設計自動化 61－3
（1992. 2. 21）

ＦＤＬを用いた ゲートレス・デザインシステム

尾藤　龍茂

日本電気（株）

各種コンピュータ開発に用いるＨＷ記述言語ＦＤＬ（Functional Description Language）ベースの設計法及びそのＣＡＤシステムを述べる。大規模化する高性能コンピュータ開発の大幅な効率化を計るため、ＬＳＩの各設計フェーズにおいて一貫して回路図を見ない、言語ベースの設計を可能にするＣＡＤシステムを構築した。本ＣＡＤシステムは、論理合成ツール、高速のＦＤＬシミュレータ、遅延保証／レイアウト／テストの自動化ツール等より構成される。本ＣＡＤシステムによる言語ベースのＬＳＩ設計を汎用コンピュータＡＣＯＳシリーズ等に適用し、従来の回路図ベースの設計に較べ、設計期間及び設計工数を大幅に削減することが出来た。

GATELESS DESIGN SYSTEM USING FDL

Tatsushige Bitoh

NEC Corporation

This paper describes a CAD system based on a high level design methodology which utilizes an HDL,called FDL(Functional Description Language).The system was developed to shorten design TATs(Turn Around Times) for increasingly complex designs. To fulfill that need, several tools were integrated around FDL so that designers didn't need to be concerned with detailed schematic diagrams in various phases of LSI design. This system consists of a logic synthesis tool,high-speed FDL simulators, timing analyzers, layout tools and test tools. This FDL based system was applied for designing chips for computers including NEC's ACOS main frame series. Compared with the previous design methodology based on schematic diagrams, drastic reduction was achieved in TAT and manpower cost.

１．まえがき

当社は、昭和５０年代半ばより、まず、メインフレーム開発にハードウェア機能記述言語ＦＤＬ（Ｆｕｎｃｔｉｏｎａｌ　Ｄｅｓｃｒｉｐｔｉｏｎ　Ｌａｎｇｕａｇｅ）の使用を開始した。それ以降、ＣＡＤシステムの機能強化を積み重ね、現在、ＦＤＬを利用した言語ベースのＬＳＩ設計法（ゲートレベルを意識しない設計＝ゲートレス・デザイン）を各種高性能情報処理機器のＬＳＩ開発に全面的に適用している。

従来の回路図ベースの設計から、ＦＤＬを利用した言語ベースの設計を採用した背景を図１．１に示す。本図に示すごとく、大規模化・複雑化する製品開発目標を達成する設計効率化を実現するためには、機能記述言語の利用による設計の高レベル化が必須である。しかし、高性能装置のＬＳＩ開発においては、機能記述言語の利点を最大限引き出したゲートレス・デザインによる大幅な効率化計るためには、図１．２に示すように、単に論理設計の効率化を計るばかりでなく、装置全体で所定の性能を実現する遅延保証の自動化等も含めたトータルな設計支援がＣＡＤシステムに求められる。

この観点より、我々は高性能ＬＳＩの開発においても、設計の各フェーズで一貫して回路図を見ないでＦＤＬ記述のみで設計を進めることが出来るＣＡＤシステムを構築し、汎用計算機ＡＣＯＳシリーズ等のＬＳＩ開発に適用して大幅な設計効率化を達成した。本論文は本ＣＡＤシステムを使用したＬＳＩ設計法およびＣＡＤシステムの概要を以降に述べる。

