Michelle da Costa Silva 427448

Correção da prova

01.)Projetar um circuito binário com portas lógicas capaz de produzir as seguintes saídas

	abcd	saidas	s2	s1	s0
0	0000	2	0	1	0
1	0001	4	1	0	0
2	0010	4	1	0	0
3	0011	4	1	0	0
4	0100	1	0	0	1
5	0101	2	0	1	0
6	0110	4	1	0	0
7	0111	4	1	0	0

	abcd	saidas	s2	s1	s0
8	1000	1	0	0	1
9	1001	1	0	0	1
10	1010	2	0	1	0
11	1011	4	1	0	0
12	1100	1	0	0	1
13	1101	1	0	0	1
14	1110	1	0	0	1
15	1111	2	0	1	0

a.) Montar a equação completa do sinal s2 por mintermos

∑m (1, 2, 3, 6, 7, 11)

f(x, y, w, z) = (x'*y'*w'*z) + (x'*y'*w*z') + (x'*y'*w*z) + (x'*y*w*z') + (x'*y*w*z) + (x*y'*w*z)

b.) Montar a equação completa do sinal s1 por maxtermos

∏M (1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14)

c.) Montar a equação completa do sinal s0 por mintermos

∑m (4, 8, 9, 12, 13, 14)

f(x, y, w, z) = (x'*y*w'*z') + (x*y'*w'*z') + (x*y'*w'*z) + (x*y*w'*z') + (x*y*w'*z) + (x*y*w*z')

d.) Montar a equação simplificada do sinal s2 pelo mapa de Veitch-Karnaugh

s2				
ab\cd	00	01	11	10
00	0	1	1	1
01	0	0	1	1

11	0	0	0	0
10	0	0	1	0

(2, 3, 6, 7) = (x'*y'*w*z') + (x'*y'*w*z) + (x'*y*w*z') + (x'*y*w*z)

= (x'*y'*w) + (x'*y*w)

= (x'*w)

(3, 11) = (x'*y'*w*z) + (x*y'*w*z)

= (y'*w*z)

(1, 3) = (x'*y'*w'*z) + (x'*y'*w*z)

= (x'*y'*z)

(x'*y'*z) + (y'*w*z) + (x'*w)

e.) Montar a equação simplificada do sinal s1 pelo mapa de Veitch-Karnaugh

s1				
ab\cd	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

(1, 3)= (X+Y+W+Z')*(X+Y+W'+Z')

= (X+Y+Z')

(2, 3, 6, 7)= (X+Y+W'+Z)* (X+Y+W'+Z')*(X+Y'+W'+Z)* (X+Y'+W'+Z')

= (X+W')

(6, 14)= (X+Y'+W'+Z)*(X'+Y'+W'+Z)

= (Y'+W'+Z)

(4, 12)= (X+Y'+W+Z)*(X'+Y'+W+Z)

= (Y'+W+Z)

(8, 9, 12, 13)= (X'+Y+W+Z)* (X'+Y+W+Z')*(X'+Y'+W+Z)* (X'+Y'+W+Z')

= (X'+W)

(9, 11)= (X'+Y+W+Z')*(X'+Y+W'+Z')

= (X'+Y+Z')

(X+Y+Z')* (X+W')* (Y'+W'+Z)* (Y'+W+Z)*(X'+W)* (X'+Y+Z')

f.) Montar a equação simplificada do sinal s0 pelo mapa de Veitch-Karnaugh

s0				
ab\cd	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
1º	1	1	0	0

(8, 9, 12, 13) = (x*y'*w'*z') + (x*y'*w'*z) + (x*y*w'*z') + (x*y*w'*z)

= (x*w')

(4, 12) = (x'*y*w'*z') + (x*y*w'*z')

= (y*w'*z')

(12, 14) = (x*y*w'*z') + (x*y*w*z')

= (x*y*z')

(x*w') + (y*w'*z') + (x*y*z')

g.) Montar a equação simplificada do sinal s2 pelas propriedades da álgebra

(x'*y'*z) + (y'*w*z) + (x'*w)

(x'*y'*z) + w* (y'*z +x')

h.) Montar a equação simplificada do sinal s1 pelas propriedades da álgebra

(X+Y+Z')* (X+W')* (Y'+W'+Z)* (Y'+W+Z)*(X'+W)* (X'+Y+Z')

(Y+Z')* (Y'+Z)*(X'+W) *(X+W')

(X xor W)' * (Y xor Z)'

i.) Montar a equação simplificada do sinal s0 pelas propriedades da álgebra

(x*w') + (y*w'*z') + (x*y*z')

y*z'* (w'+x) + (x*w')

j.) Montar a descrição por portas do Verilog do módulo capaz de calcular s2 pela expressão em (d)

```
// -------------------------
// -- j
// -------------------------

   module j (output saida, output s1, output s2, output s3, output
saida_not1, output saida_not2, input x, input z, input y, input w);

    not NOT1(saida_not1, x);
    not NOT2(saida_not2, y);

    and AND1(s1, saida_not1, saida_not2,z);
    and AND2(s2, saida_not2, w, z);
    and AND3(s3, saida_not1, w);

    or OR1(saida, s1, s2, s3);

    endmodule //j
```

k.) Montar a descrição por expressão no Verilog do módulo capaz de calcular s1 pela expressão em (h)

```
// -------------------------
// -- k
// -------------------------
   module k (output s, input x, input w, input y, input z);
   assign s = ~(x ^ w)& ~(y ^ z);

   endmodule // notgate
```

l.) Montar a descrição por portas do Verilog do módulo capaz de calcular s0, usando entradas e saídas múltiplas (em Little Endian)

```
   // -------------------------
   // -- l
   // -------------------------

    module l (output [2:0] saida,
              output [3:0] s1,
              output [3:0] s2,
              output [3:0] s3,
              output [3:0] saida_not1,
              output [3:0] saida_not2,
              input [3:0] x,
              input [3:0] z,
              input [3:0] y,
              input [3:0] w);

          not NOT1(saida_not1, w);
          not NOT2(saida_not2, z);

          and AND1(s1, saida_not1, saida_not2,y);
          and AND2(s2, saida_not1, x);
          and AND3(s3, saida_not2, x, y);

          or OR1(saida, s1, s2, s3);
          endmodule //l
```