ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

Circuitos Seqüenciais

Contadores

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

Introdução

- **Contadores são circuitos de natureza seqüencial. A cada evento de relógio permitem que o estado do circuito faça uma transição para um estado que representa o valor da próxima contagem**
- **Normalmente, além do sinal de relógio, os contadores têm entradas de habilitação (chip enable), carga, direção da contagem, inicialização e pausa**
  - Este conjunto de sinais depende de requisitos da aplicação
- **A complexidade de um contador depende da função de cálculo de próximo estado e do número de bits que será efetuada a contagem**
  - Em geral, a função de próximo estado é bastante simples e o número de bits não é representativo

- **Exemplos de contadores**
  - Contador hexadecimal de 4 bits: conta de 0000 a 1111 (0x0 a 0xF)
    - O início de uma nova contagem é feita de forma transparente
  - Contador decimal de segundos, com 6 bits: conta de 0 a 59
    - Necessita de um teste para reiniciar a contagem
  - Contador gray de 3 bits: conta de 000 a 010, sendo que a contagem é feita de forma a variar apenas um bit
    - Função de transição mais complexa que os anteriores

Exemplo de um Contador Binário de 6 Bits

```
entity Contup is
    port (
        clock, reset, load, enable: in std_logic;
        bus: in std_logic_vector(5 downto 0);
        upCountOut: out std_logic_vector(5 downto 0)
    );
end contup;
architecture RTL of Contup is
    signal upCount: std_logic_vector(5 downto 0);
begin
    upCountOut <= upCount;
    process(clock, reset)
    begin
        if reset = '1' then
            upCount <= "000000";
        elsif clock'event and clock = '1' then
            if enable = '1' then
                if load = '1' then
                    upCount <= bus;
                else
                    upCount <= upCount + 1;
                end if;
            end if;
        end if;
    end process;
end RTL;
```

(1) Determine o comportamento deste contador, fazendo um diagrama de tempos

(2) O reset é prioritário em relação ao clock? Por quê?

(3) Como modificar o contador para realizar contagem crescente/decrescente?

Exemplo de um Contador Binário de 6 Bits

1. Faça a entidade do contador
2. Faça a implementação em diagrama de blocos da arquitetura do contador
3. Determine a prioridade entre todos os sinais de controle (clock, reset, load, enable)
4. O reset é síncrono ou assíncrono? E os demais sinais de controle?
5. Determine o comportamento deste contador, fazendo um diagrama de tempos, que deve conter algumas combinações dos sinais de controle
6. Modifique a arquitetura para que a mesma tenha um reset síncrono com contagem sensível a borda de descida
7. Como modificar o contador (sinais de controle e nova arquitetura) para realizar contagem crescente/decrescente?

Contador Gray

- **Código gray: seqüência onde a mudança de estado implica apenas a variação de um bit**

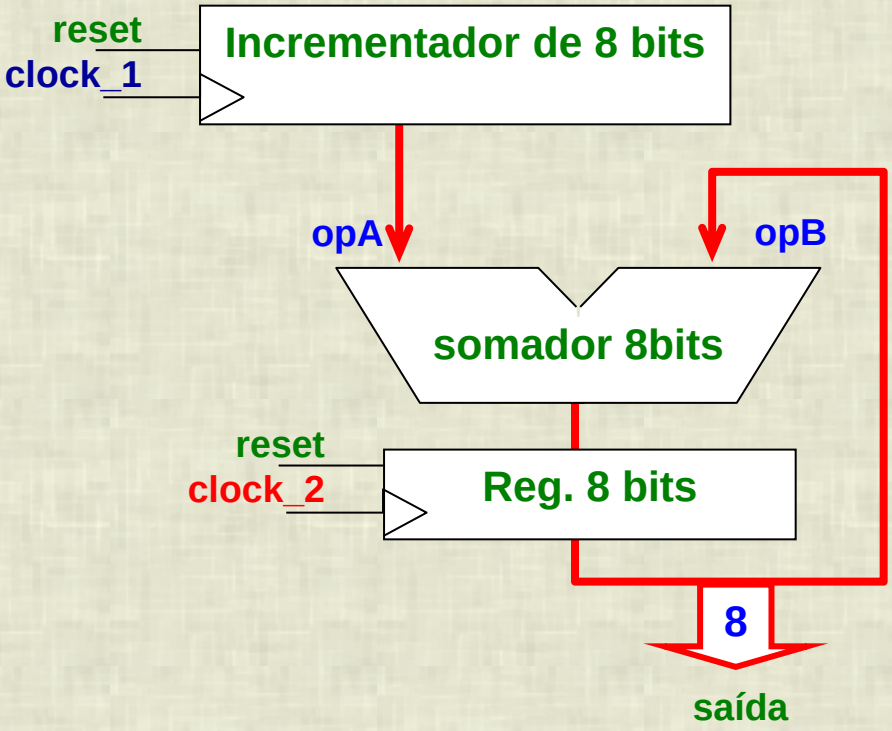
  000 → 001 → 011 → 010 → 110 → 111 → 101 → 100 → 000 → ...

- **Implementação pode ser feita por uma tabela de transição de estados**

```
architecture RTL of graycounter is
        signal clock, reset: std_logic;
        signal graycnt: std_logic_vector(2 downto 0);
begin
        process(clock, reset)
        begin
                if reset = '1'  then
                        graycnt <= "000";-- reset assíncrono
                elsif clock'event and clock = '1' then
                        case graycnt is
                                when "000"  => graycnt <= "001";
                                when "001"  => graycnt <= "011";
                                when "010"  => graycnt <= "110";
                                when "011"  => graycnt <= "010";
                                when "100"  => graycnt <= "000";
                                when "101"  => graycnt <= "100";
                                when "110"  => graycnt <= "111";
                                when "111"  => graycnt <= "101";
                                when others => null;
                        end case;
                end if;
    end process;
end RTL;
```

Exercícios

1. Descreva o circuito abaixo em VHDL (entidade e arquitetura)
   - Considere que na inicialização, o incrementador começa com 0 e o registrador começa com 2
   - Note que os clocks são diferentes. Estes devem ser implementados com a mesma freqüência e fazes invertidas



Exercícios

2. Extraia um diagrama de blocos representativo para o VHDL abaixo

```
entity Blocos is
    port (
        inicio, ck: in std_logic;
        valorCarga: in std_logic_vector(3 downto 0);
        valorOut: out std_logic_vector(1 downto 0);
        sh: out std_logic
    );
end Blocos;

architecture Blocos of Blocos is
    signal reg1, reg2: std_logic_vector(3 downto 0);
    signal r3: std_logic_vector(1 downto 0);
    signal n1, n2, n3: std_logic;
begin
    Reg_1: process(inicio)
    begin
        if inicio = '1' then
            reg1 <= valorCarga;
        end if;
    end process;
    Reg_2: process(inicio, ck)
    begin
        if inicio = '1' then
            reg2 <= valorCarga;
        elsif ck'event and ck = '1' then
            reg2 <= '0' & reg2(3 downto 1);
        end if;
    end process;
    n1 <= reg1(1) xor reg1(0);
    n2 <= reg1(3) and reg1(2);
    n3 <= n1 nand n2;
    r3 <= reg2(0) & n3;
    sh <= reg2(0);
    valorOut <= r3;
end Blocos;
```

Exercícios

3. Faça o projeto de um relógio de segundos e minutos

- O relógio tem como portas de entrada:
  - Ck (um bit) – referência de tempo base com freqüência de 25Hz
  - load (um bit) - permite ajustar a hora para um valor inicial que está definido das portas de entrada segIn e minIn
  - segIn e minIn (vetores de 6 bits) – contém valores iniciais de segundos e minutos, respectivamente
  - reset (um bit) – quando em 1 faz com que o relógio seja zerado. É assíncrono e precedente a ck
- O relógio tem como portas de saída:
  - seg e min (vetores de 6 bits) - contém os segundos e minutos atuais, respectivamente