

ENE0040 - Laboratório de Sistemas Digitais - Turma 07

Relatório do Experimento 2

Autor: Henrique Morcelles Salum

Matrícula: 232003008

Universidade de Brasília - UnB

Departamento de Engenharia Elétrica - ENE

Sumário

1	Introdução	2
1.1	Sobre o Experimento	2
1.1.1	Somador Completo	2
1.1.2	Multiplexador 4 para 1	2
1.2	Introdução Teórica	2
1.2.1	Somador Completo	2
1.2.2	Multiplexador 4 para 1	4
2	Códigos	6
2.1	Somador Completo	6
2.1.1	Descrição de Hardware	6
2.1.2	Testbench	7
2.2	Multiplexador 4 para 1	8
2.2.1	Descrição de Hardware	8
2.2.2	Testbench	9
3	Compilação	11
4	Simulação	11
5	Análise	12
5.1	Análise do Comportamento do Somador Completo	13
5.2	Análise do Comportamento do Multiplexador 4 para 1	14
6	Conclusão	15

1 Introdução

1.1 Sobre o Experimento

Este experimento é composto por duas tarefas, em ambas, deve-se implementar circuitos lógicos com VHDL e simulá-los por meio do software ModelSim, da Intel. As especificidades de cada tarefa serão exploradas a seguir.

1.1.1 Somador Completo

Na primeira tarefa, implementa-se um somador completo. As funções booleanas dos sinais de saída S e C_{out} são definidas em função das entradas A , B e C_{in} como segue:

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

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

1.1.2 Multiplexador 4 para 1

Na segunda tarefa, deve-se implementar um multiplexador de 4 para 1, utilizando dois vetores de entrada: S , de dois sinais e D , de quatro sinais. A saída é um sinal Y definido da seguinte forma:

$$Y = D_0 \cdot \overline{S_1} \cdot \overline{S_0} + D_1 \cdot \overline{S_1} \cdot S_0 + D_2 \cdot S_1 \cdot \overline{S_0} + D_3 \cdot S_1 \cdot S_0$$

1.2 Introdução Teórica

1.2.1 Somador Completo

Antes de definir o somador completo (*Full Adder*), vamos definir o meio somador (*Half Adder*). O meio somador é um circuito lógico combinacional que realiza a soma entre um bit A e um bit B . O valor máximo da soma de dois bits A e B é quando $A = 1$ e $B = 1$. O resultado dessa soma é 2, representado como 10 em binário. Dessa forma, são necessárias duas saídas para o meio somador: uma para o valor da soma na posição em que estamos e uma para o possível valor da próxima posição, que é igual a 1 quando as duas entradas são 1.

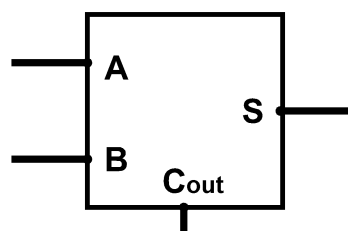


Figura 1: Representação externa do meio somador

Esse somador, porém, só consegue somar números de um bit. O que fazer, então, se quisermos somar números grandes de vários bits? Para cada posição das representações binárias das entradas, precisamos de um somador para realizar, além da soma entre o bit dessa posição de cada entrada, a soma com o *vai um* da casa anterior - chamaremos de C_{in} . O somador que cumpre esses requisitos é chamado somador completo.

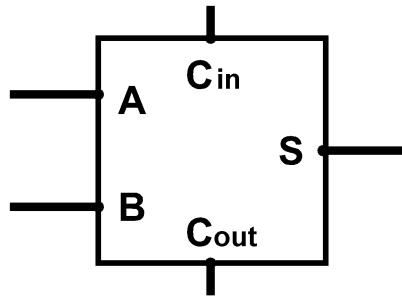


Figura 2: Representação externa do somador completo

Note que a representação acima abstrai a lógica que relaciona as saídas do somador às suas entradas. Podemos representar o somador completo pela sua representação interna, que explicita a relação entre entradas e saídas. Essa representação consiste na tradução das equações mostradas na seção “Sobre o Experimento” para um esquemático, construído em Logisim.

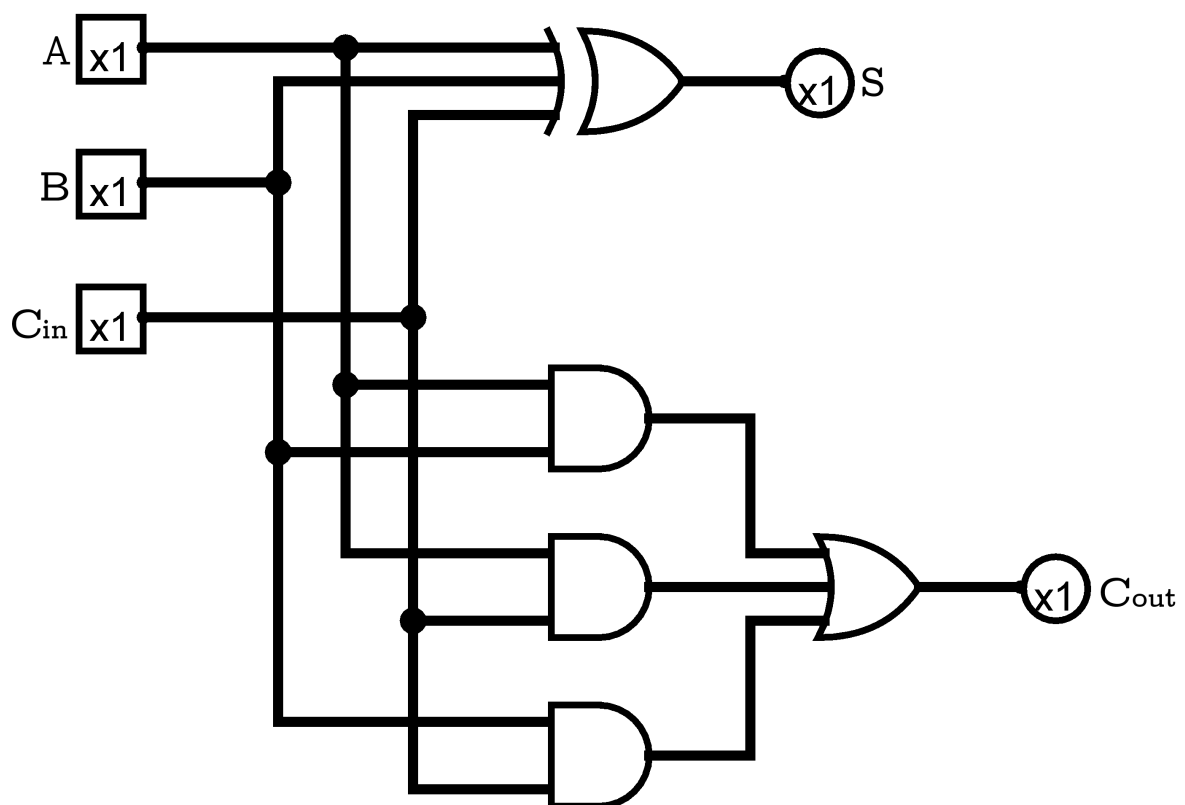


Figura 3: Representação interna do somador completo

Para realizar a soma de números grandes, basta encasquetarmos somadores completos. O número de somadores necessários é o número de casas das entradas e a entrada C_{in} de cada somador é a saída C_{out} do somador da posição anterior, com exceção da primeira casa, que tem como entrada $C_{in} = 0$. A tabela-verdade do somador completo é a que segue:

Entradas			Saídas	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tabela 1: Tabela-verdade do somador completo

1.2.2 Multiplexador 4 para 1

Um multiplexador, também chamado de Mux, é um dispositivo digital que seleciona uma entre várias entradas de dados como saída. Essencialmente, o multiplexador atua como um “comutador” controlado por entradas de seleção, determinando qual sinal será encaminhado para a saída. Esse dispositivo é extensivamente usado no projeto de máquinas de estados, pois ele é muito conveniente quando é necessário implementar diferentes lógicas dependendo do estado atual.

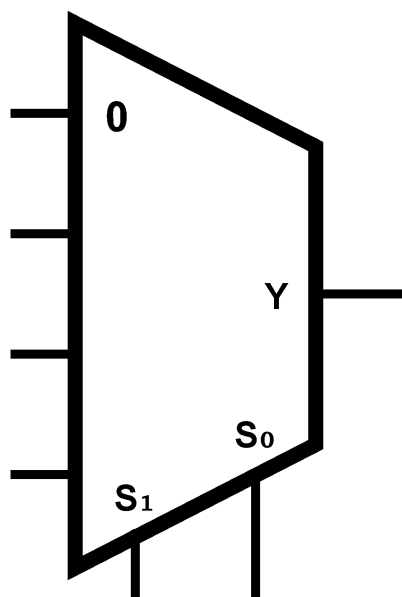


Figura 4: Representação externa do multiplexador 4 x 1

Para entender esse dispositivo, podemos separar as entradas entre entradas de dados, que são as entradas de informações a serem selecionadas, e as entradas de seleção, que são utilizadas para escolher qual das entradas de dados será enviada para a saída. O número de entradas de seleção pode ser calculado em função do número de entradas de dados pela função:

$$f(n) = \lceil \log_2 n \rceil$$

em que n é o número de entradas de dados.

