High Throughput & Low Area AES

Core Specifications

File name: *HT LA AES Core Specifications*
Version: *0.2*
Creation Date: *March 28, 2012*
Update Date: *April 1, 2012*
Authors: *Moti Litochevski*
Luo Dongjun

Table of Contents

Index of Tables

Index of Figures

1. Preface

1.1. Scope

This document describes the High Throughput and Low Area AES IP core operation and interfaces.

1.2. Revision History

Rev	*Date*	*Author*	*Description*
0.1	03/28/12	Moti Litochevski	First Draft
0.1	04/01/12	Moti Litochevski	Correct i_key_mode input signal size in the block diagram

1.3. Abbreviations

AES	Advanced Encryption Standard
KAT	Known Answer Test vectors used to test the AES core
ECB	Electronic Code Book mode of operation for the AES standard
bps	Bits per second
Bps	Bytes per second

2. Introduction

The High Throughput Low Area AES IP core implements the Rijndael encryption & decryption algorithm used in the AES standard. The standalone core implements the basic ECB mode described in publication 800-38A by NIST. Other modes can be easily implemented using the core.

The core implements both key expansion, required each time the key is changed (also after reset or power-up), and encryption/decryption algorithms. The core supports all three key lengths: 128, 192 & 256 bits, selected by an input signal. Encryption or decryption modes of operation are also selected on the fly using an input signal.

The HT LA AES core interfaces are implemented with the full width of the required data to minimize time required to load the information. Thus, the key and data interfaces are 256 bit and 128 bit respectively. The above implies that the core requires very wide buses and will not synthesis to many FPGA packages. A very simple and relatively low performance wrapper module is supplied with the core which implements a 32 bit data bus interface to write & read key & data to/from the core. This top module was used for synthesis results even though the logic also includes wide internal registers. Detailed description of the core interfaces and timing is given in the "Core Interfaces" section of this document.

To support high data rates the core enables pipeline operation of four vectors simultaneously. Using a ready output or some fixed timing the user may input up to four plain/cipher text vectors into the core for encryption or decryption. The supplied basic test bench demonstrates this functionality for all key sizes.

The following table summarizes the maximum core performance for each key size. The table also provides the maximum encryption & decryption data rate (bps) achievable by the core per clock frequency (Hz). Note that the calculation assumes that the core is encrypting or decrypting four plain / cipher text vectors using pipeline operation.

Key Size	*Operation*	*Clock Cycles*	*Maximum Data Rate*	*Maximum Data Rate with 100MHz clock*
128 bits	Key Expansion	45		
	Encryption / Decryption	42	12.19 bps/Hz	1.2Gbps
192 bits	Key Expansion	37		
	Encryption / Decryption	50	10.24 bps/Hz	1024Gbps
256 bits	Key Expansion	57		
	Encryption / Decryption	58	8.83 bps/Hz	882Mbps

Table 1: Core Performance Summary

The core was verified using Icarus Verilog simulator with two test benches: the first demonstrates basic operation of the core operation & timing and the second tests the core using all available KAT test vectors for ECB mode of operation. A more detailed description of the operation of the test

benches is given in the “Test Bench Description” section of this document.

The core uses an internal small RAM memory to store the expanded keys. This memory is synthesized as distributed memory and has two possible configurations which are only slightly different and give slightly different synthesis area & timing results. To select between the two possible configurations the user should comment / uncomment the Verilog define statement in the core top module file “aes.v”.

The following table summarizes some synthesis results of the core for different FPGA families. Note that for the results below the core was synthesized with the supplied “aes_top_example.v” as the top module and with the RAM memory define, described above, enabled which requires more logic in the FPGA but provides better timing results.

Manufacturer	*Family*	*Device*	*Device Utilization*	*Elements Utilization*	*Fmax*
Xilinx	Spartan 3	xc3s1500-4fg456	23.00%	3,110 Slices	>100MHz
Xilinx	Virtex 5	xc5vlx30-3ff324	45.00%	1,408 Slices	>135MHz
Altera	Cyclone III	ep3c10f256c6	44.00%	4,657 LEs	>110MHz
Altera	Arria II GX	ep2agx45cu17i3	13.00%	1,806 Registers 3,247 ALUTs	>145MHz

Table 2: Synthesis Results for Different FPGA Devices

The above results where obtained using the following software versions:

- Xilinx ISE Webpack 13.4
- Altera Quartus Web Edition 11.1

3. Architecture

The HT LA AES core is basically a straight forward implementation of the Rijndael algorithm. The following block diagram depicts a simplified block diagram of the core including its interfaces and modules.

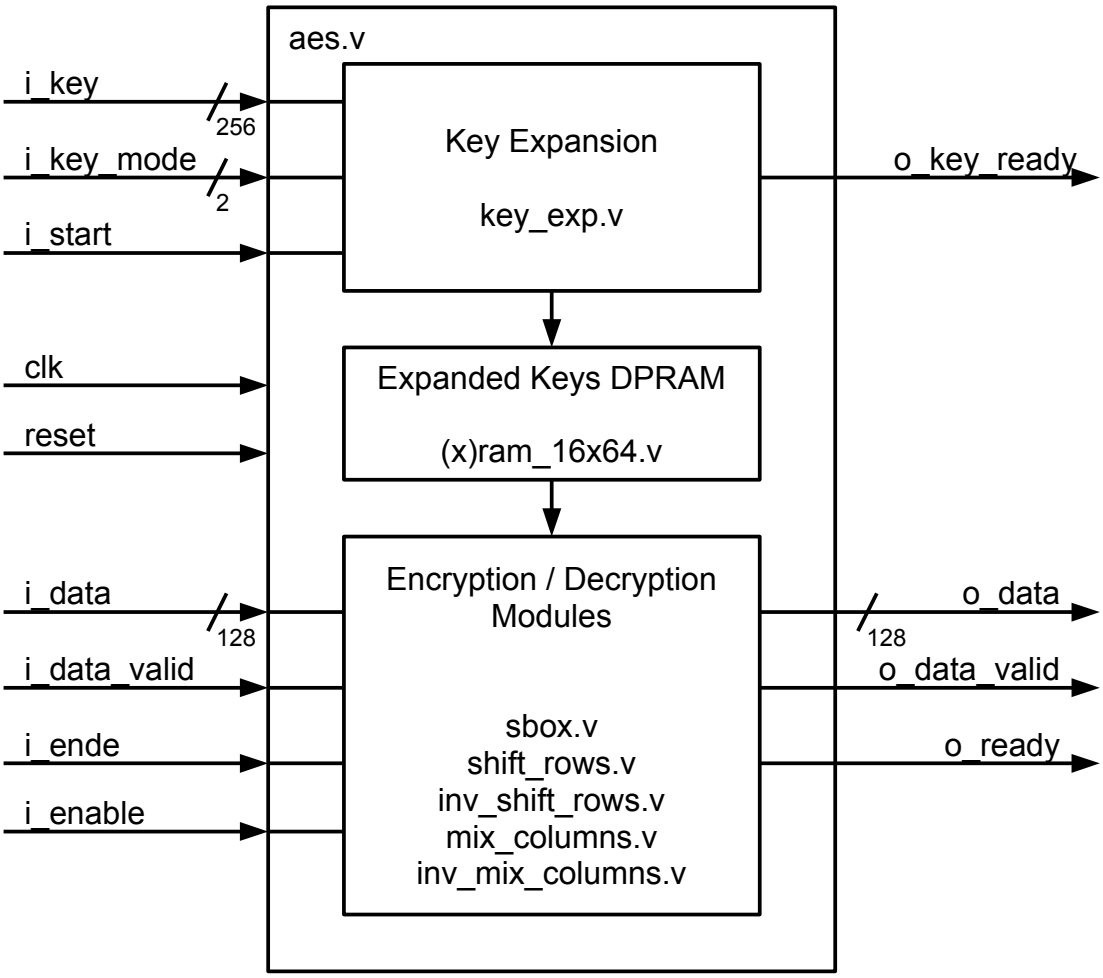

Figure 1: Simplified Core Block Diagram

Although possible, no effort was made yet to reduce the ports sizes or optimize the implementation. The core is flexible and includes all the required building blocks.

As mentioned in the introduction section, an example of a top module wrapper is provided which enables interfacing the core using a simple 32 bit data bus. Key and input data vectors can be written using the 32 bit data bus and a 4 bit address bus selecting the written 32 bit word destination. A different 2 bit address bus selects the core output data portion presented on the 32 bit output data bus.

4. Core Interfaces

The following table summarizes the core interface ports and gives a description of their functionality.

> *Note:*
> *The port direction in the table above is as defined in the core top module.*

Name	*Direction*	*Width*	*Description*
clk	input	1	Global core clock signal.
reset	input	1	Global core asynchronous reset, active high.
i_key	input	256	Key value input bus. This bus is used to input the key into the core for all key size. For 192 bits key mode the 192 upper bits [255:64] hold the key value. For 128 key mode the 128 upper bits [255:128] hold the key value.
i_key_mode	input	2	Selects the key size mode according to the following list: • 2'b00 value selects 128 bit key length • 2'b01 value selects 192 bit key length • 2'b10 value selects 256 bit key length This value should not change during operation of the core unless the key mode is changed and key expansion is restarted.
i_start	input	1	Key expansion start pulse signal. The values presented on the i_key and i_key_mode buses are used to during key expansion.
o_key_ready	output	1	Key expansion is done and the expanded key is ready for use. This signal is a level signal and will only clear after reset and during expansion of a new key triggered using the i_start input.
i_data	input	128	Plain or cipher text data input to the core. This bus holds the value of the data for the encryption or decryption operation. This bus is sampled when a new operation is started and may be changed after the operation started.
i_data_valid	input	1	Data input valid pulse indicates that the data input bus is valid and starts encryption or decryption operation if the core is ready for a new data. This signal should be be asserted if the o_ready signal was not asserted at the previous clock cycle.
i_ende	input	1	Mode of operation selection control. When this signal is set to logic low (1'b0) the core will encrypt the input data. When this signal is set to logic high (1'b1) the core decrypt the input

Name	*Direction*	*Width*	*Description*
			data.
i_enable	input	1	Core encryption / decryption modules clock enable control.
o_data	output	128	Cipher or plain text data output from the core. This bus presents the value of the data output from the encryption or decryption operation.
o_data_valid	output	1	Output data bus valid pulse indicating that the value on the o_data bus has changed and holds the result of the corresponding operation.
o_ready	output	1	Input ready for new data output. This signal indicates that the core is ready for a new input data on the next clock cycle. This signal is generated from internal state of the core and is only valid for the next clock cycle since the new data is pushed in the processing pipeline of the algorithm.

Table 3: Core Interfaces Description

5. Test Bench Description

The 'verilog\bench' directory contains two test benches files. Compilation batch files are included in the 'verilog\sim\icarus' directory used to simulate the core using Icarus Verilog. The directory includes two compilation batch files: one for the basic test bench simulation and the second for the KAT verification simulation.

The directory also includes batch file to run the simulation, 'run.bat', and another to call gtkwave to view the simulation VCD output file, 'gtk.bat'.

For the KAT verification simulation the test bench first reads the 'KAT_files.txt' file which includes the list of KAT ECB vectors files used during the simulation. The KAT files are parsed and the expected results are verified using the given key and data for the required key mode and operation. Additional files can be added using exactly the same syntax as in the original KAT files.