`include "axi\_bus.sv"

module axi\_node\_intf\_wrap

#(

parameter NB\_MASTER = 4,

parameter NB\_SLAVE = 4,

parameter AXI\_ADDR\_WIDTH = 32,

parameter AXI\_DATA\_WIDTH = 32,

parameter AXI\_ID\_WIDTH = 10,

parameter AXI\_USER\_WIDTH = 0

)

(

// Clock and Reset

input logic clk,

input logic rst\_n,

input logic test\_en\_i,

AXI\_BUS.Slave slave[NB\_SLAVE-1:0],

AXI\_BUS.Master master[NB\_MASTER-1:0],

// Memory map

input logic [NB\_MASTER-1:0][AXI\_ADDR\_WIDTH-1:0] start\_addr\_i,

input logic [NB\_MASTER-1:0][AXI\_ADDR\_WIDTH-1:0] end\_addr\_i

);

localparam AXI\_STRB\_WIDTH = AXI\_DATA\_WIDTH/8;

localparam NB\_REGION = 1;

// AXI ID WIDTHs for master and slave IPS

localparam AXI\_ID\_WIDTH\_TARG = AXI\_ID\_WIDTH;

localparam AXI\_ID\_WIDTH\_INIT = AXI\_ID\_WIDTH\_TARG + $clog2(NB\_SLAVE);

// Signals to slave periperhals

logic [NB\_MASTER-1:0][AXI\_ID\_WIDTH\_INIT-1:0] s\_master\_aw\_id;

logic [NB\_MASTER-1:0][AXI\_ADDR\_WIDTH-1:0] s\_master\_aw\_addr;

logic [NB\_MASTER-1:0][7:0] s\_master\_aw\_len;

logic [NB\_MASTER-1:0][2:0] s\_master\_aw\_size;

logic [NB\_MASTER-1:0][1:0] s\_master\_aw\_burst;

logic [NB\_MASTER-1:0] s\_master\_aw\_lock;

logic [NB\_MASTER-1:0][3:0] s\_master\_aw\_cache;

logic [NB\_MASTER-1:0][2:0] s\_master\_aw\_prot;

logic [NB\_MASTER-1:0][3:0] s\_master\_aw\_region;

logic [NB\_MASTER-1:0][AXI\_USER\_WIDTH-1:0] s\_master\_aw\_user;

logic [NB\_MASTER-1:0][3:0] s\_master\_aw\_qos;

logic [NB\_MASTER-1:0] s\_master\_aw\_valid;

logic [NB\_MASTER-1:0] s\_master\_aw\_ready;

logic [NB\_MASTER-1:0][AXI\_ID\_WIDTH\_INIT-1:0] s\_master\_ar\_id;

logic [NB\_MASTER-1:0][AXI\_ADDR\_WIDTH-1:0] s\_master\_ar\_addr;

logic [NB\_MASTER-1:0][7:0] s\_master\_ar\_len;

logic [NB\_MASTER-1:0][2:0] s\_master\_ar\_size;

logic [NB\_MASTER-1:0][1:0] s\_master\_ar\_burst;

logic [NB\_MASTER-1:0] s\_master\_ar\_lock;

logic [NB\_MASTER-1:0][3:0] s\_master\_ar\_cache;

logic [NB\_MASTER-1:0][2:0] s\_master\_ar\_prot;

logic [NB\_MASTER-1:0][3:0] s\_master\_ar\_region;

logic [NB\_MASTER-1:0][AXI\_USER\_WIDTH-1:0] s\_master\_ar\_user;

logic [NB\_MASTER-1:0][3:0] s\_master\_ar\_qos;

logic [NB\_MASTER-1:0] s\_master\_ar\_valid;

logic [NB\_MASTER-1:0] s\_master\_ar\_ready;

logic [NB\_MASTER-1:0][AXI\_DATA\_WIDTH-1:0] s\_master\_w\_data;

logic [NB\_MASTER-1:0][AXI\_STRB\_WIDTH-1:0] s\_master\_w\_strb;

logic [NB\_MASTER-1:0] s\_master\_w\_last;

logic [NB\_MASTER-1:0][AXI\_USER\_WIDTH-1:0] s\_master\_w\_user;

logic [NB\_MASTER-1:0] s\_master\_w\_valid;

logic [NB\_MASTER-1:0] s\_master\_w\_ready;

logic [NB\_MASTER-1:0][AXI\_ID\_WIDTH\_INIT-1:0] s\_master\_b\_id;

logic [NB\_MASTER-1:0][1:0] s\_master\_b\_resp;

logic [NB\_MASTER-1:0] s\_master\_b\_valid;

logic [NB\_MASTER-1:0][AXI\_USER\_WIDTH-1:0] s\_master\_b\_user;

logic [NB\_MASTER-1:0] s\_master\_b\_ready;

logic [NB\_MASTER-1:0][AXI\_ID\_WIDTH\_INIT-1:0] s\_master\_r\_id;

logic [NB\_MASTER-1:0][AXI\_DATA\_WIDTH-1:0] s\_master\_r\_data;

logic [NB\_MASTER-1:0][1:0] s\_master\_r\_resp;

logic [NB\_MASTER-1:0] s\_master\_r\_last;

logic [NB\_MASTER-1:0][AXI\_USER\_WIDTH-1:0] s\_master\_r\_user;

logic [NB\_MASTER-1:0] s\_master\_r\_valid;

logic [NB\_MASTER-1:0] s\_master\_r\_ready;

// Signals from AXI masters

logic [NB\_SLAVE-1:0][AXI\_ID\_WIDTH\_TARG-1:0] s\_slave\_aw\_id;

logic [NB\_SLAVE-1:0][AXI\_ADDR\_WIDTH-1:0] s\_slave\_aw\_addr;

logic [NB\_SLAVE-1:0][7:0] s\_slave\_aw\_len;

logic [NB\_SLAVE-1:0][2:0] s\_slave\_aw\_size;

logic [NB\_SLAVE-1:0][1:0] s\_slave\_aw\_burst;

logic [NB\_SLAVE-1:0] s\_slave\_aw\_lock;

logic [NB\_SLAVE-1:0][3:0] s\_slave\_aw\_cache;

logic [NB\_SLAVE-1:0][2:0] s\_slave\_aw\_prot;

logic [NB\_SLAVE-1:0][3:0] s\_slave\_aw\_region;

logic [NB\_SLAVE-1:0][AXI\_USER\_WIDTH-1:0] s\_slave\_aw\_user;

logic [NB\_SLAVE-1:0][3:0] s\_slave\_aw\_qos;

logic [NB\_SLAVE-1:0] s\_slave\_aw\_valid;

logic [NB\_SLAVE-1:0] s\_slave\_aw\_ready;

logic [NB\_SLAVE-1:0][AXI\_ID\_WIDTH\_TARG-1:0] s\_slave\_ar\_id;

logic [NB\_SLAVE-1:0][AXI\_ADDR\_WIDTH-1:0] s\_slave\_ar\_addr;

logic [NB\_SLAVE-1:0][7:0] s\_slave\_ar\_len;

logic [NB\_SLAVE-1:0][2:0] s\_slave\_ar\_size;

logic [NB\_SLAVE-1:0][1:0] s\_slave\_ar\_burst;

logic [NB\_SLAVE-1:0] s\_slave\_ar\_lock;

logic [NB\_SLAVE-1:0][3:0] s\_slave\_ar\_cache;

logic [NB\_SLAVE-1:0][2:0] s\_slave\_ar\_prot;

logic [NB\_SLAVE-1:0][3:0] s\_slave\_ar\_region;

logic [NB\_SLAVE-1:0][AXI\_USER\_WIDTH-1:0] s\_slave\_ar\_user;

logic [NB\_SLAVE-1:0][3:0] s\_slave\_ar\_qos;

logic [NB\_SLAVE-1:0] s\_slave\_ar\_valid;

logic [NB\_SLAVE-1:0] s\_slave\_ar\_ready;

logic [NB\_SLAVE-1:0][AXI\_DATA\_WIDTH-1:0] s\_slave\_w\_data;

