Design and IP core based implementation of a programmable 8-bits random sequence generator

Madhusudan Dey, Abhishek Singh
Variable Energy Cyclotron Centre, Kolkata
email: msdey@veccal.ernet.in, singhabhishek@veccal.ernet.in

Introduction

An FPGA based 8-bit random sequence generator has been designed using IP-CORE and a window based GUI for control has also been designed with RS232 interface.

In the nuclear physics application, there is a need to generate the random events with various distributions such as Gaussian, exponential to test the design for nuclear data acquisition. This work is an effort to simulate those events.

PRBS sequence has been generated using an LFSR (Linear Feedback Shift Register) implementation. An LFSR is a class of devices known as state machine. It is a shift register whose input bit is a linear function of its previous state. The only linear functions of single bits are XOR and XNOR. Thus it is a shift register whose input bit is driven by XOR or XNOR of some bits of overall shift register value.

Mathematical Understanding of LFSRs:

In an LFSR, the next state is some function of its previous state. The next state is deterministic. For an LFSR of size 'n' there can be maximum up to 2^n-1 different values which are periodic. The LFSR with length 2^n -1 is called maximal length LFSR. Thus the sequence is not completely random in the sense; it is periodic with some period of length referred to as its length. There can be many different patterns of same length, depending upon the tap positions of shift registers. Any LFSR can be represented by a polynomial of degree 'n', where 'n' is the LFSR size:

$P(X) = b_nX^n + b_{n-1}X^{n-1} \ldots\ldots\ldots + b_2X^2 + b_1X^1 + b_0$

Where the exponent refers to the tap positions in the shift register chain. The coefficients refer to tap weights, which are 1 for connection and 0 for no connection. b_0 and b_n are always 1. [1]

It can be shown that if the polynomial is a primitive polynomial, it will yield a maximal length sequence. For a polynomial of degree 'm', a primitive polynomial is one which cannot be factored further and also it should be a factor of X^n+1 for any minimum value of 'n', where n= 2^m–1. There can be many such primitive polynomials of degree 'm', corresponding to which there can be many maximal length tap positions. [2]

Implementation of LFSRs:

There are two different ways to implement the LFSRs with registers and XOR gates:

1) Type-I. 2) Type-II.

Type-I: In a type-I implementation of LFSRs the feedback is given to the leftmost register with input from some of the bits of register output. It is also known as **Fibonacci** type LFSR.

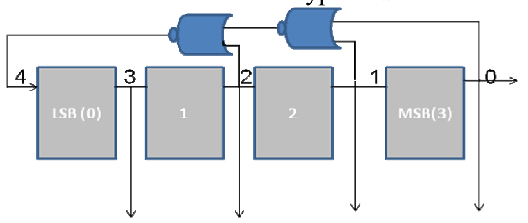

Fig.1 Type-I LFSR

Polynomial representation of above Fibonacci LFSR is:

$$P(X) = X^4 + X^2 + 1$$

Coefficient for X^3 is 0 since no tapping in that position.

Type-II: In a type-II implementation, XOR gates are feedback to different register inputs. Simply put, the XORs are inside the shift registers. This type of implementation is faster than type-I, because it involves less delay in the feedback path from input to output. This is also called **Galois** type LFSR.

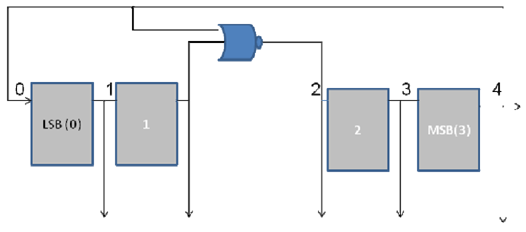

Fig.2 Type-II LFSR

Polynomial representation of above Galois LFSR is:

$$P(X) = X^4 + X^2 + 1$$

Coefficients of X^3 and X^1 are 0 since no tapping in those positions.

Hardware implementation of the design:

The digital circuit shown in fig.3 represents the schematic view of the digital circuit implemented in VHDL. the design has been implemented successfully and verified in the simulator. The simulation results are also shown which shows the variation of frequency depending on the user value input into the Clock divider module.

A GUI has been developed in Visual Studio 6.0 which is used to select the four Frequencies: 5, 10, 25, 50 MHz as PRBS generation rates.

Schematic diagram of the design:

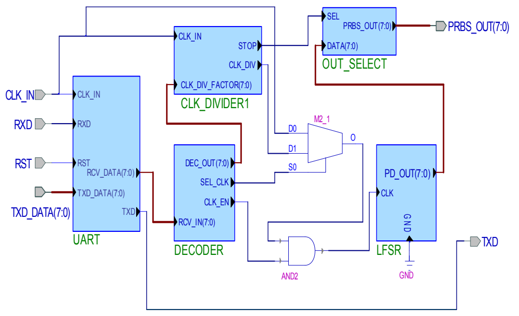

Fig.3 Schematic view of digital design

The important design modules are:

- UART module which is used for RS232 interfacing.
- LFSR IPCore used for generating the PRBS output.

The LFSR used in above design is an 8-bit Fibonacci type, with following implementation:

$$P(X) = X8 + X4 + X3 + X2 + 1.$$

This has been chosen for maximum length i.e. 255.

Simulation results:

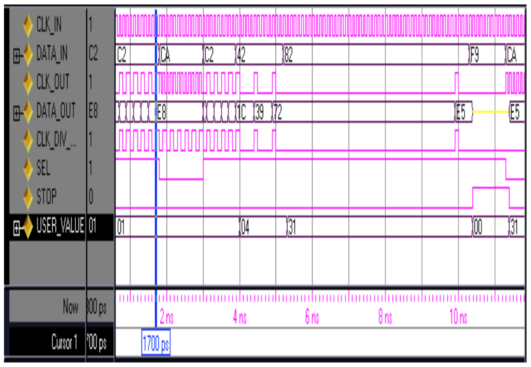

Fig.4 Simulation results

Distribution of the random events:

The random numbers thus generated in the above hardware are uniformly distributed over the range 0 to 255, since all the outputs will be having same probability of occurrence. So, we need to transform those events to poisson's distribution using the transformation below:

$$\boxed{E = -(1/a)*\ln(U)}$$

Where U is uniformly distributed random event, E is an exponentially distributed random event and 'a' is a constant parameter known as rate parameter.

The temporal variation of events in any nuclear experiment is a poisson process which can be described by an exponential probability distribution. In our next attempt, we will try to generate the exponential distribution of events by implementing above transformation on the hardware.

References:

[1] http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm

[2] http://www.math.cudenver.edu/~wcherowi/courses/m5410/m5410fsr.html