AES User Guide

# Introduction

AES is a symmetrical block encryption algorithm that is currently the most popular in the world. AES provides obfuscation of plaintext into ciphertext, based on an associated encryption key.

The AES algorithm can be implemented with various alternative block cipher modes (ECB, CBC, OFB, CFB, CTR) to give additional layers of obscurity.

# AES CORE

The AES core is designed to implement 128, 192 and 256 bit key encryption using the AES algorithm.

The AES key type is selected by

|  |  |
| --- | --- |
| Key type | Value |
| 128 | “00” |
| 192 | “01” |
| 256 | “10” |

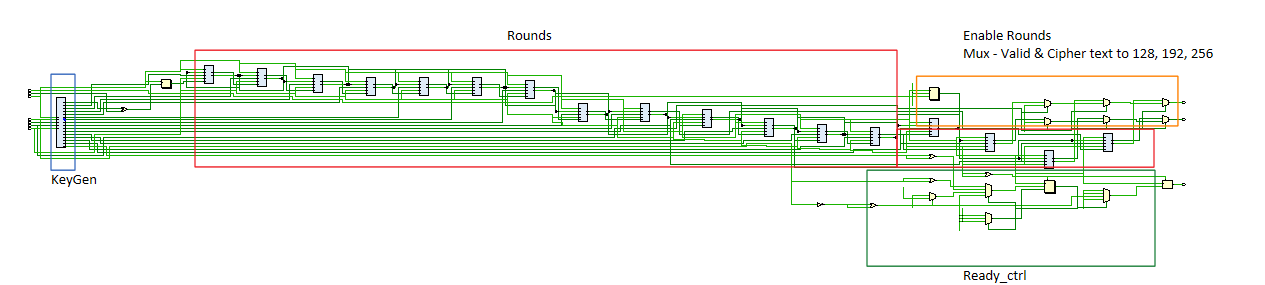
The encryption is initiated by asserting the “en”able pin to ‘1’. The user must insure that the ready pin is ‘1’ otherwise the enable will be ignored.

The AES CORE does the following

1. KeyExpansions — Derive the set of round keys from the cipher key
2. InitialRound - Initialize the state array with plaintext. Add the initial round key
   1. AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
3. Implement Rounds (keysize/32 + 6)-1
   1. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
   2. ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
   3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
   4. AddRoundKey
4. Final Round (no MixColumns)
   1. SubBytes
   2. ShiftRows
   3. AddRoundKey

Copy the final state array out as the encrypted data (ciphertext) & assert valid

See elaboration of AES CORE below



# How to implement the AES Core

The following snippet is an example of the testbench used to generate 128 and 256 key encryption.

|  |
| --- |
| en **<=** '1'**;**  plaintext **<=** x"00112233\_44556677\_8899aabb\_ccddeeff"**;**  key **<=** x"00000000\_00000000\_00000000\_00000000" **&** x"00010203\_04050607\_08090a0b\_0c0d0e0f"**;**  **wait** **until** **rising\_edge(**clk**);**  en **<=** '0'**;**  **wait** **until** **rising\_edge(**valid**);**  **wait** **for** 1 ns**;**  **report** "Plaintext = " **&** to\_hstring**(**plaintext**);**  **report** "128key = " **&** to\_hstring**(**key**);**  **report** "Ciphertext " **&** to\_hstring**(**ciphertext**);**  **report** "Expected : 69c4e0d8\_6a7b0430\_d8cdb780\_70b4c55a"**;**  **wait** **for** 100 ns**;**    **wait** **for** 1000 ns**;**  key\_type **<=** "10"**;**    en **<=** '1'**;**  plaintext **<=** x"3243f6a8885a308d313198a2e0370734"**;**  key **<=** x"2b7e151628aed2a6abf7158809cf4f3c\_762e7160f38b4da56a784d9045190cfe"**;**  **wait** **until** **rising\_edge(**clk**);**  en **<=** '0'**;**  **wait** **until** **rising\_edge(**valid**);**  **wait** **for** 1 ns**;**  **report** "Plaintext = " **&** to\_hstring**(**plaintext**);**  **report** "256key = " **&** to\_hstring**(**key**);**  **report** "Ciphertext " **&** to\_hstring**(**ciphertext**);**  **report** "Expected : 1a6e6c2c\_662e7da6\_501ffb62\_bc9e93f3"**;**  **wait** **for** 1000 ns**;**  **report** "" **severity** failure**;** |

The keytype, plaintext and en signal are used to initiate the AES encryption. The resulting valid signal informs the user the block has been encrypted.

128 – 860 ns for keygen

640ns/period for 128 – 32 clk cycles from ready

256 key setup 1100ns for keygen

1120 for 256 - 52 clk cycles from ready

20 ns a period