

FIFO_STIMULATOR



November 26, 2024

Jay Convertino

Contents

1 Usage	2
1.1 Introduction	2
1.2 Dependencies	2
1.2.1 fusesoc_info Depenecies	2
2 Architecture	2
3 Building	3
3.1 fusesoc	3
3.2 Source Files	3
3.2.1 fusesoc_info File List	3
3.3 Targets	3
3.3.1 fusesoc_info Targets	3
3.4 Directory Guide	4
4 Simulation	5
5 Module Documentation	6
5.1 fifo stimulator modules	7
5.2 textbench for stimulator	10

1 Usage

1.1 Introduction

This core contains two modules. A writer, and reader that should be placed on the output, and input of the device under test. This will stream data through till it has read all data.

1.2 Dependencies

The following are the dependencies of the cores.

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

1.2.1 fusesoc_info Dependencies

- dep
 - AFRL:utility:helper:1.0.0
- dep_tb
 - AFRL:simulation:clock_stimulator
 - AFRL:utility:sim_helper
- dep_vpi
 - AFRL:vpi:binary_file_io:1.0.0

2 Architecture

The project contains two modules `write_fifo_stimulus` and `read_fifo_stimulus`. The `read_fifo_stimulus` is used to take input data from the `wr_fifo` interface (input) and write it to a file. Essentially it goes `DUT_RD_FIFO` to `read_fifo_stimulus(WR_FIFO)`. `write_fifo_stimulus` is used to read a file and push that data to the `rd_fifo` interface (output). Essentially it goes `write_fifo_stimulus(WR_FIFO)` to `DUT_RD_FIFO`.

This core uses a custom library for reading and writing files called `vpi_binary_file_io`. This library provides multithreaded file reads using a ring buffer between processes. The core will also puncture data according to its bit type. X/Z values are tossed if they are contained in a byte.

- **tm_stim_fifo** Contains two modules write_fifo_stimulus and read_fifo_stimulus.

Please see 5 for more information per target.

3 Building

The all FIFO stimulator modules are 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 met 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/tm_stim_fifo.v': 'file_type': 'verilogSource'
- tb
 - 'tb/tb_fifo.v': 'file_type': 'verilogSource'

3.3 Targets

3.3.1 fusesoc_info Targets

- default
 - Info: Default for simulation filesset.
- sim
 - Info: Default for icarus simulation.
- sim_rand_data

Info: Random data input.

- `sim_rand_full_rand_data`

Info: Random data, random ready input.

- `sim_8bit_count_data`

Info: Counter data input.

- `sim_rand_full_8bit_count_data`

Info: Counter data, random ready input.

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 `fifo_stimulator`.
3. **tb** Contains test bench files.

4 Simulation

There is no simulation at the moment. Maybe a future addition?

5 Module Documentation

There project has multiple modules. The targets are the top system wrappers.

- **tm_stim_fifo**
- **tb_fifo**

The next sections document the module in great detail.

tm_stim_fifo.v

AUTHORS

JAY CONVERTINO

DATES

2023/01/30

INFORMATION

Brief

Generic FIFO test bench modules (stimulus).

License MIT

Copyright 2023 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.

write_fifo_stimulus

```
module write_fifo_stimulus #(
    parameter
    BYTE_WIDTH
    =
    1,
    parameter
    BYTE_SWAP
    =
    0,
    parameter
    FWFT
```



```

    =
    0,
    parameter
    FILE
    =
    "test.bin"
) ( input rd_clk, input rd_rstn, input rd_en, output reg rd_valid, output

```

Simulator core to read file data, and output it to write fifo dut.

Parameters

BYTE_WIDTH parameter	data width in bytes for data bus
BYTE_SWAP parameter	swap bytes fed to the DUT
FWFT parameter	Turn on or off first word fall through mode.
FILE parameter	input file name

Ports

rd_clk	Read clock
rd_rstn	Read reset negative
rd_en	enable read
rd_valid	read data valid
rd_data	read data
rd_empty	read has no data
eof	end of input file has been reached.

read_fifo_stimulus

```

module read_fifo_stimulus #(
    parameter
    BYTE_WIDTH
    =
    1,
    parameter
    ACK_ENA
    =
    0,
    parameter
    RAND_FULL
    =
    0,
    parameter
    FILE
    =
    "out.bin"
) ( input wr_clk, input wr_rstn, input wr_en, output reg wr_ack, input [(B\

```

Simulator core to write file data, from input over write fifo dut. This module will keep a constant ready to the dut.

Parameters

BYTE_WIDTH parameter	data width in bytes for data bus
ACK_ENA parameter	enable ack if set to 1 (does nothing at the moment, ack is not used)
RAND_FULL parameter	randomize the full output signal
FILE parameter	output file name

Ports

wr_clk	Write clock
wr_rstn	Write reset negative
wr_en	enable write
wr_ack	write data has been written
wr_data	write data
wr_full	can't write data, destination is full.
eof	Exit if end of file is reached.

tb_fifo.v

AUTHORS

JAY CONVERTINO

DATES

2023/01/30

INFORMATION

Brief

Generic FIFO test bench top.

License MIT

Copyright 2023 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.

tb_axis

Generic AXIS test bench top with verification.

Parameters

IN_FILE_NAME	Name of the input file to read from.
OUT_FILE_NAME	Name of the output file to write.
RAND_FULL	Randomize the full signal from the writer (read_fifo_stimulus) core.

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_stim_clk), .rstnv(tb_stim_rstn), .rstv() )
```

Generate a clock for the modules.

write_fifo_stim

```
write_fifo_stimulus #(
    BYTE_WIDTH(BUS_WIDTH),
    FILE(IN_FILE_NAME)
) write_fifo_stim ( .rd_clk(tb_stim_clk), .rd_rstn(tb_stim_rstn), .rd_en(~tb_stim_rstn) )
```

Read a file and output to a wr_fifo interface from the read.

read_fifo_stim

```
read_fifo_stimulus #(
    BYTE_WIDTH(BUS_WIDTH),
    RAND_FULL(RAND_FULL),
    FILE(OUT_FILE_NAME)
) read_fifo_stim ( .wr_clk(tb_stim_clk), .wr_rstn(tb_stim_rstn), .wr_en(tb_stim_rstn) )
```

Write a file from the input from a rd_fifo interface to the write.