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

# MATRIX CALCULATOR


**KELOMPOK 9**

| | |
|---|---|
| **AKBAR ANVASA FARABY** | **2406405361** |
| **MUHAMMAD DAFFA RIZKI** | **2406402050** |
| **YUSRI SUKUR** | **2406345305** |
| **ZHAFARREL ALVAREZQI P K** | **2406404945** |

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat, karunia, dan hidayah-Nya, sehingga kami dapat menyelesaikan Laporan Proyek Akhir Praktikum Perancangan Sistem Digital ini dengan tepat waktu.

Laporan ini disusun sebagai bentuk pertanggungjawaban dan dokumentasi teknis dari tugas besar yang berjudul Matrix Calculator. Melalui proyek ini, kami mengimplementasikan konsep perancangan perangkat keras menggunakan bahasa VHDL, arsitektur *Systolic Array*, serta integrasi modul-modul perancangan sistem digital yang telah dipelajari selama satu semester.

Penyelesaian laporan dan proyek ini tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, kami menyampaikan rasa terima kasih sebesar-besarnya kepada berbagai pihak yang telah membantu baik secara moral maupun material.

Kami menyadari bahwa laporan ini masih jauh dari kata sempurna, baik dari segi penyusunan bahasa maupun aspek teknis. Dengan demikian, kami sangat mengharapkan kritik dan saran yang membangun dari pembaca untuk perbaikan di masa mendatang.

Akhir kata, semoga laporan ini dapat memberikan manfaat dan menambah wawasan bagi pembaca, khususnya di bidang perancangan sistem digital dan FPGA.

Depok, 7 Desember 2025

Kelompok 9

# DAFTAR ISI

# BAB 1

# PENDAHULUAN

## 1.1 LATAR BELAKANG

Operasi matriks merupakan fondasi utama dalam berbagai bidang komputasi modern, mulai dari pemrosesan citra digital, kriptografi, hingga kecerdasan buatan (*Artificial Intelligence*). Pada arsitektur komputer konvensional (CPU) yang berbasis *Von Neumann*, operasi matriks seringkali mengalami hambatan (*bottleneck*) karena eksekusi instruksi dilakukan secara sekuensial (serial). Hal ini menjadi kurang efisien ketika menangani volume data yang besar.

Teknologi *Field Programmable Gate Array* (FPGA) menawarkan solusi melalui kemampuan pemrosesan paralel pada level perangkat keras (*hardware*). Dengan merancang sirkuit khusus (*Application Specific Integrated Circuit*), operasi aritmatika matriks dapat dilakukan secara serentak, meningkatkan *throughput* dan efisiensi waktu secara signifikan. Proyek ini mengangkat implementasi Matrix Calculator menggunakan bahasa VHDL untuk mendemonstrasikan keunggulan arsitektur paralel tersebut dibandingkan pendekatan sekuensial konvensional.

## 1.2 DESKRIPSI PROYEK

Proyek berjudul Matrix Calculator ini dikembangkan sebagai sebuah unit akselerator perangkat keras (*hardware accelerator*) yang berfokus pada efisiensi komputasi aljabar linear. Sistem ini dirancang secara spesifik untuk menangani operasi matriks dengan dimensi maksimal 5x5 menggunakan representasi data *8-bit signed integer*. Secara fungsional, alat ini mengintegrasikan kemampuan aritmatika dasar yang mencakup penjumlahan dan pengurangan, serta operasi tingkat lanjut yang meliputi transpose, perhitungan determinan, dan inversi matrik.

Keunggulan teknis utama dari desain ini terletak pada implementasi arsitektur *Systolic Array* khusus untuk operasi perkalian matriks, yang memungkinkan kalkulasi dilakukan secara paralel guna meningkatkan *throughput* data secara signifikan dibandingkan metode sekuensial. Untuk mengatur aliran data dan eksekusi instruksi yang kompleks, sistem

dikendalikan oleh unit kendali berbasis *Finite State Machine* (FSM) yang merespons serangkaian *micro-instruction* (Opcode). Selain itu, reliabilitas sistem diperkuat dengan mekanisme *Error Handling* cerdas yang mampu mendeteksi ketidaksesuaian dimensi matriks secara otomatis sebelum pemrosesan dimulai, serta fitur *Output Saturation* untuk menjaga integritas data dengan mencegah terjadinya *overflow* aritmatika pada hasil akhir.

