96 年度大學院校 積體電路設計競賽

標準單元式(Cell-Based)競賽特優組解答

研究所/大學組

一、設計結果報告

Design Report Form

隊號(Team number): cb039			
RTL category			
Design Stage	*Description*	*File Name*	
RTL Simulation	使用之 HDL 名稱 (請填入 Verilog 或 VHDL)	Verilog	
RTL Simulation	RTL 檔案名稱 (RTL Netlist file name)	DPA.v	
Gate-Level category			
Design Stage	*Description*	*File Name*	
Pre-layout Gate-level Simulation	Gate-Level 檔案名稱 (Gate-Level Netlist file name)	DPA_syn.v	
	Pre-layout sdf 檔案名稱	DPA_syn.sdf	
Physical category			
Design Stage	*Description*	*File Name or Value*	
P&R	使用之 P&R Tool (請填入 Astro 或 SOC Encounter)	Astro	
	設計資料庫檔案名稱(Library name)	DPA_astro	
	佈局檔檔案名稱(GDSII file name)	DPA_pr.gds	
	佈局面積(layout area)	(182.96) um X (179.74) um	
	佈局座標點	左下角座標點(Lower-Left Coordinate) : XLB = 0.0 YLB = 0.0 右上角座標點(Upper-Right Coordinate): XRT = 182.96 YRT = 179.74	
	DRC report file	DRC.report	
	LVS report file	LVS.report	
Post-layout Gate-level Simulation	Post-layout Gate-Level 檔案名稱	DPA_pr.v	
	Post-layout sdf 檔案名稱	DPA_pr.sdf	
	toggle count	703,339,409	
其他說明事項(Any other information you want to specify:(如設計特點 ...) 如寫不下可寫於背面			

二、暫存器轉換階層(RTL level)設計結果 (Verilog HDL)

```
module DPA (clk,reset,IM_A, IM_Q,IM_D,IM_WEN,CR_A,CR_Q);
input clk;
input reset;
output [19:0] IM_A;
input [23:0] IM_Q;
output [23:0] IM_D;
output IM_WEN;
output [8:0] CR_A;
input [12:0] CR_Q;

//-----------------------------------

reg [19:0] IM_A;
reg [23:0] IM_D;
reg [8:0] CR_A;

parameter INI     = 5'd0;
parameter FBR     = 5'd1;
parameter PNR     = 5'd2;
parameter P1AR    = 5'd3;
parameter P1SR    = 5'd4;
parameter P2AR    = 5'd5;
parameter P2SR    = 5'd6;
parameter P3AR    = 5'd7;
parameter P3SR    = 5'd8;
parameter P4AR    = 5'd9;
parameter P4SR    = 5'd10;
parameter ENINI = 5'd11;
parameter CHINI = 5'd12;
parameter IDINI = 5'd13;
parameter ENMID = 5'd14;
parameter CHMID = 5'd15;
parameter IDMID = 5'd16;
parameter ENTU1 = 5'd17;
parameter CHTU1 = 5'd18;
parameter IDTU1 = 5'd19;
parameter ENTU2 = 5'd20;
parameter CHTU2 = 5'd21;
parameter IDTU2 = 5'd22;

reg IM_WEN;



reg [4 :0] cs, ns, cs_d;
reg [7 :0] hour, minute, second;
reg [19:0] FB_addr;
reg [2 :0] photo_num;
reg [19:0] p1_addr;
reg [1 :0] p1_size;
reg [19:0] p2_addr;
reg [1 :0] p2_size;
reg [19:0] p3_addr;
reg [1 :0] p3_size;
reg [19:0] p4_addr;
reg [1 :0] p4_size;

reg [19:0] po_addr;
reg [1 :0] po_size;

reg [19:0] cycle_cnt;
reg [7 :0] out_h_cnt, out_l_cnt;
reg [2 :0] state_cnt;
reg [2 :0] state_cnt_d;
reg [2 :0] photo_cnt;

wire [8:0]   cr_addr;
reg [3:0]   cr_cnt;
reg [2:0]   time_cnt;
reg [3:0]   cr_index;
reg [23:0] cr_out;
wire [4 :0] cr_ver_addr;
wire [7 :0] cr_ver_addr_t;
reg   [3 :0] ini_addr;

reg [7 :0] r0, g0, b0;
reg [7 :0] r1, g1, b1;
reg [7 :0] r2, g2, b2;
reg [7 :0] r3, g3, b3;
reg [7 :0] rout, gout, bout;

wire [8:0] ra01, ga01, ba01;
wire [8:0] ra02, ga02, ba02;
wire [9:0] ra03, ga03, ba03;

reg [3:0] h0, h1, m0, m1, s0, s1;

wire [19:0] read_addr;
wire [19:0] write_addr;

reg [17:0] bias_addr;

wire [6:0] out_h_cnt_128, out_l_cnt_128;

assign ra01 = r0 + r1;
assign ga01 = g0 + g1;
assign ba01 = b0 + b1;

assign ra02 = r0 + r2;
assign ga02 = g0 + g2;
assign ba02 = b0 + b2;

assign ra03 = r0 + r1 + r2 + r3;
assign ga03 = g0 + g1 + g2 + g3;
assign ba03 = b0 + b1 + b2 + b3;

//assign IM_A          = (IM_WEN == 1'b1) ? read_addr : write_addr;
assign write_addr = FB_addr + {out_h_cnt, out_l_cnt};
assign read_addr = {2'b0, bias_addr} + po_addr;

//assign CR_A = cr_addr;
assign cr_addr = {2'b0, cr_index, 3'b0} + {1'b0, cr_index, 4'b0} + cr_ver_addr_t[4:0];

assign cr_ver_addr_t = out_h_cnt - 8'd232;
assign cr_ver_addr = cr_ver_addr_t[4:0];

assign out_h_cnt_128 = (&out_h_cnt[7:1]) ? out_h_cnt[7:1] : out_h_cnt[7:1] + 1'b1;
assign out_l_cnt_128 = (&out_l_cnt[7:1]) ? out_l_cnt[7:1] : out_l_cnt[7:1] + 1'b1;

//assign IM_D = (out_h_cnt > 8'd231 && out_l_cnt > 8'd151) ? cr_out : {rout, gout, bout};

always @(cs or ini_addr or IM_WEN or read_addr or write_addr)
#10 IM_A         = (cs < 5'd11) ? ini_addr : (IM_WEN == 1'b1) ? read_addr : write_addr;

always @(cr_addr)
#10 CR_A = cr_addr;

always @(out_h_cnt or out_l_cnt or cr_out or rout or gout or bout)
#10 IM_D = (out_h_cnt > 8'd231 && out_l_cnt > 8'd151) ? cr_out : {rout, gout, bout};


always @(posedge clk or posedge reset)
begin
        if(reset)
                cs <= INI;
        else
                cs <= ns;
end

always @(posedge clk or posedge reset)
begin
```

```
        if(reset)
                cs_d <= INI;
        else
                cs_d <= cs;
end

always @*
begin
        case(cs)
                INI    : ns = FBR;
                FBR    : ns = PNR;
                PNR    : ns = P1AR;
                P1AR   : ns = P1SR;
                P1SR   : if(photo_num <= 3'd1)
                        ns = ENINI;
                        else
                        ns = P2AR;
                P2AR   : ns = P2SR;
                P2SR   : if(photo_num <= 3'd2)
                        ns = ENINI;
                        else
                        ns = P3AR;
                P3AR   : ns = P3SR;
                P3SR   : if(photo_num <= 3'd3)
                        ns = ENINI;
                        else
                        ns = P4AR;
                P4AR   : ns = P4SR;
                P4SR   : ns = ENINI;
                ENINI : ns = CHINI;
                CHINI : if((&out_h_cnt) && (&out_l_cnt) && (state_cnt == 3'd4))
                        ns = IDINI;
                        else
                        ns = CHINI;
                IDINI : if(cycle_cnt == 20'd999999)
                        ns = ENMID;
                        else
                        ns = IDINI;
                ENMID : ns = CHMID;
                CHMID : if((&out_h_cnt) && (&out_l_cnt) && (state_cnt == 3'd4))
                        ns = IDMID;
                        else
                        ns = CHMID;
                IDMID : if(cycle_cnt == 20'd999999)
                        ns = ENTU1;
                        else
                        ns = IDMID;
                ENTU1 : ns = CHTU1;
                CHTU1 : if((&out_h_cnt) && (&out_l_cnt) && (state_cnt == 3'd4))
                        ns = IDTU1;
                        else
                        ns = CHTU1;
                IDTU1 : if(cycle_cnt == 20'd200000)
                        ns = ENTU2;
                        else
                        ns = IDTU1;
                ENTU2 : ns = CHTU2;
                CHTU2 : if((&out_h_cnt) && (&out_l_cnt) && (state_cnt == 3'd4))
                        ns = IDTU2;
                        else
                        ns = CHTU2;
                IDTU2 : if(cycle_cnt == 20'd999999)
                        ns = ENMID;
                        else
                        ns = IDTU2;
                default : ns = INI;
        endcase
end

always @(posedge clk or posedge reset)
begin
        if(reset)
```

```
            cycle_cnt <= 20'b0;
        else if(cycle_cnt == 20'd999999)
            cycle_cnt <= 20'b0;
        else
            cycle_cnt <= cycle_cnt + 1'b1;

end

always @(posedge clk or posedge reset)
begin
        if(reset)
            {hour, minute, second} <= 24'b0;
        else if(cs_d == INI)
            {hour, minute, second} <= IM_Q;
        else
        begin
            if(cycle_cnt == 20'd999999)
            begin
                if(second == 59)
                    second <= 8'b0;
                else
                    second <= second + 1'b1;

                if(second == 59)
                    if(minute == 59)
                        minute <= 8'd0;
                    else
                        minute <= minute + 1'b1;

                if(second == 59 && minute == 59)
                    if(hour == 23)
                        hour <= 8'b0;
                    else
                        hour <= hour + 1'b1;

            end
        end
end

always @(posedge clk or posedge reset)
begin
        if(reset)
            FB_addr <= 20'b0;
        else if(cs_d == FBR)
            FB_addr <= IM_Q;
end
always @(posedge clk or posedge reset)
begin
        if(reset)
            photo_num <= 3'b0;
        else if(cs_d == PNR)
            photo_num <= IM_Q[2:0];
end
always @(posedge clk or posedge reset)
begin
        if(reset)
            p1_addr <= 20'b0;
        else if(cs_d == P1AR)
            p1_addr <= IM_Q[19:0];
end
always @(posedge clk or posedge reset)
begin
        if(reset)
            p1_size <= 2'b0;
        else if(cs_d == P1SR)
            case(IM_Q[9:7])
                3'b001: p1_size <= 2'b01;
                3'b010: p1_size <= 2'b10;
                3'b100: p1_size <= 2'b11;
            endcase

end
```

```
always @(posedge clk or posedge reset)
begin
	if(reset)
		p2_addr <= 20'b0;
	else if(cs_d == P2AR)
		p2_addr <= IM_Q[19:0];
end
always @(posedge clk or posedge reset)
begin
	if(reset)
		p2_size <= 2'b0;
	else if(cs_d == P2SR)
		case(IM_Q[9:7])
			3'b001: p2_size <= 2'b01;
			3'b010: p2_size <= 2'b10;
			3'b100: p2_size <= 2'b11;
		endcase
end
always @(posedge clk or posedge reset)
begin
	if(reset)
		p3_addr <= 20'b0;
	else if(cs_d == P3AR)
		p3_addr <= IM_Q[19:0];
end
always @(posedge clk or posedge reset)
begin
	if(reset)
		p3_size <= 2'b0;
	else if(cs_d == P3SR)
		case(IM_Q[9:7])
			3'b001: p3_size <= 2'b01;
			3'b010: p3_size <= 2'b10;
			3'b100: p3_size <= 2'b11;
		endcase
end
always @(posedge clk or posedge reset)
begin
	if(reset)
		p4_addr <= 20'b0;
	else if(cs_d == P4AR)
		p4_addr <= IM_Q[19:0];
end
always @(posedge clk or posedge reset)
begin
	if(reset)
		p4_size <= 2'b0;
	else if(cs_d == P4SR)
		case(IM_Q[9:7])
			3'b001: p4_size <= 2'b01;
			3'b010: p4_size <= 2'b10;
			3'b100: p4_size <= 2'b11;
		endcase
end

always @(posedge clk or posedge reset)
begin
	if(reset)
		state_cnt <= 3'b0;
	else if(cs == ENINI || cs == ENMID || cs == ENTU1 || cs == ENTU2)
		state_cnt <= 3'b0;
	else if(cs == CHINI || cs == CHMID || cs == CHTU1 || cs == CHTU2)
		if(state_cnt == 3'd4)
			state_cnt <= 3'b0;
		else
			state_cnt <= state_cnt + 1'b1;
end

always @(posedge clk or posedge reset)
begin
	if(reset)
		state_cnt_d <= 3'b0;
```

```
        else
                state_cnt_d <= state_cnt;
end

always @(posedge clk or posedge reset)
begin
        if(reset)
        begin
                out_h_cnt <= 8'b0;
                out_l_cnt <= 8'b0;
        end
        else
        begin
                case(cs)
                        ENINI : begin
                                        out_h_cnt <= 8'b0;
                                        out_l_cnt <= 8'b0;
                                end
                        CHINI : if(state_cnt == 3'd4)
                                begin
                                        if(out_l_cnt == 8'd255)
                                                out_h_cnt <= out_h_cnt + 1'b1;
