UNIVERSIDADE FEDERAL DE SANTA CATARINA

Projeto Hierárquico

EEL5105 – Circuitos e Técnicas Digitais

Objetivos

- Entender o conceito de **Projeto Hierárquico**.
- Implementar **projetos** em **VHDL** usando **component** e **port map**.

Introdução ao Projeto Hierárquico

Tarefas

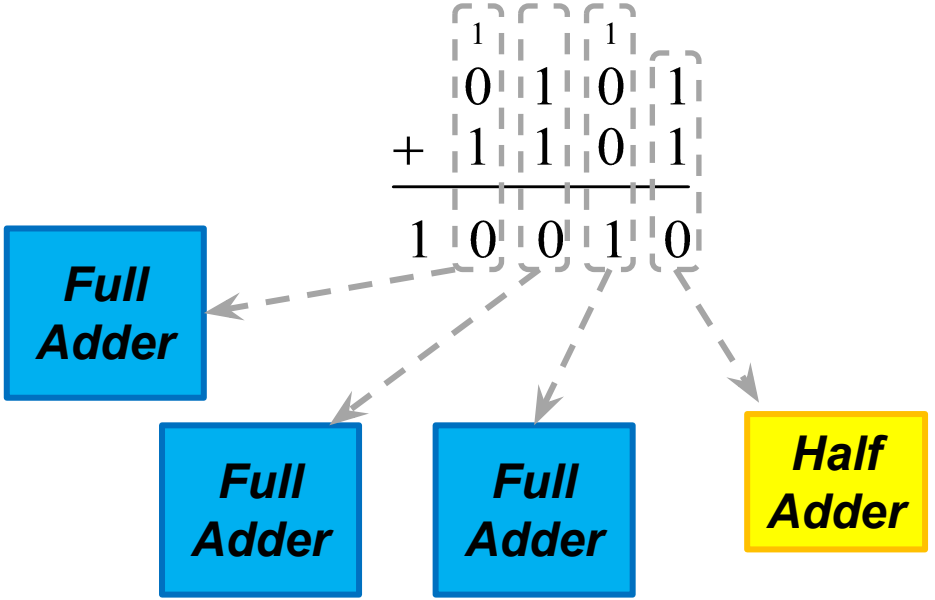
Tarefas Avançadas

Introdução ao Projeto Hierárquico

- **Projeto Hierárquico**
  - Abordagem de projeto usada não somente em **VHDL**.
  - **Idéia**: compartimentalizar o projeto em múltiplos **componentes** que podem ser **criados separadamente** e depois **integrados** e **reutilizados**.
  - Facilita a **leitura**, **entendimento** e **manutenção** do código.

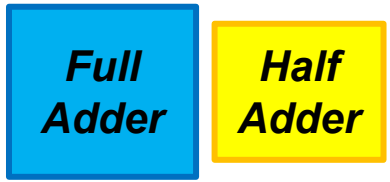
Introdução

- **Exemplo 1**: somador de números de **4 bits** construído usando um ***half adder*** e ***full adders***

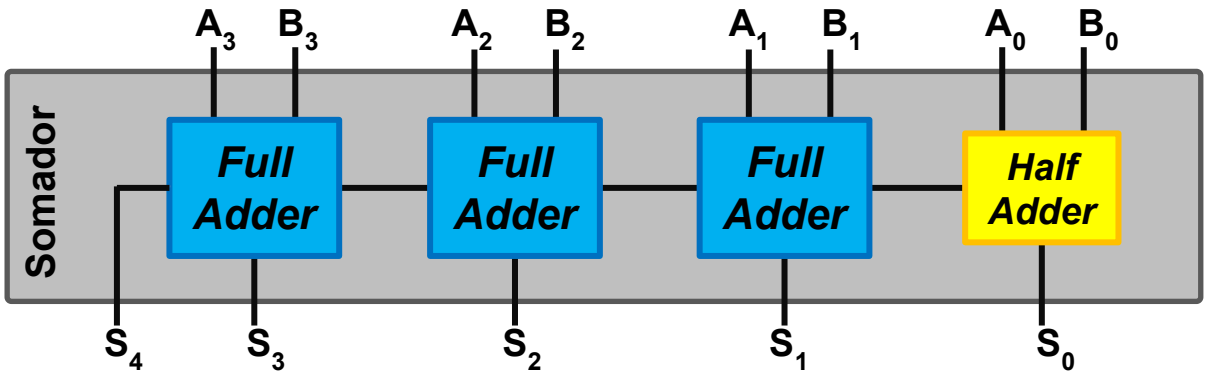


Introdução

- **Exemplo 1**: somador de números de **4 bits** construído usando um ***half adder*** e ***full adders***
  - **Componentes** internos são primeiramente projetados:

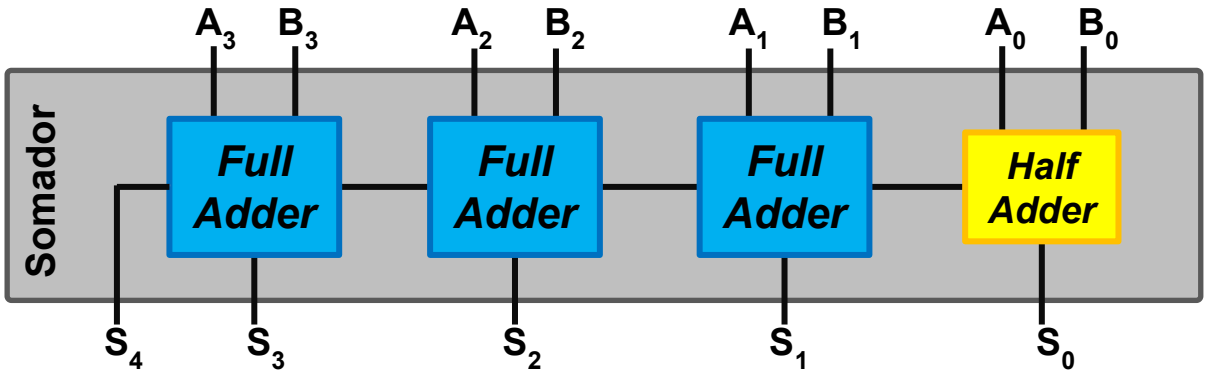


  - Em seguida, são **integrados** para construir o somador desejado:



Introdução

- **Exemplo 1**: somador de números de **4 bits** construído usando um ***half adder*** e ***full adders***
  - **Circuito**:



Hierarquia:

somador.vhd
- fulladder.vhd
- halfadder.vhd

Introdução

- **Exemplo 1**: Componentes internos são primeiramente projetados como na aula anterior:

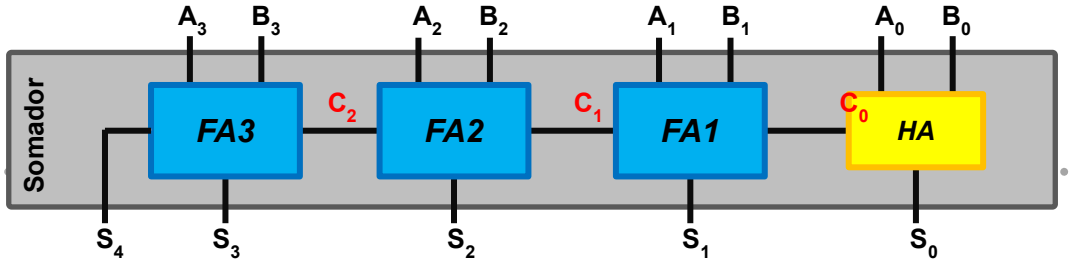
fulladder.vhd

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity fulladder is
port (A: in std_logic;
      B: in std_logic;
      Cin: in std_logic;
      S: out std_logic;
      Cout: out std_logic
      );
end fulladder;

architecture ...
```

halfadder.vhd

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity halfadder is
port (A: in std_logic;
      B: in std_logic;
      S: out std_logic;
      Cout: out std_logic
      );
end halfadder;

architecture ...
```

Introdução

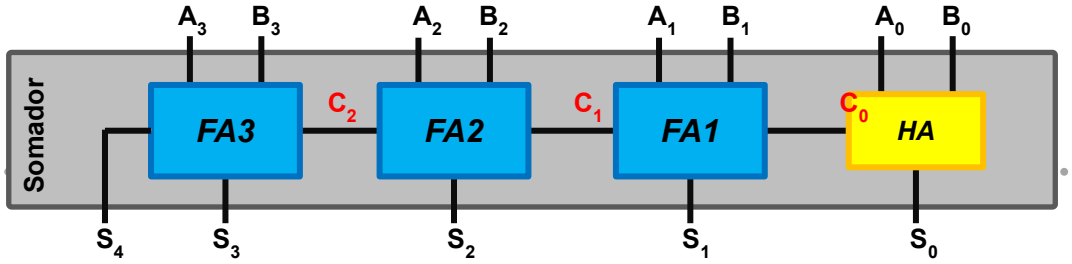


- **Exemplo 1**
  (forma **posicional** para **port map**):

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity somador is
port (A,B: in std_logic_vector(3 downto 0);
      S: out std_logic_vector(4 downto 0)
     );
end somador;

architecture soma4 of somador is
 signal C0,C1,C2: std_logic;
 component halfadder is
  port (A: in std_logic;
        B: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;
```

```
component fulladder is
  port (A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;

begin
 HA: halfadder port map (A(0),B(0),S(0),C0);
 FA1: fulladder port map (A(1),B(1),C0,S(1),C1);
 FA2: fulladder port map (A(2),B(2),C1,S(2),C2);
 FA3: fulladder port map (A(3),B(3),C2,S(3),S(4));
end soma4;
```

Introdução



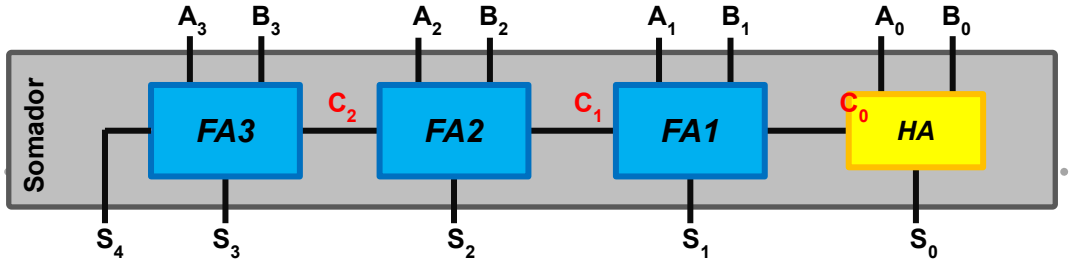
- **Exemplo 1**
  (forma **posicional** para **port map**):

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity som
port (A,B:
      S: o
);
end somador;

architecture soma4 of somador is

 signal C0,C1,C2: std_logic;

component halfadder is
 port (A: in std_logic;
       B: in std_logic;
       S: out std_logic;
       Cout: out std_logic);
end component;
```

