國立雲林科技大學

112 學年度第二學期

系統雛型設計(作業四)

HDMI 顯示名字

組別:第 8 組

學號:M11212045　M11212091

學生:李奕霖張富彥

日期:2024/5/15

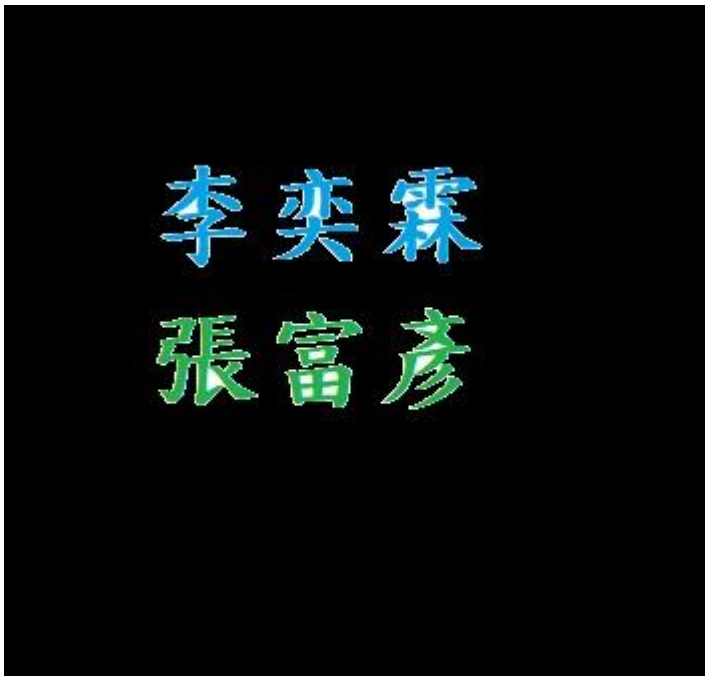
貢獻度:李奕霖 50%, 張富彥 50%

一、題目:

Homework4:

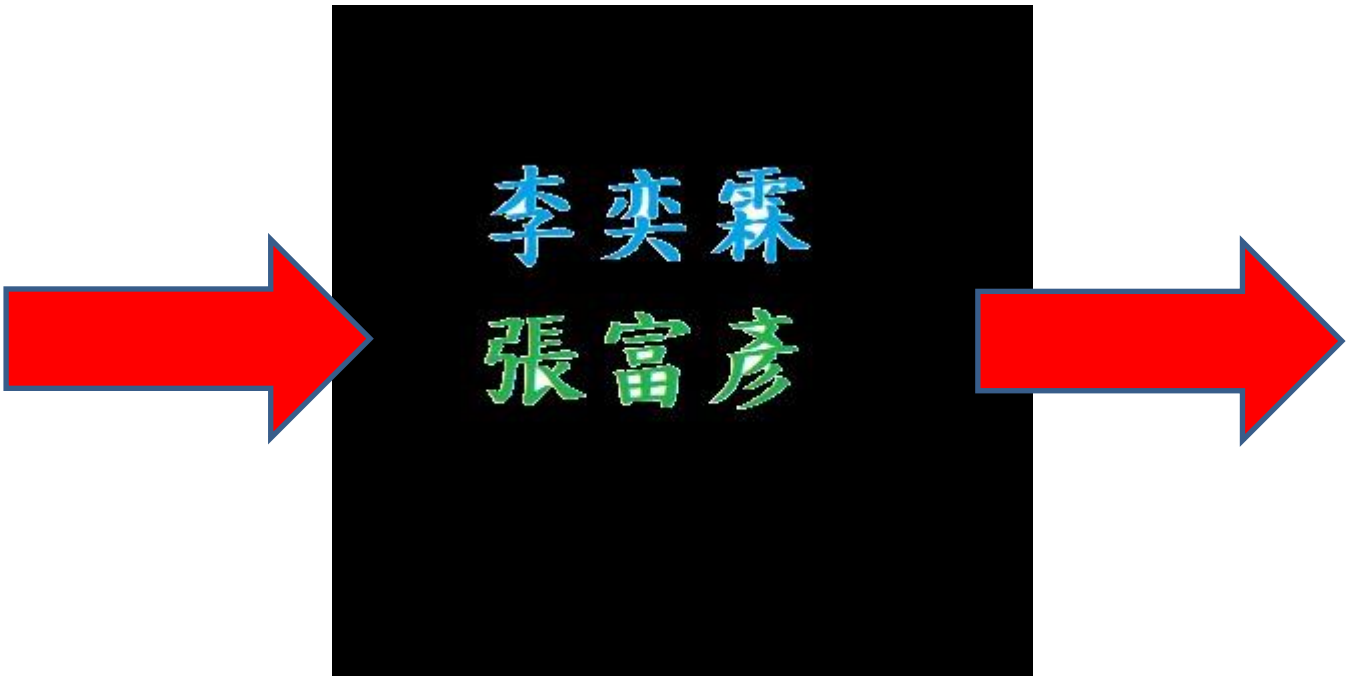
基礎題:

