高階描述語言-有限狀態機(FSM)

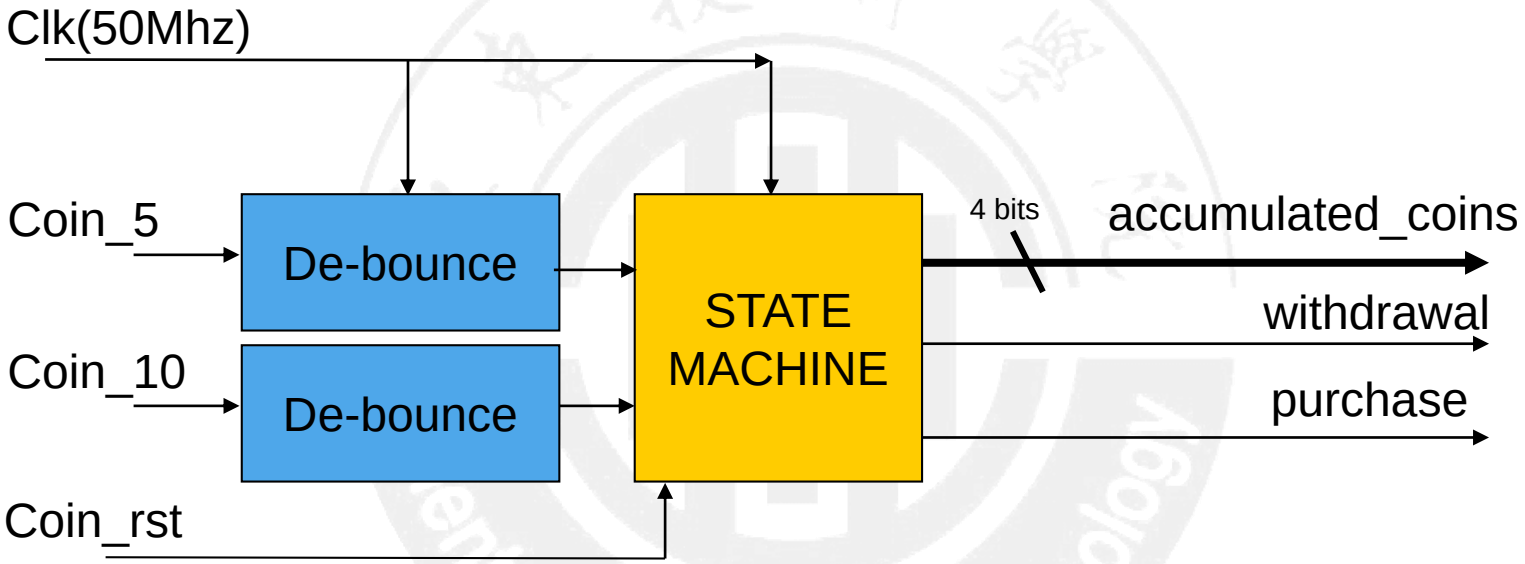
陳韋達

Description of Vending Machine

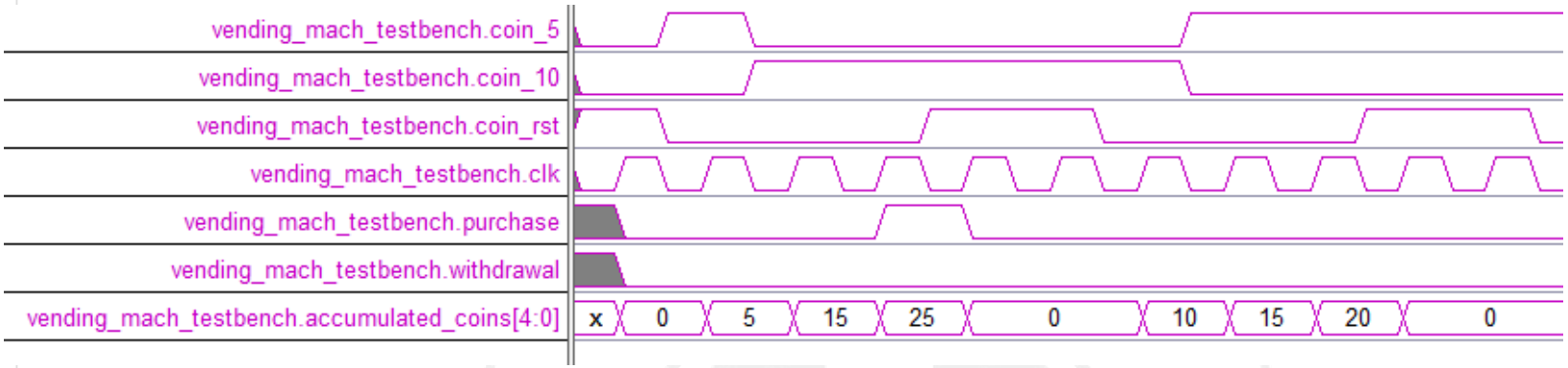
- 一個簡易販賣機，上面有投幣、退幣、購買三個功能。
- 只允許5與10元硬幣投入，而所賣的飲料值25元
- 最多可以投到30元
- 一但購買產品，將自動找5元，並且致能(enable)購買的訊號線，由邏輯"0"轉成"1"
- 當按下退幣，不論投入金額多少，都會將金額設定為"0"元

