社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＳＬＤＭ－129／2007－ＥＭＢ－4（５）
2007／3／15

チップ内ネットワークの性能要求検証および最適化のための一手法

村井 渉[†] 林 大輔[††] 中田 明夫[†] 木谷 友哉[†††] 安本 慶一[†††]
東野 輝夫[†]

† 大阪大学 大学院情報科学研究科
†† 大阪大学 基礎工学部
††† 奈良先端科学技術大学院大学 情報科学研究科
E-mail: †{w-murai,nakata,higashino}@ist.osaka-u.ac.jp, ††d-hayasi@ics.es.osaka-u.ac.jp,
†††{t-kitani,yasumoto}@is.naist.jp

あらまし 本研究では，通信の動的な振舞いを考慮したチップ内ネットワークの性能要求検証手法，および，通信スケジューリングの最適化手法を提案する．提案する性能要求検証手法では，時間ペトリネットによって，チップ内ネットワークの動的振舞いをミクロ的にモデル化する．次に，通信のレイテンシ・スループットに関する要求性能を満たし，かつ，デッドロックが起こらないことをモデル検査によって検証する．通信スケジューリング最適化手法では，アプリケーションの各通信フローに対して，まず，レイテンシ要求を満たすために当該フローが通過するルータ群に対する時間制約を導出する．次に，得られた制約に基づいて，静的スケジューリング理論に基づく TDMA タイムスロット割り当て手法を適用することにより，より効率的にルータ資源を利用し，かつ，性能要求を満たす通信スケジューリング解の探索を行う．
キーワード チップ内ネットワーク，モデル検査，時間ペトリネット，最適化，スケジューリング

A Method for Verifying Performance Requirements and Optimizing Communication Scheduling of Network-on-Chip

Wataru MURAI[†], Daisuke HAYASHI[††], Akio NAKATA[†], Tomoya KITANI[†††],
Keiichi YASUMOTO[†††], and Teruo HIGASHINO[†]

† Graduate School of Information Science and Technology, Osaka University
†† Faculty of Engineering Science, Osaka University
††† Graduate School of Information Science, Nara Advanced Institute of Science and Technology
E-mail: †{w-murai,nakata,higashino}@ist.osaka-u.ac.jp, ††d-hayasi@ics.es.osaka-u.ac.jp,
†††{t-kitani,yasumoto}@is.naist.jp

Abstract In this paper, we propose a method for verifying performance requirements of Network-on-Chip(NoC) considering dynamic communication behavior of IP cores, and an optimization method of packet scheduling to guarantee performence requirements. In the proposed verification method, a microscopic dynamic behavior of an entire NoC architecture including communication behavior of IP cores is modeled by Time Petri Nets, and then whether given latency/throughput requirements for all communications are satisfied is checked using model checking. In the proposed optimization method, first, time constraints are assigned for each NoC router so that the given latency requirement can be satisfied for each communication flow of an application, and based on that, a TDMA scheduling for each router and each flow is derived according to the static real-time scheduling theory. By iteratively applying the verification method and the refining the TDMA scheduling in a certain policy, a resource-efficient and performance-guaranteed NoC communication scheduling can be explored.
Key words Network on Chip, Model Checking, Time Petri Nets, Optimization, Scheduling

1. まえがき

近年，System on Chip (SoC) の大規模化，微細化が進むにつれて，ゲート遅延よりも配線遅延の影響が大きくなってきている [1]．一方で，SoC を構成する機能モジュール (IP コア) の間の通信アーキテクチャの主流はバスである．しかし，システムの大域的な配線の遅延により，微細化が進んでもバスクロック周波数を上げることは困難になっており，より高性能なチッ

プ内通信アーキテクチャが求められている．そのような問題の解決のため，チップ内ネットワーク (Network on Chip:NoC) の研究が盛んに行われている [1]〜[6] .

NoC ではバスの代わりにルータ (交換器) を配置し，複数の通信路を共有化するとともに通信のパイプライン化を行い，個々の配線長を短く保ちつつスループット性能の向上を図っている．NoC ルータのアーキテクチャは図 1 に示すように，複数の入力ポートに入力バッファを持ち，それぞれのポートからの入力をスイッチによって複数の出力ポートのいずれかに転送するという構成が一般的である [1]．このとき，特に NoC では回路面積・消費電力のコスト削減のため，スイッチの構造はできるだけ単純にし，バッファサイズも小さくすることが望まれる．しかし，バッファサイズが小さい場合は，ルータの転送先入力バッファの空きを待つためにデータ転送が停止する可能性があり，複数ルータが相互に入力バッファの空きを待つことによってデッドロックに陥る可能性がある．また，デッドロックに陥らない場合でも，複数の通信が同一ルータ上で競合することによりデータ転送に遅れが生じ，通信のスループットやレイテンシに関する制約を保証できない可能性がある．

