ชื่อกลุ่ม Micky บวกลบเปรียบเทียบ

สมาชิก

640 Tiw

650 Mon

685 pitch

688 ball

690 autt

ที่มา

เนื่องมาจากกลุ่มของพวกเราเห็นว่าคณิตศาสตร์นั้นได้มีความจำเป็นอยู่อย่างมาก ซึ่งในวัยเด็กนั้น การเรียนรู้ การฝึกด้วยความสนุก จะทำให้เกิดการจดจำที่ดีกว่า และในเด็กวัยนี้ ความสนุกต่างๆส่วนหนึ่งก็มาจากการเล่นกับเพื่อนๆ ทางกลุ่มของเราจึงได้คิดและทำเกมคณิตศาสตร์ และจะต้องมีการแข่งกับเพื่อนขึ้นมา จึงได้เกมเปรียบเทียบตัวเลข บวก ลบ มากกว่า น้อยกว่า โดยผู้เล่นสองคนจะต้องแข่งกันขึ้นมา

วิธีการเล่นเกม

ที่เครื่อง จะมีดวงไฟหลักๆ 4 จุด

-ที่หู ใช้นับคะแนนผู้เล่น

-ที่หัว ใช้แสดงคำถาม

-ที่เครื่องหมาย “=” ใช้แสดงคำถาม

-ตัวเลข ใช้แสดงคำถาม

การเล่นนั้น ที่จอย จะมีปุ่ม 4 ปุ่ม บวก ลบ มากกว่า น้อยกว่า โดยหากไฟที่เครื่องหมายเท่ากับ ไม่ติดขึ้น ผู้เล่นจะต้องแข่งกันวิเคราะห์ตัวเลขบนหัวของเครื่อง เพื่อจะต้องกดเครื่องหมาย

มากกว่า น้อยกว่า ที่จอยเกม นำไปใส่ระหว่างกลางของตัวเลขทั้งสอง เพื่อเสดงความสัมพันธ์ของตัวเลขให้ถูกต้อง

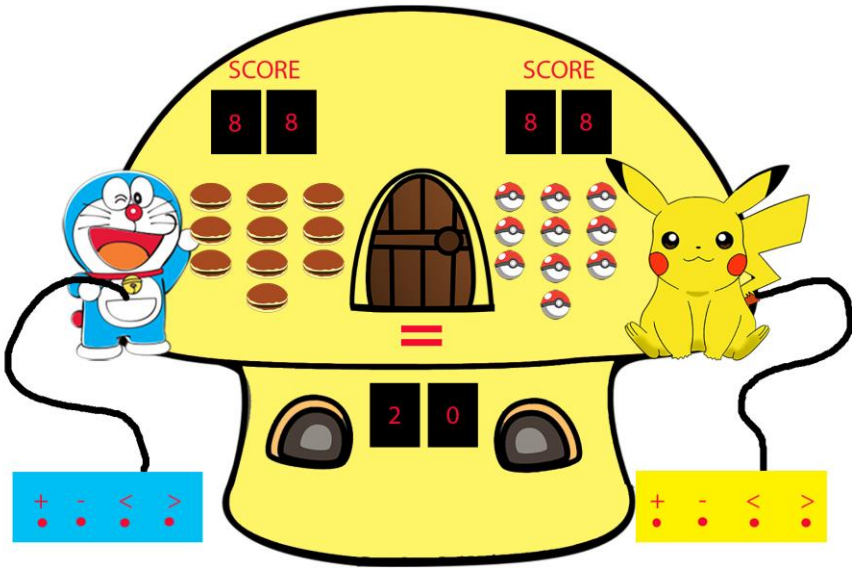
แต่หากไฟที่เครื่องหมาย เท่ากับ ติดเป็นสีแดง ผู้เล่นจะต้องดูว่า ตัวเลขที่ขึ้นที่ช่องตัวเลขนั้นเป็นคำตอบของสมการของการนำตัวเลขสองด้านข้างต้นมาทำการ บวก หรือ ลบ กัน จึงจะได้ตัวเลขที่แสดง ผู้แล่นจะต้องแข่งกันกดคำตอบที่จอยเช่นเดิม

หากผู้เล่นฝ่ายใดผ่านหนึ่งกดคำตอบที่ถูกก่อน ดวงไฟที่หูของผู้ล่นฝั่งนั้นจะติดขึ้นมา แสดงคะแนนตามจำนวนข้อที่กดถูก ผู้เล่นฝ่ายใดผ่ายหนึ่งที่กดได้ถูกต้องจนครบ 5 ครั้ง จะเป็นผู้ชนะ และตัวเกมจะไม่ยอมให้อีกฝ่ายได้กดตอบคำถามต่อไป

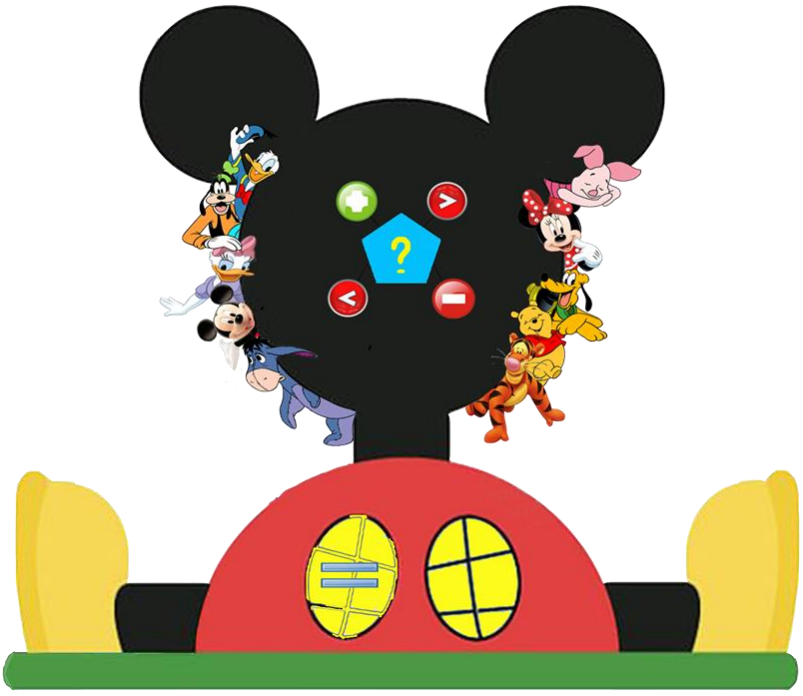
การออกแบบ

ภาพรวมและขั้นตอนการออกแบบ

เริ่มต้นจากรูปนี้เป็น ideaในการวางตัว LED และ7-segment



แล้วเปลี่ยนมาเป็น Mickey mouse'club house เพราะเป็นตัวการ์ตูนที่น่ารัก เด็กรู้จักดี และมีรูปร่างพอดีกับการวางอุปกรณ์ที่กลุ่มเราต้องการ



อุปกรณ์ในการทำรูปแบบชิ้นงาน

1.กล่องกระดาษขนาดใหญ่

2.ฟิวเจอร์บอร์ด ขนาดกลาง 2แผ่น

3.กระดาษสีดำ,เหลืองและแดง แผ่นใหญ่

4.กระดาษที่ปริ้นรูภาพตัวการ์ตูนต่างๆ

5.วงเวียน,กาว,กรรไกร,คัดเตอร์,สีเมจิก,ไม้บรรทัด,

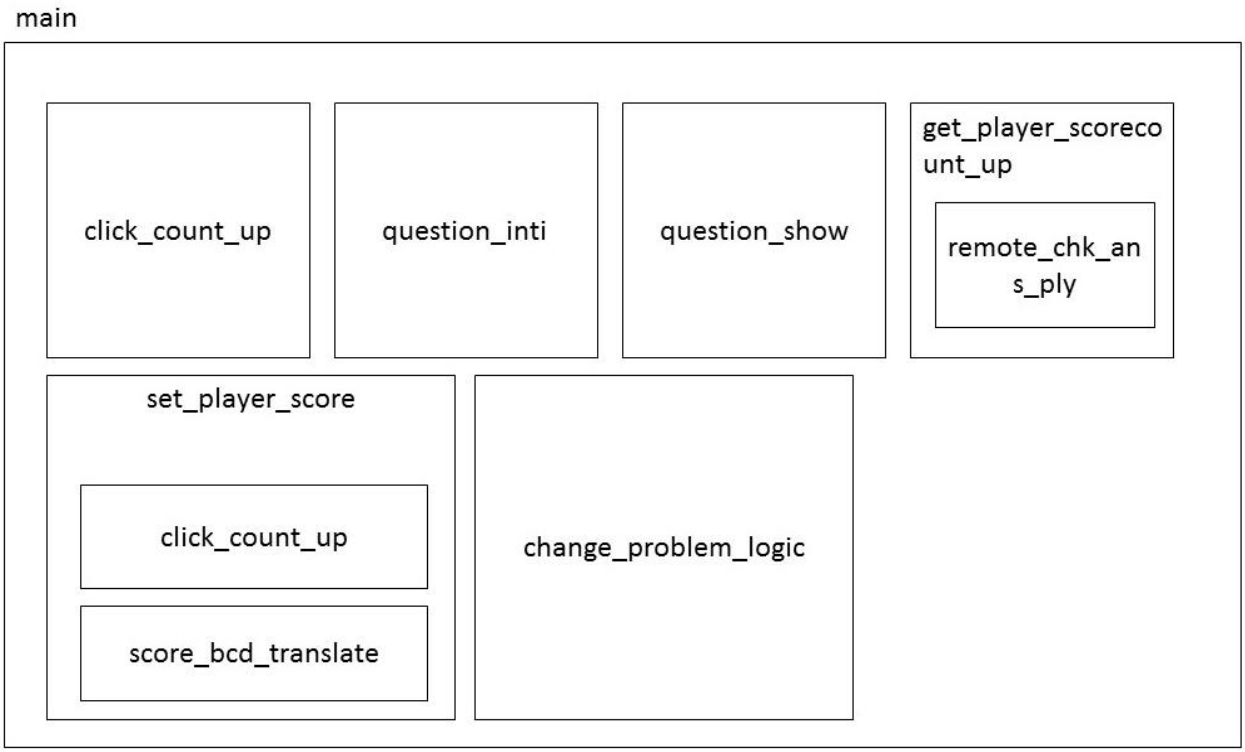
6.กล่องกระดาษแข็งขนาดเล็ก

9.เทปใส,เทปดำ

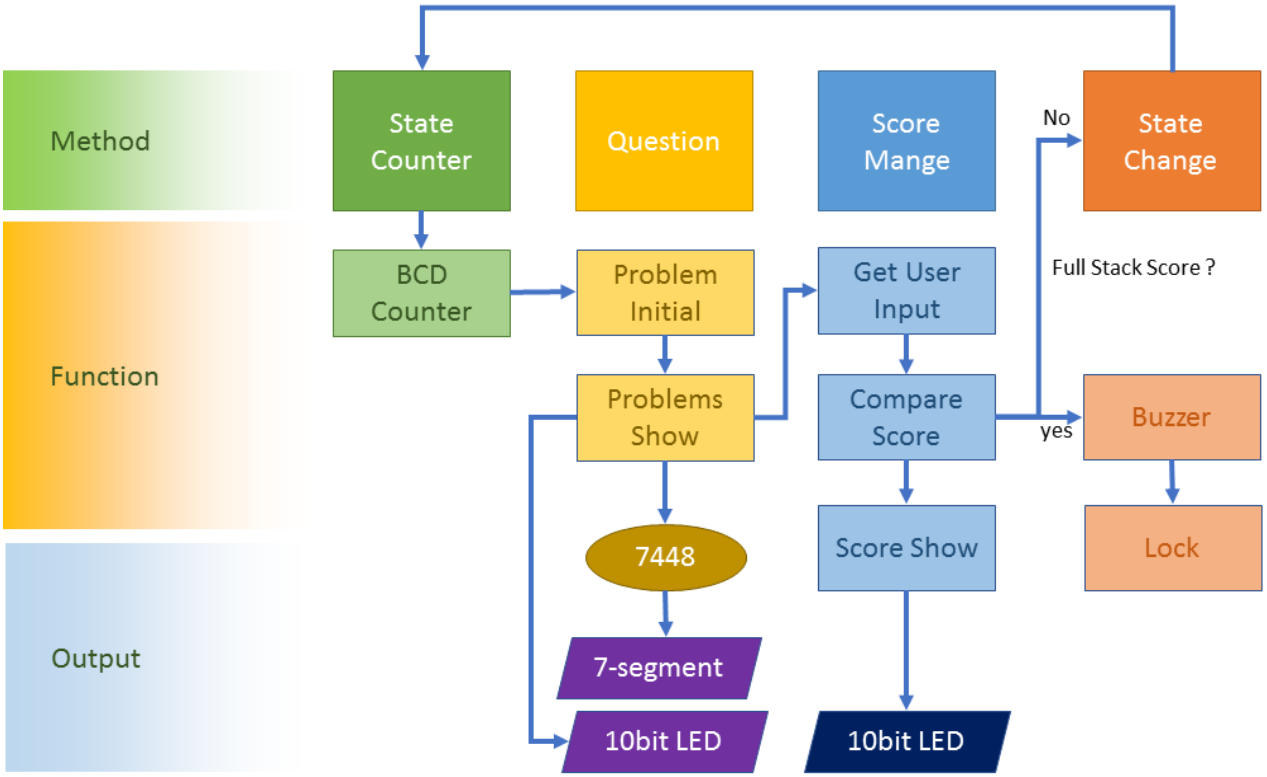
10.กระดาษทิชู

11.แผ่นสะท้อนแสง

การออกแบบ Box Diagram



ภาพรวมการออกแบบส่วน Coding



วิธีการประกอบชิ้นงาน

1.ตัดกล่องกระดาษเป็นรูปMickey mouse'club house

2..แปะรูปตัวการ์ตูนต่างๆที่กล่องกระดาษแข็งแล้วตัดแยกออกเป็นตัวๆ

3.นำรูปตัวการ์ตูนที่ตัดแล้วไปติดกับกล่องกระดาษรูปmickey mouse

4.ตัดการดาษสีต่างๆเป็นรูปmickey mouse แล้วใช้กาวแปะตามรูปที่ออกแบบไว้

5.ใช้ฟิวเจอร์บอร์ดตัดและประกอบเป็นกล่องแบบมีฝาปิดครึ่งหนึ่งแล้วนำไปติดด้านหลังของ mickey mouse

6.เจาะรูเครื่องหมายเท่ากับและตำแหน่งที่ใส่ 7-segment ตามรูที่ออกแบบไว้