Use HDMI port to display your name:



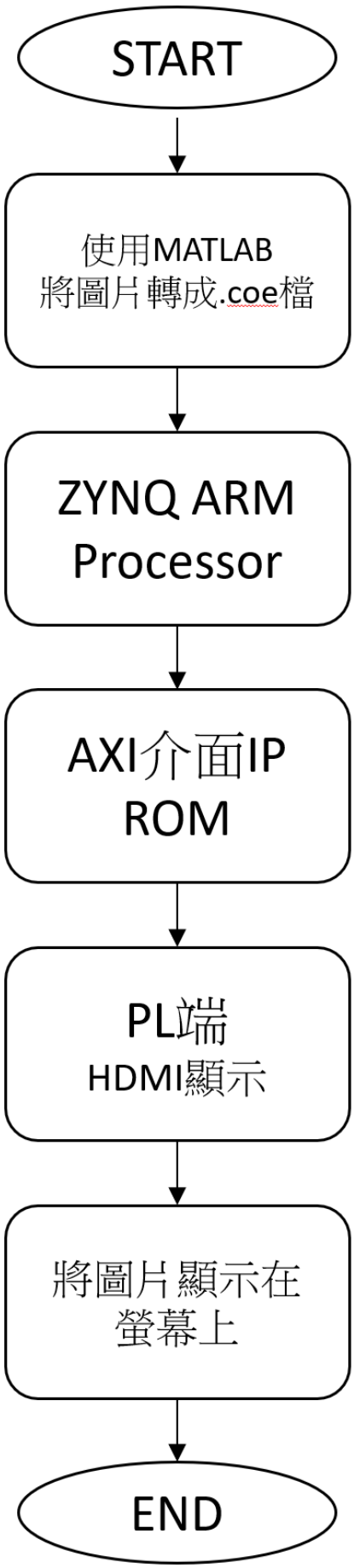
進階題:

Use HDMI port to display your name and shfit your name:

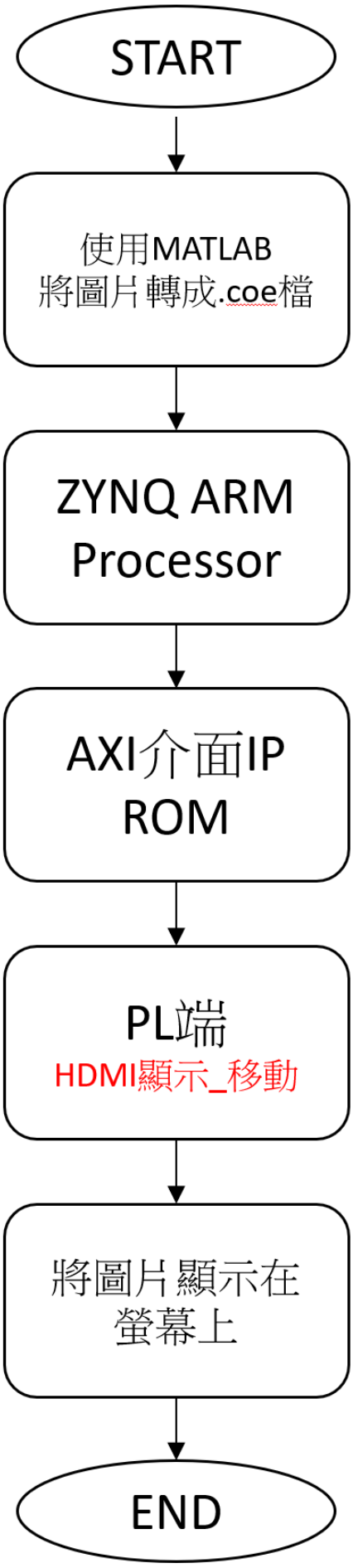


二、架構/流程圖:

基礎題:


START
使用MATLAB
將圖片轉成.coe檔
ZYNQ ARM
Processor
AXI介面IP
ROM
PL端
HDMI顯示
將圖片顯示在
螢幕上
END

進階題:


START
使用MATLAB
將圖片轉成.coe檔
ZYNQ ARM
Processor
AXI介面IP
ROM
PL端
HDMI顯示_移動
將圖片顯示在
螢幕上
END

