Universidade Federal do Rio Grande do Norte

Disciplina - Circuitos Digitais - ELE2715

Professor - Samaherni Morais Dias

Aluno(a) - Erika Costa Alves Matrícula - 2016019571

Relatório do Laboratório 06

9 de setembro de 2019

1. Introdução

Este relatório se refere à atividade prática de laboratório da disciplina de Circuitos Digitais. Esta atividade tem como objetivo de projetar um circuito em VHDL, na qual tem uma entrada de 4 bits, e essa entrada vai entrar paralelamente e os dados inseridos paralelamente serão transmitidos serialmente. E a saída será os valores definidos pela entrada de 4 bits. Por fim, o circuito deve ter entradas assíncronas para dar clear nos valores. Observe a **Figura 1** para observar melhor o que será feito.

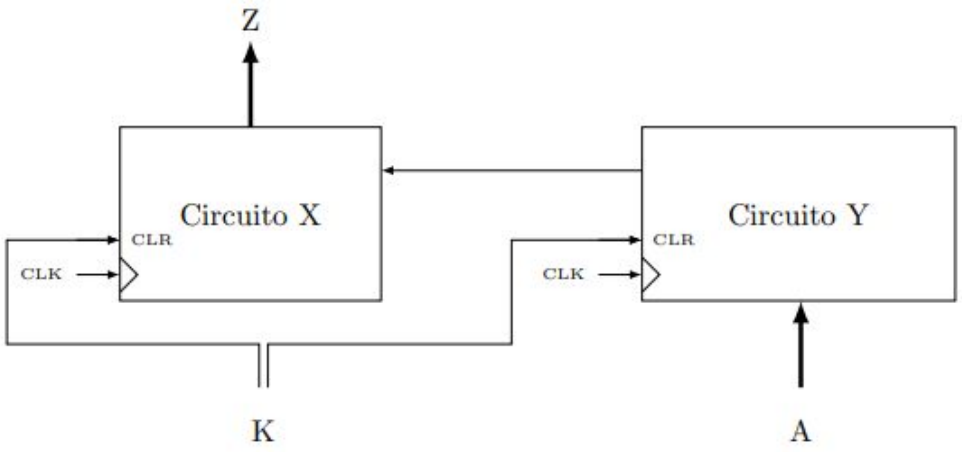

Figura 1 - Circuito proposto.

2. Desenvolvimento

Como queremos introduzir um valor paralelamente e depois serialmente, utilizaremos o conhecimento sobre armazenamento de dados utilizando componentes chamados de Flip Flop.

Primeiramente será resolvido o problema do Circuito Y, que é o circuito que vai introduzir os valores paralelamente. Para resolveremos isso é preciso ter utilizar as entradas assíncronas, em específico será utilizada a entrada *set* do Flip Flop, e logo na entrada será utilizado uma porta *nand* , pois, será utilizado uma entrada *enable* para que toda vez que ela estiver em alto o valor que queremos que seja armazenado no flip flop será colocado. Como são quatro flip flops, então o processo anterior será feito para os quatro flip flops, e além disso as saídas de cada flip flop será colocada na entrada de cada flip flop, com exceção do último flip flop que será ligado no Circuito X. Logo a cada pulso de clock os valores armazenados nos flip flops serão passados para a direita.

Vale lembrar que o uso do *nand* é por que as entradas assíncronas do flip flop são barradas. E, também, que o uso das entradas assíncronos foi essencial por que essas entradas forçam os flip flops irem para 0 ou 1 (dependendo da entrada assíncrona que for utilizar) independente do valor que estiver na entrada. A entrada *reset* leva o flip flop a 0, e a entrada entrada *set* leva o flip flop a 1. Nesse circuito será utilizado a entrada *reset* para dar um ‘clear’ nos valores tanto do circuito X como do circuito Y.

Segundamente para fazer o circuito X é mais simples, pois, é apenas um registrador de 4 bits que transfere os valores serialmente. Então para fazer isso basta ligar as saídas de cada flip flop na entrada do próximo flip flop, sendo a entrada do primeiro flip flop a saída do circuito Y. Assim, a cada pulso de clock o valor será passado pro próximo flip flop até que cada flip flop do circuito X fique com o valor de entrada inicial A. A saída Z do circuito geral vai ser a saída de cada flip flop do circuito X, sendo o primeiro flip flop o do bit mais significativo, e o último flip flop o bit menos significativo.