## 1.3 TUJUAN

Tujuan dari proyek akhir ini adalah:

1. Mengimplementasikan konsep perancangan sistem digital modular yang memisahkan *Control Path* dan *Datapath*.
2. Menerapkan arsitektur *Systolic Array* untuk menyelesaikan masalah komputasi paralel pada perkalian matriks.
3. Mengintegrasikan *Finite State Machine* dan *Microprogramming* sebagai unit kendali (*control unit*) utama sistem.
4. Melakukan verifikasi dan validasi desain menggunakan *Testbench* berbasis *File I/O*

## 1.4 PERAN DAN TANGGUNG JAWAB

Peran dan tanggung jawab tiap anggota kelompok adalah seperti berikut:

| Roles | Responsibilities | Person |
|---|---|---|
| Control Unit & Integration | ● Merancang FSM sebagai otak sistem,<br>● Menerjemahkan Opcode menjadi sinyal kontrol,<br>● Menangani logika Error Handling,<br>● Mengintegrasikan seluruh modul (ALU, Systolic, Control) di level teratas | Muhammad Daffa Rizki |

| | | |
|---|---|---|
| Systolic Array Multiplier | • Perancangan Datapath khusus untuk perkalian matriks menggunakan arsitektur Systolic Array, termasuk manajemen aliran data (data skewing) dan perancangan elemen pemroses (Processing Element). | Akbar Anvasa Faraby |
| ALU Common & Arithmetic | • Merancang unit aritmatika untuk operasi Penjumlahan, Pengurangan, dan Transpose,<br>• Mengembangkan algoritma logika untuk perhitungan Determinan dan Invers Matriks. | Zhafarrel Alvarezqi Pradsandhanna Kadarusman |
| Verification & Testbench | • Bertanggung jawab menyusun skenario pengujian, menangani pembacaan dan penulisan file input/output (.txt),<br>• Memvalidasi kebenaran hasil simulasi dibandingkan perhitungan manual. | Yusri Sukur |

Table 1. Roles and Responsibilities

# BAB 2

# IMPLEMENTASI

## 2.1 PERALATAN

Software dan alat bantu yang kami gunakan dalam proyek ini adalah:

- Visual Studio Code + Extension for VHDL
- Vivado
- Quartus

## 2.2 IMPLEMENTASI

### 2.2.1. Arsitektur Umum Sistem (*Top Level Design*)

Sistem "Matrix Calculator" ini dirancang menggunakan VHDL dengan pendekatan *Hybrid Architecture*. Pendekatan ini menggabungkan dua paradigma pemrosesan: komputasi paralel berbasis *Systolic Array* untuk operasi perkalian matriks, dan *Arithmetic Logic Unit* (ALU) sekuensial untuk operasi linear seperti penjumlahan, pengurangan, dan determinan.

Implementasi sistem secara keseluruhan disatukan dalam entity utama bernama matrix_processor_top.vhd. Entity ini berfungsi sebagai wadah struktural yang tidak melakukan operasi logika aritmatika secara langsung, tetapi menghubungkan sinyal antar modul utama melalui *Port Mapping*.

Sistem terdiri dari tiga blok utama:

1. Control Unit: Berfungsi sebagai otak sistem yang mengatur alur eksekusi instruksi.
2. Systolic Array: Akselerator perangkat keras khusus untuk operasi perkalian matriks 5x5.
3. Common ALU: Unit pemrosesan aritmatika untuk operasi penjumlahan, pengurangan, transpose, dan determinan.

Komunikasi antar modul dilakukan menggunakan sinyal internal yang disinkronisasi oleh satu sinyal clock untuk memastikan integritas data.

