FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

IMAGE TOOLS

GROUP BP08

Aisyah Arifatul Alya – 2206059383

Annisa Ardelia Setiawan – 2206059471

Giovan Christoffel Sihombing – 2206816084

Sihombing Giovano Geraldo – 2206059566

PREFACE

Dalam proyek ini, kami mengembangkan sebuah program menggunakan bahasa pemrograman VHDL (*VHSIC Hardware Description Language*) yang mampu melakukan berbagai manipulasi terhadap gambar dengan memanfaatkan konsep operasi biner pada level piksel. Program yang dibuat memiliki beberapa fitur utama, termasuk konversi warna ke skala abu-abu, biru, merah, dan hijau pada gambar yang dimasukkan.

Proses manipulasi gambar dimulai dengan konversi gambar ke format biner, di mana setiap pixel direpresentasikan dalam bentuk biner dan diolah secara terpisah. Langkah ini memungkinkan kita untuk melakukan operasi biner pada setiap pixel dengan lebih mudah. Setelah konversi, program memberikan kemampuan untuk memproses setiap pixel sesuai dengan pilihan operasi yang dipilih.

Pendekatan ini memungkinkan pengguna untuk memiliki kendali yang lebih besar terhadap estetika gambar yang akan dimanipulasi. Berbagai efek warna dan manipulasi gambar dapat diterapkan dengan mudah, memberikan fleksibilitas yang tinggi dalam menghasilkan output yang diinginkan.

Melalui laporan akhir ini, kami akan menjelaskan rinci langkah-langkah implementasi program, serta hasil yang berhasil dicapai. Diharapkan laporan ini dapat memberikan pemahaman yang baik tentang bagaimana program ini dirancang dan diimplementasikan, serta memberikan landasan bagi pengembangan lebih lanjut di bidang VHDL dan pengolahan gambar.

Depok, December 24, 2023

Group BP-08

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Pada era digitalisasi, pengolahan citra digital menjadi salah satu aspek yang cukup diperlukan dalam berbagai aplikasi, salah satunya yakni *image processing*. Gambar berwarna dapat digunakan untuk berbagai keperluan, seperti untuk keperluan estetika, untuk keperluan informasi, atau untuk keperluan penelitian. Namun, terkadang gambar berwarna dapat menjadi terlalu kompleks atau rumit untuk diproses atau ditampilkan. Dalam hal ini, gambar berwarna dapat diubah menjadi grayscale untuk mempermudah proses atau tampilannya.

Sehubungan dengan hal itu, maka kebutuhan akan aplikasi yang dapat melakukan manipulasi gambar secara efektif akan semakin meningkat. Oleh karena itu, kelompok BP-08 hadir dengan proyek "*Image Tools*" yang merupakan sebuah alat pengolahan citra menggunakan *Very High-Speed Integrated Circuit Hardware Description Language* (VHDL), yang dapat mengimplementasikan berbagai operasi seperti perubahan warna menjadi *grayscale* (hitam putih), *blue scale*, *red scale*, *green scale*, dan *scaling*.

Pengolahan Warna:

Melakukan perubahan warna pada gambar adalah salah satu aspek yang penting dalam pengolahan citra. Aplikasi "*Image Tools*" yang dirancang oleh kelompok BP-08 akan dapat mengimplementasikan:

- *Blue Scale*

  Fitur yang memungkinkan pengguna untuk mengubah gambar berwarna menjadi gambar dengan intensitas warna biru.

- *Red Scale*

  Fitur yang memungkinkan pengguna untuk mengubah gambar berwarna menjadi gambar dengan intensitas warna merah.

- *Green Scale*

  Fitur yang memungkinkan pengguna untuk mengubah gambar berwarna menjadi gambar dengan intensitas warna hijau.

- Color to Grayscale

  Fitur yang memungkinkan pengguna untuk mengubah gambar berwarna menjadi gambar dengan intensitas warna hitam putih.

Scaling (*Zoom In*/*Zoom Out*):

Image Scaling menjadi salah satu fokus utama dalam aplikasi yang dikembangkan oleh kelompok BP-08. Dengan menggunakan VHDL, kelompok BP-08 mengimplementasikan mekanisme *Image Scaling* yang efisien tanpa mengorbankan kualitas gambar. Dengan adanya fitur ini, pengguna dapat memperbesar atau memperkecil gambar sesuai dengan preferensi.

Dengan menggunakan VHDL sebagai bahasa utama dalam mengembangkan aplikasi "*Image Tools*", diharapkan dapat memiliki keunggulan tersendiri dibanding menggunakan bahasa lainnya. VHDL merupakan bahasa deskripsi *hardware* yang *powerful*. Sehingga diharapkan dalam penggunaan aplikasi ini akan memanfaatkan sumber daya *hardware* secara efisien dan memberikan kinerja yang cepat.

1.2 PROJECT DESCRIPTION

Proyek ini bertujuan untuk mengimplementasikan beberapa fungsi pengolahan gambar pada tingkat piksel dengan menggunakan bahasa pemrograman VHDL. Fungsi utama yang akan diimplementasikan melibatkan konversi gambar ke skala abu-abu, konversi gambar ke skala warna pilihan (merah, hijau, atau biru), dan perubahan ukuran gambar. Proses konversi gambar ke skala abu-abu akan dilakukan dengan mengubah setiap nilai piksel gambar menjadi tingkat keabuan yang sesuai.

Proses konversi ke warna pilihan dilakukan dengan mengubah setiap warna piksel pada gambar menjadi tingkat warna pilihan yang sesuai (merah, hijau, atau biru). Fungsi resize pada proyek ini bertujuan untuk memberikan kemampuan kepada pengguna untuk mengubah ukuran gambar sesuai dengan parameter yang diberikan, sehingga memfasilitasi fleksibilitas dalam tampilan gambar. Keseluruhan proyek ini akan diimplementasikan dalam bahasa VHDL untuk dijalankan pada perangkat keras FPGA, memungkinkan akselerasi tingkat tinggi untuk pengolahan gambar pada tingkat piksel.

1.3 OBJECTIVES

Objektif dari proyek ini adalah sebagai berikut:

1. Membangun implementasi perangkat keras menggunakan VHDL untuk pengolahan gambar pada tingkat piksel.
2. Mengubah setiap nilai piksel gambar ke tingkat keabuan yang sesuai, memungkinkan representasi gambar dalam skala abu-abu (grayscale).
3. Mengubah setiap nilai piksel gambar ke warna pilihan (merah, hijau, atau biru) dengan tingkat skala warna yang sesuai.
4. Mengimplementasikan fungsi resize untuk mengubah ukuran gambar sesuai dengan parameter yang diberikan, memfasilitasi fleksibilitas dalam tampilan gambar.

1.4 ROLES AND RESPONSIBILITIES

Tugas dan tanggungjawab yang dibagikan antara anggota kelompok adalah berikut:

Roles	Responsibilities	Person
Membuat kode dan membuat laporan	● Membuat blue scale, green scale, dan red scale. ● Membuat laporan dan ppt.	Aisyah Arifatul Alya
Membuat kode dan membuat laporan	● Membuat reverse grayscale dan membuat cpu. ● Membuat laporan dan ppt.	Annisa Ardelia Setiawan
Membuat kode dan memastikan kode dapat berjalan	● Membuat alu, dan clock pada cpu. ● Menggabungkan kode dan memastikan kode dapat berjalan sesuai dengan yang diinginkan.	Giovan Christoffel Sihombing
Membuat kode dan membuat laporan	● Membuat grayscale dan membuat decoder. ● Membuat laporan dan ppt.	Sihombing Giovano Geraldo

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Alat dan perlengkapan yang digunakan dalam project ini adalah:

- Visual Studio Code
- ModelSim
- Github

2.2 IMPLEMENTATION

Implementasi CPU menggunakan VHDL dalam proyek ini memanfaatkan pendekatan Finite State Machine (FSM) untuk mengatur siklus instruksi. Entitas CPU mendefinisikan port input seperti clock (CPU_CLK), enable (ENABLE), dan instruksi masukan (INSTRUCTION_IN). Melalui arsitektur RTL, proyek ini memperkenalkan komponen-komponen seperti DECODER dan ALU, yang bekerja bersama untuk mengatur alur eksekusi instruksi. Proses utama pada CPU membentuk FSM yang melibatkan beberapa state seperti IDLE, FETCH, DECODE, READ, EXECUTE, dan COMPLETE.

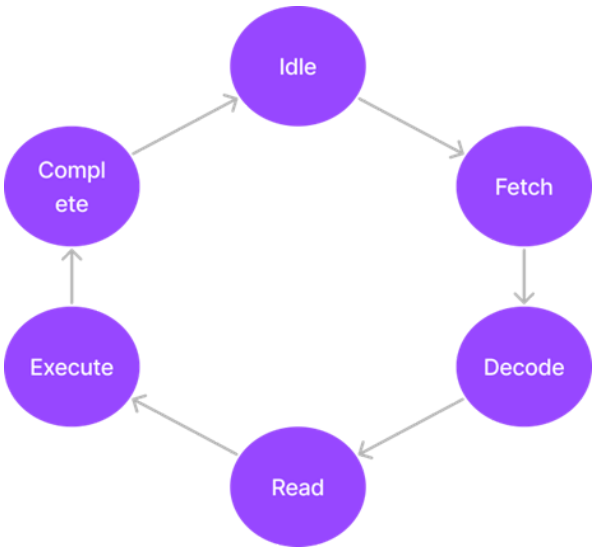

Fig 1. FSM Cycle

State awal adalah IDLE, di mana CPU menunggu sinyal enable untuk diaktifkan. Setelah diaktifkan, CPU bergerak ke state FETCH untuk mengambil instruksi dari port input INSTRUCTION_IN. State DECODE digunakan untuk memproses instruksi menggunakan komponen decoder dan mendapatkan sinyal opcode. State READ membaca nilai dari RAM sesuai dengan alamat yang dihasilkan dari proses decode. State EXECUTE melibatkan operasi ALU untuk menghasilkan nilai yang akan ditulis kembali ke RAM sesuai dengan alamat yang dihasilkan dari proses decode. Setelah instruksi selesai dieksekusi, state COMPLETE diaktifkan, dan CPU kembali ke state IDLE.

ALU dalam proyek ini mencakup berbagai komponen pengolahan gambar seperti grayscale, greenscale, redscale, dan bluescale. Setiap komponen ini diintegrasikan dengan ALU berdasarkan opcode yang diterima dari CPU. Selain itu, proyek ini juga melibatkan proses tambahan pada ALU untuk membaca gambar BMP dari file, melakukan operasi pengolahan gambar sesuai opcode yang diterima, dan menulis hasilnya ke file keluaran. Proses ini berjalan dalam loop untuk setiap piksel dan setiap baris gambar, memastikan bahwa operasi pengolahan gambar dilakukan dengan tepat.

Sebagai langkah terakhir, proyek ini mengandung proses untuk memverifikasi format file BMP, menginisialisasi variabel-variabel yang diperlukan, dan menutup file sumber dan tujuan setelah selesai. Keseluruhan implementasi ini menciptakan solusi yang efisien untuk pengolahan gambar pada tingkat instruksi menggunakan perangkat keras FPGA. Dengan menggunakan pendekatan FSM dan integrasi yang baik antara CPU, DECODER, ALU, RAM dan komponen-komponen pengolahan gambar, proyek ini memungkinkan eksekusi instruksi secara berurutan dengan optimal.

Image tools ini bekerja dengan cara membaca file gambar berwarna dalam format .bmp. File gambar .bmp merupakan format gambar yang umum digunakan dan didukung oleh berbagai perangkat lunak. File gambar .bmp terdiri dari header yang berisi informasi tentang ukuran, resolusi, dan format gambar, serta data gambar yang berisi nilai-nilai pixel gambar.

Image tools membaca header file gambar untuk mendapatkan informasi tentang ukuran dan resolusi gambar. Informasi ini kemudian digunakan untuk menginisialisasi memori yang akan digunakan untuk menyimpan data gambar.

Data gambar kemudian dibaca dan diproses untuk mengubahnya menjadi grayscale. Data gambar grayscale kemudian disimpan kembali ke memori. File gambar grayscale kemudian dapat ditulis ke disk. File gambar .bmp merupakan format gambar yang umum digunakan dan didukung oleh berbagai perangkat lunak. Format gambar .bmp memiliki struktur yang sederhana dan mudah untuk diimplementasikan dalam bahasa VHDL.

Format gambar lain, seperti file gambar .jpg atau .png, memiliki struktur yang lebih kompleks dan sulit untuk diimplementasikan dalam bahasa VHDL. Selain itu, format gambar lain juga sering menggunakan kompresi, yang dapat menyulitkan proses dekompresi dalam bahasa VHDL. Oleh karena itu, dalam proyek ini hanya file gambar .bmp yang digunakan.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Dalam tahap pengujian proyek ini, fokus utama dari Kelompok BP-08 adalah memastikan akurasi dan keandalan implementasi pada setiap skala pengolahan gambar grayscale, red scale, green scale, dan blue scale. Pengujian dilakukan untuk memverifikasi bahwa setiap piksel pada gambar berhasil dikonversi dengan benar ke tingkat keabuan yang tepat sesuai dengan skala yang diinginkan.

Kelompok BP-08 melakukan empat jenis pengujian, masing-masing terfokus pada skala pengolahan gambar yang berbeda. Pengujian grayscale dirancang untuk memastikan bahwa konversi ke skala abu-abu dilakukan dengan akurat. Sementara pengujian red scale, green scale, dan blue scale masing-masing bertujuan memverifikasi keberhasilan transformasi warna merah, hijau, dan biru pada setiap piksel gambar. Dengan demikian, hasil pengujian ini diharapkan dapat memberikan gambaran yang menyeluruh tentang keandalan dan kualitas implementasi fungsi pengolahan gambar pada tingkat piksel.

3.2 RESULT

Gambar sebelum diproses merupakan gambar sumber yang menjadi objek pengujian dalam proyek ini. Gambar ini mewakili dataset awal sebelum mengalami transformasi oleh fungsi-fungsi pengolahan gambar pada tingkat piksel yang diimplementasikan. Setiap piksel pada gambar ini memiliki warna yang bervariasi.

Fig 2. Image before processing

Hasil grayscale menunjukkan gambar yang telah melalui proses konversi, di mana setiap pikselnya diubah menjadi tingkat keabuan yang sesuai. Warna pada gambar ini telah disederhanakan menjadi skala abu-abu, menciptakan efek monokromatik yang mencerminkan tingkat keabuan yang akurat dari setiap piksel.

Fig 3. Grayscale processing

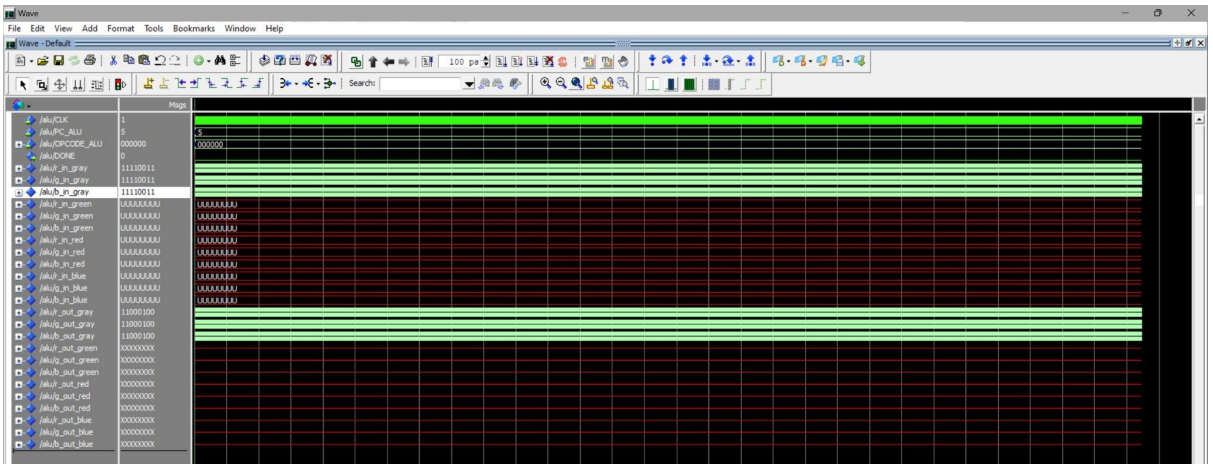

Fig 4. ModelSim run result for grayscale

Hasil red scale mencerminkan efek transformasi pada warna merah dari gambar. Setiap gambar hasil menyoroti piksel-piksel yang telah disesuaikan dengan tingkat warna ang diinginkan, menciptakan efek visual yang mencolok dan memperlihatkan perubahan warna yang telah terjadi setelah melalui proses konversi pada tingkat piksel.

Fig 5. Red scale processing

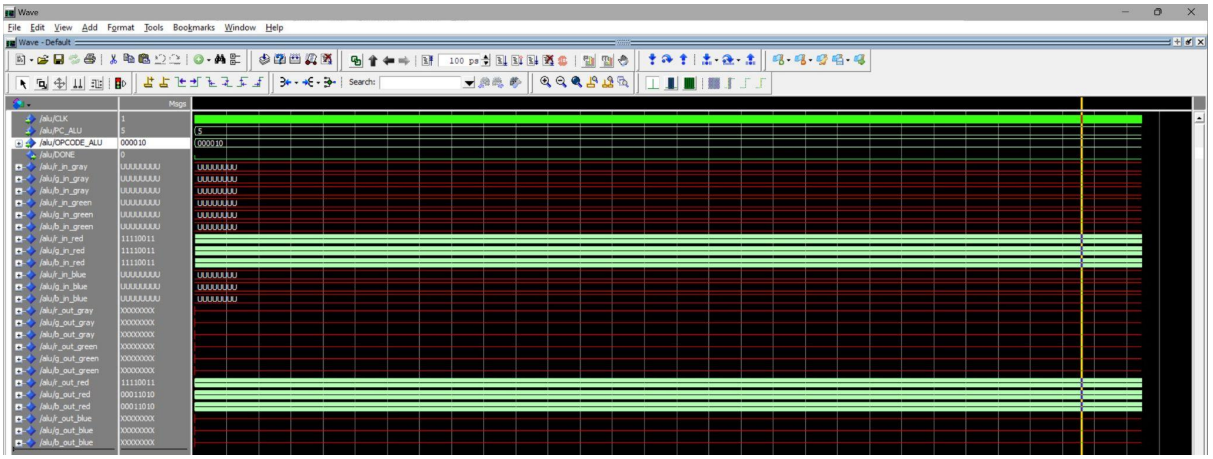

Fig 6. ModelSim run result for red scale

Hasil green scale mencerminkan efek transformasi pada warna hijau dari gambar. Setiap gambar hasil menyoroti piksel-piksel yang telah disesuaikan dengan tingkat warna hijau, menciptakan efek visual yang mencolok dan memperlihatkan perubahan warna yang telah terjadi setelah melalui proses konversi pada tingkat piksel.

Fig 7. Green scale processing

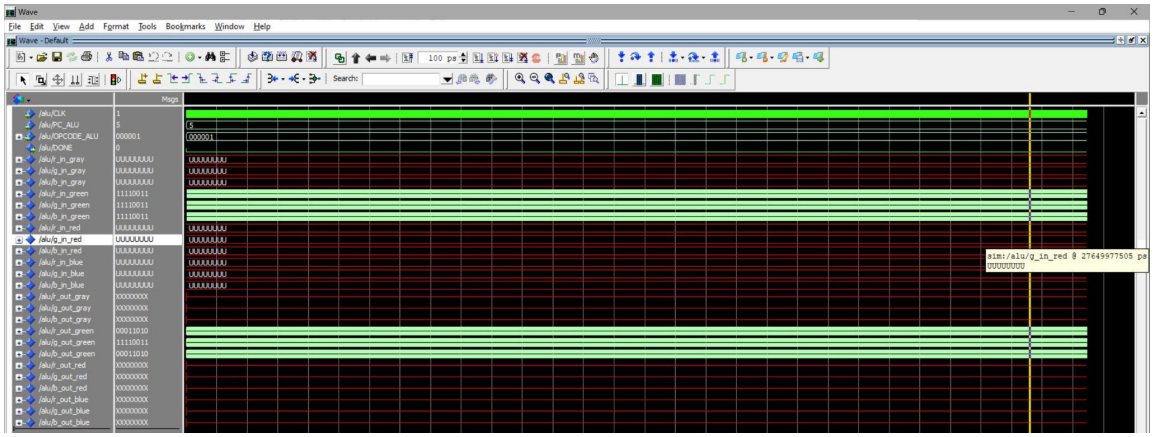

Fig 8. ModelSim run result for green scale

Hasil blue scale mencerminkan efek transformasi pada warna biru dari gambar. Setiap gambar hasil menyoroti piksel-piksel yang telah disesuaikan dengan tingkat warna biru, menciptakan efek visual yang mencolok dan memperlihatkan perubahan warna yang telah terjadi setelah melalui proses konversi pada tingkat piksel.

Fig 9. Blue scale processing

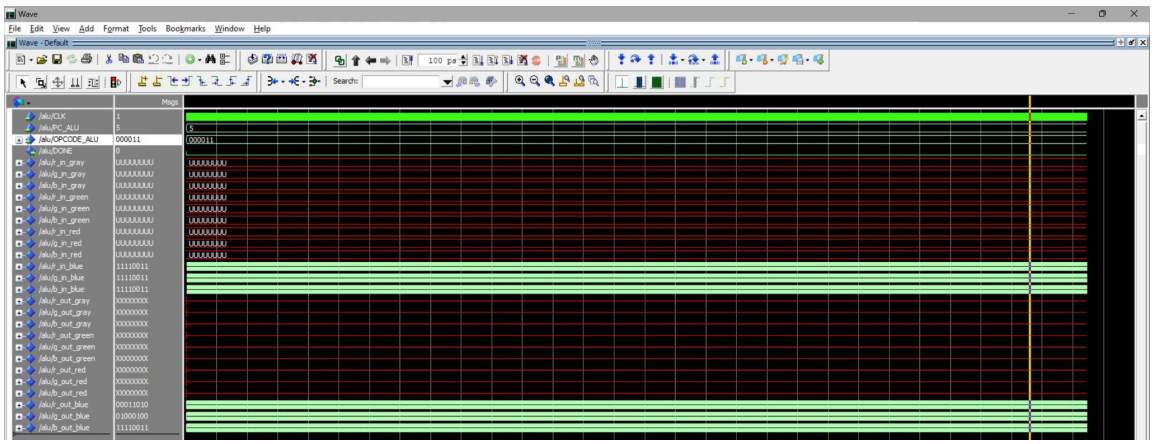

Fig 10. ModelSim run result for blue scale

3.3 ANALYSIS

Proses default color to grayscale akan mengubah gambar RGB menjadi grayscale dengan mengkalkulasikan weighted sum dari warna merah, hijau, dan biru. Output yang dihasilkan adalah output sinyal 8-bit dengan nama “luma” yang merepresentasikan value grayscale dari gambar.

Proses grayscale to red scale akan di-trigger oleh rising edge dari sinyal clk. Proses ini akan melakukan proses melakukan penskalaan merah dengan mengalikan nilai warna merah input (r_in_red, g_in_red, b_in_red) dengan bobot yang sesuai, lalu mengubah ukuran hasilnya agar sesuai dengan panjang sinyal weighted (r_weighted, g_weighted, b_weighted).

Proses grayscale to green scale akan di-trigger oleh rising edge dari sinyal clk. Proses ini akan melakukan proses melakukan penskalaan hijau dengan mengalikan nilai warna hijau input (r_in_green, g_in_green, b_in_green) dengan bobot yang sesuai, lalu mengubah ukuran hasilnya agar sesuai dengan panjang sinyal weighted (r_weighted, g_weighted, b_weighted).

Proses grayscale to blue scale akan di-trigger oleh rising edge dari sinyal clk. Proses ini akan melakukan proses melakukan penskalaan biru dengan mengalikan nilai warna biru input (r_in_blue, g_in_blue, b_in_blue) dengan bobot yang sesuai, lalu mengubah ukuran hasilnya agar sesuai dengan panjang sinyal weighted (r_weighted, g_weighted, b_weighted).

CHAPTER 4

CONCLUSION

Kesimpulan dari deskripsi proyek ini adalah bahwa proyek berhasil mengimplementasikan dua fitur utama, yaitu konversi gambar ke skala abu-abu dan konversi gambar ke skala warna pilihan (merah, hijau, atau biru). Meskipun demikian, terdapat kendala pada implementasi fungsi resize yang tidak berhasil diimplementasikan sesuai dengan yang diinginkan. Keberhasilan konversi ke skala abu-abu dan warna pilihan menunjukkan kemampuan proyek dalam melakukan manipulasi warna pada tingkat piksel dengan akurat. Namun, kendala pada fungsi resize dapat menjadi area pengembangan lebih lanjut untuk mencapai fleksibilitas tambahan dalam pengaturan ukuran gambar. Keseluruhan, proyek ini memberikan kontribusi pada pemahaman pengolahan gambar pada tingkat piksel menggunakan VHDL, meskipun masih memiliki potensi untuk diperbaiki dan dikembangkan lebih lanjut pada fitur resize.

REFERENCES

[1] “Grayscale,” *Wikipedia*, Oct. 17, 2021. https://en.wikipedia.org/wiki/Grayscale

[2] J. J. Jensen, “BMP file bitmap image read using TEXTIO,” *VHDLwhiz*, Nov. 13, 2019. https://vhdlwhiz.com/read-bmp-file/ (accessed Dec. 24, 2023).

[3] C.-C. Liu, T.-T. Lee, S.-R. Xiao, Y.-C. Lin, Y. Lin, and C.-C. Wong, “Real-Time FPGA-Based Balance Control Method for a Humanoid Robot Pushed by External Forces,” *Real-Time FPGA-Based Balance Control Method for a Humanoid Robot Pushed by External Forces*, vol. 10, no. 8, pp. 2699–2699, Apr. 2020, doi: https://doi.org/10.3390/app10082699.

[4] johonkanen, “Floating Point arithmetic in High Level VHDL,” *Hardware Descriptions*, Mar. 31, 2022. https://hardwaredescriptions.com/floating-point-in-vhdl/ (accessed Dec. 24, 2023).

[5] A. Mireles, “Monochrome vs Grayscale Photography: Key Differences,” *Shotkit*, Oct. 04, 2021. https://shotkit.com/monochrome-vs-grayscale/ (accessed Dec. 24, 2023).

APPENDICES

Appendix A: Project Schematic

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY CPU IS
    PORT (
        CPU_CLK : IN STD_LOGIC;
        ENABLE : IN STD_LOGIC;
        INSTRUCTION_IN : IN STD_LOGIC_VECTOR(49 DOWNTO 0)
    );
END ENTITY CPU;

ARCHITECTURE rtl OF CPU IS
    COMPONENT DECODER IS
        PORT (
            PROGRAM_COUNTER : IN INTEGER;
            INSTRUCTION : IN STD_LOGIC_VECTOR(0 TO 49);
            OPCODE : OUT STD_LOGIC_VECTOR(0 TO 5)
        );
    END COMPONENT DECODER;

    COMPONENT ALU IS
        PORT (
            CLK : IN STD_LOGIC;
            PC_ALU : IN INTEGER;
            OPCODE_ALU : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
            DONE : OUT STD_LOGIC
        );
    END COMPONENT ALU;

    TYPE State_type IS (IDLE, FETCH, DECODE, READ, EXECUTE, COMPLETE);
    SIGNAL state : State_type := IDLE;

    SIGNAL PC : INTEGER := 0;
    SIGNAL opcode, opcode_in : STD_LOGIC_VECTOR(5 DOWNTO 0);
    SIGNAL done : STD_LOGIC;

BEGIN

    DECODER_HAHA : DECODER PORT MAP(PC, INSTRUCTION_IN, opcode);

    PROCESS (CPU_CLK) IS
    BEGIN
        IF rising_edge(CPU_CLK) THEN
            IF ENABLE = '1' THEN
                CASE state IS
                    WHEN IDLE =>
                        PC <= PC + 1;
                        IF PC = 1 THEN
                            state <= FETCH;
                        END IF;
                    WHEN FETCH =>
                        PC <= PC + 1;
                        IF PC = 2 THEN
                            state <= DECODE;
                        END IF;
                    WHEN DECODE =>
                        PC <= PC + 1;
                        IF PC = 3 THEN
                            state <= READ;
                        END IF;
                    WHEN READ =>
                        opcode_in <= opcode;
                        PC <= PC + 1;
                        IF PC = 4 THEN
                            state <= EXECUTE;
                        END IF;
                    WHEN EXECUTE =>
                        PC <= PC + 1;
                        IF PC = 5 THEN
                            state <= COMPLETE;
                        END IF;
                    WHEN COMPLETE =>
                        PC <= 0;
                        state <= IDLE;
                END CASE;
            END IF;
        END IF;
    END PROCESS;
END ARCHITECTURE rtl;
```

Appendix B: Documentation