Varchsyn(5): 論理多段化手法

6N−5

中村 祐一　若林 一敏　藤田 友之　吉川 浩　前田 直孝

NEC

1 はじめに

論理合成システム Varchsyn においては、テクノロジ独立な回路規模の最小化に論理多段化を用いている。論理多段化は回路の論理的に共通な部分を抽出して回路規模を減少させていく最小化手法である。Varchsyn の論理多段化は、対称関数の考慮、論理式の否定を考慮したブール代数的多段化を短い処理時間で実行することができる。本稿では、この論理多段化手法、及びベンチマークデータを用いた Varchsyn 上での評価結果について述べる。

2 論理多段化手法

論理多段化は、多段論理最小化の1手法であり、$a+bc$ のような同じリテラルを含まないカーネルと呼ばれる積和、及び abc のような積項を論理回路から選択し、選択したカーネル、積項で論理回路をある変数 x とその否定 $\bar{x}$ を別の変数として取り扱う代数的な除算を行ない、共通論理の抽出を行なうことによって、回路規模を減少してゆく手法が知られている [1]。この手法では、代数的な除算を行なうため回路規模の減少能力に劣るため、ブール代数の性質を利用した論理多段化手法も提案されている [2] が、長い計算時間が必要である。

Varchsyn においては、代数的除算を行なった場合と同程度の計算時間で、代数的除算の場合に比べて、回路規模がより小さな回路を合成するために、以下の3つの手法を組み合わせて論理多段化を実行している。

否定共有化 抽出する因子の肯定と否定の両方を考慮

ダブルキューブ共有化 抽出時に積和の種類に制限を与える

対称関数最適化 対称関数の性質を用いて論理を最小化

以下に上記の各手法について詳しく述べる。

2.1 否定共有化

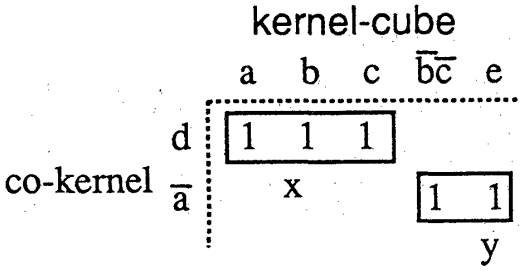
カーネルを用いた論理最小化の場合は、回路全体に関して、そのリテラル数を最大に減少させるようなカーネルを次々に選び出して代数的除算を行なう。リテラル数を最大に減少させるカーネルを選び出すためには、カーネルの積項を列に、カーネルによって代数除算を行なった場合の商を行に表した kernel-cokernel incidence 行列 (図1) を作成し、行列の各要素にそのカーネルの積項で除算したときのリテラル数を重みとして割り当て、その行列上において、最大の重みを持った長方形を選択する [3]。$f=da+db+dc+\bar{a}\bar{b}\bar{c}+\bar{a}e$ の場合は、図1のように、長方形が選択され、除算が行なわれて、のようにリテラル数10で、多段化される。

Varchsyn では、従来の代数的な除算による多段化、すなわち論理式 f を因子 x で除算する場合 $f=xg+h$ に加えて、因子 x の否定 $\bar{x}$ による除算をも考慮する。すなわち $f=xp+\bar{x}q+r$ のような多段化について、回路全体のリテラル数を最大減少させるような、因子 x を選択する。この場合、リテラル数の減少が最大になるような因子を選択できるように、上記の行列上の選択手法を拡張している。

このような因子の選択には、図2のように、kernel-cokernel incidence 行列の下部に、回路内に存在する論理式のうち、比較的変数の数が小さく否定計算が困難でないものを kernel と同様に否定行として与え、行列の要素にその因子で除算した場合に減少するリテラル数を重みとして付与した行列を作成する。

そして、その1つの否定行の要素をすべて含む最大重み L_c の長方形と、肯定部の最大重み L_a を比較して、$L_c > L_a$ の場合は、否定の因子 $\bar{x}$ を選択して除算を行ない、そうでない場合は従来と同じく x を用いて除算を行なう。$f=da+db+dc+\bar{a}\bar{b}\bar{c}+\bar{a}e$ の場合は、図2のようにのようにリテラル数7で、多段化される。

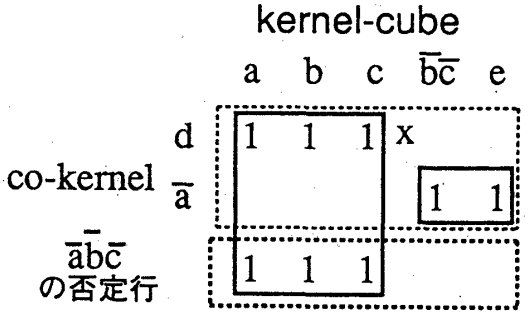
$$f=da+db+dc+\bar{a}\bar{b}\bar{c}+\bar{a}e$$



$$\begin{aligned} x &= a+b+c \\ y &= \bar{b}\bar{c}+e \\ f &= dx+\bar{a}y \end{aligned}$$

図1 kernel-cokernel incidence 行列

$$f=da+db+dc+\bar{a}\bar{b}\bar{c}+\bar{a}e$$



$$\begin{aligned} x &= a+b+c \\ f &= dx+\bar{x}+e \end{aligned}$$

図2 否定考慮 kernel-cokernel incidence 行列

この処理は、カーネルだけではなく積項に関しても同様に実行される。また、計算時間は否定を求める因子に制約を持たせていることから、従来のカーネル、積項の抽出とほぼ同等である。

2.2 ダブルキューブ共有化

ダブルキューブ共有化 [4] は、2つの積項を $D_{i,j,k}$, (i: 片方の積項のリテラル数, j: 他方の積項のリテラル数, k: 2つの積項の変数の総和) と表したとき、$D_{1,1,2}, D_{2,2,2}, D_{2,2,3}, D_{2,2,4}$ のみを積和の代数除算対象とし、積項と、$D_{i,j,k}$ の除算によるリテラル数の減少分がより大きいもので除算する手法で、除算対象因子に制限を加えることにより、1つの論理式の積項の数が非常の大きい場合を除きカーネルを用いた手法より高速に実行できる。

Varchsyn(5): Logic Decomposition Method
Y. Nakamura, K. Wakabayashi, T. Fujita,
H. Yoshikawa, N. Maeda
NEC Corporation

また、$a\bar{b}+\bar{a}b \in D_{2,2,2}$ に関して、その否定を考えた場合、$\overline{a\bar{b}+\bar{a}b} = \bar{a}b+\bar{a}\bar{b} \in D_{2,2,2}$ のようにお互いの否定が $D_{i,j,k}$ 内に存在する場合がある。このような関係は、

- $c \in D_{2,2,2}$ ならば、$\bar{c} \in D_{2,2,2}$
- $c \in D_{2,2,3}$ ならば $\bar{c} \in D_{2,2,3}$
- $c \in D_{1,1,2}$ ならば、$\bar{c}$ は2変数の積項

のような場合があり、これらに関しては、選択した因子の肯定 x と否定 $\bar{x}$ の両方で除算することとし、リテラル数の減少分はその和で評価する。

[4] では除算候補となる $D_{i,j,k}$ 及び、除算の場合のリテラル数の減少分の見積もりを除算が行なわれるたびごとに全て求める必要があるが、Varchsyn のダブルキューブ共有化では、同じ積項を表す部分をあらかじめ求めておき、その部分を除去することによって、除算後も $D_{i,j,k}$ などをすべて計算し直すことなく、インクリメンタルに実行することによって高速化を図っている。これにより、否定共有化よりも高速に多段化が実行される。

2.3 対称関数最適化

回路を表すブール式が対称関数である場合には、その性質を利用した最適化手法が提案されている [5]。この手法は以下に示すように入力の数が3または、2の recorder と呼ばれる基本的な対称関数を定義し、それを中間ノードとして、対称な部分を置き換えていく手法を用いている。recorder は、4種類提案されており、回路規模をより最小化するような recorder を選択する。

$$
\begin{aligned}
W_1(a,b):&\\
w_1 &= ab\\
w_2 &= \bar{a}b+a\bar{b}\\
W_2(a,b):&\\
w_1 &= a+b\\
w_2 &= ab\\
W_3(a,b,c):&\\
w_1 &= ab+bc+ca\\
w_2 &= \bar{a}b\bar{c}+a\bar{b}\bar{c}+\bar{a}\bar{b}c+abc\\
W_4(a,b,c):&\\
w_1 &= a+b+c\\
w_2 &= a(b+c)+bc\\
w_3 &= abc
\end{aligned}
$$

例えば、$W_3(a,b,c)$ において、w_1 は、3個の対称な入力に対して、入力の1の数が2または3の場合の論理式を表し、w_2 は、入力の1の数が1または、3の場合の論理式を表す。この $W_3(a,b,c)$ を用いて、入力の1の個数が2の場合にのみ出力が1になるような対称関数 $f = ab\bar{c}+a\bar{b}c+\bar{a}bc$ は、

$f = w_1\overline{w_2}$

$w_1 = ab+bc+ca$, $w_2 = \bar{a}b\bar{c}+a\bar{b}\bar{c}+\bar{a}\bar{b}c+abc$ に変換される。

しかし、この [5] 手法では3変数の recoder を優先的に選択するため、例えば、7変数の対称回路の場合必ず (3,3,1) のように対称変数が分割され recorder に変換され、(3,2,2) のように対称変数を分解したほうが良い結果が得られる場合を見逃してしまう。Varchsyn の論理多段化では、[6] のような経験的なパラメータを用いて、recorder に変換される対称変数の組を決定し、部分的な対称関数に対しても最適な入力変数の分割を求めることができる。

3 アルゴリズム

例えば MCNC91 のベンチマーク回路の rd73(完全対称関数)を各種手法で多段化すると、以下のようなリテラル数となる。(使用計算機は 37Mips)

手法	リテラル数	計算時間 (sec)
対称関数のみ	32	2
ダブルキューブのみ	132	8
否定共有化のみ	108	14 sec
ダブルキューブ + 否定共有化	110	10

このように、Varchsyn 上の3つの最適化手法については実験的に以下のようなリテラル数減少に関する性質が得られた。

1. 与えられた関数が対称関数である場合には、対称関数最適化は、他の手法にくらべて数分の1のリテラル数を持つ回路を合成する。
2. ダブルキューブ共有化のほうが、否定共有化より高速に実行できる。
3. ダブルキューブ共有化のみでは、共有すべき部分回路の共有化洩れが発生する。しかし、否定共有化をその後に行なうことにより、それらは完全に共有化される。

この性質から、まず対称関数最適化を行ない、次にダブルキューブ共有化と、否定共有化をその手法でのリテラル数の減少分に関する閾値 TH で切替える。

```
if (network に対称性がある){
        対称変数に関して対称関数最適化を実行する;
}
foreach_thresh-hold{
        while(ダブルキューブのリテラル数減少分 >= TH){
                ダブルキューブ共有化;
        }
        while(否定共有化でのリテラル数の減少分 >= TH){
                否定共有化;
        }
}
```

4 評価

NEC 製のワークステーション EWS4800/230(37Mips) を用いて Varchsyn 上の論理多段化システムの評価をベンチマーク上で行なった。リテラル数は、fanout 1 のものを圧縮し、EXOR, EXNOR のリテラル数を 2 として計算を行なっている。

data	多段化後	計算時間 (sec)
5xp1	124	29
duke2	393	42
sao1	44	3
sao2	142	9
bw	199	21
rd53	20	1
rd73	32	3
vg2	87	13
9sym	44	6

5 おわりに

Varchsyn における論理多段化の手法について述べた。本手法は、半代数的な論理多段化の手法を高性能かつ短い計算時間になるように、改良し組み合わせることによって、高い性能を得ることが確認された。

参考文献

[1] R. K. Brayton et al., "MIS: A Multiple-Level Logic Optimization System", IEEE Transaction on CAD, CAD-6 July 1987, pp1062-1081.

[2] K. A. Bartlett et al., "Multilevel logic minimization using implicit don't cares." IEEE Trans. CAD, CAD-7 1988, pp723-740.

[3] G. De Micheli et al., "Design System for VLSI Circuits : Logic Synthesis and Silicon Compilation" Martinus Nijhoff Publishers, 1987, pp197-248.

[4] J. Rajski, J. Vasudevamurthy, " Testability Preserving Transformations in Multi-level Logic Synthesis", IEEE ITC 1990, pp265-273.

[5] B. G Kim, et. al, " Multilevel logic synthesis of Symmetric switching functions," IEEE Trans. CAD Vol CAD-10, pp436-446, 1991

[6] 中村他,「多段論理最小化における対称関数の利用に関する一考察」, 情処第 43 回全大, 1991