様式 C-19

科学研究費補助金研究成果報告書

平成２１年３月３１日現在

研究種目：基盤研究（Ｃ）
研究期間：2006～2008
課題番号：１８５００００４３
研究課題名（和文）　モジュールの依存関係に基づく並列論理設計検証

研究課題名（英文）　Parallel Logic Design Verification Based on Module Dependence

研究代表者
平石　裕実（HIRAISHI HIROMI）
京都産業大学・コンピュータ理工学部・教授
研究者番号：４００９３２９９

研究成果の概要：形式的論理設計検証手法として記号モデル検査を取り上げ、モジュールの依存関係や内部の制御構造に着目して変数順を決めるアルゴリズムを提案し、これにより効率を数倍から数百倍向上出来ることを示した。また、分割した遷移関係を用いるアルゴリズムの中で、値が変化しない変数を早期に削除することにより、数倍から数十倍の効率向上が得られることを示した。さらに、並列論理関数処理として動的並列化法を提案し、その有効性も示した。

交付額

（金額単位：円）

	直接経費	間接経費	合　計
２００６年度	1,500,000	0	1,500,000
２００７年度	700,000	210,000	910,000
２００８年度	900,000	270,000	1,170,000
年度			
年度			
総　計	3,100,000	480,000	3,580,000

研究分野：情報工学
科研費の分科・細目：情報学　計算機システム・ネットワーク
キーワード：設計検証、モデル検査、モジュール分割、共有二分決定グラフ、
並列アルゴリズム

１．研究開始当初の背景

(1) 大規模な論理システムの設計においては、設計誤りを犯す危険性が高く、設計の正しさを確認する設計検証の必要性が認識されていた。設計検証の手法として従来用いられていた論理シミュレーションでは、全てのケースについてシミュレーションを行うことは不可能であり、シミュレーションを行わなかったケースに対して論理システムの正しさを示すことは不可能である。このため、設計の正しさを数学的手法を用いて証明する形式的設計検証手法の実用化が緊急かつ重要な課題となっていた。

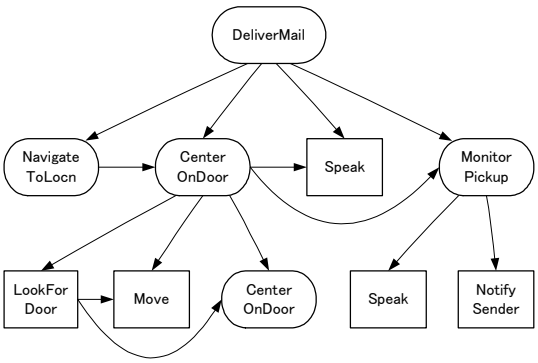
(2) 大規模論理システムは、一般に種々のモジュールを相互に結合する形で設計することが多く、例えば、システム全体を一つのVLSI で実現する SoC (System on Chip)では、IP core と呼ばれるモジュールを組み合わせて設計が行われる。さらに、IP core 自体は、種々の論理ブロックを組み合わせて構成されている。一方、ソフトウェアの分野でも、各ソフトウェアモジュールが互いに同期的あるいは非同期的に動作する並列システムが多数開発されていた。このようなシステムの設計検証では、各モジュールの状態数は比較的小さくても、システム全体の見かけ上の

状態数がモジュール数の指数に比例して大きくなる状態爆発が実用化への大きな壁となっていた。
(3) 設計検証システムを稼働させるコンピュータシステムとして、PC クラスタや、ハイパースレッディングやデュアルコア技術を用いた並列コンピュータが比較的安価に構築できるようになってきていた。

２．研究の目的
(1) 本研究の主要目的は、並列システムを構成する各モジュール間の依存関係を考慮することにより効率よく設計検証を行うアルゴリズムを明らかにすることである。特に、最も実用性が高いと注目されている記号モデル検査法を中心に取り上げ、その効率化法を明らかにする。
(2) 状態遷移関係を表す特徴関数をモジュールの依存関係に基づいて分割することによる効率化の可能性を探り、種々の分割化の下でのモデル検査アルゴリズムを明らかにする。
(3) 記号モデル検査で用いる論理関数処理には共有二分決定図(BDD, Binary Decision Diagram)が用いられることが多いが、BDD の節点数の爆発が記号モデル検査法の実用化の壁になっている。そのため、状態集合を一つの論理関数ではなく論理関数ベクトルで表現する方法や、状態遷移の制御構造に着目した変数順序の導入等により、節点数の爆発を抑えるアルゴリズムを探る。また、論理関数処理として BDD の代わりに充足可能性判定(SAT)を用いる手法についても検討する。
(4) 近年安価に構成することが可能になってきた PC クラスタや、ハイパースレッディングやマルチコア技術による並列コンピュータを利用して、並列モデル検査アルゴリズムも明らかにする。

３．研究の方法
(1) モジュールの依存関係に基づく記号モデル検査の効率化
　各モジュールが互いに非同期的に動作する並列システムでは、システム全体の遷移関係は各モジュールの遷移関係の和として表現することができ、また、各モジュールが互いに同期的に動作する並列システムでは、システム全体の遷移関係は各モジュールの積として表現できる。そこで、モジュール間の依存関係として、依存する変数の個数等による重みを導入し、依存度の強いモジュール同士をグループ化することによりシステム全体の遷移関係をバランスよく分割するアルゴリズムを構築する。
(2) 分割された遷移関係を用いた記号モデル検査
　遷移関係の分割により、分割された各遷移関係は、一般に、一部のシステム状態変数にしか依存しなくなり、また、値が変化しない状態変数も含まれることになる。このような性質に着目して、スムージング演算の適用範囲を絞ることにより、記号モデル検査の基本演算である像計算の効率化を図る。
(3) BDD の節点数爆発に対する対処
　上記(2)の手法により、BDD の節点数もかなり抑えらると考えられるが、遷移関係の制御構造に着目し、モジュールの依存関係だけでなく変数の依存関係にも着目することにより、より効果的に BDD の節点数の爆発を抑えることが可能になると考えられるため、そのような手法の評価を行う。
(4) 記号モデル検査の並列アルゴリズム
　記号モデル検査の基本演算である像計算に用いられる論理関数処理として、BDD 処理の並列アルゴリズムを構築し、これをベースにして記号モデル検査の並列化に取り組む。

４．研究成果
(1) モジュールの依存関係をベースにした順序付けによる記号モデル検査の効率化
　複数のモジュールが並行に動作するシステムでは、システム全体の遷移関係は各モジュールの遷移関係の和や積の形で表すことができる。このような場合、関係の深いモジュール同士を近くにまとめる形でグループ化することにより、記号モデル検査の効率を向上させることが出来ると考えられる。



　上図は、このようなモジュールの依存関係を示すグラフである。図中、矢印がモジュール間の依存関係を示している。このような複数モジュールからなるシステムの記号モデル検査では、各モジュールの遷移関係を順番に処理していくことになるが、その際に、関係の深いモジュールから順に処理していくことが重要である。そこで、モジュールのある処理順において、互いに関係の深いモジュール同士の処理が各々i番目とj番目に行われるときのコストを $2^{|i-j|}$と定義し、それらのコストの総和が最小になるような処理順で記号モデル検査を行う方法を提案し実験を行った。コストを考慮しない場合(順番無考慮)

と考慮した場合(順番考慮)の検証時間を以下の表に示す。

検証対象	検証時間		比率
	順番無考慮	順番考慮	
Dm13s	101.81	3.45	29.5
Dm14s	466.92	3.61	129.3
Dm15s	1073.74	3.74	287.1
Dm13a	56.90	33.51	1.7
Dm14a	79.01	49.90	1.6
Dm15a	106.44	69.93	1.5
Tt06s	8.33	2.78	3.0
Tt07s	16.71	3.90	4.3
Tt08s	24.19	4.34	5.6

この表より、検証対象にも大きく依存するものの、導入したコストの総和を最小にすることにより、モデル検査にかかる時間が数倍から数百倍短縮されており、本手法の有効性を示している。

(2) 分割された遷移関係による記号モデル検査アルゴリズム

遷移関係を分割することにより、分割された遷移関係は一般にシステム全体の一部の状態変数にしか依存しなくなるため、これを用いることにより記号モデル検査の効率を向上させることが出来る。また、各モジュールが互いに非同期に動作するようなシステムでは、分割された遷移関係には部分的には値が変化しないような変数が多数含まれることになる。このような場合、記号モデル検査の基本演算である逆像計算において、事前にそのような変数に対する処理を行っておくことで、記号モデル検査の効率向上が期待できる。具体的には、従来法では

$$g_{i+1}(\vec{s}) := \exists \vec{p}.\exists \vec{s}.(R(\vec{p},\vec{s},\vec{s}') \cdot g_i(\vec{s}))|_{\vec{s}' \leftarrow \vec{s}}$$

のような不動点計算を繰り返していたのに対し、まず、事前に、$R^0(\vec{s},\vec{s}') := \exists \vec{p}.R(\vec{p},\vec{s},\vec{s}')$ を求めておき、

$$g_{i+1}(\vec{s}) := \exists \vec{s}.(R^0(\vec{s},\vec{s}') \cdot g_i(\vec{s}))|_{\vec{s}' \leftarrow \vec{s}}$$

なる不動点計算を繰り返す手法を提案した。この提案手法の実験結果を以下に示す。

検証対象	検証時間		比率
	手法非適用	手法適用	
Dm06	217.18	8.69	25.0
Dm08	561.10	13.54	41.4
Dm10	1286.79	20.06	64.1
Dm12	2652.46	39.14	67.8
Dm14	5023.60	82.36	61.0
Tt06	149.10	7.27	20.5
Tt08	401.82	12.86	31.2
Tt10	1247.25	25.69	48.6
Tt12	5060.93	82.07	61.7

この表より、本手法により数十倍の速度向上が得られることが分かる。

(3) 遷移関係の制御構造に着目した記号モデル検査の効率化

記号モデル検査による設計検証システムとしては、CMU で開発された SMV (Symbolic Model Verifier)が有名であるが、SMV における BDD の変数順序は

① デフォルトは変数の宣言順に決定
② 動的に BDD の変数順を決める Dynamic Variable Ordering も使用できるが、非常に時間がかかり非実用的
③ より良い変数順を求めるヒューリスティックは実装されていない

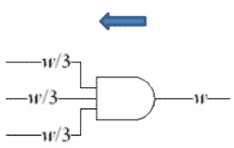
であるため、BDD の節点数爆発を生じやすい。一般に、BDD の変数順を決めるヒューリスティックとしては、

① 影響力の大きい変数を上位に持ってくる
② 互いに関係の深い変数は出来るだけ隣接させる
③ 遷移関係を表す関数を、互いに重複しない変数グループに依存する関数に分解できる場合は、変数をグルーピングした上で、各グループ内でそのグループに属する変数の順序を決定

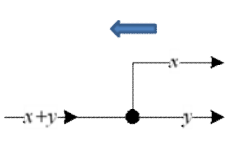
などの方法が良いと考えられる。

そこで、これらを実現するヒューリスティックとして、

① 変数出現回数：SMV において次状態を決める論理式の記述において、出現回数が多いもの(影響力が大きい変数と考えられる)を BDD の上位変数とする
② 静的グルーピング：各変数の次状態を決める論理式の条件構造に着目し、各条件部分と値部分で変数のグルーピングを行い、同一グループに属する変数同士は互いに依存するとみなし、他の変数との依存度が高いものを BDD の上位変数とする
③ 動的グルーピング：まず静的グルーピングで他の変数との依存度が最も高いものを BDD の上位変数として選択してその変数を取り除く。この操作の繰り返しで、変数の順序を決定する
④ 静的重み付け法：論理回路で表現された論理関数の出力側から入力側に向けて、下図のように各信号線に重み付けを行い、入力信号線の重みの降順に順序を決める。



ゲートの重み付け



分岐点の重み付け

⑤　動的重み付け法：まず静的重み付け法で重み付けを行い、入力信号線で最大の重みを持つものをBDDの上位変数として選択し、その信号線を取り除く。この操作の繰り返しで、変数の順序を決定する。

等のヒューリスティックの比較を行った。

検証対象M1～M10に対して上記の①～⑤を用いた時の検証時間(秒)を下表に示す。下表において⓪はヒューリスティックを用いなかった場合であり、空欄は3600秒で打ち切ったことを示す。また、黄色で塗りつぶした部分は、各検証対象に対する最小時間の部分である。

	⓪	①	②	③	④	⑤
M1	1.43	1.09	1.11	1.16	0.81	1.10
M2	6.63	0.36	0.37	0.39	0.55	0.57
M3	10.3	0.56	2.36	1.42	1.49	0.50
M4		0.07	0.08	0.08	0.05	0.07
M5		332	606	467	262	257
M6	112	0.76	0.76	1.05	1.49	1.74
M7	9.50	0.51	2.11	1.33	1.32	0.44
M8	88.4	69.7	69.9	69.9	69.8	69.9
M9		0.21	0.21	0.20	0.20	0.18
M10	3501	4.78	3.47	4.22	3.05	1.53

この結果より、一概にどれかのヒューリスティックが最適であるとは言えないが、何もヒューリスティックを用いない場合に比べて、いずれの場合も検証時間が短くなっているのがわかる。また、用いたヒューリスティック間の差はあまり大きくなく、一方で、検証対象によっては、何もヒューリスティックを用いない場合に比べて大幅な速度向上が見られる。また、検証に用いられたBDDの節点数についても、検証時間の場合とほぼ同様の結果が得られた。

(4) BDDの並列処理

記号モデル検査の並列アルゴリズムとして、BDD処理の並列アルゴリズムを構築した。

まず最初に、シャノン展開に基づき一部の論理変数の心理値を固定することにより BDDを分割し、分割された各々のBDDの処理を行う静的変数固定法の評価を行った。

14 ビットの乗算器の BDD の構築では、64並列で約 37 倍の高速化率を達成できた。また、BDD を用いた充足可能性判定問題では、64並列で約13～47倍の加速率が得られた。静的変数固定法では、真理値を固定する変数の選び方が全体の効率に大きく影響するが、どの変数の真理値を固定すればよいかを事前に求めるのは、一般に困難である。

そこで、BDD の処理の途中で、節点数の増加を極力抑えるように真理値を固定する変数を順次決定しながら並列処理を行う二種類の動的変数固定法を構築した。

①　計算の進行レベルに応じた動的固定変数選択：

全体の計算をいくつかのステップに分割して考え、あるステップまでは、すべてのCPU(またはコア)で同じBDDの構築を進める。そのステップの計算が終了した時点で、BDDの節点数が最も多い変数を一つ選び、半分のCPUでその変数値を0に固定し、残り半分のCPUではその変数の値を1に固定して計算を進める。次のステップが終了したら同様に変数を一つ選択し、その変数値を0に固定するグループと1に固定するグループに分割する。この操作を、各グループに属するCPUが一つになるまで繰り返し、それ以降は分割操作を行わずに計算を継続する。

n	並　列　度					
	1	2	4	8	16	32
12	1.00	1.72	3.05	4.90	7.55	11.3
13	1.00	1.79	3.33	5.64	9.69	15.5
14	1.00	1.98	4.34	7.96	13.0	23.3

この方法を乗算器のBDD作成に適用した時の高速化率を上表に示す。これより、14ビット乗算器で並列度4の時に4.34倍に高速化され、スーパーリニア効果が得られていることが分かる(黄色で塗りつぶした部分)。

②　BDDの総節点数に基づく動的固定変数選択：

静的固定変数選択の場合と同様に全体の計算をいくつかのステップに分割するが、各ステップの計算終了時点でのBDDの節点数を求め、それがある値(SizeLimit)以上であれば、そのBDDにおいて節点数が最も多い変数を一つ選び、その変数の値を固定する。次にSizeLimitの値を一定比率(IncRatio>1)倍して次のステップの計算にうつる。このような操作を、各グループに属するCPUが一つになるまで繰り返し、それ以降は分割操作を行わずに計算を継続する。

対象	並　列　度					
	1	2	4	8	16	32
M1	1.00	1.75	3.83	9.75	20.0	40.1
M2	1.00	1.91	3.89	8.40	9.61	18.6
M3	1.00	2.75	5.86	12.1	23.1	44.8

この方法をBDDを利用した充足可能性判定問題に適用した時の高速化率を下表に示す。これより、より多くのケースでスーパーリニア効果が得られていることが分かる(黄色で塗りつぶした部分)。

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔学会発表〕（計３件）

① 平石裕実、記号モデル検査器 SMV における BDD の変数順について、DTC 研究会、平成 21 年 3 月 29 日、箱根千代田壮
② 平石裕実、SAT アルゴリズムを利用した記号モデル検査について、DTC 研究会、平成 20 年 3 月 2 日、京都ガーデンパレス
③ 平石裕実、記号モデル検査のスケールアップについて、DTC 研究会、平成 19 年 3 月 3 日、別府豊泉荘

６．研究組織

(1) 研究代表者

平石　裕実（HIRAISHI HIROMI）
京都産業大学・コンピュータ理工学部・教授
研究者番号：４００９３２９９

(2) 研究分担者

なし

(3) 連携研究者

なし