Universidade Federal do Rio Grande do Norte
Centro de Tecnologia - CT
Departamento de Engenharia de Computação e Automação

Multiplexadores e Demultiplexadores
DCA0212.1 - Grupo 09 - Laboratório 03

Igor Michael Araujo de Macedo
Isaac de Lyra Junior
Pedro Henrique de Freitas Silva

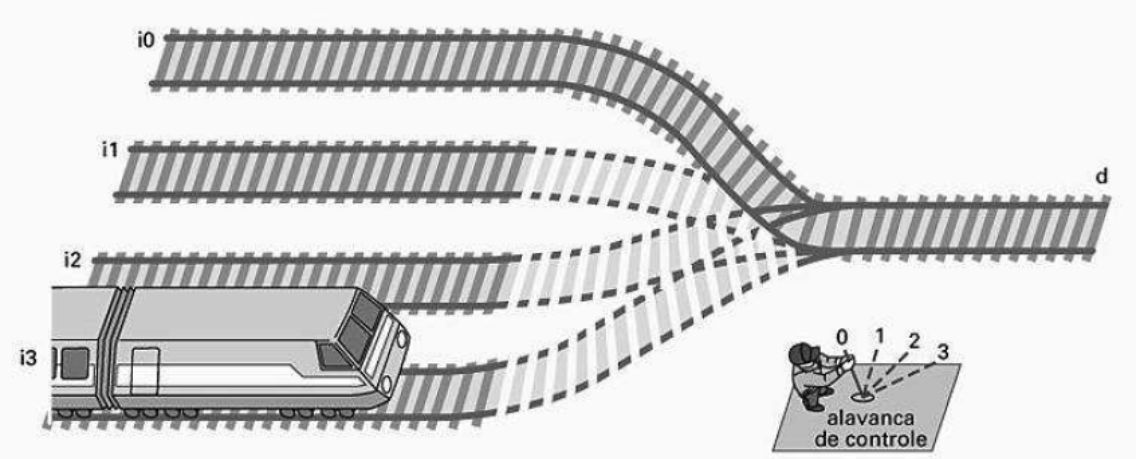
Natal, 18 de dezembro de 2021

1 INTRODUÇÃO

Um multiplexador ("mux", em forma abreviada) é um bloco construtivo usado em circuitos digitais. Um multiplexador Mx1 tem M entradas de dados e 1 saída, permitindo que apenas uma das entradas seja passa para a saída. Por vezes os multiplexadores são chamados também de seletores por que selecionam uma das entradas para ser passada à saída (VAHID, 2008).

Um aparelho de mudança de via em um parque ferroviário de manobras pode ser um facilmente comparado a um multiplexador, pois ele põe em conexão diversas vias de entrada com uma única via de saída, como mostra a Figura 1. A alavanca de controle do aparelho de mudança de via estabelece a conexão entre a via de entrada e a via de saída. O surgimento de um trem na saída dependerá de se há um trem presente na via de entrada selecionada no momento. Em um multiplexador, o controle não é feito através de uma alavanca, mas por entradas de seleção, as quais representam a conexão desejada em binário (VAHID, 2008).

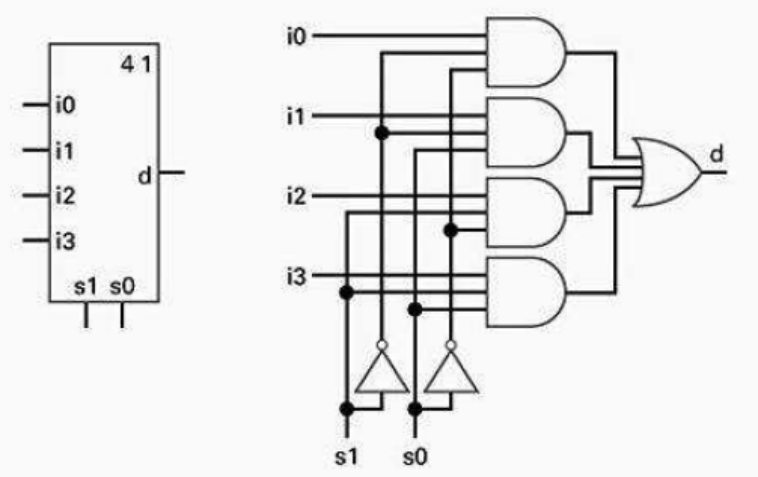
Figura 1 – Aparelho de mudança de via de um parque ferroviário de manobras



Fonte: Vahid (2008).

