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††

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，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 M 68 X の論理変更設計効率を大幅に向上した．

1. はじめに

ゲートアレイ設計の LSI で開発される論理装置を対象に機能レベルの論理設計とレイアウト設計を支援するDA (CAD) システムは，論理入力された機能論理を機能論理データベースに格納し，機能論理から論理生成したゲート論理をレイアウトした結果（レイアウト情報を含むゲート論理）をゲート論理データベースに格納する．その後，機能論理データベース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データベースを使用するディレイ解析によるディレイ検証が並行に行われる．このような並行設計検証を可能にするには論理生成に起因する一つの問題が解決されなければならない．

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の更新にも使用される．しかし，ゲート論理がレイアウト情報を含む場合に論理生成によるゲート論理更新を行うと，このレイアウト情報がすべて消失するためにレイアウトおよびディレイ検証を最初からやり直さなければならなくなる．そのため，上記の並行設計検証を可能にするには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という問題が解決されなければならない*．

この問題の解決方法は以下の二つがある．一つは機能論理データベースとゲート論理データベースの各々において論理変更時に元の設計情報に変更情報を順次追加して設計データベースを世代管理する方法[1),2)]であり，もう一つ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するインクリメンタル論理生成を使用する方法である．しかし，インクリメンタル論理生成はこれまでに，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[3)]．

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，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．本論文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．

論理構造比較に関してこれまでに，マスクパターンデータから抽出されたトランジスタ論理の接続検証を目的に，この論理の構造を平面グラフに変換し，グラフの同形判定のためのグラフ分割アルゴリズム[4)]を応用したアルゴリズム[5)]が提案されている．これに対して本論文は，インクリメンタル論理生成を目的に，

† A Gate-Level Logic Structure Comparison and Editing Algorithm and Its Application to Incremental Logic Synthesis by TAKAO SHINSHA (Systems Development Laboratory, Hitachi, Ltd.), MASATO MORITA (Kanagawa Works, Hitachi, Ltd.), JUNJI KOSHISHITA (Hitachi Software Engineering Co., Ltd.) and TAKASHIGE KUBO (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所
††† (株)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(株)

* 本論文は人手によるレイアウト最適化情報を含むレイアウト情報の保存を扱う．しかし，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いるため，人手による論理最適化情報の保存は扱わない．

ゲート論理構造をそのまま扱い，ゲート論理構造に関する類似度に基づいて論理構造を比較するゲートマトリクス法を中核とするアルゴリズムを使用している．したがって，両者は目的，論理構造の扱い方，アルゴリズムがいずれも異なっている．

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6)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，最初に第2章でインクリメンタル論理生成の概要と必要性を述べる．次にゲート論理構造比較・編集に関して，第3章で問題を定義し，第4章と第5章でアルゴリズムを述べ，第6章で本アルゴリズムの処理例を示す．最後に第7章と第8章で本アルゴリズムのインクリメンタル論理生成への適用方法と超大型計算機 M 68 X の設計への適用結果を述べる．

2. インクリメンタル論理生成

2.1 概　　要

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 GL 1′+⊿GL 1′ が機能論理 FL 1+⊿FL 1 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 ⊿FL 1 が ⊿FL 2 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 GL 1+⊿GL 2 となり，論理情報は ⊿GL 1 から ⊿GL 2 に更新できても，GL 1′ 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 GL 1′+⊿GL 2 となり，⊿GL 2 のみが更新され，GL 1′ のレイアウト情報の再利用が可能になる．ここで，⊿GL 2 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．

2.2 必　要　性

機能論理変更に対して再利用可能なレイアウト情報を保存する方法は設計データベースの世代管理とインクリメンタル論理生成の二つがある．前者の方法を使用する DA システムは，機能不良による論理変更は機能レベルで，ディレイ不良による論理変更はゲートレベルで分離して行われるので，論理変更方法に関して分離型になる．これに対して，後者の方法を使用する DA システムはインクリメンタル・レイアウトの併用により機能不良とディレイ不良による論理変更が共に機能レベルで行えるので統合型になる．換言すると，設計データベース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．両者の DA システムを論理変更設計効率の観点から比較すると，統合型の DA システムは，論理変更がすべて機能レベルで（設計者による設計論理上で）行えるので，論理装置の論理規模が大きくなればなるほど優れてくる．特に論理規模が100万ゲートを越え，数千件の論理不良が摘出される超大型計算機のような大規模論理装置を開発する場合は統合型の DA システムが不可欠である．

3. ゲート論理構造比較・編集問題

本論文のゲート論理構造比較・編集問題の概要を図

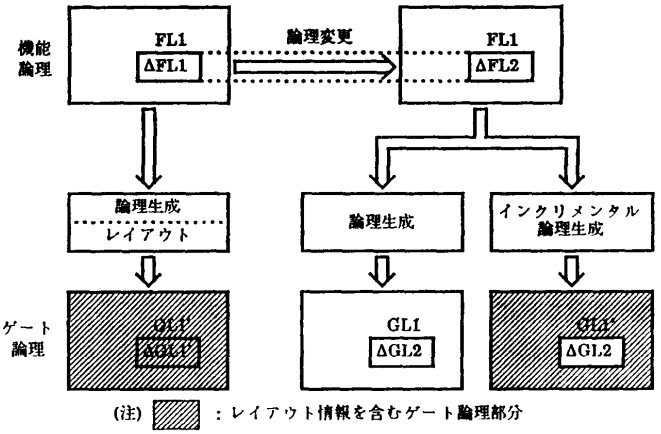


図1 インクリメンタル論理生成
Fig. 1 Incremental logic synthesis.

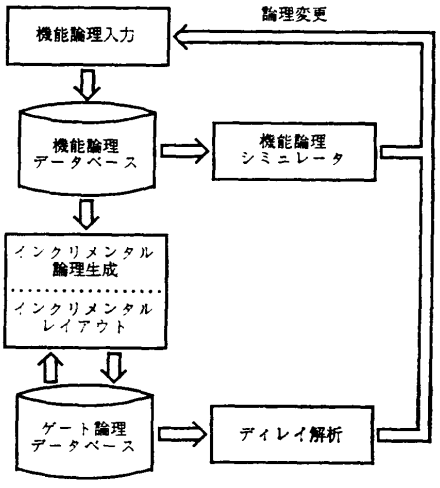


図2 設計検証サイクル
Fig. 2 The design verification cycle.

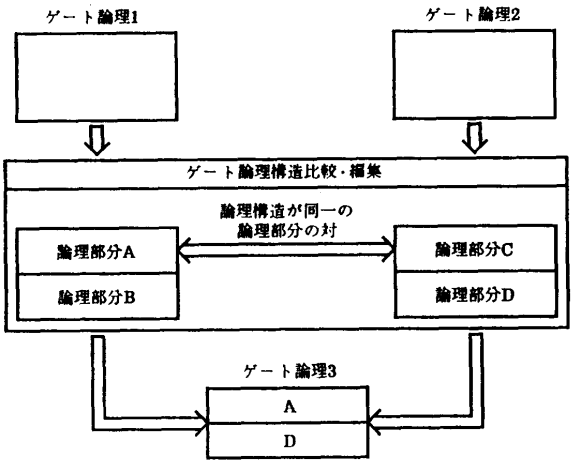


図 3 ゲート論理構造比較・編集問題
Fig. 3 The gate-level logic structure comparison and editing problem.

3に示す．本問題はゲート論理1とゲート論理2を入力し，両者のゲート論理構造を比較して論理構造が同一の論理部分の対AとCおよび論理構造が異なる論理部分BとDを認識し，AとDを編集してゲート論理3を生成するという問題である．ここで，AとBはゲート論理1に，CとDはゲート論理2に各々属する．

本問題が扱うゲート論理1～3はいずれも外部入出力端子とゲート（論理素子）およびゲートとゲートの接続関係を記述した以下のネットリストである．

（1） 外部入出力端子

外部入出力端子はその端子に付与されている端子IDにより定義される．

（2） ゲート

ゲートはそのゲートタイプと入出力ピン番号により定義される．ここで，ゲートタイプはゲートの論理機能，入出力数，入出力極性，パワーやスピードのような物理的特性の組合せにより決められている．

（3） 接続関係

上記の接続関係は外部入出力端子と入出力ピンに接続される信号線に付与される信号名により定義される．

一方，本問題が扱うゲートは以下の三つの交換が可能であり，これらの交換によりゲート論理構造の相違が生じるが，本問題はこの相違を許容して論理構造が同一として扱う．

（1） ゲート交換

ゲートタイプが同一で，入出力ピン番号が異なるゲートが存在する．これらのゲートは交換可能である．

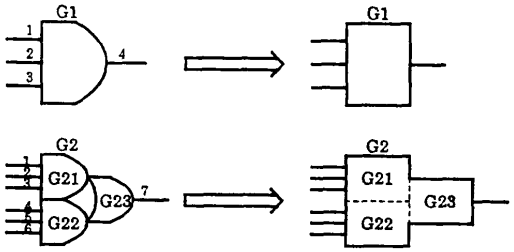


図 4 ゲートの簡略表現
Fig. 4 The simplified gate representation.

（2） サブゲート交換（ピングループ交換）

AND-OR (OR-AND) のような二段論理機能を一ゲートで実現した複合ゲートは複数の入力側の AND (OR) サブゲートと一つの出力側の OR (AND) サブゲートからなるとみなすことができる．このような複合ゲートは入力数が同一の入力側の AND (OR) サブゲートが交換可能である．例えば**図4**において，複合ゲート G2 はサブゲート G21 と G22 が交換可能である．（ピングループ1～3と4～6が交換可能である．）

（3） ピン交換

AND ゲートや OR ゲートのような単一ゲートと上記のような入力側のサブゲートは入力ピンが交換可能である．例えば図4において，単一ゲート G1 は入力ピン1～3が，サブゲート G21 は入力ピン1～3が，サブゲート G22 は入力ピン4～6が各々交換可能である．

本問題の前提条件は以下のとおりである．

（1） 上記のネットリストにおいて，外部入出力端子の端子 ID だけがユニークな情報である．

（2） 外部入出力端子とゲート（ゲート本体と入出力ピン）に何か他の情報が付加されている場合はその付加情報も含めて編集する．

以下，本論文は図4に示すゲートの簡略表現を使用してゲート論理例を示す．

4. ゲートマトリクス法

本論文のゲート論理構造比較・編集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ゲート論理1とゲート論理2をゲートレベルで比較して対応ゲート対を認識する．ゲートマトリクス法の基本思想とアルゴリズムを以下に述べる．

4.1 基本思想

二つのゲート論理間の対応ゲート対の認識率を高め

るには対応ゲート対の認識にピンレベルの相違を許容する必要がある．そのため，本論文は同一ゲートタイプのゲート対のゲート論理構造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いて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，外部入出力端子の端子 ID を使用し，ゲート論理1に属するゲートセット G_i's とゲート論理2に属するゲートセット G_j's の同一ゲートタイプの各ゲート対 (G_i, G_j) について以下の二つの基準を定義する．

（1） 外部入力端子共有率：R_{cit}

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

ここで，$N_{it}(G_i)$ は G_i の出力を決定する外部入力端子の個数を，$N_{cit}(G_i, G_j)$ は $N_{it}(G_i)$ と $N_{it}(G_j)$ を算出時の両者の外部入力端子の間の共通端子の個数を各々表す．

（2） 外部出力端子共有率：R_{cot}

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

ここで，$N_{ot}(G_i)$ は G_i の出力により出力が決定される外部出力端子の個数を，$N_{cot}(G_i, G_j)$ は $N_{ot}(G_i)$ と $N_{ot}(G_j)$ を算出時の両者の外部出力端子の間の共通端子の個数を各々表す．

上記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，これら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_{cig}

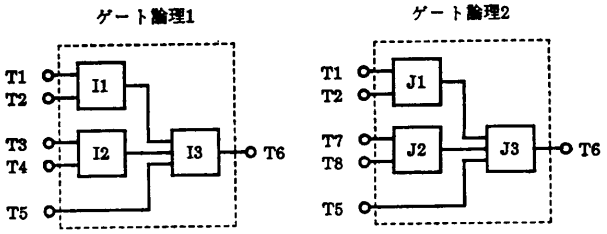
$$R_{cig}(G_i, G_j)=50\left(\frac{N_{cig}(G_i, G_j)}{N_{ig}(G_i)}+\frac{N_{cig}(G_i, G_j)}{N_{ig}(G_j)}\right)$$

ここで，$N_{ig}(G_i)$ は G_i に直接接続されている，入力ゲートと外部入力端子の個数を，$N_{cig}(G_i, G_j)$ は $N_{ig}(G_i)$ と $N_{ig}(G_j)$ を算出時の両者の入力ゲートの間の対応ゲート対の個数と両者の外部入力端子の間の共通端子の個数の総和を各々表す．

（4） 出力ゲート共有率：R_{cog}

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

ここで，$N_{og}(G_i)$ は G_i に直接接続されている，出力ゲートと外部出力端子の個数を，$N_{cog}(G_i, G_j)$ は $N_{og}(G_i)$ と $N_{og}(G_j)$ を算出時の両者の出力ゲートの間の対応ゲート対の個数と両者の外部出力端子の間の共通端子の個数の総和を各々表す．



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

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

ゲート対(I1, J1)が対応ゲート対と認識されたとき，

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

図 5 類似度の基準の算出例
Fig. 5 Similarity indexes calculation examples.

類似度の基準の算出例を図5に示す．この図において，T1～T8 は外部入出力端子を，I1～I3 と J1～J3 は同一ゲートタイプのゲートを各々表す．R_{cit}(I3, J3) は N_{it}(I3)=5(T1, T2, T3, T4, T5)，N_{it}(J3)=5(T1, T2, T7, T8, T5)，N_{cit}(I3, J3)=3(T1, T2, T5) であるから 60 になる．一方，R_{cig}(I3, J3) は N_{ig}(I3)=3(I1, I2, T5)，N_{ig}(J3)=3(J1, J2, T5) で，初期状態では N_{cig}(I3, J3)=1(T5) であるから 33 になる．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が加わり，R_{cig}(I3, J3) は 67 になる．このように，特定のゲート対に関して，R_{cit} と R_{cot} がゲート論理構造全体に依存した固定値になるのに対して，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になり，その値は一般に増大する．

ゲートマトリクス法は上記の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．

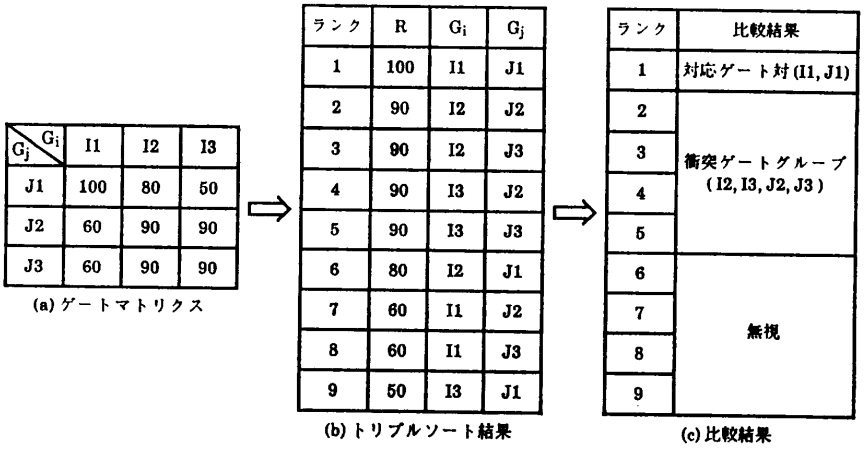
4.2 アルゴリズム

ゲートマトリクス法は R_{cit}, R_{cot}, R_{cig}, R_{cog} のいずれか（以下，R と略す）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以下に述べる．

（1） ゲートマトリクス操作

指定された R のゲートマトリクス操作手順を以下に述べる．

Step 1：与えられた後述のゲートセット（ゲートグループ）対 G_i's と G_j's の同一ゲート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



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	比較結果
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) 比較結果

図 6 ゲートマトリクス操作例
Fig. 6 A gate matrix operation example.
(a) A gate matrix, (b) The triple sort result, (c) The comparison results.

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．

Step 2: 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．

Step 3: 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．

Step 4: 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．

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

(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 が見か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，このゲート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（外部出力端子を起点にしたゲート段数）の最小値である．

(c) 上述のトリプル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 6** に示す．

(2) ゲート対応づけ操作

ゲート対応づけ操作は R_{cit}, R_{cot}, R_{cig}, R_{cog} のいずれか (R) を使用する通常操作（以下，R 操作と略す）と強制操作の5操作がある．

R_{cit} 操作手順を以下に述べる．

Step 1: 衝突ゲートテーブルの初期化を行う．

Step 2: ゲートセット対 G_i's と G_j's をセットし，R_{cit} を指定してゲートマトリクス操作を行う．

R_{cot} 操作手順は上述の R_{cit} 操作手順の Step 2 において "R_{cit}" を "R_{cot}" に置換したものである．

R_{cig} 操作手順を以下に述べる．

Step 1: 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り出す．衝突ゲートグループがなければ，本操作は終了する．

Step 2: 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，Step 1 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，Step 1 へ分岐する．そうでなければ，次の Step へ進む．

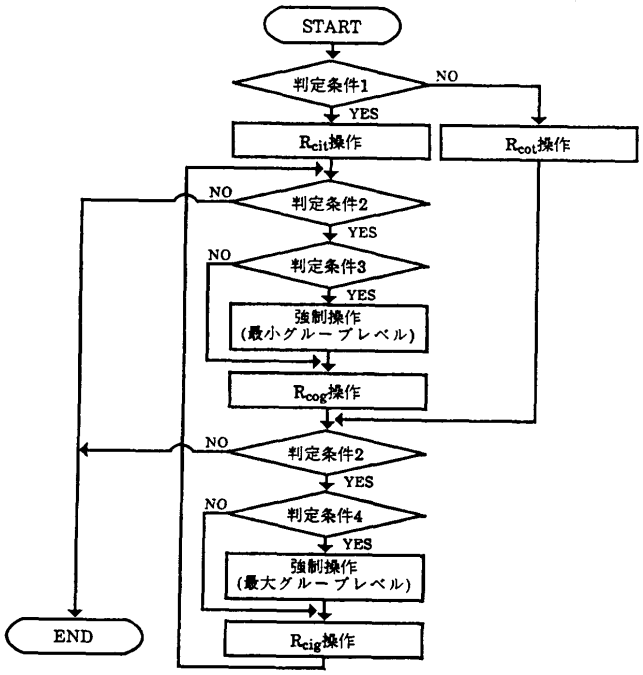
Step 3: 当該衝突ゲートグループをゲートグループ対 G_i's と G_j's に分けてセットし，R_{cig} を指定してゲートマトリクス操作を行う．ここで，このゲートマトリクス操作により新たに衝突ゲートテーブルに登録された衝突ゲートグループは今回の R_{cig} 操作の対象外とする．そして，Step 1 へ分岐する．

R_{cog} 操作手順は上述の R_{cig} 操作手順の Step 1 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 R_{cig} 操作手順の Step 3 において“R_{cig}”を“R_{cog}”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り出しは行わない．

（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7**に示す．この制御手順はゲート論理2の外部入力端子数と外部出力端子数の大小関係に依存して2種類が存在する．すなわち，外部入力端子数が外部出力端子数以上の場合は R_{cit} 操作で始め，衝突ゲートグループがなくなるまで R_{cog} 操作，R_{cig} 操作の順で各操作を繰り返す．一方，外部入力端子数が外部出力端子数未満の場合は R_{cot} 操作で始め，衝突ゲートグループがなくなるまで R_{cig} 操作，R_{cog} 操作の順で各操作を繰り返す．ここで，R_{cog} 操作，R_{cig} 操作の順で各操作を1回ず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_{cig} 操作，R_{cog} 操作の順で各操作を1回ず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：ゲート論理2において，外部入力端子数が外部出力端子数以上である．
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回ず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回ずつ続けて行っても対応ゲート対を全く認識できない．

図7 ゲート対応づけ操作の制御手順
Fig. 7 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・編集アルゴリズムはゲート論理1とゲート論理2のゲート論理構造をゲート，サブゲート，ピンの3レベルに分けて比較し，その結果を編集してゲート論理3を生成する．本アルゴリズムの処理手順を以下に述べる．

（1） 前処理

ゲート論理1とゲート論理2のネットリスト対を入力し，以降の処理の前処理としてネットテーブル対を作成する．

（2） ゲートレベルの比較処理

ゲート論理1とゲート論理2のゲートセット対 G_i's と G_j's をゲートマトリクス法を使用して比較し，対応ゲート対を認識する．

（3） サブゲートレベルの比較処理

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，入力側のサブゲートセット対 SG_i's と SG_j's をゲートマトリクス法の部分機能を使用して比較し，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_{cig} 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_{cog} 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_{cig} 操作を行うことを意味する．

（4） ピンレベルの比較処理

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，外部入出力端子，対応ゲート対，対応サブゲート対の各情報を使用してピンの接続性を比較し，対応ピン対を認識する．

