



Projecto e Controlo em Lógica Digital

0º Laboratório

Trabalho 1 – Olá Mundo

Objectivo: Escrever “Ola Mundo” em displays de 7 segmentos

Lançar o Quartus II e abrir o projecto DE2_top;

Alterar o programa para acender (estaticamente) os segmentos correctos dos displays de forma a indicarem “Ola mundo”. Sugere-se a seguinte forma:

OL Añ Und0

Trabalho 2 – Descodificador de 7 segmentos

Objectivo: descodificar números em binário para que estes sejam mostrados nos displays de 7 segmentos em base decimal. Utilizar módulos em verilog para implementar dois descodificadores. Os números serão introduzidos através dos interruptores (8) da placa em binário devendo aparecer em decimal nos displays.

Lançar o Quartus II e abrir o projecto DE2_top;

Identificar os sinais dos oito interruptores (SW) bem como os sinais correspondentes a dois displays de 7 segmentos;

Implementar um módulo que tem como entradas os “números” em binário e como output as linhas para os displays de 7 segmentos.

Implementar dois descodificadores utilizando quatro interruptores para o primeiro número e quatro interruptores para o segundo número.

Descodificadores usando lógica combinatória (mapas de Karnaugh)

Começamos por projectar um circuito que, usando os switches para representar um número em binário de 2 bits, apresente esse número no painel de 7 segmentos. Vamos implementar este circuito usando tabelas de Karnaugh.

Depois de fazermos e simplificarmos as tabelas de Karnaugh obtemos o seguinte:

A= $\neg M1 \cdot M2$
B= 0
C= $\neg M2 \cdot M1$
D= $\neg M1 \cdot M2$
E= $M2$
F= $M2 + M1$
G= $\neg M1$

Logo o código é:

```
//usar os sw 5 e 6 para introduzir um numero em binario e aparece no display
de sete segmentos (fazer logica combinatoria)
assign    HEX0[0]      =    !SW[6] && SW[5];
assign    HEX0[1]      =    1'b0;
assign    HEX0[2]      =    !SW[5] && SW[6];
assign    HEX0[3]      =    !SW[6] && SW[5];
assign    HEX0[4]      =    SW[5];
assign    HEX0[5]      =    SW[5] || SW[6];
assign    HEX0[6]      =    !SW[6];
```

Descodificadores usando “case statement”

Agora vamos apresentar, no ecran de sete segmentos:

- 1 se os switches estiverem a 0 0
- 3 se os switches estiverem a 0 1
- 5 se os switches estiverem a 1 0
- 7 se os switches estiverem a 1 1

O código é:

```
reg [1:0] sel;
sel[1]=SW[5];
sel[0]=SW[6];
case (sel)
    2'b00 : begin
        assign    HEX0[0]      =    1'b1;
assign    HEX0[1]      =    1'b0;
assign    HEX0[2]      =    1'b0;
assign    HEX0[3]      =    1'b1;
assign    HEX0[4]      =    1'b1;
assign    HEX0[5]      =    1'b1;
```

```

assign    HEX0[6]      =    1'b1;
end

    2'b01 : begin
        assign    HEX0[0]      =    1'b0;
assign    HEX0[1]      =    1'b0;
assign    HEX0[2]      =    1'b0;
assign    HEX0[3]      =    1'b0;
assign    HEX0[4]      =    1'b1;
assign    HEX0[5]      =    1'b1;
assign    HEX0[6]      =    1'b0;
    end
    2'b10 : begin
        assign    HEX0[0]      =    1'b0
assign    HEX0[1]      =    1'b1;
assign    HEX0[2]      =    1'b0;
assign    HEX0[3]      =    1'b0;
assign    HEX0[4]      =    1'b1;
assign    HEX0[5]      =    1'b0;
assign    HEX0[6]      =    1'b0;
    end
    2'b11 : begin
        assign    HEX0[0]      =    1'b0
assign    HEX0[1]      =    1'b0;
assign    HEX0[2]      =    1'b0;
assign    HEX0[3]      =    1'b1;
assign    HEX0[4]      =    1'b1;
assign    HEX0[5]      =    1'b1;
assign    HEX0[6]      =    1'b1;
    end
endcase

```