Coprocessing Architecture in System-on-programmable-chip for Monte Carlo Simulation

Stephen J.J. Ong, Z. Abdul Halim
E-mails: steojj@live.com, zaini@cedec.usm.my

Abstract— Monte Carlo simulation is mainly done in computer clusters, supercomputers or computers coupled with hardware accelerators. These processing methods are sufficiently fast; however, this speed comes at the expense of physical area and power consumption. The current study aims to use System-On-Programmable-Chip (SOPC) to accomplish the same task but using a smaller physical area. The SOPC approach takes on a coprocessing architecture to execute a Monte Carlo algorithm-based random walk on boundary (WOB) method to calculate unit cube capacitance. A time extrapolation is performed on the processing rate of the coprocessor to reveal a performance that is comparable to a 1.3 GHz Pentium 4 desktop computer. The performance results can be attributed to a streaming interface which connects the random number generator source and the coprocessor.

Keywords—coprocessing; SOPC; Monte Carlo; random walk on boundary;

I. INTRODUCTION

Scientific computing or computational science is characterized by its ever increasing demand on data processing power. It is a field of study which concerns about constructing mathematical models of real world systems and proceed to conduct quantitative analysis on the models. Given a set of defined input parameters, these models require massive amount of calculations and iterations to yield an output. Some of the more common algorithm or methods to evaluate the mathematical models are Monte Carlo (MC), Runge Kutta, Taylor series and Newton’s method.

As an example, MC methods [1] are a class of computational algorithms that rely on repeated random sampling to compute their results. In essence, this method works on a set of deterministic data to compute an estimated value of an undeterministic property. MC takes on a great amount of iterations in the order of $10^{12}$ before the estimate converges to produce a result. The massive amount of calculation and iteration results in the reliance of MC algorithm on High Performance Computing (HPC) platform.

Current HPC systems incorporate hardware accelerators in their processing nodes to speed up critical portions of the coding. Field Programmable Gate Array (FPGA) is one of the choices as hardware accelerators. FPGA-based solutions, are presently offered by vendors such as Cray (XD1 and XT5h/XR1), SRC Computers, SGI (Reconfigurable Application Specific Computing – RASC) and DRC Computers (Reconfigurable Processor Unit – RPU), which couple FPGAs with high-end conventional microprocessors [2]. These implementations are sufficiently fast, which reaches Teraflop ($10^{12}$ floating point operations per second) region, but usually require a considerable amount of physical space.

The current study aims to introduce SOPC in high-performance application. The SOPC approach takes on a coprocessing architecture to execute an MC algorithm-based random walk on boundary (WOB) method to calculate the unit cube capacitance in floating point arithmetic. The algorithm is usually ported to computer clusters, supercomputers or host computers coupled with hardware accelerators. However, in the current study, the SOPC approach is used to accomplish the same task but using a smaller physical area. There has not been much attention on SOPC as an alternative computing platform, despite an emerging interest among embedded system developers in reconfigurable computing [3]. The closest reported work should be [4], in which a novel architecture of a FPGA-based hardware accelerator is described for MC simulation to calculate interest rates.

The rest of the paper is organized as follows. Section 2 covers the background details and previous work. Section 3 briefly reviews the WOB method. Section 4 describes the coprocessing architecture. Section 5 presents results and discussion. Finally, section 6 offers the conclusion.

II. BACKGROUND

FPGA capacities are always increasing. It makes FPGA platform complex and large enough to contain the entire SOPC. This setup was identified in [3], wherein the processor is embedded within the reprogrammable fabric itself. The processor can either have a “hard” core or “soft” core. As of recently, vendors have begun to offer embedded soft-core processors that are synthesized using existing FPGA logic elements with abundant remaining logic. With the availability of such processors, it is possible to build a customized and complete processor system, equipped with communication buses, peripherals and I/O drivers onto a single FPGA chip. This results in a reduction of physical size. In this setup, the reconfigurable FPGA logic is used to implement one or more
processes that handle heavy computation while the embedded processor handles non-performance-critical tasks [5].

