Vol. 35 No. 8　　情　報　処　理　　Aug. 1994

解　説

論理設計の形式的検証

3. 形式的タイミング検証について†

木　村　晋　二††

1. はじめに

近年の論理素子技術の進歩により，高速なクロックで動作する論理回路が構成されるようになってきた．このため，論理回路の遅延が目標とするクロックに適合しているかどうか，またクロック同期におけるタイミングに関する制約が満たされているかどうかを検証することが，正しい回路を設計する上で重要な問題となっている．

この問題は，論理回路を構成する論理素子の動作において，入力変化が出力に伝わるのに，時間が必要であることに起因する．素子が遅延をもつことにより，それらから構成された論理回路の外部入力の変化が外部出力に伝わるまで，各素子の遅延時間の総和に応じた時間が必要となる．

レジスタとレジスタの間を論理素子を用いて結合したクロック同期論理回路の場合は，レジスタ間の回路部分の最大遅延時間以上のクロック周期を用いなければならない．とくに，論理回路の合成系が進歩し，回路のクロックを合成系が生成した回路の最大遅延で決定するということも行われているので，回路の遅延時間の正確な評価は非常に重要である．

一方，レジスタなどの記憶素子は，内部に論理素子のループを含み，回路が正しく動作することを検証するためには，どの経路が実際に活性化され，その遅延時間が制約を満たしているかどうかを解析しなければならない．また，これらの素子を使用する上で，クロック変化の直前および直後にデータ入力が変化してはならないなどの種々の制約がある．これらの検証では，最大遅延を与える経路を求めるよりも精密な解析が必要である．

本稿では，このような論理回路のタイミングに関する性質の解析・検証手法について述べる．まず，組合せ論理回路の最大遅延時間の評価手法と，タイミングシミュレーション手法について簡単に触れた後，種々の遅延モデルについて述べる．ついで，論理回路の実際には活性化されない経路（フォールスパス）を排除する手法[1)~5)]について解説した後，正確な遅延の検証を行うために提案されている時間記号シミュレーション手法[6),7)]，正則表現で表される時系列集合を扱うタイミング検証手法[8),9)]，実時間時相論理を用いたタイミングの記述と検証[10),11)]などについて解説する．

現状では，経路解析に基づく手法は数千ゲート程度の回路に適用できているが，正確なタイミング解析・検証手法は100ゲート程度の回路にしか適用できていない．アルゴリズムの改良が望まれる．

2. 論理回路の遅延とその解析

2.1 タイミング解析

図-1 に示す回路を考える．各ゲートの遅延時間を1とすると，C→H→K→L→M の経路が遅延時間4で最大である．この回路のように，フィードバックループがない場合には，最大遅延を与える経路を見つけることは，素子間の結線数（素子数の二乗程度）に比例した時間でできる．方法は，以下のとおりである．

1. 前方探索

（a） 外部入力 A, B, C, D, E, F に遅延値0を与える．

（b） 外部入力のみに結合されている素子G，Hに遅延値1を与える．これは，それらの素子の入力の遅延値の最大値0に素子の遅延値1を加えたものである．

† Formal Timing Verification of Logic Circuits by Shinji KIMURA (Graduate School of Information Science, Nara Institute of Science and Technology).

†† 奈良先端科学技術大学院大学情報科学研究科

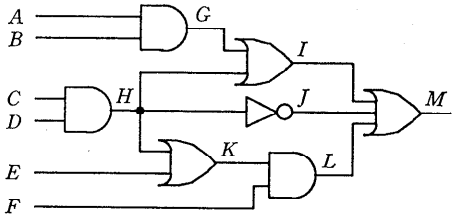


図-1 An example of a logic circuit.

(c) 遅延値1以下の素子のみを入力としてもつ素子 I, J, K の遅延値を2とする.

(d) 遅延値2以下の素子のみを入力としてもつ素子Lの遅延値を3とする.

(e) 遅延値3以下の素子のみを入力としてもつ素子Mの遅延値を4とする.

2. 後方探索

Mから, 素子の入力で最大の遅延値をもつものを順にたどる. これにより, M, L, K, H, C (D) という経路が得られる.

上記の前方探索の部分は, 外部入力から順に遅延の値が決まってゆく様子を表している.

論理回路のタイミングシミュレーションにおいても, 上記の前方探索とほぼ同じ処理を行う. ただし, 伝えるのは遅延値ではなく, 時間と値の対で表されたイベントである. 図-1 において, 時刻0で外部入力 (A, B, C, D, E, F) が (0, 0, 0, 0, 1, 1) から (0, 0, 1, 1, 0, 1) に変化したとする. イベントを端子名, 時間, 値の3項組みで表し, 初期値を与える時間を便宜上$-\infty$とする.

1. Gに関して, 入力 A, B のイベントは (A, $-\infty$, 0) (B, $-\infty$, 0) のみであるので, G のイベントも (G, $-\infty$, 0) のみとなる. $-\infty$には素子の遅延時間1を加えても$-\infty$としている.

2. Hに関して, 入力 C, D のイベントは (C, $-\infty$, 0) (C, 0, 1) (D, $-\infty$, 0) (D, 0, 1) であり, Hのイベントは, (H, $-\infty$, 0) (H, 1, 1) となる.

