Henrique Soares Assumpção e Silva
Matrícula: 2020006620
henriquesoares@dcc.ufmg.br

Trabalho em Verilog III
Introdução a Sistemas Lógicos

2021-08-26

1 Introdução

A atividade requisitava a implementação de três registradores de deslocamento com feedback linear(LFSR) em Verilog, bem como a implementação de gráficos e figuras ilustrativas para visualização dos padrões gerados pelos dados.
O programa foi desenvolvido na linguagem Verilog, compilada utilizando a ferramenta Icarus Verilog, no sistema operacional Windows 10. O programa foi inicialmente implementado e compilado em uma máquina com 16GB de RAM em um processador Intel Core I5-9400F.

2 Instruções de compilação

Para executar as implementações feitas no trabalho, basta realizar os seguintes passos:

```
>>> iverilog -o test_bench_<NUM>bit.vvp test_bench_<NUM>bit.v
>>> vvp test_bench_<NUM>bit.vvp
```

Onde <NUM> representa o número de bits do LFSR, ou seja, $<\texttt{NUM}> \in \{3, 4, 8\}$. O dados de ondas são salvos em arquivos `test_<NUM>.vcd` e podem ser visualizados por meio de programas como o GTKWave.
Todo o código deste trabalho esta disponível no seguinte repositório do GitHub:
hfill https://github.com/HenrySilvaCS/Logical-Systems-UFMG/tree/main/tp3

3 Implementação dos LFSR

Para realização desta atividade, optou-se por implementar três LFSR diferentes, de tamanhos 3,4 e 8, utilizando-se os polinômios que maximizam o período do circuito. As seções a seguir contém as implementações em Verilog, os gráficos das representações decimais em função do tempo e os bitmaps criados para os respectivos registradores.

3.1 3-BIT LFSR

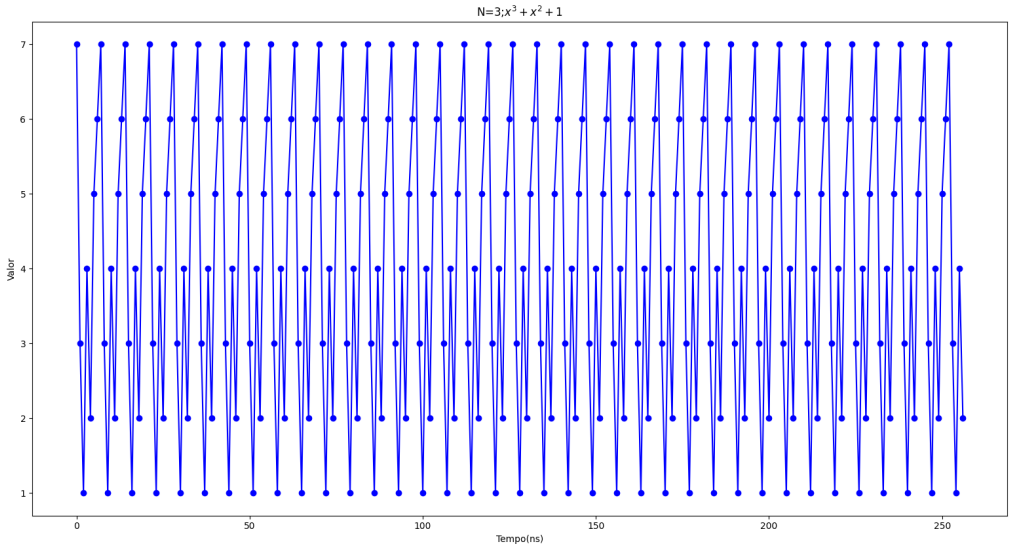
Primeiramente, destaca-se o código do registrador LFSR para 3 bits:

```
module lfsr_3bit(
    input   clock,
    input   reset,
    input   [2:0] taps,
    input   [2:0] reset_value,
    output  reg [2:0] computed_value);


    always @ (posedge clock, negedge reset)
    begin
        if (!reset)
        begin
            computed_value <= reset_value;
        end
        else
        begin
            computed_value <= computed_value >> 1;
            computed_value[2] <= computed_value[0] ^
                                 ((taps[1]) ? computed_value[1] : 0) ^
                                 ((taps[0]) ? computed_value[2] : 0);
        end
    end
endmodule
module lfsr_testbench_3bit;
    reg clk;
    reg n_rst;
    reg [2:0] polinomio;
    reg [2:0] reset;
    wire [2:0] test_computed_value;
    integer posedgectr;

    lfsr_3bit lfsr_iut(.clock(clk),
                   .reset(n_rst),
                   .taps(polinomio),
                   .reset_value(reset),
```

```
                    .computed_value(test_computed_value));
    initial
    begin
        #0
        $monitor("LSFR_3BIT (clock_atual=%d): %d", posedgectr, test_computed_value);
        $dumpfile("test_3.vcd");
        $dumpvars(0, lfsr_testbench_3bit);

        n_rst  = 1;
        clk    = 0;
        posedgectr= 0;

        #10
        reset = 3'b111;
        polinomio = 3'b110;
        n_rst = 0;

        #10
        n_rst = 1;

        #512
        $finish;
    end

    always #1 begin
        clk = !clk;
        if (clk) posedgectr = posedgectr + 1;
    end
endmodule
```

Nota-se que ele recebe como entrada a variável taps, que é uma sequência de bits representando o polinômio em questão, que possibilita que diferentes polinomios sejam testados de modo a reutilizar o código. Além disso, também é possível variar o valor inicial armazenado nos flip flops, através da variável reset_value, que por padrão é uma sequência composta apenas por 1's. Esses padrões da implementação foram mantidos para os outros dois registradores criados.
Além disso, também foi feito um gráfico representando os valores gerados em função do tempo: Também foi

Figure 1: Gráfico que mostra os valores gerados em representação decimal em função do tempo em nanosegundos. Percebe-se que o período é máximo e igual a $T = 2^3 - 1 = 7$ ciclos.



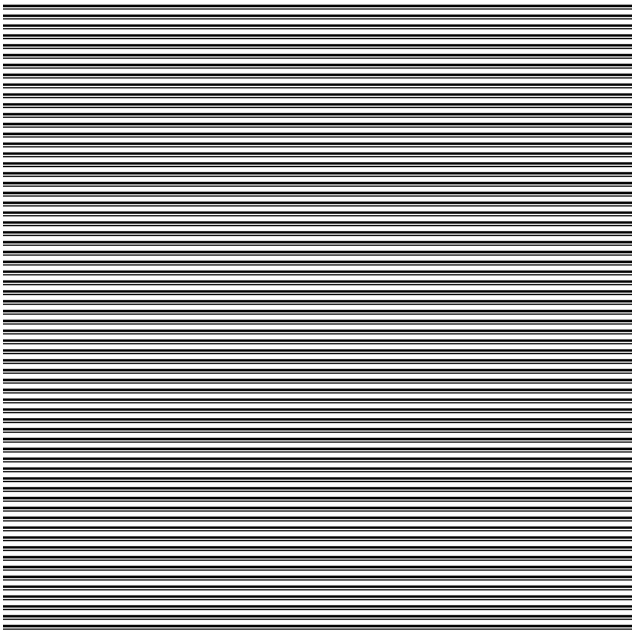
desenvolvido o bitmap para o padrão do LFSR em questão:

3.2 4-BIT LFSR

Primeiramente, destaca-se o código do registrador LFSR para 4 bits:

```
module lfsr_4bit(
    input   clock,
    input   reset,
    input   [3:0] taps,
    input   [3:0] reset_value,
    output  reg [3:0] computed_value);