NoC のデッドロック問題に対する既存の解決法としては，通信開始から終了まで通信経路上のルータを占有する方法 (Virtual Channel [1]) や，デッドロックフリーのルーティング手法 ([7] など) が提案されている．Virtual Channel はルータや通信経路の数が少ないと同時通信可能な通信路 (チャネル) 数が減少する問題がある．デッドロックフリーのルーティング手法はメッシュなどの規則的トポロジを前提としており，アプリケーションに特化した最適化を行いにくい．低コスト・低消費電力の SoC 設計のためには，アプリケーションに特化した通信性能要求を満たし，かつデッドロックのない最適な NoC を設計できることが望ましい．

既存の NoC やバスアーキテクチャの最適化手法は，通信デッドロックがないことを保証するアーキテクチャを前提に設計最適化を行う手法 [3]〜[6] と，シミュレーション検証により，より自由度の高い設計最適化を行う手法 [8] に大きく分類される．前者では最適化の自由度がより小さくなり，通信路資源の利用率に無駄が生じる可能性がある一方，後者のシミュレーションによる検証では，デッドロックが生じないことを完全に保証することは困難という問題がある．

また，既存手法の多くは全ての通信路が同時に最大通信帯域を使用するという最悪の場合の性能要求を保証するように設計最適化を行っている．しかし，アプリケーションによっては，ある IP コア間では，ある時点ではバースト的に通信を行うが，別の時点ではほとんど通信を行わないなど，通信路の使用に時間的なばらつきがある場合がある．そのような場合，各 IP コアの通信動作の動的振舞いを考慮し，ルータ資源の使用をうまくスケジュールすることにより，従来よりも少ない資源で性能要求を保証できる可能性がある．

そこで，本研究では通信の動的振舞いを考慮したチップ内ネットワークの性能要求検証手法，および，通信スケジューリングの最適化手法を提案する．対象とするチップ内ネットワークは，複数の入出力ポート，有限サイズの入力バッファ，および時分割多重スケジューラ (以下，TDMA スケジューラと呼ぶ) で構成される単純なルータ群で任意個の IP コアを結合したものとする．ネットワークトポロジは任意のものがあらかじめ与えられるものとする(注1)．TDMA スケジューラは，時間軸を複数個のタイムスロットに分割し，各タイムスロットにおいては，指定された通信フローのみにルータの使用を許可する調停機構であるとする．

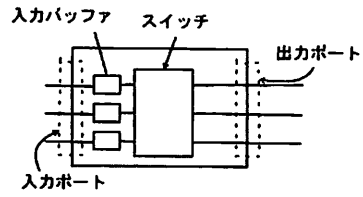

図 1　NoC ルータの一例

性能要求検証では，チップ内ネットワークの通信動作仕様，すなわち，通信フローの集合およびそれらの経路情報（通信元から通信先までに通過するルータ列），IP コアの集合およびそれらの通信動作仕様（状態遷移図，各状態における送信先/受信元 IP コア，通信量）および各通信フローに対する性能要求 (スループット/レイテンシ制約) が与えられたとき，チップ内ネットワークの各通信動作が性能要求を満たし，かつ，ネットワーク全体でデッドロックが起こらないことの検証を行う．検証は，通信の動的振舞いを時間ペトリネット [9], [10] でモデル化し，モデル検査 [11] によって行う．モデル検査には時間ペトリネットに対する既存のモデル検査ツール [12] を用いる．

通信スケジューリングの最適化では，まず，各通信フローにおいて各ルータがパケットを処理する最大時間をそのフローのレイテンシ制約を満たせるように，また，各ルータがパケットを処理する最小時間をそのルータを通るフロー群のスループット制約を満たせるように定める．次に，得られた各ルータでの各フローのパケットの最小・最大処理時間およびスループット制約に基づいて，デッドラインモノトニックスケジューリング [13], [14] を用いた TDMA タイムスロットの割り当てを行う．得られた割り当てを TDMA タイムスロット割り当ての初期解とし，このもとで前述の性能要求検証を行う．もし性能要求が満たされなければ，性能要求が満たされない原因となるフローを特定し，そのフローに対して，ヒューリスティックによるタイムスロット割り当ての改善を行う．これを性能要求が満たされるまで繰り返すことにより最適化を行う．

