同期通信可能な周期 EFSM 群のハードウェア構成法

片桐久晶[†] 桐村昌行[†] 安本慶一[††]
中田明夫[†] 東野輝夫[†] 谷口健一[†]

本論文では，複数モジュールの並行動作と協調，時間制約を扱う並行周期 EFSM 群を提案し，そのモデルを用いて記述された実時間システムの形式仕様からハードウェア回路を自動導出する手法を提案する．提案モデルでは，複数の並行に動作する EFSM 間で，イベントの同期実行によるデータ交換（マルチランデブと呼ぶ）が可能で，各イベントの実行条件として先行イベント系列の実行時刻や定数を含む線形不等式で表される時間制約が指定可能である．ただし，すべての EFSM はある一定時間周期で初期状態に戻ると仮定している．提案する導出法では，仕様に書かれたすべてのイベント系列を実装し，先行イベントの実行時刻に従って，今後スケジュール可能（実行可能）な分岐のみを選択実行する回路を生成する．また，各 I/O イベントがあらかじめ求められた実行時間範囲内に実行された場合，後続のいずれかの分岐が時間制約を満たしながら実行可能であることを保証している．提案手法に基づき，与えられた仕様から VHDL 記述を自動生成するツールを作成し，複数個の動画の同時再生用チップの仕様を与えた結果，生成される回路が面積・速度ともに実用上問題ないことを確認した．

Hardware Implementation of Concurrent Periodic EFSMs with Multi-way Synchronization

Hisaaki Katagiri,[†] Masayuki Kirimura,[†] Keiichi Yasumoto,[††]
Akio Nakata,[†] Teruo Higashino[†] and Kenichi Taniguchi[†]

This paper proposes a concurrent periodic EFSMs model and a technique to synthesize hardware circuits from specifications of real-time systems described in this model. In the proposed model, data exchange by synchronous execution of the same events in multiple EFSMs (called multi-way synchronization) can be specified like LOTOS. The executable time range of each event can be given as a logical conjunction of linear inequalities of the execution time of its preceding events, constants and integer variables where some values are input from environments. Here, we assume that every event sequence starting from the initial state in each EFSM returns to the initial state in a specified time interval. Since each EFSM has some branches and some combination of branches of those EFSMs may not be executable because of their timing constraints, the proposed synthesis technique finds only executable combination of branches from a given specification and generates a schedule table (scheduler) for event sequences in each executable combination of branches. We have developed a tool to generate the corresponding RTL-level VHDL specification from a given specification, and generated circuits from some specifications such as a video playback chip. From those experiments, we have confirmed that the performance and size of the generated circuits are reasonable for practical use.

1. まえがき

高速通信ネットワークの発展にともない，その上で動作する通信システムをハードウェアとして実装し高速化する手法の研究が注目されている．一方で，情報通信サービスの多様化により，並列動作の扱いや，各動作があらかじめ与えられた時間制約を満足する時刻に実行されること（リアルタイム性）の保証が要求されるようになってきている．通信システムの開発において，通常，最終的な製品版を得るまでには，その内部構成，および，各処理や I/O の実行タイミングなどのパラメータに何度も変更が加えられ，シミュレーションなどによる性能評価が行われる．このため，実装の初期段階では，システムの形式仕様からハード

† 大阪大学大学院基礎工学研究科情報数理系専攻
Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University
†† 滋賀大学経済学部情報管理学科
Faculty of Economics, Shiga University

ウェア回路を自動合成し素早く性能評価を行うラピッドプロトタイピングの手法として，高位合成に関する研究が数多くなされてきた[11]．

文献 1)，13) では，それぞれ仕様記述言語である SDL や Estelle を用いて，並行に動作する EFSM 群とその間の 1 対 1 の非同期通信としてモデル化された動作仕様のハードウェア化手法が提案されている．また，LOTOS[5]に基づき，複数並行プロセスとその間のイベントの同期実行による同期通信によりモデル化された動作仕様からハードウェア回路を構成する方法も提案されている[7),10),12)]．しかし，これらの手法は，いずれも各動作に時間制約を指定することができず，リアルタイム性を要求するシステムの回路化に直接適用することはできない．一方，文献 2) では，いくつかの並行プロセス（タスクグラフとして与えられる）と，使用可能なプロセッサの種類と数，各タスクのプロセッサごとの処理時間などが与えられたときに，すべてのタスクの処理が，あらかじめ与えられたデッドラインまでに完了し，かつ，使用するプロセッサのコストが最小となるようなスケジューリングの方法を提案している．しかし，並行プロセス間の通信を扱っておらず，また，I/O 動作など外部環境に依存して後続タスクの実行開始時刻が変動するようなケースを考慮していないため，通信システム用回路の設計・開発に適用するのは困難である．

本論文では，複数モジュールの並行動作と協調，および時間制約を簡潔に扱えるモデルとして，並行周期 EFSM 群を提案し，そのモデルを用いて記述された実時間システムの形式仕様からハードウェア回路を自動導出する手法を提案する．提案するモデルでは，複数の並行に動作する EFSM 間でのイベントの同期実行によるデータ交換（マルチランデブと呼ぶ）が可能で，各イベントの実行条件として先行イベント系列の実行時刻や定数を含む線形不等式で表される時間制約が指定可能である．ただし，各 EFSM は，初期状態からどのイベント系列を実行しても必ず初期状態に戻り，かつ，どの系列も一定の周期（たとえば 10 秒）で初期状態に戻るように時間制約が与えられているものと仮定する．

