組み込みシステムのアーキテクチャー設計と性能評価のためのモデル・ベース手法

坂本 佳史　豊田 学　小野 康一　河原 亮　長井 真吾　中田 武男

A Model-based Method for Architectural Design and Performance Evaluation of Embedded Systems

Yoshifumi Sakamoto, Manabu Toyota, Kouichi Ono, Ryo Kawahara, Shingo Nagai and Takeo Nakada

組み込みシステムの高性能化・高機能化により搭載されるソフトウェアは複雑化・肥大化している．既存のソフトウェア資産を有効に活用し，さらにマルチコアなどのテクノロジーにより高性能を実現するアーキテクチャー設計は，組み込みシステム開発の大きな課題である．本論文では，リバース・モデリング技術による既存システムのモデル化と，機能要求に基づく変更をそのモデルに適用する，モデルによるアーキテクチャー設計の手法について述べる．またシステム・パフォーマンスを検証するモデル・シミュレーション手法について述べる．これらの手法をアーキテクチャー設計に適用することで，高位レベルでのアーキテクチャー設計を可能とし，またシステム開発初期であるアーキテクチャー設計の段階でパフォーマンスの評価を可能とした．

As the functions and performance of embedded systems increase, software becomes more complicated and enlarged. High-performance architecture development that utilized both new technology such as the multi-core and software assets effectively, is a big issue in embedded system development. This paper reports on the method of model-based architecture design and the modeling technology of the existing system by reverse modeling and performance simulation for embedded system through model.
Through our methodology it is possible to assess architecture design through performance evaluation in the early stages of system development.

Key Words & Phrases：UML モデル，モデル駆動型開発，性能評価シミュレーション，リバース・モデリング，組み込みシステム
UML model, model driven development, performance simulation, reverse modeling, embedded system

1. はじめに

近年，マルチ・ファンクション・プリンター（MFP：Multi Function Peripheral/Printer）に代表される組み込みシステムでは，高性能化・高機能化の市場要求に対応してシステム性能は年々向上している．演算処理を担うプロセッサーなどに用いられる半導体の製造技術においては微細化・高クロック化に依存した処理性能の向上が発熱，消費電力の急激な増加により鈍化している．この問題を解決する 1 つの方法として，複数のプロセッサーによって分散処理を行うシステム・アーキテクチャーであるマルチコアの採用が挙げられる．マルチコア・アーキテクチャーを採用したシステムにおいては，並列動作が可能な複数のプロセッサーに割り当てる処理を最適化することが高いシステム性能の実現において重要である．他方，近年の組み込みシステムは高機能化に伴って肥大化した既存のソフトウェア資産を有している．さらに長年の差分開発の繰り返しの結果，ソフトウェア・アーキテクチャーは複雑化している．この肥大化・複雑化した既存のソフトウェアを解析し，有効に活用することが効果的なアセット・ベース開発を実施する上で重要である．しかしシングル・プロセッサーを前提に構築された既存ソフトウェアを並列処理に適したアーキテクチャーに変更し，システム性能を向上させることは既存ソフトウェアの解析だけで実現できないことが課題となっている．

また，アーキテクチャー開発において，構築したシステム・アーキテクチャーが性能要件を満たすことを，開発の初期段階で確認することは，製品開発におけるリスクを大幅に軽減する．詳細設計・実装などの工程が開始された後，性能要件が満たされないことが判明した場合，手戻りに要する時間と開発コストの増加による製品開発への影響は甚大なものとなる．これらのリスクを回避・軽減するために開発の初期段階においてシステム・レベルでのパフォーマンスのシミュレーションが実施できることは極めて有効である．既存

提出日:2009年11月19日　再提出日:2009年12月10日

のシミュレーションによる性能の評価手法としては，SystemCに代表されるHDL（Hardware Description Language）を用いた手法［1］［2］［3］，独自のモデル記述言語で作成されたモデルを用いるシミュレーション手法などがある［4］［5］．詳細化した設計や実装を開始する以前に，モデルによって高位レベルでシステムを抽象化して記述，性能シミュレーションの結果を反映してアーキテクチャーを確定することは，有効な手法である．

筆者らは，そのモデル記述にOMG（Object Management Group）［6］の提唱する統一モデリング言語（Unified Modeling Language：以下，UML）を採用した．UMLはエンタープライズ・システムの開発においてソフトウェア・アーキテクチャー設計に広く使われるモデリング言語である．

UMLを使用することで，システム・アーキテクチャーを設計の各段階に必要な粒度で抽象化し，その構成を可視化することが可能となり，さらにシステム・アーキテクチャーの構成変更も容易となる．またUMLはオブジェクト指向言語に基づいた記述方法であるため，システム設計・実装の段階においてモデルを活用できる．

一般的に，モデル駆動型開発は図1に示すようなV字開発プロセスで示される．この開発プロセスでは要件定義から受け入れテストまでが網羅されているが，既存システムのソフトウェア資産を有効に活用することに対してのプロセスは定義されていない．筆者らは，図2に示すアセット・ベース開発の1つの手法を加えたモデル駆動型開発手法を定義した．

技術文書である仕様書などのドキュメント解析，ソースコードの静的解析，実機処理の実行トレースによる動的振舞解析の3つのリバース・モデリング技術によって，既存ソフトウェアをアセット化するプロセスを一般的なV字プロセスに追加した．機能要求の段階において新規の要求文書や要求元へのインタビューによる要件定義の結果と，リバース・モデリングの結果に基づいた要求モデルとしての振舞モデルを作成することで新規システムの振舞を高い精度で再現することが可能になる．そのモデルを用いてモデルベース・シミュレーションを実行，シミュレーション結果をモデルに反映することで，高位レベルでの機能とパフォーマンスの評価を実現する．これらの技術群，すなわち既存技術からモデル駆動開発への橋渡し技術であるリバース・モデリング技術と，設計の初期段階のアーキテクチャー設計においてシステム・レベルでのモデルによる性能解析を行う手法をSmarter Engineeringの1つとして位置付けている．前述のモデル駆動型開発プロセスにおいて，UMLにより記述した振舞モデルに新規開発した性能評価モジュールを組み込むことで，シミュレーションに用いるモデル作成の省力化を実現した．

本論文では開発プロセスの初期段階においてモデルにより効果的なパフォーマンス・シミュレーションと評価を行い，評価結果をシステム・アーキテクチャー設計に反映する手法について述べる．

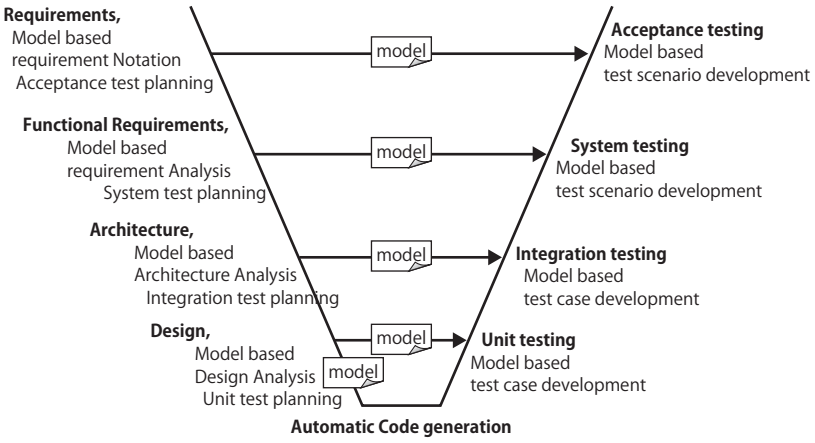

図1. 一般的なモデル駆動型開発のV字プロセス

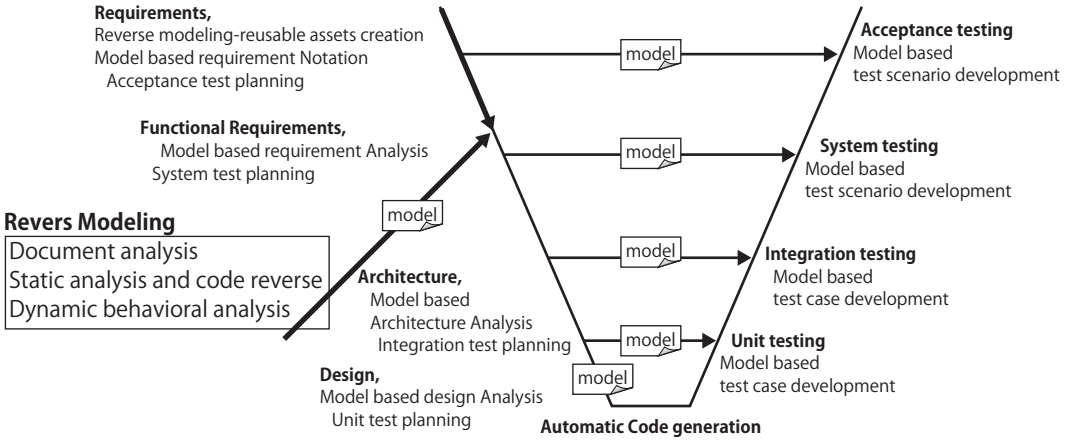

図2. われわれの提案するモデル駆動型開発プロセス

2. パフォーマンス・シミュレーション

2.1 パフォーマンスの評価対象

マルチコア・アーキテクチャーを採用する目的は，MFPの印刷処理の性能を向上することである．マルチコア化による効果の検証は，印刷性能の向上により短縮される処理時間をモデル・シミュレーションによって予測，評価する．また，AMP（Asymmetric Multiprocessing：非対称型マルチプロセッサー）構

