mealy\_sequence\_detector\_3processes：

**`timescale 1ns / 1ps**

**/////////////////////////////////////////////////////////////////**

**// Module Name: mealy\_sequence\_detector\_3processes**

**/////////////////////////////////////////////////////////////////**

**module mealy\_sequence\_detector\_3processes(**

**input clk,**

**input reset,**

**input ain,**

**output reg [3:0] count,**

**output reg yout**

**);**

**reg [2:0] state, nextstate;**

**parameter [2:0] S0=0, S1=1, S2=2, S3=3, S4=4, S5=5;**

**always @(posedge clk or posedge reset)**

**if (reset)**

**begin**

**count <= 0;**

**state <= S0;**

**end**

**else**

**begin**

**if(ain)**

**count <= count + 1;**

**state <= nextstate;**

**end**

**always @(state or ain)**

**begin**

**begin**

**yout = 1'b0;**

**case(state)**

**S0: if(!ain)**

**yout = 1;**

**S3: if(ain)**

**yout = 1;**

**endcase**

**end**

**end**

**always @(state or ain)**

**begin**

**begin**

**case(state)**

**S0: if(ain) //1st 1 counted**

**nextstate = S1;**

**else**

**nextstate = S0;**

**S1: if(ain) //2nd 1 counted**

**nextstate = S2;**

**else**

**nextstate = S1;**

**S2: if(ain) //3rd 1 counted**

**nextstate = S3;**

**else**

**nextstate = S2;**

**S3: if(ain) //4th 1 counted**

**nextstate = S4;**

**else**

**nextstate = S3;**

**S4: if(ain) //5th 1 counted**

**nextstate = S5;**

**else**

**nextstate = S4;**

**S5: if(ain) //6th 1 counted**

**nextstate = S3;**

**else**

**nextstate = S5;**

**default: nextstate = S0;**

**endcase**

**end**

**end**

**endmodule**

mealy\_sequence\_detector\_3processes\_tb：

窗体底端

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Module Name: mealy\_sequence\_detector\_3processes\_tb

//////////////////////////////////////////////////////////////////////////////////

module mealy\_sequence\_detector\_3processes\_tb(

);

reg clk;

reg reset;

reg ain;

wire [3:0] count;

wire yout;

mealy\_sequence\_detector\_3processes DUT (.clk(clk), .reset(reset), .ain(ain), .count(count), .yout(yout));

initial

begin

clk = 0;

forever

begin

#5 clk = 1;

#5 clk = 0;

end

end

initial

begin

reset = 1'b1;

ain = 1'b0;

#20; // wait for 2 clock cycles

reset = 1'b0;

#10 ain = 1'b0;

#10 ain = 1'b1;

#20 ain = 1'b0;

#60 ain = 1'b1;

#40 ain = 1'b0;

#20 ain = 1'b1;

#10 reset = 1'b1;

#10 reset = 1'b0;

#10 ain = 1'b0;

#30 ain = 1'b1;

#40;

end

endmodule

**摩尔**

`timescale 1ns / 1ps

/////////////////////////////////////////////////////////////////

// Module Name: moore\_sequence\_detector\_3processes

/////////////////////////////////////////////////////////////////

module moore\_sequence\_detector\_3processes(

input clk,

input reset,

input [1:0] ain,

output reg yout

);

reg [1:0] state, nextstate;

parameter [1:0] S0=0, S1=1, S2=2, S3=3;

always @(posedge clk or posedge reset)

if (reset)

state <= S0;

else

state <= nextstate;

always @(state or ain)

begin

case(state)

S0: case(ain)

2'b00: nextstate = S0;

2'b01: nextstate = S0;

2'b10: nextstate = S1;

2'b11: nextstate = S1;

endcase

S1: case(ain)

2'b00: nextstate = S2;

2'b01: nextstate = S0;

2'b10: nextstate = S1;

2'b11: nextstate = S1;

endcase

S2: case(ain)

2'b00: nextstate = S2;

2'b01: nextstate = S3;

2'b10: nextstate = S3;

2'b11: nextstate = S2;

endcase

S3: case(ain)

2'b00: nextstate = S0;

2'b01: nextstate = S3;

2'b10: nextstate = S3;

2'b11: nextstate = S2;

endcase

default:

nextstate = S0;

endcase

end

always @(state)

begin

case(state)

S0, S1 : yout = 0;

S2, S3 : yout = 1;

default: yout = 0;

endcase

end

endmodule

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Module Name: moore\_sequence\_detector\_3processes\_tb

//////////////////////////////////////////////////////////////////////////////////

module moore\_sequence\_detector\_3processes\_tb(

);

reg clk;

reg reset;

reg [1:0] ain;

wire yout;

moore\_sequence\_detector\_3processes DUT (.clk(clk), .reset(reset), .ain(ain), .yout(yout));

initial

begin

clk = 0;

forever

begin

#5 clk = 1;

#5 clk = 0;

end

end

initial

begin

reset = 1'b1;

ain = 2'b00;

#20; // wait for 2 clock cycles

reset = 1'b0;

#10 ain = 2'b00;

#10 ain = 2'b11;

#10 ain = 2'b10;

#10 ain = 2'b00; // toggle output to 1

#20 ain = 2'b10;

#10 ain = 2'b00; // toggle output to 0

#10 ain = 2'b11;

#10 ain = 2'b00; // set output to 1

#10 ain = 2'b01;

#10 ain = 2'b00; // reset output to 0

#10 ain = 2'b10;

#10 ain = 2'b11;

#10 ain = 2'b00; // toggle output to 1

#10 reset = 1'b1;

#10 reset = 1'b0;

#10 ain = 2'b10;

#30 ain = 2'b00;

#30;

end

endmodule