

---

## Verilog AXI Components Readme



For more information and updates: <http://alexforencich.com/wiki/en/verilog/axi/start>

GitHub repository: <https://github.com/alexforencich/verilog-axi>

### Introduction

Collection of AXI4 and AXI4 lite bus components. Most components are fully parametrizable in interface widths. Includes full cocotb testbenches that utilize cocotbext-axi.

### Documentation

#### **axi\_adapter module**

AXI width adapter module with parametrizable data and address interface widths. Supports INCR burst types and narrow bursts. Wrapper for [axi\\_adapter\\_rd](#) and [axi\\_adapter\\_wr](#).

#### **axi\_adapter\_rd module**

AXI width adapter module with parametrizable data and address interface widths. Supports INCR burst types and narrow bursts.

#### **axi\_adapter\_wr module**

AXI width adapter module with parametrizable data and address interface widths. Supports INCR burst types and narrow bursts.

#### **axi\_axil\_adapter module**

AXI to AXI lite converter and width adapter module with parametrizable data and address interface widths. Supports INCR burst types and narrow bursts. Wrapper for [axi\\_axil\\_adapter\\_rd](#) and [axi\\_axil\\_adapter\\_wr](#).

---

### **axi\_axil\_adapter\_rd module**

AXI to AXI lite converter and width adapter module with parametrizable data and address interface widths. Supports INCR burst types and narrow bursts.

### **axi\_axil\_adapter\_wr module**

AXI to AXI lite converter and width adapter module with parametrizable data and address interface widths. Supports INCR burst types and narrow bursts.

### **axi\_cdma module**

AXI to AXI DMA engine with parametrizable data and address interface widths. Generates full-width INCR bursts only, with parametrizable maximum burst length. Supports unaligned transfers, which can be disabled via parameter to save on resource consumption.

### **axi\_cdma\_desc\_mux module**

Descriptor multiplexer/demultiplexer for AXI CDMA module. Enables sharing the AXI CDMA module between multiple request sources, interleaving requests and distributing responses.

### **axi\_crossbar module**

AXI nonblocking crossbar interconnect with parametrizable data and address interface widths and master and slave interface counts. Supports all burst types. Fully nonblocking with completely separate read and write paths; ID-based transaction ordering protection logic; and per-port address decode, admission control, and decode error handling. Wrapper for [axi\\_crossbar\\_rd](#) and [axi\\_crossbar\\_wr](#).

Wrappers can generated with [axi\\_crossbar\\_wrap.py](#).

### **axi\_crossbar\_addr module**

Address decode and admission control module for AXI nonblocking crossbar interconnect.

---

### **axi\_crossbar\_rd module**

AXI nonblocking crossbar interconnect with parametrizable data and address interface widths and master and slave interface counts. Read interface only. Supports all burst types. Fully nonblocking with completely separate read and write paths; ID-based transaction ordering protection logic; and per-port address decode, admission control, and decode error handling.

### **axi\_crossbar\_wr module**

AXI nonblocking crossbar interconnect with parametrizable data and address interface widths and master and slave interface counts. Write interface only. Supports all burst types. Fully nonblocking with completely separate read and write paths; ID-based transaction ordering protection logic; and per-port address decode, admission control, and decode error handling.

### **axi\_dma module**

AXI to AXI stream DMA engine with parametrizable data and address interface widths. Generates full-width INCR bursts only, with parametrizable maximum burst length. Supports unaligned transfers, which can be disabled via parameter to save on resource consumption. Wrapper for [axi\\_dma\\_rd](#) and [axi\\_dma\\_wr](#).

### **axi\_dma\_desc\_mux module**

Descriptor multiplexer/demultiplexer for AXI DMA module. Enables sharing the AXI DMA module between multiple request sources, interleaving requests and distributing responses.

### **axi\_dma\_rd module**

AXI to AXI stream DMA engine with parametrizable data and address interface widths. Generates full-width INCR bursts only, with parametrizable maximum burst length. Supports unaligned transfers, which can be disabled via parameter to save on resource consumption.

### **axi\_dma\_wr module**

AXI stream to AXI DMA engine with parametrizable data and address interface widths. Generates full-width INCR bursts only, with parametrizable maximum burst length. Supports unaligned transfers, which can be disabled via parameter to save on resource consumption.

