CoreAES128

DirectCore

Product Summary

Intended Use

- Whenever Data is Transmitted across an Accessible Medium (wires, wireless, etc.)
- E-commerce Transactions Where Dedicated Encryption/Decryption Hardware can Ease the Load on Servers
- Personal Security Devices
- Bank Transactions Where State-of-the-Art Financial Security is Mandatory

Key Features

- Compliant with FIPS PUB 197
- ECB (Electronic Codebook) Implementation per NIST SP 800-38A
- 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

Targeted Devices

- ProASIC^{PLUS} Family
- Axcelerator Family

Core Deliverables

- Evaluation Version
  - Compiled RTL Simulation Model Fully Supported in Actel's Libero™ Integrated Design Environment (IDE)
- Netlist Version
  - Structural Verilog and VHDL Netlists (with and without I/O pads) Compatible with Actel's Designer Software Place-and-Route Tool
  - Compiled RTL Simulation Model Fully Supported in Actel's Libero IDE
- RTL Version
  - Verilog and VHDL Core Source Code
  - Core Synthesis Scripts
- Actel-Developed Testbench (Verilog and VHDL)

Synthesis and Simulation Support

- Synthesis: Synplicity, Synopsys (Design Compiler/FPGA Compiler/FPGA Express), Exemplar
- Simulation: OVI-Compliant Verilog Simulators and Vital-Compliant VHDL Simulators

Core Verification

- Actel-Developed Simulation Testbench Verifies CoreAES128 against Tests Available on the National Institute of Standards and Technology (NIST) Website: http://csrc.nist.gov/encryption/aes/rijndael/
- User can Easily Modify Testbench Using Existing Format to Add Custom Tests

General Description

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 (FIPS) Publication (PUB) 197 and is shown in Figure 1 on page 2.

The AES (Rijndael) algorithm takes as inputs 128 bits of plaintext data and 128 bits of a cipher key and after several rounds of computation, produces a 128-bit ciphered version of the original plaintext data as output.[1] During the rounds of the algorithm, the 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.

CoreAES128 consists of four main blocks (Figure 2 on page 2).

1. Data schedule logic – computes the intermediate data values at each round of the AES algorithm.
2. State correlator logic – maintains coherency between data and key schedule logic.
3. Key schedule logic – controls the intermediate key schedules at each round of the AES algorithm.
4. Key expansion logic – expands the original 128-bit key for use in encryption or decryption operations.

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.

© 2002 Actel Corporation

See Actel's website for the latest version of the datasheet.

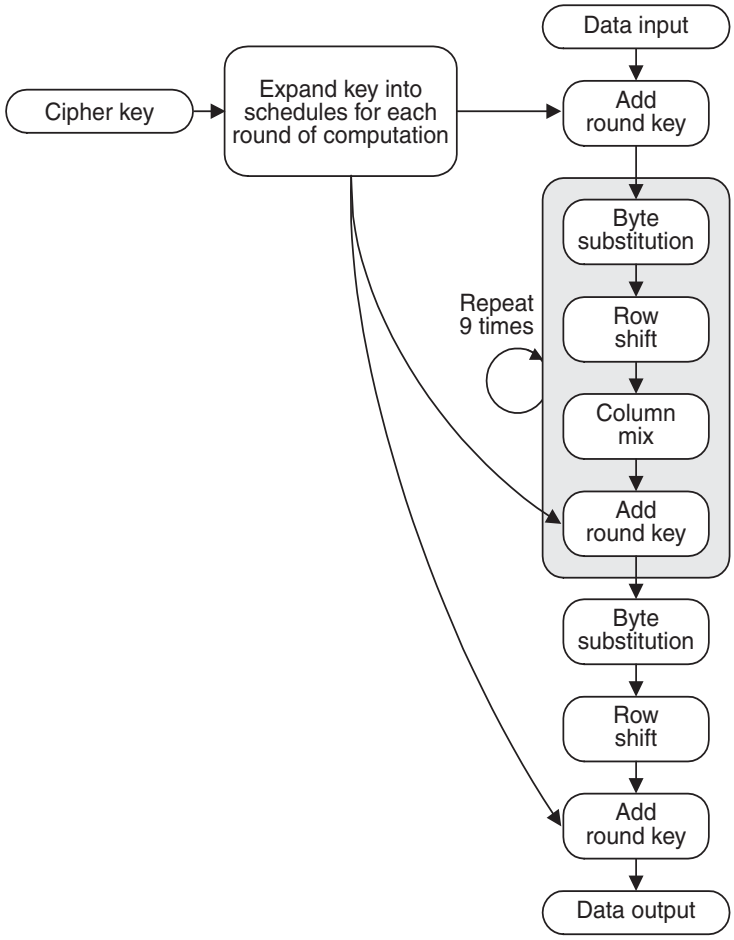

Figure 1 • *AES Algorithm (128-bit Cipher Key)*

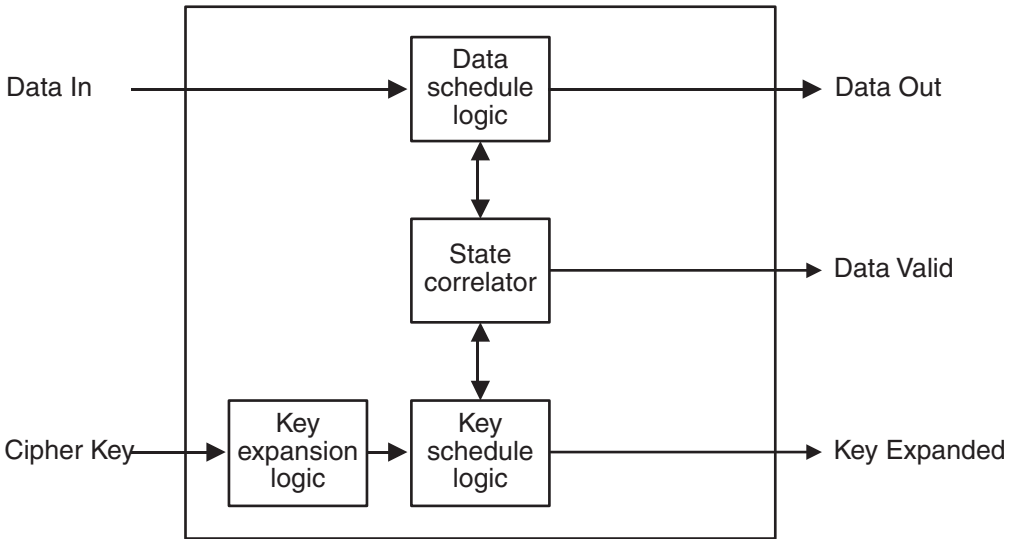

Figure 2 • *CoreAES128 Block Diagram*

Design Security

Figure 3 shows a typical system diagram. Note that the cipher key, which is the "secret" key, can be made up of FPGA logic cells preventing the possibility of design and data theft. Actel's Flash-based devices (ProASIC^{PLUS}) employ FlashLock™ technology, and Actel's antifuse-based devices (Axcelerator) use FuseLock™ technology, each of which secures the cipher key and the rest of the logic. The output of the CoreAES128 macro should be connected to registers or FIFOs, since it is only valid for one clock cycle, as shown by example in the "Encryption" section on page 6 and the "Decryption" section on page 7.