3. Iについては, 素子の入力 G, H のイベントが (G, $-\infty$, 0) (H, $-\infty$, 0) (H, 1, 1) であるので, (I, $-\infty$, 0) (I, 2, 1) となる.

4. 以下同様に, J, K, L, M についてイベントを計算すれば良い. 素子の遅延時間を加えてゆくことにより, 各素子の出力の変化時間が決まってゆく.

上記の処理で, ある素子のイベントをまとめると, 1次元の時間軸方向に並んだリストになる

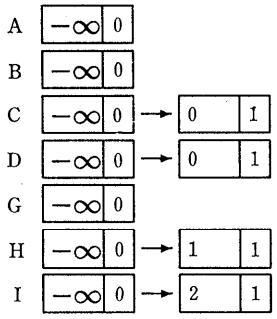


図-2 Event lists.

(図-2). 遅延のモデル, 出力の計算方法により, このようなイベントリストが陽には生成されないこともあるが, 概念的にはこのようなイベントリストの生成が基本である.

2.2 論理素子の遅延モデル

前節では, 論理素子の遅延時間を1としていたが, 素子の遅延モデルには以下で述べる種々のものがある[12),13)]. 現在通常のタイミングシミュレーションでは立ち上がり・立ち下がり遅延モデルが用いられ, 正確なタイミング解析・検証では遅延時間に幅があるあいまい遅延モデルあるいはより精密な最小・最大遅延モデルが用いられる. LSI においては, 配線遅延を考慮する必要があるが, それも素子の出力の遅延に含めて扱われることが多い.

1. 純粋遅延モデル (pure delay)

入力変化が遅延時間だけ遅れて出力にそのまま現れるもの.

2. 慣性遅延 (inertial delay)

遅延時間よりも短い入力変化は出力に現れないもの. これは, 素子を電気回路としてみたときに, 出力変化があるしきい値を越えるまでにまたもとのレベルへ戻ることをモデル化している.

3. 立ち上がり・立ち下がり遅延 (rise fall delay)

出力の立ち上がり (0→1) と立ち下がり (1→0) で異なる遅延時間を指定するもの. これは, 実際の素子の遅延時間が変化により異なることをモデル化している.

4. あいまい遅延 (umbiguity delay, bounded delay)

遅延時間として最小値と最大値の二つを指定し, 最小遅延時間と最大遅延時間の間は出力が不定であるとするもの. これは, 素子の遅延時間が製造

時の状況により異なり，ある幅の中でしか正確には指定できないことを表したものである．最小遅延時間と最大遅延時間の間を不定値にするのは，出力電圧が論理レベルの0と1のしきい値付近の場合はどちらともいえないということをモデル化している．

5. 最小・最大遅延 (min-max delay)

遅延時間として最小値と最大値の二つを指定するのはあいまい遅延と同じであるが，最小遅延時間と最大遅延時間の間は出力が不定ではなく，その間のどこかで変化しているとするもの．あいまい遅延では，解析において，不定値の部分が増殖してしまうので，悲観的すぎる結果になることがある．

これらの遅延モデルのうち，最小・最大遅延モデル以外は，一つの入力パターンに対する出力パターンは一つである．これら出力が一つとなる遅延モデルの入出力関係を**図-3** に示す．

一方，最小・最大遅延モデルでは，一つの入力パターンに対し複数の出力パターンが生じる．さらに，最小・最大遅延モデルを特徴づけるパラメータとしては，以下に示す二つのものがある．

1. 離散時間 (Discrete time, D) か，実数時間 (Real time, R) か．

2. 入力変化に対して遅延が固定 (Fix, F) であるか，可変 (Variable, V) であるか．

遅延モデルとしては，これらの組合せにより，DF, DV, RF, RV の4とおりが考えられる．

離散時間はある単位時間を仮定してその整数倍で遅延時間を表すものであり，実数時間は実数値で遅延時間を表すものである．離散時間の場合には，単位時間の整数倍以外の時間での変化は許されないが，実数時間の場合はその限りでない．なお，実数時間の場合でも遅延の値は有限記述できるという意味で，適当な単位時間を仮定して整数で表される．

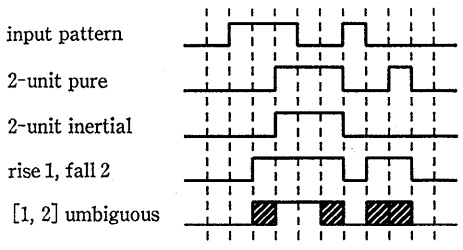


図-3 Delay models.

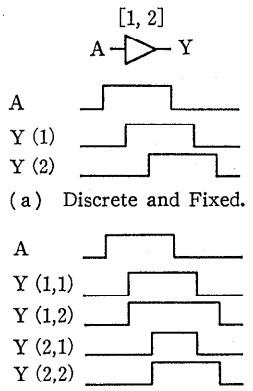


図-4 Discrete min-max timing models.

