修士論文

題目

大規模ベンチマークプログラムのための評価環境の改良と細粒度モード切換コントローラを用いた可変パイプライン段数プロセッサの評価

指導教員

近藤 利夫 教授

平成25年度

三重大学大学院 工学研究科 情報工学専攻
計算機アーキテクチャ研究室

中村 仁(412M515)

内容梗概

近年，モバイルプロセッサの分野では，性能向上に伴う消費エネルギーの増大が問題となってきており，低消費エネルギーと高性能を両立することが強く求められている．そのため，現在までに様々な低消費エネルギー化手法が提案されている．その一つとして，当研究室では，Variable Stages Pipeline(VSP) を提案している．VSP はパイプライン段数と動作周波数をプロセッサにかかる負荷に応じて動的に切換えることで，低消費エネルギーと高性能の両立を目指す手法である．VSP を効率的に用いるためには，実行中のプログラムに対して適切なパイプライン段数を予測し，動的にパイプライン段数を変更するモード切換コントローラが必要となってくる．そこで，細粒度（数百命令単位）のモード切換コントローラが提案されている．しかしながら，これまでに，試作 VSP チップに関する詳細な解析が行われていなかった．そこで，本論文では，試作を行った VSP チップを用いて詳細な評価を行う．　試作 VSP チップを用いて評価，解析を行った結果，試作チップが正常に動作することを確認し，プログラムごとに動的に閾値を割当てることにより，より効率的な実行が行えるであろうことが分かった．コントローラ自体のハードウェア量と電力評価を行った結果，コントローラのハードウェア量はプロセッサ全体の 1%程度，消費電力は平均で 2.1%程度と，十分に小さい値であると考えられる．

また，本論文では，試作チップの評価環境として用いた評価・検証用 LSI テスター Power Medusa の改良を提案する．従来の LSI テスターは，予めベンチマークプログラムを実行して作成しておいたテストパターンを用いるため，LSI テスター内の DRAM の容量の問題から数十万命令程度のベンチマークしか実行することができなかった．数十万命令程度の評価では実際のアプリケーションの様な大規模なプログラムと傾向が異なる可能性がある．そこで，LSI テスター Power Medusa を改良することで，大規模ベンチマークの実行が可能となることを示した．

Abstract

Today, mobile computers demand higher and higher performance, but increase of energy consumption becomes a crucial problem. Therefore, achieving both low energy consumption and high performance is required. Various low energy consumption and high-performance techniques have been proposed. As one of those ways in architecture level, variable stages pipeline (VSP) is proposed. VSP is a method that achieves both low energy consumption and high performance to change frequency and the number of pipeline stages according to processor workload. A mode transition controller is necesssary to use VSP efficiently, that predicts proper number of pipeline stages for a running program and dynamically changes the number of pipeline stages. The mode transition controller that changes the number of pipeline stages at fine-grained interval (tens or hundreds of cycles) is proposed. However, the detailed analysis of the fabricated VSP chip has not been performed. In this paper, a detailed analysis using the fabricated VSP chip is performed. From the evaluation and analysis results of the fabricated VSP chip, it was found that the fabricated VSP chip work correctly and VSP can execute programs more efficiently by assigning thresholds according to behavior of a running program. From the evaluation results of energy consumption, the quantity of energy consumption of the controller was 2.1% on average of the whole processor. This value is small enough.

This paper proposes improvement of the LSI tester Power Medusa which is evaluation environment of the fabricated VSP chip. The original tester uses test patterns which are generated by simulator. The total amount of test patterns are enormous so that the original tester can only evaluate benchmark programs up to several hundred thousand cycles. In many cases, evaluation results of several hundred thousand cycles are not enough to analyze and evaluate performance of a processor. This paper shows that the improved LSI tester can run more large scale benchmark programs.

三重大学大学院　工学研究科

目 次

図 目 次

表 目 次

1　まえがき

近年，モバイルプロセッサの分野では，性能向上に伴う消費エネルギーの増大が問題となってきており，低消費エネルギーと高性能を両立することが強く求められている。消費エネルギーの増加は発熱の増加やバッテリー持続時間の減少につながるためである．そのため，プロセッサの負荷に応じてプロセッサの動作を切換えることで，高性能かつ低消費エネルギーを実現する手法が検討されている．その中の手法の一つに DVFS (Dynamic Voltage and Frequency Scaling) [1] [2] と呼ばれる手法が提案されている．DVFS は，電源電圧と動作周波数が可変なアーキテクチャであり，プロセッサの負荷が高い時には電源電圧と動作周波数を高くし，負荷が低いときには性能があまり必要でないため電源電圧と動作周波数を低くすることにより，高性能かつ低消費エネルギーを実現する手法である．しかし，DVFS は将来的に消費エネルギー削減効率の低下が予想されている．なぜなら，近年 CMOS の電源電圧の低下ペースが鈍化しており，電源電圧の下げ幅が小さくなっているためである．そこで，電源電圧に依存しない低消費電力手法として，可変パイプライン段数プロセッサが提案されている．可変パイプライン段数プロセッサは電源電圧に依存しないため，将来的に効果が期待される．この可変パイプライン段数プロセッサの一手法として当研究室では，可変パイプライン段数アーキテクチャ (VSP; Variable Stages Pipeline) [3] [4] を提案している．VSP はパイプライン段数と動作周波数をプロセッサにかかる負荷に応じて動的に切換えることで，低消費エネルギーと高性能の両立を目指す手法である．具体的には，多段パイプライン構成で周波数を高くした上で高速に動作させる High Speed モードと，少段パイプライン構成で周波数を低くした上で低速で動作させるが，消費エネルギーを低減することができる Low Energy モードを用意し，この 2 つのモードを負荷に応じて動的に切換えることで低消費エネルギーかつ高性能の実現を目指している．VSP を効率的に用いるためには，実行中のプログラムに対して適切なパイプライン段数を予測し，動的にパイプライン段数を変更するモード切換コントローラが必要となってくる．そこで，細粒度（数百命令単位）のモード切換コントローラが提案されている．しかしながら，文献 [3] では，試作 VSP チップに関する詳細な解析が行われていない．そこで，本論文では，試作を行った VSP チップを用いて詳細な評価を行う．試作 VSP チップを用いて評価を行った結果，試作チップが正常に動作することを確認できた。また，プログラムごとに，動的にモード切換に用いる閾値をコント

ローラに割当てることにより，より効率的な実行が行えるであろうことが分かった．コントローラ自体のハードウェア量と電力評価を行った結果，コントローラのハードウェア量はプロセッサ全体の 1%程度，消費電力は平均で 2.1%程度と，十分に小さい値であると考えられる．

また，本論文では，試作チップの評価環境である汎用 LSI テスター Power Medusa の改良を提案する．従来の LSI テスターは，予めベンチマークプログラムを実行して作成しておいたテストパターンを用いるため，LSI テスター内の DRAM の容量の問題から数十万命令程度のベンチマークしか実行することができなかった．数十万命令程度の評価では実際のアプリケーションの様な大規模なプログラムと傾向が異なる可能性がある．そこで，汎用 LSI テスター Power Medusa を改良することで，大規模ベンチマークの実行が可能となることを示した．

2　可変パイプライン段数プロセッサ

2.1　パイプラインプロセッサ

プロセッサ内では，命令の読み込み (Fetch)，解釈 (Decode)，レジスタ読み込み (Register Read)，実行 (Execution)，結果の書き込み (Write Back) などの複数の段階からなる処理を行うことにより 1 つの命令が処理される．パイプラインプロセッサの動作を図 2.1 に示す．図 2.1 では Fetch を F，Decode を D，Register Read を R，Execution を E，Memory Access を M，Write Back を W としている．図 2.1 に示すように，それぞれの段階においての処理は異なった回路で処理されるため，1 クロックで段階ごとに並列に動作させることができる．このように段階的に分けて並列に命令を動作させるプロセッサをパイプラインプロセッサと呼ぶ．パイプラインプロセッサではパイプライン段数を分けることにより，1 クロックで動作させるべき回路は小さくなるため高周波数にすることができ，単位時間当たりに実行できる命令が増えることから高性能を実現している．しかし，パイプライン段数が増えれば，段階間で命令の処理内容を保持する装置であるレジスタが必要となるため，消費電力は増えるという欠点がある．そのため，一般的な商用プロセッサを高性能なプロセッサにするため，パイプライン段数をできるだけ増やし，高周波数で動作させたいが，単純にパイプライン段数を増やすと消費電力が大きくなるために，現在では約 10 段程度のパイプラインプロセッサが採用されている．

2.2　可変パイプライン段数アーキテクチャ

可変パイプライン段数プロセッサとは，パイプライン段数と周波数を負荷に応じて切換えることにより，負荷が高い場合には高性能を発揮させ，負荷が低い場合には性能を低下させる代わりに消費電力を抑えるという手法である．可変パイプライン段数プロセッサには，本研究室で提案している VSP と，パイプラインステージ統合 (PSU:Pipeline Stage Unification) [5] [6] [7] [8]，DPS (Dynamic Pipeline Scaling) [10] などの方式が提案されている．また，これらの可変パイプライン段数プロセッサはパイプライン段数に対して以下のような特徴を備えている．

- 多段パイプライン段数時，性能や動作はパイプライン段数が可変ではないプロセッサと同等である．

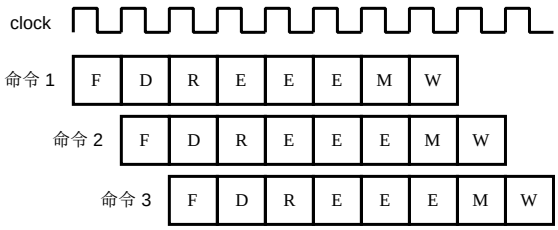

図 2.1: パイプラインプロセッサの動作

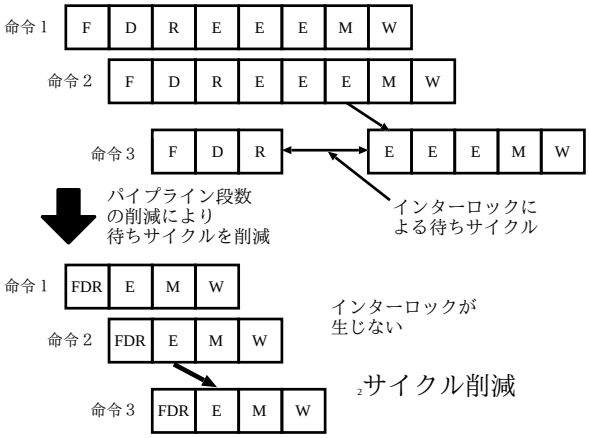

図 2.2: データ依存待ちサイクルの低減

- 少段パイプライン段数時，パイプラインレジスタへのクロック供給の必要がなくなるため，クロックドライバの消費エネルギー低減に繋がる
- 少段パイプライン段数時，データ依存による待ちサイクルの低減，分岐予測ミスペナルティ低減に繋がる（図 2.2，図 2.3）．図 2.2，図 2.3 ともに水平方向が時間軸である．

図 2.2 において，多段パイプライン段数時の状態で命令を実行中に命令 2 と命令 3 に依存関係がある場合には，命令 3 の実行を命令 2 の処理結果が得られるまで待つ必要がある．しかし，少段パイプライン段数時の状態では命令に依存関係がある場合において，実行に必要となるクロック数が 1 クロックであるため待ち時間は発生しない．図 2.3 においても，少

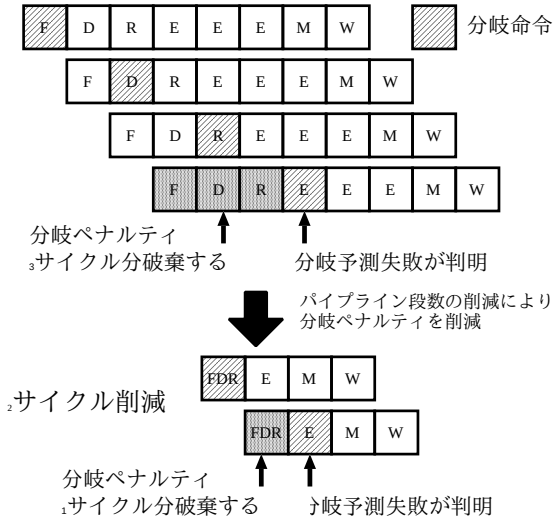

図 2.3: 分岐予測ミスペナルティ低減

段パイプライン段数時ではパイプライン段数が少ないために分岐予測ペナルティ削減ができる．

VSP などの可変パイプライン段数プロセッサでは，パイプライン段数を 2 種類以上へ切換可能であるが，今後，説明を簡潔にするために，現在 VSP で用意されている高速モードと低消費電力モードの 2 通りのパイプライン段数だけを用いて説明を行う．高速モードと低消費電力モードは図 2.4 の構成となっている．高速モードはプロセッサの性能を最大限に活かすために，パイプライン段数が多く，高周波数のモードである．低消費電力モードは消費エネルギーを抑えるために，パイプライン段数が少なく，低周波数のモードである．本節では，高速モードのパイプライン段数を 9 段，低消費電力モードのパイプライン段数を 3 段とする．

モード別のアーキテクチャの様子を図 2.5，図 2.6 を用いて述べる．ここでは，高速モードにおけるパイプラインの四段分だけを取り出して説明を行う．図 2.5，図 2.6 を比較すると，高速モードで存在していたパイプラインレジスタ B，D を，低消費電力モードでは使用しないため，その分レジスタへの電力供給を減らすことができ，低消費電力となる．また，高速モードで存在していた論理回路 A と B が，低消費電力モードでは統合されている．ここで周波数を下げてやると，この統合された論理回路が 1 クロックで動作するようになり，データ依存待ちサイクル時間の低減と分岐予測ミスペナルティの低減が実現される．

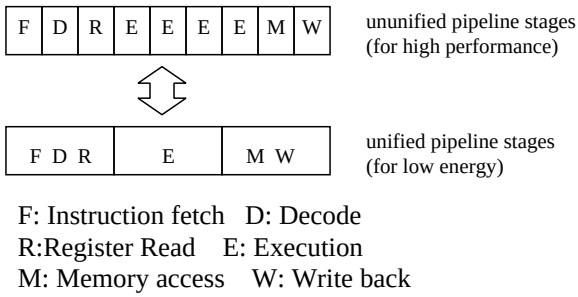

図 2.4: モード別パイプライン段数

2.3 パイプライン統合手法

可変パイプライン段数プロセッサでは，パイプライン段数を変更できるようにするため，本来はD-FFだけで構成されているパイプラインレジスタにマルチプレクサを追加し，データを次の回路へ流すのか，D-FF(Delay Flip Flop) で保存するのかを選択するようにしている．図2.7はパイプラインレジスタを使用するかどうかを選択する回路である．マルチプレクサ MUX に制御信号を与えることで選択を行うことができる．高速モードではD-FF を使用し，パイプライン段数を増やし，周波数を上げることで高性能を実現する．低消費電力モードではD-FF は使用せず，周波数を下げることにより低消費電力を実現できる．

2.4 VSP

VSP は，PSU と同様にパイプライン段数を切り換える手法であるが，VSP ではパイプラインレジスタの一部に LDS-cell (Latch D-FF Selector-cell) という特殊なセルを用いグリッチと呼ばれる無駄な信号の変化の発生を抑制することでさらに消費電力を削減している．

また，VSP では多段のパイプライン構成時を High Speed(HS) モード，少段のパイプライン構成時を Low Energy(LE) モードと呼んでおり，その特徴を以下に示す．

HS モード：

- LDS-cell は通常のパイプラインレジスタとして動作する．
- gshare 分岐予測器を搭載しており，レジスタ間接を除く無条件分岐は分岐予測器において 100%の分岐予測が可能である．

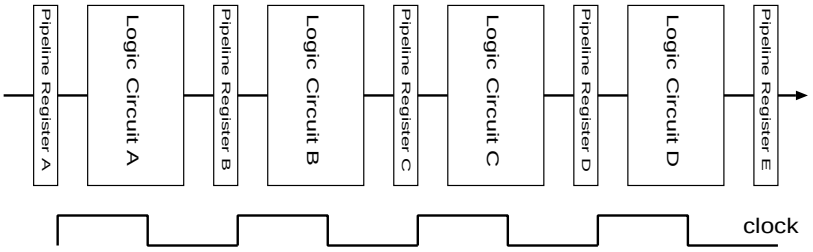

図 2.5: 高速モード

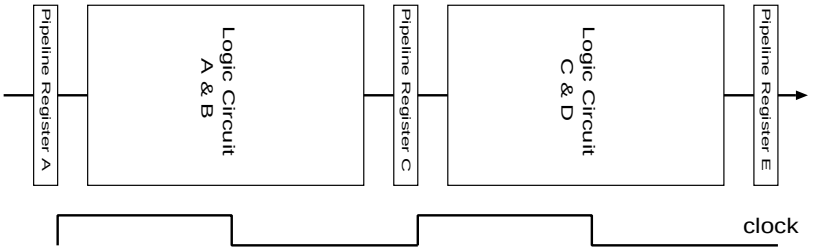

図 2.6: 低消費電力モード

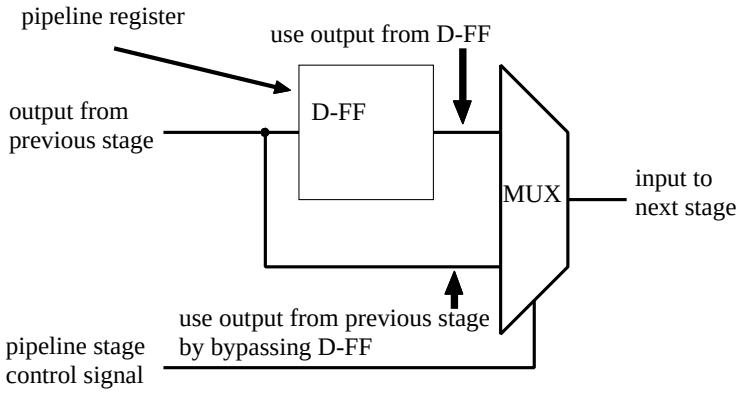

図 2.7: D-FF+MUX

- インターロックと演算結果のフォワーディング機構を搭載している.

LE モード：

- LDS-cell はグリッチの緩和を行う D ラッチとして動作する.
- 遅延分岐，遅延ロード，フォワーディングによって制御依存やデータ依存によるインターロックが発生しない.
- 分岐予測器は使用しないので停止させている.
- 分岐予測器やバイパスされて使用しなくなったパイプラインレジスタのクロックを止めることでパイプラインレジスタで消費されるエネルギーを削減することが出来る.

3　モード切換コントローラ

VSP を効率的に使用するためには，実行中のプログラムに対して最適なパイプライン段数を予測し，動的にパイプライン段数を変更する切換コントローラが必要となる．本章では，VSP に用いられるモード切換コントローラについて述べる．

3.1　従来のモード切換コントローラ

可変パイプライン段数プロセッサのためのコントローラとして，Yao らによりコントローラが既に提案されている [5]（以下,「従来のコントローラ」と呼ぶ）本節では従来のコントローラについて概括した上で，その問題点を指摘する．

このコントローラは 10 万命令単位（以下,「フェーズ」と呼ぶ）でパイプライン段数の切換えを行っている．また，このコントローラは 1 フェーズで実行されたプログラムの位置を示す signature と，その signature の値に対して最適なパイプライン段数を記憶しておく履歴表からなる．

一般にプログラムは数万命令程度で挙動を取得すると，似たような挙動を行うフェーズが現れるという特徴があることが文献 [9] で述べられている．これは，プログラムの同じ部分をアクセスする場合が多々存在し，アクセスされた部分が同じであれば挙動が似ているためである．従来のコントローラでは，異なるフェーズ間でも実行された位置が類似している場合には，同じような挙動をとるという特徴を活かしている．アクセスされた位置が類似しているフェーズごとに，最適なパイプライン段数を検出し，その後のフェーズでは検出した最適な段数で動作させるという仕組みになっている．

従来のコントローラでは，履歴表を用いた予測を行うことで予測精度の高いパイプライン段数の制御を行っている．しかし，高い精度でフェーズを検出するためには 10 万命令程度プログラムを実行する必要がある．その結果，切換の粒度が粗くなるため，細かな負荷の変動には追従できず，可変パイプライン段数プロセッサの性能は十分に発揮できない．さらに，それぞれの signature に対し，それぞれのモードで実行を行い，モードごとに比較を行った結果を次回からのモードに用いるため，性能の悪いモードで 1 度は実行しなければならない．また，多数のレジスタからなる履歴表を持つためハードウェア規模は大きなものとなる．

3.2　細粒度モード切換コントローラ

本節では，VSP で用いられている細粒度モード切換コントローラについて述べる．

3.2.1　細粒度切換えについて

本節では細粒度に切換えを行った場合の利点について述べる．従来のコントローラでは 10 万命令に 1 度の割合でモードの切換を行っている．しかし，10 万命令に 1 度の切換えでは細かい負荷の変動がある場合に対応できない．この様子を図 3.8 に示す．図 3.8 では縦軸がプロセッサに掛かる負荷，横軸が実行時間である．図 3.8 の A では負荷の変動が細かくないため，切換周期が粗粒度 (Coarse-Grain) の場合にも負荷に応じたモード切換えが可能である．しかし，図 3.8 の B,C の場合には切換周期よりも負荷の変動が細かいため負荷に対応した切換えを行うことができない．そのため，細かい負荷の変動にも対応できるよう細粒度 (Fine-Grain) の切換コントローラを用いる．細粒度にすることにより，粗粒度では対応できていなかった図 3.8 の B,C のような細かい負荷の変動にも対応できることになる．

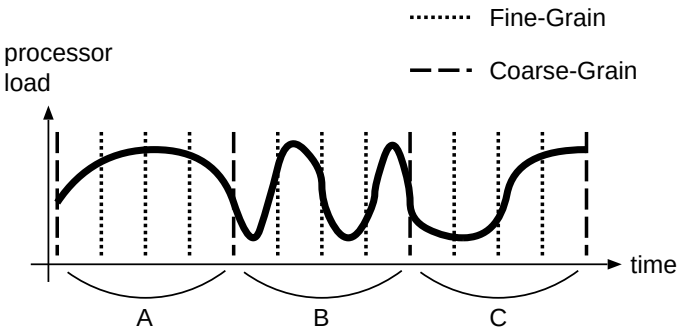

図 3.8: 切換周期における違い

3.2.2　細粒度コントローラについて

本節では細粒度切換 VSP コントローラの仕組みについて述べる．細粒度切換 VSP コントローラはプロセッサの負荷に応じてモードを切換える

手法である．負荷が高く高速な処理が必要だと考えられる場合には高速モードに，負荷が低く高速な処理が必要ない場合には低消費電力モードに切換えることで，高性能かつ低消費エネルギーが実現できる．負荷の取得方法が誤っていた場合，最適でないモードへ切換えを行うことがあるため，コントローラを設計する上で，負荷の動的検出方法は重要である．本研究では，Verilog HDL によって現在実装されているインオーダプロセッサ VSP のための切換手法である文献 [3] の切換手法を使用する．このコントローラは IPC (Instruction Per Cycle) に着目してパイプライン段数を動的に切換える手法である．一般に，プロセッサにかかる負荷とキャッシュミスおよび分岐予測ミスの回数には強い相関がある．つまり，キャッシュミスや分岐予測ミスが多発するようなプログラムでは深いパイプライン段数を有する HS モードはその性能を最大限に発揮することができず，LE モードで実行した方が電力効率が良くなる．インオーダ実行型プロセッサにおいては，キャッシュミスおよび分岐予測ミスがそのまま IPC に影響するため，IPC を監視することで 2 つの指標を同時に監視することが可能である．そこで，提案されている切換手法では，最新の 32 サイクルで完了した命令の合計数を閾値と比較してパイプライン段数を変更する．しかしながら，文献 [3] の切換手法を用いた現在の VSP の問題点として，LE モードでは分岐予測ミスが発生しないため IPC がキャッシュミスにのみ影響され，その結果，LE モードから HS モードへの移行が適切に行われない場合がある，という点が挙げられる．また，現在の VSP では消費電力を削減するために LE モードでは分岐予測器を停止しているため，LE モード時に分岐予測ミス回数を監視することはできない．そこで，分岐予測ミスが多く発生するフェーズでは比較的分岐命令の数が多いという傾向を利用し，LE モードから HS モードへの移行には最新 32 サイクルに完了した命令の合計数と分岐命令の合計数を閾値と比較してパイプライン段数の変更を行っている．文献 [3] の切換手法を用いたモード切換コントローラのブロック図を図 3.9 に示す．HS モードにおいては，完了した命令 (completed instruction) の合計数のみを条件として LE モードへの移行が行われる．反対に，LE モードでは，完了した命令の合計数とデコードされた分岐命令の合計数との両方を条件として HS モードへと移行する．

