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 05

2 de setembro de 2019

01 Introdução.

Este relatório se refere à atividade prática de laboratório da disciplina de Circuito Digitais. Esta atividade tem como objetivo de projetar um circuito lógico, na qual, a saída é de 8 bits, em BCD, e a entrada de 7 bits, na qual o bit mais significativo é bit de sinal. Com isso o objetivo é fazer o módulo da entrada, e transformar em BCD. Ver figura abaixo (**Figura 01)**.

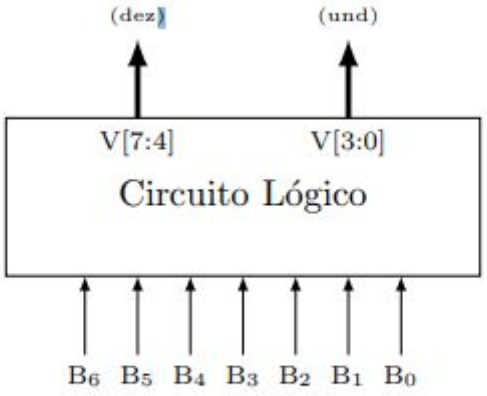


Figura 01 - Circuito Projetado.

02 Desenvolvimento.

Para desenvolver o Circuito Lógico, foi dividido ele em três partes diferentes, e com objetivos diferentes. A primeira parte seria selecionar a entrada dependendo se o bit de sinal estiver em alto ou não, caso esteja em alto o bit de sinal, a entrada será toda invertida. A segunda parte é a parte de soma, caso o bit de sinal estiver em alto, a entrada selecionada irá somar mais um, para fazer o complemento dois. E por fim, após a seleção da entrada e fazer a soma do complementar, haverá a transformação de binário para BCD.

2.1 Seleção da entrada.

Ao solucionar a questão da seleção das entradas, foi utilizado um componente bastante útil chamado de Multiplexador. O componente Multiplexador é utilizado para selecionar uma dependendo de qual seja a entrada do seletor, nesse projeto foi utilizado um multiplexador 2x1, ou seja, só tem duas entradas e uma saída, e o seletor quando estiver em alto irá escolher a segunda entrada.

Então, como há um seletor no multiplexador, esse seletor será exatamente o nosso bit de sinal, logo, quando o bit seletor estiver em baixo, ele irá escolher a entrada sem estar invertida, mas caso o bit seletor estiver em alto, ele irá escolher a entrada invertida.

2.2 Complementar dois.

Para resolver a questão do complementar dois é bem simples, basta criar um somador de seis bits, no qual a entrada A é a saída do nosso multiplexador, e a entrada B é “000000”, e para fazer a soma de 1 bit, basta colocar em alto o carry de entrada (Cin) do somador. Observe que o Cin do somador de seis bits é exatamente o bit de sinal, então, caso o bit de sinal esteja em baixo, não irá somar nada, só irá somar algo se estiver em alto. Note que se a entrada A do somador vier sem estar invertida, já que o seletor do multiplexador estava sem zero (bit de sinal), o Cin vai estar em zero, logo, não irá alterar o valor do número.

2.3 Binário para BCD.

Para solucionar essa terceira parte, foi utilizado o código do terceiro laboratório de circuito digitais.Esse laboratório em específico foi para implementar um código que transforma um número de 8 bits em uma saída BCD de 12 bits. Veja **figura 03.**

