



UNIVERSIDADE FEDERAL DE VIÇOSA
CIÊNCIA DA COMPUTAÇÃO
CFC 251-Introdução aos Sistemas Lógicos Digitais

Primeiro Trabalho Prático

Código Morse

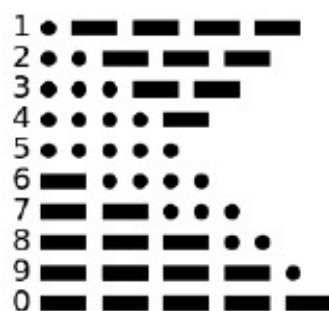
Cláudio Barbosa – 3492
Filipe Fonseca – 3502
Guilherme Corrêa – 3509
Isabella Ramos - 3474

Professor: José Augusto Miranda Nacif
Monitor: Lucas Ferreira S. Duarte

1. INTRODUÇÃO

Este trabalho visa colocar em prática e expandir os conhecimentos adquiridos em sala de aula e, para tal, será utilizado o sistema de codificação utilizado no Padrão Internacional do Código Morse. O Código Morse consiste em um sistema binário utilizado para representação/comunicação de letras, números e também sinais de pontuação.

Figura 1: Representação numérica em Código Morse



Será implementado um codificador Morse que seja capaz de converter um número (0 à 9) em seu sinal correspondente em Código Morse. Na figura 1 está representado o padrão de como a codificação numérica será feita.

2. IMPLEMENTAÇÃO

2.1. Objetivos

O objetivo principal é a implementação de dois módulos: número e transmissão, sendo que o resultado final deverá ser uma arquitetura que permita a leitura de um código binário e a exibição de seu valor equivalente em Morse.

Para tal serão utilizados software de simulação e implementação, sendo eles: Icarus Verilog para elaboração e simulação dos módulos em *Verilog*; *GTKWave* para visualização das formas de onda resultantes; *Logisim* para elaboração de circuitos com portas lógicas.

2.2. Elaboração

Para a elaboração deste trabalho foi utilizada lógica combinacional. Sendo assim o primeiro passo a ser realizado é o levantamento das equações booleanas de cada saída, que será possível com a representação da tabela verdade das mesmas (Tabela 1).

As equações obtidas foram:

$$M1: A'B'C'D + A'B'CD' + A'B'CD + A'BC'D + A'BC'D$$

$$M2: A'B'CD' + A'B'CD + A'BC'D + A'BC'D + A'BCD'$$

$$M3: A'B'CD + A'BC'D + A'BC'D + A'BCD' + A'BCD$$

$$M4: A'BC'D + A'BC'D + A'BCD' + A'BCD + AB'C'D'$$

$$M5: A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D$$

De posse da tabela verdade o objetivo é sintetizar ao máximo as equações pois serão estas simplificações que iremos utilizar. Para tal é necessário utilizar lógica booleana e mapas de Karnaugh. Abaixo estão os mapas de cada uma das saídas: M1(), M2(), M3(), M4() e M5().

Tabela 1: Tabela verdade do módulo número.

A	B	C	D	M1	M2	M3	M4	M5
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0
0	0	1	0	1	1	0	0	0
0	0	1	1	1	1	1	0	0
0	1	0	0	1	1	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1
0	1	1	1	0	0	1	1	1
1	0	0	0	0	0	0	1	1
1	0	0	1	0	0	0	0	1
1	0	1	0	X	X	X	X	X
1	0	1	1	X	X	X	X	X
1	1	0	0	X	X	X	X	X
1	1	0	1	X	X	X	X	X
1	1	1	0	X	X	X	X	X
1	1	1	1	X	X	X	X	X

Tabela 2: Mapa de Karnaugh para a saída M1

CD \ AB	00	01	11	10
00	0	1	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

Tabela 3: Mapa de Karnaugh para a saída M2

CD \ AB	00	01	11	10
00	0	1	1	1
01	1	1	0	1
11	X	X	X	X
10	0	0	X	X

Tabela 4: Mapa de Karnaugh para a saída M3

AB \ CD	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	X	X	X	X
10	0	0	X	X

Tabela 5: Mapa de Karnaugh para a saída M4

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	1	0	X	X

Tabela 6: Mapa de Karnaugh para a saída M5

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

Realizando então as simplificações através do mapa de Karnaugh agrupando o máximo de 1's possíveis é possível obter as seguintes **equações simplificadas**:

M1: $C'(D+B) + CA'B'$

M2: $BC' + A'B'C + CD'$

M3: $A'B + CD$

M4: $A'B' + AC'D'$

M5: $AC + DB + CB$

De posse das equações simplificadas é interessante realizar a representação em um circuito. O Logisim foi o software utilizado para representar como funcionará o circuito. Temos respectivamente as entradas/saídas M1 (Figura 2), M2 (Figura 3), M3 (Figura 4), M4 (Figura 5) e M5 (Figura 6).

Figura 2: Representação M1

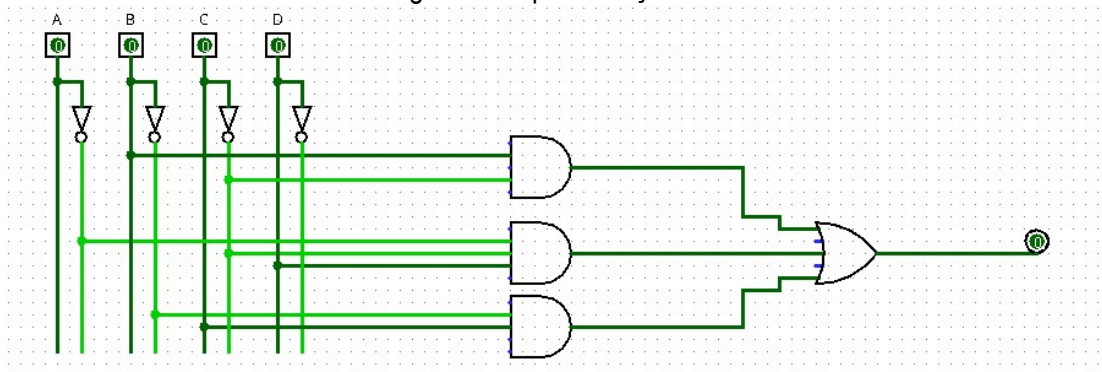


Figura 3: Representação M2

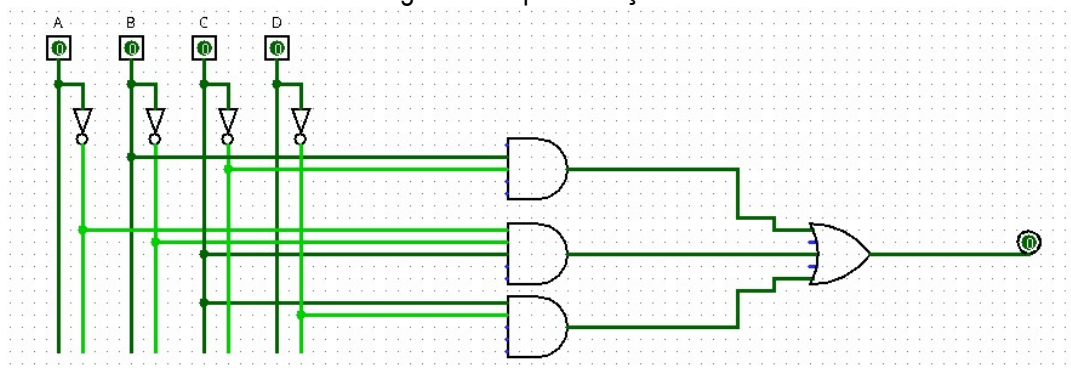


Figura 4: Representação M3'

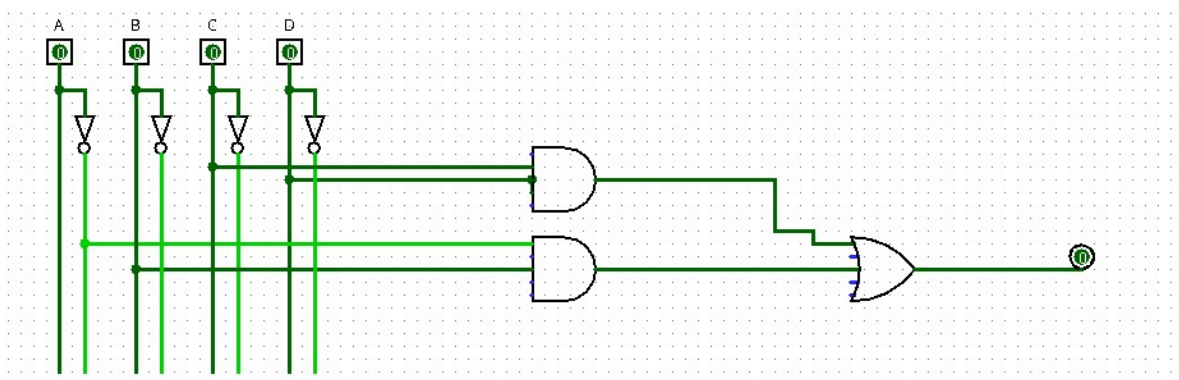


Figura 5: Representação M4

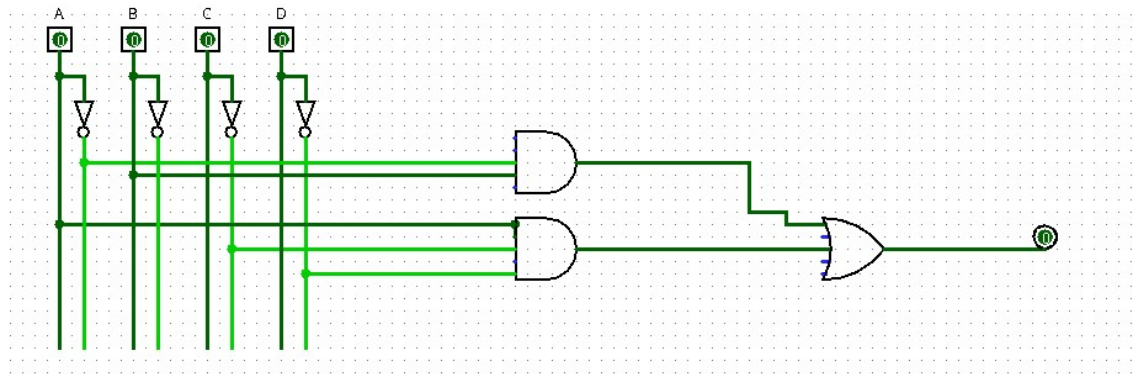
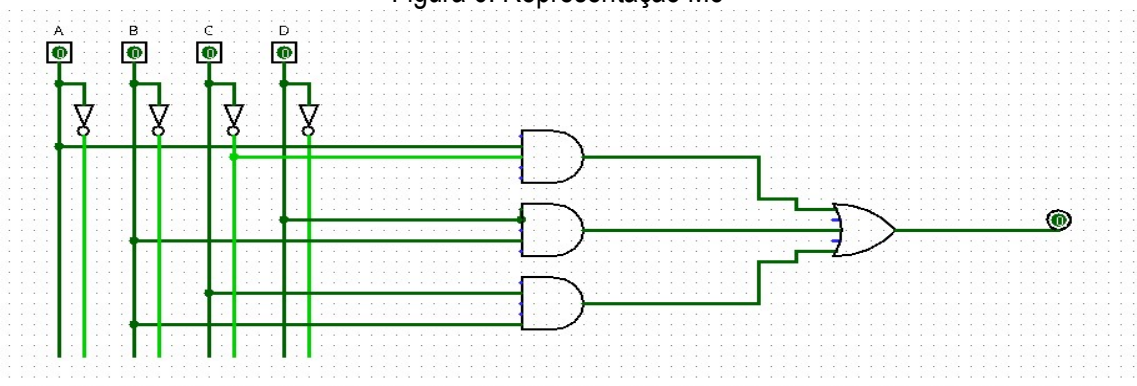


Figura 6: Representação M5



As **equações das formas canônicas** podem ser obtidas também pela tabela verdade presente na Tabela 1. Sendo elas:

a) Forma da soma dos produtos:

$$M1(A,B,C,D) = \Sigma m(1,2,3,4,5) + \Sigma d(10,11,12,13,14,15)$$

$$M2(A,B,C,D) = \Sigma m(2,3,4,5,6) + \Sigma d(10,11,12,13,14,15)$$

$$M3(A,B,C,D) = \Sigma m(3,4,5,6,7) + \Sigma d(10,11,12,13,14,15)$$

$$M4(A,B,C,D) = \Sigma m(4,5,6,7,8) + \Sigma d(10,11,12,13,14,15)$$

$$M4(A,B,C,D) = \Sigma m(5,6,7,8,9) + \Sigma d(10,11,12,13,14,15)$$

b) Forma de produto das somas:

$$M1(A,B,C,D) = \Pi M(0,6,7,8,9) + \Pi D(10,11,12,13,14,15)$$

$$M2(A,B,C,D) = \Pi M(0,1,6,7,8) + \Pi D(10,11,12,13,14,15)$$

$$M3(A,B,C,D) = \Pi M(0,1,2,8,9) + \Pi D(10,11,12,13,14,15)$$

$$M4(A,B,C,D) = \Pi M(0,1,2,3,9) + \Pi D(10,11,12,13,14,15)$$

$$M5(A,B,C,D) = \Pi M(0,1,2,3,4) + \Pi D(10,11,12,13,14,15)$$

Outra representação que pode ser utilizada é a de **mintermos e maxtermos**:

a) Mintermos:

$$M1 = A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + A'B'C'D$$

$$M2 = A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + A'BCD'$$

$$M3 = A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD$$

$$M4 = A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D$$

$$M5 = A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D$$

b) Maxtermos:

$M1 = (A+B+C+D) (A+B'+C'+D) (A+B'+C'+D') (A'+B+C+D) (A'+B+C+D')$

$M2 = (A+B+C+D) (A+B+C+D') (A+B'+C'+D') (A'+B+C+D) (A'+B+C+D')$

$M3 = (A+B+C+D) (A+B+C+D') (A+B+C'+D) (A'+B+C+D) (A'+B+C+D')$

$M4 = (A+B+C+D) (A+B+C+D') (A+B+C'+D) (A+B+C'+D') (A'+B+C+D')$

$M5 = (A+B+C+D) (A+B+C+D') (A+B+C'+D) (A+B+C'+D') (A+B'+C+D)$

Todos os dados obtidos acima compõem a elaboração do modulo número, que é aquele que realiza a tradução dos sinais de entrada para sua saída em Código Morse. Para a elaboração e simulação dos módulos foi utilizado o software Icarus Verilog.

O código de implementação e simulação assim ficaram:

```
module m(a,b,c,d,m1,m2,m3,m4,m5,reset,ready);
    input wire a,b,c,d, reset, ready;
    output reg m1,m2,m3,m4,m5;

    always @ (*) begin /*= todas as entradas
        if(reset) begin
            m1=0;
            m2=0;
            m3=0;
            m4=0;
            m5=0;
        end

        else if (ready) begin
            m1 = (b & ~c) | (~a & ~c & d) | (c & ~b); //~a & ((b &
~c) | (~c & d) | (~b & c));
            m2 = (b & ~c) | (~a & ~b & c) | (c & ~d);
            m3 = (~a & b) | (c & d);
            m4 = (~a & b) | (a & ~c & ~d);
            m5 = (a & ~c) | (d & b) | (c & b);
            //m1<=1; não bloqueante = faz receber 1 ao mesmo tempo |
apenas em clock
            //m1=1; bloqueante
        end
    end
endmodule
```

Módulo para simulação:

```
`include "m.v"

module testbench ();
    reg a,b,c,d,reset,ready;
    wire m1,m2,m3,m4,m5;

    m instancial (.a(a), .b(b), .c(c), .d(d),
        .m1(m1), .m2(m2), .m3(m3), .m4(m4), .m5(m5), .reset(reset), .ready(ready)
    );

    initial begin
        $dumpfile("m.vcd");
        $dumpvars(0, testbench);
        $display("a b c d = m1 m2 m3 m4 m5");
    end
endmodule
```

```

        $monitor("%b%b%b%b = %b %b %b %b
%b", a,b,c,d,m1,m2,m3,m4,m5);
    end

    //always @ ( * ) begin
    // #1; clock = ~clock;
    //end

    initial begin
        // #1; clock = 4'b1;
        // #1; reset = 1'b1;
        // #1; reset = 1'b0;
        reset = 1;
        #1; reset = 0;
        #1; ready=0; a=0; b=0; c=0; d=1; ready=1;
        #1; ready=0; a=0; b=0; c=1; d=0; ready=1;
        #1; ready=0; a=0; b=0; c=1; d=1; ready=1;
        #1; ready=0; a=0; b=1; c=0; d=0; ready=1;
        #1; ready=0; a=0; b=1; c=0; d=1; ready=1;
        #1; ready=0; a=0; b=1; c=1; d=0; ready=1;
        #1; ready=0; a=0; b=1; c=1; d=1; ready=1;
        #1; ready=0; a=1; b=0; c=0; d=0; ready=1;
        #1; ready=0; a=1; b=0; c=0; d=1; ready=1;
        #1; ready=0; a=0; b=0; c=0; d=0; ready=1;

        $finish;
    end

endmodule

```

Para verificar o comportamento do módulo em função das formas de onda, foi utilizado o código para simulação no software *GTKWave* que apresentou o seguinte resultado das entradas (a,b,c,d) e das saídas (m1,m2,m3,m4,m5):

Figura 7: Representação das ondas x tempo (GTKWave)



3 Considerações finais

A realização deste trabalho permitiu a utilização dos conhecimentos em lógica combinacional, Verilog e apresentou softwares importantes e que são utilizados para a simulação e também implementação do código.

A representação de ondas obtida através do programa GTKWave permite visualizar o funcionamento durante o tempo de execução. Os resultados obtidos foram de acordo com o esperado sendo que uma segunda parte deste trabalho consistirá na implementação dos módulos em um sistema FPGA.