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

Sistemas Digitais

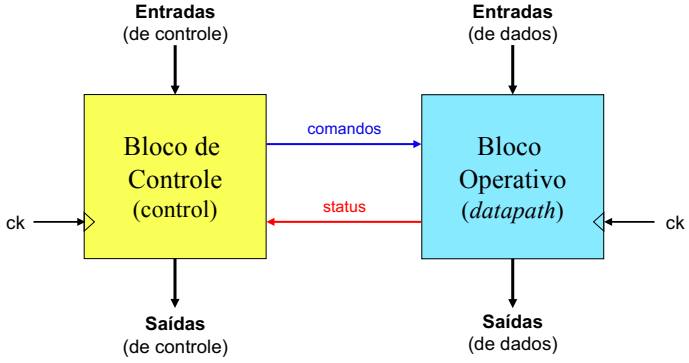
INE 5406

Aula 3T - parte 2

2. Processadores Dedicados (Blocos Aceleradores).
Processadores Dedicados: somadores sequenciais (Exemplo 1).
Estimativa de custo e de desempenho.

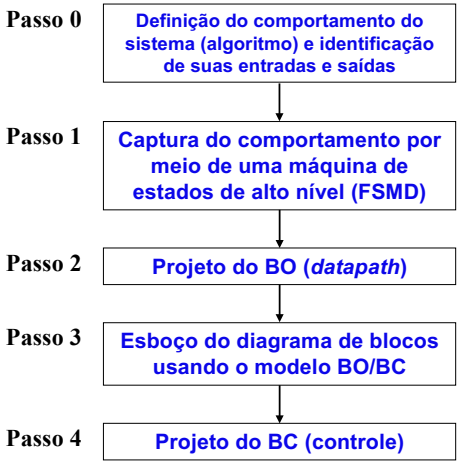
Professores:
Prof. José Luís Güntzel, Rafael Cancian e Ismael Seidel
{j.guntzel, rafael.cancian, ismael.seidel}@ufsc.br

O Modelo Bloco Operativo/Bloco de Controle* (BO/BC)



* Em inglês, *Datapath/Contro*l

Método para o Projeto RTL

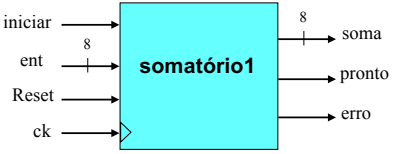


Exemplo 1: Enunciado

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").

Passo 0 (Identificação das entradas e saídas a partir do enunciado)



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

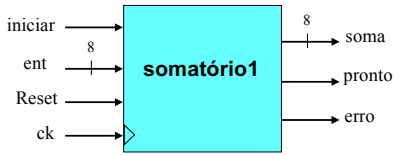
Exemplo 1: Enunciado

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**.
- Os valores dos operandos **A**, **B**, **C** e **D** vão sendo entregues a “somatório1” por meio da entrada ”ent" em bordas de relógio consecutivas, ao mesmo tempo em que o cálculo vai sendo realizado;
- 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**.

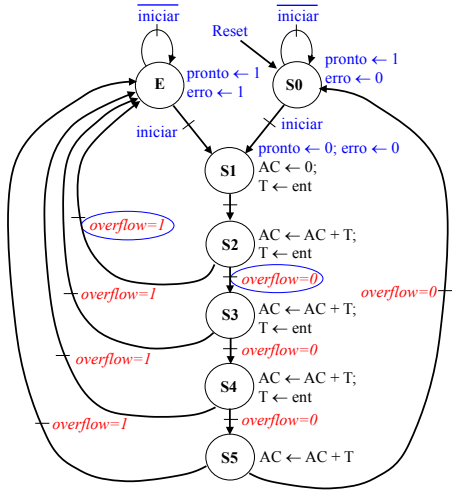
Processadores Dedicados

Exemplo 1: Passo 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

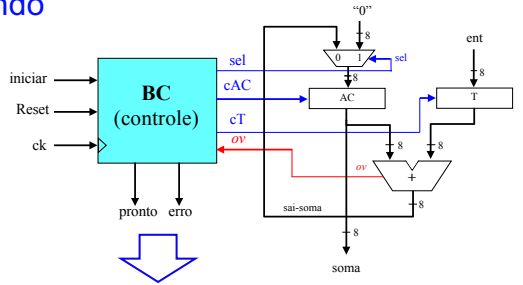
O 1º teste de overflow seria dispensável, uma vez que na primeira soma um dos operandos é zero. Porém, **iremos mantê-lo para permitir uma futura generalização do algoritmo.**