logic [NB\_SLAVE-1:0][AXI\_STRB\_WIDTH-1:0] s\_slave\_w\_strb;

logic [NB\_SLAVE-1:0] s\_slave\_w\_last;

logic [NB\_SLAVE-1:0][AXI\_USER\_WIDTH-1:0] s\_slave\_w\_user;

logic [NB\_SLAVE-1:0] s\_slave\_w\_valid;

logic [NB\_SLAVE-1:0] s\_slave\_w\_ready;

logic [NB\_SLAVE-1:0][AXI\_ID\_WIDTH\_TARG-1:0] s\_slave\_b\_id;

logic [NB\_SLAVE-1:0][1:0] s\_slave\_b\_resp;

logic [NB\_SLAVE-1:0] s\_slave\_b\_valid;

logic [NB\_SLAVE-1:0][AXI\_USER\_WIDTH-1:0] s\_slave\_b\_user;

logic [NB\_SLAVE-1:0] s\_slave\_b\_ready;

logic [NB\_SLAVE-1:0][AXI\_ID\_WIDTH\_TARG-1:0] s\_slave\_r\_id;

logic [NB\_SLAVE-1:0][AXI\_DATA\_WIDTH-1:0] s\_slave\_r\_data;

logic [NB\_SLAVE-1:0][1:0] s\_slave\_r\_resp;

logic [NB\_SLAVE-1:0] s\_slave\_r\_last;

logic [NB\_SLAVE-1:0][AXI\_USER\_WIDTH-1:0] s\_slave\_r\_user;

logic [NB\_SLAVE-1:0] s\_slave\_r\_valid;

logic [NB\_SLAVE-1:0] s\_slave\_r\_ready;

// Signals Used to configure the AXI node

logic [NB\_REGION-1:0][NB\_MASTER-1:0][AXI\_ADDR\_WIDTH-1:0] s\_start\_addr;

logic [NB\_REGION-1:0][NB\_MASTER-1:0][AXI\_ADDR\_WIDTH-1:0] s\_end\_addr;

logic [NB\_REGION-1:0][NB\_MASTER-1:0] s\_valid\_rule;

logic [NB\_SLAVE-1:0][NB\_MASTER-1:0] s\_connectivity\_map;

generate

genvar i;

for(i = 0; i < NB\_MASTER; i++)

begin

assign master[i].aw\_id[AXI\_ID\_WIDTH\_INIT-1:0] = s\_master\_aw\_id[i];

assign master[i].aw\_addr = s\_master\_aw\_addr[i];

assign master[i].aw\_len = s\_master\_aw\_len[i];

assign master[i].aw\_size = s\_master\_aw\_size[i];

assign master[i].aw\_burst = s\_master\_aw\_burst[i];

assign master[i].aw\_lock = s\_master\_aw\_lock[i];

assign master[i].aw\_cache = s\_master\_aw\_cache[i];

assign master[i].aw\_prot = s\_master\_aw\_prot[i];

assign master[i].aw\_region = s\_master\_aw\_region[i];

assign master[i].aw\_user = s\_master\_aw\_user[i];

assign master[i].aw\_qos = s\_master\_aw\_qos[i];

assign master[i].aw\_valid = s\_master\_aw\_valid[i];

assign s\_master\_aw\_ready[i] = master[i].aw\_ready;

assign master[i].ar\_id[AXI\_ID\_WIDTH\_INIT-1:0] = s\_master\_ar\_id[i];

assign master[i].ar\_addr = s\_master\_ar\_addr[i];

assign master[i].ar\_len = s\_master\_ar\_len[i];

assign master[i].ar\_size = s\_master\_ar\_size[i];

assign master[i].ar\_burst = s\_master\_ar\_burst[i];

assign master[i].ar\_lock = s\_master\_ar\_lock[i];

assign master[i].ar\_cache = s\_master\_ar\_cache[i];

assign master[i].ar\_prot = s\_master\_ar\_prot[i];

assign master[i].ar\_region = s\_master\_ar\_region[i];

assign master[i].ar\_user = s\_master\_ar\_user[i];

assign master[i].ar\_qos = s\_master\_ar\_qos[i];

assign master[i].ar\_valid = s\_master\_ar\_valid[i];

