Matrix Calculator

Final Project PSD Kelompok 09

- Akbar Anvasa Faraby 2406405361
- M. Daffa Rizki 2406402050
- Yusri Sukur 2406345305
- Zhafarrel Alvarezqi P. K 2406404945

Outline

Latar Belakang

Arsitektur Sistem

Implementasi

Hasil Simulasi

Analisis

Kesimpulan

Latar Belakang

Masalah:

- Kebutuhan akselerasi operasi matriks pada **embedded system**.
- Mengatasi limitasi pemrosesan sekuensial pada CPU standar.

Solusi:

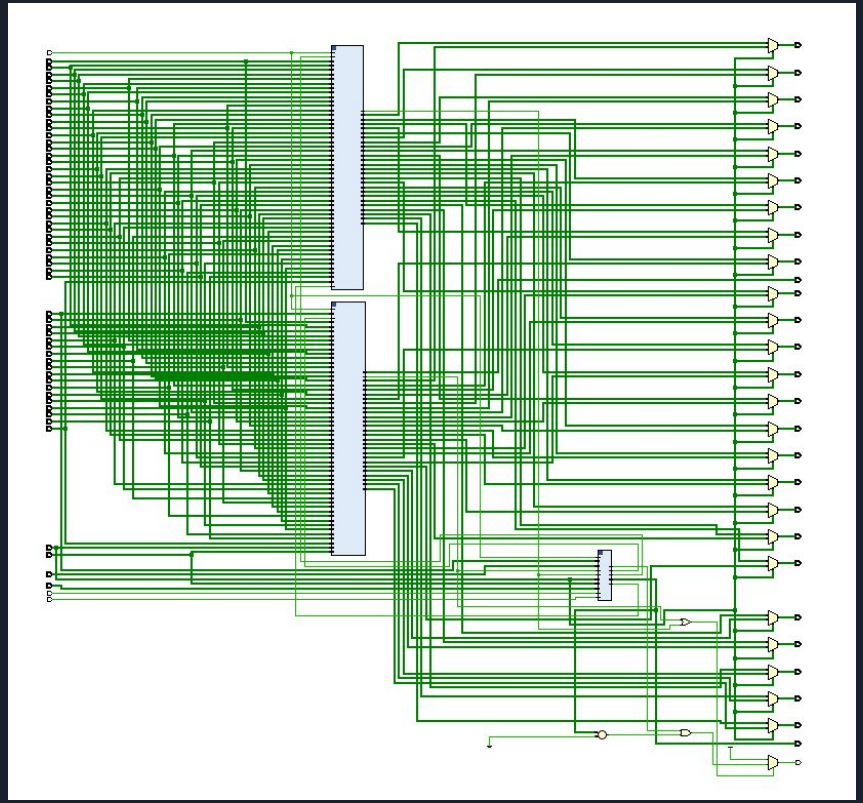
- **Modular Design:** Pemisahan antara Control Path (FSM) dan Datapath (ALU Multiplier).
- **Robustness:** Sistem dilengkapi proteksi saturasi dan validasi dimensi.

Spesifikasi:

- Input matriks maksimal 5x5
- Operasi terdiri dari Add, Sub, Mul, Transpose, Det, Inverse
- Data input 8-bit Signed Integer (rentang antara -128 s.d. 127)