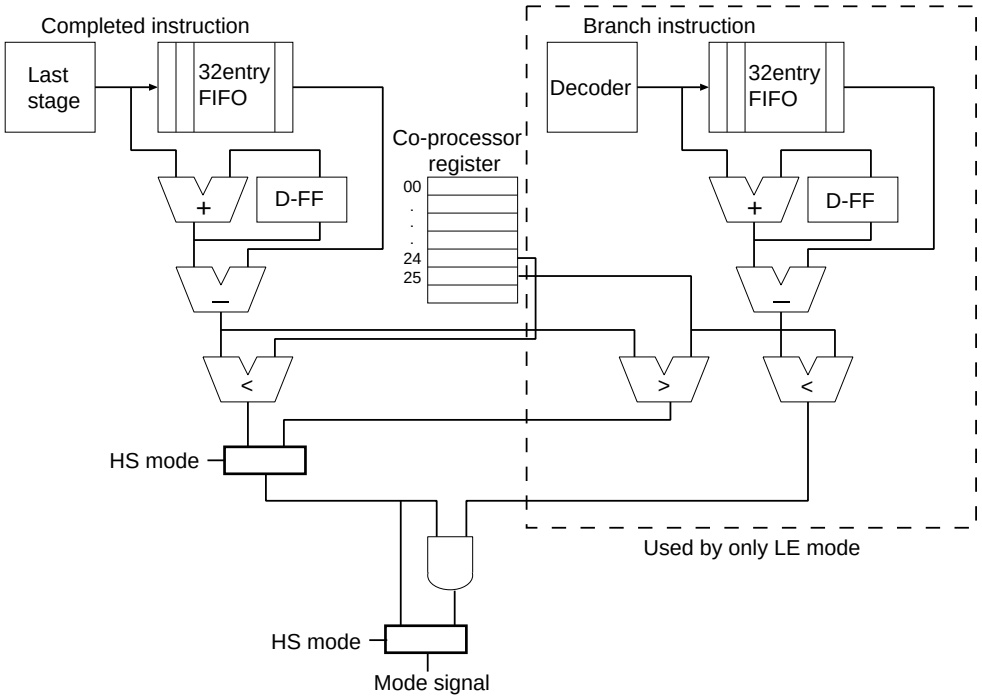

図 3.9: モード切換コントローラのブロック図

4　VSP プロセッサのチップ試作

本章では，試作を行った VSP チップについて述べる．

従来の VSP では，一意に定めた閾値を用いてチップを試作することを想定しており，モード切換に用いる閾値を動的に変更できないという問題点があった．現在までに，提案コントローラの最適な閾値の組合せをシミュレーション評価によって求めているが，実行時間が膨大となるため，シミュレーション評価では多くのベンチマークプログラムを実行できない．また評価環境の制限から，大規模なプログラムを実行できないため，実チップを用いて大規模なプログラムを実行した場合，最適な閾値が異なる可能性がある．そこで，これまでハードウェアで実装され固定化されていた閾値をプログラム中でソフトウェア的に変更できるように改造を行った．具体的には，プログラムの初期化ルーチンで，コプロセッサレジスタに閾値として用いたい値を書き込み，モード切換コントローラにこの値を渡すことで閾値の変更が可能となる．これにより，実チップ上においても閾値を変更することが可能となり，より柔軟な制御を行うことができる．

ROHM $0.18\mu m$ CMOS プロセスを用い，提案している切換手法を使用したモード切換コントローラ搭載の VSP チップの試作を行った．VSP の実装に際して，MIPS R3000 互換プロセッサを用いた．HS モードのパイプライン段数は 7 段，LE モードのパイプライン段数は 3 段となっている．VSP を適用していないベースプロセッサの試作は行っていないが，ベースプロセッサのフィジカル合成を行うことで，VSP プロセッサの追加に必要なハードウェア量を求めた．ベースプロセッサのトランジスタ数は，456,022 個であり，VSP プロセッサのトランジスタ数は 492,742 個であった．したがって，VSP の実装には，追加で 8%(36,720 トランジスタ) のハードウェアが必要となる．モード切換コントローラの実装に要したトランジスタ数は，36,720 個の内 4705 個であった．図 4.10 に試作 VSP プロセッサのチップ写真を示す．

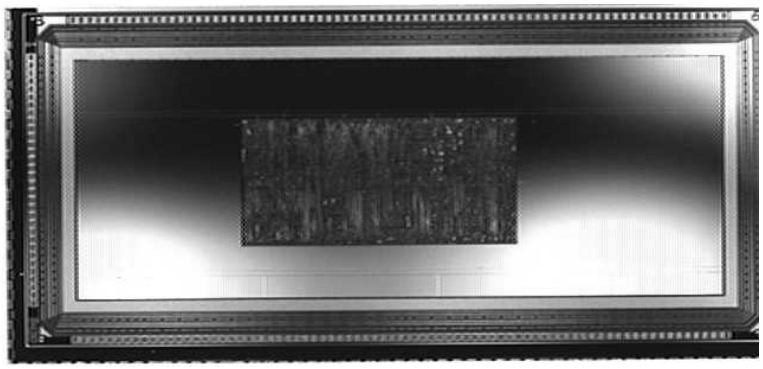

図 4.10: VSP プロセッサのチップ写真

5　性能評価

5.1　評価環境

試作したチップを用いて，実行時間と消費電力の評価を行った．試作VSP チップは，HS モード時で 100MHz，LE モード時で 25MHz で動作するように設計されているが，本論文では評価環境の制限から HS モード時の動作周波数を 35MHz，LE モード時の動作周波数を 8.75MHz とした．LSI テスター Power Medusa にて試作チップの動作テストを行い，Agilent E3640A を用いて電力測定を行った．ベンチマークプログラムには，MiBench で配布されているものの中から，文字列をクイックソートする quick sort，文字列を検索する string search，整数の 2 乗根を求める int sqrt，long 型の変数中で 1 のビット数を数える bit count を用いた．ベンチマークで使用するデータは，評価環境の制限から数十万命令程度で終了するように調整した．

