Vol.2012-SLDM-155 No.27
Vol.2012-EMB-24 No.27
2012/3/3

抽象度を変更可能な命令セットシミュレータの提案

井田健太[†1]　坂主圭史[†1]
武内良典[†1]　今井正治[†1]

本研究では，シミュレーション精度をシミュレーション中に変更可能なプロセッサ・シミュレータを提案する．提案プロセッサシミュレータは，シミュレーション中に外部のハードウェアモデルより割り込みが要求されたときに，シミュレーション精度を命令精度からサイクル精度に変更することによりリアルタイムシステムで重要な割り込みの応答サイクル数を，シミュレーションの速度を落とすこと無く評価することができる．実験では，提案シミュレータの，外部のモデルからの割り込みに対する応答と，シミュレーション速度を計測することにより，本手法がシミュレーション速度を犠牲にせずに割り込みの応答サイクル数を計測できることを確認した．

Simulation Level Changeable ISS

Kenta IDA,[†1] Keishi SAKANUSHI,[†1]
Yoshinori TAKEUCHI[†1] and Masaharu IMAI[†1]

In this study, we present a processor simulator which can change simulation level at simulation time. The proposed simulator can evaluate number of cycles to respond an interrupt request , which is an important parameter in realtime systems, without losing simulation speed. This is established by changing simulation accuracy from instruction-accurate to cycle-accurate at simulation time. And we also measured simulation speed and response to interrupts by external hardware in order to examine our approach, which results our method can measure number of cycles to respond an interrupt request without sacrificing simulation speed.

†1 大阪大学
Osaka University

1. はじめに

マイクロプロセッサやマイクロコントローラなどのプロセッサを用いた組み込み機器が我々の生活に浸透し，使用されている．組み込み機器の市場の変化は速くなっており，そのようなプロセッサで動作するソフトウェアの開発においては，短い期間で品質の高いソフトウェアを開発することが求められている．これらの要求に答えるため，ソフトウェアのデバッグや性能評価がプロセッサのシミュレータを用いて一般的に行われる．ソフトウェアのデバッグや性能評価に用いられるシミュレータは，高速かつ高精度にプロセッサの動作のシミュレーションを行うことが求められている．プロセッサ単体でシステムが構成されることはなく，多くの場合はプロセッサとその他のデジタル回路やアナログ回路などの周辺回路を組み合わせてシステムが構成される．したがって，プロセッサのシミュレータには，プロセッサに接続される外部のハードウェアに対応するシミュレーションモデルに接続し，協調してシミュレーションを行えることが求められる．

一方，プロセッサを用いた処理の実装には，割り込みが用いられることが多い．例えば，モーターを制御するシステムでは，モーターに過電流が流れたときに割り込みを発生させる回路からプロセッサに割り込みを要求し，プロセッサが割り込みの処理を行なってモータを停止させる処理が行われる．このような場合，プロセッサの割り込みの処理の応答が遅れた場合，モーターが破損する可能性がある．このように，割り込み処理には応答までの時間やその処理に要する時間が重要となることがあり，割り込みの処理の評価を正確に行うためには，サイクル精度のタイミング精度でプロセッサのシミュレーションを行う必要がある．しかし，割り込みにより実装される処理はその他の処理と比較して実行される回数が少ないが，そのシミュレーションのためだけにサイクル精度のシミュレーションを行うと他のすべての処理もサイクル精度でシミュレーションを行わなければならなくなり，全体のシミュレーション速度が著しく低下する．

そこで本稿では割り込みの処理を行うときのみサイクル精度でシミュレーションを行い，その他の処理は精度を落ちるものの高速にシミュレーションを行うことにより，全体のシミュレーションを高速に行う方法を提案する．

以降，2 章では割り込みを考慮した命令セットシミュレータに関する関連研究について述べ，3 章では本稿で提案するシミュレーション手法について述べ，4 章では提案するシミュレーションの実装方法について述べる．5 章では提案するシミュレーション手法が有効であることを確認した実験について述べ，6 章でまとめと今後の課題について述べる．

