AES ENCRYPTION AND DECRYPTION IN FPGA

AES IMPLEMENTATION IN MODELSIM

AES is the acronym for Advanced Encryption Standard. This is the documentation of a project done in MODELSIM implementing a modified AES algorithm which uses a key of size 512 bits.

The following definitions are used throughout this standard:

AES Advanced Encryption Standard

Affine A transformation consisting of multiplication by a matrix followed by Transformation the addition of a vector.

Array An enumerated collection of identical entities (e.g., an array of bytes).

Bit A binary digit having a value of 0 or 1.

Block Sequence of binary bits that comprise the input, output, State, and Round Key. The length of a sequence is the number of bits it contains. Blocks are also interpreted as arrays of bytes.

Byte A group of eight bits that is treated either as a single entity or as an array of 8 individual bits.

Cipher Series of transformations that converts plaintext to ciphertext using the Cipher Key.

Cipher Key Secret, cryptographic key that is used by the Key Expansion routine to generate a set of Round Keys; can be pictured as a rectangular array of bytes, having four rows and Nk columns.

Ciphertext Data output from the Cipher or input to the Inverse Cipher.

Inverse Cipher Series of transformations that converts ciphertext to plaintext using the Cipher Key.

Key Expansion Routine used to generate a series of Round Keys from the Cipher Key.

Plaintext Data input to the Cipher or output from the Inverse Cipher.

Rijndael Cryptographic algorithm specified in this Advanced Encryption Standard (AES).

Round Key Round keys are values derived from the Cipher Key using the Key Expansion routine; they are applied to the State in the Cipher and Inverse Cipher.

State Intermediate Cipher result that can be pictured as a rectangular array of bytes, having four rows and Nb columns.

S-box Non-linear substitution table used in several byte substitution transformations and in the Key Expansion routine to perform a onefor-one substitution of a byte value.

Word A group of 32 bits that is treated either as a single entity or as an array of 4 bytes.

The following algorithm parameters, symbols, and functions are used throughout this standard:

AddRoundKey() Transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using an XOR operation. The length of a Round Key equals the size of the State (i.e., for Nb = 4, the Round Key length equals 128 bits/16 bytes).

InvMixColumns() Transformation in the Inverse Cipher that is the inverse of MixColumns().

InvShiftRows() Transformation in the Inverse Cipher that is the inverse of ShiftRows().

InvSubBytes() Transformation in the Inverse Cipher that is the inverse of SubBytes().

K Cipher Key.

MixColumns() Transformation in the Cipher that takes all of the columns of the State and mixes their data (independently of one another) to produce new columns.

Nb Number of columns (32-bit words) comprising the State. For this standard, Nb = 4. (Also see Sec. 6.3.)

Nk Number of 32-bit words comprising the Cipher Key. For this standard, Nk = 4, 6, or 8. (Also see Sec. 6.3.)

Nr Number of rounds, which is a function of Nk and Nb (which is fixed). For this standard, Nr = 10, 12, or 14. (Also see Sec. 6.3.)

Rcon[] The round constant word array.

RotWord() Function used in the Key Expansion routine that takes a four-byte word and performs a cyclic permutation.

ShiftRows() Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.

SubBytes() Transformation in the Cipher that processes the State using a nonlinear byte substitution table (S-box) that operates on each of the State bytes independently.

SubWord() Function used in the Key Expansion routine that takes a four-byte input word and applies an S-box to each of the four bytes to produce an output word.

XOR Exclusive-OR operation. ¯ Exclusive-OR operation. ˜ Multiplication of two polynomials (each with degree < 4) modulo x 4 + 1. • Finite field multiplication.

	Key Length *(Nk words)*	Block Size *(Nb words)*	Number of Rounds *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

KEY-BLOCK-ROUND COMBINATIONS

PSUEDOCODE FOR CIPHER

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Main Encryption Module

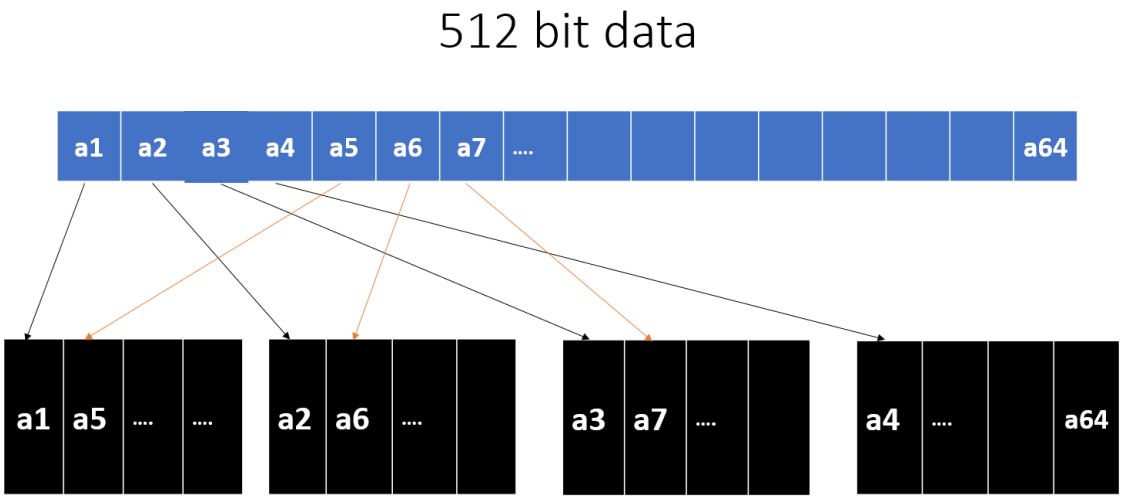
AES_ENCRYPTION_512 (DATA, CIPHER KEY, ENCRYPTED DATA)

Inputs: *Data (data to be encrypted)*

Cipher key (Key of length 512 bits using which the data is encrypted)

Outputs: *Encrypted data (Encrypted data output)*

This module takes 512-bit inputs of plain text and cipher key. The plain text is then partitioned into 4 blocks according to the scheme given below.



The basic purpose of mixing in this way is to ensure that even if the attacker cracks the first block, he will not be able to deduce anything

Now this first block is encrypted with the first 128 bits of the key, second block with second 128 bits of key and so on using standard AES algorithm.

Main Decryption Module

AES_decryption_512 (Encrypted Data, Cipher key, Data)

Inputs: *Encrypted Data (data to be decrypted)*

Cipher key (Key of length 512 bits using which the data is encrypted)

Outputs: *Decrypted data (decrypted data output)*

This module takes 512-bit inputs of encrypted data and cipher key.

The output is obtained by combining outputs in the given way.