class mem\_scoreboard extends uvm\_scoreboard;

`uvm\_component\_utils(mem\_scoreboard)

mem\_sequence\_item req,req1;

reg [1:0] sel\_bit;

reg [31:0] address;

reg [31:0] write\_data;

reg [31:0] read\_data;

// tlm port declaration

uvm\_analysis\_imp#(mem\_sequence\_item, mem\_scoreboard) analysis\_export;//item\_collected\_export

function new(string name = "mem\_scoreboard", uvm\_component parent);

super.new(name, parent);

endfunction

virtual function void build\_phase(uvm\_phase phase);

super.build\_phase(phase);

// creating memory for handles

req = mem\_sequence\_item::type\_id::create("req");

analysis\_export = new("analysis\_export", this);

req1 = mem\_sequence\_item::type\_id::create("req1");

endfunction

virtual function void write(mem\_sequence\_item req);

// //receiving packets from monitor

req1.enable=req.enable;

req1.dina=req.dina;

req1.dinb=req.dinb;

req1.addr=req.addr;

req1.wr=req.wr;

req1.slave\_sel<=req.slave\_sel;

if (req1.wr==1)

begin

sel\_bit=req1.slave\_sel;

address=req1.addr;

write\_data=req1.dina+req1.dinb;

req1.dout=read\_data;

end

else if(req1.enable==1 && req1.wr==0)

begin

if (req1.slave\_sel==sel\_bit && req1.addr==address)

begin

req1.dout=write\_data;

read\_data=req1.dout;

end

else

req1.dout=read\_data;

end

if (req1.dout == req.dout)

begin

$display("outputs matched");

req.print();

req1.print();

end

else

begin

$display("outputs not matched");

req.print();

req1.print();

end

endfunction

endclass