PSI3451

RELATÓRIO - Projeto 1 – (RAND_NUM) + (LFSR)

NOME: Gabriel Moraes da Cruz_

#USP: 10355020

DATA DE ENTREGA:_

NOTA:

Parte I: _____

Parte II.1 (anexos 1-2-3): _____

Parte II.2 (anexo 4): _____

Parte II.3 (anexos 5-6): _____

Parte II.4 (anexos 7-8-9-10): _____

TOTAL: **_____**

Instruções para a elaboração do relatório.

O relatório apresenta 2 partes.

1. Na parte I os dados devem ser preenchidos nos espaços apropriados.
2. Na parte II os dados devem ser anexados no final do relatório na ordem em que comparecem neste modelo.
3. Todos os anexos devem ser numerados (a numeração é indicada abaixo).
4. Todas os arquivos, imagens e tabelas anexadas devem **mostrar com clareza as informações solicitadas**
5. 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.

OBSERVAÇÃO: por conveniência, no final desta apostila encontra-se uma breve recordação teórica sobre o LFSR que será desenvolvido. Estas informações foram extraídas da apostila de conceitos já disponível no site da disciplina

IMPORTANTE: este projeto desenvolve o modelo do módulo RAND_NUM (gerador de números aleatórios) o qual contém o módulo LFSR conforme ilustra a figura 1b no breve resumo teórico reproduzido no final deste texto.

Parte I

Geração do LFSR e simulação por software

(PREENCHER OS CAMPOS ABAIXO)

#USP: 10335020

#USP mod 2048 (decimal): 812

#USP mod 2048 (binário): 01100101100

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

$\mathbf{x^{12} + x^{10} + x^9 + x^6 + x^4 + x^3 + 1}$

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 1 (acrescentar no final do relatório): Impressão das imagens de tela com os resultados da simulação por software (10 ciclos)

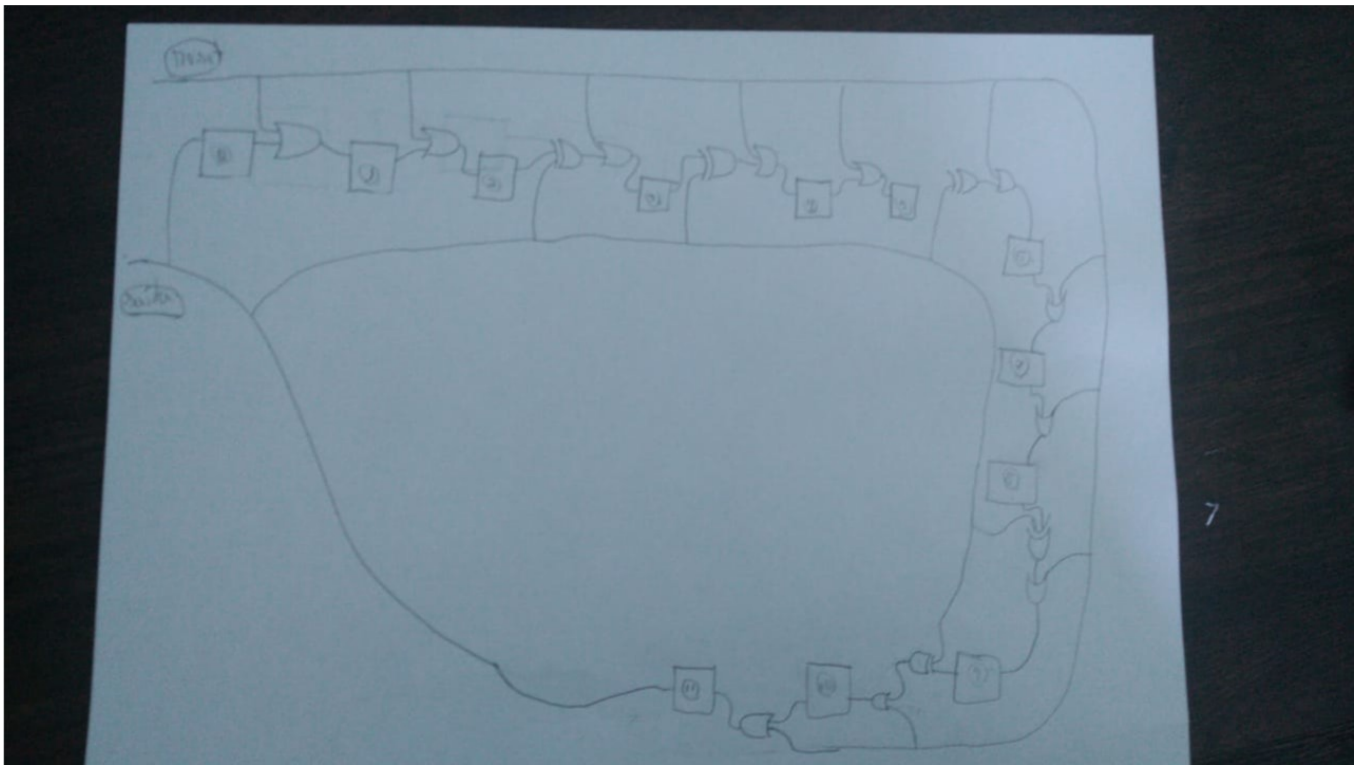
- Tabela 1 com os 10 primeiros números gerados pelo software. ATENÇÃO: apresentar os números em binário e hexadecimal.

ANEXO 2 (acrescentar no final do relatório): tabela 1 contendo os 10 estados codificados em BINÁRIO (copiados do software) e em HEXADECIMAL (para fácil identificação).

Tabela 1: Resultados da simulação (10 ciclos)

CICLO	Binário	Hexadecimal
1	100110100111	9A7
2	010100010111	517
3	101000101110	A2E
4	001000000101	205
5	010000001010	40A
6	100000010100	814
7	011001110001	671
8	110011100010	CE2
9	111110011101	F9D
10	100101100011	963

- Desenhar o circuito correspondente ao seu polinômio característico



ANEXO 3 (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.

1. **<u>Código VHDL ESTRUTURAL dos módulos RAND_NUM e LFSR (ver figura 1b no final deste texto).</u>**
   (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 <u>obrigatoriamente o comando GENERATE</u>

➢ Criar os códigos VHDL do RAND_NUM e do LFSR

ANEXO 4 (acrescentar no final do relatório): Descrições do RAND_NUM e do LFSR em VHDL.

LFSR:

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity flsr is
5   port (
6       clk : in std_logic;
7       res : in std_logic;
8       qi  : out  std_logic_vector(11 downto 0);
9       qout : out std_logic_vector(1 downto 0)
10  );
11  end entitv:
12
13  architecture arch of flsr is
14
15     component d_reg is
16     generic (t_ff: time := 2 ns);
17     port
18     (
41  begin
42
43     g1 : for i in 0 to 11 generate
44         g2: if i = 0 generate
45             --d_s(i)<= q_s(11) or res;
46             ou : or2 port map(q_s(11),res,d_s(i));
47             --d_s(i) <= q_s(11) or res;
48         end generate;
49         g3: if (i = 3) or (i=4) or (i = 6) or (i = 9) or (i = 10) generate
50             xor3 : xor2 port map(q_s(i-1),q_s(11),aux(i));
51             -- d_s(i) <= aux(i) or res;
52             or10 : or2 port map(aux(i),res,d_s(i));
53             --d_s(i) <= aux(i) or res;
54         end generate;
55         g4: if  (i=1) or (i = 2) or (i = 5) or (i = 7) or (i = 8) or (i = 11) generate
56             --d_s(i) <= q_s(i-1) or res;
57             or10: or2 port map(q_s(i-1), res, d_s(i));
58         end generate;
59         FF : d_reg port map(clk,'1',d_s(i),q_s(i));
60     end generate;
61     qout <= q_s(1 downto 0);
62     qi <= q_s;
63
64
65
66  end architecture;
```

RAND_NUM:

rand_num.vhd

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

entity rand_num is
port(
    clk, res: in std_logic;
    q_out: out std_logic_vector(7 downto 0)
);
end entity;

architecture arch of rand_num is

    component flsr is
        port (
            clk : in std_logic;
            res : in std_logic;
            qi  : out  std_logic_vector(11 downto 0);
            qout : out std_logic_vector(1 downto 0)
        );
    end component;

    signal n: std_logic_vector(1 downto 0);

    begin
        flsr_2: flsr port map(clk, res, open, n);

        q_out <= "000000" & n;

end arch ; -- arch
```

XOR_2bits:

xor2.vhd

```vhdl
Library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity xor2 is
    GENERIC(t_xor : time := 4 ns);
    PORT( x, y: IN STD_LOGIC;
          z: OUT STD_LOGIC);
END xor2;

ARCHITECTURE dataflow OF xor2 IS

BEGIN
    z <= x XOR y AFTER t_xor;
END dataflow;
```

OR_2bits:

or2.vhd

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   ENTITY or2 IS
5       GENERIC(t_or : time := 4 ns);
6       PORT( x, y: IN STD_LOGIC;
7             z: OUT STD_LOGIC);
8   END or2;
9
10  ARCHITECTURE dataflow OF or2 IS
11
12  BEGIN
13      z <= x OR y AFTER t_or;
14  END dataflow;
15
```

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.

2. **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)

➢ Código VHDL do arquivo de estímulos para simulação de RAND_NUM

ANEXO 5 (acrescentar no final do relatório): código do arquivo de estímulos

STIMULI:

stimuli.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity stimuli is
6      generic(
7          width : natural := 12;
8          CLK_PERIOD : time := 10 ns
9      );
10     port(
11         clk : out std_logic;
12         res : out std_logic;
13         rand : in std_logic_vector(1 downto 0)
14     );
15 end entity;
16
17
```

```
19 architecture test of stimuli is
20
21
22     component clock_generator --gerador de clock
23     generic(
24         CLK_PERIOD : TIME := 10 ns
25     );
26     port(
27         clk : out STD_LOGIC
28     );
29     end component;
30
31     signal clk_s : std_logic;
32
33 begin
34     clk <= clk_s;
35     clock: clock_generator port map(clk => clk_s);
36
37     sim : process
38         procedure reset_activate is
39             begin
40             wait until falling_edge(CLK_s);
41             res <= '1';
42             wait for 2*CLK_PERIOD;
43             res <= '0';
44         end procedure reset_activate;
```

```
45
46      begin
47          reset_activate;
48          wait for 20*CLK_PERIOD;
49      end process sim;
50
51  end architecture test;
52
```

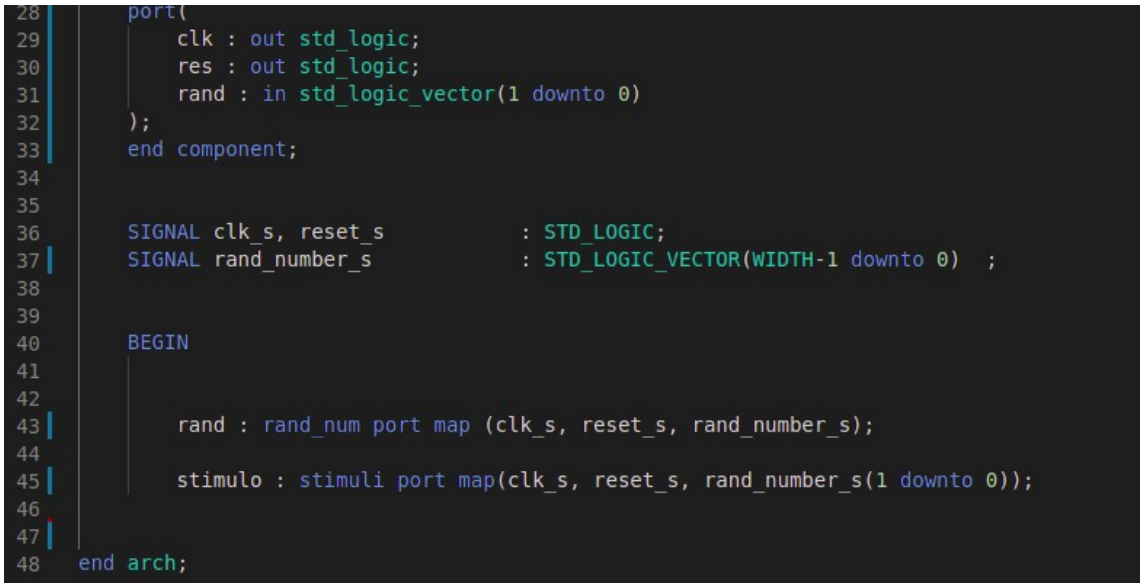
ATENÇÃO: ressaltar (sublinhar) as linhas de código correspondentes ao estabelecimento da condição inicial e o início da sequência de 10 (ou mais) ciclos. Estas linhas também podem ser identificadas através da inserção de comentários no código.

- Código VHDL do arquivo do *testbench* para simulação de RAND_NUM

ANEXO 6 (acrescentar no final do relatório): código do *testbench*

testbench_rand_num.vhd

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.all;
3   USE IEEE.numeric_std.all;
4
5
6   ENTITY tb_rand_num IS
7       GENERIC(
8               WIDTH       : NATURAL := 8
9       );
10
11  END tb_rand_num ;
12
13  ARCHITECTURE arch OF tb_rand_num IS
14
15
16      component rand_num is
17      port(
18          clk, res: in std_logic;
19          q_out: out std_logic_vector(7 downto 0)
20      );
21      end component;
22
23      component stimuli is
24      generic(
25          width : natural := 12;
26          CLK_PERIOD : time := 10 ns
27      );
```

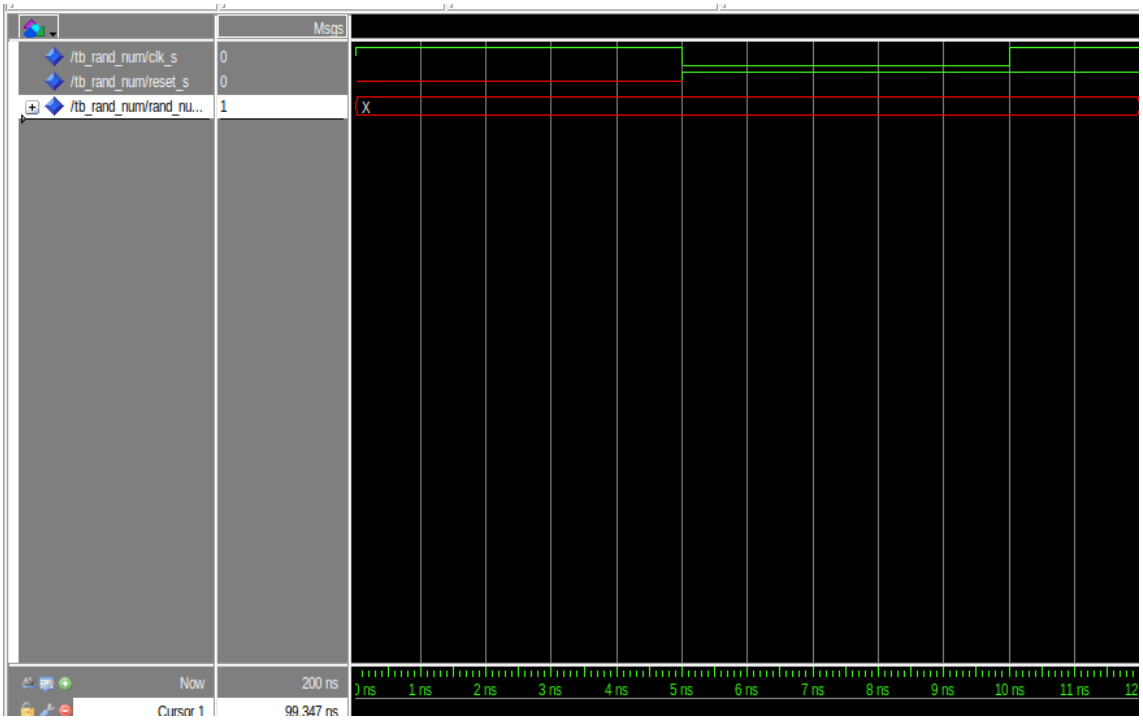


```
28      port(
29          clk : out std_logic;
30          res : out std_logic;
31          rand : in std_logic_vector(1 downto 0)
32      );
33      end component;
34
35
36      SIGNAL clk_s, reset_s           : STD_LOGIC;
37      SIGNAL rand_number_s            : STD_LOGIC_VECTOR(WIDTH-1 downto 0)  ;
38
39
40      BEGIN
41
42
43          rand : rand_num port map (clk_s, reset_s, rand_number_s);
44
45          stimulo : stimuli port map(clk_s, reset_s, rand_number_s(1 downto 0));
46
47
48  end arch;
```

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

3. <u>Resultados das simulações através do programa ModelSim</u>

- Simulação mostrando a correta a condição inicial do LFSR e do RAND_NUM

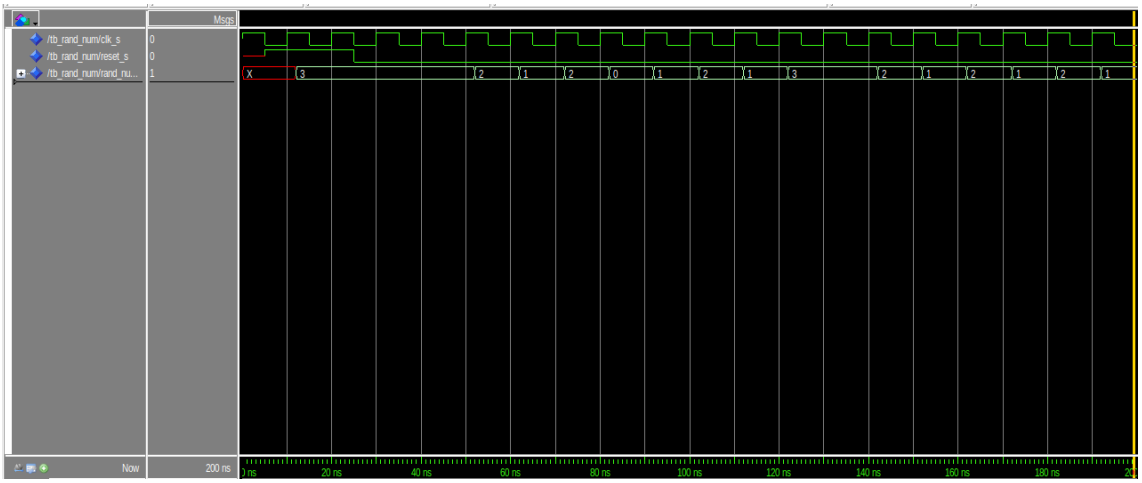


ANEXO 7 (acrescentar no final do relatório): Imagem do WAVE ilustrando a condição inicial do LFSR e do RAND_NUM

- Simulação mostrando a correta geração da sequência pseudo-aleatória

ANEXO 8 (acrescentar no final do relatório): Imagem do WAVE ilustrando a sequências das 10 saídas do RAND_NUM e do LFSR (os mesmos percorridos durante a simulação com o software).

ATENÇÃO: as saídas devem estar identificadas pelos mesmos valores HEXADECIMAIS apresentados no anexo 2 (para fácil identificação).



- Tabela 2 anotando os instantes de tempo em que ocorrem as 10 saídas relevantes do LFSR e do RAND_NUM.

ANEXO 9 (acrescentar no final do relatório): tabela 2 com os 10 valores de saída do LFSR (em hexadecimal), os 10 valores de saída do RAND_NUM (em binário) e os instantes de tempo em que ocorrem (extraídos da simulação).

Tabela 2: Resultados da simulação ModelSim

Tempo (em ns)	Saída LFSR	Saída RAND_NUM
12	9A7	3
52	517	2
62	A2E	1
72	205	2
82	40A	0
92	814	1
102	671	2
112	CE2	1
122	F9D	3
132	963	2

- Observe e comente:
  1) se os resultados das simulações por software e pelo ModelSim foram iguais

     R: Sim os resultados da parte do FLSR foram todos corretos !
  2) se a aleatoriedade dos valores de saída do módulo LFSR foi observada na saída do módulo RAND_NUM

     R: Sim, observa-se uma sequêncai aleatória de 0 a 3.

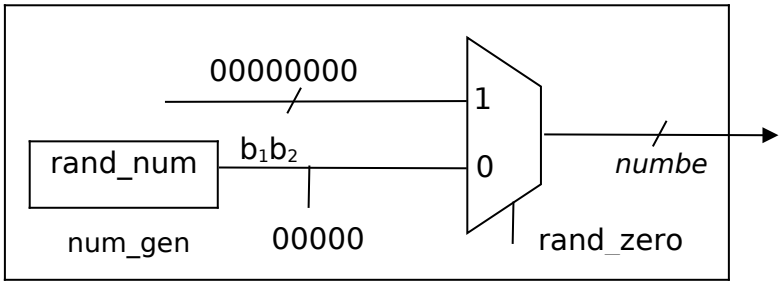
ANEXO 10 (acrescentar no final do relatório): comentários.

IMPORTANTE: Para validar o seu projeto, além deste relatório, fazer o UP-LOAD de todos os arquivos VHDL usados neste projeto:

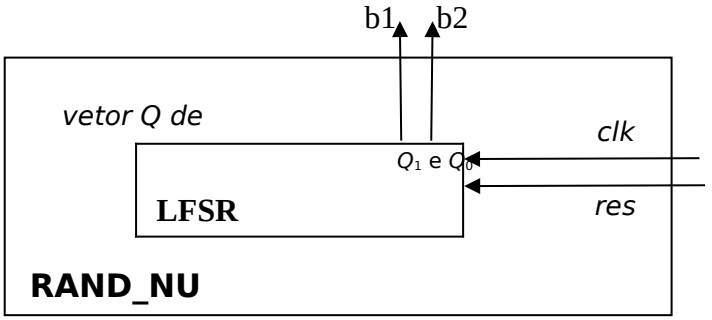
- ✓ **RAND_NUM**
- ✓ **LFSR E SEUS COMPONENTES**
- ✓ **ESTÍMULOS E TESTBENCH**

Breves considerações teóricas:

Este relatório refere-se ao projeto do LFSR, que fará parte do módulo *random_num* (figura 1b) e este, por sua vez fará parte do módulo *num_gen* (figura 1a). O LFSR atuará como gerador de números aleatórios e será integrado ao Circuito do Wisdom. Observar que serão aproveitados apenas 2 bits (Q_0 e Q_1) dos n(=12) bits do LFSR. Estes serão utilizados para a definição do endereço da posição inicial do Guru.



(a) Esquema de geração de número aleatório (aula 4)



(b) Esquema do módulo RAND_NUM contendo o LFSR

Figura 1: Esquemas detalhados de 2 módulos

O sinal RESET será usado para estabelecer a condição inicial do LFSR na configuração Galois (todos os registradores com saídas "1" (figura 2).

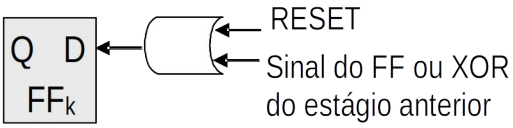


Figura 2: Esquema detalhado de cada estágio do LFSR (sem a EXOR)

No desenvolvimento do código VHDL do LFSR

Adotar:

- O modelo VHDL do DFF 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 na descrição VHDL do LFSR
- O LFSR terá obrigatoriamente 12 FFs (Q_{11} .. Q_0). As saídas (Q_1 e Q_0,) serão depois roteadas para as saídas do módulo *rand_num* (figura 1.b).