EF_I2C

APB and wishbone wrappers for the I2C master controller which is implemented in Verilog in the alexforencich/verilog-i2c repository.

Wrapped IP System Integration

```
EF_I2C_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),
        .scl_i(scl_i),
        .scl_o(scl_o),
        .scl_oen_o(scl_oen_o),
        .sda_i(sda_i),
        .sda_o(sda_o),
        .sda_oen_o(sda_oen_o)
);
```

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
scl_i	input	1	i2c scl (Serial Clock) input
scl_o	output	1	i2c scl (Serial Clock) output
scl_oen_o	output	1	i2c scl (Serial Clock) output enable
sda_i	input	1	i2c scl (Serial Data) input
sda_o	output	1	i2c scl (Serial Data) output
sda_oen_o	output	1	i2c scl (Serial Data) output enable
i2c_irq	output	1	i2c interrupt

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_I2C IP with different wrappers using Sky130 HD library and OpenLane2 flow.

Module	Number of cells	Max. freq
EF_I2C	TBD	TBD
EF_I2C_APB	TBD	TBD

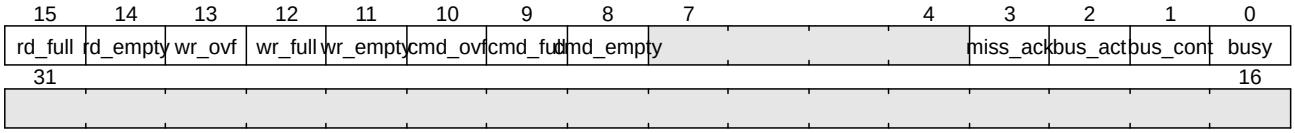
The Programmer's Interface

Registers

Name	Offset	Reset Value	Access Mode	Description
Status	0000	0x00000000	w	status register
Command	0004	0x00000000	w	bit 0-6: cmd_address, bit 8: cmd_start, bit 9: cmd_read, bit 10: cmd_write, bit 11: cmd_wr_m, bit 12: cmd_stop. Setting more than one command bit is allowed. Start or repeated start will be issued first, followed by read or write, followed by stop. Note that setting read and write at the same time is not allowed, this will result in the command being ignored.
Data	0008	0x00000000	w/r	bit 0-7: data, bit 8: data_valid, bit 9: data_last
PR	000c	0x00000000	w	prescale = Fclk / (FI2Cclk * 4)
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: w]

status register

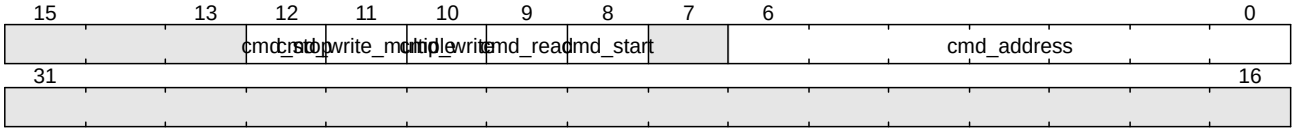


bit	field name	width	description
0	busy	1	high when module is performing an I2C operation
1	bus_cont	1	high when module has control of active bus
2	bus_act	1	high when bus is active
3	miss_ack	1	set high when an ACK pulse from a slave device is not seen; write 1 to clear

bit	field name	width	description
8	cmd_empty	1	command FIFO empty
9	cmd_full	1	command FIFO full
10	cmd_ovf	1	command FIFO overflow; write 1 to clear
11	wr_empty	1	write data FIFO empty
12	wr_full	1	write data FIFO full
13	wr_ovf	1	write data FIFO overflow; write 1 to clear
14	rd_empty	1	read data FIFO is empty
15	rd_full	1	read data FIFO is full

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

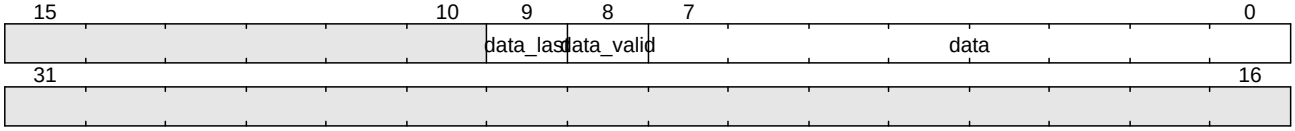
bit 0-6: cmd_address, bit 8: cmd_start, bit 9: cmd_read, bit 10: cmd_write, bit 11: cmd_wr_m, bit 12: cmd_stop. Setting more than one command bit is allowed. Start or repeated start will be issued first, followed by read or write, followed by stop. Note that setting read and write at the same time is not allowed, this will result in the command being ignored.



bit	field name	width	description
0	cmd_address	7	I2C address for command
8	cmd_start	1	set high to issue I2C start, write to push on command FIFO
9	cmd_read	1	set high to start read, write to push on command FIFO
10	cmd_write	1	set high to start write, write to push on command FIFO
11	cmd_write_multiple	1	set high to start block write, write to push on command FIFO
12	cmd_stop	1	set high to issue I2C stop, write to push on command FIFO

Data Register [Offset: 0x8, mode: w/r]

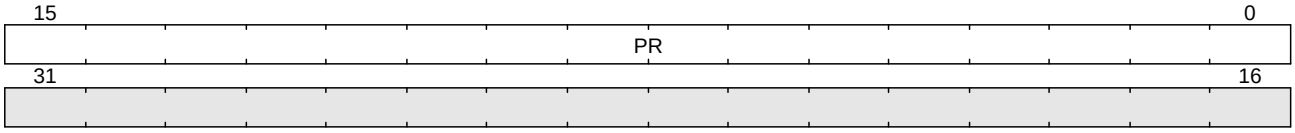
bit 0-7: data, bit 8: data_valid, bit 9: data_last



bit	field name	width	description
0	data	8	I2C data, write to push on write data FIFO, read to pull from read data FIFO
8	data_valid	1	indicates valid read data, must be accessed with atomic 16 bit reads and writes
9	data_last	1	indicate last byte of block write (write_multiple), must be accessed with atomic 16 bit reads and writes

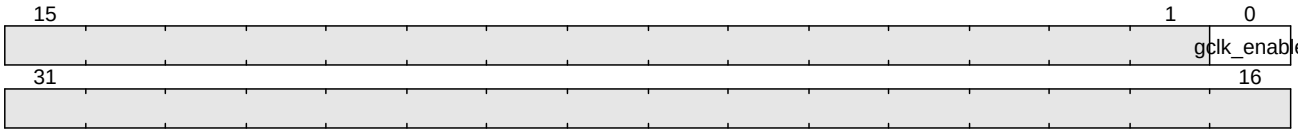
PR Register [Offset: 0xc, mode: w]

prescale = Fclk / (FI2Cclk * 4)



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	MISS_ACK	1	Slave ACK is missed
1	CMDE	1	Command FIFO is Empty
2	CMDF	1	Command FIFO is Full
3	CMDOVF	1	Command FIFO overflow; write 1 to clear
4	WRE	1	Write FIFO is Empty
5	WRF	1	Write FIFO is Full
6	WROVF	1	Write FIFO overflow; write 1 to clear
7	RDE	1	Read FIFO is Empty
8	RDF	1	Read FIFO is Full

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_I2C can be found in the Drivers directory in the EFIS (Efabless Firmware Interface Standard) repo. EF_I2C driver documentation is available here. You can also find an example C application using the EF_I2C 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_I2C` .

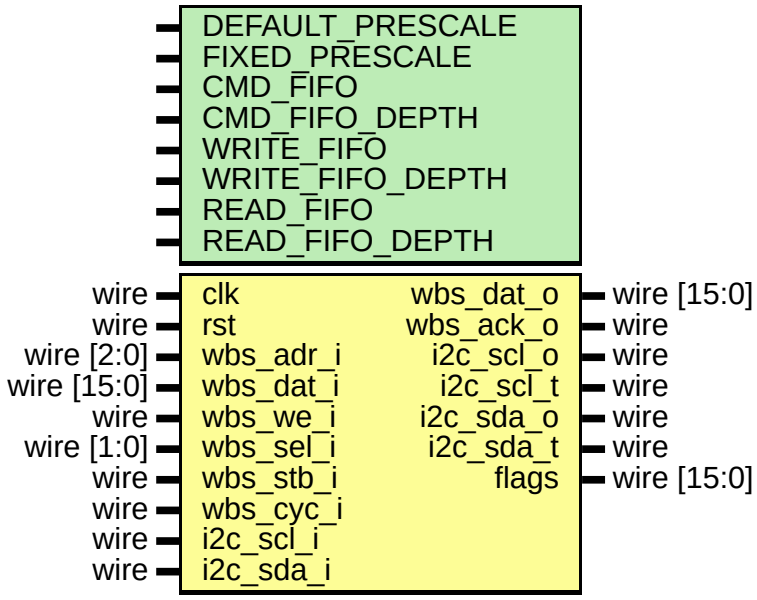
> **Note:** This method is recommended as it automatically installs EF_IP_UTIL as a dependency.

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 https://github.com/efabless/EF_I2C`

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.



Module Parameters

Parameter	Description	Default Value
DEFAULT_PRESCALE	Default value for Prescale; prescale = Fclk / (FI2Cclk * 4)	1
FIXED_PRESCALE	Prescale value is fixed or could be dynamically configured	0
CMD_FIFO	Command AXI4 FIFO enable	1
CMD_FIFO_DEPTH	Command AXI4 FIFO depth	32
WRITE_FIFO	Write AXI4 FIFO enable	1
WRITE_FIFO_DEPTH	Write AXI4 FIFO depth	32
READ_FIFO	Read AXI4 FIFO enable	1
READ_FIFO_DEPTH	Read AXI4 FIFO depth	32

Ports

Port	Direction	Width	Description
scl_i	input	1	i2c scl (Serial Clock) input
scl_o	output	1	i2c scl (Serial Clock) output
scl_oen_o	output	1	i2c scl (Serial Clock) output enable
sda_i	input	1	i2c scl (Serial Data) input
sda_o	output	1	i2c scl (Serial Data) output
sda_oen_o	output	1	i2c scl (Serial Data) output enable
i2c_irq	output	1	i2c interrupt
wbs_adr_i	input	3	wishbone input address
wbs_dat_i	input	16	wishbone input data
wbs_dat_o	output	16	wishbone data out
wbs_we_i	input	1	wishbone write enable

Port	Direction	Width	Description
wbs_sel_i	input	2	wishbone select
wbs_stb_i	input	1	wishbone chip select
wbs_ack_o	output	1	wishbone acknowledge
wbs_cyc_i	input	1	wishbone bus cycle
i2c_scl_i	input	1	i2c scl (Serial Clock) input
i2c_scl_o	output	1	i2c scl (Serial Clock) output
i2c_scl_t	output	1	i2c scl (Serial Clock) tristate
i2c_sda_i	input	1	i2c scl (Serial Data) input
i2c_sda_o	output	1	i2c scl (Serial Data) output
i2c_sda_t	output	1	i2c scl (Serial Data) tristate
flags	output	16	i2c flags