algotronix®
130-10 Calton Road
Edinburgh, Scotland
United Kingdom, EH8 8JQ
Phone: +44 131 556 9242
E-mail: cores@algotronix.com
URL: www.algotronix.com

Core Facts

Provided with Core	
Documentation	User Manual
Design File Formats	VHDL Source Code
Verification	Test Bench, Test Vectors
Instantiation templates	VHDL
Reference designs & application notes	'Getting Started', 'Choosing Configuration Options'
Additional Items	'Getting Started' Sample design demonstrates core on Xilinx Spartan 3 Eval Board
Simulation Tool Used	
Model Tech ModelSim, ISim	
Support	
Support provided by Algotronix	

Features

- Selectable internal and external data path widths of 8, 16, 32, 64 or 128 bits allow optimal area vs performance tradeoff
- Full Implementation of FIPS 197 and SP800-38A
  - 128, 192 and 256 bit keys
  - Encrypt only, Decrypt only or Encrypt/Decrypt
  - ECB, CBC, OFB, CTR, CFB1, CFB8 and CFB128 mode
- Full Implementation of AESAVS testbench
  - Regression Mode for confirming design functionality
  - Qualification Mode for generating response files for certification lab.
- Supplied as VHDL source code to allow security review
- Available under the terms of the SignOnce IP Licence

Applications

- Government/Military
- Satellite
- Secure Computer Systems
- Gaming Machines
- Storage Area Networks (SAN)
- Digital Rights Management (DRM)
- Wireless Networking

General Description

The Algotronix G3 AES Core is the third generation of Algotronix comprehensive, production tested implementation of the NIST Federal Information Processing Standard 197 (FIPS197). G3 improves over the popular G2 product by allowing the widths of internal and external data path to be selected by the user; G3 also includes optimisations which allow it to implement the AES algorithm in fewer clock cycles than leading competitors and improve clock frequency and area on Xilinx FPGAs significantly compared with G2. Wider data paths require more logic and RAM blocks but provide higher performance, thus by allowing fine control over the data path width the G3 product allows the user to achieve the required performance with the minimum area. A single G3 core licence offers all the implementation options which competitive IP vendors would split into separate 'standard', 'small' and 'fast' products making a G3 multi-project licence particularly cost effective. For very high performance (multi gigabit) applications a parallelised ECB mode where multiple data paths simultaneously operate on the same data stream can be purchased separately.

The G3 core supports all the key lengths specified in the FIPS 197 standard (128, 192 and 256 bits) and all the modes of use specified in NIST SP800-38A – ECB, CBC, OFB, CTR, CFB1, CFB8 and CFB128. The core is supplied with a comprehensive testbench which implements the NIST AESAVS test suite for AES. The testbench can be used in regression mode to verify changes to the source code or in qualification mode to generate response files for a NIST approved laboratory.

The Algotronix AES core is supplied as VHDL source code and can be configured using a number of VHDL generic parameters to select only those features which are required in order to conserve area. The core can be configured as Encryptor, Decryptor or Encryptor/Decryptor and the maximum key length and required modes can also be selected. The core can be configured to generate the key schedule in hardware or to save area a software generated key schedule can be loaded. This level of flexibility makes it easy to experiment with area/performance/functionality tradeoffs and makes it highly likely that the core will be useful in multiple projects. The wide applicability of the core makes multi-project site licences particularly attractive.

The AES core is an easy to use fully synchronous design with a single clock and an enable signal to allow the core to be started and stopped on a clock cycle by clock cycle basis to match up with external data sources. The core has been designed for efficiency in Xilinx FPGAs and makes full use of FPGA specific features such as dual port memory blocks.

Table 1a: Example Implementation Statistics with a 128 bit datapath – ECB, Encrypt Only, 128 bit key, 'Online' Hardware Key Expansion, 'Push Button' flow with Fmax specified by clock constraint

Family	Example Device	Fmax (MHz)	Slices / CLBs[1]	IOB[2]	GCLK	BRAM	MULT/ DSP48	DCM	Throughput (MBit/sec)	Design Tools
Kintex	XC7K325-1	266	435	394	1	0	0	0	3413	Vivado 2014.1
Zynq	XC7Z100-2	222	443	394	1	0	0	0	2841	Vivado 2014.1
Kintex Ultra Scale	XCKU035-1	222	263	394	1	0	0	0	2841	Vivado 2014.1
Spartan 6	XC6LX150-3	173	515	394	1	0	0	0	2225	ISE 14.7

Table 1b: Example Implementation Statistics with a 32 bit datapath – ECB, Encrypt Only, 128 bit key, 'Online' Hardware Key Expansion, 'Push Button' flow with Fmax specified by clock constraint