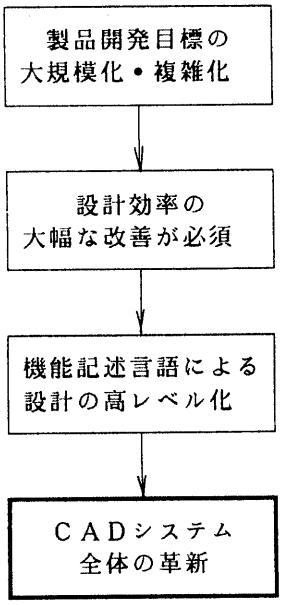


図１．１　ＦＤＬ開発の背景

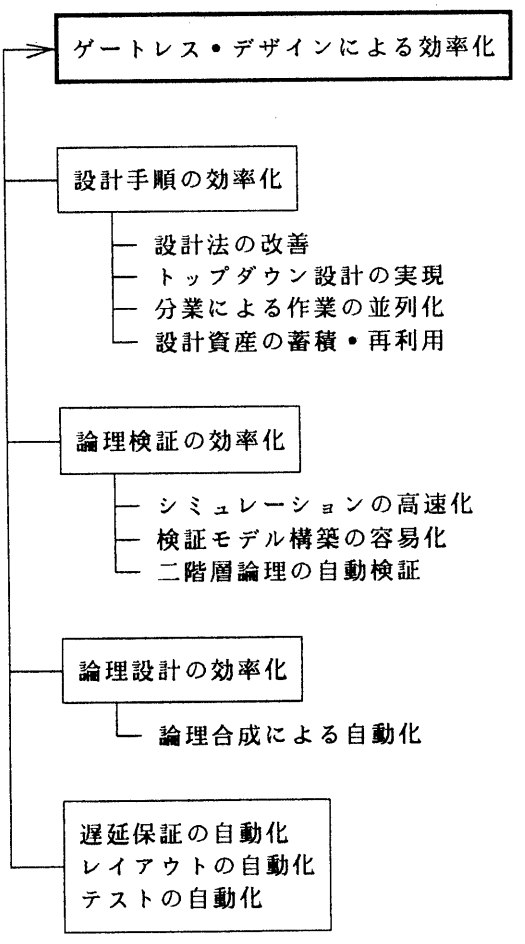


図１．２　ＣＡＤシステムの狙い

２．　ＦＤＬ利用によるＬＳＩ設計法の変化

ＬＳＩ設計作業全体から見た設計効率化及び設計期間短縮を果たすためには、論理設計作業時の作り込みバグ数を減らすことと、作り込まれたバグを検出する論理検証作業を効率化することが最も重要な条件となる。図２．１に示す我々が採用したＦＤＬを利用した新設計法は、この二つの条件を満足している。新設計法では、まず、設計者がＦＤＬ記述の作成が完了した時点で、ゲートレベルよりはるかに高速なＦＤＬレベル・シミュレータを使用して論理検証を効率的に行う。そして、設計ミスの検出および修正が終了した段階で、詳細設計以降の作業をほぼ自動的に行い、設計工数を削減するばかりでなく、人手設計作業によるバグの発生を未然に防止している。

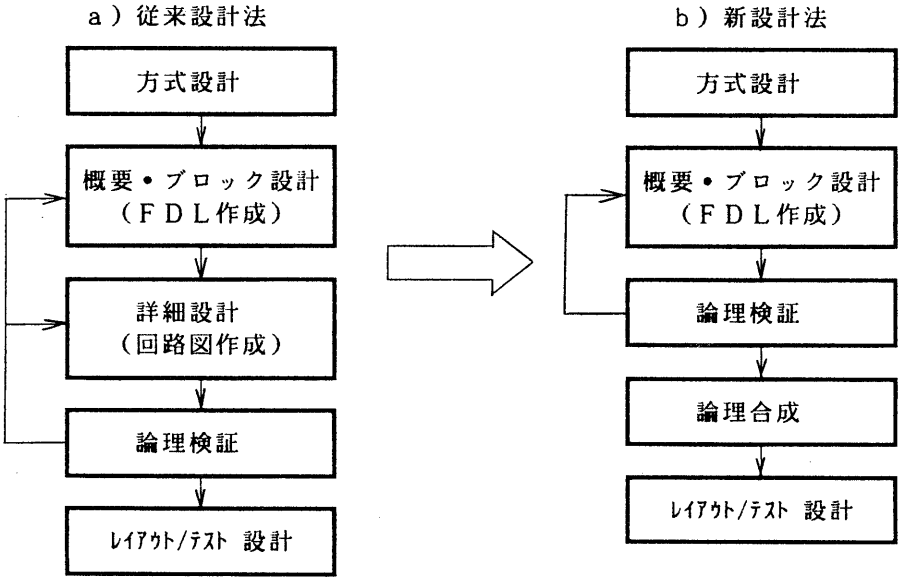


図２．１　設計法の変更

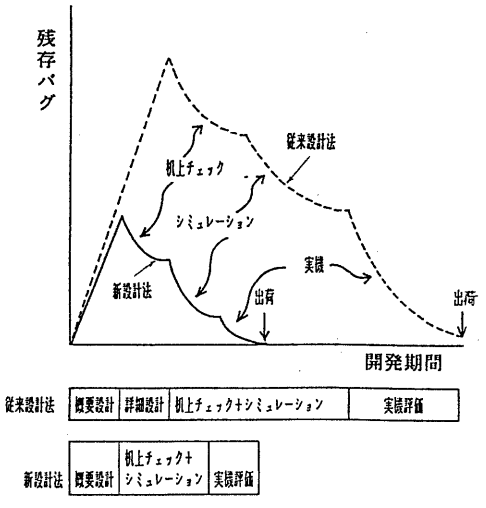


図２．２　設計期間の短縮効果

図２．２は、新設計法における設計期間の短縮効果をバグの発生と検出の観点より示す。本図に示すごとく回路図ベースの従来設計法に比べ、新設計法では①回路図を人手作成する必要が無いので作り込みバグ数が少ない、②回路図より少量のＦＤＬ記述を対象とするため机上チェックが効率的である、③シミュレーションが高速であるので短期間に多くのバグが検出できる、④その結果として、実機検査に持ち込まれるバグが少なく実機評価時間が短くなる。このように新設計法はバグの発生を最小限に押さえ、かつ残存バグをより短期間に摘出することにより、ＬＳＩ設計期間全体の大幅な短縮を可能にしている。

3．　ＦＤＬの概要

ＦＤＬ[7] は、ハードウェア論理に対する高レベルな機能表現能力を保有している一方、ＬＳＩ内の特に高性能な部分に対しては必要な詳細度に応じた記述能力を有している。よって、設計者はハードウエア・ブロック図と対応させながら、性能的な面も考慮しつつＦＤＬ記述を必要な詳細度で作成することができる。

また、ＦＤＬは非常にシンプルな言語であり、以下に示す７つの文でハードウア論理を記述することができる。このため、設計者がＦＤＬ言語を習得し利用するのが極めて容易であり、ＦＤＬベースの設計法が幅広く浸透する要因となっている。

(1) ＩＮＰＵＴ／ＯＵＴＰＵＴ／ＩＮＯＵＴ文

これらの文は、モジュール間のインタフェース信号を定義する。ＩＮＰＵＴ文は入力信号、ＯＵＴＰＵＴ文は出力信号、ＩＮＯＵＴ文は双方向信号をそれぞれ定義する。

例）
```
INPUT  CLK, DATAIN(0:32), M(64);
OUTPUT  ALUOUT(0:32), OVF;
INOUT  DBUS(0:32), CRYIO;
```

(2) ＭＯＤＵＬＥ文

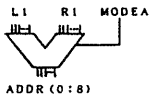
他のモジュールを呼び出すために使用する。

例）
```
MODULE  LSI1=CXXXX(LOAD  D(0:8)、CTL1、
    SOURCE  DO(0:16)、BIDIRECT  A(0:16));
```

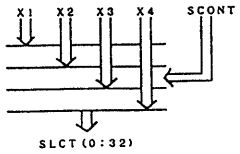
(3) ＴＥＲＭＩＮＡＬ文

組合せ回路を記述する。

例）
```
TERMINAL ADDR(0:8)<20, 30, 40>=IF MODEA
    THEN L1.ADD.R1 ELSE L1.SUB.R1;
```




```
TERMINAL SLCT(0:32)=
    CASE SCONT(0:2) OF X1, X2, X3, X4;
```

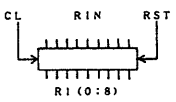


(4) ＲＥＧＩＳＴＥＲ文

レジスタ単体、あるいは、組合せ回路とレジスタをまとめて記述する。

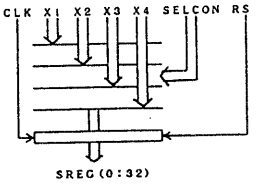
例）
```
REGISTER R1(0:8)=
  IF RST THEN 0 ELSE IF CL.UP. THEN RIN;
```

RST	C L	R 1
1	——	0
0	↑	RIN
	0. 1. ↓	HOLD




```
REGISTER SREG(0:32)=
  IF RST THEN 0 ELSE IF CLK.UP. THEN CASE
  SELCON OF X1, X2, X3, X4;
```

RST	CL	SCONT	SREG
1	—	—	0
0	↑	0	X1
		1	X2
		2	X3
		3	X4
	0. 1. ↓	—	HOLD

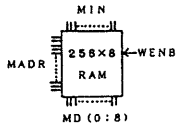


(5) ＭＥＭＯＲＹ文

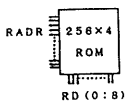
メモリ単体、あるいは、組合せ回路とメモリをまとめて記述する。

例）
```
MEMORY MD(0:256, 0:8)=
    IF WENB THEN WRITE MIN AT MADR
    & READ AT MADR;
```

WENB	action	
0	——	READ
1	WRITE	




```
MEMORY RD(0:256, 0:4)=READ AT RADR;
```



４．　ＣＡＤシステムの概要

本章では、ＦＤＬによるＬＳＩ設計を支援する主たるツールを紹介する。

４．１　設計フロー

図４．１に、２章で述べた設計法を具体化したＬＳＩ設計フローを示す。本フローの運用は、ＥＷＳ４８００及びＡＣＯＳマシンのネットワーク環境下で行われている。

本フローでのＬＳＩ設計は、通常、以下の手順で行う。

(1) まず、方式設計及びブロック設計の過程において設計者がＦＤＬ記述を作成する。

(2) ＦＤＬ記述が完成すると、ミックスレベル・シミュレータ（ＭＩＸＳ）[2] を用いて、ＦＤＬ記述に対してＬＳＩ単体のシミュレーションを行う。

(3) 同時に人手設計部分については回路図入力（ＶＩＳＴＡＳ）を行い、２階層論理検証ツール（ＥＶＥＲＹ４）[5] を用いて回路図がＦＤＬ記述と論理的に等しいことを保証する。

(4) 各ＬＳＩの単体検証が終了した時点で、各ＬＳＩを含んだシステムレベルの検証をハードウエア・シミュレータ（ＨＡＬⅢ）あるいはミックスレベル・シミュレータ（ＭＩＸＳ）で行う。この時、ＬＳＩにはＦＤＬ記述を、カード及びボードにはネットリストを使用する。

(5) システムレベルの検証と平行して論理変更の修正をＦＤＬ記述に加えながら、論理合成以降の自動化ツールを実行しＬＳＩ製造用データを作成する。

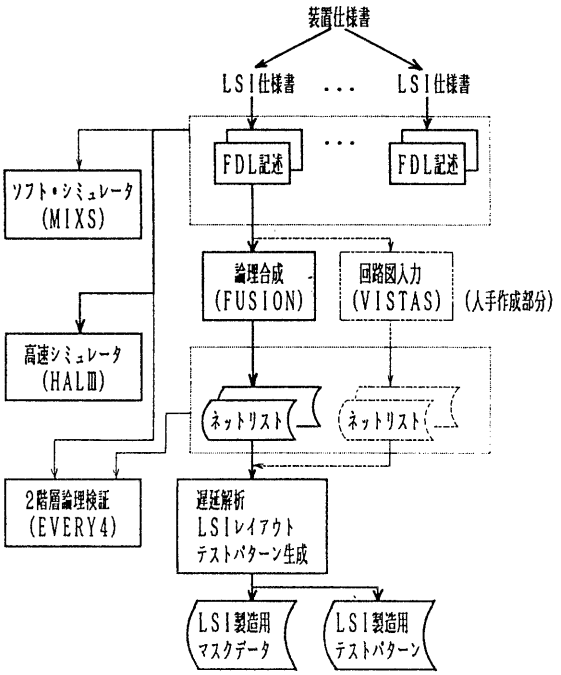


図４．１　ＬＳＩ設計フロー

本手順に示すように論理検証のためにはＦＤＬ記述に対するシミュレーションを行うだけであり、ゲートレベル・シミュレーションは行わない。また、システム・レベルの大規模なシミュレーション・モデルをハードウエア・エンジンを用いたＨＡＬⅢにより高速に検証を行うことにより、論理検証に要する期間の大幅な短縮を実現している。更に、本手順では人手設計による詳細回路の検証をＦＤＬ記述の検証と平行して行なうため、人手設計による設計期間増加は生じない。

本手順をスーパーコンピュータＳＸシリーズ、汎用計算機ＡＣＯＳシリーズ、ＥＷＳ４８００シリーズのＬＳＩ開発に適用しており、ＦＤＬ記述作成率は１００％、論理合成対象ＬＳＩはゲート数比で９０％、品種数比で約１００％を達成している。

４．２　ＦＵＳＩＯＮ（論理合成ツール）[6]

図４．２に構成を示すＦＵＳＩＯＮは、ＦＤＬ記述から面積及び遅延の最適化を行ったゲートレベルのネットリストを自動的に生成するばかりでなく、合成されたネットリスト間の必要な接続を自動的に行い、ＬＳＩチップ全体の最終的なネットリストを自動的に組み上げることが出来る。現在までに、汎用計算機やワークステーションで使用されているＣＭＯＳやＣＭＬのゲートアレイ／スタンダードセル方式の約２００種以上の高性能ＬＳＩ開発にＦＵＳＩＯＮを適用し、表４．１の適用結果例に示すような大幅な設計工数削減を実現している。このように多くの高性能ＬＳＩにおいて、論理合成を実用化するために、基本的な要件となる最適化アルゴリズム[4]以外にＦＵＳＩＯＮが工夫した点を以下に述べる。

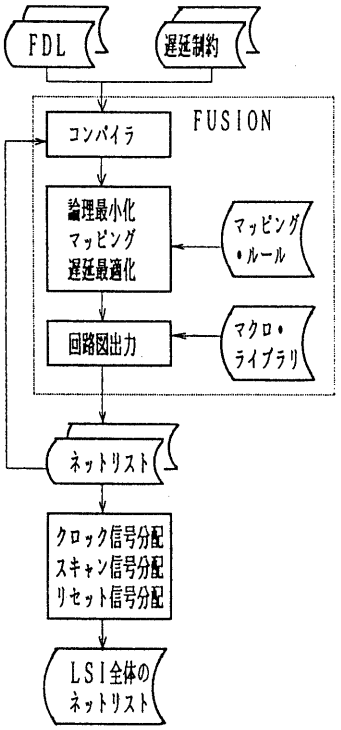


図４．２　ＦＵＳＩＯＮの構成

表４．１　ＦＵＳＩＯＮの適用例

品名	A	B	C
ＦＤＬ行数	1,214	1,383	1,253
ゲート数	12,241	12,427	6,227
設計工数	3.9人月	2.1人月	1.6人月
従来工数比	40%	21%	32%

(1) 遅延制約の入力

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ＬＳＩではモジュール化は必須である。しかし、遅延制約条件は、論理合成単位であるモジュール内ばかりでなく、複数のモジュールを組上げたチップ内で、更には複数のＬＳＩチップを組み込んだ装置内においても満足させる必要がある。このため、ＦＵＳＩＯＮでは、合成モジュールの入出力ピンに対して遅延及びファンアウトの制約を遅延制約ファイルで指定可能にすることで、合成モジュールがチップあるいは装置として組み上げられた場合にも問題なく遅延制約条件を満足させることができる。また、後述の図４．４に示すように、遅延解析システムの結果に基づいて遅延制約ファイルの自動作成・更新を可能にしており、遅延制約を正確に作成する手間を軽減している。

(2) 人手マクロとのリンク機能

論理の規則性を利用し性能向上やレイアウト面積縮小を計るデータパスの設計及び非同期回路の設計では、論理合成ではなく人手設計に頼らざるを得ない面がある。このような場合、人手介入の最も単純な方法は、人手設計する部分と論理合成する部分を分割し、ＦＤＬ記述を行うことである。しかし、この方法では、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合成可能部分が減ってしまうという欠点がある。この欠点を避けるために、ＦＵＳＩＯＮでは、ＦＤＬ中に特定のマークをして人手設計部分を識別する方法を採用している。マークに囲まれた部分はＦＵＳＩＯＮ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。設計者は自動生成された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か、標準的な回路を対象としたマクロ・ライブラリから使用回路を選択する。

4．3　HALⅢ（ハードウエア・シミュレータ）[1]

HALⅢは、ソフトウエア・シミュレータであるＭＩＸＳと同等の機能を有し、ＦＤＬ記述を直接、超高速にシミュレーションできるハードウェア・シミュレータである。ＬＳＩの設計品質を確保するにはシステムレベル・シミュレータは必須であり、HALⅢは高集積なＬＳＩを多数搭載した大規模な装置においても、高速なシミュレーションを可能にするために開発された。

(1) 超高速なシミュレーション

図４．３に示すように、HALⅢはシミュレーション用プロセッサを最大１２７台並列に動作させることにより、大規模回路を高速にシミュレーションする。各プロセッサはＦＤＬシミュレーション専用の構造となっている。

プロセッサが３１台構成の場合、当社製ゲートレベル・ソフトウエア・シミュレータの約１万倍の実行速度を持ち、約５００万ゲート規模の装置の装置試験プログラム（ファンクション・テスト部分、(2) 参照）を約３日で完了させることが出来た。

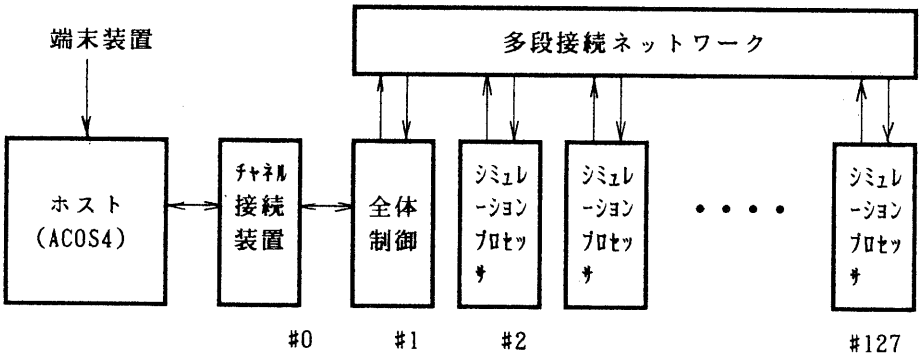


図４．３　HALⅢシステム構成

(2) 装置試験プログラムの実行

装置レベル・シミュレーションでは、人手作成のテストパターンを入力することも出来るが、通常、実機の検査に用いられる装置試験プログラムを用いる。装置試験プログラムは、装置を基本機能単位で試験するファンクション・テスト部（ＦＴ）と基本機能を任意な順序で長時間実行するランダム・テスト部（ＲＴ）とから成る。HALⅢシミュレーションでは、まずＦＴの実行により装置の基本機能を個別に確認した後、長時間ＲＴの実行により、実際の使用条件と近い形で装置の設計品質を確認する。

(3) 網羅率の測定

装置試験プログラムが装置全体をどれだけ網羅して試験したのかを知るために、HALⅢはハードウェアやファームウェアの網羅率を測定することができる。ハードウェアの網羅率として、全信号線数と値が変化した信号線数の比を用いる。また、ファームウェアの網羅率として、全ＦＷステップ数と実行ＦＷステップ数の比、あるいは、全ＦＷパス数と実行ＦＷパス数の比を用いる。

網羅率が１００％になっても設計品質を１００％保証できる訳ではないが、設計確認がどの程度行われたかを客観的に把握するため、また装置試験プログラム強化の目安データして使用している。

４．４　遅延解析／レイアウト／テスト

機能記述からの論理合成を主体とした設計法では、遅延検証、ＬＳＩレイアウト、ＬＳＩ製造用テストパターン作成・検証の各処理は完全に自動化できていることが必要である。これらの処理においてゲートレベルの回路図ベースで相当量の人手作業が発生した場合、回路図が設計者にとって馴染みが無い自動発生された回路図であるため、かなりの設計工数と期間が費やされてしまうことになる。

このような状況に陥らず真にゲートレスデザインの効果を発揮するために、これらの三つの処理の自動化を推進する以下のツールの開発を行った。

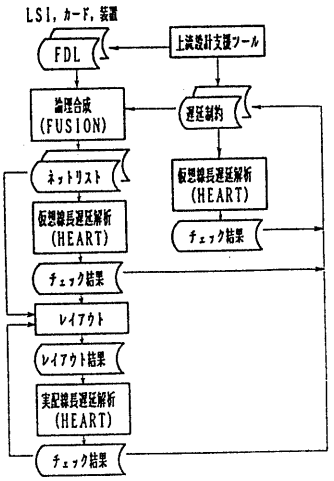


図４．４　遅延保証フロー

(1) 遅延解析ツール（ＨＥＡＲＴ）

４．２章でも述べているように、論理合成結果をＬＳＩあるいは装置に組み込んだ状態で遅延制約が満足されていなければならない。本ツールは、ＬＳＩや装置単位で網羅的な静的遅延解析を行い、論理合成結果の最終的な確認を行うばかりでなく、図４．４に示すように遅延解析結果に問題がある場合にその結果を論理合成ツールの遅延制約ファイルへフィードバックすることができる。この場合、更新された遅延制約ファイルを入力して論理合成の再実行を行い、ＬＳＩや装置レベルの遅延を保証する。

また、高性能ＬＳＩでは、レイアウト結果によって遅延制約違反がしばしば生じるため、図４．４に示すように遅延解析結果をレイアウト・ツールにもフィードバックしている。レイアウト・ツールは、遅延制約違反を起こした信号パスを優先して配置処理を行う。

(2) ＬＳＩレイアウト・ツール（ＭＡＳＴＥＲ３）[3]

前述のように、高性能ＬＳＩを論理合成で実現するために、遅延解析ツール等の結果に基づいてクリテイカル・パスを優先的に配置・配線する機能をサポートしている。更に、設計者のノウハウを十分に発揮できるように各種人手指定を容易化している。ＭＡＳＴＥＲ３では、ＦＤＬ記述上のレジスタ名やメモリ名で強制配置指定が可能であり、ＦＤＬ記述単位のモジュールや複数モジュールを統合した単位でグルーピング処理を行うことが出来る。また、回路図レベルでの細かな設計を不要にするため、クロック信号やスキャンパスやクランプ系信号のネットの付替処理を自動的に行う機能を実現している。

