CoreAES128 v3.3

Handbook

Microsemi®

Table of Contents

Introduction

Core Overview

The CoreAES128 macro implements the advanced encryption standard (AES), which provides a means of securing data. AES utilizes the Rijndael algorithm, which is described in detail in the Federal Information Processing Standards Publication (FIPS PUB) 197. The AES Rijndael algorithm (Figure 1) takes as inputs 128 bits of plaintext data and 128 bits of a cipher key. After several rounds of computation, the algorithm produces a 128-bit ciphered version of the original plaintext data as output[1]. During the rounds of the algorithm, data bits are subjected to byte substitution, data shift operations, data mixing operations, and addition (XOR) operations with an expanded version of the original
128-bit cipher key.

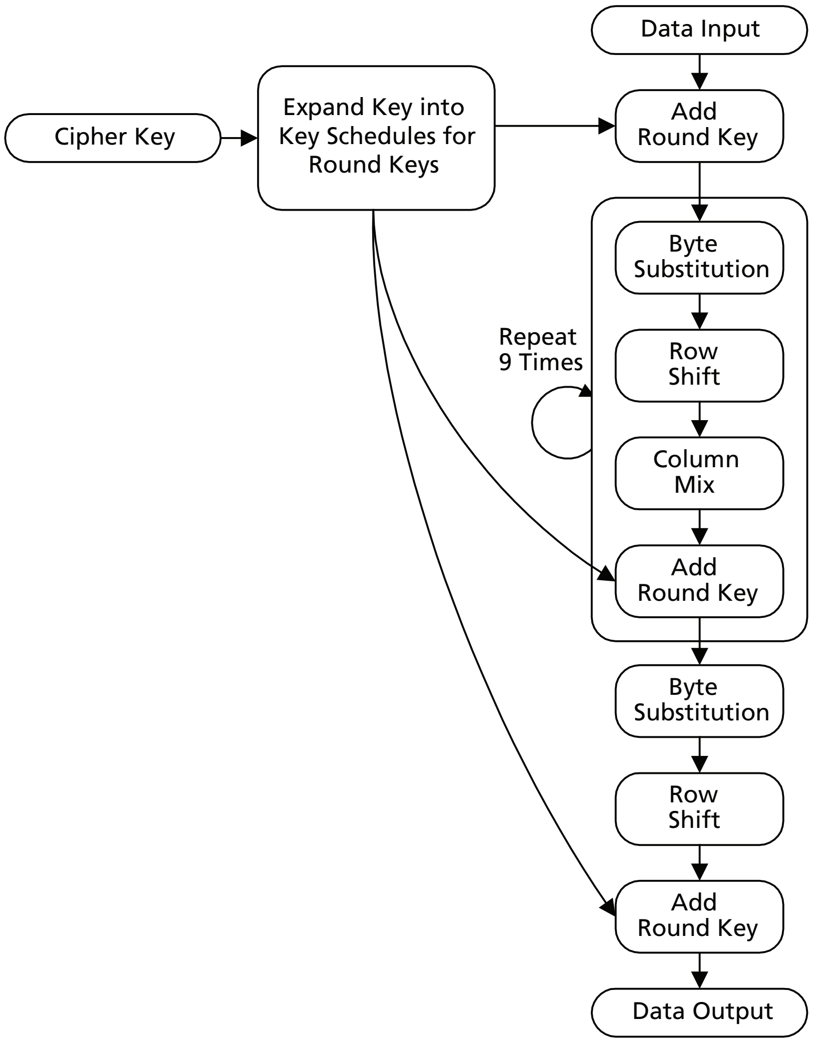


Figure 1 · AES Algorithm (128-bit cipher key)

1. *FIPS PUB 197 allows for key sizes of 128, 192, and 256 bits. However, this implementation supports a cipher key size of 128 bits only.*

Design Security

Figure 2 shows a typical system diagram. The cipher key, which is the “secret” key, can be made up of FPGA logic cells, preventing the possibility of design or data theft. Microsemi flash-based ProASIC$^{\text{PLUS}}$® devices employ FlashLock® technology, and Microsemi antifuse-based Axcelerator® devices employ FuseLock$^{\text{TM}}$ technology; each of which keeps the cipher key and the rest of the logic secure. The output of the CoreAES128 macro should be connected to registers or FIFOs, as it is only valid for one clock cycle. Refer to the example used in the “Encryption” section on page 10 and “Decryption” section on page 11.

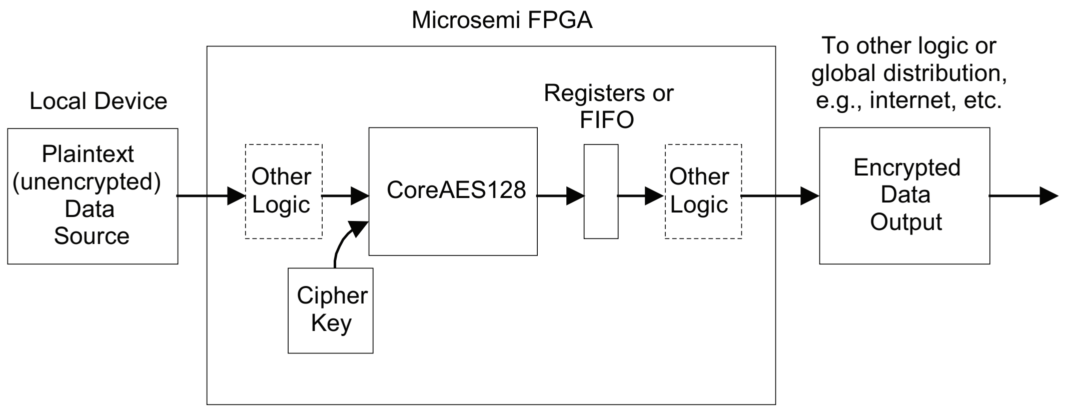


Figure 2 · Typical CoreAES128 System

Key Features

- Compliant with FIPS PUB 197
- ECB implementation per NIST SP 800-38A
- Example source code provided for cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), and counter (CTR) modes
- 128-bit cipher key
- Encryption and decryption possible with the same core
- 44-Clock cycle operation to encrypt or decrypt 128 bits of data
- Pause/resume functionality to continue encryption or decryption at will
- Provides redundant security

Core Version

This handbook supports CoreAES128 version 3.3.

Supported Tool Flows

CoreAES128 requires SmartDesign and Microsemi Libero® Integrated Design Environment (IDE) v8.4 or Libero SoC System-on-Chip (SoC) v10.

Supported Families

- IGLOO®
- IGLOOe
- ProASIC®3
- ProASIC3E
- ProASIC3L

- Fusion
- ProASICPLUS
- Axcelerator
- RTAX-S
- SmartFusion®
- SmartFusion®2
- IGLOO®2
- RTG4™

Device Utilization and Performance

Table 1 provides a summary of the implementation data for CoreAES128.

Table 1 • **CoreAES128 Device Utilization and Performance**

Family	Cells or Tiles			RAM Blocks	Utilization		Performance	Throughput
	Sequential	Combinatorial	Total		Device	%		
IGLOO	366	3,683	4,049	8	AGL1000V5	17	58 MHz	169 Mbps
IGLOOe	366	3,683	4,049	8	AGLE3000V5	6	57 MHz	166 Mbps
ProASIC3	366	3,683	4,049	8	A3P1000	17	88 MHz	256 Mbps
ProASIC3E	366	3,683	4,049	8	A3PE3000	6	84 MHz	245 Mbps
ProASIC3L	366	3,683	4,049	8	A3PE3000L	6	70 MHz	204 Mbps
Fusion	366	3,683	4,049	8	AFS600	38	75 MHz	218 Mbps
ProASICPLUS	427	4,507	4,934	24	APA1000	9	36 MHz	104 Mbps
Axcelerator	380	2,222	2,602	10	AX1000	15	90 MHz	262 Mbps
RTAX-S	380	2,222	2,602	10	RTAX1000S	15	58 MHz	169 Mbps
SmartFusion	380	3,614	3,994	8	A2F500M3G	35	91 MHz	265 Mbps
SmartFusion2	632	2,681	3,313	8	M2S150T	3	171 MHz	498 Mbps
IGLOO2	632	2,696	3,328	8	M2GL150T	3	169 MHz	492 Mbps
RTG4	1,020	3,209	4,229	8	RT4G150	3	113 MHz	328 Mbps

Note: Data in this table were achieved using typical synthesis and layout settings.

Data throughput is computed by taking the bit width of the data (128 bits), dividing it by the number of cycles (44), and multiplying it by the clock rate (performance); the result is listed in Mbps.

1 – Design Description

I/O Signals

The port signals for the CoreAES128 macro are illustrated in Figure 1-1 and described in Table 1-1. All signals are either “Input” (input only) or “Output” (output only).

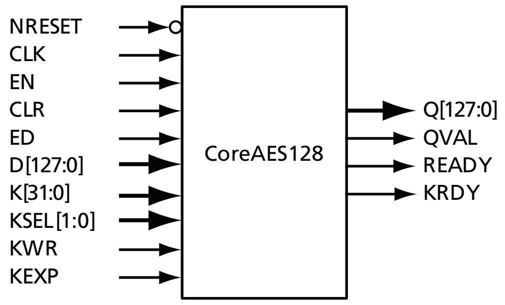


Figure 1-1 · CoreAES128 I/O Signal Diagram

***Table 1-1* • CoreAES128 I/O Signals**

Name	Type	Description
NRESET	Input	Active-low asynchronous reset
CLK	Input	System clock. Reference clock for all internal logic
EN	Input	Enable signal. Set to '1' for normal continuous encrypt/decrypt operation, set to '0' to pause.
CLR	Input	Synchronous clear signal. Set to '1' to clear logic at any time.
ED	Input	Encryption/decryption. '1' to encrypt, '0' to decrypt
D[127:0]	Input	Data in. 128-bit data input bus
K[31:0]	Input	Key. 32-bit cipher key input bus
KSEL[1:0]	Input	Key select. Selection bits to direct K[31:0] to one of the four 32-bit words comprising the internal 128-bit cipher key.
KWR	Input	Key write. Set to '1' to write K[31:0] to one of the four 32-bit words comprising the internal 128-bit cipher key.
KEXP	Input	Key expand. Set to '1' to expand the 128-bit internal key.
Q[127:0]	Output	Data out. 128-bit cipher text (encryption operation)/plaintext (decryption operation) output bus
QVAL	Output	Q Valid. '1' indicates that valid encryption/decryption data is available on Q[127:0].
READY	Output	Ready. '1' indicates that CoreAES128 has finished its initialization sequence 1,024 clock cycles after the rising edge of NRESET.
KRDY	Output	Key ready. '1' indicates that the internal 128-bit cipher key was expanded and the macro is ready for encryption/decryption.

Parameters/Generics

CoreAES128 has parameters (Verilog) and generics (VHDL) for configuring the RTL code (Table 1-2). All parameters and generics are integer types and are mapped to configuration options in the SmartDesign configuration window.

***Table 1-2* • CoreAES128 Configuration Parameters**

Parameter	Values	Description
FAMILY	11 to 25	Must be set to match the supported FPGA family: 11 – Axcelerator 12 – RTAXS 14 – ProASIC$^{\underline{PLUS}}$ 15 – ProASIC3 16 –ProASIC3E 17 – Fusion 18 – SmartFusion 19 – SmartFusion2 20 – IGLOO 21 – IGLOOe 22 – ProASIC3L 24 – IGLOO2 25 – RTG4
CFG_MODE	1 to 3	Control encryption/decryption functionality: 1 – Enables encryption functionality only 2 – Enables decryption functionality only 3 – Encryption/decryption according to ED bit input

2 – Functional Block Description

CoreAES128 encrypts the input data by rounds of computations that produce the ciphered version of input data. Figure 1 shows the block diagram for CoreAES128.

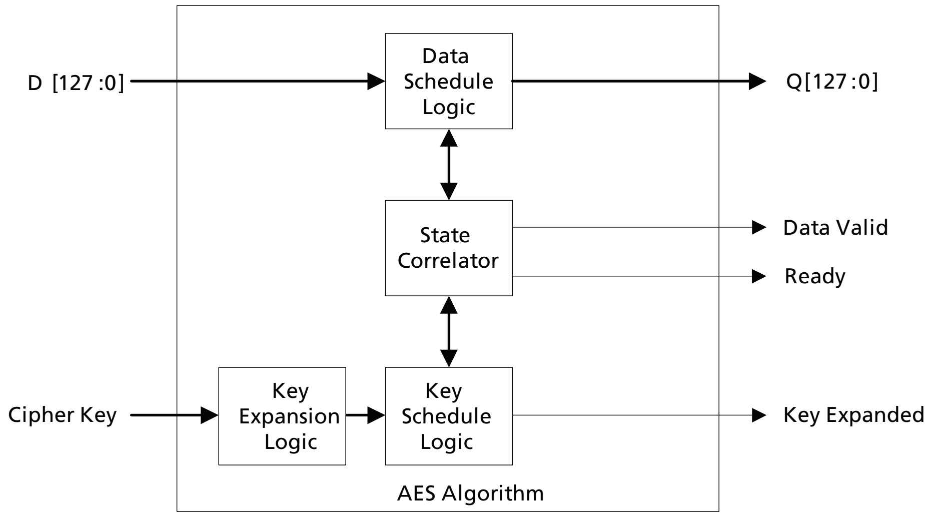


Figure 1 · CoreAES128 Algorithm Block Diagram

Data Schedule Logic

The data schedule logic block of CoreAES128 computes the intermediate data values at each round of the AES algorithm.

State Correlator Logic

The state correlator logic block for CoreAES128 maintains coherency between data and key schedule logic.

Key Schedule Logic

The key schedule logic block of CoreAES128 controls the intermediate key schedule at each round of the AES algorithm.

Key Expansion Logic

The key expansion logic block of CoreAES128 expands the original 128-bit key for use in encryption or decryption operations.

3 – CoreAES128 Protocol Overview

CoreAES128 Initialization

After a reset condition, shown in Figure 1, the CoreAES128 macro performs a self-initialization process. This initialization process takes 1,024 clock cycles, then the READY signal becomes active at logic '1'. Once READY is active, the CoreAES128 macro is ready for cipher key expansion, followed by encryption or decryption operations.

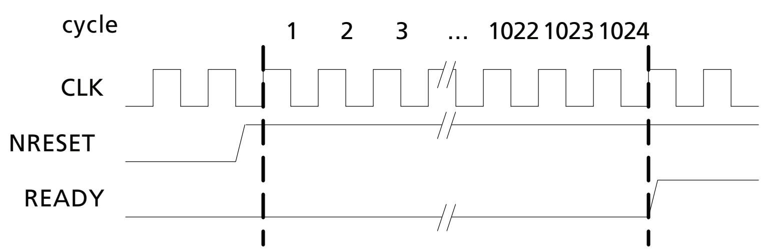


Figure 1 · CoreAES128 Initialization

CoreAES128 Operation

As shown on the left side of Figure 1 on page 3, the AES algorithm requires an expanded version of the original cipher key for use in encrypting or decrypting data. Upon a power-up condition, the cipher key and the expanded version of the cipher key are undefined. Therefore, they must be set up after the initialization process, described in the “CoreAES128 Initialization” section, and before encryption or decryption operations can take place. The procedures for writing and expanding the cipher key, described in the “Cipher Key Expansion” section, must be repeated any time a new 128-bit cipher key is required, such as after a reset or power-up condition.

Note: If the same cipher key is to be used for all encryption and decryption operations, the procedures for writing and expanding the cipher key only need to be performed once.

Cipher Key Expansion

Prior to any encryption or decryption operation, the 128-bit cipher key must be written to CoreAES128 and expanded (Figure 2 on page 10). Refer to FIPS PUB 197 for the algorithmic details of the key expansion process.

Follow the steps below to write the four 32-bit words that make up the 128-bit cipher key, and to expand the 128-bit cipher key:

1. Set EN to logic '0'.
2. Set KSEL[1:0] to '00' to select the lowest 32 bits (LSB word) of the internal 128-bit cipher key.
3. Set K[31:0] to the value of the lowest 32-bit word of the desired 128-bit cipher key.
4. Set KWR to logic '1' for one clock cycle.
5. Set KSEL[1:0] to '01' to select the second lowest 32 bits of the internal 128-bit cipher key.
6. Set K[31:0] to the value of the second lowest 32-bit word of the desired 128-bit cipher key.
7. Set KWR to logic '1' for one clock cycle.
8. Set KSEL[1:0] to '10' to select the second highest 32 bits of the internal 128-bit cipher key.
9. Set K[31:0] to the value of the second highest 32-bit word of the desired 128-bit cipher key.
10. Set KWR to logic '1' for one clock cycle.
11. Set KSEL[1:0] to '11' to select the highest 32 bits (MSB word) of the internal 128-bit cipher key.

12. Set K[31:0] to the value of the highest 32-bit word of the desired 128-bit cipher key.
13. Set KWR to logic '1' for one clock cycle.
14. Set KWR back to logic '0'.
15. Set KEXP to logic '1' for one clock cycle.
16. Set KEXP back to logic '0'.
17. Wait for 52 clock cycles.

The four 32-bit words which comprise the 128-bit cipher key can be written in any order. It is not necessary to write them in sequential order; i.e., lowest 32-bit word to highest 32-bit word.

If the KRDY signal was active at logic '1' prior to setting the KWR signal to logic '1' (from a previously expanded cipher key), it becomes inactive on the next rising clock edge after performing "Set KWR to logic '1' for one clock cycle." After 52 clock cycles, the KRDY signal becomes active; (logic '1') to indicate that the 128-bit cipher key was expanded internally. The CoreAES128 macro is now ready for encryption or decryption operations. The KRDY signal initializes to the inactive state of logic '0' after a reset condition (Figure 2), prior to the key expansion process.

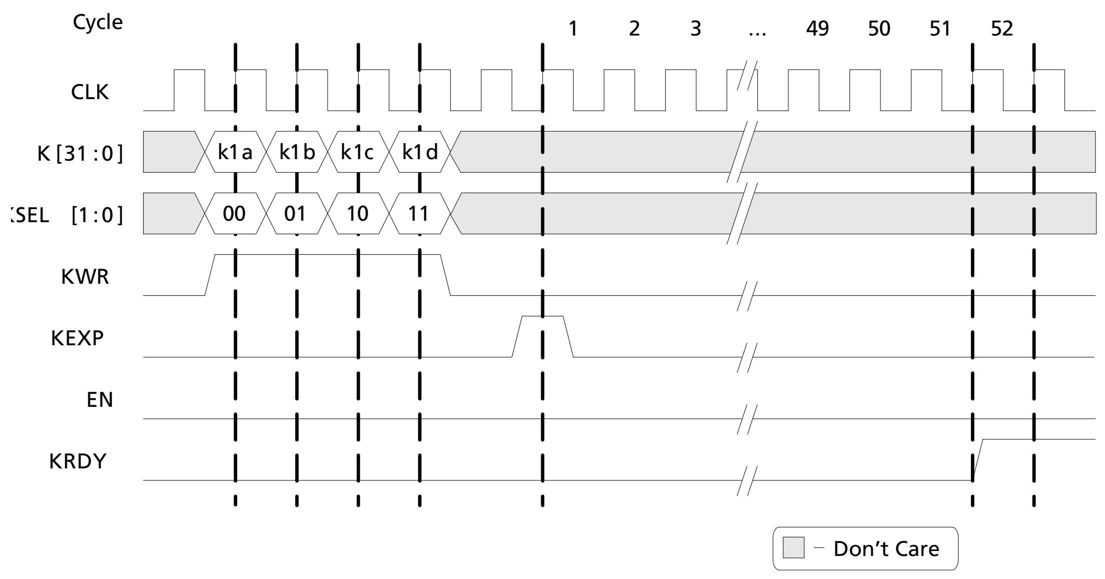


Figure 2 · Cipher Key Write and Expand

Encryption

Follow the steps below to begin the encryption data process (Figure 3 on page 11):

1. Write and expand the 128-bit cipher key, if not already done (refer to "Cipher Key Expansion" on page 9.)
2. Set D[127:0] to the plaintext data to be encrypted (d1 in Figure 3).
3. Set ED to logic '1'.

   Note: This signal is only available when parameter/generic CFG_MODE = 3. This step can be skipped if CFG_MODE is set to 1 or 2.
4. Set EN to logic '1'.
5. Wait for 44 clock cycles.

After 44 clock cycles of the EN input being held continuously at logic '1', the QVAL signal will transition from logic '0' to logic '1' and remain valid for one clock cycle. This indicates that valid ciphered (encrypted) data (q1 in Figure 3) is available on the Q[127:0] outputs. The encrypted data is only available during clock cycle 44, so you must register or latch the data on Q[127:0], using the QVAL signal as a qualifying register enable or latch enable. As shown in Figure 3, continuous encryption is possible. For example, the second 128-bit plaintext data word (d2 in Figure 3) can be immediately encrypted by setting the D[127:0] input to d2 on the rising clock edge of clock cycle 45.

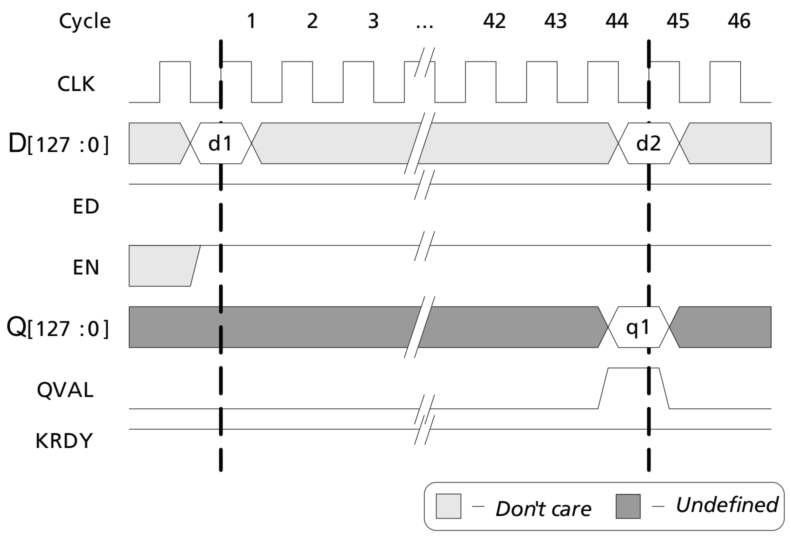


Figure 3 · Example Encryption Sequence

Decryption

Follow the steps below to begin the decryption data process (Figure 4 on page 12):

1. Write and expand the 128-bit cipher key if not already done (refer to "Cipher Key Expansion" on page 9.)
2. Set D[127:0] to the ciphertext data to be decrypted (d1 in Figure 4).
3. Set ED to logic '0'.

   Note: This signal is only available when parameter/generic CFG_MODE = 3. This step can be skipped if CFG_MODE is set to 1 or 2.
4. Set EN to logic '1'.
5. Wait for 44 clock cycles.

After 44 clock cycles of the EN input being held continuously at logic '1', the QVAL signal will transition from logic '0' to logic '1' and remain valid for one clock cycle, indicating that valid plaintext (unencrypted data, shown as q1 in Figure 4) is available on the Q[127:0] outputs. The decrypted plaintext data is only available during clock cycle 44, so you must register or latch the data on Q[127:0] using the QVAL signal as a qualifying register enable or latch enable.

As shown in Figure 4, continuous decryption is possible. For example, the second 128-bit cipher text data word (d2 in Figure 4) can be immediately decrypted by setting the D[127:0] inputs to d2 on the rising clock edge of clock cycle 45.

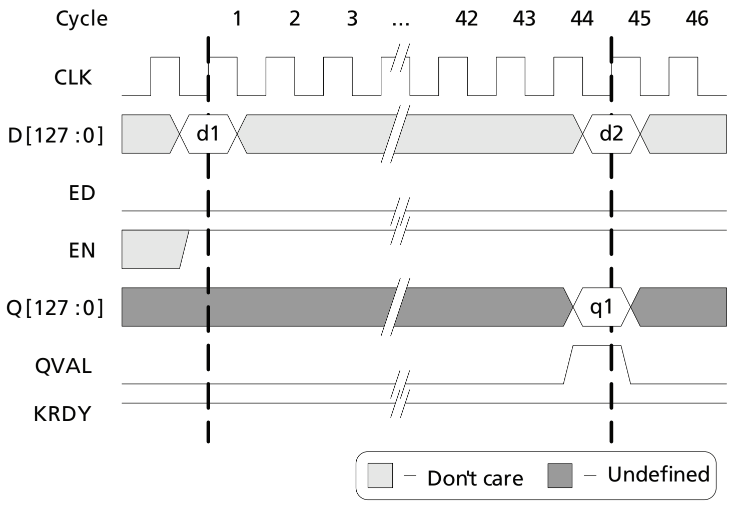


Figure 4 · Example Decryption Sequence

Pause/Resume

For normal operation, the EN input is held at logic '1'. The core can be paused by holding the EN input at logic '0' indefinitely, as shown by the example in Figure 5 on page 13, where cycle 3 of an encryption operation is paused. To resume operation, bring the EN input back to logic '1'. This functionality applies to either encryption or decryption. Note that the ED input must remain at logic '1' throughout an entire encryption cycle or at logic '0' throughout an entire decryption cycle; otherwise, unpredictable results on the Q[127:0] outputs will occur. You can use the pause/resume functionality in the case where many blocks of data are encrypted one after another. For example, if the EN input is held statically at logic '1', the data inputs need to change every 44 clock cycles to encrypt the next block. After all blocks of data are encrypted, you would then need to hold the EN input at logic ’0’. If it is left at logic '1', data will continue to be encrypted ad infinitum. When ready for the next blocks of data, resume the encryption process by holding the EN input at logic '1'. Another possibility occurs if there is an elastic buffer (FIFO) connected to the Q[127:0] output. If the FIFO is filling up with encrypted data faster than the encrypted data is being read out of the FIFO, you may want to pause the CoreAES128 macro by setting the EN input to a logic '0' when the full or almost-full flag logic from the FIFO is active. When the FIFO full or almost-full flag logic clears, the CoreAES128 macro can then resume operation by again setting the EN input to logic '1'.

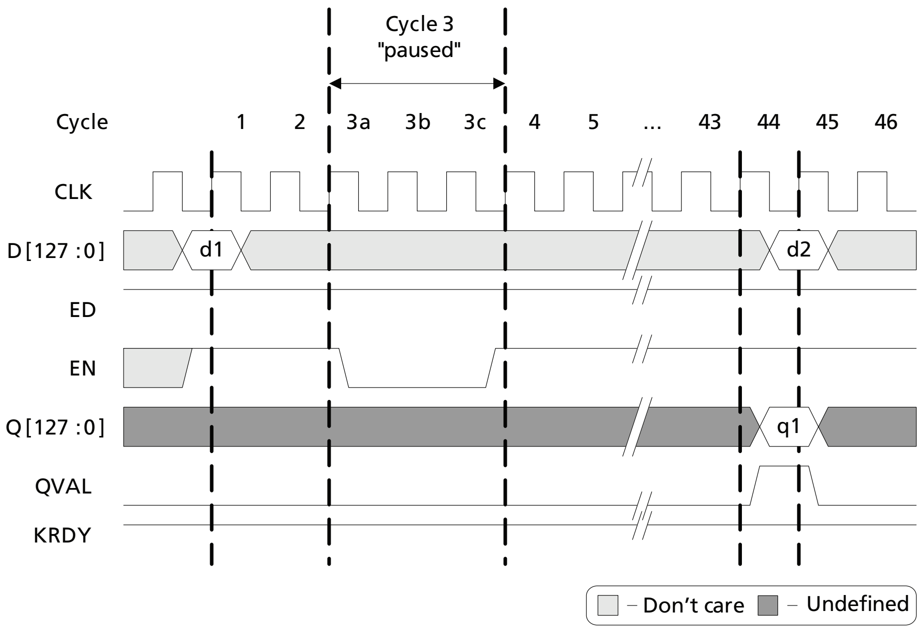


Figure 5 · Example Encryption Pause/Resume Sequence

Clear/Abort

At any point in the process of encrypting or decrypting data, you can abort the current operation by setting the CLR input to logic '1'. This will clear all current calculations within the key schedule and data schedule logic. Then you can immediately begin to write and expand a different cipher key, as described in "Cipher Key Expansion" section, or use a different data input on the very next cycle, as shown in Figure 6 on page 14, with d2 as the next 128-bit data block to be encrypted. Note that the CLR signal does not clear the 128-bit cipher key, the expanded version of the cipher key, or the KRDY signal. Only the signals NRESET, K[31:0], KWR, and KEXP affect the value of the 128-bit cipher key, the expanded version of the cipher key, and the KRDY output signal. The clear/abort functionality is provided as an additional tool. When you want to change the cipher key, possibly in the middle of an encryption or decryption sequence, you can stop the current operation immediately by holding the CLR input at logic '1' for at least one clock cycle new cipher key. After the new cipher key is expanded, new data can be encrypted. If the CoreAES128 macro is integrated into a system containing a processor, the processor may abort the encryption or decryption operation for a specific event, such as low or failing power condition.

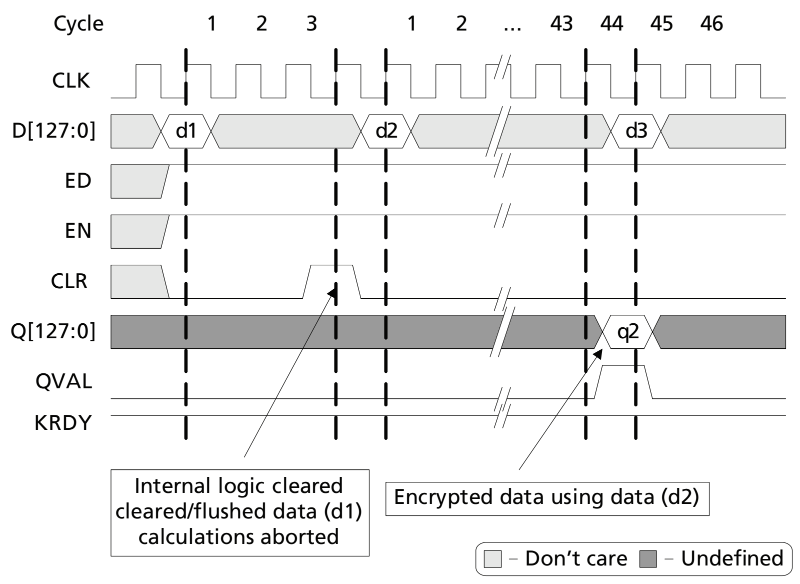

Cycle
1
2
3
1
2
...
43
44
45
46
CLK
D[127:0]
d1
d2
d3
ED
EN
CLR
Q[127:0]
q2
QVAL
KRDY
Internal logic cleared cleared/flushed data (d1) calculations aborted
Encrypted data using data (d2)
□ – Don't care ■ – Undefined

Figure 6 · Example Encryption Abort Sequence

Modes of Operation

CoreAES128 is implemented using the ECB mode of operation, per NIST SP 800-38A. Depending on the application, other modes of operation for AES may be desirable. For this reason, Microsemi provides example VHDL and Verilog source code for the CBC, CFB, OFB, and CTR modes. Refer to the *modes* directory for the wrapper design for each mode. For detailed information on specific modes of operation, refer to NIST SP 800-38A.

4 – Tool Flows

License

CoreAES128 is licensed in two ways. Depending on your license tool flow, functionality may be limited.

Obfuscated

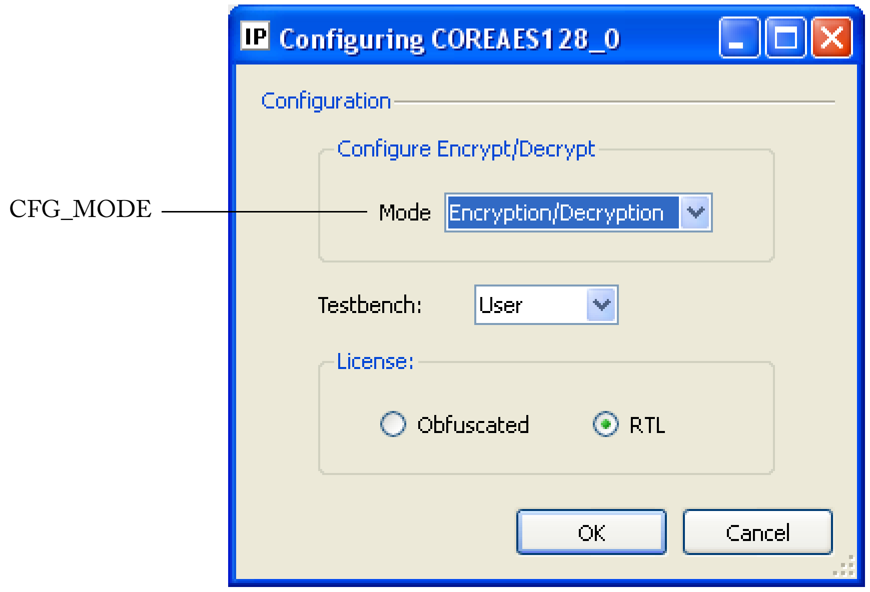
Complete RTL code is provided for the core, allowing the core to be instantiated with SmartDesign. Simulation, Synthesis, and Layout can be performed within Libero SoC. The RTL code for the core is obfuscated and the some of the testbench source files are not provided; they are precompiled into the compiled simulation library instead.

RTL

Complete RTL source code is provided for the core and testbenches.

SmartDesign

CoreAES128 is preinstalled in the SmartDesign IP Deployment design environment. The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 1. For information on using SmartDesign to configure, connect, and generate cores, refer to the Libero SoC online help.



Mode: Encryption/Decryption

Testbench: User

License: Obfuscated [✓ RTL]

Figure 1 · CoreAES128 Configuration Window

Simulation Flows

The user testbench for CoreAES128 is included in all releases. To run simulation, select the user testbench flow within SmartDesign and click **Save & Generate** on the Generate pane. The user testbench is selected through the Core Testbench Configuration GUI. When SmartDesign generates the Libero SoC project, it installs the user testbench files.

To run the user testbench, set the design root to the CoreAES128 instantiation in the Libero SoC design hierarchy pane, and click the Simulation icon in the Libero SoC Design Flow window. This will invoke Model*Sim*® and automatically run the simulation.

Synthesis in Libero SoC

Click the Synthesis icon in Libero SoC. The Synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, select the Run icon.

Place-and-Route in Libero SoC

After running Synthesis, click the Layout icon in the Libero SoC to invoke Designer. CoreAES128 requires no special place-and-route settings.

5 – Testbench Operation

User Testbench

An example user testbench is included with the obfuscated and RTL releases of CoreAES128. The obfuscated and RTL releases provide the precompiled Model*Sim* format, as well as the source code for the user testbench to ease the process of integrating and verifying the CoreAES128 macro into a design. A block diagram of the example user design and testbench is shown in Figure 1.