A coprocessor is a processing element that is used alongside a primary processing unit (usually a processor) to offload certain computations normally performed by the primary processing unit. Code acceleration is achieved by both reducing multiple instruction codes into a single instruction as hardware implementation [6]. Coprocessor is described in Lau et al. [7] and Garcia et al. [8] as one of the three ways of implementing FPGA as hardware accelerators in addition to a soft-core processor.

Kalomiros and Lygouras [5] evaluated the performance of a coprocessing system in image processing. The system consisted of a host computer and a FPGA running on Nios II embedded soft processor. In the host/co-processor architecture, FPGA executed data path which is computationally intensive, while Nios II handled the control path. The paper went on to show that the performance of the host PC without the hardware co-processor was good in the case of simple processing but fell rapidly with increasing computational complexity. The performance of hardware/software mixed architecture is nearly constant as complexity increases.

The following reviews of previous work investigated the implementation of MC simulation in FPGA-based systems. As described previously, [4] presented a novel architecture of an FPGA-based hardware accelerator for MC simulation in calculating interest rates. The large scale MC simulation has been accelerated in floating point arithmetic. The architecture involved mainly a microprocessor, random number generators, an external memory and a coprocessor. It was found that the system, running at 50 MHz, was 25 times faster than the algorithm performed on a 1.5 GHz Intel Pentium machine.

In a journal by Akila Gothandaraman et al. [2], a considerable speed-up has been achieved by using FPGA to accelerate two computational intensive kernels of Quantum MC application to calculate potential energy and wave function. Two architectures were tested. Firstly, a dual-processor 2.4 GHz Intel Xeon which were augmented with two development boards consisted of Xilinx Virtex-II Pro FPGAs achieved 3 times speedup over software-only implementation. Next, the same algorithm was ported to Cray XD1 supercomputer equipped with Xilinx Virtex-II Pro and Virtex-4 FPGAs. The hardware-accelerated application on one node of the high performance system equipped with a single Virtex-4 FPGA yielded a speedup of approximately 25 times over the serial reference code running on one node of the dual-processor dual-core 2.2 GHz AMD Opteron.

In a work reported by another paper, Xilinx Virtex 4 FPGA has been used to form the basis of massively parallel processing system (IANUS) [9]. It was a large scale on-chip parallelism configured for Monte Carlo simulation of spin glass system. It was shown that an IANUS processing board of 16 FPGAs running at 62.5 MHz was equivalent to the performance of hundreds of 1.6 GHz PC.

In another paper, fault tree analysis which was done using Monte Carlo methods has been accelerated using FPGA. A tree model, called Time-to-Failure tree, was introduced to show the relationship between the time to failure of the system and the time to failures of its components. Each time-to-failure tree was implemented as a pipelined digital circuit, which can be synthesized to FPGA. Experimental results for some benchmark fault trees showed that this method can be about 471 times faster than software based MC simulation [10].

Maxwell, a FPGA-based supercomputer, was used to implement MC simulation of GARCH option pricing model. The system consisted of a 32 CPU cluster augmented with 64 Virtex-4 Xilinx FPGAs. Hardware implementation showed that the FPGA-based implementation of the GARCH model outperformed an equivalent software implementation running on a workstation cluster with the same number of computing nodes (CPU/FPGA) by a factor of 750[11].

As discussed in the preceding paragraphs, FPGA-based implementation yields performance improvement over software-only implementation in CPU. However, these FPGA accelerators are mostly coupled to a host processor. The current study seeks to describe a SOPC coprocessing architecture for high-performance application. MC-based WOB method is used as an example. Following this approach, we present results that even a single-chip implementation is a potential alternative platform for high performance application.

III. WOB METHOD

WOB method is based on MC algorithm. It is a rapid solution to the integral equation which computes the charge distribution on a conductor’s surface [12]. Capacitance of the unit cube is calculated from the charge equation solution. It was discovered that WOB algorithm is the most efficient MC method for unit cube capacitance calculation. The said method has the smallest variance, the smallest computational complexity, and has no bias in the estimate [12]. The unit cube capacitance formula is represented as below:

$$C = \lim_{N \to \infty} \frac{1}{N} \sum_{x=1}^{N} v(y_x)^{-1}$$

(1)

In which, $v(y_x)$ is the inverse of the distance between static point $x$ and travelling point $y$. $N$ represents the number of input samples and $C$ is unit cube capacitance.

WOB method is derived from Brownian motion. In Brownian motion, a particle at time $t$ and position $p$ will make a random displacement $r$ from its previous point with regard to time and position. The resulting distribution of $r$ is expected to be Gaussian (a Normal distribution with a mean $\mu$ and a standard deviation $\sigma$).

Collaborative Microelectronic Design Excellence Centre
soft core microprocessor provided by Altera. The architecture combines a fast RNG and a fast simulation core with an on-chip microprocessor for control-oriented operations. The entire system operates at a nominal clock frequency of 50MHz. Fig. 1 and Cellula Automata Shift Register (CASR) [13] is used as the uniform number generator. It generates a uniformly distributed IEEE 754 single precision floating point number in the range of [0, 1]. However, Brownian motion, which forms the theoretical base of WOB method, is simulated using Gaussian (Normal) distribution [4]. So, the uniform random variable is transformed to normal distribution using pipelined Box Muller algorithm. The final output is two parallel independent streams of IEEE 754 single precision floating point random numbers at every clock cycle.

B. Monte Carlo Simulation Core

The customized simulation core is the implementation of (1) in hardware. Its computing nodes are made up of pipelined floating point arithmetic modules which are optimized for Altera devices. The pipelined implementation of floating point

modules also mean that they are capable of accepting one new operation every clock cycle, thus providing enormous computational density within the pipeline. The core is coupled to the system as a coprocessor.

C. Programmable Interconnect

The system is built on the manufacturer’s programmable switch interconnect fabric (Avalon) which bypasses bus contention and gives a higher-performance pipe between processors and peripherals. The Avalon is a nonblocking interface, created by the SOPC Builder tool that interconnects all the components in the system and permits multiple simultaneous master-slave transactions. It replaces the traditional shared bus of usual electronic systems.

Two types of interfaces co-exist within a single architecture. The shaded area in Fig. 1 shows the Avalon Streaming (ST) interface operates alongside the more common Avalon Memory-Mapped (MM) interface. Avalon ST interface is defined as an open channel point-to-point communication between a source-sink pair which allows successive data transfers using flow control signals, similar to handshaking protocol. The natural flow control of Avalon ST paces the flow of data among RNG, FIFO and hardware accelerator, without the need for Avalon master’s (Nios II microprocessor) involvement. Due to this, the most basic Nios II core is utilized to conserve logic usage. Fig. 2 illustrates a basic Avalon ST protocol between a data source and a data sink, where the backpressure, involving merely four signals. The readLatency parameter indicates the number of clock cycles from the moment the ready signal is asserted until the valid data are driven.

D. Nios II Microprocessor

The Nios II microprocessor from Altera presents advantages in areas such as low power consumption and small required FPGA logic area, along with the ease to configure its functionality according to various applications. These flexibility and configurability capacity allows not only to use a great variety of included peripherals, but also to create custom peripherals that can then be easily interfaced to the processor. The custom peripherals can be accessed through a relatively transparent interface written in C language, or even in assembly.
In this architecture, Nios II microprocessor is completely relieved from data transfer to execute control functions. The C program executed on Nios II initializes the parameters, configures the simulation, and communicates with the hardware. In this work context, Nios II initializes and sets up the operations of Direct Memory Access (DMA), RNG and hardware accelerator. The operation of RNG and hardware accelerator is governed through the respective control registers. Then, Nios II continuously polls the status register of RNG to detect a bit change that signifies that the number of test samples has reached the intended quantity and thus, stops the simulation.

