Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Ciências da Computação & Engenharia Eletrônica

Sistemas Digitais

INE 5406

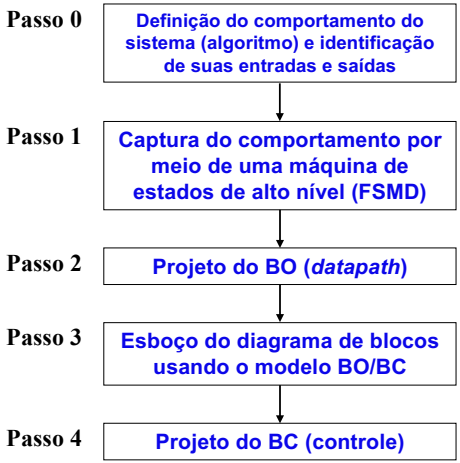
Aula 4-T

2. Processadores Dedicados (Blocos Aceleradores). Método de Projeto no Nível RT. Exemplos 3 e 4 de somadores sequenciais.

Profs. José Luís Güntzel e Cristina Meinhardt
{j.guntzel, cristina.meinhardt}@ufsc.br

Processadores Dedicados

Método de Projeto de Sistemas Digitais no Nível RT

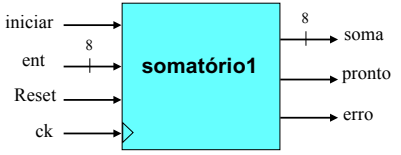


Processadores Dedicados

Aula Passada: Exemplo 1: Enunciado

SD (bloco acelerador) para cálculo de um somatório de 4 números

Especificação das interfaces: Necessita-se de um sistema digital (SD) dedicado (i.e., um bloco acelerador) capaz de realizar o cálculo **A+B+C+D**, onde **A**, **B**, **C** e **D** são números*** inteiros sem sinal**, representados em **binário com 8 bits**. Este sistema digital, doravante denominado de "somatório1", possui uma entrada de relógio ("ck"), uma entrada de reset assíncrono ("Reset"), **uma** entrada de dados com 8 bits ("ent"), uma entrada de controle denominada "iniciar", duas saídas de controle ("pronto" e "erro") e uma saída de dados de 8 bits ("soma").



* Os números fornecidos como entrada do sistema são comumente chamados de "operandos (de entrada)".

Processadores Dedicados

Aula Passada: Exemplo 1: Enunciado

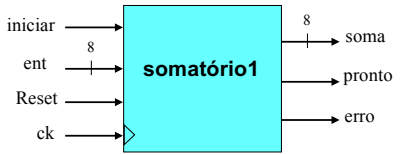
SD (bloco acelerador) para cálculo de um somatório de 4 números

Especificação do comportamento:

- Há dois estados iniciais, **S0** e **E**. Enquanto um novo cálculo não inicia, “somatório1” permanece em um destes dois estados (ver explicação no último item);
- O sinal externo “iniciar” dá o comando para iniciar um cálculo **A+B+C+D**.
- À medida que o cálculo é realizado, os valores dos operandos **A**, **B**, **C** e **D** vão sendo fornecidos pela entrada ”ent", em bordas de relógio consecutivas;
- Uma vez iniciado, o cálculo é realizado de maneira sequencial e cumulativa (i.e., cada novo operando de entrada que chega é somado ao valor acumulado até então);
- Caso ocorra *overflow* em alguma das adições, o cálculo deve terminar imediatamente, com o SD parando no estado **E** (que indica término com erro). Caso não ocorra *overflow*, o cálculo termina com o SD parando no estado **S0**.

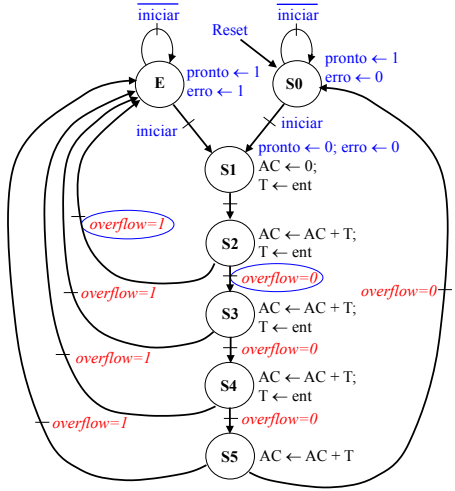
Processadores Dedicados

Aula Passada: Exemplo 1



1. AC ← 0; T← ent; // A está estável em ent
2. AC ← AC + T; T← ent; // B está estável em ent
3. AC ← AC + T; T← ent; // C está estável em ent
4. AC ← AC + T; T← ent; // D está estável em ent
5. AC ← AC + T; // O resultado final S estará em AC

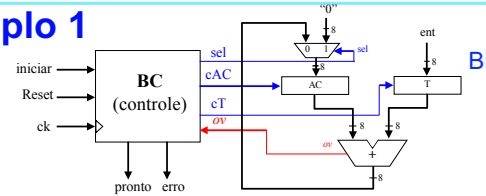
O 1º teste de overflow seria dispensável, uma vez que na primeira soma um dos operandos é zero. Porém, **iremos mantê-lo para permitir uma futura generalização do algoritmo.**



Processadores Dedicados

Aula Passada: Exemplo 1

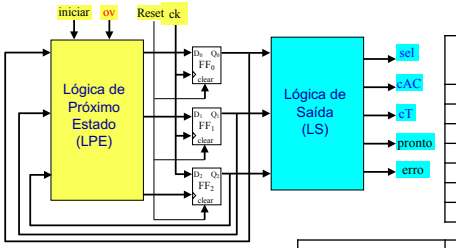
Sistema Digital somatório1



Bloco Operativo (BO)

Lógica de Próximo Estado (LPE)

Q2	Q1	Q0	iniciar	*ov*	$Q2^+$	$Q1^+$	$Q0^+$
1	**1**	**1**	0	X	**1**	**1**	**1**
1	**1**	**1**	1	X	0	0	1
0	0	0	0	X	0	0	0
0	0	0	1	X	0	0	1
0	0	1	X	X	0	1	0
0	1	0	X	0	0	1	1
0	1	0	X	1	**1**	**1**	**1**
0	1	1	X	0	1	0	0
0	1	1	X	1	**1**	**1**	**1**
1	0	0	X	0	1	0	1
1	0	0	X	1	**1**	**1**	**1**
1	0	1	X	0	0	0	0
1	0	1	X	1	**1**	**1**	**1**



Lógica de Saída (LS)

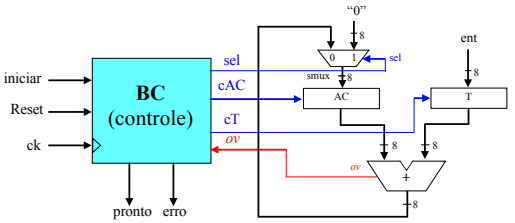
Q2	Q1	Q0	Sinais de comando			Saídas de controle	
			sel	cAC	cT	pronto	erro
1	1	1	X	0	0	1	1
0	0	0	X	0	0	1	0
0	0	1	1	1	1	0	0
0	1	0	0	1	1	0	0
0	1	1	0	1	1	0	0
1	0	0	0	1	1	0	0
1	0	1	0	1	0	0	0

T_{exec} = nº de ciclos x T = 5 ciclos x 5ns = 25 ns

	Estimativa de Custo
B.O.	576 transistores
B.C.	7 estados
	5 sinais de controle

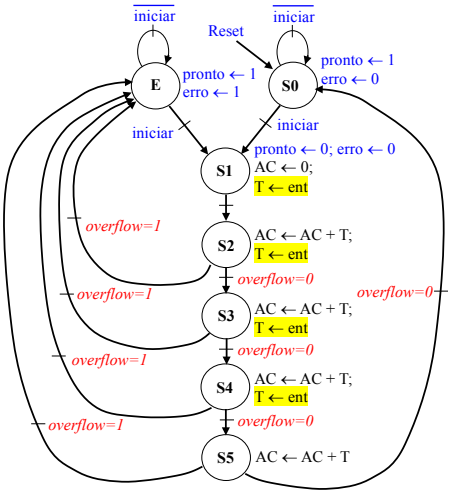
Processadores Dedicados

Aula Passada: Exemplo 1



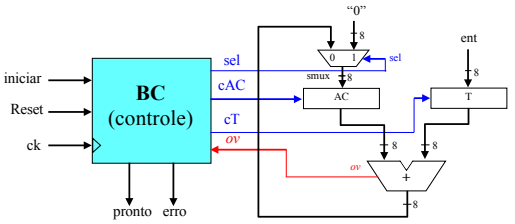
Observações sobre esta solução:

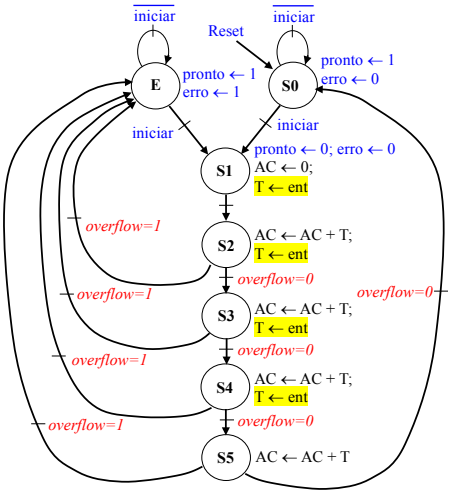
1. Ela não é configurável: só permite somar 4 operandos
2. O número de estados da FSM é diretamente proporcional ao número de operandos. Exemplos:
   - 4 operandos → 7 estados (E, S0, S1, .., S5)
   - 8 operandos → 11 estados (E, S0, S1, ... , S9)
   - 1000 operandos → 1003 estados (E, S0, S1, ... , S1001)



Processadores Dedicados

Aula Passada: Exemplo 1





O Exemplo 3 a seguir traz uma versão mais flexível de sistema digital, na qual o número de operandos de entrada pode ser configurado dentro de um intervalo [1, nro_máx]

Processadores Dedicados

Exemplo 3: Enunciado

SD (bloco acelerador) para cálculo de um somatório de *n* números

Especificação das interfaces: Necessita-se de um sistema digital (SD) dedicado (i.e., um bloco acelerador) capaz de realizar o somatório de ***n*** números inteiros sem sinal **A**, **B**, **C**, **D ...,** representados em **binário com 8 bits**, onde $0 < n <= 8$. Este sistema digital, doravante denominado de "somatório3", possui uma entrada de relógio ("ck"), uma entrada de reset assíncrono ("Reset"), **uma** entrada de dados com 8 bits ("ent"), uma entrada para receber o valor ***n***, uma entrada de controle denominada "iniciar", duas saídas de controle ("pronto" e "erro") e uma saída de dados de 8 bits ("soma").

Quantos bits precisa ter a entrada *n*?

* Os números fornecidos como entrada do sistema são comumente chamados de "operandos (de entrada)".

Processadores Dedicados

Exemplo 3: Enunciado

SD (bloco acelerador) para cálculo de um somatório de *n* números

Especificação das interfaces: Necessita-se de um sistema digital (SD) dedicado (i.e., um bloco acelerador) capaz de realizar o somatório de ***n*** números inteiros sem sinal **A**, **B**, **C**, **D ...,** representados em **binário com 8 bits**, onde $0 < n <= 8$. Este sistema digital, doravante denominado de "somatório3", possui uma entrada de relógio ("ck"), uma entrada de reset assíncrono ("Reset"), **uma** entrada de dados com 8 bits ("ent"), uma entrada para receber o valor ***n***, uma entrada de controle denominada "iniciar", duas saídas de controle ("pronto" e "erro") e uma saída de dados de 8 bits ("soma").

Quantos bits precisa ter a entrada ***n***?

Resp.:

O maior ***n*** é 8, que em binário é 1000.
Logo, ***n*** é uma variável 4 bits.

* Os números fornecidos como entrada do sistema são comumente chamados de "operandos (de entrada)".

Processadores Dedicados

Exemplo 3: Enunciado

SD (bloco acelerador) para cálculo de um somatório de n números

Especificação do comportamento:

- Há dois estados iniciais, **S0** e **E**. Enquanto um novo cálculo não inicia, "somatório3" permanece em um destes dois estados, conforme tenha terminado o cálculo anterior.
- O sinal externo "iniciar" dá o comando para iniciar um cálculo do somatório.
- À medida que o cálculo é realizado, os valores dos operandos **A**, **B**, **C**, **D** ... vão sendo fornecidos pela entrada "ent", um a cada duas bordas de relógio consecutivas.
- Assuma que o valor ***n*** é fornecido pela entrada de mesmo nome no primeiro estado do cálculo, i.e., S1.
- Uma vez iniciado, o cálculo é realizado de maneira sequencial e cumulativa (i.e., cada novo operando de entrada que chega é somado ao valor acumulado até então).
- Caso ocorra *overflow* em alguma das adições, o cálculo deve terminar imediatamente, com o SD parando no estado **E**. Caso não ocorra *overflow*, o cálculo termina com o SD parando no estado **S0**.