このような並行周期 EFSM 群を実現する回路を生成する際には，(1) 回路における時間制約の保証，(2) 選択実行の自由度，(3) 外部との I/O にかかわるイベントの実行時間の許容範囲，などが問題になる．提案する導出法では，仕様に書かれたすべての実行可能な系列を実現し，かつ，先行イベントの実行時刻に従って，今後スケジュール可能な（時間制約どおりに実行可能な）イベント系列のみを動的に選択実行する回路を生成する．また，各 I/O イベントがあらかじめ求められた範囲内に実行された場合，後続のいずれかの分岐のイベント系列が時間制約を満たしながら実行可能であることを保証する．

スケジュール可能性の保証は，各 EFSM 内でのイベントの実行順序，EFSM 間でのイベントの同期関係，およびイベントに付加された時間制約式などの情報をもとに線形不等式を構成し，線形計画法の解法を用いて，各イベントの実行タイミングを求めることにより行う．この際，各イベントの実行時刻の下限値，上限値を表すパラメータからなる制約条件を組み立て，各イベントの実行時刻の上限値と下限値の差の総和を最大化するように線形計画問題を解くことにより，各イベントの実行時刻に対する許容範囲が大きくなるよう工夫した．

与えられた並行周期 EFSM 群は，文献 6)，12) で我々が提案した手法に基づいて，各 EFSM につき同じクロックで動作する同期式順序回路と，その間のマルチランデブを制御するための組合せ論理回路として実現する．また，スケジュール不可能（実行不可能）となったイベント系列が選択実行されないよう指示する回路を構成し，組み込んだ．並行周期 EFSM 群の動作仕様から，各イベントの実行時刻の許容範囲を求めるツール，および，それらの情報からハードウェア記述言語 VHDL[3]によるレジスタ転送レベルの回路記述を自動生成するツールを作成した．評価実験として，1 つのデコーダを共用する複数の動画の再生支援用チップの動作仕様を，作成したツールに与えて VHDL による回路記述を生成し，市販の論理合成ツールを用いてネットリストを生成した．その結果，回路の面積，速度ともに実用上問題ないことを確認した．

2. 並行周期 EFSM 群の定義

各 EFSM を次の 5 字組 $\langle S, I, V, clock, \delta, init\rangle$ で定義する．ここで，S は状態の有限集合，I は状態遷移時に実行するイベントの有限集合，V は変数の有限集合，$clock$ は各状態にとどまっている経過時間を表すカウンタ，δ は遷移規則の有限集合，$init$ は初期状態と各変数の初期値の組である．通常の EFSM モデルと違い，各状態に遷移するたびに 0 にリセットされ，遷移が実行され次の状態に移るまでの経過時間をカウントする $clock$ を持つ．イベントの実行は瞬間的に行われると仮定する．システムの外部と入出力を行うイベントを **I/O イベント**と呼び，他の EFSM のイベントと同期して実行されるイベントを**同期イベント**と

呼ぶ．イベントの実行によるデータの入力・出力は，LOTOS[5]に倣い，それぞれ “$a?x$”，“$b!E$” のように表す．ここで，“a” や “b” は入出力用のゲートの名前であり，“$?x$” は入力値の変数 x への代入，“$!E$” は式 E の計算結果の出力を表す式である．なお，複数データの同時入出力（“$a?x!10$” など）も記述できる．

また，LOTOS に時間制約を記述できるように拡張した E-LOTOS[4]に倣い，“$a@?t$” により，イベント a 実行時の $clock$ の値（a の**遅延時間**と呼ぶ）を時間変数 t に取得できる．たとえば，ある状態 S において，“$a@?t$” が実行されたときに，t に 5 が代入された場合，状態 S に移ってから “$a@?t$” が実行されるまでに 5 単位時間かかったことを表す．

各変数の型として，整数型や論理型，時間型など VHDL で実装可能な任意の型を指定できる．各遷移規則は，5 字組 $\langle s_{cur}, s_{next}, a, guard, assign\rangle$ $(s_{cur}, s_{next} \in S, a \in I)$ で定義する．ここで，$guard$ は定数，EFSM 内の変数値，他の EFSM のイベントからの出力値（同期イベントの場合）とその上の演算からなる論理式で指定される遷移条件であり，$assign$ はイベントの同期実行時の任意の変数への代入文の集合である．

状態 s_{cur} において，遷移条件 $guard$ が成立し，かつ，イベント a が実行可能であるとき，イベント実行後に状態 s_{next} に移る．その際に，$assign$ で指定された変数への代入が行われる．$clock$ の値を遷移条件に用いることで，イベントの遅延時間に制約を加えることができる．たとえば，“$a?x@?t[(2 \le t) \wedge (0 \le x) \wedge (x \le 10)]$” は，イベント a が現状態で 2 単位時間以上待機した後でかつ，ゲート a から入力された整数が $0 \le x \le 10$ を満たしていれば実行される．実行後，変数 x に入力された整数が代入される．時間制約には，定数と過去に実行されたイベントの遅延時間および整数変数を含む一次不等式の論理積のみが指定できる．各状態において複数個の実行可能なイベントが存在する場合は，その中からただ 1 つのイベントが非決定的に選択実行されるものとする．

与えられた EFSM が以下の制約をともに満たす場合，その EFSM を並行周期 EFSM と呼ぶことにする．

- 初期状態からどの経路（イベント系列）をたどっても，再び初期状態に戻る経路がある．
- 初期状態からどの経路をたどっても，ある一定の時間（周期）内に，初期状態に戻るように遷移条件が付加されている．

