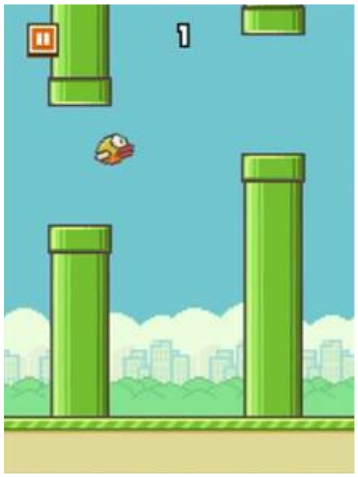
Final Project

學號: 109062174、108032053 姓名: 謝承恩、陳凱揚

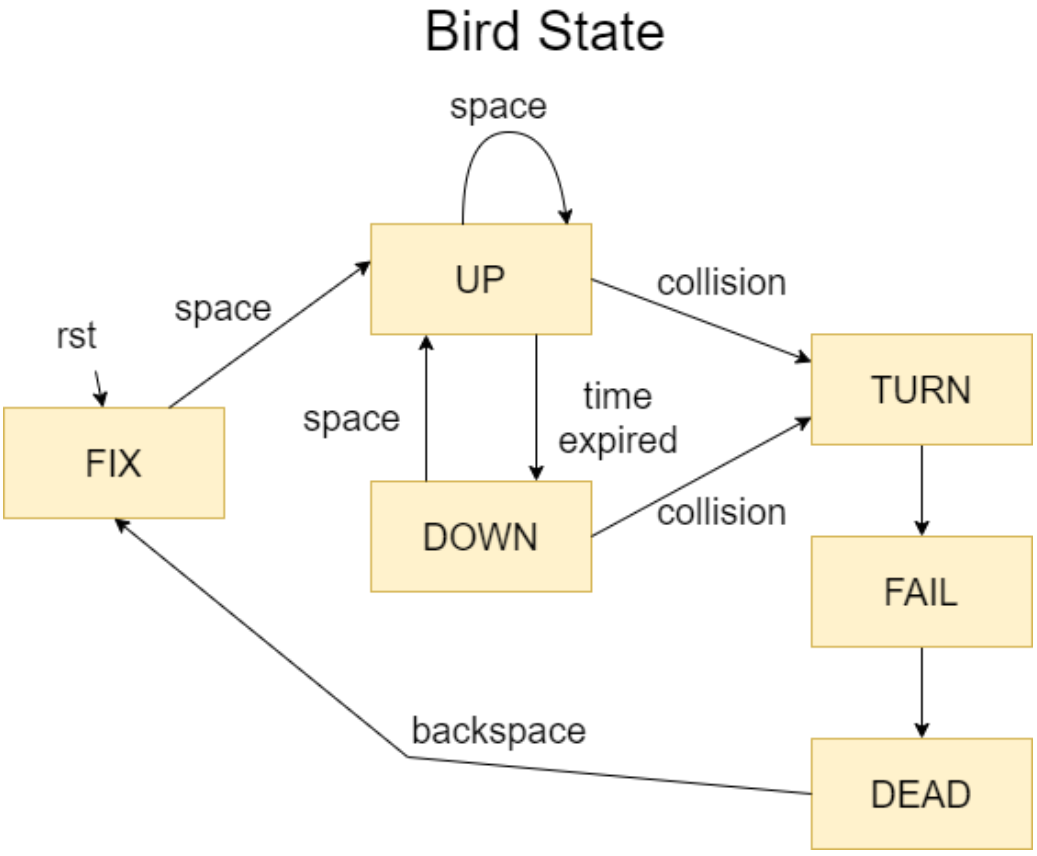
1. 設計概念：Flappy bird



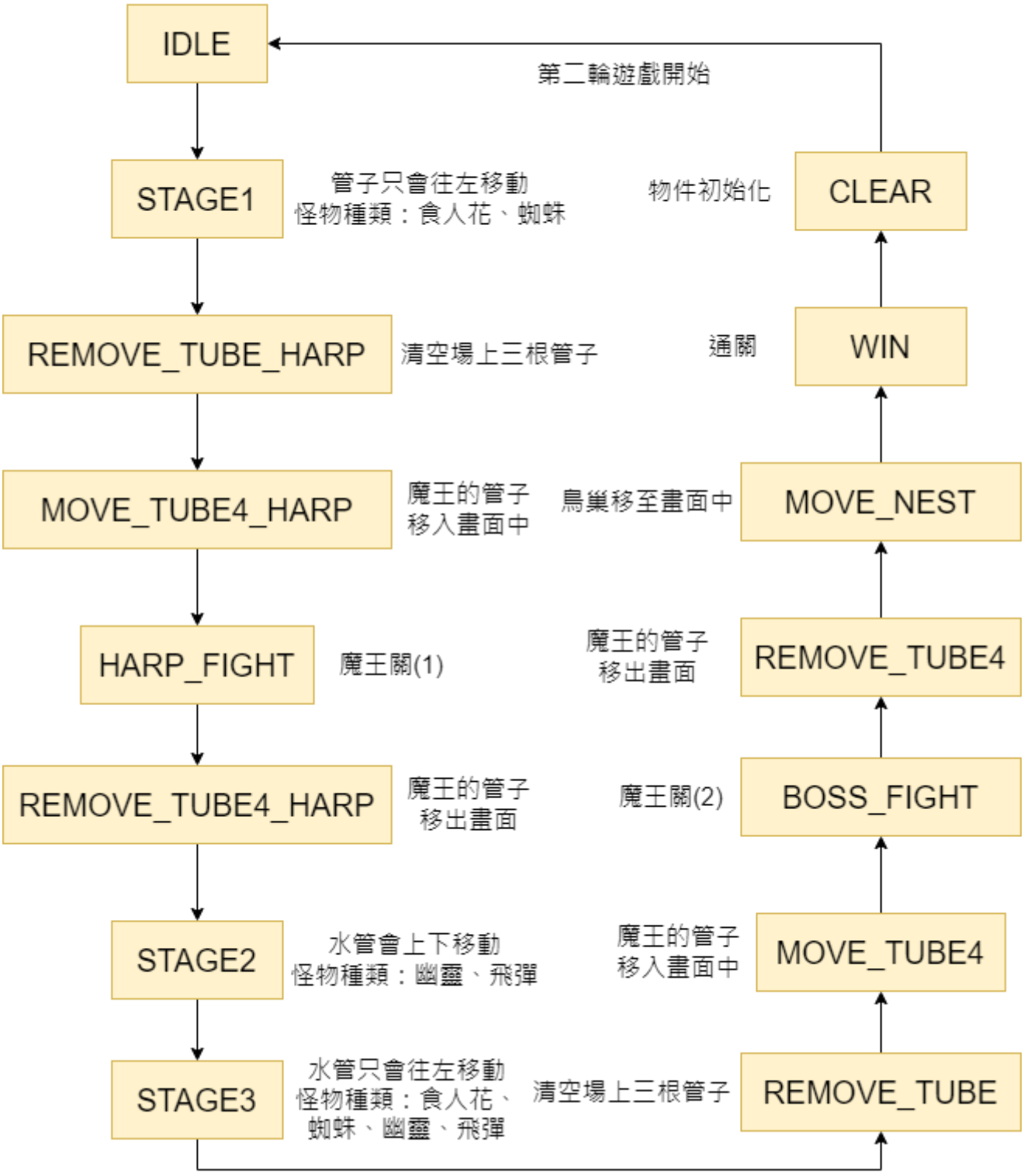
使用者透過空白鍵控制小鳥的高度，讓小鳥不斷通過水管，途中閃避各種怪物的攻擊，最後回到自己的鳥巢，若中途小鳥死掉或成功通關可以按 backspace 開啟下一輪遊戲。

2. 架構細節與方塊圖：

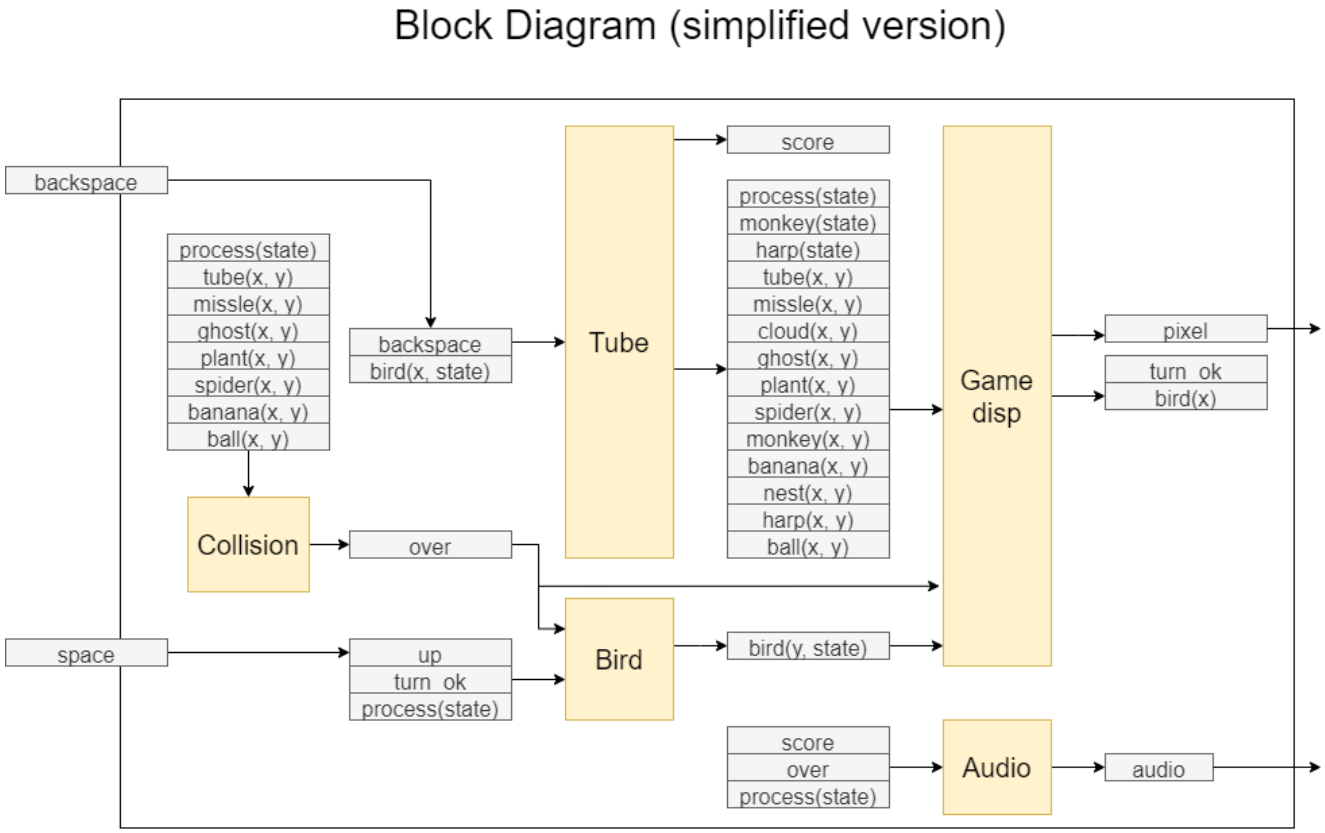
FSM：



Process State



Block_diagram：



(1) Bird.v

這個模組用來控制鳥的 y 座標，鳥的飛行狀態分為三種：UP, DOWN, FIX 分別對應到鳥在上升、下降及高度不變，上升的效果，我們的做法就是用一個 counter (move_up_cnt) 來計時，每當數到一個常數時就將鳥的 y 座標遞減 1(往上移)，下降的原理同上，但由於鳥在上升時我們想呈現短暫加速的效果，因此 UP_SEC 會比 DOWN_SEC 還小，另外 STATE_SEC 是用來控制鳥往上升的時間。

```
parameter UP_SEC = 1024*512; //數字越小，鳥上升越快
parameter DOWN_SEC = 1024*1024; //數字越小，鳥掉落越快
parameter STATE_SEC = 1024*1024*20; // 數字越大，鳥上升時間越長
```

