

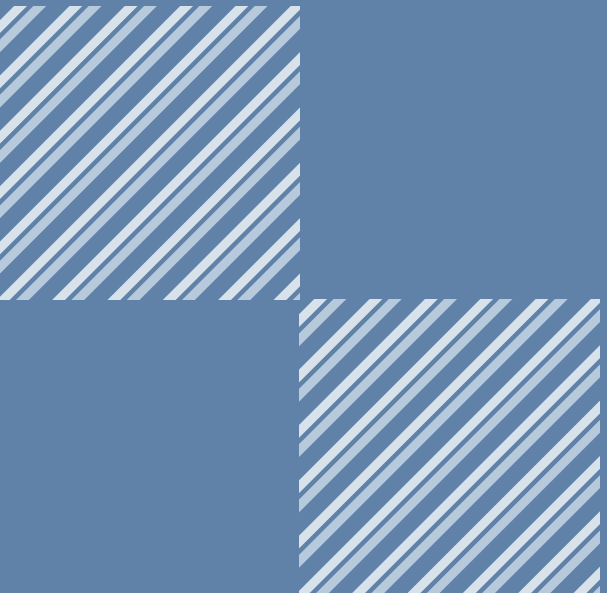
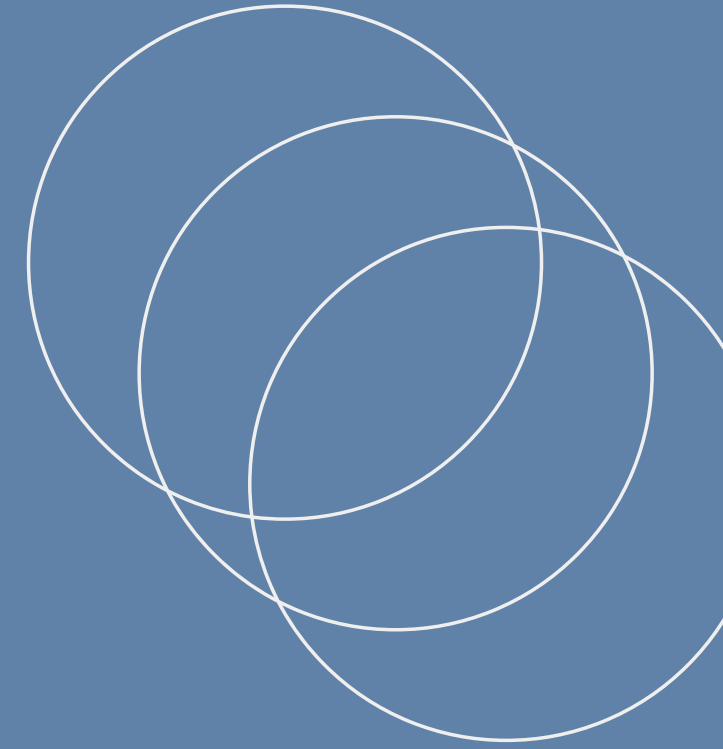
PROYEK AKHIR PSD

LINEAR ALGEBRA MATRIX SOLVER

PROGRAM STUDI TEKNIK KOMPUTER

2023

Anggota Kelompok



Anggota Kelompok



Evandita Wiratama Putra
2206059572

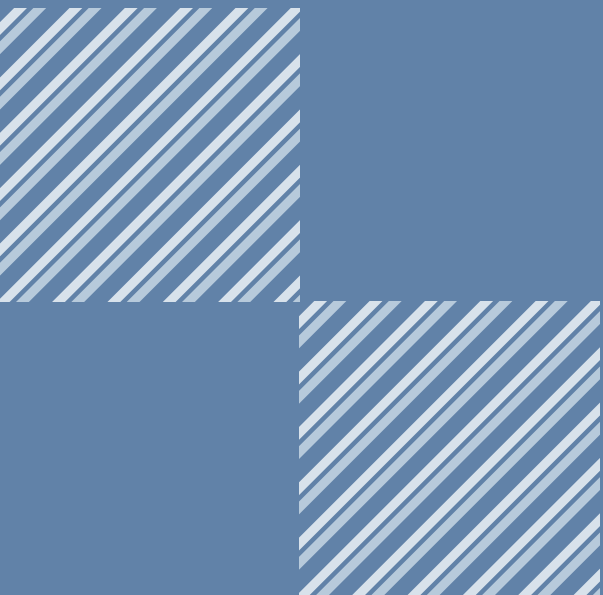
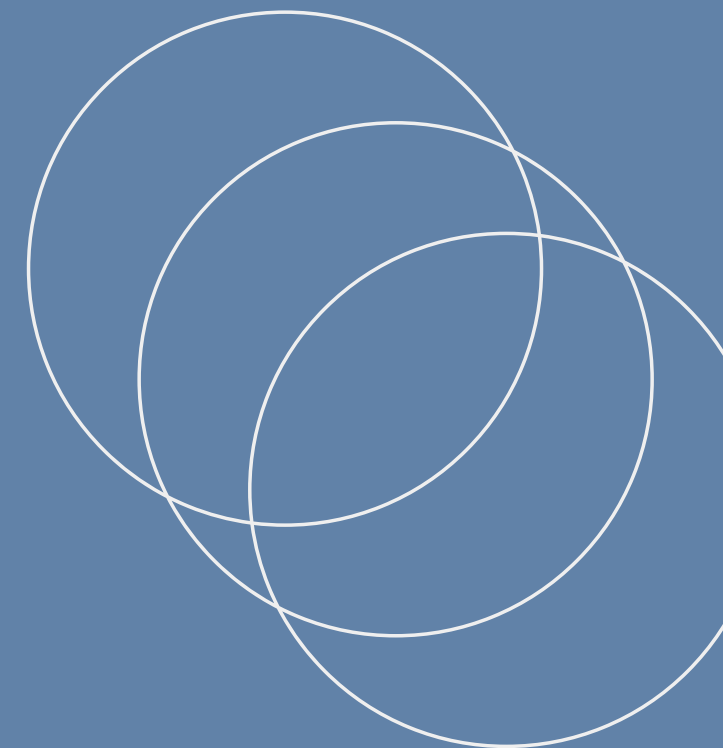
Hanif Nur Ilham Sanjaya
2206059692