提案する性能要求検証手法は通信の動的振舞いをミクロ的に解析しているのが特徴である．通信路上の通信仕様が時間的に変化する場合にも対応している．したがって，時間的に変化する通信動作を行うアプリケーションにも対応可能であり，通信路の最悪時の制約を考慮するより最適化の余地が増える．性能要求検証はモデル検査で行っており，通信の競合によるデッドロックがないことを完全に保証できる．また，提案する通信スケジューリング手法により，与えられた通信トポロジにおいて性能要求を満たす解を，TDMA タイムスロット割り当てを工夫することにより得ることができる．

以降の本稿の構成は以下の通りである．まず，2. では提案する NoC 性能要求検証手法について述べる．3. では TDMA タイムスロット割り当てによる通信スケジューリング最適化手法について述べる．最後に，4. にて本稿のまとめと今後の課題について述べる．

(注1)：本稿ではネットワークトポロジの最適化は扱わない．

2. チップ内ネットワークの性能要求検証

この章では，与えられたチップ内ネットワークが与えられた性能要求を満たしているか否かを検証する手法について述べる．

2.1 対象とするチップ内ネットワークアーキテクチャ

本研究が対象とするチップ内ネットワークアーキテクチャは，複数個の **IP コア**が複数個の**ルータ**によって相互結合されたものである．ネットワークトポロジは任意であるとする．各 IP コアは複数の (通信に関する) **状態**を持ち，各状態において，他のどの IP コアとどのくらいの量のデータを送信/受信するか，および送受信終了時の遷移先状態が定まっているとする．各 IP コアはデータを固定長の単位 (**パケット**) に分けて送受信するものとする．IP コア間の各通信は，送信元 IP コア sc，sc の送信時の状態，経路情報 (どのルータをどのような順序で通過して送信先 IP コアに届けるか)，送信先 IP コア rc，rc の受信時の状態，の 5 つ組で識別される．これを (通信) **フロー**と呼び，各パケットにはフローの識別番号 (ID) が付与されているものとする．各ルータは固定個の**入力ポート**および**出力ポート**を持ち，入力ポートから受信したパケットを経路情報で指定された出力ポートに転送する．転送方式としては蓄積交換方式 (store and forward) を仮定し，同時に高々 1 つのパケットを転送可能とする．各入力ポートには入力データを一時的に蓄積する**入力バッファ**を持つ．入力バッファは有限長の FIFO キューで実装されるものとする．複数の入力データが競合した場合の調停は **TDMA スケジューラ**で行うものとする．TDMA スケジューラのアーキテクチャとしては次のようなものを仮定する．まず，各ルータには TDMA の**周期**が与えられ，周期をいくつかの**タイムスロット**に分割し，各タイムスロットに対して，それぞれどのフローを通過させるかが指定されているとする．各ルータは，各周期開始時からの時刻 (クロックサイクル) をカウントし，現在どのフローを通過させるタイムスロットにあるかを識別する．そして，もし対応するフローの ID を持つパケットが入力バッファの先頭に到着していたら，そのパケットを経路情報で指定された出力ポートに転送する．

2.2 チップ内ネットワーク仕様

$\mathbf{N}$ を正整数の集合，$\mathbf{N_0}$ を非負整数の集合，$\mathbf{R}^+$ を非負実数の集合とする．IP を IP コアの集合とし，任意の IP コア $ip \in IP$ に対して，$S(ip)$ を ip の通信状態集合，$TR(ip) : S(ip) \times (\{send, recv\} \times F \times \mathbf{N} \times \mathbf{R}^+) \cup (\mathbf{R}^+ \times (\mathbf{R}^+ \cup \{\infty\})) \times S(ip)$ を ip の遷移関係，$s_0(ip) \in S(ip)$ を ip の初期状態とする．$(s, (send, f, n, t), s') \in TR(ip)$ ならば，ip の状態 s において n 単位個のデータを t 単位時間間隔でフロー f を通じて送信し，送信終了後に状態 s' に遷移するとする．$(s, (recv, f, n, t), s') \in TR(ip)$ ならば，状態 s において n 単位個のデータをフロー f を通じて受信し，受信終了後に状態 s' に遷移するとする．また，$(s, (t_1, t_2), s') \in TR(ip)$ $(t \in \mathbf{R}^+)$ ならば，時間区間 $[t_1, t_2]$ 内のある時間 t 後に s から s' に遷移するとする．$CB(ip) = \langle S(ip), TR(ip), s_0(ip) \rangle$ を ip の **IP コア動作仕様**と呼ぶ．R をルータの集合とし，任意の異なる IP コア $ip_1, ip_2 \in IP$，ip_1 の任意の状態 $s_1 \in S(ip_1)$，ip_2 の任意の状態 $s_2 \in S(ip_2)$，ルータの任意の有限列 $r_1, \ldots, r_k$ $(\forall k \in \mathbf{N})$ に対して，列 $f = \langle (ip_1, s_1), r_1, \ldots, r_k, (ip_2, s_2) \rangle$ を**フロー**と呼ぶ．フロー f は IP コア ip_1 の状態 s_1 から送信されたパケットがルータ $r_1, \ldots, r_k$ をこの順に経由し，IP コア ip_2 の状態 s_2 で受信されることを意味する．フローの集合を F とする．