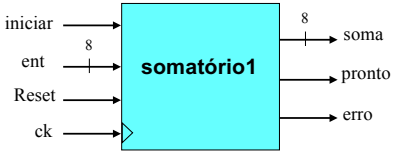
Processadores Dedicados

Exemplo 3: Passo 1 (Captura do comportamento por meio de uma FSMD)

0. Início
1. AC ← 0; T← ent; cont ← *n*; pronto ← 0;
2. **Enquanto** cont ≠ 0 **faça {**
3. AC ← AC + T; T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E

Processadores Dedicados

Comparação somatório3 x somatório1

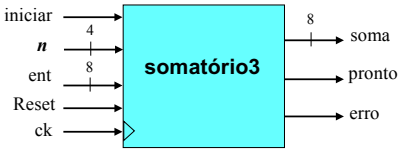


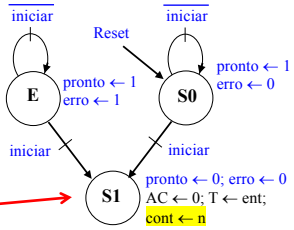
0. Início
1. AC ← 0; T← ent; cont ← *n*; pronto ← 0;
2. **Enquanto** cont ≠ 0 **faça {**
3. AC ← AC + T; T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E

0. Início
1. AC ← 0; T← ent; pronto ← 0; // A está estável em ent
2. AC ← AC + T; T← ent; // B está estável em ent
3. AC ← AC + T; T← ent; // C está estável em ent
4. AC ← AC + T; T← ent; // D está estável em ent
5. AC ← AC + T; // O resultado final S estará em AC
6. pronto ← 1; //esta linha ocorrerá em S0 e em E

Processadores Dedicados

Exemplo 3: Passo 1 (Captura do comportamento por meio de uma FSMD)



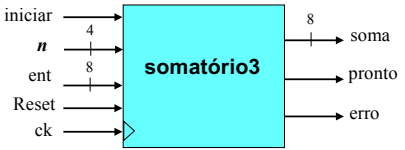


0. Início
1. AC ← 0; T← ent; cont ← *n*; pronto ← 0;
2. **Enquanto** cont ≠ 0 **faça {**
3. AC ← AC + T; T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E

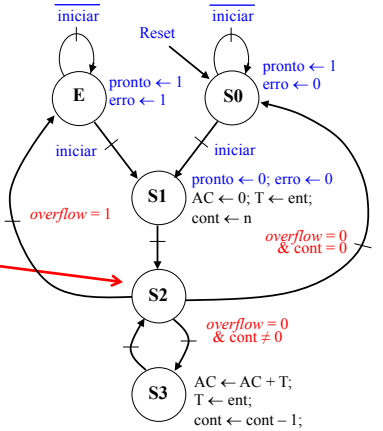
Em comparação ao Exemplo 1, os estados E e S0 são idênticos e o estado S1 se diferencia somente pela inicialização da variável "cont".

Processadores Dedicados

Exemplo 3: Passo 1 (Captura do comportamento por meio de uma FSMD)



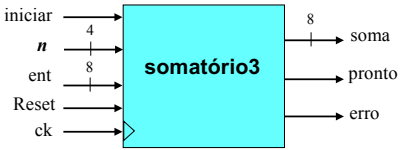
0. Início
1. AC ← 0; T← ent; cont ← *n*; pronto ← 0;
2. **Enquanto** cont ≠ 0 **faça {**
3. AC ← AC + T; T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E



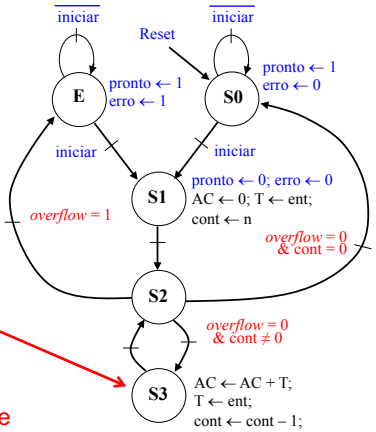
O teste " cont ≠ 0" será executado por um circuito feito de portas lógicas que possuem atraso. Por isso, precisamos prever um estado para fazer este teste.

Processadores Dedicados

Exemplo 3: Passo 1 (Captura do comportamento por meio de uma FSMD)



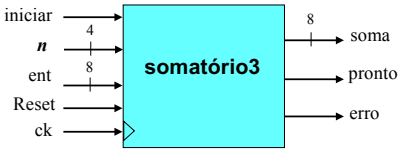
0. Início
1. AC ← 0; T← ent; cont ← *n*; pronto ← 0;
2. **Enquanto** cont ≠ 0 **faça {**
3. AC ← AC + T; T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E



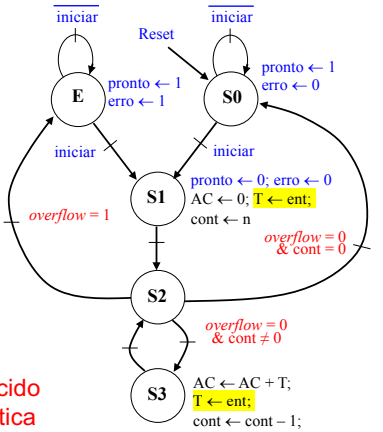
A variável que controla o laço precisa ser atualizada no corpo do laço. No corpo do laço também é feito o cálculo ” AC ← AC + T;” e um novo operando de entrada é lido.

Processadores Dedicados

Exemplo 3: Passo 1 (Captura do comportamento por meio de uma FSMD)



0. Início
1. AC ← 0; T← ent; cont ← *n*; pronto ← 0;
2. **Enquanto** cont ≠ 0 **faça {**
3. AC ← AC + T; T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E

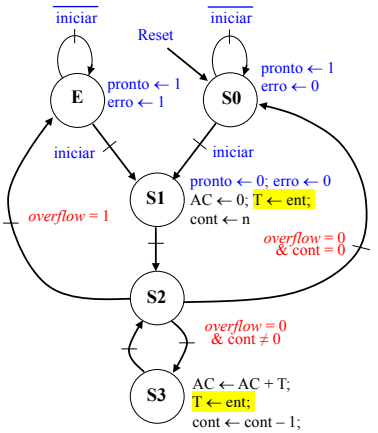


Para um correto sincronismo, um novo operando deve ser fornecido em "ent" a cada **2 ciclos** de relógio, a partir de S1. tal característica já foi prevista no próprio enunciado do exemplo.

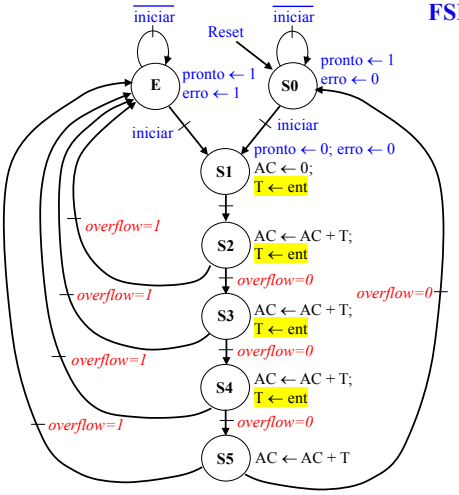
Processadores Dedicados

Comparação somatório3 x somatório1

FSMD



FSMD

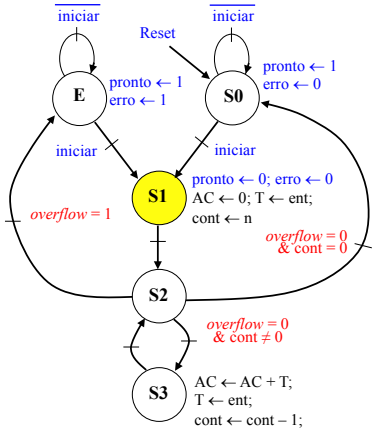


Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000



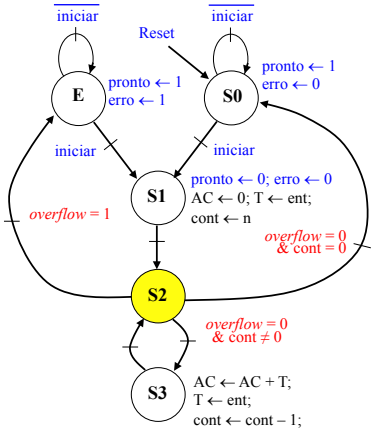
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000



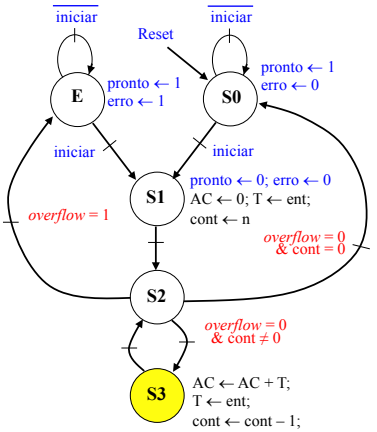
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111



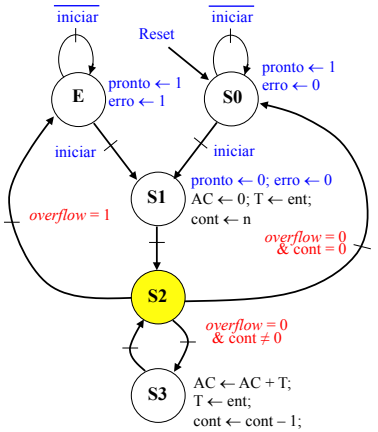
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111



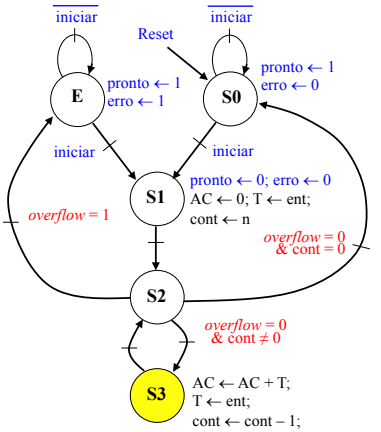
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110



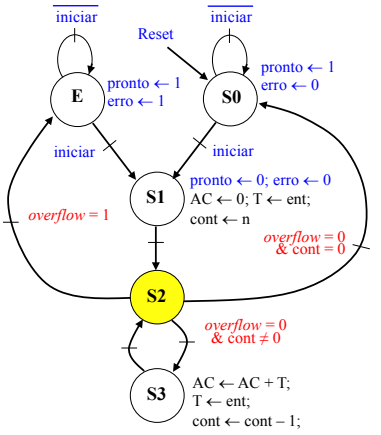
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110



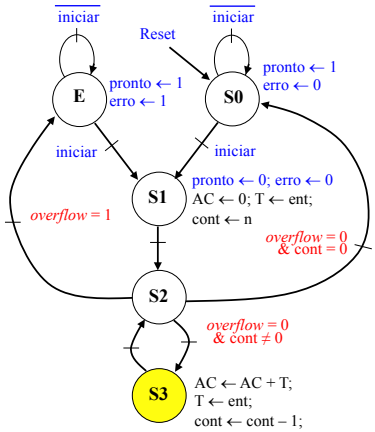
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101



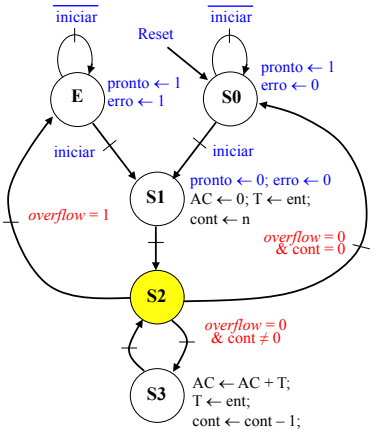
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101



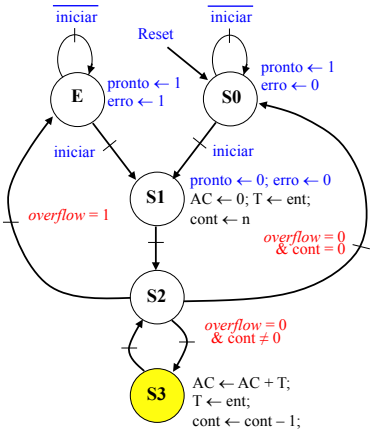
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100



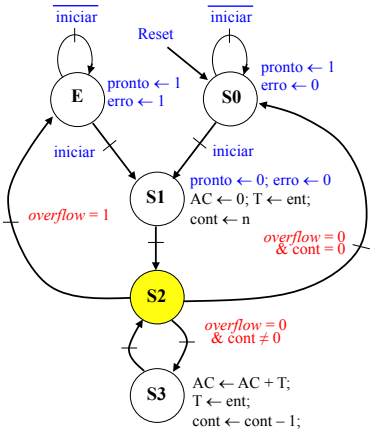
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100



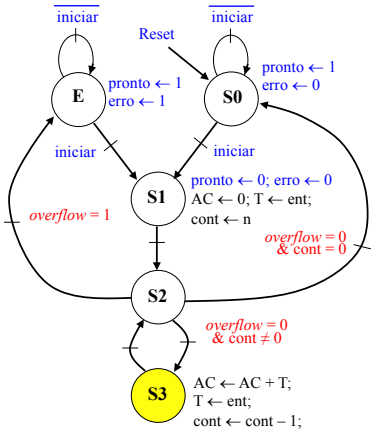
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011



