a)

module trap\_edge(

input logic async\_sig, clk, reset,

output logic trapped\_edge);

logic Q;

always\_ff @(posedge async\_sig, posedge reset) begin

if(reset) Q <= 1'b0;

else Q <= 1'b1;

end

always\_ff @(posedge clk, posedge reset) begin

if(reset) trapped\_edge <= 1'b0;

else trapped\_edge <= Q;

end

endmodule

文本

描述已自动生成

图示

描述已自动生成

b)

module shared\_access\_to\_one\_state\_machine

#(

parameter N = 32,

parameter M = 8

)

(

input target\_state\_machine\_finished,

input sm\_clk,

input logic start\_request\_a,

input logic start\_request\_b,

input [(N-1):0] input\_arguments\_a,

input [(N-1):0] input\_arguments\_b,

input reset,

input [M-1:0] in\_received\_data,

output logic finish\_a,//

output logic finish\_b,//

output logic reset\_start\_request\_a,//

output logic reset\_start\_request\_b,//

output reg [(M-1):0] received\_data\_a,

output reg [(M-1):0] received\_data\_b,

output reg [(N-1):0] output\_arguments,

output start\_target\_state\_machine//

);

logic [11:0] state;

parameter check\_start\_a = 12'b0000\_00000000;

parameter give\_start\_a = 12'b0001\_01100000;

parameter wait\_for\_finish\_a = 12'b0010\_00000000;

parameter register\_data\_a = 12'b0011\_00000010;

parameter give\_finish\_a = 12'b0100\_00001000;

parameter check\_start\_b = 12'b0101\_10000000;

parameter give\_start\_b = 12'b0110\_11010000;

parameter wait\_for\_finish\_b = 12'b0111\_10000000;

parameter register\_data\_b = 12'b1000\_10000001;

parameter give\_finish\_b = 12'b1001\_10000100;

logic select\_b\_output\_parameters, register\_data\_a\_enable, register\_data\_b\_enable;

//list the output with different state

assign register\_data\_b\_enable = state[0];

assign register\_data\_a\_enable = state[1];

assign finish\_b = state[2];

assign finish\_a = state[3];

assign reset\_start\_request\_b = state[4];

assign reset\_start\_request\_a = state[5];

assign start\_target\_state\_machine = state[6];

assign select\_b\_output\_parameters = state[7];

//select\_b\_output\_parameters: the function of this state machine output is (in pseudo-code):

assign output\_arguments = (select\_b\_output\_parameters) ? input\_arguments\_b : input\_arguments\_a;

//register\_data\_a\_enable & register\_data\_b\_enable:

always\_ff @(posedge sm\_clk) begin

if(register\_data\_a\_enable) received\_data\_a <= in\_received\_data;

if(register\_data\_b\_enable) received\_data\_b <= in\_received\_data;

end

//state machine

always\_ff @(posedge sm\_clk, posedge reset)begin

if(reset) state <= check\_start\_a;

else

case(state)

check\_start\_a: begin

if (!start\_request\_a) state <= check\_start\_b;

else if (start\_request\_a) state <= give\_start\_a;

else state <= check\_start\_a;

end

give\_start\_a: state <= wait\_for\_finish\_a;

wait\_for\_finish\_a: begin

if (!target\_state\_machine\_finished) state <= wait\_for\_finish\_a;

else if (target\_state\_machine\_finished) state <= register\_data\_a;

else state <= wait\_for\_finish\_a;

end

register\_data\_a: state <= give\_finish\_a;

give\_finish\_a: state <= check\_start\_b;

check\_start\_b: begin

if (!start\_request\_b) state <= check\_start\_a;

else if (start\_request\_b) state <= give\_start\_b;

else state <= check\_start\_b;

end

give\_start\_b: state <= wait\_for\_finish\_b;

wait\_for\_finish\_b: begin

if (!target\_state\_machine\_finished) state <= wait\_for\_finish\_b;

else if (target\_state\_machine\_finished) state <= register\_data\_b;

else state <= wait\_for\_finish\_b;

end

register\_data\_b: state <= give\_finish\_b;

give\_finish\_b: state <= check\_start\_a;

default: state <= check\_start\_a;

endcase

end

endmodule

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

c) 图示, 示意图

描述已自动生成

d) 应用程序

中度可信度描述已自动生成

when start\_request\_a is 1, the state chenge to give\_start\_a(0001/output:01100000) from check\_start\_a(0000/output:00000000)

then the state change to wait\_for\_finish\_a(0010/output:00000000)

when target\_state\_machine\_finished is 0, then the state change to wait\_for\_finish\_a(0010/output:00000000)

when target\_state\_machine\_finished is 1, then the state change to register\_data\_a(0011/output:00000010)

Then the state change to give\_finish\_a(0100/output:00001000), and check\_start\_b(0101/output:10000000)

when start\_request\_b is 1, the state chenge to give\_start\_b(0111/output:10000000) from check\_start\_b(0110/output:11010000)

then the state change to wait\_for\_finish\_b(0111/output:10000000)

when target\_state\_machine\_finished is 0, then the state change to wait\_for\_finish\_b(0111/output:10000000)

when target\_state\_machine\_finished is 1, then the state change to register\_data\_b(1000/output:10000001)

Then the state change to give\_finish\_b(0101/output:10000100), and check\_start\_a(0000/output:00000000)

If reset is 1, the state is check\_start\_a,

If (select\_b\_output\_parameters) = 1, output\_arguments = input\_arguments\_b, if (select\_b\_output\_parameters) = 0, output\_arguments = input\_arguments\_a

e)