PRACTICAL IMPLEMENTATION OF A NETWORK-BASED STOCHASTIC BIOCHEMICAL SIMULATION SYSTEM ON AN FPGA

Masato Yoshimi, Yuri Nishikawa, Yasunori Osana, Akira Funahashi
Keio University, Yokohama, Japan
email: bio@am.ics.keio.ac.jp

Noriko Hiroi
EMBL-EBI,
Wellcome Trust Genome Campus, UK

Yuichiro Shibata, Hideki Yamada
Nagasaki University
Nagasaki, Japan

Hiroaki Kitano
Kitano Symbiotic Systems Project,
ERATO-SORST, JST, Tokyo, Japan

Hideharu Amano
Keio University
Yokohama, Japan

ABSTRACT

Stochastic simulation of biochemical reaction networks are widely focused by life scientists to represent stochastic behaviors in cellular processes. Stochastic algorithm has loop- and thread-level parallelism, and it is suitable for running on application specific hardware to achieve high performance with low cost. We have implemented and evaluated the FPGA-based stochastic simulator according to theoretical research of the algorithm. This paper introduces an improved architecture for accelerating a stochastic simulation algorithm called the Next Reaction Method. This new architecture has scalability to various size of FPGA. As the result with a middle-range FPGA, 5.38 times higher throughput was obtained compared to software running on a Core 2 Quad Q6600 2.40GHz.

1. INTRODUCTION

Stochastic simulations of biochemical models now play important roles in systems biology. However, it is widely known that their numerical method, formally called a Stochastic Biochemical Simulation Algorithm (SSA), is a quite CPU-hogging application. As the size and complexity of target simulation models increase every year, improvement of algorithms and computer systems are hot issues.

The First Reaction Method (FRM) and the Direct Method (DM) proposed by Gillespie are well-known SSAs[1]. Their algorithms are quite simple and have various levels of parallelism since they are based on Monte-Carlo methods. Their hardware accelerations are hopeful solutions, and FPGAs are especially being focused for this aim by several research groups, because they can accommodate dedicated modules that are well-suited to target biochemical models[2][3]. However, as studies of stochastic algorithms advanced, now there are several computationally-efficient SSAs with more complex data flow. Our group works on an FPGA-based stochastic biochemical simulator[4][5] that applies the Next Reaction Method (NRM) [6], one of the improved SSAs. We especially aim to generate an accurate system without applying approximated algorithms, and achieve high throughput at the same time by running multiple simulation threads.

This paper studies an implementation of an FPGA-based data-driven biochemical simulator that applies NRM. By introducing a hierarchical network structure, the design allows a flexible adjustment of balance between resource utilization and performance to fit to the amount of hardware resource of the target FPGA device.

2. SSA : STOCHASTIC BIOCHEMICAL SIMULATION ALGORITHM

2.1. NRM : Next Reaction Method

Stochastic biochemical simulation is performed by obtaining time evolution of a stochastic "state" for a spatially homogeneous biochemical models. A biochemical model is defined as a list of reactions and molecular species.

Simulation proceeds by a repetition of "reaction cycles", each of which includes the following two steps: 1. selection of reactions, and 2. updating state of the model. In step 1, the current state of the model determines a single reaction and its expected time of occurrence. Then in step 2, the state is updated based on the selected reaction.

Gibson and Bruck proposed NRM in 2000[6]. Computational order of NRM is $O(\log(M))$, where M represents number of reactions in a model. NRM has more scalability for large model than FRM and DM whose computational orders are $O(M)$. NRM is often adopted as a high-speed

*The project described was supported by Grant Number R01EB007511 from the National Institute Of Biomedical Imaging And Bioengineering. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute Of Biomedical Imaging And Bioengineering or the National Institutes of Health.

978-1-4244-1961-6/08/$25.00 ©2008 IEEE.

stochastic algorithm in recent biochemical simulators such as COPASI[7].

NRM introduced two data structures. First, the algorithm employs a binary tree called an Indexed Priority Queue (IPQ) to store "putative time" τ_j, which indicates a time that reaction $R_j(j$: Reaction index : $1, 2, \cdots, M)$ occurs. Equation (1) is evaluated in only selected reaction at every reaction cycle, and previous results of this equation are obtained by referring to the IPQ.

$$\tau_j = \ln(1/r)/a_j + t \tag{1}$$

Here, r is a uniform random number in the range of $(0, 1)$ and t is the system time. a_j is called a "propensity", which is the occurrence frequency of R_j. Propensity is calculated from the number of reactants species and reaction constants defined in the list of reactions.

