AXIS_TINY_FIFO

November 21, 2024

Jay Convertino

Contents

1 Usage **2**
- 1.1 Introduction . . . 2
- 1.2 Dependencies . . . 2
  - 1.2.1 fusesoc_info Depenecies . . . 2
- 1.3 In a Project . . . 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 . . . 4
  - 3.3.1 fusesoc_info Targets . . . 4
- 3.4 Directory Guide . . . 4

4 Simulation **5**
- 4.1 iverilog . . . 5
- 4.2 cocotb . . . 5

5 Module Documentation **6**
- 5.1 axis_tiny_fifo . . . 7

1 Usage

1.1 Introduction

Pipeline method axis fifo. This fifo uses a pipeline to create a fifo. Meaning it has a latency the size of the depth. If the output receiving core is not ready the core will build up data till it is full. All data will have a latency of the depth.

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 buffer data from one bus to another. Check the code to see if others will work correctly.

2 Architecture

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

- **axis_tiny_fifo** Implement an algorithm to convert BUS data interfaces in even multiples (see core for documentation).

  The always process is the logic behind the tiny fifo.

1. When out of reset, input new data if the output is ready and the valid buffer contains a 0 in the valid buffer. Otherwise assert current data.
2. Unrolled for loop that performs the following operations.
   (a) NAND all valids lower then current register. This results in all data being shifted below and including that index.
   (b) if any have 0 for valid (no data) shift. Also shift if ready, since destination is ready to take data anyways.

Please see 5 for more information.

3 Building

The AXIS tiny FIFO 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_tiny_fifo.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.

- sim

  Info: Constant data value with file check.

- sim_rand_data

  Info: Feed random data input with file check

- sim_rand_ready_rand_data

  Info: Feed random data input, and randomize the read ready on the output. Perform output file check.

- sim_8bit_count_data

  Info: Feed a counter data as input, perform file check.

- sim_rand_ready_8bit_count_data

  Info: Feed a counter data a input, and randomize the read ready on the output. Perform output file check.

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_tiny_fifo** AXIS tiny fifo will buffer data from input to output.

The next sections document the module in great detail.

axis_tiny_fifo.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/04

INFORMATION

Brief

AXIS TINY FIFO

License MIT

axis_tiny_fifo

```
module axis_tiny_fifo #(
parameter
FIFO_DEPTH
 =
4,
parameter
BUS_WIDTH
 =
8
) ( input aclk, input arstn, output [(BUS_WIDTH*8)-1:0] m_axis_tdata, output
```

AXIS fifo that uses a shift register to buffer data. This Adds latency to the design in the amount of the FIFO_DEPTH. Though if the destination isn't ready it will build up data to that FIFO_DEPTH and overwrite any non-valid data inserted.

Parameters

FIFO_DEPTH Number of transactions to buffer.
parameter

BUS_WIDTH Width of the input/output bus in bytes.
parameter

Ports

aclk	Clock for AXIS
arstn	Negative reset for AXIS
m_axis_tdata	Output data
m_axis_tvalid	When active high the output data is valid
m_axis_tlast	Indicates last word in stream.
m_axis_tready	When set active high the output device is ready for data.
s_axis_tdata	Input data
s_axis_tvalid	When set active high the input data is valid
s_axis_tlast	Is this the last word in the stream (active high).
s_axis_tready	When active high the device is ready for input data.

VARIABLES

s_axis_tready

```
assign s_axis_tready = ~&reg_valid_buffer || m_axis_tready
```

If any valid is 0, we are ready for data

m_axis_tdata

```
assign m_axis_tdata = reg_data_buffer[0]
```

assign output data as soon as its ready

m_axis_tvalid

```
assign m_axis_tvalid = reg_valid_buffer[0]
```

assign output data as soon as its ready

m_axis_tlast

```
assign m_axis_tlast = reg_last_buffer[0]
```

assign output data as soon as its ready