

INF01175

Sistemas Digitais para Computadores A

Projeto PC-PO/HLS

Operações com Matrizes 2x2

Pedro Lubaszewski Lima (00341810)
Turma U

27 de agosto de 2024

1 Enunciado do Problema

2 Resolução PC-PO

- Fluxograma ASM
- Parte Operativa
- Parte de Controle
- Validações da Solução

3 Resolução HLS

- Programa em C
- Solução Básica com HLS
- Primeira Solução Otimizada com HLS
- Segunda Solução Otimizada com HLS

4 Comparação das Soluções

1 Enunciado do Problema

2 Resolução PC-PO

- Fluxograma ASM
- Parte Operativa
- Parte de Controle
- Validações da Solução

3 Resolução HLS

- Programa em C
- Solução Básica com HLS
- Primeira Solução Otimizada com HLS
- Segunda Solução Otimizada com HLS

4 Comparação das Soluções

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, comparar a construção manual com a solução HLS do Xilinx Vitis HLS.

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, comparar a construção manual com a solução HLS do Xilinx Vitis HLS.

Função da Saída do Sistema ($R_{2 \times 2}$)

Dadas as matrizes $A_{2 \times 2}$, $B_{2 \times 2}$ e $C_{2 \times 2}$ 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 $F(M_{2 \times 2}) = Q_{2 \times 2}$ tal que

$$q_{ij} = \begin{cases} m_{ij} & \text{se } 0 < m_{ij} \leq 128 \\ 128 & \text{se } m_{ij} > 128 \end{cases}, \forall i, j \in \{1, 2\}, \text{ então}$$

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

1 Enunciado do Problema

2 Resolução PC-PO

- Fluxograma ASM
- Parte Operativa
- Parte de Controle
- Validações da Solução

3 Resolução HLS

- Programa em C
- Solução Básica com HLS
- Primeira Solução Otimizada com HLS
- Segunda Solução Otimizada com HLS

4 Comparação das Soluções

Fluxograma ASM

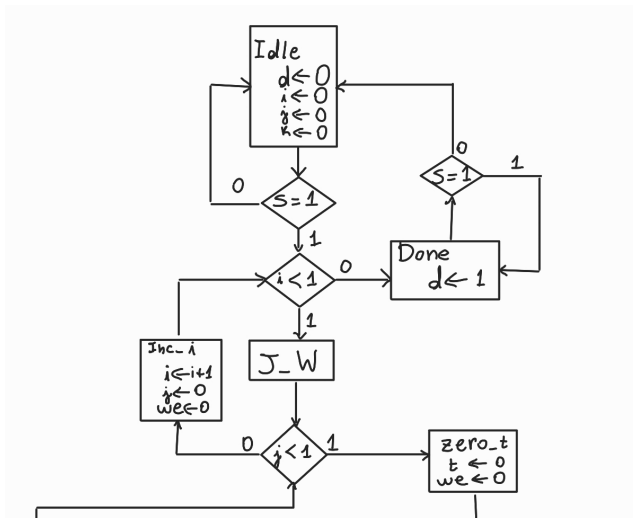


Figura 1: Fluxograma ASM – Parte 1

Fluxograma ASM

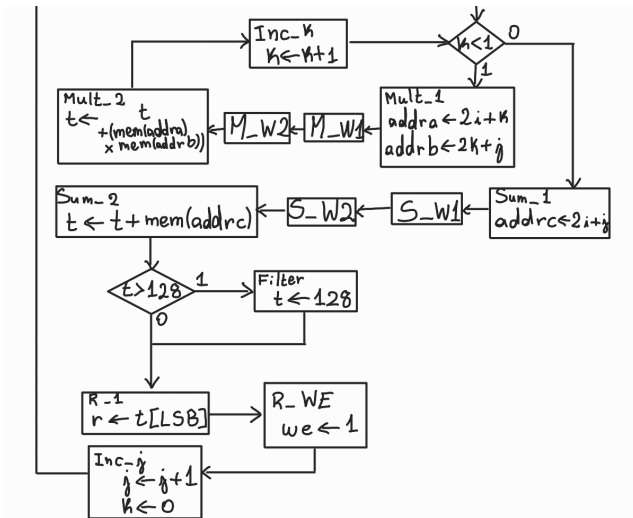


Figura 2: Fluxograma ASM – Parte 2

Datapath

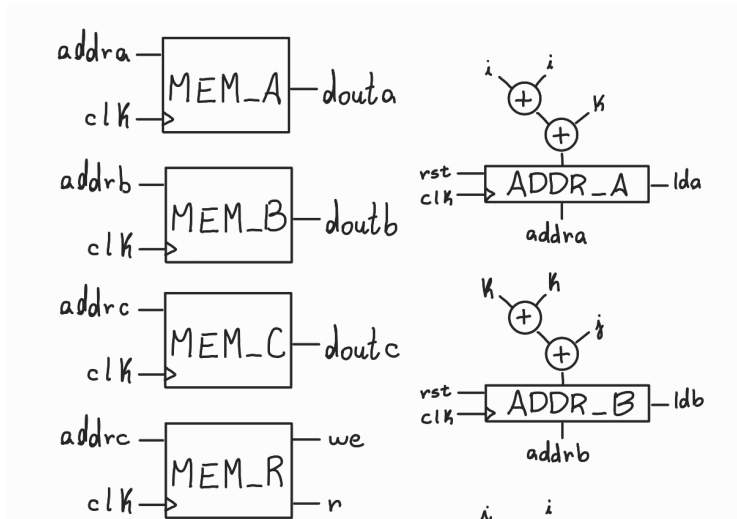


Figura 3: Parte Operativa – Parte 1

Datapath

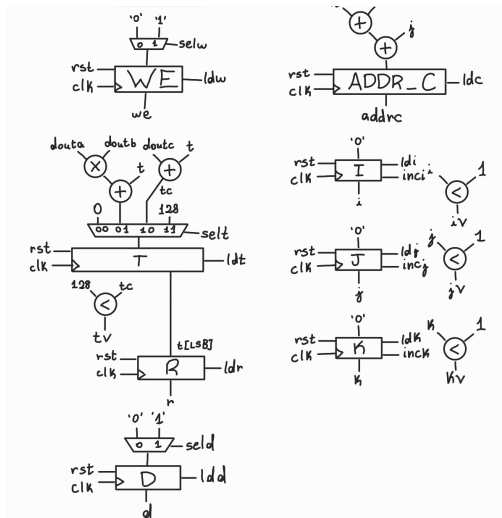


Figura 4: Parte Operativa – Parte 2

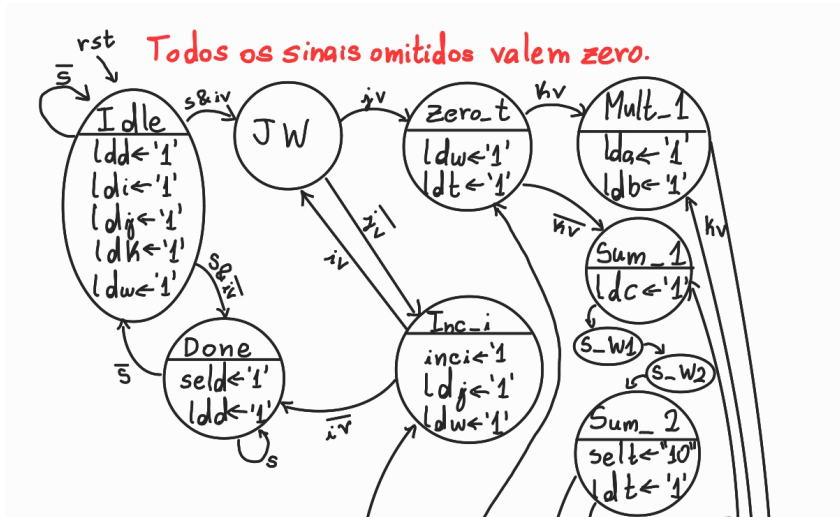


Figura 5: Parte de Controle – Parte 1

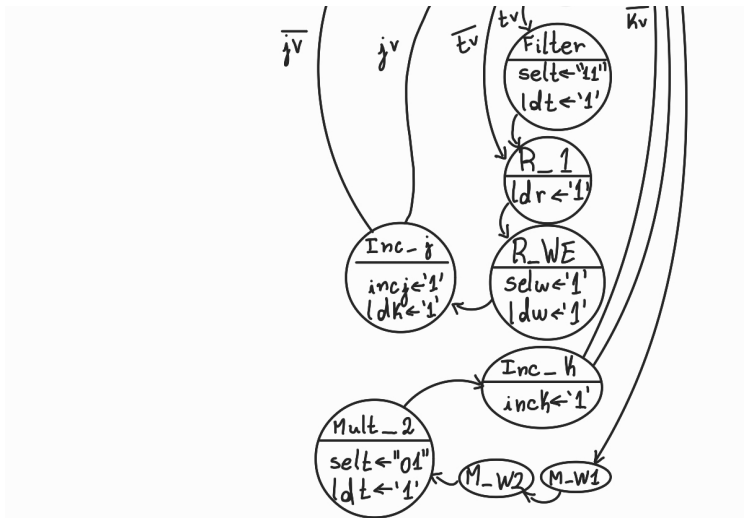


Figura 6: Parte de Controle – Parte 2

Propostas de Testes

Para testar a solução, foram criadas as seguintes entradas:

- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix};$

- $B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix};$

- $B_{NULL} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$

- $C = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}.$

Propostas de Testes

Para testar a solução, foram criadas as seguintes entradas:

- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix};$
- $B = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix};$
- $B_{NULL} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$
- $C = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}.$

Esperando as seguintes saídas:

- $R_1 = \begin{bmatrix} 128 & 15 \\ 70 & 16 \end{bmatrix},$ com A , B e C como entradas;
- $R_2 = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix},$ com A , B_{NULL} e C como entradas.

Simulação para Validação de R_1

Como a saída $doutr = [128, 15, 70, 16]$, no modo da memória *read first*, então foi validado que $R_1 = \begin{bmatrix} 128 & 15 \\ 70 & 16 \end{bmatrix}$.

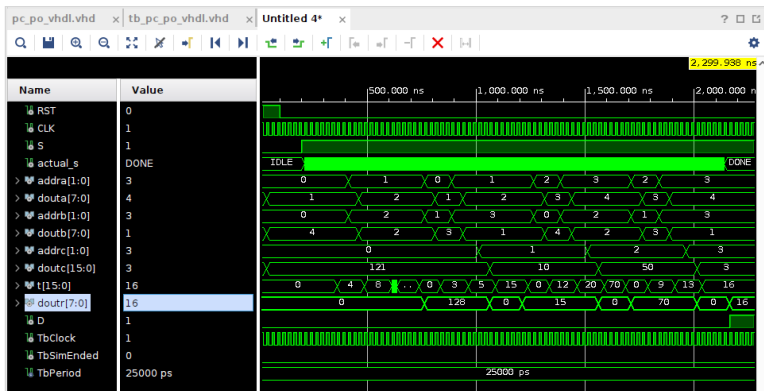


Figura 7: Forma de Onda Funcional com entradas A, B e C

Simulação para Validação de R_2

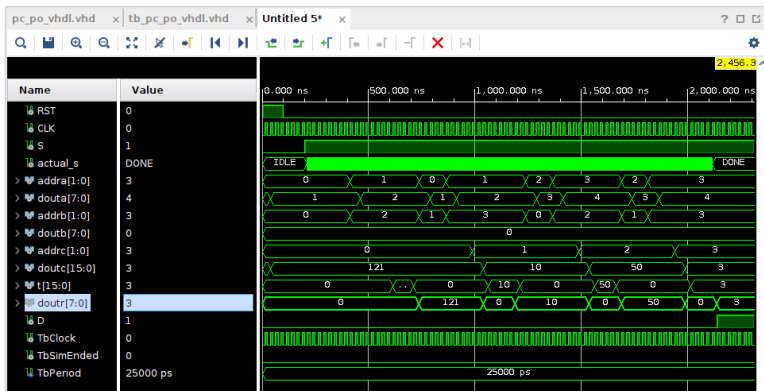


Figura 8: Forma de Onda Funcional com entradas A , B_{NULL} e C

Simulação para Validação de R_2

Como a saída $doutr = [121, 10, 50, 3]$, no modo da memória *read first*, então foi validado que $R_2 = \begin{bmatrix} 121 & 10 \\ 50 & 3 \end{bmatrix}$.

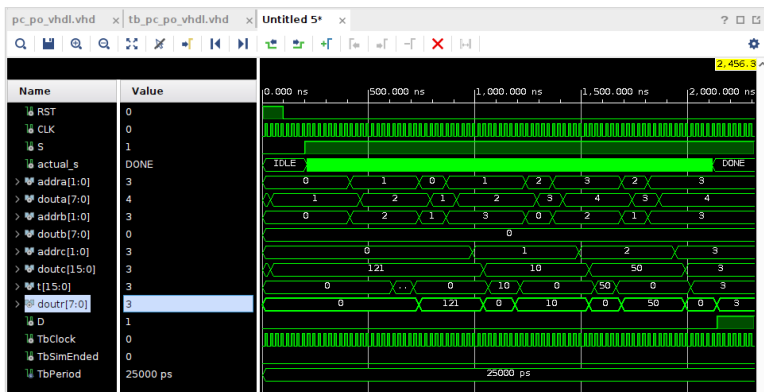


Figura 8: Forma de Onda Funcional com entradas A , B_{NULL} e C

1 Enunciado do Problema

2 Resolução PC-PO

- Fluxograma ASM
- Parte Operativa
- Parte de Controle
- Validações da Solução

3 Resolução HLS

- Programa em C
- Solução Básica com HLS
- Primeira Solução Otimizada com HLS
- Segunda Solução Otimizada com HLS

4 Comparação das Soluções

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

```
1  #include "matrix_operations.h"
2
3  void calculate_matrix (
4      matrix_a_t a[MATRIX_A.ROWS][MATRIX_A.COLUMNS],
5      matrix_b_t b[MATRIX_B.ROWS][MATRIX_B.COLUMNS],
6      matrix_c_t c[MATRIX_A.ROWS][MATRIX_B.COLUMNS],
7      result_t result [MATRIX_A.ROWS][MATRIX_B.COLUMNS])
8  {
9      matrix_c_t intermediate;
10
11     for(int i = 0; i < MATRIX_A.ROWS; i++)
12     {
13         for(int j = 0; j < MATRIX_B.COLUMNS; j++)
14         {
15             intermediate = 0;
16
17             for(int k = 0; k < MATRIX_B.ROWS; k++)
18                 intermediate += a[i][k] * b[k][j];
19
20             intermediate += c[i][j];
21
22             if (intermediate > 128)
23                 intermediate = 128;
24
25             result[i][j] = result_t (intermediate);
26         }
27     }
28 }
```

Código 2: Implementação de matrix_operations.h

Síntese da Solução Básica com HLS

Como solução básica, não se utilizou **nenhuma otimização**.

Síntese da Solução Básica com HLS

Como solução básica, não se utilizou **nenhuma otimização**.

Synthesis Summary(Basic_Solution) x

Synthesis Summary Report of 'calculate_matrix'

General Information

Date: Mon Aug 26 02:14:53 2024

Version: 2023.2.2 (Build 4101106 on Feb 9 2024)

Project: HLS

Solution: Basic_Solution(Vivado IP Flow Target)

Product family: artix7l

Target device: xc7a75t1-ftg256-2L

Timing Estimate

Target	Estimated	Uncertainty
10.00 ns	7.145 ns	2.70 ns

Performance & Resource Estimates ⓘ

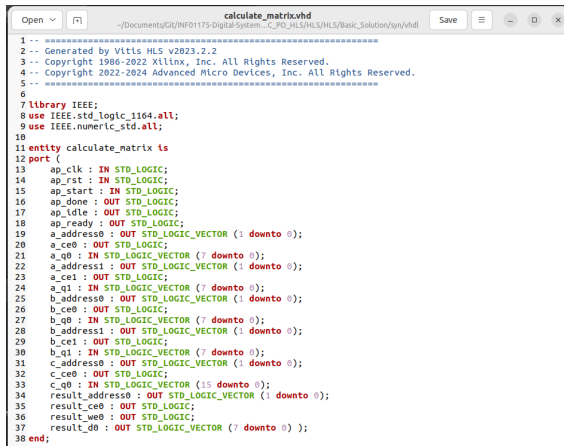
Modules

Loops

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
calculate_matrix				-	10	100.000		-	11	-	no	0	1	161	310

Figura 9: Resultado da Síntese Básica com HLS

Interface da Solução Básica com HLS

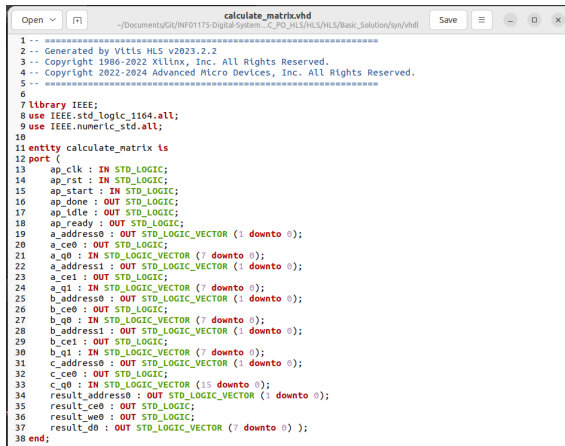


```
1 --  
2 -- Generated by Vitis HLS v2023.2.2  
3 -- Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.  
4 -- Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.  
5 --  
6  
7 library IEEE;  
8 use IEEE.std_logic_1164.all;  
9 use IEEE.numeric_std.all;  
10  
11 entity calculate_matrix is  
12 port (  
13     ap_clk : IN STD_LOGIC;  
14     ap_rst : IN STD_LOGIC;  
15     ap_start : IN STD_LOGIC;  
16     ap_done : OUT STD_LOGIC;  
17     ap_idle : OUT STD_LOGIC;  
18     ap_ready : OUT STD_LOGIC;  
19     a_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
20     a_ce0 : OUT STD_LOGIC;  
21     a_q0 : IN STD_LOGIC_VECTOR (7 downto 0);  
22     a_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);  
23     a_ce1 : OUT STD_LOGIC;  
24     a_q1 : IN STD_LOGIC_VECTOR (7 downto 0);  
25     b_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
26     b_ce0 : OUT STD_LOGIC;  
27     b_q0 : IN STD_LOGIC_VECTOR (7 downto 0);  
28     b_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);  
29     b_ce1 : OUT STD_LOGIC;  
30     b_q1 : IN STD_LOGIC_VECTOR (7 downto 0);  
31     c_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
32     c_ce0 : OUT STD_LOGIC;  
33     c_q0 : IN STD_LOGIC_VECTOR (15 downto 0);  
34     result_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
35     result_ce0 : OUT STD_LOGIC;  
36     result_we0 : OUT STD_LOGIC;  
37     result_d0 : OUT STD_LOGIC_VECTOR (7 downto 0) );  
38 end;  
39
```

Figura 10: Interface da Solução Básica com HLS

Interface da Solução Básica com HLS

Percebe-se um total de **12 IOBs**, totalizando **68 *bits*** de interface de dados.



```
1 --  
2 -- Generated by Vitis HLS v2023.2.2  
3 -- Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.  
4 -- Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.  
5 --  
6  
7 library IEEE;  
8 use IEEE.std_logic_1164.all;  
9 use IEEE.numeric_std.all;  
10  
11 entity calculate_matrix is  
12 port (  
13     ap_clk : IN STD_LOGIC;  
14     ap_rst : IN STD_LOGIC;  
15     ap_start : IN STD_LOGIC;  
16     ap_done : OUT STD_LOGIC;  
17     ap_idle : OUT STD_LOGIC;  
18     ap_ready : OUT STD_LOGIC;  
19     a_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
20     a_ce0 : OUT STD_LOGIC;  
21     a_q0 : IN STD_LOGIC_VECTOR (7 downto 0);  
22     a_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);  
23     a_ce1 : OUT STD_LOGIC;  
24     a_q1 : IN STD_LOGIC_VECTOR (7 downto 0);  
25     b_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
26     b_ce0 : OUT STD_LOGIC;  
27     b_q0 : IN STD_LOGIC_VECTOR (7 downto 0);  
28     b_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);  
29     b_ce1 : OUT STD_LOGIC;  
30     b_q1 : IN STD_LOGIC_VECTOR (7 downto 0);  
31     c_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
32     c_ce0 : OUT STD_LOGIC;  
33     c_q0 : IN STD_LOGIC_VECTOR (15 downto 0);  
34     result_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);  
35     result_ce0 : OUT STD_LOGIC;  
36     result_we0 : OUT STD_LOGIC;  
37     result_d0 : OUT STD_LOGIC_VECTOR (7 downto 0) );  
38 end;  
39
```

Figura 10: Interface da Solução Básica com HLS

Síntese da Primeira Solução Otimizada com HLS

Como primeira solução otimizada, utilizou-se `loop unroll` em cada um dos *loops* da aplicação, buscando aumentar o paralelismo da solução.

Síntese da Primeira Solução Otimizada com HLS

Como primeira solução otimizada, utilizou-se `loop unroll` em cada um dos *loops* da aplicação, buscando aumentar o paralelismo da solução.

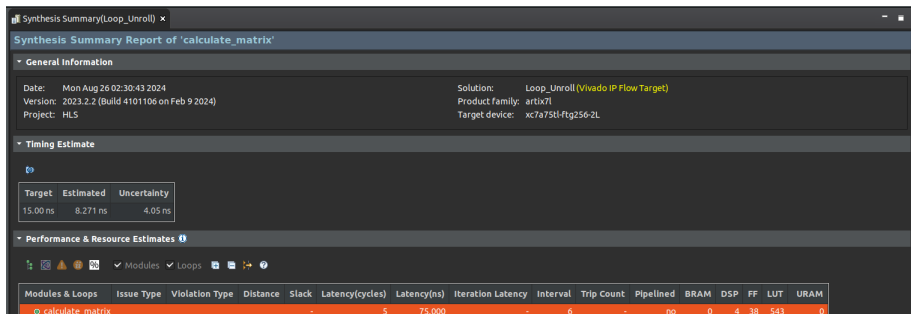
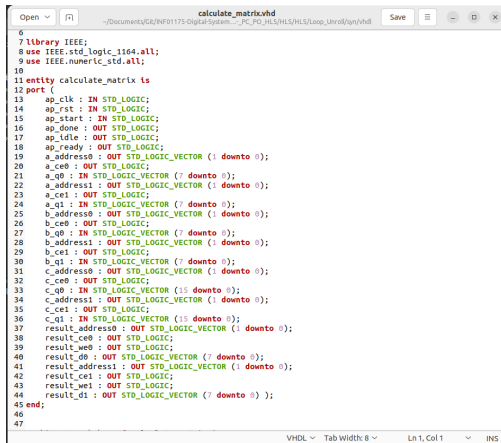


