QEMUによるHW/SW協調シミュレータの構築

高見澤 秀久† 位野木 万里† 川田 秀司‡ 石井 忠俊§ 吉川 寿広§ 荒木 大§
東芝ソリューション株式会社† 株式会社東芝 ソフトウェア技術センター‡
株式会社インターデザイン・テクノロジー§

1. はじめに

1.1. 背景

近年、組込みソフトウェア開発に、ターゲットとなる組込み機器をモデル化したシミュレータを構築し適用する取り組みが盛んである。QEMU[1]は、高速なCPUシミュレータであり、携帯端末向けのソフトウェア・スタックであるAndroidのシミュレータのベースになっていることでも知られている。さらに、QEMUは、計算機の仮想化ツールであるXen®やSun™ xVM VirtualBox™でメカニズムの一部が使用されたり、クラウド・コンピューティングを利用した、並列分散システムのソフトウェア・テスト環境であるD-クラウド[2]で使用されたりと、益々の普及が期待される。

1.2. 課題

我々は、ソフトウェア開発環境としてQEMU-0.9.1を普及させるにあたり、次のような課題に直面した。

【課題1】ターゲット機器内の周辺装置が定期的に動作する場合、そのシミュレーション用のハードウェア・モデル（HWモデル）の実行間隔として、ホストPC上でのQEMUの実行時間を使用している。しかしながら、ホストPCの負荷状況は一定ではなく、QEMUの実行時間に再現性はない。そのため、定期的に実行されるHWモデルの実行順序が、シミュレーションの実行ごとに異なってしまう可能性があり、シミュレーション実行中に発生した問題を再現できないといった問題が起こり得る。

【課題2】QEMUではCPUシミュレータとともに、多くのHWモデルが提供されている。しかしながら、HWモデルをカスタマイズするには、膨大な量のソースコードから構成されるQEMUの内部構造を理解する必要がある。そのため、ユーザがQEMUのHWモデルをカスタマイズすることは困難である。加えて、HWの並列性を考慮したHWモデルの開発も困難である。

1.3. 解決のアプローチ

我々は、上記の課題を解決するためQEMU-0.9.1を改造し、HW/SW協調シミュレータであるVisualSpec® for Embedded[3][4]と連携させる方法について検討してきた。VisualSpec® for Embeddedは、ホストPCの負荷状況に影響されないシミュレーション時間に基づいて、ターゲット・ソフトウェアのシミュレーション用のモデル（SWモデル）とHWモデルとを同期実行するための機構を備えている。これにより課題1を解決することができる。そして、SWモデルの内部構造の知識がなくても、VisualSpec® for EmbeddedのAPIや割込み通知用APIといった、SWモデルとHWモデルとを接続する汎用インタフェースに基づいて、HWモデルを開発することができる。さらに、VisualSpec® for EmbeddedのHWモデルエディタを使用することで、HWの並列動作を容易に記述することができる。これにより、課題2を解決することができる。

1.4. 本稿の構成

以下、本稿では、QEMU-0.9.1の動作について説明する。続いて、VisualSpec® for Embeddedと連携させるためのQEMUの改造方法を述べる。そして、QEMUとVisualSpec® for EmbeddedによるHW/SW協調シミュレータの実行性能を計測し、最後にまとめとする。

2. QEMU

QEMU-0.9.1では、ターゲットSWの動作をシミュレートするために、ターゲットCPU用に生成されたバイナリコードを読み込み、QEMU独自の中間コードに変換した後、ホストPC用のコードに変換する。この一連の処理はターゲットSWのリニア・ブロック単位で、シミュレーション実行時に動的に行われる。そのため、QEMUはターゲットSWを一命令ずつ解釈しながら実行する命令セット・シミュレータと比べて非常に高速である。ここで、ホストPC用に変換されたブロックは変換ブロック（Translation Block：TB）と呼ばれ、一度変換された変換ブロックは、QEMUが備えるキャッシュに保存される。そのため、繰り返し処理などにより再度その変換ブロックを実行する際の変換処理コストを削減することができ、更なる高速化を実現する。QEMUのHWモデルを実装するには、基本的には、ターゲットSWからのI/Oアクセスに対する処理をシミュレートする関数と、定期的に実行される処理をシミュレートする関数を用意する必要がある。図1にQEMUの動作例を示す。図1では、変換ブロックTB-2内で発生したI/Oアクセスにより、HWモデルのI/Oアクセス用の関数が呼び出されている。I/Oアクセス用関数の処理が終了すると、呼び出し元の変換ブロックTB-2の実行に制御を返す。

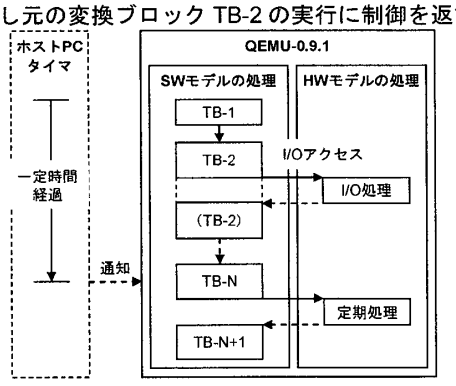

図1　QEMU-0.9.1の動作例

Construction of HW/SW Co-Simulator using QEMU
† Hidehisa TAKAMIZAWA, Mari INOKI,
Toshiba Solutions Corporation.
‡ Hideji KAWATA,
Toshiba Corporation, Corporate Software Engineering Center.
§ Tadatoshi ISHII, Toshihiro KIKKAWA, Dai ARAKI,
InterDesign Technologies, Inc.

一方、定期処理を実行する関数は、変換ブロックからは呼び出されず、一定時間が経過すると、実行中の変換ブロック TB-N の終了を待って処理が実行される。ここでの一定時間とは、前述の通り、ホスト PC 上でのシミュレータの実行時間であり、ターゲット機器上での処理時間を模擬するものではない。したがって、定期的に実行される関数の実行順序はシミュレーションの実行ごとに異なる可能性がある。

3. HW/SW 協調シミュレータの構築

VisualSpec® for Embedded の HW/SW モデル連携で使用する汎用インタフェースには、I/O アクセス用の関数、割込み通知用の関数、そしてシミュレーション時間を経過させる関数 waitfor 等が含まれる。I/O アクセス用の関数の実体は HW モデル内に、そして、割込み通知用の関数の実体は QEMU 内に、それぞれ実装する。関数 waitfor は、VisualSpec® for Embedded が提供するシミュレーション実行エンジン内に実装されている。関数 waitfor が実行されると、シミュレーション実行エンジンのスケジューリング機能により、次に実行すべきモデルを決定し、実行する。

そこで我々は、QEMU-0.9.1 を VisualSpec® for Embedded と連携させるために、QEMU に次の改造を行った。

(a) QEMU が生成する各変換ブロックの最後に、各変換ブロックを実行した際の遅延時間をエンジンに通知するための制御コードを追加する。また、制御コード内では、HW モデルで発生した割込みも検出する。

(b) QEMU で提供されている HW インタフェースを、変換ブロック内で発生した I/O アクセスを、VisualSpec® for Embedded の HW モデルに渡すよう、改造する。また、I/O アクセスの後にも、HW モデルで発生した割込みを検出する仕組みを実装する。QEMU に上記の改造を行い、VisualSpec® for Embedded と連携させた場合の動作例を図 2に示す。

図 2では、各変換ブロックに追加された制御コードで関数 waitfor を実行し、実行した変換ブロックの遅延時間をシミュレーション実行エンジンに通知している。実行エンジン内では、関数 waitfor で与えられた遅延時間に基づいてスケジューリングを行い、次に実行すべきモデルを決定している。また、HW モデルの定期処理の起動は、実行エンジンにより制御される。

