PSI3451

RELATÓRIO - Projeto 1 – (RAND_NUM com LFSR)

NOME: Vinícius de Barros Silva

#USP: 10335913

DATA DE ENTREGA: 20/06/2022

NOTA:

Parte I: (anexo 1): _____

Parte II.1 (anexo 2): _____

Parte II.2 (anexo 3): _____

Parte II.3 (anexos 4-5): _____

Parte II.4 (anexos 6-7): _____

TOTAL: **_____**

Instruções para a elaboração do relatório.

O relatório apresenta 2 partes: Parte I para a especificação do LFSR e Parte II para a realização do projeto.

1. As tabelas e quadros devem ser preenchidos nos espaços apropriados e incluídos no corpo do relatório; outros dados deverão ser anexados no final do relatório na ordem em que comparecem neste modelo.
2. Todos os anexos devem ser numerados (a numeração é indicada abaixo).
3. Todos os arquivos, imagens e tabelas anexadas devem **mostrar com clareza as informações solicitadas**
4. Dados relevantes presentes nas imagens devem ser obrigatoriamente destacados. Podem ser usados os seguintes recursos:
    a. INSERIR COMENTÁRIOS EM CÓDIGOS
    b. SUBLINHAR VALORES OU OUTROS RESULTADOS
    c. INDICAR COM SETAS DETALHES RELEVANTES DAS IMAGENS DO WAVE
    d. OUTRO recurso que permita a fácil identificação de resultados relevantes por parte do leitor.

Parte I

Geração do LFSR e simulação por software

(PREENCHER OS CAMPOS ABAIXO)

#USP: 10335913

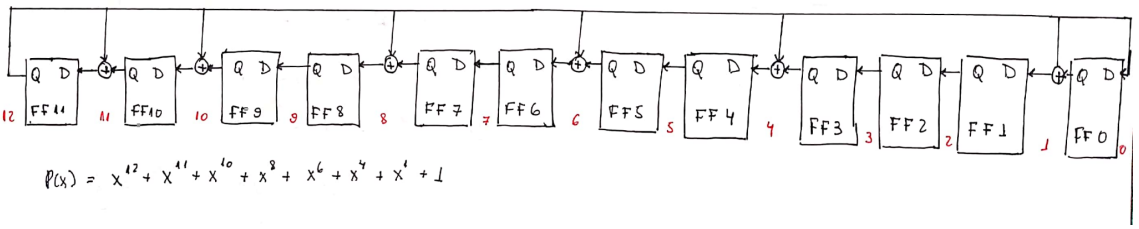
#USP mod 2048 (decimal): 1705

#USP mod 2048 (binário): 110 1010 1001

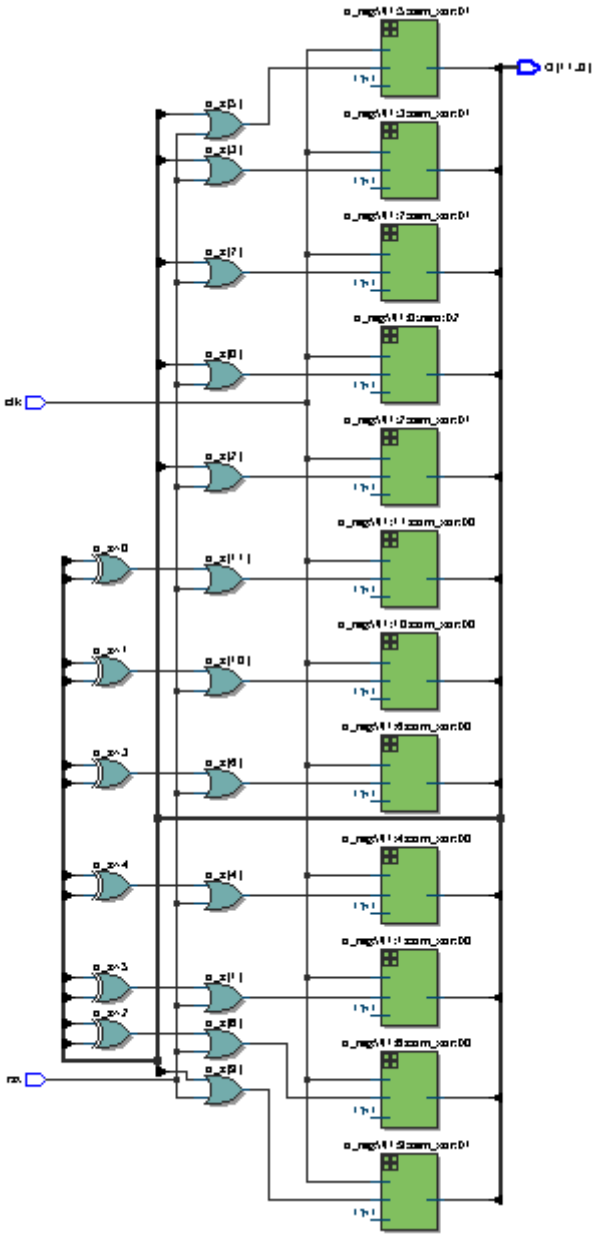
Polinômio característico resultante (INDICAR ATRAVÉS DE UM CÍRCULO AS POTÊNCIAS RELEVANTES, RISCAR AS DEMAIS):

$$p(x) = x^{12} + x^{11} + x^{10} + x^{8} + x^{6} + x^{4} + x^{1} + 1$$

Desenhar o circuito correspondente ao seu polinômio característico



ANEXO 1 (acrescentar no final do relatório): esquema do LFSR. Indicar as células REG, EXOR e OR (veja a figura 2 no final deste texto). Este esquema é o que será capturado em VHDL.



Parte II

1. Resultados das simulações do LFSR pelo software Online CRC BCH Calculator - Code Generator do site (https://leventozturk.com/engineering/crc/)

Execute o software por pelo menos 10 ciclos

ANEXO 2 (acrescentar no final do relatório): Impressão das imagens de tela com os resultados da simulação por software (10 ciclos). Resultados em hexadecimal.

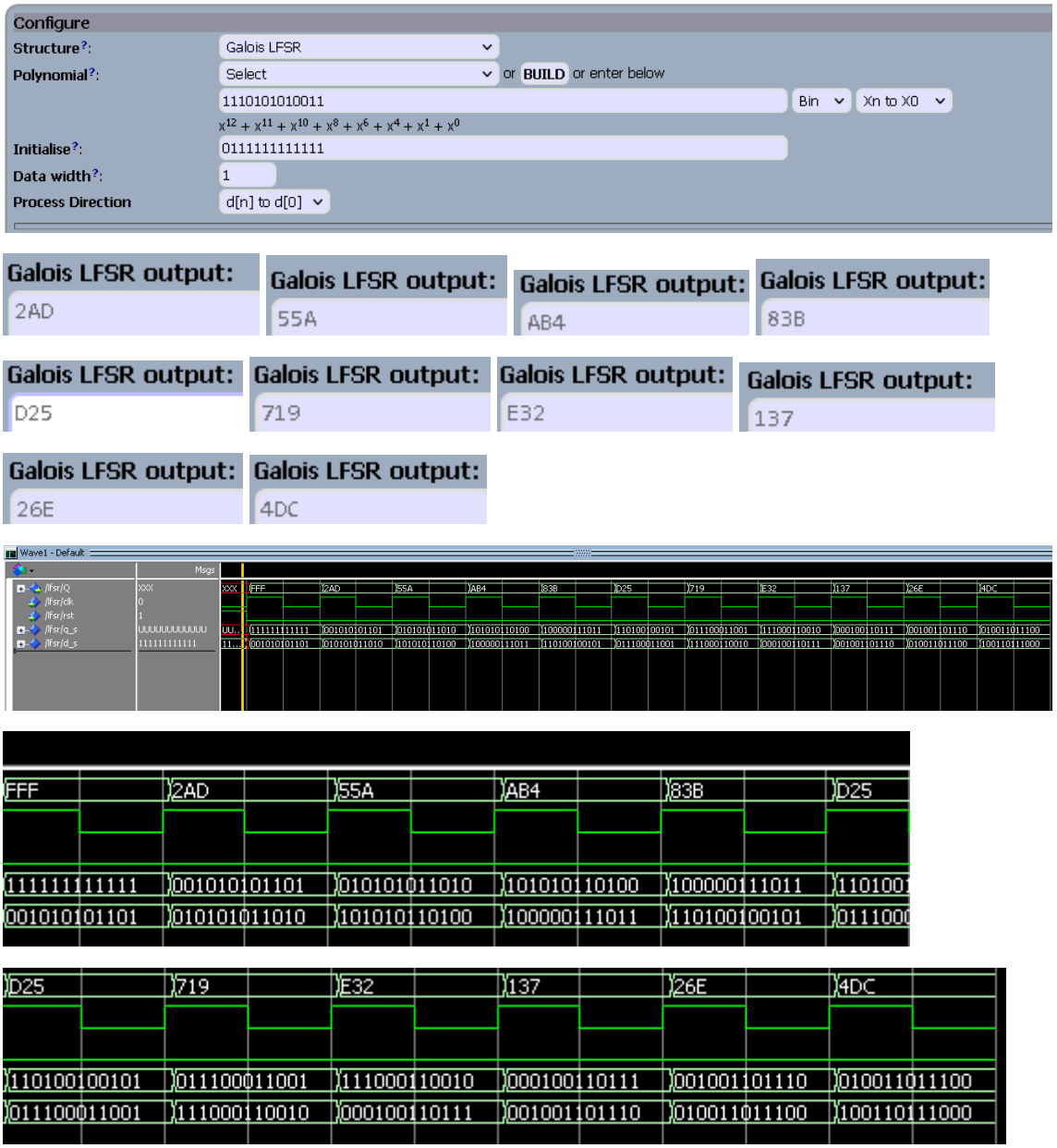


Tabela 1 com os 10 primeiros números gerados pelo software. ATENÇÃO: apresentar os números em hexadecimal.

Incluir no corpo do relatório, tabela 1 contendo os 10 estados codificados em HEXADECIMAL (copiados do software).

Tabela 1: Exemplo de resultados da simulação (10 ciclos)

No. de ciclos a partir da semente	Saída do LFSR (hex) na simulação software
1	2AD
2	55A
3	AB4

4	83B
5	D25
6	719
7	E32
8	137
9	26E
10	4DC

2. **Código VHDL ESTRUTURAL dos módulos RAND_NUM e LFSR (ver figura 1b no final deste texto).**
   (Atenção: lembrar que o código do LFSR deve obrigatoriamente respeitar as seguintes características:
   - o LFSR terá obrigatoriamente 12 FFs (Q_{11} .. Q_0). As saídas (Q_1 e Q_0,) são roteadas para o módulo RAND_NUM (figura 1.b).