成のマルチコア・アーキテクチャーではそれぞれのプロセッサー・コアに特定の OS とプロセスを割り当てるため，メモリー使用量はシングル・プロセッサー構成に比べて増加する傾向にある．組み込みシステムに搭載するメモリーは，製品コストを支配する主要コンポーネントの 1 つであるとともにシステムの性能も左右する．システム性能の低下を招くことなくシステム全体のメモリー使用量を抑制することは，アーキテクチャー設計において極めて重要である．印刷性能とメモリー使用量がパフォーマンスの評価の対象である．

2.2 モデルの構成と特徴

UML で記述したシステム・アーキテクチャーの振舞モデルは，システムの機能，すなわち振舞をモデル化したものであり，ソフトウェア開発における詳細設計に活用できる．一方，パフォーマンス・シミュレーションに用いるモデルは，振舞モデルに加えて評価の対象となる処理時間を取り扱うことが必要である．さらにメモリー使用量において，メモリー・リソースの取得・開放の動作と，それらに伴って変化する使用メモリーのサイズを管理する機能が必要となる．

そこでリバース・モデリング結果を用いて既存システムのアーキテクチャーを現す振舞モデル（図 3）を作成した．処理時間を管理するスケジューラー機能，メモリー使用量を管理するメモリー・リソース管理機能を振舞モデルに組み込む．

振舞モデルから，振舞単位の処理時間データをスケジューラー・モジュール（2.2.1 に後述）に，メモリー取得・開放情報をメモリー・リソース・モジュール（2.2.2 に後述）にそれぞれ通知することでシステム全体のパフォーマンス・シミュレーションを実現する．ここで，スケジューラー・モジュールとメモリー・リソース・モジュールを総じて性能評価モジュールと呼ぶ．この性能評価モジュールを外部モジュールとして独立させ，さらに性能評価のためのモデリング用にライブラリー化した．これによりアーキテクチャー開発者は，モデル作成において性能評価モジュールをモデルごとに個別に作成する必要がなく，アーキテクチャーの振舞モデルの検討作業に注力することが可能である．シミュレーション環境には Rational の Rhapsody［7］のモデル実行機能に加えて，

