メディア同期プロトコルの機能試験の一手法

山田 誠　森 亮憲　中田 明夫　東野 輝夫
大阪大学大学院基礎工学研究科情報数理系専攻

本稿では，並行に動作するI/O時間オートマトンモデルを用いて音声と動画の同期制御を行なうメディア同期プロトコルを記述し，それに対する機能試験手法を提案する．I/O時間オートマトンの試験において，与えられた試験系列（入出力動作列）を実行可能とするためには，その系列中のすべての動作の時間制約を満足するようなタイミングで入出力動作を実行する必要がある．しかし，試験対象となるシステムからの出力タイミングは外部から制御できないので，試験系列全体を実行可能とするような出力タイミングの時間幅を求める必要がある．本稿では，線形計画法の手法により複数のI/O時間オートマトンの試験系列をともに実行可能とするような入出力タイミング（出力の場合，その時間幅）を機械的に求める方法を考え，それを利用した機能試験法を提案する．

A Method for Functional Testing of Media Synchronization Protocols

Makoto Yamada, Takanori Mori, Akio Nakata and Teruo Higashino
Department of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University

In this paper, we propose a functional testing method of media synchronization protocols, which control the synchronization between audio and movie, described in the parallel I/O timed automaton model. In order to trace a test sequence (I/O event sequence) on the model, we need to execute each I/O event at an adequate timing which satisfies the whole timing constraint in the test sequence. However, the outputs are given from the IUT and uncontrollable. Also each output timing may affect executable timing for the succeeding I/O events in the test sequence. In this paper, we propose a technique to derive the executable timing for given test sequences, and propose a method for functional testing using the technique.

1　まえがき

近年，動画や音声など複数のメディアを扱うマルチメディアシステムが数多く利用されるようになってきている．マルチメディアシステムは実時間システムの一種とみなすことができ，QoSを保証するために，入出力動作などに対して時間制約が課されている．また，こういったマルチメディアシステムは，複数の実時間システムが協調して動作する並行システムとしてモデル化されることが多い[4][5]．従来，実時間システムを記述するためのモデルとして，時間オートマトンや時間ペトリネットなどが考えられており[1]，マルチメディアシステムの仕様記述などに利用されている[3][6]．こういったマルチメディアシステムの信頼性を向上させるための一つの手法として機能試験がある．機能試験は，実装が仕様に記述されている機能に適合しているのかどうかを調べる試験である[7]．

本稿では，音声と画像の同期制御を行うメディア同期プロトコルに対する機能試験の一手法を提案する．通常，メディア同期プロトコルは，音声再生モジュール，動画再生モジュール，制御モジュールの３つのモジュールで構成されている．各再生モジュールは，ネットワークを介して音声データや画像データを受け取る．制御モジュールは適当な間隔で，音声と画像の同期をとる．試験項目として，音声と画像を同期させる動作が正しく実行されるかや，ネットーワークの遅延などによりデータが受け取れない場合に，仕様の記述通りに代替の動作を行うことができるか，などが考えられる．

本稿で扱うメディア同期プロトコルの各モジュールは，I/O時間オートマトン[2]で記述されると仮定する．I/O時間オートマトンは，クロックおよびレジスタを持ち，各状態遷移に対してクロックやレジスタの値を表す変数の線形不等式の論理積からなる論理式を遷移条件として付加することができる．メディア同期プロトコルの機能試験を行うために，各I/O時間オートマトンに対して試験系列（状態遷移系列）を生成すると，各状態遷移に条件式が書かれているので，生成した状態遷移系列が常に実行可能であるとは限らない．このため，状態遷移系列が実行可能であるかどうかを判定する必要がある．また，状態遷移系列が実行可能であれば，その状態

遷移系列を実行するために必要なパラメータ（入出力タイミングなど）を求める必要がある．一般に試験対象システム（IUT:Implementation Under Test）への入力は指定されたタイミングで実行できるが，IUT からの出力タイミングは外部からは制御できない．このため，試験系列の実行可能性を判定する場合，与えられた試験系列全体が実行可能であるための入力タイミングのみならず，出力タイミングの時間間隔を求められることが望ましい．我々の研究グループでは，これまでに単一の I/O 時間オートマトン上の状態遷移系列に対して，その遷移系列の実行可能性を機械的に判定し，実行可能な場合に入力タイミングなどのパラメータ値を機械的に求める手法を提案している [2]．本稿では，この手法と文献 [4] の同期並行システムの時間スケジューリング法のアイデアを拡張し，複数の I/O 時間オートマトン群が協調して動作するような並行モデルに対して，それらの I/O 時間オートマトン群の状態遷移系列の集合がすべて実行可能であるための入力タイミングなどのパラメータ値，および実行時に許される出力タイミングの時間間隔を機械的に求める手法を提案する．

以降，2 章では本稿で扱うモデルについて説明し，3 章で遷移系列の実行可能性を形式的に定義する．4 章では試験系列の生成手法，並びに，実行可能な場合のパラメータ値の決定方法について述べる．5 章では適用結果について説明する．

2 並行 I/O 時間オートマトン群

2.1 I/O 時間オートマトンの定義

$M =< S,\ A,\ I/Otype,\ t,\ V,\ Pred,\ Def,\ \delta,\ s_{init},\ \{x_{1init},\ x_{2init},\ \ldots,\ x_{kinit}\} >$を **I/O 時間オートマトン**と定義する．ここで，

- SはMの状態の有限集合, $S = \{s_0, s_1, \ldots, s_n\}$
- A は Mの入出力動作名の有限集合
- $I/Otype = \{!, ?\} \cup \{?_{\mathbf{v}} | \mathbf{v} \in V\}$.
  記号?,! はそれぞれ入力, 出力を表す．入力動作において，入力値 $\mathbf{v}$を変数に代入し以降の遷移条件などに利用したい場合は$?_{\mathbf{v}}$のように表す. 出力値については, 通常直接以降の遷移条件に影響することがないので, このモデルでは省略する．
- 現在時刻を表すグローバルなクロック変数 t
- Vは変数の有限集合，$V = \{x_1, x_2, \ldots, x_k\}$
- $Pred$ は線形不等式 $P[t, x_1, x_2, \ldots, x_k]$ の**論理積**からなる論理式の集合
- Defは一次式 $f(t, \mathbf{v},\ x_1,\ x_2,\ \ldots,\ x_k)$ の変数 $x_i \in V$への代入を表す代入文 $x_i \leftarrow f(t, \mathbf{v}, x_1, x_2, \ldots, x_k)$ の集合
- δは遷移関数. $S \times A \times I/Otype \times V \rightarrow S \times V$
- $s_{init} \in S$は Mの初期状態
- $\{x_{1init}, x_{2init}, \ldots, x_{kinit}\}$ はそれぞれ変数 x_1, x_2, ..., $x_k \in V$の初期値

である． また, モデル M上の遷移は, $s, s' \in S$, $a \in A$, $\$ \in I/Otype$, $P \in Pred$, $D \subseteq Def$ のとき, $s \xrightarrow{a\$[P]D} s'$と表記する．

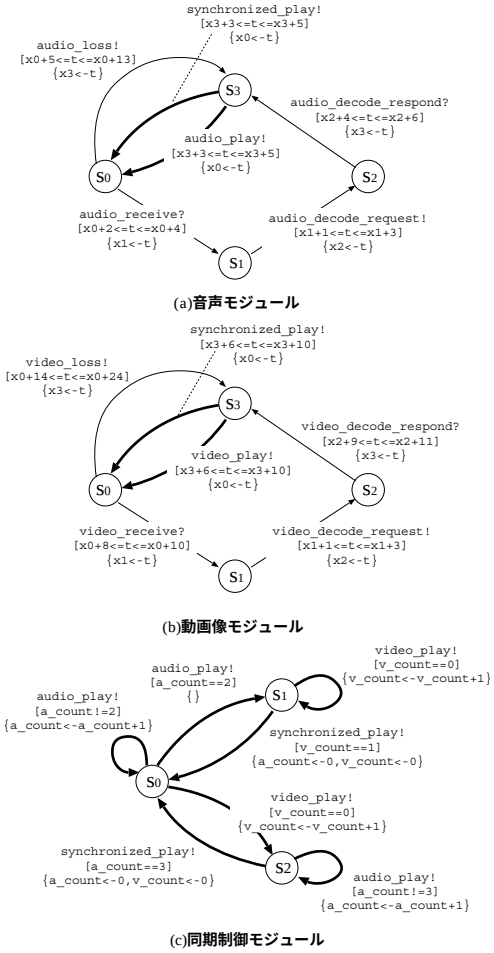

図 1: メディア同期プロトコル

図 1は送信側から送りだされる 2 つのメディア（音声と動画像）を受信し，メディア間の同期をとりながら表示するシステムを I/O 時間オートマトンモデルで記述した例である．(a)，(b) はそれぞれ，送信側から連続して送り出される音声，動画像のフレームを受信すると，外部のデコーダにデ

コードを依頼し，それが終了すると表示するという処理を繰り返す．また，このシステムでは音声の 4 フレーム目と動画の 2 フレーム目で同期をとるようにしており，その制御をモジュール (c) が行なっている．

I/O 時間オートマトンの直観的な意味は次のようになる．図 1(a) の状態 s_0において，遷移条件 $x_0 + 2 \leq t \leq x_0 + 4$ が与えられているので，時刻 2 から 4 の間は入力 audio_receive?が実行できる（x_0の初期値を 0 としている）．これが実行されると (a) の状態は s_1に遷移し，代入 $x_1 \leftarrow t$ によりそのときのクロック t の値が変数 x_1に代入される．この入力の実行時刻が 3.5 の場合 $x_1 = 3.5$ となり，出力 audio_decode_request!は時刻 4.5 から 6.5 の間のある時刻で実行され，状態 s_2へ遷移する．時刻 4 で出力されたとすると x_2には 4 が代入され，次の入力 audio_decode_respond?は $x_2 + 4 = 8 \leq t \leq x_2 + 6 = 10$ を満たすような時刻，例えば 9.5 で実行できる．このとき，出力 audio_play!か synchronized_play!は時刻 12.5 から 14.5 の間で実行されることとなり，x_0にその実行時刻を代入して状態 s_0に戻る．(a) のモジュール全体としては，音声フレームを受信するとデコーダへデコード処理を依頼し，処理された音声フレームを動画フレームと同時に，あるいは独立に出力する処理を行なうものである．(図 1(a) 中の状態 s_0においてフレームを受信できない（入力 audio_receive?が実行されない）場合には，そのフレームは失われてしまったとみなして audio_loss!という出力動作を実行して状態 s_3へ遷移する．)

2.2 並行 I/O 時間オートマトン群

本研究では図 1のように I/O 時間オートマトンが複数並行して動作するモデルによりシステムを記述することを考える．この際各オートマトンにおける時刻はシステム全体で共通のクロック変数 t を参照するものとする．また，同名の遷移動作（入出力動作）は同じ時刻に同期して実行される．

すなわち，図 1のシステムにおいて (a) の synchronized_play!が実行されるときには，(b)，(c) でも synchronized_play!が実行され，(a)，(b) 各モジュールの持つ変数（ここではともに t_0）には同じクロック変数 t の値が代入される．なおこのときの実行条件（遷移条件）は，synchronized_play!に関わるすべてのモジュール，すなわち (a)〜(c) の synchronized_play!の遷移条件の論理積となり，3 つの条件がともに成立するときのみ実行可能となる．

