# BÁO CÁO ĐỒ ÁN CE213

# THIẾT KẾ ỨNG DỤNG XỬ LÝ ẢNH TRÊN FPGA BẰNG VERILOG

**Sinh viên thực hiện: 21520599 – Hoàng Phan Thành Bách**

## I. Cơ sở lý thuyết.

- Thiết kế bộ ứng dụng xử lý ảnh trên FPGA bằng ngôn ngữ Verilog là một đề tài phục vụ cho đồ án môn học tương đối phổ biến hiện nay. Trong đề tài, em đã tập trung vào 3 tính năng chính khi xử lý ảnh là việc tăng, giảm độ sáng, tương phản ảnh và tăng sáng/tối tối đa ảnh.
- Về phần xử lý ảnh, vì Verilog không thể đọc được ảnh input đầu vào (dưới dạng bitmap file). Vì thế, cần phải thiết kế một chương trình chuyển đổi với mục đích chuyển định dạng file bitmap thành các giá trị hexa và lưu dưới dạng file hex. Sau đó dùng lệnh **$readmemh** để đọc ảnh đầu vào.
- Sau khi chuyển sang file hex, ta chia ảnh đã được chuyển thành 3 kênh màu R-G-B, từ đó ta có 3 mảng lưu trữ các giá trị màu trên.
- Để thay đổi giá trị độ sáng ảnh, ta sẽ thay đổi giá trị các pixel ảnh (từ 0 -> 255).

## II. Nội dung thực hiện

### II.1. Thiết kế Converter

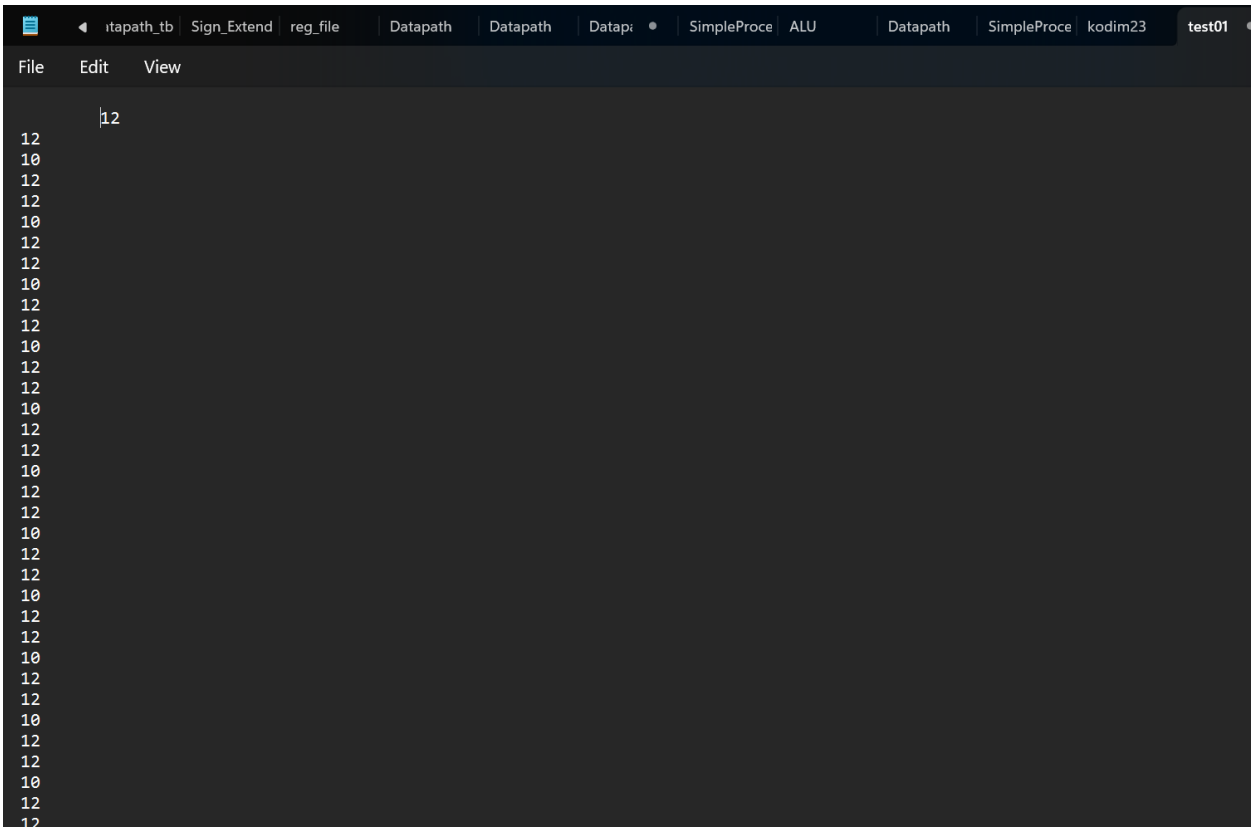
- Vì Verilog không thể đọc được định dạng ảnh bitmap file, nên ta cần thiết kế một chương trình chuyển đổi bitmap về dạng hexa. Em đã sử dụng chương trình chuyển đổi trên MATLAB để đưa ảnh đầu vào về dạng hexa.
- Chương trình:

convert.m

C: > altera > Image_Processor > convert.m

```
1   b=imread('kodim24.bmp'); % 24-bit BMP image RGB888
2
3   k=1;
4   for i=512:-1:1 % image is written from the last row to the first row
5   for j=1:768
6   a(k)=b(i,j,1); % red
7   a(k+1)=b(i,j,2); % green
8   a(k+2)=b(i,j,3); % blue
9   k=k+3;
10  end
11  end
12  fid = fopen('kodim24.hex', 'wt');
13  fprintf(fid, '%x\n', a);
14  disp('Text file write done');disp(' ');
15  fclose(fid);
```

- Input hình ảnh đầu vào sau khi qua chuyển đổi sẽ có kết quả như sau (file hexa):




## II.2. Thiết kế khối Image_Read

- Image_Read có vai trò đọc file hexa đầu vào, sau đó lưu trữ các giá trị màu R – G – B trong các biến tạm thời.

```
initial begin
    $readmemh(INFILE,total_memory,0,sizeOfLengthReal-1); // read file from INFILE
end
// use 3 intermediate signals RGB to save image data
always@(start) begin
    if(start == 1'b1) begin
        for(i=0; i<WIDTH*HEIGHT*3 ; i=i+1) begin
            temp_BMP[i] = total_memory[i+0][7:0];
        end

        for(i=0; i<HEIGHT; i=i+1) begin
            for(j=0; j<WIDTH; j=j+1) begin
                org_R[WIDTH*i+j] = temp_BMP[WIDTH*3*(HEIGHT-i-1)+3*j+0]; // save Red component
                org_G[WIDTH*i+j] = temp_BMP[WIDTH*3*(HEIGHT-i-1)+3*j+1];// save Green component
                org_B[WIDTH*i+j] = temp_BMP[WIDTH*3*(HEIGHT-i-1)+3*j+2];// save Blue component
            end
        end
    end
end
```

- Ngoài ra, trong khối Image_Read cũng có chức năng thiết kế khối FSM, chuyển đổi trạng thái. Trong đó có 4 khối trạng thái chính là IDLE, VSYNC, HSYNC và DATA.

```
// IDLE . VSYNC . HSYNC . DATA
always @(*) begin
    case(cstate)
        ST_IDLE: begin
            if(start)
                nstate = ST_VSYNC;
            else
                nstate = ST_IDLE;
        end
        ST_VSYNC: begin
            if(ctrl_vsync_cnt == START_UP_DELAY)
                nstate = ST_HSYNC;
            else
                nstate = ST_VSYNC;
        end
        ST_HSYNC: begin
            if(ctrl_hsync_cnt == HSYNC_DELAY)
                nstate = ST_DATA;
            else
                nstate = ST_HSYNC;
        end
        ST_DATA: begin
            if(ctrl_done)
                nstate = ST_IDLE;
            else begin
                if(col == WIDTH - 2)
                    nstate = ST_HSYNC;
                else
                    nstate = ST_DATA;
                end
```

- Trong đó, VSYNC không được sử dụng đến vì chỉ dùng trong trường hợp quét nhiều ảnh hoặc video.
- IDLE là trạng thái chờ, HSYN là trạng thái điều khiển cho chương trình quét các giá trị ảnh trên 1 hàng nhất định. DATA là giai đoạn xử lý dữ liệu,
- Cuối cùng, khối Image_Read cũng bao gồm việc thực hiện 3 chức năng tăng, giám độ sáng, tương phản độ sáng và tăng sáng/ tối tối đa của hình ảnh.
- Tính năng tăng giảm độ sáng:

```
// R0
tempR0 = org_R[WIDTH * row + col    ] + VALUE;
if (tempR0 > 255)
   DATA_R0 = 255;
else
   DATA_R0 = org_R[WIDTH * row + col    ] + VALUE;
// R1
tempR1 = org_R[WIDTH * row + col+1    ] + VALUE;
if (tempR1 > 255)
   DATA_R1 = 255;
else
   DATA_R1 = org_R[WIDTH * row + col+1    ] + VALUE;
// G0
tempG0 = org_G[WIDTH * row + col    ] + VALUE;
if (tempG0 > 255)
   DATA_G0 = 255;
else
   DATA_G0 = org_G[WIDTH * row + col    ] + VALUE;
```

- Bước tăng độ sáng, tempR0, tempG0, tempB0 sẽ có nhiệm vụ lưu trữ giá trị bit màu của ảnh sau khi tăng thêm với một giá trị VALUE nhằm tăng độ sáng lên. Nhưng nếu biến temp này có giá trị vượt quá 255, nó sẽ được đặt về lại 255, tức độ sáng tối đa.

```
// R0
tempR0 = org_R[WIDTH * row + col    ] - VALUE;
if (tempR0 < 0)
   DATA_R0 = 0;
else
   DATA_R0 = org_R[WIDTH * row + col    ] - VALUE;
// R1
tempR1 = org_R[WIDTH * row + col+1    ] - VALUE;
if (tempR1 < 0)
   DATA_R1 = 0;
else
   DATA_R1 = org_R[WIDTH * row + col+1    ] - VALUE;
// G0
tempG0 = org_G[WIDTH * row + col    ] - VALUE;
```

- Tương tự với bước giảm độ sáng, các biến temp sẽ lưu trữ giá trị bit màu của ảnh sau khi trừ đi với giá trị VALUE. Nếu giá trị này nhỏ hơn 0, nó sẽ được đặt về 0, tức độ sáng tối thiểu.
- Ở tính năng tương phản, ta sẽ tính giá trị bit màu trung bình của 3 màu R – G – B trong cùng điểm ảnh. Sau đó đem 255 trừ đi cho giá trị mới tìm được để tìm ra độ sáng ngược lại của bức ảnh gốc.

```
`ifdef INVERT_OPERATION
   value2 = (org_B[WIDTH * row + col  ] + org_R[WIDTH * row + col  ] +org_G[WIDTH * row + col  ])/3;
   DATA_R0=255-value2;
   DATA_G0=255-value2;
   DATA_B0=255-value2;
   value4 = (org_B[WIDTH * row + col+1  ] + org_R[WIDTH * row + col+1  ] +org_G[WIDTH * row + col+1  ])/3;
   DATA_R1=255-value4;
   DATA_G1=255-value4;
   DATA_B1=255-value4;
`endif
```

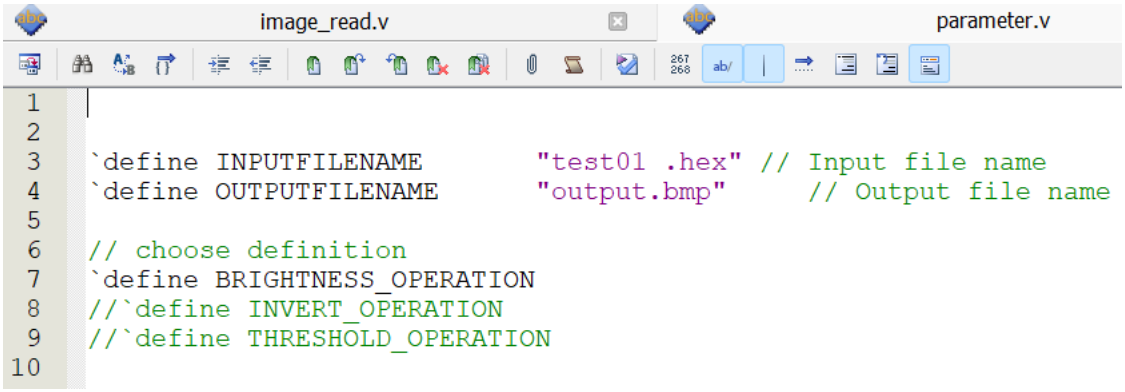
- Ở tính năng tăng/ giảm sáng tối đa, ta dựa vào giá trị THRESHOLD đã đặt từ trước. Nếu giá trị bit màu ảnh cao hơn, ta sẽ tăng sáng đến giá trị tối đa là 255, ngược lại sẽ giảm về 0.

```
`ifdef THRESHOLD_OPERATION

value = (org_R[WIDTH * row + col   ]+org_G[WIDTH * row + col   ]+org_B[WIDTH * row + col   ])/3;
if(value > THRESHOLD) begin
   DATA_R0=255;
   DATA_G0=255;
   DATA_B0=255;
end
else begin
   DATA_R0=0;
   DATA_G0=0;
   DATA_B0=0;
end
```

### II.3. Khối Parameter.

- Vì thiết kế chỉ có thể thực hiện 1 trong 3 chức năng trong một lần chạy, vì vậy, ta thiết kế khối Parameter cũng như sử dụng ifdef để xét tính năng được chọn.




### II.4 Khối Image_Write.

- Sau khi thiết kế các khối kể trên, ta cần thiết kế một khối có chức năng viết lại ảnh đã được chuyển đổi trở về dạng bitmap và đưa ra output.
- Đầu tiên, ta cần phải viết header cho ảnh bitmap chuẩn bị chuyển đổi. Để viết header, ta cần tra cứu format của các bit thông qua trang http://www.fastgraph.com/help/bmp_header_format.html.

```
// Check the website to see the value of this head
initial begin
    BMP_header[ 0] = 66;BMP_header[28] =24;
    BMP_header[ 1] = 77;BMP_header[29] = 0;
    BMP_header[ 2] = 54;BMP_header[30] = 0;
    BMP_header[ 3] =  0;BMP_header[31] = 0;
    BMP_header[ 4] = 18;BMP_header[32] = 0;
    BMP_header[ 5] =  0;BMP_header[33] = 0;
    BMP_header[ 6] =  0;BMP_header[34] = 0;
    BMP_header[ 7] =  0;BMP_header[35] = 0;
    BMP_header[ 8] =  0;BMP_header[36] = 0;
    BMP_header[ 9] =  0;BMP_header[37] = 0;
    BMP_header[10] = 54;BMP_header[38] = 0;
    BMP_header[11] =  0;BMP_header[39] = 0;
    BMP_header[12] =  0;BMP_header[40] = 0;
    BMP_header[13] =  0;BMP_header[41] = 0;
    BMP_header[14] = 40;BMP_header[42] = 0;
    BMP_header[15] =  0;BMP_header[43] = 0;
    BMP_header[16] =  0;BMP_header[44] = 0;
    BMP_header[17] =  0;BMP_header[45] = 0;
    BMP_header[18] =  0;BMP_header[46] = 0;
    BMP_header[19] =  3;BMP_header[47] = 0;
    BMP_header[20] =  0;BMP_header[48] = 0;
    BMP_header[21] =  0;BMP_header[49] = 0;
    BMP_header[22] =  0;BMP_header[50] = 0;
    BMP_header[23] =  2;BMP_header[51] = 0;
    BMP_header[24] =  0;BMP_header[52] = 0;
    BMP_header[25] =  0;BMP_header[53] = 0;
    BMP_header[26] =  1;
    BMP_header[27] =  0;
end
```

- Tiếp theo, ta lưu trữ những giá trị mới được chuyển đổi trong 1 mảng duy nhất (out_BMP).

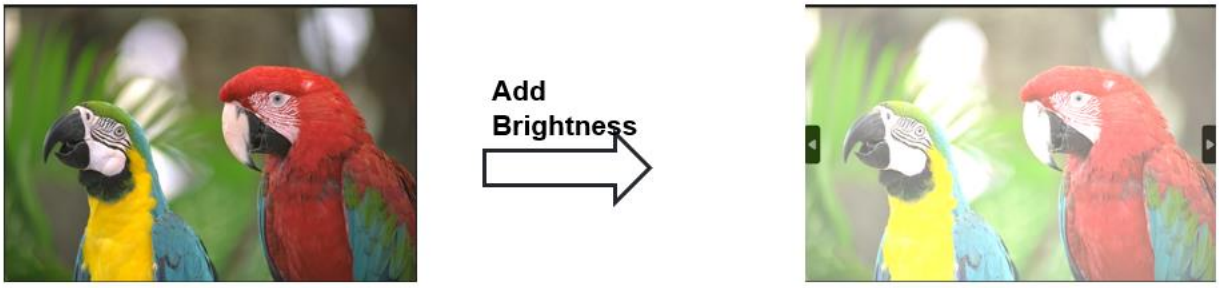
```
// Writing RGB888 even and odd data to the temp memory
always@(posedge HCLK, negedge HRESETn) begin
    if(!HRESETn) begin
        for(k=0;k<WIDTH*HEIGHT*3;k=k+1) begin
            out_BMP[k] <= 0;
        end
    end else begin
        if(hsync) begin
            out_BMP[WIDTH*3*(HEIGHT-1-1)+6*m+2] <= DATA_WRITE_R0;
            out_BMP[WIDTH*3*(HEIGHT-1-1)+6*m+1] <= DATA_WRITE_G0;
            out_BMP[WIDTH*3*(HEIGHT-1-1)+6*m  ] <= DATA_WRITE_B0;
            out_BMP[WIDTH*3*(HEIGHT-1-1)+6*m+5] <= DATA_WRITE_R1;
            out_BMP[WIDTH*3*(HEIGHT-1-1)+6*m+4] <= DATA_WRITE_G1;
            out_BMP[WIDTH*3*(HEIGHT-1-1)+6*m+3] <= DATA_WRITE_B1;
        end
    end
end
```

