Statistical Data Generation System for Scientific Applications

A. Abba, F. Caponio, P. Baruzzi, A. Geraci, G. Ripamonti
Politecnico di Milano, Dipartimento di Elettronica, via C. Golgi 40, Milan MI 20133, Italy

***Abstract* - An innovative algorithm aimed to sequence of numbers generation according to a programmable statistical distribution is presented. Main features of the technique are high efficiency, low requirements in terms of implementation resources and high generation rate. Possible applications range from scientific simulations to system testing setups. In particular as reference case study, we adopted simulation of events generated by radioactive decay processes that are at the forefront in many application and research areas in medicine as in physics. The algorithm has been implemented in a low cost multi-FPGA system. A generation rate one order of magnitude higher with respect to modern PC-based solution has been achieved.**

Keywords – Pseudorandom number generation, Deterministic random bit generation, Simulation, Testing.

I. Introduction

Many methods and techniques for pseudorandom number generation (PRNG), also known as deterministic random bit generation (DRBG), are well known and consolidated [1-2]. These are algorithms for generating sequences of numbers that approximate the properties of random numbers. Although sequences that are close to truly random can be generated, pseudorandom numbers are fundamental in practice for simulations (e.g., of physical systems with the Monte Carlo method [3]), and are central in the practice of cryptography and procedural generation.

However, an increasing number of applications shows the necessity to simulate sequences of numbers that approximate at best properties of specific statistical distributions. This is the case, for example, of simulation of events generated by non-random physical processes and initialization of system testing setups. Since this is now at the forefront in many research areas in medicine as in physics, we adopted this application as reference case study to describe the proposed technique.

The radioactive decay is the process by which an atomic nucleus of an atom loses energy by emitting particles. The distribution of energy values of emitted particles depends on the source and is referred as its energy spectrum. Of course, radioactive decay is a stochastic (i.e. random) process on the level of single atoms, in that according to quantum theory it is impossible to predict when a given atom will decay. However, given a large number of identical atoms the decay rate for the collection is predictable, and for the most cases the emission instants follow a Poisson distribution. Consequently, in order to emulate a radioactive source, the primary task is to generate the emitted particle energy values, according to the source energy spectrum, and the occurrence times following Poisson distribution.

The generation of random data which follows a statistical distribution should be treated so that generated values are as representative as possible of the physical phenomenon that produces them. For example, the generation of energy of events emitted by a ^{60}Co isotope should correspond to about 80% probability of 1.33 MeV with respect to 1.17 MeV emission. This means that the emulated spectrum should grow approaching progressively from the beginning the final shape of the spectrum.

Of course, the requirement of properly emulated randomness must be combined with the need to find a method to generate data at high efficiency.

It is well know that a trivial way of generating random numbers that follow a given distribution consists in addressing the corresponding histogram by means of a white distribution. Each time the random number addresses a bin of the histogram, the corresponding count is decreased by a unit and the bin value is outputted. If the addressed bin count is null, no output is produced. The method is inherently not efficient since the probability of finding a non-zero bin count decreases with increasing generated numbers. Other algorithms require huge amounts of memory and are therefore not suited for embedded or low-cost systems.

We propose a novel technique for generating random numbers according to an arbitrary probability distribution with high efficiency, low requirements in terms of implementation resources and high generation rate.

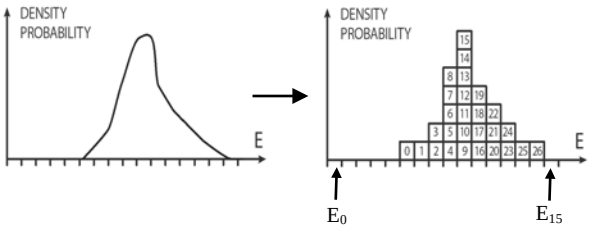

Fig.1 The plot shows a hypothetical emission energy spectrum of a radioactive decay process and its correspondent discrete representation by means of a histogram.

