EF_GPIO8

A generic 8-bit General Purpose I/O (GPIO) Peripheral with the following features

- Eight bidirectional pins.
- Input synchronizers
- Input edge detectors.
- Direction control.
- Edge and Level Interrupts generation per pin.
- Wrappers for AHB-Lite, APB and WB buses.

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_GPIO8_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),
        .io_in(io_in),
        .io_out(io_out),
        .io_oe(io_oe)
);
```

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.

External IO interfaces

IO name	Direction	Width	Description
io_in	input	8	GPIOs input
io_out	output	8	GPIOs output
io_oe	output	8	GPIOs output enable

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_GPIO8 IP with different wrappers using Sky130 HD library and OpenLane2 flow.

Module	Number of cells	Max. freq
EF_GPIO8	72	1666
EF_GPIO8_APB	476	1250
EF_GPIO8_AHBL	493	294
EF_GPIO8_WB	574	588

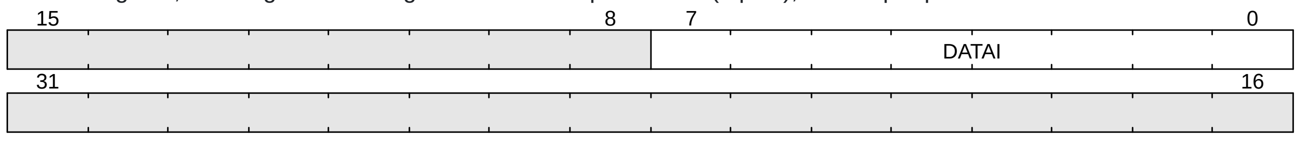
The Programmer's Interface

Registers

Name	Offset	Reset Value	Access Mode	Description
DATAI	0000	0x00000000	r	Data In Register; Reading from this register returns the pins status (8 pins); one bit per pin
DATAO	0004	0x00000000	w	Data Out Register; Writing to this register change the status of the port pins (8 pins); one bit per pin
DIR	0008	0x00000000	w	Direction Register; One bit per pin 1: output, 0: input
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

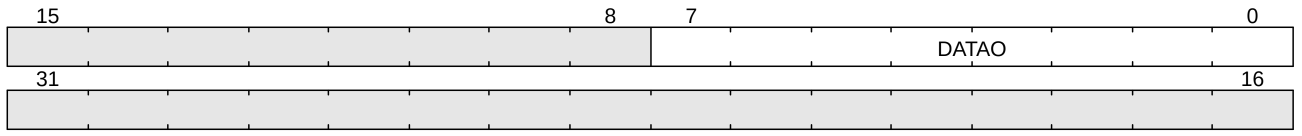
DATAI Register [Offset: 0x0, mode: r]

Data In Register; Reading from this register returns the pins status (8 pins); one bit per pin



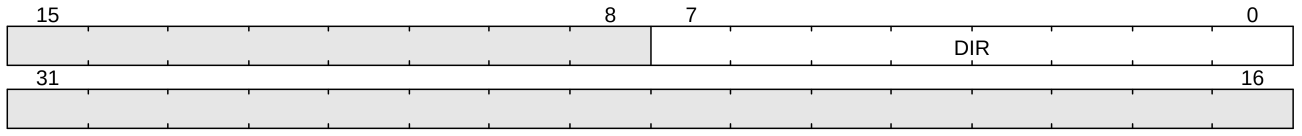
DATAO Register [Offset: 0x4, mode: w]

Data Out Register; Writing to this register change the status of the port pins (8 pins); one bit per pin



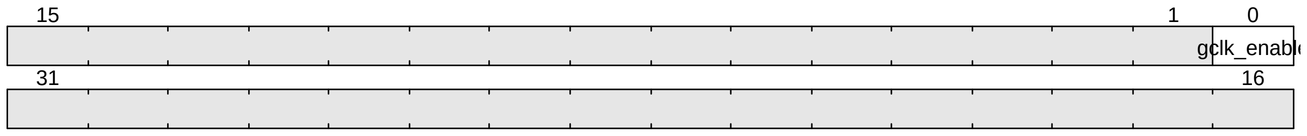
DIR Register [Offset: 0x8, mode: w]

Direction Register; One bit per pin 1: output, 0: input



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	P0HI	1	Pin 0 is high
1	P1HI	1	Pin 1 is high
2	P2HI	1	Pin 2 is high
3	P3HI	1	Pin 3 is high
4	P4HI	1	Pin 4 is high
5	P5HI	1	Pin 5 is high
6	P6HI	1	Pin 6 is high
7	P7HI	1	Pin 7 is high

Bit	Flag	Width	Description
8	P0LO	1	Pin 0 is low
9	P1LO	1	Pin 1 is low
10	P2LO	1	Pin 2 is low
11	P3LO	1	Pin 3 is low
12	P4LO	1	Pin 4 is low
13	P5LO	1	Pin 5 is low
14	P6LO	1	Pin 6 is low
15	P7LO	1	Pin 7 is low
16	P0PE	1	Pin 0 has observed a rising edge
17	P1PE	1	Pin 1 has observed a rising edge
18	P2PE	1	Pin 2 has observed a rising edge
19	P3PE	1	Pin 3 has observed a rising edge
20	P4PE	1	Pin 4 has observed a rising edge
21	P5PE	1	Pin 5 has observed a rising edge
