IMAGE COMPRESSION FOR BMP

Grup PA - 14 PSD

PA-14

Members

01 Izzan Nawa Syarif
2306266956

02 Nabiel Harits Utomo
2306267044

03 Neyla Shakira
2306250655

04 Stevie Nathania Siregar
2306242382

01

Background

Background

Pengembangan teknologi pada saat ini sangatlah pesat. Bagi para mahasiswa, pastinya kenal dengan penggunaan PDF untuk pembelajaran saat kuliah. Penggunaan PDF ini juga seringkali dipakai untuk membaca satu buku paket dari satu PDF saja. Akan tetapi, banyak sekali kekurangan PDF apabila membuat suatu buku yang tebal dijadikan satu PDF. Hal tersebut dapat membuat file PDF terlalu besar dan perlu dikurangi resolusinya.

Pada projek kali ini, kami tentunya sudah belajar tentang penggunaan VHDL. Projek kami membuat suatu image compression yang menggunakan file format BMP. Kompresi ini mempunyai tujuan agar resolusi gambar pixel yang dimasukkan ke dalam file “input.bmp” dapat berkurang dan ditujukan ke dalam file “out.bmp”.

Why we pick this project?

- Referensi yang mudah dicapai
- Pembagian pixel yang memungkinkan
- Mudah untuk diimplementasi pada VHDL
- Sudah ada contoh penggunaan colour scaling

Our goal

- Membuat kompresi file BMP berhasil
- Membuat kode vhdl ini dapat read suatu file external dan write di tempat file yang berbeda
- Menurunkan resolusi image BMP sesuai yang diinginkan
- Mengaplikasikan modul PSD yang sudah ditentukan ke dalam program

What we use

- VS Code
- ModelSim
- Github

02

Program

ReadBMP

Entitas ReadBMP ini bertugas untuk membaca “image.bmp” ini dari file yang sudah dimasukkan ke dalam folder yang sama pada project ModelSIM.

Cara kerja ReadBMP ini mengecek lebar gambar dan tingginya terlebih dahulu (referensi dari contoh colouring pad).

```
begin
    if rising_edge(clk) and start = '1' then
        -- Baca Header BMP
        for i in 0 to 53 loop
            read(bmp_file, header(i));
        end loop;

        header_out <= header;

        -- Validasi Header
        assert header(0) = 'B' and header(1) = 'M'
            report "First two bytes are not 'BM'. This is not a BMP file"
            severity failure;

        -- Mengambil Panjang dan lebar pixel
        image_width := character'pos(header(18)) +
                       character'pos(header(19)) * 2**8 +
                       character'pos(header(20)) * 2**16 +
                       character'pos(header(21)) * 2**24;

        image_height := character'pos(header(22)) +
                        character'pos(header(23)) * 2**8 +
                        character'pos(header(24)) * 2**16 +
                        character'pos(header(25)) * 2**24;

        img_w_out <= image_width;
        img_h_out <= image_height;

        -- Debug output panjang x lebar
        -- report "Image dimensions - Width: " & integer'ima
                --", Height: " & integer'image(image_height)

        -- Kalkulasi Padding
        padding := (4 - (image_width * 3) mod 4) mod 4;

        -- Membaca pixel ke array
        for row_i in 0 to image_height - 1 loop
            for col_i in 0 to image_width - 1 loop
                -- Read and store blue pixel
                read(bmp_file, char);
                blue := std_logic_vector(to_unsigned(character'pos(char), 8));
                image_data(row_i)(col_i).blue := blue;

                -- Read and store green pixel
                read(bmp_file, char);
                green := std_logic_vector(to_unsigned(character'pos(char), 8));
                image_data(row_i)(col_i).green := green;

                -- Read and store red pixel
                read(bmp_file, char);
                red := std_logic_vector(to_unsigned(character'pos(char), 8));
                image_data(row_i)(col_i).red := red;

                -- Debug output setiap pixel
                --report "Blue: " & to_string(blue) &
                        --", Green: " & to_string(green) &
                        --", Red: " & to_string(red);
            end loop;

            -- Discard padding bytes at the end of each row
            for pad_i in 1 to padding loop
                read(bmp_file, char);
            end loop;
        end loop;

        -- Output image data
        image_out <= image_data;

        -- Debug signal
        see <= image_data(0)(0).red;
```

WriteBMP

Entitas WriteBMP ini bertugas untuk otomatis membuat file “out.bmp” ini dari file yang sudah dikompresi dan dimasukkan ke file “out.bmp” ini.

Cara kerja WriteBMP ini hanya menginput file yang sudah jadi dan menggunakan colouring pad pada red, blue, dan green.

```
begin
    if rising_edge(clk) then
        if start = '1' then

            image_width := img_w_in;
            image_height:= img_h_in;
            image_data  := image_in;
            header      := header_in;
            padding := (4 - image_width*3 mod 4) mod 4;


            -- Tulis Header hasil compress ke file output
            for i in 0 to 53 loop
                write(out_file, header(i));
            end loop;

            for row_i in 0 to image_height - 1 loop
                for col_i in 0 to image_width - 1 loop
                -- blue pixel
                write(out_file,
                    character'val(to_integer(unsigned(image_data(row_i)(col_i).blue))));

                -- green pixel
                write(out_file,
                    character'val(to_integer(unsigned(image_data(row_i)(col_i).green))));

                -- red pixel
                write(out_file,
                    character'val(to_integer(unsigned(image_data(row_i)(col_i).red))));

                end loop;


                -- Write padding
                for i in 1 to padding loop
                    write(out_file, character'val(0));
                end loop;

            end loop;
```

ImageCompression

Entitas ImageCompression ini bertugas untuk melakukan kompresi file dari file “input.bmp”.

Cara kerja ImageCompression ini awalnya meng-adjust dahulu apakah pixel ini genap atau ganjil. Setelah itu, akan meng-update header kompres dan melakukan update ukuran headernya.

Lalu ia melakukan penjumlahan pixels antarblok yang dilakukannya dengan loop pada red, gree, dan blue.

```
begin
    if rising_edge(clk) and start = '1' then
        done <= '0';

        -- Kalkulasi Panjang X lebar baru
        img_w_data := img_width_in / block_size;
        img_h_data := img_height_in / block_size;

        -- Adjustment Kalo ganjil (not working properly)
        if img_width_in mod block_size /= 0 then
            img_w_data := img_w_data + 1;
        end if;
        if img_height_in mod block_size /= 0 then
            img_h_data := img_h_data + 1;
        end if;

        img_width_out <= img_w_data;
        img_height_out <= img_h_data;

        -- Update header setelah compress
        header := header_in;
        header(18) := character'val(img_w_data mod 256); -- Width LSB
        header(19) := character'val((img_w_data / 256) mod 256);
        header(22) := character'val(img_h_data mod 256); -- Height LSB
        header(23) := character'val((img_h_data / 256) mod 256);

        -- Update ukuran gambar di header
        header(34) := character'val((img_w_data * img_h_data * 3) mod 256); -- Image size LSB
        header(35) := character'val(((img_w_data * img_h_data * 3) / 256) mod 256);
        header(2)  := character'val(((img_w_data * img_h_data * 3 + 54) mod 256)); -- File size LSB
        header(3)  := character'val((((img_w_data * img_h_data * 3 + 54) / 256) mod 256));

        -- Signal keluar header
        header_out <= header;

        -- Block area calculation
        block_area := block_size * block_size;

        -- Mean filtering block_size x block_size pixel blocks
        for row_i in 0 to (img_h_data - 1) loop
            for col_i in 0 to (img_w_data - 1) loop
                row_in := row_i * block_size;
                col_in := col_i * block_size;

                -- Initialize sums
                r := 0;
                g := 0;
                b := 0;

                -- Sum pixels within the block
                for i in 0 to block_size - 1 loop
                    for j in 0 to block_size - 1 loop
                        if (row_in + i < img_height_in) and (col_in + j < img_width_in) then
                            r := r + to_integer(unsigned(image_in(row_in + i)(col_in + j).red));
                            g := g + to_integer(unsigned(image_in(row_in + i)(col_in + j).green));
                            b := b + to_integer(unsigned(image_in(row_in + i)(col_in + j).blue));
                        end if;
                    end loop;
                end loop;

                -- Kalkulasi mean
                r := r / block_area;
                g := g / block_area;
                b := b / block_area;

                -- Hasil kalkulasi di masukkan ke array
                image_data(row_i)(col_i).red   := std_logic_vector(to_unsigned(r, 8));
                image_data(row_i)(col_i).green := std_logic_vector(to_unsigned(g, 8));
                image_data(row_i)(col_i).blue  := std_logic_vector(to_unsigned(b, 8));
            end loop;
        end loop;

        -- output Gambar compress
        image_out <= image_data;
        done <= '1';
```

Array_pkg

Entitas Array_pkg ini berupa header array yang membuat suatu tipe data red, green, dan blue untuk dijadikan 8 bit.

Fungsi dari array_pkg ini berupa header dari semua entitas sebelumnya yang dapat dipakai untuk tipe data yang saling berhubungan.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package Array_pkg is
    type header_type is array (0 to 53) of character;

    type pixel_type is record
        red : std_logic_vector(7 downto 0);
        green : std_logic_vector(7 downto 0);
        blue : std_logic_vector(7 downto 0);
    end record;
    type row_type is array (0 to 1919) of pixel_type;
    type image_type is array (0 to 1079) of row_type;
end package;
```

TopLevel

TopLevel ini berfungsi untuk menggabungkan entitas yang sebelumnya, entity ini berfungsi untuk test run pada modelSIM dengan memberikan current state (Idle, Reading, Compress, Done).

TopLevel ini menggunakan clock, start (memulai read,compres, dll), serta status. Saat sudah dikompress, statusnya akan berubah menjadi 1 yang menandakan sudah di write pada file “out.bmp”

```
begin
    if rst = '1' then
        current_state <= idle;
        start_r     <= '0';
        start_comprs<= '0';
        start_wr    <= '0';
        status      <= '0';
    elsif rising_edge(clk) then
        case current_state is
            when idle =>
                start_r     <= '0';
                start_comprs<= '0';
                start_wr    <= '0';
                status      <= '0';
                start_comprs <= '0';
                if start = '1' then
                    start_r <= '1';
                    current_state <= reading;
                end if;

            when reading =>
                start_r <='0';
                if read_done = '1' then
                    current_state <= compress;
                end if;

            when compress =>
                start_comprs <= '1';
                case op_code is
                    when "0010" =>
                        compress_size <= 2;
                    when "0011" =>
                        compress_size <= 3;
                    when "0100" =>
                        compress_size <= 4;
                    when others =>
                        null;
                end case;
```

```
                if done_component = '1' then
                    start_comprs <= '0';
                    start_wr <='1';
                    current_state <= writing;
                end if;
                -- debug
                -- report "Current state: " & state_type'image(current_state);
                -- report "Write_done signal: " & std_logic'image(write_done);


            when writing =>
                if write_done = '1' then
                    start_wr <= '0';
                    current_state <= done;
                    -- debug
                    -- report "Current state: " & state_type'image(current_state);
                    -- report "write_done signal: " & std_logic'image(write_done);

                end if;

            when done =>
                -- debug
                -- report "Current state: " & state_type'image(current_state);
                -- report "write_done signal: " & std_logic'image(write_done);
                status <= '1';
                if start = '0' then
                    current_state <= idle;
                end if;

            when others =>
                current_state <= idle;
        end case;
    end if;
end process FSM;
```

TopLevel_tb

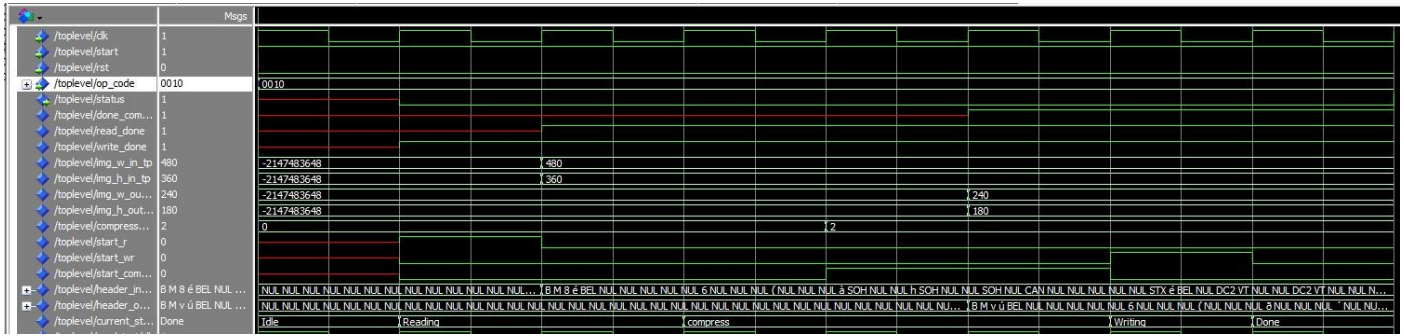
Dengan menggunakan entity dari TopLevel, kita dapat membuat testbenchnya dari TopLevel

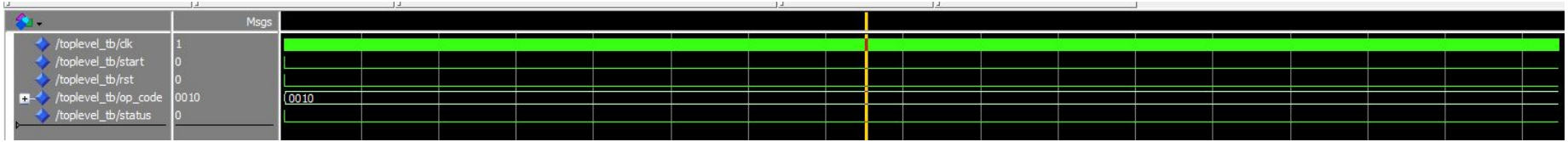
```
begin

    -- Instantiate the DUT
    DUT: TopLevel
        port map (
            clk     => clk,
            start   => start,
            rst     => rst,
            op_code => op_code,
            status  => status
        );

    -- Clock generation process
    clk_gen: process
    begin
        clk <= '0';
        wait for clk_period / 2;
        clk <= '1';
        wait for clk_period / 2;
    end process clk_gen;

    -- Stimulus process
    stim_proc: process
    begin
        -- Initial reset
        rst <= '1';
        wait for 20 ns;
        rst <= '0';

        -- Start signal activation
        start <= '1';
        op_code <= "0010"; -- Compression size = 2
        wait for 100 ns;
        start <= '0';

        -- Wait for the FSM to reach the done state (status = '1')
        wait until status = '1';

        -- Terminate the simulation after one FSM cycle
        report "Simulation completed successfully." severity note;
```

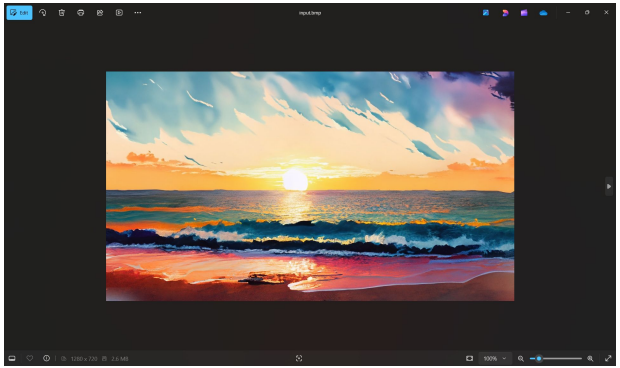
03

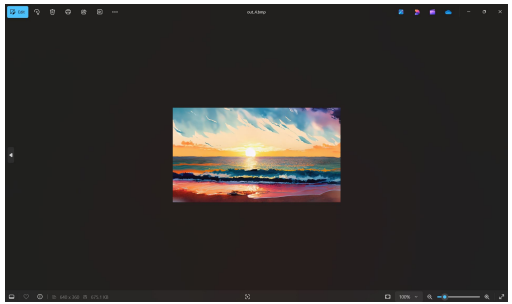
Result

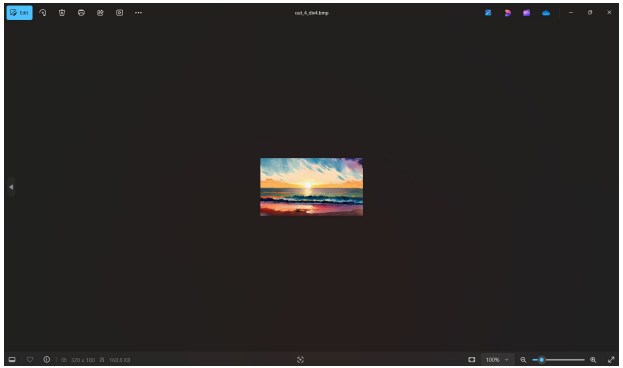
Result and Testing











Analysis

Hasil simulasi yang kami buat bisa dilihat bahwa sistem berjalan sesuai dengan yang diinginkan walaupun masih ada sedikit kekurangan seperti testbench yang tidak kelar runnya, quartus yang tidak bisa disintesis (dikarenakan array tidak ke-read), serta status yang tidak bisa berubah menjadi angka 1. Kesalahan ini menunjukkan adanya kode yang kurang stabil seperti tipe data yang belum tentu bisa disintesis sesuai dengan array pakage yang sudah dibuat.

Pada TopLevel masih banyak tipe data yang masih bertuliskan UUUU atau NULL, hal ini dikarenakan kode yang sudah dibuat akan mengganti hasil UUUU atau NULL pada saat berjalannya compression.

Pada tahap ReadBMP menggunakan colourscaler yang didapatkan dari contoh template penggunaan Proyek Akhir di emas PSD. Begitu juga dengan ImageCompressor.

Conclusion

Proyek sistem kompresi gambar yang dibuat untuk menjalankan alur kerja utama, yaitu membaca file BMP, mengompresi gambar, dan menyimpan hasil kompresi ke file BMP baru.

Setiap bagian, dari ReadBMP, ImageCompressor, hingga WriteBMP, berjalan dengan proses yang diatur oleh TopLevel entity. Hasil simulasi menunjukkan bahwa alur kontrol berjalan sesuai yang diharapkan, dengan sistem mampu membaca gambar BMP. Tidak hanya itu, program juga dapat membaca file external dan write pada file yang berbeda.

Proses ini sesuai dengan tujuan utama kompresi gambar, yaitu mengurangi resolusi gambar menggunakan teknik pemfilteran rata-rata tanpa menghilangkan data penting. Tidak hanya itu, kami juga menggunakan modul yang sesuai dengan apa yang kita tentukan.

References

[1] J. J. Jensen, “BMP file bitmap image read using TEXTIO,” *VHDLwhiz*, Nov. 13, 2019. https://vhdlwhiz.com/read-bmp-file/

[2] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, “Spatial Filters - Mean Filter,” homepages.inf.ed.ac.uk, 2003. https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm

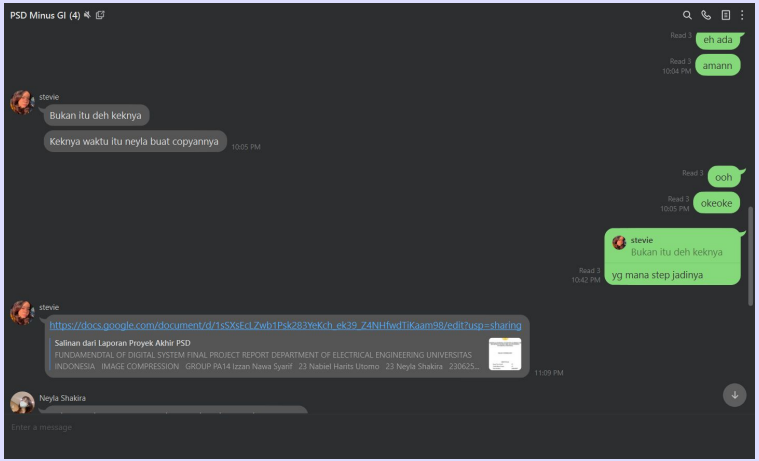
[3] Modul di Emas

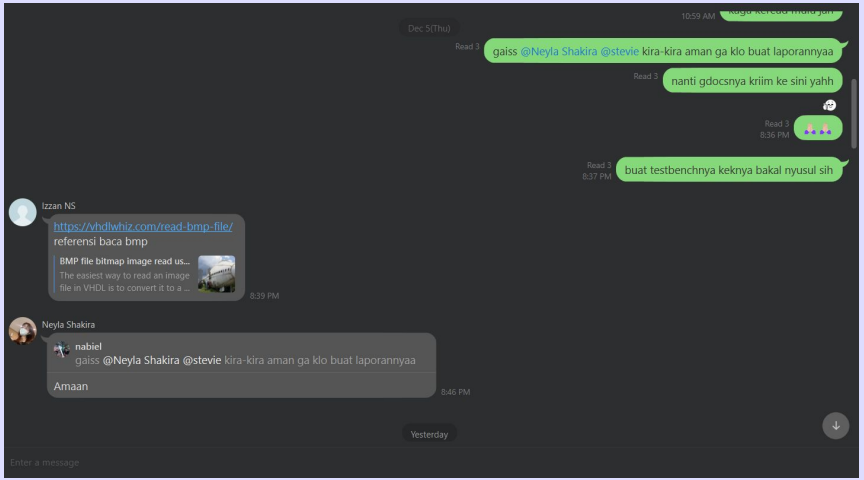
[4] Russell, “Arrays - VHDL Example. Learn to create 2D synthesizable arrays,” Nandland, Jun. 09, 2022. https://nandland.com/arrays/

Appendix A:

No quartus Syntesis

Appendix B: Documentation





THANK YOU

PA-14