AXIS_DATA_TO_AXIS_STRING

November 20, 2024

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 5 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.

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'

3.3 Targets

3.3.1 fusesoc_info Targets

- default

  Info: Default for IP intergration.
  - src
  - dep
- sim

  Info: Default simulation with const data.
  - src
  - dep
  - tb
  - dep_tb
  - IN_FILE_NAME
  - OUT_FILE_NAME
  - RAND_READY
- sim_rand_data

  Info: Use random data for input.
  - src
  - dep
  - tb
  - dep_tb
  - IN_FILE_NAME=random.bin
  - OUT_FILE_NAME=out_random.txt
  - RAND_READY
- sim_rand_ready_rand_data

Info: Random data for input, and random ready input.
  - src
  - dep
  - tb
  - dep_tb
  - IN_FILE_NAME=random.bin
  - OUT_FILE_NAME=out_random.txt
  - RAND_READY=1
- sim_8bit_count_data

  Info: Counter data input.
  - src
  - dep
  - tb
  - dep_tb
  - IN_FILE_NAME=8bit_count.bin
  - OUT_FILE_NAME=out_8bit_count.txt
  - RAND_READY
- sim_rand_ready_8bit_count_data

  Info: Counter data input, and random ready input.
  - src
  - dep
  - tb
  - dep_tb
  - IN_FILE_NAME=8bit_count.bin
  - OUT_FILE_NAME=out_8bit_count.txt
  - RAND_READY=1

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
   - **cocotb** testbench files

4 Simulation

There are a few different simulations that can be run for this core.

4.1 iverilog

iverilog is used for simple test benches for quick verification, visually, of the core.

4.2 cocotb

Future simulations will use cocotb. This feature is not yet implemented.

5 Module Documentation

There is a single async module for this core.

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

The next sections document the module in great detail.

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

Copyright 2022 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.

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 break value between multple strings
parameter

TERMINATION termination value of full string from serial port, byte only. (\n = 0A \r = 0D).
parameter

SBUS_WIDTH bus width of master (data) output
parameter

USER_WIDTH user width of master bus, only in 4 bit nibbles, and at least 4 bits.
parameter

DEST_WIDTH dest width of master bus, only in 4 bit nibbles, and at least 4 bits.
parameter

PREFIX_LEN length of following prefix strings.
parameter

DATA_PREFIX prefix for data hex strings
parameter

DEST_PREFIX prefix for destination hex strings
parameter

USER_PREFIX prefix for user hex strings
parameter

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.