22	P6PE	1	Pin 6 has observed a rising edge
23	P7PE	1	Pin 7 has observed a rising edge
24	P0NE	1	Pin 0 has observed a falling edge
25	P1NE	1	Pin 1 has observed a falling edge
26	P2NE	1	Pin 2 has observed a falling edge
27	P3NE	1	Pin 3 has observed a falling edge
28	P4NE	1	Pin 4 has observed a falling edge
29	P5NE	1	Pin 5 has observed a falling edge
30	P6NE	1	Pin 6 has observed a falling edge
31	P7NE	1	Pin 7 has observed a falling edge

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_GPIO8 can be found in the Drivers directory in the EFIS (Efabless Firmware Interface Standard) repo. EF_GPIO8 driver documentation is available here. You can also find an example C application using the EF_GPIO8 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_GPIO8` .

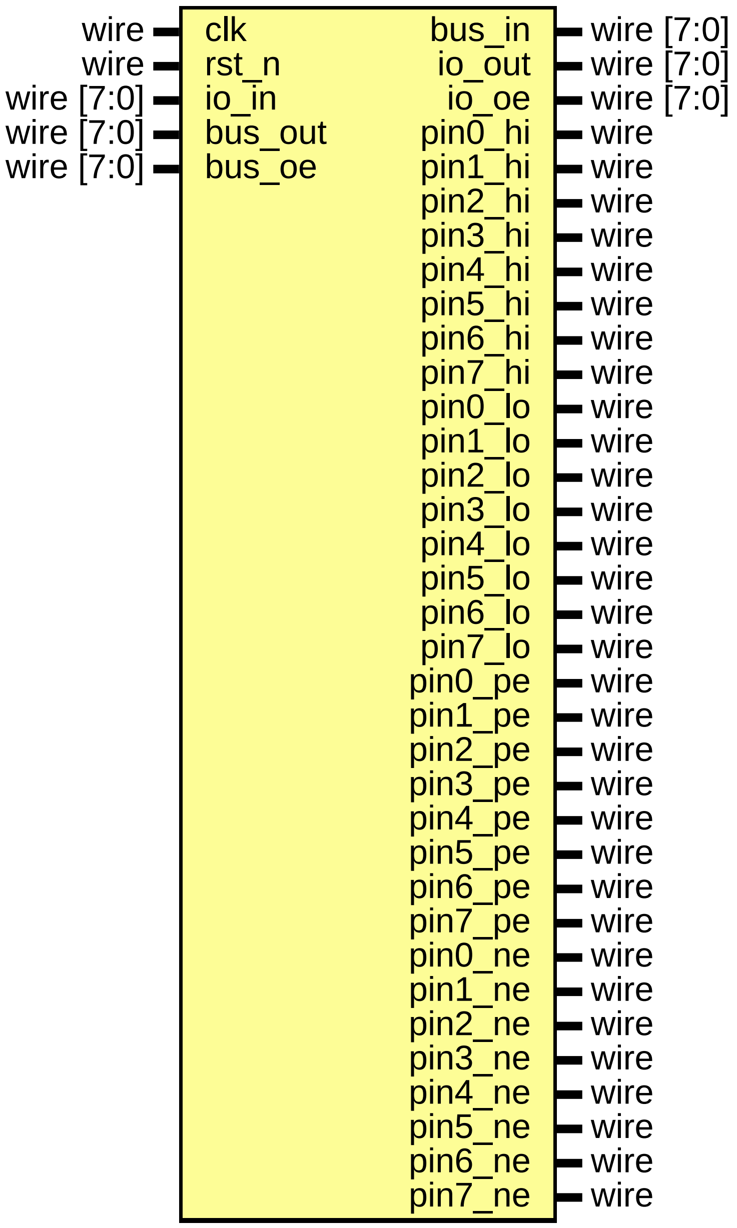
> **Note:** This method is recommended as it automatically installs EF_IP_UTIL as a dependency.

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_GPIO8`

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
io_in	input	8	GPIOs input
io_out	output	8	GPIOs output
io_oe	output	8	GPIOs output enable
bus_in	output	8	Synchronized GPIOs input connected to the bus (it drives the DATAI register)
bus_out	input	8	GPIOs output connected to the bus (it's driven by writing to DATAO register)
bus_oe	input	8	GPIOs output enable connected to the bus (it's driven by writing to DIR register)
pin0_hi	output	1	Pin 0 high flag
pin1_hi	output	1	Pin 1 high flag
pin2_hi	output	1	Pin 2 high flag

Port	Direction	Width	Description
pin3_hi	output	1	Pin 3 high flag
pin4_hi	output	1	Pin 4 high flag
pin5_hi	output	1	Pin 5 high flag
pin6_hi	output	1	Pin 6 high flag
pin7_hi	output	1	Pin 7 high flag
pin0_lo	output	1	Pin 0 low flag
pin1_lo	output	1	Pin 1 low flag
pin2_lo	output	1	Pin 2 low flag
pin3_lo	output	1	Pin 3 low flag
pin4_lo	output	1	Pin 4 low flag
pin5_lo	output	1	Pin 5 low flag
pin6_lo	output	1	Pin 6 low flag
pin7_lo	output	1	Pin 7 low flag
pin0_pe	output	1	Pin 0 positive edge flag
pin1_pe	output	1	Pin 1 positive edge flag
pin2_pe	output	1	Pin 2 positive edge flag
pin3_pe	output	1	Pin 3 positive edge flag
pin4_pe	output	1	Pin 4 positive edge flag
pin5_pe	output	1	Pin 5 positive edge flag
pin6_pe	output	1	Pin 6 positive edge flag
pin7_pe	output	1	Pin 7 positive edge flag
pin0_ne	output	1	Pin 0 negative edge flag
pin1_ne	output	1	Pin 1 negative edge flag
pin2_ne	output	1	Pin 2 negative edge flag
pin3_ne	output	1	Pin 3 negative edge flag
pin4_ne	output	1	Pin 4 negative edge flag
pin5_ne	output	1	Pin 5 negative edge flag
pin6_ne	output	1	Pin 6 negative edge flag
pin7_ne	output	1	Pin 7 negative edge flag