UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA

LABORATÓRIO DE CIRCUITOS DIGITAIS

Introdução à Linguagem de Descrição de *Hardware*

Prof.: Pedro Thiago Valério de Souza
UFERSA – Campus Pau dos Ferros
pedro.souza@ufersa.edu.br

Linguagem de Descrição de *Hardware*

- Uma linguagem de descrição de *hardware* (HDL - *Hardware Description Language*) é a linguagem textual usada para descrever um circuito digital;
- O circuito pode (ou não) ser sintetizado em uma FPGA/CPLD;
- Linguagens comuns:
  - VHDL (VHSIC - *Very High Speed Integrated Circuits - Hardware Description Language*);
  - Verilog/SystemVerilog;
- Dentre as linguagens de descrição de *hardware* destaca-se o SystemVerilog;

Linguagem de Descrição de *Hardware*

- **Vantagens na utilização do HDL:**
  - Facilidade de atualização dos projetos;
  - Projeto em um nível mais alto de abstração;
  - Redução do tempo de projeto, de testes e implementação;
  - Simplificação quanto a sua documentação;
  - Facilidade de síntese de circuitos digitais em uma FPGA.

- **Desvantagens na utilização do HDL:**
  - O *hardware* gerado geralmente é menos otimizado.

Estrutura Básica do SystemVerilog

- Todo circuito digital em SystemVerilog é denominado de módulo;
- Todo código SystemVerilog inicia com **`module`** e encerra com **`endmodule`**;
- *Case-sensitive*;
  - Todas as palavras chaves são minúsculas;
- Espaços em branco não são interpretados;
- Todos os comandos terminam em ponto-e-vírgula;
- //: Comentário em uma única linha;
- /* */: Comentário em múltiplas linhas;
- Regras para os identificadores: deve sempre iniciar com uma letra ou com _ , jamais deve iniciar com um número.

Estrutura Básica do SystemVerilog

- Estrutura básica:

```
module nome_modulo(
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    ...
    <tipo_da_porta> <tipo_do_dado> nome_da_porta
);
    Funcionamento do circuito
endmodule;
```

Observação: Portas é a comunicação do circuito com o meio.

Estrutura Básica do SystemVerilog

- Estrutura básica:

```
module nome_modulo(
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    ...
    <tipo_da_porta> <tipo_do_dado> nome_da_porta
);
    Funcionamento do circuito
endmodule;
```

  - `input` → entrada;
  - `output` → saída;
  - `inout` → bidirecional.

Estrutura Básica do SystemVerilog

- Estrutura básica:

```
module nome_modulo(
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    ...
    <tipo_da_porta> <tipo_do_dado> nome_da_porta
);
    Funcionamento do circuito
endmodule;
```

  - `bit`: assume valores 0 ou 1;
  - `logic`: assume os valores 0, 1, - (*don't care*) ou z (alta impedância);
  - `tri`: saídas em *tri-state* (admitem alta impedância), mas que podem ter múltiplos acionadores;
    - Utilizado para fazer barramentos.

Estrutura Básica do SystemVerilog

- Estrutura básica:

```
module nome_modulo(
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    ...
    <tipo_da_porta> <tipo_do_dado> nome_da_porta
);
    Funcionamento do circuito
endmodule;
```

  - Tipos legados (Verilog):
    - **reg**: semelhante ao **logic**, apresenta a capacidade de armazenamento, porém não pode ser entrada (apenas saída);
    - **wire**: semelhante ao **logic**, porém não apresenta capacidade de armazenamento (pode ser tanto entrada como saída);

Estrutura Básica do SystemVerilog

- Estrutura básica:

```
module nome_modulo(
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    ...
    <tipo_da_porta> <tipo_do_dado> nome_da_porta
);
    Funcionamento do circuito
endmodule;
```

  - Declaração de barramentos (vetor de *bits*):

```
<tipo_da_porta> <tipo_do_dado> [MSB:LSB] nome_da_porta,
```

  - Exemplo:

```
input logic [3:0] a,
```

Estrutura Básica do SystemVerilog

- Estrutura básica:

```
module nome_modulo(
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    <tipo_da_porta> <tipo_do_dado> nome_da_porta,
    ...
    <tipo_da_porta> <tipo_do_dado> nome_da_porta
);
    Funcionamento do circuito
endmodule;
```

  - Descrição do como o circuito funciona;
    - Abordagem por fluxo de dados;
    - Abordagem hierárquica;
    - Abordagem comportamental.

Descrição por Fluxo de Dados

- Descreve o circuito em termos de expressões (lógicas e aritméticas) que relacionam entradas e saídas;
- Utiliza a diretiva **`assign`**;

```
assign <saida ou dado intermediário> = (constante, segmento de vetor
                                        ou expressão);
```

Descrição por Fluxo de Dados

- **Operações Lógicas:**

Operador	Operação Realizada	Exemplo: *a* = 0 *b* = 1
~	NOT	~*a* = 1 ~*b* = 0
&	AND	*a* & *b* = 0
|	OR	*a* | *b* = 1
^	XOR	*a* ^ *b* = 1

  - Exemplos:

```
assign x = ~a;
assign y = a & b;
assign z = a | b;
assign q = a & b & c;
assign w = a ^ b;
```

Descrição por Fluxo de Dados

Exemplo 1: Utilizando o SystemVerilog, descreva um circuito digital que possua as seguintes expressões lógicas de saída:

$$F_1(a,b,c) = x + y'z$$

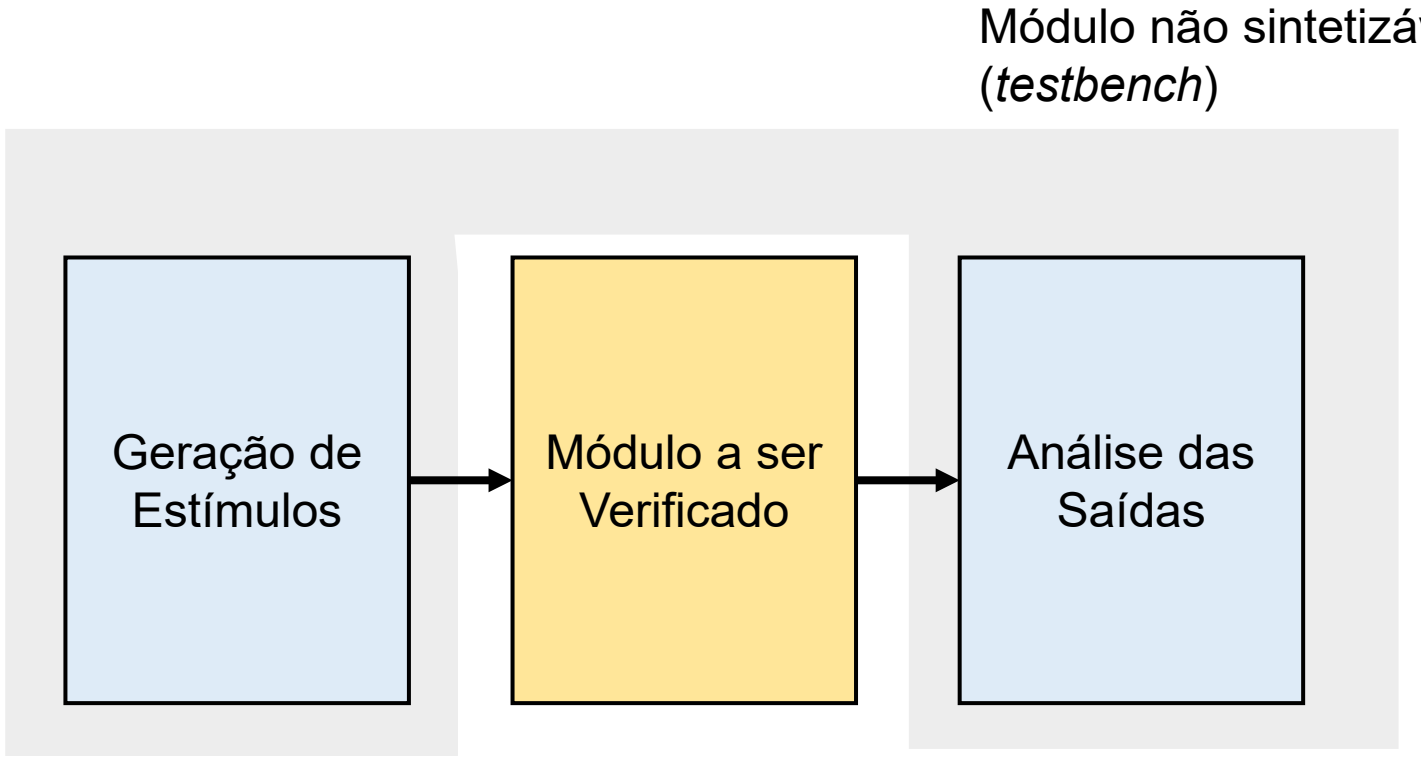
$$F_2(a,b,c) = xy' + x'z$$

Realize a simulação do circuito no ModelSim, obtendo todas as condições de entrada e elaborando a tabela-verdade.

