UNIVERSIDADE CATÓLICA DE PELOTAS
ENGENHARIA DE COMPUTAÇÃO
Projetos de Circuitos Integrados

Gabriel Harter Zoppo

Desenvolvimento de um filtro FIR totalmente paralelo com e sem pipeline

Pelotas, 30 de abril de 2022

1. Introdução:

Este projeto é referente ao primeiro trabalho da primeira avaliação da disciplina de projeto de circuitos integrados ministrada pelo professor Eduardo Antonio Cesar da Costa e tem como objetivo desenvolver um filtro FIR totalmente paralelo com 4 taps e 4 Bits com e sem registradores pipeline.

O projeto foi totalmente elaborado e compilado na linguagem de descrição de hardware *(VHDL)* e compilada no programa *Quartus II*, um software de design de dispositivo lógico programável e o *modelsim*, um software de simulação dos códigos *VHDL* possibilitando ver os resultados 13.574forma gráfica.

A arquitetura do código utilizado no trabalho é mostrado na figura 1, e nele temos, dois multiplicadores desloca soma de 4 Bits, dois somadores de 8 Bits, três registrador de 4 Bits e quatro registradores de 8 Bits.

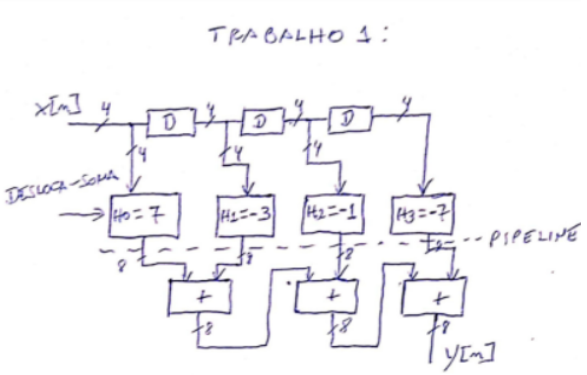

Figura 1: Filtro FIR totalmente paralelo

2. Desenvolvimento:

Para o desenvolvimento do filtro FIR semi-paralelo devem ser declarados primeiramente os elementos utilizados nesse filtro, começa-se pelos dois somadores de 8 Bits, na qual as duas entradas e a saída são de 8 Bits conforme a figura 2.

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3   USE ieee.std_logic_unsigned.all;
4
5   ENTITY somador8bits IS
6   PORT ( a, b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
7          s : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
8   END somador8bits;
9
10  ARCHITECTURE dataflow OF somador8bits IS
11
12  BEGIN
13
14  s <= a + b;
15
16  END dataflow;
```

Figura 2: Somador de 16 Bits

O próximo passo é declarar os multiplicadores desloca soma de 4 Bits, na qual a entrada é de 4 Bits e a saída e de 8 Bits, temos 4 desloca somas diferentes um para cada valor de H conforme é mostrado nas figuras 3.

Esses desloca-somas foram feitos a partir da criação do sinal negativo que recebe o X deslocado algumas posições para esquerda e posteriormente somado ou subtraido por X.

```
19  USE ieee.std_logic_1164.all;
20  USE ieee.std_logic_unsigned.all;
21
22  ENTITY deslocaSoma7 IS
23
24  PORT (E: IN STD_LOGIC_VECTOR (3 downto 0);
25        S: OUT STD_LOGIC_VECTOR (7 downto 0));
26  END deslocaSoma7;
27
28  ARCHITECTURE comportamento OF deslocaSoma7 IS
29  SIGNAL negativo: std_logic_vector(7 downto 0);
30
31  BEGIN
32     negativo <= '0' & E & "000";
33     S <= negativo - E;
34  END comportamento;
```

Figura 3: Multiplicador Desloca-Soma de H = 7

```
37  LIBRARY ieee;
38  USE ieee.std_logic_1164.all;
39  USE ieee.std_logic_unsigned.all;
40
41  ENTITY deslocaSomaN3 IS
42
43  PORT (E: IN STD_LOGIC_VECTOR (3 downto 0);
44        S: OUT STD_LOGIC_VECTOR (7 downto 0));
45  END deslocaSomaN3;
46
47  ARCHITECTURE comportamento OF deslocaSomaN3 IS
48  SIGNAL negativo: std_logic_vector(7 downto 0);
49
50  BEGIN
51     negativo <= "00" & E & "00";
52     S <= E - negativo;
53  END comportamento;
```

Figura 4: Multiplicador Desloca-Soma de H = -3

```
56  LIBRARY ieee;
57  USE ieee.std_logic_1164.all;
58  USE ieee.std_logic_unsigned.all;
59
60  ENTITY deslocaSomaN1 IS
61  PORT (E: IN STD_LOGIC_VECTOR (3 downto 0);
62        S: OUT STD_LOGIC_VECTOR (7 downto 0)
63    );
64  END deslocaSomaN1;
65
66  ARCHITECTURE comportamento OF deslocaSomaN1 IS
67  SIGNAL negativo: std_logic_vector(7 downto 0);
68
69  BEGIN
70     negativo <= "000" & E & '0';
71     S <= E - negativo;
72  END comportamento;
```

Figura 5: Multiplicador Desloca-Soma de H = -1

```
74  ----------------------------------------------------
75  LIBRARY ieee;
76  USE ieee.std_logic_1164.all;
77  USE ieee.std_logic_unsigned.all;
78
79  ENTITY deslocaSomaN7 IS
80
81  PORT (E: IN STD_LOGIC_VECTOR (3 downto 0);
82        S: OUT STD_LOGIC_VECTOR (7 downto 0));
83  END deslocaSomaN7;
84
85  ARCHITECTURE comportamento OF deslocaSomaN7 IS
86  SIGNAL negativo: std_logic_vector(7 downto 0);
87
88  BEGIN
89
90     negativo <= '0' & E & "000";
91     S <= E - negativo;
92  END comportamento;
```

Figura 6: Multiplicador Desloca-Soma de H = -7

Após os multiplicadores desloca-soma temos os dois registradores, um de 8 Bits e um de 4 Bits, conforme a figura 7 e 8.

```
95  LIBRARY ieee;
96  USE ieee.std_logic_1164.all;
97
98  ENTITY reg8b IS
99  PORT ( clk : IN STD_LOGIC;
100       D : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
101       Q : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
102 END reg8b;
103
104 ARCHITECTURE comportamento OF reg8b IS
105 BEGIN
106 PROCESS (clk)
107  BEGIN
108 IF clk'EVENT AND clk = '1' THEN
109  Q <= D;
110 END IF;
111 END PROCESS;
112  END comportamento;
```

Figura 7: Registrador de 8 Bits

```
117  ENTITY reg4b IS
118  PORT ( clk : IN STD_LOGIC;
119         D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
120         Q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
121  END reg4b;
122
123  ARCHITECTURE comportamento OF reg4b IS
124  BEGIN
125  PROCESS (clk)
126  BEGIN
127  IF clk'EVENT AND clk = '1' THEN
128  Q <= D;
129  END IF;
130  END PROCESS;
131  END comportamento;
```

Figura 8: Registrador de 4 Bits

Posteriormente será feito a declaração do My Componentes.vhd que é quando declaramos nossos componentes.Conforme mostrado nas próximas duas figuras.

```
133  ----- File my_components.vhd: ---------------
134  LIBRARY ieee;
135  USE ieee.std_logic_1164.all;
136
137  PACKAGE my_components IS
138
139  COMPONENT somador8bits IS
140  PORT ( a, b : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
141  s : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
142  END  COMPONENT;
143
144  COMPONENT deslocaSoma7 IS
145
146  PORT ( E: IN STD_LOGIC_VECTOR (3 downto 0);
147         S: OUT STD_LOGIC_VECTOR (7 downto 0)
148
149    );
150  END COMPONENT;
151
152  COMPONENT deslocaSomaN1 IS
153
154  PORT ( E: IN STD_LOGIC_VECTOR (3 downto 0);
155         S: OUT STD_LOGIC_VECTOR (7 downto 0)
156
157    );
158  END COMPONENT;
159
```

Figura 8: my_components

```
160  COMPONENT deslocaSomaN3 IS
161  
162  PORT ( E: IN STD_LOGIC_VECTOR (3 downto 0);
163         S: OUT STD_LOGIC_VECTOR (7 downto 0)
164  
165    );
166   END COMPONENT;
167  
168  COMPONENT deslocaSomaN7 IS
169  
170  PORT ( E: IN STD_LOGIC_VECTOR (3 downto 0);
171         S: OUT STD_LOGIC_VECTOR (7 downto 0)
172  
173    );
174   END COMPONENT;
175  
176  COMPONENT reg8b IS
177  PORT ( clk : IN STD_LOGIC;
178   D : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
179   Q : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
180   END COMPONENT;
181  
182  COMPONENT reg4b IS
183  PORT ( clk : IN STD_LOGIC;
184   D : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
185   Q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
186   END COMPONENT;
187  
188  
189   END my_components;
```

Figura 9: my_components

Na última parte tem-se o código do filtro FIR, com a inicialização de todos os elementos necessários para ele. Tem-se a diferença conforme mostrado na figura 10 a 13, sendo elas nas duas primeiras figuras temos os registradores pipelines e nas duas ultimas não.

```
195  ENTITY Trabalho1PCICpipeline IS
196     PORT (clk: IN STD_LOGIC;
197           X: IN STD_LOGIC_VECTOR (3 downto 0);
198           S: OUT STD_LOGIC_VECTOR (7 downto 0)
199           );
200  END Trabalho1PCICpipeline;
201  
202  ARCHITECTURE comportamento OF Trabalho1PCICpipeline   IS
203   SIGNAL RR00, RR01, RR02: STD_LOGIC_VECTOR (3 downto 0);
204   SIGNAL RR03,RR04,RR05,RR06, RM00, RM01, RM02, RM03, RM04: STD_LOGIC_VECTOR (7 downto 0);
205   SIGNAL RS00, RS01, RS02, RS03: STD_LOGIC_VECTOR (7 downto 0);
206  
207  
```

Figura 10: Filtro FIR - Com Pipeline

```
208  BEGIN
209  
210      stage_1: deslocaSoma7 port map (X, RM00);
211      stage_2: reg4b port map (clk, X, RR00);
212      stage_3: reg8b port map (clk, RM00, RR03);
213      stage_4: deslocaSomaN3 port map (RR00, RM01);
214      stage_5: reg8b port map (clk, RM01, RR04);
215      stage_6: somador8bits port map (RR03, RR04, RS00);
216  
217  
218      stage_7: reg4b port map (clk, RR00, RR01);
219      stage_8: deslocaSomaN1 port map (RR01,RM02);
220      stage_9: reg8b port map (clk, RM02, RR05);
221      stage_10: somador8bits port map (RS00, RR05, RS01);
222  
223  
224      stage_11: reg4b port map (clk, RR01, RR02);
225      stage_12: deslocaSomaN7 port map (RR02, RM03);
226      stage_13: reg8b port map (clk, RM03, RR06);
227      stage_14: somador8bits port map (RS01, RR06, RS02);
228  
229   S <= RS02;
230  
231   END comportamento;
```

Figura 11: Filtro FIR - Com Pipeline

```
164  LIBRARY ieee;
165  USE ieee.std_logic_1164.all;
166  USE work.my_components.all;
167
168  ENTITY Trabalho1PCISPipeline IS
169     PORT (clk: IN STD_LOGIC;
170           X: IN STD_LOGIC_VECTOR (3 downto 0);
171           S: OUT STD_LOGIC_VECTOR (7 downto 0)
172           );
173  END Trabalho1PCISPipeline;
174
175  ARCHITECTURE comportamento OF Trabalho1PCISPipeline IS
176  SIGNAL RR00, RR01, RR02: STD_LOGIC_VECTOR (3 downto 0);
177  SIGNAL RM00, RM01, RM02, RM03, RM04: STD_LOGIC_VECTOR (7 downto 0);
178  SIGNAL RS00, RS01, RS02, RS03: STD_LOGIC_VECTOR (7 downto 0);
```

Figura 12: Filtro FIR - Sem Pipeline

```
181  BEGIN
182
183     stage_1: deslocaSoma7 port map (X, RM00);
184     stage_2: reg4b port map (clk, X, RR00);
185     stage_3: deslocaSomaN3 port map ( RR00, RM01);
186     stage_4: somador8bits port map (RM00, RM01, RS00);
187
188
189     stage_5: reg4b port map (clk, RR00, RR01);
190     stage_6: deslocaSomaN1 port map (RR01,RM02);
191     stage_7: somador8bits port map (RS00, RM02, RS01);
192
193
194     stage_8: reg4b port map (clk, RR01, RR02);
195     stage_9: deslocaSomaN7 port map (RR02, RM03);
196     stage_10: somador8bits port map (RS01, RM03, RS02);
197
198  S <= RS02;
199
200  END comportamento;
```

Figura 13: Filtro FIR - Sem Pipeline

3. Discussões e Resultados:

Abaixo temos a arquitetura do filtro FIR sem pipeline e com pipeline gerada pelo quartus II , conforme figura 14 e 15 respectivamente , e o resultado da quantidade de registradores e pins de ambos os filtros, o sem pipeline na figura 16 e o com na figura 17 .

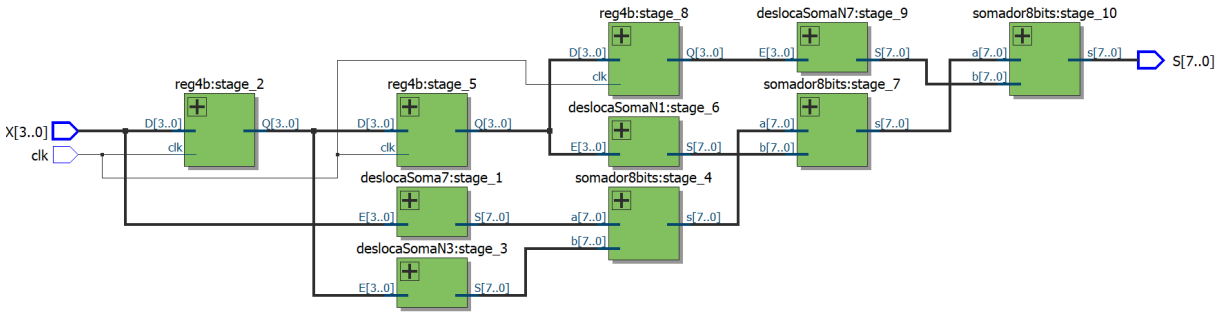

Figura 14: Arquitetura do filtro FIR totalmente paralelo - Sem Pipeline.

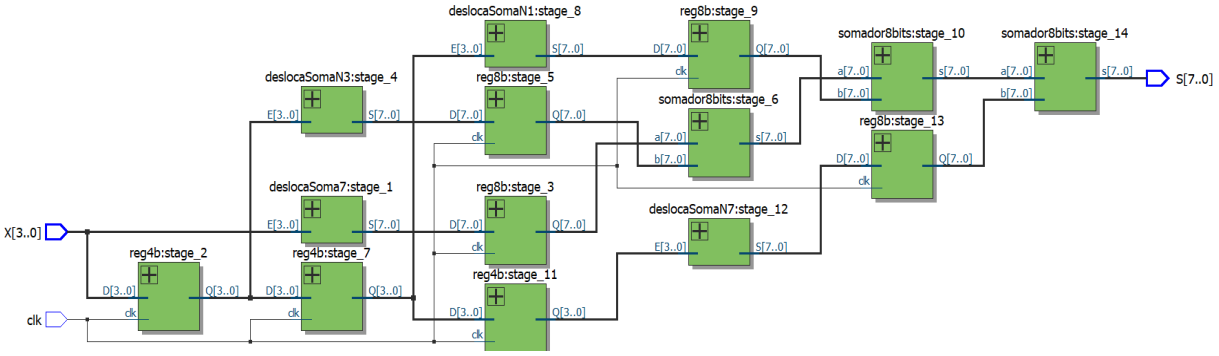

Figura 15: Arquitetura do filtro FIR totalmente paralelo - Com Pipeline.

Flow Status	Successful - Wed Apr 27 22:49:50 2022
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Trabalho1PCISPipeline
Top-level Entity Name	Trabalho1PCISPipeline
Family	Cyclone IV GX
Total logic elements	48 / 14,400 (< 1 %)
Total combinational functions	44 / 14,400 (< 1 %)
Dedicated logic registers	12 / 14,400 (< 1 %)
Total registers	12
Total pins	13 / 81 (16 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final

Figura 16: Flow Status Filtro Fir totalmente paralelo - Sem Pipeline

Flow Status	Successful - Wed Apr 27 23:13:17 2022
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Web Edition
Revision Name	Trabalho1PCICpipeline
Top-level Entity Name	Trabalho1PCICpipeline
Family	Cyclone IV GX
Total logic elements	49 / 14,400 (< 1 %)
Total combinational functions	44 / 14,400 (< 1 %)
Dedicated logic registers	36 / 14,400 (< 1 %)
Total registers	36
Total pins	13 / 81 (16 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final

Figura 17: Flow Status Filtro Fir totalmente paralelo - Com Pipeline

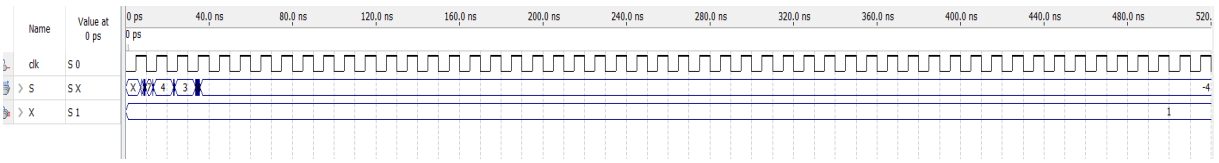

Figura 18: Testbench gerado - Sem Pipeline

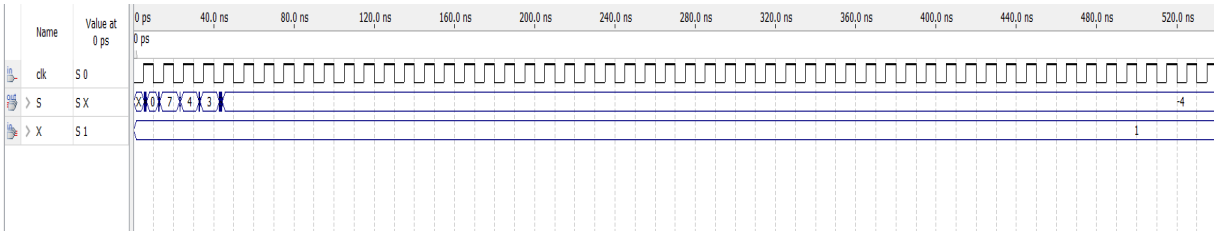

Figura 19: Testbench gerado - Com Pipeline

Conforme mostra a figura 18 e 19 temos o somatório da multiplicação dos valores dos desloca-somas com o X, que vale 1 no teste, o valor final é igual a -4 e obtemos como tempo de resposta para o sem pipeline aproximadamente 40ns e com pipeline aproximadamente 50ns.

4. Conclusão:

Teve-se um resultado bem positivo no filtro FIR sem registradores pipeline em relação ao com os registradores. Teve-se diferença no número de elementos lógicos e no número de registradores, sendo o sem pipeline obteve 48 elementos lógicos e 12 registradores contra 49 elementos lógicos e 36 registradores na versão com pipeline.

Entende-se que o o filtro FIR sem pipeline é mais rápido e menos custoso de modo geral tendo levado em consideração o tempo de reação e o número de elementos utilizados.