たとえば，最小1，最大2の遅延素子に長さ3のパルスが入力された場合を考えると，**図-4** に示すように DF では2とおりの出力しか出ないが，DV では入力パルスの各エッジで遅延時間が異なり得るので，4とおりの出力が出て，パルスの幅も変化する．また，RF や RV では，遅延の値が連続的に1と2の間のどのような値でもとれるので，無限の場合が生じる．

以下では，最小 t，最大 T の遅延を $[t, T]$ と記述する．

3. フォールスパス解析

論理回路の最大遅延時間は先に述べたように比較的簡単に求めることができる．しかし，文献1) などで，桁上げをスキップする加算回路のように，最大遅延を与える経路が実際には活性化されない場合があるということが指摘され，それ以後，実際に活性化される経路の中の最大のものを見つけるための研究が行われている[2)~5)]．

論理回路の外部入力 I から外部出力 O への経路に対し，その経路の先頭の I の信号変化が経路の終端の O の直接の信号変化となるように I 以外の外部入力を設定できるとき，その経路は静的に活性化されているという．一方，外部入力の値の組をある値から別の値へ変化させたとき，最終的に回路の外部出力が安定するときの最後の変化がある経路を通っているとき，その経路は動的に活性化されているという．静的には活性化できない場合でも動的には活性化できることがある．

経路の静的な活性化の条件について述べる．**図-5** の回路の $C \rightarrow H \rightarrow K \rightarrow L \rightarrow M$ の経路を静的

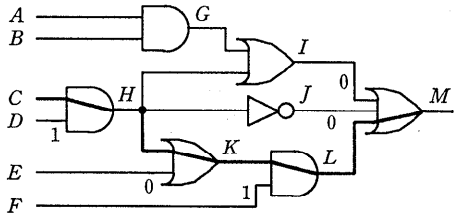


図-5 An example of the path sensitization.

に活性化するためには，経路内の各論理素子について，その経路以外の入力を経路上を変化が伝わるのを妨げないような値に設定すれば良い．そこで各素子についてみると，H の AND ゲートに関しては $D=1$, K の OR ゲートに関しては $E=0$, L の AND ゲートに関しては $F=1$, M の OR ゲートに関しては $I=J=0$ となる．これを外部入力側へたどって，このような値に設定できるかどうかを判定する．判定に当たっては，上記の条件を外部入力をもとにした論理式で表し，すべての条件を満たす外部入力の組があるかどうかを調べれば良い．これには BDD (二分決定グラフ) が有効である[14]．ちなみに，$C\rightarrow H\rightarrow K\rightarrow L\rightarrow M$ の場合は，I と J を同時に 0 とすることができず，静的に活性化できない．

一方，一時的に経路の活性化条件が成立する場合に経路が動的に活性化される．動的に活性化できる経路をもつ回路の例としては，**図-6** のようなものがある．各論理ゲートの遅延を 1 とすると，$A\rightarrow C\rightarrow E\rightarrow G$ または $B\rightarrow D\rightarrow F\rightarrow G$ の経路が遅延 3 で最長である．静的にはこれらの経路を活性化できないが，A, B に同時に立ち上がるような入力を入れると，これらの経路が活性化され，G にパルスが出力される．

このような動的な経路の活性化では，経路の活性化の条件の計算で，伝わってくる信号の値と伝わるまでの遅延を考慮して，活性化のための条件を緩めるということが行われている．たとえば 2 入力 AND ゲートに対しては，一方の入力が 0 であるならば最初に 0 になるほうの遅延がゲートの出力の遅延に直接影響を与え，両方とも 1 であるならば最後に 1 になるほうが影響を与える．一方の入力が着目している経路上にあるとして，それが 0 で確定するときにこの経路が活性化されるためには，そのときの遅延時間を t_0 として，他方が 0 となるまでの時間 t_0' が $t_0'>t_0$ でなければ

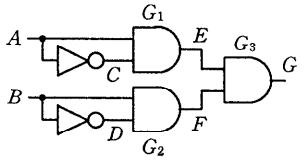


図-6 An example of false path.

ならない．また，1 で確定する場合にこの経路が活性化されるためには，そのときの遅延時間を t_1 として，他方が 1 となるまでの時間 t_1' が $t_1'<t_1$ でなければならない．経路上の各ゲートについてこのような条件を求めて，外部入力側へたどって，これらの条件を満たす入力の組があるかどうかを判定するのが，動的な経路の活性化の判定である．

図-6 の例では，$F=1$ の条件に対して，G_2 の D のほうの経路を活性化するには，B を D よりも先に 1 にするか，D を B よりも先に 0 にするかである．B の 0 から 1 への変化はこれに該当するので，F は 1 にできると判定される．G_1 でも同様である．

以上の経路の活性化の判定は，経路を指定した上での手法である．よって，実際の回路への適用では，

1. 論理回路の最大遅延を与える経路をすべて求める，
2. 上で求められた各経路に対し，活性化できるかどうかを判定する，

の二段階で行われる．ISCAS '89 ベンチマークへの適用に関しては，ほとんどのもので最大経路は活性化できないことが報告されている[5]．

4. タイミング解析の精密化

先に示したタイミングシミュレーション手法では，時間と値の組みで表されるイベントの一次元の列を扱っていた．この手法は，純粋遅延モデル，慣性遅延，立ち上がり・立ち下がり遅延，あいまい遅延には対応しているが，最小・最大遅延には対応していない．最小・最大遅延に適用するためには，回路内の各ゲートの遅延時間について，最小と最大の間のすべての場合の組合せを考えて，一つの入力に対して，そのすべての組合せの場合でシミュレーションしなければならない．これは，各ゲートの遅延の場合が 2 とおりずつだ

としても，各入力パターンごとで指数に比例したオーダになり，実際的でない．また，シミュレーションは与えられた入力に関してのみタイミング解析を行う手法で，タイミング検証のように，すべての入力に関してある性質が満たされるかどうかを判定するには適していない．

以下では，最小・最大遅延を扱うためおよびタイミング検証を行うためのイベントモデルの拡張について述べる．なお，ここでは特に最小・最大遅延に着目するが，最小・最大遅延が扱えれば，これと，慣性遅延，立ち上がり・立ち下がり遅延などを組み合わせたモデルも取扱い可能であり，拡張性は非常に高い．

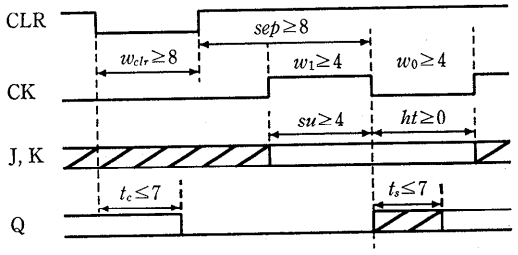
4.1 タイミング検証の例

ここでは，タイミング検証の例として，**図-7** に示すエッジトリガ型の JK フリップフロップの仕様記述と検証について考える．

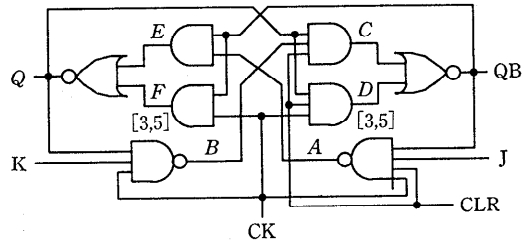
JK フリップフロップの動作は，動作を表す **表-1** と，**図-7**（a）のタイミング仕様により決められる．動作表には

1. クリア入力 CLR により出力 Q が 0 になること．
2. クロック CK の立ち上がりでは Q は変化しないこと，
3. クロック CK の立ち上がりで，Q が入力 J, K の値に応じた値になること

が示されている．



（a） Timing specification.



（b） An implementation.

図-7 A negative-edge-triggered JK flip-flop.

表-1 フリップフロップの動作表

CLR	CK	J	K	Q	next Q
0	X	X	X	X	0
1	up	X	X	0	0
1	up	X	X	1	1
1	down	0	X	0	0
1	down	1	X	0	1
1	down	X	0	1	1
1	down	X	1	1	0

また，図-7（a）のタイミングチャートには，

1. CLR の 0-パルス幅が 8 以上であること，
2. CLR が 0 になってから 7 以下で出力 Q が 0 になること，
3. CLR と CK の立ち下がりが 8 以上離れていること，

などが示されている．

図-7（b）の実現では，NAND ゲートの遅延時間を [3, 5] とし，それら以外のゲートの遅延時間を 1 としている．回路の検証を行うためには，

1. これらの仕様を記述し，
2. 回路が仕様を満たしているかどうかを判定しなければならない．

以下種々の手法で，検証をどのようにして行うかを示す．

4.2 時間記号シミュレーション

4.2.1 時間記号シミュレーション：実数時間での解析

時間記号シミュレーション手法[6] は論理回路の中の素子の遅延を記号として取り扱うもので，与えられた入力パターンに対して，素子の遅延の関係（一次不等式）に応じた出力パターンを出力する．時間記号シミュレーションで扱っているタイミングモデルは，先の分類では RF（実数時間で固定）である．

時間記号シミュレーションでは，各論理素子の値を表すのに，イベントリストに条件判定を入れて拡張したイベント木と呼ばれる有向グラフを用いる．イベント木の節点は，時刻とそのときの端子の値を表すイベント節点と，遅延記号の間の一次不等式を表す条件節点からなる．

図-8 に，簡単な回路の例とその解析結果のイベント木の例を示す．外部入力 A, B には，おのお

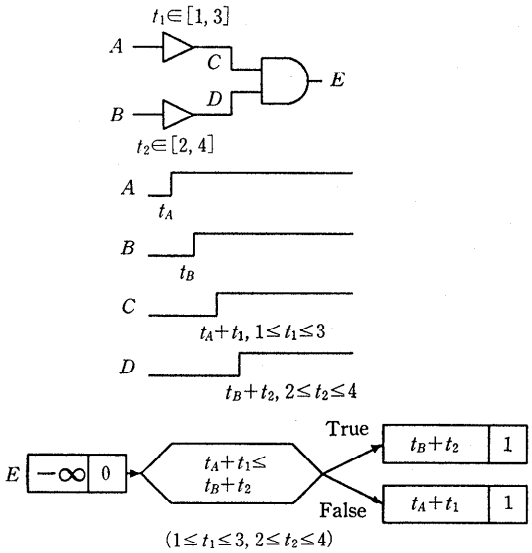


図-8 An example of event tree.

の時刻 t_A, t_B で立ち上がる入力を与える．すると，遅延素子の出力の C, D には，立ち上がり時間がおのおの t_A+t_1, t_B+t_2 のパターンが現れる．AND ゲートの解析では，C, D の立ち上がり時刻の大小で場合分けを行い，図に示すようなイベント木を得る．

JK フリップフロップの検証手法について述べる．本手法での解析は，基本的に一つの与えられた入力系列に対して回路の遅延の解析を行うものであるので，まず，回路の仕様から適当な入力パターンを抽出する必要がある．そして各入力パターンに対し，以下の処理を行う．

1. 時間を記号化した入力パターンを記述する．たとえば，CLR の初期値を 1 として，適当な時間 t_1 で 0 にして t_2 $(t_2 \geq t_1+8)$ に 1 とするような記述である．

2. 上記の入力に対してシミュレーションを行う．

3. 結果のイベント木をたどり，すべての経路で t_1+7 時刻以下に Q が 0 となっているかどうかを調べる．

本手法では，上記のように，入力パターンごとにシミュレーションを行うので，入力パターンの網羅性に問題がある．文献6)には，この手法を用いたハザード検出や非同期順序回路の検証例が示されている．

4.2.2 符号化時間記号シミュレーション：離散時間での解析

文献7)では，離散時間 (DF) での論理回路の高速な処理を行う手法として，符号化時間記号シミュレーション手法が提案されている．内部処理には BDD (二分決定グラフ) が用いられ，100 ゲート程度の組合せ論理回路の，与えられた一つの入力系列に対する解析が実際的な時間で処理できることが示されている．

符号化時間記号シミュレーションでは，遅延素子の遅延のばらつきを図-9 に示すように個別の遅延素子とセレクタによりモデル化している．回路の出力は，与えられた外部入力に対して，新たに導入されたセレクタの選択用の変数の関数として表される．

たとえば，図-9 で入力Aに時間 0 で 0 から 1 に変化するようなパターンを入れたときには，Y のイベントリストは，図に示すように (Y, $-\infty$, 0) (Y, 1, $\overline{T}\cdot 1 + T\cdot 0$) (Y, 2, 1) となる．時間 1 では，$T$ の値に応じて 0 または 1 になる．

図-7 に示す JK フリップフロップの検証について述べる．この手法でも，選ばれた入力パターンに対して，回路のシミュレーションを行うことにより検証を行う．以下に各入力パターンに関する処理を示す．

1. 入力パターンを記述する．必要であれば，制御変数を入れて，時間を可変にする．たとえば，CLR の初期値を 1 として，時刻 5 で 0 にして時刻 13 で 1 とするような記述である．制御変数 U を一つ導入すれば，図-9 と同様に時刻 12 の値を $\overline{U}\cdot 0 + U\cdot 1$ のようにして，1 に変化する時刻を 12 と 13 にできる．

2. 図-7(b)の A, B の NAND ゲートは遅延が [3, 5] でとり得る値が 3, 4, 5 の 3 とおりであるので，おのおの 2 変数ずつの時間制御変数を入れ，シミュレーションを行う．

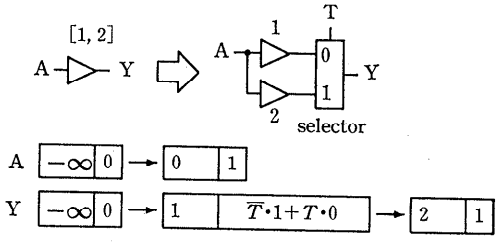


図-9 Time coded symbolic simulation.

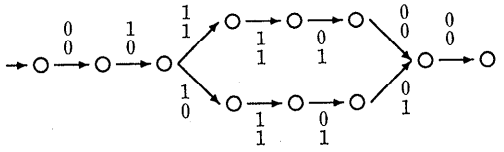
3. 結果のイベントリストをたどり，すべての経路で時刻 12 までにQが0となっているかどうかを調べる．

出力結果のイベントリストの値の論理式は，遅延の組合せに対する出力を表しているので，文献7)に示されているように，出力結果の論理式に対して処理を施すことにより，ハザード検出や遅延の大小関係を表す式などを求めることができる．本手法も入力パターンの網羅性に関して問題がある．

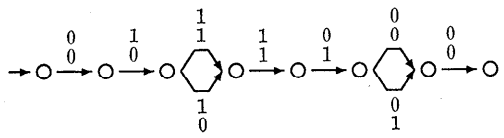
4.3 正則表現を用いたタイミング検証

正則表現論理シミュレーション手法[8),9)]は，論理回路の動作を，入力時系列の正則集合に対して模擬する方式である．最小・最大遅延における遅延時間のばらつきは，出力時系列の集合として扱われる．たとえば，図-4 の [1,2] の DF の遅延素子モデルの場合では，入力パルスに対して二つの出力があるが，それを二つの系列からなる集合として扱う．有限オートマトンで表すと，**図-10** (a)に示すようになる．この図は，遅延素子の入力として 0111000 を加えた場合で，枝に付けられたラベルは，遅延素子の入力と出力を縦方向に並べたものを表す．出力として，0011100 と 0001110 の二つの場合が表されている．外部入力系列に関しても集合を扱うことができるので，タイミングに関してすべての場合を尽くすことが容易である．

内部のデータとしては集合を扱っているが，論理回路の動作の解析では，集合としての演算のみではだめで，どの出力系列がどの外部入力系列に対して生じるのかが分からなければならない．そこで，論理回路中の素子の出力値の表現において，出力の値とそのときの外部入力の値を並べて表すという方式を用いている．



(a) Discrete and Fixed.



(b) Discrete and Variable.

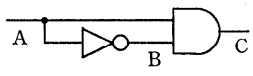
図-10 Finite automata representation of delay models.