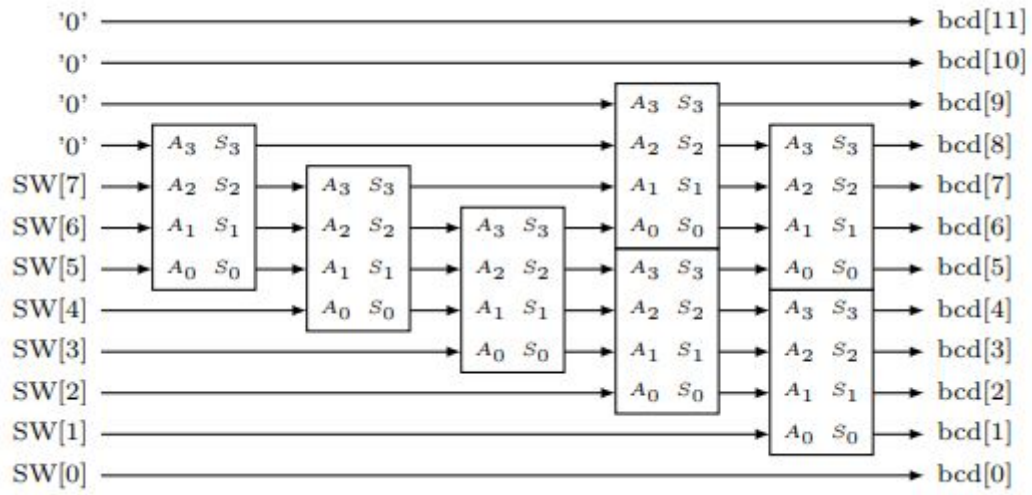
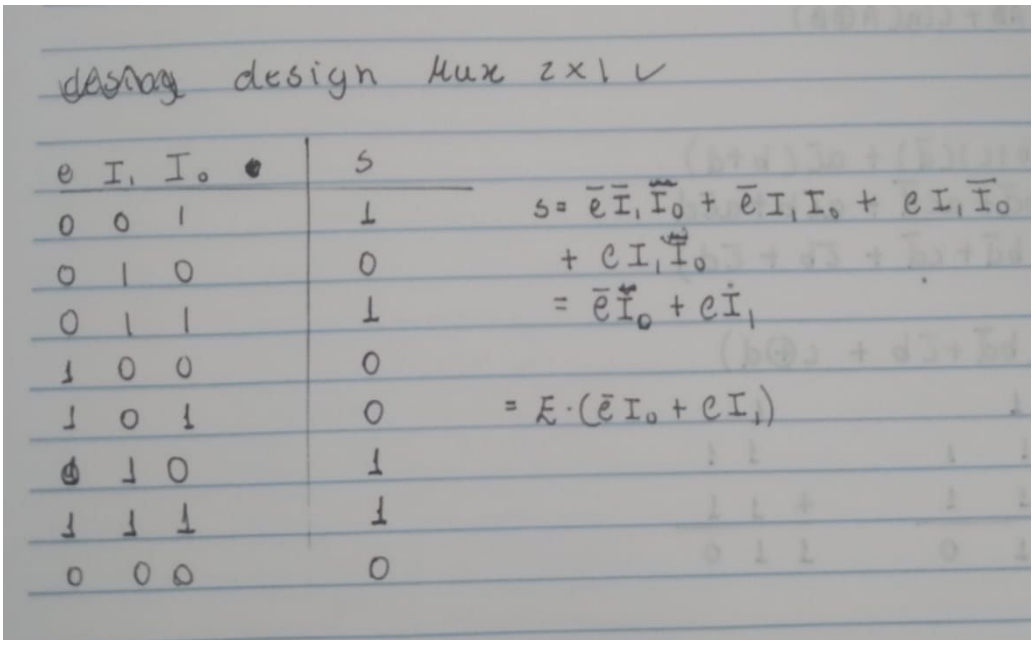


Figura 03 - Bin to BCD.

03 Implementação.

3.1 Multiplexador 2x1.

Com o objetivo de implementar um multiplexador 2x1, primeiramente foi feito uma tabela verdade que relacionasse as entradas com as saídas dependendo do seletor. Observe a **Figura 04** para ver a tabela verdade, e em seguida a **Figura 05** para observar o código implementado sobre o resultado da tabela verdade.



Note que na **Figura 05**, a entrada *enable* só serve para fazer funcionar o mux, ou seja, basta por em alto a entrada que irá funcionar, caso não esteja, a saída sempre será zero.

```
entity Mux2x1 is
port(   I0, I1  : in bit;
        s       : in bit;
        enable  : in bit;
        Y       : out bit
);
end Mux2x1;

architecture ckt of Mux2x1 is

        signal sig1, sig2       : bit;

begin

        sig1 <= (not s) and I0;
        sig2 <= s and I1;

        Y <= enable and(sig1 or sig2);

end ckt;
```

Figura 05 - Mux 2x1.

3.1 Somador 6 Bits.

Para fazer um somador de 6 bits basta apenas implementar um somador completo de 1 bit, e logo em seguida colocar o carry de saída como o carry de entrada do próximo somador completo de 1 bit, ou seja, é um circuito cascateado de somador completo de 1 bit. Veja a **Figura 06** para o somador completo e a **Figura 07** para o somador de 6 bits.

```
entity FullAdder is
port(   A, B    : in bit;
        Cin     : in bit;
        Sum     : out bit;
        Cout    : out bit
);
end FullAdder;

architecture ckt of FullAdder is

        signal sig1, sig2       : bit;

begin

        sig1 <= Cin and (A xor B);
        sig2 <= A and B;

        Sum     <= A xor B xor Cin;
        Cout    <= sig2 or sig1;

end ckt;
```

Figura 06 - Somador Completo.

```
entity Adder6Bits is
port(   A, B    : in bit_vector(5 downto 0);
        Cin     : in bit;
        Cout    : out bit;
        S       : out bit_vector(5 downto 0)
);
end Adder6Bits;

architecture ckt of Adder6Bits is

component FullAdder is
port(   A, B    : in bit;
        Cin     : in bit;
        Sum     : out bit;
        Cout    : out bit
);
end component;

        signal sig      : bit_vector(5 downto 0);

begin

        adder0 : FullAdder
        port map(A(0), B(0), Cin, S(0), sig(0));

        adder1 : FullAdder
        port map(A(1), B(1), sig(0), S(1), sig(1));

        adder2 : FullAdder
        port map(A(2), B(2), sig(1), S(2), sig(2));

        adder3 : FullAdder
        port map(A(3), B(3), sig(2), S(3), sig(3));

        adder4 : FullAdder
        port map(A(4), B(4), sig(3), S(4), sig(4));

        adder5 : FullAdder
        port map(A(5), B(5), sig(4), S(5), sig(5));

        Cout <= sig(5);

end ckt;
```

Figura 07 - Somador de 6 bits.

3.2 Implementação final.

```
entity lab05 is
port(   B       : in bit_vector(6 downto 0);
        V       : out bit_vector(7 downto 0)
);
end lab05;


architecture ckt of lab05 is
--- BIN TO BCD ---
component lab03 is
port(   SW      : in bit_vector(7 downto 0);
        HEX0    : out bit_vector(3 downto 0);
        HEX1    : out bit_vector(3 downto 0);
        HEX2    : out bit_vector(3 downto 0)
);
end component;
--- ADDER 6 BITS ---
component Adder6Bits is
port(   A, B    : in bit_vector(5 downto 0);
        Cin     : in bit;
        Cout    : out bit;
        S       : out bit_vector(5 downto 0)
);
end component;
```

```
--- MUX 2x1 ---
component Mux2x1 is
port(   I0, I1  : in bit;
        s       : in bit;
        enable  : in bit;
        Y       : out bit
);
end component;

-- SIGNALS ---

        signal MuxToAdder       :       bit_vector(5 downto 0);
        signal sig              :       bit_vector(5 downto 0);
        signal CoutAdder        :       bit;
        signal AdderToBCD       :       bit_vector(7 downto 0);
        signal cen              :       bit_vector(3 downto 0);
        signal dec              :       bit_vector(3 downto 0);
        signal uni              :       bit_vector(3 downto 0);
        signal B2               :       bit_vector(6 downto 0);

begin

        B2(6 downto 0) <= not B(6 downto 0);
```

```
                    mux0 : mux2x1
                    port map(B(0), B2(0), B(6), '1', MuxToAdder(0));

                    mux1 : mux2x1
                    port map(B(1), B2(1), B(6), '1', MuxToAdder(1));

                    mux2 : mux2x1
                    port map(B(2), B2(2), B(6), '1', MuxToAdder(2));

                    mux3 : mux2x1
                    port map(B(3), B2(3), B(6), '1', MuxToAdder(3));

                    mux4 : mux2x1
                    port map(B(4), B2(4), B(6), '1', MuxToAdder(4));

                    mux5 : mux2x1
                    port map(B(5), B2(5), B(6), '1', MuxToAdder(5));

                    --- Adder 6 bits ---

                    sig(5 downto 0) <= "000000";
    --- Adder 6 bits ---

    sig(5 downto 0) <= "000000";

    adder : Adder6bits
    port map(MuxToAdder(5 downto 0), sig(5 downto 0), B(6), CoutAdder, AdderToBCD(5 downto 0));

    --- BIN TO BCD ---

    AdderToBCD(7 downto 6) <= "00";

    bcd : lab03
    port map(AdderToBCD(7 downto 0), uni(3 downto 0), dec(3 downto 0), cen(3 downto 0));

    V(3 downto 0) <= uni(3 downto 0);
    V(7 downto 4) <= dec(3 downto 0);

end ckt;
```

