Instituto Tecnológico de Aeronáutica
Departamento de Eletrônica Aplicada

EEA-25

Lista 1

COMP 25

Daniel Araujo Cavassani

Professor:

André da Fontoura Ponchet

1. EXERCÍCIOS TEÓRICOS

1.1. Qual a definição de linguagem HDL. Forneça alguns exemplos.

As linguagens de descrição de hardware (HDL) são especialmente usadas para descrever a estrutura e o comportamento de circuitos eletrônicos, permitindo simulações, análises e síntese de circuitos, facilitando o design de sistemas digitais.

Exemplos:

- **VHDL (VHSIC Hardware Description Language):**
  Linguagem fortemente tipada e baseada em Ada, usada para descrever o comportamento e a estrutura de sistemas eletrônicos.
- **Verilog:**
  Linguagem um pouco mais flexível e menos rigorosa que o VHDL, frequentemente usada na indústria de semicondutores. Se assemelha à linguagem C em sua sintaxe.
- **SystemVerilog:**
  Extensão do Verilog, adicionando alguns recursos e funcionalidades.
- **SystemC:**
  Linguagem baseada em C++ para modelagem de sistemas eletrônicos em alto nível.

1.2. Quais as vantagens de utilização de linguagens HDL em projeto de circuitos digitais? Quais os níveis de abstração de uma linguagem HDL? Explique.

Vantagens:

- **Simulação e Verificação:**

  Possibilidade de simular o circuito antes da fabricação.

- **Reutilização de Código:**

  É possível reutilizar componentes descritos em HDL em diferentes projetos.

- **Portabilidade:**

  Descrições em HDL independem da tecnologia, isto é, podem ser implementadas em diferentes tipos de hardware.

- **Síntese Automática:**

  É possível realizar a conversão da descrição de alto nível em uma netlist, que podem ser utilizadas para fabricar o circuito.

- **Documentação Integrada:**

  A linguagem HDL pode ser utilizada como uma documentação do projeto

- **Desenvolvimento Paralelo:**

  Por ser dividido em módulos, um projeto pode ser trabalhado paralelamente.

Níveis de abstração de uma linguagem HDL:

- **Nível de Transistor:**

  Descreve o comportamento e a estrutura do circuito em termos de transistores individuais, sendo o nível mais baixo possível de abstração.

- **Nível de Porta Lógica:**

  Descreve o circuito por meio de portas lógicas, sendo mais fácil de entender do que o nível de transistor.

- **Nível de Registro:**

  Foca na transferência de dados entre registros e nas operações lógicas e aritméticas realizadas nos dados.

- **Nível de Transferência de Registro (RTL):**

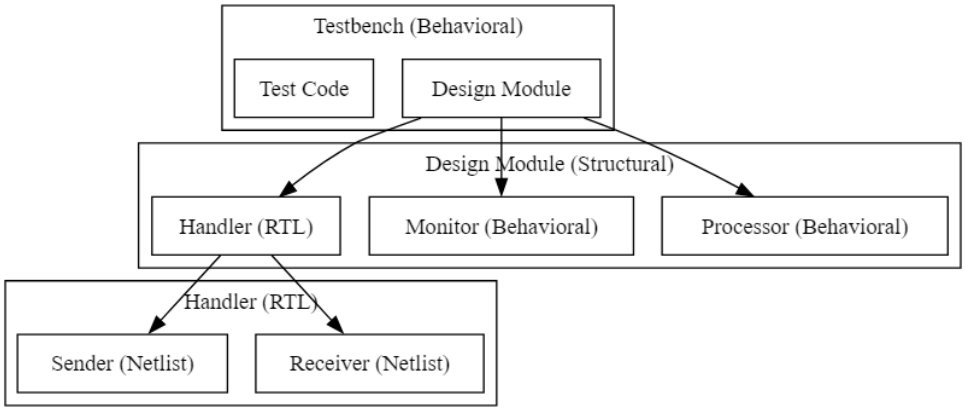
  Descreve o fluxo de dados entre blocos de lógica e os registros.

- **Nível de Sistema:**

  Maior nível de abstração, onde o foco é na integração entre os diferentes componentes do sistema.

1.3. Qual a entidade básica de um projeto descrito em linguagem Verilog? Apresente o diagrama hierárquico completo de um projeto e testbench descrito em linguagem Verilog. Explique a funcionalidade de cada bloco da hierarquia.

O módulo é a parte principal do projeto, contendo a descrição do hardware, enquanto o testbench é usado para simular e verificar o módulo. Juntos, eles formam a base para o desenvolvimento de sistemas digitais em Verilog.



Explicação dos Blocos:

- **Testbench (Behavioral)**
  - **Design Module:**
    Módulo de design estrutural que instancia os três blocos de design primários.
  - **Test Code:**
    Contém o código de teste para simular o design.
- **Design Module (Structural)**
  - **Processador (Behavioral):**
    Descreve a funcionalidade do processador.
  - **Monitor (Behavioral):**
    Descreve a funcionalidade do monitor.
  - **Handler (RTL):**
    Descreve parcialmente a funcionalidade do manipulador em código sintetizável e instancia dois submódulos.
- **Handler (RTL)**
  - **Receiver (Netlist):**
    Submódulo já sintetizado de macros do fornecedor ASIC.
  - **Sender (Netlist):**
    Submódulo feito à mão de primitivas Verilog.

1.4. Qual a diferença básica entre uma net é uma variável descritas em linguagem Verilog? Explique.

As diferenças entre “net” e “variável” podem ser vistas a seguir:

- **Net**
  - **Uso:**
    Representa conexões físicas entre elementos, como portas lógicas e flip-flops.
  - **Atualização:**
    São continuamente atualizadas. Quando uma entrada net muda, a saída é recalculada e atualizada imediatamente.
  - **Tipos Comuns:**
    “wire”, “wand”, “wor”.
  - **Atribuição Contínua:**
    São geralmente utilizadas com atribuições contínuas, que são avaliadas sempre que uma das entradas muda.
  - **Exemplo:**
    “assign net1 = input1 & input2;”
- **Variável**
  - **Uso:**
    Usadas para armazenar valores temporários dentro de procedimentos, como blocos “always” ou “initial”.
  - **Atualização:**
    São atualizadas em um ponto específico no tempo, geralmente no final de uma simulação de tempo ou de um bloco procedural.
  - **Tipos comuns:**
    “reg”, “integer”, “real”.
  - **Atribuição Procedural:**
    São usadas como atribuições procedurais dentro de blocos como “always”, com a atualização acontecendo de acordo com o fluxo de controle do bloco.

Em resumo, “NET” representa uma conexão física e é atualizada continuamente, enquanto a “VARIÁVEL” é usada para armazenar valores temporários e é atualizada em pontos específicos no tempo.

1.5. **Escreva um código Verilog para instanciar (associar pinos com as nets) o bloco apresentado na Figura 1:**

