UNIVERSIDADE DE SÃO PAULO
ENGENHARIA DE COMPUTAÇÃO
SISTEMAS DIGITAIS

Exercício-Programa 1: PCS3100

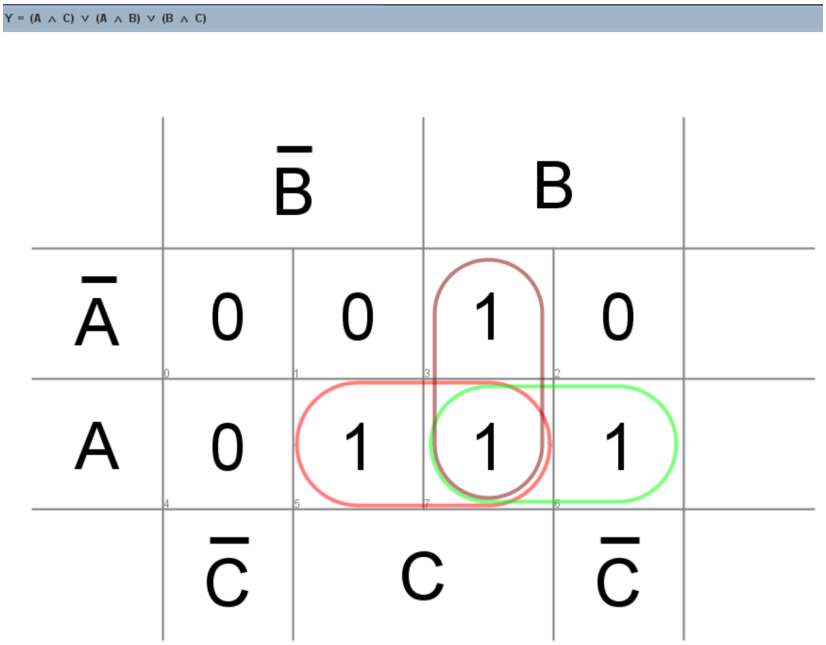
Gabriel Agra de Castro Motta

São Paulo, Outubro de 2024

Parte A:

Desenvolvimento e equação algébrica do circuito:

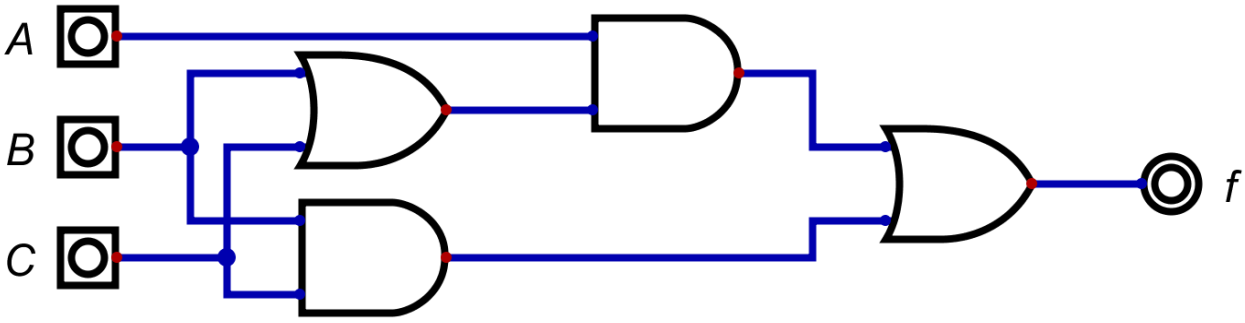
Para análise lógica do sistema, montamos um mapa de karnaugh a partir do Digital.



Mapa de Karnaugh da função

Assim, concluímos a expressão lógica do circuito como sendo:
f = AB + AC + BC

Seguindo as regras do EP de máximo de 2 inputs por porta lógica e limite de 4 portas logicas, substituímos essa expressão por:
f = ((B + C) * A) + (B * C)



Circuito representante de ((B + C) * A) + (B * C)

Descrição do Verilog do circuito:
A partir disso, montamos um código em Verilog com os inputs que atribui ao output f a expressão “((B + C) * A) + (B * C)”

```
1  module ep01A (a, b, c, f);
2      input a, b, c;
3      output f;
4
5      // coloque aqui a descricao do circuito
6      assign f = ((b | c) & a) | (b & c);
7  endmodule
```

Output f = ((B + C) * A) + (B * C)

Testbench criado para testar o circuito:
Para o testbench, testamos as 9 possibilidades de permutações distintas de inputs para testar cada output, caso-a-caso, e usamos monitor para verificar os resultados

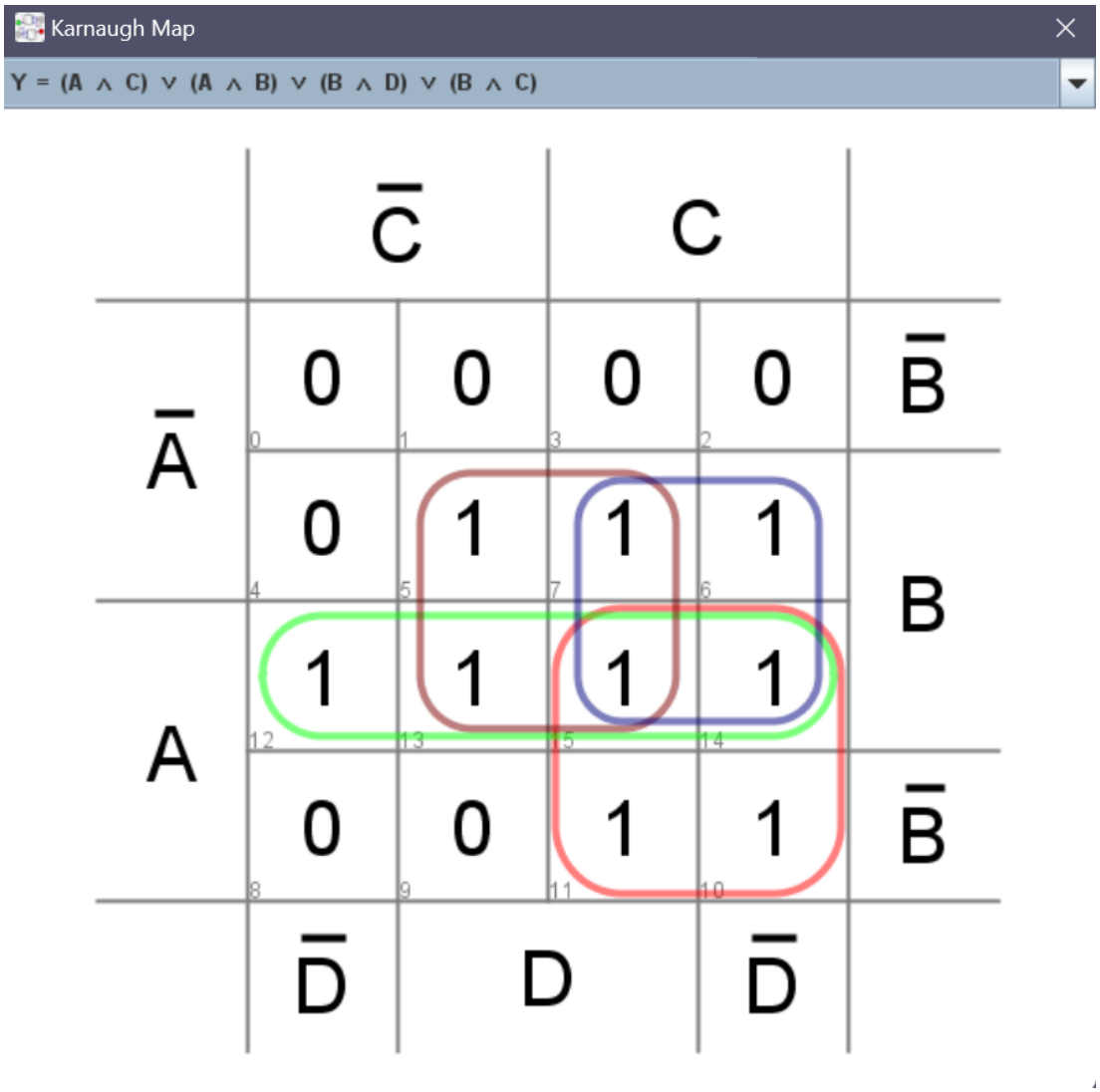
```
module tb_ep01A;
    reg a, b, c;
    wire f;

    ep01A uut (
        .a(a),
        .b(b),
        .c(c),
        .f(f)
    );

    initial begin
        $monitor("a = %b, b = %b, c = %b, f = %b", a, b, c, f);

        a = 0; b = 0; c = 0; #10;
        a = 0; b = 0; c = 1; #10;
        a = 0; b = 1; c = 0; #10;
        a = 0; b = 1; c = 1; #10;
        a = 1; b = 0; c = 0; #10;
        a = 1; b = 0; c = 1; #10;
        a = 1; b = 1; c = 0; #10;
        a = 1; b = 1; c = 1; #10;

        $finish;
    end
endmodule
```

Testbench para as 9 possibilidades com atraso adicionado

Parte B:

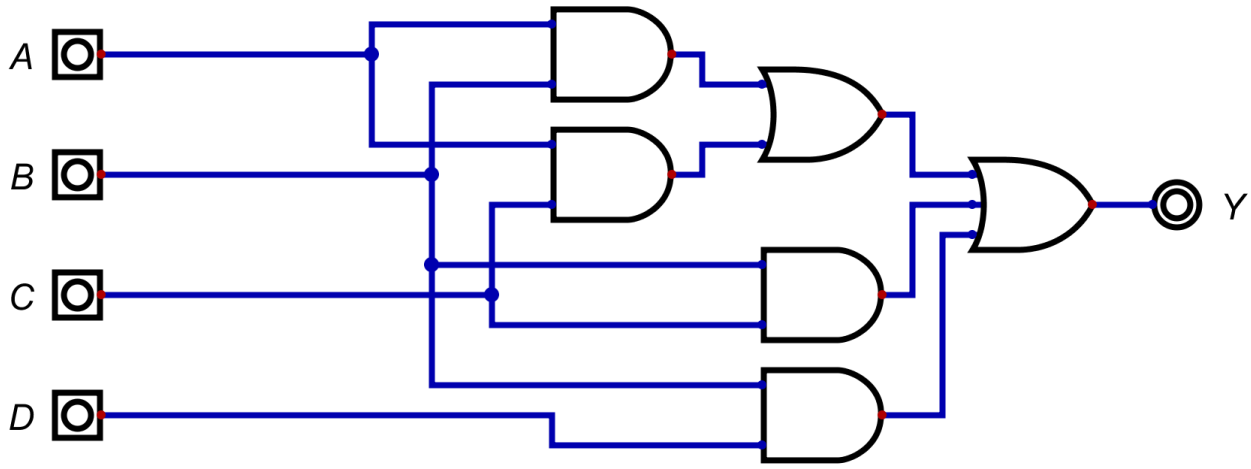
Desenvolvimento e equação algébrica do circuito:

Para análise lógica do sistema, montamos um mapa de karnaugh a partir do Digital.



Pelos implicantes primos essenciais, concluímos f = AB + AC + BC + BD

E, novamente, para cumprir com as regras do EP e também não degradar muito o sinal, descrevemos o circuito de forma alternativa, como f = (AB + AC) + BC + CD.
Assim, reduzindo o input da última porta lógica OR de 4 bits para 3 bits.



Circuito representante de (AB + AC) + BC + CD

Descrição do Verilog do circuito:
A partir disso, montamos um código em Verilog com os inputs que atribui ao output f a expressão “(AB + AC) + BC + CD”

```
1  module ep01B (a, b, c, d, f);
2      input a, b, c, d;
3      output f;
4
5      // coloque aqui a descricao do circuito
6      assign f = (a & b) | (a & c) | (b & c) | (b & d);
7  endmodule
8
```

Testbench criado para testar o circuito:
Para o testbench, testamos as 16 possibilidades de permutações distintas de inputs para testar cada output, caso-a-caso, e usamos monitor para verificar os resultados

```
module tb_ep01B;
    reg a, b, c, d;
    wire f;

    ep01B uut (
        .a(a),
        .b(b),
        .c(c),
        .d(d),
        .f(f)
    );

    initial begin
        $monitor("a = %b, b = %b, c = %b, d = %b, f = %b", a, b, c, d, f);

        a = 0; b = 0; c = 0; d = 0; #10;
        a = 0; b = 0; c = 0; d = 1; #10;
        a = 0; b = 0; c = 1; d = 0; #10;
        a = 0; b = 0; c = 1; d = 1; #10;
        a = 0; b = 1; c = 0; d = 0; #10;
        a = 0; b = 1; c = 0; d = 1; #10;
        a = 0; b = 1; c = 1; d = 0; #10;
        a = 0; b = 1; c = 1; d = 1; #10;
        a = 1; b = 0; c = 0; d = 0; #10;
        a = 1; b = 0; c = 0; d = 1; #10;
        a = 1; b = 0; c = 1; d = 0; #10;
        a = 1; b = 0; c = 1; d = 1; #10;
        a = 1; b = 1; c = 0; d = 0; #10;
        a = 1; b = 1; c = 0; d = 1; #10;
        a = 1; b = 1; c = 1; d = 0; #10;
        a = 1; b = 1; c = 1; d = 1; #10;

        $finish;
    end
endmodule
```

Parte C:

Desenvolvimento e equação algébrica do circuito:

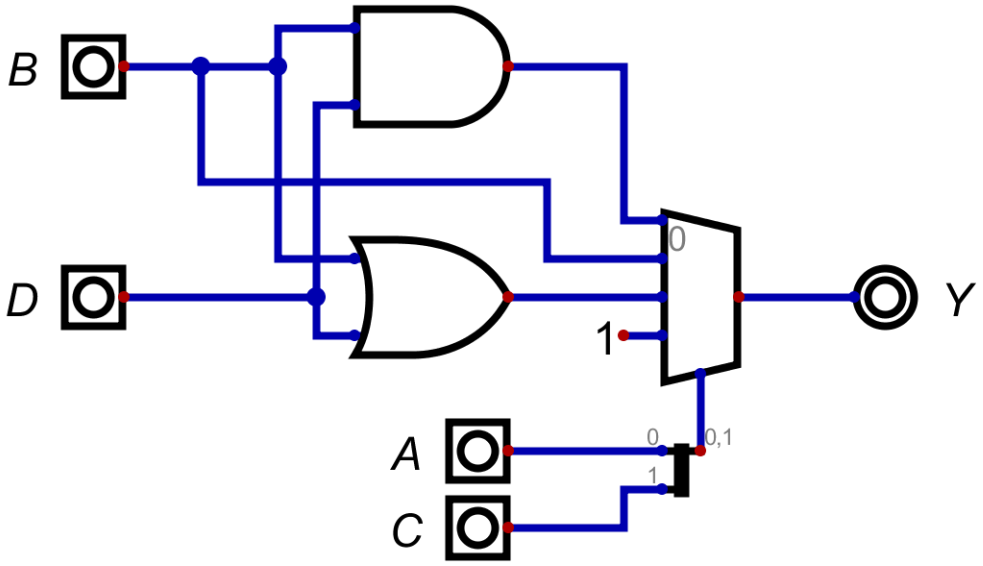
Para o circuito, utilizaremos um multiplexador tal que ‘a’ e ‘c’ sejam seletores, sendo ‘a’ o bit de maior significância. Dessa forma, tratamos o circuito em 4 casos, a partir da funcao da parte B.

a	b	c	d	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

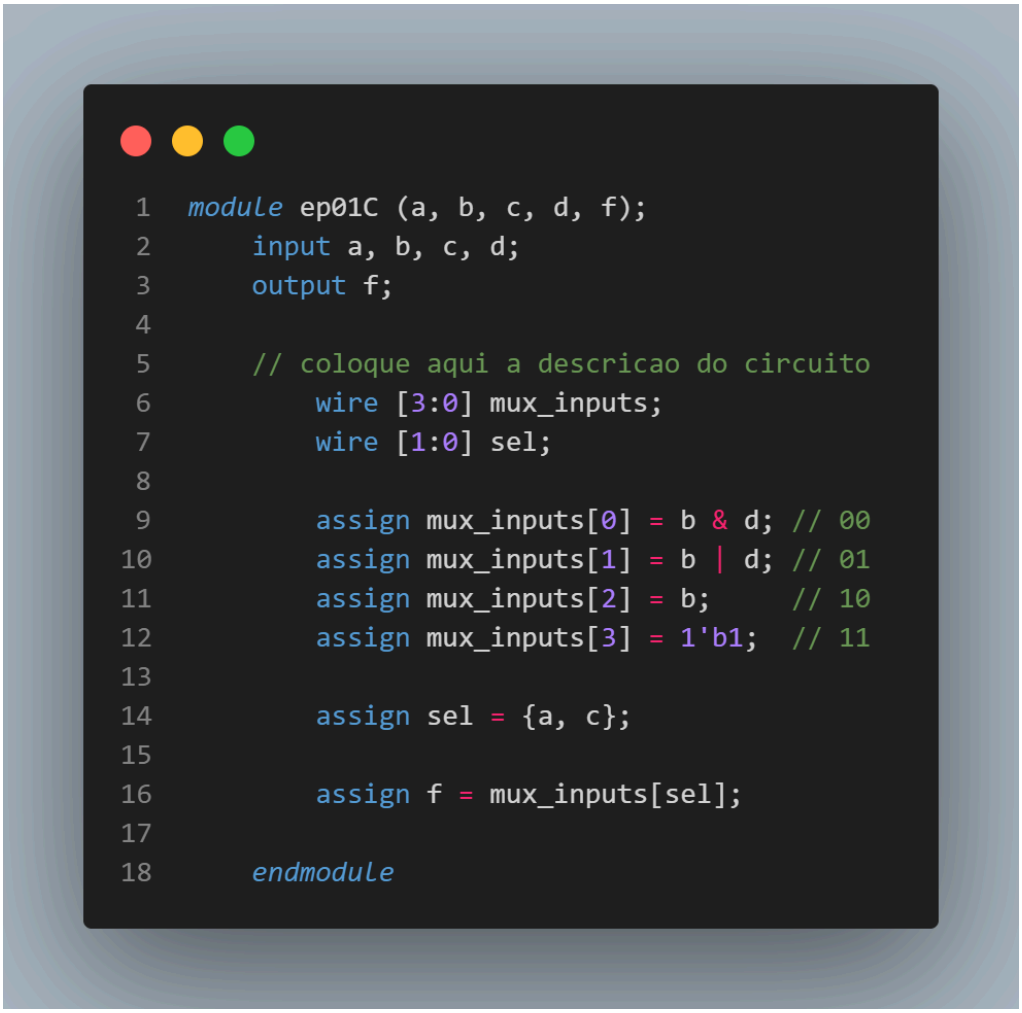
Isto é, vemos quando f é 1 nos casos quando AC = 00, AC= 01, AC =10, AC=11

- Sendo o primeiro caso AC = 00, B & D = 1
- No segundo caso AC =01 , B = 1
- No terceiro caso AC = 10, A + B = 1
- No quarto caso AC = 11, e f sempre será 1

Logo, as entradas do mux ficam “B & D”, “B”, “B | D, “1”.



Descrição do Verilog do circuito:
Para o MUX 4:1, colocamos um fio de 4 bits de input e outro fio para 2 bits de seletores. A partir disso, atribuímos a AC = 00 a expressão “B & D”, e assim por diante de acordo com o caso. Atribuímos os dois bits seletores a e c e atribuímos ao output f a posição do mux inputs definida pelo seletor ac.



```verilog
module ep01C (a, b, c, d, f);
    input a, b, c, d;
    output f;

    // coloque aqui a descricao do circuito
        wire [3:0] mux_inputs;
        wire [1:0] sel;

        assign mux_inputs[0] = b & d; // 00
        assign mux_inputs[1] = b | d; // 01
        assign mux_inputs[2] = b;     // 10
        assign mux_inputs[3] = 1'b1;  // 11

        assign sel = {a, c};

        assign f = mux_inputs[sel];

    endmodule
```

Testbench criado para testar o circuito:
O testbench continua o mesmo da questão anterior, testando as 16 possibilidades e verificando o resultado.

```v
module tb_ep01B;
    reg a, b, c, d;
    wire f;

    ep01B uut (
        .a(a),
        .b(b),
        .c(c),
        .d(d),
        .f(f)
    );

    initial begin
        $monitor("a = %b, b = %b, c = %b, d = %b, f = %b", a, b, c, d, f);

        a = 0; b = 0; c = 0; d = 0; #10;
        a = 0; b = 0; c = 0; d = 1; #10;
        a = 0; b = 0; c = 1; d = 0; #10;
        a = 0; b = 0; c = 1; d = 1; #10;
        a = 0; b = 1; c = 0; d = 0; #10;
        a = 0; b = 1; c = 0; d = 1; #10;
        a = 0; b = 1; c = 1; d = 0; #10;
        a = 0; b = 1; c = 1; d = 1; #10;
        a = 1; b = 0; c = 0; d = 0; #10;
        a = 1; b = 0; c = 0; d = 1; #10;
        a = 1; b = 0; c = 1; d = 0; #10;
        a = 1; b = 0; c = 1; d = 1; #10;
        a = 1; b = 1; c = 0; d = 0; #10;
        a = 1; b = 1; c = 0; d = 1; #10;
        a = 1; b = 1; c = 1; d = 0; #10;
        a = 1; b = 1; c = 1; d = 1; #10;

        $finish;
    end
endmodule
```