設計自動化　５０－１４
（1989.12.15）

論理合成によるＶＬＳＩ設計システム：ＦＵＳＩＯＮ

鈴木重信*，谷下久斗*，黒木加奈女*，一柳洋*，野水宣良*，吉村猛**，近藤明***

*日本電気（株）コンピュータ技術本部

**日本電気（株）Ｃ＆Ｃシステム研究所

***日本電気エンジニアリング（株）

あらまし

近年、論理合成はＶＬＳＩの設計に必須の技術となっているが、論理合成プログラムの制限から、合成だけですべての設計を終えることは難しい。主要な問題点は取り扱える規模の問題と、生成するデータパス系の回路の性能の問題である。これらの問題を解決するために本システムでは、論理合成プログラムを核にしてその制限を補完する意味で、人手マクロ機能・機能ブロック合成機能等の機能強化を行った。更に後処理として、ファンアウト調整機能・クロック再分配機能・最小遅延時間補償機能等を実現することによって実際のチップの設計に利用できるようにし、既存ＣＡＤシステムに適合するようにした。本システムは既に５０種程度のＶＬＳＩの設計に利用され設計の効率化に大きな効果をあげている。

FUSION：A VLSI Design System using Logic Synthesis

Suzuki Shigenobu*, Tanishita Hisato*, Kuroki Kaname*, Ichiryu Hiroshi*, Nomizu Nobuyoshi*, Yoshimura takeshi**, Kondoh Akira***

*Computer Engineering Division NEC Corporation

**C&C Systems Research Laboratories NEC Corporation

***NEC Engineering Corporation

1-10 Nisshin-cho, Fuchu-city, Tokyo, 183 Japan

Though logic synthesis has become indispensable for VLSI design,it is defficult to complete entire logic design by only logic synthesis. It is not able to handle large scale circuit and to synthesize hign performance data path circuit. To resolve these problems, a logic symthesis program, which is nuclear of this system, was improved at the points of manual macro facility and functional block synthesis facility. Furthermore, this system was improved at the points of fanout adjustment facility, clock redistribution facility ,and minimum delay compensation facility. As a result, it is suited to an existing CAD system. This system is applied for design of more than fifty VLSI chips and contributed to improve design productivity.

1．はじめに

近年、集積回路技術の進歩により、ＶＬＳＩの大規模化・高集積化が急速に進んできている。現在では１００Ｋゲート規模の論理ＶＬＳＩが設計可能になってきている。こうした大規模な設計を行うのに従来の設計手法で対処していたのでは効率的な設計が望めず、集積回路技術の進歩を享受出来ない。論理設計のＣＡＤ化による効率化が必須である。従来のＶＬＳＩの論理設計では、設計者は本質的な機能設計のほかにテクノロジに依存した設計ルールに従って回路の記述を行う必要があり、この後者の作業の負担が大であった。この後者の作業を自動化し、設計者を本来の論理設計作業に専念させるのが論理合成[1)-5)]のねらいである。今後の集積回路技術に見合った設計技術としては論理合成による論理設計の効率化が必要不可欠になっている。

論理合成は、機能記述言語を入力しテクノロジに依存したネットリストを出力するものであり、基本的にはコンパイル・論理の最小化・特定テクノロジ回路へのマッピングから成る。これらの研究、特に論理最小化の研究は各所で成されており、良いアルゴリズムが知られている。しかし現在の論理合成技術では論理合成を実際の設計に適用するにはいくつか問題がある。

○　論理合成プログラムが一時に適用できる回路規模には制限があり、大規模ＬＳＩを設計するには回路をいくつかのモジュールに分割して論理合成を適用する必要がある。論理設計の立場からも大規模ＶＬＳＩではモジュール化が必須である。そのため、分割した合成回路をチップにくみ上げる作業のサポートが必要である。

○　現在の論理合成の能力には限度があり、高性能コンピュータに使用するようなＶＬＳＩ設計ではどうしてもベテラン設計者のノウハウに頼らざるを得ない所がある。そのような設計上クリティカルな箇所には自動合成に対して人手介入できる機能が必要である。

○　論理合成ツールのみでＶＬＳＩの設計をするわけでなく、シミュレーション・レイアウト等の他のＣＡＤツールにリンクしなければならない。特に自動合成した回路であるが故に回路の把握が難しく、後工程の実装設計の効率が悪くなってはならない。

本システムはエンジニアリングワークステーションＥＷＳ４８００上で動作しており、論理合成プログラムを核に上記問題点を解決するため人手リンク機能等の機能強化を行い、さらに後処理として階層設計されたＶＬＳＩチップレベルの設計を可能としたシステムである。本システムにより実際に幾つかのＶＬＳＩチップが設計され、コンピュータ装置用のＶＬＳＩの標準設計手法として論理合成手法が利用されている。本稿では、ＦＵＳＩＯＮシステムの構成・機能について前記問題点解決のための項目を中心に述べ実際の適用結果について述べる。

2．システムの概要

本章では、ＦＵＳＩＯＮシステムの全体の概要について述べる。

2．1．入力言語

本システムでは入力言語としてＦＤＬ（Fuction Description Language）を用いている。図１にＦＤＬの記述例を示す。ＦＤＬはＲＴレベル記述言語であり当社においてＶＬＳＩ設計のための機能・論理シミュレーション用に広く用いられている。この言語はシステム内の入出力端子・内部端子・メモリ／レジスタ等の記憶素子・マクロモジュール等の間のデータの転送をその条件と共に記述する。ＦＤＬ中では記述性を高めるためにＣＡＳＥ文・ＩＦ文を許している。ＦＤＬは回路の論理的構造を明確に記述しており、論理合成向きの言

```
EXAMPLE
  INPUT SET,CLK,D,E,X;
  OUTPUT A;
  REG A=IF SET THEN 1
          ELSE IF CLK.UP THEN C
                        ELSE NOC;
      C=CASE X OF
           /0/ D*E
           /1/ D+E;
```

図１．ＦＤＬの記述例

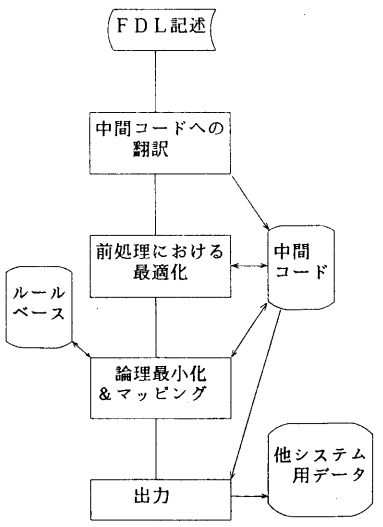


図２．論理合成プログラムの処理フロー

語である。本システムでの論理合成プログラムでは全ての記憶素子は明確に指定されているものとし、その削減などの最適化は考慮しない。

2．2．論理合成プログラム

図2に論理合成プログラム[1]の処理フローを示す。まず入力記述（ＦＤＬ）を中間コードと呼ばれるネットワーク構造に変換する。次に定数との論理演算・不要信号の除去などのテクノロジに無関係な最適化を行う。これらの処理はシステム組込みのアルゴリズム型ルールにより実現している。続いて、このネットワークを変換ルールの適用及び、論理最小化アルゴリズムの実行によって特定のテクノロジにマッピングする。最後にネットリストの形式で回路接続データ（ＶＦＩＬＥ）を出力する。このプログラムは、制御系の同期回路が主たる合成対象である。

本システムでは上記の論理合成主機能に加え、データパス系回路を人手設計してリンクする機能、テクノロジで用意された機能ブロックを有効利用する機能などを付け加えて実際のＶＬＳＩチップの作成に利用できるようにした。

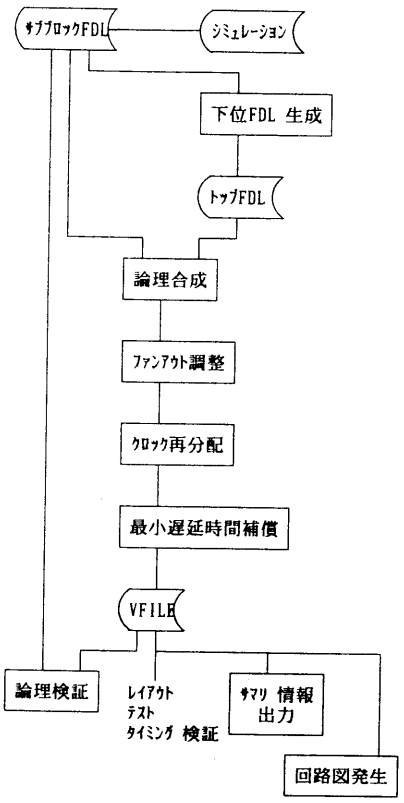


図3．本システムによる設計フロー

2．3．設計フロー

本システムによる設計手法を以下に示す。

ＶＬＳＩの仕様に従いＦＤＬを記述する。これを他チップあるいは周辺部の記述とともにシミュレーションを行い機能の検証を行う。大規模ＶＬＳＩをべたにＦＤＬで記述するのは設計の手法として得策でなく、また一時に論理合成を行うのは困難なので幾つかの機能モジュールにわけてＦＤＬを記述する。これをサブブロックと呼ぶ。これらのサブブロックに対して論理合成を行いサブブロックのネットリストを得る。サブブロック内に人手回路が含まれる場合ファンアウト調整を行い、人手回路が共通回路的な場合未使用回路の最適化を行う。各サブブロックのＦＤＬからチップ全体のＦＤＬ（各サブブロックをマクロとして扱うもの）を自動生成する。その後チップ全体のネットリストと各サブブロックのネットリストを参照してクロック信号等の再分配処理を行う。この再分配処理は論理階層を保持したまま行われる。クロック再分配の後、クロックスキューを考慮してＦ／Ｆ間の最小遅延時間補償のためディレイゲートを挿入する。最後に全体の回路に対してサマリ情報の出力及び論理回路図の出図が行われる。最終結果のネットリストに対してはもとのＦＤＬとの間の論理の一致性の自動照合が実行される。図3に本システムによる設計フローを示す。

3．論理合成

3．1．処理概要

ＦＤＬをコンパイルして得られる中間コードは内部的には論理的な構造を表現するネットワークである。図4に中間コ

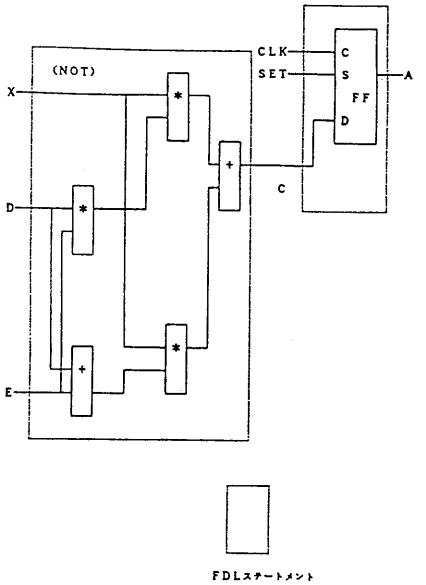


図4．中間コードの例

ードの例を示す。ノードはオペレータに対応し、アークは素子間の転送条件を示す。ノード・アークにはビット巾・極性・遅延情報・多重度等の属性が割り当てられている。このネットワーク上では否定はノードとしてではなくアークのラベルとして表現し、極性の伝搬によりインバータの削減を容易にしている。中間コードが生成された直後に不用ゲートの削除・定数演算の除去等の簡単な最適化が前処理として行われる。これらはアルゴリズム型ルールで行われる。

論理ネットワークを物理的な回路に変換する為の規則はネットワークの被変換パタンと変換パタンで表現されている。これらの規則は２種類有り、一つは論理ネットワークから論理ネットワークへの変換であり、もう一つは論理ネットワークから物理回路への変換である。論理最小化アルゴリズムは論理ネットワークから論理ネットワークへの変換を表す特殊なルールとみなしている。

変換ルール及び論理最小化アルゴリズムの適用は、ＦＤＬステートメントに対応する部分論理ネットワークに対して行われる。これは探索回数が対象回路規模及び基本素子種類数に対して指数関数的に増加するのを防ぐためである。これは全回路に対する最適性を犠牲にしているが人手設計でもほぼ同様の方法で設計されており、通常ＦＤＬ自体が中間変数をうまく抽出した形で記述されているのであまり問題はない。