Nesse experimento, implementamos, especificamente, um multiplexador 4 para 1, ou seja, um multiplexador com quatro entradas de dados e, consequentemente, $\log_2 4 = 2$ entradas de seleção, além de um sinal de saída. A representação externa do Mux 4x1 já foi apresentada e a interna está a seguir:

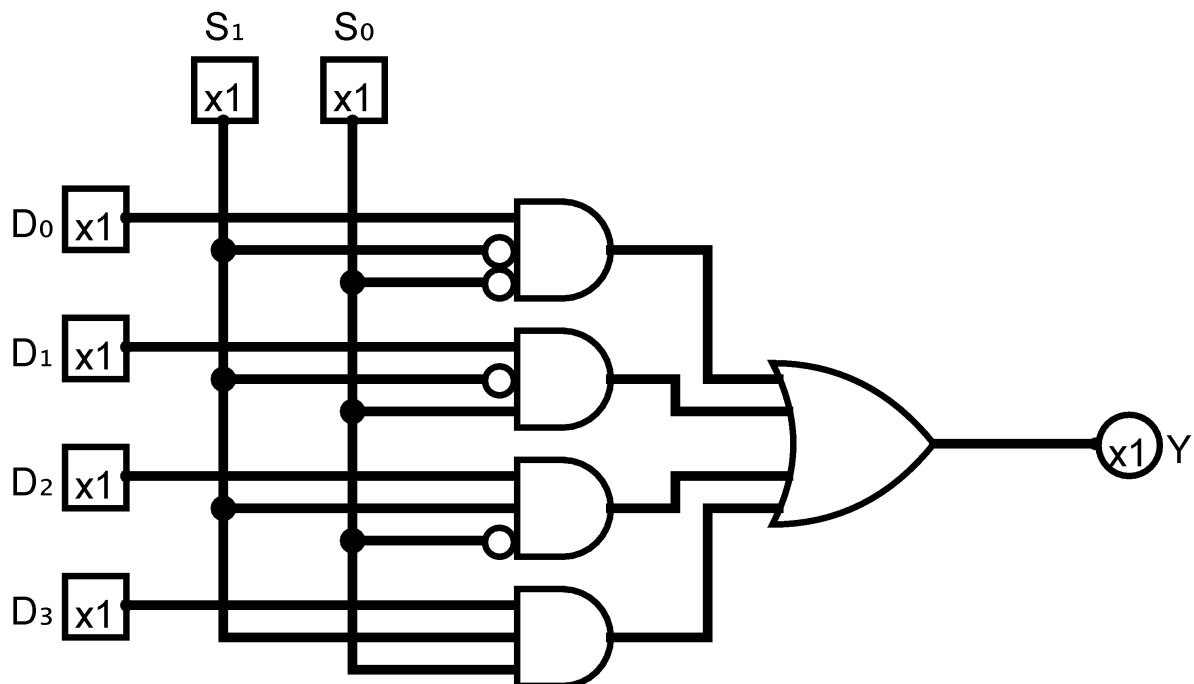


Figura 5: Representação interna do multiplexador 4x1

A tabela-verdade do Mux 4x1, na qual introduzimos a entrada de dados selecionada na saída, ao invés de fazer a tabela para cada variação dessa entrada, é a que segue:

S_1	S_0	Saída
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Tabela 2: Tabela-verdade do multiplexador 4x1

2 Códigos

Nessa seção, serão apresentados os códigos desenvolvidos, em VHDL, para implementar os sistemas digitais apresentados na introdução. Os códigos referentes ao *design* dos sistemas digitais são apresentados aqui com alguns comentários, os *testbenches*, com bem menos. Os códigos enviados junto a este relatório, porém, estão todos devidamente comentados - aqui, alguns comentários foram suprimidos para melhor apresentação.

2.1 Somador Completo

Para implementar o somador completo descrito anteriormente, foi necessário desenvolver, em VHDL, o seu **arquivo de modelo de *hardware***, composto pela entidade (*entity*), que estabelece a interface do dispositivo, e pela arquitetura (*architecture*), que estabelece a sua lógica interna.

2.1.1 Descrição de Hardware

O código escrito no arquivo de *design* do somador completo é o que segue:

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 -- Criação da entidade, que declara entradas e saídas do sistema
5 entity somadorCompleto is
6     port (
7         A, B, Cin: in std_logic; -- Entradas
8         S, Cout: out std_logic -- Saídas
9     );
10 end somadorCompleto;
11
12 -- Criação da arquitetura, que estabelece a lógica entre as entradas e
13    ↳ saídas
14 architecture behavioral of somadorCompleto is
15 begin
16     S <= A xor B xor Cin; -- Lógica da saída S
17     Cout <= (A and B) or (A and Cin) or (B and Cin); -- Lógica da saída
18    ↳ Cout
19 end architecture behavioral;
```

Código 1: Código para implementação do somador completo

A *entity* no código acima é uma representação equivalente à exposta na **Figura 2**, enquanto a *architecture* é equivalente à representação exposta na **Figura 3**.

2.1.2 Testbench

Além do código acima, foi necessário implementar, também em VHDL, um *testbench* para que se pudesse testar o comportamento e a funcionalidade do circuito digital projetado. Com ele, podemos gerar todas as combinações de sinais possíveis e analisar o comportamento do circuito sob cada um desses estímulos. O código para implementar esse *testbench* é o que segue:

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 -- A entidade do testbench é vazia pois ele nunca será instanciado
5 entity tb_somadorCompleto is
6 end tb_somadorCompleto;
7
8 architecture testbench of tb_somadorCompleto is
9     -- Declaração do componente "somadorCompleto". Pense num componente
10     --   ↳ como uma classe
11     component somadorCompleto is
12         port (
13             A, B, Cin: in std_logic;
14             S, Cout: out std_logic
15         );
16     end component somadorCompleto;
17     -- Sinais internos do testbench
18     signal A_tb: std_logic := '0';
19     signal B_tb: std_logic := '0';
20     signal Cin_tb: std_logic := '0';
21 begin
22     -- Inicialização da instância do componente - como se fosse um objeto
23     instancia_somador : component somadorCompleto
24         port map (
25             A => A_tb,
26             B => B_tb,
27             Cin => Cin_tb,
28             S => open,
29             Cout => open
```



```
29     );
30     -- Processo para variar as entradas
31     estimulos: process
32         variable i: integer := 0;
33     begin
34         if i /= 0 then
35             if i mod 2 = 0 then Cin_tb <= not Cin_tb;
36             end if;
37             if i mod 4 = 0 then B_tb <= not B_tb;
38             end if;
39             if i mod 8 = 0 then A_tb <= not A_tb;
40             end if;
41         end if;
42
43         i := i + 1;
44         wait for 6.25 ns;
45     end process estimulos;
46 end architecture testbench;
```

Código 2: Testbench para o somador completo

No código acima, o somador completo construído é instanciado e utilizado como uma “caixa-preta”; associamos às suas entradas sinais do *testbench*, o que é equivalente a conectar cabos nas entradas de um dispositivo digital, e, dentro do *process*, fazemos esses sinais oscilarem. Dessa forma, conseguimos testar o dispositivo criado.

2.2 Multiplexador 4 para 1

Como pôde ser visto, nos códigos relativos ao somador completo, a abordagem foi bastante de baixo nível: não utilizamos abstrações fornecidas pela linguagem VHDL, a função foi implementada diretamente pela equação correspondente e o *testbench* gerou todas as oito possíveis combinações das três entradas. A abordagem para o multiplexador foi diferente, como será observado nos códigos subsequentes.

2.2.1 Descrição de Hardware

O código desenvolvido em VHDL para implementar a “caixa-preta” do multiplexador 4 para 1 é o que segue:

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
```

```

3
4 entity Mux4x1 is
5     port (
6         D: in std_logic_vector(3 downto 0); -- Entradas de dados
7         S: in std_logic_vector(1 downto 0); -- Entradas de seleção
8         Y: out std_logic -- Saída
9     );
10 end entity Mux4x1;
11
12 architecture behavioral of Mux4x1 is
13 begin
14     Y <= D(3) when (S = "11") else -- Lógica da saída
15         D(2) when (S = "10") else -- definida em função
16         D(1) when (S = "01") else -- das entradas de
17         D(0) when (S = "00") else -- seleção
18         '-'; -- don't care quando as entradas de seleção são don't
19         ↪ cares, weak, alta impedância, etc.
20 end architecture behavioral;

```

Código 3: Código para implementação do multiplexador 4x1

Aqui, ao invés de implementar a função pela equação correspondente, foi utilizada uma construção de controle de fluxo ‘when’, em que apenas se definiu o comportamento do circuito para as combinações das entradas de seleção, além de se ter considerado apenas alguns valores - 0 e 1 - para elas, fazendo com que as outras possibilidades - X, Z, H, ... - gerassem saída *don't care* (-). O *testbench* do multiplexador é o que segue.

Analogamente ao somador completo, no multiplexador 4 para 1, a entidade (*entity*) é equivalente à representação da Figura 4 e a arquitetura (*architecture*), à da Figura 5.

2.2.2 Testbench

O código referente ao *testbench* do multiplexador é o que segue:

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 entity tb_Mux4x1 is
6 end entity tb_Mux4x1;

