EF_UART

UART, or universal asynchronous receiver-transmitter, is one of the most used device-to-device communication protocols. A UART enables two devices to exchange data serially without sharing the clock in a frame oriented way. The frame consists of a start bit, a number of data bits (typically one byte), a parity bit (optional) and 1-2 stop bits. EF_UART is a Soft IP with the following features:

- A configurable frame format
    - Data bits could vary from 5 to 9 bits
    - Even, odd, stick, or no-parity bit generation/detection
    - One or Two stop bit generation
- Line-break detection
- Configurable receiver timeout
- Loopback capability for testing/debugging
- Glitch Filter on RX enable
- Matching received data detection
- 16-byte TX and RX FIFOs with programmable thresholds
- 16-bit prescaler (PR) for programmable baud rate generation
- Ten Interrupt Sources:
    - RX FIFO is full
    - TX FIFO is empty
    - RX FIFO level is above the set threshold
    - TX FIFO level is below the set threshold
    - Line break detection
    - Receiver data match
    - Frame Error
    - Parity Error
    - Overrun
    - Receiver timeout

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_UART_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),
  .rx(rx),
  .tx(tx)
);
```

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
rx	input	1	RX connected to the external interface
tx	output	1	TX connected to external interface

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

Module	Number of cells	Max. freq
EF_UART	1590	277
EF_UART_APB	1943	208
EF_UART_AHBL	1973	250
EF_UART_WB	2170	83

The Programmer's Interface

Registers

Name	Offset	Reset Value	Access Mode	Description
RXDATA	0000	0x00000000	r	RX Data register; the interface to the Receive FIFO.
TXDATA	0004	0x00000000	w	TX Data register; ; the interface to the Receive FIFO.

PR	0008	0x00000000	w	The Prescaler register; used to determine the baud rate. $baud_rate = clock_freq/((PR+1)*16)$.
CTRL	000c	0x00000000	w	UART Control Register
CFG	0010	0x00003F08	w	UART Configuration Register
MATCH	001c	0x00000000	w	Match Register
RX_FIFO_LEVEL	fe00	0x00000000	r	RX_FIFO Level Register
RX_FIFO_THRESHOLD	fe04	0x00000000	w	RX_FIFO Level Threshold Register
RX_FIFO_FLUSH	fe08	0x00000000	w	RX_FIFO Flush Register
TX_FIFO_LEVEL	fe10	0x00000000	r	TX_FIFO Level Register
TX_FIFO_THRESHOLD	fe14	0x00000000	w	TX_FIFO Level Threshold Register
TX_FIFO_FLUSH	fe18	0x00000000	w	TX_FIFO Flush Register
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

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

RX Data register; the interface to the Receive FIFO. [image]

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

TX Data register; ; the interface to the Receive FIFO. [image]

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

The Prescaler register; used to determine the baud rate. $baud_rate = clock_freq/((PR +1)*16)$. [image]

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

UART Control Register [image]

bit	field name	width	description
0	en	1	UART enable
1	txen	1	UART Transmitter enable
2	rxen	1	UART Receiver enable
3	lpen	1	Loopback (connect RX and TX pins together) enable
4	gfen	1	UART Glitch Filer on RX enable

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

UART Configuration Register [image]

bit	field name	width	description
0	wlen	4	Data word length: 5-9 bits
4	stp2	1	Two Stop Bits Select
5	parity	3	Parity Type: 000: None, 001: odd, 010: even, 100: Sticky 0, 101: Sticky 1
8	timeout	6	Receiver Timeout measured in number of bits

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

Match Register [image]

RX_FIFO_LEVEL Register [Offset: 0xfe00, mode: r]

RX_FIFO Level Register [image]

bit	field name	width	description
0	level	4	FIFO data level

RX_FIFO_THRESHOLD Register [Offset: 0xfe04, mode: w]

RX_FIFO Level Threshold Register [image]

bit	field name	width	description
0	threshold	4	FIFO level threshold value

RX_FIFO_FLUSH Register [Offset: 0xfe08, mode: w]

RX_FIFO Flush Register [image]

bit	field name	width	description
0	flush	1	FIFO flush

TX_FIFO_LEVEL Register [Offset: 0xfe10, mode: r]

TX_FIFO Level Register [image]

bit	field name	width	description
0	level	4	FIFO data level

TX_FIFO_THRESHOLD Register [Offset: 0xfe14, mode: w]

TX_FIFO Level Threshold Register [image]

bit	field name	width	description
0	threshold	4	FIFO level threshold value

TX_FIFO_FLUSH Register [Offset: 0xfe18, mode: w]

TX_FIFO Flush Register [image]

bit	field name	width	description
0	flush	1	FIFO flush

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

Gated clock enable register [image]

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	TXE	1	Transmit FIFO is Empty.
1	RXF	1	Receive FIFO is Full.
2	TXB	1	Transmit FIFO level is Below Threshold.
3	RXA	1	Receive FIFO level is Above Threshold.
4	BRK	1	Line Break; 13 consecutive 0's have been detected on the line.
5	MATCH	1	Match; the received data matches the MATCH register.

6	FE	1	Framing Error; the receiver does not see a "stop" bit at the expected "stop" bit time.
7	PRE	1	Parity Error; the receiver calculated parity does not match the received one.
8	OR	1	Overrun; data has been received but the RX FIFO is full.
9	RTO	1	Receiver Timeout; no data has been received for the time of a specified number of bits.

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_UART can be found in the fw directory. EF_UART driver documentation is available here. You can also find an example C application using the EF_UART 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_UART`.

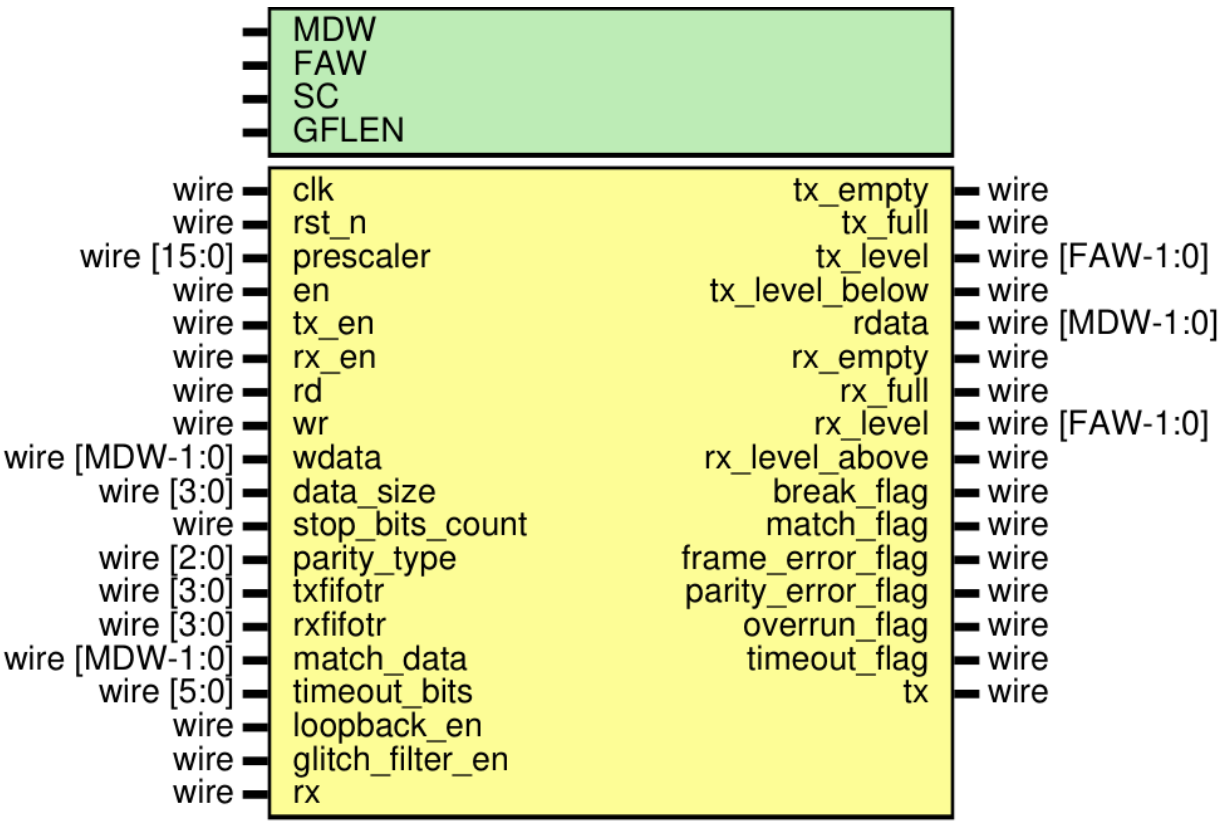
  Note: This method is recommended as it automatically installs EF IP UTIL as a dependency.

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_UART.git`

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
SC	Number of samples per bit/baud	8
MDW	Max data size/width	9
GFLEN	Length (number of stages) of the glitch filter	8
FAW	FIFO Address width; Depth = 2^AW	4

Ports

Port	Direction	Width	Description
rx	input	1	RX connected to the external interface
tx	output	1	TX connected to external interface
prescaler	input	16	Prescaler used to determine the baud rate.
en	input	1	Enable for UART
tx_en	input	1	Enable for UART transmission
rx_en	input	1	Enable for UART receiving
wdata	input	MDW	Transmission data
timeout_bits	input	6	Receiver Timeout measured in number of bits.
loopback_en	input	1	Loopback enable; connect tx to the rx

glitch_filter_en	input	1	UART Glitch Filter on RX enable
tx_level	output	FAW	The current level of TX FIFO
rx_level	output	FAW	The current level of RX FIFO
rd	input	1	Read from RX FIFO signal
wr	input	1	Write to TX FIFO signal
tx_fifo_flush	input	1	Flushes the TX FIFO.
rx_fifo_flush	input	1	Flushes the RX FIFO.
data_size	input	4	Number of data bits in the frame
stop_bits_count	input	1	Number of stop bits in the frame (could be 1 or 2)
parity_type	input	3	Type of Parity in the frame
txfifotr	input	FAW	TX FIFO Threshold
rxfifotr	input	FAW	RX FIFO Threshold
match_data	input	MDW	Match data (match flag would be raised if it matches what is received)
tx_empty	output	1	TX empty flag
tx_full	output	1	TX full flag
tx_level_below	output	1	TX level below flag
rdata	output	MDW	Received Data
rx_empty	output	1	RX empty flag
rx_full	output	1	RX full flag
rx_level_above	output	1	RX level above flag
break_flag	output	1	Break flag
match_flag	output	1	Match flag
frame_error_flag	output	1	Frame error flag
parity_error_flag	output	1	Parity error flag
overrun_flag	output	1	Overrun flag
timeout_flag	output	1	Timeout flag