# up\_1553.v

#### **AUTHORS**

## **JAY CONVERTINO**

## **DATES**

#### 2024/10/17

## **INFORMATION**

## **Brief**

uP Core for interfacing with simple 1553 communications.

## **License MIT**

Copyright 2024 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## up 1553

```
module up_1553 #(
parameter
ADDRESS_WIDTH
=
32,
parameter
BUS_WIDTH
=
4,
parameter
CLOCK_SPEED
```

```
100000000,
parameter
SAMPLE_RATE
2000000,
parameter
BIT_SLICE_OFFSET
parameter
INVERT_DATA
parameter
SAMPLE_SELECT
) ( input clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WI[
```

uP based 1553 communications device.

#### **Parameters**

**ADDRESS WIDTH** Width of the uP address port.

**BUS\_WIDTH** Width of the uP bus data port.

parameter

**CLOCK SPEED** This is the aclk frequency in Hz

parameter

SAMPLE RATE Rate of in which to sample the 1553 bus. Must be 2 MHz or more and parameter

less than aclk. This is in Hz. BIT\_SLICE\_OFFSET- Adjust where the

sample is taken from the input.

INVERT\_DATA Invert all 1553 bits coming in and out.

parameter

SAMPLE\_SELECT Adjust where in the array of samples to select a bit.

parameter

#### **Ports**

clk Clock for all devices in the core

Negative reset rstn

uP bus read request up\_rreq uP bus read ack up\_rack up\_raddr uP bus read address uP bus read data up\_rdata uP bus write request up\_wreq uP bus write ack up\_wack up\_waddr uP bus write address up\_wdata uP bus write data

i\_diff Input differential signal for 1553 bus o\_diff Output differential signal for 1553 bus

Enable output of differential signal (for signal switching on 1553 module) en\_o\_diff

Interrupt when data is received irq

# 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
```

Defines the address offset for RX FIFO

RX FIFO REGISTER				
31:24	23:16	15:0		
UNUSED	STATUS DATA	RECEIVED DATA		

Valid bits are from 23:0. Bits 23:16 are information about the data. Bit 15:0 are data.

# TX\_FIFO\_REG

```
localparam TX_FIFO_REG = 4'h4
```

Defines the address offset to write the TX FIFO.

TX FIFO REGISTER				
31:24	23:16	15:0		
UNUSED	STATUS DATA	TRANSMIT DATA		

Valid bits are from 23:0. Bits 23:16 are information about the data. Bit 15:0 are data.

# STATUS\_REG

```
localparam STATUS_REG = 4'h8
```

Defines the address offset to read the status bits.

STATUS REGISTER								
31:8	7	6	5	4	3	2	1	0
UNUSED	rx_rdata	rd_rdata	rx_rdata	r_irq_en	tx_full	tx_empty	rx_full	rx_valid

# **Status Register Bits**

 rx\_rdata
 7, 16

 rx\_rdata
 6, 17

 rx\_rdata
 5, 18

r\_irq\_en 4, 1 when the IRQ is enabled by CONTROL\_REG

tx\_full3, When 1 the tx fifo is full.tx\_empty2, When 1 the tx fifo is empty.rx\_full1, 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

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\_BIT4, Control Register offset bit for enabling the interrupt.RESET\_RX\_BIT1, Control Register offset bit for resetting the RX FIFO.RESET\_TX\_BIT0, Control Register offset bit for resetting the TX FIFO.

## **INSTANTIATED MODULES**

## inst\_axis\_1553\_encoder

# inst\_axis\_1553\_decoder

```
axis_1553_decoder #(

CLOCK_SPEED(CLOCK_SPEED),

SAMPLE_RATE(SAMPLE_RATE),

BIT_SLICE_OFFSET(BIT_SLICE_OFFSET),

INVERT_DATA(INVERT_DATA),

SAMPLE_SELECT(SAMPLE_SELECT)
) inst_axis_1553_decoder ( .aclk(clk), .arstn(rstn), .m_axis_tdata(m_axis_tcl)
```

Decode incoming differential 1553 data stream to AXIS data format.

# 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),

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),

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.