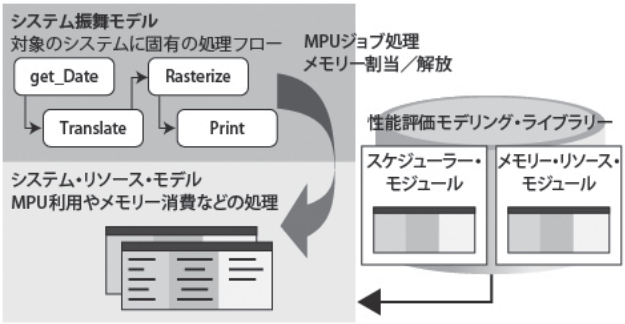

図3. アーキテクチャーの振舞モデル

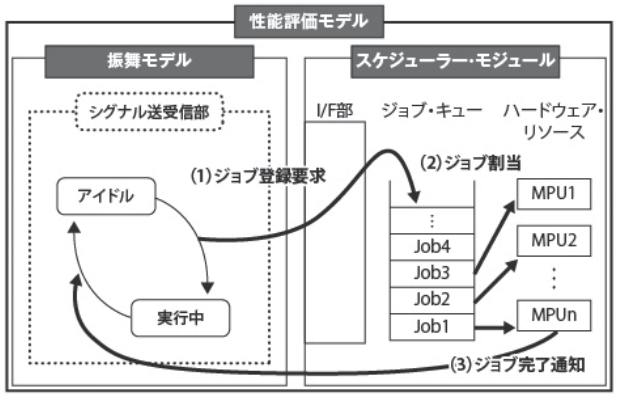

図4. スケジューラー・モジュール構成

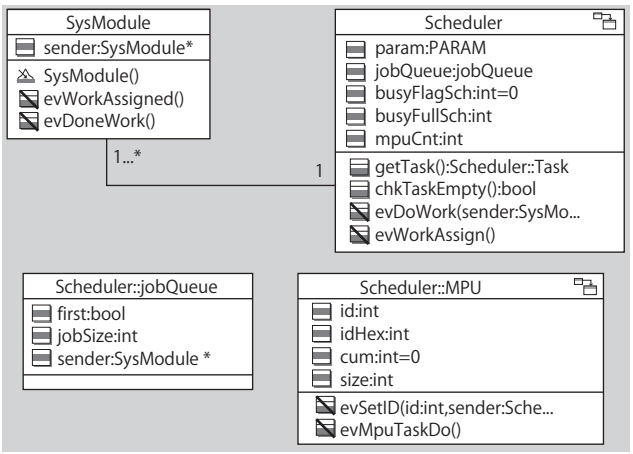

図5. スケジューラー・モジュール・クラス図

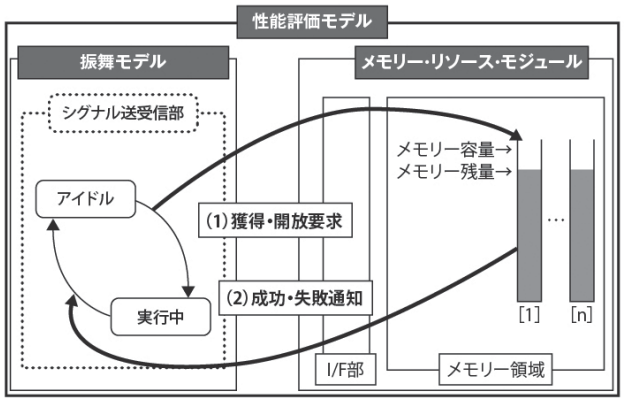

図6. メモリー・リソース・モジュール構成

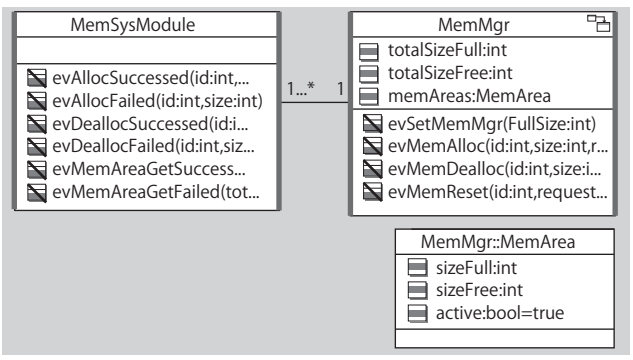

図7. メモリー・リソース・モジュール・クラス図

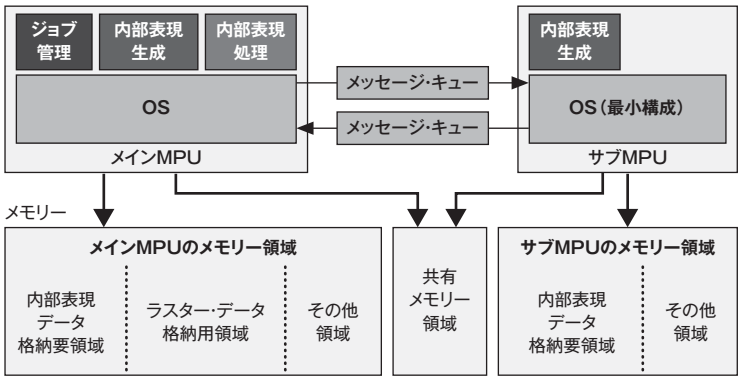

図8. マルチコア・モデル構成

マルチコア・アーキテクチャーにおいて並列処理時間をシミュレーションするタイマー機能を新規開発した.

2.2.1 スケジューラー・モジュールの構成と動作

スケジューラー・モジュール（図 4）はシステムの時間経過を表現するモデルである. スケジューラー・モジュールは振舞モデルから受け取った処理時間データを登録するジョブ・キュー, MPU などの処理を行うハードウェア・リソース, 振舞モデルに対するインターフェース部の 3 つのモデルで構成する. 振舞モデルには時間経過の表現に加えスケジューラー・モジュールへの処理時間データの登録要求を振舞モデルに送信, スケジューラー・モジュールからの処理完了を受信するシグナル送受信部モデルを追加する. スケジューラー・モジュールは振舞モデルから処理時間データをジョブとして受け取りジョブ・キュー・モデルに登録する. 次に登録されたジョブをモジュール内の 1 個もしくは複数個のハードウェア・リソース・モデルへ割り当てを行うことで処理時間データの経過をシミュレーションする. 処理完了後, 振舞モデル側に通知を行うことで振舞モデルは次の処理ステップに移行する.

スケジューラー・モジュールの UML 表現（図 5）においてスケジューラー本体を表す Scheduler クラスは, 内部クラスとしてジョブ・キューを表す JobQueue クラス, ならびにハードウェア・リソースを表す MPU クラスを持つ.

振舞モデルに対するインターフェース部は振舞モデルの基底クラスとして SysModule を定義し, 振舞モデル・クラスからの継承によりスケジューラー・モジュールと振舞モデルの間でシグナル送受信を実現する. 振舞モデルに複数のスケジューラー・モジュールを組み込むことで, 処理に応じて異なるプロセッサーを割り当てる AMP 構成を表現することを可能とした. さらにスケジューラー・モジュール内のプロセッサー並列数により SMP（Symmetric Multiprocessing : 対称型マルチプロセッサー）構成の並列度を表現することも可能である.

2.2.2 メモリー・リソース・モジュールの構成と動作

メモリー・リソース・モジュール（図 6）は, システムの機能ごとに割り当てられるメモリー・リソースの使用量を表現するモデルである. 同モジュールは機能ごとに割り当てる個別のメモリー領域と, 振舞モデルに対するインターフェース部の 2 つのモデルで構成する. 振舞モデルには, メモリー獲得・開放を表現する個所にメモリー・リソース・モジュールへのメモリー獲得・解放要求の送信と処理完了を受信する, シグナル送受信部を追加する. メモリー・リソース・モジュールは, 受け取ったメモリー獲得・開放要求に基づいて割り当てられたメモリー領域の残量からその要求の可否を判断し, 要求の成功・失敗を振舞モデルに通知する. 振舞モデルは, 受信した成功・失敗通知に応じて次の処理ステップに移行する. メモリー・リソース・モジュールの UML 表現を図 7 に示す. メモリー・リソース本体を表す MemMgr クラスは, 内部クラスとしてメモリー領域を表す MemArea クラスを持つ.

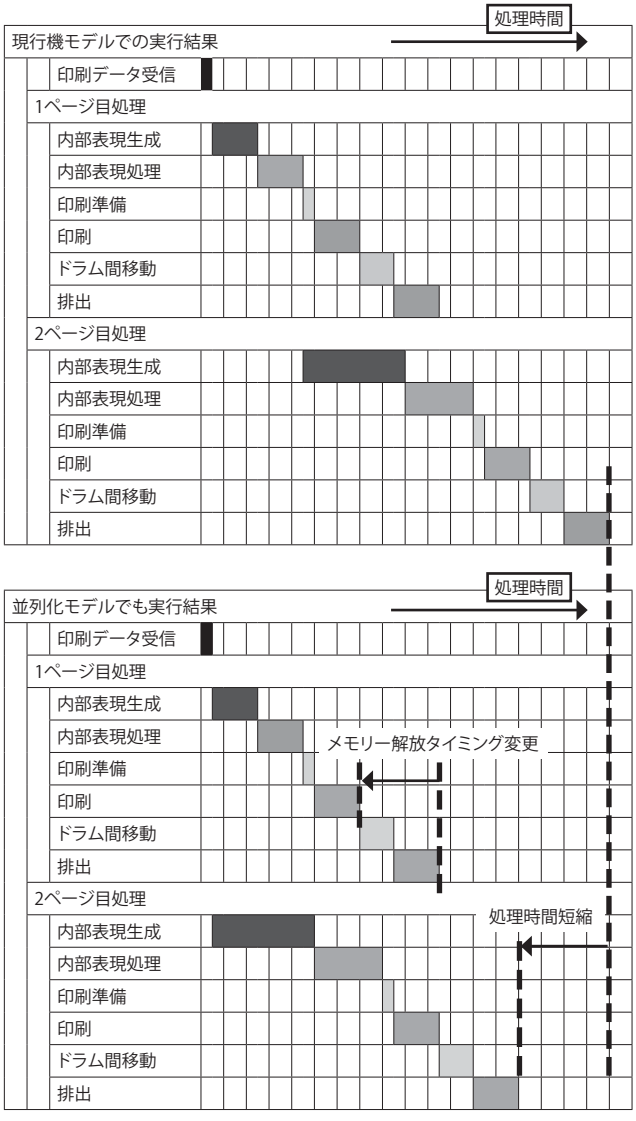

図9. プリント処理タイミング・チャート

振舞モデルに対するインターフェース部は,スケジューラー・モジュールと同様に基底クラス MemSysModule を定義し,振舞モデル・クラスからの継承によりメモリー・リソース・モジュールとの間でシグナル送受信が可能となる.

スケジューラー・モジュールによる印刷性能のシミュレーションと同時にメモリー使用量をシミュレーションすることで,振舞に対応したメモリー使用量の推移と最大値の評価を可能とした.

2.3 MFPのモデル化

MFP のモデル化にあたり対象となる動作は連続印刷である.まず,文書解析や動的解析によるリバース・モデリングの結果から現行機の振舞モデルを作成した.静的解析の結果,プリント処理は以下の処理で構成されることを確認した.

ホスト PC から受信した Post Script などの印刷記述言語データはプリンター内部のデータ処理に適した表現に変換される(内部表現生成).

その後,生成した内部表現をラスタライズ処理により画像データに変換(内部表現処理)し,印刷エンジンに送信する.印刷エンジンではスタートアップ処理である印刷準備,印刷,ドラム間移動,排出の順に処理が実行される.このほかに,これらのプリント処理の複数ジョブや,コピー/ Fax などの MFP のジョブ管理機能を行う処理がある.

次にリバース・モデリング手法である現行機の動的振舞解析とボトルネック解析を実施した.上記解析の結果から,内部表現処理にシステム・パフォーマンスのボトルネックがあり,かつ並列化により性能向上が見込める処理が存在することが判明した.そこで,このボトルネックを解消するために対象処理を並列処理する新規アーキテクチャーの振舞モデルを作成した(図 8).

新規アーキテクチャーとして AMP を採用する.振舞モデルにおいて,プロセッサーやハードウェア・アクセラレーターなど,ハードウェア・リソースでの処理を表現する個所にスケジューラー・モジュールを組み込む.内部表現生成をページ単位で分割しそれぞれの MPU に割り当てる.ジョブ管理と奇数ページの内部表現生成ならびに内部表現処理は MPU1 に,偶数ページの内部表現生成と内部表現処理を MPU2 に割り当てる.さらに既存の 4 個のハードウェア・アクセラレーターを表現するため,合計 6 個のスケジューラー・モジュールを使用する.またシステム・メモリーとして 1 個のメモリー・リソース・モジュールを使用し,各処理に割り当てたメモリー領域の使用状況を再現する.次に動的振舞解析によって取得した現行機の各処理に要する処理時間,メモリー取得と解放動作の測定データからスケジューラー・モジュール,メモリー・リソース・モジュールに設定する性能データを作成し,振舞モデルから同データを設定した.

2.4 性能評価モデルの作成

プロセッサー数 2 台の AMP 構成を適用した MFP のアーキテクチャー・モデルを作成した.パフォーマンス・シミュレーションに用いたデータは電子情報技術協会(JEITA)が定める印刷性能の評価用画像データである.またパフォーマンス・シミュレーションにおける MFP の動作は,上記の評価用画像データの連続印刷である.内部表現生成は MPU1, MPU2 それぞれに奇数ページと偶数ページを割り当てる(図 8).メイン・プロセッサーである MPU1 で動作するソフトウェアは内部表現生成が奇数ページだけであることを除いて現行システムと共通である.サブ・プロセッサーである MPU2 では最小構成の OS と偶数ページの内部表現生成が動作する.このアーキテクチャーを採用することにより,既存システムのソフトウェアからマルチコア・アーキテクチャーへの変更は内部表現の生成に限定されることになり変更規模は小規模である.

2.5 パフォーマンス・シミュレーションの実施

前述の環境におけるモデル・シミュレーションによって処理時間とメモリー使用量を評価するためのパフォーマンス・シミュレーションを実施した(図 9).シングル・プロセッサー構成を採用する現行機モデルでは,1 ページ目の内部表現処理が完了した後に 2 ページ目の内部表現生成を開始する.マルチコア・アーキテクチャーを採用した並列化モデルでは