ＦＤＬの１ステートメントに対しては以下のような方法で処理が成される。未処理のノードのうち出力側の隣接ノードが全て変換済みのノードを次に変換すべきノードとして選択する。適用可能なルールのうちファンインファンアウト情報・極性等の条件を満足するものを適用する。図５は同一ネットワークに適用可能な複数のルールを示している。ルールはコストでソートされているので無駄な探索は行われない。これらのルールによる変換結果のうちコスト最小のものを選択するが、ＦＤＬ１ステートメントにしてもすべての解を探索するのは実用的でなく、強力な探索打ち切り機構を準備している。探索の手法はDepth First Searchで行っており、図６に示すように処理の途中でネットワークを既変換部と未変換部に分けている。入力側の未変換部のコストの下限は容易に決定でき、この下限値が全体の最小コストを改善しなければ探索を打ち切る。この手法により前探索の９０％以上が削減されている。

３．２．強化機能

前節で述べた論理合成プログラムは強力なものであり、実験の結果でも人手に比べ遜色の無い回路を合成するが、全ての回路設計をサポート出来るわけではない。特にデータパス系の回路にはあまり適してなく人手設計を必要とする。又、実用時間内で処理を完了するために探索空間をしぼったことにより最適性が問題となることがある。これらの問題に対して以下の機能を用意することにより実際の計算機の設計作業の中で利用できるものとした。

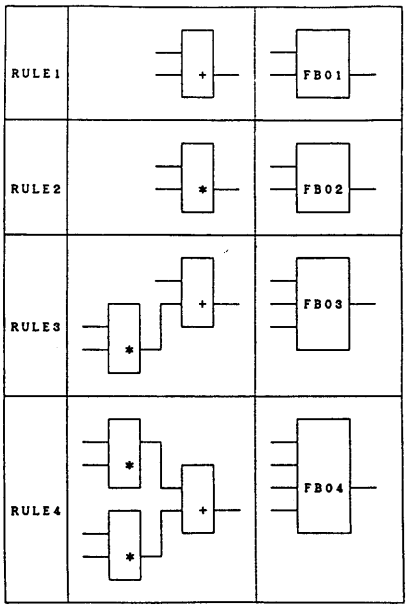


図５．変換ルールの例

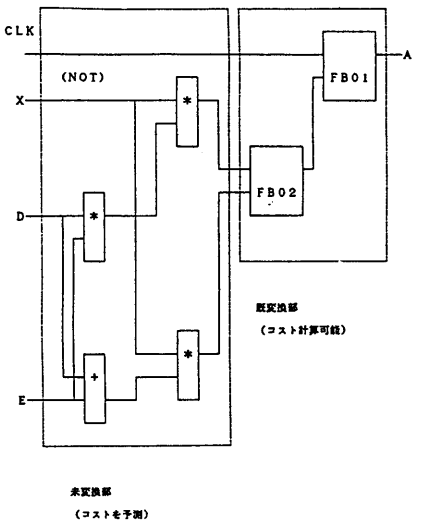


図６．コストの予測

（１）人手マクロのリンク機能

合成に対して人手介入するのに最も単純な方法は、人手設計する回路と論理合成する回路を分割し、論理合成する回路の部分に論理合成プログラムを適用することであるが、人手設計すべき部分が分散している場合ＦＤＬが細かいものになるか又は適用可能部分が減ってしまうという欠点がある。或いはＦＤＬ中にモジュール構造を取り特定のモジュールに対

応する回路を人手設計することもできるが、これはＦＤＬ全体の見易さを損なうことにつながる。

本システムでは、ＦＤＬ中に特定のマークを付加することにより、マークに囲まれた部分は論理合成プログラムがブラックボックスのマクロ回路として回路を発生する。切り口は参照・被参照の関係から自動生成するので、設計者は切り口情報に合わせてブラックボックスに相当する回路を設計する。これはＦＤＬの見易さを損なうことなく合成に対して効率的に人手介入することにつながる。図７に人手マクロのリンクの概念図を示す。

（２）機能回路生成機能

本システムではＦＤＬの特定パタンに対して特定の回路を対応させるようなルールを記述できる。これによりベイシックな変換ルールによる回路の生成以外に所望の回路を得る事が出来る。[1] この機能は主に二つの目的に利用されている。

一つは、論理合成プログラムが適していないデータパス系の回路で繰り返し使用される小規模な回路を（１）の人手マクロによらず合成可能にすることである。カウンタ・加算器等の回路を共通に登録しておくことによりＦＤＬ中で意識することなしに共通回路を利用できる。

別の利用法はテクノロジ毎に用意された機能ブロックを利用することである。通常ＬＳＩテクノロジに対応して、セレクタ・バリティジェネレータ等の機能ブロックの基本素子が

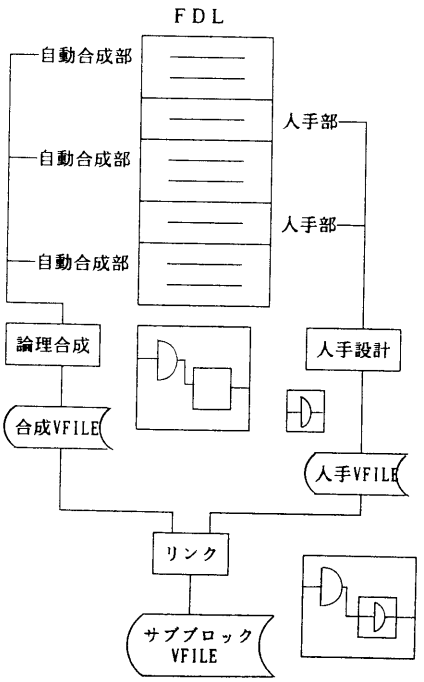


図７．人手マクロリンクの概念図

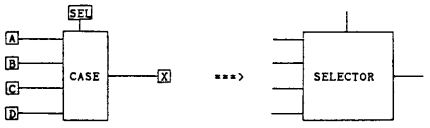


図８．機能ブロックの生成ルール

用意されているが、論理合成プログラムは中間コードから回路を生成するため、ＡＮＤ・ＯＲ等のプリミティブな素子で回路を生成する。これはピンペア数の増大を招き、チップの収容性に悪影響を及ぼす。本システムではテクノロジで用意された機能ブロックを有効利用できるようパタンを組み込んである。図８はある種のＣＡＳＥ文をセレクタブロックに変換するルールを示している。

（３）複文間の最適化

本システムでは計算時間を実用的範囲に抑えるためＦＤＬステートメント毎に最適化を行っている。通常ＦＤＬ記述で中間変数はうまく抽出されている事が多いが、中間変数の抽出だけでは最適な解が得られないことがある。つまり特定の変数が複数個所で参照されている場合、ある個所では当該変数を展開して表現し最適化を行うと良い解が得られる場合がある。このような場合設計者が印を付けることによって中間コードの段階で当該変数に相当するネットワークをコピーし、１ステートメントと同様に複数ステートメント間で最適化を行っている。

４．後処理

初めに述べたように、論理合成プログラムが一時に処理できる規模には限度があり、大規模ＶＬＳＩでは分割しての論理合成適用となる。そこで論理合成処理後にチップレベルでのチューンアップが必要となる。チューンアップの過程では結果データの見易さから、設計当初の階層関係を保持している必要があり、全ての論理変換が階層形のままで行われなければならない。又、論理合成単体ですべての設計処理が完了するわけではなく、当然既存のＣＡＤシステム　にうまくリンクしていなければならない。このため本システムでは論理合成後に適切な処理を用意することによって、論理合成によって得られた設計データをＶＬＳＩのチップレベルのデータとして既存のＣＡＤシステムに渡すようにした。以下にその内容を示す。

４．１．チップレベルの最適化処理

チップレベルの最適化処理には未使用ゲートの最適化、ファンアウト調整、クロック再分配、最小遅延時間補償ある。

（１）未使用ゲートの削除・置換

人手設計のマクロが使用された際、これらが共通的である時、使用法によっては冗長なゲートが存在する場合がある。

これは未使用の機能であったり未使用のビットであったりする。最適化処理は汎用的な論理変換プログラム[12]で行われ、変換ルールは出力オープン或いは入力オープンのブロックを検出して削除したり、入力オープンの端子を含むブロックを検出してよりファンインの少ないブロックに置き換えたりするものである。図９に未使用ゲートの削除・置換ルールを示す。

（２）ファンアウト調整

論理合成プログラムは分割されたあるブロック内だけ合成処理を行い、又、人手マクロの部分はブラックボックスとして扱うので、論理合成後のチップデータには分割合成ブロック間あるいは人手マクロとのリンク部分でファンアウト違反を起こすことがある。これらの違反箇所を人手で修正するのでは論理合成の効果が失われてしまうので、自動ファンアウト調整機能を用意している。ファンアウト違反を補正するルールには三種類有り、エラーの程度・周囲の状況に応じて適当なルールが使用される。図１０に三種類のファンアウト調整ルールを示す。この変換はやはり汎用的な論理変換プログラムで成されており、階層を保持したまま行われる。よってゲート挿入なども元の階層内に行われ、結果の確認などのため回路を追う場合も、設計者のイメージ通りの内容を追っていけば良い。

（３）クロック信号再分配

高性能の回路を設計するにはクロック分配系が等段等負荷になっていて、できるだけクロックスキューを小さく抑える必要がある。論理合成プログラムは分割単位で処理するため、全体のクロック分配系を等段等負荷にすることは出来ない。そこで論理合成後チップ全体でクロック分配系を削除し、新たな等段等負荷のクロック分配系をマージして、等負荷になるように各Ｆ／Ｆに均等にクロック信号を分配する。この処理も階層を保持した形で行われる。図１１にクロック再分配の概念を示す。

（４）最小遅延時間補償

クロックスキューは完全には除去出来ないのでＦ／Ｆ間の最小遅延時間補償が必要である。この処理はクロック系が確定しないと行なえないのでチップデータ確定後に行われる。まず回路内の各パスについてクロックスキューとＦ／Ｆのホールドタイムの合計に必要なディレイがあるかどうか調べ、不足のパスには必要ディレイを挿入するようなルールを生成する。このルールに基づいて、上述の汎用的な論理変換プログラムでディレイバッファの挿入が行われる。この処理も階層を保持した形で行われる。図１２に最小遅延時間補償のルールを示す。

４．２．変換結果の保証

人手設計の回路が含まれている場合には、論理合成後にＦＤＬと生成回路とが一致しているかのチェックが必要である。これは基本的に全体を人手設計する場合と同じである。検証方法はＦＤＬを解析して効率的かつ網羅的な検証パタンを

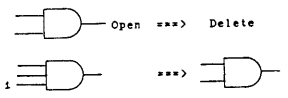


図９．未使用ゲートの削除・置換ルール

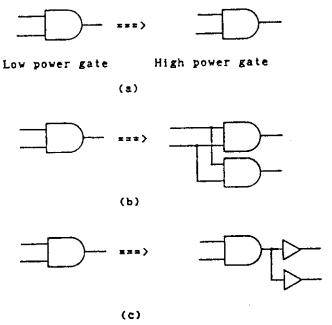


図１０．ファンアウト調整ルール

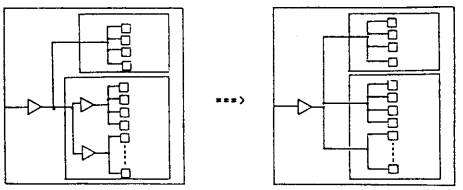


図１１．クロック再分配

図１２．最小遅延時間補償ルール

発生し、このパタンでＦＤＬ及び回路双方のシミュレーションを行い、結果のパタンを比較することで論理の一致性の検証が行われる。一致性の検証は出力端子以外にＦ／Ｆの出力端でも行われるので、不一致検出時の解析性は良い。

更に、実際のチップへの収容性判定のため、セル数・ピンペア数等のサマリ情報が出力される。

４．３．既存システムへのインタフェース

本システムで生成された回路はレイアウト・テスト・遅延解析等の既存ＣＡＤシステム[6)~10)]に入力される。これらのシステムへのインタフェースのため、本システムでは以下のような考慮を払っている。まず、生成された回路に対する回路図を発生する。この回路図の生成は、ＦＤＬ中に出現するレジスタ順を保存するように行うので、実際に記述した回路図でないものでもＦＤＬ設計者にとって見易いものとなってい

る。図１３に回路図の生成例とを対応するＦＤＬを示す。装置レベルの遅延解析などでエラーが発生した場合、人手設計と全く同様にこの回路図をグラッフィックエデイタ上で修正できる。またレイアウト設計時にフロアプランを行う場合、元の分割階層が保持されているのでこの論理マクロの構造をレイアウトマクロとして使用することが出来る。加えてＦＤＬの名称（信号名、レジスタ名）が生成回路の名称として保持されているので、各システムでの種々の指定が容易に行えるようになっている。

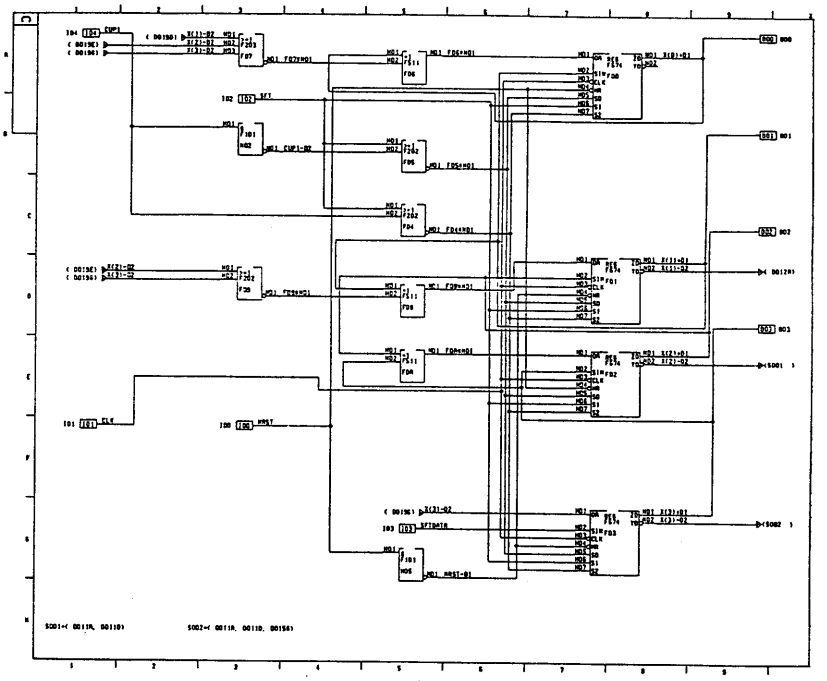
５．適用結果

本システムはすでに５０種程度のＶＬＳＩ全体の設計に利用されており、これらのＶＬＳＩの規模は１０Ｋ－６０Ｋである。本章では本システムの実際の設計への適用結果について述べる。

表１は実際の人手設計のチップについて本システムでの合成を試みゲート数の比較をした結果である。これからわかるように、本システムによって生成された回路は総体的にみて人手設計とほぼ同等の回路を生成することがわかる。この生成結果については遅延解析プログラムで何の問題もないことを確認した。

表２はある装置に使用する三チップの設計に本システムを利用した際の工数比較を示している。ここで言う工数は詳細設計以降マスクデータ作成までの工数を示しており、従来の人手設計の手法による場合を１とした値を示している。最初のチップでは工数は１／３程度にしかならないが、これは共通マクロの生成・種々の最適化ルールの作成が必要だったためである。２番目・３番目のチップの設計では最初のチップの設計で用いたルール等を利用したため、従来設計法に比べて１／５－１／７の工数で設計が完了した。

実際のコンピュータの設計では、非同期系の回路及びデー



（ａ）生成回路図

```
REGISTER  X(0:4) =
     IF MRST THEN 0                        "Master reset"
     ELSE IF CLK.DN.THEN                   "Clock"
       IF SFT THEN X(1:3)-SFTDATA          "Scan path"
       ELSE IF CUP1 THEN X .ADD. 1         "Count up"
       ELSE NOC                            "Hold"
     ELSE NOC;
```

（ｂ）対応するＦＤＬ

図１３．回路図出図例

ｻﾌﾞﾌﾞﾛｯｸ番号	人手設計ｹﾞｰﾄ数	論理合成ｹﾞｰﾄ数	論理合成／人手設計
1	400	406	102%
2	398	474	119%
3	562	513	91%
4	128	133	104%
5	975	1272	130%
6	3569	3455	97%
7	119	128	108%
8	175	148	85%
9	1530	1408	92%
10	1114	1408	126%
計	8970	9345	104%

表1．合成ゲート数の比較

チップ名	ゲート数	人手設計	論理合成
チップA	12241	1	0．36
チップB	12427	1	0．16
チップC	6227	1	0．22

表2．工数比較

タバス系の回路の一部では人手マクロのリンク機能を利用することになるが、非同期系の回路はそれほど多くなく、データパス系の回路は機能回路の生成機能である程度対応しており、制御系に比べ人手設計も容易なので人手設計に要する工数はわずかである。

6．おわりに

論理合成を利用したＶＬＳＩ設計システムについて述べた。論理合成機能がいかに向上しようとも合成では対応できない回路も存在し、一部が適用できないがために論理合成を利用できないのでは、増大する設計量に対応して効率化を進めていくのは容易ではない。本システムは論理合成プログラムを核にし、実用化のため人手マクロ・機能ブロックの生成等の人手介入の余地を与える機能強化を行い、さらに後処理としてファンアウト調整・クロック再分配・最小遅延時間補償等の処理を可能とし、既存のＣＡＤシステムとリンクすることで実際のコンピュータ設計に有効に利用できるようなシステムとした。本システムは標準的なＶＬＳＩ設計ツールとして利用されている。

今後は出来るだけ人手介入を少なくするよう論理合成プログラムの能力の向上を計っていくことに努力していく予定である。

謝辞

日頃御指導御助言を頂いた当社コンピュータ技術本部ＣＡＤ技術部高橋部長、本システム作成に御協力頂いた西木豊隆氏、山中信一氏に深く感謝します。

参考文献

[1] T.Yoshimura and S.Goto, "A Rule-Base and Algorithmic Approach for Logic Synthesis", ICCAD'86, pp.162-165

[2] D.Gregory, K.Bartlett, A.de Geus and G.Hachtel, "SOCRATES: A System for Automatically Synthesizing and Optimizing Combinational Logic", 23rd DAC(1986) pp.79-85

[3] T.Saito, H.Sugimoto, M.Yamazaki and N.Kawato,"A Rule-based Logic Circuit Synthesis for CMOS Gate Arrays", 23rd DAC(1986),pp.594-600

[4] J.A.Darringer, D.Brand, J.V.Gerbi, W.H.Joyne r, L.H.Trevillyan, "LSS: A system for Production logic synthesis", IBM J.Res.Develop. 28,no.5,pp.537-545 (September 1984)

[5] W.Joyner,Jr., L.Trevillyan, D.Brand, T.Nix and S.Gundersen, "Technology Adaptation in Logic Synthesis", 23rd DAC(1986) pp.94-100

[6] S.Kato and T.Sasaki, "FDL: A Structural Behavior Description Language", 6th International Symposium on Computer Hardware Description Language(1983) pp.137-152

[7] S.Suzuki, K.Takahashi, T.Sugimoto and M.Kuwata, "Integrated Design System for Supercomputer SX-1/SX-2", 22nd DAC(1985) pp.536-542

[8] S.Suzuki, T.Bitoh, M.Kakimoto, K.Takahashi and T.Sugimoto, "TRIP: An Automated Technology Mapping System", 24th DAC(1987) pp.523-529

[9] T.Sasaki, S.Kato, N.Nomizu and H.Tanaka,"Logic Design Verification Using Automated Test Generation", 1984 Int'l Test Conference, pp.88-94

[10]平林、鈴木、尾藤、柿本、高橋、杉本「流用設計支援エキスパートシステム：ＴＲＩＰ」情報処理学会設計自動化研究会資料、39-2 、1987.10.

[11]麻野、田中、吉村「論理自動合成システム（ＦＵＳＩＯＮ）について」第３６回情処全国大会予稿集、1988、pp1961-1964

[12]鈴木、黒木、谷下、五十嵐、飛永、水牧「統合論理設計支援システムＩＬＯＳ」第３８回情処全国大会予稿集、1989、pp1327-1337