Entradas			Saídas	
x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

Introdução à Verificação

- *Testbech*: *Script* automatizado para simulação (verificação) de módulos descritos em linguagem de descrição de *hardware*;
- Visão geral de um *testbench*:



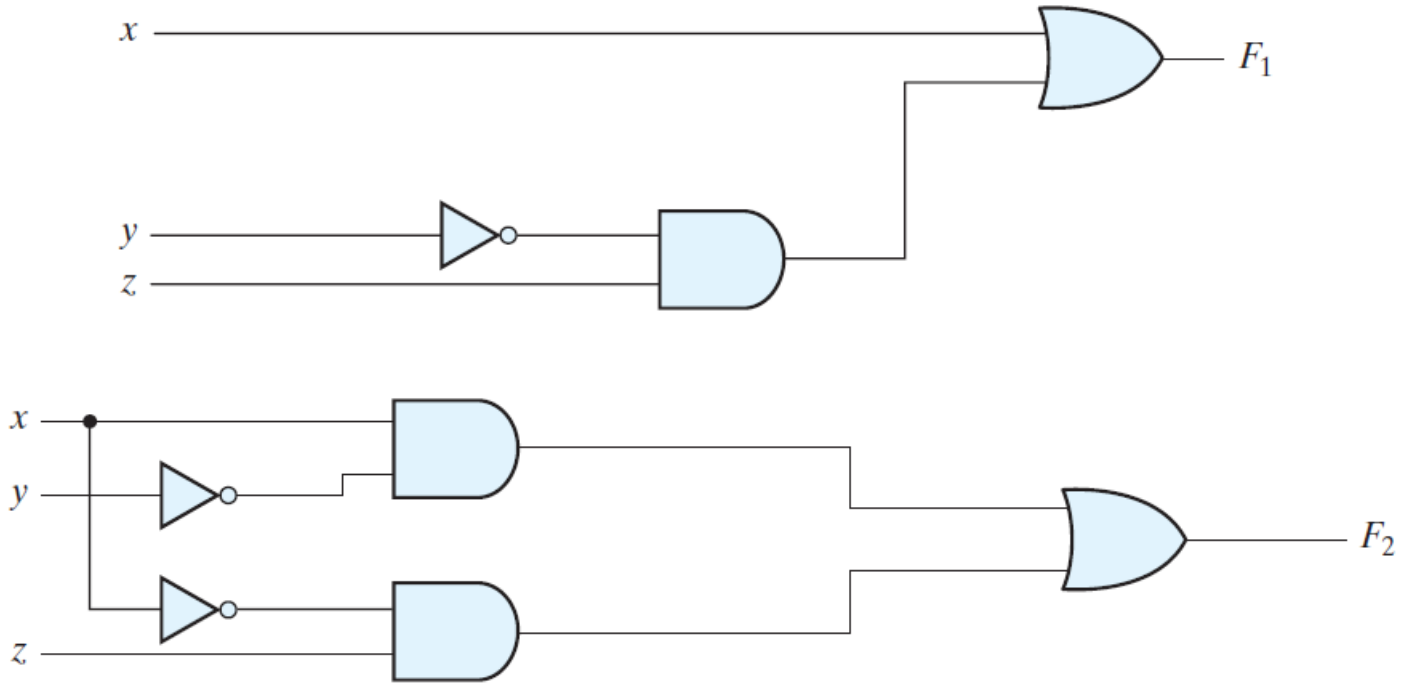
Descrição por Fluxo de Dados

- Declaração de sinais:
  - São valores intermediários, que são internos ao circuito;
  - Declarados dentro de um módulo e não podem ser “visualizados” fora deste módulo;
  - Formato de declaração de um sinal:

```
<tipo_do_sinal> nome_do_sinal;
```

Descrição por Fluxo de Dados

Exemplo 2: Utilizando o SystemVerilog, descreva um circuito digital que possua o seguinte diagrama de circuito lógico:



$$F_1(a,b,c) = x + y'z$$

$$F_2(a,b,c) = xy' + x'z$$

Entradas			Saídas	
x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0