4F-09 オペレーティングシステムの並行開発手法

4F－9

平山 秀昭[1]　大森 誉史[1]　山本 賢司[2]　鈴木 和子[3]

[1](株) 東芝　情報・通信システム技術研究所
[2]東芝システムクリエイタ株式会社 システム技術部
[3]東芝情報システム株式会社 ソフトウェア技術本部

1　はじめに

従来のソフトウェア開発はウォータフォール型に行なわれていた。しかし今日コンカレントエンジニアリング技術のソフトウェア開発への適用が研究され、コンカレント型のソフトウェア開発が注目されている [1][2][3]。新規に計算機システムを開発する場合、オペレーティングシステムのデバッグはハードウェアの試作が終ってからでないと行なえず、典型的なウォータフォール型になっていた。すなわち計算機システムの方式設計、ハードウェア開発、オペレーティングシステム開発という作業は直列化されてしまうというのが現状であった。この問題を解決するためにハードウェア動作のシミュレータを開発し、その上でオペレーティングシステムを開発するという手法が提案されている [4]。しかしシミュレータがハードウェアの動作を正確に模擬し、かつ実行速度が十分に速くなければ、この様な手法というのは全く役に立たない。今回我々は実際にハードウェア動作のシミュレータを開発し、その上に UNIX[1] を移植してみた。本論文ではシミュレータを用いたオペレーティングシステムの並行開発手法の有効性について示す。

2　ハードウェア動作のシミュレーション

ハードウェアをシミュレートするには、その目的に合わせて適切なシミュレーションレベルを選ぶ必要がある。レベルが粗すぎると目的とした機能が評価できないし、逆に細かすぎると実行時間がかかりすぎる。オペレーティングシステムのデバッグを目的とした場合には、コードそのものが実行できるよう、命令レベルでのシミュレーションが必要となる。それより粗いレベルでは、コードそのものを実行することはできず、デバッグという目的には適さない。逆にパイプライン動作等をも含めた、より細かいレベルのシミュレーションでは、不必要に実行時間が増加してしまう。

またシミュレーションの範囲としては、プロセッサ、MMU、キャッシュ(プロセッサ内部および外部)、メモリ、制御回路、I/O 装置等、様々なものが存在するが、デバッグしたいコードがどこまで及ぶのかによって、適切な範囲を選ぶ必要がある。今回我々はプロセッサ、MMU、外部キャッシュ、制御回路、I/O 装置をシミュレートした。シミュレータの構成を図 1 に示す。内部キャッシュの実装は実行速度を大きく低下させる上に、それによってデバッグが可能となるコードが少ないため対象から外した。また制御回路はソフトウェアから見える機能のみを実装した。

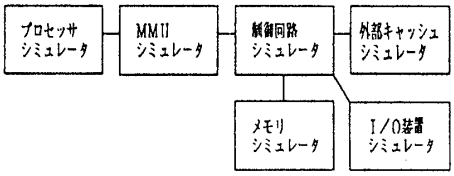

図 1: シミュレータの構成

3　オペレーティングシステムの並行開発

計算機システムを開発するためには、まず方式設計を終らせる必要がある。方式設計完了後は、ハードウェアとオペレーティングシステムの開発を行なう。後者の開発では、コーディングはハードウェア開発と並行して行なえるが、デバッグはその開発完了まで待たされる。特に UNIX 等の既存システムを移植する場合、その設計およびコーディング期間は比較的短時間であり、ハードウェアを使った実機上でのデバッグまで、かなりの時間待たされることがある。

今回我々はハードウェアの開発と並行してオペレーティングシステムのデバッグを行なうことを目的に、ハードウェア動作をソフトウェアから見えるレベルで模擬するシミュレータを開発した。パイプライン等の細かな動作はシミュレートしておらず、開発期間は実機開発に比べて短期間であった。このシミュレータを用いハードウェアの開発と並行して、プログラムのロード/デバッグが行なえる簡単なハードウェアモニタ、ブートプログラム、オペレーティングシステム、オペレーティングシステムデバッガの開発を行なった。その手順を以下に、並行開発の流れを図 2 に示す。