Block of Vending Machine

Architecture



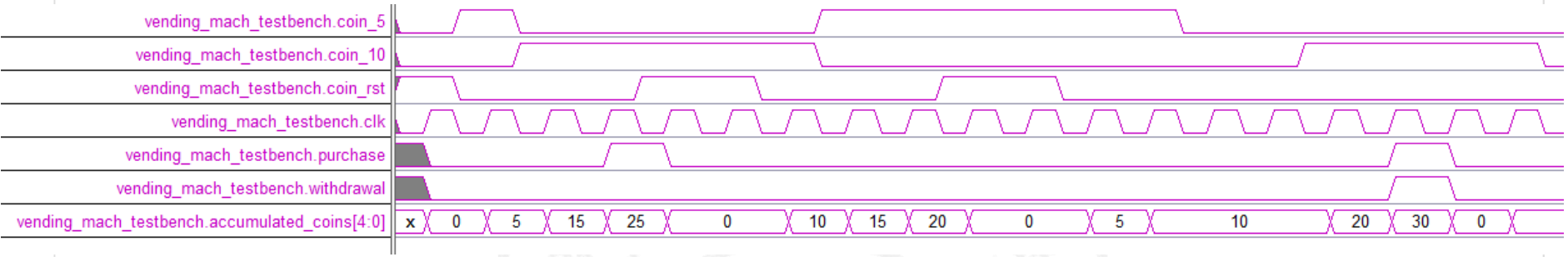
Vending machine(1)




```
module vending_mach(coin_5,coin_10,coin_rst,clk,
                    purchase,withdrawal,accumulated_coins);

input  coin_5,coin_10,coin_rst,clk;
output purchase,withdrawal;
output [4:0]accumulated_coins;
reg       purchase,withdrawal;
reg [4:0] accumulated_coins,accumulated_coins_next;
reg [2:0] current_state,next_state;

always@(posedge clk)
  if(coin_rst==1'b1 | purchase==1'd1)
    begin
     current_state<=3'd0;
     accumulated_coins<=5'd0;
    end
  else
    begin
     current_state    <=next_state;
     accumulated_coins<=accumulated_coins_next+accumulated_coins;
    end
```

```
always@(coin_5 or coin_10 or current_state or accumulated_coins)
   case(current_state)
      3'd0:if(coin_5==1'b1)
             begin
                accumulated_coins_next=5'd5;
                next_state            =3'd1;
                purchase              =1'd0;
                withdrawal            =1'b0;
             end
           else if (coin_10==1'b1)
             begin
                accumulated_coins_next=5'd10;
                next_state            =3'd2;
                purchase              =1'd0;
                withdrawal            =1'b0;
             end
           else
             begin
                next_state            =3'd0;
                accumulated_coins_next=5'd0;
                purchase              =1'd0;
                withdrawal            =1'b0;
             end
```