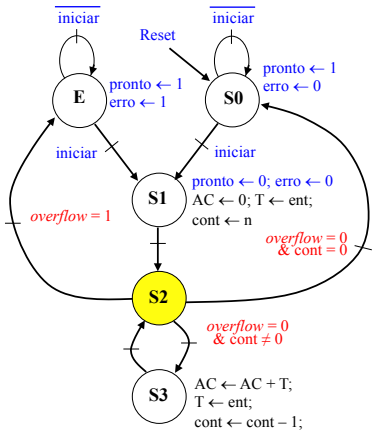
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011



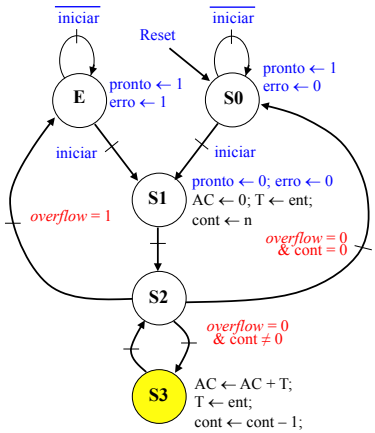
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010



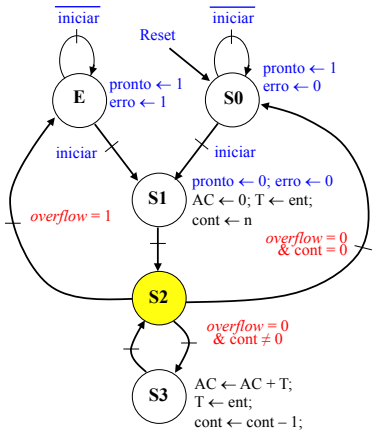
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010
S2 (7ª passagem)	0010



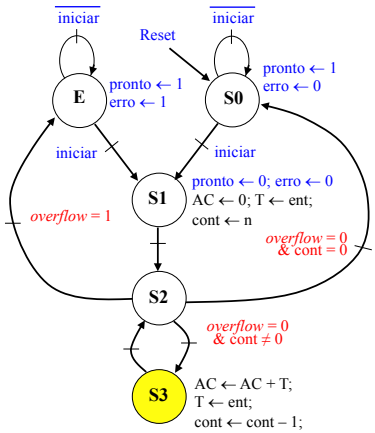
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010
S2 (7ª passagem)	0010
S3 (7ª passagem)	0001



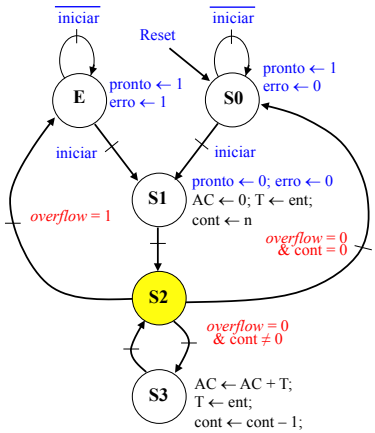
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010
S2 (7ª passagem)	0010
S3 (7ª passagem)	0001
S2 (8ª passagem)	0001



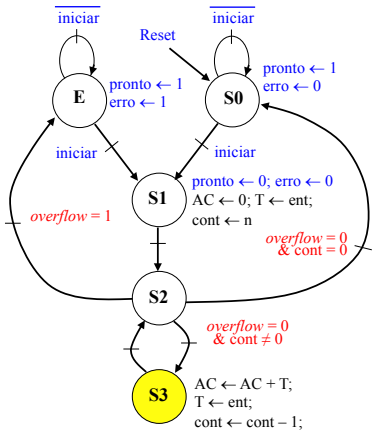
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010
S2 (7ª passagem)	0010
S3 (7ª passagem)	0001
S2 (8ª passagem)	0001
S3 (8ª passagem)	0000



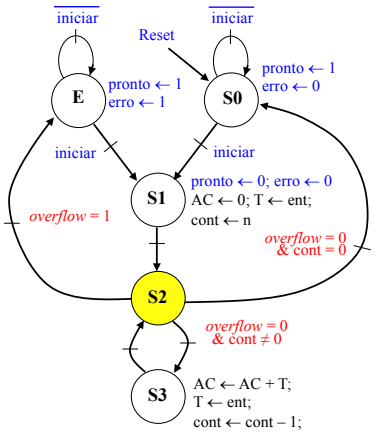
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8 (e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010
S2 (7ª passagem)	0010
S3 (7ª passagem)	0001
S2 (8ª passagem)	0001
S3 (8ª passagem)	0000
S2 (9ª passagem)	0000



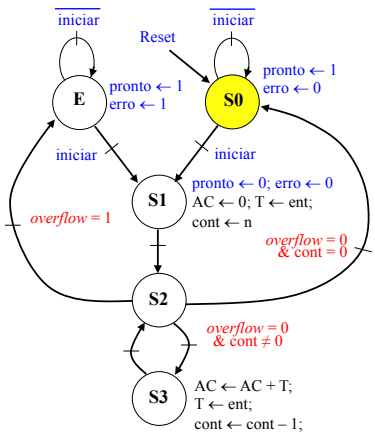
* Valor de **cont** ao sair do Estado

Processadores Dedicados

Exemplo 3:

Simulando a execução para n=8
(e supondo que não corra *overflow*)

Estado	cont*
S1	1000
S2 (1ª passagem)	1000
S3 (1ª passagem)	0111
S2 (2ª passagem)	0111
S3 (2ª passagem)	0110
S2 (3ª passagem)	0110
S3 (3ª passagem)	0101
S2 (4ª passagem)	0101
S3 (4ª passagem)	0100
S2 (5ª passagem)	0100
S3 (5ª passagem)	0011
S2 (6ª passagem)	0011
S3 (6ª passagem)	0010
S2 (7ª passagem)	0010
S3 (7ª passagem)	0001
S2 (8ª passagem)	0001
S3 (8ª passagem)	0000
S2 (9ª passagem)	0000
S0	0000



Note que **S3** (corpo do laço) foi executado **8 vezes**, mas **S2** (estado de teste) foi executado **9 vezes**.

Conclusão: o estado do teste sempre é executado uma vez mais do que o corpo do laço.

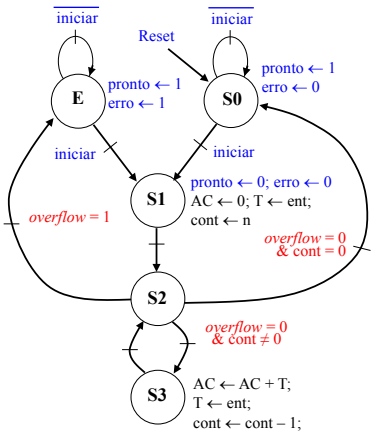
Processadores Dedicados

Exemplo 3: Passo 2 (Projeto do BO)

1ª questão para guiar o projeto do BO:
Quais são os sinais de interface do BO?

- ”n”, ”ent” e ”soma”

FSMD

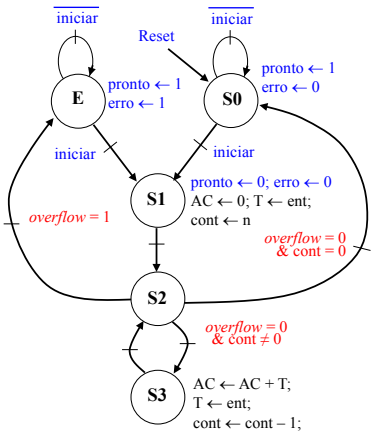


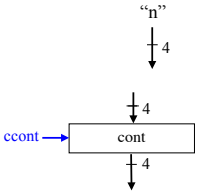
Processadores Dedicados

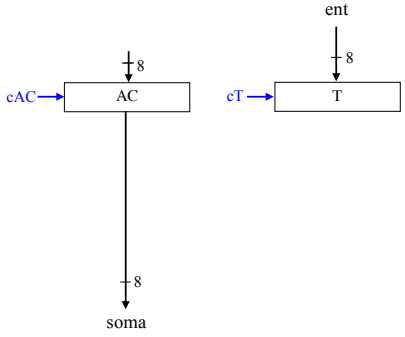
Exemplo 3: Passo 2 (Projeto do BO) 2ª questão para guiar o projeto do BO: Quais variáveis são usadas?

- Para armazenar dados: “AC” e “T”
- Para controle: ”cont” → novidade em relação ao Exemplo 1

FSMD







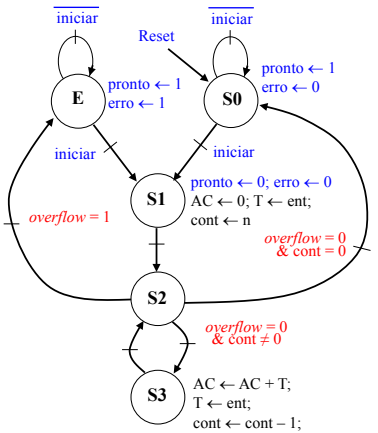
Processadores Dedicados

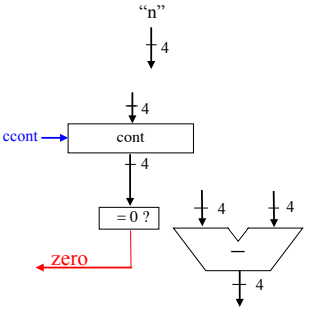
Exemplo 3: Passo 2 (Projeto do BO)

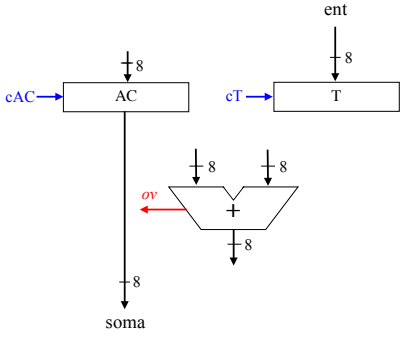
3ª questão para guiar o projeto do BO:
Quais operações são realizadas?

- Uma adição para números de 8 bits e o respectivo teste de *overflow*
- Uma subtração (para decrementar "cont")
- Um testador de zero para número de 4 bits

FSMD







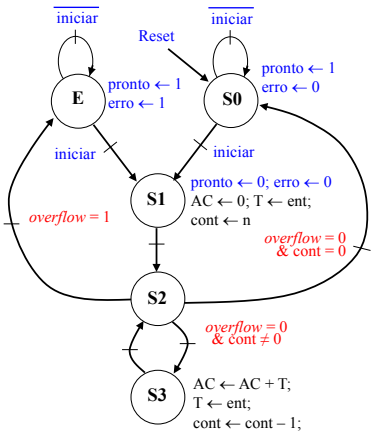
Processadores Dedicados

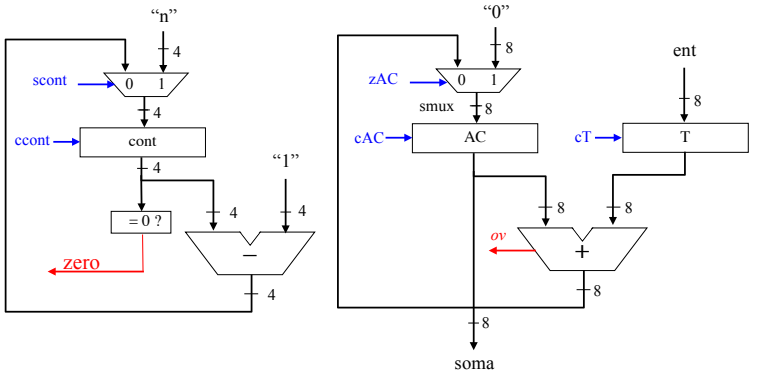
Exemplo 3: Passo 2 (Projeto do BO)

4ª questão para guiar o projeto do BO:
Quais conexões? Variáveis x operações

- T← ent;
- AC← 0; AC← AC+T; → mux2:1na entrada de AC
- cont ← n; cont ← cont – 1; → mux2:1 na entrada de cont

FSMD



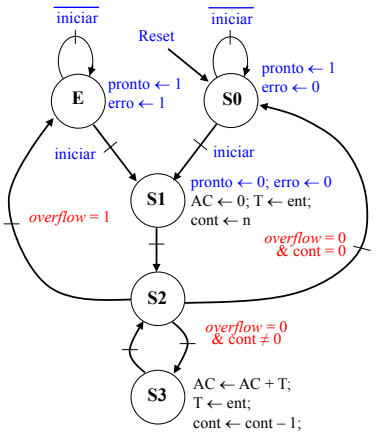


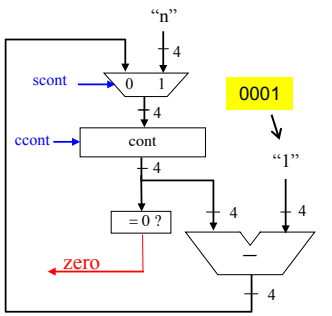
Processadores Dedicados

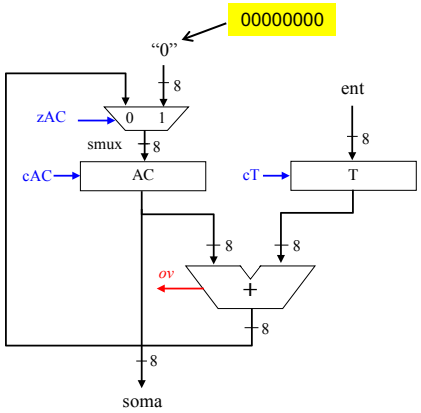
Exemplo 3: Passo 2 (Projeto do BO)

FSMD

Atenção para os valores das constantes

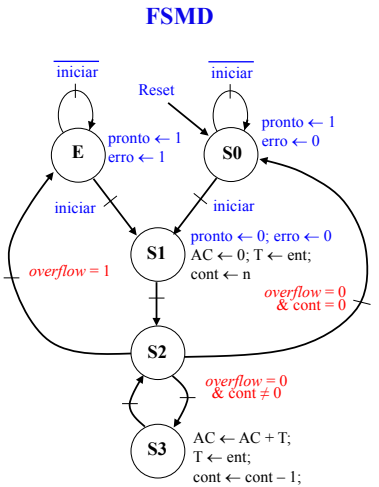




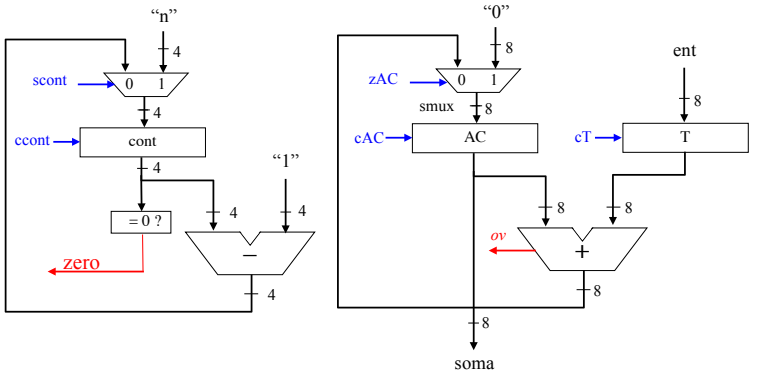


Processadores Dedicados

Exemplo 3: Passo 2 (Projeto do BO)

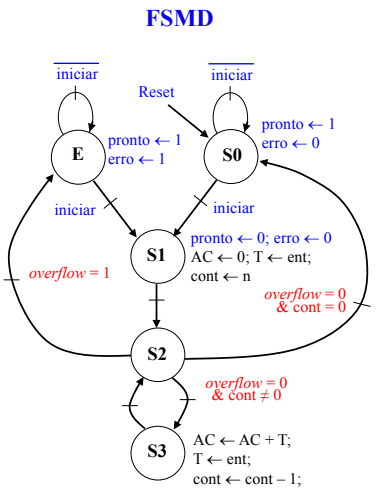


O sinal de relógio (ck) foi omitido para aumentar a clareza do esquemático.

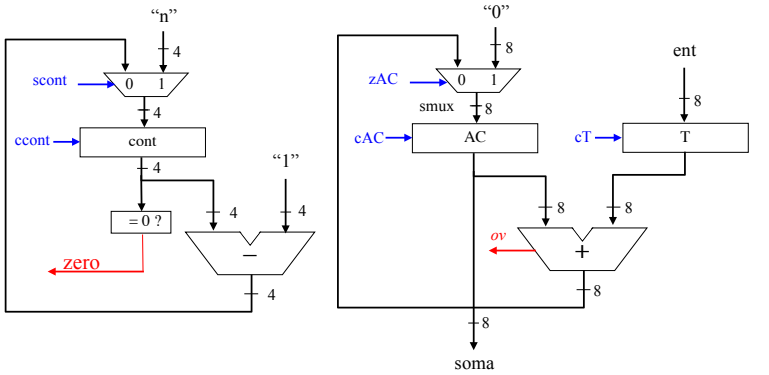


Processadores Dedicados

Exemplo 3: Passo 2 (Projeto do BO)



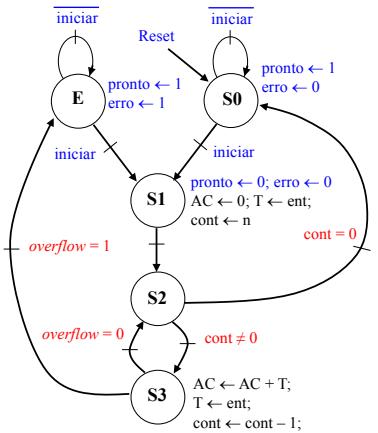
Problema: o sinal "*ov*" fica pronto em S3. Porém, ao voltar para S2 um novo valor para "*ov*" será gerado, em função dos novos valores de "AC" e de "T". Logo, o "*ov*" que interessa é aquele que foi gerado em S3.



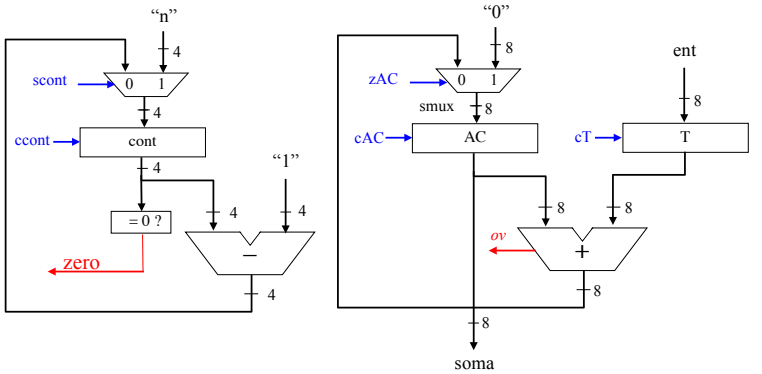
Processadores Dedicados

Exemplo 3: Passo 2 (Projeto do BO)

FSMD corrigida



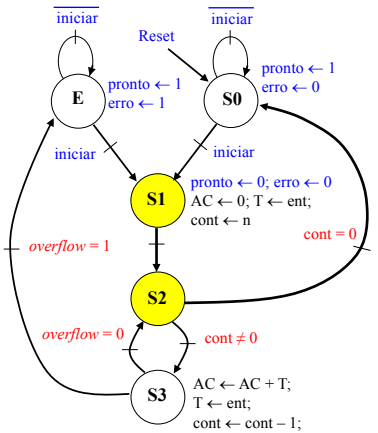
Solução: antecipar o teste do overflow para o estado S3, a fim de usar o valor apropriado de "*ov*".



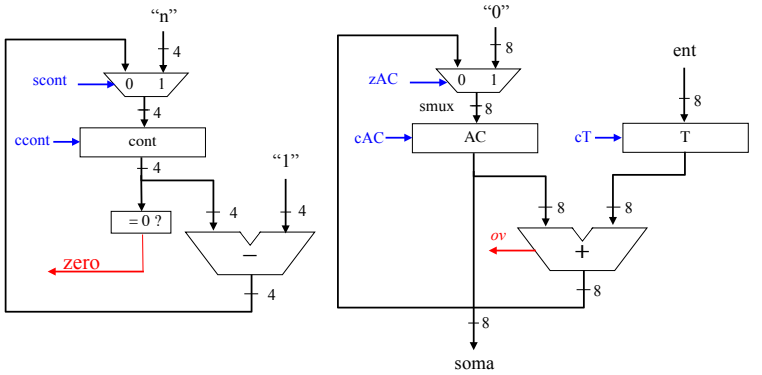
Processadores Dedicados

Exemplo 3: Passo 2 (Projeto do BO)

FSMD corrigida



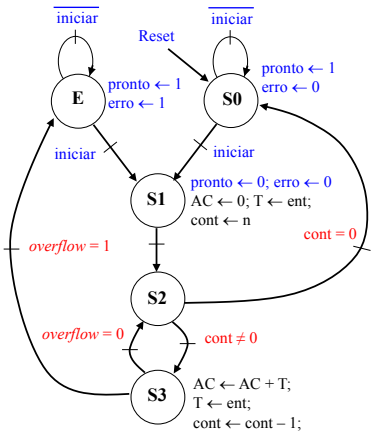
O que acontece se ***n*** =0? (Testando a robustez desta solução...)
Resp.: a execução termina em S0, com AC = 0



Processadores Dedicados

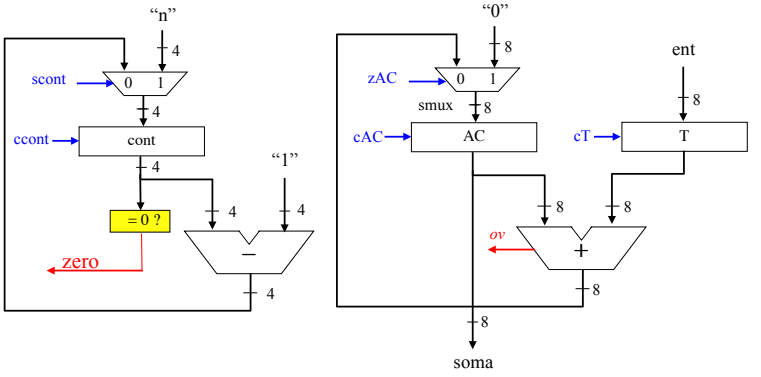
Exemplo 3: Passo 2 (Projeto do BO)

FSMD corrigida



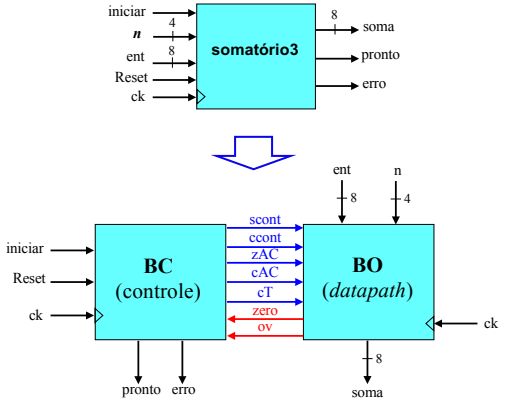
Tarefa 3:

1. Projetar um circuito combinacional para o comparador "= 0 ?"
2. Analisar o custo deste comparador (relacionar este custo com o custo de um *full adder* ...)



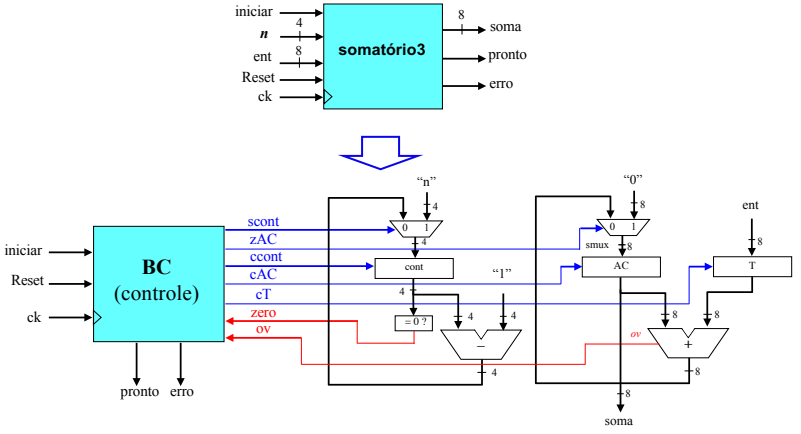
Processadores Dedicados

Exemplo 3: Passo 3 (Esboçando o Diagrama BO/BC)



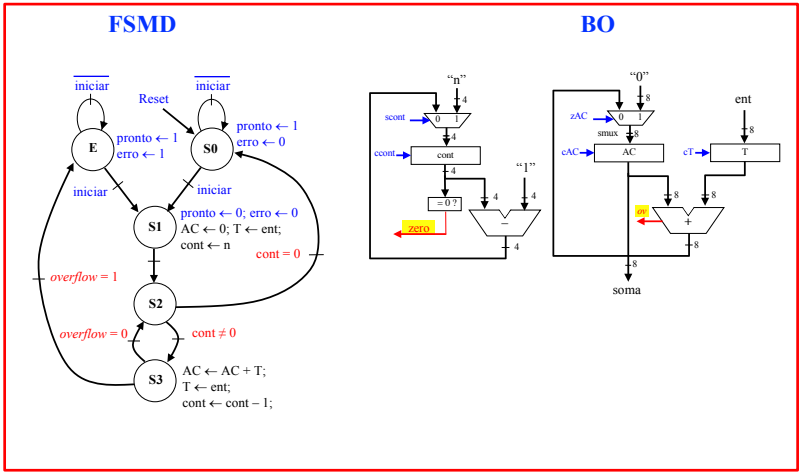
Processadores Dedicados

Exemplo 3: Passo 3 (Esboçando o Diagrama BO/BC)



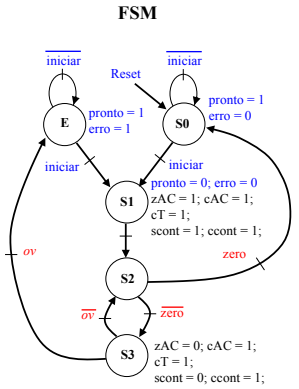
Processadores Dedicados

Exemplo 3: Passo 4 (Projeto do BC): Criando uma FSM a partir da FSMD e do BO



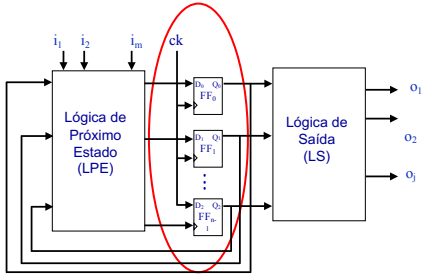
Processadores Dedicados

Exemplo 3: Passo 4 (Projeto do BC): Definindo o número de flip-flops



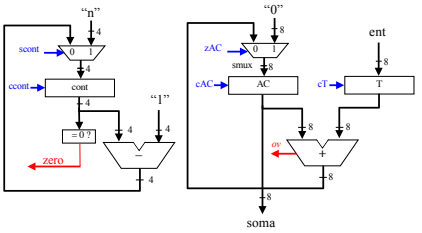
Quantos flip-flops são necessários para implementar a FSM?