図-11 に簡単な回路のシミュレーション例を示す．簡単のため，NOT ゲートの遅延を1，AND ゲートの遅延を1としている．入力としては，{0011, 1100} を加えている．図で端子Aの上部の系列集合は {(0,0)(0,0)(1,1)(1,1), (1,1)(1,1)(0,0)(0,0)} となっているが，これは，外部入力の値（この場合はA）と着目する端子の値（これもA）の値の対を表したものである．B, C についても同様である．X は不定値を表す．Bの入力 0011 に対する出力 X110 は1時刻遅れて値が決まることを表す．2入力 AND ゲートの出力集合の計算では，A, B の系列集合中の同じ外部入力系列に対応するものの間で AND 演算を行い，遅延をとって出力とする．外部入力が複数ある場合は，それらの値をすべて並べた後に着目する端子の値を並べたものを記号とする正則表現で系列集合を表す．

計算機上で正則集合を表すには有限オートマトンを用いている．また，同じ値が連続した系列を表すときには，値とその長さの組で表すようにしており，通常の論理シミュレーションのイベントリストに，分岐とループを入れて拡張したものとみることができる．分岐とループは，有限オートマトンの状態遷移の枝として表される．値とその長さの組で表すというのは，正則表現で 00000 を 0^5 と表すのに対応している．

有限オートマトンの取扱いにおいては，非決定的な状態遷移も扱うことができ，離散時間で可変な遅延モデル (DV) を，離散時間で固定の遅延モデル (DF) と同じ枠組で扱える．たとえば，図-4 の例では，各モデルの系列集合を有限オートマトンで表すと，図-10 のようになる．図は，遅延素子の入力として 0111000 を加えた場合で，枝に付けられたラベルは素子の入力と出力を縦に並べたものである．

{(0,0)(0,0)(1,1)(1,1),
(1,1)(1,1)(0,0)(0,0)}　　{(0,X)(0,0)(1,0)(1,1),
(1,X)(1,X)(0,0)(0,0)}



{(0,X)(0,1)(1,1)(1,0),
(1,X)(1,0)(0,0)(0,1)}

図-11 Simulation for a set of input strings.

実際のシミュレーションは，コンパイラ方式のシミュレーションと同様，以下の手順で行われる．

1. 素子を外部入力からの最大の段数で順序づける．

2. 決まった順序に従い，素子の出力集合を求める．

（a） 素子の入力に結合されている端子の値集合から，同じ外部入力系列に対応する系列を抽出する．

（b） 抽出された入力系列間で論理演算を行い，出力系列集合を求める．

（c） 求めた出力系列集合に対して遅延の演算を施す．

同じ外部入力系列に対応する系列の抽出や遅延の演算は有限オートマトンの直積演算を用いている．また，論理演算は，記号ごとの論理演算で行える．

JKフリップフロップの検証について述べる．まず，JKフリップフロップの仕様の各項目を正則表現で表す．たとえば，CLRの0パルス幅が8以上は，(CLR, CK, J, K, Q) の値の組を記号として

$$\{(0,X,X,X,X)^8(0,X,X,X,X)^* + (1,X,X,X,X)^*\}^*$$

という正則表現になる．ここで前半部はCLRの0の幅が8以上であることを表し，後半はCLRの1の幅が0以上であることを表す．また，Qが0で，Jが1のとき，CKの立ち下がりでQが(何時刻か遅れて) 1に変化するという仕様は，(CLR, CK, J, K, Q) の値の組を記号として

$$(1,1,1,X,0)(1,1,1,X,0)^*(1,0,X,X,0)^* (1,0,X,X,1)(1,0,X,X,1)^*$$

となる．これら，各項目を表す仕様の表す系列集合の共通集合をとることにより，仕様すべてを満たす系列集合が得られる．つぎに，この仕様に対して回路のシミュレーションを行い，Qに対応する出力集合を得る．この結果のQと，仕様で与えられたQを比較することにより，検証を行うことができる．

文献9)では，4ビットの加算器などで一回だけの任意の入力変化に対するハザード検出や，フリップ・フロップなどの非同期回路に対して仕様が満たされていることの検証を行った例が示されている．

仕様は，通常の有限オートマトンの形で与えることができるので，記述が簡単で，記述のもれが少ない．また，ハザード検出器なども有限オートマトンで簡単に表せることが多いので有用性が高い．さらに，より精密な遅延モデルも遅延演算を行う有限オートマトンを書き換えるだけでできるので，種々の遅延モデルを統合的にかつ容易に取り扱うことができる．

4.4 実時間時相論理による検証

実数時間を扱う論理体系の研究も盛んである．ここでは文献10)に従い，時相論理の一種であるCTL (Computation Tree Logic) の時相演算子に，時間の上下限を付加して拡張したTCTLについて解説する．

CTLではモデルとなる有限オートマトンの状態遷移に対し，すべての状態遷移に対して成立することを表す“∀”と，ある状態遷移に対して成立することを表す“∃”の二つの限定子を有している．また，時相演算子としては「いつか」を表す“◇”，「ずっと」を表す“□”，「ϕ_2 が成立するまでずっと ϕ_1 が成立する」を表す“$\phi_1 \mathrm{U} \phi_2$”(until) などがある．CTLの記述では，限定子と時相演算子を組み合わせて性質を表す．

TCTLでは，これらの時相演算子に<6 や≥ 2などの上下限を表す情報を付加する．たとえば，$\exists p_1 \mathrm{U}_{<3} p_2$ は，ある状態遷移があり，3単位時間までに p_2 が成立し，かつそれまでの間ずっと p_2 が成立することを表す．時間の上下限は整数で指定するが，意味自体は実数時間で与えられる．なお，離散時間で意味を与える場合もある．

文献10)では，TCTL式が，与えられたTimed Graphに対して満たされているかどうかを判定するモデルチェッキング法を提案している．Timed Graphは通常の有限オートマトンにクロック変数を導入し，状態遷移の枝にクロック変数のリセットやクロック変数に対する時間制約が記述できるようにしたものである．Timed Graphの例を**図-12**に示す．図中の○は状態を表し，○の左上のラベルは状態名を，○の中の p_1 などの記号は，各状態で成立する命題を表す．s_1 から s_2 への遷移では x という時間変数をリセットし，s_2 から s_3 への遷移では x に関して1以上2以下という条件を付けると同時に y という時間変数をリセットしている．

このTimed Graphに対して，$\exists p_1 \mathrm{U}_{<3} p_2$ とい

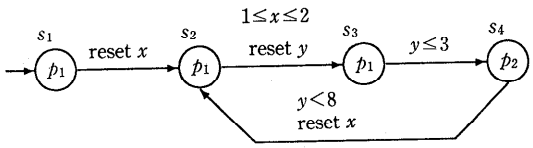


図-12 An example of a Timed Graph.

う TCTL 式を考える．初期状態 s_1 から始まる状態遷移で，

$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4$

について，s_2 から s_3 への遷移時間と s_3 から s_4 への遷移時間の和が3よりも小さい場合には $\mathrm{U}_{<3}$ の条件を満たし，かつそのような条件は x や y に付けられた条件と矛盾しない（たとえば x, y 両方とも1など）ので，一つでもそのような状態遷移があれば良いというこの TCTL 式は真となる．一方，TCTL 式 $\forall p_1 \mathrm{U}_{<3} p_2$ については，$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4$ という状態遷移で初期状態から p_2 が成立する s_4 へ行くまでに時間が3以上の場合を含む（x, y 両方が2の場合など）ので，すべての状態遷移に対して3よりも小さくなければならないというこの TCTL 式は偽となる．

真偽判定は，与えられた TCTL 式に対して，もとの Timed Graph を変形することによる．たとえば，図-12 と $\exists p_1 \mathrm{U}_{<3} p_2$ に対しては，p_1 が成立する各状態から時刻3までで行ける状態を求める必要がある．このとき，s_4 は，時刻3までで行ける場合と行けない場合とがあるので，状態を分けて新しい状態を作らねばならない．これにより一般に状態数は増加するが，最終的に作成されたグラフをたどることで，TCTL 式の真偽判定ができる．

JK フリップフロップの検証について述べる．この場合，各仕様に対し，TCTL 式を書く必要がある．たとえば，CLR の0パルス幅が8以上であるという仕様は，$\forall \Box(\overline{CLR} \rightarrow \overline{\exists \Diamond_{<8} CLR})$ となる．$\overline{CLR}$ は CLR の否定（CLR が0であること）を，→は「ならば」を，後半部はある状態遷移があって8までで CLR が1になることの否定を表す．

一方，図-7(b)に示す回路に対しては，各素子の特性をもとに Timed Graph を構成する．たとえば [3, 5] の遅延などは，**図-13** に示すようになる．図中の i は入力論理変数を，o は出力論理変数を表す．s_0 は入力と出力の両方が0である

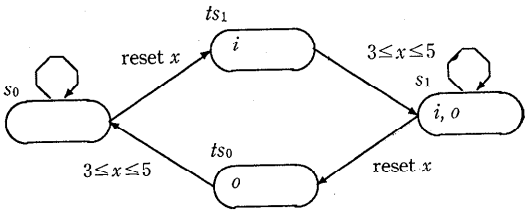


図-13 Timed Graph for a 3 input NAND gate with [3, 5] delay.

安定な状態である．s_0 で入力が変化すると x をリセットして一時的な状態 ts_1 へ遷移する．その後，3時刻以上5時刻以内に入力と出力の両方が1の安定な状態 s_1 へ遷移する．遅延付きの NAND ゲートの入出力応答なども同様に Timed Graph で記述できる．

最後に，個々のゲートの入出力関係を表す Timed Graph から，直積オートマトンの構成法の拡張を用いて全体の動作を表す Timed Graph を構成し，このモデルの上で，TCTL 式が成立するかどうかを判定する．

CTL でのモデルチェックの計算量は式のサイズとモデルのサイズの積に比例するが，TCTL では，式の中の時間制約の値に応じて Timed Graph の変形を行うので，式のサイズ，モデルのサイズ，式の中の定数の最大値，モデルのクロックに関する制約の定数の積，モデルのクロック変数の数の階乗をすべて掛け合わせた結果に比例することが示されている．また，離散時間上でモデルチェックを行う場合には，モデルのクロック変数の数の階乗の部分を減らしたものに比例した計算量でできることが示されている．

本手法は，モデルの構成の部分で状態数が爆発する可能性が大きく，実用的な回路に対してはまだ適用されていない．

5. おわりに

以上，形式的なタイミング検証手法について示した．これらの手法は，大きく分けて離散時間上で処理を行うものと，実数時間上で処理を行うものに大別できる．一般的に，離散時間上で処理を行う方式のほうが効率的であるが，解析の精度に関しては問題がある．

直感的には，離散時間の単位時間を十分細かくすれば，離散時間での解析結果と実数時間での解析結果が一致することが期待できる．事実，ハ

ザード検出問題では，文献15)に示されているように回路中の素子数の指数に比例した程度に単位時間を細かくすれば，離散時間での判定結果が実数時間での判定結果に一致することが示されている．ただし，この結果は最悪の場合を押さえたものであり，回路によっては，もっと粗い単位時間で離散時間での判定結果が実数時間での判定結果に一致する可能性がある．経路の遅延解析については，文献16)に離散時間の単位に関する結果が示されている．また，文献17)には，離散時間の単位を徐々に細かくして実数時間と同じだけの精度を得る手法が示されている．このような問題についても今後もっと研究が必要である．

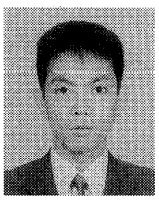
遅延時間に幅をもたせてすべての場合を尽くすことは計算量の点で困難な問題であるが，種々の効率的な手法が研究されており，今後の発展が期待される．

最後に，本資料をまとめる上でお世話になった京都産業大学平石裕実先生，ご討論いただいた富士通研究所藤田昌宏博士，論文などの資料を提供していただいた大阪大学石浦菜岐佐先生に感謝します．また，本稿を読んでコメントをいただいた本学川崎通君，および閲読者の方々に心より感謝します．最後に日頃からご討論いただく本学渡邊勝正教授ならびに神戸大学羽根田博正教授に感謝します．

参考文献

1) Benkowski, J., Meersch, E. V., Claesen, L. J. M. and De Man, H.: Timing Verification Using Statistically Sensitizing Path, *IEEE Trans. on CAD*, Vol. CAD-9, No. 10, pp. 1073-1084 (Oct. 1990).
2) Perremans, S., Claesen, L. and De Man, H.: Static Timing Analysis of Dynamically Sensitizable Paths, In *Proc. of 26th DA Conf.*, pp. 568-573 (June 1989).
3) Chen, H. C. and Du, D. H.: Path Sensitization in Critical Path Problem, In *Proc. of ICCAD-91*, pp. 208-211 (1991).
4) Devadas, S., Keutzer, K. and Malik, S.: Delay Computation in Combinational Circuits: Theory and Algorithms, In *Proc. of ICCAD-91*, pp. 176-179 (1991).
5) McGeer, P. C., Saldanha, A., Stephan, P. R., Brayton, R. K. and Sangiovanni-Vincentelli, A. L.: Timing Analysis and Delay-Fault Test Generation using Path-Recursive Functions, In *Proc. of ICCAD-91*, pp. 180-183 (1991).
6) 石浦菜岐佐，高橋瑞樹，矢島脩三：論理回路の正確なタイミング検証のための時間記号シミュレーション，情報処理学会論文誌，Vol. 31, No. 12, pp. 1832-1839 (Dec. 1990).
7) Ishiura, N., Deguchi, Y. and Yajima, S.: Coded Time-Symbolic Simulation for Timing Verification of Logic Circuits, *IEICE Transactions on Fundamentals*, Vol. E 75-A, No. 10, pp. 1247-1254 (Oct. 1992).
8) 木村晋二，羽根田博正，矢島脩三：正則表現論理シミュレーション手法に基づく非同期順序回路の検証，電子情報通信学会論文誌，Vol. J 71-D, No. 9, pp. 1786-1796 (Sep. 1988).
9) Kimura, S., Kashima, S. and Haneda, H.: Timing Verification of Logic Circuits with Combined Delay Model, *IEICE Transactions on Fundamentals*, Vol. E 75-A, No. 10, pp. 1230-1238 (Oct. 1992).
10) Alur, R., Courcoubetis, C. and Dill, D.: Model-Checking for Real-Time Systems, In *Proc. of 5th Logic in Computer Science*, pp. 414-425 (June 1990).
11) Harel, E., Lichtenstein, O. and Pnueli, A.: Explicit Clock Temporal Logic, In *Proc. of 5th Logic in Computer Science*, pp. 402-413 (June 1990).
12) Breuer, M. A. and Friedman, A. D.: *Diagnosis & Reliable Design of Digital Systems*, Computer Science Press (1976).
13) Seger, C.-J. and Brzozowski, J.: Advances in Asynchronous Circuit Theory, Part II, In *EATCS*, pp. 199-263 (1991).
14) 藤田昌宏, Clarke, M.: BDD の CAD への応用，情報処理, Vol. 34, No. 5, pp. 609-616 (May 1993).
15) Ishiura, N. and Yasuura, H.: On a Relation between Time-Models and Computation Time of Hazard Detection Problem, In *IEICE Tech. Report, COMP 88-21* (1988).
16) McGeer, P., Saldanha, A., Brayton, R. and Sangiovanni-Vincentelli, A.: Delay Models and Exact Timing Analysis, In *T. Sasao Ed., Logic Synthesis and Optimization, Kluwer Academic Publishers*, pp. 76-83 (1993).
17) Kimura, S., Tsubota, S. and Haneda, H.: Preciseness of Discrete Time Verification, *IEICE Transactions on Fundamentals*, Vol. E 76-A, No. 10, pp. 1755-1759 (Oct. 1993).

(平成6年1月5日受付)



木村 晋二（正会員）

1959年生．1982年京都大学工学部情報工学科卒業．1984年同大学院修士課程修了．1985年同大学院博士課程退学後，神戸大学工学部助手．1989年京都大学工学博士．1993年より奈良先端科学技術大学院大学情報科学研究科助教授．論理回路のタイミング検証，形式的論理設計・検証，並列CADアルゴリズムの研究に従事．電子情報通信学会，IEEE，LAシンポジウム各会員．