The algorithm has been implemented in a multi-FPGA setup that generates up to 75 Mevents per second, with a word size of 16 bits and quantized at 65,536 levels. The system can be interfaced by means of PCI-Express bus and used as co-processor providing test vectors to embedded hardware simulators or PC-based simulation software environments.

II. Methodology

As reference distribution, we consider the generic histogram depicted in Fig.1 that, for instance, represents the discretized energy emission spectrum $H(E)$ of a radioactive decay process. Aim of the technique is to generate numbers whose distribution grows approaching progressively from the beginning the shape in Fig.1. At basis of the process is a random number generator. In fact, any statistic variable x that is described by a density probability distribution $P(x)$, can be modeled by the cascade of a generator of uniformly distributed random numbers and the transform function $P(x)$ [4]. In this way, the quality of the generated statistic values depends only on the uniform number generator, which can be used for every emulated source that is characterized only by $P(x)$. Therefore, the problem is to transform a white spectrum into whatever kind of distribution. In order to simply explain how the algorithm works, let us consider that the reference histogram is composed by 16 bins, from E_0 up to E_{15}, with a maximum dynamic range equal to 16. The bin width is the spectrum resolution, while the dynamic range is the maximum height of each histogram column. The higher is the number of bins and the dynamic range, the better is the represented spectrum. However, increasing the accuracy of the spectrum is simply a matter of number of bits and this is not a problem using modern digital devices. Each column E_x of the histogram can be thought as composed by a number of small squares and represents the density probability that the event has energy between E_{x-1} and E_{x+1}; if bin x is twice higher than bin y, this means that there is twice the probability for an event to have energy E_x rather than energy E_y. The product of the column value by the bin width returns the probability. The ratio of the probabilities that an event has energy in a certain interval rather than in another one is simply the ratio between the corresponding areas below the density probability curve.

As in Fig.1, squares constituting columns are sequentially numbered. Consider the simplified case in which the total number of squares under the curve is a power of 2, e.g. $2^5=32$. Using 5 bit in the random number generator, all the 32 numbers can be obtained with the same probability, i.e. the random numbers map completely the area under the spectrum curve. Every time a random number is generated, the algorithm searches the number in the spectrum area and delivers the corresponding bin number x thus indicating the corresponding energy value E_x. If we consider again a generic x bin two times higher than a y bin, since random numbers map all the squares with equal probability, there is twice the probability that the random number picks up a square in x rather than in y column, which means that generated pulses with E_x energy are twice those with E_y energy.

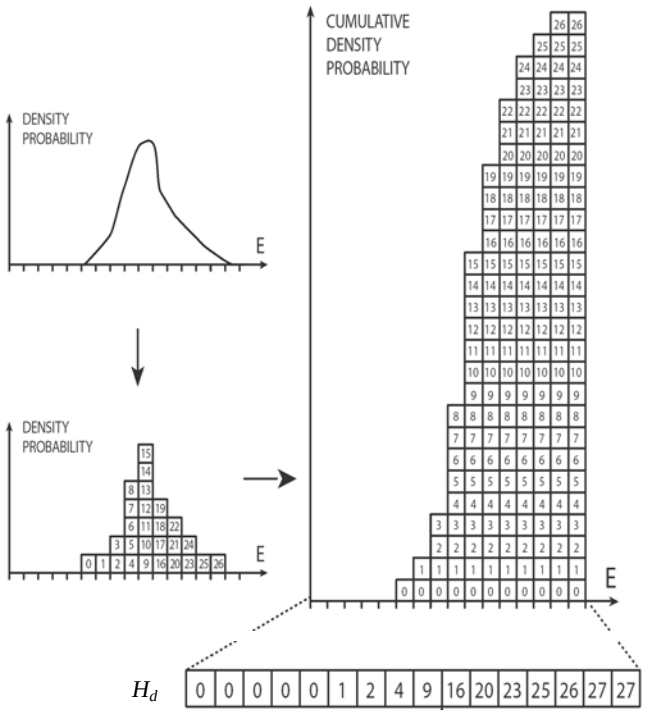

Fig.2 Conversion of the histogram of the reference probability density into the corresponding cumulative histogram.