```

```
7
8 architecture testbench of tb_Mux4x1 is
9     component Mux4x1 is
10         port (
11             D: in std_logic_vector(3 downto 0);
12             S: in std_logic_vector(1 downto 0);
13             Y: out std_logic
14         );
15     end component Mux4x1;
16
17     signal D_tb: std_logic_vector(3 downto 0) := (others => '1');
18     signal S_tb: std_logic_vector(1 downto 0) := (others => '0');
19 begin
20     instancia_Mux4x1: component Mux4x1
21         port map (
22             D => D_tb,
23             S => S_tb,
24             Y => open
25         );
26
27     -- Processo para variar as entradas de seleção
28     estimulos_seletoras: process
29         variable s_unsigned: unsigned(1 downto 0);
30     begin
31         wait for 25 ns;
32         s_unsigned := unsigned(S_tb);
33         s_unsigned := s_unsigned + 1;
34         S_tb <= std_logic_vector(s_unsigned);
35     end process estimulos_seletoras;
36
37     -- Processo para variar as entradas selecionadas
38     variacao_dados: process
39         variable i: integer;
40     begin
41         wait for 12.5 ns;
42         -- 0 índice do elemento do vetor D que é selecionado coincide com
43         -- ↪ o valor do vetor S em inteiro
44         i := to_integer(unsigned(S_tb));
45         D_tb(i) <= not D_tb(i);
```

```

45     wait on S_tb;
46     end process variacao_dados;
47 end architecture testbench;

```

Código 4: Testbench para o multiplexador 4x1

Ao contrário do *testbench* anterior, aqui não são geradas todas as combinações possíveis dos sinais de entrada. Isso se deve à natureza do multiplexador: nele, é irrelevante a variação dos sinais não selecionados em um momento da simulação. Portanto, basta que geremos todas as combinações das entradas de seleção e se verifique se o sinal de saída corresponde à entrada de dados que deve ser selecionada por essa combinação. Para que essa simulação seja mais completa, fez-se com que os sinais de dados selecionados em um momento da simulação oscilassem entre 0 e 1.

3 Compilação

Após escrever os códigos, é necessário compilá-los pelo ModelSim para que se possa simular os sistemas digitais discutidos. Caso a compilação tenha sucesso, sabemos que não houve erros nos códigos apresentados, mas ainda não podemos afirmar que a lógica para implementar os circuitos está correta; isso será analisado nas próximas seções. A seguir, está a mensagem de compilação dos códigos apresentados acima, sem nenhum erro, como pode ser visto no terminal no canto inferior da figura.

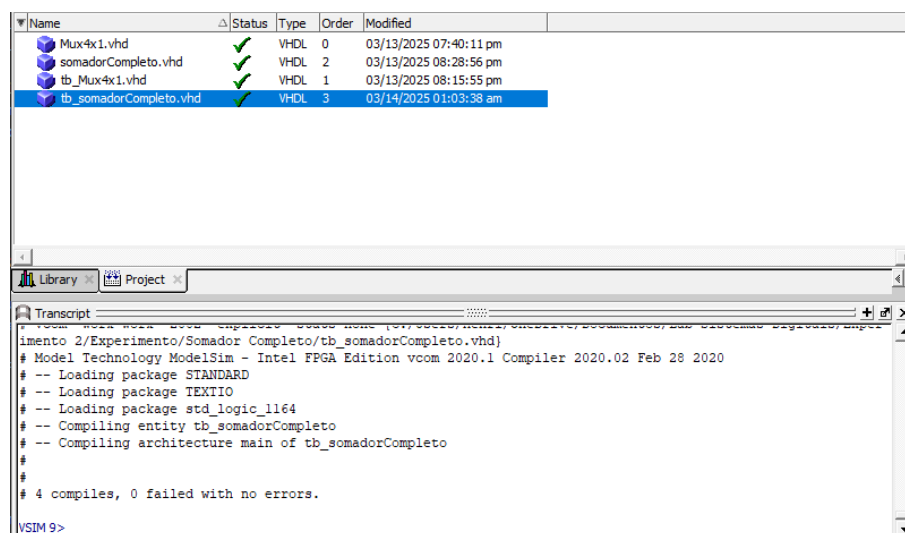


Figura 6: Compilação de todos os códigos apresentados

4 Simulação

Após compilar com sucesso os códigos apresentados, utilizamos o ModelSim para simular o comportamento dos sistemas descritos por eles. Lá, utilizamos a aba “Waves” para

analisar o comportamento dos sinais de saída para cada combinação de valores dos sinais de entrada. A forma como as entradas variam segue o que definimos nos *testbenches*, então note que as ondas referentes ao Mux 4x1 não representam todas as combinações das seis entradas. Seguem as imagens das simulações no ModelSim dos sistemas projetados.

