AES 128/192/256 (ECB)
AVALON®-MM SLAVE

Thomas Ruschival
and opencores.org

www.opencores.org

ruschi@opencores.org
Contents

1 Introduction 3

2 Interface 3

2.1 Configuration Generics ................................. 3
2.2 Signals ............................................... 4

3 Memory Map 4

3.1 Control Register ...................................... 5

4 Protocol Sequence 6

4.1 Interrupt Behavior .................................... 7

5 The Inner Core 7

6 Throughput Calculation 10

7 FPGA implementations 11

8 Simulation 12

8.1 Testbench ............................................. 12
8.2 Simulation .............................................. 12

9 Software Driver 12

9.1 Configuration ......................................... 12

10 License and Liability 13
1 Introduction

The Advanced Encryption Standard (AES) is a symmetric block cipher operating on fixed block sizes of 128 Bit and is specified for key sizes of 128, 192 and 256 Bit designed by Joan Daemen and Vincent Rijmen. The algorithm was standardized by National Institute of Standards and Technology (NIST). For more information on the algorithm see [1].

This component implements an AES encryption decryption data path in Electronic Codebook (ECB) mode with either 128,192 or 256 Bit keys. The key length is determined by generics at compile time. Also the decryption data path can be disabled by generics if it is not needed for the application.

The component provides an Avalon® Memory Mapped (Avalon-MM) slave interface to connect to an Altera® Avalon® switch fabric. The Avalon® interface is implemented in a way that it can also be used to connect to a Whishbone master if the signals are correctly mapped, see [2]. For further information about the Whishbone bus refer to [3].

2 Interface

The AES core is accessed by the interface described in this section. An Avalon® interface was chosen for its simplicity and compatibility with wishbone. Furthermore Avalon® defines interrupt request signals for slaves which would be separate signals in a Wishbone implementation. The component can be used both in polling mode or can provide an interrupt for signalling.

Unfortunately Avalon® is an Altera® proprietary technology. The actual AES core however is a self contained entity and can be embedded into other System on Chip (SoC) bus interfaces as well or used independently.

2.1 Configuration Generics

The AES core can be configured by generics shown in table 1, consequently they are provided by the Avalon® interface.

<table>
<thead>
<tr>
<th>Generic name</th>
<th>type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYLENGTH</td>
<td>NATURAL</td>
<td>Size of initial user key. Must be 128, 192 or 256.</td>
</tr>
<tr>
<td>DECRIPTION</td>
<td>BOOLEAN</td>
<td>Enables the instantiation of the decrypt data path if true.</td>
</tr>
</tbody>
</table>

Table 1: Component generics

Note: KEYLENGTH of 192 fail synthesis with Xilinx ISE® because of division by 6 in key schedule that cannot be mapped to shift operations (keyexpansion.vhd).

1 All other values raise a compilation failure.
2.2 Signals

The Avalon® MM Slave interface is described in [4], the component implements the signals shown in table 2.2. All signals are synchronous, sampled at the rising edge of the clock. The type for all signals is IEEE1164 std_logic or std_logic_vector. For signals wider than 1 Bit the range is Most Significant Bit (MSB) down to Least Significant Bit (LSB).