The second data structure is a "dependency graph" (DG) to represent dependencies among the reactions. The graph is a list of reactions whose putative time $\tau_{j,\text{new}}$ needs to be recalculated by occurrence of a certain reaction. $\tau_{j,\text{new}}$ is obtained by Equation (2).

$$\tau_{j,\text{new}} = a_{j,\text{old}}(\tau_{j,\text{old}} - t)/a_{j,\text{new}} + t \tag{2}$$

2.2. Studies to improve performance of SSA

Several challenges have been made to alleviate computational loads by refining numerical algorithms or by designing high-speed simulator. In algorithm studies, there are three major approaches to enhance computational efficiency: improvement of calculation order, optimization of models[8], and approximation with some loss of precision within allowable range[9]. Normally, simulations are executed on general purpose PCs, but now there are several implementations that focus on FPGAs and GPUs as simulation platforms[2][10]. Most of the approaches use FPGAs. They are based on approximated algorithms which are executed on dedicated hardware, and they are compiled whenever input models change. On the other hand, our implementation uses embedded memory. This gives two advantages; the amount of logic resource is unaffected by (a) different orders of reaction (ex. primary reaction, secondary reaction, etc.), and (b) scale of biochemical models.

3. NRM IMPLEMENTATION ON AN FPGA

3.1. Design concept and previous implementation

Due to the characteristic of NRM, number of arithmetic operation changes at every reaction cycle. Thus, deterministic or static data flow scheduling is inefficient to accommodate to dynamical changes of calculation type, number of its occurrence, and timing of requests. To address this problem, we proposed an architecture that multiple threaded modules (data sets of a simulation thread) access common shared modules (calculation modules). Order of data transfer is dynamically controlled. This architecture is scalable, and can be implemented on FPGAs of various capacity.

In previous implementations, we evaluated systems with following network structure : use of simple multiplexer which connects threaded and shared modules[4], and use of on-chip routers to configure an NoC-based system[5]. However, the multiplexer-based system could not maintain operating frequency high when the number of threaded module increased. In the NoC system, on the other hand, a large proportion of logic resources were required for the interconnection, but a router had unused data-paths and useless Block-RAMs. Thus NoC-based approach was not very efficient in term of area and performance ratio.

3.2. NRM implementation with hierarchical connection

Considering our past experiences as state above, this work also follows data-driven NRM implementation which can make use of increasing logic capacity of FPGAs. Like previous implementations, threaded modules are connected to shared modules with a data transfer network.

3.2.1. Data transfer network

Data packets with minimum flit size of 32-bit are transferred between threaded and shared modules. The first header flit contains routing information. The network is a hierarchical structure using specialized routers called "concentrators" and "distributors" as shown in Fig. 1. Compared to the simple multiplexer[4], the concentrator has FIFOs to prevent latency for packet transfer. This gives large flexibility to interconnection network when transferring data packets with variable length via common data path.

The concentrator checks requests from each input port, and issues transfer requests to the arbiter. Each input port has a small FIFO which stores a flit when the packet needs to wait for the arbitration and transfer. Connection is released after transmitting a tailer flit, and then the concentrator checks the next packet to transfer. The distributor refers routing information in the header flit and connects its input port to output port. This structure reduces logic resources drastically compared to the on-chip router[5], since FIFO uses 32-bits$\times 4$ registers instead of a 32-bits$\times 512$ Block-RAM.

3.2.2. Threaded module

Fig. 2 shows the block diagram of the threaded module. The threaded module has several data tables and a packet controller. The controller manages the sending and receiving of data packets[5]. The state of each thread is held in three tables; the number of species (state), IPQ and propensity.

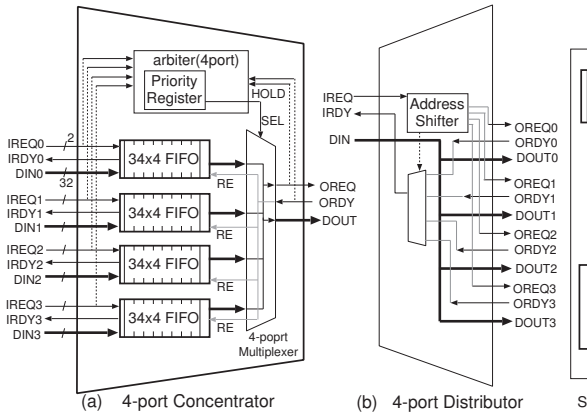

Fig. 1. A block diagram of interconnecting modules

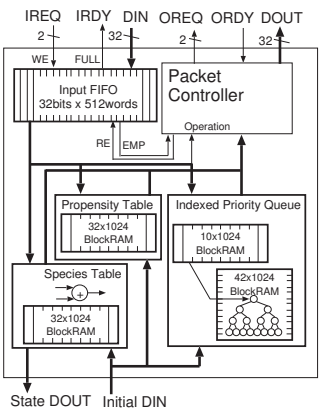

Fig. 2. A threaded module

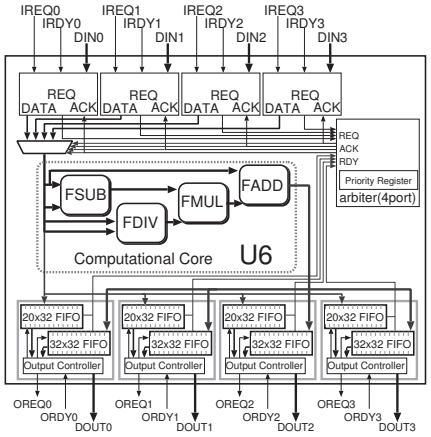

Fig. 3. A shared module (4 I/O ports)

These tables are stored in BlockRAMs to accommodate to large-scale biochemical models with relatively small logic resource consumption. The packet controller generates and transmits request packets to shared modules by NRM algorithm. It also reads incoming packets from the FIFO and then performs the specified process.

3.2.3. Shared Module

In our previous implementation[5], the shared module was pipelined, but only provided a set of I/O port. So, the pipeline bubbles were generated corresponding to the number of flits of a packet. To relax the congestion at the I/O port and reduce the bubbles in the pipeline, multiple I/O ports were provided to shared modules.

Fig. 3 shows the structure of the shared module. This is a 4-port shared module which calculates Equation (2). When a packet arrives, its routing address is stored into an address FIFO at the output port, and the request to the computational core is asserted. The arbiter selects certain request with the similar method as the concentrator, and the input data is sent to the computational core. The result of computation is stored into the FIFO at the output port, and as soon as the computation is finished, the data packet at the output port is sent back to the threaded module.

3.3. NRM implementation

We implemented two types of simulator with four threaded modules and 24 threaded modules. Fig. 4 shows an outline of connection between threaded modules and shared modules. In 4-threaded version, all shared module has four I/O ports which connect each threaded module with 7-port distributor and 6-port concentrator as shown in Fig. 4(a). In this case, the congestion is seen only when each packet tries to acquire the right to access computational cores inside the shared modules. On the other hand, in Fig. 4(b),

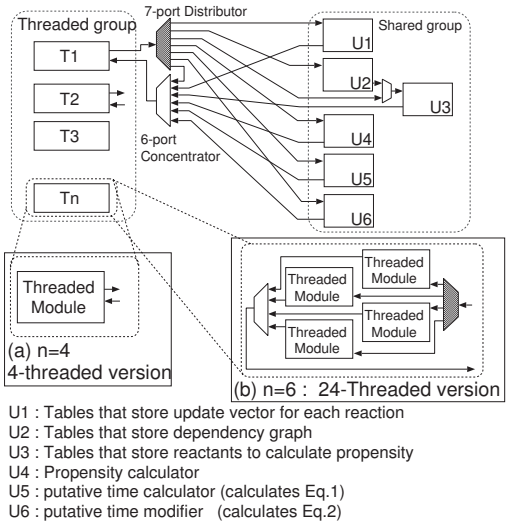

Fig. 4. Examples of data-path between threaded group and shared group : (a) NRM/FPGA/4T (b) NRM/FPGA/24T

24-threaded version has six threaded subgroups which integrates four threaded modules in the same way as 4-threaded version. This hierarchical structure can increase the number of threaded modules with a small logic resource overhead, while there is a waiting time for conflict resolution in the network unlike 4-threaded version.

4. EVALUATION

4.1. Resource utilization

The evaluation environments are TB-5V-LX50T / LX110T-PCIEXP, FPGA evaluation boards by Tokyo Electron Device [11]. All modules are written in Verilog-HDL, and contain several IP cores generated by Xilinx CORE Genera-

tor. Table 1 shows synthesis results for two configurations of NRM implementation: 4-threaded version and 24-threaded version. The network's resource consumption only doubled while the number of threaded modules increased by 6 times. Both could fit in the target FPGA device, but operational frequency of 24-configuration was degraded because of large use of hardware resource. Long wiring delay is probably due to lack of registers in the distributors. Overall, operational frequency depicted a slower declining curve as the number of threaded modules increased compared to previous implementations[4]. However, each computational core in the shared module can operate at around 170MHz. Thus, the system may run at a higher operational frequency by improving the wiring delay of the network, by such as adding a FIFO in the input or output of the distributor.

4.2. Performance evaluation

Performance evaluation in this paper used an actual biochemical model called a Heat-Shock Response(HSR), which is a model that represents expression of *E.coli* when they are exposed to elevated temperatures. HSR is a famous and relatively large-scale model that involves 28 species in 61 reactions. Cao adopts this model to analyze computational efficiency of various SSAs, and StochKit[10], a stochastic simulation framework, also provides HSR as a sample model.

Table 2 is a throughput when HSR was run on StochKit with Direct Method, hand-coded software of NRM written in C++, and our current FPGA implementation. In FPGA implementation, there was a stall at the I/O for subgroups of threaded modules, so 24-thread version required 15% more clock cycles to simulate one reaction cycle. We have found in our previous evaluation that FPGA's performance gain versus microprocessors becomes large according to the the size of biochemical models, because the model size mainly effects the latency for updating the binary tree, and this process is much more advantageous to execute on FPGA than on microprocessors[5].

To achieve higher performance, we need to study the congested portions in the network. The congestion is perhaps seen at a shared module that generates more output flits than input flits (ex. U2 in Fig. 4). The traffic may be sorted out by placing several frequently-accessed modules.

Table 1. Resource utilization and operating frequency

Execution System	NRM/FPGA/24T		NRM/FPGA/4T	
Device	5VLX110T FF1136-1		5VLX50T FF1136-1	
Slice Registers	45040	(65.16%)	22005	(76.41%)
LUTs	64631	(93.51%)	25894	(89.91%)
BlockRAM/FIFO	133	(89.86%)	33	(55.00%)
DSP48Es	24	(37.5%)	24	(50.00%)
Op. Freq.	100.56 [MHz]		141.06 [MHz]	

* Synthesis and placement and routing were done by ISE8.2i.

5. CONCLUSION AND FUTURE WORK

This paper studied the FPGA implementation and evaluation of the stochastic biochemical simulator which is based on an algorithm called NRM. Each module is categorized into two groups called threaded modules and shared modules. By modifying communication network between two groups, a hardware design can be flexibly tuned to perform well on the target FPGA device.

The evaluation result indicated that parallel execution of NRM marked a throughput enhancement of about 5.38 times. As the future work, we are planning to verify the efficiency of current design with various models using real hardware.

6. REFERENCES

[1] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *J. Comput. Phys.*, vol. 22, pp. 403–434, 1976.

[2] J. F. Keane *et al.*, "A compiled accelerator for biological cell signaling simulations," in *proc. of the 12th Intl. Symp. on FPGA*, Feb. 2004, pp. 233–241.

[3] B. P. Thurmon *et al.*, "Accelerating exact stochastic simulation using reconfigurable computing," in *The 2005 International Conference on Engineering of Reconfigurable Systems and Algorithms*, 2005.

[4] M. Yoshimi *et al.*, "FPGA Implementation of a data-driven Stochastic Biochemical Simulator with the Next Reaction Method," in *proc. of the 17th Intl. Conf. on FPL.* IEEE, Aug. 2007, pp. 254–259.

[5] M. Yoshimi *et al.*, "A framework for implementing a network-based stochastic biochemical simulator on an fpga," in *proc. of the Intl. Conf. on ICFPT'07*, Dec. 2007, pp. 193–200.

[6] M. A.Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *J. Phys. Chem. A*, vol. 104, no. 9, pp. 1876–1889, 2000.

[7] S. Hoops *et al.*, "Copasi — a complex pathway simulator," *Bioinformatics*, vol. 22, no. 24, pp. 3067–3074, Dec. 2006.

[8] Y. Cao *et al.*, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *J. Chem. Phys.*, vol. 121, no. 9, pp. 4059–4067, 2004.

[9] D. T. Gillespie, "Stochastic simulation of chemical kinetics," *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, May. 2007.

[10] H. Li, Y. Cao, L. R. Petzold, and D. T. Gillespie, "Algorithms and software for stochastic simulation of biochemical reacting systems," *Biotechnology Progress*, vol. 24, no. 1, pp. 56–61, Sep. 2007.

[11] Tokyo Electron Device, "Virtex-5 LXT/SXT PCI Express Evaluation Platform Board," http://www.inrevium.jp/eng/x-fpga-board/hibiki.html.

Table 2. Performance evaluation for HSR model

	StochKit$^{\star 1}$	NRM		
		SW$^{\star 1}$	FPGA/4T	FPGA/24T
clocks/reaction	N/A	N/A	233.93	268.81
Throughput [Mcycles/sec]	1.61	1.67	2.10$^{\star 2}$	8.98$^{\star 2}$
Gain	0.97	1.00	1.26	5.38

$^{\star 1}$ Execution environment of software program is Core 2 Quad Q6600 2.40GHz, 4GB RAM, Linux 2.6.22-14, gcc-4.1.3 (-O3).
$^{\star 2}$ Performances of FPGA is obtained by multiplying the number of threads and average clock cycles per a reaction cycle.