axis_1553.v

AUTHORS

JAY CONVERTINO

DATES

2025/06/24

INFORMATION

Brief

AXIS 1553 core

License MIT

axis_1553

```
module axis_1553 #(
parameter
CLOCK_SPEED
  =
2000000,
parameter
RX_BAUD_DELAY
  =
0,
parameter
TX_BAUD_DELAY
  =
0
) ( input wire aclk, input wire arstn, output wire parity_err, output wire t
```

AXIS 1553, simple core for encoding and decoding 1553 bus messages.

Parameters

CLOCK_SPEED	This is the aclk frequency in Hz	
parameter		
RX_BAUD_DELAY	Delay in rx baud enable. This will delay when we sample a bit (default is midpoint when rx delay is 0).	
parameter		
TX_BAUD_DELAY	Delay in tx baud enable. This will delay the time the bit output starts.	
parameter		

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
parity_err	Indicates error with parity check for receive (active high)
frame_err	Indicates the diff line went to no diff before data catpure finished.
s_axis_tdata	Input data for UART TX.
s_axis_tuser	Information about the AXIS data {S,D,TYY} (4:0)
	Bits explained below:

```
- S   = SYNC ONLY (4)
      - 1 = Send only a sync pulse specified by TYY
      - 0 = Send normal sync + data.
- D   = DELAY ENABLED (3)
      - 1 = Make sure there is a delay of 4us
      - 0 = Send out immediatly
- TYY = TYPE OF DATA  (2:0)
      - 000 = NA
      - 001 = REG (NOT IMPLIMENTED)
      - 010 = DATA
      - 100 = CMD/STATUS
```

s_axis_tvalid - When set active high the input data is valid s_axis_tready - When active high the device is ready for input data. m_axis_tdata - Output data from UART RX m_axis_tuser - Information about the AXIS data {S,D,TYY} (4:0)

Bits explained below:

```
- S   = SYNC ONLY (4)
      - 1 = Only received a sync pulse specified by TYY
      - 0 = Normal sync + data received.
- D   = DELAY BEFORE DATA (3)
      - 1 = Delay of 4us or more before data
      - 0 = No delay between data
- TYY = TYPE OF DATA      (2:0)
      - 000 NA
      - 001 REG (NOT IMPLIMENTED)
      - 010 DATA
      - 100 CMD/STATUS
```

m_axis_tvalid - When active high the output data is valid m_axis_tready - When set active high the output device is ready for data. tx_active - Active high indicates transmit is in progress. tx_diff - transmit for 1553 (output to RX) rx_diff - receive for 1553 (input from TX)

BASE_1553_CLOCK_RATE

```
localparam integer BASE_1553_CLOCK_RATE = 1000000
```

1553 base clock rate

BASE_1553_SAMPLE_RATE

```
localparam integer BASE_1553_SAMPLE_RATE = 2000000
```

Sample rate to use for the 1553 bus, set to 2 MHz

SAMPLES_PER_MHZ

```
localparam integer SAMPLES_PER_MHZ = BASE_1553_SAMPLE_RATE /
   BASE_1553_CLOCK_RATE
```

sample rate of 2 MHz to caputre transmission bits at

cycles_per_mhz

```
localparam integer CYCLES_PER_MHZ = CLOCK_SPEED / BASE_1553_CLOCK_RATE
```

calculate the number of cycles the clock changes per period

BIT_RATE_PER_MHZ

```
localparam integer BIT_RATE_PER_MHZ = SAMPLES_PER_MHZ
```

bit rate per mhz

DELAY_TIME

```
localparam integer DELAY_TIME = CYCLES_PER_MHZ * 4
```

delay time, 4 is for 4 us (min 1553 time)

SYNC_BITS_PER_TRANS

```
localparam integer SYNC_BITS_PER_TRANS = 3
```

sync bits per transmission

SYNTH_SYNC_BITS_PER_TRANS

```
localparam integer SYNTH_SYNC_BITS_PER_TRANS = SYNC_BITS_PER_TRANS *
   BIT_RATE_PER_MHZ
```

sync pulse length

PARITY_BITS_PER_TRANS

```
localparam integer PARITY_BITS_PER_TRANS = 1
```

parity bits per transmission

SYNTH_PARITY_BITS_PER_TRANS

```
localparam integer SYNTH_PARITY_BITS_PER_TRANS = PARITY_BITS_PER_TRANS *
   BIT_RATE_PER_MHZ
```

