`include "config.sv"

module instr\_ram\_wrap

#(

parameter RAM\_SIZE = 32768, // in bytes

parameter ADDR\_WIDTH = $clog2(RAM\_SIZE) + 1, // one bit more than necessary, for the boot rom

parameter DATA\_WIDTH = 32

)(

// Clock and Reset

input logic clk,

input logic rst\_n,

input logic en\_i,

input logic [ADDR\_WIDTH-1:0] addr\_i,

input logic [DATA\_WIDTH-1:0] wdata\_i,

output logic [DATA\_WIDTH-1:0] rdata\_o,

input logic we\_i,

input logic [DATA\_WIDTH/8-1:0] be\_i,

input logic bypass\_en\_i

);

logic is\_boot, is\_boot\_q;

logic [DATA\_WIDTH-1:0] rdata\_boot;

logic [DATA\_WIDTH-1:0] rdata\_ram;

assign is\_boot = (addr\_i[ADDR\_WIDTH-1] == 1'b1);

sp\_ram\_wrap

#(

.RAM\_SIZE ( RAM\_SIZE ),

.DATA\_WIDTH ( DATA\_WIDTH )

)

sp\_ram\_wrap\_i

(

.clk ( clk ),

.rstn\_i ( rst\_n ),

.en\_i ( en\_i & (~is\_boot) ),

.addr\_i ( addr\_i[ADDR\_WIDTH-2:0] ),

.wdata\_i ( wdata\_i ),

.rdata\_o ( rdata\_ram ),

.we\_i ( we\_i ),

.be\_i ( be\_i ),

.bypass\_en\_i ( bypass\_en\_i )

);

boot\_rom\_wrap

#(

.DATA\_WIDTH ( DATA\_WIDTH )

)

boot\_rom\_wrap\_i

(

.clk ( clk ),

.rst\_n ( rst\_n ),

.en\_i ( en\_i & is\_boot ),

.addr\_i ( addr\_i[`ROM\_ADDR\_WIDTH-1:0] ),

.rdata\_o ( rdata\_boot )

);

assign rdata\_o = (is\_boot\_q == 1'b1) ? rdata\_boot : rdata\_ram;

// Delay the boot signal for one clock cycle to correctly select the rdata

// from boot rom vs normal ram

always\_ff @(posedge clk, negedge rst\_n)

begin

if (rst\_n == 1'b0)

is\_boot\_q <= 1'b0;

else

is\_boot\_q <= is\_boot;

end

endmodule