

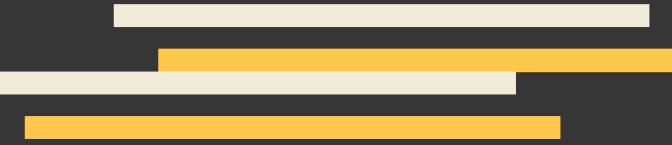
Gerador de Números **PSEUDO-ALEATÓRIOS COM LFSR**

Equipe: Amanda Vieira, Letícia Freitas e Renan Alves

Professor: Otávio Alcantara de Lima Junior

TÓPICOS ASSOCIADOS

- Introdução
- Metodologia
- Resultados
- Conclusão

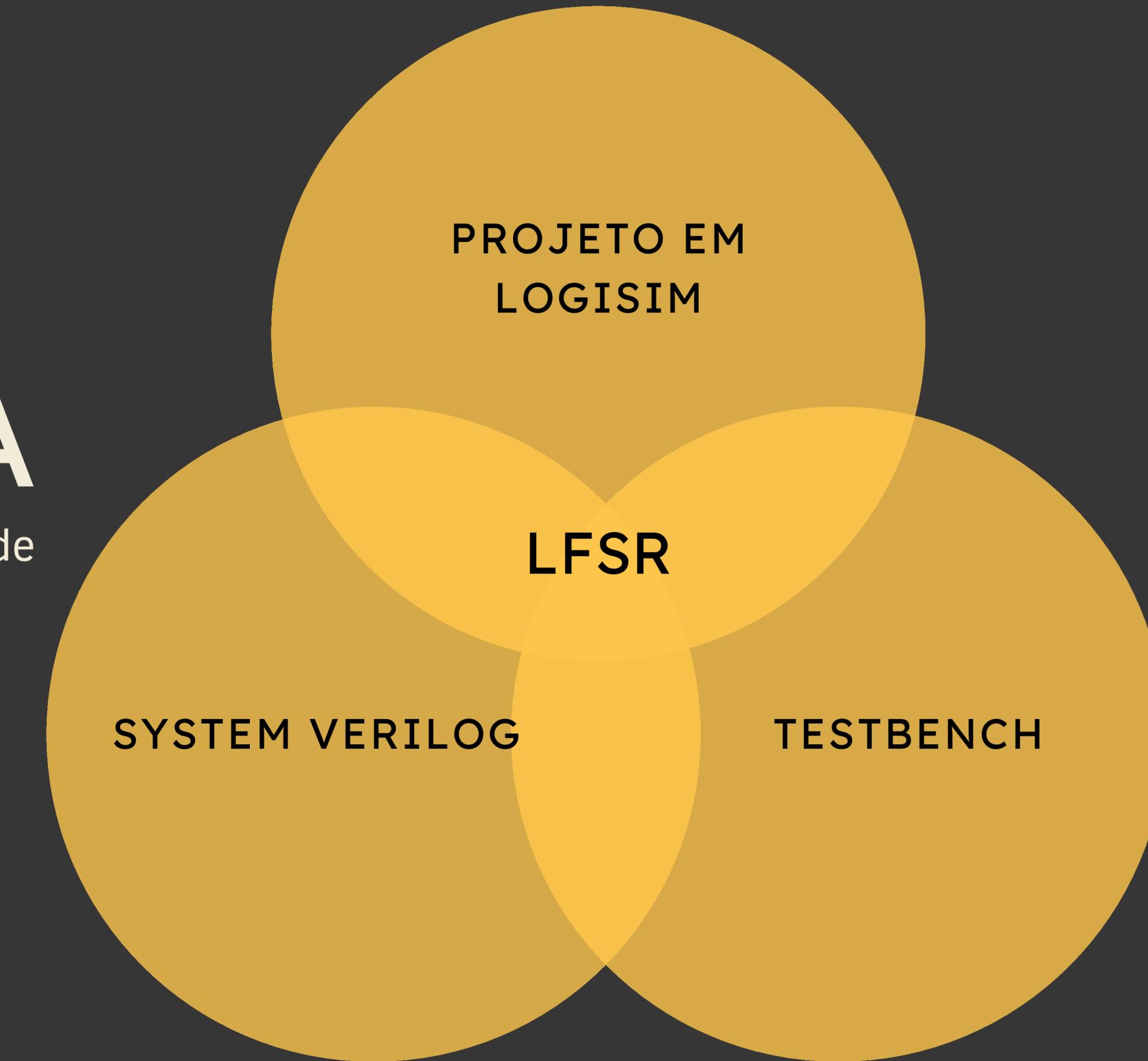


INTRODUÇÃO

Um LFSR (Linear Feedback Shift Register) é um registrador que desloca bits a cada ciclo de clock, combinando algumas de suas saídas através de operações XOR para formar a realimentação. A realimentação resultante é usada para atualizar o valor do registrador, gerando sequências pseudo aleatórias.

METOLOGIA

Em LFSR (Linear Feedback Shift Register) de 32 bits vai gerar uma sequência pseudo-aleatória deslocando seu conteúdo a cada pulso de clock e aplicando uma operação XOR nas posições específicas (taps) para realimentação

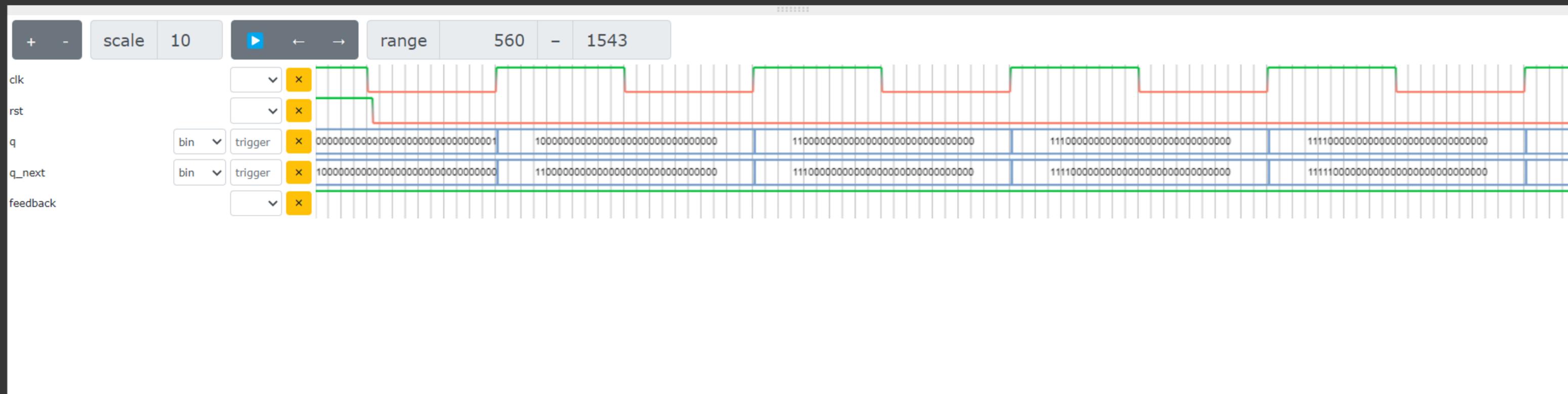


RESULTADOS EM SYSTEMVERILOG:

```
module lfsr #(parameter seed = 1)( // seed -> define um valor inicial
    input logic clk, rst,           // entradas de clock e reset
    output logic [31:0] q          // saida do vetor com 32 bits, em 'q'
);
    logic feedback;               // irá armazenar o valor de realimentação
    logic [31:0] q_reg, q_next;   // valor de 32 bits para o valor atual e o próximo valor

    always_ff @(posedge clk, posedge rst) // declara uma borda de subida para o clock e reset
    begin
        if (rst) begin
            q_reg <= seed;           // se houver um reset, inicializa o LFSR com o valor da seed
        end
        else begin
            q_reg <= q_next;         // caso contrario vai atualizar o registrador com o próximo valor
        end
    end
    // vai gerar o sinal de feedback, usando uma combinação XOR de bits específicos do q_reg
    assign feedback = q_reg[0] ^ q_reg[1] ^ q_reg[21] ^ q_reg[31]; // 0, 1, 21 e 31 são os pontos de toque da sequência
    assign q_next = {feedback, q_reg[31:1]}; // define o próximo valor do registrador como a concatenação do feedback
    assign q = q_reg; // atribui o valor do registrador à saída
endmodule
```