Declarações dos componentes já existentes (de outros arquivos)

```
component fulladder is
  port (A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
end component;

begin
 HA: halfadder port map (A(0),B(0),S(0),C0);
 FA1: fulladder port map (A(1),B(1),C0,S(1),C1);
 FA2: fulladder port map (A(2),B(2),C1,S(2),C2);
 FA3: fulladder port map (A(3),B(3),C2,S(3),S(4));
end soma4;
```

Introdução



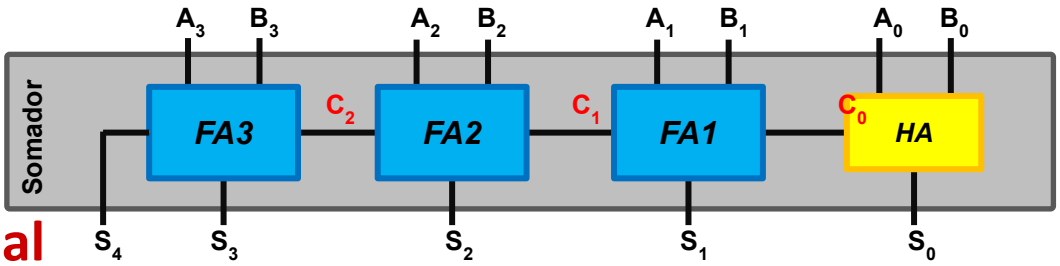
- **Exemplo 1**
  (forma **posicional** para **port map**):

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity somador is
port (A,B: in std_logic_vector(3 downto 0);
      S: out                    4 downto 0)
     );
end somador;

architecture soma4 of somador is
 signal C0,C1,C2: std_logic;
 component halfadder is
  port (A: in std_logic;
        B: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;
```

Descrição das conexões dos componentes

```
component fulladder is
  port (A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;

begin
 HA: halfadder port map (A(0),B(0),S(0),C0);
 FA1: fulladder port map (A(1),B(1),C0,S(1),C1);
 FA2: fulladder port map (A(2),B(2),C1,S(2),C2);
 FA3: fulladder port map (A(3),B(3),C2,S(3),S(4));
end soma4;
```

Introdução

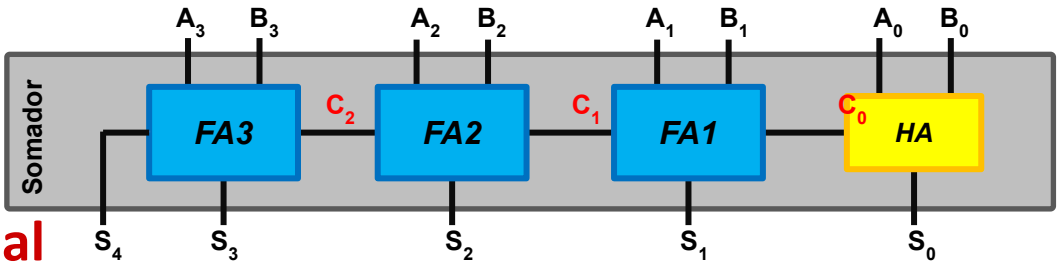


- **Exemplo 1**: port map **nominal**

```
library IEEE;
use IEEE.Std_Logic_1164.all;
entity somador is
port (A,B: in std_logic_vector(3 downto 0);
      S: out std_logic_vector(4 downto 0));
end somador;
architecture soma4 of somador is
 signal C0,C1,C2: std_logic;
 component halfadder is
  port (A: in std_logic;
        B: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;
 component fulladder is
  port (A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;
```

```
begin
 HA: halfadder port map (A => A(0),
                         B => B(0),
                         S => S(0),
                         Cout => C0);
 FA1: fulladder port map (A => A(1),
                          B => B(1),
                          Cin => C0,
                          S => S(1),
                          Cout => C1);
 FA2: fulladder port map (A => A(2),
                          B => B(2),
                          Cin => C1,
                          S => S(2),
                          Cout => C2);
 FA3: fulladder port map (A => A(3),
                          B => B(3),
                          Cin => C2,
                          S => S(3),
                          Cout => S(4));
end soma4;
```

Introdução

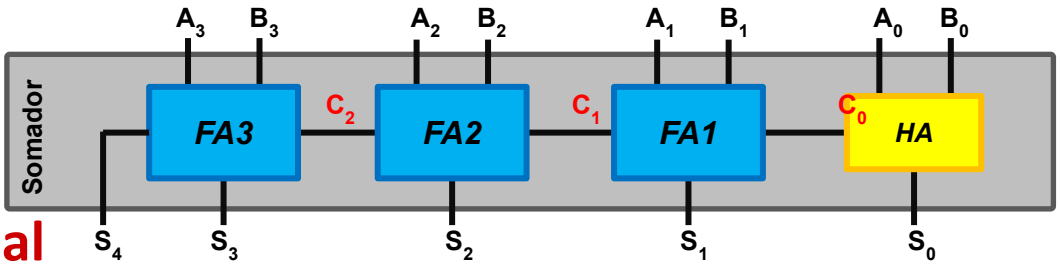


- **Exemplo 1**: port map **nominal**

```
library IEEE;
use IEEE.Std_Logic_1164.all;
entity somador is
port (A,B: in std_logic_vector(3 downto 0);
      S: out std_logic_vector(4 downto 0));
end somador;
architecture soma4 of somador is
 signal C0,C1,C2: std_logic;

 component halfadder is
  port (A: in std_logic;
        B: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;

 component fulladder is
  port (A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;

begin
 HA: halfadder port map (A => A(0),
                         B => B(0),
                         S => S(0),
                         Cout => C0);

 FA1: fulladder port map (A => A(1),
                          B => B(1),
                          Cin => C0,
                          => S(1),
                          ut => C1);
 FA2: f                   => A(2),
                          => B(2),
                          n => C1,
                          S => S(2),
                          Cout => C2);

 FA3: fulladder port map (A => A(3),
                          B => B(3),
                          Cin => C2,
                          S => S(3),
                          Cout => S(4));

end soma4;
```

Declarações dos componentes já existentes (de outros arquivos)

Introdução



- **Exemplo 1**: port map **nominal**

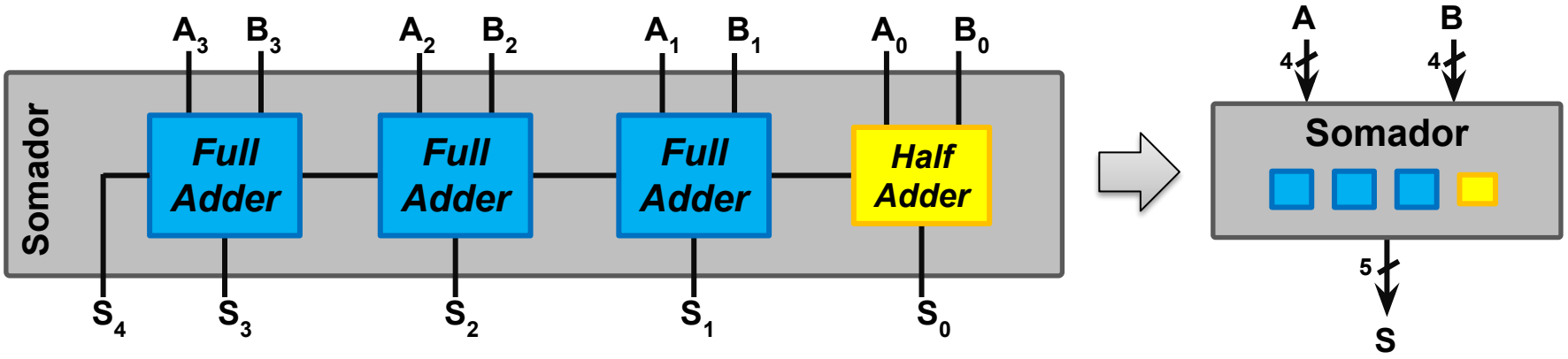
```
library IEEE;
use IEEE.Std_Logic_1164.all;
entity somador is
port (A,B: in std_logic_vector(3 downto 0);
      S: out std_logic_vector(4 downto 0));
end somador;
architecture
 signal C0,C
 component ha
  port (A: in
        B: in
        S: ou
        Cout:
 end componen
 component fulladder is
  port (A: in std_logic;
        B: in std_logic;
        Cin: in std_logic;
        S: out std_logic;
        Cout: out std_logic);
 end component;
```

Descrição das conexões dos componentes (note que múltiplas instâncias de um mesmo componente podem ser utilizadas)

```
begin
HA: halfadder port map (A => A(0),
                        B => B(0),
                        S => S(0),
                        Cout => C0);
FA1: fulladder port map (A => A(1),
                         B => B(1),
                         Cin => C0,
                         S => S(1),
                         Cout => C1);
FA2: fulladder port map (A => A(2),
                         B => B(2),
                         Cin => C1,
                         S => S(2),
                         Cout => C2);
FA3: fulladder port map (A => A(3),
                         B => B(3),
                         Cin => C2,
                         S => S(3),
                         Cout => S(4));

end soma4;
```

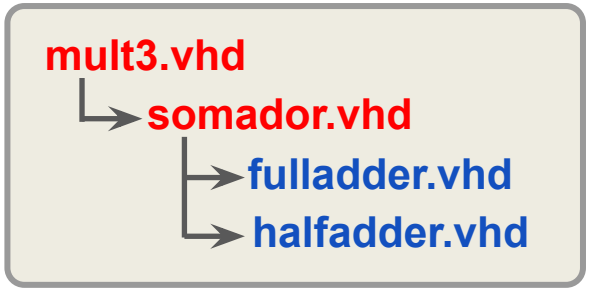
Introdução

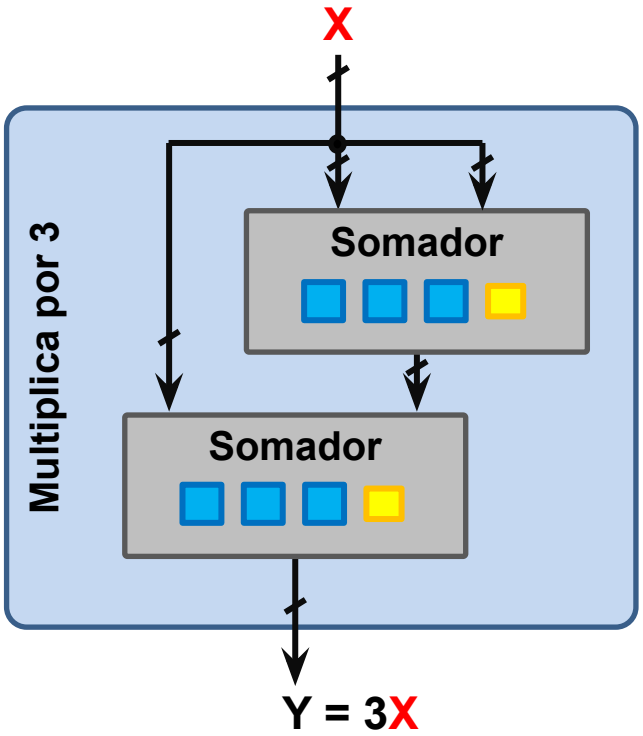
- **Exemplo 2**: multiplicador por 3 construído usando somadores, que por sua vez foram construídos com ***half adder*** e ***full adders***



Introdução

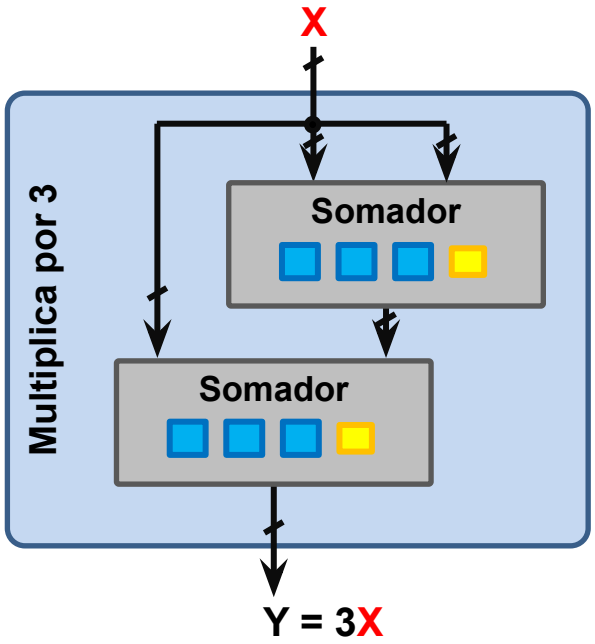
- **Exemplo 2**: multiplicador por 3 construído usando somadores, que por sua vez foram construídos com ***half adder*** e ***full adders***.





Introdução

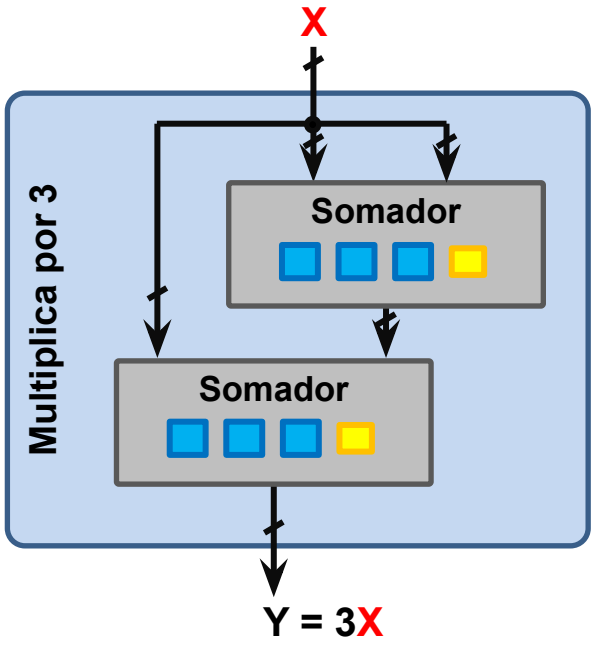
- **Exemplo 2**:
  Port map **sequencial**




```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity mult3 is
port (X: in std_logic_vector(3 downto 0);
      Y: out std_logic_vector(4 downto 0));
end mult3;

architecture mult3arch of mult3 is

 signal S: std_logic_vector(4 downto 0);

 component somador is
  port (A,B: in std_logic_vector(3 downto 0);
        S: out std_logic_vector(4 downto 0) );
 end component;

begin

 SUM1: somador port map (X, X, S);

