

Relatório 2 VHDL - Turma 05
Yan Tavares de Oliveira
202014323

Introdução

Este experimento consiste em duas etapas. Na primeira (questão 1), iremos descrever em VHDL e simular no software ModelSim um somador completo que segue as seguintes funções lógicas

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$$

Na segunda etapa (questão 2), iremos descrever em VHDL e simular no software ModelSim um multiplexador de 4 para 1, descrito pela função lógica

$$Y = D_0 \cdot \bar{S}_1 \cdot \bar{S}_0 + D_1 \cdot \bar{S}_1 \cdot S_0 + D_2 \cdot S_1 \cdot \bar{S}_0 + D_3 \cdot S_1 \cdot S_0$$

Teoria

Na questão 1, o comportamento do somador completo pode ser descrito de acordo com a imagem 1. O somador completo é utilizado para somar números binários de, no mínimo, dois dígitos.

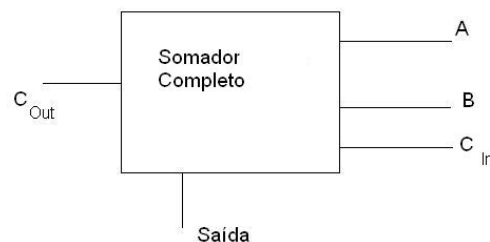


Imagem1. Representação visual de um somador completo

Na questão, devemos implementar um multiplexador 4 x 1, descrito pela imagem 2. O multiplexador em questão permite representar um número de 4 bits por meio de saídas de 1 bit e um seletor de 2 bits (representado por S).

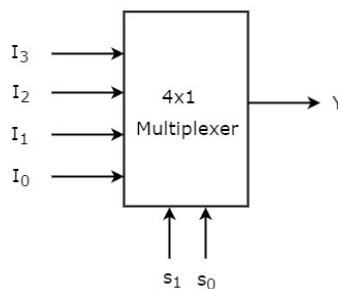
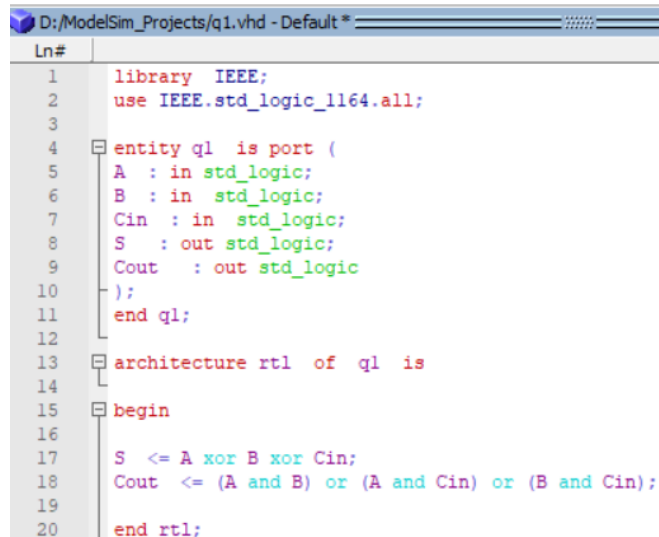


Imagem 2. Representação visual de um multiplexador 4 x 1.