（5） 編集処理

ネットテーブル上のゲート，サブゲート，ピ

ンの3レベルの対応情報に基づいてゲート論理1の対応論理部分とゲート論理2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ゲート論理3を生成する．ここで，ネットリストの編集はゲート論理内の信号線のユニーク性を保証するために非対応論理部分内の各信号線の信号名生成を伴う．また，外部入出力端子とゲートに何か他の情報が付加されている場合はその付加情報を含めて編集する．

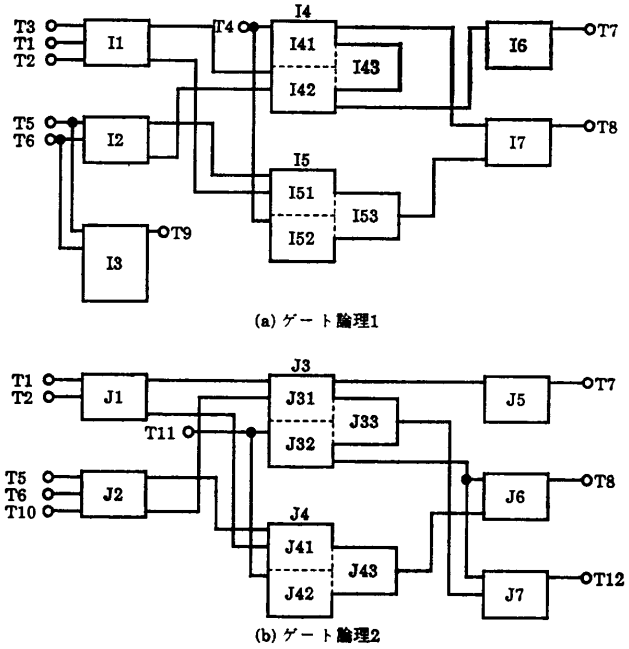


図 8 論理構造を比較・編集するゲート論理対
Fig. 8 A gate-level logic pair for logic structure to be compared and edited.
(a) Gate-level logic 1, (b) Gate-level logic 2.

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 9 R_{cit} 操作によるゲートレベルの比較結果
Fig. 9 The gate-level comparison results by an R_{cit} operation.

6. 処 理 例

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（ピンレベルの比較処理の詳細は省略する）を**図8**に示すゲート論理対を対象に以下に示す．この図において，T1～T12は外部入出力端子を表し，(I1, I2, I6, I7, J1, J2, J5, J6, J7)，(I4, I5, J3, J4) は各々同一ゲートタイプのゲートグループを表す．

（1） 図8(b)に示すゲート論理2において外部入力端子数（6個）は外部出力端子数（3個）より大きいので，R_{cit} 操作を選択する．

（2） 図8に示すゲート論理対のゲートセット対に対して R_{cit} 操作を行う．R_{cit} 操作によるゲートレベルの比較結果を**図9**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のゲート論理1とゲート論理2のゲートグループ対に対して R_{cog} 操作を行う．R_{cog} 操作によるゲートレベルの比較結果を**図 10** 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，ゲートレベルの比較処理は終了する．

（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各々の入力側のサブゲートセット対に対して R_{cig} 操作を行う．R_{cig} 操作によるサブゲートレベルの比較結果を**図 11** に示す．

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 10 R_{cog} 操作によるゲートレベルの比較結果
Fig. 10 The gate-level comparison 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 11 R_{cig} 操作によるサブゲートレベルの比較結果
Fig. 11 The subgate-level comparison results by R_{cig} operations.

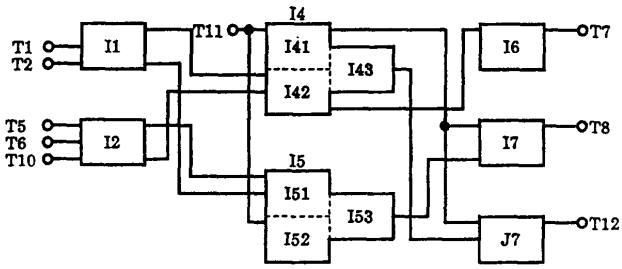


図 12 ゲート論理3
Fig. 12 The gate-level logic 3.

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比較処理は終了する．

（5） 上記の各対応ゲート対，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対してピンレベルの比較処理を行い，対応ピン対を認識する．

（6） 上記の比較結果に基づいてゲート論理1の対応論理部分とゲート論理2の非対応論理部分を選択して編集し，**図 12** に示すゲート論理3を生成する．

7. インクリメンタル論理生成への適用方法

超大型計算機用 DA システム[7),8)]の概要と本論文のゲート論理構造比較・編集アルゴリズムのインクリメンタル論理生成への適用方法を以下に述べる．

7.1 DA システムの概要

機能論理は FL (Function Logic Diagram)[9)] と呼ばれる機能論理図であり，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．

一方，ゲート論理はモジュール単位に生成され，汎用モジュールは POLARIS[10)] 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．

上記の機能論理とゲート論理において，各モジュールはモジュールを識別するためのユニークなモジュール ID を，各生成ゲートはゲートを識別するためのユニークなゲート ID を各々有している．ここで，モジュール ID は記述され，ゲート ID はモジュール ID と通番を連結して生成される．また，LSI の外部入出力端子とモジュールおよびモジュールとモジュールを接続する信号線は外部信号名が記述され，モジュール内のゲートとゲートを接続する信号線は内部信号名が生成される．ここで，両者の信号名はそれらの命名規則の相違により区別されている．

7.2 適用方法

本アルゴリズムのインクリメンタル論理生成への適用方法は以下のとおりである．

（1） ゲート論理

ゲート論理1は論理変更前の旧機能論理から論理生成されたゲート論理にレイアウトが行われたゲート論理であり，ゲート論理2は論理変更後の新機能論理から論理生成されたゲート論理である．このとき，ゲート論理3は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ゲート論理になる．ここで，ゲート論理2の生成に論理生成を使用するため，本インクリメンタル論理生成は論理生成を併用することになる．

（2） ゲート論理の付加情報

レイアウト情報はゲート，サブゲート，ピン各々の交換情報と LSI 内のゲート位置や配線順序のようなゲート本体とピンに付加される情報に分かれる．ここで，後者の情報がゲート論理の付加情報である．

（3） 処理単位

本アルゴリズムはモジュール単位に処理を行う．こ

こで，各モジュールのゲート論理1とゲート論理2はゲート ID の一部をなしているモジュール ID をキーにして検索され，この検索処理はインクリメンタル論理生成のメイン部で行われる．

（4） 外部入出力端子の端子 ID

外部入出力端子はモジュールの入出力端子であり，端子 ID の代わりに入出力端子に接続されている信号線に付与されている外部信号名を使用する．

8. 適用結果

本論文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 M 68 X 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 への適用結果をレイアウト情報の保存率と処理時間の二つの観点から以下に述べる．

8.1 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 は生成ゲート論理（ゲート論理3）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ゲート論理構造に関する類似度に基づいてゲートレベルの比較処理を行うアルゴリズムである．そのため，本アルゴリズムは，前述の4種類の類似度の基準がすべて0でなければ，すなわち，ゲート論理1とゲート論理2の間に少なくとも一つの共通の外部入出力信号（外部入出力端子）があれば，対応ゲート対の認識が可能である．これは本アルゴリズムがピンレベルの論理変更を許容したゲートレベルの論理構造比較アルゴリズムであることを意味する．この本アルゴリズムの特徴により，本インクリメンタル論理生成は $\rho=0.998$ と非常に高い比率でレイアウト情報の保存を可能にしている．

表1 適用結果
Table 1 Application results.

生成ゲート論理のゲート総数 (N_1)	1,288
レイアウト情報が保存可能なゲート数 (N_2)	1,229
ピンレベルの論理変更がないゲート数	1,039
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 25 LSI の平均値であり，3入力ゲート換算値である．

一方，本アルゴリズムはピンレベルの全面論理変更があるゲートのレイアウト情報の保存は不可能である．このようなゲートの個数はわずか3ゲートと非常に少ないので，本インクリメンタル論理生成において問題は特に生じていないが，このようなゲートのレイアウト情報の保存を可能にするには少なくとも一つの論理変更前後の外部入出力信号（外部入出力端子）の対応関係を外部から与える必要がある．

8.2 処理時間

本インクリメンタル論理生成は論理生成とゲート論理構造比較・編集からなり，これらはいずれもモジュールを処理単位として扱うことにより処理の高速化を可能にしている．ここで，ゲート論理規模が最大になるモジュールは汎用モジュールであり，1汎用モジュールは A3 サイズの機能論理図面1頁内にブール式または真理値表で記述されるため，1汎用モジュールから論理生成されるゲート論理の規模は高々 200 ゲート（論理素子）である．

上記の適用において，本インクリメンタル論理生成の処理時間は 55 秒 (on M 200 H)/2 KG 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．この適用結果は当初の目標を十分に達成しているため，これ以上詳細な評価は現在未実施である．今後の課題として，モジュールのゲート論理規模を拡大したときの取扱い可能なゲート論理規模に関する評価が残されている．

9. おわりに

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 M 68 X 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中核として使用しているため，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は 200 ゲート以下のモジュールを処理単位として扱っている

ため，その処理時間は55秒 (on M 200 H)/2 KG LSIと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．

今後の課題は本インクリメンタル論理生成の適用性拡大であり，具体的には，モジュールのゲート論理規模を拡大したときの取扱い可能なゲート論理規模に関する評価と人手による論理最適化情報の保存機能の開発が課題として残されている．

謝辞 本研究の機会を与えていただ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なら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ご意見・ご討論をいただ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）．

最後に，本論文に対して有益なコメントをいただいた査読者に深謝いたします．

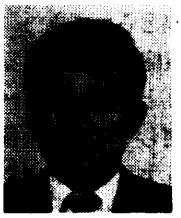
参考文献

1) Mallmann, F. P.: The Management of Engineering Changes Using the PRIMUS System, *Proc. of 17th DA Conf.*, pp. 348-361 (1980).
2) Noon, W. A., Robbins, K. N. and Roberts, M. T.: A Design System Approach to Data Integrity, *Proc. of 19th DA Conf.*, pp. 699-705 (1982).
3) Trevillyan, L.: An Overview of Logic Synthesis Systems, *Proc. of 24th DA Conf.*, pp. 166-172 (1987).
4) Kubo, N., Shirakawa, I. and Ozaki, H.: A Fast Algorithm for Testing Graph Isomorphism, *Proc. of Int. Symp. on CAS*, pp. 641-644 (1979).
5) 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).
6) 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).
7) Ohno, Y., Miyoshi, M., Yamada, N., Odaka, T., Kozawa, T. and Ishihara, K.: Principles of Design Automation System for Very Large Scale Computer Design, *Proc. of 23rd DA Conf.*, pp. 354-359 (1986).
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).
9) Ohno, Y., Miyoshi, M., Kazama, Y., Tada, O. and Sakai, T.: Design Verification of Large Scale LSI Computers, *Proc. of Int. Symp. on CAS*, pp. 443-446 (1982).
10) Shinsha, T., Kubo, T., Hikosaka, M., Akiyama, K. and Ishihara, K.: POLARIS: Polarity Propagation Algorithm for Combinational Logic Synthesis, *Proc. of 21st DA Conf.*, pp. 322-328 (1984).

（平成元年6月27日受付）
（平成2年1月16日採録）

新舎 隆夫（正会員）

昭和23年生．昭和46年横浜国立大学工学部機械工学科卒業．昭和48年東京大学工学部産業機械工学科修士課程修了．同年，(株)日立製作所システム開発研究所に入所．現在，第3部主任研究員．論理装置の設計自動化に関する研究に従事．IEEE Computer Society 会員．



森田 正人（正会員）

昭和24年生．昭和46年神戸大学工学部電気工学科卒業．同年(株)日立製作所入社．現在，神奈川工場DA設計部主任技師．この間，汎用大型計算機の開発を経て，論理設計自動化技術の開発・応用業務に従事．

越下 順二（正会員）

1960年生．1982年豊橋技術科学大学情報工学課程卒業．同年日立ソフトウェアエンジニアリング(株)入社．現在同社言語応用システム部．入社以来計算機の設計自動化・論理生成システムの開発に従事．

久保 隆重（正会員）

昭和17年生．昭和40年京都大学工学部数理工学科卒業．昭和42年同大学院修士課程（数理工学専攻）修了．同年(株)日立製作所入社．中央研究所にて汎用大型コンピュータの基本ソフトウェアの研究に従事．昭和55年より同社システム開発研究所にて，コンピュータシステムの高信頼化技術，論理設計自動化技術，汎用コンピュータ方式技術に関する研究開発に従事．現在同所企画室室長．ACM, IEEE, 電子情報通信学会各会員．