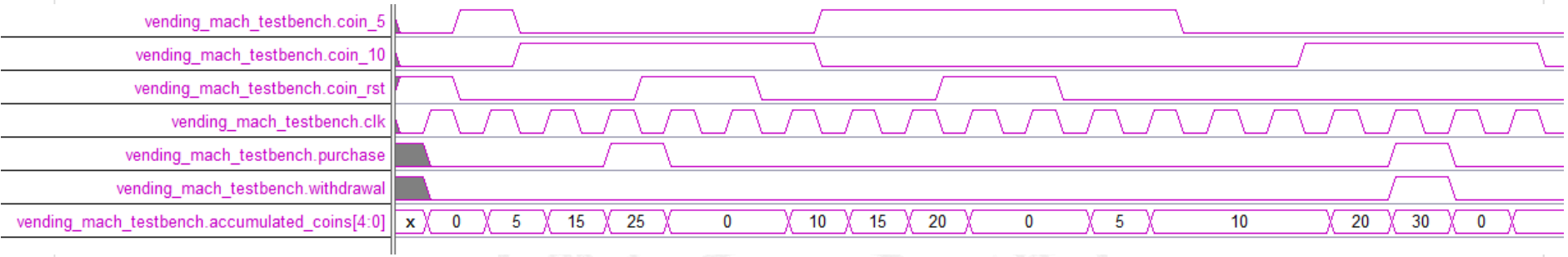
Vending machine(2)




```
3'd1:if(coin_5==1'b1)
        begin
            accumulated_coins_next=5'd5;
            next_state            =3'd2;
            purchase              =1'd0;
            withdrawal            =1'b0;
        end
    else if (coin_10==1'b1)
        begin
            accumulated_coins_next=5'd10;
            next_state            =3'd3;
            purchase              =1'd0;
            withdrawal            =1'b0;
        end
    else
        begin
            next_state            =3'd1;
            accumulated_coins_next=5'd0;
            purchase              =1'd0;
            withdrawal            =1'b0;
        end
```

```
3'd2:if(coin_5==1'b1)
        begin
          accumulated_coins_next=5'd5;
          next_state            =3'd3;
          purchase              =1'd0;
          withdrawal            =1'b0;
        end
    else if (coin_10==1'b1)
        begin
          accumulated_coins_next=5'd10;
          next_state            =3'd4;
          purchase              =1'd0;
          withdrawal            =1'b0;
        end
    else
        begin
          next_state            =3'd2;
          accumulated_coins_next=5'd0;
          purchase              =1'd0;
          withdrawal            =1'b0;
        end
```

Vending machine(3)



1. 須完成剩下 state 3~state5 的部分，
2. 撰寫測試檔並觀察FSM的波形是否正確。

```
.....
                3'd6:if(accumulated_coins==5'd30)
                        begin
                            accumulated_coins_next=5'd0;
                            purchase               =1'd1;
                            next_state             =3'd0;
                            withdrawal             =1'b1;
                        end
                    else
                        begin
                            next_state             =3'd6;
                            accumulated_coins_next=5'd0;
                            purchase               =1'd0;
                            withdrawal             =1'b0;
                        end
                default:begin end
            endcase

        endmodule
```

Test bench

```
module vending_mach_testbench;

reg       coin_5,coin_10,coin_rst,clk;
wire      purchase,withdrawal;
wire [4:0]accumulated_coins;

vending_mach U1(
                .coin_5(coin_5),
                .coin_10(coin_10),
                .coin_rst(coin_rst),
                .clk(clk),
                .purchase(purchase),
                .withdrawal(withdrawal),
                .accumulated_coins(accumulated_coins)
                );

initial
  begin
    clk=1'b0;
    coin_5=1'b0;
    coin_10=1'b0;
    coin_rst=1'b1;
  end

initial
  forever #10 clk=~clk;
```

