Design and mixed signal simulation of 4-Bit Pseudo Random Sequence Generator using eSim and NgVeri technology

Roshan Binu Paul, Muthoot Institute of Technology and Science, Kerala

April 26, 2023

Abstract

This proposal presents the design and mixed signal simulation of a 4-bit Pseudo Random Sequence Generator (PRSG) using eSim and Ngveri technology. The proposed PRSG uses a Linear Feedback Shift Register (LFSR) algorithm to generate a sequence of pseudo-random numbers. The design can be simulated using eSim, an open-source circuit simulator, and Ngveri. PRSG circuit has good statistical properties, making it suitable for use in applications such as cryptography, spread spectrum communication systems, and digital signal processing.

Connecting PRSG to the real-world could make changes in the seed values, thus generating different sequences .The quality of the sequence generated is dependent on the seed. Thus through eSim we could find the best possible solution for this problem .

1.Circuit Details

The LFSR is a shift register with feedback. It generates a sequence of pseudo-random numbers by shifting the contents of the register to the right and feeding back the output of selected stages to the input of the register. To implement a 4-bit PRSG we are using D-flip flops and Xor gate .

The pattern repeats endlessly. The length of the pattern in 15 different bit combinations. This is the maximum possible. There are at most 16 different bit combinations in a 4 bit register (2 to the power 4) however the combination 0000 is not a real possibility, since if the register is ever at 0000 it will remain forever at 0000. A maximum length sequence for a shift register of length N is referred to as msequence and is defined as: L =2^n -1.

1. Initial state: The PRSG starts with an initial state, which is a binary value stored in the four D flip-flops. This initial state determines the sequence of pseudo-random binary numbers generated by the PRSG. We are providing 0001 as the initial state

2. Feedback loop: The outputs of the D flip-flops are fed back to the input of the flip-flop and to a logic circuit that generates the next input value for the flip-flop. The feedback loop creates a sequence of binary numbers that appears random but is actually deterministic based on the feedback polynomial used.

3. Clock input: The clock input is applied to all the D flip flops in parallel. On the rising edge of the clock signal, the input value is transferred to the output of the flip-flop.

4. Output sequence: The output sequence of the PRSG is taken from the outputs of the D flip-flops. The sequence of binary numbers generated is a pseudo-random sequence based on the initial state and feedback polynomial used.

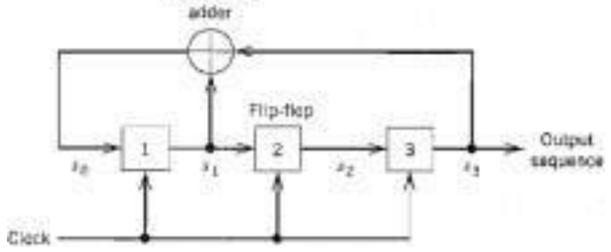


Figure 1: 4- bit PRSG

2.Schematic Diagram and analysis

The circuit schematic of the 4-Bit Pseudo Random Sequence Generator in eSim is as shown below:

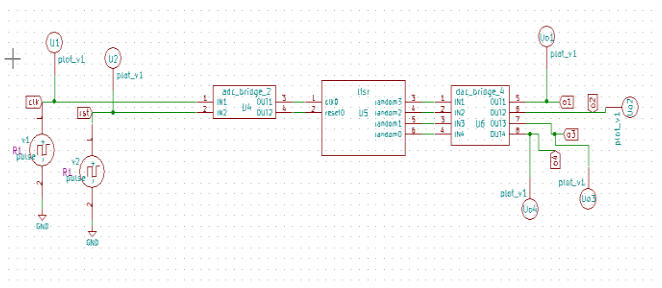
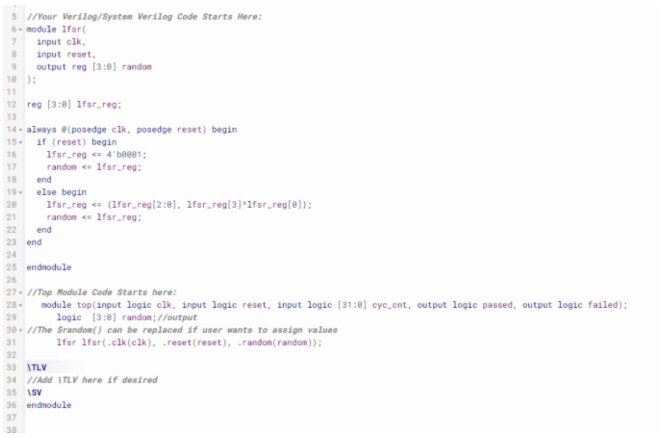
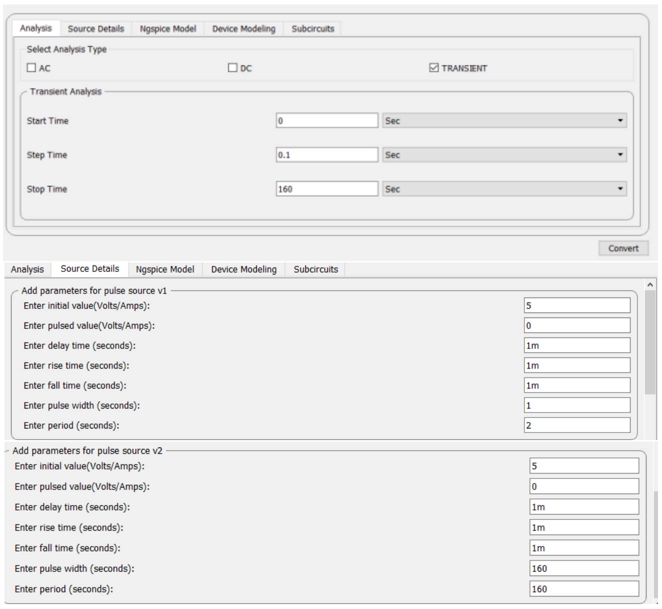


Figure 2a : schematic of 4bit PRSG



```
//Your Verilog/System Verilog Code Starts Here:
module lfsr(
  input clk,
  input reset,
  output reg [3:0] random
);

reg [3:0] lfsr_reg;

always @(posedge clk, posedge reset) begin
  if (reset) begin
    lfsr_reg <= 4'b0001;
    random <= lfsr_reg;
  end
  else begin
    lfsr_reg <= {lfsr_reg[2:0], lfsr_reg[3]^lfsr_reg[0]};
    random <= lfsr_reg;
  end
end

endmodule

//Top Module Code Starts here:
   module top(input logic clk, input logic reset, input logic [31:0] cyc_cnt, output logic passed, output logic failed);
      logic  [3:0] random;//output
//The $random() can be replaced if user wants to assign values
      lfsr lfsr(.clk(clk), .reset(reset), .random(random));

\TLV
//Add \TLV here if desired
\SV
endmodule
```

Figure 2b : Verilog code on makerchip



Analysis | Source Details | Ngspice Model | Device Modeling | Subcircuits

Select Analysis Type

☐ AC ☐ DC ☑ TRANSIENT

Transient Analysis

Start Time	0	Sec
Step Time	0.1	Sec
Stop Time	160	Sec

Convert

Analysis | Source Details | Ngspice Model | Device Modeling | Subcircuits

Add parameters for pulse source v1

Enter initial value(Volts/Amps):	5
Enter pulsed value(Volts/Amps):	0
Enter delay time (seconds):	1m
Enter rise time (seconds):	1m
Enter fall time (seconds):	1m
Enter pulse width (seconds):	1
Enter period (seconds):	2

Add parameters for pulse source v2

Enter initial value(Volts/Amps):	5
Enter pulsed value(Volts/Amps):	0
Enter delay time (seconds):	1m
Enter rise time (seconds):	1m
Enter fall time (seconds):	1m
Enter pulse width (seconds):	160
Enter period (seconds):	160

Figure 2c : analysis setup

3.Simulation Results

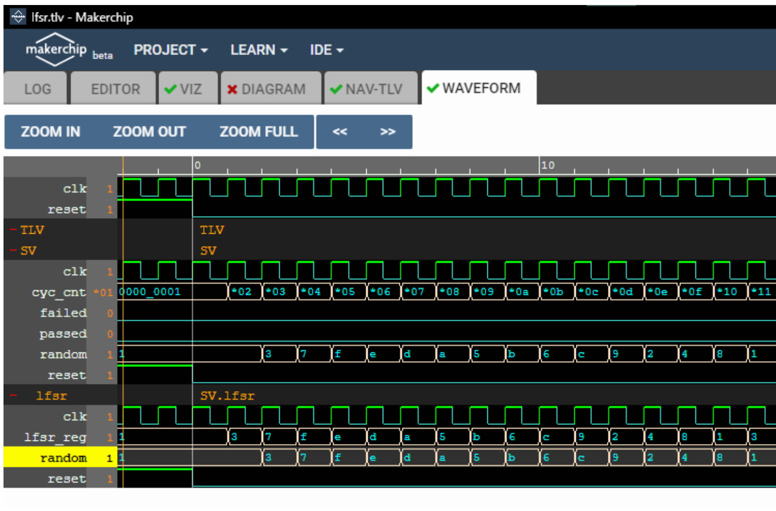


Figure 3 : simulation of MakerChip

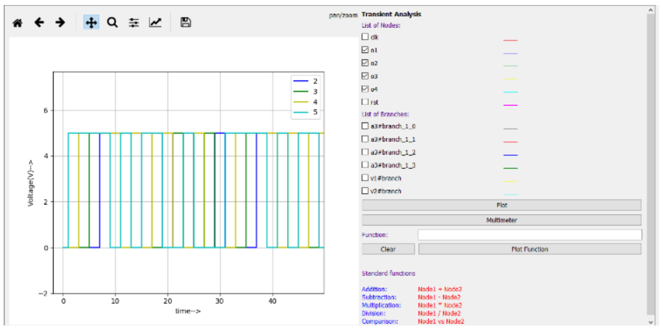
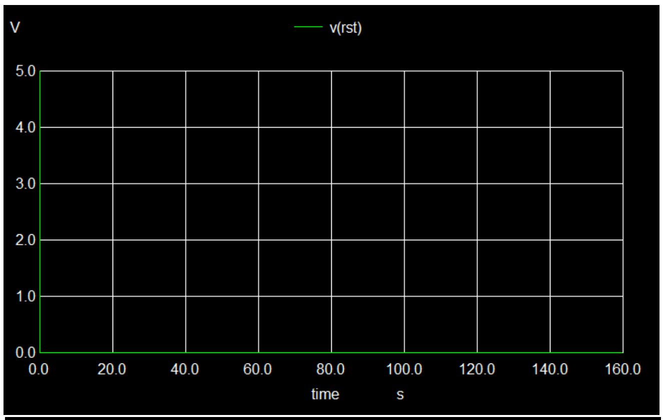
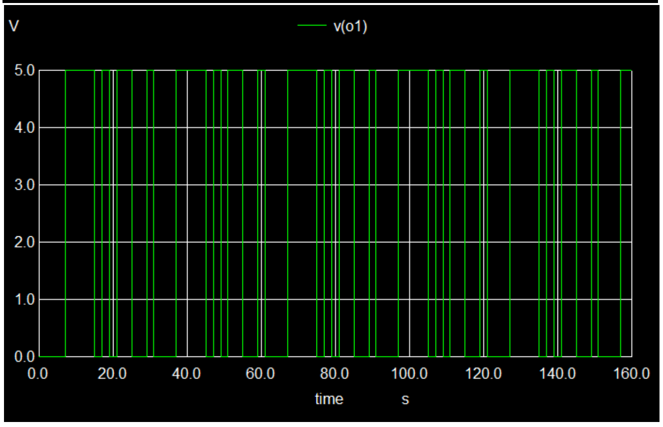


Figure 4 : simulation of output on python plot





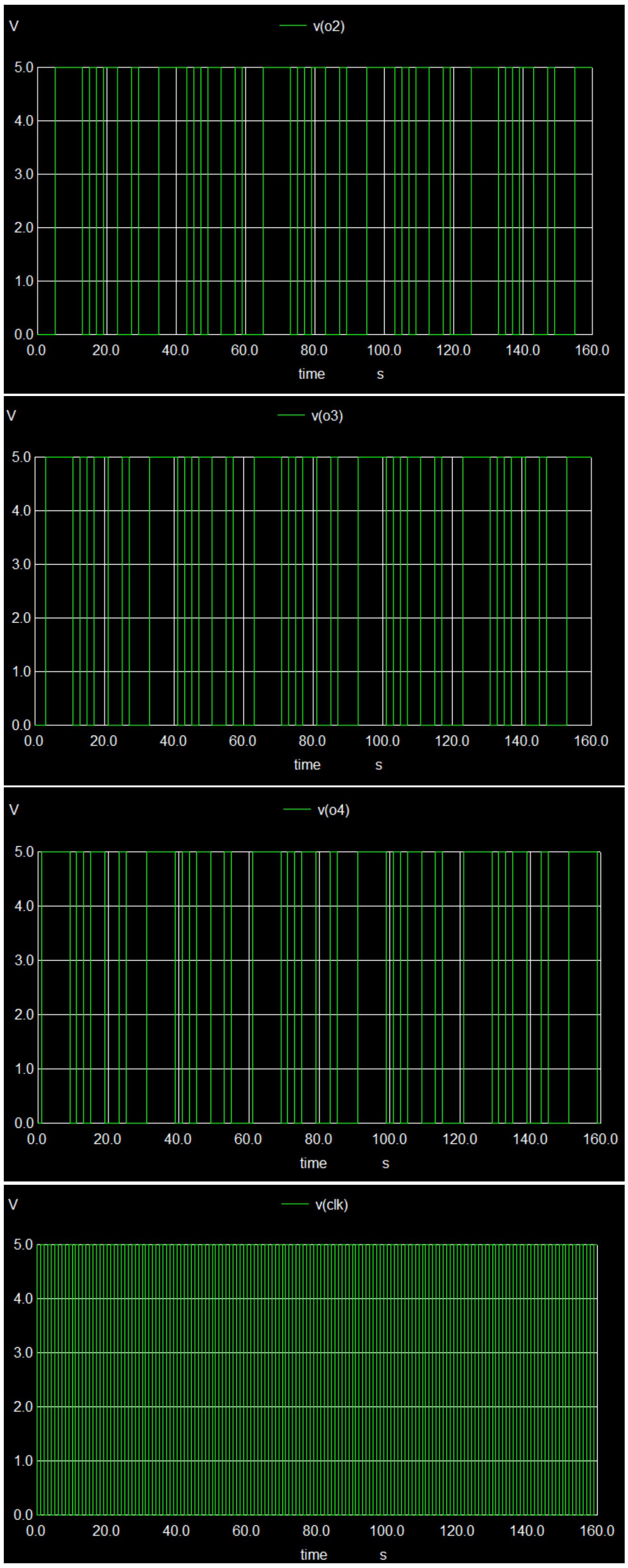


Figure 5 : simulation of output on Ngspice

4.Importance of PRSG

PRSGs have a wide range of applications, including encryption, testing, testing, and many more.

In encryption, PRSGs are used to generate a sequence of random numbers that are used to encrypt data. The encryption is based on a secret key that is generated by the PRSG, making it difficult for unauthorized users to access the encrypted data.

In testing electronic circuits, PRSGs are used to simulate a sequence of random inputs. This allows engineers to test the circuit's functionality under different input conditions and identify potential issues.

5.Conclusion

We have received maximum period of 15 using the above concept. The polynomial (x^4 + x^3 + 1) is primitive and it is implemented in this . Incase of reset 0000 as we expected are becoming 0 . Here we have set reset to high for the initial phase to make the seed value 0001 .

The obtained sequence is as follow :

0001-0011-0111-1111-1110-1101-1010-0101-1011-0110-1100-1001-0100-1000 // -0001

1-3-7-f-e-d-a-5-b-6-c-9-2-4-8 // - 1

Thus, eSim can be utilized to study and analyse PRSG . The Ngveri technology makes it easy to produce required component in quick and effective way , also the Makerchip platform provides great analysis can produce appropriate waveforms .

References

[1] Burton Rosenberg Floyd. Generation of pseudorandom numbers by a shift register.

https://www.cs.miami.edu/home/burt/learning/Csc609.022/random_numbers.html