DCA0212.1 – LABORATÓRIO DE CIRCUITOS DIGITAIS
Aula 3:
VHDL - Components

Prof. Me. Sérgio Natan Silva
sergionatan@dca.ufrn.br

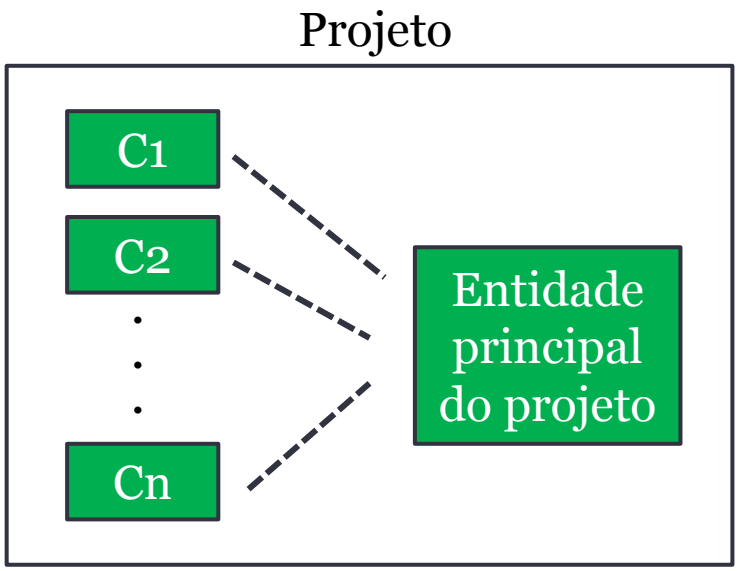
Agenda

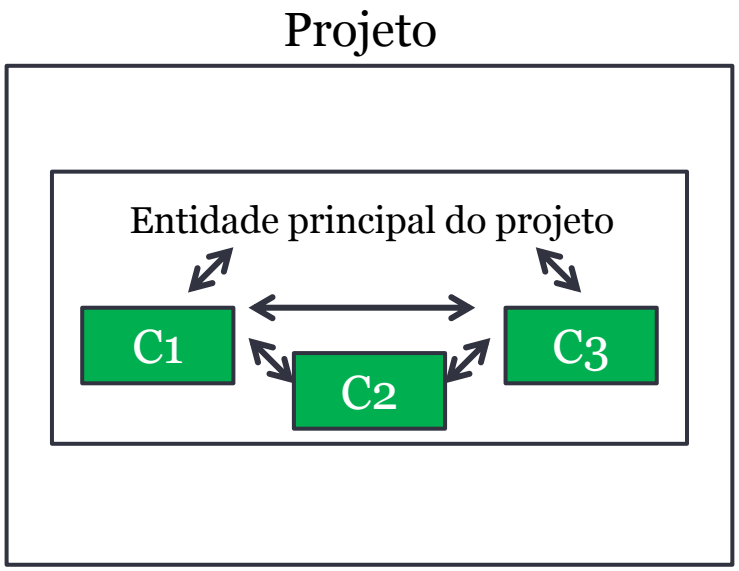
- Introdução
- Componentes
  - Criação
  - Declaração e Solicitação
  - Uso
- Exemplos
  - Somador completo de 8 bits.

Introdução

- Componentes são estruturas que possibilitam interligar entidades.
- Pode-se dizer que um componentes está para uma função ou procedimento em uma linguagem de programação convencional.
- É uma forma de trabalhar em módulos e de subdividir as tarefas entre outros de uma mesma equipe.

Introdução


Projeto
C1
C2
Cn
Entidade principal do projeto


Projeto
Entidade principal do projeto
C1
C2
C3

Componentes

- Primeiramente é necessário desenvolver algum circuito em VHDL.
- No nosso projeto vamos nos concentrar em um somador completo de 1 bit.
- O somador pode ser descrito pelas seguintes expressões lógicas:
  - s <= a XOR b XOR cin;
  - cout <= (a AND b) OR (a AND cin) OR (b AND cin);

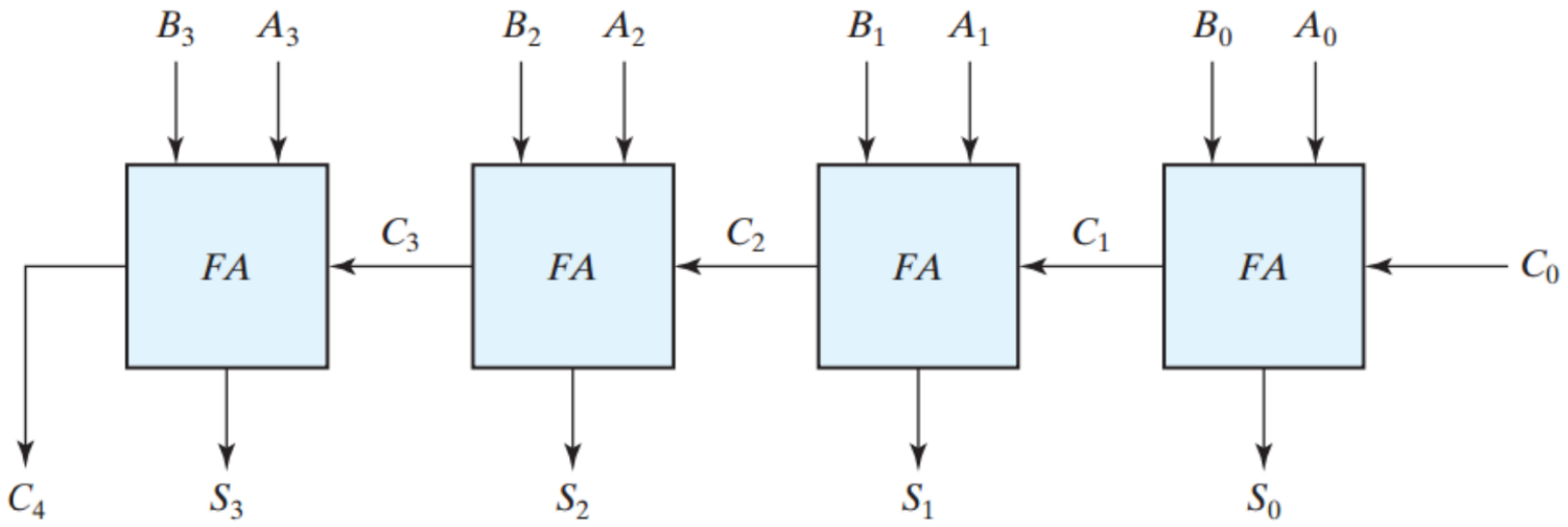
Componentes

- Crie um arquivo em VHDL que contemple as expressões demonstradas anteriormente.
- Nomeie o arquivo com o nome somador.
- Nomeie a entidade do circuito com o mesmo nome do arquivo.
- Defina os ports e a arquitetura do circuito.
- Salve.
- A partir deste ponto temos um possível circuito que pode ser utilizado como componente.

Componentes

```
entity somador is
        port(a, b, cin : in bit;
        s, cout : out bit);
end somador;
architecture comportamento of somador is
begin
        s <= a xor b xor cin;
        cout <= ( (a xor b) and cin ) or (a and b);
end comportamento;
```

Somador de 4 bits



Somador de 4 bits

```
entity somador_quatro_bits is
    port ( a : in bit_VECTOR (3 downto 0);
    b : in bit_VECTOR (3 downto 0);
    c0 : in bit;
    s : out bit_VECTOR (3 downto 0);
    c4 : out bit);
end somador_quatro_bits;
architecture comportamento of somador _quatro_bits is
signal c3, c2, c1: bit;
begin
    s(0) <= a(0) xor b(0) xor c0;
    c1 <= ( (a(0) xor b(0)) and c0 ) or (a(0) and b(0));
    s(1) <= a(1) xor b(1) xor c1;
    c2 <= ( (a(1) xor b(1)) and c1 ) or (a(1) and b(1));
    s(2) <= a(2) xor b(2) xor c2;
    c3 <= ( (a(2) xor b(2)) and c2 ) or (a(2) and b(2));
    s(3) <= a(3) xor b(3) xor c3;
    c4 <= ( (a(3) xor b(3)) and c3 ) or (a(3) and b(3));
end comportamento;
```

Usando o Componente

```
component somador is
    port(a, b, c_in : in bit;
    s, c_out : out bit);
end component;
```

Usando o Componente

```
entity somador_quatro_bits is
    port ( a : in bit_VECTOR (3 downto 0);
    b : in bit_VECTOR (3 downto 0);
    cin : in bit;
    s : out bit_VECTOR (3 downto 0);
    cout : out bit);
end somador_quatro_bits;
architecture comportamento of somador_quatro_bits is
component somador is
        port(a, b, c_in : in bit;
        s, c_out : out bit);
end component;
signal c: bit_vector(4 downto 0);
begin
    u1: somador port map (a => a(0), b => b(0), c_in => c(0), s => s(0), c_out => c(1));
    u2: somador port map (a => a(1), b => b(1), c_in => c(1), s => s(1), c_out => c(2));
    u3: somador port map (a => a(2), b => b(2), c_in => c(2), s => s(2), c_out => c(3));
    u4: somador port map (a => a(3), b => b(3), c_in => c(3), s => s(3), c_out => c(4));
    c(0) <= cin;
    cout <= c(4);
end comportamento;
```

Usando o Componente – FOR - GENERATE

```
gen: for i in 0 to 3 generate
        uut: somador port map (a => a(i), b => b(i), c_in => c(i), s => s(i), c_out =>
   c(i+1));
end generate;
c(0) <= cin;
cout <= c(4);
```