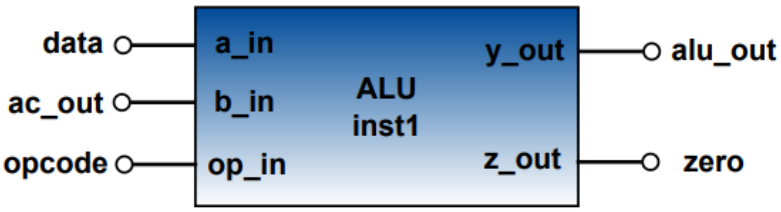


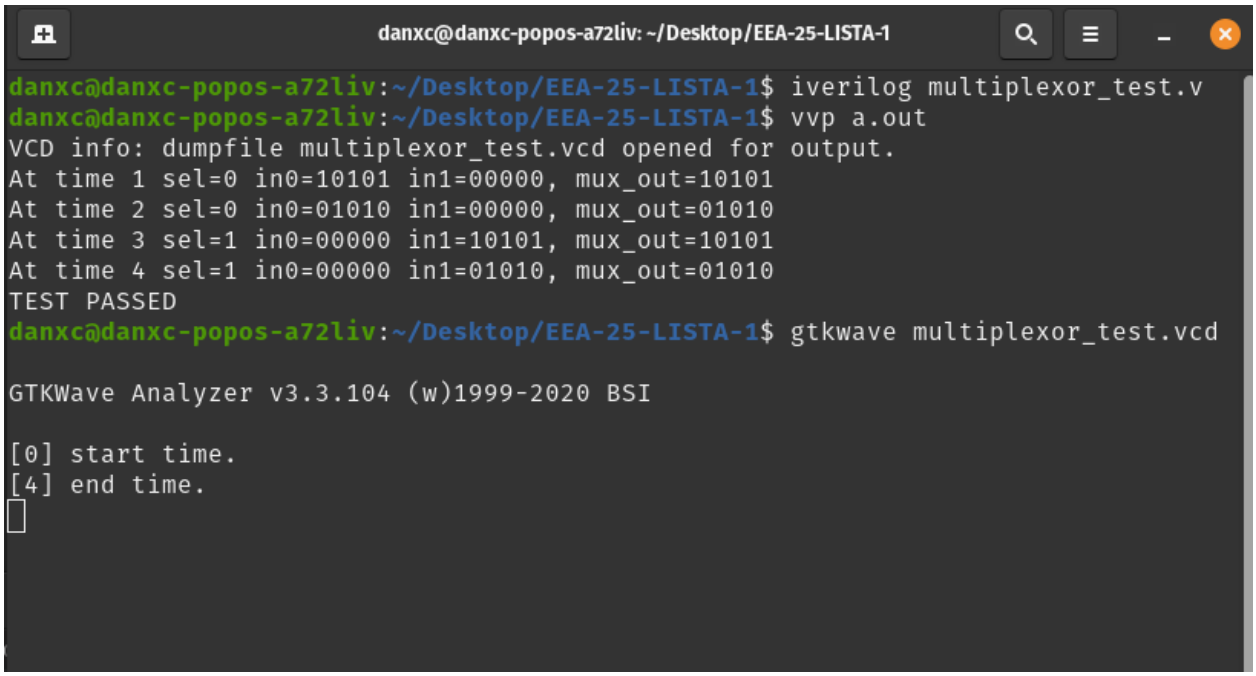
Figura 1: Diagrama do Exercício 5.

