

uart_1553_core.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/28

INFORMATION

Brief

Core that ties together all ips into a single uart to 1553 core.

License MIT

Copyright 2021 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.

uart_1553_core

```
module uart_1553_core #(
    parameter
    clock_speed
    =
    2000000,
    parameter
    uart_baud_clock_speed
    =
    2000000,
    parameter
    uart_baud_rate
```

```

    =
    2000000,
    parameter
    uart_parity_ena
    =
    0,
    parameter
    uart_parity_type
    =
    0,
    parameter
    uart_stop_bits
    =
    1,
    parameter
    uart_data_bits
    =
    8,
    parameter
    uart_rx_delay
    =
    0,
    parameter
    uart_tx_delay
    =
    0,
    parameter
    mil1553_sample_rate
    =
    2000000,
    parameter
    mil1553_rx_bit_slice_offset
    =
    0,
    parameter
    mil1553_rx_invert_data
    =
    0,
    parameter
    mil1553_rx_sample_select
    =
    0
) ( input aclk, input arstn, input uart_clk, input uart_rstn, input rx_UART,

```

Core that ties together all ips into a single uart to 1553 core.

Parameters

clock_speed = 2000000	Requested Master Clock Speed from clk wiz
uart_baud_clock_speed parameter	UART Master Clock Speed
uart_baud_rate parameter	UART BAUD rate
uart_parity_ena parameter	UART Parity enable, active high.
uart_parity_type parameter	UART Parity type
uart_stop_bits parameter	UART Number of stop bits.
uart_data_bits parameter	UART Number of data bits.

uart_rx_delay parameter	UART RX Delay to align data.
uart_tx_delay parameter	UART TX Delay to align data.
mil1553_sample_rate parameter	Sample rate for 1553, must be 2 MHz or above, and divide evenly into clock_speed.
mil1553_rx_bit_slice_offset parameter	1553 change the offset of the receive bit taken from the initial sampling.
mil1553_rx_invert_data parameter	Invert 1553 data received.
mil1553_rx_sample_select parameter	1553 select sample from initial sampling.

Ports

aclk	Master Clock
arstn	Base Reset
uart_clk	UART Master Clock
uart_rstn	UART reset
rx_UART	UART RX input
tx_UART	UART TX output
rts_UART	UART request to send
cts_UART	UART clear to send
rx0_1553	PMOD1553 RX diff
rx1_1553	PMOD1553 RX diff
tx0_1553	PMOD1553 TX diff
tx1_1553	PMOD1553 TX diff
en_tx_1553	PMOD1553 enable transmit on mux.

INSTANTIATED MODULES

mil1553_decoder

```
axis_1553_decoder #(
    CLOCK_SPEED(clock_speed),
    SAMPLE_RATE(mil1553_sample_rate),
    BIT_SLICE_OFFSET(mil1553_rx_bit_slice_offset),
    INVERT_DATA(mil1553_rx_invert_data),
    SAMPLE_SELECT(mil1553_rx_sample_select)
) mil1553_decoder ( .aclk(aclk), .arstn(arstn), .m_axis_tdata(m1553_decoder_
```

Module mil-std-1553 decoder capable of any clock rate at or above 2 MHz

decoder_fifo

```
axis_fifo #(
    FIFO_DEPTH                *
    256),                      (
    COUNT_WIDTH               *
    0),                        (
    BUS_WIDTH                 *
    2),                        (
    USER_WIDTH                *
    8),                        (
    DEST_WIDTH                *
    1),                        (
    RAM_TYPE                  *
    block"),                  ("
    PACKET_MODE               *
    0),                        (
    COUNT_DELAY               *
    0),                        (
    COUNT_ENA                 *
    0)
) decoder_fifo ( .s_axis_aclk(aclk), .s_axis_arstn(arstn), .s_axis_tvalid(m
```

FIFO for decoder data output

string_encoder

```
axis_1553_string_encoder string_encoder (
    .
    aclk(aclk),               .
    arstn(arstn),             .
    s_axis_tdata(mfifo_decoder_data),
    s_axis_tvalid(mfifo_decoder_valid),
    s_axis_tuser(mfifo_decoder_user),
    s_axis_tready(mfifo_decoder_ready),
    m_axis_tdata(mstring_encoder_data),
    m_axis_tvalid(mstring_encoder_valid),
```

```
m_axis_tready(mstring_encoder_ready)
)
```

1553 to string core

string_to_char

```
axis_data_width_converter #(
    SLAVE_WIDTH(21),
    MASTER_WIDTH(1),
    REVERSE(1)
) string_to_char ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(mstring_encoder
```

data width converter

outgoing_char_fifo

```
axis_tiny_fifo #(
    FIFO_DEPTH(4),
    BUS_WIDTH(8)
) outgoing_char_fifo ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(mstring_to
```

fifo for 1553 encoded into character string.

string_to_char

AXIS UART

incomming_char_fifo

```
axis_tiny_fifo #(
    FIFO_DEPTH(4),
    BUS_WIDTH(8)
) incomming_char_fifo ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(muart_char
```

fifo for chars to be decoded into 1553 data.

char_to_string

```
axis_char_to_string_converter #(
    master_width(21)
) char_to_string ( .aclk(aclk), .arstn(arstn), .s_axis_tdata(min_char_fifo
```

data width converter

char_to_string

```
axis_1553_string_decoder string_decoder (  
    aclk(aclk),  
    arstn(arstn),  
    s_axis_tdata(mchar_to_string_data),  
    s_axis_tvalid(mchar_to_string_valid),  
    s_axis_tready(mchar_to_string_ready),  
    m_axis_tdata(mstring_decoder_data),  
    m_axis_tvalid(mstring_decoder_valid),  
    m_axis_tuser(mstring_decoder_user),  
    m_axis_tready(mstring_decoder_ready)  
)
```

string to 1553

encoder_fifo

```
axis_fifo #(  
    FIFO_DEPTH  
    256),  
    COUNT_WIDTH  
    0),  
    BUS_WIDTH  
    2),  
    USER_WIDTH  
    8),  
    DEST_WIDTH  
    1),  
    RAM_TYPE  
    block"),  
    PACKET_MODE  
    0),  
    COUNT_DELAY
```

```

0),
COUNT_ENA
0)
) encoder_fifo ( .s_axis_aclk(aclk), .s_axis_arstn(arstn), .s_axis_tvalid(ms

```

fifo for decoder data

encoder_fifo

mil-std-1553 encoder capable of any clock rate at or over 2 MHz