マルチメディア通信と分散処理　112－3
（２００３．３．２０）

並行周期EFSM群でモデル化されたQoSルータの高信頼性設計の一手法

木谷 友哉* 高本 佳史* 安本 慶一† 中田 明夫* 東野 輝夫*

本稿では，並行周期 EFSM 群を用いてモデル化した QoS ルータなどの実時間システムの高信頼性設計の一手法を提案する．提案方法は，パラメトリックモデル検査とリアルタイムスケジューリング技術から構成される．我々は，それらを用いることにより，(1) 並行周期 EFSM 群が指定された時間内で繰り返し状態遷移できるための条件式，(2) 全ての状態遷移が実行できるための各動作の実行時間の範囲，を得ることができる．得られた結果から QoS ルータのハードウェア回路が自動生成される．

A Proposal of a High-Reliable Design of the QoS Router Modeled as Concurrent Periodic EFSMs

Tomoya KITANI* , Yoshifumi TAKAMOTO* , Keiichi YASUMOTO† ,
Akio NAKATA* and Teruo HIGASHINO*

In this paper, we propose a method to synthesize high-reliable hardware circuits of real-time systems using concurrent periodic EFSMs. The method uses parametric model checking and real-time scheduling techniques, where we can derive (1) the weakest condition that given concurrent EFSMs can execute their transitions in the specified time period repeatedly, and (2) allowable time ranges of all transitions which satisfy the condition. From the obtained results, we derive the corresponding RT-level's descriptions automatically. Based on the method, the hardware circuits of QoS routers have been derived. The details of the derivation are reported.

1 まえがき

近年，高速ネットワークやマルチメディア通信システムの発展により，実時間通信システムのハードウェア合成が重要になってきている．通常，そのような実時間通信システムは，大部分が繰り返し周期的な処理を実行する時間制約付きのモジュールとして実装される．その上で適応性のある実時間通信システムを開発するためには，処理を指定された周期で繰り返し実行でき，かつ，デッドロックを起こさないような適切な時間制約のパラメータを見つけ出す必要がある．また同時に，良い性能を得るためにできるだけ周期は短くなることが望ましい．しかしながら，適切なパラメータ値を得るためには，通常，試行錯誤を繰り返してシミュレーションや分析を行い，パラメータ値を修正しなくてはならない．

本稿で，我々は並行周期 EFSM 群でモデル化された実時間システムの高信頼性なハードウェア合成を行う方法を提案する．LOTOS やその拡張の E-LOTUS [7, 8] では，マルチランデブと呼ばれる高レベルな通信プリミティブを持つ．マルチランデブとは，複数の並行プロセスがそれぞれある条件を満たしたとき，それらのプロセス間で同期してデータ交換を行う仕組みである．

提案する手法では，文献 [10] で提案されたパラメトリックモデル検査法を使う．まず，並行周期 EFSM 群が指定された時間制約の下でデッドロックすることなく処理を周期的に繰り返し実行できるためのパラメータ条件を求める．次に，得られたパラメータ条件からその条件を満たす適切なパラメータ値を決定する．その際，整数線形計画問題の手法を用い，全ての並行周期 EFSM 群が指定した周期で処理を実行できるための各動作の実行時間の範囲を得る．得られた時間範囲を用いて，EFSM 群の実時間スケジューラを組み立てる．

実時間並行プロセスのスケジューリングには多くの研究がなされている．例として，[2] では，通信依存グラフを生成することによって並行プロセス間の通信/相互排除を扱うことができる．しかし，並行プロセスの同期点を見つけることに長い時間を要することから，その実装はコストがかかる可能性がある．同様に，[4] では，M 対 N の同期通信が考慮されている．しかしながら，LOTOS と異なり，動的な相互排除は考慮されていない．また，これらの手法は，時間制約を考慮していない．本稿では，並行周期モジュール間の高速通信/相互排除操作を持つモデルと，そのスケジューリングアルゴリズムを提案する．

我々の手法の有用性を示すため，QoS ルータで用いられる優先度付きキューイングメカニズムである WFQ (Weighted Fair Queuing) の設計と実装を行った．提案手法では，設計者は並行周期 EFSM 群で WFQ メカニズムの仕様を記述するだけでよく，開発ツールが適切なパラメータ値を生成し，自動的に対応する RT レベルの VHDL 記述を生成する．生成された VHDL 記述からは，市販されている VHDL 合成ツールを使って，対応するハードウェア回路が合成できる．合成結果より，提案手法がギガビットネットワークのための WFQ の実装に十分適用可能であることが分かった．