Vol.2012-SLDM-155 No.27
Vol.2012-EMB-24 No.27
2012/3/3

2. 関連研究

命令セットシミュレータ (ISS) はターゲットプロセッサの命令列を解釈し，その命令列がターゲットプロセッサ上で実行されたときの動作をシミュレーションするが，ISS にはパイプラインのステージ単位で命令の動作をシミュレーションしパイプラインの動作までシミュレーションを行う方法と，命令セット上の命令単位で動作をシミュレーションを行う方法がある．本稿では前者をパイプラインレベル・シミュレーション，後者を命令レベル・シミュレーションと呼ぶ．

パイプラインレベルのシミュレーションは，パイプラインの動作，言い換えればプロセッサのクロック単位での動作をシミュレーションするので，シミュレーションのタイミングの精度はサイクル精度となる．一方，命令レベル・シミュレーションでは，命令単位でシミュレーションするため，サイクル精度のタイミング精度は得られない．

命令セットシミュレータで割り込みを正確に扱う最も簡単な方法では，パイプラインレベルでシミュレーションを行い，クロックごとに割り込み要求の有無を確認し，割り込み要求があれば割り込みの処理を行う．しかしこの方法は割り込み要求の処理をクロックごとに行う必要があるので，シミュレーションをクロックサイクル精度で行うことになり，シミュレーションの速度が遅いという問題がある．

この問題に対応するため，Florian により割り込み処理を考慮した高速な命令セットシミュレータを提案している[1)]．この手法では，クロックごとに割り込みの有無を確認するのではなく，シミュレーション対象のプログラムを解析して得られるリストア・ポイントと呼ばれる時点においてのみ，割り込みの有無を確認する．このリストア・ポイントは，メモリへの書き込みアクセスの直前に挿入され，あるリストア・ポイントで割り込み要求が来ている場合，その直前のリストア・ポイントまでシミュレータの状態を復元し，そこからサイクル精度でシミュレーションを行う．この手法では，プロセッサのシミュレーション自体は高速化されるが，メモリバスなどに接続されている外部のハードウェアモデルのシミュレーションは高速化されない．

以上より，本稿では，プロセッサのシミュレーションの精度を切り替えるだけではなく，同時に外部モデルのシミュレーション方法を切り替える方法についても提案する．

時刻	t-3	t-2	t-1	t	t+1	t+2	t+3
命令終了時刻				I1	I2	I3	I4
パイプライン	IF1	ID1	EX1	WB1			
		IF2	ID2	EX2	WB2		
			IF3	ID3	EX3	WB3	
				IF4	ID4	EX4	WB4

図 1 パイプライン・シミュレーション中のパイプライン
Fig. 1 Contents of pipeline at pipeline-level simulation.

時刻	t-2	t-1	t	t+1
命令終了時刻			Ir	
パイプライン	IFr	IDr	EXr	
		IF1	ID1	
			IF2	

図 2 割り込み処理終了時のパイプライン
Fig. 2 Contents of pipeline at the end of an interrupt handler.

3. 提案手法

3.1 命令セット・シミュレータのシミュレーション精度

パイプラインレベル・シミュレーションと命令レベル・シミュレーションの間でシミュレーションのレベルを切り替えるには，パイプラインレベルから命令レベルに切り替えるときと，命令レベルからパイプラインレベルに切り替えるときで切り替える時点より前に実行を開始している命令の取り扱いを考慮しなければならない．

まず，パイプラインレベル・シミュレーションから命令レベル・シミュレーションに切り替えるときには，現在のパイプラインの状態を考慮する必要がある．パイプラインレベルのシミュレーションを行なっている場合は，パイプラインステージの各段で個別の命令を実行している．**図 1** に命令 I1，I2，I3，I4 をパイプラインレベルでシミュレーションしているときのパイプラインの様子を示す．図 1 において，時刻 t でパイプラインレベルから命令レベルに切り替えるためには，既にパイプライン中に存在する I1〜I4 の命令が，最後の WB ステージまで実行されるのを待たなければならない．パイプライン中の最後の命令である I4 が終了するのは時刻 t+3 であるので，レベルの切替まで 3 サイクル待たなければならない．

提案手法では，割り込みから通常の処理に戻るときにパイプラインレベルから命令レベルに切り替える．割り込みから通常の処理に戻るときには，一般的なプロセッサでは“ 割り込みからのリターン ”命令等の分岐命令により，通常の処理への復帰が行われる．分岐命令が実行されたときにはパイプラインのフラッシュが起こるため，図 1 での I2〜I4 の命令が実行されることはない．**図 2** に，時刻 t でリターン命令 Ir の EX ステージが実行され，パイプラインがフラッシュされる様子を示す．これにより，直後の時刻 t+1 より命令レベルでのシミュレーションを行うことが可能である．

命令レベル・シミュレーションからパイプラインレベル・シミュレーションに切り替える

Vol.2012-SLDM-155 No.27
Vol.2012-EMB-24 No.27
2012/3/3

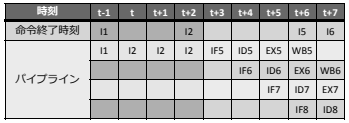

図 3 命令レベル・シミュレーションからパイプラインシミュレーションへの切替
Fig. 3 Switching from instruction-level to pipeline-level simulation

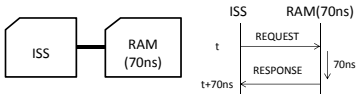

図 4 SystemC TLM におけるプロセッサと外部モデルの通信
Fig. 4 Communication between a processor and external model in SystemC TLM

ときは，切り替える直前に実行している命令が実行に複数サイクルを要する命令であるときに注意が必要である．**図 3** に，時刻 t で命令レベル・シミュレーションからパイプラインシミュレーションに切り替わるときのシミュレーションの様子を示す．

図 3 の場合は，時刻 t において 3 サイクル必要な命令 I2 が実行されているため，その実行が終了した直後の時刻 t+3 までシミュレーションレベルを切り替えることができない．しかし，実際のプロセッサにおいても 1 つの命令の実行が終了するまでは割り込みの処理を行うことはできないので，これにより割り込みの処理が行われるタイミングが不正確になることはない．

以上をまとめると，

- 割り込み要求の処理を開始するときは，現在実行中の命令を最後まで実行してから割り込みの処理へ移る，
- 割り込みの処理を終了し命令レベルに戻るときは，割り込み処理終了時のリターン命令を実行した直後に戻る，

の 2 点を行えば，割り込み時のみパイプラインレベルでのシミュレーションを行いながら，割り込みの応答サイクル数を正しく評価することが可能となる．

3.2 外部モデルの精度の変更

2 節で述べたとおり，外部モデルも含めたシミュレーション全体の速度を高速化するには，プロセッサのシミュレーション精度を切り替えるだけではなく，外部モデルのシミュレーション精度を切り替える必要がある．切り替えるタイミングに関しては，3.1 節で述べたプロセッサのシミュレーションレベルを切り替えるタイミングで切り替えればよいので，切替のときにプロセッサシミュレータのシミュレーションレベルが変更されたことを表す信号を出力し，それを外部モデルが受け取る事で外部モデルのシミュレーション精度を変更することが考えられる．しかしこの方法では，プロセッサのシミュレータに接続している外部のモデルが，プロセッサのシミュレータから出力されるレベルの切替信号に応じて，外部のモデル自身のシミュレーションの精度を変更する機能を持っている必要がある．そのため，この機能を持っていない外部モデルを使用するには，新たに外部モデルにシミュレーション精度を変更する機能を実装する必要がある．これでは従来のモデルとの互換性が失われるという問題がある．そこで，外部のモデルの変更を行わずにシミュレーションの精度を変更することを考える．

提案手法のプロセッサのシミュレータを外部の SystemC モデルと SystemC TLM[3] の仕様で接続する．これにより，外部のモデルとの通信をどのアドレスにどのようなデータを読み書きするのに抽象化して表現できる．よって，プロセッサのシミュレータと外部のモデルは接続するバスの信号に関係なく接続することが可能となる．**図 4** にプロセッサと外部 RAM モデルとの TLM における通信の様子を示す．図 4 の RAM モデルはアクセス時間が 70[ns] 必要とする．そのため，プロセッサが要求を発行してから応答が返るまでに 70[ns] のシミュレーション時間が経過している．しかし，一度のメモリアクセスごとに要求と応答を行なう部分までシミュレーションを行うと，シミュレータ上で要求や応答をシミュレーションするのにかかる時間や応答を待つ時間が無視できなくなる．そのため，SystemC TLM ではアクセス先のメモリのデータが格納されているホストマシンでのアドレスを通じて直接やり取りするために Direct Memory Interface(DMI) が用意されている．DMI を用いることにより，タイミング精度は落ちるが高速に外部モデルにアクセスすることが可能になる．この DMI の機能は外部モデルの仕様で使用の可否が決まるが，DMI が使用できない場合は通常の方法で通信を行うことにより接続性を維持することができる．

以上より，プロセッサが命令レベルで動作している場合は外部モデルと可能な限り DMI を通じて通信し，パイプラインレベルで動作している場合は通常の方法で通信を行うことにより，外部モデルのシミュレーション精度を変更することができる．

4. 実　　装

4.1 プロセッサシミュレータの実装

図 5 に，提案するプロセッサシミュレータのサイクルごとの状態遷移図を示す．各状態の詳細は以下のとおりである．

停止　プロセッサシミュレータは停止している．リセット割り込み要求が来た場合は，割り込みハンドラ移行準備処理状態に移行する．

パイプラインレベル　プロセッサシミュレータはパイプラインレベルでシミュレーションを行なっている．1 つ以上の割り込み要求がある場合は，割り込み準備 (パイプラインレ

Vol.2012-SLDM-155 No.27
Vol.2012-EMB-24 No.27
2012/3/3

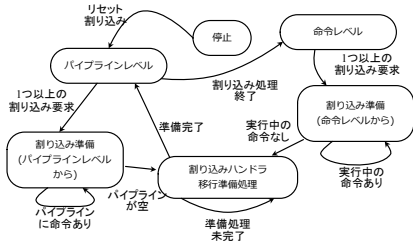

図 5 サイクルごとのプロセッサシミュレータの状態遷移
Fig. 5 Transition of the processor simulator's state every cycle.

ベルから) 状態にに移行する．割り込みからの復帰命令が実行され，パイプラインのフラッシュが行われた場合は，命令レベル状態に移行する．それ以外の場合は，パイプラインの各ステージの命令を実行する．

命令レベル プロセッサシミュレータは命令レベルでシミュレーションを行なっている．現在実行中の命令がない場合は命令をフェッチし，現在実行中の命令がある場合はその命令の実行を続ける．1 つ以上の割り込み要求がある場合は，割り込み準備 (命令レベルから) 状態にに移行する．

割り込み準備 (パイプラインレベルから) パイプラインレベルシミュレーション中に割り込み要求を検知した．プロセッサシミュレータは，パイプライン中に処理中の命令が残っているなら，すべての命令の実行が終了するまで，新しい命令のフェッチを行わずにパイプラインシミュレーションを続ける．パイプラインが空になったら割り込みハンドラ移行準備処理状態に移行する．

割り込み準備 (命令レベルから) 命令レベルシミュレーション中に割り込み要求を検知した．プロセッサシミュレータは，現在命令レベルで実行中の命令の実行が完了するまで待ち，実行が完了したら，割り込みハンドラ移行準備処理状態に移行する．

割り込みハンドラ移行準備処理 割り込みハンドラへ移行する準備を行う．これは，シミュレーション対象のプロセッサによって異なるが，一般的には割り込み終了後に復帰する命令のアドレスを退避し，割り込みハンドラの命令のアドレスを計算し，割り込みハンドラの命令のアドレスへ分岐する．準備処理が完了したら，パイプラインレベル状態に移行する．

4.2 バス・アクセサの実装

SystemC TLM では，通信を開始し要求を伝える側と，通信の対象となり要求を受け取る側が分かれており，前者をイニシエータ，後者をターゲットと言う[3]．外部モデル通信部は，プロセッサシミュレータから外部モデルとの通信要求を受け取り，SystemC TLM のイニシエータとして外部モデルに要求を伝える．

プロセッサシミュレータは，外部モデル通信部の次の 3 つの関数を呼び出して外部モデル通信部に要求を伝える．

read, write read 関数と write 関数は，プロセッサシミュレータが外部モデルと通信を行うときに呼び出される．プロセッサシミュレータはデータを読み書きするアドレスを渡す．外部モデル通信部は，現在のプロセッサシミュレータのシミュレーションレベルが命令レベルで，外部モデルが DMI を使用可能であれば DMI を使用して通信する．それ以外の場合は，通常の通信方法で通信する．

switchLevel switchLevel 関数は，プロセッサシミュレータのシミュレーションレベルが変更されたときに呼び出される．図 5 の状態遷移図で表すと，

- 割り込み準備状態から割り込みハンドラ移行準備処理へ遷移するとき，
- パイプラインレベル状態から命令レベル状態へ遷移するとき，

に呼び出される．

外部モデル通信部は，変更後のプロセッサシミュレータのシミュレーションレベルを受け取り記憶する．

以上より，read/write 関数を呼び出せば外部モデルとの通信がシミュレーションレベルに応じた方法で行われる．

5. 評価実験

5.1 実験方法

図 6 に評価実験に用いたシステムの構成を示す．このシステムでは，提案手法を用いた命令セットシミュレータに，命令メモリが直接接続され，データバスを介してデータメモリが接続されている．

実験で用いたシステムのプロセッサは Brownie STD 32[6] である．このプロセッサはワード長が 32 ビット，データバス幅が 32 ビットで，32 ビットのレジスタを 32 本内蔵している．パイプラインは IF, ID, EXE, WB の 4 ステージ構成である．また，外部からの割り込み要求には最低 5 サイクル，パイプラインの状態に応じてはパイプライン中に残っている命令を実行し終えるために，更に数サイクル追加で必要とする．

評価実験では，プロセッサに 100MHz のクロックを入力し，メモリのレジスタにロード

Vol.2012-SLDM-155 No.27
Vol.2012-EMB-24 No.27
2012/3/3

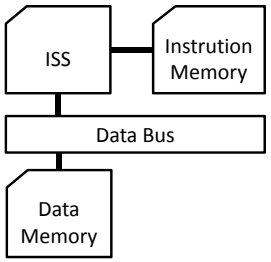

図 6 実験対象のシステムの構成
Fig. 6 Architecture of target system

表 1 割り込みハンドラ内の処理
Table 1 Process in the interrupt handler

命令		処理内容
addi	r19, r19, #1	割り込み回数カウンタを加算
sw	0(r0), r19	メモリに書きこみアクセス
reti		割り込み処理から通常処理に復帰

時刻(サイクル)		0	1	2	3	4	5	…	14
パイプライン	IF	addi	sw	reti				…	
	ID		addi	sw	reti			…	
	EXE			addi	sw	reti		…	
	WB				addi	sw	sw	…	sw

図 7 割り込みハンドラ処理中のパイプライン
Fig. 7 Contents of pipeline on handling the interrupt handler.

した後ストアするという動作を 65,536 回繰り返し，外部割り込み要求が来た場合はメモリへストアを行うプログラムを作成した．

表 1 に割り込みハンドラ内の処理を，**図 7** に割り込みハンドラが実行されるときのパイプラインの様子を示す．図 7 の時刻 0 は，割り込みハンドラに処理が移った時刻を表す．時刻 0 から 3 では，表 1 の割り込みハンドラ内の命令列が順にパイプラインのステージを進んでいる．時刻 4 で sw 命令が WB ステージに進むと，プロセッサはデータメモリに書きこみアクセスを要求する．データメモリのアクセス時間が 100ns のとき，sw 命令は 10 サイクルの間ストールし，時刻 13 で実行が完了する．一方，sw 命令と平行して時刻 4 に reti 命令の EXE ステージが処理される．reti 命令は割り込みハンドラから通常の処理への復帰命令であるので，プログラム・カウンタを復帰先のアドレスに設定し，パイプラインがフラッシュされる．しかし，直前の sw 命令がストールしているため，パイプラインのフラッシュが完了するのは sw 命令終了時である．よって，実験で使用したプログラムの割り込み処理ハンドラの実行サイクル数は，データメモリのアクセス時間が 100ns のときは 15 サイクルとなる．

このプログラムを作成したシミュレータを用いて，シミュレーションレベルの切替の有効無効，データメモリのアクセス時間，外部割り込み要求の発生間隔を変化させてシミュレーションを行い，シミュレーションに要する時間と，データメモリに対して DMI 経由ではない通常の通信が行われた回数を計測した．

表 2 アクセス時間とシミュレーション時間
Table 2 Access time and simulation time.

アクセス時間 (ns)	レベルの切替	通信回数 (回)	所要時間 (秒)
0	有効	4418	0.4
0	無効	5957	1.0
50	有効	4681	0.41
50	無効	14043	1.4
100	有効	4977	0.43
100	無効	23405	1.88

また，シミュレーションレベルの切替を有効，データメモリのアクセス時間を 100ns，外部割り込み要求の発生間隔を 1002ns として，割り込みに対する応答の様子を観測した．1002ns とした理由は，プロセッサのクロックサイクルの周期である 10ns で割り切れない数として，プロセッサのクロックサイクルの途中で外部割り込み要求が発生したときに正しく応答するかを観察するためである．

実験のためのシミュレーションは，CPU が Intel Core2 Quad Q9650 @ 3GHz，RAM が 4GB，OS が Debian/GNU Linux 6.0 の計算機上で行った．

5.2 実験結果

表 2 に，外部割り込み要求の発生間隔が 1000[ns] のときのデータメモリのアクセス時間，シミュレーションレベルの切替の有効無効，データメモリに対する通常の通信の回数と，シミュレーションに要する時間の関係を示す．表 2 より，データメモリのアクセス時間が短いときは，シミュレーションレベルの切替が有効なときと無効なときの速度の差は小さいが，データメモリのアクセス時間が長くなるほど，シミュレーションレベルの切替が有効なときの方が短時間でシミュレーションが完了していることがわかる．また，シミュレーションレベルの切替が有効なときは無効のときと比較して通常の通信が行われた回数が少なくなっており，DMI 経由での通信が行われていることが分かる．

データメモリのアクセス時間が短いときは，シミュレーション全体でデータメモリの応答を待つ時間も短くなるため，シミュレーションレベルの変更を無効にした場合でも無駄なデータメモリの待ち時間がほとんど発生しないため，シミュレーションレベルの変更を有効にした場合との差が小さくなっているが，その場合でも，シミュレーションレベルの変更を有効にした場合は無効にした場合と比較して，プロセッサシミュレータのシミュレーションレベル変更による高速化により，2 倍程度高速にシミュレーションできている．

Vol.2012-SLDM-155 No.27
Vol.2012-EMB-24 No.27
2012/3/3

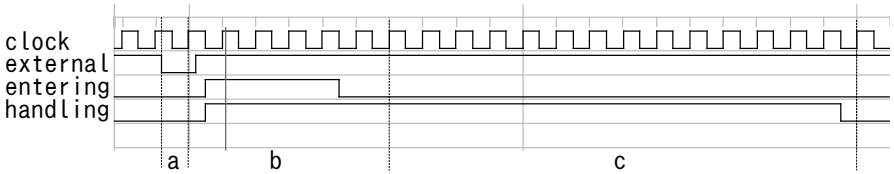

図 8 割り込みに対する応答の様子
Fig. 8 Reaction for an interrupt

データメモリのアクセス時間が長い時は，プロセッサシミュレータのシミュレーションレベル変更による高速化に加えて，通信方法の切り替えによる高速化により，シミュレーションレベルの変更を有効に場合は無効にした場合よりも 4 倍程度高速にシミュレーションできている．

