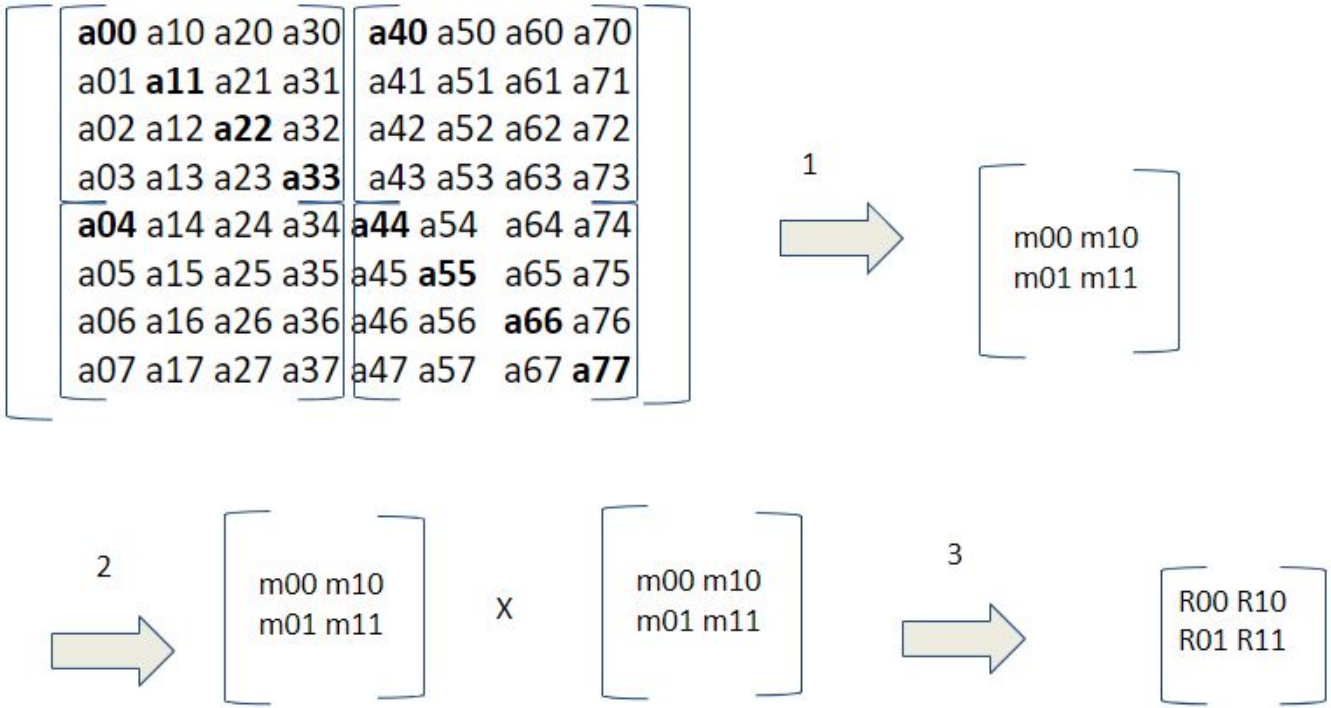
Explorando o paralelismo do Hardware: uma comparação entre algoritmo PCxPO e HLS

Henry Ribeiro Piceni (00333363)
Júlia Mombach da Silva (00281023)

Sistemas Digitais para Computadores 2023/1

Proposta


a00 a10 a20 a30 a40 a50 a60 a70
a01 a11 a21 a31 a41 a51 a61 a71
a02 a12 a22 a32 a42 a52 a62 a72
a03 a13 a23 a33 a43 a53 a63 a73
a04 a14 a24 a34 a44 a54 a64 a74
a05 a15 a25 a35 a45 a55 a65 a75
a06 a16 a26 a36 a46 a56 a66 a76
a07 a17 a27 a37 a47 a57 a67 a77
1
m00 m10
m01 m11
2
m00 m10
m01 m11
X
m00 m10
m01 m11
3
R00 R10
R01 R11

HLS

Código em C++

matrizes.h > ...

```
#ifndef __MATRIXMUL_H__
#define __MATRIXMUL_H__

#include <cmath>
using namespace std;



#define MAT_A_ROWS 8
#define MAT_A_COLS 8
#define MAT_B_ROWS 2
#define MAT_B_COLS 2
#define MAT_R_ROWS 2
#define MAT_R_COLS 2

typedef signed char mat_a_t; //8bits
typedef signed char mat_b_t; //8bits
typedef signed short result_t; //16bits

// Prototype of top level function for C-synthesis
void matrizes(
      mat_a_t a[MAT_A_ROWS][MAT_A_COLS],
      mat_b_t b[MAT_B_ROWS][MAT_B_COLS],
      result_t res[MAT_R_ROWS][MAT_R_COLS]);

#endif
```

Para o desenvolvimento do HLS, foram elaborados 4 arquivos em C++: um header, um programa principal e dois testbenches. Cada testbench é referente a uma matriz, sendo a primeira a de escolha da dupla e a segunda a solicitada pela professora.

Código em C++

```
8   //Media e bota na matriz 2x2----------
9
10    for(int i = 0; i < 4; i++){
11        for(int j = 0; j < 4; j++){
12            mat2[0][0] += mat[i][j];
13        }
14    }
15    mat2[0][0] /= 16;
16
17    for(int i = 0; i < 4; i++){
18        for(int j = 4; j < 8; j++){
19            mat2[0][1] += mat[i][j];
20        }
21    }
22    mat2[0][1] /= 16;
23
24    for(int i = 4; i < 8; i++){
25        for(int j = 0; j < 4; j++){
26            mat2[1][0] += mat[i][j];
27        }
28    }
29    mat2[1][0] /= 16;
30
31    for(int i = 4; i < 8; i++){
32        for(int j = 4; j < 8; j++){
33            mat2[1][1] += mat[i][j];
34        }
35    }
36    mat2[1][1] /= 16;
```

```
38    //Multiplicacao da matriz por ela mesma----------------------
39
40    mat3[0][0] = mat2[0][0]*mat2[0][0] + mat2[0][1]*mat2[1][0];
41    mat3[0][1] = mat2[0][0]*mat2[0][1] + mat2[0][1]*mat2[1][1];
42    mat3[1][0] = mat2[1][0]*mat2[0][0] + mat2[1][1]*mat2[1][0];
43    mat3[1][1] = mat2[0][1]*mat2[1][0] + mat2[1][1]*mat2[1][1];
```

O código principal possui apenas uma função que resolve todo o problema solicitado. A matriz 1 é dividida em 4 partes e a média de cada parte é calculada gerando uma matriz 2x2. Então essa matriz é multiplicada por ela mesma, resultando na matriz 3.

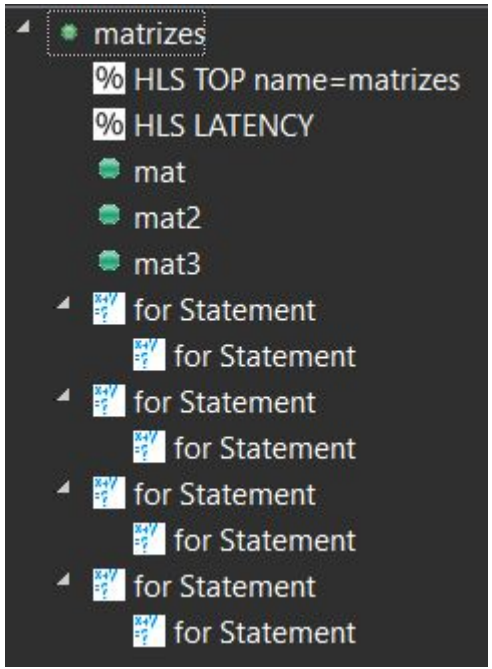
Código em C++

```
8       mat_a_t in_mat_a[8][8] = {
9           {1,1,1,1,2,2,2,2},
10          {1,1,1,1,2,2,2,2},
11          {1,1,1,1,2,2,2,2},
12          {1,1,1,1,2,2,2,2},
13          {3,3,3,3,4,4,4,4},
14          {3,3,3,3,4,4,4,4},
15          {3,3,3,3,4,4,4,4},
16          {3,3,3,3,4,4,4,4}
17      };
18
19      mat_b_t in_mat_b[2][2];
20
21      result_t hw_result[2][2], sw_result[2][2];
22      int err_cnt = 0;
```

```
6   int main(int argc, char **argv)
7   {
8       mat_a_t in_mat_a[8][8] = {
9           {1,2,3,4,5,5,5,5},
10          {-1,-2,-3,-4,-5,-6,-7,-8},
11          {1,2,3,4,5,5,5,5},
12          {-1,-2,-3,-4,-5,-6,-7,-8},
13          {1,2,3,4,5,6,7,8},
14          {-2,-2,-2,-2,-2,-2,-2,-2},
15          {1,2,3,4,5,6,7,8},
16          {-2,-2,-2,-2,-2,-2,-2,-2}
17      };
18
19      mat_b_t in_mat_b[2][2];
20
21      result_t hw_result[2][2], sw_result[2][2];
22      int err_cnt = 0;
```

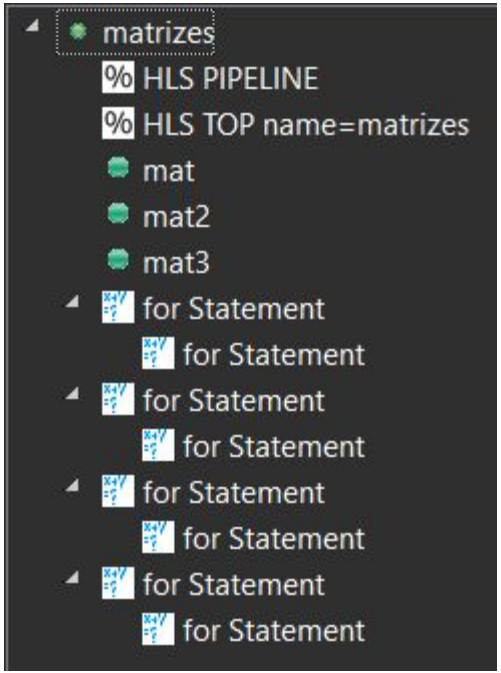
Otimização

Otimização - Latência



A diretiva de otimização de latência faz com que o programa encontre uma latência que minimize o tempo de execução do algoritmo.

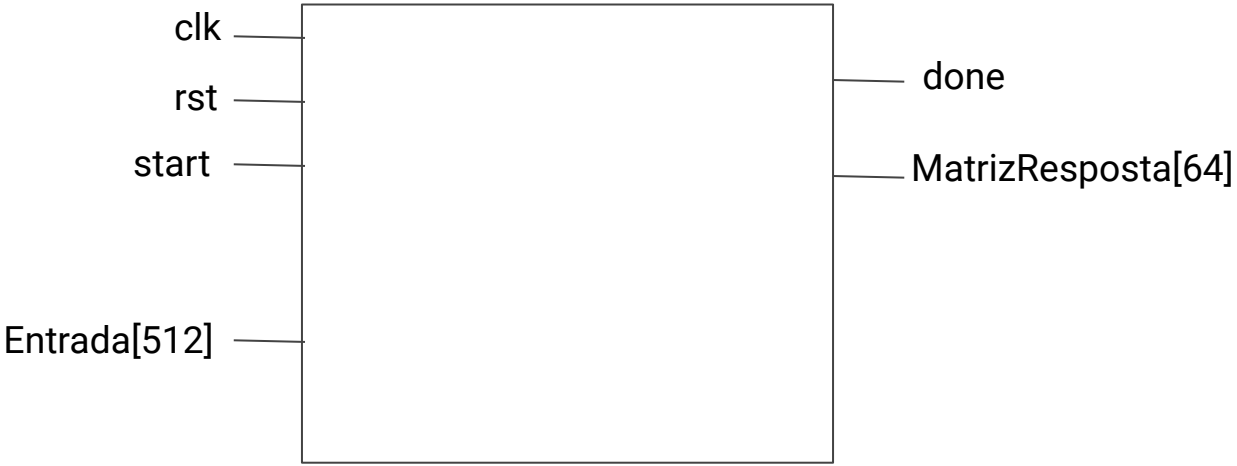
Otimização - Pipeline



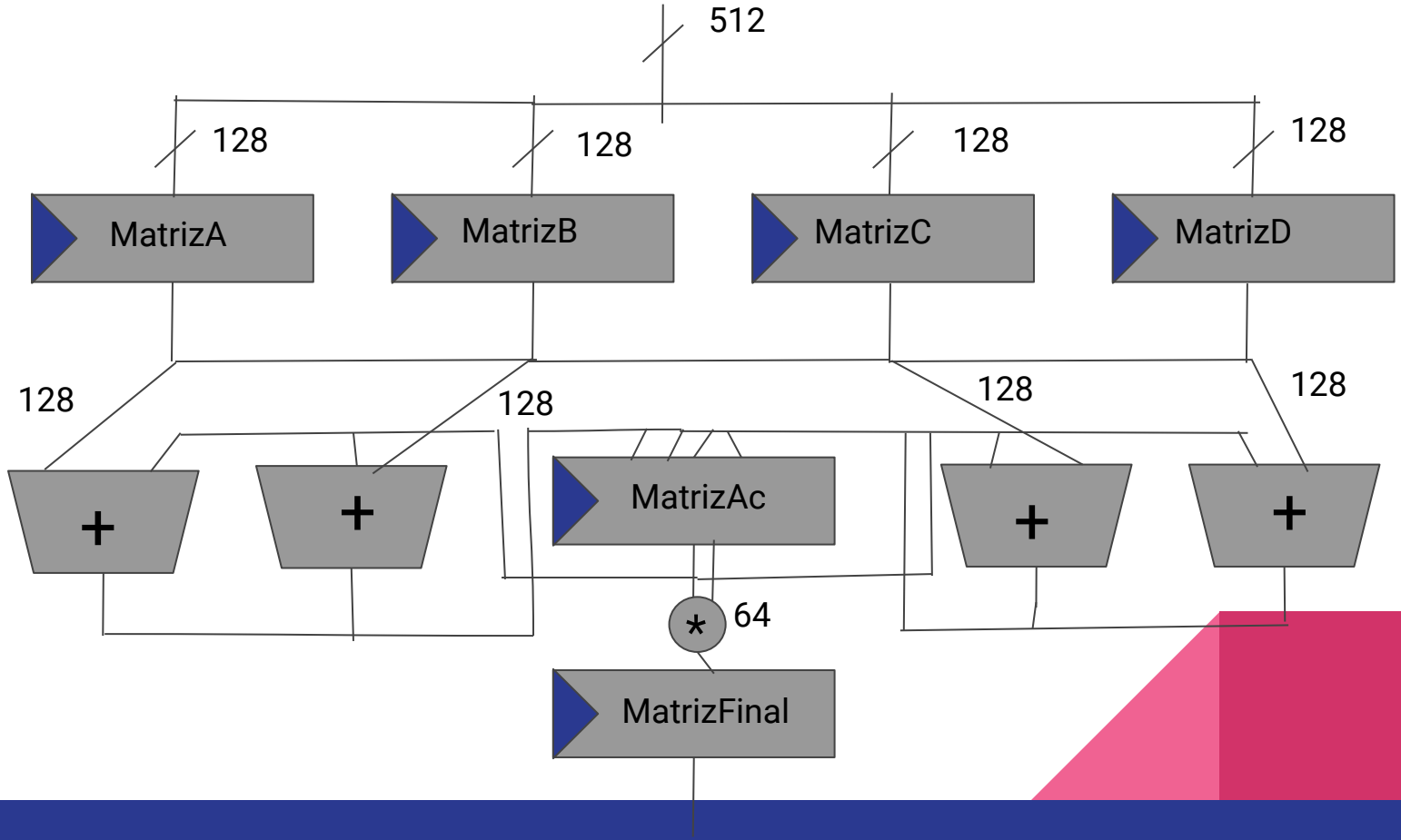
A utilização de pipeline permite a execução simultânea de operações. Fizemos a escolha dessa diretiva para otimização para que as operações do loop sejam executadas de forma concorrente.

Projeto PC-PO

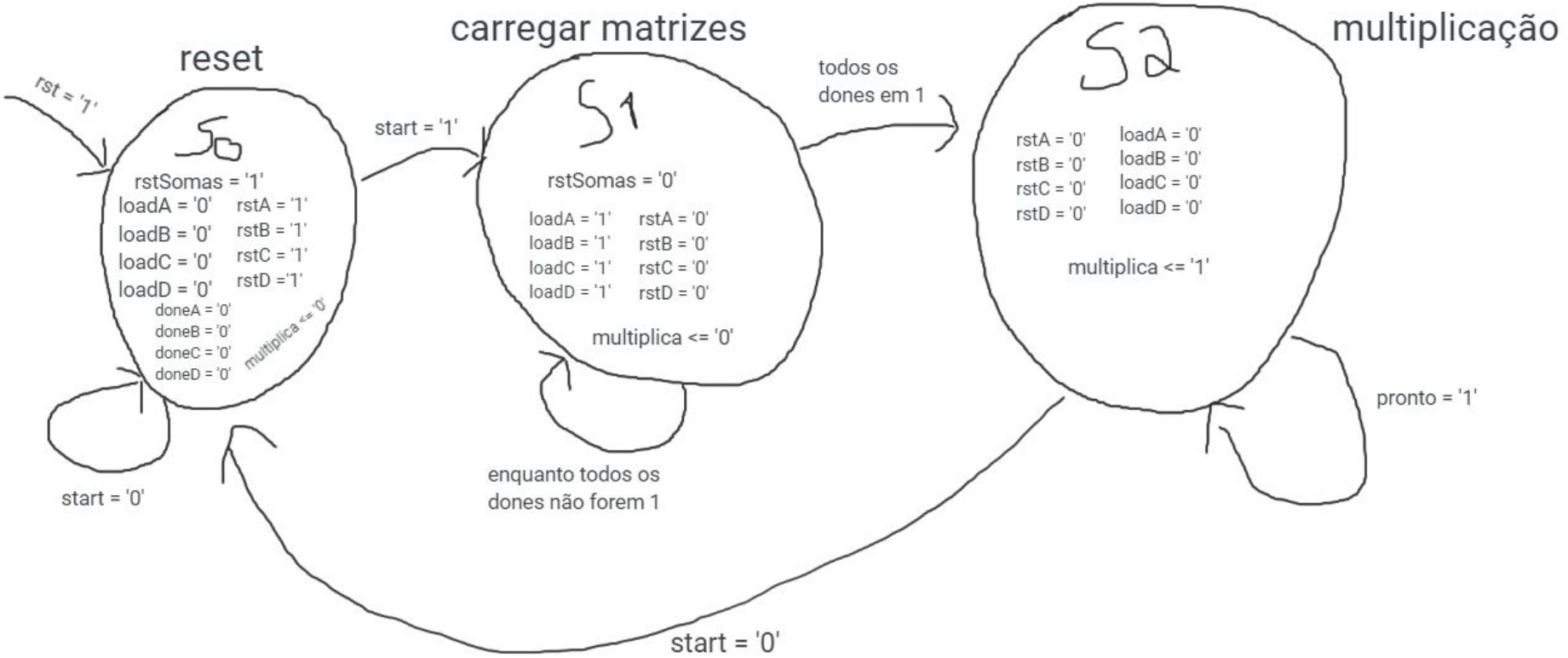
Interface



Parte de Operativa



Parte de Controle



Código PC-PO

```
process(clk, rstA)
    variable acumulador : signed(7 downto 0);
begin
    if rstA = '1' then -- reseta a matriz

        doneA <= false;
        acumulador := (others => '0');
        MatrizAc(0,0) <= (others => '0');

        for i in 0 to 3 loop
            for j in 0 to 3 loop
                MatrizA(i,j) <= (others => '0');
            end loop;
        end loop;

    elsif rising_edge(clk) then
        if loadA = '1' then -- preenche a matriz de entrada
            acumulador := (others => '0');
            for i in 0 to 3 loop
                for j in 0 to 3 loop
                    MatrizA(i,j) <= to_signed(to_integer(unsigned( entrada(8 * (i * 4
+ j) to 8 * (i * 4 + j) + 7))), 8);
                    acumulador := acumulador + MatrizA(i,j);
                end loop;
            end loop;
            MatrizAc(0,0) <= acumulador / 16;
            doneA <= true;
        else -- senão, recebe ela mesma
            for i in 0 to 3 loop
                for j in 0 to 3 loop
                    MatrizA(i,j) <= MatrizA(i,j);
                end loop;
            end loop;
        end if;

    end if;
end process;
```

```
process(clk, rstB)
    variable acumulador : signed(7 downto 0);
begin
    if rstB = '1' then -- reseta a matriz
        doneB <= false;
        acumulador := (others => '0');
        MatrizAc(0,1) <= (others => '0');

        for i in 0 to 3 loop
            for j in 0 to 3 loop
                MatrizB(i,j) <= (others => '0');
            end loop;
        end loop;

    elsif rising_edge(clk) then
        if loadB = '1' then -- preenche a matriz
