Projeto de sistemas digitais combinacionais em VHDL

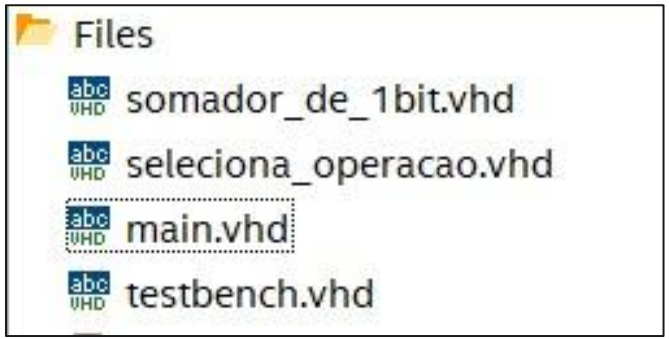
Somador subtrator de múltiplos bits configurável

Equipe 0:

Frank B. Boa Morte (202006840007)
Fernando F. Dimas (201906840031)
Mercedes M. B. Diniz (201906840030)

Implementação do projeto

1. Somador completo de 1 bit
2. Subtrator (complemento a 2 do operando B)
3. Somador completo com número de bits configurável
4. Testbench

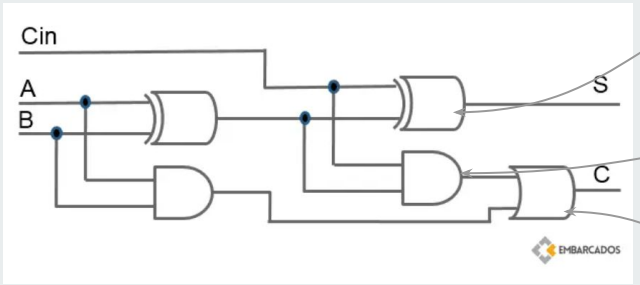


Somador completo de 1 bit

A	B	Cin	S	Cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$S = A \oplus B \oplus Cin$$

$$Cout = A.B + A.Cin + B.Cin$$



porta lógica XOR

porta lógica AND

porta lógica OR

Somador completo de 1 bit

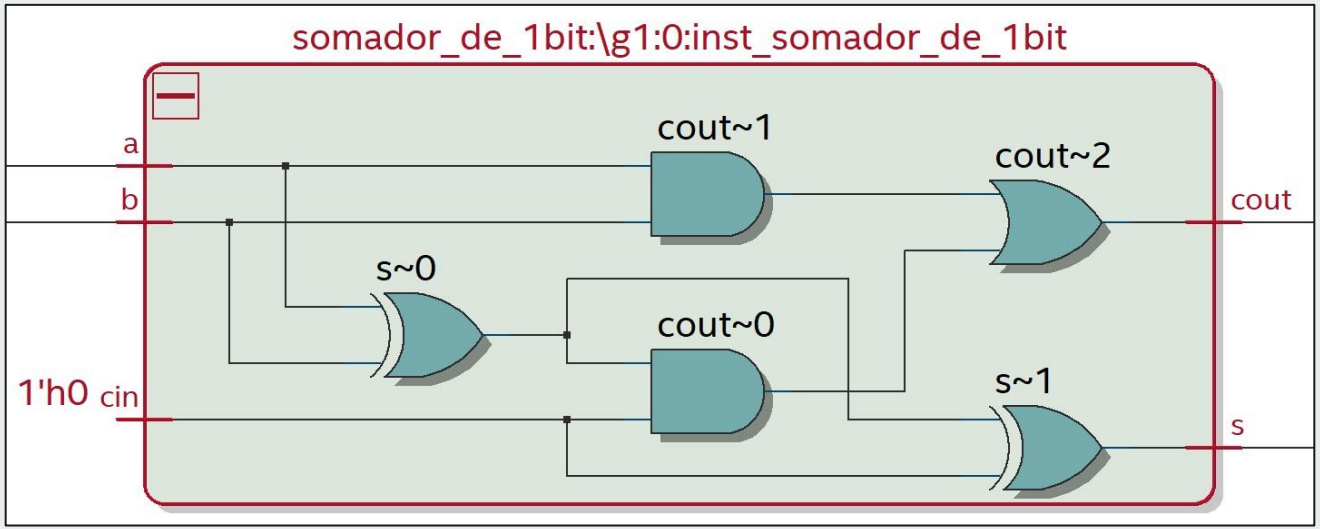
somador_de_1bit.vhd

```vhdl
-- Projeto 01 - PHI
-- Somador completo de 1 bit a partir de portas lógicas usando funções lógicas da biblioteca std_logic_1164

library IEEE;
use IEEE.std_logic_1164.all;

entity somador_de_1bit is                          -- Declaracao da entidade
    port(
        a, b, cin: in std_logic;                   -- Entradas
        s, cout: out std_logic                     -- Saidas
    );
end somador_de_1bit;

architecture comportamento of somador_de_1bit is   -- Declaracao da arquitetura
begin
    s <= cin xor (a xor b);
    cout <= (cin and (a xor b)) or (a and b);

end comportamento;
```

