1. Laboratório 5 – Circuitos Decodificadores, Mux/Demux e Drivers BCD 7 Segmentos

Objetivos:

* + Aprender como interpretar os dados da especificação técnica do fabricante;
  + Conhecer os equipamentos utilizados para efetuar as práticas;
  + Melhorar a interpretação e o desenvolvimento de projetos;
  + Comprovar o funcionamento de decodificadores e suas funções.
  + Comprovar o funcionamento de multiplexadores e demultiplexadores;
  + Comprovar a utilização de decodificadores e *drivers* de BCD para

*displays* de 7 segmentos;

* + Montagem limpa e exemplar (fiação curta e arrumada).
  + Inicio das montagens e simulações no Quartus Prime.
  + Uso da placa de simulação física de FPGA – ALTERA.

## Tempo de Execução: 4 Aulas

**Início:**  **/ / - Término:**  **/ /**

Desenvolvimento

Equipamentos e materiais necessários para realizar as experiências estão descritos nas atividades.

# HDL - Linguagem de Descrição de Hardware

Os militares americanos sentiram necessidade de desenvolver uma forma de descrever o hardware de uma forma textual mais simples do que as expressões lógicas booleanas. Eles perceberam que a intepretação de expressões booleanas e seus símbolos era de muito difícil mesmo para sistemas computadorizados de última geração. Dessa necessidade surge a linguagem de descrição de hardware para dispositivos de alta velocidade VHSIC, que depois foi batizada de VHDL.

A ALTERA, um dos primeiros fabricantes de dispositivos de lógica programável, desenvolveu linguagem de descrição de hardware voltada a facilitar o uso de seus produtos, sendo adquirida recentemente pela INTEL (Figura 39).

|  |
| --- |
| Figura 39. Exemplo de modelo FPGA ALTERA. |
|  |
| Fonte: autor |

# AHDL – ALTERA/INTEL

SUBDESIGN - Assinatura

Nesta seção são declaradas, as entradas , saídas, finos com operação bidirecional, entradas e saídas de máquinas de estado e vetores. Podemos ainda definir, de maneira opcional, o comportamento padrão podendo assumir valores GND e VCC (GND nível lógico baixo ou 0, VCC nível lógico alto ou 1). Observe esse exemplo da sintaxe para esta seção na Figura 40.

|  |
| --- |
| Figura 40. Exemplo de sintaxe da seção SUBDESIGN. |
|  |
| Fonte: autor |

VARIABLE

Esta seção é opcional, pode ou não existir em seu projeto. Normalmente é usada para declarar ou ainda gerar variáveis usadas na seção de lógica, como por exemplo, portas *tristate*, nós intermediários, *flip-flop*, registradores, contadores, máquinas de estado, entre outros, sendo muito semelhantes a variáveis usadas em linguagens de alto nível, do ponto de vista de atribuição e utilização. A Figura 41 apresenta a seção VARIABLE com a declaração dos nós intermediários utilizados no código.

|  |
| --- |
| Figura 41. Exemplo de sintaxe da seção VARIABLE. |
|  |
| Fonte: autor |

NODE – Nós intermediários

Os nós intermediários se assemelham aos nós elétricos que temos nos circuitos eletrônicos que interligam mais que dois pontos. A diferença é que um intermediário é utilizado para fornecer uma compatibilidade elétrica dentro das células, auxiliando no sentido de funcionamento dos sinais lógicos. Analisando o circuito do exemplo na Figura 42, é possível verificar que o Nó 1 é responsável em receber o sinal da variável de nível SL e encaminhar/replicar para duas entradas de portas lógicas diferentes. Já o Nó 2 por sua vez, está recebendo a resposta da operação lógica OR e direcionando para a entrada de uma porta AND e também para a saída, que fará o acionamento está bomba.

|  |
| --- |
| Figura 42. Nós intermediários e os sentidos das informações. |
|  |
| Fonte: autor |

BEGIN e END

Nesta seção teremos a lógica do projeto que está sendo implementado (Figura 43). Ela começa com o BEGIN e termina com a palavra reservada END seguida de ponto e vírgula (;).

|  |
| --- |
| Figura 43. Exemplo de sintaxe da seção BEGIN. |
|  |
| Fonte: autor |

Funções lógicas

A tabela 8 apresenta as funções lógicas e como podem ser utilizadas para a criação de expressões lógicas, manipulado as entradas e variáveis, onde podemos utilizar tanto o símbolo quanto a função propriamente dita, como por exemplo, a função lógica AND que pode ser escrita: SH AND SL, ou ainda, SH & SL.

Tabela 1. Operações lógicas em AHDL.



