wishbone_classic_block_ram.v

AUTHORS

JAY CONVERTINO

DATES

2024/03/07

INFORMATION

Brief

Wishbone classic block RAM core.

License MIT

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, inp
```

Wishbone classic block RAM core.

Parameters

ADDRESS_WIDTH Width of the axi address bus in bits.
parameter

BUS_WIDTH Bus width for data paths in bytes.
parameter

DEPTH Depth of the RAM in terms of data width words.
parameter

RAM_TYPE Used to set the ram_style atribute.
parameter

HEX_FILE Hex file to write to RAM.
parameter

Ports

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

c_PWR_RAM

```
localparam c_PWR_RAM = clogb2(
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-(
ADDRESS_WIDTH
                                                                    /
16
)-1:0] 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-(
ADDRESS_WIDTH
```

```
                                                                                /
16
)-1:0] up_waddr
```

uP write bus address

up_wdata

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

uP write bus data

INSTANTIANTED 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_w
```

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_dat
```

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