AXIS_FIFO

AFRL

AIR FORCE RESEARCH LABORATORY

December 11, 2024

Jay Convertino

Contents

1 Usage

1.1 Introduction

Standard AXIS FIFO with multiple options. The FIFO uses a AXIS interface for data in and out. It also emulates the Xilinx AXIS FIFO bugs and all. This is NOT dependent on Xilinx FPGA's and can be used on any FPGA supporting the Verilog block ram style primitive.

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:buffer:fifo
  - 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 devices. This buffer data from one bus to another. Check the code to see if others will work correctly.

2 Architecture

This AXIS FIFO is made for two modules. They are the FIFO, and AXIS FIFO control. The combination of these two provide the AXIS FIFO module. Having it made this way allows for future modules to be customized and brought in to change the FIFO's behavior. The current modules emulate the Xilinx AXIS FIFO IP core availble in Vivado 2018 and up.

AXIS FIFO control is the heart of the core when it comes to how it responds. The logic in the core is designed to emulate the Xilinx AXIS FIFO IP.

Please see 5 for more information.

3 Building

The AXIS 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_fifo.v
  - src/axis_fifo_ctrl.v
- 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.

- 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.

- 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 FIFO. 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:buffer:axis_fifo
    ↪ :1.0.0 --BUS_WIDTH=8 --FIFO_DEPTH=32
```

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; export RY=$(($RANDOM
    ↪ %32+1)); fusesoc run --target sim_cocotb AFRL:
    ↪ buffer:axis_fifo:1.0.0 --BUS_WIDTH=$i --
    ↪ FIFO_DEPTH=$RY; echo "BUS␣WIDTH:" $i "FIFO␣DEPTH:
    ↪ " $RY; done
```

5 Code Documentation

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

- **AXIS_FIFO** will buffer data from input to output.

- **AXIS_FIFO_CONTROL** emulates the Xilinx FIFO IP interface and its behavior.

- **tb_axis** Verilog test bench.

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

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

axis_fifo.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/29

INFORMATION

Brief

Wraps the standard FIFO with an axi streaming interface.

License MIT

axis_fifo

```
module axis_fifo #(
parameter
FIFO_DEPTH
 =
256,
parameter
COUNT_WIDTH
 =
8,
parameter
BUS_WIDTH
```

```
 =
1,
parameter
USER_WIDTH
 =
1,
parameter
DEST_WIDTH
 =
1,
parameter
RAM_TYPE
 =
"block",
parameter
PACKET_MODE
 =
0,
parameter
COUNT_DELAY
 =
1,
parameter
COUNT_ENA
 =
1
) ( input m_axis_aclk, input m_axis_arstn, output m_axis_tvalid, input m_ax
```

AXIS fifo

Parameters

FIFO_DEPTH
parameter
Depth of the fifo, must be a power of two number(divisable aka 256 = 2^8). Any non-power of two will be rounded up to the next closest.

COUNT_WIDTH
parameter
Data count output width in bits. Should be the same power of two as fifo depth(256 for fifo depth... this should be 8).

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

USER_WIDTH
parameter
Width of the axis user bus input/output in bits.

DEST_WIDTH
parameter
Width of the axis dest bus input/output in bits.

RAM_TYPE
parameter
RAM type setting.

PACKET_MODE
parameter
Set axis fifo to wait for tlast before allowing a read on master port output.

COUNT_DELAY
parameter
Delay count by one clock cycle of the data count clock.

COUNT_ENA
parameter
Enable count, set this to 0 to disable (only disable if read/write/data_count are on the same clock domain!).

Ports

m_axis_aclk Clock for AXIS

m_axis_arstn Negative reset for AXIS

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.

m_axis_tdata	Output data
m_axis_tkeep	Output valid byte indicator
m_axis_tlast	Indicates last word in stream.
m_axis_tuser	Output user bus
m_axis_tdest	Output destination
s_axis_aclk	Clock for AXIS
s_axis_arstn	Negative reset for AXIS
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.
s_axis_tdata	Input data
s_axis_tkeep	Input valid byte indicator
s_axis_tlast	Is this the last word in the stream (active high).
s_axis_tuser	Input user bus
s_axis_tdest	Input desitination
data_count_aclk	Clock for data count
data_count_arstn	Negative edge reset for data count.
data_count	Output that indicates the amount of data in the FIFO.

