システムＬＳＩ設計技術　97－4
（２０００．９．７）

IP再利用を促進するインタフェース記述言語 O_wL

荒 宏視　　鈴木 敬　　矢野 和男
(株) 日立製作所 中央研究所
E-mail: {k-ara, ksuzuki, k_yano}@crl.hitachi.co.jp

概要

LSIの開発工数を抑えるためには設計財産(IP)の再利用が不可欠である．しかし現状ではIPの利用者に仕様が正確に伝わらない事や、IP利用時に必要なインタフェース検証や接続モジュール設計工数が大きいという課題がある。我々はこれらの課題の解決を目的に、IPのインタフェース記述言語 O_wL を開発している。本稿ではIPの仕様を6種に分類し、それらの関係と、仕様伝達時に生じる8種の人為的バグを明らかにする。またその分析結果から、正確な仕様伝達を可能にする手法 O_wL-*Messenger* とそれを構成する2つのツールを提案する。この手法により、従来は人に依存していた仕様伝達時の人為的バグを体系的に検出できる見通しを得た。

IP Reuse Methodology based on Interface Description Language O_wL

Koji ARA　　Kei SUZUKI　　Kazuo YANO
Central Research Laboratory, Hitachi Ltd., Tokyo, Japan
E-mail: {k-ara, ksuzuki, k_yano}@crl.hitachi.co.jp

abstract

An IP(intellectual property)-based design methodology is needed to shorten the time-to-market period. However, users of IP can not handle it without huge effort because the specification of IP is ambiguous and they must prepare many modules by themselves to verify or to connect the IP module. We developed an interface description language O_wL to solve such problems. Here, we categorize the specification of IP into six kinds and analyze the relation among them. And we make it clear that there are eight kinds of bugs caused by designers and users. We propose a new specification methodology called O_wL-*Messenger* which enables the designer and the user to share the understanding of the specification, and show two tools consisting the methodology. A result of our estimation shows that O_wL-*Messenger* is able to detect those bugs systematically.

1 はじめに

大規模なシステムLSIを短期間で開発するためには、既存の設計財産(Intellectual Property、以下IP)の再利用が必要不可欠であると言われている。しかしながら実際には再利用自身はなかなか簡単ではない。IPを無修正で利用する場合でも、1個あたり12人月以上の工数を要することもある。工数を増大させる問題は次の2点である。

1) IPの仕様を正確に定義し、かつ、IP利用者に誤り無く伝達することが困難。
   このため設計途中で仕様ミスや理解の不一致などによる手戻りが何度も生じる。
2) IP利用者が自分の回路とIPを接続するまでに、多様な「インタフェース検証や接続モジュール」[1](以下、IF検証/接続モジュールと呼ぶ)を、多様な言語により、自ら作成することが必要。
   手戻りが生じたり、IPを類似のインタフェースやサブセットのインタフェースに再利用する場合にも、作り直しが必要になる。IF検証/接続モジュールを作成するための様々な言語に精通するまでの負担も大きい。

上記の問題に対して、標準化団体VSIA(Virtual Socket Interface Alliance)[1]ではオンチップバスの標準化、および、IP仕様書(ドキュメント)の書き方の標準化等を行なっている。それぞれ、「バス統一による設計工数削減」および「書式統一による理解容易な仕様書の実現」が目標である。まず、バスが完全に統一されればIPの再利用はかなり容易になるが、既に各社で使われているバスを再統一するのは難しい。他方の仕様書の標準化により、現在に比べて体系化された仕様を提供することが可能になる。しかしながら、いかんせん仕様書を書くのは人間であり、自然言語を用いて表現しなければならない。そのため仕様書の質は記述者に依存してしまう。しかし現在のところ仕様書の正確さは目視でしか確認できない。したがってVSIAの標準化活動は、体系的な仕様を実現するが、冒頭の2つの問題に対してはまだ改善の余地がある。

以上の分析から我々は、上述の2つの問題の解決を目的とするインタフェース記述言語O_wL(Object Wrapper Language)を開発した[2, 3]。O_wLは論理回路の構造記述言語とは異なり、以下の特徴を持つ。

- 外部端子に生じる全信号変化を網羅的に定義
- IPの持つ各機能毎に体系的に定義
- IPをブラックボックス的に使用する時に必要十分な情報のみ記述(IF検証/接続モジュールを生成する程度の情報も含む)

これらの特徴により、自然言語の曖昧さを解消し、かつ、従来はドキュメントにしか書かれていなかった情報の計算機処理を可能にする。

本稿の前半ではO_wLの概要と、O_wLを用いた上記の問題解決のアプローチを示す。後半は問題の一つである仕様の正確な定義と伝達に対して仕様の問題を分類し、仕様を正確に伝達するための手法O_wL-*Messenger*を提案し、予測効果を示す。

[1]IPを利用するまでにIP利用者が作成する、テストパタンやシミュレーション結果期待値、プロトコルチェッカ、プロトコル変換(ユーザ回路–IP間接続)回路等の、インタフェース関連モジュールのこと。

2 インタフェース記述言語O_wL

2.1 O_wLの概念

IPをブラックボックス的に使う際に必要な情報とは、次の2種である。

意味情報： IPの持つ機能および端子の意味。

インタフェース情報： IP内である機能が動作した場合や、IPの機能を外部から使用する際に端子に発生する信号変化の手順や制約等。

機能を実現するための詳細なアルゴリズムやIPの内部構造などは、一般的には利用者に不要である。以上の2種のうち意味情報は従来通りの仕様書に定義することを前提とし、O_wLではインタフェース情報のみを定義する。なおO_wLは、VSIAが発行する文献[4]中では、"インタフェースプロトコルの定義手法"という分類の一言語として挙げられている。

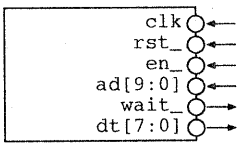
次にO_wLの用語、記述方法と計算機上での表現方法を説明する。

alphabet: 1サイクルの信号値の組合せ。図1(c)の`N`や`R,Q(Xa)`。

word: ある機能を実行したときに起こり得る信号変化順序定義。alphabet名もしくは他のword名を構成要素として、正規表現で記述する。図1(c)の`nop`や`read(Xa,Xd)`。

sentence: 対象とするインタフェースの電源投入時からの起こり得る信号変化の順序定義。alphabet名もしくはword名を構成要素として、正規表現で記述する。O_wL記述に必ず1つ存在する。

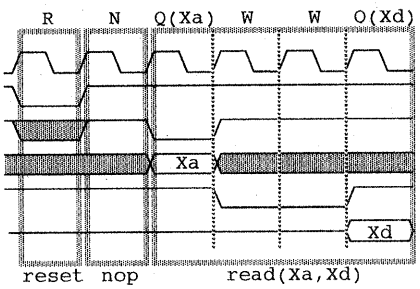
正規表現により記述したO_wLの内容は、1つの始点と終点を持つ状態遷移グラフ(State Transition Graph、以下、STGと書く)により表現する(図1(d))。1つのSTGで1つのwordを表現する。STGの各枝はalphabetもしくは他のwordを表す。wordを表すとき、そのwordのSTGを階層的に



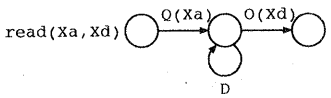
(a)サンプル回路ex1

```
interface ex1;
  port;
    input.clock       clk;
    input.control     rst_;
    input.control     en_;
    input.data  [9:0] ad;
    output.control    wait_;
    output.data [7:0] dt;
  endport
  alphabet;
    signalset all  = {clk, rst_, en_, ad, wait_, dt};
                N    : {  R,   1,   1,   x,    1,   x};
                R    : {  R,   0,   x,   x,    1,   x};
                Q(Xa): {  R,   1,   1,   x,    1,   x};
                W    : {  R,   1,   1,   x,    0,   x};
                O(Xd): {  R,   1,   1,   x,    1,  Xd};
    endsignalset
  endalphabet
  word;
    nop          : N;
    reset        : R;
    read(Xa, Xd): Q(Xa) W* O(Xd);
  endword
  sentence;
    reset [reset | nop | read(Xa,Xd) ]+;
  endsentence
endinterface
```