   - o modelo VHDL do DFF é o fornecido ao aluno (no site da disciplina).
   - os modelos VHDL das células XOR e OR devem ser copiadas e adaptadas (se for necessário) de módulos utilizados em aulas anteriores.
   - usar obrigatoriamente o comando GENERATE

   Criar os códigos VHDL do RAND_NUM e do LFSR

ANEXO 3 (acrescentar no final do relatório): Descrições do RAND_NUM e do LFSR em VHDL.
ATENÇÃO: ressaltar (sublinhar) as linhas de código do LFSR onde estão indicadas as posições dos taps e as linhas de código do RAND_NUM onde estão indicadas as conexões (2 bits) entre este módulo e o LFSR.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;


entity lfsr is
   port
   (
   Q              : out STD_LOGIC_VECTOR (11 downto 0);
   clk                  : in STD_LOGIC;
   rst                  : in STD_LOGIC
   );

end lfsr;

architecture estrutural of lfsr is

        COMPONENT d_reg
   port
```

```
    (
    clk  : in  STD_LOGIC;
    load : in  STD_LOGIC;
    d    : in  STD_LOGIC;
    q    : out STD_LOGIC
    );
    END COMPONENT;

signal q_s    : STD_LOGIC_VECTOR (11 downto 0);
signal d_s    : STD_LOGIC_VECTOR (11 downto 0);

begin


-- for generate para criar lfsr
R1 : for n in 11 downto 0 generate
-- instanciando o lfsr
                    com_xor: if (n = 11) or (n = 10) or (n=8) or (n=6) or (n=4) or (n=1)
generate
                            D0 :d_reg port map(clk,'1',d_s(n),q_s(n));
                            d_s(n) <= ((q_s(n-1) xor q_s(11)) or rst );
                    end generate com_xor;

