Relatório 5

Nome: Gabriel Cruz Vaz Santos
Matrícula: 200049038
Turma: C

● Questão 1

Introdução

Foi solicitado a implementação de um somador de palavras 4 bits utilizando somente somadores completos (3 bits de entrada e 2 bits de saída)

Teoria

Um somador completo recebe 3 bits de entrada e 2 bits de saída (S e Cout). O S de cada um dos somadores será um dos dígitos da nova palavra formada. A entrada Cin do primeiro somador completo será ‘0’, enquanto os demais Cin dos outros somadores completos vão receber os Cout do somador que o antecede. O último somador completo terá suas duas saídas, cada uma sendo um bit, correspondendo a 2 bits da palavra final (S(4) e S(3)).

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. A figura 1 e 2 é referente a implementação da entidade e arquitetura do somador de palavras e do somador completo, enquanto as demais imagens são dos testes.

Relatorio5 > somador_completo.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity somador_completo is
    port(
        A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic
    );
end somador_completo;

architecture somador_completo_arch of somador_completo is
    begin
        s <= A xor B xor Cin;
        Cout <= ( A and B ) or ( A and Cin ) or ( B and Cin );
end somador_completo_arch;
```

Relatorio5 > somador_palavras_1_tb.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity somador_palavras_1_tb is end;

architecture somador_palavras_1_arch of somador_palavras_1_tb is
    component somador_palavras_1 is
        port(
            A: in std_logic_vector (3 downto 0);
            B: in std_logic_vector (3 downto 0);
            S: out std_logic_vector (4 downto 0)
        );
    end component;

    signal entrance_i: std_logic_vector (3 downto 0) := "0000";
    signal entrance_j: std_logic_vector (3 downto 0) := "0000";

    constant time_0: time := 2 ns;
    constant time_1: time := 4 ns;
    constant time_2: time := 8 ns;
    constant time_3: time := 16 ns;
    constant time_4: time := 32 ns;
    constant time_5: time := 64 ns;
    constant time_6: time := 128 ns;
    constant time_7: time := 256 ns;

    begin
        obj: somador_palavras_1 port map (A => entrance_i, B => entrance_j, S => open);

        entrance_j(3) <= not entrance_j(3) after time_7/2;
        entrance_i(2) <= not entrance_i(2) after time_6/2;
```

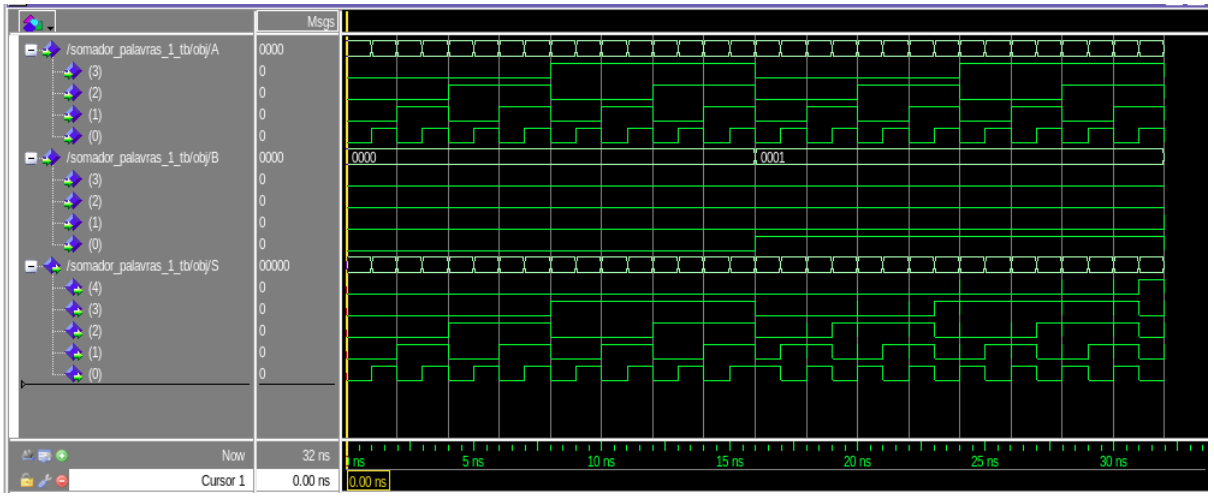
```
        entrance_j(1) <= not entrance_j(1) after time_5/2;
        entrance_j(0) <= not entrance_j(0) after time_4/2;
        entrance_i(3) <= not entrance_i(3) after time_3/2;
        entrance_i(2) <= not entrance_i(2) after time_2/2;
        entrance_i(1) <= not entrance_i(1) after time_1/2;
        entrance_i(0) <= not entrance_i(0) after time_0/2;

end somador_palavras_1_arch;
```

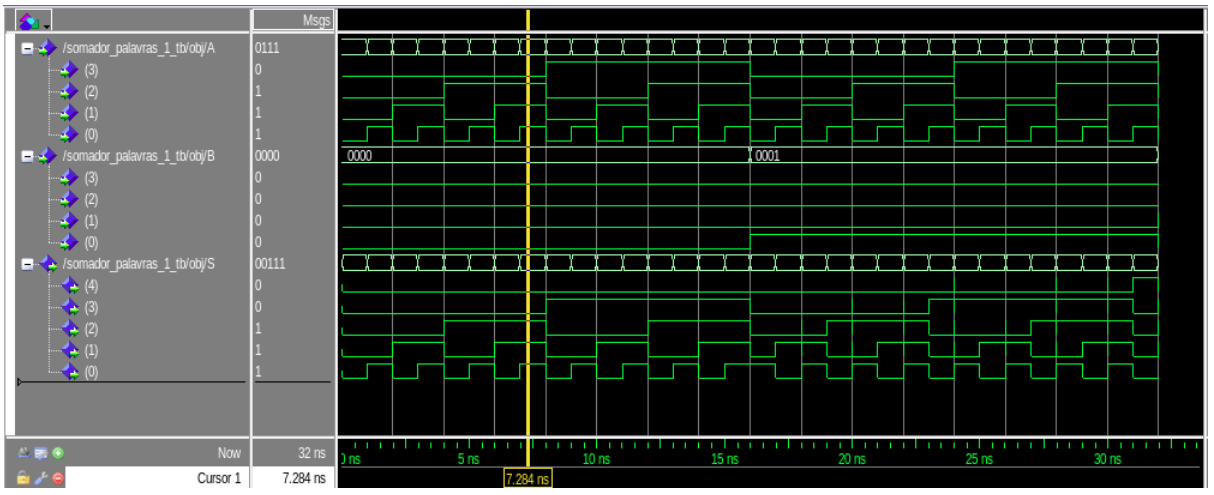
Compilação

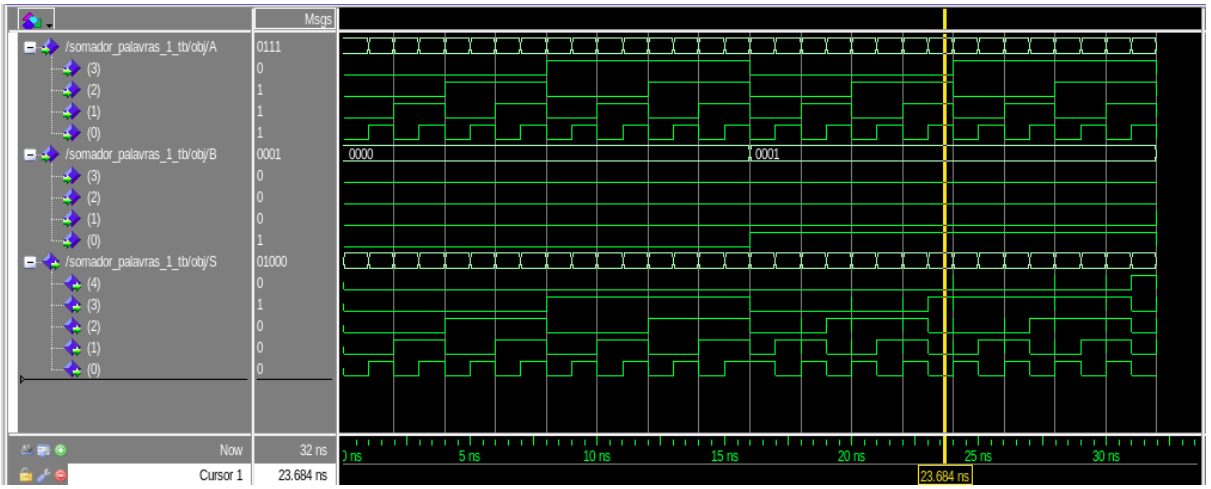
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