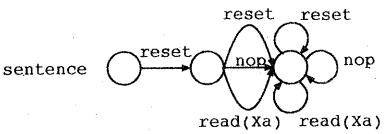
(b)ex1のOwL記述



(c)ex1の動作波形



(d)ex1のword "read"を表すSTG



(e)ex1のsentenceを表すFSM

図 1: O_wL 記述および計算機上での表現方法の例

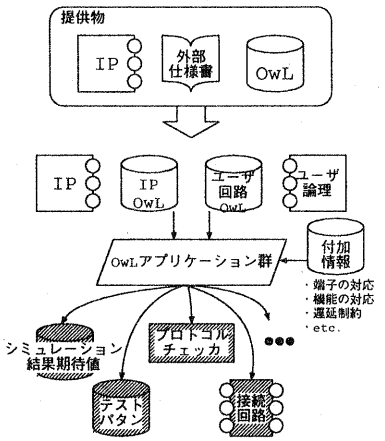


図 2: O_wL がもたらす IP 利用設計

参照する。sentence は終点の無い有限状態遷移機械 (FSM) で表現する (図 1(e))。

STG の始点と終点を結ぶ一経路は、このインタフェースに生じ得る信号変化のあるひとつのバリエーションに相当する。逆に、ある信号変化が STG の一経路に一致すれば、その信号変化は実際のインタフェースに生じ得るバリエーションであると言える。

2.2 O_wL による IP 設計期間短縮へのアプローチ

前述のように、我々は本稿冒頭の IP 利用時の 2 つの問題の解決を目的として、O_wL を開発している。具体的には、O_wL に基づく下記の 2 つの設計手法を構築中である。

1) *O_wL-Messenger*：O_wL を IP の仕様伝達媒体として用いて正確な仕様を定義し、さらに、IP の設計者と利用者の間で正確に伝える手法。
2) *O_wL-Connector*：STG と 信号変化との関係を使用し、IP の利用時に必要な IF 検証/接続モジュールを自動的に生成する手法 (図 2)。例えば、STG の能動的なトレースによりシミュレーションのパタンや期待値を自動生成したり、信号変化通りに STG をトレースを行なうプロトコルチェッカやカバレジ計測機能を自動生成する。

3 IP の仕様と仕様伝達

本章では IP 仕様と仕様伝達の現状を分析し、理想的な IP 仕様伝達に必要な手段を明らかにする。

表 1: IP の仕様分類

仕様の種類		内容
設計仕様	設計イメージ	IP のインプリメントに関し、設計者が頭の中でイメージした仕様。実現アルゴリズムや必要な内部レジスタ等を含む。
	設計仕様書 *	IP を設計するための仕様であり、文書化した設計イメージ。
	設計結果 *	設計仕様書に基づき設計された IP 実物の動作仕様。
外部仕様	設計者側外部仕様イメージ	IP 設計者が考える、IP の使用に必要十分な情報。
	外部仕様書	IP 設計者が IP 利用者に提供する情報。
	利用者側外部仕様イメージ	利用者が外部仕様書から理解した仕様。

*： 本稿では設計結果は設計仕様書通りに設計されていると仮定し、両者を同じと見なす。

3.1 IP 仕様の分類

IP を正しく使うためには、IP 設計者の伝えたい仕様と IP 利用者の理解した仕様が一致していなければならない。設計者は設計した IP に関し、実現方法の細部に至るまでの仕様を把握している。この詳細な仕様を **"設計仕様"** と呼ぶ。