INSTANTIATED MODULES

axis_fifo

```
fifo #(
                                                                          .
FIFO_DEPTH
                                                                          (
c_FIFO_DEPTH),
                                                                          .
BYTE_WIDTH
                                                                          (
c_FIFO_WIDTH),
                                                                          .
COUNT_WIDTH
                                                                          (
COUNT_WIDTH),
                                                                          .
FWFT
                                                                          (
1),
                                                                          .
RD_SYNC_DEPTH
                                                                          (
0),
                                                                          .
WR_SYNC_DEPTH
                                                                          (
0),
                                                                          .
DC_SYNC_DEPTH
                                                                          (
0),
                                                                          .
```

```
 COUNT_DELAY
                                                                         (
 COUNT_DELAY),
                                                                         .
 COUNT_ENA
                                                                         (
 COUNT_ENA),
                                                                         .
 DATA_ZERO
                                                                         (
 1),
                                                                         .
 ACK_ENA
                                                                         (
 0),
                                                                         .
 RAM_TYPE
                                                                         (
 RAM_TYPE)
) axis_fifo ( .rd_clk (m_axis_aclk), .rd_rstn (m_axis_arstn), .rd_en (s_rd_e
```

Generic FIFO that acts like a Xilinx FIFO.

axis_control

```
 axis_fifo_ctrl #(
                                                                         .
 BUS_WIDTH
                                                                         (
 BUS_WIDTH),
                                                                         .
 FIFO_WIDTH
                                                                         (
 c_FIFO_WIDTH),
                                                                         .
 USER_WIDTH
                                                                         (
 USER_WIDTH),
                                                                         .
 DEST_WIDTH
                                                                         (
 DEST_WIDTH),
                                                                         .
 PACKET_MODE(PACKET_MODE)
 ) axis_control ( .m_axis_aclk (m_axis_aclk), .m_axis_arstn (m_axis_arstn),
```

Create signals to control FIFO and provide AXIS interace.

axis_ctrl_fifo.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/29

INFORMATION

Brief

Wraps the standard FIFO with an axi streaming interface.

License MIT

axis_fifo_ctrl

```
module axis_fifo_ctrl #(
parameter
BUS_WIDTH
 =
1,
parameter
FIFO_WIDTH
 =
8,
parameter
FIFO_POWER
```

```
 =
8,
parameter
USER_WIDTH
 =
1,
parameter
DEST_WIDTH
 =
1,
parameter
PACKET_MODE
 =
0
) ( input m_axis_aclk, input m_axis_arstn, output m_axis_tvalid, input m_ax
```

AXIS fifo control

Parameters

BUS_WIDTH parameter	Width of the axis data bus input/output in bytes. FIFO_WIDTH - FIFO_POWER -
USER_WIDTH parameter	Width of the axis user bus input/output in bits.
DEST_WIDTH parameter	Width of the axis dest bus input/output in bits.
PACKET_MODE parameter	Set axis fifo to wait for tlast before allowing a read on master port output.

Ports

m_axis_aclk	Clock for AXIS
m_axis_arstn	Negative reset for AXIS
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.
m_axis_tdata	Output data
m_axis_tkeep	Output valid byte indicator
m_axis_tlast	Indicates last word in stream.
m_axis_tuser	Output user bus
m_axis_tdest	Output destination
s_axis_tlast	Is this the last word in the stream (active high).
rd_en	Active high enable of read interface.
rd_valid	Active high output that the data is valid.
rd_data	Output data
rd_empty	Active high output when read is empty.
wr_full	Active high output that the FIFO is full.

tb_axis.v

AUTHORS

JAY CONVERTINO

DATES

2024/12/09

INFORMATION

Brief

Test bench for axis_fifo 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
FIFO_DEPTH
```

```
  =
 128,
 parameter
 RAND_READY
  =
 0
)
```

Test bench for axis_fifo. 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.

FIFO_DEPTH parameter Number of transactions to buffer.

RAND_READY parameter 0 = no random ready. 1 = randomize ready.

INSTANTIATED MODULES

clk_stim

```
 clk_stimulus #(
                                                                        .
 CLOCKS(2),
                                                                        .
 CLOCK_BASE(1000000),
                                                                        .
 CLOCK_INC(10),
                                                                        .
 RESETS(2),
                                                                        .
 RESET_BASE(2000),
                                                                        .
 RESET_INC(100)
) clk_stim ( .clkv({tb_dut_clk, tb_stim_clk}), .rstnv({tb_dut_rstn, tb_stim_
```

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_stim_clk), .m_axis_arstn(tb_stim_rstn),
```

Device under test SLAVE stimulus module.