図 1 の左のグラフはある並行周期 EFSM を表したものである．この EFSM ではまず，ゲート a からある値が入力されると，その値とその時点の clock の値（遅延時間）がそれぞれ $x1$，$t1$ に代入される．次の状態では，イベント b，d に付加された時間制約により，$x1$ 単位時間経過するまではイベント d のみが実行可能であり，それ以後はイベント b のみが実行可能となる．もし，イベント b が実行されれば，イベント b の遅延時間が時間変数 $t2$ に代入され，イベント $c!x1@?t3$ が実行可能となる．

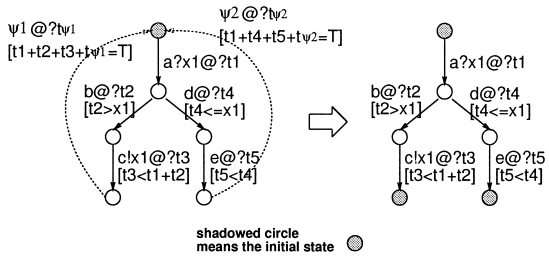

図 1 並行周期 EFSM
Fig. 1 Periodic EFSM.

周期 T でこの EFSM が初期状態に戻るようにするため，すべての初期状態から初期状態に至る経路（イベント系列）の最後の遷移として特別な仮遷移 ψ を導入する．周期 T でこの EFSM が初期状態に戻るように ψ の遷移条件に時間制約を与える．図 1 において，そのような仮遷移 ψ_1，ψ_2 を破線で示す．以下では簡単のため，図 1 の右側に示すように仮遷移の記述を省くことにする．

2.1 EFSM 間の同期指定

EFSM 間の同期関係は，LOTOS の並列オペレータを用いて以下のように指定する．

$S ::= S\ |[gate\_list]|\ S \ \mid\ S\ |||\ S \ \mid\ (S) \ \mid\ EFSM$

ここで，$EFSM$ は EFSM の名前であり，$gate\_list$ にはオペレータの両側で同期させたいイベントのゲートの並びを指定する．また，$|||$ は，オペレータの両側でイベントを同期させる必要がないことを指定する（$gate\_list = \emptyset$ に相当）．たとえば，図 4 の $M1|[a,b]|(M2|[b]|M3)$ では，イベント a は $M1$ と $M2$，あるいは $M1$ と $M3$ の間で同時に実行されなければならない．$M1$ と $M2$ の間でイベント a が同期実行される場合，イベント $a!(x1-2)@?t3$ の出力値 $(x1-2)$ がイベント $a?x3@?t10$ の変数 $x3$ に代入される．イベントの同期実行のための条件についての詳細は，LOTOS と同様であるため，ここでは省略する[5]．

3. 回路化の基本方針

仕様から回路を生成する際には，元の仕様上に記述

されている時間制約に対する保証をどのように実現するかが問題となる．たとえば，次のイベント系列

$$s1 \xrightarrow{a@?t_a[t_a \geq 2]} s2 \xrightarrow{b@?t_b[t_a+t_b \leq 5]} s3$$

において，イベント a の時間制約 $t_a \geq 2$ は，イベント a は状態 $s1$ での遅延時間が 2 以降いつでも実行可能であることを許しているため，たとえば，遅延時間 10 のときにイベント a を実行できる．ところがこの場合，次のイベント b の時間制約 $t_a + t_b \leq 5$ が満たせなくなる．一方，a が遅延時間 2 で実行された場合，b の実行は 0〜3 の遅延時間で実行できる．

イベント系列内のすべてのイベントが時間制約を満たしながら実行できる（以降，このことを**スケジュール可能**であると呼ぶ）ようにするためには，連続するイベント群が各々の時間制約を満たすように実行時間を決める必要がある．そこで提案手法では，イベント系列をスケジュール可能とするような各イベントの遅延時間の許容範囲をあらかじめ導出することにする（一般に，このような範囲は仕様と比べて狭い範囲になる）．

なお，先の例のように，先行するイベントの遅延時間に依存して，後続のイベントの遅延時間の許容範囲が変化する場合がある．上の例では，イベント a が遅延時間 h $(2 \leq h \leq 5)$ で実行されると後続イベント b は遅延時間が 0 から $(5-h)$ の間でいつでも実行可能である．このような動的スケジューリングをハードウェア回路として実装するのは必ずしも容易ではないため，提案手法では，各イベントがある範囲のどの時点で実行されても，後続のイベントの遅延時間の許容範囲が変化しないような範囲を求めることにする．たとえば，上の例では，$2 \leq t_a \leq 3$，$0 \leq t_b \leq 2$ のようにスケジューリングすれば，このイベント系列はつねに実行可能である．

以上のような静的スケジューリングを行うことにより，各イベントが実際に実行された時刻を記憶する必要がなくなり，現在時刻を表すグローバルなタイマと複数個の比較器のみで各イベントの時間制約が満たされるかどうかを判定することが可能となる．

3.1 生成する回路に対する要求

本手法では，以下の要求を満たすような回路を生成する．

- 仕様上に記述されているすべての実行可能な系列を実現する（仕様上に記述されているどの系列に対しても，その系列をスケジュール可能とするような各イベントの遅延時間の許容範囲が存在するようにする）．
- 各 I/O イベントの遅延時間の許容範囲をできるだけ大きくする．

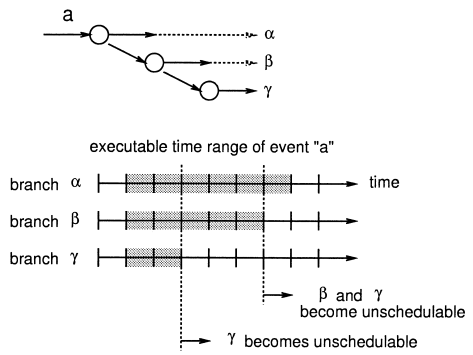

図 2 分岐が含まれる場合の先行イベントの遅延時間
Fig. 2 Executable time range of event followed by branches.

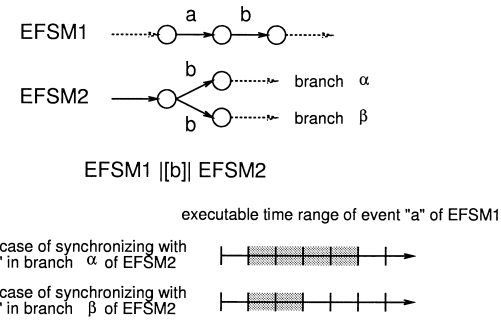

図 3 同期の組合せが複数ある場合のイベントの遅延時間
Fig. 3 Execution time range of event depending on selection of multi-way synchronization.

ただし，**図 2** の例のように，ある I/O イベント a の後に分岐がある場合，どの系列を選択するかによって，イベント a の遅延時間の許容範囲は異なる可能性がある．また，**図 3** の例のように，ある I/O イベント a の後に同期イベント（この例では，イベント b）があり，かつ，同期可能な相手が複数存在する場合（この例では，分岐 α，β 上のイベント b），どの相手と同期を行うかに依存して，イベント a の遅延時間の許容範囲は異なる可能性がある．

そこで提案手法では，各 EFSM の初期状態から初期状態に至るすべての選択可能なイベント系列の組合せそれぞれに対して，各イベントの遅延時間の許容範囲をあらかじめ導出しておき，生成する回路において，各 I/O イベントの実際の遅延時間に従って，スケジュール不可能となった系列上のイベントが選択実行されないように制御する（図 2）．

4. イベントの遅延時間の許容範囲の導出

本章では，EFSM 内の各イベントの遅延時間の許容

範囲を求めるため，EFSM 間で同期実行されるイベント，および EFSM 内でのイベントの実行順序と各イベントの時間制約を，各イベントの遅延時間を表す変数間の線形不等式として構成する．入力変数がガード式やイベントの時間制約に参照されている場合，イベント系列の実行に影響するので，入力変数についても許容範囲を求めることにする．

4.1 同期イベント組の抽出

EFSM 群の初期状態組（各 EFSM の状態がいずれも初期状態である状態組）から実行可能なイベントを順次トレースすることにより，同期イベント組に関する情報（どの EFSM 間でどの同期イベントどうしが同期実行される可能性があるのかの情報）の導出を行うことができる．このような同期を含む並行プロセス間の可達解析の方法は，従来からいくつかの方法が提案されており[9]，それらの方法をそのまま用いることも可能である．しかし，可達解析は，最悪の場合で，n^m 個のグローバル状態を探索することになり（n は EFSM の状態の数，m は同期する EFSM の個数），また，値や遷移条件の成立可否の判定を行うのは効率が悪い．

提案モデルでは，2 章で述べた制限から，すべての EFSM はある一定周期で，初期状態へ戻るイベント系列を 1 つ実行するので，EFSM 間で同期するイベントの組合せは各周期ごとにつねに一定である．そのため，イベント系列において，同じ順番で出現するイベントの組合せのみ考えれば，同期イベント組を算出できる．上記の同期イベント組を求める時間計算量は，1 つのイベント系列に含まれるイベントの最大数を N，EFSM 内で分岐しているイベント系列の最大数を B とすると，$N \cdot B^m$ で抑えられる．ここで，一般に，$N \cdot B^m \ll n^m$ である．

4.2 線形計画問題の構成とイベントの実行時刻の許容範囲の導出

3.1 節で述べたように，システム外部との I/O にかかわるイベントの実行時刻（初期状態からの経過時間）の許容範囲を求めるために，線形計画問題のパラメータとして，実行時刻の下限値と上限値を表す変数を用いる．以降，イベント a の実行時刻を表すパラメータを T_a とし，その下限値，および上限値を表すパラメータをそれぞれ $T_{a_{min}}$，$T_{a_{max}}$ と表記することにする．また，同様に変数の許容範囲を求めるために，変数 $x1$ の下限値と上限値をそれぞれ $V_{x1_{min}}$，$V_{x1_{max}}$ と表す．なお，I/O イベント以外のイベントは $T_{a_{min}} = T_{a_{max}}$ と考える．

以下では，これらを用いて線形計画問題の制約条件

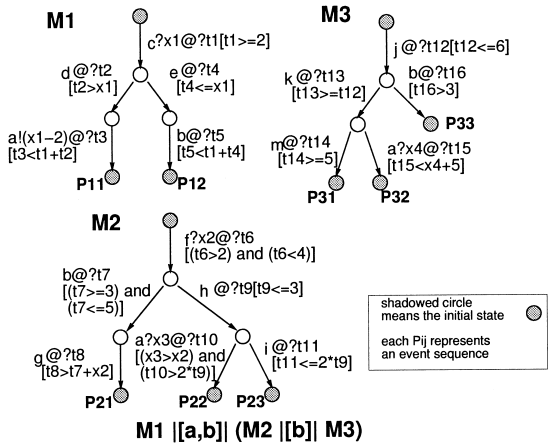

図 4 並行周期 EFSM 群の例
Fig. 4 Example of concurrent periodic EFSMs.