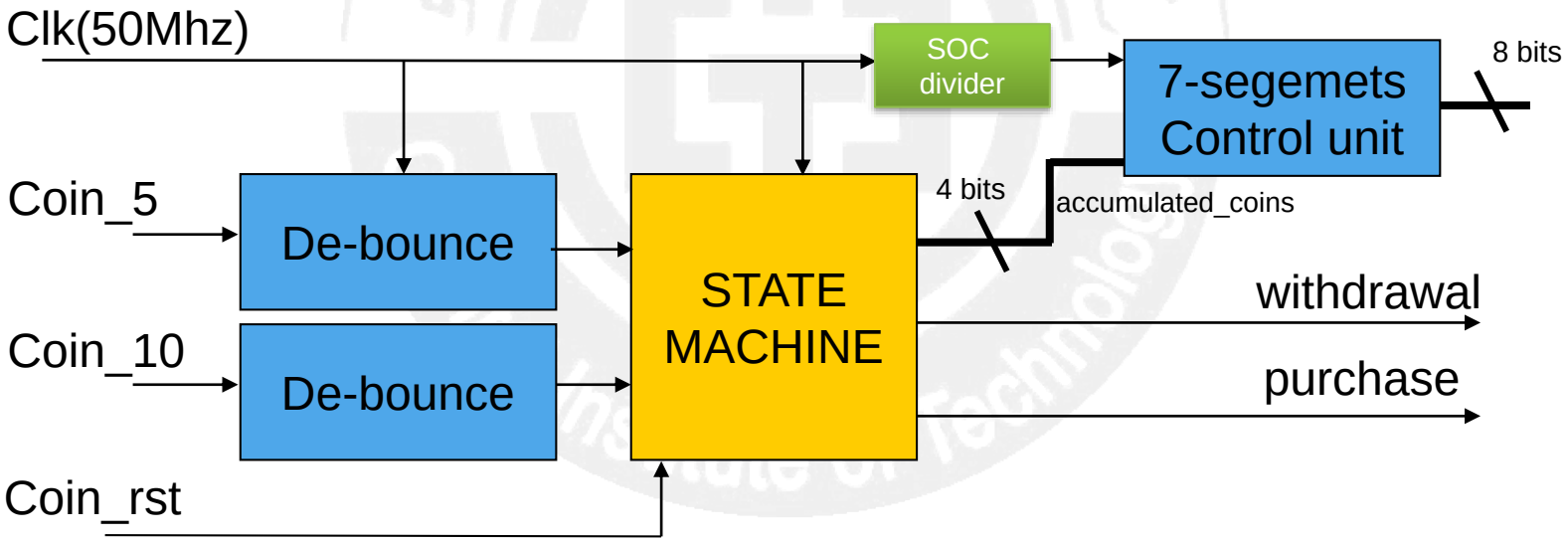
```
initial
  begin

    #  20 coin_5=1'b1;coin_10=1'b0;coin_rst=1'b0;
    #  20 coin_5=1'b0;coin_10=1'b1;
    #  20 coin_5=1'b0;coin_10=1'b1;
    #  20 coin_rst=1'b1;
```

- 須完成剩下輸入資料的部分
- 時間可以自行修改
- 最後再加上$finish;

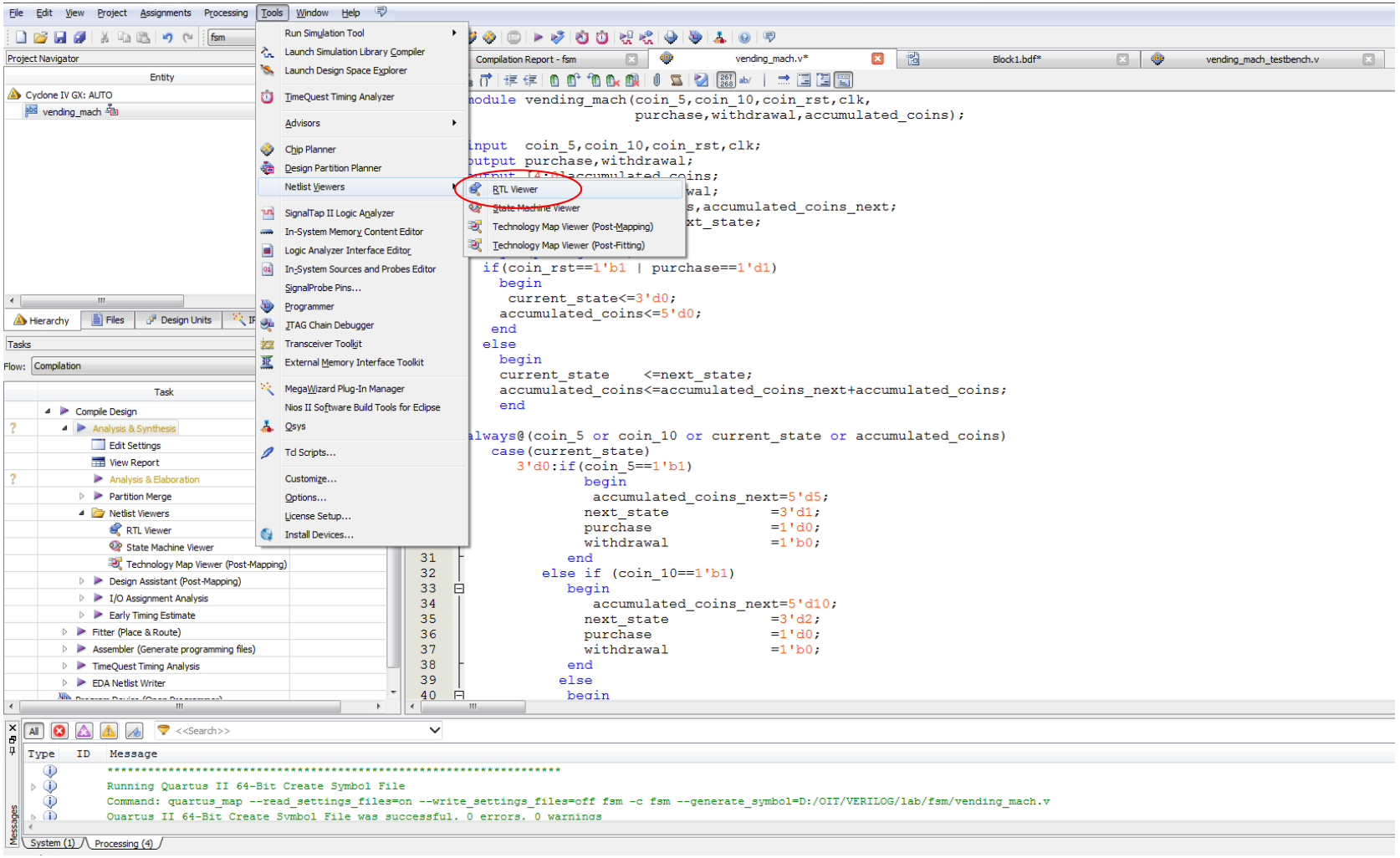
```
    # 300 $finish;
  end
endmodule
```

H.W.

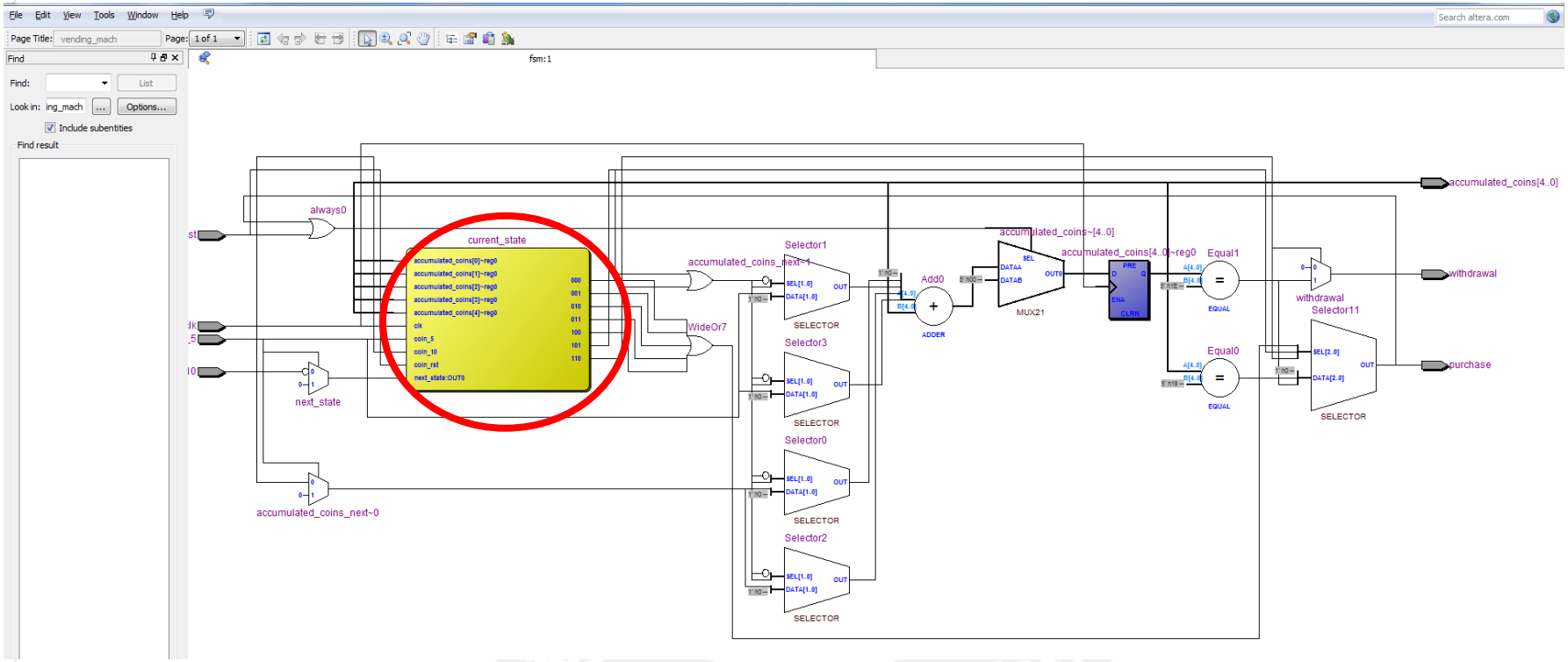
- 設計、
  - 1.完成狀態機state 2~state5的部分
  - 2.撰寫test bench
  - 3.withdrawal與purchase輸出至LED
  - 4.accumulated_coins的金額顯示至七段顯示器上