另外我們有做防止鳥超出螢幕上下邊界的設計

```
UP:begin
    next_move_down_cnt = 0;
    if(bird_y <= 20)begin
        next_move_down_cnt = 0;
        next_bird_y = bird_y;
    end
```

```
DOWN:begin
    next_move_up_cnt = 0;
    if(bird_y >= 460)begin
        next_move_down_cnt = 0;
        next_bird_y = bird_y;
    end
```

UP state 的程式碼(DOWN state 原理相同)：

```
UP:begin
    next_move_down_cnt = 0;
    if(bird_y <= 20)begin
       next_move_down_cnt = 0;
       next_bird_y = bird_y;
    end
    else if(move_up_cnt == UP_SEC)begin
        next_move_up_cnt = 0;
        if(bird_y > 10)begin
            next_bird_y = bird_y - 1;
        end
        else begin
            next_bird_y = bird_y;
        end

    end
    else begin
        next_move_up_cnt = move_up_cnt + 1;
        next_bird_y = bird_y;
    end
end
```

除了上面三種狀態，鳥還有三種狀態用來呈現死亡的動畫，分別是 TURN、FALL、DEAD，TURN state 僅是用來告知圖片應該轉向，FALL state 就是讓鳥不斷落下直到直面，DEAD state 不需要對鳥的 y 座標做處理，只需等待使用者按下 backspace 鍵後，遊戲回復到初始畫面，將鳥的 y 座標設定成螢幕中間(240 = 480 / 2) 即可

```
TURN:begin

end
FALL:begin
    next_move_up_cnt = 0;
   if(move_down_cnt == DOWN_SEC)begin
        next_move_down_cnt = 0;
        next_bird_y = bird_y + 1;
    end
    else begin
        next_move_down_cnt = move_down_cnt + 1;
        next_bird_y = bird_y;
    end
end
DEAD:begin
    if(process_state == IDLE)begin
        next_bird_y = 240;
        next_move_up_cnt = 0;
        next_move_down_cnt = 0;
    end

end
```

接下來說明鳥的 FSM：

FIX state：

當鳥在 FIX state 的時候(遊戲初始化面)，按下空白鍵就會進入 UP state

```
FIX:begin
    if(process_state != MOVE_NEST && process_state != WIN && been_ready && key_down[9'b000101001] == 1'b1)begin
        next_state = UP;
        next_state_cnt = 0;
    end
end
```

UP state:

這個狀態持續一小段時間後就會自動進入 DOWN state，因為我們的設計是鳥會自動落下，如果在 UP state 按下空白鍵就會更新 counter 的值，但如果遊戲結束(鳥撞到東西)，就會進入 TURN state，或者如果遊戲通關的話，鳥要飛回鳥巢，因此鳥巢的位置是在螢幕正中間，此時就控制鳥的 y 座標直到 240 的位置。

```
UP:begin
    if(over == 1)begin
        next_state = TURN;
    end
    else if(process_state == MOVE_NEST)begin
        if(bird_y == 240)begin
            next_state = FIX;
        end
        else if(bird_y > 240)begin
            next_state = UP;
        end
        else begin
            next_state = DOWN;
        end
    end
    else if(been_ready && key_down[9'b000101001] == 1'b1)begin
        next_state_cnt = 0;
    end
    else if(state_cnt == STATE_SEC)begin
        next_state = DOWN;
    end
    else begin
        next_state_cnt = state_cnt + 1;
    end
end
```

DOWN state:

若玩家沒有按下空白，則一直維持在 DOWN state，其他部分和 UP state 一樣，如果遊戲結束，就會進入 TURN state，或者如果遊戲通關的話，鳥要飛回鳥巢，此時就控制鳥的 y 座標直到 240 的位置

```
DOWN:begin
    if(over == 1)begin
        next_state = TURN;
    end
    else if(process_state == MOVE_NEST)begin
        if(bird_y == 240)begin
            next_state = FIX;
        end
        else if(bird_y < 240)begin
            next_state = DOWN;
        end
        else begin
            next_state = UP;
        end
    end
    else if(been_ready && key_down[9'b000101001] == 1'b1)begin
        next_state = UP;
        next_state_cnt = 0;
    end
end
```

TURN state

進到 TURN state 後，Gamp_disp module 會將小鳥的圖案轉成斜 45 度，當小鳥的圖案轉好後，會輸出一個 turn_ok 為 1 的訊號，小鳥的狀態就會進到 FALL state

```
TURN:begin
    if(turn_ok)begin
        next_state = FALL;
    end
end
```

FALL state

讓鳥不斷下降，直到掉落地面(y = 470)，則進入 FALL state

```
FALL:begin
    if(bird_y+15 > 470)begin
        next_state = DEAD;
    end
end
```

DEAD state

如果遊戲回到初始化面的話(IDLE)，則鳥會回到 FIX state(鳥保持在 y = 240 的地方，不會隨時間往下掉)

```
DEAD:begin
    if(process_state == IDLE)begin
        next_state = FIX;
        next_state_cnt = 0;
    end
end
```

(2) Tube.v

這個 module 有三個功能，第一是用來控制所有物件(除了小鳥)的座標，並將這些座標傳送到 Game_disp module，因為我們最一開始生成的物件是水管，因此取名為 Tube.v ，但後來直接將其他物件也加在這個 module，第二個功能是紀錄分數，因為我們是用管子的 x 座標判斷加分的，因此在這個 module 紀錄分數非常適合，第三是紀錄所有關卡的流程狀態(process_ state)。

首先解釋加分的機制：

這部份很容易實現，只要每當有管子的 x 座標等於 190，就將分數加 1，因為鳥的 x 座標是固定不變的(除了 IDLE state 和 MOVE state)，因此水管移動到 x = 190 時(鳥的後面)，就代表通過一根水管。

```
always @(*) begin
    next_score = score;
    if(process_state == IDLE)begin
        next_score = 0;
    end
    if(tube1_x == 190)begin
        next_score = score + 1;
    end
    if(tube2_x == 190)begin
        next_score = score + 1;
    end
    if(tube3_x == 190)begin
        next_score = score + 1;
    end
    if(tube4_x == 190)begin
        next_score = score + 1;
    end
    if(tube5_x == 190)begin
        next_score = score + 1;
    end
end
```

接著解釋關卡流程的部分：

IDLE state：

一開始會在 IDLE state(顯示初始畫面)，直到玩家按下第一次空白鍵後，才會進入 stage1(管子開始移動)，我們一開始並沒有設計這個 state，因此 reset 之後遊戲就會直接開始，但玩家可能還沒準備好，顯然這樣的設計不合理。

```
IDLE:begin
    if(bird_state != FIX)begin
        next_process_state = STAGE1;
    end
    // only cloud moving
end
```

STAGE1 state:

這個狀態水管會開始往左移動，因此不斷讓三個水管的 x 座標遞減，如果水管的 x 座標小於 100 時，我們就重置水管的 x 座標到 800，並且將高度設為一個隨機數，因此可以達到水管不斷出現的效果，當達到某個分數時(自行設定)，就會進入 REMOVE_TUBE_HARP，準備打第一個魔王。

```
STAGE1:begin

    // 管子依 clock 的頻率向左平移
    next_tube1_x = tube1_x - 1;
    next_tube2_x = tube2_x - 1;
    next_tube3_x = tube3_x - 1;

    if(score == STAGE1_END)begin
        next_process_state = REMOVE_TUBE_HARP;
    end

    // 重置管子
    if(tube1_x < 100)begin
        next_tube1_x = 800; // 柱子移動到最左邊
        next_tube1_y = rand + 150; // y 座標為隨
    end
    if(tube2_x < 100)begin
        next_tube2_x = 800;
        next_tube2_y = rand + 150;
    end
    if(tube3_x < 100)begin
        next_tube3_x = 800;
        next_tube3_y = rand + 150;
    end
end
```

REMOVE_TUBE_HARP：

這個階段是魔王關的第一個前置作業，將場上的水管全部清空，因此還是讓水管的 x 座標不斷遞減，但是當水管的 x 座標達到 100 時，就要將水管的 x 座標固定在 800, 1040, 1280(螢幕外面)，等所有水管移出螢幕後，就會進入 MOVE_TUBE4_HARP

```
    if(tube1_x < 100 || tube1_x > 700)begin
        next_tube1_x = 800; // 柱子移動到最左邊，重新回會最右邊
        next_tube1_y = rand + 150; // y 座標為隨機數
    end
    if(tube2_x < 100 || tube2_x > 700)begin
        next_tube2_x = 1040;
        next_tube2_y = rand + 150;
    end
    if(tube3_x < 100 || tube2_x > 700)begin
        next_tube3_x = 1280;
        next_tube3_y = rand + 150;
    end
end
```

```
REMOVE_TUBE_HARP:begin

    next_tube1_x = tube1_x - 1;
    next_tube2_x = tube2_x - 1;
    next_tube3_x = tube3_x - 1;
    if(tube1_x > 700 && tube2_x > 700 && tube3_x > 700)begin
        next_process_state = MOVE_TUBE4_HARP;
    end
```

MOVE_TUBE4_HARP：

這個狀態是魔王關的第二個前置作業，將玩家的水管(會限制鳥的上下高度)，以及魔王的水管(金色水管)移動到 x = 220, x = 460 的位置，等兩根水管就定位後，就會進入 HARP_FIGHT，正式開始打魔王

```
MOVE_TUBE4_HARP:begin

    next_tube5_x = tube5_x - 1;
    next_tube4_x = tube4_x - 1;
    if(tube4_x <= 460 && tube5_x <= 220)begin
        next_process_state = HARP_FIGHT;
        next_tube4_state = UP;
    end
```

HARP_FIGHT:

此時魔王的水管會不斷上下移動(魔王跟著移動)，因此 tube4 有 UP, DOWN 兩個 state

```
    else if(tube4_state == UP)begin
        next_tube4_y = tube4_y - 1;
        if(tube4_y < 110)begin
            next_tube4_state = DOWN;
        end

    end
    else if(tube4_state == DOWN)begin
        next_tube4_y = tube4_y + 1;
        if(tube4_y > 330)begin
            next_tube4_state = UP;
        end

    end
end
```

但是水管不能一開始就馬上移動，要先等魔王就定位，因此魔王在 MOVE_0 state 的時候，水管的位置固定，此外當魔王要離場時，水管也不能動，因此當魔王在 HARP_LEAVE state 的時候，水管位置也固定，等到魔王完全離開(HARP_DEAD)，就可以進入 REMOVE_TUBE4_HARP state。

```
HARP_FIGHT:begin

    if(harp_state == MOVE_0)begin
        next_tube4_y = tube4_y;
    end
    else if(harp_state == HARP_DEAD)begin
        next_process_state = REMOVE_TUBE4_HARP;
    end

    else if(harp_state == HARP_LEAVE)begin
        next_tube4_y = tube4_y;
    end
```

REMOVE_TUBE4_HARP:

這個狀態用來清場，將場面上的兩根水管(玩家的, 魔王的)移動到螢幕外，之後進入 STAGE2 state

```
REMOVE_TUBE4_HARP:begin

    next_tube4_x = tube4_x - 1;
    next_tube5_x = tube5_x - 1;
    if(tube4_x > 700 && tube5_x > 700)begin
        next_process_state = STAGE2;
    end

    if(tube4_x < 100 || tube4_x > 700)begin
        next_tube4_x = 1040;
        next_tube4_y = 240;
    end

    if(tube5_x < 100 || tube5_x > 700)begin
        next_tube5_x = 800;
        next_tube5_y = 240;
    end
end
```

STAGE2:

這個 state 水管不只要往左移動，還要上下移動

往左移動的部分和剛才 STAGE2 一樣，這部份已經解釋過

```
// 管子依 clock 的頻率向左平移
next_tube1_x = tube1_x - 1;
next_tube2_x = tube2_x - 1;
next_tube3_x = tube3_x - 1;

// 重置管子
if(tube1_x < 100)begin
    next_tube1_x = 800; // 柱子移
    next_tube1_y = rand + 150; //
end
if(tube2_x < 100)begin
    next_tube2_x = 800;
    next_tube2_y = rand + 150;
end
if(tube3_x < 100)begin
    next_tube3_x = 800;
    next_tube3_y = rand + 150;
end
```

上下移動的部分就和剛才魔王的水管上下移動一樣，每個水管都有一個 state，根據 UP 或 DOWN 來遞減或遞增 y 座標，

```
// 管子上下移動
if(tube1_state == UP)begin
    if(tube1_y <= 190)begin
        next_tube1_state = DOWN;
    end
    next_tube1_y = tube1_y - 1;
end
else begin
    if(tube1_y >= 270)begin
        next_tube1_state = UP;
    end
    next_tube1_y = tube1_y + 1;
end
```

當達到一定分數後，進入 STAGE3

```
if(score == STAGE2_END)begin
    next_process_state = STAGE3;
end
```

STAGE3:

這個階段做的事情和 STAGE1 一模一樣，水管只會往左移動

```
STAGE3:begin

    // 管子依 clock 的頻率向左平移
    next_tube1_x = tube1_x - 1;
    next_tube2_x = tube2_x - 1;
    next_tube3_x = tube3_x - 1;

    if(score == STAGE3_END)begin
        next_process_state = REMOVE_TUBE;
    end

    // 重置管子
    if(tube1_x < 100)begin
        next_tube1_x = 800; // 柱子移動到最左
        next_tube1_y = rand + 150; // y 座標
    end
    if(tube2_x < 100)begin
        next_tube2_x = 800;
        next_tube2_y = rand + 150;
    end
    if(tube3_x < 100)begin
        next_tube3_x = 800;
        next_tube3_y = rand + 150;
    end
end
```

接下來從 REMOVE_TUBE state 到 REMOVE_TUBE4 state 是第二個魔王關，這部份得邏輯和剛才第一隻魔王的部分一模一樣，因此就不再重複說明一次了，直接說明 MOVE_NEST state：此時遊戲準備通關，將帶有鳥巢的水管移至 x = 320 的位置，等鳥也就定位後(x = 300)就會進入 WIN state，

```
MOVE_NEST:begin
    next_tube3_x = tube3_x - 1;
    if(tube3_x == 320 && bird_x == 300)begin
        next_process_state = WIN;
    end
end
```

WIN state:
僅用作通知其他 module(聲音模組撥放通關音效)，水管本身不需要在移動了。

在上面的任一個狀態中，如果小鳥死掉了，玩家按下 backspace 後，就會回到 IDLE state，或者玩家通關後(WIN state) 按下 backspace 也會回到 IDLE state

接下來解釋怪物是如何釋放的，我們僅以植物為例作說明：
植物的 x 座標是綁定其中一根水管，因為植物是從水管中生起，因此 x 座標和水管的 x 座標必須相同，y 座標的部分，一開始先讓植物隱藏在水管下，並將 ready(plant_launch) 設為 1。

```
assign plant_x = tube2_x;
```

```
if(rst)begin
   plant_y = tube2_y + 80 + 30; // 躲在水管底下
   plant_launch = 1;
end
```

當分數在某個區間時(自行設定或隨機決定)且 plant_launch 為 1 時，將植物開始往上移動，植物露出水管，若植物已經隨水管移動到螢幕外(x > 640)，便將 plant_launch 設為 0，直到分數遞增 1 後，就將植物移動回水管下面，並將 plant_launch 設為 1

```
else if(score >= 6 && score <= 7 && plant_launch == 1)begin
    if(plant_y + 25 <= tube2_y + 80)begin

    end
    else begin
        plant_y = plant_y - 1;
    end
    if(plant_x > 640)begin
        plant_launch = 0;
    end
end

else if(score == 8)begin
        plant_y = tube2_y + 80 + 30; // 躲回水管底下
        plant_launch = 1;
end
```

其餘怪物的原理都差不多，只是差在飛彈是水平移動，植物和蜘蛛是上下移動，幽靈則是垂直加水平移動而已，且飛彈, 幽靈和水管的 x 座標無關。

魔王的部分(以 monkey 為例)：

魔王會有自己的 state，用來顯示不同的動作，猴子的動作總共分為 10 張圖片

猴子的 x 座標和金色水管相同

```
assign monkey_x = tube4_x;
```

魔王一開始也是躲在水管下，當 process_state 進到 BOSS_FIGHT 後，就會從管子中浮出來，並進入到 MOVE_1 state(魔王會開始動作)

```
else if(monkey_state == MOVE_0)begin
    if(process_state == BOSS_FIGHT)begin
        next_monkey_y = monkey_y - 1;
        if(monkey_y + 28 < tube4_y + 80)begin
            next_monkey_y = tube4_y + 80 -28;
            next_monkey_state = MOVE_1;
        end
    end
    else begin
        next_monkey_y = 460;
    end
end
```

接下來就是不斷依序切換猴子的狀態(圖片)，但為了增加樂趣，猴子可以選擇吃香蕉或丟香蕉，選擇的方式就是根據隨機數的值做決定

```
MOVE_9:begin
    next_monkey_y = tube4_y + 80 -28;
    if(banana_ready == 1)begin
        next_monkey_state = rand > 31 ? MOVE_1: MOVE_5;
    end
end
```

接著說明猴子丟香蕉是如何實現的

一開始香蕉的 x 座標是在螢幕外(x = 700)，y 座標跟猴子的 y 座標相同，

```
always @(posedge clk_missle, posedge rst) begin
    if(rst)begin
        banana_x = 700;
        banana_y = monkey_y;
        banana_ready = 1;
    end
```

當猴子做出丟香蕉的動作時，香蕉會直接出現在猴子左邊 20 pixel 的位置

```
else if(monkey_state == MOVE_6 && banana_ready == 1)begin
    banana_x = monkey_x - 20;
    banana_ready = 0;
end
```

