hw0 說明文件

組別:第14組

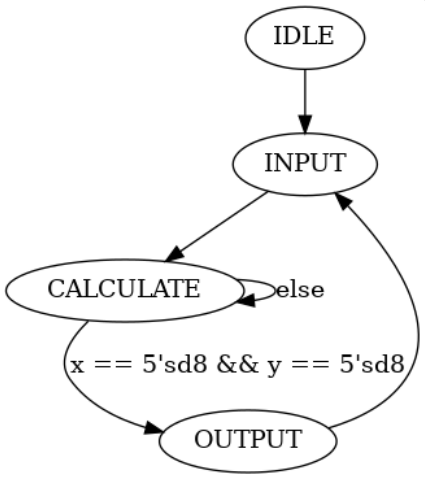
組員:黃偉峰E34106010 / 陳識博E94106096 / 黃芊 F74104040

電路設計説明

Finite State Machine

```
parameter
IDLE = 2'd0,
INPUT = 2'd1,
CALCULATE = 2'd2,
OUTPUT = 2'd3;

always @(*) begin //next_state
    case(state)
        IDLE: next_state = INPUT;
        INPUT: next_state = CALCULATE;
        CALCULATE:begin
            if(x == 5'sd8 && y == 5'sd8) next_state = OUTPUT;
            else next_state = CALCULATE;
        end
        OUTPUT: next_state = INPUT;
    endcase
end
```



根據題目要求我們將電路分為四個state，左圖分別有狀態機verilog code和FSM。

FSM 包含 **四個狀態**：

1. IDLE: reset 完成後，FSM 進入 IDLE 狀態，等待 Testbench提供 testdata
2. INPUT:接收testdata
3. CALCULATE: 從(x,y) = (1,1) 走到(x,y) = (8,8)的過程中根據mode的選擇計算candidate數量
4. OUTPUT: 輸出 candidate 數量，然後回到 INPUT 狀態。

(x, y) 移動

$$\left\{ \forall x, y \mid (x-x1)^2 + (y-y1)^2 \le r^2 \mid x, y, r \in Z \right\} \quad \begin{matrix} 1 \le x \le 8 \\ 1 \le y \le 8 \end{matrix}$$

```
always @(posedge clk) begin //x y
    case(state)
        INPUT:begin
            x <= 5'sd1;
            y <= 5'sd1;
        end
        CALCULATE:begin
            if(x == 5'sd8)begin
                x <= 5'sd1;
                y <= y + 5'sd1;
            end else begin
                x <= x + 5'sd1;
            end
        end
    endcase
end
```

題目要求計算的x, y範圍為1 <= x <= 8, 1 <= y <= 8

- INPUT state下先將x, y設定到(1,1)上
- CALCULATE state下則持續改變 x, y的值，掃描方式為：
  - 若 x = 8，則將 x 重置為 1，並使 y 加 1，換行掃描
  - 否則，將x增加 1，繼續同一行掃描
- 掃描過程模擬從左到右、由上到下的遍歷方式，確保所有 (x, y)座標在指定範圍內依序變化

Candidate 計算

$$\left\{ \forall x, y \,|\, (x-x1)^2 + (y-y1)^2 \le r^2 \,|\, x, y, r \in Z \;\; \begin{matrix} 1 \le x \le 8 \\ 1 \le y \le 8 \end{matrix} \right\}$$

```
assign In_1 = (((x)-(x1)) * ((x)-(x1)) + ((y)-(y1)) * ((y)-(y1)) - r1 * r1) <= $signed(0) ? 1 : 0;
assign In_2 = (((x)-(x2)) * ((x)-(x2)) + ((y)-(y2)) * ((y)-(y2)) - r2 * r2) <= $signed(0) ? 1 : 0;
assign In_3 = (((x)-(x3)) * ((x)-(x3)) + ((y)-(y3)) * ((y)-(y3)) - r3 * r3) <= $signed(0) ? 1 : 0;

always @(posedge clk) begin //candidate
    case(state)
        INPUT: candidate <= 0;
        CALCULATE:begin
            case(mode_reg)
                2'b00:begin
                    if(In_1) candidate <= candidate + 1;
                end
                2'b01:begin
                    if(In_1 && In_2) candidate <= candidate + 1;
                end
                2'b10:begin
                    if((In_1 && !In_2) || (!In_1 && In_2)) candidate <= candidate + 1;
                end
                2'b11:begin
                    if(((In_1 && In_2) || (In_2 && In_3) || (In_1 && In_3)) && !(In_1 && In_2 && In_3)) candidate <= candidate + 1;
                end
            endcase
        end
    endcase
end
```

根據左圖題目提供公式改寫成Verilog code，三條訊號線In_1,In_2,In_3分別計算(x,y)是否在題目提供的三顆圓圈內。

透過In_1,In_2,In_3三條訊號線來實作題目要求的mode，若該情況下的(x,y)符合該mode的點則將candidate累加1。

Mode 2’b00: A => In_1

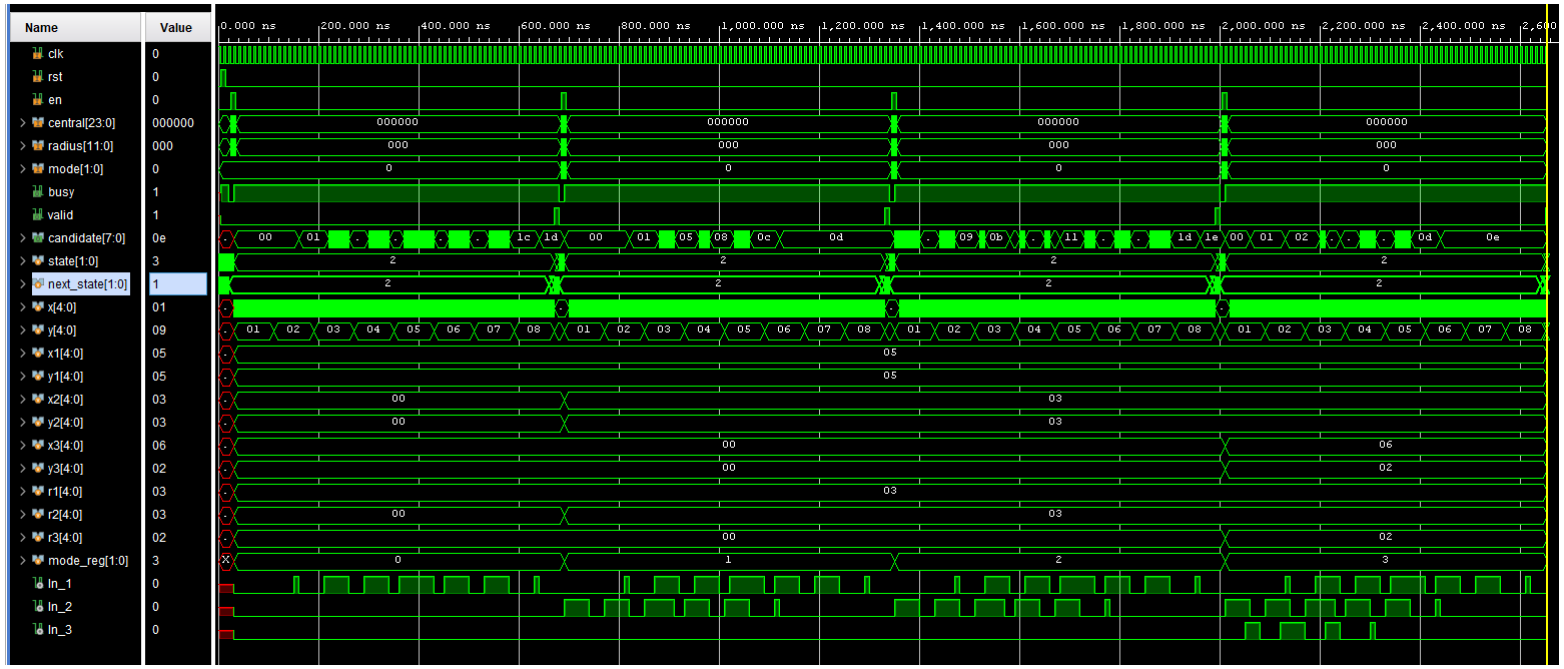
Mode 2’b01: (A∩B) => (In_1 && In_2)

Mode 2’b10: (A∪B)-(A∩B) => (In_1 && !In_2)||(!In_1 && In_2)

Mode 2’b11: (A∩B)+(B∩C)+(A∩C)-(A∩B∩C) =>(In_1 && In_2) || (In_2 && In_3) || (In_1 && In_3)) && !(In_1 && In_2 && In_3)

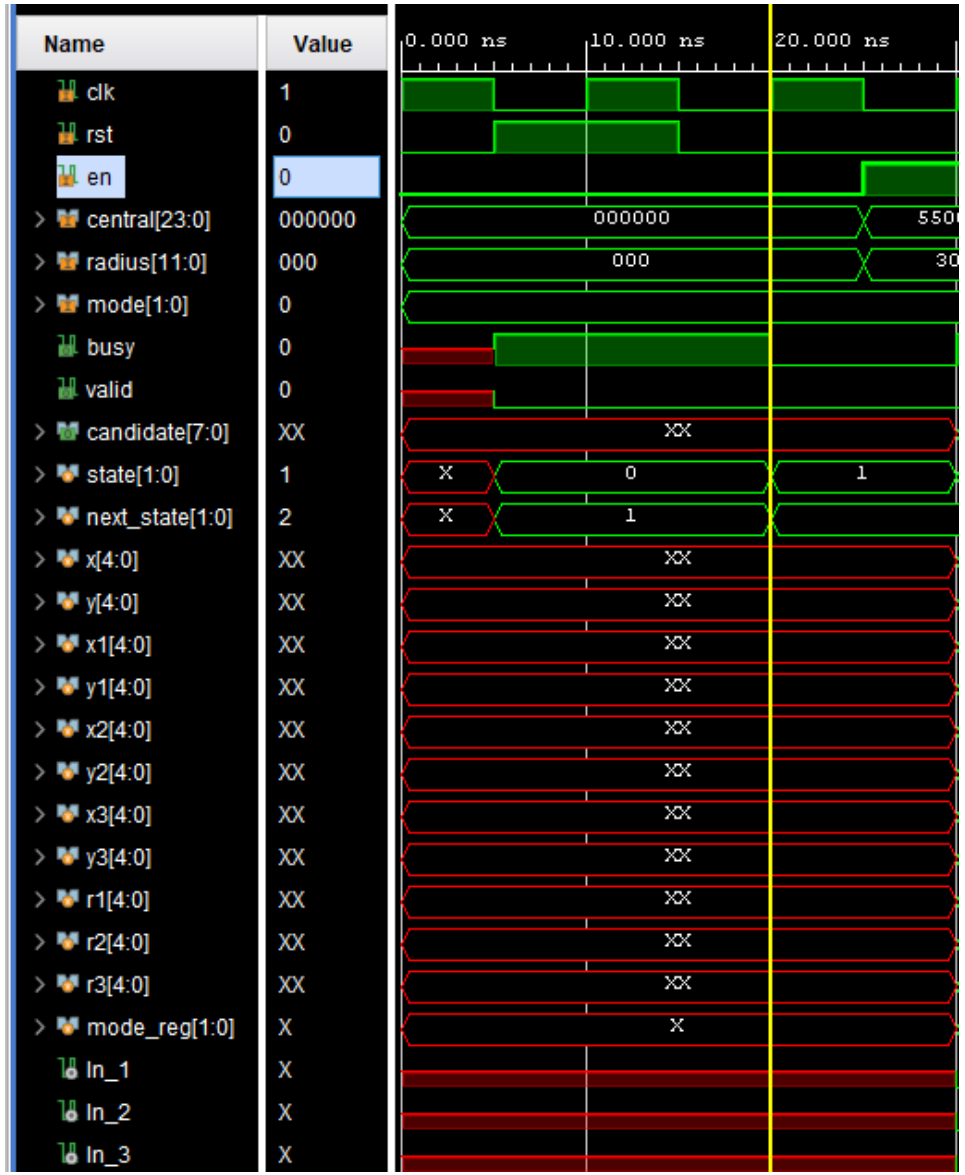
電路模擬結果

Functional Simulation



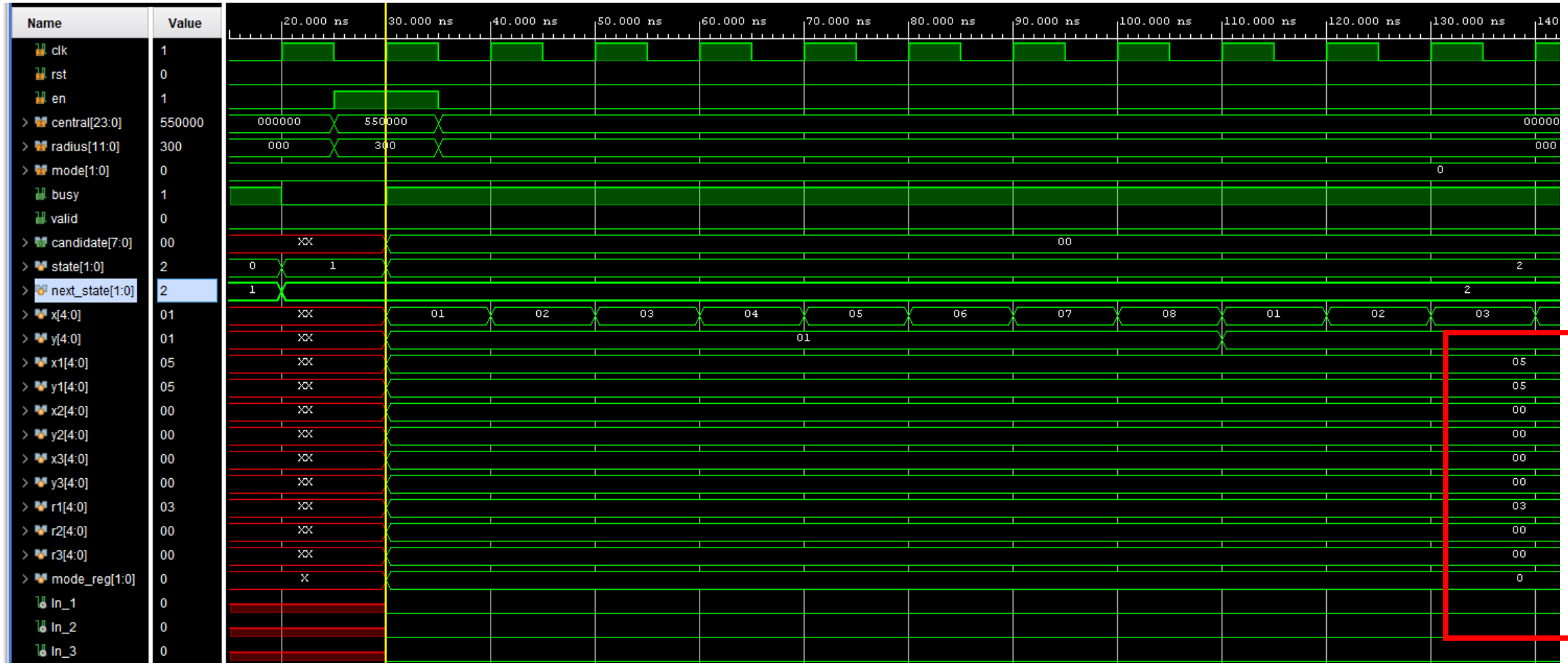
經過Testbench當中四種mode的各一筆testdata後的整體波形圖

```
INFO: [Wavedata 42-604] Simulation restarted
run all
  Mode            0 is correct!
  Mode            1 is correct!
  Mode            2 is correct!
  Mode            3 is correct!
$finish called at time : 2655 ns : File "C:/U
```

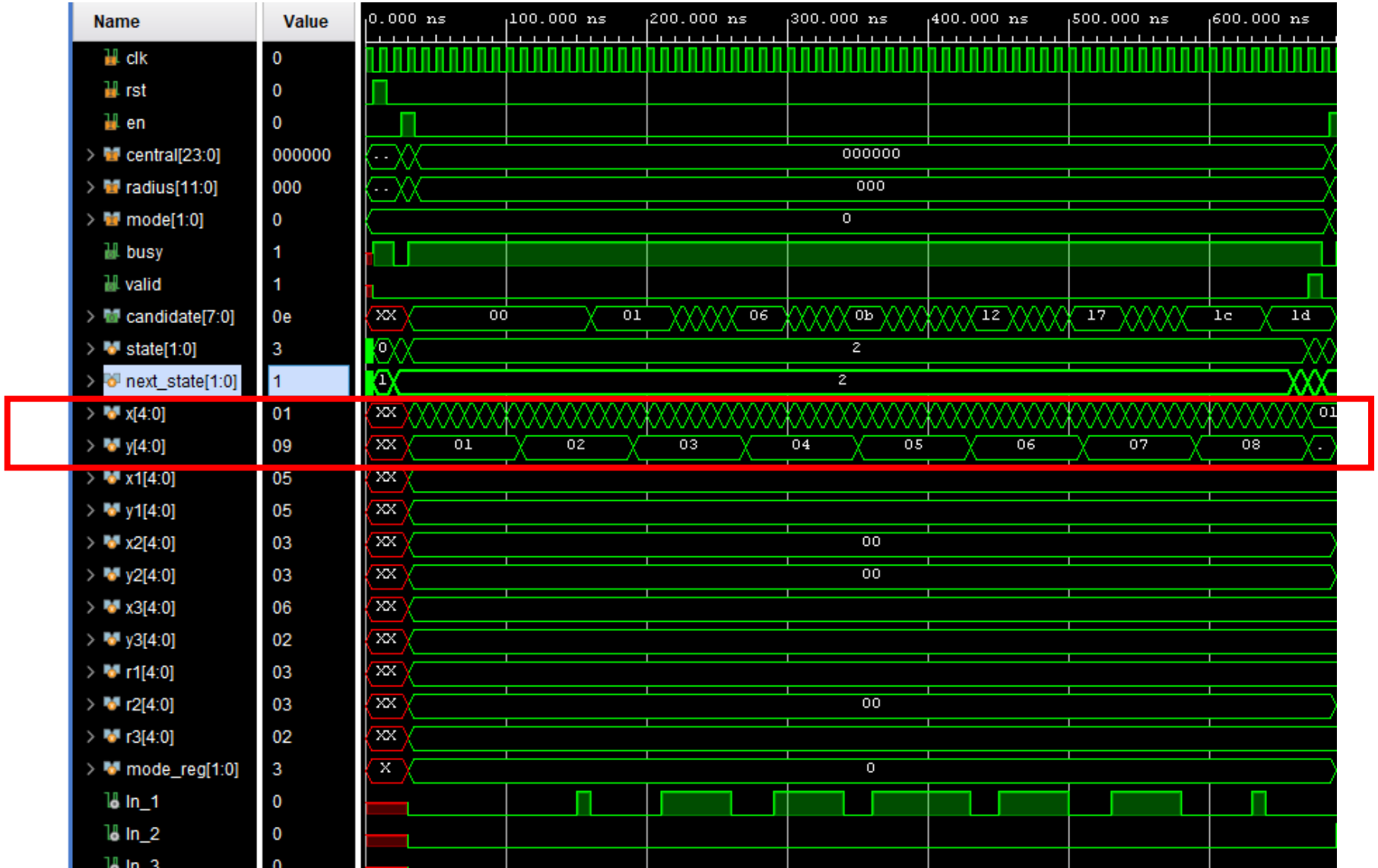


Reset訊號過後 從IDLE state轉換到 INPUT

從INPUT state轉換到CALCULATE state下，並將central,radius的data存下



在CALCULATE state下(x,y)從(1,1)到(8,8)的計算過程



在CALCULATE state下(x,y) = (8,8)，下個cycle時轉換到OUTPUT state
並將valid pull high 將candidate輸出出去