2 並行周期 EFSM

EFSM は，6 字組 $< S, I, V, clock, \delta, init >$ で定義される．ここで，S は状態の有限集合，I はイベント (例えば遷移) の有限集合，V は変数の有限集合，$clock$ は EFSM が新しく状態遷移をしたときからの経過時間のカウンタ，δ は遷移規則の有限集合，$init$ は初期状態と全変数の初期値を表現する．V 中の変数は，時間変数や入力変数，パラメータとは区別される．

イベントの実行はすぐに完了すると仮定する．システムと外部の環境とのデータ交換イベントは，I/O イベントと呼ばれる．また，異なる EFSM の同名のイベントと同期して実行が行われ，同期イベントと呼ばれる．入力，出力イベントはそれぞれ $a?x$, $b!E$ で表される．入力変数 x は，再び $a?x$ が実行されるまで不変である．さらに，$a@?t$ はイベント a の実行時間として，状態遷移

* 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
† 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

からの経過時間 $clock$ の値を時間変数 t に代入することを表す．この時間変数 t は，再び $a@?t$ が実行されるまで不変である．$clock$ の値は整数として扱われる．変数としてパラメータも使用できる．

それぞれの遷移規則は，$s_{cur} \xrightarrow{a[guard]} s_{next}$ と定義される．ここで，$guard$ は遷移条件と呼ばれる．それぞれのEFSMにおいて，遷移条件内で使われている変数の型は全て整数であると仮定する．もし遷移条件 $guard$ が状態 s_{cur} において真であり，イベント a が実行されたなら，そのEFSMは状態 s_{next} に遷移する．このとき，遷移条件内で時間変数を使うことによって，それぞれのイベントの実行時間に制約を与えることができる．例えば，$a?x@?t[(2 \leq t)and(t \leq P)and(0 \leq x)and(x \leq 10)]$ では，入力イベント $a?x$ の実行は，その状態において少なくとも2単位時間が経過した後で，かつ，P 単位時間経過するよりも前に行われなければならず，イベント a によって入力される整数の満たすべき条件は $(0 \leq x)and(x \leq 10)$ でなければならないことを示している．このイベントが実行されると，入力された値は変数 x に代入される．ここで P はパラメータとして扱われている．パラメータの具体的な値は実装段階において固定される．遷移条件として，入力変数，パラメータ，時間変数の整数線形不等式の論理積のみが指定できる．もし複数のイベントが現状態で実行可能であれば，そのうち1つのイベントのみが非決定的に選択され実行される．

あるEFSMにおいて，初期状態から始まるパス(イベントの順序)が全て初期状態へ戻る，もしくは，EFSMが必ず一定時間間隔 T で初期状態に戻ることができるように遷移条件が指定されているならば，そのEFSMを周期EFSMと呼ぶ．与えられたEFSMを周期的にするために，我々は初期状態からの全てのパスが，特別な遷移 ψ を持つと仮定する．ψ はパスの最後の遷移であり，その遷移条件はある一定間隔 T で ψ が実行されると指定する．

EFSM間のマルチランデブ関係は，以下に示すLOTOSの並行オペレータで指定される．

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

ここで，$EFSM$ はEFSMの名前，$|[gate\_list]|$ は並列オペレータであり，$gate\_list$ は両辺のEFSM間で同期するイベント名のリストである．オペレータ $|||$ は非同期の並列オペレータである．例として，図1の $M1|[a]|(M2|[b]|M3)$ では，イベント a は $M1$ と $M2$ の間で同期して実行され，イベント b は $M2$ と $M3$ の間で同期して実行される．これらの同期イベントが同時に実行可能となった時には，これらのうち1つが排他的に選択される．$M1$ と $M2$ の同期イベント a が実行されるとき，$M1$ の出力イベント "$a!x1@?t12$" の出力値 $x1$ は，$M2$ の入力イベント "$a!x2$" の入力変数 $x2$ に代入される．

ここで図1の仕様は，$M1$ がパケットの入力，$M3$ がパケットの出力を表し，$M2$ で2パターンのパケットの受け渡しを実現した，キューイングメカニズムのごく簡単なものを表現したものである．並行周期EFSMを用いることで，マルチランデブを持つ実時間システムの仕様を簡単に記述することができる．

3 パラメトリックモデル検査

パラメトリックモデル検査法は，状態遷移モデルが与えられた時間的性質を満たすようにパラメータ条件を導出する．もし導出されたパラメータ条件が満たされるならば，その状態遷移モデルにおいてその時間的性質は保持される．

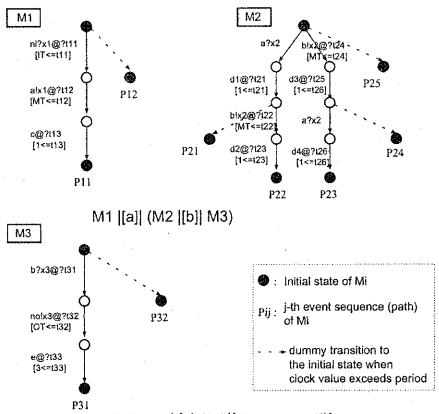

図 1: 並行周期 EFSM 群

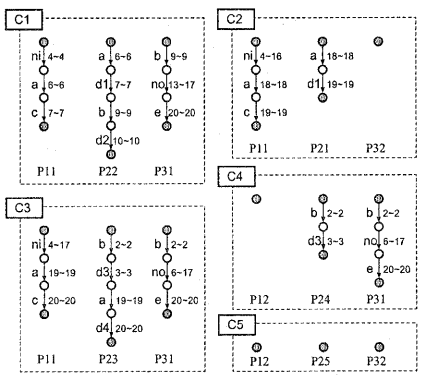

図 2: 図1のイベント実行順序の可能な組合せ例

文献[10]において，我々は周期EFSMのためのパラメトリックモデル検査法を提案した．その手法では，時間的性質は $RPCTL$ (Real-time and Parametric extension of Computation Tree Logic) で記述される．モデルを周期的であることに制限することにより，その手法では，高々3周期分の与えられた周期EFSMの振舞いを解析することで効率的にパラメータ条件を導出することができる(詳細は[10]参照)．

RPCTLの文法は表1のBNFで定義される．ここで，a は動作名，p はパラメータ変数を含む線形式，$\sim \in \{<, \leq, >, \geq, =\}$ は比較演算子である．$\sim p$ が ≥ 0 のときは省略することができる．$\langle a \rangle_{\leq p} f$ は状態 s に到達後 p 単位時間以内に a による遷移のいずれかを実行したあと，遷移先状態で f が成立するとき，かつそのときのみ s で成立する．$[a]_{\leq p} f$ は状態 s に到達後 p 単位時間以内でどの a による遷移を実行しても，遷移先状態で f が成立するとき，かつそのときのみ s で成立する．これは $\neg\langle a \rangle_{\leq p} \neg f$ と等価である．$f_1 EU_{\leq p} f_2$ は状態 s に到達後 p 単位時間以内に f_2 が成立する状態へ到達する遷移系列が存在し，f_2 が成立するまではいつでも f_1 が成立しているとき，かつそのときのみ s で成立する．$f_1 AU_{\leq p} f_2$ は状態 s からどのような遷移系列を実行しても p 単位時

表 1: RPCTL の文法

f	::=	$true$	(恒真)	$false$	(恒偽)
	|	$f \wedge f$	(論理積)	$f \vee f$	(論理和)
	|	$\neg f$	(否定)	$f \Leftarrow f$	(含意)
	|	$\langle a \rangle_{\sim p} f$	(存在 next 構文)	$[a]_{\sim p} f$	(全称 next 構文)
	|	$fEU_{\sim p} f$	(存在 until 構文)	$fAU_{\sim p} f$	(全称 until 構文)
	|	$EF_{\sim p} f$	(存在 eventually 構文)	$AF_{\sim p} f$	(全称 eventually 構文)
	|	$EG_{\sim p} f$	(存在 always 構文)	$AG_{\sim p} f$	(全称 always 構文)

間以内に f_2 が成立する状態へ到達し，f_2 が成立するまではいつでも f_1 が成立しているとき，かつそのときのみ s で成立する．

図 1 の並行周期 EFSM 群の例において，“システムはいつも周期 $Period$ 毎に特別なダミートランジション ψ を実行して初期状態に戻らなければならない” という性質を考える．この性質は RPCTL では $AG_{=T}\,(EF\,(\langle\psi\rangle\, true))$ と記述される．

このような制約から，“$Period \geq IT+2MT+OT+4$” というパラメータ条件を導出することができる．これは，デッドロックなく周期的に実行するためには，周期 $Period$ は $IT+2MT+OT+4$ 以上でなければならないことを示している．もし，IT，MT，OT の値がそれぞれ 4，2，4 であるとすると，ハードウェアとして実装するための制約から周期 $Period$ の値は少なくとも 16 でなければならない．

