入力制約監視機能をもつ会話型シミュレーション・システム ISS†

安浦寛人†† 蚊野浩†† 大井康††
木村晋二†† 石浦菜岐佐†† 矢島脩三††

大規模な論理システムを高信頼度で設計するためには一貫した設計手法が必要である．設計手法の中で回路を機能モジュールに分割して階層的に設計を進めていく構造化論理設計は一般的でかつ重要である．本稿で報告するシステムは構造化論理設計を支援するハードウェア設計言語 SHDL とデータ管理法をベースとしており，検証ツールとして会話型論理シミュレータ IS を用いる．構造化論理設計のための CAD システムでは検証期間の短縮が重要である．IS では会話機能によりシミュレーションの実行に介入し，論理の追跡を容易にしている．さらに機能モジュールの仕様の一部として記述された入力に対する制約条件をシミュレーション中に監視することにより，設計の誤りを半ば自動的に発見することを可能とした．IS の会話機能，入力制約監視機能により検証期間の短縮が可能である．SHDL はハードウェア設計言語 DDL，およびプログラム言語 PL/I の特徴を取り入れることでアルゴリズムレベルからゲートレベルまでの広範囲の記述が可能である．また設計データは機能モジュール単位に管理し，構造化論理設計に対処する．本文では論理設計・検証と CAD システムについて論じた後，システムの構成，IS の機能と検証法について述べる．

1. はじめに

集積回路技術の目覚しい進歩は大規模でより高度な機能をもつ理論システムの実現を可能にした．しかし大規模な回路では回路全体を一括して設計を進めていくことは困難である．このため回路を機能モジュールに分割し，機能モジュール単位に設計を進めていく構造化設計の手法が必要となる[1]．

構造化論理設計では各詳細化の段階で設計と検証が繰り返されるため，全システムの設計期間の短縮のためには各段階での設計・検証期間の短縮が重要である[2]．これに対して従来から用いられている論理シミュレータなどの CAD (Computer Aided Design) ツールの多くはバッチ処理方式であるためターンアラウンドが遅く，しかも検証のための論理の追跡をほとんど人手で行うため設計の誤りが容易に発見できないなど必ずしも構造化設計に適しているとはいえない．

そこで，われわれは設計検証を容易にするために入力制約監視機能をもつ会話型論理シミュレータ IS(Interactive Simulator) を中心とする設計・検証支援システム ISS (Interactive Simulation System) を開発した[3]．検証ツールである多レベル論理シミュレータ IS は中断条件の設定，入力パタンの動的変更などの会話機能，および仕様の一部として与えられた入力制約を監視する機能により検証能力を向上させており，設計・検証のサイクルを短縮できる．さらに構造化論理設計を支援するためにハードウェア設計言語 SHDL (Structured Hardware Design Language) を設計した．SHDL では設計初期から回路の機能の記述が可能な手続き的な機能記述や回路のサイズをパラメータとして与えるといったアルゴリズム的な記述，さらに，検証に利用する入力制約の記述などが可能である．また設計データは設計の単位となる機能モジュールごとに管理して構造化設計に対処する．

本論文では構造化論理設計と CAD ツール，とくに検証ツールとしての論理シミュレータについて考察した後，3章で ISS の概要を述べる．4章では入力制約監視機能をもつ会話型論理シミュレータ IS について説明し，5章において ISS を用いた検証例を示す．最後に若干の性能評価を行う．

2. 論理設計・検証と CAD ツール

2.1 構造化論理設計のための CAD システム

構造化論理設計とは一つの機能モジュールとして定義された回路を，互いに独立性の強い下位の機能モジュールに分割することにより設計を進めていく手法である．分割を進めていった機能モジュールがゲート等の基本素子，またはすでに設計済みのマクロ的な機

† ISS: An Interactive Simulation System with Input Constraints Monitoring Facility by HIROTO YASUURA, HIROSHI KANO, YASUSHI OOI, SHINJI KIMURA, NAGISA ISHIURA and SHUZO YAJIMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

能モジュールで表現されたとき設計は終了する．このような構造化論理設計を支援するCADシステムには次のことが要求される．

（1） 構造化された回路を記述する言語をもつ[4]．

（2） 構造化された回路を検証しやすいツールをもつ．

（3） 回路の構造化が容易に行える．

われわれは，(1)に関しては，設計初期からの高レベルの機能記述が可能で回路の階層性も表現できる構造化ハードウェア設計言語SHDLを新たに設計した．(2)に関しては構造化され，異なった詳細化の段階にある回路がもとの回路と同じ機能であることを検証するために会話型の多レベルシミュレーションを実現した．さらに，構造化された回路の仕様の一部を入力制約として記述し，シミュレーション中に監視することにより半ば自動的な検証を可能にした．さらに(3)に関しては，規則的な構成法が知られている回路に対する回路の大きさをパラメータとした回路の構造の記述手法や，構造化論理設計に適したデータ管理方式を採用している．

2.2 入力制約と会話型シミュレータ

設計者が仕様で与える機能モジュールのもつべき機能は，多くの場合，すべての入力パタンに対して定義されたものではない．ある一定の制約を満たすパタンに対してのみ正しく機能すればよいのであって，このことを意識して設計は進められる．一方，検証は構造化された各機能モジュールが制約を満たすパタンを入力とし，正しく機能していることを確かめることによって行われる．われわれは，回路が正常に動作するための必要条件である入力パタンに対する制約（入力制約）の記述をSHDLに導入し，これを設計検証に有効に利用する手法を提案する．

論理回路の設計検証の方法としては論理シミュレータを用いる方法が現在のところもっとも有効であると考えられている．シミュレータの有効性はいかに早く設計の誤りを発見できるかにかかっているといえるが，従来からのバッチ方式のシミュレータを用いた検証は完全にシミュレーションが終了した後に，人手で論を理追跡することにより行われるため手間がかかり，無駄なシミュレーションも多い．これに対し，シミュレーションの実行とともに設計者が各部の信号を仕様と見比べることができれば論理の追跡を行わなくてすむ．これを可能にするのが会話型シミュレータである．またすべての信号を設計者が検査するのは容易でないため，あらかじめ機能モジュール単位に入力制約を記述しておけば，この制約をシミュレータが監視することにより，制約違反の信号を供給したモジュールに誤りがあることがわかる．これが入力制約監視機能による自動検証の考え方である．これらの機能によりバッチ処理方式の欠点を補い，設計期間の短縮が実現できる．

3. 会話型シミュレーション・システムISS

3.1 構造化ハードウェア設計言語SHDL

従来から用いられているハードウェア記述言語の多くはRTレベル[5]や命令セットレベル[6]のようにシステムのある抽象化のレベルの記述をねらったもので，システムの初期の設計から論理図のレベルまでを必ずしもすべて記述，設計できるものではなかった．

SHDLはISSにおいて論理システムを記述するために開発したもので，DDL[5]およびプログラム言語PL/Iの長所を取り入れることで，ゲートレベルから機能／アルゴリズムレベルにいたる広い範囲で論理設計の支援を可能にしている．SHDLは次のような特色をもつ．

（1） 構造化論理設計のための階層的な記述ができる（STR記述）．

（2） オートマトン的な機能記述（FNA記述）と手続き的な機能記述（FNP記述）があり，設計初期からの幅広い記述が可能である．

（3） 入力制約記述が導入されており，設計検証に利用できる．

（4） パラメータ付きモジュール等のアルゴリズム的な記述が可能．

STR記述はモジュールを内部モジュールの結合状態で記述したもの．FNA記述，FNP記述はそれぞれ，DDL，PL/Iに若干の変更を加えたものである．

SHDLでは入力制約をモジュールの記述に付加することができる．SHDLに導入した入力制約には次の3種類がある．

（1） 値制約：端子の取りうる値に対する条件．

（2） 安定制約：クロック入力に対して保証されるべき信号の安定時間，セットアップ・ホールド時間を記述する．

（3） パルス制約：信号の0/1パルス幅の指定を行う．

図1に入力制約記述を含む回路の記述例を示す．*n*-bit加算器のように規則的な構成法が知られている回

```
<<MODULE>>        SN74111.FNN ;
 <<INPUT>>        CLEAR, CLOCK, J, K, PRESET ;
 <<OUTPUT>>       Q, QL ;
 <<INPUT CONSTRAINTS>>
                  VALUE  ~(CLEAR = 0 & PRESET = 0)   ,
                  PULSE              PRESET = (25,  ) ,
                  PULSE              CLEAR  = (25,  ) ,
                  PULSE              CLOCK  = (25,25) ,
                  STABLE (J, K) : U(CLOCK) = ( 0,30) ;
 <<DELAY>>        (CLEAR, PRESET) -> (Q,QL) = 21,
                  CLOCK           -> (Q,QL) = 20;
 <<TRIGGER>>      UD(CLOCK) -> (J,K) ;
 <<FUNCNAME>>     FJKMSCP ;
<<MEND>>
```

図 1　入力制約を含む SHDL 記述
Fig. 1　SHDL description with input constraints.

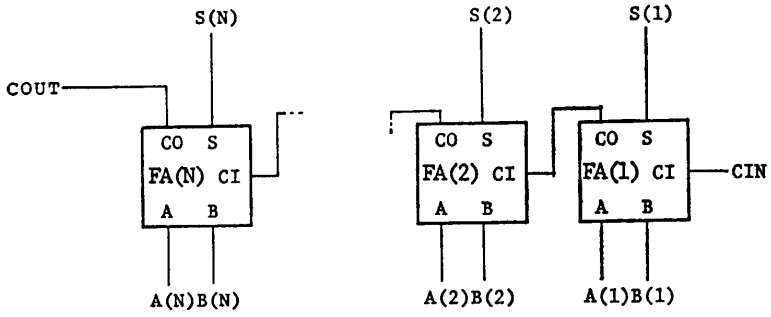

図 2 (a)　*N*-bit 順次桁上げ加算器
Fig. 2 (a)　*N*-bit ripple carry adder.

```
<<MODULE>>     ADDER(N).STR;
  <<INPUT>>    A(N), B(N), CIN ;
  <<OUTPUT>>   S(N), COUT ;
  <<STRUCTURE>>
    <<BLOCK>> FA(N)=FULLADD.FNP;
    <<CONNECTION>>
              %FOR I=1 TO N;
                   FA(I).A/A(I),
                   FA(I).B/B(I),
                   FA(I).S/S(I)
              %END;
              FA(1).CI/CIN,
              FA(N).CO/COUT
              %FOR I=1 TO N-1;
                   FA(I).CO/FA(I+1).CI
              %END;
<<MEND>>
```

図 2 (b)　*N*-bit 順次桁上げ加算器の STR 記述
Fig. 2 (b)　STR description of *N*-bit ripple carry adder.

路では回路の大きさをパラメータとして回路を記述することが有効である．われわれは，パラメータ付きモジュール記述を SHDL に導入し，同じ構成法に基づく回路を簡潔に記述できるようにした．図 2 に順次桁上げ加算器の記述を示すが，(b)の SHDL 記述も(a)の図による記述と同様，たんに順次桁上げ加算器であることを示すだけでなく，順次桁上げのアルゴリズム自身をも記述していると考えられる．このようなハードウェアのアルゴリズム的な記述は検証・解析が容易であり，大規模な論理回路の設計には不可欠であろう．

3.2　システム構成

CAD システムでは，利用者とシステムとのインタフェースの設定が重要なポイントである．われわれは，システム全体を TSS の下で使える会話型に構成することにより，利用者が柔軟にシステムを利用し，設計サイクルの短縮化を図れるようにした．

会話型シミュレーション・システム ISS のシステム構成を図 3 に示す．ISS 管理部は端末からのコマンドの解析，実行，各プログラムの起動を行う．エディタは行編集機能をもち，回路データの入力，編集を行う．トランスレータは SHDL による記述を内部データに変換する．リンカはトランスレータの出力を展開，結合してシミュレーション・モデルを作成する．STR 記述，FNA 記述，FNP 記述が混在する機能モジュールをも展開，結合できるので多レベルシミュレーションが可能である．IS については 4 章で述べる．

また ISS は論理図編集システム LODE (Logic Diagram Editing System)[7] と結合されており，論理図を用いて設計・検証を行うことにより ISS とのよりよいユーザインタフェースを得ることができる．

また，とくに構造化論理設計を支援することを目的として，構造化された回路を設計，記述するための言語としての SHDL を開発するとともに，機能モジュール単位の設計・

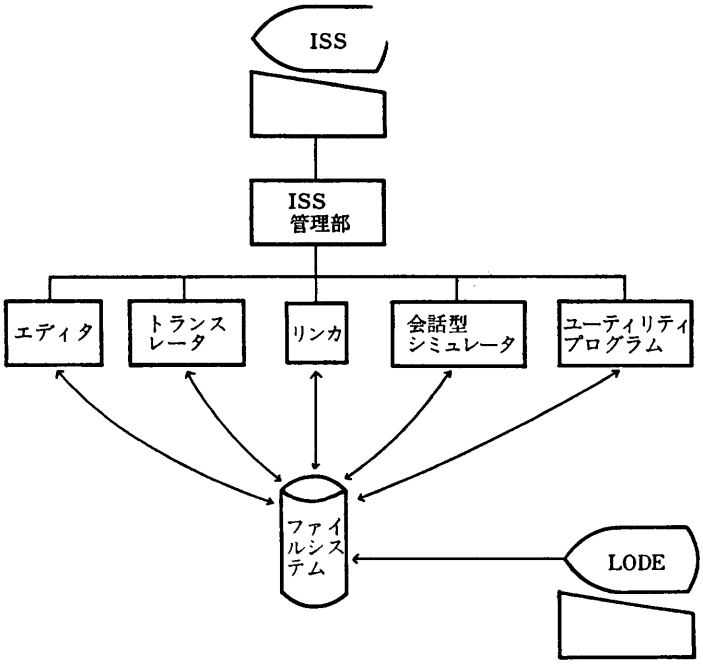

図 3 ISS のシステム構成
Fig. 3 Configuration of ISS.

検証を容易にするため，設計データを機能モジュール単位に管理する方針をとった．このデータ管理方式と各種の検索機能を用意することにより，設計者が必要とする各モジュールのデータとその相互の関係を容易に把握できるようにしている．

3.3 設計データの管理

論理設計のための CAD システムで扱う設計データは回路の仕様，機能のようなテキストデータ，結線関係のようなネットワーク状の構造をもつもの，テストパタン，シミュレーション結果などのビットストリングと全体としては明確な構造をもたないため，詳細なモデル化を行って管理することはむずかしい．しかし，構造化設計の手法を用いて階層的に設計を進める場合，機能モジュールを単位として設計を行うため，設計データの管理においても機能モジュールを一つの実体とし，そのなかの種類ごとの設計データ全体を一つの項目と見なすことにより，巨視的にはすっきりしたデータモデルを得ることができる．機能モジュールごとに設計データを集めたものをモジュールファイルと呼ぶ．モジュールファイルを単位として設計データを管理することにより，構造化論理設計に適したデータ管理が行える．

ISS におけるモジュールファイルの内容はモジュールの STR 記述，FNA 記述，FNP 記述，およびトランスレート結果，リンク結果，テストパタンなどである．

4. 会話型論理シミュレータ IS

4.1 シミュレーション方式

IS は以前に開発した多レベル論理シミュレータ SIM/D[8] に会話機能等を付加したものである．

IS のシミュレーション方式は基本的にはテーブルドリブンイベント方式であるが，クロックなどの制御信号は SHDL による記述のなかでトリガ端子宣言することができ，これを利用して，無駄なイベントを減らし，シミュレーションの高速化を図っている[3),8)]．機能モジュールの評価方法は FNA モジュールはインタプリタ方式で，FNP モジュールは PL/I ステートメントを翻訳した後，OS のダイナミックローディング機能を利用してシミュレータに組み込み，評価する．このような方式により回路のシミュレーション・モデルを簡単に変更でき，多レベルシミュレーションの実現が容易である．

4.2 会話機能

論理設計が，設計とその検証の繰返しである以上，検証期間の短縮はそのまま設計期間の短縮につながる[9]．論理シミュレーションでは利用者が実行に介入

AT	中断時刻の設定
ATCAT	設定した中断時刻の表示
EDITWAVE (EWAVE)	入力パタンの変更
END (E)	シミュレーションの終了
GO	シミュレーションの再開
LIST (L)	端子の値の表示
OFF	入力制約の監視の解除
OFFAT (OFFA)	中断時刻の解除
OFFSTEP (OFFS)	ステップ条件の解除
OFFWHEN (OFFWHN)	中断条件の解除
ON	入力制約の監視を開始
RUN (R)	シミュレーションの再開
STATUS (STA)	ブロックのメモリの値の表示
STEP (ST)	ステップ条件の設定
WHEN (WH)	中断条件の設定
WHENCAT (WHNCAT)	中断条件名の表示
WHENLIST (WHNL)	中断条件の表示
BTLIST	ブロック名の表示
ITLIST	入力端子名の表示
OTLIST	出力端子名の表示

図 4 IS サブコマンドの機能
Fig. 4 IS commands.

```
ISSSIM SHIFT_REGISTER.STR A001 TIME(10)

 ISS  INTERACTIVE SIMULATOR VER 1.1 INVOKED

SIMULATE:ON *
SIMULATE:AT 500
SIMULATE:GO
<INPUT CONSTRAINTS> TIME<  161:  170>   MODULE=SHIFT_REGISTER
000100             PULSE  (CLEAR,CLOCK)=(20,20)
SIMULATE:LIST (CLEAR CLOCK) (1:)
CLEAR <    1:  160>
11111 11111 11110 1
CLOCK <    1:  160>
00000 11111 00000 1
SIMULATE:EDITWAVE
LINENAME:CLEAR
FROM(TIME):170
ENTER VALUE
:000011111;
LINENAME:
SIMULATE:GO
<AT> TIME<  491:  500>
SIMULATE:LIST (CLEAR CLOCK R Q<2>) (1:)
CLEAR <    1:  490>
11111 11111 11110 10000 11111 11111 11111 11111 11111 1111
CLOCK <    1:  490>
00000 11111 00000 11111 00000 11111 00000 11111 00000 1111
R <    1:  490>
11111 11111 11111 11111 11111 11110 00000 00001 11111 1111
Q(1) <    1:  490>
XXXXX XXXXX XXXXX X0000 00000 00111 11111 11000 00000 0011
Q(2) <    1:  490>
XXXXX XXXXX XXXXX X0000 00000 00111 11111 11111 11111 1100
SIMULATE:WHEN SHIFT_END (Q(8)=U)
SIMULATE:GO
<WHEN> (SHIFT_END) TIME=< 1181: 1190>  **EXACTLY**
SIMULATE:LIST (Q(4) Q(5) Q(6) Q(7) Q(8)) (700:)
Q(4) <  691:  1180>
00000 00011 11111 11111 11111 11111 11111 11111 11111 1111
Q(5) <  691:  1180>
11111 11100 00000 00011 11111 11111 11111 11111 11111 1111
Q(6) <  691:  1180>
11111 11111 11111 11100 00000 00011 11111 11111 11111 1111
Q(7) <  691:  1180>
11111 11111 11111 11111 11111 11100 00000 00011 11111 1111
Q(8) <  691:  1180>
11111 11111 11111 11111 11111 11111 11111 11100 00000 0001
SIMULATE:
```

図 5　シミュレーション例
Fig. 5　Example of simulation.

し，無駄なシミュレーションを減らし，また論理の追跡を容易にする機能が必要である．そのためにはシミュレータは会話型であることが本質的であると考え，次のような会話機能をもたせた．

i）中断条件の設定：信号線の値やシミュレーション時間に関する条件が成立するとシミュレーションの実行を中断する．この中断機能により，利用者はシミュレーションの結果に応じて実行に介入でき，以下の機能を用いて効率よくシミュレーションを進めていく．

ii）入力制約の監視／解除：シミュレーション中に入力制約を監視できる．監視中に入力制約に違反する入力がいずれかのモジュールに加えられた場合には，シミュレーションの実行を中断する．

iii）シミュレーション結果の表示：指定された時刻の端子，レジスタ，メモリの値を表示できる．

iv）入力用テストパタンの変更：シミュレーションの実行途中で，テストパタンを動的に変更することができる．シミュレーション結果に適応してシミュレーションの実行を制御でき，無駄なシミュレーションを行わないですむ．

v）シミュレーションの開始／再開：中断されたシミュレーションを中断時刻からだけでなく，過去の時刻に遡って再開することができる．

IS で利用可能なコマンドとその機能を**図 4** に示す．

4.3　入力制約の監視機能

従来の論理シミュレータを用いた検証は，シミュレーション結果より誤りがあることを知り，論理を追跡することにより誤りを同定する方法で行われてい

た.

ISS においては，入力制約が記述されたサブモジュールから構成された機能モジュールについては，おのおののサブモジュールについて論理の追跡をしなくとも入力制約条件をシミュレーション中に監視することにより半ば自動的に設計の誤りを同定することが可能である．この機能を実現するためにシミュレータ本体では各端子が制約条件を満たしているかどうかの判定を各シミュレーション・ステップごとに行っている．

4.4 IS の利用

図5に IS の利用例を示す．この例ではシフトレジスタに対する入力制約の監視，中断条件の設定，入力パタンの変更を行っている．

5. ISS における設計検証

5.1 IS を用いた検証

構造化された回路は，各機能モジュール単位にほぼ独立に設計される．この回路が全体として正しく動作することを検証するには，設計の詳細化の異なる機能モジュールを結合してシミュレーションしなければならないことが多い．このため，IS では多レベルシミュレーションを実現している．

また，論理の追跡を容易にし，無駄なシミュレーションを減らすには，中断条件の設定，入力パタンの変更が有効である．

5.2 入力制約を用いた検証

論理回路は組合せ回路とメモリ（レジスタ）から構成される．組合せ回路の検証は比較的容易であるため，システム全体の検証においては組合せ回路とメモリ間のタイミング検証を効率よく行う必要がある．メモリ素子と組合せ回路により構成された順序回路を従来は次の方法で検証していた．

① 組合せ回路の論理を検証した後，遅延評価プログラムで組合せ回路の各パスごとに遅延を評価してタイミング検証を行う．

② タイミングをも扱えるシミュレータでシミュレーション後，各メモリが正しくセットされているかどうかを調べる．

② の方法で行う場合には設計者がシミュレータの精度について十分の知識をもちかつクリティカルなタイミングをも正確に扱えるシミュレータが必要とされるがこれは困難な問題である．ISS においてはメモリが満たすべきタイミング条件を安定制約，パルス制約として記述しておき，各部の制約をシミュレーション中に監視することで組合せ回路の論理とタイミングを同時に検証できる．

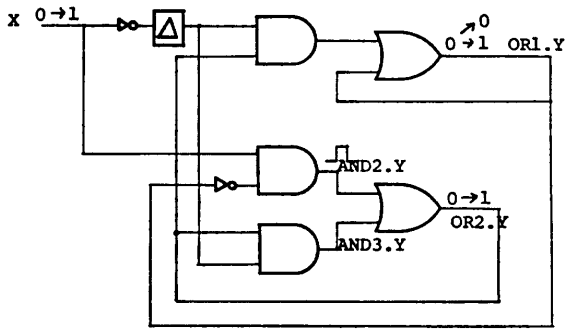

図 6 (a) 基本ハザードの可能性のある非同期回路
Fig. 6 (a) Asynchronous circuit with a possibility of essential hazard.

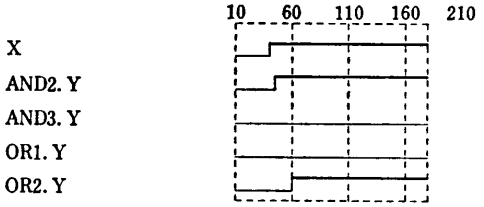

図 6 (b) 正常な動作
Fig. 6 (b) Correct operation.

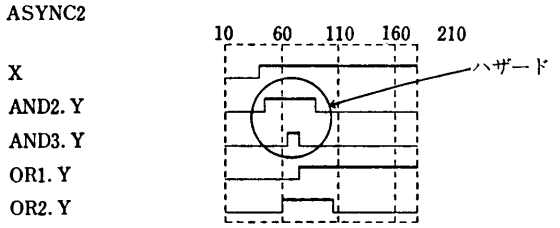

図 6 (c) ハザードによる誤動作
Fig. 6 (c) Incorrect operation by essential hazard.

非同期の順序回路では論理仕様の誤りにより**図 6**（a）の回路のように素子，配線の伝播遅延の差による基本ハザードの可能性が避けられない場合がある（図 6（b），（c））．この種のハザードは従来の長大なシミュレーション結果よりハザードを検出する‘タイミングベリファイア’[10] などのプログラムにより検出していた．ISS ではパルス制約の利用でハザードの検出を実行時に行える．

このほかにも入力制約が有効であると考えられる問題には次のものがある．

① フリップフロップ，メモリ素子に対するセットアップ／ホールド時間のチェック

② クロック線上のスパイク，ハザードの検出

表 1 IS の性能
Table 1 Performance of IS.

		FNP (Procedure)	STR (Structure)	FNA (Automaton)	Gates
Parity generator	Pattern 1	3.0	5.1	97.5	13
	Pattern 2	6.9	13.4	352.9	
	Pattern 3	2.3	6.7	351.2	
Adder	8 bit	5.1	23.9		48
	16 bit	6.9	39.7		96
	32 bit	10.9	71.7		192
Multiplier	8 bit	3.9	44.8		400
	16 bit	5.3	407.0		1,696

CPU 時間（単位：秒）HITAC M-240 H 使用

③ インタフェース回路のタイミングチェック
④ SR ラッチの不正使用の検出
⑤ コントロール線上の矛盾した入力組のチェック
⑥ バス上の競合の検査

5.3 パラメータ付きモジュールの検証

規則的な構成法に従う回路をシミュレーションで検証する場合，入力パタン数が回路のサイズの増加につれて急激に増加するものがある．たとえば 4 bit 乗算器ではパタン数が 256 であるのに対し 16 bit 乗算器では 2^{32} となる．このような回路はパラメータ付きモジュールとして記述し，小さなパラメータ値についてアルゴリズムの正しさが確認されれば，パラメータ値の大きな場合についても正しさが保証される．この手法は論理シミュレータを用いた大規模回路の検証法として有効である．

6. 性能評価

加算器，乗算器，および順序回路であるパリティ生成回路についてシミュレーションに要した CPU 時間を**表 1** に示す．この表からわかるように FNP 記述のシミュレーションは FNA 記述よりも 30〜100 倍高速であり，加算器，乗算器についてはビット幅の増大に対してもさほど影響されない．これは FNA 記述のシミュレーションが中間コードをインタプリタ方式で解釈実行しているのに対し，FNP 記述ではコンパイルされた機械語ルーチンを実行しているためである．FNA 記述，FNP 記述では遅延の指定はモジュールの入出力のパスにしかできないため，内部ブロックに対して遅延が計算される STR 記述に比べてシミュレーションの精度は落ちる．

7. おわりに

本稿では入力制約監視機能をもつ会話型論理シミュレータの論理回路の設計検証における有効性，および構造化設計を支援するシステムの構成について述べた．論理回路の設計手法とそのためのシステム構成，ハードウェア設計言語については，今後，検討の必要がある．

問題点としては，会話型システムのためのオーバヘッドからくるシミュレーション可能な回路規模の制限，シミュレーション速度の低下があるが，今後の計算機システムの改善が待たれる．また入力制約としてどのようなものが検証の上で適当であるかについては理論的な裏づけも必要であろう．

ISS は現在，京都大学情報工学教室の HITAC M-240 H，および京都大学大型計算機センターの FACOM M-382 上で稼動しており，論理回路の設計・検証に用いられている．プログラムは PL/I で書かれており，約 14,000 ステップである．

謝辞 貴重なご助言，ご討論をいただいた上林弥彦助教授，平石裕実博士はじめ矢島研究室の諸氏に深謝いたします．また IS の前身である SIM/D を開発された槌田義之氏，IS の初期の開発を進められた酒井丈嗣氏，および SHDL の設計に尽力された小野善続氏に感謝いたします．なお本研究は一部文部省科学研究費による．

参考文献

1) Caplender, H.D. and Janku, J.A.; Top-Down Approach to LSI System Design, *Comput. Des.*, Vol. 13, No. 8, pp. 143-148 (1974).

2) 川戸，斉藤，上原：ハードウェア設計言語 DDL による計算機設計支援システム，情報処理学会論文誌，Vol. 21, No. 1, pp. 67-75 (1980).
3) Sakai, T. et al.: An Interactive Simulation System for Structured Logic Design—ISS, Proc. 19th Design Automation Conf., pp. 747-755 (1982).
4) vanCleempt, W. M.: An Hierarchical Language for the Structured Description of Digital System, Proc. 14th Design Automation Conf., pp. 377-385 (1977).
5) Duke, J. R. and Dietmeyer, D. L.: A Digital System Design Language (DDL), *IEEE Trans. Comput.*, Vol. C-17, No. 9, pp. 850-861 (1968).
6) Barbacci, M. R.: Instruction Set Processor Specifications (ISPS): The Notation and Its Applications, *IEEE Trans. Comput.*, Vol. C-30, No. 1, pp. 24-40 (1981).
7) Hiraishi, H. and Yajima, S.: Logic Diagram Editing for Interactive Logic Design, Proc. the Working Conference on Picture Engineering, The 15th IBM Computer Science Symposium, Software Engineering Series, No. 3, pp. 145-169 (1981).
8) 稲垣，槌田，酒井，矢島：LSI を含む階層的論理システムのための論理シミュレータ SIM/D，情報処理学会論文誌，Vol. 21, No. 4, pp. 332-339 (1980).
9) 大野，森田，小高，宮本，三善，鬼塚：超大型計算機 HITAC M-200 H の論理シミュレーション，情報処理学会論文誌，Vol. 21, No. 5, pp. 354-365 (1980).
10) 荻原，松下，村井：順序回路のタイミング・ベリファイヤ (SIMCHK)，情報処理学会電子装置設計技術研究会 10-3 (1981).

(昭和 58 年 6 月 13 日受付)
(昭和 58 年 9 月 13 日採録)