A Pulse Generator with Poisson-Exponential Distribution for Emulation of Radioactive Decay Events

Alejandro Veiga and Enrique Spinelli

LEICI, Facultad de Ingeniería, Universidad Nacional de La Plata and CONICET, Argentina.

alejandro.veiga@ing.unlp.edu.ar

Abstract—We present an FPGA-synthesizable version of a pseudo-random pulse generator that can be used to emulate radioactive source activity. It is intended for debugging real-time digital pulse processing applications, beyond the capabilities of periodic generators. The proposed module delivers a discrete random sequence that follows the Poisson inter-arrival distribution. Operation is based on a barrel-shifted maximal-length linear feedback shift register, operating as uniform random number generator, followed by an implementation of the Bernoulli trial to emulate exponential inter-arrival times. Due to its simple design, it can operate at high clock frequencies, providing a minimum time between events of two FPGA clock cycles operating at full-speed. A small footprint Verilog module is proposed for embedding in digital processors. Attainable performance and required resources are calculated. Additionally, it is shown how digital output pulses can be width-modulated to generate, with minimum conditioning, the analog signals present in a spectroscopy detection chain.

Index Terms—random events; nuclear spectroscopy.

I. INTRODUCTION

Non-periodic behavior of nuclear radiation detectors can be challenging for a digital instrumentation system. Nowadays, sophisticated digital pulse processors (DPP) are broadly applied to pulse shape discrimination, pulse height analysis and time-to-digital conversion techniques. Since those systems and algorithms are becoming more complex, it is useful to have a precise emulation of the detection system that can be used for debugging, reducing the risk of radioactive source use. But in order to be useful, the emulator must statistically conform the behavior of real experimental data.

DPPs are designed with a processing throughput equivalent to the average rate of interesting events. However, as they are independently emitted, two consecutive events have a certain probability to arrive very close, or very far, to each other. This situation is hard to reproduce with a periodic signal generator. If a periodic high-frequency signal is used to emulate two close events, the DPP will not be able to handle the unreal data rate. Some commercial generators can produce programmable pulse trains, which may be useful to test some states, but they can't reproduce all race conditions that can occur during run-time. In certain circumstances the behavior of an electronic system may be dependent on the timing of events, producing a bug when events do not happen in the expected sequence. These situations may be rare and may arise after a long time of operating conditions.

With that issue in mind, a digital circuit that emulates the behavior of a radioactive source is proposed. Low resource FPGA implementation enables standalone operation or embedding in complex DPP systems. Besides its practical applications in debugging, it can be useful for the calibration of nuclear instruments and for teaching purposes, as it can emulate any kind of Poisson arrival in real-time.

In the next section a fast digital method for emulating inter-arrival times is proposed. As the design depends on a uniform random number generator (URNG) for operation, in the following section an efficient implementation is presented and compared with an alternative often cited in literature. In the last section some additional features that broaden the application field are included, like output conditioning to emulate analog spectroscopy signals and pulse-width modulation to emulate detector geometry.

II. EMULATING POISSON INTER-ARRIVAL TIMES

It is well known that the number of decay events per second from a radioactive source can be modeled as a Poisson process [1], [2]. A Poisson process is a stochastic counting process that expresses the probability of a given number of events occurring in a fixed interval of time, if these events occur with a constant average rate and independently of the time since the last event. The Poisson distribution arises in connection with the Poisson processes. A discrete random variable X is said to have a Poisson distribution with parameter $\lambda > 0$, if, for $k = 0, 1, 2, \dots$, the probability mass function of X is given by

$$f(\lambda; x) = \frac{(\lambda t)^k e^{-\lambda t}}{k!} \quad (1)$$

For a given $t \geq 0$, the probability distribution of the process is a Poisson distribution with parameter λt . Here $\lambda > 0$ is called the rate of the Poisson process. The expected value of a Poisson-distributed random variable is equal to λt and so is its variance. Under these circumstances the probability of an occurrence in a time interval is proportional to the length of the time interval, the occurrences are distributed uniformly on any interval and the probability of simultaneous occurrences equals zero. It can be shown that the time between each pair of consecutive events has an exponential distribution with parameter λ and each of these inter-arrival times is assumed to

be independent of other inter-arrival times. Probability density function results

$$f(\lambda; x) = \lambda e^{-\lambda t} \quad (2)$$

In a digital generator, in order to emulate Poisson inter-arrival behavior, exponential values can be generated by inverse transform sampling [3], [4],

$$t = -\frac{1}{\lambda} \log(U) \quad (3)$$

where U is a uniform random variable in $[0,1]$ and λ is the mean of the exponential [5], [6]. In this way the time until next event can be calculated. However, if outcome results in a time smaller that the time required for the calculation, the event will be lost. A minimum inter-arrival time is imposed by eq. 3 calculation, limiting the capabilities of the generator. In order to solve that problem, multiple inter-arrival times could be calculated at average rate and buffered. This would require a large memory to guarantee a low enough probability of emptying it at run time.

A more inexpensive way to reproduce the required behavior is to implement a hardware simulation of the actual emission process. That can be accomplished by running the Bernoulli trial [7] at high speed. This random experiment is characterized by only one parameter p , which represents the probability to have a success. The simplest way to obtain a Bernoulli outcome is by generating a uniform pseudo-random number U in the range $[0, 1]$ and then comparing it with the parameter p . Finally, if $U < p$, then a success event is obtained. In order to reproduce Poisson behavior and be applicable to our case, the following conditions must be met:

- 1) Time must be sliced in short intervals where only one event can occur.
- 2) Probability of event in an interval must be small and independent of previous events.

Given N intervals per second and p probability of event in that interval, the rate of the generator results

$$\lambda = Np \quad (4)$$

This is a form of random variate generation that transforms a uniform random sequence to produce a sequence with the exponential distribution. The shorter the interval of the simulation, the closer two independent events can be generated, resulting in a better time resolution of the pulse generator. Within that interval, a uniform pseudo-random number must be generated and Bernoulli trial run, what in practice results much faster and less resource consuming than calculations required for eq. 3. The proposal is to design a synchronous digital circuit that runs the trial once every clock cycle at full FPGA speed. To speedup the whole operation, the circuit can be split in two stages, as shown in Fig.1. The figure presents a layout where the uniform random number generator can be replaced in order to test two possibilities that will be presented in the next section. Both stages are pipelined implying a two cycle operation with a throughput of one possibility of event per cycle, resulting a minimum inter-arrival time of one clock cycle (two clock cycles for non-retriggerable output).

Fig. 1. Two-stage pipelined implementation of the proposed generator. Every positive edge of the PLL driven CLK the first stage produces a 32-bit uniform random number U , and the second stage updates 1-bit output of the Bernoulli trial A . Minimum inter-arrival time is two clock cycles for non-retriggerable output.

```

input [31:0] U;
output reg A;
input [4:0] RATE;
wire [31:0] MASK;
assign MASK = (1<<RATE) - 1;
always@ (posedge CLK)
begin
    if ((U[31:0]&MASK)==MASK) A <= 1;
    else A <= 0;
end
    
```

Fig. 2. Modified Bernoulli trial with variable rate.

As any bit sequence of the URNG is equally probable, the trial test can be improved using a 2^n -ones mask generated by n -bit $RATE$ register. Output is true (event is generated) if last n bits of the random number equal the mask (i.e. are all one). Registered output A is one cycle width. Fig. II presents a Verilog section of code that implements this feature. Note that $(1 \ll n) - 1$ is a simple syntax for a n -ones mask.

The number of bits of random number U determines the lowest rate of the generator. In this case 32 bits were selected. A 32-bit pattern will occur, in average, once every 2^{32} cycles, what is approximately 10 seconds when running at 400 MHz clock frequency. If a lower event rate is required, the number of bits of U can be increased.

For experiments that require long-term debugging, the period of the URNG must be long enough to guarantee non repeatability. Running at 400 MHz, a one day period would require at least a 2^{45} -cycles period length. Therefore, a 64-bit period was implemented in order to consider the generator as non periodic for all practical purposes.

The highest rate of the generator could be as high as clock frequency (one bit mask), but in this case the output would become periodic. In order to preserve Poisson behavior, λ must be much smaller than the number of intervals per second. A minimum mask of 7-bit guarantees two orders of magnitude in this relation and still provides Poisson output rate as high as 3 MHz when operating at 400 MHz. It must be noted that the output rate is logarithmic with the number of bits of the mask, providing 32 different rate options.

```

wire [31:0] U;
reg [95:0] LFSR;
always@(posedge CLK) begin
    integer i;
    LFSR[95:32] <= LFSR[63:0];
    for (i=0; i<32; i=i+1)
        LFSR[31-i] <= (LFSR[95-i] ^ LFSR[93-i]
            ^ LFSR[48-i] ^ LFSR[46-i]);
end
assign U[31:0] = LFSR[31:0];

```

Fig. 3. Proposed Verilog code for a barrel-shifted 96-bit LFSR.

III. SINGLE CYCLE UNIFORM RANDOM NUMBER GENERATORS

In order to be able to run the Bernoulli trial at fast speed in one cycle, a 32-bit uniformly distributed random number U must be generated in one clock cycle. Considering the hardware simplicity of the Bernoulli trial, the URNG will become responsible for the speed limit the pipeline can reach. We propose a fast URNG based on applying the barrel-shifting technique to a Maximum Length Linear Feedback Shift Register (ML-LFSR). Our alternative will be compared with an FPGA implementation of a maximally equidistributed combined Tausworthe random number generator.

