論理合成システムにおける最適化状態割当て

4N−10

井上　こずえ*　　中田　恒夫**
*富士通株式会社　　**（株）富士通研究所

1．はじめに

有限状態回路の自動合成において、回路を最小化することが重要である。状態割当てでは、状態遷移記述に現れる状態名を２値符号化し、論理を二段論理式の表現であるカバー形式に変換する。論理合成システムでは、状態割当てで生成した二段論理式に含まれる冗長性を取り除くために論理最適化を行う。そこでこの論理最適化の効率を上げるような状態割当てを行い、素子数の減少を図ることを考える。

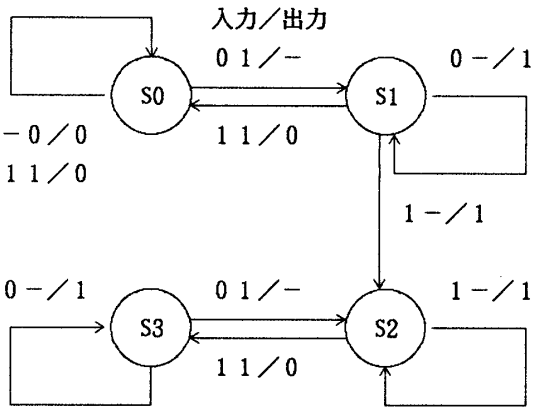

図１．状態遷移図　　−：don't care

現状態	出力			入力	出力
入力　次状態					
-0　S0 → S0	0			-0 00	00 0
11　S0 → S0	0			11 00	00 0
01　S0 → S1	−		状態	01 00	01 −
11　S1 → S0	0	...①	割当て	11 01	00 0
0-　S1 → S1	1		⟶	0- 01	01 1
1-　S1 → S2	1	...②	S0 : 00	1- 01	10 1
1-　S2 → S2	1		S1 : 01	1- 10	10 1
11　S2 → S3	0		S2 : 10	11 10	11 0
01　S3 → S2	−	...③	S3 : 11	01 11	10 −
0-　S3 → S3	1			0- 11	11 1
状態遷移記述				二段論理式（カバー 表現）	

図２．状態遷移記述と状態割当て

論理最適化は、各積項の共通部分の括りだしが処理の本質である。従って積項の共通要素を多くする状態割当て手法を考察する。

2．状態間の関連性を利用した状態割当て法

状態割当て後の論理は、カバー形式で表現された二段論理である。カバー形式の各行は、キューブで表現された積項となる。ある２つの積項の共通要素が多くなるように状態割当てを行うということは、それらに対応するキューブの共通要素が多くなるように状態を割当てることと同じである。２つのキューブ間の共通要素が多ければ、そのキューブ間のハミング距離は近くなる。１つのキューブは、入力・現状態・次状態・出力の４つの要素で構成される。この中で、入力・出力は設計仕様として与えられたものなので変えられないが、現状態・次状態は状態割当てが決める。従って現状態・次状態に着目し、各キューブ間のハミング距離がなるべく小さくなるようにコードの埋めこみを行う。具体的には、同じ遷移元（又は遷移先）を持つ状態に対してハミング距離の小さいコードを割当てていく。以下に例を示す。

図１の状態遷移図を表現したものが、図２の状態遷移記述である。S1に着目し、S0とS2に近いコードを埋めこむとキューブ①②間のハミング距離が小さくなる。同様にS2に着目し、S1とS3に近いコードを埋めこむと②③間のハミング距離が小さくなる。しかし、全ての状態に同時に着目するのは難しい。

そこで、状態遷移図において、「直接の遷移」を持つことを状態間の関連の強さとし、その強い順にハミング距離の近いコードを割当てていくことにする。実際には、状態をノードとする完全グラフを作成し、グラフのエッジに付された値（＝重み）を算出し、その重みを基にコードの埋めこみを行っていく。

重み計算の方法であるが、ＭＵＳＴＡＮＧ［１］を参照し、入力・出力・遷移元・遷移先の４つの要素を使って何通りかの実験を行ってみた。その結果、入力・出力は見ずに、遷移元だけを、或いは遷移先だけを見て重みづけを行う方法が、全体的に良い結果を生むことが多かったのでその方法を採用した。遷移元だけを見る方法をファンイン指向アルゴリズム、遷移先だ

A Method of Optimizing State Assignment,
Kozue Inoue*,Tuneo Nakata**(*:FUJITSU LIMITED,
**:FUJITSU Laboratories LTD.)

けを見る方法をファンアウト指向アルゴリズムとする。

重みづけ計算について、ファンアウト指向アルゴリズムを例に採って説明する。

ある２つの状態間の重みを算出する上で、それら２つの状態から遷移する先の状態の共通部分を数えあげ、それらの積を重みとする。共通の遷移先数が多ければ、次状態の遷移を作る論理を共通にできる可能性が高いと考える。

図１の状態遷移図に対して、ファンアウト指向アルゴリズムを適用してみる。

(1) 各現状態毎に次状態を数え上げる。

Ｓ０ ⇒ (S0:2, S1:1)

Ｓ１ ⇒ (S0:1, S1:1, S2:1)

Ｓ２ ⇒ (S2:1, S3:1)

Ｓ３ ⇒ (S2:1, S3:1)

(2) 各エッジ間の重みを計算する。例としてＳ０，Ｓ１間の重み計算を図３に示す。

(3) 算出した重みに従ってコードの埋め込みを行う。まず、最も大きな重みを持つ状態を選び、次にその状態との重みが最も大きい状態を、次にコードを割当てる状態とする。この様にして状態割当を行う順序を決め、その順に従ってハミング距離＝１となるようにコードを割り当てる。

ファンイン指向アルゴリズムは、ファンアウト指向アルゴリズムとは逆に遷移元に着目し、同様にして重みを算出する。

今回の結果としては、ファンイン指向・ファンアウト指向の両方を行った結果から良い方を採ることにする。重み計算の方法として、ファンイン指向、ファンアウト指向の両方を組み合わせたものも考えられるが、結果が必ずしも良くないので不採用とした。

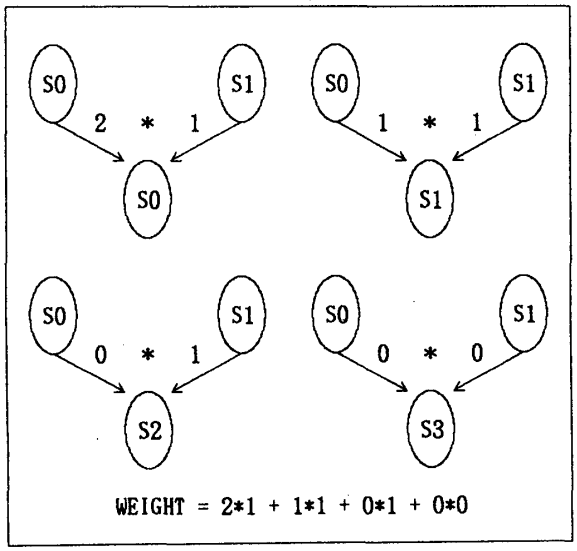

図３．ファンアウト指向アルゴリズムによる重み算出法

３．結果

本手法を'89 MCNC logic Synthesis and Optimization Benchmarks Ver.2.0 のデータに適用した結果を表１に示す。従来方法では後続処理を全く考慮に入れず、シミュレーテッドアニーリング法を用いて最適化を図っていたのだが、それと比べて20～30%のリテラル数の削減が図れた。又、ＭＵＳＴＡＮＧ，ＮＯＶＡ［２］と比較するとリテラル数はほぼ同等であり、処理時間は30～300 倍程度高速である。尚、個々のデータを見ると、本手法の最適化の効き方にばらつきがあり、多種多様な論理回路に対して絶対的なアルゴリズムを見つけることが難しいことを示唆している。

表１．実験結果（リテラル数）

ﾃﾞｰﾀ 名	今回	従来	MUSTANG	NOVA
bbara	７０	９０	６４	６１
bbtas	２７	３４	２５	２１
cse	２０５	２０５	２０６	１９０
dk14	１１２	１１４	１１７	９８
dk15	６３	６３	６９	６５
donfile	１２７	２３１	１６０	８８
ex1	２３７	２６９	２８０	２１５
tav	２３	２４	２５	２５
tbk	２９６	５７９	５４７	２８９
train11	５３	６５	３７	４３
TOTAL	1213	1674	1530	1095

４．おわりに

有限状態回路の自動合成における状態割当てに関して、後続の論理最適化を考慮した手法を試みた。従来手法と比較して20～30%のリテラル数の削減が図れた。今後は、状態遷移記述の特徴を見分け、その特質に合った状態割当てを選択するようなシステムの考察を行っていきたい。

参考文献

［１］Devadas,et.al.,MUSTANG:State Assignment of Finite State Machines Targeting Multilevel Logic Implementations,IEEE Trans.on CAD,Vol.CAD7,No 12,pp.1290-1330(1988).

［２］Villa,et.al.:NOVA:State Assignment of FSM Optimal Two-Level Logic Implementations, Proc. of 26th DAC, pp.327-332(1989).

［３］中田恒夫：状態遷移記述を利用したテスト容易化手法（第40回情報処理全国大会4M-2,1990).