図 8 に，割り込みに対する応答の様子を示す．図 8 には，4 つの信号が含まれており，以下にその信号の意味を示す．

clock プロセッサシミュレータに入力しているクロック信号

external プロセッサシミュレータに入力している割り込み要求信号．負論理となっており，割り込み要求があるときには 0 となる．

entering プロセッサシミュレータから出力される．割り込みハンドラへ移行中であることを表す．

handling プロセッサシミュレータから出力される．割り込み処理中であることを表す．

図 8 の信号は，区間 a, b, c の 3 つの区間に分けることができる．以下に各区間の意味を説明する．

区間 a 区間 a は，外部からの割り込み要求の発生から，プロセッサが割り込み処理を開始するまでを表している． 区間 a 左端の時点で，外部から割り込み要求が通知されているが，クロックサイクルの途中であったため，まだ割り込みは受け付けられていない．

区間 b 区間 b は，外部からの割り込み要求をプロセッサが検出てから，割り込みハンドラへ処理を移行する準備を終えて割り込みハンドラへ処理を移行するまでを表している．区間 b の長さは 5 サイクルとなっており，5.1 節で示した Brownie STD 32 の割り込みハンドラへ処理を移行するまでにかかるサイクル数と一致している．

区間 c 区間 c は，割り込みハンドラへ処理を移行してから，割り込みハンドラの処理が終了し，割り込み処理が終了するまでを表している．区間 c の長さは 15 サイクルとなっており，5.1 節で示した割り込みハンドラの実行サイクル数と一致している．

以上より，外部割り込み要求に対する応答を正しくシミュレーションしていることがわかる．

6. まとめと今後の課題

本稿ではシミュレーションレベルをシミュレーション時に変更可能なシミュレータを提案した．また，実験により提案シミュレータが，割り込みを用いたシステムにおける割り込みの応答の評価に用いることができる精度を持っていることを確認した．更に，シミュレーションレベルの変更を行うことにより常にパイプラインレベルでシミュレーションを行う場合よりも十分高速にシミュレーションを行うことができることを確認した．

今後の課題としては，シミュレーションレベルの変更を行うことの有効性は確認できたものの，シミュレーション速度が十分とは言えないため，Just-In Time Compile などの手法を用いてのシミュレーション速度の高速化が挙げられる．また，現在の手法では割り込み要求が来たときに現在実行中の命令の途中経過を保存して割り込み処理を行う Cortex-M 系のプロセッサに対応することができないので，そういったプロセッサに対応するということもあげられる．

参 考 文 献

1) Brandner Florian, “Precise simulation of interrupts using a rollback mechanism,” Proceedings of th 12th International Workshop on Software and Compilers for Embedded Systems, pp.71-80, 2009.
2) “IEEE Standard SystemC Language Reference Manual,” IEEE Std 1666-2005, 2005.
3) Open SystemC Initiative, “OSCI TLM-2.0 LANGUAGE REFERENCE MANUAL,” Open SystemC Initiative, July. 2009.
4) L. Cai, and D. Gajski, “Transaction level modeling: an overview,” Proc. of the 1st IEEE/ACM/IFIP international Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS ’03), pp. 19–24, Oct. 2003.
5) M. Montoreano, “Transaction Level Modeling using OSCI TLM 2.0,” Technical report, Open SystemC Initiative, May. 2007.
6) 岩戸 宏文, 稗田 拓路, 田中 浩明, 佐藤 淳, 坂主 圭史, 武内 良典, 今井 正治, “ASIP 短期開発のための高い拡張性を有するベースプロセッサの提案,” デザインガイア 2007-VLSI 設計の新しい大地を考える研究会-, 情報処理学会研究報告, vol. 2007, no. 114, pp. 133-138, 2007 年 11 月.