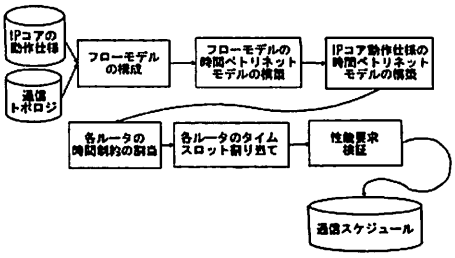

図 2　性能要求検証の流れ

$d \in \mathbf{R}^+$ を各ルータにおけるパケットの**最小処理時間** (全ルータ共通) とする．すなわち，各ルータ $r \in R$ は入力ポートから入力バッファにパケットを受信してから最短で d 単位時間後に出力ポートにパケットを転送可能とする．d はルータの動作周波数およびルータアーキテクチャより定まる値とする．任意のルータ $r \in R$ に対して，$TS(r) = \langle (t_1, f_1), \ldots, (t_k, f_k), T \rangle$ をルータ r の **TDMA スケジューラ仕様**と呼び，$T \in \mathbf{R}^+$ をルータ r の **TDMA 周期**と呼ぶ．すなわち，$t_{k+1} \stackrel{\text{def}}{=} T - \Sigma_{i=1}^{k} t_i$ としたとき，時間区間 $[(\Sigma_{i=1}^{j} t_i) + m \times T, (\Sigma_{i=1}^{j+1} t_i) + m \times T)$ $(j \in \{1, \ldots, k\}, m \in \mathbf{N_0})$ においては，r が入力バッファに保持している全パケット群のうち，f_j に属するパケットのみが r の出力ポートに送信可能とする．

このとき，$NS = \langle IP, CB, R, F, d, TS \rangle$ を**チップ内ネットワーク仕様**と定義する．

2.3 性能要求仕様

F を任意のフロー集合とする．F の各要素 $f \in F$ に対して非負実数値 $LC(f) \in \mathbf{R}^+$ が与えられ，フロー f の各送信データは送信時から $LC(f)$ 単位時間以内に受信側 IP コアに届かなければならないことを要求するとき，F から $\mathbf{R}^+$ への関数 LC を F の**レイテンシ制約**と呼ぶ．F の各要素 $f \in F$ に対して非負実数値 $TH(f) \in \mathbf{R}^+$ が与えられ，フロー f は単位時間内に $TH(f)$ 単位量のデータを送信できなければならないことを要求するとき，F から $\mathbf{R}^+$ への関数 TH を F の**スループット制約**と呼ぶ．LC と TH の組 $RS = \langle LC, TH \rangle$ を F の**性能要求仕様**と定義する．

2.4 性能要求検証問題

チップ内ネットワーク仕様 $NS = \langle IP, CB, R, F, d, TS \rangle$ が NS のフロー集合 F に対する性能要求仕様 $RS = \langle LC, TH \rangle$ を**満たす**とは，NS で指定された仕様に従い各フロー $f \in F$ の通信を行ったときに，全てのフロー $f \in F$ が常にレイテンシ制約 $LC(f)$ およびスループット制約 $TH(f)$ を満たして通信できることであると定義する．

このとき，**性能要求検証問題**とは，チップ内ネットワーク仕様 $NS = \langle IP, CB, R, F, d, TS \rangle$ および NS のフロー集合 F に対する性能要求仕様 $RS = \langle LC, TH \rangle$ が与えられたとき，NS が RS を満たすか否かを判定する問題であると定義する．

2.5 性能要求検証手法

検証は図 2 に示す流れに従い，チップ内ネットワークを時間ペトリネットでモデル化してモデル検査によって行う．

時間ペトリネットによるモデル化は次の方針で行う．まず，送信側 IP コアから受信側 IP コアに通信データが到達するまでに経由するルータの順列に着目し，各パケットの動的および時間的挙動をモデル化する．各パケットは各ルータである時間以上ある時間以内に処理されるものとし，各ルータの入力バッ

