有限オートマトンと表現等価な正則時相論理とその論理設計検証への応用†

平石裕実†† 浜口清治††
藤井寛†† 矢島脩三††

超大規模集積回路技術の進展に伴い，設計対象となる論理回路の規模もますます増大化し，設計段階において設計誤りの生じる可能性が高くなってきている．このため，設計の正しさを保証するための論理設計検証手法の確立が緊急かつ重要な課題となっている．論理設計検証においては，論理回路を記述する能力のある論理体系が必要である．従来より，論理回路の一部である組合せ論理回路に対応する論理体系として命題論理が知られているが，一般的な論理回路である順序回路やその数学的モデルである順序機械や有限オートマトンにちょうど対応する論理体系を明らかにすることが重要である．そこで，本論文ではこのような観点から，有限オートマトンの入出力系列に着目し，有限オートマトンと等価な表現能力を持ち，しかも時間の概念を陽に表現できる正則時相論理の体系を示し，さらに正則時相論理を用いた順序機械の設計検証アルゴリズムを示す．このアルゴリズムは正則時相論理式の長さに対しては非初等的であるものの構造モデルの大きさに対しては線形時間で動作する効率の良いもので，実際にこのアルゴリズムを用いたモデルチェッカを作成し，実用規模の順序機械に対して，実用的な時間と記憶量で設計検証が行えることを実証する．

1. はじめに

超大規模集積回路技術の進展に伴い，設計対象となる論理回路の規模もますます増大化し，設計段階において誤りの生じる可能性が高くなってきている．設計誤りが生じると，製品のコストや開発期間等に重大な悪影響を及ぼすため，設計の正しさを保証するための論理設計検証手法の確立が緊急かつ重要な課題となっている．

従来，論理設計の正しさを確かめるための一手法として，論理シミュレーションによる方法が広く用いられているが，論理シミュレーションではシミュレーションで用いた入力パターン以外の入力に対する設計の正しさを保証することはできない．このため，設計の正しさを保証するためには，何らかの論理体系を用いて設計の正しさを証明する必要があり，設計対象となる論理回路の仕様や設計を記述し設計検証を行うための論理体系が必要である．

従来より，論理回路の一部である組合せ論理回路に対応する論理体系として命題論理が知られているが，一般的な論理回路である順序回路やその数学的モデルである順序機械や有限オートマトンにちょうど対応し有限個の論理記号で構成できる論理体系は明らかにされておらず，このような論理体系の構築が論理設計検証の分野を始めとして並行プロセスやプロトコルの検証，ひいては数理論理学の分野においても1つの重要な研究課題となっていた．

設計検証のための論理体系として，時間の概念を陽に表現することができる時相論理[10]は，近年プロトコルや論理回路の形式的設計検証の分野において注目されており，実用的な設計検証システムの研究開発も試みられている[1)~5),11)]．しかしながら，これらのシステムで用いられている時相論理は表現能力が低く任意の有限オートマトンの設計仕様を記述することはできない．そこで，時相論理の表現能力を拡張するいくつかの試みが行われている．Wolper らは右線形文法やBüchi オートマトンに対応した時相演算子を導入することにより有限オートマトンと等価な表現能力を実現した[12),13)]が，ある有限オートマトンの仕様を記述する場合にはそのオートマトンに対応した時相演算子を用いる必要があるので，一般に任意の有限オートマトンの仕様を記述するためには無限個の時相演算子が必要となる．一方，Moszkowski はインターバル時相論理 (ITL)[9] を提案しているが，ITL は表現能力が高すぎて充足可能性判定問題が決定不能となるため，設計検証のための論理体系として用いることはできない．

そこで我々は，このような観点から有限オートマトンにちょうど対応する論理体系として，有限オートマトンの入出力系列（動作系列）の集合と考えることの

† Regular Temporal Logic Expressively Equivalent to Finite Automata and Its Application to Logic Design Verification by HIROMI HIRAISHI, KIYOHARU HAMAGUCHI, HIROSHI FUJII 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

できる正則集合に着目し，正則集合やその部分クラスであるεフリー正則集合，また無限長の系列も考慮した∞正則集合やω正則集合などに対応するいくつかの時相論理体系（正則時相論理）を提案してきた[6),14),17)]．正則時相論理は，命題論理に時間を表現するための時相論理記号として，「次の時刻」，「次の区間」，「繰り返し」を意味する3つの時相論理記号を加えて構成されており，比較的単純な構成ではあるが有限オートマトンと等価な表現能力を持っている．したがって，正則時相論理を用いることにより，設計対象である論理回路が満たすべき設計仕様を，その論理回路の入出力の関係として完全に記述することが可能になる．

本論文ではこれらの正則時相論理を整理するとともに，正則時相論理を実際の論理設計検証問題に応用するために，設計された有限オーマトンが正則時相論理で記述された設計仕様を満たすことを証明する方法として，モデルチェックと呼ばれる効率の良いアルゴリズムを示し，このアルゴリズムに基づいた論理設計検証システムを実際に試作し，状態数が数百程度の中規模の順序機械に対して実用的な時間で設計の正しさを証明できることを示す．

以下，2章では必要な用語や記法の定義を行い，有限オートマトンの有限長や無限長の動作系列に対応して4種類の時相論理を定義する．3章ではこのうちのεフリー有限正則時相論理を用いた順序機械の設計検証への基本的アプローチについて述べ，4章で検証アルゴリズムとしてモデルチェックアルゴリズムを示しその計算量や計算の複雑さについて議論する．5章では試作した検証システム（モデルチェッカ）による設計検証例を示し，本論文で提案する検証法の評価を行う．6章では本論文の主要結果をまとめる．

2. 正則時相論理

2.1 諸 定 義

本節では，正則時相論理の定義に必要となる基礎的な用語や表記法について説明する．

アルファベット Σ の記号を有限個（1個以上）並べた系列を Σ 上の有限長語と言い，Σ の記号を可算無限個並べた系列を Σ 上の無限長語という．また，記号を1つも含まない長さが0の系列を空語といい，ε で表す．Σ 上のすべての有限長語の集合を Σ^+，Σ 上のすべての無限長語の集合を Σ^ω で表し，$\Sigma^*=\Sigma^+\cup\{\varepsilon\}$，$\Sigma^\infty=\Sigma^*\cup\Sigma^\omega$ と定義する．Σ^∞ の要素を Σ 上の語と呼ぶ．

Σ 上の語 x に対して，$|x|$ で x の長さを表し，$x(i)$ は x の i 番目の記号，x^i は x の i 番目以降の部分系列を表す．$x^1=x$ である．とくに，$|\varepsilon|=0$，無限長語 x に対して $|x|=\omega$ とする．また，$i>|x|$ なる i に対して，$x(i)=\varepsilon$，$x^i=\varepsilon$ とする．

Σ 上の語 x と y の連接 xy は次のように定義される．

1. $\varepsilon x=x\varepsilon=x$.
2. $x\in\Sigma^+$, $y\in\Sigma^+\cup\Sigma^\infty$ の場合，
   - $1\le i\le|x|$ なる任意の整数 i に対して，$xy(i)=x(i)$.
   - $1\le i$ なる任意の整数 i に対して，$xy(|x|+i)=y(i)$.
3. $x\in\Sigma^\infty$ の場合，$xy=x$.

