EF_AES

APB, AHBL and wishbone wrappers for the symmetric block cipher AES (Advanced Encryption Standard) which is implemented in Verilog in the secworks/aes repository.

The wrapped IP

APB, AHBL, and Wishbone wrappers are provided. All wrappers provide the same programmer's interface as outlined in the following sections.

Wrapped IP System Integration

Based on your use case, use one of the provided wrappers or create a wrapper for your system bus type. For an example of how to integrate the wishbone wrapper:

```
EF_AES_WB INST (
        .clk_i(clk_i),
        .rst_i(rst_i),
        .adr_i(adr_i),
        .dat_i(dat_i),
        .dat_o(dat_o),
        .sel_i(sel_i),
        .cyc_i(cyc_i),
        .stb_i(stb_i),
        .ack_o(ack_o),
        .we_i(we_i),
        .IRQ(irq),
);
```

Wrappers with DFT support

Wrappers in the directory `/hdl/rtl/bus_wrappers/DFT` have an extra input port `sc_testmode` to disable the clock gate whenever the scan chain testmode is enabled.

Interrupt Request Line (irq)

This IP generates interrupts on specific events, which are described in the Interrupt Flags section bellow. The IRQ port should be connected to the system interrupt controller.

Implementation example

The following table is the result for implementing the EF_AES IP with different wrappers using Sky130 HD library and OpenLane2 flow.

Module	Number of cells	Max. freq
EF_AES	TBD	TBD
EF_AES_APB	TBD	TBD
EF_AES_AHBL	TBD	TBD
EF_AES_WB	TBD	TBD

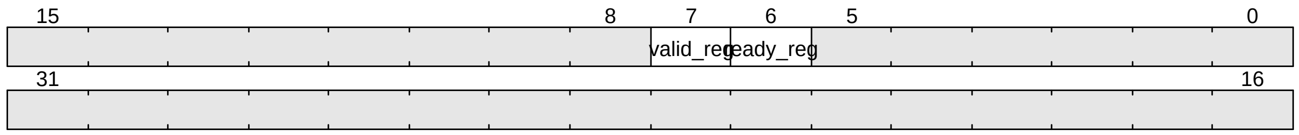
The Programmer's Interface

Registers

Name	Offset	Reset Value	Access Mode	Description
STATUS	0000	0x00000000	r	Status register bit 6: ready , bit 7: valid
CTRL	0004	0x00000000	w	Control register bit 0: Initial bit (init), bit 1: Next bit , bit 2: Encipher/Decipher control, bit 3: Key length control
KEY0	0008	0x00000000	w	Contains the bits 31-0 of the input key value
KEY1	000c	0x00000000	w	Contains the bits 63-32 of the input key value
KEY2	0010	0x00000000	w	Contains the bits 95-64 of the input key value
KEY3	0014	0x00000000	w	Contains the bits 127-96 of the input key value
KEY4	0018	0x00000000	w	Contains the bits 159-128 of the input key value
KEY5	001c	0x00000000	w	Contains the bits 191-160 of the input key value
KEY6	0020	0x00000000	w	Contains the bits 223-192 of the input key value
KEY7	0024	0x00000000	w	Contains the bits 255-224 of the input key value
BLOCK0	0028	0x00000000	w	Contains the bits 31-0 of the input block value
BLOCK1	002c	0x00000000	w	Contains the bits 63-32 of the input block value
BLOCK2	0030	0x00000000	w	Contains the bits 95-64 of the input block value
BLOCK3	0034	0x00000000	w	Contains the bits 127-96 of the input block value
RESULT0	0038	0x00000000	w	Contains the bits 31-0 of the input result value
RESULT1	003c	0x00000000	w	Contains the bits 63-32 of the input result value
RESULT2	0040	0x00000000	w	Contains the bits 95-64 of the input result value
RESULT3	0044	0x00000000	w	Contains the bits 127-96 of the input result value
IM	ff00	0x00000000	w	Interrupt Mask Register; write 1/0 to enable/disable interrupts; check the interrupt flags table for more details
RIS	ff08	0x00000000	w	Raw Interrupt Status; reflects the current interrupts status;check the interrupt flags table for more details
MIS	ff04	0x00000000	w	Masked Interrupt Status; On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect; check the interrupt flags table for more details
IC	ff0c	0x00000000	w	Interrupt Clear Register; On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared; check the interrupt flags table for more details
GCLK	ff10	0x00000000	w	Gated clock enable; 1: enable clock, 0: disable clock