**Função Descrição**

|  |  |
| --- | --- |
| **&**  **AND** | Operação Lógica AND |
| **!&**  **NAND** | Operação Lógica NAND |
| **$**  **XOR** | Operação Lógica XOR |
| **#**  **OR** | Operação Lógica OR |
| **!#**  **NOR** | Operação Lógica NOR |
| **!**  **NOT** | Operação Lógica NOT, Inversão |

TRUTH TABLE – Tabela verdade

O uso de uma tabela verdade em um projeto de lógica programável pode facilitar bastante a implementação, tendo em vista que normalmente montamos a tabela verdade no início do projeto, levantando todas as condições necessárias para o funcionamento do projeto de lógica combinacional. Neste projeto em específico (Etapa 1 e Figura 1), quando gerarmos o bloco ou formos montar a aplicação deveremos conectar a saída Bo ao sinal de entrada RBo (retorno de Bo), pois só assim teremos a execução do projeto como o descrito por sua tabela. O sinal RBo junto com sua lógica fornecem uma realimentação do sinal da bomba para informar que a bomba foi acionada, e mantê-la acionada até que o sinal do sensor de nível L (inferior) informe que o nível está baixo. A Figura 44 apresenta como pode ser implementada uma tabela verdade em AHDL.

|  |
| --- |
| Figura 44. Exemplo de tabela verdade como lógica. |

|  |
| --- |
|  |
| Fonte: autor |

ARRAY – Vetores

Quando necessitamos de vários sinais juntos, compondo uma mesma informação (vários bits paralelos), podemos declarar um vetor ou ARRAY, que permita condensar e facilitar a manipulação destes bits. Este recurso permite ainda que dois vetores possam ser concatenados facilmente. A Figura 45 apresenta o código do nosso programa de controle do sistema descrito da Figura 42 sendo executado por uma tabela verdade, mas com um vetor de entradas de 3 posições. Os vetores também podem ser utilizados para agrupar saídas, facilitando muito a manipulação e número de conexões.

|  |
| --- |
| Figura 45. Exemplo do uso de vetores ou ARRAY em projetos. |

|  |
| --- |
|  |
| Fonte: autor |

# Modularização em AHDL

Quando desenvolvemos um projeto, seja ele de forma gráfica, onde desenhamos o circuito com as portas lógicas ou quando geramos o código AHDL, podemos criar um Bloco (modularizar, criando um módulo), que poderá ser replicado quantas vezes forem necessárias dentro do projeto como se fosse um componente gráfico. A Figura 46 apresenta o código AHDL utilizado para geração do Bloco.

|  |
| --- |
| Figura 46. Arquivo para geração do módulo . |
|  |
| Fonte: autor |

Na Figura 47 temos o circuito da nossa aplicação representada na Figura 42, do nosso sistema de bombeamento. Observe que foi criado um bloco com as informações apresentadas no código AHDL. As entradas SH e SL aparecem à esquerda e a saída Bo a direita, Foram inseridos elementos gráficos representantes de Entradas (INPUT) para cada entrada do bloco, assim como um elemento de saída (OUTPUT) para o sinal de saída.

|  |
| --- |
| Figura 47. Módulo – Bloco - gerado a partir do arquivo. |
|  |
| Fonte: autor |

A modularização ajuda significativamente quando temos um sistema muito complexo e nós o fragmentamos em pequenas partes, as quais são desenvolvidas em AHDL, geramos seus blocos e por fim integramos de forma gráfica todos os blocos criados.

|  |
| --- |
| **PRÉ-RELATÓRIO DE LABORATÓRIO 5** |
| **Estudante:** |
| **Turno: Matutino ( ) Noturno ( ) Turma:** |

## (Individual)

1. Realizar uma pesquisa sobre: Decodificadores, Multiplexadores, Demultiplexadores e Display de 7 segmentos Anodo Comum e Catodo Comum de como eles funcionam. A pesquisa deverá conter no máximo 4 linhas para cada item, e deverá ser entregue no relatório.

R: Um decodificador é um circuito lógico capaz de converter um conjunto de entradas que representa um número binário de N bits em M linhas de saída. Porém, cada linha de saída será ativada somente uma por vez.

Um Multiplexador ou MUX é um circuito combinacional dedicado, ou seja, composto de portas lógicas (principalmente portas AND), possuindo duas ou mais entradas e somente uma única saída. Sua finalidade é selecionar uma de suas entradas e conectá-la eletronicamente a sua única saída.

Um Demultiplexador ou DEMUX é um circuitocombinacional dedicado possuindo uma entrada e duas oumais saídas. Sua finalidade é selecionar, através de variáveisde seleção, qual de suas saídas deve receber a informaçãopresente em sua única entrada, executando a operação inversarealizada pelo MUX.

Um display de sete segmentos, como seu nome indica, é composto de sete elementos (a, b, c, d, e, f, g) os quais podem ser ligados ou desligados individualmente. Eles podem ser combinados para produzir representações simplificadas mostrar números de 0 a 9 e opcionalmente as letras A,b,C,d,E e F. (outros símbolos são possíveis desde que se utilize um circuito decodificador adequado. A maioria dos displays também possui em 8 elemento correspondente ao ponto decimal (dp).

Catodo comum:

Nestes tipos de mostradores, o catodo comum é normalmente ligado a um

potencial de 0V, e os anodos dos leds são ligados a um resistor limitador de

corrente e a uma conexão que irá controlar o acendimento do led. Para acender é necessário colocar um potencial de VCC (nível lógico ALTO "1").

Anodo comum:

Nestes tipos de mostradores, o anodo comum é normalmente ligado a um potencial de VCC, e os catodos dos leds são ligados a um resistor limitador de corrente e a uma conexão que irá controlar o acendimento do led. Para acender é necessário colocar um potencial de GND (nível lógico BAIXO "0").

1. Projete um decodificador de 2 para 4 utilizando as ferramentas Tinkercad (quando solicitado) e QUARTUS, seguindo as seguintes solicitações:
   1. Utilizando apenas portas lógicas básicas (AND, OR e NOT) nas duas ferramentas Tinkercad e Quartus.
   2. Utilizando Linguagem AHDL e Modularização.



Diagrama em bloco do decodificador 2x4

1. Projeto: copie ou anexe o projeto completo realizado no Quartus.

|  |
| --- |
| Interface gráfica do usuário, Diagrama  Descrição gerada automaticamente com confiança médiaInterface gráfica do usuário  Descrição gerada automaticamente com confiança média |

|  |
| --- |
|  |

Projeto: copie ou anexe o projeto completo realizado no TINKERCAD.

|  |
| --- |
| Diagrama  Descrição gerada automaticamente |
| Diagrama  Descrição gerada automaticamenteDiagrama, Esquemático  Descrição gerada automaticamenteDiagrama  Descrição gerada automaticamente |

1. Código AHDL do Decodificador 2x4 – Copie ou anexe o código gerado no Quartus:

|  |  |
| --- | --- |
| **1** | SUBDESIGN decod\_2x4 ( |
| **2** | input1, input0: input; |
| **3** | output0, output1, output2, output3: output |

|  |  |
| --- | --- |
| **4** | ) |
| **5** | VARIABLE |
| **6** | No1, No2, No3, No4: NODE; |
| **7** | BEGIN |
| **8** | No1 = (NOT input1); |
| **9** | No2 = (NOT input0); |
| **10** | No3 = input1; |
| **11** | No4 = input0; |
| **12** | output0 = (No2 AND No1); |
| **13** | output1 = (No2 AND No3); |
| **14** | output2 = (No4 AND No1); |
| **15** | output3 = (No4 AND No3); |
| **16** | END; |

1. Tabela verdade do circuito decodificador 2x4.
2. Com projeto apresentado e o auxílio do datasheet, monte o circuito de teste para o CD4511 no TINKERCAD e responda:
3. Qual a função do sinal de controle LT, como realizar o seu teste e qual o resultado esperado?

R: LT (Lamp Test): Utilizado para testar todos os segmentos do display de 7 segmentos sem inserir dados BCD. Quando ativado, todos os segmentos devem acender, permitindo verificar se há algum segmento com defeito. Devido ter um inversor na entrada para obter o funcionamento normalizado do CD4511, essa entrada precisa estar em 1.

1. Qual a função do sinal de controle BI, como realizar o seu teste e qual o resultado esperado?

R: BI (Blanking Input): Controla a ativação ou desativação da saída do CD4511. Quando em nível lógico alto, a saída é determinada pelos dados de entrada. Quando em nível lógico baixo, a saída é desativada, independentemente dos dados de entrada. Devido ter um inversor na entrada, para obter o funciomaneot normalizado do CD4511, essa entrada precisa estar em 1.

1. Qual a função do sinal LE , como realizar o seu teste e qual o resultado esperado?

R: LE (Latch Enable) controla o congelamento do último dígito exibido no display de 7 segmentos. Quando o sinal LE está em nível lógico alto, o último dígito é congelado. Para o funcionamento normalizado do CD4511, essa entrada deve estar em nível lógico baixo.

1. O que acontece ao colocar o valor binário 10112 nas entradas de dados, qual o motivo da resposta?

R:O CD4511 é um decodificador BCD para display de 7 segmentos. Ao colocar o valor binário 1011 (ou 11 em decimal) nas entradas de dados desse decodificador, o display de 7 segmentos provavelmente exibirá o número correspondente, que seria "11" em decimal.Porém, vale ressaltar que o CD4511 é projetado para decodificar números BCD (Binary Coded Decimal), que vão de 0 a 9 em cada dígito. Assim, ao fornecer um valor binário de 1011 nas entradas de dados, que equivale a 11 em decimal, é possível que a saída não seja uma representação válida em um display de 7 segmentos, dependendo da implementação específica do circuito que está utilizando o CD4511. Em algumas situações, o comportamento pode ser indeterminado ou o display pode exibir um padrão de segmentos indefinido.

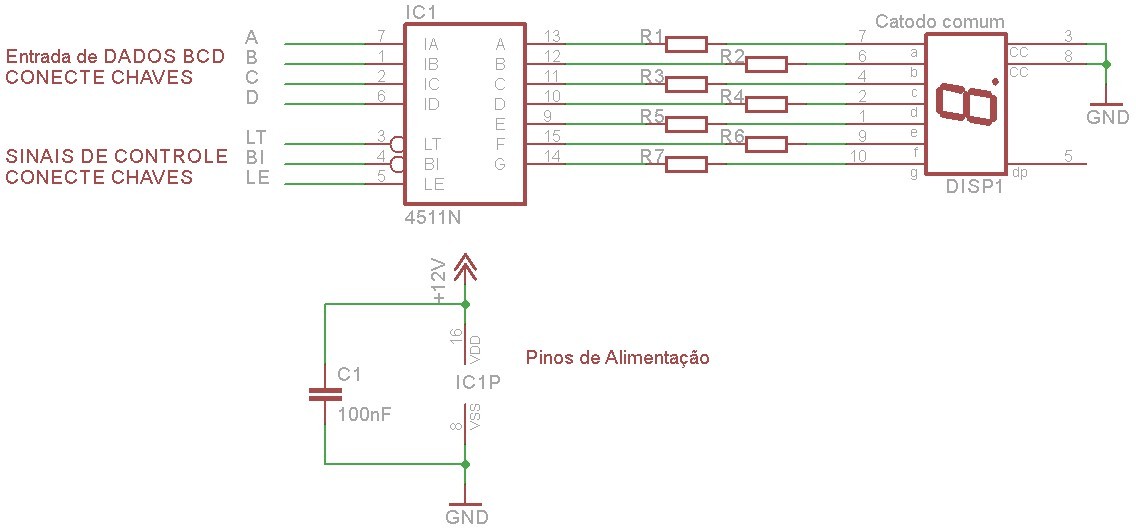


Figura 48. Circuito de teste para o CD4511.

* 1. Tabela verdade e Circuito Decodificador BCD para 7 segmentos
  2. Projete um decodificador hexadecimal para 7 segmentos para ser implementado no Quartus. Apresente aqui o código em AHDL. (utilize o material Creating Decoders AHDL como referencia)

R:

SUBDESIGN 7segmentos (

A0, B0, C0, D0: INPUT;

a, b, c, d, e, f, g : OUTPUT;

)

BEGIN

TABLE

A0, B0, C0, D0 => a, b, c, d, e, f, g;

0, 0, 0, 0 => 1, 1, 1, 1, 1, 1, 0;

0, 0, 0, 1 => 0, 1, 1, 0, 0, 0, 0;

0, 0, 1, 0 => 1, 1, 0, 1, 1, 0, 1;

0, 0, 1, 1 => 1, 1, 1, 1, 0, 0, 1;

0, 1, 0, 0 => 0, 1, 1, 0, 0, 1, 1;

0, 1, 0, 1 => 1, 0, 1, 1, 0, 1, 1;

0, 1, 1, 0 => 1, 0, 1, 1, 1, 1, 1;

0, 1, 1, 1 => 1, 1, 1, 0, 0, 0, 0;

1, 0, 0, 0 => 1, 1, 1, 1, 1, 1, 1;

1, 0, 0, 1 => 1, 1, 1, 1, 0, 1, 1;

END TABLE;

END;

1. Projete e implemente o sistema de aplicação de MUX/DEMUX, como o apresentado no diagrama a seguir, utilizando o CI 74151 e o CI 74138 , configure corretamente as entradas de habilitação e configuração, respeite os sinais mais significativos dos sinais de seleção, para que não ocorra inversão dos dados nas saídas, nem de saídas. OBS: dado entrando em CH0 deve sair em Y0 (sem inversão)

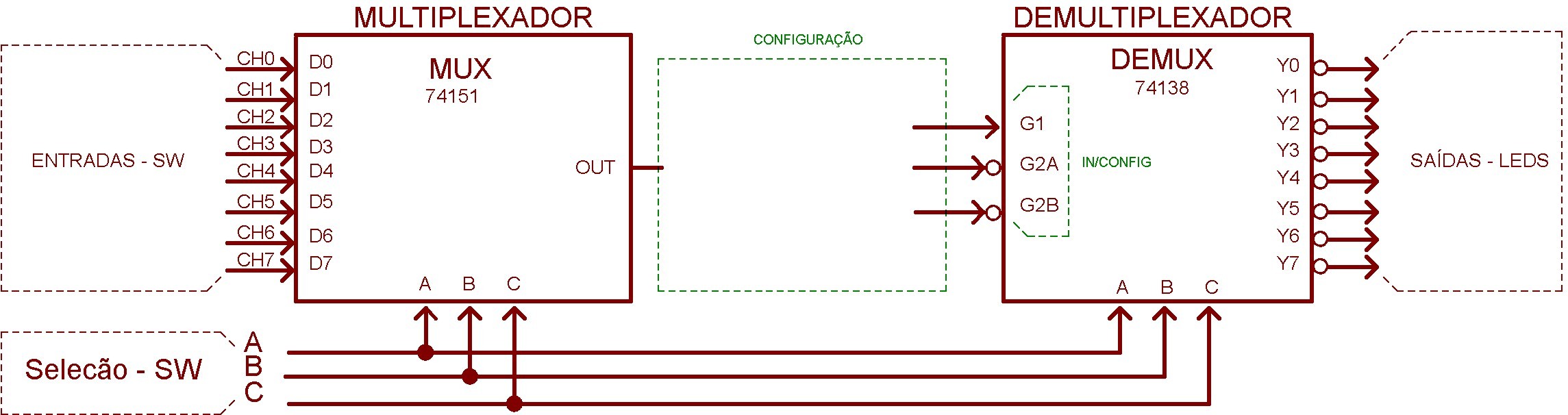


Figura 49. Diagrama em blocos do sistema multiplexador e demultiplexador

## Tabela verdade e Circuito do sistema de aplicação de MUX/DEMUX

**Esboço do Projeto do Multiplexador e Demultiplexador a ser implementado**

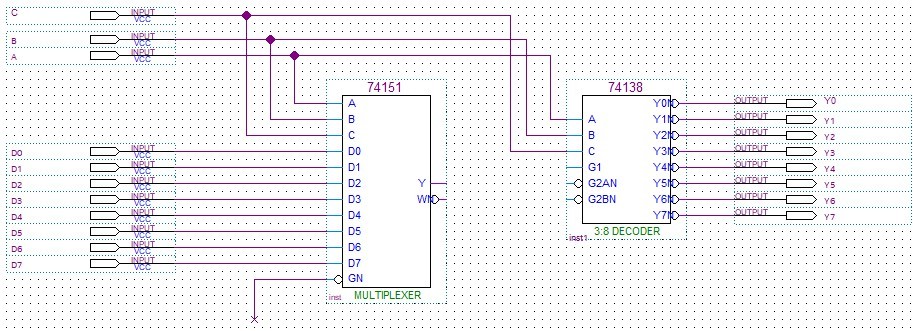


Figura 50. Circuito em blocos (incompleto) para implementação no Quartus.

|  |
| --- |
| **Folha de Vistos LAB05 – RELATÓRIO DE LABORATÓRIO**  **(imprimir para receber os vistos de apresentação)** |
| **Estudante:** |
| **Estudante:** |
| **Turno: Matutino ( ) Noturno ( ) Turma:** |

**OBSERVAÇÃO: PARA TODOS OS PROJETOS APRESENTADOS NO QUARTUS ENTREGAR JUNTO COM A FOLHA DE VISTOS:**

* **SIMULAÇÃO**
* **RELATÓRIO COM O REPORT.**

## Montagem e funcionamento dos projetos em laboratório:

* 1. Execução: projeto decodificador de 2 para 4
     1. com portas lógicas básicas: / /2024

1. em AHDL: / /2024
   1. Execução: projeto decodificador hexadecimal para 7 segmentos em AHDL:

/ /2024

* 1. Execução: projeto de Aplicação de MUX/DEMUX: / /2024

1. Compare os resultados obtidos com os descritos nas aulas teóricas. Anote as conclusões:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |