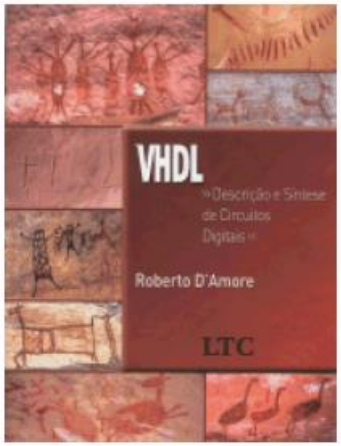
SEL0384/SEL0606 – Laboratório de Sistemas Digitais
Aula 7 – Construções Sequenciais

Prof. Dr. Maximiliam Luppe

Livro adotado:

VHDL - Descrição e Síntese de Circuitos Digitais
Roberto d'Amore

ISBN 85-216-1452-7
Editora LTC www.ltceditora.com.br



Para informações adicionais consulte: **www.ele.ita.br/~damore/vhdl**

Introdução

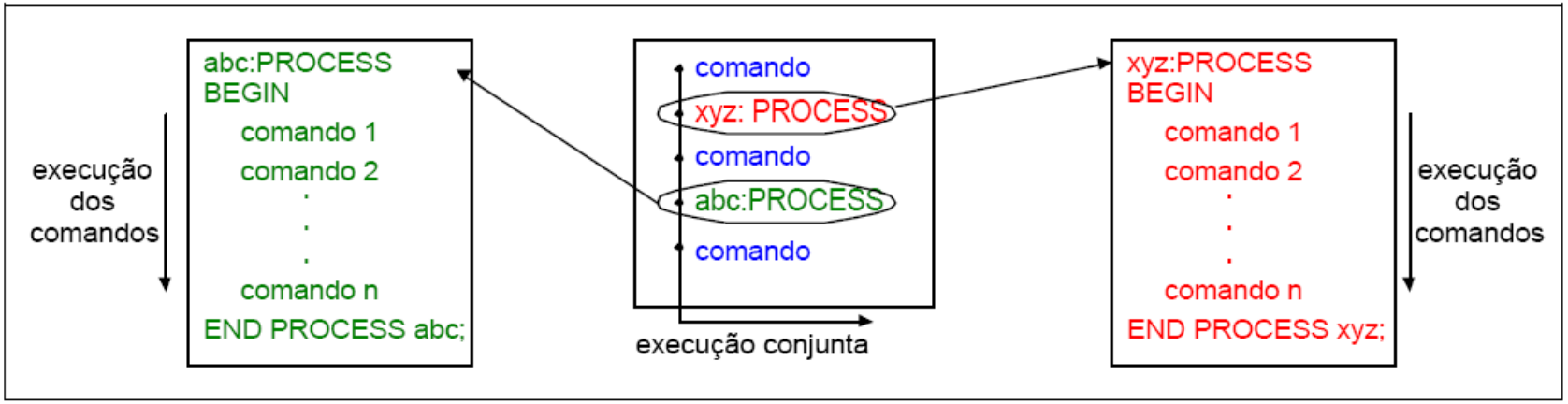
Tópicos

- Histórico
- Aspectos gerais da linguagem
- Síntese de circuitos

- Entidade de projeto
- Classes de objetos: constante, variável e sinal
- Tipos
- Operadores
- Construção concorrente WHEN ELSE
- Construção concorrente WITH SELECT
- Processos e lista de sensibilidade
- Construção seqüencial IF ELSE
- Construção seqüencial CASE WHEN
- Circuitos síncronos

Comandos seqüenciais

- **Comandos seqüenciais podem ocorrer em:**
  - processos
  - subprogramas
- **PROCESS**
  - é um comando concorrente
  - delimita uma região contendo código seqüencial
- **Lembrando**: uma descrição é composta de comandos concorrentes
  - todos são executadas concorrentemente

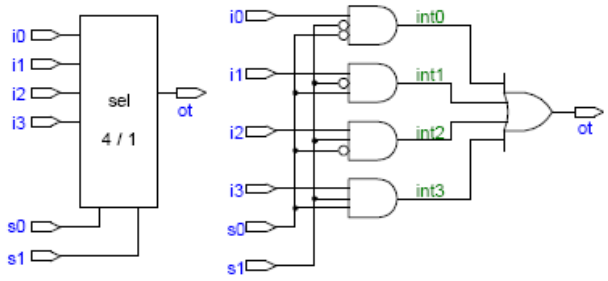


Lista de sensibilidade em processos

- **Após a comando PROCESS**:
  - possível declarar a lista de sensibilidade
- **Lista de sensibilidade**:
  - define quais sinais causam a execução do processo
- **Execução do processo ocorre se**:
  - um sinal da lista tem valor alterado
- **Iniciada a execução**:
  - comandos são avaliadas na seqüência

```
   abc: PROCESS (lista de sensibilidade)
        BEGIN
           comando_1;
           comando_2;
              ..
           comando_n;
END PROCESS abc;
```

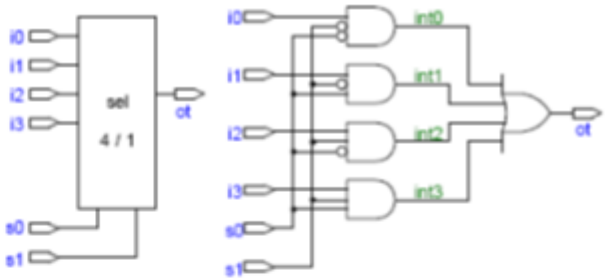
Atribuição de valor para um sinal
- região de código seqüencial



- **Exemplo** - circuito de seleção:
  - importante: nesta descrição,
    int0 int1 int2 int3 devem ser
    colocados na lista de sensibilidade

```
 1 ENTITY mux_p0 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_p0;
 6
 7 ARCHITECTURE teste OF mux_p0 IS
 8   SIGNAL int0, int1, int2, int3 : BIT;
 9 BEGIN
10   PROCESS (i0, i1, i2, i3, s0, s1, int0, int1, int2, int3)
11   BEGIN
12     ot <= int0 OR int1 OR int2 OR int3;
13     int0 <= i0 AND NOT s1 AND NOT s0;
14     int1 <= i1 AND NOT s1 AND     s0;
15     int2 <= i2 AND     s1 AND NOT s0;
16     int3 <= i3 AND     s1 AND     s0;
17   END PROCESS;
18 END teste;
```

Atribuição de valor para uma variável
- região de código seqüencial



- **Exemplo** - circuito de seleção:
  - importante: nesta descrição,
    int0 int1 int2 int3 devem ser atualizados
    antes de serem utilizados

```
 1 ENTITY mux_p0 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_p0;
 6
 7 ARCHITECTURE teste OF mux p0 IS
 8 BEGIN
 9   PROCESS (i0, i1, i2, i3, s0, s1)
10     VARIABLE int0, int1, int2, int3 : BIT;
11   BEGIN
12     int0 := i0 AND NOT s1 AND NOT s0;
13     int1 := i1 AND NOT s1 AND     s0;
14     int2 := i2 AND     s1 AND NOT s0;
15     int3 := i3 AND     s1 AND     s0;
16     ot <= int0 OR int1 OR int2 OR int3;
17   END PROCESS;
18 END teste;
```

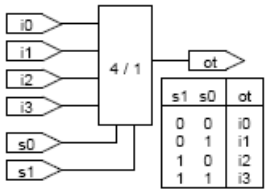
Construção IF ELSE

(similar: construção WHEN ELSE)

- **Execução condicional de um ou mais comandos seqüenciais**
- **Teste**: definido por uma lista de condições
- **Primeira condição verdadeira**: define os comandos executados
- **Condição de teste**: qualquer expressão que retorne BOOLEAN
- **Início da construção**: comando **IF**
- **Cláusulas ELSIF e ELSE**: opcionais
- **Formato da construção**:

```
IF   condicao_1 THEN
  comando_sequencial;
  comando_sequencial;
ELSIF condicao_2 THEN          -- clausula ELSIF opcional
  comando_sequencial;
  comando_sequencial;
ELSIF condicao_3 THEN
  comando_sequencial;
ELSE                          -- clausula ELSE opcional
   comando_sequencial;
END IF;
```

Exemplo: - circuito de seleção - IF ELSE



- **Comando seqüencial**
- **Nota**: s1 s2 agrupados no sinal **sel**
- **Lista de sensibilidade**: sinais i0 i1 i2 i3 sel

  - remoção de destes sinais: conseqüência?

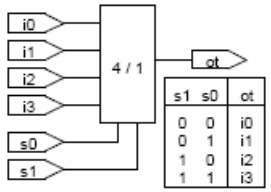
```
 1 ENTITY mux_4a IS
 2   PORT (i0, i1, i2, i3  :  IN  BIT;  -- entradas
 3         s0, s1          :  IN  BIT;  -- selecao
 4         ot              :  OUT BIT); -- saida
 5 END mux_4a;
 6
 7 ARCHITECTURE teste OF mux_4a IS
 8   SIGNAL sel : BIT_VECTOR(1 DOWNTO 0);
 9 BEGIN
10   sel <= s1 & s0;
11   abc: PROCESS (i0, i1, i2, i3, sel) --sinal "sel" inserido na lista
12   BEGIN
13     IF    sel = "00" THEN  ot <= i0;
14     ELSIF sel = "01" THEN  ot <= i1;
15     ELSIF sel = "10" THEN  ot <= i2;
16     ELSE                   ot <= i3;
17     END IF;
18   END PROCESS abc;
19 END teste;
```

Construção CASE WHEN (similar: construção WITH SELECT)

- **Execução condicional de um ou mais comandos seqüenciais**
- **A execução dos comandos**: controlada pelo valor de uma expressão
- **Todas condições da expressão de escolha devem ser consideradas**
  - não existe uma prioridade como na construção IF ELSE
- **Opções podem ser agrupadas**: - caracter | equivale a "ou"
  - TO e DOWNTO delimitam faixas de opções
- **Opções restantes**: palavra reservada OTHERS
- **Formato da construção**:

```
CASE expressao_escolha IS                                    -- expressao_escolha =

  WHEN condicao_1                => comando_b; comando_c;          -- condicao_1

  WHEN condicao_2 | condicao_3   => comando_d;        -- condicao_2 ou condicao_3

  WHEN condicao_4 TO condicao_9  => comando_d;       -- condicao_4 ate condicao_9

  WHEN OTHERS                    => comando_e; comando_f;  -- condicoes restantes
END CASE;
```

Exemplo: - circuito de seleção - CASE WHEN



- **Comando seqüencial**
- **Nota**: s1 s2 agrupados no sinal sel

```
 1 ENTITY mux_5a IS
 2   PORT (i0, i1, i2, i3  :  IN  BIT;
 3         s1, s0          :  IN  BIT;
 4         ot              :  OUT BIT);
 5 END mux_5a;
 6
 7 ARCHITECTURE teste OF mux_5a IS
 8   SIGNAL sel: BIT_VECTOR(1 DOWNTO 0);
 9 BEGIN
10   sel <= s1 & s0;
11   abc: PROCESS (i0, i1, i2, i3, sel) --sinal "sel" inserido na lista
12   BEGIN
13     CASE sel IS
14        WHEN "00"   => ot <= i0;
15        WHEN "01"   => ot <= i1;
16        WHEN "10"   => ot <= i2;
17        WHEN OTHERS => ot <= i3;
18     END CASE;
19   END PROCESS abc;
20 END teste;
```

Cuidados na descrição: comparações WHEN ELSE e IF ELSE

- **Construção WHEN ELSE**:
  - concorrente
  - 1ª condição válida: determina expressão transferida
- **Construção IF ELSE**:
  - seqüencial
  - 1ª condição válida: determina um conjunto de comandos a ser executado

```
sinal_destino <= expressao_a  WHEN opcao_1 ELSE
                 expressao_b  WHEN opcao_2 ELSE
                 expressao_c;
```

```
IF  condicao_1 THEN
  comando_a;
ELSIF condicao_2 THEN
  comando_c;
ELSE
  comando_d;
END IF;
```

Cuidados na descrição: comparações WITH SELECT e CASE WHEN

- **Construção WITH SELECT**:
  - concorrente
  - todas condições devem ser relacionadas
    - cada condição determina uma expressão transferida
- **Construção CASE WHEN**:
  - seqüencial
  - todas condições devem ser relacionadas
    - cada condição determina um conjunto de comandos a ser executado

```
WITH expressao_escolha SELECT
 sinal_destino <= expressao_a WHEN opcao_1,
                  expressao_b WHEN opcao_2,
                  expressao_c WHEN OTHERS;
```

```
CASE expressao IS
   WHEN condicao_1  => comando_a;
   WHEN OTHERS      => comando_e; comando_f;
END CASE;
```

Cuidados na descrição: emprego IF ELSE e CASE WHEN

- **Construção IF ELSE:**
  - possível uma nova expressão de escolha a cada estágio
  - excesso de liberdade pode levar a erros:
    exemplo: nem todas condições cobertas por descuido na descrição
    criação de elementos de memória desnecessários

- **Construção CASE WHEN:**
  - expressão de escolha única
  - condições de escolha não cobertas:
    mensagem de erro na etapa de compilação

- **Sempre que possível:**
  - empregar a construção CASE WHEN

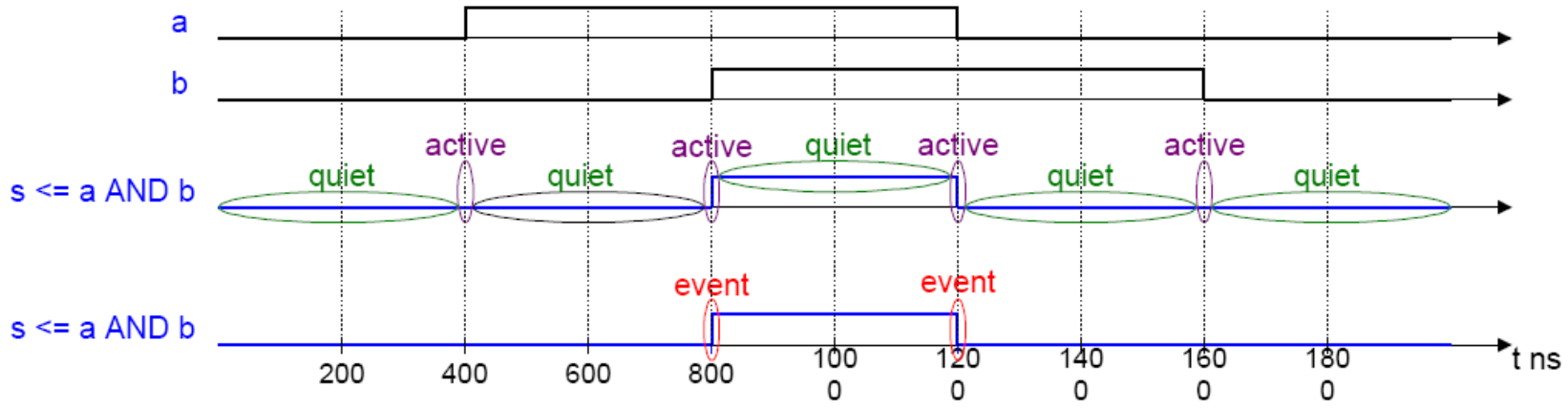
Estratégias de descrição de circuitos síncronos

Tópicos

- Registrador sensível a nível
- Registrador sensível a borda - inicialização síncrona
- Registrador sensível a borda - inicialização assíncrona
- Registrador sensível a borda com habilitação para sinal de relógio
- Máquinas de estado finito
- Contadores
- Cuidados na descrição

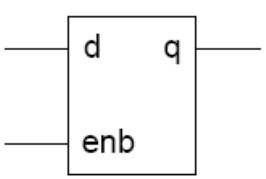
Definição de termos relativos a sinais (útil para entender alguns atributos)

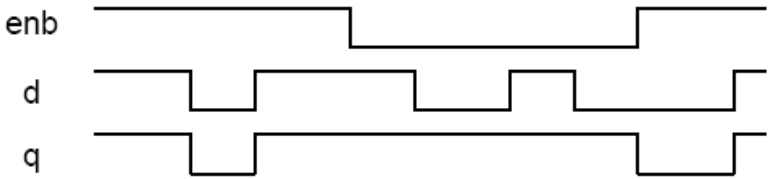
- **active**: novo valor atribuído ao sinal (o valor pode ser o mesmo)
- **quiet**: condição oposta a active
- **event**: mudança do valor de um sinal
- **Exemplo**:



Registrador sensível a nível

- **Exemplo**:

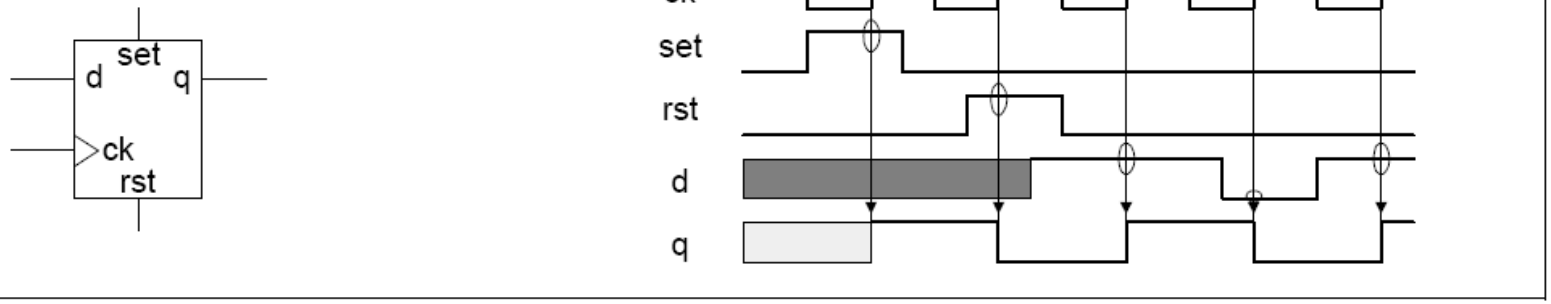




- **Formato geral**: empregando processo
  - sinal **d**: deve estar na lista de sensibilidade

```
PROCESS (enb, d)
BEGIN
  IF (enb = '1') THEN  d <= q ;
  END IF ;
END PROCESS;
```

Registrador sensível a borda - inicialização síncrona - (exemplo)

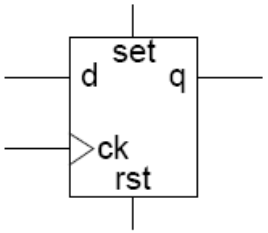


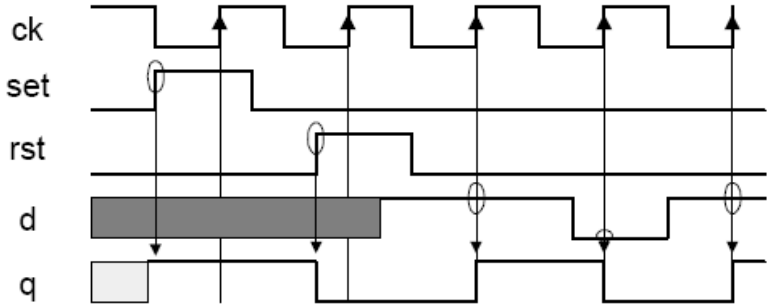
- **Formato geral**: empregando processo

  - apenas sinal ck: necessita estar na lista de sensibilidade

```
PROCESS (ck)
BEGIN
  IF (ck'EVENT and ck = '1') THEN      -- detecta borda de subida do sinal ck
    IF    rst = '1' THEN  q <= '0';    -- sequencia de eventos sincronos com ck
    ELSIF set = '1' THEN  q <= '1';    --     .
                                       --     .
                                       --     .
    ELSE                  q <= d;      -- finaliza com a memorizacao
    END IF;
  END IF;
END PROCESS;
```

Registrador sensível a borda - inicialização assíncrona - (exemplo)

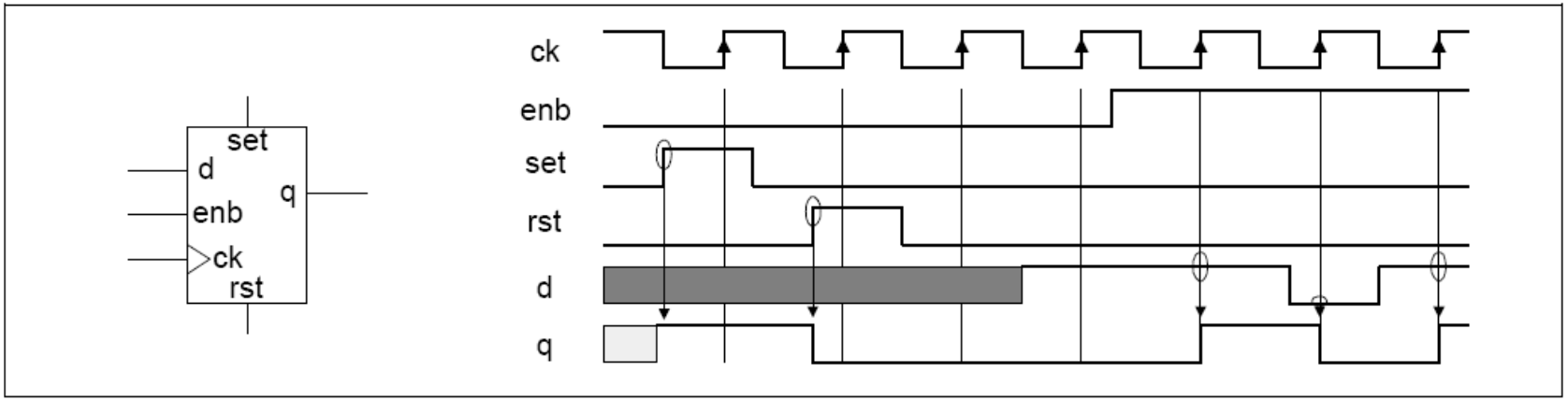




- **Formato geral**: empregando processo
  - sinais ck, rst e set necessitam estar na lista de sensibilidade

```
PROCESS (ck, rst, set)
  BEGIN
    IF       (rst = '1')          THEN q <= '0'; -- eventos assincronos
    ELSIF    (set = '1')          THEN q <= '1'; --    .
                                                 --    .
                                                 --    .
    ELSIF (ck'EVENT AND ck ='1') THEN q <=d;     -- detecta borda de subida de ck
    END IF;
 END PROCESS;
```

Registrador sensível a borda com habilitação para sinal de relógio



- **Formato geral**: empregando processo
  - sinais rst e set necessitam estar na lista de sensibilidade (para inic. assíncrona)
  - sinal enb não é necessário na lista de sensibilidade

```
PROCESS (ck, rst, set)
  BEGIN
    IF       (rst = '1')          THEN q <= '0'; -- eventos assincronos
                                                 --    .
    ELSIF (ck'EVENT AND ck ='1')                 -- detecta borda de subida de ck
      IF (enb = '1') THEN   q <=d;               -- armazena se habilitado
      END IF;
    END IF;
 END PROCESS;
```