module i2c\_translator (clk,rst,scl\_in,sda\_in,scl\_out1,sda\_out1,scl\_out2,sda\_out2);

input wire clk,rst,scl\_in,sda\_in;

output reg scl\_out1,scl\_out2;

inout wire sda\_out1,sda\_out2;

reg [7:0] s\_reg,d\_buf;

reg [3:0] bit\_cnt,slave\_st,master\_st;

reg [1:0] slave\_sel;

reg [15:0] clk\_div;

reg scl\_gen;

reg sda\_in\_out,sda\_in\_en,sda\_out1\_out,sda\_out1\_en,sda\_out2\_out,sda\_out2\_en;

assign sda\_in = sda\_in\_en ? sda\_in\_out : 1'bz;

assign sda\_out1 = sda\_out1\_en ? sda\_out1\_out : 1'bz;

assign sda\_out2 = sda\_out2\_en ? sda\_out2\_out : 1'bz;

// FSM

localparam SLAVE\_IDLE = 0,SLAVE\_ADDR = 1,SLAVE\_ACK = 2,SLAVE\_WRITE = 3;

localparam SLAVE\_READ = 4,SLAVE\_STOP = 5;

localparam MASTER\_IDLE = 0,MASTER\_START = 1,MASTER\_ADDR = 2,MASTER\_ACK = 3;

localparam MASTER\_WRITE = 4,MASTER\_WRITE\_ACK = 5,MASTER\_READ = 6,MASTER\_READ\_ACK = 7;

localparam MASTER\_STOP = 8;

// Translator visible(varitual) addresses

localparam VIRT\_ADDR1 = 7'h49;

localparam VIRT\_ADDR2 = 7'h4A;

//To physical addr 0x48

localparam TARGET\_ADDR = 7'h48;

// Slave FSM

always @(posedge clk or posedge rst)

begin

if (rst)

begin

slave\_st = SLAVE\_IDLE;

sda\_in\_en = 0;

slave\_sel = 0;

end

else

begin

case (slave\_st)

SLAVE\_IDLE: begin

slave\_st = SLAVE\_ADDR;

bit\_cnt = 7;

sda\_in\_en = 0;

end

SLAVE\_ADDR: begin

d\_buf[bit\_cnt] = sda\_in;

if (bit\_cnt == 0)begin slave\_st = SLAVE\_ACK; s\_reg=d\_buf; end

else begin bit\_cnt = bit\_cnt - 1; slave\_st = SLAVE\_ADDR; end

end

SLAVE\_ACK: begin

if (s\_reg[7:1] == VIRT\_ADDR1)

begin

slave\_sel = 1;

sda\_in\_en = 1;

sda\_in\_out = 0;

slave\_st =SLAVE\_STOP;

master\_st=MASTER\_START;

bit\_cnt = 7;

sda\_in\_en = 0;

end

else if (s\_reg[7:1] == VIRT\_ADDR2)

begin

slave\_sel = 2;

sda\_in\_en = 1;

sda\_in\_out = 0;

slave\_st =SLAVE\_STOP;

master\_st=MASTER\_START;

bit\_cnt = 7;

sda\_in\_en = 0;

end

else

begin

slave\_st = SLAVE\_IDLE;

sda\_in\_en = 0;

end

end

SLAVE\_WRITE: begin

d\_buf[bit\_cnt] = sda\_in;

if (bit\_cnt == 0)

begin

slave\_st = SLAVE\_STOP;

master\_st = MASTER\_WRITE; // trigger master FSM

s\_reg=d\_buf;

end

else begin bit\_cnt = bit\_cnt - 1; slave\_st = SLAVE\_WRITE; end

end

SLAVE\_READ: begin

sda\_in\_en = 1;

sda\_in\_out = s\_reg[bit\_cnt];

if (bit\_cnt == 0)

begin

slave\_st = SLAVE\_STOP;

end

else begin bit\_cnt = bit\_cnt - 1; slave\_st = SLAVE\_READ; end

end

SLAVE\_STOP: begin

sda\_in\_en = 0;

slave\_st = SLAVE\_STOP;

end

endcase

end

end

// clock divider for terget bus

always @(posedge clk or posedge rst)

begin

if (rst)

begin

clk\_div = 0;

scl\_gen = 1;

end

else

begin

if (clk\_div == 250)

begin // for ~100kHz from 50MHz

clk\_div = 0;

scl\_gen = ~scl\_gen;

end

else

begin

clk\_div = clk\_div + 1;

end

end

end

// Master FSM

always @(posedge scl\_gen or posedge rst)

begin

if (rst)

begin

master\_st = MASTER\_IDLE;

scl\_out1 = 1;

scl\_out2 = 1;

sda\_out1\_en = 0;

sda\_out2\_en = 0;

end

else

begin

case (master\_st)