一方、利用者は IP に関して設計者と同程度の知識を持つ必要はないが、IP を使う上で必要十分な IP の仕様を、正確に把握しなければならない。このような IP を使う上で必要な仕様を **"外部仕様"** と呼ぶ。外部仕様は 2.1節で述べた IP の「意味情報」と「インタフェース情報」を含む。

本稿では上記仕様を更に 6 種類に分類する (表 1)。

3.2 IP 仕様間の関係

表 1の各仕様は、理想的には設計仕様を表す仕様同士、外部仕様を表す仕様同士はそれぞれ一致しなければならない (図 3(a))。

しかし、IP 設計時や外部仕様作成および理解時において、誤解や間違いなどの **"人為的バグ"** がしばしば生じる。これらのバグにより、図 3(b)(c)(d) のように各仕様に不一致が生じる。これらの人為的バグは以下の A～H の 8 種類に分類できる。

A. 設計者の実装もれ (外部仕様外)

B. 設計者の実装もれ (外部仕様内)

C. 設計者の仕様未定義

D. 設計者の仕様記述洩れ

E. 設計者の仕様冗長[2]

F. 設計者の仕様記述誤り

G. 利用者の仕様読み落し

H. 設計者の仕様誤解

特に外部仕様理解時の人為的バグ (表 2*G,H*) は、代表的な動作例しか掲載されていないことや、文章のみでタイムチャートが存在しないといった、外部仕様の情報不足から生じる。

上記のバグが一時的に生じることはやむを得ないが、次の設計工程に進む前に排除する必要が有る。排除できないバグは後の設計工程でそれが設計バグとして顕在化し、手戻りが発生することになる。

VSIA による外部仕様の標準化は、外部仕様を体系的に書き、体系的に読むための環境を提供することにより、図 3のうち *B, D, G* の人為的バグを減らす効果がある。しかしながらこれも設計者の能力や仕様書の質に依存するため、問題解決に十分な方法ではない。また、人手で書く仕様書に IP のすべての動作やタイムチャートを網羅的に書くことは労力の面で現実的に難しく、情報不足による理解時の人為的バグは避けられない。

4 仕様伝達手法 O_wL-*Messenger*

4.1 人為的バグの削減方法

3.2節のように、現状では人為的バグの対策に必要な手段や媒体が存在しないため、人為的バグを排除できるか否かは設計者や利用者の能力と努力に依存している。これに対して IP 利用者および IP 設計者が可能な限り多くの人為的バグを体系的に検出する方法が必要である。以下に人為的バグの検出に必要な手段と媒体を示す。

[2]注：*E* は外部仕様を細かく定義し過ぎた部分であり、これが原因で後の仕様変更が難しくなる場合もあるが、どの細かさの外部仕様が最適かということは、外部仕様を記述した段階では未知である。また直接は設計バグにならないため、*E* を対処すべきバグ要因として扱わない。

