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

Sistemas Digitais

INE 5406

Aula 3-T

2. Processadores Dedicados (Blocos Aceleradores). O modelo BO/BC. Máquinas Sequenciais Síncronas. Método de Projeto no Nível RT. Exemplo 1 de somadores sequenciais.

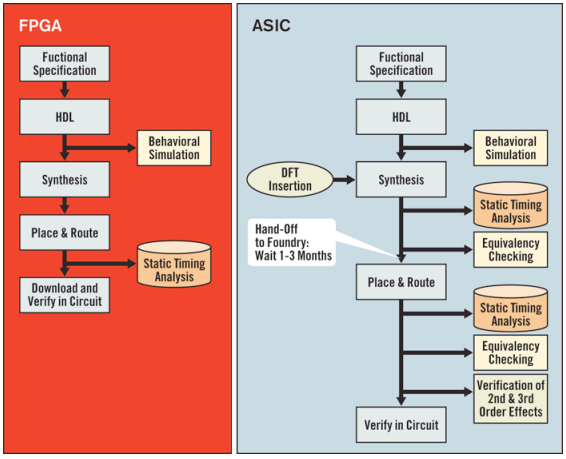
Profs. José Luís Güntzel e Cristina Meinhardt
{j.guntzel, cristina.meinhardt}@ufsc.br

Formas de Implementação do Sistema Digital

FPGAs:

- Chip pronto: projeto termina com a geração do arquivo que **configura o chip**
- É possível reconfigurar muitas vezes
- Desempenho e consumo de energia subótimos
- Intel (ex-Altera), Xilinx, Actel, Lucent

Nestes fluxos, HDL = Hardware Description Language: VHDL, Verilog, SystemC, SystemVerilog



Fonte: Xilinx

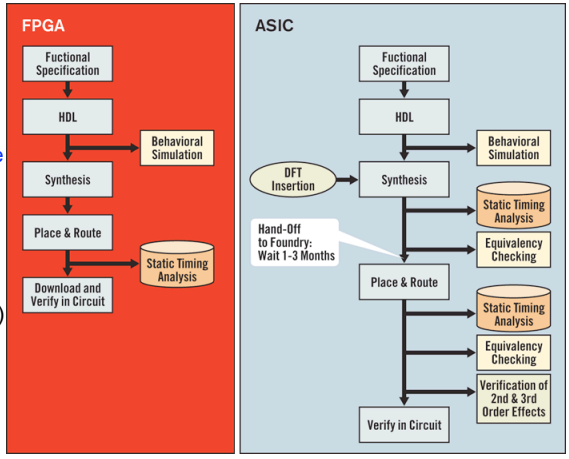
ASIC:

- Chip é **fabricado do zero**, exclusivamente para o encomendante: **projeto termina com a descrição das máscaras a serem usadas na fabricação**
- Se houver erro de projeto, fabricar de novo!
- Mais liberdade para otimizar desempenho e consumo
- *Foundries*: TSMC, GlobalFoundries, Samsung Electronics, UMC, SMIC, Intel (somente para seus próprios produtos)

Formas de Implementação do Sistema Digital

FPGAs:

- Custo: número de "Logic Elements"(ou "Lookup Tables – LUTs" ou "Logic Blocks") + número de bits de registradores + quantidade de blocos de memória + blocos de somadores etc)
- Desempenho: frequência máxima (período mínimo) do *clock*, obtida por análise de timing (*Static Timing Analysis*)



Fonte: Xilinx

ASIC:

- Custo: **número de transistores**
- Desempenho: frequência máxima (período mínimo) do *clock*, obtida por análise de timing (*Static Timing Analysis*)

Na parte teórica desta disciplina, assumiremos que os sistemas digitais serão implementados desta forma, pois ela é mais genérica do que FPGAs.

Requisitos e Especificações do Projeto

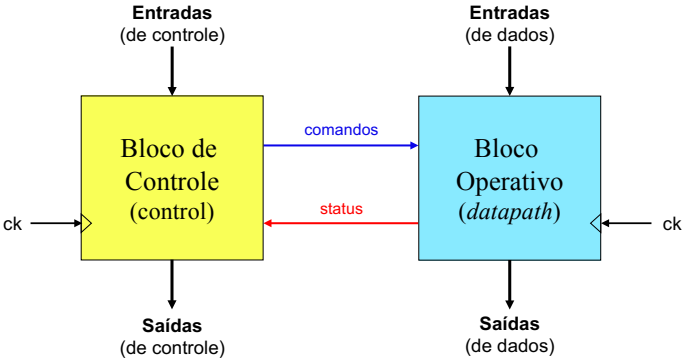
1. Qual é o custo máximo admitido?
2. Qual é o desempenho (velocidade) mínimo requerido?
3. Qual é o consumo máximo de energia admitido?
4. Qual é a potência máxima a ser dissipada?

Nesta disciplina, consideraremos apenas estes dois requisitos

Em geral, a otimização simultânea das variáveis custo, desempenho, consumo de energia e potência dissipada é difícil, pois elas são conflitantes.

O Modelo BO/BC

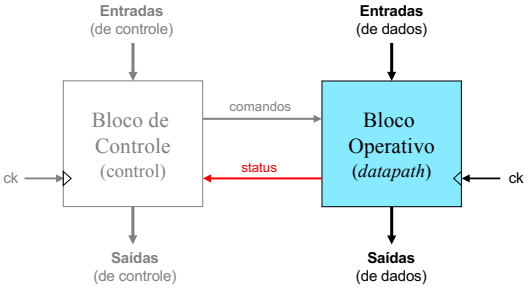
O Modelo Bloco Operativo / Bloco de Controle



Este é um modelo simples de sistemas digitais, porém didático e aplicável na maioria dos casos.

O Modelo BO/BC

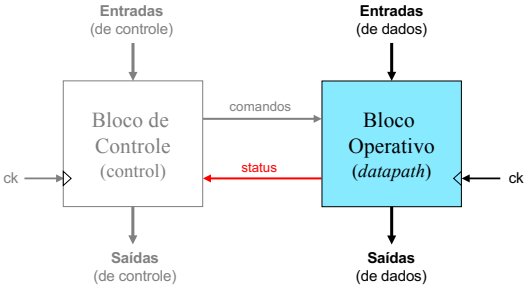
Bloco Operativo: funções



- Realiza transformações sobre dados, geralmente provenientes do ambiente 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 sequência de operações a serem realizadas (às vezes são chamados de "*flags*")

O Modelo BO/BC

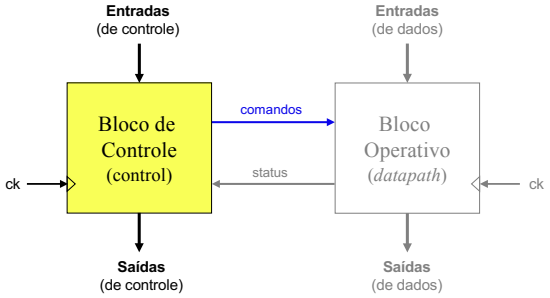
Bloco Operativo: componentes



- Unidades Funcionais (UFs): somadores, subtratores, deslocadores, multiplicadores, UFs combinadas (somadores/subtratores, ULAs)
- Elementos de armazenamento: registradores ou banco de registradores, memórias (SRAM)
- Rede de interconexão: fios, multiplexadores, barramentos + *buffers tri-state*

O Modelo BO/BC

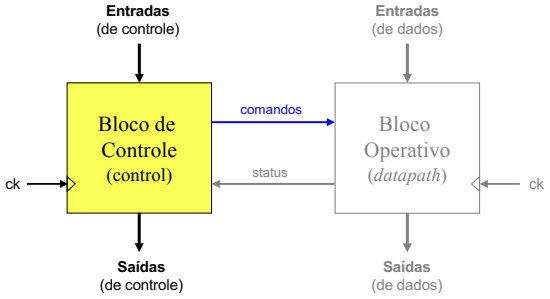
Bloco de Controle: funções



- Gera comandos, que são sinais de controle na ordem necessária para que o bloco operativo realize os passos desejados
- Recebe sinais de controle do ambiente externo: opcode, no caso de CPUs, sinais específicos (por exemplo, "iniciar"), no caso de CPUs dedicadas e de blocos aceleradores
- Pode gerar uma ou mais saídas de controle para se comunicar com outros sistemas digitais (p. ex.: "done", "bus request", "ack")

O Modelo BO/BC

Bloco de Controle: componentes



- Em um esquema monociclo: é um bloco combinacional
- Em um esquema multiciclo: é uma máquina de estados (FSM – Finite State Machine)

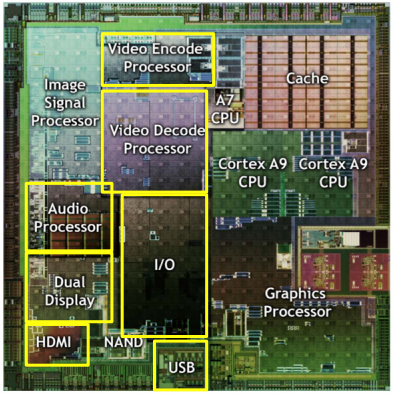
Processadores Dedicados

Classificação dos Sistemas Digitais Quanto à Aplicação

2. Processadores Dedicados ou Blocos Aceleradores (*Single-Purpose Processors*)

- Projetados para executar somente um **algoritmo específico** (uma aplicação)
- A aplicação é implementada via **hardware**
- **Alta eficiência energética & execução em tempo real**
- Exemplos: codecs de fotos e vídeos (jpeg, MPEG, H.264/AVC, VP9, HEVC), codecs de áudio, cifradores/decifradores (criptografia) etc

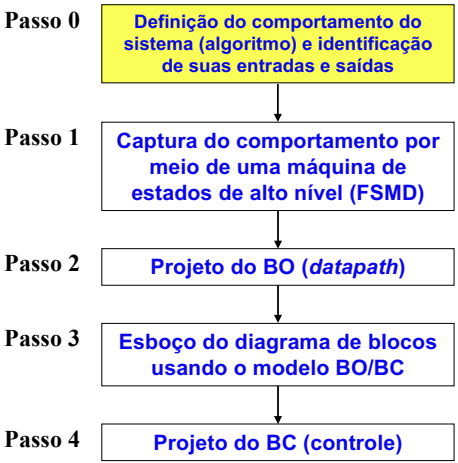
Tegra 2 (Nvidia)



Fonte: https://www.bdti.com/InsideDSP/2011/10/20/NvidiaQualcomm

Projeto no Nível RT

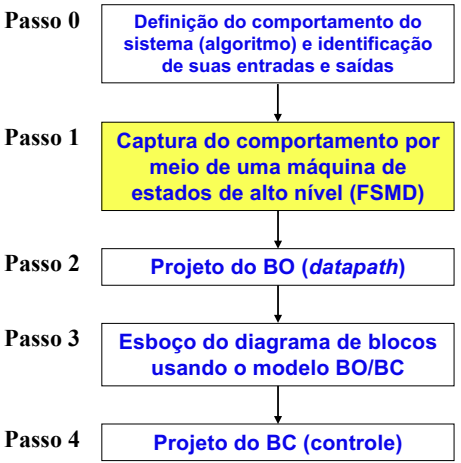
Método de Projeto



- O comportamento do sistema pode ser descrito textualmente, em linguagem humana, ou pode ser descrito sob a forma de um algoritmo, usando uma linguagem no nível de sistema, como **SystemC**, **SystemVerilog** ou mesmo **C/C++.** Esta segunda opção é preferencial para sistemas muito complexos.
- Dependendo do nível de abstração (RT ou sistema), a identificação das interfaces pode envolver o número de bits dos sinais ou os tipos de dados dos sinais (inteiros sinalizados ou não, ponto-flutuante etc).
- Sistemas muito complexos podem requerer simulação de alto nível. Neste caso, a descrição do sistema completo em SystemC, SytemVerilog ou C/C++ é imprescindível.
- Caso exista mais de algoritmo para resolver o problema, a simulação também ajuda na decisão.

Projeto no Nível RT

Método de Projeto

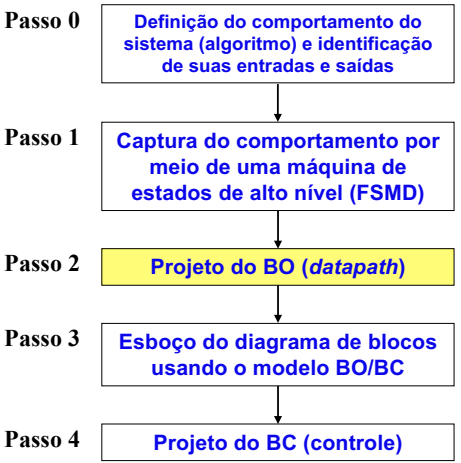


FSMD é uma extensão de uma máquina de estados, na qual:

- Entradas e saídas correspondem a dados com mais de um bit.
- Há variáveis locais para armazenar dados temporários e entradas/saídas (se necessário)
- Ações e condições podem envolver equações e expressões aritméticas (ao invés de apenas equações e expressões Booleanas).

Projeto no Nível RT

Método de Projeto

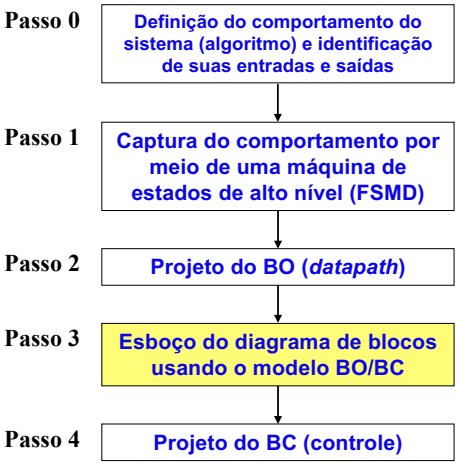


Analisando a **FSMD**, identificar:

- Os registradores para armazenar dados.
- As operações aritméticas (e lógicas) necessárias para operar os dados e para as expressões a serem usadas como condições de troca de estados.
- Selecionar os componentes do nível RT para implementar, conforme identificado no passo anterior.
- Conectar os componentes do nível RT selecionados no passo anterior.

Projeto no Nível RT

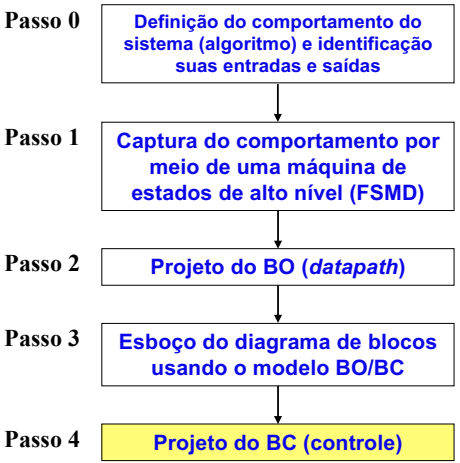
Método de Projeto



- Desenhar o diagrama de blocos segundo o modelo BO/BC.
- No desenho, identificar todas os sinais (nome e número de bits): entradas, saídas, sinais de status, sinais de comando.

Projeto no Nível RT

Método de Projeto



- A partir da FSMD e do BO projetado no passo 2, projetar a FSM que deve controlar o BO.

Projeto no Nível RT

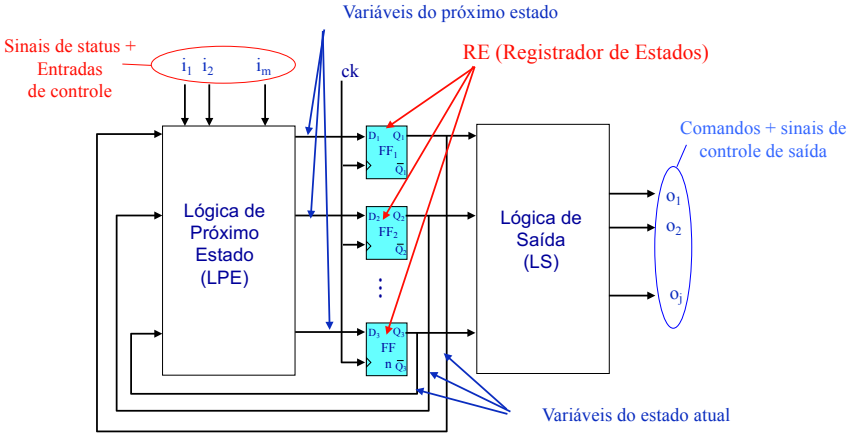
Máquinas de Estados Finitos (FSM)

- Podem ser síncronas (cadenciadas por um sinal monótono chamado relógio ou *clock*) ou assíncronas (sem relógio).
- Máquina Sequenciais Síncronas são mais utilizadas porque:
  - São mais fáceis de projetar e de validar.
  - Têm operação mais segura, são mais robustas.
- Há dois modelos: Moore e Mealy.
- Registradores podem ser vistos como Máquina Sequenciais Síncronas.

Processadores Dedicados

Modelo de Moore

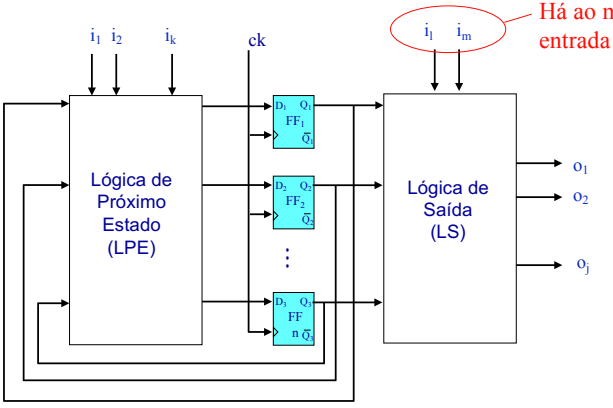
Característica principal: as saídas dependem apenas do estado atual.



Processadores Dedicados

Modelo de Mealy

Característica principal: as saídas dependem do estado atual e de entrada(s) primária(s)



Processadores Dedicados

Síntese de Circuitos Sequenciais

Roteiro para a Síntese (=Projeto)

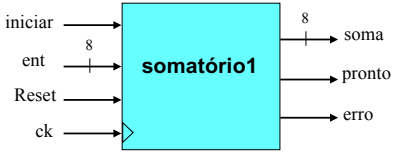
1. Determinar quantos estados são necessários (e o nº de variáveis de estado)
2. Construir o diagrama de estados, observando com cuidado o comportamento solicitado para a FSM e adotando um modelo de FSM (Moore ou Mealy):
   - Determinar as transições entre estados necessárias
   - Selecionar um estado para servir como estado inicial
3. Construir a tabela de próximo estado e a tabela das saídas
4. Escolher uma codificação para os estados e definir o tipo de flip-flops para compor o registrador de estados.
5. Sintetizar (projetar) os circuitos combinacionais: lógica de próximo estado e lógica de saída.

Processadores Dedicados

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

Exemplo 1: Enunciado

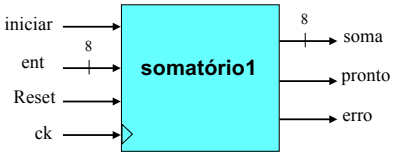
SD (bloco acelerador) para cálculo de um somatório de 4 números

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

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)



$S \leftarrow A + B + C + D$

Execução sequencial

```
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
```

Em cada linha do algoritmo (convenção somente para esta disciplina!!!)**:**

- Cada operação terá um ciclo de relógio para terminar
- Operação na mesma linha serão feitas em paralelo
- A**s atribuições ocorrerão nas bordas de relógio...**

Observe que ao término da execução, o resultado estará na variável AC (logo, AC faz o papel de S)

Processadores Dedicados

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)

Análise do Comportamento por meio de exemplo numérico

- A, B, C, D possuem 8 bits → intervalo de representação: [0, 255]
- **Suponha que A=40, B=30, C=20, D=10** (fornecidos em bordas de relógio sucessivas, nesta ordem).
- Simulação da execução:

```
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
```

Processadores Dedicados

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)

Análise do Comportamento por meio de exemplo numérico

- A, B, C, D possuem 8 bits → intervalo de representação: [0, 255]
- **Suponha que A=40, B=30, C=20, D=10** (fornecidos em bordas de relógio sucessivas, nesta ordem).
- Simulação da execução:

```
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
```

passo	AC	T
1.	0	40

Processadores Dedicados

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)

Análise do Comportamento por meio de exemplo numérico

- A, B, C, D possuem 8 bits → intervalo de representação: [0, 255]
- **Suponha que A=40, B=30, C=20, D=10** (fornecidos em bordas de relógio sucessivas, nesta ordem).
- Simulação da execução:

```
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
```

passo	AC	T
1.	0	40
2.	40	30

Processadores Dedicados

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)

Análise do Comportamento por meio de exemplo numérico

- A, B, C, D possuem 8 bits → intervalo de representação: [0, 255]
- **Suponha que A=40, B=30, C=20, D=10** (fornecidos em bordas de relógio sucessivas, nesta ordem).
- Simulação da execução:

```
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
```

passo	AC	T
1.	0	40
2.	40	30
3.	70	20

Observe que a partir do passo 3, a adição ”AC + T” pode resultar em *overflow* (inclusive o enunciado ressalta a importância de detectar a ocorrência de *overflow*, o que se jusrtifica pelo fato de a saída ”soma” possui somente 8 bits…)

Processadores Dedicados

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)

Análise do Comportamento por meio de exemplo numérico

- A, B, C, D possuem 8 bits → intervalo de representação: [0, 255]
- **Suponha que A=40, B=30, C=20, D=10** (fornecidos em bordas de relógio sucessivas, nesta ordem).
- Simulação da execução:

```
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
```

passo	AC	T
1.	0	40
2.	40	30
3.	70	20
4.	90	10

Processadores Dedicados

Exemplo 1: Passo 0 (Definição do comportamento e identificação entradas e saídas)

Análise do Comportamento por meio de exemplo numérico

- A, B, C, D possuem 8 bits → intervalo de representação: [0, 255]
- **Suponha que A=40, B=30, C=20, D=10** (fornecidos em bordas de relógio sucessivas, nesta ordem).
- Simulação da execução:

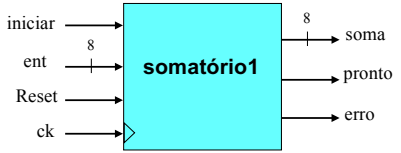
```
1. AC ← 0;  T← ent;          //  A está estável em ent
2. AC ← AC + T;   T← ent; //  B está estável em ent
3. AC ← AC + T;   T← ent; //  C está estável em ent
4. AC ← AC + T;   T← ent; //  D está estável em ent
5. AC ← AC + T;              // O resultado final S estará em AC
```

passo	AC	T
1.	0	40
2.	40	30
3.	70	20
4.	90	10
5.	**100**	**10**

No passo 5 não faz sentido atualizar a variável "T" (e portanto, ela pode permanecer com o valor anterior…)

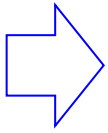
Processadores Dedicados

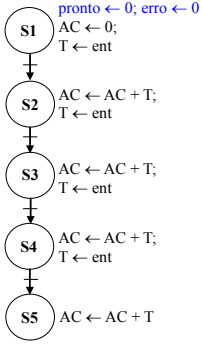
Exemplo 1: Passo 1 (Captura do comportamento por meio de uma FSMD)



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

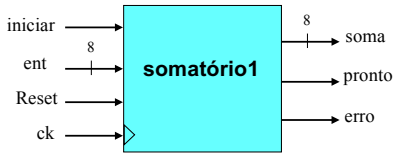
Criando um estado para cada passo do algoritmo





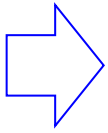
Processadores Dedicados

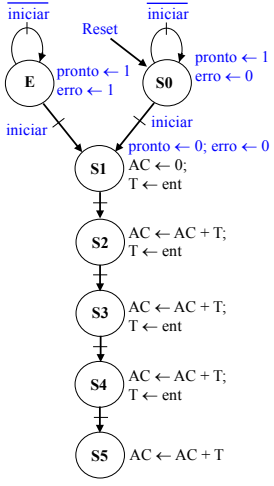
Exemplo 1: Passo 1 (Captura do comportamento por meio de uma FSMD)



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

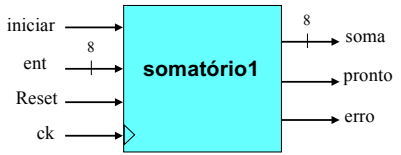
Adicionando os estados iniciais





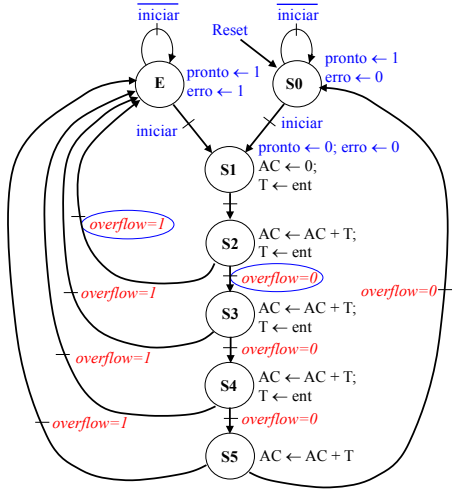
Processadores Dedicados

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



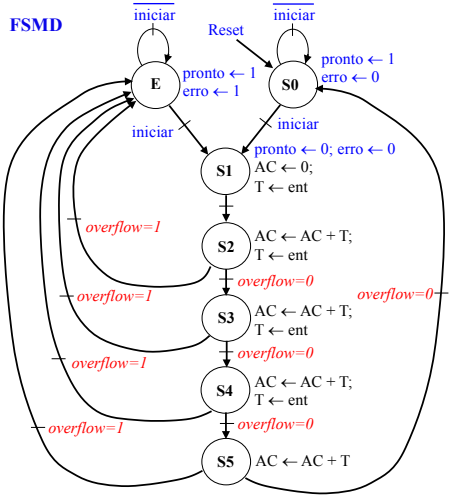
1. AC ← 0; T← ent; // A está estável em ent
2. AC ← AC + T; T← ent; // B está estável em ent
3. AC ← AC + T; T← ent; // C está estável em ent
4. AC ← AC + T; T← ent; // D está estável em ent
5. AC ← AC + T; // O resultado final S estará em AC

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

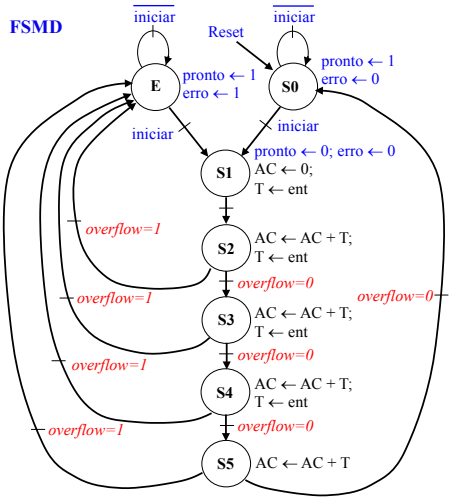
Exemplo 1: Passo 2 (Projeto do BO)



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

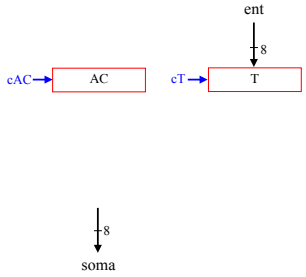
Processadores Dedicados

Exemplo 1: Passo 2 (Projeto do BO)



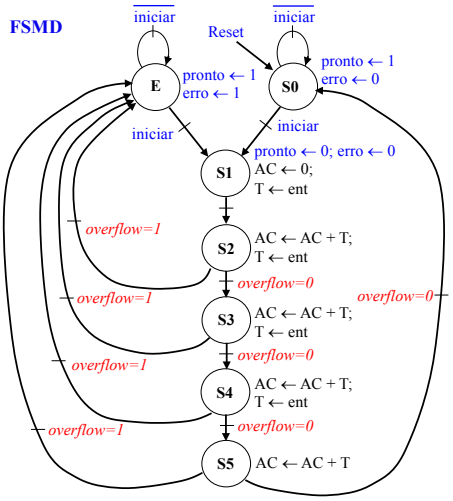
2ª questão para guiar o projeto do BO:

- Quais variáveis são usadas para armazenar dados?
- "AC" e "T"
- Logo, o BO precisa ter dois registradores. Chamemo-los de "AC" e "T".



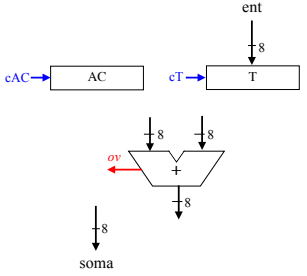
Processadores Dedicados

Exemplo 1: Passo 2 (Projeto do BO)



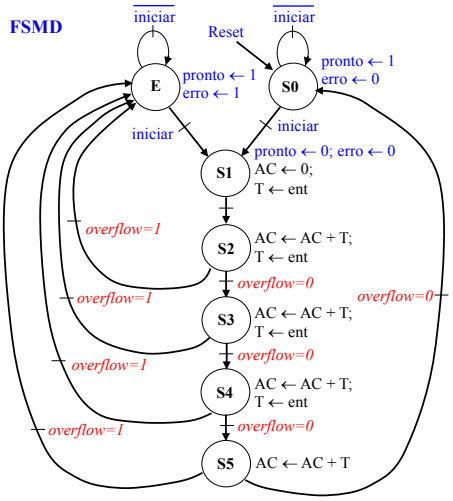
3ª questão para guiar o projeto do BO:
Quais operações são realizadas?

- Uma adição para números de 8 bits e o respectivo teste de *overflow*
- Logo, precisaremos de um somador para operandos de 8 bits com detector de *overflow*.



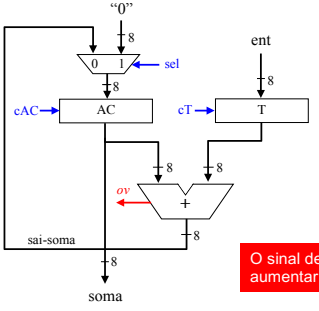
Processadores Dedicados

Exemplo 1: Passo 2 (Projeto do BO)



4ª questão para guiar o projeto do BO:

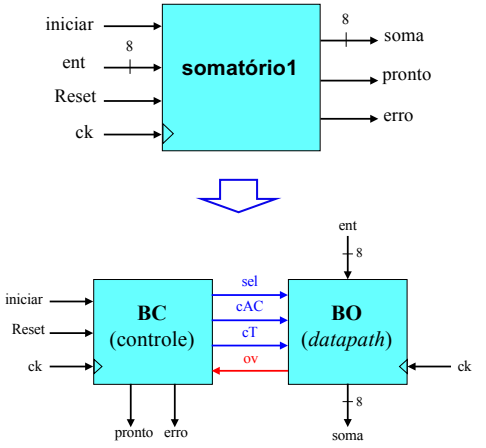
- Quais operações são realizadas sobre quais dados (incluindo-se as condições)?
- T← ent, AC← 0; AC← AC+T
- Logo, deve haver um mux2:1 na entrada de AC e conexões entre AC e +, T e + e + e AC.



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

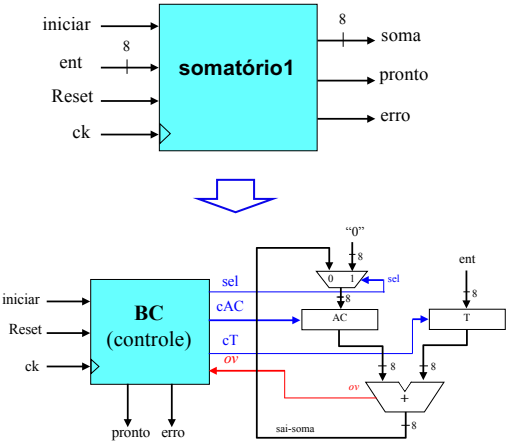
Processadores Dedicados

Exemplo 1: Passo 3 (Esboçando o Diagrama BO/BC)



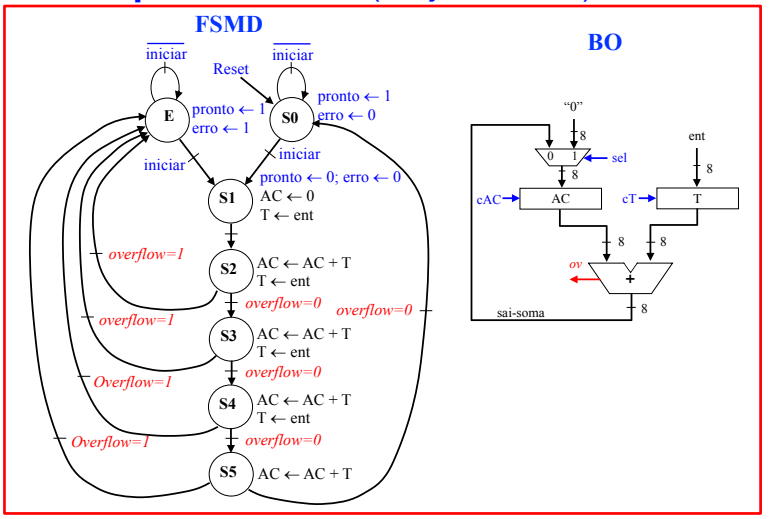
Processadores Dedicados

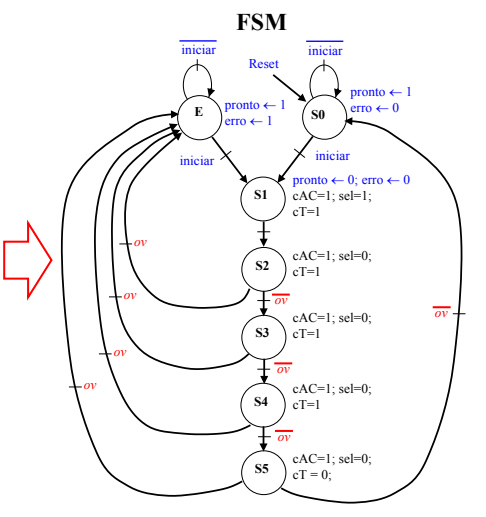
Exemplo 1: Passo 3 (Esboçando o Diagrama BO/BC)



Processadores Dedicados

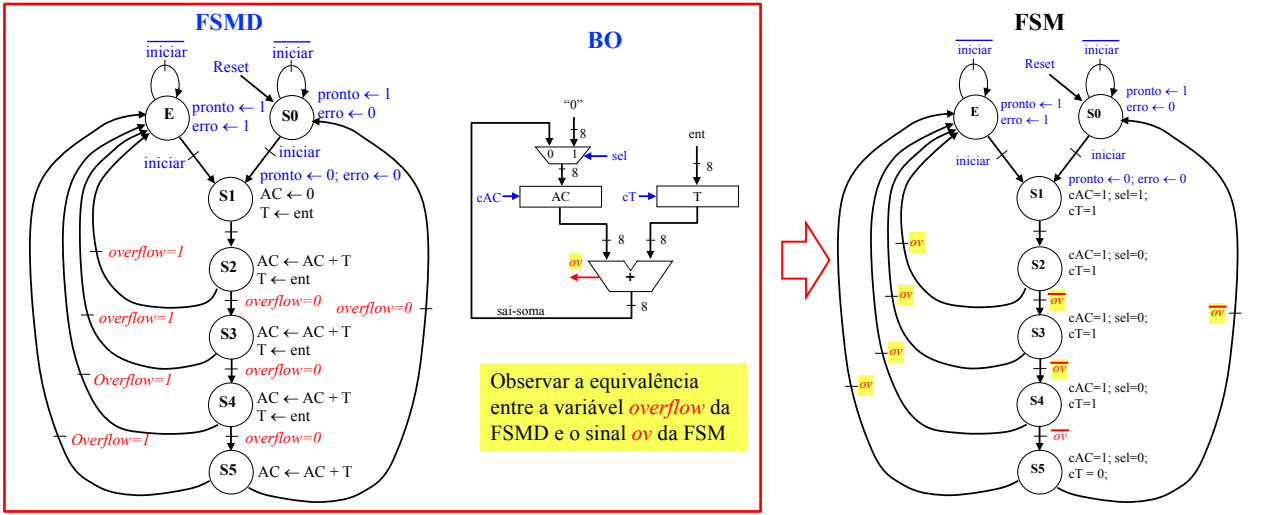
Exemplo 1: Passo 4 (Projeto do BC): Criando uma FSM a partir da FSMD e do BO





Processadores Dedicados

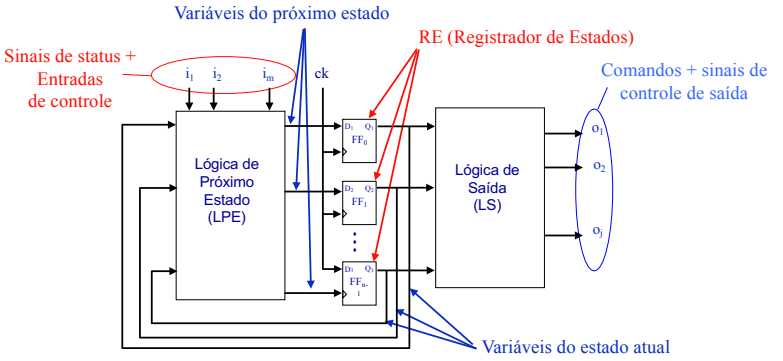
Exemplo 1: Passo 4 (Projeto do BC): Criando uma FSM a partir da FSMD e do BO



Processadores Dedicados

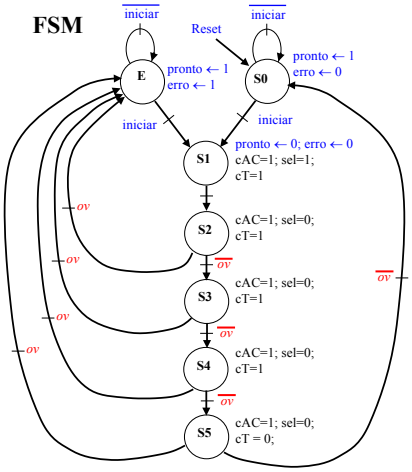
Exemplo 1: Passo 4 (Projeto do BC): Definindo o número de flip-flops

Relembrando o modelo genérico de máquina de estados (FSM) do tipo “Moore”



Processadores Dedicados

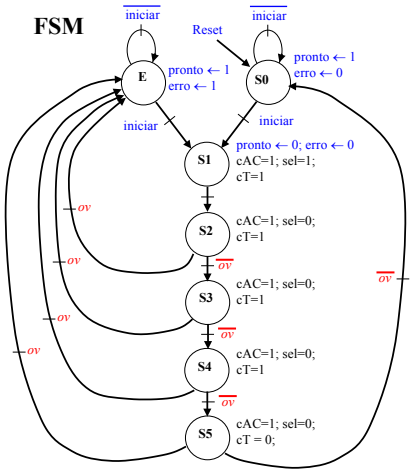
Exemplo 1: Passo 4 (Projeto do BC): Definindo o número de flip-flops



Quantos flip-flops são necessários para implementar a FSM?

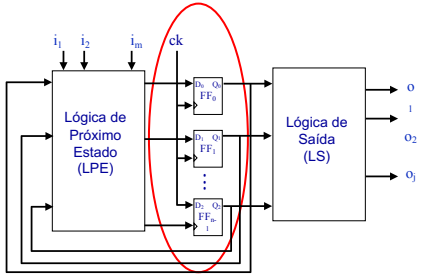
Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): Definindo o número de flip-flops



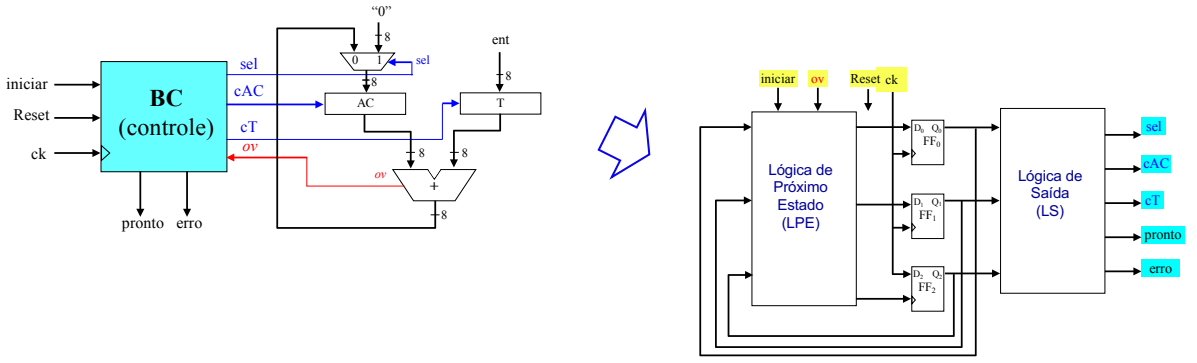
Quantos flip-flops são necessários para implementar a FSM?

Resp.: como são 7 estados (=7 combinações), são necessário, no mínimo, é $\log_2 7 = 3$ flip-flops



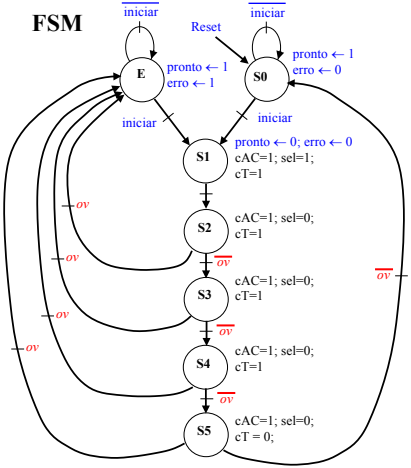
Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): Mapeando as interfaces do BC para o modelo de FSM de Moore



Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): projetando a Lógica de Próximo Estado (LPE)



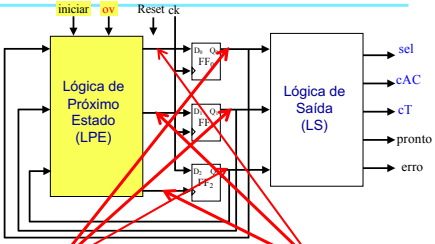
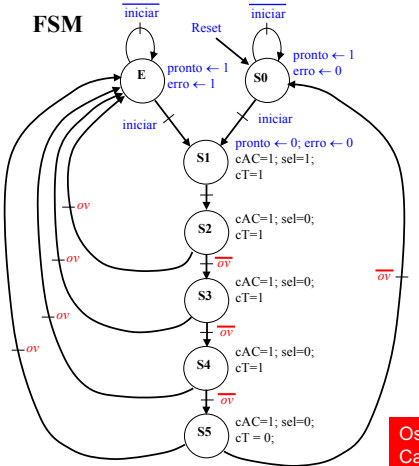


Tabela de Transição de Estados (LPE)

Estado atual	iniciar	*ov*	Próximo estado
E	0	X	**E**
E	1	X	S1
S0	0	X	S0
S0	1	X	S1
S1	X	X	S2
S2	X	0	S3
S2	X	1	**E**
S3	X	0	S4
S3	X	1	**E**
S4	X	0	S5
S4	X	1	**E**
S5	X	0	S0
S5	X	1	**E**

Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): projetando a Lógica de Saída (LS)



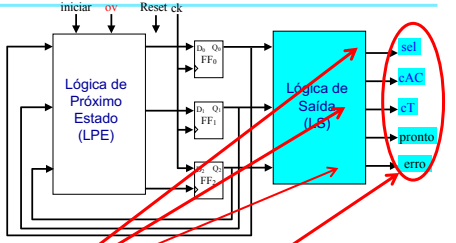


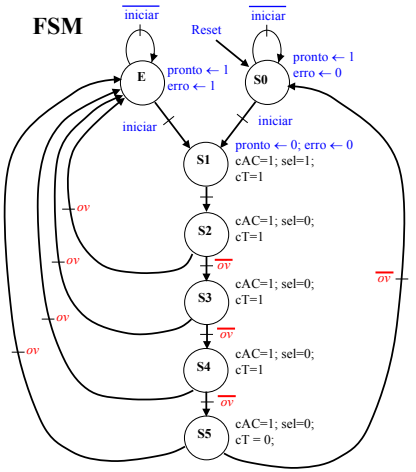
Tabela-Verdade da LS

Estado atual	Sinais de comando			Saídas de controle	
	sel	cAC	cT	pronto	erro
E	X	0	0	1	1
S0	X	0	0	1	0
S1	1	1	1	0	0
S2	0	1	1	0	0
S3	0	1	1	0	0
S4	0	1	1	0	0
S5	0	1	0	0	0

Os sinais cAC e cT só valem "1" nos estados em que houver carga nestes registradores. Caso contrário, eles devem valer "0" (nesta disciplina, eles jamais valem *don't care*)

Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): Assinalamento de Estados



Corresponde a:

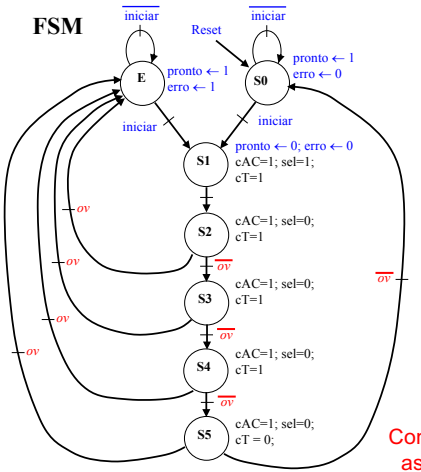
Escolher uma codificação binária para representar cada estado... Se for para minimizar o número de bits, usar $\log_2 N$, onde N é o número de estados. Exemplo de assinalamento:

Estado	Código binário dos estados		
	Q2	Q1	Q0
E	**1**	**1**	**1**
S0	**0**	**0**	**0**
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0
S5	1	0	1

Estado inicial: este é o estado para o qual a FSM vai caso o Reset seja acionado!!

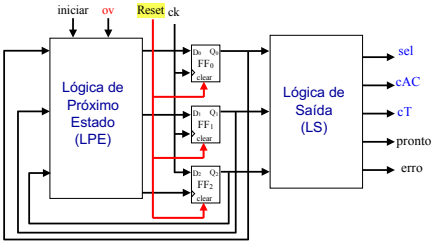
Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): Assinalamento de Estados



Estado	Código binário dos estados		
	Q2	Q1	Q0
E	**1**	**1**	**1**
S0	**0**	**0**	**0**
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0
S5	1	0	1

Estado inicial: este é o estado para o qual a FSM vai caso o Reset seja acionado!!



Consequência: a função do sinal de controle Reset corresponde a realizar um reset assíncrono no Registrador de Estados (ou seja, os flip-flops deste registrado deverão ter entrada de reset assíncrono!)

Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): projetando a Lógica de Próximo Estado (LPE)

Estado	Código binário dos estados		
	Q2	Q1	Q0
E	**1**	**1**	**1**
S0	**0**	**0**	**0**
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0
S5	1	0	1

Tabela de Transição de Estados (LPE)

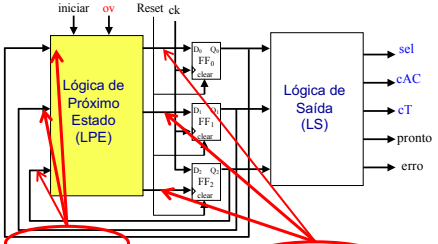
Estado atual	iniciar	*ov*	Próximo estado
E	0	X	**E**
E	1	X	**S1**
S0	0	X	**S0**
S0	1	X	**S1**
S1	X	X	**S2**
S2	X	0	**S3**
S2	X	1	**E**
S3	X	0	**S4**
S3	X	1	**E**
S4	X	0	**S5**
S4	X	1	**E**
S5	X	0	**S0**
S5	X	1	**E**

Substituindo os nomes dos estados pelos respectivos códigos binários...

Estado atual	Q2	Q1	Q0	iniciar	*ov*	$Q2^+$	$Q1^+$	$Q0^+$	Próximo estado
E	**1**	**1**	**1**	0	X	**1**	**1**	**1**	**E**
E	**1**	**1**	**1**	1	X	0	0	1	S1
S0	0	0	0	0	X	0	0	0	S0
S0	0	0	0	1	X	0	0	1	S1
S1	0	0	1	X	X	0	1	0	S2
S2	0	1	0	X	0	0	1	1	S3
S2	0	1	0	X	1	**1**	**1**	**1**	**E**
S3	0	1	1	X	0	1	0	0	S4
S3	0	1	1	X	1	**1**	**1**	**1**	**E**
S4	1	0	0	X	0	1	0	1	S5
S4	1	0	0	X	1	**1**	**1**	**1**	**E**
S5	1	0	1	X	0	0	0	0	S0
S5	1	0	1	X	1	**1**	**1**	**1**	**E**

Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): projetando a 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**

Tarefa 1: encontrar as equações minimizadas para:

$Q2^+ =$

$Q1^+ =$

$Q0^+ =$

Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): projetando a Lógica de Saída (LS)

Estado	Código binário dos estados		
	Q2	Q1	Q0
E	**1**	**1**	**1**
S0	**0**	**0**	**0**
S1	0	0	1
S2	0	1	0
S3	0	1	1
S4	1	0	0
S5	1	0	1

Substituindo os nomes dos estados pelos respectivos códigos binários...

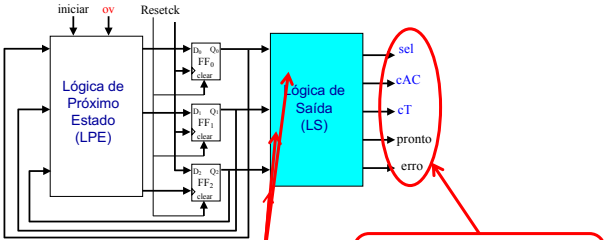
Tabela-Verdade da LS

Estado atual	Sinais de comando			Saídas de controle	
	sel	cAC	cT	pronto	erro
E	X	0	0	1	1
S0	X	0	0	1	0
S1	1	1	1	0	0
S2	0	1	1	0	0
S3	0	1	1	0	0
S4	0	1	1	0	0
S5	0	1	0	0	0

Estado atual	Q2	Q1	Q0	Sinais de comando			Saídas de controle	
				sel	cAC	cT	pronto	erro
E	1	1	1	X	0	0	1	1
S0	0	0	0	X	0	0	1	0
S1	0	0	1	1	1	1	0	0
S2	0	1	0	0	1	1	0	0
S3	0	1	1	0	1	1	0	0
S4	1	0	0	0	1	1	0	0
S5	1	0	1	0	1	0	0	0

Processadores Dedicados

Exemplo 1: Passo 4 (Projeto do BC): projetando a 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

Tarefa 2: encontrar as equações minimizadas para:

sel =

cAC =

cT =

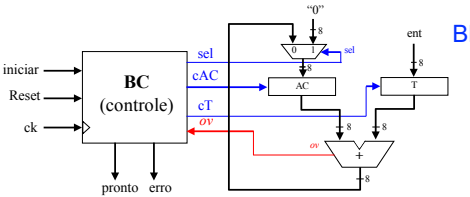
pronto =

erro =

Processadores Dedicados

Exemplo 1: Finalizando

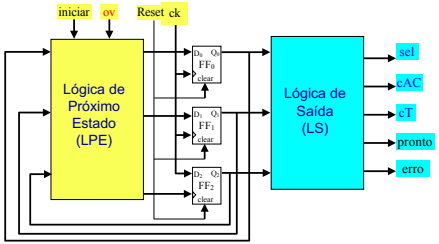
Sistema Digital
somatório1



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**

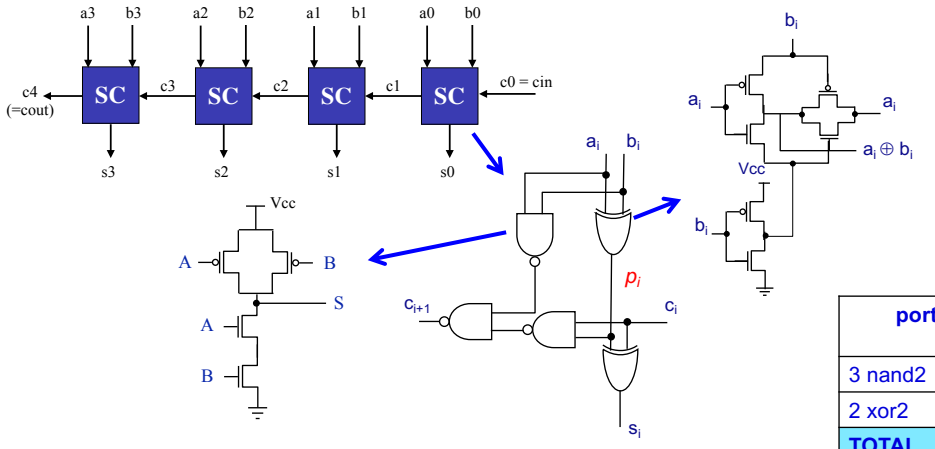


Lógica de Saída (LS)

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

Estimativa de Custo

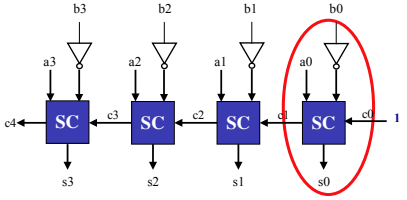
O Somador Paralelo *Carry-Ripple*: número de transistores por bit



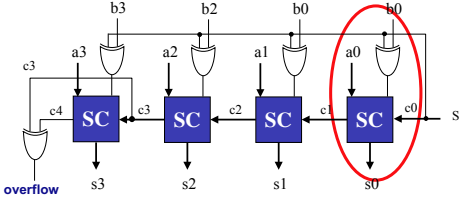
portas	Nº trans.
3 nand2	3x4
2 xor2	2x6
TOTAL	**24**

Estimativa de Custo

Subtrator e Somador/Subtrator: número de transistores por bit



portas	Nº trans.
3 nand2	3x4
2 xor2	2x6
1 inversor	1x2
TOTAL	**26**



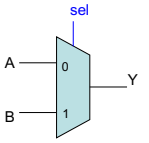
portas	Nº trans.
3 nand2	3x4
3 xor2	3x6
TOTAL	**30**

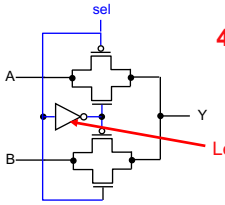
Estimativa de Custo

Mux 2:1: número de transistores por bit

Um bit (nível lógico)

$Y = \overline{sel} \cdot A + sel \cdot B$

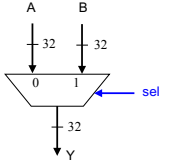


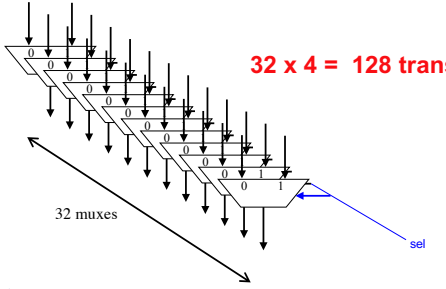


4 transistores

Lembrando que o custo do inversor é desprezado

Múltiplos bits (nível RT)

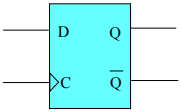




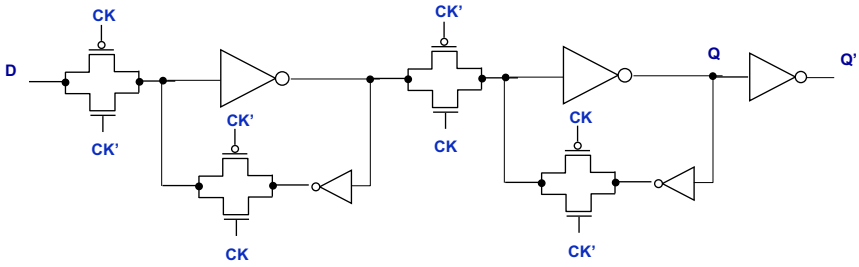
32 x 4 = 128 transistores

Estimativa de Custo

Flip-flop D mestre-escravo CMOS: número de transistores por bit



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

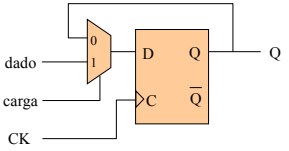


OBS: para set ou reset assíncrono, adicionar 4 transistores

Estimativa de Custo

Flip-flop D mestre-escravo CMOS controlado*:

número de transistores por bit



18+4= 22 transistores

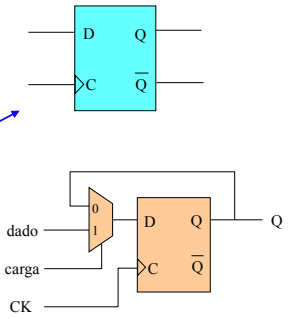
Para set ou reset assíncrono, adicionar 4 transistores

* habilitação de carga (”load/reg enable”) ou deslocamento para um lado

Estimativa de Custo

Número de Transistores: Resumo

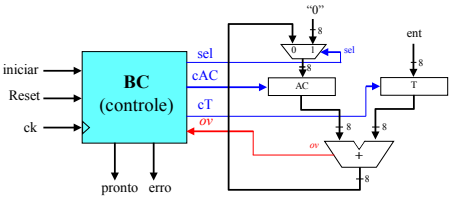
Componente RT	Custo
Somador	24n
Subtrator	26n
Somador/subtrator	30n
Mux 2:1	4n
Registrador com carga paralela (+4 transistores para set ou reset assíncrono)	18n
Registrador com carga paralela controlada (+4 transistores para set ou reset assíncrono)	22n

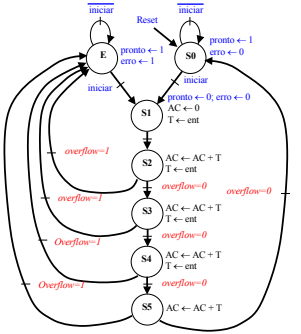


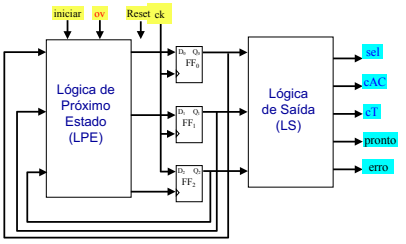
Obs: sempre que for solicitada uma estimativa de custo de sistema digital, o número de transistores por bit será fornecido.

Processadores Dedicados

Exemplo 1: Estimativa de Custo







Estimativa de custo para o BO:

Componente RT	nº de transistores
1 Somador	24n
1 Mux 2:1	4n
2 Reg. com carga paralela controlada	2 x 22n
Total de transistores	72 n = 72 . 8 = **576**

Estimativa de custo para o BC:

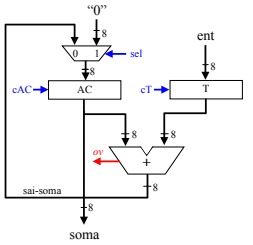
- Número de estados da FSMD/FSM: **7**
- Número de saídas distintas da LS*= **5**

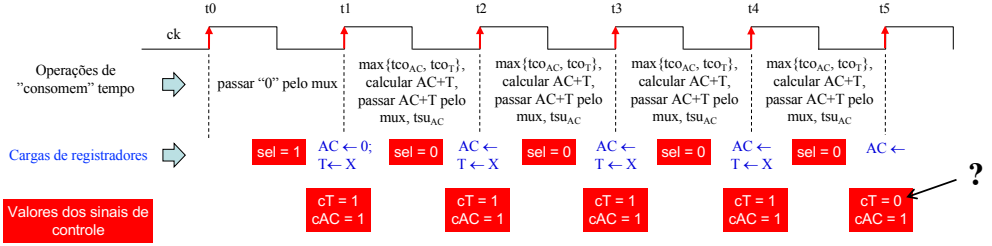
* = sinais de comando + sinais de controle de saída, diferentes entre si

Processadores Dedicados

Exemplo 1: Timing e Sinais de Controle

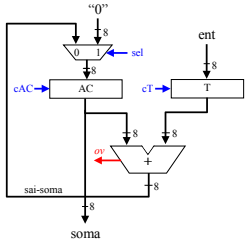
1. AC ← 0; T← ent; // A está estável em ent
2. AC ← AC + T; T← ent; // B está estável em ent
3. AC ← AC + T; T← ent; // C está estável em ent
4. AC ← AC + T; T← ent; // D está estável em ent
5. AC ← AC + T; // O resultado final S estará em AC





Processadores Dedicados

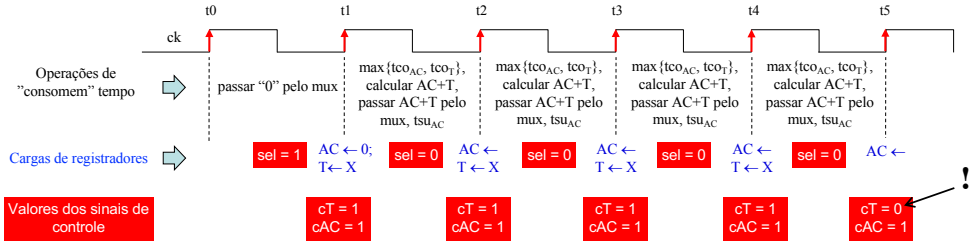
Exemplo 1: Timing e Sinais de Controle



MUITO IMPORTANTE (convenção para esta disciplina):

- **Valores possíveis para o sinal de habilitação de carga de um registrador**

1	Nos ciclos de relógio em que o registrador for carregado. Exemplo: R1← R1+ R2 significa que R1 deve ser carregado
0	Nos ciclos de relógio em que o registrador não for ser carregado.
X (don't care)	**Nesta disciplina, jamais!**



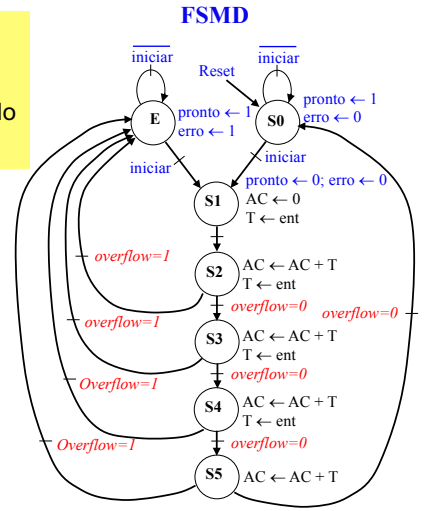
Processadores Dedicados

Exemplo 1: 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



Processadores Dedicados

Exemplo 1: 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

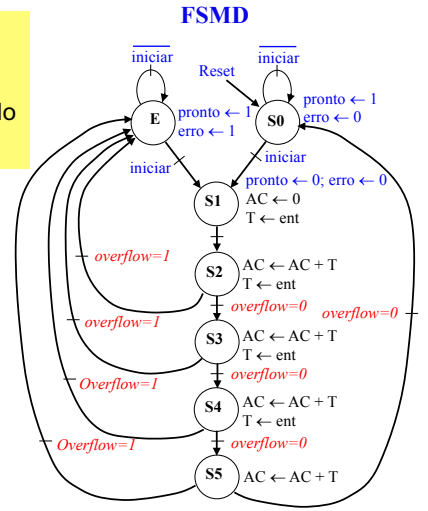
Cálculo do nº de ciclos:

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

S1, S2, S3, S4, S5

Ou seja, 5 ciclos de relógio.

- Os estados ”E” e ”S0” são desconsiderados, pois eles não realizam cálculos



Processadores Dedicados

Exemplo 1: Estimativa de Desempenho

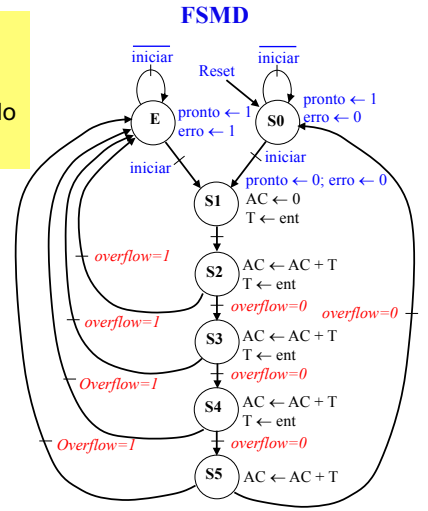
Tempo de Execução:

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

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

Estimativa de T:

- Análise de Timing, a seguir ...

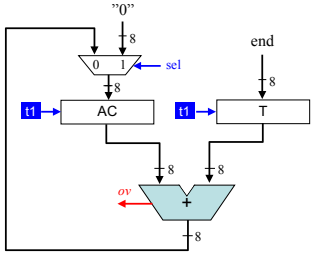


Análise de Timing: Tempos de Estabilização dos Sinais

Suponha as seguintes características temporais dos componentes

Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinal de controle (sel)	atraso	tdcontrol	0 ns

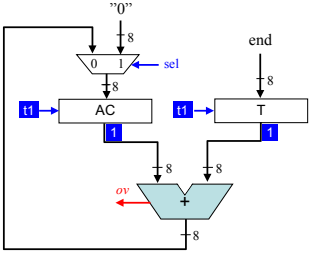
Vamos assumir t1, pois o ciclo de relógio t1-t2 é mais elucidativo



Análise de Timing: Tempos de Estabilização dos Sinais

Suponha as seguintes características temporais dos componentes

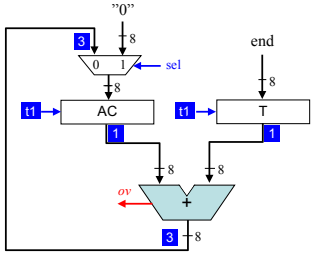
Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinal de controle (sel)	atraso	tdcontrol	0 ns



Análise de Timing: Tempos de Estabilização dos Sinais

Suponha as seguintes características temporais dos componentes

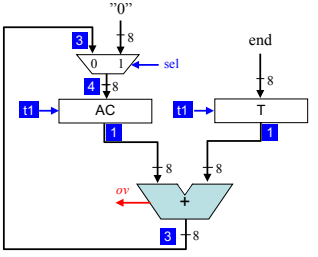
Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinal de controle (sel)	atraso	tdcontrol	0 ns



Análise de Timing: Tempos de Estabilização dos Sinais

Suponha as seguintes características temporais dos componentes

Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinal de controle (sel)	atraso	tdcontrol	0 ns



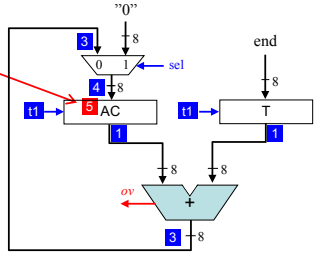
Processadores Dedicados

Análise de Timing: Tempos de Estabilização dos Sinais

Suponha as seguintes características temporais dos componentes

Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinal de controle (sel)	atraso	tdcontrol	0 ns

Considerando o tempo de setup do registrador AC



Logo, atraso crítico = D = 5 ns

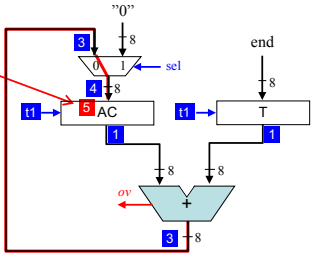
Processadores Dedicados

Análise de Timing: Caminho Crítico e Atraso Crítico

Suponha as seguintes características temporais dos componentes

Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinal de controle (sel)	atraso	tdcontrol	0 ns

Considerando o tempo de setup do registrador AC

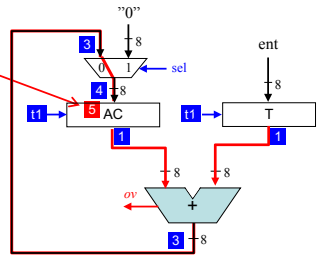


Logo, atraso crítico = D = 5 ns

Processadores Dedicados

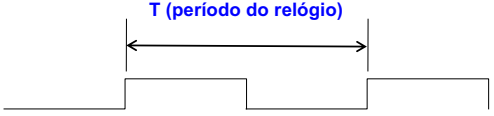
Análise de Timing: Estimativa do Período (Mínimo) do Relógio

Considerando o tempo de setup do registrador R1



Logo, atraso crítico = D = 5 ns

E período mínimo T = D = 5ns



Processadores Dedicados

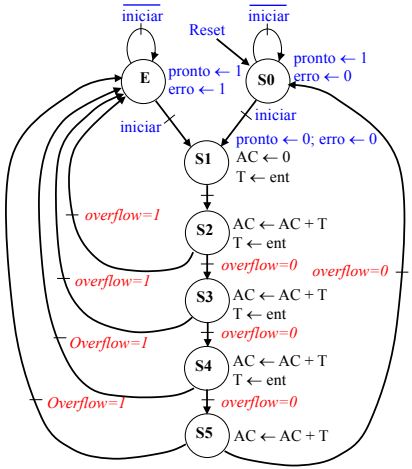
Exemplo 1: Estimativa de Desempenho

Tempo de Execução:

T_{exec} = nº de ciclos x T

= 5 ciclos x 5ns

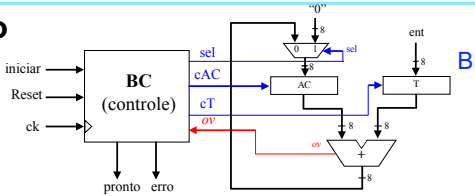
= 25 ns



Processadores Dedicados

Exemplo 1: Resumindo

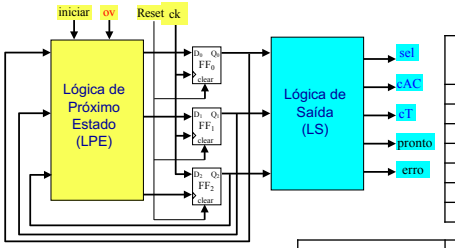
Sistema Digital somatório1



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	1	0	0

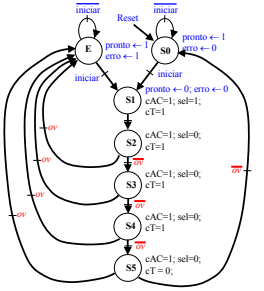
T_{exec} = nº de ciclos x T = 5 ciclos x 5ns = 25 ns

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

Processadores Dedicados

Descrição VHDL de um Sistema Digital no Nível RT

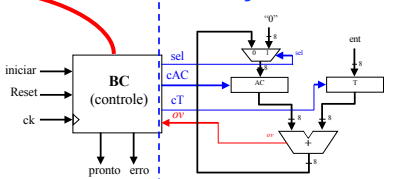
FSM:
descrição comportamental




```
ENTITY bc IS
PORT (Reset, clk, c, menor : IN STD_LOGIC;
sel, cAC, cT, pronto, erro: OUT STD_LOGIC);
END bc;


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

Bloco Operativo (BO):
descrição estrutural




```
ENTITY bo IS
PORT (clk, Ctotal, Rtotal : IN STD_LOGIC;
ent: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
ov: OUT STD_LOGIC;
soma: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
END bo;

ARCHITECTURE estrutura OF bo IS

COMPONENT somador IS
PORT (a, b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
s : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END COMPONENT;

COMPONENT registrador IS
PORT (clk, carga, reset : IN STD_LOGIC;
d : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END COMPONENT;

.....
```