

axi_lite_write_channel_decoder



December 16, 2025

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	3
3 Building	3
3.1 fusesoc	3
3.2 Source Files	4
3.2.1 fusesoc_info File List	4
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 axi_lite_write_channel_decoder	7
5.2 tb_cocotb-py	12
5.3 tb_cocotb-v	15

1 Usage

1.1 Introduction

Main function of this core is allow axi lite write channel to pass data if there is an address match. Otherwise this will block all data downstream which would initiate a transfer. Once a valid address is presented it will connect in one clock cycle.

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
 - AFRL:simple:holdbuffer:1.0.0
 - AFRL:bus:bus_addr_decoder:1.0.0

1.3 In a Project

Simply connect inline with the write address channel of the axi lite bus. Set the address to a valid value and the region to a power of two (makes the most sense anyways). and the decoder will only allow valid traffic to be passed.

```
axi_lite_write_channel_decoder #(
    .ADDRESS_WIDTH(ADDRESS_WIDTH) ,
    .BUS_WIDTH(BUS_WIDTH) ,
    .SLAVE_ADDRESS(SLAVE_ADDRESS) ,
    .SLAVE_REGION(SLAVE_REGION)
) dut (
    .connected(connected) ,
    .aclk(aclk) ,
    .arstn(arstn) ,
    .s_axi_awaddr(s_axi_awaddr) ,
    .s_axi_awprot(s_axi_awprot) ,
    .s_axi_awvalid(s_axi_awvalid) ,
```

```

.s_axi_awready(s_axi_awready),
.s_axi_wdata(s_axi_wdata),
.s_axi_wstrb(s_axi_wstrb),
.s_axi_wvalid(s_axi_wvalid),
.s_axi_wready(s_axi_wready),
.s_axi_bresp(s_axi_bresp),
.s_axi_bvalid(s_axi_bvalid),
.s_axi_bready(s_axi_bready),
.m_axi_awaddr(w_m_axi_awaddr),
.m_axi_awprot(m_axi_awprot),
.m_axi_awvalid(m_axi_awvalid),
.m_axi_awready(m_axi_awready),
.m_axi_wdata(m_axi_wdata),
.m_axi_wstrb(m_axi_wstrb),
.m_axi_wvalid(m_axi_wvalid),
.m_axi_wready(m_axi_wready),
.m_axi_bresp(m_axi_bresp),
.m_axi_bvalid(m_axi_bvalid),
.m_axi_bready(m_axi_bready)
);

```

2 Architecture

The only module is the `axi_lite_write_channel_decoder` module. It is listed below.

- **`axi_lite_write_channel_decoder`** Only allow traffic for valid addresses. (see core for documentation).

Please see 5 for more information.

3 Building

The `axi_lite_write_channel_decoder` 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 met 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/axi_lite_write_channel_decoder.v
- 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_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.
 - **specs** Contains specifications for the bus.
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

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

```
$ pip install cocotb
```

Each module has a cocotb based simulation. These use the cocotb extensions APB and uP. To install these locally use the following.cocotb

```
$ pip install --break-system-packages -e .
```

Then you must use the cocotb sim target. The targets above can be run with various parameters.

```
$ fusesoc run --target sim_cocotb AFRL:bus:  
  ↗ axi_lite_write_channel_decoder:1.0.0
```

5 Module Documentation

- **axi_lite_write_channel_decoder** axi lite write channel decoder
- **tb_cocotb-py** Cocotb python test routines
- **tb_cocotb-v** Cocotb verilog test bench

The next sections document the module in detail.

axi_lite_write_channel_decoder.v

AUTHORS

JAY CONVERTINO

DATES

2025/12/16

INFORMATION

Brief

AXI lite write channel decoder. Block address and data paths till a valid address is presented.

License MIT

Copyright 2025 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.

axi_lite_write_channel_decoder

```
module axi_lite_write_channel_decoder #(
    parameter
        integer
        ADDRESS_WIDTH
        =
        32,
    parameter
        integer
        BUS_WIDTH
        =
        4,
    parameter
        integer
        DATA_BUFFER
```

```

    =
1,
parameter
integer
TIMEOUT_BEATS
=
32,
parameter
[ADDRESS_WIDTH-1:0]
SLAVE_ADDRESS
=
32'h44A20000,
parameter
[ADDRESS_WIDTH-1:0]
SLAVE_REGION
=
32'h0000FFFF
)
(
output
wire
connected,
input
wire
aclk,
input
wire
arstn,
input
wire
[ADDRESS_WIDTH-1:0]
s_axi_awaddr,
input
wire
[2:0]
s_axi_awprot,
input
wire
s_axi_awvalid,
output
wire
s_axi_awready,
input
wire
[BUS_WIDTH*8-1:0]
s_axi_wdata,
input
wire
[BUS_WIDTH-1:0]
s_axi_wstrb,
input
wire
s_axi_wvalid,
output
wire
s_axi_wready,
output
wire
[1:0]
s_axi_bresp,
output
wire
s_axi_bvalid,
input
wire
s_axi_bready,

```

```

    output
    wire
      [ADDRESS_WIDTH-1:0]
    m_axi_awaddr,
    output
    wire
      [2:0]
    m_axi_awprot,
    output
    wire
    m_axi_awvalid,
    input
    wire
    m_axi_awready,
    output
    wire
      [BUS_WIDTH*8-1:0]
    m_axi_wdata,
    output
    wire
      [BUS_WIDTH-1:0]
    m_axi_wstrb,
    output
    wire
    m_axi_wvalid,
    input
    wire
    m_axi_wready,
    input
    wire
      [1:0]
    m_axi_bresp,
    input
    wire
    m_axi_bvalid,
    output
    wire
    m_axi_bready
)

```

AXI lite write channel decoder. Block address and data paths till a valid address is presented.

Parameters

ADDRESS_WIDTH	Width of the AXI LITE address port in bits.
parameter integer	
BUS_WIDTH	Width of the AXI LITE bus data port in bytes.
parameter integer	
DATA_BUFFER	Buffer data channel, 0 to disable.
parameter integer	
TIMEOUT_BEATS	Number of clock cycles (beats) to count till timeout. 0 disables timeout.
parameter integer	
SLAVE_ADDRESS	Array of Addresses for each slave (0 = slave 0 and so on).
parameter [ADDRESS_WIDTH- 1:0]	
SLAVE_REGION	Region for the address that is valid for the SLAVE ADDRESS.
parameter [ADDRESS_WIDTH- 1:0]	

Ports

connected	Core has established channel connection
output wire	
aclk	Input clock
input wire	

arstn	Input negative reset
s_axi_awaddr	Slave write input channel address
s_axi_awprot	Slave write input channel protection mode
s_axi_awvalid	Slave write input channel address is valid.
s_axi_awready	Slave write input channel is ready.
s_axi_wdata	Slave write input channel data
s_axi_wstrb	Slave write input channel valid bytes
s_axi_wvalid	Slave write input channel data valid
s_axi_wready	Slave write input channel is ready.
s_axi_bresp	Slave write input channel response to write(s).
s_axi_bvalid	Slave write input channel response valid.
s_axi_bready	Slave write input channel response ready.
m_axi_awaddr	Master write output channel address.
m_axi_awprot	Master write output channel protection mode.
m_axi_awvalid	Master write output channel address is valid.
m_axi_awready	Master write output channel is ready.
m_axi_wdata	Master write output channel data.
m_axi_wstrb	Master write output channel data bytes valid.
m_axi_wvalid	Master write output channel data is valid.
m_axi_wready	Master write output channel data ready.
m_axi_bresp	Master write output channel response.
m_axi_bvalid	Master write output channel response valid.
m_axi_bready	Master write output channel response ready.

INSTANTIATED MODULES

inst_addr_buffer

Buffer for the address

bus_addr_decoder

Decoder for address bus.

inst_data_resp_buffer

If data buffer enabled, this holdbuffer will be generated.

inst_data_buffer

If data buffer enabled, this holdbuffer will be generated.

tb_cocotb_axi_lite.py

AUTHORS

JAY CONVERTINO

DATES

2025/03/04

INFORMATION

Brief

Cocotb test bench

License MIT

Copyright 2025 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.

FUNCTIONS

random_bool

```
def random_bool()
```

Return a infinite 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.

increment_test_write

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

Coroutine that is identified as a test routine. Write data to the device at the proper address and region.

Parameters

dut Device under test passed from cocotb.

increment_test_random_ready_write_data(dut):

Coroutine that is identified as a test routine.

Parameters

dut Device under test passed from cocotb.

increment_test_random_ready_read_addr

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

Coroutine that is identified as a test routine. Setup to read from gpio

Parameters

dut Device under test passed from cocotb.

increment_test_timeout_no_answer

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

Coroutine that is identified as a test routine. Setup to read from gpio

Parameters

dut Device under test passed from cocotb.

increment_test_random_ready_timeout_no_answer

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

Coroutine that is identified as a test routine. Setup to read from gpio

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.

tb_cocotb.v

AUTHORS

JAY CONVERTINO

DATES

2025/03/26

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

Copyright 2025 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.BUS_WIDTH

tb_cocotb

```
module tb_cocotb #(
    parameter
        integer
        ADDRESS_WIDTH
        =
        32,
    parameter
        integer
        BUS_WIDTH
        =
        4,
    parameter
        [ADDRESS_WIDTH-1:0]
        SLAVE_ADDRESS
```

```

=
32'h44A20000,
parameter
  [ADDRESS_WIDTH-1:0]
SLAVE_REGION
=
32'h0000FFFF
)
(
output
wire
connected,
input
wire
aclk,
input
wire
arstn,
input
wire
[ADDRESS_WIDTH-1:0]
s_axi_awaddr,
input
wire
[2:0]
s_axi_awprot,
input
wire
s_axi_awvalid,
output
wire
s_axi_awready,
input
wire
[BUS_WIDTH*8-1:0]
s_axi_wdata,
input
wire
[BUS_WIDTH-1:0]
s_axi_wstrb,
input
wire
s_axi_wvalid,
output
wire
s_axi_wready,
output
wire
[1:0]
s_axi_bresp,
output
wire
s_axi_bvalid,
input
wire
s_axi_bready,
output
wire
[ADDRESS_WIDTH-1:0]
m_axi_awaddr,
output
wire
[2:0]
m_axi_awprot,
output
wire

```

```

m_axi_awvalid,
input
wire
m_axi_awready,
output
wire
[BUS_WIDTH*8-1:0]
m_axi_wdata,
output
wire
[BUS_WIDTH-1:0]
m_axi_wstrb,
output
wire
m_axi_wvalid,
input
wire
m_axi_wready,
input
wire
[1:0]
m_axi_bresp,
input
wire
m_axi_bvalid,
output
wire
m_axi_bready,
input
wire
[ADDRESS_WIDTH-1:0]
s_axi_araddr,
input
wire
[2:0]
s_axi_arprot,
input
wire
s_axi_arvalid,
output
wire
s_axi_arready,
output
wire
[BUS_WIDTH*8-1:0]
s_axi_rdata,
output
wire
[1:0]
s_axi_rresp,
output
wire
s_axi_rvalid,
input
wire
s_axi_rready,
output
wire
[ADDRESS_WIDTH-1:0]
m_axi_araddr,
output
wire
[2:0]
m_axi_arprot,
output
wire

```

```

    m_axi_arvalid,
    input
    wire
    m_axi_arready,
    input
    wire
        [BUS_WIDTH*8-1:0]
    m_axi_rdata,
    input
    wire
        [1:0]
    m_axi_rresp,
    input
    wire
    m_axi_rvalid,
    output
    wire
    m_axi_rready
)

```

Parameters

ADDRESS_WIDTH	Width of the AXI LITE address port in bits.
parameter integer	
BUS_WIDTH	Width of the AXI LITE bus data port in bytes.
parameter integer	
DATA_BUFFER	Buffer data channel, 0 to disable.
TIMEOUT_BEATS	Number of clock cycles (beats) to count till timeout. 0 disables timeout.
SLAVE_ADDRESS	Array of Addresses for each slave (0 = slave 0 and so on).
parameter [ADDRESS_WIDTH- 1:0]	
SLAVE_REGION	Region for the address that is valid for the SLAVE ADDRESS.
parameter [ADDRESS_WIDTH- 1:0]	

Ports

connected	Core has established channel connection
output wire	
aclk	Input clock
input wire	
arstn	Input negative reset
input wire	
s_axi_awaddr	Slave write input channel address
input wire [ADDRESS_WIDTH- 1:0]	
s_axi_awprot	Slave write input channel protection mode
input wire [2:0]	
s_axi_awvalid	Slave write input channel address is valid.
input wire	
s_axi_awready	Slave write input channel is ready.
output wire	
s_axi_wdata	Slave write input channel data
input wire [BUS_WIDTH* 8- 1:0]	
s_axi_wstrb	Slave write input channel valid bytes
input wire [BUS_WIDTH- 1:0]	
s_axi_wvalid	Slave write input channel data valid
input wire	
s_axi_wready	Slave write input channel is ready.
output wire	

s_axi_bresp	Slave write input channel response to write(s).
s_axi_bvalid	Slave write input channel response valid.
s_axi_bready	Slave write input channel response ready.
m_axi_awaddr	Master write output channel address.
m_axi_awprot	Master write output channel protection mode.
m_axi_awvalid	Master write output channel address is valid.
m_axi_awready	Master write output channel is ready.
m_axi_wdata	Master write output channel data.
m_axi_wstrb	Master write output channel data bytes valid.
m_axi_wvalid	Master write output channel data is valid.
m_axi_wready	Master write output channel data ready.
m_axi_bresp	Master write output channel response.
m_axi_bvalid	Master write output channel response valid.
m_axi_bready	Master write output channel response ready.

INSTANTIATED MODULES

dut

Device under test, axi_lite_wr_addr