3. Implementação

O código dos circuitos foi feito em VHDL utilizando o software ModelSim. Logo a baixo está todos os códigos utilizados para o projeto.

```
entity FFJK is
port(   clk, j, k, p, c : in bit;
        q               : out bit
);
end FFJK;

architecture ckt of FFJK is
        signal qs : bit;
begin

        process(clk, p, c)
        begin

                if p = '0' then qs <= '1';
                elsif c = '0' then qs <= '0';
                elsif clk = '1' and clk'event then
                        if j = '1' and k = '1' then qs <= not qs;
                        elsif j = '1' and k = '0' then qs <= '1';
                        elsif j = '0' and k = '1' then qs <= '0';
                        end if;
                end if;
        end process;

        q <= qs;

end ckt;
```

Figura 2 - Codigo para Flip Flop JK

```
entity CircuitoY is
port(   A       : in bit_vector(3 downto 0);
        e       : in bit;
        clock   : in bit;
        clear   : in bit;
        saida   : out bit

);
end CircuitoY;

architecture ckt of CircuitoY is
component FFJK is
port(   clk, j, k, p, c : in bit;
        q               : out bit
);
end component;

        signal aux      : bit_vector(3 downto 0);
        signal qs       : bit_vector(3 downto 0);
        signal qs2      : bit_vector(2 downto 0);

begin

        aux(0) <= not(e and A(0));
        aux(1) <= not(e and A(1));
        aux(2) <= not(e and A(2));
        aux(3) <= not(e and A(3));

        chinela0 : FFJK
        port map(clock, '0', '1', aux(0), clear, qs(0));

        qs2(0) <= not qs(0);

        chinela1 : FFJK
        port map(clock, qs(0), qs2(0), aux(1), clear, qs(1));

        qs2(1) <= not qs(1);

           chinela2 : FFJK
           port map(clock, qs(1), qs2(1), aux(2), clear, qs(2));

           qs2(2) <= not qs(2);

           chinela3 : FFJK
           port map(clock, qs(2), qs2(2), aux(3), clear, qs(3));

           saida <= qs(3);

end ckt;
```

Figura 3 - Codigo para Circuito Y

```
entity CircuitoX is
port(   entrada : in bit;
        clock   : in bit;
        clear   : in bit;
        Z       : out bit_vector(3 downto 0)
);
end CircuitoX;

architecture ckt of CircuitoX is
component FFJK is
port(   clk, j, k, p, c : in bit;
        q               : out bit
);
end component;

        signal aux      : bit_vector(3 downto 0);
        signal sig      : bit_vector(3 downto 0);

begin

        aux(0) <= not entrada;

        chinela0 : FFJK
        port map(clock, entrada, aux(0), '1', '1', sig(0));

        aux(1) <= not sig(0);

        chinela1 : FFJK
        port map(clock, sig(0), aux(1), '1', '1', sig(1));

        aux(2) <= not sig(1);

        chinela2 : FFJK
        port map(clock, sig(1), aux(2), '1', '1', sig(2));

        aux(3) <= not sig(2);

        chinela3 : FFJK
        port map(clock, sig(2), aux(3), '1', '1', sig(3));

        Z(0) <= sig(0);
        Z(1) <= sig(1);
        Z(2) <= sig(2);
        Z(3) <= sig(3);


end ckt;
```

Figura 4 - Codigo para Circuito X

```
entity lab06 is
port(   A       : in bit_vector(3 downto 0);
        K, clk  : in bit;
        enable  : in bit;
        Z       : out bit_vector(3 downto 0)
);
end lab06;

architecture ckt of lab06 is
component CircuitoY is
port(   A       : in bit_vector(3 downto 0);
        e       : in bit;
        clock   : in bit;
        clear   : in bit;
        saida   : out bit

);
end component;
component CircuitoX is
port(   entrada : in bit;
        clock   : in bit;
        clear   : in bit;
        Z       : out bit_vector(3 downto 0)
);
end component;

        signal conexao : bit;

begin

        Y       : CircuitoY
        port map(A(3 downto 0), enable, clk, K, conexao);

        X       : CircuitoX
        port map(conexao, clk, K, Z(3 downto 0));


end ckt;
```

Figura 5 - Código Final

4. Execução/Resultado

Após a implementação dos códigos em VHDL, foi feita a simulação usando o ModelSim. Foi-se feita a simulação dos circuitos a parte, e depois feita a simulação de todo o circuito junto.

