Universidade Tecnológica Federal do Paraná – Toledo
Engenharia da Computação – COENC

Lógica Reconfigurável

Código Sequencial

IF-THEN-ELSE

Tiago Piovesan Vendruscolo

Código Sequencial

- O código VHDL sempre é executado de forma concorrente. Para o código ser executado de forma sequencial, o código deve ser inserido dentro de uma estrutura que force o seu processamento de forma sequencial. Essa estrutura é chamada de **PROCESS**.
- PROCESS é utilizado sempre dentro da arquitetura.
- Dentro do PROCESS são executados os seguintes comandos sequenciais:
  - IF-THEN-ELSE;
  - CASE-WHEN;
  - FOR-LOOP;
  - WHILE-LOOP;
  - WAIT (a partir do VHDL 2008).

Código Sequencial

- Estrutura do PROCESS

Pode utilizar um nome para cada PROCESS
Ex: Label : PROCESS (a)

```
entity cod_sequencial is
port (a,b : IN std_logic;
      c,d: OUT std_logic);
end cod_sequencial;

architecture exemplo of cod_sequencial is
begin
    PROCESS (a)
      variable x: std_logic:= '1';
    begin
        if (a='1' and a'event) then
            (código)
           else
            (código)
        end if;
        c <= x;
    end PROCESS;
end exemplo;
```

Lista de sensibilidade, pode conter vários objetos.

Borda de subida (muito utilizado para controle de processos usando clock.

Final do PROCESS

Código Sequencial

- Estrutura do PROCESS

```
entity cod_sequencial is
port (a,b : IN std_logic;
      c,d: OUT std_logic);
end cod_sequencial;

architecture exemplo of cod_sequencial is
begin
    PROCESS (a)
      variable x: std_logic:= '1';
    begin
        if (a='1' and a'event) then
            x:= a AND B;
            else
            x:= a XOR B;
        end if;
        c <= x;
    end PROCESS;
end exemplo;
```

A variável x é local, existe apenas dentro desse process.

É necessário passar o valor da VARIABLE (local) x para um SIGNAL ou PORT (globais) antes de fechar o PROCESS.

<= Atribui o valor a um SIGNAL ou PORT.

:= Atribui o valor a uma VARIABLE.

Código Sequencial

- Comando WAIT UNTIL

```
entity wait_until is
    port ( d, clk : in  std_logic;
           q      : out std_logic);
end wait_until;

architecture funcao of wait_until is

begin
    process
    begin
    wait until clk'event and clk='1';
    q<=d;
    end process;
end funcao;
```

O Uso de WAIT UNTIL especifica implicitamente que somente o clk faz parte da lista de sensibilização.

São equivalentes

```
process (clk)
begin
if (clk'event and clk='1') then
```

Código Sequencial

Todas as sentenças abaixo são sintetizáveis e equivalentes:

Detecção de subida de clock:

```
IF (clk'EVENT AND clk = '1')...
IF (NOT clk'STABLE AND clk = '1')...
WAIT UNTIL (clk'EVENT AND clk = '1');
IF RISING_EDGE (clk)...
```

Detecção de descida de clock:

```
IF (clk'EVENT AND clk = '0')...
IF (NOT clk'STABLE AND clk = '0')...
WAIT UNTIL (clk'EVENT AND clk = '0');
IF FALLING_EDGE (clk)...
```

Código Sequencial

- Para efeitos de síntese de circuitos, WAIT UNTIL somente pode ser usado se ele for a primeira atribuição do processo
- Na verdade, 'EVENT é redundante no comando WAIT UNTIL e portanto poderíamos simplificar para:

WAIT UNTIL clk='1';

o que se refere ao sinal clk se tornar igual a "1"

- Entretanto, algumas ferramentas de síntese de circuitos a partir de VHDL exigem a inclusão do atributo "EVENT"

```
IF (NOT clk'STABLE AND clk = '1')...
```

Comando IF-THEN-ELSE

- Estrutura do IF-THEN-ELSE

```
process(A,controle)
begin
   if controle='1' then
      if A>3 then
         B<=1;
      else
         B<=5;
      end if;
   else
      B<=0;
   end if;
end process;
```

