



**Universidade Federal de Santa Catarina**  
**Centro Tecnológico**  
Departamento de Informática e Estatística  
Curso de Graduação em Ciências da Computação



# **Sistemas Digitais**

**INE 5406**

## **Aula 13-P**

**Descrição em VHDL, síntese e simulação de um sistema digital completo (BO + BC).**

**Prof. José Luís Güntzel**  
**[guntzel@inf.ufsc.br](mailto:guntzel@inf.ufsc.br)**

**[www.inf.ufsc.br/~guntzel/ine5406/ine5406.html](http://www.inf.ufsc.br/~guntzel/ine5406/ine5406.html)**

# Sistema Digital Completo (BO + BC)

---

## ▶ Experimento 1: Setup Experimental

1. Na pasta Meus\_documentos, criar uma pasta com o seu nome (p. ex., “Paulo”). Na pasta “Paulo”, criar uma pasta com nome de “multiplicador”. Dentro da pasta “multiplicador”, criar uma pasta com nome “BO”
2. Acessar o sítio “[www.inf.ufsc.br/~guntzel/ine5406/aula13P/acompletar/BO](http://www.inf.ufsc.br/~guntzel/ine5406/aula13P/acompletar/BO)” e baixar os arquivos VHDL para a pasta “BO” recém criada.
3. Abrir o Quartus II e criar um projeto na pasta “BO”, selecionando “bo.vhd” como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
4. Prestar atenção nas explicações que seguirão nos próximos slides

# Sistema Digital Completo (BO + BC)

---

## ▶ Experimento 1: Setup Experimental (cont.)

5. Completar o arquivo “bo.vhd”. Obs: no BO, o comprimento dos dados será  $n=4$ .
6. Compilar o projeto criado.
7. A partir do Quartus II, chame o ModelSim-Altera e inicie uma simulação com atrasos (“Gate-level Simulation”).
8. Criar um arquivo de estímulos, nomeando-o “estimulosbo.do”. Preparar os estímulos para simular a operação  $3 \times 4$  (i.e.,  $A=3$  e  $B=4$ ).

# Sistema Digital Completo (BO + BC)

---

## ▶ Experimento 2: Setup Experimental

1. Dentro da pasta “multiplicador”, criar uma pasta com nome “BC”
2. Acessar o sítio “[www.inf.ufsc.br/~guntzel/ine5406/aula13P/acompletar/BC](http://www.inf.ufsc.br/~guntzel/ine5406/aula13P/acompletar/BC)” e baixar o arquivo “bc.vhd” para a pasta “BC” recém criada.
3. Abrir o Quartus II e criar um projeto na pasta “BC”, selecionando “bc.vhd” como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
4. Prestar atenção nas explicações que seguirão nos próximos slides

# Sistema Digital Completo (BO + BC)

---

## ▶ Experimento 2: Setup Experimental (cont.)

5. Completar o arquivo “bc.vhd”.
6. Compilar o projeto criado.
7. A partir do Quartus II, chame o ModelSim-Altera e inicie uma simulação com atrasos (“Gate-level Simulation”).
8. Criar um arquivo de estímulos, nomeando-o “estimulosbc.do”. Preparar os estímulos para simular a operação 3 x 4 (i.e., A=3 e B=4).

# Sistema Digital Completo (BO + BC)

---

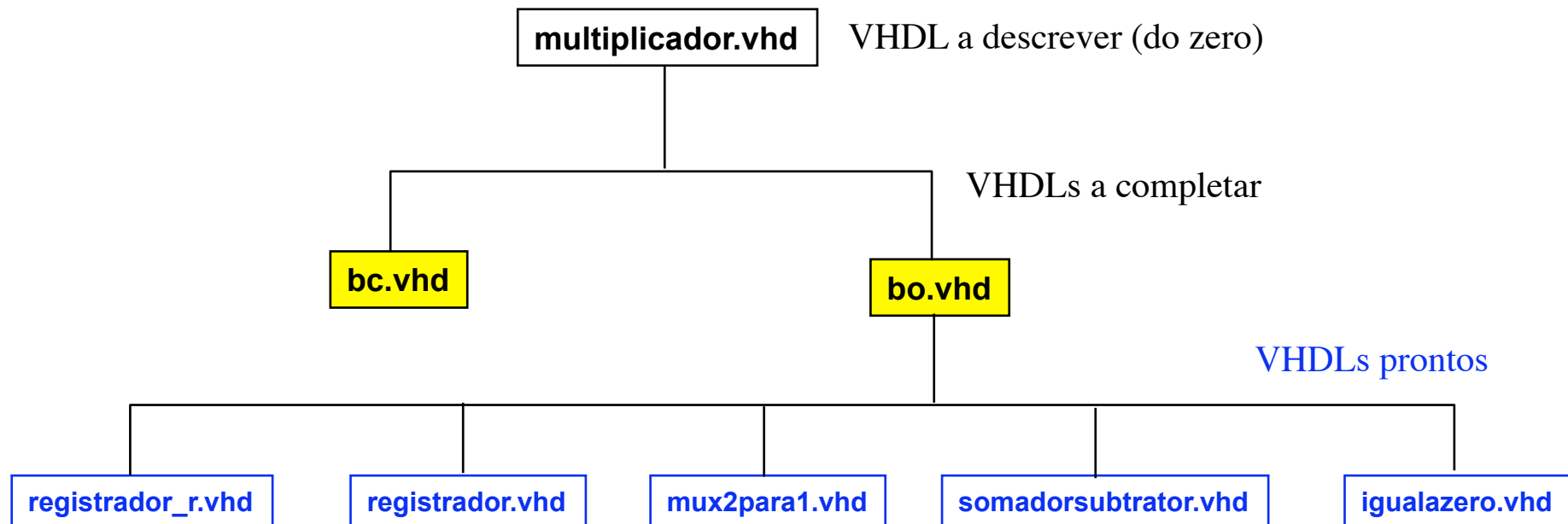
## ▶ Experimento 3: Setup Experimental

- 1 Na pasta “multiplicador”, criar um arquivo VHDL chamado “multiplicador.vhd”. Este arquivo deve juntar o BO e o BC já simulados. Ainda na pasta “multiplicador”, criar um projeto “multiplicador”, escolhendo “multiplicador.vhd” como toplevel. Escolher o dispositivo FPGA EP2C35F672C6 e selecionar o ModelSim-Altera como EDA Simulation Tool.
- 2 Compilar o projeto criado.
- 3 A partir do Quartus II, chame o ModelSim-Altera e inicie uma simulação com atrasos (“Gate-level Simulation”).
- 4 Criar um arquivo de estímulos, nomeando-o “estimulos.do”. Preparar os estímulos para simular a operação 3 x 4 (i.e., A=3 e B=4).

# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL (incompleto) do BO do Multiplicador

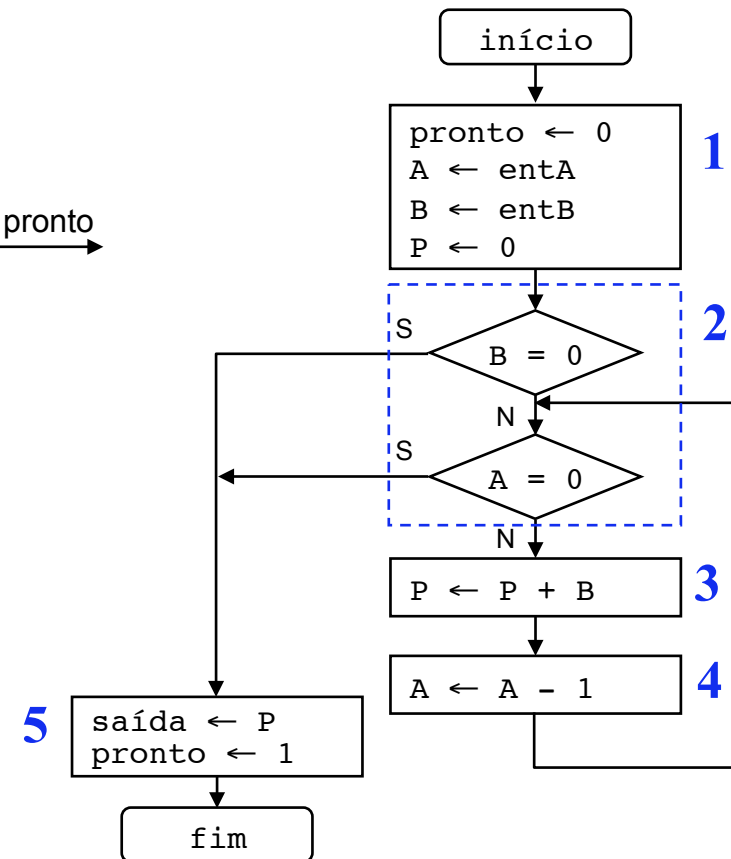
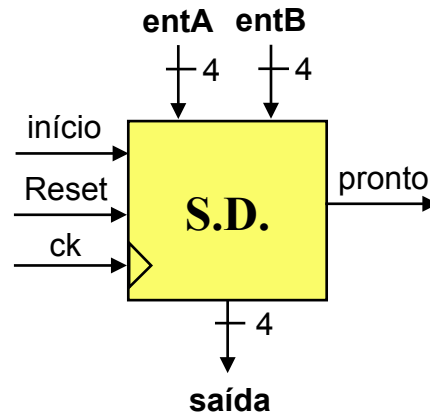
### Estrutura de Arquivos



# Sistema Digital Completo (BO + BC)

## ► Especificações: Comportamento e Interfaces

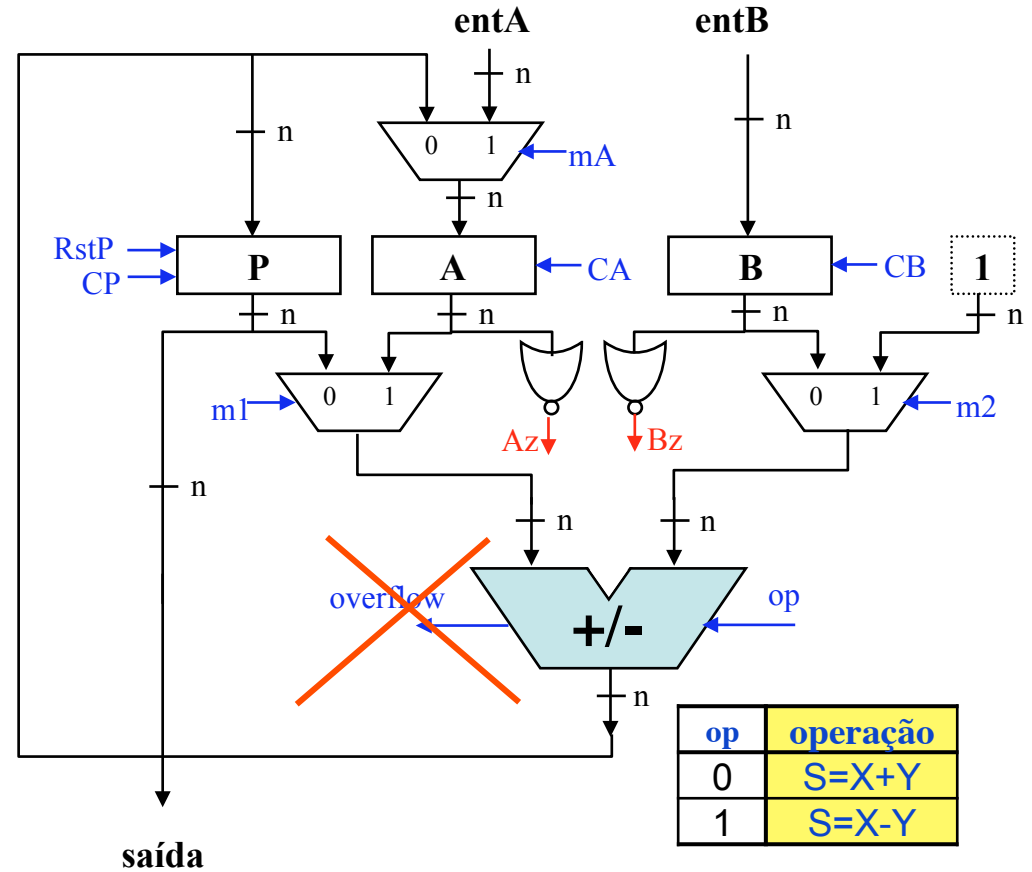
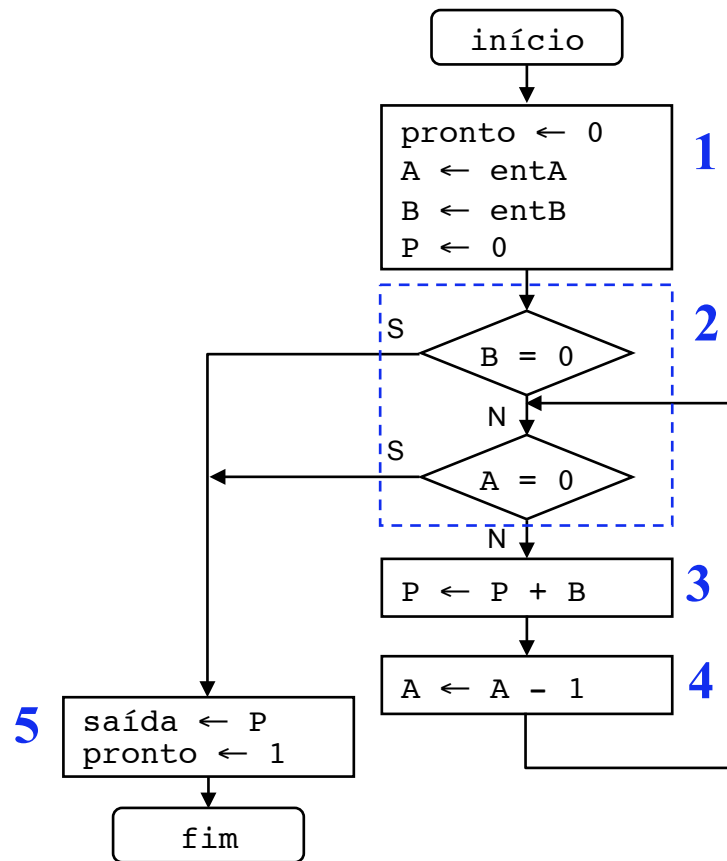
```
início
pronto ← 0;
A ← entA;
B ← entB;
P ← 0;
Se B ≠ 0 então
Enquanto A ≠ 0 faça
    início
        P ← P + B;
        A ← A - 1;
    fim
saída ← P;
pronto ← 1;
fim
```





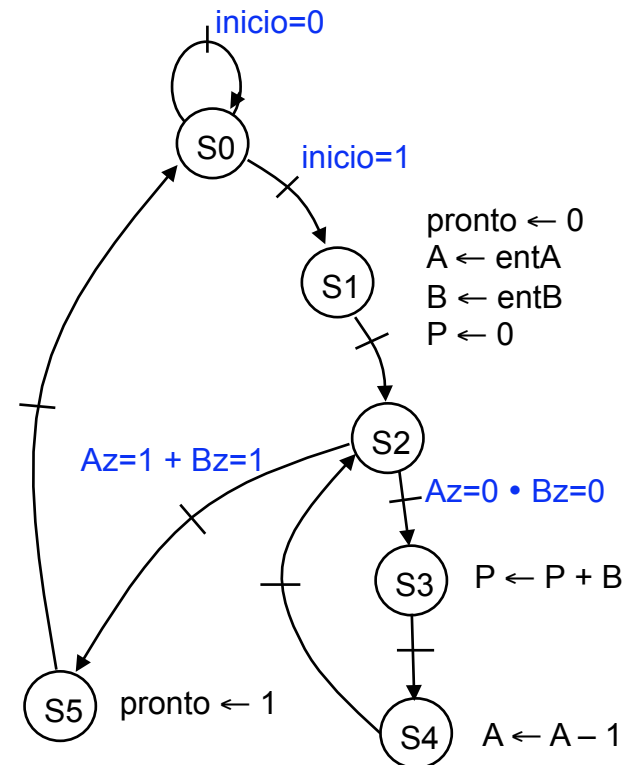
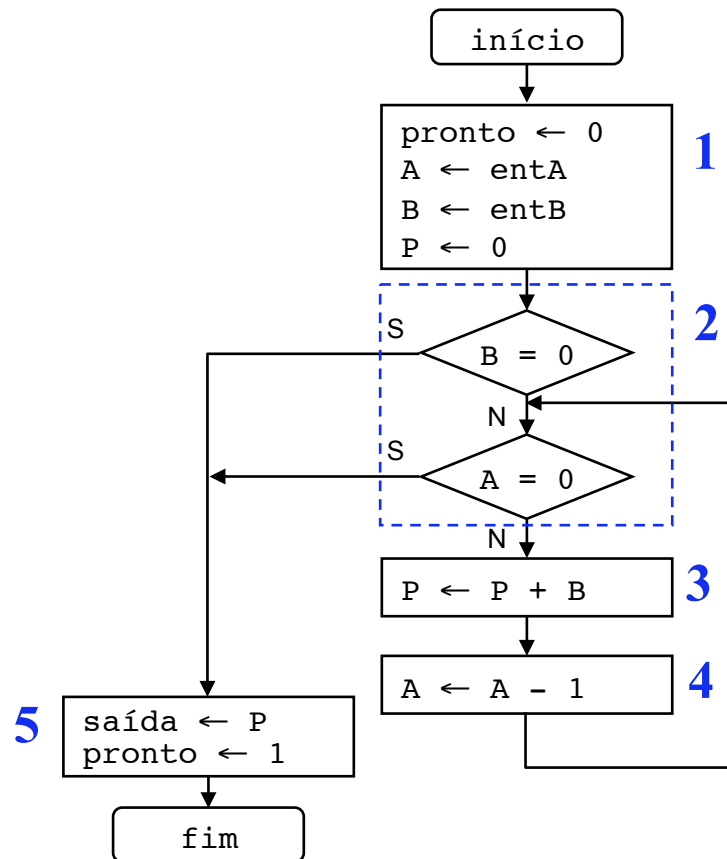
# Sistema Digital Completo (BO + BC)

## ▶ Algoritmo e Bloco Operativo Visando Custo Mínimo



# Sistema Digital Completo (BO + BC)

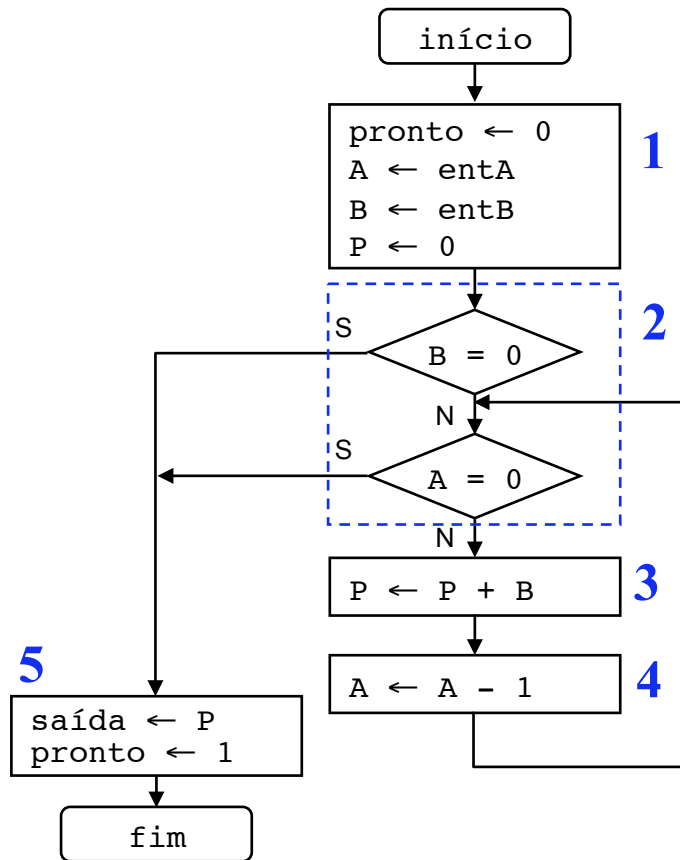
## ► Bloco de Controle Visando Custo Mínimo (Moore) Diagrama de Estados e FSMD



# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

### Tabela de Saídas (Assumindo Moore)



Estado	Reg. P		Reg. A			Somador/Sub			Saída	
	RstP	CP	mA	CA	CB	m1	m2	op	lib	pronto
S0	0	0	-	0	0	-	-	-	0	-
S1	1	0	1	1	1	-	-	-	0	0
S2	0	0	-	0	0	-	-	-	0	0
S3	0	1	-	0	0	0	0	0	0	0
S4	0	0	0	1	0	1	1	1	0	0
S5	0	0	-	0	0	-	-	-	1	1