Resp.: como são **5** estados (=**5** combinações), o cálculo do número de flip-flops é $\log_2 \mathbf{5} = 3$



Processadores Dedicados

Exemplo 3: Passo 4 (Projeto do BC): Mapeando as interfaces do BC para o modelo de FSM de Moore

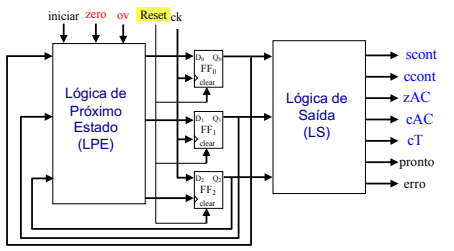


Os passos faltantes são similares aos Exemplos 1 e 2

Tarefa 4:

1. Completar os passos faltantes do Exemplo 3
2. Comparar (de maneira qualitativa) esta FSM com as dos Exemplos 1 e 2

Sugestão: tal como nos Exemplos 1 e 2, codificar o estado S0 como "000", uma vez que ele será o estado de Reset.



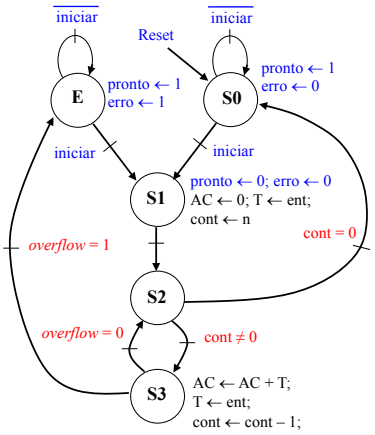
Processadores Dedicados

Exemplo 3: Estimativa de Desempenho

Tempo de Execução:

$$T_{exec} = \text{nº de ciclos x } T$$

Onde **T** é o período (mínimo) do relógio



Processadores Dedicados

Exemplo 3: Estimativa de Desempenho

Tempo de Execução:

$$T_{exec} = n\_ciclos \times T$$

- **n_ciclos** é o nº de ciclos de relógio, no pior caso, para concluir o cálculo
- **T** é o período (mínimo) do relógio

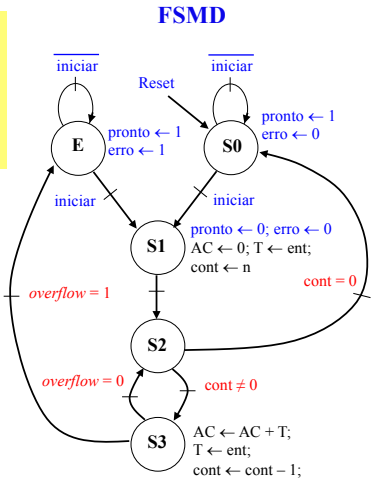
Cálculo do nº de ciclos:

- Para aprontar um cálculo de soma é necessário executar, na pior das hipóteses, a seguinte sequência de estados:

S1, n (S2, S3), S2, onde o valor máximo de n é 8

Logo, serão necessários 8x2+2 = 18 ciclos de relógio, no pior caso.

- Lembrando que os estados ”E” e ”S0” são desconsiderados, pois eles não realizam cálculos



Processadores Dedicados

Exemplo 3: Estimativa de Desempenho

Tempo de Execução:

$$T_{exec} = n\_ciclos \times T$$

- **n_ciclos** é o nº de ciclos de relógio, no pior caso, para concluir o cálculo
- **T** é o período (mínimo) do relógio

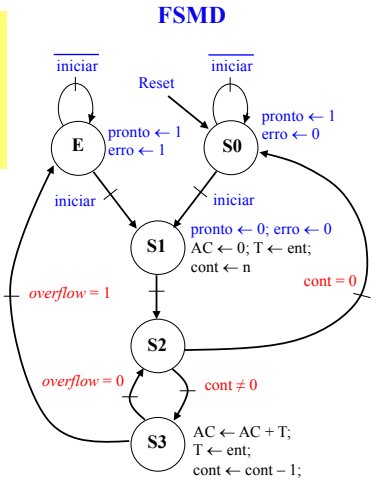
Cálculo do nº de ciclos:

- Para aprontar um cálculo de soma é necessário executar, na pior das hipóteses, a seguinte sequência de estados:

S1, n (S2, S3), S2, onde o valor máximo de n é 8

Logo, serão necessários 8x2+2 = 18 ciclos de relógio, no pior caso.

- Lembrando que os estados ”E” e ”S0” são desconsiderados, pois eles não realizam cálculos



Processadores Dedicados

Exemplo 3: Estimativa de Desempenho

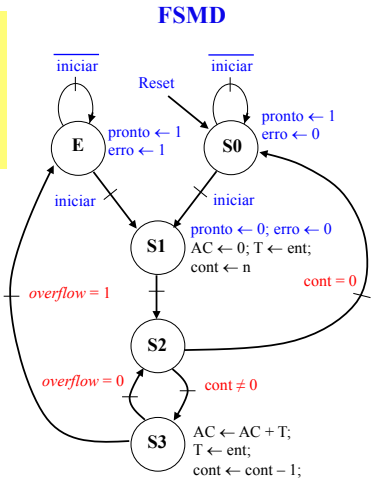
Tempo de Execução:

$$T_{exec} = n\_ciclos \times T$$

- **n_ciclos** é o nº de ciclos de relógio, no pior caso, para concluir o cálculo
- **T** é o período (mínimo) do relógio

Fica faltando a estimativa de T

Tarefa 5: Determinar T
Para tanto, realizar a análise de timing do BO de somatório3 assumindo os atrasos usados na análise de timing de somatório1



Processadores Dedicados

Exemplo 4: Enunciado

SD (bloco acelerador) para cálculo de um somatório de *n* números

Especificação geral: Necessita-se de um sistema digital (SD) dedicado (i.e., um bloco acelerador) capaz de realizar o somatório de ***n*** números inteiros sem sinal **A**, **B**, **C**, **D ...,**, representados em **binário com 8 bits**, onde $0 < n <= 8$, os quais estão armazenados em uma memória externa "Mem".

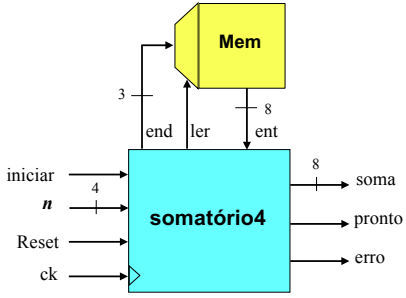
* Os números fornecidos como entrada do sistema são comumente chamados de "operandos (de entrada)".

Processadores Dedicados

Exemplo 4: Enunciado

SD (bloco acelerador) para cálculo de um somatório de *n* números

Especificação das interfaces: Este sistema digital, doravante denominado de "somatório4", possui uma entrada de relógio ("ck"), uma entrada de reset assíncrono ("Reset"), **uma** entrada de dados com 8 bits ("ent"), uma entrada para receber o valor ***n***, uma entrada de controle denominada "iniciar", duas saídas de controle ("pronto" e "erro") e uma saída de dados de 8 bits ("soma"). Além disso, ele possui uma saída de 3 bits chamada "end" para selecionar o endereço da memória onde se encontra cada um dos operandos e um sinal "ler" para ativar a leitura da memória.



* Os números fornecidos como entrada do sistema são comumente chamados de "operandos (de entrada)".

Processadores Dedicados

Exemplo 4: Enunciado

SD (bloco acelerador) para cálculo de um somatório de n números

Especificação do comportamento:

- Há dois estados iniciais, **S0** e **E**. Enquanto um novo cálculo não inicia, "somatório4" permanece em um destes dois estados, conforme tenha terminado o cálculo anterior.
- O sinal externo "iniciar" dá o comando para iniciar um cálculo do somatório.
- À medida que o cálculo é realizado, "somatório4" vai lendo da memória "Mem" os valores dos operandos A, B, C, D ... no momento adequado. Cada valor lido entra em "somatório4" pela entrada "ent".
- Assuma que o valor ***n*** é fornecido pela entrada de mesmo nome no primeiro estado do cálculo, i.e., S1.
- Uma vez iniciado, o cálculo é realizado de maneira sequencial e cumulativa (i.e., cada novo operando de entrada que chega é somado ao valor acumulado até então).
- Caso ocorra *overflow* em alguma das adições, o cálculo deve terminar imediatamente, com o SD parando no estado **E**. Caso não ocorra *overflow*, o cálculo termina com o SD parando no estado **S0**.

Processadores Dedicados

Exemplo 4: Enunciado

SD (bloco acelerador) para cálculo de um somatório de n números

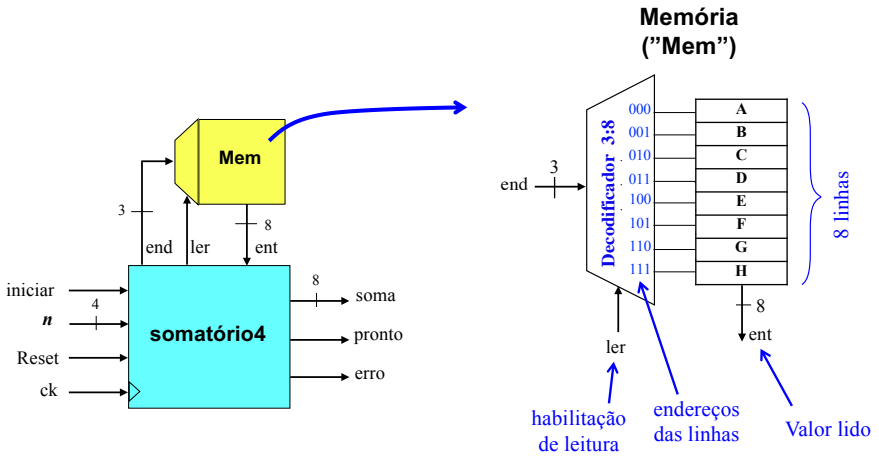
Especificação do comportamento da Memória ”Mem”:

- Considere que antes que o sinal ”iniciar” valha ”1”, os ***n*** números inteiros sem sinal **A**, **B**, **C**, **D...** Tenham sido armazenados em Mem*, um número por linha, a partir da linha cujo endereço é zero (i.e., end = 000).
- A leitura de uma linha de “Mem” é assíncrona: deve-se manter o sinal “end” estável e o sinal **ler**=1 durante um ciclo de relógio, de modo que o valor armazenado na linha de endereço ”end” fique disponível nos fios rotulados como ”ent” ainda no mesmo ciclo de relógio. (Ou seja, deseja-se que uma leitura da ”Mem” seja completada em um ciclo de relógio.)
- No nível RT representaremos por **Mem[end]** o acesso à linha de ”Mem” cujo endereço ”end”.

* Por uma questão de didática, os sinais para realizar escritas em Mem foram omitidos.

Processadores Dedicados

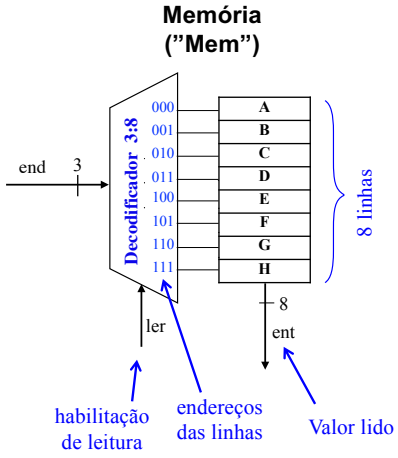
Exemplo 4: Estrutura da Memória ”Mem”



Processadores Dedicados

Exemplo 4: passos para um acesso de leitura em ”Mem”

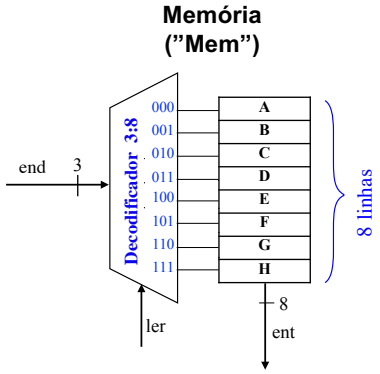
- A leitura de uma linha de “Mem” é assíncrona. Isto significa que ela independe do sinal de relógio (ck)
- O enunciado deste exemplo dá a entender que se deseja que a leitura de uma linha ocorra dentro de um ciclo de relógio



Processadores Dedicados

Exemplo 4: passos para um acesso de leitura em ”Mem”

Suponha que se deseje ler o conteúdo da linha cujo **endereço é 2** (010 em binário) no estado S3 (p. ex.):

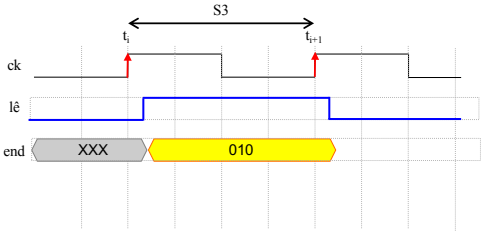


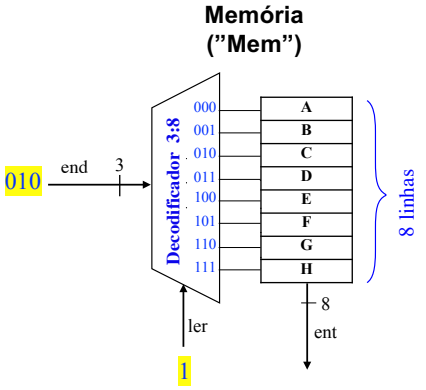
Processadores Dedicados

Exemplo 4: passos para um acesso de leitura em ”Mem”

Suponha que se deseje ler o conteúdo da linha cujo **endereço é 2** (010 em binário) no estado S3 (p. ex.):

1. Manter **ler**=1 e end=010 estáveis durante S3;



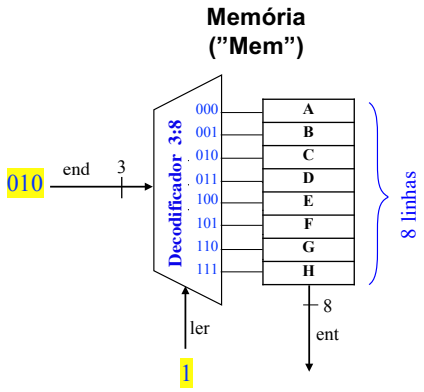


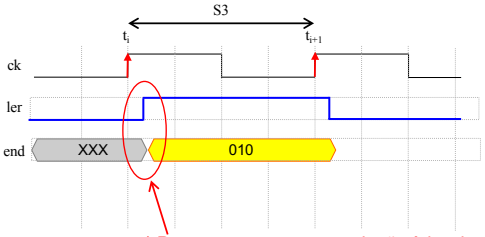
Processadores Dedicados

Exemplo 4: passos para um acesso de leitura em ”Mem”

Suponha que se deseje ler o conteúdo da linha cujo **endereço é 2** (010 em binário) no estado S3 (p. ex.):

1. Manter **ler**=1 e end=010 estáveis durante S3*;





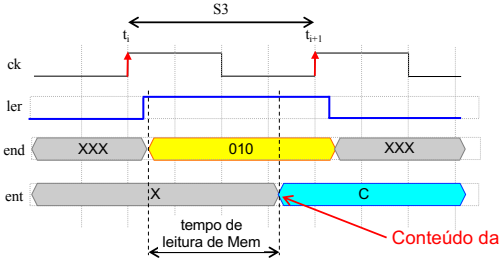
* Pequenos atrasos em relação à borda t_i são toleráveis, desde que o tempo de leitura caiba no período de ck

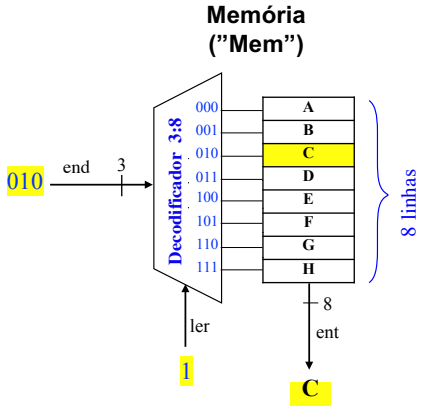
Processadores Dedicados

Exemplo 4: passos para um acesso de leitura em ”Mem”

Suponha que se deseje ler o conteúdo da linha cujo **endereço é 2** (010 em binário) no estado S3 (p. ex.):

1. Manter **ler**=1 e end=010 estáveis durante S3;
2. Ao final de S3, o valor armazenado no endereço 010 estará disponível na saída ”ent”, podendo ser carregado em um registrador de somatório4.

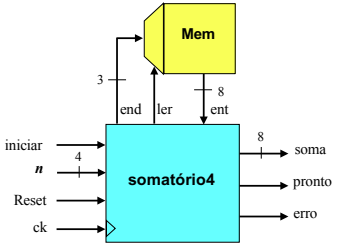




Conteúdo da linha cujo endereço é 010 está disponível na saída ”ent”

Processadores Dedicados

Exemplo 4: Passo 1 (Captura do comportamento por meio de uma FSMD)



0. Início
1. AC ← 0; cont ← *n*; i ← 0; end ← 0; pronto ← 0;
2. **Enquanto** i < cont **faça {**
3. T ← Mem[end]
4. AC ← AC + T; end ← end +1; i ← i +1;

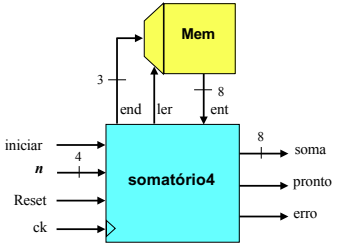
}

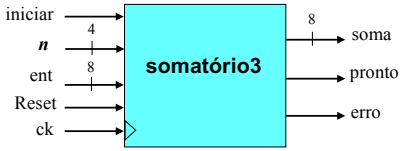
4. pronto ← 1; //será mapeado para S0 e E

A FSMD de ”somatório4” é semelhante à de ”somatório3”, com algumas diferenças devido à necessidade de acessar os operandos de entrada na memória ”Mem”.

Processadores Dedicados

Comparação somatório4 x somatório3



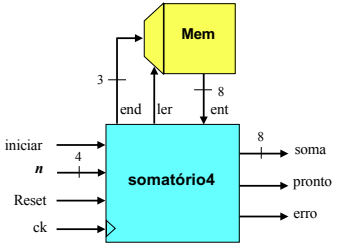



```
0. Início
1. AC ← 0; cont ← n; i ← 0; end ← 0; pronto ← 0;
2. Enquanto  i < cont  faça {
3.    T ←  Mem[end]
4.    AC ← AC + T; end ← end +1; i ← i +1;
}
4. pronto ← 1; //será mapeado para S0 e E
```

```
0. Início
1. AC ← 0; T← ent; cont ← n; pronto ← 0;
2. Enquanto  cont ≠ 0) faça {
3.     AC ← AC + T;   T← ent; cont ← cont – 1;
}
4. pronto ← 1; //esta linha ocorrerá em S0 e em E
```

Processadores Dedicados

Exemplo 4: Passo 1 (Captura do comportamento por meio de uma FSMD)

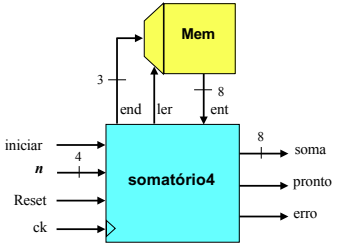


0. Início
1. AC ← 0; cont ← *n*; i ← 0; end ← 0; pronto ← 0;
2. **Enquanto** i < cont **faça {**
3. T ← Mem[end]
4. AC ← AC + T; end ← end +1; i ← i +1;
}
4. pronto ← 1; //será mapeado para S0 e E

- É preciso ler cada operando da memória "Mem"
- A leitura de cada operando requer um ciclo de relógio e portanto, a FSMD deve ter um estado específico para isso

Processadores Dedicados

Exemplo 4: Passo 1 (Captura do comportamento por meio de uma FSMD)

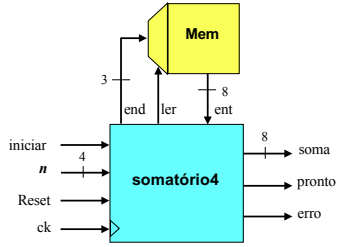



```
0. Início
1. AC ← 0; cont ← n; i ← 0; end ← 0; pronto ← 0;
2. Enquanto  i < cont  faça {
3.    T ←  Mem[end]
4.    AC ← AC + T; end ← end +1; i ← i +1;
}
4. pronto ← 1; //será mapeado para S0 e E
```

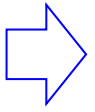
- A variável ”end” serve para indexar o acesso à memória ”Mem”.
- A variável ”i” serve para controlar o número e vezes que o laço precisa ser feito. Este número é igual a ***n***
- ”i” parte do valor zero e é incrementado. Outra possibilidade seria i partir do valor ***n*** e ser decrementado.

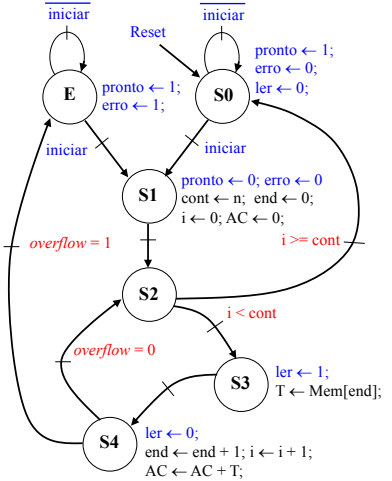
Processadores Dedicados

Exemplo 4: Passo 1 (Captura do comportamento por meio de uma FSMD)



0. Início
1. AC ← 0; cont ← *n*; i ← 0; end ← 0; pronto ← 0;
2. **Enquanto** i < cont **faça {**
3. T ← Mem[end];
4. AC ← AC + T; end ← end +1; i ← i +1;
}
4. pronto ← 1; //será mapeado para S0 e E

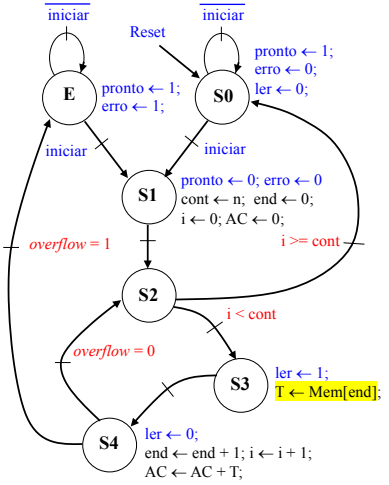




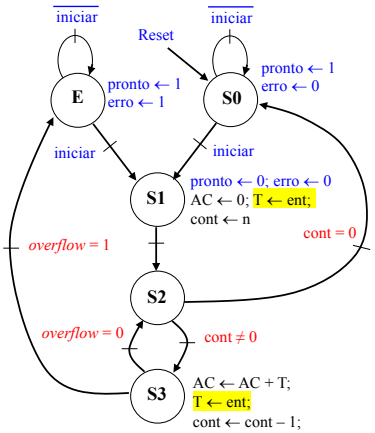
Processadores Dedicados

Comparação somatório4 x somatório3

FSMD



FSMD

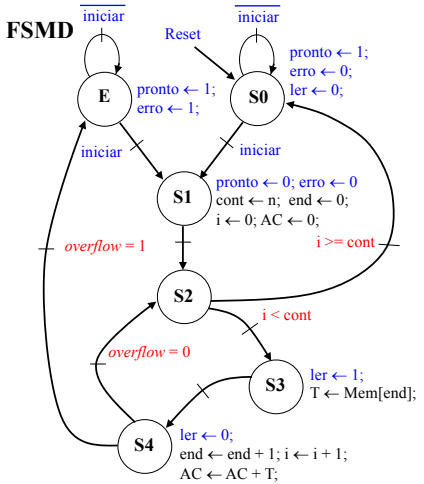


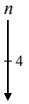
Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO)

1ª questão para guiar o projeto do BO:
Quais são os sinais de interface do BO?

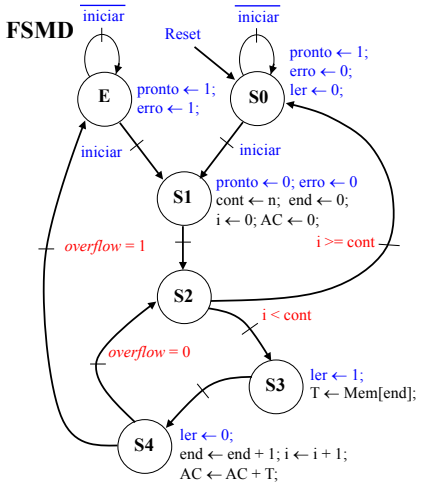
- ”n”, ”ent”, ”end” e ”soma”





Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO)

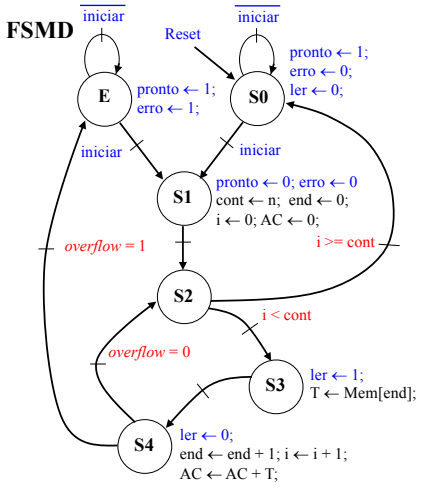


2ª questão para guiar o projeto do BO:
Quais variáveis são usadas?