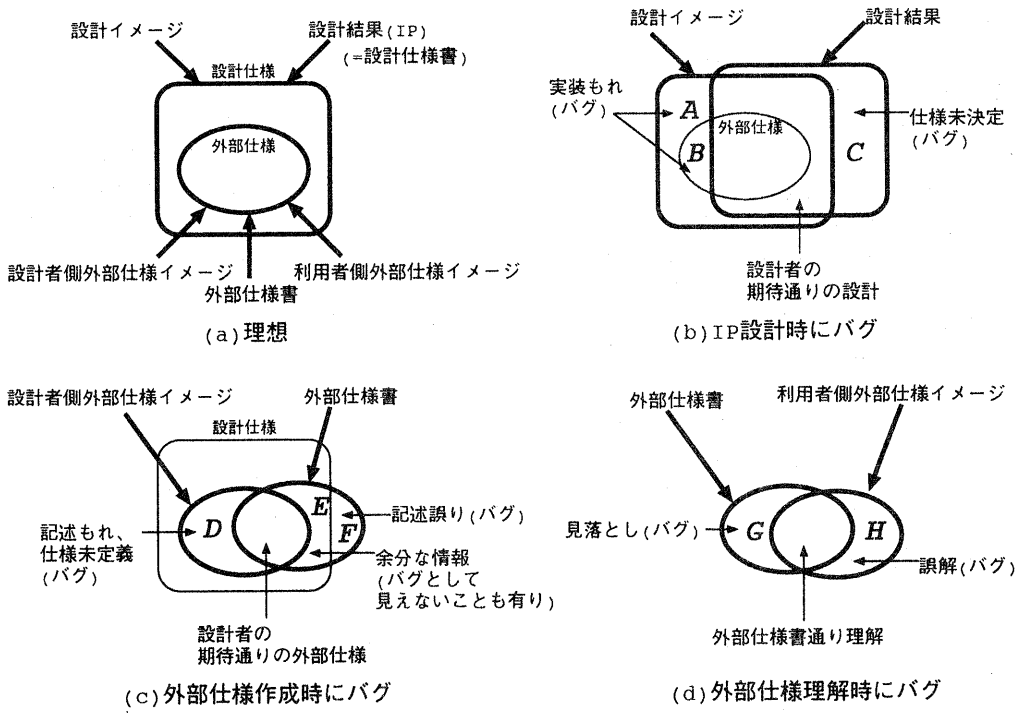


図 3: 設計と外部仕様の関係

表 2: IP 設計時/外部仕様作成時/外部仕様理解時に生じる人為的バグ要因、およびその対策に必要な手段/媒体

バグの種類		対策に必要な手段/媒体			
		IP–設計仕様間一致検証	IP–外部仕様間一致検証	対話的外部仕様確認	網羅的・一意な外部仕様
A	実装もれ (外部仕様外)	√			
B	実装もれ (外部仕様内)		√		
C	仕様未定義	√			
D	仕様記述洩れ		√	√	√
E	仕様冗長	–	–	–	–
F	仕様記述誤り		√		
G	仕様読み落し		√		
H	仕様誤解		√	√	√

√: 対策に必要な手段/媒体。脚注 2 に示す通り本稿では *E* はバグとして考慮しない。

- 設計者側では、設計結果であるIPと設計仕様の間(図3*A,C*)、および、IPと外部仕様の間(図3*B,F*)で矛盾が生じる可能性が有る。「**IP–設計仕様間一致検証手段**」および「**IP–外部仕様一致検証手段**」で削減できる。
- 従来の仕様書のように網羅性に欠けた、対話性のない仕様確認手段では、利用者に情報が伝わり切らず、誤解や見落とし(図3*D,H*)が発生してしまう。「**網羅的な外部仕様記述媒体**」が存在し、利用者がその内容のすべてを「**対話的外部仕様確認手段**」で理解できれば、主体的に人為的バグを削減できる。
- 仮に上記の手段があり、仕様書にすべての情報が盛り込まれていたとしても、仕様の表現方法が自然言語では、読む人によって必ずしも一意でないことがある。これも誤解や見落とし(図3*D,H*)の原因となっている。これは「**一意な外部仕様記述媒体**」で削減できる。

以上の手段と媒体と対応するバグの関係を表2にまとめる。

4.2 O_wL-*Messenger* の構成要素

以上の分析からO_wL記述を上記の仕様記述媒体として、上記の手段を提供する"仕様伝達手法O_wL-*Messenger*"を提案する。3.1節に示したように外部仕様書はIPの意味情報とインタフェース情報を持つ。設計者はそのうちのインタフェース情報をO_wLで記述し、意味情報は従来通りの外部仕様書に書く。

O_wL-*Messenger*はO_wLの他、「IP–外部仕様間一致検証」(以下、**検証手段**)、「対話的外部仕様確認手段」(以下、**確認手段**)で構成される。4.1節の「IP–設計仕様間一致検証手段」は、IP設計に対する一般的な論理検証のことであり、O_wL-*Messenger*では対象とせず、人為的バグの*B,D,F,G,H*を対象とする。各手段が実現する具体的な機能と、検出する人為的バグの関係を以下に示す。

まずO_wL記述の自体の網羅性と一意性から、記述するだけで図3の*D,H*を削減できる。

次に検証手段では、インタフェース情報を記す"外部仕様書"としてのO_wLと、"設計結果"であるIPを比較する。IP設計者側で比較を行なう場合には、O_wL記述に書かれるすべての仕様がIPのインタフェース上に実際に起こることを検証することにより、矛盾点となる図3の*B,F*が取り除ける。また、逆にIP側で生じ得る動作のバリエーションを網羅的に発生させ、それがO_wLに記載されているか否かを検証することにより、O_wLの洩れである図3の*D*が取り除ける。一方、IP利用者側の比較でも上記と同様に、IP側およびユーザ回路側で生じ得る動作のバリエーションを網羅的に発生させ、そのときにO_wLに書かれている通りの信号変化が生じることを検証することにより、図3の*G,H*が検出できる。

最後に確認手段ではO_wL記述の内容を表示して、インタフェース情報を確認する。対話的にすべての仕様が確認できることにより、情報不足から生じる誤解である図3の*H*が検出できる。また、仮に検証手段で取り除けなかった図3の*D*が存在する場合にも、利用者が仕様の洩れに対して疑問を持てば、確認手段で対処できる。

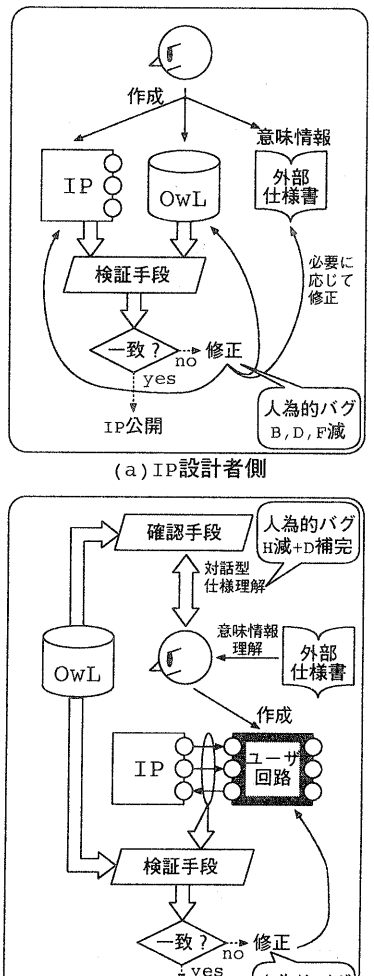


図 4: O_wL-*Messenger*

4.3 O_wL-*Messenger* のフロー

前節の内容をフローとしてまとめる。

IP 設計時 (図 4(a)) 設計者は検証手段により、IP と O_wL 記述の間の矛盾を調べ、人為的バグ B,D,F を削減する。また、矛盾点や未決定な仕様が明らかになれば意味情報を記述した外部仕様書の改善や、回路の実装洩れの解消を行なう。IP と O_wL 記述の比較を、両者が一致するまで行なう。このフローを経ることにより、検証済の IP と O_wL 記述を IP 利用者に提供できる。

IP 利用時 (図 4(b)) 利用者は意味情報を記す外部仕様書と、IP、O_wL を受け取る。仕様理解に確認手段を用いることにより人為的バグ H の発生を抑え、同時に D を補うこともできる。また検証手段を使用し、IP とユーザ回路を接続した際に、IP に許されていないプロトコルをユーザ回路が生成していないかどうかを検証し、人為的バグ G,H を検出する。

5 O_wL-*Messenger* ツール例

ここでは O_wL-*Messenger* の機能を実現するツールとして我々が開発している、検証手段としての O_wL-*Tracer* と、仕様確認手段としての O_wL-*Viewer* を説明する。

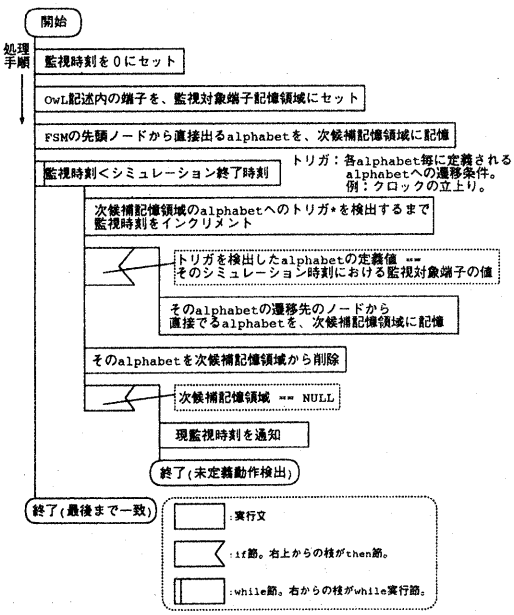


図 5: O_wL-*Tracer* 処理フロー

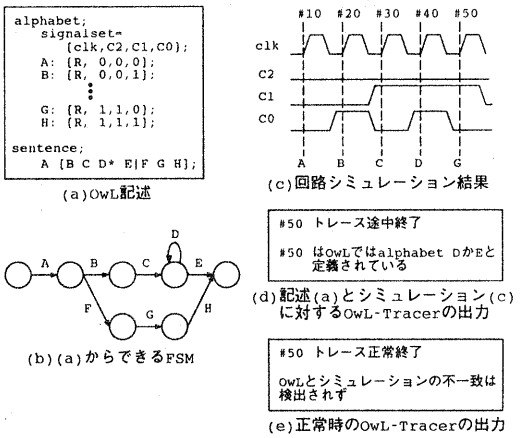


図 6: O_wL-*Tracer* 実行例

5.1 検証手段 O_wL-*Tracer*

O_wL-*Tracer* は、論理回路のインタフェースと O_wL 記述の整合性を確認する手段である。論理シミュレーション実行時に、O_wL 記述内に定義される端子の信号変化を監視しながら、その信号変化通りに FSM をトレースする。O_wL 記述に未定義な信号変化を検出したとき、そのシミュレーション時刻を出力し、処理を終了する。もしもシミュレーションの最後までトレースできたら、論理回路は O_wL に記述された範囲内で動作していることになる。図 5に O_wL-*Tracer* の処理フロー、図 6に実行例を示す。検証の範囲はシミュレーションパタンに依存するため、他にカバレジ計測や形式的手法を用いることで更なる効果が期待できる。

5.2 確認手段 O_wL-*Viewer*

O_wL-*Viewer* は対話的な波形ビューワである。従来の仕様書を比較した場合の最大の相違点は、本ツールは O_wL に書かれるすべての信号変化を確認できることである。それに加えて、O_wL が O_wL-*Tracer* で検証済であれば、波形に誤りがないことも保証できる。理解容易な単位の波形表示と対話的波形表示を実現するための機能を示す。

機能 1) IP の機能、すなわち word 単位の波形表示。

機能 2) 現在表示中の波形に対し、指定する信号線の組合せが生じるか否かの検索。

機能 3) 表示中の波形上のある時刻から他の動作のバリエーションへの対話的変更。本機能で、生じ得る全バリエーションの波形に変更できる。

これらの機能の実現方法の概略を示す。

表 3: 仕様書に含まれる人為的バグの数

記述対象	仕様書頁数	人為的バグ計	仕様洩れ (D)	仕様誤り (F)	仕様誤解 (H)
仕様書 a	42	56	14	3	29
仕様書 b	125	49	0	11	38
仕様書 c	22	21	8	9	4

※括弧内のアルファベットは表 2および図 3に対応。

機能 1) ユーザに選択された word の STG を先頭から順に追って、STG で一つの経路を探索する。STG が分岐する場合は、任意の一つを選択する。ループは高々1 回通過し、初期経路を決定する。経路が確定したら、経路上の alphabet に定義してある信号値を参照して、その通りに波形表示情報を出力する。

