HardAES Data Sheet

DPA Resistant FIPS 197 Implementation

EnforcIT® Version 5.0, July 1, 2014

Microsemi Corporation Security Solutions
1281 Win Hentschel Blvd
West Lafayette, IN 47906
Phone: (765) 775-1800
Fax: (765) 775-1700

For help, please email us at:
support@microsemi-wl.com

For information about Microsemi Corporation, please go to our web site at:
www.microsemi.com

Overview

Scope

This document is intended to provide the reader with sufficient knowledge to correctly interface the BAC with the Key Module component and to understand how to configure both to execute the FIPS 197 Advanced Encryption Standard algorithm. The interfaces for the BAC will be described, as well as some module use cases. For more information concerning the Key Module component, please reference the Key Module User Guide. Additionally, this document will provide recommended configurations for the Key Module component for use with the BAC.

HardAES Overview

HardAES is a Differential Power Analysis (DPA) resilient implementation of the FIPS 197 Advanced Encryption Standard algorithm by the application of the firmware BAC component, the HardAES Software application, and a Key Module compatible storage component.

Features

The HardAES implementation provides the following:

- DPA resistant execution of the FIPS 197 Advanced Encryption Standard algorithm
- Key length agnostic algorithm execution capable of executing AES 128, 192 or 256 with the same design
- Runtime key update
- Support for unique execution variation
- Support for Key Module PUF implementation for key modifier data

Operation

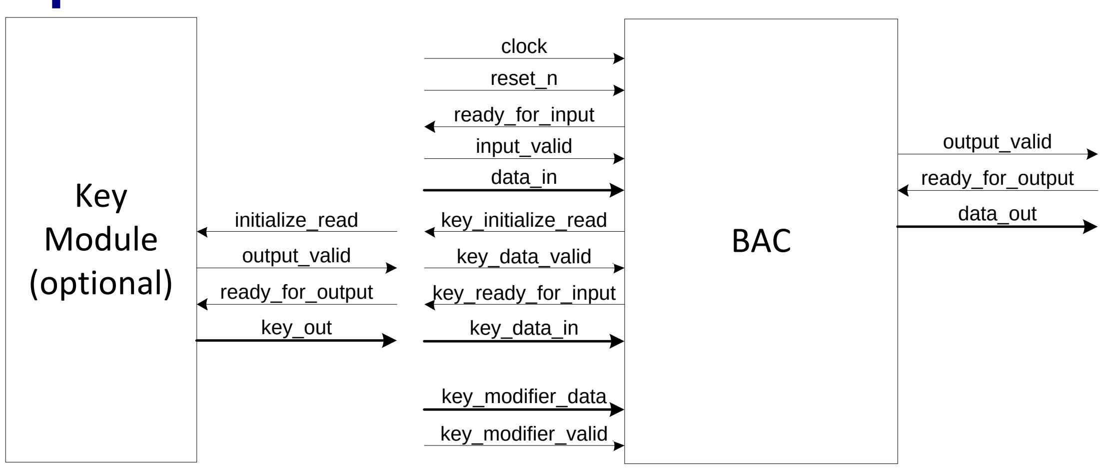

Figure 1: HardAES Interface

HardAES can utilize the BAC to parse a key generated by the HardAES software application to perform FIPS 197 AES.

Generic Parameters

The BAC has several options that are user configurable via HDL generics.

Generic	Type	Definition
`ENFORCIT_DEVICE_FAMILY`	deviceFamily	Cyclone3LS, Virtex4, Virtex5, Virtex6, Spartan6, SmartFusion2, Igloo2
`KEY_MODIFIER_DATA_WIDTH`	Integer	The width of the key modifier supplied to the BAC
`KEY_DATA_WIDTH`	Integer	The width of the key data port supplied to the BAC
`DATA_WIDTH`	Integer	The width of the data port supplied to the BAC

Table 1: Generic Descriptions

Interfaces

Hardware Interface

Signals are synchronous to the rising edge of `clock` and are active high unless otherwise noted.

Port Name	Input/Output	Width	Description
`clock`	in	1	Clock signal
`reset_n`	in	1	Active low: reset signal
`ready_for_input`	out	1	Indicates the module is ready to capture data on data_in
`input_valid`	in	1	Indicates valid data on data_in
`data_in`	in	DATA_WIDTH	Data input port
`data_out`	out	DATA_WIDTH	Data output port
`output_valid`	out	1	Indicates valid data on data_out
`ready_for_output`	in	1	Indicates the user is ready to capture data on data_out
`key_data_in`	in	KEY_DATA_WIDTH	Key data input port
`key_data_valid`	in	1	Indicates valid data on key_data_in
`key_ready_for_input`	out	1	Indicates the module is ready to capture data on key_data_in
`key_initialize_read`	out	1	Indicates the module is requesting that key data be restarted from the beginning
`key_modifier_data`	in	Configurable	Input port for unique execution data
`key_modifier_valid`	in	1	Indicates valid data on key_modifier_data

Table 2: Port Descriptions

Special Consideration

- **Note on configuration of Key Module**
  - For space savings, we recommend leveraging the Key Module as the key storage mechanism, implementing it using the EmbeddedRam storage type. The underlying RAM within the Key Module may or may not be secured.
  - If the Key Module must store generated keys of different lengths, keys that are not the same length as the largest key must be padded with zeros for proper operation.
- **Note on interoperability of Key Module and key modifier data interface**
  - The key modifier data interface is compatible with the Key Module: you must initialize read according to the Key Module interface specification to retrieve the key modifier data.
  - You may use the Key Module with PUF storage to enable per device unique execution.
- **Note on Key Data**
  - This data should be protected as if it were the original key.

Component Declaration

A VHDL package file will be delivered containing the component declaration of the BAC.

Use Cases

Use Case: DPA Resistant AES

The BAC is designed to mitigate DPA vulnerabilities with a security versus performance tradeoff.

References

NIST FIPS-197, November 2001, Advanced Encryption Standard,
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf