

# BUS\_BLOCK\_RAM



November 14, 2024

Jay Convertino

# Contents

<b>1 Usage</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Dependencies . . . . .	2
1.2.1 axi_lite_block_ram Depenecies . . . . .	2
1.2.2 wishbone_classic_block_ram Depenecies . . . . .	2
1.3 In a Project . . . . .	2
<b>2 Architecture</b>	<b>3</b>
<b>3 Building</b>	<b>3</b>
3.1 fusesoc . . . . .	4
3.2 Source Files . . . . .	4
3.2.1 axi_lite_block_ram File List . . . . .	4
3.2.2 wishbone_classic_block_ram File List . . . . .	4
3.3 Targets . . . . .	4
3.3.1 axi_lite_block_ram Targets . . . . .	4
3.3.2 wishbone_classic_block_ram Targets . . . . .	6
3.4 Directory Guide . . . . .	6
<b>4 Simulation</b>	<b>7</b>
4.1 iverilog . . . . .	7
4.2 cocotb . . . . .	7
<b>5 Module Documentation</b>	<b>8</b>
5.1 axi_lite_block_ram . . . . .	9
5.2 wishbone_classic_block_ram . . . . .	14

# 1 Usage

## 1.1 Introduction

Selectable BUS block RAM for any FPGA target. Currently supports wishbone classic or AXI lite. This will create FPGA block RAM that is accessible via the selected bus.

## 1.2 Dependencies

The following are the dependencies of the cores.

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

### 1.2.1 axi\_lite\_block\_ram Depenecies

- dep
  - AFRL:utility:helper:1.0.0
  - AFRL:ram:dc\_block\_ram:1.0.0
- dep\_tb
  - AFRL:simulation:clock\_stimulator
  - AFRL:utility:sim\_helper

### 1.2.2 wishbone\_classic\_block\_ram Depenecies

- dep
  - AFRL:utility:helper:1.0.0
  - AFRL:ram:dc\_block\_ram:1.0.0
  - AFRL:bus:up\_wishbone\_classic:1.0.0
- dep\_tb
  - AFRL:simulation:clock\_stimulator
  - AFRL:utility:sim\_helper

## 1.3 In a Project

Connect the device using the bus selected, see 5 for details

## 2 Architecture

There are two bus block RAM cores. The AXI lite block RAM and the Wishbone Classic block RAM.

AXI lite block RAM is made up of the following modules.

- **up\_axi** Convert AXI lite to the Analog Devices uP BUS. (see core for documentation).
- **dc\_block\_ram** Provides a dual clock block RAM. (see core for documentation).

This core has 1 always blocks that are sensitive to the positive clock edge.

- **register request to the acknowledge** Takes the request and registers to the acknowledge. All reads and writes will produce something.

Please see 5 for more information.

Wishbone Classic block RAM is made up of the following modules.

- **up\_wishbone\_classic** Convert Wishbone Classic to the Analog Devices uP BUS. (see core for documentation).
- **dc\_block\_ram** Provides a dual clock block RAM. (see core for documentation).

This core has 1 always blocks that are sensitive to the positive clock edge.

- **register request to the acknowledge** Takes the request and registers to the acknowledge. All reads and writes will produce something.

Please see 5 for more information.

## 3 Building

The BUS block RAM cores 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 axi\_lite\_block\_ram File List

- src
  - Type: verilogSource
  - src/axi\_lite\_block\_ram.v
- tb
  - 'tb/tb\_fifo.v': 'file\_type': 'verilogSource'
- constr
  - 'tool\_vivado ? (constr/fifo\_constr.tcl)': 'file\_type': 'SDC'

#### 3.2.2 wishbone\_classic\_block\_ram File List

- src
  - Type: verilogSource
  - src/wishbone\_classic\_block\_ram.v
- tb
  - 'tb/tb\_wishbone\_slave.v': 'file\_type': 'verilogSource'

### 3.3 Targets

#### 3.3.1 axi\_lite\_block\_ram Targets

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

- constr
- sim
  - Info: Constant data value with file check.
  - src
  - dep
  - constr
  - tb
  - dep\_tb
  - IN\_FILE\_NAME
  - OUT\_FILE\_NAME
  - RAND\_FULL
  - FIFO\_DEPTH
- sim\_rand\_data
  - Info: Feed random data input with file check
  - src
  - dep
  - constr
  - tb
  - dep\_tb
  - IN\_FILE\_NAME=random.bin
  - OUT\_FILE\_NAME=out\_random.bin
  - RAND\_FULL
  - FIFO\_DEPTH
- sim\_rand\_ready\_rand\_data
  - Info: Feed random data input, and randomize the read ready on the output. Perform output file check.
  - src
  - dep
  - constr
  - tb
  - dep\_tb
  - IN\_FILE\_NAME=random.bin
  - OUT\_FILE\_NAME=out\_random.bin
  - RAND\_FULL=1

- FIFO\_DEPTH
- sim\_8bit\_count\_data
  - Info: Feed a counter data as input, perform file check.
  - src
  - dep
  - constr
  - tb
  - dep\_tb
  - IN\_FILE\_NAME=8bit\_count.bin
  - OUT\_FILE\_NAME=out\_8bit\_count.bin
  - RAND\_FULL
  - FIFO\_DEPTH

### 3.3.2 wishbone\_classic\_block\_ram Targets

- default
  - Info: Default for IP intergration.
  - src
  - dep
- sim
  - Info: Default for IP intergration.
  - src
  - dep
  - tb
  - dep\_tb

## 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 are two different BUS block RAM modules that can be used in a project.

- **axi\_lite\_block\_ram** AXI lite block RAM
- **wishbone\_classic\_block\_ram** Wishbone Classic block RAM

The next sections document the module in great detail.

# axi\_lite\_block\_ram.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/03/07

---

## INFORMATION

---

### Brief

---

axi lite block ram

### License MIT

---

Copyright 2024 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\_block\_ram

---

```
module axi_lite_block_ram #(
  parameter
    ADDRESS_WIDTH
    =
    32,
  parameter
    BUS_WIDTH
    =
    4,
  parameter
    DEPTH
```

```

    =
    512,
    parameter
    RAM_TYPE
    =
    "block",
    parameter
    HEX_FILE
    =
    ""
) ( input aclk, input arstn, input s_axi_awvalid, input [ADDRESS_WIDTH-1:0]

```

axi lite block ram

## Parameters

<b>ADDRESS_WIDTH</b> parameter	Width of the axi address bus in bits.
<b>BUS_WIDTH</b> parameter	Bus width for data paths in bytes.
<b>DEPTH</b> parameter	Depth of the RAM in terms of data width words.
<b>RAM_TYPE</b> parameter	Used to set the ram_style attribute.
<b>HEX_FILE</b> parameter	Hex file to write to RAM.

## Ports

<b>aclk</b>	Clock for all devices in the core
<b>arstn</b>	Negative reset
<b>s_axi_awvalid</b>	Axi Lite aw valid
<b>s_axi_awaddr</b>	Axi Lite aw addr
<b>s_axi_awprot</b>	Axi Lite aw prot
<b>s_axi_awready</b>	Axi Lite aw ready
<b>s_axi_wvalid</b>	Axi Lite w valid
<b>s_axi_wdata</b>	Axi Lite w data
<b>s_axi_wstrb</b>	Axi Lite w strb
<b>s_axi_wready</b>	Axi Lite w ready
<b>s_axi_bvalid</b>	Axi Lite b valid
<b>s_axi_bresp</b>	Axi Lite b resp
<b>s_axi_bready</b>	Axi Lite b ready
<b>s_axi_arvalid</b>	Axi Lite ar valid
<b>s_axi_araddr</b>	Axi Lite ar addr
<b>s_axi_arprot</b>	Axi Lite ar prot
<b>s_axi_arready</b>	Axi Lite ar ready
<b>s_axi_rvalid</b>	Axi Lite r valid
<b>s_axi_rdata</b>	Axi Lite r data
<b>s_axi_rresp</b>	Axi Lite r resp
<b>s_axi_rready</b>	Axi Lite r ready

## up\_rreq

---

```
wire up_rreq
```

uP read bus request

## up\_rack

---

```
reg up_rack
```

uP read bus acknowledge

## up\_raddr

---

```
wire [ADDRESS_WIDTH-3:0] up_raddr
```

uP read bus address

## up\_rdata

---

```
wire [(  
  BUS_WIDTH*4  
)-1:0] up_rdata
```

uP read bus request

## up\_wreq

---

```
wire up_wreq
```

uP write bus request

## up\_wack

---

```
reg up_wack
```

uP write bus acknowledge

## up\_waddr

---

```
wire [ADDRESS_WIDTH-3:0] up_waddr
```

uP write bus address

## up\_wdata

---

```
wire [(
  BUS_WIDTH*4
)-1:0] up_wdata
```

uP write bus data

## INSTANTIATED MODULES

---

### inst\_up\_axi

---

```
up_axi inst_up_axi (
  up_rstn                                     (
  arstn),
  up_clk                                     (
  aclk),
  up_axi_awvalid(s_axi_awvalid),
  up_axi_awaddr(s_axi_awaddr),
  up_axi_awready(s_axi_awready),
  up_axi_wvalid(s_axi_wvalid),
  up_axi_wdata(s_axi_wdata),
  up_axi_wstrb(s_axi_wstrb),
  up_axi_wready(s_axi_wready),
  up_axi_bvalid(s_axi_bvalid),
  up_axi_bresp(s_axi_bresp),
  up_axi_bready(s_axi_bready),
  up_axi_arvalid(s_axi_arvalid),
  up_axi_araddr(s_axi_araddr),
  up_axi_arready(s_axi_arready),
  up_axi_rvalid(s_axi_rvalid),
  up_axi_rresp(s_axi_rresp),
  up_axi_rdata(s_axi_rdata),
  up_axi_rready(s_axi_rready),
  up_wreq(up_wreq),
  up_waddr(up_waddr),
  up_wdata(up_wdata),
  up_wack(up_wack),
```

```

up_rreq(up_rreq),
up_raddr(up_raddr),
up_rdata(up_rdata),
up_rack(up_rack)
)

```

Module instance of up\_axi for the AXI Lite bus to the uP bus.

## inst\_dc\_block\_ram

---

```

dc_block_ram #(
    RAM_DEPTH(DEPTH),
    BYTE_WIDTH(BUS_WIDTH),
    ADDR_WIDTH(ADDRESS_WIDTH),
    HEX_FILE(HEX_FILE),
    RAM_TYPE(RAM_TYPE)
) inst_dc_block_ram ( .rd_clk(ac1k), .rd_rstn(arstn), .rd_en(up_rreq), .rd_c

```

Module instance of dc\_block\_ram that connects to the uP BUS directly.

# wishbone\_classic\_block\_ram.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/03/07

---

## INFORMATION

---

### Brief

---

Wishbone classic block RAM core.

### License MIT

---

Copyright 2024 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.

## wishbone\_classic\_block\_ram

---

```
module wishbone_classic_block_ram #(
    parameter
    ADDRESS_WIDTH
    =
    32,
    parameter
    BUS_WIDTH
    =
    4,
    parameter
    DEPTH
```

```

    =
    512,
    parameter
    RAM_TYPE
    =
    "block",
    parameter
    HEX_FILE
    =
    ""
) ( input clk, input rst, input s_wb_cyc, input s_wb_stb, input s_wb_we, in

```

Wishbone classic block RAM core.

## Parameters

<b>ADDRESS_WIDTH</b> <small>parameter</small>	Width of the axi address bus in bits.
<b>BUS_WIDTH</b> <small>parameter</small>	Bus width for data paths in bytes.
<b>DEPTH</b> <small>parameter</small>	Depth of the RAM in terms of data width words.
<b>RAM_TYPE</b> <small>parameter</small>	Used to set the ram_style attribute.
<b>HEX_FILE</b> <small>parameter</small>	Hex file to write to RAM.

## Ports

<b>clk</b>	Clock for all devices in the core
<b>rst</b>	Positive reset
<b>s_wb_cyc</b>	Bus Cycle in process
<b>s_wb_stb</b>	Valid data transfer cycle
<b>s_wb_we</b>	Active High write, low read
<b>s_wb_addr</b>	Bus address
<b>s_wb_data_i</b>	Input data
<b>s_wb_sel</b>	Device Select
<b>s_wb_bte</b>	Burst Type Extension
<b>s_wb_cti</b>	Cycle Type
<b>s_wb_ack</b>	Bus transaction terminated
<b>s_wb_data_o</b>	Output data
<b>s_wb_err</b>	Active high when a bus error is present

## c\_PWR\_RAM

```

localparam c_PWR_RAM = c_logb2(
    DEPTH
)

```

power of 2 conversion of DEPTH



## c\_RAM\_DEPTH

---

```
localparam c_RAM_DEPTH = 2 ** c_PWR_RAM
```

create RAM depth based on power of two depth size.

## up\_rreq

---

```
wire up_rreq
```

uP read bus request

## up\_rack

---

```
reg up_rack
```

uP read bus acknowledge

## up\_raddr

---

```
wire [ADDRESS_WIDTH-3:0] up_raddr
```

uP read bus address

## up\_rdata

---

```
wire [(  
    BUS_WIDTH*4  
)-1:0] up_rdata
```

uP read bus request

## up\_wreq

---

```
wire up_wreq
```

uP write bus request

## up\_wack

---

```
reg up_wack
```

uP write bus acknowledge

## up\_waddr

---

```
wire [ADDRESS_WIDTH-3:0] up_waddr
```

uP write bus address

**up\_wdata**

```
wire [(
  BUS_WIDTH*4
)-1:0] up_wdata
```

uP write bus data

## INSTANTIATED MODULES

## inst\_up\_wishbone\_classic

```
up_wishbone_classic #(
    ADDRESS_WIDTH(ADDRESS_WIDTH),
    BUS_WIDTH(BUS_WIDTH)
) inst_up_wishbone_classic ( .clk(clk), .rst(rst), .s_wb_cyc(s_wb_cyc), .s_wb_data(s_wb_data), .s_wb_err(s_wb_err), .s_wb_ack(s_wb_ack), .s_wb_stb(s_wb_stb), .s_wb_we(s_wb_we), .s_wb_be(s_wb_be), .s_wb_oe(s_wb_oe), .s_wb_oe2(s_wb_oe2), .s_wb_oe3(s_wb_oe3), .s_wb_oe4(s_wb_oe4), .s_wb_oe5(s_wb_oe5), .s_wb_oe6(s_wb_oe6), .s_wb_oe7(s_wb_oe7), .s_wb_oe8(s_wb_oe8), .s_wb_oe9(s_wb_oe9), .s_wb_oe10(s_wb_oe10), .s_wb_oe11(s_wb_oe11), .s_wb_oe12(s_wb_oe12), .s_wb_oe13(s_wb_oe13), .s_wb_oe14(s_wb_oe14), .s_wb_oe15(s_wb_oe15), .s_wb_oe16(s_wb_oe16), .s_wb_oe17(s_wb_oe17), .s_wb_oe18(s_wb_oe18), .s_wb_oe19(s_wb_oe19), .s_wb_oe20(s_wb_oe20), .s_wb_oe21(s_wb_oe21), .s_wb_oe22(s_wb_oe22), .s_wb_oe23(s_wb_oe23), .s_wb_oe24(s_wb_oe24), .s_wb_oe25(s_wb_oe25), .s_wb_oe26(s_wb_oe26), .s_wb_oe27(s_wb_oe27), .s_wb_oe28(s_wb_oe28), .s_wb_oe29(s_wb_oe29), .s_wb_oe30(s_wb_oe30), .s_wb_oe31(s_wb_oe31), .s_wb_oe32(s_wb_oe32), .s_wb_oe33(s_wb_oe33), .s_wb_oe34(s_wb_oe34), .s_wb_oe35(s_wb_oe35), .s_wb_oe36(s_wb_oe36), .s_wb_oe37(s_wb_oe37), .s_wb_oe38(s_wb_oe38), .s_wb_oe39(s_wb_oe39), .s_wb_oe40(s_wb_oe40), .s_wb_oe41(s_wb_oe41), .s_wb_oe42(s_wb_oe42), .s_wb_oe43(s_wb_oe43), .s_wb_oe44(s_wb_oe44), .s_wb_oe45(s_wb_oe45), .s_wb_oe46(s_wb_oe46), .s_wb_oe47(s_wb_oe47), .s_wb_oe48(s_wb_oe48), .s_wb_oe49(s_wb_oe49), .s_wb_oe50(s_wb_oe50), .s_wb_oe51(s_wb_oe51), .s_wb_oe52(s_wb_oe52), .s_wb_oe53(s_wb_oe53), .s_wb_oe54(s_wb_oe54), .s_wb_oe55(s_wb_oe55), .s_wb_oe56(s_wb_oe56), .s_wb_oe57(s_wb_oe57), .s_wb_oe58(s_wb_oe58), .s_wb_oe59(s_wb_oe59), .s_wb_oe60(s_wb_oe60), .s_wb_oe61(s_wb_oe61), .s_wb_oe62(s_wb_oe62), .s_wb_oe63(s_wb_oe63), .s_wb_oe64(s_wb_oe64), .s_wb_oe65(s_wb_oe65), .s_wb_oe66(s_wb_oe66), .s_wb_oe67(s_wb_oe67), .s_wb_oe68(s_wb_oe68), .s_wb_oe69(s_wb_oe69), .s_wb_oe70(s_wb_oe70), .s_wb_oe71(s_wb_oe71), .s_wb_oe72(s_wb_oe72), .s_wb_oe73(s_wb_oe73), .s_wb_oe74(s_wb_oe74), .s_wb_oe75(s_wb_oe75), .s_wb_oe76(s_wb_oe76), .s_wb_oe77(s_wb_oe77), .s_wb_oe78(s_wb_oe78), .s_wb_oe79(s_wb_oe79), .s_wb_oe80(s_wb_oe80), .s_wb_oe81(s_wb_oe81), .s_wb_oe82(s_wb_oe82), .s_wb_oe83(s_wb_oe83), .s_wb_oe84(s_wb_oe84), .s_wb_oe85(s_wb_oe85), .s_wb_oe86(s_wb_oe86), .s_wb_oe87(s_wb_oe87), .s_wb_oe88(s_wb_oe88), .s_wb_oe89(s_wb_oe89), .s_wb_oe90(s_wb_oe90), .s_wb_oe91(s_wb_oe91), .s_wb_oe92(s_wb_oe92), .s_wb_oe93(s_wb_oe93), .s_wb_oe94(s_wb_oe94), .s_wb_oe95(s_wb_oe95), .s_wb_oe96(s_wb_oe96), .s_wb_oe97(s_wb_oe97), .s_wb_oe98(s_wb_oe98), .s_wb_oe99(s_wb_oe99), .s_wb_oe100(s_wb_oe100), .s_wb_oe101(s_wb_oe101), .s_wb_oe102(s_wb_oe102), .s_wb_oe103(s_wb_oe103), .s_wb_oe104(s_wb_oe104), .s_wb_oe105(s_wb_oe105), .s_wb_oe106(s_wb_oe106), .s_wb_oe107(s_wb_oe107), .s_wb_oe108(s_wb_oe108), .s_wb_oe109(s_wb_oe109), .s_wb_oe110(s_wb_oe110), .s_wb_oe111(s_wb_oe111), .s_wb_oe112(s_wb_oe112), .s_wb_oe113(s_wb_oe113), .s_wb_oe114(s_wb_oe114), .s_wb_oe115(s_wb_oe115), .s_wb_oe116(s_wb_oe116), .s_wb_oe117(s_wb_oe117), .s_wb_oe118(s_wb_oe118), .s_wb_oe119(s_wb_oe119), .s_wb_oe120(s_wb_oe120), .s_wb_oe121(s_wb_oe121), .s_wb_oe122(s_wb_oe122), .s_wb_oe123(s_wb_oe123), .s_wb_oe124(s_wb_oe124), .s_wb_oe125(s_wb_oe125), .s_wb_oe126(s_wb_oe126), .s_wb_oe127(s_wb_oe127), .s_wb_oe128(s_wb_oe128), .s_wb_oe129(s_wb_oe129), .s_wb_oe130(s_wb_oe130), .s_wb_oe131(s_wb_oe131), .s_wb_oe132(s_wb_oe132), .s_wb_oe133(s_wb_oe133), .s_wb_oe134(s_wb_oe134), .s_wb_oe135(s_wb_oe135), .s_wb_oe136(s_wb_oe136), .s_wb_oe137(s_wb_oe137), .s_wb_oe138(s_wb_oe138), .s_wb_oe139(s_wb_oe139), .s_wb_oe140(s_wb_oe140), .s_wb_oe141(s_wb_oe141), .s_wb_oe142(s_wb_oe142), .s_wb_oe143(s_wb_oe143), .s_wb_oe144(s_wb_oe144), .s_wb_oe145(s_wb_oe145), .s_wb_oe146(s_wb_oe146), .s_wb_oe147(s_wb_oe147), .s_wb_oe148(s_wb_oe148), .s_wb_oe149(s_wb_oe149), .s_wb_oe150(s_wb_oe150), .s_wb_oe151(s_wb_oe151), .s_wb_oe152(s_wb_oe152), .s_wb_oe153(s_wb_oe153), .s_wb_oe154(s_wb_oe154), .s_wb_oe155(s_wb_oe155), .s_wb_oe156(s_wb_oe156), .s_wb_oe157(s_wb_oe157), .s_wb_oe158(s_wb_oe158), .s_wb_oe159(s_wb_oe159), .s_wb_oe160(s_wb_oe160), .s_wb_oe161(s_wb_oe161), .s_wb_oe162(s_wb_oe162), .s_wb_oe163(s_wb_oe163), .s_wb_oe164(s_wb_oe164), .s_wb_oe165(s_wb_oe165), .s_wb_oe166(s_wb_oe166), .s_wb_oe167(s_wb_oe167), .s_wb_oe168(s_wb_oe168), .s_wb_oe169(s_wb_oe169), .s_wb_oe170(s_wb_oe170), .s_wb_oe171(s_wb_oe171), .s_wb_oe172(s_wb_oe172), .s_wb_oe173(s_wb_oe173), .s_wb_oe174(s_wb_oe174), .s_wb_oe175(s_wb_oe175), .s_wb_oe176(s_wb_oe176), .s_wb_oe177(s_wb_oe177), .s_wb_oe178(s_wb_oe178), .s_wb_oe179(s_wb_oe179), .s_wb_oe180(s_wb_oe180), .s_wb_oe181(s_wb_oe181), .s_wb_oe182(s_wb_oe182), .s_wb_oe183(s_wb_oe183), .s_wb_oe184(s_wb_oe184), .s_wb_oe185(s_wb_oe185), .s_wb_oe186(s_wb_oe186), .s_wb_oe187(s_wb_oe187), .s_wb_oe188(s_wb_oe188), .s_wb_oe189(s_wb_oe189), .s_wb_oe190(s_wb_oe190), .s_wb_oe191(s_wb_oe191), .s_wb_oe192(s_wb_oe192), .s_wb_oe193(s_wb_oe193), .s_wb_oe194(s_wb_oe194), .s_wb_oe195(s_wb_oe195), .s_wb_oe196(s_wb_oe196), .s_wb_oe197(s_wb_oe197), .s_wb_oe198(s_wb_oe198), .s_wb_oe199(s_wb_oe199), .s_wb_oe200(s_wb_oe200), .s_wb_oe201(s_wb_oe201), .s_wb_oe202(s_wb_oe202), .s_wb_oe203(s_wb_oe203), .s_wb_oe204(s_wb_oe204), .s_wb_oe205(s_wb_oe205), .s_wb_oe206(s_wb_oe206), .s_wb_oe207(s_wb_oe207), .s_wb_oe208(s_wb_oe208), .s_wb_oe209(s_wb_oe209), .s_wb_oe210(s_wb_oe210), .s_wb_oe211(s_wb_oe211), .s_wb_oe212(s_wb_oe212), .s_wb_oe213(s_wb_oe213), .s_wb_oe214(s_wb_oe214), .s_wb_oe215(s_wb_oe215), .s_wb_oe216(s_wb_oe216), .s_wb_oe217(s_wb_oe217), .s_wb_oe218(s_wb_oe218), .s_wb_oe219(s_wb_oe219), .s_wb_oe220(s_wb_oe220), .s_wb_oe221(s_wb_oe221), .s_wb_oe222(s_wb_oe222), .s_wb_oe223(s_wb_oe223), .s_wb_oe224(s_wb_oe224), .s_wb_oe225(s_wb_oe225), .s_wb_oe226(s_wb_oe226), .s_wb_oe227(s_wb_oe227), .s_wb_oe228(s_wb_oe228), .s_wb_oe229(s_wb_oe229), .s_wb_oe230(s_wb_oe230), .s_wb_oe231(s_wb_oe231), .s_wb_oe232(s_wb_oe232), .s_wb_oe233(s_wb_oe233), .
```

Module instance of up\_wishbone\_classic for the Wishbone Classic bus to the uP bus.

## inst\_dc\_block\_ram

```
dc_block_ram #(
    RAM_DEPTH(c_RAM_DEPTH),
    BYTE_WIDTH(BUS_WIDTH),
    ADDR_WIDTH(c_PWR_RAM),
    HEX_FILE(HEX_FILE),
    RAM_TYPE(RAM_TYPE)
) inst_dc_block_ram ( .rd_clk(clk), .rd_rstn(~rst), .rd_en(up_rreq), .rd_data
```

Module instance of `dc_block_ram` that connects to the uP BUS directly.