AES CCM Core, Xilinx Edition

February 23, 2016 Product Specification

Core Facts

Provided with Core	
Documentation	User Manual
Design File Formats	VHDL source code
Verification	Test Bench, Test Vectors
Instantiation templates	VHDL
Simulation Tool Used	
ModelSim, Vivado Simulator	
Support	
Support provided by Algotronix	

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

Features

- Full Implementation of AES Counter with CBC MAC Mode (CCM) Proposal as specified in IEEE 802.11 and NIST SP800-38C
- Based on Algotronix' G3 AES core, can implement simple AES modes as well as CCM
- 32 Bit Internal data path width and option of one or two internal AES cores provides good efficiency on typical applications
- Compatible with Algotronix' interface to MicroBlaze or PowerPC processor
- Supplied as VHDL or Verilog source code to allow security review

Applications

- Wired, Optical and Wireless Networking
- WiFi, WiMax and ZigBee wireless networking standards
- Government/Military
- Satellite

General Description

The Algotronix AES-CCM Core implements the Counter with CBC MAC mode of operation of the AES algorithm. This mode of operation is described in NIST Special Publication SP800-38C and is used in many wireless networking standards.

AES-CCM is significantly more complex than the simple standard modes of AES such as Cipher Block Chaining (CBC) specified in NIST special publication SP800-38A. Unlike simple modes of the cipher which provide only confidentiality, CCM provides both confidentiality and authentication. Authentication is the ability to detect tampering with the encrypted message as it passes between the sender and receiver and in most applications, particularly wireless applications, is essential for security. CCM mode also provides a standard method for processing data streams whose length is not a multiple of the 128 bit AES block size.

The AES-GCM mode, for which Algotronix can also supply an IP core, provides confidentiality and authentication like AES-CCM. Both CCM and GCM use the counter (CTR) mode of AES for

encryption to provide confidentiality but AES-CCM uses AES in CBC mode for authentication where AES-GCM uses a Galois field multiplier. Since AES-CCM uses AES for both confidentiality and authentication circuitry can be shared between these functions resulting in lower overall area than with GCM. However, the AES-CBC mode used for authentication is not parallelisable which limits the throughput of AES-CCM relative to AES-GCM. Another important difference is that AES-GCM can operate 'online' on a stream of incoming data as it arrives whereas AES-CCM requires that the length of the data stream is known in advance and therefore usually works on stored data.. In general, AES-CCM can provide better area efficiency at lower data rates than AES-GCM but is somewhat less flexible. AES-GCM is more flexible and can provide a higher maximum throughput through parallelisation.

The Algotronix CCM core is an 'add-on' to the G3 AES core and uses the G3 core to implement the AES part of the AES-CCM algorithm with all standard key sizes (128, 192 or 256 bits). The CCM code can be configured to use two separate AES-G3 encryptors for the confidentiality and authentication functions or to use a single encryptor shared between both functions. The CCM core implements the complete set of functionality from the NIST SP800-38C specification including Additional Authenticated Data (AAD). AAD is passed through the message authentication function so that any unauthorised changes to the authenticated data can be detected but is not encrypted. The Algotronix CCM core can be configured to minimise the latency associated with changing keys at the beginning of new CCM packets making its throughput less sensitive to packet size.

The core is supplied with a testbench which reads vectors in the file format specified in the NIST CCM Verification System (CCMVS) document and used by NIST approved test laboratories for qualification. Files containing standard known answer tests from the NIST documents are supplied, as is a much larger file of 10,000 packets generated from a software implementation of AES-CCM. This additional vector file is much more comprehensive than the NIST supplied vectors. The software program itself is also available so customers can create their own test vector files as required. The CCM testbench operates in regression mode to verify any changes you may make to the source code or can be used to process vector files supplied by a NIST approved test lab for validation purposes.

The Algotronix AES-CCM 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 also be supplied in Verilog on request. The core can be configured as Encryptor, Decryptor or Encryptor/Decryptor and the maximum key length can also be selected. The core provides hardware key schedule generation.

The AES-G3 core within the AES-CCM core can optionally be used to implement standard AES modes (ECB, CBC, OFB, CTR, CFB).

