# AES Encryption

April 2013 Reference Design RD1176

## Introduction

Cryptography enables the confidentiality of communication through an insecure channel and hence protects against unauthorized parties by preventing unauthorized alteration of use. The Advanced Encryption Standard, AES (Rijndael) algorithm is implemented as described in the NIST (National Institute of Standards and Technology) Federal Information Processing Standard. This design example implements the 128-bit block-size AES algorithm, which accepts a 128-bit plain data input word, and generates a corresponding 128-bit cipher output word using a supplied 128, 192, or 256-bit AES key.

This document provides a brief description of AES Encryption and its implementation.

The design is implemented in VHDL. The Lattice iCEcube2™ Place and Route tool integrated with the Synopsys Synplify Pro® synthesis tool is used for the implementation of the design. The design can be targeted to other iCE40™ FPGA product family devices.

## Features

The following features are supported:

- Encrypts using the AES Rijndael Block Cipher Algorithm
- Processes 128-bit data blocks with 32-bit data interface
- Parameterizable key length (can be 128, 192 or 256)
- Includes the key expansion function and s-box memory
- Completely self-contained: does not require external memory
- Internal Buffer to hold encrypted data

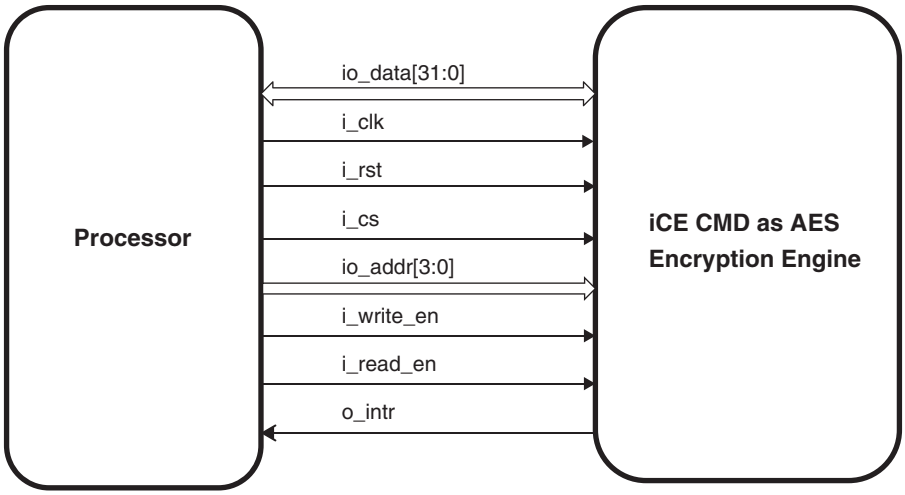
The following features are NOT supported:

- 128-bit data path for high speed systems
- Support for all Cipher Modes - CBC,CFB,OFB,CTR
- Generic 8-bit, 16-bit, 64-bit, 128-bit external interface

## System Block Diagram

Figure 1 shows the system block diagram of this reference design.

