Laboratório de Sistemas Digitais – 2025.1

Relatório 4 – Projeto Modular

João Pedro Fernandes Santos (222025342)

Turma 09

Introdução/Objetivos

Neste experimento de Sistemas Digitais, implementamos funções combinacionais em VHDL para explorar multiplexadores 4x1, inversores e decodificadores 4x16. Na primeira parte, utilizamos inversores e MUX 4x1 para gerar as saídas X e Y a partir das entradas A, B e C; na segunda, combinamos um decodificador 4x16 e um MUX 8x1 para obter a saída Z. A atividade enfatiza a codificação modular, a elaboração de tabelas-verdade e a validação por simulações no ModelSim. Ao finalizar, espera-se desenvolver a habilidade de mapear expressões booleanas em hardware e interpretar criticamente os resultados de simulação.

1. Atividade 1

1.1 Descrição da atividade

A primeira atividade propôs a implementação, em VHDL e simulação no ModelSim, de duas funções booleanas de três variáveis (X e Y) utilizando exclusivamente dois multiplexadores 4-para-1 e um inversor. Um multiplexador 4x1 é um circuito combinacional que seleciona uma entre quatro entradas de dados (D0 a D3) para encaminhar à saída (Y) com base em um vetor de seleção de dois bits (S1, S0). Na arquitetura estrutural adotada, cada componente MUX4X1 foi instanciado via port map e suas entradas de dados foram conectadas a constantes lógicas (‘0’ ou ‘1’) ou ao sinal C invertido, gerado pelo componente PORTA_INV, enquanto os sinais A e B foram concatenados para formar o vetor de seleção. Dessa forma, o multiplexador funciona como uma chave programável que escolhe diretamente o termo de produto necessário para sintetizar as expressões: $\mathbf{X = \neg A \cdot B \cdot C + A \cdot \neg B \cdot \neg C + A \cdot B}$ e $\mathbf{Y = \neg A \cdot \neg B + \neg A \cdot B \cdot \neg C + A \cdot B \cdot C}$. Sem qualquer lógica adicional na entidade principal.

Tabela 1: Tabela-verdade das expressões X e Y:

A	B	C	X	Y
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

1.2 Implementação em VHDL

A Listagem 1 apresenta o código criado para esta atividade.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SISTEMA IS
PORT (
    A, B, C : IN STD_LOGIC;
    X, Y   : OUT STD_LOGIC
  );
END SISTEMA;
ARCHITECTURE SISTEMA_ARCH OF SISTEMA IS
COMPONENT PORTA_INV IS
PORT (
      A : IN STD_LOGIC;
      Y : OUT STD_LOGIC
    );
  END COMPONENT;
COMPONENT MUX4X1 IS
PORT (
      S : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
      D : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      Y : OUT STD_LOGIC);
 END COMPONENT;

SIGNAL A_NOT, B_NOT, C_NOT :
STD_LOGIC;SIGNAL muxX_in, muxY_in :
STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL SIG: STD_LOGIC_VECTOR(1 DOWNTO 0);
```