Figura 11: Resultado da Síntese da Solução Otimizada 1 com HLS

Interface da Primeira Solução Otimizada com HLS

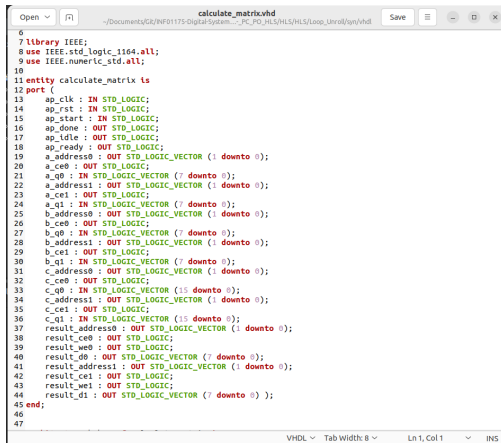


```
6
7 library IEEE;
8 use IEEE.std_logic_1164.all;
9 use IEEE.numeric_std.all;
10
11 entity calculate_matrix is
12 port (
13   ap_clk : IN STD_LOGIC;
14   ap_rst : IN STD_LOGIC;
15   ap_start : IN STD_LOGIC;
16   ap_done : OUT STD_LOGIC;
17   ap_idle : OUT STD_LOGIC;
18   ap_ready : OUT STD_LOGIC;
19   a_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
20   a_ceb : OUT STD_LOGIC;
21   a_q0 : IN STD_LOGIC_VECTOR (7 downto 0);
22   a_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
23   a_cel : OUT STD_LOGIC;
24   a_q1 : IN STD_LOGIC_VECTOR (7 downto 0);
25   b_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
26   b_ceb : OUT STD_LOGIC;
27   b_q0 : IN STD_LOGIC_VECTOR (7 downto 0);
28   b_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
29   b_cel : OUT STD_LOGIC;
30   b_q1 : IN STD_LOGIC_VECTOR (7 downto 0);
31   c_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
32   c_ceb : OUT STD_LOGIC;
33   c_q0 : IN STD_LOGIC_VECTOR (15 downto 0);
34   c_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
35   c_cel : OUT STD_LOGIC;
36   c_q1 : IN STD_LOGIC_VECTOR (15 downto 0);
37   result_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
38   result_ceb : OUT STD_LOGIC;
39   result_we0 : OUT STD_LOGIC;
40   result_d0 : OUT STD_LOGIC_VECTOR (7 downto 0);
41   result_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
42   result_cel : OUT STD_LOGIC;
43   result_we1 : OUT STD_LOGIC;
44   result_d1 : OUT STD_LOGIC_VECTOR (7 downto 0) );
45 end;
46
47
```

Figura 12: Interface da Solução Otimizada 1 com HLS

Interface da Primeira Solução Otimizada com HLS

Percebe-se um total de **16 IOBs**, totalizando **96 bits** de interface de dados.



```
6
7 library IEEE;
8 use IEEE.std_logic_1164.all;
9 use IEEE.numeric_std.all;
10
11 entity calculate_matrix is
12 port (
13     ap_clk : IN STD_LOGIC;
14     ap_rst : IN STD_LOGIC;
15     ap_start : IN STD_LOGIC;
16     ap_done : OUT STD_LOGIC;
17     ap_idle : OUT STD_LOGIC;
18     ap_ready : OUT STD_LOGIC;
19     a_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
20     a_ce0 : OUT STD_LOGIC;
21     a_q0 : IN STD_LOGIC_VECTOR (7 downto 0);
22     a_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
23     a_ce1 : OUT STD_LOGIC;
24     a_q1 : IN STD_LOGIC_VECTOR (7 downto 0);
25     b_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
26     b_ce0 : OUT STD_LOGIC;
27     b_q0 : IN STD_LOGIC_VECTOR (7 downto 0);
28     b_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
29     b_ce1 : OUT STD_LOGIC;
30     b_q1 : IN STD_LOGIC_VECTOR (7 downto 0);
31     c_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
32     c_ce0 : OUT STD_LOGIC;
33     c_q0 : IN STD_LOGIC_VECTOR (15 downto 0);
34     c_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
35     c_ce1 : OUT STD_LOGIC;
36     c_q1 : IN STD_LOGIC_VECTOR (15 downto 0);
37     result_address0 : OUT STD_LOGIC_VECTOR (1 downto 0);
38     result_ce0 : OUT STD_LOGIC;
39     result_we0 : OUT STD_LOGIC;
40     result_d0 : OUT STD_LOGIC_VECTOR (7 downto 0);
41     result_address1 : OUT STD_LOGIC_VECTOR (1 downto 0);
42     result_ce1 : OUT STD_LOGIC;
43     result_we1 : OUT STD_LOGIC;
44     result_d1 : OUT STD_LOGIC_VECTOR (7 downto 0) );
45 end;
46
47
```

Figura 12: Interface da Solução Otimizada 1 com HLS

Síntese da Segunda Solução Otimizada com HLS

Além das otimizações da versão anterior (`loop unroll`), acrescentou-se **array partitioning** com dimensão zero em todas as entradas e saídas do sistema. Isso foi feito em busca do maior nível de paralelismo possível para este circuito.

Síntese da Segunda Solução Otimizada com HLS

Além das otimizações da versão anterior (`loop unroll`), acrescentou-se **array partitioning** com dimensão zero em todas as entradas e saídas do sistema. Isso foi feito em busca do maior nível de paralelismo possível para este circuito.

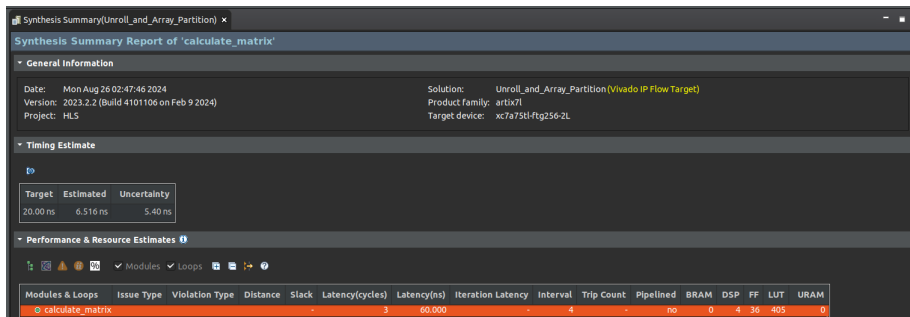
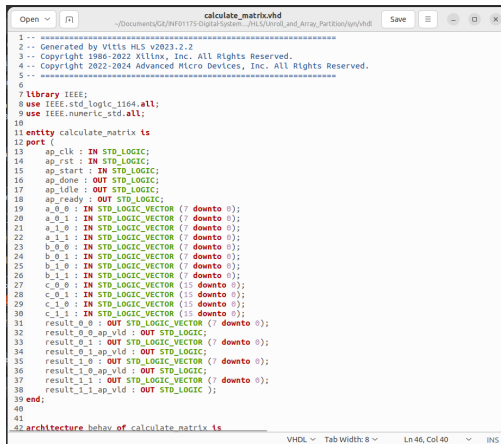


Figura 13: Resultado da Síntese da Solução Otimizada 2 com HLS

Interface da Segunda Solução Otimizada com HLS

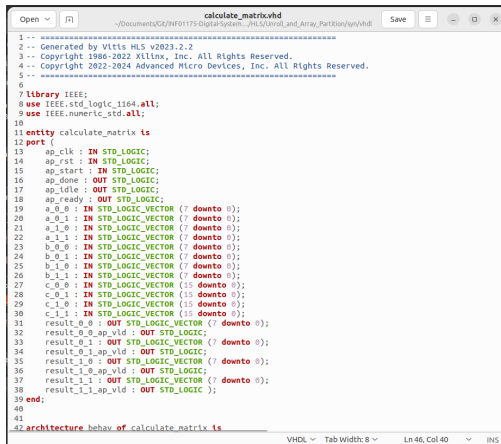


```
1 --
2 -- Generated by Vitis HLS v2023.2.2
3 -- Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
4 -- Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.
5 --
6
7 library IEEE;
8 use IEEE.std_logic_1164.all;
9 use IEEE.numeric_std.all;
10
11 entity calculate_matrix is
12 port (
13   ap_clk : IN STD_LOGIC;
14   ap_rst : IN STD_LOGIC;
15   ap_start : IN STD_LOGIC;
16   ap_done : OUT STD_LOGIC;
17   ap_idle : OUT STD_LOGIC;
18   ap_ready : OUT STD_LOGIC;
19   a_0_0 : IN STD_LOGIC_VECTOR (7 downto 0);
20   a_0_1 : IN STD_LOGIC_VECTOR (7 downto 0);
21   a_1_0 : IN STD_LOGIC_VECTOR (7 downto 0);
22   a_1_1 : IN STD_LOGIC_VECTOR (7 downto 0);
23   b_0_0 : IN STD_LOGIC_VECTOR (7 downto 0);
24   b_0_1 : IN STD_LOGIC_VECTOR (7 downto 0);
25   b_1_0 : IN STD_LOGIC_VECTOR (7 downto 0);
26   b_1_1 : IN STD_LOGIC_VECTOR (7 downto 0);
27   c_0_0 : IN STD_LOGIC_VECTOR (15 downto 0);
28   c_0_1 : IN STD_LOGIC_VECTOR (15 downto 0);
29   c_1_0 : IN STD_LOGIC_VECTOR (15 downto 0);
30   c_1_1 : IN STD_LOGIC_VECTOR (15 downto 0);
31   result_0_0 : OUT STD_LOGIC_VECTOR (7 downto 0);
32   result_0_0_ap_vld : OUT STD_LOGIC;
33   result_0_1 : OUT STD_LOGIC_VECTOR (7 downto 0);
34   result_0_1_ap_vld : OUT STD_LOGIC;
35   result_1_0 : OUT STD_LOGIC_VECTOR (7 downto 0);
36   result_1_0_ap_vld : OUT STD_LOGIC;
37   result_1_1 : OUT STD_LOGIC_VECTOR (7 downto 0);
38   result_1_1_ap_vld : OUT STD_LOGIC;
39 end;
40
41
42 architecture behav of calculate_matrix is
43
```

Figura 14: Interface da Solução Otimizada 2 com HLS

Interface da Segunda Solução Otimizada com HLS

Percebe-se um total de **16 IOBs**, totalizando **160 bits** de interface de dados.



```
1 --
2 -- Generated by Vitis HLS v2023.2.2
3 -- Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
4 -- Copyright 2022-2024 Advanced Micro Devices, Inc. All Rights Reserved.
5 --
6
7 library IEEE;
8 use IEEE.std_logic_1164.all;
9 use IEEE.numeric_std.all;
10
11 entity calculate_matrix is
12 port (
13     ap_clk : IN STD_LOGIC;
14     ap_rst : IN STD_LOGIC;
15     ap_start : IN STD_LOGIC;
16     ap_done : OUT STD_LOGIC;
17     ap_idle : OUT STD_LOGIC;
18     ap_ready : OUT STD_LOGIC;
19     a_0_0 : IN STD_LOGIC_VECTOR (7 downto 0);
20     a_0_1 : IN STD_LOGIC_VECTOR (7 downto 0);
21     a_1_0 : IN STD_LOGIC_VECTOR (7 downto 0);
22     a_1_1 : IN STD_LOGIC_VECTOR (7 downto 0);
23     b_0_0 : IN STD_LOGIC_VECTOR (7 downto 0);
24     b_0_1 : IN STD_LOGIC_VECTOR (7 downto 0);
25     b_1_0 : IN STD_LOGIC_VECTOR (7 downto 0);
26     b_1_1 : IN STD_LOGIC_VECTOR (7 downto 0);
27     c_0_0 : IN STD_LOGIC_VECTOR (15 downto 0);
28     c_0_1 : IN STD_LOGIC_VECTOR (15 downto 0);
29     c_1_0 : IN STD_LOGIC_VECTOR (15 downto 0);
30     c_1_1 : IN STD_LOGIC_VECTOR (15 downto 0);
31     result_0_0 : OUT STD_LOGIC_VECTOR (7 downto 0);
32     result_0_0_ap_vld : OUT STD_LOGIC;
33     result_0_1 : OUT STD_LOGIC_VECTOR (7 downto 0);
34     result_0_1_ap_vld : OUT STD_LOGIC;
35     result_1_0 : OUT STD_LOGIC_VECTOR (7 downto 0);
36     result_1_0_ap_vld : OUT STD_LOGIC;
37     result_1_1 : OUT STD_LOGIC_VECTOR (7 downto 0);
38     result_1_1_ap_vld : OUT STD_LOGIC;
39 end;
40
41
42 architecture behav of calculate_matrix is
```

Figura 14: Interface da Solução Otimizada 2 com HLS

1 Enunciado do Problema

2 Resolução PC-PO

- Fluxograma ASM
- Parte Operativa
- Parte de Controle
- Validações da Solução

3 Resolução HLS

- Programa em C
- Solução Básica com HLS
- Primeira Solução Otimizada com HLS
- Segunda Solução Otimizada com HLS

4 Comparação das Soluções

Comparação do PC-PO com HLS

Ao construir a solução partindo do **Fluxograma ASM**, da **Parte Operativa** e da **Parte de Controle**, obteve-se um circuito com:

- 174 LUTs;
- 43 flip flops;
- 0 DSPs;
- 4 BRAMs;
- Latência de 76 ciclos de *clock*;
- Interface de dados com 4 IOBs, 40 bits.

Comparação do PC-PO com HLS

Ao construir a solução partindo do **Fluxograma ASM**, da **Parte Operativa** e da **Parte de Controle**, obteve-se um circuito com:

- 174 LUTs;
- 43 flip flops;
- 0 DSPs;
- 4 BRAMs;
- Latência de 76 ciclos de *clock*;
- Interface de dados com 4 IOBs, 40 bits.

Através da melhor solução com **HLS**, obteve-se os seguintes resultados:

- 405 LUTs;
- 36 flip flops;
- 4 DSPs;
- 0 BRAMs;
- Latência de 3 ciclos de *clock*;
- Interface de dados com 16 IOBs, 160 bits.

Comparação do PC-PO com HLS

Ao construir a solução partindo do **Fluxograma ASM**, da **Parte Operativa** e da **Parte de Controle**, obteve-se um circuito com:

- 174 LUTs;
- 43 flip flops;
- 0 DSPs;
- 4 BRAMs;
- Latência de 76 ciclos de clock;
- Interface de dados com 4 IOBs, 40 bits.

Através da melhor solução com **HLS**, obteve-se os seguintes resultados:

- 405 LUTs;
- 36 flip flops;
- 4 DSPs;
- 0 BRAMs;
- Latência de 3 ciclos de clock;
- Interface de dados com 16 IOBs, 160 bits.

De forma geral, a solução HLS tem um **paralelismo** melhor que a PC-PO. No entanto, a solução PC-PO apresenta uma **melhor utilização de recursos** (LUTs em particular) e uma **interface mais compacta**.

Obrigado pela atenção!
Alguma dúvida?