4 スケジューリングアルゴリズム

並行周期 EFSM 群において，それぞれの周期 EFSM のイベントの順序は図 1 のように木構造で表現できる．もし，EFSM が n 個あり，それぞれの EFSM が最大 k 個のパス* を持つならば，最悪で $O(k^n)$ の実行系列の組合せがある．例として，図 1 の仕様では，5 つの可能な組合せ $C1, ..., C5$ が図 2 に示されている．しかし，それぞれのイベントが実際に実行される時間に依存していくつかのパスでは，ある一定周期で全ての EFSM が初期状態へ戻ることができないかもしれない．例として，図 1 の並行周期 EFSM において，もし $M1$ のイベント ni が時刻 6 で実行された場合，$C1(P11, P22, P31)$，$C4(P12, P24, P31)$，$C5(P12, P25, P32)$ の組合せがそれ以後実行できなくなる．提案手法では，まず時間制約を満たして実行可能なパスの全てのイベントそれぞれについての時間範囲を得る (このようなパスをスケジューリング可能なパスと呼ぶ)．基本的な考え方は，それぞれのパスの組合せでそれぞれのイベントの実行可能な時間範囲を計算し，得られた時間範囲とイベントの実際の実行時間を基にしてそれぞれの EFSM のスケジューリング可能なパスを示すスケジューラを構成することである．

4.1 実行時間範囲の導出

それぞれのパスの組合せに対して，各イベントの実行時間範囲を得るため，(a) 同期イベントの対，(b) 各 EFSM のイベントの時間的順序，(c) 各イベントの時間制約，から下記の ILP(整数線形計画) 問題を構築する．ここで，イベント a の実行時間を変数 T_a で，その下界と上界をそれぞれ $T_{a_{min}}$，$T_{a_{max}}$ で表す．

周期的なイベント系列の各組合せに対して，ILP 問題として整数線形不等式の論理和を構築する．もし，1 つのイベント系列に同じラベルのイベントが複数あるならば，それらのラベルをそれぞれ違う名前に変更する．

例として，図 1 の並行周期 EFSM 群を使用する．ここでは，図 2 の組合せ $C1$ の場合のみについて説明する．

(1) イベントの実行時間の定義

各 EFSM の初期状態から実行されたイベントの実行時間は 0 以上でなければならない．それゆえ，以下の制約が得られる．

$$0 \leq T_{ni_{min}}, 0 \leq T_{b_{M2_{min}}}, 0 \leq T_{b_{M3_{min}}}$$

(2) 各 *EFSM* のイベントの時間順序の制約

各イベントの実行時間範囲は続いて起こるイベントの実行時間よりも小さくなければならない．$M1$ のイベント系列 $P11$ を例にとると，以下のような制約が導かれる．

$$0 \leq T_{ni_{min}} \leq T_{ni_{max}} < T_{a_{M1_{min}}} \leq T_{a_{M1_{max}}} < T_{c_{min}} \leq T_{c_{max}} \leq Period$$

(3) 同期イベントに対する制約

同期イベントに関する全てのイベントは同時に実行されなければならない．よって，図 1 のイベント系列 {P11, P21} 間で同期して実行されるイベント a のための制約は以下のように導出される．

$$T_{a_{M1_{min}}} = T_{a_{M2_{min}}}, T_{a_{M1_{max}}} = T_{a_{M2_{max}}}$$
$$T_{b_{M2_{min}}} = T_{b_{M3_{min}}}, T_{b_{M2_{max}}} = T_{b_{M3_{max}}}$$

(4) 各イベントに付随する時間制約

各イベントに付随する時間制約もまた，ILP 問題により得られるものである．しかし，元々の時間変数は前の状態からの経過時間を表現するものであるから，それを現在の周期の最初からの経過時間を表す変数に変換しなくてはならない．例として，以下のイベント系列のイベント a の時間制約について考える．

$$ni?x1@?t11[...]; a!x1@?t12[...]; c@?t13[1 \leq t13]$$

T_{ni}，T_a，T_c はそれぞれ上記のイベントの実行時間であるとする．この場合，c の制約 $1 \leq t13$ は $1 \leq T_c - T_a \Leftrightarrow 1 + T_a \leq T_c$ となる．

イベント c の実行時間 T_c の範囲 $[T_{c_{min}}, T_{c_{max}}]$ は実際の T_a の値によって変化する．ここで，$T_{c_{min}}$ は T_a の最大値より大きくなければならないと仮定する．そこで，以下の制約を追加する．

$$1 + T_{a_{max}} \leq T_{c_{min}}$$

以下の目的関数のために上記の制約式の論理和を解くことで，各イベントの実行時間の範囲を得ることができる (ここで，T_i はイベント系列の i 板目の実行時間を，wt_i, w_p はそれぞれ重み係数を表す)．

$$max \sum wt_i(T_{i_{max}} - T_{i_{min}}) - w_p Period \quad (1)$$

ここで，図 1 の仕様の例に対して提案手法を応用した．そこで，仕様のパラメータに $IT = OT = 4$，$MT = 2$，

* イベントの実行系列をパスと呼ぶ

目的関数の重み係数に $w_{t_i} = 0$ を用いると，全 EFSM の最小の周期 $Period$ は 16 となる．次に，ILP 問題の制約に周期 $Period = 20$ を追加し，目的関数の重み係数を $w_{t_i} = 1$ とすると，5 つの実行系列 $C1, C2, C3, C4, C5$ の各イベントの実行時間範囲は図 2 のようになる．

4.2 スケジューリング木

前節で説明したように，各イベントの実行時間範囲はそれぞれのパスの組合せに対して得られる．いくつかの組合せに現れるイベントの実行時間の範囲は，得られた時間の範囲の和集合となる．各イベントの実行時間範囲は，実行可能な組合せが減ったとき，動的に変化するかもしれない．いくつかの組合せは，(1) 分岐状態のイベントが実行されたとき，または，(2) ある決まった時間範囲でイベントが実行されたとき，に無効になる．

例として，図 1 の EFSM 群の初期状態において，系列 $P31$ のイベント b は，系列 $P22$ のイベント b と時間範囲 $[9, 9]$ で同期するか，または，系列 $P23$ のイベント b と時間範囲 $[2, 2]$ で同期する (図 2)．よって，系列 $P31$ のイベント b の初期実行時間範囲は $[2, 2] \cup [9, 9]$ となる．また，組合せ $C1, C3, C4$ の初期の実行から，系列 $P31$ のイベント no の初期実行時間範囲は $[6, 17]$ となる．イベント b が時刻 9 で実行されたとき，実行可能な組合せは $C1$ のみであり，イベント no の時間範囲は $[13, 17]$ に狭められる．このように，どの組合せがまだ実行可能であるかについての情報を保持することによって，各イベントの実行時間範囲を知ることができる．

提案手法では，スケジュール可能な組合せと各イベントの実行時間範囲についての情報によって，図 4 に示すようなスケジューリング木を構築した．スケジューリング木の各ノードは系列の現在スケジューリング可能な組合せを，各枝はイベントの実行を示す．図 1 の並行周期 EFSM では，まず，5 つのスケジュール可能な組合せ $C1$,...,$C5$ が図 4 で示される．もし，イベント ni が時刻 4 で実行された場合，遷移 $ni[4, 4]$ が選ばれる．そして，スケジューリング木の次のノードで，$C1, C2, C3$ のみがスケジューリング可能となる．

5 回路の構成法

5.1 全体構成

ここでは，与えられた並行周期 EFSM に対応するハードウェア回路をどのように構成するか説明する．図 3 に示すように，以下の 3 つのサブモジュール，(1) 同じクロックで動作する各 EFSM に対応する順序回路，(2) マルチランデブコントローラ，(3) 各時刻で EFSM のスケジュール可能なイベント系列のみを示すスケジューラ，を用いて回路を構成する．

各 EFSM の順序回路は，(a) 現状態を保持する状態レジスタ，(b) 周期の最初からの経過時間を保持する時間レジスタ，(c)EFSM で使うデータ値を保持するその他のレジスタを持つ．EFSM は，そのクロックサイクルで現在の状態で実行可能なイベントの候補の中から 1 つを実行する．実行可能でないなら，その状態にとどまる．

マルチランデブコントローラは，(i) 同期イベントの対がそのクロックサイクルで実行可能かそうでないかをチェックする部分，(ii) 実行可能ではあるが同時に実行できない対を選択する衝突を回避する部分，からなる．上記の (i) の部分は，各同期イベント対から構成される．同期イベント対をなす全てのイベントが実行可能

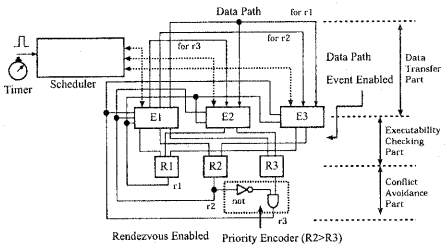

図 3: 回路の構成

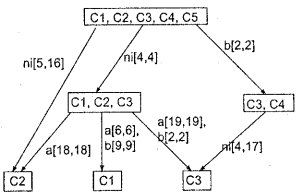

図 4: 図 1 のスケジューリング木の例

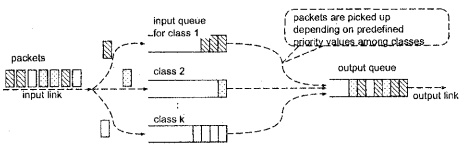

図 5: WFQ の構成

になったときその対自身が実行可能となるので，それらは AND 回路として実装される．(ii) の部分に対しては，複数のイベント対から 1 つを選択する方法はいくつか考えられるが，ここでは単純な方法としてこれらの対に優先順位を付けその順に選択する．この部分は図 3 のようにプライオリティエンコーダとして実装可能である．詳細は [9] にて報告されている．また，スケジューラ回路の構成法についても，詳細は [9] にて報告されている．

5.2 ツール

我々は，E-LOTOS で記述された仕様から，イベントの実行時間や変数の範囲を得るためのツールを実装した．そのツールは ILP 問題を解く *lp_solve* とよばれるフリーウェアを使用している．また，オリジナルの仕様や上記のツールによって得られた情報から，RT レベルの VHDL 仕様を生成する，ハードウェア合成ツールも実装している．上記の 2 つのツールと市販の VHDL 合成ツールを用いることで，提案した並行周期 EFSM モデルで書かれた高位の仕様から，ハードウェア回路を自動的に合成することが可能である．

6 アプリケーション記述と評価実験

6.1 優先度付きキューイングメカニズム

現在，提案されている優先度付きキューイングメカニズムには deficit round robin(DRR), weighted fair queuing (WFQ)[11], class-based queuing(CBQ) など，様々なものがある．ここでは，正確なパケットスケジューリングを供給し，他よりも複雑な計算を必要とする WFQ を例として取り上げる．

WFQ メカニズムでは，各パケットはそれぞれ *class* と呼ばれる優先順位を持っており，図 5 のように *class* によって異なるキューを使用する．各クラスは固定値の優先度を持っており，その優先度の割合に応じて各クラスの出力確率が決まる．例えば，3 つのクラス class1, class2, class3 がそれぞれ 0.2, 0.5, 0.3 の優先度を持っているとする．このとき全てのキューがアクティブならば，class2 には全体の 50%の帯域が割り当てられる．もし，class1，class2 の 2 つのキューのみがアクティブなら，帯域全体をこの 2 つのキューで各優先度によって比例配分される (詳細は [11] 参照)．

図 6 では，WFQ メカニズムを以下の 4 つの部分に分けて構成した．

$$(Pi[[a]|(Q(1)|||...|||Q(CMAX))|[b]|Po)|[a,b]|Sch$$

ここで，各 $Q(i), 1 \leq i \leq CMAX$ は第 i 番目のキューのパケットを出し入れを取り扱う．合計で，$CMAX+3$ 個の周期 EFSM が並列に動作される．図 6 において，n_in と n_out はそれぞれネットワークからルータへの入力リンクと出力リンクに対応する．そして，a と b は，それぞれ，Pi と $Q(1),...,Q(CMAX)$ の中の 1 つとの，$Q(1),...,Q(CMAX)$ の中の 1 つと Po とのデータ受け渡しの内部ゲートに対応する．

容易にスケジューリングアルゴリズムを修正して置き換えることができるように，ここでは制約志向の記述スタイルを使って Sch を別のモジュールとして作成する．図 7 に WFQ スケジューラの仕様例を示す．ここでは，スケジューラは Pi, $Q(i)$, Po のゲート a, b の動作制約によって記述される．これは，DDR のような単純なスケジューリングアルゴリズムに簡単に置換え可能である．図 6 と図 7 に示される仕様では，以下に示す制約パラメータを使用する．

P	全 EFSM の周期
$CMAX$	クラスの最大数
$QMAX$	各キューの最大深さ
$Rate[i]$	クラス i の優先度
MT	各キュー から (へ) のパケット リード (ストア) 時間
OT	出力リンクからのパケット出力時間
CLP	代入などの動作の実行所要時間 (=1 クロックサイクル)

また，以下の変数も使用する．

B	アクティブなクラスの集合． B は CMAX ビットの整数型変数で表現される．
$T[i,k[i]]$	クラス i のキューの $k[i]$ 番目のパケットが転送終了する時刻．
ar	$\sum_{j \in B} Rate[i]$: アクティブなキューの優先度の総和．
$queue[i]$	クラス i のキューを表現する変数．

図 6 において，全ての EFSM は時間周期 P で周期的に動作する．EFSM Pi はゲート n_in からパケットを入力し，そのパケットのクラスに応じて $Q(i)$ へ同期ゲート a を通じて転送する．また，パケットが到着しないときはなにもしない．$Q(1),...,Q(CMAX)$ は，以下の働きを行う．(i) 高々1 つの EFSM $Q(i)$ が同期ゲート a からクラス i のパケットを取り出し，それを $queue[i]$ の最後尾に加える．(ii) 高々1 つの EFSM$Q(j), j \neq i$ が同期ゲート b へ $queue[j]$ の先頭のパケットを送信し，そのパケットをキューから削除する．(iii) それ以外の EFSM $Q(k)$ はなにもしない．EFSM Po は同期ゲート b からパケットを取り出し，同期ゲート n_out から出力する．

