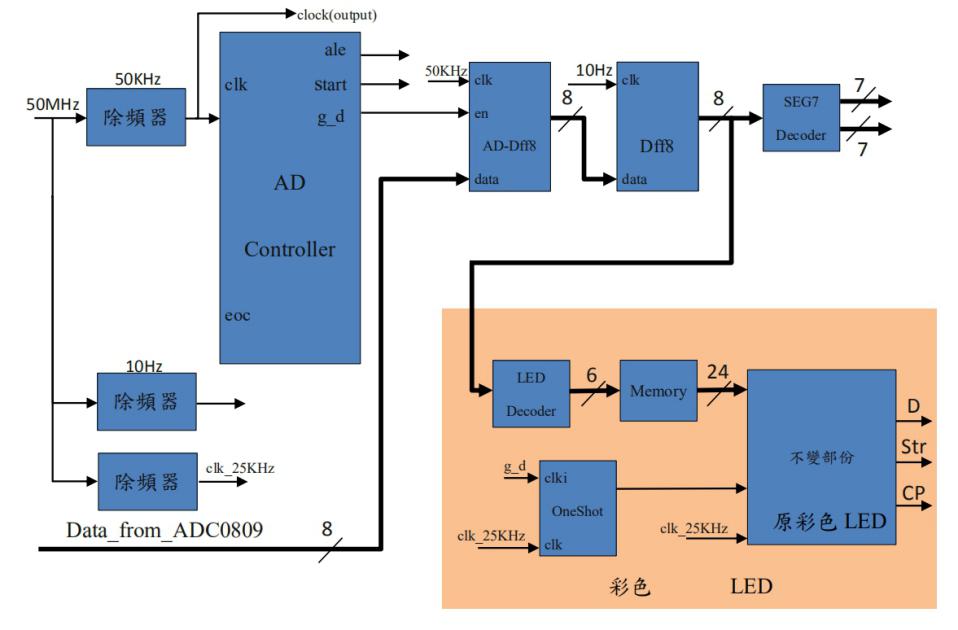
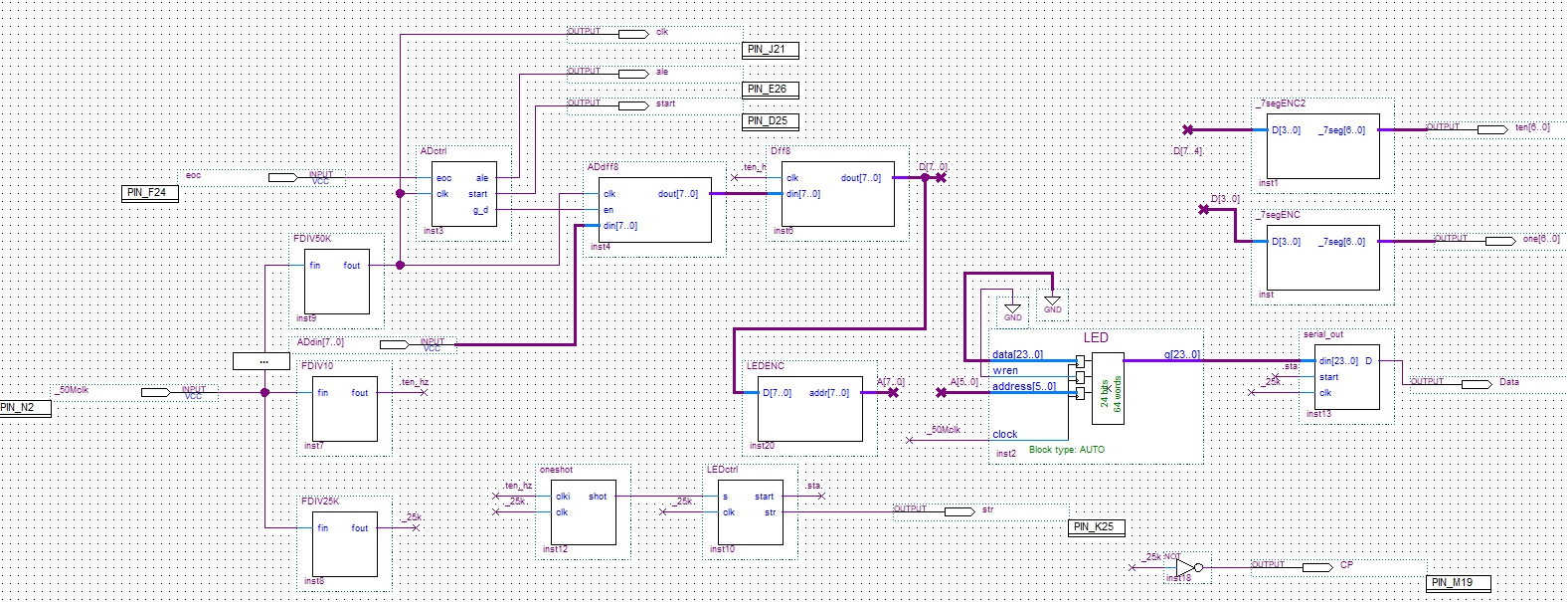
**數位實驗6：類比轉數位訊號控制電路**