                                        out_l_cnt <= out_l_cnt + 1'b1;
                                end
                        ENMID : begin
                                        out_h_cnt <= 8'd232;
                                        out_l_cnt <= 8'd152;
                                end
                        CHMID : if(state_cnt == 3'd4)
                                begin
                                        if(out_l_cnt == 8'd255)
                                                out_h_cnt <= out_h_cnt + 1'b1;

                                        if(out_l_cnt == 8'd255)
                                                out_l_cnt <= 8'd152;
                                        else
                                                out_l_cnt <= out_l_cnt + 1'b1;
                                end
                        ENTU1 : begin
                                        out_h_cnt <= 8'b0;
                                        out_l_cnt <= 8'b1;
                                end
                        CHTU1 : if(state_cnt == 3'd4)
                                begin
                                        if(out_h_cnt <= 8'd231)
                                        begin
                                                if(out_l_cnt[7:1] == 7'd127)
                                                        out_h_cnt <= out_h_cnt + 1'b1;
                                                if(out_h_cnt[0] == 1'b0)
                                                        if(out_l_cnt == 8'd255)
                                                                out_l_cnt <= 8'b0;
                                                        else
                                                                out_l_cnt <= out_l_cnt + 2'd2;
                                                else
                                                        if(out_l_cnt == 8'd254)
                                                                out_l_cnt <= 8'b1;
                                                        else
                                                                out_l_cnt <= out_l_cnt + 2'd2;
                                        end
                                        else
                                        begin
                                                if(out_l_cnt == 8'd255)
                                                        out_h_cnt <= out_h_cnt + 1'b1;
                                                if(out_l_cnt <= 8'd150) // !!!
                                                begin
                                                        out_l_cnt <= out_l_cnt + 2'd2;
                                                end
                                                else
                                                begin
                                                        if(out_l_cnt == 8'd255)
                                                                if(out_h_cnt[0] == 1'b0)
                                                                        out_l_cnt <= 8'b0;
                                                                else
                                                                        out_l_cnt <= 8'b1;
                                                        else
                                                                out_l_cnt <= out_l_cnt + 1'b1;
                                                end
                                        end
                                end
                        ENTU2 : begin
                                        out_h_cnt <= 8'b0;
                                        out_l_cnt <= 8'b0;
                                end
                        CHTU2 : if(state_cnt == 3'd4)
                                begin
                                        if(out_h_cnt <= 8'd231)
                                        begin
                                                if(out_l_cnt[7:1] == 7'd127)
                                                        out_h_cnt <= out_h_cnt + 1'b1;
                                                if(out_h_cnt[0] == 1'b0)
                                                        if(out_l_cnt == 8'd254)
                                                                out_l_cnt <= 8'b1;
                                                        else
                                                                out_l_cnt <= out_l_cnt + 2'd2;
                                                else
                                                        if(out_l_cnt == 8'd255)
                                                                out_l_cnt <= 8'b0;
                                                        else
                                                                out_l_cnt <= out_l_cnt + 2'd2;
                                        end
                                        else
                                        begin
                                                if(out_l_cnt == 8'd255)
                                                        out_h_cnt <= out_h_cnt + 1'b1;
                                                if(out_l_cnt <= 8'd150) // !!!
                                                begin
                                                        out_l_cnt <= out_l_cnt + 2'd2;
                                                end
                                                else
                                                begin
                                                        if(out_l_cnt == 8'd255)
                                                                if(out_h_cnt[0] == 1'b0)
                                                                        out_l_cnt <= 8'b1;
                                                                else
                                                                        out_l_cnt <= 8'b0;
                                                        else
                                                                out_l_cnt <= out_l_cnt + 1'b1;
                                                end
                                        end
                                end
                        endcase
                end

                end

                always @(posedge clk or posedge reset)
                begin
                        if(reset)
                                photo_cnt <= 3'b1;
                        else if(cs == ENTU1)
                                if(photo_cnt == photo_num)
                        photo_cnt <= 3'b1;
                                else
                        photo_cnt <= photo_cnt + 1'b1;
                end


                always @(cs or state_cnt)
                begin
                        #10
                        IM_WEN = 1'b1;
                        if(cs == CHINI || cs == CHMID || cs == CHTU1 || cs ==CHTU2)
                                if(state_cnt == 3'd4)
                        IM_WEN = 1'b0;
```

```
end

always @(posedge clk or posedge reset)
begin
	if(reset)
	begin
		{r0, g0, b0} <= 24'b0;
		{r1, g1, b1} <= 24'b0;
		{r2, g2, b2} <= 24'b0;
	end
	else if(cs == CHINI || cs == CHMID || cs == CHTU1 || cs ==CHTU2)
	begin
		if(state_cnt_d == 3'd0)
			{r0, g0, b0} <= IM_Q;
		if(state_cnt_d == 3'd1)
			{r1, g1, b1} <= IM_Q;
		if(state_cnt_d == 3'd2)
			{r2, g2, b2} <= IM_Q;
	end

end

always @(IM_Q)
begin
	{r3, g3, b3} <= IM_Q;
end

always @*
begin
	case(photo_cnt)
		3'd1 : begin po_addr = p1_addr; po_size = p1_size; end
		3'd2 : begin po_addr = p2_addr; po_size = p2_size; end
		3'd3 : begin po_addr = p3_addr; po_size = p3_size; end
		3'd4 : begin po_addr = p4_addr; po_size = p4_size; end
		default : begin po_addr = p1_addr; po_size = p1_size; end
	endcase
end

always @*
begin
	case(po_size)
		2'b01 : begin
				case({out_h_cnt[0], out_l_cnt[0]})
					2'b00 : {rout, gout, bout} = {r0 , g0 , b0};
					2'b01 : {rout, gout, bout} = {ra01[8:1], ga01[8:1], ba01[8:1]};
					2'b10 : {rout, gout, bout} = {ra02[8:1], ga02[8:1], ba02[8:1]};
					2'b11 : {rout, gout, bout} = {ra03[9:2], ga03[9:2], ba03[9:2]};
				endcase
			end
		2'b10 : begin
				{rout, gout, bout} = {r0 , g0 , b0};
			end
		2'b11 : begin
				{rout, gout, bout} = {ra03[9:2], ga03[9:2], ba03[9:2]};
			end
		default : begin
				{rout, gout, bout} = {r0 , g0 , b0};
			end
	endcase
end

always @(posedge clk or posedge reset)
begin
	if(reset)
		cr_cnt <= 4'b0;
	else if(cs == ENINI || cs == ENMID || cs == ENTU1 || cs == ENTU2)
		cr_cnt <= 4'b0;
	else if(cs == CHINI || cs == CHMID || cs == CHTU1 || cs == CHTU2)
	begin
		if(state_cnt == 3'd4)
		begin
			if(out_h_cnt > 8'd231 && out_l_cnt > 8'd151)
```

```
                                if(cr_cnt == 4'd12)
                                        cr_cnt <= 4'd0;
                                else
                                        cr_cnt <= cr_cnt + 1'b1;
                end
        end
end
always @(posedge clk or posedge reset)
begin
        if(reset)
                time_cnt <= 3'b0;
        else if(cs == ENINI || cs == ENMID || cs == ENTU1 || cs == ENTU2)
                time_cnt <= 3'b0;
        else if(cs == CHINI || cs == CHMID || cs == CHTU1 || cs == CHTU2)
        begin
                if(state_cnt == 3'd4)
                begin
                        if(out_h_cnt > 8'd231 && out_l_cnt > 8'd151)
                                if(cr_cnt == 4'd12)
                                        time_cnt <= time_cnt + 1'b1;
                end
        end
end

always @*
begin
        case(time_cnt)
                3'd0 : cr_index = h1;
                3'd1 : cr_index = h0;
                3'd2 : cr_index = 4'd10;
                3'd3 : cr_index = m1;
                3'd4 : cr_index = m0;
                3'd5 : cr_index = 4'd10;
                3'd6 : cr_index = s1;
                3'd7 : cr_index = s0;
        endcase
end

always @*
begin
        case(cr_cnt)
                4'd0   :   cr_out = {24{CR_Q[12]}};
                4'd1   :   cr_out = {24{CR_Q[11]}};
                4'd2   :   cr_out = {24{CR_Q[10]}};
                4'd3   :   cr_out = {24{CR_Q[9]}};
                4'd4   :   cr_out = {24{CR_Q[8]}};
                4'd5   :   cr_out = {24{CR_Q[7]}};
                4'd6   :   cr_out = {24{CR_Q[6]}};
                4'd7   :   cr_out = {24{CR_Q[5]}};
                4'd8   :   cr_out = {24{CR_Q[4]}};
                4'd9   :   cr_out = {24{CR_Q[3]}};
                4'd10 :   cr_out = {24{CR_Q[2]}};
                4'd11 :   cr_out = {24{CR_Q[1]}};
                4'd12 :   cr_out = {24{CR_Q[0]}};
                default : cr_out = {24{CR_Q[0]}};
        endcase
end


always @*
begin
        case(hour)
                8'd0   : {h1, h0} = {4'd0, 4'd0};
                8'd1   : {h1, h0} = {4'd0, 4'd1};
                8'd2   : {h1, h0} = {4'd0, 4'd2};
                8'd3   : {h1, h0} = {4'd0, 4'd3};
                8'd4   : {h1, h0} = {4'd0, 4'd4};
                8'd5   : {h1, h0} = {4'd0, 4'd5};
                8'd6   : {h1, h0} = {4'd0, 4'd6};
                8'd7   : {h1, h0} = {4'd0, 4'd7};
                8'd8   : {h1, h0} = {4'd0, 4'd8};
                8'd9   : {h1, h0} = {4'd0, 4'd9};
                8'd10 : {h1, h0} = {4'd1, 4'd0};
                8'd11 : {h1, h0} = {4'd1, 4'd1};
                8'd12 : {h1, h0} = {4'd1, 4'd2};
                8'd13 : {h1, h0} = {4'd1, 4'd3};
                8'd14 : {h1, h0} = {4'd1, 4'd4};
                8'd15 : {h1, h0} = {4'd1, 4'd5};
                8'd16 : {h1, h0} = {4'd1, 4'd6};
                8'd17 : {h1, h0} = {4'd1, 4'd7};
                8'd18 : {h1, h0} = {4'd1, 4'd8};
                8'd19 : {h1, h0} = {4'd1, 4'd9};
                8'd20 : {h1, h0} = {4'd2, 4'd0};
                8'd21 : {h1, h0} = {4'd2, 4'd1};
                8'd22 : {h1, h0} = {4'd2, 4'd2};
                8'd23 : {h1, h0} = {4'd2, 4'd3};
                default : {h1, h0} = {4'd0, 4'd0};
        endcase
end

always @*
begin
        case(minute)
                8'd0   : {m1, m0} = {4'd0, 4'd0};
                8'd1   : {m1, m0} = {4'd0, 4'd1};
                8'd2   : {m1, m0} = {4'd0, 4'd2};
                8'd3   : {m1, m0} = {4'd0, 4'd3};
                8'd4   : {m1, m0} = {4'd0, 4'd4};
                8'd5   : {m1, m0} = {4'd0, 4'd5};
                8'd6   : {m1, m0} = {4'd0, 4'd6};
                8'd7   : {m1, m0} = {4'd0, 4'd7};
                8'd8   : {m1, m0} = {4'd0, 4'd8};
                8'd9   : {m1, m0} = {4'd0, 4'd9};

                8'd10   : {m1, m0} = {4'd1, 4'd0};
                8'd11   : {m1, m0} = {4'd1, 4'd1};
                8'd12   : {m1, m0} = {4'd1, 4'd2};
                8'd13   : {m1, m0} = {4'd1, 4'd3};
                8'd14   : {m1, m0} = {4'd1, 4'd4};
                8'd15   : {m1, m0} = {4'd1, 4'd5};
                8'd16   : {m1, m0} = {4'd1, 4'd6};
                8'd17   : {m1, m0} = {4'd1, 4'd7};
                8'd18   : {m1, m0} = {4'd1, 4'd8};
                8'd19   : {m1, m0} = {4'd1, 4'd9};

                8'd20   : {m1, m0} = {4'd2, 4'd0};
                8'd21   : {m1, m0} = {4'd2, 4'd1};
                8'd22   : {m1, m0} = {4'd2, 4'd2};
                8'd23   : {m1, m0} = {4'd2, 4'd3};
                8'd24   : {m1, m0} = {4'd2, 4'd4};
                8'd25   : {m1, m0} = {4'd2, 4'd5};
                8'd26   : {m1, m0} = {4'd2, 4'd6};
                8'd27   : {m1, m0} = {4'd2, 4'd7};
                8'd28   : {m1, m0} = {4'd2, 4'd8};
                8'd29   : {m1, m0} = {4'd2, 4'd9};

                8'd30   : {m1, m0} = {4'd3, 4'd0};
                8'd31   : {m1, m0} = {4'd3, 4'd1};
                8'd32   : {m1, m0} = {4'd3, 4'd2};
                8'd33   : {m1, m0} = {4'd3, 4'd3};
                8'd34   : {m1, m0} = {4'd3, 4'd4};
                8'd35   : {m1, m0} = {4'd3, 4'd5};
                8'd36   : {m1, m0} = {4'd3, 4'd6};
                8'd37   : {m1, m0} = {4'd3, 4'd7};
                8'd38   : {m1, m0} = {4'd3, 4'd8};
                8'd39   : {m1, m0} = {4'd3, 4'd9};


                8'd40   : {m1, m0} = {4'd4, 4'd0};
                8'd41   : {m1, m0} = {4'd4, 4'd1};
                8'd42   : {m1, m0} = {4'd4, 4'd2};
                8'd43   : {m1, m0} = {4'd4, 4'd3};
                8'd44   : {m1, m0} = {4'd4, 4'd4};
                8'd45   : {m1, m0} = {4'd4, 4'd5};
```

```
          8'd46   : {m1, m0} = {4'd4, 4'd6};
          8'd47   : {m1, m0} = {4'd4, 4'd7};
          8'd48   : {m1, m0} = {4'd4, 4'd8};
          8'd49   : {m1, m0} = {4'd4, 4'd9};

          8'd50   : {m1, m0} = {4'd5, 4'd0};
          8'd51   : {m1, m0} = {4'd5, 4'd1};
          8'd52   : {m1, m0} = {4'd5, 4'd2};
          8'd53   : {m1, m0} = {4'd5, 4'd3};
          8'd54   : {m1, m0} = {4'd5, 4'd4};
          8'd55   : {m1, m0} = {4'd5, 4'd5};
          8'd56   : {m1, m0} = {4'd5, 4'd6};
          8'd57   : {m1, m0} = {4'd5, 4'd7};
          8'd58   : {m1, m0} = {4'd5, 4'd8};
          8'd59   : {m1, m0} = {4'd5, 4'd9};


          default : {m1, m0} = {4'd0, 4'd0};
     endcase
end

always @*
begin
     case(second)
          8'd0   : {s1, s0} = {4'd0, 4'd0};
          8'd1   : {s1, s0} = {4'd0, 4'd1};
          8'd2   : {s1, s0} = {4'd0, 4'd2};
          8'd3   : {s1, s0} = {4'd0, 4'd3};
          8'd4   : {s1, s0} = {4'd0, 4'd4};
          8'd5   : {s1, s0} = {4'd0, 4'd5};
          8'd6   : {s1, s0} = {4'd0, 4'd6};
          8'd7   : {s1, s0} = {4'd0, 4'd7};
          8'd8   : {s1, s0} = {4'd0, 4'd8};
          8'd9   : {s1, s0} = {4'd0, 4'd9};

          8'd10   : {s1, s0} = {4'd1, 4'd0};
          8'd11   : {s1, s0} = {4'd1, 4'd1};
          8'd12   : {s1, s0} = {4'd1, 4'd2};
          8'd13   : {s1, s0} = {4'd1, 4'd3};
          8'd14   : {s1, s0} = {4'd1, 4'd4};
          8'd15   : {s1, s0} = {4'd1, 4'd5};
          8'd16   : {s1, s0} = {4'd1, 4'd6};
          8'd17   : {s1, s0} = {4'd1, 4'd7};
          8'd18   : {s1, s0} = {4'd1, 4'd8};
          8'd19   : {s1, s0} = {4'd1, 4'd9};

          8'd20   : {s1, s0} = {4'd2, 4'd0};
          8'd21   : {s1, s0} = {4'd2, 4'd1};
          8'd22   : {s1, s0} = {4'd2, 4'd2};
          8'd23   : {s1, s0} = {4'd2, 4'd3};
          8'd24   : {s1, s0} = {4'd2, 4'd4};
          8'd25   : {s1, s0} = {4'd2, 4'd5};
          8'd26   : {s1, s0} = {4'd2, 4'd6};
          8'd27   : {s1, s0} = {4'd2, 4'd7};
          8'd28   : {s1, s0} = {4'd2, 4'd8};
          8'd29   : {s1, s0} = {4'd2, 4'd9};

          8'd30   : {s1, s0} = {4'd3, 4'd0};
          8'd31   : {s1, s0} = {4'd3, 4'd1};
          8'd32   : {s1, s0} = {4'd3, 4'd2};
          8'd33   : {s1, s0} = {4'd3, 4'd3};
          8'd34   : {s1, s0} = {4'd3, 4'd4};
          8'd35   : {s1, s0} = {4'd3, 4'd5};
          8'd36   : {s1, s0} = {4'd3, 4'd6};
          8'd37   : {s1, s0} = {4'd3, 4'd7};
          8'd38   : {s1, s0} = {4'd3, 4'd8};
          8'd39   : {s1, s0} = {4'd3, 4'd9};


          8'd40   : {s1, s0} = {4'd4, 4'd0};
          8'd41   : {s1, s0} = {4'd4, 4'd1};
          8'd42   : {s1, s0} = {4'd4, 4'd2};
```

```
                8'd43   : {s1, s0} = {4'd4, 4'd3};
                8'd44   : {s1, s0} = {4'd4, 4'd4};
                8'd45   : {s1, s0} = {4'd4, 4'd5};
                8'd46   : {s1, s0} = {4'd4, 4'd6};
                8'd47   : {s1, s0} = {4'd4, 4'd7};
                8'd48   : {s1, s0} = {4'd4, 4'd8};
                8'd49   : {s1, s0} = {4'd4, 4'd9};

                8'd50   : {s1, s0} = {4'd5, 4'd0};
                8'd51   : {s1, s0} = {4'd5, 4'd1};
                8'd52   : {s1, s0} = {4'd5, 4'd2};
                8'd53   : {s1, s0} = {4'd5, 4'd3};
                8'd54   : {s1, s0} = {4'd5, 4'd4};
                8'd55   : {s1, s0} = {4'd5, 4'd5};
                8'd56   : {s1, s0} = {4'd5, 4'd6};
                8'd57   : {s1, s0} = {4'd5, 4'd7};
                8'd58   : {s1, s0} = {4'd5, 4'd8};
                8'd59   : {s1, s0} = {4'd5, 4'd9};


                default : {s1, s0} = {4'd0, 4'd0};
        endcase
end

always @*
begin
        case(po_size)
                2'b01 :         case(state_cnt)
                                3'd0 : bias_addr = {4'b0, out_h_cnt[7:1] , out_l_cnt[7:1]};
                                3'd1 : bias_addr = {4'b0, out_h_cnt[7:1] , out_l_cnt_128 };
                                3'd2 : bias_addr = {4'b0, out_h_cnt_128    , out_l_cnt[7:1]};
                                3'd3 : bias_addr = {4'b0, out_h_cnt_128    , out_l_cnt_128 };
                                default : bias_addr = {out_h_cnt[7:1]        , out_l_cnt[7:1]};
                        endcase
                2'b10, 2'b00 : bias_addr = {2'b0, out_h_cnt , out_l_cnt};
                2'b11 : case(state_cnt)
                                3'd0 : bias_addr = {out_h_cnt, 1'b0, out_l_cnt, 1'b0};
                                3'd1 : bias_addr = {out_h_cnt, 1'b0, out_l_cnt, 1'b1};
                                3'd2 : bias_addr = {out_h_cnt, 1'b1, out_l_cnt, 1'b0};
                                3'd3 : bias_addr = {out_h_cnt, 1'b1, out_l_cnt, 1'b1};
                                default : bias_addr = {out_h_cnt, 1'b1, out_l_cnt, 1'b1};
                        endcase
        endcase
end

always @(posedge clk or posedge reset)
begin
        if(reset)
                ini_addr <= 4'b0;
        else if(ini_addr <= 4'd10)
                ini_addr <= ini_addr +1'b1;
end

endmodule
```