を構成する方法を**図 4** の並行周期 EFSM 群を例に説明する．

(1) イベントの実行順序に関する時間制約
あるイベントに続くイベントの実行時刻は，そのイベントの実行時刻より後でなければならない．また，3 章の方針に従い，連続するイベントの実行時刻の許容範囲が重ならないよう制約式を構成する．したがって，たとえば，図 4 の M1 のイベント系列 $P12$ からは，以下の制約条件が得られる．

$$T_{c_{min}} \le T_{c_{max}} < T_{e_{min}} \le T_{e_{max}} < T_{b_{min}} \le T_{b_{max}} \quad (1)$$

(2) マルチランデブを行うイベント間の時間制約
4.1 節で求めた同期イベント組の集合の各要素は同じ許容時間内に実行されなければならない．たとえば，図 4 のイベント系列の組 $\{P12, P21, P33\}$ ではイベント b の同期実行が指定されているため，以下の制約条件が得られる（$T_{b_{M1}}$ は $M1$ のイベント b の実行時刻を表し，その下限値と上限値をそれぞれ $T_{b_{M1_{min}}}$，$T_{b_{M1_{max}}}$ と表す）．

$$T_{b_{M1_{min}}} = T_{b_{M2_{min}}} = T_{b_{M3_{min}}} \quad (2)$$

$$T_{b_{M1_{max}}} = T_{b_{M2_{max}}} = T_{b_{M3_{max}}} \quad (3)$$

(3) 各イベントに指定された時間制約
各イベントに指定された時間制約は，それをそのまま制約条件とすればよい．ただし，式中に含まれる時間変数は，直前のイベントが実行されてからの経過時間を表しているので，各イベントの実行時刻を表すパラメータで置き換える．たとえば，図 4 のイベント系列 $P12$

$c?x1@?t1[...]; e@?t4[...]; b@?t5[t5 < t1 + t4]$

のイベント b の時間制約から得られる制約条件

$[t5 < t1 + t4]$ は，$t1$，$t4$，$t5$ をそれぞれ，T_c，$T_e - T_c$，$T_b - T_e$ に置き換えることにより，以下のようになる．

$$(T_b - T_e) < T_c + (T_e - T_c)$$

この式を簡単化すると，

$$T_b < 2T_e \tag{4}$$

なる式が得られる．同様にイベント e についても考えると以下のように置き換えられる．

$$T_e - T_c \leq V_{x1}$$
$$\Rightarrow T_e \leq V_{x1} + T_c \tag{5}$$

T_c，V_{x1} がそれぞれ $[T_{c_{min}}, T_{c_{max}}]$，ならびに $[V_{x1_{min}}, V_{x1_{max}}]$ の範囲の値をとることから，式 (5) の右辺の式の最小値は $V_{x1_{min}} + T_{c_{min}}$ である．式 (5) では，T_e の上限値はこの値かそれ以下の値と考えられる．同様に，式 (1) では，$T_{e_{min}}$ は $T_{c_{max}}$ よりも大きな値をとる．以上より，イベント e について以下の制約式が得られる．

$$T_{c_{max}} < T_{e_{min}} \leq T_{e_{max}} \leq V_{x1_{min}} + T_{c_{min}} \tag{6}$$

同様に，式 (1) と式 (4) により，イベント b について以下のような制約式が得られる．

$$T_{e_{max}} < T_{b_{min}} \leq T_{b_{max}} < 2T_{e_{min}} \tag{7}$$

以上のように置き換えを行った制約式の下で，

$$max \left\{ \sum_i w_{t_i}(T_{i_{max}} - T_{i_{min}}) + \sum_j w_{v_j}(V_{j_{max}} - V_{j_{min}}) \right\} \tag{8}$$

（ここで，T_i はイベント系列の i 番目のイベントの実行時刻，V_j は j 番目の変数を表す．また，w_{t_i}，w_{v_j} は適当な重み付け定数である）

なる目的関数を設定し，これが最大となるように線形計画問題を解くことにより，各イベントの実行時刻の下限値，および上限値が求められる．なお，各イベントの時間範囲に重み付けを行うことにより，特定の I/O イベントの時間範囲を他より大きくとることも可能である．

なお，3.1 節で述べたように，どの分岐を実行するかに依存して，各イベントの実行時刻の許容範囲は異なるため，提案手法では，各分岐ごとにイベントの実行時刻の許容範囲を求める．また，各イベント系列に同期イベントが含まれる場合には，どの EFSM のイベントと同期するかに依存して，その同期イベントの実行時刻の許容範囲が異なるかもしれない．したがって，同期イベントを含むイベント系列は，同期の組合せごとに別々に上記の制約式を構成し，それぞれについてイベントの実行時刻の許容範囲を求める．

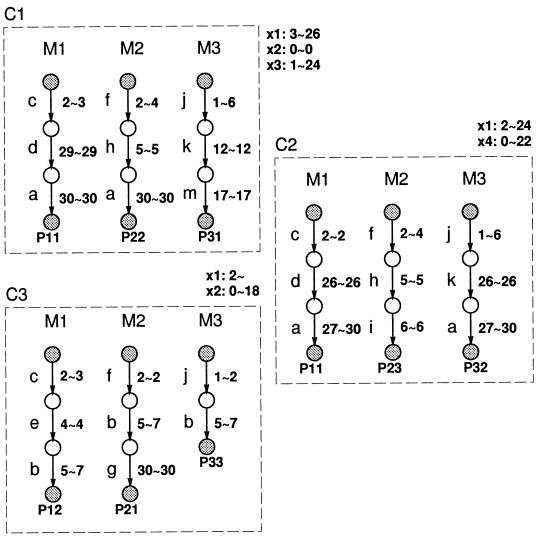

図 5 イベントの実行時刻の許容範囲の導出例
Fig. 5 Executable time range of each event derived from example in Fig. 4.

4.3 各イベントの実行時刻の許容範囲の導出例

ここでは，図 4 に示す並行周期 EFSM 群に対して，上述の手法を適用した結果，導出された各イベントの実行時刻の許容範囲を示す．ここでは，以下のような式を目的関数とし，これを最大化するような解を求めた．すべての EFSM の周期は 30 単位時間とした．

$$(T_{a_{max}} - T_{a_{min}}) + (T_{b_{max}} - T_{b_{min}})$$

実行可能な 3 通りのイベント系列の組それぞれについて導出された各イベントの実行時刻の許容範囲を**図 5** に示す．

5. 回路の構成法

5.1 全 体 構 成

提案手法では，最終的な回路を以下の部分から構成する．

(1) 同クロックで動作する各 EFSM を実現する順序回路
(2) マルチランデブ制御部
(3) スケジュール不可能となった系列上のイベントが選択実行されないように，各 EFSM に対して指示を送るスケジューラ

このうち，(1)，(2) については，我々が文献 6)，12) で提案している方法を用いて実現する．本章では，主に，(3) についての詳細を述べる．また，生成される回路の構成を**図 6** に示す．

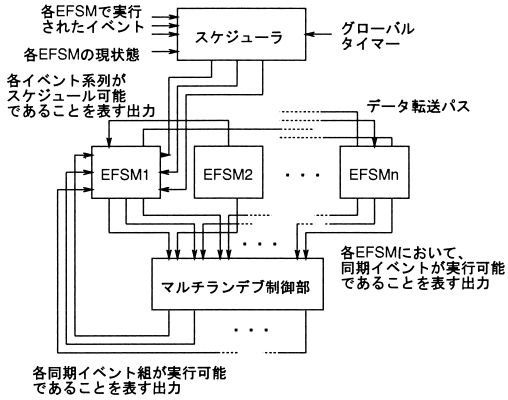

図 6 ハードウェアの全体構成
Fig. 6 Architecture of derived hardware circuit.

5.2 マルチランデブ制御部

マルチランデブ制御部は，

(1) 各クロック周期ごとに各同期イベント組が実行可能であるかどうかの判定を行う同期判定部
(2) 排他的にしか実行できない複数の同期イベント組が同時に実行可能となった場合に実行させる同期イベント組の選択を行う競合回避部

から構成した[6),12)]．

(1) の同期判定部では，(トレースの際に抽出した) 同期イベント組ごとに同期判定回路を設ける．同期判定回路は，同期イベント組に属する同期イベントを実行するすべての EFSM から，その同期イベントが現状態から実行可能であるかどうかを示す信号を受け取る．すべての EFSM からの出力が真ならば，その同期イベント組は実行可能となる．その場合は，同期判定回路から各 EFSM に対して同期が可能であることを示す出力を返す．この判定回路は，n 入力 (n は同期を行う EFSM の個数) の AND 回路で実現できる．

(2) の競合回避部では，互いに競合する (排他的にしか実行できない) 同期イベント組の集合の情報から，それらの同期イベント組が同時に実行可能となった場合に，いずれか 1 つを選択するための回路である．選択の方法はいくつか考えられるが，本手法では，あらかじめ設定しておいた優先度に従って選択を行うこととした．この選択を行う回路は，プライオリティエンコーダを用いて実現できる．

また，同期イベントの実行可能性判定に，他の EFSM のイベントの出力値を必要とする場合に備えるため，各同期イベント組の属する EFSM 間にデータ転送用パスを設置した．データ転送用パスは，複数の同期イベント組で同時に同期判定される可能性を考慮して，基本的に同期イベント組ごとに設置するが，ある条件を満たす同期イベント組をグループ化し，パスを共用させることで，効率化を図った．詳細については文献 6)，12) で述べている．

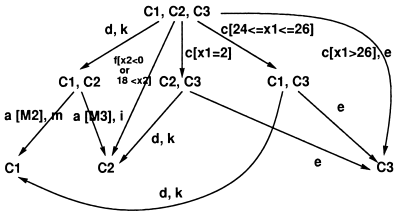

図 7 図 4 の並行周期 EFSM 群のスケジュール木
Fig. 7 Scheduling tree for concurrent periodic EFSMs in Fig. 4.

5.3 スケジューラ

各イベントの遅延時間の許容範囲は，4 章で述べたように，各イベント系列ごと，および，同期する組合せごとに求められているため，イベントが実行されていくに従い，各イベントの遅延時間の許容範囲は狭まるが，範囲が変化するのは，イベント系列が選択実行された場合と同期の組合せが複数あるような同期イベントが実行された場合のみである．

したがって，各 EFSM でどのイベント系列が選択実行されたか，また，どの組合せで同期イベントが実行されたかの情報から，各時点での各イベントの遅延時間の許容範囲と，各イベント系列のスケジュール可能性が分かる．

提案手法では，4.1 節で求めた同期イベント組と，各 EFSM 内での分岐の情報から，**図 7** のようなスケジュール可能なイベント系列の組の移り変わりを表す木 (**スケジュール木**と呼ぶ) を構成する．図 4 の並行 EFSM 群では図 5 に示したように，3 つのイベント系列の組 $C1$，$C2$，$C3$ が実行可能である．図 7 の木の各節にはその時点で実行可能なイベント系列の組を示し，各枝には EFSM の分岐を表すイベント，あるいは EFSM 間の同期イベントが書かれている．各枝に書かれたイベントが実行されると，その後に実行可能なイベント系列の組が変化する．たとえば，M1 のイベント c の実行後，イベント d が実行された場合，次に現れる同期イベント a が実行可能となるためには，$C1$，$C2$ のいずれかの組のイベント系列が実行されなければならない ($C3$ の M1 のイベント系列には a が含まれないため選択できない)．その後，M1 のイベント a が M2 のイベント a と同期実行される場合は $C1$ が選択され，M3 のイベント a と同期実行される場合は $C2$ が選択される．このように各節に書かれたイベント系列の組の時間制約を満たしながら各々のイ

ベント系列を実行すると，指定した周期時間後に再び根に戻る仕組みになっている．

イベント系列の組合せごとの各イベントの実行時刻の許容範囲は，4 章の方法ですでに定数として求められている．そこで，提案手法では，(1) 木の各節点を状態機械における状態と見なし，各状態における各イベントの実行時刻の許容範囲を出力する回路（表形式のデータを記憶する ROM として実現可能）と，(2) 現在時刻が出力された範囲に収まっているかどうかを出力する比較回路からなるスケジューリング回路を構成した．

スケジューリング回路の出力を各 EFSM でのイベントの実行条件として用いることで，スケジュール可能性を考慮したイベントの実行を行うことができる．

5.4 VHDL 回路記述の生成系

本研究では，提案手法の有用性を確認するために，仕様記述言語 E-LOTOS [4] の構文を用いて記述された動作仕様からハードウェア記述言語 VHDL [3] によるレジスタ転送レベルの回路記述を自動生成するツールを実際に作成した．ただし，対象となる E-LOTOS のクラスは，本論文で提案した並行周期 EFSM 群に変換可能なものに限られる．また，線形計画問題を解くためのツールとして，lp_solve [8] と呼ばれるフリーウェアを利用した．これらのツールによって生成された VHDL の回路記述は，市販の CAD システムを用いて論理合成を行い，FPGA 上などへ実装することが可能である．

6. アプリケーションの記述

本章では，評価実験のための動作仕様例として，最大 3 人が参加可能なテレビ会議システムのクライアント（各端末）において他の参加者の動画の再生を支援するハードウェア回路動作の仕様を提案モデルを用いて記述する．

各クライアントには，他の 2 人分の参加者の動画ストリームが個別の回線を通じて周期的にフレーム単位で受信され，用意されたメモリバッファに格納される．ただし，回線上のジッタにより，フレームの届く時間間隔は変化するものとする．クライアントのデスクトップ上には，各参加者の動画を表示するためのウィンドウが用意されており，受信した動画の各フレームは外部ユニットであるデコーダによりデコードされた後，対応するウィンドウに表示される．また，各参加者の出席，および退席は自由に行われる（出席していない場合には，その参加者用のウィンドウには何も表示されない）．デコーダは 1 フレームあたり 35 単位時間でデコードでき，すべての EFSM の周期はそれぞれ 100 単位時間とする．

6.1 動画再生システムの記述例

システムは 4 種類の EFSM から構成する．

- 表示部：各参加者の動画フレームを適切な品質で表示する．表示品質は参加者の人数に応じて決められる．
- 受付部：各参加者の出席，および退席の要求を受け付ける．
- デコード部：要求されたフレームのデコードを行う．デコード部は複数の表示部により共用される．また，実際のデコード処理は，外部の演算ユニットを利用して行うものとする．
- 管理部：現在の参加者数を保持する．

各表示部は，提供可能なすべての品質における動作内容を系列として記述し，現在の品質値に応じて，その中から適切な 1 つを選択するようにする．この仕様例では，「1 周期に 2 フレーム表示」，「1 周期に 1 フレーム表示」という 2 通りの品質の動作内容を定義している．どちらが選ばれるかは，現在の参加者数によって決まる．なお，出席していない場合には，動画の再生は行わない．各表示部は，入力イベント vin により，各メモリバッファからフレームデータを受け取ると，同期イベント req を通じて，デコード部に対してフレームデータを渡す．1 つのデコード部に対して，複数の表示部から同時にデコード要求が出された場合は，その中のいずれか 1 つが排他的に選択される．よって，デコードに 35 単位時間かかるため時間制約を満たせず，ユーザの数によって 1 つ，またはすべての表示部が 1 周期に 2 フレーム（高画質）表示することができない．このような場合，スケジュール可能な系列（低画質）が動的に選ばれる．表示部は同期イベント ans を通じてデコードを依頼したデコード部からデコード後のデータを読み出し，出力イベント $vout$ により画面に出力する．

6.2 適用結果

提案手法の有用性を確認するため，前章で述べた回路記述自動生成ツールに上述の仕様を入力として与えた．仕様を入力として与えてから回路記述が生成されるまでの時間は，線形計画問題を解く時間も含めて約 2 秒であった．

生成された VHDL による回路記述からは，市販の論理合成ツールを用いてテクノロジマッピングを行うことにより，その回路の面積と最小クロック周期を求めることができる．論理合成ツールとして SYNOPSYS 社の Design Compiler を用いて，CMOS，0.5 micron

のテクノロジで合成した結果，論理合成が終了するまでに約 5 時間要した．得られた回路の面積はおよそ 3500 ゲート，最小クロック周期は 150 ns であり，面積，速度ともに実用上問題ないと考えてよいと思われる．

7. あとがき

本論文では，複数モジュールの並行動作と協調，および時間制約を扱えるモデルとして，並行周期 EFSM 群を提案し，そのモデルを用いて記述された動作仕様からハードウェアを自動導出する手法を提案した．

提案手法によって生成される回路は，仕様に書かれたすべての分岐を実現し，かつ，先行イベントの遅延時間に従って，今後スケジュール可能なイベント系列のみを動的に選択実行する回路である．また，各 I/O イベントごとに実行時刻の許容範囲を求め，その範囲内に実行されたならば後続のいずれかの分岐が時間制約を満たしながら実行可能であることを保証している．

