AXIS_DATA_TO_AXIS_STRING

AIR FORCE RESEARCH LABORATORY

May 20, 2025

Jay Convertino

Contents

1 Usage

1.1 Introduction

All data is converted to a string with prefixs for each of the 3 input ports. All strings are terminated with a single byte, and each has a delimiter between them. All output characters will be feed out from the buffer till it is exhausted. While the core is outputing data it will not be ready to take in any other data. There is no down clock cycle time between output and input, outside of the time to output the total amount in the buffer. Essentially no wasted cycles.

1.2 Dependencies

The following are the dependencies of the cores.

- fusesoc 2.X
- iverilog (simulation)
- cocotb (simulation)

1.2.1 fusesoc_info Depenecies

- dep
  - AFRL:utility:helper:1.0.0
- dep_tb
  - AFRL:simulation:axis_stimulator
  - AFRL:simulation:clock_stimulator
  - AFRL:utility:sim_helper

1.3 In a Project

Simply use this core between a sink and source AXIS devices. This will convert from input data into an output string one character at a time. Check the code to see if others will work correctly.

2 Architecture

The only module is the axis_data_to_axis_string module. It is listed below.

- **axis_data_to_axis_string** Implement an algorithm to convert input data to ASCII string (see core for documentation).

The only always process converts the input data to a stream of ASCII strings.

1. If the counter has data and the desitination device is ready, output data and decrement buffer counter.
2. If the input data is valid and the counter is 0, meaning previous ASCII string is exhausted, take input data and create ASCII string.
   (a) Insert data prefix into buffer.
   (b) Using unrolled for loop encode tdata input into ASCII HEX and insert into buffer (4 bit nibbles 0 to F).
   (c) Insert delimiter into buffer.
   (d) Insert user prefix into buffer.
   (e) Using unrolled for loop encode tuser input into ASCII HEX and insert into buffer (4 bit nibbles 0 to F).
   (f) Insert delimiter into buffer.
   (g) Insert destination prefix into buffer.
   (h) Using unrolled for loop encode tdest input into ASCII HEX and insert into buffer (4 bit nibbles 0 to F).
   (i) Insert string termination into buffer.
   (j) Counter is set to string length.

Please see **??** for more information.

3 Building

The AXIS data to AXIS string core is written in Verilog 2001. They should synthesize in any modern FPGA software. The core comes as a fusesoc packaged core and can be included in any other core. Be sure to make sure you have meet the dependencies listed in the previous section. Linting is performed by verible using the lint target.

3.1 fusesoc

Fusesoc is a system for building FPGA software without relying on the internal project management of the tool. Avoiding vendor lock in to Vivado or Quartus. These cores, when included in a project, can be easily integrated and targets created based upon the end developer needs. The core by itself is not a part of a system and should be integrated into a fusesoc based system. Simulations are setup to use fusesoc and are a part of its targets.

3.2 Source Files

3.2.1 fusesoc_info File List

- src
  - ’src/axis_data_to_axis_string.v’: ’file_type’: ’verilogSource’
- tb
  - ’tb/tb_axis.v’: ’file_type’: ’verilogSource’
- tb_cocotb
  - ’tb/tb_cocotb.py’: ’file_type’: ’user’, ’copyto’: ’.’
  - ’tb/tb_cocotb.v’: ’file_type’: ’verilogSource’

3.3 Targets

3.3.1 fusesoc_info Targets

- default

  Info: Default for IP intergration.
- lint

  Info: Lint with Verible
- sim

  Info: Default simulation with const data.
- sim_8bit_count_data

  Info: Counter data input.
- sim_rand_ready_8bit_count_data

  Info: Counter data input, and random ready input.
- sim_cocotb

  Info: Cocotb unit tests

3.4 Directory Guide

Below highlights important folders from the root of the directory.

1. **docs** Contains all documentation related to this project.
   - **manual** Contains user manual and github page that are generated from the latex sources.
2. **src** Contains source files for the core
3. **tb** Contains test bench files for iverilog and cocotb

4 Simulation

There are a few different simulations that can be run for this core. All currently use iVerilog (icarus) to run. The first is iverilog, which uses verilog only for the simulations. The other is cocotb. This does a unit test approach to the testing and gives a list of tests that pass or fail.

4.1 iverilog

All simulation targets that do NOT have cocotb in the name use a verilog test bench with verilog stimulus components. These all read in a file and then write a file that has been processed by the data width converter. Then the input and output file are compared with a MD5 sum to check that they match. If they do not match then the test has failed. All of these tests provide fst output files for viewing the waveform in the there target build folder.

4.2 cocotb

To use the cocotb tests you must install the following python libraries.

```
$ pip install cocotb
$ pip install cocotbext-axi
```

Then you must use the cocotb sim target. In this case it is sim_cocotb. This target can be run with various bus and fifo parameters.

```
$ fusesoc run --target=sim_cocotb AFRL:string:
    ↪ axis_data_to_axis_string:1.0.0 --SBUS_WIDTH=4 --
    ↪ USER_WIDTH=8
```

The following is an example command to run through various parameters without typing them one by one.

```
$ for i in {1..32}; do sleep 5; fusesoc run --target
    ↪ sim_cocotb AFRL:string:axis_data_to_axis_string
    ↪ :1.0.0 --SBUS_WIDTH=$i ; echo "SLAVE␣WIDTH:" $i;
    ↪ done
```

5 Code Documentation

Natural docs is used to generate documentation for this project. The next lists the following sections.

- **axis_data_to_axis_string** AXIS data to AXIS string, convert input data to a ASCII string.

- **tb_axis** Verilog test bench.

- **tb_cocotb verilog** Verilog test bench base for cocotb.

- **tb_cocotb python** cocotb unit test functions.

axis_data_to_axis_string.v

AUTHORS

JAY CONVERTINO

DATES

2022/09/19

INFORMATION

Brief

Parse raw binary data into ASCII string output.

License MIT

axis_data_to_axis_string

```
module axis_data_to_axis_string #(
parameter
DELIMITER
 =
";",
parameter
TERMINATION
 =
"\n",
parameter
SBUS_WIDTH
 =
1,
parameter
```

```
USER_WIDTH
 =
4,
parameter
DEST_WIDTH
 =
4,
parameter
PREFIX_LEN
 =
1,
parameter
DATA_PREFIX
 =
"#",
parameter
DEST_PREFIX
 =
"&",
parameter
USER_PREFIX
 =
"*"
) ( input aclk, input arstn, input [(SBUS_WIDTH*8)-1:0] s_axis_tdata, input
```

Parse raw binary data into ASCII string output.

Parameters

DELIMITER parameter	break value between multple strings
TERMINATION parameter	termination value of full string from serial port, byte only. (\n = 0A \r = 0D).
SBUS_WIDTH parameter	bus width of slave (data) input
USER_WIDTH parameter	user width of slave bus, only in 4 bit nibbles, and at least 4 bits.
DEST_WIDTH parameter	dest width of slave bus, only in 4 bit nibbles, and at least 4 bits.
PREFIX_LEN parameter	length of following prefix strings.
DATA_PREFIX parameter	prefix for data hex strings
DEST_PREFIX parameter	prefix for destination hex strings
USER_PREFIX parameter	prefix for user hex strings

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
s_axis_tdata	Input data
s_axis_tvalid	When set active high the input data is valid
s_axis_tuser	User data to convert.
s_axis_tdest	Destination data to convert
s_axis_tready	When active high the device is ready for input data.
m_axis_tdata	Output data

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.

VARIABLES

s_axis_tready

```
assign s_axis_tready = (
                                                              (counter == 0) ? 1 &
arstn
:
0
)
```

ready if count is zero, this is a FWFT so no worries in pumping out data.

m_axis_tdata

```
assign m_axis_tdata = char_buffer[STRING_LEN*8-1 -:8]
```

output whatever is in the character buffer.

m_axis_tvalid

```
assign m_axis_tvalid = (
                                                                     counter > 0 ?
1
:
0
)
```

Counter greater than 0? Valid output is available.

tb_axis.v

AUTHORS

JAY CONVERTINO

DATES

2022/10/24

INFORMATION

Brief

Test bench for axis_data_to_axis_string using axis stim and clock stim.

License MIT

tb_axis

```
 module tb_axis #(
 parameter
 IN_FILE_NAME
  =
 "in.bin",
 parameter
 OUT_FILE_NAME
  =
 "out.bin",
 parameter
 RAND_READY
  =
 0
)
```

Test bench for axis_data_to_axis_string. This will run a file through the system and write its output. These can then be compared to check for errors. If the files are identical, no errors. A FST file will be written.

Parameters

IN_FILE_NAME parameter	File name for input.
OUT_FILE_NAME parameter	File name for output.
RAND_READY parameter	0 = no random ready. 1 = randomize ready.

INSTANTIATED MODULES

clk_stim

```
 clk_stimulus #(
                                                                      .
 CLOCKS(1),
                                                                      .
 CLOCK_BASE(1000000),
                                                                      .
 CLOCK_INC(1000),
                                                                      .
 RESETS(1),
                                                                      .
 RESET_BASE(2000),
                                                                      .
 RESET_INC(100)
) clk_stim ( .clkv(tb_dut_clk), .rstnv(tb_dut_rstn), .rstv() )
```

Generate a 50/50 duty cycle set of clocks and reset.

slave_axis_stim

```
 slave_axis_stimulus #(
                                                                      .
 BUS_WIDTH(BUS_WIDTH),
                                                                      .
 USER_WIDTH(USER_WIDTH),
                                                                      .
 DEST_WIDTH(DEST_WIDTH),
                                                                      .
 FILE(IN_FILE_NAME)
) slave_axis_stim ( .m_axis_aclk(tb_dut_clk), .m_axis_arstn(tb_dut_rstn), .r
```

Device under test SLAVE stimulus module.

dut

```
 axis_data_to_axis_string #(
                                                                      .
 DELIMITER(";"),
                                                                      .
 TERMINATION("\n"),
                                                                          .
 SBUS_WIDTH(BUS_WIDTH),
                                                                          .
 USER_WIDTH(USER_WIDTH),
                                                                          .
 DEST_WIDTH(DEST_WIDTH),
                                                                          .
 PREFIX_LEN(1),
                                                                          .
 DATA_PREFIX("#"),
                                                                          .
 DEST_PREFIX("&"),
                                                                          .
 USER_PREFIX("*")
) dut ( .aclk(tb_dut_clk), .arstn(tb_dut_rstn), .m_axis_tdata(tb_dut_data),
```

Device under test, axis_data_width_converter

master_axis_stim

```
 master_axis_stimulus #(
                                                                          .
 BUS_WIDTH(BUS_WIDTH),
                                                                          .
 USER_WIDTH(USER_WIDTH),
                                                                          .
 DEST_WIDTH(DEST_WIDTH),
                                                                          .
 RAND_READY(RAND_READY),
                                                                          .
 FILE(OUT_FILE_NAME)
) master_axis_stim ( .s_axis_aclk(tb_dut_clk), .s_axis_arstn(tb_dut_rstn),
```

Devie under test MASTER stimulus module.

tb_cocotb.v

AUTHORS

JAY CONVERTINO

DATES

2024/12/12

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

tb_cocotb

```
module tb_cocotb #(
parameter
DELIMITER
 =
";",
parameter
TERMINATION
 =
"\n",
parameter
SBUS_WIDTH
 =
1,
parameter
```

```
USER_WIDTH
 =
4,
parameter
DEST_WIDTH
 =
4,
parameter
PREFIX_LEN
 =
1,
parameter
DATA_PREFIX
 =
"#",
parameter
DEST_PREFIX
 =
"&",
parameter
USER_PREFIX
 =
"*"
) ( input aclk, input arstn, input [(SBUS_WIDTH*8)-1:0] s_axis_tdata, input
```

Test bench for data to string converter. This will run a file through the system and write its output. These can then be compared to check for errors. If the files are identical, no errors. A FST file will be written.