synth parity bits per transmission

DATA_BITS_PER_TRANS

```
localparam integer DATA_BITS_PER_TRANS = 16
```

data bits per transmission

SYNTH_DATA_BITS_PER_TRANS

```
localparam integer SYNTH_DATA_BITS_PER_TRANS = DATA_BITS_PER_TRANS *
   BIT_RATE_PER_MHZ
```

synth data bits per transmission

BITS_PER_TRANS

```
localparam integer BITS_PER_TRANS = DATA_BITS_PER_TRANS +
   PARITY_BITS_PER_TRANS
```

non sync bits per transmission

TOTAL_BITS_PER_TRANS

```
localparam integer TOTAL_BITS_PER_TRANS = DATA_BITS_PER_TRANS +
   PARITY_BITS_PER_TRANS + SYNC_BITS_PER_TRANS
```

bits per transmission with sync

SYNTH_BITS_PER_TRANS

```
localparam integer SYNTH_BITS_PER_TRANS = SYNTH_DATA_BITS_PER_TRANS +
   SYNTH_PARITY_BITS_PER_TRANS
```

synth bits per trans without sync

TOTAL_SYNTH_BITS_PER_TRANS

```
localparam integer TOTAL_SYNTH_BITS_PER_TRANS =
   SYNTH_DATA_BITS_PER_TRANS + SYNTH_SYNC_BITS_PER_TRANS +
   SYNTH_PARITY_BITS_PER_TRANS
```

synth bits per trans with sync

TOTAL_SYNTH_BYTES_PER_TRANS

```
localparam integer TOTAL_SYNTH_BYTES_PER_TRANS =
   TOTAL_SYNTH_BITS_PER_TRANS/8
```

sync bits per trans with sync

BIT_PATTERN

```
localparam [(
BIT_RATE_PER_MHZ
)-1:0]BIT_PATTERN = {{BIT_RATE_PER_MHZ/2{1'b1}}, {BIT_RATE_PER_MHZ/2{1'b0}}
```

create the bit pattern. This is based on outputing data on the negative and positive. This allows the encoder to run down to 1 mhz.

SYNTH_CLK

```
localparam [SYNTH_DATA_BITS_PER_TRANS-1:0]SYNTH_CLK = {
DATA_BITS_PER_TRANS{BIT_PATTERN}
 }
```

synth clock is the clock constructed by the repeating the bit pattern. this is intended to be a representation of the clock. Captured at a bit_rate_per_mhz of a 1mhz clock.

SYNC_CMD_STAT

```
localparam [SYNTH_SYNC_BITS_PER_TRANS-1:0]SYNC_CMD_STAT = {
                                                          {
SYNTH_SYNC_BITS_PER_TRANS/2{1'b1}},
                                                          {
SYNTH_SYNC_BITS_PER_TRANS/2{1'b0}}
}
```

sync pulse command

SYNC_DATA

```
localparam [SYNTH_SYNC_BITS_PER_TRANS-1:0]SYNC_DATA = {
                                                          {
```

```
SYNTH_SYNC_BITS_PER_TRANS/2{1'b0}},
                                                                                {
SYNTH_SYNC_BITS_PER_TRANS/2{1'b1}}
}
```

sync pulse data

CMD_DATA

```
localparam CMD_DATA = 3'b010
```

tuser decode for data

CMD_DATA

tuser decode for command

INSTANTIATED MODULES

uart_baud_gen_tx

Generates TX BAUD rate for UART modules using modulo divide method.

uart_baud_gen_rx

```
mod_clock_ena_gen #(
                                                                                .
CLOCK_SPEED(CLOCK_SPEED),
                                                                                .
DELAY(RX_BAUD_DELAY)
) uart_baud_gen_rx ( .clk(aclk), .rstn(arstn), .start0(1'b0), .clr(s_clr_clk
```

Generates RX BAUD rate for UART modules using modulo divide method.

inst_sipo

```
sipo #(
                                                                                .
BUS_WIDTH(TOTAL_SYNTH_BYTES_PER_TRANS)
) inst_sipo ( .clk(aclk), .rstn(arstn), .ena(ena_rx), .rev(1'b0), .load(r_
```

Captures RX data for 1553 receive

inst_piso

```
piso #(
                                                                                .
BUS_WIDTH(TOTAL_SYNTH_BYTES_PER_TRANS),
                                                                                .
DEFAULT_RESET_VAL(0),
```

```
                                                                        .
 DEFAULT_SHIFT_VAL(0)
) inst_piso ( .clk(aclk), .rstn(arstn), .ena(ena_tx), .rev(1'b0), .load(s_ax
```

Generates TX data for 1553 transmit