This component has only output signals driven by registers, no input signals are directly combinatorially connected to the output signals, thus combinational loops are avoided. All signals are active high. This component does not support burst transfers.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Width</th>
<th>In/Out</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>1</td>
<td>in</td>
<td>Avalon® bus clock, also used to drive the core.</td>
</tr>
<tr>
<td>reset</td>
<td>1</td>
<td>in</td>
<td>Synchronous reset signal for Avalon® bus interface. The core itself is designed without need for reset signals.</td>
</tr>
<tr>
<td>writedata</td>
<td>32</td>
<td>in</td>
<td>Input data to write to location designated by address. Bit 31 is most significant Bit.</td>
</tr>
<tr>
<td>address</td>
<td>5</td>
<td>in</td>
<td>Word offset to the component's base address. The memory map of the component for the respective offset is described in [3]. Only full 32-Bit words can be addressed no byte addressing is implemented.</td>
</tr>
<tr>
<td>write¹</td>
<td>1</td>
<td>in</td>
<td>If asserted enable write of data at writedata to location designated by address.</td>
</tr>
<tr>
<td>read¹</td>
<td>1</td>
<td>in</td>
<td>If asserted output data at location designated by address to readdata.</td>
</tr>
<tr>
<td>readdata</td>
<td>32</td>
<td>out</td>
<td>Data output port for reading data at the location defined by address. Bit 31 is most significant Bit.</td>
</tr>
<tr>
<td>waitrequest</td>
<td>1</td>
<td>out</td>
<td>Asserted if writedata was not accepted, this is the case if the key expansion is not yet complete and a new is written to the KEY address range without previous de-assertion of the KEY_VALID Bit</td>
</tr>
<tr>
<td>irq</td>
<td>1</td>
<td>out</td>
<td>If Interrupt behavior is enabled IRQ will be asserted when the operation has terminated. For use of interrupt see [4.1]</td>
</tr>
</tbody>
</table>

Table 2: Avalon® Bus interface signals

3 Memory Map

The AES core Avalon® slave has an address space of 31 words accessible through the offset described by the signal address, see 2.2. This address space is divided into three main sections for the 4-word input data, the 4-word result of the operation and the user key. The actual length of the user key can vary between 4, 6 and 8 words depending on the keysize. For control signals and status information of the component and a control word is provided. The memory mapping

¹read and write are mutually exclusive and must not be asserted simultaneously.
is described in table 3

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00–0x07</td>
<td>KEY</td>
<td>Initial user key that will be used for encryption and decryption. The most significant word is written to offset 0x00. This memory section is write-only to the Avalon® interface.</td>
</tr>
<tr>
<td>0x08–0x0B</td>
<td>DATA</td>
<td>Input data, can be either interpreted as cyphertext for decryption or plain text for encryption. The most significant word shall be written to offset 0x08. This memory section is write-only to the Avalon® interface.</td>
</tr>
<tr>
<td>0x10–0x13</td>
<td>RESULT</td>
<td>Result of the operation. The most significant word of the result at offset 0x10. This memory section is read-only to the Avalon® Interface.</td>
</tr>
<tr>
<td>0x14–0x1E</td>
<td>—</td>
<td>reserved</td>
</tr>
<tr>
<td>0xF</td>
<td>CTRL</td>
<td>Control and status word of the component can be read and written. Detailed description see §3.1</td>
</tr>
</tbody>
</table>

Table 3: Memory map of the AES core Avalon® slave

### 3.1 Control Register

The AES Core offers the register CTRL to control the function of the core and poll its status. The control register can be accessed in read and write mode. When writing to the register reserved Bits shall be assigned a value of 0. Individual Bits have following functionality described in table §3.1

In case of an Avalon® Bus reset this register is set to 0x00000000 thus invalidating all previously written keys and resetting the AES core.
### 4 Protocol Sequence

The AES component appears as memory mapped peripheral. All writes are fundamental slave write transfers, see [4] and take one clock cycle of the Avalon® bus clock \( \text{clk} \). It is not necessary to write all words of a input parameter successively or in one transfer. Bursts are not supported.

Before any AES operation can be started the initial user key has to be written to \( \text{KEY} \) segment of the memory map. After the user key is transferred to the component the \( \text{KEY}_\text{VALID} \) Bit must be set to start the key expansion. This Bit can be set simultaneously with \( \text{DEC} \) or \( \text{ENC} \) Bit of the control register. To invalidate the previous key and use another key the \( \text{KEY}_\text{VALID} \) must be deasserted for at least one Avalon® bus clock cycle During this cycle the new key can already be transferred.

Once a key is passed and marked valid data blocks can be transferred to the \( \text{DATA} \) segment of the memory map. The AES operation is started by asserting the \( \text{ENC} \) Bit for encryption or \( \text{DEC} \) Bit for decryption. While asserting \( \text{ENC} \) or \( \text{DEC} \) the \( \text{KEY}_\text{VALID} \) Bit must be kept asserted. The \( \text{ENC} \) or \( \text{DEC} \) Bit respectively is deasserted by the component after completing the requested operation. The result of the operation can be read from the \( \text{RESULT} \) area of the memory and is not cleared. It will be overwritten by succeeding operations.

The underlying AES core uses the Finite State Machine (FSM) shown in [1] for processing of the data. The signals \( \text{data}_\text{stable} \) and \( \text{key}_\text{stable} \) are accessible over the control status word \( \text{CTRL} \) [3] \( \text{key}_\text{ready} \) is a signal driven by the key generator when all keys are expanded. The signal

---

\[1\] ENC and DEC are mutually exclusive and must not be asserted simultaneously.
round_index is the counter for the rounds and the address to select a round key. NO_ROUNDS is the total number of rounds the processing takes, a constant defined by the generic KEYLENGTH. The AES standard in[1] defines 10 rounds for 128 Bit key, 12 rounds for a 192 Bit key and 14 rounds for a 256 Bit key. Thus depending on the key length the processing of a data block needs at maximum 15 clock cycles from data_stable=1 to completion, if the key is already expanded.

![Finite State Machine of encryption and decryption process](image)

**Figure 1:** Finite State Machine of encryption and decryption process

### 4.1 Interrupt Behavior

By setting IRQ_ENA in the control register the component is configured to issue interrupt requests. If IRQ_ENA is asserted the interrupt request IRQ 2.2 will be set when the computation has completed in addition to clearing the ENC or DEC Bit. The IRQ 2.2 signal will remain set until clearing IRQ_ENA or a read operation on the RESULT area of the components address range.

### 5 The Inner Core

The algorithmic core is divided into two separate data paths one for encryption and a second for decryption operation. The two data paths are independent, however they share the key expansion component which provides decrypt and encrypt keys (which are the same only in opposite
order). Each data path is controlled by its own FSM if configured by the generic DECRYPTION 2.1
the decryption data path is included and some multiplexers are generated for the shared signals,
e.g. result or roundkey_index.
For reference the encryption data path of aes_core.vhd is given in figure 2. The decryption data
path is left for the reader or any other author of this document.
Figure 2: Encrypt data path of the AES core as implemented in aes_core.vhd
6 Throughput Calculation

The Avalon® interface communicates a 32-Bit DWORD per clock cycle. Therefore a key is transmitted in 4 to 8 cycles plus one cycle to activate keyexpansion with the control word 3.1. A payload data block or the result consist always of 4 DWORDs, thus it takes 4 cycles to send data to the core, one cycle to activate the computation with the control register 3.1 and 4 cycles to retrieve the data.

The keyexpansion component computes one column of a roundkey each clock cycle. AES takes, depending on the key length, 10, 12 or 14 roundkeys with each 4 columns, see [1]. The keyexpansion therefore takes 40, 48 or 56 cycles until the encryption or decryption can start. The roundkeys are stored until invalidated, see [4] thus this step is only needed once after power-up until the key changes.

The AES core computes one iteration (round) of the Rijndael-Algorithm each clock cycle, thus a 128 Bit data block is encrypted or decrypted in 10, 12 or 14 cycles plus an initial round.

The maximum throughput $T_{max}[\text{Bits}]$ depends on the maximum operation frequency $f_{max}$ and the key length which influences the number of rounds $N_{rnd} \in \{10, 12, 14\}$.

$$T_{max} = \frac{(1 + N_{rnd}) \cdot 128\text{Bit}}{f_{max}}$$

(1)

Note: Equation [1] assumes that the roundkeys are already generated and does not include the constant of 4+1+4 Avalon® bus cycles for transmission of data, activation and result retrieval.
7 FPGA implementations

The component has only been implemented and tested on an Altera® Cyclone-II EP2C35 FPGA. For this setup a Makefile is provided in ./sys/Altera_Quartus9.1. All other values in the table are only results of synthesis and are not verified on actual hardware.

The design is kept vendor independent in generic VHDL. AES SubByte component is specially designed using M4K block RAM as dual-port ROM. For non-Altera® FPGAs a second VHDL architecture exists also trying to make use of ROM functions of the target chips however the success varies on RTL compiler capabilities. Later versions of Altera® Quartus-II show the same results whether M4K blocks are used or the generic version in selected.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Target FPGA¹</th>
<th>LE / Slices</th>
<th>HW RAM</th>
<th>f&lt;sub&gt;max&lt;/sub&gt;[Mhz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 Bit Key, encrypt + decrypt</td>
<td>Xilinx® Spartan3A XC3S1400A-5FG484</td>
<td>- / 1609</td>
<td>18 RAMB16BWE</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>Xilinx® Virtex5 XC5VLX30-3FF324</td>
<td>- / 297</td>
<td>18 18k-Blocks 4 36k-Blocks</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td>Altera® Cyclone-II EP2C35F484C8</td>
<td>1937 / -</td>
<td>39912 Bits in 22 M4K-Blocks</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Altera® StratixII EP2S30F484C5</td>
<td>585 / -</td>
<td>39912 Bits in 22 M4K-Blocks</td>
<td>103</td>
</tr>
<tr>
<td>128 Bit Key, encrypt + decrypt</td>
<td>Xilinx® Spartan3A XC3S1400A-5FG484</td>
<td>- / 1523</td>
<td>18 RAMB16BWE</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>Altera® Cyclone-II EP2C35F484C8</td>
<td>1776 / -</td>
<td>39912 Bits in 22 M4K-Blocks</td>
<td>65</td>
</tr>
<tr>
<td>256 Bit Key, encrypt</td>
<td>Xilinx® Spartan3A XC3S1400A-5FG484</td>
<td>- / 680</td>
<td>14 RAMB16BWE</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>Xilinx® Virtex5 XC5VLX30-3FF324</td>
<td>- / 297</td>
<td>10 18k-Blocks 4 36k-Blocks</td>
<td>268</td>
</tr>
<tr>
<td></td>
<td>Altera® Cyclone-II EP2C35F484C8</td>
<td>969 / -</td>
<td>22528 Bits in 14 M4K</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>Altera® StratixII EP2S30F484C5</td>
<td>524 / -</td>
<td>22528 Bits in 14 M4K</td>
<td>145</td>
</tr>
<tr>
<td>128 Bit Key, encrypt</td>
<td>Xilinx® Spartan3A XC3S1400A-5FG484</td>
<td>- / 594</td>
<td>14 RAMB16BWE</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>Altera® Cyclone-II EP2C35F484C8</td>
<td>797 / -</td>
<td>22528 Bits in 14 M4K</td>
<td>95</td>
</tr>
</tbody>
</table>

Table 5: resource usage on different targets and configuration

All configurations in table use hardware key expansion. Downloading of software generated roundkeys is not yet supported. The decryption and encryption data paths share a common keyexpansion block, multiplexing the address signals is one of the main reasons for regression of the maximum frequency f<sub>max</sub> of the configuration compared to encryption only versions.

⁰Synthesized with Altera® Quartus-II® Web edition Version 9.1 or Xilinx® ISE 9.1 Webpack
¹This table is not meant to be a benchmark between FPGAs of different vendors, it is only a rough estimation for the user of the core. The FPGA families cannot be compared easily, see also [5] and [6] for further details.
8 Simulation

8.1 Testbench

In ./bench/VHDL/ a "self-checking testbench" is provided which runs tests for a default TESTKEYSIZE is 256 Bit. For different key lengths the constant TESTKEYSIZE has to be changed appropriately. Expected results for all test cases and key lengths are included. The expected results were generated by AES Calculator applet, written by Lawrie Brown from ADFA, Canberra Australia [7]. The testbench consists of a sequence of 5 test cases:

1. load key1, load data1, encrypt : (basic encryption test)
2. key1, data1, decrypt: (basic decryption test)
3. key1, data1, encrypt: (test if internal state was changed)
4. key1, data2, encrypt: (encryption test with new data)
5. key2, data2, encrypt: (encryption test with new key)

8.2 Simulation

The component library is “avs_aes_lib”. All files are expected to be compiled into this library as all files depend at least on the package avs_aes_lib.avs_aes_pkg.

A Makefile for Mentor Graphics® Modelsim® is given in ./sim/. The default make target simaes will create the library “avs_aes_lib” and a “work” library, compile all files and run a testbench.

9 Software Driver

This AES Core Avalon® slave was also tested on a NiosII® processor. To use it in software a simple driver is provided in ./sw/ among with an example program of the basic usage. The driver consist of the two files avs_aes.c and avs_aes.h. Find more detailed description in the doxygen documentation in ./doc/sw/html.

9.1 Configuration

To be adapted to different address mappings and key sizes two macros are use in avs_aes.h:

<table>
<thead>
<tr>
<th>define</th>
<th>default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYWORDS</td>
<td>8</td>
<td>Key size in 32 Bit words</td>
</tr>
<tr>
<td>AES_BASEADDR</td>
<td>0x40000</td>
<td>Base address at which the AES Core is mapped to the Avalon® switch-fabric</td>
</tr>
</tbody>
</table>

Table 6: user changeable macros in header
10 License and Liability

The “AES 128/192/256 (ECB) Avalon®-MM Slave” component, all its subcomponents and documentation (like this document you are reading) are published under following license:

Copyright (c) 2009, Thomas Ruschival - All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

• Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

• Neither the name of the organization nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Note: The term “SOFTWARE” in the above licence applies in this case not only to software as executable code but also to documentation, hardware description or compiled netlists for actual target hardware. As Chips generally don’t just reproduce “the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution” the data sheet of the product must also contain it.

Altera, Cyclone-II, Stratix-II, Quartus, NIOS and Avalon are registered trademarks of the Altera Corporation 101 Innovation Drive, San Jose CA USA
Xilinx, Spartan3A and Virtex5 are registered trademarks of Xilinx Inc. 2100 Logic Drive, San Jose CA USA
Mentor Graphics and ModelSim are registered trademarks of Mentor Graphics Corporation 8005 SW Boeckman Road, Wilsonville OR USA
List of Acronyms

Advanced Encryption Standard (AES)
NIST approved symmetric block cypher

Electronic Codebook (ECB)
application of a cypher algorithm without further processing of the blocks

Finite State Machine (FSM)
Behavioural Model with finite number of states and transitions

Least Significant Bit (LSB)
least value bit in a vector

Most Significant Bit (MSB)
highest value bit in a vector

National Institute of Standards and Technology (NIST)
US standardisation office

System on Chip (SoC)
System of separate functional interacting together implemented on a single chip

Glossary

Bit
Binary Digit, atomary information unit

Byte
String of Bits - nowadays mostly a string of 8 Bits, also called oktett

Master
Entity initiating and controlling communication.

memory mapped
Method of addressing peripheral components like Avalon Slaves via the same address bus as main memory

Slave
Entity responding to communication requests by a Master.

switch fabric
Interconnect between IP-Cores providing arbitration and glue logic. Altera® Avalon® term
References


Change History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Chapter</th>
<th>Description</th>
<th>Date</th>
<th>Reviewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>all</td>
<td>initial document</td>
<td>2009/02/01</td>
<td>T. Ruschival</td>
</tr>
<tr>
<td>0.2</td>
<td>all</td>
<td>added interrupt</td>
<td>2009/03/25</td>
<td>T. Ruschival</td>
</tr>
<tr>
<td>0.3</td>
<td>all</td>
<td>added generics</td>
<td>2009/04/20</td>
<td>T. Ruschival</td>
</tr>
<tr>
<td>0.4</td>
<td>all</td>
<td>cleanup for opencores.org</td>
<td>2009/05/20</td>
<td>T. Ruschival</td>
</tr>
<tr>
<td>0.5</td>
<td>all</td>
<td>final release</td>
<td>2010/03/07</td>
<td>T. Ruschival</td>
</tr>
<tr>
<td>0.6</td>
<td>3,6</td>
<td>fixed memory map, added test-bench description</td>
<td>2010/04/02</td>
<td>T. Ruschival</td>
</tr>
<tr>
<td>0.7</td>
<td>3,6</td>
<td>fixed typos</td>
<td>2010/04/03</td>
<td>T. Ruschival</td>
</tr>
</tbody>
</table>