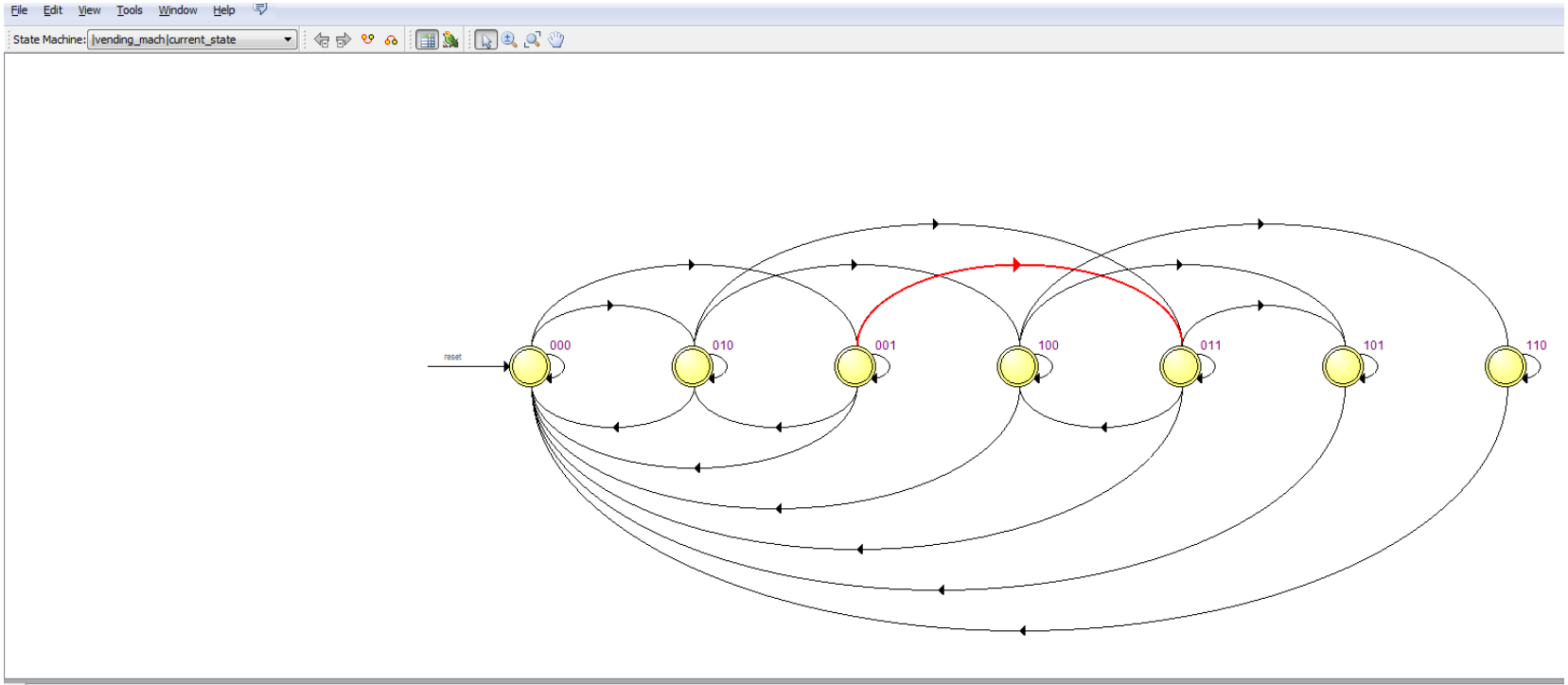
RTL view(1)