Processadores Dedicados

Exemplo 1: Finalizando

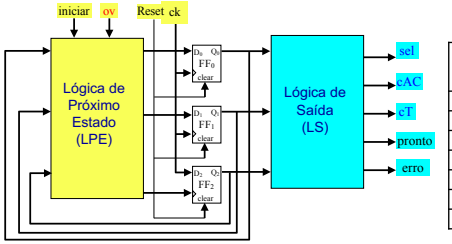
Sistema Digital somatório1



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**

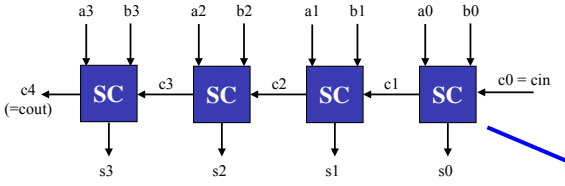


Lógica de Saída (LS)

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

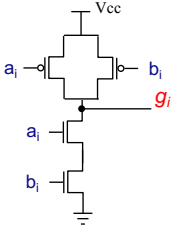
Estimativa de Custo

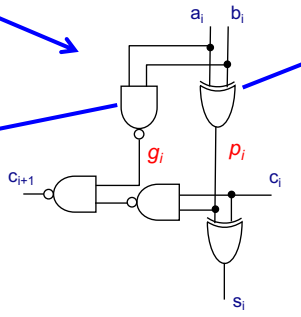
O Somador Paralelo *Carry-Ripple:* número de transistores

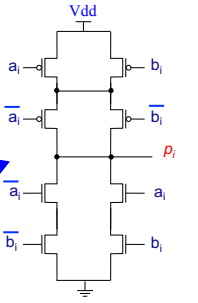


Nº de transistores por SC:

Portas	Nº de Transistores
3 nand2	3 x 4 = 12
2 xor	2 x 12 = 24
Total	**36**

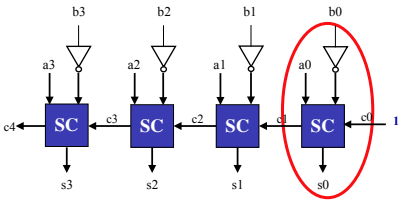






Estimativa de Custo

Subtrator e Somador/Subtrator: número de transistores



Portas	Nº de Transistores
3 nand2	3 x 4 = 12
2 xor	2 x 12 = 24
1 inv	1 x 2 = 2
Total	**38**

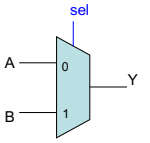
Portas	Nº de Transistores
3 nand2	3 x 4 = 12
3 xor	3 x 12 = 36
Total	**48**

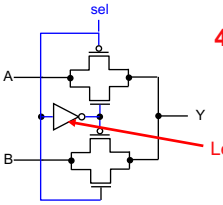
Estimativa de Custo

Mux 2:1: número de transistores

Um bit (nível lógico)

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

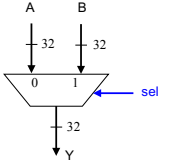


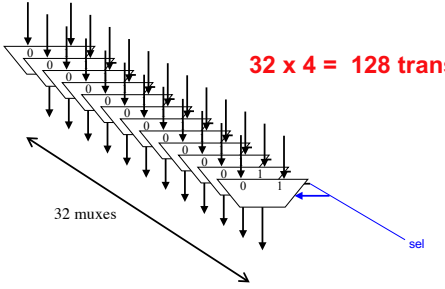


4 transistores

Lembrando que o custo do inversor é desprezado

Múltiplos bits (nível RT)

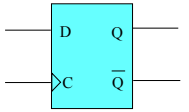




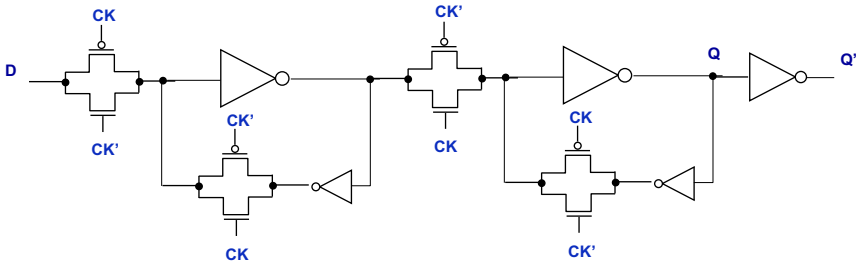
32 x 4 = 128 transistores

Estimativa de Custo

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



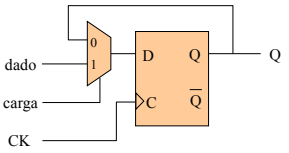
18 ou 20 transistores
(eventualmente, podemos considerar somente um inversor para o clock de todos os bits)



OBS: para set ou reset assíncrono, adicionar 4 transistores

Estimativa de Custo

Flip-flop D mestre-escravo CMOS com sinal de carga (enable)*: número de transistores



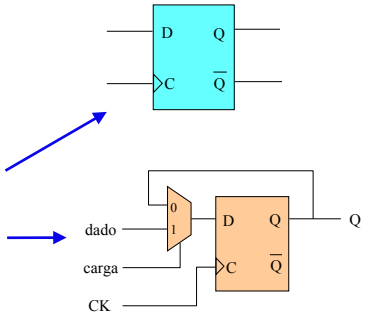
18+4= 22 transistores
Para set ou reset assíncrono, adicionar 4 transistores

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

Estimativa de Custo

Número de Transistores: Resumo

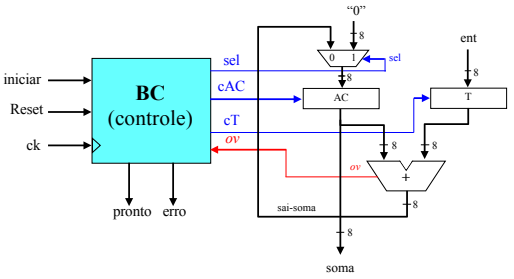
Componente RT	Custo
Somador	36n
Subtrator	38n
Somador/subtrator	48n
Mux 2:1	4n
Registrador com carga paralela (+4 transistores para set ou reset assíncrono)	18n
Registrador com carga paralela controlada (+4 transistores para set ou reset assíncrono)	22n

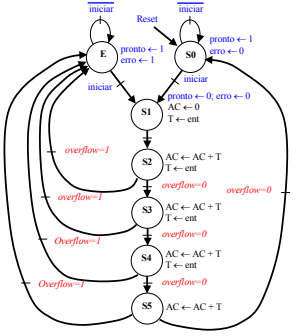


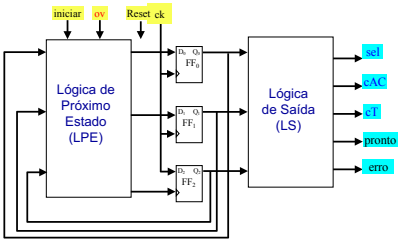
Obs: sempre que for solicitada uma estimativa de custo de sistema digital, o número de transistores por bit será fornecido.

Processadores Dedicados

Exemplo 1: Estimativa de Custo







Estimativa de custo para o BO:

Componente RT	nº de transistores
1 Somador	36n
1 Mux 2:1	4n
2 Reg. com carga paralela controlada	2 x 22n
Total de transistores	84 n = 84 x 8 = **672**

Estimativa de custo para o BC:

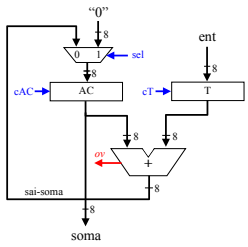
- Número de estados da FSMD/FSM: **7**
- Número de saídas distintas da LS*= **5**

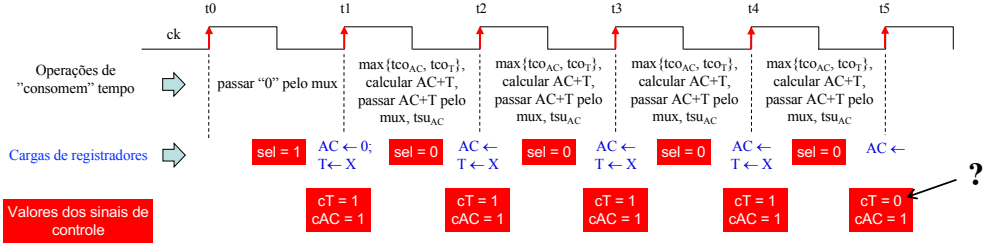
* = sinais de comando + sinais de controle de saída, diferentes entre si

Processadores Dedicados

Exemplo 1: Timing e Sinais de Controle

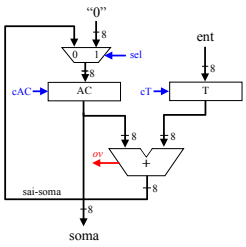
1. AC ← 0; T← ent; // A está estável em ent
2. AC ← AC + T; T← ent; // B está estável em ent
3. AC ← AC + T; T← ent; // C está estável em ent
4. AC ← AC + T; T← ent; // D está estável em ent
5. AC ← AC + T; // O resultado final S estará em AC





Processadores Dedicados

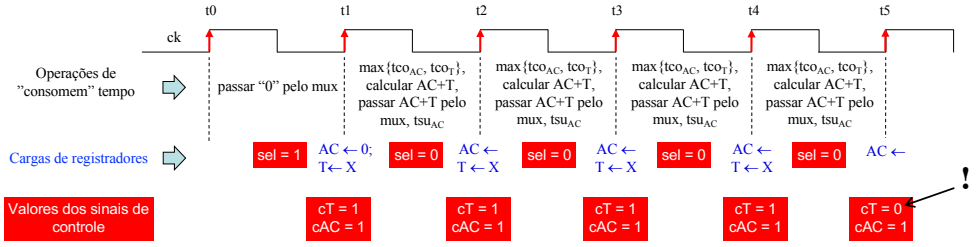
Exemplo 1: Timing e Sinais de Controle



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!**



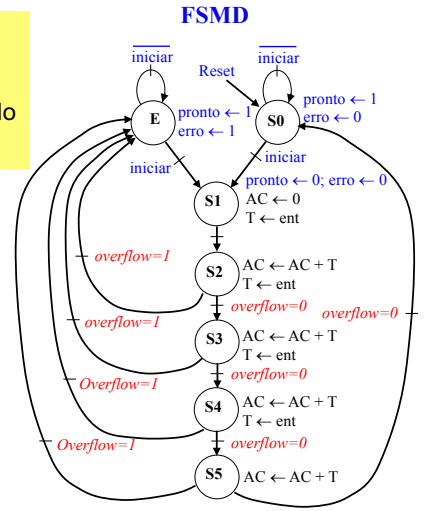
Processadores Dedicados

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

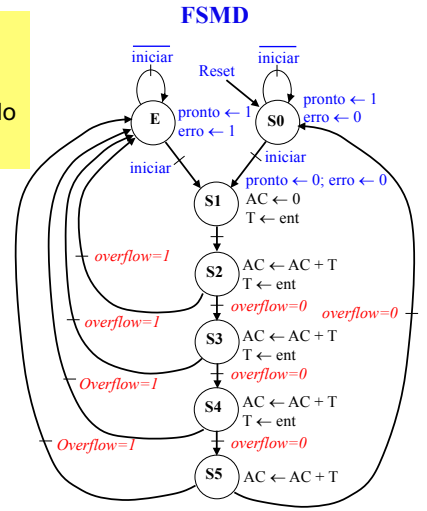
Cálculo do nº de ciclos:

- 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

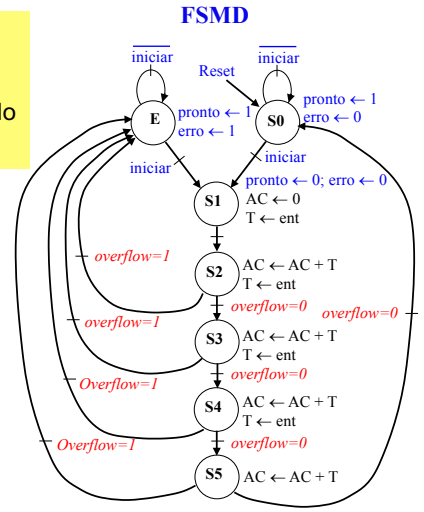
Tempo de Execução:

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

- **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:

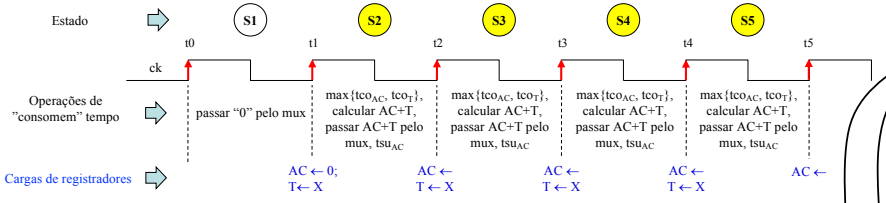
- Análise de Timing (*Static Timing Analysis* - STA) , a seguir ...



Processadores Dedicados

Exemplo 1: Estimativa de Desempenho

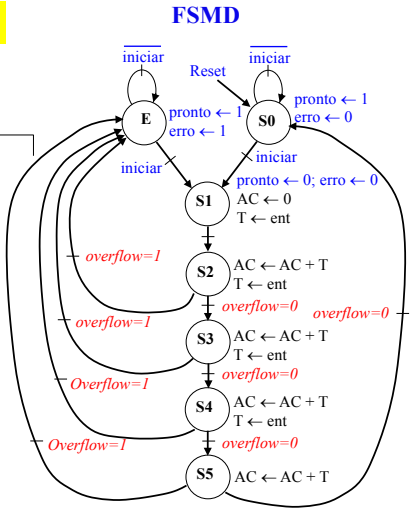
ATENÇÃO: nesta disciplina, o período do relógio (T) é invariável!



Para estimar o período do relógio (T):

1. **Identificar os estados que potencialmente demoram mais para serem executados (analisar a FSMD)**
2. **Realizar a análise de timing de cada um destes estados**

2.1 Em geral, assume-se que os sinais de controle têm atraso=0 e assim, pode-se realizar a análise de timing somente do BO.

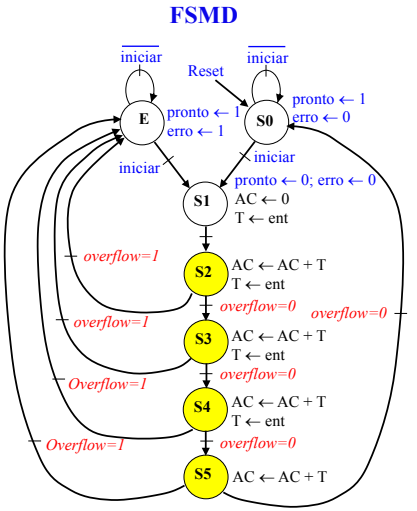


Exemplo 1: Estimativa de Desempenho

Para estimar o período do relógio (T):

1. **Identificar os estados que potencialmente demoram mais para serem executados (analisar a FSMD)**

- A operação nível RT **AC ← AC + T,** faz uso de um somador (no caso, de 8 bits) e por isso, é a mais lenta de todas as realizadas no Exemplo 1
- As operações nível RT que só escrevem em registradores, tais como **AC ← 0** e **T ← ent**, são mais rápidas, pois seu atraso corresponde somente ao tempo de setup (tsu) ou ao tempo de carga (tco) do registrador de destino
- Os estados S2, S3, S4 e S5 são os mais críticos, pois todos eles realizam a operação mais lenta, que é **AC ← AC + T**. Então, basta realizar a análise de timing desta operação ocorrendo no B.O. (Vide próximos slides)

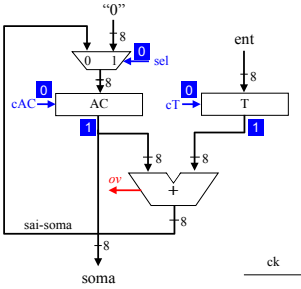


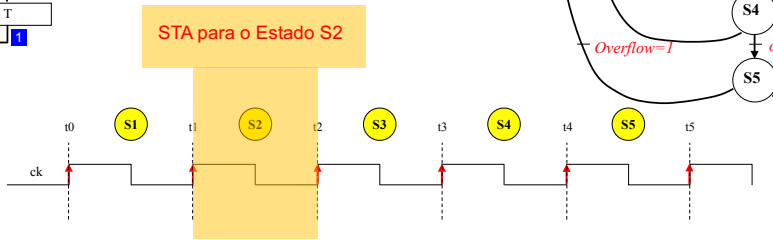
Processadores Dedicados

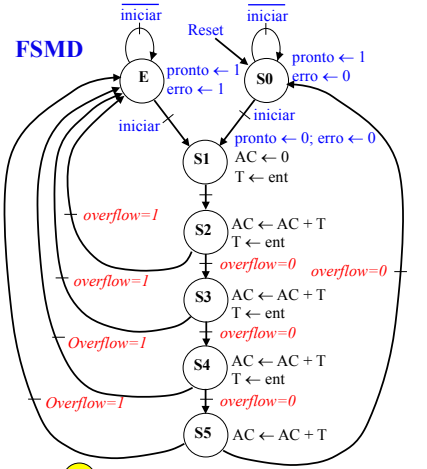
Exemplo 1: *Static Timing Analysis* (STA)

Assumindo 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
Sinais de comando sel, cAC, cT	atraso	tdcontrol	0 ns





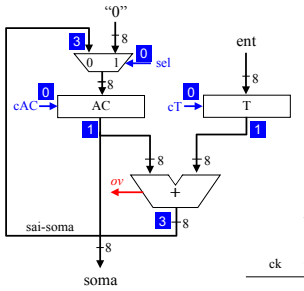


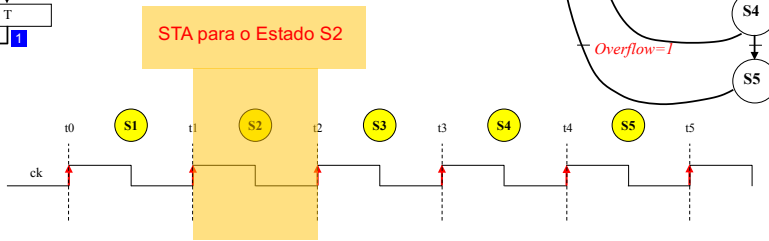
Processadores Dedicados

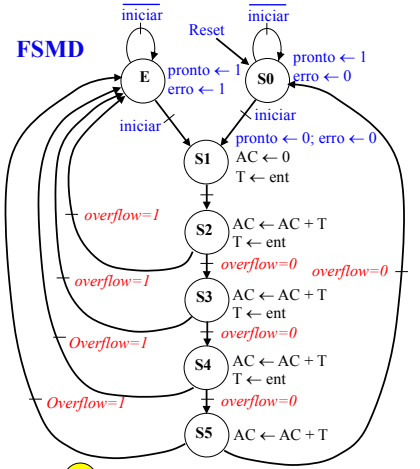
Exemplo 1: *Static Timing Analysis* (STA)

Assumindo 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
Sinais de comando sel, cAC, cT	atraso	tdcontrol	0 ns





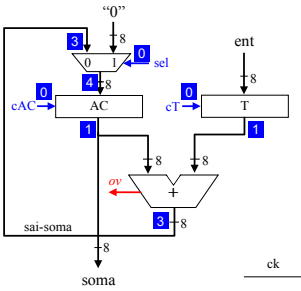


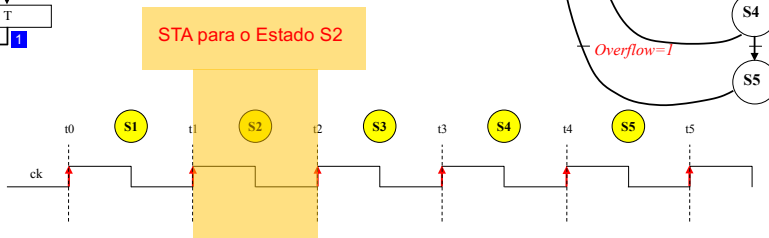
Processadores Dedicados

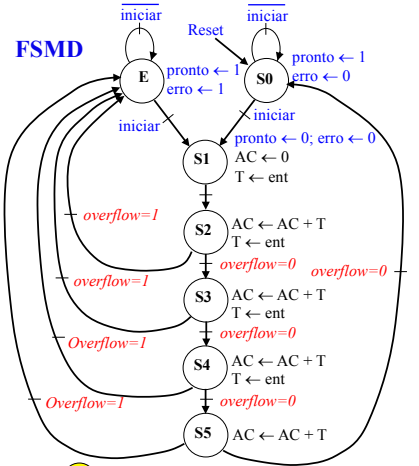
Exemplo 1: *Static Timing Analysis* (STA)

