計算機アーキテクチャ 134－8
（1999．8．2）

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の開発

田中 一成 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

概要

LSI技術の進歩により、これまでソフトウェアで行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。本稿では、その応用の一つとして、待ち行列を用いた可変構造シミュレーションシステムを提案し、その実現方法について述べる。シミュレーションは、種々のパラメータを少しずつ変え、何度も実行するので、多大なシミュレーション時間を費やすため、そのハードウェア化は非常に有益である。クロックドリブン型とイベントドリブン型の2つの方式を設計した。また、各方式について、RTLでの記述と論理合成の作業の際に得られた予測周波数を用いて評価した。その結果、イベントドリブン型に関しては、提案した方式はソフトウェアで行う方式に比べ300倍以上の性能が出た。

Reconfigurable Simulation System with Programmable Devices

Kazunari Tanaka, Keizo Saisho, and Akira Fukuda
Graduate School of Information Science, Nara Institute of Science and Technology

Abstract

With progress in LSI technology, many problems solved at software-level can be solved at hardware-level. In this paper, we propose a method of reconfigurable hardware simulation system using queuing model as one of applications of them and describe the system architecture. Since simulation takes a huge simulation time because iterative execution varying parameters are performed, it is suitable for the subject of the proposed system. We design two methods: a clock driven type and an event driven type. Both methods are evaluated using RTL-level description and an expected clock frequency obtained at synthesizing logic. As the result, for event driven type, the performance of our method marks three hundred times over the performance of the method with software only.

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と比較してはるかに高速に処理できる。一方、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

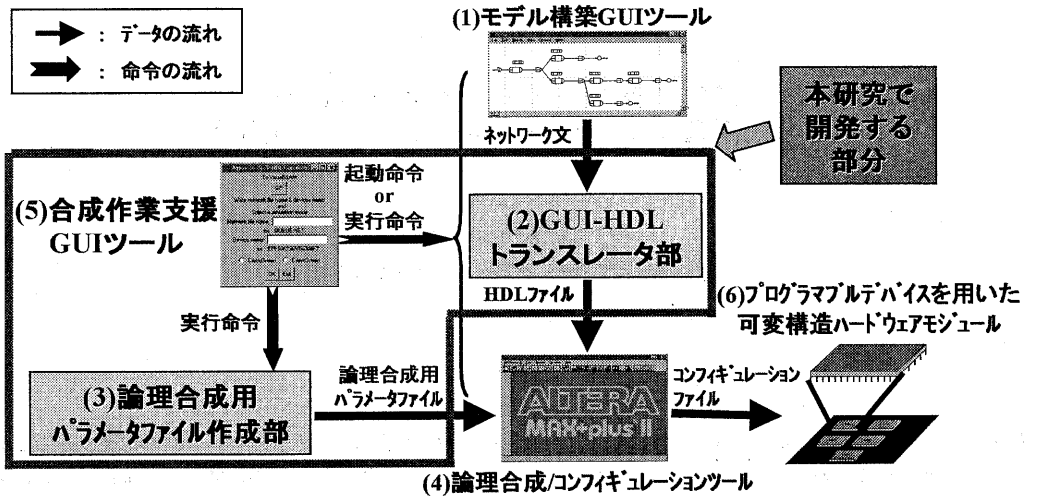

図 1: 本システムの構成

なった。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) を用いることにより、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することが可能となってきた。そのため、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。

本稿では、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、その設計、実装、および、評価について述べる。シミュレーションモデルとしては、待ち行列を用いたシミュレーションを扱う。待ち行列を用いたシミュレータを用いて解析を行なう場合、種々のパラメータを少しずつ変えて何度も実行するため、多くのシミュレーション時間を要する。このようなシステムをハードウェア化できれば、実行時間を激減させることが期待できる。また、シミュレータ自体は変化させないので、同じハードウェアを用いることができる。本システムでは、クロックドリブン型とイベントドリブン型の 2 つの方式について設計し、論理合成作業の際に得た各モデルの回路の予測動作周波数と、各回路を RTL(Register Transfer Level) 記述した VHDL ファイルを用いて評価を行なった。本稿では、紙面の都合のため、イベントドリブン型回路における評価のみ載せる。

2 可変構造シミュレーションシステムの概要

本節で、本システムの概要を述べる。本システムの構成を図 1 に示す。

(1) シミュレーションモデル構築ツール
(2) HDL トランスレータ
(3) 論理合成用パラメータファイル作成部
(4) 論理合成ツール
(5) 合成作業支援 GUI ツール
(6) ハードウェアモジュール

このうち (1), (4), (6) は既存のものを使用し、(2), (3), (5) の開発を行う。このシステムを用いたシミュレーション実行は以下の流れになる。

(i) 解析対象をモデル構築ツールを用いて、待ち行列ネットワークにモデル化する (図 1(1))。
(ii) 合成作業支援 GUI ツールに、合成作業のための必要事項を入力する (図 1(5))。
(iii) (ii) で入力したデータを基にして、GUI-HDL トランスレータ部と論理合成用パラメータファイル作成部で、合成作業に必要なシミュレータの回路を記述したハードウェア記述言語 (HDL) ファイルと、合成パラメータを記述したファイルを生成する (図 1(2)(3))。

(iv) (iii) で生成したファイルを用いて、論理合成作業を行ない、コンフィギュレーション用データを生成する (図 1(4))。

(v) コンフィギュレーション用データを用いてコンフィギュレーション作業を行ない、プログラマブルデバイスを用いた可変構造ハードウェアモジュール上にシミュレータの回路を実現する (図 1(4))。

(vi) 可変構造ハードウェアモジュールシステム上でシミュレーションを実行し、その結果を収集する (図 1(6))。

3 シミュレーションモデル構築 GUI ツール

シミュレーションモデル構築ツールは既存のツールを使用する。本システムでは Prinsker Corporation 社製の市販汎用ツールである Visual SLAM[3] の機能の一つである Network Builder を使用する。我々の研究室ではマルチプロセッサシステムにおけるスケジューリングアルゴリズムの評価のために Visual SLAM を使用しており、このソフトウェアに関するノウハウが蓄積されている。このため、HDL へのトランスレータの作成の際、他のシミュレーションツールを用いた場合よりも容易になると考え Visual SLAM を使用することにした。

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。

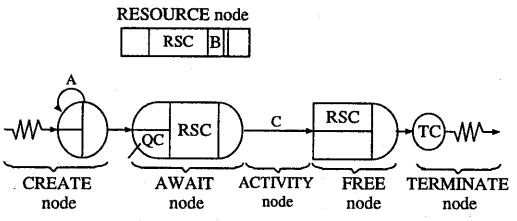

図 2: Visual SLAM で表した A/B/1 待ち行列モデル

図 2 では、待ち行列ネットワーク内での各イベントの処理の流れをモデル化している。CREATE ノードにおいて、到着間隔分布を指定 (図中の記号 A) して呼の発生を行ない、AWAIT ノードにおいて、待ち行列 (Queue) の最大容量と使用する RESOURCE ノードを指定している。RESOURCE ノードでは、同時に何個のイベントが滞在できるかを指定している (図中の記号 B)。そして、ACTIVITY ノードで、サービス時間分布を指定 (図中の記号 C) しており、その後の FREE ノードで、RESOURCE ノードの解放の制御を行なっている。また、TERMINATE ノードでは、シミュレーション終了の条件として、サービス終了数を指定できる。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 のモデルから生成されたネットワーク文の記述は図 3 のようになる。この生成されたネットワーク文をもとにして、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。

```
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

4 可変構造シミュレーションシステムの実装

4.1 HDL トランスレータ

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を用いて、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。図 4 はトランスレータのユーザーインターフェースを示す。本システムで生成する待ち行列用シミュレータの回路は、図 5 に示すように、モデル

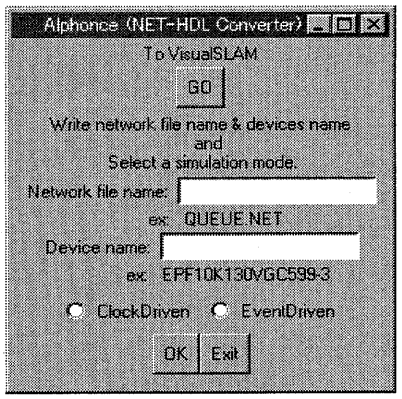

図 4: 合成作業支援ツール (HDL トランスレータ)

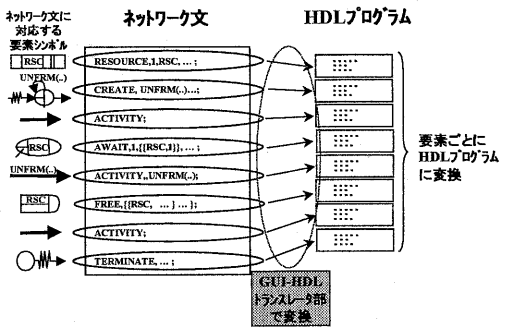

図 5: HDL トランスレータ部の概念図

```
CREATE ノードの変換
CREATE port map (AR => CREATE_FLG, XO => CR_XO,
                 X1 => CR_X1,X2 => CR_X2,
                 CLK => CLK, RESET => RESET);

AWAIT1 ノードの変換
AWAIT port map (Q_CNT => Q_CNT1, AR => RS_AR1,
                TK => TK1,CLK => CLK,
                RESET => RESET);

ACTIVITY ノードの変換
ACTIVITY port map (AR => AC_AR1, XO => AC1_XO,
                   X1 => AC1_X1,X2 => AC1_X2, SET =>
                    SET1, CLK => CLK, RESET => RESET);
FREENODE
FREE port map (FREE_FLG => FREE_FLG1, AR =>
               AC_AR1, CLK => CLK,RESET => RESET);
```

図 6: HDL プログラムへの変換例

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 ファイルになる。図 3 のネットワーク文の変換例を図 6 に示す。

4.2 論理合成作業

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。この作業には市販のツールである MAX+PLUS II を使用する。MAX+PLUS II 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 [5]。またプログラマブルデバイスはアルテラ社製 FLEX10K130V(回路規模約 13 万ゲート) を使用している [4]。

図 7: 論理合成作業の概要

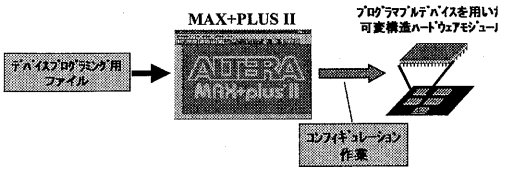

図 8: コンフィギュレーション作業

論理合成用パラメータファイル作成部では、合成作業支援ツール (図 4) で入力されたパラメータを基にして、論理合成作業に用いる ACF(Assignment & Configuration File) を生成する。コンフィギュレーション用出力ファイルの内容に変更を及ぼす可能性がある情報は、すべて ACF に記述される [5]。

このツールを用いて、HDL トランスレータで生成した VHDL ファイルと、論理合成用パラメータファイル作成部で生成した ACF の 2 つのファイル

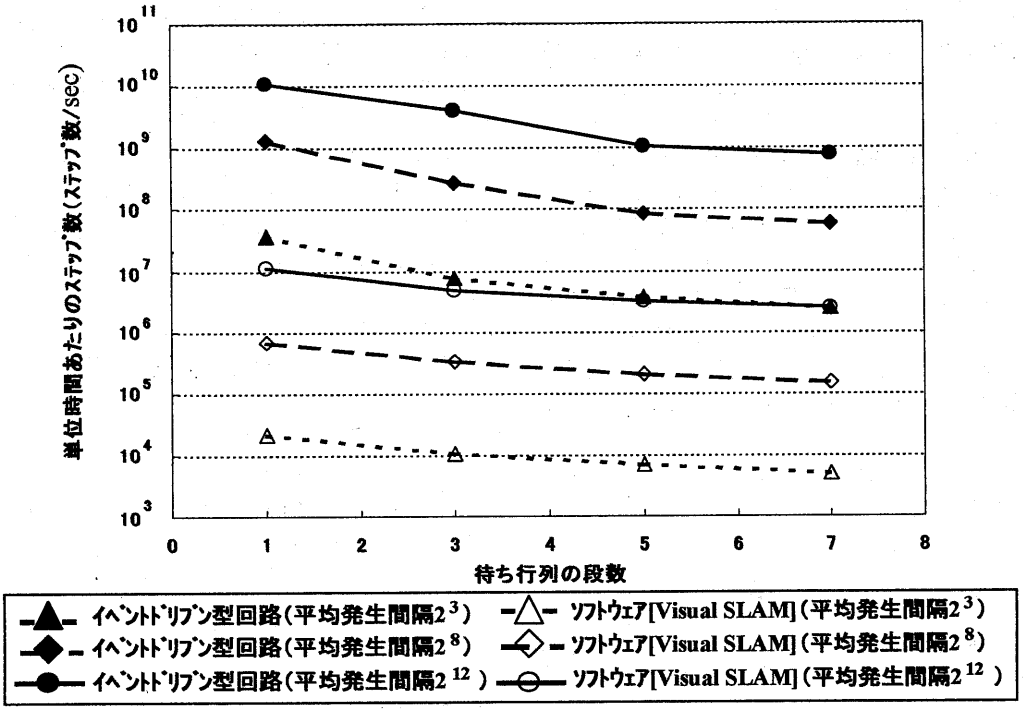

図 9: ソフトウェアとイベントドリブン型回路の性能比較

を利用して論理合成を行ない、デバイスプログラミング用の出力ファイルを作成する。

その後、図 8 に示すように、作成されたデバイスプログラミング用ファイルを使用し、コンフィギュレーション作業を行なう。

5 実行結果及び評価

本節では、従来のソフトウェアシミュレーションとイベントドリブン型でのハードウェアシミュレーションについて、様々な組み合わせによる性能比較、および、その考察を行なう。評価対象モデルには、直列型モデルである GI/G/s/256 待ち行列モデル ($s = 1, 3, 5, 7$) を用いる。

性能を表す単位には、単位時間あたり (1 秒あたり) に計算処理できるシミュレーションクロック数 (ステップ数) を用いる。シミュレーションを行なう場合、ユーザは、シミュレーション時刻やサービス数をシミュレーション終了の値として用いる。さらに、ユーザが、シミュレーションが高速処理 “できる”、“できない” を判断する際の基準には、実際にシミュレーションを実行した時間を用いる。よって、“1 ステップ/sec” を性能を表す単位を用いることにする。

現在、実際にシミュレーションを実行して結果を得る段階まで実装が完成していないため、本システムの性能は論理合成作業の際に得た各モデルの回路の予測動作周波数と、各回路を RTL(Register Transfer Level) 記述した VHDL ファイルを用いて行なったシミュレーションを使用して算出している。文献 [1] においては、乱数発生部を実現出来ていなかったので、動作周波数を文献 [2] を参考にして決定して評価していた。待ち行列の段数により変化するが、イベントドリブン型回路における予測動作周波数は、段数が 1 段の場合は 7.77MHz であり、5 段の場合は 10.43MHz であった。

比較対象とするソフトウェアシミュレーションにはモデル構築 GUI ツールとして用いた、Visual SLAM (ソフトウェアツール) で同じモデルをシミュレー

ションしたものを用いる。ソフトウェアシミュレーションはPentiumII 300MHz, メモリ384MBを搭載したパーソナルコンピュータ上で実行した。

図9に、ソフトウェアシミュレーションとイベントドリブン型回路を比較した結果を示す。呼の平均発生間隔を変化させてシミュレーションを行った。

イベントドリブン型回路は、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する。ソフトウェアシミュレーションでも同様の方法でシミュレーションを行っている[3]。イベントドリブン型回路には、動作プロセスの中に比較プロセスを含む。

この図から、ソフトウェアシミュレーション、イベントドリブン型回路のどちらの場合においても、以下のことがわかる。

- 呼の平均発生間隔が短くなるにつれて、単位時間あたりのステップ数が減少する。
- 待ち行列の段数が増加するにつれて、単位時間あたりのステップ数が減少する。

単位時間あたりのステップ数の減少の割合が、ソフトウェアシミュレーション、イベントドリブン型回路のどちらもほぼ同じであり、イベントドリブン型回路はソフトウェアシミュレーションと比較して、常に300倍以上の性能を保っている(最小で314倍)。

イベントドリブン型回路は動作プロセスの中に比較プロセスを含むので、待ち行列の段数が増加するに従い、比較プロセス部分がボトルネックになる。また、呼の平均発生間隔が短くなる場合も同様に、ボトルネックになる。そのため、待ち行列の段数が非常に多段になるか、または、呼の平均発生間隔が極端に短くなると単位時間あたりのステップ数が減少する。

以上の結果より、今回評価に用いた直列型モデルについては、図9から分かるように、ソフトウェアシミュレーションに対するイベントドリブン型回路のシミュレーション計算処理能力の優位性が判明した。これは文献[1]における性能予測を検証している。

6 まとめ

本稿では、プログラマブルデバイスを用いた可変構造マシン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、その構成について述べた。また、RTL(Register Transfer Level)記述したVHDLファイルを用いて、本システムの評価も行なった。その結果、イベントドリブン型動作方式を用いると、ソフトウェアで行なう方式に比べ常に300倍以上速度向上が見込めることが分かった。今回、可変構造モジュール上での評価が出来なかったのは、シミュレーション回路とコンピュータとの入出力回路の実装ができていないためであり、今後、開発していく予定である。現在、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、複雑なアルゴリズムを必要とする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] 野口 裕, 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, 98-ARC-131, pp.83-88, 1998.

[2] 山本 欧, 柴田 裕一郎, 天野 英晴, "可変構造を持つ, 並列システム解析用モデルシミュレーションシステム", 並列処理シンポジウム JSPP'98 論文集, pp.235-302, 1998.

[3] 構造計画研究所, "Visual SLAM システム解説書", 1997.

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

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