up_uart.v

AUTHORS

JAY CONVERTINO

DATES

2024/02/29

INFORMATION

Brief

uP Core for interfacing with axis uart emulates Xilinx UART lite.

License MIT

up_uart

```
module up_uart #(
parameter
ADDRESS_WIDTH
 =
32,
parameter
BUS_WIDTH
 =
4,
parameter
CLOCK_SPEED
 =
100000000,
parameter
```

```
BAUD_RATE
 =
2000000,
parameter
PARITY_ENA
 =
0,
parameter
PARITY_TYPE
 =
0,
parameter
STOP_BITS
 =
1,
parameter
DATA_BITS
 =
8,
parameter
RX_DELAY
 =
0,
parameter
RX_BAUD_DELAY
 =
0,
parameter
TX_DELAY
 =
0,
parameter
TX_BAUD_DELAY
 =
0
) ( input clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WID
```

uP based uart communications device.

Parameters

ADDRESS_WIDTH parameter	Width of the uP address port, max 32 bit.
BUS_WIDTH parameter	Width of the uP bus data port.
CLOCK_SPEED parameter	This is the aclk frequency in Hz
BAUD_RATE parameter	Serial Baud, this can be any value including non-standard.
PARITY_ENA parameter	Enable Parity for the data in and out.
PARITY_TYPE parameter	Set the parity type, 0 = even, 1 = odd, 2 = mark, 3 = space.
STOP_BITS parameter	Number of stop bits, 0 to crazy non-standard amounts.
DATA_BITS parameter	Number of data bits, 1 to crazy non-standard amounts.
RX_DELAY parameter	Delay in rx data input.
RX_BAUD_DELAY parameter	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).

TX_DELAY parameter	Delay in tx data output. Delays the time to output of the data.
TX_BAUD_DELAY parameter	Delay in tx baud enable. This will delay the time the bit output starts.

Ports

clk	Clock for all devices in the core
rstn	Negative reset
up_rreq	uP bus read request
up_rack	uP bus read ack
up_raddr	uP bus read address
up_rdata	uP bus read data
up_wreq	uP bus write request
up_wack	uP bus write ack
up_waddr	uP bus write address
up_wdata	uP bus write data
irq	Interrupt when data is received
tx	transmit for UART (output to RX)
rx	receive for UART (input from TX)
rts	request to send is a loop with CTS
cts	clear to send is a loop with RTS

DIVISOR

```
localparam DIVISOR = BUS_WIDTH/2
```

Divide the address register default location for 1 byte access to multi byte access. (register offsets are byte offsets).

FIFO_DEPTH

```
localparam FIFO_DEPTH = 16
```

Depth of the fifo, matches UART LITE (xilinx), so I kept this just cause

REGISTER INFORMATION

Core has 4 registers at the offsets that follow.

RX_FIFO_REG	h0
TX_FIFO_REG	h4
STATUS_REG	h8
CONTROL_REG	hC

RX_FIFO_REG

```
localparam RX_FIFO_REG = 4'h0 >> DIVISOR
```

Defines the address offset for RX FIFO

RX FIFO REGISTER	
31:8	7:0
UNUSED	RECEIVED DATA

Valid bits are from DATA_BITS:0, which are data.

TX_FIFO_REG

```
localparam TX_FIFO_REG = 4'h4 >> DIVISOR
```

Defines the address offset to write the TX FIFO.

TX FIFO REGISTER	
31:8	7:0
UNUSED	TRANSMIT DATA

Valid bits are from DATA_BITS:0, which are data.

STATUS_REG

```
localparam STATUS_REG = 4'h8 >> DIVISOR
```

Defines the address offset to read the status bits.