                    sem_xor: if (n=9) or (n=7) or (n=5) or (n=3) or (n=3) or (n=2) generate
                            D1 :d_reg port map(clk,'1',d_s(n),q_s(N));
                            d_s(n) <= (q_s(n-1) or rst);
                    end generate sem_xor;

                    zero: if n = 0 generate
                            D2 :d_reg port map(clk,'1',d_s(n),q_s(n));
                            d_s(n) <= (q_s(11) or rst);
                    end generate zero;

end generate R1;

Q <= q_s;

end architecture estrutural;


library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;


entity rand_num is
        generic
    (
    WIDTH    : natural := 8
    );
```

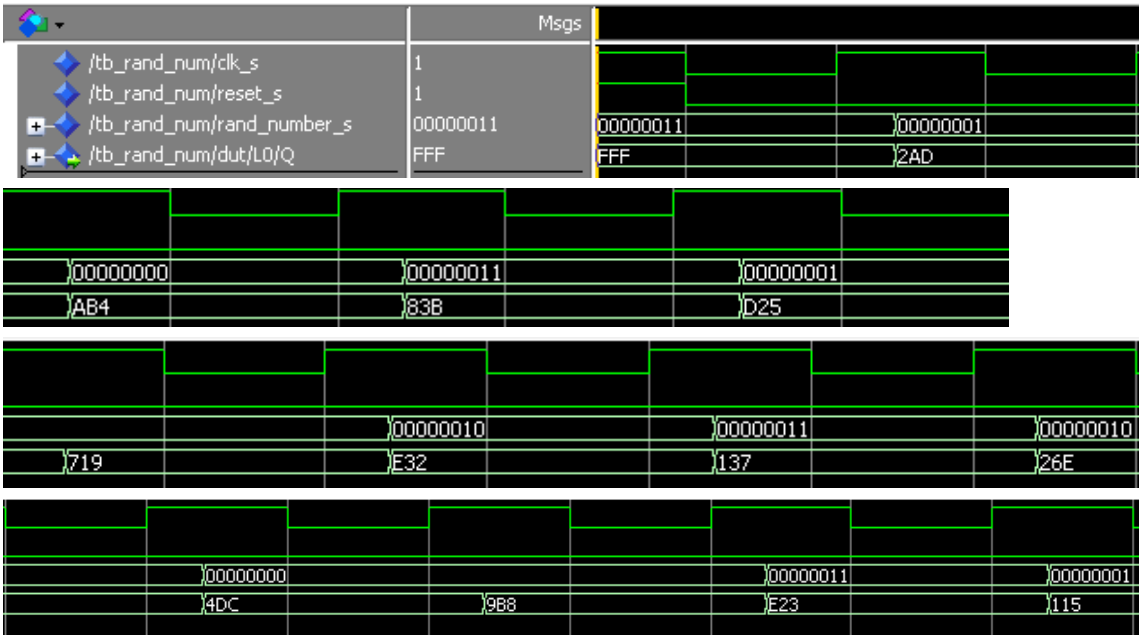
```
   port
   (
   clk  : in  STD_LOGIC;
   rst: in  STD_LOGIC;
   rand_number          : out  STD_LOGIC_VECTOR(WIDTH-1 downto 0)
   );
end rand_num;

architecture structural of rand_num is
-- declara componente lfsr
         COMPONENT lfsr
   port
   (
   Q              : out STD_LOGIC_VECTOR (11 downto 0);
   clk                     : in STD_LOGIC;
   rst                     : in STD_LOGIC
   );
   END COMPONENT;

signal q_s    : STD_LOGIC_VECTOR (11 downto 0);
--signal q2 : std_logic_vector (1 downto 0);

begin
-- liga o sinal q_s ao Q do lfsr
         L0 : lfsr
   PORT MAP(q_s,clk,rst);
-- liga o sinal nas saidas concatenando zeros
   rand_number <=  (  0          => q_s(0),
         1          => q_s(1),
         others => '0');
end structural;
```

3. **Códigos VHDL para o arquivo de estímulos e para o respectivo *testbench* para a simulação do módulo RAND_NUM (lembrando que o LFSR é um sub-módulo) através do ModelSim.**
   (Atenção: lembrar que os estímulos devem obrigatoriamente mostrar (na carta de tempos no Wave) as seguintes situações:
   - A condição inicial do LFSR
   - A sequência das 10 saídas do LFSR (em hexadecimal) demonstrando serem as mesmas obtidas na simulação por software (os resultados devem estar visíveis na figura).
   - A sequência de 10 saídas do módulo RAND_NUM (2 bits)



(imagem wave_rand_num_10_ciclos.png anexada)

Código VHDL do arquivo de estímulos para simulação de RAND_NUM

ANEXO 4 (acrescentar no final do relatório): código do arquivo de estímulos
ATENÇÃO: ressaltar (sublinhar) as linhas de código correspondentes ao estabelecimento da condição inicial e o início da sequência de 10 (ou mais) ciclos. Estas linhas também podem ser identificadas através da inserção de comentários no código.

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

-- teste para o rand_num
entity stimuli_rand_num is
        generic
   (
   CLK_PERIOD       : TIME    := 10ns
   );
   port
   (
   clk                  : out STD_LOGIC;
   rst                  : out STD_LOGIC
   );

end stimuli_rand_num;

architecture test of stimuli_rand_num 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
        (
        clk       => clk_s
        );


   sim : process

        procedure inicia_reset is
        begin
                rst <= '0';
        end procedure inicia_reset;

        procedure reset_activate is     -- reset activation procedure
        begin
                wait until falling_edge(clk_s);
                rst <= '1';
                wait for CLK_PERIOD;
                rst <= '0';
        end procedure reset_activate;



   begin
        -- inicia reset em 0
        inicia_reset;
        wait for CLK_PERIOD;
        -- depois de reset em 0, seta o reset para dar inicio ao funcionamento do
lsfr
        reset_activate;
        wait for 22*CLK_PERIOD;
        -- após 22 ciclos reseta denovo
        reset_activate;
```

```
        wait;
    end process sim;
end architecture test;
```

ANEXO 5 (acrescentar no final do relatório): código do *testbench*
ATENÇÃO: ressaltar (sublinhar) as linhas de código que indicam os componentes presentes no testbench e as ligações entre eles.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.numeric_std.all;


ENTITY tb_rand_num IS
   GENERIC(
            WIDTH          : NATURAL := 8
   );

END tb_rand_num ;

ARCHITECTURE arch OF tb_rand_num IS


   COMPONENT rand_num IS
      GENERIC(
               WIDTH                  : NATURAL  := 8
      );

      port(
               clk, rst          : IN  STD_LOGIC;
               rand_number            : OUT STD_LOGIC_VECTOR(WIDTH-1
downto 0)
            );
   END  COMPONENT;


   COMPONENT stimuli_rand_num IS

      port(
               clk, rst: OUT  STD_LOGIC
            );
   END  COMPONENT;


   SIGNAL clk_s, reset_s            : STD_LOGIC;
   SIGNAL rand_number_s                  :    STD_LOGIC_VECTOR(WIDTH-1
downto 0)  ;
```

```
BEGIN

        dut : rand_num
                                        GENERIC MAP(WIDTH)
                                        PORT MAP(
                                                clk      => clk_s,
                                                rst      => reset_s,
                                                rand_number => rand_number_s
                                        );

        stimuli : stimuli_rand_num

                                        PORT MAP(
                                                clk      => clk_s,
                                                rst      => reset_s
                                        );