7.นำกล่องกระดาษขนาดเล็กมาเจาะรูติดแผ่นสะท้อนแสงใส่LED แล้วนำกระดาษทิดชูมาปิดที่รูทับไว้แล้วจึงนำไปติดในกล่องบริเวณช่องแสดงเครื่องหมายเท่ากับ

8.ตัดฟิวเจอร์บอร์ดติดด้านหลังmickey mouse เพื่อยึดโครงสร้างให้แข็งแรง

9.ใส่ตะเกียบค่ำด่านหลังเพื่อเพิ่มความแข็งแรง

10.เจาะรูให้สายที่ใช้สำหรับแป่นกดทั้ง2ข้างออกมาจากกล่องด้านหลัง

11.ติดLED ,7-segment และสายไฟ ทุกอุปกรณ์เข้ากับบอร์ดเพื่อให้เกมส์ทำงานได้

ผลการทำงาน

การทำงานของเครื่องเป็นที่น่าพอใจ ตัวเครื่องทำงานตามที่ออกแบบไว้ได้อย่างดี แต่แล้วปัญหามักจะเกิดตอนที่เปิดเครื่อง เครื่องเกมของกลุ่มเรานั้น ในบางครั้ง เมื่อเกิดเครื่อ เครื่องจะไม่เริ่มที่ state แรกตามที่ออกแบบ แต่คำถาม คำตอบและการนับคะแนนที่ตั้งไว้ ทำงานตาม stateได้ตามปกติ เพียงแต่จะไม่เริ่มที่คำถามแรกตลอดเท่านั้น ปัญหานี้แก้ได้โดยเบิร์นโค้ดตัวเดิม ลงไปในเครื่องซ้ำอีกครั้ง ก็จะสามารถแก้ปัญหานี้ได้