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 AES-CCM 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 modern FPGAs and makes full use of FPGA specific features such as dual port memory blocks.

Contact Algotronix to obtain performance and area statistics for the core in the configuration of interest.

Functional Description

The main functional blocks as shown in Figure 1 .

AES-G3

This block implements the ECB mode of the AES Algorithm for use by CCM. Although CCM is described as using AES in counter (CTR) mode it requires specific capabilities from the counter implementation which are not available in standard AES counter mode therefore it is built on the G3 implementation of ECB with its own customised implementation of the counter rather than using the G3 implementation of CTR mode. The CCM core can be configured to provide access to the internal AES-G3 core to provide standard AES modes as well as CCM.

CCM Mode Logic

This block contains the datapaths required to route data between the various computational units within the core under the control of the CCM-Control block.

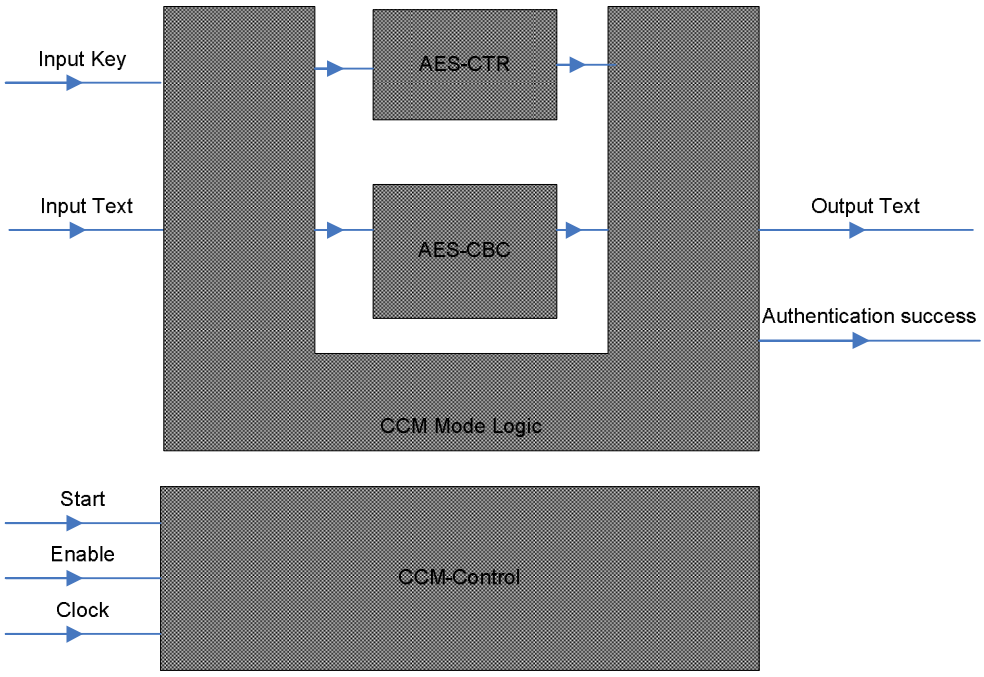

Figure 1, AES-CCM Core Block Diagram

Compilation Options

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.

- **cipher_function** - specifies whether an Encryptor, Decryptor or Encryptor/Decryptor is required. In the case of CCM there is little area advantage to specifying Encryptor or Decryptor rather than the more general Encryptor/Decryptor.
- **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 256 bits then the core would deal with 256, 192 and 128 bit keys. Longer key sizes require additional processing rounds in the AES algorithm and may increase overall processing time. There is also a small area advantage to specifying a shorter maximum key length.
- **data_path_width** – Width of the data path within the AES-CCM core and the external buses supplying input and output data to the core. The first release of the core supports 32 bit data_path_width only but this will be extended over time to support 128, 64, 32, 16 and 8 bit widths. The AES-G3 core within AES-CCM already has this capability. Wider data path widths require significantly more area but offer a near linear reduction in the number of clock cycles required to process data i.e. with a 128 bit data path the core will require about half the number of clock cycles to process a particular CCM block it would with 64 bit datapath. Maximum clock frequency may be lower in designs with wide data paths because wire lengths are increased.
- **ctr_mode_counter_width** – Length of the counter used when the AES-G3 core within AES-CCM is required to implement the simple CTR mode (configuration parameter omit_ctr_mode = false). This parameter does not affect the counter used in the AES-CCM encryptor which also uses CTR mode: the implementation of this counter is as specified in the CCM standard.
- **force_output_low_until_valid** – When true the simple mode AES core will hold the output_text bus 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. The circuits which provide this function are on the critical timing path and require some additional area so in designs which stress area or performance and in which it is certain there is no security problem with partially encrypted values appearing on the core output this parameter should be set to false.
- **use_single_aes_unit** – When true the same AES unit is used for the AES-CBC and AES-CTR functions. This saves area at the expense of throughput.
- **omit_mode** – This array specifies which of the AES simple modes should be implemented in addition to AES-CCM. If all the values are set to true the core will only provide AES-CCM functionality, this will allow multiplexing logic to be omitted and result in area savings and in some cases higher performance. As well as the additional multiplexing required within AES-CCM some of the basic AES modes are quite resource intensive so only those modes which are actually required should be selected. When all the simple modes are omitted the aes_mode input signal to the core is not required and can be tied to zero.
- **target_device** – In the Xilinx edition of the CCM core only Xilinx FPGAs are supported and target device must be set to XILINX for older FPGA families with 4 input LUTs or XILINX_VIRTEX_5 for new FPGA families with 6 input LUTs.

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. Descriptions of all signal I/O are provided in Table 2.

Signal	Signal Direction	Description
clock	input	Clock – active on rising edge
reset	input	Reset – active high. A global constant USE_ASYNCHRONOUS_RESET is provided to select between synchronous and asynchronous reset, synchronous reset is recommended for Xilinx FPGAs and may result in reduced area.
enable	input	Module clock enable – 0: module is inactive, 1: module runs. This signal can be used for flow control to pause the core when the system is not ready to supply or to accept data.
use_ccm_mode	input	When '1' the core implements the CCM mode, otherwise it implements the simple AES mode specified aes_core_simple_mode. This signal is only significant if the core is configured to provide simple AES modes as well as CCM mode.
aes_core_simple_mode	input	Connected to the mode input on the AES G3 core inside the AES_CCM core when use_ccm_mode = '0'. Specifies the simple mode to be implemented – logic for the selected mode must also have been configured into the core by setting the corresponding omit_mode parameter to false.
Initial_value	input	Initial Value input for simple AES modes. Can be tied to zero if not required.
key_length	input	Specifies the length of the key that is being used – 128, 192 or 256 bits. See also the key_datapath_size compilation option. Only keys up to the size specified in key_datapath_size may be selected e.g. if key_datapath_size generates hardware for a 192 bit key then KeyLength may be 128 or 192 bits but not 256 bits.
nonce_length	input	Number of bytes in the nonce. The CCM algorithm requires that the lengths of the various fields are known in advance before processing starts, this is usually achieved by buffering packets in memory prior to processing
payload_length	input	Number of bytes in the payload – plaintext in the case of encryption and ciphertext plus MAC in the case of decryption
associated_data_length	input	Number of bytes in the associated authenticated data. This data is authenticated but is transferred as plaintext. The associated_authenticated_data is input to the core through input_text to allow the MAC to be calculated but is not transferred through to the output_text port. In most cases the core is processing data buffered in memory so it is not useful to supply unchanged data on the output to write back into RAM. If required the core can be customized to copy AAD data through to output_text for a 'flow through' application.
mac_length	input	Number of bytes in the MAC at the end of the message. The core will calculate a 128 bit MAC internally but not all of the 16 bytes need be appended to the message. Note that the likelihood of two different messages having the same MAC code decreases with MAC length and therefore short MACs provide weaker authentication.
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 block of encryption operations. The key and the control signals mode, key_length and do_encrypt must be stable before asserting start. The key is assumed to have changed and the key schedule will be recalculated, if the AES unit is operating in KC_OFFLINE mode this will involve a delay of the same time required to encrypt a block of data.
input_text [data_path_width-1:0]	input	Data input. Width is set by the data_path_width generic parameter. With narrower bus widths multiple clock cycles are required to transfer a 128 bit block of text. The most significant word is transferred first. This input is used to provide the additional authenticated data and the payload. In the case of decryption the payload includes the message authentication code (MAC) as well as the Ciphertext. Note that because of the appended MAC the input data stream is longer than the output data stream for encryption and the output data

