ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

Circuitos Seqüenciais

Descrição VHDL

prof. Dr. César Augusto M. Marcon
prof. Dr. Edson Ifarraguirre Moreno

Processos

- **Circuitos seqüenciais são normalmente descritos, em VHDL, dentro de processos com lista de sensitividade**
  - Um processo VHDL somente é avaliado quando pelo menos um dos sinais da lista de sensitividade varia. Caso nenhum destes varie, os sinais dentro do processo mantém-se inalterados. Ou seja, o valor anterior é memorizado
- **Lista de sensitividade**
  - **Contém os sinais que, quando variam, implicam na avaliação do processo**
  - **Em HW representam os sinais que controlam a operação do algoritmo descrito no processo**
- **Processos são utilizado para descrever algoritmos**
  - Permitem tanto descrições seqüenciais como paralelas
  - Nesta disciplina utilizaremos essencialmente para a descrição de circuitos seqüenciais
- Exemplo:

```
process(A, B)
begin
    if A = '1' then
        x <= '0';
    end if;
    if B = '1' then
        x <= din;
    end if;
end process;
```

Neste processo, o sinal **X** somente pode variar se **A** ou **B** variarem

PERGUNTA: O que acontece com **X** se ambos variarem para o valor ‘1’ ?

Atribuição de Variáveis

- Variáveis VHDL são sinais com comportamento igual à variáveis de software com escopo local, mas persistência de valor
  - São declaradas e usadas internamente a processos
  - As atribuições são seqüenciais, ou seja, a ordem da variáveis importa
  - O valor das variáveis é mantido ao término do processo, sendo este usado na próxima avaliação do processo

Exemplo:

```
process(teste)
        variable a: std_logic := '1';
        variable b: std_logic;
        variable c: integer := 3;
begin
        b := a;
        b := not b;   -- Qual o valor de b antes e depois?
        c := c + teste;
end process;
```

Comando Condicional If

- **Utilizado em processos, podendo ser aninhado**
  - Exemplo:

```
process(A, B, control)
begin
    if control = '1' then
        if X > 3 then
            Z <= B;
        else
            Z <= A;
        end if;
    else
        Z <= A;
    end if;
end process;
```

- **A seqüência na qual estão definidos os “ifs” implica na prioridade das ações**

```
if x then
    T := A;
end if;
if y then
    T := B;
end if;
if z then
    T := C;
end if;
```


```
if z then
    T := C;
elsif y then
    T := B;
elsif x then
    T := A;
end if;
```

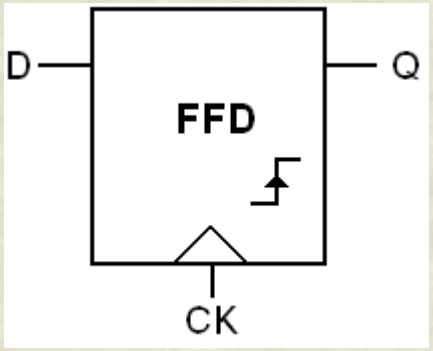
Comando de Seleção Case

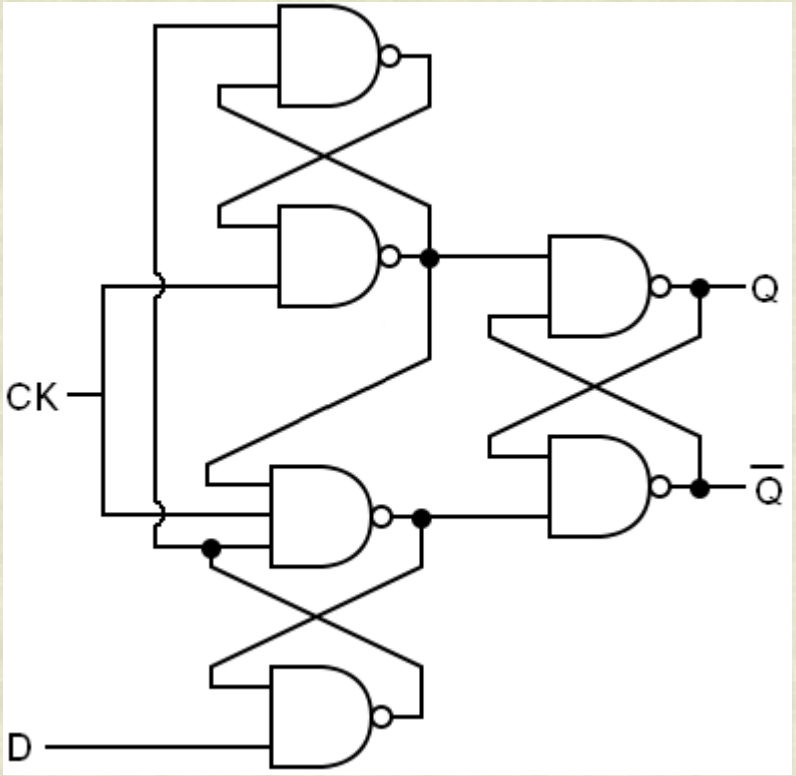
- O comando case é uma forma semelhante a apresentada pelo comando if-eslif
- É uma estrutura, muitas vezes preferida para a descrição de máquinas de estado e decodificadores
- Exemplo:

```
case opcode is
   when x"00" =>                  -- escolha simples
        Bloco de Ações_1;
   when x"01" | x"A1" =>          -- pipe representa um "ou"
        Bloco de Ações_2;
   when x"02" to x"A0" =>         -- intervalo
        Bloco de Ações_3;
   when others =>                 -- restante
        null;
end case;
```

Descrição VHDL de um Flip-Flop D

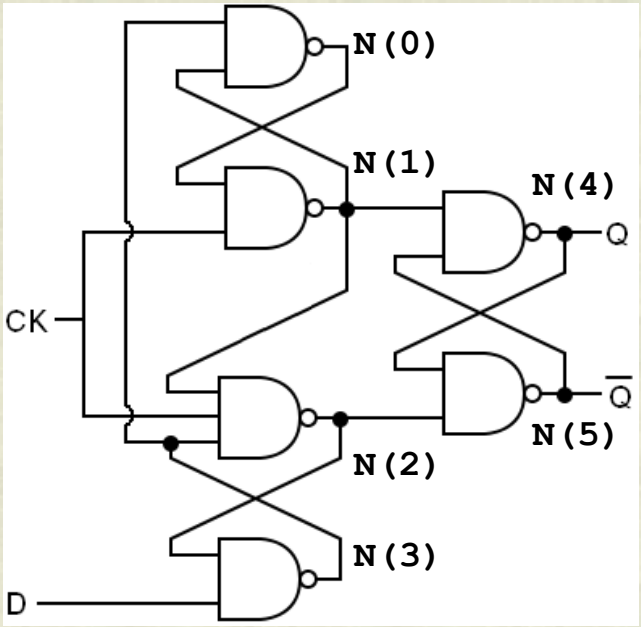
- Dados os FFDs vistos na aula passada e ilustrados abaixo, como seriam possíveis descrições VHDLs dos mesmos?





Descrição VHDL de um Flip-Flop D

- Descrição estrutural

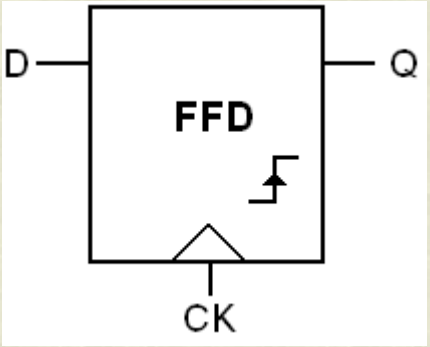



```
entity FlipFlopD_Estrutural is
    port
    (
       ck, d: in std_logic;
       q: out std_logic
    );
end FlipFlopD_Estrutural;

architecture FFD of FlipFlopD_Estrutural is
   signal N: std_logic_vector(5 downto 0);
begin
   N(0) <= N(1) nand N(3);
   N(1) <= N(0) nand ck;
   N(2) <= not (N(2) and ck and N(3));
   N(3) <= N(2) nand d;
   N(4) <= N(1) nand N(5);
   N(5) <= N(2) nand N(4);
   q <= N(4);
end FFD;
```

Descrição VHDL de um Flip-Flop D

- Descrição comportamental




```
entity FlipFlopD is
    port
    (
       ck, d: in std_logic;
       q: out std_logic
    );
end FlipFlopD;

architecture FFD of FlipFlopD is
begin
    process(ck)
    begin
        if ck'event and ck = '1' then
              q <= d;
        end if;
    end process;
end FFD;
```

Registradores

- Registrador é um elemento de memorização que armazena um vetor de bits
- Logicamente equivalem a um vetor de Flip-flops do tipo D
  - Representação e funcionalidades semelhantes
- São normalmente declarados em processos, tendo sinais de controle (clock, reset, etc.) inseridos em uma lista de sensitividade
- Exemplo:

```
process(ck, reset)
begin
    if reset = '1' then
        reg <= (others =>'0');
    elsif ck'event and ck = '1' then
        reg <= entrada;
    end if;
end process;
```

- Exercício:
  - Descreva o diagrama de blocos do registrador acima
  - Quantos bits tem o registrador da descrição VHDL acima?

Registradores

– Arquitetura de registrador de 8 bits com chip enable, e reset assíncrono ao relógio

```
entity regnbit is
    port
    (
        ck, rst, ce: in std_logic;
        D: in  std_logic_vector(7 downto 0);
        Q: out std_logic_vector(7 downto 0)
    );
end regnbit;

architecture regn of regnbit is
begin
    process(ck, rst)
    begin
        if rst = '1' then
            Q <= (others => '0');
        elsif ck'event and ck = '0' then
            if ce = '1' then
                Q <= D;
            end if;
        end if;
    end process;
end regn;
```

Registradores

Exercícios baseados na descrição do slide anterior:

1. Faça a descrição de um registrador de 16 bits sensível a borda de descida e com reset síncrono
2. O que aconteceria se tirássemos o sinal rst (reset) da lista de sensitividade do processo?
3. Qual o efeito de declarar um reset síncrono ao invés de um reset assíncrono ao clock? Neste caso, o que aconteceria se tirássemos o sinal rst da lista de sensitividade do processo?

Registrador de Deslocamento

- Registrador de deslocamento armazena um vetor de bits e a cada evento de relógio desloca um número programável de estágios os bits em uma direção igualmente programável

- Pode ser utilizado para operações de criptografia, multiplicação e divisão em potência de 2, serialização e desserialização, ...

- Exemplo de registrador de deslocamento de 3 bits com reset, sensível à borda de subida do relógio. Este desloca o registrador um estágio para a esquerda a cada ciclo de relógio:

```
process(clock, reset)
begin
    if reset = '1' then
        R <= (others => '0');
    elsif clock'event and clock = '1' then
        R(0) <= entrada;
        R(1) <= R(0);
        R(2) <= R(1);
    end if;
end process;
```

Registrador de Deslocamento

Exercícios

1. **Desenhe o circuito do registrador utilizando flip-flops D**
2. **A ordem das atribuições para R(0), R(1) e R(2) é importante ? O que ocorreria se fosse uma linguagem de programação tipo C?**
3. **Escreva o código para um registrador de 8 bits com deslocamento à esquerda e a direita. Utilize um sinal de controle chamado direção ('0' - deslocamento para a direita, '1' - deslocamento para a esquerda)**
4. **Faça uma entidade e arquitetura de um circuito registrador, que receba um sinal de inicialização e armazena no registrador sempre que a porta reset tiver o valor '1'. Considere que o reset é síncrono ao relógio. O registrador de entrada deve ter 8 bits, e o resultado deve ter 16 bits. Também existe uma porta de 3 bits indicando qual o deslocamento que deve ser feito (1 a 8 vezes) . Considere que o deslocamento é sempre para a direita**
5. **Faça um registrador de deslocamento que tenha uma carga em paralelo controlada pelo sinal load. O número de ciclos para deslocar pode ser de 1 a 4, dependendo do sinal shift. O sentido de deslocamento pode ser esquerda ou direita, conforme sinal sentido. O registrador tem 16 bits**
6. **Faça um circuito serializador que recebe um sinal (em paralelo de 8 bits e transforme em um sinal serial de 1 bit). Considere os sinais de controle ck (relógio que dá a cadência da serialização), load (carga do valor paralelo – operação na borda de subida de ck), rst (reset do sistema), ready (sinal que informa que todo o número já serializou). Faça a entidade e a arquitetura e faça um diagrama de blocos ilustrativo.**

Considerações

- **Atribuição dentro/fora de processos**

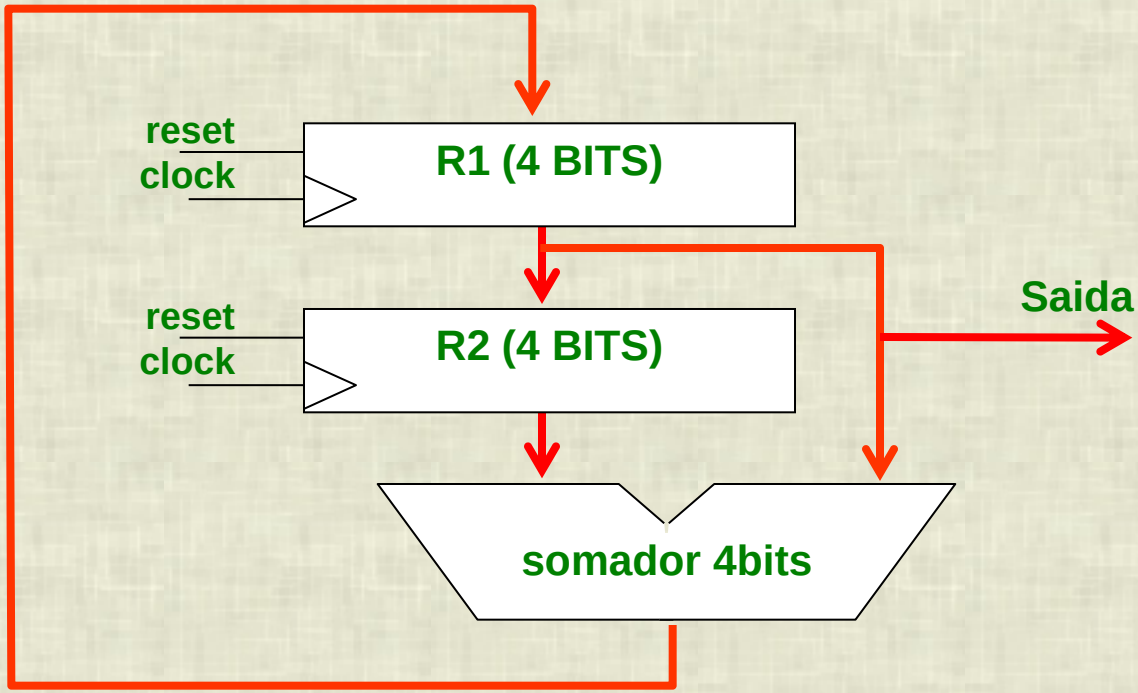
```
process(clock)
begin
    if clock'event and clock = '1' then
        A <= entrada;
        B <= A;
        C <= B;
        Y <= B and not (C);     -- dentro do process
    end if;
end process;
X <= B and not (C);             -- fora do process
```

Qual a diferença de comportamento nas atribuições X e Y?

- **Conclusão**
  - **Sinais atribuídos em processos, com controle de clock, serão sintetizados com flip-flops**
  - **Sinais fora de processos ou em processos sem variável de sincronismo (clock) serão sintetizados com lógica combinacional**

Exercícios

1. Descreva em VHDL o circuito ilustrado abaixo
   - **Quando o sinal de reset for ‘1’, os registradores R1 e R2 armazenam “0001” e “0000”, respectivamente**
   - **Diga o que este circuito faz**



Exercícios

2. Faça um test bench em vhdl para o circuito do exercício 1 e determine o conteúdo de R1 e R2 para os 5 primeiros ciclos de relógio
3. Para gerar o clock e reset utilize dentro do test bench o seguinte código:

```
reset <= '1', '0' after 5ns;
process
begin
    clock <= '1' after 10ns, '0' after 20ns;
    wait for 20ns;
end process;
```

4. Calcule qual a freqüência de relógio que está sendo simulada
5. Faça agora uma modificação no relógio para que o mesmo tenha uma freqüência de 333MHz, inicie com 0 e o intervalo de tempo que fica em 1 seja o dobro do intervalo de tempo que fica em 0