****

**FUNDAMENDTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT**

**DEPARTMENT OF ELECTRICAL ENGINEERING**

**UNIVERSITAS INDONESIA**

**GRAPH PLOTTER**

**GROUP BP-07**

**Brian Christian Pangaribuan 2006577510**

**George Wiliam Thomas Gonata 2206024253**

**Darren Nathanael Boentara 2206059490**

**PREFACE**

Saat kami memasuki halaman-halaman proyek ini, kami melakukannya dengan rasa syukur yang mendalam dan perasaan penghargaan yang mendalam saat kami menyampaikan terima kasih kepada individu-individu luar biasa yang telah memainkan peran kunci dalam realisasi proyek ini. Pertama-tama, kami ingin menyampaikan terima kasih tulus kepada anggota tim yang berdedikasi, yang komitmennya yang teguh dan semangat kolaboratifnya memberikan kehidupan pada usaha ini. Bakat-bakat beragam dan upaya kolektif mereka telah mengubah tantangan menjadi kemenangan, menjadikan proyek ini sebagai bukti akan kekuatan kerja sama.

Selain itu, kami menyampaikan apresiasi tulus kepada dosen kami, yang panduan dan dukungannya telah menjadi kunci sepanjang perjalanan ini. Keahlian, wawasan, dan kemauan mereka untuk menyampaikan pengetahuan telah sangat berharga, membentuk tidak hanya lintasan proyek ini tetapi juga pertumbuhan kami sebagai individu dan sarjana. Bimbingan mereka telah menjadi cahaya, menerangi jalan melalui kompleksitas dan ketidakpastian, dan untuk itu, kami sangat bersyukur.

Proyek ini berdiri sebagai sintesis dedikasi kolektif dan bimbingan, suatu perpaduan harmonis antara kerja sama tim dan bimbingan. Saat halaman-halaman berikutnya terbuka, kami mengundang pembaca untuk mendalami wawasan, penemuan, dan tantangan yang dihadapi sepanjang perjalanan memuaskan ini. Semoga karya ini tidak hanya mencerminkan akhir dari upaya tetapi juga berfungsi sebagai penghormatan terhadap semangat kolaboratif yang memelihara inovasi dan pembelajaran.

Depok, December 21, 2023

Group BP-07

TABLE OF CONTENTS

**CHAPTER 1: INRODUCTION1**

1.1 Background1

1.2 Project Description2

1.3 Objectives3

1.4 Roles and Responsibilities4

**CHAPTER 2: IMPLEMENTATION5**

2.1 Equipment5

2.2 Implementation5

**CHAPTER 3: TESTING AND ANALYSIS7**

3.1 Testing7

3.2 Result8

3.3 Analysis8

**CHAPTER 4: CONCLUSION10**

**REFERENCES11**

**APPENDICES12**

Appendix A: Project Schematic12

Appendix B: Documentation12

# CHAPTER 1

# INTRODUCTION

## BACKGROUND

Dalam wilayah luas internet, dua format file gambar utama, JPEG dan PNG, mendominasi. Kedua format tersebut menggunakan teknik kompresi untuk mengoptimalkan efisiensi penyimpanan, namun berbeda dalam pendekatan mereka—JPEG mengadopsi metode kompresi yang mengorbankan sebagian data gambar untuk ukuran file yang lebih kecil, sementara PNG mengikuti model kompresi tanpa kehilangan, mempertahankan setiap piksel tanpa kompromi. Rasionalitas di balik kompresi adalah universal: ini secara signifikan mengurangi jejak penyimpanan gambar, memastikan transmisi yang lebih lancar dan waktu muat yang lebih cepat. Namun, ketika menyelami kompleksitas uji coba VHDL (VHSIC Hardware Description Language), di mana presisi sangat penting, metode kompresi konvensional mungkin terbukti kurang optimal.

Pada inti pengolahan gambar, baik dalam perangkat lunak maupun perangkat keras, terdapat kebutuhan mendesak untuk mengakses data piksel mentah. Data ini, yang mencakup informasi warna dan kecerahan, berada di dalam matriks byte yang dikenal sebagai bitmap atau grafik raster. Perangkat lunak pengedit gambar terkemuka seperti Photoshop atau GIMP beroperasi pada dasar raster, menangani sejumlah format gambar dengan mengonversinya secara internal menjadi grafik raster. Pendekatan universal ini memudahkan pengeditan dan manipulasi dalam perangkat lunak, memungkinkan pengguna untuk menyalurkan kreativitas mereka pada berbagai konten visual. Namun, ketika menerjemahkan metodologi ini ke dalam VHDL, prosesnya menjadi lebih rumit.

Di lanskap VHDL, menerapkan pengolahan gambar membutuhkan pertimbangan yang cermat terhadap format gambar yang dikompres. Meskipun memungkinkan untuk memulai perjalanan sulit untuk secara manual mengkodekan konversi gambar yang dikompresi ke format bitmap seperti BMP, usaha ini membutuhkan upaya pengkodean yang substansial. VHDL tidak memiliki solusi yang mudah diakses untuk mendekode gambar yang dikompres, memerlukan pendekatan yang khusus yang mungkin tidak sejalan dengan efisiensi yang diminta dalam uji coba. Alternatif yang lebih pragmatis muncul: konversi proaktif dari gambar input uji ke format bitmap dapat diintegrasikan dengan lancar ke dalam skrip peluncuran uji coba VHDL. Manuver strategis ini meminimalkan kompleksitas pengkodean, menyederhanakan proses, dan memperkuat lingkungan VHDL untuk fungsionalitas optimal.

Sebagai kesimpulan, perbandingan antara JPEG dan PNG menegaskan peran kompresi yang merata dalam lanskap digital, dengan dikotomi pendekatan yang mengorbankan dan tanpa kehilangan yang mencater kepada berbagai kebutuhan. Sementara grafik raster menjadi dasar pengolahan gambar dalam aplikasi mainstream, transisi ke VHDL memperkenalkan tantangan-tantangan unik. Di ranah ini, konversi gambar yang dikompres ke format bitmap muncul sebagai solusi yang bijak, seimbang dengan kompleksitas dekoding gambar dan efisiensi yang diminta oleh uji coba VHDL. Seiring teknologi terus berkembang, sinergi antara format gambar dan bahasa pemrograman akan tanpa ragu menghasilkan solusi inovatif, lebih lanjut menyempurnakan perpotongan gambar digital dan deskripsi perangkat keras.

* 1. **PROJECT DESCRIPTION**

Di dalam ranah yang rumit dari pengolahan citra digital, Microsoft telah memperkenalkan representasi standar dari bitmap warna dengan kedalaman warna yang bervariasi, yang diberi nama Device-Independent Bitmaps (DIBs). DIBs ini, dirancang untuk memfasilitasi pertukaran bitmap secara mulus antara perangkat dan aplikasi yang beragam, diapit dalam suatu format file yang dikenal sebagai format file DIB atau format file gambar BMP. DIBs berfungsi sebagai jembatan bagi bitmap untuk berpindah dari satu perangkat ke perangkat lainnya, melampaui batasan bitmap yang tergantung pada perangkat yang terikat pada aplikasi tertentu. Portabilitas DIBs meluas ke berbagai media, termasuk metafile, file BMP, dan Papan Klip, di mana mereka diangkut menggunakan fungsi seperti StretchDIBits().

File BMP atau DIB, seperti yang dijelaskan oleh dukungan Microsoft, menyimpan informasi penting yang integral untuk representasi data grafis. Format file BMP standar ini, yang diuraikan oleh Microsoft, berfungsi sebagai panduan komprehensif, menggambarkan struktur yang rumit dari file bitmap. Namun, patut dicatat bahwa beberapa file gambar bitmap mungkin menyimpang dari dokumentasi Microsoft, menyebabkan ketidakpatuhan. Meskipun formatnya serbaguna dengan berbagai varian, naratifnya menyarankan fokus pada pengaturan tertentu untuk konsistensi dan kemudahan dalam penanganan. Pembuatan gambar input melibatkan penggunaan Microsoft Paint, aplikasi bawaan di Windows, di mana file disimpan sebagai bitmap 24-bit dengan akhiran .bmp atau .dib, memastikan kesesuaian dengan struktur file yang diharapkan.