            acumulador := (others => '0');
            for i in 0 to 3 loop
                for j in 0 to 3 loop
                    --MatrizB(i,j) <= to_signed(to_integer(unsigned( B(8 * (i * 4 +
j) to 8 * (i * 4 + j) + 7))), 8);
                    MatrizB(i,j) <= to_signed(to_integer(unsigned( entrada(8 * (i * 4
+ j) + 128 to 8 * (i * 4 + j) + 135))), 8);
                    acumulador := acumulador + MatrizB(i,j);
                end loop;
            end loop;
            MatrizAc(0,1) <= acumulador / 16;
            doneB <= true;

        else -- senão, recebe ela mesma
            for i in 0 to 3 loop
                for j in 0 to 3 loop
                    MatrizB(i,j) <= MatrizB(i,j);
                end loop;
            end loop;
        end if;

    end if;
```

```
-- Quando tudo estiver pronto, multiplica a matriz "MatrizAc" por ela mesma
process(multiplica)
begin
    if doneA = true and doneB = true and doneC = true and doneD = true then
        MatrizFinal(0,0) <= MatrizAc(0,0) * MatrizAc(0,0) + MatrizAc(1,0) *
MatrizAc(0,1);
        MatrizFinal(0,1) <= MatrizAc(0,0) * MatrizAc(0,1) + MatrizAc(1,1) *
MatrizAc(0,1);
        MatrizFinal(1,0) <= MatrizAc(1,0) * MatrizAc(0,0) + MatrizAc(1,1) *
MatrizAc(1,0);
        MatrizFinal(1,1) <= MatrizAc(0,1) * MatrizAc(1,0) + MatrizAc(1,1) *
MatrizAc(1,1);
        pronto <= '1';
    end if;
end process;

-- LIGANDO OS FIOS DE SAÍDA
done <= pronto;
MatrizResposta(0 to 15) <= STD_LOGIC_VECTOR(MatrizFinal(0,0));
MatrizResposta(16 to 31) <= STD_LOGIC_VECTOR(MatrizFinal(0,1));
MatrizResposta(32 to 47) <= STD_LOGIC_VECTOR(MatrizFinal(1,0));
MatrizResposta(48 to 63) <= STD_LOGIC_VECTOR(MatrizFinal(1,1));
```