- Para armazenar dados: “AC” e “T”
- Para controle: ”cont”, ”**i”, ”end”**

Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO)



2ª questão para guiar o projeto do BO:
Quais variáveis são usadas?

- Para armazenar dados: "AC" e "T"
- Para controle: "cont", "**i**", "**end**"

Examinando a relação entre **i** e **end** . Suponha **n**=8:

Passagem por S2	**cont (=n)**	**i**	**end**	Mem[**end**] ← leitura de Mem
1ª	1000	0000	000	Mem[000]
2ª	1000	0001	001	Mem[001]
3ª	1000	0010	010	Mem[010]
4ª	1000	0011	011	Mem[011]
5ª	1000	0100	100	Mem[100]
6ª	1000	0101	101	Mem[101]
7ª	1000	0110	110	Mem[110]
8ª	1000	0111	111	Mem[111]
9ª (e sai do laço)	1000	1000	---	Não ler Mem

n=8: valores armazenados na memória (nº máx.)

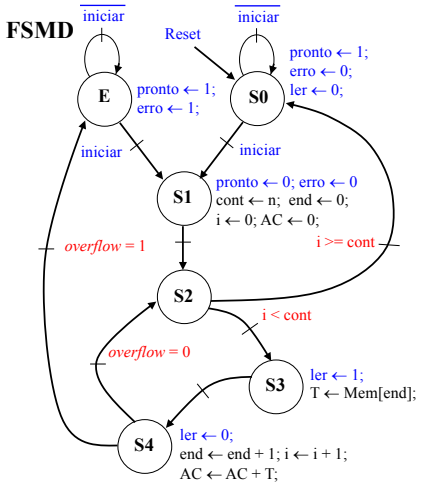
Conclusão: a variável **end** corresponde aos **3 bits** menos significativos da variável **i** e portanto, ambas podem ser implementadas pelo mesmo registrador

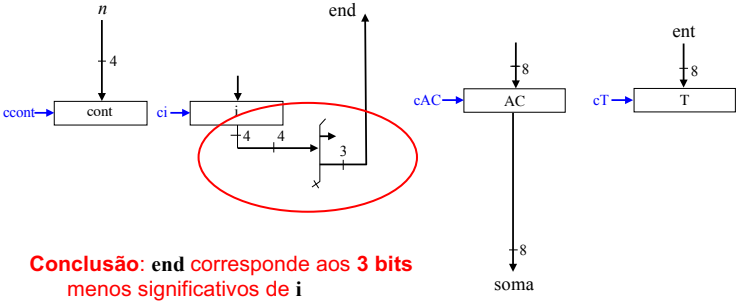
Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO)

2ª questão para guiar o projeto do BO:
Quais variáveis são usadas?

- Para armazenar dados: "AC" e "T"
- Para controle: "cont", "**i" (que irá conter "end")**

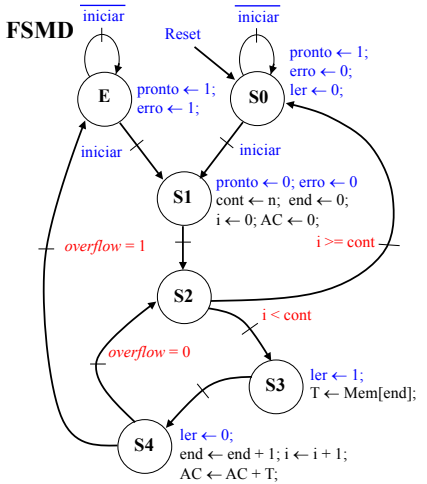




Conclusão: **end** corresponde aos **3 bits** menos significativos de **i**

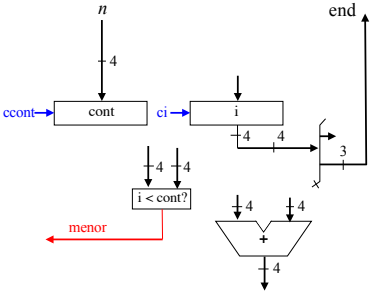
Processadores Dedicados

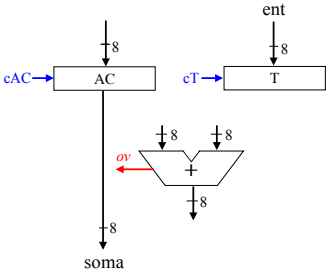
Exemplo 4: Passo 2 (Projeto do BO)



3ª questão para guiar o projeto do BO:
Quais operações são realizadas?

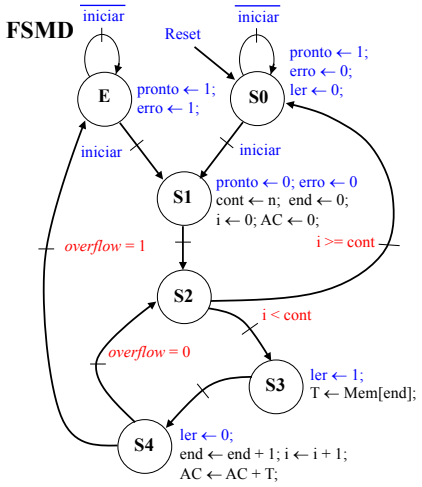
- Uma adição para números de 8 bits e o respectivo teste de *overflow*
- Uma adição de 4 bits (sem *overflow*)
- Uma comparação entre dois números de 4 bits





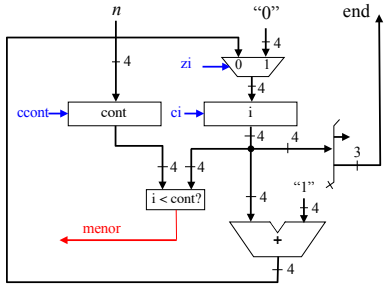
Processadores Dedicados

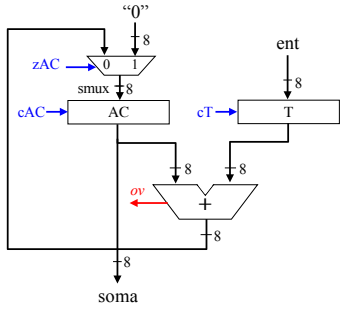
Exemplo 4: Passo 2 (Projeto do BO)



4ª questão para guiar o projeto do BO:
Quais conexões? Variáveis x operações

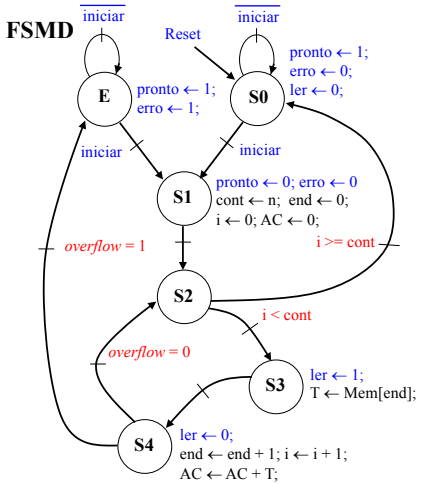
- T← ent;
- AC← 0; AC← AC+T; → mux2:1 na entrada de AC
- cont ← n;
- i ← 0; i ← i + 1; → mux2:1 na entrada de i
- Teste: i < cont



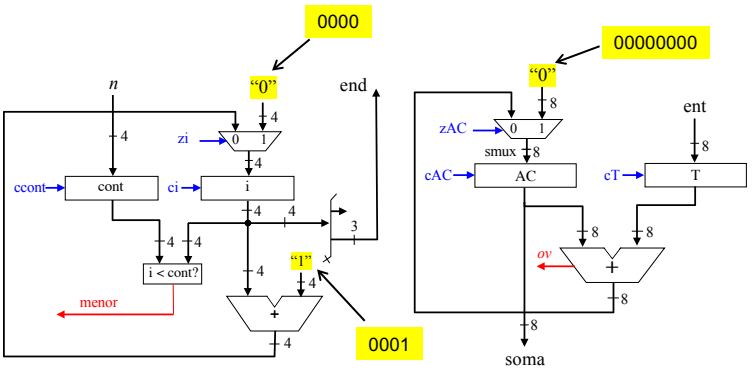


Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO)

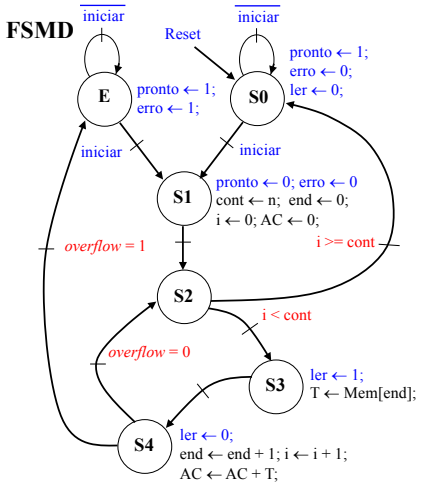


Atenção para os valores das constantes

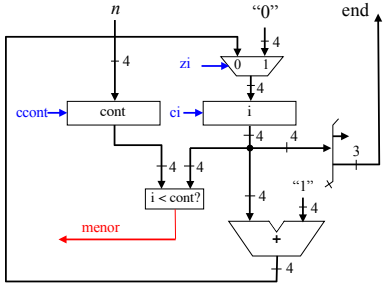


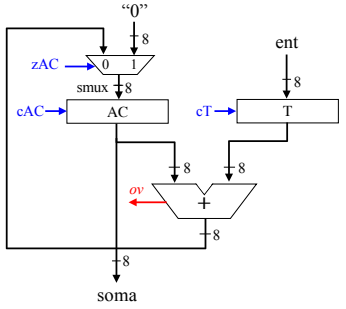
Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO)



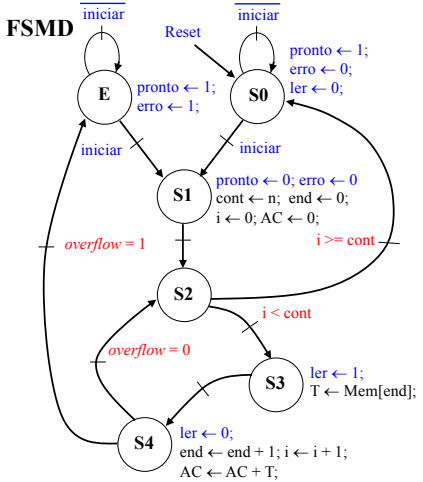
O sinal de relógio (ck) foi omitido para aumentar a clareza do esquemático.

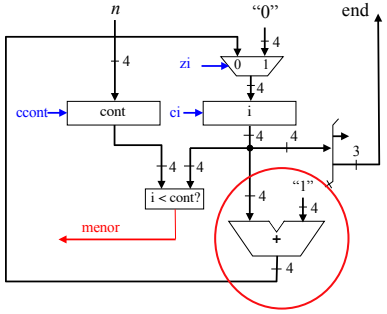


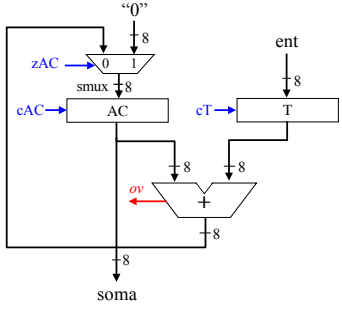


Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO) Otimizando o somador que incrementa o **i**

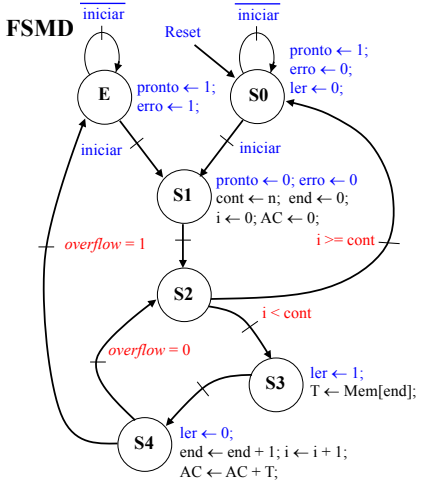






Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO) Otimizando o somador que incrementa o **i**

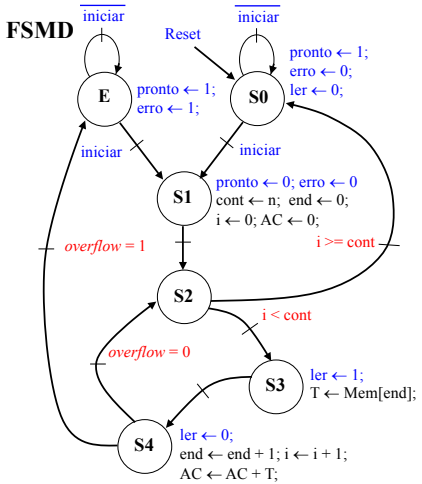


- O maior valor que a variável **i** pode assumir após o cálculo **i ← i + 1;** é 1000 (8)
- 1000 (8) é o único valor que possui o bit mais significativo em 1
- Conclusão: podemos usar o C_{out} do somador para obter o "1" mais significativo, e este somador pode ter apenas 3 bits (economizaremos 1 *full adder*)

