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

CIRCUITOS DIGITAIS

REGISTRADORES

DCA0212.1 - Laboratório 4

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

Natal, 14 de janeiro de 2022

1. INTRODUÇÃO

O presente relatório tem por objetivo desenvolver os conceitos teóricos estudados sobre circuitos digitais de forma prática. Serão abordados conceitos de circuitos sequenciais, mais especificamente os elementos armazenadores de dados flip-flop tipo D e latch tipo D. O desenvolvimento se dará utilizando VHD, linguagem de descrição de *hardware*, para descrever cada um desses elementos; em seguida, serão desenvolvidos dois registradores de 4 bits, cada um utilizando um dos tipos. Por fim, será utilizado o *software* ModelSim para fazer as simulações necessárias.

2. DESENVOLVIMENTO

Para construir as lógicas que implementam os registradores propostos neste trabalho, foi necessário primeiramente entender como tais registradores funcionam. Sabendo que este tipo de circuito é sequencial, necessita-se da atenção de implementar os circuitos baseando-se em uma estrutura ordenada, tendo em vista que estes circuitos têm funcionamento assíncrono.

2.1 Latch D

O circuito sequencial latch D possui uma entrada D, que representa o dado armazenado, uma entrada habilitadora EN e uma saída Q. A saída do circuito se dá pelo *enable* EN, quando com nível lógico alto, a saída passa a ter o valor de D; quando com nível lógico baixo, a saída estabiliza com o valor que a entrada D possuía no momento exatamente antes do EN passar de alto para baixo. Para isso, foi utilizado o operador condicional IF. O código-fonte deste componente pode ser visto na Figura 1.

Figura 1 - Código VHDL do latch D

```
ENTITY latch_D IS
    PORT (
      D: IN  bit;
      EN: IN  bit;
      Q: OUT bit
    );
END latch_D;

ARCHITECTURE logic OF latch_D IS

BEGIN

PROCESS(D, EN)
  BEGIN
    IF (EN = '1') THEN
        Q <= D;
    END IF;
END PROCESS;

END logic;
```

Fonte: Autores.

2.2 Registrador de 4 bits baseado em Latch D

A implementação deste tipo de registrador é totalmente baseada na implementação do Latch D, que armazena apenas um bit. Sabendo disto e tendo o circuito Latch D de um bit já implementado, para implementar o registrador de 4 bits baseado em Latch D basta apenas concatenar 4 Latch's D de um bit. Feito isso, é necessário apenas configurar as entradas e saídas de cada um dos circuitos latch D. O código de implementação deste circuito pode ser visto na Figura 2.

Figura 2: Código VHDL do registrador de 4 bits baseado em Latch D

```
------- latch D 4 bits ----------

ENTITY latch_D_4BITS IS
    PORT (
      D3, D2, D1, D0: IN  bit;
      EN: IN  bit;
      Q3, Q2, Q1, Q0: OUT bit
    );
END latch_D_4BITS;

ARCHITECTURE logic OF latch_D_4BITS IS

  COMPONENT latch_D IS
    PORT (
      D: IN  bit;
      EN: IN  bit;
      Q: OUT bit
    );
  END COMPONENT;

BEGIN

  S3: latch_D port map(D3, EN, Q3);
  S2: latch_D port map(D2, EN, Q2);
  S1: latch_D port map(D1, EN, Q1);
  S0: latch_D port map(D0, EN, Q0);

END logic;
```

Fonte: Autores

2.3 Flip-flop D

Para a construção do componente do flip-flop D foi utilizado as notas de aula para estudar o comportamento esperado do componente, sabendo que o componente deve ser

capaz de armazenar o dado de entrada a cada *rising edge* do clock foi possível a implementação do componente em código VHDL, onde pode ser visto na Figura 3.

Figura 3 - Código VHDL do flip-flop D

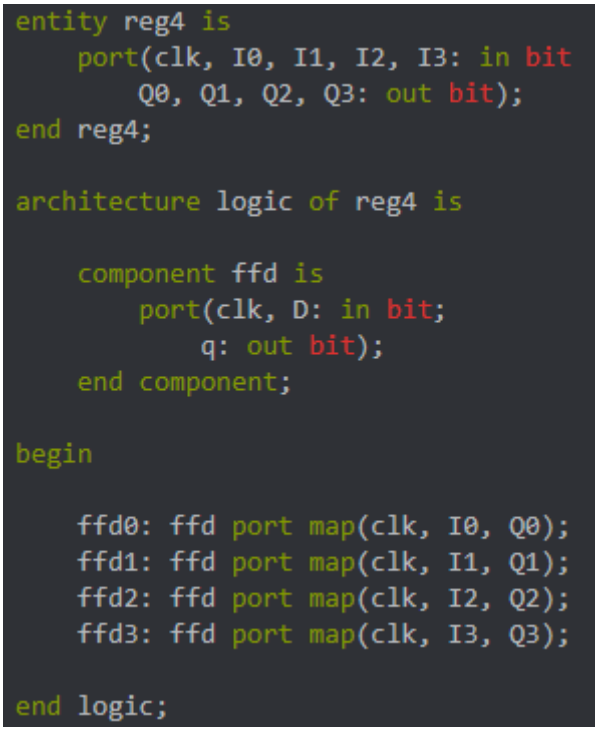
```
entity ffd is
    port(clk, D: in bit;
         q: out bit);
end ffd;

architecture logic of ffd is

    signal qS: bit;

begin
process(clk)
begin
    if clk = '1' and clk'event then
        qS <= D;
    end if;
end process;
q<=qS;
end logic;
```

Fonte: Autores.

2.4 Registrador de 4 bits baseado em Flip-flop D

O registrador de 4 bits foi implementado utilizando o flip-flop D já implementado como componente, tal componente tem como inputs os 4 bits que se deseja armazenar e o clock do sistema e em sua saída possui os 4 bits que estão alocados no componente no momento. O resultado da implementação em linguagem VHDl do registrador baseado em flip-flop D pode ser visto na Figura 4.

Figura 4 - Código VHDL do registrador de 4 bits baseado em flip-flop D



```
entity reg4 is
    port(clk, I0, I1, I2, I3: in bit
        Q0, Q1, Q2, Q3: out bit);
end reg4;

architecture logic of reg4 is

    component ffd is
        port(clk, D: in bit;
            q: out bit);
    end component;

begin

    ffd0: ffd port map(clk, I0, Q0);
    ffd1: ffd port map(clk, I1, Q1);
    ffd2: ffd port map(clk, I2, Q2);
    ffd3: ffd port map(clk, I3, Q3);

end logic;
```

Fonte: Autores.

3. RESULTADOS

Para simular os códigos implementados foi utilizado o *software* ModelSim. Para isso ser possível foi criado um arquivo de extensão ".do" para cada um dos componentes implementados para testar o seu funcionamento. Os códigos dos arquivos de simulação, bem como os resultados de cada entidade implementada, podem ser vistos abaixo.

3.1 Latch D

Para a simulação do Latch D, foi utilizado o arquivo de simulação da Figura 5, o qual força valores para EN e D de tal forma que teste todas as combinações possíveis (no caso $2^2 = 4$ casos).

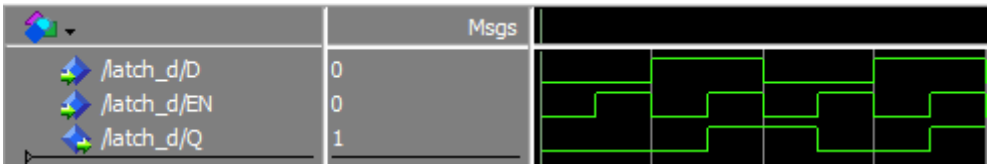
Figura 5: Arquivo de simulação do circuito sequencial Latch D

```
vsim latch_D

add wave *

force EN 0 0, 1 25 -repeat 50
force D 0 0, 1 50 -repeat 100

run 200
```

Fonte: Autores

Como pode ser visto na Figura 6, o funcionamento está de acordo com o que foi explicado na seção 2.1. Por exemplo, quando $D = 1$ (nível lógico baixo) e $EN = 0$, Q continua com nível baixo, apenas quando $D = 1$ e $EN = 1$ que $Q = 1$.

Figura 6: Simulação do circuito sequencial Latch D.



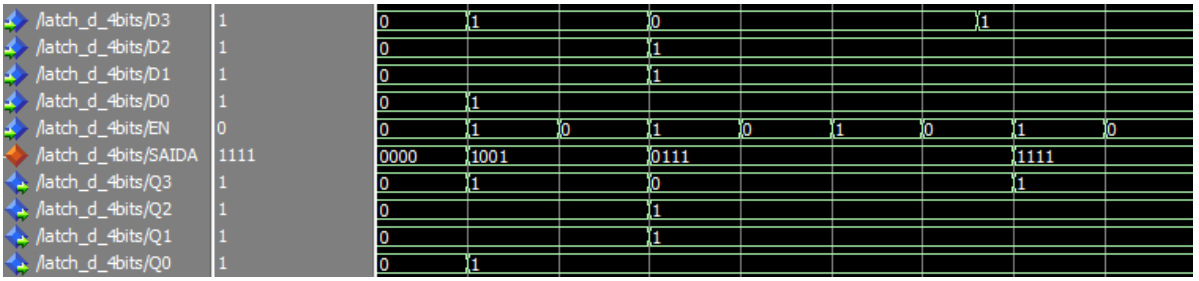
Fonte: Autores

3.2 Registrador baseado em Latch D

A simulação deste circuito é realizada a partir de diferentes entradas e ao mesmo tempo que são trocadas as entradas fazemos a entrada que habilita o registro dos dados (EN). No primeiro momento todas as entradas estão zeradas e a habilitação do circuito também, e a partir disso são aplicadas entradas "válidas" e de acordo com a entrada EN

podemos ver que ele registra corretamente todas as entradas que são colocadas. Esta explicação pode ser melhor observada na Figura 7.

Figura 7: Simulação do circuito registrador de 4 bits baseado em Latch D



Fonte: Autores

Figura 8: Arquivo de simulação do circuito registrador de 4 bits baseado em Latch D

```
vsim latch_D_4BITS

add wave *

force D3  0 0, 1 50, 0 150, 1 330;
force D2  0 0, 0 50, 1 150, 1 330;
force D1  0 0, 0 50, 1 150, 1 330;
force D0  0 0, 1 50, 1 150, 1 330;

force EN 0 0, 1 50 -repeat 100;

run 1500;
```

Fonte: Autores

3.3 Flip-flop D

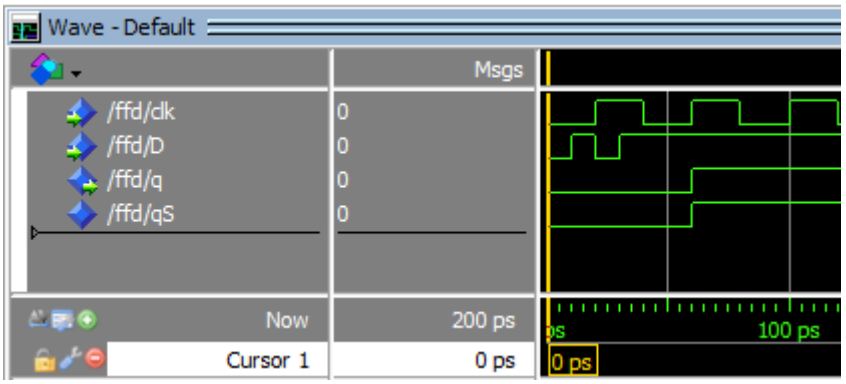
Para simular o Flip-flop D de 1 bit a entrada foi alternada entre 0 e 1 enquanto o clock estava em nível lógico baixo e alto, para testar se a saída também iria alternar, o que não faria sentido, já que o Flip-flop D só alterna no *rising edge* do clock, foi verificado que o componente estava funcionando corretamente quando a saída só alternou quando a entrada estava em nível lógico alto e o clock estava no *rising edge*. O código responsável por simular o Flip-flop D, bem como a sua simulação pode ser visto nas Figuras 9 e 10.

Figura 9 - Conteúdo do arquivo .do responsável por simular o Flip-flop D

```
vsim ffd

add wave *

force D 0 0, 1 10, 0 20, 1 30
force clk 0 0, 1 20 -repeat 40

run 200
```

Fonte: Autores

Figura 10 - Simulação do Flip-flop D



Fonte: Autores

3.4 Registrador de 4 bits baseado em Flip-flop D

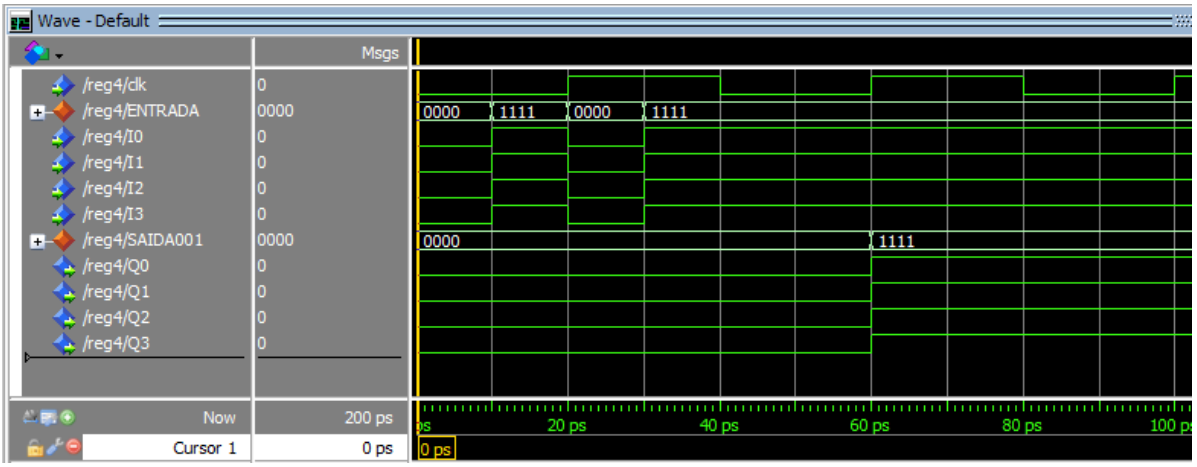
Similarmente a simulação do Flip-flop D, para testar o registrador de 4 bits baseado em Flip-Flop D, foi introduzida em sua entrada os valores “0000” e “1111” durante o período do clock em nível lógico baixo e alto, onde não foi alterado o valor de saída. O valor só foi alterado quando o *clock* estava no *rising edge* e o valor armazenado foi de “1111” que estava em sua entrada. As Figuras 11 e 12 ilustram o conteúdo do código responsável pela simulação do registrador e também o resultado da simulação.

Figura 11 - Conteúdo do arquivo .do responsável por simular o Registrador de 4 bits baseado em Flip-flop D

```
vsim reg4

add wave *

force I0 0 0, 1 10, 0 20, 1 30
force I1 0 0, 1 10, 0 20, 1 30
force I2 0 0, 1 10, 0 20, 1 30
force I3 0 0, 1 10, 0 20, 1 30
force clk 0 0, 1 20 -repeat 40

run 200
```

Fonte: Autores

Figura 12 - Simulação do Registrador de 4 bits baseado em Flip-flop D



Fonte: Autores

4. CONCLUSÃO

O presente trabalho teve como objetivo pôr em prática os conhecimentos estudados sobre Latch's e flip-flops, bem como um dos componentes principais dos circuitos digitais, os registradores. Para isso, foram implementados e simulados circuitos Latch D, Flip-flop D e registradores de 4 bits baseados nos componentes implementados anteriormente. Para esta solução construímos quatro entidades utilizando a linguagem VHD, os quais o código-fonte completo pode ser visto no Anexo A, duas referentes ao Latch D e o registrador de 4 bits baseado em Latch D e as outras duas referentes ao Flip-flop D e o registrador de 4 bits baseado nele. 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

```
LIBRARY ieee;
USE ieee . std_logic_1164 . ALL;

--=============================================================---
                  -- L A T C H    D
--=============================================================---
ENTITY latch_D IS
    PORT (
      D: IN  bit;
      EN: IN  bit;
      Q: OUT bit
    );
END latch_D;

ARCHITECTURE logic OF latch_D IS

BEGIN

PROCESS(D, EN)
  BEGIN
    IF (EN = '1') THEN
        Q <= D;
    END IF;
END PROCESS;

END logic;

--=============================================================---
-- R E G I S T R A D O R  D E  4  B I T S - L A T C H  D
--=============================================================---
ENTITY latch_D_4BITS IS
    PORT (
      D3, D2, D1, D0: IN  bit;
      EN: IN  bit;
      Q3, Q2, Q1, Q0: OUT bit
    );
END latch_D_4BITS;

ARCHITECTURE logic OF latch_D_4BITS IS

  COMPONENT latch_D IS
    PORT (
      D: IN  bit;
      EN: IN  bit;
      Q: OUT bit
    );
  END COMPONENT;

SIGNAL Qaux3, Qaux2, Qaux1, Qaux0 : bit;
```

```
BEGIN

  S3: latch_D port map(D3, EN, Q3);
  S2: latch_D port map(D2, EN, Q2);
  S1: latch_D port map(D1, EN, Q1);
  S0: latch_D port map(D0, EN, Q0);

END logic;

--=============================================================
              -- F L I P - F L O P   D
--=============================================================
entity ffd is
  port(clk, D: in bit;
       q: out bit);
end ffd;

architecture logic of ffd is

  signal qS: bit;

begin
process(clk)
begin
  if clk = '1' and clk'event then
       qS <= D;
  end if;
end process;
q<=qS;
end logic;

--=============================================================
-- R E G I S T R A D O R  D E  4  B I T S - FF  D
--=============================================================
entity reg4 is
  port(clk, I0, I1, I2, I3: in bit;
       Q0, Q1, Q2, Q3: out bit);
end reg4;

architecture logic of reg4 is

  component ffd is
      port(clk, D: in bit;
           q: out bit);
  end component;

begin

  ffd0: ffd port map(clk, I0, Q0);
  ffd1: ffd port map(clk, I1, Q1);
```

```
  ffd2: ffd port map(clk, I2, Q2);
  ffd3: ffd port map(clk, I3, Q3);

end logic;
```