STATUS Register [Offset: 0x0, mode: r]

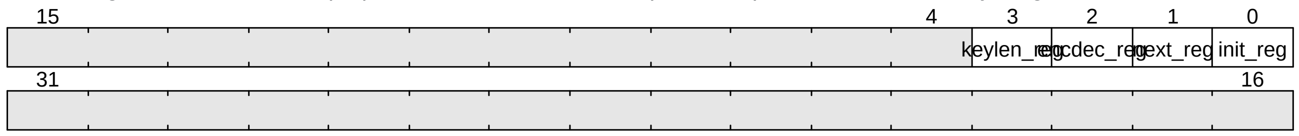
Status register bit 6: ready , bit 7: valid



bit	field name	width	description
6	ready_reg	1	Ready to start
7	valid_reg	1	Result is valid

CTRL Register [Offset: 0x4, mode: w]

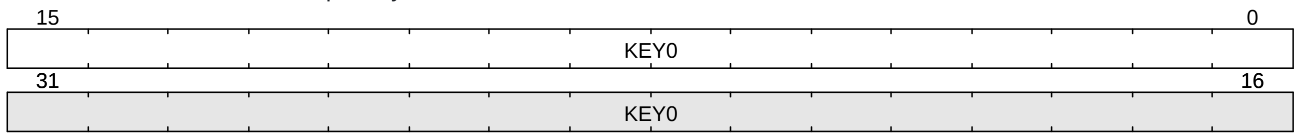
Control register bit 0: Initial bit (init), bit 1: Next bit , bit 2: Encipher/Decipher control, bit 3: Key length control



bit	field name	width	description
0	init_reg	1	Initial bit
1	next_reg	1	Next bit
2	encdec_reg	1	Encipher/Decipher control (“0” means Decipher “1” means Encipher)
3	keylen_reg	1	Key length control (“0” means 128 bit key length “1” means 256 bit key length")

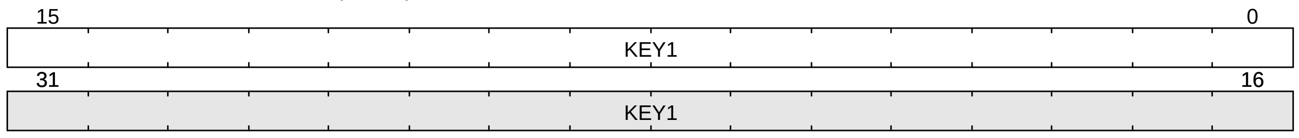
KEY0 Register [Offset: 0x8, mode: w]

Contains the bits 31-0 of the input key value



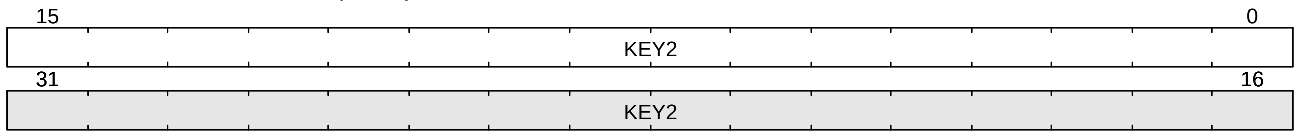
KEY1 Register [Offset: 0xc, mode: w]

Contains the bits 63-32 of the input key value



KEY2 Register [Offset: 0x10, mode: w]

Contains the bits 95-64 of the input key value



KEY3 Register [Offset: 0x14, mode: w]

Contains the bits 127-96 of the input key value

15	KEY3	0
31	KEY3	16

KEY4 Register [Offset: 0x18, mode: w]

Contains the bits 159-128 of the input key value

15	KEY4	0
31	KEY4	16

KEY5 Register [Offset: 0x1c, mode: w]

Contains the bits 191-160 of the input key value

15	KEY5	0
31	KEY5	16

KEY6 Register [Offset: 0x20, mode: w]

Contains the bits 223-192 of the input key value

15	KEY6	0
31	KEY6	16

KEY7 Register [Offset: 0x24, mode: w]

Contains the bits 255-224 of the input key value

15	KEY7	0
31	KEY7	16

BLOCK0 Register [Offset: 0x28, mode: w]