Ivander Andreas Wijaya
2206031896

Monica Vierin Pasman
2206029405



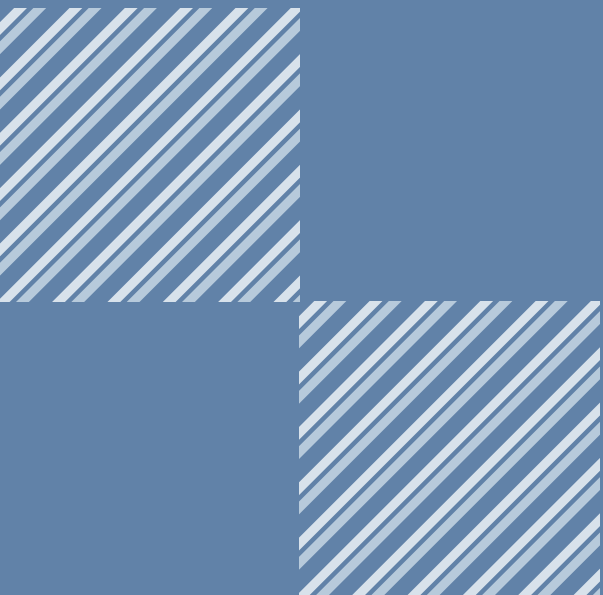
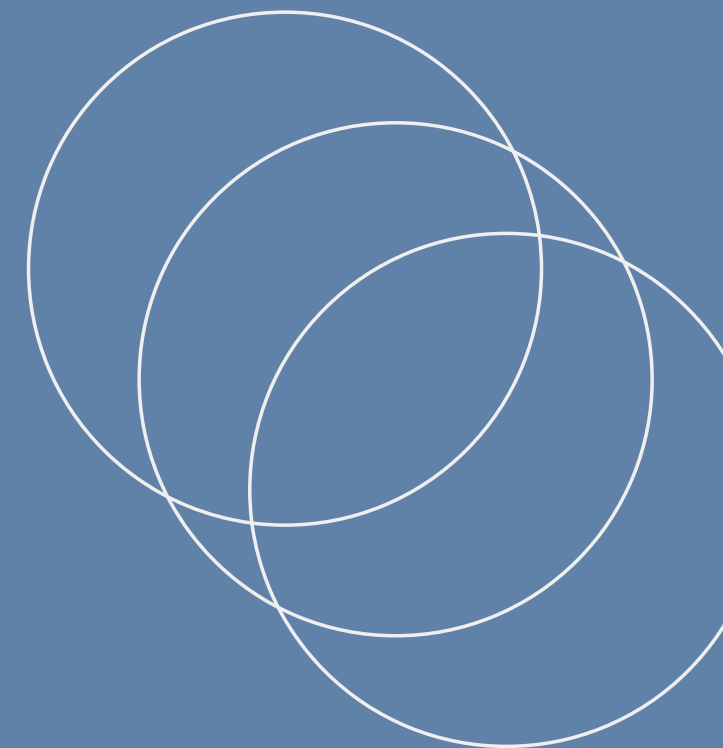
Latar Belakang



Latar Belakang

Penyelesaian permasalahan matematika sering digunakan dalam implementasi rangkaian digital. Maka dari itu, untuk memenuhi kebutuhan tersebut sekaligus mempercepat kinerja processor kami membuat 'Linear Algebra Matrix Solver' yang berfungsi menyelesaikan berbagai perhitungan matriks 3x3. Program ini mengimplementasikan VHDL agar nantinya dapat disintesis dan diimplementasikan pada perangkat FPGA.