end arch;
```

4. Resultados das simulações através do programa ModelSim

Simulação do RAND_NUM mostrando a correta a condição inicial do LFSR

ANEXO 6 (acrescentar no final do relatório): Imagem do WAVE ilustrando a condição inicial do LFSR

ATENÇÃO: a condição deve estar identificada por seta ou círculo.

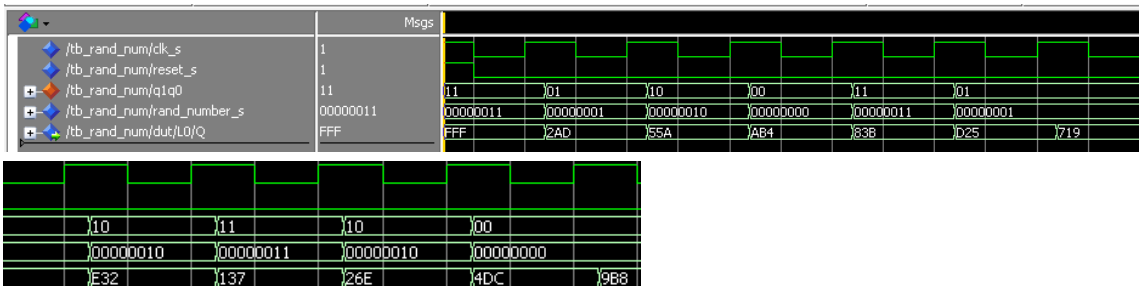
(imagem condicao_inicia_lfsr.png)

Simulação do RAND_NUM mostrando a correta geração da sequência pseudo-aleatória

ANEXO 7 (acrescentar no final do relatório): Imagem do WAVE ilustrando: a) a sequências de 10 saídas do LFSR (os mesmos percorridos durante a simulação com o software); b) a saída de RAND_NUM com os dois bits b1 e b0 destacados.

ATENÇÃO: as saídas do LFRS devem estar identificadas pelos mesmos valores HEXADECIMAIS apresentados no anexo 2 e Tabela 1 (para fácil identificação). Os valores de saída do LFSR e do RAND_NUM deverão estar evidenciadas com setas ou círculos.

(imagem wave_rand_num_destaque_q0q1_10_ciclos.png)



Apresentar, nesta seção, uma tabela como a Tabela 2, de exemplo, com os 10 primeiros números gerados após a semente pelo software e pelo hardware. Use HEX para facilitar comparação.

Tabela 2. Exemplo de tabela a apresentar

No. de ciclos a partir da semente	Saída do LFSR (hex) na simulação software	Saída do LFSR (hex) na simulação hardware
1	2AD	2AD
2	55A	55A
3	AB4	AB4
4	83B	83B
5	D25	D25
6	719	719
7	E32	E32
8	137	137
9	26E	26E
10	4DC	4DC

Observe e comente em um quadro, como no exemplo abaixo:

1) se os resultados das simulações por software e pelo ModelSim foram iguais.
2) Foram resultados esperados? não esperados? por que?

Os resultados para simulação por software e por hardware foram os mesmos. Isso era realmente o comportamento esperado caso o código estivesse certo. O software nos fornece a sequência de números geradas por determinado LFSR de acordo com seu polinômio específico, a implementação desse LFSR utilizando flip flops, xor e or deve ter a mesma sequência.

Apresentar, nesta seção, uma tabela como a Tabela 3, de exemplo, com os 10 primeiros números aleatórios gerados pelo LFSR e os gerados no RAND_NUM. Use HEX para facilitar comparação.

Tabela 3. Exemplo de tabela a apresentar

No. de ciclos a partir da semente	Saída do LFSR (hex)	Saída do Rand_num (com 2 bits: MSB, LSB)
1	2AD	00000001 (01)
2	55A	00000010 (10)
3	AB4	00000000 (00)
4	83B	00000011 (11)
5	D25	00000001 (01)
6	719	00000001 (01)
7	E32	00000010 (10)
8	137	00000011 (11)
9	26E	00000010 (10)
10	4DC	00000000 (00)

Observe e comente em um quadro, como no exemplo abaixo:

1) se a aleatoriedade dos valores de saída do LFSR foi observada na saída do módulo RAND_NUM.
2) compare a pseudo-aleatoriedade do LFSR e dos 2 bits de saída.

Sim, podemos observar em RAND_NUM a aleatoriedade gerada pela saída do LFSR, já que o RAND_NUM é gerado a partir da sequência do LFSR. A sequência gerada pelo LFSR demora a repetir valores dado o número grande de possibilidades para o próximo estado, sendo um número de 12 bits, porém a sequência gerada pelos dois bits extraídos do LFSR tem apenas 4 possibilidades, isso causa um aumento na aleatoriedade, dado ser muito mais provável a repetição das possibilidades.