



รายงาน

Pacman report

โดย

- | | |
|--------------------------------|--------------------------|
| 1.นายจากรุกิตติ์ ปานเอี่ยม | รหัสนักศึกษา 65070501006 |
| 2.นายจิรภัทร ด่อล่าห์ | รหัสนักศึกษา 65070501007 |
| 3.นางสาวนันทิกานต์ สกุลเนตร | รหัสนักศึกษา 65070501034 |
| 4.นายภูริภัทร อภิรักษ์โชคิติมา | รหัสนักศึกษา 65070501043 |

นักศึกษาชั้นปีที่ 2

เสนอ

ผศ.สันนิษฐ์ สระแก้ว

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา CPE 222

DIGITAL ELECTRONICS AND LOGIC DESIGN

คณะวิศวกรรมศาสตร์ สาขาวิศว์คอมพิวเตอร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

Pacman

บทนำ

เกมเป็นสื่อบันเทิงออนไลน์ที่ได้รับความนิยมอย่างแพร่หลายไม่ว่าจะทั้งเด็กหรือผู้ใหญ่ ผู้ชายหรือผู้หญิง เพราะเนื่องด้วยเกมนั้นมีคลายประเภททำให้สามารถรับผู้เล่นได้อย่างมากมายและต้นทางที่ทำให้เราไม่สามารถปฏิเสธได้เลยว่าเกมเป็นหนึ่งในอุตสาหกรรมที่ใหญ่เป็นอันดับต้นๆของโลกซึ่งมุ่ลค่าของตลาดนั้นมาจากการเป็นอย่างมากโดยเกมนั้นเป็นสื่อบันเทิงอย่างนึงที่อยู่กับเรามาอย่างช้านานซึ่งหนึ่งในเกมที่ถ้าหากเราพูดขึ้นไปบอกได้เลยว่า 9 ใน 10 จะรู้จักเป็นอย่างแน่นอนนั้นคือ

เกม Pacman

หากเราในฐานะนักศึกษาของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธุรังษ์ฯ โภคโนโลยีและความรู้ด้าน Hardware ที่อาจารย์สอนมาใช้ในการพัฒนาเกม “Pacman” เพื่อพัฒนาความรู้ และฝึกฝนฝีมือเพื่อนักศึกษาสามารถนำประสบการณ์ที่ได้รับจากโครงการในครั้งนี้ไปต่อยอดเพื่อ สร้างเกมที่สามารถ พัฒนาและยกระดับวงการเกมในประเทศไทยให้ดียิ่งขึ้น

วัตถุประสงค์

- เพื่อนำความรู้ที่ได้จากรายวิชา CPE 222 มาต่อยอดให้ก่อเกิดประสบการณ์เพื่อนำไปใช้ในอนาคต
- เพื่อสร้างเกมที่สามารถเป็นสื่อบันเทิงเพื่อเพิ่มความสนุกสนานให้แก่ผู้เล่นได้
- เพื่อสร้างเกมที่สามารถเป็นสื่อที่สามารถให้ผู้เล่นนำไปสร้างสายสัมพันธ์อันดีกับเพื่อนๆได้

ขอบเขต

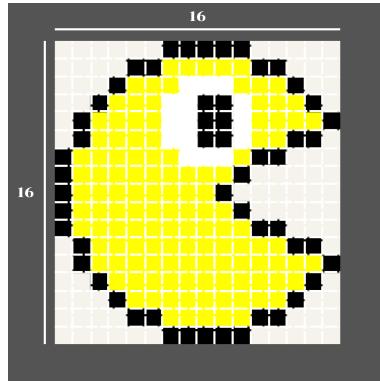
- รูปแบบเกมที่้าวอิงจากเกม Pacman
- การบังคับเคลื่อนที่โดยใช้บอร์ด FPGA Basy3 : Xilinx Artix - 7 และ บูม BtnL BtnR BtnU BtnD ในการบังคับการเคลื่อนที่ ของ Pacman
- งบประมาณการทำ project 0 บาท
- Coding โดย ใช้ ภาษา verilog ในการเขียน
- ใช้ภาษา Python ในการ generate rom memory

วัสดุอุปกรณ์

- บอร์ด FPGA Basy3 : Xilinx Artix - 7
- จอแสดงผลที่รองรับ VGA Port
- สายเชื่อมต่อ USB to micro USB

Rom :

ในส่วนของการแสดงผล เราเก็บรายละเอียดของภาพในเกมทั้งหมดในรูปแบบของ ROM เพื่อให้ง่ายต่อการเรียกใช้ โดยการเก็บรูปแบบนี้เป็นการแบ่งรูปภาพเป็น pixel และกำหนดสีของแต่ละ pixel ให้ออกมาเป็นภาพใหญ่



โดยเราจะเก็บค่าสีในเลข 12 Bit ไว้ใน template เดียว กันหมดดังรูปภาพต่อไปนี้ :

```
module rom
(
    input wire clk,
    input wire [3:0] row,
    input wire [3:0] col,
    output reg [11:0] color_data
);

(* rom_style = "block" *)

//signal declaration
reg [3:0] row_reg;
reg [3:0] col_reg;

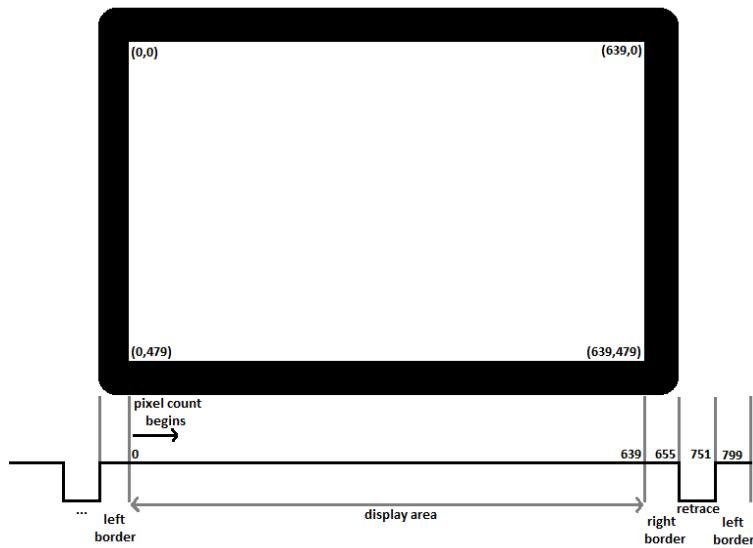
always @ (posedge clk)
begin
    row_reg <= row;
    col_reg <= col;
end

always @*
case ({row_reg, col_reg})
    6'b000000: color_data = 12'b011011011110;
    ...
    default: color_data = 12'b00000000000000;
endcase
endmodule
```

การแสดงผลหน้าจอ Display :

การแสดงผลของเกมเรามีความกว้างอยู่ที่ 640 pixel และความสูงอยู่ที่ 480 pixel ซึ่งหมายความว่ามีทั้งหมด 307,200 pixel แน่นอน ว่าการกำหนดสีของทั้งหมดจะทำให้เสียเวลาและไม่มีความแม่นยำ เราจึงใช้ python แปลงรูปภาพที่เราต้องการออกมานเป็นโค้ดเพื่อประหยัดเวลา

การบอกตำแหน่งของแต่ละ pixel เราจะใช้ x และ y เป็นตัวบอกตำแหน่ง โดยที่ pixel บนซ้าย (pixel แรก) คือตำแหน่งที่ (0,0) ซึ่ง การที่จะทำงานในแต่ละ pixel เราจำเป็นต้องสร้าง module VGA เพื่อช่วยบอกตำแหน่ง (output) ของ pixel ที่เรากำลังทำงานอยู่ โดย VGA จะ มี output ออกมานเป็นตำแหน่งของ x และ y ที่เรากำลังทำงานอยู่



Python : To generate a rom memory

```
1  from PIL import Image
2
3  def convert_to_12_bit_color(pixel):
4      if isinstance(pixel, tuple):
5          # RGB mode
6          r, g, b = pixel
7      else:
8          # Grayscale mode
9          r, g, b = pixel, pixel, pixel
10
11     # Convert RGB values to 12-bit color
12     r = (r >> 4) & 0b1111
13     g = (g >> 4) & 0b1111
14     b = (b >> 4) & 0b1111
15     return (r << 8) | (g << 4) | b
16
17 def generate_rom_template(image_path):
18     # Open the image
19     img = Image.open(image_path)
20
21     # Convert to RGB mode if not in RGB
22     img = img.convert('RGB')
23
24     # Resize the image to 16x16 pixels (width x height)
25     img = img.resize((16, 16))
26
27     # Get pixel data
28     pixels = list(img.getdata())
29
30     # Generate ROM template
31     rom_template = ""
32     for i, pixel in enumerate(pixels):
33         color_data = convert_to_12_bit_color(pixel)
34         rom_template += f"8'b{i:08b}: color_data = 12'b{color_data:012b};\n"
35
36     return rom_template
37
38 # Example usage
39 image_path = "star.png"
40 rom_template = generate_rom_template(image_path)
41
42 # Save the ROM template to a file
43 with open("star_rom.v", "w") as f:
44     f.write(rom_template)
```

หลักการทำงาน

โค้ด Python นี้ทำหน้าที่ดำเนินการแปลงภาพให้กลายเป็นข้อมูลสีแบบ 12 บิตและสร้างไฟล์ ROM ในรูปแบบของไฟล์ Verilog ด้วยการใช้ไลบรารี PIL

(Python Imaging Library)

ฟังก์ชัน : convert_to_12_bit_color

```
3  def convert_to_12_bit_color(pixel):
4      if isinstance(pixel, tuple):
5          # RGB mode
6          r, g, b = pixel
7      else:
8          # Grayscale mode
9          r, g, b = pixel, pixel, pixel
10
11     # Convert RGB values to 12-bit color
12     r = (r >> 4) & 0b1111
13     g = (g >> 4) & 0b1111
14     b = (b >> 4) & 0b1111
15
16     return (r << 8) | (g << 4) | b
```

ทำหน้าที่ : แปลงค่าสีของแต่ละพิกเซลในรูปภาพเป็นเลข 12 บิต โดยการดึงค่าสีแดง (R), สีเขียว (G), และสีน้ำเงิน (B) จากพิกเซล ถ้ารูปภาพเป็นโหมด RGB ให้ใช้ค่าสี R, G, B จากพิกเซล แต่ถ้าเป็นโหมด Grayscale ให้ใช้ค่าสีเดียวกันทั้งสามสี จากนั้นทำการเปลี่ยนแปลงค่าสี R, G, B เป็นเลข 12 บิต และรวมค่าสีเป็นเลข 12 บิตและส่งคืนค่านั้นออกไป

ฟังก์ชัน : generate_rom_template

```
17  def generate_rom_template(image_path):
18      # Open the image
19      img = Image.open(image_path)
20
21      # Convert to RGB mode if not in RGB
22      img = img.convert('RGB')
23
24      # Resize the image to 16x16 pixels (width x height)
25      img = img.resize((16, 16))
26
27      # Get pixel data
28      pixels = list(img.getdata())
29
30      # Generate ROM template
31      rom_template = ""
32      for i, pixel in enumerate(pixels):
33          color_data = convert_to_12_bit_color(pixel)
34          rom_template += f"8'b{i:08b}: color_data = 12'b{color_data:012b};\n"
35
36
37  return rom_template
```

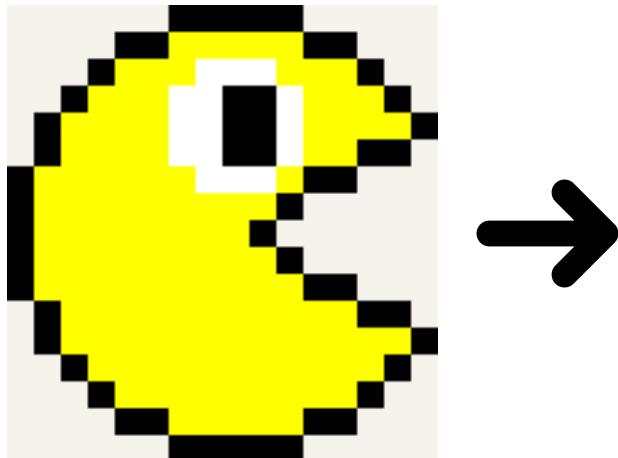
ทำหน้าที่ : รับที่อยู่ของไฟล์ภาพเป็นอินพุต โดยเปิดไฟล์ภาพด้วย PIL และแปลงเป็นโหมด RGB (ถ้ายังไม่ได้เป็น RGB) จากนั้นปรับขนาดของภาพให้มีขนาด 16x16 พิกเซล จากนั้นดึงข้อมูลของพิกเซลทั้งหมดในรูปภาพ และสร้างเทมเพลต ROM โดยวนลูปผ่านแต่ละพิกเซลเพื่อเก็บข้อมูลสีแบบ 12 บิตของแต่ละพิกเซลเข้าไปในเทมเพลต ROM ในรูปแบบของไฟล์ Verilog

ตัวอย่าง : การใช้งาน

```
37
38 # Example usage
39 image_path = "star.png"
40 rom_template = generate_rom_template(image_path)
41
42 # Save the ROM template to a file
43 with open("star_rom.v", "w") as f:
44     f.write(rom_template)
```

ในส่วนนี้มีการใช้งานตัวอย่างโดยกำหนดที่อยู่ของไฟล์ภาพ ("star.png") และเรียกใช้ฟังก์ชัน `generate_rom_template` เพื่อสร้างไฟล์เพลต ROM จากภาพดังกล่าว และบันทึกไฟล์เพลต ROM ลงในไฟล์ชื่อ "star_rom.v"

ตัวอย่าง : การแปลงรูป Pacman (16x16 pixels) เป็น code 12 bit



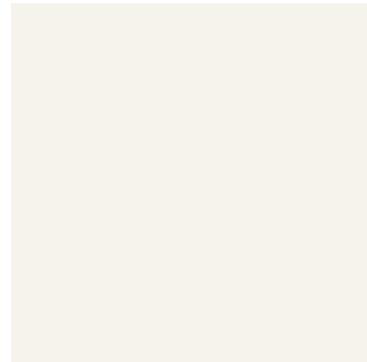
```
1  8'b00000000: color_data = 12'b111111111110;
2  8'b00000001: color_data = 12'b111111111110;
3  8'b00000010: color_data = 12'b111111111110;
4  8'b00000011: color_data = 12'b111111111110;
5  8'b00000100: color_data = 12'b110111011101;
6  8'b00000101: color_data = 12'b110011001011;
7  8'b00000110: color_data = 12'b001000100001;
8  8'b00000111: color_data = 12'b001000100000;
9  8'b00001000: color_data = 12'b001000100000;
10 8'b00001001: color_data = 12'b001000100000;
11 8'b00001010: color_data = 12'b001000100001;
12 8'b00001011: color_data = 12'b110011001011;
13 8'b00001100: color_data = 12'b110111011101;
14 8'b00001101: color_data = 12'b111111111110;
15 8'b00001110: color_data = 12'b111111111110;
16 8'b00001111: color_data = 12'b111111111110;
17 8'b00010000: color_data = 12'b111111111110;
```

Logic Design : module

vga_sync : module ที่ใช้สำหรับการแสดงผลหน้าจอ (Display) ผ่าน vga port โดย output จะเป็น ตำแหน่งแกน x และ y

Background_rom : เป็น module ที่เก็บ memory(ROM) สำหรับแสดงผล Background

ตัวอย่างภาพ Background :



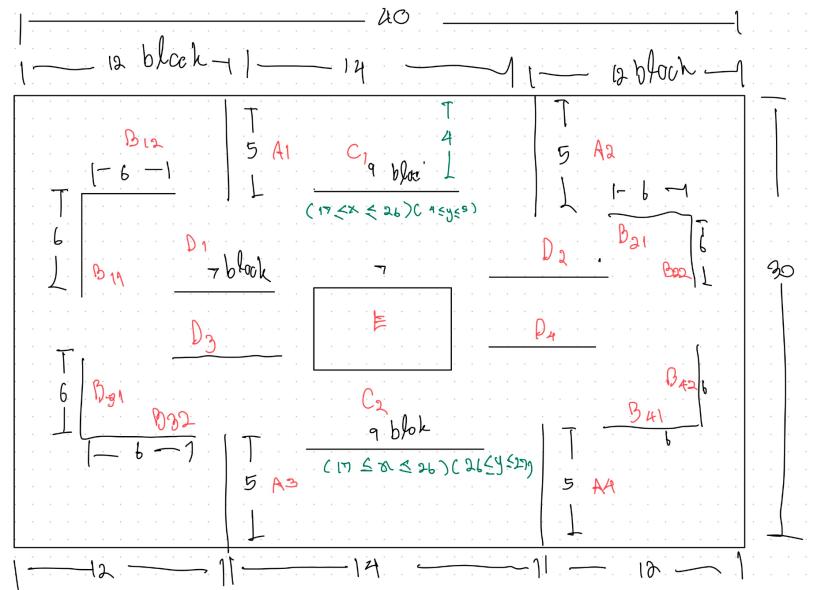
(256x256 pixels)



Walls_rom : เป็น module ที่เก็บ memory(ROM) สำหรับปิซเซ่ใน “Wall.v”

Walls : module สำหรับการแสดงผลกำแพง โดยใช้ Rom จาก “Walls_rom

ตัวอย่าง Code สำหรับ Walls : กำหนดตำแหน่งในการสร้าง wall ที่ตำแหน่งต่อไปนี้



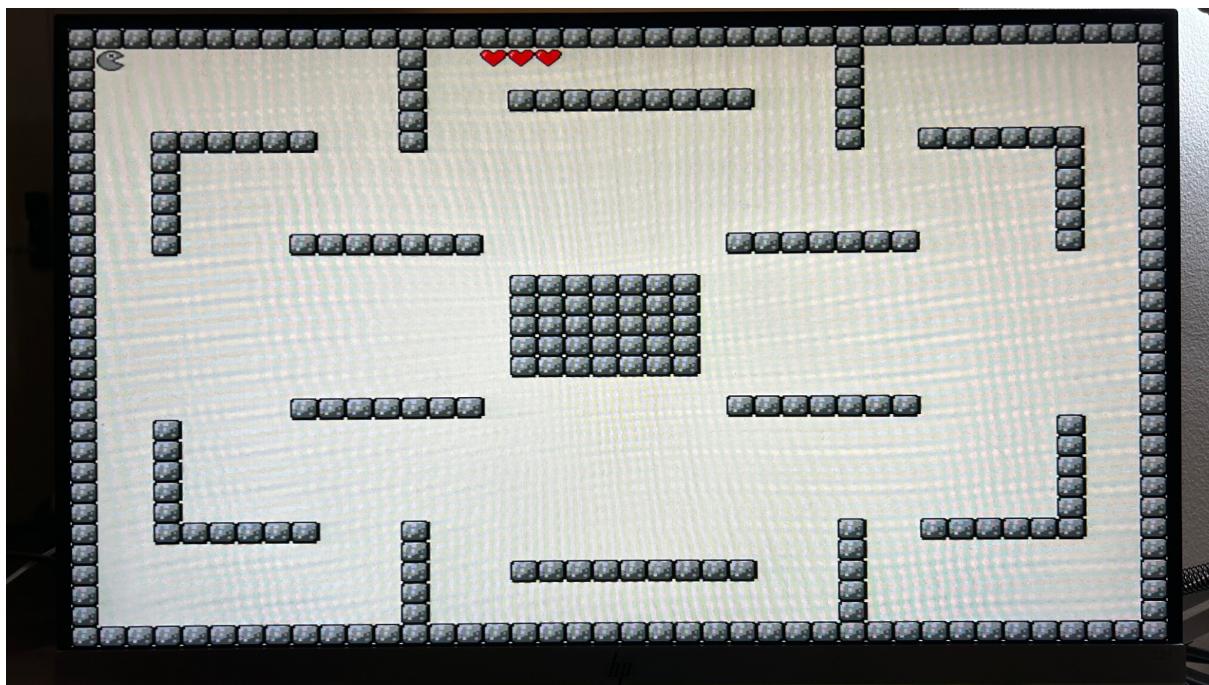
(รูปภาพแสดงตำแหน่งในการสร้าง Wall บนหน้าจอ)

```

9
10    reg [4:0] row;
11    reg [3:0] col;
12
13    wire [11:0] walls_color_data;
14
15    Walls_rom Walls_unit (.clk(clk), .row(row), .col(col), .color_data(walls_color_data));
16
17    always @*
18    begin
19        //default
20        rgb_out = 12'b000000000000;
21        walls_on = 0;
22        row = 0;
23        col = 0;
24
25    if(video_on)
26        begin
27
28            //top wall
29    if (y >= 0 && y <= 15)
30        begin
31            row = y;
32            col = x;
33    if(walls_color_data != 12'b011011011110)
34        begin
35            begin
36                rgb_out = walls_color_data;
37                walls_on = 1;
38            end

```

(ตัวอย่าง code ที่ใช้สร้างกำแพงด้านบนสุดของจอ y <= 15 คือขนาดของ pixel)



(ภาพแสดงตัวอย่างกำแพง)

Pacman_rom : เป็น module ที่เอาไว้เก็บ memory ของ pacman สำหรับนำไปแสดงผล

Pacman_control : เป็น module ที่ใช้สำหรับแสดงผล pacman(นำมาจาก “pacman_rom”) และควบคุมการเคลื่อนในแนวแกน x y

ตัวอย่าง : ถ้ากดปุ่ม button left ให้ลับตำแหน่งของ pacman ไป 1 pixel

```
180      left:
181          begin
182              if(time_reg > 0)
183                  time_next = time_reg - 1;
184              else if(time_reg == 0)
185                  begin
186                      if(s_x_reg >= 17 && !check_wall(s_x_reg - 1, s_y_reg))
187                          s_x_next = s_x_reg - 1;
188                      if(!btnL || btnR || btnU || btnD)
189                          begin
190                              motion_state_next = no_dir;
191                              start_next = 0;
192                          end
193                      if(start_reg > TIME_MIN) // If we have been moving for more than TIME_MIN
194                          begin
195                              time_next = start_reg - TIME_STEP;
196                              start_next = start_reg - TIME_STEP;
197                          end
198                      else
199                          begin
200                              time_next = start_reg;
201                              start_next = start_reg;
202                          end
203                  end
204          end
```

ตัวอย่าง : การคำนวน row กับ col เพื่อไปใช้แสดงผล pacman

```
287  /*************************************************************************/
288  /*          Pacman display area logic                                */
289  /*************************************************************************/
290 //Pacman display area boundaries
291 wire [3:0] row;
292 wire [3:0] col;
293
294 //current pixel coordinates - current sprite coordinates = pixel coordinates within pacman area
295 assign col = (dir_reg == LEFT && pacman_area) ? T_W - 1 - (x - s_x_reg) :
296             (dir_reg == RIGHT && pacman_area) ? x - s_x_reg :
297             (dir_reg == UP && pacman_area) ? x - s_x_reg :
298             (dir_reg == DOWN && pacman_area) ? x - s_x_reg : 0;
299
300 assign row = (dir_reg == LEFT && pacman_area) ? y - s_y_reg :
301             (dir_reg == RIGHT && pacman_area) ? y - s_y_reg :
302             (dir_reg == UP && pacman_area) ? y - s_y_reg :
303             (dir_reg == DOWN && pacman_area) ? y - s_y_reg : 0;
304
305 //Pacman rom
306 //vector for rom color_data output
307 wire [11:0] color_data_pacman, color_data_pacman_ghost;
308 Pacman_rom Pacman_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_pacman));
309 Pacman_Ghost_rom Pacman_ghost_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_pacman_ghost));
310
311 //vector to signal when vga_sync is in the pacman area
312 wire pacman_area;
313 assign pacman_area = (x >= s_x_reg && x <= s_x_reg + T_W - 1 && y >= s_y_reg && y <= s_y_reg + T_H - 1) ? 1 : 0;
```

Blue_Ghost_rom : เป็น module สำหรับเก็บ memory สำหรับแสดงผลผีเสื้า

Ghost_blue : เป็น module ที่ใช้ในการแสดงผลผีสีฟ้า(นำมาราบ “Blue_Ghost_rom”) และ เคลื่อนไหวขึ้นอยู่กับ ตำแหน่งของ pacman

ตัวอย่าง : การคำนวณเส้นทางการเดินต่อไปของผีสีฟ้า โดยอ้างอิงจากตำแหน่งของ pacman

- ถ้าผีอยู่ทางขวาของ pacman =>เคลื่อนที่ทางซ้าย
- ถ้าผีอยู่ทางซ้ายของ pacman =>เคลื่อนที่ทางขวา
- ถ้าผีอยู่ด้านบนของ pacman =>เคลื่อนที่ลง
- ถ้าผีอยู่ด้านล่างของ pacman =>เคลื่อนที่ขึ้น

```
92    always @*
93        begin
94            //default
95            dir_next = dir_reg;
96            if(pacman_x < s_x_reg)
97                |   dir_next = LEFT;
98            else if(pacman_x > s_x_reg)
99                |   dir_next = RIGHT;
100           else if(pacman_y < s_y_reg)
101               |   dir_next = UP;
102           else if(pacman_y > s_y_reg)
103               |   dir_next = DOWN;
104        end
```

(รูปภาพแสดงตัวอย่างการกำหนดการเคลื่อนของผี

ตัวอย่าง : การคำนวณ row กับ col เพื่อไปใช้แสดงผลผีสีฟ้า

```
162  ****
163  /*
164   *          Blue ghost display area logic
165   */
166  wire [3:0] row;
167  wire [3:0] col;
168
169  //current pixel coordinates - current sprite coordinates = pix coordinates within display area
170  assign col = (dir_reg == LEFT && ghost_blue_area) ? T_W - 1 - (x - s_x_reg) :
171      |   (dir_reg == RIGHT && ghost_blue_area) ? x - s_x_reg :
172      |   (dir_reg == UP && ghost_blue_area) ? x - s_x_reg :
173      |   (dir_reg == DOWN && ghost_blue_area) ? x - s_x_reg : 0;
174
175  assign row = (dir_reg == LEFT && ghost_blue_area) ? y - s_y_reg :
176      |   (dir_reg == RIGHT && ghost_blue_area) ? y - s_y_reg :
177      |   (dir_reg == UP && ghost_blue_area) ? y - s_y_reg :
178      |   (dir_reg == DOWN && ghost_blue_area) ? y - s_y_reg : 0;
179
180  //Blue ghost rom
181  wire [11:0] color_data_Blue_ghost;
182  Blue_Ghost_rom Blue_Ghost_rom_unit (.clk(clk), .row(row), .col(col), .color_data(color_data_Blue_ghost));
183
184  //check if pixel is within display area
185  wire ghost_blue_area;
186  assign ghost_blue_area = (x >= s_x_reg && x <= s_x_reg + T_W - 1 && y >= s_y_reg && y <= s_y_reg + T_H - 1) ? 1 : 0;
```

Red_Ghost_rom : เป็น module สำหรับเก็บ memory ในการแสดงผลผีสีแดง

Ghost_red : เป็น module ที่ใช้ในการแสดงผลผีสีแดง (นำมายจาก “Red_Ghost_rom”) และเคลื่อนไหวตามตำแหน่งของ pacman

Check_collision : เป็น module ที่ใช้ในการเช็คการชนกันของ pacman และผีทั้งสอง โดยจะคิดจากถ้า ตำแหน่งของ pacman เหลือมากัน 3 pixel

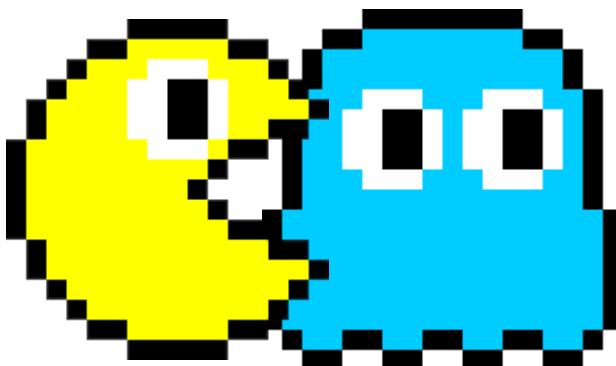
ตัวอย่าง : การเช็คการชนกันโดยการเทียบ pixel ที่ทับกัน

การเช็คการชนจะขึ้นอยู่กับทิศทาง pacman :

- ถ้าทิศทางของ pacman = ซ้าย :
 - ให้เช็คโดยการนำ pixels แรกสุด - 13 (ขนาด pacman คือ 16 pixels) ว่าเท่ากับ ตำแหน่งของผีไหม
- ถ้าทิศทางของ pacman = ขวา :
 - ให้เช็คโดยการนำ pixels แรกสุด + 13 (ขนาด pacman คือ 16 pixels) ว่าเท่ากับ ตำแหน่งของผีไหม

```
22     if(direction == LEFT)
23         begin
24             //if pacman and blue ghost are within each other's display area
25             if(pacman_x - 13 <= blue_x && pacman_x >= blue_x - 13 && pacman_y - 13 <= blue_y && pacman_y >= blue_y - 13)
26                 collided = 1;
27
28             //if pacman and red ghost are within each other's display area
29             if(pacman_x - 13 <= red_x && pacman_x >= red_x - 13 && pacman_y - 13 <= red_y && pacman_y >= red_y - 13)
30                 collided = 1;
31         end
32
33     //if direction of pacman is right
34     else if(direction == RIGHT)
35         begin
36             //if pacman and blue ghost are within each other's display area
37             if(pacman_x + 13 >= blue_x && pacman_x <= blue_x + 13 && pacman_y - 13 <= blue_y && pacman_y >= blue_y - 13)
38                 collided = 1;
39
40             //if pacman and red ghost are within each other's display area
41             if(pacman_x + 13 >= red_x && pacman_x <= red_x + 13 && pacman_y - 13 <= red_y && pacman_y >= red_y - 13)
42                 collided = 1;
43         end
```

(ตัวอย่าง code สำหรับการตรวจสอบการชนของ pacman และ ผีทั้ง 2 สี)