接著香蕉便開始水平移動，此時香蕉的 y 座標就固定了，x 座標不斷遞減，直到飛出邊界

```
else if(banana_ready == 0)begin //香蕉正在飛
    if(banana_x < 50)begin // 重置香蕉
        banana_x = 700;
        banana_y = monkey_y;
        banana_ready = 1;
    end
    else begin
        banana_x = banana_x - 1;
    end
end
else if(banana_ready == 1)begin
    banana_y = monkey_y;
end
```

魔王有一個生命值，當魔王關開始後就會開始計時，當到達設定的數字後就會自動死亡

```
always @(*) begin
    if(monkey_state != MOVE_0)begin
        next_monkey_life = monkey_life + 1;
    end
    else if(process_state == IDLE)begin
        next_monkey_life = 0;
    end
    else begin
        next_monkey_life = monkey_life;
    end
end
```

(3) Game_disp.v

這個 module 用來顯示所有的物件，但由於程式碼實在太過龐大，因此我僅挑選比較特別的部分做說明。

1. 圖案若可以重複畫，就不要儲存

我們的背景圖片只有 18*30 pixel，比小鳥還小，因為背景圖片大多地方都重複，因此不需要儲存整張圖片

```
always @(*)begin
    if(v_cnt < 451)begin
        add_1 = 0;
    end
    else begin
        add_1 = (h_cnt % 18) + 18*(v_cnt - 451);
    end

end
```

，

另外像 GAME START ,GAME WIN , GAME OVER 等字樣我們也是用程式達到放大 4 倍的效果，不然記憶體真的會不夠用

```
// 遊戲結束 76*76 放大後左上角座標 244,164
always @(*)begin
    add_3 = ((h_cnt - 244)>>1) + 76*((v_cnt - 164)>>1) ;
end

// flappybird 94*16 中心點(320, 140) 左上角 226, 124
always @(*)begin
    add_40 = ((h_cnt - 226)>>1) + 94*((v_cnt - 124)>>1) ;
end
// gamestart 94*14 中心點(320, 346) 左上角 226, 332
always @(*)begin
    add_41 = ((h_cnt - 226)>>1) + 94*((v_cnt - 332)>>1) ;
end
// gamewin 82*14 中心點(320, 140) 左上角 238, 126
always @(*)begin
    add_42 = ((h_cnt - 238)>>1) + 82*((v_cnt - 126)>>1) ;
end
```

水管也是指儲存一小截，其他往上下延伸的部分也是重複畫，但因為我們有水管上下移動的關卡，這個部分就會比較麻煩，但只要有水管的座標，就還是做得到。

```
always @(*)begin
    if(v_cnt > (tube1_y + 80) + 16)begin
        add_13 = 19*60 + (h_cnt - tube1_x + 30);
    end
    else begin
        add_13 = (v_cnt - (tube1_y+ 80 + 10) + 10)*60 + (h_cnt - tube1_x + 30);
    end
end
```

2. 處理物體重疊時的顯示問題：

比如當鳥和雲重疊、鳥和水管重疊、鳥和怪物重疊、鳥和鳥巢重疊、魔王和雲重疊...等，需要做特別處理，因為每張圖案都有自己的背景，若不做特別處理，圖案重疊時圖案的背景色就會顯示出來，因此針對有可能發生重疊的物件，我們都會做處理，方法簡單來說就是當兩種物件顯示的範圍重疊時，比如小鳥和雲，此時小鳥的背景要變成雲，而不是小鳥原本的背景(白色)，因此當小鳥的記憶體讀到的值為 12'hfff (白色)時，就將輸出的 pixel 接到雲的記憶體，這樣就可以達到我們像要的效果，其他物件的重疊也都是利用這個邏輯

```
if(data_out_5 == 12'hfff)begin
else if((h_cnt > cloud1_x - 20) && (h_cnt < cloud1_x + 20) && (v_cnt > cloud1_y - 10) && (v_cnt < cloud1_y + 10))begin
    rgb = data_out_7; // cloud1
end
```

3. Gameover 時背景要變暗：

當小鳥死掉後遊戲背景要變暗，並顯示 game over 的圖(但 game over 不會變暗)，我的作法是當遊戲結束時就將每個 pixel 的 rgb 值「分別」向右 shift 一個 bit，這樣就可以達到變暗的效果，但 game over 圖片的區域仍然維持原本的顏色

```
always @(*) begin
    if(bird_state != DEAD || ((h_cnt > 320 - 76) && (h_cnt < 320 + 76) && (v_cnt > 240 - 76) && (v_cnt < 240 + 76)))begin
        pixel = rgb;
    end
    else begin
        pixel = {rgb[11:8] >> 1 , rgb[7:4] >> 1, rgb[3:0] >> 1};
    end
end
```

4. 動態的影像生成：

   小鳥的翅膀會動、植物的嘴巴會動、幽靈的斗篷會動、魔王會有攻擊動作…等，前三者用一個 1 bit counter 就可以做到，比如 counter == 1 就放第一張圖，counter == 0 就放第二張圖，然後不斷交錯，魔王的動畫就必須搭配魔王的狀態，看目前是哪個狀態就放哪張圖案

(4) Collision.v

用來判斷發生碰撞的 module，我們僅以飛彈做說明，因為判斷碰撞的邏輯每種怪物都是一樣的

當小鳥的 x 座標和飛彈的 x 座標重疊後()，要先判斷小鳥目前是在飛彈的上方還是下方，若小鳥在飛彈上方，此時若小鳥的底部超過飛彈的頂部，就判斷發生碰撞，若小鳥在飛彈下方，此時若小鳥的頂部超過飛彈的底部，就判斷發生碰撞。

```
// 飛彈
if((bird_x + 15 >= missle_x - 20) && (bird_x - 15 <= missle_x + 20))begin
    if(bird_y <= missle_y)begin // 鳥在飛彈上面
        if(bird_y + 15 >= missle_y - 15)begin
            next_collision = 1;
        end
    end
    else begin // 鳥在飛彈下面
        if(bird_y - 15 <= missle_y + 15)begin
            next_collision = 1;
        end
    end
end
```

若關卡回到初始狀態(IDLE)，將 collision reset 成 0，另外因為我們有做無碰撞模式，因此當 SW0 拉起時，collision 就會一直維持 0

```
if(process_state == IDLE || SW0 == 1)begin
    next_collision = 0;
end
```

(5) Random.v

一個非常簡單的亂數產生器，feed 為 1 個 bit 的值，由 rand 的其中兩個 bit 做 XOR 得到，接著就將 feed 不斷插入 rand 的第 0 個 bit 並將 rand 往左 shift，並且輸出 rand [6:0] 作為亂數

```
reg [20:0] rand;
reg [20:0] next_rand;
wire feed;

assign feed = rand[20] ^ rand[17];
assign out = rand[6:0];

always @ (posedge clk, posedge rst)
begin
   if(rst)begin
      rand <= ~(20'b0);
   end
   else begin
      rand <= next_rand;

   end

end

always @ (*)
begin
   next_rand = {rand[19:0], feed};
end
```

(6) Top.v

Top module 將所有訊號連接，並且處理 backspace 的輸入，當小鳥死掉或遊戲通關後使用者按下 backspace，此時 backspace 訊號拉成 1，直到 process_state 回到 IDLE，backspace 訊號拉回 0

```
always @(*) begin
    next_backspace = backspace;
    if(key_down[9'b001100110] == 1 && been_ready == 1 && (process_state == WIN || bird_state == DEAD))
        next_backspace = 1;
    end
    else if(process_state == IDLE)begin
        next_backspace = 0;
    end
end
```

(7) Audio.v

在這個 module 中，提供的功能相當簡單且實用，也就是在平時輸出背景音樂，並隨時接受撥放不同音效的需求，並在音效結束後，繼續撥放背景音樂。而在這個遊戲中，提供了三種音效，分別為得分、碰撞、通關。其架構與 lab8 大致相似，首先將所有 input 做 debounce 和 onepulse，接著傳進自製的 freq_gen module 中，其會傳回 freqL 和 freqR，最後即可使用 note_gen 和 speaker_control 等 module 產出所需的 input。

freq_gen 為主要控制聲音狀態的 module，裡面有 4 種 state，分別代表撥背景音樂及 3 個不同音效，由下圖可看出任何音效（S1、S2、S3）都可以打斷其他音效而撥出，並在音效撥完後就會回到 S0，也就是不斷循環撥放背景音樂。此外，由於 module 的簡單架構，使我們可以簡單修改就能一直加入新的音效，也很容易與主結構整合起來，整體上相當方便。

而我們所選用的背景音樂為 wii music，我認為他的節奏相當適合於這種電子的像素遊戲中，實際效果也搭配得不錯；得分聲為與原版遊戲相同的簡單兩個音所組成；通關聲為瑪莉歐遊戲的通關聲；最後的碰撞聲反而是較困難的部分，一開始想要模擬出碰撞後的爆炸聲，但發現只靠方波的聲音實在效果有限，後來改為使用一組由多個連續低音所組成的音效，希望製造出失敗的效果，意外的得到不錯的結果。

```
// next_state
always @* begin
    next_state = state;
    case(state)
        S0: begin
            if(in1) next_state = S1;
            else if(in2) next_state = S2;
            else if(in3) next_state = S3;
        end
        S1: begin
            if(beat1 == 4'd15) next_state = S0;
            else if(in2) next_state = S2;
            else if(in3) next_state = S3;
        end
        S2: begin
            if(beat2 == 4'd15) next_state = S0;
            else if(in1) next_state = S1;
            else if(in3) next_state = S3;
        end
        S3: begin
            if(beat3 == 7'd127) next_state = S0;
            else if(in1) next_state = S1;
            else if(in2) next_state = S2;
        end
    endcase
end
```

```
freq_gen f(
    .clk(clk),
    .rst(rst),
    .in1(in1p), // input1
    .in2(in2p), // input2
    .in3(in3p), // input3
    .freqL(freqL), // output1
    .freqR(freqR) //output2
);
```

3. 實作完成度/難易度說明/分工

(1) 實作完成度：我們的完成度遠超當初 propasal 的內容

proposal : 畫面只有鳥和水管，玩家利用空白鍵控制鳥的高度

實際完成：除了上述內容以外，新增以下功能：

- 背景會有飄動的雲
- 增加背景音樂、碰撞音效、通關音效
- 新增遊戲字幕(開始、失敗、通關)
- 水管上下移動的關卡
- 增加 4 種怪物、2 種魔王，並且有攻擊的動畫
- 為遊戲加上結尾(回到鳥巢)
- 在通關(或 game over)後按 backspace 能夠回到初始畫面

(2) 難易度說明：

我自己覺得我們的作品不簡單，但我想老師以及助教看完所有人的作品後，心中自有定論，因此這部份我就不多著墨，畢竟應該不會有人說自己的作品很簡單

(3) 分工：

109062174 謝承恩

- VGA 顯示

108032053 陳凱揚

- 音效模組、圖片素材背景處理

4. 測試完整度：

以我們現有的設計來說，整體應該是沒有 bug 的，至少沒有會影響遊戲體驗的 bug，我們在 demo 前一天不斷測試我們的程式，從晚上 6 點一直到 demo 當天的凌晨 6 點，我們嘗試了各種物體的碰撞是否正常(五根管子、植物、蜘蛛、幽靈、子彈、魔法球、香蕉)，然後嘗試了如果小鳥中途沒有死亡，順利通過第一輪，第二輪所有東西也都可以正常運作，第三輪也沒問題，以及小鳥如果在第一輪中途死掉後，重新開始一輪也沒有問題，接下來第二、三輪也沒有問題，以及開啟無敵模式跑一輪也沒有問題、開啟無敵模式在中途切換成碰撞模式也確實會發生碰撞，以及在各種關卡死掉後按 backspace 也都能順利回到初始畫面。