提案した並行周期 EFSM 群モデルでは，すべての EFSM が初期状態に至る各イベント系列を同じ周期 T で実行終了するよう制限することにより，同期するイベント（遷移）の EFSM 間での組合せが限定されるため，他の EFSM との同期イベントを含むイベント系列をスケジュール可能とするような各イベントの実行時刻を実用的な時間で求めることが可能になっている．

また，マルチメディアシステムなどでは，動画や音声の再生などシステムの挙動の大半が周期処理からなること，異なる周期を持つ処理は，それらの処理の周期の最小公倍数に機械的に展開する，などの方法により本手法が適用可能なこと，などから，提案手法の適用範囲は広いと考えられる．

動画再生支援チップの生成実験により，本手法がマルチメディアを含む回路の動作仕様記述に十分なクラスを対象としていること，また，生成された回路が時間制約を保証しつつ動作し，面積および性能の点においても実用的な水準であることなどを確認した．

今後の課題は，限られた数のリソースを EFSM 間でできるだけ共有できるように線形計画問題においてリソースのコスト関数を統合し，リソースのコストを全体で小さくすることなどがあげられる．

参考文献

1) Csopaki, G. and Turner, K.J.: Modelling Digital Logic in SDL, *Proc. Joint Int. Conf. on 10th Formal Description Techniques and 17th Protocol Specification, Testing and Verification* (*FORTE/PSTV'97*), pp.367–382 (1997).
2) Dave, B.P., Lakshminarayana, G. and Jha, N.K.: COSYN: Hardware-Software Co-Synthesis of Embedded Systems, *Proc. 34th Design Automation Conf.* (*DAC'97*), pp.703–708 (1997).
3) IEEE: IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1993 (1994).
4) ISO: Final Committee Draft 15437 on Enhancements to LOTOS, ISO/IEC JTC1/SC21/WG7 (1998).
5) ISO: Information Processing System, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807 (1989).
6) Katagiri, H., Yasumoto, K., Kitajima, A., Higashino, T. and Taniguchi, K.: Hardware Implementation of Communication Protocols Modeled by Concurrent EFSMs with Multi-Way Synchronization, *Proc. 37th Design Automation Conf.* (*DAC'2000*), pp.762–767 (2000).
7) Kloos, C.D., Moro, T.M., Filho, G.R. and Lopez, A.M.: VHDL Generation from a Timed Extension of the Formal Description Technique LOTOS within the FORMAT Project, *Microprocessing and Microprogramming*, 38, pp.589–596 (1993).
8) Michel, B. and Jeroen, D.: LP_SOLVE. ftp://ftp.es.ele.tue.nl/pub/lp_solve
9) Quemada, J., Larrabeiti, D. and Pavón, S.: Compressing the State Space Representation of LOTOS Specifications, *Proc. 6th IFIP Int. Conf. on Formal Description Techniques* (*FORTE'93*), pp.19–34 (1993).
10) Sisto, R.: A Method to Build Symbolic Representations of LOTOS Specifications, *Proc. 15th IFIP Int. Symp. on Protocol Specification, Testing and Verification* (*PSTV-XV*), pp.331–346 (1995).
11) Smith, J. and Micheli, De.G.: Automated Composition for Hardware Components, *Proc. 35th Design Automation Conf.* (*DAC'98*) (1998).
12) Yasumoto, K., Kitajima, A., Higashino, T. and Taniguchi, K.: Hardware Synthesis From Protocol Specifications in LOTOS, *Proc. Joint Int. Conf. on 11th Formal Description Techniques and 18th Protocol Specification, Testing, and Verification* (*FORTE/PSTV'98*), pp.405–420 (1998).
13) Wytrebowicz, J.: Hardware Specification Gen-

erated from Estelle, *Proc. 15th IFIP Int. Symp. on Protocol Specification, Testing and Verification* (*PSTV-XV*), pp.435–450 (1996).

(平成 12 年 7 月 28 日受付)
(平成 13 年 1 月 11 日採録)

片桐　久晶

平成 12 年大阪大学大学院基礎工学研究科博士前期課程修了．同年（株）東芝入社．現在，ハードディスク用制御 LSI の設計に従事．

桐村　昌行（学生会員）

平成 12 年大阪大学基礎工学部情報科学科卒業．現在，同大学大学院博士前期課程在学中．実時間システムや分散システムの仕様記述・実装法に関する研究に従事．

安本　慶一（正会員）

平成 3 年大阪大学基礎工学部情報工学科卒業．平成 7 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．現在同大学助教授．博士（工学）．平成 9 年モントリオール大学客員研究員．通信プロトコルや分散システムの形式仕様記述・実装法に関する研究に従事．

中田　明夫（正会員）

平成 4 年大阪大学基礎工学部情報工学科卒業．平成 9 年同大学大学院基礎工学研究科物理系専攻博士後期課程修了．博士（工学）．同年広島市立大学情報科学部助手．現在，大阪大学大学院基礎工学研究科助手．実時間システムや分散システムの仕様記述と検証法，プロセス代数，時相論理等の研究に従事．

東野　輝夫（正会員）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．同年同大学助手．平成 2，6 年モントリオール大学客員研究員．現在，大阪大学大学院基礎工学研究科教授，工学博士．分散システム，通信プロトコル等の研究に従事．電子情報通信学会，ACM 各会員．IEEE Senior Member．

谷口　健一（正会員）

昭和 40 年大阪大学工学部電子工学科卒業．昭和 45 年同大学大学院博士課程修了．同年同大学助手．現在，同大学大学院基礎工学研究科教授．工学博士．この間，計算理論，ソフトウェアやハードウェアの仕様記述・実現・検証の代数的手法および支援システム，関数型言語の処理系，分散システムや通信プロトコルの設計・検証法等に関する研究に従事．