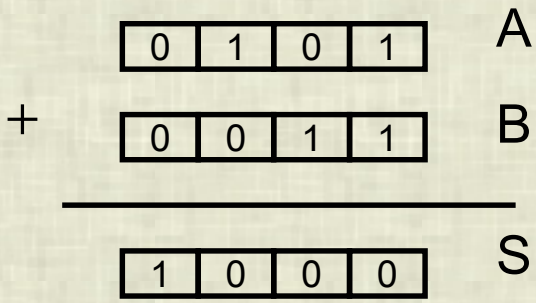
FUNDAMENTOS DE SISTEMAS DIGITAIS

# Projeto de Somador com e sem Sinal

**prof. Dr. Edson Ifarraguirre Moreno**

# Planejando a Descrição de um Somador

- **Como descrever uma soma?**
  - S <= A + B;
- **Como esta soma pode ser realizada em hardware?**
  - Dividir as entradas e saídas em vetores de bits
- **Exemplo:**
  - A e B são vetores de 4 bits
  - A contém 5 e B contém 3




  - S igual a 8

# Dividindo para Conquistar (Somador de 1 bit)

- **O que fazer com cada bit?**
  - Descrever em hardware uma função que implementa a operação de soma deste bit
- **Qual é a função?**
  - Supondo a soma de dois bits, para cada par de bit somado existem duas saídas: o vai um (carry) e o resultado da soma
- **Como implementar a função?**
  - Por exemplo uma tabela verdade

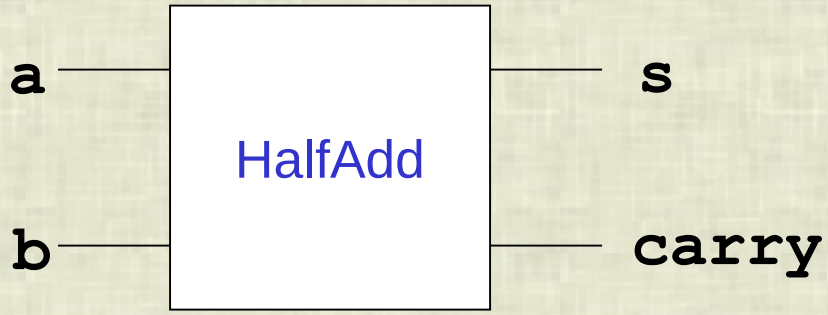
| a | b | s (soma) | carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# Dividindo para Conquistar (Somador de 1 bit)

- **Qual é o próximo passo?**
  - Extrair as funções da  tabela verdade
    - s = a xor b
    - carry = a and b
- **E agora?**
  - Descrever as funções em Hardware
- **Qual recurso utilizar?**
  - Linguagem VHDL

# Dividindo para Conquistar (Somador de 1 bit)

- **INTERFACE EXTERNA: entity**
  - Especifica somente a interface
  - Não contém definição do comportamento




```
entity HalfAdd is
      port
       (
            a, b: in std_logic;
            s, carry: out std_logic
       );
end HalfAdd;
```

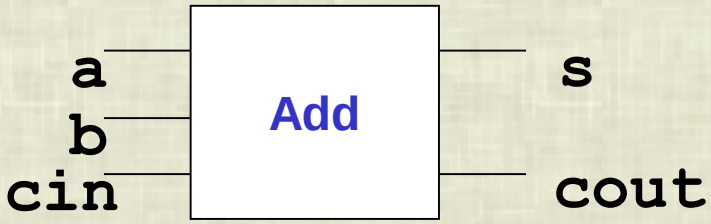
# Dividindo para Conquistar (Somador de 1 bit)

- COMPORTAMENTO : architecture
  - Especifica o comportamento da *entity*
  - Deve ser associada a uma *entity* específica
  - Uma *entity* pode ter associada várias *architectures* (diferentes formas de implementar um mesmo módulo)

```
architecture HA of HalfAdd is
begin
   s <= a xor b;
   carry <= a and b;
end HA;
```

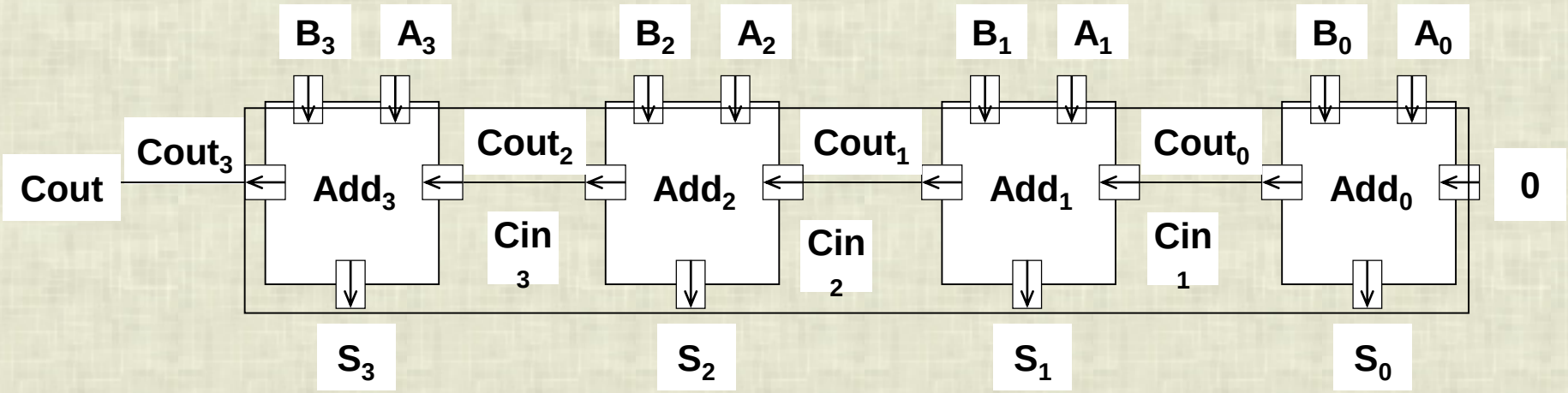
# Somador de 1 bit Completo

- A arquitetura HD, apresentada é suficiente para descrever uma soma de um estágio?
  - Não, falta considerar o vai um do estágio anterior
- **Exercício:**
  - Fazer um novo par entidade-arquitetura que implementa uma soma completa. Fazer a tabela verdade e a entidade e a arquitetura.
  - Chame esta entidade de **Add** (será usada mais adiante)




- Como fazer agora para conseguir implementar todo o vetor?
  - Uma possibilidade é implementar vários módulos de 1 bit em um par entidade-arquitetura

# Somador Completo de 4 Bits




```
entity Adder4Bits is
      port
       (
            A, B: in std_logic_vector(3 downto 0);
            cout: out std_logic;
            S : out std_logic_vector(3 downto 0)
      );
end Adder4Bits;
```

# Somador Completo de 4 Bits

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

architecture Somador of Adder4Bits 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;
```

- **Perguntas e exercícios:**
  - A descrição acima é estrutural? Porque?
  - Para que serve o cout do somador de 4 bits, já que não há mais estágios
  - Faça um somador de 8 bits, tendo como base o somador de 4 Bits descrito acima

# Exercícios

1. Implementar um somador de 8 bits utilizando um somador de 4 bits

2. Implementar um testbench para cada um dos somadores

3. Implementar o somador de 8 bits empregando o comando *FOR GENERATE*