TechnicalReportson Informationanc
ComputerSciencefrom Kochi
Vol. 8 (2016),No. 9

並列処理によるアルゴリズム高速化手法の研究

Acceleration Method of Algorithms by Parallel Processing

山崎貴也　　村岡 道明
Takaya Yamasaki　　Michiaki Muraoka
高知大学 理学部（情報科学コース）

1. まえがき

近年,マルチコアプロセッサが普及し,マルチコアを用いた並列アルゴリズムの高速化の研究[1]が行われている.本研究では先行研究で提案した簡易論理合成手法を並列化し,マルチコアを用いて理想的に並列に動いた場合と OpenMP で実際に並列に動いた場合とを比較,評価することで,理想的な並列環境と実環境による高速化率を考察する.

2. 簡易論理合成アルゴリズム

本手法は比較的小規模の論理回路の最適化（最小化）を対象とする簡易な簡約化手法である．本合成アルゴリズムはクワインマクラフスキー法を基本として考えられている.本研究ではこの簡易論理合成アルゴリズムを並列化して,高速化を試みる.

3. 並列化

本研究では合成手法を並列に並べ,部分回路のグループを並列に実行させる．図 1 には,部分回路への分割方法を示す.

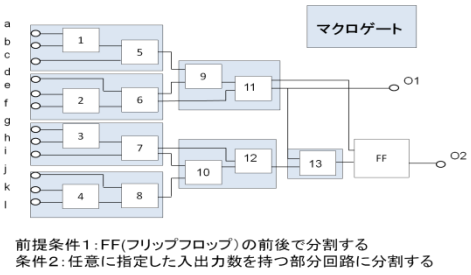

図 1．分割方法

図 1 では入力数制限を 3 入力とし,前提条件として FF の前後で分割している.各分割した部分回路をマクロゲートという.マクロゲートを各並列に割り当てて,並列に論理合成を行う.並列に合成を行う図を図 2 に示す.

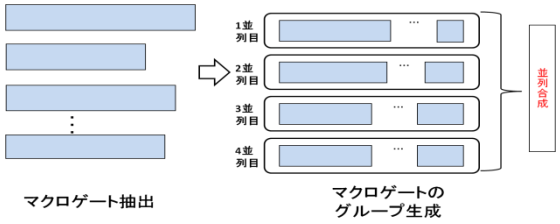

図 2．割り当てのイメージ図

図 2 は割り当てのイメージ図である.マクロゲートをグループに分け,各グループを並列に合成を実行する.

4. 評価

評価はターゲットプロセッサ上での理想的な並列実行環境をシミュレーションできる VisualSpec と,実際の実機上での並列実行ができる OpenMP と二つの環境で実験を行う.VisualSpec での評価結果を表 1 に,OpenMP での評価結果を表 2 に示す.

表 1．評価結果（VisualSpec）

	実効時間/CPUTime [msec]			高速化率	
回路名	逐次(a)	4並列(b)	8並列(c)	a/b[倍]	a/c[倍]
decode16	199	60	44	3.32	4.52
encode16	312	98	68	3.18	4.59
pc	96	33	26	2.91	3.69
sftreg16	787	207	105	3.80	7.50
fadder8	122	33	17	3.70	7.18
eccgc4	141	50	30	2.82	4.70
spwm16	1,100	279	142	3.94	7.75
dwnct16	340	94	61	3.62	5.57
sfregsel32	691	177	95	3.90	7.27
sramspi	772	223	134	3.46	5.76
alu	949	251	139	3.78	6.83
cmbfunc16	3,551	937	500	3.79	7.10
平均	755	204	113	3.52	6.04

表 2．評価結果（OpenMP）

	実効時間/ElapsedTime[ms]			高速化倍率	
回路名	逐次(a)	4並列(b)	8並列(c)	a/b[倍]	a/c[倍]
decode16	43	26	29	1.65	1.48
encode16	38	23	26	1.65	1.46
pc	33	21	22	1.57	1.50
sftreg16	49	30	32	1.63	1.53
fadder8	40	23	25	1.74	1.60
eccgc4	46	29	31	1.59	1.48
spwm16	100	56	59	1.79	1.69
dwnct16	75	49	50	1.53	1.50
sfregsel32	87	49	52	1.78	1.67
sramspi	81	52	56	1.56	1.45
alu	99	59	62	1.68	1.60
cmbfunc16	212	127	133	1.67	1.59
平均	75	45	48	1.65	1.55

VisualSpec での評価結果を見ると,高速化率の平均は 4 並列時に 3.52 倍,8 並列時に 6.04 倍となった.一方で OpenMP での評価結果は高速化率の平均は 4 並列時に 1.65 倍,8 並列時に 1.55 倍となった.

5. まとめ・考察

VisualSpec では並列数を上げる毎に実行時間が削減出来ることが推定できた．OpenMP では実プロセッサ数と同数の並列数までしか,実行時間が減っていかなかった.使用できるプロセッサ数が増えると,OpenMP でも並列数を上げる毎に実行時間を削減できると考えられる.

6. あとがき

均等な負荷をコアへ割り当てる手法の検討や,メニーコア CPU での実験・評価,合成アルゴリズムの GPGPU を用いた並列化の適応が今後の課題である.

参考文献

[1]竹内勇矢, 村岡道明, ”マルチコアを用いた高速並列論理シミュレーション手法” SLDM/VLD/Reconf 合同研究会 2015 年 1 月