UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
CONCEPÇÃO ESTRUTURADA DE CIRCUITOS INTEGRADOS

PROJETO MIPS – ENTREGA 6

THIAGO ALVES DE ARAUJO

2016019787

Sumário

1 Tabela de figuras

2 ULA 1 bit

Agora vamos desenvolver uma ULA (alu) de 1 bit. Abaixo podemos ver uma ilustração deste modulo.

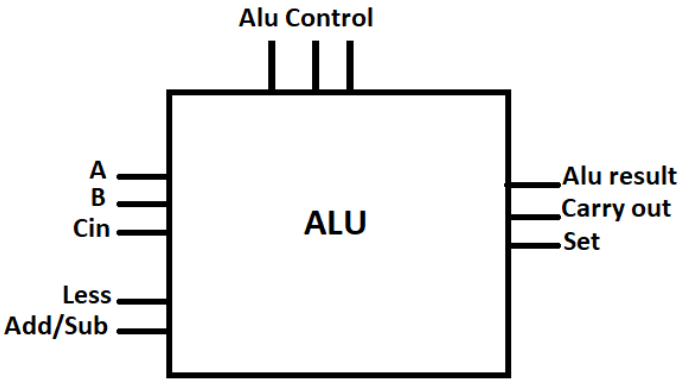

Figura 1 ULA - Diagrama

2.1 Goldem model

Abaixo segue o trecho do código utilizado para gerar os *goldem vectors.*

```
96    for(int i = 0; i < NUM_BITS_CONTROL; i++){
97        ALUControls[8].reset(i);
98    }
99
100   for(int i = 8; i < 12; i++){
101       ALUControls[i] = ALUControls[8];
102       carryIn[i].reset(NUM_BITS - 1);
103   }
104
105   for(int i = 8; i < 12; i++){
106       bitset<NUM_INPUTS> bitsetAux(i);
107
108       for(int j = 0; j < NUM_INPUTS; j++){
109           if(!j && bitsetAux.test(j))
110               SrcsB[i].set(NUM_BITS - 1);
111           else if(j==1 && bitsetAux.test(j))
112               SrcsA[i].set(NUM_BITS - 1);
113       }
```

Figura 2 ULA - Goldem Model

2.2 Goldem vector

A partir do *goldem model* criado anteriormente, geramos o seguinte vetor de testes. Vamos utilizar 60 vetores de teste onde a cada 8 linhas testamos uma posição diferente do multiplexador de 8 bits para assim podermos abranger todas as possibilidades das entradas A, B e Cin em cada operação (menos a operação less que possui apenas quatro testes). Da esquerda para a direita temos os três bits de controle da ula, os três bits de entrada, o bit de seleção da operação, o less e as três saídas que são set, resultado da operação e carry. É valido observar que os espaços entres as linhas mostradas abaixo são apenas para melhor visualização do que foi feito.

```
000_000_0_0_000
000_001_0_0_100
000_010_0_0_100
000_011_0_0_001
000_100_0_0_100
000_101_0_0_001
000_110_0_0_011
000_111_0_0_111

101_000_0_0_010
101_001_0_0_110
101_010_0_0_110
101_011_0_0_011
101_100_0_0_110
101_101_0_0_011
101_110_0_0_001
101_111_0_0_101

001_000_0_0_000
001_001_0_0_100
001_010_0_0_110
001_011_0_0_011
001_100_0_0_110
001_101_0_0_011
001_110_0_0_011
001_111_0_0_111
```

Figura 3 ULA - Goldem vectors

2.3 Testbench

Abaixo podemos ver o código em SystemVerilog responsável por testar a ula.

```
1   `timescale 1ns/100ps
2
3   module alu_tb;
4
5   int counter, errors, aux_error;
6   logic clk,rst;
7   logic [2:0]ALUcontrol;
8   logic SrcA, SrcB, cin, addSubSignal, less;
9   logic set, set_esperado;
10  logic ALUresult, ALUresult_esperado;
11  logic cout, cout_esperado;
12  logic [11:0]vectors[60];
13
14  alu dut(ALUcontrol, SrcA, SrcB, cin, addSubSignal, less, set, ALUresult, cout);
15
16  initial begin
17      $display("Iniciando Testbench");
18      $display("| ALUcontrol | SrcA | SrcB | cin | addSubSignal | less || set | set_esperado || ALUresult | ALUresult_esperado || cout | cout_esperado |");
19      $readmemb("alu_tv.tv",vectors);
20      counter=0; errors=0;
21      rst = 1;
22      #15;
23      rst = 0;
24  end
25
26  always begin
27      clk=1; #10;
28      clk=0; #5;
29  end
30
31  always @(posedge clk)
32      if(~rst)
33      begin
34          //Atribui valores do vetor nas entradas do DUT e nos valores esperados
35          {ALUcontrol, SrcA, SrcB, cin, addSubSignal, less, set_esperado, ALUresult_esperado, cout_esperado} = vectors[counter];
36      end
37
38  always @(negedge clk)   //Sempre (que o clock descer)
```

```
always @(negedge clk)   //Sempre (que o clock descer)
     if(~rst)
     begin
        aux_error = errors;

     assert ((set === set_esperado) && (ALUresult === ALUresult_esperado) && (cout === cout_esperado))

     else  errors = errors + 1;

     if(counter === 0 ) $display("|----------------------------------------------------AND----------------------------------------------------");
     if(counter === 8 ) $display("|----------------------------------------------------NAND---------------------------------------------------");
     if(counter === 16) $display("|----------------------------------------------------OR-----------------------------------------------------");
     if(counter === 24) $display("|----------------------------------------------------NOR----------------------------------------------------");
     if(counter === 32) $display("|----------------------------------------------------XOR----------------------------------------------------");
     if(counter === 40) $display("|----------------------------------------------------SOMA---------------------------------------------------");
     if(counter === 48) $display("|----------------------------------------------------SUB----------------------------------------------------");
     if(counter === 56) $display("|----------------------------------------------------LESS---------------------------------------------------");

     if(aux_error === errors)
        $display("|    %b     |  %b   |  %b   |  %b  |     %b        |  %b   ||  %b  |     %b        ||    %b     |        %b          ||  %b   |
     else
        $display("|    %b     |  %b   |  %b   |  %b  |     %b        |  %b   ||  %b  |     %b        ||    %b     |        %b          ||  %b   |

        counter++;  //Incrementa contador dos vertores de teste

        if(counter == $size(vectors)) //Quando os vetores de teste acabarem
        begin
           $display("Testes Efetuados  = %0d", counter);
           $display("Erros Encontrados = %0d", errors);
           #10
           $stop;
        end

     end

endmodule
```

Figura 4 ULA - Testbench

2.4 Modelo duv

Para o modulo ULA temos oito entras e três saídas. Para isso, basta pegarmos os 30 bits mais significativos e concatenarmos eles com 2 bits zero.

```
module alu(input logic[2:0] ALUcontrol, input logic SrcA, SrcB, cin, addSubSignal, less, output logic set, ALUresult, cout);

   logic B_or_Complement2;
   logic [7:0]out;

   and   andULA(out[0], SrcA, SrcB);
   nand nandULA(out[5], SrcA, SrcB);
   or     orULA(out[1], SrcA, SrcB);
   nor   norULA(out[3], SrcA, SrcB);
   xor   xorULA(out[4], SrcA, SrcB);

   xor addSub(B_or_Complement2, SrcB, addSubSignal);

   somador somadorULA(SrcA, B_or_Complement2, cin, OutputSomador, cout);

   assign out[2] = OutputSomador;
   assign out[6] = OutputSomador;
   assign set = OutputSomador;
   assign out[7] = less;

   mux8 muxULA(out, ALUcontrol, ALUresult);

endmodule
```

Figura 5 ULA - Modelo duv

Também foi utilizado um modulo somador completo como podemos ver abaixo.

```
module meioSomador(input logic in1, in2, output logic out_s0, cout_0);

assign out_s0 = in1 ^ in2;
assign cout_0 = in1 & in2;

endmodule

module somador(input logic inA, inB, cin, output logic out_s, cout);

   logic carry1, carry2, out_s1;

   meioSomador u1(inA, inB, out_s1, carry1);
   meioSomador u2(out_s1, cin, out_s, carry2);
   or u3(cout, carry1, carry2);

endmodule
```

Figura 6 ULA - Somador completo DUV

2.5 Simulação RTL

Com o modulo pronto, podemos iniciar a simulação RTL. Como podemos ver na imagem abaixo, o modulo está se comportando como o esperado. Separando as operações por blocos temos AND e NAND

```
# Iniciando Testbench
# | ALUcontrol | SrcA | SrcB | cin | addSubSignal | less || set | set_esperado || ALUresult | ALUresult_esperado || cout | cout_esperado |
# |-------------------------------------------------------AND-------------------------------------------------------------------------|
# |    000     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# |    000     |  0   |  0   |  1  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    000     |  0   |  1   |  0  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    000     |  0   |  1   |  1  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    000     |  1   |  0   |  0  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    000     |  1   |  0   |  1  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    000     |  1   |  1   |  0  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    000     |  1   |  1   |  1  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  1   |       1       | OK
# |-------------------------------------------------------NAND------------------------------------------------------------------------|
# |    101     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  0   |       0       | OK
# |    101     |  0   |  0   |  1  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    101     |  0   |  1   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    101     |  0   |  1   |  1  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    101     |  1   |  0   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    101     |  1   |  0   |  1  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    101     |  1   |  1   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    101     |  1   |  1   |  1  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  1   |       1       | OK
```

OR, NOR e XOR

```
# |-------------------------------------------------------OR--------------------------------------------------------------------------|
# |    001     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# |    001     |  0   |  0   |  1  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    001     |  0   |  1   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    001     |  0   |  1   |  1  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    001     |  1   |  0   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    001     |  1   |  0   |  1  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    001     |  1   |  1   |  0  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    001     |  1   |  1   |  1  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  1   |       1       | OK
# |-------------------------------------------------------NOR-------------------------------------------------------------------------|
# |    011     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  0   |       0       | OK
# |    011     |  0   |  0   |  1  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    011     |  0   |  1   |  0  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    011     |  0   |  1   |  1  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    011     |  1   |  0   |  0  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    011     |  1   |  0   |  1  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    011     |  1   |  1   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    011     |  1   |  1   |  1  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  1   |       1       | OK
# |-------------------------------------------------------XOR-------------------------------------------------------------------------|
# |    100     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# |    100     |  0   |  0   |  1  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  0   |       0       | OK
# |    100     |  0   |  1   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    100     |  0   |  1   |  1  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    100     |  1   |  0   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    100     |  1   |  0   |  1  |      0       |  0   ||  0  |      0       ||     1     |          1          ||  1   |       1       | OK
# |    100     |  1   |  1   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    100     |  1   |  1   |  1  |      0       |  0   ||  1  |      1       ||     0     |          0          ||  1   |       1       | OK
```

Soma, Subtração e Less

```
# |-------------------------------------------------------SOMA------------------------------------------------------------------------|
# |    010     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# |    010     |  0   |  0   |  1  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    010     |  0   |  1   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    010     |  0   |  1   |  1  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    010     |  1   |  0   |  0  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    010     |  1   |  0   |  1  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    010     |  1   |  1   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    010     |  1   |  1   |  1  |      0       |  0   ||  1  |      1       ||     1     |          1          ||  1   |       1       | OK
# |-------------------------------------------------------SUB-------------------------------------------------------------------------|
# |    110     |  0   |  0   |  0  |      1       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    110     |  0   |  0   |  1  |      1       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    110     |  0   |  1   |  0  |      1       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# |    110     |  0   |  1   |  1  |      1       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    110     |  1   |  0   |  0  |      1       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |    110     |  1   |  0   |  1  |      1       |  0   ||  1  |      1       ||     1     |          1          ||  1   |       1       | OK
# |    110     |  1   |  1   |  0  |      1       |  0   ||  1  |      1       ||     1     |          1          ||  0   |       0       | OK
# |    110     |  1   |  1   |  1  |      1       |  0   ||  0  |      0       ||     0     |          0          ||  1   |       1       | OK
# |-------------------------------------------------------LESS------------------------------------------------------------------------|
# |    111     |  0   |  0   |  0  |      0       |  1   ||  0  |      0       ||     1     |          1          ||  0   |       0       | OK
# |    111     |  0   |  0   |  1  |      0       |  1   ||  0  |      0       ||     1     |          1          ||  0   |       0       | OK
# |    111     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# |    111     |  0   |  0   |  0  |      0       |  0   ||  0  |      0       ||     0     |          0          ||  0   |       0       | OK
# Testes Efetuados  = 60
# Erros Encontrados = 0
```

Figura 7 ULA - Transcript da simulação RTL

Também podemos ver o *RTL view* gerado a partir do modulo testado.

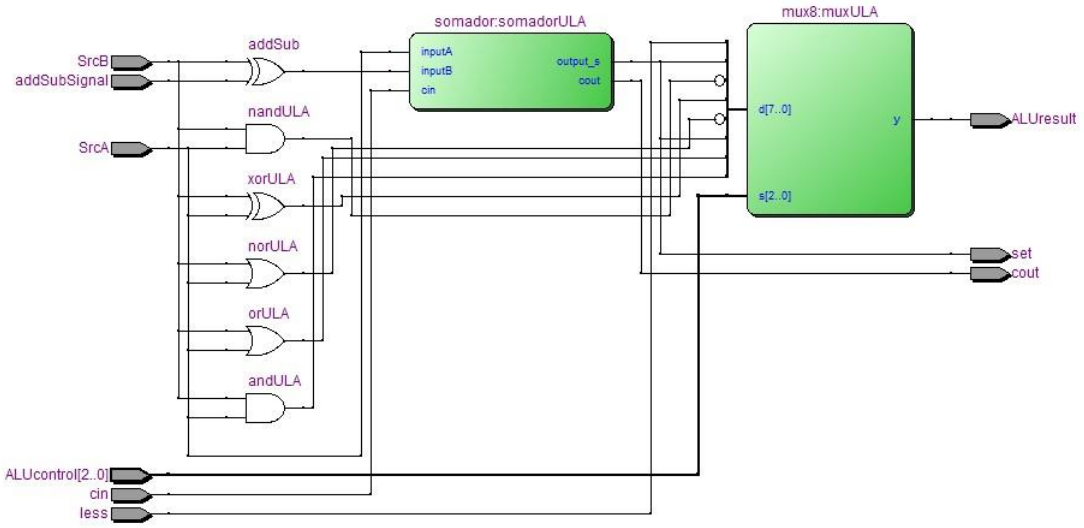

Figura 8 ULA - RTL view

2.6 Simulação *Gate Level*

Agora iniciamos a simulação Gate Level. Iniciamos o clock com um valor consideravelmente baixo e vamos subindo até que nenhum erro seja encontrado. Para este modulo, foram feitos dois testes. É possível observar que à medida que o clock aumenta, o número de erros vai diminuindo. Como foram feitos muito testes, abaixo esta apenas um trecho do transcript, porem podemos ver o número de testes efetuados e o número e erros.

```
# |-------------------------------------------------------------------------SOMA-------------------------------------------------------------------------|
# |    010    | 0 | 0 | 0 |    0    | 0 || 1 |    0    || 1 |    0    || 0 |    0    | ERRO
# |    010    | 0 | 0 | 1 |    0    | 0 || 0 |    1    || 0 |    1    || 0 |    0    | ERRO
# |    010    | 0 | 1 | 0 |    0    | 0 || 0 |    1    || 0 |    1    || 1 |    0    | ERRO
# |    010    | 0 | 1 | 1 |    0    | 0 || 1 |    0    || 1 |    0    || 0 |    1    | ERRO
# |    010    | 1 | 0 | 0 |    0    | 0 || 0 |    1    || 0 |    1    || 1 |    0    | ERRO
# |    010    | 1 | 0 | 1 |    0    | 0 || 1 |    0    || 1 |    0    || 0 |    1    | ERRO
# |    010    | 1 | 1 | 0 |    0    | 0 || 1 |    0    || 1 |    0    || 1 |    1    | ERRO
# |    010    | 1 | 1 | 1 |    0    | 0 || 0 |    1    || 0 |    1    || 1 |    1    | ERRO
# |-------------------------------------------------------------------------SUB--------------------------------------------------------------------------|
# |    110    | 0 | 0 | 0 |    1    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    110    | 0 | 0 | 1 |    1    | 0 || 1 |    0    || 1 |    0    || 0 |    1    | ERRO
# |    110    | 0 | 1 | 0 |    1    | 0 || 1 |    0    || 1 |    0    || 0 |    0    | ERRO
# |    110    | 0 | 1 | 1 |    1    | 0 || 0 |    1    || 0 |    1    || 0 |    0    | ERRO
# |    110    | 1 | 0 | 0 |    1    | 0 || 1 |    0    || 1 |    0    || 1 |    1    | ERRO
# |    110    | 1 | 0 | 1 |    1    | 0 || 0 |    1    || 0 |    1    || 1 |    1    | ERRO
# |    110    | 1 | 1 | 0 |    1    | 0 || 0 |    1    || 0 |    1    || 1 |    0    | ERRO
# |    110    | 1 | 1 | 1 |    1    | 0 || 1 |    0    || 1 |    0    || 0 |    1    | ERRO
# |-------------------------------------------------------------------------LESS-------------------------------------------------------------------------|
# |    111    | 0 | 0 | 0 |    0    | 1 || 0 |    0    || 0 |    1    || 1 |    0    | ERRO
# |    111    | 0 | 0 | 0 |    0    | 1 || 0 |    0    || 1 |    1    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 1 |    0    || 0 |    0    | ERRO
# |    111    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# Testes Efetuados  = 60
# Erros Encontrados = 57
```

Figura 9 ULA - clk #8

```
# |-------------------------------------------------------------SOMA-------------------------------------------------------------|
# |    010    | 0 | 0 | 0 |    0    | 0 || 1 |    0    || 1 |    0    || 0 |    0    | ERRO
# |    010    | 0 | 0 | 1 |    0    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    010    | 0 | 1 | 0 |    0    | 0 || 0 |    1    || 1 |    1    || 1 |    0    | ERRO
# |    010    | 0 | 1 | 1 |    0    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    010    | 1 | 0 | 0 |    0    | 0 || 0 |    1    || 1 |    1    || 1 |    0    | ERRO
# |    010    | 1 | 0 | 1 |    0    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    010    | 1 | 1 | 0 |    0    | 0 || 1 |    0    || 1 |    0    || 1 |    1    | ERRO
# |    010    | 1 | 1 | 1 |    0    | 0 || 1 |    1    || 1 |    1    || 1 |    1    | OK
# |-------------------------------------------------------------SUB--------------------------------------------------------------|
# |    110    | 0 | 0 | 0 |    1    | 0 || 0 |    1    || 1 |    1    || 1 |    0    | ERRO
# |    110    | 0 | 0 | 1 |    1    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    110    | 0 | 1 | 0 |    1    | 0 || 1 |    0    || 1 |    0    || 0 |    0    | ERRO
# |    110    | 0 | 1 | 1 |    1    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    110    | 1 | 0 | 0 |    1    | 0 || 1 |    0    || 1 |    0    || 1 |    1    | ERRO
# |    110    | 1 | 0 | 1 |    1    | 0 || 1 |    1    || 1 |    1    || 1 |    1    | OK
# |    110    | 1 | 1 | 0 |    1    | 0 || 0 |    1    || 1 |    1    || 1 |    0    | ERRO
# |    110    | 1 | 1 | 1 |    1    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |--------------------------------------------------------------LESS--------------------------------------------------------------|
# |    111    | 0 | 0 | 0 |    0    | 1 || 0 |    0    || 1 |    1    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 1 || 0 |    0    || 1 |    1    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# Testes Efetuados  = 60
# Erros Encontrados = 28
```

Figura 10 ULA - clk #9

```
# |-------------------------------------------------------------SOMA-------------------------------------------------------------|
# |    010    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# |    010    | 0 | 0 | 1 |    0    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    010    | 0 | 1 | 0 |    0    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    010    | 0 | 1 | 1 |    0    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    010    | 1 | 0 | 0 |    0    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    010    | 1 | 0 | 1 |    0    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    010    | 1 | 1 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    010    | 1 | 1 | 1 |    0    | 0 || 1 |    1    || 1 |    1    || 1 |    1    | OK
# |-------------------------------------------------------------SUB--------------------------------------------------------------|
# |    110    | 0 | 0 | 0 |    1    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    110    | 0 | 0 | 1 |    1    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    110    | 0 | 1 | 0 |    1    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# |    110    | 0 | 1 | 1 |    1    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    110    | 1 | 0 | 0 |    1    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |    110    | 1 | 0 | 1 |    1    | 0 || 1 |    1    || 1 |    1    || 1 |    1    | OK
# |    110    | 1 | 1 | 0 |    1    | 0 || 1 |    1    || 1 |    1    || 0 |    0    | OK
# |    110    | 1 | 1 | 1 |    1    | 0 || 0 |    0    || 0 |    0    || 1 |    1    | OK
# |--------------------------------------------------------------LESS--------------------------------------------------------------|
# |    111    | 0 | 0 | 0 |    0    | 1 || 0 |    0    || 1 |    1    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 1 || 0 |    0    || 1 |    1    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# |    111    | 0 | 0 | 0 |    0    | 0 || 0 |    0    || 0 |    0    || 0 |    0    | OK
# Testes Efetuados  = 60
# Erros Encontrados = 0
```

Figura 11 ULA - clk #10

Por fim, podemos ver os sinais de entrada e saída do modulo.

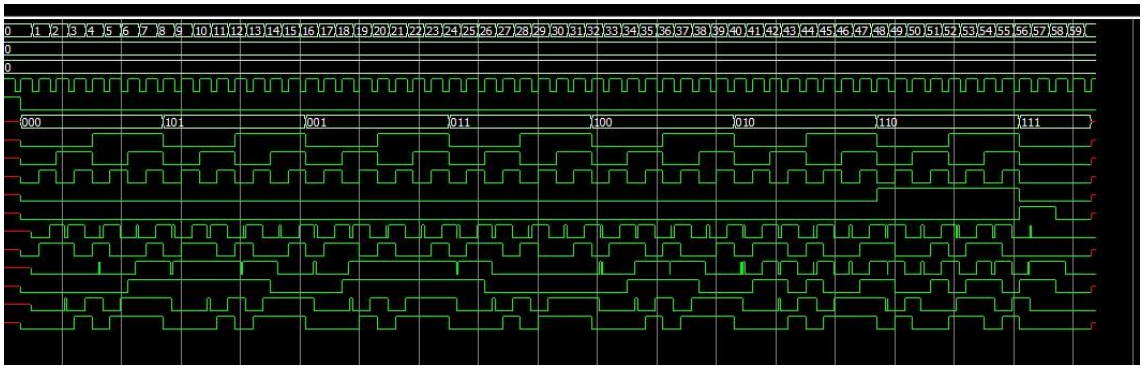

Figura 12 ULA - SInais de entrada e saida