E. Nios II Memory Hierarchy

The Nios II memory hierarchy starts with a local register of the CPU at the highest level, followed by a slower but still fast on-chip random-access memory (RAM) and lastly, a slower external memory. The type of Nios II core used in this work is most basic and does not have cache memory. Avalon ST is the medium of transfer for data stream between RNG, FIFO and coprocessor. The streaming connection between RNG and coprocessor as seen in Nios II memory hierarchy is depicted in Fig. 3.

Based on Fig. 3, the coprocessor lies between external memory and on-chip RAM. The conventional method of data transfer from RNG to coprocessor involves the processor to read data from RNG and perform write operation to coprocessor. In terms of hierarchical view, data transaction traverses the hierarchy twice (read and write). Since the operation of Avalon ST interface does not involve Nios II processor, memory access latency which is usually associated with lower memory hierarchy is eliminated. Moreover, the inclusion of point-to-point streaming interface removes the need for the addressing mode. The absence of access latency and addressing mode enables data transfer to occur at every clock cycle.

V. RESULTS

The system hardware is integrated in SOPC Builder, a tool by Altera to facilitate the integration of modules in an SOPC system. Nios II application code is written and compiled in Nios II Software Build Tools. Then the hardware-software environment is then co-simulated in Mentor Graphics Modelsim-Altera by performing a 30ms functional simulation. The number of processed input samples, \( N \) is noted at various intervals throughout the simulation. The value of \( N \) reflects the operating state of the core because these input samples are consumed by MC simulation core. The value of \( N \) is plotted against simulation time to produce a graph as seen in Fig. 4.

The plotted data are approximated as a linear function with an R-square value of 1. The linear function has a gradient of approximately 50MHz system clock frequency. This is an expected trend because the system clock frequency remains unchanged throughout the simulation. Therefore, the system is predictably linear at any instance, except for the simulation core stalls. However, this scenario can be avoided by using a good RNG. Thus, the uncertainty of trend extrapolation is greatly reduced. Using the linear equation in Fig. 4, the simulation time is extrapolated to 30s. The system is able to process \( 1.5 \times 10^6 \) input samples (\( N = 1.5 \times 10^6 \)). A logic synthesis in Altera Quartus II reveals that the maximum tolerable clock frequency of our system is 153MHz, with a logic utilization of 60.7% using Stratix II EP2S30F484C3 device. If the linear function gradient is replaced by 153MHz instead of 50MHz, the system is capable to process \( 4.6 \times 10^7 \) input samples (\( N = 4.6 \times 10^7 \)) within the same 30s simulation time. This theoretical result is comparable to the performance obtained by a 1.3 GHz P4 desktop computer in a previous study [12].

![Number of Samples versus Simulation Time](image)

Figure 4. Number of input samples versus simulation time

![Memory hierarchy view of coprocessing architecture](image)
The impressive performance of coprocessing can be attributed to the streaming connection between RNG, FIFO and MC simulation core. The natural flow control of Avalon ST paces the flow of data without the need for Nios II’s involvement, thus allowing distinct separation of software and hardware function. Software solely executes control function while hardware is responsible for data processing. The uninterrupted hardware execution enables continuous data stream processing at every clock cycle.

VI. CONCLUSION

MC algorithm has been successfully implemented using the SOPC coprocessing approach. Moreover, SOPC does not suffer from physical area size issue as the entire system is contained within a single FPGA chip. On the other hand, a graph extrapolation shows that the proposed coprocessing architecture is able to complete a theoretical computation of \(4.6 \times 10^{12}\) input samples in 30 seconds, which is comparable to a 1.3 GHz Pentium 4 desktop computer. This theoretical result is achieved at the maximum clock frequency of 153MHz of our particular architecture design. The impressive performance of coprocessing can be attributed to the streaming connection among RNG, FIFO and MC simulation core.

ACKNOWLEDGMENT

This work was supported by a grant (Grant no. 1001/PCEDEC/814058) from Collaborative Microelectronic Design Excellence Centre. Writers would also like to thank Universiti Sains Malaysia for allowing access to their facilities.

REFERENCES