ChipCraft®

Datasheet

CC-AES-AXI Documentation

1.1

Scope

This document describes the CC-AES-AXI IP core. Module features and configuration registers are described. The document contains integration guide that covers synthesis options and instantiation example for easy implementation in customer's environment.

European Funds
Smart Growth

NaviSoC

European Union
European Regional
Development Fund

Contents

1. AES Module

1.1 Functionality

- Encryption and decryption operation,
- ECB mode only,
- 128 bit data block,
- programmable 128, 192 and 256 bit key length,
- interrupt flag for calculation and key expansion operation.

1.2 Overview

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm established by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES module implements AES 128, 192 and 256 algorithms specified in the "Advanced Encryption Standard (AES)" document announced as Federal Information Processing Standards (FIPS) Publication 197. The module contains both encryption and decryption datapath. Key expansion is performed on-the-fly, requiring only initial decryption key calculation.

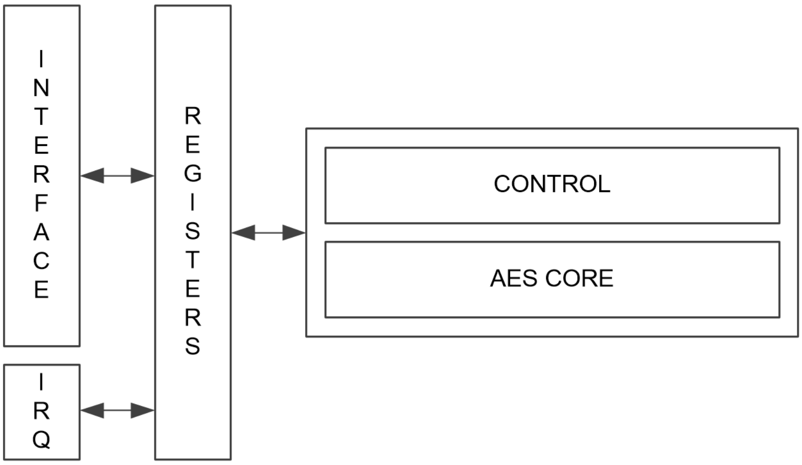

Figure 1.1. AES block diagram.

Figure 1.1 presents the block diagram of the AES module. It is composed of configuration registers and main AES core.

1.2.1 Throughput

Table 1.1 presents the number of clock cycles required to perform decryption key initialization. This operation is required every time the key changes. Table 1.2 presents the AES module throughput for different key lengths.

Table 1.1. Decryption key initialization

Key length	APB cycles	AES cycles	Sum
128-bit	20	13	33
192-bit	28	15	43
256-bit	36	17	53

Table 1.2. Encryption/decryption throughput

Key length	APB cycles	AES cycles	Sum	Bits/cycle
128-bit	36	13	49	2.61
192-bit	36	15	51	2.51
256-bit	36	17	53	2.42

1.3 Interrupts

The AES module has one interrupt source.

1.3.1 AES Interrupt

The AES Interrupt is signalized by IF flag in the STATUS register (1.4.7). AES interrupt occurs when key expansion is completed or data calculation is completed, depending on current command executed. The interrupt flag is cleared after writing 1 to IC bit in Command Register (independently of other command executed or even without any other command execution) (1.4.6).

1.4 Configuration Registers

1.4.1 Registers List

The core is controlled through registers mapped into memory address space. Not implemented locations are read as zeros.

Address Offset	Register	Name
0x00-0x1C	KEY	Key Registers
0x20-0x2C	INPUT	Data Input Registers
0x30-0x3C	OUTPUT	Data Output Registers
0x40	CTRL	Control Register
0x44	CMD	Command Register
0x48	STATUS	Status Register

1.4.2 Key Registers

Address: 0x00

31	0
KEY[31:0]	
R/W	
0	

Address: 0x04

31	0
KEY[63:32]	
R/W	
0	

Address: 0x08

31	0
KEY[95:64]	
R/W	
0	

Address: 0x0C

31	0
KEY[127:96]	
R/W	
0	

Address: 0x10

31	0
KEY[159:128]	
R/W	
0	

Address: 0x14

31	0
KEY[191:160]	
R/W	
0	

Address: 0x18

31	0
KEY[223:192]	
R/W	
0	

Address: 0x1C

31	0
KEY[255:224]	
R/W	
0	

KEY[255:0] *AES Key*

For Key Length of:

128 Key Registers 0x00 - 0x0C will be used.

192 Key Registers 0x00 - 0x14 will be used.

256 Key Registers 0x00 - 0x1C will be used.

However it is recommended to always write all registers, unused with 0 value. For example when using 192 bits length key registers 0x18 and 0x1C should be set to zeros. Write operation to any of Key Registers will set KEY_READY bit in Status register to 0 until Key Expansion Command will be executed.

1.4.3 Data Input Registers

Address: 0x20

31	0
INPUT[31:0]	
R/W	
0	

Address: 0x24

31	0
INPUT[63:32]	
R/W	
0	

Address: 0x28

31	0
INPUT[95:64]	
R/W	
0	

Address: 0x2C

31	0
INPUT[127:96]	
R/W	
0	

INPUT[127:0] *Input Data*

Read/write registers for input data.

1.4.4 Data Output Registers

Address: 0x30

31 … 0
OUTPUT[31:0]
R/W
0

Address: 0x34

31 … 0
OUTPUT[63:32]
R/W
0

Address: 0x38

31 … 0
OUTPUT[95:64]
R/W
0

Address: 0x3C

31 … 0
OUTPUT[127:96]
R/W
0

OUTPUT[127:0] *Output Data*

Read only registers for output data.

1.4.5 Control Register

Address: 0x40

31	30	...	...	...	...	9	8
		...	...	...	...		
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
				IE	KEY_LEN[1:0]		DECRYPT
R	R	R	R	R/W	R/W		R/W
0	0	0	0	0	0		0

DECRYPT *AES Operation*

0 AES encryption mode.

1 AES decryption mode.

KEY_LEN[1:0] *AES Key Length*

00 Key length is set to 128 bit. Key registers 0x00 - 0x03 will be used.

01 Key length is set to 192 bit. Key registers 0x00 - 0x05 will be used.

10 Key length is set to 256 bit. Key registers 0x00 - 0x07 will be used.

11 Restricted.

Any change to key length requires key expansion command to be executed for proper AES calculation. Any change to key length will set KEY_READY bit in Status Register to 0 until new key expansion command will be executed.

IE *AES Interrupt Enable*

0 Interrupt disabled.

1 Interrupt enabled.

1.4.6 Command Register

Address: 0x44

31	30	...	...	...	...	9	8
		...	...	...	...		
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
					IC	EXPAND	START
W	W	W	W	W	W	W	W
0	0	0	0	0	0	0	0

START *Start AES Encryption/Decryption*

Writing 1 to this bit will execute AES calculations - encryption or decryption depending on DECRYPT bit in Control Register. This bit has lower priority than KEY_EXP - if both are set Key Expansion Command will be executed.

EXPAND *AES Key Expansion*

Writing 1 to this bit will execute Key Expansion Command. This bit has higher priority than START - if both are set Key Expansion Command will be executed.

IC *AES Interrupt Flag Clear*

Writing 1 to this bit will clear Interrupt Flag in Status register. This bit is indepentent of other bits in Command Register. Regardless of other bits and if any command is executed Interrupt Flag will be cleared if 1 is written to this bit.

1.4.7 Status Register

Address: 0x48

31	30	...	...	...	...	9	8
		...	...	...	...		
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
					KEY_READY	IF	BUSY
R	R	R	R	R	R	R	R
0	0	0	0	0	0	0	0

BUSY *AES Operation*

0 AES module is ready for calculations.

1 AES module is performing calculations (either Key Expansion, Encryption or Decryption).

IMPORTANT NOTE: When AES is busy only read operations on host interface will be executed. Any write operation will be held in WAIT state until AES completes calculations and BUSY is set to 0.

IF *AES Interrupt Flag*

0 AES module calculations (either Key Expansion, Encryption or Decryption) was not performed, hasn't finished or Interrupt flag was cleared.