dut

```
 axis_fifo #(
                                                                         .
 FIFO_DEPTH(FIFO_DEPTH),
                                                                         .
 COUNT_WIDTH(8),
                                                                         .
 BUS_WIDTH(BUS_WIDTH),
                                                                         .
 USER_WIDTH(USER_WIDTH),
                                                                         .
 DEST_WIDTH(DEST_WIDTH),
                                                                         .
 RAM_TYPE("block"),
                                                                         .
 PACKET_MODE(0),
                                                                         .
 COUNT_DELAY(1),
                                                                         .
 COUNT_ENA(1)
) dut ( .s_axis_aclk(tb_stim_clk), .s_axis_arstn(tb_stim_rstn), .s_axis_tval
```

Device under test, axis_fifo

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_coctb.v

AUTHORS

JAY CONVERTINO

DATES

2024/12/10

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

tb_cocotb

```
module tb_cocotb #(
parameter
FIFO_DEPTH
 =
256,
parameter
COUNT_WIDTH
 =
8,
parameter
BUS_WIDTH
```

```
 =
1,
parameter
USER_WIDTH
 =
1,
parameter
DEST_WIDTH
 =
1,
parameter
RAM_TYPE
 =
"block",
parameter
PACKET_MODE
 =
0,
parameter
COUNT_DELAY
 =
1,
parameter
COUNT_ENA
 =
1
) ( input m_axis_aclk, input m_axis_arstn, output m_axis_tvalid, input m_ax
```

Test bench for axis_fifo. 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

FIFO_DEPTH
parameter
Depth of the fifo, must be a power of two number(divisable aka 256 = 2^8). Any non-power of two will be rounded up to the next closest.

COUNT_WIDTH
parameter
Data count output width in bits. Should be the same power of two as fifo depth(256 for fifo depth... this should be 8).

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

USER_WIDTH
parameter
Width of the axis user bus input/output in bits.

DEST_WIDTH
parameter
Width of the axis dest bus input/output in bits.

RAM_TYPE
parameter
RAM type setting.

PACKET_MODE
parameter
Set axis fifo to wait for tlast before allowing a read on master port output.

COUNT_DELAY
parameter
Delay count by one clock cycle of the data count clock.

COUNT_ENA
parameter
Enable count, set this to 0 to disable (only disable if read/write/data_count are on the same clock domain!).

Ports

m_axis_aclk Clock for AXIS

m_axis_arstn Negative reset for AXIS

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.

m_axis_tdata	Output data
m_axis_tkeep	Output valid byte indicator
m_axis_tlast	Indicates last word in stream.
m_axis_tuser	Output user bus
m_axis_tdest	Output destination
s_axis_aclk	Clock for AXIS
s_axis_arstn	Negative reset for AXIS
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.
s_axis_tdata	Input data
s_axis_tkeep	Input valid byte indicator
s_axis_tlast	Is this the last word in the stream (active high).
s_axis_tuser	Input user bus
s_axis_tdest	Input desitination
data_count_aclk	Clock for data count
data_count_arstn	Negative edge reset for data count.
data_count	Output that indicates the amount of data in the FIFO.

INSTANTIATED MODULES

dut

```
 axis_fifo #(
                                                                        .
 FIFO_DEPTH(FIFO_DEPTH),
                                                                        .
 COUNT_WIDTH(COUNT_WIDTH),
                                                                        .
 BUS_WIDTH(BUS_WIDTH),
                                                                        .
 USER_WIDTH(USER_WIDTH),
                                                                        .
 DEST_WIDTH(DEST_WIDTH),
                                                                        .
 RAM_TYPE(RAM_TYPE),
                                                                        .
 PACKET_MODE(PACKET_MODE),
                                                                        .
 COUNT_DELAY(COUNT_DELAY),
                                                                        .
 COUNT_ENA(COUNT_ENA)
) dut ( .s_axis_aclk(s_axis_aclk), .s_axis_arstn(s_axis_arstn), .s_axis_tval
```

Device under test, axis_fifo

tb_cocotb.py

AUTHORS

JAY CONVERTINO

DATES

2024/12/09

INFORMATION

Brief

Cocotb test bench

License MIT

FUNCTIONS

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.

single_word

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

Coroutine that is identified as a test routine. This routine tests for writing a single word, and then reading a single word.

Parameters

dut Device under test passed from cocotb.

full_empty

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

Coroutine that is identified as a test routine. This routine tests for writing till the fifo is full, Then reading from the full FIFO.

Parameters

dut Device under test passed from cocotb.

random_ready

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

Coroutine that is identified as a test routine. This routine tests for randomized ready from the sink.

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.