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

Lógica Programável

INE 5348

Aula 10

Projeto de Sistemas Digitais no Nível RT. Componentes do nível RT. O modelo BO / BC (datapath x controle). Estudo de caso. Projeto do multiplicador por somas sucessivas (sol.1- custo mínimo).

Prof. José Luís Güntzel
guntzel@inf.ufsc.br

www.inf.ufsc.br/~guntzel/ine5348/ine5348.html

Projeto de Sistemas Digitais no Nível RT

Tipos e Características dos Sistemas Digitais

Classificação Segundo a Temporização

1. SDs Assíncronos:

- Não possuem um sinal de relógio para prover o cadenciamento das operações
- As operações são vistas como eventos encadeados (ou independentes)
- Possui um custo maior em termos de recursos, pois é necessário implementar protocolos de comunicação entre os blocos do sistema
- São tolerantes a variações na temporização

Projeto de Sistemas Digitais no Nível RT

Tipos e Características dos Sistemas Digitais

Classificação Segundo a Temporização

2. SDs Síncronos:

- Utilizam um sinal de relógio para prover o cadenciamento das operações
- O funcionamento é quebrado em passos denominados operações
- Cada operação geralmente leva um ciclo de relógio para ser realizada, mas há esquemas alternativos (ex.: *chaining*, *pipelining*)
- Em um ciclo de relógio uma ou mais operações podem ser realizadas simultaneamente
- As técnicas de projeto existentes permitem abstrair-se detalhes do comportamento

Tipos e Características dos Sistemas Digitais

Classificação Segundo as Funcionalidades

1. Sistemas Digitais de Aplicação Específica (ASICs):

- Realizam somente um algoritmo
- Oferecem pouca ou nenhuma programabilidade (i.e., alteração da funcionalidade)
- O projeto é feito de modo a otimizar a execução do algoritmo implementado, visando a aplicação específica (máximo desempenho com o mínimo custo e eventualmente, mínimo consumo de energia)
- Exemplos: codificadores/decodificadores de imagens

Tipos e Características dos Sistemas Digitais

Classificação Segundo as Funcionalidades

2. Sistemas Digitais de Propósito Geral:

- Podem ser programados para executar diversos algoritmos (ou virtualmente, todos os algoritmos)
- Para tanto, são projetados para realizar um **conjunto de instruções**
- São otimizados para realizar o conjunto de instruções para o qual são projetados (e não um algoritmo ou uma classe de algoritmo)
- Exemplos: microprocessadores

Projeto de Sistemas Digitais no Nível RT

Tipos e Características dos Sistemas Digitais

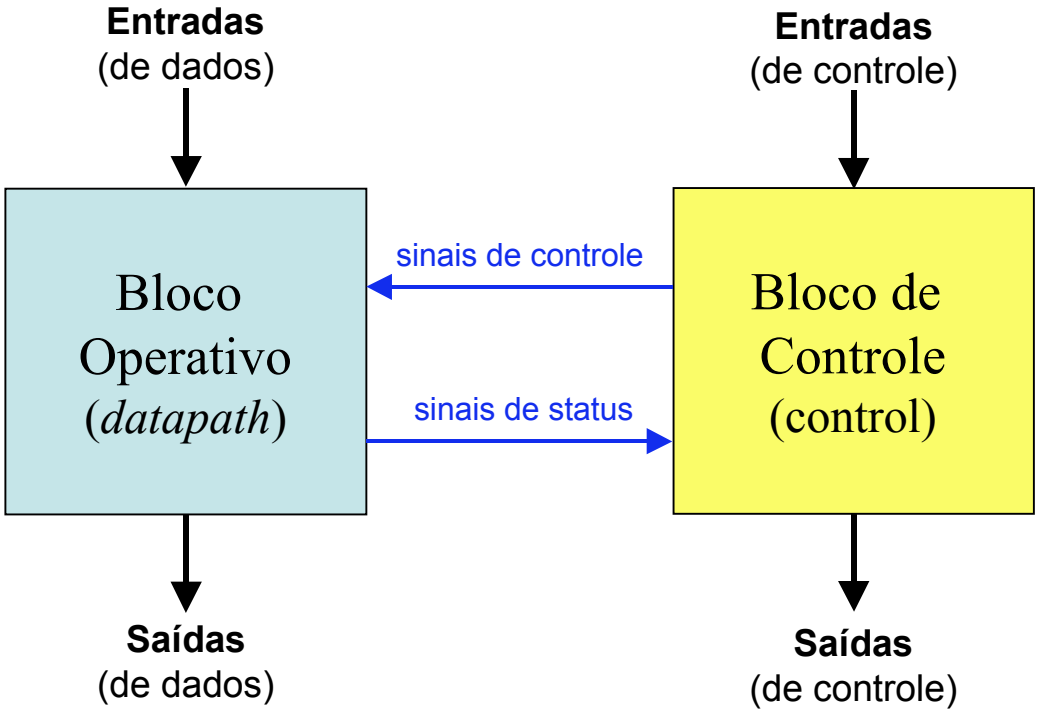
Classificação Segundo as Funcionalidades

3. Sistemas Digitais Programáveis, de Aplicações Específicas:

- Podem ser programados para executar uma função ou um algoritmo pertencente a uma determinada classe.
- São projetados para realizar um **conjunto de instruções** apropriado à classe de problema a que se destinam
- São otimizados para realizar o conjunto de instruções para o qual são projetados (e não um algoritmo ou uma classe de algoritmo)
- Exemplos: microcontroladores, DSPs (*Digital Signal Processors*) e GPUs (*Graphic Processing Units*)

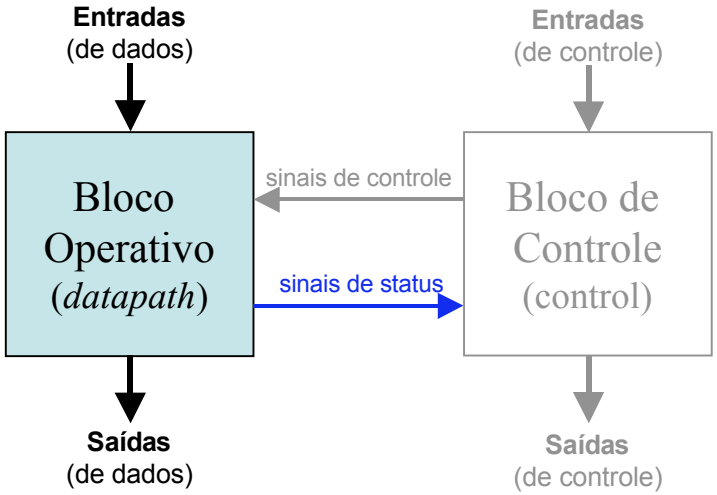
Projeto de Sistemas Digitais no Nível RT

O Modelo Bloco Operativo / Bloco de Controle



Projeto de Sistemas Digitais no Nível RT

O Modelo Bloco Operativo / Bloco de Controle

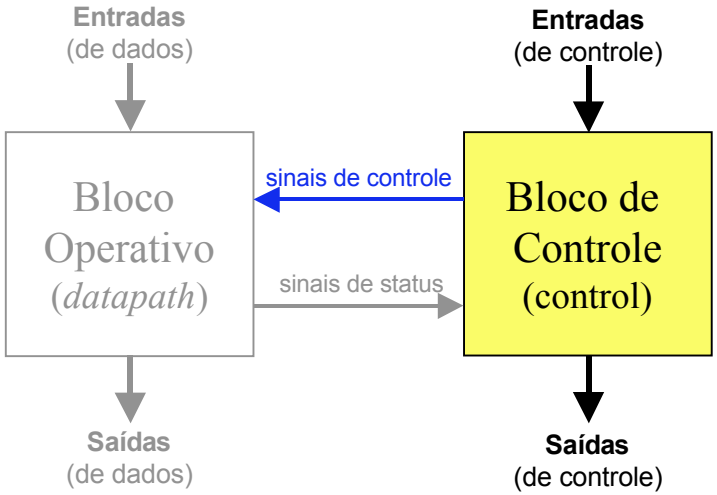


Bloco Operativo:

- Realiza transformações sobre dados, geralmente provenientes do mundo externo
- As transformações são realizadas em um ou mais passos, cada passo demorando um ciclo de relógio
- Gera sinais de "status" que são usados pelo Bloco de Controle para definir a seqüência de operações a serem realizadas (às vezes são chamados de "*flags*")

Projeto de Sistemas Digitais no Nível RT

O Modelo Bloco Operativo / Bloco de Controle

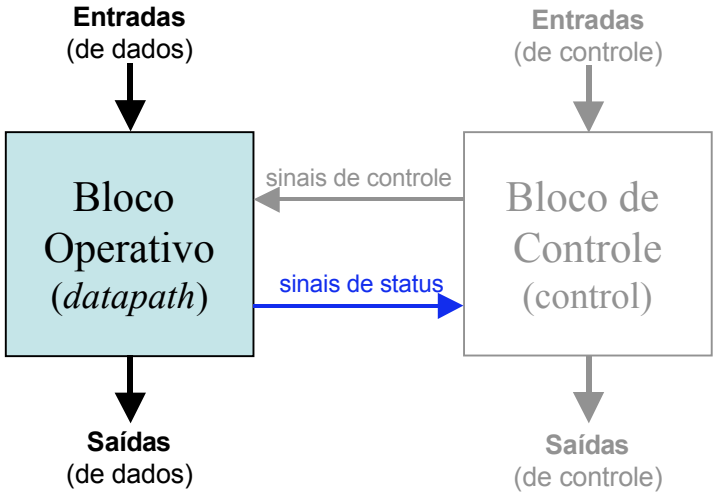


Bloco de Controle:

- Gera os sinais de controle necessários e na ordem correta, coordenando os recursos do bloco operativo
- Recebe sinais de controle do mundo exterior, podendo ser desde um simples "iniciar" até um código de operação ("opcode", dos processadores)
- Pode gerar uma ou mais saídas de controle para se comunicar com outros sistemas digitais (p. ex.: "done", "bus request", "ack")

Projeto de Sistemas Digitais no Nível RT

Os Componentes do Nível RT

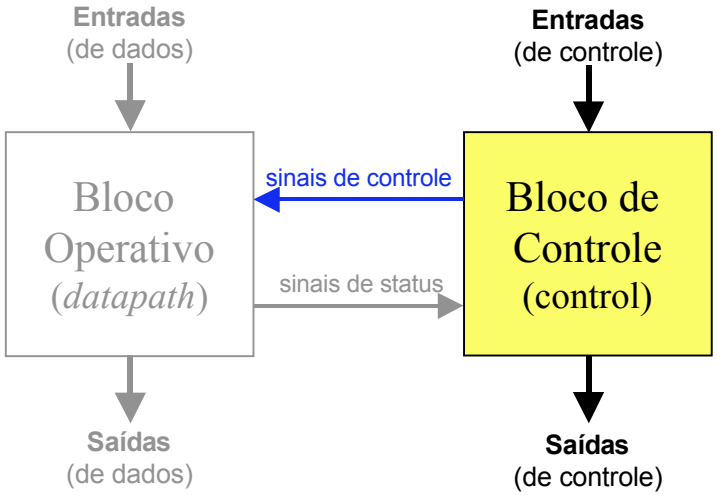


Bloco Operativo:

- Unidades Funcionais (UFs). Exemplos: somadores, subtratores, deslocadores, multiplicadores, UFs combinadas (somadores/subtratores, ULAs)
- Elementos de armazenamento: registradores, banco de registradores (vários registradores, mas com limitação de portas de entrada/saída), fifos, memórias RAM (geralmente, SRAM)
- Rede de interconexão: fios, multiplexadores, barramentos + *buffers tri-state*

Projeto de Sistemas Digitais no Nível RT

Os Componentes do Nível RT



Bloco de Controle:

- Implementado por uma ou mais FSMs usando um dos seguintes métodos:
  - *Hardwired*: registrador de estados + circuitos lógicos ou
  - Usando ROM: registrador de estados + circuito comb. + um ou mais blocos ROM ou
  - Microprogramada: registrador-contador + circuito comb. + um ou mais blocos ROM (subcaso da anterior…)

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.

