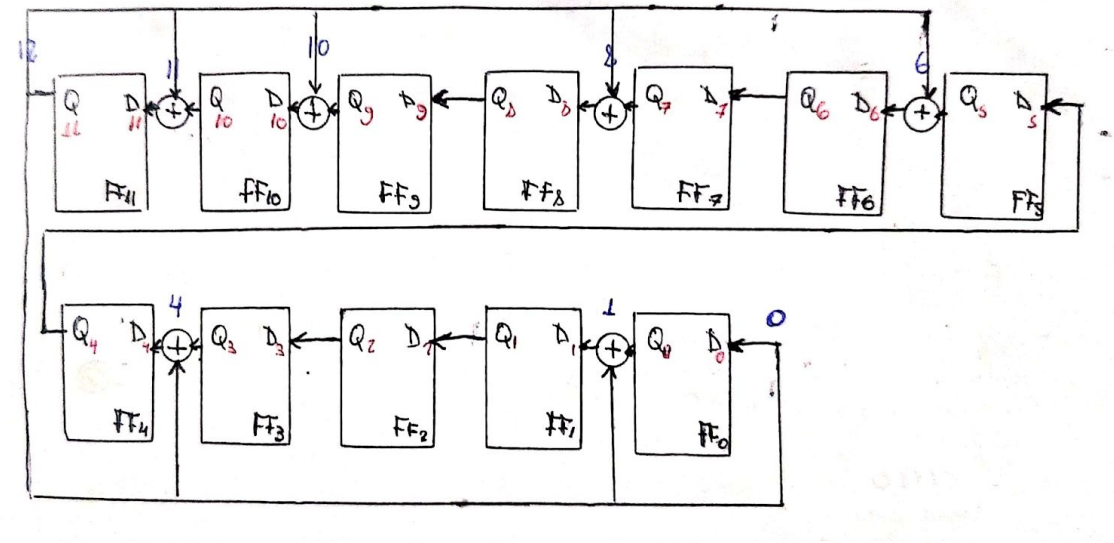
Relatório 1 do projeto

Vinícius de Barros Silva

Para obter o polinômio que designa o funcionamento do LFSR-Galois faz-se a conta (NºUSP mod 2048), no caso o número USP é 10335913, portanto o resultado é 1705, em binário 0110 1010 1001, o que resulta no polinômio $x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^4 + x + 1$.

A estrutura do circuito LFSR é



A descrição em VHDL do projeto é dada por :

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.NUMERIC_STD.all;
4
5
6  entity lfsr is
7
8      generic
9      (
10     WIDTH   : natural := 12
11     );
12     port
13     (
14     Q               : out STD_LOGIC_VECTOR (WIDTH-1
15     clk             : in STD_LOGIC;
16     rst             : in STD_LOGIC
17     );
18
19 end lfsr;
20
21 architecture estrutural of lfsr is
22
23     COMPONENT d_reg
24     port
25     (
26     clk  : in  STD_LOGIC;
27     load : in  STD_LOGIC;
28     d    : in  STD_LOGIC;
29     q    : out STD_LOGIC
30     );
31     END COMPONENT;
32
33 signal q_s  : STD_LOGIC_VECTOR (WIDTH-1 downto 0);
34 signal d_s  : STD_LOGIC_VECTOR (WIDTH-1 downto 0);
35
36 begin
37
38
39 -- for generate para criar lfsr
40 R1 : FOR N IN WIDTH-1 downto 0 GENERATE
```

```
-- instanciando o lfsr
            D0 :d_reg
            PORT MAP(clk,'1',d_s(N),q_s(N));

END GENERATE R1;

-- Mapeando a saida e entrada de cada d_reg

-- Para o Dreg 0
         d_s(0) <= q_s(11) or rst ;
         Q(0) <= q_s(0);