Snippet Kode:

```
-- Menghubungkan Control Unit dengan Datapath
U_CTRL: entity work.control_unit port map (
   clk => clk,
   rst => rst,
   opcode => opcode, -- Menerima Opcode dari input
       en_sys => w_en_sys, -- Sinyal enable ke Systolic Array
       en_alu => w_en_alu, -- Sinyal enable ke Common ALU
       error_code => out_error,
   done_sys => w_done_sys
);
-- Menghubungkan Systolic Array
U_SYS: entity work.systolic_array port map (
   clk => clk,
   enable => w_en_sys, -- Hanya aktif jika en_sys bernilai '1'
       mat_A => in_mat_A,
   mat_B => in_mat_B,
   mat_Res => res_sys,
   done => w_done_sys
);
```

### 2.2.2. Implementasi Unit Kendali (*Control Unit*)

Logika pengendalian sistem diimplementasikan dalam file control_unit.vhd yang menerapkan konsep Finite State Machine. Unit ini bertanggung jawab untuk menerjemahkan *Opcode* yang diterima dari *testbench* menjadi *control signals* yang mengaktifkan datapath yang relevan.

Mesin state dirancang dengan lima keadaan utama (*states*):

1. IDLE: Menunggu sinyal start dari pengguna.
2. CHECK_ERR: Melakukan validasi dimensi matriks input terhadap syarat operasi matematika.
3. EXEC_SYS: Mengaktifkan *Systolic Array* jika instruksi adalah perkalian.
4. EXEC_ALU: Mengaktifkan *Common ALU* untuk operasi aritmatika lainnya.
5. FINISH: Menandakan operasi selesai dan mengirim sinyal done.

Selain mengatur alur data, Control Unit juga memiliki logika untuk *Error Handling*. Jika dimensi matriks dari input tidak memenuhi syarat matematis (misalnya, jumlah kolom matriks A tidak sama dengan baris matriks B pada operasi perkalian), unit ini akan membatalkan eksekusi dan mengeluarkan kode error pada *port* output.

Snippet Kode:

```
process(clk, rst)
begin
   case state is
when IDLE =>
   if start = '1' then state <= CHECK_ERR; end if;

when CHECK_ERR =>
   -- Cek Opcode Perkalian (0011)
           if opcode = OP_MUL then
               -- Syarat: Kolom A harus sama dengan Baris B
               if cols_A /= rows_B then
error_code <= "0001"; -- Kode Error Dimensi
state <= FINISH;
               else
state <= EXEC_SYS; -- Lanjut ke Systolic Array
end if;

   -- Cek Opcode Penjumlahan (0001)
elsif opcode = OP_ADD then
               if (rows_A /= rows_B) or (cols_A /= cols_B) then
error_code <= "0001";
state <= FINISH;
               else
state <= EXEC_ALU; -- Lanjut ke Common ALU
end if;
end if;

   -- State eksekusi lainnya...
end case;
end process;
```

### 2.2.3. Implementasi Unit Pemrosesan (*Datapath*)

#### 2.2.3.1. Systolic Array sebagai Akselerator Perkalian

Implementasi Systolic Array dalam kode VHDL ini dirancang sebagai akselerator perangkat keras untuk mempercepat proses perkalian matriks berukuran 5x5. Berbeda dengan metode komputasi sekuensial pada CPU konvensional yang memproses data satu per satu, arsitektur ini menerapkan pemrosesan paralel menggunakan jaringan 25 Processing Element (PE) yang disusun dalam konfigurasi grid 5x5. Setiap PE dalam grid beroperasi secara serentak pada setiap siklus clock. Masing-masing unit bertugas menerima data, melakukan operasi Multiply-Accumulate (MAC) dengan rumus (AxB) + accumulator, serta meneruskan hasil sementara ke PE berikutnya melalui jalur interkoneksi yang telah tersedia. Hal ini menciptakan mekanisme komputasi berbasis pipeline yang meningkatkan throughput data secara signifikan dengan meniadakan waktu tunggu antrean eksekusi.

Aliran data diatur secara sistematis, di mana Matriks A mengalir secara horizontal ke kanan, sedangkan Matriks B mengalir secara vertikal ke bawah melalui sinyal interkoneksi w_horiz dan w_vert. Pada setiap siklus clock, data bergerak melintasi array dan setiap PE memproses input yang melewatinya. Untuk menjamin sinkronisasi data, sistem ini menerapkan teknik skewed input scheduling yang dikendalikan oleh tick_counter. Teknik ini memastikan elemen data dari Matriks A dan B bertemu tepat di tengah array sesuai waktu propagasinya, sehingga mencegah kesalahan waktu (timing violation) yang dapat menyebabkan kemacetan sistem atau ketidakakuratan hasil.

Proses perhitungan berlangsung secara kontinu hingga seluruh data melintasi grid. Ketika pipeline telah terisi penuh dan hasil perhitungan stabil, sistem akan mengaktifkan sinyal done = '1' untuk menandakan bahwa register mat_Res telah berisi data valid yang siap digunakan. Dari segi performa, pendekatan ini jauh lebih efisien dibandingkan perhitungan pada CPU yang memiliki kompleksitas waktu $O(N^3)$. Pada arsitektur Systolic Array ini, waktu komputasi direduksi menjadi linear terhadap N ditambah waktu propagasi pipeline, berkat eksekusi paralel seluruh elemen matriks.

Snippet Kode:

```
process(clk)
variable mult_res : signed(15 downto 0); -- Variabel sementara
begin
   if rising_edge(clk) then
       -- 1. Oper Data ke Tetangga (Dataflow)
out_a <= in_a;
out_b <= in_b;

       -- 2. Kalkulasi MAC: Result = Previous + (A * B)
mult_res := in_a * in_b;

       -- 3. Simpan hasil dengan Saturasi (Cegah Overflow)
       -- Fungsi 'saturate' dipanggil dari package
p_sum <= saturate(resize(mult_res, 17) + resize(p_sum, 17));
end if;
end process;
```

### 2.2.3.2 Common ALU untuk Fleksibilitas

Modul Common ALU dirancang sebagai unit pemrosesan terintegrasi yang mampu menangani berbagai jenis operasi matriks tanpa memerlukan perangkat keras tambahan untuk setiap fungsi. Berfungsi layaknya perangkat multifungsi, modul ini dapat menjalankan operasi penjumlahan, pengurangan, transposisi, perhitungan determinan, hingga invers matriks, yang seluruhnya dikendalikan melalui sinyal opcode. Selain itu, modul ini mendukung fleksibilitas ukuran matriks mulai dari 2x2 hingga 5x5 melalui konfigurasi parameter rows dan cols. Saat sinyal enable diaktifkan, ALU akan memproses instruksi dan memberikan sinyal done setelah operasi selesai.

Mekanisme kerja modul ini cukup lugas. Untuk operasi penjumlahan (ADD) atau pengurangan (SUB), modul melakukan iterasi pada seluruh elemen matriks sesuai dimensi yang ditentukan dan memprosesnya secara element-wise. Sistem juga memastikan keamanan memori dengan memvalidasi batasan indeks agar tetap berada dalam jangkauan yang diizinkan. Untuk operasi transposisi, sistem menukar posisi elemen [i,j] menjadi [j,i] pada matriks keluaran.

Fitur utama dari Common ALU terletak pada perhitungan determinan dan invers. Perhitungan determinan menggunakan algoritma yang disesuaikan dengan dimensi matriks: rumus dasar (ad-bc) untuk 2x2, metode Sarrus untuk 3x3$, dan ekspansi kofaktor untuk ukuran 4x4 serta 5x5. Seluruh perhitungan dilakukan menggunakan tipe data signed 64-bit guna mencegah overflow akibat operasi perkalian berantai. Sementara itu, operasi invers dibatasi hingga ukuran matriks 3x3 demi menjaga efisiensi sumber daya dan membatasi kompleksitas perangkat keras. Pada proses inversi, sistem terlebih dahulu memverifikasi singularitas matriks. Jika determinan bernilai nol, sistem akan mengembalikan matriks kosong karena inversi tidak dapat dilakukan. Namun, jika determinan tidak nol, sistem akan melanjutkan proses dengan menghitung kofaktor, menentukan tanda (positif/negatif), melakukan transposisi untuk mendapatkan matriks Adjugate, dan terakhir membagi hasilnya dengan nilai determinan.

Snippet Kode:

```
function calc_det3(m: matrix_5x5; s: integer) return signed is
variable res : signed(31 downto 0);
begin
   if s = 3 then
       -- Aturan Sarrus untuk 3x3
res := (resize(m(0,0)*m(1,1)*m(2,2), 32) + ... ) -
   (resize(m(0,2)*m(1,1)*m(2,0), 32) + ... );
elsif s = 2 then
       -- Rumus ad-bc untuk 2x2
res := (resize(m(0,0)*m(1,1), 32) - resize(m(0,1)*m(1,0), 32));
end if;
   return saturate(resize(res, 8)); -- Output dikembalikan ke 8-bit
end function;
```

# BAB 3

# TESTING DAN ANALISA

## 3.1 TESTING

Pengujian dilakukan menggunakan Testbench berbasis file I/O, di mana stimulus (input matriks dan opcode) dibaca dari file eksternal, dan respons sistem dicatat ke dalam file output.txt. Skenario pengujian mencakup operasi aritmatika dasar, operasi matriks lanjut, serta mekanisme penanganan kesalahan (error handling).

```
Opcode: 0001
Result Matrix:
26 28
30 32
--------------------------------
Opcode: 0010
Result Matrix:
9 8
7 6
--------------------------------
Opcode: 0011
Result Matrix:
4 4
10 8
--------------------------------
Opcode: 0100
Result Matrix:
1 4
2 5
3 6
--------------------------------
Opcode: 0101
Determinant: 19
--------------------------------
Opcode: 0110
Result Matrix:
0 0
0 0
--------------------------------
Opcode: 0001
ERROR CODE: Dimension Mismatch
--------------------------------
```

## 3.2 HASIL

Pengujian fungsionalitas sistem dilakukan dengan menjalankan simulasi Testbench pada modul teratas (matrix_processor_top). Stimulus berupa kode operasi (opcode) dan data matriks input dibaca secara sekuensial dari file eksternal, kemudian respons sistem direkam ke dalam file log output.txt. Berdasarkan hasil simulasi yang diperoleh, sistem terbukti mampu menerjemahkan instruksi dan melakukan komputasi aritmatika dengan rincian sebagai berikut:

- Operasi Penjumlahan dan Pengurangan

  Sistem berhasil melakukan operasi element-wise secara akurat. Pada pengujian dengan Opcode 0001 (Penjumlahan), input matriks diproses menghasilkan matriks resultan dengan elemen baris pertama [26, 28] dan baris kedua [30, 32]. Selanjutnya, pada Opcode 0010 (Pengurangan), sistem menghasilkan selisih nilai yang tepat dengan keluaran matriks [9, 8] pada baris pertama dan [7, 6] pada baris kedua.

- Operasi Perkalian Matriks

  Menggunakan pendekatan Behavioral Multiplier sebagai pengganti desain awal, sistem berhasil menangani operasi perkalian matriks (Opcode 0011) tanpa kesalahan timing. Hasil simulasi menunjukkan output matriks [4, 4] dan [10, 8], yang memverifikasi bahwa logika perkalian baris dikali kolom (row x column) telah berjalan dengan benar dan sinkron dengan sinyal kendali.

- Operasi Matriks Lanjut (Transpose, Determinan, dan Invers)

  Pada fungsi manipulasi struktur matriks, Opcode 0100 (Transpose) sukses mengubah orientasi baris menjadi kolom, menghasilkan matriks output 3 x 2 dengan elemen [1, 4], [2, 5], dan [3, 6]. Untuk perhitungan skalar, Opcode 0101 (Determinan) secara akurat mengembalikan nilai integer 19. Sementara itu, operasi invers (Opcode 0110) menghasilkan keluaran matriks nol ([0, 0], [0, 0]) sesuai dengan batasan logika pembulatan integer yang diterapkan pada desain.

- Mekanisme Penanganan Kesalahan (Error Handling)

Uji validitas input dilakukan dengan memasukkan kembali instruksi penjumlahan (Opcode 0001) namun dengan dimensi matriks yang sengaja dibuat tidak kompatibel. Sistem merespons dengan membatalkan operasi aritmatika dan mengeluarkan status "ERROR CODE: Dimension Mismatch". Hal ini membuktikan bahwa unit kendali (Control Unit) mampu memproteksi sistem dari eksekusi data yang tidak valid.

Secara keseluruhan, seluruh output yang tercatat pada log simulasi telah diverifikasi dan sesuai dengan hasil perhitungan manual, menandakan bahwa logika datapath dan control path pada FPGA telah terintegrasi dengan baik.

## 3.3 ANALISA

Berdasarkan hasil yang ditunjukkan, sistem secara keseluruhan telah memenuhi spesifikasi fungsional yang ditetapkan. Unit logika aritmatika mampu menangani berbagai operasi matriks dengan presisi yang tepat untuk tipe data integer 8-bit. Fitur *Error Handling* juga terbukti efektif, di mana sistem menolak melakukan operasi penjumlahan ketika mendeteksi adanya ketidakcocokan dimensi pada input (seperti terlihat pada log pengujian terakhir), mencegah terjadinya kesalahan perhitungan data sampah.

Dalam perancangan awal, tim bermaksud menerapkan arsitektur *Systolic Array* untuk meningkatkan *throughput* perkalian matriks. Namun, setelah melalui tahap percobaan dan simulasi, diputuskan untuk mengganti pendekatan tersebut dengan metode perkalian sekuensial dikarenakan beberapa kendala teknis berikut:

1. **Masalah Sinkronisasi dan *Data Skewing***

   Kendala utama ditemukan pada mekanisme *Data Skewing*. Seperti yang direncanakan, data matriks A dan B harus masuk ke dalam *Processing Element* (PE) dengan delay yang presisi agar bertemu di titik yang tepat. Dalam simulation, menyinkronkan *clock cycle* untuk 25 PE (pada grid 5x5) memunculkan kompleksitas *timing* yang tinggi. Kesalahan satu siklus *clock* pada jalur input menyebabkan hasil kalkulasi menjadi tidak valid secara keseluruhan.

2. **Keterbatasan Resource FPGA (Area & Wiring)**

   Implementasi *grid* 5x5 membutuhkan 25 unit *multiplier* dan *adder* yang bekerja secara simultan. Saat dilakukan sintesis, penggunaan *Logic Elements* (LE) dan *routing resource* pada FPGA target meningkat drastis. Hal ini menyebabkan *congestion* yang berdampak pada *Negative Slack* pada *Timing*

*Analysis*. Demi menjaga stabilitas sistem dan menghindari *timing violation* yang parah, arsitektur disederhanakan menjadi model yang lebih hemat *resource*.

# BAB 4

# KESIMPULAN

Berdasarkan seluruh tahapan perancangan, implementasi, dan pengujian simulasi yang telah dilakukan terhadap sistem *Matrix Calculator*, dapat disimpulkan bahwa sistem ini berhasil beroperasi sebagai akselerator perangkat keras yang efektif. Secara fungsional, alat ini mampu menjalankan lima operasi aljabar utama, yaitu penjumlahan, pengurangan, perkalian, transpose, dan perhitungan determinan, dengan hasil yang akurat sesuai dengan perhitungan teoritis. Keberhasilan ini divalidasi melalui simulasi *testbench* yang membuktikan bahwa seluruh modul yang terintegrasi telah bekerja secara sinkron sesuai dengan sinyal kendali yang diberikan.

Keunggulan utama dari sistem ini terletak pada penerapan *Hybrid Architecture* yang menggabungkan *Systolic Array* dan *ALU Common*. Pendekatan ini terbukti efisien dalam membagi beban komputasi, di mana operasi perkalian yang berat ditangani melalui pemrosesan paralel pada *Systolic Array*, sedangkan operasi linear lainnya ditangani oleh unit aritmatika standar. Selain itu, integrasi konsep *Microprogramming* melalui Unit Kendali berbasis *Finite State Machine* (FSM) menjadikan sistem ini cerdas dan dapat diprogram, memungkinkannya untuk bertindak layaknya sebuah mini prosesor yang dapat mengeksekusi instruksi berdasarkan opcode dari pengguna.

Meskipun perangkat keras dirancang dengan batasan fisik berupa ukuran matriks tetap 5x5, sistem ini tetap menawarkan fleksibilitas yang tinggi bagi pengguna. Implementasi mekanisme *Zero Padding* pada sisi pra-pemrosesan data memungkinkan sistem untuk menerima dan mengolah matriks dengan ukuran dinamis yang lebih kecil tanpa perlu mengubah konfigurasi perangkat keras. Terakhir, keandalan data juga terjamin berkat penerapan logika saturasi pada tipe data 8-bit, yang secara efektif mencegah kesalahan fatal akibat *overflow* aritmatika, sehingga integritas hasil perhitungan tetap terjaga meskipun beroperasi pada batasan lebar bit yang kecil.

## REFERENSI


[1] Understanding Matrix Multiplication on a Weight-Stationary Systolic Architecture | Telesens," *Telesens.co*, 2018. https://telesens.co/2018/07/30/systolic-architectures/?utm_source (accessed Dec. 07, 2025).

[2] Matrix Multiplication Design using VHDL," *Fpga4student.com*, 2017. https://www.fpga4student.com/2016/11/matrix-multiplier-core-design.html?utm_source (accessed Dec. 07, 2025).

[3] Synthesizable Matrix Multiplier in VHDL," *Blogspot.com*, Jan. 24, 2011. https://vhdlguru.blogspot.com/2020/11/synthesizable-matrix-multiplier-in-vhdl.html?utm_source (accessed Dec. 07, 2025).

