


M4 Atividade Vivencial 2 – ARQUITETURA DE SISTEMAS DIGITAIS

Projetar, implementar, integrar e simular uma ULA de 8 bits.

Gabriel Hoffmann

Link do projeto: <https://www.edaplayground.com/x/WHsK>

Código testbench.sv

```
testbench.sv   
1 module base_test();  
2   reg [7:0] a, b;  
3   reg cin;  
4   wire [7:0] s, s_and, s_or, s_n;  
5   reg [2:0] mux_input;  
6  
7   full_adder_8_bit fa8 (a, b, cin, s, cout);  
8   and_8_bit and8 (a, b, s_and);  
9   or_8_bit or8 (a, b, s_or);  
10  not_8_bit n8(a, s_n);  
11  comparison_8_bit c8(a, b, maior, menor, igual, maior_igual, menor_igual);  
12  mux_8_bit m8 (a, mux_input, s_mux);  
13  
14  initial begin  
15  
16    $dumpfile("dump.vcd");  
17    $dumpvars(1);  
18  
19    $display("\nCaso de Teste 1");  
20  
21    a = 8'b10100100;  
22    b = 8'b10100101;  
23    cin= 1'b0;  
24    mux_input = 3'b110;  
25    #10;  
26  
27    $display("\nResultado Somador");  
28    $display("A: %b", a);  
29    $display("B: %b", b);  
30    $display("S: %b", s);  
31    $display("Cin: %b Cout %b", cin, cout);  
32  
33    $display("\nResultado AND");  
34    $display("A: %b", a);  
35    $display("B: %b", b);  
36    $display("&: %b", s_and);  
37  end
```

```

38     $display("\nResultado OR");
39     $display("A: %b", a);
40     $display("B: %b", b);
41     $display("|: %b", s_or);
42
43     $display("\nResultado NOT");
44     $display("A: %b", a);
45     $display("~: %b", s_n);
46
47     $display("\nResultado Comparador");
48     $display("A: %b", a);
49     $display("B: %b", b);
50     $display(">: %b, <: %b, ==: %b, >=: %b, <=: %b", maior, menor, igual, maior_igual, menor_igual);
51
52     $display("\nResultado MUX");
53     $display("A: %b", a);
54     $display("Mux Input: %b", mux_input);
55     $display("Saida: %b", s_mux);
56
57     $display("\nCaso de Teste 2");
58
59     a = 8'b01000111;
60     b = 8'b00110100;
61     cin= 1'b1;
62     mux_input = 3'b001;
63     #10;
64
65     $display("\nResultado Somador");
66     $display("A: %b", a);
67     $display("B: %b", b);
68     $display("S: %b", s);
69     $display("Cin: %b Cout %b", cin, cout);
70
71     $display("\nResultado AND");
72     $display("A: %b", a);
73     $display("B: %b", b);
74     $display("&: %b", s_and);
75
76     $display("&: %b", s_and);
77
78     $display("\nResultado OR");
79     $display("A: %b", a);
80     $display("B: %b", b);
81     $display("|: %b", s_or);
82
83     $display("\nResultado NOT");
84     $display("A: %b", a);
85     $display("~: %b", s_n);
86
87     $display("\nResultado Comparador");
88     $display("A: %b", a);
89     $display("B: %b", b);
90     $display(">: %b, <: %b, ==: %b, >=: %b, <=: %b", maior, menor, igual, maior_igual, menor_igual);
91
92     $display("\nResultado MUX");
93     $display("A: %b", a);
94     $display("Mux Input: %b", mux_input);
95     $display("Saida: %b", s_mux);
96
97 end
98 endmodule

```

Código design.sv

```
1 module full_adder_1_bit (input a, b, cin, output o, cout);
2   wire r_xor1;
3
4   xor xor1 (r_xor1, a, b);
5   xor xor2 (o, r_xor1, cin);
6
7   wire r_and1, r_and2;
8
9   and and1 (r_and1, cin, r_xor1);
10  and and2 (r_and2, a, b);
11
12  or or1 (cout, r_and1, r_and2);
13 endmodule
14
15 module full_adder_8_bit (a, b, cin, o, cout);
16   input [7:0] a, b;
17   input cin;
18   output [7:0] o;
19   output cout;
20   wire [6:0] aux_cout;
21
22   full_adder_1_bit fa0 (a[0], b[0], cin, o[0], aux_cout[0]);
23   full_adder_1_bit fa1 (a[1], b[1], aux_cout[0], o[1], aux_cout[1]);
24   full_adder_1_bit fa2 (a[2], b[2], aux_cout[1], o[2], aux_cout[2]);
25   full_adder_1_bit fa3 (a[3], b[3], aux_cout[2], o[3], aux_cout[3]);
26   full_adder_1_bit fa4 (a[4], b[4], aux_cout[3], o[4], aux_cout[4]);
27   full_adder_1_bit fa5 (a[5], b[5], aux_cout[4], o[5], aux_cout[5]);
28   full_adder_1_bit fa6 (a[6], b[6], aux_cout[5], o[6], aux_cout[6]);
29   full_adder_1_bit fa7 (a[7], b[7], aux_cout[6], o[7], cout);
30
31 endmodule
32
33 module and_8_bit (a, b, o);
34   input [7:0] a, b;
35   output [7:0] o;
36
37   assign o = a & b;
38 endmodule
39
```

```

32
33 module and_8_bit (a, b, o);
34     input [7:0] a, b;
35     output [7:0] o;
36
37     assign o = a & b;
38 endmodule
39
40 module or_8_bit (a, b, o);
41     input [7:0] a, b;
42     output [7:0] o;
43
44     assign o = a | b;
45 endmodule
46
47 module not_8_bit (a, o);
48     input [7:0] a;
49     output [7:0] o;
50
51     assign o = ~a;
52 endmodule
53
54 module comparison_8_bit (input [7:0] a, b, output maior, menor, igual, maior_igual, menor_igual);
55     assign maior = a > b;
56     assign menor = a < b;
57     assign igual = a == b;
58     assign maior_igual = a >= b;
59     assign menor_igual = a <= b;
60 endmodule
61
62 module mux_8_bit (input [7:0] a, input [2:0] s, output o);
63     wire [7:0] and_result;
64
65     assign and_result[0] = ~s[0] & ~s[1] & ~s[2] & a[0];
66     assign and_result[1] = s[0] & ~s[1] & ~s[2] & a[1];
67     assign and_result[2] = ~s[0] & s[1] & ~s[2] & a[2];
68     assign and_result[3] = s[0] & s[1] & ~s[2] & a[3];
69     assign and_result[4] = ~s[0] & ~s[1] & s[2] & a[4];
70     assign and_result[5] = s[0] & ~s[1] & s[2] & a[5];
71     assign and_result[6] = ~s[0] & s[1] & s[2] & a[6];
72     assign and_result[7] = s[0] & s[1] & s[2] & a[7];
73
74     assign o = | and_result;
75 endmodule

```

Log

Caso de Teste 1

Resultado Somador

A: 10100100

B: 10100101

S: 01001001

Cin: 0 Cout 1

Resultado AND

A: 10100100

B: 10100101

&: 10100100

Resultado OR

A: 10100100

B: 10100101

|: 10100101

Resultado NOT

A: 10100100

~: 01011011

Resultado Comparador

A: 10100100

B: 10100101

>: 0, <: 1, ==: 0, >=: 0, <=: 1

Resultado MUX

A: 10100100

Mux Input: 110

Saida: 0

Caso de Teste 2

Resultado Somador

A: 01000111
B: 00110100
S: 01111100
Cin: 1 Cout 0

Resultado AND

A: 01000111
B: 00110100
&: 00000100

Resultado OR

A: 01000111
B: 00110100
|: 01110111

Resultado NOT

A: 01000111
~: 10111000

Resultado Comparador

A: 01000111
B: 00110100
>: 1, <: 0, ==: 0, >=: 1, <=: 0

Resultado MUX

A: 01000111
Mux Input: 001
Saída: 1
Finding VCD file...
./dump.vcd

[2022-11-06 19:47:38 EST] Opening EPWave...

Done

Gráfico