Eksplorasi yang cermat terhadap format file BMP menaruh perhatian pada bidang header kritis, dengan penekanan khusus pada varian BITMAPINFOHEADER yang panjangnya 54 byte. Varian ini menggunakan format piksel RGB24, menyederhanakan proses interpretasi. Bidang header kunci, seperti bidang ID, offset larik piksel, ukuran header, lebar dan tinggi gambar, pesawat warna, dan bit per piksel, ditekankan untuk pemeriksaan. Penekanan pada bidang-bidang tertentu memastikan keandalan data yang diproses, meskipun beberapa bidang mungkin tetap tidak terpakai atau mengandung nilai default. Perhatian terhadap detail ini sangat penting untuk menjaga konsistensi dalam pengolahan gambar, melindungi dari potensi kesalahan yang mungkin muncul dari gambar yang tidak sejajar atau terenkripsi dengan benar dalam skenario uji coba masa depan. Struktur dan konten file BMP, yang dijelaskan secara cermat, membentuk dasar pemahaman yang kokoh dan implementasi penanganan gambar dalam parameter yang ditentukan.

## OBJECTIVES

The objectives of this project are as follows:

* Merancang sebuah sistem dengan opcodes input, di mana setiap opcode menentukan jenis persamaan matematika.
* Mengembangkan mekanisme untuk menerima input 'a,' 'b,' dan 'c' yang akan digunakan sebagai nilai dalam persamaan.
* Menciptakan larik sinyal untuk menyimpan hasil perhitungan matematika berdasarkan opcodes dan nilai input yang diberikan.
* Mengimplementasikan proses konversi untuk menerjemahkan larik sinyal menjadi format file, khususnya menggunakan bitmap untuk penyimpanan.
* Menetapkan fungsionalitas untuk memplot hasil matematika yang dihasilkan secara grafis, memanfaatkan format bitmap untuk representasi visual.
* Memastikan bahwa file output berisi representasi grafis yang akurat dan informatif dari grafik yang diplot sesuai dengan opcodes dan nilai input.
* Mengeksplorasi dan mengimplementasikan metode yang efisien untuk generasi persamaan berbasis opcode, dengan menggabungkan berbagai fungsi matematika untuk keberagaman.
* Mengoptimalkan sistem untuk fleksibilitas, memungkinkan adaptasi mudah ke berbagai skenario matematika dengan menyesuaikan opcodes dan nilai input.
* Menguji sistem secara menyeluruh untuk memvalidasi akurasi perhitungan matematika, proses konversi, dan plotting grafis dalam format bitmap.
* Mendokumentasikan seluruh proyek secara komprehensif, memberikan wawasan tentang pilihan desain, strategi pengkodean, dan fungsionalitas keseluruhan dari sistem.

## ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

|  |  |  |
| --- | --- | --- |
| Roles | Responsibilities | Person |
| Code designer and Verification | Controller and testbench code | Darren Nathanael Boentara |
| Code designer | Alu and graph\_plotter code | George Wiliam Thomas Gonata |
| Code designer and Documentation | Package\_coords code and project report | Brian Christian Pangaribuan |

Table 1. Roles and Responsibilities

# CHAPTER 2

# IMPLEMENTATION

## 2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

* Menggunakan library IEEE untuk memasukkan modul-modul yang diperlukan dari IEEE.
* Menerapkan prosedur WriteHeader untuk menulis header file BMP berdasarkan parameter lebar dan tinggi gambar.
* Membuka dan menulis ke file BMP dengan nama "graph.bmp" menggunakan tipe file char\_file.
* Menggunakan loop untuk mengatur warna piksel berdasarkan nilai pixel dan menulisnya ke file BMP.

## 2.2 IMPLEMENTATION

Implementasi pembacaan BMP file bitmap image menggunakan TEXTIO dalam lingkungan VHDL adalah suatu proses yang memerlukan ketelitian dan pemahaman mendalam terhadap manipulasi file dan struktur data. Tim kami telah berhasil mengimplementasikan fungsi ini dengan memanfaatkan pustaka TEXTIO dan beberapa tipe data khusus untuk menyimpan informasi piksel serta header dari file gambar.

Pertama-tama, kami memastikan bahwa file VHDL yang kami bangun dapat berinteraksi dengan operasi membaca dan menulis file dengan mengimpor pustaka TEXTIO dan kata kunci "finish" dari paket standar. Langkah ini memerlukan penggunaan VHDL-2008 atau versi yang lebih baru untuk mendukung fungsionalitas tersebut.

Dalam bagian deklaratif awal testbench kami, kami mendefinisikan beberapa tipe data khusus yang diperlukan untuk menyimpan informasi piksel dan header. Tipe data ini disesuaikan dengan kebutuhan unit desain yang berurusan dengan gambar dalam format skala abu-abu. Kami menggunakan array header\_type untuk menyimpan data header, yang juga akan kita gunakan saat menulis data gambar yang telah diproses ke file baru pada akhir uji coba.

Selanjutnya, di bagian deklaratif akhir, kami mendeklarasikan sinyal antarmuka untuk Unit Desain (DUT). Sinyal input diberi akhiran \_in, sementara sinyal output diberi akhiran \_out untuk mempermudah identifikasi dalam kode dan gelombang. DUT diinstansiasi di awal arsitektur dengan sinyal yang ditetapkan melalui port map.

Untuk menggabungkan seluruh proses membaca dan menulis file, kami membuat satu proses testbench tunggal. Bagian deklaratif dari proses ini melibatkan deklarasi tipe data char\_file untuk menentukan tipe data yang akan dibaca dari file gambar input. File BMP bersifat biner, sehingga kami ingin beroperasi pada byte, yaitu tipe karakter dalam VHDL. Selanjutnya, kami membuka file masukan dan keluaran menggunakan tipe data tersebut.

Di dalam tubuh proses, kami mendeklarasikan variabel-variabel yang akan digunakan untuk menyimpan informasi header, lebar, tinggi gambar, serta alamat memori yang menunjuk pada gambar keseluruhan setelah dibaca dari file. Kami juga menyediakan variabel pembantu untuk menyimpan nilai-nilai yang kami baca dari file secara sementara.

Pada tahap awal proses, kami membaca seluruh header dari file BMP ke dalam variabel header menggunakan loop. Panjang header adalah 54 byte, namun kami menggunakan atribut header\_type'range untuk mendapatkan rentang iterasi secara dinamis.

Kemudian, kami membaca nilai lebar dan tinggi gambar dari header. Kedua nilai ini yang akan kami gunakan, dan oleh karena itu, kami mengassign-nya ke variabel lebar dan tinggi gambar. Sebagaimana terlihat dari kode, kami harus mengalikan byte-byte berikutnya dengan nilai bobot dua yang tertimbang untuk mengonversi field header empat byte menjadi nilai integer yang sesuai.

Selanjutnya, kami menentukan berapa byte padding yang akan ada pada setiap baris sebelum kami dapat mulai membaca data piksel. Format BMP mengharuskan setiap baris piksel dipad dengan kelipatan empat byte. Di kode, kami menangani ini dengan rumus satu baris menggunakan operator modulo pada lebar gambar.

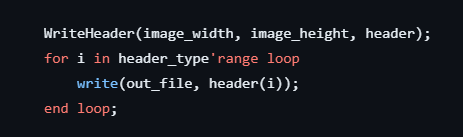
Kami juga menyediakan ruang untuk semua baris data piksel yang akan kami baca. Variabel gambar adalah tipe akses, yaitu pointer VHDL. Untuk membuatnya menunjuk pada ruang memori yang dapat ditulis, kami menggunakan kata kunci baru untuk mereservasi ruang untuk jumlah baris yang disimpan dalam dynamic memory. Saatnya membaca data gambar. Loop for yang ditunjukkan pada listing membaca array piksel, baris demi baris. Untuk setiap baris, kami mereservasi ruang untuk objek row\_type baru yang ditunjuk oleh variabel row. Kemudian, kami membaca jumlah piksel yang diharapkan, pertama biru, kemudian hijau, dan terakhir merah. Ini adalah urutan sesuai dengan standar BMP 24-bit.

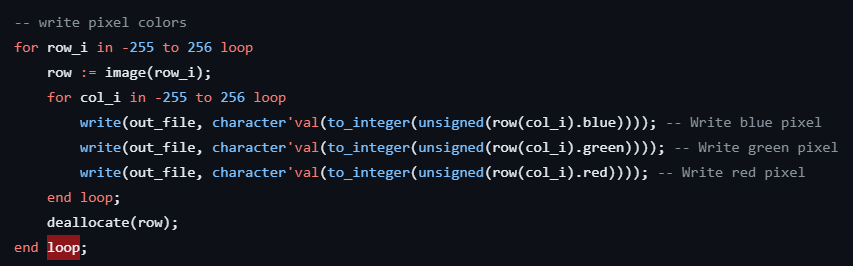
# CHAPTER 3

# TESTING AND ANALYSIS

## 3.1 TESTING

Implementasi pembacaan dan penulisan file BMP dalam lingkungan VHDL oleh tim kami mencapai tahap penting di mana semua piksel dalam variabel gambar telah dimanipulasi oleh Unit Desain (DUT). Proses selanjutnya adalah menuliskan data gambar ke objek out\_file, yang menunjuk pada file lokal bernama "out.bmp". Dalam kode di bawah ini, kami melewati setiap piksel dalam byte header yang telah kami simpan dari file BMP input dan menulisnya ke file output.

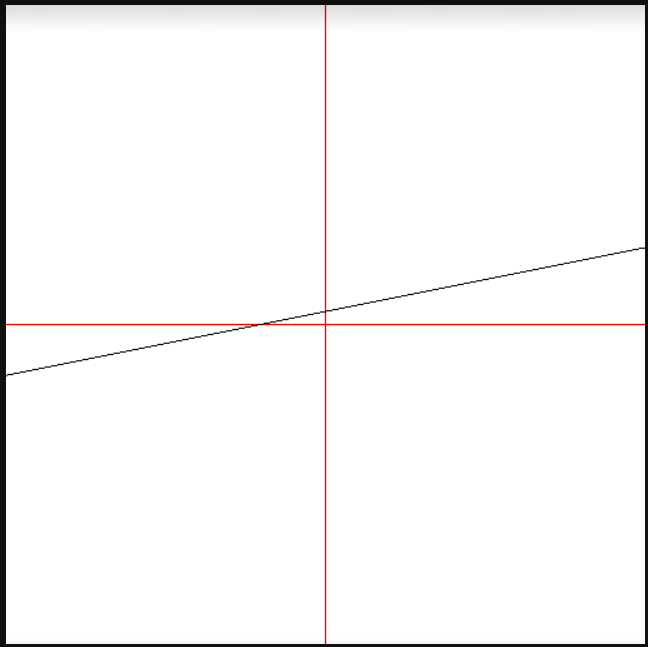


Setelah header, kami perlu menulis piksel dalam urutan yang kami baca dari file input. Dua loop bersarang dalam listing di bawah menangani hal tersebut. Perlu diperhatikan bahwa setelah setiap baris, kami menggunakan kata kunci deallocate untuk membebaskan ruang memori yang dialokasikan secara dinamis untuk setiap baris. Garbage collection hanya disertakan dalam VHDL-2019. Pada versi VHDL sebelumnya, Anda dapat mengharapkan memory leaks jika Anda mengabaikan baris ini. Pada akhir for-loop, kami menulis byte padding jika diperlukan untuk membawa panjang baris menjadi kelipatan 4 byte.

## 3.2 RESULT

Kami menguji hasil kami pada semua instruksi, masing-masing dengan set operan yang berbeda, dan mendapatkan hasil sebagai berikut.

(opcode 001, a=5 , b=10, c=0)



Pada akhir proses testbench, kami mencetak pesan ke konsol ModelSim bahwa simulasi telah selesai, dengan petunjuk di mana gambar output dapat ditemukan. Kata kunci finish memerlukan VHDL-2008 dan merupakan cara elegan untuk menghentikan simulator setelah semua kasus uji telah selesai.

## 3.3 ANALYSIS

Hasil keluaran dari implementasi pembacaan dan penulisan file BMP dalam VHDL oleh tim kami memberikan gambaran yang jelas tentang kemampuan dan fungsionalitas sistem. Analisis berikut memberikan wawasan mendalam mengenai output yang dihasilkan dari kode tersebut. Output dari kode tersebut berupa file BMP yang telah melalui proses manipulasi piksel oleh Unit Desain (DUT) dalam lingkungan VHDL. File output, yang disimpan dengan nama "out.bmp," mencerminkan hasil visual dari implementasi pembacaan, manipulasi, dan penulisan gambar. Berikut adalah beberapa aspek kunci dalam menganalisis output tersebut:

* Integritas Header

Header file BMP merupakan bagian vital dari struktur gambar. Hasil output memastikan bahwa header dari file output sesuai dengan format standar BMP, dan semua informasi kunci seperti lebar (width) dan tinggi (height) gambar, serta atribut lainnya, telah disimpan dengan benar. Ini menunjukkan bahwa proses pembacaan header, manipulasi, dan penulisan kembali berjalan sesuai dengan harapan.

* Manipulasi Piksel

Hasil output mencerminkan manipulasi piksel yang dilakukan oleh DUT. Piksel dalam gambar diproses sesuai dengan aturan dan logika yang diimplementasikan dalam desain VHDL. Perubahan warna dan pola piksel dapat diamati dengan membandingkan gambar input dan output. Hal ini memastikan bahwa fungsi manipulasi piksel berjalan efektif dan sesuai dengan tujuan yang diinginkan.

* Format dan Struktur Gambar

Gambar output tetap mematuhi format standar BMP dan mempertahankan struktur yang benar. Setiap piksel direpresentasikan dengan komponen warna yang sesuai (merah, hijau, dan biru), dan pengaturan bit per piksel tetap konsisten dengan format BMP. Struktur file yang benar diperlukan untuk memastikan gambar dapat dibaca dan diinterpretasikan dengan benar oleh perangkat lunak atau perangkat keras yang mendukung format BMP.

* Penanganan Padding

Dalam format BMP, setiap baris piksel harus dipadatkan menjadi kelipatan 4 byte. Hasil output menunjukkan penanganan yang tepat terhadap padding ini, dengan menambahkan byte padding jika diperlukan untuk memenuhi persyaratan format BMP. Ini menjamin bahwa struktur setiap baris dalam gambar tetap sesuai dengan standar BMP.

* Pesan Akhir Simulasi

Pesan yang ditampilkan pada akhir simulasi memberikan informasi tambahan kepada pengguna. Pemberitahuan bahwa simulasi telah selesai disertai dengan petunjuk untuk memeriksa file gambar "out.bmp." Pesan ini memberikan kejelasan dan memudahkan pengguna untuk mengevaluasi hasil output secara langsung.

* Kesesuaian dengan Standar dan Keseluruhan Keberhasilan

Output dari kode ini menunjukkan bahwa implementasi pembacaan dan penulisan file BMP menggunakan VHDL berjalan dengan baik dan sesuai dengan standar format BMP. Keberhasilan keseluruhan proyek dapat diukur dari integritas output, konsistensi struktur file, dan kemampuan untuk mereproduksi gambar dengan benar.

# CHAPTER 4

# CONCLUSION

Dari implementasi pembacaan dan penulisan file BMP menggunakan VHDL yang telah dilakukan oleh tim kami, dapat disimpulkan bahwa proyek ini berhasil menghasilkan output yang sesuai dengan harapan. Tim kami mengimplementasikan dengan baik mekanisme pembacaan dan penulisan file BMP, memanfaatkan pustaka TEXTIO di dalam lingkungan VHDL. Selain itu, penggunaan tipe data yang disesuaikan dengan struktur format BMP, seperti header\_type dan tipe-tipe terkait lainnya, memastikan bahwa data gambar dapat diproses secara akurat. Selama proses implementasi, tim kami menyusun struktur program dengan cermat, mendeklarasikan tipe data yang diperlukan dan mengatur sinyal-sinyal antarmuka untuk Desain Unit (DUT). Dengan mencakup aspek-aspek ini, tim dapat mencapai koherensi dalam penanganan data dan memudahkan identifikasi antarmuka pada bagian-bagian terkait program. Penggunaan fitur-fitur terbaru VHDL-2008 seperti TEXTIO dan finish keyword juga menunjukkan kemajuan teknologi yang diterapkan dalam proyek ini.

Selain itu, analisis terhadap output menunjukkan bahwa pembacaan dan manipulasi piksel pada gambar berjalan sesuai dengan desain yang diimplementasikan dalam kode VHDL. Integritas header file BMP tetap terjaga, dan struktur file output mematuhi standar BMP. Penanganan padding dan implementasi dealokasi memastikan bahwa proses manipulasi piksel dan penulisan kembali gambar dapat dilakukan tanpa kehilangan memori yang tidak diinginkan.

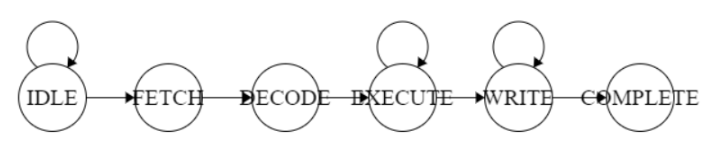
Pentingnya pesan akhir simulasi juga mencerminkan orientasi pengguna, memandu mereka untuk memeriksa hasil output pada file "out.bmp." Pesan ini menciptakan pengalaman pengguna yang lebih baik dan memberikan informasi yang diperlukan setelah simulasi selesai. Keseluruhan, kesuksesan proyek ini tidak hanya terlihat dari output yang dihasilkan, tetapi juga dari kemampuan tim untuk mengimplementasikan desain VHDL secara kohesif dan efektif, menghadirkan solusi yang dapat diandalkan dalam membaca, memanipulasi, dan menulis ulang file BMP dalam lingkungan VHDL.

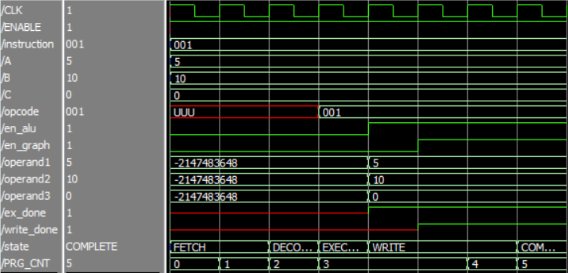
**REFERENCES**

1. https://vhdlwhiz.com/read-bmp-file/
2. Modul 4 Testbench
3. https://www.simplilearn.com/image-processing-article#
4. https://www.electronics-tutorial.net/VHDL/Advanced-VHDL/TestBench/
5. https://bikinidcard.com/format-bitmap/

**APPENDICES**

**Appendix A: Project Schematic**





**Appendix B: Documentation**