A linear-feedback shift register (LFSR) [8] is a shift register whose input bit is a linear function (usually XOR) of its previous state. If well designed, they can produce a sequence of bits which appears random with a very long cycle. The initial value of the LFSR is called the seed and, because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. A maximum length LFSR (ML-LFSR) [9] generates a pseudo-random binary sequence that is also periodic, but reproduces every binary sequence (except the zero vector) that can be represented by the shift registers (i.e., for length- m registers they produce a sequence of length $2^m - 1$). As a consequence, ML-LFSRs have a continuous uniform distribution.

In practice, a ML-LFSR can be shifted in one clock cycle, but in that case it will only provide one pseudo-random bit at a time. In order to generate a 32-bit random number, either 32 clock cycles would be required or 32 parallel generators must be implemented, as in [10]. Our proposal is to design a barrel-shifted LFSR that can generate 32 pseudo-random bits in one clock cycles. For that to be possible, a ML-LFSR with all taps bigger than 32 is required. In our case a 96-bit implementation was selected (taps 96, 94, 49, 47), which is barrel-shifted 32 times in one cycle. Fig. III provides a register-level Verilog implementation of the proposed generator. The last 32 bits of the LFSR provide our 32-bit random output. The remaining 64 bits provide a 2^{64} period length, suitable for long term

```

reg [31:0] U, S0, S1, S2;
always@(posedge CLK) begin
    S0 <= (((S0 < 32'hFFFFFFFE) << 12)
        ^ (((S0 < 13) ^ S0) >> 19));
    S1 <= (((S1 < 32'hFFFFFFF8) << 4)
        ^ (((S1 < 2) ^ S1) >> 25));
    S2 <= (((S2 < 32'hFFFFFFF0) << 17)
        ^ (((S2 < 3) ^ S2) >> 11));
    U <= S0 ^ S1 ^ S2;
end

```

Fig. 4. Verilog version of Tausworthe generator as proposed in [12].

experiments. Given tap distribution, number of simultaneous shifts could be increased up to 46, reducing the output rate to one event every two days with a period length of a month.

In order to compare the performance and required resources of our URNG, a maximally equidistributed combined Tausworthe random number generator [11] was implemented as proposed in [12]. Tausworthe generators are a kind of multiplicative recursive generators that allow an easy and fast single cycle implementation. They have good statistical properties when they are combined. Fig. III gives a Verilog version of the C code proposed in the reference. Registers S0, S1 and S2 must be initialized to any nonzero values. The generator has a period length of approximately 2^{88} cycles. It requires three 32-bit registers for internal operation and a fourth one to store the 32-bit combined random number, 32 more registers than required for our generator.

IV. IMPLEMENTATION

Both stages of the Poisson generator were designed as a synthesizable register-transfer level (RTL) core, coded in Verilog hardware description language. A development platform for Altera FPGA was used to test its performance. Terasic DE0 kit [13] includes speed grade 6 Cyclone III 3C16 FPGA [14] with several peripherals and memory.

TABLE I
FPGA RESOURCES REQUIRED FOR IMPLEMENTATION OF THE GENERATOR
LC: ALTERA LOGIC CELLS, REG: REGISTERS, LUT: LOOKUP TABLES.

	f_{max}	LC	Reg	LUT
Tausworthe	428.6 MHz	128	128	96
Bernoulli Trial		69	1	69
Total		197	129	163
LFSR	462.3 MHz	97	96	45
Bernoulli Trial		69	1	69
Total		161	97	109

The Verilog model of the two-stage pipelined generator was compiled using Quartus II (32-bit) version 13.1.0 Web Edition. Table I summarizes the resources required for implementation of both pipeline stages, for two different URNG options. One of four on-chip phase-locked loop modules (PLL) was used to clock the generator. The PLL was programmed using Altera

Fig. 5. Analog circuit required to emulate the behavior of pulse shaping front-end electronics. Input A receives a short digital pulse from the FPGA. Output B emulates the output of a charge sensitive preamplifier and output C provides a pseudo-gaussian shaping of the preamplifier pulses.