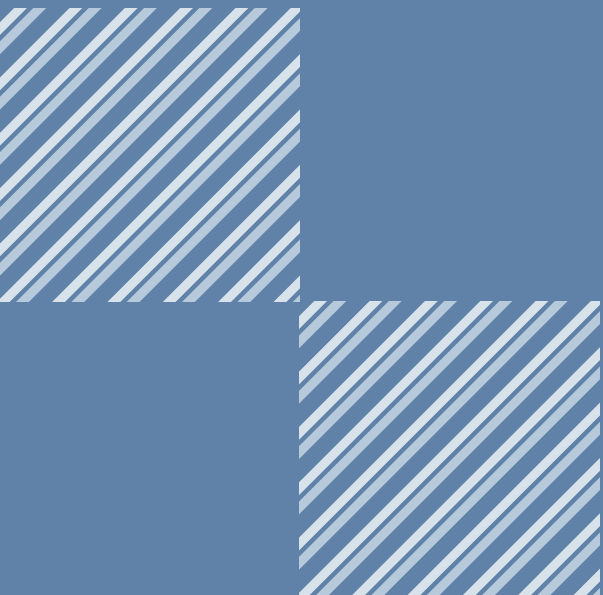
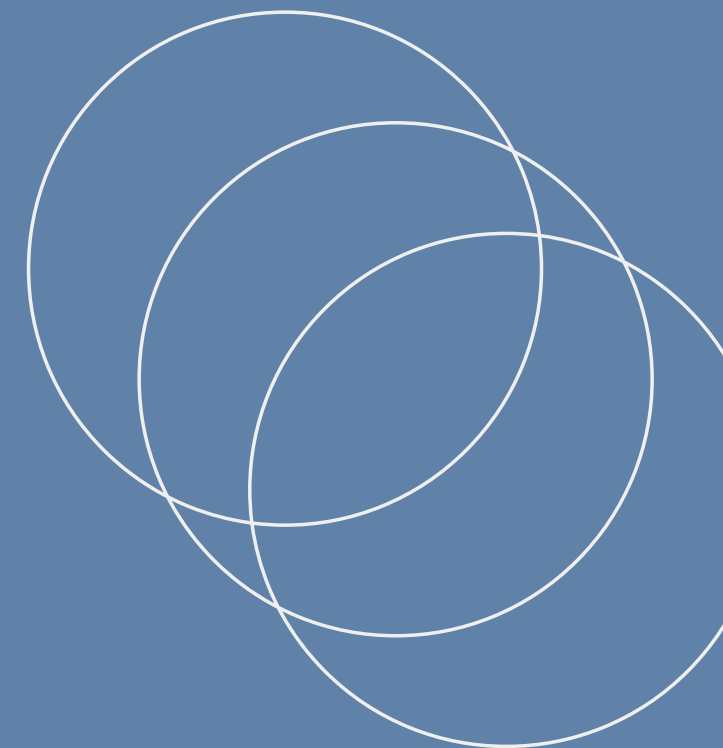
Deskripsi Proyek



Deskripsi Proyek

Proyek ini melibatkan perancangan sebuah komputer sederhana menggunakan VHDL untuk menyelesaikan persamaan matriks 3×3 . Sistem ini terdiri dari komponen-komponen kunci: sebuah MatrixProcessor yang mengoordinasikan semua operasi, sebuah ALU untuk melakukan operasi matriks, sebuah DECODER untuk mendekode instruksi, dan sebuah RAM untuk manajemen memori. Seluruh proses dikontrol oleh Finite State Machine yang diimplementasikan di MatrixProcessor, memastikan setiap langkah dilakukan dalam urutan yang benar. Desain ini memungkinkan pemrosesan matriks secara efisien dan paralel, menjadikannya cocok untuk aplikasi yang membutuhkan komputasi matriks berkinerja tinggi.

