

# 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
)
```

```

    input
    clk,
    input
    rstn,
    input
    up_rreq,
    output
    up_rack,
    input
    [ADDRESS_WIDTH-(BUS_WIDTH/2)-1:0]
    up_raddr,
    output
    [(BUS_WIDTH*8)-1:0]
    up_rdata,
    input
    up_wreq,
    output
    up_wack,
    input
    [ADDRESS_WIDTH-(BUS_WIDTH/2)-1:0]
    up_waddr,
    input
    [(BUS_WIDTH*8)-1:0]
    up_wdata,
    input
    [1:0]
    rx_diff,
    output
    [1:0]
    tx_diff,
    output
    tx_active,
    output
    irq
)

```

uP based 1553 communications device.

## Parameters

|                                   |   |
|-----------------------------------|---|
| <b>ADDRESS_WIDTH</b><br>parameter | Width of the uP address port, max 32 bit. |
| <b>BUS_WIDTH</b><br>parameter     | Width of the uP bus data port.            |
| <b>CLOCK_SPEED</b><br>parameter   | This is the ack frequency in Hz           |

## Ports

|  |                                   |
|--|-----------------------------------|
| <b>clk</b><br>input  | Clock for all devices in the core |
| <b>rstn</b><br>input   | Negative reset                    |
| <b>up_rreq</b><br>input                                      | uP bus read request               |
| <b>up_rack</b><br>output                                     | uP bus read ack                   |
| <b>up_raddr</b><br>input [ADDRESS_WIDTH-(BUS_WIDTH/ 2)- 1:0] | uP bus read address               |
| <b>up_rdata</b><br>output [(BUS_WIDTH* 8)- 1:0]              | uP bus read data                  |

|   |   |
|---|---|
| <b>up_wreq</b><br><i>input</i> [(BUS_WIDTH* 8)- 1:0]                | uP bus write request  |
| <b>up_wack</b><br><i>output</i> [(BUS_WIDTH* 8)- 1:0]               | uP bus write ack  |
| <b>up_waddr</b><br><i>input</i> [ADDRESS_WIDTH-(BUS_WIDTH/ 2)- 1:0] | uP bus write address  |
| <b>up_wdata</b><br><i>input</i> [(BUS_WIDTH* 8)- 1:0]               | uP bus write data   |
| <b>rx_diff</b><br><i>input</i> [1:0]                                | Input differential signal for 1553 bus  |
| <b>tx_diff</b><br><i>output</i> [1:0]                               | Output differential signal for 1553 bus   |
| <b>tx_active</b><br><i>output</i> [1:0]                             | Enable output of differential signal as this indicates tx is active (for signal switching on 1553 module) |
| <b>irq</b><br><i>output</i> [1:0]                                   | Interrupt when data is received   |

## 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

## DATA\_BITS

```
localparam DATA_BITS = 21
```

Number of bits in RX/TX FIFO that are valid.

## REGISTER INFORMATION

Core has 4 registers at the offsets that follow.

|                    |    |
|--------------------|----|
| <b>RX_FIFO_REG</b> | h0 |
| <b>TX_FIFO_REG</b> | h4 |
| <b>STATUS_REG</b>  | h8 |
| <b>CONTROL_REG</b> | hC |

## RX\_FIFO\_REG

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

Defines the address offset for RX FIFO

| RX FIFO REGISTER |             |               |
|------------------|-------------|---------------|
| 31:20            | 20:16       | 15:0          |
| UNUSED           | STATUS DATA | RECEIVED DATA |

Valid bits are from 20:0. Bits 20:16 are status bits information about the data. Bit 15:0 are data.

#### Status Bits

{S:1,D:1,TY:3}

**TY** Type is 3 bits, 000 NA, 001 = REG, 010 = DATA, 100 = CMD/STATUS

**D** Delay Enabled is 1 bit, 1 is there was be a delay of 4 us or more, or 0 no delay.

**S** Sync only when 1, normal is 0

#### TX\_FIFO\_REG

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

Defines the address offset to write the TX FIFO.

| TX FIFO REGISTER |             |               |
|------------------|-------------|---------------|
| 31:20            | 20:16       | 15:0          |
| UNUSED           | STATUS DATA | TRANSMIT DATA |

Valid bits are from 20:0. Bits 20:16 are status bits information about the data. Bit 15:0 are data.

#### Status Bits

{S:1,D:1,TY:3}

**TY** Type is 3 bits, 000 NA, 001 = REG, 010 = DATA, 100 = CMD/STATUS

**D** Delay Enabled is 1 bit, 1 is there must be a delay of 4 us or more, or 0 no delay.

**S** Sync only when 1, normal is 0

#### 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          | Parity err | frame err | rx hold en | irq en | tx full | tx empty | rx full | rx vali |

#### Status Register Bits

**parity\_err** 7, When 1 an error in the RX parity check has occurred

|                   |   |
|-------------------|---|
| <b>frame_err</b>  | 6, When 1 an error in the RX frame has occurred (manchester data 2'b11 or 2'b00). |
| <b>rx_hold_en</b> | 5, When 1 the RX HOLD is enable by <b>CONTROL_REG</b>                             |
| <b>irq_en</b>     | 4, When 1 the IRQ is enabled by <b>CONTROL_REG</b>                                |
| <b>tx_full</b>    | 3, When 1 the tx fifo is full.  |
| <b>tx_empty</b>   | 2, When 1 the tx fifo is empty.   |
| <b>rx_full</b>    | 1, When 1 the rx fifo is full.  |
| <b>rx_valid</b>   | 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 | ENA_RX_HOLD_BIT | UNUSED | RST_RX_BIT | RST_TX_BIT |

See Also: **ENABLE\_INTR\_BIT**, **RESET\_RX\_BIT**, **RESET\_TX\_BIT**

## Control Register Bits

|                           |   |
|---------------------------|---|
| <b>ENABLE_INTR_BIT</b>    | 4, Control Register offset bit for enabling the interrupt.        |
| <b>ENABLE_RX_HOLD_BIT</b> | 3, Control that RX will hold its clock on non diffs for a moment. |
| <b>RESET_RX_BIT</b>       | 1, Control Register offset bit for resetting the RX FIFO.         |
| <b>RESET_TX_BIT</b>       | 0, Control Register offset bit for resetting the TX FIFO.         |

## inst\_axis\_1553

Decode/Encode differential 1553 data stream to AXIS data format.

## inst\_rx\_fifo

Buffer up to 16 items output from the axis\_1553\_encoder.

## inst\_tx\_fifo

Buffer up to 16 items to input to the axis\_1553\_decoder.