```

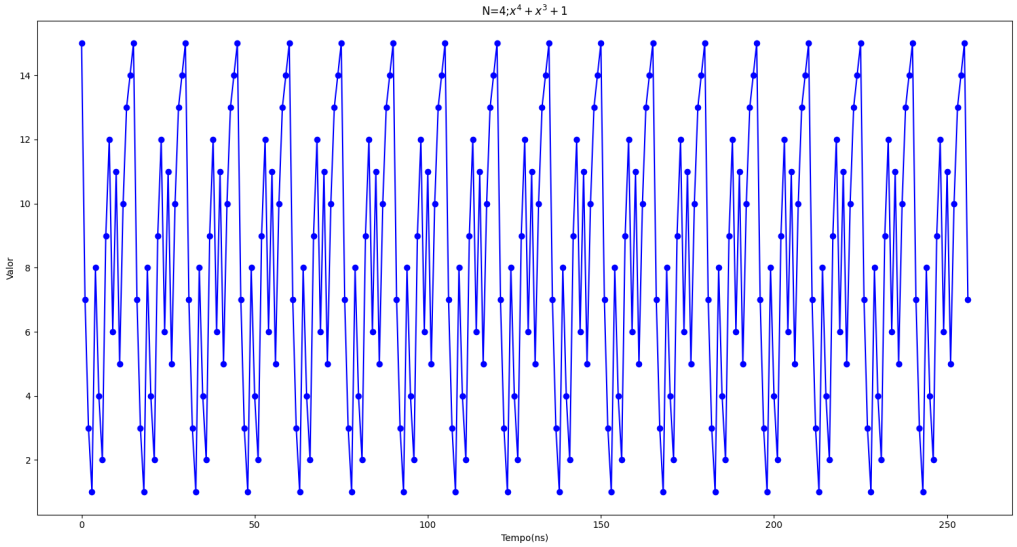
Figure 2: Bitmap do LFSR de 3 bits para o polinômio $x^3 + x^2 + 1$.



```
    always @ (posedge clock, negedge reset)
    begin
        if (!reset)
        begin
            computed_value <= reset_value;
        end
        else
        begin
            computed_value <= computed_value >> 1;
            computed_value[3] <= computed_value[0] ^
                                 ((taps[2]) ? computed_value[1] : 0) ^
                                 ((taps[1]) ? computed_value[2] : 0) ^
                                 ((taps[0]) ? computed_value[3] : 0);
        end
    end
endmodule
module lfsr_testbench_4bit;
    reg clk;
    reg n_rst;
    reg [3:0] polinomio;
    reg [3:0] reset;
    wire [3:0] test_computed_value;
    integer posedgectr;

    lfsr_4bit lfsr_iut(.clock(clk),
                  .reset(n_rst),
                  .taps(polinomio),
                  .reset_value(reset),
                  .computed_value(test_computed_value));
    initial
    begin
        #0
        $monitor("LSFR_4BIT (clock_atual=%d): %d", posedgectr, test_computed_value);
        $dumpfile("test_4.vcd");
        $dumpvars(0, lfsr_testbench_4bit);

        n_rst  = 1;
        clk    = 0;
        posedgectr= 0;

        #10
        reset = 4'b1111;
        polinomio = 4'b1100;
        n_rst = 0;

        #10
        n_rst = 1;

        #512
        $finish;
    end

    always #1 begin
        clk = !clk;
        if (clk) posedgectr = posedgectr + 1;
    end
endmodule
```

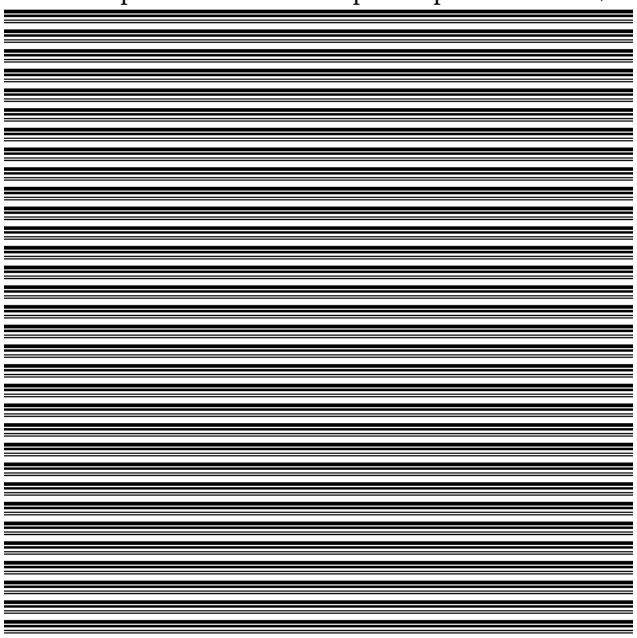
Além disso, também foi feito um gráfico representando os valores gerados em função do tempo: Também foi

Figure 3: Gráfico que mostra os valores gerados em representação decimal em função do tempo em nanosegundos. Percebe-se que o período é máximo e igual a $T = 2^4 - 1 = 15$ ciclos.



desenvolvido o bitmap para o padrão do LFSR em questão:

Figure 4: Bitmap do LFSR de 4 bits para o polinômio $x^4 + x^3 + 1$.



3.3 8-BIT LFSR

Primeiramente, destaca-se o código do registrador LFSR para 8 bits:

```
module lfsr_8bit(
    input   clock,
    input   reset,
    input   [7:0] taps,
    input   [7:0] reset_value,
    output  reg [7:0] computed_value);


    always @ (posedge clock, negedge reset)
    begin
```

```
        if (!reset)
        begin
            computed_value <= reset_value;
        end
        else
        begin
            computed_value <= computed_value >> 1;
            computed_value[7] <= computed_value[0] ^
                                 ((taps[6]) ? computed_value[1] : 0) ^
                                 ((taps[5]) ? computed_value[2] : 0) ^
                                 ((taps[4]) ? computed_value[3] : 0) ^
                                 ((taps[3]) ? computed_value[4] : 0) ^
                                 ((taps[2]) ? computed_value[5] : 0) ^
                                 ((taps[1]) ? computed_value[6] : 0) ^
                                 ((taps[0]) ? computed_value[7] : 0);
        end
    end
endmodule
module lfsr_testbench_8bit;
    reg clk;
    reg n_rst;
    reg [7:0] polinomio;
    reg [7:0] reset;
    wire [7:0] test_computed_value;
    integer posedgectr;

    lfsr_8bit lfsr_iut(.clock(clk),
                  .reset(n_rst),
                  .taps(polinomio),
                  .reset_value(reset),
                  .computed_value(test_computed_value));
    initial
    begin
        #0
        $monitor("LSFR_8BIT (clock_atual=%d): %d", posedgectr, test_computed_value);
        $dumpfile("test_8.vcd");
        $dumpvars(0, lfsr_testbench_8bit);

        n_rst  = 1;
        clk    = 0;
        posedgectr= 0;

        #10
        reset = 8'b11111111;
        polinomio = 8'b10111000;
        n_rst = 0;

        #10
        n_rst = 1;

        #512
        $finish;
    end

    always #1 begin
        clk = !clk;
        if (clk) posedgectr = posedgectr + 1;
    end
endmodule
```

Além disso, também foi feito um gráfico representando os valores gerados em função do tempo: Também foi desenvolvido o bitmap para o padrão do LFSR em questão:

Figure 5: Gráfico que mostra os valores gerados em representação decimal em função do tempo em nanosegundos. Percebe-se que o período é máximo e igual a $T = 2^8 - 1 = 255$ ciclos.

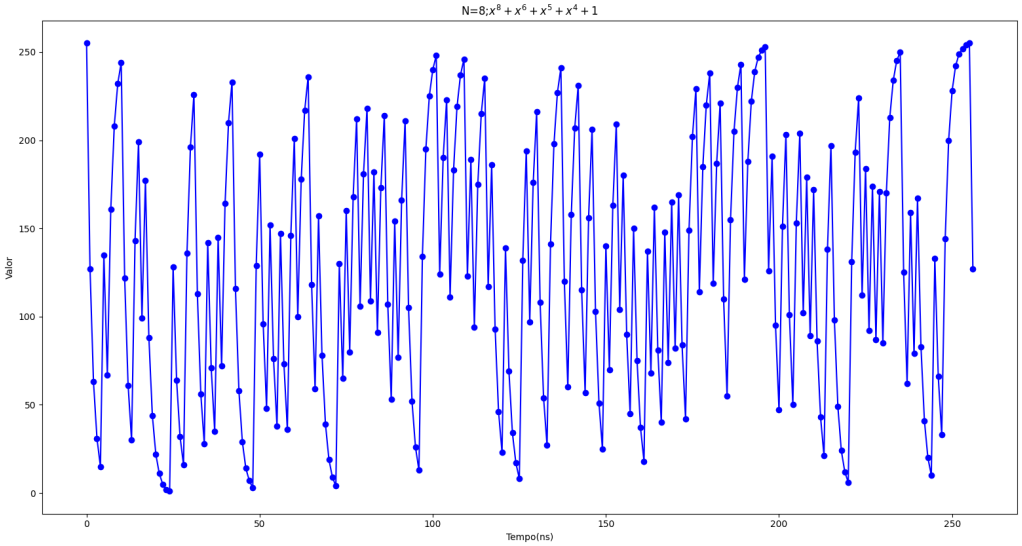


Figure 6: Bitmap do LFSR de 8 bits para o polinômio $x^8 + x^6 + x^5 + x^4 + 1$.