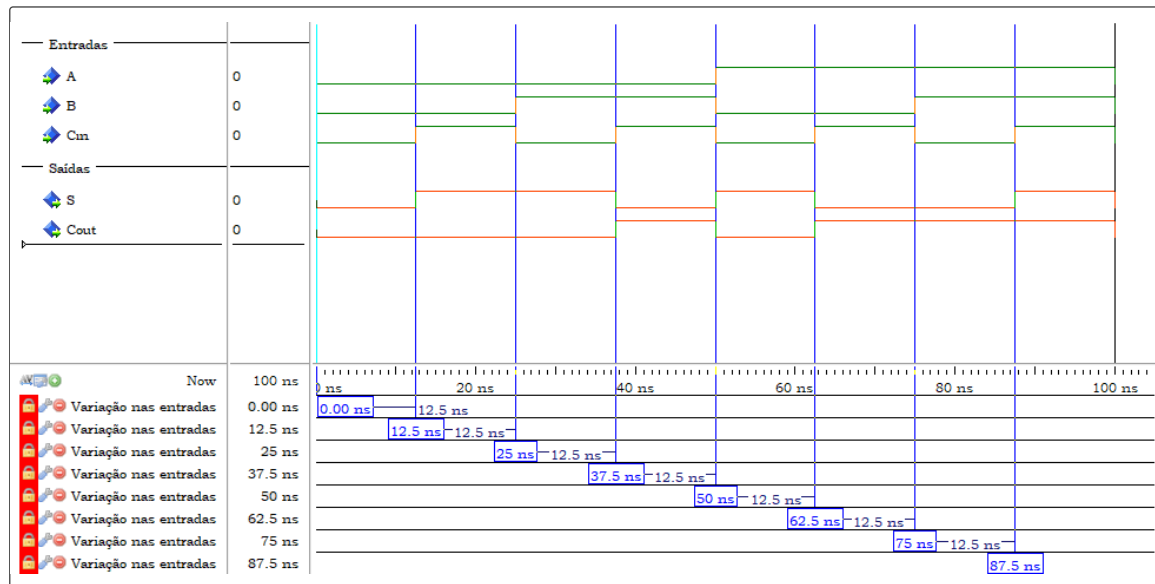


Figura 7: Simulação em forma de onda binária do somador completo

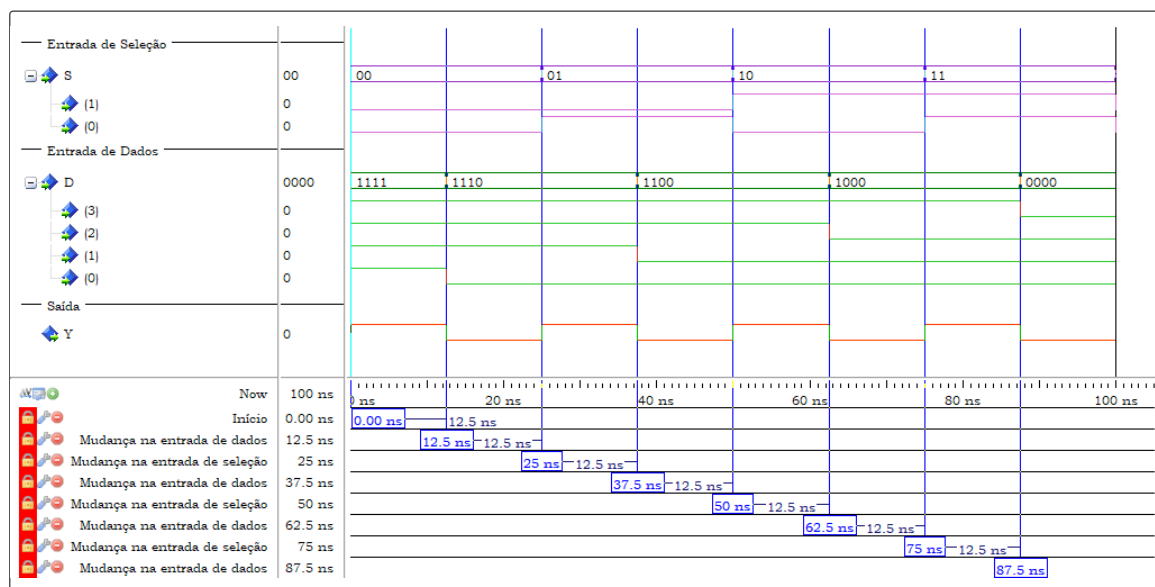


Figura 8: Simulação em forma de onda binária do multiplexador

5 Análise

Para analisar os resultados obtidos, basta olhar para os valores das ondas das entradas e saídas entre cada mudança dos sinais de entrada. Os cursores estão posicionados de forma

que, ao selecionar cada um deles, vemos todas as possíveis combinações das entradas e saídas correspondentes na aba à esquerda da tela.

5.1 Análise do Comportamento do Somador Completo

Abaixo, exibimos os *prints* que denotam o comportamento dos sinais de saída S e C_{out} para cada combinação dos sinais de entrada A , B e C_{in} .

