情報処理学会論文誌　コンピューティングシステム　Vol. 1　No. 3　120–135 (Dec. 2008)

FPGAを用いた高スループット確率モデル生化学シミュレータの設計と評価

吉見真聡[†1]　西川由理[†1]　長名保範[†2]
舟橋啓[†1]　広井賀子[†3]　柴田裕一郎[†4]
山田英樹[†4]　北野宏明[†5]　天野英晴[†1]

確率モデル生化学シミュレーションアルゴリズム（SSA）は，生化学モデルの確率的挙動を厳密に計算できる手法として知られている．しかしその一方で，SSAを用いてモデルを評価するためには，膨大な回数の演算が繰り返されるので，高スループットな実行環境が求められている．本論文では，FPGAを用いて計算効率の良いSSAであるNext Reaction Method（NRM）を高スループットに実行するハードウェアの構造について述べ，性能評価を行う．このハードウェアは，シミュレーションスレッドの状態を保持するスレッドモジュールが，データ転送網を介して算術演算器やデータメモリを共有する構造を持ち，マルチスレッド実行を行う．また，モジュール間の接続を変更することで，特定のFPGAに限定されない構成の変更が可能である．ミドルレンジのFPGA上に構成することを想定して16スレッドの並列実行を行うハードウェアを実装し，評価用によく利用されるHSRモデルによるRTLシミュレーションを通してスループットを計測した結果，Core 2 Quad Q6600 2.40 GHzのシングルスレッド実行と比べて約4.2倍，HSRを並べて作成した反応数の多い仮想のモデルでは約5.4倍のスループットの向上が可能であることを確認した．

Design and Evaluation of an FPGA-based Stochastic Biochemical Simulator for High-throughput Execution

Masato Yoshimi,[†1] Yuri Nishikawa,[†1]
Yasunori Osana,[†2] Akira Funahashi,[†1]
Noriko Hiroi,[†3] Yuichiro Shibata,[†4]
Hideki Yamada,[†4] Hiroaki Kitano[†5]
and Hideharu Amano[†1]

Stochastic biochemical simulation algorithms (SSAs) are generally known as exact methods to trace stochastic behaviors of target biochemical models. Due to vast amount of computation attributed to the nature of Monte Carlo Method, which SSAs are originated from, there is a strong urge for high-throughput execution environment. This paper proposes an FPGA implementation of a stochastic simulation system based on a computationally-efficient SSA called the Next Reaction Method, and studies the evaluation results of area and throughput in detail. The system conducts high-throughput multi-thread execution, using multiple thread modules accessing shared arithmetic and data modules. The network between modules are configurable, and supports flexible network structure according to target FPGAs. In order to evaluate the proposed design, the stochastic simulation system, which is capable of running 16 threads in parallel, was implemented on a middle-range FPGA. As the result of comparing the throughput in RTL simulation with software simulation run on Core 2 Quad Q6600, the system marked 4.2 times higher throughput using a real biochemical model called HSR. When several versions of virtually large-scale models were tested on the same simulation environment, maximum of 5.4 times higher throughput was confirmed.

1. はじめに

確率モデルによる化学反応系の計算機シミュレーションは，細胞のシステムに特有の確率的挙動を再現できることから，システムバイオロジーにおいて重要な役割を果たすようになった．しかし，確率モデル生化学シミュレーションアルゴリズム（Stochastic Biochemical Simulation Algorithm：以下SSA）を用いたモデルの評価には多大な計算時間を要することが広く知られており．シミュレーション対象となる生化学モデルの複雑化にともない，高速実行手法の重要性が高まっている．このため，計算時間を抑えるアルゴリズムや，高速実行システムに関する研究がさかんに行われている．

SSAは様々なレベルの並列性を内包しており，専用ハードウェアを用いた高速化が期待できる．特に，FPGA（Field Programmable Gate Array）のような書き換え可能なデバイスを用いると，対象の生化学モデルに応じて専用のハードウェアを生成でき，コスト性能比に優れた計算システムの実装が可能になる．このため，複数の研究グループがFPGAを

†1 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†2 成蹊大学理工学部情報科学科
Faculty of Science and Technology, Seikei University
†3 European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Trust Genome Campus
†4 長崎大学工学部情報システム工学科
Department of Computer and Information Sciences, Nagasaki University
†5 科学技術振興機構北野共生システムプロジェクト
Kitano Symbiotic Systems Project, ERATO-SORST, Japan Science and Technology Agency

用いた高速化の可能性を模索している[1)–4)].

著者らの研究グループでも，反応数の多い生化学モデルに対して計算効率が良い Next Reaction Method（NRM）を対象アルゴリズムとして，FPGA 上で高速実行するシステムの開発を行ってきた[5)–7)].

本論文では，NRM を高スループットに実行する計算システムの構造について述べる．中程度のサイズの FPGA に構成可能で，ロジック資源の規模に合わせて性能向上できる，より実用性の高い実装を行い，実際の生化学モデルを用いて性能評価を行った．

2. SSA: Stochastic Biochemical Simulation Algorithm

2.1 確率モデル生化学シミュレーション

確率モデルによる生化学モデルのシミュレーションはモンテカルロ法に基づいており，ある時刻における生化学モデルの状態の確率分布を求める目的で使用される．生化学モデルは，以下のように定義される．

- M 種類の化学反応と，N 種類の分子が存在する．これを，反応数 M の生化学モデルと呼ぶ．
- 反応 R_j（j：反応番号：$1, 2, \cdots, M$）は，式 (1) のように表現される．

$$R_j: \quad S_a + S_b \xrightarrow{c_j} S_c + S_d \tag{1}$$

反応は，モデル内部に存在する分子 S_i（$i = 0, \cdots, N$）と確率的反応速度定数 c_j を用いて，反応物（式 (1) 左辺）および生成物（式 (1) 右辺）の組で構成される．

- モデルの時刻 t と各分子 S_i の分子数 X_i の組はモデルの状態を表す．

SSA は，モデルの初期時刻と初期分子数があたえられると，化学反応の発生 1 回を反応サイクルとして，以下の反応選択，状態更新の 2 つの処理を繰り返し，生化学モデルの状態の時間変化を計算する．

(1)　モデルの現在の状態をもとに，次に発生する反応を選択し，その発生時刻を求める．
(2)　選択された反応に従って状態とモデル時刻を更新する．

反応 1 サイクルには，モデルの反応数 M に応じた命令ステップ数の計算を要する．統計的に有意な解析を行うために十分なサンプルを得るには，同一パラメータのモデルについて，異なる乱数系列を用いて数千回以上のシミュレーションを行う必要がある．

2.2 NRM: Next Reaction Method

生化学反応系の確率的なモデリング手法とシミュレーションアルゴリズムは，Gillespie による提案[8)] 以来，様々な改良が進められてきた．精度を犠牲にせず，物理的基礎に厳密に従う SSA の中でも，Gibson らによる Next Reaction Method（以下，NRM）[9)] は，計算効率が良いアルゴリズムとして知られている．

NRM の基礎となった First Reaction Method（SSA の一種．以下，FRM）は，反応選択の処理において，式 (2) を用いて全反応の発生予測時刻 τ_j を求め，そのうち最小値を示した反応を発生反応として選択する．

$$\tau_j = \ln(1/r)/a_j + t \tag{2}$$

r は $(0, 1)$ の範囲の一様乱数，t は現在のモデル時刻を示す．a_j は反応 R_j の反応物の分子数と反応定数から，2 回の乗算で求められる propensity と呼ばれる反応発生頻度である．

NRM は，2 種類のデータ構造を導入することで，従来の SSA の時間計算量 $O(M)$ を $O(\log(M))$ へと改善したアルゴリズムである．それらのデータ構造とは，Indexed Priority Queue（以下，IPQ）と呼ばれる反応番号をインデックスとして持つ 2 分木と，Dependency Graph（以下，DG）と呼ばれる各反応の関係を示す依存グラフである．DG には，反応発生にともなう状態更新で発生予測時刻が変わる反応がリストされる．

NRM では，第 1 反応サイクルのみ，すべての反応の τ_j を計算し，IPQ に格納する．IPQ の 2 分木は，親ノードの値が子ノードの値よりもつねに小さくなるよう構成されるので，発生反応が根ノードから読み出される．第 2 反応サイクル以降の計算手順を**図 1** に示す．反応選択の後，状態が更新される．同時に，選択した反応に応じて，DG から反応のリストが読

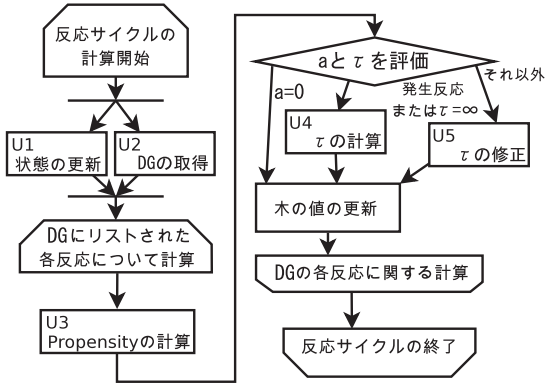


図 1　NRM における第 2 反応サイクル以降の計算手順
Fig. 1　An operation procedure of one reaction cycle in NRM after the second reaction cycle.

み出される．発生した反応について，式 (2) を用いて次の新しい発生時刻を求める（図 1 の関数 U4）．その後，DG にリストされた反応について，発生予測時刻の修正を行う．修正後の発生予測時刻 $\tau_{j,\mathrm{new}}$ を求めるための処理は，状態更新前後の propensity である $a_{j,\mathrm{old}}$ と $a_{j,\mathrm{new}}$，および IPQ から読み出された修正前の $\tau_{j,\mathrm{old}}$ の 3 つの値を判定して選択される．もし $a_{j,\mathrm{new}} = 0$ ならば，反応 R_j は発生しなくなるので $\tau_{j,\mathrm{new}} = \infty$ とする．また，$a_{j,\mathrm{old}} = 0$ かつ $a_{j,\mathrm{new}} \neq 0$ の場合は式 (2)（関数 U4）により新しい $\tau_{j,\mathrm{new}}$ を計算する．それ以外の場合は式 (3)（関数 U5）を用いて $\tau_{j,\mathrm{old}}$ を修正する．

$$\tau_{j,\mathrm{new}} = a_{j,\mathrm{old}}(\tau_{j,\mathrm{old}} - t)/a_{j,\mathrm{new}} + t \tag{3}$$

DG の反応リストの計算が終了し，IPQ が更新された後，IPQ の根ノードには次のサイクルで発生する反応の番号と時刻が格納されている．

NRM は，1 反応サイクルのうちで計算を行う対象が発生反応と DG にリストされた反応のみであるため，算術演算の回数は他の SSA と比べて少ない．また，2 分木の更新時間が大きな割合を占めることが特徴である．これに要する時間計算量は $O(\log(M))$ なので，反応数の多い生化学モデルに対して効率が良いアルゴリズムである．E-Cell3[10] や COPASI[11] などの確率モデルシミュレーション機能を持つ生化学シミュレータは，その高速性に着目し，NRM を使用している．

3. 関 連 研 究

3.1 SSA の性能向上

確率モデルによる生化学シミュレーションは，計算時間の問題を改善するために，アルゴリズムや高速実行システムの研究がさかんに行われている．

アルゴリズムの研究では，前節で述べた NRM[9] および Cao らによる Optimized Direct Method[12] など，計算量の改善やモデルに基づく効率化を導入したアルゴリズムが広く利用されている．また，精度の悪化をある程度許容して高速化を図るアプローチとしては，explicit/implicit τ-leaping 法なども開発されている．現在に至るシミュレーションアルゴリズムの概要と展開は，文献 13) にまとめられている．また，生化学モデル記述言語 SBML からのモデル変換，シミュレーションの実行，MATLAB を用いた結果の解析までを含めた確率モデル生化学シミュレーションツールキット StochKit[14] が公開されている．

ハードウェア処理による高速化のアプローチでは，特に FPGA を対象とする研究がさかんである．Gillespie のアルゴリズムを改良し，生化学モデルを入力として専用回路を生成するシステム[1)–3)] や，処理を部分的に FPGA に任せるシステム[4)] が開発され，評価が行われている．また，PC クラスタを用いた評価[15)] や，GPU でシミュレーションを実行するシステムの開発[14)] も進められている．著者らの研究グループでも，2004 年から FPGA を対象とした SSA 実行システムの研究を行っており[16)]，2006 年からは NRM を対象アルゴリズムとして開発を行っている[5)]．

3.2 FPGA を用いた SSA の高速実行システム

FPGA を用いた SSA の高速実行システムは，複数の研究グループによる開発がそれぞれ進められている．Keane らは，並列処理が可能なシミュレーションアルゴリズムを開発し，モデルごとに専用のハードウェアを生成する計算システムを開発している[3)]．計算時間の短さが特徴であり，浮動小数点演算を使用せず，SSA の各反応の計算をそれぞれハードウェアモジュールに変換して並列に演算を行うことにより，1 サイクルの計算が 3 クロックサイクルで完了する．このシステムは $M = 32$ 以下のモデルを用いて性能が評価され，Pentium4 2.0 GHz による NRM の実行と比べて，約 20 倍の性能向上が報告されている．しかし，1 クロックサイクルでシステムの状態更新や反応の計算を行う必要があるため，Keane らのシステムではすべてのデータをレジスタで保持したうえで，反応ごとに専用の計算モジュールを持つ必要がある．そのため，シミュレータを構成できるモデルの最大反応数は，FPGA のロジック資源の制約を受ける．

また，Thurmon らは DIMM スロット接続型 FPGA 搭載カードを使用し，Direct Method（SSA の一種．以下，DM）のアルゴリズムのうち，propensity の計算と発生反応決定にかかわる計算を FPGA が担うホスト PC–FPGA 協調計算システムを開発している[4)]．このシステムは $M = 14$ のモデルなどを用いて性能評価が行われ，Pentium III 1 GHz を用いた C++プログラムの実行と比較して，FRM の約 10 倍，NRM の約 2.24 倍の高速化を報告している．モデルごとに専用ハードウェアを生成せず，最大反応数にも制約を受けないが，ホスト PC–FPGA 間の大量のデータ通信が速度向上のボトルネックになっている．

3.3 FPGA を用いた First Reaction Method の実装

著者らは以前の研究において，FPGA を用いて FRM を実行するハードウェア（以下 FRM-FPGA）を実装し，評価を行った[17)]．FRM-FPGA の概要を**表 1** に示す．FRM における 1 反応サイクルの計算の大部分はデータ依存性のない浮動小数点演算で構成されることに着目し，深いパイプラインを持つ算術演算回路に間断なくデータを投入することでスループット向上を図った．また，確率モデル生化学シミュレーションが持つスレッドレベルの並列性を利用し，複数シミュレーションタスクの並列処理を行うことにした．そのため，2 スレッドのシミュレーションを実行する FRM 実行ユニットを FPGA 上に 3 ユニット構

表 1　FRM-FPGA の評価環境
Table 1　Evaluation environment of FRM-FPGA.

FRM-UNIT（演算モジュール）の数	3
FRM-UNIT の実行スレッド数	2
実行スレッド合計	6
実装対象の FPGA	Virtex-II Pro（XC2VP70-5）
動作周波数	106.29 MHz
面積（33,088 Slices）	24,496（74.03%）
組み込み乗算器（328 個）	78
BlockRAM（328 個）	54

表 2　C++プログラムの実行環境
Table 2　Execution environment of C++ program code.

CPU	Intel Core 2 Quad Q6600（2.40 GHz）
Memory	3.5 GB（実装 4.0 GB）
OS	Linux 2.6.22-14（x86-32bit）
Compiler	gcc 4.1.3（-O3）1 コア

成し，計 6 スレッドのマルチスレッド実行を行うハードウェアを実装した．シミュレーションデータは FPGA の組み込みメモリ（Virtex-II Pro の BlockRAM：以下 BlockRAM）に格納され，ハードウェアを再構成せずに $M = 1024$ までの生化学モデルについてシミュレーションを実行できる．

FRM と，計算効率が良いとされる NRM の演算性能を定量的に比較するために，それぞれを C++言語で実装した（以下 FRM-SW，NRM-SW）．これらのコードは，マルチコア CPU 向けの並列プログラミング技術を使用しておらず，評価はシングルスレッドでの実行結果である．これらのソフトウェアプログラムと FRM-FPGA について，**表 2** に示す環境で実行し，計測したスループットと反応数の関係を**図 2** にまとめた．

FRM-FPGA については，RTL シミュレーションにより実行時間を求めた．測定したモデルは，Lotka モデル[18]（$M = 4$，$N = 4$）を複数配置し，M の多い生化学モデルを仮想的に定義して使用した．図 2 では，n セットの Lotka モデルからなる生化学モデルを nLotka で示している．

図 2 に示したように，FRM-FPGA はパイプラインの効率利用とマルチスレッド実行により 1 Lotka（$M = 4$）の生化学モデルで FRM-SW の約 6.64 倍，16 Lotka（$M = 64$）以上では約 40 倍のスループット向上が確認できる．しかし，FRM は演算の回数を削減する

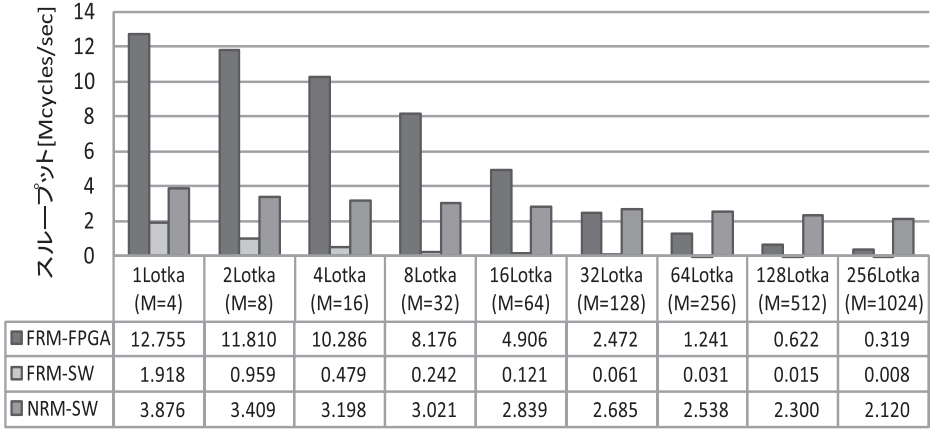


図 2　スループットの比較（単位：Mcycles/sec）
Fig. 2　Throughput comparison (unit: Mcycles/sec).

DG を持っておらず，反応サイクルごとに全反応について式 (2) を計算するので反応サイクルあたりの演算効率が悪い．反応数 M の多いモデルのシミュレーションには不適であり，32 Lotka（$M = 128$）で逆転してしまう．NRM は統計上 FRM と同一の結果が得られることが証明されているので[9]，反応数が大きい場合は，FRM-FPGA が NRM よりも有効な高速化手法であるとはいい難い．

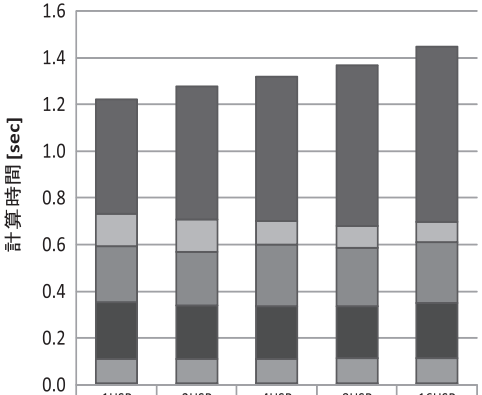
現在では NRM よりも高速とされる SSA の研究も進められている[12] が，時間計算量で NRM を超えるアルゴリズムは開発されていない．そこで本論文では，NRM の持つスレッドレベル，演算レベルの並列性を抽出して FPGA 実装（以下 NRM-FPGA）を行い，反応数が増大しても速度向上が可能な計算システムを実装できることを確認する．

4. FPGA を用いた NRM 実行システム

本章では，FPGA を用いて NRM 実行システムを設計するために，NRM のアルゴリズムを解析し，高速実行の方法を検討する．

4.1 NRM の解析

まず，NRM の実行時間や特徴を調査するために，表 2 に示した環境で，反応数を変えて NRM-SW を実行した場合のプロファイルを取得した．その結果を**図 3** に示す．また，このときの図 1 における各関数のコール回数を計測した結果を**図 4** に示す．シミュレー

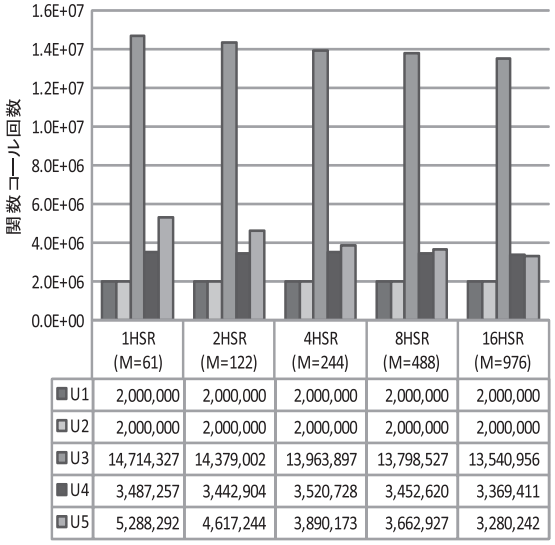


	1HSR (M=61)	2HSR (M=122)	4HSR (M=244)	8HSR (M=488)	16HSR (M=976)
IPQ更新	0.490	0.568	0.621	0.686	0.749
τ修正(U5)	0.137	0.142	0.098	0.094	0.087
τ計算(U4)	0.240	0.227	0.264	0.251	0.260
Propensity計算(U3)	0.243	0.231	0.226	0.222	0.238
状態更新(U1)	0.103	0.104	0.105	0.106	0.104
その他	0.010	0.008	0.009	0.009	0.010
合計	1.223	1.279	1.322	1.368	1.448

図 3　NRM による HSR モデルの計算時間とその内訳（単位：秒）
Fig. 3　Calculation time and its breakout for HSR model in NRM.

ション対象の生化学モデルは，図 2 の Lotka モデルと同様に，大腸菌 *E.coli* の Heat-Shock Response（HSR）モデル（$M = 61$，$N = 28$）を複数配置したモデルを使用した．nHSR は n セットの HSR モデルを配置した生化学モデルを示している．モデルの反応数は，現在実用的と考えられる $M \leq 1000$ の範囲で測定した．HSR モデルは，Cao らが各 SSA の分析に使用しているほか，StochKit [14] のテストケースとして付属されており，評価用として標準的な生化学モデルである．実行時間の内訳は，2×10^6 反応サイクルの計算に要する時間から算出した．図 3 に示したように，計算時間は反応数 M の対数に比例して緩やかに増大する．また，計算時間の増大は IPQ の 2 分木の更新時間に起因する．

また，NRM の計算時間は反応数以外にも DG のリスト長の影響を受ける．DG にリストされた反応数が多ければ，図 4 の最内ループの実行回数が増える．計測を行った反応数の多いモデルは，各 HSR モデルのうち発生しやすい反応が根ノード付近に集中する．発生しやすい反応とは，反応物の分子数や反応定数が他の反応と比べて大きい反応である．このよ



	1HSR (M=61)	2HSR (M=122)	4HSR (M=244)	8HSR (M=488)	16HSR (M=976)
U1	2,000,000	2,000,000	2,000,000	2,000,000	2,000,000
U2	2,000,000	2,000,000	2,000,000	2,000,000	2,000,000
U3	14,714,327	14,379,002	13,963,897	13,798,527	13,540,956
U4	3,487,257	3,442,904	3,520,728	3,452,620	3,369,411
U5	5,288,292	4,617,244	3,890,173	3,662,927	3,280,242

図 4　NRM の関数コール回数
Fig. 4　Number of function calls in NRM.

うな反応は propensity の値が高く，予測発生時刻が現在時刻から近い値となり，IPQ の根ノード付近にとどまることが多いため，選択されやすい（発生しやすい）反応である．反応数に従って U3 の呼び出し回数が減少しているので，HSR モデルの特徴として発生しやすい反応の DG のリスト長が短いことがあげられる．

4.2　FPGA を用いた NRM 実行システムの検討

図 3 に示したように，NRM の実行時間は反応サイクルの計算の繰返しで占められている．また，NRM が実行中に確保するメモリ領域は，約 $24 \times M$ KB（中間状態 $20 \times M$ KB，モデルのパラメータ $4 \times M$ KB 程度）で，非常に小さい．これらの情報をもとに，NRM を FPGA 上で高速実行する方法を検討する．

まず，反応サイクルの一連の計算のすべてを FPGA 上で実行し，初期値の入力と経過の出力を除いては，ホスト PC との協調処理を行わない．これは，反応サイクルが単純な計算の組合せで構成されているので，ホスト PC が処理の一部を担うような構造では，他の処理に比べて通信にかかわるオーバヘッドが大きくなってしまうためである．

また，スレッドレベル並列性を利用し，FPGA 上に NRM を実行するモジュールを複数

構成してマルチスレッド実行によるスループットの向上を図る．以降は，NRM 1 スレッドの中間状態を保持し，図 1 のフローチャートに従って計算を行うハードウェアモジュールをスレッドモジュールと呼ぶ．FPGA の BlockRAM はシミュレーションデータを十分に収めることができるので，データの種類ごとに細かな分散メモリを構成し，それらに並列アクセスすることで処理の高速化を図る．

算術演算部分の高速化手法としては，パイプライン化した演算コアへデータを連続投入してスループットを上げる方法がある．一般的に浮動小数点算術演算器は回路規模が大きいため，スレッドモジュールが演算コアを共有することで効率化を図る．以降，演算コアとその I/O を含むハードウェアモジュールを共有モジュールと呼ぶ．

NRM は選択された反応ごとに，算術演算処理の種類や回数，タイミングが変動するため，前述のモジュール間のデータ転送を静的にスケジューリングすると，パイプラインの空きが大きくなってしまう．そのため，各スレッドモジュールの処理要求を順序制御し，共有モジュールへ受け渡すデータ転送網でモジュール間を接続し，演算コアを使用するための待ち時間を短縮する．このような構造の NRM 実行システムは接続網の構成変更が容易であるため，配置するスレッドモジュールの数を調節することで，FPGA のサイズに合わせた構成をとることができる．

図 5 に，このような構造の NRM 実行システムのモジュール接続図を示す．各スレッドモジュールは NRM 1 スレッドの実行に必要なデータを保持しており，計算の進展に応じて共有モジュールへデータを転送し，結果を受け取る．共有モジュールは 5 種類あり，シミュレーションの状況に依存しない生化学モデルのパラメータが格納されるテーブル（図 5 の U1，U2）と，算術演算を行うモジュール（U3，U4，U5）で構成される．これらの共有モジュールは，図 1 に示された入力を受け取り，処理を行った結果を出力として送り出す．

ここで問題となるのは，モジュール間の接続方式である．最も単純な方式は，これらをマルチプレクサ（MUX）で接続する方式で，まずこれを試みた[6]．しかし，MUX は内部で信号をラッチせず，1 クロックサイクルの間に入力，調停，出力を行う構造としたため，接続するスレッドモジュールが増えると遅延が増大し，マルチスレッド実行による利得が得られなくなってしまうという問題点があった．次に，最近利用が進んでいる Network-on-Chip（NoC）を用いて接続する方法を試みた[7]．この NoC 型システムでは，共有モジュールへの経路が階層化されたため遅延の増大を防ぐことができた．しかし，ルータの出力ポートに BlockRAM を使用していたうえ，スイッチ部分にクロスバが使用されており，本来通信が行われないスレッドモジュール間にも転送経路が存在していたため，要求するロジック資源

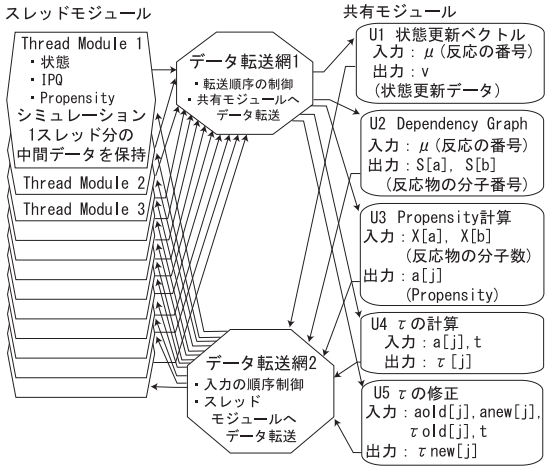


図 5　NRM 実行システムにおけるモジュール間の接続
Fig. 5　Module connection diagram of NRM execution system.

に対して性能が見合わない構造になっていた．

そこで，これらをふまえて 5.5 節で後述する Distributor と Concentrator からなる面積対性能比に優れた独自のネットワークを用いる方式を用いることにした．

5. NRM 実行システムの実装

本章では，NRM 実行システムを構成するスレッドモジュール，共有モジュール，データ転送網の構造と実装について述べる．各モジュールは Verilog-HDL で実装し，浮動小数点演算器やメモリには Xilinx CORE Generator で生成した単精度浮動小数点演算モジュールや BlockRAM を使用している．

5.1　モジュール間のデータ転送方式

各モジュール間はデータパケットを介して転送が行われる．データパケットは 32 ビットを 1 フリットして，ヘッダフリットと 1 フリット以上のペイロードフリットで構成されている．ヘッダフリットには処理内容や，転送網のルーティング手順などの情報が格納される．ペイロードフリットには，共有モジュールでの計算の入力となるデータが格納される．データ転送網では，ヘッダフリットのルーティング情報を参照し，ソースルーティングで転送を行う．その際，モジュール間に接続される制御信号と，FIFO を用いた転送制御を用いて，

後続フリットによる書き潰しや他のパケットによる割込みの発生を回避する．

5.2　スレッドモジュールの構造

スレッドモジュールは，シミュレーション 1 スレッドの中間データを保持し，図 1 の手順に従って NRM の計算を行う．

図 6 に，スレッドモジュールのブロック図を示す．中間データは，生化学モデルの状態（時刻と分子数），IPQ，Propensity の 3 種類のデータで保持できる．これらを保持するメモリは，1,024 ワードの Dual-port BlockRAM で構成した．このことにより，スレッドモジュールは最大で $M = 1023$，$N = 1024$ の生化学モデルの計算が可能となる．$M = 1024$ でない理由は，IPQ が 2 分木を持つため，0 番地のアドレスを使用することができないからである．IPQ は 2 分木における各反応の位置を格納した 10-bits × 1024 のインデックステーブルと，（反応番号，発生予測時刻）の組が格納される 42-bits × 1024 の 2 分木からなる．IPQ は読み出しと更新の 2 種類の機能を担うコントローラが付加されている．IPQ の性質の保持は，5 クロックサイクルで行われる上方向または下方向のレコードと値の比較および交換を繰り返すことで行う．

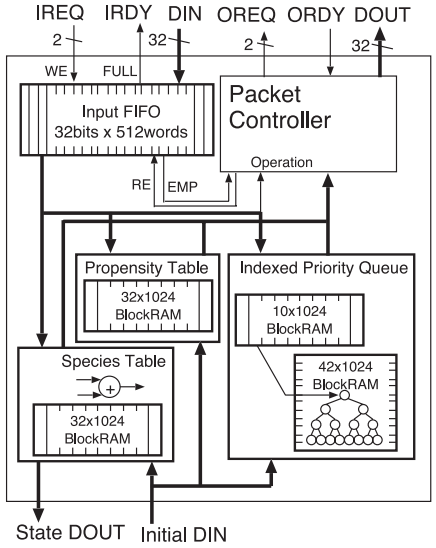


図 6　スレッドモジュールの構造
Fig. 6　Structure of the threaded module.

スレッドモジュールは，他のモジュールからのパケットを受け取る受信用 FIFO と，送信用のコントローラを持つ．スレッドモジュールは，モジュール間の通信ポートとは別に，各 BlockRAM にアクセスするための外部入出力用ポートを持っており，初期値の書き込みおよび結果の読み出しに使用される．

5.3　スレッドモジュールにおける NRM の実行方法

計算手順の制御および外部との通信のため，スレッドモジュール内のパケットコントローラがパケットを生成し，各共有モジュールへ送出する．共有モジュールから返送された結果のパケットは FIFO に蓄えられる．パケットコントローラは 8 つの状態を持つステートマシンであり，処理手順に従って FIFO や各メモリからデータを読み出して次の処理を決定する．

反応 1 サイクルの計算におけるパケットコントローラの状態遷移と送受信パケットの関係を**図 7** に示す．楕円はパケットコントローラの状態，太陽記号は送信パケットの種類，状態遷移のエッジの入力は受信パケットの種類または条件を示している．反応 1 サイクルの計算は，START 状態から始まり，図 1 に従って進展する．最初に計算される U1，U2 の計算は並列実行できるため，IPQ の 2 分木の根データから発生した反応の番号を読み出して U1，U2 宛のパケットを生成し，連続して送出する．各共有モジュール宛のパケットは，そのペイロードとして図 5 の共有モジュールの “入力” に示された値を持つ．その後 IDLE 状態に遷移し，受信用 FIFO にパケットが入力されるまで待つ．FIFO にパケットが入力され始めたとき，FETCH 状態に遷移して FIFO から受信パケットの読み出しを行う．受信パケットが U1 からのものだった場合，“更新” 状態に遷移してペイロードフリットのデータに従っ

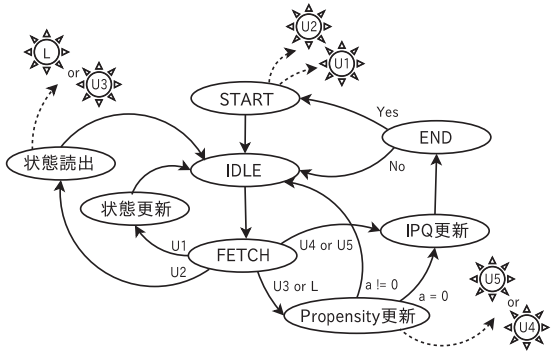


図 7　反応 1 サイクルにおけるパケットコントローラの状態遷移と送受信パケット
Fig. 7　State transition of the packet controller and send/receive packet in a reaction cycle.

て生化学モデルの状態（分子数）の更新を行う．同様に，U2 からのパケットだった場合は，“読み出し” 状態に遷移し，U1 の処理が終わっていれば U3 宛のパケットを，終わっていなければ U3 宛の正しいパケットを送出できないので，自身宛（L：ループパケット）に受信パケットを送出し，処理の終了を待つ．U1 のパケットには発生反応も含めて予測反応発生時刻を修正しなければならない反応の数が含まれており，その数と同数のパケットが，U2 パケットとして返送される．U3 またはループパケットが読み出された場合は，“Propensity 更新” 状態に遷移し，該当する反応の propensity を更新する．この際，$a_{j,\mathrm{old}}$，$a_{j,\mathrm{new}}$，$\tau_{j,\mathrm{old}}$ の値を参照し，パケット（U4 または U5）の送出または IPQ の 2 分木更新など，処理を選択する．こうして，U1 からのパケットで得られた回数，“IPQ 更新” 状態を通過すると，反応サイクルの計算は終了する．

このとき，IPQ の 2 分木の根に格納されているデータのうち，反応番号が次の反応サイクルで発生する反応となり，発生予測時刻の値がその反応が起こるモデル時刻となる．

5.4　共有モジュール

共有モジュールは，入出力ポート，演算コア，アービタで構成される．4.2 節で述べたように，演算コアは 5 種類ある．それらはデータメモリ主体のもの（U1，U2）と算術演算主体のもの（U3，U4，U5）の 2 つに大別される．スレッドモジュールの入出力ポートは 1 セットで固定であるが，共有モジュールの入出力ポート数はスレッドモジュール数や転送網の形状に応じて可変である．それらの例として，入出力ポートを 1 セット持つ U2，および 2 セット持つ U4 のブロック図を，**図 8**，**図 9** にそれぞれ示す．入力ポートは共有モジュールに到着したパケットを内部のレジスタに格納し，演算コアの使用権をアービタに要求する．アービタは，入力ポートからの要求を検出して使用権を割り当て，対応するペイロードフリットを演算コアへ渡す役割を担う．演算コアは，入力されたデータに対し，図 5 に示した各処理を行う．出力ポートは，対応する入力ポートが受信したパケットのヘッダフリットと演算コアから出力された演算結果を内部の FIFO に格納し，それらをもとに生成したパケットを送信元のスレッドモジュールへ返送する．

データメモリ主体の演算コアは，反応番号を入力として，状態更新ベクトル（U1）または反応物の分子番号（U2）からなる可変長のリストを出力する．これは，2 次元配列の読み出しなので，図 8 に示したように，演算コアはデータメモリとポインタメモリを接続した 2 段階の構造を持つ．反応番号をアドレスとして，ポインタメモリからリストのデータ数とデータメモリの先頭アドレスが読み出され，それらをもとにデータメモリからリストが読み出される．データメモリの読み出し中は，アービタは別のパケットに対して使用権の割当てを行わない．出力ポートでは，接続網部分での衝突による待ち時間を減らすため，リストのデータ 1 つで 1 パケットを構成して送出する．

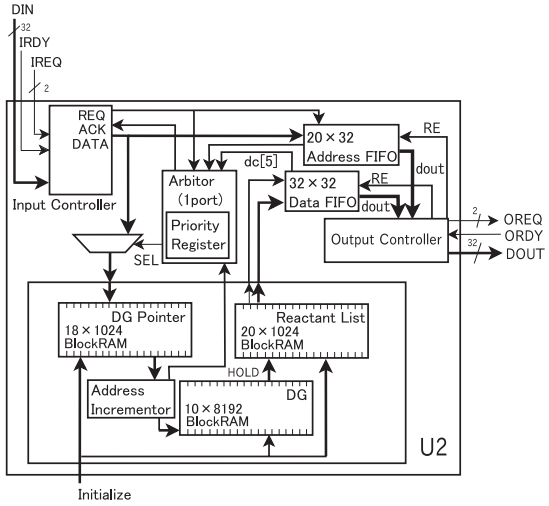


図 8　共有モジュール U2（Dependency Graph）の構造（入出力ポート 1 つ）
Fig. 8　Structure of shared module U2 (Dependency Graph) with a set of I/O port.

一方，算術演算主体の演算コアはパイプライン化されており，1 クロックサイクルごとに連続して要求を処理する構造を持つ．U4（図 9）の演算コアは，単精度浮動小数点数を使用した式 (2) の計算を行う．$(0, 1)$ の乱数は Linear Feedback Shift Register（LFSR）を使用した M 系列による一様乱数から $(1, 2)$ の乱数を作り，1.0 を減算することによって生成する．また，自然対数 e を底とする対数は 2 次の補間法で計算する．生成した乱数は生化学モデルのデータとは独立に生成できるため，FIFO に蓄えられて必要に応じて取り出される．乱数生成モジュールは必要に応じて交換できる．U3，U5 もこれらに準じた形で実装した．

U4 は入力パケットが 3 フリットなので，入出力ポートが 1 セットである場合，パイプラインの稼働率は最大 33%である．これを改善してモジュールの利用効率を上げるためには，入出力ポートを増やして入力部分での転送待ち時間を減らす方法が有効である．その一方で，U1，U2 はパイプライン処理ができないため，入出力ポート数を増やすことによる待ち時間の低減は期待できない．図 1 に示したように，U1，U2 は反応サイクルごとにたかだか 1 回しか使用されない．そのため，この部分での待ち時間が与える影響は小さいと予測され

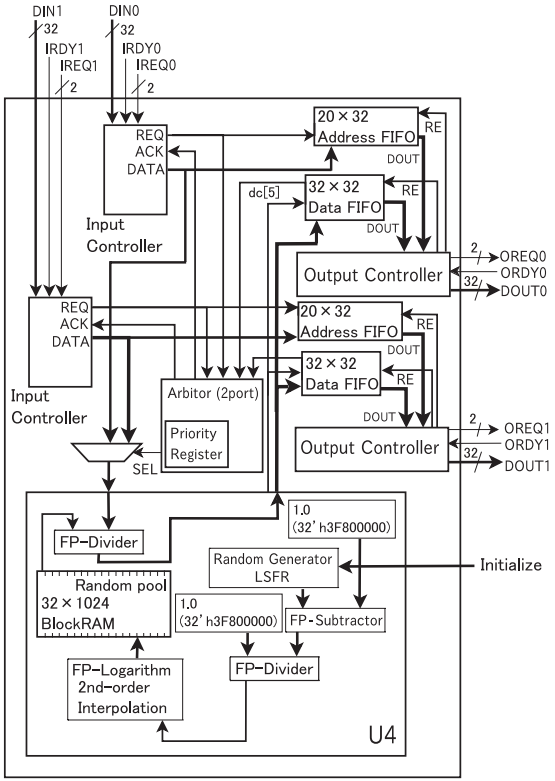


図 9　共有モジュール U4（τ 計算）の構造（入出力ポート 2 つ）
Fig. 9　Structure of shared module U4 (calculates τ) with two sets of I/O port.

るが，待ち時間の低減方法としては U1，U2 を複数配置して対応する方法が考えられる．

5.5　データ転送網

データ転送網は，スレッドモジュールと共有モジュールを接続し，パケットの転送を制御する．データ転送網は，複数の入力の順序を制御して 1 つの出力にまとめる Concentrator と，入力をヘッダフリットの情報に従って選択されたポートに出力する Distributor を組み合わせて構成される．データパケットは各スレッドモジュールから送出され，共有モジュールで計算を行った結果が戻る，というデータ通信の特徴から，これら 2 種類のモジュールで十分な接続網を構成できる．**図 10** に，入出力ポートが 4 セットの Concentrator と Distributor の

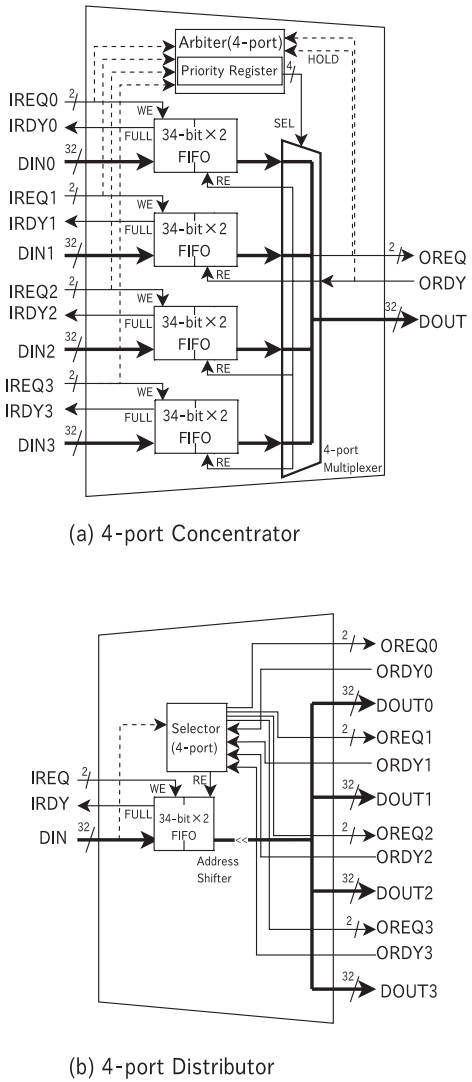


図 10　接続網を構成するモジュール（4 ポートの例）
Fig. 10　Examples of 4-port interconnection modules.

ブロック図を示す．両者のポートの数は可変である．これらの各入力ポート部分は 34-bit × 2 の FIFO を持ち，多段接続の際の配線遅延の増大を抑える．

Concentrator では，複数の入力ポートからのデータパケット転送要求を調停し，1 つの出力ポートからの出力にまとめる機能を持つ．アービタは各入力ポートからの制御情報を参照し，出力ポートのデータ信号線との接続を行う．Distributor では，ヘッダフリットのルーティング情報に従って出力先ポートを選択し，パケットを転送する．

5.6　NRM 実行システム

スレッドモジュールと共有モジュールをデータ転送網で接続し，NRM 実行システムを構成する．**図 11** に，4 つのスレッドモジュールを持つ NRM 実行システムの構成を示す．この実行システムでは，共有モジュールに生化学モデルのパラメータを書き込み，各スレッドモジュールに初期値を格納して計算を開始すると，スレッドモジュールは，5.3 節で述べた手順に従って反応サイクルの計算を繰り返す．初期値として設定された反応サイクル数ごとに，生化学モデルの状態を出力する．

図 11 のような，p 個のスレッドモジュールと 1 セットの入出力ポートを持つ共有モジュールの間が 1 組の Concentrator，Distributor で接続されたハードウェアを Tp の NRM 実行システムと呼ぶ．Tp の実行システムは，データ転送網においてパケットが集中する部分に関する調査が容易であり，構造を規則的に拡張できるので，面積および性能は p を変更して評価を行った．

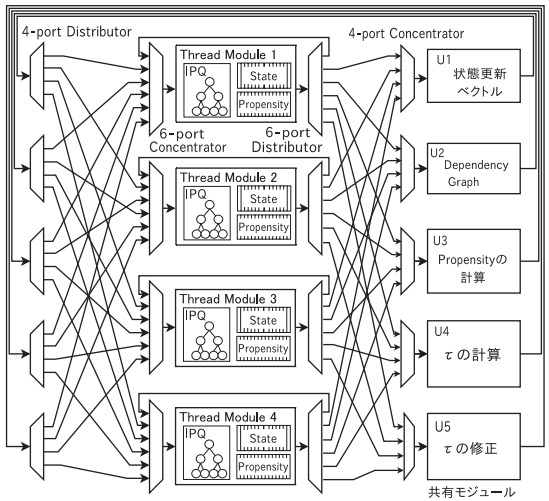


図 11　4 個のスレッドモジュールを持つ NRM 実行システムの構造
Fig. 11　Structure of NRM execution system with 4 threaded modules.

Tp の実行システムは各共有モジュールの使用頻度が等しい場合には妥当な構造であるが，偏りがある場合は，使用頻度が高い共有モジュールの入出力部分が性能を抑えるボトルネックになる．図 4 に示したように，U3 の共有モジュールの使用頻度が高くなることは明らかである．そのため，動作周波数の下落が許容範囲内で，かつ U3 に対する通信がボトルネックになると考えられる T16 の実行システムを対象に，接続網部分を一部変更した実行システム T16C を実装した．T16C は，U3 の入出力ポートを 4 セットに拡張した U3C を使用し，各ポートが 4 つのスレッドモジュールと 4 ポートの Concentrator，Distributor を介して接続され，U3 の演算コアのパイプライン稼働率の向上を図った実行システムである．

6. 評　　価

本章では，東京エレクトロンデバイスが販売している FPGA 評価ボード TB-5V-LX110T-PCIEXP [19]（搭載 FPGA：XC5VLX110T-FF1136）を想定し，前章で述べた NRM 実行システムの評価について述べる．合成および配置配線には Xilinx ISE8.2i を使用し，性能は RTL シミュレーションで計測した．

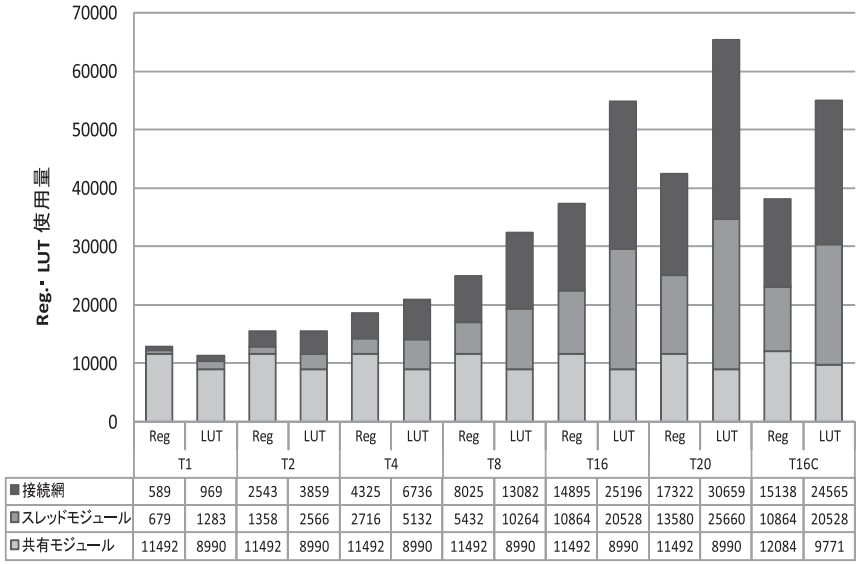
6.1　各モジュールの面積評価

まず，前章で述べた各モジュールの単体での面積，動作周波数を評価し，**表 3** にまとめた．この結果をもとに，FPGA 上に配置可能なスレッドモジュール数について検討する．スレッドモジュールは FPGA のうち LUT を約 2.1%（= 1283/69120 × 100），BlockRAM を約 4.1%使用する．このことから，FPGA 上に構成できるスレッドモジュールの最大数は，

表 3　各モジュールの面積と動作周波数
Table 3　Area and operating frequency of each module.

	Thread	U1	U2	U3	U4	U5	U3C
Registers	679	161	215	1,028	7,231	2,857	1,620
LUTs	1,283	348	384	993	5,154	2,111	1,774
BlockRAM/FIFO	6	5	7	4	6	2	9
DSP48Es	0	0	0	5	14	5	5
Max. Delay [ns]	6.40	5.44	5.42	5.73	5.84	4.29	5.73
Op. Freq. [MHz]	156.30	183.82	184.54	174.43	171.38	233.10	174.43

⋆ XC5VLX110T-FF1136: Slice 69,120: LUTs 69,120: BlockRAM/FIFO 148: DSP48E 64



	Reg	LUT	Reg	LUT	Reg	LUT	Reg	LUT	Reg	LUT	Reg	LUT	Reg	LUT
	T1		T2		T4		T8		T16		T20		T16C	
接続網	589	969	2543	3859	4325	6736	8025	13082	14895	25196	17322	30659	15138	24565
スレッドモジュール	679	1283	1358	2566	2716	5132	5432	10264	10864	20528	13580	25660	10864	20528
共有モジュール	11492	8990	11492	8990	11492	8990	11492	8990	11492	8990	11492	8990	12084	9771

図 12　NRM 実行システムのリソース使用量
Fig. 12　Resource utilization of NRM execution system.

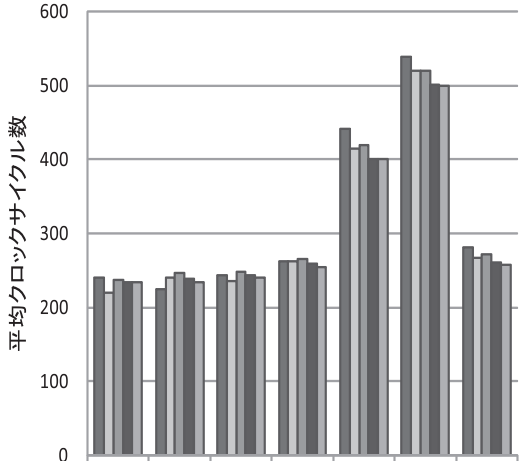
FPGA が持つ BlockRAM の数に制約される．U1，U2 は，生化学モデルのパラメータを保持するために BlockRAM を多く使用し，演算コアの制御ロジックが単純なことから他のモジュールと比べ，ロジック資源は少ない．また，主に浮動小数点演算器で演算コアが構成される U3，U4，U5 は，使用するロジック資源が多い．

6.2　NRM 実行システムの面積評価

実装した NRM 実行システムの面積を**図 12** に，動作周波数を**表 4** にまとめた．図 12 に示したように，T20 のシステムでも FPGA に収まる範囲である．表 4 に示したように，FPGA のロジック面積に余裕があるスレッドモジュール数が T8 以下のシステムでは，動作周波数が下落せず，クリティカルパスは表 3 で示されたように，スレッドモジュールの内部にある．しかし一方で，T16，T20，T16C の NRM 実行システムではポート数が増大した Concentrator，Distributor がクリティカルパスとなり，遅延が増大する．これらのシステムでは，接続網部分が，配線用に多くの LUT を使用していることが分かる．T16 と T16C を比べると，Slice Register が 1,000 程度（約 2.24%）大きくなっているものの，LUT 数

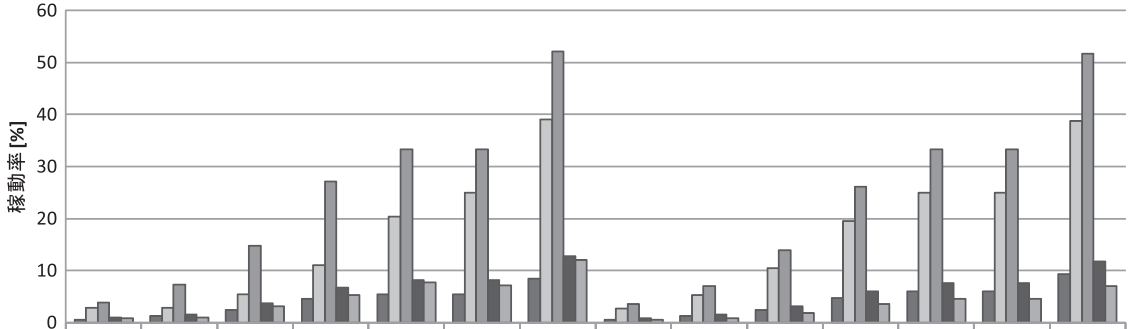
表 4　NRM 実行システムの動作周波数
Table 4　Operation frequency of NRM execution system.

実行システム	動作周波数 [MHz]
T1	156.30
T2	156.30
T4	156.30
T8	155.62
T16	141.30
T20	120.69
T16C	138.60



	T1	T2	T4	T8	T16	T20	T16C
1HSR (M=61)	239.68	224.51	244.21	262.21	441.06	538.07	281.74
2HSR (M=122)	220.63	239.54	235.98	261.81	415.21	520.46	267.35
4HSR (M=244)	237.28	247.34	248.68	265.76	419.34	520.64	271.45
8HSR (M=488)	233.96	239.43	242.71	258.42	399.76	501.43	261.11
16HSR (M=976)	234.55	233.67	239.67	255.23	399.86	499.64	257.24

図 13　1 反応サイクルの計算に要する平均クロックサイクル数
Fig. 13　Average number of clock cycles to calculate one reaction cycle.



	T1	T2	T4	T8	T16	T20	T16C	T1	T2	T4	T8	T16	T20	T16C
	HSR1 (M=61)							HSR16 (M=976)						
U1	0.62	1.33	2.45	4.56	5.43	5.51	8.50	0.64	1.28	2.49	4.68	5.97	5.97	9.28
U2	2.87	2.87	5.52	11.09	20.33	24.99	39.04	2.68	5.29	10.43	19.58	25.00	25.00	38.74
U3	3.82	7.36	14.79	27.11	33.32	33.33	52.06	3.57	7.05	13.91	26.11	33.33	33.33	51.66
U4	0.95	1.57	3.67	6.72	8.12	8.22	12.76	0.83	1.57	3.16	5.98	7.66	7.61	11.75
U5	0.87	1.06	3.21	5.36	7.80	7.23	12.02	0.51	0.93	1.89	3.60	4.52	4.59	6.98

図 15　各演算コアの稼働率（単位：%）
Fig. 15　Operation rate of each functional core (unit: %).

や動作周波数はほぼ同等である．算術演算主体の共有モジュールでは，U3C のような入出力を複数セット持つことにより，各入出力ポートに接続されるスレッドモジュールを分割すると，面積の大きな増加をともなわずに演算コアの稼働率の上限を上げることができる．

6.3　NRM 実行システムの性能評価

実装した各 NRM 実行システムの性能を，スレッドモジュール数と反応数の双方で評価する．そのため，4.1 節でも使用した HSR を n セット配置した nHSR について，それぞれ 50,000 反応サイクルの RTL シミュレーションを実行した．その結果について，反応 1 サイクルの計算に要する平均クロックサイクル数を**図 13** に，データパケットの種類ごとに計測した隣接モジュールへ 1 フリット送信する際の平均待ち時間を**図 14** に，各共有モジュールの演算コアの稼働率を**図 15** にそれぞれまとめた．

まず，スレッドモジュール数が性能に与える影響について検討する．図 13 に示したように，スレッドモジュール数の増加に従い，クロックサイクル数は増大する．特に T16，T20 は顕著であるが，これは図 14 に示したように，U3 に対するデータ転送の待ち時間が原因である．演算コアの稼働率は，図 15 に示したようにスレッドモジュール数に比例して増大するが，T16，T20 では，U3 の演算コアの稼働率が最大の 33.3%で飽和し，計算全体を律

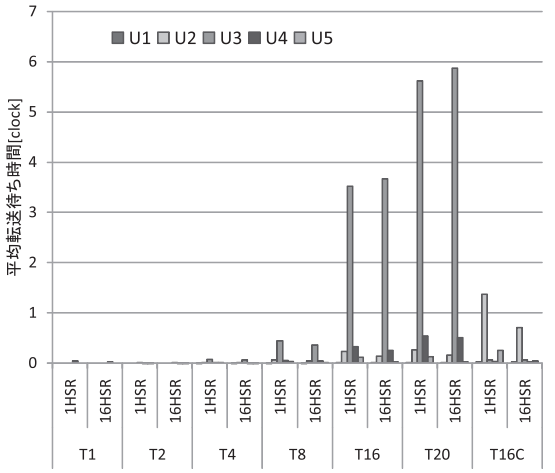


図 14　各パケットの平均転送待ち時間
Fig. 14　Average waiting time to transfer each packet.

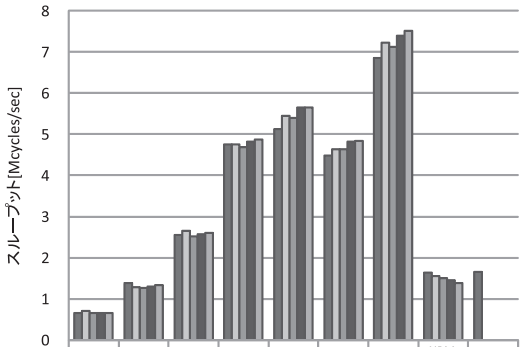
速している．4 セットの入出力ポートを持つ U3C の最大稼働率は 100%なので，T16C は U3 部分のボトルネックを解消している．

次に，反応数とクロックサイクル数との関係について検討する．図 4 に示したように，並べた HSR モデルの数が増えると，IPQ のノード数は増大するが，U3 と U5 の関数コール回数は減少する．

データ通信網にパケットの追い抜きや経路分散の機能がないこと，および次の反応サイクルは現反応サイクルの完了を開始条件とすることから，データ転送網の 1 カ所がボトルネックになると，それが全体の計算速度を抑える原因になってしまう．そのため，稼働率と転送待ち時間を比較しながら，最適な接続構造を明らかにすることが今後の課題として考えられる．

6.4　スループットの比較

図 16 に，各実行システムのスループットと汎用プロセッサにおける NRM プログラムで計測したスループットをまとめた．また，参考値として StochKit の DM によるスループットも示した．FPGA を用いた実行システムのスループットは，動作周波数を反応 1 サイクルに要するクロックサイクル数（図 13）で除算し，スレッドモジュール数を乗じることで求めた．動作周波数の値は，T1 から T8 までは 150 MHz，T16，T16C では 135 MHz，T20 では 120 MHz を使用した．



	T1	T2	T4	T8	T16	T20	T16C	NRM-SW	StockKit
1HSR (M=61)	0.652	1.392	2.560	4.748	5.126	4.486	6.854	1.636	1.661
2HSR (M=122)	0.708	1.290	2.649	4.755	5.445	4.638	7.223	1.564	
4HSR (M=244)	0.659	1.264	2.514	4.684	5.391	4.636	7.113	1.513	
8HSR (M=488)	0.668	1.306	2.576	4.817	5.655	4.814	7.395	1.462	
16HSR (M=976)	0.666	1.338	2.609	4.878	5.654	4.831	7.506	1.382	

図 16　スループットの比較（単位：Mcycles/sec）
Fig. 16　Comparison of throughput (unit: Mcycles/sec).

T1 から T8 まではスループットの線形な増大が認められる．しかし，図 16 に示したように，クロックサイクル数の増大と動作周波数の下落により，T20 のシステムでは T16 の性能に劣るようになってしまう．一方，T16C は U3 のボトルネックを解消している．

図 3 に示したように，マイクロプロセッサは反応数に応じて，IPQ へのアクセス時間が増大する．その一方で，NRM 実行システムでは反応数が性能下落の要因になっておらず，HSR モデルでは反応数の多いモデルの方がスループットが向上しており，共有モジュールの使用回数がクロックサイクル数に影響すると考えられる．

本論文で対象としている FPGA には，T8 を 2 つ構成することができる．2 つの T8 は独立しており並列実行可能なので，スループットは 2 倍になる．これは T16C よりも高速であるので，単一の実行システムの計算効率の検討とともに，FPGA 上に複数の実行システムを構成する際の性能の調査は今後の課題である．

最も高い性能を示した単一の T16C の実行システムでは，NRM-SW と比べて約 4.2 倍から 5.4 倍のスループット向上が認められた．StochKit と比較すると，HSR モデルの場合，DM での実行の方が若干 NRM-SW よりも高速であるとの結果が得られた．また，反応数が大きい場合も，ソフトウェア実行と比べて性能差は広がっており，$M \leq 1023$ の生化学システムにおいては性能のスケーラビリティが確認されたということができる．

7.　まとめと今後の展開

本論文では，確率モデル生化学シミュレーションアルゴリズムの一種である Next Reaction Method を FPGA で実行するシステムを実装し，その評価を行った．

NRM 実行システムにはスレッドモジュールと共有モジュールが配置され，それらがデータ転送網で接続される構造を持っている．NRM 実行システムでは複数のスレッドモジュールが独立したシミュレーションを並列実行し，高スループット化を図る．計算時間や面積は，スレッドモジュールの数や接続構造を変えて評価した．また，転送待ちが生じる部分を改良した実行システムについても実装，評価を行い，待ち時間を低減する方法について検討した．

HSR モデルをシミュレーション対象として性能評価を行い，Core 2 Quad Q6600 2.40 GHz

における NRM の実行と比べ約 4.2 倍，HSR を並べて作成した仮想的な反応数の多い生化学モデルでは約 5.4 倍のスループット向上を確認した．

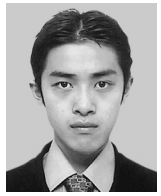
NRM 以降のチューニングされた SSA では，生化学モデルのパラメータや状態に着目して高速化を検討する段階にある[12]．今後は，実機による性能測定やホスト PC を含めた統合的な実行環境の構築を行う．また，アルゴリズムだけでなく，生化学モデルの性質に着目した実行システムの改良も行っていく予定である．

謝辞　本研究は東京大学大規模集積システム設計教育研究センターを通し，ケイデンス株式会社の協力で行われた．

参 考 文 献

1) Lok, L.: The need for speed in stochastic simulation, *Nature Biotechnology*, Vol.22, No.8, pp.964–965 (2004).
2) Salwinski, L. and Eisenberg, D.: In silico simulation of biological network dynamics, *Nature Biotechnology*, Vol.22, No.8, pp.1017–1019 (2004).
3) Keane, J.F., Bradley, C. and Ebeling, C.: A Compiled Accelerator for Biological Cell Signaling Simulations, *The 12th Int. Symp. on Field-Programmable Gate Arrays* (*FPGA*), pp.233–241 (2004).
4) Thurmon, B.P., McCollum, J.M., Peterson, G.D., Cox, C.D., Samatova, N.F., Sayler, G.S. and Simpson, M.L.: Accelerating Exact Stochastic Simulation using Reconfigurable Computing, *International Conference on Engineering of Reconfigurable Systems and Algorithms* (2005).
5) Yoshimi, M., Osana, Y., Iwaoka, Y., Nishikawa, Y., Kojima, T., Funahashi, A., Hiroi, N., Shibata, Y., Iwanaga, N., Kitano, H. and Amano, H.: An FPGA Implementation of High Throughput Stochastic Simulator for Large-Scale Biochemical Systems, *The 16th International Conference on Field Programmable Logic and Applications* (*FPL'06*), pp.227–232 (2006).
6) Yoshimi, M., Iwaoka, Y., Nishikawa, Y., Kojima, T., Osana, Y., Funahashi, A., Hiroi, N., Shibata, Y., Iwanaga, N., Yamada, H., Kitano, H. and Amano, H.: FPGA Implementation of a data-driven Stochastic Biochemical Simulator with the Next Reaction Method, *The 17th International Conference on Field Programmable Logic and Applications* (*FPL'07*), IEEE, pp.254–259 (2007).
7) Yoshimi, M., Nishikawa, Y., Kojima, T., Osana, Y., Funahashi, A., Hiroi, N., Shibata, Y., Yamada, H., Kitano, H. and Amano, H.: A Framework for Implementing a Network-Based Stochastic Biochemical Simulator on an FPGA, *International Conference on Field-Programmable Technology* (*ICFPT'07*), pp.193–200 (2007).
8) Gillespie, D.T.: A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions, *Journal of Computational Physics*, Vol.22, pp.403–434 (1976).
9) Gibson, M.A. and Bruck, J.: Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels, *Journal of Physical Chemistry A*, Vol.104, No.9, pp.1876–1889 (2000).
10) Takahashi, K., Yugi, K., Hashimoto, K., Yamada, Y., Pickett, C.J.F. and Tomita, M.: A multi-algorithm, multi-timescale method for cell simulation, *Bioinformatics*, Vol.20, No.4, pp.538–546 (2004).
11) Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P. and Kummer, U.: COPASI — a COmplex PAthway SImulator, *Bioinformatics*, Vol.22, No.24, pp.3067–3074 (2006).
12) Cao, Y., Li, H. and Petzold, L.: Efficient formulation of the stochastic simulation algorithm for chemically recting systems, *Journal of Chemical Physics*, Vol.121, No.9, pp.4059–4067 (2004).
13) Gillespie, D.T.: Stochastic Simulation of Chemical Kinetics, *Annual Review of Physical Chemistry*, Vol.58, pp.35–55 (2007).
14) Li, H., Cao, Y., Petzold, L.R. and Gillespie, D.T.: Algorithms and Software for Stochastic Simulation of Biochemical Reacting Systems, *Biotechnology Progress*, Vol.24, No.1, pp.56–61 (2007).
15) Schwehm, M.: Parallel Stochastic Simulation of Whole-Cell Models, *Proc. 2nd International Conference on Systems Biology*, pp.333–341 (2001).
16) Yoshimi, M., Osana, Y., Fukushima, T. and Amano, H.: Stochastic Simulation for Biochemical Reactions on FPGA, *The 14th International Conference on Field Programmable Logic and Applications*, Lecture Notes in Computer Science, Vol.3203, pp.105–114, Springer (2004).
17) 吉見真聡，長名保範，岩岡　洋，西川由理，小嶋利紀，柴田裕一郎，岩永直樹，舟橋　啓，広井賀子，北野宏明，天野英晴：FPGA を用いた確率モデル生化学シミュレータ，情報処理学会論文誌：コンピューティングシステム，Vol.48, No.SIG 3 (ACS 17), pp.45–58 (2007).
18) Gillespie, D.T.: Exact Stochastic Simulation of Coupled Chemical Reactions, *The Journal of Physical Chemistry*, Vol.81, No.25, pp.2340–2461 (1977).
19) Tokyo Electron Device: Virtex-5 LXT/SXT PCI Express Evaluation Platform Board. http://www.inrevium.jp/eng/x-fpga-board/hibiki.html

(平成 20 年 5 月 9 日受付)
(平成 20 年 8 月 20 日採録)



吉見　真聡（学生会員）

2006 年慶應義塾大学大学院理工学研究科修士課程修了．現在，同大学院後期博士課程在学中．2006 年より日本学術振興会特別研究員．リコンフィギャラブルシステムの研究に従事．電子情報通信学会会員．

西川　由理（学生会員）

2008 年慶應義塾大学大学院理工学研究科修士課程修了．現在，同大学院後期博士課程在学中．2008 年より日本学術振興会特別研究員．計算機アーキテクチャ，リコンフィギャラブルシステムの研究に従事．

長名　保範

2006 年慶應義塾大学大学院理工学研究科後期博士課程修了．工学博士．現在，成蹊大学理工学部情報科学科助教．計算機アーキテクチャ，システムバイオロジー，比較ゲノム等の研究に従事．電子情報通信学会，IEEE-CS 各会員．

舟橋　　啓（正会員）

2000 年慶應義塾大学大学院理工学研究科後期博士課程修了．工学博士．現在，同大学理工学部生命情報学科講師．システム生物学，計算生物学の研究に従事．IEEE，IEEE-CS 各会員．

広井　賀子

2002 年東京大学大学院医学系研究科博士課程修了．医学博士．現在，European Bioinformatics Institute（EMBL-EBI）研究員．システム生物学，計算生物学，分子生物学の研究に従事．

柴田裕一郎（正会員）

2001 年慶應義塾大学大学院理工学研究科後期博士課程修了．工学博士．現在，長崎大学工学部情報システム工学科准教授．リコンフィギャラブルシステムの研究に従事．電子情報通信学会，IEEE-CS 各会員．

山田　英樹

2007 年長崎大学工学部情報システム工学科卒業．現在，同大学大学院生産科学研究科修士課程在学中．リコンフィギャラブルシステムの研究に従事．

北野　宏明

1991 年京都大学博士号（工学博士）取得．現在，ソニーコンピュータサイエンス研究所取締役副所長．特定非営利活動法人システム・バイオロジー研究機構会長．慶應義塾大学大学院理工学研究科客員教授．システムバイオロジー・人工知能・ロボティクス等の研究に従事．

天野　英晴（正会員）

1986 年慶應義塾大学大学院理工学研究科電気工学専攻博士課程修了．工学博士．現在，同大学理工学部情報工学科教授．計算機アーキテクチャ，リコンフィギャラブルシステムの研究に従事．電子情報通信学会，IEEE 各会員．