Figura 08 - Código Final.

04 Conclusão.

Por fim, para concluir a execução do código e do projeto, fizemos uma simulação para verificar que o código está correto como previsto. Foi colocado como entrada no primeiro exemplo (**Figura 09)** B como “000 0100” (4 em binário), e no segundo exemplo (**Figura 10)** foi colocado B como “111 1100” (-4 em binário).

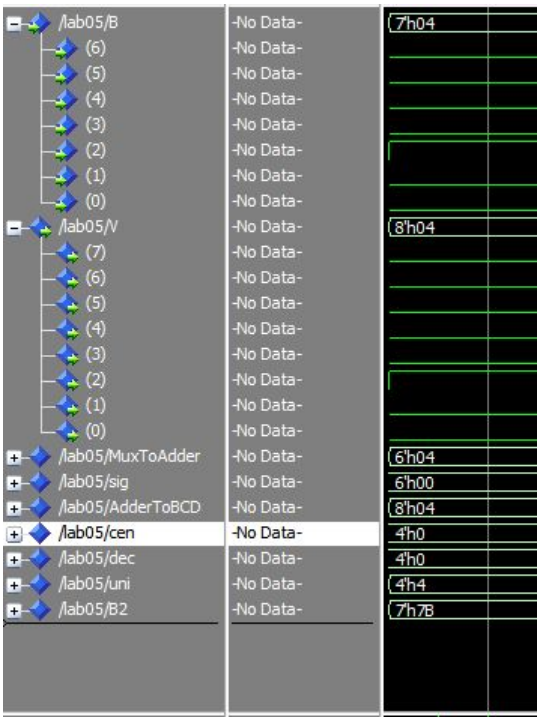


Figura 09

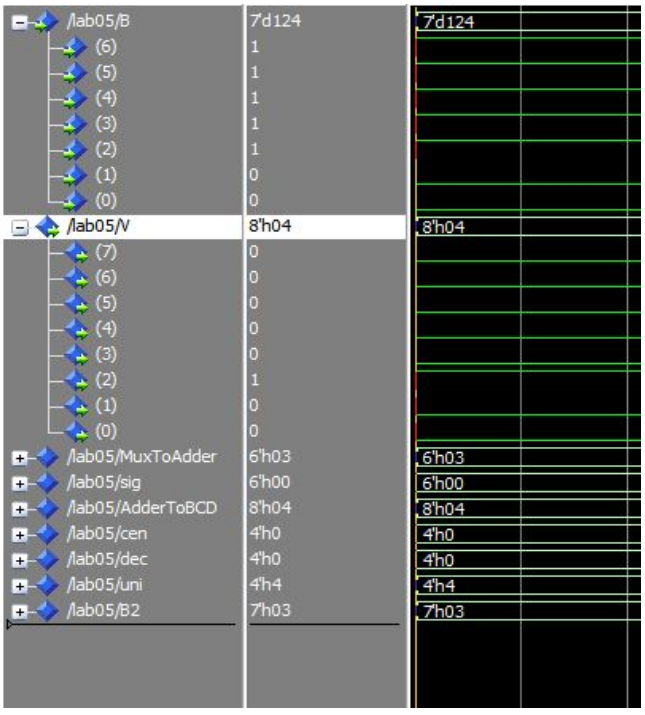


Figura 10