



**LAPORAN PROYEK AKHIR PERANCANGAN SISTEM DIGITAL  
DEPARTEMEN TEKNIK ELEKTRO  
UNIVERSITAS INDONESIA**

**Microcoded Lanczos-3 Image Upscaling Engine ( $\mu$ -LISE)**

**KELOMPOK 22**

<b>Danish Putra Devananda</b>	<b>2406354202</b>
<b>Jonathan Christopher</b>	<b>2406349423</b>
<b>Mohammad Ariq Haqi</b>	<b>2406431271</b>
<b>Reyhan Batara</b>	<b>2406348950</b>

## TABLE OF CONTENT

<b>TABLE OF CONTENT.....</b>	<b>2</b>
<b>PREFACE.....</b>	<b>3</b>
<b>BAB 1: INTRODUCTION.....</b>	<b>4</b>
1.1 BACKGROUND.....	4
1.2 PROJECT DESCRIPTION.....	4
1.3 OBJECTIVES.....	5
1.4 ROLES AND RESPONSIBILITIES.....	5
<b>CHAPTER 2: IMPLEMENTATION.....</b>	<b>5</b>
2.1 EQUIPMENT.....	6
2.2 IMPLEMENTATION.....	6
<b>CHAPTER 3: TESTING AND ANALYSIS.....</b>	<b>8</b>
3.1 TESTING & RESULT.....	8
3.2 ANALYSIS.....	9
<b>CHAPTER 4: CONCLUSION.....</b>	<b>11</b>
<b>REFERENCES.....</b>	<b>13</b>
<b>APPENDICES.....</b>	<b>14</b>
Appendix A: Project Schematic.....	14
Appendix B: Documentation.....	15

## PREFACE

Puji dan syukur kehadiran Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga laporan proyek akhir Praktikum Perancangan Sistem Digital yang berjudul “*Microcoded Lanczos-3 Image Upscaling Engine ( $\mu$ -LISE)*” dapat diselesaikan dengan baik. Ucapan terima kasih kami sampaikan kepada asisten laboratorium *Digital Laboratory* UI, serta teman - teman yang telah berkontribusi dalam proses pengerjaan proyek akhir ini.

Laporan ini disusun untuk melengkapi proyek akhir Praktikum Perancangan Sistem Digital Tahun Ajaran 2025/2026. Laporan ini membahas tentang detail proyek yang telah kami rancang yaitu *image upscaler* menggunakan algoritma *Lanczos-3* yang diimplementasikan pada *Field Programmable Gate Array* dengan bahasa pemrograman VHDL. Laporan ini juga meliputi latar belakang, deskripsi, hasil serta analisis dari proyek yang telah kami buat.

Adapun karena adanya keterbatasan pengetahuan dan pengalaman, kami menyadari bahwa masih terdapat beberapa kekurangan dalam pengerjaan dan penyusunan laporan proyek akhir ini yang harus diperbaiki. Oleh karena itu, kami sangat mengharapkan adanya kritik maupun saran, sehingga dapat menjadi bahan evaluasi bagi kami kedepannya. Kami juga memohon maaf apabila terdapat kesalahan dan kekurangan dalam penyusunan laporan proyek akhir ini.

Depok, 7 Desember 2025

Kelompok 22

# **BAB 1**

## **INTRODUCTION**

### **1.1 BACKGROUND**

Digital image processing merupakan aspek yang sangat penting dan krusial dalam sistem Modern Computation. Salah satu operation yang paling umum digunakan adalah image upscaling, dimana sistem ini akan meningkatkan resolusi atau meningkatkan ketajaman dalam suatu photo or file. Seperti misalnya, Photo yang pecah dan berbentuk kotak kotak atau buram akan ditingkatkan resolusinya oleh Image Upscaling ini.

Algorithm Lanczos-3 Resampling merupakan Algorithm yang dikenal dapat menghasilkan kualitas gambar yang tinggi dengan fungsi jendela Sinc, dimana secara teoritis merupakan filter rekonstruksi ideal untuk meminimalkan *aliasing*, Algoritma lini menggunakan jendela 3-loba yang menawarkan keseimbangan optimal antara ketajaman dan kehalusan dibandingkan dengan metode metode lainnya termasuk dengan metode Bicubic. Untuk mengimplementasikan algoritma matematika kompleks ini kepada Hardware (FPGA), memiliki tantangan tersendiri karena keterbatasan sistem operation yang dinamakan floating-point.

Proyek kami akan mengimplementasikan Image upscaling ini ke dalam VHDL dengan menerapkan Microcoded Architecture kedalamnya Kami juga akan menggunakan aritmatika fixed-point untuk melakukan upscaling citra secara efisien pada lingkungan FPGA. Namun dengan hal ini, kami tertantang untuk membuat project ini untuk membuat suatu sistem Image Upscaling yang terbaik dan one and only yang pernah dibuat sebelumnya.

### **1.2 PROJECT DESCRIPTION**

Dalam Project ini, kami akan membuat image upscaler dengan Microcoded Lancos-3 dimana hal ini merupakan sistem digital yang mampu membaca data pixel dari file external, memprosesnya menggunakan filter konvulsi, dan menghasilkan luaran citra yang telah diperbesar resolusinya. Ada beberapa komponen Modular yang digunakan dalam Project VHDL ini, yaitu: Microcode ROM(untuk menyimpan koefisien filter), input RAM (untuk penyangga data), Datapath unit(untuk operasi aritmatikanya), dan

Control Unit (FSM) untuk mengatur proses. Sistem menggunakan interface file .txt standar untuk verifikasi simulasi.

### 1.3 OBJECTIVES

- Merancang dan mengimplementasikan algoritma interpolasi Lanczos-3 ke dalam level Register transfer Level (RTL) menggunakan VHDL.
- Menerapkan teknik Microprogramming untuk penyimpanan koefisien filter guna fleksibilitas desain dan penggunaan aritmatika fixed point untuk efisiensi komputasi FPGA
- Membangun FSM yang kuat untuk mengelola sinkronisasi data, pengalaman memory, dan alur eksekusi operasi aritmatika
- Memverifikasi kebenaran fungsional sistem melalui Behavioral Simulation menggunakan test bench yang komprehensif

### 1.4 ROLES AND RESPONSIBILITIES

Berikut Adalah pembagian tugas pada kelompok kami yang sudah didiskusikan lebih lanjut sesuai dengan Keterampilan masing masing

Roles	Nama	Responsibilities
Role 1	Danish	Laporan Proyek Akhir + README.md
Role 2	Jonathan	Code + Algorithm Improvements
Role 3	Ariq	Code Analysis + PowerPoint Slide
Role 4	Reyhan	Laporan Proyek Akhir + PowerPoint Slide

## CHAPTER 2

### IMPLEMENTATION

#### 2.1 EQUIPMENT

Alat yang digunakan untuk membuat Project ini adalah

- Modelsim (Platform utama untuk mengembangkan dan Menjalankan project ini)
- Vivado (Platform yang digunakan untuk testing/percobaan lebih lanjut pada project ini)
- Bitmap (Platform yang digunakan untuk Representasi Digital pada Gambar Input)

#### 2.2 IMPLEMENTATION

Implementasi sistem Microcoded Lanczos-3 Image Upscaling Engine ini akan dirancang secara modular dengan arsitektur Top Level Module yang akan diintegrasikan pada beberapa komponen utama untuk memproses citra digital. Berdasarkan Arsitektur sistem yang telah kami rancang, implementasi akan terbagi menjadi dua blok logika utama yaitu Controller dan Resample engine.

1. **Arsitektur Sistem (Top Module):** Modul utama ini akan berfungsi sebagai pembungkus yang menghubungkan Input Image Buffer yang merupakan penyangga data masuk dan Output Image Buffer yang bertugas sebagai penyangga hasil upscale. Sinyal kendali utama meliputi Clock, Reset, dan Start. Alur kerja sistem diatur oleh Finite State Machine (FSM) yang akan berpindah dari status Standby, Processing, hingga Done.
2. **Resample Engine & Lanczos-3 Kernel:** Inti dari pemrosesan ini adalah Resample Engine yang berfungsi untuk melakukan perhitungan matematika berat menggunakan fungsi Sinc. Implementasi ini menggunakan Window Size sebesar 6 (radius/support 3 pixel). Fungsi compute Sinc akan menerima nilai X yang merupakan jarak antara piksel asli dan posisi pixel baru untuk menghasilkan bobot (weight) yang menentukan seberapa besar pengaruh piksel tetangga

terhadap piksel baru. Jika nilai absolut  $x < 1.0 \times 10^{-5}$ , maka bobot dianggap 1.0 untuk menghindari kesalahan pembagian nol

3. **Border Replicator:** Salah satu tantangan utama dalam konvolusi citra ini adalah penanganan pixel di bagian border. Algoritma Lanczos-3 membutuhkan data "tetangga" di sekitar pixel target. Untuk pixel yang berada di ujung gambar dimana tidak memiliki tetangga, kami akan mengimplementasikan teknik Border Replicator. Teknik ini menciptakan "tetangga palsu" dengan menduplikasi warna piksel terluar atau disebut sebagai clamping, sehingga perhitungan interpolasi tetap dapat dilakukan tanpa menimbulkan artefak hitam pada pinggiran gambar

## CHAPTER 3

### TESTING AND ANALYSIS

#### 3.1 TESTING & RESULT

Cara kerja dari Lanczos-3 Image Upscaler ini sendiri akan dilakukan dengan melakukan simulation Test Bench yang akan dirancang untuk memproses file gambar berformat BMP(Bitmap)

##### **Mekanisme Pengujian:**

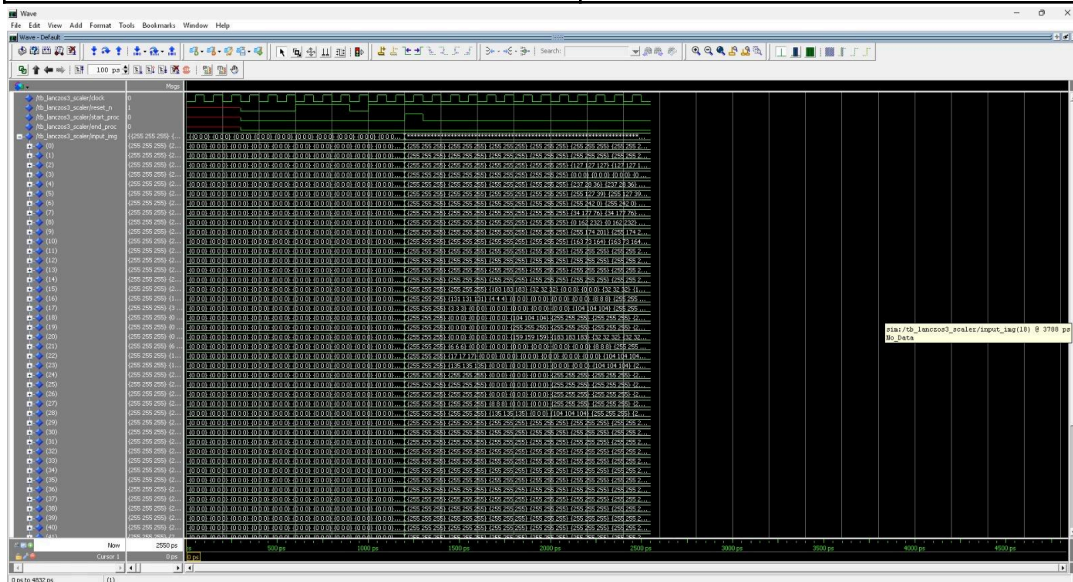
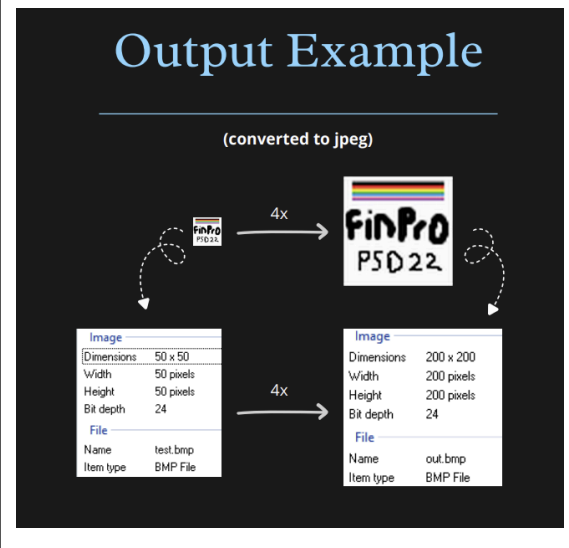
- **Pembacaan Header:** Testbench membaca 54 byte pertama dari file input (test.bmp) sebagai header. Sistem memverifikasi validitas file dengan memeriksa signature 'BM' pada dua byte pertama.
- **Kalkulasi Dimensi:** Sistem membaca informasi lebar dan tinggi gambar asli dari header, kemudian menghitung dimensi output baru berdasarkan faktor skala (4X). Header file output diperbarui dengan dimensi baru (200X200 pixel) agar file hasil (out.bmp) valid dan dapat dibuka oleh sistem operasi.
- **Proses Konvolusi:** Untuk setiap piksel output, sistem menghitung koordinat sumber (src\_x, src\_y) dan mengambil sampel warna (RGB) dari window 6x6. Sampel ini dikalikan dengan bobot Sinc yang telah dihitung sebelumnya, lalu dijumlahkan untuk menghasilkan nilai piksel akhir 9.

##### **Hasil Pengujian:**

Berdasarkan simulasi, sistem berhasil melakukan upscaling pada citra input berukuran 50 x 50 pixel menjadi 200 x 200 piksel (skala 4x). Gambar "FinPro PSD 22" yang dihasilkan menunjukkan peningkatan resolusi yang signifikan dengan dimensi file yang tervalidasi 10.



Output Example yang telah kami ambil dari PPT



### 3.2 ANALYSIS

Setelah melakukan uji pada Project ini, kami mendapatkan hasil yang cukup memuaskan dan perbedaan yang mumpuni, keberhasilan ini bisa dilihat pada hasil output yang sudah saya cantumkan di atas dan dari gambar tersebut kita bisa melihat bahwa hasilnya, pixel berhasil ditambahkan walaupun tidak sebagai image upscaler pada umumnya, namun setidaknya hasilnya jauh lebih baik dibandingkan input sebelumnya. Kalau kita menggunakan metode metode lain seperti contoh interpolasi sederhana(Nearest Neighbor) yang membuat gambar menjadi pixelated, algorithm Lanczos ini menghasilkan output yang jauh lebih smooth dan tajam. Hal ini bisa dicapai karena penggunaan Sinc

Function yang bekerja untuk filter rekonstruksi yang mempertahankan detail dari frekuensi tinggi pada citranya. Meskipun hasil kualitas visualnya lebih baik, Lanczos membutuhkan sumber daya komputasi yang lebih besar karena melibatkan operasi perkalian dan pembagian floating point atau fixed point di dalam implementasi FPGA untuk menghitung bobot sinc pada windows  $6 \times 6$ . Namun, penggunaan arsitektur Microcoded dan Border Replicator membuktikan bahwa hasilnya efektif untuk menangani kompleksitas ini secara terstruktur, memastikan setiap pixel, termasuk pada bagian tepinya akan terproses secara akurat.

## **CHAPTER 4**

### **CONCLUSION**

Pada Project ini kami bisa menyimpulkan bahwa Microcoded Lanczos-3 image upscaler ini berhasil dirancang menggunakan VHDL dengan menggunakan teknik Microcoded untuk penyimpanan koefisien filter dan aritmatika Fixed-Point yang terbukti secara teoritis yang mampu menggantikan Floating-Point Operation yang cukup kompleks, Sehingga memungkinkan kami untuk mengimplementasikan hal yang lebih efisien pada FPGA ini. Proyek ini akan mendemonstrasikan integrasi alat desain yang efektif dengan menggunakan 3 platform untuk melakukannya.

Team Developer kami lebih memilih Modelsim karena lebih gampang untuk dibaca dan lebih terstruktur sesuai dengan modul modul yang dilakukan oleh mereka sebelumnya, Beberapa Team kami menggunakan Vivado untuk melakukan testing pada kode mereka untuk mencari tahu apakah kode bekerja dengan baik. Modelsim dan Vivado memiliki fungsi yang sama untuk memverifikasi logika desain melalui simulasi behavioral dan kedua alat ini akan memastikan bahwa kode yang ditulis bebas dari kesalahan syntax dan logic dasar sebelum diimplementasikan ke fisik. Namun disini, input dan output akan diletakan pada file untuk mengetahui gambaran outputnya secara langsung dengan menggunakan gambar dengan file .bmp , karena VHDL sebenarnya hanya bisa mengkonversi gambar menjadi format .txt agar VHDL bisa membaca dan memproses intensitas pixel secara langsung.

Simulation menunjukkan bahwa sistem telah berhasil melakukan operation upscaling dengan faktor skala 4x dengan mengubah resolusi citra input 50x50 pixel menjadi 200x200 pixel secara akurat. Mekanisme pembacaan dan penulisan header BMP yang diimplementasikan memungkinkan sistem untuk menghasilkan file keluaran yang valid dan dapat langsung dibuka oleh komputer, membuktikan bahwa algorithm ini tidak hanya berjalan secara teoritis, namun juga secara functional dalam menangani format file yang nyata. Penerapan Border Replicator menjadi kunci keberhasilan dari kualitas visual pada project kami ini dan dengan teknik duplicating pixel tepi, sistem mampu mengatasi masalh artefak hitam pada pinggiran gambar yang sering terjadi pada algorithm konvulsi standar, membuat interpolasi Lanczos-3 dapat diterapkan secara merata hingga ke batasan terluar pada citra. Secara keseluruhan project ini

berhasil mencapai tujuannya untuk menerjemahkan kompleksitas matematis rumus Sinc ke dalam deskripsi Hardware(VHDL) yang efisien, menghasilkan citra yang lebih tajam dan halus jika dibandingkan dengan method interpolasi yang dasar. Struktur kode yang memisahkan Control Unit (FSM) dan Datapath memberikan fleksibilitas bagi Developer selanjutnya walaupun detail implementasi akhir masih dalam tahap penyempurnaan dari Development Team kami, arsitektur modern ini telah membuktikan bahwa memudahkan proses debugging dan verifikasi parsial pada setiap blok functionnya.

## REFERENCES

- [1] AMD Xilinx, "Vivado Design Suite User Guide: Synthesis (UG901)," *Xilinx Documentation*, v2023.1, 2023. [Online]. Available: <https://docs.xilinx.com/r/en-US/ug901-vivado-synthesis>. [Accessed: Dec. 07, 2024].
- [2] MathWorks, "Resize image - MATLAB imresize," *MathWorks Documentation*, 2024. [Online]. Available: <https://www.mathworks.com/help/images/ref/imresize.html>. [Accessed: Dec. 07, 2024].
- [3] Analog Devices, "Digital Signal Processing 101: Fixed-Point vs Floating-Point," *Analog Devices Technical Library*, 2021. [Online]. Available: <https://www.analog.com/en/resources/technical-articles/fixed-point-vs-floating-point-dsp.html>. [Accessed: Dec. 07, 2024].
- [4] **Siemens Digital Industries Software**, "ModelSim HDL Simulator - ASIC & FPGA Design," *Siemens EDA Software*. [Online]. Available: <https://eda.sw.siemens.com/en-US/ic/modelsim/>. [Accessed: Dec. 07, 2024].
- [5] **Intel Corporation**, "Video and Image Processing Suite User Guide: Scaler II Algorithm," *Intel FPGA Documentation*, 2024. [Online]. Available: <https://www.intel.com/content/www/us/en/docs/programmable/683246/current/scaler-ii-algorithm.html>. [Accessed: Dec. 07, 2024].
- [6] **Doulos**, "TextIO - Reading and Writing Files in VHDL," *Doulos Knowledge Base*, 2024. [Online]. Available: <https://www.doulos.com/knowhow/vhdl/file-io-in-vhdl/>. [Accessed: Dec. 07, 2024].

## APPENDICES

### Appendix A: Project Schematic

RTL Result:

Finite-State Diagram:

File	Screenshot
lanczos3_scaler.vhd	<pre> graph LR     Standby((STANDBY)) -- Reset --&gt; Standby     Standby -- start_proc = '1' --&gt; Processing((PROCESSING))     Processing -- engine_done = '1' --&gt; DoneState((DONE_STATE))     DoneState -- end_proc = '1' --&gt; Standby         </pre> <p>The diagram shows three states: STANDBY, PROCESSING, and DONE_STATE. STANDBY has a self-loop for 'Reset'. A transition from STANDBY to PROCESSING occurs when 'start_proc = '1''. From PROCESSING, a transition to DONE_STATE occurs when 'engine_done = '1''. Finally, a transition from DONE_STATE back to STANDBY occurs when 'end_proc = '1''.</p>
lanczos_resample_engine.vhd	<pre> graph LR     Idle((IDLE)) -- go = '1' --&gt; BorderPrep((BORDER_PREP))     BorderPrep -- border_done = '1' --&gt; Convolve((CONVOLVE))     Convolve -- for loop done --&gt; OutputWait((OUTPUT_WAIT))     OutputWait -- cnt = 2 --&gt; Complete((COMPLETE))     Complete --&gt; Idle         </pre> <p>The diagram shows five states: IDLE, BORDER_PREP, CONVOLVE, OUTPUT_WAIT, and COMPLETE. Transitions are: IDLE to BORDER_PREP on 'go = '1'', BORDER_PREP to CONVOLVE on 'border_done = '1'', CONVOLVE to OUTPUT_WAIT on 'for loop done', OUTPUT_WAIT to COMPLETE on 'cnt = 2', and a return from COMPLETE to IDLE.</p>
border_replicator.vhd	<pre> graph LR     Start((start)) --&gt; WaitTrigger((WAIT_TRIGGER))     WaitTrigger -- trigger = '1' --&gt; CopyCenter((COPY_CENTER))     CopyCenter -- loop done --&gt; ReplicateBorders((REPLICATE_BORDERS))     ReplicateBorders -- 3 loops done --&gt; Finish((FINISH))     Finish -- clk --&gt; WaitTrigger         </pre> <p>The diagram shows four states: WAIT_TRIGGER, COPY_CENTER, REPLICATE_BORDERS, and FINISH. Transitions are: an initial 'start' to WAIT_TRIGGER, WAIT_TRIGGER to COPY_CENTER on 'trigger = '1'', COPY_CENTER to REPLICATE_BORDERS on 'loop done', REPLICATE_BORDERS to FINISH on '3 loops done', and FINISH back to WAIT_TRIGGER on 'clk'.</p>

Appendix B: Documentation

Action	Screenshot
Diskusi di Group WhatsApp	