UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA - CT

CIRCUITOS DIGITAIS

CONTADORES

DCA0212.1 - Laboratório 5

Igor Michael Araujo de Macedo
Isaac de Lyra Júnior
Pedro Henrique de Freitas Silva
Vinícius Silva do Carmo

Natal, 28 de janeiro de 2022

1. INTRODUÇÃO

O presente relatório tem por objetivo desenvolver os conceitos teóricos estudados sobre circuitos digitais de forma prática, mais especificamente sobre circuitos contadores síncronos. O desenvolvimento se dará utilizando VHDL, linguagem de descrição de hardware, para descrever contadores crescentes e decrescentes de forma comportamental. Por fim, será utilizado o software ModelSim para fazer as simulações necessárias.

2. DESENVOLVIMENTO

Para construir as lógicas que implementam os contadores propostos neste trabalho, se faz necessário, primeiramente, conhecê-los. Existem dois tipos de contadores: os síncronos e os assíncronos; no primeiro tipo, a mudança na contagem se dá a cada pulso de *clock*, já o segundo não depende dele. Neste trabalho serão desenvolvidos do tipo síncrono. Em seguida, é necessário entender o funcionamento dos contadores, que como o próprio nome sugere, é um circuito que realiza contagens crescentes e/ou decrescentes, a partir de determinados valores pré-definidos ou não, e que podem possuir comandos como o de habilitar contagem, reiniciar e de resetar.

O desenvolvimento do trabalho será dividido em três partes: a primeira será implementar um contador crescente a partir do código VHDL, em descrição comportamental, disponibilizado pelos professores da disciplina; e na segunda e terceira parte serão desenvolvidos os contadores de 4 bits crescente e decrescente, respectivamente, que possuem uma saída para indicar o término da contagem.

2.1 Contador Crescente Comportamental

Primeiro foi pedido para que fosse implementado um contador crescente comportamental, cujo código VHDL foi dado na atividade e está exposto na Figura 1. O contador possui 3 entradas do tipo *BIT*, sendo uma de *clock*, uma de *load* e uma de *reset*, também conta com uma entrada do tipo *INTEGER* de entrada de dados que pode variar de 15 até 0, possuindo como saída apenas a saída de dados do tipo *INTEGER* que também pode variar de 15 até 0.

Figura 1 - Código do Contador Comportamental

```
entity CONTADOR is
port(
    clk: in bit; --entrada de clock
    ld: in bit; --carrega os dados
    reset: in bit; --reiniciar
    data: in integer range 15 downto 0; -- entrada de dados
    q: out integer range 15 downto 0); --saída de dados
end CONTADOR;

architecture logic of CONTADOR is
    begin process(clk,reset)
        variable qv: integer range 15 downto 0; --variável para a saída

        begin
        if(reset = '1') then --reinicia a contagem
            qv := 0;
        elsif(clk ' event and clk = '1') then
            if(ld = '1') then --ld nível alto, saída será igual a data
                qv := data;
            else
                if(qv >= 9) then --contar no máximo até 9
                    qv := 0;
                else --incrementar 1 unidade
                    qv := qv + 1;
                end if;
            end if;
        end if;
        q <= qv;
    end process;
end;
```

Fonte: Autores.

2.2 Contador Crescente de 4 bits

Para a implementação do contador crescente de 4 bits foi utilizado basicamente o mesmo arranjo de entradas e saídas da entidade implementada anteriormente, com a adição apenas da saída “tc”. Tal saída deve ir para o valor 1 quando a contagem chegar no valor final da contagem, ou seja, no valor 15 ou “1111” em binário, permanecendo com o valor 0 enquanto o limite não for atingido, além disso o contador deve resetar sua contagem sempre que chegar no valor final. O código de implementação em VHDL pode ser visto na Figura 2.