Contains the bits 31-0 of the input block value

15	BLOCK0	0
31	BLOCK0	16

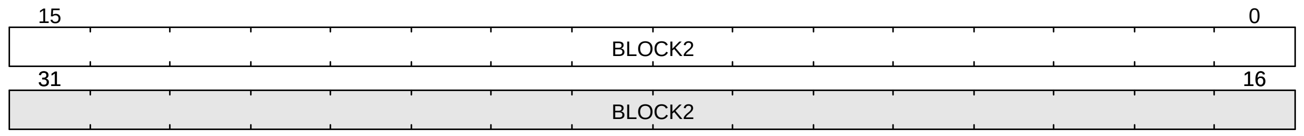
BLOCK1 Register [Offset: 0x2c, mode: w]

Contains the bits 63-32 of the input block value

15	BLOCK1	0
31	BLOCK1	16

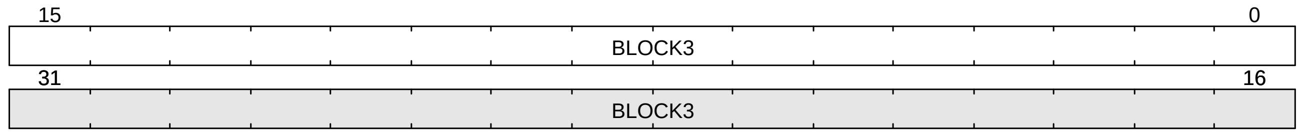
BLOCK2 Register [Offset: 0x30, mode: w]

Contains the bits 95-64 of the input block value



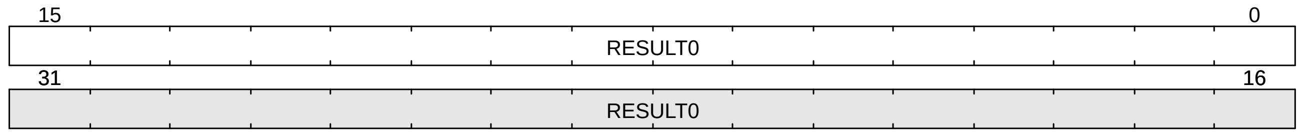
BLOCK3 Register [Offset: 0x34, mode: w]

Contains the bits 127-96 of the input block value



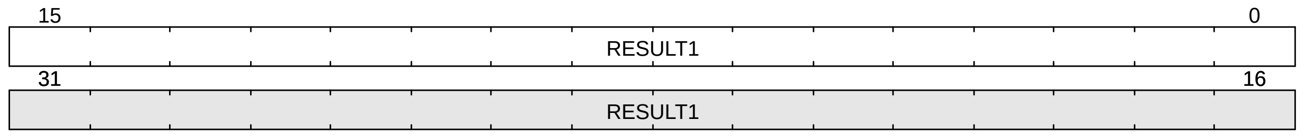
RESULT0 Register [Offset: 0x38, mode: w]

Contains the bits 31-0 of the input result value



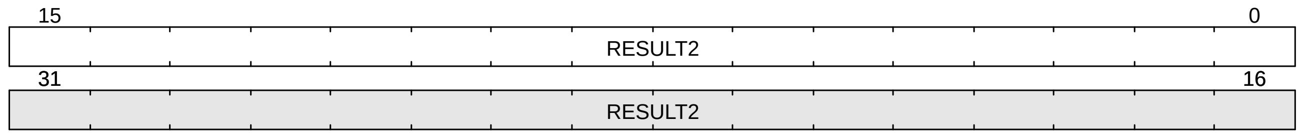
RESULT1 Register [Offset: 0x3c, mode: w]

Contains the bits 63-32 of the input result value



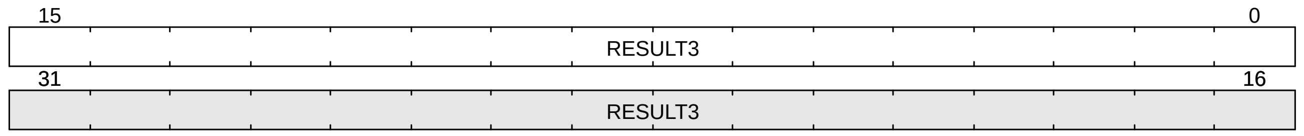
RESULT2 Register [Offset: 0x40, mode: w]