5.2　評価結果

試作 VSP チップに搭載されている細粒度モード切換コントローラの有効性について検証するため，ベンチマークプログラム毎に各命令の実行時間の内訳を求め，プロセッサに掛かる負荷の特徴を求めた．ベンチマークプログラム毎の各命令の実行時間の内訳は，HS モード固定でベンチマークプログラムを実行した場合と LE モード固定でベンチマークプログラムを実行した場合それぞれにおいて求めた．HS モード固定，LE モード固定でベンチマークプログラムを実行した場合の各命令の実行時間の内訳を図 5.11 に示す．ラベル “Add&sub”, “Mult”, “Bit”, “Compare” , “Shift” は各算術命令の割合を表す．具体的には，“Add&sub” は加減算命令, “Mult” は乗算命令, “Bit” は論理演算命令, “Compare” は比較命令 , “Shift” は左右へのシフト命令を示す．また，“Cond.Branch” は条件付き分岐命令, “Branch” は無条件分岐命令, “Load” はデータの読み込みを行うロード命令, “Store” はデータの書き込みを行うストア命令をそれぞれ表す．“Stall” は分岐命令の分岐先の予測に失敗したことによる分岐予測ミスが原因となって発生したストールを除外したストールを表す．“Bpmiss Stall” は分岐予測ミスが原因となって発生したストールを表す．図 5.11, 後述の図 5.12 を用いて分岐命令数と分岐予測ミスの関係性について考察

することで，LE モードにおいて，モード切換に最新の 32 サイクルの分岐命令を用いることが妥当かどうか判断を行う．

消費エネルギーと実行時間の評価結果を図 5.12 に示す．各グラフの左側の縦軸は LE モード固定でベンチマークプログラムを実行した場合の結果を 1 として正規化した実行時間を表す．各グラフの右側の縦軸は HS モード固定でベンチマークプログラムを実行した場合の結果を 1 として正規化した消費エネルギーを表す．各グラフの横軸はモード切換に用いる閾値の組み合わせを示す．これらの閾値は，上から順に，最新の 32 サイクルに発生した分岐命令の総数，HS モードにおいて最新の 32 サイクルで完了した命令数，LE モードにおいて最新の 32 サイクルで完了した命令数とそれぞれ比較される．横軸の HS と LE は VSP を HS モード，LE モード固定でそれぞれ実行させた場合の実行結果である．横軸の HS MCU は VSP を細粒度モード切換コントローラを用いて，HS モード固定で実行させた場合の実行結果である．各グラフの右から左に行くにつれ，HS モードでの実行サイクル数が増え，LE モードでの実行サイクル数は減るように閾値の組み合わせを並べている．

図 5.12 よりベンチマークの実行時間が増えるにつれ，消費エネルギーが減少する傾向がある．性能があまり必要のないときには，消費エネルギーを削減し，性能が求められるようなときには，高性能を発揮させることができれば，実行時間の増加を抑えながら，消費エネルギーを大きく削減することができる．quick sort（図 5.12(A)）では，ストールの割合がすべてのベンチマークプログラムの中で最も低かった．また，プログラムのデータ依存が弱いため，HS では，LE と比較して実行時間が 73%減少した．この値は 4 つのベンチマークプログラムの中でも最大である．bit count（図 5.12(D)）では,HS では，LE と比較して 49%の実行時間の減少が見られた．quick sort（図 5.12(A)）では，“Cond. Branch” の割合は全てのベンチマークプログラムの中で最も高く，Branch prediction miss / Stall の割合が約 69 %と最も高かった．その他のベンチマークプログラムに関しては，“Cond. Branch” の割合は quick sort よりも低く，Branch prediction miss / Stall の割合も低かった．よって，多くの条件分岐命令を持つプログラムの Branch prediction miss / Stall の割合は高くなる可能性がある．分岐命令の合計数を監視することによって，LE モードにおいて，HS モードへ以降した際のことを考慮した IPC を得る助けになると考えられる．LE モードにおいて，HS モードへ移行した際のことを考慮した IPC を得ることができれば，HS モードへと移行した際に分岐予測ミ

スが発生し，IPC が低下する事態を防ぐことができる．

VSP はモード切換コントローラを用いることで HS と比較して，消費エネルギーを削減することができる．LE において，HS と比較して，平均 44%の消費エネルギーを削減することができた．また，DVFS は供給電圧を細粒度で下げることができないが，VSP は細粒度モード切換コントローラを用いることで短いプログラムを最適化することができる．さらに，VSP と DVFS を同時に用いることで，電源電圧とパイプライン段数が可変となり，消費エネルギーの更なる効率化が可能であると考えられる．

一般にプログラムを実行させる際，バッテリ残量を気にしなくてもよいときは，実行時間が短くなる閾値に設定することで，高性能を引き出すことができる．その一方で，バッテリ残量が少ないときには，閾値を消費エネルギーの低くなるものに設定することで，バッテリの駆動時間を延ばすことができるようになる．また，細粒度の負荷の変動に合わせてモード切換ができれば，実行時間の増加を抑えつつ，消費エネルギーを大きく削減できると考えられる．

試作 VSP チップを用いて，コントローラ自体の電力評価を行った結果，コントローラの消費エネルギーは平均 2.1%程度と，十分に小さい値であると考えられる．コントローラ自体のハードウェア量の評価を行った結果，コントローラのハードウェア量は，4705 トランジスタとプロセッサ全体の 1%程度であった．この値は，従来コントローラの実装に 50,000 トランジスタ程度必要なことと比較しても，十分に小さいと思われる．

5.3 詳細評価と改良手法

5.3.1 追加電力評価

前述の評価結果では，4 つのベンチマークプログラムで閾値を揃えて実行していなかったため，プログラム毎に傾向が異なるかどうかが説明できていなかった。そこで，モード切換に用いる閾値の組み合わせを 4 つのベンチマークプログラムで揃えて変化させた場合に，実行するプログラムによって実行時間と消費エネルギーの傾向が異なるということを示すため，試作 VSP チップを用いて実行時間と消費エネルギーの追加評価を行った．評価環境については前節と同一のものを使用した．追加評価結果を図 5.13 に示す．実行時間は，LE モード固定でベンチマークプログラムを実行した場合の結果を 1 として正規化され，消費エネルギーは，HS

モード固定でベンチマークプログラムを実行した場合の結果を 1 として正規化されている．各グラフの横軸はモード切換えに用いる閾値の組み合わせを示す．これらの閾値は，左から順に，最新の 32 サイクルに発生した分岐命令の総数，HS モードにおいて最新の 32 サイクルで完了した命令数，LE モードにおいて最新の 32 サイクルで完了した命令数とそれぞれ比較される．横軸の HS と LE は VSP を HS モード，LE モード固定でそれぞれ実行させた場合の実行結果であり，HS MCU は VSP を細粒度モード切換コントローラを用いて HS モード固定で実行させた場合の実行結果である．

全体の傾向としては，閾値の増加に応じて実行時間が増加し，消費エネルギーが減少する傾向がある．これは，基本的には閾値が増加することで，VSP が LE モードで動作するサイクル数が増加し，HS モードで動作するサイクル数が減少するためである．閾値を与えてモード切換を行った 4 つのプロット点に関して，quick sort を除いた 3 つのベンチマークプログラム（図 5.13(F)，図 5.13(G)，図 5.13(H)）では，平均して 150 サイクル以内に 1 回のモード切換が行われているが，quick sort（図 5.13(E)）では，平均して約 400 サイクルに 1 回のモード切換と，モード切換えの頻度が他の 3 つのベンチマークよりも低頻度であった．これは，quick sort はデータ依存が弱く，HS モード時の ipc が高くなる傾向にあるため，LE モードへの切換条件をあまり満たさないからだと考えられる．追加評価により，プログラムにより実行時間と消費エネルギーの傾向が異なるため，最適な閾値の組み合わせが異なるであろうことが分かった．今後は，プログラム毎に最適な閾値を決定する手法の提案が必要である．

5.3.2 分岐命令数による分岐予測ミスの分布

分岐命令数と分岐予測ミス発生回数の関係性を検証することによって，細粒度モード切換コントローラの有効性を証明する．具体的には，HS モード固定で各ベンチマークプログラムを実行した場合の分岐予測ミスの発生数とストール率を，分岐予測ミスが発生した時点の過去 32 サイクルの分岐命令の数で分類する．分類結果を表 5.1〜表 5.4 に示す．表 5.1〜表 5.4 のストール率 (32cycle) は，過去 32cycle の分岐命令数毎のストールの割合を，ストール率（全体）はプログラム全体でのストールの割合をそれぞれ示す．いずれのベンチマークプログラムでも分岐命令数が 10 以上の場合，分岐予測ミスは発生しなかった．分岐予測ミスが最も発生する分岐命令数は，quick sort，string search，int sqrt，bit count でそれぞれ 5,

表 5.1: 過去 32 サイクルの分岐命令数に応じた
分岐予測ミス発生回数とストール率 (Quick sort)

	Quick sort		
	予測ミス (回)	ストール率 (32cycle)(%)	ストール率 (全体)(%)
0	0	30.8	3.2
1	14	25.2	7.7
2	99	30.8	24.0
3	28	25.7	13.4
4	43	1.6	11.1
5	364	1.8	17.5
6	287	15.1	12.8
7	104	17.1	7.1
8	20	19.6	3.1
9	2	19.5	0.2
10	0	0.0	0.0
計	961		100.0

表 5.2: 過去 32 サイクルの分岐命令数に応じた
分岐予測ミス発生回数とストール率 (String search)

	String search		
	予測ミス (回)	ストール率 (32cycle)(%)	ストール率 (全体)(%)
0	0	42.0	0.6
1	0	41.9	0.4
2	9	32.2	1.3
3	13	48.8	14.7
4	66	37.8	80.9
5	84	25.3	1.3
6	36	22.0	0.5
7	12	22.2	0.2
8	3	10.4	0.0
9	2	11.8	0.0
10	0	0.0	0.0
計	225		100.0

表 5.3: 過去 32 サイクルの分岐命令数に応じた
分岐予測ミス発生回数とストール率 (Int sqrt)

	Int sqrt		
	予測ミス (回)	ストール率 (32cycle)(%)	ストール率 (全体)(%)
0	0	44.4	25.2
1	0	48.1	3.4
2	359	36.9	24.7
3	12	46.6	5.1
4	25	39.5	25.0
5	84	32.3	2.9
6	31	40.5	13.1
7	13	16.1	0.3
8	3	18.2	0.2
9	2	40.4	0.1
10	0	0.0	0.0
計	529		100.0

表 5.4: 過去 32 サイクルの分岐命令数に応じた
分岐予測ミス発生回数とストール率 (Bit count)

	Bit count		
	予測ミス (回)	ストール率 (32cycle)(%)	ストール率 (全体)(%)
0	0	54.1	68.9
1	0	44.1	5.3
2	7	86.4	2.8
3	12	6.5	0.2
4	760	32.9	5.4
5	33	41.0	6.7
6	12	88.6	2.7
7	8	1.9	0.1
8	4	44.1	5.2
9	2	28.6	2.6
10	0	0.0	0.0
計	838		100.0

5，2，4 となった．quick sort は，他のベンチマークと比べても，分岐命令数が多くみられるときの分岐予測ミスの発生数が多かった．string search は，分岐命令数 5 で分岐予測ミスが最も発生し，5 から離れる毎に分岐予測ミスの発生数は徐々に少なくなる．int sqrt と bit count では，全体の 6 割以上の分岐予測ミスが同一の分岐命令数で発生している．ストール率については，全体に大きな偏りが見られた．32cycle，全体ともに，分岐命令数が少ないところで，ストール率が最大になるという傾向が見られた．また，一方で分岐命令数が 7 以上観測される箇所で，ストール率の減少が見られた．特に，int sqrt と bit count について，分岐命令数 0 での全体のストール率が最大となっている．このように，ストール率が高くなっているところを HS モードではなく LE モードで実行させることができれば，更なる消費エネルギーの削減が可能になると考えられる．ストール率に基づいたモード切換コントローラの改良手法について，次節で述べる．