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 tende a rendimento do processo de fabricação ("*yield*").

Projeto de Sistemas Digitais no Nível RT

Aspectos a Serem Considerados no Projeto

2. Desempenho

- O atraso crítico determina a máxima freqüência de funcionamento.
- Para atingir uma meta de desempenho o projetista pode:
  - Escolher uma tecnologia de fabricação (CMOS) mais recente, com transitores 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*").

Aspectos a Serem Considerados no Projeto

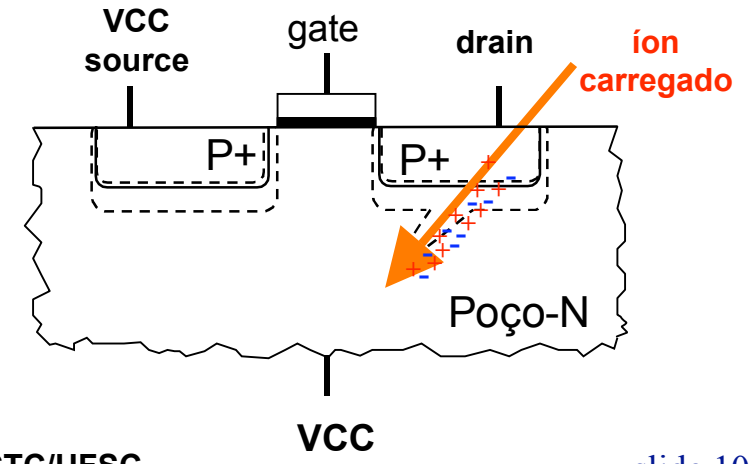
4. Testabilidade

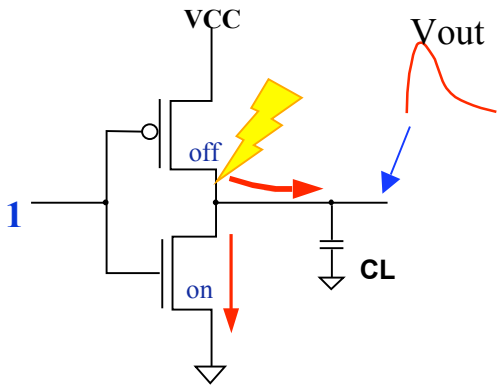
- No circuito integrado não se tem acesso aos pontos internos, apenas aos pinos de E/S.
- Geralmente, é necessário inserir modificações e até mesmo blocos de hardware que facilitem o teste do sistema digital.
- A fase de teste corresponde a aprox. 50% do custo total de desenvolvimento de um chip.

Aspectos a Serem Considerados no Projeto

5. Tolerância (ou Robustez) a Falhas

- Sistemas para aplicações críticas (aviões, satélites, usinas nucleares, freios ABS, *airbags* etc) precisam ser tolerantes a falhas (transientes e permanentes).
- Os sistemas digitais fabricados em tecnologias CMOS mais recentes (ditas nanométricas) são suscetíveis a falhas transientes oriundas da colisão de partículas carregadas (SETs e SEUs).





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 a 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

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
  saída ← P;
  pronto ← 1;
fim
```

OBS: o algoritmo poderia estar descrito em C, C++, Java, SystemC etc

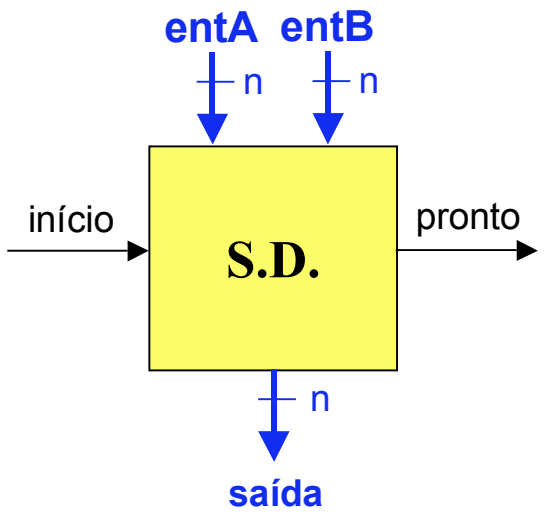
Projeto de Sistemas Digitais para Aplicações Específicas

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
  saída ← P;
  pronto ← 1;
fim
```

Interfaces



Projeto de Sistemas Digitais no Nível RT

Projeto de Sistemas Digitais para Aplicações Específicas

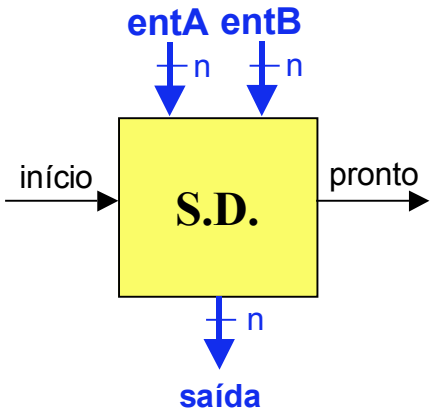
Exemplo 1: Informações contidas em um algoritmo

```
início
  pronto ← 0;                    Sinal de controle (de saída) é inicializado com zero
  A ← entA;
  B ← entB;                      Valores lidos das entradas de dados são atribuídos a variáveis
  P ← 0;                         Variável auxiliar é inicializada com zero
  Se B ≠ 0 então
  Enquanto A ≠ 0 faça            Testes (geram sinais de status para o BC)
    início
    P ← P + B;
    A ← A - 1;                   Variáveis são usadas em operações aritméticas
    fim
  saída ← P;                     Resultado da operação é disponibilizado na saída de dados
  pronto ← 1;                    Sinal de controle (de saída) é setado para indicar o término
fim
```

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
  saída ← P;
  pronto ← 1;
fim
```



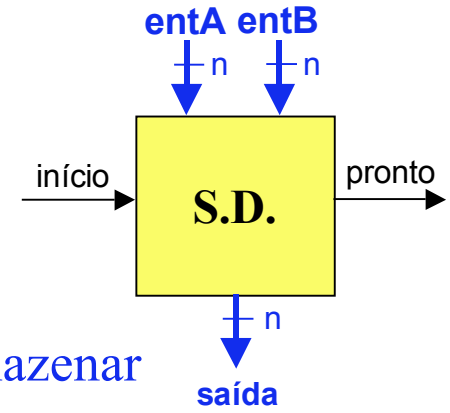
Um algoritmo contém informações sobre:

- 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.)

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
  saída ← 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, saída, 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

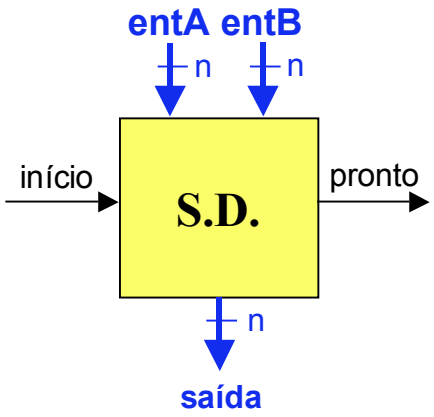
```
início
  pronto ← 0;
  A ← entA;
  B ← entB;
  P ← 0;
  Se B ≠ 0 então
  Enquanto A ≠ 0 faça
    início
    P ← P + B;
    A ← A - 1;
    fim
  saída ← P;
  pronto ← 1;
fim
```

- 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:

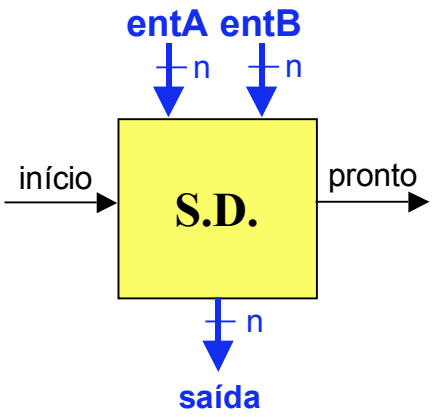
- O SD possua duas entradas de dados
- O custo de implementação deve ser mínimo
- O SD não precisa ter alto desempenho (e não há restrição quanto ao desempenho mínimo necessário)



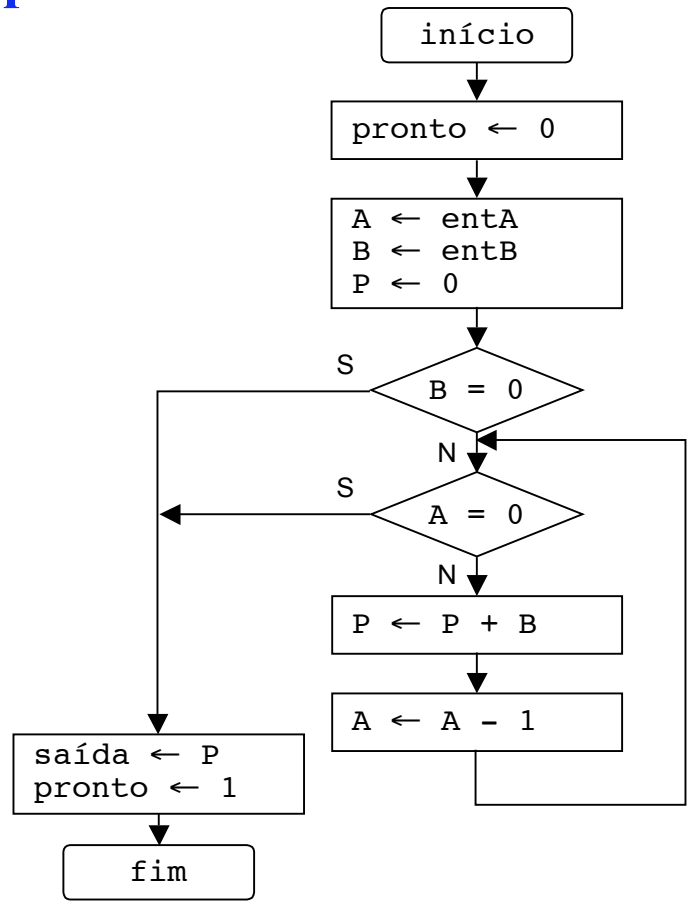

```
início
  pronto ← 0;
  A ← entA;
  B ← entB;
  P ← 0;
  Se B ≠ 0 então
  Enquanto A ≠ 0 faça
    início
    P ← P + B;
    A ← A - 1;
    fim
  saída ← P;
  pronto ← 1;
fim
```

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
  saída ← P;
  pronto ← 1;
fim
```

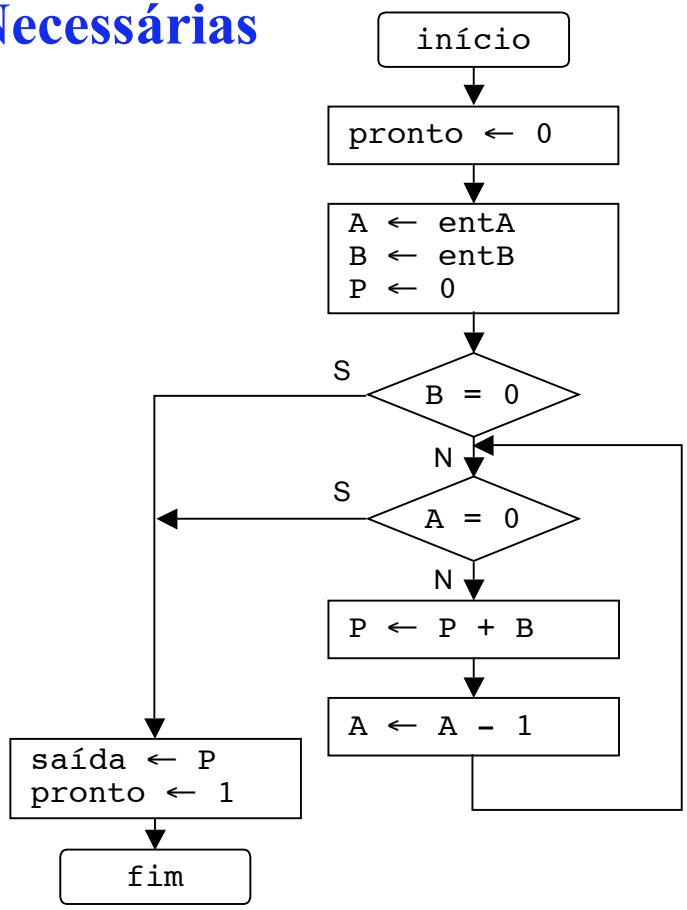


- Iremos assumir que somente uma operação ocorre por ciclo de relógio
- As operações que podem ocorrer em paralelo estão em uma mesma caixa…

Projeto do Bloco Operativo Visando Custo Mínimo

Solução 1: Unidades Funcionais (UFs) Necessárias

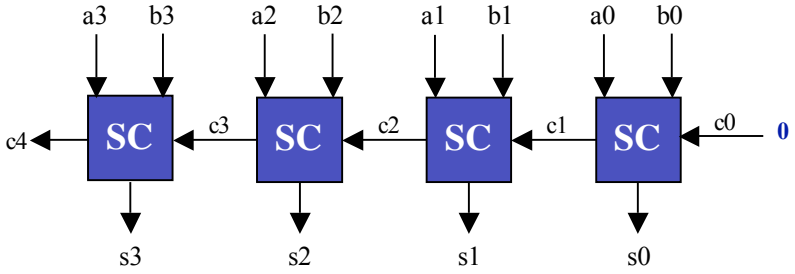
- Operações necessárias: “+” e “-” (na verdade, seria um decremento, mas vamos assumir subtração)
- As operações “+” e”-” são usadas em ciclos de relógio diferentes. Logo, poderemos usar **um somador/subtrator**, que é mais barato que um somador mais um subtrator



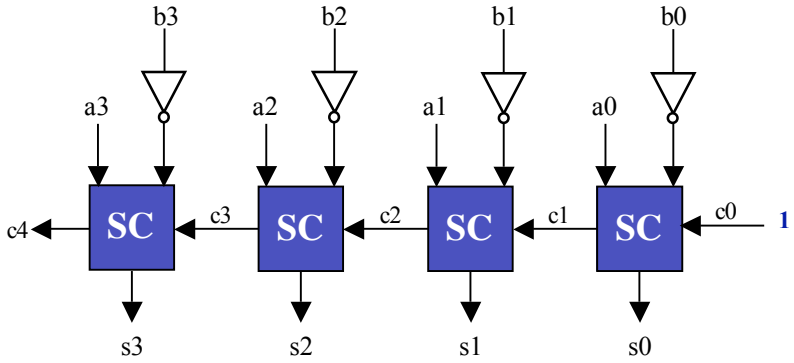
Projeto do Bloco Operativo Visando Custo Mínimo

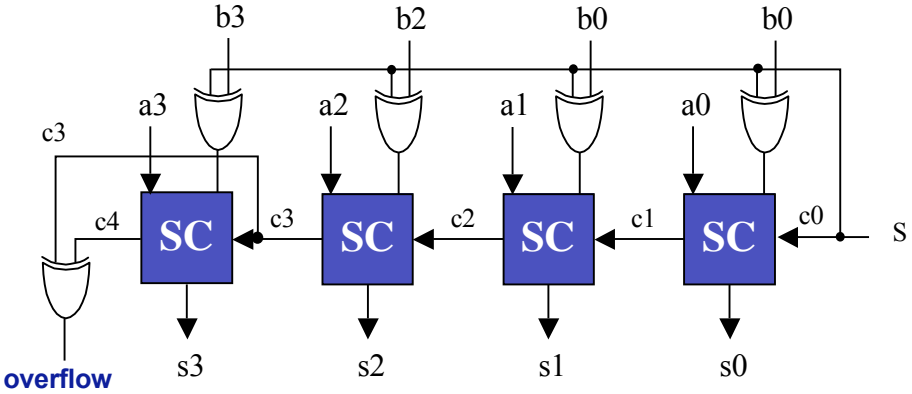
Solução 1: Custo de UFs *Versus* Custo de UFs Combinadas

Somador de 4 bits



Subtrator de 4 bits

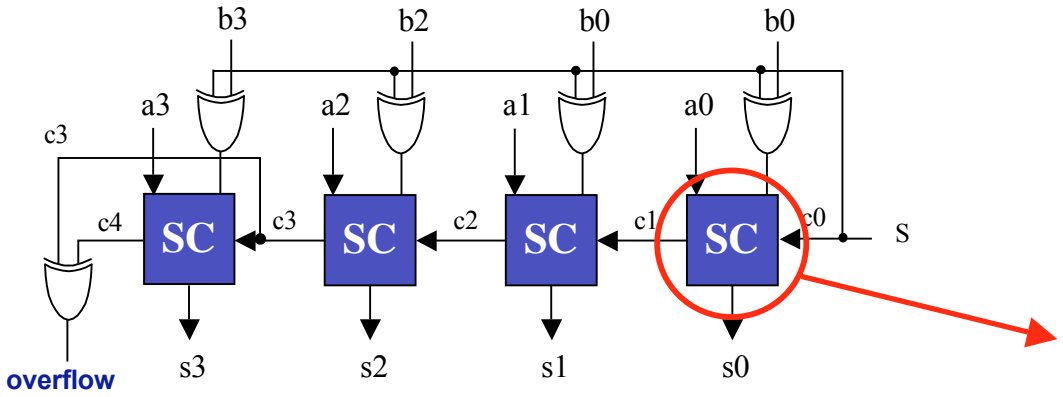


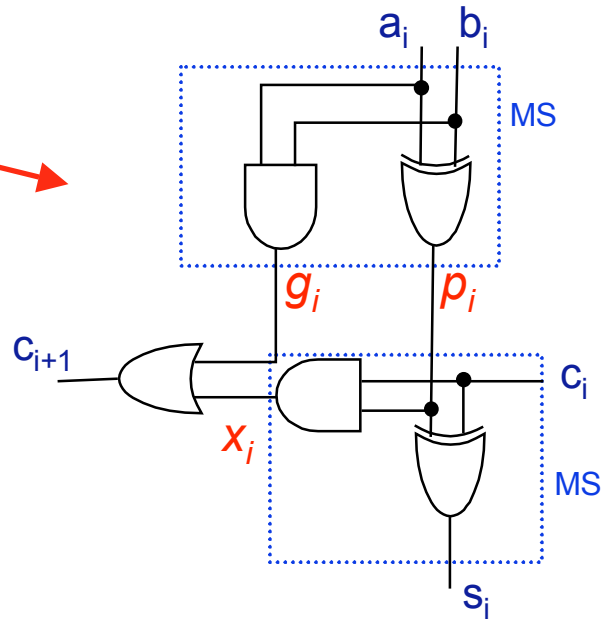


Somador/Subtrator de 4 bits

Projeto do Bloco Operativo Visando Custo Mínimo

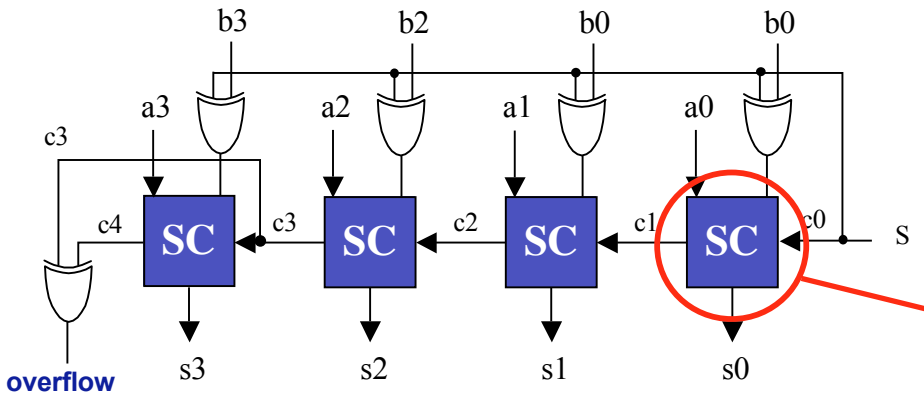
Calculando o Custo do Somador/Subtrator



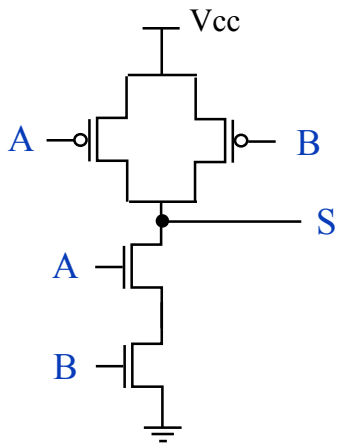


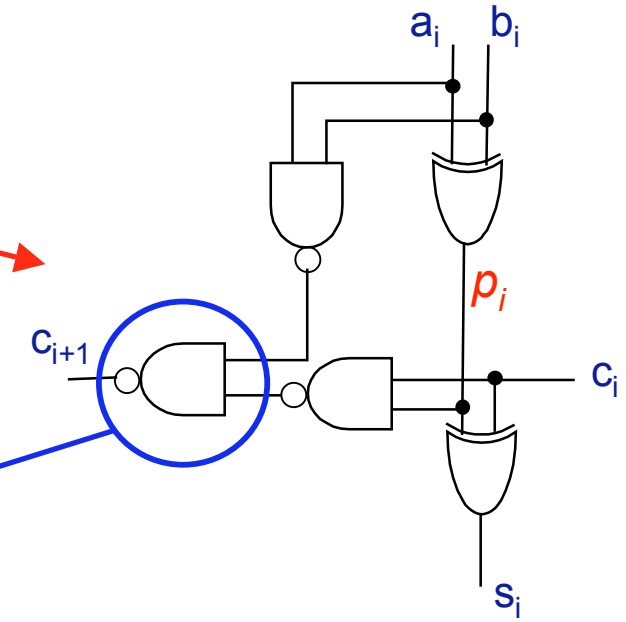
Projeto do Bloco Operativo Visando Custo Mínimo

Calculando o Custo do Somador/Subtrator



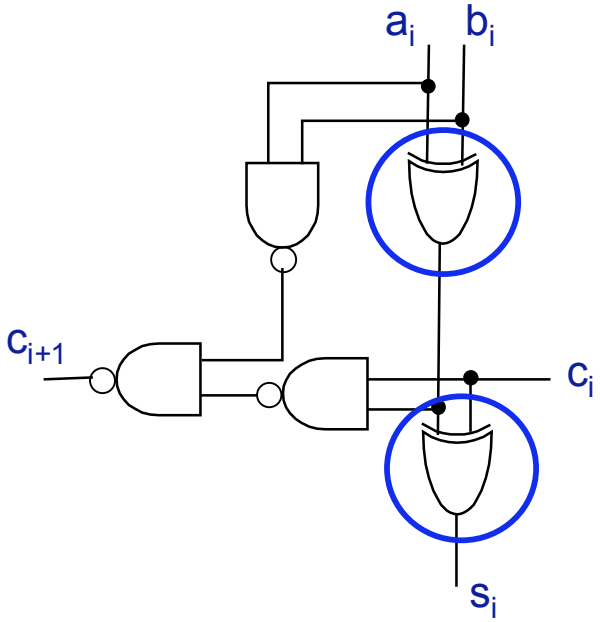
Mas em CMOS…



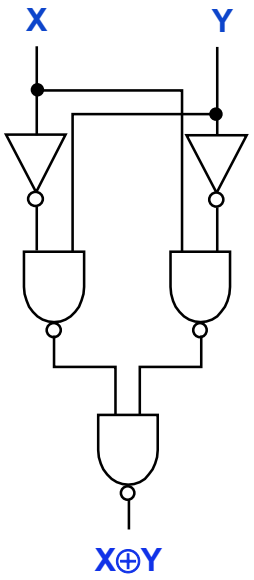


Projeto do Bloco Operativo Visando Custo Mínimo

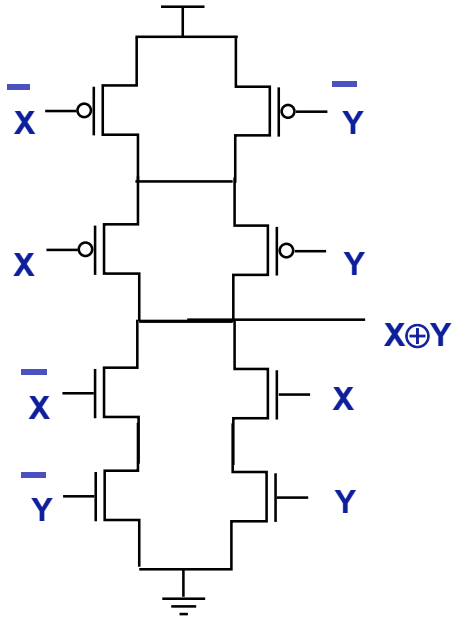
Calculando o Custo do Somador/Subtrator



Algumas Implementações CMOS para a xor



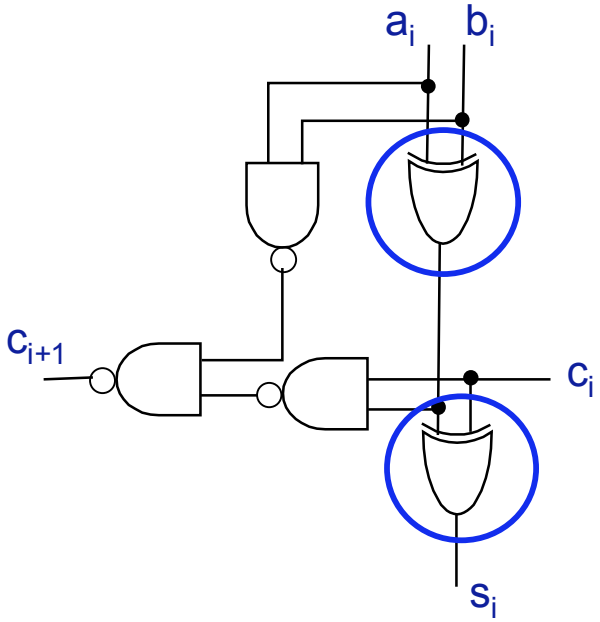
16 transistores



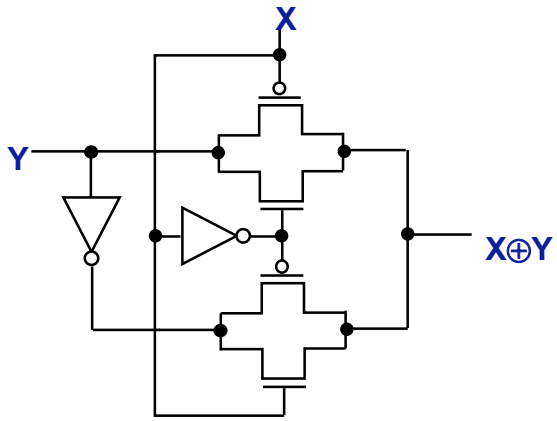
12 transistores
(necessita de 2 inversores)

Projeto do Bloco Operativo Visando Custo Mínimo

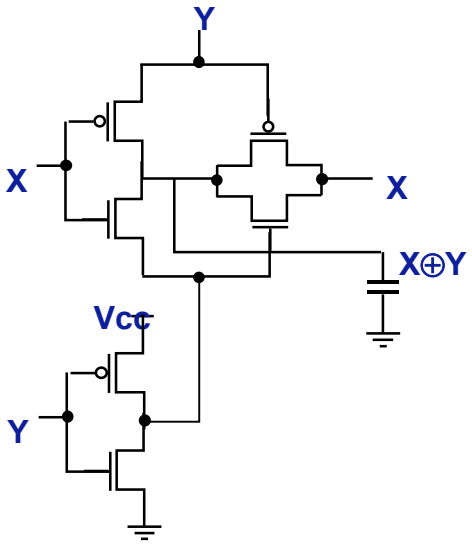
Calculando o Custo do Somador/Subtrator



Algumas Implementações CMOS para a xor



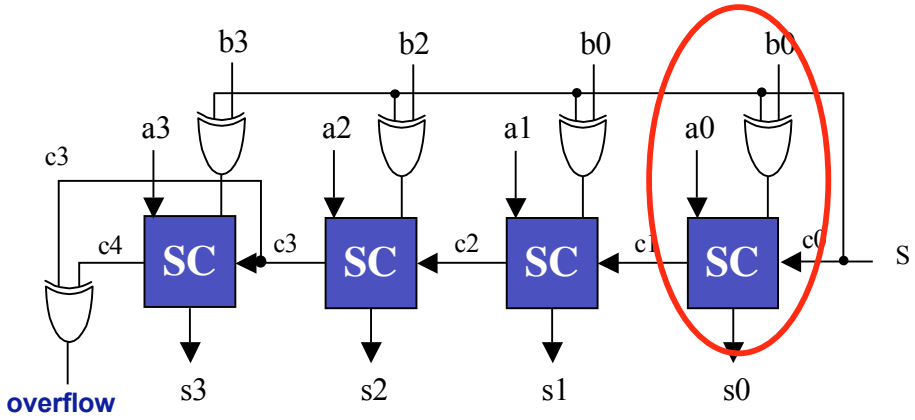
8 transistores



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:

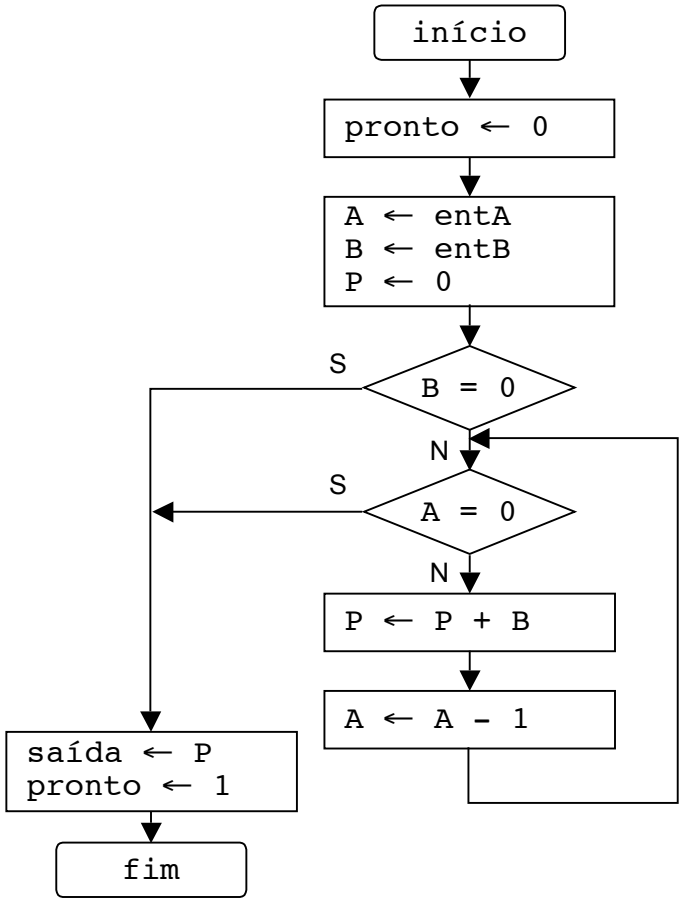
- 3 portas xor: 3 x 6 = 18 transistores
- 3 portas nand de duas entradas: 3 x 4 = 12 transistores
- Logo, custo de um bit = 30 transistores (ignorando-se a xor que calcula o overflow)

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 precisar, no máximo, três registradores



Projeto do Bloco Operativo Visando Custo Mínimo

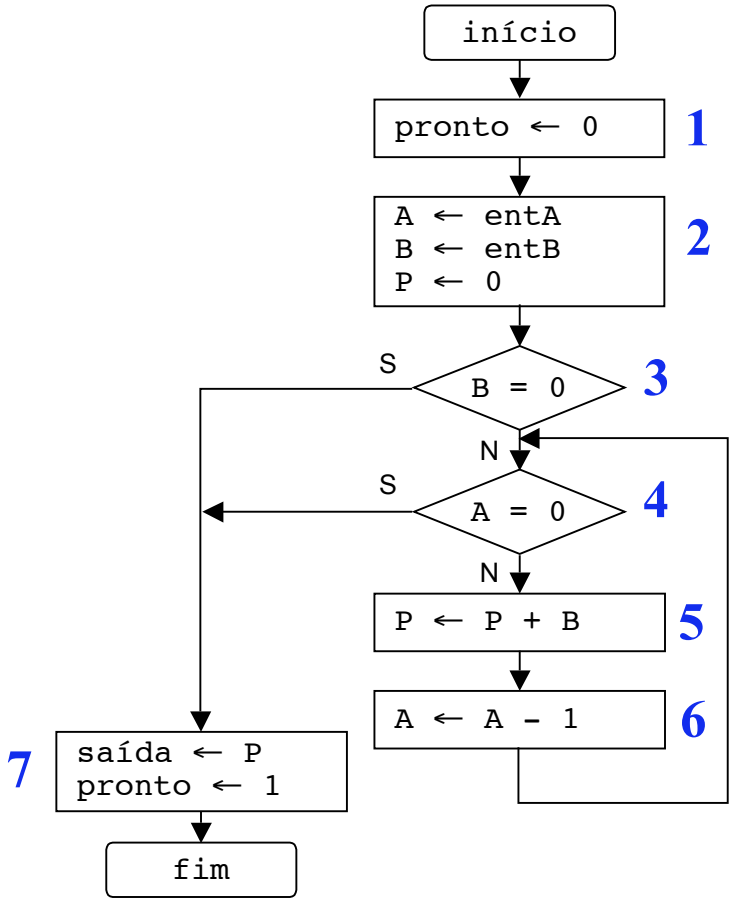
Solução 1: Registradores

Análise do tempo de vida das variáveis:

	1	2	3	4	5	6	7
A			X	X	X	X	
B			X	X	X	X	
P			X	X	X	X	X

as 3 variáveis são escritas no final do passo 2

Existe ao menos um passo no qual as 3 variáveis **"estão vivas".** Logo, são necessários **3 registradores.** Chamemo-los de **A, B e P**

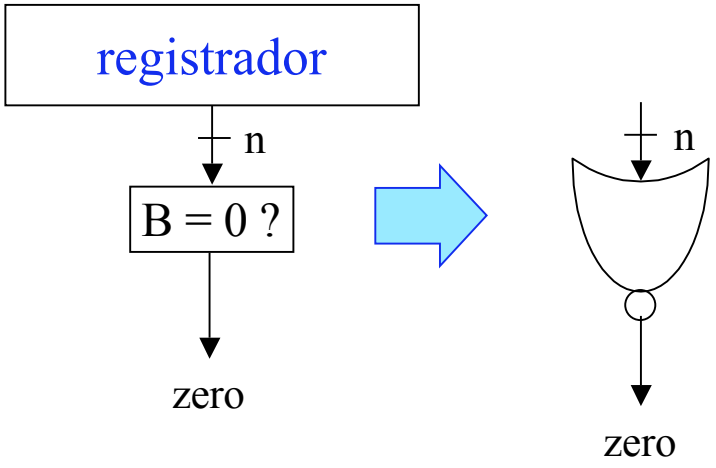


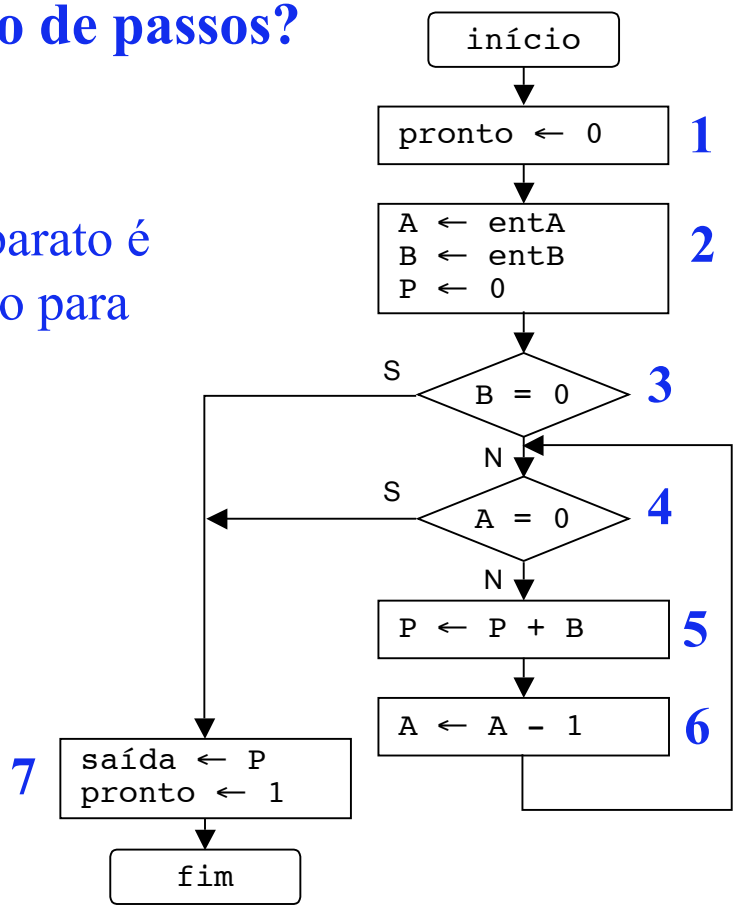
Projeto do Bloco Operativo Visando Custo Mínimo

Dúvida: não dá para reduzir o número de passos?

Resposta: sim!

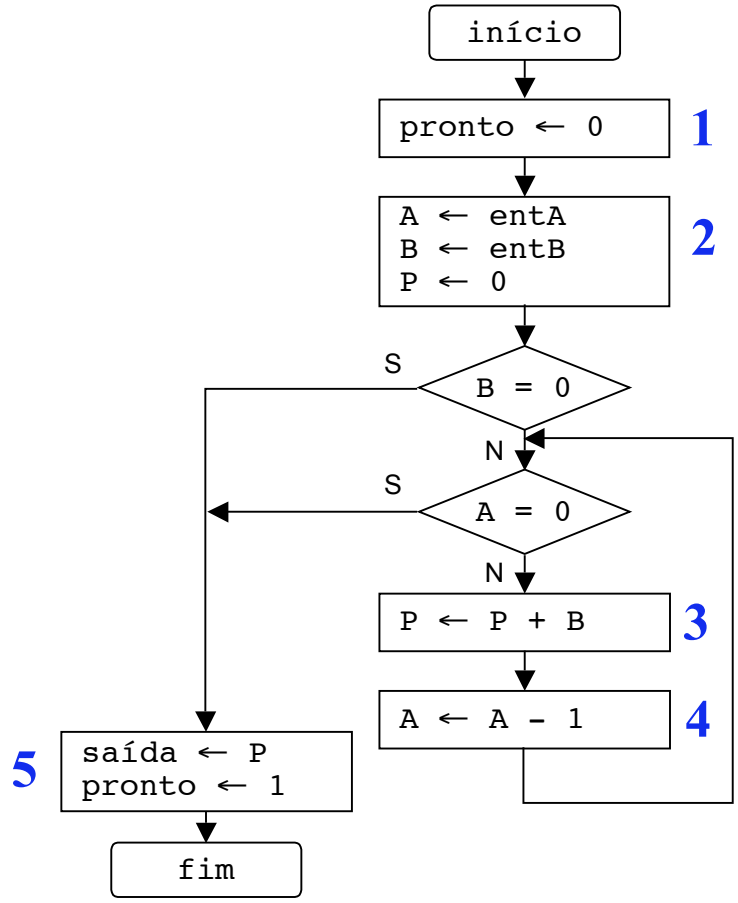
No caso de comparações com zero, o mais barato é utilizar um circuito combinacional específico para isto:





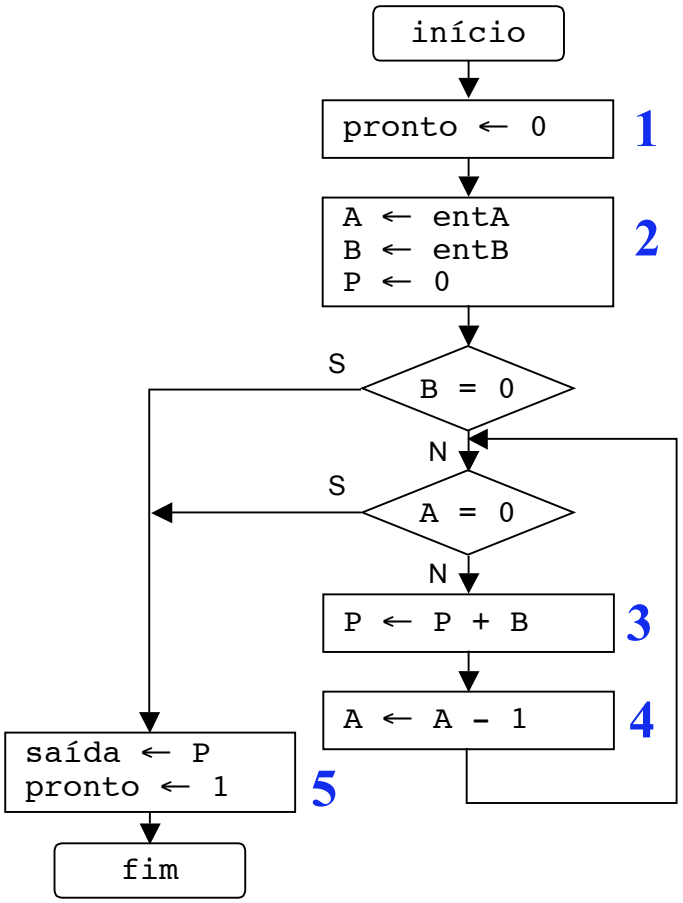
Projeto do Bloco Operativo Visando Custo Mínimo

Solução 1: Novo Fluxograma…



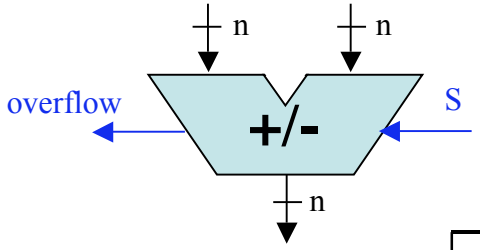
Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco Operativo Visando Custo Mínimo



Solução 1: elementos para o BO:

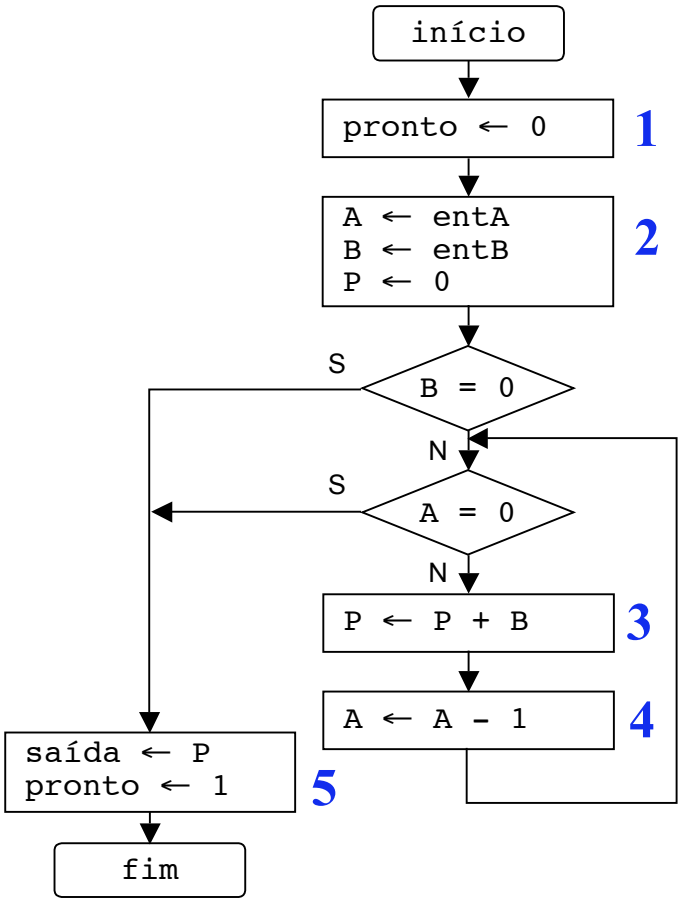
- 1 somador/subtrator
- 3 registradores com carga paralela, A, B e P, sendo P com reset assíncrono
- Rede de interconexão apropriada

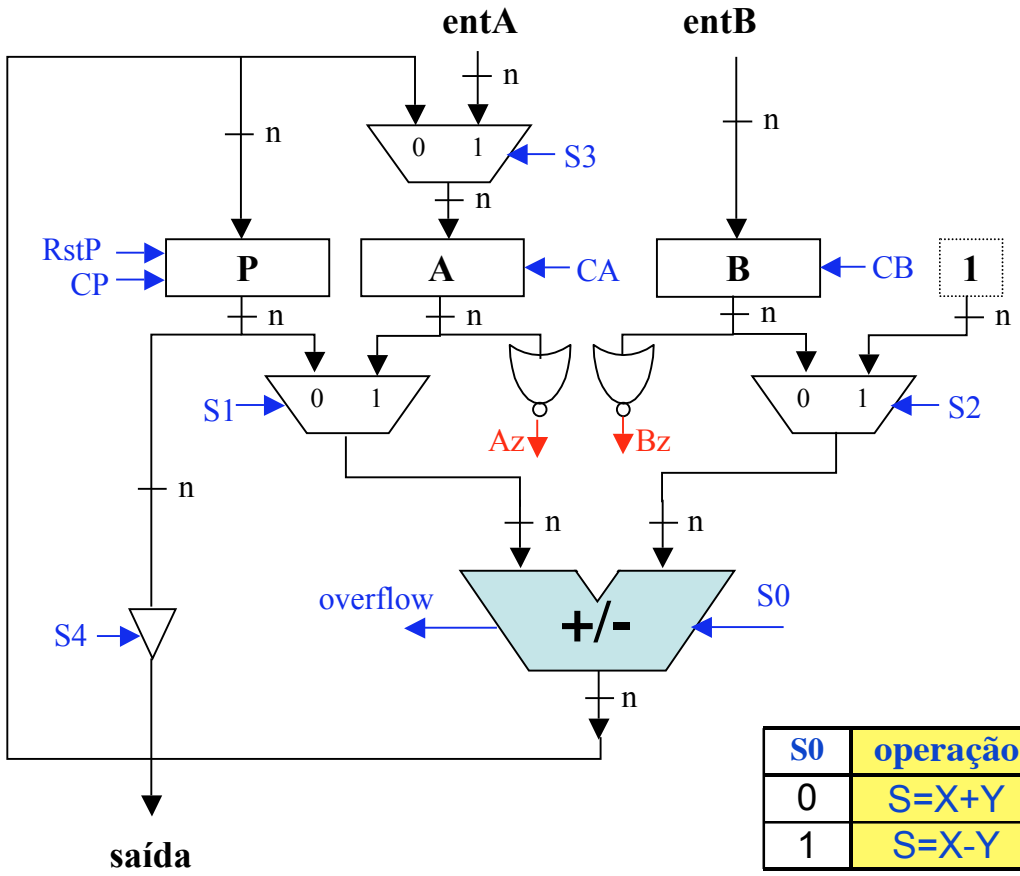


S	operação
0	S=X+Y
1	S=X-Y

Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco Operativo Visando Custo Mínimo

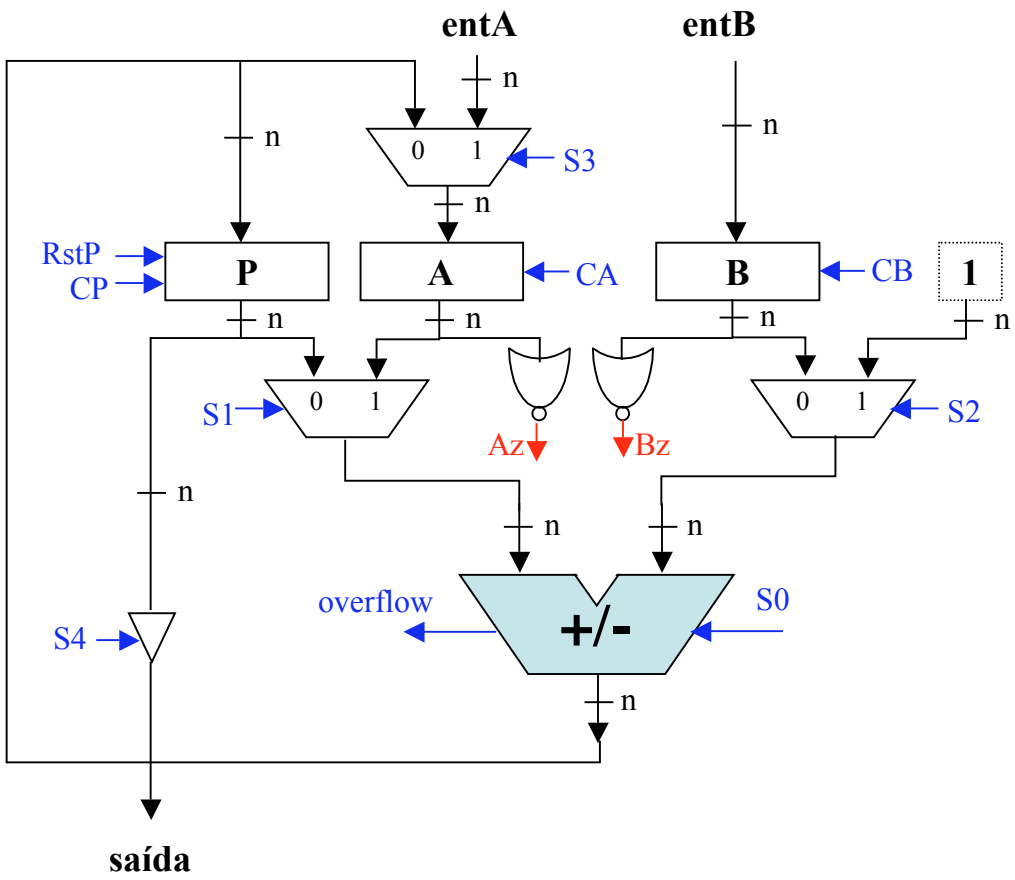




S0	operação
0	S=X+Y
1	S=X-Y

Projeto de Sistemas Digitais no Nível RT

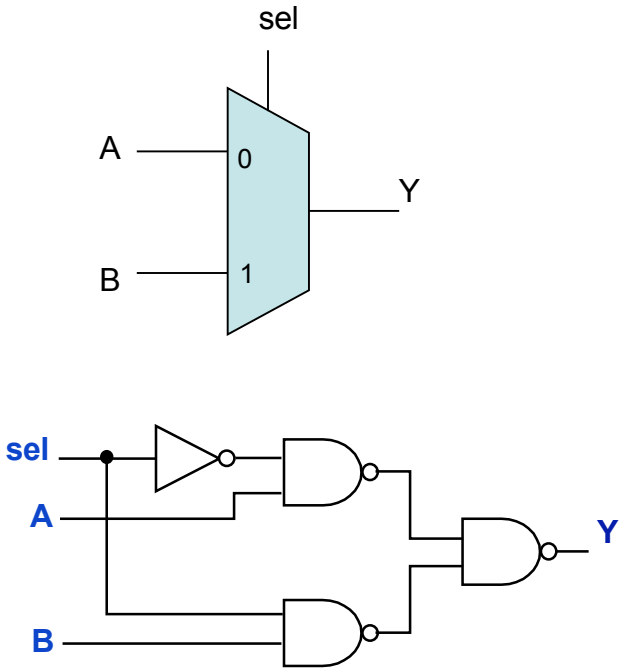
Cálculo do Custo do Bloco Operativo



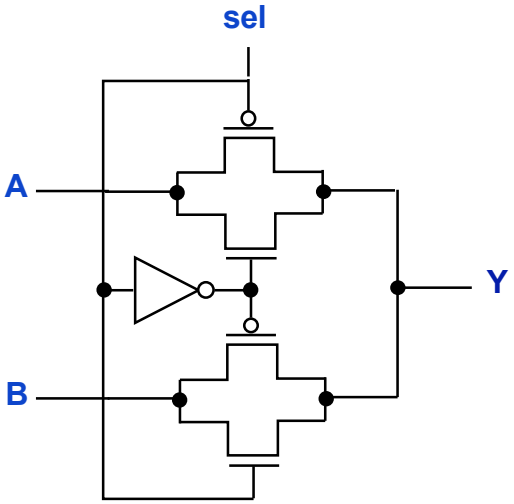
- somador/subtrator de n bits: 30n transistores
- E o resto?

Estimativa do Custo do Bloco Operativo

Custo do Mux 2:1 em CMOS



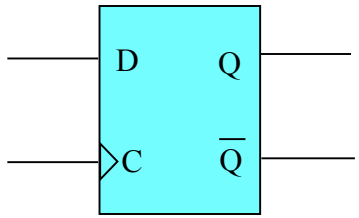
14 ou 12 transistores



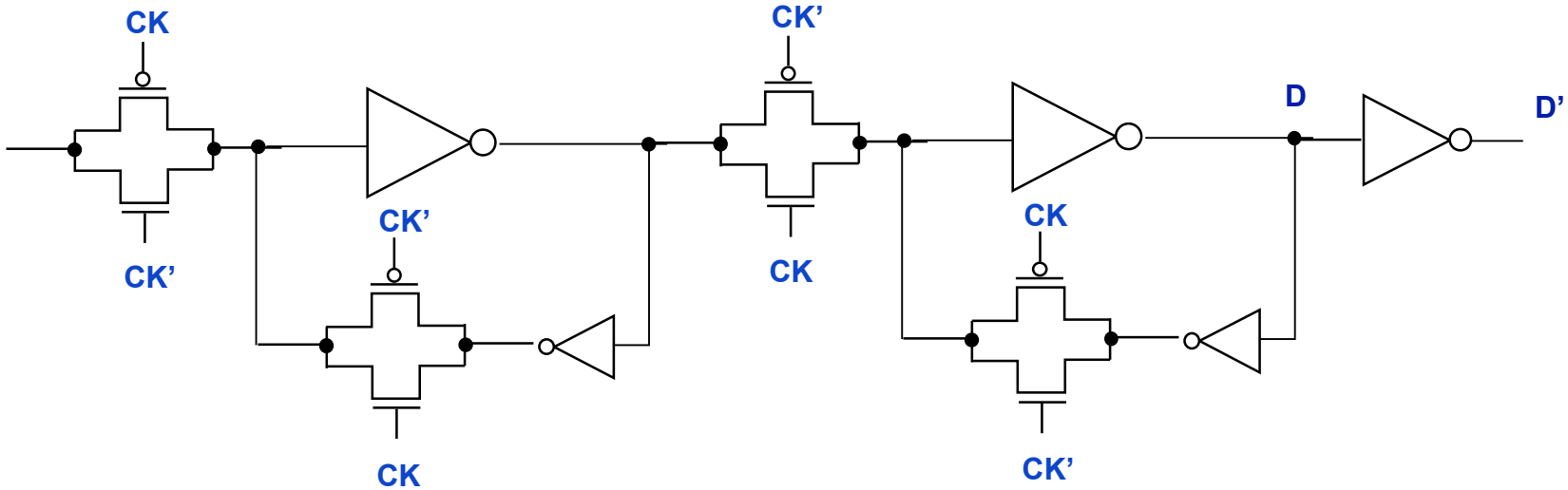
6 ou 4 transistores
(mais usado)

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)

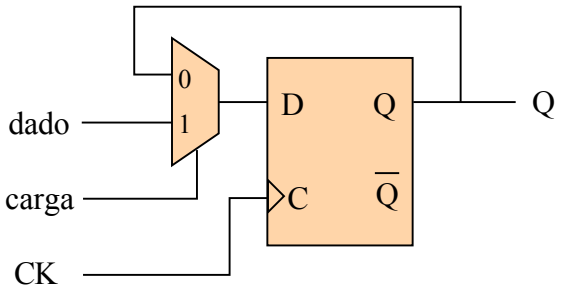


OBS: para set ou reset assíncrono, adicionar 2 transistores

Estimativa do Custo do Bloco Operativo

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

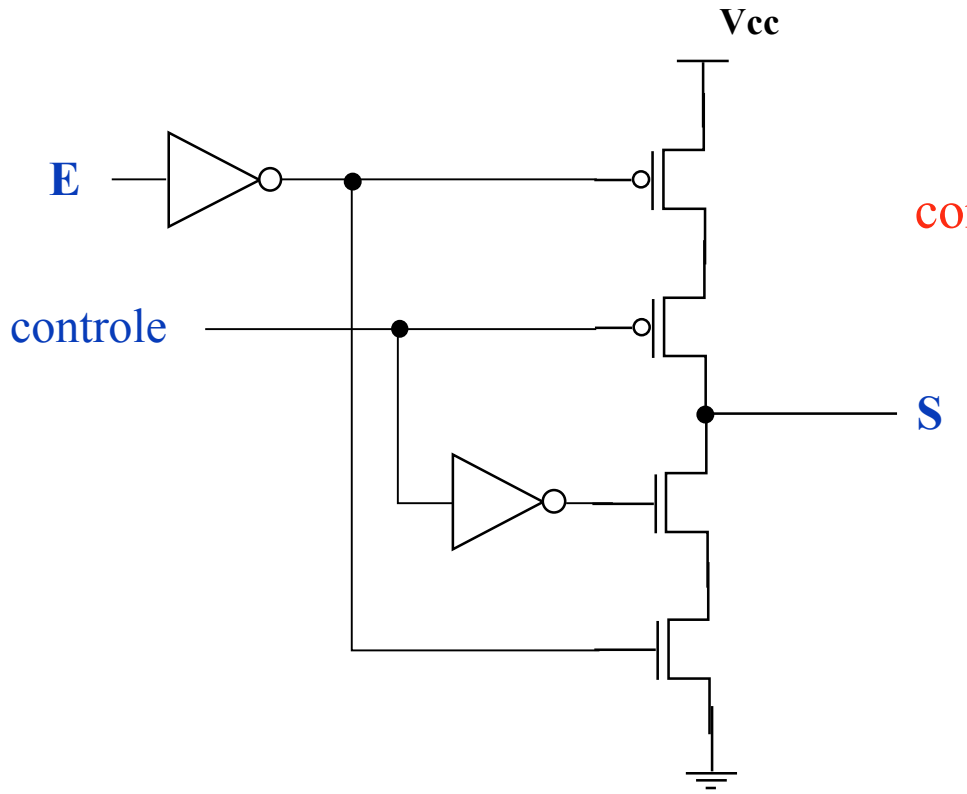
18+4= 22 transistores



OBS: para set ou reset assíncrono, adicionar 2 transistores

Estimativa do Custo do Bloco Operativo

Custo de um *Buffer Tri-State* (Não-inversor)



6 ou 8 transistores
(eventualmente, podemos considerar somente um inversor para controlar n *buffers*)

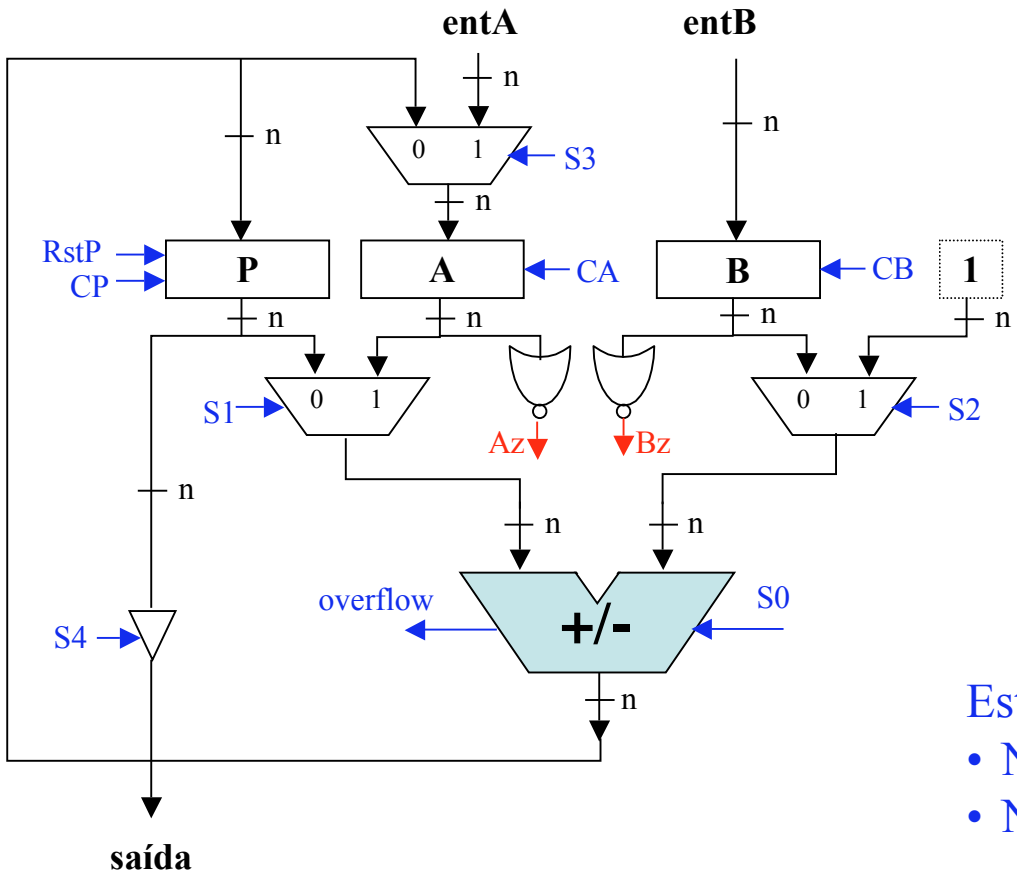
Estimativa do Custo do Bloco Operativo

Resumo

Componente RT	Custo
Somador	24n
Subtrator	26n
Somador/subtrator	30n
Mux 2:1	4n
Registrador com carga paralela (+2 transistores para set ou reset assíncrono)	18n
Registrador com carga paralela controlada (+2 transistores para set ou reset assíncrono)	22n
Buffer tri-state não inversor	6n

Projeto de Sistemas Digitais no Nível RT

Estimativa do Custo do Bloco Operativo



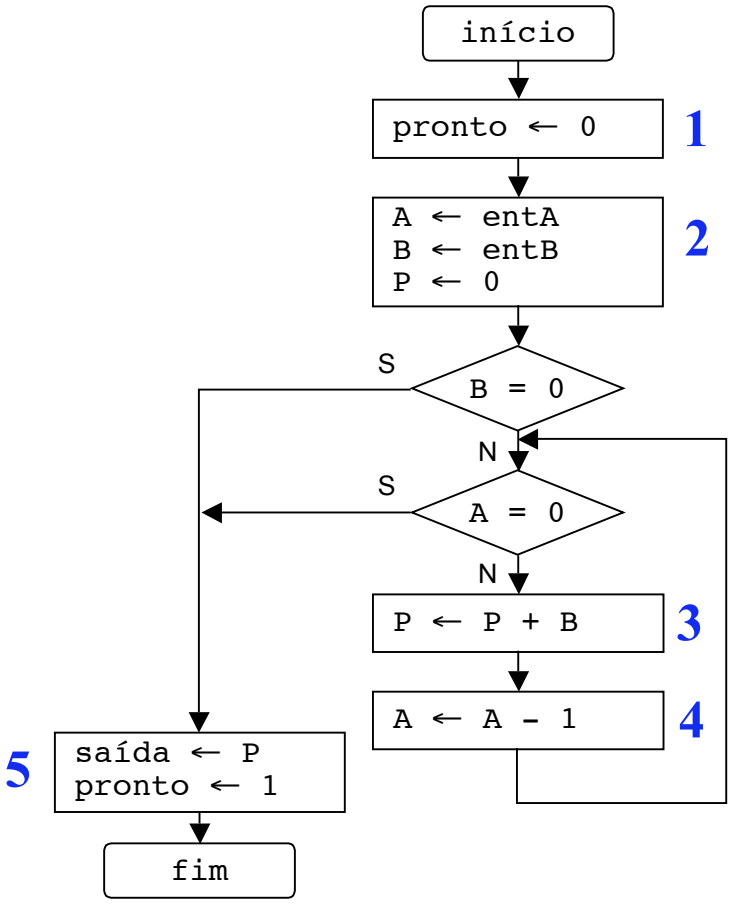
Custo do BO 1	Custo
1 Somador/subtrator	30n
3 Muxes 2:1	3x4n=12n
2 Registradores com carga paralela controlada	2x22n=44n
1 Registrador com carga paralela controlada e reset assíncrono	24n
1 conjunto de *buffers tri-state* não inversores	6n
Total	**116n**

Estimativa de custo para o BC:

- Número de estados: 5 ou 6
- Número de sinais de controle = **9**

Projeto de Sistemas Digitais no Nível RT

Estimativa do Desempenho do Bloco Operativo



Se n = 4 bits:

- Maior inteiro sem signal: 15 (⇒1111)
- Pior caso: A=15, B≠0
- Seqüência de execução: 1, 2, 15x(3,4), 5 = 33 passos (**33 ciclos de relógio**)

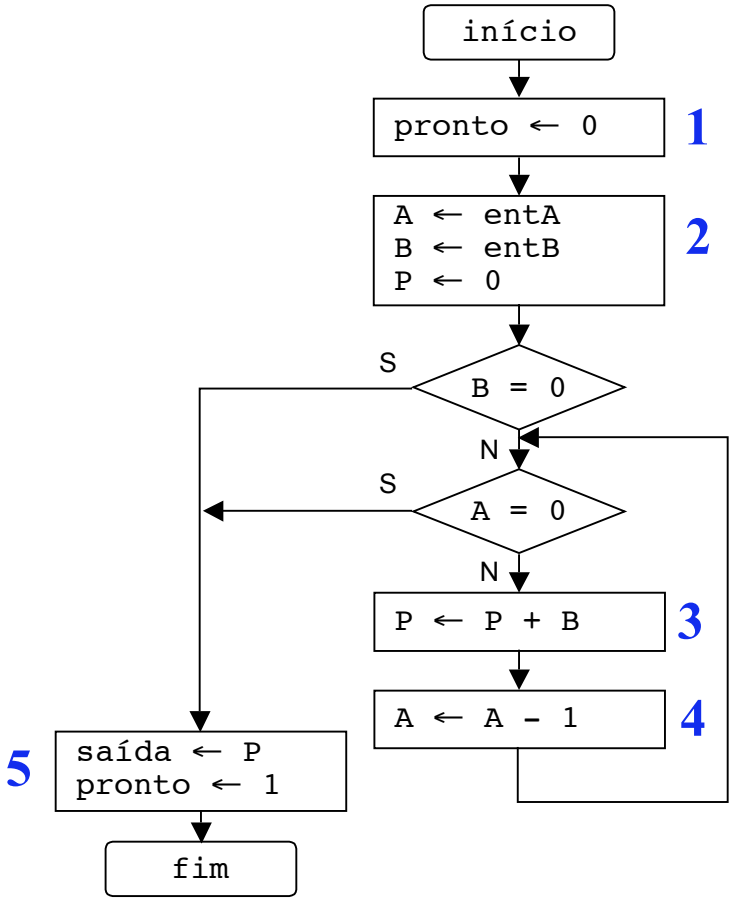
Generalizando para n bits:

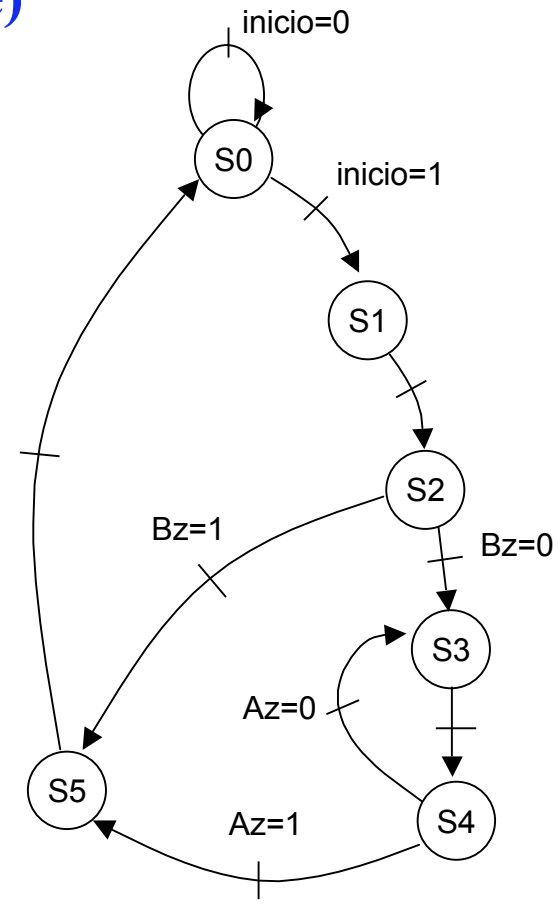
- Maior inteiro sem signal: 2^n-1
- Pior caso: A= 2^n-1 , B≠0
- Seqüência de execução: 1, 2, (2^n-1)x (3,4), 5 = (2^{n+1}-2)+3 passos (=~ **2^{n+1} ciclos de relógio**)

Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco de Controle Visando Custo Mínimo

Diagrama de Estados (Assumindo Moore)

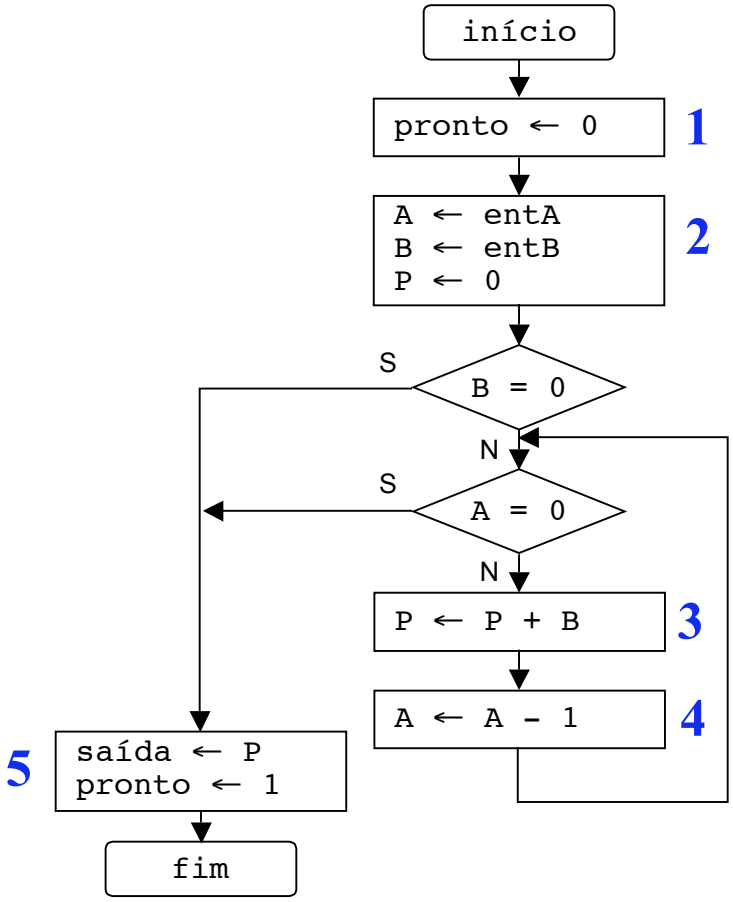




Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco de Controle Visando Custo Mínimo

Tabela de Transição de Estados (Assumindo Moore)

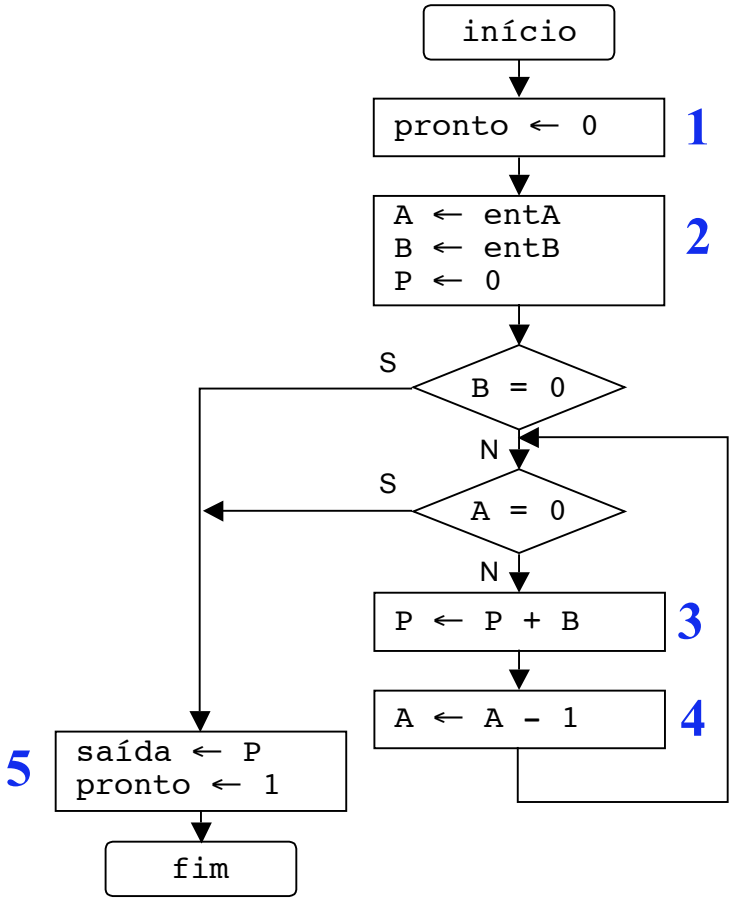


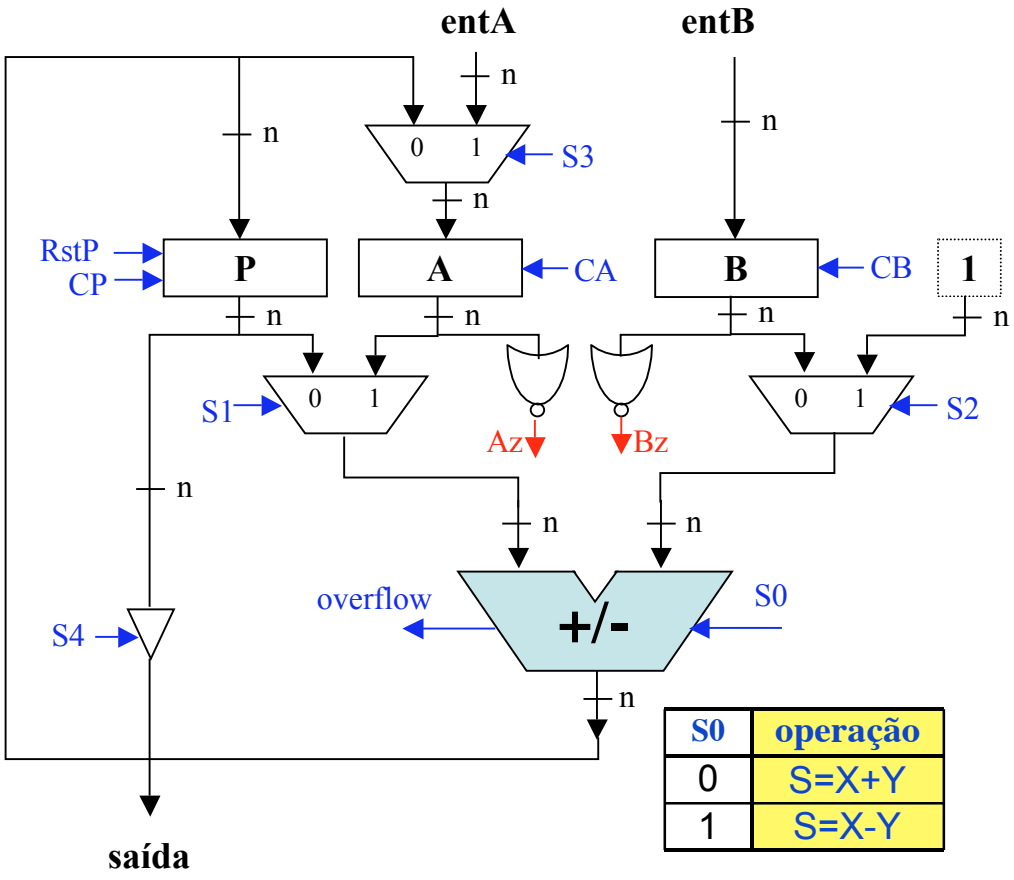
Estado atual	Entradas			Próx. Estado
	início	BZ	AZ	
S0	0	-	-	S0
	1	-	-	S1
S1	-	-	-	S2
S2	-	0	-	S3
	-	1	-	S5
S3	-	-	-	S4
S4	-	-	0	S3
	-	-	1	S5

Projeto de Sistemas Digitais no Nível RT

Projeto do Bloco de Controle Visando Custo Mínimo

Tabela de Saídas (Assumindo Moore)



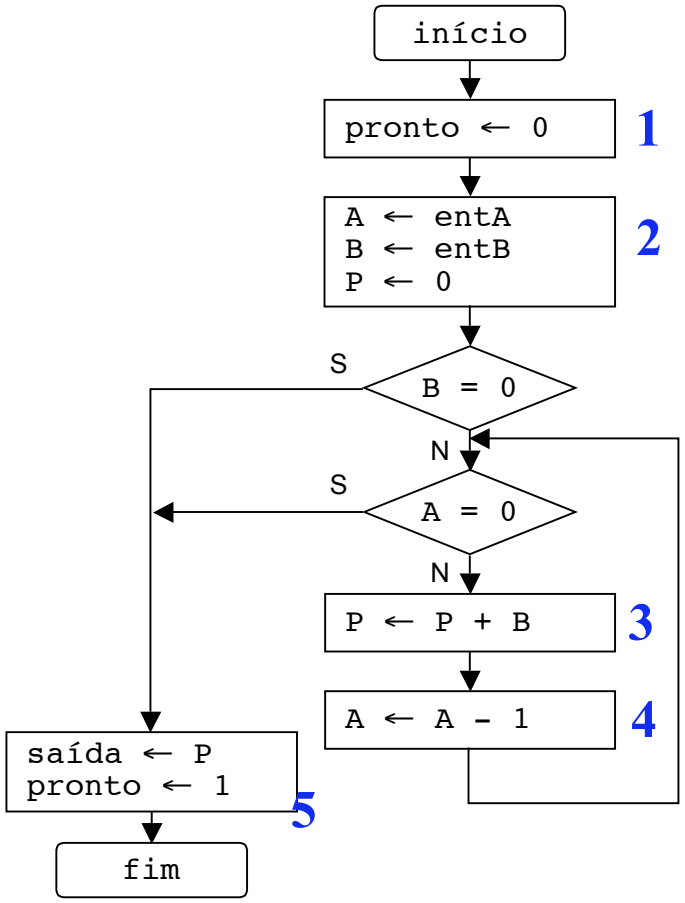


S0	operação
0	S=X+Y
1	S=X-Y

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	S3	CA	CB	S1	S2	S0	S4	pronto
S0	0	0	-	0	0	-	-	-	0	-
S1	0	0	-	0	0	-	-	-	0	0
S2	1	0	1	1	1	-	-	-	0	0
S3	0	1	-	0	0	0	0	0	0	0
S4	0	0	0	1	0	1	1	1	0	0
S5	0	0	-	0	0	-	-	-	1	1

1 sinal 1 sinal 1 sinal

RstP = CB = S3
S1 = S2 = S0
S4 = pronto

5 sinais