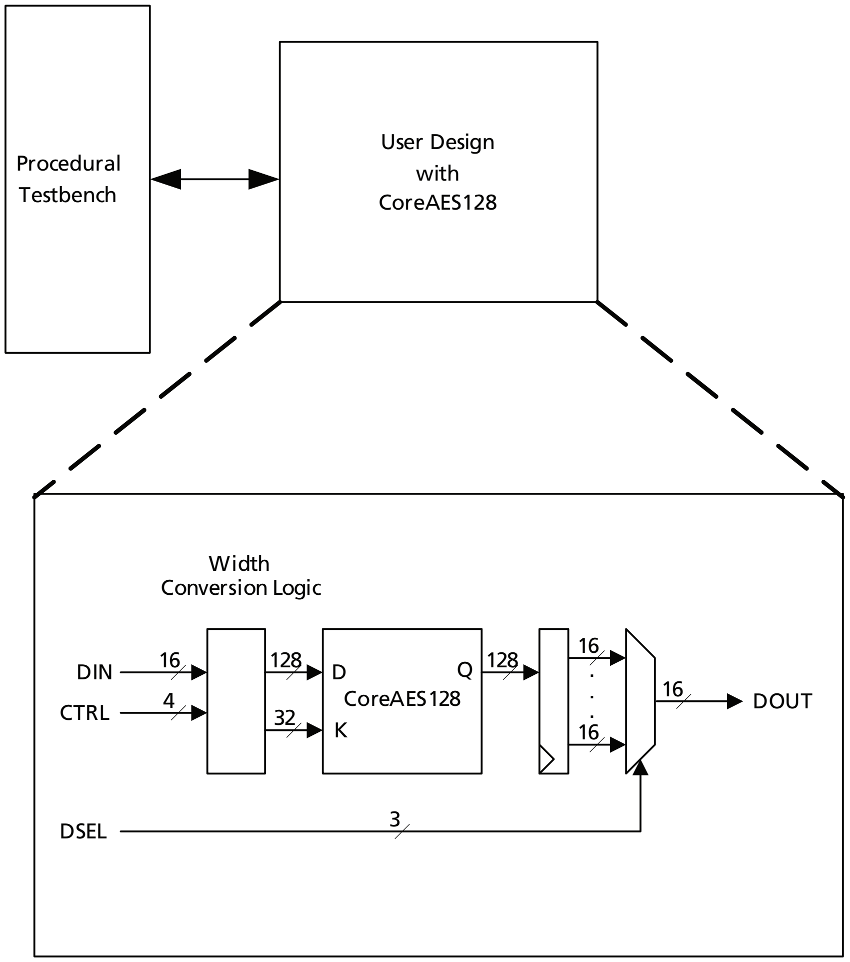


Figure 1 · Example User Design and User Testbench

The user testbench includes a simple example design that serves as a reference for users that want to implement their own designs. RTL source code for the example design and user testbench, shown in Figure 1, "wraps" around the CoreAES128 macro.

The source code for each user testbench includes example support routines to aid the user in testing an embedded system that contains the CoreAES128 macro. Refer to the comments in the user testbench source code for details on the support routines (tasks for Verilog testbenches, functions and procedures for VHDL testbenches.)

6 – Ordering Information

Ordering Codes

CoreAES128 can be ordered through your local Microsemi sales representative. It should be ordered using the following number scheme: CoreAES128-XX, where XX is listed in Table 1.

Table 1 • **Ordering Codes**

XX	Description
OM	OM for Obfuscated RTL source – multiple-use license
RM	RTL for RTL source – multiple-use license

Note: CoreAES128-OM is included free in the Libero SoC Catalog if you have a valid Libero SoC license

7 – Export Restrictions

CoreAES128 is subject to strict export controls and is licensable under the U.S. Department of Commerce Export Administration Regulations, the U.S. Department of State International Traffic in Arms Regulations, or other laws, government regulations, or restrictions. Microsemi is in the process of obtaining additional permissions to ship CoreAES128 to a wider audience. The licensee will not import, export, re-export, divert, transfer, or disclose CoreAES128 without complying strictly with the export control laws and all legal requirements in the relevant jurisdictions, including, without limitation, obtaining the prior approval of the U.S. Department of Commerce or the U.S. Department of State, as applicable.

A – List of Document Changes

The following table lists critical changes that were made in revisions of the document.

Revision	Changes	Page
Revision 6 (November 2014)	Added RTG4 information to "Supported Families" section and to all Device Utilization tables.	4, 5, 7,
	Updated the Handbook for CoreAES128 v3.3 release.	NA
Revision 5 (February 2014)	Updated "Supported Families" section and added IGLOO2.	4
Revision 4 (December 2012)	Added SmartFusion and SmartFusion2 in "Supported Families" section.	4
Revision 3 (February 2009)	Updated the Handbook for CoreAES128 v3.0 release.	NA
Revision 2 (January 2005)	Added ProASIC3 and ProASIC3E devices in "Supported Families" section.	4
Revision 1 (March 2003)	Initial release of CoreAES128 Handbook.	NA

B – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

50200122-6/11.14