# up_1553.v

## AUTHORS

## JAY CONVERTINO

## DATES

## 2024/10/17

## INFORMATION

### Brief

uP Core for interfacing with simple 1553 communications.

### License MIT



### 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
 =
0,
parameter
INVERT_DATA
 =
0,
parameter
SAMPLE_SELECT
 =
0
) ( input clk, input rstn, input up_rreq, output up_rack, input [ADDRESS_WI
```

uP based 1553 communications device.

## Parameters

| | |
|---|---|
| **ADDRESS_WIDTH**<br>parameter | Width of the uP address port. |
| **BUS_WIDTH**<br>parameter | Width of the uP bus data port. |
| **CLOCK_SPEED**<br>parameter | This is the aclk frequency in Hz |
| **SAMPLE_RATE**<br>parameter | Rate of in which to sample the 1553 bus. Must be 2 MHz or more and less than aclk. This is in Hz. BIT_SLICE_OFFSET- Adjust where the sample is taken from the input. |
| **INVERT_DATA**<br>parameter | Invert all 1553 bits coming in and out. |
| **SAMPLE_SELECT**<br>parameter | Adjust where in the array of samples to select a bit. |

## 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 |
| **i_diff** | Input differential signal for 1553 bus |
| **o_diff** | Output differential signal for 1553 bus |
| **en_o_diff** | Enable output of differential signal (for signal switching on 1553 module) |
| **irq** | Interrupt when data is received |

## 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 | PC | DI | Delay | irq_en | tx_full | tx_empty | rx_full | rx_valid |

## Status Register Bits

**PC** 7, Parity check passed?

**DI** 6, Build time option to invert data from the core, 1 is active.

**Delay** 5, Message had a 4uS delay.

**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
```

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_1553_encoder

```
axis_1553_encoder #(
                                                                    .
CLOCK_SPEED(CLOCK_SPEED),
                                                                    .
SAMPLE_RATE(SAMPLE_RATE)
) inst_axis_1553_encoder ( .aclk(clk), .arstn(rstn), .s_axis_tdata(tx_rdata[
```

Encode incoming AXIS data into a differential 1553 data stream

## 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_td
```

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),
                                                                    .
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.