- A sequencia indica a prioridade nos testes das opções:

```
if x='1' then
   A := 0;
end if;
if y='1' then
   A := 2;
end if;
```

< Equivalentes >

```
if x='1' then
   A := 0;
elsif y='1' then
   A:= 2;
end if;
```

Comando IF-THEN-ELSE

- Uso do RESET:
  - Reset Assíncrono: Reseta os registradores independente do clock.

```
PROCESS (clk, rst)
BEGIN
   IF (rst='1') THEN
      registrador <= '0';
   ELSIF (clk'EVENT AND clk='1') THEN
--
--código sequencial
--
   END IF;
END PROCESS;
```

Para funcionar de forma assíncrona, o sinal “rst” deve estar na lista de sensibilidade

Comando IF-THEN-ELSE

- Uso do RESET:
  - Reset Síncrono: Reseta os registradores obedecendo o clock.

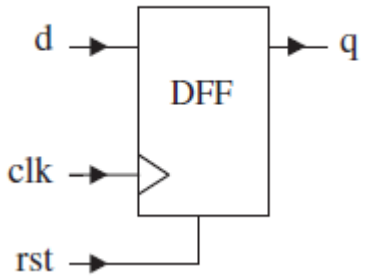
Nesse caso, apenas o clock entra na lista de sensibilidade.

```
PROCESS (clk)
BEGIN
   IF (clk'EVENT AND clk='1') THEN
      IF (rst='1') THEN
         registrador <= '0';
      ELSE
--
--código sequencial
--
      END IF;
   END IF;
END PROCESS;
```

Comando IF-THEN-ELSE

- Exercício 1: Projete um FF do tipo D, com clock na borda de subida e sinal de reset assíncrono (quando rst = ‘1’, a saída vai a zero).

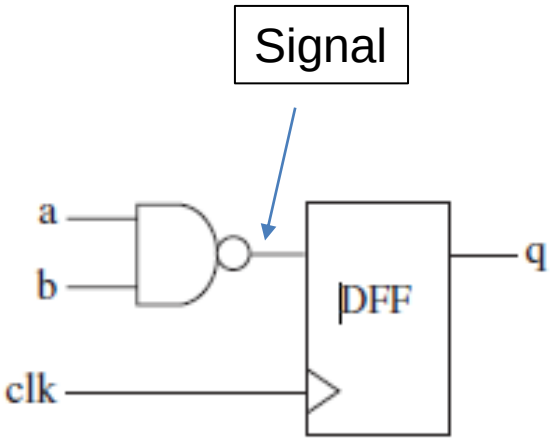
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------
ENTITY flipflop IS
PORT ( d, clk, rst: IN STD_LOGIC;
q: OUT STD_LOGIC);
END flipflop;
---------------------------------------
ARCHITECTURE behavior OF flipflop IS
BEGIN
PROCESS (rst, clk)
BEGIN
   IF (rst='1') THEN
      q <= '0';
   ELSIF (clk'EVENT AND clk='1') THEN
      q <= d;
   END IF;
END PROCESS;
END behavior;
```



Cada vez que rst e clk mudam, o código dentro do PROCESS é executado.

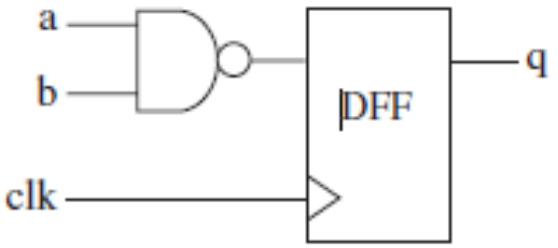
Comando IF-THEN-ELSE

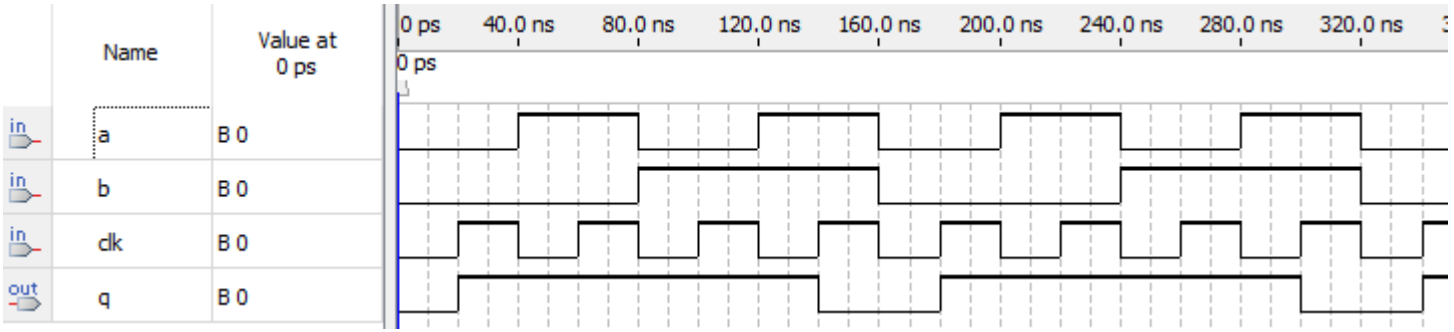
- Exercício 2: Projete o circuito proposto abaixo:
  - Flip-flop tipo D com clk na borda de subida, em conjunto com uma porta NAND na entrada.



Comando IF-THEN-ELSE

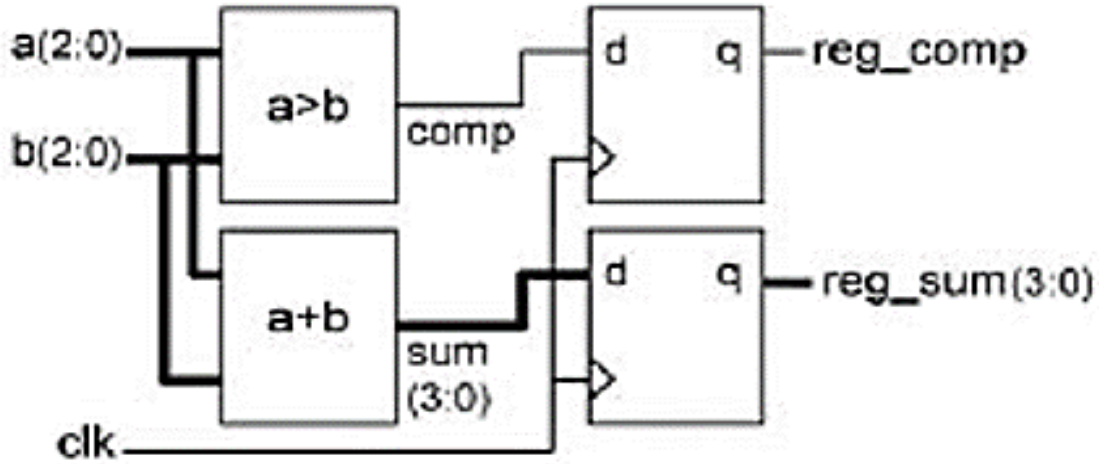
```
ENTITY exercicio1 IS
  PORT ( a, b, clk: IN std_logic;
         q: OUT std_logic);
END exercicio1;
-------------------------------------
ARCHITECTURE funcao OF exercicio1 IS
  SIGNAL temp : std_logic;
BEGIN
  temp <= a NAND b;
  PROCESS (clk)
  BEGIN
    IF (clk'EVENT AND clk='1') THEN
    q<=temp;
    END IF;
  END PROCESS;
END funcao;
```





Comando IF-THEN-ELSE

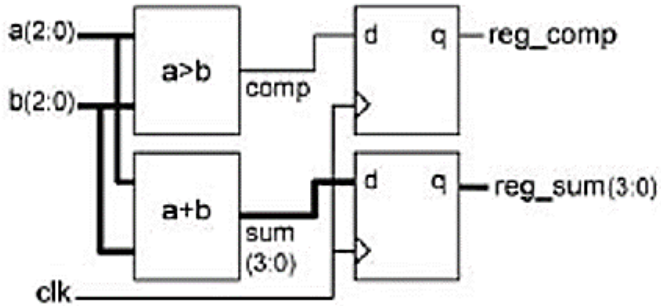
- Exercício 3: Projete e simule o circuito somador – comparador com registrador abaixo:



- Dica: Aula sobre o comando WHEN-ELSE.

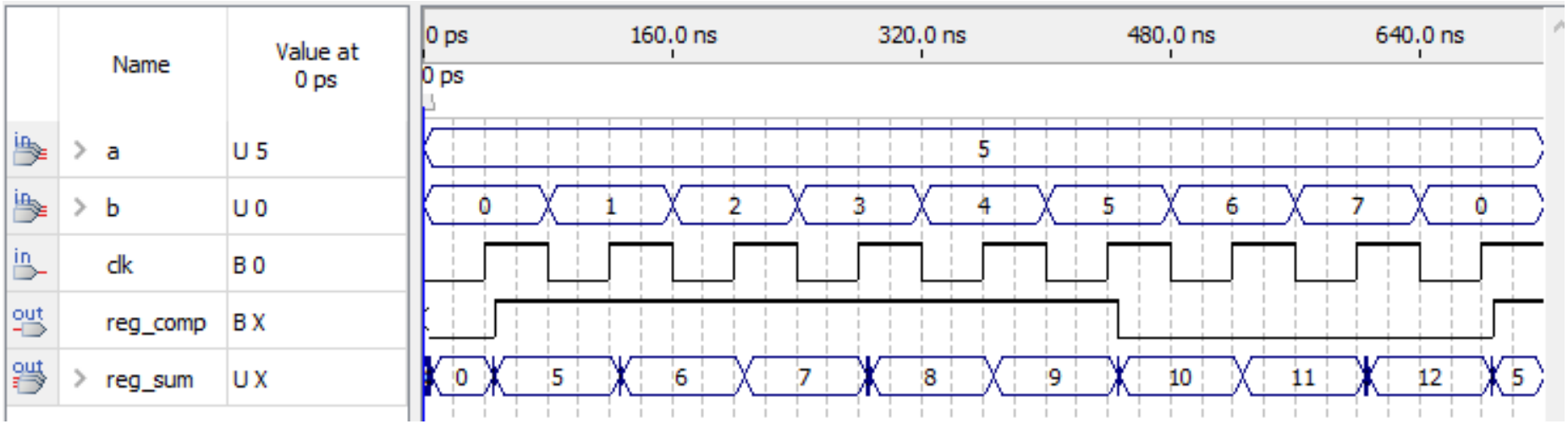
Comando IF-THEN-ELSE

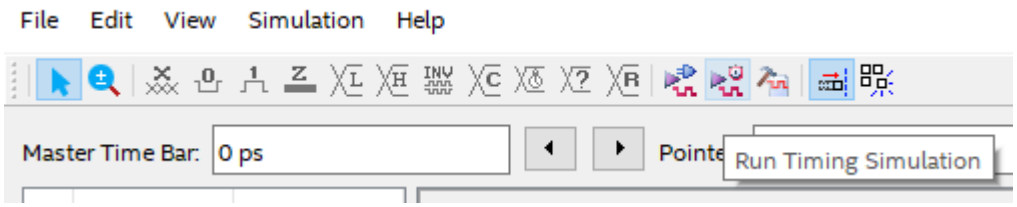
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY reg_comp_add IS
    PORT (clk: IN STD_LOGIC;
      a, b: IN INTEGER RANGE 0 to 7;
      reg_comp: OUT STD_LOGIC;
      reg_sum: OUT INTEGER RANGE 0 TO 15);
END reg_comp_add;

ARCHITECTURE funcao of reg_comp_add IS
   SIGNAL comp: STD_LOGIC;
   SIGNAL sum: INTEGER RANGE 0 TO 15;
BEGIN
   comp <= '1' WHEN a>b ELSE '0';
   sum <= a + b;
   PROCESS (clk)
   BEGIN
      IF (clk'EVENT AND clk='1') THEN
         reg_comp<=comp;
         reg_sum<=sum;
      END IF;
   END PROCESS;
END funcao;
```