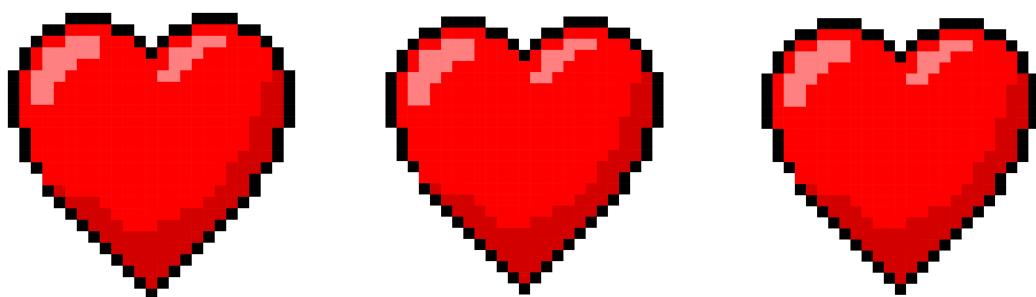
Hearts_rom : เป็น module ที่จะเก็บ memory สำหรับภาพของหัวใจ

Hearts_display : เป็น module ที่ใช้ในการแสดงผลหัวใจที่รับ memory มาจาก “Hearts_rom” ที่เหลืออยู่ โดยจะรับ input ว่าเหลืออยู่เท่าไหร่มาจาก module “Game_state_machine”

ตัวอย่าง : การเช็คจำนวนหัวใจที่เหลืออยู่เพื่อนำมาแสดงผล

```
26     //if 1 heart(left)
27     if(x >= 240 && x < 256 && y >= 16 && y < 32)
28         begin
29             col = x-240;
30             if(num_hearts > 0)                                //if num_hearts > 0 (1,2,3) left heart is on
31                 row = y - 16;
32
33             else                                                 //else (0) left heart is off
34                 row = y;
35             hearts_on = 1;
36         end
37
38
39     //if 2 hearts(center)
40     if(x >= 256 && x < 272 && y>= 16 && y < 32)
41         begin
42             col = x-256;
43             if(num_hearts > 1)                                //if num_hearts > 1 (2,3) center heart is on
44                 row = y - 16;
45
46             else                                                 //else (0,1) center heart is off
47                 row = y;
48             hearts_on = 1;
49         end
50
51     //if 3 hearts(right)
52     if(x >= 272 && x < 288 && y >= 16 && y<32)
53         begin
54             col = x-272;
55             if(num_hearts > 2)                                //if num_hearts > 2 (3) right heart is on
56                 row = y - 16;
57
58             else                                                 //else (0,1,2) right heart is off
59                 row = y;
60             hearts_on = 1;
61         end
62
63     end
```

(ตัวอย่าง code ในการแสดงผลหัวใจทั้ง 3 อัน)



Game_state_machine : เป็น module ที่บอกรถสถานะของเกม และ คงความคุมจำนวนของหัวใจ
สถานะของเกมมีดังต่อไปนี้ :

Start : เป็น state ที่เรากำหนดค่าเริ่มต้นเพื่อรอผู้เล่นพร้อม (กดปุ่ม Start เพื่อเริ่มเกม)

```

start:
begin
    if(start_btn_posedge)
begin
    game_state_next = playing;
    game_reset = 1;
    game_en_next = 1;
end
end

```

(ตัวอย่าง code ของสถานะ “start” ของเกม)

Playing : เป็น state ที่บ่งบอกว่าเกมกำลังเล่นอยู่ และ ถ้าเกิดมีการชนขึ้นมา ก็จะทำการหักหัวใจของผู้เล่น และให้ย้ายไป state ที่ “Hit” และถ้าหัวใจเหลือ 1 และเกิดการชน ก็จะให้ สถานะของเกม = gameover

```

playing:
begin
    if(collision)
begin
    if(hearts_reg == 1)      //if has 1 heart left
begin
    hearts_next = hearts_reg - 1;      //decrement heart
    game_state_next = gameover;
    game_en_next = 0;
end
else
begin
    game_state_next = hit;
    if (timer_reg == 0) // Only set the timer if it's not already set
        timer_next = 200000000;
    hearts_next = hearts_reg - 1;
end
end
end

```

(ตัวอย่าง code ของสถานะ “playing” ของเกม)

Hit : เป็น state ที่เอาทำการหน่วงเวลา(2 seconds) ตอนที่เกิดการชน

ตัวอย่าง : หลังจากตรวจพบการชน ก็จะกำหนดให้ time = 200000000 (2 seconds) และถ้า time = 0 เมื่อไหร่ ถึงจะเกิดการชนอีกรอบได้หมายความว่า ทุกครั้งที่เกิดการชน จะเป็นอันดับประมาณ 2 วินาที

```

hit:
begin
    if(timer_reg > 0)//pacman can not hit again until timer is 0 (wait 2 seconds)
        begin
            timer_next = timer_reg - 1;
        end
    else
        begin
            game_state_next = playing;
            timer_next = 0;
        end
end

```

(ตัวอย่าง code ของสถานะ “hit” ของเกม)

Gameover : เป็น state ที่เอาบ่งบอกว่า gameover และถ้ามีการกดปุ่ม start ก็จะทำ Reset เกมและเริ่มใหม่

```

gameover:
begin
    if(start_btn_posedge)
        begin
            game_state_next = start;
            game_reset = 1;
            hearts_next = 3;
        end
end

```

(ตัวอย่าง code ของสถานะ “gameover” ของเกม)

Star_rom : เป็น module ที่เอาไว้เป็น memory ของ star

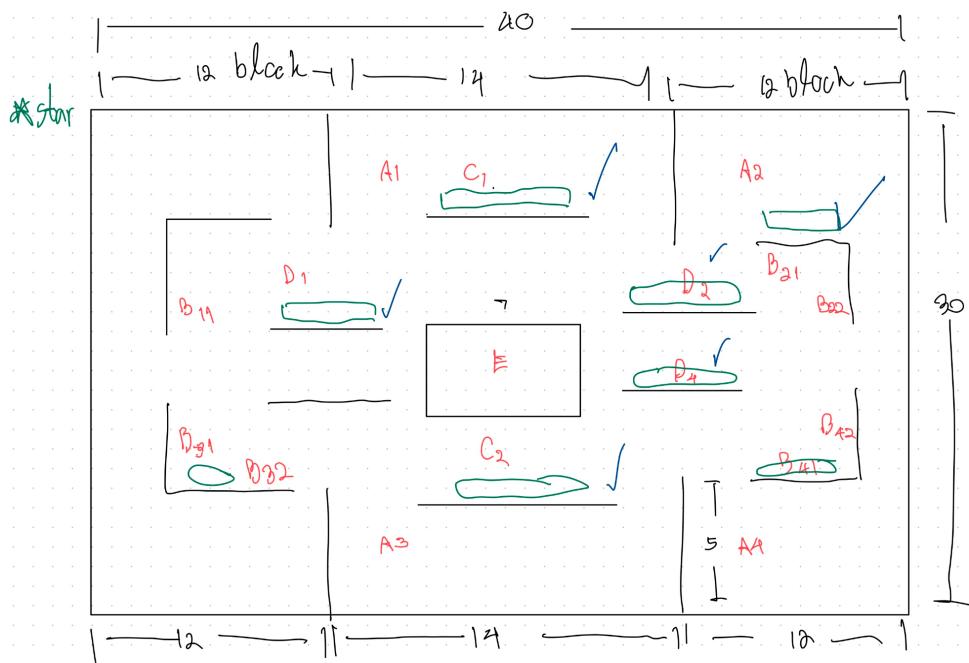
Star : เป็น module นำเอา memory จาก module “Star_rom” มาทำการ spawn star และค่อยเช็คว่า pacman เก็บดาวแล้วหรือไม่
ตัวอย่าง : ตำแหน่งของดาวจะถูกกำหนดเอาไว้ให้ spawn ได้ที่

ตำแหน่งดังต่อไปนี้ :

- B32, B21, B41



- C1, C2
- D1, D2, D4



(รูปภาพแสดงตำแหน่งที่ star สามารถ spawn ได้)

การ spawn star และ การตรวจสอบการชนของ pacman กับดาว

Spawn : สร้าง state สำหรับการ spawn ดาว ซึ่งดาวจะเกิดได้แค่ 1 ดวงต่อครั้ง และถ้า pacman มาเก็บ ก็จะได้ทำการ respawn ใหม่

ตัวอย่าง : code สำหรับการ spawn ที่ Block "B21"

```
else if(area_select_reg == 1) //B21
begin
    star_y_next = 64;
    star_x_next = B21_reg;
end
```

(ตัวอย่าง code สำหรับการ spawn ดาวที่ตำแหน่ง B21)

Waiting : เป็น state ที่เอาไว้ตรวจสอบการชนของ pacman กับ ดาว โดยถ้า pixels ของ pacman เหลือม กับ pixel ของดาว 3 pixels ก็จะถือว่าเก็บดาวแล้ว

ตัวอย่าง : code สำหรับการเช็คการชน เมื่อ ทิศทางของ pacman = ข้าย

```
//if pacman collide with facing LEFT
if(direction == LEFT && pacman_x - 13 <= star_x_reg && pacman_x >= star_x_reg - 13 && pacman_y - 13 <= star_y_reg && pacman_y >= star_y_reg - 13)
begin
    star_state_next = respawn;
    new_score_next = 1;
end
```

(ตัวอย่าง code สำหรับการตรวจสอบการชนของ pacman กับ ดาว เมื่อ pacman มีทิศทางซ้าย)

ตัวอย่าง : การกำหนดค่าตำแหน่ง (row and col) สำหรับแสดงผลดาว

```
//sprite coordinate register to display stars
wire [3:0] col;
wire [3:0] row;

//current pixel coordinate - star coordinate = pixel coordinate of star
assign col = x - star_x_reg;
assign row = y - star_y_reg;
wire [11:0] color_data_star;

Star_rom Star_rom_unit (.clk(clk), .col(col), .row(row), .color_data(color_data_star));

//assign stars_on for output
assign stars_on = (x >= star_x_reg && x < star_x_reg + 16 && y >= star_y_reg && y < star_y_reg + 16 && color_data != 12'b111111111110)? 1 : 0;

//assign rgb_out for output
assign color_data = color_data_star;
```

(ตัวอย่าง code สำหรับการกำหนดค่าตำแหน่งในการแสดงผลดาวในหน้าจอ)

gameover_rom : ใช้เก็บ memory สำหรับข้อความ “GAMEOVER” เพื่อไปใช้ใน module “gameover_display”

gameover_display : นำ Rom จาก module “gameover_rom” มาใช้สำหรับแสดงผล ข้อความ “GAMEOVER” บนหน้าจอเมื่อเกม over

ตัวอย่าง :