Por fim, depois de projetar e fazer um estudo de como foi feito os circuitos é possível fazer uma análise sobre o quão útil é a utilização de armazenadores de memória e a forma como é passada as informações bit a bit. As formas de transmissão de informação digital é de grande importância, claro, com auxílio da frequência - o Clock - , pois, é utilizada até mesmo na área de telecomunicação e computação.

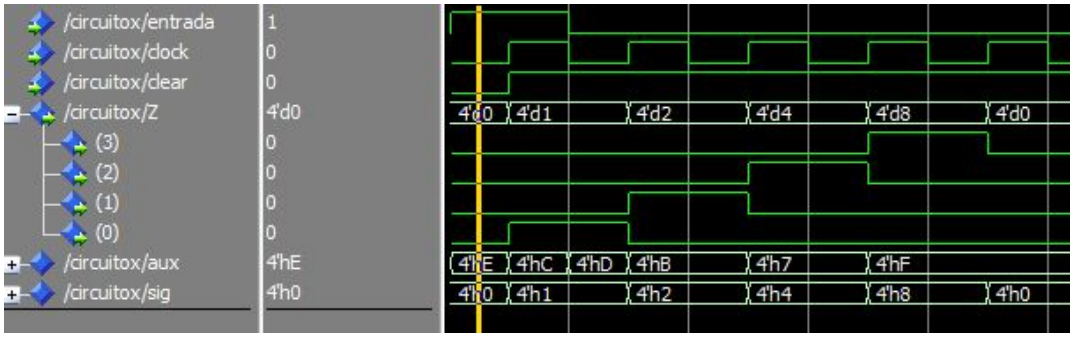

Figura 6 - Circuito X

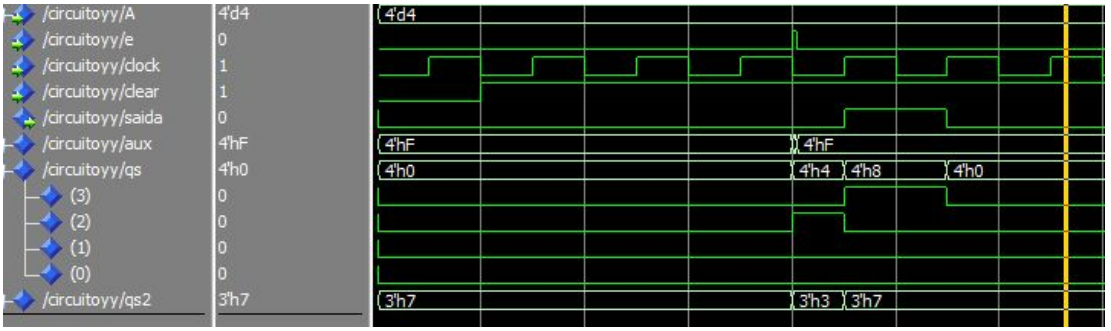

Figura 7 - Circuito Y

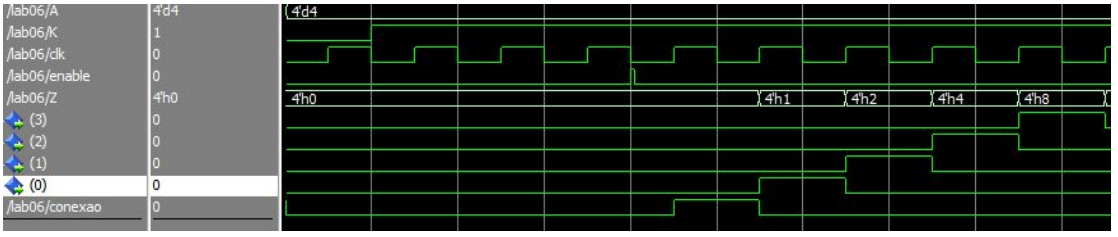

Figura 8 - Simulação Final