Comando IF-THEN-ELSE

Simulação no tempo





Comando IF-THEN-ELSE

- Exemplo: Comparação utilizando BOOLEAN

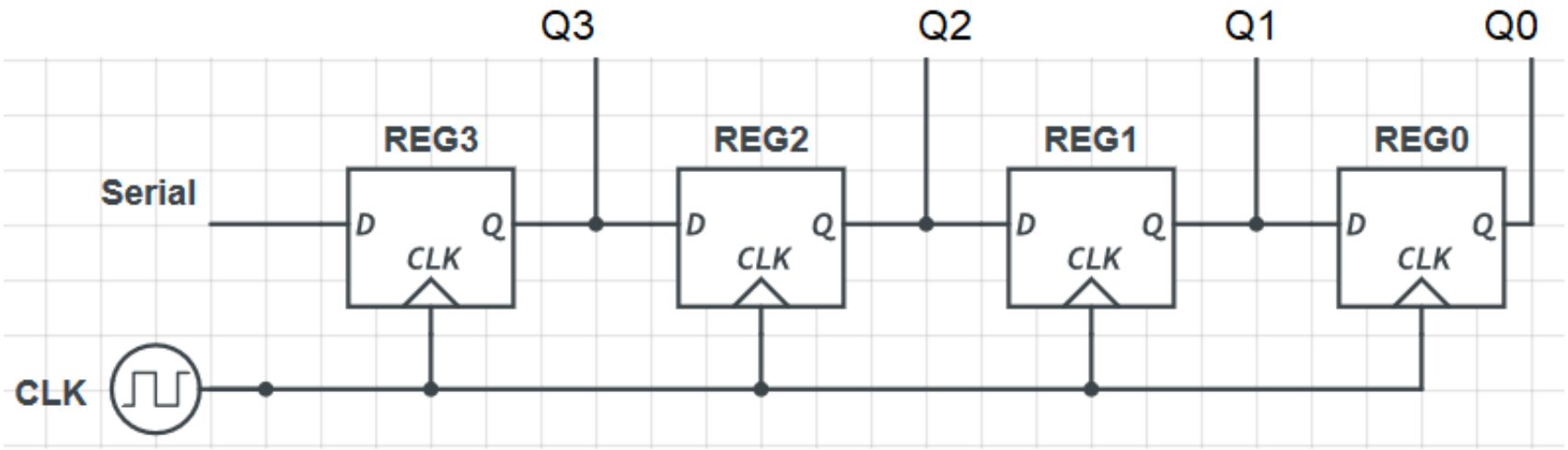
```
ENTITY comp IS
    PORT (A, B: IN STD_LOGIC;
          IGUAL: OUT STD_LOGIC;
          IGUAL_B: OUT BOOLEAN);
END comp;

ARCHITECTURE funcao of comp IS
   SIGNAL COMPARACAO: BOOLEAN;
BEGIN
   PROCESS (A, B)
   BEGIN
      COMPARACAO <= (A = B);
      IGUAL_B <= (A = B);
         IF COMPARACAO THEN
            IGUAL<='1';
         ELSE
            IGUAL<='0';
         END IF;
   END PROCESS;
END funcao;
```

```
IGUAL_B: BUFFER BOOLEAN
```

O que fazer caso queira utilizar diretamente “IGUAL_B”?

Comando IF-THEN-ELSE

- Exercício 4: Projete e implemente na FPGA o registrador-deslocador abaixo, com entrada serial.



- Utilizar como entrada serial: KEY [0], para clk: KEY [1] e para saída paralela[Q0-3]: LED[0-3].

Comando IF-THEN-ELSE

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY shiftreg1 IS
      PORT ( serial, clk :IN STD_LOGIC;
             Q           :BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0));
END shiftreg1;

ARCHITECTURE funcao OF shiftreg1 IS
BEGIN
    PROCESS (clk)
    BEGIN
      IF clk'EVENT AND clk = '1' THEN
                   Q(0)  <=  Q(1);
                   Q(1)  <=  Q(2);
                   Q(2)  <=  Q(3);
                   Q(3)  <=  serial;
      END IF;
    END PROCESS;
END funcao;
```

BUFFER: Utilizado quando um sinal de saída precisa ser lido internamente.

Próxima aula

- Códigos sequenciais: CASE - WHEN