***Figure 1. System Block Diagram***



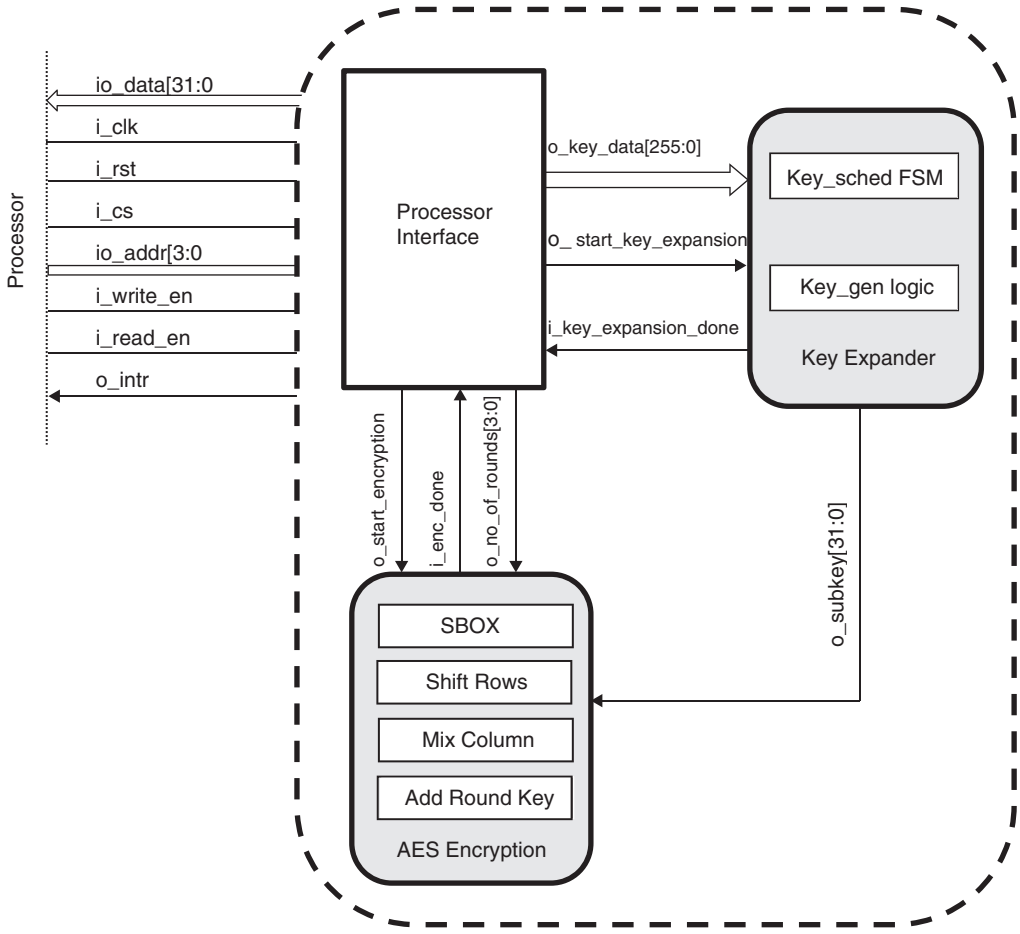

## Signal Description

***Table 1. Signal Description***

| Signal | Width | Type | Description |
|---|---|---|---|
| i_clk | 1 | Input | System Clock |
| i_rst | 1 | Input | Asynchronous Active High System Reset |
| i_cs | 1 | Input | Active high chip select form processor |
| i_read_en | 1 | Input | Active high read enable from processor |
| i_write_en | 1 | Input | Active high write enable from processor |
| o_intr | 1 | Output | Active high interrupt to processor |
| i_addr | 4 | Input | Register Address |
| io_data | 32 | Inout | Data bus |

# Design Module Description

***Figure 2. Design Module Description***




# Configurable Parameters

## key_length

This parameter configures the length of the key to be used. Its default value is 256. It can be configured to 192 or 128 as well

## Register Map

There are a set of predefined registers, which controls the data flow and operation of AES Engine. Table 2 summarizes the available registers and gives a brief description.

***Table 2. AES Register and Address mapping***

| Address | Register | Description |
|---|---|---|
| 0X0 | AES_STATUS | AES Engine Status Register |
| 0X4 | AES_CONTROL | AES Engine Control Register |
| 0X8 | AES_DATA | Write and Read to/from the AES Engine |
| 0XC | AES_KEY | Write Register-key data |

## AES status register

This read-only register holds the basic status information of the AES Engine. The Table 3 shows the contents of the AES status register in detail. All the status bits are active high (e'1').

***Table 3. AES Status Register***

| Bit31-6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|
| X | CRYPTDONE | EXPDONE | IDATFULL | KEYFULL | RST | BUSY |

- BUSY: This bit is set during encoding, decoding and during key expansion.
- RST: This bit is set after a reset through the AES_CONTROL register and will be deactivated with the first valid write access to one of the AES registers.
- KEYFULL: This bit is set as soon as processor writes 128/192/256 bits of key to the Key data register. Power-On reset or RESET command clears KEYFULL bit of status register. Also, this status bit is cleared as soon as processor starts writing to AES Key register and goes active as soon as AES Engine receives the necessary number of bits defined in KEY LENGTH of Control Register.
- DATFULL: This bit is set as soon as processor writes 128 bits of Data to the Data register. Power-On reset or RESET command clears IDATFULL bit of status register. Also, this status bit is cleared as soon as processor starts writing to AES Data register and goes active as soon as AES Engine receives 128-bits of data. This bit again cleared when AES Engine read whole 128-bits of data and started processing it from buffer. A new set of data could be written to data register when it is cleared.
- EXPDONE: Indicates completion of Key Expansion process. Processor can check this bit when AES Engine generates an interrupt.
- CRYPTDONE: Indicates completion of encryption process. Processor can check this bit when AES engine generates an interrupt. This bit cleared as soon as processor starts reading from AES data register.

## AES Control Register

This read-write register controls the operation of AES Engine. The Table 4 shows the contents of the AES control register in detail.

***Table 4. AES Control Register***

| Bit31-2 | Bit1 | Bit0 |
|---|---|---|
| X | Command | Word |

A command word defines the required action of the AES Engine. RESET operation clears KEY, Data, Status and Control Registers. Valid command words are listed in Table 5 Control Register – Command Fields.

***Table 5. Control Register - Command Field***

| Command | Description |
|---|---|
| 00 | NOP |
| 01 | RESET |
| 10 | KEY SETUP |
| 11 | AES DECRYPT |

Note that (1). Requesting an action during active operation (busy) of the AES Engine may corrupt the results of this operation. (2). changing the key length during active operation (busy) of the AES core may corrupt the results of this operation. There is no need to reset the contents prior sending a new command.

### AES Key Register

The AES key write-only register is used to write key data to the AES Engine, MSB word first mode.

### AES Data Register

The AES Data read-write register is used to write data to and read data from the AES Engine. 128 bit cipher and plain data is transferred to and from the AES Engine by sending/reading the MSB word first.

## Design Details

AES Encryption is the process of transforming information using Rijndael block cipher algorithm to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The Encrypted data can only be deciphered if one has the password or the Key. The Encryption algorithm is a set of well defined steps to transform data from a readable format to an encoded format using the Key. The result of the process is encrypted information, referred to as cipher text.

AES is an iterated block cipher with a fixed block size of 128 and a variable key length. The AES algorithm operates on 128 bits of data and generates 128 bits of output. The length of the key used to encrypt this input data can be 128, 192 or 256 bits. AES has been designed as a substitution-permutation network (SPN) and encryption process uses 10, 12 and 14 encryption rounds for key length of 128, 192 and 256 bits respectively.

The AES Engine consists of three units. The Processor Interface unit holds all the registers required for the communication with processor. The Key Expander module holds and manipulates the key data. The AES Encryption module holds data as well as AES state and performs the AES transformations like ciphering.

### Processor Interface

An Interface between the processor and the AES system is based on a generic parallel bus protocol. This processor interface controls the movement of control commands, address and data between the processor and the AES Engine. Since the data movement uses 32-bit mode, it has to be de-serialized into a 128-bit data and sent to Encryption engine. In a similar fashion, 128-bit data from the AES Engine is serialized (32-bit mode) before sending it to the processor.

This module is also responsible for

- Generating control signals to trigger key expansion and encryption process.
- Supply the data and key from its buffer to the appropriate modules depending upon the control signals.
- Write the encrypted data to its internal buffer from appropriate modules depending upon the control signals.
- Generating an interrupt to the processor when AES Engine completes Key Expansion and AES Encryption of 128-bits of data.

### Key Expansion

Key Expansion operation is triggered when control register receives KEY_SETUP command word. The AES algorithm takes the Cipher Key K, and performs a Key Expansion routine to generate a key schedule. Nb defines the number of columns of 32 bits in the 128-bit data, which is Nb =128/32 =4. Similarly, Nk defines the number of columns of 32 bits of key, which is, Nk = 128/32= 4. For key length of 192 and 256 the values of Nk will be 192/32= 6 and 256/32= 8 respectively.

The number of rounds Nr =10 when Nk= 4 and changes to 12 and 14 for Nk =6 and Nk =8. The Key Expansion generates a total of Nb (Nr + 1) words: the algorithm requires an initial set of Nk words, and each of the Nr rounds requires Nk words of key data. As soon as key expansion process is finished, the BUSY bit and EXPDONE bits of the AES status register is set and processor is interrupted after which processor checks the status of the AES engine and determines the AES engine has finished key expansion process.

