Jawad ibrahem

207416207

**EXE Home 4**

Code Of the FSM :

module SM\_Arbiter(clk,req,rst,gnt);

input clk;

input rst;

input [3:0] req;

output reg [3:0] gnt;

reg [2:0] state ,nextState;

parameter IDLE = 3'b000;

parameter GNT0 = 3'b001;

parameter GNT1 = 3'b010;

parameter GNT2 = 3'b011;

parameter GNT3 = 3'b100;

always @(posedge clk or posedge rst)

begin

if(rst)

state <= IDLE;

else

state <=nextState;

end

always @(state)

begin

gnt<=4'b0000;

case(state)

IDLE:gnt <= 4'b0000;

GNT0:gnt <= 4'b0001;

GNT1:gnt <= 4'b0010;

GNT2:gnt <= 4'b0100;

GNT3:gnt <= 4'b1000;

default:gnt <= 4'b0000;

endcase

end

always @(state or req)

begin

case(state)

IDLE :case(req)

4'b0000: nextState <= IDLE;

4'b0001: nextState <= GNT0;

4'b0010: nextState <= GNT1;

4'b0100: nextState <= GNT2;

4'b1000: nextState <= GNT3;

default: nextState <= IDLE;

endcase

GNT0: if(req[0]) nextState <= GNT0;

else nextState <= IDLE;

GNT1: if(req[1]) nextState <= GNT1;

else nextState <= IDLE;

GNT2: if(req[2]) nextState <= GNT2;

else nextState <= IDLE;

GNT3: if(req[3]) nextState <= GNT3;

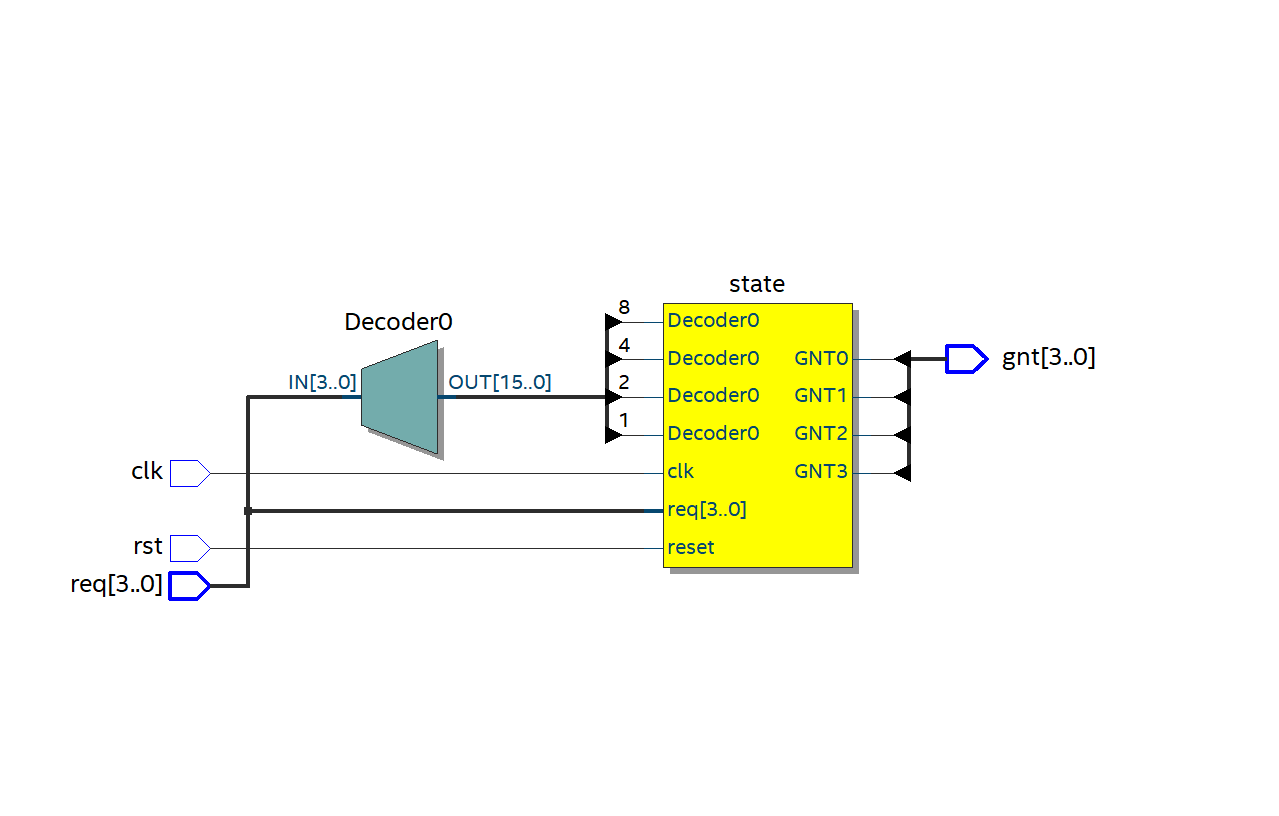
else nextState <= IDLE;

endcase

end

endmodule

RTL View :



State machine :

Diagram

Description automatically generated

Code of the testbench :

`timescale 10ns/10ns

module SM\_Arbiter\_tb;

parameter clk = 2;

integer i;

reg t\_clk , t\_rst;

reg [3:0] t\_req;

wire [3:0] t\_gnt;

reg [3:0] test\_data[4:0];

SM\_Arbiter DUT(.clk(t\_clk),.rst(t\_rst),.req(t\_req),.gnt(t\_gnt));

initial begin

t\_clk = 0;

test\_data[0] = 4'b0001;

test\_data[1] = 4'b0010;

test\_data[2] = 4'b0100;

test\_data[3] = 4'b1000;

test\_data[4] = 4'b0000;

i = 1;

end

always

forever #(clk/2) t\_clk = ~t\_clk;

initial begin

t\_rst = 1'b1;

t\_req = 4'b0000;

#2 t\_rst = 1'b0;

for(i=0 ; i<=4 ; i =i+1)

begin

#4

t\_req = test\_data[i];

end

end

endmodule

the result of the test bench :