三、 模擬結果 (Gate-level Post-layout Simulation)

測試樣本 2:

```
ncverilog: 06.10-p001: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
TOOL:   ncverilog       06.10-p001: Started on May 16, 2008 at 09:26:43 CST
ncverilog
        testfixture.v
        ../DPA_pr.v
        tsmc13_neg.v
        +access+r
        +loadpli1=tglib:tg
        +define+tb2
…略

ncsim> run
FB init addr = 000cd000
=======  tick  0,  time=12:12:59  ========
 check at   0.401 sec
   -- image saved , filename =   tb2_image00.xpm
   -- check PASS
 check at   0.999 sec
=======  tick  1,  time=12:13: 0  ========
   -- check PASS
 check at   1.401 sec
   -- image saved , filename =   tb2_image01.xpm
   -- check PASS
 check at   1.999 sec
=======  tick  2,  time=12:13: 1  ========
   -- check PASS


total toggle count = 156075648



        ****************************
        **                        **         /|__/|
        **  Congratulations !!    **       / 0,0  |
        **                        **     /______   |
        **  Simulation Complete!! **    /^ ^ ^ \   |
        **                        **   |^ ^ ^ ^ |w|
        *************** ************    \m___m__|_|


Simulation complete via $finish(1) at time 2000002 US + 0
./testfixture.v:266         $finish;
ncsim> exit
TOOL:   ncverilog       06.10-p001: Exiting on May 16, 2008 at 09:29:56 CST  (total: 00:03:13)
```