Family	Example Device	Fmax (MHz)	Slices[1]	IOB[2]	GCLK	BRAM	MULT/ DSP48	DCM	Throughput (MBit/sec)	Design Tools
Spartan 6	XC6LX150-3	212	423	106	1	0	0	0	678	ISE 14.7

Table 1c: Example Implementation Statistics with a 64 bit datapath – ECB, Encrypt Only, 64 bit key, 'Online' Hardware Key Expansion, 'Push Button' flow with Fmax specified by clock constraint

Family	Example Device	Fmax (MHz)	Slices[1]	IOB[2]	GCLK	BRAM	MULT/ DSP48	DCM	Throughput (MBit/sec)	Design Tools
Spartan 6	XC6LX150-3	190	609	202	1	0	0	0	1216	ISE 14.7

Table 1d: Example Implementation as Table 1a except with AES SBoxes mapped into Block RAM rather than LUTs.

Family	Example Device	Fmax (MHz)	Slices[1]	IOB[2]	GCLK	BRAM (RAMB 18)	MULT/ DSP48	DCM	Throughput (MBit/sec)	Design Tools
Spartan 6	XC6LX150-3	172	307	394	1	10	0	0	2206	ISE14.7

Notes:

1) Actual slice count dependent on percentage of unrelated logic – see Mapping Report File for details

2) Assuming all core I/Os and clocks are routed off-chip, which is not the intended usage. The core interface is designed to provide flexibility inside a larger FPGA design and is usually not directly connected to pins. Devices were chosen to meet I/O requirement, smaller devices could be used if ports were not directly connected to pins.

3) Area and performance numbers can vary significantly depending on synthesis and mapping options and software version.

Functional Description

The main functional blocks as shown in Figure 1 .

ECB Data Path

This block implements the ECB mode of the AES Algorithm. All other modes of AES are built on top on this basic encryption operation.

Key Schedule

This block calculates the round keys for each stage of the AES algorithm based on the key supplied by the user. A compilation option allows the user to omit this unit to save area in which case the core must be supplied with the complete keyschedule. This option may make sense if the key changes relatively infrequently and there is a microprocessor available elsewhere in the system to calculate the keyschedule.

AES Mode Logic

This block contains the feedback paths and additional logic required to implement the more complex modes of AES – CBC, OFB, CFB1, CFB8, CFB128 and CTR. Compilation parameters are supplied so only the logic for those modes which are required by the user will be instantiated.

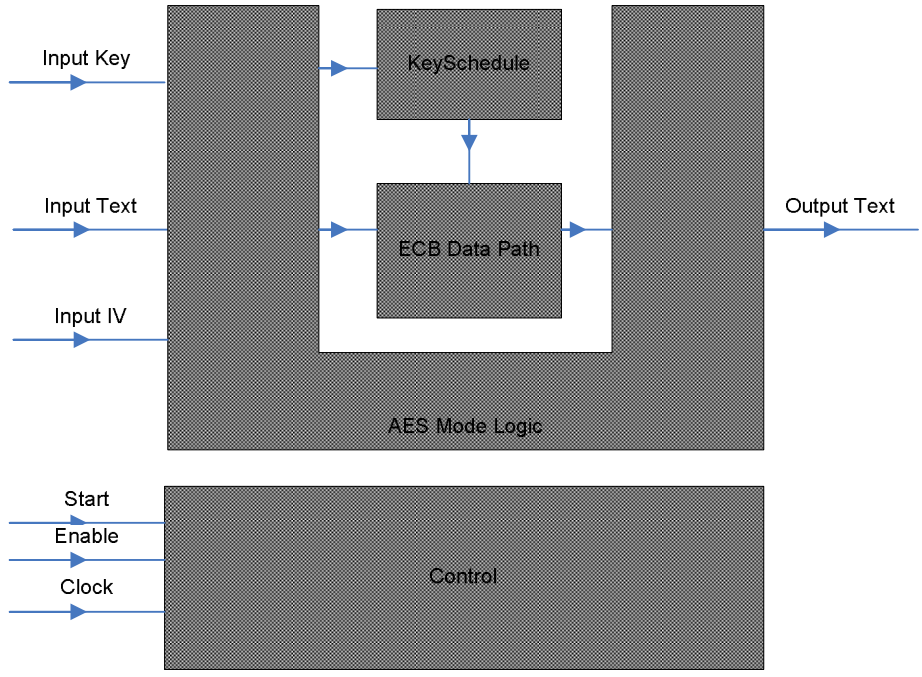

Figure 1, AES Core Block Diagram

Core Modifications

The core can be configured easily using a set of VHDL generic parameters. Normally, it is unnecessary for users to modify the design source code although the code is supplied and they are free to do so if they wish. Algotronix can also customise the core as a service for users with particular requirements which are not met by the standard product.

The following compilation options are specified by editing constant definitions in the aes_parameters_package file. This is the only file in the AES core release which will normally require to be edited by the user.

- **cipher_function** - specifies whether an Encryptor, Decryptor or Encryptor/Decryptor is required.
- **max_crypt_size** – specifies the maximum key length the core should implement. The user can select any key length up to and including this maximum during operation using the key_length signal. For example, if Max_Crypt_Size is aes256 then the core would deal with 256, 192 and 128 bit keys.
- **internal_data_path_width** – This parameter specifies the width of the datapath in the main encryption unit. The wider the datapath the more logic will be required and the fewer clock cycles required per encryption. Possible values are 8, 16, 32, 64 and 128 bits.
- **external_data_path_width** – This parameter specifies the width of the datapath in the unit that interfaces with the users design and implements the various cipher modes. This unit is only used during I/O operations which are a relatively small fraction of the total processing time. Specifying a lower data path width for this unit than the main encryption data path can save area without having a large effect on performance. Possible values are 8, 16, 32, 64 and 128 bits. The external data path width must be less than or equal to the internal data path width. If the internal data path width is less than 32 bits then the external data path width must equal the internal data path width.
- **implement_sboxes_in_ram** – specifies that FPGA RAM blocks rather than logic gates should be used to implement SBoxes and Inverse SBoxes. For modern Xilinx FPGAs with 64 bit LUT memories implementation using logic resources is area efficient and slightly faster than implementation in RAM blocks. The best approach is usually to set this parameter according to balance overall resource utilitsation (e.g. if the user design outside of the AES core requires many LUTs but does not use much block RAM then set to true to minimise the AES core's use of LUTs and take advantage of block RAM that would otherwise be unused).
- **keyschedule_shares_sboxes** – specifies that the SBoxes provided in the main encryption datapath should be used by the Keyschedule hardware rather than providing dedicated Sboxes in the hardware keyschedule unit. This means that the hardware keyschedule unit cannot operate at the same time as the main datapath so KC_ONLINE cannot be used. With KC_OFFLINE this option is normally preferred since it saves several RAM blocks.
- **omit_ecb_mode, omit_cbc_mode, omit_ofb_mode, omit_ctr_mode, omit cfb1_mode, omit_cfb8_mode, omit_cfb128_mode** - Used to request that logic to support cipher modes that will not be required is omitted from the design. The CTR and CFB modes in particular require quite large amounts of additional logic.
- **ctr_mode_counter_width** – Specifies the width of the counter used in CTR mode in bits. The counter is placed in the least significant bit positions. The counter and the non-counter bit positions are initialised from the initial value. The counter width **must** be wide enough to ensure that it will not wrap round and output the same value twice for the longest possible data stream between key changes. If the counter is too short CTR mode security will be compromised. Making the counter unnecessarily wide (e.g. selecting 128 bits) is wasteful of area and may result in a slower critical path and hence lower performance.

- **keyschedule_calculation** – one of the following options:
  - KC_OFFLINE – this is the most common option because it is particularly flexible and area efficient. The keyschedule is calculated each time the start signal is pulsed high and a key is loaded. The keyschedule is then stored in a memory block and used for each data block to be encrypted. Thus, there is a delay while the keyschedule is calculated before the first block in a chain of blocks can be encrypted but after that the encryption takes place at full speed. OFFLINE mode allows the SBoxes to be shared between the keyschedule unit and the encryption datapath which often reduces the overall number of RAM blocks required.
  - KC_ONLINE – the keyschedule is calculated 'online' as it is required. This option allows the encryption operation to start immediately after the key is loaded, avoids the need for a keyschedule buffer memory and can sometimes provide a higher operating clock frequency. ONLINE can only be used with data path widths of 32 bits or less because the keyschedule algorithm is specified to calculated a 32 bit round key each clock cycle and cannot generate 64 or 128 bit round keys directly – the memory buffer provided in OFFLINE mode is required to change the round key width. Also, the keyschedule algorithm generates the round keys in the order required for encryption – thus ONLINE can only be used with CFB, OFB and CTR modes which use the basic AES ECB core in encrypt for both encrypt and decrypt operations and for Encrypt only ECB and CBC designs.
  - KC_USER – this option allows the user to calculate the complete keyschedule with software and load the keyschedule directly into the buffer memory rather than supplying an encryption key and having the hardware keyschedule unit calculate the keyschedule. This option is useful to save area in systems with a microprocessor and where the key changes relatively infrequently.
  - KC_USER_OVERLAPPED – this option is the same as USER except that a dual-bank buffer for the keyschedule is provided so that the user can write a new keyschedule while the core continues to operate using the previous keyschedule. This option eliminates the latency associated with switching keyschedules at the expense of additional block RAM in the keyschedule unit.