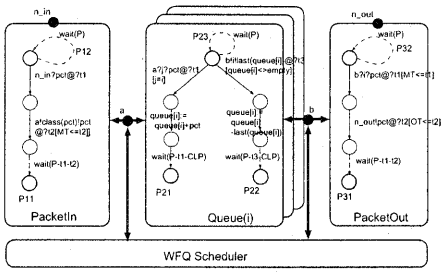

図 6: WFQ のモジュール

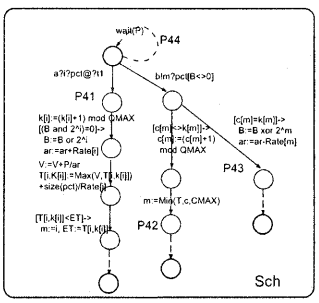

図 7: WFQ スケジューラ

また，アクティブなキューがないときはなにもしない．上記の仕様では，ただ 1 つのパケットのみが周期 P で処理できる．さらなる並列化を望みたいときは，Pi と Po の仕様を以下のように多重化したものに修正すればよい．

$$((Pi|||...|||Pi)|[a]|(Q(1)...Q(CMAX))|[b]|$$
$$(Po|||...|||Po))|[a,b]|Sch$$

図 7 は，WFQ アルゴリズムを基に $CMAX$ 個のキューから出力リンクへ出力させるべきパケットの順序をどう決めているかを示している．そこでは，各パケットの転送終了期待時刻の計算がなされる．各パケットが到着したとき，期待時間は図 7 の $P41$ にある B や ar の値を基に計算される．早い期待時間を持つパケットは，$P42$, $P43$ の同期ゲート b から出力が可能となる．系列 $P42$ において，$Min(T,c,CMAX)$ は $CMAX$ 個のキューの中で最も早い期待時間をもつパケットのクラス番号を計算する部分である．この計算は $CMAX/2$ 個の比較器を必要とし，$log_2 CMAX$ クロックサイクルを消費する．

6.2 評価実験

5 章で示したツールを用いて，図 6，図 7 に示す WFQ の仕様から，RT レベルのハードウェア回路を合成した．

回路合成の前に，前節で説明した制約パラメータの適切な値を割り当てておく必要がある．パラメータ条件は 3 説に示した技術によって，$MT + OT \leq Period$ かつ $4CLP + MT \leq Period$ となった．もし，$OT \geq 4CLP$ ならば，$OT + MT \leq Period$ はデッドロックを回避するために保たなければならない．

WFQ 回路のゲート n_in, n_out は，内部メモリ $queue[1]$, $\cdots$, $queue[CMAX]$ に 16 ビットのデータバスで連結されているものとする．よって，もし 400MHz DDR-SDRAM を用い，パケット最大長を 1,500 bytes とすれ

EFSM\event	*n_in?pct*	*a?i?pct*	*b!i!pct*	*n_out!pct*
Pi	[1,375]	[2250,2250]	—	—
Q(i)	—	[2250,2250]	—	—
Q(j)	—	—	[1875,1875]	—
Q(k)	—	—	—	—
Po	—	—	[1875,1875]	[3750,3750]
Sch	—	[2250,2250]	[1875,1875]	—

表 2: 周期 $Period = 3,751\ n$ sec. の時の各イベントの実行時間範囲

ば，パラメータ MT は $1,500*8/16/400MHz = 1.875\mu sec$ となる．ここで，$queue[i]$ の各エントリはメモリ上のどこかを指すポインタであるとする．$QMAX = 256$ と仮定すると，それらのポインタは 8 ビットの整数として実装される．

次に，OT と CLP を決定する．OT はパケットを出力バッファにコピーする時間と考えられる．そこで $OT = MT = 1875$ とする．全ての変数はレジスタとして，全ての関数は論理回路の組合せとして実装されるとする．CLP は合成された回路では，1 つの遷移を実行する 1 クロックサイクルとなる．もし，合成された回路の最大周波数を MC とするなら，CLP は $1/MC$sec となる．同様に，5 節で説明したように，組合せ論理回路として各多重同期の同期条件の評価を実装するので，各多重同期 (やそのモジュール間でのデータ転送) も，$1/MC$ sec で実行される．

そして，まず，図 6 と図 7 の仕様から時間制約を削除したものでハードウェア回路を合成する．仕様から我々のツールを使って VHDL 記述を得，ALTERA 社のツール Quartus II を使って FPGA デバイス $FLEX10KEEP10K200S$ に回路を実装した場合の情報を得た．その結果，回路の最大クロック周波数として 40.49MHz，ロジックゲート数約 960 ロジックエレメントの合成結果を得られた．よって，$CLP \approx 1/40MHz = 250n$ sec と分かった．