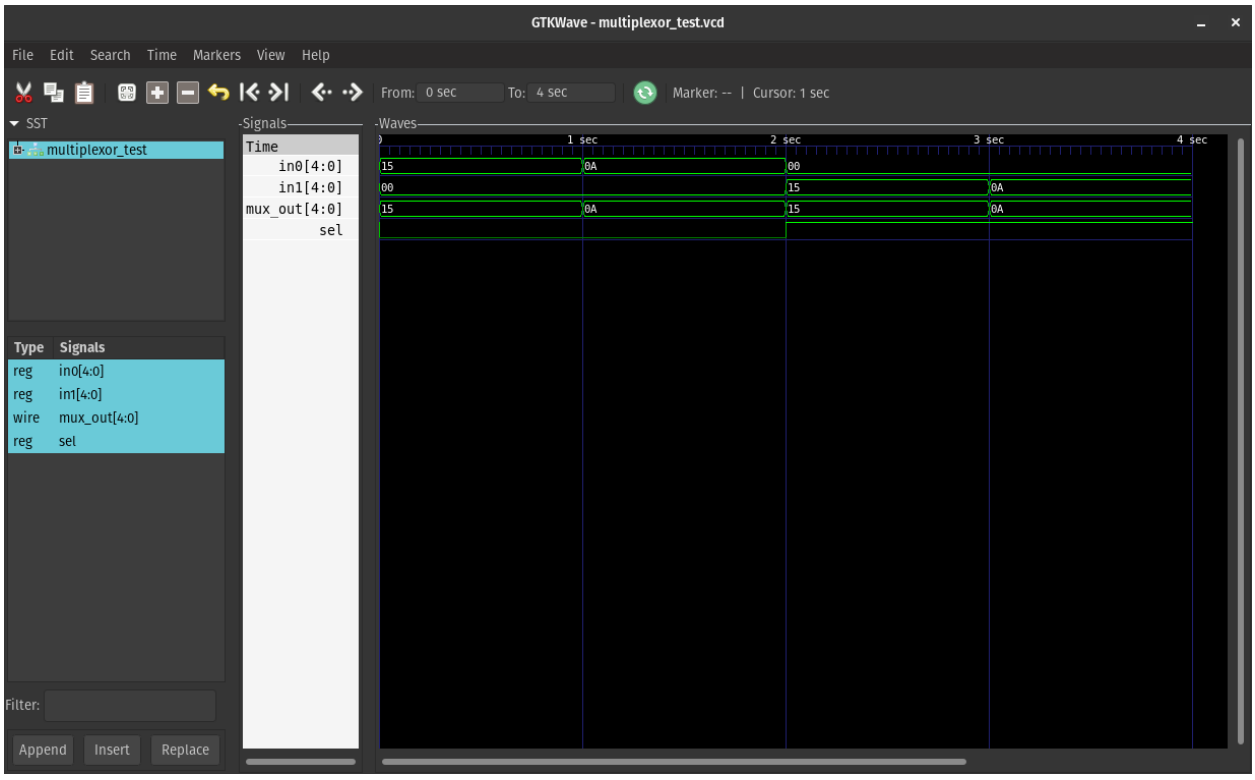
O código pode ser escrito da seguinte forma:

```
module top;
  // Declaração das nets
  wire data;
  wire ac_out;
  wire opcode;
  wire alu_out;
  wire zero;

  // Instanciação do módulo ALU
  ALU inst1 (
    .a_in(data),    // Associação do pino a_in com a net data
    .b_in(ac_out),  // Associação do pino b_in com a net ac_out
    .op_in(opcode), // Associação do pino op_in com a net opcode
    .y_out(alu_out),// Associação do pino y_out com a net alu_out
    .z_out(zero)    // Associação do pino z_out com a net zero
  );
  // Outras partes do código
endmodule
```

2. SIMULAÇÃO


danxc@danxc-popos-a72liv: ~/Desktop/EEA-25-LISTA-1
danxc@danxc-popos-a72liv:~/Desktop/EEA-25-LISTA-1$ iverilog multiplexor_test.v
danxc@danxc-popos-a72liv:~/Desktop/EEA-25-LISTA-1$ vvp a.out
VCD info: dumpfile multiplexor_test.vcd opened for output.
At time 1 sel=0 in0=10101 in1=00000, mux_out=10101
At time 2 sel=0 in0=01010 in1=00000, mux_out=01010
At time 3 sel=1 in0=00000 in1=10101, mux_out=10101
At time 4 sel=1 in0=00000 in1=01010, mux_out=01010
TEST PASSED
danxc@danxc-popos-a72liv:~/Desktop/EEA-25-LISTA-1$ gtkwave multiplexor_test.vcd
GTKWave Analyzer v3.3.104 (w)1999-2020 BSI
[0] start time.
[4] end time.


GTKWave - multiplexor_test.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 4 sec
Marker: -- | Cursor: 1 sec
SST
multiplexor_test
Type Signals
reg in0[4:0]
reg in1[4:0]
wire mux_out[4:0]
reg sel
Filter:
Append Insert Replace
Signals
Time
in0[4:0]
in1[4:0]
mux_out[4:0]
sel
Waves
1 sec
2 sec
3 sec
4 sec
15 0A 00
00 15 0A
15 0A 15 0A