Um multiplexador de quatro entradas, conhecido como multiplexador 4x1, tem quatro entradas de dados i3, i2, i1 e i0, duas entradas de seleção s1 e s0, e uma saída de dados d. Um diagrama de blocos de um multiplexador pode ser visto na Figura 2.

Figura 2 – Multiplexador 4x1, na esquerda o símbolo em forma de bloco e na direita a estrutura interna



Fonte: Vahid (2008).

Diante disto, o presente relatório tem por fim apresentar a atividade de laboratório da disciplina de Circuitos Digitais do curso de Engenharia da Computação da Universidade Federal do Rio Grande do Norte sobre a implementação de multiplexadores utilizando a linguagem VHDL (*VHSIC Hardware Description Language*).

2 METODOLOGIA

Para a realização do laboratório foi utilizado como referência o arquivo em PDF da atividade disponibilizado pelos docentes, além do livro-texto do Vahid (2008). Todo o código foi implementado e simulado utilizando o software Quartus Prime Lite.

2.1 MUX 2x1

Foi pedido na atividade de laboratório para que fosse implementado um MUX 2x1 de duas maneiras diferentes, uma utilizando circuitos lógicos e outra utilizando a descrição comportamental.

2.1.1 Implementação utilizando circuitos lógicos

Para implementar o MUX 2x1 utilizando circuitos lógicos, foi utilizado a tabela verdade de um multiplexador 2x1 para construir a equação que define um multiplexador 2x1. Com a equação em mãos foi construído uma entidade no VHDL em que denominamos *MUX2x1logico*, que tem 3 entradas, sendo I0 e I1 as entradas referentes aos dados e S a entrada seletora, e apenas uma saída. O código desta entidade pode ser visto na Figura 3.

Figura 3 – Implementação da entidade do multiplexador 2x1 com portas lógicas

```
-- MUX 2x1 LOGICO
entity MUX2x1logico is
  port(I0, I1, S: in bit;
       O: out bit);
end MUX2x1logico;

architecture hardware of MUX2x1logico is

begin
  -- saida
  O <= (I0 and (not S)) or (I1 and S);

end hardware;
```

Fonte: Autores.

2.1.2 Implementação utilizando a descrição comportamental

Para implementar utilizando a descrição comportamental foi pensado em como um multiplexador 2x1 deve definir a sua saída a partir com as entradas. A partir disto, foi criado uma entidade denominada de *MUX2x1comp*, também com 3 entradas, sendo 2 de dados e 1 seletora, e uma saída. O código na íntegra desta entidade está na Figura 4.

Figura 4 – Implementação da entidade do multiplexador 2x1 utilizando descrição comportamental

```
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=---
  -- M U X    2 x 1    C O M P O R T A M E N T A L
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=---
entity MUX2x1comp is
  port(I0, I1, S: in bit;
          O: out bit);
end MUX2x1comp;

architecture hardware of MUX2x1comp is

begin
  -- saida
  with S select
    O <= I0 when '0', I1 when '1';
end hardware;
```

Fonte: Autores.

2.2 MUX 4x1

Para o caso das implementações do MUX 4x1, foi pedido para que fosse implementado primeiramente utilizando a descrição comportamental e mais uma vez utilizando o componente do MUX 2x1.

2.2.1 Implementação utilizando a descrição comportamental

Para implementar o MUX 4x1 utilizando a descrição comportamental foi utilizado a mesma metodologia utilizada ao implementar o MUX 2x1 comportamental. Logo, foi criado uma entidade denominada de *MUX4x1comp*, que tem como portas 5 entradas, sendo 4 de dados e 1 seletora de barramento de 2 bits, e apenas 1 saída. O código em VHDL da entidade implementada pode ser visto na Figura 5.

Figura 5 – Implementação da entidade do multiplexador 4x1 utilizando descrição comportamental

```
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=---
  -- M U X    4 x 1    C O M P O R T A M E N T A L
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=---
entity MUX4x1comp is
    port(I0, I1, I2, I3: in bit;
            S: in bit_vector(1 downto 0);
            O: out bit);
  end MUX4x1comp;

  architecture hardware of MUX4x1comp is

  begin
    -- saida
    with S select
      O <= I0 when "00",
        I1 when "01",
        I2 when "10",
        I3 when OTHERS;
  end hardware;
```

Fonte: Autores.

2.2.2 Implementação utilizando MUX 2x1

Para a implementação do MUX 4x1 utilizando como componente o MUX 2x1, foi criado uma entidade denominada de *MUX4x1comp2*, com apenas 2 entradas, sendo uma de barramento de 4 bits, referente aos dados e outra de barramento de 2 bits, referente aos bits de seleção, e apenas uma saída de 1 bit. A lógica desenvolvida para esta entidade pode ser vista na Figura 8

Figura 6 – Implementação do multiplexador 4x1 utilizando a componente do multiplexador 2x1

```
-----------------------------------------------------
  -- M U X   4 x 1   C O M P O R T A M E N T A L 2
-----------------------------------------------------
entity MUX4x1comp2 is
  port(
    I: in bit_vector(3 downto 0);
    S: in bit_vector(1 downto 0);
    O: out bit
  );
end MUX4x1comp2;

architecture hardware of MUX4x1comp2 is

  component MUX2x1comp is
    port(I0, I1, S: in bit;
         O: out bit);
  end component;

  signal O1, O2: bit;

begin
  -- saida

  M1: MUX2x1comp port map(I(0), I(1), S(0), O1);
  M2: MUX2x1comp port map(I(2), I(3), S(0), O2);
  M3: MUX2x1comp port map(O1, O2, S(1), O);

end hardware;
```

Fonte: Autores.

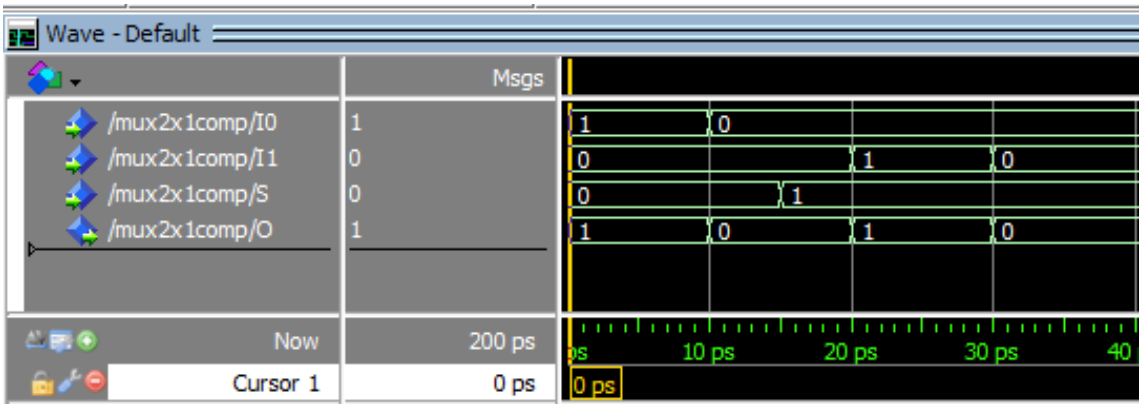
3 RESULTADOS

Para simulação do código desenvolvido no laboratório foi utilizado o software ModelSim.