assign s\_master\_ar\_ready[i] = master[i].ar\_ready;

assign master[i].w\_data = s\_master\_w\_data[i];

assign master[i].w\_strb = s\_master\_w\_strb[i];

assign master[i].w\_last = s\_master\_w\_last[i];

assign master[i].w\_user = s\_master\_w\_user[i];

assign master[i].w\_valid = s\_master\_w\_valid[i];

assign s\_master\_w\_ready[i] = master[i].w\_ready;

assign s\_master\_b\_id[i] = master[i].b\_id[AXI\_ID\_WIDTH\_INIT-1:0];

assign s\_master\_b\_resp[i] = master[i].b\_resp;

assign s\_master\_b\_valid[i] = master[i].b\_valid;

assign s\_master\_b\_user[i] = master[i].b\_user;

assign master[i].b\_ready = s\_master\_b\_ready[i];

assign s\_master\_r\_id[i] = master[i].r\_id[AXI\_ID\_WIDTH\_INIT-1:0];

assign s\_master\_r\_data[i] = master[i].r\_data;

assign s\_master\_r\_resp[i] = master[i].r\_resp;

assign s\_master\_r\_last[i] = master[i].r\_last;

assign s\_master\_r\_user[i] = master[i].r\_user;

assign s\_master\_r\_valid[i] = master[i].r\_valid;

assign master[i].r\_ready = s\_master\_r\_ready[i];

assign s\_start\_addr[0][i] = start\_addr\_i[i];

assign s\_end\_addr[0][i] = end\_addr\_i[i];

end

endgenerate

generate

genvar j;

for(j = 0; j < NB\_SLAVE; j++)

begin

assign s\_slave\_aw\_id[j] = slave[j].aw\_id[AXI\_ID\_WIDTH\_TARG-1:0];

assign s\_slave\_aw\_addr[j] = slave[j].aw\_addr;

assign s\_slave\_aw\_len[j] = slave[j].aw\_len;

assign s\_slave\_aw\_size[j] = slave[j].aw\_size;

assign s\_slave\_aw\_burst[j] = slave[j].aw\_burst;

assign s\_slave\_aw\_lock[j] = slave[j].aw\_lock;

assign s\_slave\_aw\_cache[j] = slave[j].aw\_cache;

assign s\_slave\_aw\_prot[j] = slave[j].aw\_prot;

assign s\_slave\_aw\_region[j] = slave[j].aw\_region;

assign s\_slave\_aw\_user[j] = slave[j].aw\_user;

assign s\_slave\_aw\_qos[j] = slave[j].aw\_qos;

assign s\_slave\_aw\_valid[j] = slave[j].aw\_valid;

assign slave[j].aw\_ready = s\_slave\_aw\_ready[j];

assign s\_slave\_ar\_id[j] = slave[j].ar\_id[AXI\_ID\_WIDTH\_TARG-1:0];

assign s\_slave\_ar\_addr[j] = slave[j].ar\_addr;

assign s\_slave\_ar\_len[j] = slave[j].ar\_len;

assign s\_slave\_ar\_size[j] = slave[j].ar\_size;

assign s\_slave\_ar\_burst[j] = slave[j].ar\_burst;

assign s\_slave\_ar\_lock[j] = slave[j].ar\_lock;

assign s\_slave\_ar\_cache[j] = slave[j].ar\_cache;

assign s\_slave\_ar\_prot[j] = slave[j].ar\_prot;

assign s\_slave\_ar\_region[j] = slave[j].ar\_region;

assign s\_slave\_ar\_user[j] = slave[j].ar\_user;

assign s\_slave\_ar\_qos[j] = slave[j].ar\_qos;

assign s\_slave\_ar\_valid[j] = slave[j].ar\_valid;

assign slave[j].ar\_ready = s\_slave\_ar\_ready[j];

assign s\_slave\_w\_data[j] = slave[j].w\_data;

assign s\_slave\_w\_strb[j] = slave[j].w\_strb;

assign s\_slave\_w\_last[j] = slave[j].w\_last;

assign s\_slave\_w\_user[j] = slave[j].w\_user;

assign s\_slave\_w\_valid[j] = slave[j].w\_valid;

assign slave[j].w\_ready = s\_slave\_w\_ready[j];

assign slave[j].b\_id[AXI\_ID\_WIDTH\_TARG-1:0] = s\_slave\_b\_id[j];

assign slave[j].b\_resp = s\_slave\_b\_resp[j];

assign slave[j].b\_valid = s\_slave\_b\_valid[j];

assign slave[j].b\_user = s\_slave\_b\_user[j];

assign s\_slave\_b\_ready[j] = slave[j].b\_ready;

assign slave[j].r\_id[AXI\_ID\_WIDTH\_TARG-1:0] = s\_slave\_r\_id[j];

assign slave[j].r\_data = s\_slave\_r\_data[j];

assign slave[j].r\_resp = s\_slave\_r\_resp[j];

assign slave[j].r\_last = s\_slave\_r\_last[j];

assign slave[j].r\_user = s\_slave\_r\_user[j];

assign slave[j].r\_valid = s\_slave\_r\_valid[j];

assign s\_slave\_r\_ready[j] = slave[j].r\_ready;

end

endgenerate

axi\_node

#(

.AXI\_ADDRESS\_W ( AXI\_ADDR\_WIDTH ),

.AXI\_DATA\_W ( AXI\_DATA\_WIDTH ),

.N\_MASTER\_PORT ( NB\_MASTER ),

.N\_SLAVE\_PORT ( NB\_SLAVE ),

.AXI\_ID\_IN ( AXI\_ID\_WIDTH\_TARG ),

.AXI\_USER\_W ( AXI\_USER\_WIDTH ),

.N\_REGION ( NB\_REGION )

)

