PRNG using FPGA

Anikesh Parashar (21114012)

Introduction

Random numbers play an important role in a wide range of engineering and scientific domains, such as cryptography, machine learning, communications, simulations, etc. These applications often require sequences of numbers that are either truly random or sufficiently unpredictable to simulate randomness. However, generating truly random numbers in digital hardware is inherently difficult. True random number generators (TRNGs) typically rely on analog processes or unpredictable physical phenomena, such as thermal noise or radioactive decay — resources that are either unavailable or impractical in many digital and embedded systems.

In contrast, Pseudorandom Number Generators (PRNGs) offer a practical and efficient alternative. These are algorithmic methods that produce sequences of numbers that appear statistically random, even though they are generated deterministically from an initial seed. PRNGs are especially important in hardware-constrained environments such as Field-Programmable Gate Arrays (FPGAs), where simplicity, speed, and resource efficiency are critical.

This project investigates the design and FPGA implementation of a PRNG using a Linear Feedback Shift Register (LFSR). The LFSR is a widely used digital structure for generating pseudorandom bit sequences with minimal hardware overhead. The goal of this project is to understand the theoretical basis of LFSRs, implement a parameterizable version in Verilog, verify its behavior via simulation, and evaluate its suitability for synthesis on an FPGA.

Pseudorandom Numbers

Unlike true random number generators that rely on physical processes, PRNGs use deterministic algorithms to generate sequences of bits that appear random. A PRNG typically requires:

- An initial seed (starting state),
- A recurrence relation (or transformation logic),
- And a mechanism to update the internal state.

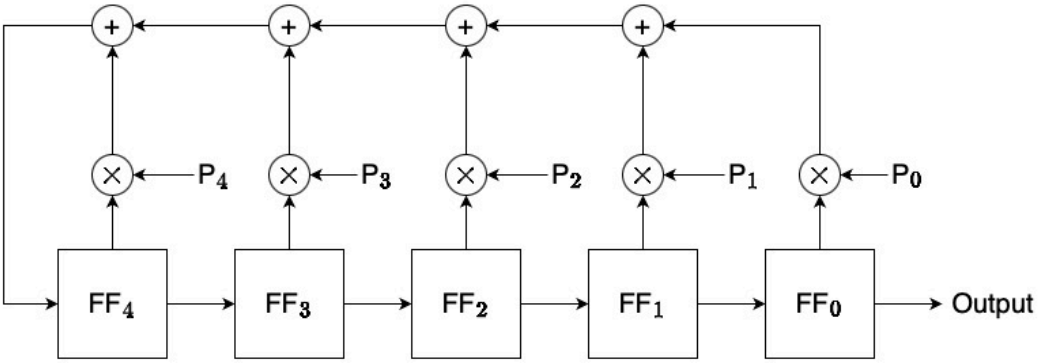
Despite being deterministic, PRNGs are widely acceptable in engineering applications where the randomness needs to be reproducible or doesn't require high entropy. For hardware implementations, PRNGs must be:

- Fast and lightweight,
- Synchronous with the system clock,
- Able to produce long non-repeating sequences.

The LFSR satisfies all these requirements and is commonly used in test pattern generation, scramblers, and simple simulations.

Linear Feedback Shift Register

LFSR (Linear Feedback Shift Register) is a Shift Register, whose input is a linear function of its states/internal bit contents, specifically the XOR of a few selected bits and whose output is the last bit contained by it. The selected bits are defined by its feedback polynomial. On each clock cycle, the register shifts its contents and appends a new bit derived from the XOR of the tapped bits.



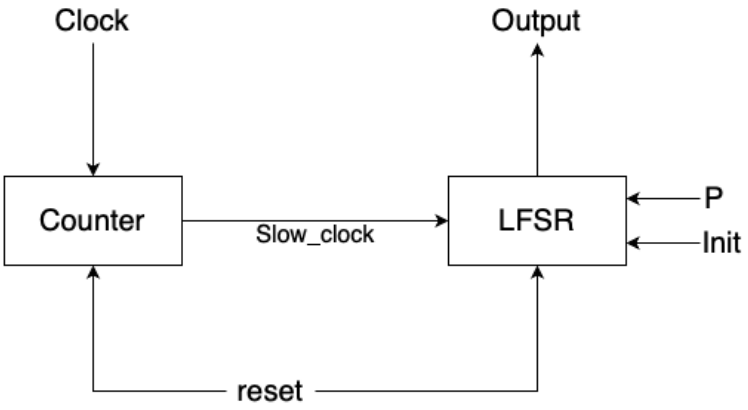
A 5-bit LSFR can be described as shown in the figure above. If s_{i-5}, s_{i-4}, s_{i-3}, s_{i-2} and s_{i-1} denote the current states of the flip flops FF_4, FF_3, FF_2, FF_1 and FF_0 respectively, the next state s_i can be determined as:

$$s_i = P_0 \cdot s_{i-1} + P_1 \cdot s_{i-2} + P_2 \cdot s_{i-3} + P_3 \cdot s_{i-4} + P_4 \cdot s_{i-5}$$

where P_i denotes whether the corresponding bit in FF_i will be selected or not. This information is also encapsulated in the feedback polynomial of the LFSR: $x^5 + P_4x^4 + P_3x^3 + P_2x^2 + P_1x^1 + P_0$.

Implementation

A single 5-bit LFSR was chosen to be implemented. The initial state of the registers was 10101 and the feedback polynomial was $x^5 + x^2 + x^1$. An LFSR generates a single bit for each clock cycle. Since it would be infeasible to physically observe the output, the clocking rate of the LFSR was reduced to 0.1/1 cycle per second by using an artificial clock signal as input to the LFSR. The artificial clock signal was generated using a counter which flips the signal level of the artificial clock every 49999999/499999999 clock cycles.



Here, reset is used as an external input, while P and Init are constants. Both slow_clock and output are outputted from the board as LED signals.

Hardware: The synthesis and implementation was done for a Zynq 7000 ZC702 Evaluation Board (xc7z020clg484-1) board using Xilinx Vivado. The source codes and constraint file have been attached to this report.

Observations and Conclusions

- LFSR behaves as expected, with a deterministic pseudo random output
- Proper selection of the feedback polynomial is essential to ensure full-period sequences.
- We can have multi-bit outputs by using different LFSRs with different period for each bit

References

- Xilinx. http://xilinx.com
- ASIC World Tutorials – LFSR: https://www.asic-world.com/verilog/tut_lfsr.html
- Douglas R. Stinson. 1995. Cryptography: Theory and Practice (1st. ed.). CRC Press, Inc., USA.
- IEEE 1364-2005. *Standard for Verilog Hardware Description Language*.