設計自動化　40－16
(1987. 12. 18)

時間記号シミュレーションについて
On Time-Symbolic Simulation

石浦 菜岐佐　矢島 脩三
Nagisa Ishiura　Shuzo Yajima

京都大学工学部
Faculty of Engineering, Kyoto University

あらまし

論理回路のタイミング検証の一手法として、時間記号シミュレーションを新たに提案する。従来の記号シミュレーションが信号値を論理式として扱うのに対し、時間記号シミュレーションは時刻を代数式として扱う。即ち、ゲート遅延や入力の与えられる時刻を変数で表現することによって、その値のばらつきをモデル化し、シミュレーション時刻をこの変数を含む式のままで扱うことにより、精密なタイミング解析を行おうとするものである。本稿では、時間記号シミュレーションのアルゴリズム、時刻を表す式の大小比較や簡単化の手法、及びハザード検出や非同期順序回路の検証への応用について述べる。

Abstract

As a new approach for timing verification of logic circuits, we propose *a time-symbolic simulation*. While the conventional symbolic simulator treats signal values as logical expressions, the time-symbolic simulator treats time as algebraic expressions. In this article, we describe algorithms for time-symbolic simulation, techniques for comparison and reduction of algebraic expressions representing simulation time, and its application to the hazard detection and the verification of asynchronous sequential circuits.

1. はじめに

近年の集積回路技術の進歩に伴い、より大規模で高機能な論理システムが実現されるようになっているが、その設計検証を如何に行うかという問題が、CAD/DA(計算機援用設計/設計自動化)技術に課せられた大きな課題となっている。その中でもタイミングに関する検証は、複雑で困難な場合が多く、大規模な論理回路はタイミング検証の容易な同期式順序回路として設計されることが多い。しかし、通信制御などの周辺論理や、記憶素子など非同期回路として設計されるものに対しては、仕様通りの動作をするかどうか、あるいはハザード、発振、競合によるエラーが発生しないかどうかを調べるために、論理的動作との連携をも考慮した精密なタイミング解析が必要になる。このように微妙なタイミングが問題になる場合には、製造条件や使用環境の変化による素子遅延のばらつきを考慮する必要が生じる。論理シミュレーションではこのような遅延のばらつきを、不定値を用いた最大/最小遅延シミュレーション[1]により解析するが、現実よりも悲観的な結果を出力する(必要以上の不定値を出力する)ことが知られており、これによって正確なタイミング検証を行うことは難しい。また、時相論理やオートマトン理論に基づく形式的検証手法[2]の中には、遅延のばらつきを扱えるものもあるが、ほとんどが離散時間を仮定している。これは、離散時間は現実の連続時間の近似であるという考えに立つものであるが、離散時間が連続時間の十分な近似になるための(離散時間のきざみ幅等に関する)条件、及びその様な条件下において実用的な計算量で処理が行えるかについての明確な議論は少ない[6]。

本稿ではこのような問題を解決する一手法として、時間記号シミュレーションという新たなシミュレーション手法を提案する。従来から提案されている記号シミュレーション[3),4)]が、信号線の信号値を記号として扱うのに対し、時間記号シミュレーションは時間を記号として扱う。即ち、ゲート遅延や入力の変化する時刻を変数で表現することによってばらつきをモデル化し、シミュレーション時刻をこの変数を含む式のままで扱うことにより、精密なタイミング解析を行おうとするものである。シミュレーション時刻が定数でないため、従来のシミュレーション・アルゴリズムでは計算が困難であるが、我々が以

前に提案した時間優先評価アルゴリズム(Tアルゴリズム)[5]の考え方に基づけば、時刻が変数を含む式である場合にもシミュレーションが可能になる。また、時刻を表す式の簡単化や比較の方法は、式の形が変数の一次式に限られることに着目し、線型計画法に帰着して処理する。時間記号シミュレーションを用いれば、ハザード検出問題や非同期順序回路の検証が精密に行え、さらに、ハザードが発生しない条件、非同期順序回路が正常に動作する条件を、遅延を表す変数の不等式として求めることができる。また、これまでに提案されているほとんどの検証系が離散時間を仮定しているのに対し、本稿の時間記号シミュレーションは連続時間を扱うことができるのも一つの特長である。

本稿では、時間記号シミュレーションの概念とアルゴリズム及びその応用の可能性について述べる。以下2章では、時間記号シミュレーションのアルゴリズムの基礎となるTアルゴリズムについて述べた後、3章でアルゴリズムの詳細、4章で時間記号シミュレーションの適用例について述べる。

2. SアルゴリズムとTアルゴリズム

ゲート遅延を考慮した論理シミュレーションの基本的アルゴリズムとしては、イベント法[1]が広く用いられている。イベント法は回路中の信号線の信号値変化をイベントとしてとらえ、その伝播を計算することによりシミュレーションを行うものである。従来より、タイム・ホイール(あるいはタイム・キュー)と呼ばれるデータ構造を用いてイベントを時刻ごとに管理し、

(S1) シミュレーション時刻を逐次(あるいは次のイベントが発生する時刻まで)進める。

(S2) その時刻において発生するイベントを取り出し、その影響を受けるゲートの出力を計算する。

(S3) その結果ゲート出力が変化すれば、新たにイベントとして登録する。

という手順を繰り返すことによりシミュレーションを行う方法が一般的である。このようなアルゴリズムは時間を逐次進めながら、各時刻において空間方向の計算を行うという観点から空間優先評価のアルゴリズム(以下Sアルゴリズム)と言うことができる。これに対し我々は、空間の各点(具体的には各ゲート)毎に時間方向に計算を行うことによってシミュレーションを行う、時間優先評価アルゴリズム(以下Tアルゴリズム)を提案している[5]。Tアルゴリズムはフィードバック・ループを含む回路のシミュレーションに対しては複雑になるが、組合せ回路のシミュレーションに対しては単純で、効率も良くなる。

Tアルゴリズムでは、信号線の信号値系列は図2.1に示すようなイベントのリストで表現される。但し、リストの先頭には初期値を表すイベント(時刻が$-\infty$)が、末尾にはシミュレーションの終了を表すイベント(時刻が∞)が来る。全ての入力について、信号値系列が与えられているか既に計算されているゲートを評価可能ゲートと呼ぶことにする。Tアルゴリズムによる組合せ回路のシミュレーションは、(T1)~(T4)のようになる。

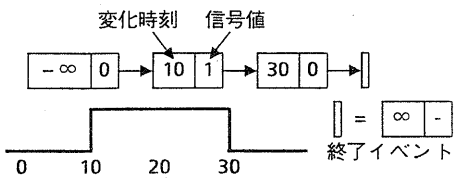


図2.1 信号値の表現

(T1) 全入力が外部入力につながるゲートを評価可能ゲートとして登録する。

(T2) (T3)~(T4)を評価可能ゲートが無くなるまで繰り返す。

(T3) 評価可能ゲートを取り出し、出力の信号値系列を計算する。

(T4) (T3)の結果新たに評価可能ゲートが発生すればこれを登録する。

上記(T3)におけるゲート出力の信号値系列は、(E1)~(E4)により計算される。

(E1) 入力の初期値イベントから出力の初期値イベントを計算する。

(E2) (E3)~(E4)を入力に発生するイベントが無くなるまで繰り返す。

(E3) 入力イベント・リスト中の未評価のイベントのうち、時刻が最小のものを選ぶ。

(E4) このイベントを評価し、出力を計算する。もし出力が変化すれば、新しいイベントとして出力イベント・リストの最後尾に付加する。

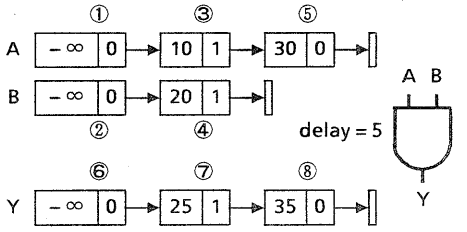


図2.2 出力の計算

図2.2はTアルゴリズムによるシミュレーションの例である。

(1) ①②より初期値⑥を計算する。

(2) ③を評価する。出力は0のまま変化しないので、出力イベントの追加は行わない。

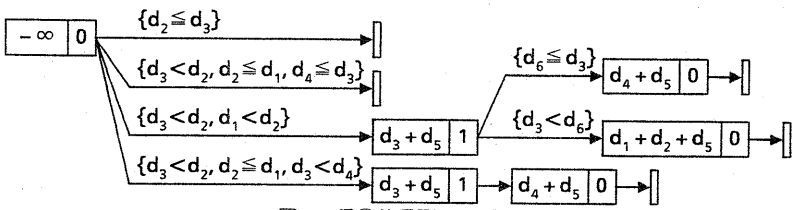


図3.1 信号値系列の表現

(3)④を評価する。出力が変化するので出力イベント⑦をリストに追加する。⑦の時刻は④の時刻にゲートの遅延を加えたものである。

(4)(3)と同様⑤を評価し、⑧を追加する。

なお、立上り/立下り遅延や慣性遅延を考える場合には、一度出力に追加したイベントを取り消す操作が必要になるが、これに関しても容易に行うことができる。

3. 時間記号シミュレーションのアルゴリズム

3.1 時間変数

時間記号シミュレーションでは、ゲート遅延や入力の与えられる時刻のばらつきを正確にモデル化するために、ゲートの遅延や入力の与えられる時刻を変数で表す。この変数を時間変数と呼ぶ。時間変数のとる値の上限下限が与えられる場合には、{0<d, d≦5}等のように変数に関する不等式集合でこれを表す。この不等式集合を、変数制限条件と呼ぶ。時間変数は、変数制限条件のもとで可能な全ての連続値を取り得るものとする。時間記号シミュレーションでは、シミュレーション時刻は時間変数の一次式で表現される。

3.2 Tアルゴリズムよる時間記号シミュレーション

時間が定数ではなく時間変数による式で表される場合、従来のように時刻を逐次進めるSアルゴリズムでシミュレーションを行うことは難しい。そこで本稿では、まずTアルゴリズムに基づく計算手法を考える。対象は組合せ回路に限る。

時間記号シミュレーションにおける信号線の信号値系列は、図3.1のようにイベントを節点とする有向木で表される。有向枝には、時間変数の一次不等式の集合が割当てられている。有向枝はイベントの発生順序を示し、有向木の根には初期値を表すイベントが、葉にはシミュレーションの終了を表すイベントがくる。イベントの発生時刻は時間変数の一次式で与えられる。1つのイベントから複数の有向枝が出ている場合は、条件によってそのうちのいずれかが起こることを示している。有向枝につけられた不等式集合は、その有向枝が選択される条件を表しており、分岐条件と呼ばれる。根から葉に至るパス上にあるイベントの系列は、この信号線に発生しうるイベントの系列を表している。パス上にある有向枝の不等式集合の和集合は、そのようなイベント系列が発生する条件を表しており、パス条件と呼ばれる。

シミュレーションは、時間が定数のTアルゴリズムと同様に、入力の信号値系列が全て計算されたゲートについて出力の信号値系列を計算する、という操作を繰り返し行うことにより進められる。但し、ゲートの出力の信号値系列の計算は、次節で詳しく述べるように、時間が定数の場合に比べて複雑になる。

3.3 出力信号値系列の計算

出力信号値系列の計算も、基本的には時間が定数の場合と同じであるが、以下(B1)、(B2)に述べるような相違がある。

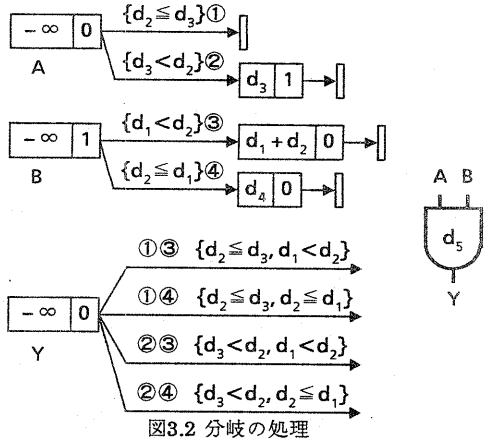


図3.2 分岐の処理

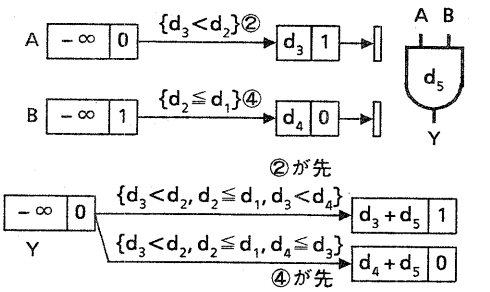


図3.3 時刻最小のイベントが決定できない場合

(B1) ある入力信号線に次に起こり得るイベントは複数存在し得るので、それぞれが起こった場合について計算を行わなければならない。例えば図3.2のように入力A、Bの次のイベントとしてそれぞれ①と②、③と④が起こり得るときには、それぞれ①③、①④、②③、②④が起こるという全ての組合せについて、次の出力イベントを計算しなければならない。このとき、次の出力イベントを指す有向枝には、分岐条件として入力イベントの組合せに対応する条件(それぞれの有向枝についた分岐条件の和集合)を付ける。もし、ここまでのパス条件や変数制限条件から、ある組合せは起こり得ないと結論できれば、これを排除する必要がある。例えば、これまでのパス条件に、$d_3<d_1$が含まれていれば、$\{d_1<d_2, d_2\leqq d_3, d_3<d_1\}$を同時に満たす$(d_1, d_2, d_3)$は存在しないので、①③の組合せは起こり得ない場合として排除しなければならない。

(B2) 上記場合分けにより、各入力について次のイベントが一意に決まっても、イベントの発生時刻が記号を含む式で表されているため、一般に時刻最小のものを一意に決めることはできない。従ってここでも場合分けを行って計算を進める必要がある。例えば図3.2において②④の組合せを選んだ場合、d_3とd_4の大小が決定できなければ、それぞれのほうが小さい場合についての計算を行わなければならない(図3.3)。このように場合分けを行った場合には、(B1)と同様に次の出力イベントを指す有向枝に、分岐条件として$d_3<d_4$等を付加する。また、ここまでのパス条件や変数制限条件から、あるイベントの時刻が最小になり得ないと判断できる場合には、これを排除する必要がある。

図3.4は図3.2で与えられた入力に対し、出力を計算した結果である。(a)~(f)のパスは、それぞれ以下に示す場合分けに対応するものである。

(a) ①③の組合せが起こる場合。
(b) ①④の組合せが起こる場合。
(c) ②③の組合せで、②が先に起こる場合。
(d) ②③の組合せで、③が先に起こる場合。
$d_3<d_2$より明らかに$d_3<d_1+d_2$は成立しないので、この場合は排除される。
(e) ②④の組合せで、②が先に起こる場合。
(f) ②④の組合せで、④が先に起こる場合。

図3.5は図3.4を整理したものであり、これが求める結果である。なお、本節では簡単のため立上り/立下り遅延や慣性遅延については言及しなかったが、実現は容易である。但し、場合分けの数は増加すると考えられる。

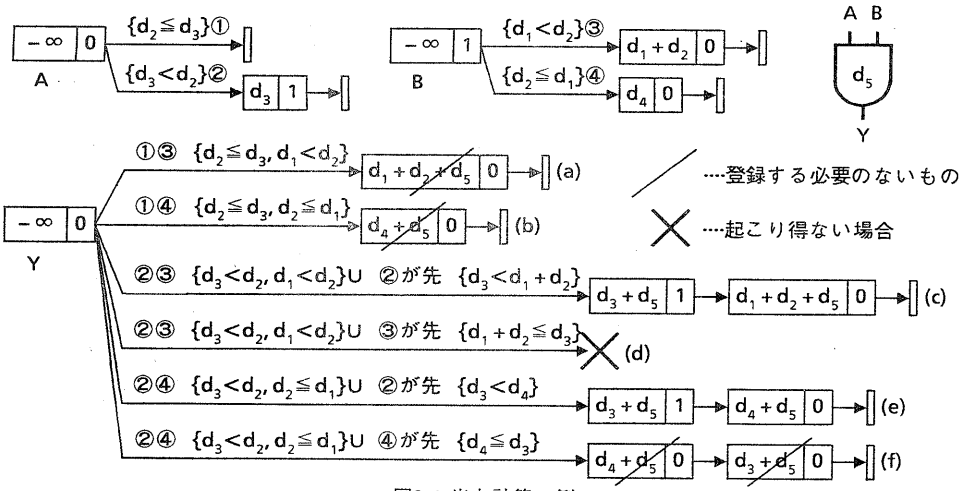


図3.4 出力計算の例

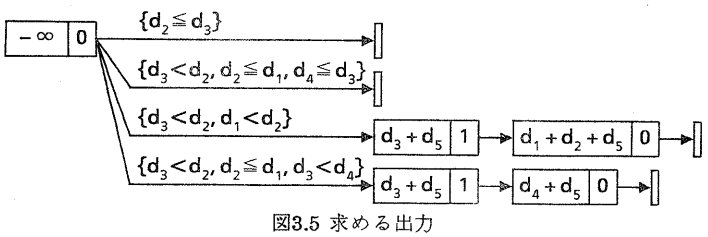


図3.5 求める出力

3.4 記号式の扱い

前節で述べた処理を実行するためには、時間変数による式で表される時刻を扱わなければならない。必要な操作をまとめると以下の通りである。

(C1) 時刻、遅延の加算。

(C2) 不等式の簡単化($d_3+d_2<d_1+d_2 \rightarrow d_3<d_2$等)。

(C3) (B1)(B2)において、パス条件と変数制限条件から起こり得ない場合を排除すること。

(C4) 分岐条件集合中の不要な不等式を削除すること。($\{d_1<d_2, d_3<d_4, d_1+d_3<d_2+d_4, d_1<d_2+d_3\}$のうち、後の二つは最初の二つより導けるので不要である。)

このような代数式の操作には、項書き換えシステムを用いるアプローチも考えられるが、扱う式が時間変数の一次式に限られることから、より容易に処理することができる。(C1)は係数どうしの加算により実現できる。(C2)は不等式を$\Sigma_{i=1,n}a_id_i+c\leqq 0$及び$\Sigma_{i=1,n}a_id_i+c<0$という標準形で表すことにより実現できる。(C3)は、新しいパス条件と変数制限条件を表す不等式集合を、時間変数の連立一次不等式とみてその解の存在を判定する問題である。これは線型計画法において制約式が基底を持つかどうかという問題に帰着できる。(C4)は、ある不等式集合C中の不等式qが不要であるのは$(C-\{q\})\cap\{q'\}$(但し、q'はqの不等号を反転させたもの)が連立一次不等式として解を持たないときであるので、(C3)に帰着できる。さらに、線型計画法を用いれば、変数制約条件の下であるパス条件を成立させることのできる時間変数の最大/最小値を求めることも可能である。

3.5 Sアルゴリズムによる時間記号シミュレーション

3.2~3.3節では、Tアルゴリズムに基づく時間記号シミュレーションの計算法について述べたが、原理的にはSアルゴリズムに基づくシミュレーションも可能である。信号値やイベントの表現は、時刻が時間変数を含む式であることを除いて従来のSアルゴリズムと同様である。但し、シミュレーションの場合分けを行うので、現在の処理中の場合に対応するパス条件が記録される。また、イベントの管理に関しては、単なる集合を用いる。計算手順を(P1)~(P4)に示す。

(P1) 外部入力に対するイベントを登録する。

(P2) 登録されているイベントが無くなるまで、(P3)~(P4)を繰り返す。

(P3) 登録されているイベントの中で、パス条件と変数制約条件から時刻が最小と判定できるものがあれば、それを取り出す。最小と判定できるものが無ければ、場合分けを行って最小のものを決め、これを取り出す。場合分けの条件はパス条件に追加する。

(P4) (P3)で取り出したイベントの影響を受けるゲートの出力を計算し、変化があれば新しいイベントとして登録する。

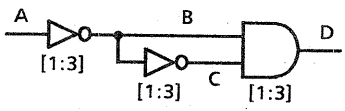
この計算手法では、(P3)における場合分けの数がTアルゴリズムによるものに比べて極めて大きくなるため、小さな問題に対しても現実的な時間で実行することは困難であると考えられる。しかし、このアルゴリズムはフィードバック・ループのある回路に対しても適用できるため、今後計算量を削減できる可能性について検討して行きたい。

4. 応用

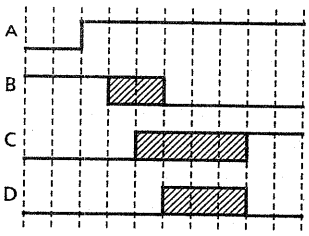
4.1 ハザードの検出

前節のシミュレーション手法を用いれば、ある組合せ回路にある入力変化を与えたときのハザードの有無を直ちに判定することができる。時間記号シミュレーションは、時間変数が与えられた変数制限条件のもとでとり得る全ての可能な連続値を尽くしているので、ハザードのハザードの有無を正確に判定できる。また、時間変数のとる値によってハザードの有無が変わる場合には、それぞれの条件を求めることができる。

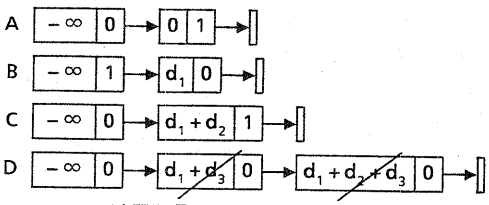
図4.1(a)は実際にはハザードが発生しないにも拘わらず、最大/最小遅延シミュレーションではハザードが有ると判定されてしまう(図4.1(b))例である。時



(a) 回路例



(b) 最大/最小遅延シミュレーション



(b) 時間記号シミュレーション

図4.1 ハザードの検出

間記号シミュレーションではこのような例も正確に扱うことができる(図4.1(c))。

4.2 非同期順序回路の検証

基本モード非同期順序回路が、与えられた状態遷移表通りの動作を行うかどうかを検証する問題を考える[7),8)]。但し、状態割当て及び状態変数と回路中の信号線との対応は既知であるものとする。本稿で述べたTアルゴリズムに基づくシミュレーション手法では、組合せ回路しか扱うことはできない。そこで、文献2)のようにフィードバック・ループを切断する方法により、検証を行う。

(V1) 状態変数に対応した信号線を切断することにより、回路を組合せ回路とみなす。

(V2) 状態遷移表より、切断した信号線に現れるイベント系列を求める。

(V3) 回路の外部入力に与えられるイベント系列、及び(V2)で求めた切断点のイベント系列を、(V1)により得た組合せ回路の入力として時間記号シミュレーションを行う。

(V4) (V3)の結果を、期待値である外部出力のイベント系列、及び切断点のイベント系列と比較することにより、回路が期待通りの動作をするかどうかを調べる。

例としてTフリップ・フロップの検証例を示す。図4.2の(a)、(b)はそれぞれ状態遷移表、回路の実現例であり、(c1)~(c4)は、Xへの入力及びZに期待される出力をタイムチャートで表したものである。状態変数に対応するY_1、Y_2を切断する。状態遷移表から(c1)~(c4)に対応してY_2に現れるべきイベントの系列を求める。ここでは、combinational hazard、sequential hazardが起きないY_2を求めている。また、Y_1は出力Zに一致するのでZに対する期待値イベント系列を用いる。こうして求めたイベント系列及びXに対するイベント系列(図4.2(d1)~(d4))を入力として時間記号シミュレーションを行う。シミュレーション結果は図4.2(e1)~(e4)のとおりである。それぞれを期待値である図4.2の(d1)~(d4)と照合することにより、次が導ける。

(R1) 回路が正常に動作する条件は、

$$d_3 \leqq d_1 + d_5 \leqq d_4$$

が成立する場合である。Y_1とY_2の求め方より、この条件が満たされていればcombinational hazard及びsequential hazardは発生しない。

(R2) このときのXからZまでの遅延t_1及びt_3は、

$$t_1 = d_1 + d_5 + d_7,$$
$$t_3 = d_4 + d_7,$$

である。

但し、(V4)の期待値との照合や結果の整理が3.4で述べた手法のみで行えるかどうかについては、現在検討中である。

5. おわりに

新しいタイミング検証のアプローチとして時間記号シミュレーションを提案し、その実現手法、応用の可能性について述べた。現在、本稿で述べた手法に基づいてシミュレータを試作中である。今後、シミュレーションの効率化、条件式の整理法、非同期回路の検証への適用法などの問題点について検討を進めていきたい。

謝辞

本学の安浦寛人助教授、平石裕実助教授、高木直史氏、高橋瑞樹氏、湊真一氏を始め、御討論頂いた矢島研究室の諸氏に感謝します。

参考文献

1) M. A. Breuer, A. D. Friedman : *Diagnosis & Reliable Design of Digital Systems*, p.308, Computer Science Press (1976).

2) 木村晋二, 羽根田博正: 系列集合論理シミュレータを用いた論理回路の設計検証について, 第34回情処全大7F-6, pp. 2013~2014 (March 1987).

3) W. C. Carter, W. H. Joyner and D. Brand: Symbolic Simulation for Correct Machine Design, *Proc. 16th DA Conf.*, pp. 280~286 (1979).

4) 北嶋雅哉, 高木直史, 矢島脩三: 論理関数のグラフ表現を用いた記号シミュレーション, 本誌 (Dec. 1987).

5) 石浦菜岐佐, 安浦寛人, 矢島脩三: 時間優先評価アルゴリズムによる論理シミュレーションの高速化, 情処論文, Vol. 26, No. 3, pp. 459~466 (May 1985).

6) 安浦寛人, 石浦菜岐佐: ハザード検出問題の計算複雑さについて, 信学技報COMP86-84, pp.29~37 (Jan. 1987).

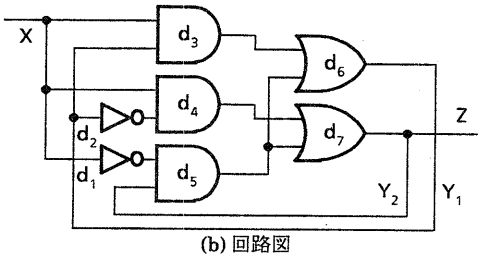
7) S. H. Unger: *Asynchronous Sequential Switching Circuits*, p.290, Wiley-Interscience (1969).

8) D. Mange: Analysis and Synthesis of Logic Systems, p. 416, Artech House (1986).

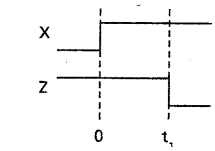
Y_1 Y_2	$Y_1'Y_2'$ X=0	$Y_1'Y_2'$ X=1	Z'
0 0	0 0	0 1	0
0 1	1 1	0 1	1
1 1	1 1	1 0	1
1 0	0 0	1 0	0

は安定状態を表す

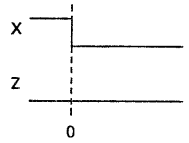
(a) 状態遷移表



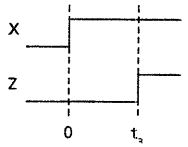
(b) 回路図



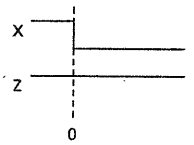
(c1) タイムチャート



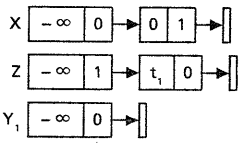
(c2) タイムチャート



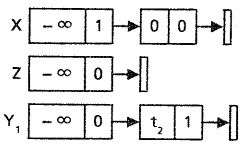
(c3) タイムチャート



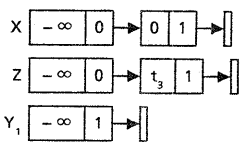
(c4) タイムチャート



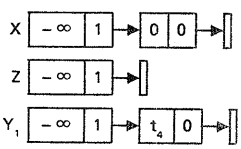
(d1) 入力及び期待値



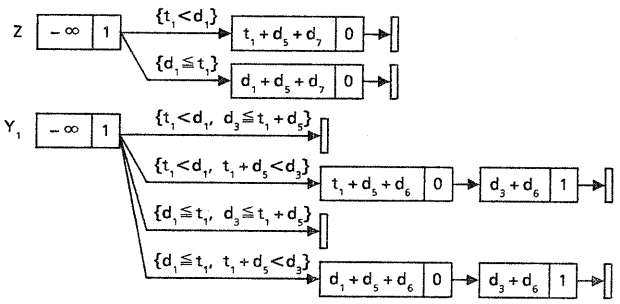
(d2) 入力及び期待値



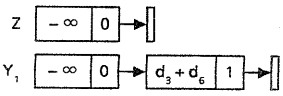
(d3) 入力及び期待値



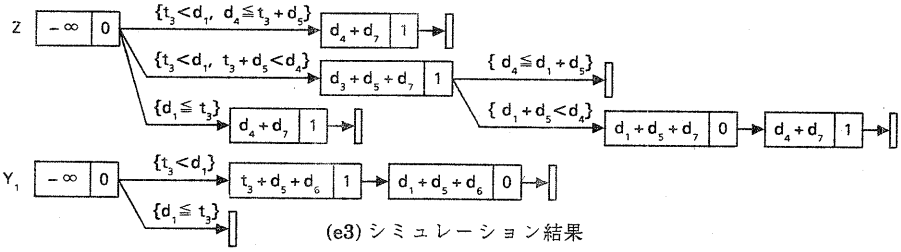
(d4) 入力及び期待値



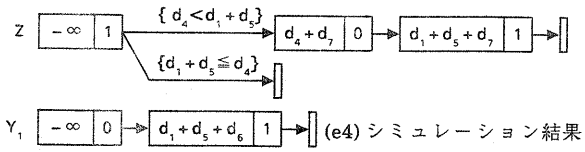
(e1) シミュレーション結果



(e2) シミュレーション結果



(e3) シミュレーション結果



(e4) シミュレーション結果

図4.2 Tフリップ･フロップの検証