DesignWare ARC nSIM

一切の妥協を排し、優れた速度、精度、可視性を実現した命令セット・シミュレータ

シノプシス　R&Dエンジニア　Igor Böhm

命令セット・シミュレータ（ISS）は、プロセッサ・アーキテクチャの設計空間探索や検証だけでなく、コンパイラ、オペレーティング・システム、アプリケーションの開発にも欠かせないツールです。しかしISSはその用途によって求められる条件が大きく異なるため、あらゆるニーズに対応したISSを開発するのは大変困難です。たとえばハードウェア検証では、現実には起こり得ないようなランダムに生成したコーナーケースのシナリオであってもアーキテクチャのビヘイビア・レベルで絶対的精度が求められますが、コンパイラ開発では実チップが完成する前に最適化コンパイラを作成できるように、機能の正しさ、性能、そして充実したプロファイリング・フィードバック機能が要求されます。

このため、通常はアーキテクチャの精度を犠牲にしてシミュレーション速度を向上させるというトレードオフが行われますが、シノプシスのARC nSIMなら、このような妥協は一切必要ありません。本稿では、ARC nSIMがどのようにして高いシミュレーション速度とアーキテクチャ精度を両立させているかについて、シノプシスのR&Dエンジニア、Igor Böhmがご説明します。ARC nSIMではJIT（Just-In-Time）コンパイル・テクノロジによってシミュレーション速度を最大限に高め、場合によっては最終的な実チップを上回る速度を達成します。また、nSIMはRTL検証メソドロジに深く統合されており、内部検証プロセスを高速化するだけでなく、シミュレーション・レベルでアーキテクチャの正しさを保証します。

nSIM Pro：シミュレーション速度の飛躍的な向上

動的コンパイルは、プログラムのシミュレーション速度を実行時に高速化する技術として定評があり、JITコンパイルとも呼ばれます。動的コンパイルの基本的な考え方とは、プロファイル情報が最も多く得られる時点、すなわち実行時にマシン固有コードの生成と最適化を行うというものです。事実、シミュレーション速度を向上させる最適化手法の中には、実行時に得られる動的な情報がなければ事実上適用できないものもあります。経験的に言って、動的コンパイラではインタプリタ型シミュレーションに比べ最大で約10倍の高速化が可能です（図1）。

動的コンパイルでは、実行時に必ず発生するオーバーヘッドがプログラムの全体的な実行時間に影響します。動的コンパイルに費やす時間と全体的な実行時間の間には、トレードオフの関係があります。動的コンパイルを積極的に実行すると非常に効率のよいネイティブ・コードが生成されますが、コンパイルに時間がかかって全体的なシミュレーション時間は長くなります。一方、動的コンパイルによるコード最適化に費やす時間を短くすると、シミュレーションしたプログラムの実行時性能が十分に得られません。nSIM Proの動的コンパイル・インフラストラクチャには、動的コンパイルで発生するレイテンシを抑えつつシミュレーション時間の高速化を図る3つの革新的な手法が採用されています。