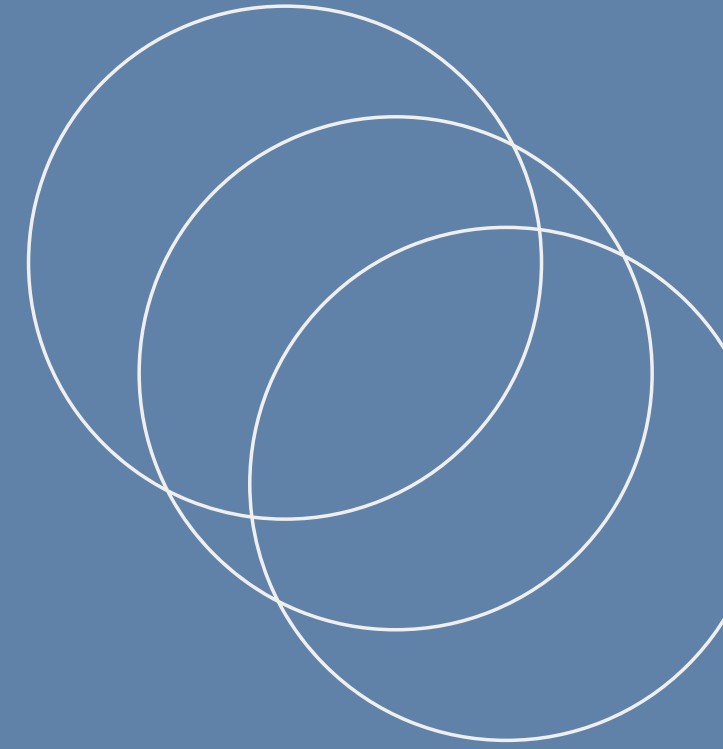
Tujuan

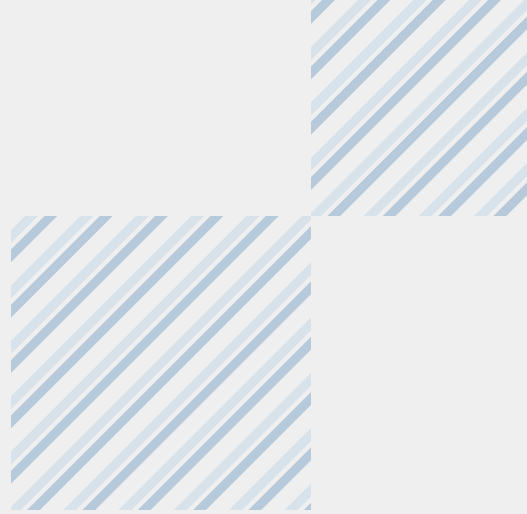


Tujuan

01. Merancang komponen-komponen penting simple computer (MatrixProcessor, ALU, DECODER, RAM) menggunakan VHDL.
02. Mengimplementasikan operasi matriks pada ALU.
03. Mengontrol alur jalannya program dengan FSM.
04. Memverifikasi fungsionalitas melalui simulasi dan mengimplementasikan desain pada perangkat FPGA.
05. Mendemonstrasikan kemampuan VHDL dalam merancang sistem digital kompleks.

Equipment





Equipment



VSCode
Code editor



ModelSim
Simulator

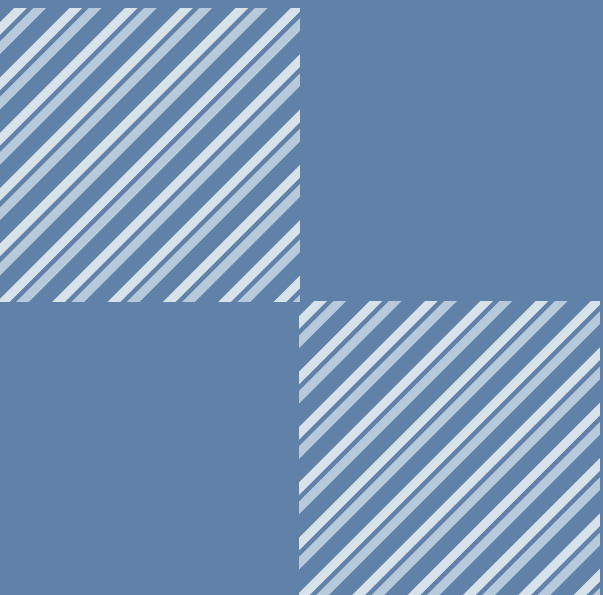
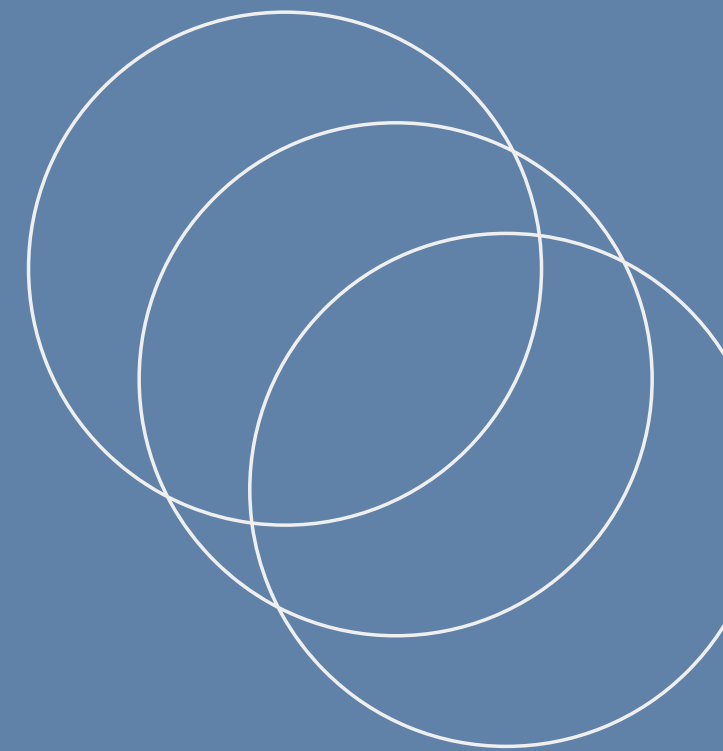