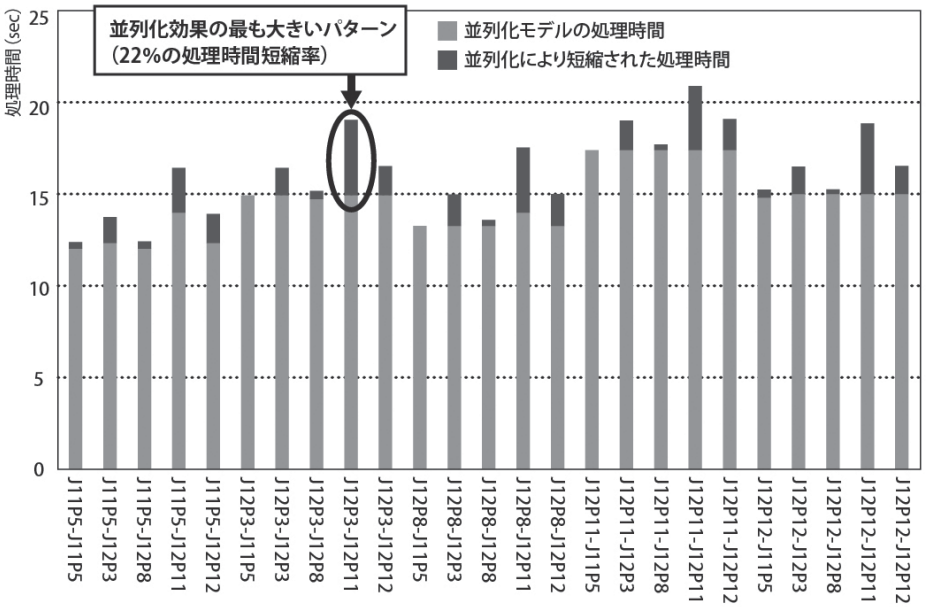

図10. 2ページ連続印刷時の並列化効果

このタイミングを変更し，1 ページ目の内部表現生成と同時に 2 ページ目の内部表現生成を開始する．この並列化の効果により，2 ページ目の排出完了までの処理時間の短縮が可能となる．印刷準備が終了した以降の処理は MFP のプリンター部分の機械的な動作に制約を受けるため，マルチコア化による処理時間短縮は望めないと判断した．

評価用画像データから内部表現生成と処理時間比率の異なる 5 種類のデータを組み合わせて合計 25 パターンで 2 ページ連続印刷のモデル・シミュレーションを実施した（図 10）．

このシミュレーションにおける処理時間の評価結果により，最大 22.0%，平均 8.3% の処理時間短縮が達成でき，マルチコア化による印刷性能の向上を確認した．

メモリー使用量においては A3 用紙サイズの連続印刷時にラスター・データ格納用メモリー領域の不足，すなわち印刷中にメモリー不足によるストールが発生することをシミュレーションによって確認した（図 11）．

原因は内部表現生成を並列化したことによりラスター・データを生成するための作業領域の確保によるメモリー使用量の増加である．

ラスター・データの生成におけるメモリー使用量は用紙の大きさと連続印刷の枚数に比例して増加するが，図 11 に示すように連続して 10 ページの印刷を行った場合，最初と最後のページを除いて計 8 回のメモリーストールが発生する．

この問題に対してメモリー領域の開放タイミングを現行の排出（用紙排出処理完了）後から印刷処理完了後へ変更することによって（図 9），メモリー使用量のピークはメモリー容量全体の 94%となりストールは解消される（図 12）．

この変更によって現行システムに搭載されるメモリー・サイズを拡大することなく並列処理を実現することが可能であることをシミュレーションで確認した．

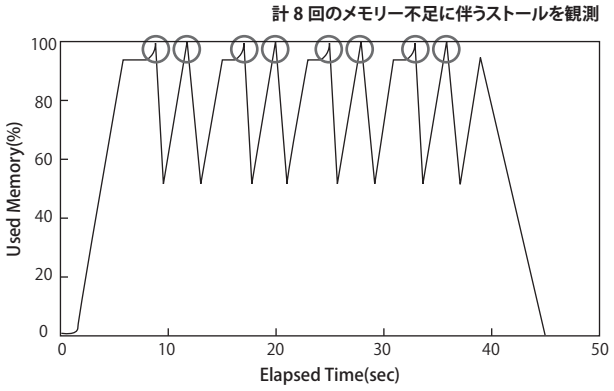

図11. メモリー使用量の遷移

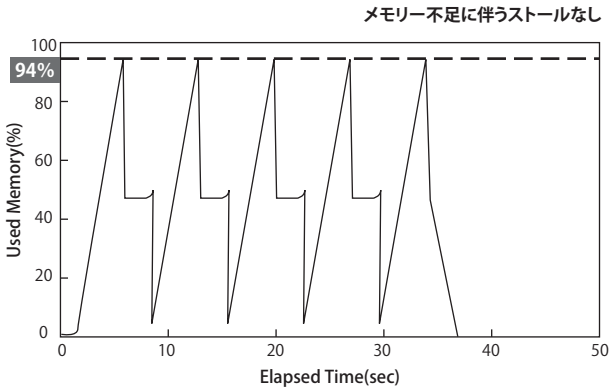

図12. タイミング変更後 メモリー使用量遷移

3. 結論

システム・アーキテクチャー設計の初期段階において，システムを UML モデルで作成し，それを用いたモデル・シミュレーションにより，性能の評価を実現し，マルチコア構成を採用することによる性能向上の有効性を示すことができた．さらにメモリー・リソースのモデル化とモデル・シミュレーションにより，マルチコア構成の採用に伴うメモリー使用量の評価を可能としシステムに搭載するメモリー容量を現行機から増加することなく性能向上が実現できることを示した．以上のように本論文で提案する手法は，アーキテクチャーの変更に伴うシステム性能の検討に有効と考える．