5. 困難與解決辦法

(1) Vga 顯示：我覺得這部份本身就是一個很大的挑戰，因為牽涉到很多物件同時在螢幕上移動，但我覺得這真的就是一步一步慢慢來，從一開始只有一張背景圖案，然後加上一隻鳥，再加上一根水管…，所有東西都是這樣慢慢加上去，到後來越來越熟悉後，要加新的東西就會比較快，

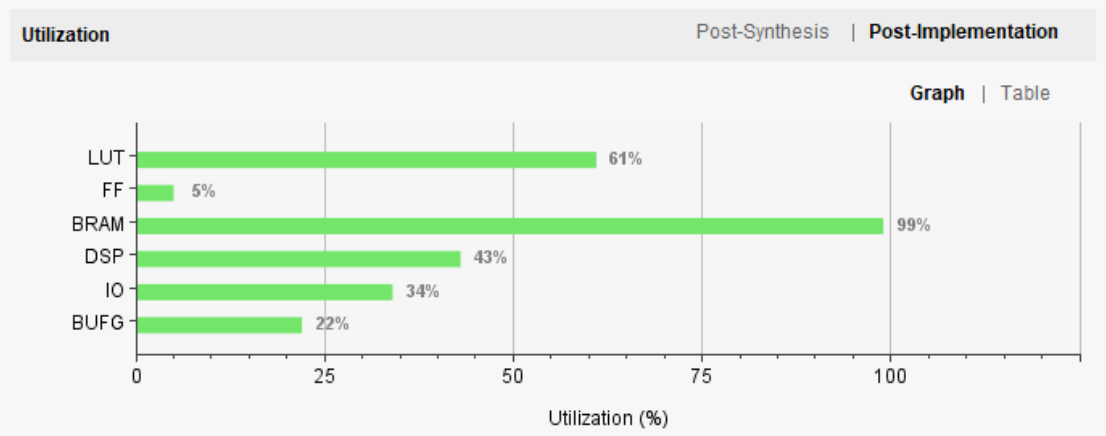
(2) 水管是隨機產生的，但是水管的圖片是固定大小，因此要讓程式動態決定水管顯示的區域，其實只要有水管的 y 座標就能夠做到，因為有水管的 y 座標就能夠知道水管開口的位置，等 v_cnt 數到水管開口的位置時，就將原圖的水管開口印出，否則就一直重複的印水管壁

```
always @(*)begin
    if(v_cnt > (tube1_y + 80) + 16)begin
        add_13 = 19*60 + (h_cnt - tube1_x + 30);
    end
    else begin
        add_13 = (v_cnt - (tube1_y+ 80 + 10) + 10)*60 + (h_cnt - tube1_x + 30);
    end
end
```

(3) 物件重疊時要讓畫面看起來很流暢：

對於每種可能重疊的物件(不一定是碰撞)，都要做處理，否則重疊時會看起來圖片有破損的感覺，只要當物件發生重疊時，將上層圖像的背景替換成下層物件的圖像，而不是輸出上層圖像原本的背景，然後因為可能發生重疊的物件有蠻多種的，所以要一一去做處理，但邏輯都一樣，所以像顯示小鳥的部分就會多很多判斷式，目的就是為了處理小鳥和其他物件重疊的問題。

(4) 記憶體空間不足：我們總共是用了 51 個 BRAM，因為有小鳥、水管、怪物(4 種)、雲朵(6 朵)、魔王(2 種)、遊戲字幕(4 種) …，此外小鳥又分兩張飛行圖案、兩張死亡圖案，有些怪物要會動，因此也要不同圖案，然後猴子是花最多記憶體的，總共 11 張圖案，因為要讓牠呈現非常流暢的攻擊動畫，這些東西全部加起來要不少記憶體，但就是用剛才講的方法，可以重複畫的東西就不要儲存，最後我們成功將記憶體使用率控制在 99%



(5) 當遊戲結束後可以回到初始畫面，並初始化所有物件，開始第二輪遊戲：這部分我們真的花很多時間處理，但是我們堅持要將這個功能做出來，因為學期初老師就有說過，你的設計必須要能夠從最後一個狀態回到初始狀態，不能每次遊戲結束都要按 reset 才能重新開始，這樣是不合理的設計。其實這部分就是要仔細檢查每個 module 的每個物件、每個變數，在初始狀態應該要在什麼位置、是什麼值，不能有些東西 reset 了，有些卻還沒，但因為我們的物件真的太多了，過程中一直有一些物件沒有及時被 reset，最後花了很多時間才讓所有東西都能夠正確的歸位。

6. 心得討論：

做完這個 final project 我覺得最大的收穫就是對 vga 的控制變得非常熟悉，加上我們的程式算是很龐大，每個東西都有自己的狀態(鳥的狀態、關卡的狀態、魔王的狀態…)，要怎麼讓整個關卡流程及每個物件按照我們的想法去進行，真的是一件不容易的事，尤其是遊戲結束後要將所有物件的位置、狀態回復，這部份我們處理了很久，我覺得很多時候問題本身並不是很困難，真正困難的是當程式碼到達 4, 5 千行的時候，很多東西都會互相影響，這個時候就很難找到真正引發問題的所在，有時候甚至要退回上一個版本重新再寫一遍。但是有了這種比較大型的程式開發經驗，我覺得自己的實力真的提升許多，我對待這個作品的態度是真的把它當作一個遊戲來開發，不僅僅是一個作業而已，我希望能夠做出讓人眼睛為之一亮的作品。

7. 想對老師或助教說的話

真的很感謝老師與助教這麼用心的指導我們，除了上課時間與 demo 時間外，在 eeclass 的討論區也一定會回覆我們的問題，整個學期下來收穫非常多，尤其最後做 final project 的部分，在學期初的時候我並不覺得自己能做出這樣的作品，最後我們突破了原先對自己的期待，看到成品的當下真的有些感動，而 lab 的部分，我覺得難易度適中，不至於太難，但還是要花一些時間寫，這門課除了讓我在這個領域打下扎實的基礎外，也讓我對數位設計產生興趣，未來我會想繼續修其他相關課程(DIC lab, VLSI system design…)

8. 笑話