# up_wishbone_pipeline.v

## AUTHORS

## JAY CONVERTINO

## DATES

## 2024/03/01

## INFORMATION

### Brief

Wishbone Pipeline Slave to uP interface

### License MIT



## up_wishbone_pipeline

```
module up_wishbone_pipeline #(
parameter
ADDRESS_WIDTH
 =
16,
parameter
BUS_WIDTH
 =
4
) ( input clk, input rst, input s_wb_cyc, input s_wb_stb, input s_wb_we, inp
```

Wishbone Classic slave to uP up_wishbone_classic

### Parameters

| | |
|---|---|
| **ADDRESS_WIDTH** parameter | Width of the Wishbone address port in bits. |
| **BUS_WIDTH** parameter | Width of the Wishbone bus data port in bytes. |

### Ports

| | |
|---|---|
| **clk** | Clock |
| **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 |
| **up_rreq** | uP bus read request |
| **up_rack** | uP bus read ack |
| **up_raddr** | uP bus read address |
| **up_rdata** | uP bus read data |
| **up_wreq** | uP bus write request |
| **up_wack** | uP bus write ack |
| **up_waddr** | uP bus write address |
| **up_wdata** | uP bus write data |

# VARIABLES

## valid

```
assign valid = s_wb_cyc & s_wb_stb & ~r_rst[0]
```

Indicate valid request from wishbone.

## s_wb_stall

```
assign s_wb_stall = ~up_ack & s_wb_cyc & ~r_rst[0]
```

if we have not ack'd, cyc is active stall the bus

## up_rreq

```
assign up_rreq = (
                                                    r_up_rreq ? r_up_rreq &

s_wb_cyc
:
valid & ~s_wb_we & ~r_rst[0]
)
```

Convert wishbone read requests to up read requests

## up_wreq

```
assign up_wreq = (
                                                    r_up_wreq ? r_up_wreq &

s_wb_cyc
:
valid & s_wb_we & ~r_rst[0]
)
```

Convert wishbone write requests to up write requests

## up_ack

```
assign up_ack = (
up_rack |
up_wack
)
```

ack is ack for both, or them so either may pass

## s_wb_ack

```
assign s_wb_ack = up_ack & ~r_rst[0]
```

combined uP ack is wishbone ack.

## up_raddr

```
assign up_raddr = (
                                                    ~s_wb_we & ~r_rst[0] ?

s_wb_addr
:
0
)
```

assign wishbone address to read port if selected

## up_waddr

```
assign up_waddr = (
                                                    s_wb_we & ~r_rst[0] ?
s_wb_addr
:
0
)
```

assign wishbone address to write port if selected