Contains the bits 95-64 of the input result value



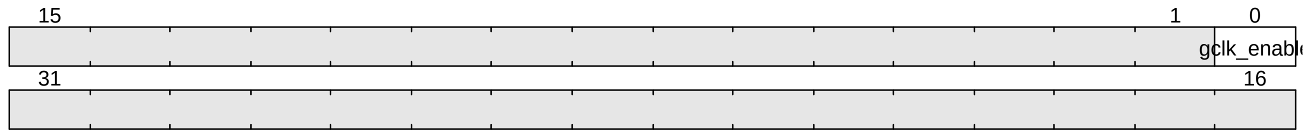
RESULT3 Register [Offset: 0x44, mode: w]

Contains the bits 127-96 of the input result value



GCLK Register [Offset: 0xff10, mode: w]

Gated clock enable register



bit	field name	width	description
0	gclk_enable	1	Gated clock enable; 1: enable clock, 0: disable clock

Interrupt Flags

The wrapped IP provides four registers to deal with interrupts: IM, RIS, MIS and IC. These registers exist for all wrapper types.

Each register has a group of bits for the interrupt sources/flags.

- `IM` [offset: `0xff00`]: is used to enable/disable interrupt sources.
- `RIS` [offset: `0xff08`]: has the current interrupt status (interrupt flags) whether they are enabled or disabled.
- `MIS` [offset: `0xff04`]: is the result of masking (ANDing) RIS by IM.
- `IC` [offset: `0xff0c`]: is used to clear an interrupt flag.

The following are the bit definitions for the interrupt registers:

Bit	Flag	Width	Description
0	VALID	1	Result is valid
1	READY	1	Ready to start

Clock Gating

The IP includes a clock gating feature that allows selective activation and deactivation of the clock using the `GCLK` register. This capability is implemented through the `ef_util_gating_cell` module, which is part of the common modules library, ef_util_lib.v. By default, the clock gating is disabled. To enable behavioral implmentation clock gating, only for simulation purposes, you should define the `CLKG_GENERIC` macro. Alternatively, define the `CLKG_SKY130_HD` macro if you wish to use the SKY130 HD library clock gating cell, `sky130_fd_sc_hd__dlclkp_4` .

Note: If you choose the OpenLane2 flow for implementation and would like to enable the clock gating feature, you need to add `CLKG_SKY130_HD` macro to the `VERILOG_DEFINES` configuration variable. Update OpenLane2 YAML configuration file as follows:

```
VERILOG_DEFINES:
- CLKG_SKY130_HD
```

Firmware Drivers:

Firmware drivers for EF_AES can be found in the Drivers directory in the EFIS (Efabless Firmware Interface Standard) repo. EF_AES driver documentation is available here. You can also find an example C application using the EF_AES drivers here.

Installation:

You can install the IP either by cloning this repository or by using IPM.

1. Using IPM:

- [Optional] If you do not have IPM installed, follow the installation guide here
- After installing IPM, execute the following command `ipm install EF_AES` .

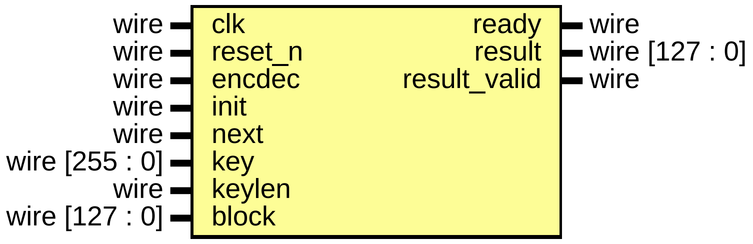
> **Note:** This method is recommended as it automatically installs EF_IP_UTIL as a dependency.

2. Cloning this repo:

- Clone EF_IP_UTIL repository, which includes the required modules from the common modules library, ef_util_lib.v.
  `git clone https://github.com/efabless/EF_IP_UTIL.git`
- Clone the IP repository `git clone github.com/efabless/EF_AES`

The Wrapped IP Interface

> ***NOTE:*** This section is intended for advanced users who wish to gain more information about the interface of the wrapped IP, in case they want to create their own wrappers.



Ports

Port	Direction	Width	Description
encdec	input	1	Encipher/Decipher control
init	input	1	Initial bit
next	input	1	Next bit
ready	input	1	ready to start
key	input	256	key value
keylen	input	1	key length 128 or 256
block	input	128	block value
result	output	128	result value
result_valid	output	1	result is valid