EFFICIENT SCHEDULING OF RATE LAW FUNCTIONS FOR ODE-BASED MULTIMODEL BIOCHEMICAL SIMULATION ON AN FPGA

Naoki Iwanaga*, Yuichiro Shibata*, Masato Yoshimi†, Yasunori Osana†, Yow Iwaoka†, Tomonori Fukushima†, Hideharu Amano†, Akira Funahashi††, Noriko Hiroi††, Hiroaki Kitano††, Kiyoshi Oguri*

*Nagasaki Univ   †Keio Univ   ††Kitano Symbiotic Systems Project ERATO-SORST

Introduction

A demand for high speed simulation of cellular systems has been rapidly growing in the field of bioinformatics.Whole-cell simulation is still a challenge for both biologists and computer scientists.

We are developing a reconfigurable ordinary differential equation(ODE)-based biochemical simulation system called ReCSiP.The structure of ReCSiP can be tailored to each simulation target by using FPGAs, and high speed simulation is carried out without losing the flexibility.Moreover, in order to ensure wide versatility for simulation targets, ReCSiP adopts Systems Biology Markup Language(SBML) as a front-end.

Using ReCSiP front-end software, simulation models written in SBML are translated into hardware ODE solvers and are directly executed on an FPGA. The heart of ReCSiP is ODE solvers which calculate concentrations of substances in the simulation target for each time step integrating rate law functions. The source of ReCSiP's performance advantage is deep pipeline structure enabled by perfect static scheduling. SBML predefines 33 frequently used rate law functions, therefore efficient arithmetic scheduling of these functions is crucial.

Scheduling of Solver Cores

A solver core, which is the reaction-specific circuits to calculate the change in concentrations for each time step, consist of some IEEE-754 compliant single-precision floating point arithmetic modules and shift registers. In order to achieve high throughput operation, fully pipelined arithmetic modules are designed. Here, some scheduling alternatives are possible, and there are tradeoffs between performance and required resources.

The following critical path first scheduling policy intends to minimize the execution latency.

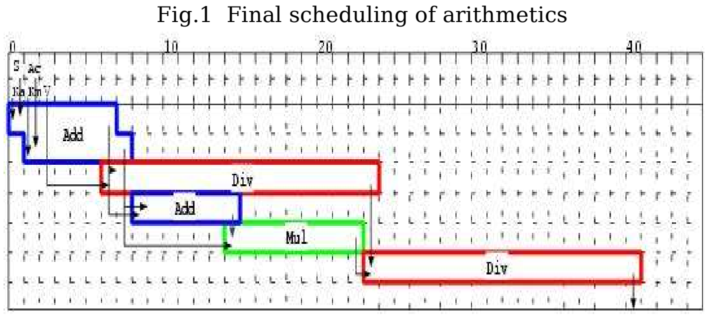
1)The pipeline pitch P of the given rate law function is determined. In ReCSiP, a solver core has two input ports; one for reading concentration data and the other for reading coefficients. Therefore, P is determined as

$$P = \max(N_x, N_k)$$

2)The arithmetic operations in the critical path of the corresponding data flow graph(DFG) are scheduled in an as soon as possible(ASAP) manner.

3)Then, the other arithmetic operations are scheduled so as not to affect the critical path operations as shown in Fig.1.

Fig.1 Final scheduling of arithmetics



On the other hand, by inserting some shift registers between arithmetic units in the critical path, starting time of each arithmetic unit might be adjusted so that more arithmetic units are shared. While this sharing first scheduling policy would alleviate the hardware cost, the execution latency of the solver core would be degrated.

Fig.2 Required FPGA resources comparison

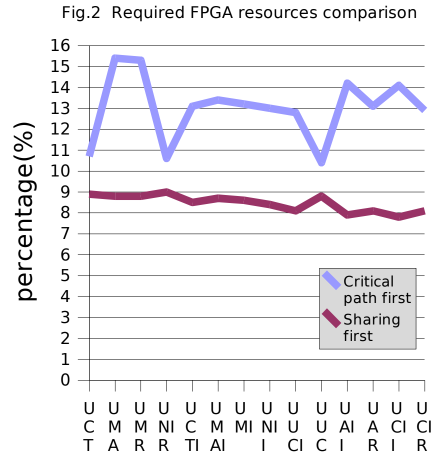
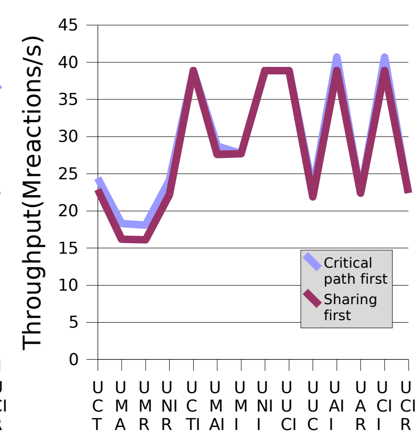


Fig.3 Performance comparison



Evaluation

The target device is Xilinx XC2VP70. As shown in Fig.2 and 3, a solver core occupies 10 to 15% of FPGA slices with the critical path first scheduling, while 7.8 to 9.0% of slices are used by stretching out the critical path operation. In terms of the required number of slices, 33.8% of hardware reduction ratio is achieved in average with the sharing first scheduling.

Solver cores calculate the velocity for 15 to 40 million reactions a second. Applying the sharing first scheduling, the execution latency of the solvers is increased by up to 4 clock cycles. Moreover, the maximum frequency is decreased by about 5 Mhz in average, since high degree of arithmetic sharing results in generation of complicated multiplexers.

Fig.4 Required FPGA resources comparison

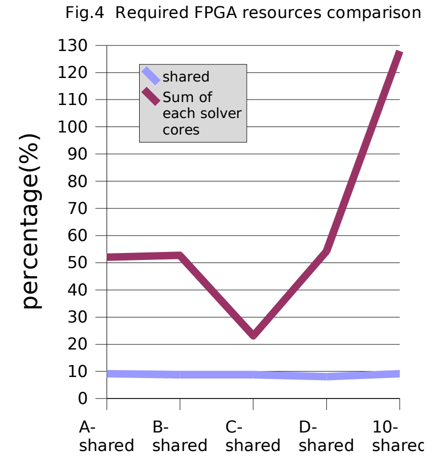
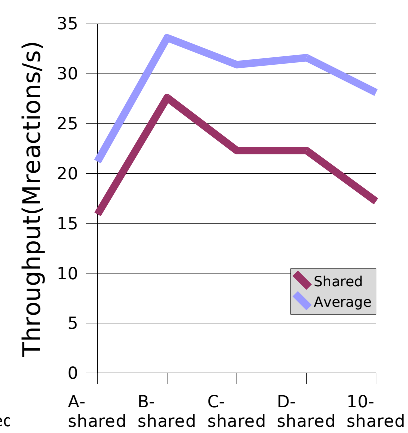


Fig.5 Performance comparison



Some of the predefined functions have similar formulas. Therefoer, shared function solver cores that calculate multiple types of predefined functions can be implemented by sharing some arithmetic units.
Here, 5 types of shared functions are designed. Fig.4 and 5 show the implementation results of shared function solver cores when the critical path first scheduling policy is applied. Functions sharing efficiently reduces the hardware cost of solver cores, performance is degraded.

Conclusion

Two arithmetic scheduling policies for solver cores of biochemical rate law functions are presented, and the tradeoffs between performance and hardware costs are analyzed. By stretching the critical path operation and sharing arithmetic units, hardware resources of solver cores were reduced by 33.8% in average, with up to 11.5% throughput degradation. In addition, by sharing 2 to 4 rate law functions into a solver core, further hardware resource were effectively reduced alleviating the performance degradation.