測試樣本 3:

```
ncverilog: 06.10-p001: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
TOOL:   ncverilog       06.10-p001: Started on May 16, 2008 at 09:39:37 CST
ncverilog
        testfixture.v
        ../DPA_pr.v
        tsmc13_neg.v
        +access+r
        +loadpli1=tglib:tg
        +define+tb3
…略

ncsim> run
FB init addr = 0000810c
=======  tick  0,  time= 3:59:59  ========
 check at   0.401 sec
   -- image saved , filename =   tb3_image00.xpm
   -- check PASS
 check at   0.999 sec
=======  tick  1,  time= 4: 0: 0  ========
   -- check PASS
 check at   1.401 sec
   -- image saved , filename =   tb3_image01.xpm
   -- check PASS
 check at   1.999 sec
=======  tick  2,  time= 4: 0: 1  ========
   -- check PASS


total toggle count = 155260172



        ****************************
        **                        **       /|__/|
        **  Congratulations !!    **      / 0,0  |
        **                        **     /_____   |
        **  Simulation Complete!! **    /^ ^ ^ \  |
        **                        **   |^ ^ ^ ^ |w|
        *************** ************    \m___m__|_|


Simulation complete via $finish(1) at time 2000002 US + 0
./testfixture.v:266         $finish;
ncsim> exit
TOOL:   ncverilog       06.10-p001: Exiting on May 16, 2008 at 09:42:43 CST  (total: 00:03:06)
```

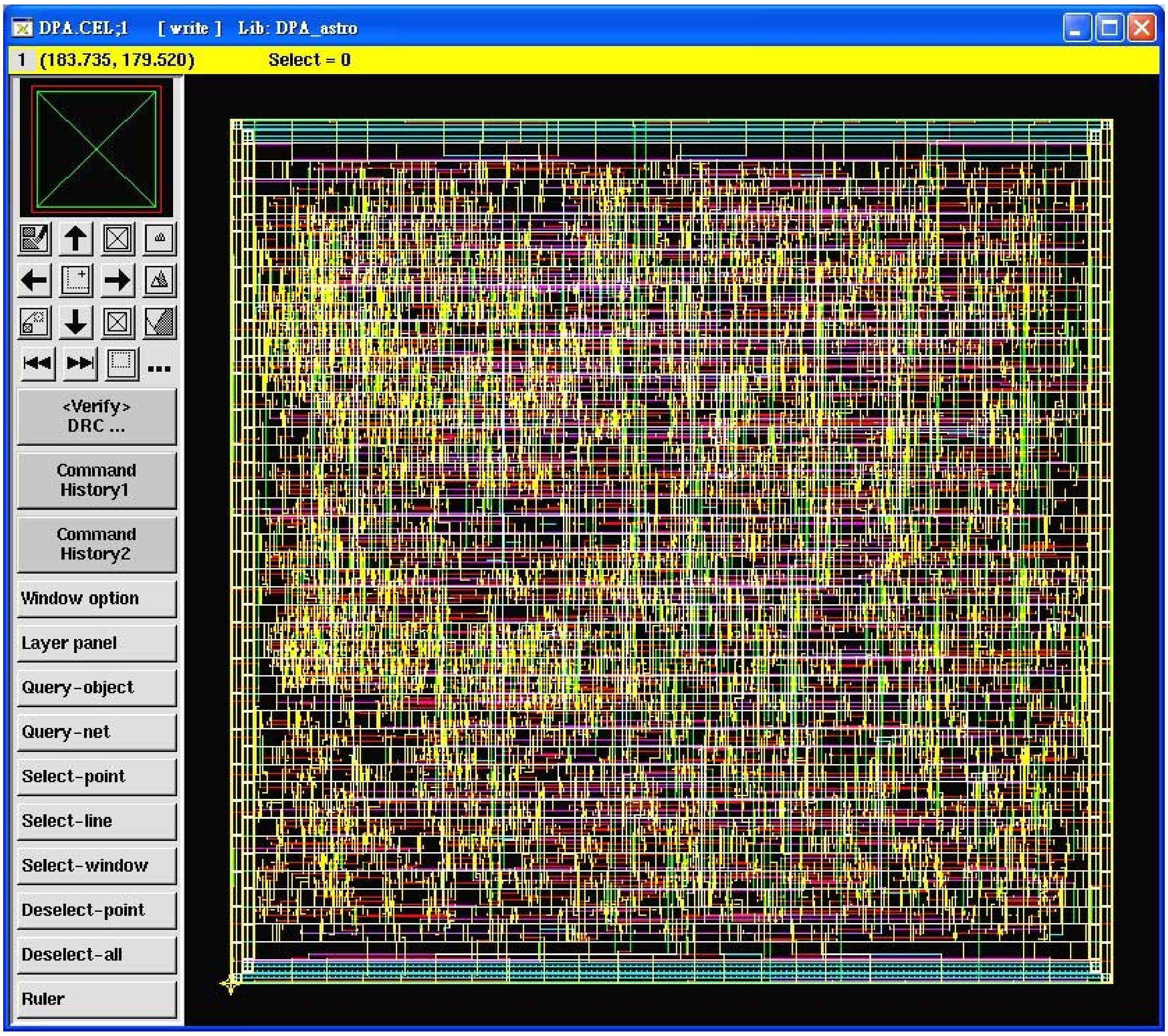
測試樣本 7:

```
ncverilog: 06.10-p001: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
TOOL:   ncverilog       06.10-p001: Started on May 16, 2008 at 09:43:50 CST
ncverilog
        testfixture.v
        ../DPA_pr.v
        tsmc13_neg.v
        +access+r
        +loadpli1=tglib:tg
        +define+tb7
…略

ncsim> run
FB init addr = 000f0000
=======  tick  0,  time=23:59:57  ========
 check at   0.401 sec
   -- image saved , filename =   tb7_image00.xpm
   -- check PASS
 check at   0.999 sec
=======  tick  1,  time=23:59:58  ========
   -- check PASS
 check at   1.401 sec
   -- image saved , filename =   tb7_image01.xpm
   -- check PASS
 check at   1.999 sec
=======  tick  2,  time=23:59:59  ========
   -- check PASS
 check tint image at   2.20001 sec
   -- image saved , filename = tb7_image02_t.xpm
   -- check PASS
 check at   2.401 sec
   -- image saved , filename =   tb7_image02.xpm
   -- check PASS
 check at   2.999 sec
=======  tick  3,  time= 0: 0: 0  ========
   -- check PASS
 check at   3.401 sec
   -- image saved , filename =   tb7_image03.xpm
   -- check PASS
 check at   3.999 sec
=======  tick  4,  time= 0: 0: 1  ========
   -- check PASS
 check tint image at   4.20001 sec
   -- image saved , filename = tb7_image04_t.xpm
   -- check PASS
 check at   4.401 sec
   -- image saved , filename =   tb7_image04.xpm
   -- check PASS
 check at   4.999 sec
=======  tick  5,  time= 0: 0: 2  ========
   -- check PASS
 check at   5.401 sec
   -- image saved , filename =   tb7_image05.xpm
   -- check PASS
 check at   5.999 sec
=======  tick  6,  time= 0: 0: 3  ========
   -- check PASS
 check tint image at   6.20001 sec
   -- image saved , filename = tb7_image06_t.xpm
   -- check PASS
 check at   6.401 sec
   -- image saved , filename =   tb7_image06.xpm
   -- check PASS
 check at   6.999 sec
=======  tick  7,  time= 0: 0: 4  ========
   -- check PASS
 check at   7.401 sec
   -- image saved , filename =   tb7_image07.xpm
   -- check PASS
 check at   7.999 sec
=======  tick  8,  time= 0: 0: 5  ========
   -- check PASS


total toggle count = 703339409
```

```
        ****************************
        **                        **        /|__/|
        **  Congratulations !!    **      / 0,0  |
        **                        **    /_____   |
        **  Simulation Complete!! **   /^ ^ ^ \  |
        **                        **  |^ ^ ^ ^ |w|
        *************** ************   \m___m__|_|

Simulation complete via $finish(1) at time 8000002 US + 0
./testfixture.v:266         $finish;
ncsim> exit
TOOL:   ncverilog       06.10-p001: Exiting on May 16, 2008 at 09:57:24 CST  (total: 00:13:34)
```

佈局與驗證

(a) 佈局圖


DPA.CEL;1 [write] Lib: DPA_astro
1 (183.735, 179.520) Select = 0
<Verify> DRC ...
Command History1
Command History2
Window option
Layer panel
Query-object
Query-net
Select-point
Select-line
Select-window
Deselect-point
Deselect-all
Ruler

(b) on-line DRC/LVS (Astro)驗證

DRC 驗證結果：

```
DRC Error Summary
Error Type          Num   Description
Met5 Thin&Fat2       0    metal5 minimum spacing [0.2 and 10.005] = 0.6
Met5 Fat&Fat         0    metal5 minimum spacing [0.39 and 0.39] = 0.24
Met5 Fat&Fat2        0    metal5 minimum spacing [0.39 and 10.005] = 0.6
Met5 Fat2&Fat2       0    metal5 minimum spacing [10.005 and 10.005] = 0.6
Met5 Overlap         0    metal5 & blockage overlap
Met5 Notch           0    metal5 notch (0.21)
Met5 FatNotch        0    metal5 Fat[0.39] notch (0.24)
Met5 Fat2Notch       0    metal5 Fat2[10.005] notch (0.6)
Met5 MinEncArea      0    metal5 minimum enclosed area = (265)
Via5 Width           0    via5 minimum width = 0.19
Via5 Spacing         0    via5 minimum spacing = 0.22
Via5&Via5Blk Sp      0    via5 & via5Blockage minimum spacing = 0.29
Via5&Via5Blk Ov      0    via5 & via5Blockage overlap
Met6 Width           0    metal6 minimum width = 0.2
Met6 Spacing         0    metal6 minimum spacing = 0.21
Met6 Thin&Fat        0    metal6 minimum spacing [0.2 and 0.39] = 0.24
Met6 Thin&Fat2       0    metal6 minimum spacing [0.2 and 10.005] = 0.6
Met6 Fat&Fat         0    metal6 minimum spacing [0.39 and 0.39] = 0.24
Met6 Fat&Fat2        0    metal6 minimum spacing [0.39 and 10.005] = 0.6
Met6 Fat2&Fat2       0    metal6 minimum spacing [10.005 and 10.005] = 0.6
Met6 Overlap         0    metal6 & blockage overlap
Met6 Notch           0    metal6 notch (0.21)
Met6 FatNotch        0    metal6 Fat[0.39] notch (0.24)
Met6 Fat2Notch       0    metal6 Fat2[10.005] notch (0.6)
Met6 MinEncArea      0    metal6 minimum enclosed area = (265)
Via6 Width           0    via6 minimum width = 0.19
Via6 Spacing         0    via6 minimum spacing = 0.22
Via6&Via6Blk Sp      0    via6 & via6Blockage minimum spacing = 0.29
Via6&Via6Blk Ov      0    via6 & via6Blockage overlap
Met7 Width           0    metal7 minimum width = 0.2
Met7 Spacing         0    metal7 minimum spacing = 0.21
Met7 Thin&Fat        0    metal7 minimum spacing [0.2 and 0.39] = 0.24
Met7 Thin&Fat2       0    metal7 minimum spacing [0.2 and 10.005] = 0.6
Met7 Fat&Fat         0    metal7 minimum spacing [0.39 and 0.39] = 0.24
Met7 Fat&Fat2        0    metal7 minimum spacing [0.39 and 10.005] = 0.6
Met7 Fat2&Fat2       0    metal7 minimum spacing [10.005 and 10.005] = 0.6
Met7 Overlap         0    metal7 & blockage overlap
Met7 Notch           0    metal7 notch (0.21)
Met7 FatNotch        0    metal7 Fat[0.39] notch (0.24)
Met7 Fat2Notch       0    metal7 Fat2[10.005] notch (0.6)
Met7 MinEncArea      0    metal7 minimum enclosed area = (265)
Via7 Width           0    via7 minimum width = 0.36
Via7 Spacing         0    via7 minimum spacing = 0.35
Via7&Via7Blk Sp      0    via7 & via7Blockage minimum spacing = 0.35
Via7&Via7Blk Ov      0    via7 & via7Blockage overlap
Met8 Width           0    metal8 minimum width = 0.4
Met8 Spacing         0    metal8 minimum spacing = 0.42
Met8 Thin&Fat        0    metal8 minimum spacing [0.4 and 10] = 0.6
Met8 Fat&Fat         0    metal8 minimum spacing [10 and 10] = 0.6
Met8 Overlap         0    metal8 & blockage overlap
Met8 Notch           0    metal8 notch (0.42)
Met8 FatNotch        0    metal8 Fat[10] notch (0.6)
Met8 MinEncArea      0    metal8 minimum enclosed area = (565)
Poly Width           0    poly minimum width = 0.13
Poly Spacing         0    poly minimum spacing = 0.18
Poly Overlap         0    poly & blockage overlap
Poly Notch           0    poly notch (0.18)
Cont Width           0    polyCont minimum width = 0.16
Cont Spacing         0    polyCont minimum spacing = 0.18
Met1 Width           0    metal1 minimum width = 0.16
Met1 Spacing         0    metal1 minimum spacing = 0.18
Met1 Thin&Fat        0    metal1 minimum spacing [0.16 and 0.3] = 0.22
Met1 Thin&Fat2       0    metal1 minimum spacing [0.16 and 10.005] = 0.6
Met1 Fat&Fat         0    metal1 minimum spacing [0.3 and 0.3] = 0.22
Met1 Fat&Fat2        0    metal1 minimum spacing [0.3 and 10.005] = 0.6
Met1 Fat2&Fat2       0    metal1 minimum spacing [10.005 and 10.005] = 0.6
Met1 Overlap         0    metal1 & blockage overlap
Met1 Notch           0    metal1 notch (0.18)
Met1 FatNotch        0    metal1 Fat[0.3] notch (0.22)
```

```
Met1 Fat2Notch        0      metal1 Fat2[10.005] notch (0.6)
Met1 MinEncArea       0      metal1 minimum enclosed area = (200)
Via1 Width            0      via1 minimum width = 0.19
Via1 Spacing          0      via1 minimum spacing = 0.22
Via1&Via1Blk Sp       0      via1 & via1Blockage minimum spacing = 0.29
Via1&Via1Blk Ov       0      via1 & via1Blockage overlap
Met2 Width            0      metal2 minimum width = 0.2
Met2 Spacing          0      metal2 minimum spacing = 0.21
Met2 Thin&Fat         0      metal2 minimum spacing [0.2 and 0.39] = 0.24
Met2 Thin&Fat2        0      metal2 minimum spacing [0.2 and 10.005] = 0.6
Met2 Fat&Fat          0      metal2 minimum spacing [0.39 and 0.39] = 0.24
Met2 Fat&Fat2         0      metal2 minimum spacing [0.39 and 10.005] = 0.6
Met2 Fat2&Fat2        0      metal2 minimum spacing [10.005 and 10.005] = 0.6
Met2 Overlap          0      metal2 & blockage overlap
Met2 Notch            0      metal2 notch (0.21)
Met2 FatNotch         0      metal2 Fat[0.39] notch (0.24)
Met2 Fat2Notch        0      metal2 Fat2[10.005] notch (0.6)
Met2 MinEncArea       0      metal2 minimum enclosed area = (265)
Via2 Width            0      via2 minimum width = 0.19
Via2 Spacing          0      via2 minimum spacing = 0.22
Via2&Via2Blk Sp       0      via2 & via2Blockage minimum spacing = 0.29
Via2&Via2Blk Ov       0      via2 & via2Blockage overlap
Met3 Width            0      metal3 minimum width = 0.2
Met3 Spacing          0      metal3 minimum spacing = 0.21
Met3 Thin&Fat         0      metal3 minimum spacing [0.2 and 0.39] = 0.24
Met3 Thin&Fat2        0      metal3 minimum spacing [0.2 and 10.005] = 0.6
Met3 Fat&Fat          0      metal3 minimum spacing [0.39 and 0.39] = 0.24
Met3 Fat&Fat2         0      metal3 minimum spacing [0.39 and 10.005] = 0.6
Met3 Fat2&Fat2        0      metal3 minimum spacing [10.005 and 10.005] = 0.6
Met3 Overlap          0      metal3 & blockage overlap
Met3 Notch            0      metal3 notch (0.21)
Met3 FatNotch         0      metal3 Fat[0.39] notch (0.24)
Met3 Fat2Notch        0      metal3 Fat2[10.005] notch (0.6)
Met3 MinEncArea       0      metal3 minimum enclosed area = (265)
Via3 Width            0      via3 minimum width = 0.19
Via3 Spacing          0      via3 minimum spacing = 0.22
Via3&Via3Blk Sp       0      via3 & via3Blockage minimum spacing = 0.29
Via3&Via3Blk Ov       0      via3 & via3Blockage overlap
Met4 Width            0      metal4 minimum width = 0.2
Met4 Spacing          0      metal4 minimum spacing = 0.21
Met4 Thin&Fat         0      metal4 minimum spacing [0.2 and 0.39] = 0.24
Met4 Thin&Fat2        0      metal4 minimum spacing [0.2 and 10.005] = 0.6
Met4 Fat&Fat          0      metal4 minimum spacing [0.39 and 0.39] = 0.24
Met4 Fat&Fat2         0      metal4 minimum spacing [0.39 and 10.005] = 0.6
Met4 Fat2&Fat2        0      metal4 minimum spacing [10.005 and 10.005] = 0.6
Met4 Overlap          0      metal4 & blockage overlap
Met4 Notch            0      metal4 notch (0.21)
Met4 FatNotch         0      metal4 Fat[0.39] notch (0.24)
Met4 Fat2Notch        0      metal4 Fat2[10.005] notch (0.6)
Met4 MinEncArea       0      metal4 minimum enclosed area = (265)
Via4 Width            0      via4 minimum width = 0.19
Via4 Spacing          0      via4 minimum spacing = 0.22
Via4&Via4Blk Sp       0      via4 & via4Blockage minimum spacing = 0.29
Via4&Via4Blk Ov       0      via4 & via4Blockage overlap
Met5 Width            0      metal5 minimum width = 0.2
Met5 Spacing          0      metal5 minimum spacing = 0.21
Met5 Thin&Fat         0      metal5 minimum spacing [0.2 and 0.39] = 0.24
```

LVS 驗證結果：

```
LVS Error Summary
Error Type            Num    Description
Floating Port         0      Floating ports have been detected by LVS.
Floating Net          0      Floating Nets have been detected by LVS.
SHORT                 0      SHORTS have been detected by LVS.
OPEN                  0      OPENS have been detected by LVS.
ElecEquivalent        0      Electrical-Equivalent Errors have been detected by LVS.
MustJoin              0      MustJoin Errors have been detected by LVS.
Min Area              0      Minimum Area Errors have been detected by LVS.
```