ファには高々バッファサイズの数しかパケットが存在しないようにモデル化する．転送先ルータの入力バッファが満杯の場合はバッファに空きが生じるまで待つようにする．また，各ルータが複数フローのパケットを同時に一つのみ転送可能であることを，相互排他によって表し，TDMA スケジューラの各タイムスロットにおいて転送を許されたフローのパケットのみを転送する動作をモデル化する．IP コアの通信動作は有限状態機械で表し，送信動作は指定した数のパケットを指定した時間間隔で送信して次状態に進む動作で，受信動作は指定した数のパケットを受信したら次状態に進む動作で表す．性能要求としては，各通信フロー毎にそのフローのレイテンシ制約/スループットが与えられているものとする．スループット制約は各送信動作におけるパケットを送信する最小時間間隔の逆数とし，レイテンシ制約は送信 IP コアから受信 IP コアまでパケットが到達するまでの最大時間とする．

性能要求検証のため，スループット制約違反およびレイテンシ制約違反を検出し，モデル全体の動作をデッドロックさせる動作をモデル化し，デッドロック検証に帰着して行う．スループット制約違反検出のため，ある時点で送信したパケットが次のルータに転送される前に次のパケットを送信したら，モデル全体の動作を停止させデッドロックになるようにモデル化する．レイテンシ制約違反検出のため，フローにおけるパケット送信と同時にダミーデータをダミーのルータに送信し，受信側は当該フローにおいてデータを受信すると同時にダミーのルータからデータを取り去るようにモデル化し，ダミールータにレイテンシ制約以上滞留すると全体の動作を停止させ，デッドロックになるようにモデル化する．これらの方針により，与えられたチップ内ネットワーク仕様および性能要求から，ネットワークにおける通信の動的振舞いを表現し，かつ，スループット違反またはレイテンシ違反があればデッドロックするような時間ペトリネットモデルを構成し，デッドロックするか否かを既存ツールを用いて検証する．

2.5.1 フローのモデル化

各フロー $f \in F$ 毎に時間ペトリネットへの変換を行う．フローのデータ転送はノード間のデータ転送の繰り返しである．したがって，ここではノード間転送のモデルの動作を説明する．ノード間のデータ転送は入力バッファにデータが到着している状態 Arrive，データを出力可能な状態 Ready，転送処理を行っている状態 Trans の 3 つの状態に分け，順に遷移していくことで表現する (図 3)．前のノードからパケットが送られてくるとバッファにデータが到着している状態 Arrive になる．処理しているパケットがなければ，即座に出力可能状態 Ready になる．ルータアーキテクチャで定められた転送処理時間が経過すると転送処理状態 Trans になる．出力可能状態では次ノードの入力バッファが空いているか，ルータモデルのタイムスロットが自分のフローの割り当て時間であるかを確認し，どちらも満たしていれば，次のノードの Arrive 状態へ遷移する．

2.5.2 ルータのモデル化

ルータが一度に一つのパケットのみを処理するという動作は，異なるフロー間のパケットの相互排他動作によりモデル化する．フロー同士の競合が起こったとき，時分割多重方式で優先するフローを決定する TDMA スケジューラは次のようにモデル化する．各ルータごとに存在する TDMA スケジューラを，システムの周期の開始時からあらかじめ決められた時間が経過後，状態が遷移するものとしてモデル化する (図 4)．各状態においては，指定されたタイムスロット割り当てで指定されたフローに転送許可を与え続け，ある時間経過すると次の状態に遷移するようにモデル化する．

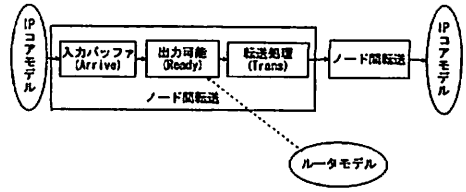

図 3 データ転送のモデル化

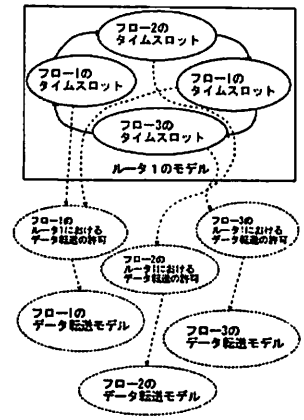

図 4 TDMA ルータのモデル化

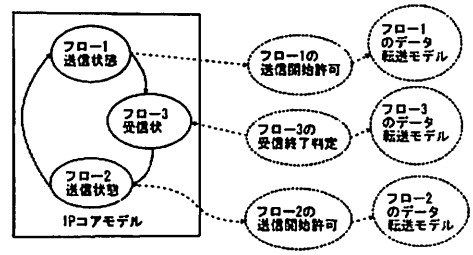

図 5 IP コア動作仕様のモデル化

2.5.3 IP コア動作仕様のモデル化

IP コア動作仕様はフローの送信状態と受信状態の 2 種類の状態を含む．このとき，フローの送信状態は送信フローのデータ転送モデルとつながっており，スループット制約から求まる送信間隔でパケットの送信動作を行う．全てのデータを送信したら次の状態に移る，受信状態では受信フローのデータ転送モデルと接続しており，受信フローの全てのパケットが送られてくるまで待機し，その後次の状態へ遷移する．図 5 はフロー 1 を送信し，フロー 3 を受信した後，フロー 2 を送信するという動作をする IP コアのモデルの概要図である．

3. 通信スケジューリングの最適化手法

通信スケジューリングの最適化では，まず，パケットが各ルータに滞留する時間制約を求め，次に，その時間制約に基づいて各ルータの TDMA タイムスロット割り当てをタスクスケジューリング理論に基づいて定めることにより，通信スケジューリン

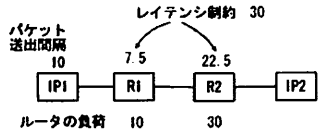

図 6 ルータの転送時間割り当ての例

グの初期解を求める．次に，初期解において性能要求が満たされるか否かを前述の手法により検証する．もし満たせない場合は，各ルータにおける時間制約およびTDMA スロット割り当ての改善を行い，再び性能要求検証を行う．これを，性能要求が満たされるまで繰り返すことにより，通信スケジューリングの最適化を行う．

3.1 初期解の見積もり

通信スケジューリングの初期解の見積もりは次のように行う．まず，各フローにおいてパケットが各ルータに滞留する最小時間を動作周波数およびルータアーキテクチャより，最大時間をそのフローのレイテンシ制約を満たせるように時間制約を定める．具体的には，まず，当該フローが経由する各ルータに対して，そのルータの負荷，すなわち，単位時間に処理する最大パケット数を，当該ルータを通る全フローのスループット制約の逆数の総和として求める．次に，当該フローのレイテンシ制約の値をフローが経由するルータ群の負荷に応じて比例配分し，得られた値を当該フローのパケットが当該ルータに滞留すべき最大時間として定める．次に，得られた各ルータでの各フローのパケットの最小・最大処理時間およびスループット制約に基づいて，デッドラインモノトニックスケジューリング[13],[14]を用いたTDMA タイムスロットの算出を行う．

3.1.1 ルータの時間制約の見積もり

ルータの時間制約の見積もりとは，任意のフロー $f = \langle (ip_1, s_1), r_1, \ldots, r_k, (ip_2, s_2) \rangle \in F$ に対して，f のレイテンシ制約 $LC(f)$ を満たすように各ルータ r_i $(i = 1, \ldots, k)$ における f の各パケットの処理時間の上限 D_i を見積もることである．本研究では，レイテンシ制約 $LC(f)$ の値をルータの負荷に応じて比例配分することで決定する方針を採る．ここで，ルータ $r \in R$ の負荷とは，r が単位時間に処理すべき最大パケット量，すなわち，r を通過する全フロー f のスループット制約 $TH(f)$ の合計，と定義する．フロー f の送信 IP コア ip_1 の状態 s_1 からのパケット送信間隔には，スループット制約の逆数 $1/TH(f)$ を設定する．

図 6 は転送時間割り当ての一例である．この例ではレイテンシ制約が 30，パケット送出間隔が 10 である．レイテンシ制約の値を経路上のルータに割り振る．ルータ R1 と R2 の負荷の比が 1:3 であるので，割り当て時間の比も同じ割合になるように決定する．

ルータの負荷に応じてパケットの処理量が多くなり，複数のフローが競合する確率が大きくなるが，レイテンシ制約に応じて時間制約をゆるくすることで，制約を満たすようパケットの処理を行うことができる．

3.1.2 TDMA タイムスロットの見積もり

ルータ $r \in R$ の TDMA タイムスロット見積もりとは，r を通過するフローの集合 $F(r)$ が与えられたとき，$F(r)$ の全フローを性能要求を満たせるような r の TDMA スケジューラ仕様 $TS(r) = \langle (t_1, f_1), \ldots, (t_k, f_k), T \rangle$ を見積もることである．

このとき，r による各フロー f_i のパケットの処理を，周期 $T_i = 1/TH(f_i)$，処理時間 d，(周期開始時からの) 相対デッドライン D_i であるような周期タスク τ_i とみなし，リアルタイムタスクスケジューリング問題に帰着する．相対デッドライン D_i は前節にて見積もった f_i のパケットの r における最大処理時間である．

