64(compact)	On the fly
	Lookup table	128	On the fly
	Lookup table	128	In Memory

The main test objects are performance, resource utilization and throughput/area ration, shown in the following table

Item	Data rate (bits/Clock cycle)	Maximum Frequency (MHz)	Maximum Throughput (Mbps)	Area(slices)	Throughput/Area (Mbps/slice)	Device
------	---------------------------------	-------------------------------	---------------------------------	--------------	---------------------------------	--------

8 Time schedule

Dates	Progress	Goals
February 23	first version of the project specification	1 Background Reading&Research 2 Clarify the goal and the main architecture 3 Clarify the parameters needed to compare with eSTREAM
March 1	final project specification	1 Clarify the whole design and all the needed details 2 For s-box, pure-logic s-box and compact architecture
March 24, 26	first progress report	1 Finish the s-box design, with look-up table 2 Understand the logic implementation
April 14, 16	second progress report	1 Finish the pure-logic s-box design 2 Finish the compact architecture design
April 28, 30	final progress report	Finish all the coding and simulation work
May 8	project reports	Try to improve the performance
May 12	Final reports, presentation	As it should be

9 Possible Changes Areas

- (1)Architecture of implementation might change when using pure logic instead of lookup table.
- (2)Pipeline design might be introduced between the stages of the logic and some other component

10 Tentative Contents of Final Report

Item	Description
Background and Description	Including the topic choice, motivation and backgrounds
Final Architecture & Specification	The final architecture used and how the whole functionality works
Alternative Design	If there is other design mode and why it's not chosen
Source Code and Major Modules	As it should be
Test Schemes	Test methods, test tools, test vectors and results, to make sure the correctness.
Deviations, Problems and unresolved issues	As it should be
Results	The performance; throughput, clock speed; the resource utilization; timing for both simulation and post-synthesis, post-route, and etc.
Optimization Considerations	Pipelining, Pure logic instead of lookup table, Architecture improvement, Parallel processing, CAD tool options, others
Comparison with other eSTREAM ciphers	Compare the results with others
Conclusion	As it should be.

11 List of literature

- [1] Tim Good and Mohammed Benaissa, "AES as Stream Cipher on a Small FPGA",
- [2] Kris Gaj and Pawel Chodowiec, "FPGA and ASIC Implementations of AES", book chapter,
- [3] NIST, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)", Federal Information Processing Standards Publication 197, November 26, 2001
- [4] Network Working Group, R. Housley, "RFC 3686 - Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", Internet RFC/STD/FYI/BCP Archives, January 2004
- [5] Paweł Chodowiec and Kris Gaj, "Very Compact FPGA Implementation of the AES Algorithm", Cryptographic Hardware and Embedded Systems -- CHES 2003: 5th
- [6] H. Lipmaa, P. Rogaway, and D. Wagner, "CTR-Mode Encryption, Comments to NIST concerning AES Modes of Operations", Symmetric Key Block Cipher Modes of Operation Workshop, 2000
- [7] Kris. Gaj, "AES implementations in software and hardware", ECE746 Lecture, GMU

12 Other Things

- (1) Compact mode requires folded registers, which should be carefully treated.
- (2) The efficiency comparison should be also important. (How many resources could give same security level).