Operatively, the histogram is converted into the equivalent cumulative histogram (Fig.2), i.e. the cumulative energy spectrum $H_c(E_x)$ is computed as integral function of the energy spectrum $H(E)$. An array is loaded with the cumulative spectrum and only one memory cell per bin is required. Using the cumulative spectrum, it is still possible to identify the bin that contains the generated random number by means of an extension of the described algorithm. For instance, (see Figs. 2 and 3), if the random number is 18, it is easy to see that it belongs to the memory cell E_{10}, which contains a number that is higher than 18 (20), while the preceding cell contains a number that is lower (16); this means that bin E_{10} contains the squares that go from 16 to 19, exactly the range in which 18 belongs. So the output energy value correspondent to the random number 18 is 10.

The fundamental advantage of this approach is the lack of required memory. In fact, alternative solutions could be faster but heavier for memory resources. For instance, using a memory composed by a number of cells equal to the number of bins multiplied by the dynamic range, that is the maximum number of squares available to draw the spectrum, each cell would contain the corresponding bin number, while the random number would be used to address the memory. In this way, no search algorithm is necessary since the data bus directly returns the bin number when the random number addresses the memory and the procedure is significantly faster. Nevertheless, this solution has the serious drawback of the memory occupancy. In fact, considering a histogram represented by 65,536 bins (i.e. 16 bit) with dynamic range equal to 65,536 (i.e. 16 bit), 8 GB of memory are necessary.

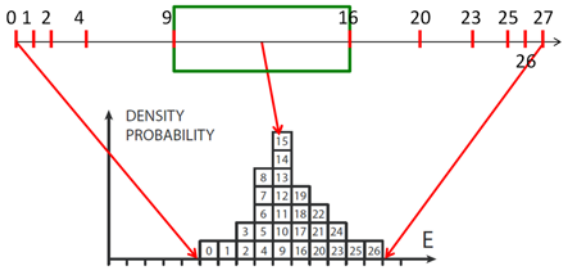

Fig.3 The marked rectangle represents the range of values of the bin E_{10}, which is the largest and consequently most likely to contain the randomly generated number. This non-uniform probability is the key of conversion from white to shaped distribution.

That has to be compared with the proposed approach, which in same operative conditions requires only 256 kB of memory.
In order to increase memory saving, we compress data storing in the array only bins with counts greater than a fixed threshold. In this case, it is important to remap each bin number of the cumulative spectrum with the correct energy value. A look-up table (LUT) is therefore necessary of size equal to the number of non-zero bins of the spectrum. Only if less than half of bins are non-zero elements, the compression is useful and performed.

III. Implementation

The system implementation is partitioned between the generation of the vector H_d, which is calculated storing only the top value of each bin of the histogram H_c (Fig. 3), and the generation of random number. The first task is performed only once and consequently has no impact on generation efficiency. On the contrary, the generation of random numbers is a critical issue in terms of operation frequency and therefore it has to be performed by means of dedicated hardware resources. Considerations on operating frequency, power dissipation and memory access bandwidth led us to choose a FPGA instead of temporal computing devices such as DSP or GPU based solutions. As above stated, conversion from white to any distribution is performed by searching the position p of the generated random number x into the vector H_d and returning p as output. Therefore, two are the main tasks of the FPGA device: the generation of random numbers x and the search of their position p into the vector H_d through a modified version of the binary search algorithm.

Efficient and accurate algorithms for random number generation are fundamental in many fields of application [5], from process simulation to cryptography. In the present application, the linear feedback shift register (LFSR) algorithm has been implemented that is a machine independent algorithm characterized by arbitrary long repetition periods, excellent statistic properties, high generation speed and limited resource expense [5]. It only needs an m-bit shift register and 1 to 3 XOR gates, and thus the resulting circuit is very small and its operation is extremely simple and fast. Furthermore, since the period grows exponentially with the size of the register, we can easily generate a large non-repetitive sequence (e.g. with a 64 bit generator running at 1 GHz, the period is more than 500 years).

Task of spectrum modulation is to look in which interval between two numbers of the vector H_d a random number is included. A very high performance algorithm to perform that in a sorted vector is the binary search [5]. Comparing the target to the middle item in the list, if the target does not match but is greater, the comparison is repeated in the upper half of the list. Otherwise, the lower half of the list is considered. The method halves the number of items to check each time, reaching convergence in logarithmic time with respect to the number of iterations.

IV. Hardware Implementation

Before developing a customized hardware implementation, the generation algorithm has been validated on Xilinx FPGA devices Virtex-5 FX110T and Spartan 6 LX-25. Let's consider the storage of distributions with dynamic range of 24 bit and resolution of 16 bit, i.e. 65536 bins. Each distribution needs a memory allocation of 1.57 Mbit to be stored. With 8 Mbit internal dual-port memory, the Virtex-5 device allows the storage of 5 distributions, namely the simultaneous operation of 5 independent generators. Considering that memories are dual-port and operating frequency can be 300 MHz, the minimum rate is 83 Mnumbers generated per second that corresponds to the maximum value of 17 clock cycles for every binary search.

Drawbacks of this solution are the cost of the device (above 1000 USD) and also the limit of 5 generators that can run simultaneously. The use of cheaper but smaller devices organized as an array has been investigated. The implemented algorithm within the FPGA consists only of a state machine and two comparators for each generator. The Xilinx FPGA Spartan-6 LX-25 is a low-cost device that has enough resources to implement the algorithm of random number generation and research but not enough memory (1 Mbit) to store even only one distribution. Consequently external asynchronous SRAM resources were attached to the FPGA device with the limit of pins available for connection. In practice, the adopted BGA FG484 package of the FPGA device leaves less than 260 pins available and each module of 2 Mbit selected low-cost RAM needs 24 data, 16 address and 2 control lines, which means 6 SRAM at most connected. At the operating frequency of 100 MHz, the single FPGA device should access memory 34 times at worst, which means a minimum rate of 17 Mnumbers generated per second.

The cost of a single computing cell, i.e. FPGA device – SRAM modules – configuration memory, is below 100 USD that is more than one order of magnitude less than Virtex-5 solution. Of course, the generation rate is sensibly slower, but 6 generators can run simultaneously. Doing a cost/benefit analysis, we decided to adopt this second approach.