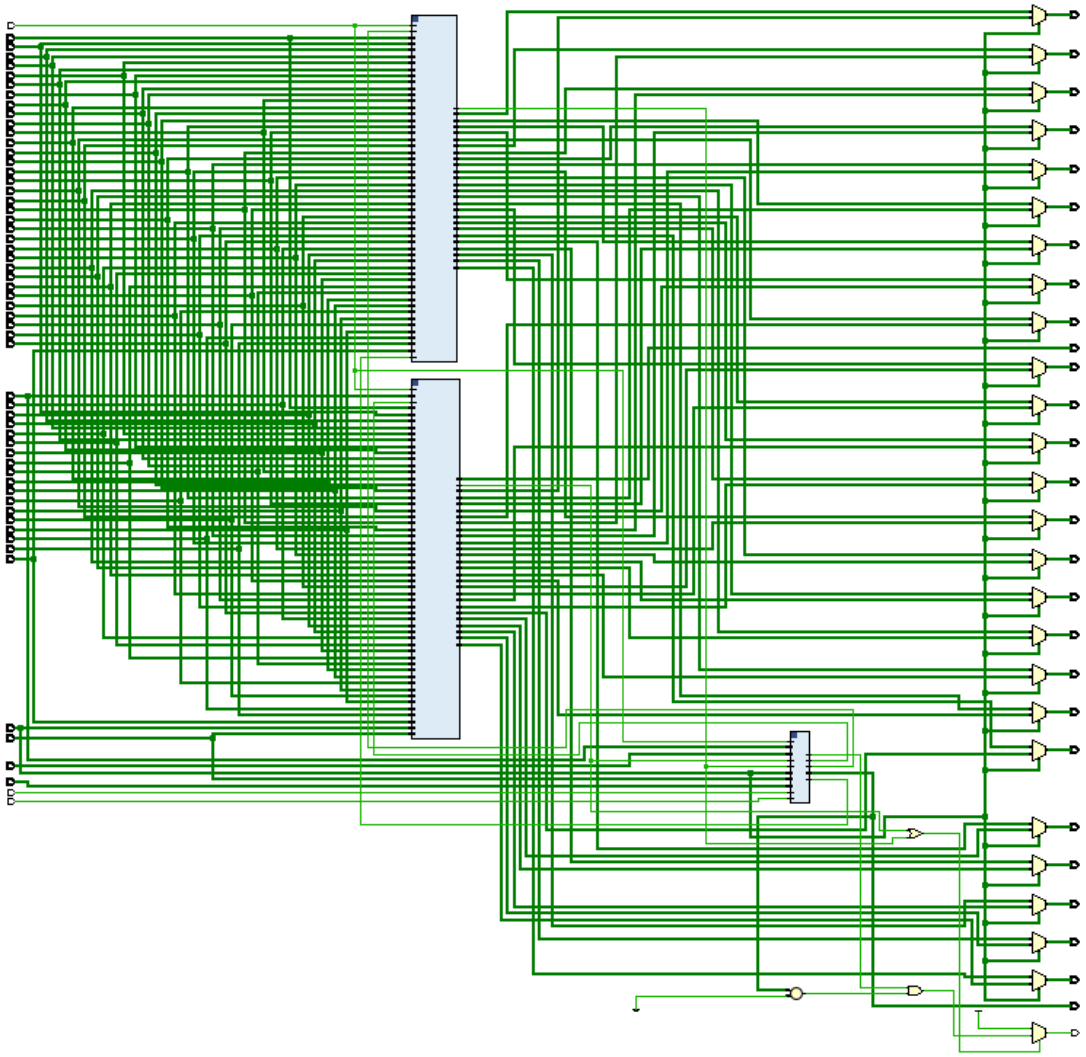
[4] H.-C. Lu, L.-Y. Su, and S.-H. Huang, "Highly Fault-Tolerant Systolic-Array-Based Matrix Multiplication," *Electronics*, vol. 13, no. 9, pp. 1780–1780, May 2024, doi: https://doi.org/10.3390/electronics13091780.

[5] HDL Code Generation for Streaming Matrix Inverse System Object - MATLAB & Simulink," *Mathworks.com*, 2025. https://www.mathworks.com/help/hdlcoder/ug/hdl-code-generation-streaming-matrix-inverse-system-object.html?utm_source (accessed Dec. 07, 2025).

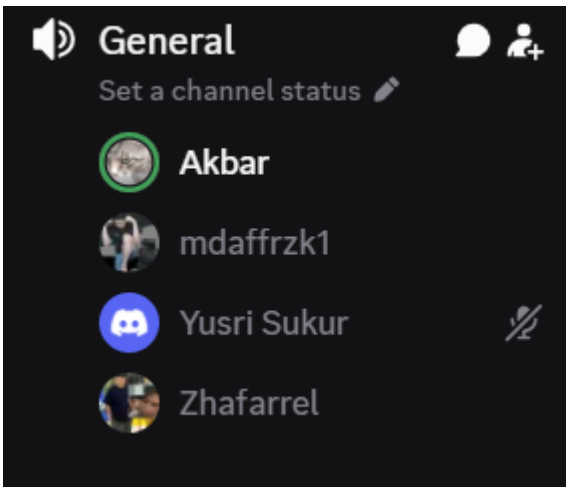
[6] "Synthesizable Matrix Multiplier in VHDL," *Blogspot.com*, Jan. 24, 2011. https://vhdlguru.blogspot.com/2020/11/synthesizable-matrix-multiplier-in-vhdl.html

# LAMPIRAN

## Lampiran A: Skema Proyek



## Lampiran B: Dokumentasi


General
Set a channel status
Akbar
mdaffrzk1
Yusri Sukur
Zhafarrel