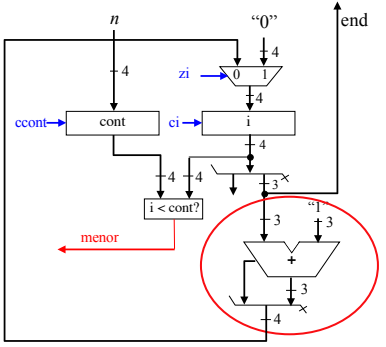
Passagem por S2	**cont (=n)**	**i**	**end**	Mem[**end**]
1ª	1000	0000	000	Mem[000]
2ª	1000	0001	001	Mem[001]
3ª	1000	0010	010	Mem[010]
4ª	1000	0011	011	Mem[011]
5ª	1000	0100	100	Mem[100]
6ª	1000	0101	101	Mem[101]
7ª	1000	0110	110	Mem[110]
8ª	1000	0111	111	Mem[111]
9ª (e sai do laço)	1000	1000	---	Não ler Mem

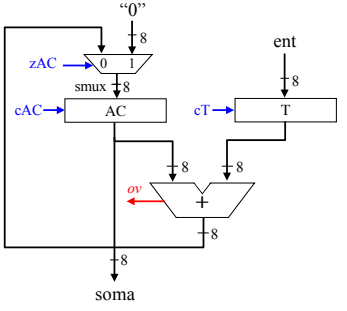
Processadores Dedicados

Exemplo 4: Passo 2 (Projeto do BO) Otimizando o somador que incrementa o **i**



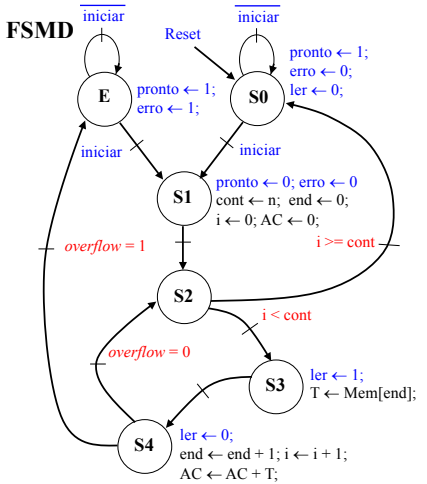
Conclusão: podemos usar o C_{out} do somador para obter o "1" mais significativo, e este somador pode ter apenas 3 bits (economizaremos 1 *full adder*)





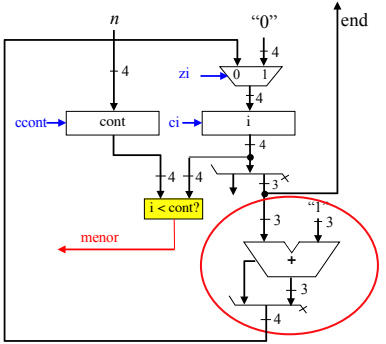
Processadores Dedicados

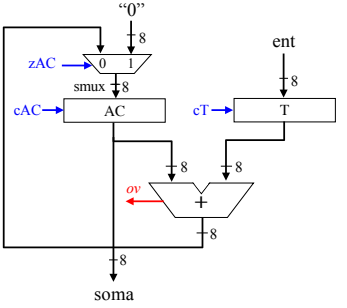
Exemplo 4: Passo 2 (Projeto do BO)



Tarefa 6:

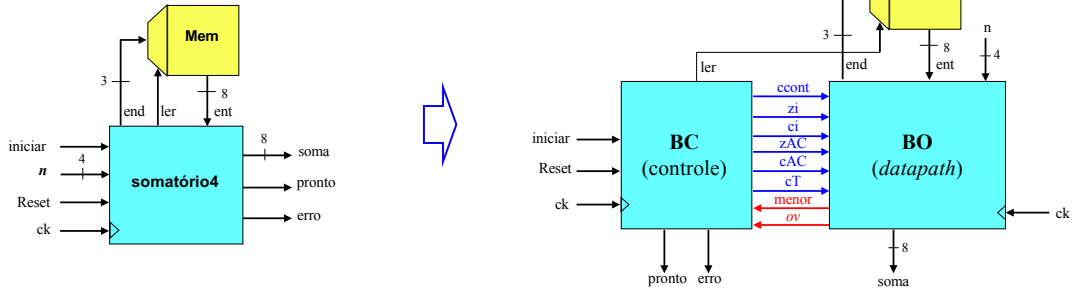
1. Projetar um circuito combinacional para o comparador "i < cont?"
2. Analisar o custo deste comparador (relacionar este custo com o custo de um *full adder* ...)





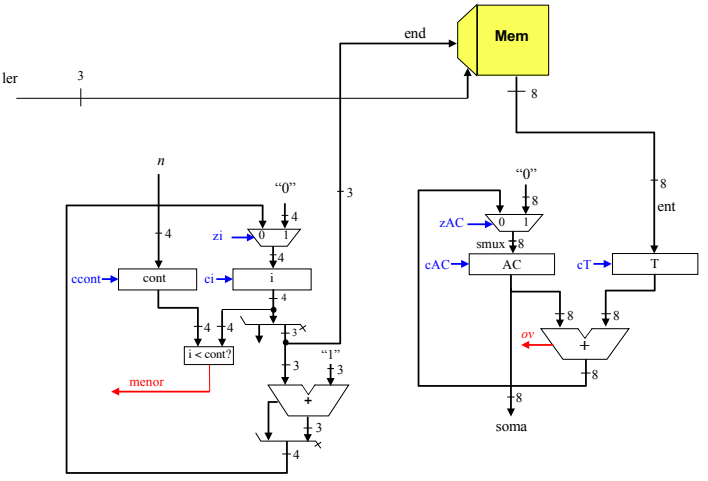
Processadores Dedicados

Exemplo 4: Passo 3 (Esboçando o Diagrama BO/BC)



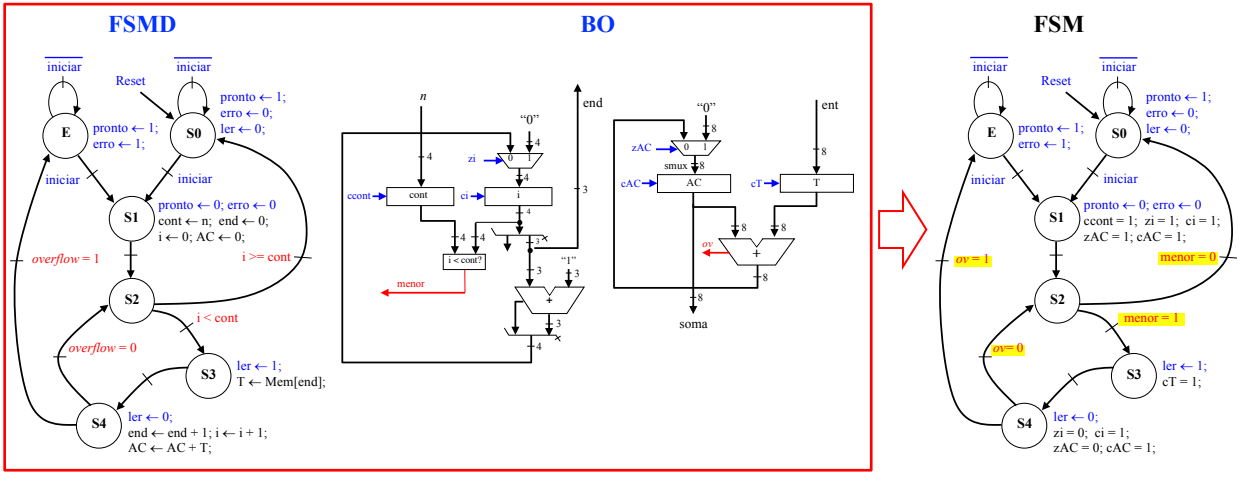
Processadores Dedicados

Exemplo 4: Passo 3 (Conexões com a Memória "Mem")



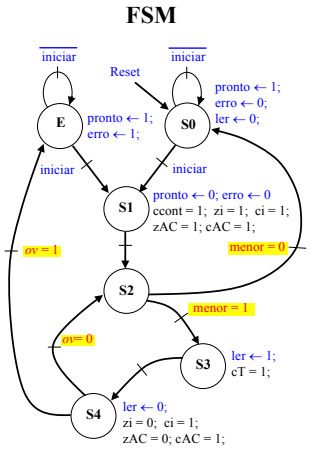
Processadores Dedicados

Exemplo 4: Passo 4 (Projeto do BC): Criando uma FSM a partir da FSMD e do BO



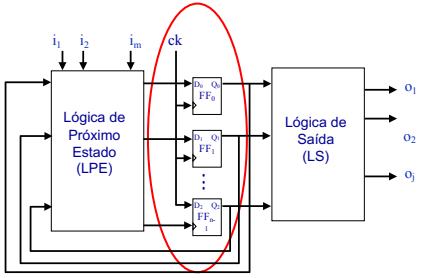
Processadores Dedicados

Exemplo 4: Passo 4 (Projeto do BC): Definindo o número de flip-flops



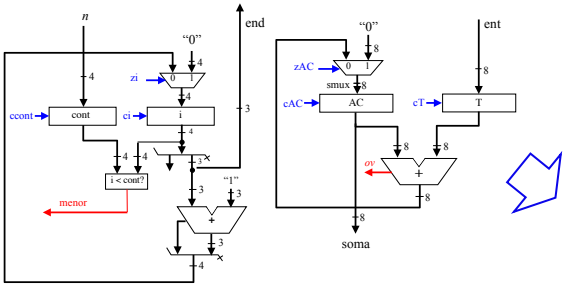
Quantos flip-flops são necessários para implementar a FSM?

Resp.: como são **6** estados (=**6** combinações), o cálculo do número de flip-flops é $\log_2$ **6** = 3



Processadores Dedicados

Exemplo 4: Passo 4 (Projeto do BC): Mapeando as interfaces do BC para o modelo de FSM de Moore

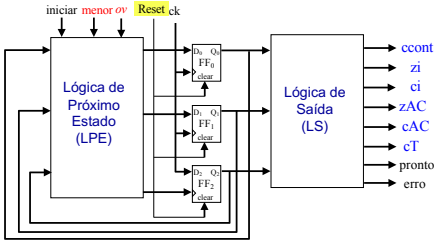


Os passos faltantes são similares aos demais Exemplos

Tarefa 7:

1. Completar os passos faltantes do Exemplo 4
2. Comparar (de maneira qualitativa) esta FSM com as dos Exemplos 1 e 3

Sugestão: tal como nos Exemplos anteriores, codificar o estado S0 como "000", uma vez que ele será o estado de Reset.



Processadores Dedicados

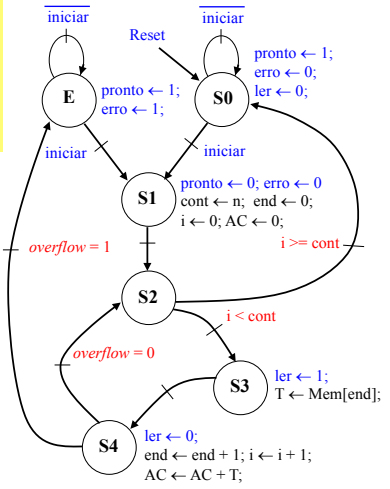
Exemplo 4: Estimativa de Desempenho

Tempo de Execução:

$$T_{exec} = n\_ciclos \times T$$

- **n_ciclos** é o nº de ciclos de relógio, no pior caso, para concluir o cálculo
- **T** é o período (mínimo) do relógio

FSMD



Processadores Dedicados

Exemplo 4: Estimativa de Desempenho

Tempo de Execução:

$$T_{exec} = n\_ciclos \times T$$

- **n_ciclos** é o nº de ciclos de relógio, no pior caso, para concluir o cálculo
- **T** é o período (mínimo) do relógio

Cálculo do nº de ciclos:

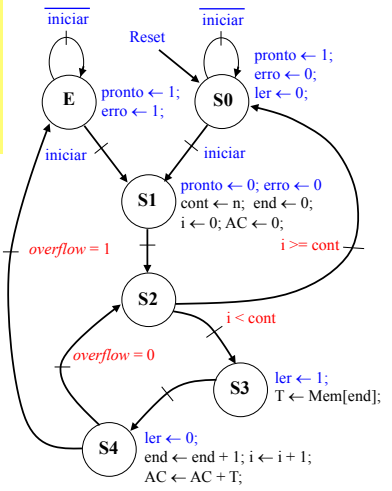
- Para aprontar um cálculo de soma é necessário executar, na pior das hipóteses, a seguinte sequência de estados:

S1, n (S2, S3, S4), S2, onde o valor máximo de n é 8

Logo, serão necessários 8x3+2 = 26 ciclos de relógio, no pior caso.

- Lembrando que os estados ”E” e ”S0” são desconsiderados, pois eles não realizam cálculos

FSMD



Processadores Dedicados

Exemplo 4: Estimativa de Desempenho

Tempo de Execução:

$$T_{exec} = n\_ciclos \times T$$

- **n_ciclos** é o nº de ciclos de relógio, no pior caso, para concluir o cálculo
- **T** é o período (mínimo) do relógio

Fica faltando a estimativa de T

Tarefa 8: Determinar T
Para tanto, realizar a análise de timing do BO de somatório4 assumindo os atrasos usados na análise de timing de somatório1

FSMD