Σ 上の語の集合を Σ 上の言語という．Σ 上の言語 L_1 と L_2 の連接 L_1L_2 は $L_1L_2=\{xy\mid x\in L_1, y\in L_2\}$ と定義する．また，Σ 上の言語 L の連接に関する種々の閉包 L^+，L^*，L^ω，L^∞ を各々 $L^+=\cup_{i=1}^{\infty}L^i$，$L^*=\cup_{i=0}^{\infty}L^i$，$L^\omega=\{x_1x_2\ldots\mid x_1, x_2, \ldots\in L\cap\Sigma^+\}$，$L^\infty=L^*\cup L^\omega$ と定義する．ここに $L^0=\{\varepsilon\}$，$L^{i+1}=L^iL(i\ge0)$ である．

定義1（正則集合） Σ 上の言語の族 $\mathcal{R}$ に対して次の生成規則を考える．

- A1. 空集合 $\emptyset$ は $\mathcal{R}$ に属する．
- A2. $s\in\Sigma$ のとき，集合 $\{s\}$ は $\mathcal{R}$ に属する．
- A3. $L_1, L_2\in\mathcal{R}$ のとき，$L_1\cup L_2$, L_1L_2 は $\mathcal{R}$ に属する．
- A4. $L\in\mathcal{R}$ のとき，L^* は $\mathcal{R}$ に属する．
- A5. $L\in\mathcal{R}$ のとき，L^ω は $\mathcal{R}$ に属する．

Σ 上の言語 L が規則 A1-A5 の有限回の適用により得られる言語の族 $\mathcal{R}$ の要素であるとき，L を∞正則集合という．このとき，とくに $L\subseteq\Sigma^\omega$ ならば L はω正則集合，$\varepsilon\notin L$ ならば L はεフリー∞正則集合という．また，L が，上記規則より A5 を取り除いて規則 A1-A4 の有限回の適用により得られる言語の族の要素であるとき，L は単に正則集合といわれ，このとき，とくに $\varepsilon\notin L$ ならば L はεフリー正則集合と呼ばれる．

2.2 正則時相論理の定義

正則時相論理は，通常の命題論理に対して，時間の概念を扱う演算子として「次の時刻」，「次の区間」，「繰り返し」を意味する3つの時相演算子「○」，「：」，「[:]」を追加したものである．

定義2（シンタックス） AP を原始命題の集合とす

る．このとき，次の生成規則 B1-B3 の有限回の適用により得られる式を正則時相論理式という．

B1. $p\in AP$ のとき，p は正則時相論理式である．

B2. η が正則時相論理式のとき，$(\neg\eta)$, $(\bigcirc\eta)$, $(\boxdot\eta)$ は正則時相論理式である．

B3. η, ξ が正則時相論理式のとき，$(\eta\vee\xi)$, $(\eta:\xi)$ は正則時相論理式である．

Σ を状態の集合とするとき，正則時相論理式の真偽値は Σ に含まれる状態の系列に対して定義される．このとき，正則時相論理式の真偽値を定義する領域を D とすると，$D=\Sigma^+$ の場合は ε フリー有限正則時相論理，$D=\Sigma^*$ の場合は有限正則時相論理，$D=\Sigma^\omega-\{\varepsilon\}$ の場合は ε フリー無限正則時相論理，$D=\Sigma^\omega$ の場合は無限正則時相論理という．

ε フリー有限正則時相論理は文献 6), 7) で用いた時相論理と同じであり，有限正則時相論理は文献 16), 17) で用いた時相論理と同じであり，ε フリー無限正則時相論理は文献 14), 15) で用いた時相論理と同じである．

形式的には，正則時相論理式の真偽値は次のように定義される．

定義 3（線形モデル） 状態の集合 Σ と原始命題の意味関数 I の組 $M=(\Sigma, I)$ を正則時相論理の線形モデルという．I は各状態に対して真となる原始命題の集合を与える関数で，ε フリー有限／無限正則時相論理の場合は $I:\Sigma\to 2^{AP}$，有限／無限正則時相論理の場合は $I:\Sigma\cup\{\varepsilon\}\to 2^{AP}$ である．

定義 4（正則時相論理式の真偽値） 線形モデル M の状態系列 σ に対して正則時相論理式 η が真となるとき，$M, \sigma\vDash\eta$ と表現する．とくに M が明確なときは，単に $\sigma\vDash\eta$ と書く．p を原始命題，η, ξ を正則時相論理式，D を正則時相論理の定義域，$\sigma\in D$ として，

C1. $\sigma\vDash p\Leftrightarrow p\in I(\sigma(1))$.

C2. $\sigma\vDash(\neg\eta)\Leftrightarrow\sigma\nvDash\eta$.

C3. $\sigma\vDash(\eta\vee\xi)\Leftrightarrow\sigma\vDash\eta$ または $\sigma\vDash\xi$.

C4. $\sigma\vDash(\bigcirc\eta)\Leftrightarrow$
- ε フリーの場合：$|\sigma|\geq 2$ かつ $\sigma^2\vDash\eta$.
- それ以外の場合：$\sigma^2\vDash\eta$.

C5. $\sigma\vDash(\eta:\xi)\Leftrightarrow$
- $|\sigma|\neq\omega$ の場合：$\sigma=\sigma_1\sigma_2$, $\sigma_1\vDash\eta$, $\sigma_2\vDash\xi$ なる $\sigma_1, \sigma_2\in D$ が存在する．
- $|\sigma|=\omega$ の場合：$\sigma=\sigma_1\sigma_2$, $|\sigma_1|\neq\omega$, $|\sigma_2|=\omega$, $\sigma_1\vDash\eta$, $\sigma_2\vDash\xi$ なる $\sigma_1, \sigma_2\in D$ が存在するか，あるいは，$\sigma\vDash\eta$.

C6. $\sigma\vDash(\boxdot\eta)\Leftrightarrow$
- $|\sigma|\neq\omega$ の場合：$\sigma=\sigma_1\sigma_2\ldots\sigma_m$, $\sigma_i\vDash\eta(1\leq\forall i\leq m)$ なる $\sigma_i\in D(1\leq\forall i\leq m)$ が存在する．
- $|\sigma|=\omega$ の場合：$\sigma=\sigma_1\sigma_2\ldots\sigma_m$, $\sigma_i\vDash\eta(1\leq\forall i\leq m)$, $|\sigma_i|\neq\omega(1\leq\forall i<m)$, $|\sigma_m|=\omega$ なる $\sigma_i\in D(1\leq\forall i\leq m)$ が存在するか，あるいは，$\sigma=\sigma_1\sigma_2\ldots$, $i\geq 1$ なる任意の i に対して，$\sigma_i\vDash\eta$, $|\sigma_i|\neq\omega$ なる $\sigma_i\in D$ が存在する．

直感的には，「$\bigcirc\eta$」は次の時刻から始まる系列に対して η が成立することを意味し，「$\eta:\xi$」は系列の前半で η が成立し後半で ξ が成立することを意味し，「$\boxdot\eta$」は η が繰り返し成立することを意味している．

本稿では，以下「$\wedge$」，「$\Rightarrow$」，「$\equiv$」，「V_T」，「V_F」の記号で，各々通常の命題論理の「論理積」，「含意」，「等価」，「恒真式」，「恒偽式」を表し，単項演算子は 2 項演算子よりも優先順位が高いものとし，不要な括弧 '(', ')' は省略できるものとする．

正則時相論理式 η に対して，$M, \sigma\vDash\eta$ なる線形モデル M と状態系列 $\sigma\in D$ が存在するとき，η は充足可能であるという．

2.3 正則時相論理の表現能力

η を正則時相論理式とし，η を真にする系列の集合を $L\langle\Sigma, I\rangle(\eta)=\{\sigma\mid\sigma\in D, \sigma\vDash\eta\}$ で表す．また，混乱の恐れがないときは，$L\langle\Sigma, I\rangle(\eta)$ を単に $L(\eta)$ と略記する．このとき，正則時相論理の定義 4 より次の補題が成立する．

補題 1 p を原始命題，η, ξ を正則時相論理式とする．正則時相論理の線形モデル $M=(\Sigma, I)$ に対して，

1. $L(p)=\{s\mid p\in I(s)\}\{\sigma\mid\sigma\in D\}\cup\{\varepsilon\mid\varepsilon\in D, p\in I(\varepsilon)\}$.
2. $L(\neg\eta)=D-L(\eta)$.
3. $L(\eta\vee\xi)=L(\eta)\cup L(\xi)$.
4. $L(\bigcirc\eta)=\Sigma L(\eta)\cup(\{\varepsilon\}\cap L(\eta))$.
5. $L(\eta:\xi)=L(\eta)L(\xi)$.
6. $L(\boxdot\eta)=L(\eta)^+\cup(L(\eta)^*\cap D)$.

あるタイプの正則時相論理（ε フリー有限正則時相論理，有限正則時相論理，ε フリー無限正則時相論理，無限正則時相論理）$\mathcal{T}$ において，$\mathcal{T}$ の任意の時相論理式 η に対して $L\langle\Sigma, I\rangle(\eta)$ が Σ 上の系列集合のクラス $\mathcal{R}$ の要素であり，逆に，$\mathcal{R}$ に属する任意の系列集合 R に対して，$L\langle\Sigma, I\rangle(\eta)=R$ となる $\mathcal{T}$ の時相論理式 η と原始命題の意味関数 I が存在するとき，$\mathcal{T}$ は $\mathcal{R}$ に表現等価であるという．

前節で定義した 4 つのタイプの正則時相論理の表現

能力について以下の定理がなりたつ.

定理1 (正則時相論理の表現能力)

1. εフリー有限正則時相論理はεフリー正則集合と表現等価である[6].
2. 有限正則時相論理は正則集合と表現等価である[17].
3. εフリー無限正則時相論理はεフリー∞正則集合と表現等価である[14),15].
4. 無限正則時相論理は∞正則集合と表現等価である.

(証明) 4の場合についてのみ証明する. まず, 補題1より無限正則時相論理式ηに対して, $L(\eta)$が∞正則集合になることは容易に示せる. 逆に, RをΣ上の∞正則集合とする. この時, $AP=\{p_s \mid s\in\Sigma\}$, $I(p_s)=\{s\}$とし, $F(R)$で$L(\eta)=R$となる無限時相論理式ηを表すものとすると,

1. $F(\emptyset)=V_F$,
2. $F(\{s\in\Sigma\})=p_s\wedge\neg p_0\wedge\bigcirc p_0$,
3. $F(R_1\cup R_2)=F(R_1)\vee F(R_2)$,
4. $F(R_1R_2)=F(R_1):F(R_2)$,
5. $F(R^*)=p_0\vee\boxdot F(R_1)$,
6. $F(R^\omega)=\boxdot(F(R)\wedge Fin\wedge\neg p_0)$,

となる. ここに, p_0は$I(p_0)=\{\varepsilon\}$なる原始命題で, $\neg p_0\wedge\bigcirc p_0$は長さ1のすべての系列の集合を表し, $Fin\triangleq\neg(V_T:V_F)$は, すべての有限長系列の集合(空語εを含む)を意味している.

有限オートマトンの動作系列(入出力系列)は, 有限長の動作系列に着目した場合は(εフリー)正則集合, 無限長の動作に着目した場合はω正則集合, 両者に着目した場合は(εフリー)∞正則集合と等価であるので, 正則時相論理は有限オートマトンと等価な時相論理体系であり, 任意の有限オートマトンの仕様を動作系列の形で正則時相論理により記述することが可能である.

3. 順序機械の設計検証

3.1 仕様と設計の記述

正則時相論理を用いた順序機械の設計検証問題を考える. 設計対象の順序機械は, 決定性のMealy型あるいはMoore型の順序機械で, nビットの入力信号$X=\{x_1,x_2,\cdots,x_n\}$とmビットの出力信号$Z=\{z_1,z_2,\cdots,z_m\}$を持つものとする. このような順序機械の1つの動作系列は$\rho(k)\triangleq\{x_i \mid k$番目の入力において$x_i=1\}\cup\{z_j \mid k$番目の出力において$z_j=1\}$で定義される$2^{X\cup Y}$上の系列$\rho$で表すことができる.

設計仕様としては, 設計対象の順序機械の可能な動作系列の集合を与えるものとする. ここでは, とくに, 任意の長さの有限動作系列を考え, 入力信号x_iに対して$x_i=1$のときに限り真となる原始命題$p_{x_i}(1\le i\le n)$, 出力信号z_jに対して$z_j=1$のときに限り真となる原始命題$p_{z_j}(1\le j\le m)$を用いて, 設計対象の順序機械において可能な任意の長さの有限動作系列の集合をεフリー有限正則時相論理で記述するものとする. 無限の長さの動作系列についてもεフリー無限正則時相論理により記述することができるが, 誌面の都合上無限長の動作に対する検証問題については別稿に譲る. 以下, εフリー有限正則時相論理を単にRTLと書く.

例として, **図1**に示すように出力zの初期値が0で, 入力xの値が1のときに限り次の時刻で出力zの値が反転するような1入力1出力のTフリップフロップの設計検証を考える.

$x=1$のときにのみ真となる原始命題p_xと$z=1$のときにのみ真となる原始命題p_zを用いると, このTフリップフロップの仕様は次のように記述できる.

$$\eta\triangleq p_x\equiv(p_z\equiv\bigcirc\neg p_z)$$
$$spec\triangleq\neg p_z\wedge\Box(\eta\vee LEN1)$$

ここに, $LEN1\triangleq\neg\bigcirc V_T$, $\Diamond\xi\triangleq\xi\vee(V_T:\xi)$, $\Box\xi\triangleq\neg\Diamond\neg\xi$である.

ηは現在の入力xの値が1のときにのみ現時刻と次の時刻で出力zの値が異なることを意味している. また, $LEN1$, $\Diamond\xi$, $\Box\xi$は各々,「系列の長さが1」,「現時刻以降いつかξが成立する」,「現時刻以降常にξが成立する」ことを意味しており, Tフリップフロップの仕様$spec$は出力zの初期値が0で以後常にηが成立することを意味している. ここで, $spec$の定義において$LEN1$を用いているのは, 有限長の動作系列の最後の時刻においてηが存在しない次の時刻を参照して偽となるのを無視するためである.

このTフリップフロップの設計は, **図2**に示すようにMealy型あるいはMoore型の順序機械で与えられるものとする. このとき, 設計検証の問題は,「設計された順序機械のすべての可能な有限長の動作系列に

入出力系列の例 x: 0010111001 z: 0001101000

図1 Tフリップフロップ
Fig. 1 T flipflop.

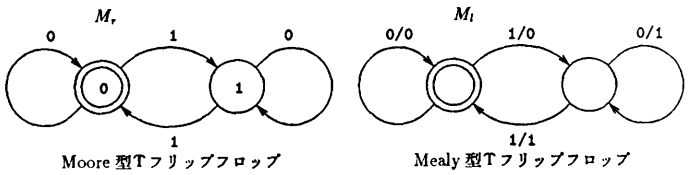


図 2 Tフリップフロップの設計
Fig. 2 Two designs of T flipflop.

対して，仕様として与えられた RTL 式 *spec* が真となることを確かめる」という問題になる．

3.2 構造モデル

順序機械のすべての可能な動作系列に対して RTL 式の真偽判定を行うのを容易にするために，次のような構造モデルを考える．

定義 5 (構造モデル) 4つ組 $K=(\Sigma, I, R, \Sigma_0)$ を構造モデルという．ここに，

1. (Σ, I) は RTL の線形モデル，
2. $R\subseteq\Sigma\times\Sigma$ は状態の間の遷移関係を表す Σ 上の2項関係，
3. $\Sigma_0\subseteq\Sigma$ は初期状態の集合である．

構造モデルは Kripke モデル[8]に初期状態の集合を加えたものである．構造モデル $K=(\Sigma, I, R, \Sigma_0)$ の有限長の状態系列 $\pi=s_1s_2\cdots s_n\in\Sigma^+$ が $1\leq i<n-1$ なる i に対して $(s_i, s_{i+1})\in R$ を満たすとき，π を K 上の s_1 からの有限長の道という．

構造モデル K に対する RTL 式 η の真偽値を次のように定義する．

定義 6 (構造モデルに対する真偽値) 構造モデル K の状態 s からのある有限長の道 σ に対して $\sigma\vDash\eta$ となるとき，η は $\langle K, s\rangle$ のもとで真 ($\langle K, s\rangle$-true) であるといい，そうでないときは，η は $\langle K, s\rangle$ のもとで偽 ($\langle K, s\rangle$-false) であるという．また，ある初期状態 $s_0\in\Sigma_0$ に対して η が $\langle K, s_0\rangle$-true となるとき，η は K のもとで真 (K-true) といい，そうでないときは，η は K のもとで偽 (K-false) という．

さて，$M=(X, Z, S, \delta, \lambda, s_0)$ を決定性の順序機械とする．ここに，

1. X は2値の入力信号線の有限集合，
2. Z は2値の出力信号線の有限集合，
3. S は状態の有限集合，
4. $s_0\in S$ は初期状態，
5. $\delta: 2^X\times S\to S$ は次状態関数，
6. λ は出力関数で，
   - Moore 型の場合は，$\lambda: S\to 2^Z$ なる S 上の全域関数，
   - Mealy 型の場合は，$\lambda: 2^X\times S\to 2^Z$ で δ と同じ定義域をもつ関数，

である．

このとき，順序機械 M に対して，M に対応する構造モデル K を次のように構成する．

$$\Sigma\triangleq\{s_{i,j}\mid s_i\in S, j\in 2^X, \delta(j, s_i)\text{ が定義されている}\}$$

$$I(s_{i,j})\triangleq\begin{cases}\{p_x\mid x\in j\}\cup\{p_z\mid z\in\lambda(j, s_i)\} & \text{Mealy 型の場合}\\ \{p_x\mid x\in j\}\cup\{p_z\mid z\in\lambda(s_i)\} & \text{Moore 型の場合}\end{cases}$$

$$R\triangleq\{(s_{i,j}, s_{i',j'})\mid s_{i,j}, s_{i',j'}\in\Sigma, \delta(j, s_i)=s_{i'}\}$$

$$\Sigma_0\triangleq\{s_{0,j}\in\Sigma\}$$

Tフリップフロップに対応する構造モデルを**図 3** に示す．順序機械 M の状態遷移図における枝の数を $|E|$ とすると，M に対応する構造モデル K のサイズは次のようになる．

$$O(|\Sigma|)=O(|E|)=O(|\Sigma|2^{|X|})$$
$$O(|R|)=O(|E|2^{|X|})=O(|\Sigma|2^{2|X|})$$
$$O(|\Sigma|+|R|)=O((|\Sigma|+|E|)2^{|X|})$$

この構造モデルの構成方法より明らかなように，M の初期状態からの可能な動作系列と M に対応する構造モデル K の初期状態からの道は1対1に対応する．したがって，順序機械 M の設計検証問題を M に対応する構造モデル K 上で考えると，「K の初期状態からのすべての有限長の道に対して *spec* が真になる」あるいは，「$\neg$*spec* が K-false である」ことを確かめれば良い．これは次章で述べるモデルチェック法により求めることができる．

4. モデルチェック法

構造モデル K と RTL 式 η に対し，η が K-true であるかどうかを判定することをモデルチェックと呼

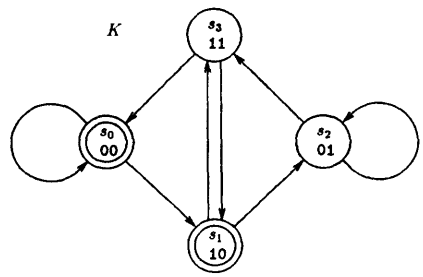


図 3 Tフリップフロップの構造モデル
Fig. 3 Structure model of T flipflop.

ぶ．モデルチェックアルゴリズムについて述べる前に，RTL 式に対する微分演算について述べる．

4.1 ε フリー有限正則時相論理式の微分

RTL 式 η の状態 s による微分を η/s で表す．η/s は，現状態 s において η が成立するために次状態で成り立つべき必要十分条件を RTL 式で表現したものであり，次のように定義される．

$$p/s \triangleq \begin{cases} V_T & p \in I(s) \text{ の場合} \\ V_F & \text{それ以外の場合} \end{cases}$$

$$(\neg)\eta/s \triangleq \neg(\eta/s)$$

$$(\eta \vee \xi)/s \triangleq (\eta/s) \vee (\xi/s)$$

$$(\bigcirc\eta)/s \triangleq \eta$$

$$(\eta : \xi)/s \triangleq \begin{cases} \xi \vee ((\eta/s) : \xi) & s \vDash \eta \text{ の場合} \\ (\eta/s) : \xi & \text{それ以外の場合} \end{cases}$$

$$(\boxdot\eta)/s \triangleq \begin{cases} (\eta/s) \vee ((\eta/s) : \boxdot\eta) \vee \boxdot\eta & s \vDash \eta \text{ の場合} \\ (\eta/s) \vee ((\eta/s) : \boxdot\eta) & \text{それ以外の場合} \end{cases}$$

また，η の状態系列 $\sigma = s_1 s_2 \cdots s_n \in \Sigma^+$ による微分を

$$\eta/\sigma \triangleq ((\cdots((\eta/s_1)/s_2)\cdots)/s_n)$$

と定義し，さらに，便宜上 $\eta/\varepsilon \triangleq \eta$ とする．とくに，$V_T/s = V_T$, $V_F/s = V_F$ である．

微分演算の定義より明らかに次の補題が成り立つ．

補題 2 σ を $|\sigma| \geq 2$ なる Σ 上の有限長系列とすると，$\sigma \vDash \eta$ となるための必要十分条件は，$\sigma^2 \vDash \eta/\sigma$ (1)である．

定理 2 構造モデル $K=(\Sigma, I, R, \Sigma_0)$ の状態 s に対して η が $\langle K, s\rangle$-true となるための必要十分条件は，$s \vDash \eta$ となるか，または，$(s, s') \in R$ なるある状態 s' に対して η/s が $\langle K, s'\rangle$-true となることである．

(証明) η が $\langle K, s\rangle$-true であるための必要十分条件は，定義6より s からはじまる K 上のある有限長の道 σ に対して $\sigma \vDash \eta$ となることである．したがって，補題2より，これは，$s \vDash \eta$ または $(s, s') \in R$ なるある状態に対して η/s が $\langle K, s'\rangle$-true になることと等価である．

定理2の条件の検査や微分においては長さ1の状態系列 s に対する η の真偽値を求める必要があるが，これは式の構成に関して再帰的に求めることができる．

1. $s \vDash p \Leftrightarrow p \in I(s)$.
2. $s \vDash \eta \vee \xi \Leftrightarrow s \vDash \eta$ または $s \vDash \xi$.
3. $s \vDash \neg\eta \Leftrightarrow s \nvDash \eta$.
4. 常に $s \nvDash \bigcirc\eta$.
5. 常に $s \nvDash \eta : \xi$.
6. $s \vDash \boxdot\eta \Leftrightarrow s \vDash \eta$.

4.2 モデルチェックアルゴリズム

定理2で述べた必要十分条件を構造モデル上で深さ優先探索により調べることでモデルチェックを行うこ

```
procedure Verify(K, η)
  begin
    for all s ∈ Σ0
      begin
        if Label(s, η) ≠ 'F' then
          if Check(K, s, η) = 'T' then return 'T';
      end
    return 'F';
  end
end of procedure

procedure Check(K, s, η)
  begin
    (x, ξ) := Derivation(s, η);
    if x = 'T' then return 'T';
    if ξ = V_T then return 'T';
    if ξ = V_F then
      begin
        Addlabel(s, η, 'F');
        return 'F';
      end
    Addlabel(s, η, 'C');
    for all s' such that (s, s') ∈ R
      begin
        x := Label(s', ξ);
        if x = NIL then
          if Check(K, s', ξ) = 'T' then return 'T';
      end
    Addlabel(s, η, 'F');
    return 'F';
  end
end of procedure
```

```
procedure Derivation(s, η);
  begin
    switch(η){
    case η ∈ AP:
      if η ∈ I(s) then return ('T', V_T);
      else return ('F', V_F);
    case η = ¬η1:
      (x, ξ) := Derivation(s, η1);
      if x = 'T' then return ('F', ¬ξ);
      else return ('T', ¬ξ);
    case η = η1 ∨ η2:
      (x1, ξ1) := Derivation(s, η1);
      (x2, ξ2) := Derivation(s, η2);
      if (x1 = 'T' or x2 = 'T') then x := 'T';
      else x := 'F';
      return (x, ξ1 ∨ ξ2);
    case η = ○η1:
      return ('F', η1);
    case η = η1 : η2:
      (x, ξ) := Derivation(s, η1);
      ξ := ξ : η2;
      if x = 'T' then ξ := ξ ∨ η2;
      return ('F', ξ);
    case η = ⊡η1:
      (x, ξ) := Derivation(s, η1);
      ξ := ξ ∨ (ξ : η);
      if x = 'T' then ξ := ξ ∨ η;
      return (x, ξ);
    }
  end
end of procedure
```

図 4 モデルチェックアルゴリズム
Fig. 4 Model checking algorithm.

とができる.

アルゴリズム 1 (モデルチェックアルゴリズム)

入力: 構造モデル $K=(\Sigma, I, R, \Sigma_0)$ と RTL 式 η.

出力: η が K-true ならば 'T', さもなくば 'F'.

方法: **図 4** の Verify(K, η) による.

図4において, Check(K, s, η) は η が $\langle K, s\rangle$-true のときに 'T' そうでないときには 'F' を返すプロシジャーである. Addlabel (s, η, x) は s と η の組に対してラベル x をふるプロシジャーでラベル 'F' は η が $\langle K, s\rangle$-false であることを意味し, ラベル 'C' は η が $\langle K, s\rangle$-true であるかどうかを現在調査中であることを意味している. これらのラベルは同じ s と η の組に対して Check が2回以上呼ばれることを避けるために用いている. ラベルとして η が $\langle K, s\rangle$-true であることを示すラベル付けは行っていないが, これはいったん Check が 'T' を返すと以降は Check を新たに呼び出すことなく Verify が 'T' になるからである. Label (s, η) は s と η の組のラベルを返す関数でラベルがない場合には NIL を返す.

Verify (K, η) は, すべての初期状態 $s\in\Sigma_0$ に対して, η が $\langle K, s\rangle$-false であることがすでに判明していなければ順次 Check (K, s, η) を呼び出し, Check が1度でも 'T' を返すと Verify も 'T' を返し, そうでなければ 'F' を返す.

Check (K, s, η) は, まず Derivation (K, s, η) を呼び出す. Derivation (K, s, η) は, $s\vdash\eta$ の判定と η/s の計算を同時に行い, ξ に η/s を代入し, x には $s\vdash\eta$ の場合には 'T', $s\nvdash\eta$ の場合には 'F' を代入する. そして, $\xi=\eta/s$ が V_T あるいは x ='T' の場合は 'T' を返し, $\xi=\eta/s$ が V_F の場合は s と η の組に 'F' をラベル付けして 'F' を返す. それ以外のときは, s と η の組に 'C' をラベル付けし, 次に s のすべての次状態 s' に対し順次 Label $(s', \eta/s)$ を調べ, ラベル付けがなされていなければ再帰的に Check $(K, s', \eta/s)$ を呼び出し, 'T' が返されたときにのみ 'T' を返す. それ以外のときには, s と η の組に対して 'F' をラベル付けしてから 'F' を返す.

図 5 にTフリップフロップに対してモデルチェックを行った例を示す. 図5において, 実線の箱は Check (K, s, η) が箱の中に記された引数で呼ばれたことを表しており, 点線の箱は箱の中の状態と式の組に対してすでにラベル 'F' または 'C' が付されていたことを表している. 枝に付けられた数字は, 構造モデル上での深さ優先探索の探索順序を示している. 各 Check の呼び出しはすべて値 'F' で終了し, 最終的に Verify $(K, \neg spec)$ は 'F' を返す. すなわち, これは図2に示した設計が $spec$ を満たしていることを意味している.

4.3 正当性と計算量

まず RTL 式 η を繰り返し微分したときに得られる式の数が高々有限個であることを示す.

$D^*(\eta)\triangleq\{\eta/\sigma\,|\,\sigma\in\Sigma^*\}$ とする. $D^*(\eta)$ の定義において, $V_T\vee\xi=V_T$, $V_F\vee\xi=\xi$, 論理和のべき等則, 結合則, 可換則により等しくなる式は同一の式であると見なす.

補題 3 $D^*(\eta)$ は有限集合である.

(証明) RTL 式の構成法に関して再帰的に証明する.

1. $D^*(p)=\{p, V_T, V_F\}$.
2. $D^*(\neg\eta)=\{\neg\xi\,|\,\xi\in D^*(\eta)\}$.
3. $D^*(\eta_1\vee\eta_2)\subseteq\{\xi_1\vee\xi_2\,|\,\xi_1\in D^*(\eta_1),\ \xi_2\in D^*(\eta_2)\}$.
4. $D^*(\bigcirc\eta)=\{\bigcirc\eta\}\cup D^*(\eta)$.
5. $E_1\triangleq\{\vee[\Theta]\,|\,\Theta\in 2^{D^*(\eta_2)}\}$, $E_2\triangleq\{\xi_1:\eta_2\,|\,\xi_1\in D^*(\eta_1)\}$ とすると
   $D^*(\eta_1:\eta_2)\subseteq\{\nu_1\vee\nu_2\,|\,\nu_1\in E_1,\ \nu_2\in E_2\}$.
6. $F_1\triangleq\{\vee\Theta\,|\,\Theta\in 2^{D^*(\eta)\cup\{\boxdot\eta\}}\}$, $F_2\triangleq\{\xi_1:\boxdot\eta\,|\,\xi_1\in D^*(\eta)\}$ とすると,
   $D^*(\boxdot\eta)\subseteq\{\nu_1\vee\nu_2\,|\,\nu_1\in F_1,\ \nu_2\in F_2\}$.

ここに, $\vee[\Theta]$ は Θ に含まれるすべての式の論理和

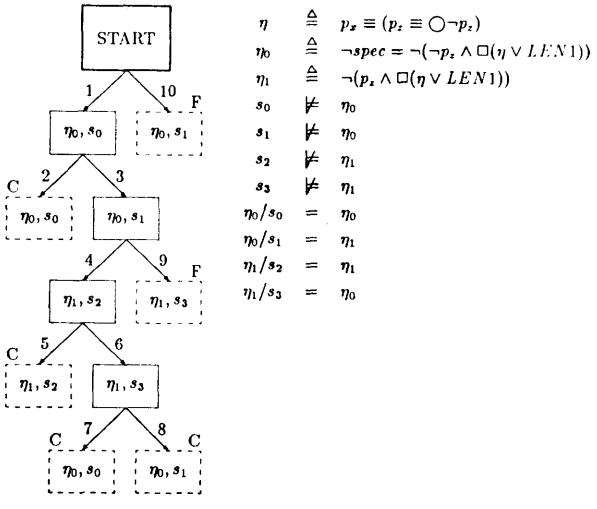


図 5 モデルチェックによるTフリップフロップの設計検証
Fig. 5 Verification of T flipflop by the model checking algorithm.