MASTER\_IDLE: begin

sda\_out1\_en =0;

sda\_out2\_en =0;

master\_st=MASTER\_IDLE;

end

MASTER\_START: begin

if (slave\_sel == 1)

begin

scl\_out1 = 1;

sda\_out1\_en = 1;

sda\_out1\_out = 0;

s\_reg={TARGET\_ADDR ,s\_reg[0]};

end

else if (slave\_sel == 2)

begin

scl\_out2 = 1;

sda\_out2\_en = 1;

sda\_out2\_out = 0;

s\_reg={TARGET\_ADDR ,s\_reg[0]};

end

bit\_cnt = 7;

master\_st = MASTER\_ADDR;

end

MASTER\_ADDR: begin

if (slave\_sel == 1)

begin

scl\_out1 = 0;

sda\_out1\_out = s\_reg[bit\_cnt];

scl\_out1 = 1;

if (bit\_cnt == 0) master\_st= MASTER\_ACK;

else begin bit\_cnt = bit\_cnt - 1; master\_st=MASTER\_ADDR; end

end

else if (slave\_sel == 2)

begin

scl\_out2 = 0;

sda\_out2\_out = s\_reg[bit\_cnt];

scl\_out2 = 1;

if (bit\_cnt == 0) begin master\_st = MASTER\_ACK; end

else begin bit\_cnt = bit\_cnt - 1; master\_st<=MASTER\_ADDR; end

end

end

MASTER\_ACK: begin

if (slave\_sel == 1)

begin

scl\_out1 = 0;

sda\_out1\_en = 0;

scl\_out1 = 1;

if (sda\_out1== 0)

begin

if (s\_reg [0]== 0)

begin

s\_reg = s\_reg ;

bit\_cnt = 7;

master\_st = MASTER\_IDLE;

slave\_st = SLAVE\_WRITE;

end

else

begin

bit\_cnt = 7;

master\_st = MASTER\_READ;

end

end

end

else if (slave\_sel == 2)

begin

scl\_out2 = 0;

sda\_out2\_en =0;

scl\_out2 = 1;

if (sda\_out2== 0)

begin

if (s\_reg [0]== 0)

begin

s\_reg = s\_reg ;

bit\_cnt = 7;

master\_st = MASTER\_IDLE;

slave\_st = SLAVE\_WRITE;

end

else

begin

bit\_cnt = 7;

master\_st = MASTER\_READ;

end

end

end

else master\_st = MASTER\_STOP; // no ACK/

end

MASTER\_WRITE: begin

if (slave\_sel == 1)

begin

scl\_out1 = 0;

sda\_out1\_en = 1;

sda\_out1\_out = s\_reg[bit\_cnt];

scl\_out1 = 1;

if (bit\_cnt == 0) master\_st = MASTER\_WRITE\_ACK;

else begin bit\_cnt = bit\_cnt - 1; master\_st = MASTER\_WRITE; end

end

else if (slave\_sel == 2)

begin

scl\_out2 = 0;

sda\_out2\_en = 1;

sda\_out2\_out = s\_reg[bit\_cnt];

scl\_out2 = 1;

if (bit\_cnt == 0) master\_st = MASTER\_WRITE\_ACK;

else begin bit\_cnt = bit\_cnt - 1; master\_st = MASTER\_WRITE; end

end

end

MASTER\_WRITE\_ACK: begin

if (slave\_sel == 1)

begin

scl\_out1 = 0;

sda\_out1\_en = 0;

scl\_out1 = 1;

if (sda\_in == 0) master\_st = MASTER\_STOP;

else master\_st = MASTER\_IDLE;

end

else if (slave\_sel == 2)

begin

scl\_out2 = 0;

sda\_out2\_en = 0;

scl\_out2 = 1;

if (sda\_in == 0) master\_st = MASTER\_STOP;

else master\_st = MASTER\_IDLE;

end

end

MASTER\_READ: begin

if (slave\_sel == 1)

begin

scl\_out1 = 0;

sda\_out1\_en =0;

s\_reg[bit\_cnt] = sda\_out1;

scl\_out1 =1;

if (bit\_cnt ==0) master\_st = MASTER\_READ\_ACK;

else begin bit\_cnt = bit\_cnt - 1; master\_st = MASTER\_READ; end

end

else if (slave\_sel == 2)

begin

scl\_out2 = 0;

sda\_out2\_en = 0;

s\_reg[bit\_cnt] = sda\_out2;

scl\_out2 = 1;

if (bit\_cnt == 0) master\_st = MASTER\_READ\_ACK;

else begin bit\_cnt = bit\_cnt - 1; master\_st = MASTER\_READ; end

end

end

MASTER\_READ\_ACK: begin

if (slave\_sel == 1)

begin

scl\_out1 = 0;

sda\_out1\_en = 1;

scl\_out1 = 1;

sda\_out1\_out = 1;

master\_st = MASTER\_IDLE;

slave\_st = SLAVE\_READ;

end

else if (slave\_sel == 2)

begin

scl\_out2 = 0;

sda\_out2\_en = 1;

scl\_out2 = 1;

sda\_out2\_out = 1;

master\_st = MASTER\_IDLE;

slave\_st = SLAVE\_READ;

end

end

MASTER\_STOP: begin

if (slave\_sel == 1)

begin

scl\_out1 = 1;

sda\_out1\_en = 1;

sda\_out1\_out = 1;

end

else if (slave\_sel == 2)

begin

scl\_out2 = 1;

sda\_out2\_en = 1;

sda\_out2\_out = 1;

end

master\_st = MASTER\_IDLE;

end

endcase

end

end

endmodule