$TS(r) = \langle (t_1, f_1), \ldots, (t_k, f_k), T \rangle$ の見積もりを次のように導出する．まず，スケジューリングの前処理として全体の周期 T とフローの優先度を求める．T は各 $T_i (i = 1, \ldots, k)$ の最小公倍数とする．フロー f_i の優先度は相対デッドライン D_i の値がより小さいフローがより大きくなるようにする．つまり，フロー群 $f_1, \ldots, f_k$ を対応する相対デッドラインの値 $D_1, \ldots, D_k$ の昇順でソートする．次に，時刻 0 にすべてのタスク $\{\tau_i\}_{i \in \{1,\ldots,k\}}$ が同時に到着したと仮定して，デッドラインモノトニックスケジューリングを実行する．すなわち，まず，現在時刻 t を 0 とし，t において同時に到着したフロー群 F_t のうち最も相対デッドライン D_i の小さいフロー f_i にタイムスロット (t, f_i) を割り当てる．割り当てるタイムスロットの最小長はルータにおけるパケットの最小処理時間 d とする．すなわち，t を $t + d$ に更新し，F_t を $F_{t-d} \setminus \{f_i\}$ と時刻 t で到着するフロー群の和集合に更新する．

もし $t < T$ かつ F_t が空集合ならば，t を F_t が空でないような最小の値まで増加させる．これを繰り返し，$t = T$ となった時点で，全てのフローへのタイムスロット割り当て $\langle (t_1, f_1), \ldots, (t_k, f_k) \rangle$，すなわち，ルータ r の TDMA スケジューラ仕様の見積もり $TS(r)$ を得る．タイムスロット割り当てを求める全体の流れは図 7 の通りである．また，タイムスロット割り当て例を図 8 に示す．3 つのフロー (F1,F2,F3) があり，全体の周期内で F1 は 3 つ，F2,F3 は 2 つのタスクを処理する場合の例である．フローの優先度は F1,F3,F2 の順に高いとしている．時刻 0 では全てのフローのタスクが到着する．フローの優先度順に処理するため，F1 のタスクからタイムスロットを割り当てる．F3 のタスクは F1 で割り当てられた直後のタイムスロットが割り当てられる．その後，再び F2 と F3 のタスクが同時に到着する．優先度に従い，F3 から割り当てる．F2 のタスクはその直後に割り当てられようとするが，その時刻には F1 のタスクが新たに到着する．そのため，F1 のタスクが先に割り当てられ，F2 のタスクはその直後にタイムスロットが割り当てられる．

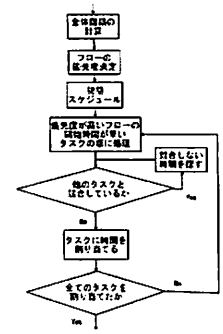

図 7 タイムスロット割り当てのフローチャート

3.2 解の改善

見積もったタイムスロット割り当てにおいて，もし性能要求が満たされなければ，次の方針で解の改善を行う．まず，性能

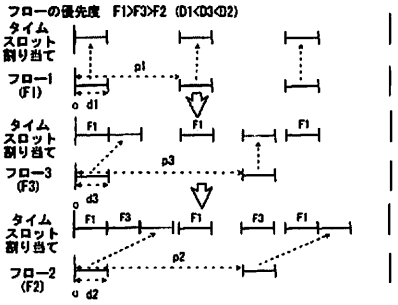

図 8 タイムスロット割り当て例

要求が満たされない原因となるフローを特定し，そのフローに対して，まず，各ルータにおけるパケット最大処理時間制約を改善する．

導出した時間制約で性能要求を満たせないということは，フロー間での通信の競合により，各ルータにおいて指定した時間制約以上に待たされていることになる．これを解消するためには，通信の遅れが生じたフローの優先度を上げてやればよい．すなわち，そのフローのレイテンシ制約をより厳しくして，各ルータにおける相対デッドラインを早めてやれば，デッドラインモノトニックスケジューリングにおける優先度を上げることができる．ただし，レイテンシ制約を厳しくしすぎると，スケジューリング不可能になり，動作周波数を上げる必要が生じ，電力効率の悪い解が得られてしまう．そこで，まず，競合するフローのうち，最も当該フローの遅れの原因になっている可能性が高いフローを選択し，そのフローに奪われているルータ資源利用時間を，当該フローのレイテンシ制約から減じる．具体的には，当該フローの各ルータを共有する他のフローのうち，(スループット制約)*(共有ルータ数) の値が最も大きいフローの共有ルータにおけるパケット最大処理時間制約の総和を，当該フローのレイテンシ制約から減じ，得られた値を新しいレイテンシ制約とみなして，各ルータの負荷に応じて比例配分した値を当該フローの各ルータにおける新しいパケット最大処理時間制約とする．