$$S = A \oplus B \oplus Cin$$

$$Cout = A.B + A.Cin + B.Cin$$

Somador completo de 1 bit



Selecionando a operação

Valor de entrada para selecionar o valor de B

seleciona_operacao.vhd

```
1  -- Projeto 01 - PHI
2  -- Configura a operação de soma ou subtração atravez da seleção da entrada b (que pode ser b ou seu complemento)
3
4  library IEEE;
5  use IEEE.std_logic_1164.all;
6  use ieee.numeric_std.all;
7
8  entity seleciona_operacao is                              -- Declaracao da entidade
9      generic(
10         nbits    :    integer := 4                -- número de bits
11     );
12     port(
13         input_B: in std_logic_vector (nbits-1 downto 0);          -- Entradas
14         sel_B : in std_logic;                                        -- Seleciona a entrada
15         B : out std_logic_vector (nbits-1 downto 0)               -- Saidas
16     );
17 end seleciona_operacao;
18
19 architecture comportamento of seleciona_operacao is       -- Declaracao da arquitetura
20     -- Cria um vetor do mesmo tamanho que B com o bit menos significativo igual a  1:
21     constant one : unsigned(B'RANGE) := (0 => '1', others => '0');
22
23 begin
24     -- Atribui o valor de B de acordo com a seleção (0 = mantem o valor | 1 = atribui o compçemento de 2)
25     -- calculo do complemento de 2: inverte os bits de B e soma 1
26
27     B <= input_B when sel_B='0' else std_logic_vector(unsigned (not input_B) + one);
28 end comportamento;
```

Saída B recebe entrada B normal se sel_B=’0’ , se não recebe o complemento de 2 de B’

Complemento de 2 do B

Recebe a entrada B

Saída B depende do valor de sel_B

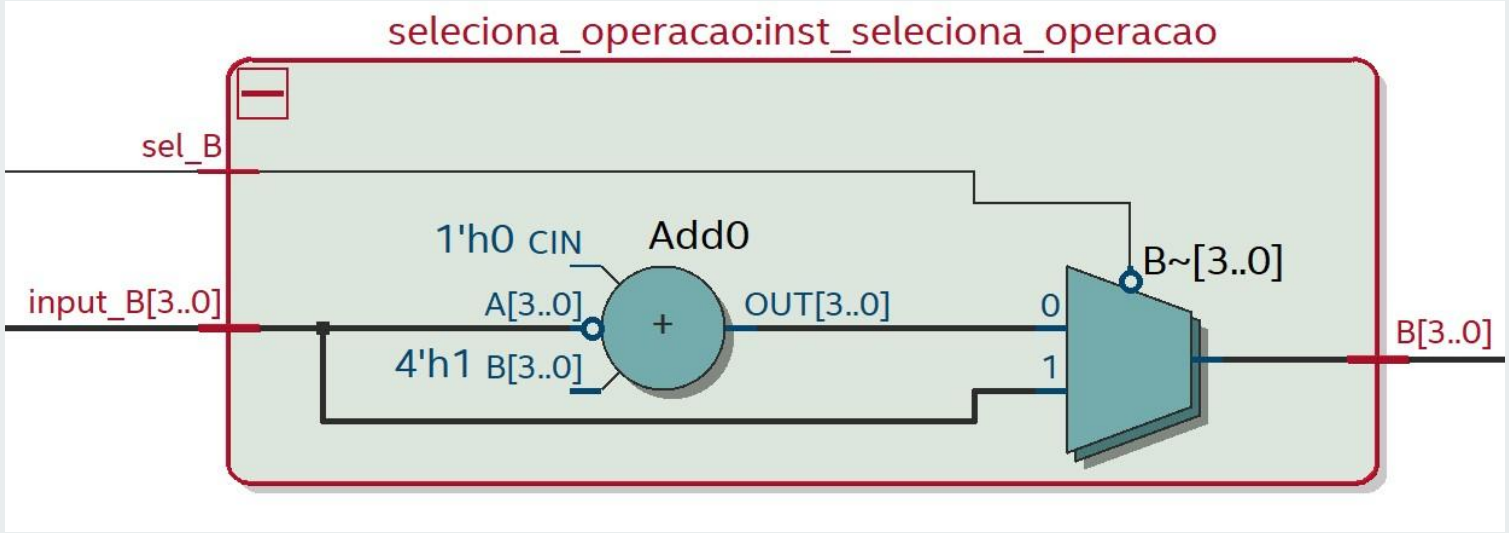
Vetor de nbits com o bit menos significativo igual a 1

Negação da entrada B somada com o 1 (Complemento de 2)

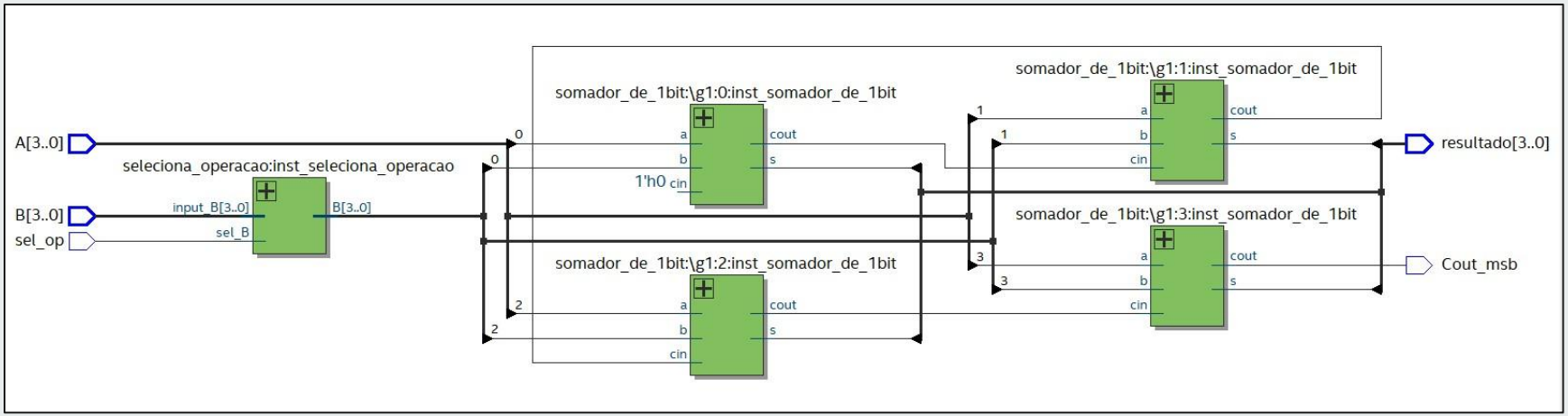
seleciona_operacao.vhd

```
1  -- Projeto 01 - PHI
2  -- Configura a operação de soma ou subtração atravez da seleção da entrada b (que pode ser b ou seu complemento)
3
4  library IEEE;
5  use IEEE.std_logic_1164.all;
6  use ieee.numeric_std.all;
7
8  entity seleciona_operacao is                          -- Declaracao da entidade
9      generic(
10         nbits    :    integer := 4              -- número de bits
11     );
12     port(
13         input_B: in std_logic_vector (nbits-1 downto 0);         -- Entradas
14         sel_B : in std_logic;                                    -- Seleciona a entrada
15         B : out std_logic_vector (nbits-1 downto 0)              -- Saidas
16     );
17 end seleciona_operacao;
18
19 architecture comportamento of seleciona_operacao is      -- Declaracao da arquitetura
20     -- Cria um vetor do mesmo tamanho que B com o bit menos significativo igual a 1:
21     constant one : unsigned(B'RANGE) := (0 => '1', others => '0');
22
23 begin
24     -- Atribui o valor de B de acordo com a seleção (0 = mantem o valor | 1 = atribui o compçemento de 2)
25     -- calculo do complemento de 2: inverte os bits de B e soma 1
26
27     B <= input_B when sel_B='0' else std_logic_vector(unsigned (not input_B) + one);
28 end comportamento;
```

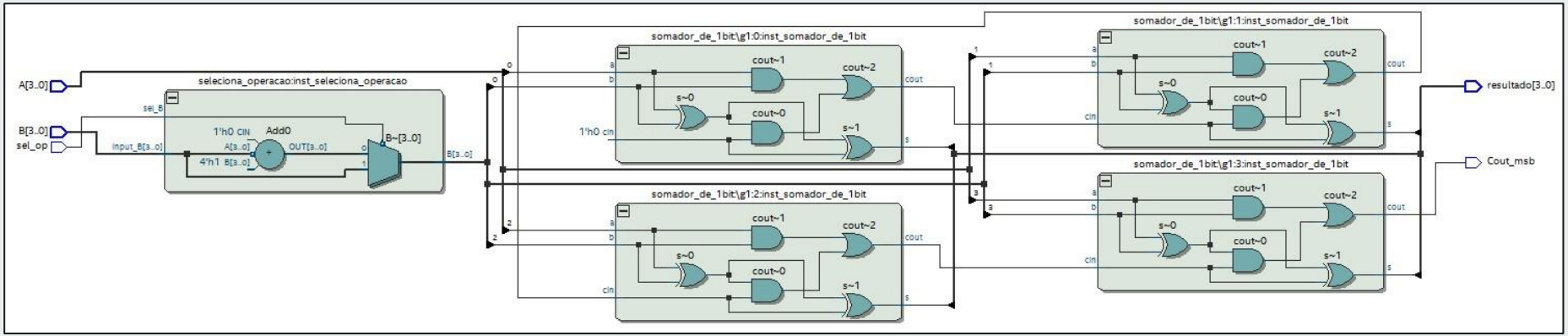
Selecionando a operação e fazendo o Complemento de 2 do B



Somador completo com número de bits configurável



Somador completo com número de bits configurável



Somador completo com número de bits configurável (Visão Geral)

main.vhd

```vhdl
-- Projeto de sistemas digitais combinacionais em VHDL - Proj.01 - PHI
-- Somador de múltiplos bits configurado para permitir selecionar soma ou subtração

library IEEE;
use ieee.std_logic_1164.all;

entity main is
    generic(
        nbits    :    integer := 4                          -- número de bits
    );
    port(
end main;

architecture comportamento of main is

    -- Declarando o componente somador_de_1bit
    component somador_de_1bit is
        port(
    end component;

    -- Declarando o componente seleciona_operacao
    component seleciona_operacao is
        port(
    end component;

    -- Declaração das constantes e variaveis:
    signal  Cout_somadores : std_logic_vector(nbits downto 0) := (others => '0');  -- Recebe a saida Cout dos blocos somadores
    signal  Cout_invetB : std_logic_vector(nbits downto 0) := (others => '0');     -- Recebe a saida Cout dos blocos inverte B

    signal saidaB2 : std_logic_vector(nbits-1 downto 0);                           -- Saida do complemento de 2 da entrada B
    signal entradaBEscolhida : std_logic_vector(nbits-1 downto 0);                 -- Valor da entrada B escolhida

begin

    -- Gerando os blocos duplicados:
    g1: for i in 0 to nbits-1 generate
        -- Instanciando os blocos do somador_de_1bit
        inst_somador_de_1bit : somador_de_1bit
            port map(
    end generate g1;

    -- Instanciando os blocos do seleciona_operacao
    inst_seleciona_operacao: seleciona_operacao
        port map(
    Cout_msb <= Cout_somadores(nbits);  -- salvando o Cout mais significativo

end comportamento;
```

Somador completo com número de bits configurável

entity da main: define as portas de entrada e saída da visão externa do componente

variável que atribui a quantidade de bits que o somador poderá operar

seleciona se a operação será soma (0) ou subtração (1)

operadores (vetor de bits)

armazena o resultado da operação feita em cada bit

```
1  -- Projeto de sistemas digitais combinacionais em VHDL - Proj.01 - PHI
2  -- Somador de múltiplos bits configurado para permitir selecionar soma ou subtração
3
4  library IEEE;
5  use ieee.std_logic_1164.all;
6
7  entity main is
8      generic(
9          nbits    :    integer := 4                    -- número de bits
10     );
11     port(
12         A, B : in std_logic_vector(nbits-1 downto 0);
13         sel_op : in std_logic;
14         resultado : out std_logic_vector(nbits-1 downto 0);
15         Cout_msb : out std_logic
16     );
17 end main;
18
```

Somador completo com número de bits configurável

architecture da main: descreve as relações existentes entre as interfaces de entrada e saída (visão interna)

região de declaração de constantes sinais e componentes

```
19  architecture comportamento of main is
20
21      -- Declarando o componente somador_de_1bit
22      component somador_de_1bit is
23          port(
24              a, b, cin: in std_logic;                    -- Entradas
25              s, cout: out std_logic                      -- Saidas
26          );
27      end component;
28
29      -- Declarando o componente seleciona_operacao
30      component seleciona_operacao is
31          port(
32              input_B: in std_logic_vector(nbits-1 downto 0);       -- Entradas
33              sel_B : in std_logic;                                 -- Seleciona a entrada
34              B : out std_logic_vector(nbits-1 downto 0)            -- Saidas
35          );
36      end component;
37
38      -- Declaração das constantes e variaveis:
39      signal  Cout_somadores : std_logic_vector(nbits downto 0) := (others => '0');  -- Recebe a saida Cout dos blocos somadores
40      signal  Cout_invetB : std_logic_vector(nbits downto 0) := (others => '0');     -- Recebe a saida Cout dos blocos inverte B
41
42      signal saidaB2 : std_logic_vector(nbits-1 downto 0);                          -- Saida do complemento de 2 da entrada B
43      signal entradaBEscolhida : std_logic_vector(nbits-1 downto 0);                -- Valor da entrada B escolhida
44
45  begin
```

Somador completo com número de bits configurável

região de código concorrente (descreve o sistema.)

```
45  begin
46
47      -- Gerando os blocos duplicados:
48      g1: for i in 0 to nbits-1 generate
49          -- Instanciando os blocos do somador_de_1bit
50          inst_somador_de_1bit : somador_de_1bit
51              port map(
52                  a => A(i),
53                  b => entradaBEscolhida(i),
54                  cin => Cout_somadores(i),
55                  s => resultado(i),
56                  cout => Cout_somadores(i+1)
57              );
58
59      end generate g1;
60
61      -- Instanciando os blocos do seleciona_operacao
62      inst_seleciona_operacao: seleciona_operacao
63          port map(
64              input_B => B,
65              sel_B => sel_op,
66              B => entradaBEscolhida
67          );
68
69      Cout_msb <= Cout_somadores(nbits);   -- salvando o Cout mais significativo
70
71  end comportamento;
```

constrói os blocos do somador de 1 bit

constrói o bloco do seletor

função generate: utilizada para duplicar os blocos do somador de 1 bits de acordo com o número de bits que se deseja operar

Testbench (Visão Geral)

para configurar o número de bits é necessário alterar esses dois campos

construção dos processos

definição dos valores que serão atribuídos ao seletor de operação

sinais que interligam os blocos

seleciona se a operação será soma (0) ou subtração (1)

testbench.vhd*

```
1   -- Projeto 01 - PHI
2   -- Testbench para avaliar a funcionalidade desenvolvida
3
4   library ieee;
5   use ieee.std_logic_1164.all;
6   use ieee.numeric_std.all;
7   use std.textio.all;
8
9   entity testbench is
10  end testbench;
11
12  architecture test of testbench is
13      -- Declarando o componente main
14      component main is
15          generic(
16          nbits   :   integer := 4                        -- número de bits
17          );
18          port(
24      end component;
25
26      -- Declarando as constantes e variaveis
27      constant nbits : integer := 4;   -- número de bits
28
29      constant somasub_val : std_logic_vector(63 downto 0) := x"0F0F0F0F0F0F0F0F";
30      signal input_a, input_b, result : std_logic_vector(nbits-1 downto 0);
31      signal sel_somasub : std_logic := '0';
32      signal cout_msb : std_logic;
33      signal clock : std_logic := '0';
34
35  begin
36      -- Instanciando a main
37      dut : main
38          port map(
39              A => input_a,
40              B => input_b,
41              sel_op => sel_somasub,
42              resultado => result,
43              Cout_msb => cout_msb
44          );
45
46      -- Processo 01: seleciona se a operação será soma ou subtração
47      selecionando_op: process (clock)
59      -- Processo 02: ler os valores das entradas A e B do arquivo de texto e atribui para os respctivos inputs
60      lendo_entradas: process (clock)
79      -- Processo 03: escreve o resultado e qual foi a respectiva operação no arquivo de texto
80      armazenando_resultados: process (clock)
94      clock <= not clock after 5 ns;  -- variando o clock
95
96  end test;
```

```
47    selecionando_op: process (clock)
48       variable ptr_mux : integer := 0;
49    begin
50       if rising_edge(clock) then
51          sel_somasub  <= somasub_val(ptr_mux);
52          ptr_mux := ptr_mux + 1;
53          if ptr_mux = 64 then
54             ptr_mux    := 0;
55          end if;
56       end if;
57    end process;
```

Processo 1: Seleciona se a operação será Soma ou Subtração.

valores variam de acordo com o determinado na constante “somasub_val”

```
somasub_val : std_logic_vector(63 downto 0) := x"0F0F0F0F0F0F0F0F";
```

após percorrer todo o vetor de valores, se reinicia

Processo 2: ler os valores das entradas A e B do arquivo de texto e atribui para os respectivos inputs

Lê os dados de entrada contidos no arquivo “entradas.txt”

```
60    lendo_entradas: process (clock)
61       file F: TEXT open READ_MODE is "C:\Users\Mercedes\QuartusLite\Projetos\PHI_VHDL\projetos\proj01_phi_equipe0\arq_entrada_saida\entradas.txt";
62       variable L: LINE;
63       variable entrada : integer;
64    begin
65       if rising_edge(clock) then
66          if not endfile(F) then
67             READLINE(F, L);
68             READ(L, entrada);
69             input_a <= std_logic_vector(to_unsigned(entrada, nbits));
70
71             READLINE(F, L);
72             READ (L, entrada);
73             input_b <= std_logic_vector(to_unsigned(entrada, nbits));
74
75          end if;
76       end if;
77    end process;
```

Lê a entrada com um valor inteiro

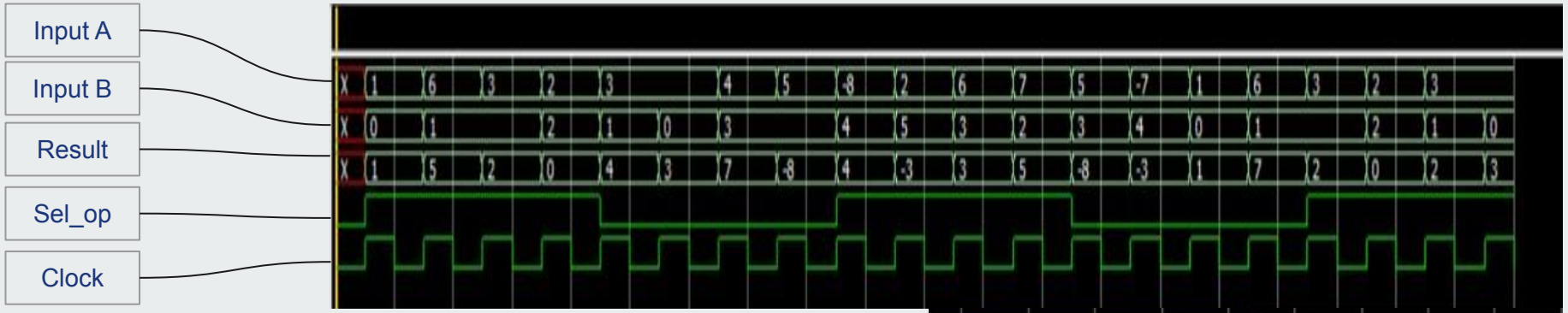
Em cada subida do pulso do clock, se lê um valor contido em uma linha, e o atribui para a entrada A e outro para a entrada B

Processo 3: Escreve o resultado e qual foi a respectiva operação no arquivo de texto.

Armazena dados de interesse no arquivo "saida.txt"

```
80    armazenando_resultados: process (clock)
81       file F: TEXT open WRITE_MODE is "C:\Users\Mercedes\QuartusLite\Projetos\PHI_VHDL\projetos\proj01_phi_equipe0\arq_entrada_saida\saida.txt";
82       variable L: LINE;
83    begin
84       if rising_edge(clock) then
85          WRITE (L, to_bit(cout_msb));
86          WRITE (L, ' ');
87          WRITE (L, to_integer(unsigned(result)));
88          WRITE (L, ' ');
89          WRITE (L, to_bit(sel_somasub));
90          WRITELINE (F, L);
91       end if;
92    end process;
```

Na subida do pulso do clock (1) os valores do resultado, Count_msb e o identificador da operação são salvos



Simulação (Testbench)

A Simulação foi executada por 200 ns

Entradas e Saídas