Simulações

Matriz 1

Matriz A:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 \end{pmatrix}$$

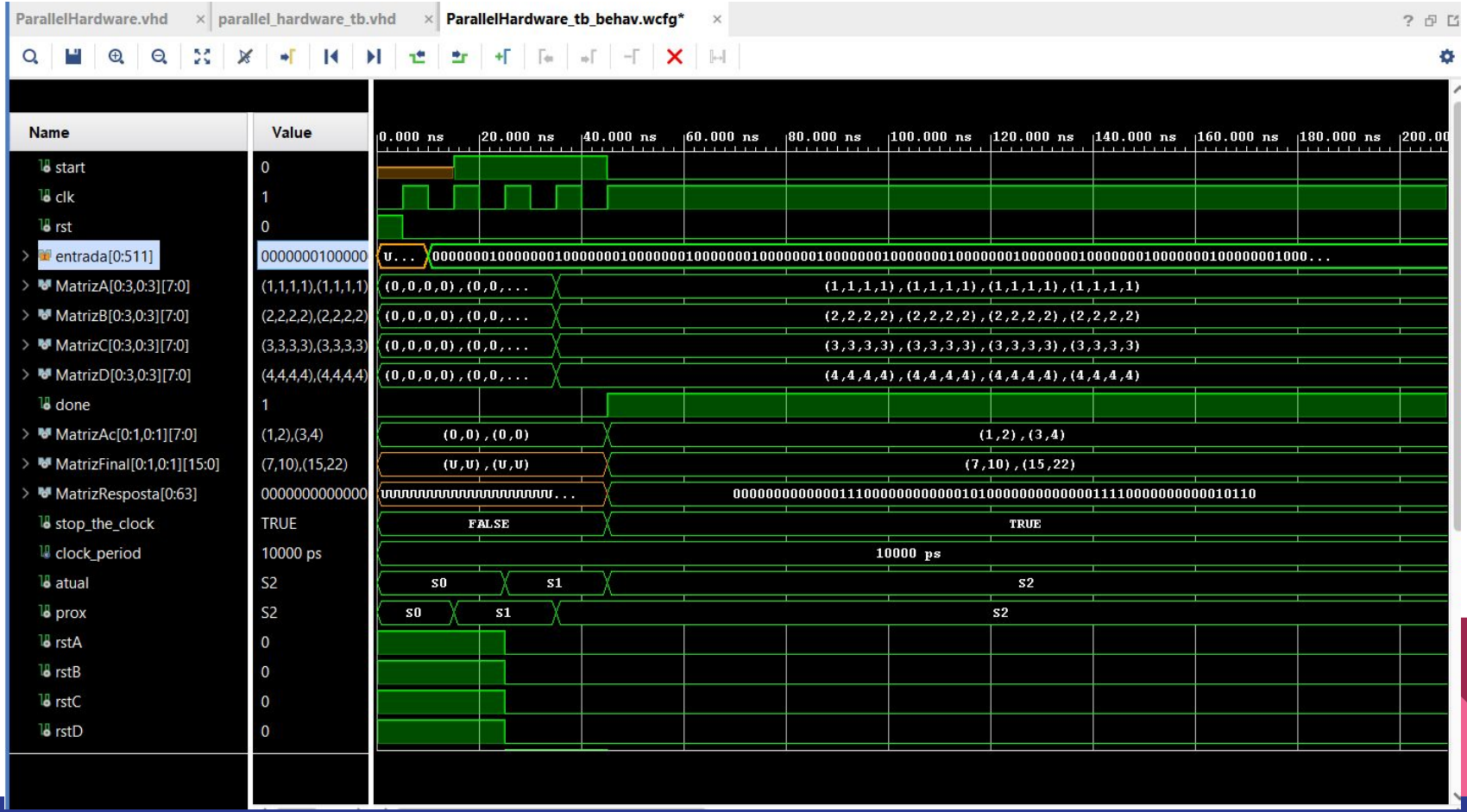
Matriz Ac:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Matriz Final:

$$\begin{pmatrix} 7 & 10 \\ 15 & 22 \end{pmatrix}$$

Waveform


ParallelHardware.vhd
parallel_hardware_tb.vhd
ParallelHardware_tb_behav.wcfg*
Name
Value
start
0
clk
1
rst
0
entrada[0:511]
0000000100000
MatrizA[0:3,0:3][7:0]
(1,1,1,1),(1,1,1,1)
MatrizB[0:3,0:3][7:0]
(2,2,2,2),(2,2,2,2)
MatrizC[0:3,0:3][7:0]
(3,3,3,3),(3,3,3,3)
MatrizD[0:3,0:3][7:0]
(4,4,4,4),(4,4,4,4)
done
1
MatrizAc[0:1,0:1][7:0]
(1,2),(3,4)
MatrizFinal[0:1,0:1][15:0]
(7,10),(15,22)
MatrizResposta[0:63]
0000000000000
stop_the_clock
TRUE
clock_period
10000 ps
atual
S2
prox
S2
rstA
0
rstB
0
rstC
0
rstD
0
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
140.000 ns
160.000 ns
180.000 ns
200.00
FALSE
TRUE
10000 ps
S0
S1
S2

Matriz 2

Matriz A:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\ -1 & -2 & -3 & -4 & -5 & -6 & -7 & -8 \\ 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\ -1 & -2 & -3 & -4 & -5 & -6 & -7 & -8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 \end{pmatrix}$$

