計算機アーキテクチャ 139－25
（２０００．８．５）

可変構造シミュレーションシステムRiSPの機能拡張

横畠正大* 田中一成*[1] 最所圭三** 福田晃*
* 奈良先端科学技術大学院大学 情報科学研究科 ** 香川大学 工学部 信頼性情報システム工学科

LSI技術の進歩の結果，従来ソフトウェアで行なわれていた処理を、ハードウェアレベルで実現することが可能となってきている。我々は，その応用の一つとして、シミュレータのハードウェア化の一手法を提案するとともに、提案手法を実現するための可変構造シミュレーションシステムRiSPの構築を行なっている。旧システムでは単純なシミュレーションモデルにのみ対応していた。本稿では、このシステムをより複雑なシミュレーションモデルに対応できるよう。システムの各部分を設計しなおし機能拡張を行った。この機能拡張により、待ち行列モデルにおいて頻繁に用いられている、複数のキュー・サーバ間の選択に対応した条件分岐をハードウェア化できるようにした。

Extension of the Reconfigurable Simulation System, RiSP

Masahiro Yokohata, Kazunari Tanaka*, Keizo Saisho**, and Akira Fukuda**
* Graduate School of Information Science, Nara Institute of Science and Technology
** Department of Reliability-based Information System Engineering, Faculty of Engineering, Kagawa University

With progress in LSI technology, programmable devices such as FPGA appear. Many processes can be executed on hardware instead of software. We have proposed a method of a reconfigurable hardware simulation system, called RiSP, and contructed the system. Since the system can execute only simple simulation models, it can not be used practically. In this paper, the system is extended to be practically used by reconsidering a design of parts of the system. By the extension of functions, the system can execute more complex simulation models such as tree type conditional branches.

1 はじめに

計算機で処理を行うとき、その処理に特化したハードウェアを用いることにより、ソフトウェアのみで同様の処理を行った場合に比べてはるかに高速に処理できる。しかし、ソフトウェアに比べて汎用性は低く、他の処理に柔軟に対応できないなどの問題が生じる。近年のLSI技術の進歩の結果、Field Programmable Gate Array(FPGA), Complex Programmable Logic Device(CPLD) などの大規模なプログラマブルロジックデバイス (PLD) が開発され、これらのデバイス上に実用規模の回路が構成可能となっている。ソフトウェアレベルでハードウェア機能を記述できるハードウェア記述言語 (HDL) と組み合わせることにより、これまでソフトウェアで行なわれていた汎用的な処理をハードウェアレベルで実現できる。我々はこのようなプログラマブルデバイスを用いて、ハードウェア構成を柔軟に変更できるシステムの研究を行なっている [1]。

我々の研究室では、その応用の一つとして、可変構造シミュレーションシステムを提案している。待ち行列を用いたシミュレーションモデルを扱う。これまで開発してきた可変構造シミュレーションシステムRiSP([rísp]: **R**econfigurable **S**imulation System with **P**rogrammable**D**evices)(以後、旧システム)では、待ち行列モデルの直列型構造とツリー型構造の無条件分岐についてのみ対応したプロトタイプであった。このプロトタイプは実用的なシミュレーションを実行できるほど機能面で充実しておらず、複雑な解析モデルへの対応が出来ていなかった。そこで、このプロトタイプの各部分を設計しなおし、複雑な解析モデルにも対応できるようにシステムの機能拡張を行った。この機能拡張により、待ち行列モデルに

[1]現在、警察庁に所属

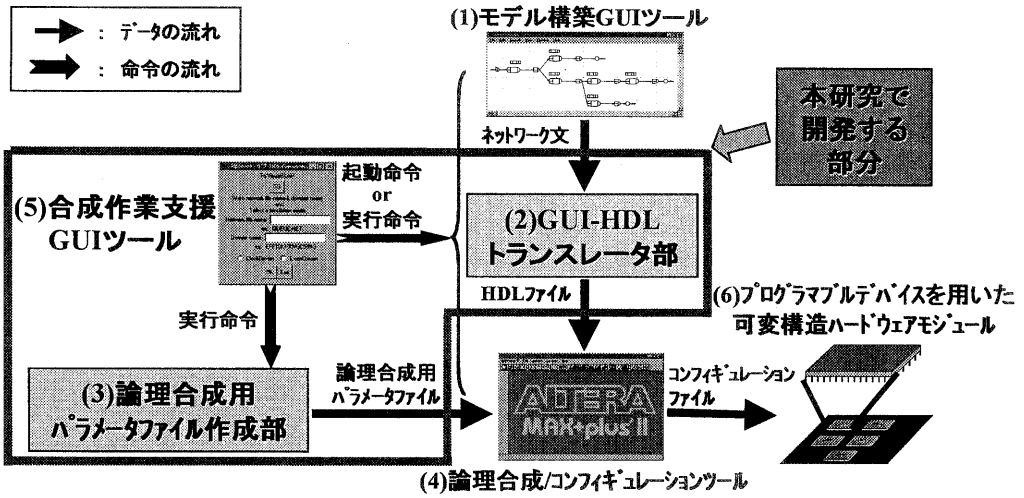

図 1: 可変構造シミュレーションシステムの構成

おいて頻繁に用いられている、複数のキュー・サーバ間の選択に対応した条件分岐を扱えるようになった。

2 可変構造シミュレーションシステムの概要

可変構造シミュレーションシステムの構成を図 1 に示す。

このうち (1) モデル構築 GUI ツール、(4) 論理合成／コンフィギュレーションツール、(6) 可変構造ハードウェアモジュールは既存のものを使用し、(2)GUI-HDL トランスレータ部、 (3) 論理合成用パラメータファイル作成部、(5) 合成作業支援 GUI ツールの開発を行っている。今回、機能拡張を行うのは (2) の GUI-HDL トランスレータ部である。

2.1 シミュレーションモデル構築 GUI ツール

モデル構築 GUI ツールは既存のツールとして、Prinsker Corporation 社製の市販汎用ツールである Visual SLAM[2] の機能の一つである Network Builder を使用する。

この GUI ツールを用いて、解析したいシステムの待ち行列モデルをグラフィカルな状態で記述する。図 2は、この GUI ツールを用いて、基本的な A/B/1 モデルの待ち行列モデルの例を表したものである。

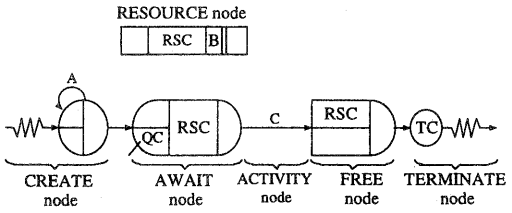

図 2: Visual SLAM で表した A/B/1 待ち行列モデル

モデル化終了後、Visual SLAM は、この図から、ネットワーク文を生成する。例えば、図 2のモデルから図 3に示すネットワーク文が生成される。この生成されたネットワーク文をもとにして、GUI-HDL トランスレータ部で、シミュレーション回路用 HDL ファイルを生成する。

2.2 GUI-HDL トランスレータ部

GUI-HDL トランスレータ部では、作成した解析モデルに従って GUI ツールが生成するネットワーク文から、実現するシミュレータの回路を HDL 記述に変換する。論理合成/コンフィギュレーションツールを考慮して、出力する HDL には VHDL を採用する。トランスレータ本体の作成には Perl を用い、トランスレータ表示部は Tcl/Tk を用いて作成した。本システムで生成する待ち行列用シミュレータの回

```
RESOURCE,1,RSC,B,{1};
CREATE,A,,,,1;
ACTIVITY;
AWAIT,1,{{RSC,1}},ALL,QC,NONE,1;
ACTIVITY,,C;
FREE,{{RSC,1}},1;
ACTIVITY;
TERMINATE,TC;
```

図 3: A/B/1 待ち行列モデルから生成されるネットワーク文の例

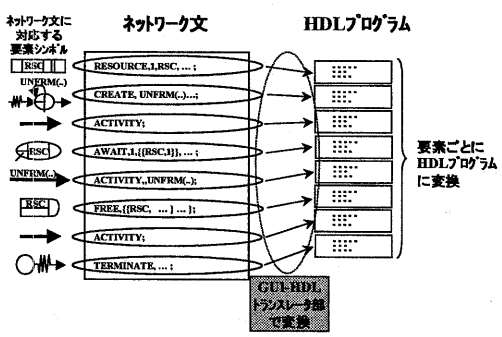

図 4: GUI-HDL トランスレータ部の概念図

路は、図 4に示すように、モデル構築 GUI ツールで作成したモデルの各要素を、回路動作方式に従った回路に置き換える形式で生成する。

この変換により、Network Builder で作成した各ノードに対応する VHDL 記述の回路コンポーネント間の接続を記述した VHDL ファイルが生成される。

2.3 論理合成／コンフィギュレーションツール

GUI-HDL トランスレータで作成した VHDL ファイルをハードウェアモジュール上に実現するためには論理合成、コンフィギュレーション作業が必要である。この作業にはアルテラ社製の論理合成・コンフィギュレーションツールである MAX+PLUS II[4] を使用する。またプログラマブルデバイスはアルテラ社製 FPGA FLEX10K130V(回路規模約 13 万ゲート) を使用している [3]。

3 分岐への対応

旧システムでは待ち行列モデルの直列型構造とツリー型構造の無条件分岐についてのみに対応したプロトタイプであった。このプロトタイプにより、速度面における有用性は証明されたが、実用的なシミュレーションを実行できるほど機能面で充実しておらず、複雑な解析モデルへの対応が出来ていなかった。そこで、本稿では、このプロトタイプの各部分を設計しなおし、待ち行列モデルにおいて頻繁に用いられている、複数のキュー・サーバ間の選択に対応した条件分岐をハードウェア化できるように新たな機能を拡張する。以下では、より複雑な待ち行列モデルへ対応するための、条件分岐に関する機能拡張について述べる。

Visual SLAM 上では、条件分岐は SELECT ノードと呼ばれるノードによって記述される。この SELECT ノードは本システムで解析対象としている待ち行列モデルにおいてはキューとサーバー間の選択に用いられる。図 5に SELECT ノードの使用例を示す。中央のノードが SELECT ノード、その左側の 2 つのノードが QUEUE ノード、右側の実線のアークが SERVICE アクティビティである。SELECT ノードを使用して、QUEUE ノード、SELECT ノード、SERVICE アクティビティの順番に接続した場合には多様な条件分岐に対応することが可能となる。SELECT ノードは、特定のキュー選択規則を指定する Queue Selection Rule(QSR) に基づいて複数の待ち行列の中からキューを選択したり、サーバ選択規則を指定する Server Selection Rule(SSR) に基づいて利用可能なサーバの中からサーバを選択するために用いられる。

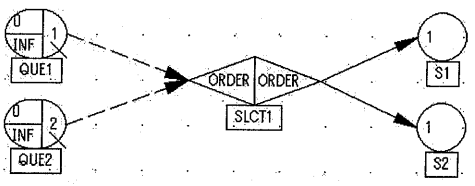

図 5: SELECT ノードの使用例

3.1 Queue Selection Rule(QSR) に基づくキュー選択

3.1.1 QSR で指定できる機能

複数のキューからどの呼をサーバへ送るかの選択方法として以下のものがある。

(1) ORDER：SELECT ノードの QUEUE ラベルの記述順に従い選択可能性を調べる。

(2) CYCLIC：サイクリックに選択可能性を調べる。前回選ばれた待ち行列の次の行列から選択する。

(3) RANDOM：ランダムに選ぶ。どの待ち行列も等確率で選ばれる。

(4) LAVERAGE：現時点までの平均滞留要素数がもっとも多い待ち行列を選ぶ。

(5) SAVERAGE：現時点までの平均滞留要素数がもっとも少ない待ち行列を選ぶ。

(6) LWAIT：待ち行列の先頭の要素の待ち時間がもっとも多い待ち行列を選ぶ。

(7) SWAIT：待ち行列の先頭の要素の待ち時間がもっとも少ない待ち行列を選ぶ。

(8) LNUM：現時点で，待っている要素の数がもっとも多い待ち行列を選ぶ。

(9) SNUM：現時点で，待っている要素の数がもっとも少ない待ち行列を選ぶ。

(10) LAVAIL：現時点で待つことの出来る余裕がもっとも多い待ち行列を選ぶ。

(11) SAVAIL：現時点で待つことの出来る余裕がもっとも少ない待ち行列を選ぶ。

3.1.2 QSR の実現

Visual SLAM には 3.1.1節で述べた機能以外にも SLAM 以外の言語で記述された外部関数を用いた QSR の機能が存在するが、それらについては設計は行っていない。なぜならそれらの機能はソフトウェア/ハードウェア両面において設計する必要があるが、可変構造シミュレーションシステムは現在のところ、シミュレーションモデルをすべて可変構造ハードウェアモジュール上で動作する仕様になっており、これらの機能の実現は難しいためである。ソフトウェア／ハードウェア・コデザインシステムへのシステム拡張は今後の課題である。

3.1.1節の (1) ～ (11) の機能を実現するためには、旧システムのキューとサーバを設計しなおし、各種情報をキュー、セレクタ、サーバに保持させるようにする。また、これらの機能を実現するためには SELECT ノードの各機能に対応したキューとサーバを個々に設計する必要がある。これらの仕様について述べる。

キュー側に必要な機能は次の通りである。

- キューの番号
- キューに滞留している呼の個数 (カウンタ (i))
- カウンタ (i) の値をクロックごとに加算していった総和（カウンタ (ii)）
- キューの最大容量の値
- 呼の発生順に付けられた通し番号

セレクタ側に必要な機能は次のとおりである。

- 前回の選択で選択したキュー番号
- サーバに空きがあるかどうかを示すフラグ
- (3) の機能を実現するために必要な乱数発生器

回路動作方式によって、これらの情報は異なった手順で格納される。

はじめに、クロックドリブン型回路の場合について述べる。本方式は、シミュレーションクロックを構成回路間の共通クロック信号と単純同期させて動作する方式である。この回路動作方式では，各情報は以下のようにして格納される。図 6に回路の構成を示す。

- 発生した呼に呼番号を付ける。この情報は呼が保持する。
- 乱数発生器から発生した呼はキューに格納され、カウンタ (i) を 1 増やす。
- セレクタから選択されると、キューは呼を放出し、カウンタ (i) を 1 減らす。
- カウンタ (ii) はカウンタ (i) のクロックごとの値を加算して格納していく。

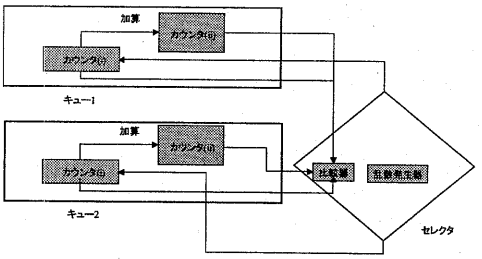

図 6: QSR 回路 (クロックドリブン)

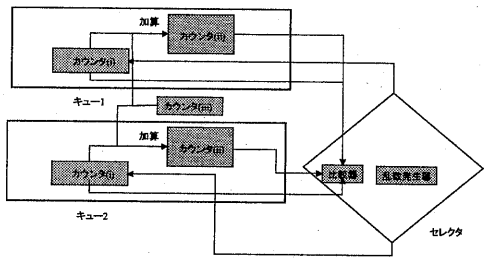

図 7: QSR 回路 (イベントドリブン)

- 比較器はカウンタ (ii) の値を比較し QSR に応じた結果をセレクタに出力する。

次に、イベントドリブン型回路の場合について述べる。本方式は、構成回路間の共通クロック信号のクロックエッジに合わせて、次にイベントが発生するまでの時間を計算して、その間の上述の無駄なシミュレーション時間をいっきに進めて実行時間を短縮する動作方式である。この回路動作方式では，各情報は以下のようにして格納される。図 7に回路の構成を示す。

- イベント発生間隔をカウントするカウンタ (iii) を設置する。
- イベントが発生する度にカウンタ (iii) をリセットする。
- 発生した呼は各キューに格納され、そのときキューのカウンタ (i) を 1 増やす。
- セレクタから選択されると、キューから呼が放出され、カウンタ (ii) を 1 減らす。
- カウンタ (ii) は各イベントごとにカウンタ (i)× カウンタ (iii) の値を加算していく。
- 比較器はカウンタ (ii) の値を比較し、各機能に応じた結果を出力する。

以上のようにそれぞれの回路動作方式で情報の格納方法は異なるが、条件分岐の各機能を実現するアルゴリズムと必要な情報は同一である。

3.2 Server Selection Rule(SSR) に基づくサーバ選択

3.2.1 SSR で指定できる機能

複数あるサーバの中から、キューから送られた呼をどのサーバへ送るかの選択方法として以下のものがある。

(1) POR：アクティビティの記述順に従い空きアクティビティを選ぶ。

(2) CYC：サイクリックに空きアクティビティを選ぶ。前回選ばれたアクティビティの次のアクティビティから順に選ぶ。

(3) LBT：現時点までの使用時間ががもっとも多いアクティビティを選ぶ。

(4) SBT：現時点までの使用時間ががもっとも少ないアクティビティを選ぶ。

(5) LIT：空き時間がもっとも長く続いたアクティビティを探す。

(6) SIT：空き時間がもっとも短く続いたアクティビティを探す。

(7) RAN：ランダムに選ぶ。どのアクティビティも等確率で選ばれる。

3.2.2 SSR の実現

SSR についても 3.2節で述べた以外に外部関数を必要とする機能が存在するが 3.1.1節で述べたように、ソフトウェア/ハードウェア両面において設計する必要があるため、現在のシステムでは実現は困難である。

3.2節の (1) ～ (7) の機能を実現するために、キューとサーバを設計しなおし、必要な各種情報をキューとサーバに保持させるようにする。また、これらの機能を実現するためには SELECT ノードの各機能に対応したキューとサーバを個々に設計する必要がある。これらの仕様について述べる。図 8に回路の構成を示す。

サーバ側に必要な機能は次の通りである.

- サーバの番号
- サーバがサービス中であるかどうかを示すフラグ (i)
- フラグ (i) が上がってから下るまでのクロックをカウントするカウンタ (i)
- カウンタ (i) の値の最大値を保持するレジスタ (i)
- サービス終了時のシミュレーションクロックを保持するレジスタ (ii)

セレクタ側に必要な機能は次のとおりである。

- 前回の選択で選択したサーバ番号
- (7) の機能を実現するために必要な乱数発生器

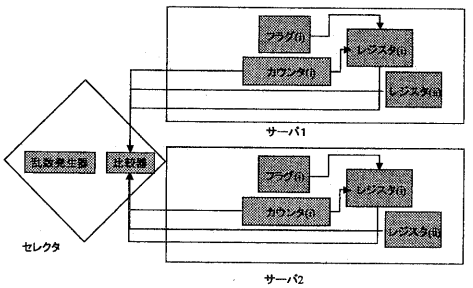

図 8: SSR 回路

4 まとめ

本稿では、可変構造シミュレーションシステムの機能拡張について述べた。

旧システムでは待ち行列モデルの直列型構造とツリー型構造の無条件分岐についてのみに対応したプロトタイプであった。このプロトタイプにより、速度面における有用性は証明されたが、実用的なシミュレーションを実行できるほど機能面で充実しておらず、複雑な解析モデルへの対応が出来ていなかった。本稿では、このプロトタイプの各部分を設計しなおし、システムの機能拡張を行い、待ち行列モデルにおいて頻繁に用いられている、複数のキュー・サーバ間の選択に対応した条件分岐をハードウェア化できるように新たな機能を拡張した。

現在、機能拡張部分についての設計および一部の実装が終了している。またシミュレーション実行中の各ノードにおける統計情報を取得する手段が実装されていないため、機能拡張した部分についての評価ができるまで至っていない。本稿で述べた機能拡張部分は現在も開発中であり、統計情報取得の機能も併せて実装し機能拡張後のシステムの評価を行う予定である。

可変構造シミュレーションシステムは現在のところ、シミュレーション回路全体をひとまとまりとしてイベントドリブン型回路で構成しており、システム全体が逐次処理になってしまう。このため、呼の発生間隔などを基に並列処理を行なうことのできる部分に分けて、各部分をイベントドリブン型回路にするなどの工夫が必要である。今後は、この問題点を考慮しつつ、大規模な待ち行列モデルシミュレーションが実行できるシステムに拡張する予定である。

参考文献

[1] 田中 一成, 最所 圭三, 福田 晃, “プログラマブルデバイスを用いた可変構造シミュレーションシステム”, 情報処理学会アーキテクチャ研究会, 99-ARC-134, pp.43-48, 1999.

[2] 構造計画研究所, “Visual SLAM システム解説書”, 1997.

[3] 日本アルテラ株式会社, “FLEX10K データシート”, 1998.

[4] 日本アルテラ株式会社, “MAX+PLUS II オンライン・ヘルプ”, 1998.