RESULTADOS EM FORMAS DE ONDA:

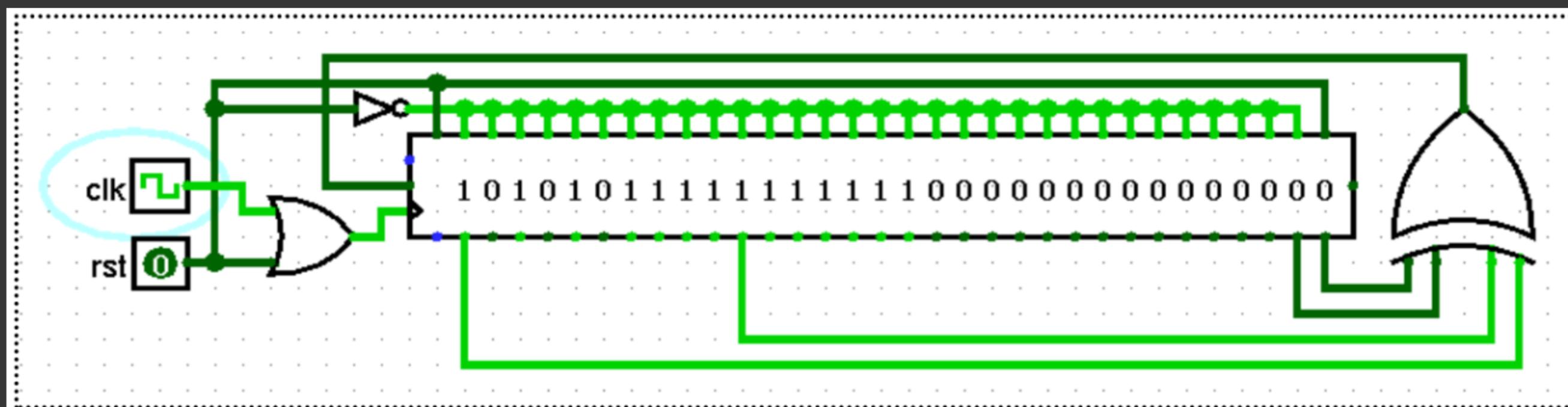


RESULTADOS EM TESTBENCH:

```
testbench.sv ×

1 module tb_lfsr;
2   localparam seed = 1;
3   logic clk, rst;
4   logic [31:0] q;
5
6   localparam N = 100;
7   logic [31:0] testvectors [N-1:0];
8   logic [31:0] expected_q;
9
10  lfsr #(seed) dut(clk, rst, q);
11  always #5 clk = ~clk;
12
13  initial begin
14    $dumpfile("dump.vcd");
15    $dumpvars(0, clk, q);
16
17    clk = 1; rst = 0;
18    rst = 1; @(posedge clk); rst = 0;
19
20    $readmemb("testvectors.txt", testvectors, 0, N-1);
21    for (int i = 0; i < N; i++) begin
22      expected_q = testvectors[i];
23      @(posedge clk);
24
25      assert(q === expected_q);
26      else $error(
27        "Erro na linha %0d: q=%b, expected_q=%b",
28        i+1, q, expected_q
29      );
30    end
31
32    $finish;
33  end
34 endmodule
35
```

RESULTADOS EM LOGISIM:





CONCLUSÃO

A implementação do LFSR em LogiSim e SystemVerilog foi bem-sucedida, com o circuito em LogiS, código em SystemVerilog e TestBench comprovando a eficácia da realimentação e do sinal de reset. Ambas as abordagens demonstraram que o LFSR é um método eficiente para gerar sequências pseudo-aleatórias, destacando sua importância em circuitos digitais.

Referências:

- FPGA e Verilog - Aula 39 - Introdução à Verificação
(Testbench) ([youtube.com](https://www.youtube.com))
- Proakis, J. G., & Salehi, M. (2007). Digital Communications
(5th ed.). McGraw-Hill.
- Golomb, S. W. (1967). Shift Register Sequences. Aegean
Park Press.