Figura 2 - Código do Contador Crescente de 4 bits

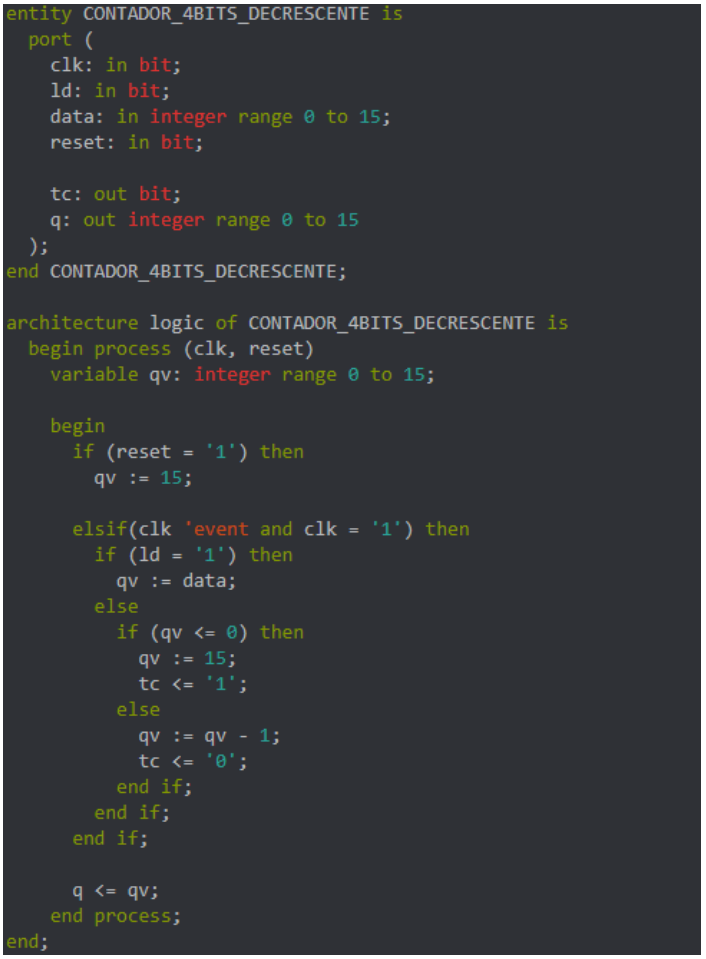
```
entity CONTADOR_4BITS_CRESCENTE is
  port (
    clk: in bit;
    ld: in bit;
    data: in integer range 0 to 15;
    reset: in bit;

    tc: out bit;
    q: out integer range 0 to 15
  );
end CONTADOR_4BITS_CRESCENTE;

architecture logic of CONTADOR_4BITS_CRESCENTE is
  begin process (clk, reset)
    variable qv: integer range 0 to 15;

    begin
      if (reset = '1') then
        qv := 0;

      elsif(clk 'event and clk = '1') then
        if (ld = '1') then
          qv := data;
        else
          if (qv >= 15) then
            qv := 0;
            tc <= '1';
          else
            qv := qv + 1;
            tc <= '0';
          end if;
        end if;
      end if;

      q <= qv;
    end process;
  end;
```

Fonte: Autores.

2.3 Contador Decrescente de 4 bits

Por fim, foi implementado um contador decrescente de 4 bits, onde deve ser capaz de contar de 15 até 0 ou de “1111” até “0000”, em binário. Também possuindo a saída “tc” que indica quando a contagem atingir o último valor da contagem, neste caso, o “0000”. O código da implementação em VHDL está na Figura 3.

Figura 3 - Código do Contador Decrescente de 4 bits



```
entity CONTADOR_4BITS_DECRESCENTE is
  port (
    clk: in bit;
    ld: in bit;
    data: in integer range 0 to 15;
    reset: in bit;

    tc: out bit;
    q: out integer range 0 to 15
  );
end CONTADOR_4BITS_DECRESCENTE;

architecture logic of CONTADOR_4BITS_DECRESCENTE is
  begin process (clk, reset)
    variable qv: integer range 0 to 15;

    begin
      if (reset = '1') then
        qv := 15;

      elsif(clk 'event and clk = '1') then
        if (ld = '1') then
          qv := data;
        else
          if (qv <= 0) then
            qv := 15;
            tc <= '1';
          else
            qv := qv - 1;
            tc <= '0';
          end if;
        end if;
      end if;

      q <= qv;
    end process;
end;
```