1 AES module calculations (either Key Expansion, Encryption or Decryption) was finished and Interrupt Flag was not cleared.

For Interrupt Flag to be set value 1 in IE bit in Control Register is required.

KEY_READY *AES Key Ready*

0 Key registers or KEY_LEN was modified without Key Expansion command. Decryption operation output can be corrupted. Key Expansion Command should be executed.

1 Key expansion was performed. AES module is ready for calculations.

1.5 Implementation

1.5.1 Design Structure

The synthesible RTL IP core part (*AXI/rtl* and *AES/rtl* folder) utilizes Verilog 2005 HDL. The testbench part (*AXI/tb* and *AES/tb* folder) uses SystemVerilog language.

```
AXI
├── rtl
│   └── AXI_PERIPH
│       └── amba_axilite_apb_bridge.v
└── tb
    └── AXI_PERIPH
        ├── APB
        │   └── virtual_APB_slave.sv
        ├── AXI
        │   └── tb_amba_axilite_tasks.sv
        ├── common
        │   └── timescale.v
        ├── run
        │   └── ncvlog_amba_axilite_apb_bridge.sh
        ├── tests
        │   └── tb_read_write_test.sv
        └── tb_amba_axilite_apb_bridge.sv
AES
├── rtl
│   ├── aes_top_apb.v
│   ├── aes_top_axilite.v
│   ├── aes_apb.v
│   ├── aes_core.v
│   └── aes_includes.v
└── tb
    ├── tasks
    │   ├── tb_aes_top_tasks_includes.v
    │   └── tb_aes_top_tests_includes.v
    ├── run
    │   └── ncvlog_apb_aes.sh
    └── FIPS_vectors
        ├── test_kat_gfsbox_128_ciphertext.txt
        ├── test_kat_gfsbox_128_plaintext.txt
        ├── test_kat_gfsbox_192_ciphertext.txt
        ├── test_kat_gfsbox_192_plaintext.txt
        ├── test_kat_gfsbox_256_ciphertext.txt
        ├── test_kat_gfsbox_256_plaintext.txt
        ├── test_kat_keysbox_128_ciphertext.txt
        ├── test_kat_keysbox_128_key.txt
        ├── test_kat_keysbox_192_ciphertext.txt
        ├── test_kat_keysbox_192_key.txt
        ├── test_kat_keysbox_256_ciphertext.txt
        ├── test_kat_keysbox_256_key.txt
        ├── test_kat_varkey_128_ciphertext.txt
        ├── test_kat_varkey_128_key.txt
        ├── test_kat_varkey_192_ciphertext.txt
        ├── test_kat_varkey_192_key.txt
        └── test_kat_varkey_256_ciphertext.txt
```

```
│   ├── test_kat_varkey_256_key.txt
│   ├── test_kat_vartxt_128_ciphertext.txt
│   ├── test_kat_vartxt_128_key.txt
│   ├── test_kat_vartxt_192_ciphertext.txt
│   ├── test_kat_vartxt_192_key.txt
│   ├── test_kat_vartxt_256_ciphertext.txt
│   └── test_kat_vartxt_256_key.txt
└── tb_aes_top.sv
```

1.5.2 Simulation Flow

The IP Core is provided with self-checking testbench to verify the correct operation of the IP prior to use in a design. The testbench is divided into two environments. The first one tests the aes_top_apb module. Self-checking testbench includes KAT tests recommended by NIST for AES verification. To run the simulation using Cadence® Incisive® Enterprise Simulator run *ncvlog_apb_aes.sh* script located in *AES/tb/run* folder. The simulation should end with reporting no errors. The second environment tests the AXI4-Lite to APB3 converter. To run the simulation using Cadence® Incisive® Enterprise Simulator run *ncvlog_amba_axilite_apb_bridge.sh* script located in the *AXI/tb/AXI_PERIPH/run* folder. The aes_top_axilite top module is composed of the aes_top_apb core and the amba_axilite_apb_bridge AXI4-Lite to APB3 converter.

1.5.3 Clock and Reset

The CC-AES-AXI utilizes a fully synchronous design with one positive edge clocking domain and negative asynchronous reset assertion. External reset synchronizer has to be used to ensure synchronous reset deassertion.

1.5.4 Constraints

In most cases only module output ports are registered. Therefore, it is a good practice to reserve the entire clock cycle for module inputs combinational logic and set minimal input delay (*set_input_delay* command). Registered outputs leave the entire clock cycle for external logic (*set_output_delay* command).

1.5.5 Configuration Options

The table below shows the generic parameters of the core.

Generic name	Description	Range	Default
DECRYPTION_PATH	Configure AES core decryption path. Set to zero if decryption is not needed and save about 40% area.	0,1	1

1.5.6 Signals Description

Signal name	Description	I/O	Active	Type
ACLK	Synchronous clock	I	rising	clock
ARESETn	Asynchronous reset	I	low	reset
AWADDR[6:2]	AXI4-Lite write address	I	data	comb.
AWPROT[2:0][1]	AXI4-Lite write address protection type	I	data	comb.
AWVALID	AXI4-Lite write address valid	I	high	comb.
AWREADY	AXI4-Lite write address ready	O	high	reg.
WDATA[31:0]	AXI4-Lite write data	I	data	comb.
WSTRB[3:0]	AXI4-Lite write strobe	I	high	comb.
WVALID	AXI4-Lite write valid	I	high	comb.
WREADY	AXI4-Lite write ready	O	high	reg.
BRESP[1:0]	AXI4-Lite write response	O	data	reg.
BVALID	AXI4-Lite write response valid	O	high	reg.
BREADY	AXI4-Lite write respnse ready	I	high	comb.
ARADDR[6:2]	AXI4-Lite read address	I	data	comb.
ARPROT[2:0][1]	AXI4-Lite read address protection type	I	data	comb.
ARVALID	AXI4-Lite read address valid	I	high	comb.
ARREADY	AXI4-Lite read address ready	O	high	reg.
RDATA[31:0]	AXI4-Lite read data	O	data	reg.
RRESP[1:0]	AXI4-Lite read response	O	data	reg.
RVALID	AXI4-Lite read valid	O	high	reg.
RREADY	AXI4-Lite read ready	I	high	comb.
interrupt	AES interrupt	O	high	reg.
busy	AES busy	O	high	reg.

[1] Signal is not used in the design.

1.5.7 Instantiation

```
icg
icg_aes_u (
    .E(ARVALID|AWVALID|aes_busy),
    .clk(ACLK),
    .gclk(aes_clk),
    .scan_enable(scan_enable));

aes_top_axilite
aes_top_axilite_u (
    .ACLK(aes_clk),
    .ARESETn(ARESETn),
    .AWADDR(AWADDR[6:2]),
    .AWPROT(AWPROT),
    .AWVALID(AWVALID),
    .AWREADY(AWREADY),
    .WDATA(WDATA),
    .WSTRB(WSTRB),
    .WVALID(WVALID),
    .WREADY(WREADY),
    .BRESP(BRESP),
    .BVALID(BVALID),
    .BREADY(BREADY),
    .ARADDR(ARADDR[6:2]),
    .ARPROT(ARPROT),
    .ARVALID(ARVALID),
    .ARREADY(ARREADY),
    .RDATA(RDATA),
    .RRESP(RRESP),
    .RVALID(RVALID),
    .RREADY(RREADY),
    .interrupt(aes_interrupt),
    .busy(aes_busy));
```

1.6 Revision History

Doc. Rev.	Date	Comments
1.1	11-2018	Editorial corrections in 1.5.7 Instantiation section.
1.0	10-2018	First Issue.

ChipCraft Sp. z o.o. Dobrzańskiego 3 lok. BS073, 20-262 Lublin, POLAND www.chipcraft-ic.com

ⓒ2018 ChipCraft Sp. z o.o. CC-AES-AXI-Doc_112018.

ChipCraft®, ChipCraft logo and combination of thereof are registered trademarks or trademarks of ChipCraft Sp. z o.o. All other names are the property of their respective owners.

Disclaimer: ChipCraft makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. ChipCraft does not make any commitment to update the information contained herein.