LBP 程式說明(修改版)

楊承峰

1. 全部存取版

A. 狀態機

i. state_INPUT

將 travesal 全部位址並將所取得的 gray_data 存到暫存陣列 data_array[127:0][127:0]裡。

ii. state_CALCULATE

透過迴圈將 gray_data 與中心位址的 gray_data 比大小並計算 LBP 值後存入暫存陣列 lbp_value[15876:0]裡。

iii. state_OUTPUT

travesal 全部的 lbp_addr 並依序將 lbp_array 的資料存入對應的 lbp_data。

```
//FSM
always@(posedge clk or posedge reset) begin
    if(reset)
        current_state <= state_INPUT;
    else
        current_state <= next_state;
end

always @(*) begin
    case(current_state)
        state_INPUT: begin
            if(column_count==14'd127&row_count==14'd127)
                next_state=state_CALCULATE;
            else
                next_state=state_INPUT;
        end
        state_CALCULATE:
            next_state=state_OUTPUT;

        state_OUTPUT:
            next_state=state_OUTPUT;
        state_IDLE: next_state=state_IDLE;
    endcase
end
```

(row_count、column、count 為控制 gray_addr 的參數)

B. input

將 gray_data 存入 data_array 哩，colum、row_count 為計次用)。

```
always@(posedge clk or posedge reset) begin
        if(reset) begin
            row_count<=-14'd1;
            column_count<=-7'd1;
            for(i=0;i<=14'd127;i=i+1)begin
                for(j=0;j<=14'd127;j=j+1) begin
                    data_array[i][j]<=0;
                end
            end
        end
        else if(gray_ready)begin
            data_array[row_count][column_count]<=gray_data;
            if(column_count==7'd127) begin
                column_count<=7'd0;
                row_count<=row_count+14'd1;
            end
            else
                column_count<=column_count+14'd1;
        end
        else begin
            row_count<=row_count;
            column_count<=column_count;
            for(i=0;i<=14'd127;i=i+1)begin
                for(j=0;j<=14'd127;j=j+1) begin
                    data_array[i][j]<=data_array[i][j];
                end
            end
        end
end
```

C. calculate

```
//calculate
always @(*) begin
    if(current_state==state_CALCULATE) begin
        lbp_count=14'd129;
        lbp_value[0]=14'd0;
        for(i=14'd1;i<=14'd15876;i=i+1)begin
            x={lbp_count[6],lbp_count[5],lbp_count[4],lbp_count[3],lbp_count[2],lbp_count[1],lbp_count[0]};
            y={lbp_count[13],lbp_count[12],lbp_count[11],lbp_count[10],lbp_count[9],lbp_count[8],lbp_count[7]};
            g0=data_array[y-7'd1][x-7'd1];
            g1=data_array[y-7'd1][x];
            g2=data_array[y-7'd1][x+7'd1];
            g3=data_array[y][x-7'd1];
            gc=data_array[y][x];
            g4=data_array[y][x+7'd1];
            g5=data_array[y+7'd1][x-7'd1];
            g6=data_array[y+7'd1][x];
            g7=data_array[y+7'd1][x+7'd1];

            if(g0<gc) lbp_value[i][0] = 1'b0;
            else lbp_value[i][0] =1'b1;

            if(g1<gc) lbp_value[i][1] = 1'b0;
            else lbp_value[i][1] =1'b1;

            if(g2<gc) lbp_value[i][2] = 1'b0;
            else lbp_value[i][2] =1'b1;

              if(g3<gc) lbp_value[i][3] = 1'b0;
              else lbp_value[i][3] =1'b1;

              if(g4<gc) lbp_value[i][4] = 1'b0;
              else lbp_value[i][4] =1'b1;

              if(g5<gc) lbp_value[i][5] = 1'b0;
              else lbp_value[i][5] =1'b1;

              if(g6<gc) lbp_value[i][6] = 1'b0;
              else lbp_value[i][6] =1'b1;

              if(g7<gc) lbp_value[i][7] = 1'b0;
              else lbp_value[i][7] =1'b1;

              if(lbp_count[6]&lbp_count[5]&lbp_count[4]&lbp_count[3]&lbp_count[2]&lbp_count[1])
                  lbp_count=lbp_count+3;
              else
                  lbp_count=lbp_count+1;
          end
    end
end
```

