PSI3451

RELATÓRIO - Projeto 1 – (RAND_NUM com LFSR)

NOME: Gabriel Lujan Bonassi

#USP: 11256816

DATA DE ENTREGA: 14/06/2023

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:
    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.
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):

$x^{12} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + 1$

- 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). **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

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 finais do RAND_NUM e 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.

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 (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.sim:/tb_rand_num

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.

- 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.

sim:/tb_rand_num

- 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 hsim:/tb_rand_numardware
1	81F	81F

2	7DF	7DF
3	FBE	FBE
4	89D	89D
5	6DB	6DB
6	DB6	DB6
7	C8D	C8D
8	EFB	EFB
9	A17	A17
10	3CF	3CF

- 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 foram iguais, o que era esperado, tendo em vista que ambas as ferramentas devem executar o mesmo procedimento.

- 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	81F	00000011 (11)
2	7DF	00000011 (11)
3	FBE	00000010 (10)
4	89D	00000001 (01)
5	6DB	00000011 (11)
6	DB6	00000010 (10)
7	C8D	00000001 (01)
8	EFB	00000011 (11)
9	A17	00000011 (11)
10	3CF	00000011 (11)

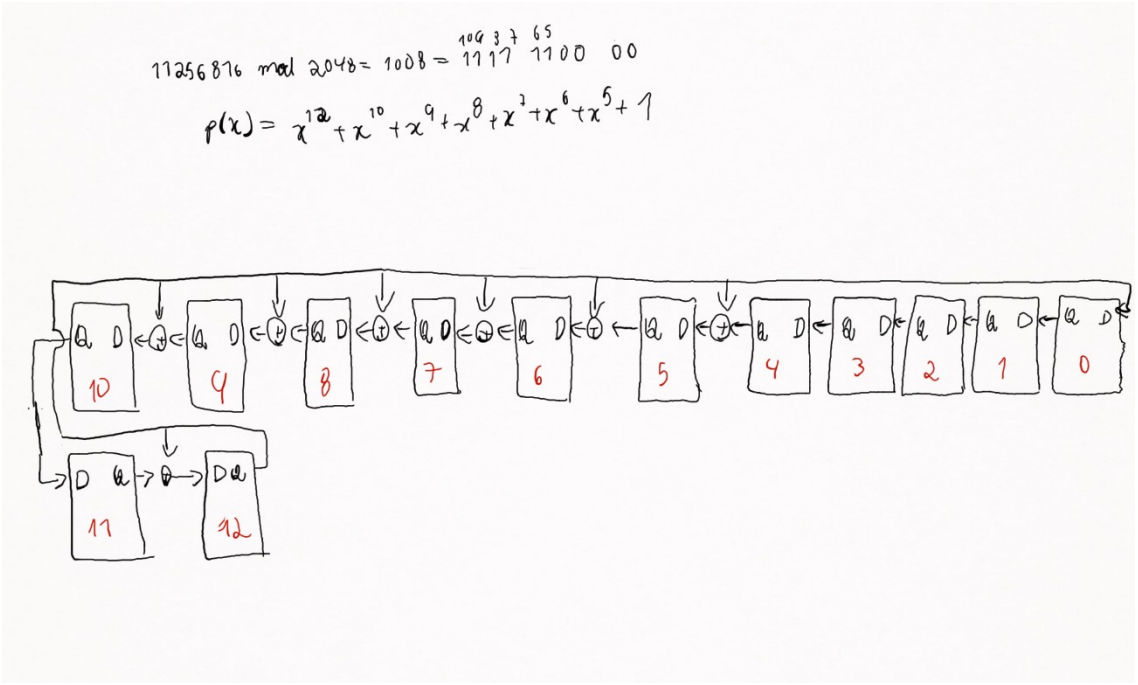
- Observe e comente em um quadro, como a no exemplo abaixo:
  1) como os valores gerados pelo LFSR afeta a aleatoriedade dos 2 bits das saídas do módulo RAND_NUM?

2) Sabendo-se que o LFSR gera uma sequência pseudo-aleatória, como você caracteriza a sequência que aparece à saída do RAND_NUM?.

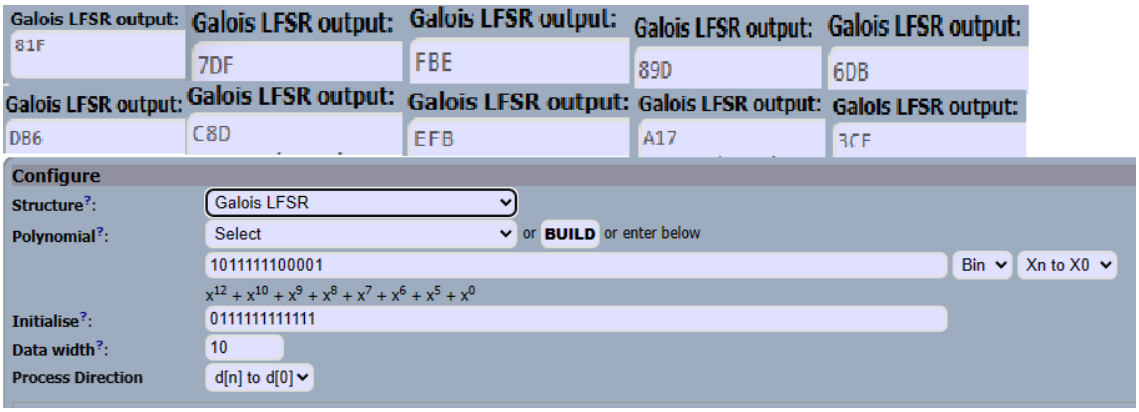
A aleatoriedade dos valores de saída do LFSR de fato foi observada na saída do modulo RAND_NUM, já que o RAND_NUM é gerado a partir da sequência do LFSR.
A sequência que aparece na saída do RAND_NUM é muito mais aleatória que aleatória que a saída do LFSR, pois é muito mais provável a repetição das possibilidades

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)
```

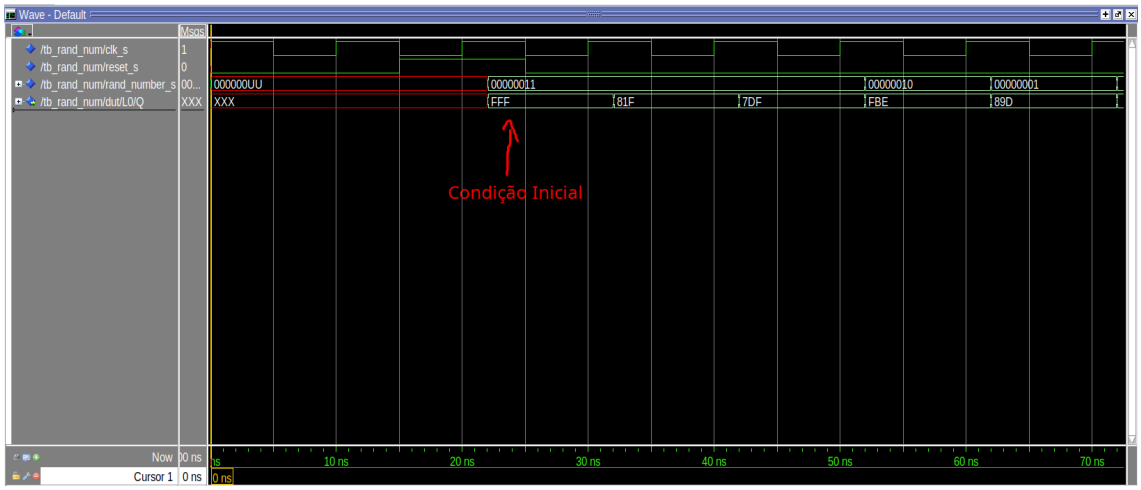
```
            PORT MAP(
                clk => clk_s,
                rst => reset_s,
                rand_number => rand_number_s
            );

        stimuli : stimuli_rand_num

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

end arch;
```

Anexo 6:



Anexo 7: