設計自動化 38-3
（1987・7・23）

ミックスシミュレータFALの ASIC設計への適用

山岸 邦彦，市村 徹，矢野 栄一，関根 優年

（株） 東芝 超LSI研究所

LSI設計の大規模化に伴なって、その設計検証はますます困難になってきた。この困難さを解決する方法として、ハードウェアの動作及び構造に即したモデリングをHDLで記述し、そのモデルをHDLシミュレータにより検証する方法が提唱されている。また、大規模なLSIの設計検証の効率を向上させるために、そのモデルとしてRTレベル，logic レベルの混在したようなHDL記述を用いる必要がでてきている。このようなモデルの検証ツールとして、我々は“ミックスレベルシミュレータFAL”の開発を行ってきた。実際に大規模なLSIの設計検証におけるFALの有効性を確認している。今回はFALによる設計検証をより効率的に行うため、今までの適用例についてまとめたのでそれらの結果を報告する。

An Application on ASIC Design with Mixed-level Simulator FAL

Kunihiko YAMAGISHI , Tohru ICHIMURA , Eiichi YANO , Masatoshi SEKINE

VLSI Research Center, TOSHIBA Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 211, Japan

This paper describes a structure model at the novel mixed level simulator FAL. The structural models for standard products LSIs are named as a high level model in this paper. To comfirm the utilization of the high level model, an example of the super integration which consists of the Z80 microprocessor, a clock generator and two user-defined logic circuits, is simulated with FAL. The total circuit of the super integration can be simulated concurrently with FAL. FAL was also applied to the functional design verification of several VLSIs. All the results of the investigation demonstrates the effectiveness of FAL and the high level models for ASIC design.

1　はじめに

半導体製造技術の急激な技術革新により、ＬＳＩの集積度は急速に高くなっている。それにともなって、より大規模かつ複雑なシステムを実現するＬＳＩが設計されている。しかし、ＬＳＩの設計が大規模かつ複雑になるにしたがって、ＬＳＩ作成前に完全な設計検証を行なうことは、ますます困難になってきている。

この困難を解決するために、以下の２つの方法による設計法が提案されている。１つは、規模の大きい既存の回路を標準回路（メガセル）としてライブラリ化し、使用する部品数を減らしていく方法である。もう１つは、ＲＴＬ（Register Transfer Level）で機能記述を行い、論理合成および自動レイアウトによりＬＳＩを実現する方法である。

上記どちらの手法をとるにしても、設計の検証を行なう上では、ＬＳＩの"behavioral model"が重要である。すなわち、ハードウェアの動作および構造に即した記述が可能なＲＴＬによるモデリングを行いそれらをlogic レベルと混在させながら検証することが必要となる。

以上の視点に立って、我々はミックスレベルシミュレータＦＡＬの開発を行ってきた。ＦＡＬではＬＳＩレベルのセル（メガセル）であるＳＭＣ（Super Macro Cell）をライブラリーとして使用することができ、さらにユーザが定義したマクロセルとのミックスシミュレーションを行うことが可能である。

今回は、これまで開発を行ってきたＦＡＬのＡＳＩＣ設計におけるさらに有効な適用方法を提供するために、機能記述の方法や、既存のＬＳＩをＳＭＣライブラリとして用いる場合のモデリングの手法、及びライブラリーの作成方法について検討した。

2　ＡＳＩＣ設計

一般にＬＳＩの回路設計は以下のフローに従って行われる。

1）．要求仕様からのシステムのモデリング
2）．機能仕様のまとめ（機能設計）
3）．機能分割およびハードウエアによる回路構成
（論理設計）
4）．構成回路の性能評価および修正

上記の各段階の内どこに重点を置くかは、対象回路が新しいシステム，類似回路，既存回路のバージョンアップ等のいずれであるかに依存する。

ＡＳＩＣでは既存回路をＬＳＩ化してシステムの性能向上を計る場合が多く、その設計手段として、ＧＡ（Gate Array）やＳＩ（Super Integration）[1] といった方法がとられている。これらの方法を用いることによって、ＬＳＩを短期間に設計することが可能となる。さらにＳＩでは、ＣＰＵ，ＤＭＡ，ＳＩＯなどのＬＳＩからなる回路を１つのＬＳＩに集積することができるため、システムのコンパクト化も容易になる。

これらの方式では、既存のＬＳＩを部品として使用するので、設計上の重点は機能設計に置かれることになる。そのため、ＨＤＬ（Hardware Design Language）による設計が行われている。

ＨＤＬによるハードウェアの設計は、ソフトウェアの開発と類似点が多い。（Table 1）　しかし、ソフトウェアと比べて、ハードウェアの場合、並列動作という点が複雑であるためＨＤＬシミュレータにより検証を行なうことが必要となる。実際にＨＤＬシミュレータを用いることによって、ハードウェアもソフトウェアと同様の手順で開発することができた。

このような大規模なＡＳＩＣの設計検証のためには機能レベルのモデルが重要であるが、回路のインプリメントを考慮するうえでは、ディレイを含める必要がある。そこでＨＤＬに遅れの表記を導入した。

2．1　ミックスレベルシミュレータＦＡＬ

システム検証を行なうためのＣＡＤのシステム構成図をFig.１に示す。本ＣＡＤシステムではシステムの初期設計段階からの支援が可能である。シミュレーションの入力としては、テスト記述言語ＴＳＴＬ２（Toshiba Standerd Test Language 2）[2] から変換したテストベクトルと、汎用アセンブラによってアセンブルされたマイクロプログラムとを使用する。検証が終わったＲＴレベル記述から、論理合成システム[3] を使って論理記述を生成する。これを用いて論理シミュレーションまたは遅延解析を行なう。

設計検証後、スーパーマクロセルを含んだ論理記述をもとに、階層的なレイアウト[4] を行なう。必要に応じてミックスレベルシミュレーションでのタイミングの確認を行なう。

このように、このＣＡＤシステムでは、機能設計からレイアウトまでを一貫して支援することができる。スーパーマクロセルに対しては、ＲＴレベルによるミックス

	HARDWARE DESIGN	SOFTWARE DESIGN
SPECIFICATION	FUNCTIONAL/AC/DC	FUNCTIONAL/EXEC-TIME
DESCREPTION	DIAGRAM/TRUTH TABLE	SADT/HIPO/FLOW-CHART
LANGUAGE	BEHAVIOR/RT/LOGIC	HIGH LEVEL/ASSEMBLE/MACHINE CODE
HIERARCHY	CHIP/BLOCK/CELL	MAIN/SUBROUTINE/STATEMENT
CONNECTION	DATA/ADDR/CONTROL-PIN	ARGUMENT/COMMON
FLOW	DATA/CONTROL/STATE	DATA/CONTROL
COMPONENT	LOGIC-GATE/SSI/MSI/LSI	OPETATOR/SUBROUTINE
TEST	TEST-VECTOR(B/O/H)	TEST-DATA(DATA-STRUCTURE)
IMPLEMENTATION	LOGIC-CIRCUIT	CODING
EXECUTE	SIMULATOR (CONCURRENT)	COMPUTER(SEQUENTIAL)

TABLE 1　ハードウェア設計とソフトウェア設計の比較

レベルシミュレーションを行い、レイアウト時には、中身のないブロックとして配置配線を行なったのち、手書きレイアウトパターンをはめこむ。ここでベーシックセルにより構成されたブロックは、展開されて配線遅延が割り当てられ、スーパーマクロセルと結合されて、遅れ付きシュレーションが行なわれる。

今後は、ＲＴレベルのライブラリが整備されることで、回路検証の最終段階がＲＴレベルとなる。このように、設計者が関与する設計レベルを上位のみとし、以後の過程を自動化することが設計期間の短縮には特に重要である。

ミックスレベルシミュレータ　ＦＡＬ（Functional And Logic）[5][6][7] は、システム全体を一括して検証することができる。またfunction記述，状態記述を含むＲＴレベル記述，およびモジュールを構成要素とする構造記述によって、対象回路を階層的に扱うことが可

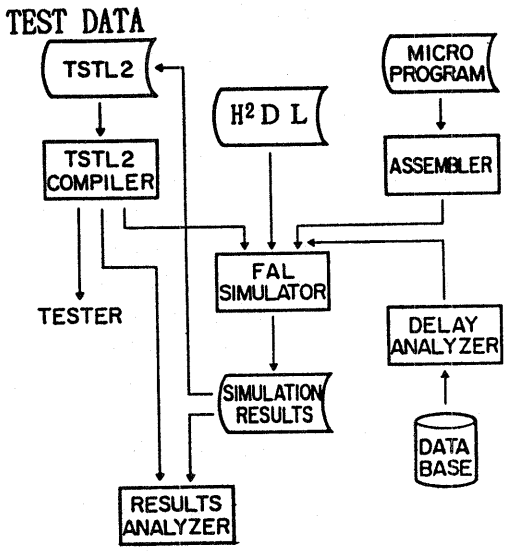


Fig.1　ＣＡＤシステムの構成

能である。[8]　これらに対応して、3通りのＨＤＬ（Ｈ２ＤＬ：Hierarchical Hardware Design Language [9]）が存在する。このうちＲＴレベル記述が重要であり、論理記述の代替として使用されている。ＦＡＬでは、文を単位として扱い並列に処理している。さらに、Vector型の信号における連結や、sub-vectorの切出し処理、条件文、条件式などの扱い等も可能なように拡張されている。

また、ＲＴレベルのモジュールを論理レベルの代替として使用している場合には、遅れの考慮が必要になるため、ＲＴレベルにも遅延素子を取り入れて整合性を取っている。同期型回路でも入力信号エッジからのタイミング情報を表わす必要があるため、この遅延素子を使用している。function記述は定義したプロセスの適用対象を限定する形式である。function記述内では、subroutine等を使用することで“機能”の階層性を実現している。この機能の階層性と、モジュールの階層性とは明確に分離されている。

２．２　ＨＤＬによる既存のＬＳＩのモデリング

既存のＬＳＩをＳＭＣライブラリーとして用いるためにＲＴレベル記述でのモデリングが必要となる。ＬＳＩのモデリングを行う際には、モデリングの手順（特に遅れの割り付け方法）、記述レベルの選択などが問題となる。

２．２．１　モデリングの概要

ＬＳＩのデータシート[10]から得られる情報としては、１）機能の概要，２）入出力ピンの機能，３）ブロック図およびシステム構成，４）基本動作の説明，５）状態遷移図と基本タイミングの説明，６）命令動作（各マシーン・サイクルの各ステートでの動作）の一覧表，７）ＡＣ特性およびタイミング図　である。すなわち、これらがＬＳＩの外部仕様書となるわけである。

このデータをもとにして、以下の手順にしたがってモデリングが進められる。

１）．ピン定義，ブロック図，システム構成からモデル内部で使用する部品の定義を行なう。
２）．状態遷移図をもとに、状態の定義，遷移条件を決める。
３）．制御条件と状態とを関連させて、制御を各状態に割り付ける。
４）．基本タイミングをもとにデータの流れを各状態内に割付ける。
５）．１）～４）をモデルの骨格として各命令動作を実現するために内部での部品の制御方法を決定し、詳細な設計を行なう。
６）．上述したＨＤＬをもとに、モデルの機能動作の確認を行った後に、ＡＣ特性とタイミング図をもとに遅れを割り付けていく。
７）．ＬＳＩ生産用のテストベクトルを使ってモデルの検証を行う。
８）．各ピンについて、ドライバビリティ，負荷容量等を含めた定義情報をライブラリに登録する。

以上のような手順で開発されたモデルをＳＭＣに使用する。このモデルではデータシートにより定義されているシステム構成要素のレベルまでの動作を見ることができる。このＳＭＣモデルは、配線の詳細な遅れも考慮した論理レベルの回路ブロックと組み合わせて、全体の動作確認を行うために使用する。このために、ＲＴレベルではありながらタイミングを詳細に実現している。遅れは内部とoutputのドライバー部とに割振り、outputの負荷による遅れを正しく見積もっている。

２．２．２　ミックスレベルシミュレータの記述レベルによる比較

先に述べたようにＦＡＬでは３つの記述レベルが使用できる。シミュレーションを高速に行うには、上位レベルの記述を用いたほうがより有効である。実際にどの程度高速になるかを調べるため以下の様な実験を行った。

ＲＴレベルと論理レベルとで、記述レベルによるシミュレーション速度の違いを比較するために、簡単なＣＰＵをＲＴレベルで記述し、論理合成により論理記述を得て両者のシミュレーションを行った。ＲＴレベルの記述は、単一ブロック構成のもの（ＳＣＰＵ１）と、３ブロック構成のもの（ＳＣＰＵ３）との２通りを用意した。論理レベルでは、vector型の信号を使用した場合（ＳＣＰＵＶ）と、１ビット型の信号を使用した場合（ＳＣＰＵＢ）の２通り用意し、合計４通り

Circuit name	Kind	Language level	CPU Time (sec)		events×bit
			Compile link	Set up execution	events×bit/sec
SCPU1	1 block	RT+function	0.3	0.5	3.2 k
			0.7	0.2	14.9 k
SCPU3	3 block	RT+function	0.3	0.6	6.9 k
			1.1	0.8	8.8 k
SCPUV	vector	logic +RT+function	1.4	0.9	114.6 k
			6.0	11.4	10.2 k
SCPUB	bit	logic+logic	1.9	2.2	118.9 k
			6.9	30.4	4.0 k

Table 2　回路記述によるＣＰＵ時間と実行速度の比較

の記述による比較を行った。(Table 2)　合成された論理回路はＧＡのベーシックセルを使用して、700ゲートの規模である。ここでベーシックセルは論理レベルの記述を使用している。

bit 型信号のみのＳＣＰＵＢでは、コンパイル，リンク，シミュレーション全てにわたって、ＳＣＰＵ１に比べて６倍，１０倍，１５０倍の計算時間が必要である。イベント処理速度は、vector型の処理[11]が大きくてbit 型の２～３倍の向上が見られる。ＲＴレベルでは、イベントの発生数が少ないためにシミュレーション時間は、ロジックレベルに比べて大巾に短縮されている。

この例では、同一の回路動作を実現する各記述に対して行ったものであるが、このシミュレーション結果からわかることは次のことである。

従来のシミュレータの性能の指標である単位時間当たりのイベント処理量は、記述レベルまで考慮すると、指標として用いるには不十分である。発生したイベント数の比が　1：37程度であり、イベント処理速度が1：3.7 であることから、1：137 程度のシミュレーション時間の短縮が見込まれる。

以上のように、logic レベルで記述する場合とＲＴレベルで記述する場合ではシミュレーションの実行時間が大きく異なってくることが解った。また同じＲＴレベルでもモデリングの方法によって実行速度に差がでてくる。従ってＳＭＣのライブラリーを作成する場合にそのモデリングを適切に行うことがいかに重要であるかが解る。

2．3　ライブラリ

ライブラリとしては、ＳＳＩ，ＭＳＩに対応するベーシックセルと、ＬＳＩに対応するＳＭＣの２通りがある。回路の記述は、論理レベルとＲＴレベルとを使用している。SMCの対象がＬＳＩ規模であるので、function記述（単純な入出力関係のみ）で書くことは困難である。function記述は、ＲＴレベルの中で、ＡＬＵやDecoder等に利用している。利用者からは論理レベルとＲＴレベルとは何等の区別もなく、単にライブラリ開発者の開発法の違いだけである。ピン情報等の外殻情報は他のレイアウト情報と一緒にデータベースに登録されている。これらの情報をもとに、レイアウト前では予想配線長による遅れ、レイアウト後では実配線長による遅れを計算しシミュレーションを行なう。ＳＩではＳＭＣ内のレイアウトは固定であるので、ライブラリ内にある内部遅延の値を使用する。ライブラリのＳＭＣの遅れに関する情報は、Fig.2に示すようにデータシートのＡＣ特性そのままのものである。以上の事からわかるように、ＳＭＣではpin にＩ／Ｏバッファを有するＬＳＩだけを想定していて、このバッファにより、ＳＭＣの内部と外部が適切に分離されていると仮定しているのである。この仮定は、ＳＩやＳＣでのマクロセルの使用法と矛盾せず、ＬＳＩでの配線容量や拡散容量等の付加の影響は、バッファに吸収されて、ＳＭＣ内部の動作の遅れに影響を与える事はない。

```
MODULE CELL-NAME ;
    ┆
    TERMINAL term-name  kup, kdn, C1 ;
    TERMINAL term-name  kup, kdn, C1 ;
        ┆
    ACSPEC term-name1,term-name2 ;
        SYMBOL delay#,delay-name,mode,max-dly,min-dly,typ-dly ;
        SYMBOL delay#,delay-name,mode,max-dly,min-dly,typ-dly ;
        ┆
    ENDACSPEC ;

    ACSPEC term-name1,term-name2 ;
        ┆
    ENDACSPEC ;
    ┆
ENDMODULE ;
```

Fig.2　SMCに必要なタイミング情報

ブロック	記述レベル	回路規模	コンパイル時間
TOP	STRUCTURE	−	0.88
Z80CPU,CGC	RT+FUNCTION	4 k gate	4.71
RAMDOM部	LOGIC	2 k gate	1.00
I/O BLOCK 部	LOGIC	0.5 k gate	0.23

リンク時間 …… 5.46 sec
シミュレーション時間 …… 0.07 sec/clock

Table 3　ミックスシミュレータＦＡＬの適用例

3．適用結果

ミックスシミュレータＦＡＬをＳＩに適用して、その可能性を検討した。（Table 3） 対象回路は Z80のmicro-processor （Z80CPU）， clock generator（Z80CGC）で、これらをＲＴレベルで記述し、2 K ゲートと 500ゲートの論理レベルからなる、4ブロック構成のＬＳＩとしてある。Z80CPUのＲＴレベル記述は自動発生を行なった部分が多い。Z80CPUのＡＬＵ部分はfunction記述を用いた。Z80CPU自身の動作確認は生産用のテストベクトルで行い、発見されたバクは coding miss 17 件，mis-understanding 15 件，mis-spec 3 件であった。Z80CPUに必要なデータ領域は、240 Kbyte であり、残り1Mbyte を論理ゲート回路に使用している。回路全体での実行速度は、15 MIPS のmachine で0.07 sec/clock である。全回路に460step のテストベクトルを入力して要した時間は70 secであった。この結果はリアルチップによるシミュレーションとほぼ同等の結果である。

Fig. 3 にその他の適用例から得られた結果を示す。(a) 図は、ゲート規模と記述量の関係を示したもので、ほぼ一次式で近似できることがわかる。(b) 図では記述の仕方による実行時間の比較を行なっている。結果は大きく分散しており記述の善し悪しでシミュレーション時間が変化することを示している。(c) 図からわかるように、回路によって、event ×bit ／sec の値は一定でない。これはＨＤＬ命令の実行や、vector型信号の処理の影響と考えられる。したがって、event ×bit ／sec の値はシミュレータの性能を測るための有効な指標とならない。

したがって、ソフトウェアと同様に同一の記述言語を使用しても、記述の仕方の善し悪しによって“ＨＤＬプログラム”の実行速度は大きく異なる。この速度低下の主要因は（1）トリガー記述を多用すること、（2）条件文を多用すること、等である。すなわち、無駄な文の評価やイベントの発生の評価が多発してしまうためである。これを防ぐ為には、文を複文化して複文全体にＡＴを使用するか状態記述を行なう。条件文は入れ子にする。 TERMINAL はなるべく用いないようにする、などの工夫が必要である。

4．まとめ

behaviorレベル，ＲＴレベル，structure レベルのＨＤＬを入力とするミックスレベルシミュレータＦＡＬを開発しＡＳＩＣ開発に適用する場合について検討を行った。

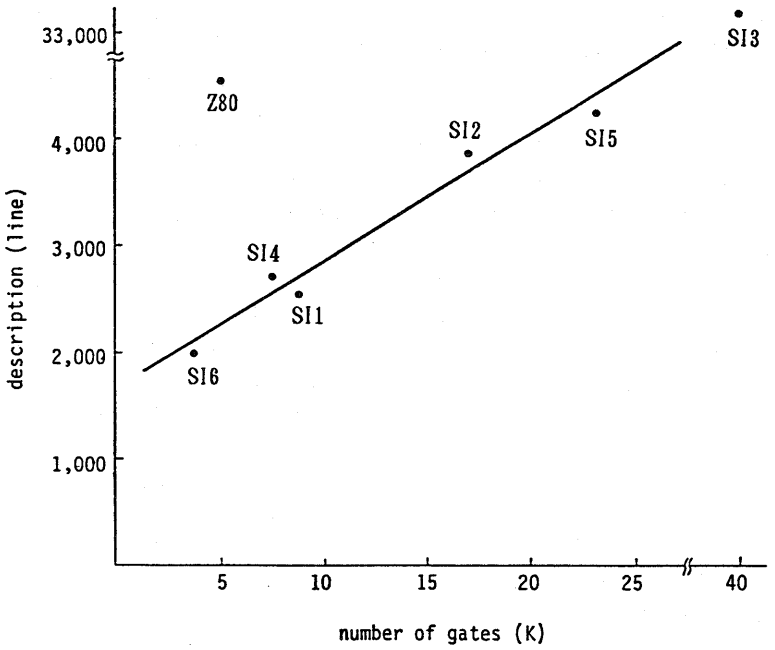
ミックスレベルシミュレータの使用目的としては、（1）マイクロ命令も含めた機能検証と、（2）ＲＴレベルでの論理検証とにわけられる。ＲＴレベルの書き方は、条件文や状態記述など多様である。書き方により機能中心或いは構造中心となるため、両目的に使用することが容易であるが、その記述に従って、シミュレーション速度も4～5倍の範囲で変化する。

ＲＴ記述を論理合成にかけて得られた論理回路記述と、もとのＲＴ記述とで、シミュレーション時間を比較して１３７倍の比を得た。また、それらの結果から、イベント処理速度はシミュレータの性能比較のための指標としては不十分であることがわかった。

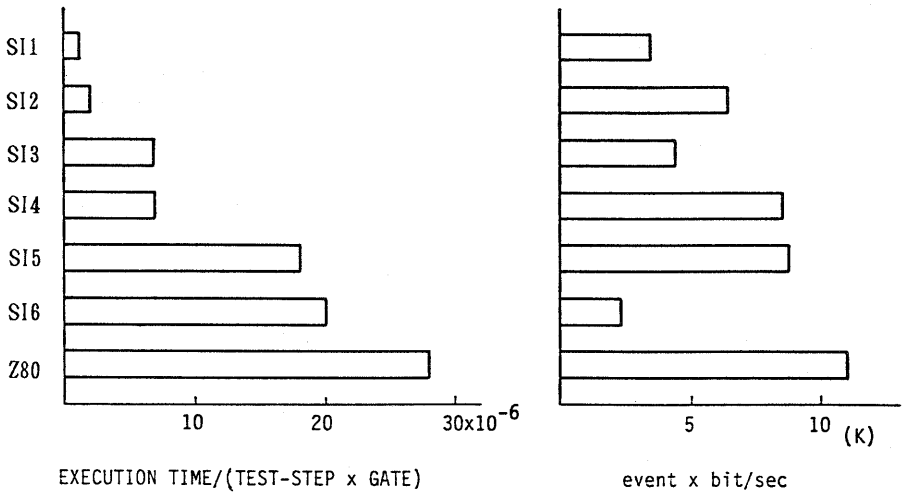
ＡＳＩＣ開発に必要なライブラリ作成技術を検証するため、Ｚ８０ファミリのモデルをデータシートをもとに開発した。モデルの記述レベルを標準化するためにデータシート中のブロックダイヤグラム程度の内部構造を持つようにしている。このモデルに、タイミング情報を取り入れるために、ＲＴレベルに遅れ素子を導入している。モデルの完全さを保障するために生産用のテストベクトルを使用している。

このように整備されたライブセラリを用いて、ＦＡＬの会話型デバッグ機能によりシミュレーションを行うことで、ＡＳＩＣ開発においてその開発期間の短縮が可能となる。特に、実現されるＬＳＩの論理回路を意識したＲＴレベル記述を行うことにより、機能検証から論理設計検証，タイミング設計検証への移行を迅速に行うことができる。また、機能検証時のシミュレーション結果出力をテストデータと同じ形式とすることによって、論理検証時テストベクトル作成のため時間を省略することが可能となる。

以上のように、ミックスレベルシミョレータＦＡＬは、ＡＳＩＣ開発を行う上で開発期間を短縮するために極めて有効であると言える。



(a) ゲート数とＨＤＬ記述量との関係



(b) ゲート換算でのシミュレーション時間

(c) イベント処理速度

Fig.3　回路の違いによる諸量の比較

参考文献

[1] K.Nagao, et al, "Super Integration", IEEE 1985 Custom Integrated Circuit Conference pp.267-271

[2] 立石，新田　第35回情報処理学会全国大会（発表予定）

[3] M.Miyata, et al, "Automatic Logic Synthesis from H^2DL Description", Proc.of ISCAS 85 pp.643,646

[4] 三橋，他，"カスタムＶＬＳＩのための統合レイアウトシステム"　電子通信学会回路とシステム研究会 CAS85-140 pp.643-646

[5] 市村，他，"ミックスシミュレータＦＡＬシステム"　第30回情報処理学会全国大会　pp.1945-1946

[6] 上田，他，"ミックスシミュレータＦＡＬのコンパイラ"　第31回情報処理学会全国大会 pp.1575-1576

[7] 相原，他，"インターフェースシミュレータの開発"　第33回情報処理学会全国大会　pp.2159-2160

[8] 武井，他，"ミックスシミュレータＦＡＬの階層処理"　第31回情報処理学会全国大会　pp.1577-1578

[9] 西尾，他，"階層的ハードウェア設計言語H^2ＤＬの言語仕様"　情報処理学会設計自動化研究会　22-2(1984.7.17)

[10] （株）東芝　集積回路事業部，"８ビットマイクロコンピュータTLCS-48,Z80,85データブック" JED2AA

[11] 武井，他，"機能論理シミュレータＦＡＬを核とした設計検証システム"　電子通信学会回路とシステム研究会 CAS86-184 pp.79-86

[12] 渡辺，他，"機能シミュレータＦＡＬにおける中断点処理"　第29回情報処理学会全国大会 pp.1721-1722