

# up\_apb3.v

---

## AUTHORS

---

JAY CONVERTINO

---

## DATES

---

2024/03/19

---

## INFORMATION

---

### Brief

---

APB3 slave to uP interface

### 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.

## up\_apb3

---

```
module up_apb3 #(
    parameter
    ADDRESS_WIDTH
    =
    16,
    parameter
    BUS_WIDTH
    =
    4
) ( input clk, input rstn, input [ADDRESS_WIDTH-1:0] s_apb_paddr, input [0:0] s_apb_wdata,
```

APB3 slave to uP interface

## Parameters

<code>ADDRESS_WIDTH</code> <small>parameter</small>	Width of the APB3 address port in bits.
<code>BUS_WIDTH</code> <small>parameter</small>	Width of the APB3 bus data port in bytes.

## Ports

<code>clk</code>	Clock
<code>rstn</code>	negative reset
<code>s_apb_paddr</code>	APB3 address bus, up to 32 bits wide.
<code>s_apb_psel</code>	APB3 select per slave (1 for this core).
<code>s_apb_penable</code>	APB3 enable device for multiple transfers after first.
<code>s_apb_pready</code>	APB3 ready is a output from the slave to indicate its able to process the request.
<code>s_apb_pwrite</code>	APB3 Direction signal, active high is a write access. Active low is a read access.
<code>s_apb_pwdata</code>	APB3 write data port.
<code>s_apb_prdata</code>	APB3 read data port.
<code>s_apb_pslverror</code>	APB3 error indicates transfer failure, not implimented.
<code>up_rreq</code>	uP bus read request
<code>up_rack</code>	uP bus read ack
<code>up_raddr</code>	uP bus read address
<code>up_rdata</code>	uP bus read data
<code>up_wreq</code>	uP bus write request
<code>up_wack</code>	uP bus write ack
<code>up_waddr</code>	uP bus write address
<code>up_wdata</code>	uP bus write data

## VARIABLES

---

### valid

---

```
assign valid = s_apb_psel & s_apb_penable & rstn
```

This will add an extra clock cycle. since enable happens after select. both are needed to use the device.

### s\_apb\_pslverror

---

```
assign s_apb_pslverror = 1'b0
```

APB3 error is always 0, no error.

### up\_waddr

---

```
assign up_waddr = s_apb_paddr[ADDRESS_WIDTH-1:shift]
```

up\_waddr and s\_apb\_addr are a direct mapping.

## up\_waddr

---

up\_raddr and s\_apb\_addr are a direct mapping.

## up\_wdata

---

```
assign up_wdata = s_apb_pdata
```

up\_wdata and s\_apb\_pdata are a direct mapping.

## s\_apb\_prdata

---

```
assign s_apb_prdata = up_rdata
```

s\_apb\_prdata and up\_rdata are a direct mapping.

## up\_wreq

---

```
assign up_wreq = valid & s_apb_pwrite
```

uP write request is a combination of the APB3 valid and APB3 write select (active high is write).

## up\_rreq

---

```
assign up_rreq = valid & ~s_apb_pwrite
```

uP read request is a combination of the APB3 valid and APB3 write select (active low is read).

## s\_apb\_pready

---

```
assign s_apb_pready = (
    up_rack |                                up_wack |
    valid                                     ~
) & rstn
```

Diagrams seem to indicate that we should indicate ready when not sel and enable, which is why valid is complimented.