設計自動化　45−1
(1988. 12. 15)

# ＬＳＩディレイテストシステム

## A DELAY TEST SYSTEM FOR LOGIC LSI'S

白鳥　文彦　　岸田　邦明　　石山　俊　　池田　光二 *
坂本　頼之 **
Fumihiko SHIROTORI　Kuniaki KISHIDA　Shun ISHIYAMA　Koji IKEDA *
Yoriyuki SAKAMOTO **
*：日立製作所　**：日立コンピュータエンジニアリング
*：Hitachi,Ltd. **：Hitachi Computer Engineering Co.,Ltd.

あらまし　論理ＬＳＩの遅延故障を検出するためのテストデータを自動生成するシステムについて故障モデル及びそれにもとづく検出率を定義し、故障を検出するためのテスト手順を明示した。　システムを実用化するにあたり、テスト容易化設計の定着のためのザインルールチェック機能、及びシステムの処理中断・リスタート機能の採用によりシステム性能の向上を図っている。　本システムは情報処理システム用論理ＬＳＩに適用し有効性を確認している。

Abstract　This paper presents the definition of a fault model and the measure of fault coverage for a delay test system, which detects the delay fault in logic LSI chips.　By adopting a design rule checking to gain high testability and a interruption and resumption strategy, we succeeded in developing the system with high performance.　The delay test system has been applied to logic LSI chips for information processing systems and justified its effectiveness.

## １．　はじめに

情報処理技術の急速な進歩に伴い、システムの大規模化・高性能化が図られており使用するＬＳＩも高速・高集積化が要求されている。

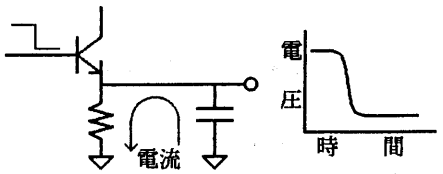
特にメインフレームやスーパーコンピュータ用の高速論理ＬＳＩにおいては信号伝播遅延時間の異常（遅延故障）によって正しい動作が行なわれないことが知られており、計算機の信頼性を確保するためには、遅延故障を検出する手段の開発が必要不可欠となっている。

本論文では、論理ＬＳＩの遅延故障を検出するためのテストデータを自動生成するディレイテストシステムを超大形計算機及びスーパーコンピュータ用論理ＬＳＩに適用したので、システム概要及びシステム性能向上手法について述べる。

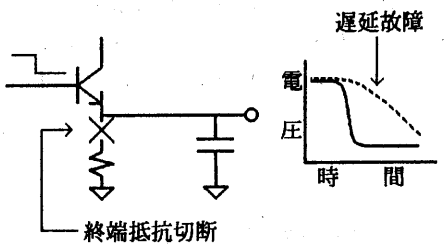
## 2．　ディレイテスト概要

### 2．1　遅延故障モデル

本システムが対象とする遅延故障の例を図１に示す。




（1）　正常回路




（2）　遅延故障回路

図１　遅延故障例

バイポーラトランジスタのベース部に立ち下がり信号が印加された場合、エミッタ部の寄生容量に蓄えられた電荷は終端抵抗を介し高速に解放され正常動作が行なわれる。(図１（１))ＬＳＩ製造工程での異物等により終端抵抗が切断されると、解放時間が極端に遅くなり、立ち下がり信号伝播遅延故障となる。(図１（２))

このような遅延故障は、ストローブ時間の長いＤＣファンクションテストでは検出できず、計算機組立後の動作不良となる。

ＤＣファンクションテストのストローブ時間を短くする手法でもある程度の故障を検出可能であるが、スタック故障検出を目的としたステテトパターンでは充分な効果は得られず、一方論理ＬＳＩの高速化が著しく進み遅延故障が情報処理システムの信頼性を大きく左右するに至り、専用のデータによるテストが必要となった。

### 2．2　テスト方式

ディレイテストは、ＬＳＩ内部の各ゲート間のパスディレイが許容範囲内にあることを検証するものである。[1][2][3][6]

しかし、ＬＳＩ内部のゲートディレイは直接観測することは困難なので、本システムではラッチの論理値を自由に制御できるスキャン論理の組込みを前提とし、以下の区間のパスディレイを検証する。

（１）　ラッチーラッチ
（２）　ＬＳＩピンーラッチ
（３）　ラッチーＬＳＩピン
（４）　ＬＳＩピンーＬＳＩピン

図２にテストの方式を示す。

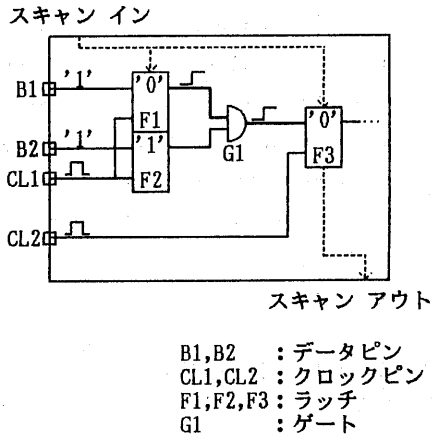



図２　テスト方式

テスト手順は以下の通りである。

STEP1：　Ｆ１→Ｇ１→Ｆ３のパスを活性化するための条件を決定する。

STEP2： 活性化条件に従い各ラッチ、データピンを初期設定する。

（Ｆ１←０，Ｆ２←１，Ｆ３←０
Ｂ１←１，Ｂ２←１）

STEP3： ＣＬ１、ＣＬ２にクロックパルスを下記の間隔で印加する。

パルス間隔 ＝ CL1→F1のディレイ値
＋ F1→F2のディレイ値
－ CL2→F1のディレイ値

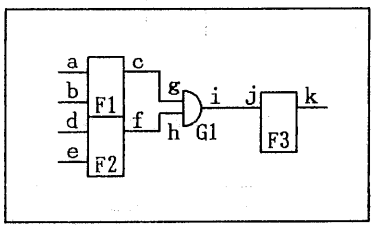
STEP4： Ｆ３の論理値をスキャンアウトして遅延故障の有無を判定する。

Ｆ３の論理値：（１→正常
０→遅延故障有）

以上の４つのステップを繰り返して、ＬＳＩ全体の遅延故障を検出する。

## ２．３ 活性化率の定義

作成したテストデータの品質は図３に定義する活性化率で評価される。




ピン対 －－ （a-c，b-c，d-f，e-f
g-i，h-i，j-k）

$$\text{活性化率} = \frac{\sum_{\text{活性化パス上}} \{\text{ピン対} \times (\text{立上り}+\text{立下り})\}}{\sum_{\text{全ゲート}} \{\text{ピン対} \times (\text{立上り}+\text{立下り})\}}$$

図３ 活性化率の定義

ピン対はゲートの入出力ピンの組み合わせを意味し、各ピン対に対して立上り／立下り遅延故障が仮定され、活性化率１００％で全ての故障が検出される。 但し、スキャン系パス上のピン対は故障仮定対象外としている。

本手法では、全ての故障を検出するために、２．２節で挙げた（１）から（４）までの全てのパスを活性化する必要がなく、活性化パス数／全パス数で評価するよりも効率良いデータ作成が可能となる。

## ２．４ システム概要

図４にシステムの概要を示す。

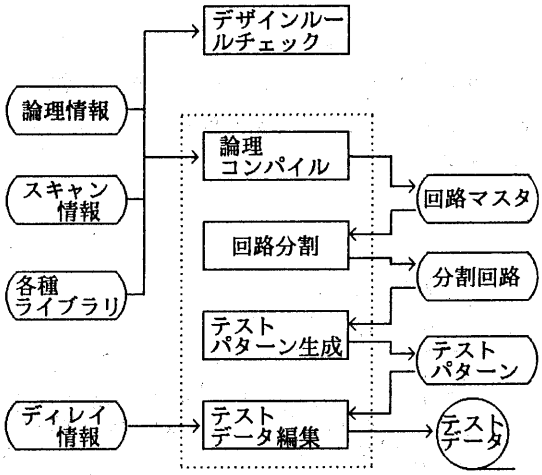



図４ システム構成

（１） デザインルールチェック

論理設計段階でディレイテスト適用不可の論理を検出する。 デザインルールの内容については次章でのべる。

（2）　論理コンパイル

論理情報、スキャン情報、各種ライブラリの内容をシステム内部のデータ形式に編集し、回路マスタファイルを作成する。

（3）　回路分割

ＬＳＩ全体をラッチ又はＬＳＩピンで区切り（サブ回路と呼ぶ）分割回路ファイルを作成する。（図５参照）

サブ回路内部は組み合わせ回路となるのでテストパターン生成が容易となる。

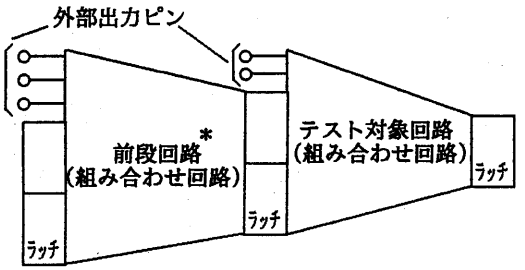
（4）　テストパターン生成

サブＬＳＩ毎にディレイテストパターンを成生する。

テストパターン生成は、一次元活性化法を基本とした専用アルゴリズムを用いている。[4]

（5）　テストデータ編集

サブＬＳＩ毎のテストパターンにテスト基準となるディレイ値をディレイ情報をもとに付加し、テストデータを出力する。




＊：ラッチから立上り／立ち下り信号を発生させるため
テスト対象回路と共にサブ回路に含める。

図５　サブ回路の定義

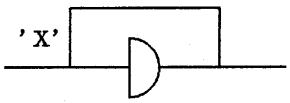
## ３．　システム性能向上手法

### ３．１　デザインルールチェックと ルール違反への対応

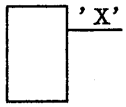
本システムを適用し充分な効果を得るためには、論理設計の段階でいくつかのデザインルールを守る必要がある。　このためルール違反の有無を検出する機能を設け、テスト容易化設計を論理設計者に意識・定着させている。

また、ルール違反を完全に除去することが困難な場合があることを考慮し、ルール違反部を全体の論理から切り離し、テストパターン発生対象外とする手法を採用した。　具体的には、違反の影響が最小となる箇所に'X'(不定)値を挿入して誤ったパターン発生を抑止する。　この手法により'X'値を含むパスのみが活性化対象外となり、活性化率の極端な低下を防ぐことができる。

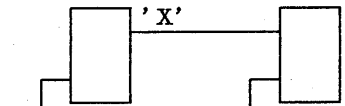
図６にデザインルール違反と'X'値挿入の例を示す。




（1）　ループ論理




（2）　制御不可ラッチ




（3）　同相転送

図６　デザインルール違反例

## ３．２　処理時間増加への対応

テストデータ作成処理時間はＬＳＩのゲート数と共に論理の複雑度に大きく左右される。

我々はいくつかの処理効率改善手法を盛り込み、その一部は既に報告済みであるが、[5] 特定の論理構成で効率が悪くなる、あるいは活性化できないなど、いわゆる得意、不得意が生じている。　すなわち多くのＬＳＩでは下記の関数に従うが、一部の複雑な論理構成のＬＳＩでは極端に処理時間が増大する場合がある。

$$T = A + B \times G + C \times G^{D}$$

Ｔ　：　全体処理時間
Ｇ　：　ＬＳＩゲート数
Ａ，Ｂ，Ｃ　：　係数
Ｄ＝２～３

図７に処理時間の特に大きな品種の処理別時間内訳の例を示す。　テストパターン発生処理だけが極端に大きくなっており、このような品種では特定ジョブのＣＰＵ占有によるＴＡＴ(Turn-Around Time)悪化防止のために設けられたＣＰＵ使用時間制限により、テストパターン発生処理途中で処理が打ち切られるため、活性化率が極端に低下する。

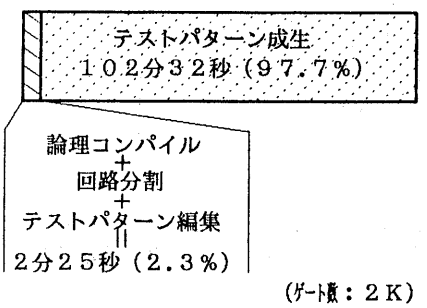



図７　処理別時間内訳例(処理時間特大品種)

図８、図９に２ＫゲートＬＳＩでの活性化率と処理時間の分布を示す。　全体の約３％の品種で処理の打ち切りが発生し、活性化率は低下している。（図８、図９の斜線部）

また、処理別時間内訳は図７の例と同様テストパターン成生以外は数％であった。

そこで、処理の打ち切りによりテストパターン発生対象から漏れた回路に着目してテストデータを作成するため、リスタート手法を採用した。

本システムにおけるリスタート機能は、短期間に多品種のテストデータを作成する運用形態に即した方式となっており、つぎの２点が特徴となっている。

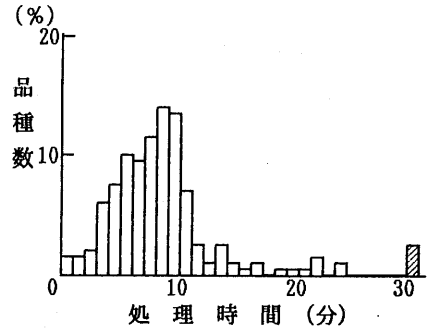



図８　処理時間分布

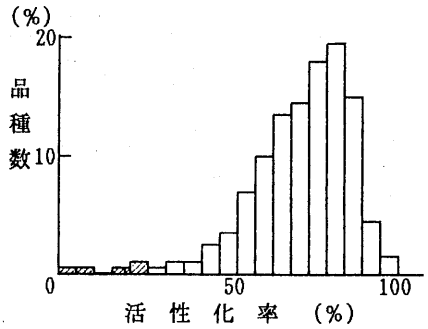



図９　活性化率分布

（１）　リスタートのために特別なバックアップエリア等の準備が不要であること。

（２）　リスタート前後の入力ファイル等の履歴チェックの自動化により、ユーザのリスタートに伴うチェック工数を最少限とすること。

リスタート機能の内容は以下のようになっている。（図１０参照）

（１）　中断機能

処理打ち切り時間に達した場合それまでに成生したテストパターンに加え、以下の情報をテストデータファイルのあらかじめ確保されたエリアに編集・出力する。

（ａ）　仮定した故障に対する、検出／未検出／未試行の区分（故障管理情報）

（ｂ）　入力ファイル、ライブラリ等の履歴管理情報

（２）　リスタート機能

（ａ）　履歴チェック

既作成のテストデータ中の処理打ち切り時の履歴管理情報を入力、リスタート時と比較し、同じであることをチェックする。

（ｂ）　回路分割

処理打ち切り時にサブ回路情報の保持は行なわず、リスタート毎に回路分割処理を行なう。

サブ回路情報の保持には専用のバックアップエリアの確保が必要であり、また前述のように回路分割を含むテストパターン成生以外の処理時間は、処理打ち切り発生品種の場合全体処理時間の数％であることから、回路分割処理をオーバーヘッドさせ、運用性を優先させた。

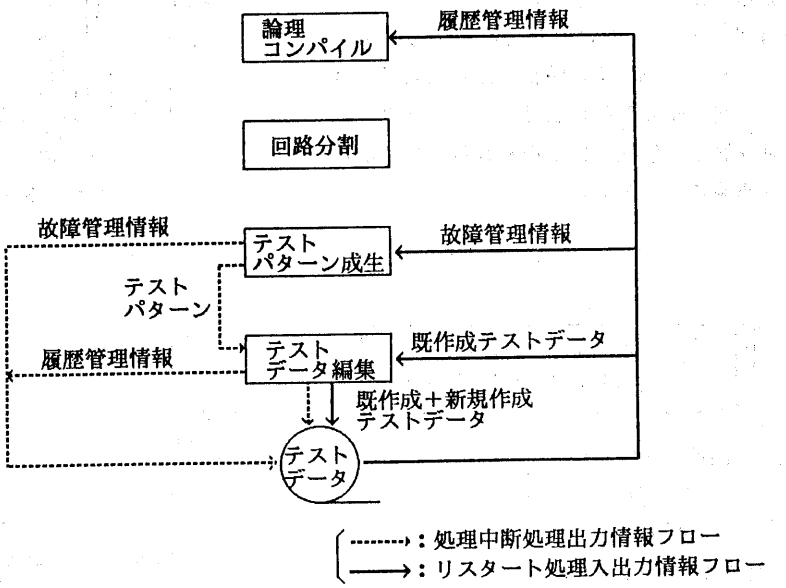



図１０　リスタート機能概要

（ｃ）　テストパターン生成

既作成のテストデータ中の故障管理情報を入力し、テストパターン生成処理を行なう。　この際、原理的に検出不可能な故障はテストパターン生成の対象外とすることで効率の良いリスタートを図っている。

（ｄ）　テストデータ編集

既作成と新規作成のテストデータをマージして出力する。

表１に２Ｋゲートベンチマークデータ８品種でのリスタート試行結果を示す。

表１　リスタート試行結果

| データNo. | 活性化率 (%) リスタート前 a | リスタート後 b | 改善量 b－a | リスタート回数 |
|---|---|---|---|---|
| 1 | 40.25 | 78.15 | 30.90 | 1 |
| 2 | 4.48 | 15.13 | 10.65 | 3 |
| 3 | 51.54 | 84.77 | 33.23 | 1 |
| 4 | 39.18 | 77.65 | 38.47 | 3 |
| 5 | 53.74 | 88.95 | 35.21 | 1 |
| 6 | 40.06 | 89.27 | 49.21 | 1 |
| 7 | 65.18 | 79.04 | 13.06 | 1 |
| 8 | 59.03 | 76.65 | 17.62 | 1 |
| 平均 | 44.06 | 73.82 | 29.76 | 1.5 |

８品種平均では、リスタート　１．５回で約３０％の活性化率向上改善となり、本手法が有効であることを確認した。

図１１は表１中のデータNo.４のリスタートと処理時間の制限を解除した場合の処理時間による活性化率の推移を示したものである。

リスタートと制限解除の最終活性化率は共に７７．６５％であり処理時間は各々１０５分、１１４分 で差は ９分 である。　履歴チェック等のリスタート固有処理及び回路分割等のオーバーヘットを合わせたリスタートによる処理時間増加率は ８．５％ であり、回路分割処理をオーバーヘットさせた影響は小さいと考える。

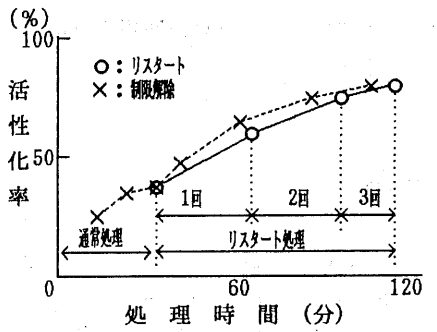



図１１　処理時間推移

## 4．　おわりに

論理ＬＳＩの遅延故障を検出するテストデーデータを自動生成するシステムの概要と性能向上手法について報告した。

本システムは実用化済みであり、情報処理システム用ＬＳＩの品質保証に有効であることを確認している。

今後は、更に高速化、高集積化の進むＬＳＩへの幅広い対応が課題である。

## 謝　辞

本研究、論文をまとめるにあたり、貴重な御助言、御協力をいただいた関係各位に深く感謝いたします。

## 参考文献

[1] E. P. Haieh et al.,
"Delay test generation",
Proc,. 14th DA Conf., PP. 486-491,1977.

[2] T. M. Storey et al.,
"Delay test simulation",
Proc., 14th DA Conf., pp. 492-494,
1977.

[3] J. D. Lesser and J. J. Shedletsky,
"An experimental delay test generator for LSI logic",
IEEE Trans.Comput., Vol. C-29, NO. 3,
pp. 235-248, 1980.

[4] T. Hayashi et al.,
"A delay test generator for Logic LSI",
IEEE FTCS-14., pp. 146-149, 1984.

[5] K. Kishida et al.,
"A delay test system for high speed logic LSI's",
Proc., 23th DA Conf., PP. 786-790,
1986.

[6] A. K. Pramanick and S. M. Reddy,
"On the detection of delay Faults",
Proc., 1988 Int. Test Conf., PP. 845-856, 1988.