上記の議論によれば，設定パラメータとして $CMAX = 8$, $MT = 1,875n$ sec., $OT = 1,875n$ sec.,and $CLP = 250n$ sec. を使った．

全 EFSM の周期ができるだけ短くなるように上記の設定を用いた図 1 の仕様を我々のツールを適用した．そして，得られた周期は 3,751 $n$sec であった．表 2 にイベントの実行時間範囲を示す．ここで，単位時間は n sec. とする．

周期として $P = 3,751n$ sec. を使うとき，結果の回路は最大で秒間 266,595 パケットを処理可能である (これはパケットサイズを 1,500 bytes としたとき 3.19 Gbps に相当する)．より速いメモリやデータバスを使用すると，最大処理能力はさらに向上できる．以上より，提案手法がギガビットネットワークのための WFQ の実装に十分適用可能である．

上記の情報を使って，スケジューリングメカニズムを含む WFQ の全回路の合成を行った．最大クロック周波数は 37.74 MHz，回路規模は 1,074 ロジックエレメントであった．ここで，回路の大部分がキューのアドレステーブル (ただし，RAM へのゲートとそのコントローラは含まれていない) と整数値の比較器に使われている．スケジューリングメカニズムに使用されたロジックエレメントは約 114 である．この結果は，提案するスケジューリングメカニズムがシンプルで効果的であることを示している．この結果は，十分実用的であると思われる．

7 まとめ

本稿では，(a) 並行実時間処理を扱うのに適当な並行周期 EFSM モデル，(b) パラメトリックモデル検査法と実時間スケジューリング技術を基にした高位合成法，を提案した．提案手法を用いて WFQ メカニズムを持つ QoS ルータ回路の合成を行った．実験結果より，提案手法は優先度付きキューイングメカニズムのような実時間システムを実際に設計し開発するのに十分役立つと考えられる..

制限された資源を EFSM 間で共有することによって，得られる回路のコストを削減することが可能である．資源の共有を考慮した回路合成法を考案することが今後の課題である．

参考文献

[1] R. Alur, T. A. Henzinger, and P. Ho, "Automatic symbolic verification of embedded systems," *IEEE Trans. on Soft. Eng.*, vol. 22, no. 3, pp. 181–201, 1996.

[2] O. Bringmann, W. Rosenstiel and D. Reichardt : "Synchronization detection for multi-process hierarchical synthesis", *Proc. of Int. Symp. of System Synthesis (ISSS98)*, 1998.

[3] B. P. Dave and N. K. Jha : "CASPER: Concurrent Hardware-Software Co-Synthesis of Hard Real-Time Aperiodic and Periodic Specifications of Embedded System Architectures", *Proc. of 1998 Design Automation and Test in Europe (DATE '98)*, pp.118-125, 1998.

[4] W. Ecker and M. Huber : "VHDL based communication and synchronization synthesis", *Proc. of European Design Automation Conf. (EURO-DAC '95)*, 1995.

[5] M. Hennessy and H. Lin, "Symbolic bisimulations," *Theoretical Computer Science*, vol. 138, pp. 353–389, 1995.

[6] T. A. Henzinger, P-H. Ho and H. Wong-Toi : "HYTECH: A Model Checker for Hybrid Systems", International Journal on Software Tools for Technology Transfer, vol. 1, no. 1-1, pp.110–122, 1997.

[7] ISO : "Information Processing System, Open Systems Interconnection LOTOS", *ISO 8807*, 1989.

[8] ISO : "Final Committee Draft 15437 on Enhancements to LOTOS", *ISO/IEC JTC1/ SC21/ WG7*, 1998.

[9] H. Katagiri, K. Yasumoto, A. Kitajima, T. Higashino and K. Taniguchi : "Hardware Implementation of Communication Protocols Modeled by Concurrent EFSMs with Multi-Way Synchronization", *Proc. of 37th Design Automation Conf. (DAC'2000)*, pp. 762–767, 2000.

[10] A. Nakata, and T. Higashino: "Deriving Parameter Conditions for Periodic Timed Automata Satifying Real-Time Temporal Logic Formulas," in *Proc. of 21st IFIP Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE2001)*, pp. 151-166, 2001.

[11] A. Parekh and B. Gallager : "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks — The Single Node Case —", *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, pp. 344–357, 1993.

[12] C. A. Vissers, G. Scollo, M. v. Sinderen, and E. Brinksma: "Specification Styles in Distributed Systems Design and Verification", *Theoretical Computer Science*, vol. 89, no. 1, pp. 179 – 206, 1991.