# **PSI3451**

# **RELATÓRIO - Projeto 1 – (RAND\_NUM com LFSR)**

# **NOME: Gabriel Lujan Bonassi**

# **#USP: 11256816**

**DATA DE ENTREGA: 12/06/2023**

**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 (siga a numeração deste roteiro).
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:
   1. INSERIR COMENTÁRIOS EM CÓDIGOS
   2. SUBLINHAR VALORES OU OUTROS RESULTADOS
   3. INDICAR COM SETAS DETALHES RELEVANTES DAS IMAGENS DO WAVE
   4. OUTRO recurso que permita a fácil identificação de resultados relevantes por parte do leitor.
5. O único upload adicional requerido é o dos arquivos VHDL utilizado na sua simulação Modelsim. ]
6. Edite este relatório de acordo com a sua conveniência; pode apagar as linhas com as instruções se assim desejar, mas mantenha todos os itens e enumeração.

**Parte I**

**Geração do LFSR e simulação por software**

(PREENCHER OS CAMPOS ABAIXO)

#USP: 11256816

#USP mod 2048 (decimal): 1008

#USP mod 2048 (binário com 11 bits): 01111110000

Polinômio característico resultante (APAGAR OS GRAUS NÃO CONDIZENTES AO SEU POLINÔMIO):

x12 + x10 + x9 + x8 + x7 + x6 + x5 + 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**

# **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). **Obrigatório:** 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 | 81F |
| 2 | 7DF |
| 3 | FBE |
| 4 | 89D |
| 5 | 6DB |
| 6 | DB6 |
| 7 | C8D |
| 8 | EFB |
| 9 | A17 |
| 10 | 3CF |

1. **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 (Q11 .. Q0­). As saídas (Q1 e Q0,) 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\_NUMe do LFSR

**ANEXO 3** (acrescentar no final do relatório): Descrições finais do RAND\_NUMe do LFSR em VHDL, utilizadas em sua simulação.

**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.

1. **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 (semente)
* 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)
* 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.

* Código VHDL do arquivo do *testbench* para simulação de RAND\_NUM

**ANEXO 5** (acrescentar no final do relatório): código do arquivo do *testbench*

**ATENÇÃO**: ressaltar (sublinhar) as linhas de código que indicam os componentes presentes no testbench e as ligações entre eles.

1. **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.

* 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 sequência da semente mais as 10 próximas 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 obrigatoriamente 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.

* 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 | 3CF | A14 |
| 2 | A14 | 5FF |
| ... |  |  |
| ... |  |  |
| 10 | 001 | 23F |
|  |  |  |

* 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 não ocorreram como esperava. Esperava-se que... etc

* 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 RAND\_NUM (hex) | Saída do Rand\_num  ( com 2 bits: MSB, LSB) |
| 1 | A14 | 00000000 (00) |
| 2 | 5FF | 00000011 (11) |
| ... |  | 00000000 (00) |
| ... |  |  |
| 10 | 23E | 00000010 (02) |

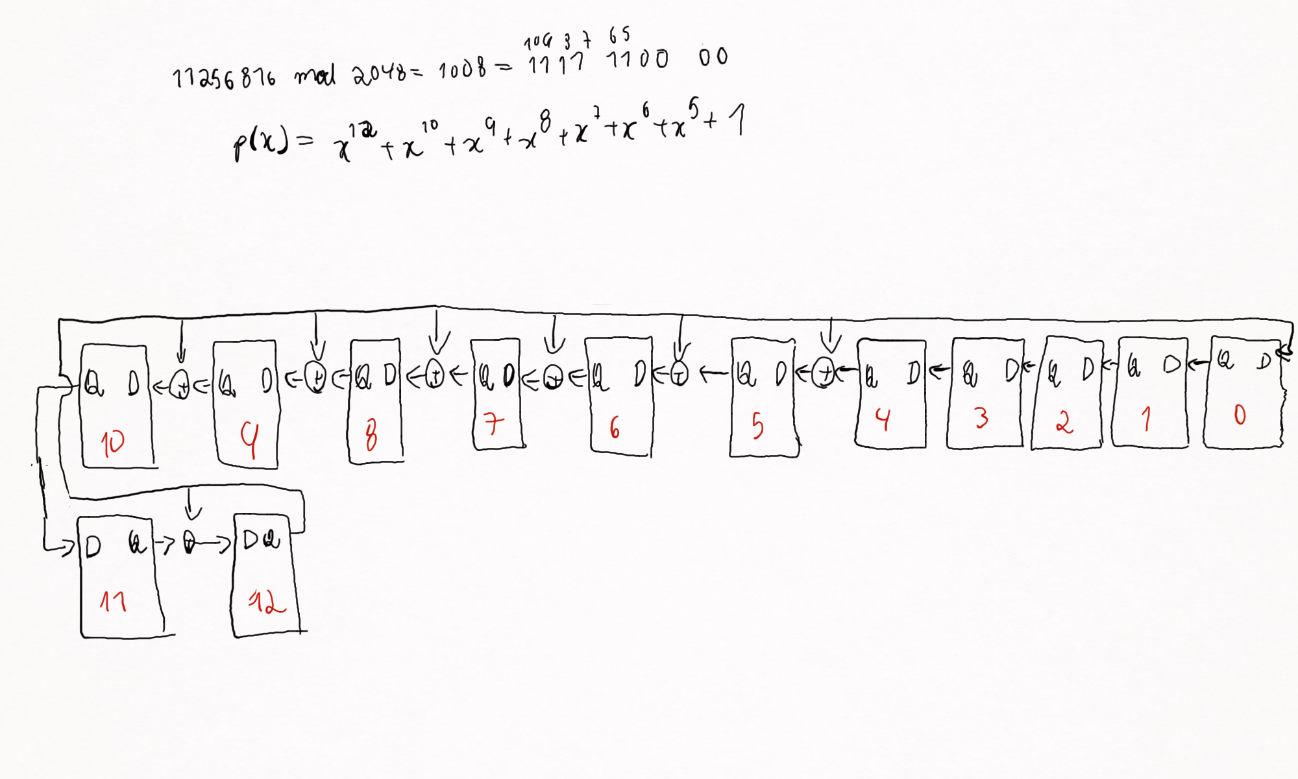
* Observe e comente em um quadro, como a no exemplo abaixo:

1. como os valores gerados pelo LFSR afeta a aleatoriedade dos 2 bits das saídas do módulo RAND\_NUM?
2. Sabendo-se que o LFSR gera uma sequência pseudo-aleatória, como você caracteriza a sequência que aparece à saída do RAND\_NUM?.

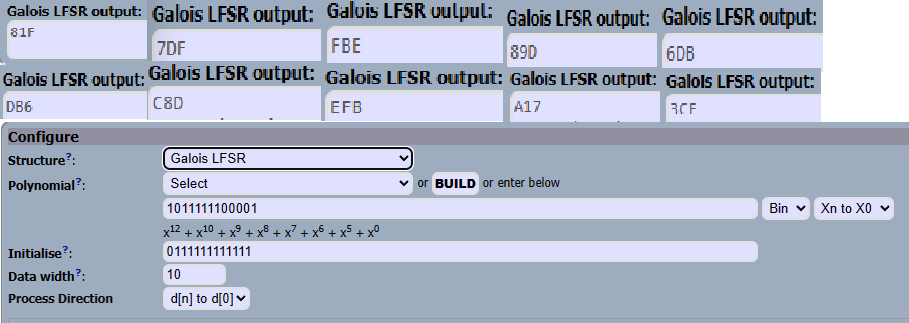
Os 2 bits comportam-se...

**IMPORTANTE: Para validar o seu projeto, além do upload deste relatório em pdf, fazer o mesmo com arquivo zip de todos os arquivos VHDL usados neste projeto.**

**Anexo 1:**

****

**Anexo 2:**

****

**Anexo 3:**

entity lfsr is

    port

    (

        Q : out STD\_LOGIC\_VECTOR (11 downto 0);

        clk : in STD\_LOGIC;

        rst : in STD\_LOGIC

    );

end lfsr;

architecture arch 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 s\_q : STD\_LOGIC\_VECTOR (11 downto 0);

signal s\_d : STD\_LOGIC\_VECTOR (11 downto 0);

begin

R1 : for n in 11 downto 0 generate

            --Posicoes dos taps

            xor: if (n = 10) or (n = 9) or (n=8) or (n=7) or (n=6) or (n=5) or (n=1) generate

                D0 :d\_reg port map(clk,'1',s\_d(n),s\_q(n));

                s\_d(n) <= ((s\_q(n-1) xor s\_q(11)) or rst );

            end generate xor;

            no\_xor: if (n=11) or (n=4) or (n=3) or (n=2) or (n=1) generate

                D1 :d\_reg port map(clk,'1',s\_d(n),s\_q(N));

                s\_d(n) <= (s\_q(n-1) or rst);

            end generate no\_xor;

            zero: if n = 0 generate

                D2 :d\_reg port map(clk,'1',s\_d(n),s\_q(n));

                s\_d(n) <= (s\_q(11) or rst);

            end generate zero;

end generate R1;

Q <= s\_q;

end architecture arch;

entity rand\_num is

    generic

    (

        WIDTH : natural := 8

    );

    port

    (

        clk : in STD\_LOGIC;

        rst : in STD\_LOGIC;

        random\_number : out STD\_LOGIC\_VECTOR(WIDTH-1 downto 0)

    );

end rand\_num;

architecture structural of rand\_num is

    COMPONENT lfsr

    port

    (

        Q : out STD\_LOGIC\_VECTOR (11 downto 0);

        clk : in STD\_LOGIC;

        rst : in STD\_LOGIC

    );

    END COMPONENT;

signal s\_q  : STD\_LOGIC\_VECTOR (11 downto 0);

begin

    L0 : lfsr PORT MAP(s\_q,clk,rst);

    --ligando os modulos

    random\_number <=  (    0      => s\_q(0),

            1      => s\_q(1),

            others => '0');

end structural;

**Anexo 4:**

entity stimuli\_rand\_num is

        generic

    (

        CLK\_PERIOD : TIME := 10ns

    );

    port

    (

        clk : out STD\_LOGIC;

        rst : out STD\_LOGIC

    );

end stimuli\_rand\_num;

architecture arch 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

        );

    stimuli : process

        procedure reset\_begin is

        begin

            rst <= '0';

        end procedure reset\_begin;

        procedure reset\_activate is

        begin

            wait until falling\_edge(clk\_s);

            rst <= '1';

            wait for CLK\_PERIOD;

            rst <= '0';

        end procedure reset\_activate;

    begin

        reset\_begin; --inicia sem reset

        wait for CLK\_PERIOD;

        reset\_activate; --seta o reset para startar o LFSR

        wait for 13\*CLK\_PERIOD; --espera 13 ciclos e reseta

        reset\_activate;

        wait;

    end process stimuli;

end architecture arch;

**Anexo 5:**

ENTITY tb\_rand\_num IS

    GENERIC(

        WIDTH : NATURAL := 8

    );

END tb\_rand\_num ;

ARCHITECTURE arch OF tb\_rand\_num IS

    --componente rand\_num

    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;

    --componente gerador de estimulos

    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;