卒業論文

題目

ヘテロジニアスマルチコアにおけるプログラムの実行最適化を目指したFabScalarの分岐予測器の拡張

指導教員

佐々木　敬泰

2013年

三重大学 工学部 情報工学科
計算機アーキテクチャ研究室

三好　聖二 (409851)

内容梗概

特徴の異なるプログラムやプログラム中の特徴の異なるフェーズを効率的に実行するための手段として，ヘテロジニアスマルチコアが注目されている．ヘテロジニアスマルチコアとは様々な構成のコアを搭載し，それぞれの特徴の異なるコアを状況に合わせて使い分けることで，計算能力の向上や消費電力の低減を目指すものである．しかし，特徴の異なる複数のコアの設計やそれぞれのコアに対するキャッシュシステム及びそれらを相互に接続するバス網などの組み合わせが膨大となるため，設計・検証に要する時間が膨大となり，これが実用の面で大きな障害となっている．

この問題を解決するために，任意の構成のスーパースカラコアを自動生成するツールセットとしてFabScalarが提案されている．FabScalarはスーパースカラのWay数やパイプライン段数のようなパラメータを与えることでスーパースカラコアを自動的に構成する．設計者はFabScalarのパラメータを変更することで，プログラムの特徴に応じたスーパースカラコアを生成できる．しかし，FabScalarに実装されている分岐予測器は，精度の低いBimodalという方式の分岐予測器１種類のため，プログラムの特徴を考慮した上で異なる分岐予測器を実装するといったような，分岐予測器方面からのコア性能の最適化を行えない．そこで本研究ではFabScalar上への新たな分岐予測器の実装を容易にする機構を実装することで，将来のFabScalarのより多様なプログラムへの最適なコア生成を目指す．

本研究では，将来の分岐予測器のFabScalar上への追加実装を支援するため，多くの分岐予測器の要素技術であるGlobal History Register(GHR)を実装する．また，GHRの設計の妥当性を評価するため，GHRを利用する分岐予測器としてGshare分岐予測器・Hybrid分岐予測器の２つの分岐予測器をFabScalar上に実装し，評価する．

その結果，GHRを利用した分岐予測器は従来のBimodal分岐予測器に比べ，ほぼ全ての評価で分岐予測精度が向上したため，GHRの設計が妥当であることがわかったわかった．

Abstract

Heterogeneous multi-core architectures attract many attentions since it can streamline executions of diverse programs and program phases with various features efficiently. Heterogeneous multi-core provides multiple differently-designed superscalar core types. By selecting the most appropriate core for a program to execute, it can achieve the most high performance with proper energy consumption. However, design and verification costs become large accroding to the number of different core types, cache systems and bus systems.

The FabScalar toolset is proposed to solve this problem. FabScalar can generate diverse superscalar cores automatically, and it can be modified the specifications of a superscalar core by changing parameters. But branch predictor implemented in FabScalar is only Bimodal, and therefore user of FabScalar can not optimize performance in the area of branch predictor. This paper proposes to implement of various branch predictors in FabScalar in order to generate more diverse core of FabScalar. As first step, In order to help additional implementation of branch predictors we implement the GHR in FabScalar. The GHR is key elements of many branch predictors and complex circuit which is connected to many unit in a processor core. In addition, we implement the Gshare and Hybrid branch predictor using the designed GHR in FabScalar for evaluation the validity of design of the GHR.

In the result, these branch predictors used the GHR has higher performance than existing Bimodal branch predictor in the almost all of evaluation. For that reason, it is verified that design of the GHR is relevant.

目 次

図 目 次

表 目 次

1 はじめに

近年，多様なプログラムをより効率的に実行するための手段として，構成の異なるスーパースカラコアを用いたヘテロジニアスマルチコアプロセッサが注目されている．しかし，構成の異なる各コアの構成，それぞれのコアに対する様々なキャッシュシステム，それらを相互に接続するバス網の組み合わせの膨大さから，設計・検証にかかる時間が膨大となり，実用の面で大きな障害となっている．

この問題を解決するために，任意の構成のスーパースカラコアを自動生成するツールセットとして FabScalar[1] が提案されている．FabScalar はパラメータを与えることでスーパースカラコアを自動的に構成する．設計者は FabScalar のパラメータを変更することで，様々なプログラムの特徴に応じたスーパースカラコアを生成できる．しかし，実装されている分岐予測器が，精度の低い Bimodal という方式の分岐予測器 1 種類のため，プログラムの特徴を考慮した上で異なる分岐予測器を実装するといったような，分岐予測器方面からのコア性能の最適化を行えない．そこで本研究では FabScalar 上への新たな分岐予測器の実装を容易にする機構を実装することで，将来の FabScalar のより多様なプログラムへの最適なコア生成を目指す．

本研究では，将来の分岐予測器の FabScalar 上への追加実装を支援するため，多くの分岐予測器の要素技術である Global History Register(GHR) を実装する．また，GHR の設計の妥当性，分岐予測器方面からの最適化の有効性を評価するため，GHR を利用する分岐予測器として Gshare 分岐予測器 [2]・Hybrid 分岐予測器 [2] の 2 つの分岐予測器を FabScalar 上に実装し，評価する．

2 背景

2.1 ホモジニアスからヘテロジニアスへの転換とその問題点

近年，特徴の異なるプログラムやプログラム中の特徴の異なるフェーズを効率的に実行するために，構成の異なるスーパースカラコアを複数個用いたヘテロジニアスマルチコアが注目されている．ヘテロジニアスマルチコアとは，それぞれの構成の異なるコアの得意とする処理・プログラムが異なることから，それぞれのコアをプログラムの特徴にあわせて使い分けることで，計算能力の向上や消費電力の低減を目指すものである．しかし，構成の異なる各コアの構成，それぞれのコアに対する様々なキャッシュシステム，それらを相互に接続するバス網の組み合わせが膨大となるため，従来のホモジニアスマルチコアに対して，ヘテロジニアスマルチコアは設計・検証にかかる時間が膨大となり，これが実用上の大きな障害となっている．

2.2 FabScalar とその問題点

前述の問題を解決するために FabScalar というツールが提案されている．FabScalar は様々な構成のスーパースカラコアの SystemVerilog コードを自動生成するツールセットである． FabScalar のパラメータによって任意のスーパースカラコアを構成する．設計者は FabScalar のパラメー

タを変更することで，様々なプログラムの特徴に応じたスーパースカラコアを生成できる．しかし，実装されている分岐予測器が，精度の低い Bimodal という方式の分岐予測器 1 種類のため，プログラムの特徴を考慮した上で異なる分岐予測器を実装するといったような，分岐予測器方面からのコア性能の最適化を行えない．

3 FabScalar

ヘテロジニアスマルチコアは従来のホモジニアスマルチコアに比べて設計・検証にかかる時間が膨大となることが，実用上の大きな障害となっている．

この問題に対して様々な構成のスーパースカラコアの SystemVerilog コードを自動生成するツールセット，FabScalar がノースカロライナ州立大学から提案されている．FabScalar はパラメータファイルに Way 数，パイプライン段数，演算ユニット数などのパラメータを変更・記述することで，様々な構成のスーパースカラコアを自動生成する．

3.1 スーパースカラ

FabScalar の生成するコアはスーパースカラ構造を基本とする．

スーパースカラはプロセッサ性能の向上を図る手法の一つである．

図 3.1 下側にスーパースカラの概念図を示す．図 3.1 上側のような従来のシングルパイプラインに対して，フェッチや演算ユニットを複数並列に並べることによって並行して複数の命令を処理することが可能になる．

構造が複雑化し，処理能力も命令レベルの並列性に制限されるが，プロセッサのクロック周波数以上の性能を発揮することが可能となるため，

近年の汎用プロセッサのほとんどに採用されている．

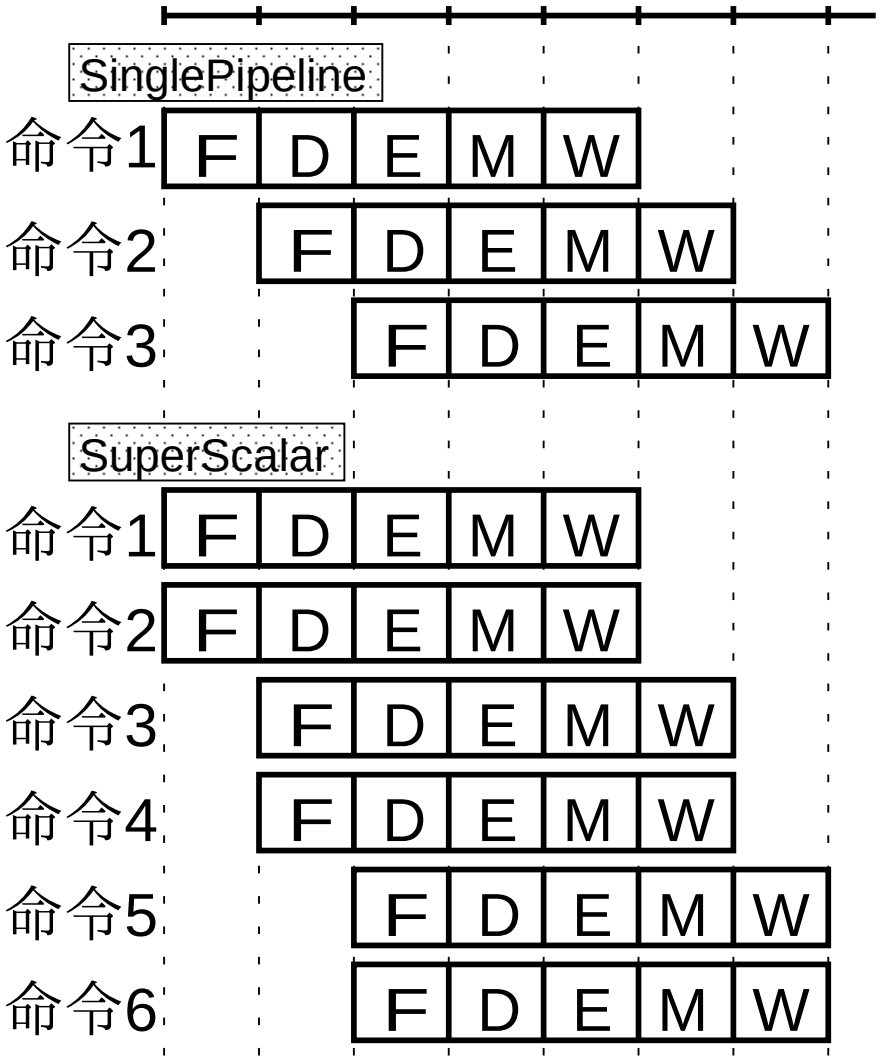

図 3.1: スーパースカラ

3.2 ヘテロジニアスマルチコア

現在の多くのプロセッサは図 3.2 左のようなホモジニアスマルチコアを採用している．ホモジニアスマルチコアは全てのコアの構造が同一であ

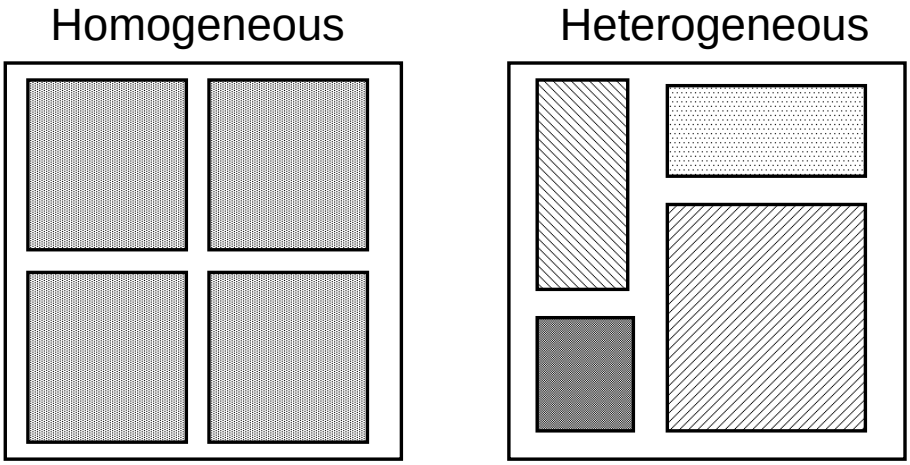

図 3.2: ヘテロジニアスマルチコア

り，それぞれのコアを汎用的に使い回せること，コア数を増やすことが容易なことを強みとする．しかし，特徴の異なる様々なアプリケーション群に対し，コア性能が過剰であったり不足であったりして，電力効率が悪い．

そこで，特徴の異なるコアを組み合わせることで電力効率の向上を図るヘテロジニアスマルチコアが注目されている．図 3.2 右にその構造を示す．ヘテロジニアスマルチコアは，状況に応じて最も適切なコアを動作させることで高性能と省電力の両立を狙う．しかし，構造の異なる複数のコア，それぞれのコアに対するキャッシュシステム，それらを相互に接続するバスシステムの組み合わせが膨大となり，従来のホモジニアスマルチコアに比べて設計・検証コストが大きく増大しており，これが実用

上で大きな障害となっている.

4 関連研究

4.1 分岐予測器

条件分岐命令について分岐方向を予測する分岐予測を行う機構である．プロセッサの性能向上に伴いパイプラインの深化などが進んだため，分岐命令によるパイプラインのストールを避け，効率的に分岐命令を含むプログラムを実行するために，近年のほとんど全てのプロセッサに搭載されている．

プロセッサの性能を向上させるためのパイプラインの深化やスーパースカラ化などによって，現代のプロセッサでは分岐命令を含む数十個の命令が投機的に実行される．分岐予測が外れると，これらの命令の実行が無駄になってしまうため，分岐予測の精度の向上が計算性能の向上に大きく貢献する．そのため，様々な分岐予測器が提案されている．

4.1.1 2 bit 飽和カウンタ

2 bit 飽和カウンタは様々な分岐予測器の構成要素として利用される．図 4.3 に 2 bit 飽和カウンタの状態遷移を示す．

Taken は分岐した，Not Taken は分岐しなかったことで，実行された分岐命令が Taken であったか Not Taken であったかによって状態を遷移させる．2 bit なので状態は 4 つであり，それぞれ Strongly Not Taken は

分岐しない確率が高い，Weakly Not Taken は確率は低いが分岐しない，Strongly Taken は高い確率で分岐する，Weakly Taken は確率は低いが分岐する，といったように予測する．

これらの４つの状態を２ bit の００～１１にそれぞれ割り当て，Taken, Not Taken を０，１に割り当てて更新する．飽和カウンタなので００をデクリメントしても００であり，１１をインクリメントしても１１である．

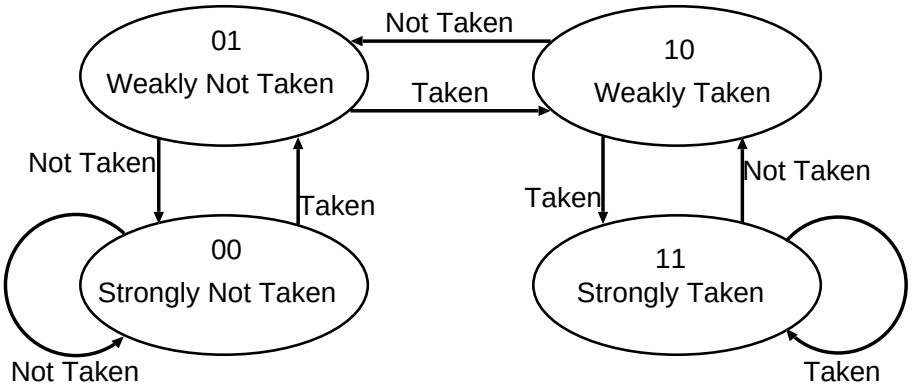

図 4.3: ２ bit 飽和カウンタ

4.1.2 Bimodal 分岐予測器

分岐命令アドレス (プログラムカウンタ) の一部のビット列を用いて２ bit 飽和カウンタの配列である Pattern History Table(PHT) を参照し，PHT の値によって当該分岐命令が分岐するかどうかを予測する．この方式は，当該分岐命令の PHT の各エントリの２ bit 飽和カウンタの記憶した過去

のふるまいという，ローカルな情報のみを使用する．実装が容易な反面，予測精度が高くないという問題点がある．現在の FabScalar に実装されている唯一の分岐予測器である．Bimodal 分岐予測器の構成を図 4.4 に示す．

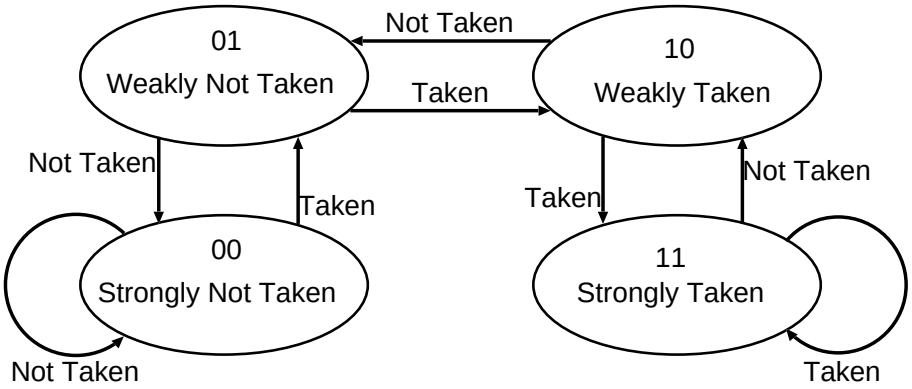

図 4.4: Bimodal 分岐予測器

4.1.3 Global 分岐予測器

図 4.5 に Global 分岐予測器の構成を示す．

Global History Register と呼ばれる，直近に実行した N 個の分岐命令の結果を格納したレジスタを用いて，2 bit 飽和カウンタの配列である PHT を参照する．この方式では Bimodal のような 2 bit 飽和カウンタの持つ過去のふるまいに加えて GHR のグローバルの情報も使用する．実装は容易であるが，それほど予測精度が高くなく，使用されることはあまりない．

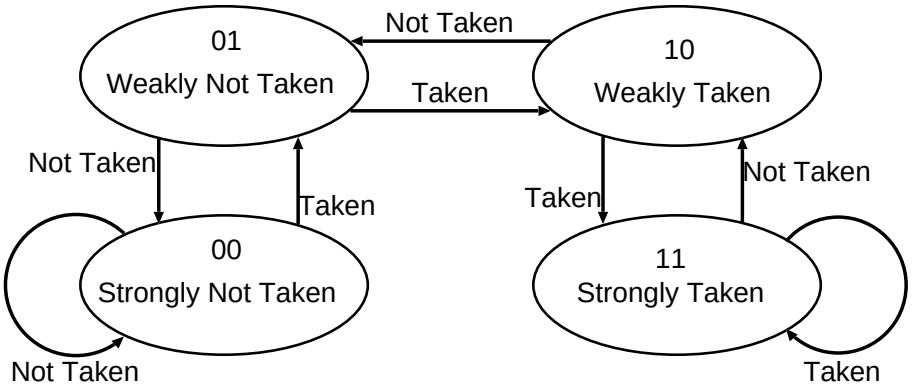

図 4.5: Global 分岐予測器

4.1.4 Gselect 分岐予測器

図**??**に Gselect 分岐予測器の構成を示す.

前述の Bimodal 分岐予測器と Gshare 分岐予測器を組み合わせたもので，GHR とプログラムカウンタをビット結合したものを用いて PHT を参照する．両方を用いることで Bimodal 分岐予測器や Global 分岐予測器より効果的に PHT のエントリを使用できるため，分岐予測精度がよい.

4.1.5 Gshare 分岐予測器

図 4.7 に Gshare 分岐予測器の構成を示す.

GHR とプログラムカウンタの XOR を用いて PHT を参照する．Gselect での PHT のインデックスには無駄が多いため，XOR を用いてハッシュ化することによってより効果的に PHT のエントリを使用できる．Gselect

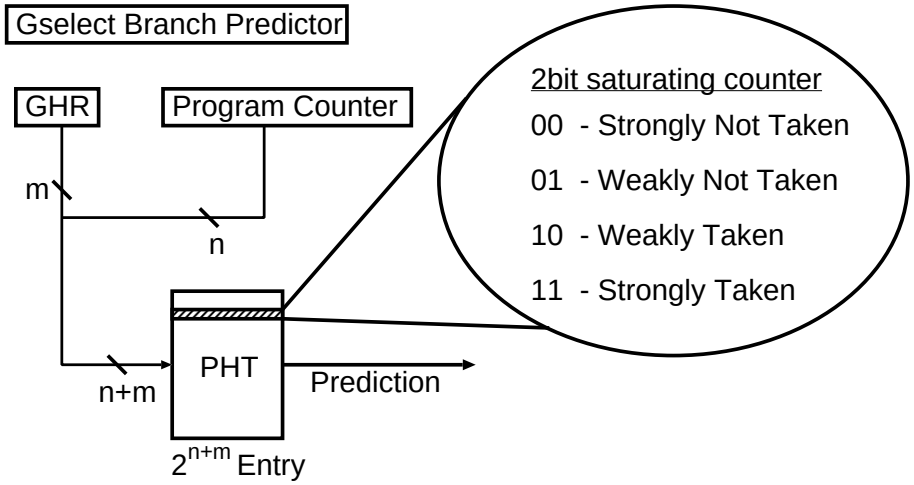

図 4.6: Gselect 分岐予測器

より若干良い予測精度を持つ.

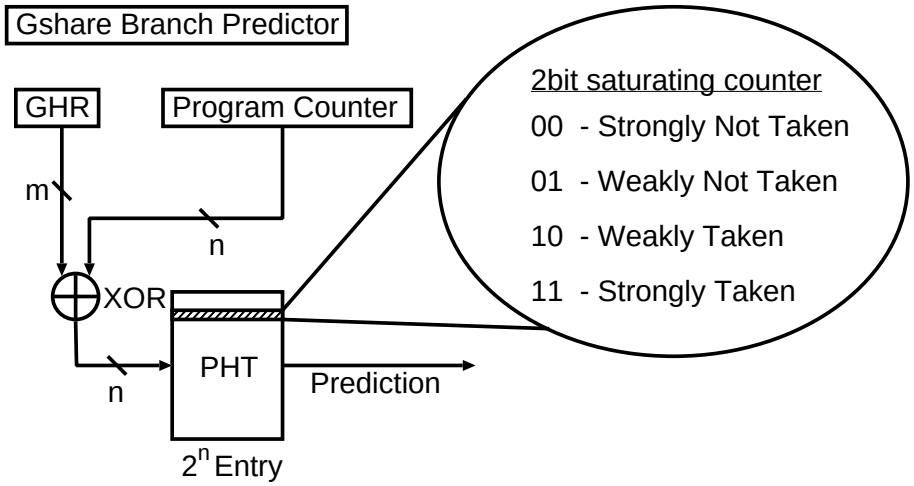

図 4.7: Gshare 分岐予測器

4.1.6 Hybrid 分岐予測器

図 4.8 に Hybrid 分岐予測器の構成を示す.

複数の分岐予測器を併用するもので，メタ予測器 (Choice PHT) を使用し，予測する分岐命令に対してより高い的中率を持つ分岐予測器の結果を採用する．それぞれの分岐予測器の方式にはそれぞれ異なる特徴と長所があること，ターゲットとなるアプリケーションごとに最適な分岐予測器が異なることから，状況に応じて分岐予測器を使い分けることでより効果的に分岐命令を予測できる．

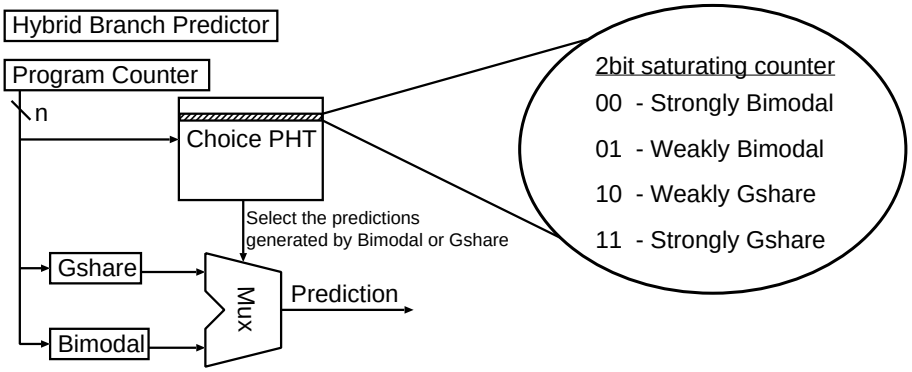

図 4.8: Hybrid 分岐予測器

5 Global History Register(GHR)

分岐予測には大きく分けて，ローカルな情報を使用するものと，グローバルな分岐パターンを使用するものがあり，後者の手法ではGHRと呼ばれる直近のn個の分岐命令の履歴を格納するレジスタが使用される．図5.9にスーパースカラプロセッサにおけるGHRの構成を示す．本研究で実装するGshare分岐予測器，Hybrid分岐予測器や最新の予測器の一つであるITTAGE[3]など，現在提案されている多くの分岐予測器はこのレジスタを利用するため，GHRはその他の分岐予測器の実装にも利用できる．

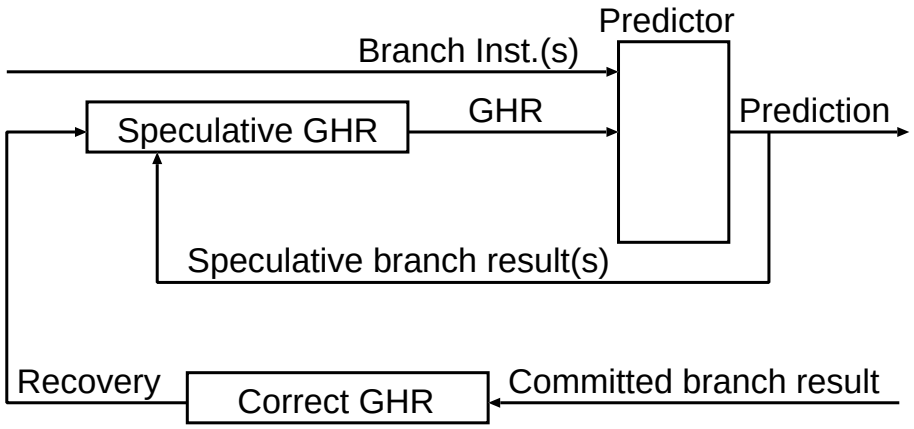

図 5.9: Global History Register

FabScalarの生成するスーパースカラプロセッサでは複数の分岐命令が同時に投機的に実行される．投機実行される分岐命令の分岐予測の性能を向上するためには，GHRを投機的に更新する必要がある．そこで，本

研究で実装する GHR は，分岐命令毎に分岐予測器 (Predictor) の結果を用いて投機的に更新する Speculative GHR(SGHR) と，コミットされた正しい分岐命令の結果を用いて更新する Correct GHR(CGHR) に分割する．分岐命令の予測には SGHR を使用し生成された予測を書き戻して更新する，CGHR は投機的な更新が誤っていた場合，すなわち予測結果が間違っていた時に SGHR を正しい状態にリカバリするために使用する．

GHR を用いた分岐予測器として，本研究では Gshare 分岐予測器と Hybrid 分岐予測器を実装する．

5.1 FabScalar への GHR 実装上の問題点とアプローチ

GHR を実装するには，GHR だけではなく GHR を活用するための周辺回路も追加する必要がある．通常，周辺回路は FabScalarno 内部仕様に強く依存するが，FabScalar はパラメータの操作によって内部仕様そのものが変化する．そこで，GHR とその周辺回路を依存するパラメータの違いにあわせて細分化し，細分化されたそれぞれが内部仕様・パラメータに適合するように実装することで FabScalar 上への汎用的な GHR の実装を図る．また，既存のマクロに加えて，‘USE_GHR という GHR とその周辺回路を管理するマクロを FabScalar に追加する．これによって GHR を使用しない場合に残留する周辺回路・配線を排除する．加えて，分岐予

測器によって利用される機構である GHR を分岐予測器の外側にモジュール化して実装することによって，分岐予測器のインターフェースの簡略化・共通化を図る．その結果，FabScalar 上に新たに分岐予測器を実装する際には GHR とその周辺回路についての仕様を把握する必要がなく，マクロ操作のみで GHR を実装することが可能になるため，GHR を使用する分岐予測器の実装が容易になる．

6 性能評価

評価には，SPEC2000INT より bzip，gap，gzip，mcf，parser，vortex の６つのベンチマークを利用し，評価する．それぞれのベンチマークを 1000 万命令実行した．図??〜図??に分岐予測器内の PHT のサイズを 256 エントリから 64k エントリまで変化させた場合の，オリジナルの FabScalar に搭載されている Bimodal および，Gshare，Hybrid 分岐予測器の評価結果を示す．Hybrid 分岐予測器は同数のエントリを持つ Bimodal 分岐予測器と Gshare 分岐予測器を併用する．縦軸は分岐予測的中率であり，数値が高いほど分岐予測器としての性能が高い．横軸は分岐予測器内の PHT のエントリ数である．Bimodal 分岐予測器はエントリ数が 4k の時に性能が飽和し，4k より大きくしても的中率に変化がなかった．8k エントリ以上では Gshare/Hybrid 分岐予測器は Bimodal 分岐予測器より結果がよく，テーブルサイズが大きくなるほど的中率が向上した．

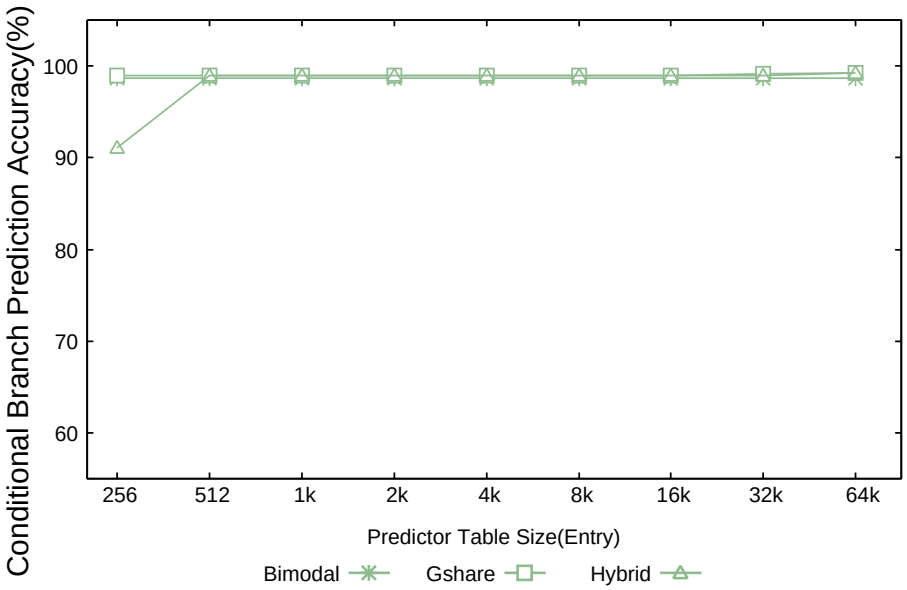

図 6.10: bzip

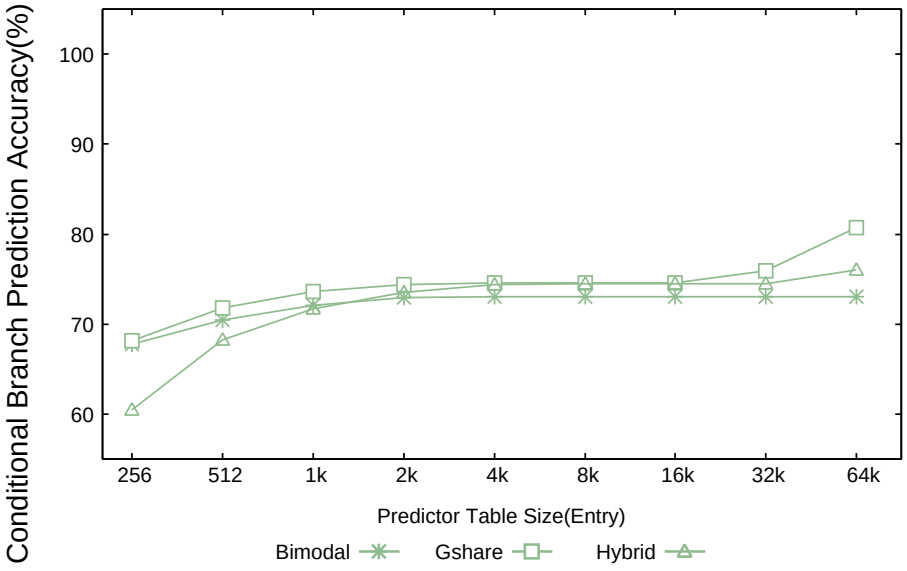

図 6.11: gap

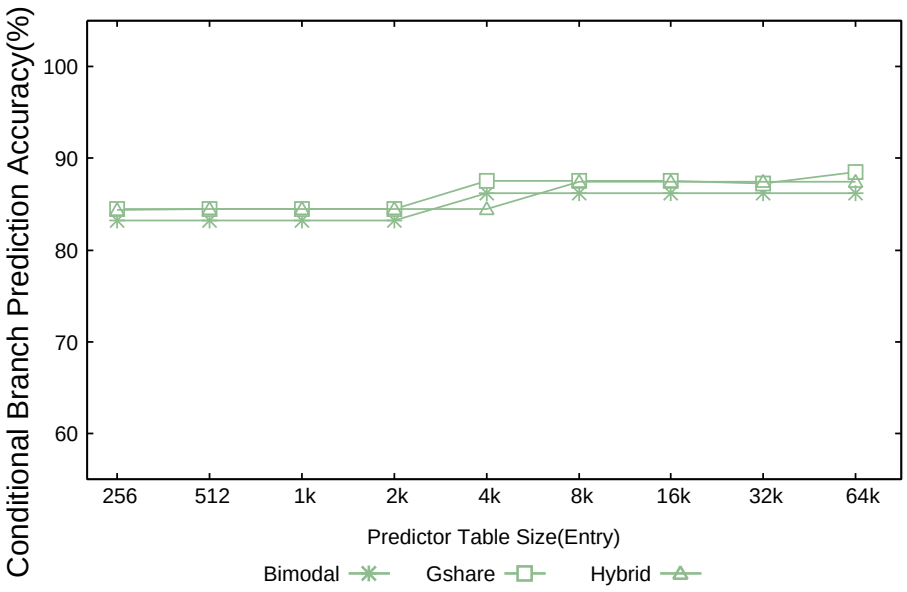

図 6.12: gzip

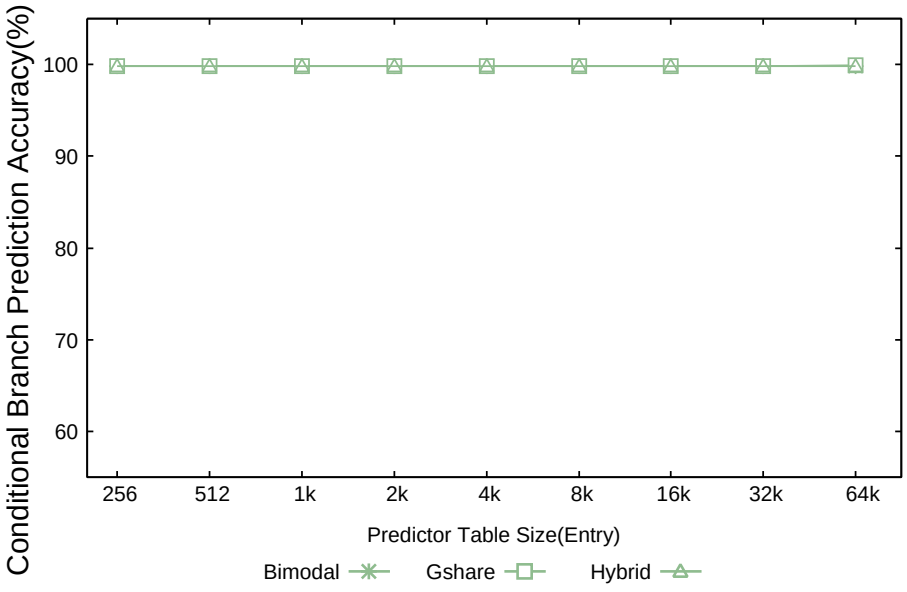

図 6.13: mcf

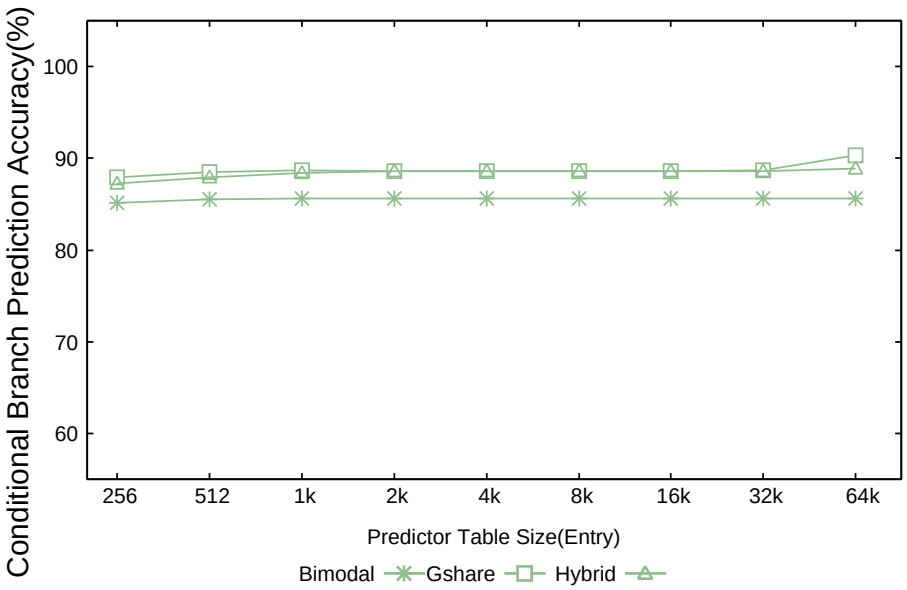

図 6.14: parser

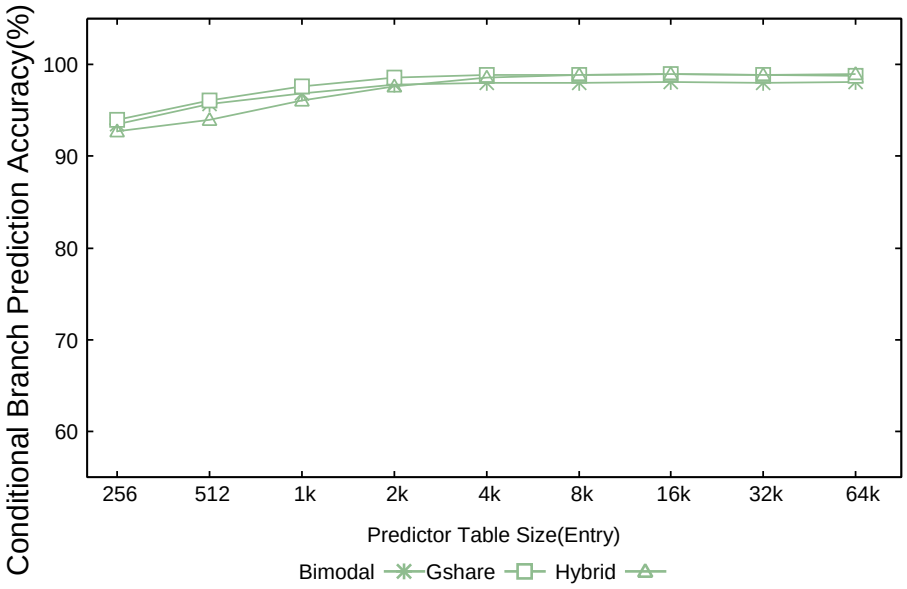

図 6.15: vortex

7 考察

評価結果より，mcf ではどの分岐予測器も予測精度に差がないため，よりハードウェア規模の小さい Bimodal 分岐予測器が適当である．また，gap では Gshare 分岐予測器が最も予測精度が高いことから，Gshare 分岐予測器を実装すべきである．これらのことからプロセッサの性能向上のために，分岐予測器の最適化の必要性が確認できた．また，GHR を利用する Gshare 分岐予測器，Hybrid 分岐予測器が従来の Bimodal 分岐予測器よりほぼ全ての評価で上回ったことから，GHR の設計が妥当であると判断できる．

8 おわりに

本研究では，FabScalar の生成するコアの分岐予測器面の最適化を図り，Gshare 分岐予測器，Hybrid 分岐予測器を実装した．また，FabScalar への将来の分岐予測器の追加実装を助ける目的で，多くの分岐予測器に採用されている機構である Global History Register(GHR) を実装した．また，これらの実装の妥当性を評価するため SPEC2000INT を使用して評価を行った．

その結果，Gshare・Hybrid 分岐予測器は従来の Bimodal 分岐予測器より高い予測精度を示したため，Gshare・Hybrid 分岐予測器の設計が妥当であると確認できた．また，これらの分岐予測器に利用される GHR の設計が妥当であること判断できる．

今後の課題として，より長いシミュレーションでの性能評価，GHR を利用するより多様な分岐予測器の実装を行うことでより多様なプログラムへ最適化された FabScalar のコア生成を図ることが挙げられる．

謝辞

本研究を進めるにあたり，ご指導を頂いた指導教員の佐々木敬泰助教，並びに近藤利夫教授，大野和彦講師に感謝致します．また、日常の議論を通じて多くの知識や示唆を頂いた計算機アーキテクチャ研究室の皆様に感謝します．最後に，事務等を担当して頂いた田中さんに感謝します．

参考文献

[1] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi and E. Rotenberg. FabScalar: Composing Synthesizable RTL Designs of Arbitrary Cores within a Canonical Superscalar Template. Proceedings of the 38th IEEE/ACM International Symposium on Computer Architecture (ISCA-38), pp. 11-22, June 2011.

[2] S. McFarling. Combining Branch Predictors. Technical report, WRL Technical Note TN-36, Digital Equipment Corporation, 1993.

[3] Andre Seznec and Pierre Michaud. A case for(partially)-tagged geometric history length predictors. Journal of Instruction Level Parallelism, April 2006.