axi\_node\_i

(

.clk ( clk ),

.rst\_n ( rst\_n ),

.test\_en\_i ( test\_en\_i ),

.slave\_awid\_i ( s\_slave\_aw\_id ),

.slave\_awaddr\_i ( s\_slave\_aw\_addr ),

.slave\_awlen\_i ( s\_slave\_aw\_len ),

.slave\_awsize\_i ( s\_slave\_aw\_size ),

.slave\_awburst\_i ( s\_slave\_aw\_burst ),

.slave\_awlock\_i ( s\_slave\_aw\_lock ),

.slave\_awcache\_i ( s\_slave\_aw\_cache ),

.slave\_awprot\_i ( s\_slave\_aw\_prot ),

.slave\_awregion\_i ( s\_slave\_aw\_region ),

.slave\_awqos\_i ( s\_slave\_aw\_qos ),

.slave\_awuser\_i ( s\_slave\_aw\_user ),

.slave\_awvalid\_i ( s\_slave\_aw\_valid ),

.slave\_awready\_o ( s\_slave\_aw\_ready ),

.slave\_wdata\_i ( s\_slave\_w\_data ),

.slave\_wstrb\_i ( s\_slave\_w\_strb ),

.slave\_wlast\_i ( s\_slave\_w\_last ),

.slave\_wuser\_i ( s\_slave\_w\_user ),

.slave\_wvalid\_i ( s\_slave\_w\_valid ),

.slave\_wready\_o ( s\_slave\_w\_ready ),

.slave\_bid\_o ( s\_slave\_b\_id ),

.slave\_bresp\_o ( s\_slave\_b\_resp ),

.slave\_buser\_o ( s\_slave\_b\_user ),

.slave\_bvalid\_o ( s\_slave\_b\_valid ),

.slave\_bready\_i ( s\_slave\_b\_ready ),

.slave\_arid\_i ( s\_slave\_ar\_id ),

.slave\_araddr\_i ( s\_slave\_ar\_addr ),

.slave\_arlen\_i ( s\_slave\_ar\_len ),

.slave\_arsize\_i ( s\_slave\_ar\_size ),

.slave\_arburst\_i ( s\_slave\_ar\_burst ),

.slave\_arlock\_i ( s\_slave\_ar\_lock ),

.slave\_arcache\_i ( s\_slave\_ar\_cache ),

.slave\_arprot\_i ( s\_slave\_ar\_prot ),

.slave\_arregion\_i ( s\_slave\_ar\_region ),

.slave\_aruser\_i ( s\_slave\_ar\_user ),

.slave\_arqos\_i ( s\_slave\_ar\_qos ),

.slave\_arvalid\_i ( s\_slave\_ar\_valid ),

.slave\_arready\_o ( s\_slave\_ar\_ready ),

.slave\_rid\_o ( s\_slave\_r\_id ),

.slave\_rdata\_o ( s\_slave\_r\_data ),

.slave\_rresp\_o ( s\_slave\_r\_resp ),

.slave\_rlast\_o ( s\_slave\_r\_last ),

.slave\_ruser\_o ( s\_slave\_r\_user ),

.slave\_rvalid\_o ( s\_slave\_r\_valid ),

.slave\_rready\_i ( s\_slave\_r\_ready ),

.master\_awid\_o ( s\_master\_aw\_id ),

.master\_awaddr\_o ( s\_master\_aw\_addr ),

.master\_awlen\_o ( s\_master\_aw\_len ),

.master\_awsize\_o ( s\_master\_aw\_size ),

.master\_awburst\_o ( s\_master\_aw\_burst ),

.master\_awlock\_o ( s\_master\_aw\_lock ),

.master\_awcache\_o ( s\_master\_aw\_cache ),

.master\_awprot\_o ( s\_master\_aw\_prot ),

.master\_awregion\_o ( s\_master\_aw\_region ),

.master\_awqos\_o ( s\_master\_aw\_qos ),

.master\_awuser\_o ( s\_master\_aw\_user ),

.master\_awvalid\_o ( s\_master\_aw\_valid ),

.master\_awready\_i ( s\_master\_aw\_ready ),

.master\_wdata\_o ( s\_master\_w\_data ),

.master\_wstrb\_o ( s\_master\_w\_strb ),

.master\_wlast\_o ( s\_master\_w\_last ),

.master\_wuser\_o ( s\_master\_w\_user ),

.master\_wvalid\_o ( s\_master\_w\_valid ),

.master\_wready\_i ( s\_master\_w\_ready ),

.master\_bid\_i ( s\_master\_b\_id ),

.master\_bresp\_i ( s\_master\_b\_resp ),

.master\_buser\_i ( s\_master\_b\_user ),

.master\_bvalid\_i ( s\_master\_b\_valid ),

.master\_bready\_o ( s\_master\_b\_ready ),

.master\_arid\_o ( s\_master\_ar\_id ),

.master\_araddr\_o ( s\_master\_ar\_addr ),

.master\_arlen\_o ( s\_master\_ar\_len ),

.master\_arsize\_o ( s\_master\_ar\_size ),

.master\_arburst\_o ( s\_master\_ar\_burst ),

.master\_arlock\_o ( s\_master\_ar\_lock ),

.master\_arcache\_o ( s\_master\_ar\_cache ),

.master\_arprot\_o ( s\_master\_ar\_prot ),

.master\_arregion\_o ( s\_master\_ar\_region ),

.master\_aruser\_o ( s\_master\_ar\_user ),

.master\_arqos\_o ( s\_master\_ar\_qos ),

.master\_arvalid\_o ( s\_master\_ar\_valid ),

.master\_arready\_i ( s\_master\_ar\_ready ),

.master\_rid\_i ( s\_master\_r\_id ),

.master\_rdata\_i ( s\_master\_r\_data ),

.master\_rresp\_i ( s\_master\_r\_resp ),

.master\_rlast\_i ( s\_master\_r\_last ),

.master\_ruser\_i ( s\_master\_r\_user ),

.master\_rvalid\_i ( s\_master\_r\_valid ),

.master\_rready\_o ( s\_master\_r\_ready ),

.cfg\_START\_ADDR\_i ( s\_start\_addr ),

.cfg\_END\_ADDR\_i ( s\_end\_addr ),

.cfg\_valid\_rule\_i ( s\_valid\_rule ),

.cfg\_connectivity\_map\_i ( s\_connectivity\_map )

);

assign s\_valid\_rule = '1;

assign s\_connectivity\_map = '1;

endmodule