投機的マルチスレッド実行モデルのSystemCによるモデリング

高済 健吾　横田 隆史　大津 金光　馬場 敬信 †
宇都宮大学工学部情報工学科‡

1 はじめに

システムLSIや組み込みシステムの設計において、システム記述言語を用いた設計手法の適用例が拡大しつつある。だが、一定の構成方式・実装方式では有効性が確認されているものの、適用手法が確立されておらず試行錯誤を余儀なくされる事例も少なくない。

本稿では、設計手法の確立していない事例として均質な処理ユニットを複数用いての投機的マルチスレッド実行モデルを選び、SystemC[1] でのモデリングを試みる。

2 投機的マルチスレッド処理

2.1 処理の概要

我々が前提としている投機的マルチスレッド処理は以下のように行われる。まずプログラムを処理ごとにスレッドと呼ばれる単位に分割する。スレッドは投機スレッドと非投機スレッドの2種類があり、ソースコードを抜き出すことで生成する。非投機スレッドは逐次的に実行し、投機スレッドは並列に実行する。

投機的マルチスレッド処理の様子を図1に示す。実行プログラムが投機対象部分に到達するまでは非投機スレッドを単一の処理ユニットで実行し、投機対象部分に到達すると投機スレッドを図に示すように起動する。次々に投機スレッドが起動し、複数の投機スレッドを同時に実行している状態になる。

投機が成功した場合はデータをそのまま書き戻し、処理を継続する。投機が失敗した場合には当該スレッドとその後の全ての投機スレッドの処理を廃棄し、別の処理ユニットで新たに非投機スレッドを起動して計算し直す。オリジナルのプログラムのループ部分の中で実行頻度の高い処理(ホットパス)を投機スレッドに割り当て、投機処理が成功することによって、プログラムの高速化を期待できる。

2.2 投機的データの管理

投機的マルチスレッド処理では複数のスレッドが同時に処理を実行する。このため複数のスレッドから一斉に、あるいは元々のプログラムの実行順序とは違う順序でデータの読み込みや書き込みが行われることがある。これによってプログラムのデータ依存関係が破壊され、正しい実行結果を得ることはできない。また投機処理が

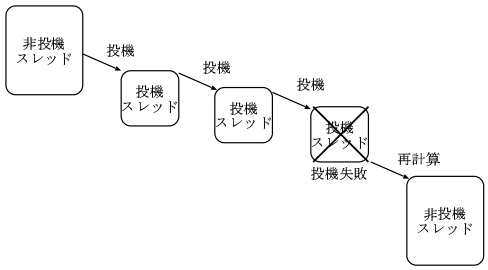

図 1: 投機的マルチスレッド処理

失敗した場合には、失敗時のデータを無効化して投機前の状態から計算し直す必要がある。このためには投機処理の実行前のデータを確保していなければならない。

このデータ依存違反の解決や、投機処理の失敗時用のデータ確保のためのデータ管理機構が必要となる。このようなデータ管理の方式として、Speculative Versioning Cache(SVC)が挙げられる[2]。SVCは投機的マルチスレッド処理において、それぞれの処理ユニットに対してキャッシュを用意し、キャッシュデータとスレッドを制御するVersion Control Logic(VCL)を置くことでデータの一貫性を保つ方式である。

3 SystemCによる実行モデルの記述

本研究ではアプリケーションにSPECint95のcompressを選択し、SVC方式を利用して投機的マルチスレッド実行モデルを記述した。投機対象部分は圧縮を行うcompess()のループ内のホットパス部分である。SystemCの設計手順に従い、最も抽象度が高く、時間概念のないアンタイムド機能モデルでモデリングを行った。

3.1 スレッド生成

SystemCはC++をベースにしたシステム記述言語である。このためC言語で書かれたcompressのソースコードを処理ユニットモジュールのメンバ関数とすることでスレッドを生成できる。生成したスレッドは処理ユニットのスレッドプロセスとしてSC_THREADで登録する。

ここでcompressのmain関数内での処理フローを図2に示す。compressの処理はテキストデータの初期化を行うfill_text_buffer()、初期化したテキストデータにデータを追加するadd_line()、テキストの圧縮/解凍どちらを行うか判定するspec_select_action()、圧縮解凍後にテキストデータの照合を行うcompare_buffer()から成る。spec_select_action()で圧縮を行うcompress()が呼び出される。add_line()からcompare_buffer()のループ部分は25回行われる。

本来の投機的マルチスレッド処理のスレッド生成方法に従えば、fill_text_buffer()からspec_select_action()

* A modeling of a speculative multi-thread execution model using SystemC
† Kengo Takasumi, Takashi Yokota, Kanemitsu Ootsu, and Takanobu Baba
‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

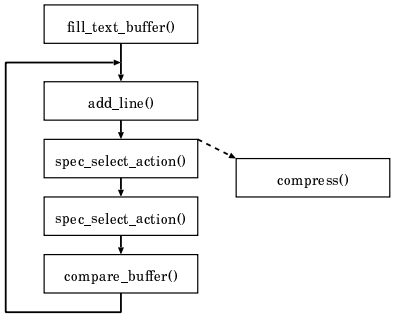

図 2: compress 処理フロー (main() 内)

までの部分と二回目の spec_select_action() から compare_buffer() までの部分で非投機スレッドを構成する。しかし、今回は fill_text_buffer() から compare_buffer() まで関数ごとにコードを抜き出して非投機スレッドを生成した。これは fill_text_buffer() はデータ初期化のため一度しか実行しないが、他の関数はループ部分になっているため、まとめて非投機スレッドを構成すると fill_text_buffer() もループに含まれ、初期化を何度も行う無駄な処理ができるためである。

投機スレッドは投機対象の compress() のホットパス部分のコードを抜きだして生成した。この投機スレッドには投機成功/失敗を示すフラグを用意し、成功/失敗が判明するとその情報を VCL に報告する。

3.2 投機的マルチスレッド処理システム

システムの構成を図 3に示す。本研究で記述したシステムはスレッドを実行する処理ユニット、スレッドの計算結果を格納するキャッシュ、メモリ、キャッシュデータとスレッドを管理する VCL の 4 つのモジュールから成り、それらを構成要素ごとに SystemC で記述した。各モジュールは SystemC プログラムの main() で呼び出すことで組合せることができ、処理ユニットの台数を変更することも可能である。今回、処理ユニットとキャッシュは 16 台で構成した。これは同一の処理ユニット、キャッシュを 16 回インスタンシエートすることで実現できる。

スレッドは投機/非投機にかかわらず VCL が制御する。あらかじめ各スレッドに固有の番号 (スレッド番号) を割り当てておく。VCL は図 2の順序でスレッドが実行されるよう、処理ユニットに対し起動すべきスレッド番号を送る。投機実行時には各処理ユニットに対して投機スレッドの番号を送り続け、処理ユニットから投機失敗報告があれば再計算のスレッド番号を一度送り、その後また投機スレッドの番号を送る。

処理ユニットはスレッド番号をチェックし、それに従って実行するスレッドを切り替える。スレッドの実行を終了次第、終了情報を VCL へ報告し、投機スレッドを実行した場合は同時に投機成功/失敗情報も送信する。

キャッシュ-処理ユニット間、キャッシュ-メモリ間にはデータバスを接続し、またキャッシュ-VCL 間にはデータの状態や別のキャッシュへの read 要求、write 要求を送るバスを繋ぐ。それぞれのバスは SystemC で既に用意

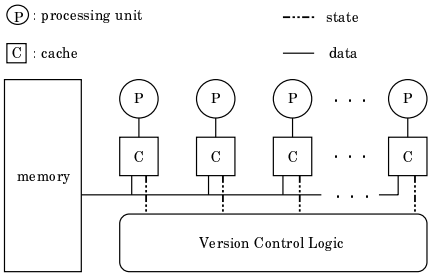

図 3: 投機的マルチスレッド処理システム

されている sc_fifo チャネルを利用することで実現した。この sc_fifo チャネルは main() で呼び出す際に通信するデータ型を指定しなければならない。そのため、投機処理のために必要なデータの型を調べ、その型に合わせてチャネルをインスタンシエートした。

メモリ、キャッシュはアドレスを受け取った後、データ読み込み/書き込みを行うように記述した。メモリの構成についても、あらかじめ投機処理に必要なデータの型及び容量を調査し、それに合わせてメモリ確保を行った。

4 今後の課題

本研究では対象アプリケーションに compress を選び投機的マルチスレッド処理実行モデルを記述した。今回のモデリングでは compress のプログラムからコードを抜きだすことでスレッドを生成し、順々に番号を割り当て制御する。このため記述したモデルは compress に特化した形になっているが、別のアプリケーションに適用する際には今回と同様にスレッドの生成、スレッド番号割り当て、キャッシュやメモリ、バスの再構成を行うことで容易に実現することができる。様々なアプリケーションに対して適用することで設計パターンを模索し、上位設計手法として確立させる予定である。

5 おわりに

本稿では SPECint95 の compress を題材に、投機的マルチスレッド実行モデルの SystemC を用いたモデリングを行った。今後は SPECint95 の他のプログラムについて適用し、将来的には設計パターンの考えを導入することにより抽象度の高い上位設計手法として発展させる計画である。

謝辞 本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (B)14380135，同 (C)14580362，若手研究 14780186) の援助による。

参考文献

[1] Thorsten Grötker 他 訳：柿本勝 他 “SystemC によるシステム設計” 丸善株式会社

[2] Sridhar Gopal 他: “Speculative Versioning Cache” High Performance Computing Architecture 1998