



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

LINEAR ALGEBRA MATRIX SOLVER

GROUP BP06

Evandita Wiratama Putra	2206059572
Hanif Nur Ilham Sanjaya	2206059692
Ivander Andreas Wijaya	2206031896
Monica Vierin Pasman	2206029405

PREFACE

Assalamualaikum Wr. Wb., salam sejahtera bagi kita semua.

Puji syukut kita panjatkan ke hadirat Allah, atas berkat dan rahmatnya kita dapat menyelesaikan proyek akhir praktikum perancangan sistem digital. Laporan proyek ini disusun sebagai pemenuhan dari komponen penilaian proyek akhir sebagai syarat penyelesaian praktikum perancangan sistem digital.

Praktikum ini telah membawa banyak sekali pengalaman dan pengetahuan dalam memahami perancangan dari konsep sistem digital yang telah kami peroleh dari kelas. Kami berharap kemampuan yang kami dapatkan selama mengikuti praktikum ini dapat kami terapkan kedepannya.

Kami menyadari bahwa laporan ini masih jauh dari kata sempurna, sehingga kami sangat menghargai kritik dan saran yang membangun dari pembaca untuk perbaikan dari laporan ini.

Akhir kata, kami ucapan terima kasih yang sebesar - besarnya kepada semua pihak yang sudah membantu dan mendukung kami selama proses praktikum berlangsung dan penyusunan laporan proyek ini.

Wassalamualaikum Wr. Wb, damai sejahtera bagi kita semua.

Depok, Desember 24, 2023

Group BP06

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION.....	4
1.1 BACKGROUND.....	4
1.2 PROJECT DESCRIPTION.....	5
1.3 OBJECTIVES.....	6
1.4 ROLES AND RESPONSIBILITIES.....	7
CHAPTER 2: IMPLEMENTATION.....	8
2.1 EQUIPMENT.....	8
2.2 IMPLEMENTATION.....	8
CHAPTER 3: TESTING AND ANALYSIS.....	11
3.1 TESTING.....	11
3.2 RESULT.....	16
3.3 ANALYSIS.....	21
CHAPTER 4: CONCLUSION.....	22
REFERENCES	
APPENDICES	

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Dalam era modern seperti sekarang, sistem digital telah menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari kita, mulai yang sangat dekat dengan kita seperti smartphone sampai yang agaknya lebih advanced seperti supercomputer. Inti dari sistem-sistem ini adalah rangkaian digital yang menjalankan berbagai fungsi. Salah satu fungsi tersebut adalah kemampuan untuk menyelesaikan masalah matematika, yang sangat penting dalam berbagai bidang, seperti teknik, fisika, dan ilmu komputer.

Salah satu permasalahan matematika yang penting dan sering digunakan adalah menyelesaikan operasi matriks pada aljabar linear. Menyelesaikan sistem-sistem ini dengan cepat dan akurat penting dalam banyak aplikasi, termasuk dalam komputer grafis, pembelajaran mesin, dan masalah optimisasi.

Secara tradisional, masalah-masalah ini telah diselesaikan dengan menggunakan perangkat lunak yang berjalan pada prosesor umum. Namun, seiring dengan terus meningkatnya permintaan akan kinerja yang lebih tinggi, ada minat yang meningkat dalam menggunakan solusi perangkat keras untuk mempercepat perhitungan-perhitungan ini. Salah satu bahasa deskripsi perangkat keras yang dapat digunakan untuk merancang dan mengimplementasikan solusi-solusi ini adalah VHDL (VHSIC Hardware Description Language).

VHDL memungkinkan desain rangkaian digital yang kompleks pada tingkat abstraksi yang tinggi. VHDL banyak digunakan di industri dan akademisi untuk merancang sistem digital, dan sangat cocok untuk merancang sistem yang menyelesaikan masalah-masalah matematika, berkat dukungannya terhadap paralelisme dan kontrol yang tepat terhadap sumber daya perangkat keras.

Berangkat dari latar belakang tersebut, kami mengajukan 'Linear Algebra Matrix Solver' yang merupakan komputer sederhana yang dirancang khusus untuk menyelesaikan berbagai permasalahan matriks pada aljabar linear. Rangkaian matriks prosesor ini akan

mengolah Instruction Set Architecture (ISA) yang dapat memberikan solusi atas permasalahan matriks berordo 3x3. ISA matriks tersebut memiliki opcode yang dapat digunakan untuk melakukan berbagai operasi matriks, seperti penjumlahan matriks A dan B, pengurangan matriks A dan B, perkalian matriks A dan B, pencerminan matriks A terhadap sumbu x, pencerminan matriks A terhadap sumbu y, pencerminan matriks A terhadap sumbu z, transpose matriks A, dan kofaktor matriks A. Selain itu, terdapat tambahan determinan matriks yang ditempatkan pada bagian dataflow.

1.2 PROJECT DESCRIPTION

Projek ini melibatkan desain dan implementasi simple computer menggunakan VHDL (VHSIC Hardware Description Language) yang dapat menyelesaikan persamaan matriks 3x3. Simple computer ini sendiri terdiri dari beberapa komponen, yang masing-masing memainkan peran penting dalam jalannya operasi sistem.

MatrixProcessor: Entitas utama yang mengkoordinasikan semua komponen. MatrixProcessor mendefinisikan dan menghubungkan semua komponen lain (ALU, DECODER, dan RAM). MatrixProcessor juga mengontrol alur eksekusi program menggunakan Finite State Machine (FSM) dengan state IDLE, FETCH, DECODE, READ_MEM, EXECUTE, dan COMPLETE.

ALU (Arithmetic Logic Unit): Komponen ini melakukan operasi aritmatika pada matriks. ALU menerima beberapa input termasuk program counter, opcode, dan dua set operand (A dan B). ALU menghasilkan determinan matriks dan hasil operasi-operasi matriks yang dipilih sesuai opcode.

DECODER: Komponen ini menerima instruksi dan program counter sebagai input. Decoder menghasilkan opcode dan alamat untuk tiga operand. Opcode menentukan operasi yang akan dilakukan pada matriks, dan alamat menunjukkan lokasi operand di memori.

RAM: Komponen ini berfungsi sebagai memori program. RAM menerima beberapa input termasuk program counter, dua alamat RAM, nilai input matriks 3x3, dan sinyal enable untuk write. RAM menghasilkan nilai output matriks 3x3 dari register RAM yang dipilih.

Tujuan dari proyek ini adalah berhasil mengimplementasikan sistem ini dan memastikan bahwa program dapat menyelesaikan permasalahan matriks 3x3 dengan akurat.

penulisan kode VHDL untuk setiap komponen, simulasi sistem untuk memverifikasi fungsionalitasnya, dan sintesis desain untuk mengimplementasikannya pada perangkat FPGA yang sesuai. Proyek juga ini akan mendemonstrasikan kemampuan dan fleksibilitas VHDL untuk merancang sistem digital kompleks dan memberikan pengalaman praktis dalam desain dan implementasi sistem digital.

1.3 OBJECTIVES

Tujuan-tujuan dari proyek adalah sebagai berikut:

1. Mendesain dan mengimplementasikan komponen-komponen utama dari sistem (Matrix Processor, ALU, DECODER, and RAM) menggunakan VHDL
2. Mengimplementasikan operasi matriks pada ALU untuk menyelesaikan masalah matriks 3×3 .
3. Men-decode ISA yang diberikan menggunakan DECODER untuk mengontrol jalannya operasi pada ALU.
4. Mengimplementasikan manajemen memori menggunakan RAM untuk menyimpan dan mengambil data matriks.
5. Mengintegrasikan semua komponen dalam Matrix Processor dan memastikan bahwa semua komponen bekerja sama dengan benar.
6. Mensimulasikan sistem untuk memverifikasi fungsionalitasnya dan memastikan dapat menyelesaikan persamaan matriks 3×3 dengan akurat.
7. Mensintesis desain untuk mengimplementasikannya pada perangkat FPGA yang sesuai.
8. Mendapatkan pengalaman praktis dalam desain dan implementasi sistem digital menggunakan VHDL.
9. Mendemonstrasikan kemampuan dan fleksibilitas VHDL untuk merancang sistem digital kompleks.

1.4 ROLES AND RESPONSIBILITIES

Berikut ini peran dan tanggung jawab setiap anggota kelompok.

Roles	Responsibilities	Person
Ketua	Komponen RAM, ALU, laporan bab 3, ppt, dan gambar rangkaian pada README.	Evandita Wiratama Putra
Anggota	Komponen Matrix Processor, laporan bab 1, ppt, dan snippet code pada README.	Hanif Nur Ilham Sanjaya
Anggota	Komponen Testbench, kata pengantar laporan, dan laporan bab 4.	Ivander Andreas Wijaya
Anggota	Komponen Decoder, ALU, laporan bab 2, dan README.	Monica Vierin Pasman

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

Alat-alat yang digunakan dalam pembuatan proyek akhir Perancangan Sistem Digital adalah sebagai berikut.

- Visual Studio Code
- ModelSim
- Quartus
- GitHub

2.2 IMPLEMENTATION

Concurrent Circuit Design

Concurrent circuit design merupakan rangkaian yang bekerja secara bersamaan (paralel). Perubahan input yang diberikan ke dalam desain ini akan langsung mempengaruhi perubahan output pada program. Pada program Linear Algebra Matrix Solver, concurrent design diterapkan menggunakan blok process yang terdapat pada setiap komponen. Pada komponen Matrix Processor, blok process diterapkan untuk menjalankan perpindahan state secara paralel. Pada komponen Decoder sendiri, blok proses diterapkan untuk memecah instruksi menjadi opcode dan operand secara paralel. Sementara itu, ALU menerapkan blok process untuk mengeksekusi instruksi matriks 3x3 secara bersamaan; dan RAM menerapkan blok process untuk memasukkan nilai ke dalam matriks yang akan melakukan operasi Aljabar Linear maupun menyimpan nilai ke dalam matriks yang menerima hasil operasi tersebut.

Sequential Circuit Design

Sequential circuit design merupakan rangkaian yang akan menginisiasi program ketika terjadi perubahan sinyal pada sensitivity list. Eksekusi dari desain ini dilakukan secara berurutan (satu per satu). Pada program Linear Algebra Matrix Solver, sequential design diterapkan menggunakan if statement yang terdapat pada komponen RAM untuk

mengeksekusi blok program sesuai kondisi RAM_WR. Selain itu, terdapat juga penerapan case statement pada (1) komponen ALU yang mengeksekusi blok program sesuai kondisi OPCODE dan (2) komponen Matrix Processor yang mengeksekusi blok program sesuai kondisi current_state.

Structural Circuit Design

Structural circuit design merupakan rangkaian yang bertugas menggabungkan seluruh komponen yang telah dibuat ke dalam satu komponen yang berfungsi untuk menghubungkan seluruh komponen tersebut menjadi satu rangkaian sirkuit digital yang kompleks. Pada program Linear Algebra Matrix Solver, structural design diterapkan di dalam komponen Matrix Processor yang mengatur alur dan proses input/output seluruh rangkaian.

Finite State Machine

Finite state machine merupakan penggambaran cara kerja dari rangkaian sekuensial dalam VHDL. FSM menerapkan blok process dan menggambarkan cara kerjanya menggunakan state-state yang berpindah sesuai case yang ditentukan. Pada program Linear Algebra Matrix Solver, FSM diterapkan dalam komponen Matrix Processor yang berfungsi mengendalikan alur program. Alur tersebut dimulai dari state IDLE yang tidak melakukan apapun sampai nilai ENABLE-nya berubah menjadi 1. Saat nilai ENABLE-nya berubah menjadi 1, FSM akan lanjut ke state FETCH yang berfungsi untuk menerima instruksi. Instruksi yang telah diterima akan masuk ke dalam state DECODE untuk dipecah menjadi opcode dan operand. Operand yang telah dipecah menjadi tiga bagian akan dilakukan pengisian. Operand yang bertugas untuk menjalankan operasi bersama dengan opcode akan diinisialisasi pada state READ_MEM. Setelah berhasil dieksekusi, hasilnya akan disimpan ke dalam operand yang bertugas untuk menyimpan hasil operasi pada state EXECUTE. Pada akhirnya, program akan lanjut ke state COMPLETE untuk memberitahukan bahwa seluruh proses telah selesai dan akan kembali ke state IDLE.

Microprogramming

Microprogramming merupakan sebuah teknik yang melibatkan penggunaan micro instruction set yang diimplementasikan menggunakan FSM. Pada program Linear Algebra Matrix Solver, microprogramming diterapkan di dalam komponen Matrix Processor bersamaan dengan penerapan FSM seperti yang telah dijelaskan sebelumnya.

Testbench

Testbench merupakan kode dalam VHDL yang bertujuan untuk memberikan input ke dalam entity komponen lain untuk menguji output dari suatu program. Untuk menerapkan testbench, bagian entity dari komponen ini dikosongkan dan digantikan dengan penggunaan port map pada bagian architecture. Pada program Linear Algebra Matrix Solver, testbench diterapkan dengan menguji komponen Matrix Processor yang bertindak sebagai CPU dari simple computer ini. Setelah membuat blok port map, komponen ini memasukkan input enable dan INSTRUCTION_IN untuk dilakukan testing supaya keakuratan program dapat teruji dengan benar.

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Dalam memastikan kebenaran dari fungsionalitas desain rancangan Matrix Processor, kami memutuskan untuk melakukan serangkaian pengujian dengan menggunakan simulasi pada ModelSim untuk instruksi berikut:

1. 000000100000000001

Instruksi ini akan menjalankan operasi **penjumlahan** matriks pada memori dengan alamat indeks ke-0 dan ke-1, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-2.

2. 001000110000000001

Instruksi ini akan menjalankan operasi **pengurangan** matriks pada memori dengan alamat indeks ke-0 dan ke-1, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-3.

3. 010001000000000000

Instruksi ini akan menjalankan operasi **pencerminan terhadap sumbu-x** untuk matriks pada memori dengan alamat indeks ke-0, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-4.

4. 011001010000000000

Instruksi ini akan menjalankan operasi **pencerminan terhadap sumbu-y** untuk matriks pada memori dengan alamat indeks ke-0, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-5.

5. 100001100000000000

Instruksi ini akan menjalankan operasi **pencerminan terhadap sumbu-z** untuk matriks pada memori dengan alamat indeks ke-0, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-6.

6. 1010011100000000000

Instruksi ini akan menjalankan operasi **transpose** untuk matriks pada memori dengan alamat indeks ke-0, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-7.

7. 1100100000000000000

Instruksi ini akan menjalankan operasi **kofaktor** untuk matriks pada memori dengan alamat indeks ke-0, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-8.

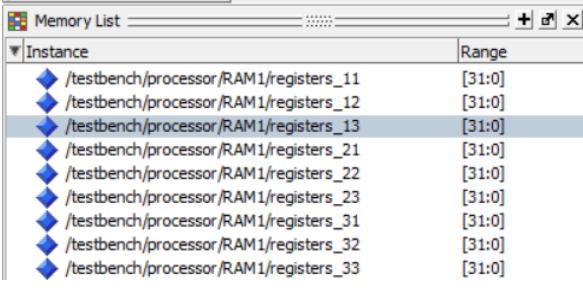
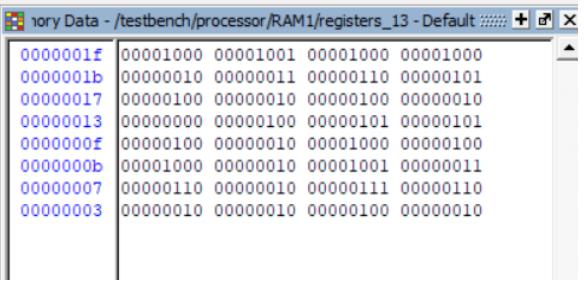
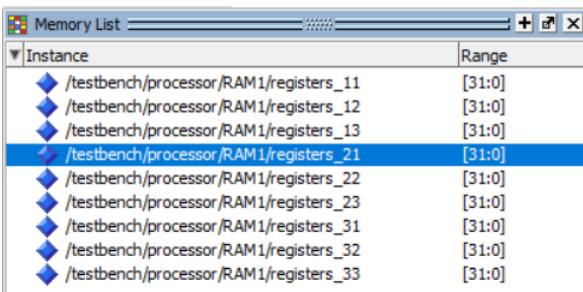
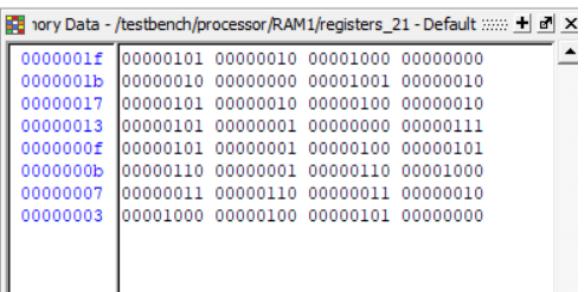
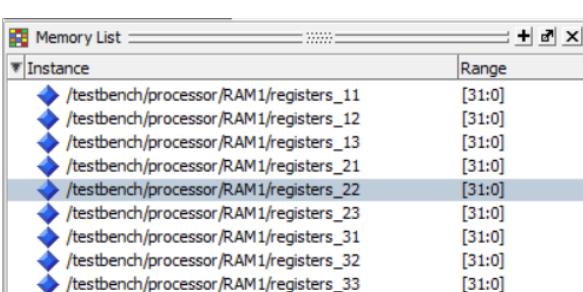
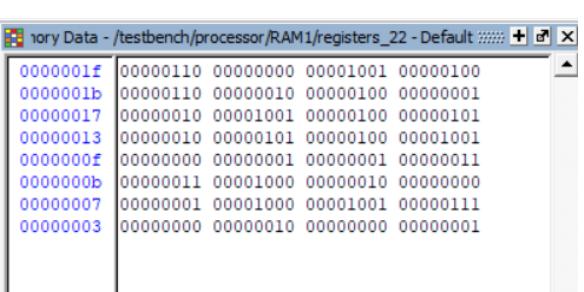
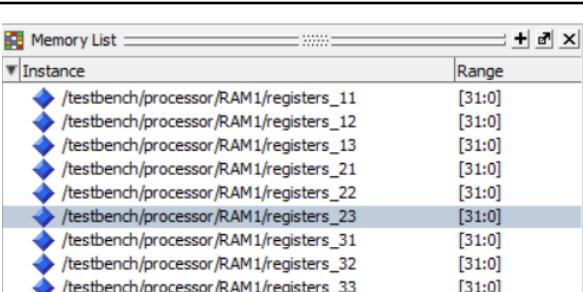
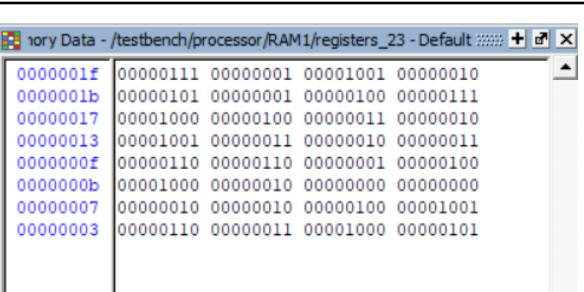
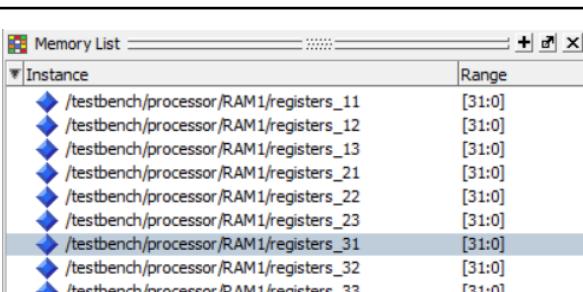
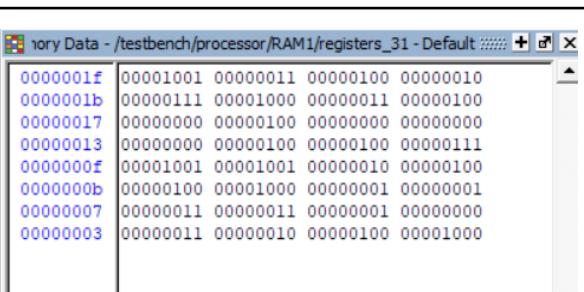
8. 1110100100000000001

Instruksi ini akan menjalankan operasi **perkalian** matriks pada memori dengan alamat indeks ke-0 dan ke-1, lalu menyimpan hasilnya pada memori dengan alamat indeks ke-9.

Perlu diperhatikan juga bahwasanya nilai awal untuk setiap elemen pada matriks di dalam memori adalah sebagai berikut:

Memory List		Memory Data - /testbench/processor/RAM1/registers_11 - Default	
Instance	Range	Address	Data
/testbench/processor/RAM1/registers_11	[31:0]	0000001f	00000101 00000100 00000001 00000100
/testbench/processor/RAM1/registers_12	[31:0]	0000001b	00000100 00001001 00000111 00000000
/testbench/processor/RAM1/registers_13	[31:0]	00000017	000000010 00000010 00000010 00000100
/testbench/processor/RAM1/registers_21	[31:0]	00000013	00000110 00000000 00000001 00000100
/testbench/processor/RAM1/registers_22	[31:0]	0000000f	00000011 00001000 00001000 00000101
/testbench/processor/RAM1/registers_23	[31:0]	0000000b	00000001 00000110 00000101 00000100
/testbench/processor/RAM1/registers_31	[31:0]	00000007	00000010 00000001 00001001 00000010
/testbench/processor/RAM1/registers_32	[31:0]	00000003	00000001 00000110 00000110 00000000
/testbench/processor/RAM1/registers_33	[31:0]		

Memory List		Memory Data - /testbench/processor/RAM1/registers_12 - Default	
Instance	Range	Address	Data
/testbench/processor/RAM1/registers_11	[31:0]	0000001f	00001001 00000000 00000101 00000111
/testbench/processor/RAM1/registers_12	[31:0]	0000001b	00000010 00000001 00000110 00000011
/testbench/processor/RAM1/registers_13	[31:0]	00000017	000000001 00000001 00000110 00000010
/testbench/processor/RAM1/registers_21	[31:0]	00000013	00000100 00000011 00000101 00000100
/testbench/processor/RAM1/registers_22	[31:0]	0000000f	00000111 00001001 00000101 00000100
/testbench/processor/RAM1/registers_23	[31:0]	0000000b	00000010 00001001 00000001 00000011
/testbench/processor/RAM1/registers_31	[31:0]	00000007	00000000 00000011 00000011 00000100
/testbench/processor/RAM1/registers_32	[31:0]	00000003	00000010 00000011 00001001 00000100
/testbench/processor/RAM1/registers_33	[31:0]		

The screenshot displays two memory dump windows side-by-side. The left window is titled "Memory List" and shows the range from 0 to 31. The right window is also titled "Memory List" and shows the range from 32 to 63. Both windows have a header with "Instance" and "Range". Below the header, there is a list of memory locations with their corresponding binary values.

Instance	Range
0000001f	00001001 00000011 00000000 00000000
0000001b	00001001 00000011 00000100 00000111
00000017	00000110 00001001 00000001 00000010
00000013	00000111 00000101 00000011 00000101
0000000f	00000011 00001001 00000001 00000111
0000000b	00000001 00000110 00000101 00000111
00000007	00000100 00000011 00000111 00001000
00000003	00000111 00000101 00000000 00000000

Instance	Range
0000001f	00000110 00000101 00001001 00000110
0000001b	00000101 00000101 00000011 00000110
00000017	00000010 00000100 00000000 00000000
00000013	00000000 00000001 00000100 00000101
0000000f	00000001 00001000 00000000 00000111
0000000b	00000010 00001001 00000010 00000101
00000007	00001001 00000101 00000111 00000000
00000003	00000010 00000001 00000001 00000000

Adapun nilai desimal dari matriks-matriks yang terlibat untuk tahap uji kebenaran desain, yakni sebagai berikut:

1. Matriks[0]

0	4	2
0	1	5
8	0	0

2. Matriks[1]

6	9	4
5	0	8
4	0	1

3. Matriks[2]

6	3	2
4	2	3
2	5	1

4. Matriks[3]

1	2	2
8	0	6
3	7	2

5. Matriks[4]

2	9	3
2	7	9
0	8	0

6. Matriks[5]

9	3	7
3	9	4
1	7	7

7. Matriks[6]

1	3	2
6	8	2
3	3	5

8. Matriks[7]

2	0	6
6	1	2
3	7	2

9. Matriks[8]

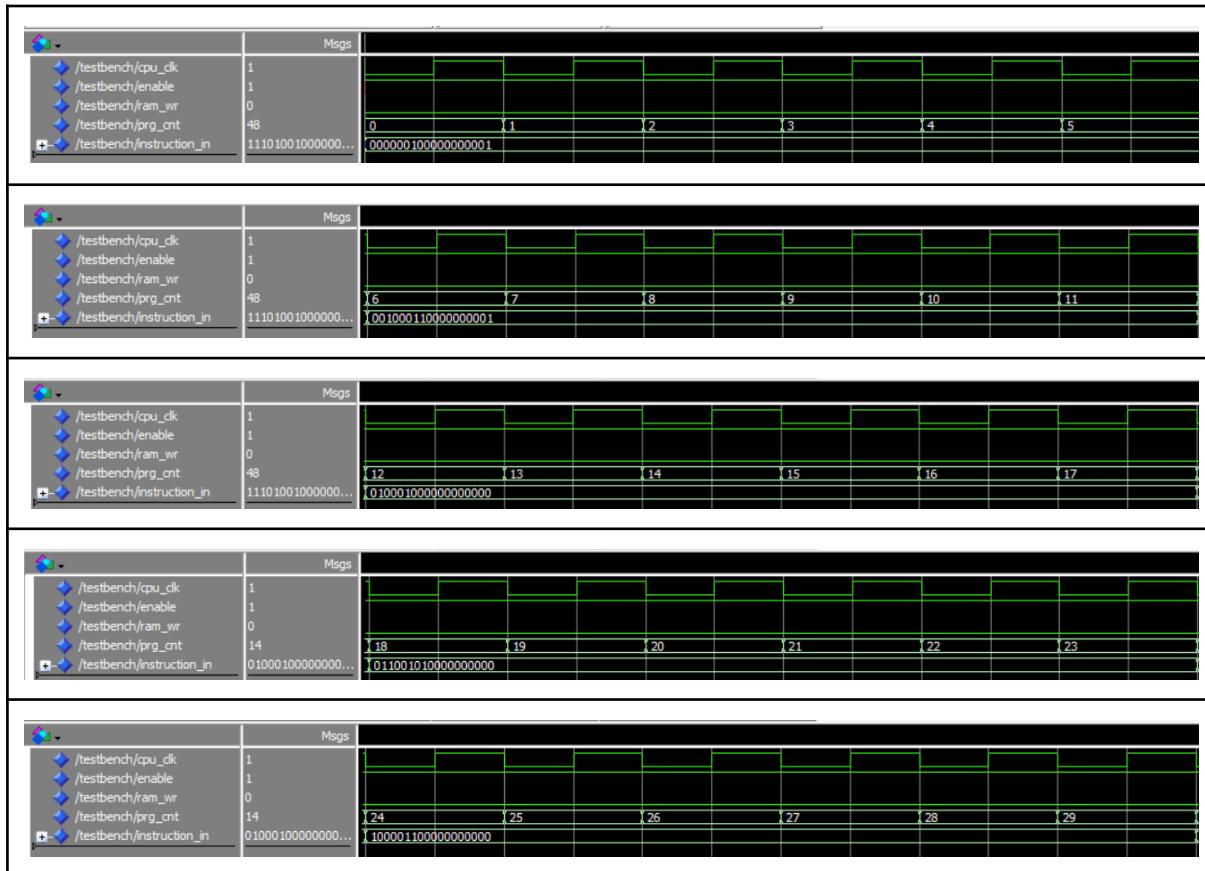
4	3	3
8	0	0
1	7	5

10. Matriks[9]

5	1	9
6	2	0
1	5	2

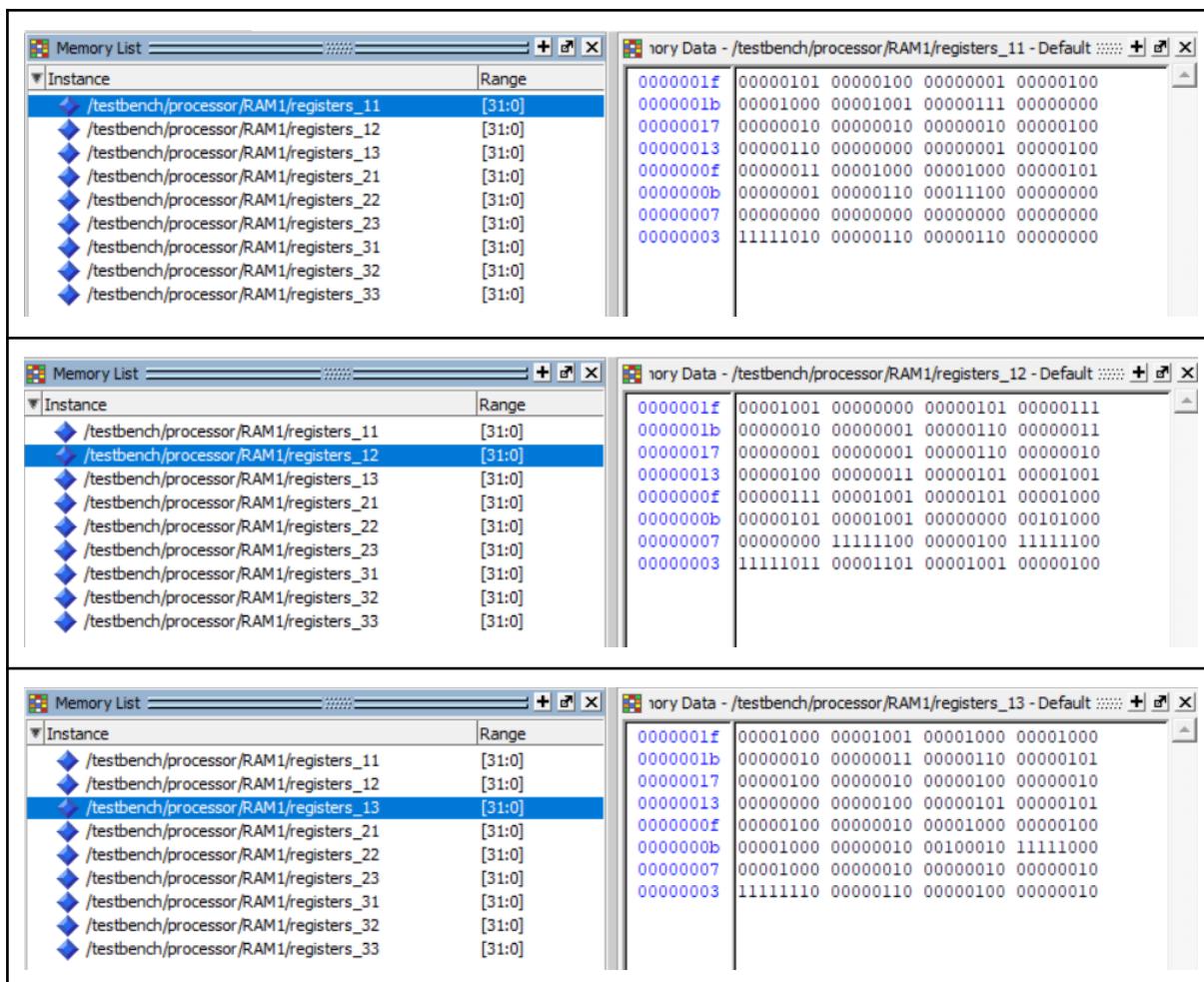
3.2 RESULT

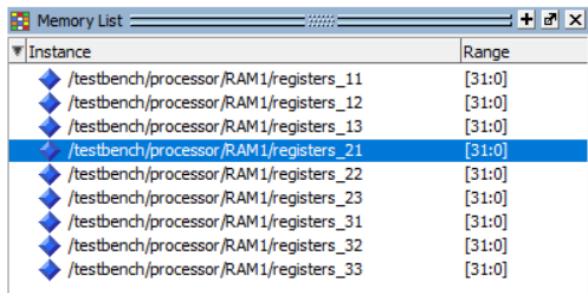
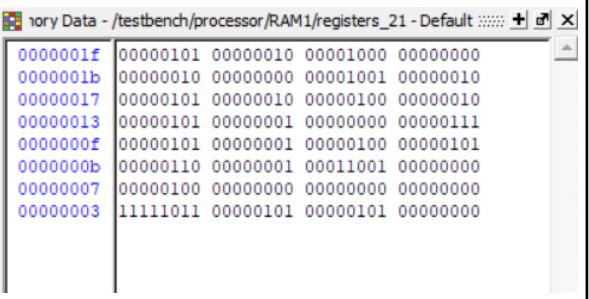
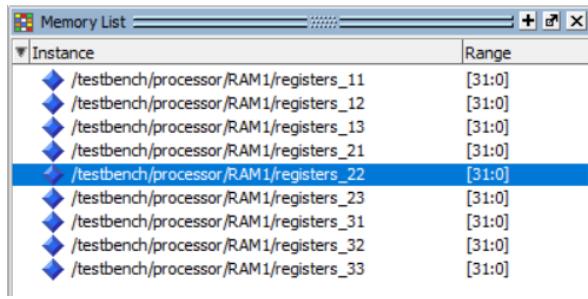
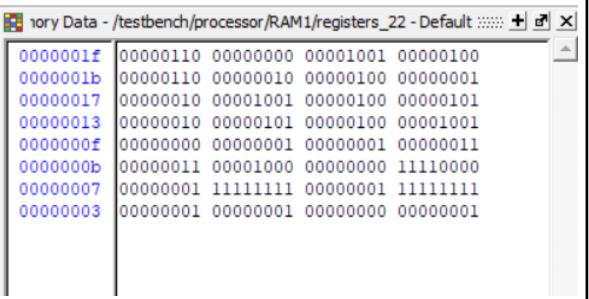
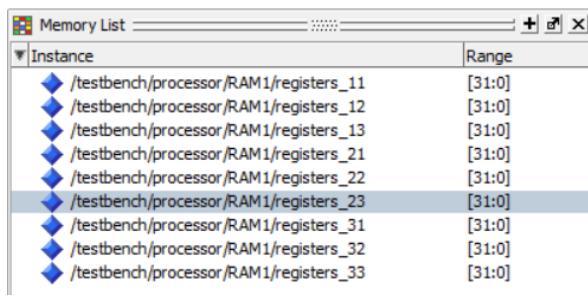
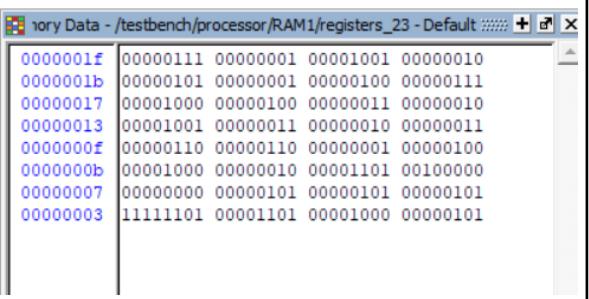
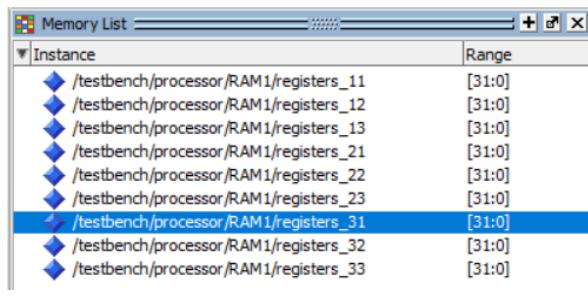
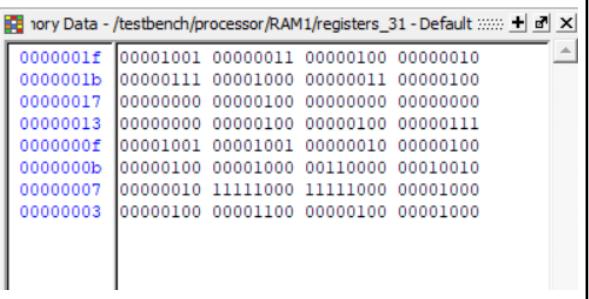
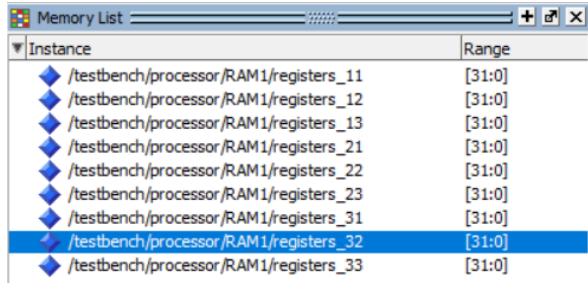
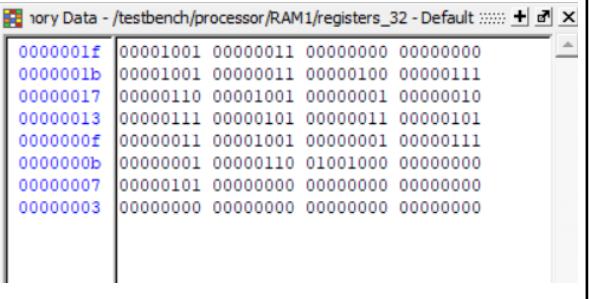
Setelah menjalankan uji simulasi menggunakan kode testbench yang telah dirancang, maka akan didapatkan hasil waveform sebagai berikut:





Adapun hasil terbaru untuk seluruh matriks pada memori, yakni sebagai berikut:



The screenshot shows two windows side-by-side. The left window is titled 'Memory List' and contains a table with columns 'Instance' and 'Range'. It lists 10 memory instances from /testbench/processor/RAM1/registers_11 to /testbench/processor/RAM1/registers_33, each with a range of [31:0]. The right window is titled 'Memory Data - /testbench/processor/RAM1/registers_33 - Default' and displays a table of 16 rows of binary data. The first few rows are: 0000001f, 00000110; 000001b, 00000101; 00000017, 00000010; 00000013, 00000000; 000000f, 00000001; 0000000b, 00000100; 00000007, 00000101; and 00000003, 00000011. The last row is 11111111, 00000001.

Instance	Range
/testbench/processor/RAM1/registers_11	[31:0]
/testbench/processor/RAM1/registers_12	[31:0]
/testbench/processor/RAM1/registers_13	[31:0]
/testbench/processor/RAM1/registers_21	[31:0]
/testbench/processor/RAM1/registers_22	[31:0]
/testbench/processor/RAM1/registers_23	[31:0]
/testbench/processor/RAM1/registers_31	[31:0]
/testbench/processor/RAM1/registers_32	[31:0]
/testbench/processor/RAM1/registers_33	[31:0]

0000001f	00000110	00000101	000001001	00000110
0000001b	00000101	00000101	00000011	00000110
00000017	00000010	00000100	00000000	00000000
00000013	00000000	00000001	00000100	00000101
000000f	00000001	00001000	00000000	00000111
0000000b	00000101	00001001	00100000	00000000
00000007	00000000	00000000	00000000	00000000
00000003	11111111	00000001	00000001	00000000

Untuk 10 matriks yang terlibat dalam uji kebenaran desain, didapatkan nilai desimal sebagai berikut:

1. Matriks[0]

0	4	2
0	1	5
8	0	0

2. Matriks[1]

6	9	4
5	0	8
4	0	1

3. Matriks[2]

6	13	6
5	1	13
12	0	1

4. Matriks[3]

-6	-5	-2
-5	1	-3
4	0	-1

5. Matriks[4]

0	-4	2
0	-1	5
8	0	0

6. Matriks[5]

0	4	2
0	1	5
-8	0	0

7. Matriks[6]

0	-4	2
0	-1	5
-8	0	0

8. Matriks[7]

0	0	8
4	1	0
2	5	0

9. Matriks[8]

0	40	-8
0	-16	32
18	0	0

10. Matriks[9]

28	0	34
25	0	13
48	72	32

3.3 ANALYSIS

Dengan membandingkan matriks pada memori untuk sebelum dan sesudah dijalankan testbench, maka akan didapatkan beberapa informasi berupa:

1. Matriks[2] akan berisikan hasil penjumlahan dari Matriks[0] dan Matriks[1]
2. Matriks[3] akan berisikan hasil pengurangan dari Matriks[0] dan Matriks[1]
3. Matriks[4] akan berisikan hasil pencerminan Matriks[0] terhadap sumbu-x
4. Matriks[5] akan berisikan hasil pencerminan Matriks[0] terhadap sumbu-y
5. Matriks[6] akan berisikan hasil pencerminan Matriks[0] terhadap sumbu-z
6. Matriks[7] akan berisikan hasil transpose dari Matriks[0]
7. Matriks[8] akan berisikan kofaktor dari Matriks[0]
8. Matriks[9] akan berisikan hasil perkalian dari Matriks[0] dan Matriks[1]

Hasil-hasil yang didapatkan sudah dicek ulang menggunakan alat yang dapat diandalkan untuk mengecek kebenarannya, dan didapatkan bahwasanya Matrix Processor yang kami rancang sudah memiliki fungsionalitas sesuai dengan harapan kami.

CHAPTER 4

CONCLUSION

Kesimpulannya, kami mencoba untuk menyelesaikan masalah komputasional matriks secara efisien dan cepat, dimana kebanyakan prosesor sekarang tidak didesain untuk menghitung operasi matriks dengan cepat. Dengan adanya “Linear Algebra Matrix Solver” ini kami berharap agar komputasi matriks pada sistem komputer yang kedepannya akan banyak sekali digunakan pada berbagai aplikasi, seperti pengolahan grafis, pelatihan model kecerdasan buatan, dan mempercepat kecepatan eksekusi permasalahan matriks dengan adanya solusi yang kami tawarkan. Idealnya solusi kami ini digunakan sebagai sebuah akselerator dari prosesor utama sebagai unit komputasi yang dikhkususkan untuk komputasi matriks.

REFERENCES

- [1]H. Anton and C. Rorres, *Elementary Linear Algebra*. 2014.
- [2]M. R. Mano, C. R. Kime, and T. Martin, *Logic & Computer Design Fundamentals*. Pearson, 2015.
- [3]S. D. Brown and Z. G. Vranesic, *Fundamentals of digital logic with VHDL design*. Chennai: McGraw-Hill Education (India) Private Limited, 2012.

APPENDICES

Appendix A: Project Schematic

MatrixProcessor

Decoder

ALU

RAM

Appendix B: Documentation