ハードウェアが完成すると、まずハードウェアの正常動作を確認するために、多数のテストプログラムを実行しなければならない。ここでもしテストプログラムが正常に動作しなかった場合、その原因がハードウェアにあるのか、プログラムにあるのかを判定するのに時間を要する。しかし事前にシミュレータ上で動作確認が行なわれていれば、判定が容易になる。またテストプログラムを実行するためには、テストプログラムのロード/デバッグを行なうためのハードウェアモニタが必要となるが、その開発もシミュレータ上で行なった。

ハードウェアの正常動作が確認されると、いよいよオペレーティングシステムのデバッグに移るわけだが、そのためにはオペレーティングシステムを実機上にロードするブートプログラムが必要となる。オペレーティングシステムがロードされ実行ができるようになると、そのデバッグを行なうためのオペレーティングシステムデバッガが必要となる。オペレーティングシステムを開発するために必要なこれら全てのプログラムと、オペレーティングシステム自身を事前にシミュレータ上でデバッグしておいた。ハードウェア完成後は既にシミュレータ上で動作することが確認されているプログラムを移すだけなので、ハードウェア

Concurrent Engineering for Operating System Development
Hideaki HIRAYAMA,Shigefumi OHMORI:
Information & Communication Systems Laboratory
TOSHIBA Corporation
Kenji YAMAMOTO:
TOSHIBA System Creator Corporation
Kazuko SUZUKI:
TOSHIBA Information Systems (Japan) Corporation

[1]UNIX オペレーティングシステムは、UNIX System Laboratories, Inc. が開発し、ライセンスしています。

完成後のソフトウェアのデバッグ期間を大幅に短縮することが期待できる。

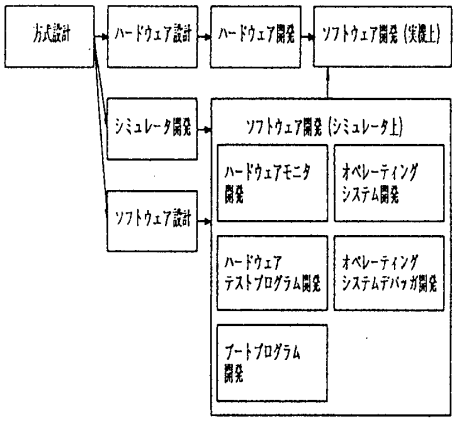

図 2: 並行開発の流れ

4 評価

今回実際に開発したハードウェア動作のシミュレータ上に UNIX を移植してみた。オペレーティングシステムのデバッグにこの種のシミュレータが利用できるかどうかは、主に実行速度に依存している。シミュレータの有効性を検証するために、UNIX の初期化処理の各フェーズで実行される命令数と所要時間を測定した。結果を表 1 に示す。なお実行は SPARCstation[2][3] 10 Model 51(プロセッサは 40MHz の SuperSPARC[4]、1MB の外部キャッシュと 128MB のメモリを実装) にて行なった。

表 1: UNIX の初期化処理に必要な命令数と実行時間

	命令数 [K 命令]	実行時間 [分]
カーネルメモリアロケータの初期化完了まで	3,436	4
MMU 管理処理の初期化完了まで	10,334	14
ルートファイルシステムのマウント完了まで	56,832	78

一般にハードウェア完成後の UNIX のデバッグでは、初期化処理に多くの時間を要する。表 1 によると 1 時間半程度で UNIX の初期化処理が完了し、ハードウェア完成前に UNIX の初期化処理のデバッグを終えることができ、この様な手法が有効であることが確認された。