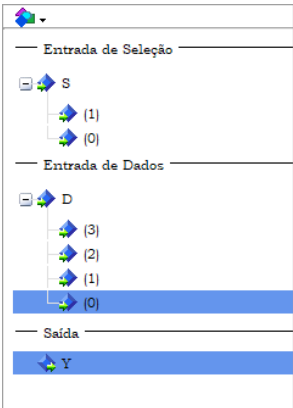
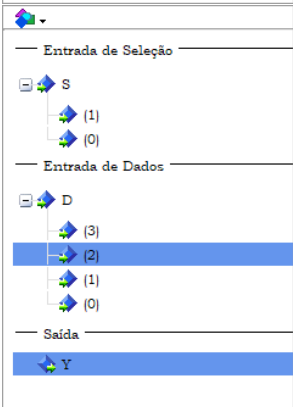
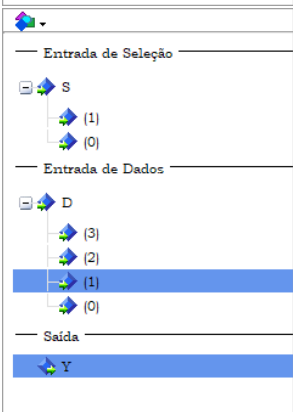
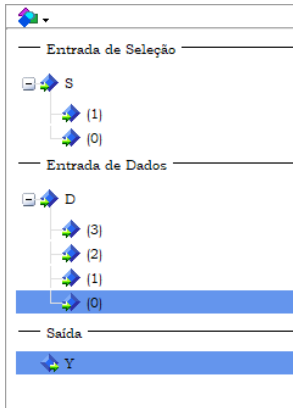
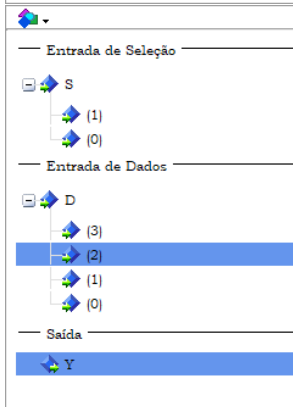
<div><div></div><div>Entradas</div><div>A0B0Cin0</div><div>Saídas</div><div>S0Cout0</div></div>	Msgs	<div><div></div><div>Entradas</div><div>A0B0Cin1</div><div>Saídas</div><div>S1Cout0</div></div>	Msgs
<div><div></div><div>Entradas</div><div>A0B1Cin0</div><div>Saídas</div><div>S1Cout0</div></div>	Msgs	<div><div></div><div>Entradas</div><div>A0B1Cin1</div><div>Saídas</div><div>S0Cout1</div></div>	Msgs
<div><div></div><div>Entradas</div><div>A1B0Cin0</div><div>Saídas</div><div>S1Cout0</div></div>	Msgs	<div><div></div><div>Entradas</div><div>A1B0Cin1</div><div>Saídas</div><div>S0Cout1</div></div>	Msgs
<div><div></div><div>Entradas</div><div>A1B1Cin0</div><div>Saídas</div><div>S0Cout1</div></div>	Msgs	<div><div></div><div>Entradas</div><div>A1B1Cin1</div><div>Saídas</div><div>S1Cout1</div></div>	Msgs

Figura 9: Todas as combinações possíveis de entradas e as saídas correspondentes do somador completo

Percebe-se que o comportamento está de acordo com o esperado: a concatenação das saídas C_{out} e S corresponde à soma dos bits de entrada A , B e C_{in} .

5.2 Análise do Comportamento do Multiplexador 4 para 1

Para analisar os resultados da simulação do Mux 4x1, observou-se a saída *Y* para cada combinação da entrada de seleção. Abaixo, estão exibidos os *prints* que mostram o comportamento do circuito simulado pelo ModelSim para cada uma dessas combinações. Perceba que cada linha representa uma combinação das entradas de seleção e, portanto, uma entrada de dados selecionada, enquanto cada coluna representa um possível valor do sinal de entrada de dados selecionado.

	<div>Msgs</div> <table><tr><td>Entrada de Seleção</td><td></td></tr><tr><td>S</td><td>00</td></tr><tr><td>(1)</td><td>0</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Entrada de Dados</td><td></td></tr><tr><td>D</td><td>1111</td></tr><tr><td>(3)</td><td>1</td></tr><tr><td>(2)</td><td>1</td></tr><tr><td>(1)</td><td>1</td></tr><tr><td>(0)</td><td>1</td></tr><tr><td>Saída</td><td></td></tr><tr><td>Y</td><td>1</td></tr></table>	Entrada de Seleção		S	00	(1)	0	(0)	0	Entrada de Dados		D	1111	(3)	1	(2)	1	(1)	1	(0)	1	Saída		Y	1
Entrada de Seleção																									
S	00																								
(1)	0																								
(0)	0																								
Entrada de Dados																									
D	1111																								
(3)	1																								
(2)	1																								
(1)	1																								
(0)	1																								
Saída																									
Y	1																								
	<div>Msgs</div> <table><tr><td>Entrada de Seleção</td><td></td></tr><tr><td>S</td><td>10</td></tr><tr><td>(1)</td><td>1</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Entrada de Dados</td><td></td></tr><tr><td>D</td><td>1100</td></tr><tr><td>(3)</td><td>1</td></tr><tr><td>(2)</td><td>1</td></tr><tr><td>(1)</td><td>0</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Saída</td><td></td></tr><tr><td>Y</td><td>1</td></tr></table>	Entrada de Seleção		S	10	(1)	1	(0)	0	Entrada de Dados		D	1100	(3)	1	(2)	1	(1)	0	(0)	0	Saída		Y	1
Entrada de Seleção																									
S	10																								
(1)	1																								
(0)	0																								
Entrada de Dados																									
D	1100																								
(3)	1																								
(2)	1																								
(1)	0																								
(0)	0																								
Saída																									
Y	1																								
	<div>Msgs</div> <table><tr><td>Entrada de Seleção</td><td></td></tr><tr><td>S</td><td>01</td></tr><tr><td>(1)</td><td>0</td></tr><tr><td>(0)</td><td>1</td></tr><tr><td>Entrada de Dados</td><td></td></tr><tr><td>D</td><td>1110</td></tr><tr><td>(3)</td><td>1</td></tr><tr><td>(2)</td><td>1</td></tr><tr><td>(1)</td><td>1</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Saída</td><td></td></tr><tr><td>Y</td><td>1</td></tr></table>	Entrada de Seleção		S	01	(1)	0	(0)	1	Entrada de Dados		D	1110	(3)	1	(2)	1	(1)	1	(0)	0	Saída		Y	1
Entrada de Seleção																									
S	01																								
(1)	0																								
(0)	1																								
Entrada de Dados																									
D	1110																								
(3)	1																								
(2)	1																								
(1)	1																								
(0)	0																								
Saída																									
Y	1																								
	<div>Msgs</div> <table><tr><td>Entrada de Seleção</td><td></td></tr><tr><td>S</td><td>10</td></tr><tr><td>(1)</td><td>1</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Entrada de Dados</td><td></td></tr><tr><td>D</td><td>1000</td></tr><tr><td>(3)</td><td>1</td></tr><tr><td>(2)</td><td>0</td></tr><tr><td>(1)</td><td>0</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Saída</td><td></td></tr><tr><td>Y</td><td>0</td></tr></table>	Entrada de Seleção		S	10	(1)	1	(0)	0	Entrada de Dados		D	1000	(3)	1	(2)	0	(1)	0	(0)	0	Saída		Y	0
Entrada de Seleção																									
S	10																								
(1)	1																								
(0)	0																								
Entrada de Dados																									
D	1000																								
(3)	1																								
(2)	0																								
(1)	0																								
(0)	0																								
Saída																									
Y	0																								
	<div>Msgs</div> <table><tr><td>Entrada de Seleção</td><td></td></tr><tr><td>S</td><td>01</td></tr><tr><td>(1)</td><td>0</td></tr><tr><td>(0)</td><td>1</td></tr><tr><td>Entrada de Dados</td><td></td></tr><tr><td>D</td><td>1100</td></tr><tr><td>(3)</td><td>1</td></tr><tr><td>(2)</td><td>1</td></tr><tr><td>(1)</td><td>0</td></tr><tr><td>(0)</td><td>0</td></tr><tr><td>Saída</td><td></td></tr><tr><td>Y</td><td>0</td></tr></table>	Entrada de Seleção		S	01	(1)	0	(0)	1	Entrada de Dados		D	1100	(3)	1	(2)	1	(1)	0	(0)	0	Saída		Y	0
Entrada de Seleção																									
S	01																								
(1)	0																								
(0)	1																								
Entrada de Dados																									
D	1100																								
(3)	1																								
(2)	1																								
(1)	0																								
(0)	0																								
Saída																									
Y	0																								

	Msgs		Msgs
Entrada de Seleção		Entrada de Seleção	
S	11	S	11
(1)	1	(1)	1
(0)	1	(0)	1
Entrada de Dados		Entrada de Dados	
D	1000	D	0000
(3)	1	(3)	0
(2)	0	(2)	0
(1)	0	(1)	0
(0)	0	(0)	0
Saída		Saída	
Y	1	Y	0

Figura 10: Saídas para todas as combinações das entradas de seleção e de dados selecionadas

Nota-se, em especial, os sinais destacados em cada coluna - sempre o sinal de saída Y e o sinal de entrada selecionado D_n . É fácil perceber que aquele é igual a este para qualquer combinação das entradas de seleção, ou seja, que a saída é igual ao sinal selecionado, como é, por definição, em um multiplexador.

6 Conclusão

Nesse experimento, simulou-se, com sucesso, dois dispositivos digitais de suma importância para a eletrônica digital: o somador completo e o multiplexador 4 para 1. No relatório, percebeu-se o sucesso do experimento e pôde-se analisar devidamente os resultados obtidos pelas simulações, por meio das quais foi possível analisar minuciosamente esses dois circuitos e o seu comportamento para cada estímulo possível.

Concluindo, não foram observadas discrepâncias entre o comportamento esperado e o observado, podendo-se concluir, portanto, o sucesso do experimento.