三、MATLAB 程式將圖片轉.coe 檔

```
clear;
clc;
img = imread('name_new.png');        %讀取圖片
fid = fopen('name_new.coe', 'w');    %建立 COE

fprintf(fid, 'memory_initialization_radix=16;\n');
fprintf(fid, 'memory_initialization_vector=\n');

m = size(img);  %取得圖片尺寸，m(1)為高，m(2)為寬
%disp(m);
for i = 1:m(1)        %  height
    for j = 1:m(2)    %   Width
        %合併 RGB
        fprintf(fid, '%02X%02X%02X,\n' ,img(i,j,1), ...  % R
                                img(i,j,2), ...  % G
                                img(i,j,3));     % B
    end
end

fseek(fid, -2, 1); % 最後的逗號用分號取代
fprintf(fid, ';');

fclose(fid);
```

四、Verilog 程式碼

基礎題:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2024/05/09 15:51:06
// Design Name:
// Module Name: dataTimming
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module dataTimming(
    input clk,
    input rst,
    output hs,
    output vs,
    output de,
    output[7:0] rgb_r,
    output[7:0] rgb_g,
    output[7:0] rgb_b
);

parameter H_ACTIVE = 16'd1280;
parameter H_FP = 16'd110;
parameter H_SYNC = 16'd40;
parameter H_BP = 16'd220;
parameter V_ACTIVE = 16'd720;
parameter V_FP  = 16'd5;
```

```
parameter V_SYNC  = 16'd5;
parameter V_BP  = 16'd20;

parameter mem_w             = 9'd350;
parameter mem_h             = 9'd335;

parameter H_TOTAL = H_ACTIVE + H_FP + H_SYNC + H_BP;
parameter V_TOTAL = V_ACTIVE + V_FP + V_SYNC + V_BP;

reg hs_reg;
reg vs_reg;
reg hs_reg_d0;

reg vs_reg_d0;
reg[11:0] h_cnt;
reg[11:0] v_cnt;
reg[11:0] active_x;
reg[11:0] active_y;
reg[7:0] rgb_r_reg;
reg[7:0] rgb_g_reg;
reg[7:0] rgb_b_reg;
reg h_active;
reg v_active;
wire video_active;
reg video_active_d0;
assign hs = hs_reg_d0;
assign vs = vs_reg_d0;
assign video_active = h_active & v_active;
assign de = video_active_d0;
assign rgb_r = rgb_r_reg;
assign rgb_g = rgb_g_reg;
assign rgb_b = rgb_b_reg;

reg [16:0] rom_addr = 0;
wire [23:0] rom_out;

blk_mem_gen_0 rom_0(
    .addra(rom_addr),
    .clka(clk),
    .douta(rom_out)
);
```

```
always@(posedge clk or posedge rst)
begin
    if(rst)
        begin
            hs_reg_d0 <= 1'b0;
            vs_reg_d0 <= 1'b0;
            video_active_d0 <= 1'b0;
        end
    else
        begin
            hs_reg_d0 <= hs_reg;
            vs_reg_d0 <= vs_reg;
            video_active_d0 <= video_active;
        end
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        h_cnt <= 12'd0;
    else if(h_cnt == H_TOTAL - 1)
        h_cnt <= 12'd0;
    else
        h_cnt <= h_cnt + 12'd1;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        active_x <= 12'd0;
    else if(h_cnt >= H_FP + H_SYNC + H_BP - 1)
        active_x <= h_cnt - (H_FP[11:0] + H_SYNC[11:0] + H_BP[11:0] - 12'd1);
    else
        active_x <= active_x;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        active_y <= 12'd0;
    else if(v_cnt >= V_FP + V_SYNC + V_BP - 1)
        active_y <= v_cnt - (V_FP[11:0] + V_SYNC[11:0] + V_BP[11:0] - 12'd1);
```

```
    else
        active_y <= active_y;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        v_cnt <= 12'd0;
    else if(h_cnt == H_FP  - 1)
        if(v_cnt == V_TOTAL - 1)
            v_cnt <= 12'd0;
        else
            v_cnt <= v_cnt + 12'd1;
    else
        v_cnt <= v_cnt;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        hs_reg <= 1'b0;
    else if(h_cnt == H_FP - 1)
        hs_reg <= 1'b1;
    else if(h_cnt == H_FP + H_SYNC - 1)
        hs_reg <= 1'b0;
    else
        hs_reg <= hs_reg;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        h_active <= 1'b0;
    else if(h_cnt == H_FP + H_SYNC + H_BP - 1)
        h_active <= 1'b1;
    else if(h_cnt == H_TOTAL - 1)
        h_active <= 1'b0;
    else
        h_active <= h_active;
end

always@(posedge clk or posedge rst)
```

```
begin
    if(rst)
        vs_reg <= 1'd0;
    else if((v_cnt == V_FP - 1) && (h_cnt == H_FP - 1))
        vs_reg <= 1'b1;
    else if((v_cnt == V_FP + V_SYNC - 1) && (h_cnt == H_FP - 1))
        vs_reg <= 1'b0;
    else
        vs_reg <= vs_reg;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        v_active <= 1'd0;
    else if((v_cnt == V_FP + V_SYNC + V_BP - 1) && (h_cnt == H_FP - 1))
        v_active <= 1'b1;
    else if((v_cnt == V_TOTAL - 1) && (h_cnt == H_FP - 1))
        v_active <= 1'b0;
    else
        v_active <= v_active;
end

reg [20:0] R_h_pos,R_v_pos;
reg [2:0] R_state;
always @(posedge clk) begin
    if(active_x > 464+R_h_pos && active_x < 815+R_h_pos &&  active_y > 199+R_v_pos &&
active_y < 535+R_v_pos) begin
        rom_addr <= (active_x - 465+R_h_pos) + (active_y - 200+R_v_pos) * 350 + 1;
    end
    else begin
        rom_addr <= 0;
    end
end

always@(posedge clk) begin
    if(active_x > 464+R_h_pos && active_x < 815+R_h_pos &&  active_y > 199+R_v_pos &&
active_y < 535+R_v_pos) begin
            rgb_r_reg <= rom_out[23:16];
            rgb_g_reg <= rom_out[15:8];
            rgb_b_reg <= rom_out[7:0];
    end
```

```
    else begin
        rgb_r_reg <= 0;
        rgb_g_reg <= 0;
        rgb_b_reg <= 0;
    end
end
```

進階題:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2024/05/09 15:51:06
// Design Name:
// Module Name: dataTimming
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module dataTimming(
    input clk,
    input rst,
    output hs,
    output vs,
    output de,
    output[7:0] rgb_r,
    output[7:0] rgb_g,
    output[7:0] rgb_b
);

parameter H_ACTIVE = 16'd1280;
parameter H_FP = 16'd110;
parameter H_SYNC = 16'd40;
parameter H_BP = 16'd220;
parameter V_ACTIVE = 16'd720;
parameter V_FP  = 16'd5;
parameter V_SYNC  = 16'd5;
parameter V_BP  = 16'd20;
```

```
parameter mem_w             = 9'd350;
parameter mem_h             = 9'd335;

parameter H_TOTAL = H_ACTIVE + H_FP + H_SYNC + H_BP;
parameter V_TOTAL = V_ACTIVE + V_FP + V_SYNC + V_BP;

reg hs_reg;
reg vs_reg;
reg hs_reg_d0;

reg vs_reg_d0;
reg[11:0] h_cnt;
reg[11:0] v_cnt;
reg[11:0] active_x;
reg[11:0] active_y;
reg[7:0] rgb_r_reg;
reg[7:0] rgb_g_reg;
reg[7:0] rgb_b_reg;
reg h_active;
reg v_active;
wire video_active;
reg video_active_d0;
assign hs = hs_reg_d0;
assign vs = vs_reg_d0;
assign video_active = h_active & v_active;
assign de = video_active_d0;
assign rgb_r = rgb_r_reg;
assign rgb_g = rgb_g_reg;
assign rgb_b = rgb_b_reg;

reg [16:0] rom_addr = 0;
wire [23:0] rom_out;

blk_mem_gen_0 rom_0(
    .addra(rom_addr),
    .clka(clk),
    .douta(rom_out)
);


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

```
    if(rst)
        begin
            hs_reg_d0 <= 1'b0;
            vs_reg_d0 <= 1'b0;
            video_active_d0 <= 1'b0;
        end
    else
        begin
            hs_reg_d0 <= hs_reg;
            vs_reg_d0 <= vs_reg;
            video_active_d0 <= video_active;
        end
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        h_cnt <= 12'd0;
    else if(h_cnt == H_TOTAL - 1)
        h_cnt <= 12'd0;
    else
        h_cnt <= h_cnt + 12'd1;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        active_x <= 12'd0;
    else if(h_cnt >= H_FP + H_SYNC + H_BP - 1)
        active_x <= h_cnt - (H_FP[11:0] + H_SYNC[11:0] + H_BP[11:0] - 12'd1);
    else
        active_x <= active_x;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        active_y <= 12'd0;
    else if(v_cnt >= V_FP + V_SYNC + V_BP - 1)
        active_y <= v_cnt - (V_FP[11:0] + V_SYNC[11:0] + V_BP[11:0] - 12'd1);
    else
        active_y <= active_y;
```

```
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        v_cnt <= 12'd0;
    else if(h_cnt == H_FP  - 1)
        if(v_cnt == V_TOTAL - 1)
            v_cnt <= 12'd0;
        else
            v_cnt <= v_cnt + 12'd1;
    else
        v_cnt <= v_cnt;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        hs_reg <= 1'b0;
    else if(h_cnt == H_FP - 1)
        hs_reg <= 1'b1;
    else if(h_cnt == H_FP + H_SYNC - 1)
        hs_reg <= 1'b0;
    else
        hs_reg <= hs_reg;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        h_active <= 1'b0;
    else if(h_cnt == H_FP + H_SYNC + H_BP - 1)
        h_active <= 1'b1;
    else if(h_cnt == H_TOTAL - 1)
        h_active <= 1'b0;
    else
        h_active <= h_active;
end

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

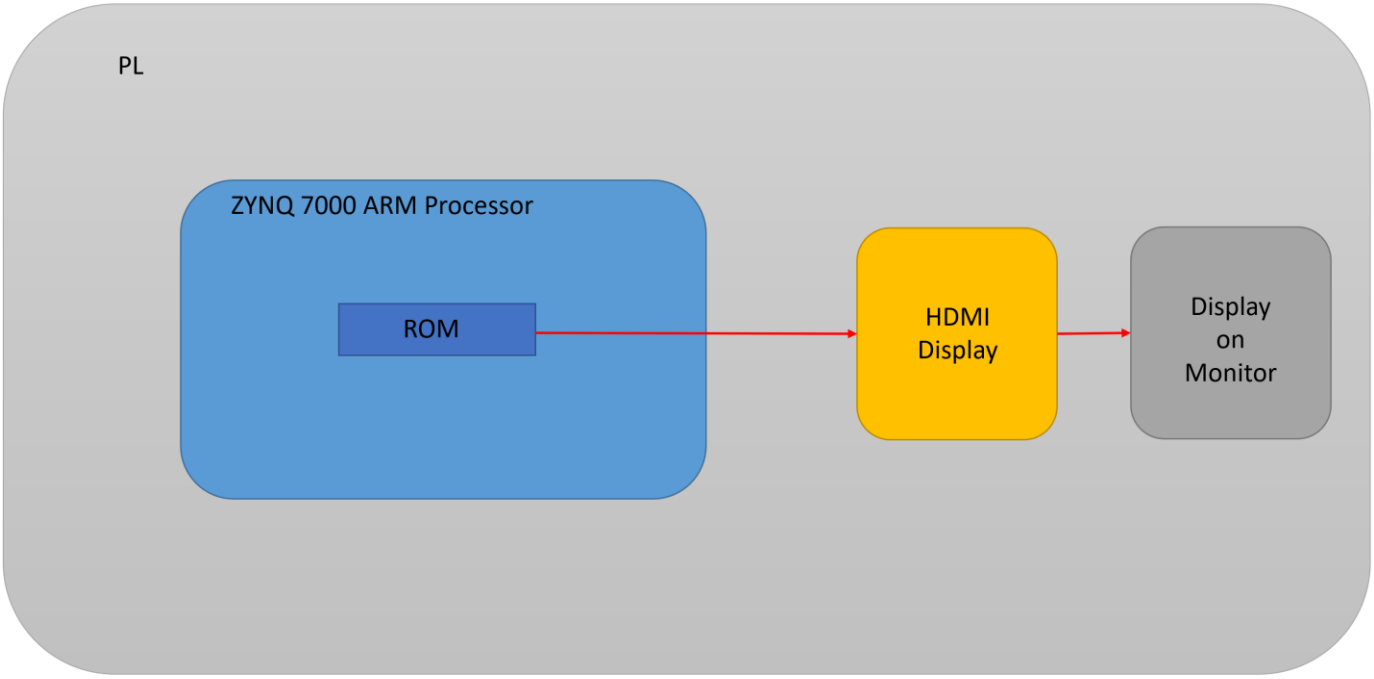
```
        vs_reg <= 1'd0;
    else if((v_cnt == V_FP - 1) && (h_cnt == H_FP - 1))
        vs_reg <= 1'b1;
    else if((v_cnt == V_FP + V_SYNC - 1) && (h_cnt == H_FP - 1))
        vs_reg <= 1'b0;
    else
        vs_reg <= vs_reg;
end

always@(posedge clk or posedge rst)
begin
    if(rst)
        v_active <= 1'd0;
    else if((v_cnt == V_FP + V_SYNC + V_BP - 1) && (h_cnt == H_FP - 1))
        v_active <= 1'b1;
    else if((v_cnt == V_TOTAL - 1) && (h_cnt == H_FP - 1))
        v_active <= 1'b0;
    else
        v_active <= v_active;
end

reg [20:0] R_h_pos,R_v_pos;
reg [2:0] R_state;
always @(posedge clk) begin
    if(active_x > 464+R_h_pos && active_x < 815+R_h_pos &&  active_y > 199+R_v_pos &&
active_y < 535+R_v_pos) begin
        rom_addr <= (active_x - 465+R_h_pos) + (active_y - 200+R_v_pos) * 350 + 1;
    end
    else begin
        rom_addr <= 0;
    end
end

always@(posedge clk) begin
    if(active_x > 464+R_h_pos && active_x < 815+R_h_pos &&  active_y > 199+R_v_pos &&
active_y < 535+R_v_pos) begin
            rgb_r_reg <= rom_out[23:16];
            rgb_g_reg <= rom_out[15:8];
            rgb_b_reg <= rom_out[7:0];
    end
    else begin
        rgb_r_reg <= 0;
```

```
        rgb_g_reg <= 0;
        rgb_b_reg <= 0;
    end
end

/*always@(posedge clk)begin
    if(en_l) R_h_pos <= R_h_pos - 50;
    else if(en_r) R_h_pos <= R_h_pos + 50 ;
    else if(en_u) R_v_pos <= R_v_pos + 50 ;
    else if(en_d) R_v_pos <= R_v_pos - 50 ;
    else begin
        R_h_pos <= 0;
        R_v_pos <= 0;
    end
end*/

reg R_vs_reg1,R_vs_reg2;
wire W_vs_neg;
wire O_vs;
assign O_vs =   (v_cnt < V_SYNC) ? 1'b0 : 1'b1;
always @(posedge clk or posedge rst)
begin
    if(rst)
        begin
            R_vs_reg1   <=  1'b0        ;
            R_vs_reg2   <=  1'b0        ;
        end
    else
        begin
            R_vs_reg1   <=  O_vs        ;
            R_vs_reg2   <=  R_vs_reg1   ;
        end
end
assign W_vs_neg = ~R_vs_reg1 & R_vs_reg2 ;

always@(posedge clk or posedge rst)
begin
    if(rst)
        begin
            R_h_pos <=  21'd0   ;
            R_v_pos <=  21'd0   ;
            R_state <=  2'b00   ;
```

```
        end
    else if(W_vs_neg)
        begin
            case(R_state)

                2'b00: // ?�����k�贬?

                    begin
                        R_h_pos     <=  R_h_pos + 1 ;

                        if(R_h_pos + (mem_w*2) == H_ACTIVE) // �p�G�I��k?��

                            R_state <=  2'b01       ;
                        else
                            R_state <=  2'b00       ;
                    end

                2'b01: // ?������贬?

                    begin
                        R_h_pos     <=  R_h_pos - 1 ;

                        if(R_h_pos == 1) // �p�G�I��k?��

                            R_state <=  2'b00       ;
                        else
                            R_state <=  2'b01       ;
                    end
                default:R_state <=  2'b00           ;
            endcase
        end
end

endmodule
```

五、AXI 介面 IP

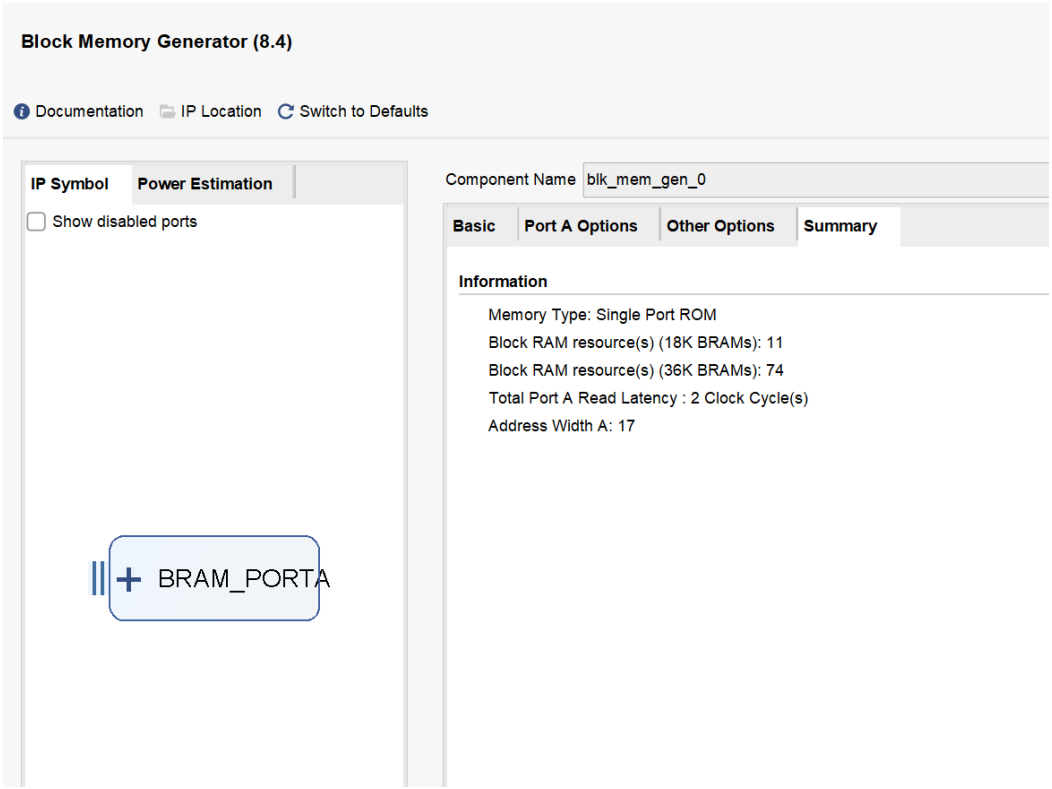
Interface:



ROM IP:

Width:24 bit

Depth:117250



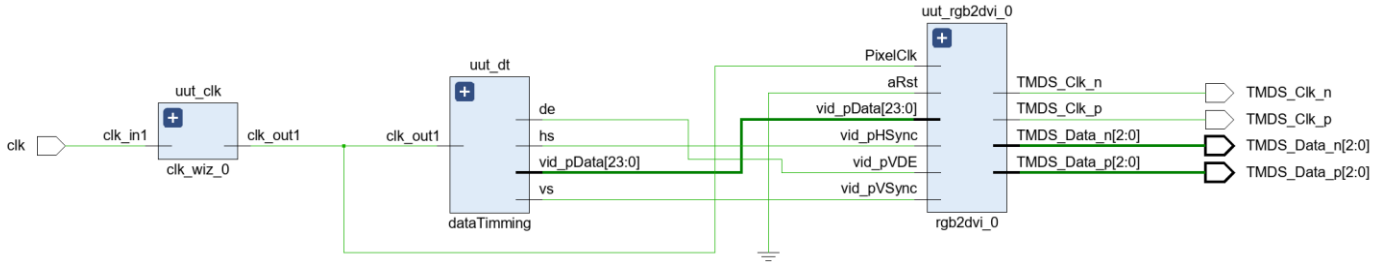
六、腳位定義

基礎題&進階題::

```
##HDMI Tx

set_property -dict { PACKAGE_PIN L17   IOSTANDARD TMDS_33  } [get_ports { TMDS_Clk_n }];
#IO_L11N_T1_SRCC_35 Sch=hdmi_tx_clk_n
set_property -dict { PACKAGE_PIN L16   IOSTANDARD TMDS_33  } [get_ports { TMDS_Clk_p }];
#IO_L11P_T1_SRCC_35 Sch=hdmi_tx_clk_p
set_property -dict { PACKAGE_PIN K18   IOSTANDARD TMDS_33  } [get_ports
{ TMDS_Data_n[0] }]; #IO_L12N_T1_MRCC_35 Sch=hdmi_tx_d_n[0]
set_property -dict { PACKAGE_PIN K17   IOSTANDARD TMDS_33  } [get_ports
{ TMDS_Data_p[0] }]; #IO_L12P_T1_MRCC_35 Sch=hdmi_tx_d_p[0]
set_property -dict { PACKAGE_PIN J19   IOSTANDARD TMDS_33  } [get_ports
{ TMDS_Data_n[1] }]; #IO_L10N_T1_AD11N_35 Sch=hdmi_tx_d_n[1]
set_property -dict { PACKAGE_PIN K19   IOSTANDARD TMDS_33  } [get_ports
{ TMDS_Data_p[1] }]; #IO_L10P_T1_AD11P_35 Sch=hdmi_tx_d_p[1]
set_property -dict { PACKAGE_PIN H18   IOSTANDARD TMDS_33  } [get_ports
{ TMDS_Data_n[2] }]; #IO_L14N_T2_AD4N_SRCC_35 Sch=hdmi_tx_d_n[2]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD TMDS_33  } [get_ports
{ TMDS_Data_p[2] }]; #IO_L14P_T2_AD4P_SRCC_35 Sch=hdmi_tx_d_p[2]
set_property -dict { PACKAGE_PIN R19   IOSTANDARD LVCMOS33 } [get_ports { HDMI_OEN[0] }];
#IO_0_34 Sch=hdmi_tx_hpdn

##Clock
set_property -dict { PACKAGE_PIN H16   IOSTANDARD LVCMOS33 } [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];
```

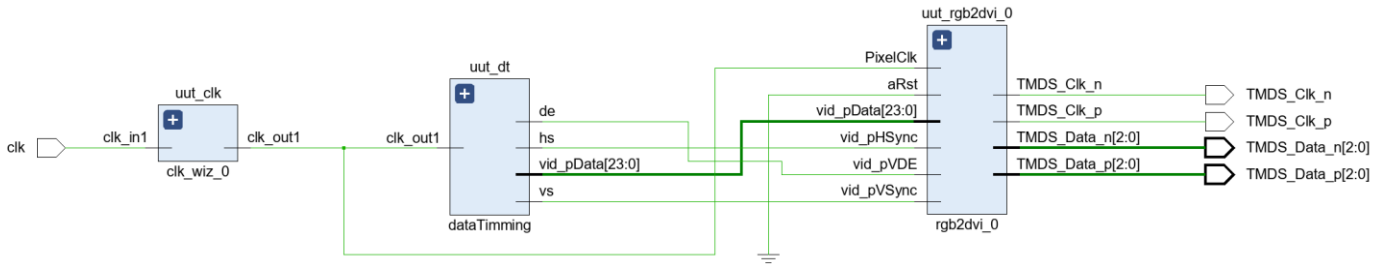
七、執行結果

基礎題:



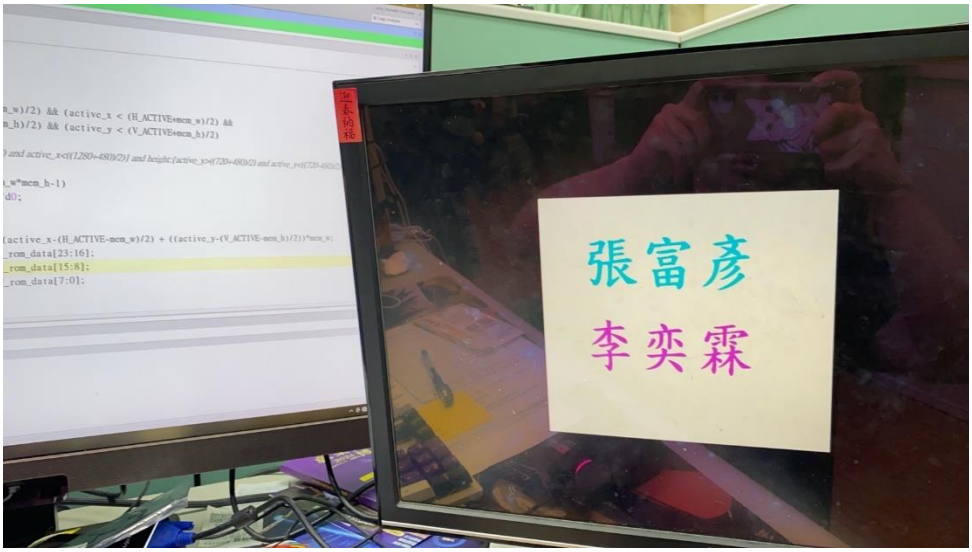
合成後電路

進階題:



合成後電路

基礎題執行結果:



進階題可由此連結觀看執行結果:

https://www.youtube.com/watch?v=4GImAh7SSdw

八、心得

李奕霖:

本次作業首先是如何用 MATLAB 把圖片轉成.coe 檔，然後將.coe 檔放入 ROM IP 內進行存取，接著就是這次較為困難的地方了，如何做 HDMI 的顯示，這邊困擾我們好久，首先我們是使用助教給的 color bar 的 IP 去觀察 HDMI 如何的顯示在螢幕上，接著觀察它的掃瞄方式來印出我們假設的圖形，例如黑白色的方塊圖等。然後再開始印出我們的名字，這邊就遇到了相當大的問題，因為我們是用 ROM IP 所以導致時序會慢一個 Cycle，這也導致我們顯示會無法鎖住畫面會不斷的掃描，造成嚴重破圖。一開始不知道這個狀況，但後面詢問過我們的指導老師蕭宇宏老師後才了解原來是這個狀況所造成這樣的結果，讓我們學到了寶貴的一課。

移動的部分，我們使用自行設置的一個脈衝去控制移動的狀態機，但效果有點不太理想，但有達到老師要的標準，這我們後續有機會在進行修正及探討。

張富彥:

這次作業首先是如何用 MATLAB 把圖片轉成.coe 檔，然後將.coe 檔放入 ROM IP 內進行存取，前面我們就想了一下要如何去做，畢竟是第 1 次做這個，也學到其實有相當多的方法把圖片轉成.coe 檔，後面需要看 HDMI 的 SPEC 來了解助教的 IP 在做什麼，接著就是這次較為困難的地方了，如何做 HDMI 的顯示，這邊困擾我們一陣子，因為雖然知道 code 的動作，但要如何使用是一大問題，首先我們是使用助教給的 color bar 的 IP 去觀察 HDMI 如何的顯示在螢幕上，接著觀察它的掃瞄方式來印出我們假設的圖形，例如彩條圖、黑白色的方塊圖等。然後再開始印出我們的名字，這邊就遇到了相當大的問題，我們的結果是嚴重破圖，後面得知因為我們是用 ROM IP 所以導致時序會慢一個 Cycle，這也導致我們顯示會無法鎖住畫面會不斷的掃描。後面詢問過我們的指導老師蕭宇宏老師後才了解原來是這個狀況所造成這樣的結果，也讓我們學到了寶貴的一課。

移動的部分，我們使用自行設置的一個脈衝去控制移動的狀態機，但效果有點不太理想，但有達到老師要的標準，這邊我們後續會在進行修正及探討，看看是哪方面出了問題。