

George Mason University
Department of Electrical and Computer Engineering
ECE746 – Secure Telecommunication Systems

S-BOX vs. T-BOX BASED
ITERATIVE ARCHITECTURE OF AES

Bhupathi Kakarlapudi

Nitin Alabur

Contents

1. Introduction	2
2. Design entry, target implementation and CAD tools used	2
3. Detailed specification of the input and output of the circuit	3
3.1 Inputs to the interface	3
3.2 Outputs from the interface	4
4. Brief description of the function performed by the circuit, references to standards and detailed descriptions of algorithms	4
5. Procedures for testing the functionality and performance of the circuit	4
6. Parameters to be determined using the implementation tools	5
7. Plan of simulation experiments to be performed using the circuit	5
8. Time schedule, intermediate goals to be achieved	6
by the dates of progress reports	
9. Possible areas, where the specification can change depending	6
on the progress of the project	
10. Tentative table of contents of the final report	6
11. List of literature	7

1. Introduction

In 1997 NIST (National Institute of Standards and Technology) started a contest to develop a Federal Information Processing Standard (FIPS) that specifies an encryption algorithm capable of protecting sensitive government information well into the next century. The algorithm was to be used by the U.S. Government and, on a voluntary basis, by the private sector. The winner of the contest, Rijndael algorithm was finally chosen after 5 years of extensive analysis and standardization process which was open and transparent process unlike its predecessor (DES) Data Encryption Standard.

AES hardware implementations on Xilinx Virtex, using S-boxes has been considered by NIST and the algorithm has also been implemented on the Altera Flex, Acex and Apex devices (both S-boxes and T-boxes)^[1] by Viktor Fischer and Milos Drutarovsky.

The AES algorithm has only been implemented using S-boxes and the T-box implementations have been realized in software and hardware implementations on Altera devices. Because of the significant performance improvement by the T-box implementations over the S-box, our motivation is to implement the algorithm using T-boxes in hardware and to compare the performance of both the versions on low cost Xilinx Spartan 3 and on high performance Xilinx Virtex 5 families. The implementations will be capable of handling key sizes of 128, 192 and 256 bits and a block size of 128 bits. Although the encryption and the decryption units can be designed separately, the implementations will have both the units using the common modules and the encryption or the decryption feature is selected based on an external select signal.

The aim is to implement the non-pipelined and the pipelined versions of S-box and T-box architectures on Spartan 3 series and compare the performance factors (max throughput/area ratio) with that of Virtex 5 series. Also all the implementations will be capable of handling keys of length 128, 192 and 256 bits

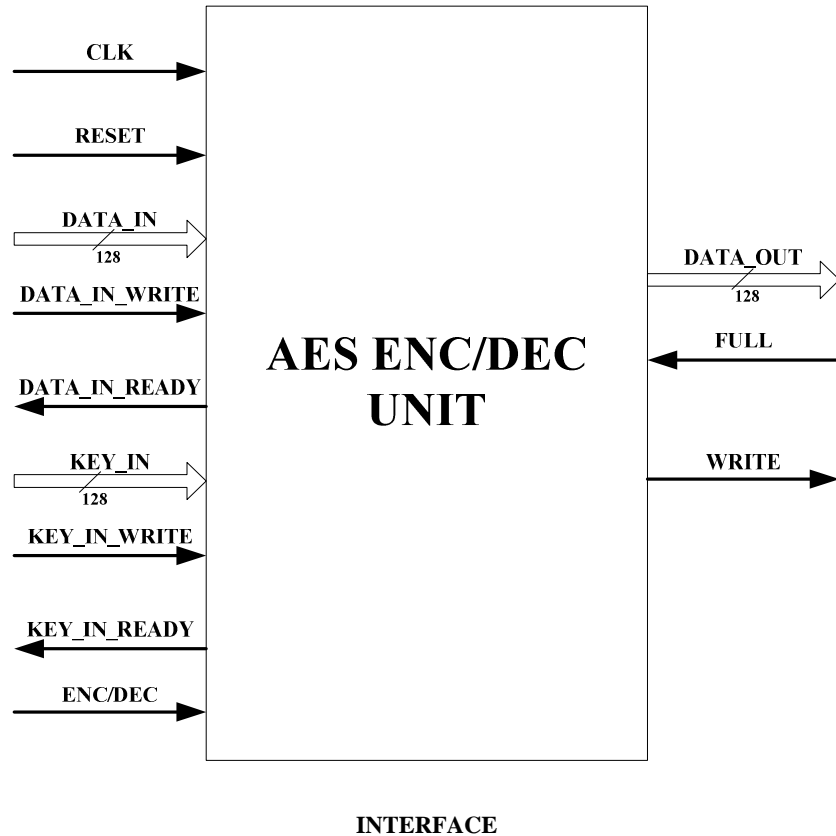
Possible areas for future work:

- ASIC implementations of the S-box and the T-box architectures.
- Suggest suitable device based on the analysis of possible mapping methods for the Xilinx devices.
- Implementation in CBC, CFB, OFB, CTR, GCM, CCM modes

2. Design entry, target implementation and CAD tools used

Design entry is through VHDL and the target devices are Xilinx (Spartan and Virtex) and ASIC. CAD tools to be used are Aldec Active HDL for design entry and simulation, Synplicity Synplify Pro and Xilinx ISE webpack for FPGA synthesis and implementation, Design Analyzer and Synopsys Prime Time for ASIC synthesis and static timing analysis.

3. Detailed specification of the input and output of the circuit



3.1 Inputs to the interface

DATA_IN: The input block of data that is to be encrypted or decrypted (bus width depends on the target device or the minimum required bus width).

KEY_IN: 128 bit key input, that is later to be converted into round keys, calculated in parallel to the round operations.

KEY_IN_WRITE: The interface reads the KEY_IN value only when this signal is high.

CLK: System clock

FULL: The interface will freeze all the internal operations when this signal is high and resume from the present state when the signal goes low.

DATA_IN_WRITE: The interface reads the DATA_IN value only when this signal is high.

ENC/DEC: Based on this signal the interface will perform either the encryption or decryption on the input data.

RESET: Master reset, when high all the internal states are set to the initial state.

3.2 Outputs from the interface

DATA_OUT: Output data from the interface. The data is either cipher text or plain text based on ENC/DEC input.

DATA_IN_READY: This signal is generated when the interface is ready to accept DATA_IN.

KEY_IN_READY: This signal is generated when the interface is ready to accept KEY_IN.

WRITE: This signal is high when the output is being generated by the interface.

4. Brief description of the function performed by the circuit, references to standards and detailed descriptions of algorithms

As per the Advanced Encryption Standard by NIST^[2] the data input or the block size is required to be 128 bits wide with key sizes of 128, 192 and 256 bits wide. The implementation in this project will be with key sizes 128, 192, 256 bits requiring a total of 10, 12 and 14 iterations respectively^[3].

The S-box architecture has look up tables that are used to perform the sub bytes operation and all other operations are performed according to their descriptions.

In T-box architecture, all the round operations are performed by the table look ups and XORs^[3]. The last iteration for the T-box does not include the mix column operation.

The pipelined versions of both the architectures are to be implemented by inner round pipelining, where each individual stage internally since the feedback of data from the last round to the first round during the first 9 iterations constricts the ability of pipelining the complete architecture. Although the architectures can also be pipelined via loop unrolling, it requires a very large area and may be realized on larger Virtex series.

5. Procedures for testing the functionality and performance of the circuit

The simulator in use:

Aldec Active HDL

The source of test vectors: The test vectors given at the link below on the NIST website for the AES algorithm, will be used for the validating the implementations. This source of test vectors

provides the state values after every iteration of all the intermediate rounds, for a given key and data input. The source can be found here^[4].

Procedure

The format of the input stimuli will be in the hexadecimal format. The primary stage of testing would be during the process of debugging the individual modules and later the debugging can be carried out for modules that are higher in the hierarchy. Once the complete design is debugged and almost free of bugs, further testing can be carried by giving the same test vectors to different designs (pipelined, non- pipelined, S-box, T-box) and the outputs compared with each other.

6. Parameters to be determined using the implementation tools

Tentative comparison tables:

Key Size	Parameter		S-box		T-box	
			Non-Pipelined	Pipelined	Non-Pipelined	Pipelined
128	Throughput					
	Area	CLB Slices				
		Memory				
	Throughput/Area _{CLB}					
	Latency					

Key Size	Parameter		S-box		T-box	
			Non-Pipelined	Pipelined	Non-Pipelined	Pipelined
192	Throughput					
	Area	CLB Slices				
		Memory				
	Throughput/Area _{CLB}					
	Latency					

Key Size	Parameter		S-box		T-box	
			Non-Pipelined	Pipelined	Non-Pipelined	Pipelined
256	Throughput					
	Area	CLB Slices				
		Memory				
	Throughput/Area _{CLB}					
	Latency					

Will be determining the suitability of the available devices based on the resource utilization (LUTs, CLB Slices, Block ROMs, available IO resources). Suggest modifications to the design based on resources available on the target devices.

7. Plan of simulation experiments to be performed using the circuit

The only circuit that may be required would be to compare different types of designs. The circuit would have both the designs in the circuit in parallel, with a single common data input (input will

be given after splitting in the required format for the respective designs) and the individual outputs being compared by another simple logic (just an XOR gate comparing both the outputs).

For implementing these architectures on smaller devices of the Spartan 3 series with less IO pins, the data and the key input will be having a reduced width based on the number of available IO pins and will be registered within the device will be given as 128 bit inputs to the implemented architecture interface.

8. Time schedule, intermediate goals to be achieved by the dates of progress reports

Finish Coding – March 23(S-box to be completed first and then T-box)

Complete debugging – April 6th (First progress report due)

Pipelined Version of both S and T boxes – April 16th (Second report due)

Final Progress report due - April 28 - 30

Debugged Pipeline version – April 31st

Project report and draft viewgraphs – May 8th

Final project reports, presentation, and source & data files – May 12th

9. Possible areas, where the specification can change depending on the progress of the project

The ASIC implementations may remain incomplete if the pipelined version requires more time than expected.

10. Tentative table of contents of the final report

Abstract

Introduction

AES standard and AES hardware architectures (explanation of all the rounds and key scheduling, encryption and decryption, sharing of resources between enc/dec)

Results of implementation

Discussion of the T-Box and S-Box implementations realized

Comparison of results with known implementations

Conclusion

References

11. References

1. <http://www.kemt.fei.tuke.sk/publication/Drutarovsky/ches2001.pdf>
2. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
3. http://ece.gmu.edu/courses/ECE746/project/S08_Project_resources/AES.pdf
4. Appendix C (page 35) - <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
5. http://ece.gmu.edu/courses/ECE746/viewgraphs_S08/lecture4_AES_implementations_2.pdf