- Sau cùng, ta viết lại file bitmap dựa trên header và giá trị màu đã lưu.

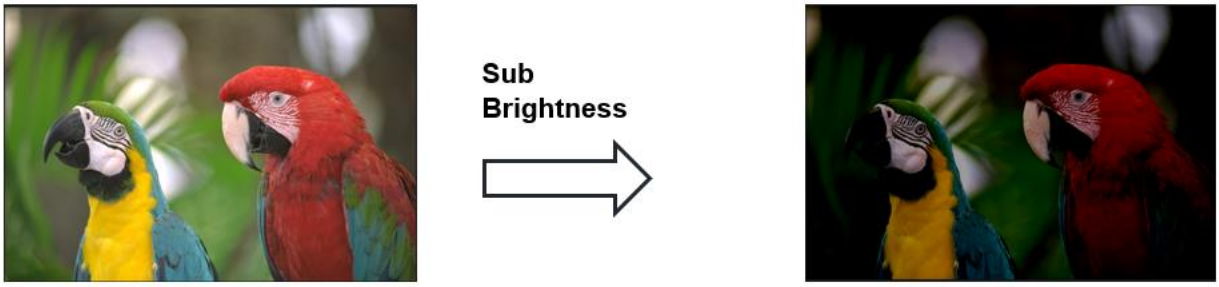
```
120 initial begin
121     fd = $fopen(INFILE, "wb+");
122 end
123 always@(Write_Done) begin // once the processing was done, bmp image will be created
124     if(Write_Done == 1'b1) begin
125         for(i=0; i<BMP_HEADER_NUM; i=i+1) begin
126             $fwrite(fd, "%c", BMP_header[i][7:0]); // write the header
127         end
128
129         for(i=0; i<WIDTH*HEIGHT*3; i=i+6) begin
130       // write R0B0G0 and R1B1G1 (6 bytes) in a loop
131             $fwrite(fd, "%c", out_BMP[i  ][7:0]);
132             $fwrite(fd, "%c", out_BMP[i+1][7:0]);
133             $fwrite(fd, "%c", out_BMP[i+2][7:0]);
134             $fwrite(fd, "%c", out_BMP[i+3][7:0]);
135             $fwrite(fd, "%c", out_BMP[i+4][7:0]);
136             $fwrite(fd, "%c", out_BMP[i+5][7:0]);
137         end
138     end
139 end
```

## III. Kết quả

- Bộ ứng dụng đã thực hiện thành công các chức năng được đề ra
    - Tăng/ giảm độ sáng:



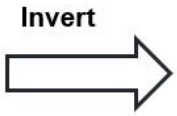

Ảnh gốc



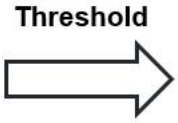

Ảnh gốc

    - Tương phản:

Ảnh gốc




- Tăng/ giảm sáng tối đa:




- Nhận xét:

  + Đã ứng dụng được các kiến thức đã học về Verilog, kết hợp một số công cụ thiết kế mạnh mẽ như Quartus, ModelSim để thực hiện đề tài trên

  + Xử lý ảnh là một hướng đi phổ biến và qua đề tài này em đã được cơ hội tìm hiểu thêm những kiến thức liên quan đến lĩnh vực này.

  + Mô phỏng thành công, nhưng đề tài vẫn cần nạp kit trong tương lai để kiểm tra quá trình chạy thực tế.