---

### **axi\_dp\_ram module**

AXI dual-port RAM with parametrizable data and address interface widths. Supports FIXED and INCR burst types as well as narrow bursts.

### **axi\_fifo module**

AXI FIFO with parametrizable data and address interface widths. Supports all burst types. Optionally can delay the address channel until either the write data is completely shifted into the FIFO or the read data FIFO has enough capacity to fit the whole burst. Wrapper for [axi\\_fifo\\_rd](#) and [axi\\_fifo\\_wr](#).

### **axi\_fifo\_rd module**

AXI FIFO with parametrizable data and address interface widths. AR and R channels only. Supports all burst types. Optionally can delay the address channel until either the read data FIFO is empty or has enough capacity to fit the whole burst.

### **axi\_fifo\_wr module**

AXI FIFO with parametrizable data and address interface widths. WR, W, and B channels only. Supports all burst types. Optionally can delay the address channel until the write data is shifted completely into the write data FIFO, or the current burst completely fills the write data FIFO.

### **axi\_interconnect module**

AXI shared interconnect with parametrizable data and address interface widths and master and slave interface counts. Supports all burst types. Small in area, but does not support concurrent operations.

Wrappers can be generated with [axi\\_interconnect\\_wrap.py](#).

### **axi\_ram module**

AXI RAM with parametrizable data and address interface widths. Supports FIXED and INCR burst types as well as narrow bursts.

---

### **axi\_ram\_rd\_if module**

AXI RAM read interface with parametrizable data and address interface widths. Handles bursts and presents a simplified internal memory interface. Supports FIXED and INCR burst types as well as narrow bursts.

### **axi\_ram\_wr\_if module**

AXI RAM write interface with parametrizable data and address interface widths. Handles bursts and presents a simplified internal memory interface. Supports FIXED and INCR burst types as well as narrow bursts.

### **axi\_ram\_wr\_rd\_if module**

AXI RAM read/write interface with parametrizable data and address interface widths. Handles bursts and presents a simplified internal memory interface. Supports FIXED and INCR burst types as well as narrow bursts. Wrapper for [axi\\_ram\\_rd\\_if](#) and [axi\\_ram\\_wr\\_if](#).

### **axi\_register module**

AXI register with parametrizable data and address interface widths. Supports all burst types. Inserts simple buffers or skid buffers into all channels. Channel register types can be individually changed or bypassed. Wrapper for [axi\\_register\\_rd](#) and [axi\\_register\\_wr](#).

### **axi\_register\_rd module**

AXI register with parametrizable data and address interface widths. AR and R channels only. Supports all burst types. Inserts simple buffers or skid buffers into all channels. Channel register types can be individually changed or bypassed.

### **axi\_register\_wr module**

AXI register with parametrizable data and address interface widths. WR, W, and B channels only. Supports all burst types. Inserts simple buffers or skid buffers into all channels. Channel register types can be individually changed or bypassed.

---

### **axil\_adapter module**

AXI lite width adapter module with parametrizable data and address interface widths. Wrapper for [axi\\_adapter\\_rd](#) and [axi\\_adapter\\_wr](#).

### **axil\_adapter\_rd module**

AXI lite width adapter module with parametrizable data and address interface widths.

### **axil\_adapter\_wr module**

AXI lite width adapter module with parametrizable data and address interface widths.

### **axil\_cdc module**

AXI lite clock domain crossing module with parametrizable data and address interface widths. Wrapper for [axi\\_cdc\\_rd](#) and [axi\\_cdc\\_wr](#).

### **axil\_cdc\_rd module**

AXI lite clock domain crossing module with parametrizable data and address interface widths.

### **axil\_cdc\_wr module**

AXI lite clock domain crossing module with parametrizable data and address interface widths.

### **axil\_crossbar module**

AXI lite nonblocking crossbar interconnect with parametrizable data and address interface widths and master and slave interface counts. Fully nonblocking with completely separate read and write paths; FIFO-based transaction ordering protection logic; and per-port address decode, admission control, and decode error handling. Wrapper for [axil\\_crossbar\\_rd](#) and [axil\\_crossbar\\_wr](#).

Wrappers can generated with [axil\\_crossbar\\_wrap.py](#).

### **axil\_crossbar\_addr module**

Address decode and admission control module for AXI lite nonblocking crossbar interconnect.