ALTPLL IP Core. All timing constraints were met for PLL operating at 400 MHz. The maximum attainable frequency of operation, for Cyclone III grade 6 as reported by Altera TimeQuest Timing Analyzer, resulted 462.3 MHz for LFSR and 428.6 MHz for Tausworthe generator. The proposed LFSR implementation, besides requiring one less 32-bit register, demands 25% less logic elements and runs 8% faster than the Tausworthe implementation. The architecture of LFSR 4-taps feedback fits efficiently in Cyclone III 4-input LUTs [14]. The implementation requires only 1% of the logic cells (LC) available in Cyclone III FPGA. Its small footprint makes the design suitable for embedding in larger designs. Both implementations were experimentally verified to satisfy statistical properties. Operating at 400 MHz with RATE=16 (results λ of 6100 events/s), inter-arrival times were recorded using Ortec 566 Time to Amplitude Converter in 2 ms range and Tracor Northern TN-7200 Multi Channel Analyzer operating in 4096 channels. After 10 minutes, exponential result was fit and λ reconstructed to less than 1% error with nearly zero autocorrelation.

V. SHAPING THE OUTPUT AND EMULATING CORRELATED EVENTS

In order to broaden the application spectrum of the generator, several additional features are proposed. Besides dealing with timing situations, these features can help to more accurately emulate the behavior of a real detector and associated front-end electronics.

An analog output stage, as proposed in Fig. 5, is an implementation of a simple pulse shaping network for spectroscopy. A charge sensitive preamplifier for the short digital input pulse (output B) is followed by a CR-RC shaping amplifier without baseline restoration (output C) that produces a pseudo-gaussian pulse with amplitude proportional to pulse energy. Maxim 4477 is a dual wideband, low-noise, low-distortion operational amplifier suitable for this application. The first stage feedback resistor value must be calculated to prevent pileup saturation at high output rates. Fig. 6 (right) shows how a spectroscopy system can be tested in different stages with the analog outputs of the pulse generator.

Fig. 6. LEFT: Example application to debugging a spectroscopy system. Signals A, B and C are used to test charge preamplifier, shaping amplifier or digital processor. RIGH: Single (1) and double-width (2) modulated output pulse (A), preamplifier output (B) and gaussian-shaped amplifier signal (C).

Additionally, a simple pulse-width modulation technique can be used to emulate different trajectories or interactions between radiation and detector. Different pulse energies can be produced assigning a random width of more than one clock cycle to the pulse. Once integrated and shaped by the amplifiers, quantized-amplitude pulses can be used to test multi-channel analyzers and digital pulse processors. This technique is schematized in Fig. 6 (left). Note that widening the output pulse degrades timing resolution.

Finally, it is worth mentioning that several generators can be combined in order to emulate more elaborate situations, as correlated events. For that purpose seeds can be externally generated and stored if repetition of the sequence is required.

REFERENCES

- [1] F. A. Haight, *Handbook of the Poisson Distribution*, New York: John Wiley & Sons, pp. 119-120, 1967.
- [2] F. Cannizzaro, G. Greco, S. Rizzo and E. Sinagra, "Results of the measurements carried out in order to verify the validity of the poisson-exponential distribution in radioactive decay events," *The International Journal of Applied Radiation and Isotopes*, vol. 29, pp. 649-652, 1978.
- [3] R. C. Cheung, "Hardware Generation of Arbitrary Random Number Distributions From Uniform Distributions Via the Inversion Method," *IEEE Transactions on VLSI Systems*, vol. 15, pp. 952-962, 2007.
- [4] L. Devroye, *Non-Uniform Random Variate Generation*, New York: Springer-Verlag, pp. 27-29, 1986.
- [5] D. E. Knuth, *Seminumerical Algorithms. The Art of Computer Programming*, Addison Wesley, vol. 2, pp. 128-130, 1969.
- [6] M. Arkani, "A Flexible Multichannel Digital Random Pulse Generator Based on FPGA," *World Journal of Nuclear Science and Technology*, vol. 3, pp. 109-116, 2013.
- [7] A. Papoulis, *Bernoulli Trials. Probability, Random Variables and Stochastic Processes*, 2nd ed., New York: McGraw-Hill, pp. 57-63, 1984.
- [8] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," *Xilinx Application Note*, 052, 1996.
- [9] P.H. Scholfield, "Shift Registers Generating Maximum-Length Sequences," *Electronic Technology*, vol. 37, pp. 389-394, 1960.
- [10] D. Ponikvar, "Generator of Pseudo Random Pulses," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, vol. 83, pp. 295-299, 1993.
- [11] R.C. Tausworthe, "Random Numbers Generated by Linear Recurrence Modulo Two," *Mathematics and Computation*, vol. 19, pp. 201-209, 1965.
- [12] P. L'Ecuyer, "Maximally Equidistributed Combined Tausworthe Generators," *Mathematics of Computation*, vol. 65, pp. 203-213, 1996.
- [13] Altera DE0 Board, <http://de0.terasic.com/>
- [14] Altera Cyclone III Device Handbook, https://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf