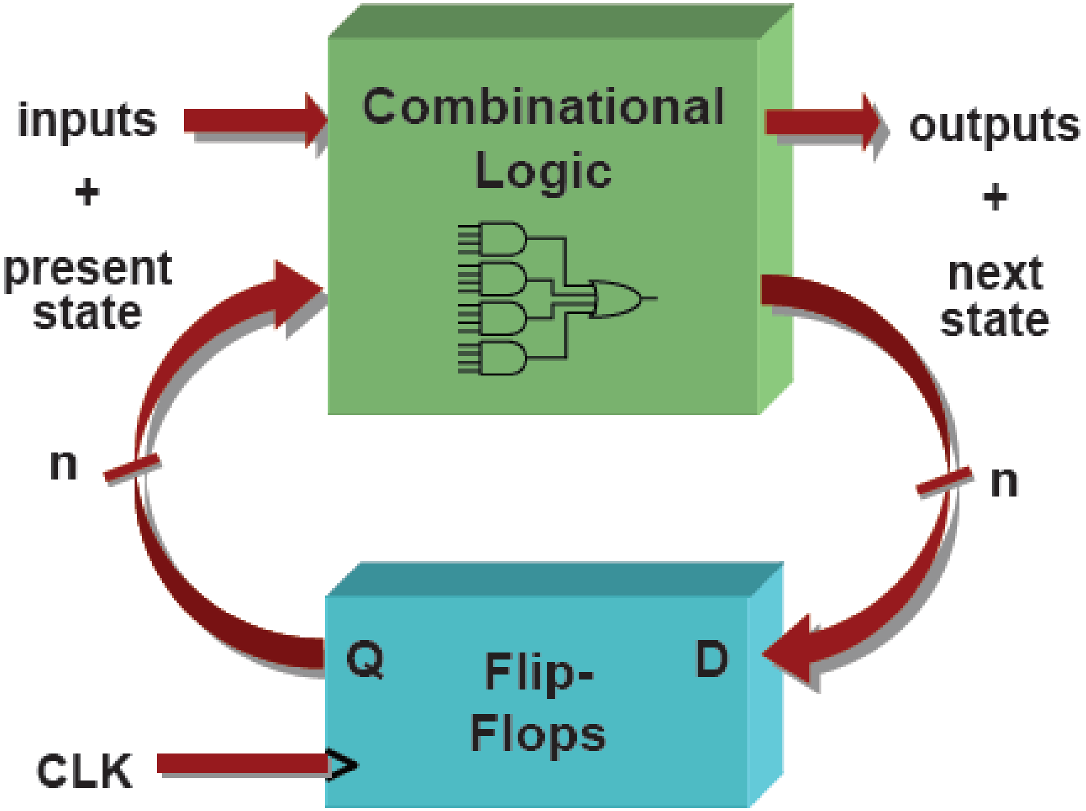
EDA讲座2

主要内容

- ✓ 时序电路的FSM模型
- ✓ 时序always block
- ✓ 时序电路模块的HDL描述
- ✓ FSM的HDL描述
- ✓ EDA实验二内容

时序电路的FSM模型

- Finite State Machines (FSMs) are a useful abstraction for sequential circuits with centralized “states” of operation
- At each clock edge, combinational logic computes *outputs* and *next state* as a function of *inputs* and *present state*

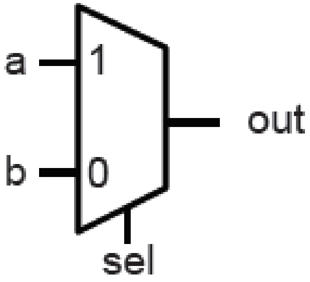


The Sequential always Block

- Edge-triggered circuits are described using a sequential always block

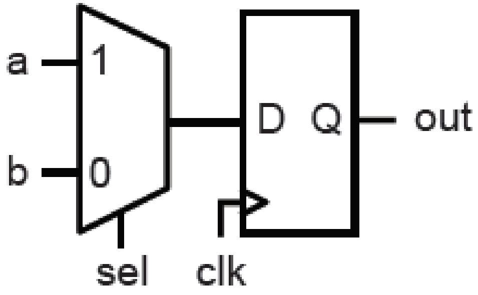
Combinational

```
module combinational(a, b, sel,
                          out);
  input a, b;
  input sel;
  output out;
  reg out;
  always @ (a or b or sel)
  begin
    if (sel) out = a;
    else out = b;
  end
endmodule
```



Sequential

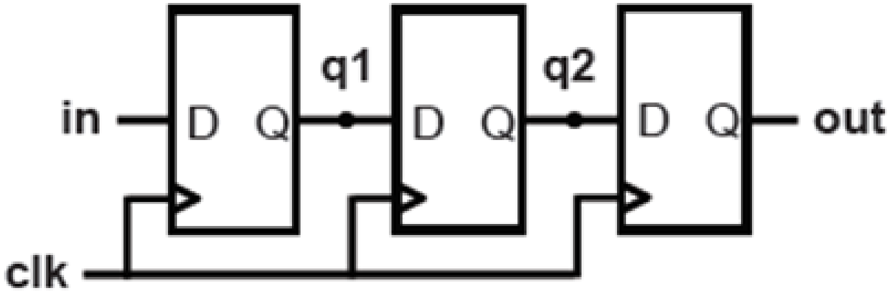
```
module sequential(a, b, sel,
                  clk, out);
  input a, b;
  input sel, clk;
  output out;
  reg out;
  always @ (posedge clk)
  begin
    if (sel) out <= a;
    else out <= b;
  end
endmodule
```



非阻塞式赋值和阻塞式赋值

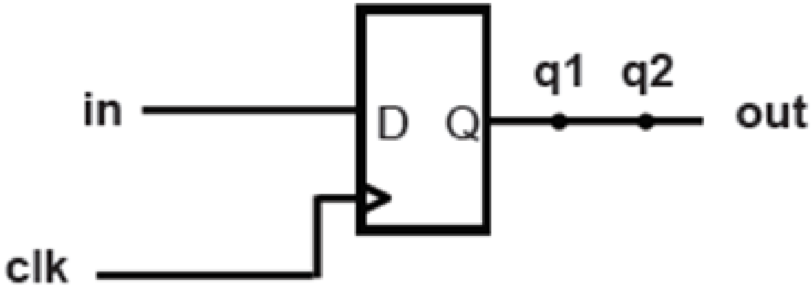
```
always @ (posedge clk)
begin
  q1 <= in;
  q2 <= q1;
  out <= q2;
end
```

“At each rising clock edge, *q1*, *q2*, and *out* simultaneously receive the old values of *in*, *q1*, and *q2*.”



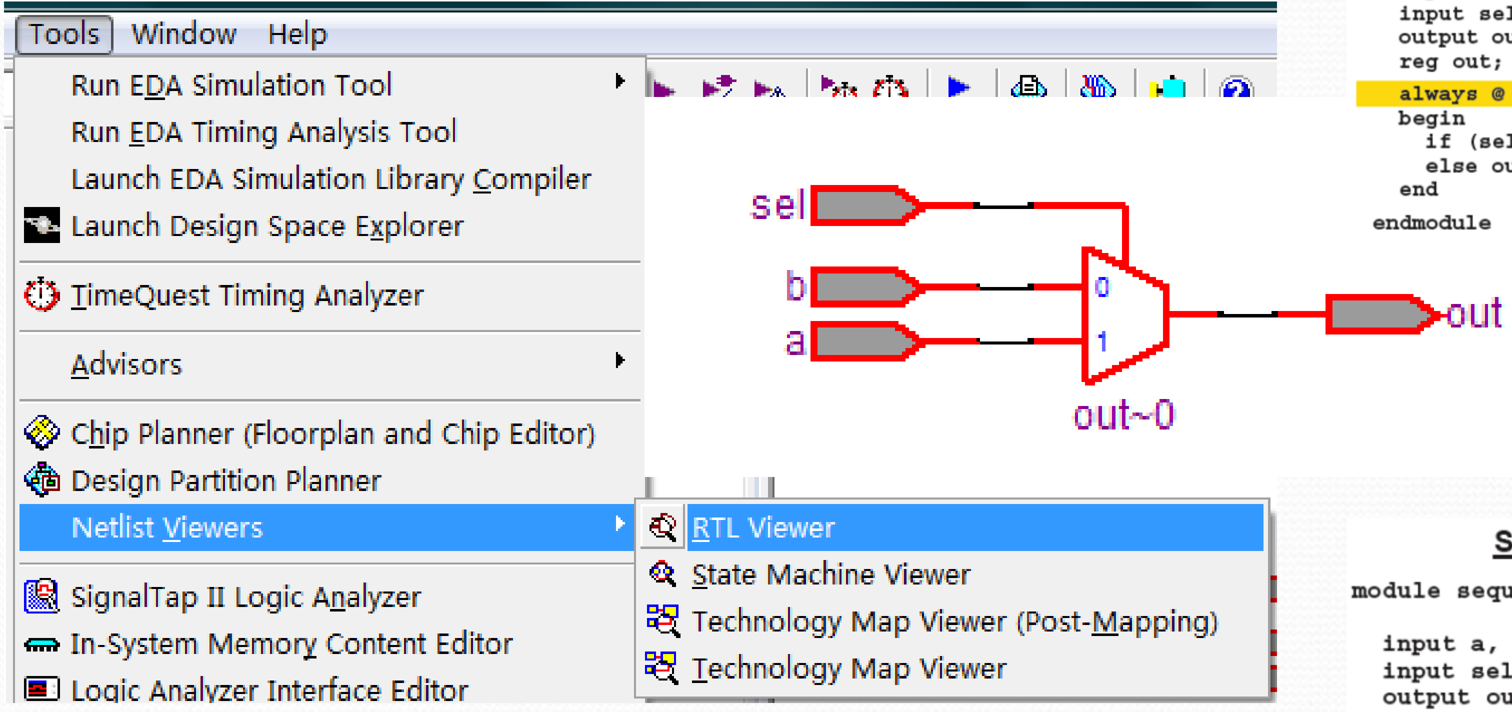
```
always @ (posedge clk)
begin
  q1 = in;
  q2 = q1;
  out = q2;
end
```

“At each rising clock edge, $q1 = in$.
After that, $q2 = q1 = in$.
After that, $out = q2 = q1 = in$.
Therefore $out = in$.”



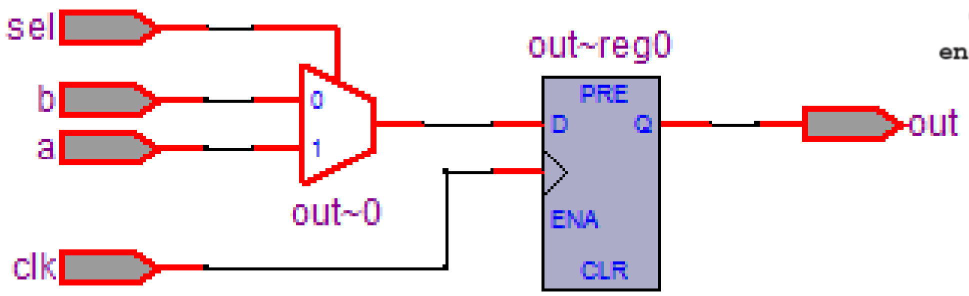
Combinational

```
module combinational(a, b, sel,
                      out);
  input a, b;
  input sel;
  output out;
  reg out;
  always @ (a or b or sel)
  begin
    if (sel) out = a;
    else out = b;
  end
endmodule
```

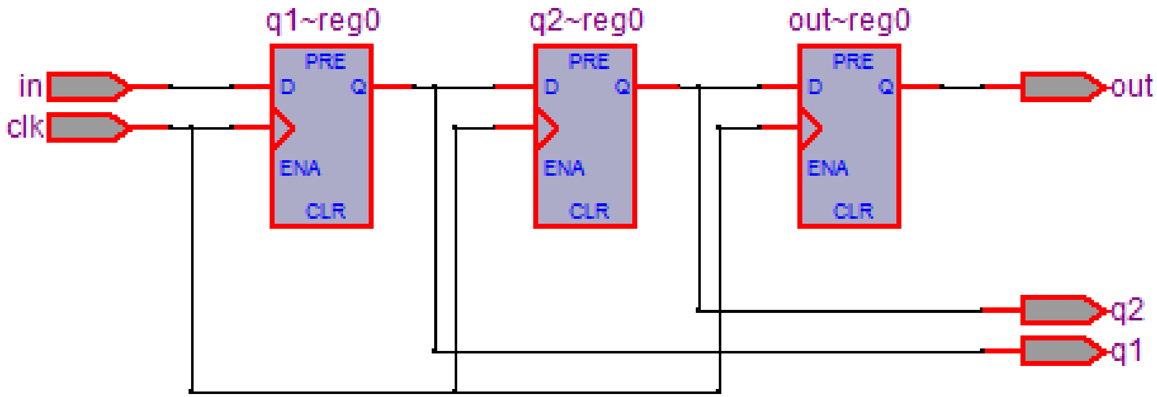

Tools
Window
Help
Run EDA Simulation Tool
Run EDA Timing Analysis Tool
Launch EDA Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner (Floorplan and Chip Editor)
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
RTL Viewer
State Machine Viewer
Technology Map Viewer (Post-Mapping)
Technology Map Viewer
sel
b
a
0
1
out
out~0

Sequential

```
module sequential(a, b, sel,
                   clk, out);
  input a, b;
  input sel, clk;
  output out;
  reg out;
  always @ (posedge clk)
  begin
    if (sel) out <= a;
    else out <= b;
  end
endmodule
```

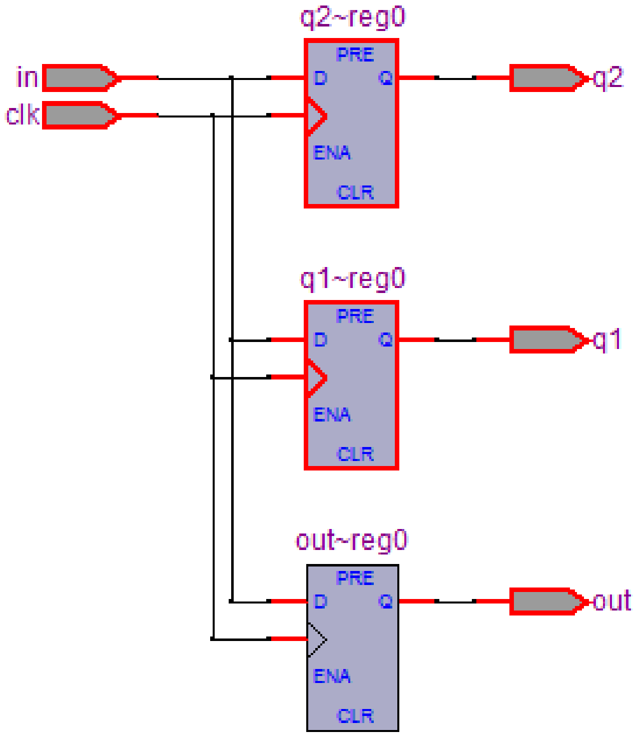


非阻塞式赋值



```
always @ (posedge clk)
begin
   q1 <= in;
   q2 <= q1;
   out <= q2;
end
```

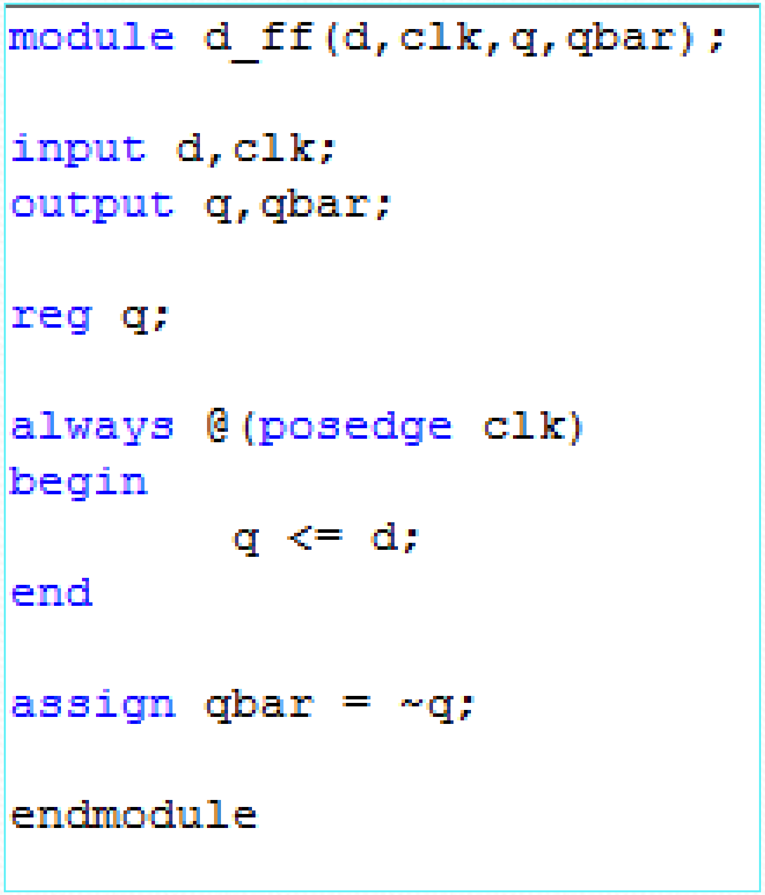
```
always @ (posedge clk)
begin
   q1 = in;
   q2 = q1;
   out = q2;
end
```



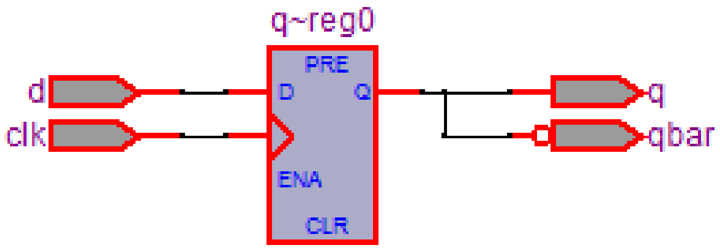
阻塞式赋值

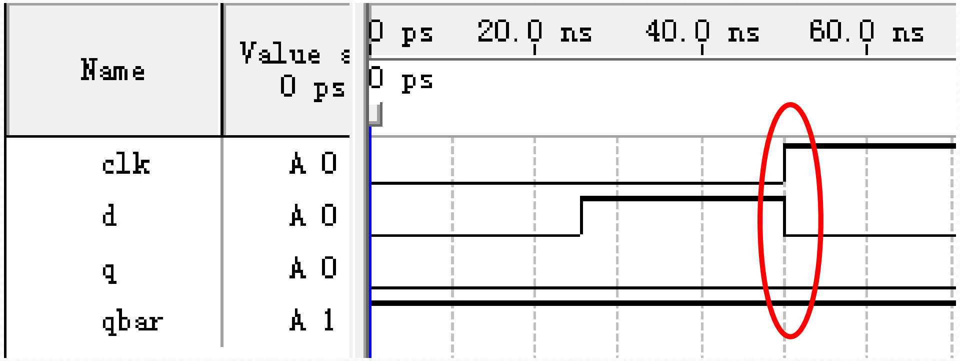
时序电路模块的HDL描述

D触发器

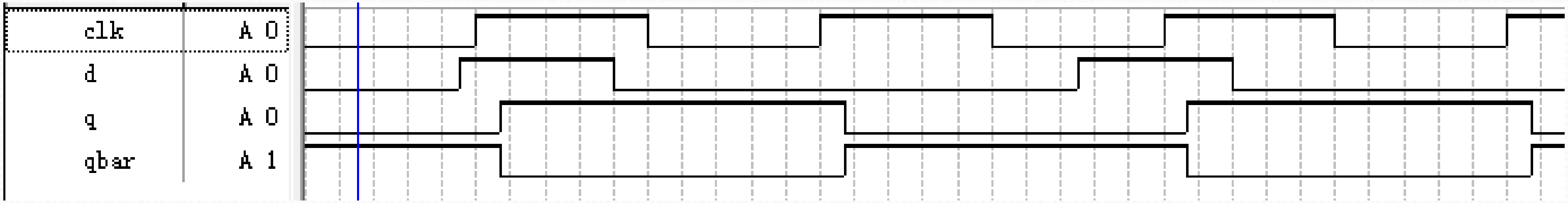


```
module d_ff(d,clk,q,qbar);

input d,clk;
output q,qbar;

reg q;

always @(posedge clk)
begin
        q <= d;
end

assign qbar = ~q;

endmodule
```



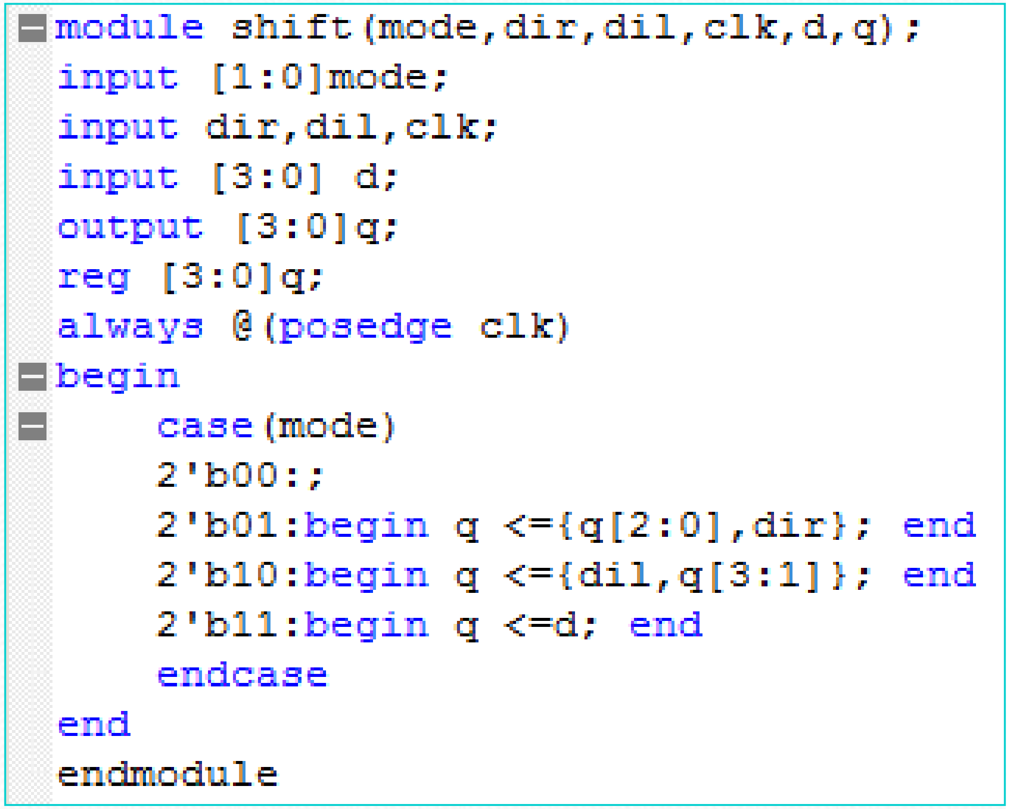


```
Name      Value a
          0 ps      0 ps
clk       A 0
d         A 0
q         A 0
qbar      A 1
```

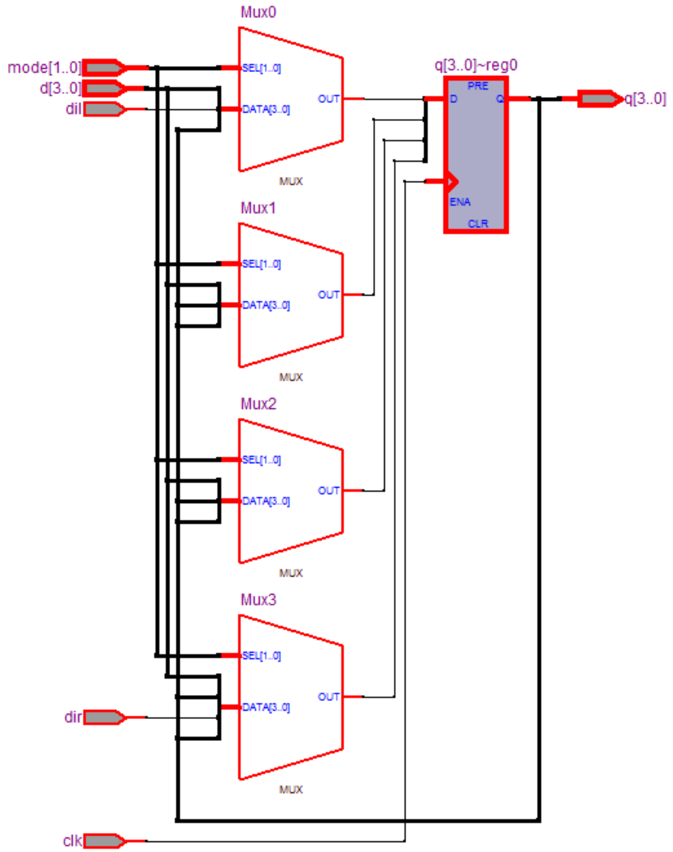


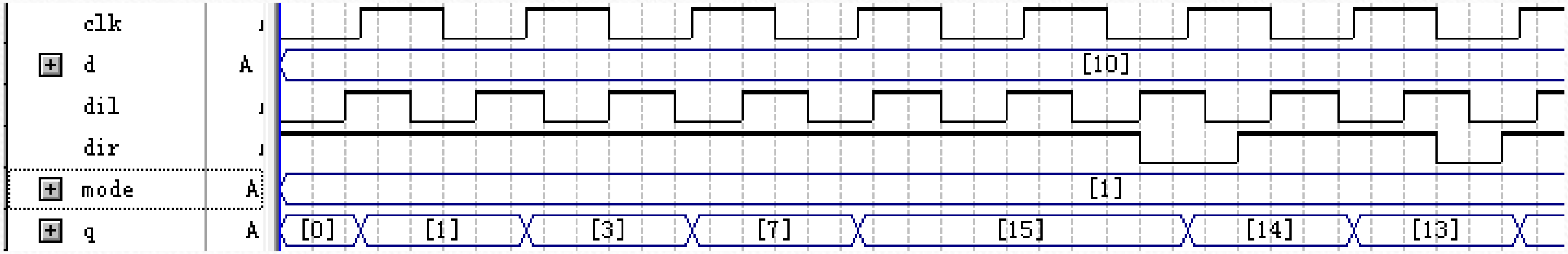
```
clk       A 0
d         A 0
q         A 0
qbar      A 1
```

移位寄存器



```
module shift(mode,dir,dil,clk,d,q);
 input [1:0]mode;
 input dir,dil,clk;
 input [3:0] d;
 output [3:0]q;
 reg [3:0]q;
 always @(posedge clk)
begin
     case(mode)
     2'b00:;
     2'b01:begin q <={q[2:0],dir}; end
     2'b10:begin q <={dil,q[3:1]}; end
     2'b11:begin q <=d; end
     endcase
 end
 endmodule
```



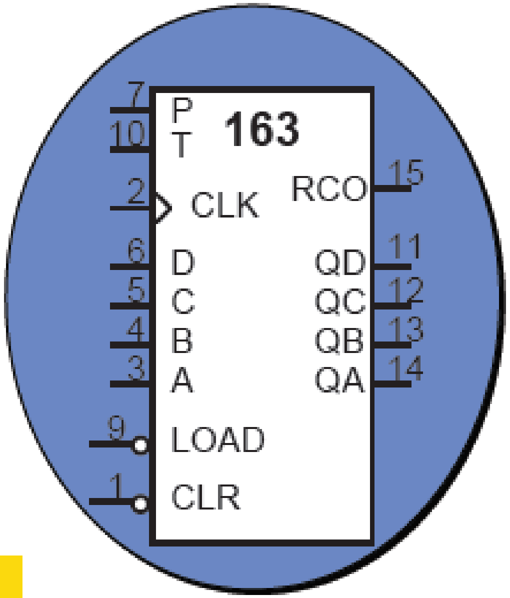


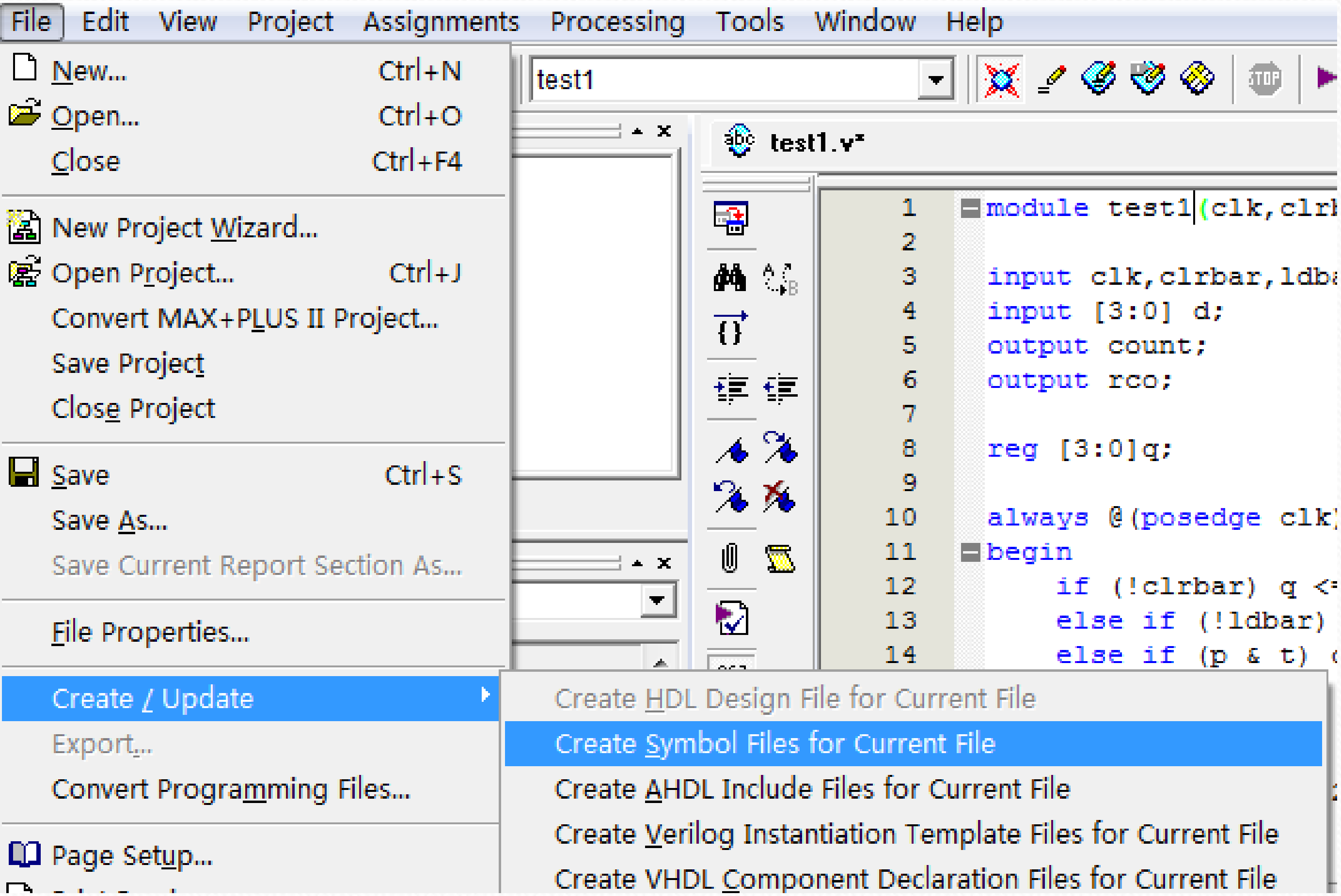
计数器（74163）

```
module counter(LDbar, CLRbar, P, T, CLK, D,
               count, RCO);
  input LDbar, CLRbar, P, T, CLK;
  input [3:0] D;
  output [3:0] count;
  output RCO;
  reg [3:0] Q;

  always @ (posedge CLK) begin
    if (!CLRbar) Q <= 4'b0000;
    else if (!LDbar) Q <= D;
    else if (P && T) Q <= Q + 1;
  end

  assign count = Q;
  assign RCO = Q[3] & Q[2] & Q[1] & Q[0] & T;
endmodule
```

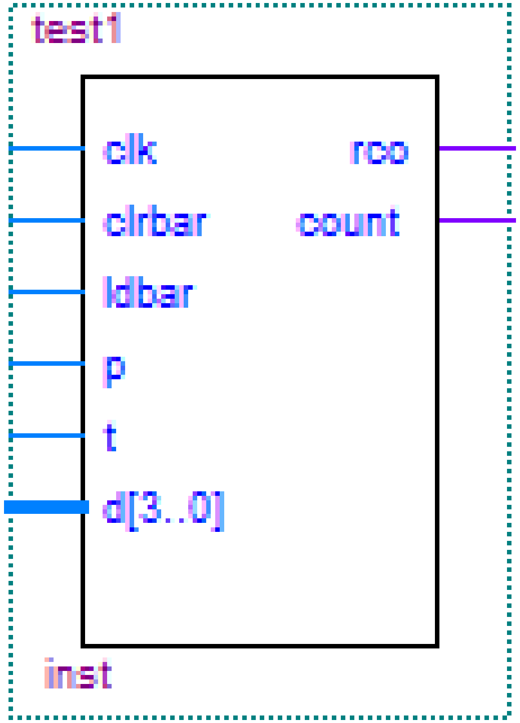
priority logic for control signals

RCO gated by T input



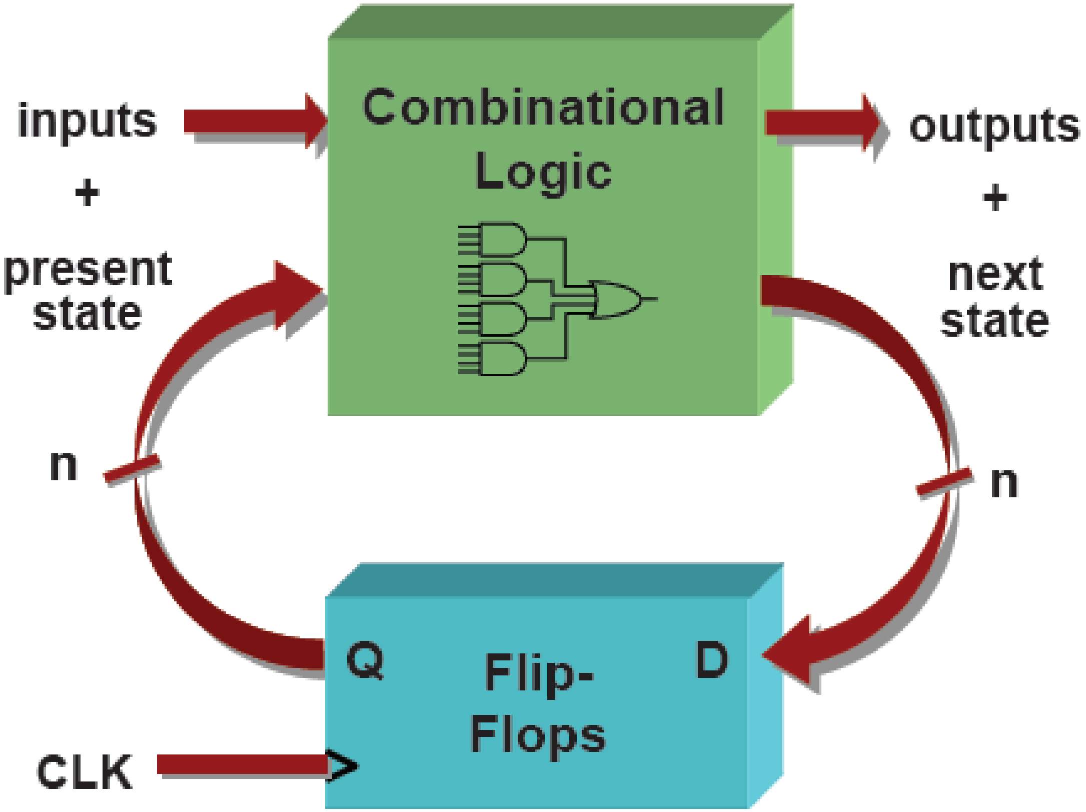

File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Convert MAX+PLUS II Project...
Save Project
Close Project
Save Ctrl+S
Save As...
Save Current Report Section As...
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Create HDL Design File for Current File
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
test1
test1.v*

```
module test1(clk,clrb

input clk,clrbar,ldba
input [3:0] d;
output count;
output rco;

reg [3:0]q;

always @(posedge clk)
begin
    if (!clrbar) q <=
    else if (!ldbar)
    else if (p & t) 
```

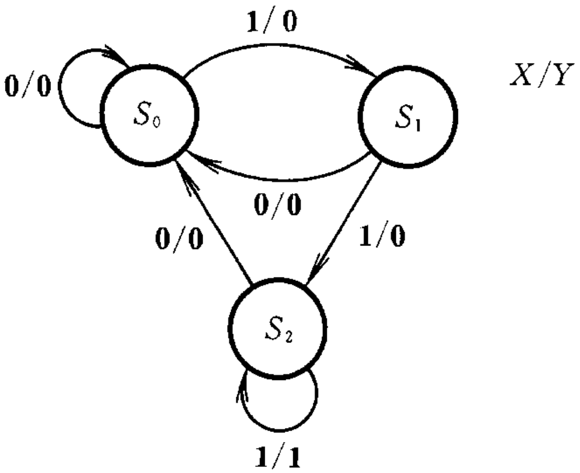


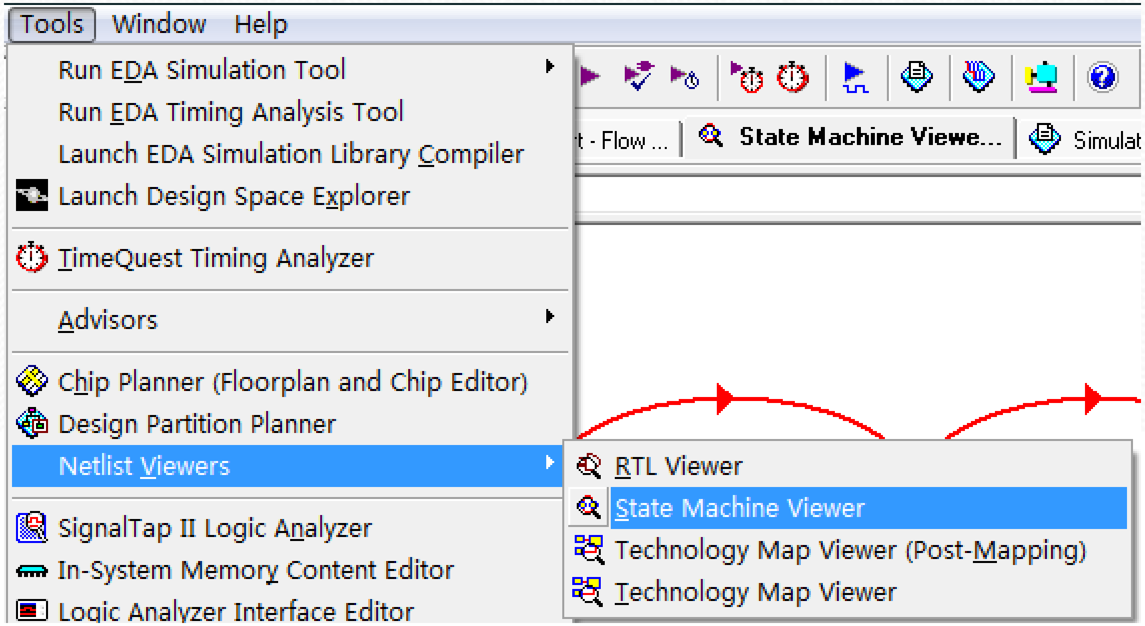
FSM的HDL描述

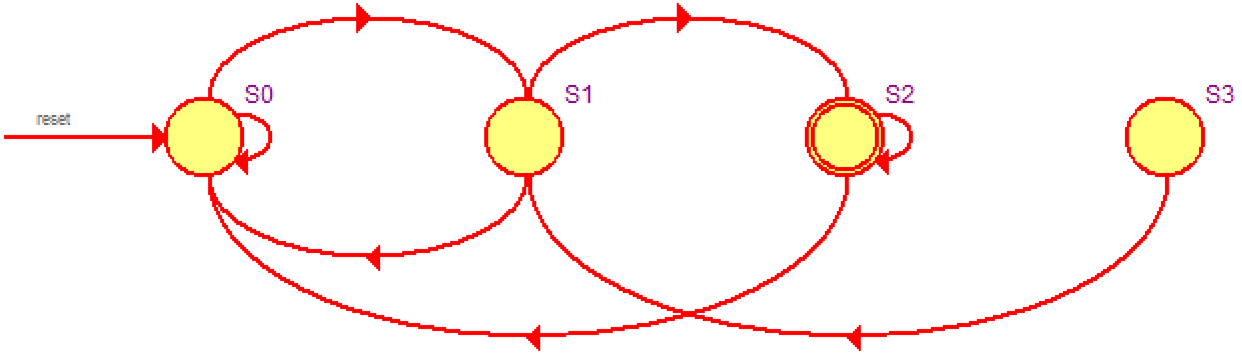
- Finite State Machines (FSMs) are a useful abstraction for sequential circuits with centralized “states” of operation
- At each clock edge, combinational logic computes *outputs* and *next state* as a function of *inputs* and *present state*

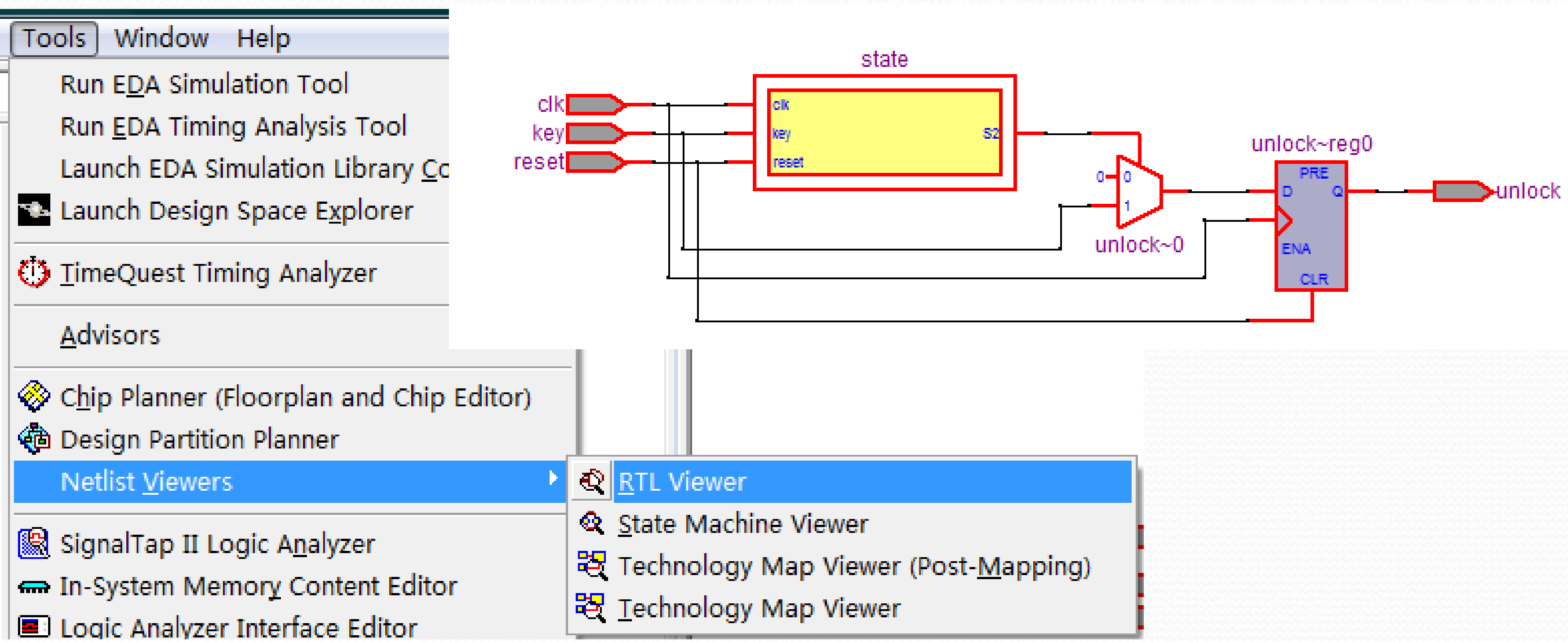


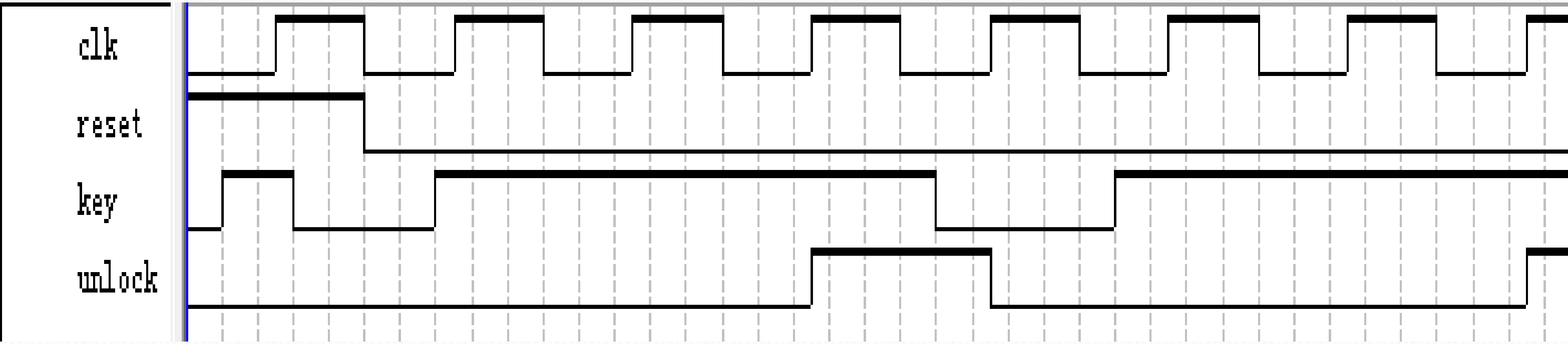
```
module mylock(clk,key,reset,unlock);
    input clk, key, reset;
    output reg unlock;
    reg [1:0]state;

    parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
    always @ (posedge clk or posedge reset) begin
        if (reset)  begin state <= S0; unlock <= 0;end
        else
            case (state)
                S0:
                    if (key) begin state <= S1;unlock <= 0;end
                    else begin state <= S0;unlock <= 0;end
                S1:
                    if (key) begin state <= S2;unlock <= 0;end
                    else begin state <= S0;unlock <= 0;end
                S2:
                    if (key) begin state <= S2;unlock <= 1;end
                    else begin state <= S0;unlock <= 0;end
                S3:
                    if (key) begin state <= S1;unlock <= 0;end
                    else begin state <= S1;unlock <= 0;end
            endcase
    end
endmodule
```

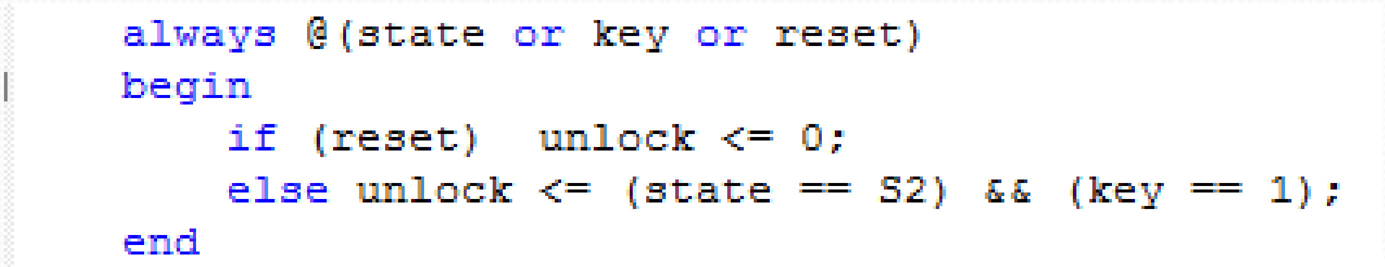



Tools
Window
Help
Run EDA Simulation Tool
Run EDA Timing Analysis Tool
Launch EDA Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner (Floorplan and Chip Editor)
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
RTL Viewer
State Machine Viewer
Technology Map Viewer (Post-Mapping)
Technology Map Viewer
t - Flow ...
State Machine Viewe...
Simulat

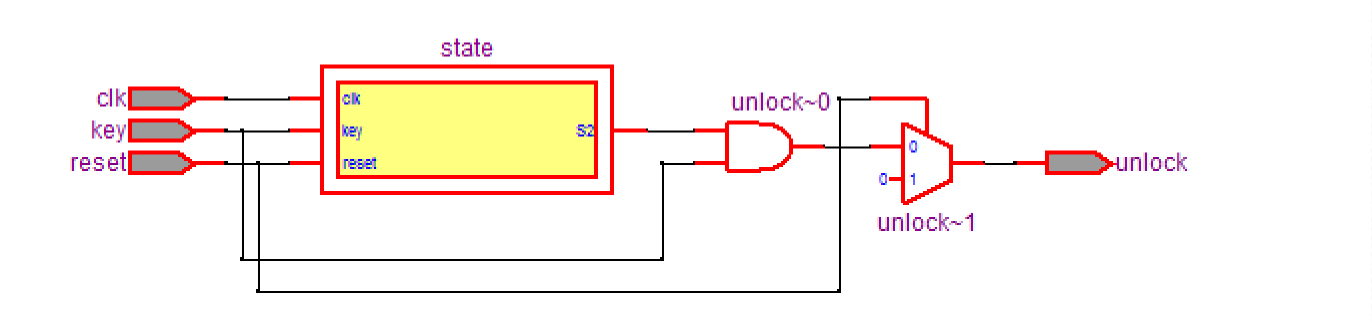

reset
S0
S1
S2
S3

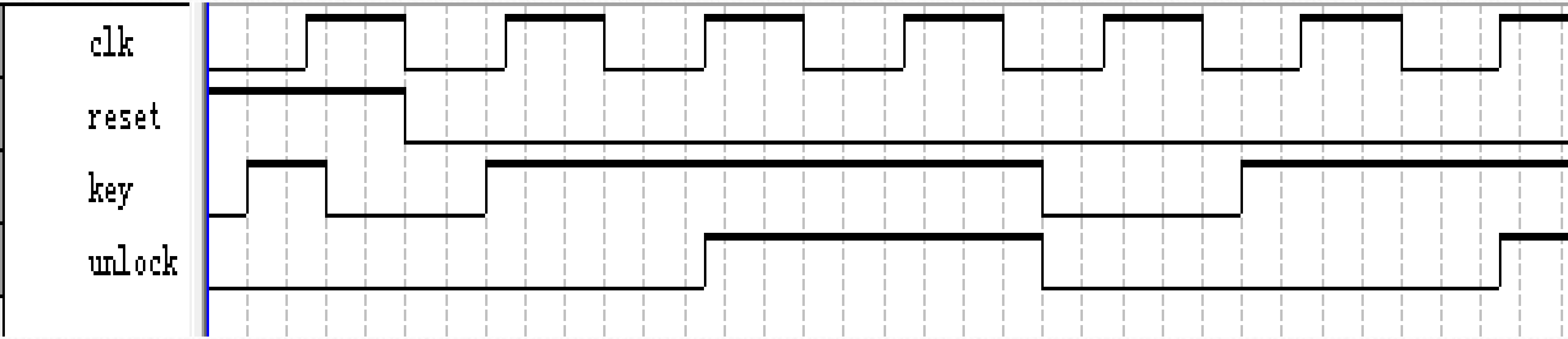

Tools
Window
Help
Run EDA Simulation Tool
Run EDA Timing Analysis Tool
Launch EDA Simulation Library Co
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner (Floorplan and Chip Editor)
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
RTL Viewer
State Machine Viewer
Technology Map Viewer (Post-Mapping)
Technology Map Viewer
state
clk
key
reset
clk
key
reset
S2
0
0
1
unlock~0
unlock~reg0
PRE
D
Q
ENA
CLR
unlock


clk
reset
key
unlock

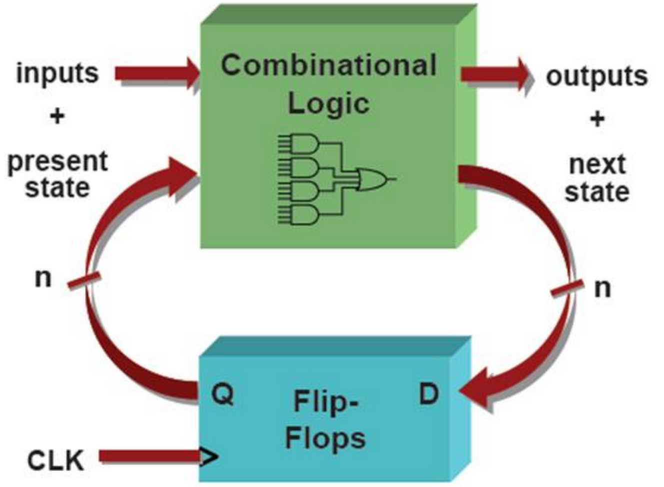


```
always @(state or key or reset)
begin
    if (reset)  unlock <= 0;
    else unlock <= (state == S2) && (key == 1);
end
```





```
always @ (posedge clk)
    begin cur_state <= next_state; end


always @ (cur_state or reset or key) begin
    if (reset) begin next_state <= S0;end
    else
        case (cur_state)
            S0:next_state <= ...;
            S1:...
        endcase
    end

always @ (cur_state or reset or key)
    begin
        if (reset)  unlock <= 0;
        else unlock <= (cur_state == S2) && (key == 1);
    end
```



EDA实验二内容

Vending machine

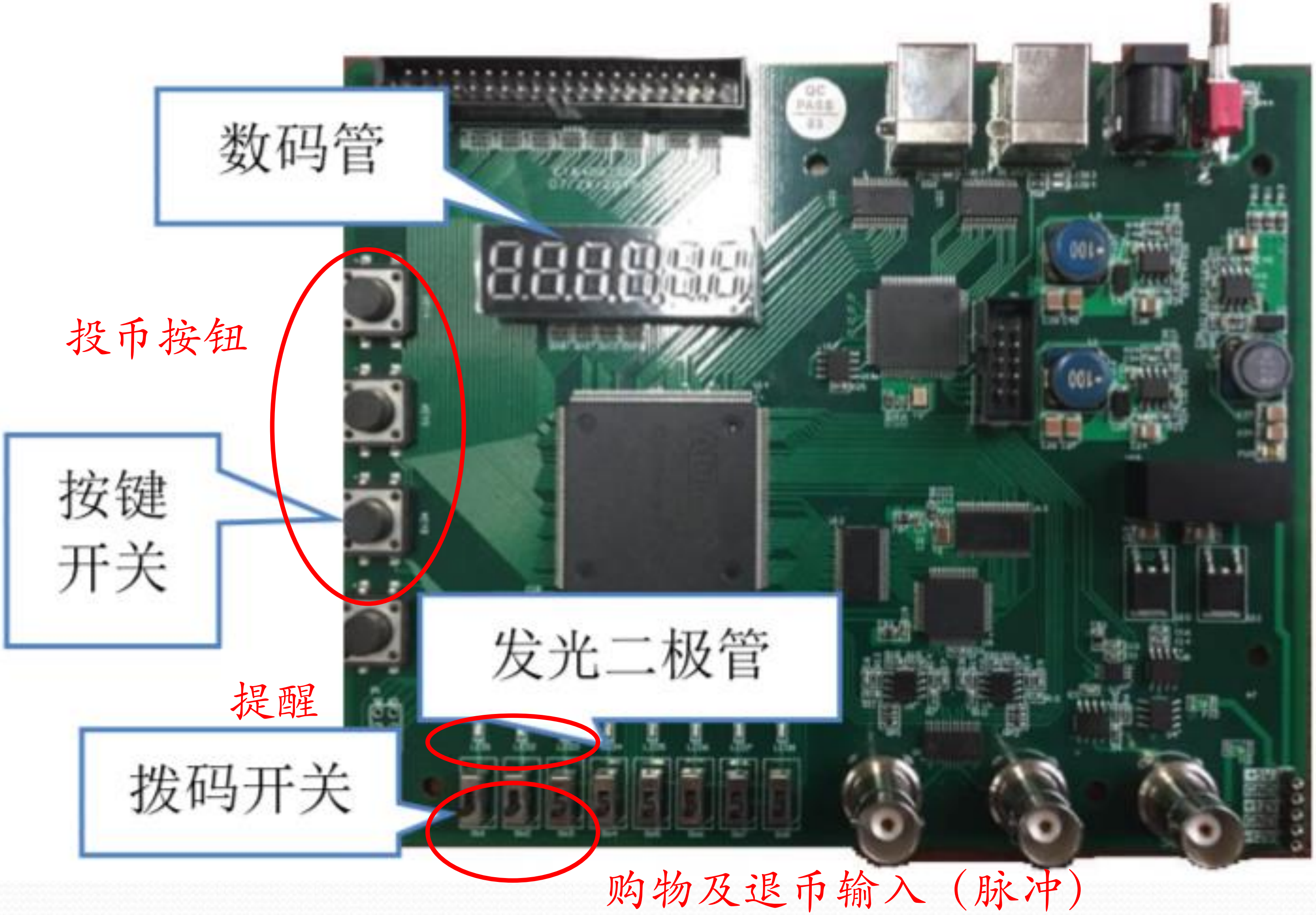
利用实验板上的拨码开关和按键开关模拟投币、购物和退币输入，用发光二极管模拟各种提示信息，用数码管显示余额，实现一个自动售货机内部控制电路。要求满足如下规格:

1）可接受5角、1元和5元的投币，每次购买允许投入多种不同币值的钱币；用3只数码管显示当前投币金额，如055表示已投币5.5元；

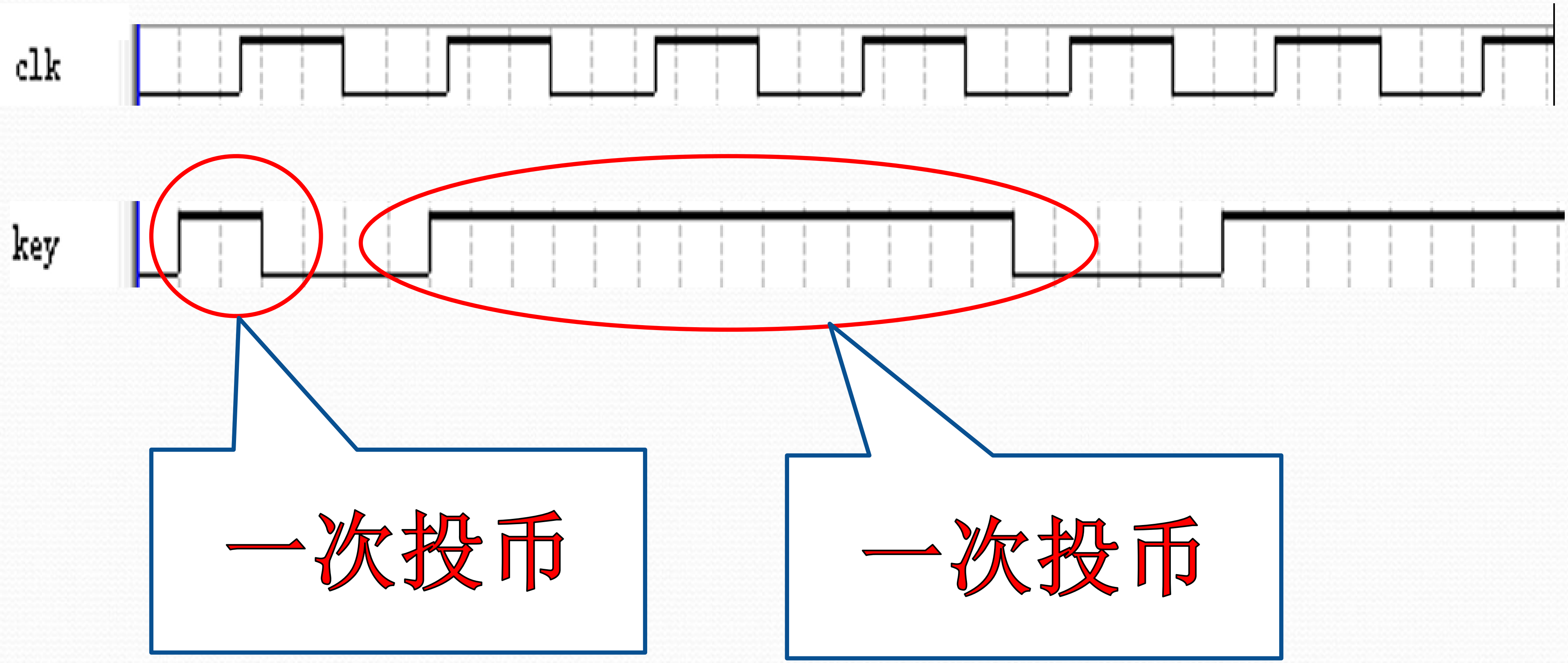
2）可售出价格分别为1.5元和2.5元的商品，假设用户每次购买时只选择单件、一种商品；允许用户多次购买商品，每次购买后，可以进行补充投币；

3）选择购买商品后，如果投币金额不足，则提醒；否则，售出相应的商品，并提醒用户取走商品；

4）若用户选择退币，则退回余下的钱，并提醒用户取钱。



实验板上有40MHz的时钟信号，对应FPGA引脚号为PIN_152，自动售货机的工作时钟及数码管循环扫描显示的时钟可由该40MHz分频得到。



一次投币

一次投币