7E−5

端子位置を考慮した論理多段化の一手法およびその評価

矢野 純一† 坂本 健† 安浦 寛人‡ 田丸 啓吉†

†京都大学工学部 ‡九州大学大学院総合理工学研究科

1 はじめに

従来のLSIの論理合成ツールではそのほとんどが回路のトポロジーの情報のみを考えて回路の合成を行っており、回路のレイアウトなどのジオメトリーの情報は考慮していないのが普通であった。これは、論理設計の段階でレイアウトを考慮しようとすると、問題が複雑になってしまうためである。しかし論理設計の段階でレイアウトを考慮することが出来ると、最終的に出力されるチップの性能(レイアウト面積、遅延)の向上を図ることが出来ると思われる[1][2]。

本稿ではそのようなレイアウトを考慮した論理多段化の一手法として、端子位置を考慮した論理多段化という手法を提案する。そして回路の多段化時に使用する評価関数を従来のリテラル数に加えて入力変数間の距離というパラメータを加えたものとし、それを実際に論理合成ツールに組み込んでその有効性を確かめている。また、その論理合成結果をレイアウト合成ツールでレイアウトにすることによって、この手法が最終的なレイアウトに対して有効に働いていることも確かめている。

2 端子位置を考慮した論理多段化

今回提案する端子位置を考慮した論理多段化においては、以下の事を仮定している。

1. 回路のレイアウトの形状は矩形である
2. 回路のレイアウトはスタンダードセル(ゲートアレイ)で実現される
3. 回路の論理は2段簡単化された形で与えられる
4. 回路のレイアウト情報として、回路のレイアウトの大きさ(縦、横の寸法)、外部入出力端子位置(レイアウトの周上にある)が与えられる

そしてこのような仮定のもとで、できるだけレイアウト面積(ゲート面積、配線面積)を小さくするような多段論理回路を生成し、結果としてネットリストおよび各素子の概略的な位置を出力する。

3 端子位置を考慮した論理多段化の処理

3.1 入力変数間の距離

従来、論理多段化の際の回路規模の評価関数としてはリテラル数が主に使用されている。しかしこれだけでは回路のレイアウトに関する情報を考慮することはできない。

A Method of Logic Synthesis with Consideration for Terminal Location and its Evaluation

Junichi YANO†, Takeshi SAKAMOTO†, Hiroto YASUURA‡, and Keikichi TAMARU†

†Kyoto University, ‡Kyushu University

そこで新たな評価関数として入力変数間の距離を導入する。論理関数 f に対する入力変数間の距離とは、外部入力変数及び中間変数によって表されている論理関数 $f(x_1, x_2, x_3, \cdots x_n)$ に対して

$$D(f) = \max_{1 \le i,j \le n} d(x_i, x_j)$$

で定義されるものである。ただし $d(x_i, x_j)$ は変数 x_i と x_j に相当する外部入力端子や、ゲートのレイアウト上の座標間の距離である。そしてこのような入力変数間の距離 D の大小によってレイアウトにおける配線の影響を考慮することができる。

3.2 多段化手法および評価関数

今回提案する論理多段化において使用している多段化手法[3][4]には以下に述べるようなものがある。そして各手法に対して、その評価関数に前節で述べた入力変数間の距離を導入している。

1. カーネルによるWeak Division(図1(a))

与えられた各論理式の共通因子(カーネル)を新たに中間変数で表し、共通因子を中間変数で置き換えることによって論理関数の簡単化を行うものである。この場合の評価関数 fm は、まず多段化を行なったことによって新たに生じる配線の長さ(L_A, L_B, L_C)を入力変数間の距離によって近似的に表し、これを従来のリテラル数だけの評価関数に加えた次式を用いている。

$$\begin{aligned} fm &= (n_c - 1)\cdot\sum_{i=1}^{n_k} lk_i + (n_k - 1)\cdot\sum_{i=1}^{n_c} lc_i - n_c \\ &\quad -K(L_A + L_B + L_C) \\ L_A &= \sum_{i=1}^{n_k} D(k_i),\ L_B = D\left(\sum_{i=1}^{n_k} k_i\right),\ L_C = \sum_{i=1}^{n_c} D(c_i) \end{aligned}$$

n_c コカーネルの数　n_k カーネルの積項数
c_i 各コカーネル　k_j カーネルの各積項
lk_i カーネルの各積項に含まれるリテラル数
lc_i 各コカーネルに含まれるリテラル数

ただし K は定数であり、リテラル数と配線長のトレードオフを決定するものである。

2. 共通ゲートの抽出(図1(b))

複数のAND(OR)ゲートから共通な入力を括り出すという手法である。この場合の評価関数 fm は、1.と同様に入力変数間の距離を使った次式を用いている。

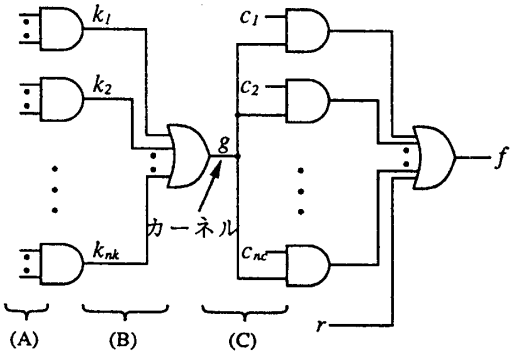
$$\begin{aligned} fm &= (n-1)\cdot l_e - n - K(L_A + L_B) \\ L_A &= D(e),\ L_B = \sum_{i=1}^{n} D(b_i) \end{aligned}$$

n ゲート数　e 共通な入力
l_e 共通な入力の数　b_i 各ゲートの非共通な入力

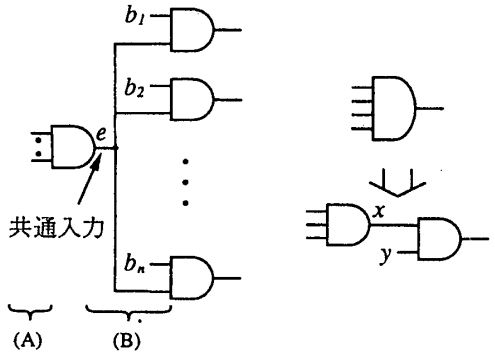
3. ファンイン調整 (図 1(c))
ファンイン制限を越えたゲートの分解を行なうものである。これも以下のような評価関数 fm を用いることによってレイアウトを考慮した分解が出来る。

$$fm = D(x) + D(y)$$

x 括り出す入力 y 残りの入力



(a) カーネルによる Weak Division



(b) ゲートの抽出　　(c) ファンイン調整

図 1: 多段化の各手法

3.3 処理の流れ

多段化の処理はカーネルによる Weak Division、共通ゲートの抽出、ファンイン調整の順に行なわれる。そして各段階ではそれぞれ次のような手順で処理が行われる。

1. 与えられた論理式から全ての因子を取り出し、リテラル数、入力変数間の距離等を計算する。
2. 取り出した因子の中から前節で述べた評価関数が最大となるような因子の組を選択する。
3. 選択された因子によって回路の論理の変換を行い、各素子の再配置を行う。

このような処理を評価関数が 0 以下となるまで回路に適用し、回路の多段化を行う。

これらの処理のうち、従来手法と異なる点として、回路の変形の際に各素子の再配置も行なっている。これは入力変数間の距離を計算する際に中間変数のレイアウト上の座標が必要であるためと、最終的な出力として多段化した回路のネットリストに加えて、各素子のレイアウト上の位置も出力するためである。

4 実験結果

本手法で提案したアルゴリズムの有効性を確かめるために、入力変数間の距離を評価関数に加えた場合と、加えなかった場合について種々の回路に対して多段化を行ない、それぞれの場合の概略配線長を比較した。さらにこの多段化結果を元に、レイアウト合成ツール Solo[5] を使用して各回路のレイアウト合成を行ないその面積を比較した。それらの実験結果が表 1である。表中の各値は入力変数間の距離を評価関数に加えずに回路を合成した場合の値を 100 としたときの入力変数間の距離を評価関数に加えた場合の値である。

この表より、従来のリテラルだけを評価関数として使った場合に比べて入力変数間の距離というパラメータを加えた評価関数を使用した場合の方が概略配線長が小さくなっていることが確認できる。またレイアウト合成結果においてもレイアウトの面積、特に配線の面積の減少が確認でき、本手法の有効性を確認できた。ただし回路によってその効果はかなり開きがあり、表中の squar5 のように全く効果が見られないものもある。なお、本手法が特に有効なのは表中の thresh8(8 入力のスレッシュホールド関数) などのような対称関数の場合であることもわかっている。

表 1: 実験結果

回路名	概略配線長	レイアウト面積	
		全体面積	配線面積
xor5	92.3	97.3	93.5
rd53	96.0	99.1	95.3
rd73	95.0	98.0	95.6
squar5	98.3	101.3	100.0
thres8	82.3	86.8	75.2

5 おわりに

本稿では、レイアウトを考慮した論理多段化の手法として端子位置を考慮した論理多段化の手法を提案した。端子位置を考慮するために入力変数間の距離というパラメータを新しく導入し、多段化における効果を実際のレイアウトにおいて実験によって確かめた。今後の課題としては、組合せ回路の多段化の他の手法への本手法の応用、論理合成の他の問題 (順序回路合成など) への拡張、などが考えられる。

参考文献

[1] M.Pedram and N.Bhat, *"Layout Driven Technology Mapping"*, Proc. of 28-th DAC, pp.99-105, 1991.

[2] P.Abouzeid, K.Sakouti, G.Saucier and F.Poirot, *"Multilevel Synthesis Minimizing the Routing Factor"*, Proc. of 27th DAC, pp.365-368, 1990.

[3] R.K.Brayton, *"Multi-level Logic Synthesis"*, ICCAD-90 Tutorial, 1990.

[4] R.K.Brayton and C.T.McMullen, *"The Decomposition and Factorization of Boolean Expressions"*, Proc. of ICCAS-82, pp.49-54, 1982.

[5] *"SOLO1400 Reference Manual"*, European Silicon Structures, 1990.