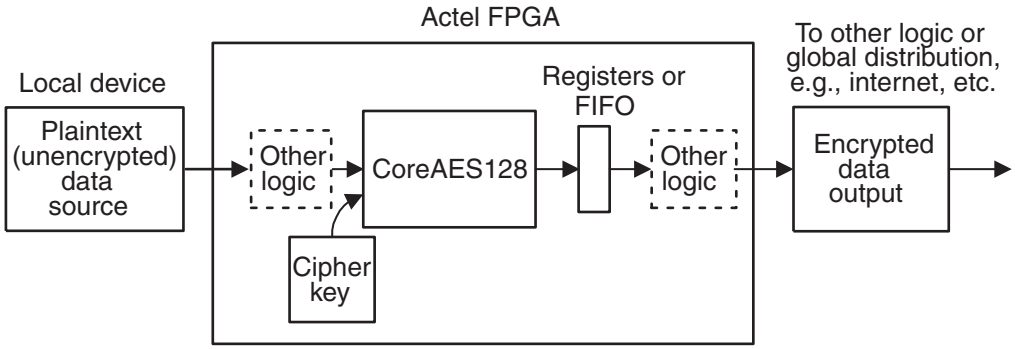

Figure 3 • *Typical CoreAES128 System*

CoreAES128 Device Requirements

The CoreAES128 macro has been implemented into Actel's ProASIC^{PLUS} and Axcelerator device families. A summary of the implementation data is listed in Table 1.

Table 1 • *CoreAES128 Device Utilization and Performance*

Family	Cells or Tiles		RAM blocks	Utilization		Performance	Throughput
	Sequential	Combinatorial		Device	Total		
ProASIC^{PLUS}	316	5,239	24	APA450-STD	46%	35 MHz	102 Mbps
Axcelerator	425	2,687	10	AX500-3	39%	100 MHz	291 Mbps

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

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

CoreAES128 Verification

The comprehensive verification simulation testbench (included with the Netlist and RTL versions of the core) verifies the CoreAES128 macro against test cases listed on the NIST website for AES: http://csrc.nist.gov/encryption/aes/rijndael/.

The verification testbench applies several tests to the CoreAES128 macro, including: variable text tests, variable key tests, table tests, and Monte Carlo tests. Using the supplied user testbench as a guide, the user can easily customize the verification of the core by adding or removing tests.

I/O Signal Descriptions

The port signals for the CoreAES128 macro are defined in Table 2 on page 4 and illustrated in Figure 4 on page 4. All signals are either "Input" (input-only) or "Output" (output-only).

CoreAES128 Initialization

After a reset condition, as illustrated in Figure 5 on page 4, the CoreAES128 macro performs a self-initialization process. This initialization process takes 1,024 clock cycles to perform, after which, the READY signal becomes active at logic '1.' Once READY is active, the CoreAES128 macro is ready for cipher key expansion, followed by encrypt or decrypt operations.

CoreAES128 Operation

As shown on the left side of Figure 1 on page 2, 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 setup after the initialization process, described in the "CoreAES128 Initialization" section on page 3, and before encryption or decryption operations can take place. The following procedures (located in the "Cipher Key Expansion" section on page 5) for writing and expanding the cipher key must be repeated anytime 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 following procedures for writing and expanding the cipher key only need to be performed once.

Table 2 • *CoreAES128 I/O Signal Descriptions*

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	Encrypt/decrypt: '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 four 32-bit words comprising internal 128-bit cipher key
KWR	Input	Key write: set to '1' to write K[31:0] to one of four 32-bit words comprising 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 ciphertext (encrypt operation) / plaintext (decrypt operation) output bus
QVAL	Output	Q Valid: '1' indicates that valid encrypt/decrypt 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

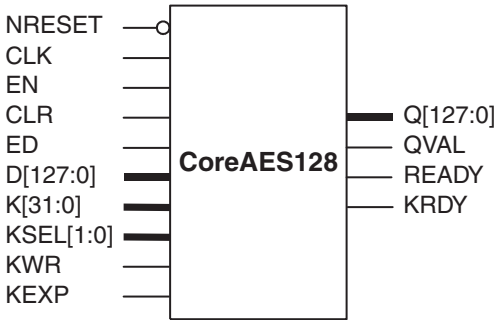

Figure 4 • *CoreAES128 I/O Signal Diagram*

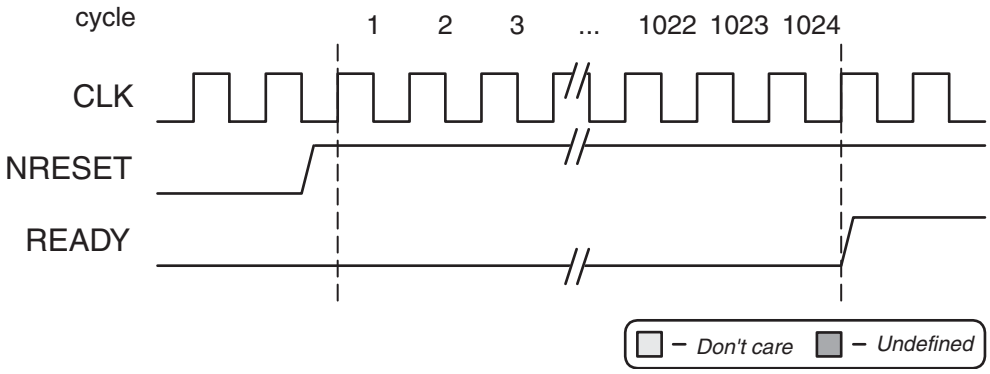

Figure 5 • *CoreAES128 Initialization*

Cipher Key Expansion

Prior to any encryption or decryption operation, the 128-bit cipher key needs to be written to CoreAES128, and expanded, as illustrated in Figure 6. Refer to FIPS PUB 197 for the algorithmic details of the key expansion process.

To write the four 32-bit words that make up the 128-bit cipher key, and to expand the 128-bit cipher key, the following procedures need to be performed:

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.

Note that 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 a logic '1' value, 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 step 4 in the list above. After 52 clock cycles, the KRDY signal becomes active, i.e., logic '1,' to indicate that the 128-bit cipher key was expanded internally, and 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, as illustrated in Figure 6, and prior to the key expansion process.

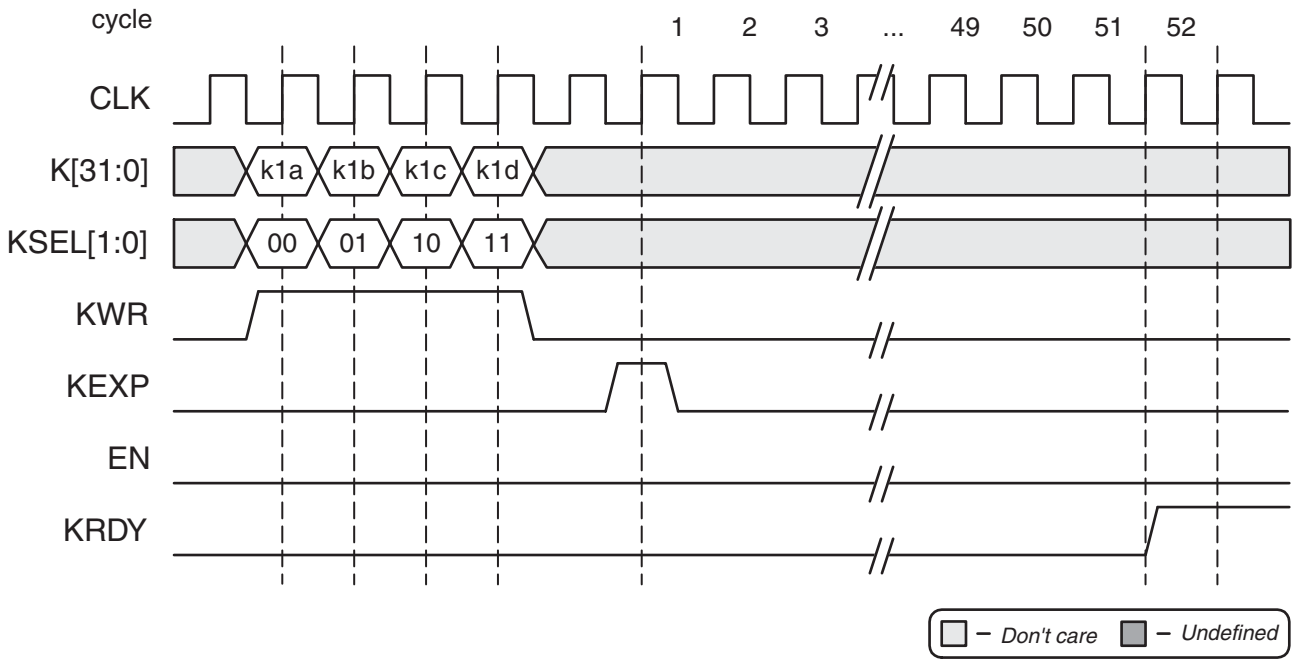

Figure 6 • *Cipher Key Write and Expand*

Encryption

To begin the process of encrypting data, as shown in Figure 7, perform the following procedures:

1. Write and expand the 128-bit cipher key, if not already done (refer to the "Cipher Key Expansion" section on page 5).
2. Set D[127:0] to the plaintext data ("d1" in Figure 6) to be encrypted.
3. Set ED to logic '1.'
4. Set EN to logic '1.'
5. Wait for 44 clock cycles.

After 44 clock cycles of the EN input being held continuously at a logic '1' value, 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 7) is available on the Q[127:0] outputs. Note that the encrypted data is only available during clock cycle 44, thus the user 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 7, continuous encryption is possible. For example, the second 128-bit plaintext data word ("d2" in Figure 7) can be immediately encrypted by setting the D[127:0] inputs to d2 on the rising clock edge of clock cycle 45.

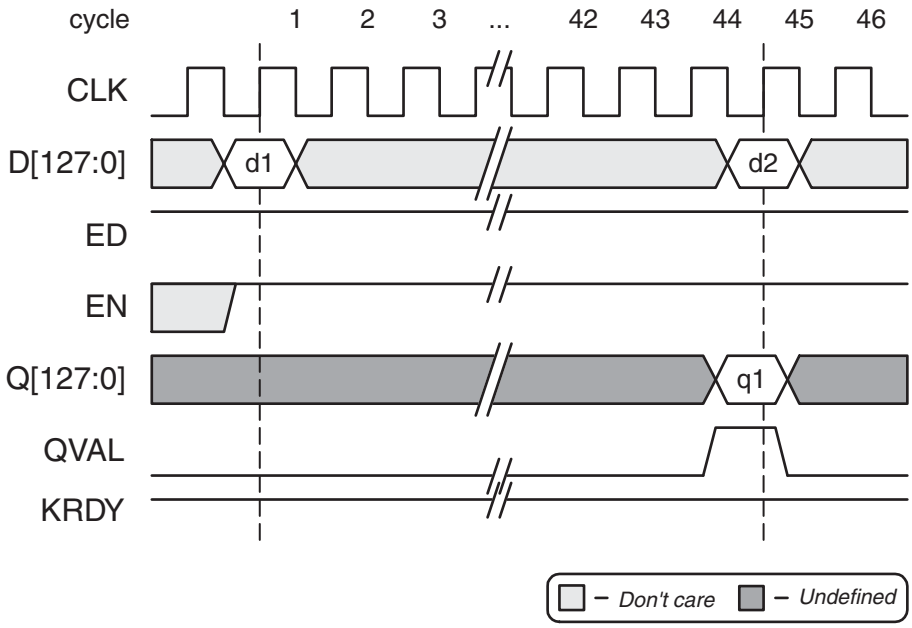

Figure 7 • *Example Encryption Sequence*

Decryption

To begin the process of decrypting data, as shown in Figure 8, perform the following procedures:

1. Write and expand the 128-bit cipher key, if not already done (refer to the "Cipher Key Expansion" section on page 5).
2. Set D[127:0] to the ciphertext data ("d1" in Figure 7) to be decrypted.
3. Set ED to logic '0.'
4. Set EN to logic '1.'
5. Wait for 44 clock cycles.

After 44 clock cycles of the EN input being held continuously at a logic '1' value, 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 8) is available on the Q[127:0] outputs. Note that the decrypted plaintext data is only available during clock cycle 44, thus the user 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 8, continuous decryption is possible. For example, the second 128-bit ciphertext data word ("d2" in Figure 8) can be immediately decrypted by setting the D[127:0] inputs to d2 on the rising clock edge of clock cycle 45.

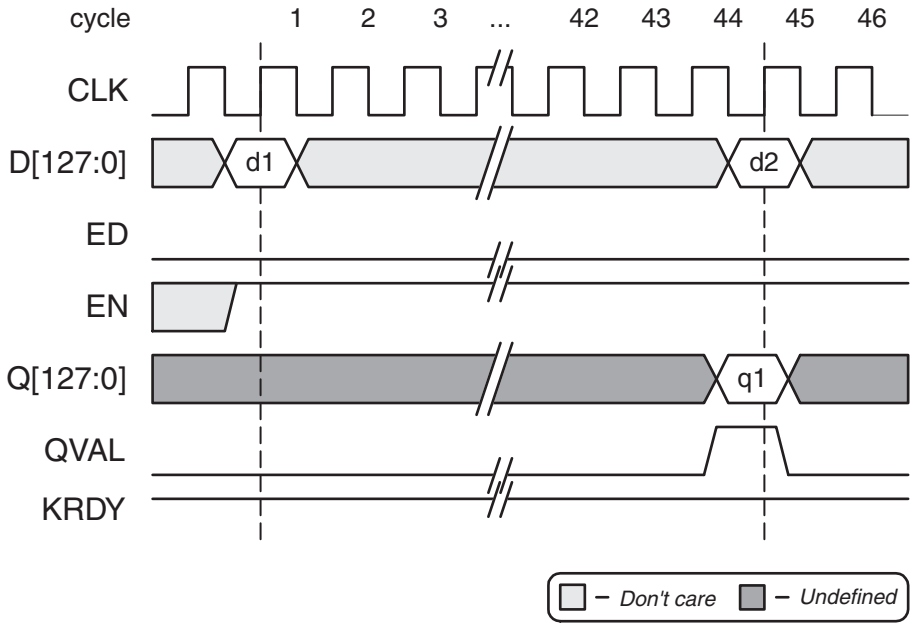

Figure 8 • *Example Decryption Sequence*

Pause/Resume

For normal operation, the EN input is held at a logic '1' value. The core can be paused by holding the EN input at a logic '0' value, indefinitely, as shown by the example in Figure 9 where cycle 3 of an encryption operation is paused. To resume operation, bring the EN input back to a logic '1' value. 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.

The pause/resume functionality is provided as an aid to the user. One possible use for the pause functionality is a case where many blocks of data are encrypted one after another. For example, if the EN input is held statically at a logic '1' value, the data inputs need to change every 44 clock cycles to encrypt the next block. After all blocks of data are encrypted, the user would then need to hold the EN input at a logic '0' value. If it is left at a logic '1,' data will continue to be encrypted ad infinitum. When ready for the next blocks of data, the user can then resume the encryption process by holding the EN input at a logic '1' value. Another possibility occurs if the user has an elastic buffer (FIFO) connected to the Q[127:0] outputs. If the FIFO is filling up with encrypted data faster than the encrypted data is being read out of the FIFO, the user 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 a logic '1' value.

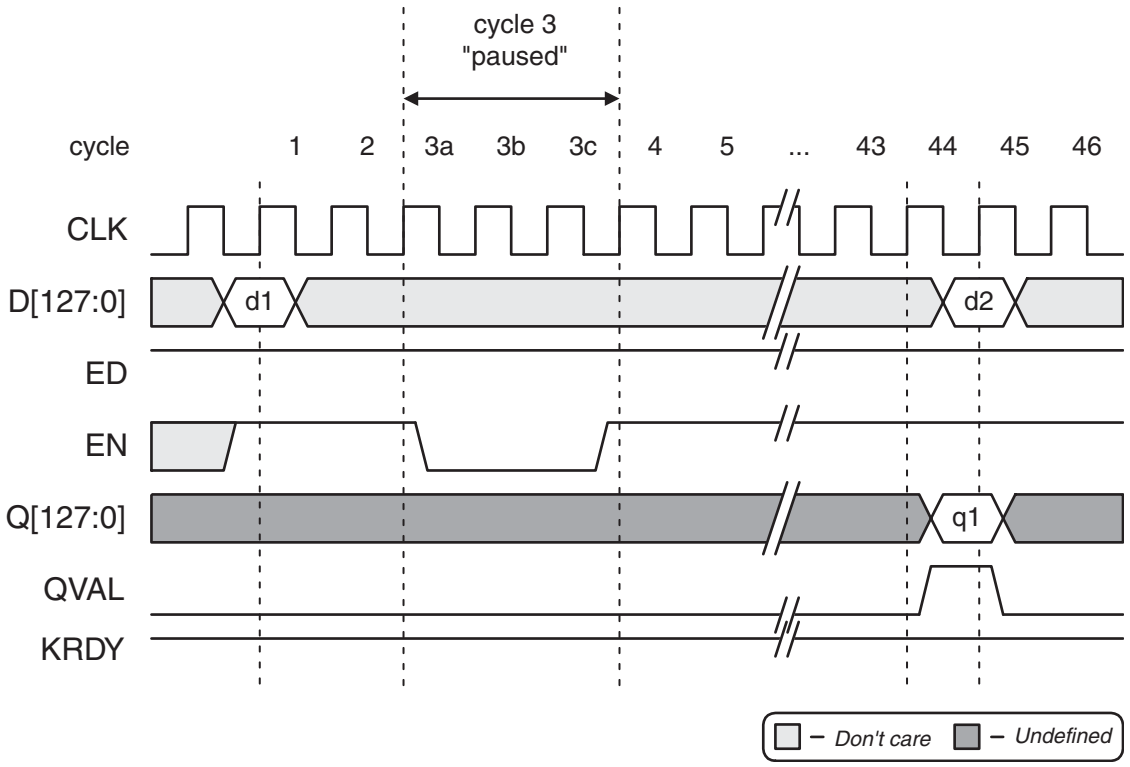

Figure 9 • *Example Encryption Pause/Resume Sequence*

Clear/Abort

At any point in the process of encrypting or decrypting data, the user 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, the user can immediately begin to write and expand a different cipher key, as described in the "Cipher Key Expansion" section on page 5, or use a different data input on the very next cycle, as shown in Figure 10, 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, nor 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 another aid to the user. An example of its use occurs when the user wants to change the cipher key, possibly in the middle of an encryption or decryption sequence. Immediately, the user can stop the current operation simply by holding the CLR input at a logic '1' value for at least one clock cycle and immediately commence on the following clock cycle with writing and expanding a 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 wish to abort the encryption or decryption operation for some specific event (e.g., low or failing power condition).

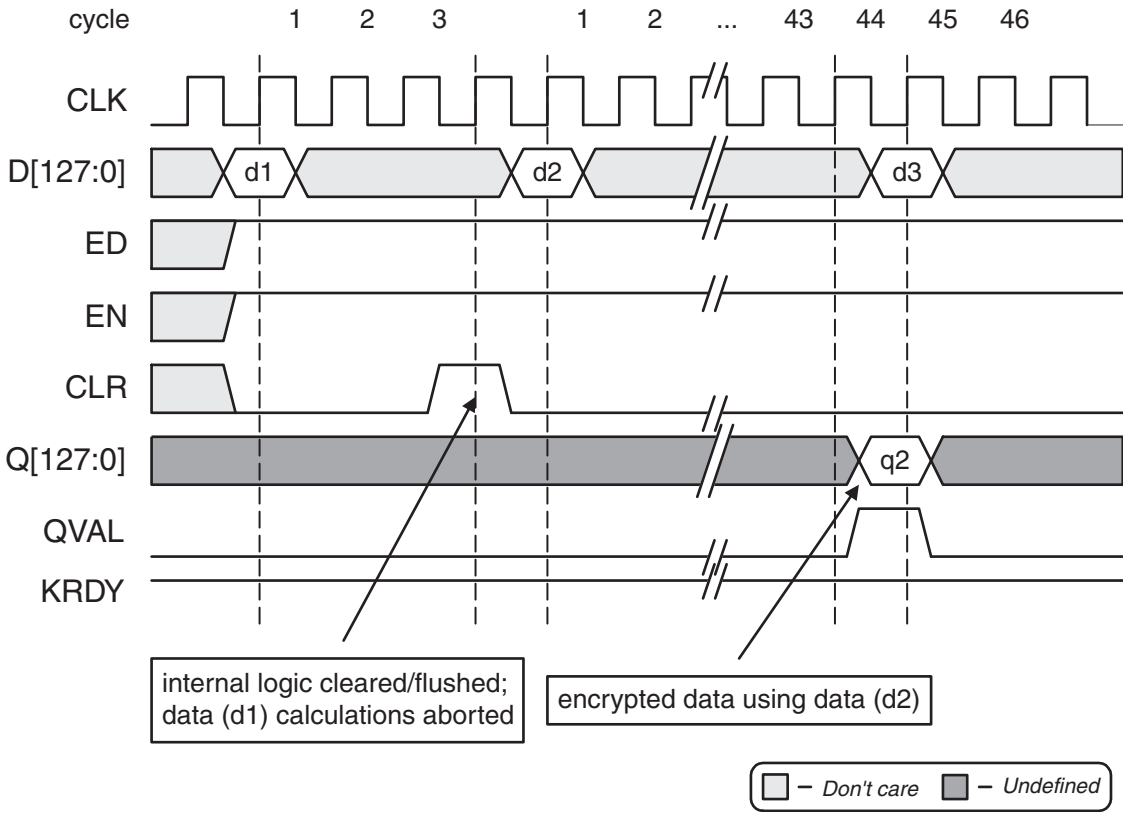

***Figure 10** • Example Encryption Abort Sequence*

Ordering Information

Order CoreAES128 through your local Actel sales representative. Use the following number convention when ordering: CoreAES128-XX, where XX is listed in Table 3.

Table 3 • *Ordering Codes*

XX	Description
EV	Evaluation Version
SN	Single-use Netlist for use on Actel devices
AN	Netlist for unlimited use on Actel devices
AR	RTL for unlimited use on Actel devices
UR	RTL for unlimited use and not restricted to Actel devices

Export Restrictions

CoreAES128 is subject to strict export controls and is licensable under the U.S. Department of Commerce's Export Administration Regulations, the U.S. Department of State's International Traffic in Arms Regulations, or other laws, government regulations or restrictions. Actel is in the process of obtaining additional permissions to ship CoreAES128 to a wider audience. The licensee will not import, export, reexport, 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.

Datasheet Categories

Product Definition

This version of the datasheet is the definition of the product. A prototype may or may not be available. Data presented is subject to significant changes.

Advanced

This version of the datasheet provides nearly complete information for a prototype IP product. Code is fully operational, but may not support all features expected in the production release. A prototype core and a preliminary testbench are available.

Production (unmarked)

This version of the datasheet contains complete information on the final core. All components are fully operational and the core has been thoroughly verified.

Actel and the Actel logo are registered trademarks of Actel Corporation.
All other trademarks are the property of their owners.

http://www.actel.com

Actel Europe Ltd.
Maxfli Court, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Tel: +44 (0)1276 401450
Fax: +44 (0)1276 401490

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
USA
Tel: (408) 739-1010
Fax: (408) 739-1540

Actel Asia-Pacific
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Tel: +81 03-3445-7671
Fax: +81 03-3445-7668

51700011-0/12.02