- **force_output_low_until_valid** – When true the core will hold the output low at all times when valid output data is not present. When this signal is false the circuitry to hold the output zero will be omitted, saving some area. In this case the core output 'output_text' will show the values at intermediate rounds of the cipher as well as the final round. This data is not fully encrypted and, if available to an attacker, could compromise security of both the key and data. Therefore, this parameter should only be set to false if the user design which contains the core can guarantee that an attacker will not be able to monitor the core output directly.

Core I/O Signals

The core signal I/O have not been fixed to specific FPGA device pins to provide flexibility for interfacing with user logic, normally core I/Os will connect to signals within th euser design and the multiple wide busses will be multiplexed together at a higher level in the system before being taken to I/O pins. Descriptions of all I/O signals are provided in Table 2.

Signal	Signal Direction	Description
clock	input	Clock – active on rising edge
reset	input	Asynchronous reset – active high. Usually connected to FPGA global reset. For Xilinx it is recommended to use synthesis options to convert to a synchronous reset to allow a more efficient mapping to FPGA resources.
enable	input	Module clock enable – 0: module is inactive, 1: module runs
mode	input	Mode signal – specifies which mode of AES is to be implemented. See also the omit_* compilation options in the section below. If compilation options have specified that logic for a particular mode should be omitted then incorrect behaviour will result if that mode is selected.
key_length	input	Specifies the length of the key that is being used – 128, 192 or 256 bits. See also the max_crypt_size compilation option. Only keys up to the size specified in max_crypt_size may be specified e.g. if max_crypt_size generates hardware for a 192 bit key then KeyLength may be 128 or 192 bits but not 256 bits.
do_encrypt	input	Specifies whether the core should operate in Encrypt or Decrypt mode. This input is only significant if the compilation option cipher_function is set to EncryptDecrypt i.e. hardware for both encryption and decryption has been included.
start	input	Starts a new encryption operation or block of operations in the chained modes. The control signals mode, key_length and do_encrypt are sampled and the parameters fixed for the next operation. The key is assumed to have changed and the keyschedule is recalculated (or loaded if the compilation option User_Calculates_Keyschedule is active).
keyschedule_start	input	Used in KC_USER, KC_USER_OVERLAPPED and KC_OFFLINE_OVERLAPPED keyschedule calculation modes ONLY, this signal is ignored in other modes. Indicates the start of a new user keyschedule. Key_length, do_encrypt and mode signals are sampled when this signal is high. The address counter is cleared so the next word of keyschedule to be loaded is the first word of the keyschedule.
user_keyschedule_load_enable	input	Used in KC_USER, KC_USER_OVERLAPPED keyschedule calculation modes only. Enable signal to indicate that the core should load the next word of keyschedule data on the high going clock edge. This signal allows loading the keyschedule to be spread out over a long time period which is convenient when the calculation is being done by a microprocessor and there is a time delay between each word being available.
keyschedule_bank_select	input	Used in KC_USER_OVERLAPPED and KC_OFFLINE_OVERLAPPED keyschedule calculation modes only. Specifies which bank of keyschedule memory should be used by the datapath circuitry. This signal is sampled when start or clear is active. After reset the core will use bank 0. Inverting this signal before bringing start high will swap the 'new' keyschedule over with the 'active' keyschedule.
new_keyschedule_ready	output	Used in KC_OFFLINE_OVERLAPPED keyschedule calculation mode only. Indicates that the overlapped keyschedule calculation is complete and it is possible to switch to the new keyschedule.
advanced_load_text	output	High on the clock cycle immediately preceding load_text going active. This signal can be used for flow control in conjunction with the enable signal to the core to stop processing if the next block of text is not ready.
load_text	output	Load flag – high when input_text is being loaded.
load_key	output	Load flag – high when the key is being loaded in KC_ONLINE or KC_OFFLINE keyschedule calculation modes. Not used in KC_USER or KC_USER_OVERLAPPED