機能 2) ユーザに指示された信号値の組合せと等しい alphabet が出現するか否かを調べる。存在する場合にはその alphabet を通過する経路に対応する波形表示情報を出力する。

機能 3) 経路上に STG の分岐が有る場合には、波形上の該当する時刻に分岐を表すマークを表示する (図 7上)。さらに GUI 上でこのマークを指示したとき、STG 上の対応するノードからの分岐を選択肢として表示する (図 7中)。ユーザはここから経路を選択することにより、所望の波形に変更する (図 7下)。

6 O_wL-*Messenger* の効果

仕様書の実例から、現在の IP 設計時と外部仕様作成時および理解時に 3.2節に示す人為的バグがどの程度生じているかを評価した。結果を表 3に示す。今回評価の対象としたのはバスなどの外部仕様書である。仕様書のレビューなどにより人為的バグを計測した。なお、ここに誤字脱字等の誤りは含まない。

検出した人為的バグを分析した結果、原理的には表中の "仕様誤解" は O_wL-*Viewer* で、"仕様洩れ"、"仕様記述誤り" は O_wL-*Tracer* で検出可能であることが確認できた。従来は、人為的バグは設計者と利用者の能力と努力によってしか検出できなかったが、この結果から、O_wL-*Messenger* というツールサポートにより体系的に検出できる見通しを得た。

7 まとめ

IP を再利用する際に IP の仕様を利用者に正確に伝達することが困難なことや、IP 利用時の利用者で用意するインタフェース検証や接続のモジュールが多いことが問題となっている。

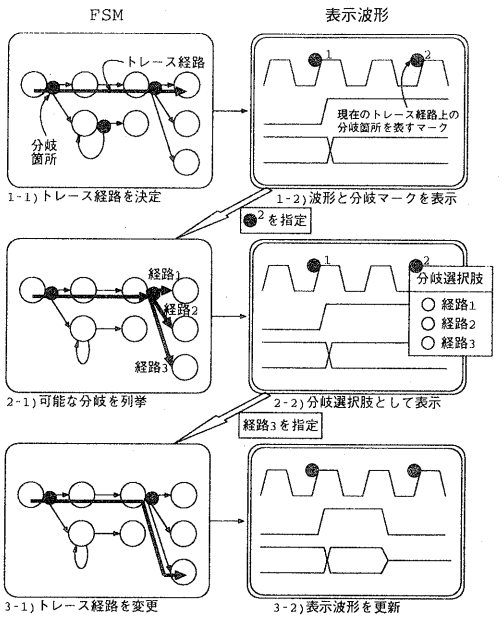


図 7: O_wL-*Viewer* 機能 3) 実行例

この問題の解決を目標に我々はインタフェース記述言語 O_wL を提案している。本稿では仕様を正確に伝達するための手法 O_wL-*Messenger* の概念と実現例を示し、評価した結果、従来は人手で探すしかなかった人為的バグを検出できる見通しを得た。

O_wL のような言語は、言語自体とその言語を使うツールが広く普及しなければ意味が無いため、言語仕様のオープン化を検討している。今後は早急に O_wL-*Messenger* および O_wL を用いた設計支援手法 O_wL-*Connector* を具体化して、実際の設計で使用するための環境を整備したい。

参考文献

[1] Virtual Socket Interface Alliance (VSIA). http://www.vsi.com.

[2] 荒宏視, 鈴木敬, 矢野和男. IP 再利用のためのインタフェース記述言語 O_wL の提案. DA シンポジウム, pp. 15–20, 1999.

[3] Kei Suzuki, Kouji Ara, and Kazuo Yano. O_wL: An Interface Description Language for IP Reuse. In *Proceedings of the Custom Integrated Circuits Conference*, pp. 403–406, May 1999.

[4] System-Level Design Development Working Group. System-Level Interface Behavioral Documentation. *VSIA*, Mar. 2000.