หากสังเกตุดีๆ คำตอบที่ช่องตัวเลขจะเรียงกัน 0-9 แท้จริงแล้วขั้นแรกในโค้ด เราได้เขียนโค้ดไว้เพื่อทดสอบการทำงานของเครื่อง ตัวเลขนั้นไว้ดูstate ว่าอยู่ในstateใด แต่เมื่อเราตั้งใจจะเปลี่ยนเป็นโจทย์จริงๆ เครื่องกลับทำงานผิดพลาดอย่างไม่น่าเชื่อ ดังนั้น เราจึ่งทำการเปลี่ยนโจท์ให้เข้ากับตัวเลข state แทน ก็เป็นการแก้ปัญหาอย่างหนึ่งเพื่อให้เครื่องทำงานได้

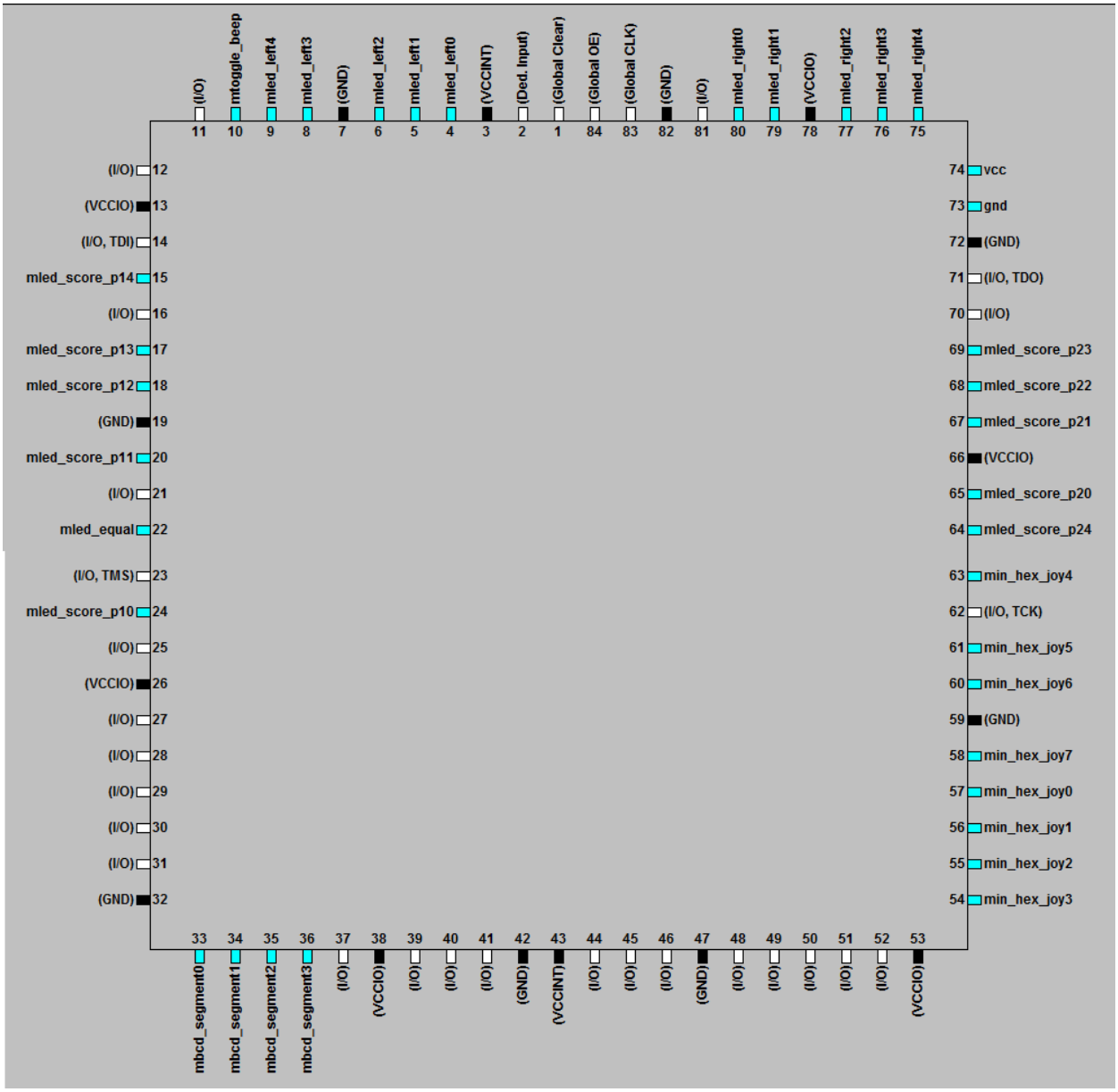
สรุปผล

ขณะทำการเล่น เครื่องเกมที่สร้างขึ้นทำงานได้เป็นอย่างดี ปัญหาไม่ได้เกิดขึ้น เด็กๆที่ได้เล่นกับเพื่อนก็ยิ้มแย้ม ได้รับความสนุก ความรู้และการฝึกฝนทางคณิตศาสตร์เบื้องต้นและอาจได้รับแรงบันดาลใจในการเรียนต่อไป

ภาคผนวก

Verilog code

https://github.com/kandation/CPE212_project/blob/master/05_main_v0/main.v



หากคอมไพล์ไม่ผ่าน maxplusII บอกว่าให้ใช้ clock ที่เตรียมไว้ ให้ไปที่

Assign > Global Project Synthesis > Automatic Global ติ๊ก clock ออกไปเพื่อไม่ให้ max plus บังคับเราใช้clock ของระบบ

```
module main (vcc, gnd, mled_left,
mtoggle_beep, mled_right, mled_equal,
mbcd_segment, mled_score_p1,
mled_score_p2, min_hex_joy);

//Not finished @091259 0130
  output [4:0]mled_left,mled_right,
mled_score_p1, mled_score_p2;
  output mled_equal, vcc, gnd,
mtoggle_beep;
  output [3:0]mbcd_segment;
  //output [3:0]aut_segment;
  input [7:0]min_hex_joy;

  wire [3:0]wo_bcd_left, wo_bcd_right,
wo_bcd_segment, wo_ans;
  wire wo_tog_equal;
  wire [3:0]wo_state;
  wire inw_trigger;

  wire wo_ret_score_p1,
wo_ret_score_p2;

  wire [3:0]w_anssel;
  wire [1:0]w_player;

  reg vcc, gnd;

  wire wo_toggle_beep;

//Global Pin for extranal
  always @ ( 1 ) begin
    vcc = 1;
    gnd = 0;
  end

  //starting state with zero
  click_count_up cup(wo_state,
inw_trigger);

  //initial question
  //module question_inti (bcd_left,
bcd_right, tog_equal, bcd_segment,
bcd_ans, state);
  question_inti q1(wo_bcd_left,
wo_bcd_right, wo_tog_equal,
wo_bcd_segment, wo_ans, wo_state);

  //show_aut_segment
a1(aut_segment,wo_bcd_segment);
```

```
  //Show question in display
  //module question_show (led_left,
led_right, led_equal, bcd_segment,
in_left, in_right, in_equal, in_segment);
  question_show qs1(mled_left,
mled_right, mled_equal,
mbcd_segment, wo_bcd_left,
wo_bcd_right, wo_tog_equal, wo_state);

  //LocalMethod: wait player press
buttom
  //module
get_player_score(ret_score_p1,
ret_score_p2, rm_in_bcd, prob);
  get_player_score
gps1(wo_ret_score_p1,
wo_ret_score_p2, min_hex_joy, wo_ans,
wo_toggle_beep);

  //module
set_player_score(toggle_beep,
led_score_p1, led_score_p2,
tog_score_p1, tog_score_p2);
  set_player_score
sps1(wo_toggle_beep, mled_score_p1,
mled_score_p2, wo_ret_score_p1,
wo_ret_score_p2);

  and(mtoggle_beep, wo_toggle_beep,
wo_toggle_beep);

  //LocalMethod: change state
  //module
change_problem_logic(is_active,
score_p1, score_p2);
  change_problem_logic
ch1(inw_trigger, wo_ret_score_p1,
wo_ret_score_p2);


endmodule // main

module show_aut_segment(bcd,in);
  output [3:0]bcd;
  input [3:0]in;
  reg [3:0]bcd;
  always @(1) begin
    bcd = in;
```

```
  end
endmodule

module question_inti (bcd_left,
bcd_right, tog_equal, bcd_segment,
bcd_ans, bcd_state);
  output [3:0]bcd_left, bcd_right;
  output [3:0]bcd_segment, bcd_ans;
  output tog_equal;
  input [3:0]bcd_state;

  reg [3:0]bcd_left, bcd_right;
  reg [3:0]bcd_segment, bcd_ans;
  reg tog_equal;


  always @ (bcd_state) begin
    if (bcd_state == 0) begin
      bcd_left = 3;
      bcd_right = 2;
      tog_equal = 0;
      bcd_segment = 3;
      bcd_ans = 3;
    end
    else if (bcd_state == 1) begin
      bcd_left = 2;
      bcd_right = 1;
      tog_equal = 1;
      bcd_segment = 2;
      bcd_ans = 2;
    end
    else if (bcd_state == 2) begin
      bcd_left = 1;
      bcd_right = 1;
      tog_equal = 1;
      bcd_segment = 4;
      bcd_ans = 1;
    end
    else if (bcd_state == 3) begin
      bcd_left = 2;
      bcd_right = 5;
      tog_equal = 0;
      bcd_segment = 0;
      bcd_ans = 4;
    end
    else if (bcd_state == 4) begin
      bcd_left = 5;
      bcd_right = 1;
      tog_equal = 1;
      bcd_segment = 6;
```

```
  bcd_ans = 2;
end
else if (bcd_state == 5) begin
  bcd_left = 1;
  bcd_right = 4;
  tog_equal = 1;
  bcd_segment = 4;
  bcd_ans = 1;
end
else if (bcd_state == 6) begin
  bcd_left = 5;
  bcd_right = 1;
  tog_equal = 0;
  bcd_segment = 3;
  bcd_ans = 3;
end
else if (bcd_state == 7) begin
  bcd_left = 4;
  bcd_right = 3;
  tog_equal = 1;
  bcd_segment = 2;
  bcd_ans = 1;
end
else if (bcd_state == 8) begin
  bcd_left = 1;
  bcd_right = 2;
  tog_equal = 0;
  bcd_segment = 2;
  bcd_ans = 4;
 end
 else if (bcd_state == 9) begin
  bcd_left = 4;
  bcd_right = 5;
  tog_equal = 0;
  bcd_segment = 4;
  bcd_ans = 1;
 end
 /*else begin
  bcd_left = 10;
  bcd_right = 10;
  tog_equal = 10;
  bcd_segment = 0;
  bcd_ans = 10;
 end */
 end
endmodule // question_inti

module question_show (led_left,
led_right, led_equal, bcd_segment,
in_left, in_right, in_equal, in_segment);
```

```
/*
  low-level module to show 5 digits led-problem and equal and segment
  @input [bcd]data and [active hight]in_equal
  @output [active hight]all led
  @delay xxx ns
  @
*/
  output [4:0]led_left, led_right;
  output [3:0]bcd_segment;
  output led_equal;
  input [3:0]in_left, in_right, in_segment;
  input in_equal;

  reg [4:0]led_left, led_right;
  reg [3:0]bcd_segment;
  reg led_equal;
  always @(1) begin
    case (in_left)
      0: led_left = 5'b00000;
      1: led_left = 5'b00001;
      2: led_left = 5'b00011;
      3: led_left = 5'b00111;
      4: led_left = 5'b01111;
      5: led_left = 5'b11111;
      //10: led_left = 5'b10101; //State test
      default: led_left = 5'b01010;
    endcase
  end

  always @ (1)
  begin

    case (in_right)
      0: led_right = 5'b00000;
      1: led_right = 5'b00001;
      2: led_right = 5'b00011;
      3: led_right = 5'b00111;
      4: led_right = 5'b01111;
      5: led_right = 5'b11111;
      //10: led_right = 5'b10101; //State test
      default: led_right = 5'b01010;
    endcase
  end

  always @ (1) begin
    bcd_segment = in_segment;
    led_equal = in_equal;
```

```
  end

endmodule // question_show

module score_bcd_translate (led_player,
bcd_player_score);
/*
  low-level module to show 5 digits led-
score
  @output [5bit][active hight] led array
  @input trigger
  @design this function for Right-hand
player if you want
    to use LH player you should invert-
side of LED ex
    12345 -> 54321 it's eazy ways
  @delay 7.2ns [091259 0011]
*/
  input [3:0]bcd_player_score;
  output [4:0]led_player;
  reg [4:0]led_player;


  //Working always
  always @ (1) begin
    case (bcd_player_score)
      0: led_player = 5'b00000;
      1: led_player = 5'b00001;
      2: led_player = 5'b00011;
      3: led_player = 5'b00111;
      4: led_player = 5'b01111;
      5: led_player = 5'b11111;
      default: led_player = 5'b00000;
    endcase
  end
endmodule

module click_count_up(bcd, trigger);
  output [3:0]bcd;
  input trigger;
  reg [3:0]bcd;


  always @ ( posedge trigger ) begin
    case (bcd)
      0: bcd = 1;
      1: bcd = 2;
      2: bcd = 3;
      3: bcd = 4;
      4: bcd = 5;
```

```
      5: bcd = 6;
      6: bcd = 7;
      7: bcd = 8;
      8: bcd = 9;
      9: bcd = 0;
      default: bcd = 0;
    endcase
  end
endmodule
module
change_problem_logic(is_active,
score_p1, score_p2);
/*
  low-level module for compair 2 player
score for send toggle data next problem
  @output [1-bit] 1 if one of person have
corrcet answer
  @input [1bit][active hight] one score of
perple or both
*/
  output is_active;
  input score_p1, score_p2;
  or(is_active, score_p1, score_p2);

endmodule
```

```
module set_player_score(toggle_beep,
led_score_p1, led_score_p2,
tog_score_p1, tog_score_p2);
  output [4:0]led_score_p1,
led_score_p2;
  output toggle_beep;
  input tog_score_p1, tog_score_p2;
  reg toggle_beep;
  wire [3:0]score_p1, score_p2;

  click_count_up cku_p1(score_p1,
tog_score_p1);
  click_count_up cku_p2(score_p2,
tog_score_p2);

  score_bcd_translate
sbcdt1(led_score_p1, score_p1);
  score_bcd_translate
sbcdt2(led_score_p2, score_p2);

  always @(1) begin
    if (score_p1 >= 5 || score_p2 >= 5)
begin
      toggle_beep = 1;
```

```
    end
    else begin
      toggle_beep = 0;
    end
  end

endmodule



module get_player_score(ret_score_p1,
ret_score_p2, rm_in_bcd, prob,
toggle_beep);
/*
  check player answer with problem
answer and get who player give score
  @output [active hight] 1-bit
score_player1 OR score_player2
  @input [bcd] remote_raw_bcd ,
problem_bcd_answer
*/
  output ret_score_p1;
  output ret_score_p2;
  input [7:0]rm_in_bcd;
  input [3:0]prob;
  input toggle_beep;
  wire [3:0]player_ans;
  wire [1:0]players;

  reg ret_score_p1, ret_score_p2;
  remote_chk_ans_ply rcap(player_ans,
players, rm_in_bcd);

  always @(1)
  begin
    if (player_ans == prob &&
!toggle_beep)
      begin
        case (players)
          1:ret_score_p1 = 1'b1;
          2:ret_score_p2 = 1'b1;
          default:
            begin
              ret_score_p1 = 1'b0;
              ret_score_p2 = 1'b0;
            end
        endcase
      end
    else
      begin
        ret_score_p1 = 1'b0;
```

```
        ret_score_p2 = 1'b0;
      end
  end
endmodule

module remote_chk_ans_ply(anssel,
player, in_hex);
/*
  check answer from remote both
  @retrun choice 1-4 [3:0]and player 1-2
[1:0]
  @input [active low] raw bcd[7:0] from
remote
*/
  output [3:0]anssel;
  output [1:0]player;
  input [7:0]in_hex;

  reg [3:0]anssel;
  reg [1:0]player;

  always @(1)
    begin
      case (in_hex)
        8'b11111110 :
begin
  anssel = 4;
  player = 2;
end
8'b11111101 :
begin
  anssel = 3;
  player = 2;
end
8'b11111011 :
begin
  anssel = 2;
  player = 2;
end
8'b11110111 :
begin
  anssel = 1;
  player = 2;
end
8'b11101111 :
begin
  anssel = 4;
  player = 1;
end
8'b11011111 :
```

```
begin
  anssel = 3;
  player = 1;
end
8'b10111111 :
begin
  anssel = 2;
  player = 1;
end
8'b01111111 :
begin
  anssel = 1;
  player = 1;
end
default:
  begin
    anssel = 0;
    player = 0;
  end
endcase
end
endmodule
```