 SUM2: somador port map (X, S(3 downto 0), Y);

end mult3arch;
```

Introdução

- **Exemplo 2**:
  Port map **nominal**




```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity mult3 is
port (X: in std_logic_vector(3 downto 0);
      Y: out std_logic_vector(4 downto 0) );
end mult3;

architecture mult3arch of mult3 is

 signal S: std_logic_vector(4 downto 0);

 component somador is
  port (A,B: in std_logic_vector(3 downto 0);
        S: out std_logic_vector(4 downto 0) );
 end component;

begin

 SUM1: somador port map (A => X,
                         B => X,
                         S => S);

 SUM2: somador port map (A => X,
                         B => S(3 downto 0),
                         S => Y);

end mult3arch;
```

Introdução ao Projeto Hierárquico

Tarefas

Tarefas Avançadas

Tarefas

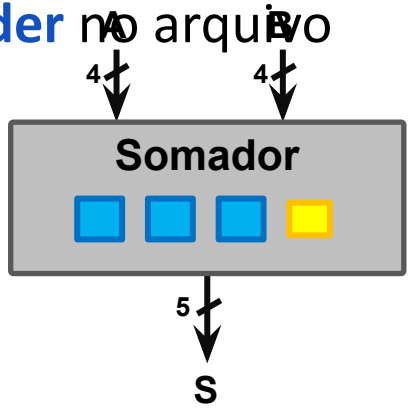
Observação importante: mapeamento de **ports** para **SW/LEDR/KEY** precisa ser feito apenas para a **top level entity** (arquivo principal do projeto). Ou seja, é preciso:

- Conectar **ports** da **top level entity** a **SW/LEDR/KEY** usando o **Mapper**;

  ou
- Nomear as **ports** da **top level entity** como **SW/LEDR/KEY**.

Tarefa 1

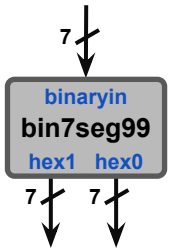
- Faça a implementação do **Somador de 4 Bits** mostrado anteriormente. Para tal, siga os seguintes passos:
  - Faça a implementação do **half adder** no arquivo **halfadder.vhd**.
  - Similarmente, faça a implementação do **full adder** no arquivo **fulladder.vhd**.
  - Faça a implementação do **somador de 4 bits** no arquivo **somador.vhd**.
  - Finalmente, faça o mapeamento de portas e a emulação do seu circuito.



Tarefa 2

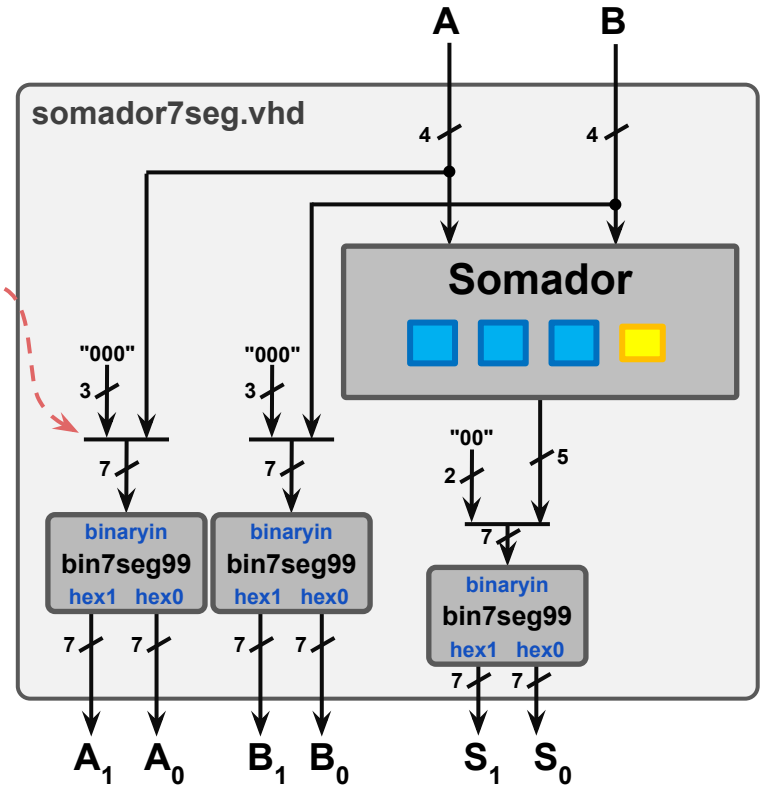
- Implementar um **Somador de 4 Bits** com entradas e saídas apresentadas em displays de 7 segmentos, usando o componente **bin7seg99** disponível no Moodle.
  - **bin7seg99**: recebe entradas de 7 bits e converte o valor binário dessas entradas para displays de 7 segmentos, funcionando para valores de entrada de **0** a **99**.

```
entity bin7seg99 is
    port (
        binaryin: in std_logic_vector (6 downto 0);
        hex1, hex0: out std_logic_vector (6 downto 0)
    );
end bin7seg99;
```



Tarefa 2

- Somador de palavras de **4 bits** com saída para **display de 7 segmentos**:
  - **Dica**: use ***signals*** para fazer as **concatenações**.
    Exemplo: `concatA <= "000" & A;`
  - Utilize o seguinte mapeamento (pode ser via ***Mapper***):
    - A_1 para **HEX5**;
    - A_0 para **HEX4**;
    - B_1 para **HEX3**;
    - B_0 para **HEX2**;
    - S_1 para **HEX1**;
    - S_0 para **HEX0**;
    - **A** para **SW(7 downto 4)**;
    - **B** para **SW(3 downto 0)**.

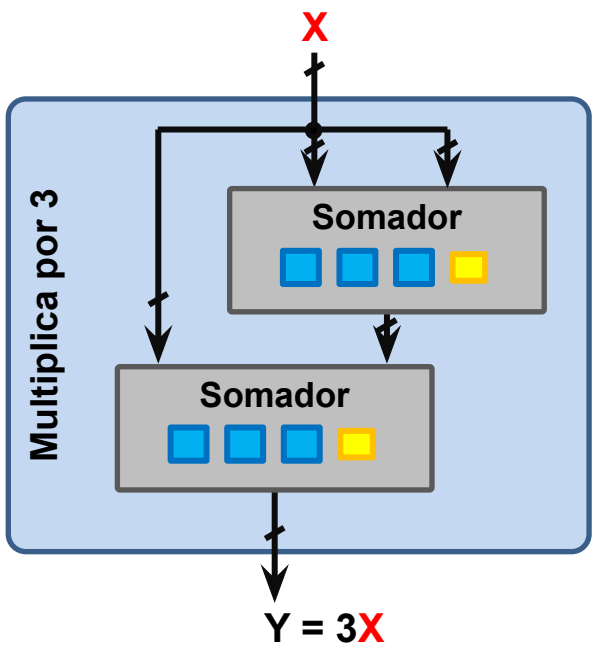


Tarefa 2

- Como nos laboratórios anteriores, vamos colocar agora o **somador4bits** para funcionar no **DE1-SoC** usando o **Quartus II**.
- Para tal:
  - Configure o **somador4bits** como **Top Level** no **Emulador Web**.
  - Exporte o projeto para o **DE1-SoC/Quartus II**.
  - Abra o projeto exportado no **Quartus II**:
    - **File > Open project…**
    - **Processing > Start Compilation…**
  - Faça a gravação no **DE1-SoC**.
    - Ver slides das **aulas anteriores**.

Tarefa 3

- Faça a implementação do **Multiplicador por 3** discutido anteriormente seguindo passos similares aos usados para implementar o **Somador de 4 Bits**.



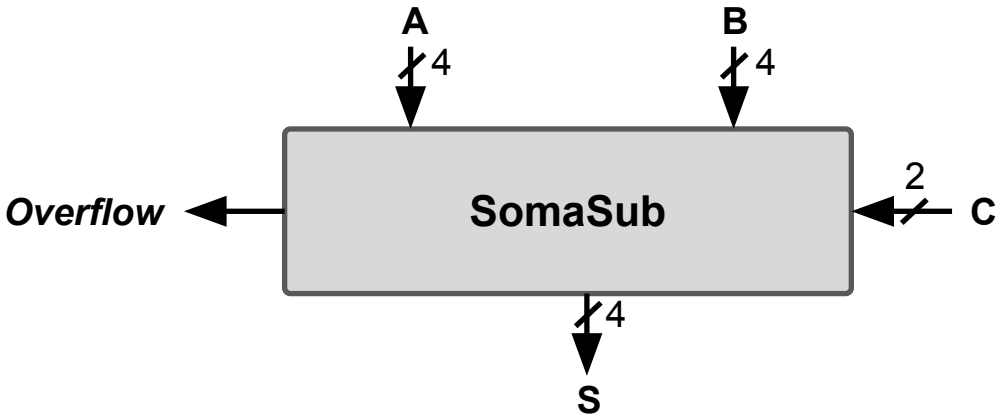
Introdução ao Projeto Hierárquico
Tarefas

Tarefas Avançadas

Tarefa Avançada 1

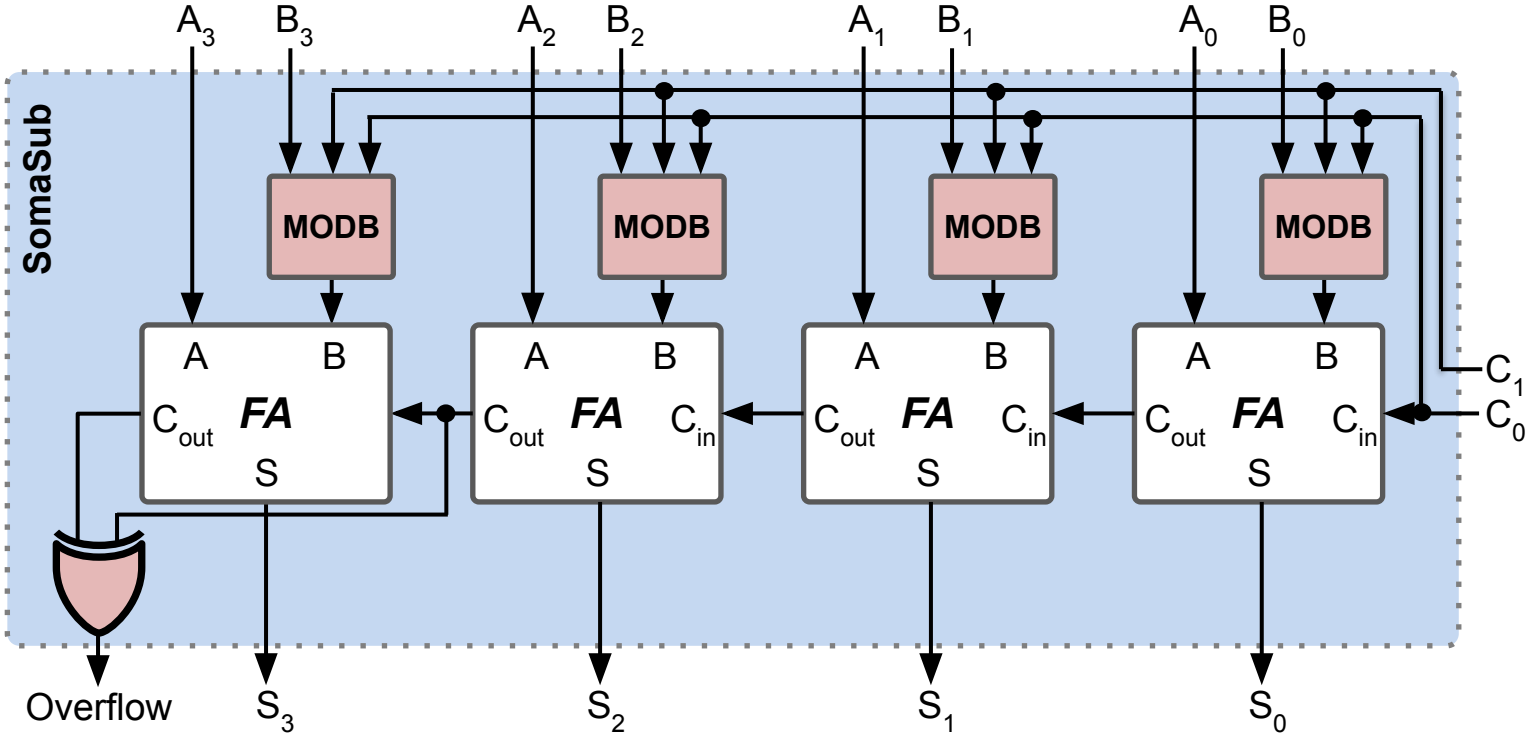
- Implementar um **circuito somador/subtrator de 4 bits** capaz de realizar **soma** ou **subtração** com dois operandos, ou ainda **incremento** e **decremento** de um dos operandos.

C	Operação
0 0	A + B
0 1	A + 1
1 0	A – 1
1 1	A – B



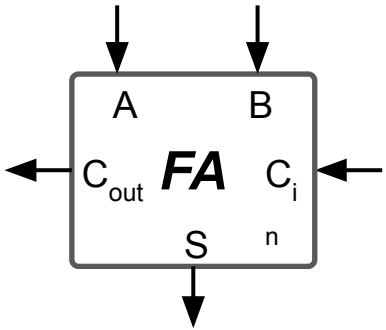
Tarefa Avançada 1

- **Somador/subtrator** a ser implementado:

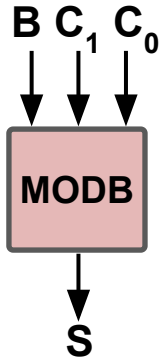


Tarefa Avançada 1

- Implementação do **Somador/subtrator**
  - Primeiramente implementar **componentes** básicos:



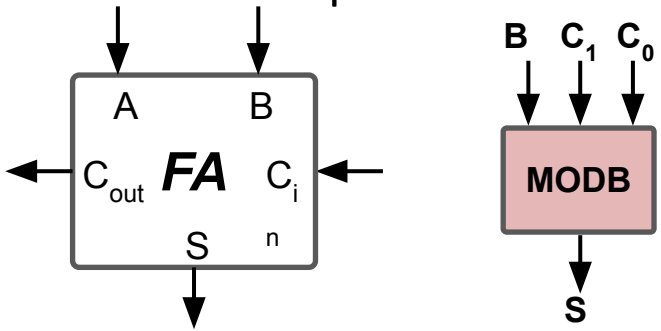
A B C_{in}	S	C_{out}
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1



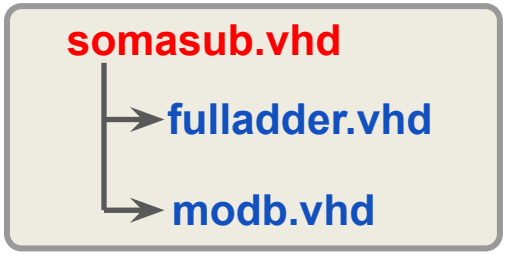
C_1 C_0 B	S
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	0

Tarefa Avançada 1

- Implementação do **Somador/subtrator**
  - Primeiramente implementar **componentes** básicos:



  - Em seguida, integrá-los em um arquivo **somasub.vhd**.
  - Finalmente, conectar com as chaves e leds do kit.



Tarefa Avançada 2

- Implemente um **somador de números de 8 bits** usando um **half adder** e **full adders**. Para tal, pesquise sobre a estrutura **for generate** disponível em **VHDL** e use-a para fazer essa implementação.