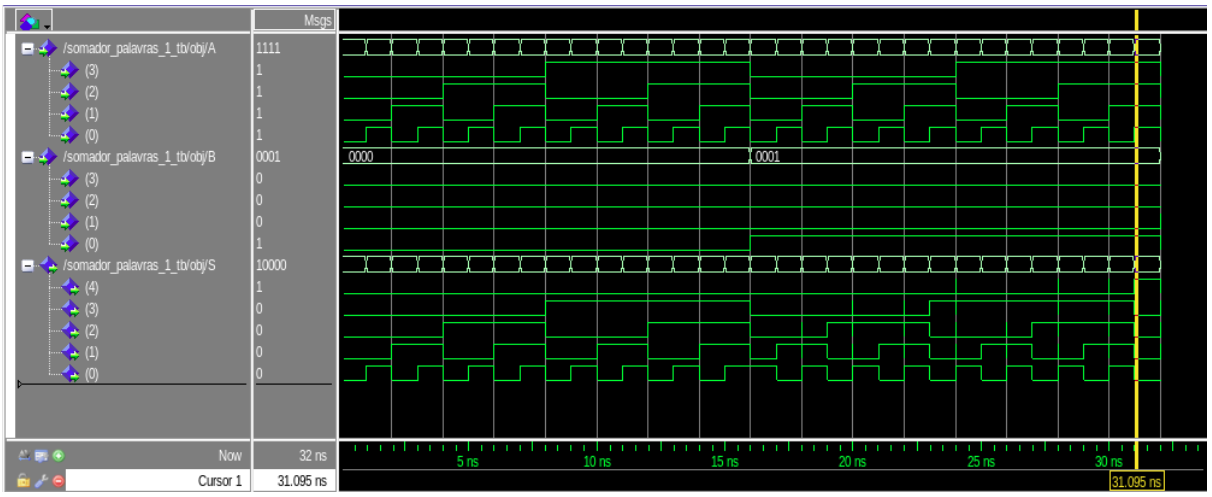
```
# Compile of somador_palavras_1.vhd was successful.
# Compile of somador_completo.vhd was successful.
# Compile of somador_palavras_1_tb.vhd was successful.
```

Simulador









Análise

A entidade somador_palavras_1 possui 8 bits de entrada no total, 4 bits na entrada A e 4 bits na entrada B. Cada um dos bits da saída S será a soma dos respectivos bits de A e B, levando em consideração também os Cin que chega do Cout dos outros somadores

Na imagem 2 da simulação note que todos os Cin de cada um dos somadores completos utilizados no somador de palavras será 0, pois em nenhum momento existe uma soma de ‘1’ + ‘1’.

Enquanto isso, a imagem 3 temos que o Cin de alguns dos somadores completos utilizados será ‘1’, pois temos a soma ‘1’ +’1’. A(0) + B(0) = ‘1’ + ‘1’ = ‘10’. Com isso, S(0) será igual a ‘0’ e a saída Cout do primeiro somador será ‘1’ e o Cin do segundo somador completo será ‘1’ também. Esse fenômeno se repete sempre que tivermos essa soma.

Conclusão

Nesse experimento conseguimos com êxito descrever um somador de palavras de 4 bits utilizando 4 somadores completos. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.

- Questão 2

Teoria

Foi solicitado a implementação de um somador de palavras 4 bits utilizando o operador ‘+’ do pacote STD_LOGIC_ARITH. A entidade possuirá 2 vetores de entrada com 4 bits cada e uma saída de 5 bits.

Teoria

O operador ‘+’ do pacote STD_LOGIC_ARITH permite a soma das entradas A e B de acordo com seu índice, levando em conta cada bit. utilizamos também o tipo *unsigned* para conseguirmos utilizar o operador ‘+’ com os vetores de entrada, que normalmente apenas aceitam tipos inteiros e reais.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. A figura 1 é referente a implementação da entidade e arquitetura do somador de palavras utilizando o operador ‘+’, enquanto as demais imagens são dos testes.