-- Para os FFs com xor
        d_s(1) <= ((q_s(0) xor q_s(11)) or rst);
        Q(1) <= q_s(1);

        d_s(4) <= ((q_s(3) xor q_s(11)) or rst);
        Q(4) <= q_s(4);

        d_s(6) <= ((q_s(5) xor q_s(11)) or rst);
        Q(6) <= q_s(6);

        d_s(8) <= ((q_s(7) xor q_s(11)) or rst);
        Q(8) <= q_s(8);

        d_s(10) <= ((q_s(9) xor q_s(11)) or rst);
        Q(10) <= q_s(10);

        d_s(11) <= ((q_s(10) xor q_s(11)) or rst);
        Q(11) <= q_s(11);
-- Para os FFs sem xor
        d_s(2) <=  (q_s(1) or rst) ;
        Q(2) <= q_s(2);

        d_s(3) <=  (q_s(2) or rst) ;
        Q(3) <= q_s(3);

        d_s(5) <=  (q_s(4) or rst) ;
        Q(5) <= q_s(5);

        d_s(7) <=  (q_s(6) or rst) ;
        Q(7) <= q_s(7);

        d_s(9) <=  (q_s(8) or rst) ;
        Q(9) <= q_s(9);

end architecture estrutural;
```

Segue o código do testbench :

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity tb_rand_num is

GENERIC (
    WIDTH: natural :=12
);

end tb_rand_num;

architecture test of tb_rand_num is

    component rand_num
     generic
    (
    WIDTH   : natural := 12
    );
    port
    (
    clk  : in  STD_LOGIC;
    rst: in  STD_LOGIC;
    Q0    : out  STD_LOGIC;
    Q1   : out STD_LOGIC
    );


    end component;

    component stimuli_rand_num is
        generic
    (
    WIDTH   : natural := 12
    );
    port
    (
    clk             : out STD_LOGIC;
    rst             : out STD_LOGIC
```

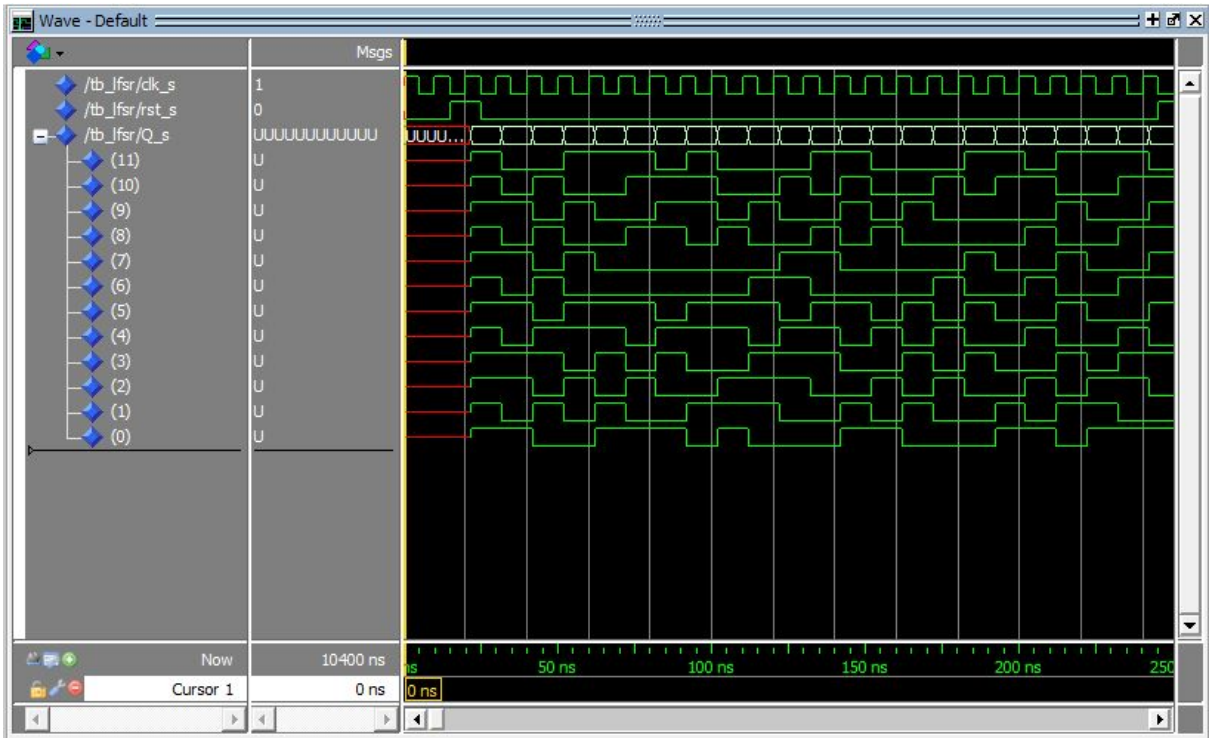
```
41      );
42      end component;
43
44  -- sinais
45  signal clk_s, rst_s : STD_LOGIC;
46  signal Q0_s : STD_LOGIC;
47  signal Q1_s : STD_LOGIC;
48
49  begin
50
51      -- Instantiate DUT
52      dut : rand_num
53          port map
54          (
55          clk             => clk_s,
56          rst             => rst_s,
57          Q0              => Q0_s,
58          Q1              => Q1_s
59          );
60
61      -- Instantiate stimuli generation module
62      test : stimuli_rand_num
63          port map
64          (
65          clk             => clk_s,
66          rst             => rst_s
67          );
68
69  end architecture test;
```

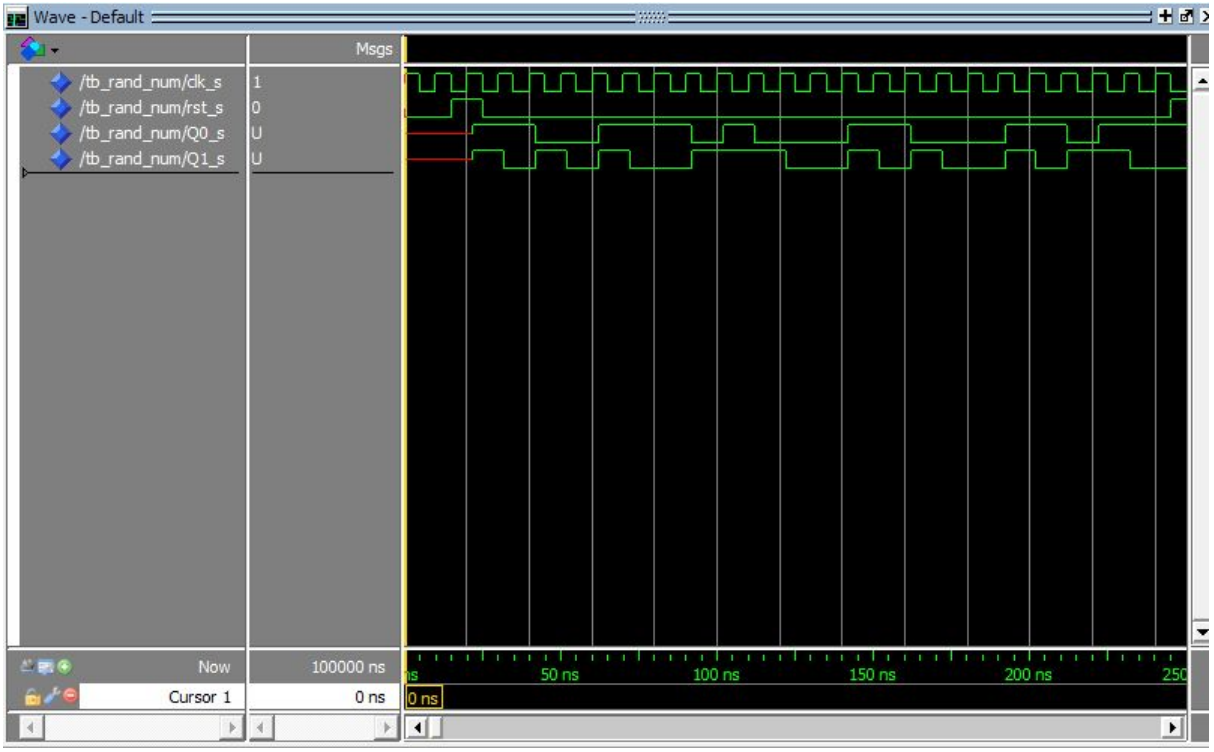
O código para o componente Stimuli segue anexado abaixo :

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- teste para o lsfr galois
entity stimuli_lfsr is
        generic
    (
    CLK_PERIOD      : TIME  := 10ns;
    WIDTH   : natural := 12
    );
    port
    (
    clk             : out STD_LOGIC;
    rst             : out STD_LOGIC
    );

end stimuli_lfsr ;

architecture test of stimuli_lfsr is
    signal clk_s : STD_LOGIC;

    component clock_generator
        generic (
        CLK_PERIOD      : TIME  := 10ns
        );

        port (
        clk : out STD_LOGIC
        );

end component ;

begin


    clk <=clk_s;

    clock: clock_generator
        port map
```

```
41          (
42          clk     => clk_s
43          );
44
45
46      sim : process
47
48          procedure inicia_reset is
49          begin
50              rst <= '0';
51          end procedure inicia_reset;
52
53          procedure reset_activate is    -- reset activation procedure
54          begin
55              wait until falling_edge(CLK_s);
56              rst <= '1';
57              wait for CLK_PERIOD;
58              rst <= '0';
59          end procedure reset_activate;
60
61
62
63      begin
64          -- inicia reset em 0
65          inicia_reset;
66          wait for CLK_PERIOD;
67          -- depois de reset em 0, seta o reset para dar inicio ao funcionamento do lsfr
68          reset_activate;
69          wait for 22*CLK_PERIOD;
70          -- após 22 ciclos reseta denovo
71          reset_activate;
72
73
74
75          wait;
76      end process sim;
77  end architecture test;
```

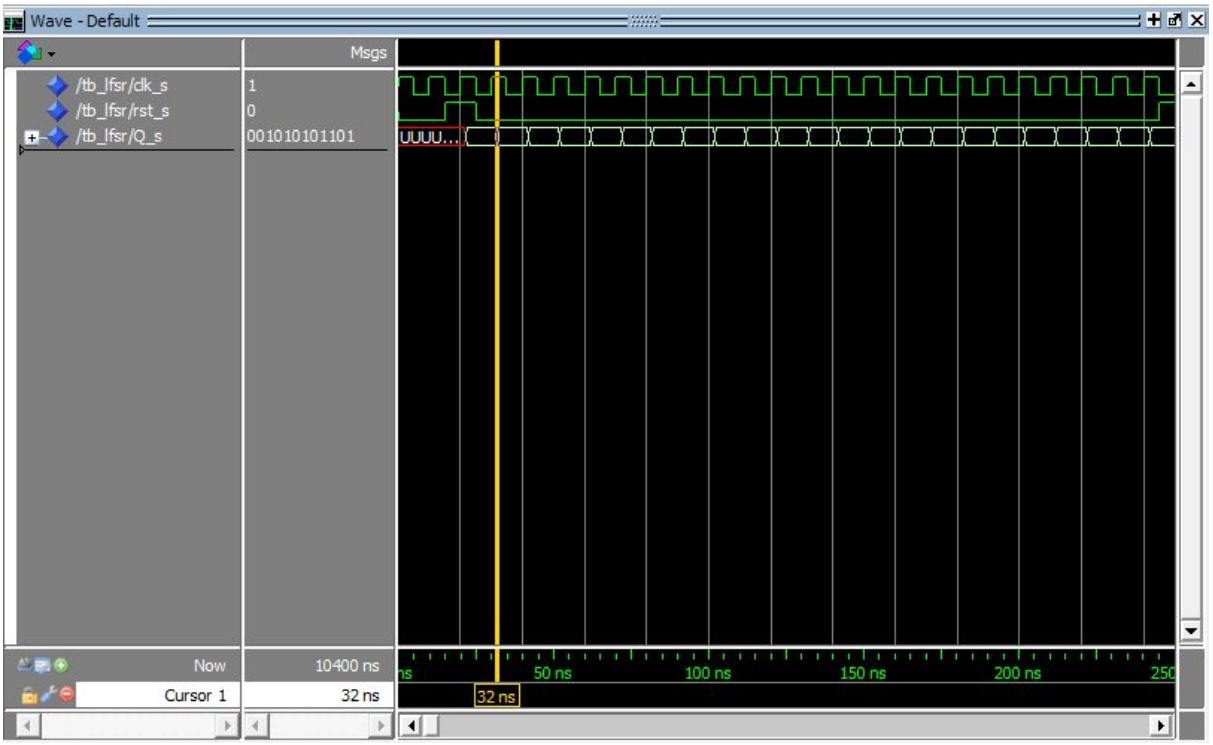
As cartas de tempos referentes ao funcionamento do LFSR e do rand_num são respectivamente :

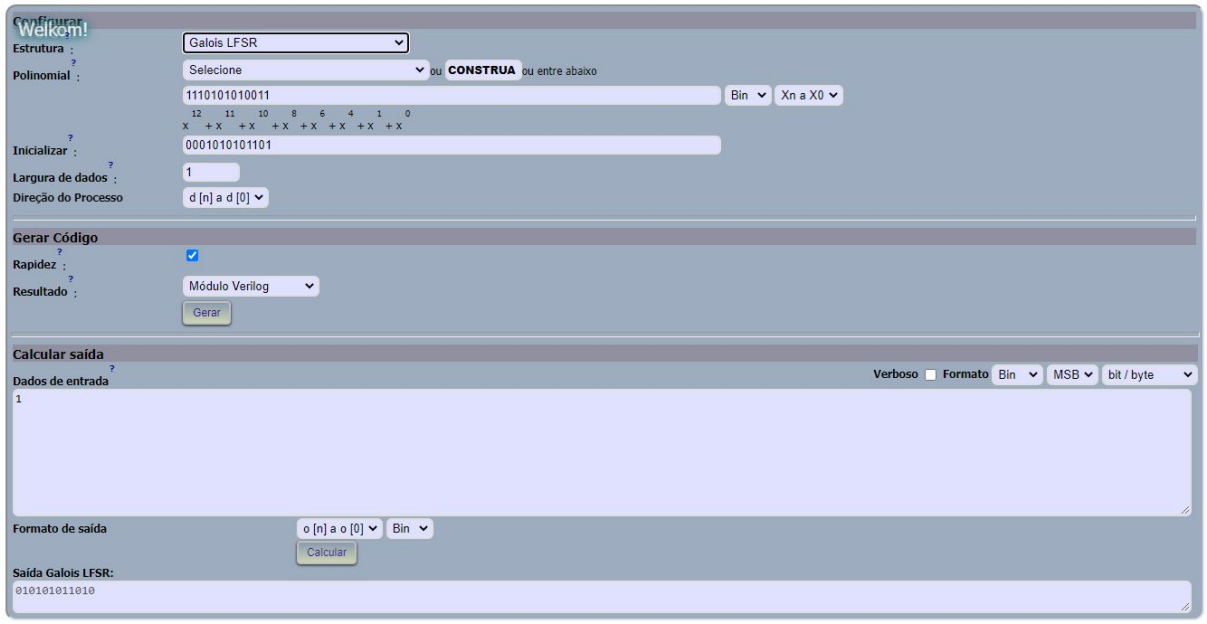


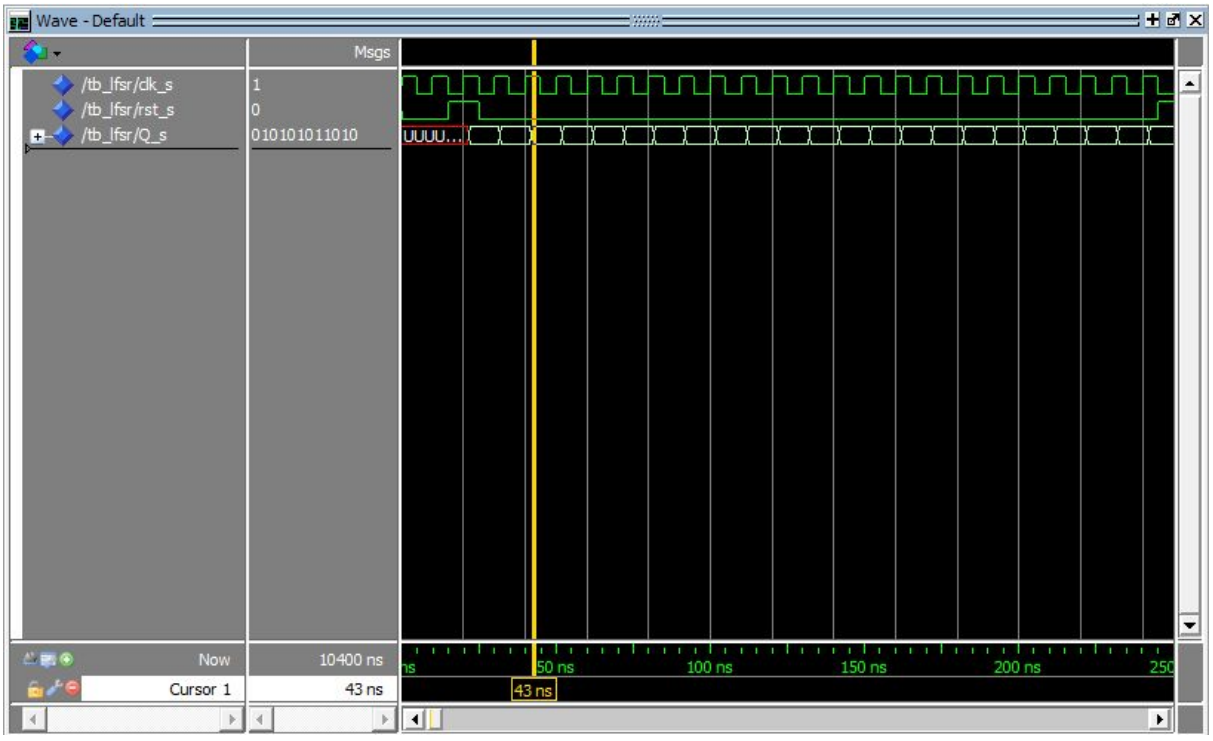


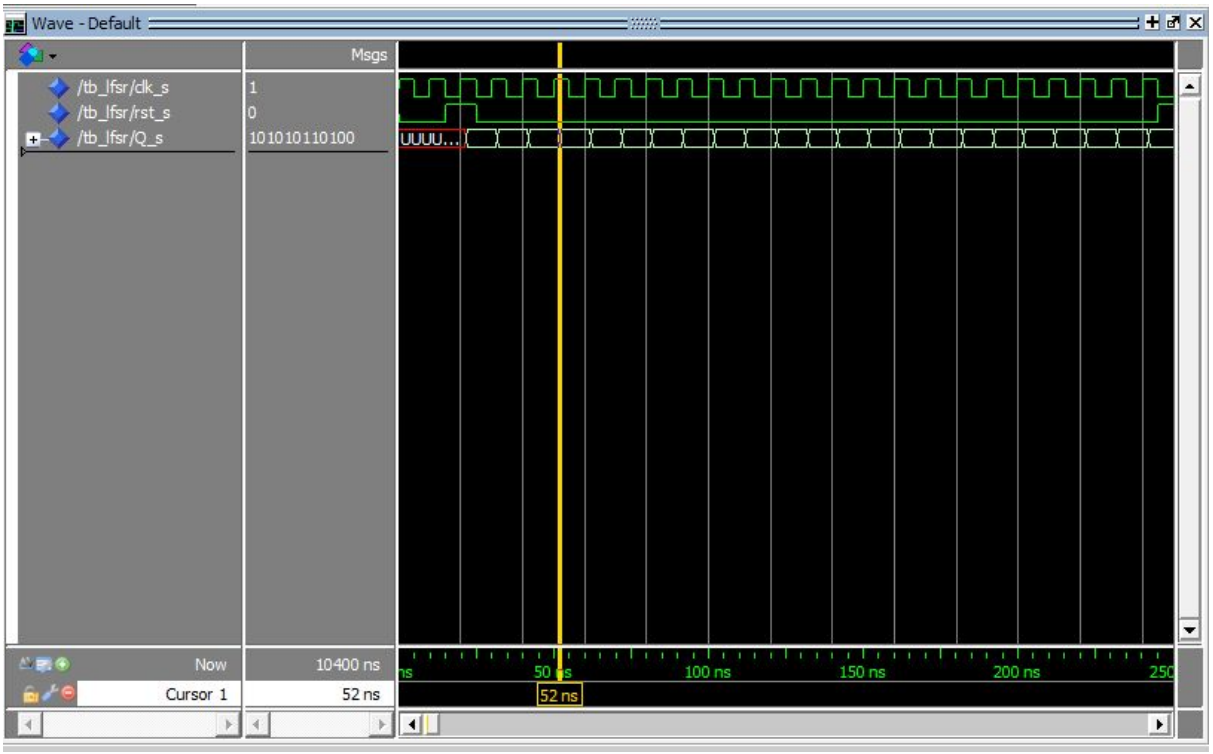
Pode-se observar através da carta de tempos que o funcionamento do LFSR apresentou coerência com o resultado do LFSR gerado pelo site (https://leventozturk.com/engineering/crc/).
Para facilitar a visualização, seguem anexadas 3 interações com a identificação da saída Q:


Hoşgeldiniz!
Estrutura :
Galois LFSR
Polinomial :
Selecione
ou
CONSTRUA
ou entre abaixo
1110101010011
Bin
Xn a X0
12 11 10 8 6 4 1 0
X + X + X + X + X + X + X + X
Inicializar :
0111111111111
Largura de dados :
1
Direção do Processo
d [n] a d [0]
Gerar código
Rapidez :
Resultado :
Módulo Verilog
Gerar
Calcular saída
Dados de entrada
Verbose
Formato
Bin
MSB
bit / byte
1
Formato de saída
o [n] a o [0]
Bin
Calcular
Saída Galois LFSR:
001010101101


Wave - Default
Msgs
/tb_lfsr/clk_s
/tb_lfsr/rst_s
/tb_lfsr/Q_s
1
0
001010101101
UUUU...
Now
10400 ns
Cursor 1
32 ns
50 ns
100 ns
150 ns
200 ns
32 ns


Configurar
Welkom!
Estrutura :
Galois LFSR
Polinomial :
Selecione
ou
CONSTRUA
ou entre abaixo
1110101010011
Bin
Xn a X0
12 11 10 8 6 4 1 0
x + x + x + x + x + x + x + x
Inicializar :
0001010101101
Largura de dados :
1
Direção do Processo
d [n] a d [0]
Gerar Código
Rapidez :
Resultado :
Módulo Verilog
Gerar
Calcular saída
Dados de entrada
Verboso
Formato
Bin
MSB
bit / byte
1
Formato de saída
o [n] a o [0]
Bin
Calcular
Saída Galois LFSR:
010101011010


Wave - Default
Msgs
/tb_lfsr/clk_s
/tb_lfsr/rst_s
/tb_lfsr/Q_s
1
0
010101011010
UUUU...
Now
10400 ns
Cursor 1
43 ns
50 ns
100 ns
150 ns
200 ns
43 ns



As demais interações do site estão na pasta juntas ao documento pdf aqui presente, como também pode-se encontrar as imagens do Wave e todos os códigos em vhdl utilizados para desenvolver o LFSR e o rand_num.

A Fim de melhorar a aleatoriedade do gerador de números utiliza-se um LFSR de 12 bits para gerar o vetor aleatório de 2 bits.Uma vez que a sequência do LFSR de 12 bits passa por todos números até se repetir novamente, já a sequência menor de dois bits gerada a partir da sequência de 12 bits, repete seus números durante o ciclo da sequência maior . Tendo isso em vista, pode-se dizer que a sequência de 2 bits apresenta um comportamento aparentemente aleatório. Vale a pena notar que a sequência de 2 bits não é puramente

aleatória, já que segue uma outra sequência que depende da sequência pseudo-aleatória de 12 bits, e para ser aleatória precisaria ser gerada a partir de uma medida de algo realmente aleatório na natureza.