```
BEGIN

INV_A: PORTA_INV PORT MAP (A=>A, Y=>A_NOT);
INV_B: PORTA_INV PORT MAP (A=>B, Y=>B_NOT);
INV_C: PORTA_INV PORT MAP (A=>C, Y=>C_NOT);
SIG <= A & B;
MUXX_IN(0) <= '0';
  MUXX_IN(1) <= C;
  MUXX_IN(2) <= C_NOT;
  MUXX_IN(3) <='1';
 MUXX: MUX4X1 PORT MAP (
    S => SIG,
    D => MUXX_IN,
    Y => X
  );
MUXY_IN(0) <= '1';
  MUXY_IN(1) <= C_NOT;
  MUXY_IN(2) <= '0';
  MUXY_IN(3) <= C;
  MUXY: MUX4X1 PORT MAP (
    S => SIG,
    D => MUXY_IN,
    Y => Y );
END SISTEMA_ARCH;
```

Listagem 1: Implementação do circuito para resolução das expressões X e Y.

O código define uma entidade VHDL chamada **SISTEMA** com três entradas lógicas (A, B e C) e duas saídas (X e Y). Na arquitetura estrutural **SISTEMA_ARCH**, são declarados dois componentes principais: um inversor genérico **PORTA_INV** e um multiplexador 4×1 **MUX4X1**. Três instâncias de **PORTA_INV** produzem as versões negadas de A, B e C, respectivamente (A_NOT, B_NOT e C_NOT), usando mapeamento posicional de portas. Em seguida, A e B são concatenados para formar o vetor de seleção de dois bits **SIG**, que servirá de controle para ambos os multiplexadores. Todos os sinais intermediários — vetores **muxX_in**, **muxY_in** e o próprio **SIG** — são declarados como **SIGNAL**, garantindo comunicação interna entre componentes.

Para sintetizar as funções lógicas desejadas, cada MUX4X1 recebe em **D(0…3)** constantes lógicas (‘0’ ou ‘1’) ou o sinal C em sua forma direta e invertida, de modo que o valor de saída corresponda exatamente a um dos produtos das expressões. No primeiro multiplexador (instância **MUXX**), a combinação de SIG escolhe entre ‘0’, C, ¬C ou ‘1’ para gerar X; no segundo (instância **MUXY**), SIG seleciona entre ‘1’, ¬C, ‘0’ ou C para produzir Y. Esta implementação foi feita com base na decomposição das expressões de X e Y em termos que dependem apenas de A e B e de C ou ¬C.

É importante lembrar que os componentes MUX4X1 e PORTA_INV, não tem sua lógica no código principal do sistema, eles estão modularizados presentes em outros arquivos presentes no Model sim, são eles:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity PORTA_INV is
 port (
  A : in  STD_LOGIC;
  Y : out STD_LOGIC
 );
end PORTA_INV;


architecture PORTA_INV_ARCH of PORTA_INV is
begin
 Y <= not A;
end PORTA_INV_ARCH;
```

Listagem 2 : Código do componente: PORTA_INV

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;


ENTITY MUX4X1 IS

PORT (S: IN STD_LOGIC_VECTOR(0 TO 1);

D:IN STD_LOGIC_VECTOR(0 TO 3);

Y: OUT STD_LOGIC);

END MUX4X1;


ARCHITECTURE MUX4X1_ARCH OF MUX4X1 IS

BEGIN

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));

END MUX4X1_ARCH;
```

Listagem 3: Código do componente: MUX4X1

1.3 Simulação

Para verificar a corretude do código implementado, a entidade SISTEMA foi submetida a uma simulação no ModelSim em que os sinais de entrada A, B e C receberam ondas quadradas com períodos de 50 ms, 100 ms e 200 ms, respectivamente. Dessa forma, durante cada intervalo de 200 ms, todas as oito combinações possíveis de A, B e C foram exercitadas de maneira sistemática: A alterna rapidamente, B num ritmo intermediário e C mais lentamente, permitindo observar em detalhe como as saídas X e Y reagem a cada mudança. Nas figuras de waveform, cada nova transição de A ou B corrige automaticamente o vetor de seleção SIG, enquanto as transições de C ativam ou inibem os termos de produto correspondentes nas entradas D dos multiplexadores.

Conforme previsto pelas expressões $X=A^{-}B\,C+A\,B^{-}C^{-}+A\,B$ e $Y=A^{-}B^{-}+A^{-}B\,C^{-}+A\,B\,C$, observa-se que, para cada combinação de A e B, a saída X escolhe entre ‘0’, C, ¬C ou ‘1’, e Y escolhe entre ‘1’, ¬C, ‘0’ ou C, exatamente nos instantes de transição de SIG. Em particular, quando SIG muda de “01” para “10” (por exemplo, em t = 150 ms), X passa a seguir a forma de onda de C_NOT em vez de C, e Y inverte seu valor somente nos instantes em que o termo correspondente está ativo. Esses comportamentos conferem plena equivalência entre a rede estrutural de dois MUX4X1 mais inversor e as funções lógicas desejadas, confirmando a implementação correta.

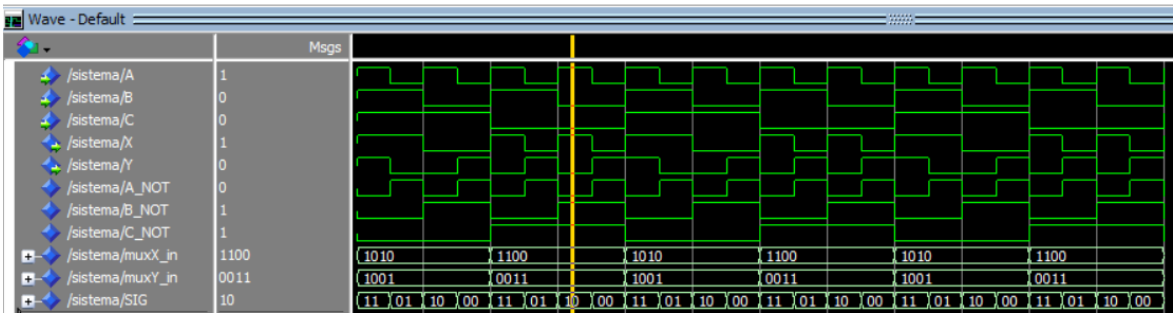

Figura 1: Simulação do SISTEMA, caso A,B,C =1,0,0. É possível notar o funcionamento dos sinais e como a saída do sistema corresponde ao esperado na tabela verdade da Tabela 1, na qual para esses valores de A.B e C : X=1 e Y=0 como na simulação.

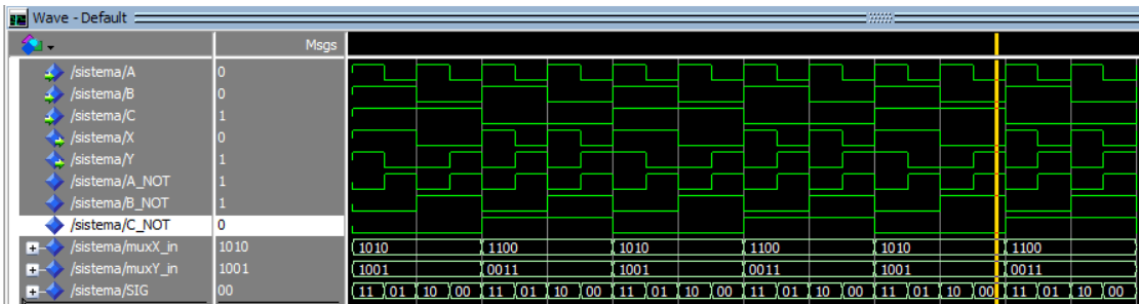

Figura 2: Simulação do SISTEMA, caso A,B,C =0,0,1. Novamente, é possível notar o funcionamento dos sinais e como a saída do sistema corresponde ao esperado na tabela verdade da Tabela 1, na qual para esses valores de A.B e C; X=0 e Y=1 como na simulação.

2. Atividade 2

2.1 Descrição da atividade

A segunda atividade propôs a implementação, em VHDL e simulação no ModelSim, de uma função booleana de sete variáveis (A, B, C, D, E, F e G) usando apenas um decodificador 4-para-16, um multiplexador 8-para-1 e portas OR. A expressão completa a ser sintetizada é **Z=F G + A B C D E⁻ F⁻ G + A⁻ B⁻ C⁻ D⁻ E⁻ F⁻ G + A B C⁻ E F G⁻ + A⁻ B C D E⁻ F G⁻ + A B C D E F⁻ G⁻ + A B C⁻ D⁻ E F⁻ G⁻.**