Códigos

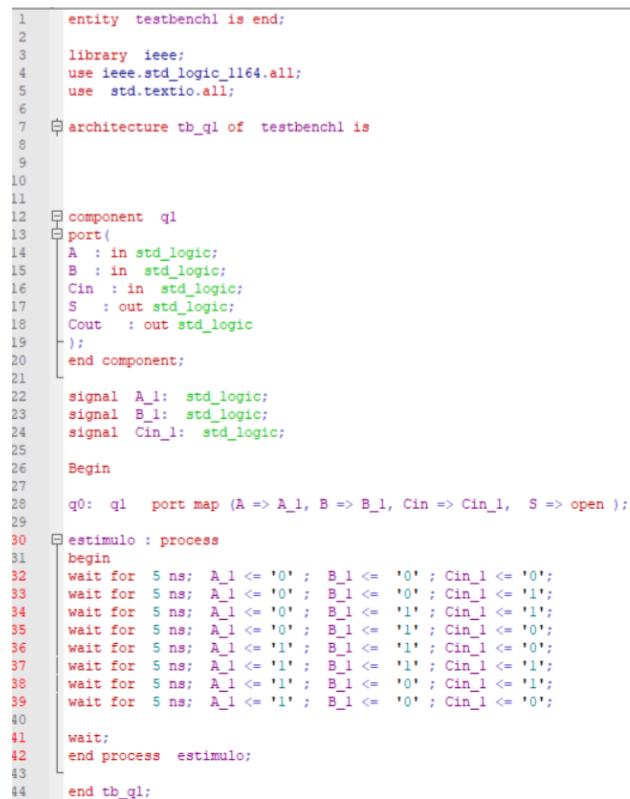
Na **questão 1**, para implementar o somador completo descrito na introdução, foi criada uma simulação no software ModelSim com 3 bits de entrada (A, B e Cin), gerando dois bits de saída (S e Cout).



```
D:/ModelSim_Projects/q1.vhd - Default *
Ln#
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  entity q1 is port (
5      A : in std_logic;
6      B : in std_logic;
7      Cin : in std_logic;
8      S : out std_logic;
9      Cout : out std_logic
10 );
11 end q1;
12
13 architecture rtl of q1 is
14
15 begin
16
17     S <= A xor B xor Cin;
18     Cout <= (A and B) or (A and Cin) or (B and Cin);
19
20 end rtl;
```

Imagem 3. Código principal da questão 1

Foi também criado um código auxiliar de testbench para gerarmos os estímulos necessários para abranger todas as combinações possíveis de entradas.



```
1  entity testbench1 is end;
2
3  library ieee;
4  use ieee.std_logic_1164.all;
5  use std.textio.all;
6
7  architecture tb_q1 of testbench1 is
8
9
10
11
12  component q1
13  port(
14      A : in std_logic;
15      B : in std_logic;
16      Cin : in std_logic;
17      S : out std_logic;
18      Cout : out std_logic
19  );
20  end component;
21
22  signal A_1: std_logic;
23  signal B_1: std_logic;
24  signal Cin_1: std_logic;
25
26  Begin
27
28  q0: q1 port map (A => A_1, B => B_1, Cin => Cin_1, S => open );
29
30  estimulo : process
31  begin
32      wait for 5 ns; A_1 <= '0' ; B_1 <= '0' ; Cin_1 <= '0';
33      wait for 5 ns; A_1 <= '0' ; B_1 <= '0' ; Cin_1 <= '1';
34      wait for 5 ns; A_1 <= '0' ; B_1 <= '1' ; Cin_1 <= '1';
35      wait for 5 ns; A_1 <= '0' ; B_1 <= '1' ; Cin_1 <= '0';
36      wait for 5 ns; A_1 <= '1' ; B_1 <= '1' ; Cin_1 <= '0';
37      wait for 5 ns; A_1 <= '1' ; B_1 <= '1' ; Cin_1 <= '1';
38      wait for 5 ns; A_1 <= '1' ; B_1 <= '0' ; Cin_1 <= '1';
39      wait for 5 ns; A_1 <= '1' ; B_1 <= '0' ; Cin_1 <= '0';
40
41      wait;
42  end process estimulo;
43
44 end tb_q1;
```

Imagem 5. Código auxiliar da questão 1

Na **questão 2**, para implementar o multiplexador descrito na introdução, foi criada uma simulação no software ModelSim com 2 vetores lógicos de entrada (D com 4 bits e S com 2 bits), gerando um bit de saída (Y).

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.all;
3
4  entity q2 is
5  port(
6
7      S : in STD_LOGIC_VECTOR(1 downto 0);
8      D: in STD_LOGIC_VECTOR(3 downto 0);
9      Y: out STD_LOGIC
10
11  );
12  end q2;
13
14  architecture rtl of q2 is
15  begin
16
17      Y <= ( D(0) and not S(1) and not S(0) ) or ( D(1) and not S(1) and S(0) ) or ( D(2) and S(1) and not S(0) ) or ( D(3) and S(1) and S(0) );
18
19  end rtl;
20
21

```

Imagem 5. Código principal da questão 2

Foi também criado um código auxiliar de testbench para gerarmos os estímulos necessários para abranger todas as combinações possíveis de entradas.

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3
4  ENTITY testbench IS
5  END testbench;
6
7  ARCHITECTURE behavior OF tb_q2 IS
8
9  COMPONENT q2
10  PORT(
11      S : in STD_LOGIC_VECTOR(1 downto 0);
12      D: in STD_LOGIC_VECTOR(0 to 3);
13      Y: out STD_LOGIC
14  );
15  END COMPONENT;
16
17  signal clkq : std_logic_vector(3 downto 0) := "0000";
18  signal S : std_logic_vector(1 downto 0) := "00";
19
20  BEGIN
21
22  uut: q2 PORT MAP (
23      D => clkq,
24      S => S,
25      Y => open
26  );
27
28  clkq(0) <= not clkq(0) after 1 ns;
29  clkq(1) <= not clkq(1) after 2 ns;
30  clkq(2) <= not clkq(2) after 4 ns;
31  clkq(3) <= not clkq(3) after 8 ns;
32
33  stim_proc: process
34  begin
35
36      wait for 16 ns;
37
38      S <= "10";
39
40      wait for 16 ns;
41
42      S <= "11";
43
44      wait for 16 ns;
45
46      S <= "01";
47
48      wait for 16 ns;
49
50      end process;
51
52  END;
53

```

Imagem 6. Código auxiliar da questão 2

Compilação

Abaixo estão as mensagens de compilação de ambos os projetos, com nenhum erro sendo apresentado em nenhum dos casos

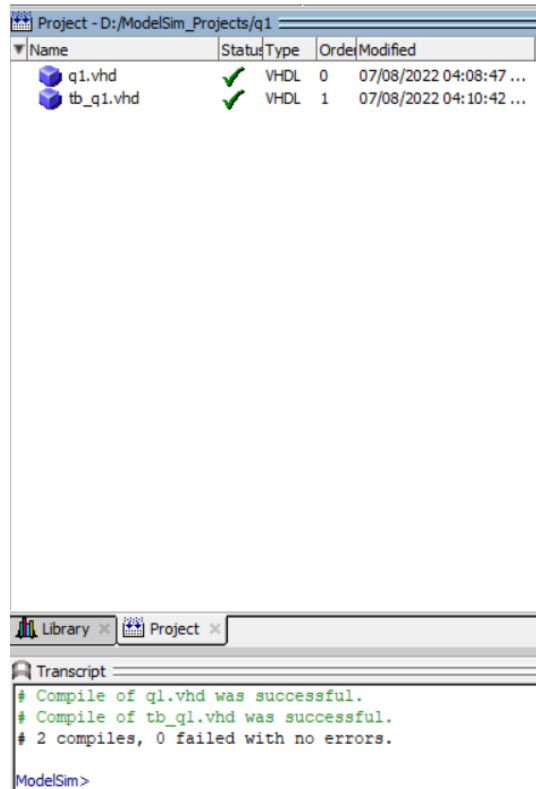


Imagem 7. Mensagem de compilação da questão 1

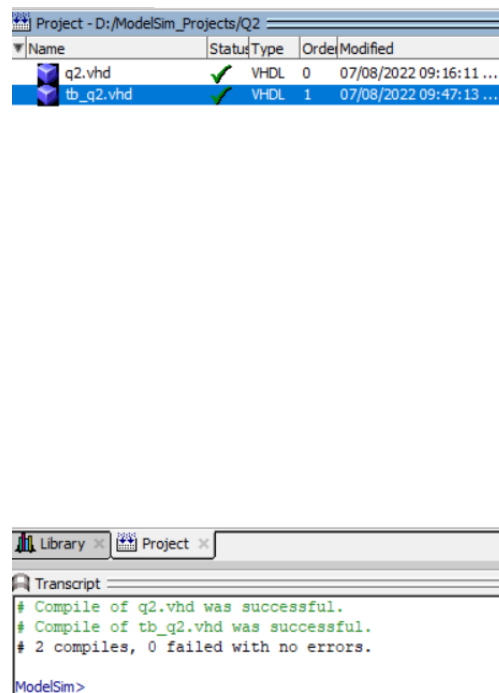


Imagem 8. Mensagem de compilação da questão 2

Simulação

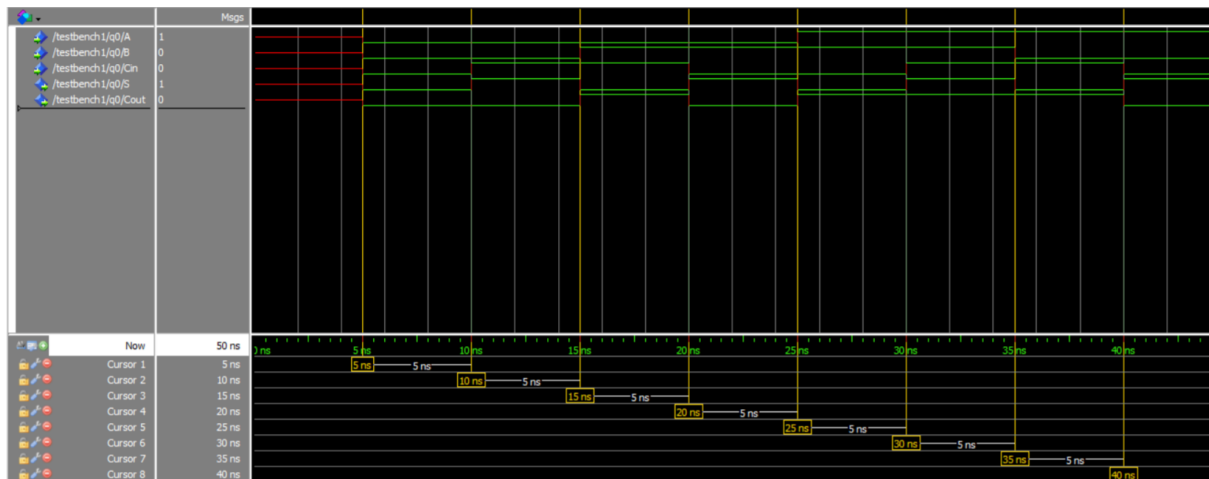


Figura 9. Simulação em forma de onda binária da questão 1.

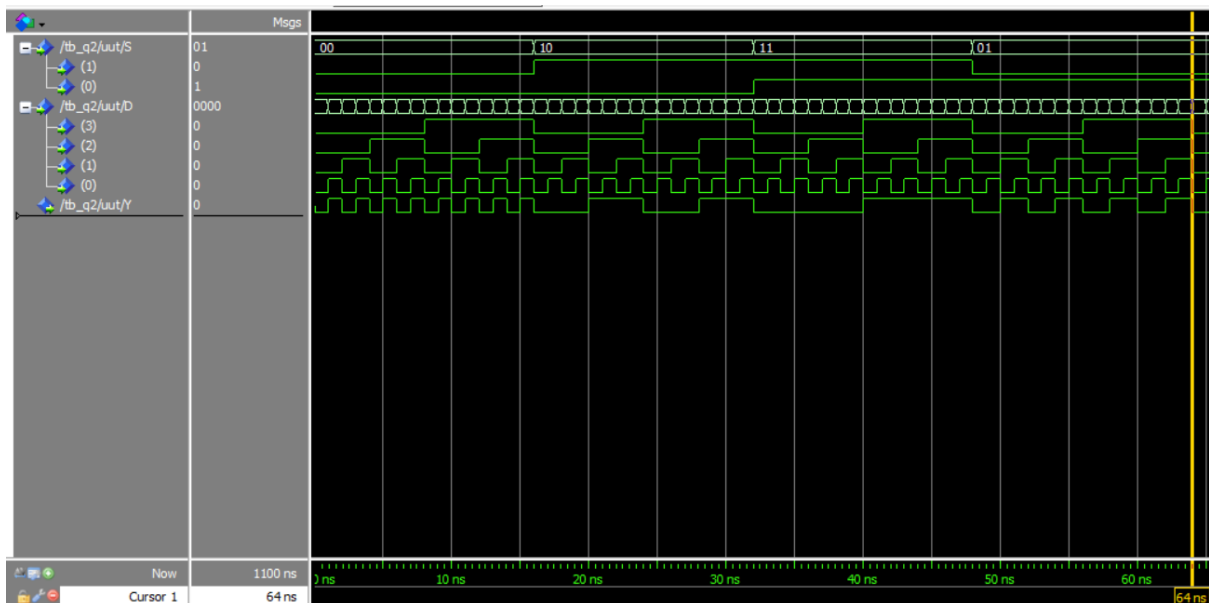
































Figura 10. Simulação em forma de onda binária da questão 2.











Análise

Para determinar a ordem dos bits utilizados na simulação, foi utilizado o Código de Gray, ou seja, aconteceu a mudança de um bit para cada iteração. No projeto 1, foi decidido um tempo de espera de 5 segundos entre cada iteração, enquanto no código 2, foi decidido um tempo de espera de 16 segundos, que é justamente um período da onda “clkq” do código q2, que abrange todas as possíveis variações presentes em D (vetor de 4 bits). Pelas simulações geradas, é possível notar que obtemos resultados esperados em ambos os casos. Abaixo estão representadas todas as variações de saídas e entradas para a questão 1.

 /testbench1/q0/A	0	 /testbench1/q0/A	0
 /testbench1/q0/B	0	 /testbench1/q0/B	0
 /testbench1/q0/Cin	1	 /testbench1/q0/Cin	0
 /testbench1/q0/S	1	 /testbench1/q0/S	0
 /testbench1/q0/Cout	0	 /testbench1/q0/Cout	0

 /testbench1/q0/A	0	 /testbench1/q0/A	0
 /testbench1/q0/B	1	 /testbench1/q0/B	1
 /testbench1/q0/Cin	0	 /testbench1/q0/Cin	1
 /testbench1/q0/S	1	 /testbench1/q0/S	0
 /testbench1/q0/Cout	0	 /testbench1/q0/Cout	1

 /testbench1/q0/A	1	 /testbench1/q0/A	1
 /testbench1/q0/B	1	 /testbench1/q0/B	1
 /testbench1/q0/Cin	1	 /testbench1/q0/Cin	0
 /testbench1/q0/S	1	 /testbench1/q0/S	0
 /testbench1/q0/Cout	1	 /testbench1/q0/Cout	1

 /testbench1/q0/A	1	 /testbench1/q0/A	1
 /testbench1/q0/B	0	 /testbench1/q0/B	0
 /testbench1/q0/Cin	0	 /testbench1/q0/Cin	1
 /testbench1/q0/S	1	 /testbench1/q0/S	0
 /testbench1/q0/Cout	0	 /testbench1/q0/Cout	1

Conclusão

Neste experimento conseguimos descrever com sucesso o comportamento de duas entidades muito utilizadas no mundo da tecnologia - Um somador completo e um multiplexador 4 x 1. Não foram encontrados erros na obtenção dos resultados ou em qualquer etapa dos experimentos.