**Intel Quartus
Prime**
Analisis dan sintesis



GitHub
Manajemen proyek

Implementation



MatrixProcessor

```
CASE current_state IS
  WHEN IDLE =>
    -- When idle state, wait for enable to be '1'
    IF ENABLE = '1' THEN
      next_state <= FETCH;
      Program_Counter <= Program_Counter + 1;
    ELSE
      next_state <= IDLE;
    END IF;

  WHEN FETCH =>
    -- When fetch, receive instruction input
    INSTRUCTION_s <= INSTRUCTION_IN;
    Program_Counter <= Program_Counter + 1;
    next_state <= DECODE;

  WHEN DECODE =>
    -- When decode, pass arguments to decoder component
    Program_Counter <= Program_Counter + 1;
    next_state <= READ_MEM;

  WHEN READ_MEM =>
    -- When read memory, read from RAM and store as operands
    RAM_ADDR_A_s <= OP_ADDR2_s;
    RAM_WR_s <= '0';
    next_state <= EXECUTE;
    Program_Counter <= Program_Counter + 1;

  WHEN EXECUTE =>
    -- When execute, pass arguments to ALU component, then write back to RAM
    RAM_WR_s <= '1';
    RAM_ADDR_A_s <= OP_ADDR1_s;
    next_state <= COMPLETE;
    Program_Counter <= Program_Counter + 1;

  WHEN COMPLETE =>
    -- When complete, print report statement to notify instruction complete
    REPORT "Instruction complete at Program Counter " & INTEGER'image(program_counter);
    next_state <= IDLE;
    Program_Counter <= 0;

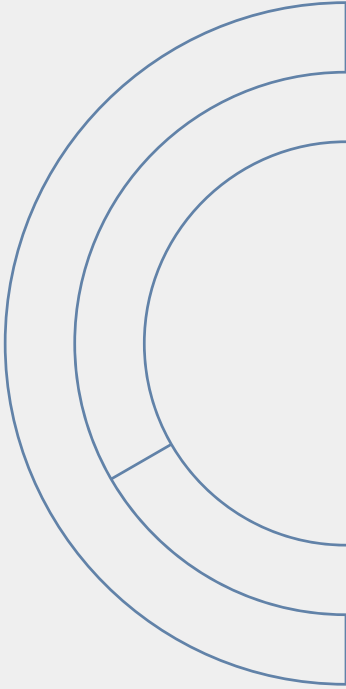
  WHEN OTHERS =>
    next_state <= IDLE;
END CASE;
```

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.

Decoder

```
INSTRUCTION <= INSTRUCTION_IN;  
OPCODE <= INSTRUCTION(17 DOWNT0 15); -- Function select  
OP1_ADDR <= INSTRUCTION(14 DOWNT0 10); -- Register DA  
OP2_ADDR <= INSTRUCTION(9 DOWNT0 5); -- Register AA  
OP3_ADDR <= INSTRUCTION(4 DOWNT0 0); -- Register BA
```

Komponen ini menerima ISA 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

```
if RAM_WR = '1' then
    registers_11(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_11;
    registers_12(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_12;
    registers_13(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_13;
    registers_21(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_21;
    registers_22(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_22;
    registers_23(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_23;
    registers_31(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_31;
    registers_32(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_32;
    registers_33(to_integer(unsigned(RAM_ADDR_A))) <= RAM_MATRIX_IN_33;
    -- Ketika sinyal enable bernilai '0', maka mengeluarkan 2 buah nilai matriks
else
    RAM_MATRIX_OUT_11_A <= registers_11(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_12_A <= registers_12(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_13_A <= registers_13(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_21_A <= registers_21(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_22_A <= registers_22(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_23_A <= registers_23(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_31_A <= registers_31(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_32_A <= registers_32(to_integer(unsigned(RAM_ADDR_A)));
    RAM_MATRIX_OUT_33_A <= registers_33(to_integer(unsigned(RAM_ADDR_A)));

    RAM_MATRIX_OUT_11_B <= registers_11(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_12_B <= registers_12(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_13_B <= registers_13(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_21_B <= registers_21(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_22_B <= registers_22(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_23_B <= registers_23(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_31_B <= registers_31(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_32_B <= registers_32(to_integer(unsigned(RAM_ADDR_B)));
    RAM_MATRIX_OUT_33_B <= registers_33(to_integer(unsigned(RAM_ADDR_B)));
end if;
```

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.

ALU

```
INSTRUCTION <= INSTRUCTION_IN;  
OPCODE <= INSTRUCTION(17 DOWNTO 15); -- Function select  
OP1_ADDR <= INSTRUCTION(14 DOWNTO 10); -- Register DA  
OP2_ADDR <= INSTRUCTION(9 DOWNTO 5); -- Register AA  
OP3_ADDR <= INSTRUCTION(4 DOWNTO 0); -- Register BA
```

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.

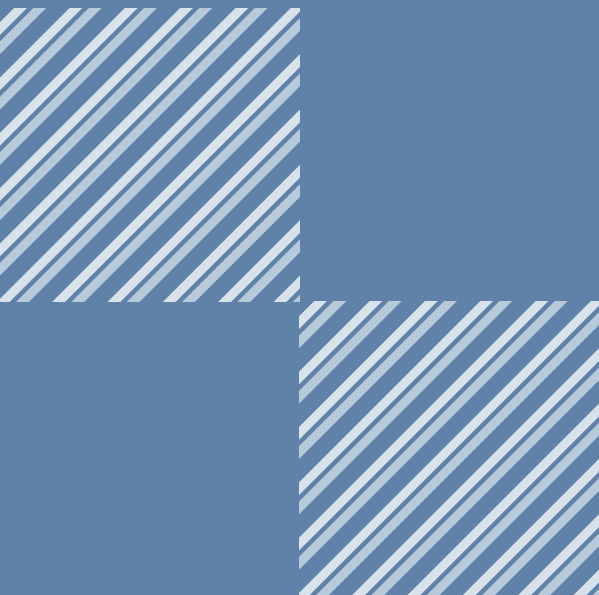
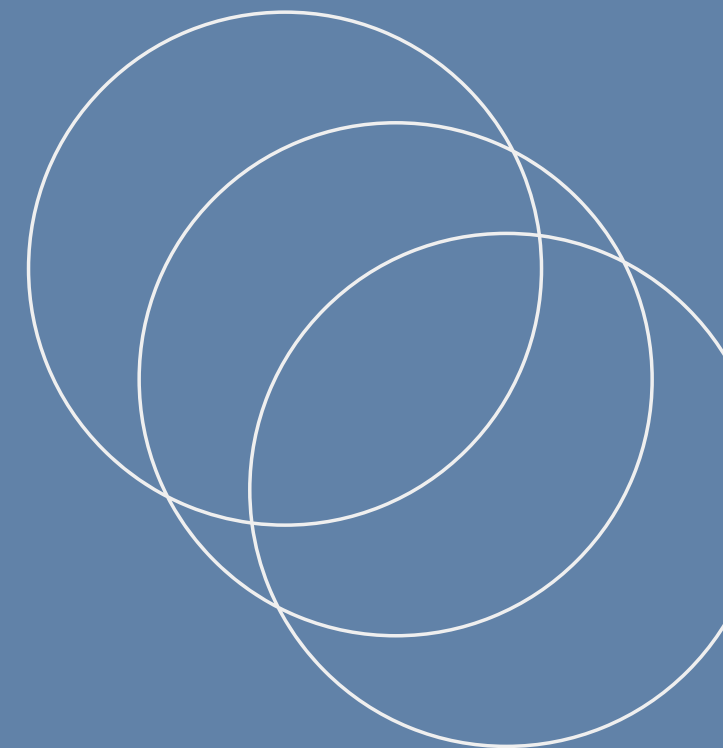
TestBench

```
enable <= '1';  
INSTRUCTION_IN <= "000000100000000001";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "001000110000000001";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "010001000000000000";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "011001010000000000";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "100001100000000000";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "101001110000000000";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "110010000000000000";  
WAIT FOR period;  
  
INSTRUCTION_IN <= "111010010000000001";  
WAIT;
```

Program ini digunakan untuk memverifikasi fungsionalitas dari MatrixProcessor. Di sini terdapat clock dan memberikan serangkaian instruksi (ISA) kepada MatrixProcessor, mensimulasikan bagaimana sistem akan beroperasi. Hasilnya kemudian dapat diamati untuk memastikan bahwa sistem berfungsi seperti yang diharapkan.



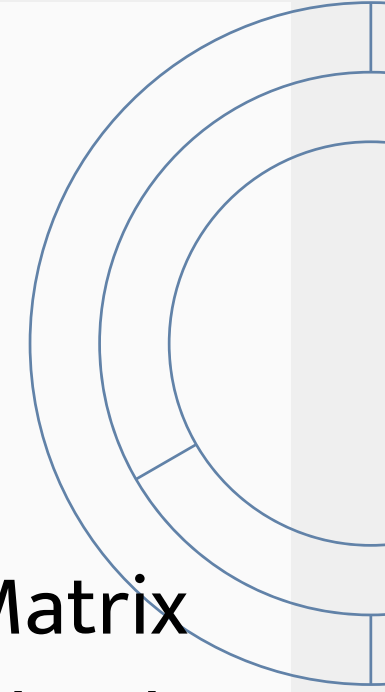
Simulasi



Simulasi



Untuk menguji kebenaran daripada desain Matrix Processor yang telah kami rancang, maka sebuah simulasi menggunakan testbench diperlukan untuk memastikan proses perhitungan untuk operasi pada matriks telah sesuai dengan harapan kami.



Simulasi



Kami akan melakukan testing untuk setiap instruksi yang tersedia processor, yakni:

1. Penjumlahan Matriks
 2. Pengurangan Matriks
 3. Pencerminkan Matriks terhadap sumbu-x
 4. Pencerminkan Matriks terhadap sumbu-y
 5. Pencerminkan Matriks terhadap sumbu-z
 6. Transpose Matriks
 7. Kofaktor Matriks
 8. Perkalian Matriks
- 

Simulasi

Untuk melakukan operasi yang tersedia pada instruksi tersebut, kami akan menggunakan 2 matriks yang terdapat pada ram, yakni matriks[0] dan matriks[1], dimana:

$$\text{Matriks}[0] = \begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix}$$

$$\text{Matriks}[1] = \begin{bmatrix} 6 & 9 & 4 \\ 5 & 0 & 8 \\ 4 & 0 & 1 \end{bmatrix}$$

Simulasi

Penjumlahan Matriks[0] dan Matriks[1]

+ ◆	/matrixprocessor/RAM_MATRIX_IN_11_s	00000110
+ ◆	/matrixprocessor/RAM_MATRIX_IN_12_s	00001101
+ ◆	/matrixprocessor/RAM_MATRIX_IN_13_s	00000110
+ ◆	/matrixprocessor/RAM_MATRIX_IN_21_s	00000101
+ ◆	/matrixprocessor/RAM_MATRIX_IN_22_s	00000001
+ ◆	/matrixprocessor/RAM_MATRIX_IN_23_s	00001101
+ ◆	/matrixprocessor/RAM_MATRIX_IN_31_s	00001100
+ ◆	/matrixprocessor/RAM_MATRIX_IN_32_s	00000000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_33_s	00000001
+ ◆	/matrixprocessor/DETER_MAT_A_s	00010000
+ ◆	/matrixprocessor/DETER_MAT_B_s	01110011
+ ◆	/matrixprocessor/DETER_MAT_D_s	01101001
◆	/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 6 & 9 & 4 \\ 5 & 0 & 8 \\ 4 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 6 & 13 & 6 \\ 5 & 1 & 13 \\ 12 & 0 & 1 \end{bmatrix}$$

Simulasi

Pengurangan Matriks[0] dan Matriks[1]

+ /matrixprocessor/RAM_MATRIX_IN_11_s	11111010
+ /matrixprocessor/RAM_MATRIX_IN_12_s	11111011
+ /matrixprocessor/RAM_MATRIX_IN_13_s	11111110
+ /matrixprocessor/RAM_MATRIX_IN_21_s	11111011
+ /matrixprocessor/RAM_MATRIX_IN_22_s	00000001
+ /matrixprocessor/RAM_MATRIX_IN_23_s	11111101
+ /matrixprocessor/RAM_MATRIX_IN_31_s	00000100
+ /matrixprocessor/RAM_MATRIX_IN_32_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_33_s	11111111
+ /matrixprocessor/DETER_MAT_A_s	00010000
+ /matrixprocessor/DETER_MAT_B_s	01110011
+ /matrixprocessor/DETER_MAT_D_s	01100011
/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 6 & 9 & 4 \\ 5 & 0 & 8 \\ 4 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -6 & -5 & -2 \\ -5 & 1 & -3 \\ 4 & 0 & -1 \end{bmatrix}$$

Simulasi

Pencerminan Matriks[0] terhadap sumbu-x

+ /matrixprocessor/RAM_MATRIX_IN_11_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_12_s	11111100
+ /matrixprocessor/RAM_MATRIX_IN_13_s	00000010
+ /matrixprocessor/RAM_MATRIX_IN_21_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_22_s	11111111
+ /matrixprocessor/RAM_MATRIX_IN_23_s	00000101
+ /matrixprocessor/RAM_MATRIX_IN_31_s	00001000
+ /matrixprocessor/RAM_MATRIX_IN_32_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_33_s	00000000
+ /matrixprocessor/DETER_MAT_A_s	00010000
+ /matrixprocessor/DETER_MAT_B_s	01110011
+ /matrixprocessor/DETER_MAT_D_s	11110000
/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -4 & 2 \\ 0 & -1 & 5 \\ 8 & 0 & 0 \end{bmatrix}$$

Simulasi

Pencerminan Matriks[0] terhadap sumbu-y

+ /matrixprocessor/RAM_MATRIX_IN_11_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_12_s	00000100
+ /matrixprocessor/RAM_MATRIX_IN_13_s	00000010
+ /matrixprocessor/RAM_MATRIX_IN_21_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_22_s	00000001
+ /matrixprocessor/RAM_MATRIX_IN_23_s	00000101
+ /matrixprocessor/RAM_MATRIX_IN_31_s	11111000
+ /matrixprocessor/RAM_MATRIX_IN_32_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_33_s	00000000
+ /matrixprocessor/DETER_MAT_A_s	00010000
+ /matrixprocessor/DETER_MAT_B_s	01110011
+ /matrixprocessor/DETER_MAT_D_s	11110000
/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ -8 & 0 & 0 \end{bmatrix}$$

Simulasi

Pencerminan Matriks[0] terhadap sumbu-z

+ /matrixprocessor/RAM_MATRIX_IN_11_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_12_s	11111100
+ /matrixprocessor/RAM_MATRIX_IN_13_s	00000010
+ /matrixprocessor/RAM_MATRIX_IN_21_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_22_s	11111111
+ /matrixprocessor/RAM_MATRIX_IN_23_s	00000101
+ /matrixprocessor/RAM_MATRIX_IN_31_s	11111000
+ /matrixprocessor/RAM_MATRIX_IN_32_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_33_s	00000000
+ /matrixprocessor/DETER_MAT_A_s	00010000
+ /matrixprocessor/DETER_MAT_B_s	01110011
+ /matrixprocessor/DETER_MAT_D_s	00010000
/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -4 & 2 \\ 0 & -1 & 5 \\ -8 & 0 & 0 \end{bmatrix}$$

Simulasi

Transpose Matriks[0]

+ /matrixprocessor/RAM_MATRIX_IN_11_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_12_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_13_s	00001000
+ /matrixprocessor/RAM_MATRIX_IN_21_s	00000100
+ /matrixprocessor/RAM_MATRIX_IN_22_s	00000001
+ /matrixprocessor/RAM_MATRIX_IN_23_s	00000000
+ /matrixprocessor/RAM_MATRIX_IN_31_s	00000010
+ /matrixprocessor/RAM_MATRIX_IN_32_s	00000101
+ /matrixprocessor/RAM_MATRIX_IN_33_s	00000000
+ /matrixprocessor/DETER_MAT_A_s	00010000
+ /matrixprocessor/DETER_MAT_B_s	01110011
+ /matrixprocessor/DETER_MAT_D_s	00010000
/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 0 & 8 \\ 4 & 1 & 0 \\ 2 & 5 & 0 \end{bmatrix}$$

Simulasi

Kofaktor Matriks[0]

+ ◆	/matrixprocessor/RAM_MATRIX_IN_11_s	00000000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_12_s	00101000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_13_s	11111000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_21_s	00000000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_22_s	11110000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_23_s	00100000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_31_s	00010010
+ ◆	/matrixprocessor/RAM_MATRIX_IN_32_s	00000000
+ ◆	/matrixprocessor/RAM_MATRIX_IN_33_s	00000000
+ ◆	/matrixprocessor/DETER_MAT_A_s	00010000
+ ◆	/matrixprocessor/DETER_MAT_B_s	01110011
+ ◆	/matrixprocessor/DETER_MAT_D_s	00000000
◆	/matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 40 & -8 \\ 0 & -16 & 32 \\ 18 & 0 & 0 \end{bmatrix}$$

Simulasi

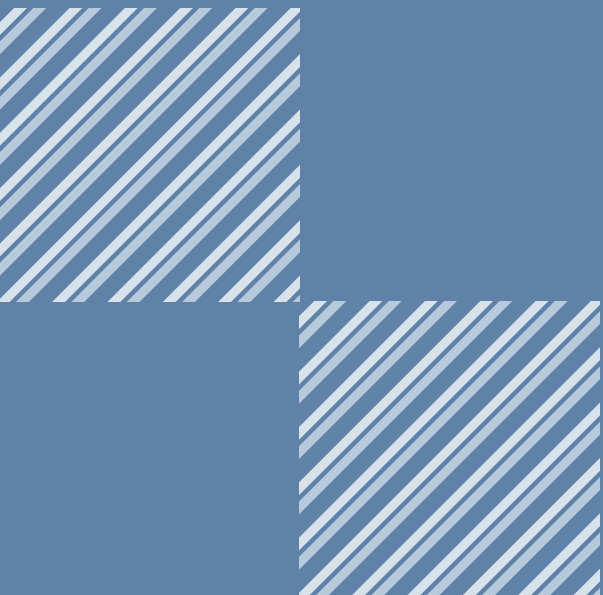
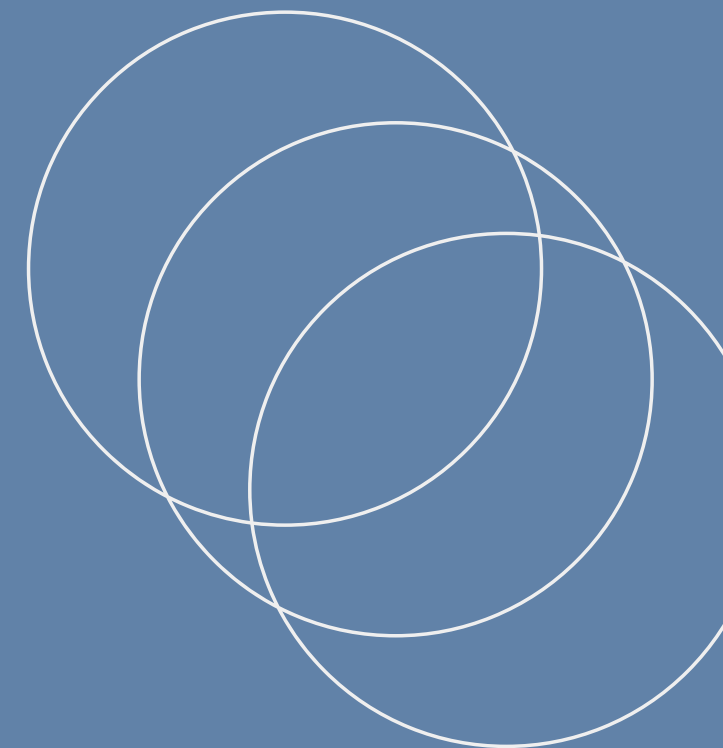
Perkalian Matriks[0] dan Matriks[1]

+ ◆ /matrixprocessor/RAM_MATRIX_IN_11_s	00011100
+ ◆ /matrixprocessor/RAM_MATRIX_IN_12_s	00000000
+ ◆ /matrixprocessor/RAM_MATRIX_IN_13_s	00100010
+ ◆ /matrixprocessor/RAM_MATRIX_IN_21_s	00011001
+ ◆ /matrixprocessor/RAM_MATRIX_IN_22_s	00000000
+ ◆ /matrixprocessor/RAM_MATRIX_IN_23_s	00001101
+ ◆ /matrixprocessor/RAM_MATRIX_IN_31_s	00110000
+ ◆ /matrixprocessor/RAM_MATRIX_IN_32_s	01001000
+ ◆ /matrixprocessor/RAM_MATRIX_IN_33_s	00100000
+ ◆ /matrixprocessor/DETER_MAT_A_s	00010000
+ ◆ /matrixprocessor/DETER_MAT_B_s	01110011
+ ◆ /matrixprocessor/DETER_MAT_D_s	00110000
◆ /matrixprocessor/current_state	COMPLETE

$$\begin{bmatrix} 0 & 4 & 2 \\ 0 & 1 & 5 \\ 8 & 0 & 0 \end{bmatrix} \begin{bmatrix} 6 & 9 & 4 \\ 5 & 0 & 8 \\ 4 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 28 & 0 & 34 \\ 25 & 0 & 13 \\ 48 & 72 & 32 \end{bmatrix}$$

Conclusion



Kesimpulan

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 dikhususkan untuk komputasi matriks.



Terima Kasih

...