3 使用モデルにおける系列の実行可能性

一般に実時間プロトコルの試験では，テスターから IUT への入力は指定したタイミングで与えることができても，テスターへの出力は IUT の出力タイミングを制御できないため，そのタイミングをあらかじめ固定することはできない．また，そのタイミングに依存して，後続の入出力動作の実行可能時刻が変化する可能性がある．このため試験に際しては，系列に含まれる出力動作が指定された範囲内のどの時点で行なわれても，与えられた試験系列を実行可能とするような入力動作の実行タイミングが存在することが望ましい．我々の研究グループでは，この点を考慮した I/O 時間オートマトンモデル上での適合性試験手法を提案しており，その中で系列の実行可能性について議論している [2]．本章では，文献 [2] で定義されている must/may トレース可能性について紹介し，複数の I/O 時間オートマトンを並行に動作させた場合への拡張を考える．

3.1 一つの系列の実行可能性

I/O 時間オートマトンでは，各遷移の実行時に変数の値を更新していくため，ある遷移系列の実行可能性を判定するためには，各遷移の実行で変数の値がどのように変化していくかを考慮する必要がある．そこで，与えられた遷移系列中の各変数の値や遷移条件を変数の初期値や遷移の実行時刻，入力値を表す変数からなる式（シンボリックトレース）に置き換える [2].

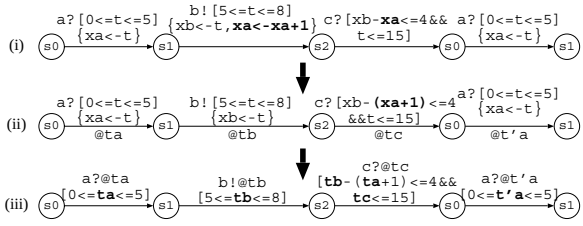

図 2: シンボリックトレース

図 2(i) の系列に関してシンボリックトレースを構成するには，まず変数への代入に着目し，各変数を先行する動作の実行時刻，変数の初期値あるいは入力値を表す変数で置き換える．次に，得られた (ii) のような系列に対して各動作の実行時刻を表す変数を用意し（図中 t_a，t_b，t_c），各入出力動作の遷移条件をこれらの変数で表現する．最終的に，

図 2の系列のシンボリックトレースは $a?@t_a[0 \leq t_a \leq 5], b!@t_b[5 \leq t_b \leq 8], c?@t_c[t_b - (t_a + 1) \leq 4 \leq t_c \leq 15], a?@t'_a[0 \leq t'_a \leq 5]$ となる．

このようにして生成されたシンボリックトレース wに対して，w中の各出力動作がどんなタイミングで実行されたとしても，後続の系列を実行できるような適当な各入力動作の実行タイミングが存在するとき，wは **must トレース可能**であるという．一方，w中の各出力動作がある適当なタイミングで実行されたときに限り，後続の系列が実行可能となるような適当な各入力動作の実行タイミングが存在するとき，wは **may トレース可能**であるという．

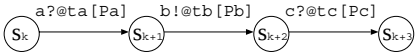

図 3: 1 つの系列の must トレース可能性

図 3の例を用いて，must トレース可能性の定義を示す．図 3の系列は，a という入力動作，b という出力動作を経た後，c という入力動作を行なう系列（シンボリックトレース）である．なお各遷移に添えられた記号については，その実行時刻（t_a，t_b，t_c）とその遷移条件（$[P_a]$，$[P_b]$，$[P_c]$）をそれぞれ表すものとする．この系列の must トレース可能性の判定を行なうために，次のように系列の後から must トレース可能であるための条件を表す式を構成してゆく．まず，$c?$は入力であるため，この動作を可能とする実行タイミングが存在しさえすればよく，それを表す式は $\exists t_c[P_c]$ となる．この入力動作 $c?$に先行する系列については，この式が成立すればこれ以降の動作は実行可能であると考えることができる．次の $b!$という動作に関しては出力であることから,「その出力動作（b）が可能となる実行タイミングが存在し，かつ，その実行タイミングによらず後続の系列は実行可能」という条件を付加する．これを表す式は，$\exists t_b[P_b] \wedge \forall t_b[[P_b] \Rightarrow \exists t_c[P_c]]$ となりこの式が，出力動作 $b!$以降が実行可能となる条件となる．最後に先頭の入力動作 $a?$に関しては，$a?$の実行タイミングが存在し，かつ，後続の系列も実行可能となるような条件を考える．

$\exists t_a[[P_a] \wedge \exists t_b[P_b] \wedge \forall t_b[[P_b] \Rightarrow \exists t_c[P_c]]]$

なる式がそれを表す式となり，結果この式が成り立つとき，かつそのときに限り図 3の系列は must トレース可能であると判定される．

なお may トレース可能性に関しては，図 3の系列がすべて入力であるとみなして判定式を構成する．

3.2 並行に動作する系列の実行可能性

複数の I/O 時間オートマトンが並行して動作する仕様における系列の must トレース可能性の定義も 1 系列の場合と類似した例を用いて述べる．

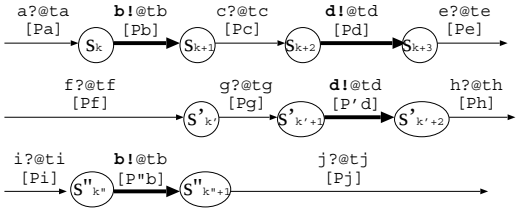

図 4: 並行に動作する系列の must トレース可能性

図 4は，3 つの I/O 時間オートマトンが並行に動作する仕様から得られた系列の例である．同名の遷移動作（図中 $b!$と $d!$ の出力動作）は同期して（同時に）実行される．このような複数の系列の実行可能性に関しても，3.1 節の場合と同様に系列の後から判定式を構成してゆく．

まず，各系列の最後尾の動作 $e?$，$h?$および $j?$ はそれぞれ独立して実行される入力動作であるため，これらすべての最後尾の動作が（別々の時刻に）実行可能となるための条件は，$\exists t_e[P_e] \wedge \exists t_h[P_h] \wedge \exists t_j[P_j]$ となる．これは $\exists t_e \exists t_h \exists t_j[[P_e] \wedge [P_h] \wedge [P_j]]$ と等価な式であり，この式をφ_1とおく．次に，この 3 つの入力動作の最も直前に実行される同期出力動作 $d!$ について,「この出力動作がともに実行可能なタイミング t_dが存在し，かつ t_dが実行可能なタイミングであるならば後続の系列は常に実行可能である」という条件を考える．この条件はすなわち $\exists t_d[[P_d] \wedge [P'_d]] \wedge \forall t_d[[P_d] \wedge [P'_d] \rightarrow \varphi_1]$（$\varphi_2$とおく）となり，この条件が満たされるときに限り $e?$，$h?$および $j?$についてもその実行が可能となる．さらに，$c?$と $g?$の各々独立して実行される入力動作について，$\exists t_c \exists t_g[[P_c] \wedge [P_g] \wedge \varphi_2]$ という条件（φ_3とおく）が得られる．以降，同様に条件式を構成してゆき，同期して実行される出力動作 $b!$については $\exists t_b[[P_b] \wedge [P''_b]] \wedge \forall t_b[[P_b] \wedge [P''_b] \rightarrow \varphi_3]$（$\varphi_4$とおく）が得られ，最終的には $\exists t_a \exists t_f \exists t_i[[P_a] \wedge [P_f] \wedge [P_i] \wedge \varphi_4]$ なる条件（$TrCond$ とおく）が，入力動作 $a?$，$f?$および $i?$を考慮して構成される．すなわち，この $TrCond$ が真となるときに限り，図 4の 3 つの系列は同時に実行可能な系列となる．

4 提案する機能試験法

一般に機能試験では，仕様に記述された様々な機能を満たすようにIUTが動作するかどうかをチェックする．このとき試験項目となる機能には，出力動作の実行タイミングに依存する（先行する出力動作の実行タイミングで遷移先が変わる 等）ものも多く含まれ，すべての試験系列についてmustトレース可能となるような系列を得ることは難しいと考えられる．

本稿で提案する試験手法では，文献 [4] で提案されている静的スケジューリングのアイデアを利用し，試験系列について少なくとも実行可能となるような入出力タイミングが存在するかどうかを調べ，存在するならばその入出力タイミング（出力に関してはその範囲）を求める．これはmayトレース可能性の判定にほかならないが，系列が実行可能となる出力タイミングの時間間隔ができるだけ大きくなるようにすることで，mustトレース可能ではなくとも実行可能性に対する出力タイミングからの影響を小さくできる．また，あらかじめ系列が実行可能となる出力タイミングの間隔の情報を得ることで，機能試験時に試験系列が実行不可能になったことをいち早く検出することができる．

4.1 並行I/O時間オートマトン群の試験系列

本手法ではまず，試験を行ないたい機能項目について，その機能を確認できる系列を並行I/O時間オートマトン群から抽出し，各変数の値の更新を考慮した系列（シンボリックトレース）へと変換する．それらの系列から得られる，各動作の実行時刻に関する制約条件に対して以降で説明する文献 [4] のスケジューリング手法に基づいた処理を行ない，存在すれば，実行可能となる各系列の入出力タイミングを得る．

図1のメディア同期プロトコルについて，音声と映像の同期（同時出力）動作に関して試験を行なう場合には，(a)，(b)，(c) 各モジュールについて同期動作synchronized_play!に関わる系列を抽出してシンボリックトレースに変換する（なお図 1(c) のモジュールについては，回数をカウントすることによる同期制御を行なうのみで時間制約を課していないため，以降の説明では省略する）．

ここで考慮する系列のシンボリックトレースは図5の2つ．これらは，図 1(a)，(b) それぞれのモジュールでs_0（初期状態）からs_1，s_2，s_3を経てs_0へ戻るという動作を，synchronized_play!が実行されるまで（(a) は4回，(b) は2回）繰り返し，最後にs_1へ遷移するという系列のシンボリックトレースである．各遷移$a@t_a$の遷移条件は，a の実行時刻t_aや，先行する動作の実行時刻を表す変数などからなる線形不等式の論理積で表されると仮定しているので，これらの式は$\alpha \leq t_a$または$t_a \leq \beta$の形の式の論理積で表される．この2つの系列の実行タイミングの制約条件を表す論理積について次のような処理を行い，実行可能な入力タイミングと出力タイミングの時間間隔を求める．

(1) 各動作は，それに先行する動作よりも先に実行されることはない．すなわち次の式が成り立つべきであり，これを制約に加える．
$$t_{a1} \leq t_{a2} \leq t_{a3} \leq \ldots \leq t'''_{a0} \leq t''''_{a1}$$
$$t_{v1} \leq t_{v2} \leq t_{v3} \leq \ldots \leq t'_{v0} \leq t''_{v1}$$

(2) 出力動作の実行タイミングt_xについては，その最大，最小を表す変数t_{xmin}，t_{xmax}を新たに用意し，系列の制約条件$\alpha \leq t_x \leq \beta$を$\alpha \leq t_{xmin} \leq t_{xmax} \leq \beta$と置き換える．

なお，同期して実行される出力動作（上の系列の synchronized_play!）の実行タイミングに関しては，t'''_{a0}とt'_{v0}に関する制約を同じ変数の組（t_{0min}，t_{0max}）で置き換える。すなわち，音声モジュールにおける系列の synchronized_play!に対する制約条件は$[t'''_{a3}+3 \leq t_{0min} \leq t_{0max} \leq t'''_{a3}+5]$となり，動画モジュールでは$[t'_{v3}+6 \leq t_{0min} \leq t_{0max} \leq t'_{v3}+10]$となる．

(3) 線形計画法によりこれらの制約から得られる線形不等式を解く際の目的関数として，出力動作の実行タイミングの時間間隔の合計$\Sigma(t_{imax}-t_{imin})$が最大となるようにする（各$t_{imin}$，$t_{imax}$は同期出力動作の実行タイミングの間隔を表す変数）．

(4) 線形計画法により解を求める．

このようにして得られた解が，系列を実行可能とする入出力動作の実行タイミングとなる（出力動作は，$[t_{imin}, t_{imax}]$の範囲であれば以降の系列が実行可能と判断する）．なお，解が得られない場合はその系列は実行不可能と判断する．

5 適用例

図1のメディア同期プロトコルに対して，同期動作およびタイムアウト処理の試験を行なう系列に

モジュール (a) の系列	モジュール (b) の系列
audio_receive?@$t_{a1}[t_{a0init}+2 \le t_{a1} \le t_{a0init}+4]$	video_receive?@$t_{v1}[t_{v0init}+8 \le t_{v1} \le t_{v0init}+10]$
audio_decode_request!@$t_{a2}[t_{a1}+1 \le t_{a2} \le t_{a1}+3]$	video_decode_request!@$t_{v2}[t_{v1}+1 \le t_{v2} \le t_{v1}+3]$
audio_decode_respond?@$t_{a3}[t_{a2}+4 \le t_{a3} \le t_{a2}+6]$	video_decode_respond?@$t_{v3}[t_{v2}+9 \le t_{v3} \le t_{v2}+11]$
audio_play!@$t_{a0}[t_{a3}+3 \le t_{a0} \le t_{a3}+5]$	video_play!@$t_{v0}[t_{v3}+6 \le t_{v0} \le t_{v3}+10]$
audio_receive?@$t'_{a1}[t_{a0}+2 \le t'_{a1} \le t_{a0}+4]$	video_receive?@$t'_{v1}[t_{v0}+8 \le t'_{v1} \le t_{v0}+10]$
$\vdots$	$\vdots$
synchronized_play!@$t'''_{a0}[t'''_{a3}+3 \le t'''_{a0} \le t'''_{a3}+5]$	**synchronized_play!**@$t'_{v0}[t'_{v3}+6 \le t'_{v0} \le t'_{v3}+10]$
audio_receive?@$t''''_{a1}[t'''_{a0}+2 \le t''''_{a1} \le t'''_{a0}+4]$	video_receive?@$t''_{v1}[t'_{v0}+8 \le t''_{v1} \le t'_{v0}+10]$

図 5: 同期出力動作 synchronized_play!の試験系列（シンボリックトレース）

ついて本手法を適用した.

5.1 同期動作の試験系列

図 5の 2 つのシンボリックトレースに対して適用したところ，次のような解が得られた（時間間隔が得られているものは，出力動作のタイミングである）.

$t_{a1}=2$	$3 \le t_{a2} \le 5$	$t_{a3}=9$	$12 \le t_{a0} \le 14$
$t'_{a1}=16$	$17 \le t'_{a2} \le 19$	$t'_{a3}=23$	$26 \le t'_{a0} \le 28$
$t''_{a1}=30$	$31 \le t''_{a2} \le 33$	$t''_{a3}=37$	$40 \le t''_{a0} \le 42$
$t'''_{a1}=44$	$45 \le t'''_{a2} \le 47$	$t'''_{a3}=51$	$54 \le t_0 \le 56$
$t''''_{a1}=58$			
$t_{v1}=8$	$9 \le t_{v2} \le 11$	$t_{v3}=20$	$26 \le t_{v0} \le 28$
$t'_{v1}=36$	$37 \le t'_{v2} \le 39$	$t'_{v3}=48$	$54 \le t_0 \le 56$
$t''_{v1}=64$			

入力をこのタイミングで与え，出力がこの範囲に収まるタイミングで実行されればこの系列は実行可能となり，IUT の振舞いをチェックすることができる．すなわち同期出力動作の試験が可能となる.

5.2 タイムアウト処理の試験系列

タイムアウト処理については，タイムアウトの遷移が行なわれた後の同期動作が正しく行なわれれば，タイムアウト処理そのものも正しく行なわれたと見なせると考える.

ここでは同期動作の試験を行なう図 5の系列のうち，モジュール (a) の系列の一部（3 周目の audio_receive?@t''_{a1}，audio_decode_request!@t''_{a2}，audio_decode_respond?@t''_{a3}）をタイムアウト処理（audio_loss!@t''_{a3}）で置き換えたものについて適用し，以下のような結果を得た.

$t_{a1}=2$	$3 \le t_{a2} \le 5$	$t_{a3}=9$	$14 \le t_{a0} \le 16$
$t'_{a1}=18$	$19 \le t'_{a2} \le 21$	$t'_{a3}=25$	$28 \le t'_{a0} \le 30$
	$35 \le t''_{a3} \le 37$		$40 \le t''_{a0} \le 40$
$t'''_{a1}=44$	$45 \le t'''_{a2} \le 47$	$t'''_{a3}=51$	$54 \le t_0 \le 56$
$t''''_{a1}=58$			
$t_{v1}=8$	$9 \le t_{v2} \le 11$	$t_{v3}=20$	$26 \le t_{v0} \le 28$
$t'_{v1}=36$	$37 \le t'_{v2} \le 39$	$t'_{v3}=48$	$54 \le t_0 \le 56$
$t''_{v1}=64$			

6 あとがき

本稿では，複数の I/O 時間オートマトンが協調して並行に動作するモデルによりメディア同期プロトコルを記述し，それに対する機能試験の手法を提案した．また，線形計画法を利用した静的スケジューリングの手法を利用して，試験系列を実行可能とする出力タイミングの間隔をできるだけ広くする方法を考案した．実用的なメディア同期プロトコルに対して，本手法を適用し，その有効性を評価することなどが今後の課題である.

参考文献

[1] R. Alur and D. L. Dill : “A theory of timed automata”, Theoretical Computer Science, Vol. 126, pp.183-235 (1994).

[2] T. Higashino, A. Nakata, K. Taniguchi and A. R. Cavalli : “Generating test cases for a timed I/O automaton model”, Proc. of 12th IFIP Workshop on Testing of Communicating Systems (IWTCS’99), pp.197-214 (Sept. 1999).

[3] C. M. Huang and C. Wang : “Synchronization for Interactive Multimedia Presentations”, IEEE MULTIMEDIA, Vol. 5, No. 4, pp.44-62 (Oct.-Nov. 1998)

[4] H. Katagiri, M. Kirimura, K. Yasumoto and T. Higashino and K. Taniguchi : “Hardware Implementation of Concurrent Periodic EFSMs”, Proc. of Joint International Conference on 13th Formal Description Techniques and 20th Protocol Specification, Testing, and Verification (FORTE/PSTV2000), pp.285-300 (Oct. 2000).

[5] D. Lee and M. Yannakakis : “Principles and Methods of Testing Finite State Machines - A Survey”, Proc. of the IEEE, Vol. 84, No. 8 (1996)

[6] T. D. C. Little and A. Ghafoor : “Synchronization and storage models for multimedia objects”, IEEE Journal of Selected Areas in Communications, Vol. 8, No. 3, pp.413-427 (Apr. 1990)

[7] V. Misic, S. T. Chanson and S. C. Cheung : “Towards a Framework for Testing Distributed Multimedia Software Systems”, Proc. International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE98) (Apr. 1998).