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_arith.all;
4
5   entity somador_palavras_2 is
6       port(
7           A: in std_logic_vector (3 downto 0);
8           B: in std_logic_vector (3 downto 0);
9           S: out std_logic_vector (4 downto 0)
10      );
11
12  end somador_palavras_2;
13
14  architecture somador_palavras_2_arch of somador_palavras_2 is
15      begin
16          S <= unsigned('0' & A) + unsigned('0' & B);
17  end somador_palavras_2_arch;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity somador_palavras_2_tb is end;

architecture somador_palavras_2_arch of somador_palavras_2_tb is
    component somador_palavras_2 is
        port(
            A: in std_logic_vector (3 downto 0);
            B: in std_logic_vector (3 downto 0);
            S: out std_logic_vector (4 downto 0)
        );
    end component;

    signal entrance_i: std_logic_vector (3 downto 0) := "0000";
    signal entrance_j: std_logic_vector (3 downto 0) := "0000";

    constant time_0: time := 2 ns;
    constant time_1: time := 4 ns;
    constant time_2: time := 8 ns;
    constant time_3: time := 16 ns;
    constant time_4: time := 32 ns;
    constant time_5: time := 64 ns;
    constant time_6: time := 128 ns;
    constant time_7: time := 256 ns;
```

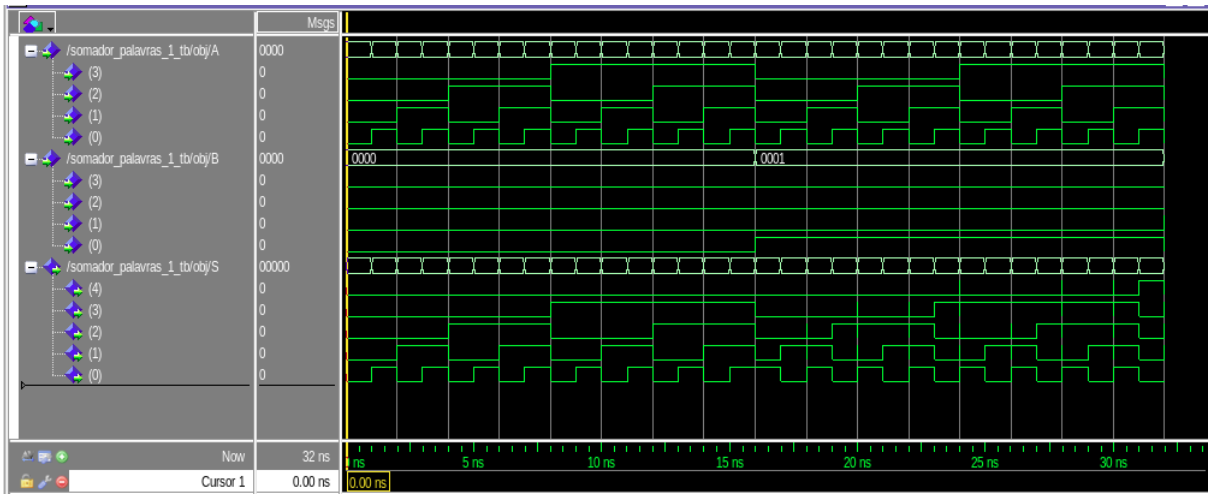
```vhdl
    begin
        obj: somador_palavras_2 port map (A => entrance_i, B => entrance_j, S => open);

        entrance_j(3) <= not entrance_j(3) after time_7/2;
        entrance_j(2) <= not entrance_j(2) after time_6/2;
        entrance_j(1) <= not entrance_j(1) after time_5/2;
        entrance_j(0) <= not entrance_j(0) after time_4/2;
        entrance_i(3) <= not entrance_i(3) after time_3/2;
        entrance_i(2) <= not entrance_i(2) after time_2/2;
        entrance_i(1) <= not entrance_i(1) after time_1/2;
        entrance_i(0) <= not entrance_i(0) after time_0/2;