を表す．帰納法の仮定より，上式の右辺はすべて有限集合となる．

［アルゴリズム1の正しさの証明］ アルゴリズムの停止性とループを検出したときの処理以外は，K-trueの定義と定理2よりアルゴリズムが正しいことは明らかである．

停止性：補題3より，アルゴリズム実行中に微分で生成される異なる式の個数は有限個であり，また，Checkは同一の状態と式の組に対して高々一度しか実行されないので，アルゴリズム1は必ず停止する．

ループの処理：Check (K, s, η) がループを検出した場合，すなわち，$(s, s') \in R$ なる s' に対して Label $(s', \xi) =$ 'C'$(\xi = \eta/s)$ となった場合を考える．この場合，Check (K, s, η) は Label $(s', \xi) =$ 'F' の時と同様に扱っているが，これが正しいことは次のように示せる．s' から s までのループ上の状態系列を σ とする．Label $(s', \xi) =$ 'C' であるので $\xi/\sigma = \xi$，また σ の任意の prefix 系列 σ_p に対して $\sigma_p \not\vDash \xi$ である．ここで，ある系列 $\tau \in \{\sigma\}^+$ に対して $\tau\sigma_p \vDash \xi$ となると仮定すると，$\xi/\sigma = \xi$ であるので，補題2を繰り返し適用することにより $\sigma_p \vDash \xi$ となり矛盾する．したがって，ξ はこのループにより生成される無限系列の任意の有限 prefix 系列に対して偽となるので，Label $(s', \xi) =$ 'F' の場合と同様に扱ってよいことが分かる．

次にアルゴリズム1の計算量の評価を行う．$\mathcal{N} \triangleq |\mathcal{D}^*(\eta)|$ とし，$\mathcal{L}$ で $\mathcal{D}^*(\eta)$ の要素である式に含まれる演算子数の最大値を表すものとする．また，$|\eta|$ で式 η に含まれる演算子の個数を表す．

定理 3 アルゴリズム1の計算時間は，$O((|\Sigma|(\mathcal{L}\log\mathcal{L} + \log\mathcal{N}) + |R|\log\mathcal{N})\mathcal{L}\mathcal{N})$ である．

（証明） まず Derivation (s, ξ) に要する時間を評価する．Derivation では，$s \vDash \xi$ が成り立つかどうかの判定と ξ/s の計算を同時に行っているが，ξ の各部分式 ξ' に対してボトムアップに $s \vDash \xi'$ かどうかの判定を行うことにより全体で $|\xi|$ に比例する時間で $s \vDash \xi$ が成り立つかどうかの判定を行うことができる．ξ/s の計算においては，ξ のいくつかの部分式 ξ' について $s \vDash \xi'$ かどうかの判定が必要となる場合があるがこれはすでに求まっており，また，1回の微分演算においてある部分式やその微分が2回表れることがあるが（「：」や「□」の微分の場合）式をグラフ表現で表し共通の部分式を表す部分グラフを共有することにより ξ の演算子1個につき高々3個の演算子を追加するだけで良いので ξ/s の計算は基本的には $|\xi|$ に比例する時間で行える．しかしながら，アルゴリズムの停止性を保証するためには，ξ/s を論理和のべき等則，可換則，結合則を用いて簡単化する必要がある．この簡単化は，論理和節の各項をソートすることにより行えるが，項のソートに全体で $O(|\xi/s|\log|\xi/s|)$ 回の式の比較が必要で1回の比較は $|\xi/s|$ に比例する時間で行える．したがって，Derivation に要する時間は，$\xi, \xi/s \in D^*(\eta)$ であるので，$O(\mathcal{L}^2\log\mathcal{L})$ である．

次に Addlabel と Label であるが，$D^*(\eta)$ に含まれる式をソートしながら管理することにより，ともに $\log\mathcal{N}$ 回の比較で実現することができ，1回の比較に $\mathcal{L}$ に比例する時間がかかるので，Addlabel と Label に要する時間は，$O(\mathcal{L}\log\mathcal{N})$ である．

さて，$D^*(\eta)$ に含まれる各式に対して，Check は再帰呼び出しの回数も含めて全体で $O(|\Sigma|)$ 回呼び出されるが，この $|\Sigma|$ 回の Check の実行を通して，Derivation と Addlabel は全体で $|\Sigma|$ 回呼び出され，Label は最大すべての状態のすべての次状態に対して呼び出されるので全体で $|R|$ 回呼び出されることになる．したがって，アルゴリズム1が要する時間は，$O((|\Sigma|(\mathcal{L}\log\mathcal{L} + \log\mathcal{N}) + |R|\log\mathcal{N})\mathcal{L}\mathcal{N})$ である．

モデルチェック問題の決定性チューリング機械 (DTM) の領域複雑度に関して次の定理が得られる．

補題 4[6] ε フリー有限正則時相論理の充足可能性判定問題の DTM 領域複雑度は非初等的である．

定理 4 ε フリー有限正則時相論理式のモデルチェック問題の DTM 領域複雑度は，非初等的である．

（証明） RTL 式の充足可能性判定問題が初等的時間で RTL 式のモデルチェック問題に帰着できることを示す．RTL 式 η に表れる原始命題の集合を AP' とする．$2^{|AP'|}$ 個の状態を持つ状態集合 Σ を考え，I を Σ から $2^{AP'}$ 上への1対1写像とし $R = \Sigma \times \Sigma$, $\Sigma_0 = \Sigma$ とする．このとき構造モデル $K_\varepsilon = (\Sigma, I, R, \Sigma_0)$ を考えると，η が充足可能であることと，η が K_ε-true であることとが等価になる．また，K_ε は $O(2^{2|\eta|})$ の時間で構成できる．

5. モデルチェッカの実現と検証例

モデルチェック問題の計算の複雑さは，定理4により式の長さに対して非初等的になるが，4.2 節で示したモデルチェックアルゴリズムは定理3より式を固定して考えると構造モデルのサイズ $(|\Sigma| + |R|)$ に対して比例する時間でモデルチェックを行うことができ

る．そこで，このアルゴリズムの評価を行うために，実際に RTL のモデルチェッカを sun 3/60 上に実現し，実用的な規模の順序機械の設計検証を行った．

5.1 RTL モデルチェッカ

RTL モデルチェッカでは，RTL 式は基本的には2分木の形で記憶されている．2分木の各節点は演算子または原始命題を表す．与えられた RTL 式とその部分式を表す木は登録されている．微分によって作り出された部分式のうち，同じ形の式が再び現れる可能性の高いものは登録されている式と比較し，登録されている式に同じ形のものがあれば，その部分式を表す木は解放し，登録されているもので置き換え，記憶量を節約している．新しい形の式ならば，新たに登録する．

この RTL モデルチェッカは，仕様として与えられた RTL 式の真偽判定を行う以外に，RTL 式が偽になるときには，反例として式を偽にする構造モデル上の状態系列を出力することができ，また，この状態系列上で RTL 式が偽になる理由を，原因となる部分式と，その部分式が偽になる原因である構造モデルの状態を出力することによって示すこともできる．この機能は，設計が正しくないことが判明したときに設計誤り箇所を発見するのに有効である．

5.2 信号制御器の検証

RTL モデルチェッカの有効性を調べるために，南北に走る道路と東向き一方通行の道路の交差点に設置された信号の制御器[1]の設計検証を行った．

信号制御器は3つの入力信号 (N, S, E)，3つの出力信号 (N_GO, S_GO, E_GO)，5つの内部信号をもつ．N, S, E はそれぞれ，1台以上の車が北，南，東に向かって交差点を横切ろうとしていることを表し，N_GO, S_GO, E_GO はそれぞれ，北，南，東向きの信号機が青であることを表す．信号制御器は Moore 機械として設計されている．1つは設計誤りのあるもので，43状態からなる．もう1つは設計の正しいもので，31状態からなる．対応する構造モデルの状態数はそれぞれ344状態，248状態である．

RTL 式で記述された，信号制御器の仕様 *spec* を図6に示す．*nocoli* は南北方向の信号機と東向きの信号機が同時に青にはならないことを表す．*ic* は入力制約であり，N, S, E が成立するなら，これらはそれぞれ N_GO, S_GO, E_GO が成立するまで，成立し続けることを表す．*asn* 4, *ass* 4, *ase* 4 はそれぞれ北，南，東向きに交差点を横切ろうとする車があり，直角方向には車のない状態を表す．*ngoby* 4, *sgoby* 4, *egoby* 4 は，対応する向きの信号機が，現在を含めて4単位時間以内に青になることを表す．*spec* は，直交する向きの信号機が同時には青にならず，また，交差点に入ろうとする車があり，直角方向の車がなければ，入力制約を満たす限りその方向の信号が4単位時間以内に青になるという仕様を表す．

$$
\begin{aligned}
len2 &\triangleq \bigcirc(LEN1)\\
len3 &\triangleq \bigcirc(len2)\\
lengt3 &\triangleq \neg(LEN1 \vee len2 \vee len3)\\
nocoli &\triangleq \Box(\neg(E\_GO \wedge (N\_GO \vee S\_GO)))\\
icn &\triangleq \neg((\Box(N \wedge \neg N\_GO)) : \neg N)\\
ics &\triangleq \neg((\Box(S \wedge \neg S\_GO)) : \neg S)\\
ice &\triangleq \neg((\Box(E \wedge \neg E\_GO)) : \neg E)\\
ic &\triangleq \Box(icn \wedge ics \wedge ice)\\
asn4 &\triangleq N \wedge \neg E \wedge lengt3\\
ass4 &\triangleq S \wedge \neg E \wedge lengt3\\
ase4 &\triangleq E \wedge \neg(N \vee E) \wedge lengt3\\
ngoby4 &\triangleq N\_GO \vee \bigcirc(N\_GO \vee \bigcirc(N\_GO \vee \bigcirc N\_GO))\\
sgoby4 &\triangleq S\_GO \vee \bigcirc(S\_GO \vee \bigcirc(S\_GO \vee \bigcirc S\_GO))\\
egoby4 &\triangleq E\_GO \vee \bigcirc(E\_GO \vee \bigcirc(E\_GO \vee \bigcirc E\_GO))\\
delay4 &\triangleq ic \Rightarrow (\Box(asn4 \Rightarrow ngoby4) \wedge \Box(ass4 \Rightarrow sgoby4) \wedge \Box(ase4 \Rightarrow egoby4))\\
spec &\triangleq nocoli \wedge delay4
\end{aligned}
$$

図6 信号制御器の仕様
Fig. 6 Specification of a traffic controller.

spec は89個の演算子を持つ．モデルチェッカは誤りのある設計に対し ¬*spec* が真になることを0.2秒で出力し，モデルチェックのために新たに62個の節点を RTL 式の記憶のために使用した．正しい設計に対しては1.1秒で ¬*spec* が偽になることを出力し，新たに322の節点を用いた．これらは実用的に十分な時間と記憶量であると言える．

5.3 DMA コントローラの検証

より状態数の多い順序機械の検証の例として，DMA コントローラ[2]の設計検証を行った．DMA コントローラは5つの入力信号と15の出力信号を持つ Moore 機械として設計されている．1つは設計誤りのあるもので (Bad Design) 392状態からなり，対応する構造モデルは12,544状態を持つ．もう1つは設計の正しいもので (Good Design) 272状態からなり，対応する構造モデルは8,704状態を持つ．

図7 に DMA コントローラの満たすべき性質（アサーション）を示す．*as* 1 は *MemReq* が常に真（信号線の値が1）なら *ActivateComparator* は常に偽（信号線の値が0）であり，*MemReq* が常に偽であれ

$$
\begin{aligned}
len2 &\triangleq \bigcirc LEN1\\
lengt2 &\triangleq \neg(LEN1 \vee len2)\\
as1 &\triangleq (\Box MemReq \Rightarrow \Box\neg ActivateComparator) \wedge (\Box\neg MemReq \Rightarrow \Box\neg MemGrant)\\
as2 &\triangleq \Box((CpuReq \wedge \neg DmaReq) \Rightarrow \Diamond((MemFinished \wedge lengt2) \Rightarrow \bigcirc\bigcirc\neg CpuReq))\\
as3 &\triangleq \Box((\neg TransferReq \wedge \bigcirc TransferReq) \Rightarrow \bigcirc(\Diamond(DeviceReady \Rightarrow (LEN1 \vee \bigcirc DmaReq))))\\
as4 &\triangleq \Box(DmaReq \Rightarrow \Diamond(ActivateComparator \Rightarrow \Diamond(ComparatorSet \Rightarrow\\
&\quad (LEN1 \vee \bigcirc(DmaEnd \vee DmaCont)))))\\
as5 &\triangleq \neg\Diamond(ActivateComparator \wedge MemGrant)\\
as6 &\triangleq \Box(\neg(TransferReq \wedge \bigcirc TransferReq \wedge (DmaType \oplus \bigcirc DmaType)) \vee LEN1)\\
as7 &\triangleq \Box((DmaDone \wedge ComparatorSet) \Rightarrow (DmaEnd \vee \Box DmaDone \vee (\Box DmaDone : DmaEnd)))\\
as8 &\triangleq \Box((\neg DmaDone \wedge ComparatorSet) \Rightarrow (DmaCont \vee \Box(\neg DmaDone) \vee\\
&\quad (\Box\neg DmaDone : DmaCont)))\\
as9 &\triangleq \Box(\neg(\neg ActivateComparator \wedge \bigcirc ActivateComparator \wedge ComparatorSet))\\
asall &\triangleq as1 \wedge as2 \wedge as3 \wedge as4 \wedge as5 \wedge as6 \wedge as7 \wedge as8 \wedge as9
\end{aligned}
$$

図 7 DMA コントローラに対するアサーション
Fig. 7 Assertions for a DMA controller.

ば *MemGrant* も常に偽であることを表す. *as* 2 は, *CpuReq* が真で *DmaReq* が偽ならば *MemFinished* が真になって 2 時刻後に *CpuReq* が立ち下がるということが常にいつか起こることを表す. *as* 3 は *TransferReq* が立ち上がれば, *DeviceReady* が真である次の時刻で *DmaReq* が真であるという状態がいつか起こることを表す. *as* 4 は *DmaReq* が真で *ActivateComparator* と *ComparatorSet* がいつか真になるなら, 次の時刻で *DmaEnd*, *DmaCont* のどちらかが真であることを表す. *as* 5 は *ActivateComparator* と *MemGrant* が同時に真にはならないことを表す. *as* 6 は *TransferReq* が真の間 *DmaType* は値が変わらないことを表す. *as* 7, *as* 8 は *ComparatorSet* が真のときは, *DmaEnd* か *DmaCont* が真になるまで *DmaDone* は値が変わらないことを表す. *as* 9 は *ComparatorSet* が *ActivateComparator* の立ち上がる直前に真ではないことを表す. *asall* は *as* 1〜*as* 9 の論理積である.

表 1 DMA コントローラの検証例
Table 1 Verification of the DMA controller.

	DMA コントローラ (5 入力信号, 15 出力信号)						
	Bad Design (392 状態) 構造モデル 12544 状態				Good Design (272 状態) 構造モデル 8704 状態		
アサーション	#Op.	結 果	時 間 (秒)	#Node	結 果	時 間 (秒)	#Node
*as*1	10	O. K.	0. 8	4	O. K.	1. 5	4
*as*2	13	O. K.	46. 1	46	O. K.	29. 5	46
*as*3	10	O. K.	16. 0	13	O. K.	9. 6	13
*as*4	9	O. K.	18. 2	15	O. K.	11. 4	15
*as*5	3	O. K.	14. 0	1	O. K.	8. 8	1
*as*6	8	O. K.	18. 2	14	O. K.	11. 9	14
*as*7	8	Fail	0. 6	12	O. K.	9. 9	12
*as*8	11	Fail	1. 0	12	O. K.	9. 9	12
*as*9	6	Fail	0. 7	6	O. K.	10. 3	6
asall	86	Fail	3. 5	397	O. K.	70. 6	915

表 1 に設計検証の結果を示す. 各行は, 各アサーションに対する検証の結果を表す. 列 '#Op.' にアサーションの式に含まれる演算子の数を, 列 '#Node' には設計検証の過程で RTL 式の記憶に必要となる節点の数を示す. この数にはアサーションを表す RTL 式の記憶に必要な節点数は含まない. 検証は各アサーションに対して別々に行った. 誤りのある設計は, *as* 7, *as* 8, *as* 9, *asall* を満足しない. 設計検証に必要な時間は, 演算子の数が 3〜86 の RTL 式に対して 0. 6〜70. 6 秒であり, 実用的な時間で検証が行えている. 特に, 設計に誤りがある場合, 非常に早く誤りの存在を検出することができる.

正しい設計の検証の場合, *as* 1〜*as* 9 に対して別々に検証を行うと合計で約 100 秒かかる. これに対し, *asall* は *as* 1〜*as* 9 の論理積であるが, 設計が *asall* を満足することを検証するのに約 70 秒しか必要としない. これは *asall* についての検証では, *as* 1〜*as* 9 を同時に扱っているため, ある部分式に関する情報が他の部分式の真偽判定に使えるためである. しかし, モデルチェッキングで必要となる新たな節点数は, *as* 1〜*as* 9 に対しては合計 123 にすぎないが, *asall* に対しては 915 となる.

6. おわりに

有限オートマトンと等価な表現能力を持つ時相論理として，有限オートマトンの有限長／無限長の動作に着目して，4つのタイプの正則時相論理を示した．また，正則時相論理を用いた順序機械の設計検証問題を取り上げ，εフリー有限正則時相論理による順序機械の設計検証の方法として，設計した順序機械より構造モデルを作成し，仕様のεフリー正則時相論理式の真偽判定を構造モデル上で行うモデルチェック法を示した．このモデルチェック問題の計算の複雑度は，仕様の式の長さに対して非初等的という膨大な複雑さを持つことが判明したが，構造モデルの大きさに対しては線形の時間で動作する効率の良いモデルチェックアルゴリズムを示し，実際にこのアルゴリズムに基づいたモデルチェッカを作成して，実用規模の順序機械の設計検証が実用的な時間と記憶量で実現できることを実証し，形式的論理設計検証への1つのアプローチとして正則時相論理によるモデルチェック法が実用的な意味において有効であることを示した．

謝辞 本研究の遂行にあたり，信号制御器とDMAコントローラの設計データを提供してくださったCMUのE. M. Clarke教授に感謝します．また，種々の有益な御議論をいただいた本学高木直史博士，石浦菜岐佐氏をはじめ矢島研究室の諸氏に感謝します．なお，本研究は一部文部省科学研究費による．

参考文献

1) Browne, M. C., Clarke, E. M., Dill, D. L. and Mishra, B.: Automatic Verification of Sequential Circuits Using Temporal Logic, *IEEE Trans. Comput.*, Vol. C-35, No. 12, pp. 1035-1044 (1986).
2) Clarke, E. M., Bose, S., Browne, M. C. and Grumberg, O.: The Design and Verification of Finite State Hardware Controllers, Technical Report, CMU-CS-87-145, Carnegie Mellon University (1987).
3) Clarke, E. M. and Emerson, E. A.: Synthesis of Synchronization Skeletons for Branching Time Temporal Logic, *Proc. Workshop on Logic of Programs*, pp. 52-71 (1981).
4) Clarke, E. M., Emerson, E. A. and Sistla, A. P.: Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach, Technical Report, CMU-CS-83-152, Carnegie Mellon University (1983).
5) Fujita, M., Tanaka, H. and Motooka, T.: Verification with Prolog and Temporal Logic, *Proc. 6th Int. Symp. Computer Hardware Description Language*, pp. 103-114 (1983).
6) Hiraishi, H.: Design Verification of Sequential Machines Based on a Model Checking Algorithm of ϵ-Free Regular Temporal Logic, Technical Report, CMU-CS-88-195, Carnegie Mellon University (1988).
7) Hiraishi, H.: Design Verification of Sequential Machines Based on ϵ-Free Regular Temporal Logic, *Proc. 9th Int. Symp. Computer Hardware Description Language*, pp. 249-263 (1989).
8) Hughes, G. E. and Cresswell, M. J.: *An Introduction to Modal Logic*, Methuen, London (1977).
9) Moszkowshi, B.: Reasoning about Digital Circuits, Technical Report, STAN-CS-83-790, Stanford University (1983).
10) Rescher, N. and Urquhart, A.: *Temporal Logic*, Springer-Verlag, Wien (1971).
11) Uehara, T., Saito. T., Maruyama, F. and Kawato, N.: DDL Verifier and Temporal Logic, *Proc. 6th Int. Symp. Computer Hardware Description Language*, pp. 91-102 (1983).
12) Wolper, P.: Temporal Logic Can Be More Expressive, *Proc. of 22nd Annual Symposium on Foundations of Computer Science*, pp. 340-348 (1981).
13) Wolper, P., Vardi, M. Y. and Sistla, A. P.: Reasoning about Infinite Computation Paths, *Proc. 24th Symp. on the Foundations of Computer Science*, pp. 185-194 (1983).
14) 浜口，平石，矢島：ω正則集合と等価な表現能力を持つ時相論理，電子情報通信学会技術研究報告，COMP 88-8 (May 1988).
15) 浜口，平石，矢島：∞正則時相論理のωモデルチェッキングアルゴリズムを用いた順序機械の形式的検証，電子情報通信学会技術研究報告，CO-MP 89-24 (June 1989).
16) 平石，浜口，矢島：正則時相論理の充足可能性判定アルゴリズム，情報処理学会論文誌，Vol. 30, No. 3, pp. 366-374 (1989).
17) 平石，矢島：正則集合と表現等価な正則時相論理 RTL，情報処理学会論文誌，Vol. 28, No. 2, pp. 117-123 (1987).

(平成元年8月31日受付)
(平成2年4月17日採録)

平石 裕実（正会員）

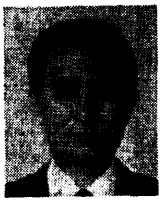
昭和26年生．昭和48年京都大学工学部電子工学科卒業．昭和50年同大学院修士課程（電気工学第二）修了．同年京都大学工学部情報工学教室助手．昭和59年同教室講師．昭和62年同教室助教授．工学博士．論理設計検証，論理設計用CAD，計算機グラフィクス等の研究に従事．電子情報通信学会会員．

浜口 清治（正会員）

昭和39年生．昭和62年京都大学工学部情報工学科卒業．同大学院博士課程在学中．論理設計検証，時相論理の研究に従事．電子情報通信学会会員．

藤井 寛

昭和41年生．平成元年京都大学工学部情報工学科卒業．同大学院修士課程在学中．形式的設計検証，設計検証システム，時相論理の研究に従事．電子情報通信学会会員．



矢島 脩三（正会員）

昭和8年生．昭和31年京都大学工学部電気工学科卒業．同大学院博士課程修了．工学博士．昭和36年より京都大学工学部に勤務，昭和46年情報工学科教授．昭和35年京都大学第一号計算機KDC-1を設計稼動，以来，計算機，論理設計，オートマトン等の研究教育に従事．著書は「電子計算機の機能と構造」(岩波，57年）等．本学会元常務理事，元会誌編集委員(地方)，元JIP編集委員．電子情報通信学会元評議員およびオートマトンと言語研専元委員長，North-Holland出版元IPL編集委員，IEEE Senior Member.