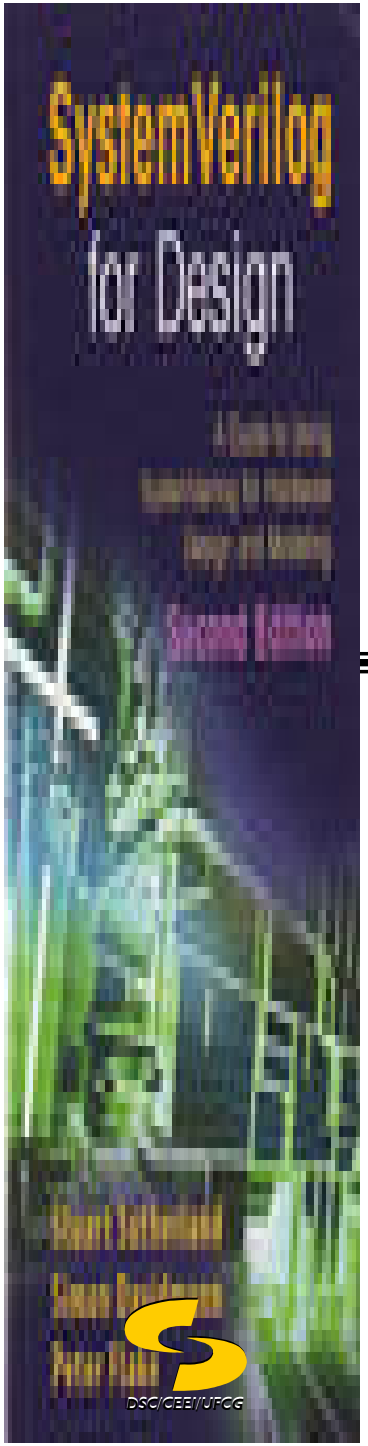


Universidade Federal de Campina Grande

Departamento de Sistemas e Computação

Curso de Bacharelado em Ciência da Computação

Organização e Arquitetura de Computadores I

Linguagem de Descrição de Hardware (Parte I - Adicional)

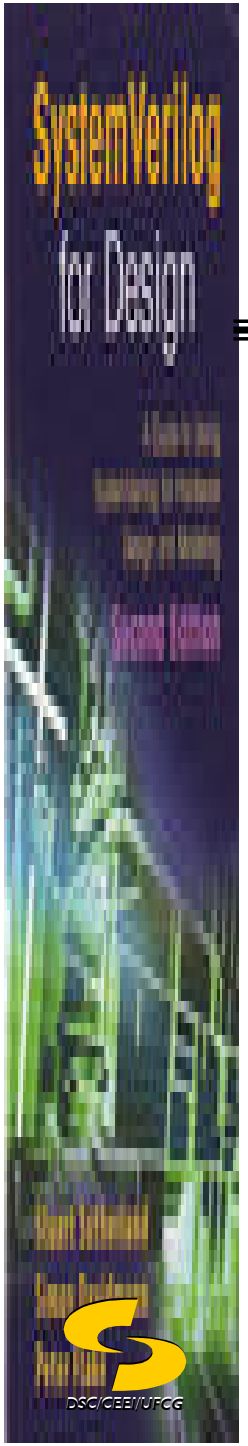
Profª Joseana Macêdo Fechine Régis de Araújo

joseana@computacao.ufcg.edu.br

Carga Horária: 60 horas

Tópicos

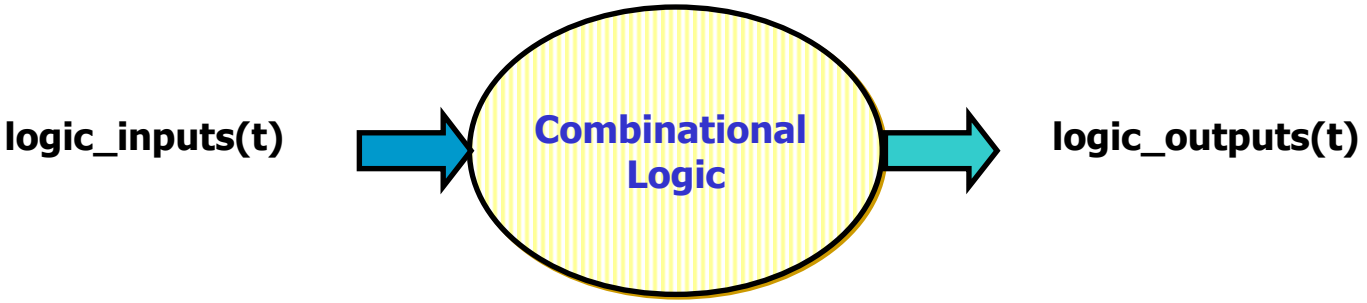
- ***Hardware Description Language*** **(HDL)**
  - Conceitos Básicos – continuação
    (Lógica Combinacional)



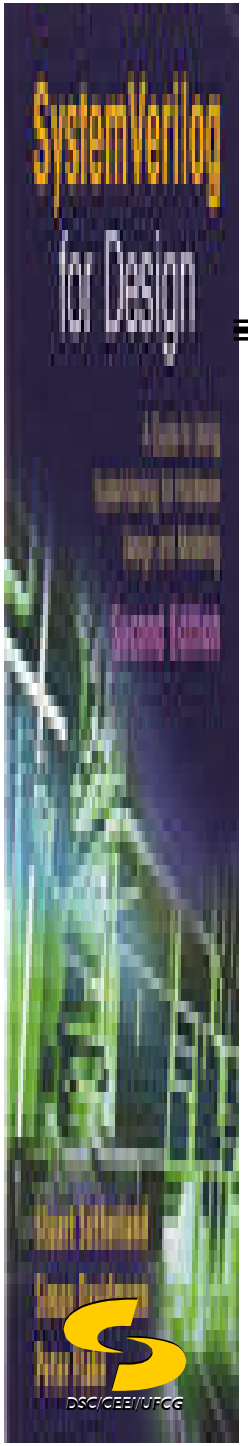
SystemVerilog: Síntese de Lógica combinatória

- Função lógica combinatória pode ser representada por:

logic_output(t) = f(logic_inputs(t))

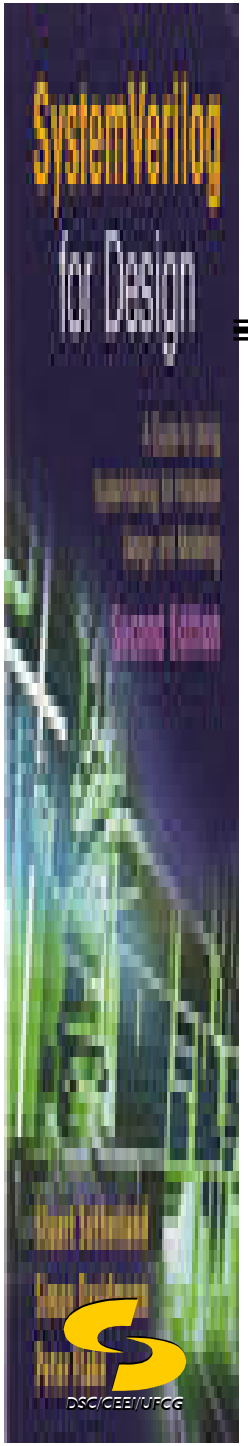


- Regras
  - Evite modelagem dependente de tecnologia, ou seja, implemente a função, não o *timing*.
  - Lógica combinatória não pode ter realimentação.
  - Especifique a saída de uma função combinatória para todas as possíveis combinações de entrada.



Estilos para Lógica Combinatória Sintetizável

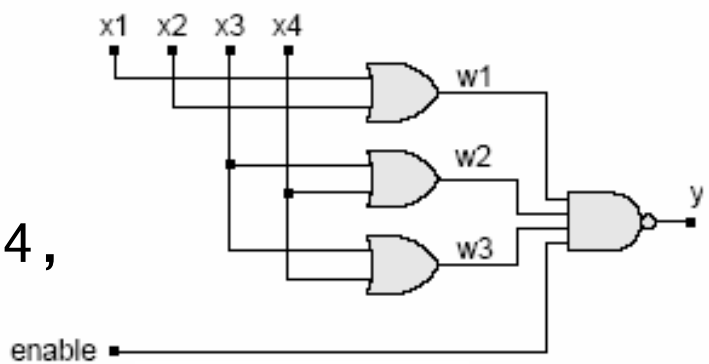
- Os seguintes estilos são possíveis:
  - *Netlist* de instâncias de portas e de primitivas Verilog (completamente estrutural).
  - UDP (*User Defined Primitive*) combinatória (somente algumas ferramentas)
  - Funções
  - Atribuições permanentes
  - Instruções comportamentais
  - *Tasks* sem controle de atraso ou de eventos
  - Interconexões dos elementos acima



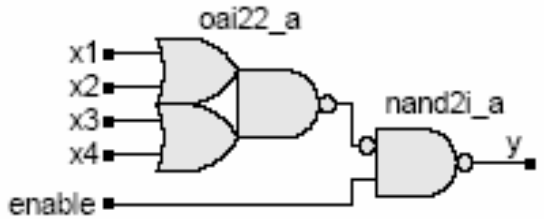
Síntese de Lógica Combinatória – *Netlist* de Portas

- Ferramentas de síntese otimizam uma *netlist* feita a partir de primitivas Verilog
- Exemplo:

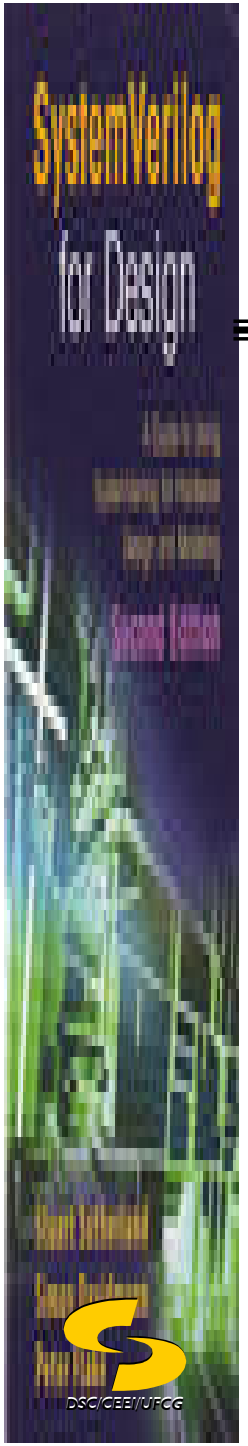
```
module or_nand_1 (
   input enable, x1, x2, x3, x4,
   output logic y);
   logic w1, w2, w3;
   or (w1, x1, x2);
   or (w2, x3, x4);
   or (w3, x3, x4); // redundant
   nand (y, w1, w2, w3, enable);
endmodule
```