Parameters

DELIMITER parameter	break value between multple strings
TERMINATION parameter	termination value of full string from serial port, byte only. (\n = 0A \r = 0D).
SBUS_WIDTH parameter	bus width of master (data) output
USER_WIDTH parameter	user width of master bus, only in 4 bit nibbles, and at least 4 bits.
DEST_WIDTH parameter	dest width of master bus, only in 4 bit nibbles, and at least 4 bits.
PREFIX_LEN parameter	length of following prefix strings.
DATA_PREFIX parameter	prefix for data hex strings
DEST_PREFIX parameter	prefix for destination hex strings
USER_PREFIX parameter	prefix for user hex strings

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
s_axis_tdata	Input data
s_axis_tvalid	When set active high the input data is valid
s_axis_tuser	User data to convert.
s_axis_tdest	Destination data to convert
s_axis_tready	When active high the device is ready for input data.

m_axis_tdata	Output data
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.

INSTANTIATED MODULES

dut

```
 axis_data_to_axis_string #(
                                                                          .
 DELIMITER(DELIMITER),
                                                                          .
 TERMINATION(TERMINATION),
                                                                          .
 SBUS_WIDTH(SBUS_WIDTH),
                                                                          .
 USER_WIDTH(USER_WIDTH),
                                                                          .
 DEST_WIDTH(DEST_WIDTH),
                                                                          .
 PREFIX_LEN(PREFIX_LEN),
                                                                          .
 DATA_PREFIX(DATA_PREFIX),
                                                                          .
 DEST_PREFIX(DEST_PREFIX),
                                                                          .
 USER_PREFIX(USER_PREFIX)
) dut ( .aclk(aclk), .arstn(arstn), .m_axis_tdata(m_axis_tdata), .m_axis_tva
```

Device under test, axis_data_to_axis_string

tb_cocotb.py

AUTHORS

JAY CONVERTINO

DATES

2024/12/09

INFORMATION

Brief

Cocotb test bench

License MIT

FUNCTIONS

create_string

```
def create_string(
dut,
tx_frame
)
```

Return a string equal to the core output

Parameters

dut	device under test passed from cocotb test function

tx_frame	transmitted frame data

Returns: String

random_bool

```
def random_bool()
```

Return a infinte cycle of random bools

Returns: List

start_clock

```
def start_clock(
dut
)
```

Start the simulation clock generator.

Parameters

dut	Device under test passed from cocotb test function

reset_dut

```
async def reset_dut(
dut
)
```

Cocotb coroutine for resets, used with await to make sure system is reset.

Parameters

dut	Device under test passed from cocotb.

conversion_test

```
@cocotb.test()
async def conversion_test(
dut
)
```

Coroutine that is identified as a test routine. This routine tests for correct output of strings by comparing to a python genrated version.

Parameters

dut	Device under test passed from cocotb.

conversion_test_random_ready

```
@cocotb.test()
async def conversion_test_random_ready(
dut
)
```

Coroutine that is identified as a test routine. This routine tests for correct output of strings by comparing to a python genrated version, with ready randomized.

Parameters

dut	Device under test passed from cocotb.

in_reset

```
@cocotb.test()
async def in_reset(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if device stays in unready state when in reset.

Parameters

dut	Device under test passed from cocotb.

no_clock

```
@cocotb.test()
async def no_clock(
dut
)
```

Coroutine that is identified as a test routine. This routine tests if no ready when clock is lost and device is left in reset.

Parameters

dut	Device under test passed from cocotb.