		mode where the user supplies the whole keyschedule rather than a key.
output_valid	output	Valid flag – high when output_text is valid.
advanced_output_valid	output	High on the AES round immediately preceding output_valid. The number of clock cycles per round depends on the internal data bus width – with a 128 bit databus this signal will go high for a single clock cycle, with an 8 bit databus it will be high for 16 clock cycles. This signal gives advanced warning that the core is about to input and output data and can be used for flow-control by external control circuitry which stops the core using the enable signal until the system is ready to provide new input data and accept output data.
input_text[width-1:0]	input	Data input: current word of the 128-bit plain text. The width of this bus is specified by the data_path_width generic parameter. Where the data path is less than 128 bits wide multiple words are required to transfer a 128 bit AES data block – e.g. with a 32 bit data path 4 words are transferred in successive clock cyles.
output_text[width-1:0]	output	Data output: current word of the 128-bit cipher text
initial_value[width-1:0]	input	Current word of the 128-bit initial value for the chained modes of operation (ECB mode does not use the initial_value).
input_key[internal_data_path_width - 1:0]	input	The width of this bus is the same as the main datapath. If the keyscedule calculation mode is KC_USER or KC_USER_OVERLAPPED the entire keyschedule is input through this bus.

Table 2: Core I/O Signals.

Verification Methods

Algotronix supplies a comprehensive VHDL testbench for the core which supports the standard AESAVS test suite with additional vectors from the SP800-38A publication to test the various AES modes. The testbench allows simulation of the design source code and also post place and route timing simulation. The testbench can be used in Regression mode to confirm the functionality of the core against known 'golden' test vectors provided by Algotronix or in Qualification mode to generate response files from vectors supplied by a NIST approved certification laboratory.

To provide immediate confidence that the core works correctly in hardware the 'Getting Started' Application note provided with the core supplies with VHDL code and design files to demonstrate the core running on a Xilinx Spartan 3 evaluation board. This low cost board is available directly from Xilinx.

This core is in production with multiple design-ins on several FPGA families.

Recommended Design Experience

It is recommended that the user is familiar with the VHDL language and with the Xilinx design flow. The core can also be instantiated inside a wrapper to allow use with a Verilog design flow.

Selection of the cipher mode of use of AES has implications for overall security, ease of use and performance and it is recommended that if the user is not a specialist in cryptography advice should be taken when selecting the appropriate mode for the application.

Available Support Products

Algotronix supplies two application notes and assoicated source code free of charge with the core:

Getting Started: This application note is intended to be the equivalent of 'Hello World' in C – it is a very simple wrapper providing the minimum logic required to instantiate the core on an evaluation board and carry out an encryption. The results are displayed on the seven segment LEDs on the Spartan evaluation board. The getting started design is also useful as an initial 'confidence test' when bringing up the core on a new circuit board designed by the user since it requires that the board supply only a clock, a reset signal and LEDs to display the result. This application note also provides example timing waveforms for use of the core.

Choosing Configuration Options for the G3 AES Core: This application note provides additional information on the various modes of the AES algorithm and how to select the appropriate one for your application. It also describes the various generic parameters in more detail and provides guidance on the best options.

Ordering Information

This product is available directly from Algotronix under the terms of the SignOnce IP License. Please contact Algotronix for pricing and additional information about this product using the contact information on the front page of this datasheet. To learn more about the SignOnce IP License program, contact Algotronix or visit the web:

Email: commonlicense@xilinx.com
URL: www.xilinx.com/ipcenter/signonce

Export Control

Strong encryption technology such as AES is the subject of international export regulations. Algotronix is located in the United Kingdom and export of this core is regulated by the UK government.

The core is freely available within the European Union and can be supplied immediately to the following countries: United States, Australia, New Zealand, Canada, Norway, Switzerland, Japan.

Export to other countries requires an export licence. The UK government's Department of Business, Enterprise and Regulatory Reform (previously known as Department of Trade and Industry) publishes information on their website (www.berr.gov.uk) which gives an indication of average export licence processing times for various countries and the percentage of licence requests which are granted. For many countries obtaining an export licence can be done relatively quickly and with only a small amount of additional paperwork.

It is the the responsibility of the customer to comply with all applicable requirements with respect to re-export of products containing the AES technology.

Related Information

Industry Information

The AES standard documents FIPS197, SP800-38A and AESAVS are available from the National Institute of Standards and Technology, Computer Security Resouce Center website (www.csrc.nist.gov).

Xilinx Programmable Logic

For information on Xilinx programmable logic or development system software, contact your local Xilinx sales office, or:

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
URL: www.xilinx.com

URL: www.algotronix.com

Version Control Information	
Subversion Revision Number	**50**
Date	**2014/06/10 11:41:34**
Document	Aes G3 Data Sheet, Xilinx Edition
Status (blank field indicates OK/no warnings)	
	(Table auto-updates, do not edit field values by hand)