Assumindo 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
Sinais de comando sel, cAC, cT	atraso	tdcontrol	0 ns





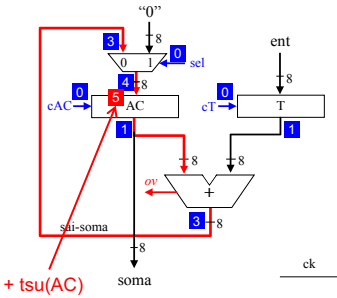


Processadores Dedicados

Exemplo 1: *Static Timing Analysis* (STA)

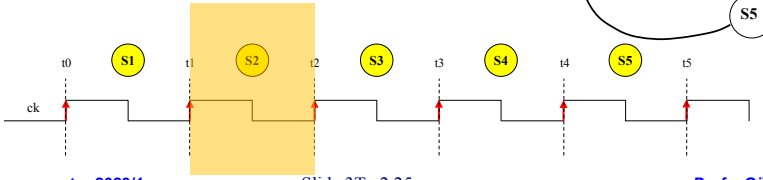
Assumindo as seguintes características temporais dos componentes

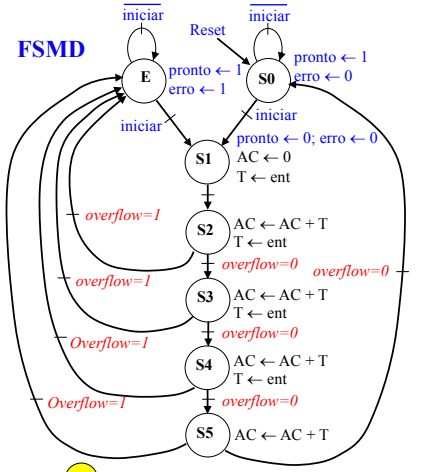
Componente	Característica	Símbolo	Valor
Registradores AC, T	tempo de setup	tsu	1 ns
Registradores AC, T	tempo de hold	th	0,5 ns
Registradores AC, T	tempo de carga	tco	1 ns
Somador completo (*full adder*)	atraso	tds	0,25 ns
Mux 2:1	atraso	tdmux	1 ns
Sinais de comando sel, cAC, cT	atraso	tdcontrol	0 ns



Logo, atraso crítico = D = 5 ns

E período mínimo T >= D = 5ns



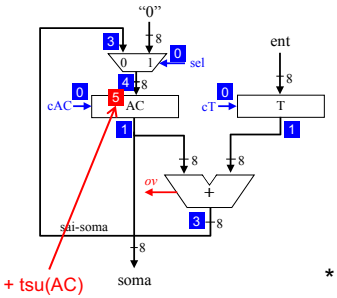


Processadores Dedicados

Exemplo 1: *Static Timing Analysis* (STA)

Assumindo 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
Sinais de comando sel, cAC, cT	atraso	tdcontrol	0 ns



Também seria possível estimar o atraso crítico D por*:

$$D = \max\{ tco_{AC} , tco_{T} \} + 8.tds + tdmux + tsu_{AC} =$$
$$= 1\ ns + 8\ .\ 0{,}25\ ns + 1\ ns + 1\ ns = \mathbf{5\ ns}$$

* Tal cálculo poderias ser empregado por projetistas muito experientes...

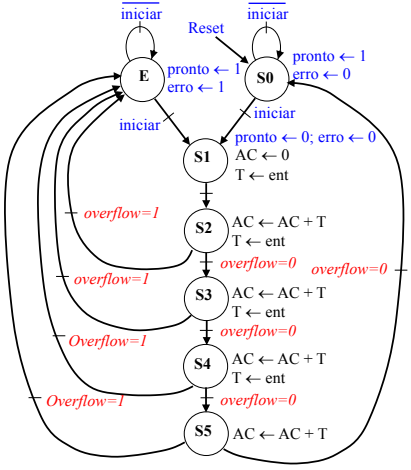
Exemplo 1: Estimativa de Desempenho

Tempo de Execução:

T_{exec} = nº de ciclos x T

= 5 ciclos x 5ns

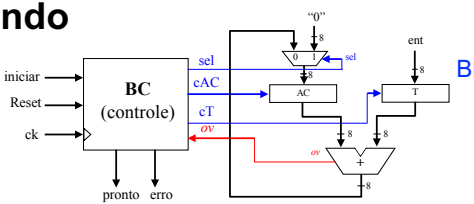
= 25 ns



Processadores Dedicados

Exemplo 1: Resumindo

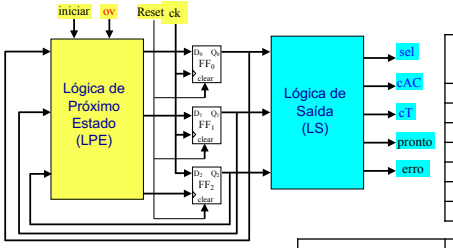
Sistema Digital somatório1



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)

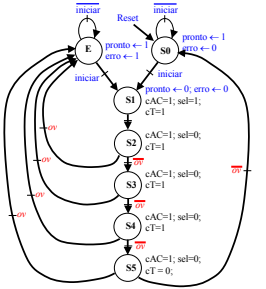
Q2	Q1	Q0	Sinais de comando			Saídas de controle	
			sel	cAC	cT	pronto	erro
1	1	1	X	0	0	1	1
0	0	0	X	0	0	1	0
0	0	1	1	1	1	0	0
0	1	0	0	1	1	0	0
0	1	1	0	1	1	0	0
1	0	0	0	1	1	0	0
1	0	1	0	1	0	0	0

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

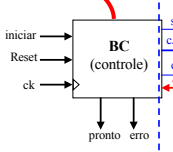
	Estimativa de Custo
B.O.	672 transistores
B.C.	7 estados
	5 sinais de controle

Descrição VHDL de um Sistema Digital no Nível RT

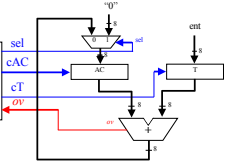
BC:
descrição comportamental da FSM




```
ENTITY bc IS
PORT (Reset, clk, iniciar, ov: 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=>
```



BO (Bloco Operativo):
descrição estrutural



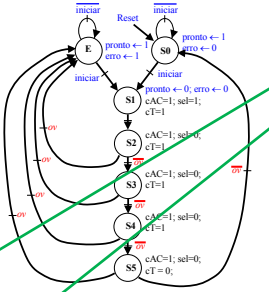

```
ENTITY bo IS
PORT (clk, sel, cAC, cT: IN STD_LOGIC;
ent: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
soma: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
ov: OUT STD_LOGIC;
END bo;

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;

.....
```

Descrição VHDL de um Sistema Digital no Nível RT

Somatorio1.vhd
descrição raiz (instancia bo.vhd e bc.vhd)

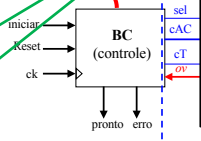
```
ENTITY somatorio1 IS
PORT (Reset, clk, iniciar : IN STD_LOGIC;
ent : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
soma: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
erro, pronto : OUT STD_LOGIC);
END somatorio1;

ARCHITECTURE estrutura OF somatorio1 IS

 COMPONENT bo IS
 PORT (clk, sel, cAC, cT: IN STD_LOGIC;
   ent: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
   soma: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   ov: OUT STD_LOGIC;
 END COMPONENT;

 COMPONENT bc IS
  PORT (Reset, clk, iniciar, ov: IN STD_LOGIC;
   cAC, sel, cT, pronto, erro: OUT STD_LOGIC);
END COMPONENT;

SIGNAL sel, cAC, cT, ov: STD_LOGIC;

 BEGIN
   b_operativo: bo PORT MAP (clk, sel, cAC, cT, ent, soma, ov);
   b_controle: bc PORT MAP  (Reset, clk, iniciar, ov, sel, cAC, cT);
END estrutura;
```

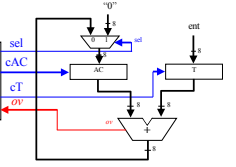
BC:
descrição comportamental da FSM




```
ENTITY bc IS
PORT (Reset, clk, iniciar, ov: 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=>
```

BO (Bloco Operativo):
descrição estrutural






```
ENTITY bo IS
PORT (clk, sel, cAC, cT: IN STD_LOGIC;
ent: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
soma: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
ov: OUT STD_LOGIC;
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;

.....
```

Descrição VHDL de um Sistema Digital no Nível RT

Hierarquia dos Arquivos VHDL