x、y 控制當前 lbp_addr 之 x、y 座標，lbp_count 則為所屬位址，i 為控制個數，將計算之 LBP 值存入 lbp_value。

D. outpout

主要透過 temp 來設定 lbp_value 傳給 lbp_data 的順序。

```
always@(*) begin
    if(lbp_valid) begin
        if(lbp_addr[6] & lbp_addr[5] & lbp_addr[4] & lbp_addr[3] & lbp_addr[2] & lbp_addr[1]&(lbp_addr!=14'd127))
            next_lbp_addr = lbp_addr + 14'd3;
        else if(lbp_addr>14'd16254) lbp_addr=14'd16254;
        else next_lbp_addr = lbp_addr + 14'd1;
    end
    else next_lbp_addr = lbp_addr;
end

always@(posedge clk or posedge reset) begin
    if(reset) temp<=14'd0;
    else if (lbp_valid&temp<=14'd15875) begin
        temp<=temp+14'd1;
    end
end

always@(*) begin
    next_lbp_data=lbp_value[temp];
end
```

E. Simulation Result

```
--------------------------------------------------------------------------------------------------------------------
                                                Congratulations!
                                            You have passed all patterns!
--------------------------------------------------------------------------------------------------------------------
$finish called at time : 322700 ns : File "D:/homework/project_1/project_1.srcs/sources_1/imports/LBP/testfixture.v" Line 198
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testfixture_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 750us
launch_simulation: Time (s): cpu = 00:00:05 ; elapsed = 00:00:07 . Memory (MB): peak = 1613.414 ; gain = 0.000
```

2. 三排存取版

A. 狀態機

i. state_INPUT

初次先前 3 排的 gray_addr 先 travesal，第二回則存一排，並將所得到的 gray_data 存入暫存陣列 data_array[2:0][127:0]中。

ii. state_CALCULATE

透過迴圈將 gray_data 與中心位址的 gray_data 做比大小並計算出 lbp 值在存入 lbp_value[127:0]中。

iii. state_OUTPUT

travesal 相應的一排 lbp_addr，並依序存入 lbp_data。

iv. state_SWITCH

將 data_array 中第 2、3 排交換成 1、2 排，再回 state_INPUT。

B. input

若為第 1 週期則存入首 3 排，第 2 週期後則存一排放入第 3 排暫存。

```
always@(posedge clk or posedge reset) begin
        if(reset) begin
            column_count<=-14'd1;
            row_count<=14'd0;
            for(i=0;i<=14'd2;i=i+1)begin
                for(j=0;j<=14'd127;j=j+1) begin
                    data_array[i][j]<=0;
                end
            end
        end
        else if(gray_ready& current_state==state_INPUT)begin
            if(row_count>=2) data_array[2][column_count]<=gray_data;
            else
             data_array[row_count][column_count]<=gray_data;

            if(column_count==14'd127) begin
                column_count<=14'd0;
                row_count<=row_count+14'd1;
            end
            else column_count<=column_count+14'd1;
        end
        else begin
            row_count<=row_count;
            column_count<=column_count;
            for(i=0;i<=14'd2;i=i+1)begin
                for(j=0;j<=14'd127;j=j+1) begin
                    data_array[i][j]<=data_array[i][j];
                end
            end
```

C. calculate

i 為控制個數及位址，將計算之 LBP 值存入 lbp_value。

```
//calculate
always @(*) begin
    if(current_state==state_CALCULATE) begin
            for(i=14'd1;i<=14'd126;i=i+1)begin
                g0=data_array[0][i-7'd1];
                g1=data_array[0][i];
                g2=data_array[0][i+7'd1];
                g3=data_array[1][i-7'd1];
                gc=data_array[1][i];
                g4=data_array[1][i+7'd1];
                g5=data_array[2][i-7'd1];
                g6=data_array[2][i];
                g7=data_array[2][i+7'd1];
                if(g0<gc) lbp_value[i][0] = 1'b0;
                else lbp_value[i][0] =1'b1;
                if(g1<gc) lbp_value[i][1] = 1'b0;
                else lbp_value[i][1] =1'b1;
                if(g2<gc) lbp_value[i][2] = 1'b0;
                else lbp_value[i][2] =1'b1;
                if(g3<gc) lbp_value[i][3] = 1'b0;
                else lbp_value[i][3] =1'b1;
                if(g4<gc) lbp_value[i][4] = 1'b0;
                else lbp_value[i][4] =1'b1;
                if(g5<gc) lbp_value[i][5] = 1'b0;
                else lbp_value[i][5] =1'b1;
                if(g6<gc) lbp_value[i][6] = 1'b0;
                else lbp_value[i][6] =1'b1;
                if(g7<gc) lbp_value[i][7] = 1'b0;
                else lbp_value[i][7] =1'b1;

            lbp_value[0]=8'd0;
            lbp_value[127]=lbp_value[126];
    end
end
```

D. output

temp 來控制 lbp_value 存給 lbp_data 的順序。

```
always@(*) begin
    if(lbp_valid& current_state==state_OUTPUT) begin
        if(lbp_addr[6] & lbp_addr[5] & lbp_addr[4] & lbp_addr[3] & lbp_addr[2] & lbp_addr[1]&~lbp_addr[0]&(lbp_addr!=14'd127)&(lbp_addr!=14'd16254))
            next_lbp_addr = lbp_addr+14'd3;
         else if(lbp_addr>=14'd16254) next_lbp_addr=14'd16254;
        else next_lbp_addr = lbp_addr + 14'd1;
    end
    else next_lbp_addr = lbp_addr;
end

always@(posedge clk or posedge reset) begin
    if(reset)begin
        temp<=14'd0;
    end
    else if (lbp_valid&temp<=14'd15875&current_state==state_OUTPUT) begin
        temp<=temp+14'd1;
    end
    else temp<=temp;
end

always@(*) begin
    next_lbp_data=lbp_value[temp];
end
```

E. switch

第二、三行之 data_array 往前移，以便讓之後資料存入第三排中。

```
always@(*) begin//switch
    if(current_state==state_SWITCH) begin
        for(i=0;i<=127;i=i+1) begin
            temp_array[i]=data_array[1][i];
            data_array[1][i]=data_array[2][i];
            data_array[0][i]=temp_array[i];
        end
    end
    else begin
        for(i=0;i<=127;i=i+1) begin
            data_array[0][i]=data_array[0][i];
            data_array[1][i]=data_array[1][i];
            data_array[2][i]=data_array[2][i];
        end
    end
end
```

F. Simulation Result

```
----------------------------------------------------------------------------------------------------------------------
                                                Congratulations!
                                            You have passed all patterns!
----------------------------------------------------------------------------------------------------------------------
$finish called at time : 328950 ns : File "D:/homework/project_1/project_1.srcs/sources_1/imports/LBP/testfixture.v" Line 198
xsim: Time (s): cpu = 00:00:05 ; elapsed = 00:00:06 . Memory (MB): peak = 1605.168 ; gain = 0.000
INFO: [USF-XSim-96] XSim completed. Design snapshot 'testfixture_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 750us
launch_simulation: Time (s): cpu = 00:00:05 ; elapsed = 00:00:09 . Memory (MB): peak = 1605.168 ; gain = 0.000
```