3.1 Simulações

3.1.1 MUX 2x1

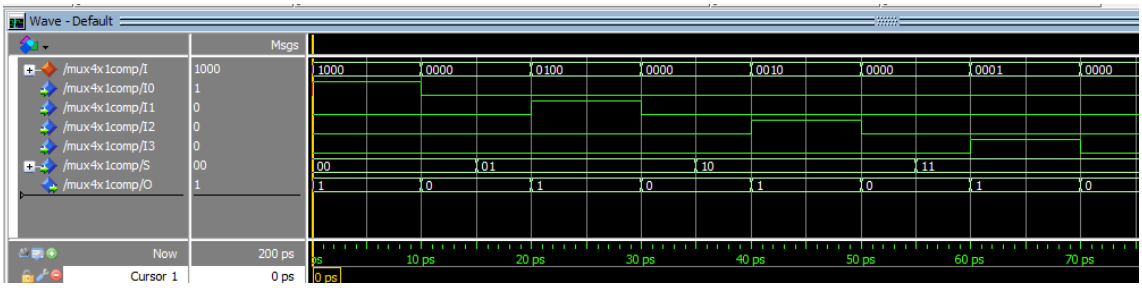
Figura 7 – Simulação do multiplexador 2x1 implementado



Fonte: Autores.

3.1.2 MUX 4x1

Figura 8 – Simulação do multiplexador 4x1 implementado



Fonte: Autores.

4 CONCLUSÃO

O trabalho teve como objetivo pôr em prática os conhecimentos estudados sobre multiplexadores e testar a implementação e simulação destes por meio do VHDL. Para isso, foi implementado um multiplexadores 2x1 e 4x1 utilizando circuitos lógicos e também descrição comportamental, onde não é utilizado circuitos lógicos durante a implementação. Para a solução foi construído quatro entidades utilizando a linguagem VHDL, duas referentes ao multiplexador 2x1, sendo uma delas utilizando circuitos lógicos e outra utilizando descrição comportamental, e duas referentes ao multiplexador 4x1, sendo uma delas utilizando descrição comportamental e outra utilizando a entidade do multiplexador 2x1 como componente. Todas as entidades foram testadas e simuladas utilizando o software ModelSim. Após todo o processo, a simulação retornou os resultados esperados e, com isso, podemos concluir que foi possível extrair o máximo de conhecimento e proveito do problema.

Referências

VAHID, F. *Sistemas Digitais: Projeto, Otimização e HDLs*. [S.l.]: Artmed Bookman, 2008.

ANEXO A – Código VHDL

```
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--
          -- M U X    2 x 1    L O G I C O
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--
entity MUX2x1logico is
  port(I0, I1, S: in bit;
          O: out bit);
end MUX2x1logico;

architecture hardware of MUX2x1logico is

begin
  -- saida
  O <= (I0 and (not S)) or (I1 and S);

end hardware;

--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--
  -- M U X    2 x 1    C O M P O R T A M E N T A L
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--
entity MUX2x1comp is
  port(I0, I1, S: in bit;
          O: out bit);
end MUX2x1comp;

architecture hardware of MUX2x1comp is

begin
  -- saida
  with S select
    O <= I0 when '0', I1 when '1';
end hardware;

--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--
  -- M U X    4 x 1    C O M P O R T A M E N T A L
--=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--
entity MUX4x1comp is
    port(I0, I1, I2, I3: in bit;
            S: in bit_vector(1 downto 0);
            O: out bit);
  end MUX4x1comp;

  architecture hardware of MUX4x1comp is
```

```
begin
  -- saida
  with S select
    O <= I0 when "00",
      I1 when "01",
      I2 when "10",
      I3 when OTHERS;
end hardware;

-------------------------------------------------------------
  -- M U X    4 x 1    C O M P O R T A M E N T A L 2
-------------------------------------------------------------
entity MUX4x1comp2 is
  port(
    I: in bit_vector(3 downto 0);
    S: in bit_vector(1 downto 0);
    O: out bit
  );
end MUX4x1comp2;

architecture hardware of MUX4x1comp2 is

  component MUX2x1comp is
    port(I0, I1, S: in bit;
         O: out bit);
  end component;

  signal O1, O2: bit;

begin
  -- saida

  M1: MUX2x1comp port map(I(0), I(1), S(0), O1);
  M2: MUX2x1comp port map(I(2), I(3), S(0), O2);
  M3: MUX2x1comp port map(O1, O2, S(1), O);

end hardware;
```