pré-síntese

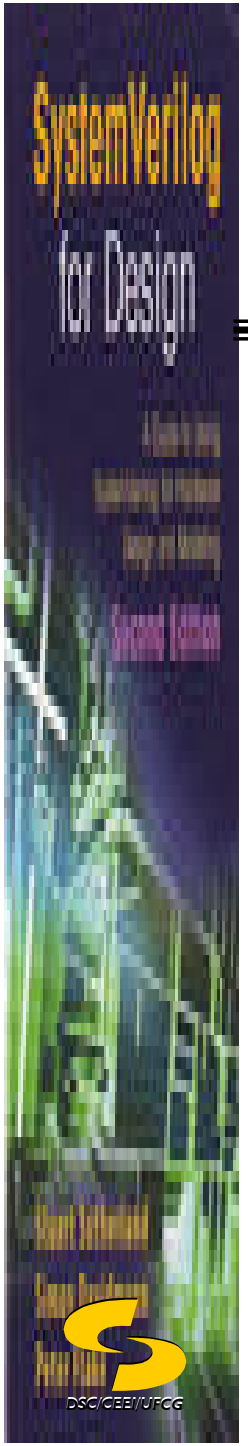


pós-síntese



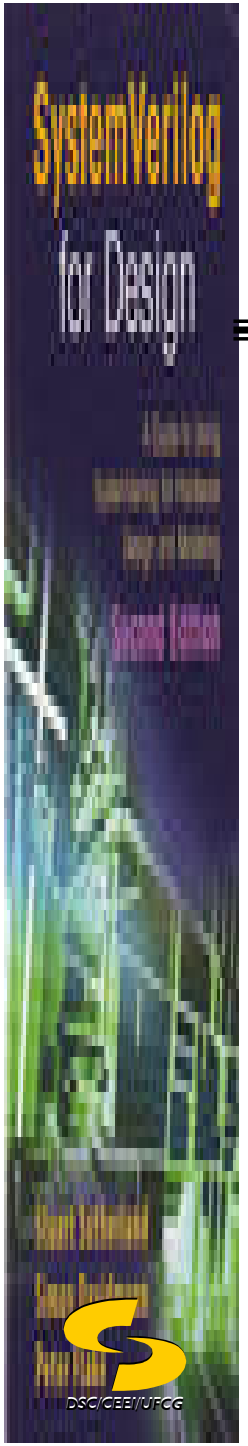
Síntese de Lógica Combinatória – *Netlist* de Portas

- Passos gerais da síntese:
  1) Portas lógicas são traduzidas em equações booleanas.
  2) As equações booleanas são otimizadas.
  3) Equações booleanas são mapeadas para células de biblioteca.
- Funções complexas, que são modeladas por portas primitivas, não podem ser mapeadas para células de biblioteca mais complexas (p.ex. somador, multiplicador).
- Diretivas de síntese permitem preservar a estrutura, caso células equivalentes existam na biblioteca.



Construções a serem evitadas para Síntese combinatória

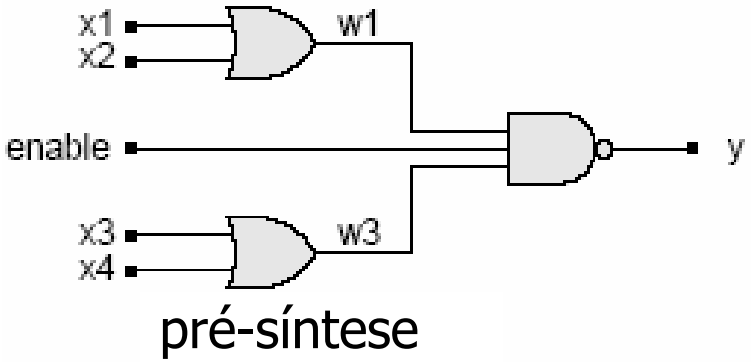
- Controle de evento de transição
- Múltiplos eventos de controle com o mesmo comportamento
- Eventos derivados (apelidos de eventos)
- Laços de realimentação
- Atribuições que contenham controle por evento ou controle por atraso
- Blocos **fork** ... **join** (processos paralelos)
- Instruções **wait**
- Instruções externas de **disable**
- Laços com controle de tempo
- Laços que dependam de dados
- *Task* que contém controle de tempo
- UDPs (*User Defined Primitives*) seqüenciais



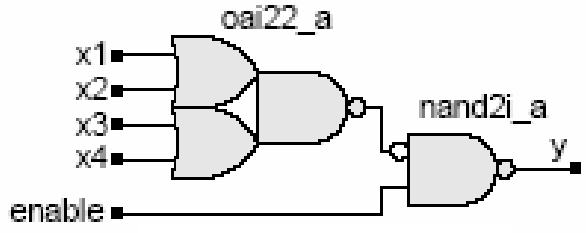
Síntese de Lógica Combinatória – Atribuição permanente

Exemplo:

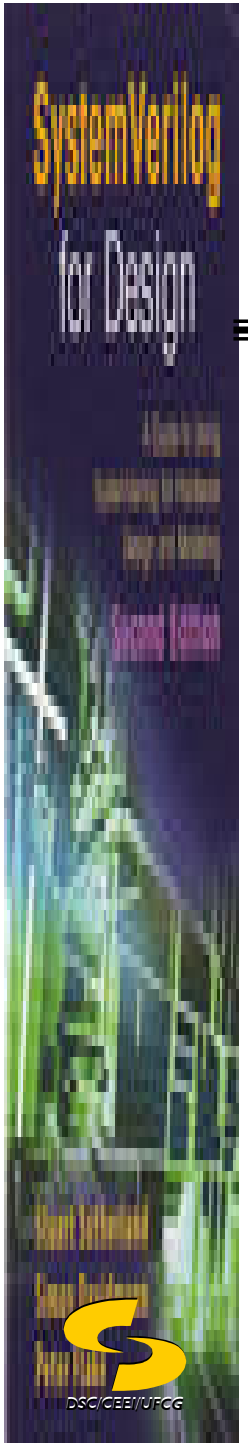
```
module or_nand_2 (
  input enable, x1, x2, x3, x4,
  output logic y);
  always_comb y <= !(enable & (x1 | x2) & (x3 | x4));
endmodule
```



pré-síntese



pós-síntese



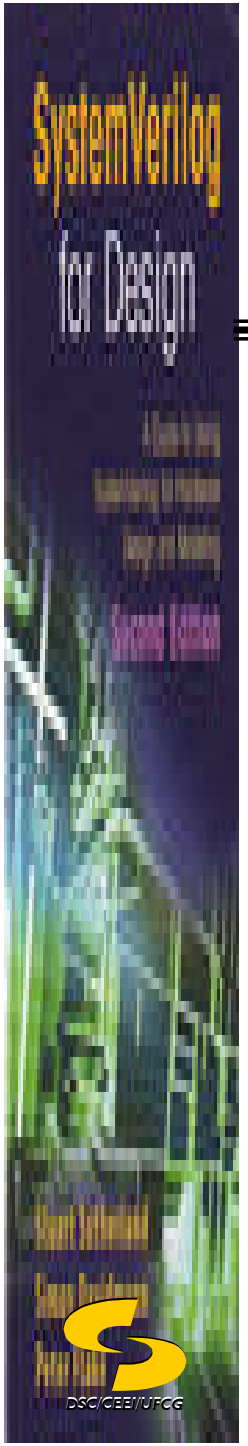
Síntese de Lógica Combinatória – Estilo comportamental

Exemplo:

```
module or_nand_3 (
   input enable, x1, x2, x3, x4;
   output logic y);

   always_comb
     if (enable)
       y <= !((x1 | x2) & (x3 | x4));
     else
       y <= 1; // operand is a constant.
endmodule
```

Obs*:* **Todas** as entradas para o comportamento precisam ser incluídas na lista de eventos, se não um ***latch*** será inferido.



Síntese de Lógica Combinatória – Funções

Exemplo:

```
module or_nand_4 (
   input enable, x1, x2, x3, x4;
   output logic y);
   always_comb y <= or_nand(enable, x1, x2, x3, x4);

   function or_nand;
      input enable, x1, x2, x3, x4;
      begin
         or_nand <= ~(enable & (x1 | x2) & (x3 | x4));
      end
   endfunction
endmodule
```