(3) 製造用テストパターン発生・検証ツール（ＦＵＴＵＲＥ）

ＬＳＩの製造検査用の高検出率のテストパターンを自動発生することは、テスト設計工数を削減するだけでなく、製品の信頼性を確保する上でも、極めて重要である。我々は、制御回路等のランダム・ロジックにはスキャンパス回路を、ＲＡＭやＲＯＭ等の特殊回路にはＢＩＳＴ回路を採用し、スキャンパスを用いた高検出率のテストパターンを自動生成している。スーパーコンピュータＳＸ－３や汎用超大型機ＡＣＯＳ３８００等で使用しているＣＭＬゲートアレイでは、人手介入無しで平均９８％以上の検出率のテストパターンを自動発生できた。

5．　今後の課題

以上説明したＦＤＬ言語ベースの設計をサポートするＣＡＤシステムを構築し、多くのＬＳＩ開発に適用することにより、ＬＳＩ設計期間及び設計期間を１／２以下に削減することが出来た。しかし、更に改善を計るため、ＣＡＤシステムの機能拡張への要求は高まる一方である。我々のＣＡＤシステムが今後取り組むべき主要な課題を以下に述べる。

(1) 設計上流工程のサポート強化

ＦＤＬはブロック図レベルのＬＳＩ論理仕様を記述する言語として利用されてきた。しかし、設計者がＬＳＩ論理仕様を記述する際、表やフローチャートやタイムチャートを用いて表現したほうが効率的な場合がある。これらの表現手段をＦＤＬ記述と組合せて、上流工程の設計効率の改善を推進する必要がある。また、上流工程の設計品質がＬＳＩ開発全体の効率を大きく左右する要因であるとの観点より、我々は上流設計支援システムＯＺ（オズ）の一次版を完成し、ＬＳＩ開発への適用を開始した。

(2) 詳細設計以降のより完全な自動化の実現

ＦＤＬ記述が完成した後の設計作業を完全には自動化できない最大要因は、遅延保証問題である。ＬＳＩを組み込んだ装置全体が所定のクロックサイクルで動作することを保証するためには、単に論理合成ツールを強化すれば済む話ではなく、カードやボードのレイアウト・ツール、ＬＳＩレイアウト・ツール、論理分割ツール等を含んだＣＡＤシステム全体で遅延保証を実現する仕組みを作り上げねばならない。特に、高性能装置の開発においては、論理合成ツールの配線長予測と連動してＬＳＩ内の配置・配線を行う遅延考慮レイアウト・ツールの必要性が高い。

［参考文献］

(1) 高崎他「ＨＡＬⅢ：機能レベル・ハードウエア・シミュレータ・システム」情報処理学会論文誌、１９９１、Ｖｏｌ．３２　Ｎｏ．１、ＰＰ．８６－９９
(2) 佐々木他「ＭＩＸＳ（ミックスレベルシミュレータ）の概念」他、情報処理学会第２１回全国大会論文集、１９８０、ＰＰ１１５９－１１６８
(3) 矢部他「高性能ＬＳＩのレイアウト設計法」、設計自動化研究会資料、１９８９、設計自動化４６－１
(4) 吉川他「ＴＩＭＩＮＧ　ＯＰＴＩＭＩＺＡＴＩＯＮ　ＯＮ　ＭＡＰＰＥＤ　ＣＩＲＣＵＩＴ」、２８ｔｈ　ＤＡＣ、１９９１、ＰＰ１１２－１１７
(5) 飛永他「論理検証ツールＥＶＥＲＹ４」他、情報処理学会第３８回全国大会論文集、１９８９、ＰＰ１３３５－１３３６
(6) 鈴木他「論理合成によるＶＬＳＩ設計システム：ＦＵＳＩＯＮ」、信学技法、１９８９、ＶＬＤ８９－８８，ＰＰ．３７－４４
(7) 加藤他「ＦＤＬ：Ａ　ＳＴＲＵＣＴＵＲＡＬ　ＢＥＨＡＩＶＩＯＲ　ＤＥＳＣ－ＲＩＰＴＩＯＮ　ＬＡＮＧＵＡＧＥ」、ＣＨＤＬ、１９８３、ＰＰ．１３７－１５２