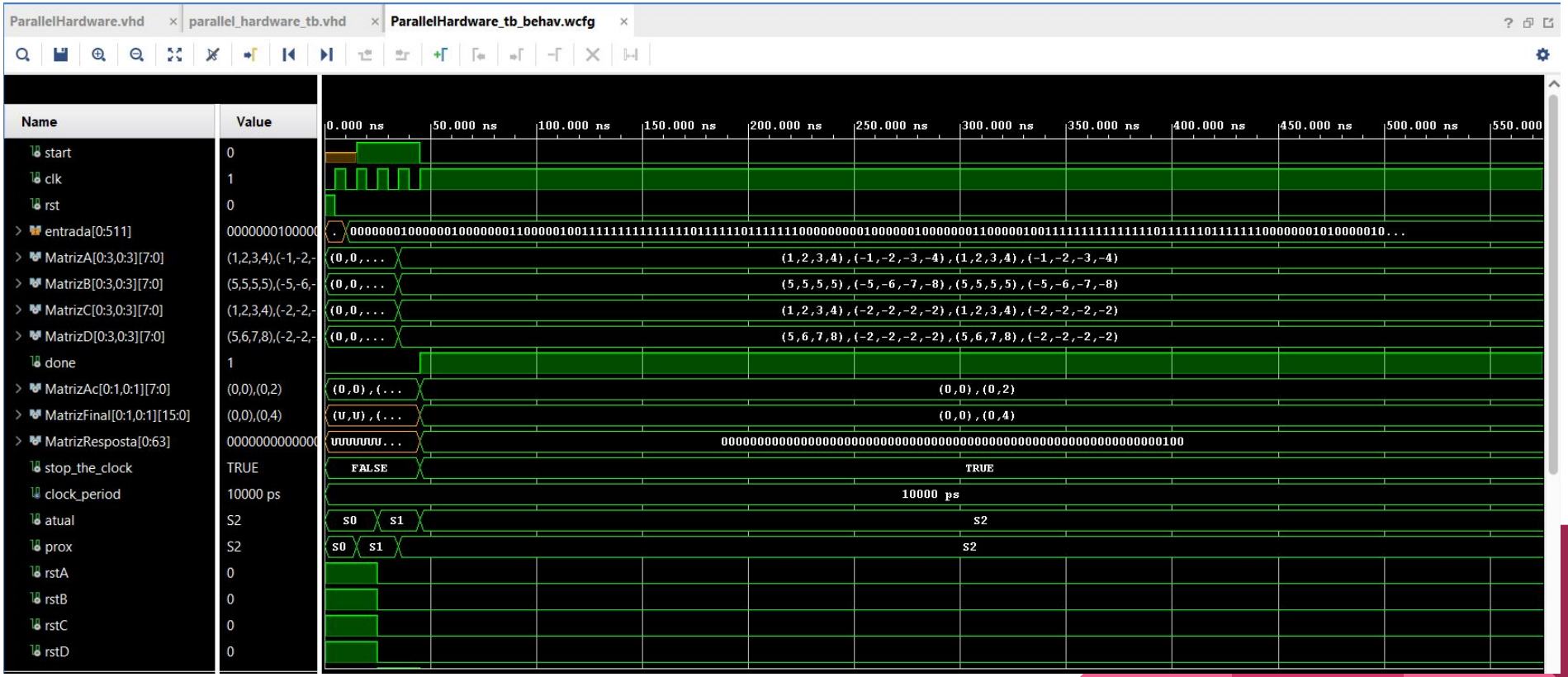
Matriz Ac:

$$\begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$$

Matriz Final:

$$\begin{pmatrix} 0 & 0 \\ 0 & 4 \end{pmatrix}$$

Waveform


ParallelHardware.vhd
parallel_hardware_tb.vhd
ParallelHardware_tb_behav.wcfg
Name
Value
start
0
clk
1
rst
0
entrada[0:511]
MatrizA[0:3,0:3][7:0]
(1,2,3,4),(-1,-2,-
MatrizB[0:3,0:3][7:0]
(5,5,5,5),(-5,-6,-
MatrizC[0:3,0:3][7:0]
(1,2,3,4),(-2,-2,-
MatrizD[0:3,0:3][7:0]
(5,6,7,8),(-2,-2,-
done
1
MatrizAc[0:1,0:1][7:0]
(0,0),(0,2)
MatrizFinal[0:1,0:1][15:0]
(0,0),(0,4)
MatrizResposta[0:63]
stop_the_clock
TRUE
clock_period
10000 ps
atual
S2
prox
S2
rstA
0
rstB
0
rstC
0
rstD
0
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.000 ns
350.000 ns
400.000 ns
450.000 ns
500.000 ns
550.000
FALSE
TRUE
10000 ps
S0
S1
S2

Comparações

Tabela comparativa

HLS

Solução	Clock	Latência	BRAM	DSP	FF	LUT	URAM
Otimização Latência	7,511 ns	840 ns	0	2	374	1283	0
Otimização Pipeline	8,981 ns	360 ns	0	2	167	1169	0

PC-PO

Latência	BRAM	DSP	FF	LUT	URAM
5cc	0	0	531	867	0