File Edit View Project Assignments Processing Tools Window Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
RTL Viewer
State Machine Viewer
Technology Map Viewer (Post-Mapping)
Technology Map Viewer (Post-Fitting)
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
Transceiver Toolkit
External Memory Interface Toolkit
MegaWizard Plug-In Manager
Nios II Software Build Tools for Eclipse
Qsys
Tcl Scripts...
Customize...
Options...
License Setup...
Install Devices...
Project Navigator
Entity
Cyclone IV GX: AUTO
vending_mach
Hierarchy
Files
Design Units
Tasks
Flow: Compilation
Task
Compile Design
Analysis & Synthesis
Edit Settings
View Report
Analysis & Elaboration
Partition Merge
Netlist Viewers
RTL Viewer
State Machine Viewer
Technology Map Viewer (Post-Mapping)
Design Assistant (Post-Mapping)
I/O Assignment Analysis
Early Timing Estimate
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Compilation Report - fsm
vending_mach.v*
Block1.bdf*
vending_mach_testbench.v
module vending_mach(coin_5,coin_10,coin_rst,clk,
purchase,withdrawal,accumulated_coins);
input coin_5,coin_10,coin_rst,clk;
output purchase,withdrawal;
accumulated_coins_next;
if(coin_rst==1'b1 | purchase==1'd1)
begin
current_state<=3'd0;
accumulated_coins<=5'd0;
end
else
begin
current_state <=next_state;
accumulated_coins<=accumulated_coins_next+accumulated_coins;
end
always@(coin_5 or coin_10 or current_state or accumulated_coins)
case(current_state)
3'd0:if(coin_5==1'b1)
begin
accumulated_coins_next=5'd5;
next_state =3'd1;
purchase =1'd0;
withdrawal =1'b0;
end
else if (coin_10==1'b1)
begin
accumulated_coins_next=5'd10;
next_state =3'd2;
purchase =1'd0;
withdrawal =1'b0;
end
else
begin
Type ID Message
Running Quartus II 64-Bit Create Symbol File
Command: quartus_map --read_settings_files=on --write_settings_files=off fsm -c fsm --generate_symbol=D:/OIT/VERILOG/lab/fsm/vending_mach.v
Quartus II 64-Bit Create Symbol File was successful. 0 errors, 0 warnings
System (1)
Processing (4)

RTL view(2)


File Edit View Tools Window Help
Page Title: vending_mach
Page: 1 of 1
Find
Find:
List
Look in: ing_mach
Options...
Include subentities
Find result
fsm:1
always0
current_state
accumulated_coins[0]~reg0
accumulated_coins[1]~reg0
accumulated_coins[2]~reg0
accumulated_coins[3]~reg0
accumulated_coins[4]~reg0
clk
coin_5
coin_10
coin_rst
next_state:OUT0
000
001
010
011
100
101
110
next_state
accumulated_coins_next~0
accumulated_coins_next~4
WideOr7
Selector1
Selector3
Selector0
Selector2
SELECTOR
Add0
ADDER
accumulated_coins~[4..0]
MUX21
accumulated_coins[4..0]~reg0
Equal1
Equal0
EQUAL
withdrawal
Selector11
accumulated_coins[4..0]
withdrawal
purchase

RTL view(3)



	Source State	Destination State	Condition
1	000	010	(!coin_5).(coin_10).(!coin_rst)
2	000	001	(coin_5).(!coin_rst)
3	000	000	(!next_state).(coin_rst) + (next_state)
4	001	011	(!coin_5).(coin_10).(!coin_rst)
5	001	010	(coin_5).(!coin_rst)
6	001	001	(next_state).(!coin_rst)
7	001	000	(coin_rst)
8	010	100	(!coin_5).(coin_10).(!coin_rst)
9	010	011	(coin_5).(!coin_rst)
10	010	010	(next_state).(!coin_rst)
11	010	000	(coin_rst)
12	011	101	(!coin_5).(coin_10).(!coin_rst)
13	011	100	(coin_5).(!coin_rst)

State Table

Transitions Encoding

點選condition下的選項，所對應FSM的狀態，會以紅色顯示，如上圖