STATUS REGISTER								
31:8	7	6	5	4	3	2	1	0
UNUSED	PE	FE	OE	irq_en	tx_full	tx_empty	rx_full	rx_valid

Status Register Bits

PE 7, Parity error, active high on error

FE 6, Frame error, active high on error

OE 5, Overrun error, active high on error

irq_en 4, 1 when the IRQ is enabled by CONTROL_REG

tx_full 3, When 1 the tx fifo is full.

tx_empty 2, When 1 the tx fifo is empty.

rx_full 1, When 1 the rx fifo is full.

rx_valid 0, When 1 the rx fifo contains valid data.

CONTROL_REG

```
localparam CONTROL_REG = 4'hC >> DIVISOR
```

Defines the address offset to set the control bits.

CONTROL REGISTER				
31:5	4	3:2	1	0
UNUSED	ENA_INTR_BIT	UNUSED	RST_RX_BIT	RST_TX_BIT

See Also: ENABLE_INTR_BIT, RESET_RX_BIT, RESET_TX_BIT

Control Register Bits

ENABLE_INTR_BIT 4, Control Register offset bit for enabling the interrupt.

RESET_RX_BIT 1, Control Register offset bit for resetting the RX FIFO.

RESET_TX_BIT 0, Control Register offset bit for resetting the TX FIFO.

INSTANTIATED MODULES

inst_axis_uart

```
axis_uart #(
                                                                    .
BAUD_CLOCK_SPEED(CLOCK_SPEED),
                                                                    .
BAUD_RATE(BAUD_RATE),
                                                                    .
PARITY_ENA(PARITY_ENA),
                                                                    .
PARITY_TYPE(PARITY_TYPE),
                                                                    .
STOP_BITS(STOP_BITS),
                                                                    .
DATA_BITS(DATA_BITS),
                                                                    .
RX_DELAY(RX_DELAY),
                                                                    .
RX_BAUD_DELAY(RX_BAUD_DELAY),
                                                                    .
TX_DELAY(TX_DELAY),
                                                                    .
TX_BAUD_DELAY(TX_BAUD_DELAY),
                                                                    .
BUS_WIDTH(BUS_WIDTH)
) inst_axis_uart ( .aclk(clk), .arstn(rstn), .parity_err(s_parity_err), .fra
```

UART instance with AXIS interface for TX/RX

inst_rx_fifo

```
fifo #(
                                                                                .
FIFO_DEPTH(FIFO_DEPTH),
                                                                                .
BYTE_WIDTH(BUS_WIDTH),
                                                                                .
COUNT_WIDTH(8),
                                                                                .
FWFT(1),
                                                                                .
RD_SYNC_DEPTH(0),
                                                                                .
WR_SYNC_DEPTH(0),
                                                                                .
DC_SYNC_DEPTH(0),
                                                                                .
COUNT_DELAY(0),
                                                                                .
COUNT_ENA(0),
                                                                                .
DATA_ZERO(0),
                                                                                .
ACK_ENA(0),
                                                                                .
RAM_TYPE("block")
) inst_rx_fifo ( .rd_clk(clk), .rd_rstn(rstn & r_rstn_rx_delay[0]), .rd_en(s
```

Buffer up to 16 items output from the axis_1553_encoder.

inst_tx_fifo

```
fifo #(
                                                                                .
FIFO_DEPTH(FIFO_DEPTH),
                                                                                .
BYTE_WIDTH(BUS_WIDTH),
                                                                                .
COUNT_WIDTH(8),
                                                                                .
FWFT(1),
                                                                                .
RD_SYNC_DEPTH(0),
                                                                                .
WR_SYNC_DEPTH(0),
                                                                                .
DC_SYNC_DEPTH(0),
                                                                                .
COUNT_DELAY(0),
                                                                                .
COUNT_ENA(0),
                                                                                .
DATA_ZERO(0),
                                                                                .
ACK_ENA(0),
                                                                                .
RAM_TYPE("block")
) inst_tx_fifo ( .rd_clk(clk), .rd_rstn(rstn & r_rstn_tx_delay[0]), .rd_en(s
```

Buffer up to 16 items to input to the axis_1553_decoder.