またシミュレータ上でプログラムを実行する場合、実際のハードウェア上でプログラムを実行するのに比べて、どの程度の速度差になるのかを評価するために、幾つかのベンチマークプログラムを、SPARCstation 10 上、および SPARC station 10 上に実装されたシミュレータ上で実行し、処理時間を測定してみた。結果を表 2 に示す。

qsort はクイックソート、sprintf は int 型変数から 8 進、10 進、16 進データへのフォーマット変換、cal はカレンダの作成 (ただし実際の表示は行なわない) を行なうプログラムで、SPARCstation 10 上での実行時間を 1 とした場合の、シミュレータ上での相対的な実行時間を示している。

[2]SPARC は、米国における米国 SPARC International, Inc. の登録商標です。SPARC 商標の付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

[3]SPARCstation は、米国における米国 SPARC International, Inc. の登録商標であり、米国 Sun Microsystems 社が独占的に使用許諾を受けている商標です。

[4]SuperSPARC は、SPARC International, Inc. より Texas Instruments 社が使用許諾を受けている商標です。

表 2: SPARCstation 10 と SPARCstation 10 上に実装されたシミュレータの実行速度の比較

	SPARCstation 10	SPARCstation 10 上に実装されたシミュレータ
qsort	1	778
sprintf	1	2,800
cal	1	2,922

表 2 によると本シミュレータの実行速度は、実機の実行速度の 3000 倍程度の遅さである。すなわち実機では 1 秒で行なわれる処理を、本シミュレータでは 50 分程度で行なうことになる。この実行時間の大半は、プロセッサ、MMU、外部キャッシュの動作を模擬する部分で占められていて、それらの動作を考えると、ほぼ妥当な実行速度であると考えられる。

シミュレータの実行速度は UNIX の初期化処理を行なうために十分なものであり、実際に我々はハードウェアの完成前にハードウェアモニタ、ハードウェアテストプログラム、ブートプログラム、オペレーティングシステム、オペレーティングシステムデバッガの開発を行なうことができた。これによってハードウェア完成後のソフトウェアのデバッグ期間を大幅に短縮することが期待できる。

シミュレータによってハードウェア動作が正確に模擬されているかという点に関しては、UNIX の初期化処理が正常に行なえたということから問題がないと言える。制御回路を操作するコードのデバッグでは、シミュレータ開発者とオペレーティングシステム開発者の間で、ハードウェア動作の理解が異ることがあったが、このような場合はハードウェア開発者と議論を行なうことにより、誤った理解を訂正することができた。

またシミュレータは実機に比べて、外部キャッシュや制御回路の観測が容易であり、オペレーティングシステムの開発者が誤ったコーディングを行なった場合に、実機に比べて容易に問題を判定することができた。

5 おわりに

従来オペレーティングシステムの開発は、一般のソフトウェアの開発に比べて旧式な手法を採っていた。この主な原因はオペレーティングシステムがハードウェアに強く依存しているという特殊性にあった。しかしこの様な開発手法で十分な訳ではなく、可能な限り改善していくべきものである。この論文ではハードウェアの動作を模擬するシミュレータを利用することにより、従来ウォータフォール型であったオペレーティングシステムの開発を、コンカレント型に行なう手法の有効性について示した。

参考文献

[1] Y.V. Ramana Reddy, Kanakanahalli Srinivas, V. Jagannathan, and Raghu Karinthi, "Computer Support for Concurrent Engineering", *IEEE COMPUTER*, January 1993.

[2] Duvvuru Sriram, and Robert Logcher, "The MIT Dice Project", *IEEE COMPUTER*, January 1993.

[3] 村上憲稔, "ソフトウェアのライフサイクル管理", 情報処理 Vol. 33 No. 8, August 1992.

[4] 武内和明, 矢木孝幸, 森良哉, "OS 検証用の新コンピュータ・アーキテクチャ・シミュレータ", 情報処理学会第 45 回 (平成 4 年後期) 全国大会.