Arsitektur Sistem

RTL Schematic

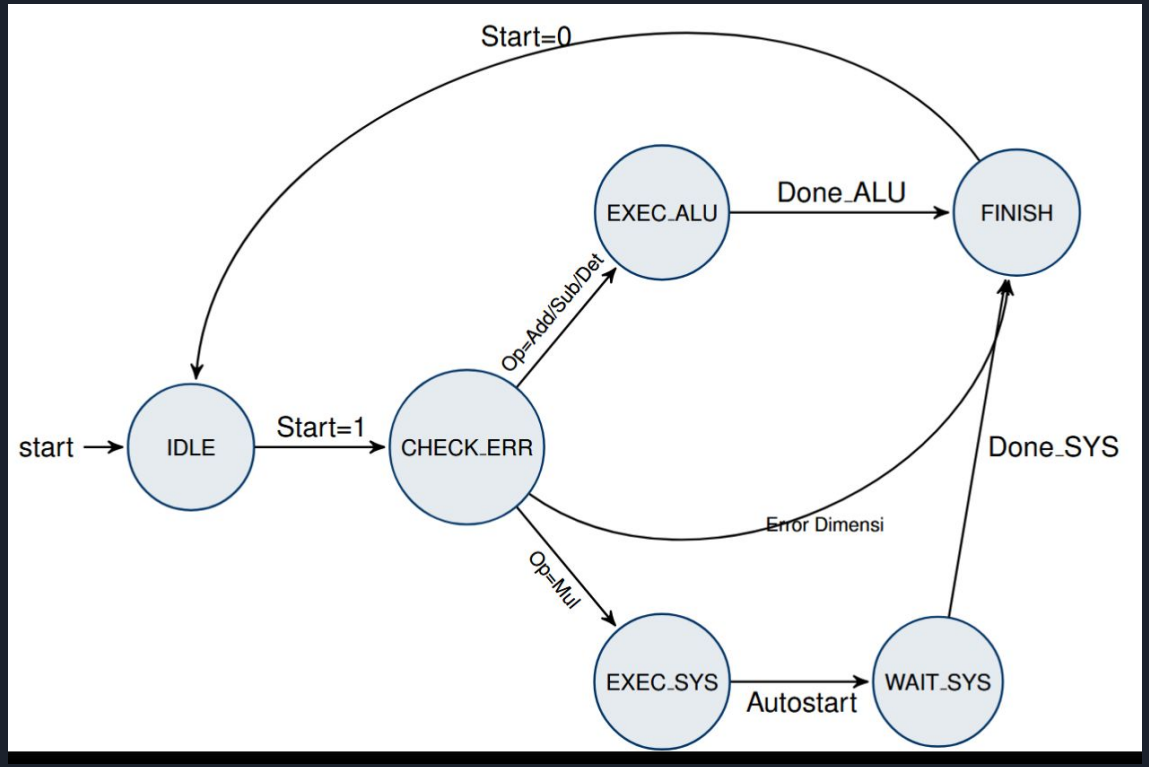


Komponen Utama:

- **Control Unit:** FSM untuk manajemen *state* dan *error handling*.
- **ALU Common**: Menangani operasi aritmatika dasar (Add, Sub) dan kompleks (Det, Inv).
- **Matrix Multiplier:** Unit khusus untuk operasi perkalian matriks $O(N^3)$

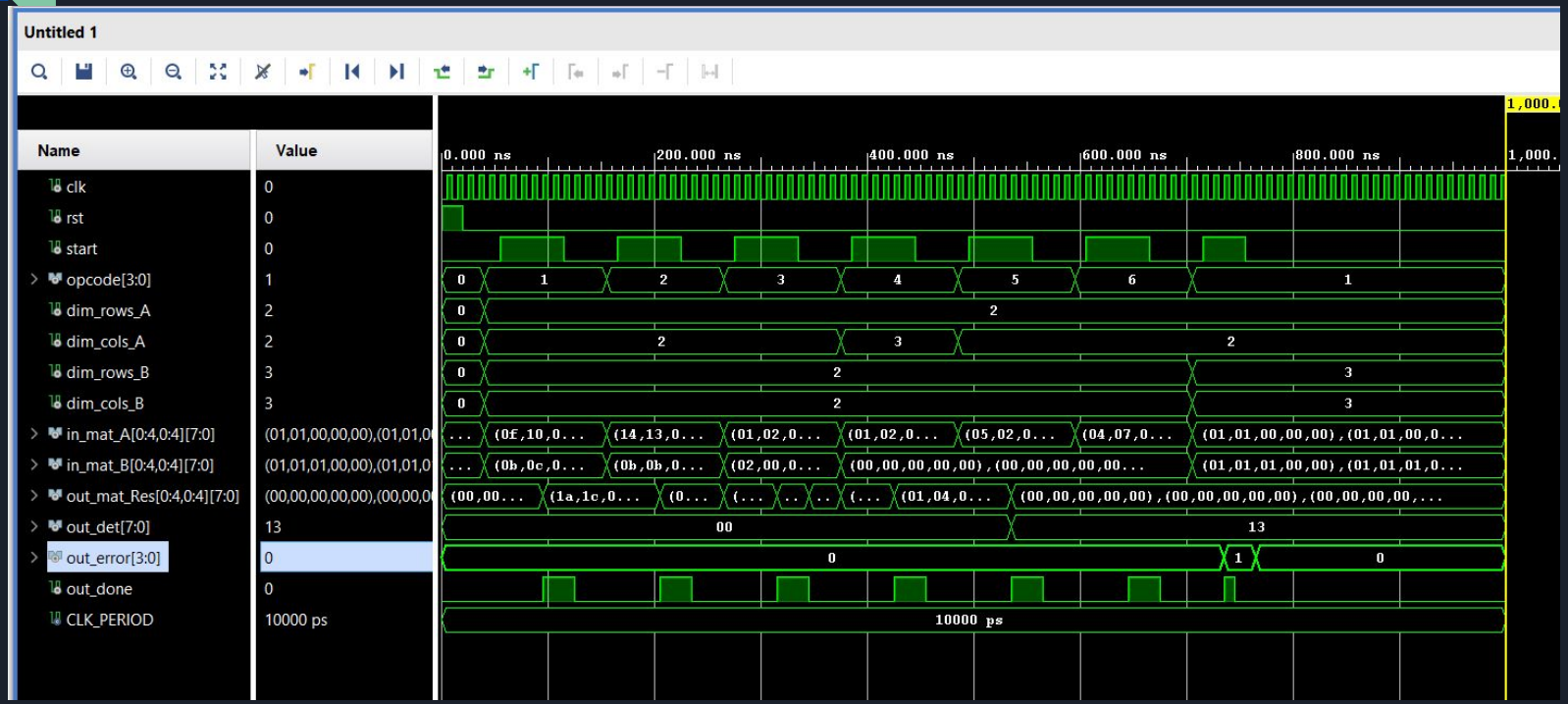
Arsitektur Sistem

Control Unit (FSM)



Arsitektur Sistem

Waveform



Implementasi

Unit Perkalian Matriks (Behavioral)

```
-- Loop Baris & Kolom
for i in 0 to MAX_SIZE-1 loop
  for j in 0 to MAX_SIZE-1 loop
    sum := (others => '0');
    -- Dot Product Loop
    for k in 0 to MAX_SIZE-1 loop
       mult_val := mat_A(i,k) *
            mat_B(k,j);
       sum := sum + resize(mult_val, 32);
    end loop;
    -- Assign with Saturation
    result_reg(i,j) <= saturate(sum);
  end loop;
end loop;
```

Desain Multiplier:

- Menggunakan pendekatan Behavioral Loop pada VHDL.
- Melakukan operasi Triple Loop (Baris, Kolom, Dot-Product) dalam satu blok proses.
- **Saturasi:** Hasil penjumlahan dikunci pada range 8-bit (-128 s.d. 127) untuk mencegah *overflow* aritmatika.

Implementasi

Unit Perkalian Matriks (Behavioral)

Mekanisme Saturasi (matrix pkg.vhdl): Menjamin data output tetap valid dalam format 8-bit signed.

```
function saturate(val : signed) return signed is
begin
    if val > 127 then
        return to_signed(127, 8); -- Clamping Max
    elsif val < -128 then
        return to_signed(-128, 8); -- Clamping Min
    else
        return resize(val, 8); -- Normal
    end if;
end function;
```

Operasi Kompleks:

- **Determinan:** Menggunakan ekspansi kofaktor rekursif (Det5 memanggil Det4, dst).
- **Inverse**: Menggunakan metode Adjoin, dibatasi max 3x3 karena kompleksitas pembagian integer.

Implementasi

Common ALU Aritmatika & Transpose (Behavioral)

Satu modul terintegrasi yang menangani banyak operasi (kecuali perkalian) berdasarkan sinyal Opcode.

```
when OP_ADD =>
    for i in 0 to MAX_SIZE - 1 loop
        for j in 0 to MAX_SIZE - 1 loop
            -- Validasi dimensi input user
            if i < rows and j < cols then
                -- Resize ke 17-bit untuk calculation headroom
                temp_calc := resize(mat_A(i, j), 17)
                           + resize(mat_B(i, j), 17);

                -- Simpan hasil dengan saturasi
                mat_Res_reg(i, j) <= saturate(temp_calc);
            end if;
        end loop;
    end loop;
    done_reg <= '1';
```

- **Arsitektur Behavioral:** Menggunakan process(clk) yang merespons sinyal opcode.
- **Dynamic Sizing:** Menggunakan pengecekan if i < rows and j < cols di dalam loop. Ini memungkinkan ALU memproses matriks 2x2, 3x3, hingga 5x5 tanpa mengubah hardware.

Implementasi

Common ALU Aritmatika & Transpose (Behavioral)

Satu modul terintegrasi yang menangani banyak operasi (kecuali perkalian) berdasarkan sinyal Opcode.

```
-- Function Determinan 3x3
function get_det3(m : matrix_5x5) return signed is
    variable term1, term2, ... : signed(63 downto 0);
begin
    -- Implementasi Aturan Sarrus
    term1 := resize(resize(m(0, 0), 20) * resize(m(1, 1), 20) *
     resize(m(2, 2), 20), 64);
    -- ... (term 2 sampai 6) ...
    res := term1 + term2 + term3 - term4 - term5 - term6;
    return res;
end function;

-- Pemanggilan di Process
when OP_DET =>
    if rows = 3 then det_temp := get_det3(mat_A);
    elsif rows = 4 then det_temp := get_det4(mat_A);
    -- ...
```

- **Modularitas:** Rumus matematika rumit dipisah ke dalam FUNCTION VHDL (get_det3, get_det4, get_cofactor_val) agar kode utama bersih
- **Logika Invers:** Menggunakan metode Adjoin (Cofactor/Determinan)

Hasil Simulasi

Test case 3: Perkalian (Opcode 0011)

Input Matriks A	Input Matriks B	Output FPGA
$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	$\begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix}$	$\begin{pmatrix} 4 & 4 \\ 10 & 8 \end{pmatrix}$
*Status: **MATCH** (Sesuai perhitungan manual)*		

Test case 7: Error Handling Penjumlahan (Opcode 0001)

- Input: Matriks A (2 x 2) + Matriks B (3 x 3)
- Output: Error code "0001" (Dimension Mismatch)
- Control Unit berhasil mendeteksi kesalahan sebelum eksekusi

Analisis

Analisis Desain

- Resource Usage: Penggunaan loop behavioral pada multiplier memungkinkan sintesis optimal oleh tools FPGA (DSP slices).

Isu pada operasi Inverse

Terdapat **limitasi integer division**

- Rumus: $A^{-1} = (1 / \det(A)) \times Adj(A)$
- Karena menggunakan **Integer 8-bit**, pembagian elemen Adj(A) dengan det(A) sering menghasilkan 0, jika X < det
- Contoh: 4/10 = 0 (pada integer)
- Solusi untuk kedepannya, dengan mengimplementasikan Fixed Point atau Floating Point Unit

Kesimpulan

- Desain berhasil mengintegrasikan FSM, ALU, dan Multiplier dalam satu top-level entity.
- Seluruh operasi dasar (Add, Sub, Mul) dan Error Handling berfungsi sesuai spesifikasi input.txt.
- Fitur saturasi efektif mencegah *overflow* aritmatika pada operasi signed 8-bit.

Terima Kasih