		stream is shorter than the input data stream for decryption.
input_nonce	input	Data input to supply the nonce. The nonce must be unique for every packet encrypted with the same key and is often created using an external counter . It is convenient to provide a separate input for the nonce which can be connected directly to a counter rather than multiplexing this data onto the input_text bus.
load_text	output	Load flag – high when input_text is being loaded. The user may bring enable low during the clock cycle when load_text = 1 if the external system is not ready to provide more input text.
load_key	output	Load flag – high when the key is being loaded.
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. This signal gives advanced warning that the core is about to input and output data and can be used for flow-control. External circuitry can stop the core using the enable signal until the system is ready to provide new input data and accept output data. Advanced_output_valid is only generated when the generic parameter generate_advanced_output_valid is true – see the comments on this generic parameter.
output_text [data_path_width-1:0]	output	Data output: in the current version of the core this bus is 128 bits wide and the 128 bit block of text is transmitted in a single clock cycle.
input_key [data_path_ width-1:0]	input	Bus to input the key for the AES encryptor. It may take multiple clock cycles to load the 256, 192 or 128 bit key over the bus. The most significant data word is transferred first. In the case of a 128 bit input_key bus and 192 bit key only the most significant 64 bits on the input_key bus are used on the second clock cycle. This bus is the same width as the input_text bus and may be connected directly to input_text if the key and text data come from the same source since the two busses never transfer data in the same clock cycle.
authentication_success	output	On decrypt indicates whether the hash function has successfully authenticated the data stream.
authentication_valid	output	High when the authentication_success signal is valid in a decryption operation.

Table 2: Core I/O Signals.

Description of Operation

Processing a packet of CCM data is initiated by pulsing the start signal high for one clock cycle. The control signals which set the operating mode of the core for this packet must be valid at this time: do_encrypt, key_length, use_ccm_mode, aes_core_simple_mode, nonce_length, associated_data_length, payload_length and mac_length.

In this description the sequence of activity is described but the number of clock cycles between phases of activity is generally not documented. The user circuit should synchronise to the CCM core using the load_text, load_ke, output_valid and authentication_valid signals rather than by assuming a set number of clock cycles between various operations. Future updates to the core may have slightly different delays between phases as a consequence of changes to improve latency and/or throughput.

Simple AES Mode

If the core has been instructed to implement a simple AES operation (use_ccm_mode = 0, and at least one of the compilation generic parameters in the omit_mode array which control inclusion of the simple AES modes is false) then the core will carry out the following sequence of operations:

1. load_key will be brought high and the 128, 192 or 256 bit key value will be loaded over the input_key bus. This may require multiple clock cycles depending on the bus width, most significant words will be transferred first.

2. load_text will be brought high and the first input text and Initial Value (IV) for the simple AES operation will be loaded over the input_text and initial_value buses. Again, this may take multiple clock cycles.

3. The AES core will carry out the encryption operation. This will take a variable number of clock cycles depending on the key length and data_path_width.

4. The AES core will bring load_text and output_valid high and will simultaneously output the newly processed 128 bit block of plaintext/ciphertext using the output_text bus and input the next text block for processing using the input_text bus.

5. The core will process the new data and loop back to step 4 to output the results and input the next block.

In simple AES mode the CCM core is providing direct access to the G3 AES core within it and the timing, apart from the loading of IV over the input_text bus before the operation starts is the same as for G3 AES. Signals specific to CCM mode such as authentication_success are not valid. Instead, of processing a fixed length packet as in CCM mode in simple modes the core will continue to process a stream of data with the same key until the start signal is brought high again. As with AES G3, flow control is achieved by setting enable = 0 when the core indicates using the load_text and output_valid signals that it wishes to carry out a data transfer and the external circuitry is not ready.

