INF01175
Sistemas Digitais para Computadores A

Projeto PC-PO/HLS

Operações com Matrizes 2x2

Pedro Lubaszewski Lima (00341810)
Turma U

23 de agosto de 2024

Sumário

Enunciado do Problema

Objetivo

Projetar e descrever em VHDL um circuito que multiplique duas matrizes, some com uma terceira matriz e filtre a matriz resultado de acordo com a definição do cálculo abaixo. Além disso, compararar a construção manual com a solução HLS da Vitis.

Enunciado do Problema

Objetivo

Projetar e descrever em VHDL um circuito que multiplique duas matrizes, some com uma terceira matriz e filtre a matriz resultado de acordo com a definição do cálculo abaixo. Além disso, compararar a construção manual com a solução HLS da Vitis.

Função da Saída do Sistema (R_{2x2})

Dadas as matrizes A_{2x2}, B_{2x2} e C_{2x2} tais que a_{ij}, b_{ij}, $c_{ij} \in \mathbb{N}$ e $a_{ij} < 255$, $b_{ij} < 255$, $c_{ij} < 65535$, $\forall i, j \in \{1, 2\}$, e

$$F(M) = \begin{cases} m_{ij} & \text{se } 0 < m_{ij} \leqslant 128 \\ 128 & \text{se } m_{ij} > 128 \end{cases}, \text{ então}$$

$$R = F[(A \times B) + C]$$

Fluxograma ASM

Algoritmo em C

```
1   #ifndef __MATRIXOP_H__
2   #define __MATRIXOP_H__
3
4   #include <cmath>
5   using namespace std;
6
7   #define MATRIX_A_ROWS 2
8   #define MATRIX_A_COLUMNS 2
9   #define MATRIX_B_ROWS 2
10  #define MATRIX_B_COLUMNS 2
11
12  typedef unsigned char  matrix_a_t ; // 8 bits
13  typedef unsigned char  matrix_b_t ; // 8 bits
14  typedef unsigned short  matrix_c_t ; // 16 bits
15  typedef unsigned char  result_t ; // 8 bits
16
17  void  calculate_matrix (
18          matrix_a_t  a[MATRIX_A_ROWS][MATRIX_A_COLUMNS],
19          matrix_b_t  b[MATRIX_B_ROWS][MATRIX_B_COLUMNS],
20          matrix_c_t  c[MATRIX_A_ROWS][MATRIX_B_COLUMNS],
21          result_t   result [MATRIX_A_ROWS][MATRIX_B_COLUMNS]);
22
23  #endif
```

Código 1: *Header* de `matrix_operations.cpp`

Algoritmo em C

```c
#include "matrix_operations.h"

void calculate_matrix (
        matrix_a_t  a[MATRIX_A_ROWS][MATRIX_A_COLUMNS],
        matrix_b_t  b[MATRIX_B_ROWS][MATRIX_B_COLUMNS],
        matrix_c_t  c[MATRIX_A_ROWS][MATRIX_B_COLUMNS],
        result_t    result[MATRIX_A_ROWS][MATRIX_B_COLUMNS])
{
    matrix_c_t  intermediate[MATRIX_A_ROWS][MATRIX_B_COLUMNS];

    for(int i = 0; i < MATRIX_A_ROWS; i++)
    {
        for(int j = 0; j < MATRIX_B_COLUMNS; j++)
        {
            intermediate[i][j] = 0;

            for(int k = 0; k < MATRIX_B_ROWS; k++)
                intermediate[i][j] += a[i][k] * b[k][j];

            intermediate[i][j] += c[i][j];

            if(intermediate[i][j] > 128)
                intermediate[i][j] = 128;

            result[i][j] = result_t(intermediate[i][j]);
        }
    }
}
```

Código 2: Implementação de `matrix_operations.h`