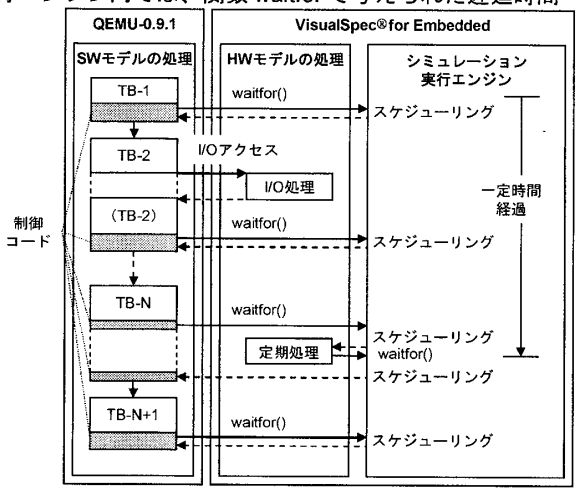

図 2 QEMU-0.9.1 と VisualSpec® for Embedded を連携させた場合の動作例

4. 実験

著者らは、3 章の(a)(b)の改造、および、シミュレーションの実行速度を向上させるためのチューニングを行った QEMU-0.9.1 と VisualSpec® for Embedded とを連携した協調シミュレータの、シミュレーションの実行速度を計測した。QEMU の改造箇所は、ホスト環境が x86 Windows、ターゲット環境が ARM に該当する箇所を対象とした。また、QEMU が備える HW モデルは、ターゲット機器の主記憶をシミュレートするためのモデル以外は一切使用せず、すべての I/O アクセスは、VisualSpec® for Embedded で用意した HW モデルに送るようにした。

QEMU 上では、TOPPERS/ASP カーネル[5]を動作させた。ASP カーネル上では、サンプルプログラムを改造し、3 つのタスクが、起動したらすぐに 10 ミリ秒間の待ち状態になる、という処理を繰り返し実行するプログラムを用意した。HW モデルは、簡易的なタイマモデル、割込みコントローラモデル、そして USART モデルを、VisualSpec® for Embedded で実装した。ターゲット CPU のクロック周期は 10 ナノ秒とした。

上記プログラムを、Dynabook SS Intel® Core™2 Duo CPU 2.53GHz、2.86GB RAM、Microsoft® Windows® XP Professional SP3 で実行したところ、実行性能は約 97MIPS であった。

また、著者らは、3 章の(a)(b)の改造によるオーバヘッドを計測するため、I/O アクセスおよび HW 割込みが発生しない簡単なプログラムを、3 章の(a)(b)の改造を行った QEMU-0.9.1 と、行っていない QEMU-0.9.1 とでそれぞれ実行し、実行性能を比較した。その結果、3 章の(a)(b)の改造によるオーバヘッドは、約 1.72 倍であり、妥当な結果であると考えられる。

5. まとめ

我々は、ソフトウェア開発環境として QEMU-0.9.1 を普及させる上での課題を解決するため、QEMU-0.9.1 と VisualSpec® for Embedded とを連携させた HW/SW 協調シミュレータを構築した。そして、我々が構築したシミュレータの実行性能を、実験により計測した。

今後は、このシミュレータの実行性能の妥当性を検証するとともに、従来は実機で行っていた試験を、このシミュレータによって、どの程度代替できるか検証する必要がある。

参考文献.

[1] Fabrice Bellard, “QEMU, a Fast and Portable Dynamic Translator”, USENIX 2005 Annual Technical Conference, FREENIX Track.

[2] 坂西隆之, 小泉仁志, 神林亮, 佐藤三久, “プログラムテスト環境を提供するクラウドコンピューティングシステムの検討”, SWoPP2009.

[3] VisualSpec® for Embedded http://www.interdesigntech.co.jp/modules/tinyd0/index.php?id=11

[4] Dai Araki, Hidehisa Takamizawa, “Co-simulation of Multi-ECU Hardware/Software System using Timing Back-annotation Method”, ISOCC 2009.

[5] TOPPERS プロジェクト http://www.toppers.jp/