設計自動化 48－2
（1989. 7. 11）

ゲート論理構造比較・編集アルゴリズムとインクリメンタル論理生成への適用

新舎 隆夫[+] 森田 正人[++] 越下 順二[+++] 久保 隆重[+]

[+]㈱日立製作所 システム開発研究所
[++]㈱日立製作所 神奈川工場
[+++]日立ソフトウェアエンジニアリング㈱

本論文はインクリメンタル論理生成に関するものである．インクリメンタル論理生成の目的は，レイアウト設計工程以降において，論理変更が機能レベルで行える手段を提供することにあり，インクリメンタル論理生成の課題は，機能論理変更に対して再利用可能な，論理変更前のゲート論理内のレイアウト情報を最大限保存することにある．本論文では，インクリメンタル論理生成を論理生成とインクリメンタル処理に分離して，インクリメンタル処理をゲート論理構造比較・編集により行うというアプローチを採用し，ゲートマトリクス法を中核とするゲート論理構造比較・編集アルゴリズムを提案する．本アルゴリズムを使用するインクリメンタル論理生成は，９９％以上のレイアウト情報の保存により，超大型計算機Ｍ６８Ｘの論理変更設計効率を大幅に向上した．

A Gate-Level Logic Structure Comparison and Editing Algorithm and Its Application to Incremental Logic Synthesis

Takao SHINSHA[+] Masato MORITA[++] Junji KOSHISHITA[+++] Takashige KUBO[+]

[+]Systems Development Laboratory, Hitachi, Ltd.
1099 Ohzenji Asao-ku Kawasaki-shi,215 Japan
[++]Kanagawa Works, Hitachi, Ltd.
[+++]Hitachi Software Engineering Co., Ltd.

This paper deals with incremental logic synthesis. The objective of incremental logic synthesis is to provide logic designers with a means which enables them to perform logic changes at a function level in and after the layout design stage. The task of incremental logic synthesis is to preserve layout information as much as possible, which is reusable against functional-level logic changes, in the gate-level logic before the logic changes. This paper takes an approach that incremental logic synthesis is divided into logic synthesis and incremental processing, which is accomplished by gate-level logic structure comparison and editing. This paper also presents the gate-level logic structure comparison and editing algorithm with the gate matrix method as its core. The incremental logic synthesis using this algorithm has greatly increased logic change design efficiency of the very large scale computer, M68X, by preserving more than 99% of the layout information.

1．　まえがき

近年，論理装置の大規模・複雑化が伸展するにつれて，論理設計工数低減がますます重要な課題になっている．この課題に対処する有力な方法の一つが機能論理からゲート論理を生成する論理生成であり，論理生成はゲート論理の初期生成だけでなく，機能論理変更時のゲート論理更新にも使用される．しかし，論理生成によるゲート論理更新は，論理情報のみのゲート論理を扱う論理設計工程では可能であるが，レイアウト情報を含むゲート論理を扱うレイアウト設計工程以降では不可能である．その理由は，ゲート論理の論理情報は更新可能でも，ゲート論理内のレイアウト情報がすべて消失するために，レイアウト及びディレイ検証を最初からやり直さなければならないという問題が生じるからである．この問題を解決するには，機能論理変更時に機能論理変更部分に対応するゲート論理部分だけを更新し，再利用可能なレイアウト情報を保存するインクリメンタル論理生成が不可欠である．

本論文はインクリメンタル論理生成に関するものである．インクリメンタル論理生成の目的は，レイアウト設計工程以降において，論理変更が機能論理レベルで行える手段を提供することにあり，その課題は機能論理変更に対して再利用可能なレイアウト情報を最大限保存することにある．*

インクリメンタル論理生成はその重要性は認識されているものの，その試みは見当らない．[1]本論文では，インクリメンタル論理生成を論理生成とインクリメンタル処理に分離し，インクリメンタル処理をゲート論理構造比較・編集により行うというアプローチを採用している．具体的に述べると，最初に，論理変更後の新機能論理から論理生成により中間ゲート論理（レイアウト情報を含まない）を生成する．次に，論理変更前の旧ゲート論理（レイアウト情報を含む）と中間ゲート論理の論理構造を比較し，レイアウトにより論理構造が異なる論理部分の対（論理構造が同一の論理部分の対を含む）ＡとＣ，論理変更により論理構造が異なる論理部分ＢとＤを各々認識する．ここで，ＡとＢは旧ゲート論理に属し，ＣとＤは中間ゲート論理に属する．最後に，ＡとＤを選択して編集し，論理変更後の新ゲート論理（再利用可能なレイアウト情報を含む）を生成する．

ゲート論理構造比較に関してこれまでに，論理接続検証[2]を目的に，グラフの同形判定のためのグラフ分割アルゴリズム[3]を応用したアルゴリズム[4]が提案されている．このアルゴリズムはＬＳＩ単位にＬＳＩの入出力端子の対応関係が与えられているという前提下で，論理構造が同一のゲートグループ対を認識する．これに対して，本論文のアルゴリズムはモジュール（機能論理の構成要素）単位にモジュールの切口入出力信号名がユニークであるという前提下で，ゲート論理をゲート，サブゲート，ピンの３レベルに分け，対応ゲート対，対応サブゲート対，対応ピン対を順次認識する．そして，対応ゲート対と対応サブゲート対の認識に，再利用可能なレイアウト情報の保存率を高めるために，ピンレベルの論理変更を許容したゲートマトリクス法を使用している．ゲートマトリクス法は同一タイプのゲート対（サブゲート対）のゲート論理構造全体に関する類似度に基づいて対応ゲート対（対応サブゲート対）を認識する方法であり，類似度の基準として入力信号，出力信号，入力ゲート，出力ゲート各々の共有率を使用している．両者のアルゴリズムを比較すると，本アルゴリズムは取扱う論理規模は小さいが，論理構造比較の精度は高いといえる．

本論文のインクリメンタル論理生成のアプローチは文献5)で提案済みであり，本論文の目的はゲート論理構造比較・編集アルゴリズムの提案にある．本論文では，第２章でインクリメンタル論理生成の概要を，第３章でゲート論理構造比較・編集の概要を，第４章でゲートマトリクス法を，第５章でゲート論理構造比較・編集アルゴリズムを各々述べ，第６章でこのアルゴリズムの処理例を示し，第７章で超大型計算機Ｍ６８Ｘの設計への適用結果を述べる．

* 本論文は最適なゲート論理を生成する論理生成を前提にしており，人手による論理最適化情報の保存は扱わない．

2．　インクリメンタル論理生成

ゲートアレイ設計のＬＳＩを対象にした本論文のインクリメンタル論理生成の概要を図１に示す．この図において，今，ゲート論理ＧＬ1’＋ΔＧＬ1’が機能論理ＦＬ１＋ΔＦＬ１から論理生成とレイアウトにより生成されているとする．ここで，’はゲート論理がレイアウト情報を含むことを表す．次に，機能論理ΔＦＬ１がΔＦＬ２に変更されたとする．このとき，論理変更後の機能論理に対して論理生成を行うと，生成結果はＧＬ１＋ΔＧＬ２となり，論理情報はΔＧＬ１からΔＧＬ２に更新できても，ＧＬ1’のレイアウト情報はすべて消失してしまう．一方，インクリメンタル論理生成を行うと，生成結果はＧＬ1’＋ΔＧＬ２となり，ΔＧＬ２のみが更新され，ＧＬ1’のレイアウト情報の再利用が可能になる．ここで，

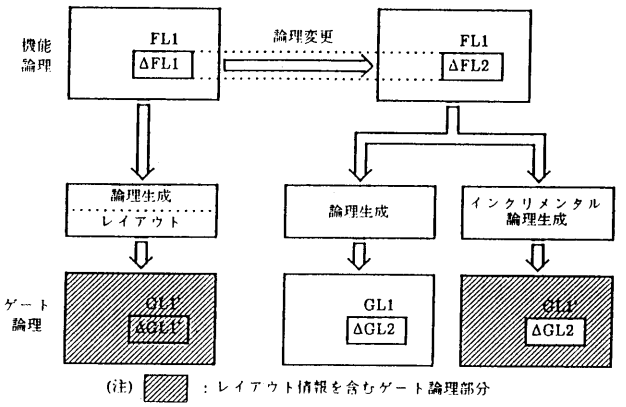


図1　インクリメンタル論理生成
Fig. 1　Incremental Logic Synthesis.

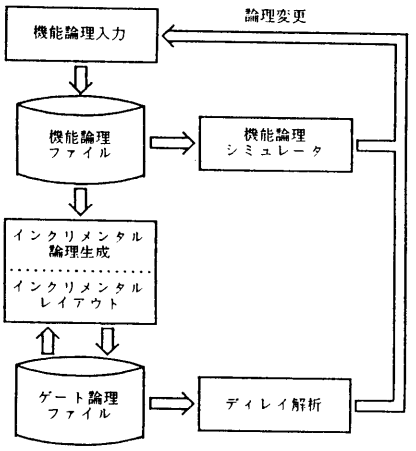


図2　設計検証サイクル
Fig. 2　The Design Verification Cycle.

ΔＧＬ２のレイアウト情報は既存のレイアウト情報を考慮して再レイアウトを行うインクリメンタル・レイアウトにより補充される．

上述のインクリメンタル論理生成とインクリメンタル・レイアウトを組合せると，設計ファイルの一元管理が可能になり，図２に示す設計検証サイクルを構成することが可能になる．すなわち，機能論理ファイルを使用する機能論理シミュレータによる機能検証とゲート論理ファイルを使用するディレイ解析によるディレイ検証が並行して行え，設計検証効率を大幅に向上することが可能になる．

本論文では，インクリメンタル論理生成を論理生成とインクリメンタル処理に分離し，インクリメンタル処理をゲート論理構造比較・編集により行うというアプローチを採用し，インクリメンタル論理生成を実現している．

３．　ゲート論理構造比較・編集

本論文のゲート論理構造比較・編集は以下の４種類のゲート論理を扱う．

（１）原ゲート論理：論理変更前の旧機能論理から論理生成により生成されたゲート論理．

（２）旧ゲート論理：原ゲート論理にレイアウトが行われた，レイアウト情報を含むゲート論理．

（３）中間ゲート論理：論理変更後の新機能論理から論理生成により生成されるゲート論理．

（４）新ゲート論理：新機能論理と論理等価で，論理変更に対して再利用可能なレイアウト情報を含む，求めるゲート論理．

これらのゲート論理に関して，原ゲート論理と旧ゲート論理はレイアウトにより，原ゲート論理と中間ゲート論理は論理変更により各々論理構造が異なっている．ここで，レイアウトによる論理構造の相違の要因にはゲート置換，ピングループ（サブゲート）交換，ピン交換の３種類があり，論理変更による論理構造の相違の要因にはゲートの追加，削除，ゲートタイプ変更と信号線の追加，削除，信号名変更の６種類がある．このとき，新ゲート論理は，論理構造の保存に関して，レイアウトによる論理構造の相違を許容し，論理変更による論理構造の相違を許容しない論理になる．これらのゲート論理間の論理構造の相違の詳細は文献5)を参照されたい．

本論文のゲート論理構造比較・編集の概要を図３に示す．ゲート論理構造比較・編集は旧ゲート論理と中間ゲート論理の論理構造を比較し，レイアウトにより論理構造が異なる論理部分の対（論理構造が同一の論理部分の対を含む）ＡとＣ，論理変更により論理構造が異なる論理部分ＢとＤを各々認識し，ＡとＤを選択して編集し，新ゲート論理を生成する．ここで，ＡとＢは旧ゲート論理に属し，ＣとＤは中間ゲート論理に属する．この方法は，上述の論理構造の保存だけでなく，ＬＳＩ内のゲート位置や配線順序のようなゲート自身及びピン自身に付加されているレイアウト情報の保存も可能である．

４．　ゲートマトリクス法

本論文のゲート論理構造比較アルゴリズムの中核をなすものがゲートマトリクス法であり，ゲートマトリクス法は旧ゲート論理と中間ゲート論理の間の対応ゲート対を認識するための方法である．ゲートマトリクス法の前提条件，基本思想，アルゴリズムを順次述べる．

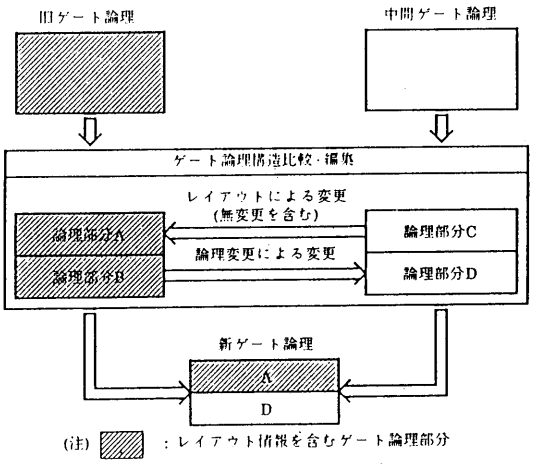


図3　ゲート論理構造比較・編集

Fig.3　Gate-Level Logic Structure Comparison and Editing.

4．1　前提条件

本論文で扱う機能論理はＦＬ (Function Logic Diagram) [6]であり，ＦＬは以下の３種類のモジュールを構成要素として記述される．

（1）汎用モジュール：複数ゲートからなる一つの論理単位で，その機能がブール式または真理値表で個別に定義されるモジュール．

（2）標準モジュール：複数ゲートからなる一つの論理単位で，標準的なマクロ論理として，そのゲート論理がライブラリに登録されているモジュール．

（3）基本モジュール：ゲートと１対１に対応するモジュール．

論理生成はモジュール単位に行われ，汎用モジュールはＰＯＬＡＲＩＳ[7]により，標準モジュールはライブラリ展開により，基本モジュールはコピーにより，各々のゲート論理が生成される．ここで，各モジュールはユニークなモジュールＩＤを有しており，このＩＤは生成される各ゲートのゲートＩＤの一部として引継がれる．論理生成システムの詳細は文献8)を参照されたい．

上述のシステム仕様では，生成されるゲート論理において，モジュールＩＤとモジュールの切口入出力信号名だけがユニークな情報である．それゆえ，ゲートマトリクス法はモジュール単位に，モジュールの切口入出力信号名がユニークであるという前提下で対応ゲート対を認識する．

4．2　基本思想

機能論理変更に対して再利用可能なレイアウト情報の保存率を高めるには，対応ゲート対の認識にピンレベルの論理変更を許容する必要がある．そのため，本論文では，同一タイプのゲート対のゲート論理構造全体に関する類似度という概念を導入し，類似度の最も大きいゲート対が対応ゲート対であるという基本思想に基づき，ゲートマトリクス法を考案している．

このような類似度の基準はゲート論理内のユニークな情報を使用して定義する必要がある．そこで，モジュールの切口入出力信号名を使用し，同一タイプの旧ゲート論理に属するゲートG_iと中間ゲート論理に属するゲートG_jのゲート対(G_i,G_j)について以下の二つの基準を定義する．

（1）入力信号共有率：R_{cil}

$$R_{cil}(G_i,G_j)=50\left(\frac{N_{cil}(G_i,G_j)}{N_{il}(G_i)}+\frac{N_{cil}(G_i,G_j)}{N_{il}(G_j)}\right)$$

ここで，$N_{il}(G_i)$はG_iの出力を決定する切口入力信号の個数を，$N_{cil}(G_i,G_j)$は$N_{il}(G_i)$算出時の切口入力信号と$N_{il}(G_j)$算出時の切口入力信号の間の共通信号の個数を各々表す．

（2）出力信号共有率：R_{col}

$$R_{col}(G_i,G_j)=50\left(\frac{N_{col}(G_i,G_j)}{N_{ol}(G_i)}+\frac{N_{col}(G_i,G_j)}{N_{ol}(G_j)}\right)$$

ここで，$N_{ol}(G_i)$はG_iの出力信号により出力が決定される切口出力信号の個数を，$N_{col}(G_i,G_j)$は$N_{ol}(G_i)$算出時の切口出力信号と$N_{ol}(G_j)$算出時の切口出力信号の間の共通信号の個数を各々表す．

上述の基準のいずれかを使用してゲートの対応づけを行うと，一般にいくつかの対応ゲート対が認識可能である．そこで，この対応ゲート対の情報を使用し，以下の二つの基準を定義する．

（3）入力ゲート共有率：R_{cis}

$$R_{cis}(G_i,G_j)=50\left(\frac{N_{cis}(G_i,G_j)}{N_{is}(G_i)}+\frac{N_{cis}(G_i,G_j)}{N_{is}(G_j)}\right)$$

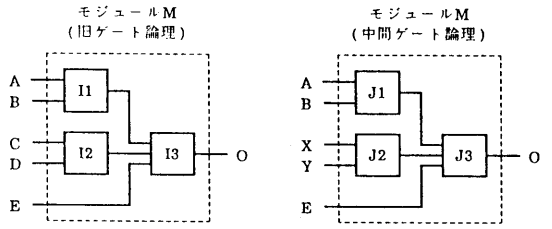
ここで，$N_{is}(G_i)$はG_iに直接接続されている，入力ゲートと切口入力信号の個数を，$N_{cis}(G_i,G_j)$は$N_{is}(G_i)$算出時の入力ゲートと切口入力信号と$N_{is}(G_j)$算出時の入力ゲートと切口入力信号の間の対応ゲート対と共通信号の個数を各々表す．

（4）出力ゲート共有率：R_{cos}

$$R_{cos}(G_i,G_j)=50\left(\frac{N_{cos}(G_i,G_j)}{N_{os}(G_i)}+\frac{N_{cos}(G_i,G_j)}{N_{os}(G_j)}\right)$$

ここで，$N_{os}(G_i)$はG_iに直接接続されている，出力ゲートと切口出力信号の個数を，$N_{cos}(G_i,G_j)$は$N_{os}(G_i)$算出時の出力ゲートと切口出力信号と$N_{os}(G_j)$算出時の出力ゲートと切口出力信号の間の対応ゲート対と共通信号の個数を各々表す．

類似度の基準の算出例を図４に示す．この図において，$R_{cil}(I3,J3)$は，$N_{il}(I3)=5(A,B,C,D,E)$，$N_{il}(J3)=5(A,B,X,Y,E)$，$N_{cil}(I3,J3)=3(A,B,E)$であるから60となる．一方，$R_{cis}(I3,J3)$は，$N_{is}(I3)=3(I1,I2,E)$，$N_{is}(J3)=3(J1,J2,E)$で，初期状態では$N_{cis}(I3,J3)=1(E)$であるから33となる．



$$R_{cil}(I3,J3) = 50(\frac{3}{5}+\frac{3}{5}) = 60$$

$$R_{cig}(I3,J3) = 50(\frac{1}{3}+\frac{1}{3}) = 33$$

ゲート対(I1,J1)が対応ゲート対と認識されたとき、

$$R_{cig}(I3,J3) = 50(\frac{2}{3}+\frac{2}{3}) = 67$$

図4　類似度の基準の算出例
Fig.4 Similarity Indexes Calculation Examples.

しかし，ゲート対(I1,J1)が対応ゲート対と認識されたときはN_{cig}(I3,J3)に１が加わるため，R_{cig}(I3,J3)は67となる．このように，特定のゲート対に関して，R_{cil}とR_{col}がゲート論理構造全体に依存した固定値となるのに対して，R_{cig}とR_{cog}はゲートの対応づけの進捗状況に依存した可変値となり，その値は一般に増大する．

ゲートマトリクス法は上述の４種類の類似度の基準を所定の手順で使用し，対応ゲート対を順次認識する．

４．３　アルゴリズム

ゲートマトリクス法はR_{cil}, R_{col}, R_{cig}, R_{cog}のいずれかを使用する通常のゲート対応づけ操作と強制的なゲート対応づけ操作の５操作からなる．ここで，通常のゲート対応づけ操作は各々ゲートマトリクス操作を共通操作として使用する．ゲートマトリクス操作，ゲート対応づけ操作，ゲート対応づけ操作の制御を順次述べる．

（１）ゲートマトリクス操作

指定された類似度の基準Rのゲートマトリクス操作手順を以下に述べる．

Step１：与えられたゲートセット対（旧ゲート論理に属するゲートセットG_i'sと中間ゲート論理に属するゲートセットG_j'sの対）の同一タイプの各ゲート対(G_i,G_j)についてR(G_i,G_j)を算出し，ゲートマトリクスを作成する．ここで，ゲートマトリクスは，行方向の項目がG_iで，列方向の項目がG_jで，マトリクス要素がR(G_i,G_j)である．

Step２：有効なマトリクス要素をトリプル(R(G_i,G_j),G_i,G_j)の形式に変換する．

Step３：トリプルセットを，第１キーはR(G_i,G_j)で降順に，第２キーはG_iで昇順に，第３キーはG_jで昇順に各々ソートする．以下，ソート後のランクがｃのトリプルを([R(G_i,G_j)]$_c$,[G_i]$_c$,[G_j]$_c$)で表す．

Step４：ランクが１のトリプルから順に以下のトリプル操作を行う．

（ａ）以下の論理式を満たすならば，(G_i,G_j)を対応ゲート対として認識する．

([R(G_i,G_j)]$_c$=[R(G_i,G_j)]$_{c+1}$) AND

(([G_i]$_c$=[G_i]$_{c+1}$) OR ([G_j]$_c$=[G_j]$_{c+1}$))=FALSE

（ｂ）そうでなければ，これは一つ以上のG_i'sと一つ以上のG_j'sが見掛け上対応している状態（１対１の対応は除く）を表す．このとき，([R(G_i,G_j)]$_c$,G_i,[G_j]$_c$)を満たすG_i'sと([R(G_i,G_j)]$_c$,[G_i]$_c$,G_j)を満たすG_j'sを衝突ゲートグループとして認識し，このグループをそのグループレベルに従って衝突ゲートテーブルに登録する．ここで，グループレベルは衝突ゲートG_i'sの各G_iのレベル（切口出力信号を起点にしたゲート段数）の最小値である．

（ｃ）上述の操作により対応ゲートまたは衝突ゲートと認識されたG_i,G_jのいずれかを含む，ランクがｃより大きいトリプルを無視する．

ゲートマトリクス操作例を図５に示す．

（２）ゲート対応づけ操作

ゲート対応づけ操作は，R_{cil}, R_{col}, R_{cig}, R_{cog}のいずれかを使用する通常操作（以下，類似度の基準Rを使用するゲート対応づけ操作をR操作と略す）と強制操作の５操作がある．

R_{cil}操作手順を以下に述べる．

Step１：衝突ゲートテーブルの初期化を行う．

Step２：当該モジュールのG_i'sとG_j'sの対をセットし，R_{cil}を指定してゲートマトリクス操作を行う．

R_{col}操作手順は，上述のStep２において”R_{cil}”を”R_{col}”に置換したものである．

R_{cig}操作手順を以下に述べる．

Step１：グループレベルの降順に衝突ゲートテーブルから衝突ゲートグループを一つ取出す．衝突ゲートグループがなければ，本操作は終了する．

Step２：当該衝突ゲートグループにおいて，対応ゲートと認識されたゲートがあれば，そのゲートを除外する．その結果，一つのG_iと一つのG_jが残っているならば，これらを対応づけ，Step１へ分岐する．また，G_i's，G_j'sのいずれか一方だけが残っているならば，これらを無視し，Step１へ分岐する．そうでなければ、次のStepへ進む．

Step３：当該衝突ゲートグループを構成しているゲートをG_i'sとG_j'sの対に分けてセットし，R_{cig}を指定してゲートマトリクス操作を行う．ここで，このゲートマトリクス操作により新たに衝突ゲートテーブルに登録された衝突ゲートグループは今回のR_{cig}操作の対象外とする．

G_j \ G_i	I1	I2	I3
J1	100	80	50
J2	60	90	90
J3	60	90	90

(a) ゲートマトリクス

ランク	R	G_i	G_j
1	100	I1	J1
2	90	I2	J2
3	90	I2	J3
4	90	I3	J2
5	90	I3	J3
6	80	I2	J1
7	60	I1	J2
8	60	I1	J3
9	50	I3	J1

(b) トリプルソート結果

ランク	認識結果
1	対応ゲート対(I1, J1)
2	衝突ゲートグループ (I2, I3, J2, J3)
3	
4	
5	
6	無視
7	
8	
9	

(c) 認識結果

図5　ゲートマトリクス操作例
Fig.5 A Gate Matrix Operation Example.
(a) A Gate Matrix.
(b) The Triple Sort Result.
(c) The Identification Results.

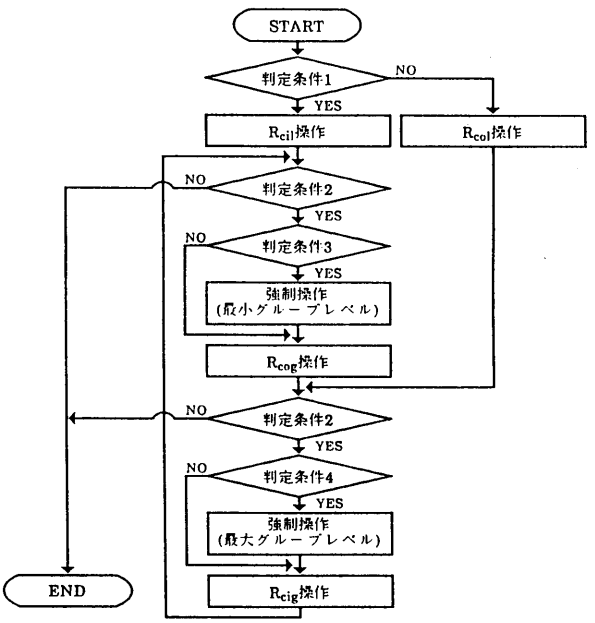
Step４：Step１へ分岐する．

R_{cos}操作手順は，上述のStep１において”グループレベルの降順に”を”グループレベルの昇順に”に，上述のStep３において”R_{cis}”を”R_{cos}”に各々置換したものである．

強制操作は，後述のゲート対応づけ操作の制御で指定されたグループレベル（最大，最小のいずれか）の衝突ゲートグループにおいて，G_iが最小なものとG_jが最小なものを無条件に対応づける．ここで，当該衝突ゲートグループの衝突ゲートテーブルからの取出しは行わない．

（3）ゲート対応づけ操作の制御

ゲート対応づけ操作の制御手順を図６に示す．この制御手順は，中間ゲート論理の切口入力信号数と切口出力信号数の大小関係に依存して２種類が存在する．すなわち，切口入力信号数が切口出力信号数以上の場合は，R_{cil}操作で始め，衝突ゲートグループがなくなるまで，R_{cos}操作，R_{cis}操作の順で各操作を繰返す．一方，切口入力信号数が切口出力信号数未満の場合は，R_{col}操作で始め，衝突ゲートグループがなくなるまで，R_{cis}操作，R_{cos}操作の順で各操作を繰返す．ここで，R_{cos}操作，R_{cis}操作の順で各操作を１回づつ続けて行っても対応ゲート対を全く認識できない場合は，最小グループレベルを指定して強制操作を行う．また，R_{cis}操作，R_{cos}操作の順で各操作を１回づつ続けて行っても対応ゲート対を全く認識できない場合は，最大グループレベルを指定して強制操作を行う．



判定条件１：中間ゲート論理において、切口入力信号数が切口出力信号数以上である．
判定条件２：衝突ゲートグループが存在する．
判定条件３：R_{cog}操作、R_{cig}操作の順で各操作を1回づつ続けて行っても対応ゲート対を全く認識できない．
判定条件４：R_{cig}操作、R_{cog}操作の順で各操作を1回づつ続けて行っても対応ゲート対を全く認識できない．

図6　ゲート対応づけ操作の制御手順
Fig.6 The Control Procedure of Gate Matching Operations.

5. ゲート論理構造比較・編集アルゴリズム

本論文のゲート論理構造比較・編集アルゴリズムを以下に述べる．

（1）前処理

指定されたモジュールの旧ゲート論理のネットリストと中間ゲート論理のネットリストの対を入力し，ネットテーブルの対を作成する．以降の処理はこのネットテーブルの対を使用して行う．

（2）対応ゲート対の認識

当該モジュールの旧ゲート論理のゲートセットG_i'sと中間ゲート論理のゲートセットG_j'sの対に対して上述のゲートマトリクス法を使用し，対応ゲート対を認識する．

（3）対応サブゲート対の認識

ゲート区分が複合ゲートの各対応ゲート対(G_i, G_j)について，G_iのサブゲートセットSG_i'sとG_jのサブゲートセットSG_j'sの対に対してゲートマトリクス法の部分機能を使用し，対応サブゲート対を認識する．ここで，ゲートマトリクス法の部分機能の使用は，ゲートの代わりにサブゲートを使用し，最初にR_{c1s}操作を行い，衝突サブゲートグループが存在する場合は次にR_{cos}操作を行い，それでも衝突サブゲートグループが存在する場合は最後に強制操作とR_{c1s}操作を行うことを意味する．

（4）対応ピン対の認識

各対応ゲート対(G_i,G_j)，ゲート区分が複合ゲートの対応ゲート対の場合は各対応サブゲート対(SG_i,SG_j)の各ピン対(P_i,P_j)について，ピン交換性を考慮しながら，切口入出力信号名，対応ゲート対情報，対応サブゲート対情報を使用してピンの接続対応性を調べ，対応ピン対を認識する．

（5）後処理

ネットテーブル上のゲート，サブゲート，ピン各々の対応情報に基づき，旧ゲート論理の対応論理部分と中間ゲート論理の非対応論理部分を選択し，それらをネットリストに編集して出力する．ここで，ネットリストの編集は非対応論理部分内の各ゲートのゲートＩＤ生成と非対応論理部分内の各信号線の信号名生成を伴う．

6. 処理例

図７に示すゲート論理対を対象に，本論文のゲート論理構造比較・編集アルゴリズムの処理例（ピンレベルの認識処理の詳細は省略する）を以下に示す．

（1）図７（b）に示す中間ゲート論理において，切口入力信号数（６本）は切口出力信号数（３本）より大きいので，R_{c1i}操作を選択する．

（2）図７に示すゲート論理対のゲートセット対についてR_{c1i}操作を行う．R_{c1i}操作によるゲートレベルの認識結果を図８に示す．この操作により，(I1,J1)と(I2,J2)が対応ゲート対として，(I4,I5,J3,J4)と(I6,I7,J5,J6,J7)が衝突ゲートグループとして各々認識される．

（3）グループレベルの昇順に，各衝突ゲートグループを構成している旧ゲート論理と中間ゲート論理のゲートセット対についてR_{cos}操作を行う．R_{cos}操作によるゲートレベルの認識結果を図９に示す．この操作により，(I6,J5)，(I7,J6)，(I5,J4)，(I4,J3)が対応ゲート対として各々認識される．その結果，衝突ゲートグループが存在しなくなるので，ゲートレベルの認識処理は終了する．

（4）対応ゲート対(I4,J3)と(I5,J4)はゲート区分が複合ゲートであるので，これらの対応ゲート対の各々のサブゲートセット対についてR_{c1s}操作を行う．ここで，各ゲートは入力側のサブゲートが交換可能である．R_{c1s}操作によるサブゲートレベルの認識結果を図１０に示す．この操作により，(I42,J31)，(I41,J32)，(I51,J41)，(I52,J42)が対応サブゲート対として各々認識される．その結果，衝突サブゲートグループは存在しないので，サブゲートレベルの認識処理は終了する．

（5）上述の各対応ゲート対，ゲート区分が複合ゲートの対応ゲート対の場合は各対応サブゲート対の各ピン対について対応ピン対を認識する．

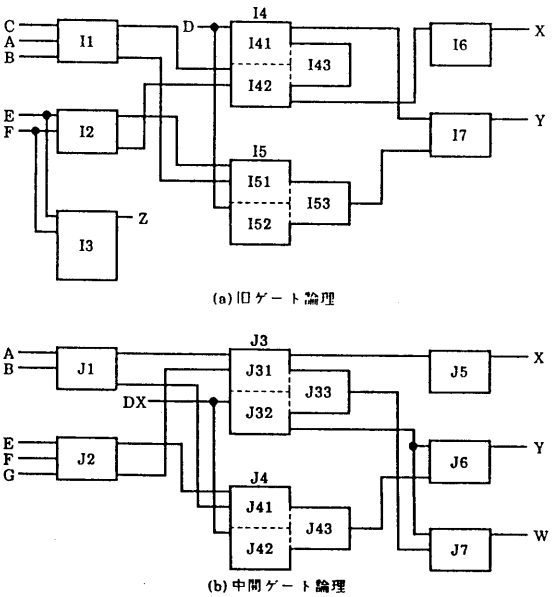


図7 論理構造を比較・編集するゲート論理対

Fig.7 A Gate-Level Logic Pair for Logic Structure to be Compaired and Edited.
(a) Current Gate-Level Logic.
(b) Intermediate Gate-Level Logic.

（6）上述の認識結果に基づき，旧ゲート論理の対応論理部分と中間ゲート論理の非対応論理部分を選択して編集し，図１１に示す新ゲート論理を生成する．この図において，追加ゲートJ7のゲートＩＤ生成（J7の変更）は本処理例の理解を容易にするために省略している．

G_j \ G_i	I1	I2	I3	I4	I5	I6	I7
J1	83	0				67	67
J2	0	83				50	50
J3				67	67		
J4				67	67		
J5	50	67				67	67
J6	50	67				67	67
J7	50	67				67	67

⟹ ・対応ゲート対：(I1, J1), (I2, J2)
・衝突ゲートグループ：
(I4, I5, J3, J4)…グループランク2
(I6, I7, J5, J6, J7)…グループランク1

図8　R_{cil}操作によるゲートレベルの認識結果
Fig.8 The Gate-Level Identification Results by a R_{cil} Operation.

G_j \ G_i	I6	I7
J5	100	0
J6	0	100
J7	0	0

⟹ ・対応ゲート対：(I6, J5), (I7, J6)

G_j \ G_i	I4	I5
J3	42	67
J4	75	100

⟹ ・対応ゲート対：(I5, J4), (I4, J3)

図9　R_{cog}操作によるゲートレベルの認識結果
Fig.9 The Gate-Level Identification Results by R_{cog} Operations.

SG_j \ SG_i	I41	I42
J31	0	100
J32	0	0

⟹ ・対応サブゲート対：(I42, J31), (I41, J32)

SG_j \ SG_i	I51	I52
J41	100	0
J42	0	0

⟹ ・対応サブゲート対：(I51, J41), (I52, J42)

図10　R_{cig}操作によるサブゲートレベルの認識結果
Fig.10 The Subgate-Level Identification Results by R_{cig} Operations.

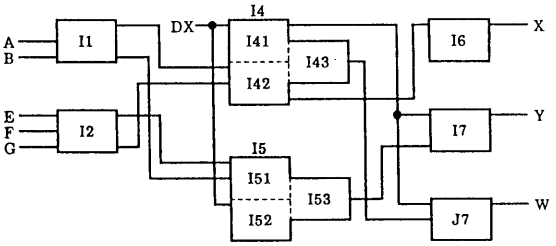


図11　新ゲート論理
Fig.11 The Updated Gate-Level Logic.

7．　適用結果

上述のゲート論理構造比較・編集アルゴリズムを使用するインクリメンタル論理生成を開発し，超大型計算機Ｍ６８Ｘの設計に適用した．本論文では，種々の機能論理変更が行われた２５種のＥＣＬ ＬＳＩへの適用結果を述べる．

（1）レイアウト情報の保存率

レイアウト情報の保存率ρを次式で定義する．

$$\rho = \frac{N_2}{N_1 - N_3}$$

ここで，N_1は新ゲート論理のゲート総数を，N_2はレイアウト情報が保存可能なゲート数を，N_3は追加ゲート数を各々表す．また，レイアウト情報はゲートレベルだけでなく，追加ピンを除くピンレベルのものも含んでいる．

本インクリメンタル論理生成の適用結果を表１に示す．本インクリメンタル論理生成はピンレベルの全面論理変更がある３ゲートを除くすべてのゲートのレイアウト情報を保存しており，ρ＝0.998である．このように，非常に高い比率でレイアウト情報の保存を実現している理由は，ピンレベルの論理変更を許容したゲートマトリクス法を使用しているからである．

表1　適用結果
Table1 Application Results.

新ゲート論理のゲート総数 (N_1)	1288
レイアウト情報が保存可能なゲート数 (N_2)	1229
ピンレベルの論理変更がないゲート数	1039
ピンレベルの一部論理変更があるゲート数	190
レイアウト情報が保存不可能なゲート数	3
ピンレベルの全面論理変更があるゲート数	3
追加ゲート数 (N_3)	56

(注) ゲート数は25LSIの平均値であり、3入力ゲート換算値である．

（2）処理時間

本インクリメンタル論理生成の処理時間は５５秒(on M200H)/2KG LSIであり，その内訳は論理生成の処理時間が５４秒で，ゲート論理構造比較・編集の処理時間が１秒である．

８．　むすび

ゲート論理構造比較・編集アルゴリズムを使用するインクリメンタル論理生成を開発し，超大型計算機Ｍ６８Ｘの設計に適用した．

本インクリメンタル論理生成はピンレベルの論理変更を許容したゲートマトリクス法を使用しているため，９９％以上のレイアウト情報の保存が可能である．また，本インクリメンタル論理生成の処理時間は５５秒(on M200H)/2KG LSIであり，高速に処理可能である．その結果，本インクリメンタル論理生成により，レイアウト設計工程以降でも論理変更を機能論理レベルで行うことを可能にし，論理変更設計効率を大幅に向上した．

本インクリメンタル論理生成は人手による論理最適化情報の保存を扱わない．インクリメンタル論理生成の適用性を拡大するために，論理最適化情報の保存を扱うインクリメンタル論理生成が今後の課題である．

謝　辞　本研究の機会を与えて頂いたシステム開発研究所の川崎淳所長と石原孝一郎部長並びに神奈川工場の大野泰廣部長，また，本研究内容について有益な御意見・御討論を頂いた神奈川工場の土屋洋次主任技師と坂田谷義憲技師に深謝いたします．（所属は当時のもの）

参考文献

1) Trevillyan,L. : An Overview of Logic Synthesis Systems, Proc. of 24th DA Conf., pp.166-172(1987).
2) 築添,小澤 : レイアウト設計検証CADと手法, 情報処理, Vol.25, No.10, pp.1106-1111 (1984).
3) Kubo,N., Shirakawa,I. and Ozaki,H : A Fast Algorithm for Testing Graph Isomorphism, Proc. of Inter. Symp. on CAS, pp.641-644 (1979).
4) Takashima,M., Mitsuhashi,T., Chiba,T. and Yoshida,K. : Programs for Verifying Circuit Connectivity of MOS/LSI Mask Artwork, Proc. of 19th DA Conf., pp.544-550 (1982).
5) Shinsha,T., Kubo,T., Sakataya,Y., Koshishita,J. and Ishihara,K. : Incremental Logic Synthesis through Gate Logic Structure Identification, Proc. of 23rd DA Conf., pp. 391-397(1986).
6) Ohno,Y., Miyoshi,M., Kazama,Y., Tada,O. and Sakai,T. : Design Verification of Large Scale LSI Computers, Proc. of Inter. Symp. on CAS, pp.443-446(1982).
7) Shinsha,T., Kubo,T., Hikosaka,M., Akiyama,K. and Ishihara,K. : POLARIS : Polarity Propagation Algorithm for Combinational Logic Synthesis, Proc. of 21th DA Conf., pp. 322-328(1984).
8) Tsuchiya,Y., Morita,M., Ikariya,Y., Tsurumi, E., Mori,T. and Yanagita,T. : Establishment of Higher Level Logic Design for Very Large Scale Computer, Proc. of 23rd DA Conf.,pp. 366-371(1986).