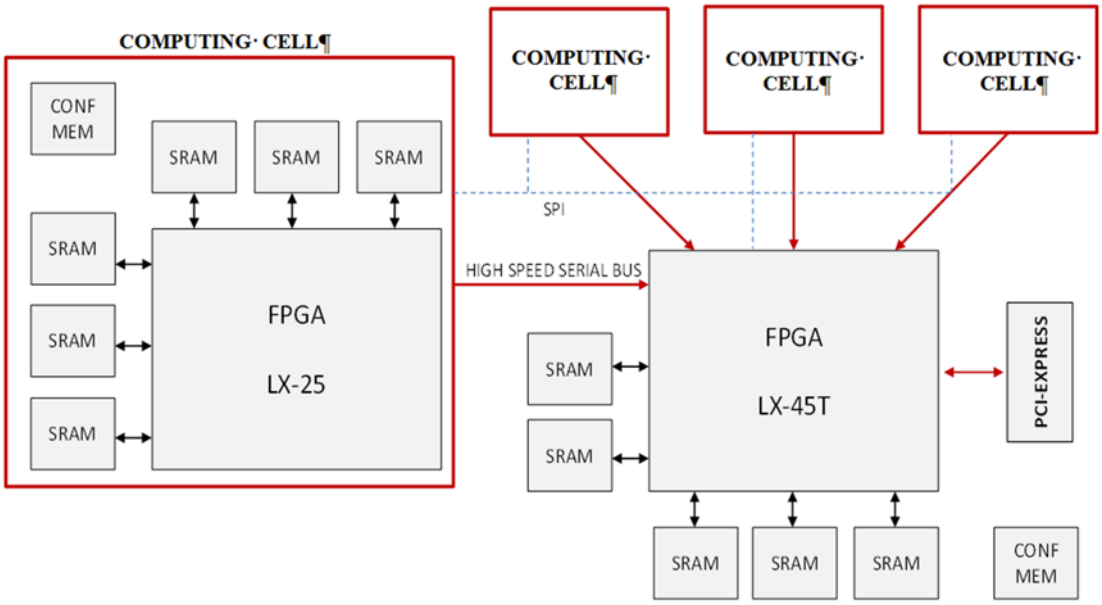

Fig.4 Block diagram of the proposed multi-FPGA architecture. The computing cell plays the role of emulators and the LX-45T device is also devoted to manage the communication to the host PC and to system clocking. The link between communication device and computing cell is performed by high-speed serial bus. The SPI bus is used to initialize the reference distribution and access to local register file in each computing cell.

The number of linked computing cells is limited by the adopted PCI-Express communication bus. In a Xilinx Spartan 6-45T FPGA device with built-in PCI-Express blocks, we implemented a master DMA controller, whose transfer rate to host PC is just below 200 MB/s per lane. Since we decided to use just 1 lane in order to make the system compliant with common 1x PCI-Express slots, the maximum transfer rate of 16 bit random numbers is 100 Mnumbers/s that corresponds to a maximum number of 5 connected computing cells.

Overall, at a cost equal to ¼ with respect to the use of a single Virtex-5 FPGA device, the available system is much more versatile as it can simulate from 30 independent distributions at a rate of 3 Mnumbers generated per second to 1 distribution at a rate of 88 Mnumbers generated per second.

Figure 4 shows a block diagram of the realized prototype. There are 4 Xilinx FPGA Spartan-6 LX-25 devices, which implement the function of generation. Besides PCI-Express communication task to host PC, the Xilinx FPGA Spartan-6 LX-45 device also implements 5 generators. The connection architecture among computing cell LX25 and LX45T devices is a star configuration through serial bus lines operating at 300 MHz. Initialization of computing cells is performed by means of a SPI bus. Each computing cell contains 6 asynchronous SRAM modules, so as not to suffer from pipeline delay and realize a true random access. The SRAM access time is 10 ns.

In order to verify the convenience to use the system in place of available PC-based solutions, the developed algorithm was implemented in C language and run on a Core i7 945 PC over-clocked at 4.5 GHz. Using 1 core, the generation rate is about 5 Mnumbers/s. Parallelizing the algorithm, however, the rate does not increase significantly and settles down to 7.5 Mnumbers generated per second, probably because of the bottleneck due to memory access. Consequently, the proposed system shows a speedup of 12 times at a cost only half the CPU alone and with power dissipation of 15 W compared to 100 W of the PC based solution. In addition, there are no substantial benefits in the use of GPGPU since the local memory of each multiprocessor (both texture and constant) is too small to hold the cumulative vector H_d.

V. Conclusions

An algorithm for getting statistic properties from a histogram of events has been conceived and implemented. The algorithm has efficiency equal to 100% and has been validated through simulation also with reference to the specific application of emulation of radiation detection setups.

The system has been prototyped and is being fully tested on a processing digital platform based on a FPGA device.

The proposed solution based on FPGA has been shown to achieve a level of quality/price ratio even better than PC-based counterparts at the state of the art.

References

[1] M. Luby, "Pseudo-randomness and Cryptographic Applications", Princeton Univ. Press, 1996.
[2] L. Devroye, "Non-Uniform Random Variate Generation", Springer Verlag, New York, 1986.
[3] K. Binder, D.W. Heermann, "Monte Carlo Simulation in Statistical Physics", Springer, 2010.
[4] A. Papoulis, S.U.Pillai, "Probability, random variables and stochastic processes", McGraw-Hill, 4th Ed., 2002.
[5] D.E.Knuth, "Art of Computer Programming, Volume 2: Semi-numerical Algorithms", Addison-Wesley, 3rd Ed., 1997.