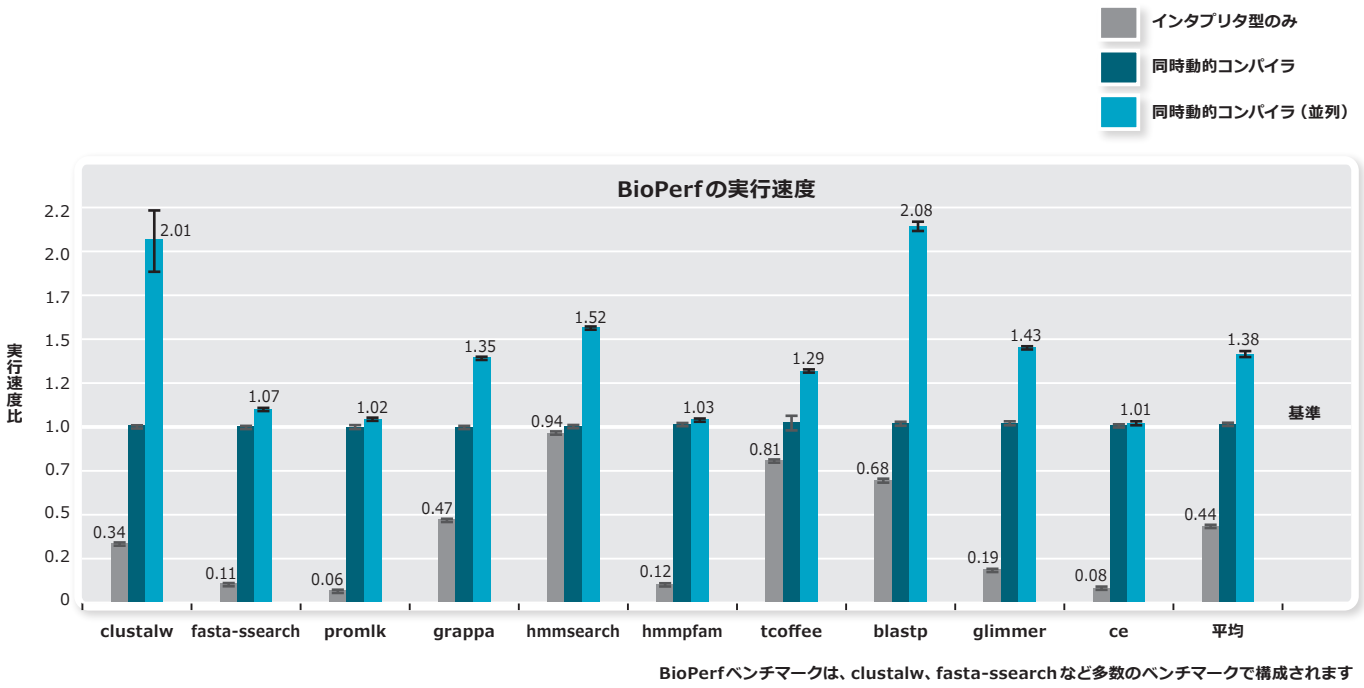

図1. BioPerf ベンチマーク・スイートの実行結果。以下の3通りでシミュレーション速度を比較。
(a) nSIM のインタプリタ型シミュレーションのみを利用した場合
(b) nSIM Proで同時動的コンパイラを1つのみ使用した場合
(c) nSIM Proで同時動的コンパイラを複数並列に使用した場合

1. **適応型優先付け：**最近実行したプログラム領域を優先的に最適化
2. **並列コンパイル：**これらの領域を並列にコンパイル
3. **同時動的コンパイル：**ターゲット・プログラムのシミュレーションとコンパイルを同時進行

適応型優先付け

シミュレーション時に行われる動的コンパイルは、なるべく実行頻度の高いプログラム領域に限定して最適化を適用するのが理想ですが、何をもって「実行頻度が高い」とするかはアプリケーションによって異なります。従来、動的コンパイルを利用したシステムでは経験的にしきい値を決定するか、ユーザーがしきい値を設定するのが一般的でした。しかし、ユーザーがどのようなプログラムを実行するかは、システム側で事前に知ることができず、また、各アプリケーションに適したしきい値をユーザーが見つけるには時間と手間がかかるという問題がありました。nSIMでは、プログラムのホットスポット選択アルゴリズムを自動的に調整することによってこの問題を解決しています。これにより、nSIMではEEMBC® CoreMark®などの小規模な組込みベンチマークであっても、あるいはSPEC CPU™ 2006ベンチマークに含まれるGCC Cコンパイラのシミュレーションのような大規模なベンチマークであっても、最高のパフォーマンスが得られます。

並列コンパイル

動的コンパイルのレイテンシを効果的に削減するには、単位時間当たりにコンパイルされるアプリケーション・ホットスポットの数を増やしてワークロードのスループットを高める必要があります。このため、nSIMはプロファイル済みコードを解析し、依存関係のないプログラム領域を並列にJITコンパイルするようにしています。

同時動的コンパイル

動的コンパイルのオーバーヘッドをさらに軽減するため、nSIMでは動的コンパイルを実行しながらインタプリタ型シミュレーションも同時進行するようになっています。このため、JITコンパイラが特定のプログラム・ホットスポットに対するコード生成を完了するまでシミュレーションが一時停止するということがありません。シミュレーション中のアプリケーションは一時停止することなく、常に快適な応答性を維持しながら高速に動作します。これは、リアルタイムのシミュレーション速度が要求されるようなアプリケーションや、完全なシステムOSシミュレーションのようにユーザーとのやりとりを伴うようなシミュレーションでは特に重視される点です。

コンパイルするプログラム領域を動的に検出・選択するとともに、並列コンパイルおよび同時動的コンパイルを組み合わせる革新的な手法を導入したnSIMは、ワークロードの変化に自動的に適応し、現在のマルチコア・シミュレーション・ホストの並列性を効果的に利用しつつ、シミュレーションとコンパイルを同時に実行するなど、真の意味でスケーラブルなシミュレータとなっています。図1からも明らかなように、nSIMでインタプリタ型シミュレータのみを使用した場合に比べ、nSIM Proに内蔵された並列コンパイルと同時動的コンパイルを利用した方がシミュレーション速度は大幅に向上します。

実チップと同じ動作を実現したnSIM

プロセッサ・シミュレータの動作が最終的なハードウェア製品の動作と一致していなければ、開発サイクルの終盤に変更作業を強いられ、大きなコストが発生します。nSIMはハードウェア検証プロセスに「ゴールデン・マスター・モデル」として深く統合されており、この検証メソドロジを使用することでnSIMとRTLの動作が正確に一致するようにしています。

新しい命令セット・アーキテクチャ（ISA）の機能はすべて、プログラマのリファレンス・マニュアルに存在する記述に基づいてRTLとnSIMにインプリメントされます。最後に、検証の段階でRTLとnSIMの両方をロック・ステップ動作させ、各ステップの後にRTLとnSIMモデルのアーキテクチャ / マイクロアーキテクチャ・ステートを詳細に比較します。一般に、この検証メソドロジをオンライン検証またはコ・シミュレーション検証と呼びます。

nSIMは、コ・シミュレーションを利用して毎秒数千のランダム生成テストおよびダイレクト・テストを実行し、RTLとの同期をとりながら動作が正しいことを確認します。オンライン検証ではエラーを即座にピンポイントで特定できるため、オフライン検証手法に比べ検証プロセスが大幅に高速化します。オフライン検証では、実行時ほど多くのトレース情報が含まれていないトレース・ファイルの事後解析に長時間を費やしますが、オンライン検証ではこのような作業は不要です。しかも、オフライン検証は命令トレースを使用するため、ファイル・サイズやファイル・ストレージといった現実的な制約が存在します。コ・シミュレーションにはこのような制約はなく、数十億もの命令シミュレーションを実行してRTLとnSIMに対して容易にストレステストが行え、RTLとnSIMの両方でより高いテスト・カバレッジが得られます。

nSIMはRTL検証プロセスに深く統合されているため、nSIMシミュレータを用いて開発したプログラムは最終的なハードウェア上でも確実に同じ動作が得られます。また、現実にはユーザーが直面しないようなコーナーケースもRTL開発の段階ではテストと検証が必要ですが、そのようなコーナーケースでも動作の同一性が保証されます。

詳細情報

- nSIMのウェブサイト　http://www.synopsys.com/dw/ipdir.php?ds=sim_nSIM
- nSIMのデータシート　https://www.synopsys.com/dw/doc.php/ds/cc/arc_nsim.pdf

著者紹介

Igor Böhm：シノプシスのR&Dエンジニアで、ARC nSIMシミュレータ製品担当テクニカル・リード。英国エジンバラ大学の研究グループProcessor Automated Synthesis by iTerative Analysis（PASTA）プロジェクトに参加し、一般的な市販ハードウェア上で実チップを上回るシミュレーション速度を実現する命令セット・シミュレータ（ISS）用動的コンパイル・インフラストラクチャを開発。ARCシミュレータに採用されたこの技術は、現在シノプシスがライセンスを提供。エジンバラ大学のInstitute for Computing Systems Architectureにて博士号、ケプラー大学（オーストリア、リンツ）のInstitute for Systems Softwareにて修士号を取得。