Journal of Computational Physics 360 (2018) 93–103

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Random number generators for large-scale parallel Monte Carlo simulations on FPGA

Y. Lin, F. Wang *, B. Liu

Department of Physics, Xiamen University, Xiamen, Fujian 361005, China

ARTICLE INFO

Article history:
Received 22 July 2017
Received in revised form 22 December 2017
Accepted 17 January 2018
Available online xxxx

Keywords:
FPGA
Random number generator
Monte Carlo simulation
Ising model

ABSTRACT

Through parallelization, field programmable gate array (FPGA) can achieve unprecedented speeds in large-scale parallel Monte Carlo (LPMC) simulations. FPGA presents both new constraints and new opportunities for the implementations of random number generators (RNGs), which are key elements of any Monte Carlo (MC) simulation system. Using empirical and application based tests, this study evaluates all of the four RNGs used in previous FPGA based MC studies and newly proposed FPGA implementations for two well-known high-quality RNGs that are suitable for LPMC studies on FPGA. One of the newly proposed FPGA implementations: a parallel version of additive lagged Fibonacci generator (Parallel ALFG) is found to be the best among the evaluated RNGs in fulfilling the needs of LPMC simulations on FPGA.

1. Introduction

Monte Carlo (MC) simulations have been widely used in many areas of computational physics. Large-scale MC simulations require a high level of parallelization to make the simulation time manageable. MC simulations are communication intensive tasks, which are difficult to parallelize on central processing unit (CPU) based computing platforms due to the "Von Neumann bottleneck" [1]. Special computing platforms based on graphical processing unit (GPU) and field programmable gate array (FPGA) have been utilized for such simulations [2–5]. A high-end FPGA chip has millions of logical elements (LEs) and compatible amount of reconfigurable interconnecting resources that can be programmed to perform large-scale parallel Monte Carlo (LPMC) simulations efficiently [2,3,6–8]. The highest MC Ising spin update speed (1.6 ps per spin) is achieved on a FPGA system recently [8]. FPGA has also been applied in wide-ranging computational fields such as spin glass [7,9,10], computational chemistry [11], neural networks [12], artificial intelligence [13], nuclear medicine [14], and finance [15].

Random number generator (RNG) is an essential element of any MC simulation; the quality of the used RNG can influence the quality or even the validity of the simulation results [3,16,17]. A LPMC simulation system requires a large number of parallel and independent random number sequences, therefore requires RNGs with small memory and logical resource footprint; such requirements place stringent conditions on suitable RNGs. Many RNGs that are popular on CPU based platforms turn out not suitable for FPGA based systems. For example, Mersenne Twister [18] has a large number of state variables and hence use a large amount of logical resource. Another unsuitable popular RNG is KISS [19], which relies on addition and multiplication operations in serial steps thus requires a very large amount of logical circuitry to parallelize.

* Corresponding author.
E-mail address: fumingw@xum.edu.cn (F. Wang).

https://doi.org/10.1016/j.jcp.2018.01.029
0021-9991/

As a result, FPGA based LPMC studies usually have to use less-popular or roll-your-own RNGs which generally have a small number of state variables and rely on operations that are easily implementable and parallelizable.

Janus, a FPGA based LPMC system, uses a modified version of additive lagged Fibonacci generator (ALFG) proposed by Parisi and Rapuano [17] to simulate a variety of Ising and spin glass systems [7,9,10]; Gilman et al. use 24 copies of linear-feedback shift register (LFSR), which produce a 24-bit random number at each system clock, to simulate 3D Ising lattices [20,21]; Lin et al. used combined 43-bit LFSR and 37-bit cellular automata shift register (CASR) to simulate 2D Ising lattices up to 1024×1024 [2]; they also used combined 52-bit Leap LFSR and 37-bit CASR to simulate 2D Ising lattices up to 2048×2048 [3], where Leap LFSR is a hardware accelerated multi-iteration version of LFSR [22], equivalent to the multi-copy LFSR used by Gilman et al.; Francisco et al. use combined local 12-bit LFSR and global 32-bit LFSR to simulate 2D Ising lattices [8].

These less-popular or roll-your-own RNGs used in previous FPGA based studies need more careful tests to ensure their quality. Furthermore, LPMC studies tend to have a large number of independent RNGs and extremely long simulations, increasing the possibility of correlations not only within a bit stream from a single RNG instance but also among bit streams from multiple RNG instances. Previous studies have shown that problematic correlation effects can manifest themselves in LPMC studies [3]. It is therefore necessary to carry out more tests on these RNGs, especially application tests on large systems and for long periods of simulation times. In addition, it is desirable to tryout new RNGs on FPGA, especially well-known high-quality ones on CPU based systems. In this study, we run empirical and application tests on RNGS from previous FPGA based LPMC studies, including the Parisi–Rapuano RNG which is a modified version of ALFG [17], Leap LFSR which is similar to multi-copy LFSR used in [22], combined 52-bit Leap LFSR and 37-bit CASR [3], and combined 43-bit LFSR and 37-bit CASR [2]. For sake of comparison, linear congruential generator (LCG) [23], a known low-quality RNG, is also included in this study.

In addition, we implement and test two well-known high-quality RNGs on FPGA: xorshift128 and ALFG with extremely long lags for possible usage in FPGA based LPMC systems. Xorshift128 has been tested in many CPU based studies [24–26]; it has a small memory footprint of only 128 bits and therefore is well-suited for combinatory circuitry implementation on FPGA. ALFG has a relatively large memory footprint, and ALFG with short lags is known to fail certain empirical tests [17,27], however, ALFG with very long lags has a very good quality [27,28]. FPGA gives us capabilities to deal with both of these two issues simultaneously; instead of using multiple instances of ALFG with a short lag, we propose to implement a single instance of ALFG with an extremely long lag (4423) in combinatory circuits that can execute multiple iterations and hence provide multiple random numbers within a single system clock. As a result, the quality of the RNG is very high and the effective logical resource per random number is very small. We shall call this implementation scheme Parallel ALFG. For the sake of references, we refer to the above mentioned RNGs as RNG No. 1 to No. 7, respectively.

Testing suites DIEHARD and NIST are used for empirical tests while simulations of small and large-scale 2D Ising lattices at the critical temperature are used for application based ones. For application tests, the internal energies of 16×16 Ising lattices are simulated and compared with the theoretical value, and scaling properties of isothermal susceptibility of Ising lattices up to 2048×2048 are compared against finite size scaling (FSS) theory [3]. The FPGA system used in this study is an off-shelf FPGA development board DE4 from Terasic®. The FPGA chip on this board is an Altera Stratix IV GX FPGA (EP4SGX530C2), which has 531,200 LEs and 27,376 Kb memory. The FPGA based Ising lattice simulation system is developed using Verilog hardware description language [29] and runs with a system clock of 100 MHz. Details of the designs are reported elsewhere [2].

The paper is constructed as the following. In Section 2, we give detailed description of our FPGA implementation of the tested RNGs with focus on No. 2 (Leap LFSR), No. 6 (xorshift128) and No. 7 (Parallel ALFG), which all need specially designed combinatory circuitries to accelerate their iteration process. In Section 3, we present implementation details and testing results of NIST and DIEHARD empirical tests and that of the two types of 2D critical Ising lattice application tests. In Section 4 and Section 5, we give discussions and conclusions of this study.

2. Parallel random number generators on FPGA

LPMC studies require a large number of independent and extremely long random number sequences which can only be obtained by carefully designed parallel versions of RNGs. There are commonly two methods of constructing a parallel RNG from a serial RNG: the *leap-frog* method and the *independent-sequence* method [23]. The *leap-frog* method divides a single sequence from a serial RNG in a round-robin way among all the parallel processing units. One way of implementing a *leap-frog* parallel RNG for a system with N parallel processing units is to create a serial RNG instance for each of the parallel processing units, and to iterate each RNG instance N times instead of the usual one time before delivering a random number. With an identical seed and suitable initial iteration status, the parallel RNG instances would have divided the random sequence among themselves, with the ith processing unit generating random numbers $x_i, x_{i+N}, x_{i+2N}, \cdots$ from the whole sequence. It is possible for only a few serial RNGs, such as LCG and LFSR, to realize such leap forward in a single step instead of actually iterating N times on a CPU based system [30]. On the other hand, a large class of serial RNGs can be hardware accelerated on FPGA to achieve this leap forward effect [17,22], and thus provide more opportunities of parallelizing RNGs with the *leap-frog* method. Furthermore, an RNG instance on FPGA can execute multiple leap forward actions simultaneously, acting like multiple RNG instances that can provide multiple random numbers simultaneously. RNG No. 7 and, to a lesser extent, RNG No. 1 are such examples and their implementations will be discussed in Section 2.7.