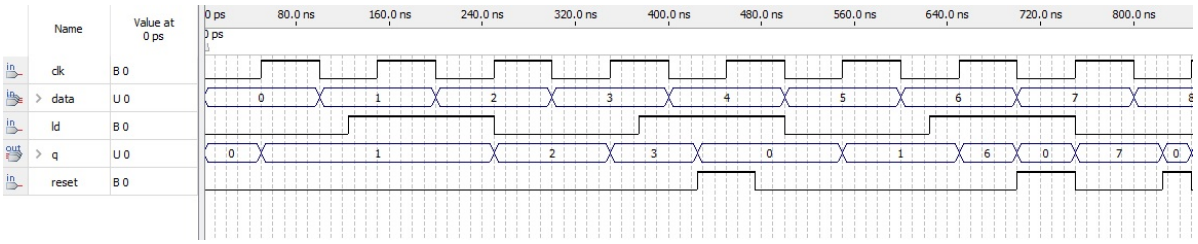
Fonte: Autores.

3. RESULTADOS

Os resultados obtidos neste trabalho foram produzidos no software ModelSim, onde neste pudemos tanto implementar os códigos de contadores desenvolvidos quanto simular o funcionamento destes para averiguar se a implementação foi bem sucedida. Os contadores seguiram todos o mesmo padrão, mudando apenas a capacidade e o sentido de contagem. Assim, todos eles contam com pulsos oscilatórios, uma chave para habilitar o carregamento de dados e uma chave para resetar a contagem.

Primeiramente desenvolvemos a estrutura do circuito contador crescente, onde este tem a capacidade de contar de 0 até 9, e após isso retornar ao zero e recomeçar o processo. O código fonte da implementação deste circuito em VHDL pode ser visto no Anexo A. Para verificar o correto funcionamento deste circuito, foram simuladas algumas ondas apresentando as entradas e sendo nos devolvidas às respectivas saídas. Esta simulação pode ser vista na Figura 4.

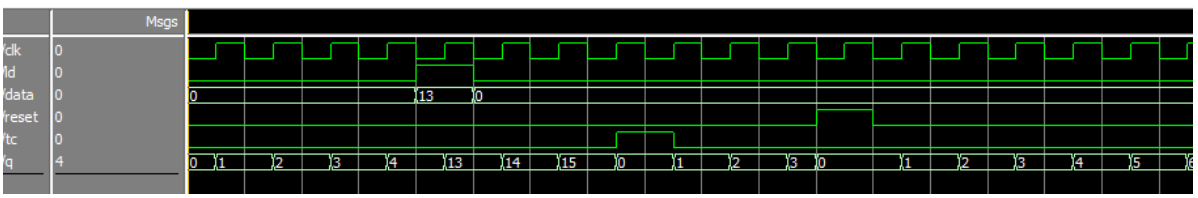
Figura 4 - Simulação do funcionamento do circuito contador crescente de 0 a 9.



Fonte: Autores.

Para testar o contador crescente de 4 bits, o teste realizado consistiu em habilitar o clock e observar a contagem crescente começando do valor 0, após o quarto ciclo de clock foi adicionado o valor 13 na entrada de dados e foi alterado o valor lógico da entrada de load, desta forma o valor 13 foi carregado no contador, que continuou sua contagem até atingir o limite e reiniciar sua contagem. Repare que a saída “tc” foi para nível lógico alto durante um ciclo de clock quando o contador atingiu seu limite. Na Figura 5 é possível observar o resultado do teste realizado.

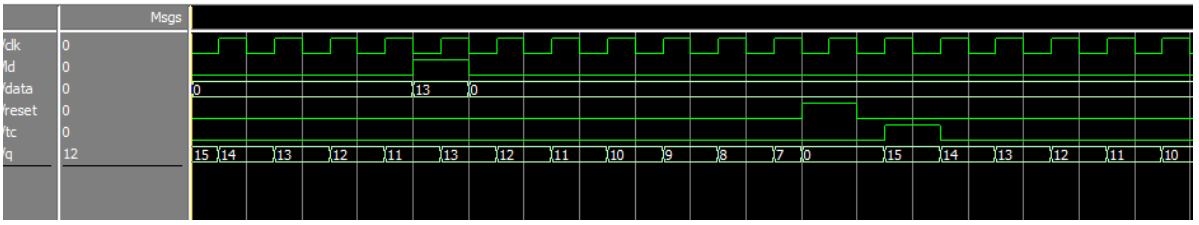
Figura 5 - Simulação do funcionamento do circuito contador crescente de 0 a 15.



Fonte: Autores.

De forma semelhante ao teste anterior, foi habilitado o clock e observado o comportamento do contador decrescente começando do valor 15 e continuando sua contagem até o valor 11, onde a entrada de load foi para nível lógico alto e carregado o valor 13 no contador, que continuou sua contagem a partir deste valor, ao chegar no valor 7, a entrada de reset foi habilitada, zerando o contador, que reiniciou sua contagem em seguida. É importante observar que a saída “tc” também ficou em nível lógico alto quando o contador atingiu o seu limite, que neste caso foi o 0. Na Figura 6 é possível observar o resultado do teste realizado.

Figura 6 - Simulação do funcionamento do circuito contador decrescente de 15 a 0.



Fonte: Autores

4. CONCLUSÃO

O presente trabalho teve como objetivo pôr em prática os conhecimentos estudados sobre contadores. Para isso, foram implementados e simulados circuitos de contadores crescentes e decrescentes de 4 bits. Para esta solução construímos três entidades utilizando a linguagem VHD, os quais o código-fonte completo pode ser visto no Anexo A, sendo duas contadores crescentes e outra referente ao contador decrescente. Todas as entidades foram testadas e simuladas utilizando o software Modelsim. Após todo o processo, a simulação retornou os resultados esperados e, com isso, podemos concluir que foi possível extrair o máximo de conhecimento e proveito do problema.

ANEXO A - CÓDIGO-FONTE DOS CONTADORES

```
entity CONTADOR is
port(
    clk: in bit; --entrada de clock
    ld: in bit; --carrega os dados
    reset: in bit; --reiniciar
    data: in integer range 15 downto 0; -- entrada de dados
    q: out integer range 15 downto 0); --saída de dados
end CONTADOR;

architecture logic of CONTADOR is
    begin process(clk,reset)
          variable qv: integer range 15 downto 0; --variável
para a saida

        begin
        if(reset = '1') then --reinicia a contagem
            qv := 0;
        elsif(clk ' event and clk = '1') then
               if(ld = '1') then --ld nível alto, saída será
igual a data
                qv := data;
            else
                if(qv >= 9) then --contar no máximo até 9
                    qv := 0;
                else --incrementar 1 unidade
                    qv := qv + 1;
                end if;
            end if;
        end if;
        q <= qv;
    end process;
end;

entity CONTADOR_4BITS_CRESCENTE is
  port (
    clk: in bit;
    ld: in bit;
    data: in integer range 0 to 15;
    reset: in bit;

    tc: out bit;
    q: out integer range 0 to 15
  );
end CONTADOR_4BITS_CRESCENTE;

architecture logic of CONTADOR_4BITS_CRESCENTE is
  begin process (clk, reset)
    variable qv: integer range 0 to 15;
```

```
    begin
      if (reset = '1') then
        qv := 0;

      elsif(clk 'event and clk = '1') then
        if (ld = '1') then
          qv := data;
        else
          if (qv >= 15) then
            qv := 0;
            tc <= '1';
          else
            qv := qv + 1;
            tc <= '0';
          end if;
        end if;
      end if;

      q <= qv;
    end process;
  end;

entity CONTADOR_4BITS_DECRESCENTE is
  port (
    clk: in bit;
    ld: in bit;
    data: in integer range 0 to 15;
    reset: in bit;

    tc: out bit;
    q: out integer range 0 to 15
  );
end CONTADOR_4BITS_DECRESCENTE;

architecture logic of CONTADOR_4BITS_DECRESCENTE is
  begin process (clk, reset)
    variable qv: integer range 0 to 15;

    begin
      if (reset = '1') then
        qv := 15;

      elsif(clk 'event and clk = '1') then
        if (ld = '1') then
          qv := data;
        else
          if (qv <= 0) then
            qv := 15;
            tc <= '1';
          else
```

```
            qv := qv - 1;
            tc <= '0';
          end if;
        end if;
      end if;

      q <= qv;
    end process;
end;
```