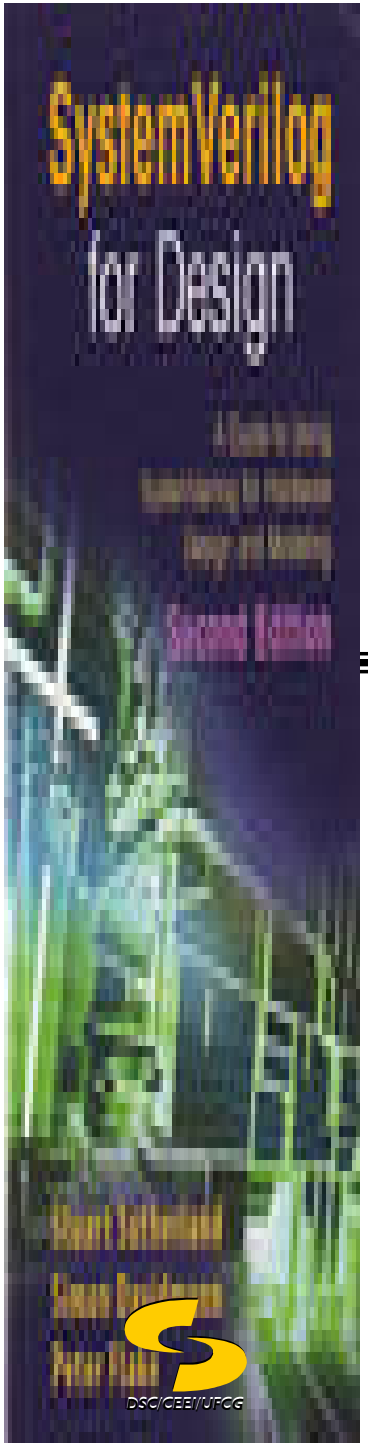


Universidade Federal de Campina Grande

Departamento de Sistemas e Computação

Curso de Bacharelado em Ciência da Computação

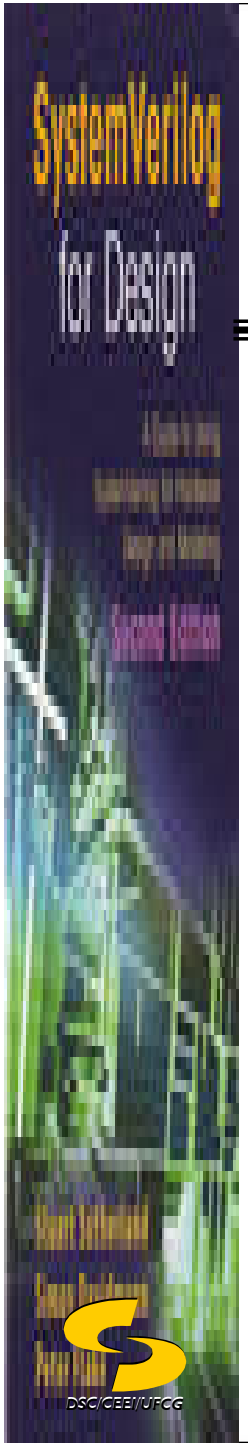
Organização e Arquitetura de Computadores I

Linguagem de Descrição de Hardware (Parte I)

Profª Joseana Macêdo Fechine Régis de Araújo

joseana@computacao.ufcg.edu.br

Carga Horária: 60 horas



Tópicos

- ***Hardware Description Language*** **(HDL)**
  - Conceitos Básicos

Hardware Description Language (HDL)

- Para que precisamos de uma Linguagem de Descrição de Hardware?
- Modelar, Representar e simular hardware digital
  - Concorrência
  - Paralelismo
  - Semântica para valores de sinais no tempo
- Construções e semântica especiais
  - Transições (bordas) de valores de sinais
  - Atrasos de propagação de sinais
  - Verificação de condições temporais

Hardware Description Language (HDL)

- A unidade básica – o módulo
- Módulo (`module`)
  - Descreve a funcionalidade do circuito
  - Define terminais (pinos, portas) de entrada e saída
- Exemplo: Um computador
  - Funcionalidade: Realizar operações programadas
  - Portas de entrada/saída: conector para teclado, mouse, conector VGA, conector USB

Hardware Description Language (HDL)

- **Comentários**

  // comentário de uma linha só

  /* outra forma de comentário de uma linha */

  /* inicio de comentário com múltiplas linhas
  todo text é ignorado
  termina com a linha abaixo
  */