**第十四組**

**許銘森B063011058**

**整體架構圖**

****

****

**各模組的Verilog code和波形模擬**

**除頻器 50K：**

module FDIV50K(fin,fout);

input fin;

output fout;

wire [31:0] \_DIVN,DIVN;

reg [31:0] count;

reg fout;

assign DIVN=32'd1000; // DIVN為被除數，50M/1000=50k

assign \_DIVN = DIVN >>1;

always @(posedge fin)

if(count>=DIVN-32'd1)

count=0;

else

count = count +32'd1;

always @ (negedge fin)

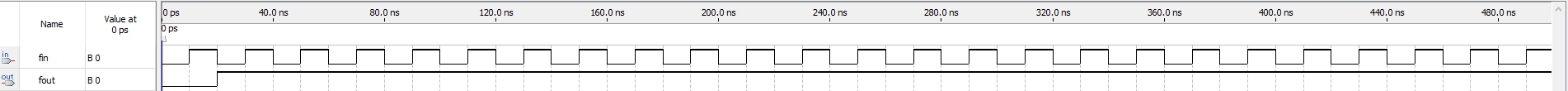
if(count>=\_DIVN) //count超過\_DIVN時輸出為0，小於則輸出為1

fout=1'd0;

else

fout=1'd1;

endmodule

****

**除頻器 25K：**

module FDIV20K(fin,fout);

input fin;

output fout;

wire [31:0] \_DIVN,DIVN;

reg [31:0] count;

reg fout;

assign DIVN=32'd2000; // DIVN為被除數，50M/2000=25k

assign \_DIVN = DIVN >>1;

always @(posedge fin)

if(count>=DIVN-32'd1)

count=0;

else

count = count +32'd1;

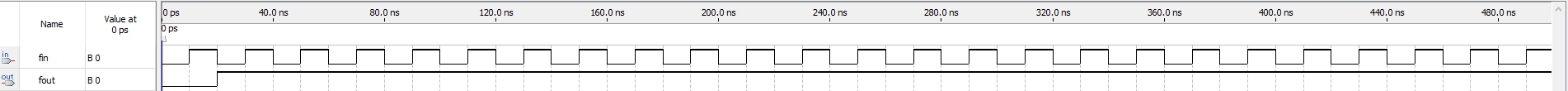
always @ (negedge fin)

if(count>=\_DIVN) //count超過\_DIVN時輸出為0，小於則輸出為1

fout=1'd0;

else

fout=1'd1;

****endmodule

module FDIV20K(fin,fout);

input fin;

output fout;

wire [31:0] \_DIVN,DIVN;

reg [31:0] count;

reg fout;

assign DIVN=32'd5000000; // DIVN為被除數，50M/5000000=20k

assign \_DIVN = DIVN >>1;

always @(posedge fin)

if(count>=DIVN-32'd1)

count=0;

else

count = count +32'd1;

always @ (negedge fin)

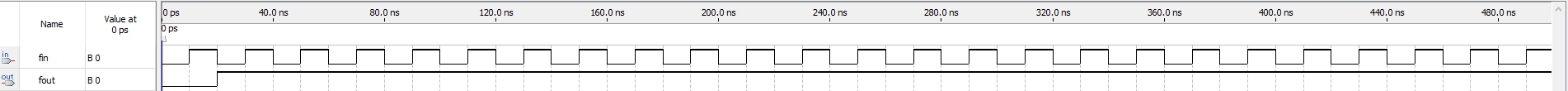
if(count>=\_DIVN) //count超過\_DIVN時輸出為0，小於則輸出為1

fout=1'd0;

else

fout=1'd1;

endmodule

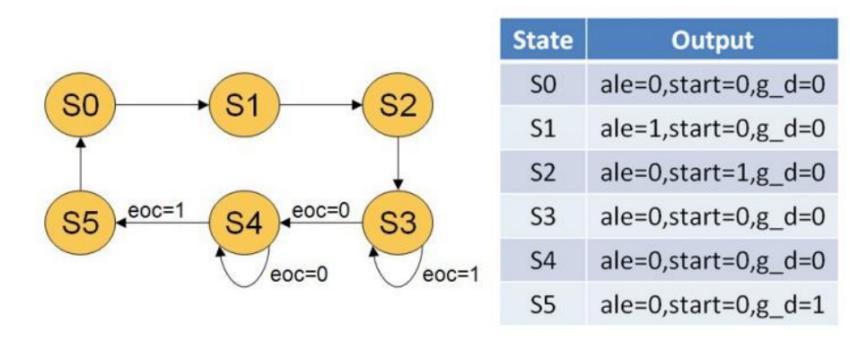
****

**AD Controller：**

module ADctrl(eoc,clk,ale,start,g\_d);

input eoc,clk;

output ale,start,g\_d;

reg[2:0]ADctr;

reg[2:0]cs,ns;

always @(posedge clk)

cs<=ns;

assign ale=ADctr[2];

assign start=ADctr[1];

assign g\_d=ADctr[0];

always@(cs)

case(cs) //每個狀態的輸出

3'd0:ADctr<=3'b000; //ADctr的3個位元分別代表 ale,start,g\_d

3'd1:ADctr<=3'b100;

3'd2:ADctr<=3'b010;

3'd3:ADctr<=3'b000;

3'd4:ADctr<=3'b000;

3'd5:ADctr<=3'b001;

endcase

always@(\*) //轉換狀態的條件

case(cs)

3'd0:ns=3'd1;

3'd1:ns=3'd2;

3'd2:ns=3'd3;

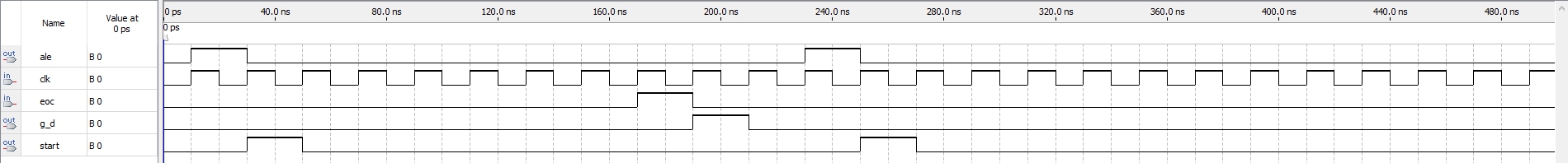
3'd3:ns=(eoc)?3'd3:3'd4; //當eoc為1時維持原狀態，為0時進入下一個狀態

3'd4:ns=(eoc)?3'd5:3'd4; //當eoc為0時維持原狀態，為1時進入下一個狀態

3'd5:ns=3'd0;

endcase

endmodule

從上圖可見狀態輸出波形的確與上面的狀態圖一致

**Ad Dff8：**

module ADdff8(clk,en,din,dout);

input clk,en;

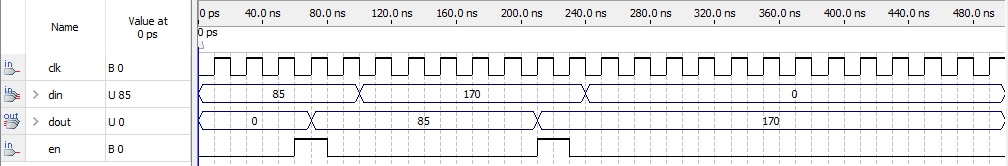
input [7:0]din;

output reg[7:0]dout;

always @(posedge clk)

dout=(en==1'd1)?din:dout; //當enable為1時dout=din，為0時dout則不變

endmodule

****

**Dff8：**

module Dff8(clk,din,dout);

input clk;

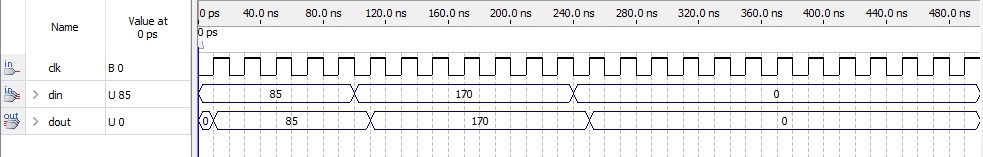
input [7:0]din;

output reg[7:0]dout;

always@(posedge clk) //正緣觸發時dout才輸出

dout=din;

endmodule

****

**Seg7\_Decoder：**

module \_7segENC(D,\_7seg);

input [3:0]D;

output [6:0]\_7seg;

reg [6:0]\_7seg;

always @(D)

begin

case(D)

4'd0:\_7seg=7'b1000000;

4'd1:\_7seg=7'b1111001;

4'd2:\_7seg=7'b0100100;

4'd3:\_7seg=7'b0110000;

4'd4:\_7seg=7'b0011001;

4'd5:\_7seg=7'b0010010;

4'd6:\_7seg=7'b0000010;

4'd7:\_7seg=7'b1111000;

4'd8:\_7seg=7'b0000000;

4'd9:\_7seg=7'b0011000;

4'd10:\_7seg=7'b0001000;

4'd11:\_7seg=7'b0000011;

4'd12:\_7seg=7'b1000110;

4'd13:\_7seg=7'b0100001;

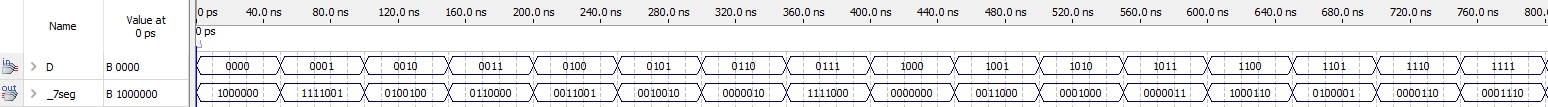
4'd14:\_7seg=7'b0000110;

4'd15:\_7seg=7'b0001110;

endcase

end

endmodule



**LED\_Decoder：**

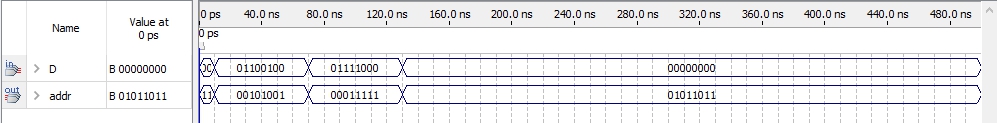
module LEDENC(D,addr);

input [7:0]D;

output [7:0]addr;

assign addr=(8'd183-D)>>1; //把0°的數位IC碥碼值減掉熱敏電阻的IC碥碼值除2

endmodule

****

**OneShot：**

module oneshot(clki,clk,shot);

input clki,clk;

output shot;

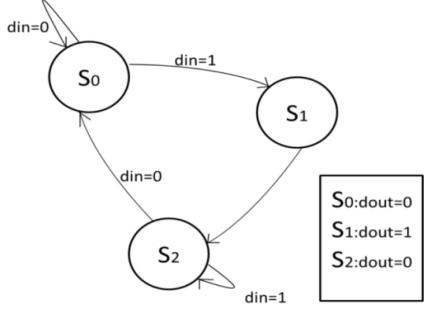
reg shot;

reg [1:0]cs,ns;

always@(posedge clk)

cs<=ns;

always@(cs)

case(cs)

2'd0:shot<=1'b0;

每個state要進行的變化

2'd1:shot<=1'b1;

2'd2:shot<=1'b0;

default:shot<=1'b0;

endcase

always@(cs)

case(cs)

state 變化條件

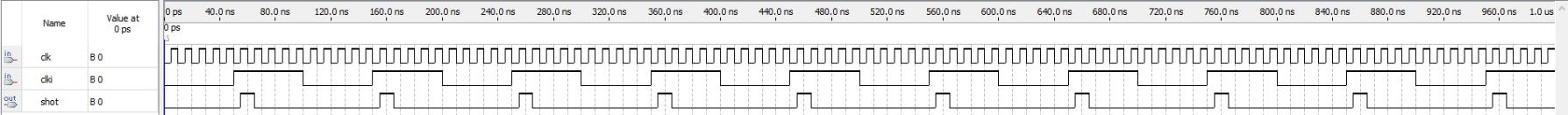
2'd0:ns<=(clki)?2'd1:2'd0;

2'd1:ns<=2'd2;

2'd2:ns<=(clki)?2'd2:2'd0;

default:ns<=2'd0;

endcase

endmodule

**Serial Out：**

module serial\_out(din,start,clk,D);

input start,clk;

input [23:0] din;

reg [23:0] store;

output wire D;

always @(posedge clk or posedge start)

begin

if(start)

store = din;

else

begin

進行輸出移位

store[23] = store[22];

store[22] = store[21];

store[21] = store[20];

store[20] = store[19];

store[19] = store[18];

store[18] = store[17];

store[17] = store[16];

store[16] = store[15];

store[15] = store[14];

store[14] = store[13];

store[13] = store[12];

store[12] = store[11];

store[11] = store[10];

store[10] = store[9];

store[9] = store[8];

store[8] = store[7];

store[7] = store[6];

store[6] = store[5];

store[5] = store[4];

store[4] = store[3];

store[3] = store[2];

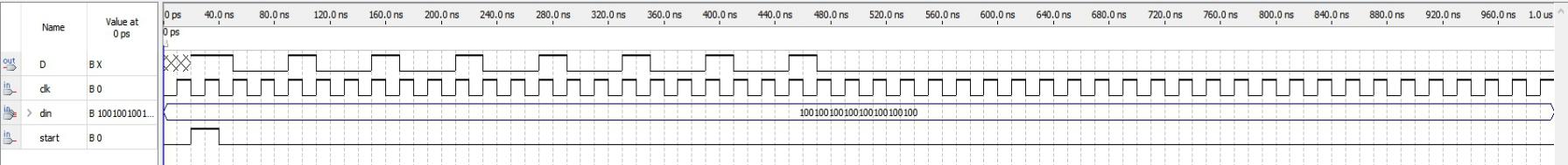
store[2] = store[1];

store[1] = store[0];

end

end

assign D = store[23];

endmodule

**實驗心得**

這次實驗用到了LED電路，只是把位置產生器拿掉，透過LED解碼器把IC的輸出當成LED電路的記憶體位置。這兩次的實驗都用了verilog來寫IC控制模組，其中狀態機的概念經常用到，要控制IC就需要清楚IC裏的運作，時脈的分析十分重要，要等IC發出eoc訊號，方可繼續下一步。實驗時把輸出脚位的位元順序接錯導致出來的值一直跳來跳去，花了很多時間才發現這個問題，正常的輸出溫度的改變應該是要一度一度的改變，不會直接跳。總括而言，這次實驗十分有趣，只要把概念弄懂就可以輕鬆完成。