修士論文　　2005 (平成 17) 年度

論文題目

FPGA を用いた
確率モデル生化学シミュレータの設計と評価

慶應義塾大学大学院理工学研究科
開放環境科学専攻

学籍番号 80420792　吉見 真聡

指導教員　教　授　天野 英晴

慶應義塾大学大学院理工学研究科

2006 年 3 月

修 士 論 文 要 旨

開放環境科学専攻	学籍番号 80420792	フリガナ 氏名	ヨシミ マサト 吉見 真聡

(論文題名)

FPGA を用いた確率モデル生化学シミュレータの設計と評価

(内容の要旨)

細胞全体を含めた化学反応を計算機上で再現しようとする試みは，1970 年代から検討されており，1990 年代の計算機の計算能力の飛躍的な向上とその導入に係わるコストの低下により，ある程度の実用的なアプローチが可能になってきている．2000 年代に入り，実験とシミュレーションを繰り返し，細胞の挙動を生化学システムとして数理モデル化しようとする研究が行われており，生物学分野・計算機分野の学際的な領域としてシステム生物学 (Systems Biology) は注目を集めている．また，計算機上のモデルを表現するモデル記述言語 (SBML) 等，基本的な計算環境が整備されてきており，システム生物学の研究はさらに発展することが予想される．

しかし，計算機の能力は急速に向上しているにもかかわらず，シミュレーションに要する計算時間の問題は依然として存在している．計算機能力の向上に応じて，生化学システムも大規模化しており，細胞全体を扱うような，反応が数百種類定義された生化学システムのシミュレーション手法の確立には課題が多く存在している．

特に，Gillespie の開発した確率モデル生化学シミュレーションアルゴリズム (SSA) は，計算サイクルを反応 1 回とするモンテカルロ法であるため，計算回数は膨大になる．SSA 実行に長大な計算時間が掛かることは広く知られており，高速化に関して，アルゴリズム，実行環境の両面から研究が続けられている．

本研究では，確率モデル生化学シミュレーションの高速実行環境を，FPGA のようなリコンフィギャラブルシステムを用いて低コストで提供することを目的に，2 種類の SSA をそれぞれハードウェア処理する回路を実装した．ひとつは，First Reaction Method(FRM) と呼ばれる SSA 中で最も基本的なアルゴリズムであり，パイプライン化された演算ユニットに連続的にデータを投入することにより，Xeon 2.80GHz で実行する場合の約 60 倍のスループットを実現した．また，Next Reaction Method(NRM) と呼ばれる高効率な SSA に関して，データ部分・演算部分をネットワーク状に接続することにより，回路面積の有効利用とスループットの線形向上が実現できることを示した．NRM はデータユニット 1 つだけの場合の評価を取り，反応 100 種類以上の大規模な生化学システムの場合，Xeon 2.80GHz でのソフトウェア実行の 0.5 倍程度の計算能力が得られた．

慶應義塾大学大学院理工学研究科前期博士課程

(内容の要旨は約 25 行程度で記入のこと)

Summary of M. S. Thesis

School	Student No.	SURNAME, Firstname
Open and Environmental System	80420792	YOSHIMI, Masato

Title

Design and Evaluation for a Scalable Stochastic Biochemical Simulator on FPGA

The attempt for simulating a whole-cell behavior as a sequence of chemical reactions on computers has been explored since 1970s. A practical approach for cell simulation becomes possible due to the decrease of the cost related to a technological advance of high-performance computers in 1990s. The systems biology which challenges to build a mathematical models for cell behavior through repeating the experiment and the computer simulation attracts attention as an interdisciplinary area in the biology and the computer science.

However, the problem at the calculation time required for the simulation exists still though the ability of the computer has improved rapidly. The biochemical systems also are making to a large scale according to the improvement of the computer performance, and a lot of problems exist in the establishment of the simulation technique of biochemical systems in which hundreds of kinds of reactions that treat the entire cell are defined.

Especially, stochastic biochemical simulation algorithm (SSA) developed by Gillespie is a Monte Carlo method of which one time of the reaction is the calculation cycle, the calculation frequency becomes enormously. So, the research is continued from both sides of the algorithm and the execution environment for speed-up.

In this research, two kinds of high-throughput FPGA-based stochastic biochemical simulator are implemented and evaluated for the accomplishment a low-cost and a high-performance SSA execution environment. One executes the most basic SSA that was called First Reaction Method (FRM). By continuously injectioning data to the pipelined functional unit, it is archived about 60 times higher throughput than executing on Xeon 2.80 GHz.

Moreover, the other was executes highly effective SSA that was called Next Reaction Method(NRM-FPGA). It is able to achieve an effective use for the circuit area and a linear improvement of throughput by connecting the data-unit and the functional-unit as the network. It took the evaluation for the NRM-FPGA which contains only one data-unit(NRM-DU1), and the calculation ability is archived about x0.5 to x1.0 times compared with the software execution on Xeon 2.80GHz at a large-scale biochemical systems in which more than 100 reactions are defined. NRM-FPGA is able to mount 10-15 data-units, it indicates possibilities high-througput solution for the large-scale biochemical simulation.

Graduate School of Science and Technology, KEIO UNIVERSITY

(The summary of thesis should be written within 35 lines)

目 次

図 目 次

表 目 次

第 1 章

緒論

細胞全体を含めた一連の化学反応を計算機上で再現しようとする試みは，1970 年代から検討されてきており，化学反応の組合せで表現された生化学システムの，分子濃度や分子数の時間変化を計算する手法が複数開発されてきている．1990 年代の，計算機の計算能力の飛躍的な向上と，その導入，運用維持に係わるコストの低下により，種々の生物の遺伝子配列や，細胞内の代謝系などに関する定量的なデータが蓄積されつつあり，それらを利用しやすい環境が整いつつある．

2000 年代に入り，実験とシミュレーションを繰り返し，細胞の挙動を生化学システムとして数理モデル化しようとする研究が行われており，生物学分野・計算機分野の学際的な領域としてシステム生物学 (Systems Biology) は注目を集めている．また，計算機上のモデルを表現するライブラリ (SBML) 等，基本的な計算環境が整備されてきており，システム生物学の研究はさらに発展することが予想される．生化学シミュレーションはモデル化に必要なデータが決して充分とはいえず，シミュレーション自体はまだ研究途上の課題であるが，生物学実験には常に倫理的な問題が絡むことや，個体の生長に時間を要することなどから，特に高等生物を対象にした実験は困難であり，シミュレーションの実現に対する期待はきわめて大きい．

計算機を利用してシミュレーションを行う，という研究は比較的古くから行われており，1983 年に開発された Kinsim[1] に汎用シミュレータのはじまりを見ることができる．1980 年代から 1990 年代にかけて Gepasi[2], DBsolve[3] などに代表される，代謝系のモデル化・シミュレーション・パラメータ最適化を行うソフトウェアが開発されるようになり，1990 年代後半になると計算機の性能向上や入手可能なデータの増加に伴って，細胞内の代謝反応すべてをシミュレーションするための E-Cell[4], The Virtual Cell[5] といったシミュレータの開発が行われている．

しかし，計算機の能力は急速に向上しているにもかかわらず，シミュレーションに要する計算時間の問題は依然として存在している．計算機能力の向上に応じて，生化学システムも大規模化しており，細胞全体を扱うような，反応が数百種類定義された生化学システムのシミュレーション手法の確立には課題が多く存在している．また，代謝系を微分方程式にモデル化して解析的にシミュレーションを行う解析モデルシミュレーションに対して，生化学システムで発生する反応を確率を使って計算する確率モデルシミュレーションアルゴリズム (SSA) が 1976 年に Gillespie[6] によって開発された．1990 年代後半より，細胞の生長と分裂の挙動をシミュレーションする STOCKS[7], 分子の立体構造や共有結合の規則変化をシミュレーションする STOCHSIM[8] といった，確率モデルで大規模生化学システムにアプローチするシミュレータが開発されてきている．特に，SSA は計算サイクルを反応 1 回とするモンテカルロ法であるため，計算回数は膨大になる．SSA 実行に長大な計算時間が掛かることは広く知られており，高速化に関して，アルゴリズム，実行環境の両面から研究が現在でも続けられている．その一方で分子動力学や流体力学といった分野では，専用計算機が性能や成果の面で大きな成果を挙げてきており，ハードウェア処理による高速化へ

の期待は大きい.

生化学シミュレーションにおいては，さまざまなアルゴリズムが組み合わせて使用され，新しいアルゴリズムも次々に導入されるなど，専用ハードウェアの設計は困難である．しかし，可変構造型ハードウェアによる計算システム (リコンフィギャラブルシステム) は，問題を専用計算機のように直接ハードウェア処理することによって高い性能を発揮することが可能できる上に，回路をソフトウェア的に書き換える柔軟性を兼ね備えており，生物学分野への応用が有望であると考えられる.

本研究では，FPGA を搭載した生物学計算向けハードウェア ReCSiP(Reconfigurable Cell Simulation Platform)[9] を対象として，確率モデル生化学シミュレーションの高速実行環境を低コストで提供することを目的に，2 種類の SSA をそれぞれハードウェア処理する回路 (FRM-FPGA, NRM-FPGA) を実装し，その評価を行った.

FRM-FPGA は，First Reaction Method(FRM)[10] と呼ばれる SSA 中で最も基本的なアルゴリズムであり，パイプライン化された演算ユニットに連続的にデータを投入することにより，Xeon 2.80GHz で実行する場合の約 60 倍のスループットを実現した.

NRM-FPGA は，Next Reaction Method(NRM)[11] と呼ばれる高効率な SSA に関して，データ部分・演算部分をネットワーク状に接続することにより，回路面積の有効利用とスループットの線形向上を実現できることを示した．NRM はデータユニット 1 つだけの場合の予備評価を取った．反応 100 種類以上の大規模な生化学システムの場合，Xeon 2.80GHz でのソフトウェア実行の 0.5 倍程度の計算能力が得られた．NRM-FPGA に関しては，ReCSiP2 ボード搭載の中規模な FPGA(XC2VP70-5) に 10 個から 15 個のデータユニットを構成可能であると考えられるため，ソフトウェア実行と比較して 10 倍程度高スループットな計算環境を提供することができることがわかった.

本論文の第 2 章では，生化学システムのシミュレーション方法に関して，解析モデル，確率モデルおよびその発展的なアルゴリズムの概要とその性質，FPGA を用いた SSA の高速化手法の現状についてまとめ，第 3 章では ReCSiP を対象に，SSA を実行する FRM，NRM の 2 つの手法に関する回路の設計と実装について述べる．第 4 章では実装された 2 つの回路について，FRM-FPGA はシステム全体についての評価，NRM-FPGA はデータユニットを 1 つ持つ NRM-DU1 の回路について評価を述べ，第 5 章で結論と今後の展開について述べる.

第 2 章

背景

本章では，生化学反応を計算機を用いてシミュレーションを実行する手法についてまとめる．

また，近年研究が盛んである確率モデルシミュレーションアルゴリズムについて，提案されている複数のアルゴリズムの計算と精度の比較を行い，計算時間に関する問題点を述べる．さらに，ハードウェアを用いた高速な確率モデルシミュレータについて性能比較を行い，FPGA をはじめとするハードウェアシミュレータの持つ問題点について述べる．

2.1 システム生物学

システム生物学とは，システムレベルでの生命の理解を目指すアプローチである [12]．生命現象は遺伝子や代謝のネットワーク，細胞内外のさまざまな構造と，細胞間のシグナル伝達などによって成り立っており，これらの複雑なネットワークと構造をシステムとして理解することがこれからの生物学の中心的な課題であると考えられる．

1953 年の Watson と Crick による DNA の 2 重らせん構造の発見 [13] 以来，分子生物学はタンパク質や核酸でできた分子機械としての生命の構造解明を試み，その結果としていくつもの遺伝子やタンパク質の働きや配列・構造が判明しているが，それらの働きに関しては個々の部品に関する，比較的単純な場合についてしかわかっていない．しかし，システムとしての生命現象の理解，その動的な挙動の解析や実験が必要であり，実験から得られる大量のデータを効率よく分析し，仮説とモデルを立て，それに基づいた実験を行い，その結果に基づいて再び仮説やモデルを検証するというアプローチが必要である．

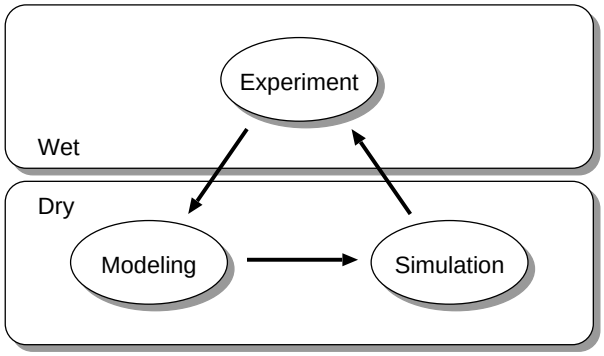


図 2.1: システム生物学のアプローチ

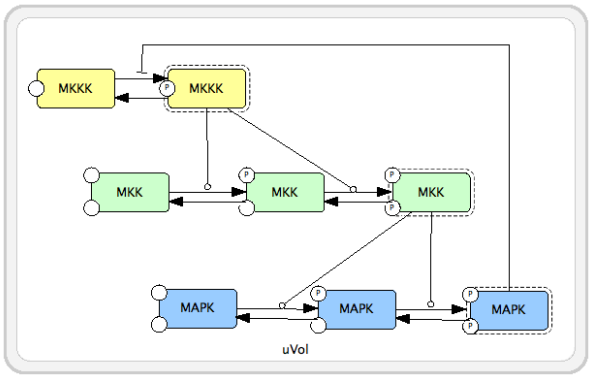


図 2.2: 反応経路の例

したがって，図 2.1 に示すように，システム生物学では従来からの wet な実験に加えて，数理モデリングと計算機シミュレーションを用いた，dry な実験が重要になる．モデリング・シミュレーションの分野は多岐にわたっており，微分方程式や確率モデルを用いた代謝回路の研究，ベイジアンネットワーク・ペトリネットなどを用いた遺伝子制御ネットワークの研究や，細胞膜などを力学的にシミュレーションすることによる細胞の機械的な構造の研究などが行われている [14]．いずれも計算コストの大きな問題であり，高性能な計算機が必要とされている．

2.2 解析モデルシミュレーション

前節で述べたように，現代の生命科学においては計算機による様々なシミュレーションが重要な役割を果たすが，計算に要する時間の長さは深刻な問題である．

2.2.1 生化学反応系モデル

解析モデル生化学シミュレーションは，いくつかの化学反応から構成されるシステムが，ある初期状態から出発した場合に，経時的な反応系中の物質濃度の変化を数値的に求めるものである．実験結果をもとにあるシステムのモデルを作り，システム内部のさまざまなパラメータを計算機上で変更することにより，システムの外乱に対する応答の特性などを調べることができる．実験とシミュレーションの反復により，より正確なモデルを構築することが可能になり，正確なモデルの構築はより深いレベルでの生命現象の理解を可能にする．

生化学反応系のモデルは，複数の化学反応のモデルの集合体であり，その土台となるのは

- 反応系に含まれる物質のリスト
- 反応系を構成する反応のリスト

の 2 つの要素で構成された反応経路である．図で表せばたとえば図 2.2 の例のようになり，物質を表すノードと，反応を表す矢印で構成された図になる．

システムの挙動を知るには反応経路に加えて，各物質の初期濃度と，各反応の反応機構モデル及びパラメータを知る必要があり，これらが与えられれば，反応による物質の増減を数値的に求

めることができる．それぞれの物質の消費・生成速度を表す反応速度式は次に述べるように，それぞれの反応機構の反応速度を数理的に表した常微分方程式を用いて記述されるのが一般的であり，この場合にはシステム内の反応をモデル化した常微分方程式すべてを連立微分方程式として数値的に解くことになる．

2.2.1.1 反応速度式

化学反応の反応速度 (基質 S の消費速度であり，生成物 P の生成速度) は一般に質量作用則によって表され，一次反応

$$S \xrightarrow{k_1} P \tag{2.1}$$

の反応速度 v は基質 S の濃度を $[S]$ として，

$$v = k_1[S] \tag{2.2}$$

のように記述することができる．二次以上の反応でも同様に，

$$S_1 + S_2 \xrightarrow{k_1} P \tag{2.3}$$

であればふたつの基質の濃度 $[S_1]$, $[S_2]$ を用いて，

$$v = k_1[S_1][S_2] \tag{2.4}$$

のように表現できることが知られている [15]．しかし，細胞内で起きる化学反応には酵素をはじめとして，反応速度を調節するさまざまな因子が複雑に絡み合っており，質量作用則だけですべての反応速度を記述するのは困難である．そのため，各種の反応機構をモデル化した近似式が用いられており，たとえば Michaelis と Menten によるもの [16] が酵素反応のモデルとして有名である．

Michaelis-Menten モデルでは，基質 S，酵素 E，酵素-基質複合体 ES，生成物 P の反応を式 2.5 のように表現する．k_1, k_2, k_3, k_4 は各反応の反応速度定数である．

$$S + E \underset{k_2}{\overset{k_1}{\rightleftarrows}} ES \xrightarrow{k_3} P + E \tag{2.5}$$

この式は，まず酵素 E と基質 S の複合体である ES が形成されて，そこで酵素の作用により生成物 P が生成される，というモデルを表している．これを，個々の反応速度式に展開すると，次の式 2.6〜式 2.8 のようになる．

$$v_1 = k_1[S][E] \tag{2.6}$$
$$v_2 = k_2[ES] \tag{2.7}$$
$$v_3 = k_3[ES] \tag{2.8}$$
$$\tag{2.9}$$

さらに各物質の濃度変化はこれを用いて

$$\frac{d[S]}{dt} = -v_1 + v_2 = -k_1[S][E] + k_2[ES] \tag{2.10}$$
$$\frac{d[P]}{dt} = v_3 - v_4 = k_3[ES] - k_4[P][E] \tag{2.11}$$
$$\frac{d[E]}{dt} = -v_1 + v_2 + v_3 = -k_1[S][E] + (k_2 + k_3)[ES] \tag{2.12}$$
$$\frac{d[ES]}{dt} = v_1 - v_2 - v_3 = k_1[S][E] - (k_2 + k_3)[ES] \tag{2.13}$$

のように表すことができる．

しかし，実験で観測できるのは $[S]$ や $[P]$ とその変化であり，この式に現れる k_1, k_2, k_3 といった値を求めることは困難である．そこで，実験結果から得られる最大反応速度 V_m のようなパラメータを用いて，単純な $S \rightarrow P$ の反応として表現した式を用いるのが実用的である．

式 2.5 において，定常状態とは $[S]$ や $[P]$ は変化しているが，$[E]$ と $[ES]$ は変化しない状態，すなわち

$$\frac{d[E]}{dt} = -d[ES]dt = 0 \tag{2.14}$$

が成立している状態である．この場合，式 2.13 の右辺第 1 項と第 2 項は等しくなるので，

$$k_1[S][E] = (k_2 + k_3)[ES] \tag{2.15}$$

$[ES]$ についての式に変形して

$$[ES] = \frac{k_1[S][E]}{k_2 + k_3} \tag{2.16}$$

を得る．ここで，Michaelis-Menten 定数 K_m を

$$K_m = \frac{k_2 + k_3}{k_1} \tag{2.17}$$

と定めると，式 2.16 は

$$[ES] = \frac{[S][E]}{K_m} \tag{2.18}$$

となる．ここで，酵素の総量を E_0 とすると

$$E_0 = [E] + [ES] \tag{2.19}$$

であるから，

$$[E] = E_0 - [ES] \tag{2.20}$$

これを式 2.18 に代入して

$$[ES] = \frac{[S](E_0 - [ES])}{K_m} \tag{2.21}$$

を得る．これを変形すると，

$$[ES]\frac{K_m}{[S]} = E_0 - [ES] \tag{2.22}$$

$$[ES]\left(1 + \frac{K_m}{[S]}\right) = E_0 \tag{2.23}$$

$$[ES] = E_0\frac{[S]}{[S] + K_m} \tag{2.24}$$

となり，これを用いると P の生成速度を

$$\frac{d[P]}{dt} = k_3[ES] = k_3E_0\frac{[S]}{[S] + K_m} \tag{2.25}$$

のように書くことができる．E_0 は実験で測定することのできないパラメータであるが，k_3E_0 は基質が飽和したときの最大反応速度 V_m と考えることができ，これは実験で測定することができる．これを用いればさらに

$$\frac{d[P]}{dt} = V_m\frac{[S]}{[S] + K_m} \tag{2.26}$$

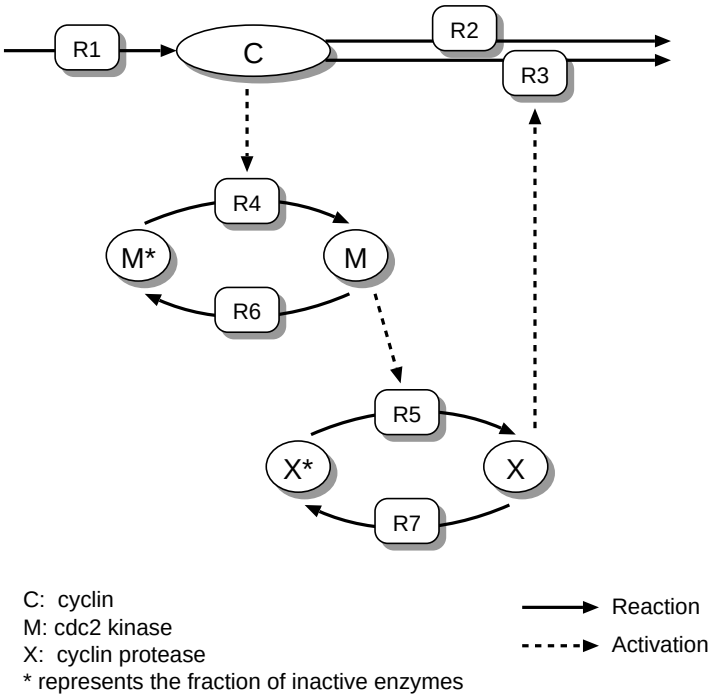


図 2.3: Minimal Mitotic Oscillator モデルの反応経路図

のようにして P の生成速度を得られる．

K_m は，式 2.26 で $[S] = K_m$ とすると

$$\frac{d[P]}{dt} = V_m \frac{K_m}{2K_m} = \frac{V_m}{2} \tag{2.27}$$

と表すことができ，$[S] = K_m$ のとき $d[P]/dt$ が最大反応速度の半分になることがわかる．したがって，K_m は反応速度が $V_m/2$ となるような $[S]$ である．

以上のことから，実験で求めることのできる値である K_m, V_m と基質濃度 $[S]$ を用いて，式 2.5 に示す Michaelis-Menten 型の反応速度式を

$$v = \frac{V_m[S]}{K_m + [S]} \tag{2.28}$$

と表すことができる．

Michaelis-Menten モデル以外にもさまざまな反応機構の反応速度を表すための式が存在し，反応経路中の反応ひとつひとつについて適切な反応機構・反応速度式とそのパラメータを与えることで，シミュレーションのための生化学モデルを構成することができる．

2.2.2 モデルの例: Minimal Mitotic Oscillator

Minimal mitotic oscillator モデル [17] は細胞周期中の cdc2 キナーゼの活性変化に関する実験結果から構築されたモデルである．このモデルの反応経路は cyclin, cdc2 kinase, cyclin protease の 3 つの物質で構成されるカスケードで，図 2.3 のようになっている．Cyclin は一定の速度で生成され，

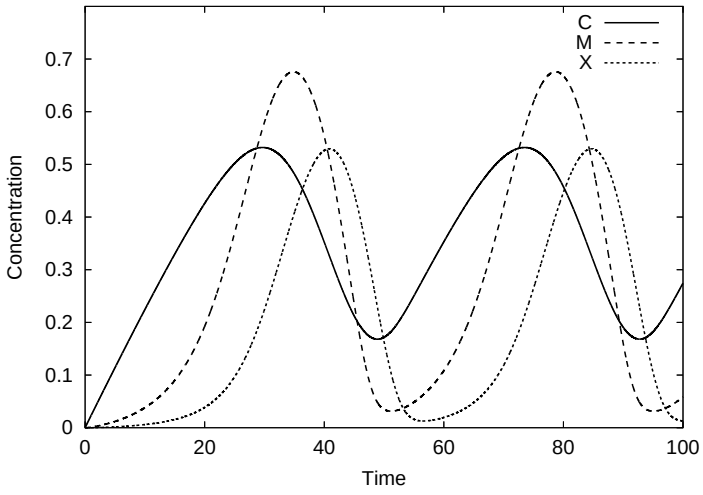


図 2.4: Minimal Mitotic Oscillator モデルの挙動

cyclin が cdc2 kinase を活性化し，cdc2 kinase が cyclin protease を活性化することによって cyclin が分解される．カスケードの終端からの負のフィードバックによって cyclin, cdc2 kinase, cyclin protease の濃度は図 2.4 のように振動を繰り返す.

図中の C (cyclin), M (cdc2 kinase), X (cyclin protease) の濃度をそれぞれ $[C]$, $[M]$, $[X]$ とすると，minimal mitotic oscillator モデルは，次の式 2.29〜2.35 で与えられる．

R_1: cyclin の生成

$$v_1 = V_i \quad (V_i = 0.023) \tag{2.29}$$

R_2: cyclin の分解

$$v_2 = K_d[C] \quad (K_d = 0.00333) \tag{2.30}$$

R_3: cyclin protease による cyclin の分解

$$v_3 = \frac{V_d[C][X]}{K_d + [C]} \quad (K_d = 0.00333,\ V_d = 0.1) \tag{2.31}$$

R_4: cdc2 kinase の活性化

$$v_4 = \frac{V_{m1}[C](1 - [M])}{(1 + K_1 - [M])(K_c + [C])} \quad (K_1 = 0.1,\ K_c = 0.3,\ V_{m1} = 0.5) \tag{2.32}$$

R_5: cyclin protease の活性化

$$v_5 = \frac{V_{m3}[M](1 - [X])}{K_3 + (1 - [X])} \quad (K_3 = 0.1,\ V_{m3} = 0.2) \tag{2.33}$$

R_6: cdc2 kinase の不活性化

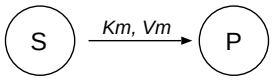
$$v_6 = \frac{V_2[M]}{K_2 + [M]} \quad (K_2 = 0.1,\ V_2 = 0.167) \tag{2.34}$$

R_7: cyclin protease の不活性化

$$v_7 = \frac{V_4[X]}{K_4 + [X]} \quad (K_4 = 0.1,\ V_4 = 0.1) \tag{2.35}$$

```
<model name="sample_model">
  <listOfCompartments>
    <compartment name="cell"/>
  </listOfCompartments>
  <listOfSpecies>
    <specie name="S" initialAmount="0.3" compartment="cell"/>
    <specie name="P" initialAmount="0"   compartment="cell"/>
  </listOfSpecies>
  <listOfReactions>
    <reaction name="Reaction1" reversible="false">
      <listOfReactants>
        <specieReference specie="S" />
      </listOfReactants>
      <listOfProducts>
        <specieReference specie="P" />
      </listOfProducts>
      <kineticLaw formula="uui(S,km,vm)">
        <listOfParameters>
          <parameter name="vm" value="0.01" />
          <parameter name="km" value="0.1" />
        </listOfParameters>
      </kineticLaw>
    </reaction>
  </listOfReactions>
</model>
```

a) SBML 記述



b) 反応経路

図 2.5: SBML によるモデル記述の例

2.2.3 モデル記述言語

前節で述べたようなモデルを計算機上で取り扱う際には，モデルを設計するためのツールやシミュレータ，分析ツールなどさまざまなソフトウェアを利用することになるが，作業を効率化するためにはこれらのツール間で使える共通のモデル記述言語が必須となる．このために，SBML (Systems Biology Markup Language)[18] や CellML[19] などの XML ベースの規格が定められ，多くのツールがこれらに対応している．XML を用いることで，互換性を維持しつつ言語の仕様を拡張したり，ツール独自の情報をファイル内に記述したりすることが容易になる．

図 2.5 に SBML によるモデル記述と，その表現する反応経路の図を示す．この図に示されるように，SBML によるモデル定義は，最も基本的な場合，

- モデルの名称 (`<model>`)
  - 区画のリスト (`<listOfCompartments>`)
    - 区画の名称 (`<compartment>`)
      これによって，細胞，細胞質，核などの区画を定義し，物質や反応をそれぞれの区画に配置することができる．区画間の包含関係も記述することができる．
  - 物質のリスト (`<listOfSpecies>`)
    - 物質の定義 (`<specie>`)
      物質の名前，初期濃度，配置される区画名などを記述する．
  - 反応のリスト (`<listOfReactions>`)
    - 反応の定義 (`<reaction>`)

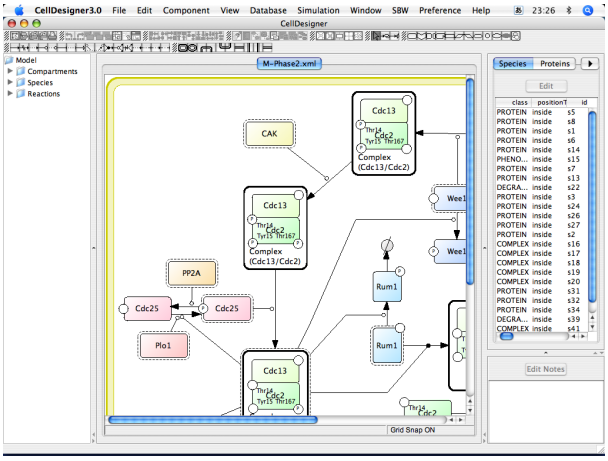


図 2.6: CellDesigner

- 反応物のリスト (<listOfReactants>)
  反応で消費される物質を定義する．
- 生成物のリスト (<listOfProducts>)
  反応で生成される物質を定義する．
- 反応速度式 (<KineticLaw>)
  反応速度式とそのパラメータの値 (<parameter>) を定義する．

などの項目を含んでおり，これを用いてシミュレーションを行うために必要な情報をさまざまなツールで共有することができる．ツール固有の情報は，各ブロック内に<annotation>ブロックを設け，そこにツール毎の namespace を用いて記述することができるようになっているほか，モデルを描画する際のレイアウト情報など，多くのツールが付加するようになった情報の記述方法を標準として策定していくなど，規格の更新も行われている．

2.2.4 設計ツール

モデルを記述したり読んだりする際に，XML を直接手作業で記述するのでは作業効率が悪く，大きなモデルを構築するのは困難であるため，CellDesigner[20][21], JDesigner[22], PathwayLab[23] などの GUI を用いたツールが多く開発されている．たとえば，図 2.6 は CellDesigner のスクリーンショットであり，ドローイングツールのように画面上に物質や反応を描いていくことで，視覚的に SBML のモデルを作成し，物質の初期濃度や反応速度式などを設定することができる．

これらのツールの多くは，シミュレータや分析ツールなどと連携しており，モデルの作成から分析までを一貫して行えるようになっている．また，最近では大きな反応経路をモデル化する際に，モデルの妥当性を示すために反応系中の反応や物質について出典を記述したり，そこから文献データベースへ接続したりといった機能が活用されるようになってきており，設計ツールは情報を集約するツールとしての役割をも果たすようになりつつある．

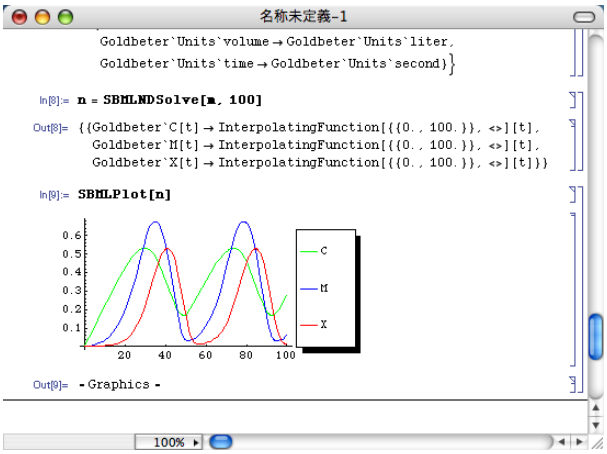


図 2.7: MathSBML

2.2.5 シミュレータ

Jarnac[24], Copasi[25], MathSBML[26] などのシミュレータも数多く開発されている．これらのシミュレータではいずれも結果を視覚化したり，パラメータを最適化するための独自の特徴的なインタフェイスを持っており，さまざまなシミュレータをうまく使い分けることで目的とする結果を得ることができる．また，設計ツール，シミュレータ，分析ツールなどのソフトウェア間を接続するためのソフトウェア基盤として，SBW (Systems Biology Workbench)[27] が開発されている．

図 2.7 は MathSBML のスクリーンショットである．MathSBML は Wolfram Research 社の汎用科学技術計算ソフトウェアである Mathematica 上で動作するシミュレータで，SBML ファイルを読み込んで連立微分方程式に変換し，シミュレーションをする機能を提供する．

2.2.6 データベース

モデル記述方式の標準化や，関連するツールの整備の結果，反応経路をデータベース化して再利用しようという機運が高まり，KEGG[28][29] や BioModels.net[30] などのデータベースが構築されている．KEGG は独自のモデル記述方式をとっているが，SBML へのトランスレータ [31] も開発されており，SBML 対応の処理系で KEGG のモデルを利用することが可能である．データベースとともに，論文誌に掲載される論文のモデルを SBML を用いて記述し，電子ジャーナルで本文とともに配布されるケースも増えてきている．また，SBML や CellML などは記述の自由度が高いため，モデルをデータベースに登録するにあたって，再利用性を確保するためにモデルが最低限含むべき情報のガイドラインの策定 [32] も行われている．

KEGG や BioModels.net のような，一般的なデータベースのほかにも線虫に関するデータやモデルを集めた Wormbase[33] や，大腸菌に着目した EcoCyc[34] のように，特定のモデル生物に関する知識の集積をはかるプロジェクトも推進されている．

表 2.1: 解析モデル・確率モデルの差異

	解析モデル	確率モデル
分子の計算単位	濃度	数
計算単位	タイムステップ	反応発生
計算単位あたりの計算量	$O(N)$	$O(N)$ または $O(\log(N))$
計算時間	タイムステップ	反応発生

† N = 化学反応システムで発生する反応の種類

2.3 確率モデルシミュレーション

確率モデルシミュレーションアルゴリズム (SSA : Stochastic Simulation Algorithm) は，1976 年に Gillespie が開発したモンテカルロ法を用いた生化学シミュレーションの一手法である．

2.3.1 確率モデルシミュレーションの概要

化学反応の確率モデルシミュレーションは，時間 t で複数種類の分子が一様に分布している空間 (体積 V) について，

1. 何らかの反応が次に発生する時間 (τ)
2. 発生する反応の種類 (μ)

を計算し，分子の数を更新するプロセスを繰り返す手法である．化学反応システム内に存在する分子の数の時間変化を計算することが，確率モデルシミュレーションであると言える．

τ と μ の計算方法は，現在まで複数の方法が提案されている (詳細は 2.3.4).

τ と μ を求める際に乱数を利用するため，使用する乱数列が異なる場合，乱数種以外の初期パラメータが同一でも，分子数の時間変化は異なった軌道を描く．そのため，シミュレーション結果は，数千回以上の実行結果から統計的に解析される必要がある．一般的に τ と μ の計算式は，解析モデルのシミュレーションアルゴリズムと比較して単純であるが，計算回数が膨大になるため，確率モデルシミュレーションには長い計算時間が必要であることが知られている．

2.3.2 生化学反応システムの記述法

SSA は，以下の 3 項目について定義されたものを生化学反応システムと見なし，シミュレーションを実行することができる．反応システムの時間を t で表し，シミュレーション開始時間を t_0 としている．

1. **分子の種類と初期値**
   m 種類の分子 $S_1, \cdots, S_m$ と，初期状態 $\overrightarrow{X(t_0)} = \{X_1(t_0), \cdots, X_m(t_0)\}$ ($X_i(t)$: 整数)
2. **反応の種類と形式**
   (a) n 種類の反応 $R_1, \cdots, R_n$ について，確率反応速度定数 $\overrightarrow{k} = \{k_1, \cdots, k_n\}$ (k_j : 実数)
   (b) 状態更新ベクトル $\overrightarrow{\nu}_j$

- 反応 R_j 発生の際の，分子数の増減を示すリスト
  - R_j の発生要因となる分子 (反応物:Reactants) のリスト
  - R_j によって生成される分子 (生成物:Products) のリスト

例)

$$R_x : \quad X_a + X_b \xrightarrow{k_x} X_c + X_d \tag{2.36}$$

- R_x は反応物 X_a と X_b の分子によって発生する反応である
- 反応 R_x は，確率反応速度定数 k_x で発生する
  - k_j と生化学システムの温度や体積等から求められる値 c_j を反応のパラメータとして用いることが多い
- R_x の発生の結果，分子 X_a，X_b がそれぞれ 1 分子減る
- 同時に，分子 X_c，X_d がそれぞれ 1 分子増える

SSA では，分子数と確率反応速度定数から，propencity function が計算される．propencity の計算は，近似を用いない全ての SSA で使用されるパラメータである．前述の反応 R_x の propencity a_j は，以下の式 2.37 で求められる．

$$a_x = X_a \times X_b \times k_x \tag{2.37}$$

また，式 2.38 のような反応 R_y の場合，propencity a_y は式 2.39 で求められる．

$$R_y : \quad X_a + X_a \xrightarrow{k_y} X_e \tag{2.38}$$

$$a_y = \frac{X_a \times (X_a - 1)}{2} \times k_y \tag{2.39}$$

式 2.40 のような反応 R_z の場合，propencity a_z は式 2.41 で求められる．

$$R_z : \quad X_a \xrightarrow{k_z} X_f \tag{2.40}$$

$$a_z = X_a \times k_z \tag{2.41}$$

このことから，反応 R_j の propencity とは，“その反応の反応物の組み合わせ (combination) に確率反応速度定数を乗算したもの” と言うことができる．反応 R_z のように反応物が 1 個である反応を 1 次反応，反応 R_x，R_y のように反応物が 2 個である反応を 2 次反応と呼ぶ．

2.3.3 ベンチマーク生化学システム

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評価や，FPGA に実装した回路の性能を計測するためのベンチマークとして，以下に示す生化学システムを使用する．

2.3.3.1 Heat Shock Response Model(HSR)

HSR は，温度が上がったときに，バクテリア *E.Coli* がどんな挙動をするかを記述したモデルである [35]．構成タンパク質が変性 (展開) するくらい温度が高くなったとき，*E.Coli* はいくつかの現象を示す．温度が高くなると，heat shock sigma factor σ_{32} が急速に生成される．自由 σ_{32} 分子は

表 2.2: Lotka System

R_1	$X_1 + X_2$	$\xrightarrow{c_1}$	$X_2 + X_2$
R_2	$X_2 + X_3$	$\xrightarrow{c_2}$	$X_3 + X_3$
R_3	X_2	$\xrightarrow{c_3}$	X_4
R_4	X_3	$\xrightarrow{c_4}$	X_4

RNA ポリメラーゼ (RNAP) と結合して，複合 σ_{32} となる．RNAP は，監視酵素 (chaperon enzyme) をコードしてる遺伝子の転写を始める．この監視酵素は変性タンパク質を再結合するか，集合しないように分解する反応を発生させる．解析モデルと確率モデルについてそれぞれ詳細に解説した論文が存在している．

2.3.3.2 Linear Chain System(LCS)

LCS は $M+1$ 種類の分子が連鎖的に M 種類の反応を発生するモデルである [35]．基本的に使用されるパラメータは $a_j(x) = cX_j, c = 1$ が使用され，初期状態は $X(t) = \{10000, 0, \cdots, 0\}$ が与えられる．依存率は $\mathbb{D} = 1$ である．

$$S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_{M+1} \tag{2.42}$$

2.3.3.3 A Totally Independent System(TIS)

TIS は M 種類の分子の減衰反応である [35]．初期状態は $X(t) = \{1000, 1000, \cdots, 1000\}$ である以外は，LCS と同じパラメータが与えられる．最も疎なシステムであり，依存率 $\mathbb{D} = 0$ である．

$$S_j \rightarrow \emptyset, \qquad j = 1, \cdots, M \tag{2.43}$$

2.3.3.4 Lotka System(LTS)

LTS(Lotka system) は，Gillespie が SSA の提案の際に，アルゴリズムの適用例として使用された [6]．Lotka sytem は 4 種類の反応と 4 種類の分子で構成される小規模な反応システムである表 2.2 に反応のリストを示す．

Lotka system は，生態系のシミュレーションの基本的なモデルとして例に用いられる問題である．表 2.3 のように意味付けると理解しやすい．

図 2.8，図 2.9 に Lotka system を用いてシミュレーションを実行した際の，分子 X_2，分子 X_3 の時間変化を図示する．20 万回の反応発生までの分子数の時間変化を描画し，シミュレーションアルゴリズムには Direct Method(2.3.5) を用いている．図 2.8，図 2.9 を生成したシミュレーションでは，以下のパラメータを設定している．2 回の実行結果を求めた際の差異は乱数種のみである．

- 初期分子数 $\overrightarrow{X}(t_0) = \{100000, 1000, 1000, 0\}$
- 確率反応速度定数 $\overrightarrow{k} = \{0.0002, 0.01, 10.0, 10.0\}$
- 反応パラメータ $\overrightarrow{c} = \overrightarrow{k}$

表 2.3: Lotka System の例

分子	X_1	草 (食料：入力)
	X_2	ウサギ (草食動物)
	X_3	オオカミ (肉食動物)
	X_4	動物の死骸 (廃棄物：出力)
反応	R_1	ウサギが草を食べて繁殖する
	R_2	オオカミがウサギを捕食して繁殖する
	R_3	ウサギが自然死する
	R_4	オオカミが自然死する

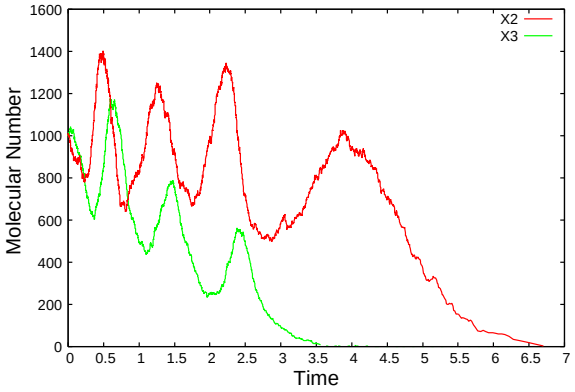


図 2.8: Lotka system の実行結果 (乱数種 A)

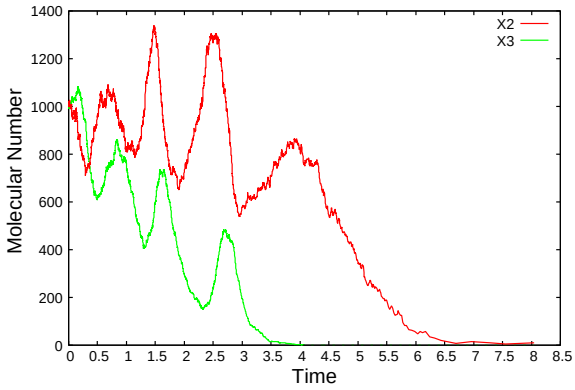


図 2.9: Lotka system の実行結果 (乱数種 B)

- 状態更新ベクトル

$$\nu_1 = \{-1, 1, 0, 0\} \tag{2.44}$$
$$\nu_2 = \{0, -1, 1, 0\} \tag{2.45}$$
$$\nu_3 = \{0, -1, 0, 1\} \tag{2.46}$$
$$\nu_4 = \{0, 0, -1, 1\} \tag{2.47}$$

LTS は，$\overrightarrow{k} = \{0.002, 0.01, 10.0, 10.0\}$，$\overrightarrow{X(t_0)} = \{100000, 1000, 1000, 0\}$ と定義し，反応 X_1 が減少しない定数として定義すると，X_2，X_3 が振動する生化学システムである．すなわち，反応 R_1 は以下の式で置き換えられる．上部に傍線のついた分子 $\bar{X}$ は，反応によって分子数が変化しない分子を示している．この場合の 20 万反応の実行結果を，図 2.10，図 2.11 に示す．2 回の実行結果の違いは乱数種のみである．

$$R_1 : \bar{X}_1 + X_2 \xrightarrow{c_1} X_2 + X_2$$

2.3.4 First Reaction Method

First Reaction Method[10] は，Gillespie が 1976 年に提案した最も初期の SSA である．
FRM では，以下のようにシミュレーションが進展する．

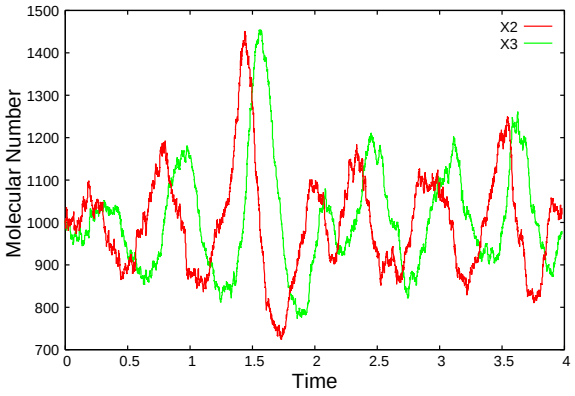


図 2.10: Lotka system の実行結果 (乱数種 A)

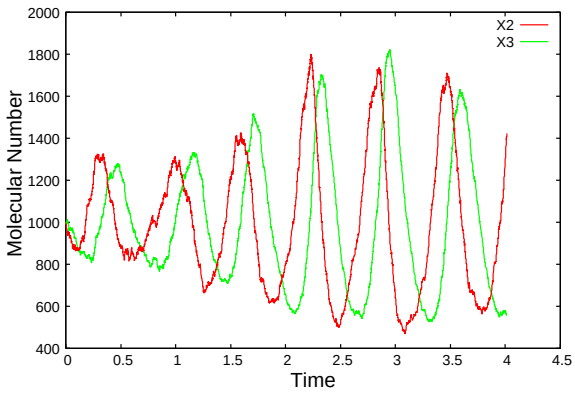


図 2.11: Lotka system の実行結果 (乱数種 B)

1. 分子の初期値 $\overrightarrow{X}$，乱数種 r_0，時間 $t = 0$ を設定する
2. 全ての反応 R_j $(j = 1, \cdots, N)$ について，式 2.48 より “予測発生時間 τ_j” を計算する

$$\tau_j = \frac{1}{a_j} \ln\left(\frac{1}{r_j}\right) \tag{2.48}$$

   - a_j は反応 R_j の propencity
   - r_j は $[0, 1)$ の範囲の実数型一様乱数

3. 発生反応 R_μ として，式 2.48 で得られた τ_j のうち，最小値を持つ反応を選択する

$$\text{発生反応 } R_\mu = \text{反応番号}\left(\min\left\{\overrightarrow{\tau}\right\}\right) \tag{2.49}$$

$$\tau_\mu = \min\left\{\overrightarrow{\tau}\right\} \tag{2.50}$$

4. システム時間に，反応が発生するまでの時間 τ_μ を加算する

$$t \leftarrow t + \tau_\mu \tag{2.51}$$

5. 分子の状態を更新する

$$\overrightarrow{X} \leftarrow \overrightarrow{X} + \overrightarrow{\nu_\mu} \tag{2.52}$$

6. 2 に戻る

FRM は，前述の 2. から 6. までの計算を反応 1 サイクルとして，以下のようなシミュレーション終了条件に合致するまでサイクルを繰り返す.

- 所定の回数反応が起こる
- システム時間が規定の時間に達する
- 特定の分子が設定された数に達する

N 種類の反応が発生する生化学システムについて，FRM の計算量は以下の通りである.

- 1 サイクルの計算に乱数を N 個消費
- 毎サイクル，全種類の反応について τ_j を計算
- 計算量は $O(N)$

2.3.5 Direct Method

Direct Method (DM)[6] は，FRM と数学的に等価なアルゴリズムであるが，計算機を用いた場合により有効な手法として知られている．

DM では，以下のようにシミュレーションが進展する．

1. 分子の初期値 $\overrightarrow{X}$，乱数種 r_0，時間 $t = 0$ を設定する
2. 全ての反応 R_j $(j = 1, \cdots, N)$ について，propencity a_j を計算する
3. propencity の総和 a_0 を計算する (式 2.53)

$$a_0 = \sum_{j=1}^{N} a_j \tag{2.53}$$

4. 2 個の範囲 $[0, 1)$ の実数型一様乱数 r_1，r_2 を用いて，発生反応 R_μ および反応発生時間 τ を計算する

$$\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right) \tag{2.54}$$

$$\sum_{j=1}^{\mu-1} a_j < r_2 a_0 \leq \sum_{j=1}^{\mu} a_j \tag{2.55}$$

5. システム時間に，反応が発生するまでの時間 τ_μ を加算する

$$t \leftarrow t + \tau_\mu \tag{2.56}$$

6. 分子の状態を更新する

$$\overrightarrow{X} \leftarrow \overrightarrow{X} + \overrightarrow{\nu_\mu} \tag{2.57}$$

7. 2 に戻る

N 種類の反応が発生する生化学システムについて，DM の計算量は以下の通りである．

- 1 サイクルの計算に乱数を 2 個消費
- 毎サイクル，全種類の反応について propencity を計算
- 計算量は $O(N)$

2.3.6 Next Reaction Method

Gillespie が開発した 2 種類のオリジナル SSA[10][6] の計算時間は，対象システム内で定義された反応数 N に線形に比例する (計算量 $O(N)$)．Next Reaction Method (NRM) は，Gibson と Bruck による大規模な生化学システムに対応する SSA である [11]．NRM は，Gillespie の First Reaction Method(FRM)[10] に Indexed Priority Queue(IPQ)，Dependency Graph(DG) と呼ばれる 2 種類のデータ構造を導入し，FRM と統計的に等価でありながら，計算量を $O(\log(N))$ に向上したアルゴリズムである．

NRM は，すべての反応の発生予測時間のうち最小値である反応 (最も直近に起こる反応) を，発生反応 R_μ として選択する．最小値の探索を IPQ を使用して効率化し，τ_j の修正回数を DG によって最小限に抑えることで，高速な計算が可能になっている．また，計算量が $O(\log(N))$ であることから，大規模な生化学システムのシミュレーションが可能であると考えられている．

2.3.6.1 Indexed Priority Queue

IPQ は，2 分木を構成するメモリと 1 次元配列のメモリの組で作られるデータ構造である (図 2.12)．2 分木は，反応番号とその発生予測時間の組 (j, τ_j) をノードとし，親ノードの τ_j は子ノードの τ_j よりも常に小さいという性質を満たす．1 次元配列のメモリは，ノード R_j の 2 分木上の位置を示すポインタテーブルである．

2 分木がその性質を満たす状態であれば，根ノードには全ノード中で最小の τ_j を持つノードが格納されている．また，根ノードの反応番号 j から，発生した反応を決定することができる．

また，特定の反応の発生予測時間は，ポインタテーブルから目的の反応の位置を調べ，2 分木から読み出すことで知ることができる．

IPQ が持つべき機能は，以下の 3 つである．

ADD 2 分木の末端にノードを追加する．木の親方向のノードと τ_j を比較，交換を行い，木の性質を保つ

UPDATE 2 分木の任意の位置のノードの τ_j を更新し，親方向あるいは子方向のノードと τ_j を比較，交換を行い，木の性質を保つ

READ 2 分木の任意の位置のノードを読み出す

ノードの交換の際，同時にポインタテーブルの値も交換する．2 分木のノード数は生化学システムの反応数 N と等しく，大規模システムではノードの交換時間がシミュレーション時間の中で支配的になるため，計算量が $O(\log(N))$ となる．

2.3.6.2 Dependency Graph

ある反応が発生して分子数が変化した場合，他の反応の発生予測時間が変化する場合がある．DG は，R_j が起こった時に，τ_j が修正される反応を列挙したリストである．例として図 2.13 のような反応 $R_1 - R_4$ が発生する生化学システムを定義する．反応 R_2 が発生した場合，修正が必要な反応は，R_2 で増減する分子が Products に含まれる反応である．R_2 発生により，分子 B，C が増加し，D が減少する．分子 B，C，D を Products に持つ反応は R_1，R_3 なので，τ_1, τ_3 を修正しなければならない．そのため，DG の R_2 のエントリには R_1，R_3 が記述される．τ_j の修正は，式 2.58 に従う．

$$\tau_{j,\text{new}} = \frac{a_{j,\text{old}}}{a_{j,\text{new}}}(\tau_{j,\text{old}} - \tau_\mu) + \tau_\mu \tag{2.58}$$

Gibson らによると，51 種類の分子と 71 種類の反応で構成される λ ファージモデルの場合，1 回の反応発生による τ_j 平均修正回数は $D = 4.2$ 回である [11]．Cao らは，反応が数百種類程度の生化学システムでは，反応数 N だけでなく，平均 τ_j 修正回数 D も計算時間に影響を与えることを明らかにしている [35].

2.3.6.3 Next Reaction Method — アルゴリズム

NRM は，分子数の時間変化を以下の手順で計算する．

STEP1 初期値を設定する

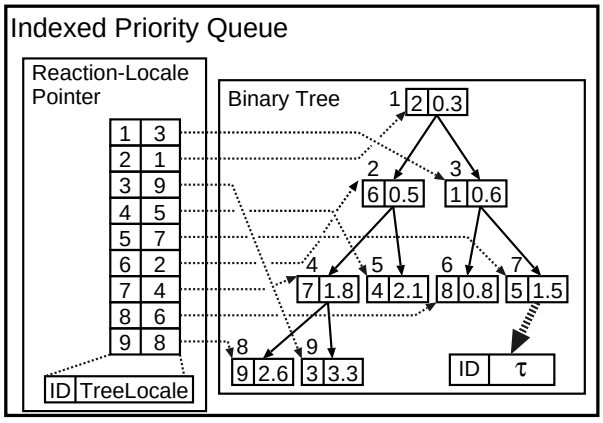


図 2.12: IPQ の構成

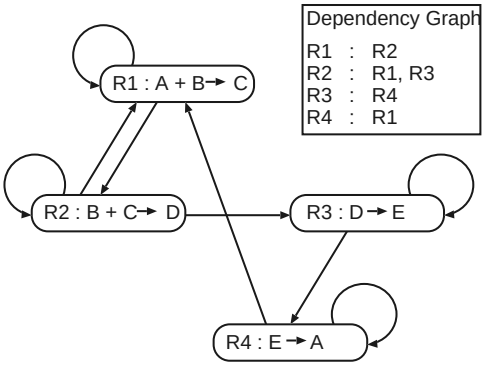


図 2.13: DG の構成

- 初期分子状態ベクトル $\overrightarrow{X(t_0)}$
- DG の生成
- システム時間 $t \leftarrow 0$

STEP2 全反応について，式 2.54 から τ_j を計算し，IPQ に格納 (ADD) する

STEP3 2 分木の根ノードを，発生反応 R_μ として選択する

- 分子状態を更新する ($\overrightarrow{X(\tau_\mu)} \leftarrow \overrightarrow{X(t)} + \overrightarrow{\nu_\mu}$)
- システム時間を根ノードの値で更新する ($t \leftarrow \tau_\mu$)

STEP4 R_μ の次の発生予測時間 τ_μ を式 2.48 の結果と現在時間 t を加算して求め，ノードを更新 (UPDATE) する

STEP5 DG から R_μ 発生時に修正する τ_j のリストを得る．リストにある各反応の τ_j を 2 分木から読み出し，式 2.58 によって τ_j を更新する

STEP6 STEP3 に戻る

N 種類の反応が発生する生化学システムについて，NRM の計算量は以下の通りである．

- 1 サイクルの計算に乱数を 1 個消費
- 毎サイクル，修正が必要な反応について propencity を計算
- 計算量は $O(\log(N))$

2.3.7 Optimized Direct Method

Cao らは FRM，DM，NRM の計算コストについて詳細な検討を行い，DM を改良した Optimized Direct Method (ODM) を提案している [35].

2.3.7.1 DM と NRM の計算時間の比較

計算時間について，DM と NRM の比較する．ソフトウェアによる実行について FRM と DM を比較した場合，1 サイクルあたりの乱数生成数，および τ_j の計算回数等の点から，DM の方が効率的であることは明かであるため，DM と NRM の比較を行っている．2 つの方法の主な違いを表 2.4 に示す.

表 2.4: DM と NRM の処理の比較

	DM			
	主な処理	コスト	計算量	
(1)	乱数 r_1，r_2	$2C_{\text{rand}}$	—	2 回
(2)	M 個の a_j	C_p	$O(M)$	$2MC_{\text{MULT}}$
(3)	a_0	C_{a_0}	$O(M)$	MC_{ADD}
(4)	τ(式 2.54)	C_τ	—	1 回
(5)	μ(式 2.55)	C_s	$O(M)$	$\mathbb{S}\left(C_{\text{ADD}} + C_{\text{COMP}}\right)$
(6)	その他	C_{rest}	—	—
	NRM			
	主な処理	コスト	計算量	
(1)	乱数 r	C_{rand}	—	1 回
(2)	a_α	$C_{p'} + C_g$	$O(\mathbb{D})$	$2\mathbb{D}C_{\text{MULT}}$
(3)	τ_μ(式 2.54)	C_τ	—	1 回
(4)	τ_α	C_α	$O(\mathbb{D})$	$(\mathbb{D}-1)\left(C_{\text{MULT}} + C_{\text{DIV}}\right)$
(5)	木の再構成	C_{heap}	$O[\mathbb{D}\log_2(M)]$	$\mathbb{U}{C_H}^\nabla$
(6)	その他	C_{rest}	—	—

$^\nabla$ 1 回のノード交換のコストを C_H とする

$\mathbb{D}$ は，NRM において 1 サイクルあたりの平均 τ_j 更新回数である．生化学システムが大規模かつ疎である場合 (ある反応が他の反応に影響が無い場合)，NRM は有利である．どのくらい疎であると有利であるかは Gibson らは明確に示していない.

Cao らは HSR，LCS，TIS の 3 つのサンプルモデルで比較を行っている．(2.3.3 参照)．実行環境は Pentium4 1.4Ghz で，OS には Linux を用いている．その結果を表 2.5 に示す．LCS，TIS では $M = 600$ とし，$T = 30$ まで計算を行っている．

表 2.5: DM と NRM における CPU コスト

HSR	DM	C_{a_0}	C_s	C_p	$2C_{\text{rand}}$	C_τ	C_{updateX}	C_{rest}	Total
	CPU cost (s)	15.48	14.98	11.71	4.09	6.45	7.36	9.30	86.8
	Persentage[%]	22.24	21.52	16.82	5.88	9.27	10.57	13.36	
	NRM	C_{heap}	C_α	$C_{p'} + C_g$	C_{rand}	C_τ	C_{updateX}	C_{rest}	Total
	CPU cost (s)	99.16	23.34	19.34	1.74	6.36	7.89	6.42	170.4
	Persentage[%]	60.00	14.12	11.70	1.05	3.85	4.77	3.89	
LCS	DM	C_{a_0}	C_s	C_p	$2C_{\text{rand}}$	C_τ	C_{updateX}	C_{rest}	Total
	CPU cost (s)	0.66	0.05	0.56	0.03	0.03	0.65	0.05	2.13
	Persentage[%]	32.54	2.46	27.50	1.45	1.63	31.86	2.56	
	NRM	C_{heap}	C_α	$C_{p'} + C_g$	C_{rand}	C_τ	C_{updateX}	C_{rest}	Total
	CPU cost (s)	0.17	0.03	0.03	0.02	0.03	0.81	0.05	1.07
	Persentage[%]	14.8	2.61	2.61	1.74	2.61	70.43	4.35	
TIS	DM	C_{a_0}	C_s	C_p	$2C_{\text{rand}}$	C_τ	C_{updateX}	C_{rest}	Total
	CPU cost (s)	1.26	1.27	1.07	0.03	0.03	1.67	0.06	5.39
	Persentage[%]	23.38	23.56	19.85	0.56	0.56	30.98	1.11	
	NRM	C_{heap}	C_α	$C_{p'} + C_g$	C_{rand}	C_τ	C_{updateX}	C_{rest}	Total
	CPU cost (s)	0.27	0	0.12	0.03	0.07	1.64	0.19	2.63
	Persentage[%]	10.70	0	4.74	1.19	2.77	64.82	7.51	

表 2.5 から以下のことがわかる．HSR では，NRM のツリー再構築のコストが大きい．実際に用いられる生化学システムにはこのような依存度 D が大きいものが多い．LCS は，NRM の計算時間が DM の半分になっている．これはツリー再構築時間が小さいことが原因である．TIS は最も疎なシステムであり，ツリーの更新がサイクルごとに 1 回だけのため，NRM の方が高速になっている．

2.3.7.2 DM と NRM の計算コストの解析

DM と NRM の計算コストを詳細に解析するために，$\mathbb{S}$，$\mathbb{D}$，$\mathbb{U}$ の 3 つの値を定義する．

Average Search Depth $\mathbb{S}$

DM における式 2.55 の平均探索深度．処理内容は ADD と COMPARE．生化学システムに依存し，式 2.59 から $\mathbb{S}$ の値を得る．

$$\mathbb{S} = \frac{\sum_j^M j k_j}{\sum_j^M k_j} \tag{2.59}$$

ここで，k_j はシミュレーション中に反応 R_j が起こる回数である．R_j ごとに探索回数は固定 (j 回) である．全反応が同程度起こりやすいならば，$\mathbb{S} = M/2$ となる．実際の生化学システムでは，発生反応の回数に大きな偏りがあるので $\mathbb{S} \neq M/2$ になると考えられる．

Averate Weighted Degree $\mathbb{D}$:

NRM の DG の平均次数 (頂点ごとの辺の数)．生化学システムに依存する値で，式 2.60 で $\mathbb{D}$

の値が得られる．

$$\mathbb{D} = \frac{\sum_j^M d_j k_j}{\sum_j^M k_j} \tag{2.60}$$

式 2.60 において，d_j は反応 R_j の次数を示す．$\mathbb{D}$ は反応 R_j が発生した時に，予測発生時間を更新しなければならない反応の数の平均と言える．

Update Depth $\mathbb{U}$:

1 回のツリー再構築に必要な命令数 $\mathbb{U}$.

$$\mathbb{D} \leq \mathbb{U} \leq \mathbb{D}\log_2(M) \tag{2.61}$$

式 2.61 において，1 回の再構築におけるツリーノードの移動はたかだか $\log_2(M)$ 回である．生化学システムは各種パラメータの桁がマルチスケールであることが多いため，反応ごとに発生回数に偏りがあり，特定の反応が多数回発生することが多い．多数回発生する反応は，それらはツリーの根周辺をうろうろするため，$\mathbb{U}$ は $\mathbb{D}$ に近い値を取る (表 2.6).

表 2.6: 各生化学システムの計算コスト

	M	$\mathbb{S}$	$\mathbb{D}$	$\mathbb{U}$
HSR	61	26.3	8.79	10.33
LCS	600	16	2	2.001
TIS	600	300	1	5.15

計算コストの比較

Cao らは，以下の理由から，DM が NRM よりも計算コストが小さくなることを予測した.

1. まだ生化学システムが NRM で利益が出るほど大きくない
2. DM は大規模システム向けに最適化されていない
3. NRM 提案時に CPU コストの計算が完全でない

DM と NRM の計算コストの差は以下の式で表される (参考:表 2.4).

$$C_{\text{DM}} - C_{\text{NRM}} = C_{\text{rand}} + C_1 + C_2 - C_{\text{heap}} \tag{2.62}$$

ここで，$C_1 = C_{a_0} + C_s - C_\alpha$，$C_2 = C_p + C_{p'} - C_g$ である．$C_{\text{DM}} - C_{\text{NRM}}$ が負の値になることは，以下の点から予測される．

1. MULT や DIV は ADD や COMPARE よりも計算時間が多い
2. $C_p >> C_{p'}$ だが，$C_{p'}$ には DG の読み出しコスト C_g(オーダー $O(\mathbb{D})$) が必要
3. 実際の生化学システムでは，$C_s < C_\alpha$(DM の μ 探索は NRM の τ_α 修正よりも小さい)
4. ツリーデータにアクセスするコストが大きい (HSR の C_{heap} を参照)

2.3.7.3 Optimized direct method — アルゴリズム

DM のボトルネックは $O(M)$ の C_{a_0}，C_p と，$O((S))$ の C_s であることがわかっている．計算コストを削減するために，DM に対し 2 つの最適化を適用した Optimized Direct Method(ODM) が開発された．

表 2.7: ODM の CPU コスト (HSR)

ODM	C_{a_0}	C_s	C_p	$2C_{\text{rand}}$	C_τ	C_{updateX}	C_{rest}
CPU cost(s)	15.27	4.21	9.27	6.23	6.90	7.97	12.13
Persentage[%]	24.64	6.79	14.96	10.05	11.13	12.86	19.57

大規模システムはマルチスケールで，いくつかの反応だけが頻繁に起こることが多いため，第 1 の最適化として，C_s についてコスト削減 (表 2.7 参照) を行っている．HSR は，6 種類の反応が全体の 90%，12 種類の反応が全体の 99% を占める．ODM では，DM を式 2.55 から，$\mathbb{S}$ を小さくするような実装に最適化している．これは，生化学システムの反応の順番 (インデックス) を変えることで実現できる．順番は，起こりやすい反応順に並べて計算する (このときの $\mathbb{S}$ を $\mathbb{S}^*$ とする．$S^* \ll M/2$).

$$k_i > k_j \qquad \text{for} \qquad i < j \tag{2.63}$$

数回のシミュレーションを DM で実行し，適切な順番で反応をソートする．反応発生回数順に並び替えることで，HSR で $S^* = 3.37$，実行時間は 76.5 秒に減少した (11.87% の効率化を達成)．並びを逆順 (起こりにくい順) にすると，$S^* = 58.6$，実行時間は 102.2 秒になった (最大 25.12% の効率化が図れた).

第 2 の最適化として，C_{a_0}，C_p について，$D \ll M$ の場合に効果が出るように，NRM のアイデアを DM に適用する．propencity a_i は毎サイクル全てを再計算する必要はない．そのため，各サイクルごとに必要最低限の a_j を再計算することにする (DG の類似のものを導入する)．この計算の削減を適用すると，計算コストが $C_p \approx C_{p'} + C_g$ になる．毎サイクル a_0 を計算するのではなく，a_0 から古い $a_{i,\text{old}}$ を減算して，新しい $a_{i,\text{new}}$ を加算する．このことにより，計算量は $O(M)$ から $O(\mathbb{D})$ になる．LCS1 回のシミュレーション実行時間は平均 0.86 秒 (DM2.13 秒：NRM1.07 秒，詳細は表 2.8)，TIS1 回のシミュレーション実行時間が平均 3.72 秒 (DM は 5.39 秒：NRM2.63 秒，詳細は表 2.9) に改善された．

表 2.8: ODM の CPU コスト (LCS)

ODM	$C_{a_0} + C_p$	C_s	$2C_{\text{rand}}$	C_τ	C_{updateX}	C_{rest}
CPU cost(s)	0.02	0.05	0.03	0.02	0.67	0.06
Persentage[%]	2.35	5.88	3.53	2.35	78.82	7.06

$D \ll M$ かつ $S^* \approx M/2$ のときは ODM は NRM よりも計算コストが高いが，実際の生化学システムはマルチスケールのため $S^* \ll M/2$ の場合が多く，ODM は効果的であると考えられる．

表 2.9: ODM の CPU コスト (TIS)

ODM	$C_{a_0}+C_p$	C_s	$2C_{\text{rand}}$	C_τ	C_{updateX}	C_{rest}
CPU cost(s)	0.20	1.42	0.08	0.06	1.88	0.08
Persentage[%]	5.38	38.17	2.15	1.61	50.54	2.15

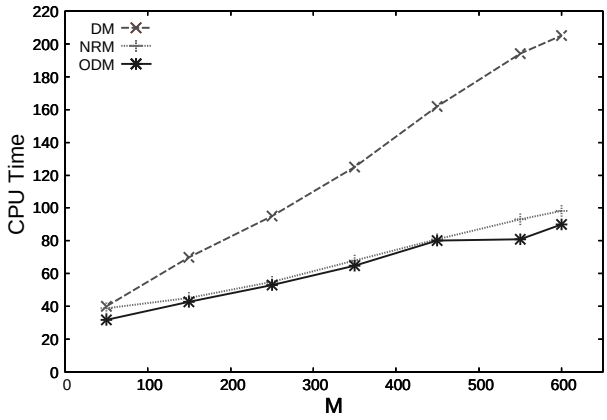


図 2.14: DM，NRM，ODM における分子数 (システム規模)M と CPU 時間の比較

2.3.7.4 ODM における考察

生化学システム規模とコストの比較

図 2.14 に，LCS で M を変えたときの計算時間を示す.

このことから，$N \leq 1000$ 程度の中規模な生化学システムにおいては，ODM は NRM よりも効果的なアルゴリズムであることが示された.

Spatially Inhomogeneous System

疎なシステム ($\mathbb{D}$ が小さい生化学システム) では，ODM はより大きなアドバンテージを持つと考えられる．その一例として，物質が空間上に一様に分布していない場合について考察する．

2 次元の 10×10 の正方格子に区切られ，各格子は (i, j) でラベル付けされている空間を設定する．次に，分子は空間上に A 分子と B 分子が存在し，拡散係数に従ってランダムに移動すると仮定する．この “移動” を反応として定義する．反応 $R_1 - R_4$ は，分子 A が上下左右の隣接格子に移動する反応，反応 $R_5 - R_8$ は，分子 B が上下左右の隣接格子に移動する反応を示す．B は反応定数 k_9 で崩壊する反応 (R_9) と，触媒 A と衝突すると反応定数 k_{10} で分裂する反応 (R_{10}) の 2 種類の化学反応が発生する．各正方形の内部分子をそれぞれ $A_{i,j}$，$B_{i,j}$ と名付ける．このような生化学システムを Spatially Inhomogeneous System(SIS) と呼ぶこととする．

$$
\begin{aligned}
R_1 - R_4\text{(A の拡散)}: &\quad A_{i,j} \xrightarrow{k_1-k_4} A_{i\pm1,j\pm1} \\
R_5 - R_8\text{(B の拡散)}: &\quad B_{i,j} \xrightarrow{k_5-k_8} B_{i\pm1,j\pm1} \\
R_9\text{(B の崩壊)}: &\quad B_{i,j} \xrightarrow{k_9} \emptyset \\
R_{10}\text{(B の分裂)}: &\quad B_{i,j} + A_{i,j} \xrightarrow{k_{10}} 2B_{i,j} + A_{i,j}
\end{aligned}
$$

崩壊の反応定数を $k_9 = 1.0$，分裂の反応定数 $k_{10} = 2.8$，拡散係数を A，B ともに $k_1 = k_2 = \cdots = k_8 = 0.2$，初期分子数を $B_{i,j} = 1$，$A_{10,10} = 1$，$A_{i,j} = 0(i, j = 1, \cdots, 9)$ と設定し，シミュレーションを実行した．シミュレーションの終了条件として以下の 3 項目を設定した．

- 時間が $t = 100$ になったとき
- $\sum_{i=1}^{9}\sum_{j=1}^{9} B_{i,j} = 10000$ になったとき (survival of B)
- $\sum_{i=1}^{9}\sum_{j=1}^{9} B_{i,j} = 0$ になったとき (extinction of B)

シミュレーション実行の結果，約 30% の確率で survival of B 状態になった．このような生化学システムは，解析モデルでは常に絶滅するシステムとして記述されるものであり，同一の条件であっても，確率モデルでは取り得る状態の分布を見ることが可能になっている．シミュレーションに掛かる計算時間を含めて，空間の状態は大きく変動した．しかし，一回のシミュレーションに掛かる時間は様々であったが，多数回のシミュレーションを実行した場合，その総計としての計算時間はほぼ一定になった．シミュレーション 10000 回分の実行時間は，DM で 2045 秒，NRM で 770 秒，ODM で 406 秒だった．その内訳を表 2.10 に示す．

2.4 複合的な生化学シミュレーションアルゴリズム

2.4.1 τ-leap Method

τ-leap Method は Gillespie によって提案された，微小時間単位で確率モデルの計算を行うシミュレーションアルゴリズムである [36]．厳密な確率モデルシミュレーションアルゴリズムでは，反

DM	C_{a0}	C_s	C_p	$2C_{\text{rand}}$	C_τ	$C_{\text{update}X}$	C_{rest}
CPU cost	581.38	786.51	353.82	19.25	17.8	129.2	157.68
Percentage	28.42	38.45	17.3	0.94	0.87	6.32	7.71
NRM	C_{heap}	C_α	$C_{p'} + C_g$	C_{rand}	C_τ	$C_{\text{update}X}$	C_{rest}
CPU cost	332.54	86.37	58.68	10.21	1.7.28	119.08	145.91
Percentage	43.05	11.22	7.62	1.33	2.24	15.45	18.94
ODM	$C_{a_0} + C_p$	C_s		$2C_{\text{rand}}$	C_τ	$C_{\text{update}X}$	C_{rest}
CPU cost	64.83	43.75		17.13	18.84	115.28	144.45
Percentage	16.03	10.82		4.24	4.66	28.51	35.72

表 2.10: SIS における CPU コストの比較

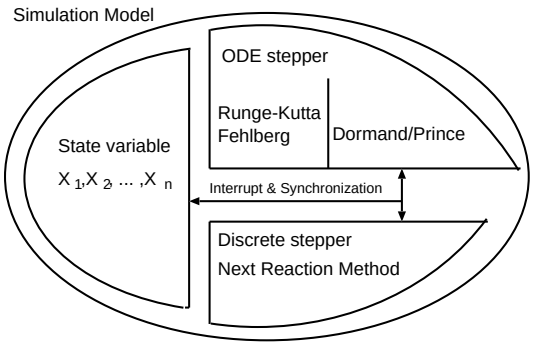


図 2.15: E-Cell3 のシミュレーションモデル

応ごとに計算を繰り返す必要があり，反応発生ごとに各反応の発生確率が更新される．確率モデルシミュレーションでは，1 つ 1 つの反応について，発生順や時間を記録するほどの厳密さは要求されない．そのため，分子がシステム内に多く存在している場合は τ-leap Method を使って微小時間単位で計算を進め，分子数が 0 になってしまう場合や特定の条件を満たす場合のみ，厳密な確率モデルを用いる方法が有効であると Gillespie は述べている．

システムの状態 (各分子の分子数を表すベクトル) を $\overrightarrow{X}$ とし，それぞれの反応 R_j $(1, \cdots, M)$ の発生確率を $a_j(\overrightarrow{X})$ とする．微小時間 τ の間に，分子数の増減による $a_j(\overrightarrow{X})$ の変化が許容しうる範囲内であるとき，時間 $[t, t+\tau)$ で反応 R_j が発生する回数 k_j は，ポアソン分布の確率密度関数 $P(a_j(\overrightarrow{X}), \tau)$ に従う乱数から求められる．微小時間ごとに，各反応 R_j の反応発生確率 $a_j(\overrightarrow{X})$ を計算し，反応発生回数をポアソン乱数から求め，その結果を状態 $\overrightarrow{X}$ に反映させる．反応発生確率はシステムの状態 $\overrightarrow{x}$，すなわち，システム内に存在する分子数で変化するため，誤差が発生する．Gillespie は，4 種類の反応が存在するシステムを仮定して，τ-leap Method が厳密な確率モデルシミュレーションアルゴリズムと比較してどの程度誤差が発生するか評価している．その結果，許容誤差係数 ϵ に適切な値を与えることで，適切な近似ができることがわかった．シミュレーション対象のシステムごとに ϵ の値は異なるため，現在の τ-leap Method は一般的なシステムのシミュレーションに利用することはできない．適切な ϵ の値の決定法や，一般的なシステムに対応できるアルゴリズムは現在開発中である．

τ-leap Method は有効な確率モデルシミュレーションの近似法として期待できる評価が得られている．

2.4.2 E-Cell3

E-Cell[4] は慶應義塾大学富田研究室で開発中の解析モデルによる細胞シミュレータである．E-Cell3[37] は，解析モデルのアプローチに加え，確率モデルによるシミュレーションも可能な統合細胞シミュレータである．

E-Cell3 は，C++で記述されたオブジェクト指向の細胞シミュレータである．E-Cell3 では，シミュレーション対象となるシステムで発生する反応を stepper と呼ばれるクラスにそれぞれ割り当てる．各 stepper で計算される分子数の増減は，微分方程式 (ODE) によるアプローチ (解析モデル)，離散イベントあるいは離散時間によるアプローチ (確率モデル) の 3 種類から選択できる．

シミュレーション対象モデルは複数の stepper を内包し，シミュレーションの進展とともに，各 stepper からの割り込みがあるかどうかを調べる．割り込みがあった場合は，そのシステム時間で

全 stepper の同期を取り，分子数を更新する．

解析モデルの stepper では，求める精度によって ODE の解法を指定することができる．ODE の解法は，陽の Runge-Kutta 法を基にした，2 次の Felhberg 法と 4 次の Dormand/Prince 法が用意されている．ODE でのシミュレーションを指定された stepper は，タイムステップ単位で分子数の変化を計算する．このとき，タイムステップの前後の分子数の変化を評価し，システム状態の変動が許容範囲内に収まるようにタイムステップを増減させることができる．分子変動が大きい部分はタイムステップを縮めてシステム状態の詳細な変化を調べ，変動が小さい部分はタイムステップを伸ばして計算速度を向上させている．各 stepper は自分が計算中のシステム時間を割り込みメソッドとして他の stepper に通知する．微小時間後のシステム時間が最小の stepper に同期しながら全体のシミュレーションが進んでいく．

確率モデルの stepper は，NRM を用いており，乱数生成法には Mersenne-Twister 法を使用している．確率モデルシミュレーションでは分子数の詳細な変化を見ることができる代わりに，stepper 同士の同期も厳密に行う必要がある．そのため確率モデルの stepper は誤差の許容値をパラメータ ξ として持っている．例えば $\xi = 0.1$ ならば，分子数が最も少ない分子が 10% 変化したときの時間を，割り込みメソッドで同期するシステム時間として設定する．

上記のアルゴリズムを評価するため，熱ショック応答システムについて，反応を解析モデルのみで構成したモデル，確率モデルのみで構成したモデル，両者を複合したモデルの 3 つを作り，実行して計算速度を調べた．複合モデルは解析モデルの約 2.6 倍，確率モデルの約 351 倍高速であった．複合モデルが他のモデルより高速になった原因について，反応の一部を解析モデルが担うため，確率モデルよりも高速であり，固い微分方程式を確率モデルで計算することで解析モデルよりもさらに高速になっていることが挙げられる．

2.5 確率モデル生化学シミュレータ

2.5.1 STOCKS

Kierzek は原核生物の遺伝子発現を解析するソフトウェア STOCKS(STOChastic Kinetic Simulation) を開発した [7].

2.5.1.1 STOCKS の特徴

STOCKS は Gillespie が提案していた計算方法 (Direct Method) に加え，生化学シミュレーションに合わせた “体積増大および細胞分裂に対応” および “ランダムプール” の 2 つの機能が実装されている．

体積増大および細胞分裂に対応

STOCKS は，細胞を生化学システムと見なしてシミュレーションする際に，シミュレーション中に細胞が生長する場合や，細胞分裂が起こる場合に対応している．

この機能は以下の項目が前提として仮定されている．

- 細胞は一世代で体積が 2 倍になる
- 細胞分裂が発生すると体積は半分 (初期値) に戻る

これは，Gillespie のアルゴリズムにおいてシステムの体積 V の代わりにシミュレーション時間 t を変数とする関数 $V(t)$ を使うことで実現している．$V(t)$ は式 2.64 で表される式である．T は細胞の一世代の時間を示す．式 2.64 により，細胞の体積が細胞の一世代の時間で線形に増大していく反応を表現することができる．

$$V(t) = 1 + \frac{t}{T} \tag{2.64}$$

具体的なシミュレーション方法は，毎サイクルの時刻 t から $V(t)$ を再計算し，確率反応速度定数 c_j の値を，$V(t)$ の値を使って更新していくことで実現する．$t = T$ すなわち $V(t) = 2$ になったとき，細胞の一世代が終わったことになり，細胞分裂が発生する．確率反応速度定数 c_j は，反応が 1 次反応の場合，式 2.65 で求められる．

$$c_j = \frac{k_j}{N_A V} \tag{2.65}$$

N_A はアボガドロ数である．反応が 2 次反応の場合，確率反応速度定数 c_j は，式 2.66 で求められる．

$$c_j = \frac{2k_j}{N_A V} \tag{2.66}$$

細胞分裂が発生すると，体積および c_j の値が初期値に戻り，そのときに存在する各分子の数を 2 で除算する．DNA をモデル化する分子は除算の前に 2 倍しておき，分裂後も同じ数を保持するようになっている．このような線形の体積変化で細胞生長，細胞分裂を表現する方法は，原核生物のシステムを対象としたシミュレーションでは妥当な結果が得られている [38][39].

ランダムプール

任意の数の分子が存在する “ランダムプール” を用意しておき，分子の数を変動させる機能が追加されている．これは，他からの小さな影響を大量に受けて数が変動する分子がある場合に，有効であると考えられる方法である．STOCKS では，RNA ポリメラーゼのモデリングにこのランダムプールの機能を使用している．

RNA ポリメラーゼのランダムプールは以下のような方法が実装されている．

- ランダムプール内の分子は，サイクルごとに分子数が毎回変更される
- 分子数は a_j を計算する前に，そのときの分子数の平均と標準偏差を用いて正規分布する乱数を得，その値が設定される
- 体積の増加とともにプール内の分子数の平均値は増大するが，濃度は一定のままになる

ランダムプールは RNA ポリメラーゼ以外の分子にも適用でき，また反応分子数の明示的なモデル化が難しい大規模プロセスのバッファにもなると Kierzek は予測している．ただし，ランダムプール機能は，多くの場合に妥当な結果が得られることが考えられるとは言え，数学的に正しさが証明されているわけではないので，ランダムプールを使用する場合には注意が必要とされている．

2.5.1.2 計算時間の問題に対するアプローチ

STOCKS は計算コストが大きく，実行時間は数時間から数日に及ぶため，UNIX オペレーティングシステムのバックグラウンドジョブとして実行することに向いた設計になっている．

そのための方法として，入力ファイル，制御ファイル，出力・ログファイルの 3 種類のテキストファイルでシミュレーションを定義している．

- **入力ファイル**
  入力ファイルにはシステムの仕様が格納される．反応物の構成と初期値，および確率反応速度定数が記述されている．同時に，細胞分裂のルールも記述されている．

- **制御ファイル**
  制御ファイルにはどの反応物の数をモニタするのかが記述される．プログラムは制御ファイルに記述されたシミュレーション時間間隔で反応物の数を記録する．シミュレーションの実行回数，乱数の種も制御ファイルで設定する．

- **出力・ログファイル**
  出力・ログファイルには出力とログが出力されるディレクトリパスを記述する
  - 出力ディレクトリに書き出されるファイルは軌跡 (Trajectry) ファイルと呼ばれる．軌跡ファイルは 2 カラムのテキストファイルで，制御ファイルに記述された時間間隔で時間の関数として分子の数を記録している．Gnuplot のような作図ソフトウェアで使用しやすい形になっている
  - ジョブが中断された場合に復帰できるように，システム内の全ての反応分子数を格納している

2.5.1.3 STOCKS の乱数生成

STOCKS では Marsaglia と Zaman の 2^{144} サイクルの乱数発生器を使っている [40].

2.5.1.4 STOCKS — ユーティリティプログラム

STOCKS にはデータの解析用のユーティリティプログラムが 4 つ提供されている．

1. **軌道平均算出プログラム**
   モンテカルロ方であるため，同じ分子をモニタしているシミュレーションの軌跡ファイルが複数作成される．軌道平均算出プログラムは，これらについて指定した時間間隔で分子数の平均・標準偏差を計算する．

2. **分子数加減算プログラム**
   2 つの軌道ファイルから，ある時間における分子数を加減算するプログラムである．いくつかの分子数の合計を知りたい場合に使用する．

3. **分子数平均算出プログラム**
   軌道ファイルの任意の部分について，分子数の平均を計算するプログラムである．分子数が定常状態に達しているときに使用する．

4. **軌道-1 次関数比較プログラム**
   軌道ファイルの分子数が描く形について，適当な部分に一次関数を合わせることによって，軌道の挙動を確認するプログラムである．分子数が一定速度で増加または減少しているときに使用する．

2.5.2 STOCKS の計算時間

Kierzek は STOCKS の提案に際し，2 つの反応について例示している．Kierzek が言及しているように，計算の目的は，反応物質の詳細なモデル構築ではなく，ソフトウェアの性能評価にある．PentiumIII 800MHz プロセッサを用いて計算時間の評価を行っている．

- **原核生物の転写と翻訳の開始頻度と遺伝子発現の確率変動**
  提案した大腸菌 (*E.coli*) のタンパク質合成速度レベルと，LacZ 遺伝子の mRNA レベルに関する実験データから Kierzek らが提案した運動モデルをシミュレーションするものである．シミュレーションは 100 個の独立した軌道について，細胞 10 世代の期間 (2100 秒間) の計算を行い，100 回のモンテカルロ反復の結果を蓄積し，タンパク質の時間変動をプロットした．このシミュレーションに約 22 時間掛かっている．

- ***E.coli* の LacZ，LacY の遺伝子表現およびそのタンパク質の酵素/転送活性**
  STOCKS が酵素反応と少数の反応分子を含むシステムに適用できることをテストするものである．細胞 1 世代 (2100 秒) の 1 つの軌道の計算に約 2.5 時間掛かる．また，単一の軌道について細胞 10 世代のシミュレーションに約 90 時間掛かる．

2.6 FPGA: Field-Programmable Gate Array

前節までで本研究の生物学的背景について述べた．本節と次節では，本研究で用いる FPGA の構成と，それを用いた計算機システムの実例について概説する．

FPGA は，商用プログラマブルデバイスとして CPLD (Complex Programmable Logic Device) と並んで多く出荷されているデバイスであり，LUT (Look-Up Table) を用いて細粒度のプログラマブルロジック回路を実現することで，大規模な回路を実装可能な特徴を持つ [41][42].

2.6.1 FPGA の構造

FPGA は，LUT を基本ロジックブロックとするプログラマブルデバイスで，n 入力 m 出力の LUT は入力 n ビット，出力 m ビットの任意の論理関数を実現するためのメモリである．多くの商用 FPGA は 4 ビット 1 出力の LUT で構成されており，この場合にはひとつの LUT はアドレス幅 4 ビット，ワード幅 1 ビットのメモリであると考えることができる．FPGA ではこのメモリを，SRAM や Flash ROM，Anti-Fuse ROM などで構成し，値を書き込むことで論理関数を表現する．本研究で用いた Xilinx 社の Virtex-II / Virtex-II Pro シリーズは，LUT を SRAM によって構成した SRAM 型 FPGA である．SRAM 型 FPGA は一般的な CMOS 半導体プロセスで製造されるため，最新のプロセスを利用することができ，プロセスの微細化による回路規模の増大の恩恵を最大限に受けることができる．

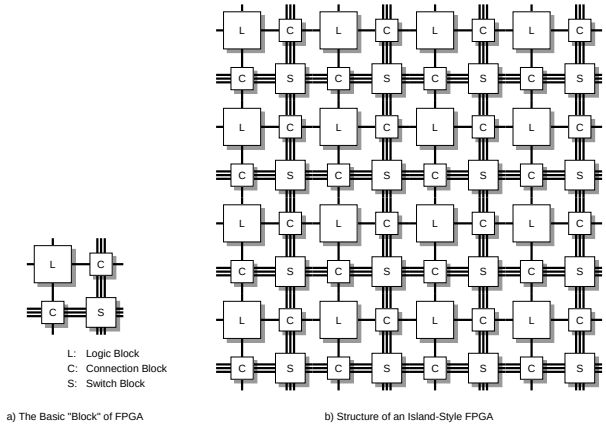


図 2.16: Island-style FPGA の基本ブロックと全体の構造

図 2.16 は，現在の主要な商用 FPGA で採用されている Island-style と呼ばれるアーキテクチャである．このアーキテクチャでは,

- Logic block
- Connection block
- Switch block

の 3 つのブロックで FPGA を構成する．図 2.16(a) は FPGA を構成する基本的なひとつのブロックであり，これを多数並べることで図 2.16(b) のように大きな回路を構成する．Island-style FPGA では構造が均一であるため，ブロック数を変化させることでロジックサイズ変えた製品を容易に作ることができる．

Logic block は LUT を含む，プログラマブルロジック自体を実現するためのブロックであり，隣接する connection block への配線 (図中では 1 本線で表現) を持つ．Connection block は，隣接する logic block への配線と，FPGA 中のグローバルな配線 (図中では 3 本線で表現) との間を接続するパストランジスタを用いたプログラマブルスイッチである．縦横のグローバル配線の交点には，やはりプログラマブルスイッチである switch block が設置されており，これが縦横の配線の間の接続を実現している．このように，FPGA はロジックと配線の双方でプログラマビリティを持つ．

2.6.2 FPGA アーキテクチャの例: Xilinx 社 Virtex-II シリーズ

ここでは，FPGA のアーキテクチャの一例として，Xilinx 社の Virtex-II アーキテクチャについて述べる．このアーキテクチャは同社の Virtex-II シリーズ [43] ならびに Virtex-II Pro シリーズ [44] に共通のものである．

2.6.2.1 CLB の構成

前節で述べた logic block は，LUT だけで構成されるものではなく，実際には図 2.17 に示されるような複雑な構造になっている．図 2.17 (a) は，前節の logic block に相当する CLB (Configurable Logic Block) の構成であり，ひとつの CLB には 4 つの slice と呼ばれるブロックが含まれる．スラ

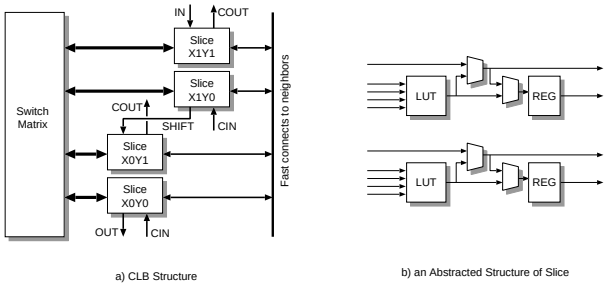


図 2.17: Virtex-II FPGA の CLB の構成

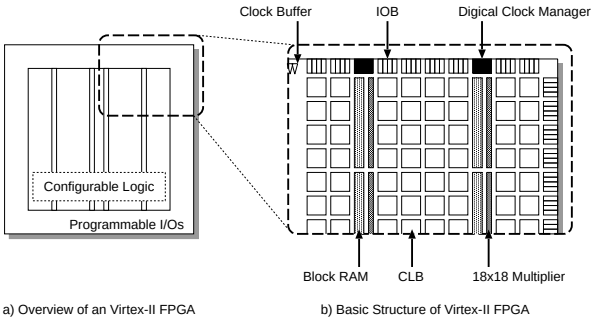


図 2.18: Virtex-II FPGA の構成

イスは図 2.17 (b) に表されるような構造になっており，4 入力 1 スライスの LUT とレジスタを各 2 つずつ備えており，LUT を使って組み合わせ回路，レジスタを使って順序回路を構成することができる．LUT とレジスタの間にはマルチプレクサが設置されており，LUT の出力をレジスタを経由せずにそのまま出力したり，複数の LUT をマルチプレクサによってカスケード接続し，4 入力以上の論理関数を構成することもできる．

また，図 2.17 (b) では省略されているが，図 2.17 (a) に示すように，各スライスにはいくつかの種類の回路を効率よく構成するための専用配線が用意されている．これには，LUT をシフトレジスタとして用いる際に，複数の LUT やスライスにまたがって長いシフトレジスタを効率よく構成するためのシフト用チェーンである IN/OUT や，全加算器を構成する際にキャリーを効率よく接続するための CIN/COUT などがあり，回路面積の削減や動作周波数の向上などに貢献している．

2.6.2.2 Virtex-II FPGA 全体の構成

次に，Virtex-II FPGA の構成を図 2.18 に示す．Virtex-II は，チップ周辺にプログラマブル I/O ブロック (IOB) が配置されている．IOB は CMOS インターフェイス以外にも LVTTL, HSTL, SSTL, LVDS, PCI, AGP など各種の信号規格に対応できるプログラマブルな構成で，DDR 入出力用のレジスタも備えている．

チップ上のロジック部は CLB が碁盤の目状に配置され，その間に接続用の配線が配置される．CLB の間には，縦方向にメモリブロック (BlockRAM) および組み込み乗算器 (18x18 bit) のストライプが数カ所配置される．メモリや乗算器などは，頻繁に使われるが LUT で構成した場合に大きな面積を必要とするため，これらを専用のハードマクロとして組み込むことで回路面積を抑え，

また回路の動作周波数を向上させることができる．

Virtex-II Pro はほぼ同様の構成であるが，BlockRAM や組み込み乗算器と同様の発想で，組み込みプロセッサとして PowerPC 405 と，マルチギガビット・トランシーバをハードマクロとして組み込んでいる．これにより，従来外付け部品によって実現されてきた制御用のマイクロプロセッサや高速作動シリアル通信のインタフェイスの 1 チップ化を実現し，ネットワーク機器等の強力な入出力能力と高速な処理の両方が求められるアプリケーションで力を発揮している．

2.7 FPGA を用いた高性能計算システム

2.7.1 FPGA を用いた計算処理の利点

従来，科学技術計算をハードウェアの力により高速化するには，GRAPE[45] に代表されるような専用計算機を開発する必要があった．専用計算機による計算処理のメリットとしては，

- 必要な演算器をひとつのチップ上に構成し，順番に接続することで，深いパイプラインを構成し，同時に多数の演算器を動かすことにより高いスループットを得ることができる
- 深いパイプラインを構成した場合にも，マイクロプロセッサのようなパイプラインのストールが発生しないため，高い実効性能が期待できる
- チップやボード上のレジスタやメモリを柔軟に利用できるため，小さなメモリブロックを多数並列にアクセスすることにより，大きなバンド幅を得ることができる

といった点が挙げられる．専用計算機の場合には，これらの利点を活かして問題をそのままハードウェア化して計算を行うため，非常に高い性能が期待できる．その一方で，他の問題を解くために利用できないことや，高い開発コストが必要とされることなどが問題である．

これに対して FPGA は，専用ハードウェアのように問題を直接ハードウェアで解くことができるという利点をもちつつ，回路をソフトウェア的に変更できるため，高いコスト対性能比と柔軟性の両立が期待できる (図 2.19)[46]．さらに，FPGA の回路容量は年々拡大しており，表 2.11 に示すように，1990 年代初めにリリースされた 350nm プロセスの FPGA と，2005 年に出荷開始された 90nm プロセスの FPGA とでは，その LUT 数に 25 倍以上の差がある．大きな回路容量の FPGA が手に入ることで，浮動小数点演算を含むアプリケーションの実装が可能になり，近年では FPGA をコプロセッサとして搭載した商用計算機も出現している．本章の以下の部分では，FPGA を用いた商用計算機の例と，FPGA の Bioinformatics への応用例について述べる．

2.7.2 商用計算機への採用例

2.7.2.1 Cray XD-1 (Cray Inc.)

XD-1[47] は，AMD Opteron プロセッサをベースにした並列計算機であり，2 プロセッサ構成の基板を 1 つの筐体あたり 6 枚格納したものを 12 台までひとつのキャビネットに納めることができ，さらにそれを相互に接続してシステム規模を拡大することができる (図 2.20)．最大で 1 筐体あたりのプロセッサ数は 12，メモリ容量は 96GB である．

この，マイクロプロセッサを搭載した基板には Xilinx 社の FPGA がひとつ搭載されており，これをコプロセッサとして利用することでさらに高い性能を得ることもできる．この FPGA は PCI バ

表 2.11: 各世代の FPGA のプロセス・LUT 数と電源電圧 (Xilinx 社)

プロセス	シリーズ	型番	LUT 数	電源電圧
350nm	XC4000	XC4085XLA	7,448	3.3V
250nm	XC4000	XC40250XV	20,102	2.5V
220nm	Virtex	XCV1000	27,648	2.5V
180nm	Virtex-E	XCV2000E	43,200	1.8V
150nm	Virtex-II	XC2V8000	104,882	1.5V
130nm	Virtex-IIPro	XC2VP125	125,136	1.5V
90nm	Virtex-4	XC4VLX200	200,448	1.2V

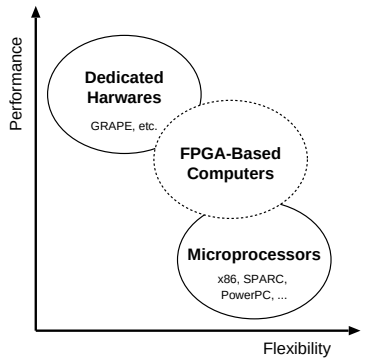


図 2.19: 性能と柔軟性のトレードオフ

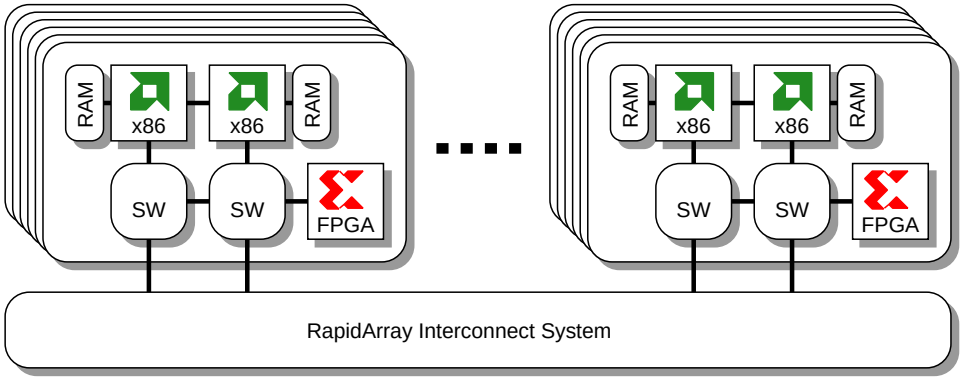


図 2.20: Cray XD-1 の構成

ス経由ではなく，プロセッサ結合網に直結されているため，プロセッサ–FPGA 間で高速にデータを転送することができる．この FPGA を用いたソリューションとして，Cray 社から暗号化，FFT，乱数生成，符号化・復号などの処理や，光学，量子力学，塩基配列の検索などのアプリケーションを高速に実行するためのソリューションが提供されている．また，ユーザがこの FPGA に独自のアプリケーションアクセラレータを実装 [48] することも可能である．

2.7.2.2 SRC-7 (SRC Computers Inc.)

SRC-7[49] は，XD-1 と比べるとより汎用的に FPGA を利用しようというコンセプトで開発されている．FPGA 用のライブラリとして倍精度浮動小数点の演算ライブラリなどが用意されており，専用のプログラミング環境でプロセッサ–FPGA 間の負荷分散を実現できるように配慮されている．開発用の言語としては C のほかに Fortran などもサポートされており，広い分野の科学技術計算ユーザをターゲットとしている．

XD-1 では，マイクロプロセッサと FPGA の数の比が固定されているが，SRC-7 ではマイクロプロセッサを搭載したボードと FPGA を搭載したボードは分離されており，共通のスイッチに接続する設計になっているため，それぞれの数を柔軟に変更することができる (図 2.21)．また，マイクロプロセッサベースのシステムと，スイッチを接続するインタフェイスとして DIMM スロット装着型のインタフェイス (SNAP Interface) を用いることで 6.4GB/s と，1GB/s の 133MHz PCI-X バスよりも広い通信バンド幅を実現している．

2.7.2.3 PROGRAPE-3 (Riken / Tokyo Electron Device)

PROGRAPE-3 (Bioler-3) は，(株) 東京エレクトロンデバイス，千葉大学，理化学研究所などによって開発されたボードであり，Xilinx 社の Virtex-II Pro FPGA (XC2VP70-6) を 4 つ搭載した PCI カードである．PCI カードであるため，通常の PC に組み込んでアクセラレータとして利用することが可能であり，天体間の重力のシミュレーション (多体問題) で 236GFLPS を達成している [50][51].

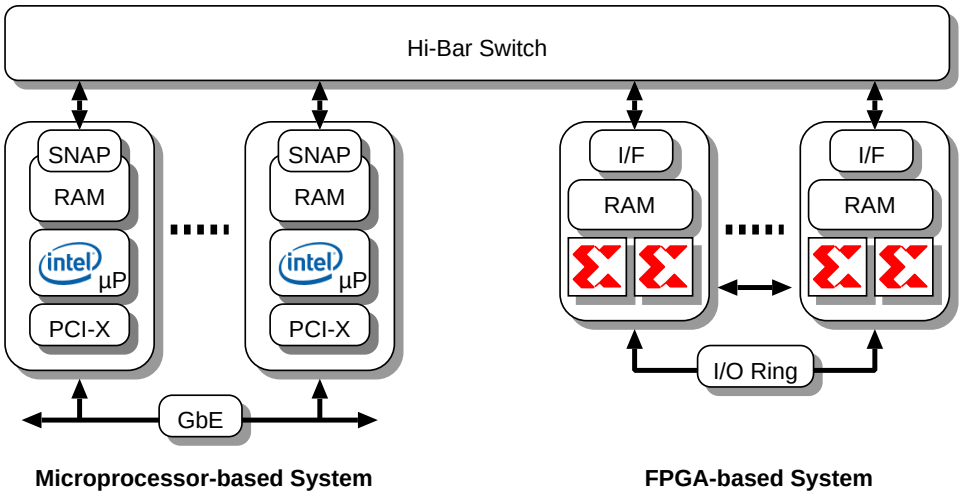


図 2.21: SRC-7 の構成

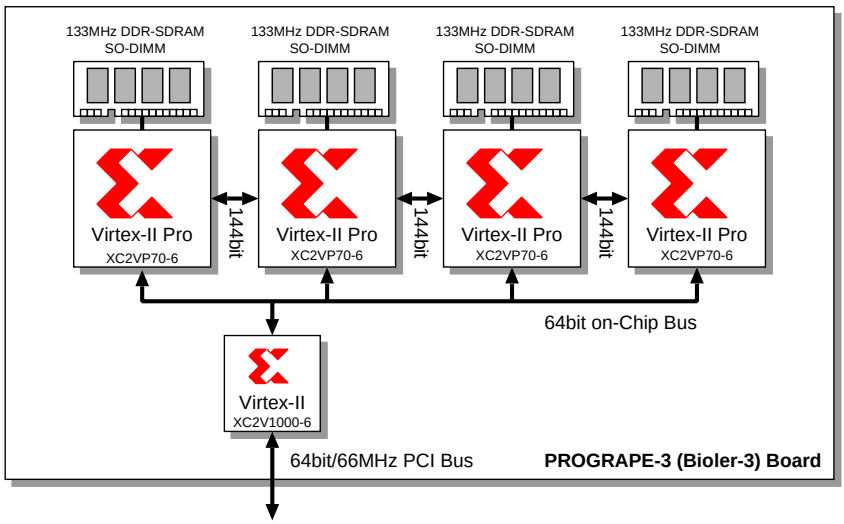


図 2.22: PROGRAPE-3 (Bioler-3) の構成

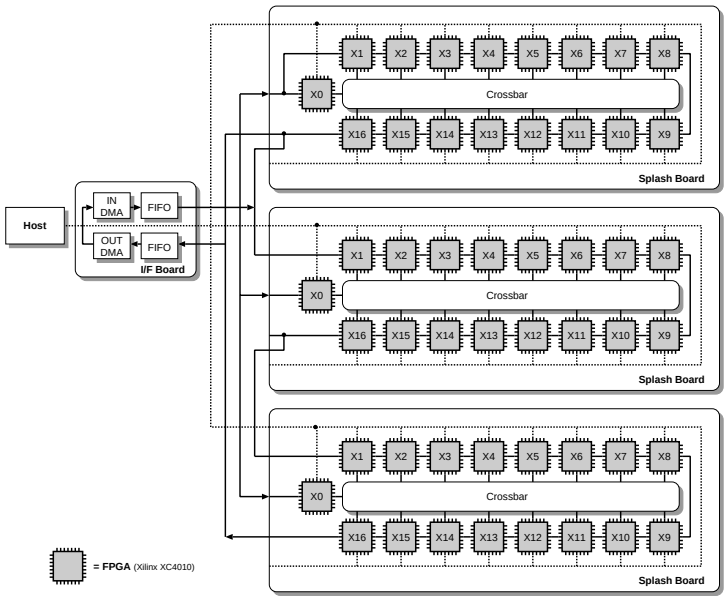


図 2.23: Splash 2 の構成

2.7.3 Bioinformatics 分野での応用例

汎用の計算システムに採用されるようになる以前から，Bioinformatics 分野むけのアクセラレータとしての FPGA の利用が数多く試みられてきた．ここでは，Bioinformatics における，FPGA の従来からの主要なアプリケーション領域やその実装例を概説する．

2.7.3.1 配列の相同性検索

新しく解読された遺伝子やアミノ酸の配列と，機能が同定されているものの配列の相同性を検索することで，その機能を推定することができる．したがって，各種生物のゲノムが解読されている現在，膨大なデータベースを対象に配列の類似度や相違点を検索することが生命現象を構成する部品の機能を解明する上で重要な役割を果たしている．

これらの配列検索には，Smith-Waterman アルゴリズム [52] や Pairwise Hidden Markov Model (PHMM)[53] などが用いられるが，いずれも細かなデータの比較の繰り返しを含むため，FPGA を利用した実装が盛んに開発されている [54]．古くは 1990 年代はじめの Splash 2[55] が，CRAY-2 の 330 倍の性能をマークしたことで有名である．このマシンは図 2.23 に示すような構成になっており，ひとつのボードにクロスバで結合した制御用に 1 個 (X0)，演算用に 16 個 (X1〜X16)，合計 17 個の Xilinx XC4010 FPGA を搭載している．複数のボードを使ってリニアシストリックアレイを構成して処理を行ったり，あるいはクロスバから命令を分配することで SIMD 構成の処理を行うことができる．Splash 2 はのちに WILDFIRE という名称で Annapolis Micro Systems 社から販売 [56] された．

近年では FPGA を用いた大規模なシステムによる解析も行われており，BEE2[57] などのシステムが開発されている．BEE2 は，5 つの Virtex-II Pro FPGA をひとつの基板に搭載した FPGA クラスタシステムである．

また，古くから用いられている 2 本の配列を対象とするアルゴリズムだけでなく，3 本以上の配列のアライメントを調べるアルゴリズムの実装 [58] なども行われている．

2.7.3.2　分子動力学シミュレーション

タンパク質の複雑な立体構造とその機能との関係は生命現象においてきわめて重要であるが，その解明にはタンパク質とその他の分子の間の相互作用を計算機上でシミュレーションすることが重要である．分子動力学シミュレーションは生物学以外の分野にも広く適用できるため古くから研究されており，PC クラスタなどの並列システムや MD-GRAPE などの専用計算機を用いた手法 [59] と合わせて，FPGA を用いたアプローチ [60] も多く研究・開発されている．

2.7.3.3　画像処理アプリケーション

DNA マイクロアレイや顕微鏡の画像からの知識抽出もシステム生物学において重要な役割を果たしている．特に，顕微鏡画像からの知識抽出は画像フィルタリングの過程を含むため，長い計算時間を要する場合がある．顕微鏡で連続的に画像を取得する場合，オンザフライで処理が終了しないと未処理の画像を蓄積するために膨大なストレージを必要とする可能性があり，このような場合には処理の速度が，システム内のストレージの規模とコストを大きく左右する．

本研究で開発されたシミュレータの動作確認用プラットフォームとして開発された FPGA ボードである ReCSiP-2 Board 上でも，線虫 *C.elegans* の初期胚の細胞系譜を顕微鏡画像から自動構築するシステム [61] のアクセラレータを開発し，ソフトウェアによる処理では画像処理用に 32 台の PC クラスタを利用していたものを，処理速度を落とさずに FPGA ボードを組み込んだ PC 1 台で置き換えることに成功した [62].

このように，Bioinformatics の分野でも，FPGA を用いて膨大なデータを高速に処理しようという試みは多数行われてきた．しかし，本研究がスタートした 2002 年の時点では，代謝回路などの生化学シミュレーションの高速化に関する取り組みはまだ発表されておらず，本研究はその先駆けであるといえよう．

2.8　関連研究：FPGA を用いた確率モデル生化学シミュレーションの高速化

2.8.1　FPGA を用いた確率モデル生化学シミュレーションの動向

確率モデル生化学シミュレータを FPGA 上に構成して高速な計算を実現する研究が，2004 年以降，活発になってきている．

Lok は FPGA を用いた確率モデル生化学シミュレーションが大きな効果を上げることをアルゴリズムの性質から示した [63]．Salwinski らはそれを近似アルゴリズムを用いて実現し，ソフトウェアプログラムと比較した場合の精度の検証を行っている [64].

また，Keane らは，確率モデル生化学シミュレータを，生化学システム毎に回路を生成するシステムを構築した [65]．Pentium4 2.0GHz の NRM との計算速度の比較を行い，約 20 倍の速度向上を達成している．しかし，Keane らも同様に近似アルゴリズムを用いており，計算サイクルを反応ごとではなく，タイムスライスで区切ることから，生化学システム毎に精度の検証が必要で

あること，生化学システムの規模が大きくなると FPGA に収まらない規模にシミュレータの回路面積が増大すること等，複数の問題が存在している．具体的には，反応が 120 種類定義された生化学システムで，ハイエンドな FPGA(Virtex-II XC2V8000) の回路面積を超えてしまう．このように，FPGA を用いた高速な確率モデル生化学シミュレータの研究は，現在注目されているものの，高速化の方法論が確立されていない状況である．

2.8.2 本研究の位置付け

本研究は，2001 年から開発が始まった FPGA 搭載 PCI ボード ReCSiP(Reconfigurable Cell Simulation Platform)[66] 上において，確率モデルシミュレーションの実現を目標に開始された．2004 年に Direct Method を用いて Lotka system のシミュレーションを実行する回路が実装され，Athlon 2800+を用いたソフトウェア実行に比べ，約 105.13 倍の計算時間の高速化が図れることが示された [67]．パイプライン化された浮動小数点演算器を組み合わせ，複数のシミュレーションを並列に実行することで高いパフォーマンスを実現していることが，当時のシミュレータ回路を関連研究と比較した時の違いである．回路は Lotka System の計算に特化しており，他の生化学システムのシミュレーションのためには，そのための専用の回路を生成しなければならず，高速化が実現できるものの，回路生成および合成，配置配線，動作検証に掛かる時間が無視できない問題が存在していた．また，生化学システムが大きくなると並列実行数が下がり，スループットが大きく下落する上，回路面積が増大し，大規模な生化学システムのシミュレーション回路は FPGA に収まらない，という致命的な問題が存在していた．

2004 年後半から，FPGA 上で FRM を実行する回路の実装を行った [68]．シミュレーション中のデータを FPGA(Virtex-II) が多く持つ組込メモリ (BlockRAM) に保持することで，システム規模の回路面積に対する依存が小さい構成を目指している．FRM が行う算術演算は 1 種類のみ (式 2.48) であり，パイプライン化された演算ユニットに次々と分子数データを投入することにより，高スループットな演算が可能となった．本研究の 2 つの実装のうちのひとつである，FRM-FPGA はこれらの成果を元に実装されたものである．

2005 年より，生化学システム規模に対するスケーラビリティに加え，FPGA の面積に対する並列実行数のスケーラビリティを実現する回路構成についての検討を行った [69].

実行アルゴリズムは NRM を採用し，NRM を近似しないシミュレータ回路を実現した．また，従来の問題点であった，化学システムの規模と回路面積の問題，生化学システムに依存しない回路構成を実現するアーキテクチャとして，データユニット，演算ユニットをスイッチを介して接続するネットワークを FPGA 上に構成する方法が検討された [70]．本研究の NRM-FPGA は，FPGA を用いたネットワークベースの高速演算環境の検討を目的に実装されている．

第 3 章

確率モデル生化学シミュレータの設計と実装

本章では，FPGA を用いて SSA を実行する回路として，FRM ベースのシミュレータ，NRM ベースのシミュレータの 2 つについての設計について説明する．

3.1 FRM-確率モデルシミュレータ

3.1.1 FRM-FPGA の仕様

3.1.1.1 仕様の概要

FPGA 上で FRM を実行する回路の設計に際して，以下の仕様を定める．

1. 少なくとも 1000 種類以上の分子と反応が定義された生化学システムの計算が可能
   - 生化学システムごとの回路生成，合成，配置配線が不必要
   - 生化学システムの規模と，回路面積に相関の無いこと
2. propencity や τ_j は単精度浮動小数点実数 (36bit) を用いて演算
   - 分子数は 32bit の整数
   - 複数のシミュレーションを並列に実行し，高スループットな演算を行う
3. 2 次反応までの反応に対応

3.1.1.2 大規模生化学システムに対応した構造

関連研究 (2.8) で示されたように，FPGA を用いて SSA を実行する研究の多くは，対象生化学システムごとに回路を生成する方法を利用している．この方法は，以下の 2 つの問題により，対策する必要がある．

1. 数百種類の反応と分子で定義された生化学システムでハイエンドな Virtex-II の回路規模を超えてしまう問題 [65]

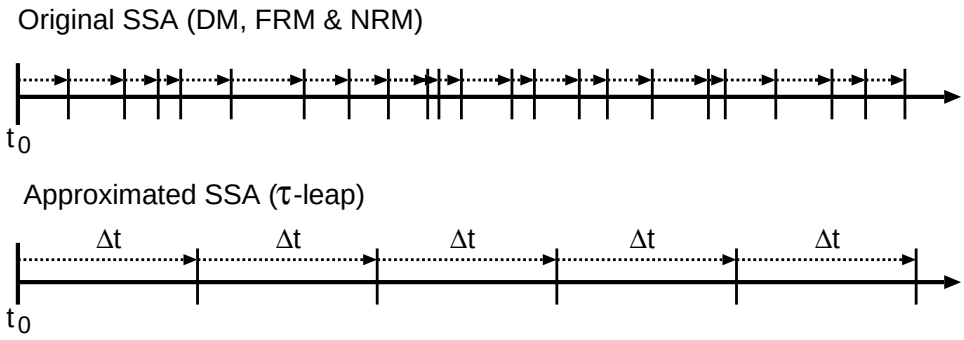


図 3.1: オリジナル SSA と近似アルゴリズムの違い

2. 対象システムごとに回路の合成，配置配線，動作検証に無視できないコストが掛かる場合がある問題

これらの問題に対し，シミュレーションで使用される変数の保持を BlockRAM のような組込メモリを利用することで，一定規模の生化学システムまで回路規模の増加を抑えることで対応する．FRM において，生化学システムごとに異なる部分は，生化学システムに定義された反応および分子の数と，各反応の propencity を求める際に指定する分子の 3 種類だけである．前者は，計算中の生化学システムの反応数および分子数を保持するレジスタを用意することで対応できる．後者は，各反応の propencity の計算に必要な分子を指定するテーブルを持つことで対応できる．

Xilinx 社の Virtex-IIPRO は，18bit x1024entry を 1 単位とする BlockRAM を持つ．これを効率利用しつつできるだけ大規模な生化学システムをシミュレーションするために，1024 entry を生化学システム規模を規定する単位として定める．この基準を利用した場合，最小構成で最大 1024 種類の反応と分子が定義された生化学システムのシミュレーションを，回路面積の増大を招かず，回路の再生成が不必要なシミュレータが実現できる．より大きな生化学システムに対しても，BlockRAM の entry を 1bit ずつ増やすことで対応でき，大規模生化学システムに対するスケーラビリティを保持できる．

3.1.1.3　浮動小数点を用いた演算

関連研究 (2.8) の多くは，SSA をそのまま計算するのではなく，SSA の近似アルゴリズムを利用したり，従来浮動小数点を利用する計算を整数演算で置き換えたりと，FPGA 上で SSA を高速実行するための改変が行われている．

アルゴリズムを変更した場合，結果の確率分布がオリジナルの SSA と同一になることが保証できなくなる．図 3.1 に，オリジナル SSA と近似 SSA の違いの概要を示す．オリジナル SSA は，反応が起こるごとに分子数とシステム時間を更新する．反応 1 回ごとの不規則な時間間隔でシミュレーションが進展するため，定義された生化学システムを厳密に計算することができる一方，膨大な計算の繰り返しが要求される．近似アルゴリズムは，あるタイムステップ Δt をシミュレーション進展の 1 単位として計算を行う．これは，時間 t から $t + \Delta t$ の間に，全反応が発生したか否かを同時に計算し，分子数の更新を行うアルゴリズムである．反応ごとに分子数を更新しないため，SSA の厳密さが失われる．この場合，生化学システムごとに適切な Δt の値を探索して設定しなければならない．また，厳密な Δt の値は，定義された全反応中，最も起こりやすい反応の最短発生時間を設定しなければならず，高速の実現は困難である．

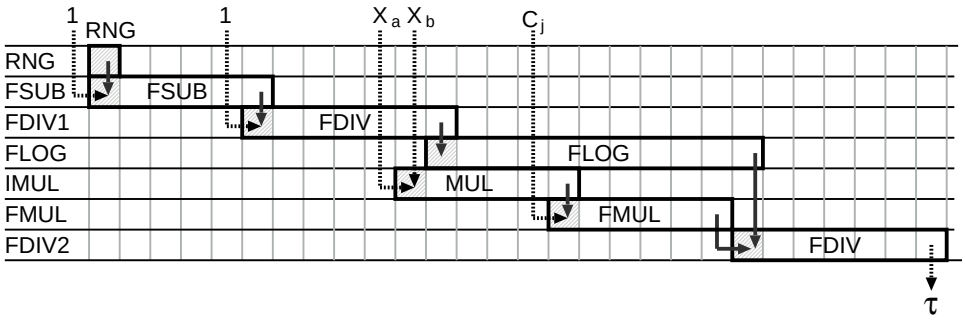


図 3.2: Functional Unit の構成例 (式 2.37-2.41,2.48)

生化学システムは，一般的にマルチスケールであり，分子数や速度定数が分子や反応によって 10^7 倍以上の差が存在する場合がある．このような生化学システムの場合，32bit 程度の整数では計算精度が十分でない．実用的な生化学システムのシミュレーションのためには，SSA の性質上，分子数は整数型で保持するべきであるが，τ と μ の決定には実数型を用いる必要がある．

一般的に FPGA 上で浮動小数点実数)(FP) の算術演算を実装した場合，動作周波数を高く保つためには，パイプラインを深く刻む必要がある．しかし，現在の汎用プロセッサの動作周波数は FPGA の動作周波数と比べて数十倍高速であるため，入力から結果を得るまでに数クロックから数十クロック掛かる演算を逐次的に実行していく方法では汎用プロセッサを超える性能を実現することは不可能である．そのため，パイプライン化された FP 演算器を組み合わせた機能ユニット (FU:Functional Unit) に対し，連続的にデータを投入することで高スループットな演算を実現する．

図 3.2 に，propencity の計算および式 2.48 の計算を行う FU(FRM-TAU) の例を示す．FRM-TAU は，最大 2 種類の分子数と反応番号 j から，propencity を求めた上で，反応の予測発生時間 τ_j を計算する．FRM-TAU の演算部は 7 つの演算ユニットが接続されることで構成されており，パイプラインピッチは 1 である．このため，毎クロック連続的にデータを投入することで，1 クロックに 1 反応分の τ_j を計算することができる．

パイプライン化された演算ユニットに次々にデータを投入し，τ_j を求める計算 (式 2.48) に掛かる時間を隠蔽することで，高速な演算を実現する設計を行う．

3.1.1.4　2 次反応の制限

生化学システムで発生する化学反応は，分子の分解あるいは分子同士の衝突により発生すると言えるため，2 次反応まで対応するシミュレータであれば，多くの生化学システムのシミュレーションが可能である．FRM-FPGA は 2 次反応までの化学反応が定義された生化学システムをシミュレーションできる回路とする．

3.1.2　FRM-FPGA の設計

FRM ベースの SSA 実行回路は，大きく分けて 3 つの部分に分割される．

- 浮動小数点 (FP) 演算を行う演算ユニット (FRM-FU)
- シミュレーションデータを保持するータユニット (FRM-DU)
- FRM-FU と FRM-DU を接続し，適切にデータを投入するコントローラユニット (FRM-CU)

3.1.2.1 FRM-FU

FRM-FU は，さらに以下の 3 つに分類される．

1. Propencity を計算する演算部分 (PF-FU)

$$a_x = X_a \times X_b \times c_x \tag{3.1}$$
$$a_y = \frac{X_a \times (X_a - 1)}{2} \times c_y \tag{3.2}$$
$$a_z = X_a \times c_z \tag{3.3}$$

2. 以下の式 3.4 から，予測発生時間 τ_j を計算する部分 (TC-FU)

$$\tau_j = \frac{1}{a_j} \ln\left(\frac{1}{r}\right) \tag{3.4}$$

3. システムで定義された全反応の τ_j から最小値を求める部分 (MIN-FU)

$$\mu = \text{GetReactionID}\left(\min\left(\overrightarrow{\tau}\right)\right) \tag{3.5}$$

Propencity-Unit(PF-FU)

仕様上の制限により，シミュレーションする化学反応は 2 次反応までであるため，propencity は式 3.1-3.3 の 3 種類の演算が可能であれば，求めることができる．反応番号 j と分子数 X_a，X_b を入力とし，τ_j を求めるために，以下の演算ブロックで構成された演算ユニットを使用する．

- 反応型テーブル
- 32bit 整数乗算ブロック
- 整数型-実数型変換ブロック
- FP 乗算ブロック
- 反応定数テーブル

これらを組み合わせて構成された PF-FU を図 3.3 に示す．反応型テーブルは，反応 R_j が式 3.1-3.3 のどの形の反応であるかを指定するテーブルである．反応型が式 3.1 の場合，X_a，X_b が整数乗算ブロックの入力となる．反応型が式 3.2 の場合，X_a と $X_b - 1$ が整数乗算ブロックの入力となる．乗算結果が右に 1bit シフトされる．X_a，X_b は整数であるため，ビットシフトによる定数除算が可能である．

乗算結果を実数型型に変換し，入力 j をアドレスとしてテーブルから読み出された反応パラメータ c_j と乗算する．反応パラメータ c_j は実数型でテーブルに格納されている．反応型テーブルや反応パラメータテーブルを BlockRAM のような組込メモリを利用して構成し，シミュレーション実行前に書き換えることで，回路を変更せずに様々な生化学システムに対応できる構成を取る．

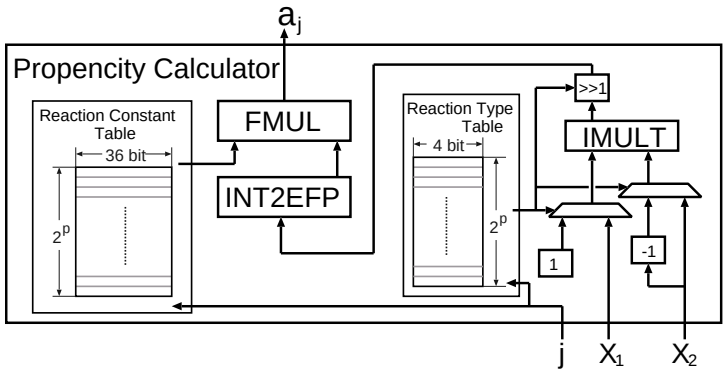


図 3.3: FRM における propencity 計算ユニット PF-FU

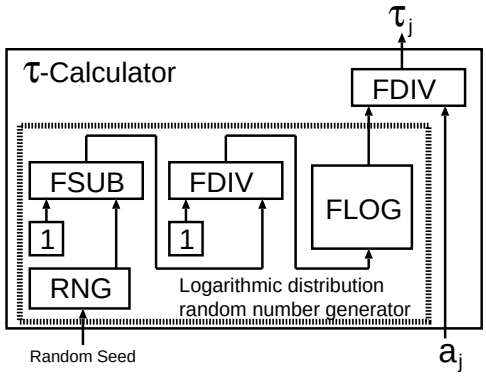


図 3.4: τ_j 計算ユニット TC-FU

τ-calculator(TC-FU)

反応予測発生時間 τ_j の計算には，propencity a_j と $[0,1]$ の範囲の乱数 r から計算できる．図 3.4 に TC-FU の構成を示す．使用する演算ブロックを以下に示す.

- FP 除算ブロック $\times 2$
- FP 対数演算ブロック
- FP(加) 減算ブロック
- 乱数生成ブロック

入力 a_j から式 3.4 によって τ_j を計算する場合は，入力 a_j を対数分布する乱数 $\ln(1/r)$ で除算すればよい．そのため，乱数 $\ln(1/r)$ を供給する演算ユニットと，FP 除算器を接続することで実現する．対数分布する乱数は $[0,1]$ の範囲の実数型乱数を $\ln(x)$ を計算する対数演算ユニットに入力して生成する．乱数生成器は，167bit の LSFR(Linear Feedback Shift Register) により M 系列で 1 クロックに 26bit の乱数を生成する演算ユニットを実装して使用する．$[0,1]$ の範囲の乱数は，26bit の乱数を FP の仮数部として用い，正の符号 (0) と指数部として 127(0) を付加することで $[1,2]$ の範囲の一様乱数 r_p を作り，そこから 1.0 を減算することで生成する．

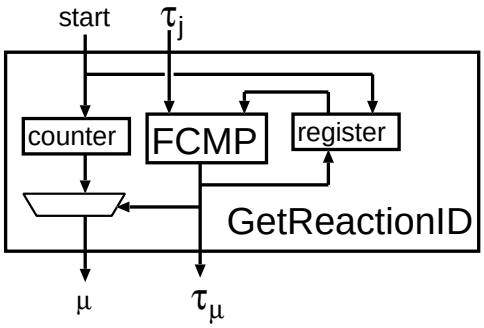


図 3.5: GetReactionID ユニット

GetReactionID (MIN-FU)

式 3.5 の GetReactionID は，引数として与えられたベクトルに格納された値のうち，最小値とその配列のインデックス番号を返す関数である．これを実現する FU である MIN-FU は，FP 比較器と τ_j を格納するレジスタおよび “現在は何番目の入力か” を示すカウンタから構成される．

MIN-FU は τ_j の入力を 1 ポートと，τ_1 の入力であることを示す start 信号を入力とする．他の 2 つの FU とは違い，MIN-FU は 1 クロックに 1 つずつ入力される，τ_j $(j = 1, \cdots, N)$ の中から，最小の τ_j の値とその j を出力する．ここで使用する FP 比較器は，2 つの入力から小さい値を出力する演算器である．

MIN-FU は，計算中の生化学システムに定義されている反応数 N を 1 サイクルとして計算が進展する．MIN-FU の挙動は以下の通り．

STEP1 start 信号を Enable にすると同時に，FCMP に τ_1 と $+\infty$ を入力 (register に τ_1 が入る)

- counter を 1 に初期化 ($j \leftarrow 1$)

STEP2 FCMP に τ_{j+1} と register の値を入力

- counter$\leftarrow$ counter $+1$
- $j \leftarrow j + 1$
- FCMP の出力が変わった場合 (入力された τ_{j+1} がそれまでの最小値だった場合)counter の値と τ_{j+1} を μ ポートと τ_μ ポートに出力

STEP3 $j \neq N$ ならば，STEP2 に戻る

STEP4 このときの出力が，$\min(\overrightarrow{\tau})$ と μ

MIN-FU はパイプラインピッチが 1 ではなく N である．N 個の τ_j をクロック毎に連続して MIN-FU に入力することで，$\min(\overrightarrow{\tau})$ とそれが何番目の入力 (μ) であったかを得ることができる．

3.1.2.2 FRM-DU

FRM-DU は，シミュレーションの進行状態を保持するメモリである．FRM において，単一のシミュレーション実行中に保持する必要があるデータを以下に示す．

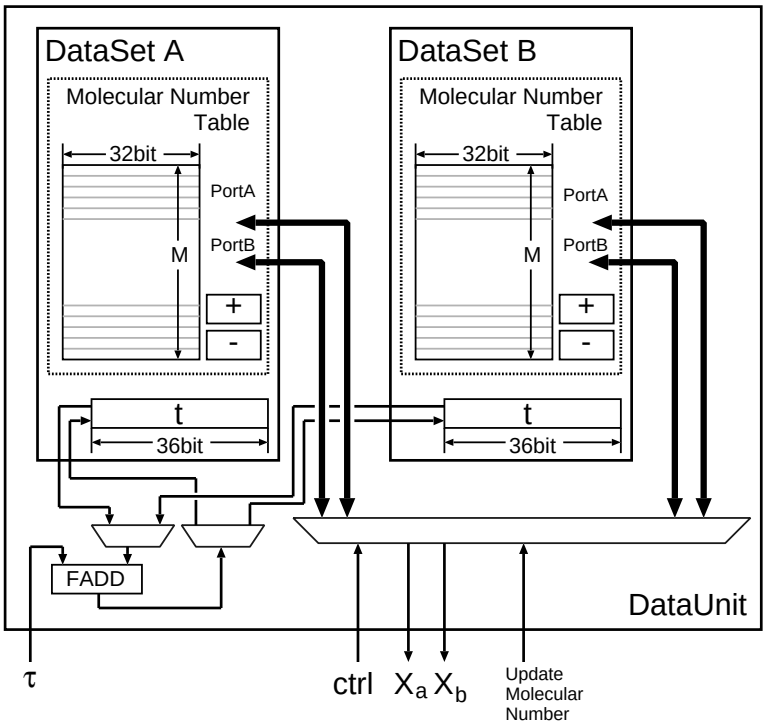


図 3.6: データユニットの構成

- 分子数 $\vec{X}$
- 現在のシステム時間 t

図 3.6 に FRM-DU の構成を示す．分子数は 32bit のデータ幅を持ち，最小で 1024 エントリの BlockRAM に格納される．コントローラは各反応 R_j の Reactants をもとに，分子数テーブルから分子数を読み出し，PF-FU に入力する．また，MIN-FU から出力された τ_μ とシステム時間 t を加算する．データユニットは，CU から μ を元に入力された分子数更新データを使用し，分子数を更新する．DU は内部に複数データセットが配置され，読み出すデータセットは CU が指定する．FU はパイプライン化されているため，1 クロックに 1 つの τ_j を計算できる．しかし，全反応の τ_j を毎サイクル計算しなければならないため，生化学システムの反応数 N が数十以上に大きくなった場合，多数のデータセットが配置されていたとしても，スループットはほぼ同一になる．そのため，データセットを 2 つ配置することで，τ_μ が得られてから分子数更新までのパイプラインの空きを隠蔽する．DU 内部のデータセットをそれぞれ DS_A，DS_B と呼ぶ．

3.1.2.3 FRM-CU

FRM-CU は，各反応の Reactants が格納される Reactants テーブルと，反応が何種類定義されているかを格納するレジスタ，反応番号を示すカウンタ，計算をどちらのデータセットに対して適用するかを示すフラグで構成される．図 3.7 に FRM-CU の構成を示す．

計算開始とともに，反応 R_1 から順に propencity 計算に必要な分子番号が Reactants テーブルから読み出され，DU に出力される．DU の DS_A から分子数が読み出され，FU に入力される．定義

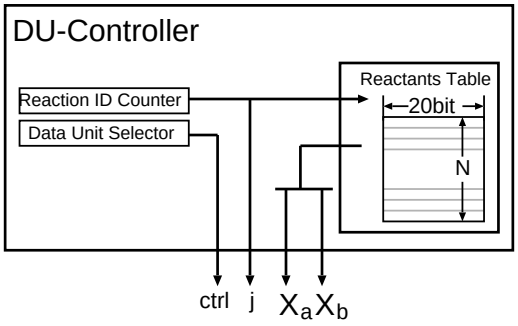


図 3.7: FRM-CU の構成

.

されたすべての反応について Reactants が読み出されたとき，データセット指定フラグが切り替わり，DU から分子数を読み出すデータセットが DS_B に変更される．

3.2 NRM-確率モデルシミュレータ

3.2.1 NRM-FPGA 仕様

FPGA 上で NRM を実行する回路の設計に際して，3.1.1.1 節で決定した FRM-FPGA の仕様に加え，以下の仕様を定める．

1. 近似を用いず，NRM のアルゴリズムを実行する
2. データユニットと演算ユニットを独立した設計とし，回路規模の増大とともに並列実行数を増やせる構成とする
3. ユニット間をスイッチで接続し，複数データユニットで共有される演算ユニットを共有する構造とする

3.2.2 NRM の動作解析

NRM は演算とデータの観点から，以下の 3 つの部分に分類できる．

1. 浮動小数点 (FP) 演算
   - k_j のテーブルを含む propencity の計算 (式 2.37-2.41)
   - 式 2.48，式 2.58 の計算
2. 生化学システムごとに設定されるが，実行中に値が不変の参照用データ
   - 状態更新ベクトル ν
   - Dependency Graph
   - Reactants テーブル

3. 生化学システムの状態を保持するデータテーブル

    - Indexed Priority Queue
    - 分子数テーブル

各分類について，詳細に検討する．第 1 項，第 2 項の FP 演算器やメモリ (BlockRAM) を組み合わせて実装されたハードウェアモジュールを演算ユニット (FU) と呼ぶ． 第 1 項の FU は，propencity 演算，式 2.48 の演算，式 2.58 の演算を行う 3 種類であり，それぞれ PF-FU，TC-FU，TR-FU と呼ぶ．第 2 項の FU は，生化学システムごとに設定され，シミュレーション実行中は値が変化しない定数テーブルが 3 種類あり，状態更新ベクトルテーブル (UM-FU)，Dependency Graph(DG-FU)，Reactants テーブル (RT-FU) である．第 3 項のシミュレーション状態を保持するデータテーブルをデータユニット (DU) と呼ぶ． DU は最小構成で IPQ(2 分木とポインタテーブル)，分子数テーブルが必要であり，計算時間とのトレードオフにより，propencity テーブル等が追加される．

動作周波数が汎用 PC のプロセッサと比べて数十分の一である FPGA を用いて，アルゴリズムを近似せずに単一のシミュレーションを高速に実行することは困難である．現在実装されている FPGA 用の FP 演算器は，全てパイプライン化されているため，1 クロックに 1 回の演算を実行することができる [71]．そのため，多数の DU と少数の FU を適切に接続した回路を構成し，FU のパイプラインを埋める通信を行うことで，複数回分のシミュレーションの並列実行による高スループットな演算の実現を目指す．このような高速化の方法は，生化学シミュレーションの主な用途が反応パラメータの決定を目的とするパラメータサーベイであること，さらに SSA はモンテカルロ法であり，同一パラメータによる多数回の実行結果が必要であることから，十分に合理的であると言える．

3.2.3 データ転送網によるユニット間接続

現在までに FU と 20 個の DU をクロスバで接続した NRM シミュレータを実装し，評価を行っている [72]．このシミュレータは，各 DU に割り振られる各 FU への通信権が時分割であり，木の再構成の待ち時間をワーストケースで動作する．対象の生化学システムが大きい場合に，計算速度の下落が大きいこと，20 個の DU でも FU のパイプラインの使用率が低く，DU 数が増えた場合にクロスバの面積増加が深刻であることなど，いくつかの問題が存在していた．

そこで，DU-FU 間が少数のポートを持つ複数のバスやルータを介して接続された NRM 実行回路を提案する．図 3.8 に 5-port のルータを用いた DU を 1 個だけ持つ NRM 実行回路の例を示す．ルータを介して DU を複数接続することで，同時に動作する DU の数を増やすことができる．全体をクロスバで接続する方法と比べ，DU 数が増えた場合に，ユニット間接続に必要な面積の増大が小さいため，より効率的な面積利用が可能になると考えられる．また，実行する FPGA の回路規模が異なる場合にも柔軟に対応できるため，回路面積の増大に比例したスループットの向上も期待できる．

NRM は必ず，DU から何らかのデータを FU に送り，演算あるいはテーブル引きの結果が，FU から送信元 DU に返送される手順を繰り返して計算が進展していく．単一 DU の，初期化処理を除いた反応 1 サイクルの演算は，以下の手順で行われる．

1. 2 分木の根ノードの反応番号 μ を UM-FU に送り，増減する分子番号を得，分子の数を更新する

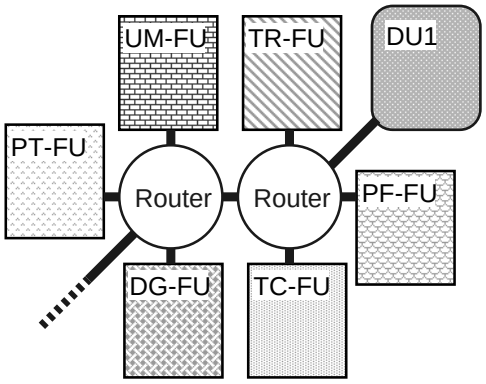


図 3.8: データ転送網を用いた NRM 回路の例

2. μ を RT-FU に送り，R_μ の Reactants を得る
3. 得られた Reactants を PF-FU に送り $a_{\mu,\text{new}}$ を得る
4. $a_{\mu,\text{new}}$ を TC-FU に送り，$\tau_{\mu,\text{new}}$ を得る
5. μ を DG-FU に送り修正が必要な反応番号 x を得る
6. τ_x を TR-FU に送り $\tau_{x,\text{new}}$ を受け取る
7. 他に修正すべき反応があれば，(5) に戻る
8. (1) に戻る

3.2.4 Data Unit

シミュレーション状態を保持するために必要なデータは以下の 3 種類である．

1. IPQ(2 分木，ポインタテーブル)
2. 分子状態テーブル
3. Propencity Table

これらのメモリは，Virtex-IIPRO の BlockRAM を使用している．BlockRAM は 1 つあたり 18bits ×1024 エントリであることから，各メモリのエントリは 1024 を使用している．2 分木はアドレス 0 番地を使用しないため，この構成では，最大で反応 1023 種類，分子 1024 種類で構成される生化学システムのシミュレーションを実行することができる．2 分木は，BlockRAM 外部に木のノードの比較，交換を行うコントローラ (Tree Controller) を実装して実現している．より上位のモジュールからは，命令入力 (4bit)，ノード入力 (46bit)，根ノード値 (46bit)，出力ノード値 (46bit) の 4 種類の入出力でコントローラが制御される．2 分木は参考文献 [72] で実装された 2 分木から，先読み機能を削除したものである．これは，先読みのための回路のオーバヘッドが大きいことによる．ノードの交換には 3 クロック掛かり，交換と同時にポインタテーブルの値も交換される．a_j は必要になるごとに再計算することが可能だが，実行時間の短縮のため，Propencity Table に保持している．DU 外部との通信は，専用のコントローラが行う．

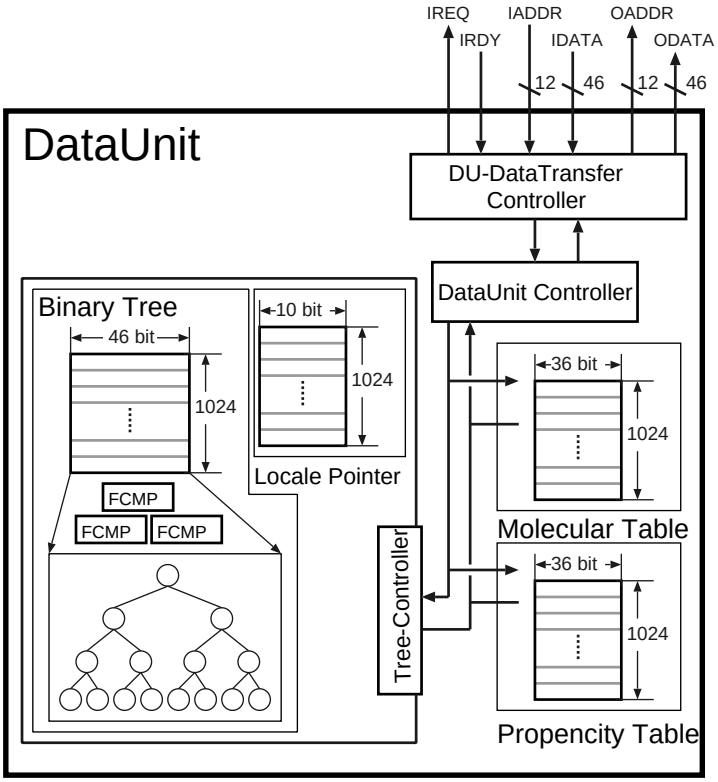


図 3.9: Data Unit の構成

3.2.5 Functional Unit

3.2.5.1 FP 演算を行う FU

FU は 2 分類 計 6 種類必要になる．第 1 分類の FU はパイプライン化された FP 演算器の組み合わせで構成され，PF-FU，TC-FU，TR-FU の 3 種類ある．第 1 分類の FU について， PF-FU を例に説明する．PF-FU は，反応番号 j と何種類かの分子の数を入力とし，a_j を計算結果として出力する FU である．PF-FU は内部に k_j テーブル，反応型テーブルを持っており，入力された反応番号 j をアドレスとしてテーブルを引くことでその反応の k_j および反応型を得ることができる．現在の PF-FU は 2 次反応までの a_j を計算できるが，同じインタフェースで 3 次以上の反応にも対応することが可能である．2 次反応までの a_j の計算は式 2.37-2.41 の 3 種類であり，反応型は式 2.37-2.41 を選択するためのラベルである．分子数は整数のため，Reactants の組合せ数は，入力のビットシフトと整数の乗算で実現できる．図 3.10 に PF-FU の構成を示す．PF-FU の入出力部には DU と同じ信号線があり，外部と通信が行われる．

入力から計算結果の出力までに掛かる数十段分のパイプラインの占有率が，高スループットを実現するための評価基準のひとつになる．

3.2.5.2 定数テーブルを持つ FU

FU の第 2 分類は，シミュレーション中に利用される DG や Reactants に関するテーブルを持つユニットであり，DG-FU，UM-FU，RT-FU の 3 種類存在する．これらのテーブルの値は，シミュレーション対象である生化学システムごとに設定され，実行中は値が変化しない．

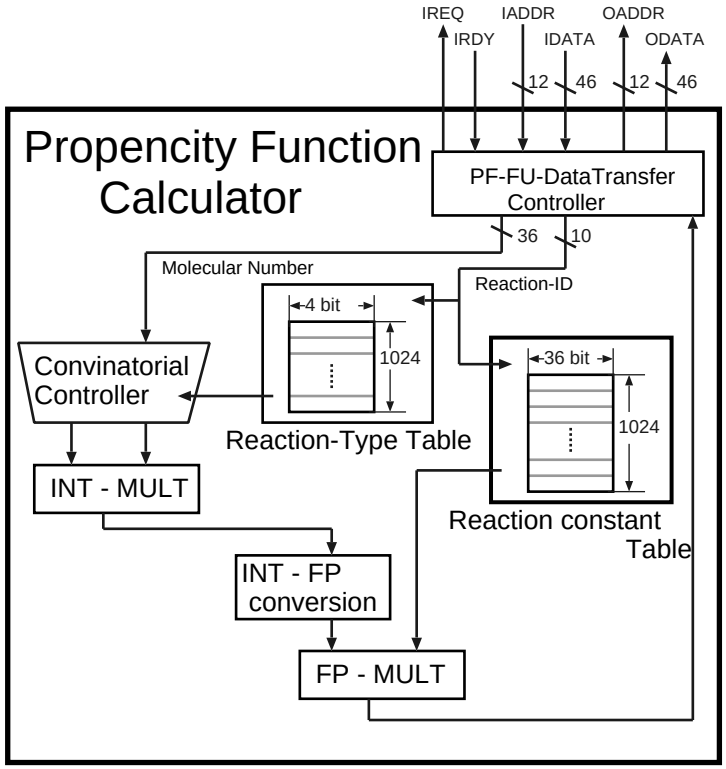


図 3.10: PF-FU の構成

第 2 分類の FU について， DG-FU を例に説明する．DG-FU は反応番号 j を入力とし，R_j に関する DG のリストを出力する FU である．DG のリストの数は反応ごとに異なるため，十分に大きな DG 本体メモリと，各反応のリストが本体メモリのどの位置から始まるかが格納されているポインタテーブルで構成される (図 3.11)．現在の実装では，DG-FU からは反応を 1 種類ずつ返送する．入力として，反応番号 j とオフセット x を受け取り，DG(Pointer$(j) + x$) を返送する．

3.2.6 Router (Switching Unit)

3.2.6.1 Router の構成

DU1 個の試験評価を取るために，簡易なルータの実装を行い，ユニット間の接続に使用している．実装したルータが持つ機能を以下に示す．

- 5 ポートの簡易なルータ
- 転送は 1 クロック
- 衝突検出機能無し
- アドレス併走方式 (アドレス 12bit，データ 46bit)
- ソースルーティング方式
- 1flit 毎に転送を行い，バスの占有などは行わない

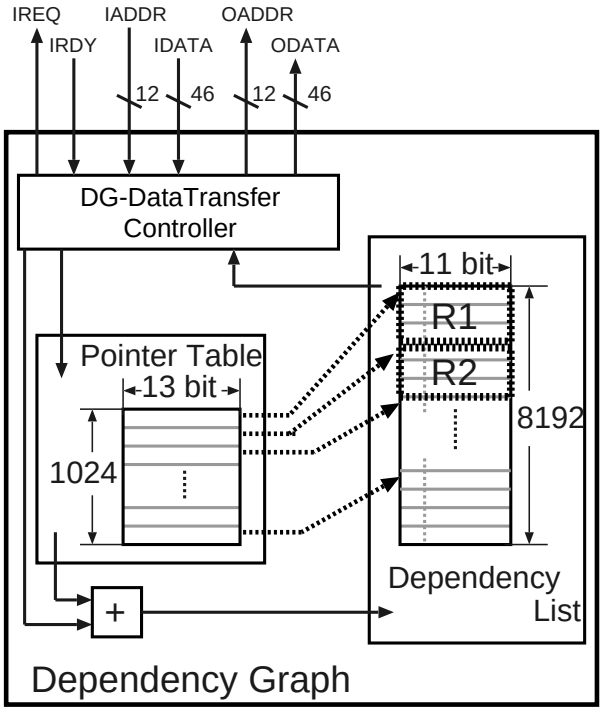


図 3.11: DG-FU の構成

実装したルータは 5 つのポートを持つ．転送されるデータの 1 単位 (データフリット) は，転送先アドレスとデータを 1 セット，58 ビット幅で構成される (図 3.12)．アドレス部分とデータ部分は明確に分離されている．アドレス部分はユニット間の転送アドレスだけが格納されており，データ転送のルーティングのみに使用される．データ部分は，SSA を計算するために必要なデータが格納される．

3.2.6.2　ルーティング方式

データフリットのアドレス部分は，ユニット間のデータ転送に必要なルーティングに使用される．図 3.12 にデータフリットのアドレス部分の構成を示す．データ転送は宛先ユニットまでの経路をアドレス中に保持するソースルーティング方式で行う．アドレスは最大 3 ホップ分の経路を保持し，ユニット間に最大 3 つのルータを経由することができる．現在の実装では，Burst mode ビット，Exeption Flag は使用していない．

図 3.13 にルータのポートと入出力信号を示す．DU とルータ，ルータ間，ルータと FU でのデータの転送は，ordy 線と req 線で衝突を検出できる構造に成っている．NRM は，DU から何らかの FU に対してデータを転送し，FU で適切な処理を行った上で送信元の DU へ結果を返送する，という性質を持つ．

ルータでのルーティングは以下の手順で行われる．

1. 転送されたデータのアドレスの 1st hop を取り出す
2. 2st hop，3rd hop を 3 ビット左にシフトする．右端の 3 ビットには入力ポートの番号が格納される

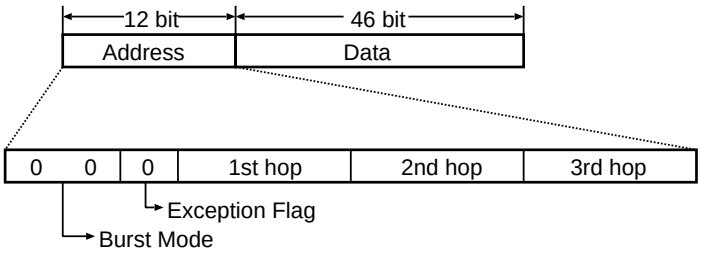


図 3.12: データフリットの構成

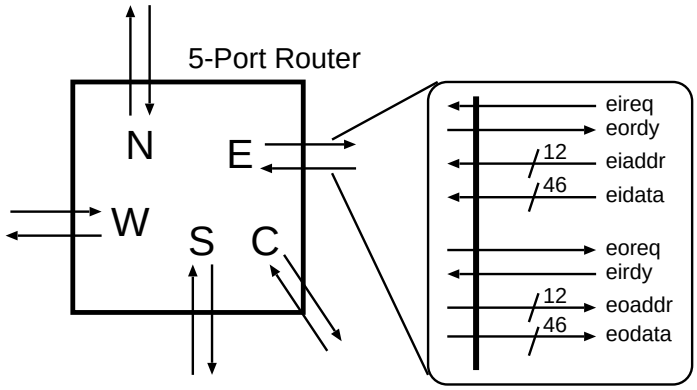


図 3.13: Router の構成

3. hop 先のポートが空くまで待つ．空いている場合は次クロックに宛先に転送される

FU では以下の処理を行う．

1. フリットを受け取ったら，データ部分を FU に入力する
2. アドレス部分の 3rd hop と 1st hop を入れ替える
3. FU から結果が得られるまで待つ

以上の手順で，DU が特定の FU へ転送する適切なアドレスを設定してフリットを送出すれば，同一の経路で FU から送信元の DU へ結果が返送することができる．

第 4 章

評価

4.1 ソフトウェアによる生化学シミュレーションの速度評価

FPGA を用いた確率モデル生化学シミュレータを評価するために，動作検証・速度計測用に，各アルゴリズムを用いて SSA を実行するソフトウェアプログラム SSA-SW を C++を用いて実装した.

4.1.1 SSA-SW の概要

SSA-SW の概要は以下の通り.

- FRM，DM，NRM の 3 つのアルゴリズムを選択して SSA を実行する
  - 実行アルゴリズムに合わせて，FRM-SW，DM-SW，NRM-SW と呼ぶ
- 実数フォーマットには単精度浮動小数点実数を用いる
- 乱数生成アルゴリズムには，Mersenne-Twister[73] を用いる

FRM-SW，DM-SW の実行には，生化学システムと実行環境を定義する以下の 3 つの設定ファイルが必要となる.

1. **分子設定ファイル molecular.txt**
   生化学システムを定義する分子の数と，初期分子数が記述される．例として LTS(2.3.3 節参照) の molecular.txt を図 4.1 に示す.

```
# Lotka System
4           # この生化学システムで定義される分子の数 (4 種類)
100000      # 分子 S1 の初期分子数
1000        # 分子 S2 の初期分子数
1000        # 分子 S3 の初期分子数
0           # 分子 S4 の初期分子数
```

図 4.1: LTS の分子設定ファイル molecular.txt

2. **反応設定ファイル reaction.txt**
   生化学システムを定義する反応の形と状態更新ベクトルが記述される．LTS の reaction.txt を図 4.3 に示す．各反応に関する記述は，図 4.2 に示したフォーマットで定義される．

```
[Reactants の数] [数が変更される分子の数] [反応定数] {[分子番号  増減数]...}
```

図 4.2: 反応設定ファイルにおける反応定義

```
# Lotka System
4                                   # 定義される反応の数 (4 種類)
# Reactions
2 1 0.0002 0 1     1 1              # 反応 R1 に関する記述
2 2 0.01   1 2     1 -1   2 1       # 反応 R2 に関する記述
1 2 10.0   1       1 -1   3 1       # 反応 R3 に関する記述
1 2 10.0   2       2 -1   3 1       # 反応 R4 に関する記述
```

図 4.3: LTS の反応設定ファイル reaction.txt

3. **シミュレーション設定ファイル settings.txt**
   実行する SSA のアルゴリズムと，シミュレーションに関する設定が記述される．10 万回反応が発生するまでの分子数の変化を 1 反応毎に出力し，実行アルゴリズムとして DM を用いる場合の settings.txt の例を図 4.4 に示す.

```
0          # 実行するアルゴリズム (0:DM,1:FRM,2:NRM)
100000     # シミュレーション終了までの反応回数 (10 万反応まで)
1          # 結果を出力する反応間隔 (1 反応毎)
```

図 4.4: シミュレーション設定ファイル settings.txt の例

NRM-SW の実行には，上記 3 つの設定ファイルに加え，Dependency Graph に関する設定ファイル dependency.txt が必要になる．

4. **Dependency Graph 設定ファイル dependency.txt**
   各反応に関して，発生時の修正反応番号を列挙したファイルである．図 4.5 に反応記述フォーマットを，図 4.6 に LTS の dependency.txt を示す.

```
[修正反応の数] {[反応番号]...}
```

図 4.5: Dependency Graph 設定ファイルのフォーマット

```
4           # 生化学システムに定義された反応数
2    2 3    # 反応 R1 に関する記述
3    1 3 4  # 反応 R2 に関する記述
2    1 2    # 反応 R3 に関する記述
1    2      # 反応 R4 に関する記述
```

図 4.6: LTS の Dependency Graph 設定ファイル

表 4.1: SSA-SW の実行環境

CPU	Xeon 2.80GHz
Memory	4.0 GB
OS	Linux 2.4.31
Compiler	g++ 3.3.5 (-O3)

4.1.2 SSA の実行時間の比較

FRM，DM，NRM の各 SSA-SW について，ソフトウェアでの計算速度を比較した．実行環境を表 4.1 に示す．

2.3.3 節で述べた Linear Chain System(LCS) について，反応の数 N を 1 から 1000 まで変化させたときの，各 SSA の反応規模と計算時間の関係を表 4.2 および図 4.7 に示す．計算時間は反応が 10 万回発生するまでに掛かった時間を RDTSC 命令を用いて計測している．

ソフトウェアの計算時間から，各 SSA を用いた場合のスループットを表 4.3 および図 4.8 に示す．スループットを求めるために以下の計算式を用いた．

$$\text{Througput [Mcycle/sec]} = \frac{0.1\ \text{[Mcycle]}}{\text{Execution Time [sec]}} \tag{4.1}$$

表 4.2: SSA-SW の計算時間の比較 (LCS)

Algorithm	FRM	DM	NRM
System scale$N$	[sec]	[sec]	[sec]
10	0.422	0.102	0.081
50	1.470	0.361	0.105
100	2.767	0.653	0.114
500	13.078	3.161	0.136
1000	26.392	6.245	0.143

表 4.3: SSA-SW のスループットの比較 (LCS)

Algorithm	FRM	DM	NRM
N	[Mcycle/sec]	[Mcycle/sec]	[Mcycle/sec]
10	0.237	0.980	1.235
50	0.068	0.277	0.952
100	0.036	0.153	0.877
500	0.008	0.032	0.735
1000	0.004	0.016	0.699

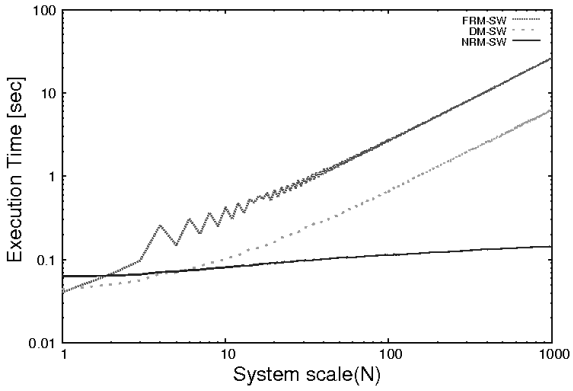


図 4.7: SSA-SW の計算時間の比較 (LCS)

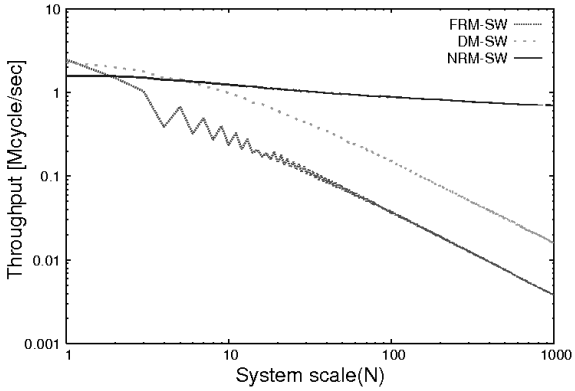


図 4.8: SSA-SW のスループットの比較 (LCS)

4.2 FRM-FPGA

4.2.1 面積評価

実装した FRM-FPGA を Xilinx ISE7.1i で合成，配置配線を行った結果を表 4.4 に示す．対象デバイスは ReCSiP2 ボード搭載 FPGA である Virtex-II PRO(XC2VP70 FF1517-5) を選択した．FRM-UNIT は演算ユニット (FU)，データユニット (DU)，コントローラユニット (CU) を各 1 つずつ持つ FRM シミュレータ回路である．FRM-UNIT はデータセットを 2 つ持つ DU を使用して 2 並行で計算を実行する．また，FPGA の面積に余裕がある限り，FRM-UNIT を複数持つことが可能であるため，XC2VP70 上に 3 つの FRM-UNIT を構成した場合 (FRM-FPGA) の評価を示している．FRM-FPGA は 2 並行 3 並列で合計 6 回分のシミュレーションが実行される．

4.2.2 性能評価

FRM-FPGA の計算能力について検討する．

LTS で反応が 10 万回発生するまでの分子数の変化について，図 4.9，図 4.10 に FRM-FPGA を用いた RTL シミュレーションの実行結果，図 4.11 にソフトウェアによる実行結果をそれぞれ示す．

DS_A と DS_B では，共通の乱数生成モジュールを使用するが，図 4.9，図 4.10 に示されたように使用する乱数が異なるため，分子数の変化は異なる軌道を描く．

解析モデルシミュレーションでは，X_2，X_3 は定常状態として記述されるため，$X_2 = 1000$, $X_3 = 1000$ の定数になってしまう．確率モデルでは，X_2，X_3 が振動する様子を観測することが可

表 4.4: FRM 実行回路の面積評価

	Slices$^{\alpha}$	Mult.	Block RAM	Freq. [MHz]
FRM-FPGA	25082	78	54	87.71
FRM-UNIT	8567	26	18	90.95
FU	7639	26	8	100.22
DU	989	-	8	106.83
CU	54	-	2	135.61

$^{\alpha}$ XC2VP70-5 : 33088 slices,
328 Multipliers and 328 BlockRAMs

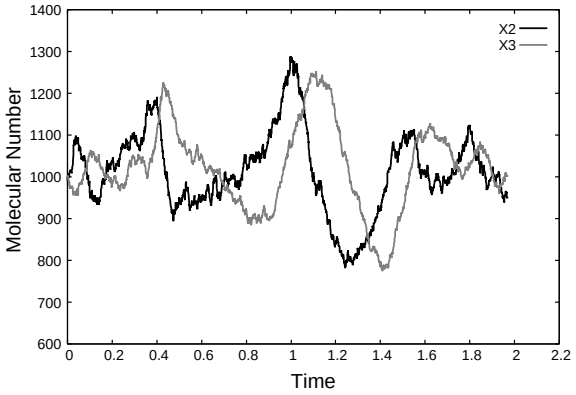


図 4.9: FRM-UNIT による LTS の出力 (DS_A)

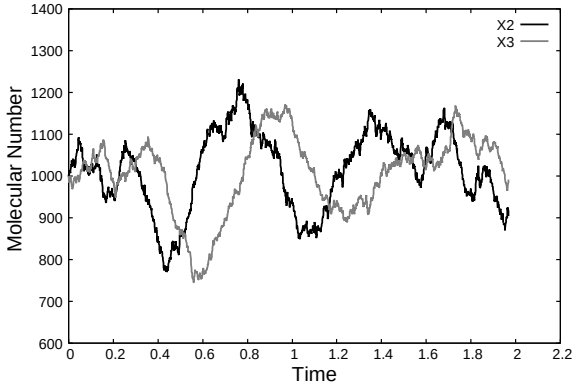


図 4.10: FRM-UNIT による LTS の出力 (DS_B)

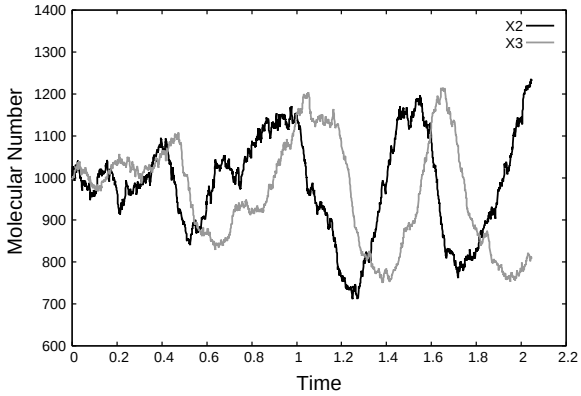


図 4.11: C++プログラムによる LTS の出力

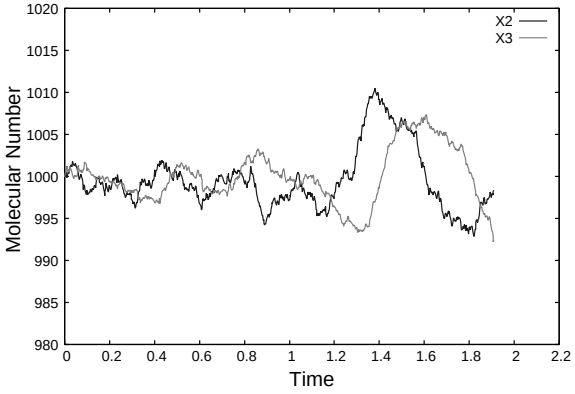


図 4.12: FRM-FPGA1000 回の実行結果の平均

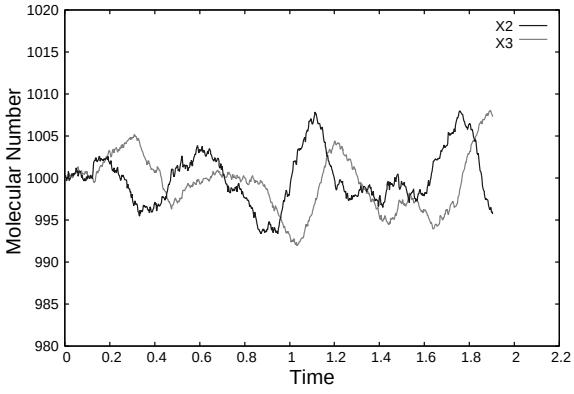


図 4.13: FRM-SW1000 回の実行結果の平均

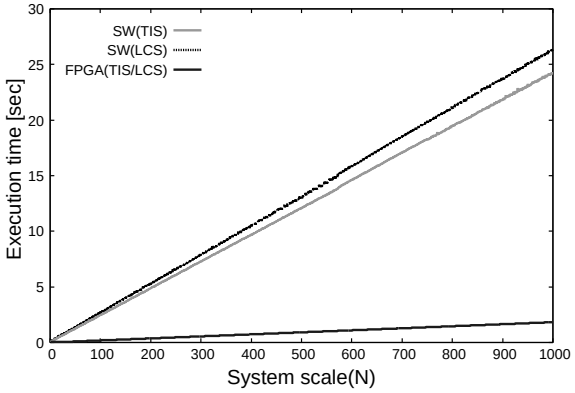


図 4.14: 計算時間の評価

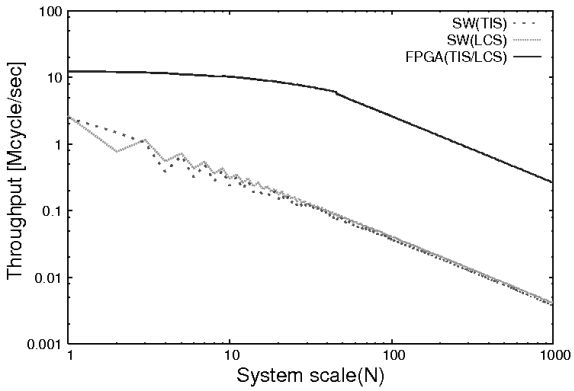


図 4.15: スループットの評価

能である．確率モデルは 1 回の実行ではシステムの挙動の一般性を得られないため，シミュレーションの精度を厳密に測定することは困難である．FRM-FPGA，FRM-SW を 1000 回実行した結果について，反応が 10 万回発生するまでの LCS の分子数の平均をそれぞれ図 4.12，図 4.13 に示す．図 4.12，図 4.13 から，$X_2 = 1000$，$X_3 = 1000$ を中心に $\pm 1.5\%$ の範囲で定常状態を保っており，FRM-FPGA が妥当な結果を出力していることがわかる．

図 4.14 に TIS，LCS について，N を 1 から 1000 まで増加させたときに，10 万回の反応が発生するまでに必要な計算時間の変化を示す．FRM-FPGA は，1 回のシミュレーション実行時間で，同時に 6 回分の計算が行われるため，単位時間あたりのスループットは 6 倍になる．ソフトウェアおよび FRM-FPGA のスループットについて，システム規模との関係を図 4.15 に示す．

図 4.17，図 4.16 にソフトウェア実行と比較したときの FRM-FPGA の 1 回のシミュレーション計算時間およびスループットの向上率を示す．表 4.5 に，計算時間とスループットに関する評価を示す．

FRM は全反応の τ_j を毎サイクル計算する必要があるため，反応の規模 N に比例した計算時間が必要になる．LCS は TIS に比べ，propencity の計算に関わる乗算の回数が 1 回多いため，計算時間が長くなっている．FRM-UNIT は，propencity を計算時間固定の FU を使用して求めるため，計算時間は定義された反応の数 N に比例し，反応の形には依存しない．FRM-UNIT は，システム規模が大きくなるにつれて各 DS が FU を占有する時間が長くなるため，スループットが下落する．しかし，τ_j の計算をパイプライン化した FU で行い，FU に間断無くデータを入力して τ_j の

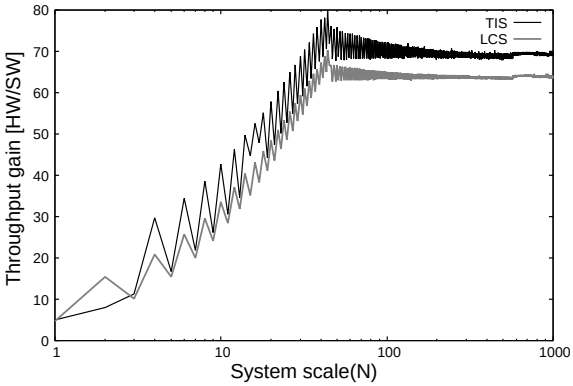


図 4.16: スループットの向上率

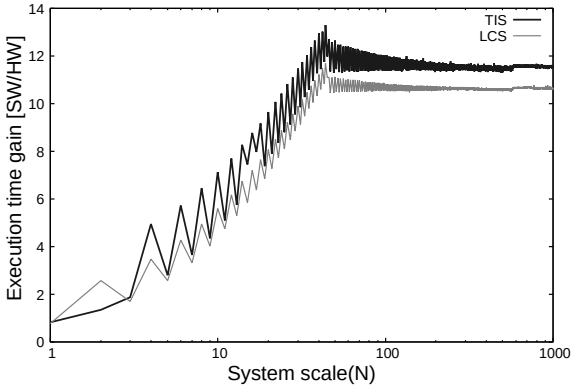


図 4.17: 計算時間の向上率

表 4.5: 計算時間とスループットの評価

System	N	計算時間 [sec]		速度向上率 (SW/HW)
		SW	HW	
TIS	10	0.332	0.059	5.60
	100	2.493	0.230	10.83
	1000	24.212	2.283	10.61
LCS	10	0.422	0.059	7.11
	100	2.767	0.230	12.01
	1000	26.392	2.283	11.56
LTS	4	0.118	0.052	2.27

System	N	スループット [Mcycle/sec]		スループット向上率 (HW/SW)
		SW	HW	
TIS	10	0.301	10.120	33.58
	100	0.041	2.605	63.25
	1000	0.004	0.263	63.64
LCS	10	0.237	10.120	42.69
	100	0.038	2.605	68.20
	1000	0.004	0.263	69.38
LTS	4	0.847	11.440	13.51

表 4.6: 面積・動作周波数の評価

Modules	Slices$^{\alpha}$ [%]	18x18 Mult.$^{\alpha}$	Block RAM$^{\alpha}$	Freq. [MHz]
DU	1180(3.57)	-	8	81.65
TC-FU	7055(21.32)	13	6	100.22
TR-FU	3700(11.18)	4	-	102.84
PF-FU	1274(3.85)	13	2	134.43
UM-FU	58(0.18)	-	7	102.28
DG-FU	44(0.13)	-	6	144.20
RT-FU	7(0.02)	-	2	190.15
SW-5port	159(0.48)	-	-	166.53
NRM-DU1	12086(36.53)	31	30	81.65

$^{\alpha}$ Capacity(XC2VP70) : 33088 slices, 328 BlockRAM and 328 Multipliers

計算に掛かる時間を隠蔽することで，大規模生化学システムの計算においても，ソフトウェア実行と比較して高い性能向上率を維持している．

TIS は定義可能な生化学システムの中で，もっとも単純な反応で構成された生化学システムである．他の生化学システムは，規模 N が同じであれば，ソフトウェアでのスループットは TIS よりも低くなる．そのため，図 4.17 から，FRM-FPGA は $N \geq 46$ 以上の生化学システムにおいて単一の計算速度で約 10 倍，図 4.16 から，スループットは約 60 倍の高速化が実現できたと言える．

4.3 NRM-DU1

4.3.1 面積評価

Verilog-HDL を用いて実装を行い，Xilinx ISE7.1i を用いて Virtex-IIPRO の BlockRAM，組み込み乗算器などを生成して使用している．また，Xilinx ISE7.1i を用いて合成，配置配線を行った．ターゲットデバイスとして Virtex-IIPRO XC2VP70 FF1517-5 を選択している．

データ転送網で接続された FPGA による SSA-NRM シミュレータの予備評価として，図 3.8 のような DU を 1 つだけ持つシミュレータ (NRM-DU1) を実装した．その面積および動作周波数に関する評価を表 4.6 に示す．評価に用いられた FU は，データが衝突しない前提で実装された簡単なコントローラのみを保持している．

DU は約 1180 slice，ルータは約 159 slice であるため，DU を 15 個程度搭載した SSA-NRM シミュレータが構築できると考えられる．各 FU のパイプラインの占有率は 2 %に満たないため，FU は各 1 個あれば十分に対応できる．

4.3.2 性能評価

アルゴリズムの挙動を詳細に解析するために，NRM を実行するソフトウェアプログラムを C++ で実装し，2.3.3 節で定義した 3 種類の生化学ベンチマークを用いて速度の評価を行った．

表 4.7: NRM-DU1 のスループットの評価

System	N	D	SW	HW$^{\nabla}$	HW/SW
LTS	4	2	1.01	0.29	0.29
TIS	4	0	1.94	0.95	0.49
	10	0	1.64	0.93	0.57
	100	0	1.10	0.88	0.80
	1000	0	0.74	0.84	1.13
LCS	4	1	1.11	0.47	0.42
	10	1	0.89	0.46	0.51
	100	1	0.77	0.43	0.56
	1000	1	0.60	0.41	0.69

$^{\nabla}$ Operating at 81.65MHz
$^{\nabla}$ 単位は [Mcycles/sec]

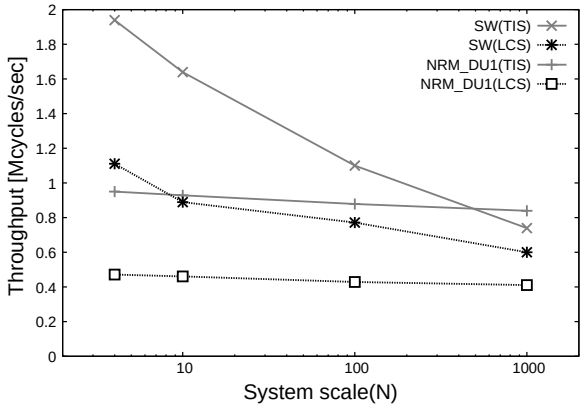


図 4.18: NRM-DU1 のスループットの評価

LTS および N の値を変えたときの TIS，LCS を対象の生化学システムとして，汎用プロセッサ，NRM-DU1 のスループットに関する評価を表 4.7 に示す．N は生化学システムで定義された反応数，D は 1 サイクルあたりの τ_j の平均修正回数である．

LTS のような小規模な生化学システムの場合，FPGA による高速実行の効果が得られていない．TIS と LCS の結果から，D の値が大きいほど，NRM-DU1 の演算性能が低くなることがわかる．これは，サイクルごとに τ_j の修正 (3.2.3 節の STEP5-7) が平均 D 回実行されるためである．τ_j の修正は依存関係が木の更新のみのため，DG からのデータの受信や TR-FU への計算要求は D 回分連続して実行する方法にすることで，スループットの向上が可能である．

また，生化学システムが大規模になるほど，ソフトウェアのスループットの下落が大きいことがわかる．この理由として，木の再構成時間が汎用プロセッサに比べて高速であることが挙げられる．TIS のように各反応が他の反応に全く影響を及ぼさない特殊なシステムを除いて，単一の DU の計算速度は汎用プロセッサに劣るが，複数の DU を配置する回路面積の余裕があることから，10 個-15 個程度の DU の並列実行により，大きな演算能力を得られると期待できる．

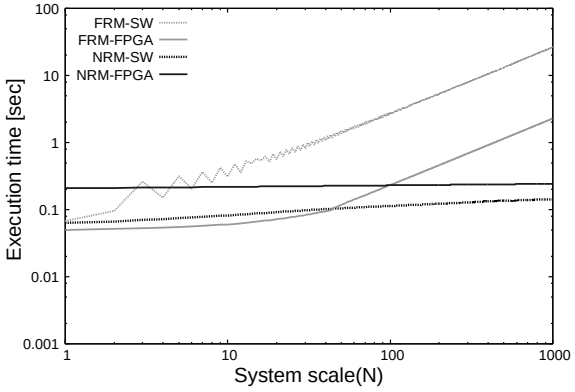


図 4.19: 計算時間の評価

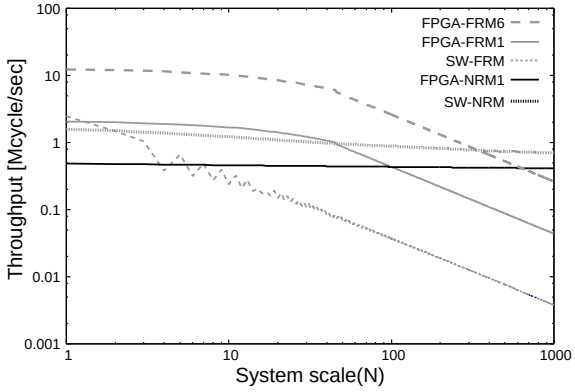


図 4.20: スループットの評価

4.4 SSA-FPGA に関する考察

図 4.19 に，実装した 2 種類の SSA-FPGA(FRM-FPGA，NRM-FPGA) と FRM-SW について，規模 N を変えた LCS で，10 万回反応が発生するまでの計算時間の比較を示す．また，図 4.20 に，SSA-FPGA と FRM-SW のスループットの比較を示す．

4.4.1 FRM-FPGA に関する考察

図 4.19 から，FRM-SW はアルゴリズムの性質通り，計算時間が生化学システムの規模 N(定義された反応の数) に比例して増加していることがわかる．特に，FRM は N 回の乱数生成と τ_j の計算を毎サイクル行うため，実行時間の増加は非常に大きい．FRM-FPGA も，浮動小数点演算の隠蔽が可能であるとは言え，1 クロックで使用できる演算器が固定であるため，N の増加とともに計算時間が増加している．FRM-SW，FRM-FPGA ともに計算量は $O(N)$ である．Keane らの研究は，反応規模が大きくなった場合は，並列実行する演算ユニットを増やすため，NRM-SW と比較して 20 倍程度の速度向上を維持している [65]．しかし，Keane らの方法では回路規模の増大に対する問題と，近似アルゴリズムの問題を回避することができない．

そこで，本論文での実装である FRM-FPGA の回路規模と計算時間のトレードオフを考える．FRM を近似せずに実行する以上，回路を増大させずに計算量を削減することは不可能である．FRM-FPGA が 2 回分のシミュレーションを 3 セット並列に実行していることから，計算時間を削減するために，まず各 FRM-UNIT が 1 回分のシミュレーションだけを実行する方法が考えられる．2 回分のシミュレーションは，生化学システム定義で最後に定義されている反応の τ_N に関する分子番号が読み出されてから，μ が求められ，分子数が更新されるまでの 46 クロックを隠蔽することを目的として採用した方法である．FRM-UNIT の計算時間は以下の式で表される．

$$1\text{サイクルの計算時間 [clock]} \begin{cases} N + 46 (N < 46) \\ 2N + 2 (N \geq 47) \end{cases} \tag{4.2}$$

図 4.19 からもわかるとおり，$N < 46$ の生化学システムであれば計算時間の増加は少ないが，$N \geq 47$ からは計算時間の増加が大きくなる．大規模な生化学システムにおいては，1 つの FRM-UNIT は 1 回分のシミュレーションを実行することで，シミュレーションの実行時間を短縮することができる．この際，FRM-FPGA 全体のスループットが半減し，1 回分を実装した場合は，2

回分実行した場合よりもスループットが下落するため，1 回のシミュレーションの計算時間とスループットの兼ね合いから実行数が選択されることになる．小規模な生化学システムでは汎用プロセッサでも高速な演算が可能であるため，FRM-UNIT のシミュレーション実行数をさらに増やす方法は効果が薄い．N の増加とともに各 DS が FU を占有する時間が長くなるため，3 個以上の DS の並行実行を行っても，N が 23 程度で単位時間当たりのスループットは 2 並行の場合と同一になる．そのため，FRM-UNIT は 2 並行での実行が $N \geq 47$ の生化学システムで最適な構成であると言える．

図 4.7 で評価されたとおり，FRM，DM はアルゴリズムの性質上，計算量は $O(N)$ である．また，NRM もアルゴリズムの性質上，計算量は $O(\log(N))$ であることがわかる．FRM と NRM は統計的に等価なアルゴリズムと言われていることから，FRM-FPGA と NRM-SW に関して比較を行う．

図 4.20 を見ると，計算量の違いから，FRM-FPGA は 6 回分のシミュレーションを同時に実行するにもかかわらず，$N = 700$ 程度で NRM よりもスループットが小さくなってしまっていることがわかる．単一のシミュレーションの実行時間では $N = 50$ で計算時間は同一になり，その後は FRM での実行の方が遅くなる．FRM-FPGA の計算時間を短くするためには並行実行数を 1 つにすることを述べたが，さらに，複数の FRM-UNIT を用いて，単一のシミュレーションを実行する方法が考えられる．FRM-FPGA は 3 つの FRM-UNIT を持っているため，FRM-UNIT を同時に使用し，同時に 3 反応ずつ τ_j を計算する．並行実行を 1 つにする方法を同時に用いることにより，複数回分のシミュレーションの並列実行のメリットが無くなる代わりに，計算時間は約 6 分の 1 に短縮することが可能である．この方法により，反応 N が数百種類定義された生化学システムまで，NRM-SW よりも高速なシミュレーションが可能になると考えられる．

4.4.2 NRM-FPGA に関する考察

FRM-FPGA では N に比例する計算時間が要求されるため，大規模な生化学システムでは NRM-SW よりも計算時間が長くなってしまう，という問題が存在している．NRM-FPGA に関する評価は，DU を 1 つ持つ NRM-DU1 のみであるが，NRM-FPGA と NRM-SW に関して比較と検討を行う．

図 4.19 や図 4.18 を見ると，計算時間の増加が NRM-SW に比べて小さく，大規模生化学システムに関して効果が得られることが予想される．シミュレーション実行時間の点では，NRM-SW よりも長い時間が掛かっているが，10 個程度の DU を用いて並列実行が可能であることから，NRM-SW よりも大きなスループットが得られると考えられる．図 4.18 を見ると，NRM-SW は N が大きいときに，TIS($D = 0$) に比べて LCS($D = 1$) のスループットの下落が大きい．NRM-FPGA も D の値が増えると計算時間が大きく増加するが，4.3.2 節で言及した DG 読出-τ_j 修正の部分の連続実行により，D が大きい場合の計算時間の増加を抑えることが可能である．D は生化学システムの複雑度を示す尺度であり，実際のシミュレーションに用いられる生化学システムは D がより大きいと考えられる．Gibson らは，λ ファージモデルが $D = 4.2$ であると述べており [11]，D が大きい場合に NRM-FPGA は有利である．

問題として，NRM-DU1 が，DU が複数になったときのネットワークの混雑を考慮していないことが挙げられる．6 種類の FU を持ち，多数の DU がそれぞれの FU に通信を掛ける関係上，通信に掛かる時間は大きな問題になると考えられる．データの通信方法に関する問題を考える上で，FU の種類は少ないことが望まれる．表 4.6 から，UM-FU，DG-FU，RT-FU のデータテーブルとしての FU は面積が小さいため，各 DU が保持する方法が考えられる．シミュレーション中の定数テー

ブル参照を各 FU が保持することで，ネットワーク上に転送されるデータ量を抑えることが可能になる．Virtex-II Pro の BlockRAM の使用量は大きくないため，10 個程度の DU を持つ NRM-FPGA を構成することは可能であると考えられる．

DU を複数持つ NRM-FPGA の実現には，データ転送方法について検討が必要である．アドレス併走方式，かつ 58 ビットで構成されるデータフリットはルータの面積の点で効率が悪い．現在のルータは最小限の機能しか持たないため，面積が小さくなっているが，衝突検出やバーストモード (バスの占有) などの機能を追加した場合，面積が大きくなる．ネットワークの混雑と併せて，問題の性質に合わせて NRM-FPGA の構成を考えるために，これらの検討を行う必要がある．

第 5 章

結論

本研究では，確率モデル生化学シミュレーション (SSA) の高速実行環境を，FPGA を用いて低コストで提供することを目的に，FRM，NRM と呼ばれる 2 種類の SSA をそれぞれハードウェア処理する回路を実装した．

FRM-FPGA では，パイプライン化された演算ユニットに連続的にデータを投入することにより，浮動小数点演算に掛かる時間を隠蔽し，さらに複数回分のシミュレーションの並列実行により，Xeon 2.80GHz を用いた FRM のソフトウェア実行と比較して，約 60 倍のスループットを実現した．

NRM-FPGA では，高効率なアルゴリズムである NRM の計算を，近似を用いることなく FPGA 上で高スループットに実現するためにネットワーク状に演算ユニット，データユニットを接続する方法を提案した．FPGA 上にネットワークを構築して様々な問題に柔軟に対応できる高速アプリケーション実行環境の有効性を示すために，データユニットを 1 つだけ持つ NRM-DU1 を実装し，その初期評価を行った．その結果，Xeon 2.80GHz を用いた NRM のソフトウェア実行と比較して，約 0.5 倍から同程度の計算速度が得られることがわかり，DU の並列実行による高スループットな演算環境の可能性を示すことができた．

今後の展開については，

- NRM の通信パターンの詳細な解析
- 省面積，高効率なルータおよび各ユニットの通信機構
- 複数 DU を用いた高スループット NRM-FPGA の開発
- FPGA を用いたネットワークベースの他のアプリケーション実行環境への適用の模索

などの検討と実装が挙げられる．

謝辞

本研究の機会を与えてくださり，また常にご指導，ご助言をいただきました
天野 英晴 教授　に，深く感謝いたします．

研究を進めるにあたり，システムバイオロジーの見地からの多くの有益な知識を頂きました，独立行政法人 科学技術振興機構 北野共生システムプロジェクト
舟橋 啓氏　広井 賀子氏　に感謝いたします．

浮動小数点演算器をはじめ研究について助言，論文の査読をおこなってくださった，
長崎大学　柴田 裕一郎 講師，　岩永 直樹氏　に感謝致します．

研究グループ配属時から多くのアドバイスをいただき，また論文のチェックを行ってくださった
長名 保範氏　に感謝いたします．氏の研究に関する様々な考え方は現在の私に大きな影響を与えています．

設計の良否，実装の妥当性について多くの助言をいただき，論文の査読を行ってくださった岩岡 洋氏　に感謝いたします．効果的な実装，評価に関して，氏の協力により説得力を持たせることができました．

本研究を含めた FPGA を用いた生化学シミュレータの研究に協力いただきました
西川 由理氏　小嶋 利紀氏　に感謝いたします．彼らの研究，実装に対する真摯な姿勢には，とても勇気づけられました．

研究グループの皆様は私の誇りです．

日々の研究に際し，多くの助言と，生活のアドバイスをいただいた
天野研究室の全てのみなさま　に感謝します．

大切な家族から，祖母の　川上 榮子氏　に感謝します．
私の日々の生活は彼女の支援により成り立っています．

私がここまで来られたのは全て皆様のおかげです．今まで私と関わってくださった全ての皆様に限りない感謝を．

ありがとうございました．

参考文献

[1] Bruce A. Barshop, Richard F. Wrenn, and Carl Frieden. Analysis of numerical methods for computer simulation of kinetic processes: Development of kinsim — a flexible, portable system. *Analytical Biochemistry*, Vol. 130, pp. 134–145, 1983.

[2] Mendes Pedro. Gepasi: a software package for modelling the dynamics, steady states and control of biochemical and other systems. *Computer Applications in the Biosciences*, Vol. 9, No. 5, pp. 563–571, Oct. 1993.

[3] Igor Goryanin, T. Charles Hodgman, and Evgeni Selkov. Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, Vol. 15, No. 9, pp. 749–758, Sep. 1999.

[4] Masaru Tomita, Kenta Hashimoto, Kouichi Takahashi, Thomas Simon Shimizu, Yuri Matsuzaki, Fumihiko Miyoshi, Kanako Saito, Sakura Tanida, Katsuyuki Yugi, J. Craig Venter, and Clyde A. HutchisonIII. E-cell: software environment for whole-cell simulation. *Bioinformatics*, Vol. 15, No. 1, pp. 72–84, Jan. 1999.

[5] James Schaff, Charles C. Fink, Boris Slepchenko, John H. Carson, and Leslie M. Loew. A general computational framework for modeling cellular struc ture and function. *Biophysical Journal*, Vol. 73, pp. 1135–1146, Sep. 1997.

[6] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, Vol. 81, No. 25, pp. 2340–2461, Dec. 1977.

[7] Andrzej M. Kierzek. Stocks: Stochastic kinetic simulations of biochemical systems with gillespie algorithm. *Bioinformatics*, Vol. 18, No. 3, pp. 470–481, Mar. 2002.

[8] Nicolas Le Novère and Thomas Simon Shimizu. Stochsim: modelling of stochastic biomolecular processes. *Bioinformatics*, Vol. 17, No. 6, pp. 575–576, Jun. 2001.

[9] Yasunori Osana, Tomonori Fukushima, Masato Yoshimi, and Hideharu Amano. An FPGA-based acceleration method for metabolic simulation. *IEICE Trans. on Information and Systems*, Vol. E87-D, No. 8, pp. 2029–2037, Aug. 2004.

[10] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, Vol. 22, pp. 403–434, 1976.

[11] Michael A.Gibson and Jehoshua Bruck. 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.

[12] Hiroaki Kitano. Prespectives on systems biology. *New Generation Computing*, Vol. 18, No. 1, pp. 199–216, 2000.

[13] Crick FH Watoson JD. Molecular structure of nucleic acids: a structure for deoxiribose nucleic acid. *Nature*, Vol. 171, pp. 737–738, 1953.

[14] 北野宏明. システムバイオロジー –生命をシステムとして理解する–. 秀潤社, Jul. 2001.

[15] D.Fell. *Understanding the Control of Metabolism*. Portland Press, 1997.

[16] L.Michaelis and Maud L.Menten. Die kinetik der invertinwirkung. *Biochemische Zeitschrift*, Vol. 49, pp. 333–369, 1913.

[17] Albert Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. In *Proceedings of the National Academy of Sciences*, Vol. 88, pp. 9107–9111, Oct. 1991.

[18] M. Hucka, A. Finney, B.J. Bornstein, S.M. Keating, B.E. Shapiro, J. Matthews, B.L. Kovitz, M.J. Schilstra, A. Funahashi, J.C. Doyle, and H. Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: The systems biology markup language (sbml) project. *IEE Systems Biology*, Vol. 1, No. 1, pp. 41–53, 2004.

[19] Catherine M. Lloyd, Matt D.B. Halstead, and Poul F. Nielsen. CellML: its future, present and past. *Progress in Biophysics and Molecular Biology*, Vol. 85, pp. 433–450, Jun 2004.

[20] Akira Funahashi, Naoki Tanimura, Mineo Morohashi, and Hiroaki Kitano. CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, Vol. 1, No. 5, pp. 159–162, Nov. 2003.

[21] The Systems Biology Institute. Celldesigner.org. `http://celldesigner.org/`.

[22] Computational Systems Biology Group at Keck Graduate Institute. JDesigner. `http://sbw.kgi.edu/`.

[23] InNetics AB. Pathwaylab. `http://innetics.com/`.

[24] H.M.Sauro. Jarnac: a system for interactive metabolic analysis. In *The 9th International Bio-ThermoKinetics Meeting*, pp. 221–228. Stellenbosch University Press, 2000.

[25] Copasi. `http://www.copasi.org/`.

[26] Bruce E. Shapiro, Michael Hucka, Andrew Finney, and John Doyle. MathSBML: a package for manipulating SBML-based biological models. *Bioinformatics*, Vol. 20, No. 16, pp. 2829–2831, Apr. 2004.

[27] H. Sauro, M. Hucka, A. Finny, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano. Next generation simulation tools: The systems biology workbench and biospice integration. *Omics*, Vol. 7, No. 4, pp. 355–572, Dec. 2003.

[28] Kegg: Kyoto encyclopedia of genes and genomes. `http://www.genome.ad.jp/kegg/`.

[29] M.Kanehisa, S.Goto, S.Kawashima, and A.Nakaya. The kegg databases at genomenet. *Nucleic Acids Research*, Vol. 30, No. 1, pp. 42–46, Jan. 2002.

[30] Biomodels. `http://www.biomodels.net/`.

[31] Akiya Jouraku, Akira Funahashi, and Hiroaki Kitano. KEGG2SBML. `http://www.sbml.org/kegg2sbml.html`.

[32] Nicolas Le Novere, Andrew Finney, Michael Hucka, Upinder S Bhalla, Fabien Campagne, Julio Collado-Vides, Edmund J Crampin, Matt Halstead, Edda Klipp, Pedro Mendes, Poul Nielsen, Herbert Sauro, Bruce Shapiro, Jacky L Snoep, Hugh D Spence, and Barry L Wanner. Minimum information requested in the annotation of biochemical models (miriam). *Nature Biotechnology*, Vol. 23, pp. 1509–1515, Dec. 2005.

[33] Wormbase. `http://www.wormbase.org/`.

[34] Peter D. Karp, *et al.* The ecocyc database. *Nucleic Acids Research*, Vol. 30, No. 1, pp. 56–58, Jan. 2002.

[35] Yang Cao, Hong Li, and Linda Petzold. Efficient formulation of the stochastic simulation algorithm for chemically recting systems. *Journal of Chemical Physics*, Vol. 121, No. 9, pp. 4059–4067, 2004.

[36] Daniel T. Gillespie. Approximate acclerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, Vol. 115, No. 4, pp. 1717–1733, Jul. 2001.

[37] Kouichi Takahashi, Katsuyuki Yugi, Kenta Hashimoto, Yohei Yamada, Christopher J. F. Pickett, and Masaru Tomita. A multi-algorithm, multi-timescale method for cell simulation. *Bioinformatics*, Vol. 20, No. 4, pp. 538–546, Mar. 2004.

[38] Andrzej M. Kierzek, Jolanta Zaim, and Piotr Zielenkiewicz. The effect of transcription and translation initiation frequencies on the stochastic fluctuations in prokaryotic gene expression. *The Journal of Biological Chemistry*, Vol. 276, No. 11, pp. 8165–8172, Mar 2001.

[39] Adam Arkin, John Ross, and Harley H.McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected escherichia coli cells. *Genetics*, Vol. 149, pp. 1633–1648, Aug. 1998.

[40] G. Marsaglia, A. Zaman, and WW Tsang. Toward a universal random number generator. *Letters in Statistics and Probability*, Vol. 9, No. 1, pp. 35–39, Jan. 1990.

[41] Stephen D. Brown, Robert J. Francis, Jonathan Rose, and Zvonko G. Vranesic. *File-Programmable Gate Arrays*. Kluwer Academic Publishers, 1992.

[42] Vaughn Betz, Jonathan Rose, and Alexander Marquardt. *Architecture and CAD for Deep-submicron FPGAs*. Kluwer Academic Publishers, 1999.

[43] Xilinx Inc. *Virtex-II Platform FPGA User Guide*, v2.0 edition, Mar. 2005.

[44] Xilinx Inc. *Virtex-II Pro and Virtex-II Pro X FPGA User Guide*, v4.0 edition, Mar. 2005.

[45] Atsushi Kawai, Toshiyuki Fukushige, Jun ichiro Makino, and Makoto Taiji. Grape-5: A special-purpose computer for n-body simulations. *Publ. of the Astronomical Society of Japan*, Vol. 52, pp. 659–676, Aug. 2000.

[46] Jürgen Becker and Reiner Hartenstein. Configware and morphware going mainstream. *Journal of System Architecture*, Vol. 49, No. 4-6, pp. 127–142, 2003.

[47] Cray Inc. Cray XD-1 supercomputer. `http://www.cray.com/products/xd1/index.html`.

[48] Tsuyoshi Hamada and Naohiro Nakasato. Pgr: A software package for reconfigurable super-computing. In *Proceedings of the 15th International Conference on Field Programmable Logic and Applications*, pp. 366–373, Aug. 2005.

[49] SRC Computers Inc. SRC-7 reconfigurable general purpose computing system. `http://www.srccomp.com/`.

[50] Naohito Nakasato and Tsuyoshi Hamada. Astrophysical hydrodynamics simulations on a reconfigurable system. In *Proceedings of the 13th IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 279–280, Apr. 2005.

[51] Tsuyoshi Hamada, Naohito Nakasato, and Toshikazu Ebisuzaki. A 236 gflops astrophysical simulation on a reconfigurable super-computer. In *Proceedings of IEEE/ACM SC 2005 Conference*, Nov. 2005.

[52] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, Vol. 147, pp. 195–197, 1981.

[53] R. Durbin, S. Eddy, A. Korgh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[54] C.W.Yu, K.H.Kwong, K.H.Lee, and P.H.W. Leong. A smith-waterman systolic cell. In *Proceedings of the 14th International Conference on Field-Programmable Logic and Applications*, Vol. 2778 of *Lecture notes in Computer Science*, pp. 375–384. Springer Verlag, Sep. 2003.

[55] Jeffrey M. Arnold, Duncan A. Buell, and Elaine G. Davis. Splash 2. In *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, pp. 316–322. ACM Press, 1992.

[56] Annapolis Micro Systems Inc. Wildfire. `http://www.annapmicro.com/`.

[57] Chen Chang, John Wawrzynek, and Robert W. Brodersen. Bee2: a high-end reconfigurable computing system. *Design & Test of Computers*, Vol. 22, No. 2, pp. 114–125, Mar. 2005.

[58] Yoshiki Yamaguchi, Tsutomu Maruyama, and Akihiko Konagawa. Three-dimensional dynamic programming for homology search. In *Proceedings of the 14th International Conference on Field-Programmable Logic and Applications*, Vol. 3203 of *Lecture notes in Computer Science*, pp. 505–515. Springer Verlag, Aug. 2004.

[59] Y.Komeiji, H.Yokohama, M.Uebayashi, M.Taiji, T.Fukushige, D.Sugimoto, R.Takata, A.Shimizu, and K.Itsukashi. A high performance system for molecular dynamics simulation of biomolecules using a special-purpose computer. In *Proceedings of Pacific Symposium on Biocomputing*, pp. 472–487, Jan 1996.

[60] Yongfeng Gu, Tom Van Court, and Martin Herbordt. Accelerating molecular dynamics simulations with reconfigurable circuits. In *Proceedings of the 15th International Conference on Field-Programmable Logic and Applications*, pp. 475–480, Aug. 2005.

[61] S.Onami, S.Hamahashi, M.Nagasaki, S.Miyano, and H.Kitano. chapter Automatic acquisition of cell lineage through 4D microscope and analysis of early *C. elegans* embryogenesis, pp. 33–55. Foundations of Systems Biology. MIT Press, 2001.

[62] 福島知紀. FPGA による線虫 *C.elegans* 初期胚の細胞系譜自動構築シス テムの高速化. Master's thesis, 慶應義塾大学大学院理工学研究科, 神奈川県横浜市, Mar. 2005.

[63] Larry Lok. The need for speed in stochastic simulation. *Nature Biotechnology*, Vol. 22, No. 8, pp. 964–965, Aug. 2004.

[64] Lukasz Salwinski and David Eisenberg. In silico simulation of biological network dynamics. *Nature Biotechnology*, Vol. 22, No. 8, pp. 1017–1019, Aug. 2004.

[65] John F. Keane, Christopher Bradley, and Carl Ebeling. A compiled accelerator for biological cell signaling simulations. In *The 12th Int. Symp. on Field-Programmable Gate Arrays(FPGA)*, pp. 233–241, Feb. 2004.

[66] Yasunori Osana, Tomonori Fukushima, and Hideharu Amano. Implementation of ReCSiP: a reconfigurable cell simulation platform. In *The 13th International Conference on Field Programmable Logic and Applications*, Vol. 2778 of *Lecture Notes in Computer Science*, pp. 766–775. Springer, Sep. 2003.

[67] Masato Yoshimi, Yasunori Osana, Tomonori Fukushima, and Hideharu Amano. Stochastic simulation for biochemical reactions on FPGA. In *The 14th International Conference on Field Programmable Logic and Applications*, Vol. 3203 of *Lecture Notes in Computer Science*, pp. 105–114. Springer, Aug. 2004.

[68] 吉見真聡, 長名保範, 福島知紀, 天野英晴. 確率モデルを用いた化学反応シミュレーションの fpga によ る高速化. 第 4 回リコンフィギャラブルシステム研究会 33, 電子情報通信学会, Sep. 2004.

[69] Masato Yoshimi, Yasunori Osana, Yow Iwaoka, Akira Funahashi, Noriko Hiroi, Yuichiro Shibata, Naoki Iwanaga, Hiroaki Kitano, and Hideharu Amano. The design of scalable stochastic biochemical simulator on FPGA. In *2005 IEEE International Conference on Field Programmable Technology*, pp. 339–340. IEEE, Dec. 2005.

[70] 吉見真聡, 長名保範, 岩岡洋, 西川由理, 小嶋利紀, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴. データ転送網を用いた確率モデル生化学シミュレータの fpga への実装の検討. 第 123 回システム LSI 設計技術 (SLDM) 研究会 105, 電子情報通信学会, Jan. 2006.

[71] 吉見真聡, 長名保範, 岩岡洋, 柴田裕一郎, 岩永直樹, 天野英晴. FPGA を用いた科学技術計算向け浮動小数点算術演算器の 設計. 第 26 回研究会 1-7, パルテノン研究会, May. 2005.

[72] 吉見真聡, 長名保範, 岩岡洋, 福島知紀, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴. ヒープ木を用いた確率モデル生化学シミュレータの FPGA への実装. RECONF2005 9, 電子情報通信学会, May. 2005.

[73] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-rnadom number generator. *ACM Transactions on Modeling and Computer Simulation*, Vol. 8, No. 1, pp. 3–30, Jan. 1998.

[74] ANSI/IEEE Std 754-1985. *IEEE Standard for Binary Floating-Point Arithmetic*, 1985.

[75] 中森章. マイクロプロセッサ・アーキテクチャ入門. No. 20 in TECH I. CQ 出版社, 2004.

[76] Digital Core Design. *http://www.dcd.pl/*.

付 録

確率モデル生化学シミュレーションアルゴリズムの導出

A.1 Gillespie のアルゴリズム

ある空間上の生化学システムについて，その挙動を数学的に記述する 2 つの方法がある．

解析モデル (deterministic approach) は，分子濃度の時間変化を，微分方程式の組で表現する方法である．

また，確率モデル (stochastic approach) は，分子数の時間変化を，単一の微分差分方程式で記述されるランダムウォーク過程と見なす方法である．確率モデルの計算はマスタ方程式と呼ばれる確率・統計的なアプローチから開発された手法であるが，マスタ方程式は数学的に解答することが困難である．しかし，マスタ方程式を直接的に扱うことなく，それと等価な数値計算を行う方法を用いることで，計算機を使った計算を行うことができる．

Gillespie の開発した “Stochastic Simulation Algorithm” [6](以下 SSA) は，この確率モデルを用いて生化学システムの挙動を計算するアルゴリズムである．

SSA は次のような問題に解答することを目的として開発された．

> N 種の分子が一様に分布する一定の体積 V の中で，相互反応する M 種類の化学反応があり，ある初期時間に各分子の分子数が与えられた場合，これらの分子数は時間の経過とともにどのように変化するか．

A.1.1 解析的モデルでの解法

この問題を解くために，従来，問題を常微分方程式で表現して解析する解析モデルでのアプローチが使用されてきた．

ある時刻 t における V 内の i 番目の分子の数 (分子集合レベル．一般的には分子数など) が，時刻 t を変数とする非線形の関数 $X_i(t)$ $(i = 1, \ldots, N)$ の式で記述されている生化学システムを定義する．この生化学システムで，M 種類の各分子が連続的に変化すると見なせるならば，以下のような一次の常微分方程式を構成することができる．

$$\frac{dX_i}{dt} = f_i(X_1, \cdots, X_N) \qquad (i = 1, \cdots, N) \tag{A.1}$$

右辺の関数 f_i は，反応 M の構造と反応速度定数によって決定される．これらの方程式は，反応速度方程式 (reaction-rate equation) と呼ばれる．$X_1(t), \ldots, X_N(t)$ の変化の軌道を導くことが，各分子の時間変化になる．

この反応速度方程式の解析的な解法は，単純なシステムで利用されることが多いため，計算機でこれらの方程式を数値的に計算する．この方法は，微分方程式で近似していることから，生化学システムの時間変化が連続的かつ決定的であることを暗黙に仮定する．しかし，分子数は離散的な整数値であるから，生化学システムは時間的に非連続な状態を取る．さらに，発生する反応は決定的ではない．もし，分子の挙動がなんらかの方程式に支配されるものと見なしたとしても，システム内のすべての分子の位置と速度を計算しない限りは，システムの挙動を計算することは不可能である．

式 A.1 のような方程式を解く方法は，いくつかの生化学システムの解析には非常に良い精度を示している．しかし，解析モデルによるシミュレーションは，システムの時間挙動を微分方程式で記述するため，初期値の小さな違いが，シミュレーションの進行とともに指数関数的に拡大してしまう．また，遺伝子発現に代表されるような，ごく短い時間で少数の分子が急激に反応する場合には，反応発生にランダムな要素が含まれ，適切なシミュレーションができない．

生態系や微小な生化学システム，非線形なシステムの挙動に注目が集まっている現在においては，分子集合レベルを十分高精度に計算できる方法が必要になる．

SSA は，生化学システムは離散的であり，かつ確率的な挙動を示す，という前提から，生化学システムの挙動を計算する方法である．

A.1.2　確率的モデルによる化学反応の定式化

化学反応は，適当な 2 種類以上の分子が衝突したときに発生する．SSA は，熱平衡状態の分子反応システムで起こる分子の衝突を計算するものである．

A.1.2.1　分子の衝突

ある体積 V 内で，平衡状態にある 2 種類の気体分子 S_1 と S_2 の混合物からなるシステムがあるとする．単純化のため，S_1 分子と S_2 分子はそれぞれ半径 r_1，r_2 の球体であると考える．S_1 分子と S_2 分子の中心が重なったときに必ず反応が発生し，半径 $r_{12} = r_1 + r_2$ の分子になるとする (図 A.1).

V 内でこのような衝突が発生する頻度 (反応の速度) を計算する．

従来の反応速度を導出する方法を以下に述べる．まず，任意の分子の対から反応が起こる分子を選択する．そして，S_2 分子に関して S_1 分子の相対的な運動速度 v_{12} を求める．さらに，微小時間 Δt の間に，S_1 分子が S_2 分子に対して移動する分の体積 (Collision Volume : 衝突体積) を計算する．衝突体積 ΔV_{coll} は以下の式 A.2 で求められる (図 A.2).

$$\Delta V_{\text{coll}} = \pi r_{12}^2 \cdot v_{12} \Delta t \tag{A.2}$$

もし，時間 t で S_2 分子の中心が ΔV_{coll} 内に存在していたとすると，2 つの分子は微小時間 $(t, t+\Delta t)$ で衝突する．

この方法は，微小時間 Δt を小さくしていくことで，反応速度の精度は上がることになる．しかし，極限 $\Delta V_{\text{coll}} \to 0$ においては，衝突体積中に存在する分子は 0 か 1 になってしまう．このよ

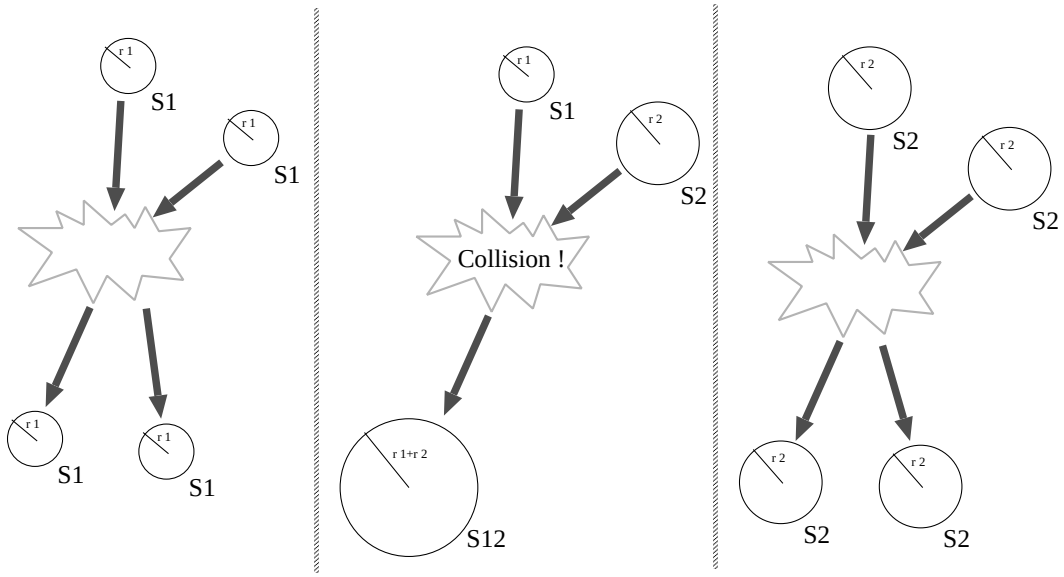


図 A.1: 分子の衝突

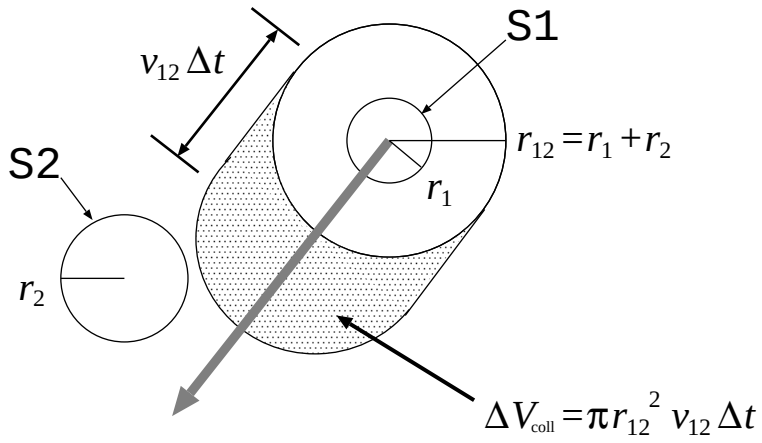


図 A.2: 衝突体積 ΔV_{coll}

うに，微小時間で計算していく方法は，厳密であるとは言い難いため，以下のように確率を用いた方法を考える．

システムは熱平衡状態であるから，分子は V 内で一様に分布している．そのため，時刻 t において任意の S_2 分子の中心が ΔV_{coll} にある確率は，単純に $\Delta V_{\text{coll}}/V$ で与えられる．よって，S_1 分子と S_2 分子の反応速度分布を平均した場合，以下のような式 A.3 が得られる．これは，システム全体の体積に対する衝突体積の割合に等しい．

$$\overline{\left(\frac{V_{\texttt{coll}}}{V}\right)} = \frac{\pi r_{12}^2 \overline{v_{12}} \Delta t}{V} = S_1\text{-}S_2 \text{ の分子が次の微小時間 } \Delta t \text{ で衝突する確率の平均} \tag{A.3}$$

マクスウェルの速度分布より，相対速度の平均 $\overline{v_{12}}$ は $(8kT/\pi m_{12})^{1/2}$ と等しい．k はボルツマン定数，T は絶対温度，m_{12} は換算質量 $m_1 m_2/(m_1+m_2)$ である．式 A.3 の計算は S_1 分子，S_2 分子が V 内にそれぞれ 1 個ずつある場合であるが，S_1 分子の数として X_1，S_2 分子の数として X_2 が与えられたとすると，式 A.3 は，式 A.4 のように表される．

$$\overline{\left(\frac{V_{\texttt{coll}}}{V}\right)} = \frac{X_1 X_2 \pi r_{12}^2 \overline{v_{12}} dt}{V} = \text{微小時間 } (t, t+dt) \text{ で } S_1\text{-}S_2 \text{ の衝突が発生する確率} \tag{A.4}$$

式 A.4 により，反応発生数を確定的には計算できないが，有限時間内に V 内で発生する衝突の確率を計算することが可能である．式 A.4 は決定的な反応速度式のかわりに，確率的なマルコフ過程を構成しており，過去のシステムの挙動には依存せず，その時点での分子数と “単位時間あたりの衝突確率” によって，熱平衡状態の分子を特徴付けるものである．

反応確率速度定数 “The Stochastic Reaction Constant”

化学反応の “反応速度” は, “適当な分子の単位時間あたりの衝突確率 = 単位時間あたりの反応発生確率” と表現することができることを前節で述べた．

V 内で S_1 分子と S_2 分子が式 A.5 のように反応すると仮定する．

$$R_1 : \quad S_1 + S_2 \rightarrow 2S_2 \tag{A.5}$$

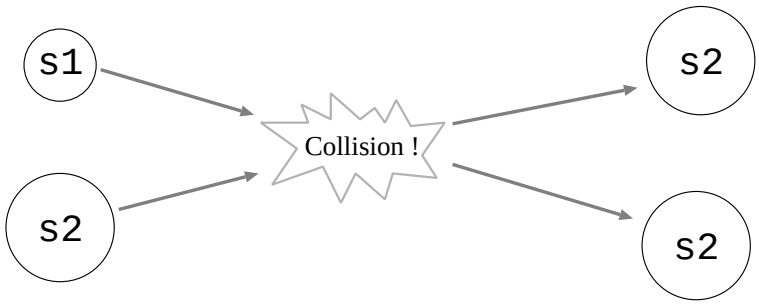


図 A.3: 仮定する反応

式 A.3 より，2 つの分子の物理的性質とシステムの温度だけに依存する定数 c_1 を設定する．

$$c_1 dt = \text{微小時間 } dt \text{ で特定の } S_1\text{-}S_2 \text{ 分子対が作用 (反応 } R_1 \text{ が発生) する確率} \tag{A.6}$$

もし，V 内の時間 t において，S_1 分子の分子数が X_1，S_2 分子の分子数が X_2 存在するならば，以下の式が成立する．

$$X_1X_2c_1dt = \text{反応 } R_1 \text{ が微小時間 } (t, t+dt) \text{ で } V \text{ 内のどこかで発生する確率} \tag{A.7}$$

式 A.7 を一般化して，熱平衡状態にある体積 V 内の N 種類の S_i 分子が X_i の数だけ存在する混合物について，M 種類の化学反応 R_μ が相互作用すると仮定すると，分子の物理的性質とシステム温度に依存する式 A.8 で表される M 種類の定数 c_μ を仮定できる．

$$c_\mu dt = \text{反応 } R_\mu \text{ に関わる分子が無限微小時間 dt で衝突する確率の平均} \tag{A.8}$$

式 A.8 における “平均” とは，その反応 R_μ に関係する分子の組み合わせの総数に c_μ を掛けた値が，反応 R_μ が次の微小時間 $(t, t+dt)$ で V 内のどこかで発生する確率になることを意味している．

A.1.2.2　反応速度定数と反応確率速度定数の関係

反応確率速度定数 c_μ は反応速度定数 k_μ と密接に関係している．式 A.5 のようなある反応 R_1 について，以下のような関係がある．

$$k_1 = \frac{Vc_1\langle X_1X_2\rangle}{\langle X_1\rangle\langle X_2\rangle} \qquad (\langle x\rangle \text{ は } x \text{ の平均値}) \tag{A.9}$$

式 A.9 の右辺は $\langle X_1X_2\rangle = \langle X_1\rangle\langle X_2\rangle$ と見なせるので，

$$k_1 = Vc_1 \tag{A.10}$$

もし反応 R_1 が 2 つではなく 3 つの反応分子を持つならば，V の代わりに V^2 を使用する．また，もし反応 R_1 がただ一つの反応分子を持つ (isomerization : 異性化) ならば，$V = 1$ になる．

次に，反応 R_1 の逆反応を以下の式 A.11 から考える．

$$R_2 : \quad 2S_1 \rightarrow S_1 + S_2 \tag{A.11}$$

式 (A.5) から，定数 c_2 が決定される (S_1 の分子の特定の組が微小時間で衝突し，R_2 の反応を発生する確率の平均)．V 内の S_1 分子の組み合わせの数は $X_1(X_1-1)/2!$ になる．

$$k_2 = \frac{Vc_2\langle\frac{X_1(X_1-1)}{2!}\rangle}{\langle X_1\rangle\langle X_1\rangle} = \frac{Vc_2}{2} \tag{A.12}$$

A.1.2.3　確率モデルにおける生化学システムの時間経過の計算

M 種類の相互反応をする N 種類の分子からなる生化学システムの時間経過を計算する．これは基本的な仮説である式 A.6 から求められる．

マスタ方程式

SSA の生化学システムの時間経過の計算は，生化学システムのマスタ方程式を設定し，それを解くことで求められる．マスタ方程式を数学的に解答することは困難で，SSA では後に述べるマスタ方程式を解くことと同義の処理を行ってシステムの時間経過を計算する．

マスタ方程式で最も重要な要素は主要確率関数 (grand probability function) である．生化学システムにおいて，主要確率関数は式 A.13 で表される．

$$\begin{aligned} P(X_1, X_2, \cdots, X_N; t) \equiv & V \text{ 内の時刻 } t \text{ で，} S_1 \text{ 分子が } X_1 \text{ 存在し，} \\ & \text{かつ } S_2 \text{ 分子が } X_2 \text{ 存在し} \cdots, \\ & \text{かつ } S_N \text{ 分子が } X_N \text{ 存在する確率} \end{aligned} \tag{A.13}$$

式 A.13 は，時刻 t の生化学システムの状態の確率 (stochastic state) を表している．たとえば，X_i に関する k 次の P のモーメント (k 乗平均) は，式 A.14 で与えられる．

$$\begin{aligned} X_i^{(k)}(t) \equiv \sum_{X_1=0}^{\infty} \cdots \sum_{X_N=0}^{\infty} X_i^k P(X_1, \cdots, X_N; t) \\ (i = 1, \cdots, N; k = 0, 1, 2, \cdots) \end{aligned} \tag{A.14}$$

X_i に関する k 次の P のモーメントは，“時刻 t で V にある分子数 X_i の平均の k 乗” である．ここでの “平均” とは，A.6 で定義された確率モデルでの平均反応発生確率を使い，同じ初期値 (X_ν) で，時間 0 から t まで非常に多くのシミュレーションを実行を繰り返したときの X_i の平均のこと意味している．

毎回のシミュレーションで S_i 分子の分子数 X_i は異なるが，これらの値の k 次の累乗の平均値は，シミュレーションを非常に多く実行することで，$X_i^{(k)}$ に収束していく．$k = 1, k = 2$ で表される 1 次モーメント，2 次モーメントは特に有用である．$k = 1$ での $X_i^{(k)}$ は時間 t で V 内に存在する S_i 分子の数の平均を示しており，$k = 2$ では式 A.15 を計算することで，母平均まわりの 2 次モーメント (分散) を求めることができる．

$$\begin{aligned} X_i^{(k)}(t) \equiv \sum_{X_1=0}^{\infty} \cdots \sum_{X_N=0}^{\infty} X_i^k P(X_1, \cdots, X_N; t) \\ (i = 1, \cdots, N; k = 0, 1, 2, \cdots) \end{aligned} \tag{A.15}$$

解析モデルにおける反応速度の式 A.1 で現れている $X_i(t)$ は，1 次のモーメント $X_i^{(1)}$ を近似しているが，完全な等式になることは少ない．

マスタ方程式は，$P(X_1, \cdots, X_N; t)$ で表される時間経過の関数である．$P(X_1, \cdots, X_N; t)$ は，システムが時刻 $t + dt$ で状態 $(X_1, \cdots, X_N)$ になる $1 + M$ 通りの方法があり，それぞれの確率の合計である．この P は，式 A.6 から確率の加法と乗法を使用して求められる．

$$P(X_1, \cdots, X_N; t + dt) = P(X_1, \cdots, X_N; t) \left[1 - \sum_{\mu=1}^{M} a_\mu dt \right] + \sum_{\mu=1}^{M} B_\mu dt \tag{A.16}$$

ここで，ある数量 a_μ を定義する．

$$\begin{aligned} a_\mu dt \equiv & c_\mu dt \times \{ \text{ 状態 } (X_1, \cdots, X_X) \text{ で反応 } R_\mu \text{ が起こる分子の組み合わせ数 } \} \\ = & \text{ 時刻 } t \text{ で状態 } (X_1, \cdots, X_N) \text{ のシステムで，} \\ & (t, t + dt) \text{ において } R_\mu \text{ 反応が発生する確率} \end{aligned} \tag{A.17}$$

式 A.16 の第 1 項は，生化学システムが時刻 t において状態 $(X_1, \cdots, X_N)$ になり，時刻 $t+dt$ までその状態 (反応が起こらない) のままでいる確率である．第 2 項の $B_\mu dt$ は，時刻 t でシステムが状態 $(X_1, \cdots, X_N)$ になり，時刻 $t+dt$ で R_μ の反応が発生する確率である．

式 A.16 からマスタ方程式の導出をすることができる．

$$\frac{\partial}{\partial t}P(X_1, \cdots, X_N; t) = \sum_{\mu=1}^{M} [B_\mu - a_\mu(X_1, \cdots, X_N; t)] \tag{A.18}$$

特別な場合を除いて，マスタ方程式の記述は簡潔であるが，それを解くことは困難である．さらに，反応速度方程式とは異なり，独立変数の数と性質が原因で，計算機を使って数値的に解を導くことも容易ではない．全ての反応 R_μ が単純な単分子反応でなければ，モーメントを導出するための式は常に高次のモーメントを含んでいるため，特定の X_i や $\Delta_i(t)$ のモーメントの時間変化を方程式で表して計算するのは無限大の時間が掛かってしまう．マスタ方程式は正確で簡潔に表現することはできるが，実際的な数値計算を行う上では有用ではない．そのため，確率モデルでは，結果的にマスタ方程式と同じ解答を得られる方法を用い，計算機でシミュレーション可能な方法を用いる．

A.1.2.4　反応確率分布関数

生化学システムについて，確率的方法を用いた場合に，時間経過を計算する方法について述べる．

時刻 t で状態 $(X_1, \cdots, X_N)$ のシステムがあるとすると，このシステムの時間経過を計算するには，いつ，どの反応が起こるのかを明らかにする必要がある．

そのため，以下の式 A.19 で定義される関数 $P(\tau, \mu)$ を導入する．

$P(\tau, \mu) \equiv V$ 内の状態 $(X_1, \cdots, X_N)$ で，微小時間 $(t+\tau, t+\tau+d\tau)$ の間に R_μ の反応が起こる確率 (A.19)

関数 $P(\tau, \mu)$ は反応確率分布関数と呼ばれる．τ は次の反応が起こるまでの時間，μ は発生する反応の種類を示している．

この生化学システムで定義された反応 R_μ について，式 A.20 で定義される関数 h を計算する．

$h_\mu \equiv$ 状態 $(X_1, \cdots, X_N)$ において，R_μに関係する分子の組み合わせ数　　$(\mu = 1, \cdots, M)$ (A.20)

もし反応 R_μ が $S_1 + S_2 \to S$ という形式であるならば，$h_\mu = X_1 X_2$ となり，R_μ が $2S_1 \to S$ という形式ならば，$h_\mu = \frac{1}{2}X_1(X_1 - 1)$ になる．一般的に，h_μ は変数 $X_1, \cdots, X_N$ の複合関数である．

状態 $(X_1, \cdots, X_N)$ において，時刻 $(t, t+dt)$ で反応が起こらない確率を $P_0(t)$ とする．また，R_μ が時刻 $(t+\tau, t+\tau+d\tau)$ で発生する確率を a_μ とすると，a_μ は式 A.21 で表される．

$$a_\mu d\tau \equiv h_\mu c_\mu d\tau \tag{A.21}$$

よって，$P(\tau, \mu)$ は以下のように定義される．

$$P(\tau, \mu)d\tau = P_0(\tau) \cdot a_\mu d\tau \tag{A.22a}$$

右辺の $P_0(\tau)$ の導出について，$[1 - \sum_\nu a_\nu d\tau']$ が状態 $(X_1, \cdots, X_N)$ の時間 $d\tau'$ で反応が発生しない確率である事を利用し，以下の式から $P_0(\tau)$ を得る．

$$P_0(\tau' + d\tau') = P_0(\tau') \cdot \left[1 - \sum_{\nu=1}^{M} a_\nu d\tau'\right] \tag{A.22b}$$

$$P_0(\tau) = \exp\left[-\sum_{\nu=1}^{M} a_\nu d\tau'\right] \tag{A.22c}$$

式 A.22c を式 A.22a に代入することで，反応確率密度関数 $P(\tau, \mu)$ を計算することができる．

$$P(\tau,\mu) = \begin{cases} a_\mu \exp(-a_0\tau) & \text{if } 0 \leqslant \tau < \infty \text{ and } \mu = 1, \cdots, M \\ 0 & \text{otherwise} \end{cases} \tag{A.23}$$

A.1.3 Direct Method

本節では，生化学システムにおいて，いつ，どの反応が発生するのかを計算する方法を述べる．以下の方法は Direct Method と呼ばれ，First Reaction Method，Next Reaction Method とも統計的に等価なアルゴリズムである．

これは，一様な乱数から反応確率密度関数 $P(\tau, \mu)$ に分布する乱数を作り出して τ, μ を計算することで明らかになる．$(0, 1)$ の一様乱数 r_1，r_2 から，以下の式を用いて τ と μ を生成する．

$$\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right) \tag{A.24a}$$

$$\sum_{\nu=1}^{\mu-1} a_\nu < r_2 a_0 \leqslant \sum_{\nu=1}^{\mu} a_\nu \qquad (\mu : \text{Integer}) \tag{A.24b}$$

この 2 式より $P_2(\mu) = a_\mu / a_0$ に従う整数の乱数 μ と，確率密度関数 $P_1(\tau) = a_0 \exp(-a_0\tau)$ に従う乱数 τ が生成され，$P_1(\tau) \cdot P_2(\mu) = P(\tau, \mu)$ を求めることができる．

シミュレーションは Step 1. から Step 3. の繰り返しで進行する (図 A.1.3).

Step 0. 初期化

M 種類の反応定数 $c_1, \cdots, c_M$ と，N 個の初期分子数 $X_1, \cdots X_N$ に初期値を格納する．時刻 t と反応数カウンタ n を 0 に初期化する．乱数生成器を初期化する．

Step 1.

この時点での分子の数ごとに，M つの値 $a_1 = h_1 c_1, \cdots, a_M = h_M c_M$ を計算し，a_0 に a_ν の合計値を代入する．

Step 2.

$(0, 1)$ の一様乱数 r_1 と r_2 を生成し，τ と μ を計算する．

Step 3.

τ によって t を増加する．反応 R_μ が影響を与える分子の数を調節する (例えば，R_μ が式 A.5 の反応だったならば，X_2 が 1 増加し，X_1 が 1 減少する)．反応数カウンタ n をインクリメントする．これを Step 1. に戻って繰り返す．

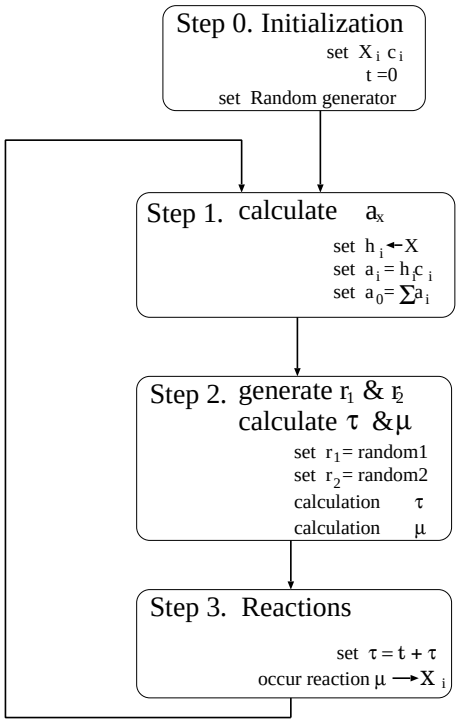


図 A.4: 確率モデルシミュレーションアルゴリズム

Step 1. から Step 3. の繰り返しの中のどこか，あるいは任意の t または n の間隔において，$(X_1, \cdots, X_N, t)$ の値を出力する．また，t あるいは n が何らかの値に到達したときや，a_0 が 0 になったときには計算を終了する．

A.1.4　確率モデルの利点と制限

SSA の利点と制限を述べる．

このシミュレーションアルゴリズムは，基礎的な仮定 (式 A.6) の形式で記述された生化学システムについて，厳密なシミュレーションが可能である．また，SSA は，従来の微分方程式の数値解法のように，微小時間 Δt ごとに計算を繰り返す方法をとらず，“次の反応までの時間” を計算するため，分子数が急激に変化するようなシステムにも有効である．このシミュレーションアルゴリズムを計算機上で実行した場合，メモリスペースの要求量が少ないことも利点の一つとして挙げられる．N 種類の分子と M 種類の反応では，N 種類の分子数と M 個の c_ν，$M+1$ 個の a_ν の変数を格納するメモリスペースが確保できれば良い．このことは，FPGA 上に SSA を実装する上でも大きな利点となる．

また，シミュレーションの過程や結果を読み取ることが容易であることも利点のひとつである．シミュレーションサイクルごとに分子数が数値的に評価できるため，平均，分散，相関などを計算することが容易である．

確率モデルのシミュレーションアルゴリズムは，反応数に比例した計算時間が掛かる．そのため，シミュレーションできる分子の種類，数，反応数は適切な数を指定し，計算時間を適切に抑

えなければならない．シミュレーションが可能なシステムは分子が一様分布している状態に限られ，分子数に偏りがある場合は，システムの大きさを変更するなどして対応する必要がある．また，このシミュレーションアルゴリズムから信頼できる結果を得るためには，信頼性のある乱数生成器を使用しなければならない．この信頼性に関する標準を得ることは非常に困難である．さらに，良い統計精度を得るためには，複数回のシミュレーションの評価を取る必要があり，その実行回数は計算時間などのコストとのトレードオフになる．

A.1.5 Lotka システム

化学反応の確率モデルシミュレーションを開発した Gillespie は，確率モデルシミュレーションを適用した例として，4 つの生化学システムについて言及している．それは，不可逆異性化，Lotka システム，Blusselator システム，Oregonator システムの 4 つである．不可逆異性化反応は逆反応の起こらない不可逆変化する分子の反応であり，Gillespie はこの反応のシミュレーションを用いて確率モデルの具体的な検証を行っている．その他の 3 つの生化学システムはより複雑なモデルとして挙げ，検証している．不可逆異性化反応は，後に述べる Lotka システムの一部として使われる基本的なモデルである (式 A.25c のみで記述されるモデルが不可逆異性化反応である)．

Lotka システムは，1920 年に Lotka が以下のような自触媒反応を観察したことから研究が始まった．

$$\bar{X_1} + X_2 \xrightarrow{c_1} 2X_2 \tag{A.25a}$$

$$X_2 + X_3 \xrightarrow{c_2} 2X_3 \tag{A.25b}$$

$$X_3 \xrightarrow{c_3} Z \tag{A.25c}$$

このモデルは，Volterra が開発した捕食者-被食者関係に対応する生態系の数学的モデルから研究されているモデルである．Volterra は上記の反応を以下のような微分方程式で記述し，これに対応する反応速度方程式の使用法について研究した．

$$\frac{dX_1}{dt} = c_1 X_1 X_2 - c_2 X_2 X_3 \tag{A.26a}$$

$$\frac{dX_3}{dt} = c_2 X_2 X_3 - c_3 X_3 \tag{A.26b}$$

式 A.25b は捕食種 X_3 が被食種 X_2 を食べて自分を再生産する様子を表現している．また，式 A.25a は X_2 が食料 X_1 を食べて自分を再生産する様子を記述している．ここでは X_1 はただ消耗されるだけで，増加しないものと仮定している．異性化式 A.25c は X_3 の死を表現している．

定常状態は $dX_2/dt = dX_3/dt = 0$ の条件を満たしているシステム状態であることは明らかである．

Lotka システムは化学反応の単純なモデルの一例であるが，これは以下のような単純なルールに支配される生態系を考えることでどのような反応であるかを容易に理解することができる．ここでの Lotka 反応モデルのシミュレーションに際しては，前述の式 A.25 に加え，X_2 の死，および X_1 は反応によって減少しない，という条件を追加している．

付 録

浮動小数点演算器

B.1 FPGA 向け浮動小数点算術演算器の実装

B.2 浮動小数実数の表現

IEEE754 は，実数を 32 ビットまたは 64 ビットで表現する方法を規定している [74][75]．32 ビット表現を単精度，64 ビット表現を倍精度と呼び，どちらもデータは符号部，指数部，仮数部の 3 つの部分から構成される．単精度浮動小数点実数では，1 ビットの符号部 s，8 ビットの指数部 e，23 ビットの仮数部 m を持つ．倍精度の場合は，1 ビットの符号部，11 ビットの指数部，52 ビットの仮数部から構成される．

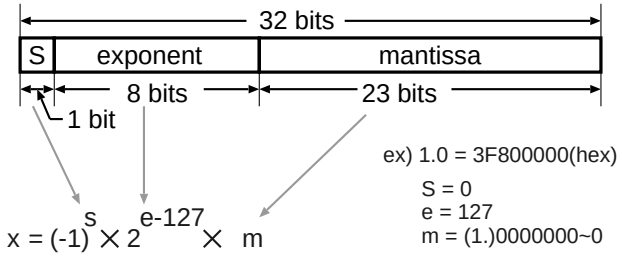


図 B.1: 単精度浮動小数点フォーマット

実数 x を単精度浮動小数点実数フォーマットで表現する場合，s, e, m には以下の式から得られる値が格納される (図 B.1).

$$x = (-1)^s \times 2^{e-127} \times m \tag{B.1}$$

指数部 e は 127 だけバイアスが掛けられており，仮数部の小数点の位置を移動するために使用される．仮数部 m は $[1, 2)$ の範囲の固定小数点実数で，最上位の 1 ビット (正規化ビット:常に 1) を省略した小数点以下 23bit を示す．例えば，$x = 1.0$ の場合は，符号は正，指数は 0，仮数は 1.0 であるため，$s = 0, e = 127, m = 0$ の値が格納される．

浮動小数点実数フォーマットは，特別な値を表す特殊データ形式が定義されている．

非数 (NaN) : e が最大値で m が 0 以外の数.

無限大 ($\pm\infty$) : e が最大値で m が 0 の数 (s で正負を表現)

ゼロ (± 0) : e および m が 0 (s により正負のゼロがある)

NaN は，さらに m の最上位 bit(仮数部の小数点以下第一位) が 1 のものは SNaN，0 のものは QNaN と呼ばれる．

IEEE754 では，以下の 5 つの例外が定義されている．

無効浮動小数点演算例外： 無限大や NaN の演算，整数へのフォーマット変換時の不正などで発生

浮動小数点ゼロ除算例外： 0 でない値を 0 で除算した場合に発生

浮動小数点オーバーフロー例外： 計算結果が表現可能な最大値よりも大きくなった場合に発生

浮動小数点アンダーフロー例外： 計算結果が表現可能な最小値よりも小さくなった場合に発生

浮動小数点精度落ち例外： 丸め時に丸め用の追加 3 ビットのいずれかが 1 であった場合に発生．

B.2.1 丸め処理

浮動小数点実数の性質から，計算の結果，仮数部に収まらないビット数の値が求められる場合がある．演算中は仮数部のビット数を増やすことで精度の下落を防いでいるが，演算結果は浮動小数点フォーマットで表現されなければならないため，表現可能な形に値の微調整を行わなければならない．丸めとは，仮数部で表現可能なビット数からあふれたビットについて，切り上げまたは切り捨てを行い，結果の近似値を求めることである．

丸めは，浮動小数点フォーマットの仮数部の下位に，Round bit，Guard bit，Sticky bit と呼ばれる 3 ビットを追加し，それらを利用して実現される．これらのビットの初期値は 0 で，仮数部が右シフトされる際に上位のビットが格納される．Sticky bit には，Sticky bit よりも右にあふれた値の論理和が格納される．

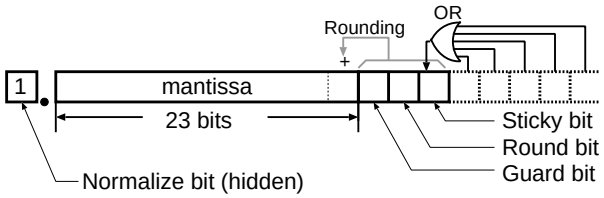


図 B.2: 丸め処理のための追加ビット

IEEE754 では 4 つの丸めモードが定義されている (表 B.1)．演算処理の後，指定された丸めモードに従って追加ビットが丸められ，繰上げが発生した場合は，仮数部に加算される．

表 B.1: 丸めモードの種類

Rounding Mode	手法
RN (Round to Nearest)	表現可能な最も近い値に丸める
RZ (Round toward Zero)	0 の方向へ丸める (切捨て)
RP (Round Upload)	$+\infty$ 方向へ丸める
RM (Round Downward)	$-\infty$ 方向へ丸める

B.2.2 基本算術演算処理

一般的に，浮動小数点実数の四則演算は，以下のステップで実行される．

1. **Unpack :** 入力を符号部，指数部，仮数部に分割し，仮数部に正規化ビット，丸め用追加ビットを追加する
2. **演算処理 :** 演算を行う
   (a) 出力の符号ビットの決定
   (b) 指数部の調整
   (c) 仮数部を入力とする整数演算
3. **Repack :** 丸めモードに基づいて仮数部の丸めを行い，演算結果の符号部，指数部，仮数部を結合する．

B.3 加減算器

浮動小数点のデータは，([符号ビット][絶対値]) で表されるという性質から，入力値の符号によっては加算であっても絶対値同士の減算を行ったり，減算であっても絶対値の加算が行われたりする場合がある．そのため，加算と減算は同一の回路で構成する．加減算結果 q の符号 q_s とその絶対値 $|q|$ は，入力 a, b の符号 s_a, s_b と絶対値 $|a|$，$|b|$ の大小関係を用いて，表 B.2 で決定される．

加減算は以下の 6 ステップの処理で計算される (図 B.3).

1. 絶対値 $|a|$，$|b|$ の比較，ゼロと無限大の判定
2. 比較結果と符号 s_a，s_b，演算子から，表 B.2 に従う結果の符号 s_q と $|a|$，$|b|$ に適用する演算を決定する
3. 絶対値が小さい方の仮数部を，指数部の差だけ右シフトする．仮数部の LSB には，シフトアウトしたビットの論理和が格納される
4. 仮数部同士の加減算を行う (表 B.2 の演算)
5. MSB が 1 になるように結果をシフトする (m_q)
6. 大きい方の指数部をシフトした数だけ加減算する (e_q)

計算結果がゼロになる場合，無限大になる場合には，ステップ (6) で適切な値が出力される．オーバーフロー，アンダーフローもステップ (6) で指数の値を判定することで検出することが可能である．

B.4 乗算器・除算器

浮動小数点フォーマットの乗算，除算は，概念的に以下のステップで実行される (図 B.4).

1. 符号ビットの決定

表 B.2: 加減算処理

s_a	Operation	s_b	Compare	s_q	$\|q\|$
$+$	$+$	$+$	$*^{\dagger}$	$+$	$\|a\|+\|b\|$
$-$	$+$	$-$	$*^{\dagger}$	$-$	$\|a\|+\|b\|$
$+$	$-$	$-$	$*^{\dagger}$	$+$	$\|a\|+\|b\|$
$-$	$-$	$+$	$*^{\dagger}$	$-$	$\|a\|+\|b\|$
$+$	$-$	$+$	$\|a\|>\|b\|$	$+$	$\|a\|-\|b\|$
			$\|a\|=\|b\|$	$+^{\ddagger}$	0
			$\|a\|<\|b\|$	$-$	$\|b\|-\|a\|$
$-$	$-$	$-$	$\|a\|>\|b\|$	$-$	$\|a\|-\|b\|$
			$\|a\|=\|b\|$	$+^{\ddagger}$	0
			$\|a\|<\|b\|$	$+$	$\|b\|-\|a\|$
$+$	$+$	$-$	$\|a\|>\|b\|$	$+$	$\|a\|-\|b\|$
			$\|a\|=\|b\|$	$+^{\ddagger}$	0
			$\|a\|<\|b\|$	$-$	$\|b\|-\|a\|$
$-$	$+$	$+$	$\|a\|>\|b\|$	$-$	$\|a\|-\|b\|$
			$\|a\|=\|b\|$	$+^{\ddagger}$	0
			$\|a\|<\|b\|$	$+$	$\|b\|-\|a\|$

† Don't care
‡ If rounding mode is RM, a sign is (-0) n

2. 指数部の整数加算・減算

3. 仮数部の整数乗算・除算

浮動小数点データの乗算・除算の結果の符号は，入力が同符号ならば正，異符号ならば負であるため，入力の符号ビットの排他的論理和 (XOR) となる．

浮動小数点乗算 浮動小数点の乗算は，指数部同士の加算と仮数部同士の乗算で計算できる．

$$q_s = a_s \oplus b_s \tag{B.2}$$
$$q_e = a_e + b_e - 127(\text{or}126) \tag{B.3}$$
$$q_m = a_m \times b_m \tag{B.4}$$

指数部はバイアスが 127 だけ加算されているため，指数部同士の加算の際にバイアス部分を減算して調整する必要がある．仮数部の整数乗算部分には，Virtex-II PRO の乗算ブロックを使用している．27 ビットの正規化数 (仮数部の MSB が 1) 同士の乗算結果は 54 ビットになる．乗算結果は 54 ビット目が 1 になる場合と，54 ビット目が 0 で 53 ビット目が 1 になる場合のどちらかである．結果の指数部は，仮数部の乗算結果の MSB が 1 である場合は 127 を，0 の場合は 126 を減算することで調整する．指数部の減算は，入力の指数部の上位に 1 ビットの 0 を追加して 9 ビット幅で行う．オーバーフローは指数部の加算結果，126 または 127 の減算結果の MSB が両方とも 1 である場合に発生する．また，アンダーフローは，指数部の加算結果の MSB が 0 で減算結果の MSB が 1 の場合に発生する．

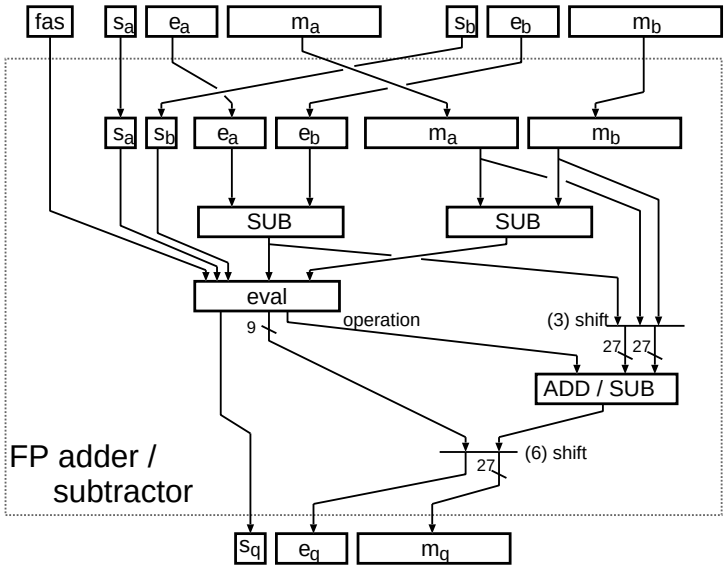


図 B.3: 加減算器の構成

浮動小数点除算 乗算の場合と同様に，除算は指数部の減算と仮数部の除算で計算できる．

$$q_s = a_s \oplus b_s \tag{B.5}$$

$$q_e = a_e - b_e + 127 \tag{B.6}$$

$$q_m = a_m \div b_m \tag{B.7}$$

結果の指数部の調整も，乗算と同様の方法で実現できる．入力として正規化数のみを対象とする場合，仮数部の除算は必ず MSB が 1 である 27 ビットの除算が行われる．このとき，結果は $1.\cdots$，もしくは $0.1\cdots$ になる．整数除算の必ずしも割り切れるとは限らないため，浮動小数点の除算結果として十分な範囲まで商を求める必要がある．単精度浮動小数点フォーマットにおいては，最初に 1 が現れる場所から 27 ビット分の商が求められれば十分である．Sticky bit はそれよりも右側に存在するビットの論理和が格納されるという性質から，27 ビット目は，26 ビット目の商と同時に求められる剰余の論理和が格納される．仮数部の整数除算は，基数 2 の減算シフト型の除算アルゴリズムで実行される．整数除算 $a \div b$ を行う場合，以下の順序で計算が行われる．

1. 1 ビット左シフトで 2 倍の値が得られる性質を利用して，除数 b，$b \times 2$，$b \times 3$ の 3 つの値を計算する
2. $a - b$，$a - 2b$，$a - 3b$ をそれぞれ計算し，表 B.3 にしたがい，結果が負にならない最大のものを 2 ビット分の商として選択する．剰余は減算結果になる
3. 剰余の LSB に 2 ビットの 0 を追加し新被除数 a とする
4. 最初に 1 が出たビットから 26 ビット分の商が求められるまでステップ (2) に戻って繰り返す

上記の計算の 1 回目のステップ (2) で求められる商は必ず $01, 10, 11$ のいずれかになる．またステップ (2)-(4) の計算で，商が 2 ビットずつ求められる．そのため，26 ビット分の商を

表 B.3: 基数 2 の除算処理

$a-b$	$a-2b$	$a-3b$	quotient	reminder
$-$	$*$[a]	$*$[a]	00	a
$+$	$-$	$*$[a]	01	$a-b$
$+$	$+$	$-$	10	$a-2b$
$+$	$+$	$+$	11	$a-3b$

[a] Don't care

求めるためには，ステップ (2)-(4) の計算を最大 14 回繰り返す必要がある．オーバーフロー，アンダーフローは乗算と同様の方法で検出が可能である．

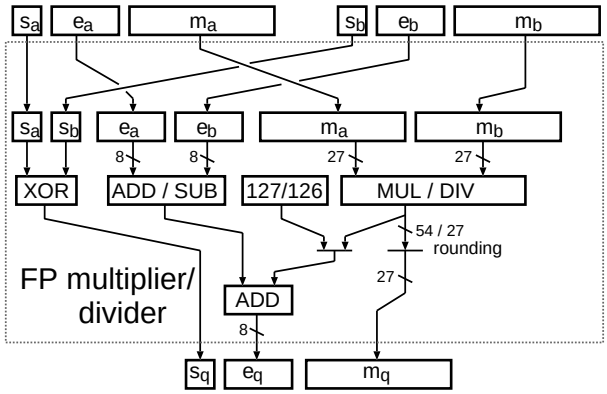


図 B.4: 乗算器/除算器の構成

B.5 対数計算器

対数計算器は，二次補間を利用して入力の対数を計算する演算ユニットである．対数の底は任意の値を指定できる．対数の底を α，入力を $x = (-1)^{s_x} \times 2^{(e_x-127)} \times m_x$ $(s_x = 0; 1 \leq m_x < 2)$ とすると，対数 $\log_\alpha(x)$ は以下の計算から導かれる．

$$\begin{aligned}\log_\alpha(x) &= \log_\alpha\left((-1)^{s_x} \times 2^{(e_x-127)} \times m_x\right) \\ &= \log_\alpha\left(2^{(e_x-127)}\right) \times \log_\alpha(m_x) \\ &= \frac{(e_x-127) \times \log_2(m_x)}{\log_2 \alpha}\end{aligned}$$

この変換により，$[1,2)$ の範囲の値 m_x について，対数 $\log_2(m_x)$ を計算できれば，任意の底の対数を計算できる演算ユニットを構成することができる．$\log_2(m_x)$ は，二次補間を利用して計算する．二次補間は，線形近似による一次補間を行ったうえで，さらにそれを補正することで行われる．

$[1,2)$ の範囲において，$\log_2(x)$ は図 B.5 のような形になる．本実装における一次補間では，図 B.5 の曲線を 1024 本の直線に分割し，入力 x がどの直線上のどの位置にあるかを計算することで実現される．分割数が 1024 であるため，x の小数点以下 10 ビットを用いて直線を選択し，x の値

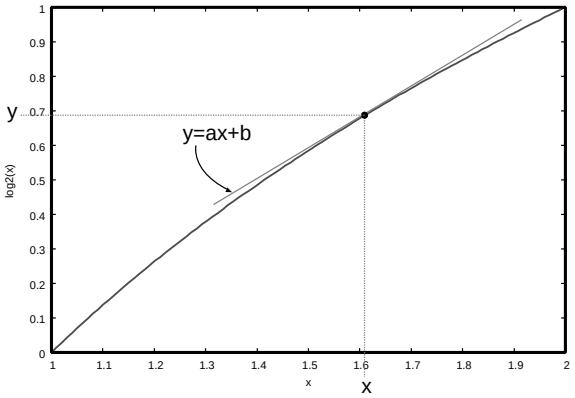


図 B.5: 対数演算の 1 次補間

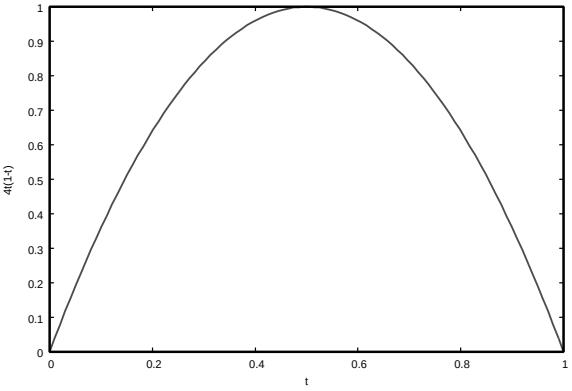


図 B.6: 対数演算の 2 次補間

から直線上の位置を計算する．一次補間で使用される各直線と，実際の $\log_2(x)$ の誤差は，図 B.6 のような二次曲線の形になる．各直線について，$\log_2(x)$ との最大誤差の値を保持しておき，直線上の位置 (x の小数点以下 11 ビットから 20 ビット) と最大誤差値を用いて一次補間で求められた値に加算する．これが，二次補間である．

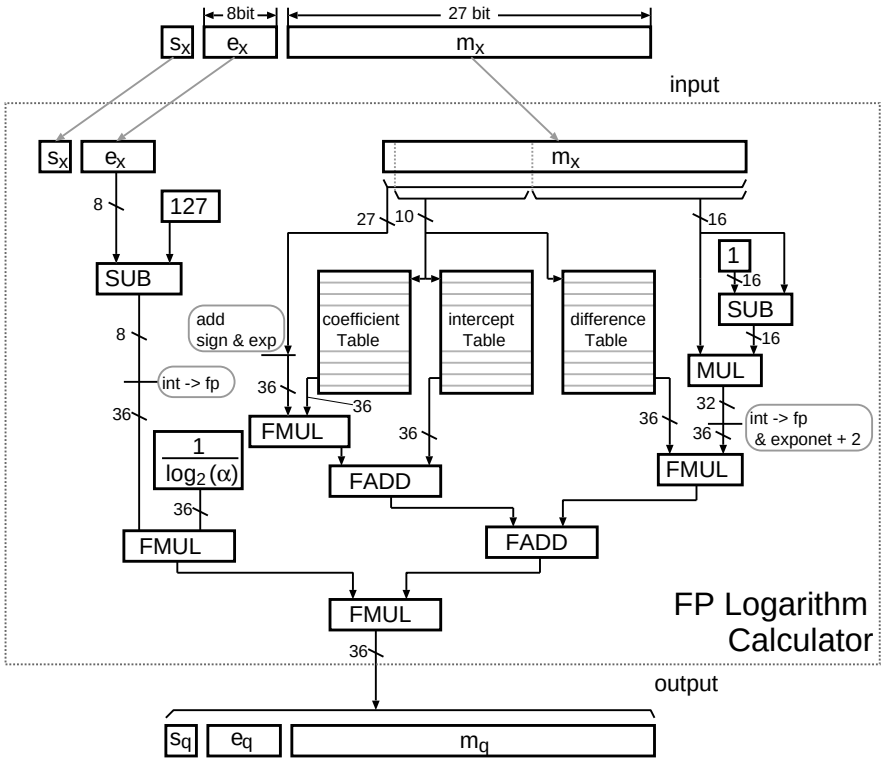


図 B.7: 対数計算器の構成

実装では，以下のような 3 つのテーブルを用意し，$\log_2(x)$ の計算を行う．

- $[1, 2)$ の範囲を等間隔に 1024 分割する
- 点 x_n と x_{n+1} を結ぶ直線の傾き a_n と切片 b_n を求め，傾きテーブルと切片テーブルに格納する

- 各範囲ごとに，線形近似 $y = ax + b$ と $\log_2(x)$ の差の最大値 d_n を格納するテーブル

1. 仮数部の MSB を除く上位 10 ビット (n とする) を用いて，傾きテーブル，切片テーブルより傾き a_n，切片 b_n を読み出す
2. $a_n \times m_x + b_n$ を計算し，一次補間の値 y_{m_x} を計算する
3. 仮数部の小数点以下 11 ビット目から 20 ビット目の 10 ビット (値を t とする) を用いて，$d_n \times t \times (1 - t)$ を計算し，y_{m_x} に加算する (二次補間)

こうして求められた $\log_2(m_x)$ に $e_x - 127$ を加算し，$\log_2(\alpha)$ の逆数で除算を行えば，$\log_\alpha(x)$ を求めることができる．本実装では，あらかじめ $1/\log_2 \alpha$ の値を演算器内に設定しておく方法を取った (図 B.7).

B.6 浮動小数点演算器の評価

B.6.1 合成結果

表 B.4: 演算器の評価

Modules	Stages	Slices[%]	Mult.[%]	BlockRAM[%]	Freq.[MHz]
adder/subtractor	6	794/33080(2.40)	-	-	125.8
multiplier	8	265/33080(0.80)	4/328(1.22)	-	221.0
divider	17	1717/33080(5.19)	-	-	148.7
comparator	1	22/33080(0.07)	-	-	127.8
unpack	1	19/33080(0.05)	-	-	249.0
repack	1	35/33080(0.11)	-	-	177.7
logarithmic calculator	32	2834/33080(8.57)	6/328(1.83)	13/328(3.96)	103.8

浮動小数点実数の四則演算および対数計算の演算器の合成，配置配線結果を表 B.4 に示す．実装には verilog-HDL を用い，合成，配置配線ツールは Xilinx ISE6.3i を使用した．ターゲットデバイスには，ReCSiP2 が搭載している FPGA である Virtex-II PRO(XC2VP70-5) を選択した．また，整数乗算ブロック，BlockRAM など，Virtex-II PRO の IP core として利用できる場合は，Xilinx CORE Generator 6.3i で生成したものを利用している．表 B.4 から，加減算器に比べ，乗算器の面積が非常に小さくなっている．これは，仮数部の整数乗算部分を Virtex-II PRO が持つ整数乗算専用の乗算ブロックを使用しているため，使用 LUT 数が少なくなっていることがその理由として考えられる．除算器は，27 ビットの整数減算器を $42 (= 3 \times 14)$ 個持つため，使用面積が大きくなっている．対数計算ユニットについて，Virtex-II PRO の BlockRAM は 18bit×1024entry が 1 単位であるため，テーブルのエントリを 1024 の倍数で持つことは妥当な選択である．対数計算を大量に行う科学技術計算アプリケーションでは，対数を計算する専用モジュールを利用する方法も効果が高いと考えられる．表 B.4 より，Virtex-II PRO 上に浮動小数点演算器を数十個構成可能であることがわかる．

B.6.2 面積・性能比較

表 B.5: 演算器の比較

Modules	Stages			Slices			Mult		Freq.[MHz]		
	DCD	Ours	gain	DCD	Ours	gain	DCD	Ours	DCD	Ours	gain
DFPADD	5	6	1.2	611	794	1.3	-	-	96	126	1.3
DFPMUL	7	8	1.1	742	265	0.4	4	4	96	221	2.3
DFPDIV	15	17	1.1	1723	1717	1.0	-	-	72	149	2.1
DFPCOMP	1	1	1.0	70	22	0.3	-	-	149	128	0.9

実装した演算器の実用性について，Desital Core Design(DCD) 社が商用に提供している Xilinx デバイス向け浮動小数点演算 IP コアを比較対象として評価を行った [76]．DCD 社が発表している IP コアのデータシートと実装した演算器の比較を表 B.5 に示す．DCD 社の IP コアは Virtex-II PRO(Speed Grade 7) のものを使用している．表 B.5 は各項目左から，DCD 社の IP コアの情報，本研究報告で行われた実装，倍率 (= [本実装の評価値]/[DCD 社の公表値]) を表している．Virtex-II PRO の Speed Grade が異なるため，単純な比較は困難であるが，Speed Grade 7 と 5 では 7 の方が動作周波数が高い．そのため，以降の検討では，少なくとも表 B.5 に示された動作周波数の倍率の差が存在すると仮定する．

表 B.4 と表 B.5 を比較すると，以下の傾向がわかる．

- DCD 社 IP コアはパイプライン段数が短い
- 本実装は動作周波数が高い

浮動小数点乗算器について，本実装の乗算器は DCD 社提供のもの比べて，Unpack，Repack を分離しているとは言え面積が半分以下である．これは，DCD 社の IP コアが，入出力が非正規化数の場合に備えた実装になっているためと考えられる．浮動小数点比較器 (comparator，DFPCOMP) は，DCD 社の IP コアが $A > B$，$A = B$，$A < B$ の 3 つの比較結果を出力できることに対し，本実装の比較器は $A \geq B$ であるか否かの判定のみを行うものであるため，面積，パイプライン段数に大きな違いが現れている．

両者とも，1 クロックに 1 回の演算を行うことができる設計であるため，同一動作周波数で計算できる演算回数は等しい．計算結果が同一の演算器の次の入力になる場合は，パイプライン段数が小さい方が有利である．DCD 社の設計は演算の時間が短く，データに依存性が大きい処理を高速に計算することができる．それに対し，本実装では細かくパイプラインを刻み，1 クロックで行う演算量が少ないため，パイプラインが長くなる代わりに高速な動作が可能になっている．このような設計は，データに演算単位の依存性が無い計算を大量に行う場合に有利な構成であると言える．具体的なアプリケーションは，ハードウェアを用いた科学技術計算や，動画像に代表されるマルチメディア処理のアクセラレーションが考えられる．

面積について，パイプライン段数が長くなる構成から，他の部分の計算を待つ場合にシフトレジスタを構成するため，本実装の演算器は使用面積が大きくなっている．ターゲットデバイスについて，DCD 社の IP コアが小規模，中規模の FPGA を対象としていることに対し，本実装の演算器の主な対象は大規模 FPGA と大量のメモリを搭載したシステムである．ホスト PC の CPU の動作周波数と FPGA の動作周波数には数十倍の開きがある．PC に FPGA ボードを接続してホス

トの処理を FPGA ボードで高速化する際には，演算器のパイプラインを埋め，数十個の浮動小数点演算器を並列動作させることで実現される．データに依存性が比較的少ない処理について，大型の FPGA に本実装の浮動小数点演算器を数十個構成し，100MHz 程度の動作周波数で処理を行う場合，数 GFLOPS の計算能力が期待できる．本実装は，細かいパイプライン刻みの演算器を用いた高速な動作により，FPGA を用いたソフトウェアの高速化の有効性を示したと言える．

付 録

ReCSiP Board概略

ReCSiP Board は，ReCSiP の動作確認のためのリファレンス環境として開発された基板であり，Xilinx 社製の FPGA を搭載した PCI ボードである．ReCSiP-1, 2, 2.1 の 3 種類のボードが開発されており，現在は ReCSiP-2/2.1 Board を用いてシミュレータが稼働している．

図 C.1 に各ボードの概要を示し，以下で各ボードの構成について述べる．

C.1 各ボードの構成部品

C.1.1 ReCSiP-1 Board

ReCSiP-1 Board(図 C.2) は，Virtex-II (XC2V6000) と，同期 SRAM を搭載した基板であり，PCI インタフェイス部の動作確認など，後継基板の開発のためのノウハウ蓄積に貢献した．QL5064 の開発と並行して基板の設計が行われたため，QL5064 が BGA ソケットによる実装になっており，Virtex-II のコンフィギュレーションも MultiLINX cable と QL5064 の双方から行えるようになっている．

メモリは SRAM, DRAM ともに，2 チップ 1 組でそれぞれ 36bit, 32bit のデータ幅で FPGA に接続されており，SRAM は合計 4 組，DRAM は 1 組の構成である．

C.1.1.1 主要部品

- FPGA: Xilinx Virtex-II (XC2V6000-4BF957C)
- SRAM: Micron 18Mb DCD SyncBurst SRAM (MT58L1ML18D: 1Mx18) x 8
- DRAM: Micron MT48LC16M16 166MHz SDR SDRAM (16Mx16) x 2

C.1.1.2 ジャンパ設定

ReCSiP-1 Board のジャンパ・コネクタ類の設定を表 C.1 に示す．通常は PCI バス，QL5064 経由で Virtex-II の configuration を行い，PCI バスの 66MHz 動作を許可するため，これらを設定するジャンパ J3 はオープンでよい．ロジックアナライザ等で信号を観測する場合のために，J6, J7 で合計 40 本のユーザ利用可能なピンヘッダが用意されている．

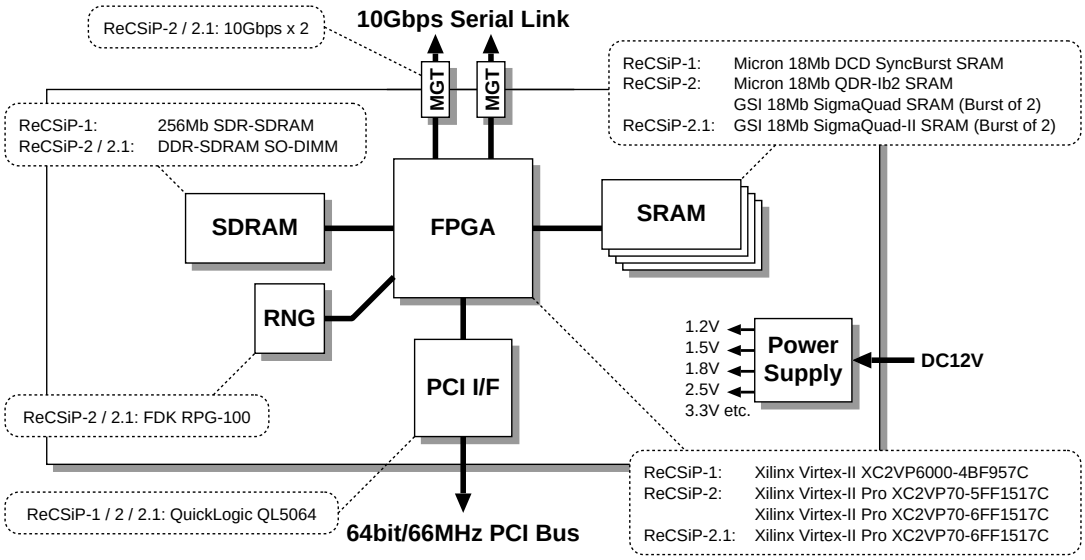


図 C.1: ボードの概要

図 C.2: ReCSiP-1 Board

表 C.1: ReCSiP-1 Board のジャンパ・コネクタ

部品番号	ピン番号	機能
J1	1-8	MultiLINX ケーブルで FPGA の構成情報を転送するためのジャンパ．1 番ピンから順に D0-D7 を接続
J2	1-7	MultiLINX ケーブルの制御系信号用．1 番ピンから順に BUSY, WRITE, CS, INIT, PROG, DONE, CCLK を接続する．8 番ピンは未使用
J3	1-2	Open: QL5064 経由で Virtex-II を configuration Close: MultiLINX ケーブル経由で Virtex-II を configuration
J3	3-4	Open: PCI バスの 66MHz 動作を許可 Close: PCI バスを常に 33MHz で動作させる
J4	1-4	Virtex-II の JTAG 信号線．1 番ピンから順に TMS, TDO, TDI, TCK
J6 J7	1-20 1-20	デバッグ用に，Virtex-II に直結されており，ユーザが自由に利用可能
J15	1-6	MultiLINX ケーブル等に電源を供給するためのコネクタ．1,2 番ピンが 3.3V, 5,6 番ピンが 5V で，3,4 番ピンが GND

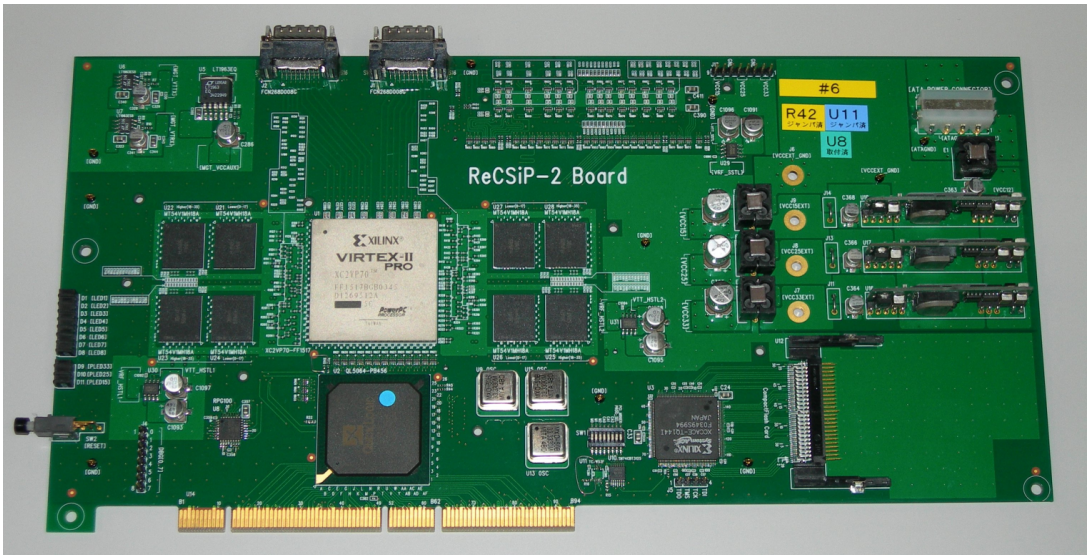


図 C.3: ReCSiP-2 Board

C.1.2 ReCSiP-2 Board

ReCSiP-2 Board (図 C.3) は，ReCSiP-1 Board で得られたノウハウを応用して設計された基板で，Virtex-II Pro と QDR-SRAM への変更が行われ，FDK 社の物理乱数生成チップが搭載されている．SRAM は QDR になって，DDR でのデータ転送になったが，データ幅は変わらず 36bit であり，DRAM は直接基板には実装せず，モジュールを基板裏面の DDR-SDRAM SO-DIMM 用のソケットに挿す形とした．

XC2VP70-5 と Micron 製の SRAM (166MHz QDR) を搭載した基板と，XC2VP70-6 と GSI Technology 製の SRAM (200MHz QDR) を搭載した基板の 2 種類が存在するが，両者は互換品であり，利用方法に差異は存在しない．SRAM は ReCSiP-1 Board 同様に 2 チップひと組，36bit 幅で FPGA に接続されており，合計 4 セットとなる．

C.1.2.1 主要部品

- FPGA: Xilinx Virtex-II Pro (XC2VP70-5FF1517C / 6FF1517C)
- SRAM: Micron 18Mb QDR-Ib2 SRAM (MT54V1MH18A) / GSI Technology 18Mb SigmaQuad SRAM (GS8180QV18) x 8
- DRAM: 200pin DDR-SDRAM SO-DIMM Socket x 1
- RNG: FDK RPG-100

C.1.2.2 スイッチ設定

表 C.2 が，ReCSiP-2 Board のジャンパ及びスイッチの一覧である．Slave SelectMAP モードでは {M0, M1, M2} = {0, 1, 1} であるため，通常 SW1 は 1 番と 7 番のみ ON にして動作させることになる．

C.1.2.3 マルチギガビットトランシーバ

Virtex-II Pro の MGT (Multi Gigabit Transceiver) に接続されているコネクタは J1, J2 であり，それぞれ 4 レーンの双方向差動チャネルを持つ．J1/J2 は Infiniband 4X 等で用いられている HSSDC コネクタであるが，基板上の配線の関係上，ピン配置は独自のもの (表 C.3) となっている．

差動シリアル信号線の受信側にはコンデンサが挿入されており，2.5Gbps 前後での利用を想定した AC カップリングになっている．

C.1.2.4 オシレータ

ReCSiP-2 Board 上には次に挙げる 3 つのオシレータソケットが存在する．

- U15: ローカルバスクロック．QL5064 と Virtex-II Pro の両方に接続される．33MHz〜66MHz の範囲で動作させることが可能
- U9: ユーザクロック: Virtex-II Pro 内部のユーザロジックをローカルバスよりも高い周波数で動作させたい場合などに利用することが可能．不要な場合はオシレータを挿入しなくてもよい

表 C.2: ReCSiP-2 Board のジャンパ・スイッチ設定

部品番号	ピン番号	機能
SW1	1-3	Virtex-II Pro の configuration mode を設定する信号．1 番ピンから順に M0, M1, M2 に接続されている．スイッチはプルアップされており，on で low, off で high になる
	4-6	SystemACE CF の configuration address を設定する信号．4 番ピンから順に CA0, CA1, CA2 に接続されている．スイッチはプルダウンされており，on で high, off で low になる
	7	On: QL5064 から configuration Off: SystemACE CF から configuration
	8	On: PCI バスを常に 33MHz で動作させる Off: PCI バスの 66MHz 動作を許可
J4	1-8	ユーザ用のデバッグ端子
J5	1-4	Virtex-II Pro および SystemACE CF の JTAG チェイン．1 番ピンから順に TDO, TMS, TCK, TDI に接続されている
J12	1-8	JTAG ケーブルなどに接続するための電源ピン．1,2 番ピンが 3.3V, 4,5 番ピンが 2.5V, 7,8 番ピンが 1.5V で，3,6 番ピンが GND である

表 C.3: ReCSiP-2 Board の MGT コネクタのピン配置

レーン	0	1	2	3
TX P	S15	S11	S7	S3
TX N	S16	S12	S8	S4
RX P	S14	S10	S6	S2
RX N	S13	S9	S5	S1

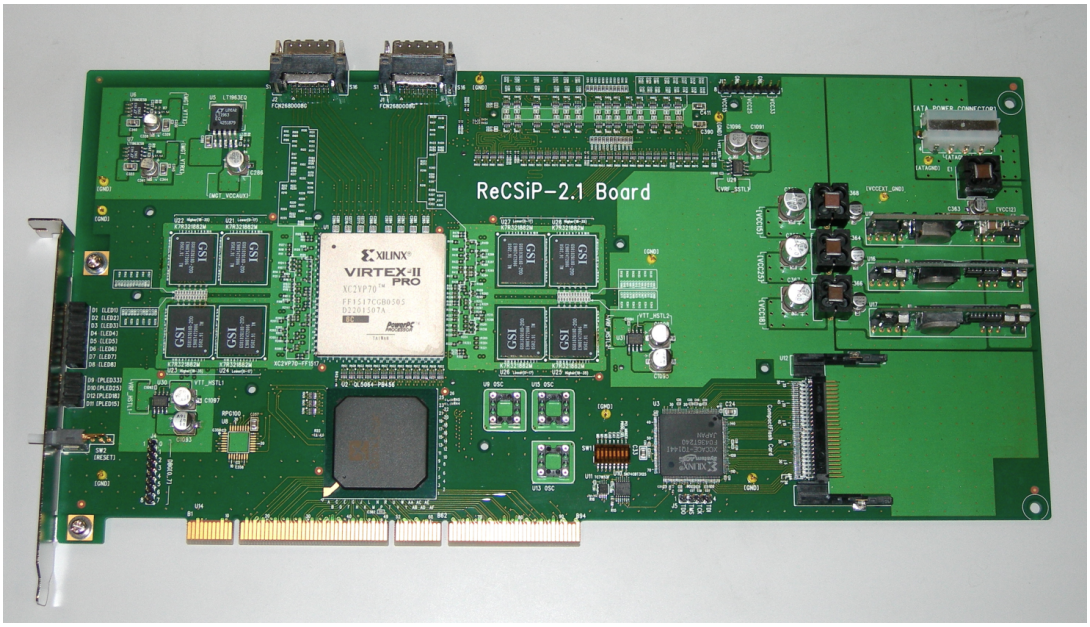


図 C.4: ReCSiP-2.1 Board

- U13: SystemACE CF 用クロック: 通常は 33MHz のオシレータを挿入する

これらの他に，MGT 用のリファレンスクロックとして U4 (EPSON EG-2121: 125.000MHz LVDS) が搭載されている．

C.1.3 ReCSiP-2.1 Board

ReCSiP-2.1 Board (図 C.4) は，ReCSiP-2 Board の FPGA のピン配置などの不具合を修正し，さらに SRAM を QDR-II SRAM 相当品に変更した基板である．搭載している SRAM は 18Mb であるが，アドレス線は 36/72/144Mb 対応分まで結線されており，交換することで容量拡大が可能になっている．

C.1.3.1 主要部品

- FPGA: Xilinx Virtex-II Pro (XC2VP70-6FF1517C)
- SRAM: GSI Technology 18Mb SigmaQuad-II SRAM (GS8182Q18) x 8
- DRAM: 200pin DDR-SDRAM SO-DIMM Socket x 1
- RNG: FDK RPG-100

ReCSiP-2.1 Board は，一部の部品が変更になった他は ReCSiP-2 Board と同等である．

表 C.4: ReCSiP Board の PCI vendor, product, revision コード

	Vendor	Product	Rev.	FPGA	SRAM
ReCSiP-1 Board	0x3614	0x3086	0x01	XC2V6000-4	Micron SyncBurst
ReCSiP-2 Board			0x02	XC2VP70-5	Micron QDR-I
			0x03	XC2VP70-6	GSI SigmaQuad
ReCSiP-2.1 Board			0x04	XC2VP70-6	GSI SigmaQuad-II

表 C.5: PCI インタフェイスのベースアドレスレジスタ設定

BAR	Address width (Size)	Type	役割
0	8 (256B)	Memory	QL5064 システムレジスタ
1	27 (128MB)	Memory	ReCSiP Board ローカルバス
2	8 (256B)	I/O	ReCSiP Board FPGA コンフィギュレーション

C.2 PCI インタフェイス

各ボードの PCI インタフェイス部には QuickLogic QL5064 を使用しており，3.3V/5V, 32bit/64bit, 33MHz/66MHz の各モードで，マスタ/スレーブとして動作可能である．Vendor, product, revision の各コードを表 C.4 に，ベースアドレスレジスタ (BAR) の設定を表 C.5 に，それぞれ示す．ベースアドレスレジスタは 3 つ使用されており，BAR0 には QL5064 自体のシステムレジスタ空間，BAR1 にはローカルバスのアドレス空間，BAR2 には QL5064 から Virtex-II/II Pro をコンフィギュレーションするためのコントローラのアドレス空間が，それぞれマップされている．

C.3 ローカルバス

C.3.1 概要

QL5064 の FPGA 部分にはローカルバスを制御するためのロジックが書き込まれ，PCI バスとローカルバスの間のブリッジとして動作する．ローカルバスは表 C.6 に示す信号線で QL5064 と Virtex-II の間を接続しており，PCI バスとはクロックが切り離されている．ローカルバスのクロックはボード上の水晶発振器から供給されており，任意の周波数のものに交換して動作させることが可能である．

なお，ローカルバスは 64bit 幅であるが，アドレス空間は 27bit である．

C.3.2 スレーブアクセス

ローカルバスの slave read/write 動作におけるタイミングを図 C.5 に示す．

スレーブアクセスでは QL5064 が PCI バスのスレーブ，ローカルバスのマスタとして動作し，Virtex-II/II Pro がローカルバスのスレーブとなる．27bit のアドレス空間はホスト CPU の物理メモ

表 C.6: ローカルバスの信号線

信号名	方向	論理	能書き
SYSCLK	in	なし	ローカルバスクロック．基板上のオシレータから QL5064 と Virtex-II のクロックピンに供給される
L_RSTX	out	負	システムリセット信号．PCI バスのリセット信号に連動して動作
L_DRQX	out	負	スレーブ転送要求信号
L_DAKX	in	負	L_DRQX に対する acknowledge
L_ADSX	i/o	負	転送サイクルの開始とアドレスの転送を示すアドレスストローブ．スレーブアクセス時は出力 (QL5064→Virtex-II) で，マスタアクセス時は入力
L_WEX	i/o	なし/負	Write 動作を示す信号で，スレーブアクセス時は出力，マスタアクセス時は入力
L_BSTMX	i/o	負	バースト転送を示す
L_RDYOX	out	負	QL5064 がデータ転送可能であることを示す
L_RDYIX	in	負	Virtex-II がデータ転送可能であることを示す
L_AD[63:0]	i/o	正	アドレス/データ信号線．スレーブアクセス時は最初に L_ADSX と同時にアドレスが転送され，続いてデータが転送される．マスタアクセス時は L_ADSX と同時にアドレスが，続いて転送バイト数が転送され，続いてデータが転送される
L_BEX[7:0]	i/o	なし/負	バイトイネーブル信号．スレーブアクセス時には出力，マスタアクセス時には入力
L_INTX	in	負	割り込み信号
L_S64X	in	負	コントロールレジスタ選択信号
L_DBSYX	out	負	PCI bus の状態を示す．Low の場合には PCI ライトマスタアクセスは行えない

* 「方向」は QL5064 からみた信号の入出力方向

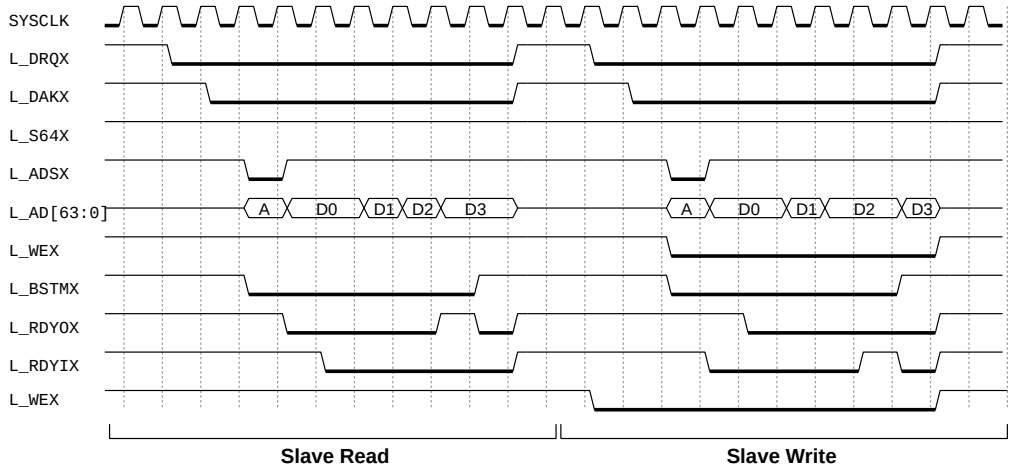


図 C.5: ローカルバスの slave read/write 動作

リアドレス空間内にマッピングされており，このアドレス空間に対して read/write を行うとアドレスの下位 27 ビットをローカルバスのアドレスとして Virtex-II/II Pro に対するアクセスが行われる．

このアクセスモードではまず QL5064 が L_DRQX を assert し，これに対して Virtex-II/II Pro が L_DAKX を assert することで転送が開始される．転送の始めにはまず ADSX と AD を用いてアドレスが転送され，その後は L_RDYIX, L_RDYOX[1]を用いてハンドシェイクを行いながら，連続したアドレス空間に 8 バイトずつデータが転送される．アドレスは 8 バイト境界に基づくため，一連の転送の最初と最後のワードが 8 バイト境界を埋めきれない場合，L_BEX を用いてバイト単位での転送になることがあるが，一連の転送の最初と最後のワード以外で L_BEX が利用されることはない．転送サイクルは L_DRQX が negate された時点で転送は終了となる．

C.3.3 マスタアクセス

PCI バスが他のデバイスの DMA によって使用されていなければ QL5064 が PCI バスのマスタ，ローカルバスのスレーブに，Virtex-II/II Pro がローカルバスのマスタとなってデータ転送を行うことができる．他のデバイスによる DMA が行われているかどうかは L_DBSYX を用いて確認することができ，この信号が High であればマスタ転送を開始することができる．

マスタ転送モードは Virtex-II/II Pro 側が L_ADSX を assert し，L_AD のビット [31:0] に転送先のアドレスを出力することで開始される．この場合の転送先アドレスは，ホスト側の 32bit の物理アドレスである．アドレスを出力した次のクロックサイクルでは L_AD のビット [23:0] に転送するバイト数を出力する．転送バイト数を出力した後はスレーブ転送時と同様に L_RDYOX, L_RDYIX を用いてハンドシェイクを行いながらデータを転送する．

なお，マスタアクセスを行う場合には，PCI-localbus ブリッジの FIFO が適切に動作するように FIFO Almost Full の閾値を初期化する必要がある．このレジスタはベースアドレスレジスタ 0 のオフセット 0x68 にあり，64bit の 0 を書き込むこで初期化が行われる．初期化動作は一度だけ行えばよいので，デバイスドライバのロード時などに実行するのが適切である．

[1] L_RDYOX, L_RDYIX は，受け手→送り手の順で準備完了を通知するための信号であり，read と write で，QL5064 と Virtex-II/II Pro のどちらが先に assert するかが異なる

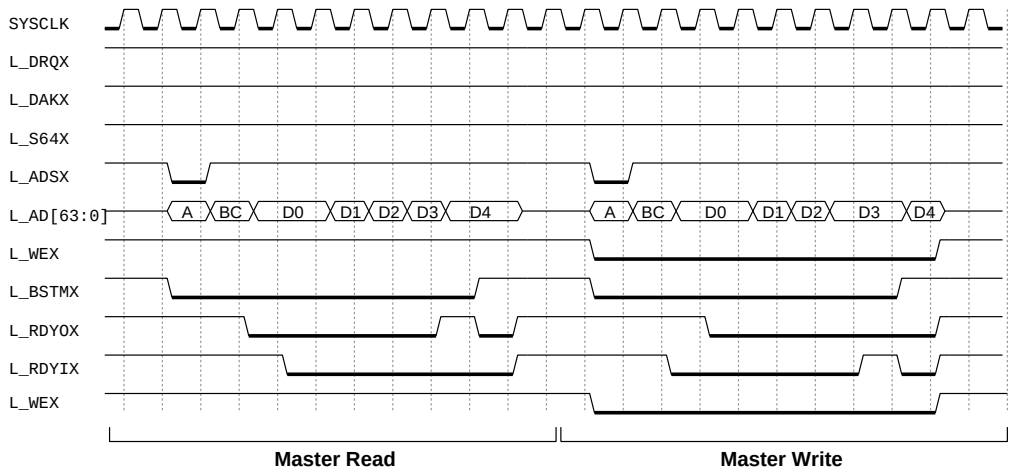


図 C.6: ローカルバスの master read/write 動作

C.4 Virtex-II/II Pro コンフィギュレーション機構

QL5064 にはローカルバス–PCI バスのブリッジの他に Virtex-II/II Pro をコンフィギュレーションするための機構も実装されており，ベースアドレスレジスタ 2 の I/O 空間にマップされている．コンフィギュレーションは Slave SelectMAP モードで行うようになっており，表 C.7 に示すような信号線で Virtex-II/II Pro に接続されている．

コンフィギュレーションを行う場合，ホスト側のソフトウェアでは

1. コントロールレジスタ (0x00) の CFR(ビット 0) に 1 を書き込んでコンフィギュレーション動作を要求
2. ステータスレジスタ (0x08) の CFA(ビット 1) が 1 になるまで待機
3. データレジスタ (0x08 のビット 7:0) に 1 バイトずつデータを書き込み
4. ステータスレジスタ (0x08) の DONE(ビット 2) が 1 であれば正常終了

という手順になる．このコンフィギュレーション機構を経由して FPGA に書き込む構成情報は Parallel PROM からコンフィギュレーションを行う場合の，PROM の内容と同じものでよいので，Xilinx のツールが生成する bit ファイルから，

```
% promgen -u 0000 example.bit -p mcs -x 32M -o example.mcs
```

のように MCS86 形式のファイルを生成することでコンフィギュレーションに必要なデータを得ることができる．

表 C.7: Virtex-II/II Pro コンフィギュレーション機構の外部信号

信号名	方向	論理	機能
PROG	out	負	FPGA に対する configuration 要求信号
INIT	in	負	FPGA の内部 RAM クリア信号．Low のとき FPGA は初期化動作中で，High になるとコンフィギュレーション可能
CCLK	out	なし	Configuration 用のクロック信号．ローカルバスのクロックを 2 分周したものが使われ，コンフィギュレーション関係の信号はこれに同期して動作する
D[0:7]	out	正	コンフィギュレーション用のデータ線．D0 が最上位ビットになる
CS	out	負	FPGA へのチップセレクト信号．CS が Low になった次の CCLK 立ち上がりからデータが取り込まれる
WRITE	out	負	D の方向を示す．この信号が Low のときコンフィギュレーションデータの書き込みとなり，High のときは読み出しとなるが，読み出しはこのコンフィギュレーション機構ではサポートされない
BUSY	in	正	FPGA 側がコンフィギュレーションデータの処理で busy になったことを示す．CCLK が 50MHz を超えない場合には無視してよい
DONE	in	正	コンフィギュレーション中は Low になり，終了すると High になる

表 C.8: コンフィギュレーション機構の制御レジスタ

アドレス	レジスタ名	ビット	名称	機能
0x00 (write)	Control	31:1	Reserved	予約
		0	CFR	Configuration request: 1 を書くことでコンフィギュレーション，0 を書くことでコンフィギュレーション機構をリセット
0x08 (read)	Status	31:8	Reserved	予約
		7	REQ	CFR の状態を読み出し．CFR はコンフィギュレーション終了で 0 になる
		6:4	Reserved	予約
		3	BUSY	FPGA に接続されている BUSY 信号
		2	DONE	FPGA に接続されている DONE 信号
		1	INIT	FPGA に接続されている INIT 信号
		0	CFA	Configuration acknowledge: FPGA のコンフィギュレーション許可ビット．CFR に 1 を書き込んだあと，CFA が 1 になってからコンフィギュレーションデータレジスタにデータを書き込む
0x08 (write)	Configuration Data	31:8	Reserved	予約
		7:0	Data	FPGA のコンフィギュレーションデータを書込み

付 録

発表論文

論文誌

1. 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”, 先進的計算基盤システムシンポジウム SACSIS’06, ACS 論文誌 15 号同時投稿 (投稿中)

2. 長名保範, 吉見真聡, 岩岡洋, 小嶋利紀, 西川由理, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “FPGA を用いた汎用生化学シミュレータ ReCSiP”, 電子情報通信学会論文誌 (査読中)

3. Yasunori Osana, Tomonori Fukushima, Masato Yoshimi and Hideharu Amano, “An FPGA-Based Acceleration Method for Metabolic Simulation”, IEICE Trans. on Information and Systems, vol.E87-D, no.8, pp.2029–2037, Aug, 2004.

国際会議

1. Masato Yoshimi, Yasunori Osana, Yow Iwaoka, Akira Funahashi, Noriko Hiroi, Yuichiro Shibata, Naoki Iwanaga, Hiroaki Kitano and Hideharu Amano, “The Design of Scalable Stochastic Biochemical Simulator on FPGA”, IEEE International Conference on Field Programmable Technology(FPT’05), pp.339-340, Dec, 2005.

2. Masato Yoshimi, Yasunori Osana, Tomonori Fukushima and Hideharu Amano, “Stochastic Simulation for Biochemical Reactions on FPGA”, The 14th International Conference on Field Programmable Logic and Applications(FPL’04), Lecture Notes in Computer Science, vol.3203, pp.105-114,Aug,2004.

3. Masato Yoshimi, Yasunori Osana, Tomonori Fukushima, Hideharu Amano, “Implementation and Evaluation of Stochastic Simulation of Chemical Reaction Model on FPGA”, CoolchipsVII, pp.83, Apr, 2004.

4. Yasunori Osana, Yow Iwaoka, Tomonori Fukushima, Masato Yoshimi, Akira Funahashi, Noriko Hiroi, Yuichiro Shibata, Naoki Iwanaga, Hiroaki Kitano and Hideharu Amano, “A Framework for ODE-Based Multimodel Biochemical Simulations on an FPGA”, Proccedings of the 15th Field Programmable Logic and its applications(FPL’05), pp.574–577, Aug, 2005.

5. Naoki Iwanaga , Yuichiro Shibata , Masato Yoshimi , Yasunori Osana , Yow Iwaoka , Tomonori Fukushima , Hideharu Amano , Akira Funahashi , Noriko Hiroi , Hiroaki Kitano and Kiyoshi Oguri, “Efficient Scheculing of Rate Law Functions for ODE-based Multimodel Biochemical Simulation on an FPGA”, Proccedings of the 15th Field Programmable Logic and its applications(FPL’05), pp.666–669, Aug, 2005.

6. Yasunori Osana, Tomonori Fukushima, Masato Yoshimi, Yow Iwaoka, Akira Funahashi, Noriko Hiroi, Yuichiro Shibata, Hiroaki Kitano and Hideharu Amano, “An FPGA-Based, Multi-model Simulation Method for Biochemical Systems”, Proccedings of the 19th International Parallel and Distributed Processing Symposium / Reconfigurable Architecture Workshop(RAW’05), Apr,2005.

研究会

1. 吉見真聡, 長名保範, 岩岡洋, 西川由理, 小嶋利紀, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “データ転送網を用いた確率モデル生化学シミュレータの FPGA への実装の検討”, 第 123 回システム LSI 設計技術 (SLDM) 研究会,VLD2005-105, 慶應義塾大学, pp.47-52, Jan. 2006.

2. 吉見真聡, 長名保範, 岩岡洋, 福島知紀, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “ ヒープ木を用いた確率モデル生化学シミュレータの FPGA への実装 ”, 第 1 回リコンフィギャラブルシステム研究会 (信学技報 RECONF2005-1-7)・京都大学 (京都), pp.37–42, May, 2005.

3. 吉見真聡, 長名保範, 岩岡洋, 柴田裕一郎, 岩永直樹, 天野英晴, “FPGA を用いた科学技術計算向け浮動小数点算術演算器の設計”, 第 26 回パルテノン研究会, 中央大学 (後楽園キャンパス), pp.49–55, May, 2005.

4. 吉見真聡, 長名保範, 福島知紀, 天野英晴, “ 確率モデルを用いた化学反応シミュレーションの FPGA による高速化 ”, 第 4 回リコンフィギャラブル研究会 長崎大学 (長崎) ,pp.220-226,Sep 2004.

5. 長名保範, 岩永直樹, 吉見真聡, 岩岡洋, 小嶋利紀, 西川由理, 舟橋啓, 広井賀子, 柴田裕一郎, 北野宏明, 天野英晴, “FPGA を用いた生化学シミュレータ ReCSiP におけるハードウェアリソース消費に関する考察”, 第 123 回システム LSI 設計技術 (SLDM) 研究会,VLD2005-107, 慶應義塾大学, pp.59-64, Jan. 2006.

6. 西川由理, 長名保範, 吉見真聡, 岩岡洋, 小嶋利紀, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “生化学シミュレータ ReCSiP における数値積分機構の性能改善手法”, 第 123 回システム LSI 設計技術 (SLDM) 研究会,VLD2005-106, 慶應義塾大学, pp.53-58, Jan. 2006.

7. 長名保範, 吉見真聡, 岩岡洋, 小嶋利紀, 西川由理, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “FPGA を用いた生化学シミュレータ ReCSiP のシミュレーション制御機構”, リコンフィギャラブルシステム研究会 (信学技報 RECONF2005-39), Sep, 2005.

8. 岩岡洋, 長名保範, 吉見真聡, 小嶋利紀, 西川由理, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “FPGA を用いた生化学シミュレータ向け SBML 処理系の構築”, リコンフィギャラブルシステム研究会 (信学技報 RECONF2005-40), Sep, 2005.

9. 岩永直樹, 柴田裕一郎, 吉見真聡, 長名保範, 岩岡洋, 福島知紀, 天野英晴, 舟橋啓, 広井賀子, 北野宏明, 小栗清, “FPGA を用いた生化学シミュレータにおける反応速度関数スケジューリングに関する検討”, 第 1 回リコンフィギャラブルシステム研究会 (信学技報 RECONF2005-1-8) 京都大学 (京都), pp.43-48, May, 2005.

10. 長名保範, 岩岡洋, 福島知紀, 吉見真聡, 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 を用いた生化学シミュレータ ReCSiP 向けの数値積分機構の実装と評価”, 第 1 回リコンフィギャラブルシステム研究会 (信学技報 RECONF2005-1-9)・京都大学 (京都), pp.49-54, May, 2005.

11. 長名保範, 福島知紀, 吉見真聡, 岩岡洋, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴, “FPGA を用いた生化学シミュレータ用の SBML 処理系の構築”, 第 153 回 計算機アーキテクチャ研究会 (SHINING 2005)・那覇市共済会館 (沖縄), 情報処理学会研究報告,Vol.2005, No. 7, pp.13-18, Jan, 2005.

12. 岩岡洋, 長名保範, 福島知紀, 吉見真聡, 舟橋啓, 広井賀子, 柴田裕一郎, 岩永直樹, 北野宏明, 天野英晴,“SBML 対応細胞シミュレータ環境の構築”, コンピュータシステム研究会第 5 回 (信学技報 VLD2004-108) 慶應大学 (横浜), pp.63-68,Jan, 2005.

13. 長名保範, 福島知紀, 吉見真聡, 天野英晴,“ FPGA を用いた細胞内代謝系のマルチモデルシミュレーション ”, 第 4 回リコンフィギャラブルシステム研究会 (長崎大学), pp49-54, Sep, 2004.

14. 長名保範, 福島知紀, 吉見真聡, 天野英晴, “FPGA による細胞内代謝系のシミュレーション”, 第 1 回リコンフィギャラブルシステム研究会・熊本大学 (熊本), pp43-48,Sep, 2003.

海外発表

1. Yow Iwaoka, Yasunori Osana, Masato Yoshimi, Toshinori Kojima, Yuri Nishikawa, Akira Funahashi, Noriko Hiroi, Yuichiro Shibata, Naoki Iwanaga, Hiroaki Kitano and Hideharu Amano, “A System Design of Accelerating Biochemical Simulations on Programmable Hardware”, The 10th Forum on Software Platforms for Systems Biology, Oct, 2005.