- **Números**

  decimal, hexadecimal, binario

  6'd33, 8'hA6, 4'b1101

  unsized decimal form

  size base form

  include underlines, +,-

Hardware Description Language (HDL)

- **Cadeias de caracteres**
  - "Delimite usando aspas numa mesma linha"
  - limitados a 1024 caracteres
- ***Identifier***

  A ... Z

  a ... z

  0 ... 9

  Underscore
- Primeiro caractere de um ***identifier*** não pode ser um dígito
- **Verilog diferencia letras maiúsculas de minúsculas.**

Tipos de dados

- Palavra: Um vetor de bits

```
logic [ msb : lsb ] nome;
```

- Exemplos

```
logic fio;          // um fio (1 bit)
logic [3:0] cabo; // Um cabo de 4 fios
```

- Um vetor de palavras

```
logic [msb:lsb] memory1 [lower:upper];
```

- Exemplo

```
// um vetor de 64 palavras de 4 bits:
logic [3:0] mem1 [0:63];
// um vetor de 5 palavras de 1 bit:
logic mem2 [4:0];
```

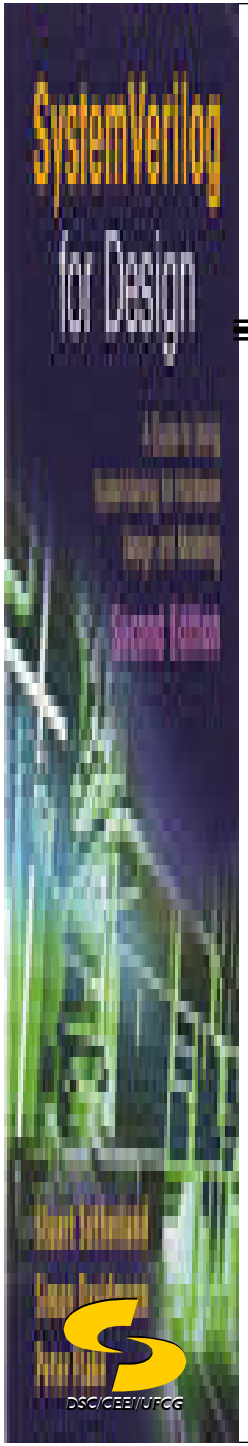
Tipos de dados - signed

- Uma vetor de bits definido com logic representa por default um número sem sinal.
- Representação com sinal em complemento de 2:

```
logic signed [ msb : lsb ] nome;
```

- Exemplo

```
logic signed [7:0] saldo; // de -128 a 127
```



Tipos de dados compostos

- Definição de um novo tipo de dado

```
typedef struct { logic …;
logic …; } nome_do_tipo;
```

- Exemplo de declaração do novo tipo

```
typedef struct { logic a;
logic [3:0] b; } abba_t;
```

- Exemplo de declaração de barramento usando o novo tipo

```
abba_t mamamia;
```

Lógica Combinacional

- A atribuição combinacional tem caráter permanente

  Formato: `always_comb net <= expression;`

  Exemplo: `always_comb sum <= a ^ b;`

- Todas as atribuições combinacionais valem (executam) simultaneamente.

- A ordem das atribuições dentro do arquivo Verilog não tem efeito.

Hardware Description Language (HDL)

- **Definição geral**

```
module module_name (
   port_list );
   …
   variable declaration;
   …
   description of behavior
endmodule
```

- **Exemplo**

```
module HalfAdder (
  input logic A, B,
  output logic Sum, Carry);

  always_comb Sum <= A ^ B;
  //^ denotes XOR
  always_comb Carry <= A & B;
  // & denotes AND
endmodule
```

Hardware Description Language (HDL)

- Várias atribuições podem ser acomodados num bloco begin … end como em Pascal (equivalente a { … } em C e Java).

```
…
always_comb begin
   Sum <= A ^ B;
   Carry <= A & B;
end
…
```

Hardware Description Language (HDL)

- Todas as operadores de C e Java, mais alguns especiais (ver livro).

  Concatenação { }

  Exemplo: `{ a, 4'b1010 }`

  Repetição: `{ vezes { expressão } }`

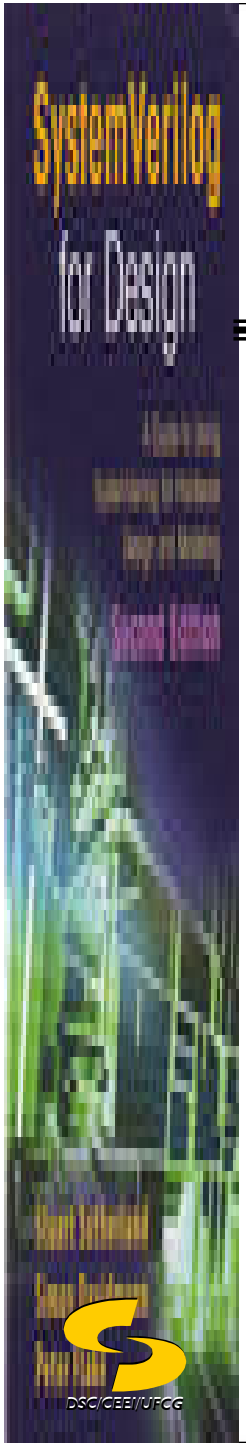
  Exemplo: `{ 4{a} }`

HDL - Construções Procedurais

- Existem duas:
  - initial: Executa uma única vez no início da simulação, NÃO sintetizável
  - always_comb: Executa repetidamente, sintetizável
- Exemplo:

```
…
initial begin
   Sum <= 0;
   Carry <= 0;
end
…
```

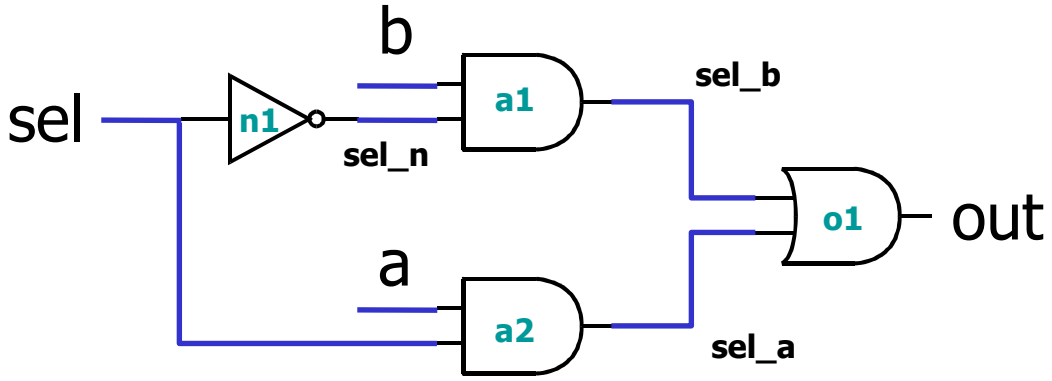
```
…
always_comb begin
   Sum <= A ^ B;
   Carry <= A & B;
end
…
```

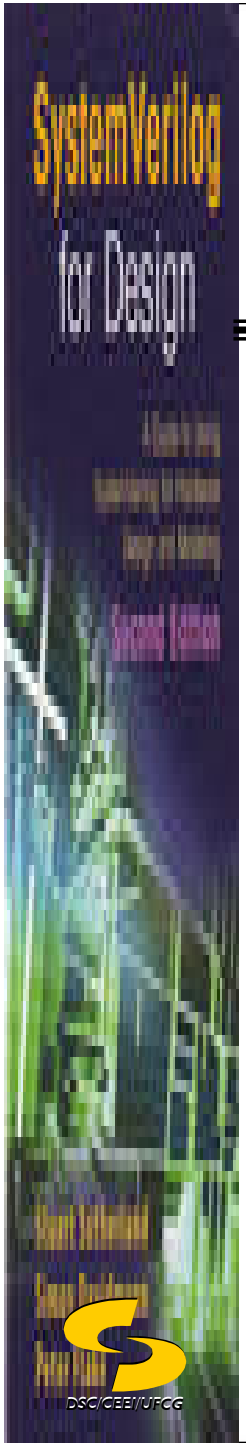


HDL - Estilos de Descrição

- **Estrutural**: representa circuitos lógicos usando primitivas da linguagem Verilog
- Exemplo:

```
not n1(sel_n, sel);
and a1(sel_b, b, sel_n);
and a2(sel_a, a, sel);
or  o1(out, sel_b, sel_a);
```





HDL - Modelo estrutural

- Execução: Concorrente
- Formato (portas lógicas primitivas):

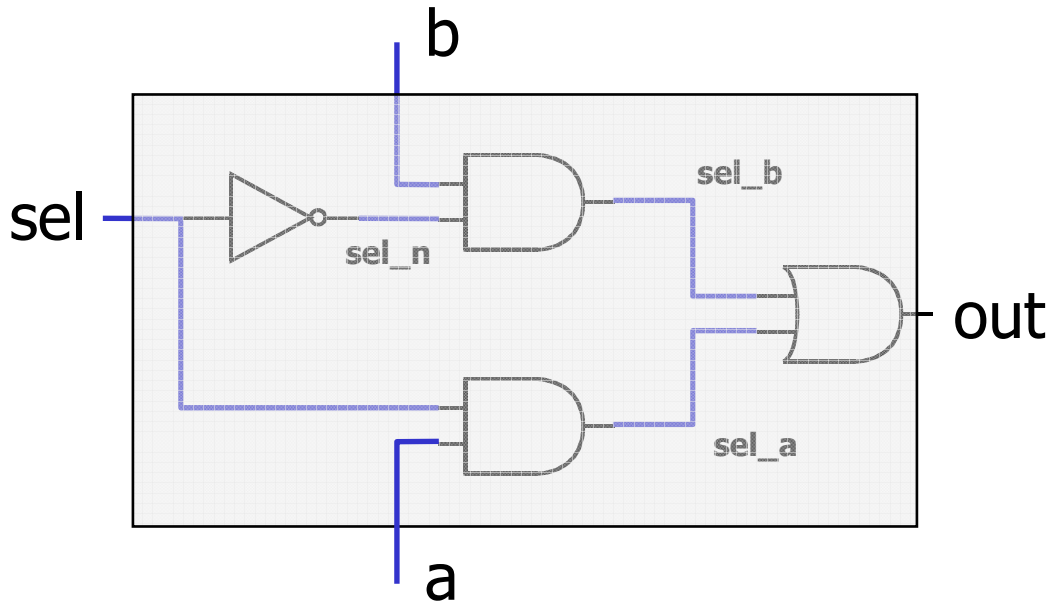
```
and G2(Carry, A, B);
```

  - Primeiro parâmetro (Carry) – Output
  - Outros parâmetros (A, B) - Inputs

HDL - Estilos de Descrição

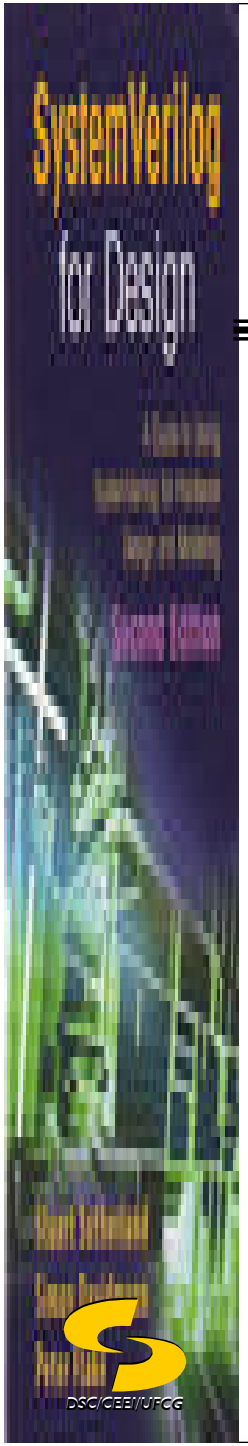
- **Fluxo de dados**: Representa sinais de saída em função de sinais de entrada
- Exemplo:

```
always_comb out <= (sel & a) | (~sel & b);
```



HDL - Modelo de Fluxo de Dados

- Usa atribuição permanente para sinais
  - Format: `always_comb [ delay ] net <= expression;`
  - Exemplo: `always_comb sum <= a ^ b;`
- *delay*: Atraso de propagação da expressão para o sinal.
- Todas as atribuições permanentes executam simultaneamente.
- A ordem das atribuições dentro do arquivo SystemVerilog não tem efeito sobre a execução.



HDL - Modelo de Fluxo de Dados

- Atraso
  - Exemplo: `always_comb #2 sum <= a ^ b;`
  - "#2" indica 2 unidades de tempo
  - Sem especificação de atraso: 0 (*default*)
- Associação entre unidade de tempo e tempo simulado
  - `` `timescale `` unidade/resolução
  - Exemplo: `` `timescale `` 1 ns / 10 ps
    - 1 unidade de tempo = 1 ns
    - resolução é 10 ps (0.01 ns)

HDL - Modelo de Fluxo de Dados

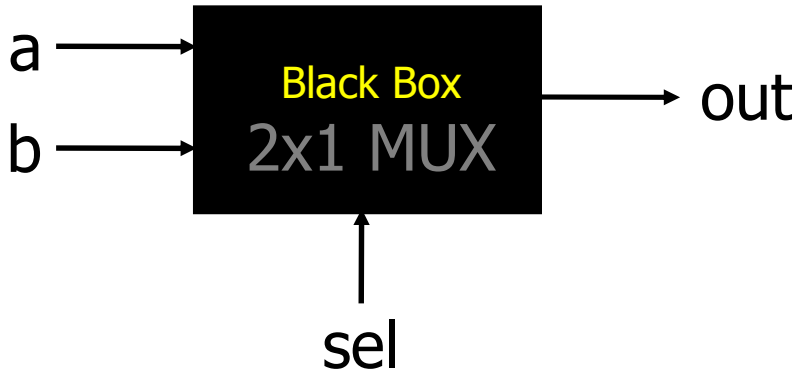
- Exemplo:

```
`timescale 1ns/10ps
module HalfAdder (input logic A, B,
                  output logic Sum, Carry);
  always_comb #3 Sum <= A ^ B;
  always_comb #6 Carry <= A & B;
endmodule
```

HDL - Estilos de Descrição

- **Comportamental**: representa o comportamento na forma de um algoritmo.
- Exemplo:

```
always_comb
 if (sel == 0)
   out <= b;
 else out <= a;
```



HDL - Modelo comportamental

- **Exemplo:**

```
module mux_2x1(input logic a, b, sel,
               output logic out);
  always_comb
     if (sel == 1) out <= a;
     else out <= b;

endmodule
```

HDL - Instruções Condicionais

- Format:

```
if (condition)
  procedural_statement
else if (condition)
  procedural_statement
else
  procedural_statement
```

- Exemplo:

```
if (reset)
   Q <= 0;
else
   Q <= D;
```

HDL - Instruções Condicionais

Instrução ***Case***

- Exemplo:

```
case (X)
    2'b00: Y <= A + B;
    2'b01: Y <= A - B;
    2'b10: Y <= A / B;
endcase
```

HDL - Diretivas de Compilação

- `define – (similar ao #define em C) usado para definir parâmetro global
- Exemplo:

```
`define BUS_WIDTH 16
 logic [ `BUS_WIDTH - 1 : 0 ] System_Bus;
```

- `include – usado para incluir outro arquivo
- Exemplo

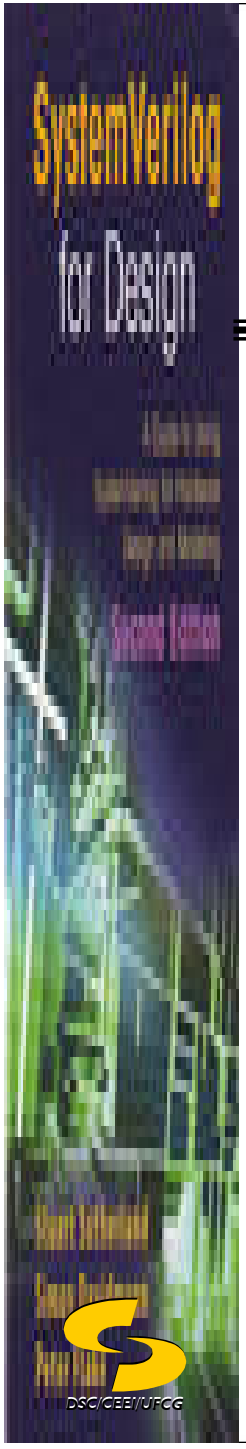
```
`include "./fulladder.sv"
```

HDL - Estilos de Descrição

- **RTL** (*Register Transfer Level*): descreve o que acontece a cada transição ativa do sinal de relógio

- Exemplo:

```
always_ff @(posedge clock) begin
 pisca <= ~pisca;
end
```



HDL - Memórias

- Um vetor de registradores

```
logic [ msb : lsb ] memory1 [ upper : lower ];
```

- Exemplos:

```
logic [ 3 : 0 ] mem [ 63 : 0 ];
// Um array de 64 registradores de 4 bits

logic mem [ 4 : 0 ];
// Um array de 5 registradores de 1 bit.
```