A arquitetura estrutural instanciou um **DECODER4X16** para gerar, a partir dos sinais A–D, os 16 mintermos "dec_out(0)" a "dec_out(15)", e um **MUX8X1** cujo vetor de seleção (E, F, G) escolhe entre sete entradas D(0…7) — cada uma formada pela combinação OR de mintermos apropriados (ou pelo produto F·G, ou pela constante '0') —, roteando o termo correto para Z. As portas OR externas agruparam os dec_out necessários para cada Dn antes da multiplexação. A lógica interna dos componentes **DECODER4X16** e **MUX8X1** encontra-se detalhada em arquivos separados no projeto (EXP_3_2.vhd e EXP_3_1.vhd, referentes às atividades 2 e 1 do exercício 3, respectivamente), compilados em conjunto com o sistema. Por fim, a simulação no ModelSim confirmou que, em todas as 128 combinações possíveis de A…G, a saída Z obedece exatamente à tabela-verdade especificada no roteiro.

A	B	C	D	E	F	G	Z
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0
0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	0
0	0	0	0	1	1	1	1

Tabela 2: Tabela verdade da expressão Z possui 128 linhas em função de Z ter 7 valores de entradas, na Tabela 2 foram representados apenas as 8 primeiras possibilidades 0000000 até 0000111, a fim de garantir um conjunto de casos de teste base.

2.2 Implementação em VHDL

A Listagem 4 apresenta o código criado para esta atividade.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SISTEMA2 IS
  PORT (
    A, B, C, D, E, F, G: IN STD_LOGIC;
    Z: OUT STD_LOGIC
  );
END SISTEMA2;
ARCHITECTURE SISTEMA2_ARCH OF SISTEMA2
ISCOMPONENT DECODER4X16 IS
    PORT (
      A: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      Y: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
    );
  END COMPONENT;
COMPONENT MUX8X1 IS
    PORT (
      S: IN STD_LOGIC_VECTOR(2 DOWNTO 0);
      D: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      Y: OUT STD_LOGIC
    );
  END COMPONENT;
SIGNAL DECO_IN    : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL DECO_OUT   : STD_LOGIC_VECTOR(15 DOWNTO
0);
SIGNAL MUX_IN     : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL SEL        : STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL Z_MUX      : STD_LOGIC;
SIGNAL FG         : STD_LOGIC;
```

```
BEGIN

DECO_IN <= A & B & C & D;
SEL <= E & F & G;
DEC: DECODER4X16 PORT MAP (
      A => DECO_IN,
      Y => DECO_OUT
   );

MUX_IN(0) <= DECO_OUT(0);
MUX_IN(1) <= '0';
MUX_IN(2) <= '0';
MUX_IN(3) <= '0';
MUX_IN(4) <= DECO_OUT(11);
MUX_IN(5) <= DECO_OUT(13);
MUX_IN(6) <= DECO_OUT(7);
MUX_IN(7) <= '0';

MUX : MUX8X1 PORT MAP (
      S => SEL,
      D => MUX_IN,
      Y => Z_MUX
   );

FG <= F AND G;
Z <= FG OR Z_MUX;

END SISTEMA2_ARCH;
```

Listagem 4: Implementação do circuito para solução da saída Z.

A entidade **SISTEMA2** define sete sinais de entrada (A, B, C, D, E, F e G) e uma saída (Z), adotando uma arquitetura estrutural que combina um decodificador 4×16, um multiplexador 8×1 e uma porta lógica OR. Primeiro, os sinais A–D são

concatenados em **DECO_IN** e fornecidos ao componente **DECODER4X16**, que gera um vetor **DECO_OUT(0...15)**, onde cada bit corresponde a um mintermo distinto de A–D. Simultaneamente, os sinais E, F e G formam o vetor **SEL**, usado como linha de seleção para o **MUX8X1**.

Em **MUX_IN(0…7)** são armazenados quatro mintermos de DECO_OUT (0, 11, 13 e 7) intercalados com constantes '0', de modo que, para cada combinação de EFG, o multiplexador roteie para sua saída **Z_MUX** exatamente o conjunto de mintermos necessário para a expressão de Z. O termo independente de A–D, dado por F·G, é calculado em **FG** via porta AND e, por fim, a saída Z é obtida pela OR entre **FG** e **Z_MUX**. Não há operações aritméticas ou lógicas adicionais na arquitetura principal: toda a função booleana de sete variáveis é sintetizada pelo arranjo estrutural de componentes pré-definidos.

É importante lembrar que os componentes MUX8X1 e DECODER16X4, não tem sua lógica no código principal do sistema, eles estão modularizados presentes em outros arquivos presentes no Model sim, são eles:

```
LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MUX8X1 IS

PORT (S: IN STD_LOGIC_VECTOR(2 DOWNTO 0);

D:IN STD_LOGIC_VECTOR(7 DOWNTO 0);

Y: OUT STD_LOGIC);

END MUX8X1;

ARCHITECTURE MUX8X1_ARCH OF MUX8X1 IS

BEGIN

Y <= D(0) WHEN (S(0) = '0' AND S(1) = '0' AND S(2) = '0')
ELSE

D(1) WHEN (S(0) = '1' AND S(1) = '0' AND S(2) = '0') ELSE

D(2) WHEN (S(0) = '0' AND S(1) = '1' AND S(2) = '0') ELSE

D(3) WHEN (S(0) = '1' AND S(1) = '1' AND S(2) = '0') ELSE

D(4) WHEN (S(0) = '0' AND S(1) = '0' AND S(2) = '1') ELSE

D(5) WHEN (S(0) = '1' AND S(1) = '0' AND S(2) = '1') ELSE

D(6) WHEN (S(0) = '0' AND S(1) = '1' AND S(2) = '1') ELSE

D(7) WHEN (S(0) = '1' AND S(1) = '1' AND S(2) = '1');

END MUX8X1_ARCH;
```

Listagem 5:Código de componente:MUX8X1

```
ENTITY DECODER4X16 IS

PORT (A: IN STD_LOGIC_VECTOR(3 DOWNTO 0);

Y:OUT STD_LOGIC_VECTOR(15 DOWNTO 0));

END DECODER4X16;

ARCHITECTURE DECODER4X16_ARCH OF
DECODER4X16 IS

BEGIN

WITH A SELECT

Y <= "0000000000000001" WHEN "0000",

"0000000000000010" WHEN
"0001","0000000000000100" WHEN
"0010","0000000000001000" WHEN
"0011","0000000000010000" WHEN
"0100","0000000000100000" WHEN
"0101","0000000001000000" WHEN
"0110","0000000010000000" WHEN
"0111","0000000100000000" WHEN
"1000","0000001000000000" WHEN
"1001","0000010000000000" WHEN
"1010","0000100000000000" WHEN
"1011","0001000000000000" WHEN
"1100","0010000000000000" WHEN
"1101","0100000000000000" WHEN
"1110","1000000000000000" WHEN
"1111","0000000000000000" WHEN OTHERS;

END DECODER4X16_ARCH;
```

Listagem 6: Código de Componente:DECODER16X4

2.3 Simulação

Para verificar a corretude do SISTEMA2, a simulação foi conduzida no ModelSim aplicando-se ondas quadradas aos sinais de entrada com períodos que dobram a cada bit: A alterna a cada 50 ms, B a cada 100 ms, C a cada 200 ms, D a cada 400 ms, E a cada 800 ms, F a cada 1600 ms e G a cada 3200 ms. Dessa forma, em um único ciclo de 3200 ms, são exercitadas todas as 128 combinações possíveis de A…G. No waveform, cada transição de D altera simultaneamente o vetor DECO_IN, gerando o respectivo bit ativo em DECO_OUT; em seguida, mudanças em E, F e G modificam o vetor SEL do MUX8X1, direcionando para Z_MUX o agrupamento correto de mintermos ou ‘0’, conforme a atribuição em MUX_IN.

Além disso, sempre que F e G assumem ‘1’, o sinal FG ativa diretamente Z via a porta OR final, independentemente de Z_MUX, conforme a expressão Z = FG OR Z_MUX. Observa-se que, nos instantes em que SEL muda de “010” para “011” (por exemplo, em t = 1100 ms), Z passa a refletir FG se este for ‘1’; em outros intervalos, Z segue fielmente o valor roteado pelo MUX8X1, confirmando que a combinação do decodificador, multiplexador e portas lógicas implementa corretamente cada mintermo da função.

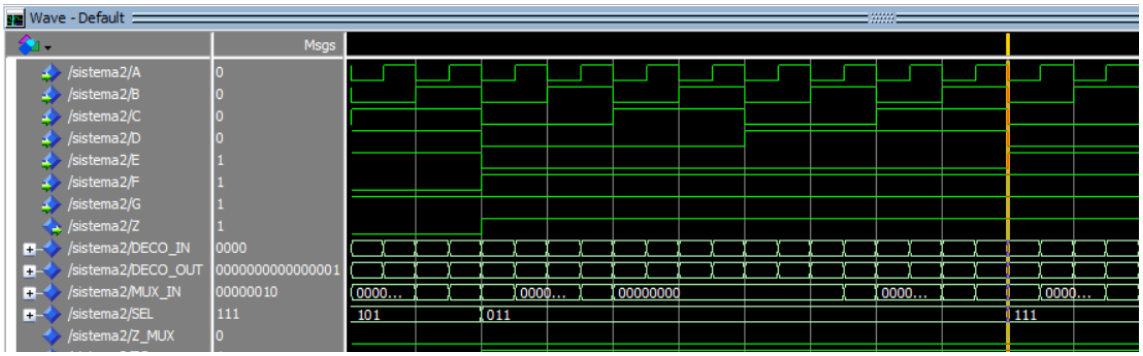

Figura 3: Simulação do SISTEMA2, caso E,F,G =1,1,1. É possível notar o funcionamento dos sinais e como a saída do sistema corresponde ao esperado na tabela verdade da Tabela 2, na qual para esses valores de A.B e C : Z=1 como na simulação.

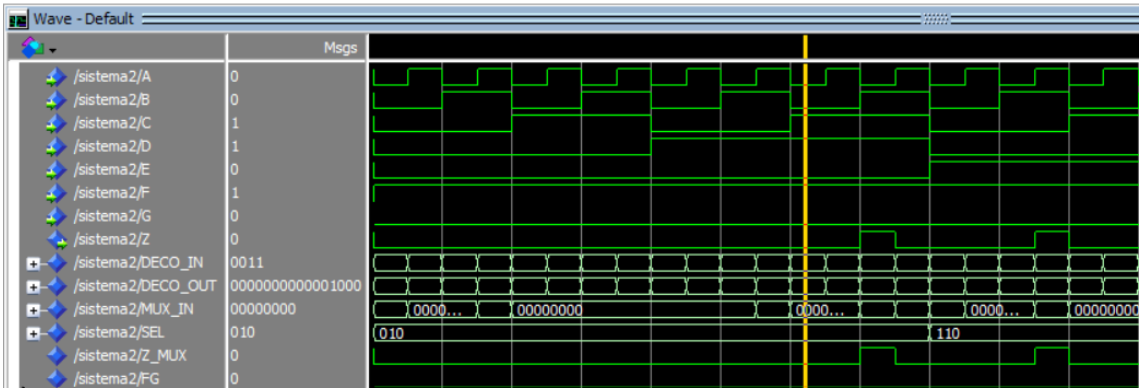

Figura 4: Simulação do SISTEMA2, caso C,D,F =1,1,1. É possível notar o funcionamento dos sinais e como a saída do sistema corresponde ao esperado na tabela verdade da Tabela 2, na qual para esses valores de A.B e C : Z=0 como na simulação.