

Datasheet for Advanced Encryption Standard

Introduction

Rijndael is a block cipher, designed by Belgian cryptologists Joan Daemen and Vincent Rijmen as a candidate algorithm for the Advanced Encryption Standard (AES). NIST has selected Rijndael as the proposed AES algorithm. The block cipher Rijndael is designed to use only simple whole-byte operations. It supports 128-bit key size with 128 bit fixed input data size.

Functional Description:

Rijndael is an iterated block cipher, meaning that the initial input block (plaintext) and cipher key undergoes multiple transformation cycles before producing the output

Features:

- Simple user interface.
- Support 128 AES key size.
- Sequential and parallel version available
- Separate core for encryption, decryption and round key generation.
- 128-bit wide data path processes.

(ciphertext). The algorithm can operate over a 128-bit length keys; a 128-bit to encrypt data blocks. Fig 1 shows the block diagram of AES core.

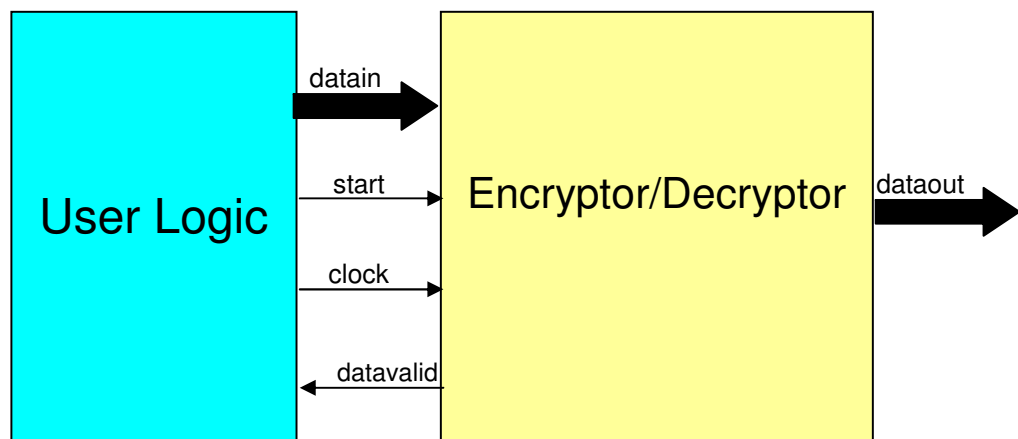


Fig 1 AES Core Block Diagram

Before starting the encryption or decryption process, the round key for the corresponding key and plaintext is calculated. A behavioral code is written to make this task easier. There are separate cores for encryption and decryption and are designed to support 128-bit key size. There are 9 rounds for 128 bit

key.

The interface of the core has been designed to be extremely simple to use and can be integrated into any system. All the stages in the core are unrolled to get maximum throughput.

I/O Ports:

I/O ports for AES core are summarized in Table1.

Table 1 I/O Ports

Port Name	Direction	Width (bit)	Description
palintext	input	128	Data input

clock	input	1	System Clock. All registers are synchronized to rising edge of this signal.
start	input	1	This signal will go high when a new set of data is ready to process.
datavalid	output	1	This signal goes high when valid dataout is available on dataout bus.
ciphertext	output	128	Data output

Encryption Core:

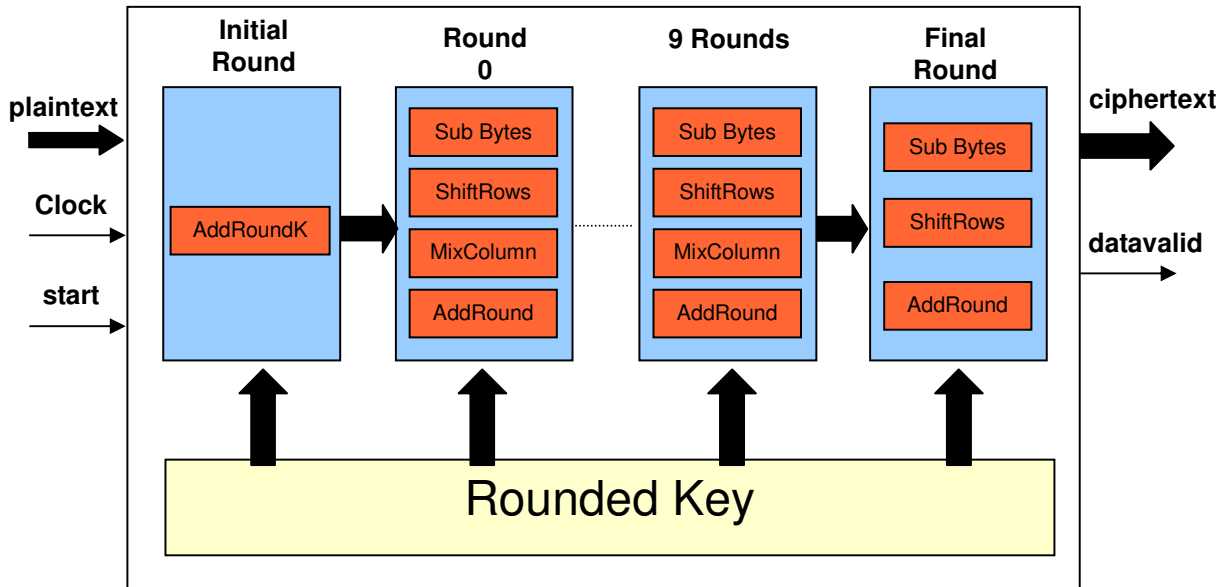


Fig 2 Encryption Block Diagram

Process of encryption is described above in fig 2. The encryptor core accepts a 128-bit plaintext input word, and generates a corresponding 128-bit ciphertext output word using 128-bit AES key. Multiple blocks in the encryption process are described below:-

a) **Initial Round**:- In this round the input data is XORed with the generated round key.

b) **Round 0 - Round 9**:- Each round comprises of multiple processes like byte substitution, shift row, mix column and key addition. The output of one round is input to other round. The output from round 9 is fed

to final round.

c) **Final Round**:- In this round data and key goes through three processes byte substitution, shift row and key addition respectively. The output of this round is 128-bit ciphertext output word.

Decryption Core:

Decryption process is described in fig 3. The decryption core provides the reverse function. It generates 128-bit plaintext from 128-bit ciphertext, using 128-bit AES key.

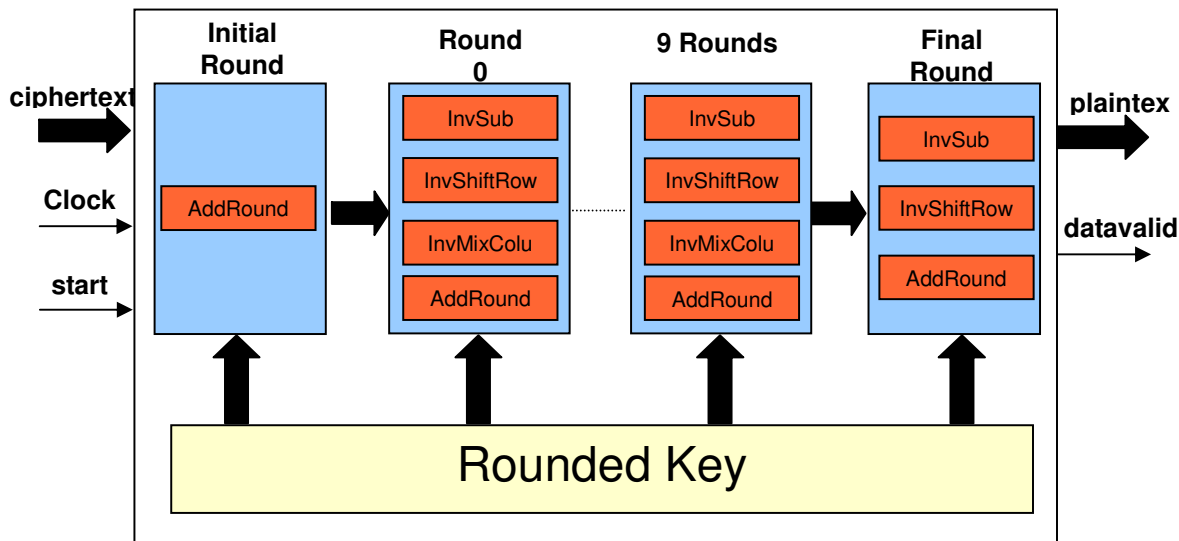


Fig 3 Decryption Block Diagram

Multiple blocks in the decryption process are described below:-

- a) **Initial Round**:- In this round the input data is XORed with the round key.
- b) **Round 0 - Round 9**:- Each round comprises of multiple processes like byte substitution, shift row, mix column and key addition. The output of one round is input to

other round. The output from round 9 is fed to final round.

- c) **Final Round**:- In this round data and key goes through three processes byte substitution, shift row and key addition respectively. The output of this round is 128-bit ciphertext output word.

Performance:

a) Sequential Architecture

Device	LUT's	Slice Count	Frequency(Mhz)
Virtex-4 (xc4vlx25-ff676)	2723	1639	283
Virtex-5 (xc5vlx30-ff676)	1088	306	350

b) Parallel Architecture

Device	LUT's	Slice Count	Frequency(Mhz)
Virtex-5 (xc5vlx30-ff676)	11052	3842	347MHz

Verification:

The AES controller module has been verified with following approaches:

- Exhaustive Functional/Timing simulation
- Prototyped on Xilinx Virtex 4 development board

Deliverables:

- Verilog RTL source code
- Test benches
- Synthesis and Simulation scripts
- Detailed user documentation, including RTL source code documentation