EECS 2070 02 Digital Design Labs 2019

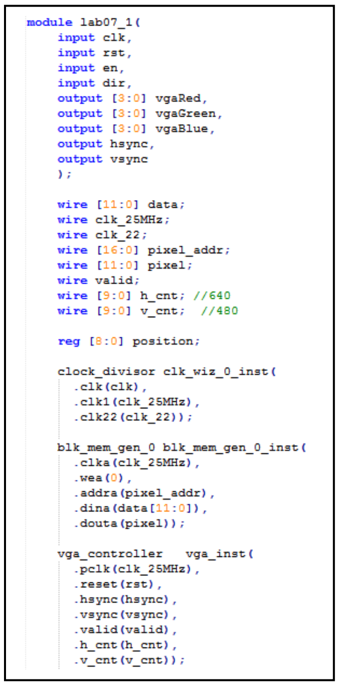
Lab 7

學號:107062361　姓名:許珉濠

1. 實作過程

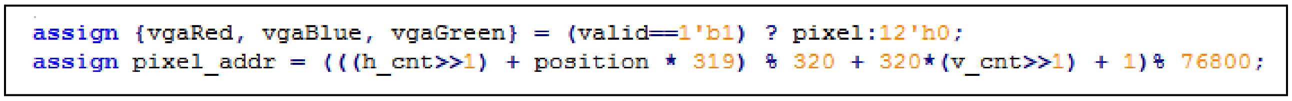
- **lab07_1**

  The picture which is shown below is designed based on **demo2** sample code. Since the **demo2** feature is scroll the picture from "**top to bottom**" recursively. In this lab part, we just need to scroll the picture from "**left to right**" or "**right to left**".



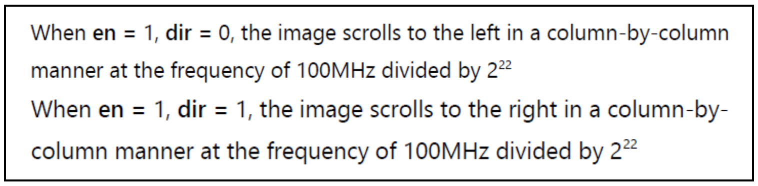
```
module lab07_1(
    input clk,
    input rst,
    input en,
    input dir,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output hsync,
    output vsync
    );

    wire [11:0] data;
    wire clk_25MHz;
    wire clk_22;
    wire [16:0] pixel_addr;
    wire [11:0] pixel;
    wire valid;
    wire [9:0] h_cnt; //640
    wire [9:0] v_cnt;  //480

    reg [8:0] position;

    clock_divisor clk_wiz_0_inst(
      .clk(clk),
      .clk1(clk_25MHz),
      .clk22(clk_22));

    blk_mem_gen_0 blk_mem_gen_0_inst(
      .clka(clk_25MHz),
      .wea(0),
      .addra(pixel_addr),
      .dina(data[11:0]),
      .douta(pixel));

    vga_controller   vga_inst(
      .pclk(clk_25MHz),
      .reset(rst),
      .hsync(hsync),
      .vsync(vsync),
      .valid(valid),
      .h_cnt(h_cnt),
      .v_cnt(v_cnt));
```

I use **pixel_addr** to scroll the picture according to **position**'s value. Since the requirement is scroll the picture horizontally, I just do addition of the **position** to **pixel_addr** without multiply it by 320 because the multiplication by 320 will modify the vertical side.



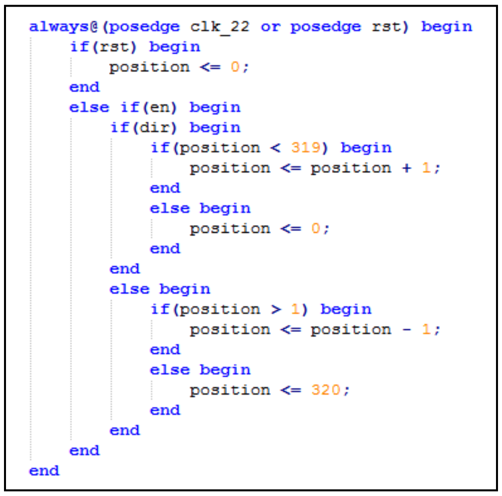
```
assign {vgaRed, vgaBlue, vgaGreen} = (valid==1'b1) ? pixel:12'h0;
assign pixel_addr = (((h_cnt>>1) + position * 319) % 320 + 320*(v_cnt>>1) + 1)% 76800;
```

With the always block below, the **position** will change based on **en** and **dir**.



When **en** = 1, **dir** = 0, the image scrolls to the left in a column-by-column manner at the frequency of 100MHz divided by 2^{22}

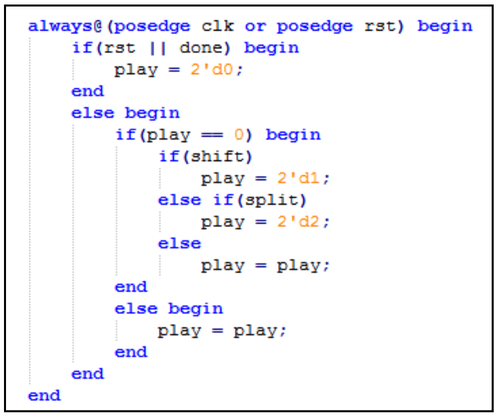
When **en** = 1, **dir** = 1, the image scrolls to the right in a column-by-column manner at the frequency of 100MHz divided by 2^{22}



```
always@(posedge clk_22 or posedge rst) begin
    if(rst) begin
        position <= 0;
    end
    else if(en) begin
        if(dir) begin
            if(position < 319) begin
                position <= position + 1;
            end
            else begin
                position <= 0;
            end
        end
        else begin
            if(position > 1) begin
                position <= position - 1;
            end
            else begin
                position <= 320;
            end
        end
    end
end
```

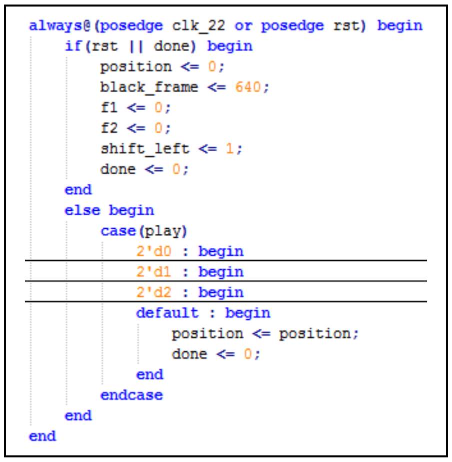
- **lab07_2**

  In this lab, the initial code is similar to **lab07_1**. The following picture is always block to change the state based on button. It will store the value in 2-bit register **play**. The initial **play**'s value is 0. The center button which is reset button will modify **play**'s value into 0. The shift and split button will only works if **play**'s value is 0. When the shift button is pressed, the **play**'s value will become 1. If the split button is pressed, the **play**'s value will become 2.



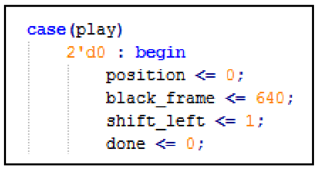
```
always@(posedge clk or posedge rst) begin
    if(rst || done) begin
        play = 2'd0;
    end
    else begin
        if(play == 0) begin
            if(shift)
                play = 2'd1;
            else if(split)
                play = 2'd2;
            else
                play = play;
        end
        else begin
            play = play;
        end
    end
end
```

For the always block below, it will modify image position, frame, flag, etc. When reset is performed, it will set each value to its initial value. For the shift feature, it mainly use **black_frame** and **shift_left**. Since it requires the image shift from right to left, and continued with shift top to bottom. I use **shift_left** as flag to determine the process of shift right to left is done. On the other hand, the **black_frame** will be the position of black frame. Its value is 640 due to shift is done from right to left. For **position**, **f1**, and **f2** are the position of image, frame 1 to move up-down, and frame 2 to move left-right respectively.



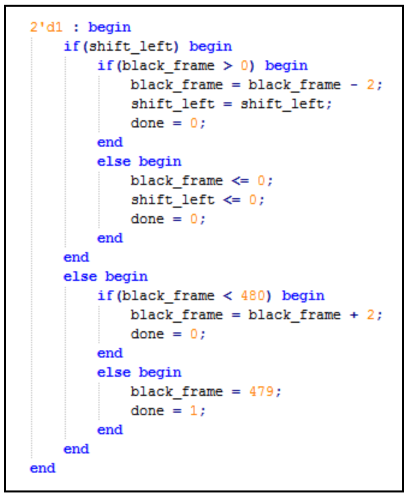
```
always@(posedge clk_22 or posedge rst) begin
    if(rst || done) begin
        position <= 0;
        black_frame <= 640;
        f1 <= 0;
        f2 <= 0;
        shift_left <= 1;
        done <= 0;
    end
    else begin
        case(play)
            2'd0 : begin
            2'd1 : begin
            2'd2 : begin
            default : begin
                position <= position;
                done <= 0;
            end
        endcase
    end
end
```

When the play value is 0, I set the value which is similar to reset in order to avoid error.



```
case(play)
    2'd0 : begin
        position <= 0;
        black_frame <= 640;
        shift_left <= 1;
        done <= 0;
```

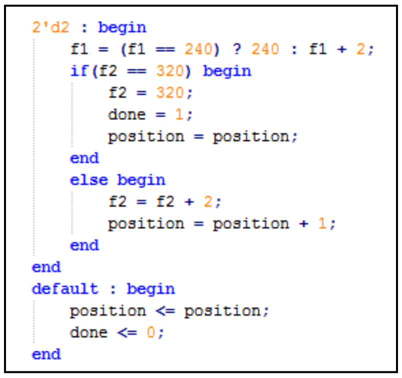
If the **shift_left** value is 1, it will perform shift from right to left. Then, the **black_frame** will be reduced by 2 repetitively due to adjust the speed in demo video. After **black_frame** is 0, assign the **shift_left** as 0 to start the shift top-bottom.



```
2'd1 : begin
    if(shift_left) begin
        if(black_frame > 0) begin
            black_frame = black_frame - 2;
            shift_left = shift_left;
            done = 0;
        end
        else begin
            black_frame <= 0;
            shift_left <= 0;
            done = 0;
        end
    end
    else begin
        if(black_frame < 480) begin
            black_frame = black_frame + 2;
            done = 0;
        end
        else begin
            black_frame = 479;
            done = 1;
        end
    end
end
```

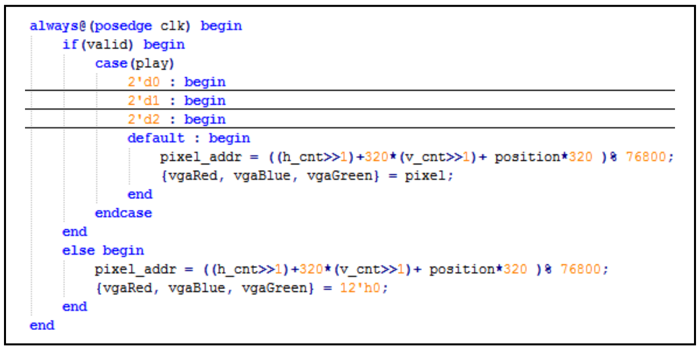
Afterwards, **black_frame** is increased by 2 repetitively until it reach the vertical maximum size which is 480. Therefore, **done** is assigned to 1 to indicate the shift is finished. If **done** value is 1, it will reset the value to initial condition.

Concerning to **play**'s value is 2, **f1**'s incremention is limited to 240 since it handles vertical frame and **f2**'s incremention is limited to 320 because it handles the horizontal frame. If **f2**'s value is 320, the done is assigned to 1 because the last frame that is shifted is **f2**.



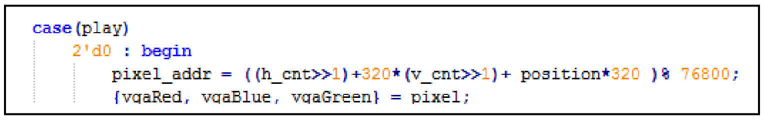
```
2'd2 : begin
    f1 = (f1 == 240) ? 240 : f1 + 2;
    if(f2 == 320) begin
        f2 = 320;
        done = 1;
        position = position;
    end
    else begin
        f2 = f2 + 2;
        position = position + 1;
    end
end
default : begin
    position <= position;
    done <= 0;
end
```

The picture below is the always block control the value of **pixel_addr** and {**vgaRed, vgaBlue, vgaGreen**} which are in charge to show or manipulate the output image.



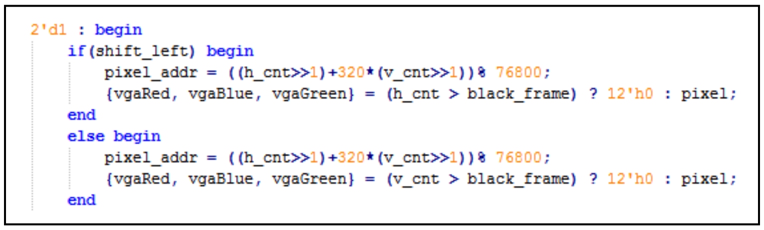
```
always@(posedge clk) begin
    if(valid) begin
        case(play)
            2'd0 : begin
            2'd1 : begin
            2'd2 : begin
            default : begin
                pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+ position*320 )% 76800;
                {vgaRed, vgaBlue, vgaGreen} = pixel;
            end
        endcase
    end
    else begin
        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+ position*320 )% 76800;
        {vgaRed, vgaBlue, vgaGreen} = 12'h0;
    end
end
```

The output image will be shown same as the original image if the **play**'s value is 0.



```
case(play)
    2'd0 : begin
        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+ position*320 )% 76800;
        {vgaRed, vgaBlue, vgaGreen} = pixel;
```

In the shift image feature, **pixel_addr** is assigned with **"(h_cnt>>1)+320*(v_cnt>>1)"** since the output image does not need to move and it only needs black frame moves.

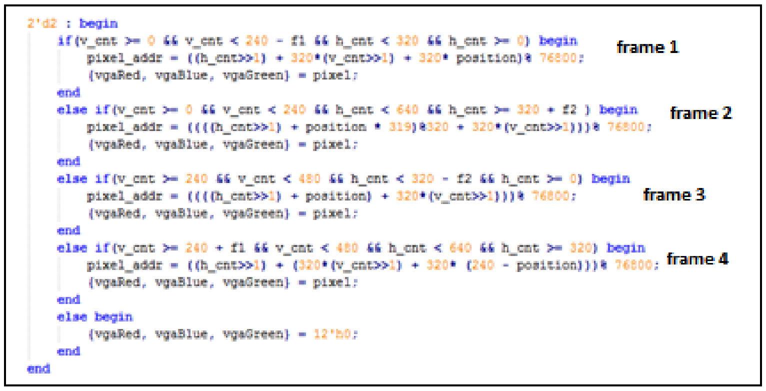


```
2'd1 : begin
    if(shift_left) begin
        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))% 76800;
        {vgaRed, vgaBlue, vgaGreen} = (h_cnt > black_frame) ? 12'h0 : pixel;
    end
    else begin
        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))% 76800;
        {vgaRed, vgaBlue, vgaGreen} = (v_cnt > black_frame) ? 12'h0 : pixel;
    end
```

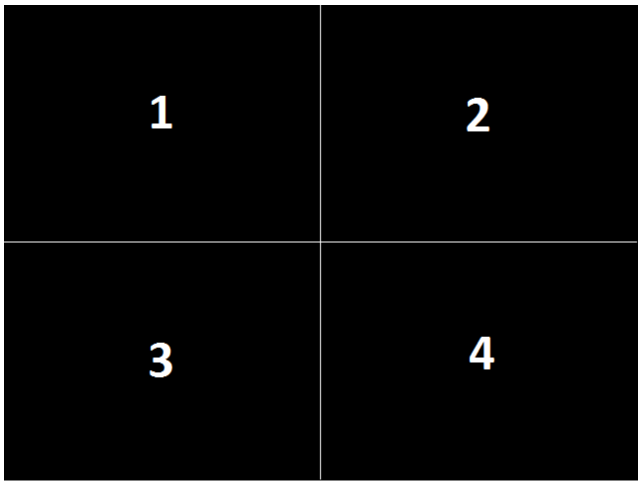
In shift right-left case, {**vgaRed, vgaBlue, vgaGreen**} is assigned with **12'h0** to show black frame for the horizontal image frame that is bigger than **black_frame** value.

For the opposite statement, {**vgaRed, vgaBlue, vgaGreen**} is assigned with **12'h0** for the vertical image frame that is bigger than **black_frame**.

Pertaining to split image feature, the two frame **f1** and **f2** will limit the vertical and horizontal frame respectively.



```
2'd2 : begin
    if(v_cnt >= 0 && v_cnt < 240 - f1 && h_cnt < 320 && h_cnt >= 0) begin
        pixel_addr = ((h_cnt>>1) + 320*(v_cnt>>1) + 320* position)% 76800;
        {vgaRed, vgaBlue, vgaGreen} = pixel;
    end
    else if(v_cnt >= 0 && v_cnt < 240 && h_cnt < 640 && h_cnt >= 320 + f2 ) begin
        pixel_addr = ((((h_cnt>>1) + position * 319)%320 + 320*(v_cnt>>1)))% 76800;
        {vgaRed, vgaBlue, vgaGreen} = pixel;
    end
    else if(v_cnt >= 240 && v_cnt < 480 && h_cnt < 320 - f2 && h_cnt >= 0) begin
        pixel_addr = ((((h_cnt>>1) + position) + 320*(v_cnt>>1)))% 76800;
        {vgaRed, vgaBlue, vgaGreen} = pixel;
    end
    else if(v_cnt >= 240 + f1 && v_cnt < 480 && h_cnt < 640 && h_cnt >= 320) begin
        pixel_addr = ((h_cnt>>1) + (320*(v_cnt>>1) + 320* (240 - position)))% 76800;
        {vgaRed, vgaBlue, vgaGreen} = pixel;
    end
    else begin
        {vgaRed, vgaBlue, vgaGreen} = 12'h0;
    end
end
```



The number is the frame number

In the first if statement, it will filter the frame which horizontal size is 0 to 319, and vertical size is from 0 to 239 – f1. The f1 is incremented until it reach 240. In order to move the image, **position** will be assigned with position on vertical side of **pixel_addr**. Thus, output image will move to top.

For the first else-if statement, it will filter the frame which horizontal size is between 320 + f2 to 639, and vertical size is from 0 to 239. The f2 is incremented until it reach 320. In order to move the image, **position** will be assigned with position on horizontal side of **pixel_addr**. Thus, output image will move to right.

About the second else-if statement, it will filter the frame which horizontal size is between

0 to 319 – f2, and vertical size is from 240 to 479. The f2 is incremented until it reach 320. In order to move the image, **position** will be assigned with position on horizontal side of **pixel_addr**. Thus, output image will move to left.

As regards to last statement, it will filter the frame which horizontal size is 320 to 639, and vertical size is from 240 + f1 to 479. The f1 is incremented until it reach 240. In order to move the image, (240 – **position)** will be assigned with position on vertical side of **pixel_addr**. Thus, output image will move to down.

2. 學到的東西與遇到的困難

The things that I learn in this lab is how to show and modify image in monitor through FPGA board. The things that is made are several animation from Powerpoint's Animations such as float in, fly in, split, etc. When I design the code, there are a lot problem occurs in lab07_2 like the image can only be shifted around 80% of the screen. In the end, I found it turns out that the register **black_frame** that I declared is too small. It only can hold 512 bit, whereas the screen size is 640 bit.

3. 想對老師或助教說的話

沒有。