今回は，アーキテクチャー開発時の制約として既存ソフトウェアの変更量を抑制するため，内部表現生成，内部表現処理の一部を MPU2 に振り分ける AMP 構成の検討に限ってシミュレーションを行ったが，UML で記述したスケジューラー・モジュールでは，1 個のモジュール内のプロセッサー多重度を 2 以上に設定することで，SMP 構成のシミュレーションを行うことが可能である．

今後の課題として，システムの性能を大きく左右するコミュニケーション，すなわちバス接続や通信の帯域幅のモデル化とシミュレーションによる評価が重要であると考える．

謝辞

今回の性能評価のデータ測定にご協力いただいたお客様，MFP ソフトウェア開発研究部の皆様，ならびに，助言いただいたプロジェクト・メンバーの皆様に感謝いたします．

参考文献

［1］ T. Kempf, K. Karuri, S. Wallentowitz, G. Ascheid,R. Leupers, and H. Meyr: "A SW performance estimation framework for early system-level-design using fine-grained instrumentation" Design, Automation and Test in Europe（DATE 2006）, pages 468-473, Munich, Germany,（March 2006）.

［2］ H. Posadas, F. Herrera, P. Sanchez, E. Villar, and F. Blasco: "System-level performance analysis in SystemC" Design, Automation, and Test in Europe（DATE 2004）, pages 378-383, Paris, France,（February 2004）.

［3］ V. Reyes, W. Kruijtzer, T. Bautista, G. Alkadi, and A. Nunez: "A unified system-level modeling and simulation environment for MPSoC design: MPEG-4 decoder case study" Design, Automation

and Test in Europe（DATE2006）, pages 474-479, Munich, Germany,（March 2006）.
［4］ G. Madl, N. Dutt, and S. Abdelwahed: "Performance estimation of distributed real-time embedded systems by discrete event simulations" ACM Conference on Embedded Systems Software（EMSOFT2007）, pages 183-192, Salzburg, Austria,（October 2007）.
［5］ J. M. Paul, D. E. Thomas, and A. S. Cassidy: "High-level modeling and simulation of single-chip programmable heterogeneous multiprocessors" ACM Transactions on Design Automation of Electronic Systems（TODAES）, 10(3):431-461,（July 2005）.
［6］ Object Management Group, http://www.omg.org/
［7］ Rational Rhapsody, http://www-06.ibm.com/software/jp/rational/products/rhapsody/productline/

日本アイ・ビー・エム株式会社
グローバルビジネスサービス
組み込みソフトウェア・コンサルティング
PMP，ICP シニア・プロジェクトマネジャー
坂本 佳史 Yoshifumi Sakamoto

［プロフィール］
1985 年日本 IBM 入社．論理回路設計，システム設計・開発などに従事．その後，プロジェクト・マネジャーとして主に組み込み機器の開発プロジェクトに従事．

日本アイ・ビー・エム株式会社
マイクロエレクトロニクス事業
スタッフ R&D エンジニア
豊田 学 Manabu Toyota

［プロフィール］
1995年，日本IBM入社．主に製造業のお客様向けハードウェア開発サービスに従事．

日本アイ・ビー・エム株式会社
東京基礎研究所
アドバイザリー・リサーチャー
小野 康一 Kouichi Ono

［プロフィール］
1994 年より日本 IBM 東京基礎研究所に所属．テクニカルマスター．モデル駆動開発技術，ソフトウェア検証技術，移動エージェント技術，Web アプリケーション開発支援技術，プログラム解析技術などの研究に従事．情報処理学会，日本ソフトウェア科学会，IEEE-CS，ACM 各会員．

日本アイ・ビー・エム株式会社
東京基礎研究所 リサーチャー
河原 亮 Ryo Kawahara

［プロフィール］
2007 年，日本 IBM 入社．組み込みシステムを対象としたソフトウェア工学分野の研究開発に従事．現在は東京基礎研究所にて UML による性能評価手法の研究に従事．

日本アイ・ビー・エム株式会社
マイクロエレクトロニクス事業
ソフトウェア・エンジニア
長井 真吾 Shingo Nagai

［プロフィール］
2005 年，日本 IBM 入社．ソフトウェア・エンジニアとして，Cell/B.E. 搭載ブレード・サーバーによるシステム開発業務に従事．2008 年より，組み込みシステムのシミュレーション技術の研究開発業務に従事．

日本アイ・ビー・エム株式会社
東京基礎研究所／次世代組込み技術
アドバイザリー・リサーチャー
中田 武男 Takeo Nakada

［プロフィール］
1986 年，日本 IBM 入社．東京基礎研究所にて並列処理システム・アーキテクチャー，並列処理アルゴリズム，マルチプロセッサー用キャッシュの研究のほか，ThinkPad 省電力設計，ASIC/SoC 設計・検証の技術開発に従事．IEEE，電子情報通信学会，情報処理学会各会員．