なお，レイテンシ・スループット制約の値およびパケット最小処理時間の値によっては，デッドラインモノトニックスケジューリングの解が存在せず，TDMA タイムスロットの値の導出ができない場合がある．その場合は，まず，性能要求が満たされない場合と同様に，タイムスロットの導出ができないルータを通過する各フローに対してパケット最大処理時間制約の改善を行う．この改善を行っても当該スケジューリングの解が得られない場合は，ルータの動作周波数を上げ，パケット最小処理時間制約を減らすことにより改善を行う．この改善により消費電力を増大するが，ルータの性能はいくらでも向上可能であるため，いつかは性能要求を満たす解を得ることができる．

4. あとがき

本稿では，状態によって通信動作を変化させる IP コア間を結合するチップ内ネットワークの性能要求検証手法および通信スケジューリング最適化手法を提案した．モデル検査手法を援用することにより，デッドロックフリーを保証しないチップ内ネットワークアーキテクチャに対しても，デッドロックが起こらないこと，および，レイテンシ/スループット制約を満たすか否かの完全な検証が可能となる．また，通信スケジューリング最適化手法により，ネットワークトポロジを変更することなく，性能要求を満たすことが保証される各ルータの TDMA タイムスロットの最適解を探索することが可能となる．

ただし，本手法では，TDMA によるスケジューリングが不可能な場合はルータの動作周波数を上げることになるが，これは消費電力の観点から望ましいことではなく，本来は通信トポロジの改善による最適設計探索を行うべきである．また，モデル検査による性能要求検証は，チップ内ネットワークの規模が増大すると計算時間が増大する問題がある．性能要求検証の効率化および通信トポロジの最適設計探索手法の開発は今後の課題である．

文 献

[1] T. Bjerregaard and S. Mahadevan: "A survey of research and practices of network-on-chip", ACM Computing Surveys, **38**, 1, pp. 1–51 (2006).

[2] J. Hu and R. Marculescu: "Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints", Proc. of the Design, Automation and Test in Europe Conf. and Exhibition (DATE 2004), IEEE Computer Society Press, pp. 234–239 (2004).

[3] K. Srinivasan, K. S. Chatha and G. Konjevod: "Linear programming based techniques for synthesis of network-on-chip architectures", Proc. of the 22nd IEEE Int. Conf. on Computer Design (ICCD 2004), IEEE Computer Society Press, pp. 422–429 (2004).

[4] K. Srinvasan and K. S. Chatha: "SAGA: Synthesis technique for guaranteed throughput noc architectures", Proc. of the 10th Asia and South Pacific Design Automation Conf. (ASP-DAC 2005), IEEE Press, pp. 489–494 (2005).

[5] S. Stuijk, T. Basten, M. Geilen, A. H. Ghamarian and B. Theelen: "Resource-efficient routing and scheduling of time-constrained network-on-chip communication", Proc. of the 9th EUROMICRO Conf. on Digital System Design (DSD 2006), IEEE Computer Society Press (2006).

[6] K. Srinivasan, K. S. Chatha and G. Konjevod: "Application specific network-on-chip design with guaranteed quality approximation algorithms", Proc. of the 12th Asia and South Pacific Design Automation Conf. (ASP-DAC 2007), IEEE Press, pp. 184–190 (2007).

[7] C. J. Glass and L. M. Ni: "The turn model for adaptive routing", Proc. of the 19th Annual Int. Symp. on Computer Architecture (ISCA 1992), pp. 278–287 (1992).

[8] S. Pasricha, N. Dutt, E. Bozorgzadeh and M. Ben-Romdhane: "Floorplan-aware automated synthesis of bus-based communication architectures", Proc. of the 42nd ACM/IEEE Int. Design Automation Conf. (DAC 2005), ACM Press, pp. 565–570 (2005).

[9] T. Murata: "Petri nets: Properties, analysis and applications", Proceedings of the IEEE, **77**, 4, pp. 541–580 (1989).

[10] B. Berthomieu and M. Diaz: "Modeling and verification of time dependent systems using time Petri nets", IEEE Trans. on Software Engineering, **17**, 3, pp. 259–273 (1991).

[11] E. M. Clarke, O. Grumberg and D. A. Peled: "Model Checking", MIT Press (1999).

[12] B. Berthomieu, P. O. Ribet and F. Vernadat: "The tool TINA – construction of abstract state spaces for petri nets and time petri nets", Int. Journal of Production Research, **42**, 14, pp. 2741–2756 (2004).

[13] N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings: "Hard real-time scheduling: The deadline monotonic approach", Proc. of the 8th IEEE Workshop on Real-Time Operating Systems and Software, pp. 127–132 (1991).

[14] 白川, 竹垣：“リアルタイムシステムとその応用”, システム制御情報ライブラリー, 第 22 巻, 朝倉書店 (2001).