High Quality FPGA Optimized Random Number Generator

V. Navya Deepthi
Department of ECE
SRM University
Chennai, India

Ruhan Bevi
Department of ECE
SRM University
Chennai, India

V. Sai Keerthi
Department of ECE
SRM University
Chennai, India

ABSTRACT
In this paper we designed a new type of Random number generator by using shift registers and LUT with D-FF as input to it. The algorithm used to generate random numbers is realized using simple xor circuit and implemented on a Virtex II FPGA from Xilinx. This designed block indicate a good sequence of random numbers which is used in high-speed data processor, Testing Instruments, Finding Laser Range, Time-of-flight mass spectrometry experiments etc. The randomness of this type of RNG is tested using NIST statistical test and this method has produced good results.

Keywords: Random Number Generator, Field programmable gate array.

1. INTRODUCTION

The fast generation of random numbers is essential for many tasks. One of the major fields of application are Monte Carlo simulation, for example widely used in the areas of financial mathematics and communication technology. Monte Carlo applications are ideally suited to field programmable gate arrays (FPGAs) because of the highly parallel nature of the applications, and because it is possible to take advantage of hardware features to create very efficient random number generators (RNGs). In particular, uniform random bits are extremely cheap to generate in an FPGA, as large numbers of bits can be generated per cycle at high clock rates using lookup tables [1], or first-in-first out (FIFO) queues [2]. Many applications are reliant on uniform random numbers, such as monte-carlo integration, simulated annealing, and financial simulations. Such applications require huge amounts of processing power, while offering plenty of scope to exploit fine-grain and coarse-grain parallelism, and so are often ideally suited to implementation in FPGAs. In order to function correctly, these applications require many parallel streams of high quality, large period, uncorrelated random number generators, and efficient hardware implementations offer an attractive solution. However, existing methods such as LFSR [3], Tausworthe generators [11] and Cellular Automata based generators cannot provide all of these features at once.

In this paper we introduce a new class of random number generators where every bit of the state is equally random, allowing large numbers of parallel number streams to be produced from one large period generator. The new The new RNGs introduced here are part of a large family of RNGs, all of which are based on binary linear recurrences. This family includes many of the most popular contemporary software generators, such as the Mersenne Twister (MT19937) [4], the Combined Tausworthe (Taus113) [11], and TT800 [5]. Many of these software generators have been adapted for use in FPGAs [6]–[8], but the convenience of mapping an algorithm designed for word-level software comes at the cost of reduced efficiency, flexibility and low precision.

The rest of the paper is organized as follows. In Section II, we give an overview of previous work. Section III shows proposed method and description of block diagram of proposed method. Section IV shows synthesis results of the original and the improved implementation and elaborates on the excessive quality tests that we have applied. Finally, Section V concludes the paper.

2. PREVIOUS WORK

Existing technology proposed the two most common types of hardware random number generators which are Linear Feedback Shift Registers (LFSRs) and Tausworthe generators, both based on binary linear recurrences modulo 2 with primitive characteristic polynomials [7], and Cellular Automata (CA) generators. Other algorithms are used for Specialized tasks, such as the Blum Blum Shub algorithm [10] for cryptographic random numbers, but are not considered here. Binary linear recurrence based generators work by forming each new bit in the state from a linear combination of the bits in the previous state. The advantage of this type of generator is that the state-transition function is easily and efficiently implemented in LUTs, state xi+n can be determined from state xi in O(log2(n)) steps, and that the period length is only one less than the theoretical maximum. However, current generators from this family suffer from poor statistical quality and other drawback are:

- Implementations on FPGA limited by minimal delay.
- Limited to the operation frequency.
- Input clock is made half the time delay to pass through the delay line.

3. PROPOSED METHOD

The proposed architecture relies on multiple parallel programmable delay lines implemented as a series of programmable interconnection points and these delay lines are connected to shift register and LUT. The realization of these delay lines is presented in a precise adjustment of each delay can be made using dynamic reconfiguration by modifying the routing of each interconnection. As this sort of fine-grain adjustment is subject to process variations, a calibration

process that takes into account programmable interconnection points delay variations and clock skew within the global distribution tree is also proposed. Specifically, it shows how to create a family of generators using shift registers and LUT with delay flip flop as input, which is used to achieve to achieve high quality and high precision random values. The architecture of proposed method is shown in fig 1.

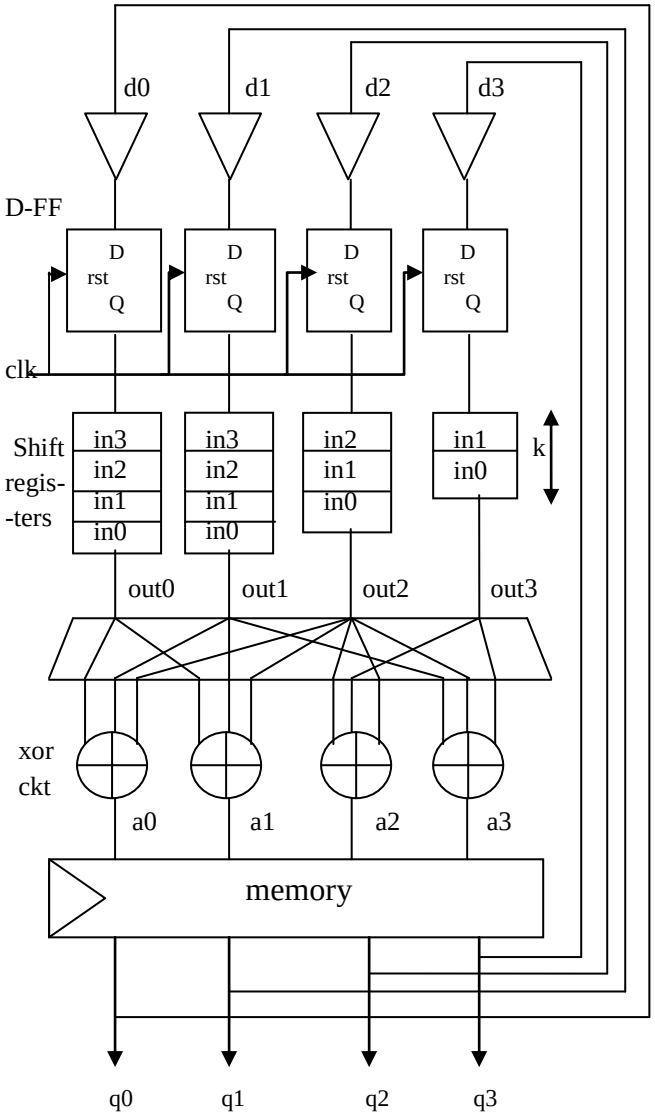

Fig 1: Architecture of proposed method

The architecture explains the ability to adjust the delay deference between each line compare with high resolution delay line architecture with in the FPGA. On the other hand, the dynamic range of a set of parallel lines is limited by the adjustment range of each delay line. Because it is unrealistic to aim for a dynamic range that Spans over the entire clock period, multiple parallel line sets can be used given that their inputs are also regularly delayed. The ouput of D-FF is given to shift registers where shifting of bits done and this values are given to xor circuit to get the random values.

The main benefits of the proposed hardware architecture are the following:

- Fast development process.
- Flexible Clock frequency.
- Greater design flexibility and reduced minimal delays.
- These proposed works achieve delay lines with a 1-ps resolution.

4. RESULT

The simulation of given random number generator is done in Xilinx and fig 2 shows the graphical representation of different resources versus registers of proposed RNG design.

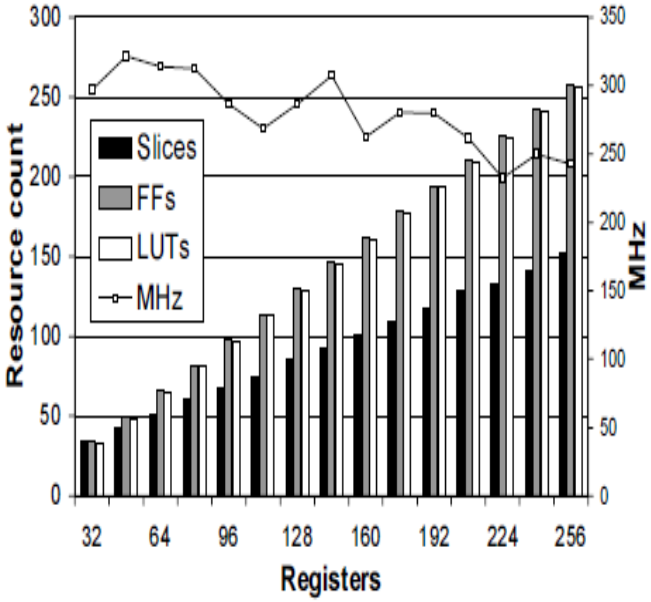

Fig 2: Graphical representation of different resources Vs registers of proposed design

A comparison of different parameters like quality, speed etc of proposed design with prior design is summarized in Table 1.

Table 1. Summary of Comparison of different parameters of proposed design with prior design

RNG	LFSR	LUT-SR	LUT-SR WITH D-FF
Period	64	64	64
FFs	291	68	66
LUTs	321	65	65
Frequency(MHz)	272	310	319
Throughput(GB/s)	9.0	20.4	19.8
Average Power (mw)	10	5.59	4.49

The design and testing of this random number generator is done on a FPGA system. There are several substantial test suites for testing RNGs [13]. We present results from the US National Institute of Standards and Technology (NIST) Statistical Test Suite for Random and Psuedo Random Number Generators for cryptography applications [12]. The NIST test suite produces a summary report for each file of random bits it tests. The table that follows is a result of running the NIST suite over the set of data produced by our RNG. The table consists of ten columns labeled c1 through c10 and a column containing, a Proportion column, and a column containing the name of test for that row.

Each test in the NIST suite is run over a large number of sets of bits from the file to be tested. The statistic that is generated from each of these runs is called a P-value and it represents the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested [12]. For example, if you got a P-value of 0.95 this would mean that 95% of the sequences produced by an ideal RNG would look less random than your sequence. Thus very small P-values are bad. Table 2 shows the NIST Statistical Test Results.

With these kind of tests one expects to get a range of P-value. The range from 0 to 1 is divided into ten bits, labeled in this report C1 through C10. The number in each of these columns represents the number of tests that had a P-value in the corresponding range. We would expect that a perfect RNG would have P-values evenly spread over the range 0 to 1. The column labeled P-Value is a chi-square test on the preceding spread of P-values over the range of 0 to 1. It is a P-values. The documentation that accompanies the suite indicates that: "If P-Value [the number in the column labeled P-Value]>=0.001, then the sequences can be considered to be uniformly distributed".

The Proportion column indicates the number of P-values that were above the 0.01 confidence interval. It is acceptable for a few individual tests to fail. The test suite will indicate a problem by flagging the Proportion number with an "*". In our case, none of the tests indicate failure.

TABLE 2. NIST RESULTS

STATISTICAL TEST	P-VALUE	PROPORTION OF P-VALUES
Frequency	0.657544	0.9868
Block-frequency	0.452066	0.9878
Cusum	0.278924	0.9911
Cusum	0.178883	0.9925
Runs	0.198952	0.9940
Long-Run	0.789169	0.9915
Rank	0.381589	0.9944
FFT	0.008760	0.9922
Aperiodic-Template	0.906420	0.9955
Periodic Template	0.183515	0.9923
Universal	0.917459	0.9864
Apen	0.236357	0.9868
Random-Excursions	0.890464	0.9843
Random-Excursions-V	0.666836	0.9924
Serial	0.773060	0.9888
Lempel-ziv	0.000107	0.9917
Linear-complexity	0.984963	0.9840

5. CONCLUSION AND FUTURE WORK

This paper presents a new type of FPGA Random Number Generator, using shift registers and LUT with delay flip-flop as input to it. These RNGs takes advantage of the ability to configure LUTs as independent shift-registers, allowing high-precision and high-quality long period generators to be implemented using only a small amount of logic. In addition the period and quality scale with the number of output bits, unlike generators adapted from software. A key advantage of the proposed method over previous FPGA optimised uniform random number generators is that they can be reconstructed using a simple algorithm. This RNG design passes the NIST statistical tests and has been proved to generate extremely high quality random sequence.

6. REFERENCES

[1] D. B. Thomas and W. Luk, "High quality uniform random number generation using LUT optimised state-transition matrices," J. VLSI Signal Process., vol. 47, no. 1, pp. 77–92, 2007.

[2] D. B. Thomas and W. Luk, "FPGA-optimised high-quality uniform random number generators," in Proc. Field Program. Logic Appl. Int. Conf., 2008, pp. 235–244.

[3] P. L'Ecuyer, "Tables of maximally equidistributed combined LFSR generators," Math. Comput., vol. 68, no. 225, pp. 261–269, 1999.

[4] D. B. Thomas and W. Luk, "FPGA-optimised uniform random number generators using luts and shift registers," in Proc. Int. Conf. Field Program. Logic Appl., 2010, pp. 77–82.

[5] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator,"ACM Trans. Modeling Comput. Simulat., vol. 8, no. 1, pp. 3–30,Jan. 1998.

[6] M. Saito and M. Matsumoto, "SIMD-oriented fast mersenne twister: A 128-bit pseudorandom number generator," in Monte-Carlo and Quasi-Monte Carlo Methods. New York: Springer-Verlag, 2006, pp. 607–622.

[7] F. Panneton, P. L'Ecuyer, and M. Matsumoto, "Improved long-period generators based on linear recurrence modulo 2," ACM Trans. Math.Software, vol. 32, no. 1, pp. 1–16, 2006.

[8] M. Matsumoto and Y. Kurita, "Twisted GFSR generators II," ACM Trans.Modeling Comput. Simulat., vol. 4, no. 3, pp. 254–266, 1994.

[9] Marc-Andre Daigneault and Jean Pierre David, "A High-Resolution Time-to-Digital Converter on FPGA Using Dynamic Reconfiguration," IEEE transactions on instrumentation and measurement,vol.60,no.6,2011.

[10] S. Konuma and S. Ichikawa, "Design and evaluation of hardware pseudorandom number generator mt19937," IEICE Trans. Inf. Syst.,vol. 88, no. 12, pp. 2876–2879, 2005.

[11] Pierre L'Ecuyer "Maximally Equidistributed combined tausworthe generators". Mathematics of Computation,65(213):203–213, 1996.

[12] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M.Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," National Institute of Standard and Technology Special Publication 800-22 Revision 1, August 2008.

[13] Marsaglia, G.,Diehard: A battery of tests for RNG, 1985, http://stat.fsu.edu/~geo/diehard.html.