Technical Reports on Information and
Computer Science from Kochi
Vol. 8 (2016), No. 6

簡易合成法と並列処理を用いた論理合成法の高速化法

An Acceleration Method of Logic Synthesis using Compact Synthesis and Parallel Processing

蘆苅　将大　豊永　昌彦　村岡　道明
Masahiro Ashikari　Masahiko Toyonaga　Michiaki Muraoka
高知大学大学院　理学専攻(情報科学分野)

1. まえがき

現在, LSI やシステムの微細化や大規模化が進んでいる. それに伴い, 回路の遅延時間や面積の大きさが問題となっている. そこで問題となっているクリティカルパス部分の最適化を高速に行う必要がある. 本稿では論理回路を部分的に高速に合成し,さらに最適化する手法を提案する.

2. 簡易論理合成手法

今回の手法は市販ツール(論理合成ツール)で合成した回路を部分回路(マクロ)に分割し, 再合成を行う. 部分回路に分割することで高速な再合成を可能とする.　本最適化合成手法を用いた設計フローを以下に示す.

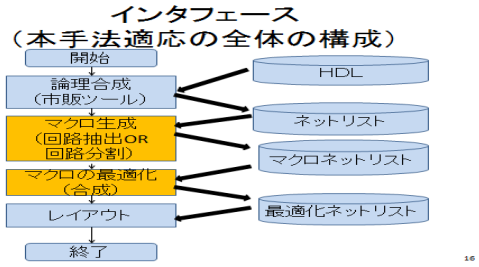

図 1 最適化合成フローチャート

図 1 のマクロの最適化合成について次に詳述する.

マクロに分割された部分回路の最適化を行う基本アルゴリズムは多出力の場合を含め, 以下の5ステップで構成される.

(ⅰ) 全入出力の真理値表から出力が1になる行を取り出しそれらを並べた表をつくる.

(ⅱ) 表の中の行のハミング距離が1になるものを総当たりに調べ, 2 つの行を 1 つにマージする.

(ⅲ) ハミング距離が1のものが無くなるまで(ⅱ)を繰り返す.

(ⅳ) 出力が1になるカバー数が少ない項を含むものから選択し, 論理式を得る.

(ⅴ) 多出力の場合, 論理が共通する部分を同一の回路として共有化する.

3. 並列化

簡易合成方法を並列化手法に対応させるには回路をどのようにコアに割り当てて並列化するかが問題となる.

今回の論理合成手法は、出力が 1 になる行を総当たりにマージを行っていく。そのため、出力が1になる行が多いとマージ時間も指数関数的に上がっていく。そこで出力が1になる行が各並列間で平坦化するようにマクロゲートを割当てる。割りてる例を図 2 に示す。

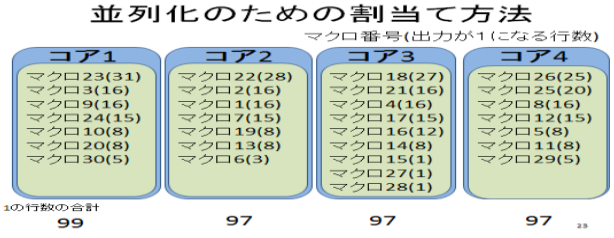

図 2 割当て方法

4. 評価

性能評価としていくつかの回路に対して本手法を適応した結果をゲート数で比較する. また, 本手法の処理速度を計測した. マルチコアによる並列化結果は逐次処理と4 および 8 並列での処理速度を比較した.

表 1 ゲート数比較

回路名	回路分割のマクロの入出力数	変換後のゲート数	市販論理合成ツールのゲート数
decode16	3	31	32
	4	32	
	5	33	
	6	31	
encode16	3	69	58
	4	61	
	5	59	
	6	57	
pc	3	76	72
	4	84	
	5	91	
	6	90	
sftreg16	3	55	74
	4	66	
	5	67	
	6	62	
fadder8	3	112	63
	4	113	
	5	114	
	6	112	
eccgc4	3	179	107
	4	126	
	5	155	
	6	214	
spwm16	3	285	143
	4	201	
	5	189	
	6	181	

表 2　高速化評価

回路名	逐次[ps](a)	4並列[ps](b)	8並列[ps](c)	a/b[倍]	a/c[倍]
decode16	198,918,520,000	60,170,680,000	44,092,840,000	3.3	4.5
encode16	312,523,840,000	98,405,880,000	68,696,520,000	3.2	4.5
pc	95,754,280,000	33,252,200,000	25,745,200,000	2.9	3.7
sftreg16	787,797,280,000	207,100,640,000	105,478,720,000	3.8	7.5
fadder8	122,262,320,000	33,123,720,000	17,506,400,000	3.7	7
eccgc4	141,411,960,000	49,914,000,000	30,379,760,000	2.8	4.7
spwm16	1,096,184,080,000	278,634,680,000	142,170,000,000	3.9	7.7

5. まとめ・考察

対象回路を小規模回路に分割し、分割された回路を簡易論理合成方法により高速に合成する手法を提案した. さらに, マルチコアプロセッサを用い, 並列処理を行うことで更なる高速化を図った。本手法を実用的な回路に適応した結果, ランダム回路については手設計とほぼ同等の結果を得ることができた。ゲート数が増加してしまう回路の理由は, 市販ツールでの四則演算やシフト演算などを行う一般的な回路では, あらかじめ手設計で設計された最適化されている回路をライブラリに入れており, そのライブラリの回路を用いることでよい回路を生成している. そのため, そのような一般的な回路では, 市販ツールと比べゲート数が増えたと考える. さらに, マルチコアによる高速化を適応すると, 4 並列で最大で 3.9 倍, 8 並列で最大 7.7 倍とコア数に応じた高速化を実現することができた. しかし, 中には理想的な高速化率が得られない回路に関しては、マージ部分がコア間で差が出ないように割当てたが、必ずしも計算量に比例するとは限らないためと考えられる.

6. 今後の課題

今回の評価より, 並列や回路に応じた最適な割当て方法, 入出力・並列数の増加, 大規模回路への対応, 消費電力や遅延時間が課題として挙げられる.