

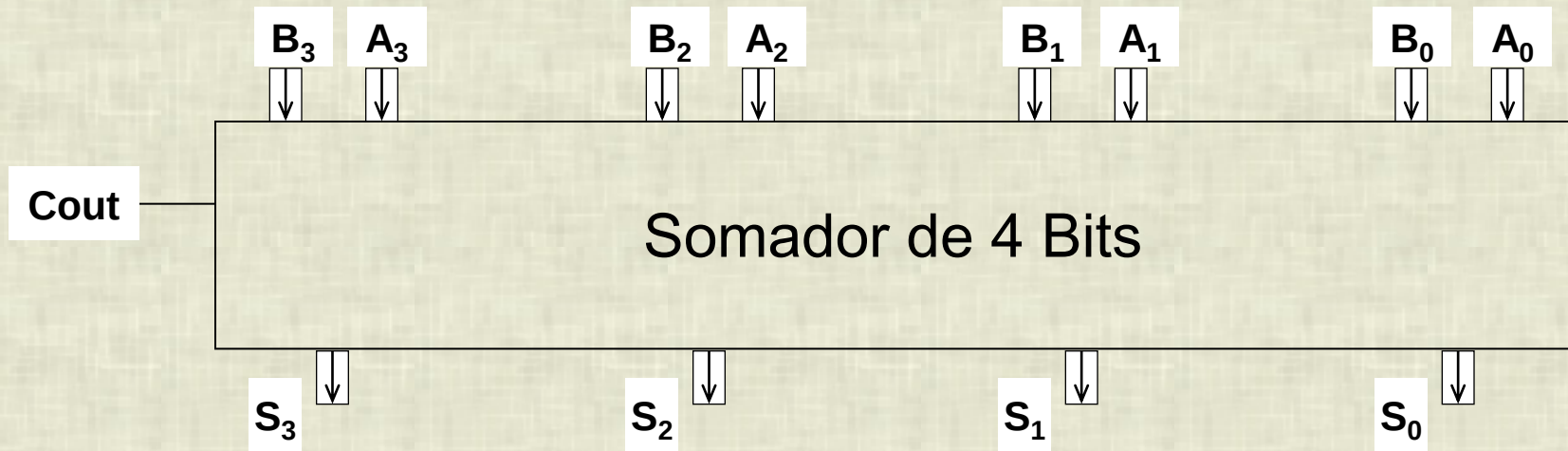
ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

Projeto de Somador com e sem Sinal

Descrição Comportamental

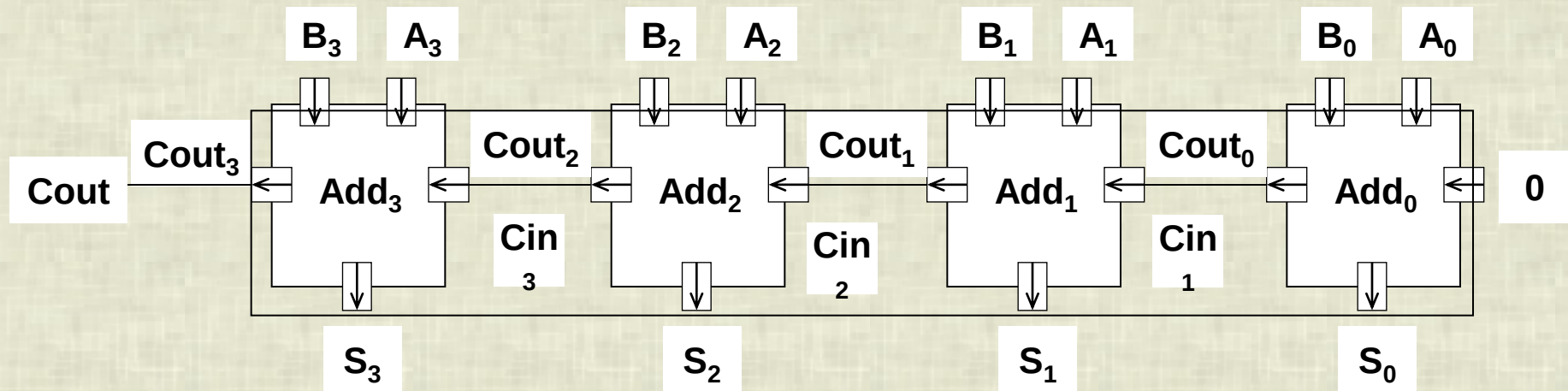
prof. Dr. César Augusto M. Marcon
prof. Dr. Edson Ifarraguirre Moreno

Somador Estrutural de 4 Bits (Entidade)



```
entity Adder is
  port
  (
    A, B: in std_logic_vector(3 downto 0);
    cout: out std_logic;
    S : out std_logic_vector(3 downto 0)
  );
end Adder;
```

Somador Estrutural de 4 Bits (Arquitetura)



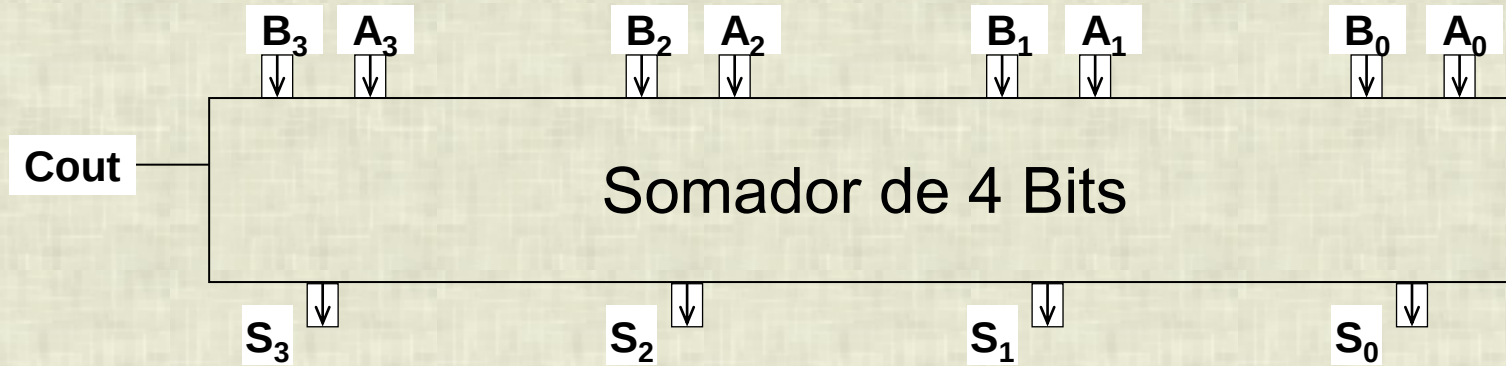
```

library IEEE;
use IEEE.std_logic_1164.all;

architecture Somador of Adder is
    signal c: std_logic_vector(3 downto 0);
begin
    A0: entity Add port map(cin=>'0', A=>A(0), B=>B(0), cout=>c(0), s=>S(0));
    A1: entity Add port map(cin=>c(0), A=>A(1), B=>B(1), cout=>c(1), s=>S(1));
    A2: entity Add port map(cin=>c(1), A=>A(2), B=>B(2), cout=>c(2), s=>S(2));
    A3: entity Add port map(cin=>c(2), A=>A(3), B=>B(3), cout=>c(3), s=>S(3));
    Cout <= c(3);
end Somador;

```

Somador Comportamental de 4 Bits (Entidade)



- A mesma entidade do somador comportamental. Porque?
 - Portas de entrada e saída são iguais

```
entity Adder is
  port
  (
    A, B: in std_logic_vector(3 downto 0);
    cout: out std_logic;
    S : out std_logic_vector(3 downto 0)
  );
end Adder;
```

Somador Comportamental de 4 Bits (Arquitetura)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
```

Nova bibliotecas para
implementar somas

```
entity Adder is
    ...
end Adder;
```

Vetores de **5 bits** para implementar o carry

```
architecture SomaComp of Adder is
    signal iA, iB, SOMA: std_logic_vector(4 downto 0);
begin
```

```
    iA <= '0' & A;
    iB <= '0' & B;
```

Concatena bit mais significativo com 0

```
    SOMA <= iA + iB;
```

Soma de 5 bits

```
    S <= SOMA(3 downto 0);
```

Apenas 4 bits são atribuídos para a saída

```
    carry <= SOMA(4);
```

Último bit da soma é o carry

```
end SomaComp;
```

Exercícios

1. Descrevendo o qualificador de overflow de forma comportamental...

```
V <= '1' when (iA(3) xnor iB(3))='1' and (SOMA(3) xor i(3))='1'
      else
        '0';
```

- A descrição está completa? Teste as possibilidades
- Faça os demais qualificadores (Negativo e Zero), **de forma comportamental**

2. Faça um test-bench e teste todas as possibilidades e verifique que a implementação acima atende o desejado
3. Faça agora um subtrator, utilizando a descrição estrutural. DICA: use o carry de entrada do circuito. Note que, em complemento de 2, $-Y = \text{NOT}(Y) + 1$
4. Faça agora um subtrator e veja se o resultado obtido com overflow é o esperado. Caso não seja, o que deve ser feito para recalcular o overflow?