## AES Encryption

The AES Encryption process works as follows: 1). Processor writes the key data to AES Key register. Now the key setup operation has to be performed. The key setup operation is requested by writing the KEY_SETUP command to the AES Control register. The completion of the key setup operation is flagged by the interrupt signal and the busy bit in the AES Status register. 2). Processor writes plain data to AES Data register. The 128 bit input data block is delivered by four subsequent write commands to the AES Data register. Finally the encoding operation is started by writing the command AES_ENCRYPT to the AES Control register. The completion of the encoding operation is flagged by the interrupt signal and the busy bit in the AES status register. Now the encrypted data may be read from the AES Data register by four subsequent read commands. The key remains in the module until a new key is written or a reset command is performed.

# Timing Diagram

***Figure 3. AES Data Write Access***

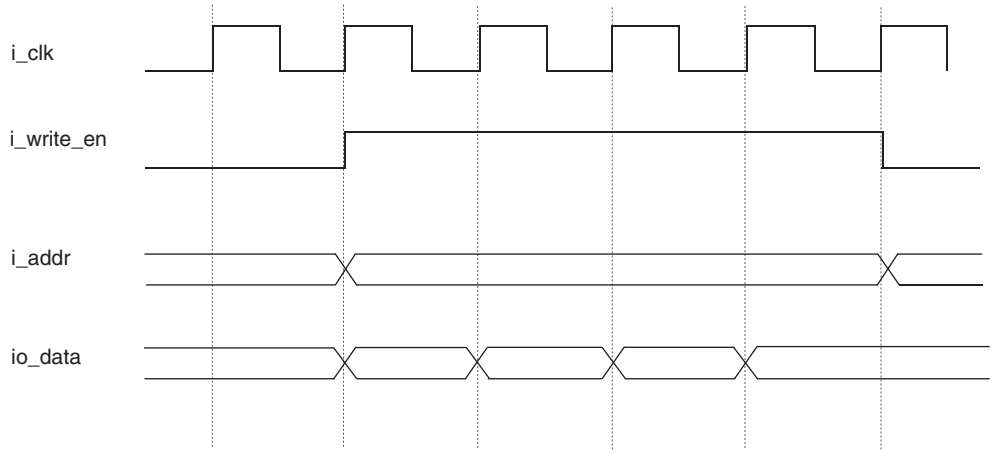



***Figure 4. Data Read Access***

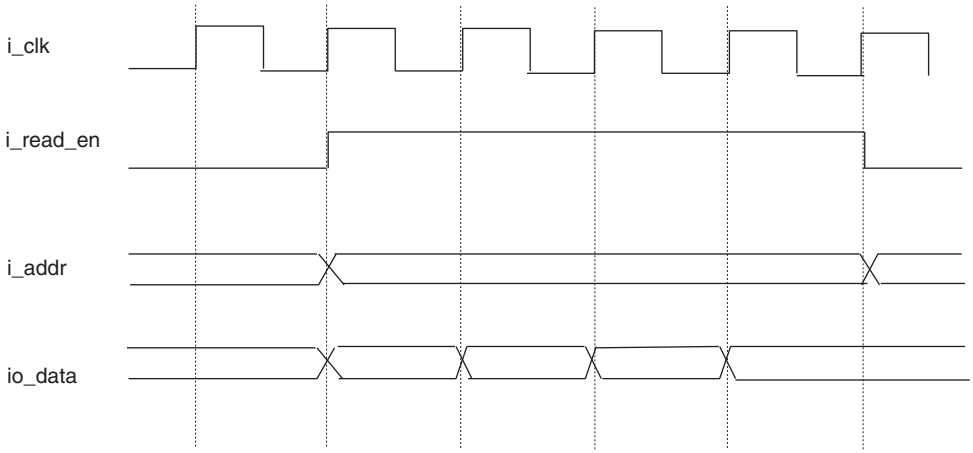


***Figure 5. : Key Expansion Process***

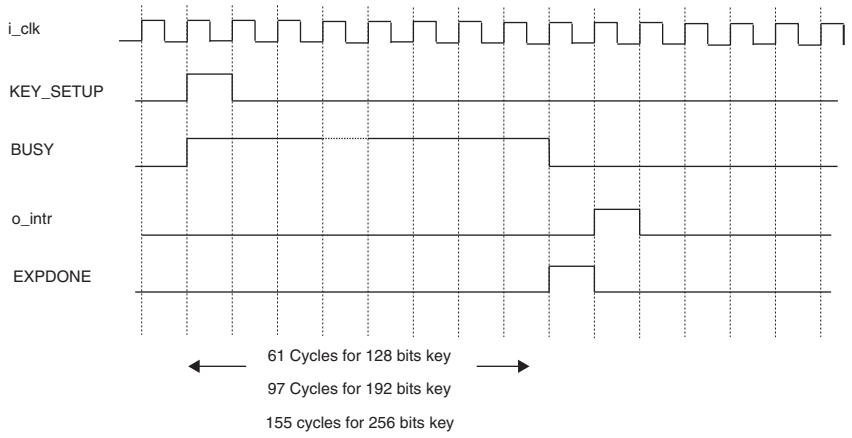
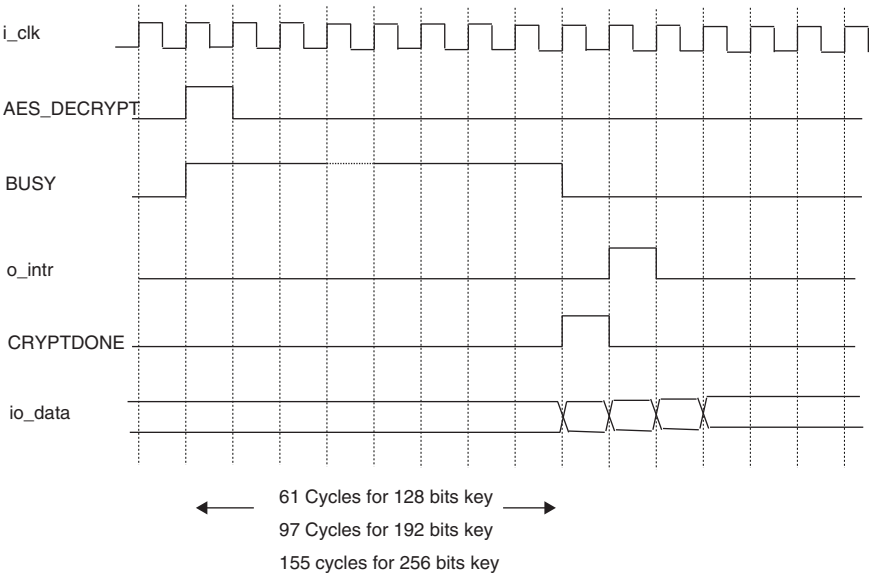
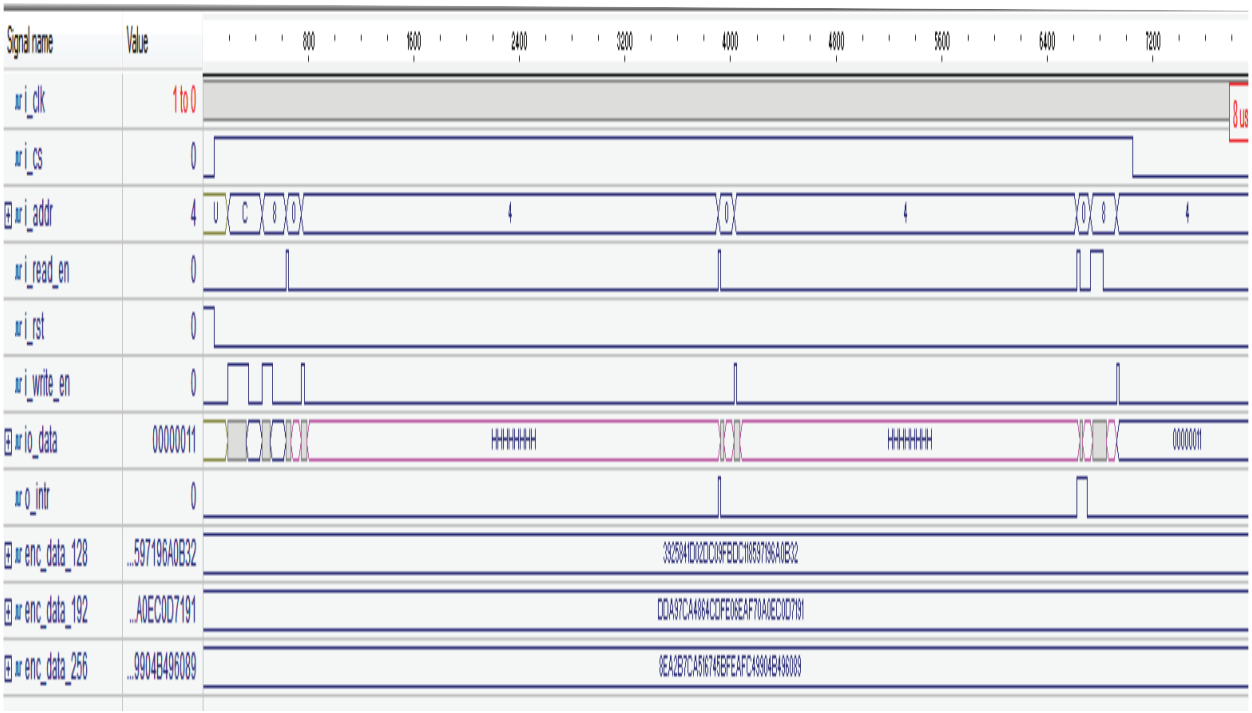



***Figure 6. AES Encryption Process***



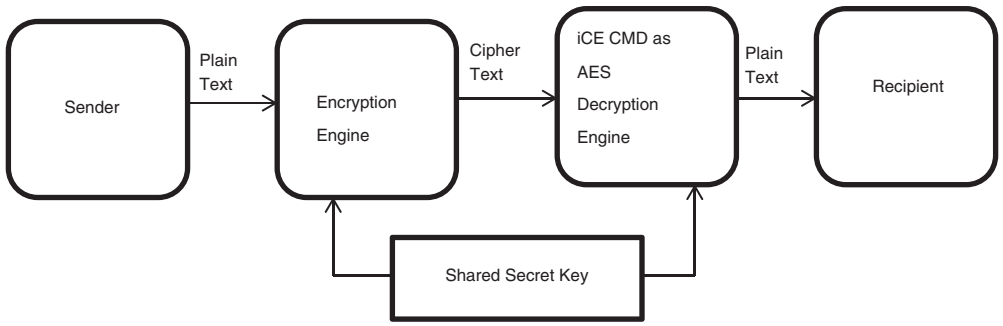
## Simulation Waveforms

***Figure 7. Simulation Waveforms***




## Usage Examples

***Figure 8. Usage Example***




The sender creates a cipher text message by encrypting the plain text message with the iCE40 AES Encryption Engine and a shared secret key. The sender writes the key data to AES Key register. Then the key setup operation is performed. Then the sender writes plain data. The 128 bit input data block is delivered by four subsequent write commands to the AES Data register. This is followed by the encoding operation. Now the encrypted data may be read from the AES Data register by four subsequent read commands. The sender sends the cipher text message to the recipient.

The recipient decrypts the cipher text message back into plain text with the shared secret key.

## Operation Sequence

1. Use the aese.do to run the simulation.
2. The scripts also runs the RTL simulation waveform.
3. Simulation starts with RESETIng the FPGA
4. Loading the control register:

   a. Supply appropriate address of the control register and enable signals.
   b. Supply the content of the control register.

5. Loading the key register content:

   a. Supply the address of key register and enable signals.
   b. Supply appropriate key values.

6. Loading data register content:

   a. Supply the address of data register and enable signals
   b. Supply appropriate data values.

7. Apply status register read command with status register address and appropriate valid signals.
8. Send key setup command to start key expansion process.
9. Read the status after key expansion is done.
10. Load data register with decrypted key values.
11. Send start decryption command.
12. Read the status register after finishing the decryption process.
13. Repeat step #1 through step #12 for testing with 192 and 256 bits key with appropriate key lengths.

## Implementation

This design is implemented in VHDL and Verilog language. When using this design in a different device, density, speed or grade, performance and utilization may vary.

***Table 6. Performance and Resource Utilization***

| Family | Language | Utilization (LUTs) | $f_{MAX}$ (MHz) | I/Os | Architectural Resources |
|---|---|---|---|---|---|
| iCE40[1] | VHDL and Verilog | 2522 | >50 | 75 | (365/440)PLBs |

1. Performance and utilization characteristics are generated using iCE40HX4K-TQ144 with iCEcube2 design software.

## References

- iCE40 Family Handbook

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

## Revision History

| Date | Version | Change Summary |
|---|---|---|
| April 2013 | 01.0 | Initial release. |