end somador_palavras_2_arch;
```

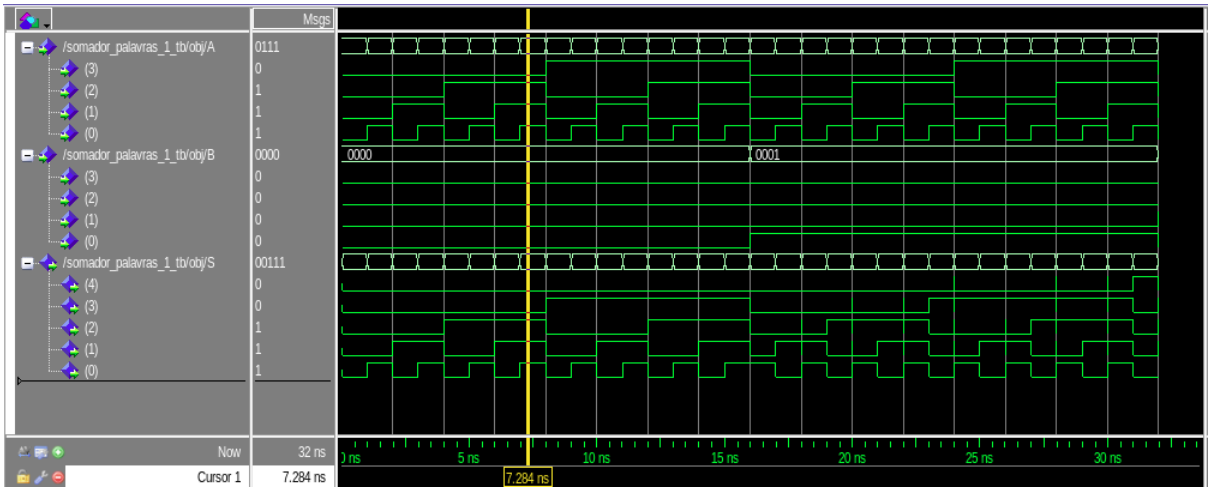
Compilação

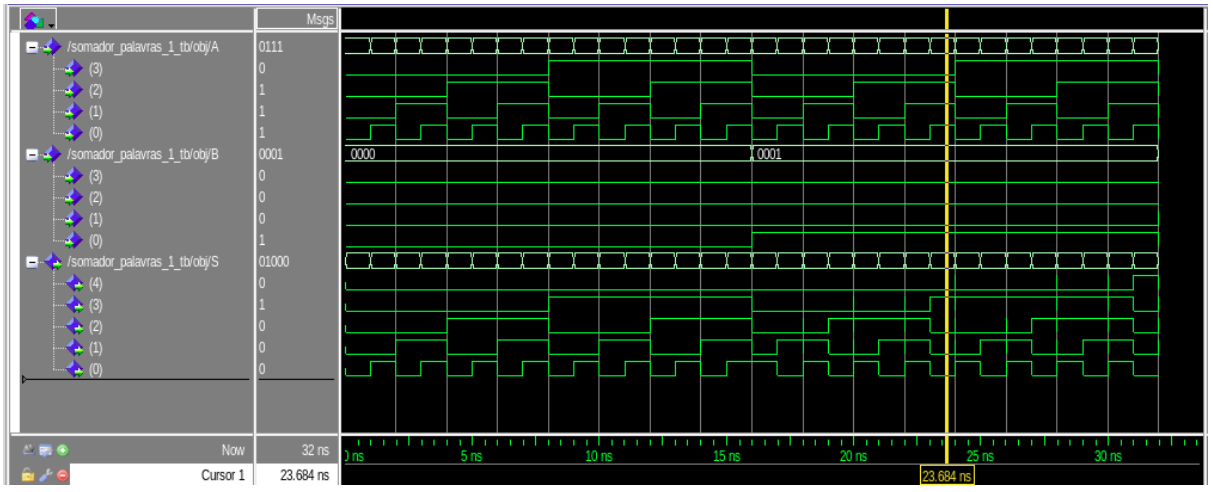
Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

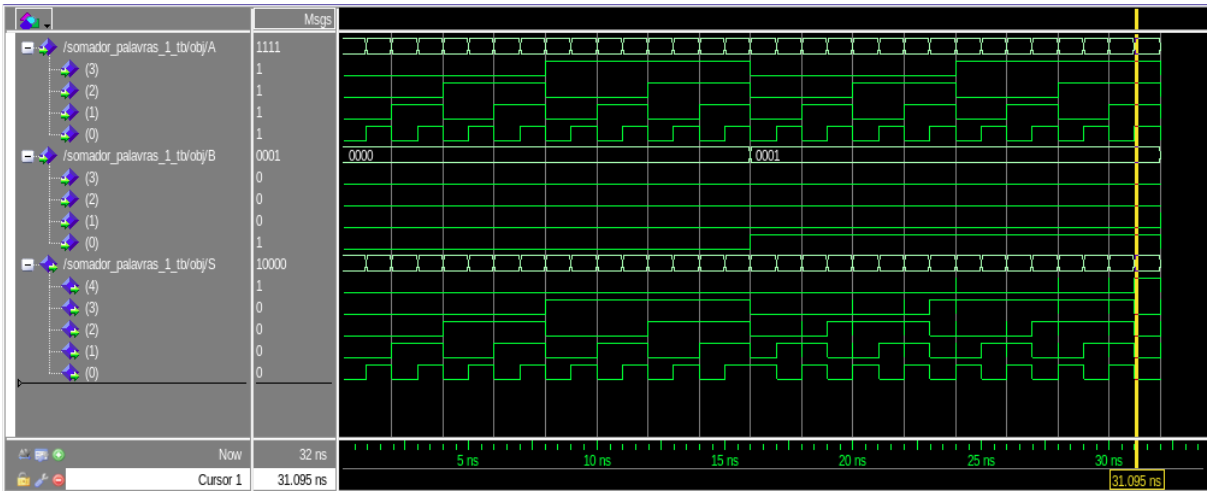
```
# Compile of somador_palavras_2.vhd was successful.
# Compile of somador_palavras_1.vhd was successful.
# Compile of somador_completo.vhd was successful.
# Compile of somador_palavras_1_tb.vhd was successful.
# Compile of somador_palavras_2_tb.vhd was successful.
# 5 compiles, 0 failed with no errors.
```

Simulador


Msgs
/somador_palavras_1_tb/obj/A
0000
(3)
0
(2)
0
(1)
0
(0)
0
/somador_palavras_1_tb/obj/B
0000
(3)
0
(2)
0
(1)
0
(0)
0
/somador_palavras_1_tb/obj/S
00000
(4)
0
(3)
0
(2)
0
(1)
0
(0)
0
0000
0001
Now
32 ns
Cursor 1
0.00 ns
0 ns
5 ns
10 ns
15 ns
20 ns
25 ns
30 ns
0.00 ns


Msgs
/somador_palavras_1_tb/obj/A
0111
(3)
0
(2)
1
(1)
1
(0)
1
/somador_palavras_1_tb/obj/B
0000
(3)
0
(2)
0
(1)
0
(0)
0
/somador_palavras_1_tb/obj/S
00111
(4)
0
(3)
0
(2)
1
(1)
1
(0)
1
0000
0001
Now
32 ns
Cursor 1
7.284 ns
5 ns
10 ns
15 ns
20 ns
25 ns
30 ns
7.284 ns


Msgs
/somador_palavras_1_tb/obj/A
0111
(3)
0
(2)
1
(1)
1
(0)
1
/somador_palavras_1_tb/obj/B
0001
(3)
0
(2)
0
(1)
0
(0)
1
/somador_palavras_1_tb/obj/S
01000
(4)
0
(3)
1
(2)
0
(1)
0
(0)
0
0000
0001
Now
32 ns
Cursor 1
23.684 ns
5 ns
10 ns
15 ns
20 ns
25 ns
30 ns
23.684 ns



Análise

A entidade somador_palavras_2 possui 8 bits de entrada no total, 4 bits na entrada A e 4 bits na entrada B. Cada um dos bits da saída S será a soma dos respectivos bits de A e B, através do operador '+'.

Note que mesmo sendo feito de formas diferentes, o somador de palavras de 4 bits desenvolvido possui a mesma simulação da questão 1. Esse resultado já era esperado tendo em vista que na prática trata-se da 'mesma entidade', porém feita de uma forma diferente.

Conclusão

Nesse experimento conseguimos com êxito descrever um somador de palavras de 4 bits utilizando o operador '+' do pacote STD_LOGIC_ARITH. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.

- Questão 3

Introdução

Foi solicitado fazer um testbench para testar e simular o somador de palavras desenvolvido nas questões acima.

Teoria

O testbench possui 256 diferentes combinações para os valores das entradas A e B , aguardando 500 ns entre as combinações. Com isso, compara as saídas do somador de palavras da questão 1 e da questão 2. O esperado é que sejam sempre iguais, tendo em vista que são a mesma entidade executadas de formas diferentes. Caso as saídas não sejam iguais dentro da mesma combinação, deve-se imprimir uma mensagem de erro durante a simulação.

Código

Os códigos foram programados no ambiente de desenvolvimento integrado Visual Studio Code, na linguagem VHDL e compilados pelo software do ModelSim. As figuras 1 e 2 são referentes a entidade testbench de testbench e as figuras 3 e 4 são referentes ao teste do topmodule, cujo foi observado durante a simulação.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity testbench is
    port(
        f_dut: in std_logic_vector (4 downto 0);
        f_gm: in std_logic_vector (4 downto 0);
        A: out std_logic_vector (3 downto 0);
        B: out std_logic_vector (3 downto 0)
    );
end testbench;

architecture testbench_arch of testbench is
begin
    process

    begin
        report "Iniciando testebench..." severity NOTE;
        for i in 0 to 15 loop
            A <= std_logic_vector(to_unsigned(i, 4));
            for j in 0 to 15 loop
                B <= std_logic_vector(to_unsigned(j, 4));
                wait for 500 ns;
                assert (f_gm = f_dut) report "Falhou!" severity ERROR;
            end loop;
        end loop;

        report "Teste Finalizado!" severity NOTE;
        wait;
    end process;

end testbench_arch;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity top_module is end top_module;

architecture top_module_arch of top_module is
    component somador_palavras_1 is
        port(
            A: in std_logic_vector (3 downto 0);
            B: in std_logic_vector (3 downto 0);
            S: out std_logic_vector (4 downto 0)
        );
    end component;

    component somador_palavras_2 is
        port(
        A: in std_logic_vector (3 downto 0);
        B: in std_logic_vector (3 downto 0);
        S: out std_logic_vector (4 downto 0)
    );
    end component;

    component testbench is
        port(
            f_dut: in std_logic_vector (4 downto 0);
            f_gm: in std_logic_vector (4 downto 0);
            A: out std_logic_vector (3 downto 0);
            B: out std_logic_vector (3 downto 0)
        );
    end component;
```

```vhdl
    signal A_1, B_1: std_logic_vector (3 downto 0);
    signal f_dut_1, f_gm_1: std_logic_vector (4 downto 0);

    begin
        obj_0: somador_palavras_1 port map (A => A_1,B => B_1, S => f_dut_1);
        obj_1: somador_palavras_2 port map (A => A_1,B => B_1, S => f_gm_1);
        obj_2: testbench port map (f_dut => f_dut_1, f_gm => f_gm_1, A => A_1, B => B_1);

end top_module_arch;
```

Compilação

Os códigos acima foram compilados para garantir seu funcionamento, através do compilador do Modelsim. Não apresentam erros de sintaxe;

```
# Loading project Relatorio5
# Compile of somador_palavras_2.vhd was successful.
# Compile of somador_palavras_1.vhd was successful.
# Compile of somador_completo.vhd was successful.
# Compile of somador_palavras_1_tb.vhd was successful.
# Compile of somador_palavras_2_tb.vhd was successful.
# Compile of testbench.vhd was successful.
# Compile of top_module.vhd was successful.
# 7 compiles, 0 failed with no errors.
```

Simulação

```
VSIM 4> run
# ** Note: Iniciando testebench...
#    Time: 0 ps  Iteration: 0  Instance: /top_module/obj_2
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /top_module/obj_1
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 0 ps  Iteration: 0  Instance: /top_module/obj_1
# ** Note: Teste Finalizado!
#    Time: 128 us  Iteration: 0  Instance: /top_module/obj_2

VSIM 5>
```

Análise

A simulação comprova que ambos os somadores de palavras desenvolvidos nas questões 1 e 2 possuem as mesmas saídas dentre as mesmas entradas. Caso fosse encontrado alguma diferença, seria impressa uma mensagem de erro no console do ModelSim.

Conclusão

Nesse experimento conseguimos com êxito realizar o testbench para verificar se o somador desenvolvido na questão 1 do experimento possui mesma saída do que o da questão 2. As simulações se comportaram da maneira esperada e não foram encontrados erros de sintaxe no código.