`ifndef AXI\_BUS\_SV

`define AXI\_BUS\_SV

`include "config.sv"

////////////////////////////////////////////////////////////////////////////////

// Here, only generic definitions and functions are defined //

// . It is not designed to alter these functions. //

////////////////////////////////////////////////////////////////////////////////

`define OKAY 2'b00

interface AXI\_BUS

#(

parameter AXI\_ADDR\_WIDTH = 32,

parameter AXI\_DATA\_WIDTH = 64,

parameter AXI\_ID\_WIDTH = 10,

parameter AXI\_USER\_WIDTH = 6

);

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

logic [AXI\_ADDR\_WIDTH-1:0] aw\_addr;

logic [2:0] aw\_prot;

logic [3:0] aw\_region;

logic [7:0] aw\_len;

logic [2:0] aw\_size;

logic [1:0] aw\_burst;

logic aw\_lock;

logic [3:0] aw\_cache;

logic [3:0] aw\_qos;

logic [AXI\_ID\_WIDTH-1:0] aw\_id;

logic [AXI\_USER\_WIDTH-1:0] aw\_user;

logic aw\_ready;

logic aw\_valid;

logic [AXI\_ADDR\_WIDTH-1:0] ar\_addr;

logic [2:0] ar\_prot;

logic [3:0] ar\_region;

logic [7:0] ar\_len;

logic [2:0] ar\_size;

logic [1:0] ar\_burst;

logic ar\_lock;

logic [3:0] ar\_cache;

logic [3:0] ar\_qos;

logic [AXI\_ID\_WIDTH-1:0] ar\_id;

logic [AXI\_USER\_WIDTH-1:0] ar\_user;

logic ar\_ready;

logic ar\_valid;

logic w\_valid;

logic [AXI\_DATA\_WIDTH-1:0] w\_data;

logic [AXI\_STRB\_WIDTH-1:0] w\_strb;

logic [AXI\_USER\_WIDTH-1:0] w\_user;

logic w\_last;

logic w\_ready;

logic [AXI\_DATA\_WIDTH-1:0] r\_data;

logic [1:0] r\_resp;

logic r\_last;

logic [AXI\_ID\_WIDTH-1:0] r\_id;

logic [AXI\_USER\_WIDTH-1:0] r\_user;

logic r\_ready;

logic r\_valid;

logic [1:0] b\_resp;

logic [AXI\_ID\_WIDTH-1:0] b\_id;

logic [AXI\_USER\_WIDTH-1:0] b\_user;

logic b\_ready;

logic b\_valid;

// Master Side

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

modport Master

(

output aw\_valid, output aw\_addr, output aw\_prot, output aw\_region,

output aw\_len, output aw\_size, output aw\_burst, output aw\_lock,

output aw\_cache, output aw\_qos, output aw\_id, output aw\_user,

input aw\_ready,

output ar\_valid, output ar\_addr, output ar\_prot, output ar\_region,

output ar\_len, output ar\_size, output ar\_burst, output ar\_lock,

output ar\_cache, output ar\_qos, output ar\_id, output ar\_user,

input ar\_ready,

output w\_valid, output w\_data, output w\_strb, output w\_user, output w\_last,

input w\_ready,

input r\_valid, input r\_data, input r\_resp, input r\_last, input r\_id, input r\_user,

output r\_ready,

input b\_valid, input b\_resp, input b\_id, input b\_user,

output b\_ready

);

// Slave Side

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

modport Slave

(

input aw\_valid, input aw\_addr, input aw\_prot, input aw\_region,

input aw\_len, input aw\_size, input aw\_burst, input aw\_lock,

input aw\_cache, input aw\_qos, input aw\_id, input aw\_user,

output aw\_ready,

input ar\_valid, input ar\_addr, input ar\_prot, input ar\_region,

input ar\_len, input ar\_size, input ar\_burst, input ar\_lock,

input ar\_cache, input ar\_qos, input ar\_id, input ar\_user,

output ar\_ready,

input w\_valid, input w\_data, input w\_strb, input w\_user, input w\_last,

output w\_ready,

output r\_valid, output r\_data, output r\_resp, output r\_last, output r\_id, output r\_user,

input r\_ready,

output b\_valid, output b\_resp, output b\_id, output b\_user,

input b\_ready

);

endinterface

`endif