Universidade Federal de Santa Catarina

Centro Tecnológico

Departamento de Informática e Estatística

Curso de Graduação em Ciências da Computação

Sistemas Digitais

INE 5406

Aula 10-T

4. Projeto de Sistemas Digitais no Nível RT. Aumentando o Nível de Abstração. Estudo de caso e Exploração do Espaço de Soluções: multiplicador por somas sucessivas (sol.1- custo mínimo).

Prof. José Luís Güntzel

guntzel@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5406/ine5406.html

4. Projeto de Sistemas Digitais no Nível RT

Aumentando o Nível de Abstração

Maior abstração permite:

- Tratar problemas mais complexos
- Explorar o espaço de soluções possível

Aplicação:

- Quando o problema que se deseja solucionar pode ser apresentado como um algoritmo

Formas de representação:

- Fluxograma
- Trecho de código em pseudocódigo ou em alguma linguagem (ex.: C, C++, Java, SystemC)

4. Projeto de Sistemas Digitais no Nível RT

Aspectos a Serem Considerados no Projeto

1. Custo de Implementação (Fabricação)
2. Desempenho
3. Consumo de Energia
4. Testabilidade
5. Tolerância (ou Robustez) a Falhas

A otimização simultânea destas variáveis é difícil, pois muitas são conflitantes entre si. Vejamos o porquê.

4. Projeto de Sistemas Digitais no Nível RT

Aspectos a Serem Considerados no Projeto

1. Custo de Implementação (Fabricação)

- Depende do número de transistores, quantidade de conexões, número de pinos de E/S e tipo de encapsulamento.
- Área do chip: quanto maior a área, menor o rendimento do processo de fabricação ("*yield*").

4. Projeto de Sistemas Digitais no Nível RT

Aspectos a Serem Considerados no Projeto

2. Desempenho

- O atraso crítico determina a máxima frequência de funcionamento.
- Para atingir uma meta de desempenho o projetista pode:
  - Escolher uma tecnologia de fabricação (CMOS) mais recente, com transistores menores (e portanto, mais cara).
  - Otimizar o projeto elétrico e/ou lógico.
  - Mudar a arquitetura do sistema, aumentando/inserindo paralelismo
  - Alterar o algoritmo, aumentando o grau de paralelismo.

Aspectos a Serem Considerados no Projeto

3. Consumo de Energia

- Importantíssimo para aplicações portáteis, pois determina a duração da bateria (tecnologia de armazenamento de energia não evolui com a mesma rapidez que a Microeletrônica).
- Dissipação do calor do chip requer um projeto térmico cuidadoso e pode incorrer em custos extras com encapsulamento especial (mais caro) e ventilação forçada ("*cooler*").

4. Projeto de Sistemas Digitais no Nível RT

Aspectos a Serem Considerados no Projeto

4. Testabilidade

- No circuito integrado não se tem acesso aos pontos internos, apenas aos pinos de E/S.
- Geralmente, é necessário inserir modificações e até mesmo blocos de hardware que facilitem o teste do sistema digital.
- A fase de teste corresponde a aprox. 50% do custo total de desenvolvimento de um chip.

4. Projeto de Sistemas Digitais no Nível RT

Projeto de Sistemas Digitais para Aplicações Específicas

Colocação do Problema

Dado um algoritmo (i.e., uma descrição comportamental), projetar um **SD** (sistema digital) capaz de implementá-lo, atendendo às restrições e aos requisitos de projeto, no que se refere a:

1. Custo de Implementação
2. Desempenho
3. Consumo de Energia
4. Testabilidade
5. Tolerância (ou Robustez) a Falhas

4. Projeto de Sistemas Digitais no Nível RT

Projeto de Sistemas Digitais para Aplicações Específicas

Exemplo 1: Considere o seguinte algoritmo

```
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
  mult ← P;
  pronto ← 1;
fim
```

OBS: o algoritmo poderia estar descrito em C, C++, Java, SystemC etc, ou já estar representado sob a forma de uma FSMD

4. Projeto de Sistemas Digitais no Nível RT

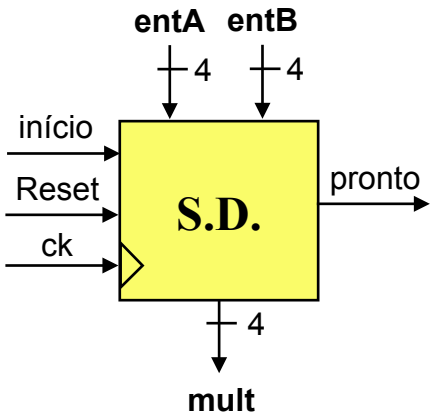
Projeto de Sistemas Digitais para Aplicações Específicas

Exemplo 1: Uma especificação melhorada

Comportamento

```
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
  mult ← P;
  pronto ← 1;
fim
```

Interfaces



4. Projeto de Sistemas Digitais no Nível RT

Projeto de Sistemas Digitais para Aplicações Específicas

Exemplo 1: Informações contidas em um algoritmo

```
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
  mult ← P;
  pronto ← 1;
fim
```

Sinal de controle (de saída) é inicializado com zero

Valores das entradas “entA” e “entB” são atribuídos a variáveis A e B

Variável auxiliar é inicializada com zero

Testes (geram sinais de status para o BC)

Variáveis são usadas em operações aritméticas

Resultado da operação é disponibilizado na saída de dados

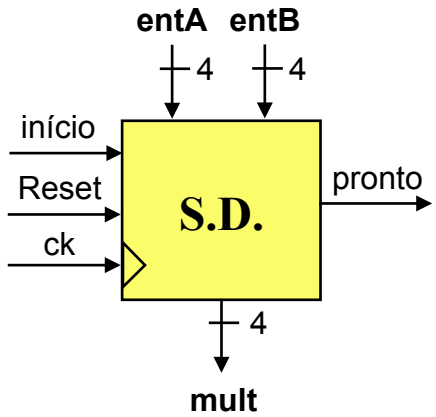
Sinal de controle (de saída) é setado para indicar o término

4. Projeto de Sistemas Digitais no Nível RT

Projeto de Sistemas Digitais para Aplicações Específicas

Exemplo 1: Informações contidas em um algoritmo

```
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
  mult ← P;
  pronto ← 1;
fim
```



Um algoritmo contém informações sobre:

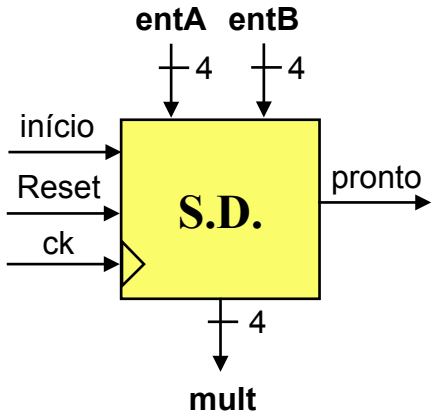
- As operações que devem ser realizadas sobre os dados (usadas no projeto do B.O.)
- O fluxo de execução (usadas no projeto do B.C.)

O algoritmo pode ser representado graficamente por meio de um **fluxograma** ou por uma **FSMD**

Projeto de Sistemas Digitais para Aplicações Específicas

Exemplo 1

```
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
  mult ← P;
  pronto ← 1;
fim
```



Neste algoritmo:

- Há variáveis que servem para armazenar dados (A, B, P)
- Há variáveis que são apenas interfaces de entrada e saída (entA, entB, mult, pronto)
- Deve haver UFs para realizar as operações especificadas
- Associados aos testes deve existir sinais de status que o B.C. Usa para tomar as decisões

Projeto de Sistemas Digitais para Aplicações Específicas

Exemplo de Algoritmo

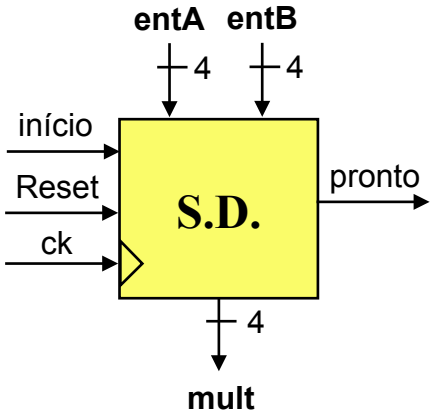
```
início
  pronto ← 0;
  A ← entA;
  B ← entB;
  P ← 0;
  Se B ≠ 0 então
  Enquanto A ≠ 0 faça
    início
    P ← P + B;
    A ← A - 1;
    fim
  mult ← P;
  pronto ← 1;
fim
```

- Até aqui, nada foi especificado a respeito do desempenho e do custo da implementação
- Explorando a relação custo x desempenho:
  Uma operação por ciclo de relógio x várias operações por ciclo.

Projeto do Bloco Operativo Visando Custo Mínimo

Exemplo 1: Projetar um BO para o SD que implementa o algoritmo abaixo, assumindo que:

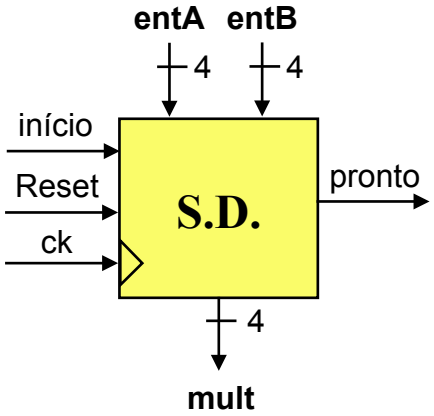
- Este SD deve possuir duas entradas de dados
- O custo de implementação deve ser mínimo
- O SD não precisa ter alto desempenho (e não há restrição quanto ao desempenho mínimo necessário)



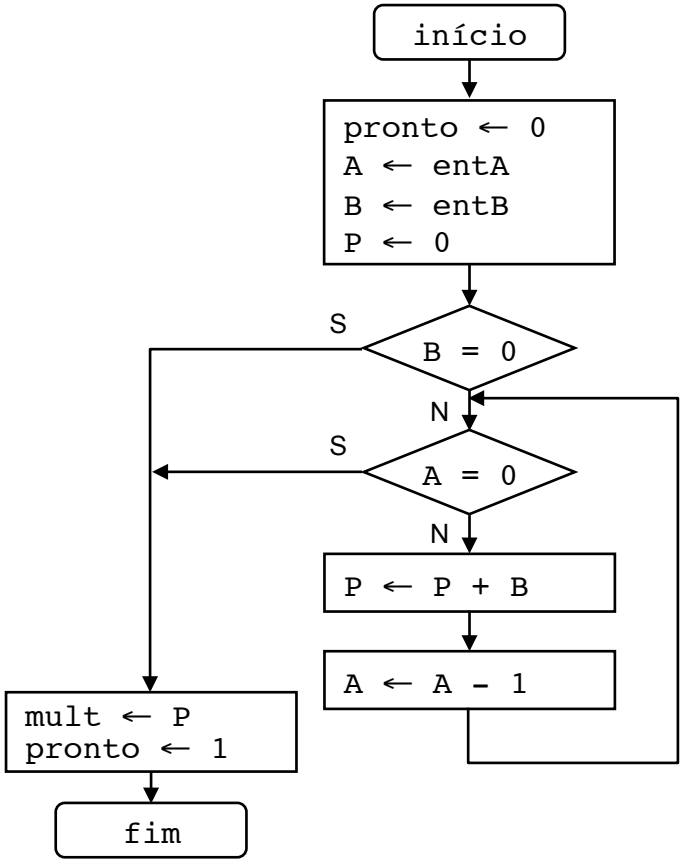

```
início
  pronto ← 0;
  A ← entA;
  B ← entB;
  P ← 0;
  Se B ≠ 0 então
  Enquanto A ≠ 0 faça
    início
    P ← P + B;
    A ← A - 1;
    fim
  mult ← P;
  pronto ← 1;
fim
```

Projeto do Bloco Operativo Visando Custo Mínimo

Solução 1: Reestruturando o Algoritmo para custo mínimo




```
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
  mult ← P;
  pronto ← 1;
fim
```

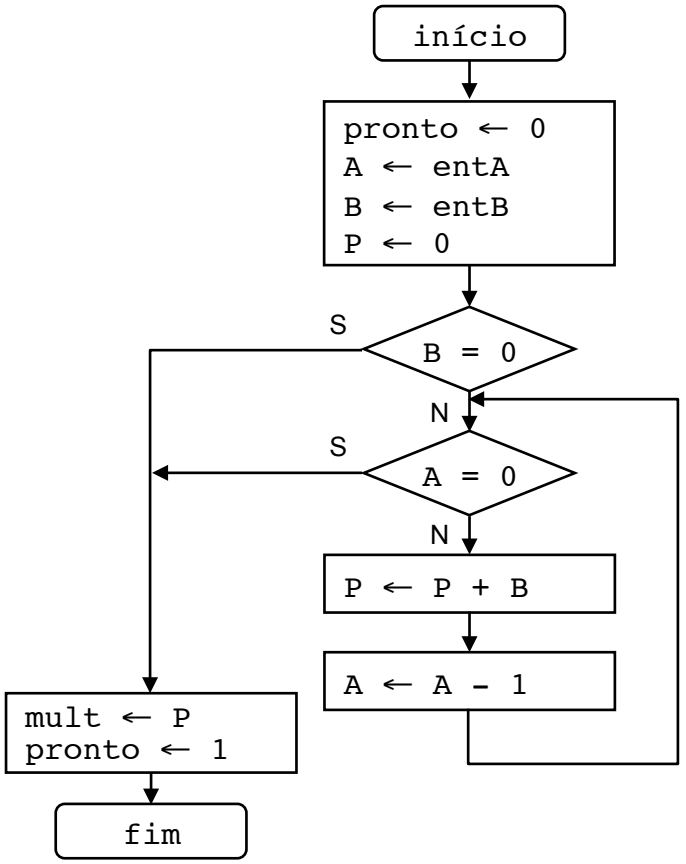


- Iremos assumir que somente uma operação ocorre por ciclo de relógio
- As operações que podem ocorrer em paralelo estão em uma mesma caixa… (observe que "pronto" é uma saída de controle.)

Projeto do Bloco Operativo Visando Custo Mínimo

Solução 1: Unidades Funcionais (UFs) Necessárias

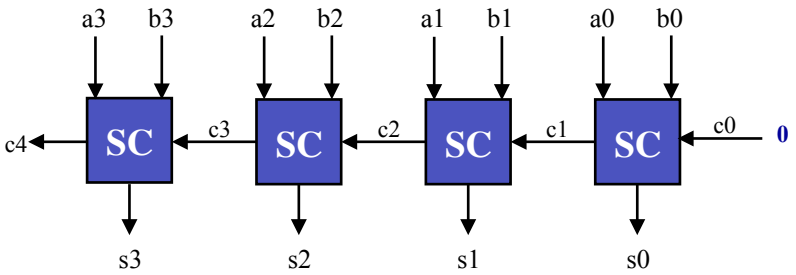
- Operações necessárias: “+” e “-” (na verdade, seria um decremento, mas vamos assumir subtração)
- As operações “+” e”-” são usadas em ciclos de relógio diferentes. Logo, poderemos usar **um somador/subtrator**, que é mais barato que um somador mais um subtrator



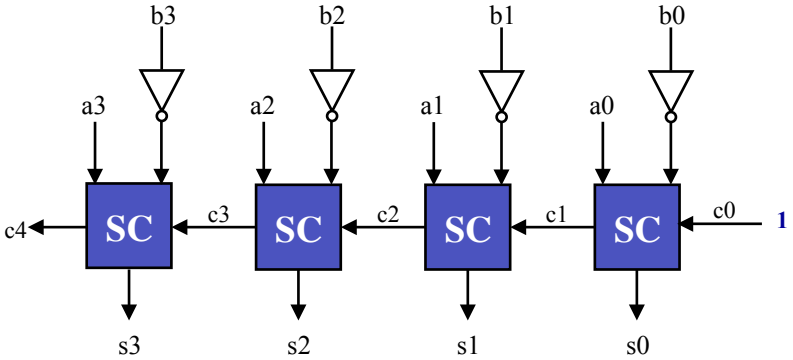
Projeto do Bloco Operativo Visando Custo Mínimo

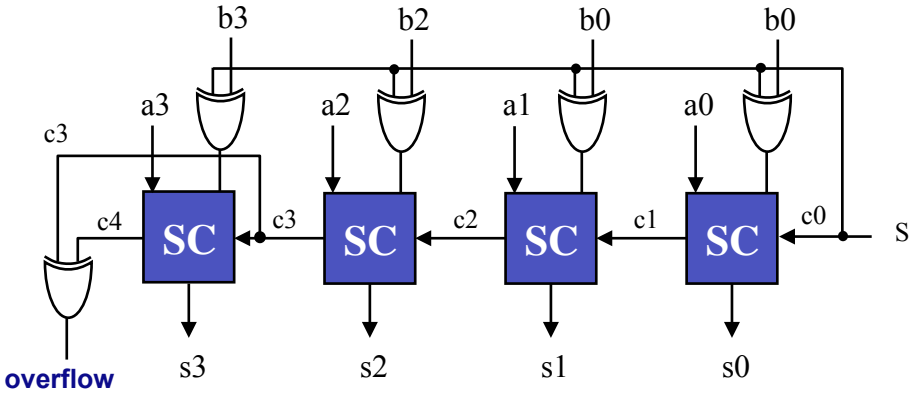
Solução 1: Custo de UFs *Versus* Custo de UFs Combinadas

Somador de 4 bits



Subtrator de 4 bits

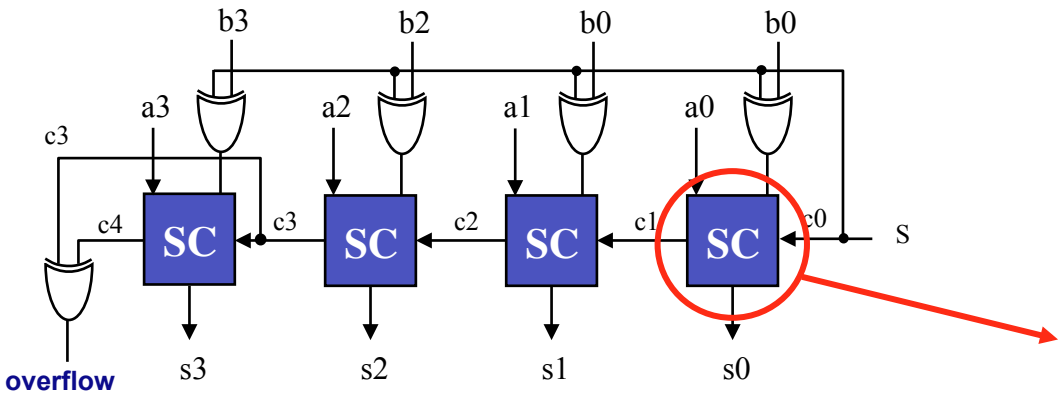


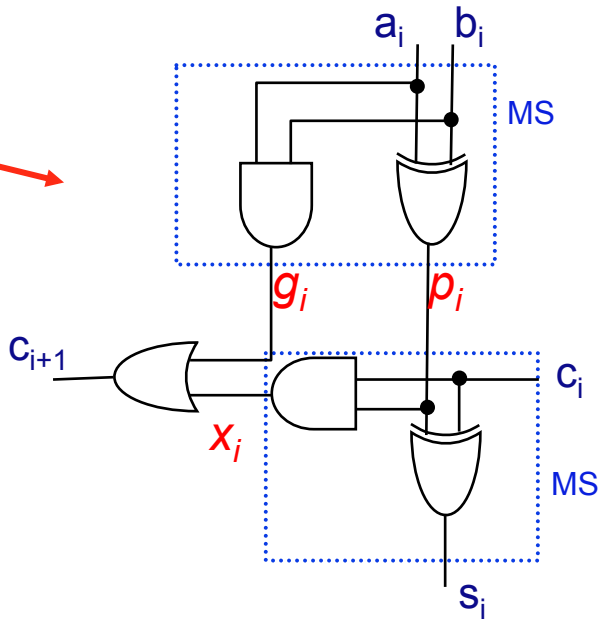


Somador/Subtrator de 4 bits

Projeto do Bloco Operativo Visando Custo Mínimo

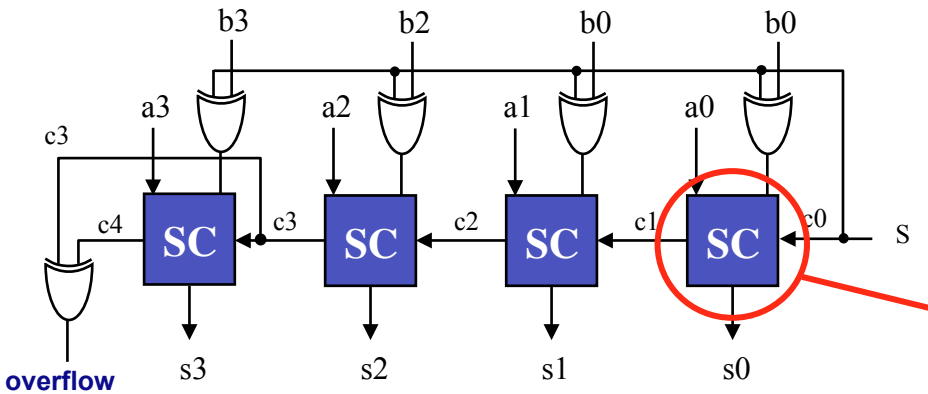
Calculando o Custo do Somador/Subtrator



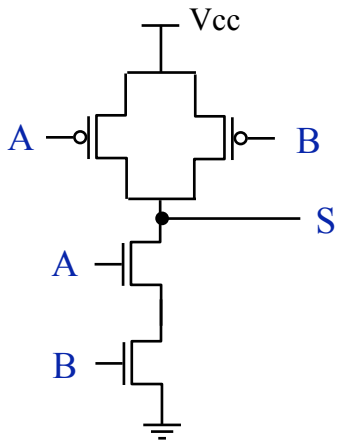


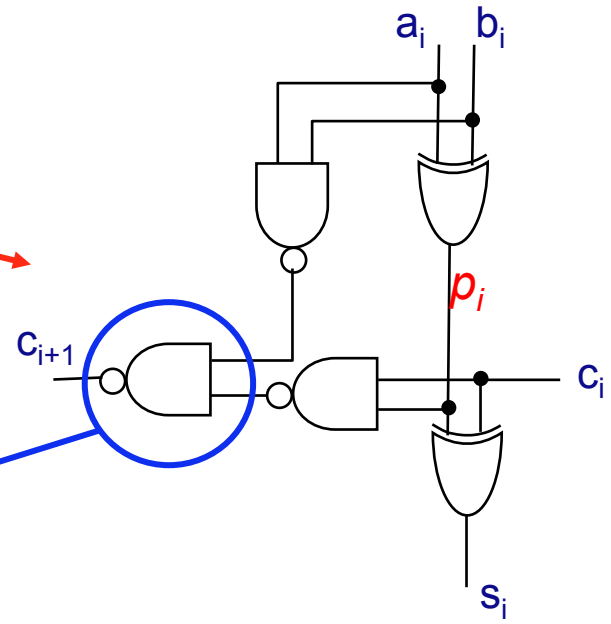
Projeto do Bloco Operativo Visando Custo Mínimo

Calculando o Custo do Somador/Subtrator



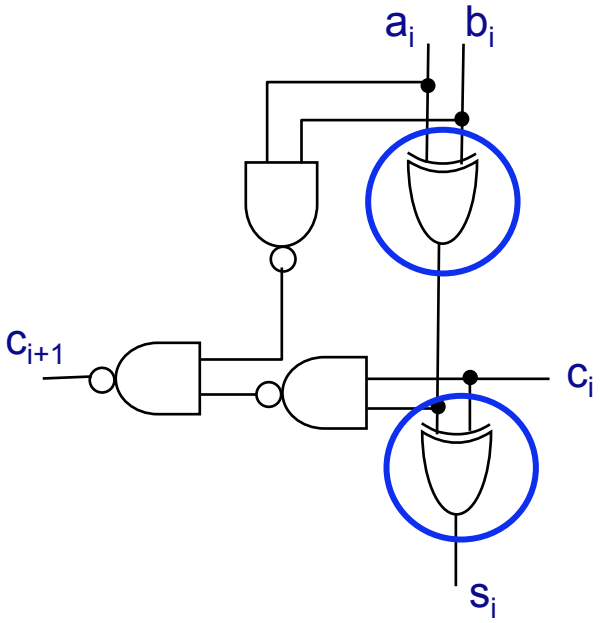
Mas em CMOS…



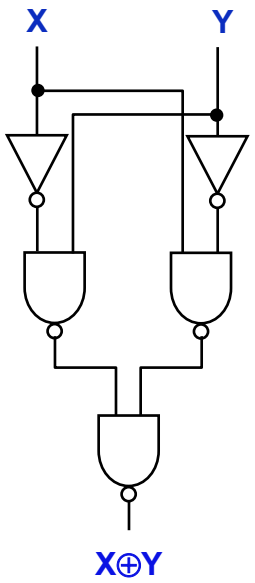


Projeto do Bloco Operativo Visando Custo Mínimo

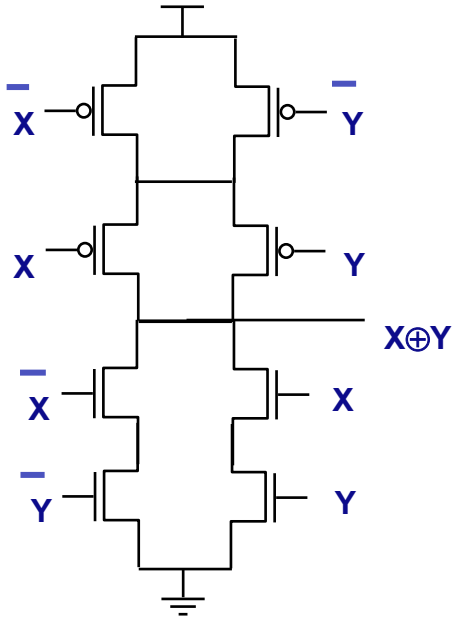
Calculando o Custo do Somador/Subtrator



Algumas Implementações CMOS para a xor



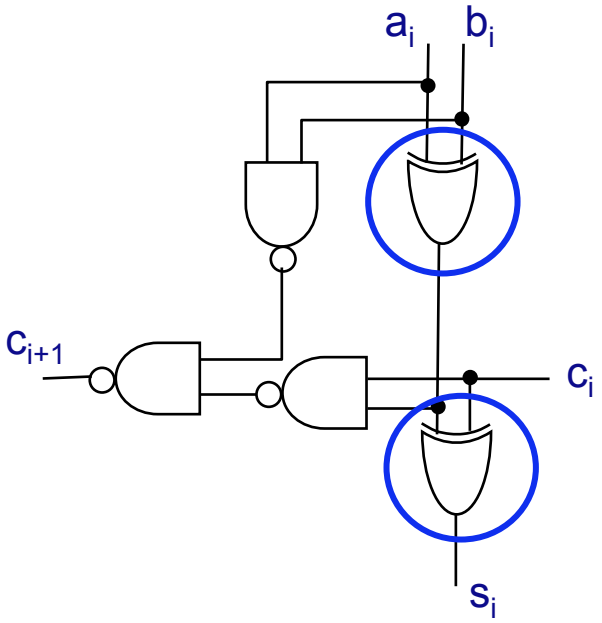
16 transistores



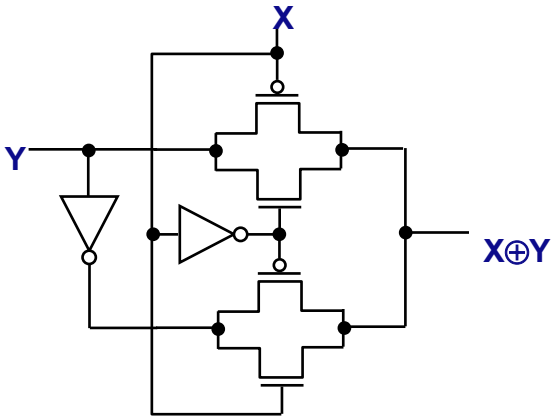
12 transistores
(necessita de 2 inversores)

Projeto do Bloco Operativo Visando Custo Mínimo

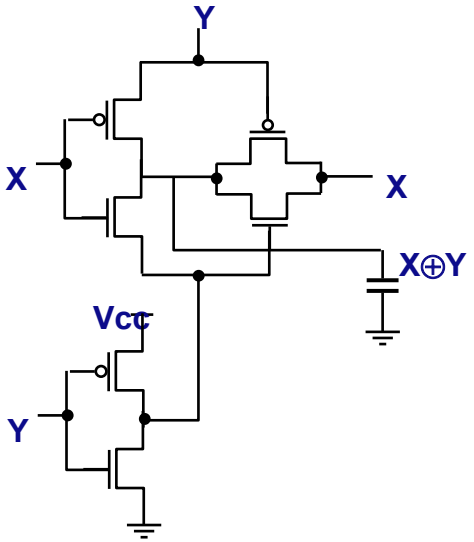
Calculando o Custo do Somador/Subtrator



Algumas Implementações CMOS para a xor



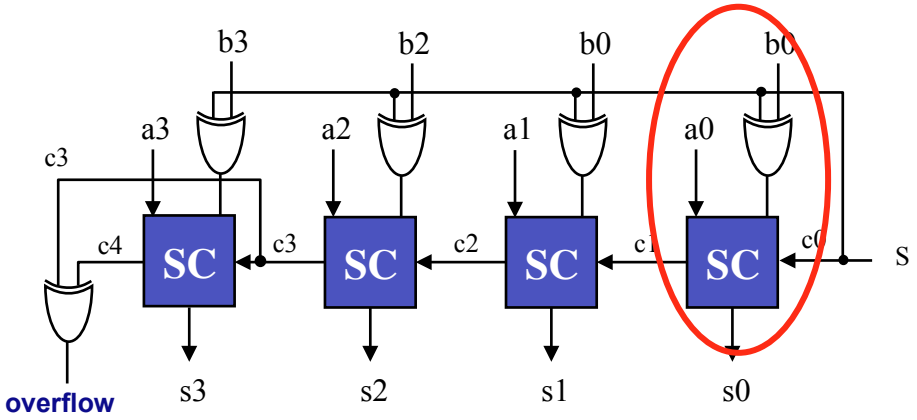
8 transistores



6 transistores
(é a mais usada)

Projeto do Bloco Operativo Visando Custo Mínimo

Calculando o Custo do Somador/Subtrator



Custo do Somador/subtrator, por bit:

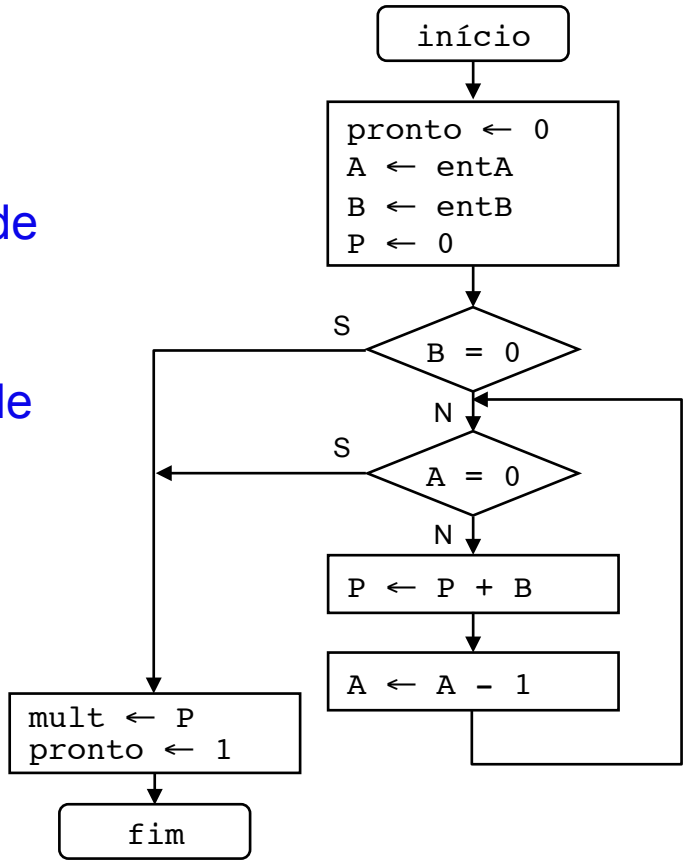
- 3 portas xor: 3 x 6 = 18 transistores
- 3 portas nand de duas entradas: 3 x 4 = 12 transistores
- Logo, custo de um bit = 30 transistores (ignorando-se a xor que calcula o overflow)

Custo de um somador/subtrator de n bits: 30n transistores

Projeto do Bloco Operativo Visando Custo Mínimo

Solução 1: Registradores

- Há três variáveis (p/ dados): **A**, **B** e **P**
- Iremos considerar que **mult** é apenas uma saída de dados (e portanto, não necessita de um registrador)
- Quantos registradores serão necessários? Será preciso fazer uma "análise do tempo de vida das variáveis"...



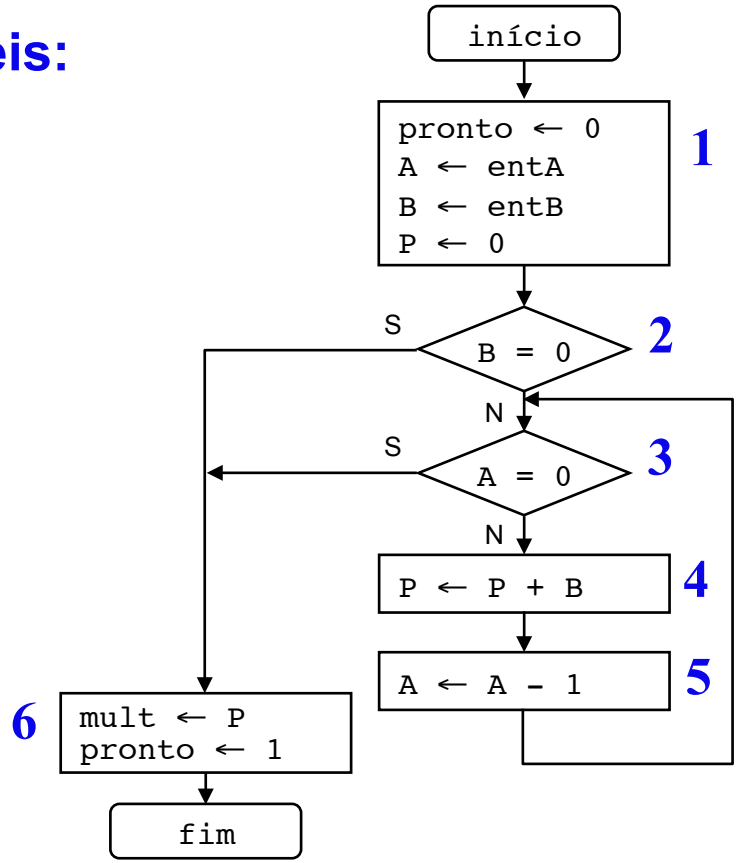
Projeto do Bloco Operativo Visando Custo Mínimo

Solução 1: Registradores

Análise do tempo de vida das variáveis:

A análise acima considera que uma variável está "viva" desde o ciclo de relógio subsequente ao ciclo no qual ela recebe um valor novo até o último ciclo no qual ela é consultada. ("Janelas podem ocorrer...)

	1	2	3	4	5	6
A		X	X	X	X	
B		X	X	X	X	
P		X	X	X	X	X



Projeto do Bloco Operativo Visando Custo Mínimo

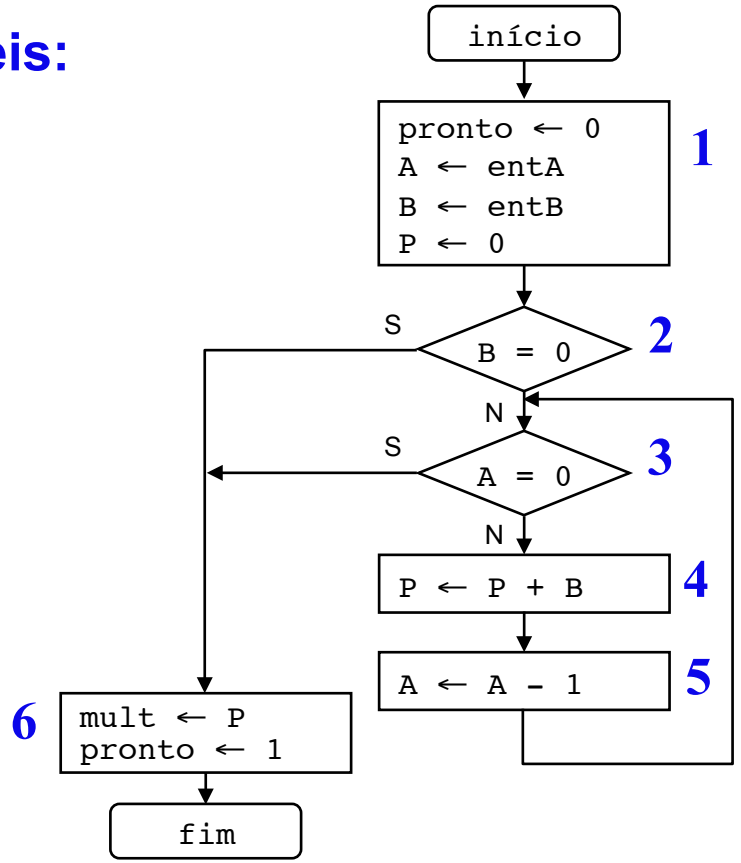
Solução 1: Registradores

Análise do tempo de vida das variáveis:

	1	2	3	4	5	6
A		X	X	X	X	
B		X	X	X	X	
P		X	X	X	X	X

as variáveis A, B e P são escritas na borda de relógio que encerra o passo 1 e dá início ao passo 2

O número máximo de variáveis simultaneamente **"vivas" é 3**. Logo, são necessários **3 registradores.** Chamemo-los de **A, B e P.**

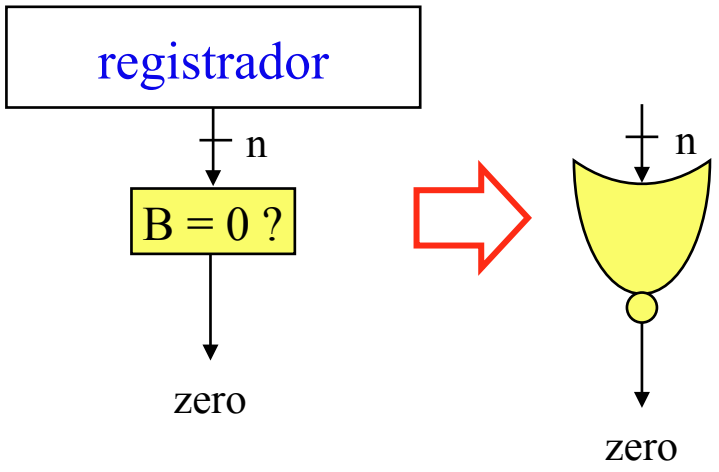


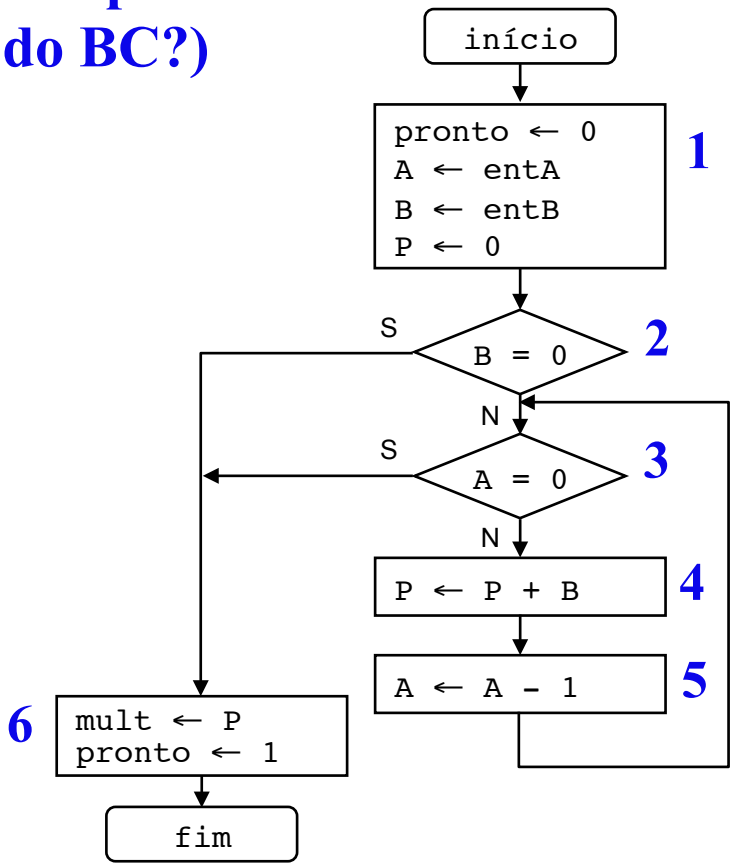
Projeto do Bloco Operativo Visando Custo Mínimo

Dúvida: não dá para reduzir o número de passos?
(e com isto, reduzir também o custo do BC?)

Resposta: sim!

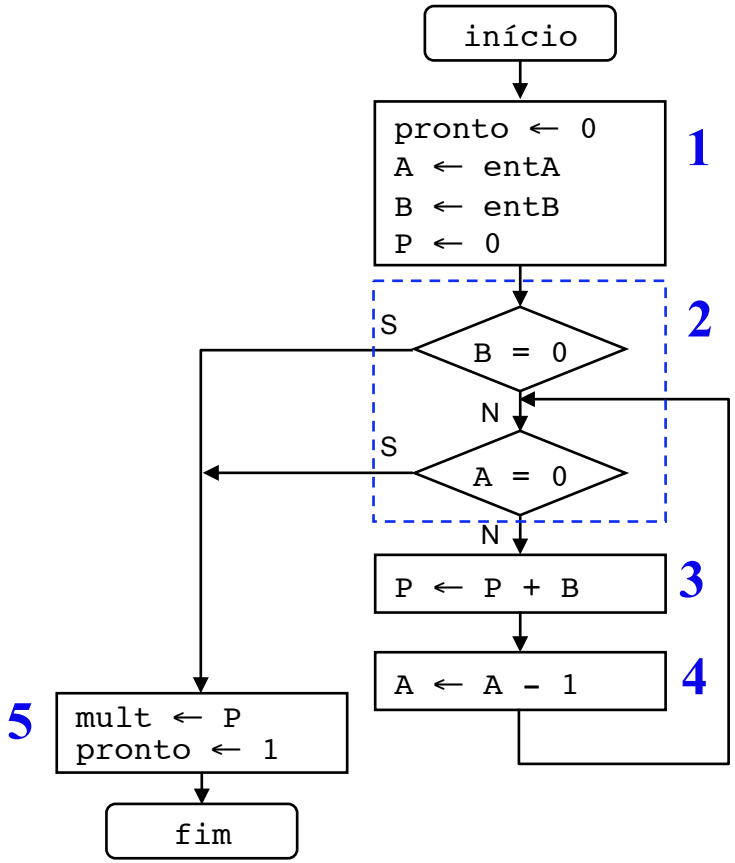
Se usarmos dois comparadores, poderemos realizar os passos 2 e 3 em um único estado

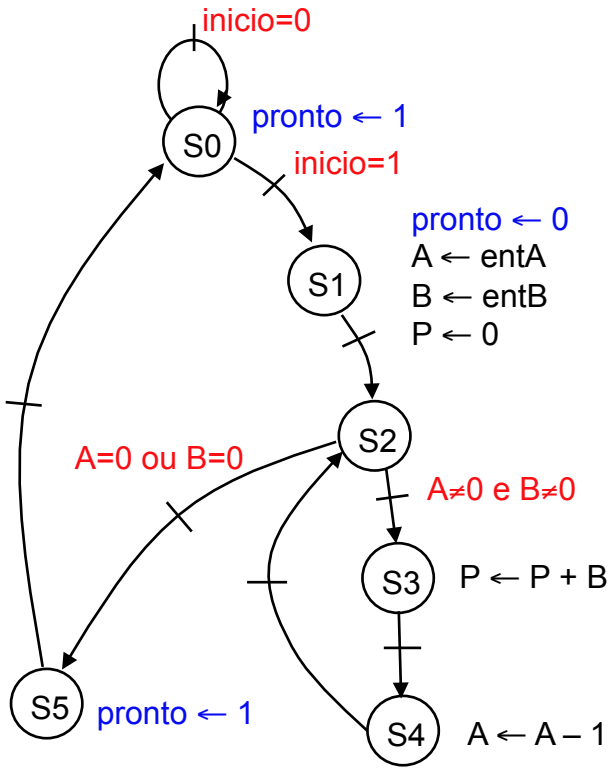




Projeto do Bloco Operativo Visando Custo Mínimo

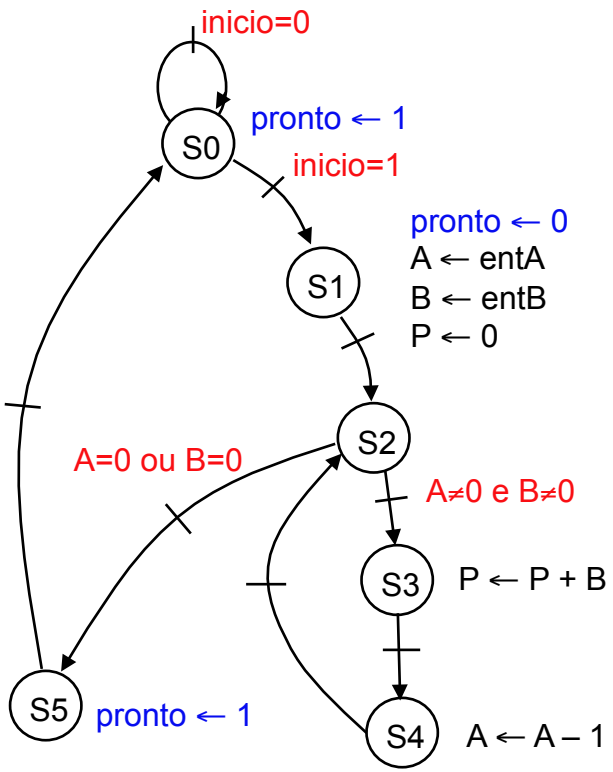
Solução 1: Novo Fluxograma e FSMD equivalente





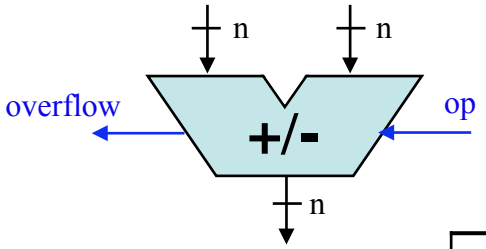
4. Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco Operativo Visando Custo Mínimo



Solução 1: elementos para o BO:

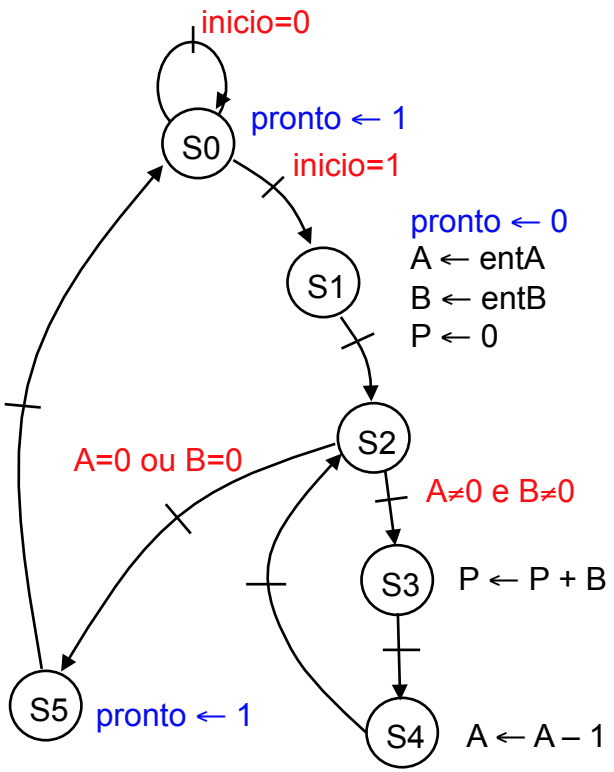
- 1 somador/subtrator
- 3 registradores com carga paralela, A, B e P, sendo P com reset assíncrono
- Rede de interconexão apropriada

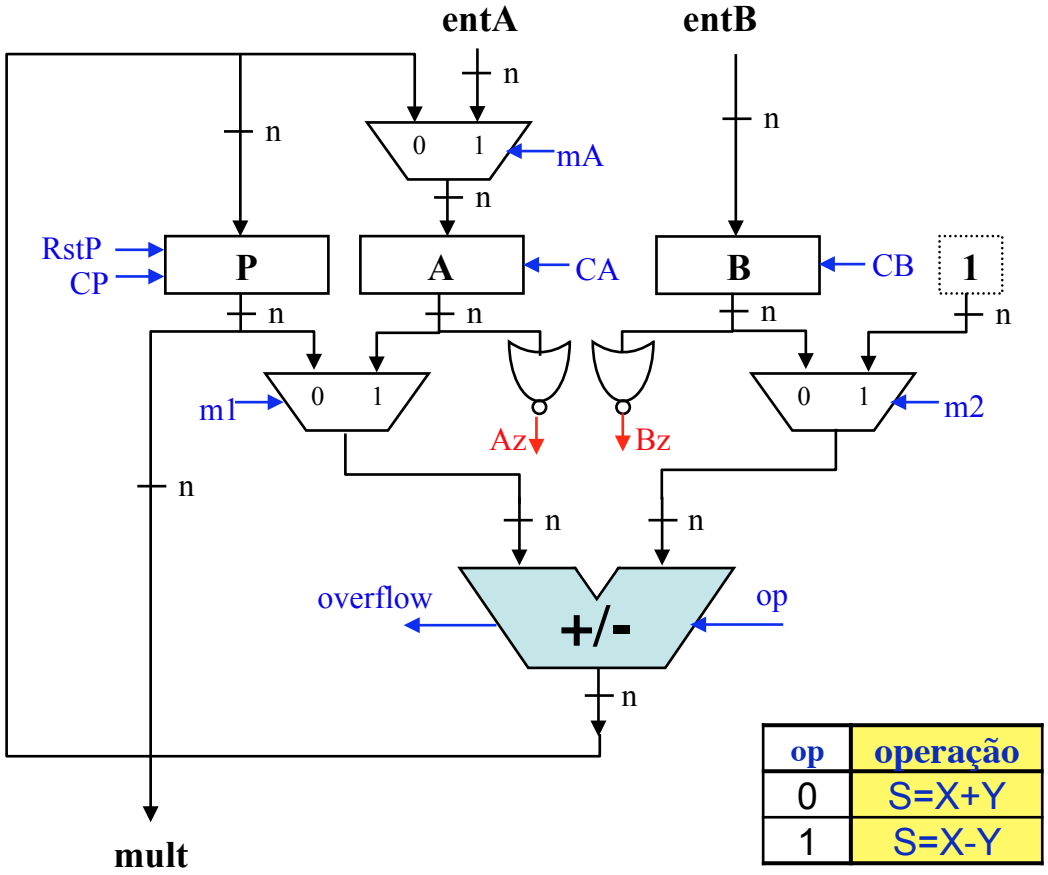


op	operação
0	S=X+Y
1	S=X-Y

4. Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco Operativo Visando Custo Mínimo





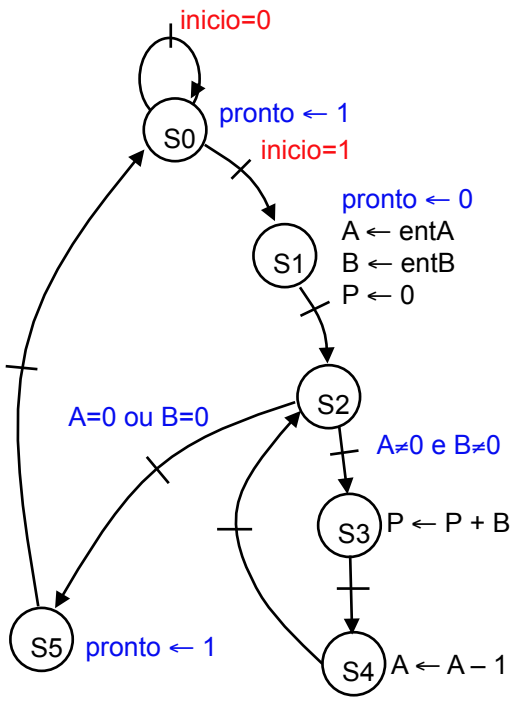
op	operação
0	S=X+Y
1	S=X-Y

4. Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco Operativo Visando Custo Mínimo

Verificação do Funcionamento

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

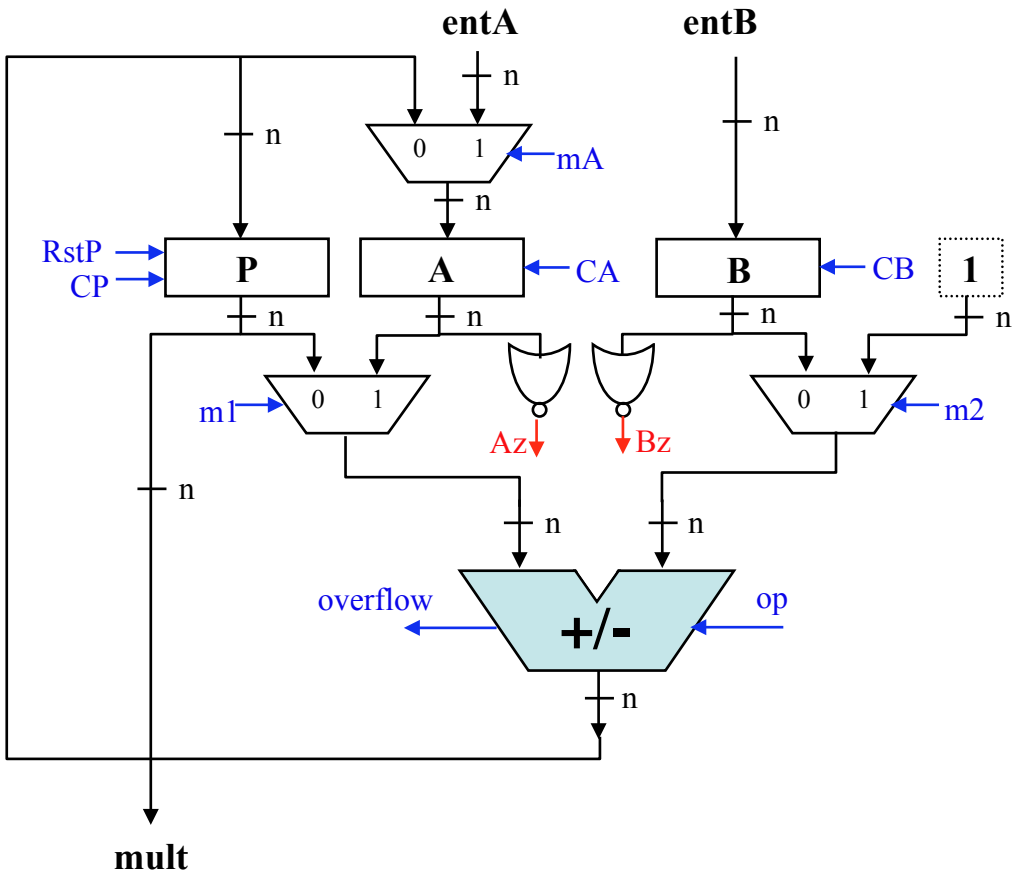


Conteúdo dos registradores (e valor do sinal de status Az)

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

Tempo de execução: 12 ciclos de relógio (+ nº de ciclos em que ficar em S0)

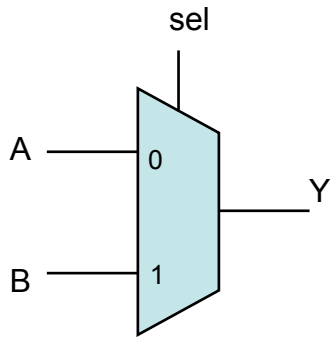
Cálculo do Custo do Bloco Operativo

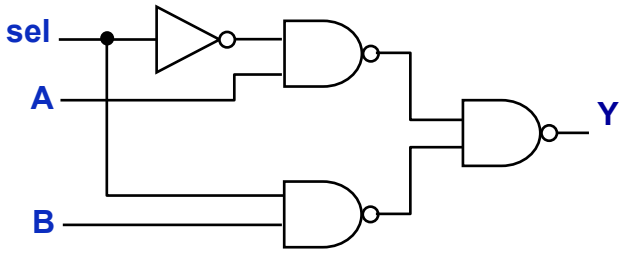


- somador/subtrator de n bits: 30n transistores
- E o resto?

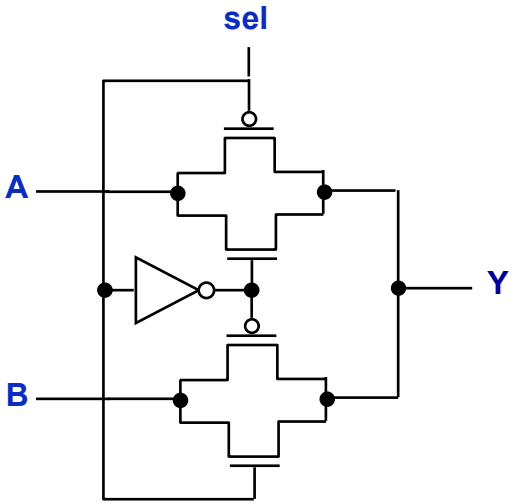
Estimativa do Custo do Bloco Operativo

Custo do Mux 2:1 em CMOS





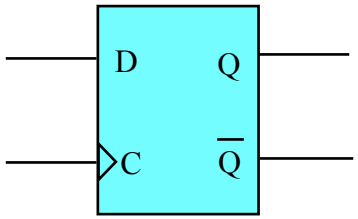
14 ou 12 transistores



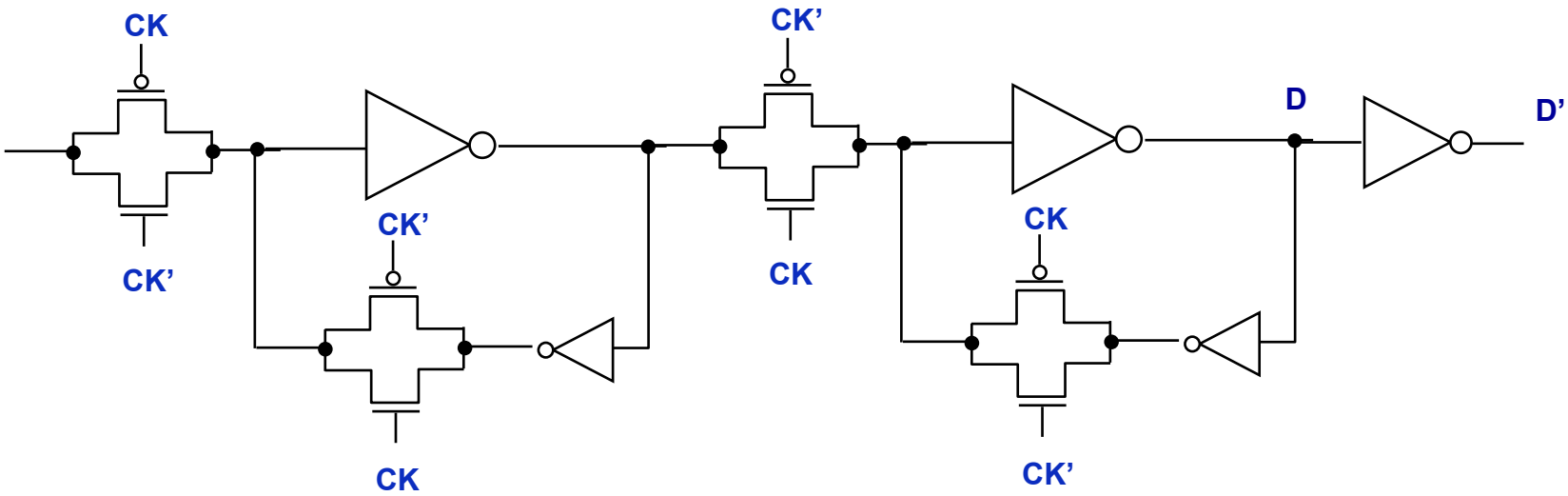
6 ou 4 transistores
(mais usado)

Estimativa do Custo do Bloco Operativo

Custo de um Flip-flop D mestre-escravo CMOS



18 ou 20 transistores
(eventualmente, podemos considerar somente um inversor para o clock de todos os bits)

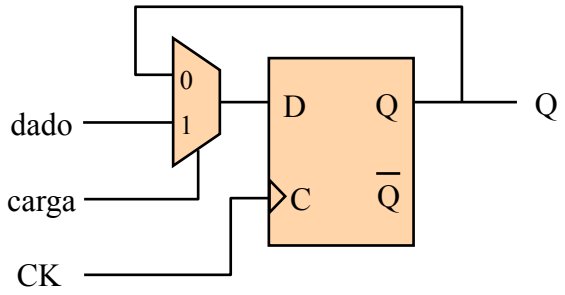


OBS: para set ou reset assíncrono, adicionar 2 transistores

Estimativa do Custo do Bloco Operativo

Custo de um Flip-flop D mestre-escravo CMOS com habilitação de carga paralela

18+4= 22 transistores

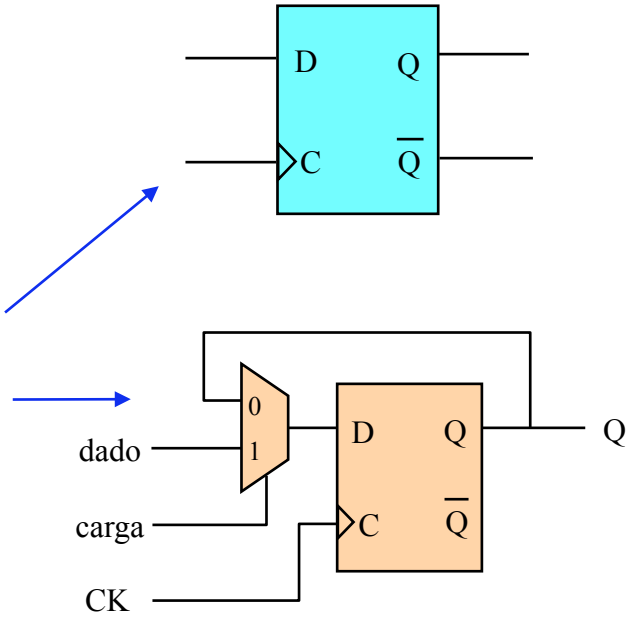


OBS: para set ou reset assíncrono, adicionar 2 transistores

Estimativa do Custo do Bloco Operativo

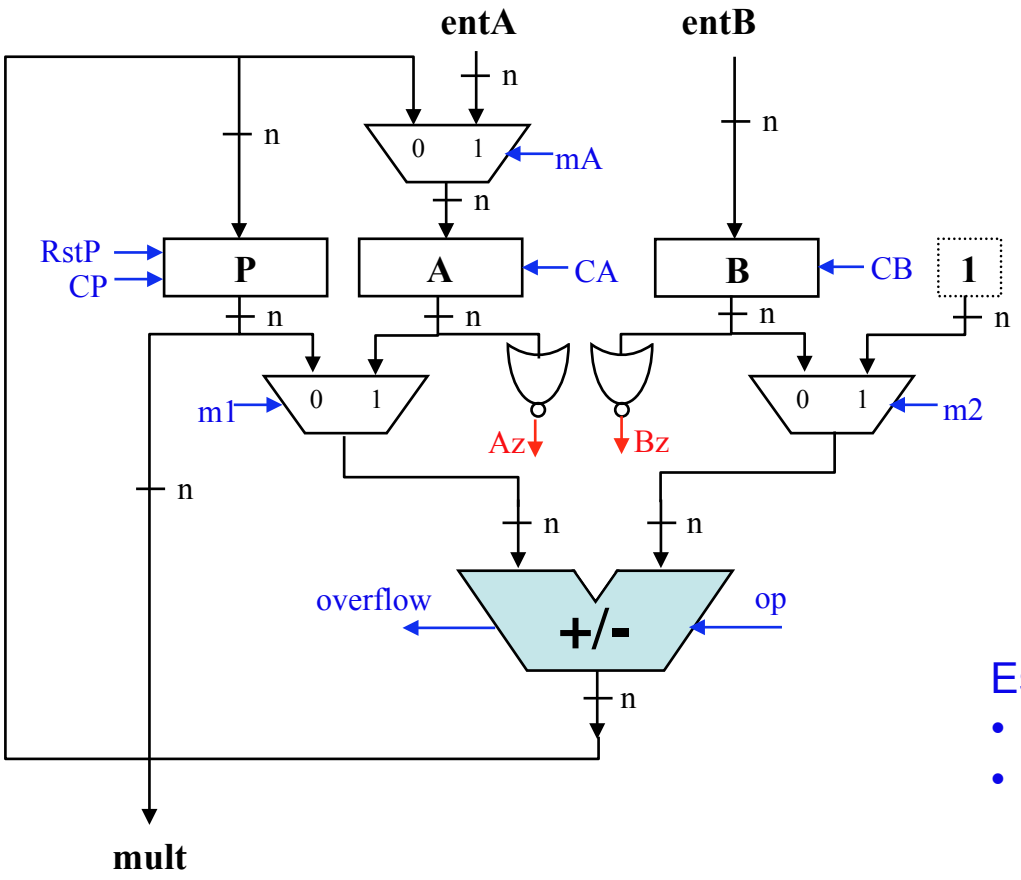
Resumo

Componente RT	Custo
Somador	24n
Subtrator	26n
Somador/subtrator	30n
Mux 2:1	4n
Registrador com carga paralela (+4 transistores para set ou reset assíncrono)	18n
Registrador com carga paralela controlada (+4 transistores para set ou reset assíncrono)	22n



4. Projeto de Sistemas Digitais no Nível RT

Estimativa do Custo do Bloco Operativo



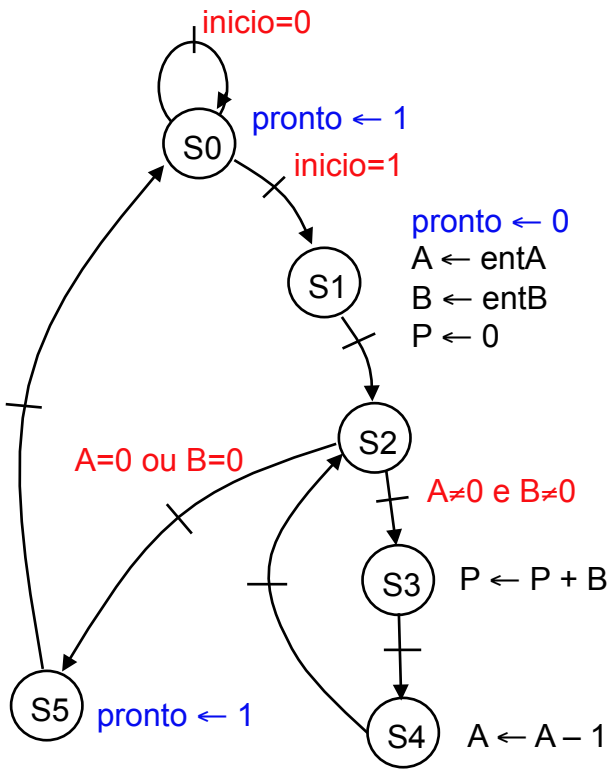
Custo do BO 1	Custo
1 Somador/subtrator	30n
3 Muxes 2:1	3x4n=12n
2 Registradores com carga paralela controlada	2x22n=44n
1 Registrador com carga paralela controlada e reset assíncrono	26n
Total	**112n**

Estimativa de custo para o BC:

- Número de estados: 5 ou 6
- Número de sinais de controle = **8**

4. Projeto de Sistemas Digitais no Nível RT

Estimativa do Desempenho do Bloco Operativo



Se n = 4 bits:

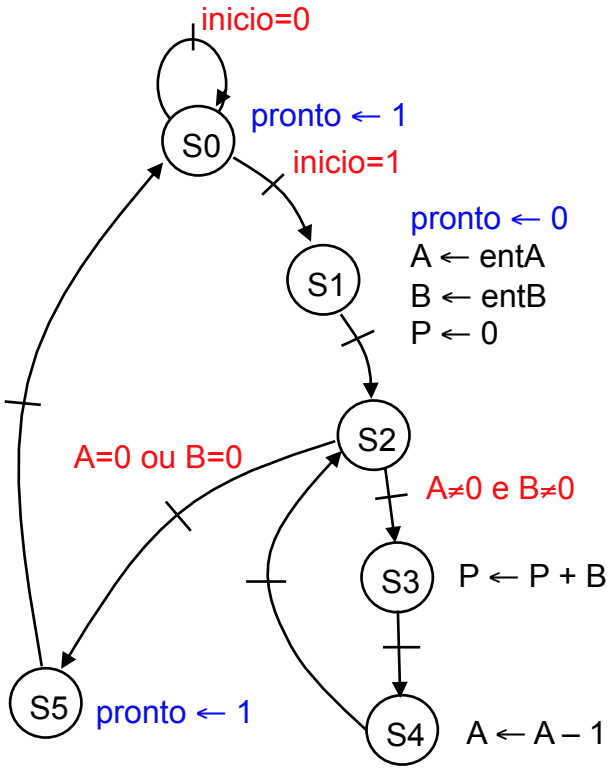
- Maior inteiro sem sinal: 15 (⇒1111)
- Pior caso: A=15, B≠0
- Sequência de execução: S1, 15 x[S2,S3,S4], S2, S5 = **48 ciclos de relógio**

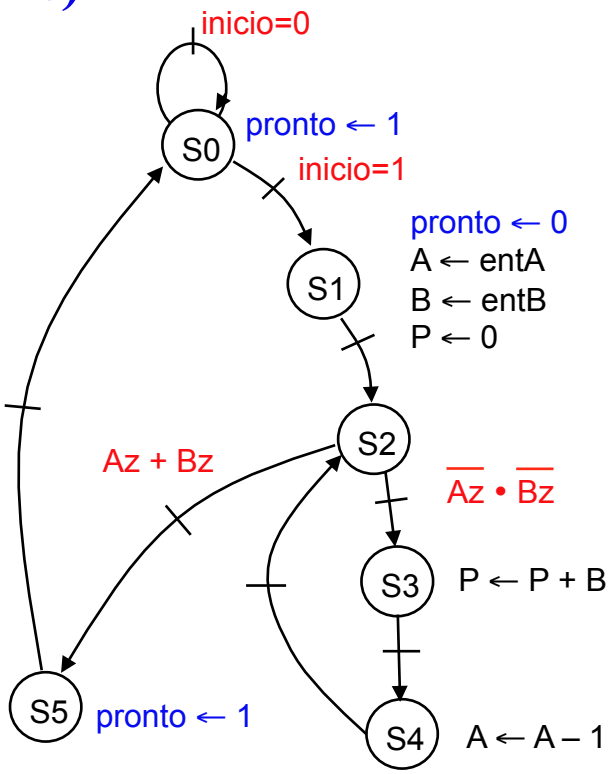
Generalizando para n bits:

- Maior inteiro sem sinal: 2^n-1
- Pior caso: A= 2^n-1 , B≠0
- Sequência de execução: S1, (2^n-1)x [S2,S3,S4], S2, S5 = 3x(2^n-1)+3 ciclos de relógio = **3x2^n ciclos de relógio**

Projeto do Bloco de Controle Visando Custo Mínimo

Diagrama de Estados (Assumindo Moore)

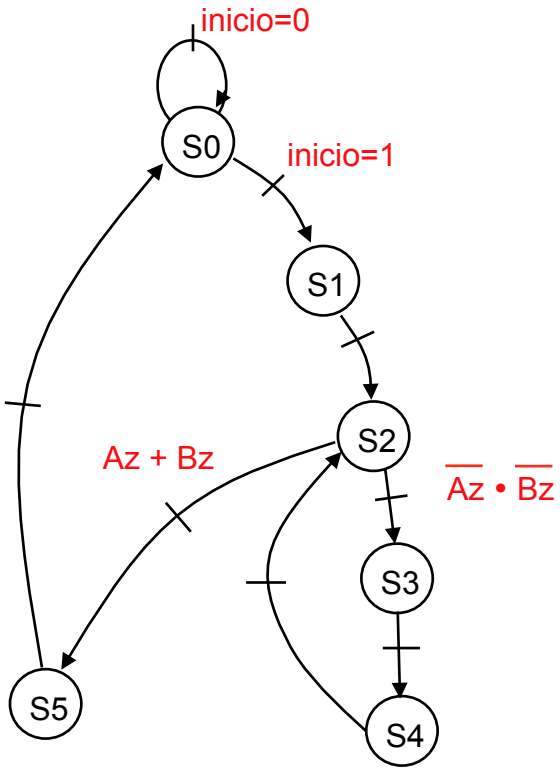




Note que as condições "Az + Bz" e "$\overline{Az} \cdot \overline{Bz}$" são complementares.

Projeto do Bloco de Controle Visando Custo Mínimo

Tabela de Transição de Estados (Assumindo Moore)

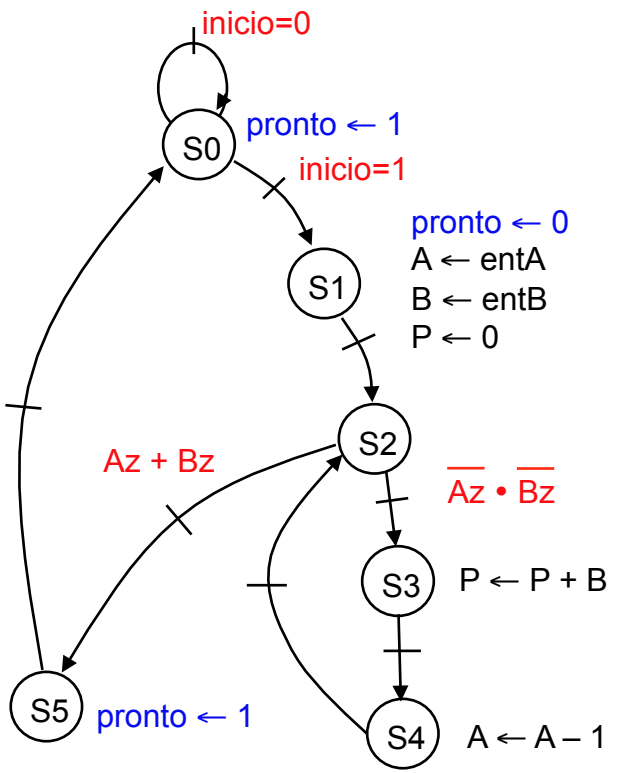


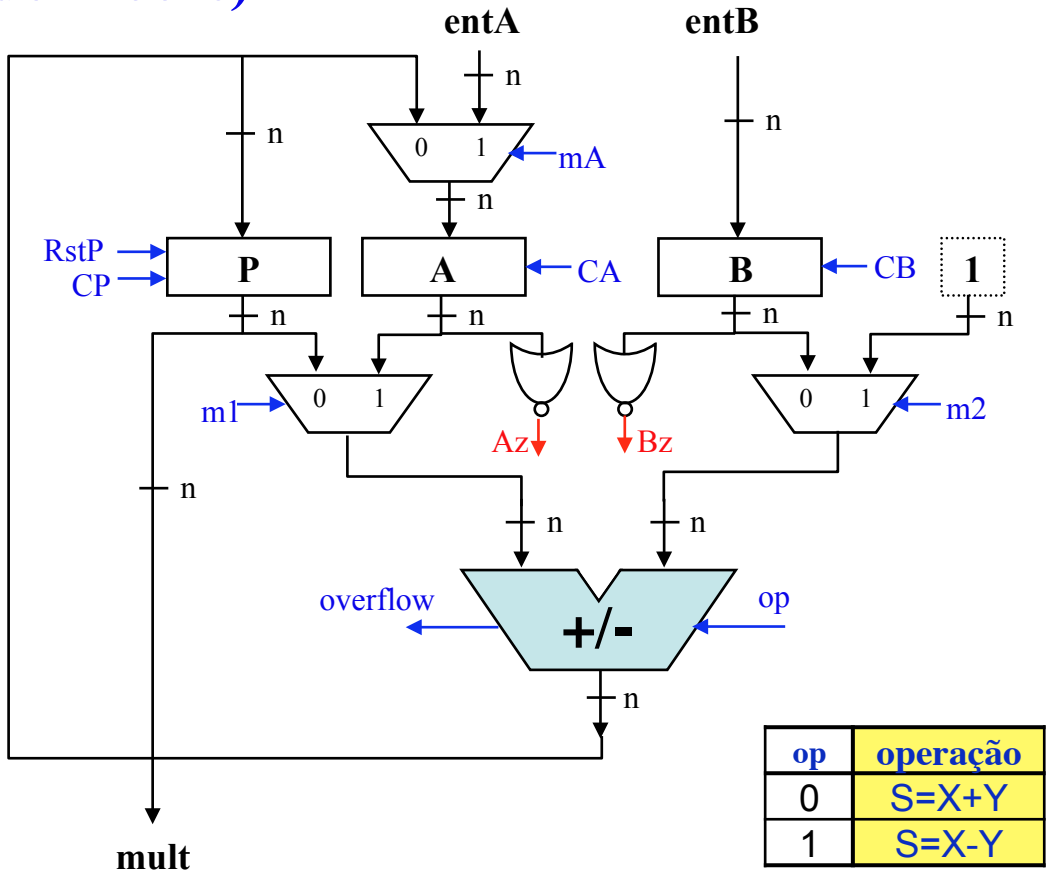
Estado atual	Entradas			Próx. Estado
	início	BZ	AZ	
S0	0	-	-	S0
	1	-	-	S1
S1	-	-	-	S2
S2	-	0	0	S3
	-	0	1	S5
	-	1	0	S5
	-	1	1	S5
S3	-	-	-	S4
S4	-	-	-	S2
S5	-	-	-	S0

4. Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco de Controle Visando Custo Mínimo

Tabela de Saídas (Assumindo Moore)



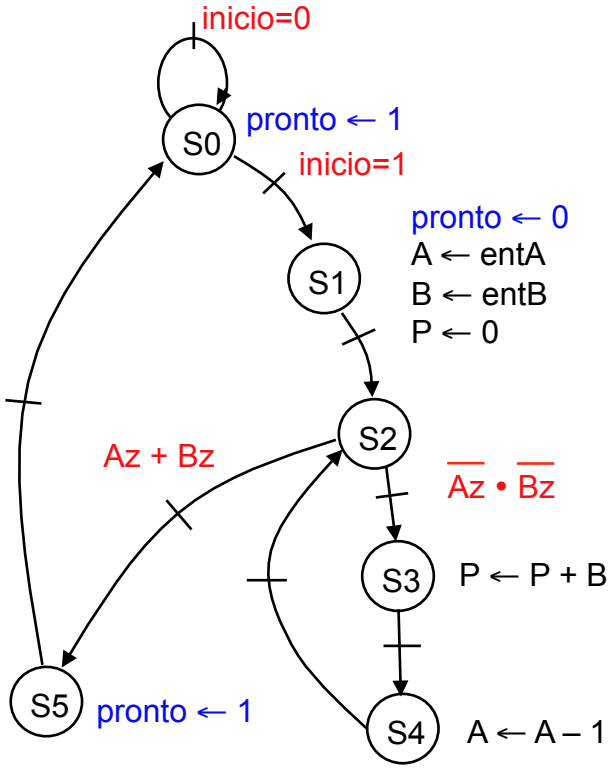


op	operação
0	S=X+Y
1	S=X-Y

4. Projeto de Sistemas Digitais no Nível RT

Projeto do 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	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

1 sinal **1 sinal**

RstP = mA = CB
CP
CA = op = m1 = m2
pronto

4 sinais