---

### **axil\_crossbar\_rd module**

AXI lite nonblocking crossbar interconnect with parametrizable data and address interface widths and master and slave interface counts. Read interface only. Fully nonblocking with completely separate read and write paths; FIFO-based transaction ordering protection logic; and per-port address decode, admission control, and decode error handling.

### **axil\_crossbar\_wr module**

AXI lite nonblocking crossbar interconnect with parametrizable data and address interface widths and master and slave interface counts. Write interface only. Fully nonblocking with completely separate read and write paths; FIFO-based transaction ordering protection logic; and per-port address decode, admission control, and decode error handling.

### **axil\_interconnect module**

AXI lite shared interconnect with parametrizable data and address interface widths and master and slave interface counts. Small in area, but does not support concurrent operations.

Wrappers can be generated with [axil\\_interconnect\\_wrap.py](#).

### **axil\_ram module**

AXI lite RAM with parametrizable data and address interface widths.

### **axil\_reg\_if module**

AXI lite register interface with parametrizable data and address interface widths. Can be used to assemble a set of control registers across multiple modules and hierarchy levels without complicated arbitration logic. Wrapper for [axil\\_reg\\_if\\_rd](#) and [axil\\_reg\\_if\\_wr](#).

### **axil\_reg\_if\_rd module**

AXI lite register interface with parametrizable data and address interface widths. Read direction only. Can be used to assemble a set of control registers across multiple modules and hierarchy levels without complicated arbitration logic.

---

### **axil\_reg\_if\_wr module**

AXI lite register interface with parametrizable data and address interface widths. Write direction only. Can be used to assemble a set of control registers across multiple modules and hierarchy levels without complicated arbitration logic.

### **axil\_register module**

AXI lite register with parametrizable data and address interface widths. Inserts skid buffers into all channels. Channel registers can be individually bypassed. Wrapper for [axil\\_register\\_rd](#) and [axil\\_register\\_wr](#).

### **axil\_register\_rd module**

AXI lite register with parametrizable data and address interface widths. AR and R channels only. Inserts simple buffers into all channels. Channel registers can be individually bypassed.

### **axil\_register\_wr module**

AXI lite register with parametrizable data and address interface widths. WR, W, and B channels only. Inserts simple buffers into all channels. Channel registers can be individually bypassed.

### **Common signals**

```
1  awid      : Write address ID
2  awaddr    : Write address
3  awlen     : Write burst length
4  awsize    : Write burst size
5  awburst   : Write burst type
6  awlock    : Write locking
7  awcache   : Write cache handling
8  awprot    : Write protection level
9  awqos     : Write QoS setting
10 awregion  : Write region
11 awuser    : Write user sideband signal
12 awvalid   : Write address valid
13 awready   : Write address ready (from slave)
14 wdata     : Write data
15 wstrb     : Write data strobe (byte select)
16 wlast     : Write data last transfer in burst
17 wuser     : Write data user sideband signal
```



---

```

18 wvalid    : Write data valid
19 wready    : Write data ready (from slave)
20 bid       : Write response ID
21 bresp     : Write response
22 buser     : Write response user sideband signal
23 bvalid    : Write response valid
24 bready    : Write response ready (from master)
25 arid      : Read address ID
26 araddr    : Read address
27 arlen     : Read burst length
28 arsize    : Read burst size
29 arburst   : Read burst type
30 arlock    : Read locking
31 arcache   : Read cache handling
32 arprot    : Read protection level
33 arqos     : Read QoS setting
34 arregion  : Read region
35 aruser    : Read user sideband signal
36 arvalid   : Read address valid
37 arready   : Read address ready (from slave)
38 rid       : Read data ID
39 rdata     : Read data
40 rresp     : Read response
41 rlast     : Read data last transfer in burst
42 ruser     : Read data user sideband signal
43 rvalid    : Read response valid
44 rready    : Read response ready (from master)

```

## Common parameters

```

1 ADDR_WIDTH      : width of awaddr and araddr signals
2 DATA_WIDTH     : width of wdata and rdata signals
3 STRB_WIDTH      : width of wstrb signal
4 ID_WIDTH        : width of *id signals
5 AWUSER_ENABLE   : enable awuser signal
6 AWUSER_WIDTH    : width of awuser signal
7 WUSER_ENABLE    : enable wuser signal
8 WUSER_WIDTH     : width of wuser signal
9 BUSER_ENABLE    : enable buser signal
10 BUSER_WIDTH     : width of buser signal
11 ARUSER_ENABLE   : enable aruser signal
12 ARUSER_WIDTH    : width of aruser signal
13 RUSER_ENABLE    : enable ruser signal
14 RUSER_WIDTH     : width of ruser signal