CCM Operation

This section describes operation of the core when implementing the AES-CCM mode. In this configuration there is a single or dual internal AES-G3 encryption core. In the present version of the AES-CCM core data_path_width is fixed at 32 bits but future versions will support 8, 16, 32, 64 or 128 bits – the internal AES-G3 core already has this capability.

As with simple mode, operation is triggered by the start signal being brought high for one clock cycle. The core then carries out the following sequence of operations.

1. load_key is brought high and the AES key is loaded. This may take multiple clock cycles depending on data_path_width and key length.
2. The core starts to create the header blocks for the CBC encryption and the initial counter value for the CTR mode encryptor. This is based on the nonce and the nonce_length, additional_data_length, palyoad_length and mac_length inputs to the core. The requirement that the length values are known in advance before the AES-CCM operation starts follows from the fact that they are used to generate the first few input blocks in the AES-CBC encryption. AES-CBC is a feedback based non-reversable cipher in which the output from each step depends on all previous steps so the encryption of this header information must take place before the first data encryption.
3. The load_nonce signal is brought high and the first word of the nonce is loaded. There must always be nonce data in a valid CCM packet. Further words of nonce data are loaded in subsequent cycles until the entire nonce – as determined by the nonce_length input is loaded.

The final word of nonce is filled starting from the most significant bit and may contain unused bytes if the nonce does not divide evenly into words e.g. a five byte nonce would not divide evenly into 4 byte (32 bit) words on the input_nonce port.

4. The core now moves on to process the additional authenticated data (AAD) if present (additional_data_length = 0 is allowed). Blocks of AAD text are loaded and processed by the AES-CBC unit until all additional_data_length bytes of AAD are processed.
5. The core now processes the plaintext/ciphertext 'payload' component of the packet. A zero length payload is possible in which case the core moves straight on to process the MAC. There will always be a MAC, mac_length = 0 is not allowed. On encrypt the MAC is appended to the ciphertext to form the complete payload on the core output. On decrypt the MAC supplied with the payload input to the core is compared with the value calculated internally to determine the authentication_success signal.
6. In a decrypt operation only, the authentication_valid signal will go high when the authentication_success status is being output on the authentication_success signal.
7. The core now enters a passive state waiting for the start signal which initiates processing the next packet.

Verification Methods

Verification of the AES-G3 core within AES-CCM is through a comprehensive VHDL testbench for the G3 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. The AES testbench is supplied along with the CCM product so that customers can check the AES implementation thoroughly.

In the CCM product a second CCM specific testbench is provided which tests the CCM specific functionality using the vectors provided in the NIST SP800-38C document which defines the AES-CCM algorithm and the NIST CCM Verification System document which decribes the standardised test vector format used by validation laboratories. Algotronix also supplies a Windows application which can create test vector files for AES-CCM this application is based on the well known open-source implementation of AES by Brian Gladman. This program can generate CCM test vectors with random data and random numbers of bytes in the various fields in order to thoroughly test the core control logic. This testbench supplied with the core includes a file with several thousand random vectors generated by this program. Each of these vectors involves many individual AES encryptions because AES-CCM operates on complete packets of data and requires two AES encryptions for each block of payload.

Recommended Design Experience

It is recommended that the user is familiar with the VHDL or Verilog language and with the Xilinx design flow and simulation tools. The coding standards used when creating the product allow automatic translation of the core into Verilog without loss of readability and the core can be supplied in Verilog on request.

It is recommended that the user has a background in data security or takes appropriate advice when considering how to implement AES-CCM in a larger system.

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 in addition 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 Department of Business, Enterprise and Regulatory Reform 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 and the AES-CCM standard documents SP800-38C and CCMVS are available from the National Institute of Standards and Technology, Computer Security Resouce Center website (www.csrc.nist.gov). The IETF document RFC3610 specifies their version of the AES-CCM algorithm and is available for download from www.ietf.org. The IEEE version of AES-CCM used in the wireless networking is specified in the 802.11i standard.

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	**78**
Date	2016/02/23 09:44:54
Document	Aes Ccm Data Sheet, Xilinx Edition
Status (blank field indicates OK/no warnings)	
	(Table auto-updates, do not edit field values by hand)