1 sinal

1 sinal

1 sinal

RstP = mA = cB

CP

CA = op = m1 = m2

lib = pronto

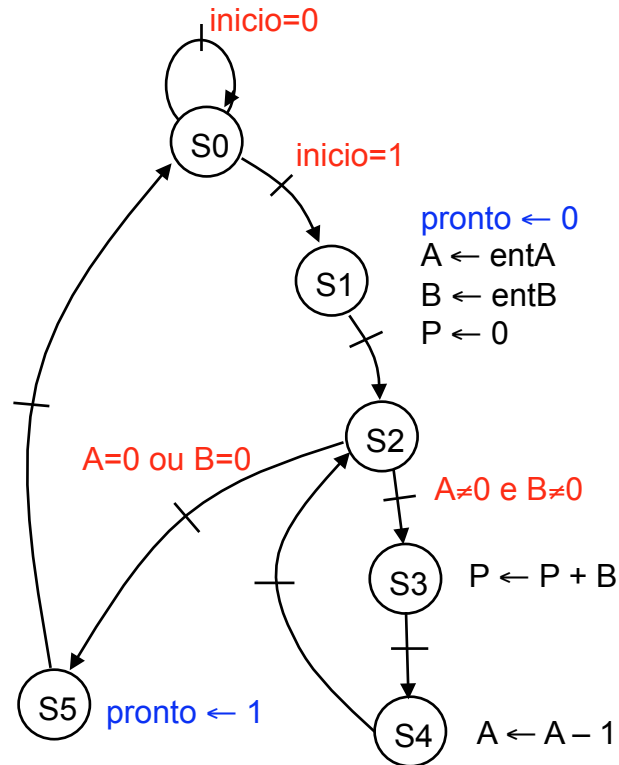
4 sinais

# Sistema Digital Completo (BO + BC)

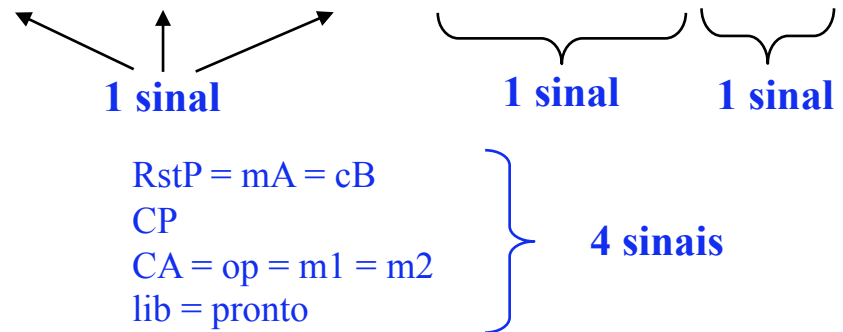
## ► Bloco de Controle Visando Custo Mínimo

### Tabela de Saídas e FSMD

### Solução da aula teórica



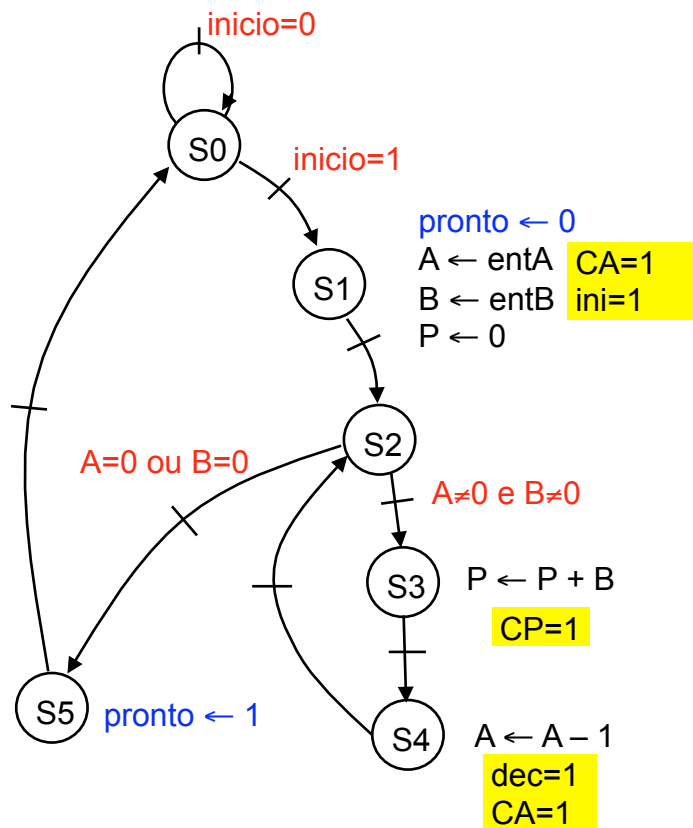
Estado	Reg. P		Reg. A			Somador/Sub			Saída
	RstP	CP	mA	CA	CB	m1	m2	op	pronto
S0	0	0	-	0	0	-	-	-	1
S1	1	0	1	1	1	-	-	-	0
S2	0	0	-	0	0	-	-	-	0
S3	0	1	-	0	0	0	0	0	0
S4	0	0	0	1	0	1	1	1	0
S5	0	0	-	0	0	-	-	-	1



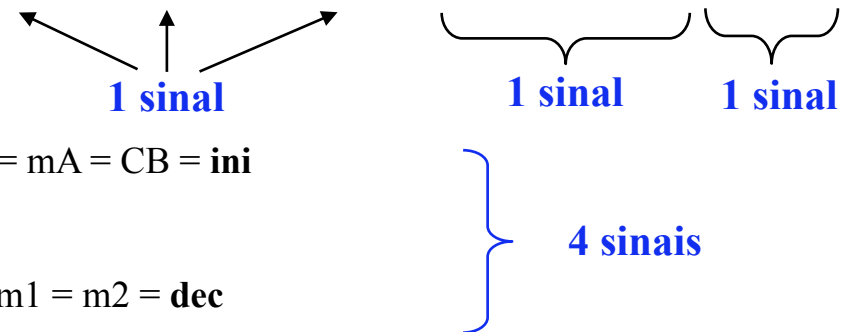
# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

Solução que adotaremos para descrever em VHDL: don't cares = 0



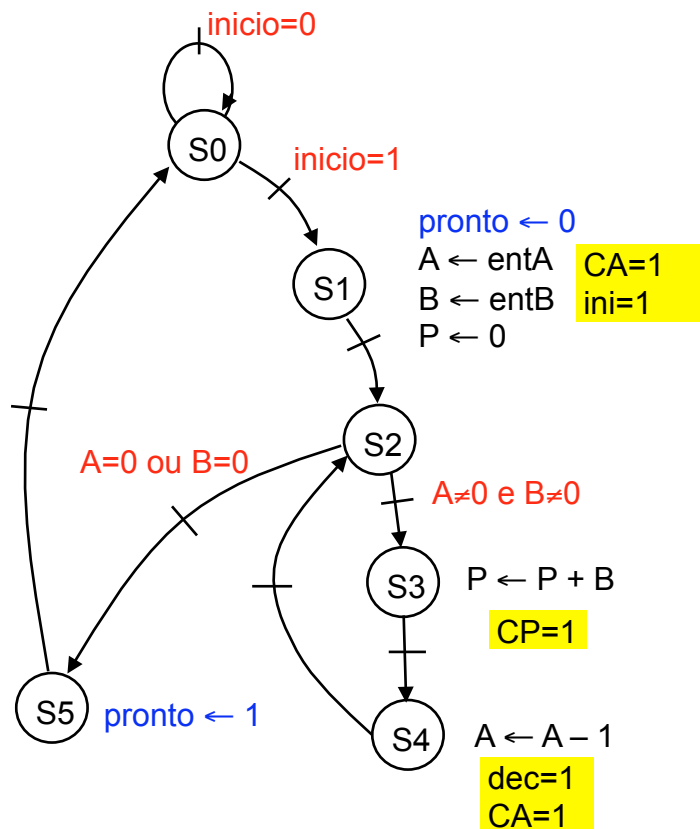
Estado	Reg. P		Reg. A			Somador/Sub			Saída
	RstP	CP	mA	CA	CB	m1	m2	op	pronto
S0	0	0	-	0	0	-	-	-	1
S1	1	0	1	1	1	-	-	-	0
S2	0	0	-	0	0	-	-	-	0
S3	0	1	-	0	0	0	0	0	0
S4	0	0	0	1	0	1	1	1	0
S5	0	0	-	0	0	-	-	-	1



# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

### FSMD/FSM



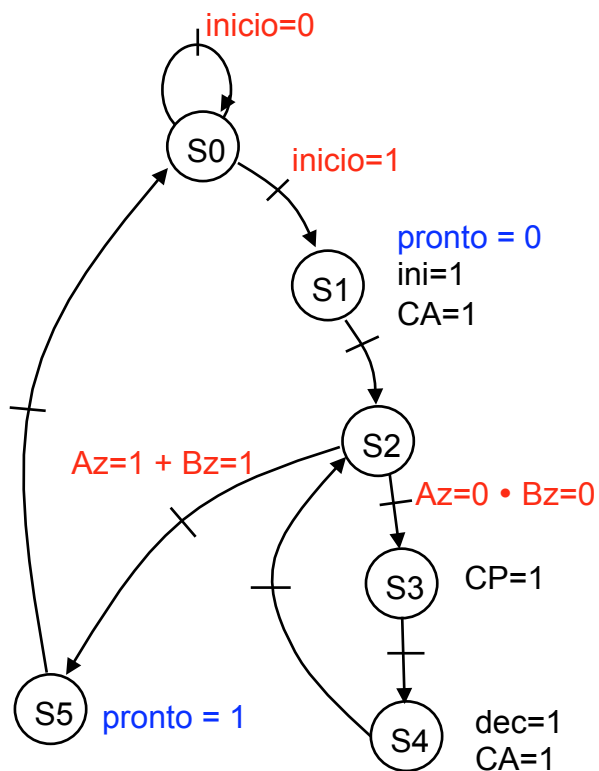
Estado	ini	CP	CA	dec
S0	0	0	0	0
S1	1	0	1	0
S2	0	0	0	0
S3	0	1	0	0
S4	0	0	1	1
S5	0	0	0	0

$RstP = mA = CB = ini$   
**CP**  
**CA**  
 $op = m1 = m2 = dec$

# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

### FSM



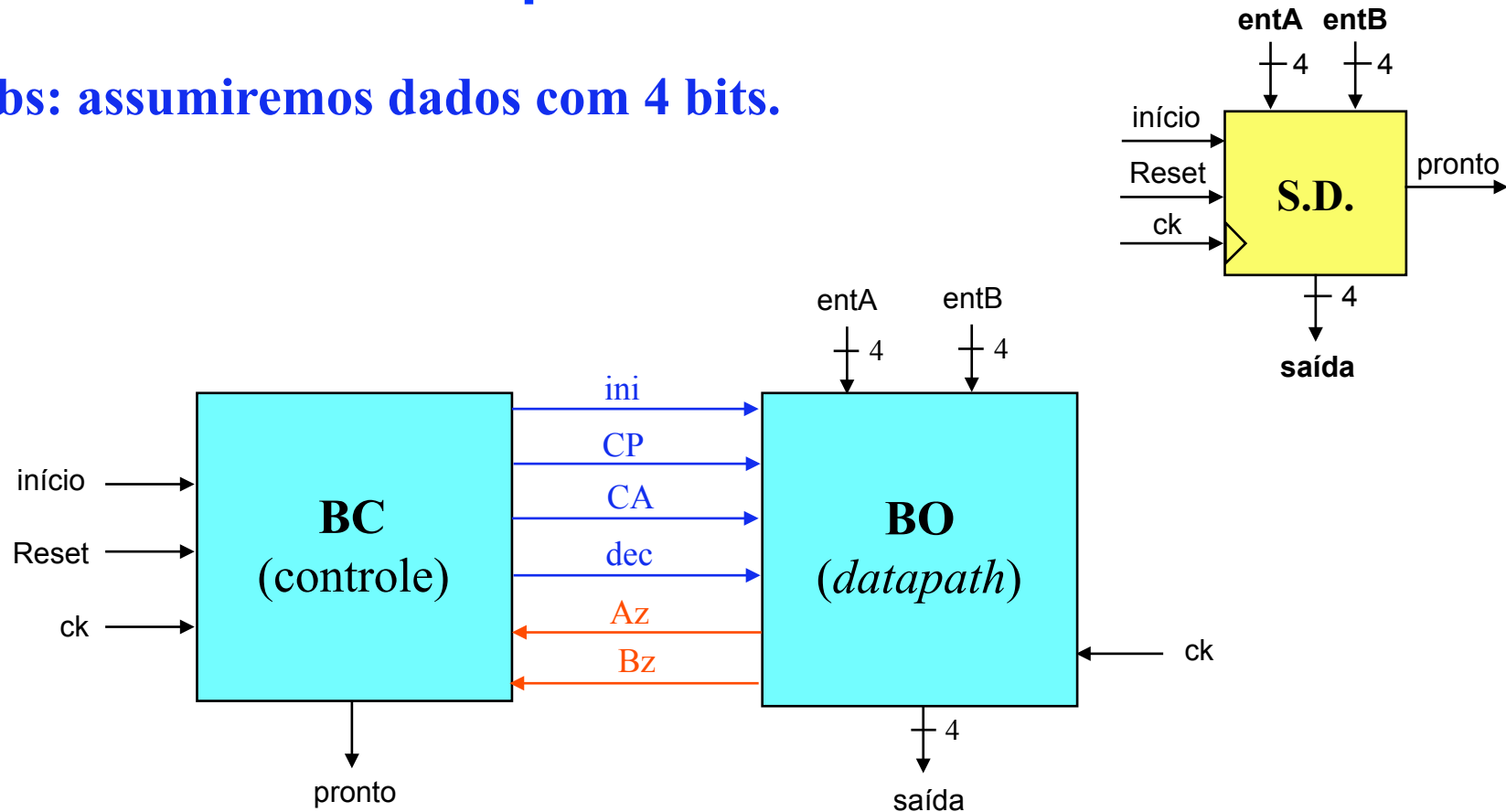
Estado	ini	CP	CA	dec
S0	0	0	0	<b>0</b>
S1	1	0	1	<b>0</b>
S2	0	0	0	<b>0</b>
S3	0	1	0	0
S4	0	0	1	1
S5	0	0	0	0

$RstP = mA = CB = ini$   
**CP**  
**CA**  
 $op = m1 = m2 = dec$

# Sistema Digital Completo (BO + BC)

## ► Interfaces do Multiplicador

Obs: assumiremos dados com 4 bits.

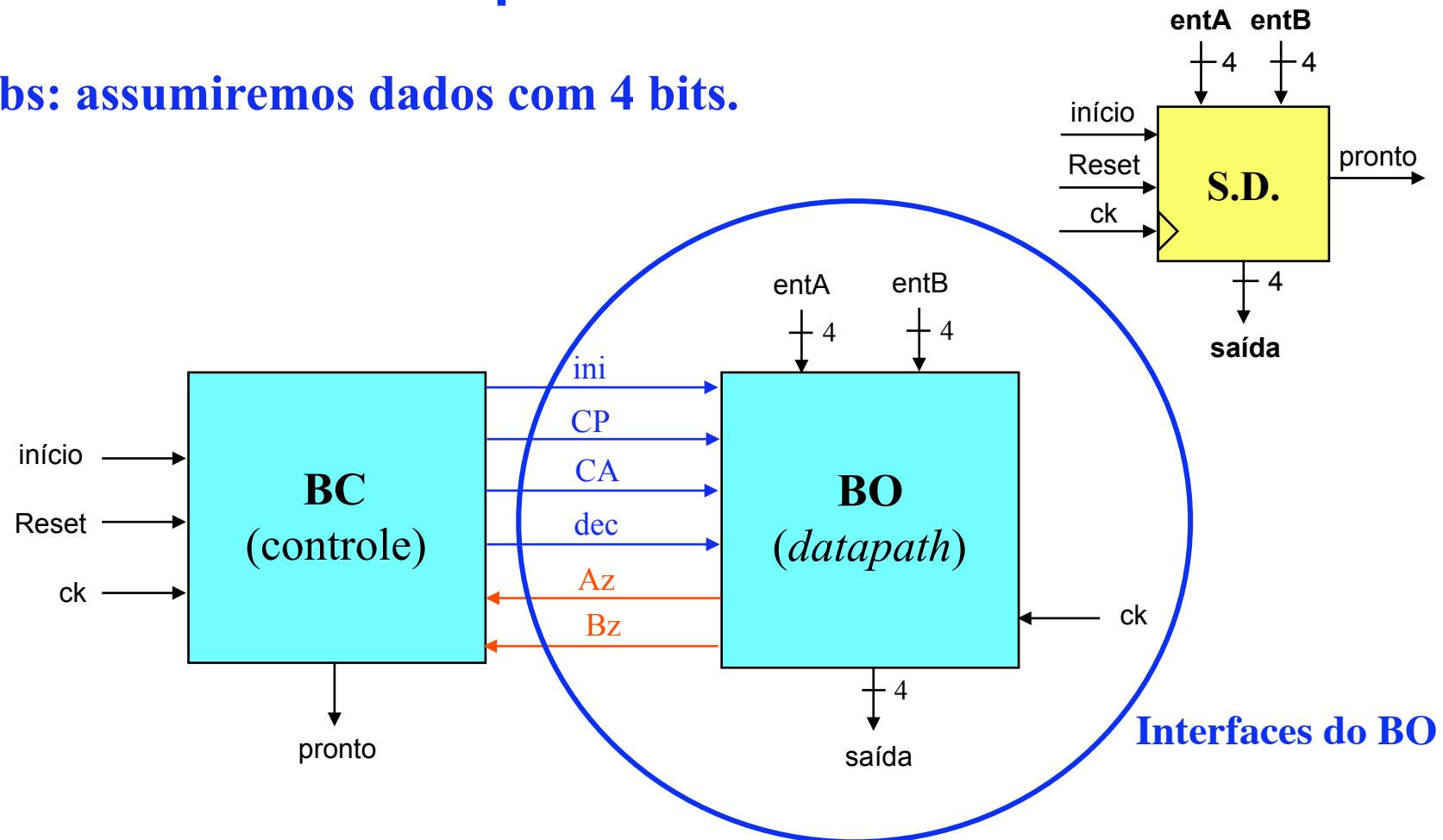




# Sistema Digital Completo (BO + BC)

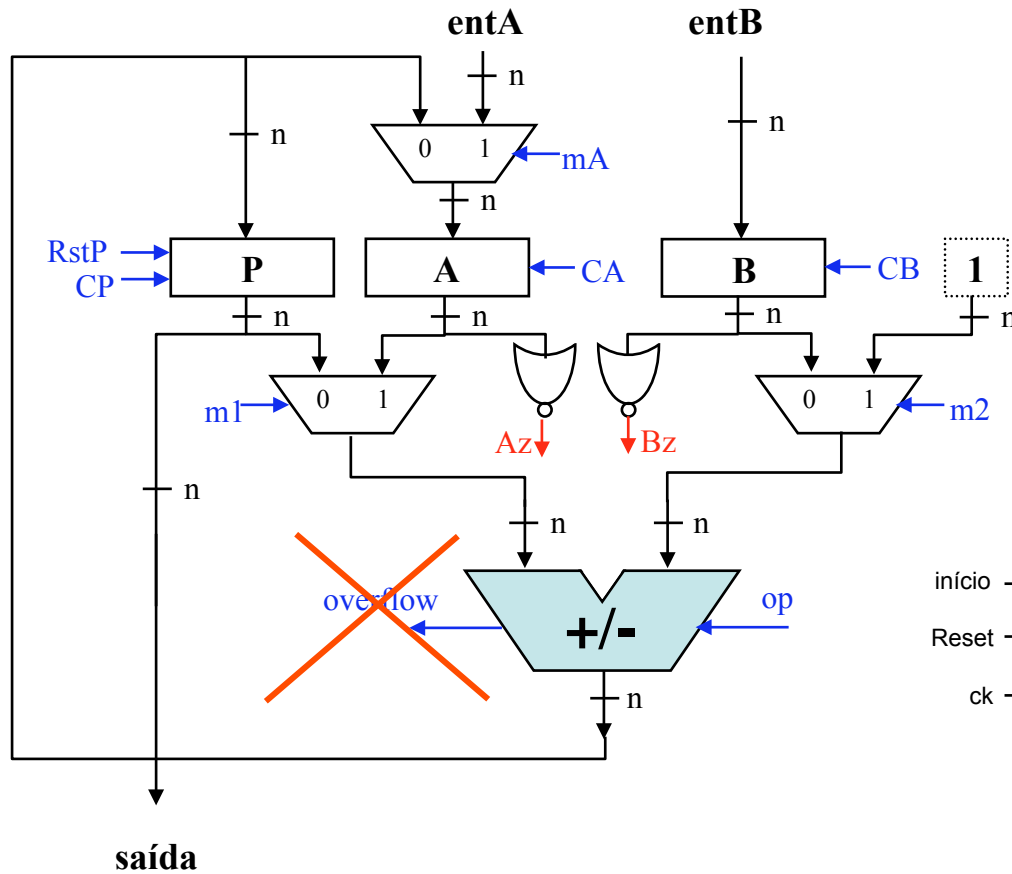
## ► Interfaces do Multiplicador

Obs: assumiremos dados com 4 bits.



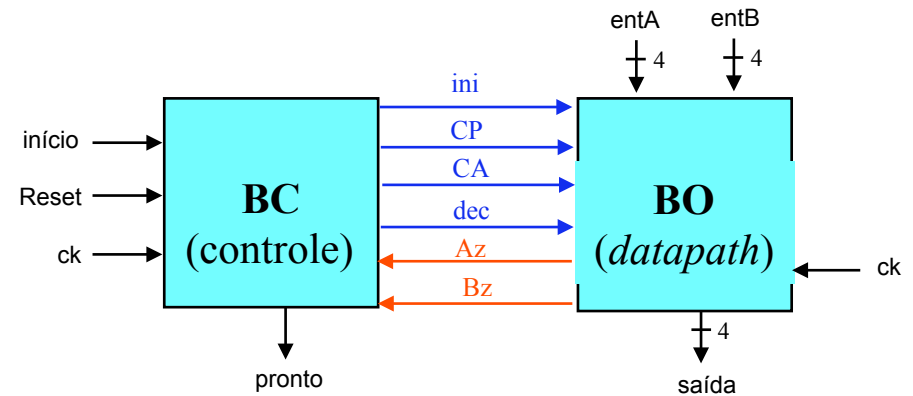
# Sistema Digital Completo (BO + BC)

## ▶ Interfaces do BO



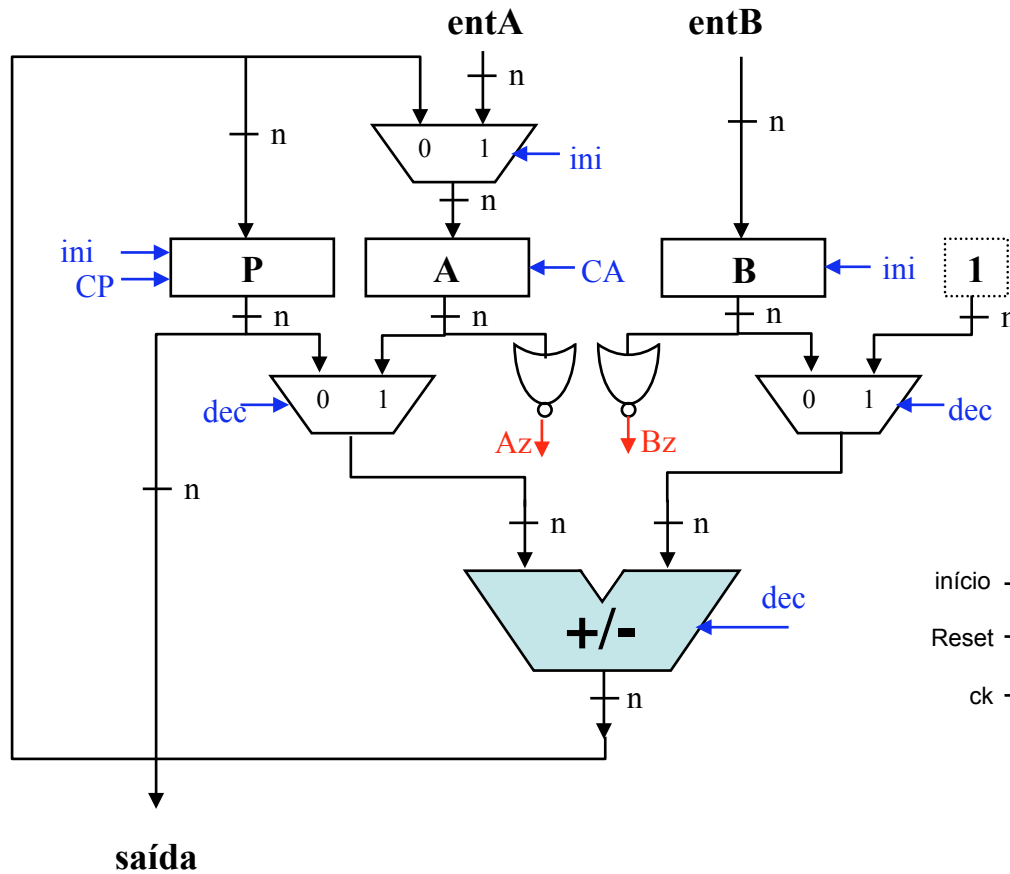
Agrupamento (e novos nomes dos sinais de controle)

RstP = mA = CB = ini  
 CP  
 CA  
 op = m1 = m2 = dec



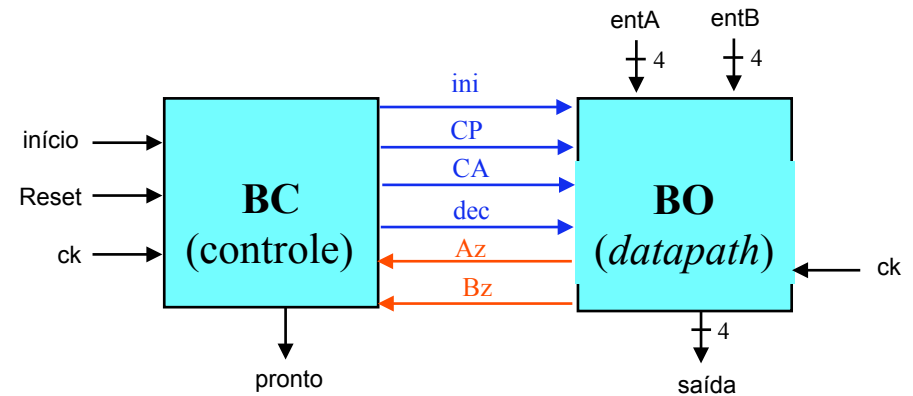
# Sistema Digital Completo (BO + BC)

## ▶ Interfaces do BO



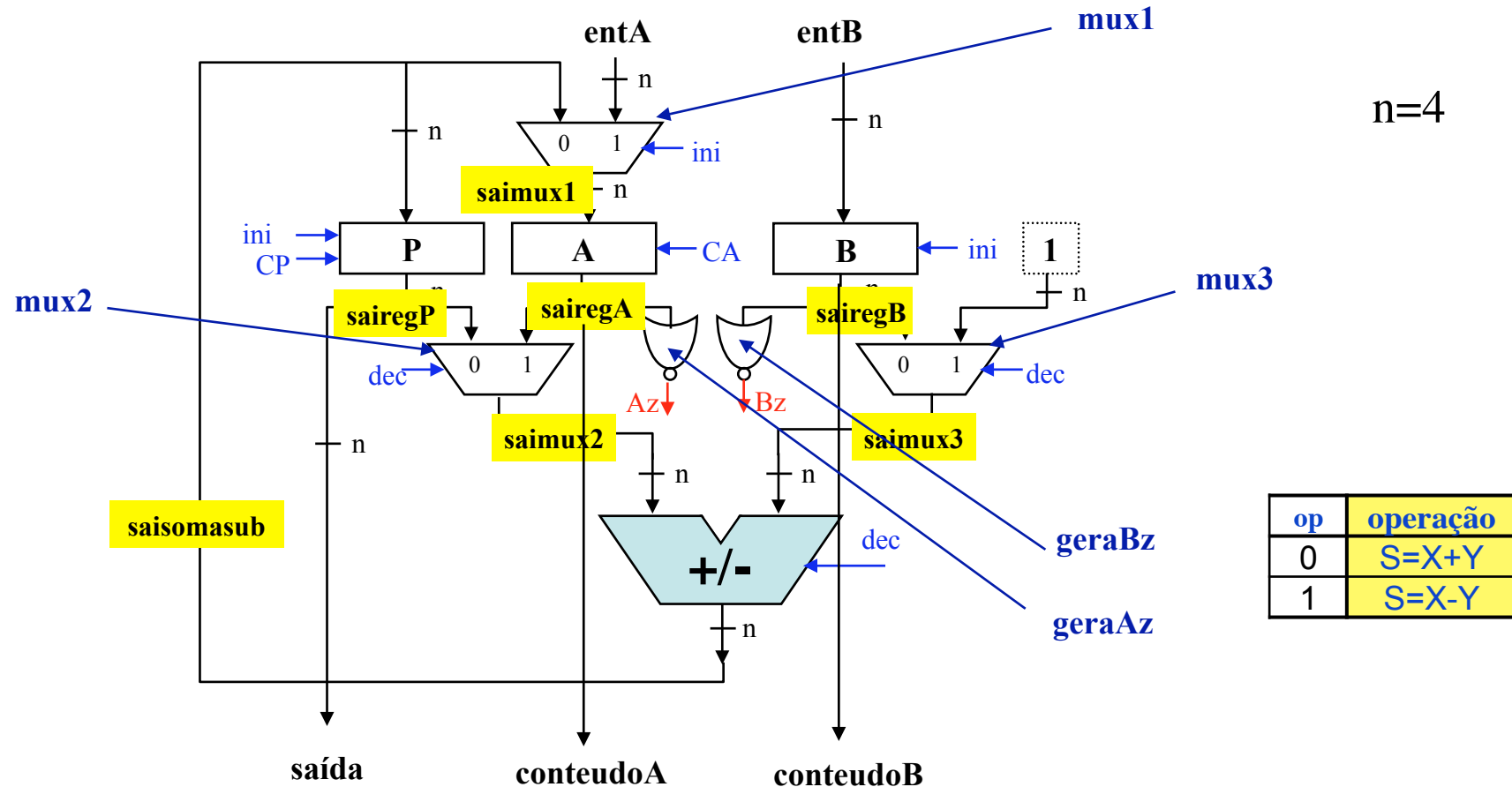
Agrupamento (e novos nomes dos sinais de controle)

RstP = mA = CB = **ini**  
 CP  
 CA  
 op = m1 = m2 = **dec**



# Sistema Digital Completo (BO + BC)

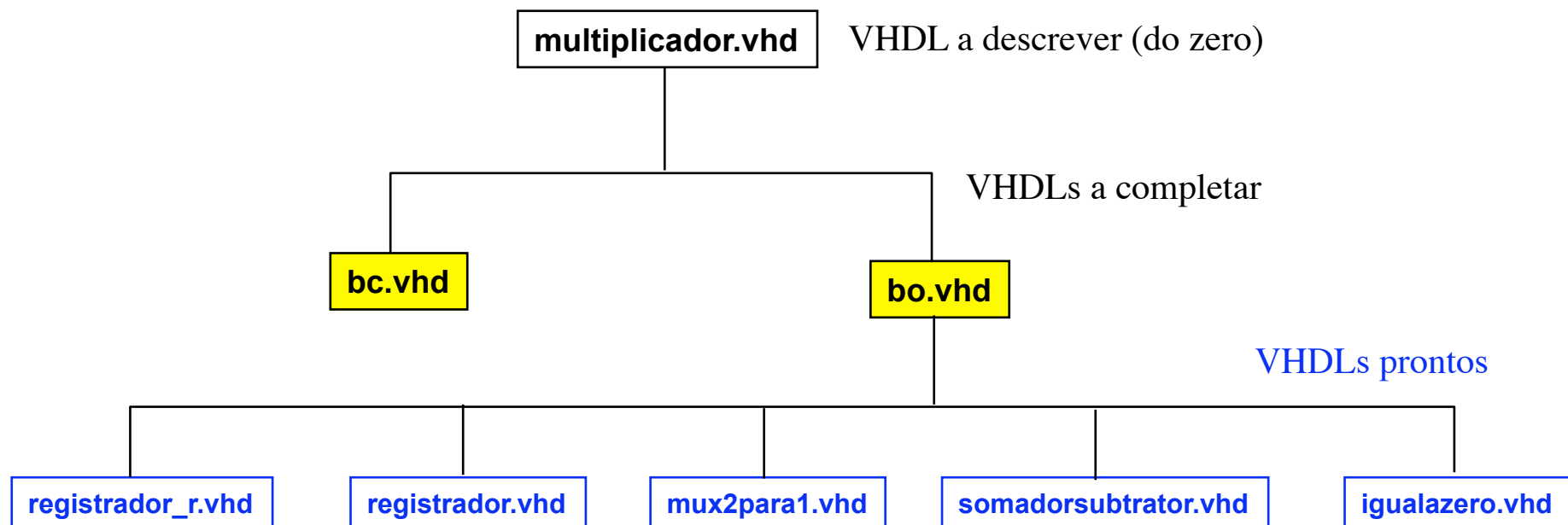
## ► Nomes para os Sinais e Componentes do BO (Usados nos arquivos VHDL fornecidos)



# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL (incompleto) do BO do Multiplicador

### Estrutura de Arquivos



# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL (incompleto) do BO do Multiplicador

```
ENTITY bo IS
PORT (clk : IN STD_LOGIC;
      ini, CP, CA, dec : IN STD_LOGIC;
      entA, entB : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      Az, Bz : OUT STD_LOGIC;
      saida, conteudoA, conteudoB : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END bo;
```

-- comentarios

ARCHITECTURE estrutura OF bo IS

```
    COMPONENT registrador_r IS
    PORT (clk, reset, carga : IN STD_LOGIC;
          d : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
    END COMPONENT;
```

```
    COMPONENT registrador IS
    PORT (clk, carga : IN STD_LOGIC;
          d : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
```

```
    END COMPONENT;
    COMPONENT mux2para1 IS
    PORT ( a, b : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          sel: IN STD_LOGIC;
          y : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
    END COMPONENT;
```

# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL (incompleto) do BO do Multiplicador

```
COMPONENT somadorsubtrator IS
PORT (a, b : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      op: IN STD_LOGIC;
      s : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END COMPONENT;
```

```
COMPONENT igualazero IS
PORT (a : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      igual : OUT STD_LOGIC);
END COMPONENT;
```

```
SIGNAL saimux1, saimux2, saimux3, sairegP, sairegA, sairegB, saisomasub: STD_LOGIC_VECTOR (3 DOWNTO 0);
```

BEGIN

```
mux1: mux2para1 PORT MAP ( );
regP: registrador_r PORT MAP ( );
regA: registrador PORT MAP ( );
regB: registrador PORT MAP ( );
mux2: mux2para1 PORT MAP ( );
mux3: mux2para1 PORT MAP ( );
somasub: somadorsubtrator PORT MAP ( );
geraAz: igualazero PORT MAP ( );
geraBz: igualazero PORT MAP ( );
```

a completar

```
saida <= sairegP;
conteudoA <= sairegA;
conteudoB <= sairegB;
```

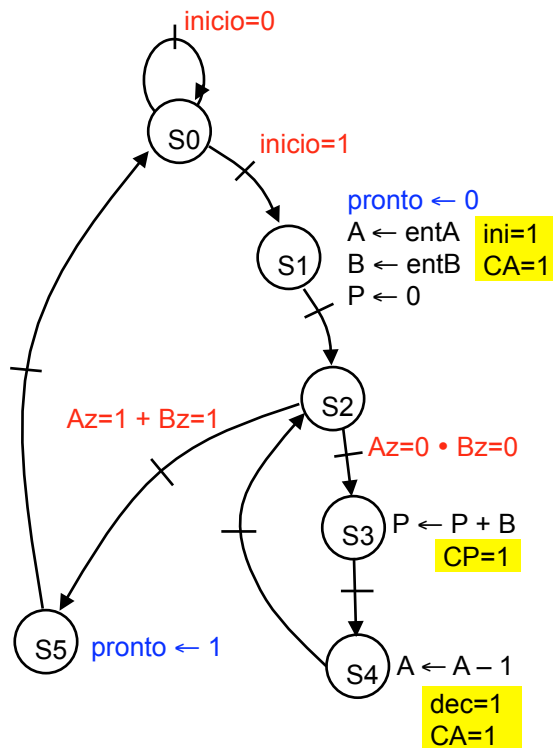
END estrutura;

# Sistema Digital Completo (BO + BC)

## ▶ Planejando a Simulação do BO

Simular 3 x 4: entA = 3, entB = 4 (ao final, A=0 e P=12)

Conteúdo dos registradores e valor de Az durante a execução de 3x4



	S0	S1	S2	S3	S4	S2	S3	S4	S2	S3	S4	S2	S5
<b>A</b>	X	3	3	3	2	2	2	1	1	1	0	0	0
<b>B</b>	X	4	4	4	4	4	4	4	4	4	4	4	4
<b>P</b>	X	0	0	4	4	4	8	8	8	12	12	12	12
<b>Az</b>	X	0	0	0	0	0	0	0	0	0	0	1	0

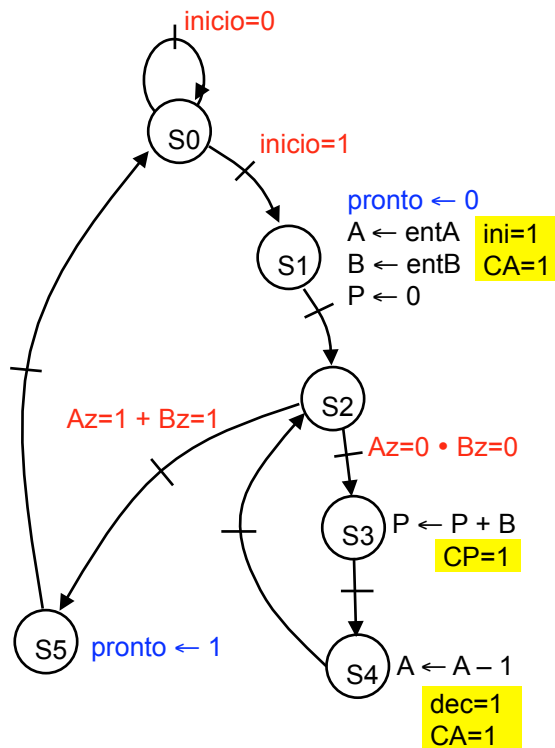
- Os valores desta tabela devem ser confrontados com o resultado da simulação do BO projetado.
- Para realizar a simulação do BO é necessário determinar com precisão os valores dos sinais de comando ini, CA, CP e dec para cada estado (basta olhar a FSM ao lado...)



# Sistema Digital Completo (BO + BC)

## ▶ Planejando a Simulação do BO

Simular 3 x 4: entA = 3, entB = 4 (ao final, A=0 e P=12)



Conteúdo dos registradores e valor de Az durante a execução de 3x4

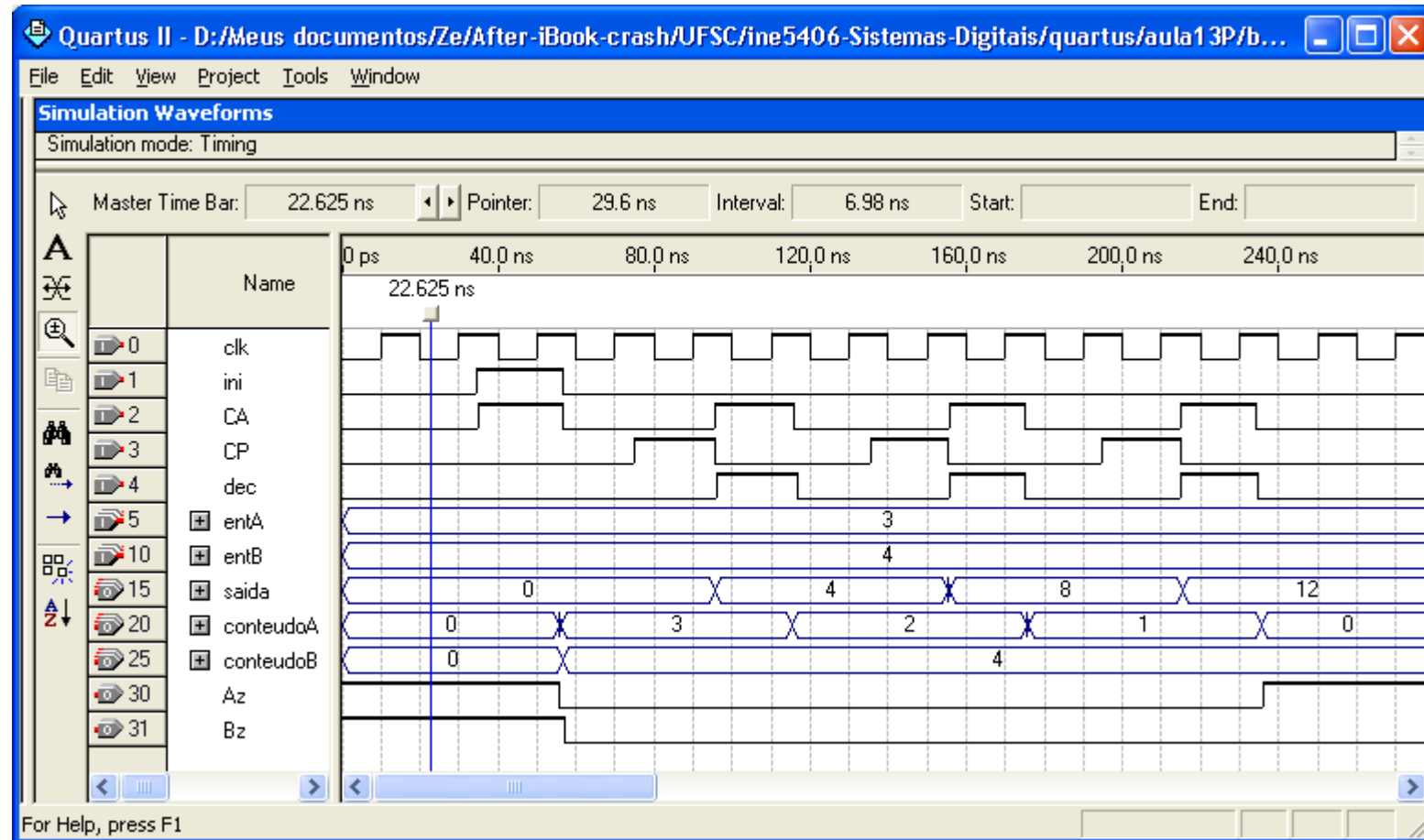
	S0	S1	S2	S3	S4	S2	S3	S4	S2	S3	S4	S2	S5
<b>A</b>	X	3	3	3	2	2	2	1	1	1	0	0	0
<b>B</b>	X	4	4	4	4	4	4	4	4	4	4	4	4
<b>P</b>	X	0	0	4	4	4	8	8	8	12	12	12	12
<b>Az</b>	X	0	0	0	0	0	0	0	0	0	0	1	0

<b>ini</b>	0	1	0	0	0	0	0	0	0	0	0	0	0
<b>CA</b>	0	1	0	0	1	0	0	1	0	0	1	0	0
<b>CP</b>	0	0	0	1	0	0	1	0	0	1	0	0	0
<b>dec</b>	0	0	0	0	1	0	0	1	0	0	1	0	0

Sinais de comando para a execução de 3x4 (i.e., sinais que comandam a operação 3x4 no BO; devem ser usados para simular o BO)

# Sistema Digital Completo (BO + BC)

## ▶ Simulação do BO

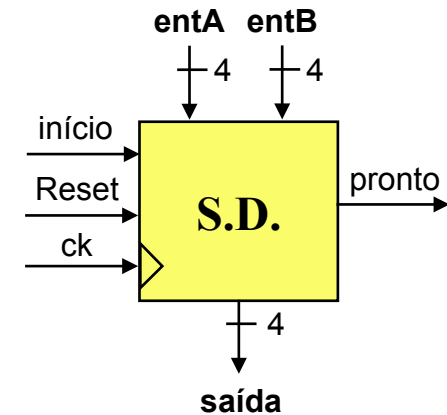
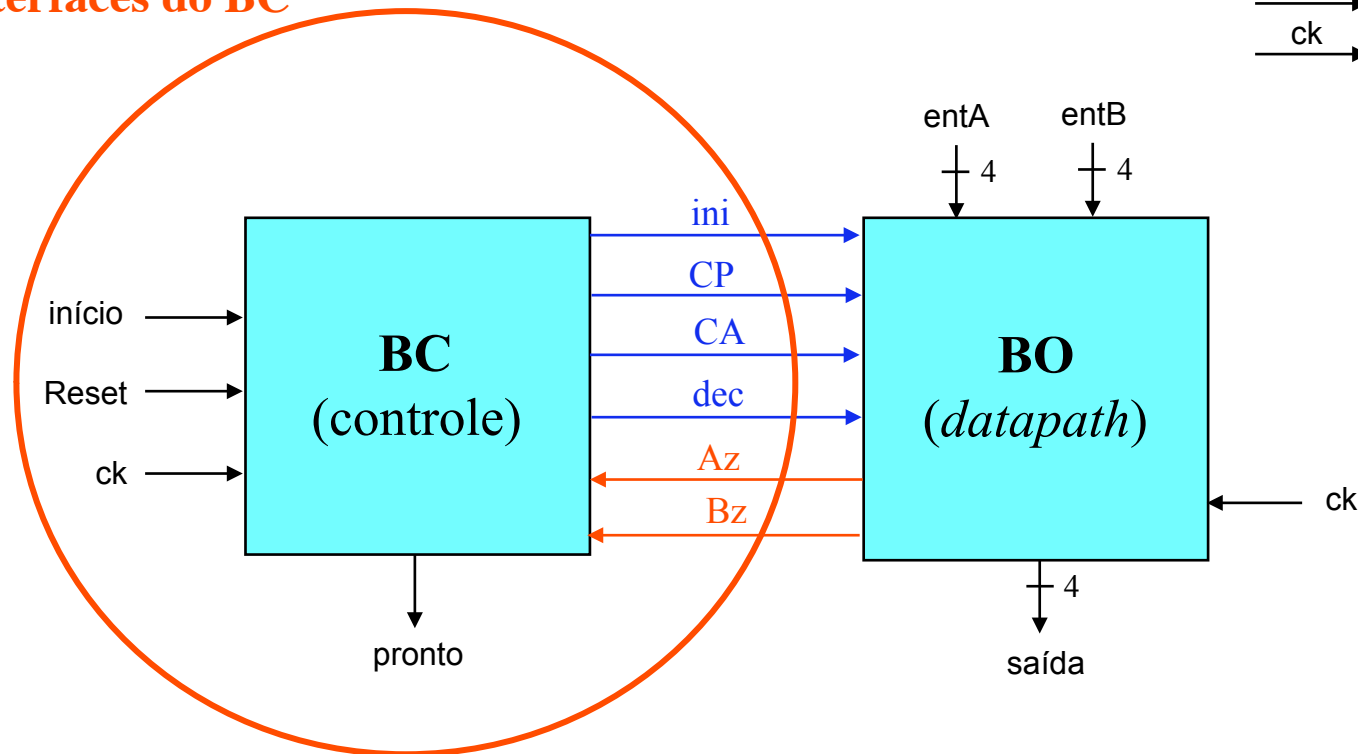


# Sistema Digital Completo (BO + BC)

## ► Interfaces do Multiplicador

Obs: assumiremos dados com 4 bits.

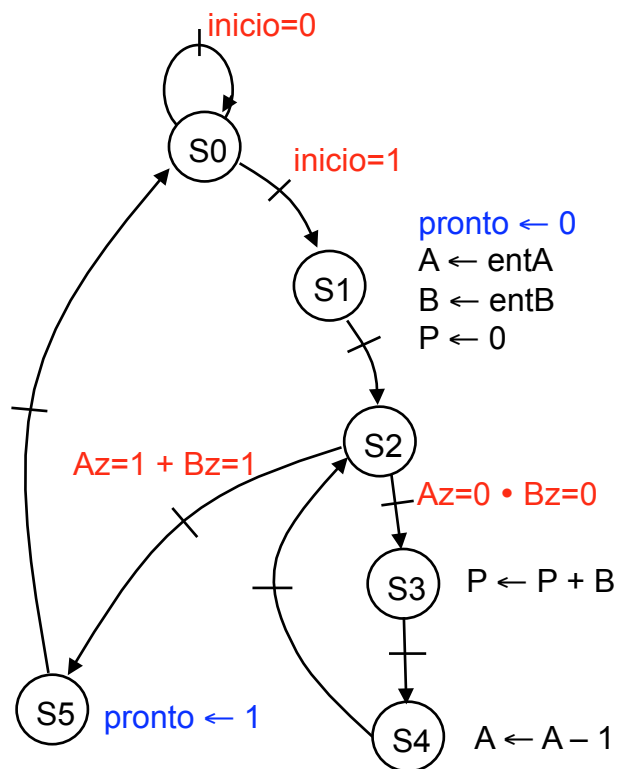
### Interfaces do BC



# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

### Interfaces de Controle, FSMD e Tabela de Transição de Estados

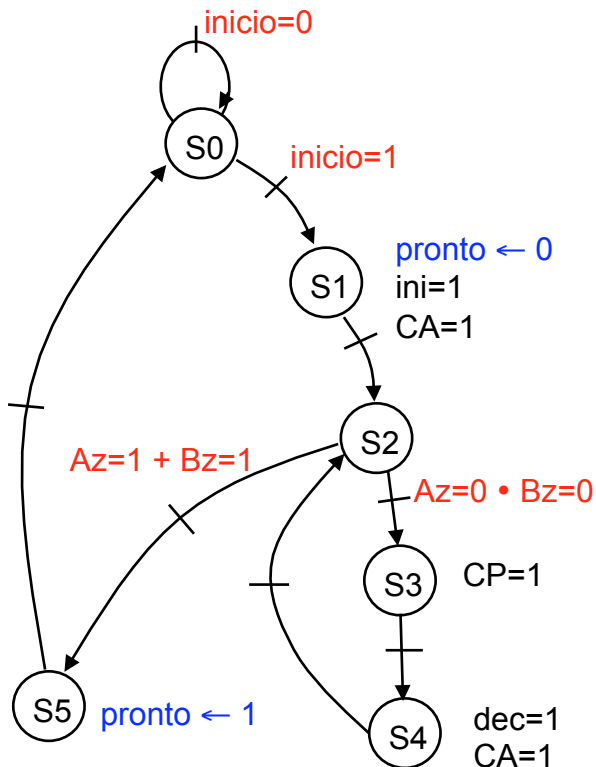


Estado atual	Entradas			Próx. Estado
	início	BZ	AZ	
S0	0	-	-	S0
	1	-	-	S1
S1	-	-	-	S2
S2	-	0	0	S3
	-	0	1	S5
	-	1	0	S5
	-	1	1	S5
S3	-	-	-	S4
S4	-	-	-	S2
S5	-	-	-	S0

# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

### Tabela de Transição de Estados e FSM

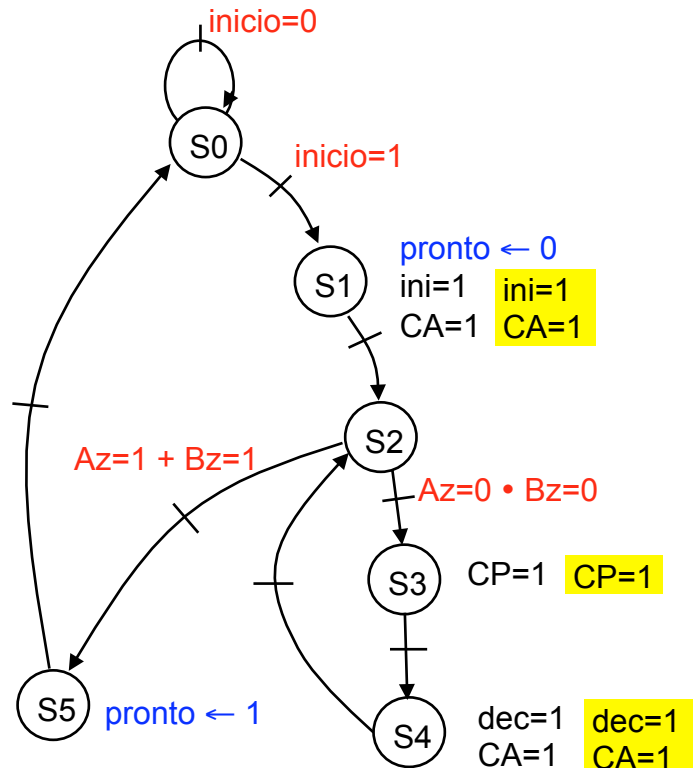


Estado atual	Entradas			Próx. Estado
	início	BZ	AZ	
S0	0	-	-	S0
	1	-	-	S1
S1	-	-	-	S2
S2	-	0	0	S3
	-	0	1	S5
	-	1	0	S5
	-	1	1	S5
S3	-	-	-	S4
S4	-	-	-	S2
S5	-	-	-	S0

# Sistema Digital Completo (BO + BC)

## ► Bloco de Controle Visando Custo Mínimo

### Tabela de Saídas (e FSMs)



Estado	ini	CP	CA	dec	pronto
S0	0	0	0	0	0
S1	1	0	1	0	0
S2	0	0	0	0	0
S3	0	1	0	0	0
S4	0	0	1	1	0
S5	0	0	0	0	1

Sinais de comando (underlined in original): ini, CP, CA, dec. Saída: pronto.

# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL do BC do Multiplicador

```
ENTITY bc IS
PORT (Reset, clk, inicio : IN STD_LOGIC;
      Az, Bz : IN STD_LOGIC;
      pronto : OUT STD_LOGIC;
      ini, CA, dec, CP: OUT STD_LOGIC );
END bc;

-- comentarios

ARCHITECTURE estrutura OF bc IS
    TYPE state_type IS (S0, S1, S2, S3, S4, S5 );
    SIGNAL state: state_type;
BEGIN
    -- Logica de proximo estado (e registrador de estado)
    PROCESS (clk, Reset)
    BEGIN
        if(Reset = '1') THEN
            state <= S0 ;
        ELSIF (clk'EVENT AND clk = '1') THEN
            CASE state IS
                WHEN S0 =>
                    -- completar
                WHEN S1 =>
                    -- completar
                WHEN S2 =>
                    -- completar
```

a completar

# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL do BC do Multiplicador

```
-- completar WHEN S3 =>  
  
-- completar WHEN S4 =>  
  
-- completar WHEN S5 =>
```

a completar

```
END CASE;  
END IF;  
END PROCESS;
```

Continua na próxima transparência...



# Sistema Digital Completo (BO + BC)

## ▶ Código VHDL do BC do Multiplicador

```
-- Logica de saida  
PROCESS (state)  
BEGIN  
    CASE state IS  
        WHEN S0 =>  
            ini <= '0';  
            CA <= '0';  
            dec <= '0';  
            CP <= '0';  
            pronto <= '0';
```

```
        WHEN S1 =>  
            -- completar  
        WHEN S2 =>  
            -- completar  
        WHEN S3 =>  
            -- completar  
        WHEN S4 =>  
            -- completar  
        WHEN S5 =>  
            -- completar
```

```
    END CASE;  
END PROCESS;  
END estrutura;
```

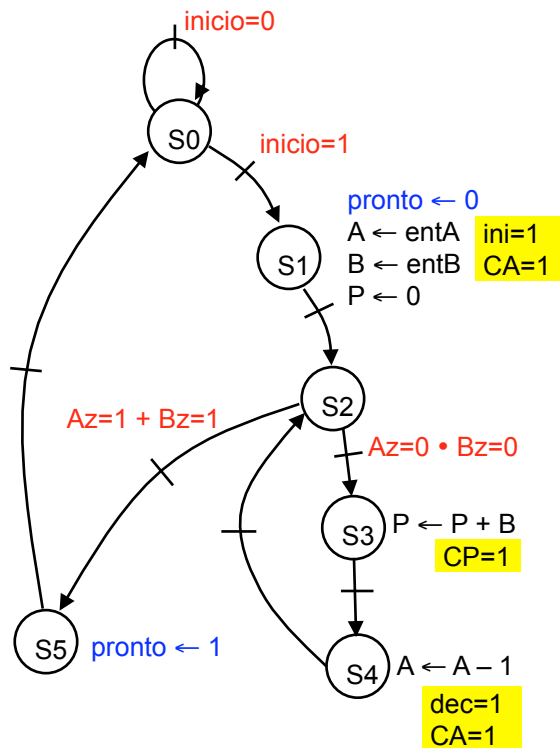
a completar

# Sistema Digital Completo (BO + BC)

## ▶ Planejando a Simulação do BC

Sinais de entrada usados pelo BC para decidir trocas de estados

	S0	S1	S2	S3	S4	S2	S3	S4	S2	S3	S4	S2	S5
<b>Az</b>	X	0	0	0	0	0	0	0	0	0	0	1	0
<b>ini</b> <b>cio</b>	1	0	0	0	0	0	0	0	0	0	0	0	0



Conteúdo dos registradores (para acompanhamento da execução...)

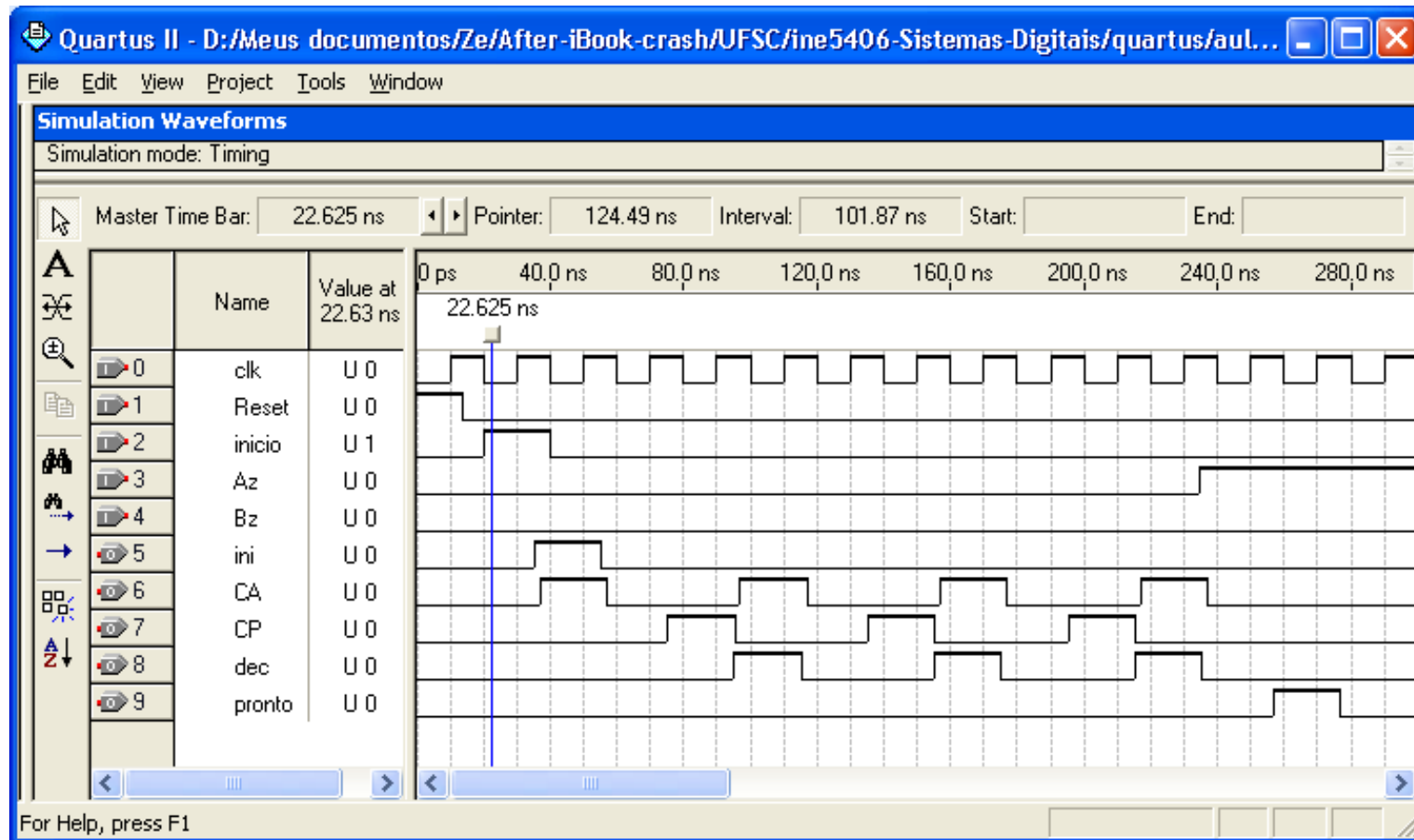
<b>A</b>	X	3	3	3	2	2	2	1	1	1	0	0	0
<b>B</b>	X	4	4	4	4	4	4	4	4	4	4	4	4
<b>P</b>	X	0	0	4	4	4	8	8	8	12	12	12	12

<b>ini</b>	0	1	0	0	0	0	0	0	0	0	0	0	0
<b>CA</b>	0	1	0	0	1	0	0	1	0	0	1	0	0
<b>CP</b>	0	0	0	1	0	0	1	0	0	1	0	0	0
<b>dec</b>	0	0	0	0	1	0	0	1	0	0	1	0	0

Sinais de comando que devem ser gerados pelo BC para a execução de 3x4 (comparar com o resultado da simulação do BC)

# Sistema Digital Completo (BO + BC)

## ▶ Simulação do BC

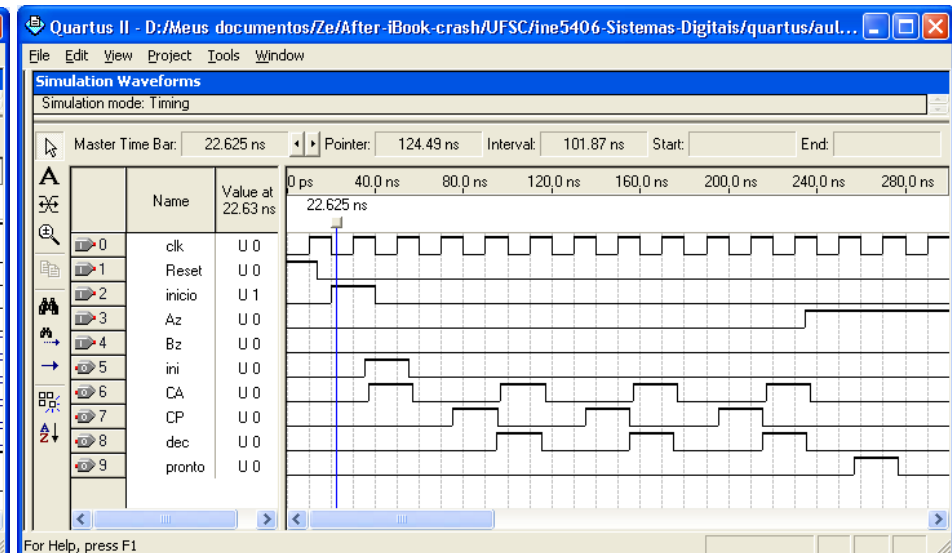
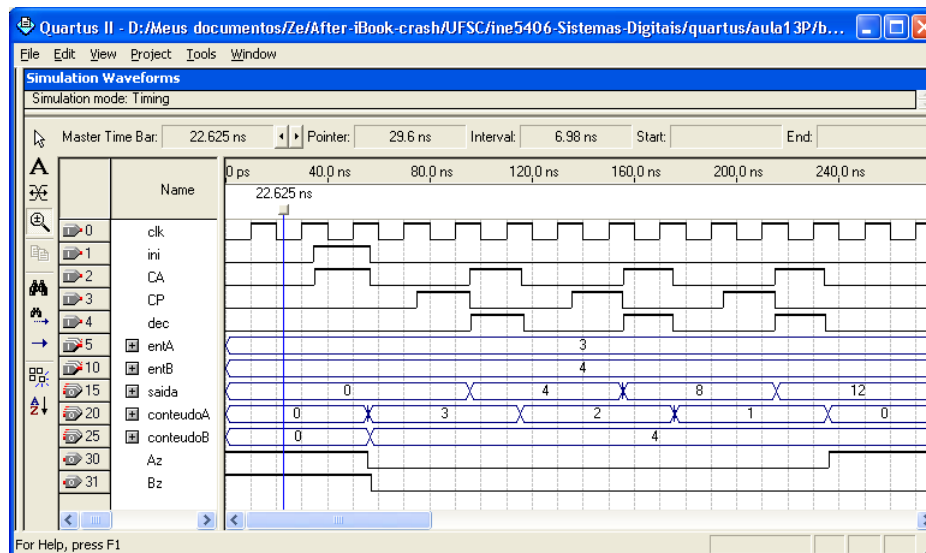


# Sistema Digital Completo (BO + BC)

## ▶ Confrontando as Simulações de BO e de BC (p/ 3x4)

Resultado da simulação do BO

Resultado da simulação do BC



# Sistema Digital Completo (BO + BC)

---

## ▶ Outros Experimentos (p/ Casa)

- Repetir as simulações de BO, BC e multiplicador para os seguintes operandos (entA x entB)
  - 4 x 3
  - 3 x 6
- O que ocorreu com o resultado da segunda operação acima?
- Modifique o VHDL de modo que este problema não ocorra mais.
- Tornar o código VHDL parametrizável, com relação ao número de bits dos operandos