```

## Source Files

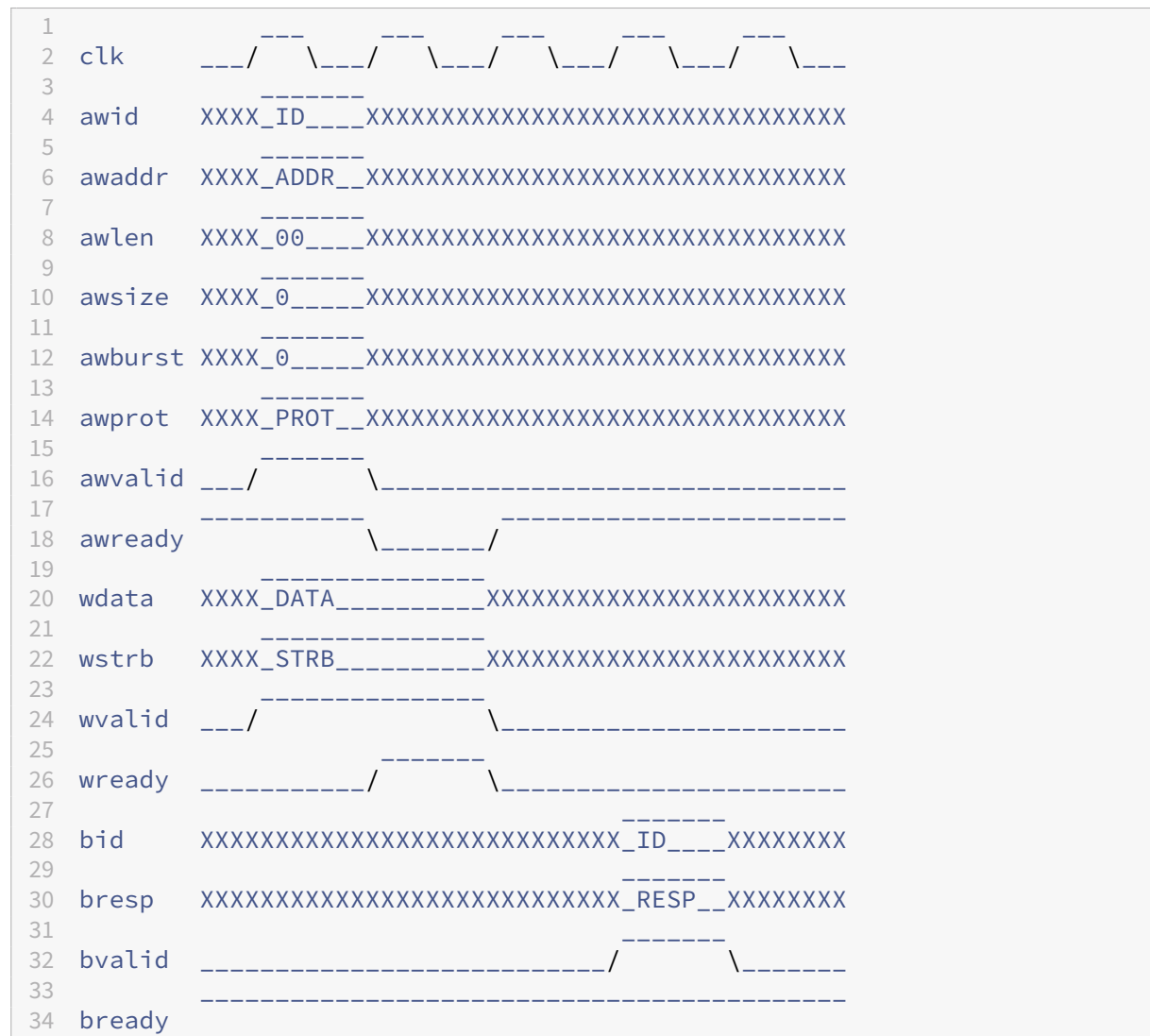
---

1	rtl/arbiter.v	: Parametrizable arbiter
2	rtl/axi_adapter.v	: AXI lite width converter
3	rtl/axi_adapter_rd.v	: AXI lite width converter (read)
4	rtl/axi_adapter_wr.v	: AXI lite width converter (write)
5	rtl/axi_axil_adapter.v	: AXI to AXI lite converter
6	rtl/axi_axil_adapter_rd.v	: AXI to AXI lite converter (read)
7	rtl/axi_axil_adapter_wr.v	: AXI to AXI lite converter (write)
8	rtl/axi_cdma.v	: AXI central DMA engine
9	rtl/axi_cdma_desc_mux.v	: AXI CDMA descriptor mux
10	rtl/axi_crossbar.v	: AXI nonblocking crossbar interconnect
11	rtl/axi_crossbar_addr.v	: AXI crossbar address module
12	rtl/axi_crossbar_rd.v	: AXI crossbar interconnect (read)
13	rtl/axi_crossbar_wr.v	: AXI crossbar interconnect (write)
14	rtl/axi_dma.v	: AXI DMA engine
15	rtl/axi_dma_desc_mux.v	: AXI DMA descriptor mux
16	rtl/axi_dma_rd.v	: AXI DMA engine (read)
17	rtl/axi_dma_wr.v	: AXI DMA engine (write)
18	rtl/axi_dp_ram.v	: AXI dual-port RAM
19	rtl/axi_fifo.v	: AXI FIFO
20	rtl/axi_fifo_rd.v	: AXI FIFO (read)
21	rtl/axi_fifo_wr.v	: AXI FIFO (write)
22	rtl/axi_interconnect.v	: AXI shared interconnect
23	rtl/axi_ram.v	: AXI RAM
24	rtl/axi_ram_rd_if.v	: AXI RAM read <b>interface</b>
25	rtl/axi_ram_wr_if.v	: AXI RAM write <b>interface</b>
26	rtl/axi_ram_wr_rd_if.v	: AXI RAM read/write <b>interface</b>
27	rtl/axi_register.v	: AXI register
28	rtl/axi_register_rd.v	: AXI register (read)
29	rtl/axi_register_wr.v	: AXI register (write)
30	rtl/axil_adapter.v	: AXI lite width converter
31	rtl/axil_adapter_rd.v	: AXI lite width converter (read)
32	rtl/axil_adapter_wr.v	: AXI lite width converter (write)
33	rtl/axil_cdc.v	: AXI lite CDC
34	rtl/axil_cdc_rd.v	: AXI lite CDC (read)
35	rtl/axil_cdc_wr.v	: AXI lite CDC (write)
36	rtl/axil_crossbar.v	: AXI lite nonblocking crossbar
	interconnect	
37	rtl/axil_crossbar_addr.v	: AXI lite crossbar address module
38	rtl/axil_crossbar_rd.v	: AXI lite crossbar interconnect (read)
39	rtl/axil_crossbar_wr.v	: AXI lite crossbar interconnect (write
	)	
40	rtl/axil_interconnect.v	: AXI lite shared interconnect
41	rtl/axil_ram.v	: AXI lite RAM
42	rtl/axil_reg_if.v	: AXI lite register <b>interface</b>
43	rtl/axil_reg_if_rd.v	: AXI lite register <b>interface</b> (read)
44	rtl/axil_reg_if_wr.v	: AXI lite register <b>interface</b> (write)
45	rtl/axil_register.v	: AXI lite register
46	rtl/axil_register_rd.v	: AXI lite register (read)
47	rtl/axil_register_wr.v	: AXI lite register (write)
48	rtl/priority_encoder.v	: Parametrizable priority encoder

---

## AXI4-Lite Interface Example

### Write



### Read



---

```

11
12 arbust XXXX_0_____XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
13
14 arprot XXXX_PROT_____XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15
16 arvalid ___/_____ \_____
17 _____
18 arready
19
20 rid      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX_ID_____XXXXXXX
21
22 rdata    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX_DATA_____XXXXXXX
23
24 rresp    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX_RESP_____XXXXXXX
25
26 rvalid   _____/_____ \_____
27 _____
28 rready

```

## Testing

Running the included testbenches requires cocotb, cocotbext-axi, and Icarus Verilog. The testbenches can be run with pytest directly (requires cocotb-test), pytest via tox, or via cocotb makefiles.