5.3.3 モード切換コントローラの改良手法と評価

前述のように，分岐命令数が 0，1，2 のときのストール率が高く，HS モードで実行した場合，実行時間，消費エネルギーが余分に必要となる可能性があると考えられる．そこで，本節では，モード切換コントローラの改良を提案する．既存のコントローラは，LE モードから HS モードへの切換の指標として，最新の 32cycle の完了命令数と分岐命令数を用いている．提案手法では，プログラム中のストール率の高くなる箇所の HS モードでの実行を避けるため，既存の条件に加え，分岐命令数が 2 より多いかどうか，という条件を加える．この条件によりストール率の高いところを LE モードで実行させることができると考えられる．改良手法のブロック図を図 5.14 に示す．

また，提案手法をハードウェア記述言語 Verilog HDL を使用して実装し，シミュレーション評価を用いて，実行時間と消費エネルギーを求めた．評価結果を図 5.15 に示す．左側のグラフは実行時間，右側のグラフは消費エネルギーを表す．各グラフは改良前のコントローラの評価結果を 1 として正規化されている．各グラフの横軸は閾値の組み合わせを表す．閾値の組み合わせは前節と同じものを用いた．

4 つのベンチマークプログラムの中でも分岐命令数 2 以下でのストール率が低かった string search では，最大でも 2.1%程度しか消費エネルギーを削減することができなかった．その一方で，分岐命令数 0 のときの全体

のストール率が 68.9%と最も高かった bit count では，消費エネルギーを最大 23.0%削減することができた．平均では，4.3%実行時間が延びてしまったが，消費エネルギーを 2.6%削減することができた．実行時間の増加は，HS モードで実行すべき部分を LE モードで実行しているケースが残っているためと考えられる．本実験より，提案手法を用いることで消費エネルギーを下げられることは明らかとなった．今後は，分岐命令数が改良手法で定めた閾値を下回った場合でも HS モードで実行する方が電力効率が高くなる条件を求め，より高精度なモード切換コントローラを開発する必要がある．

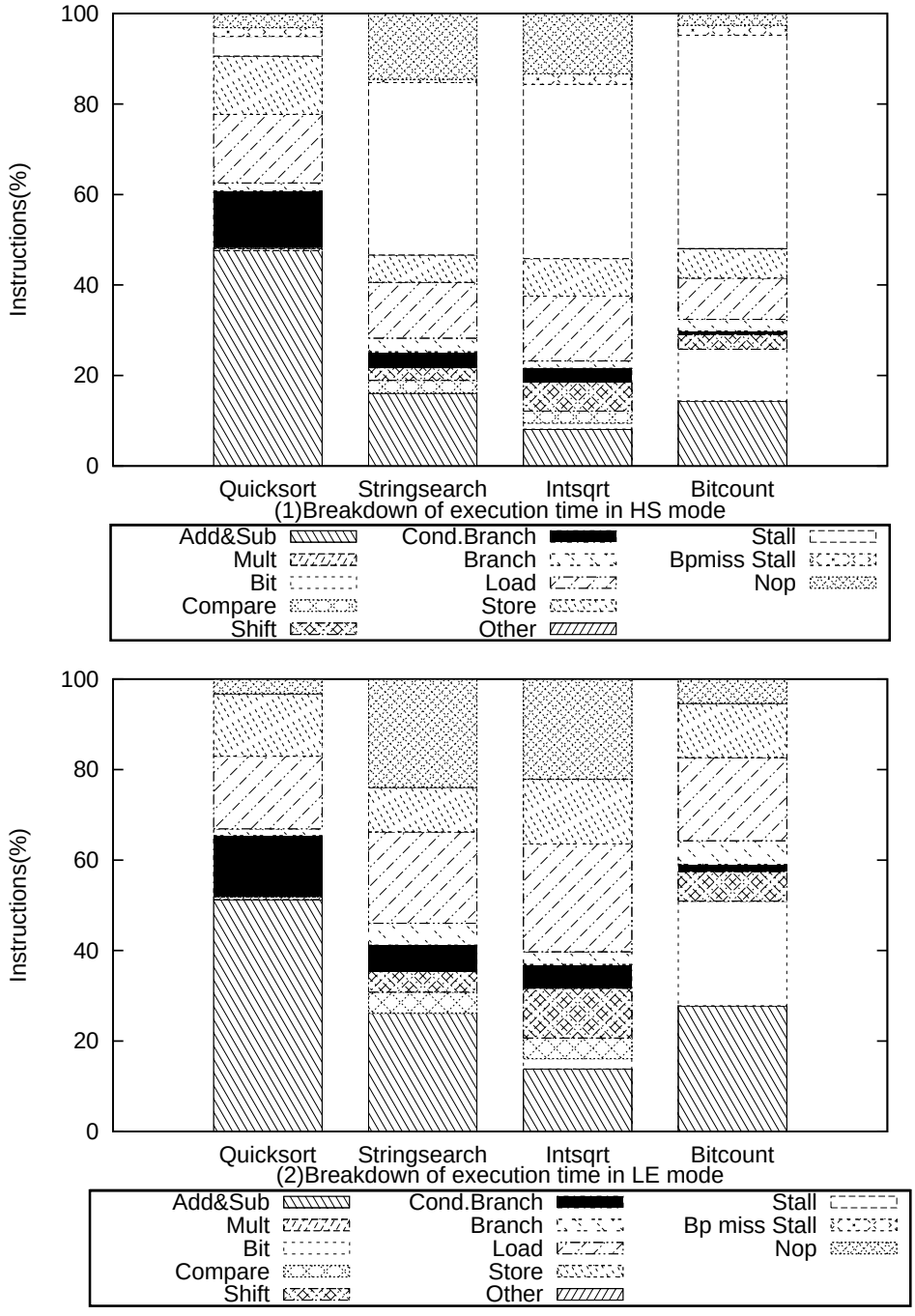

図 5.11: 実行時間の内訳 (HS モード&LE モード)

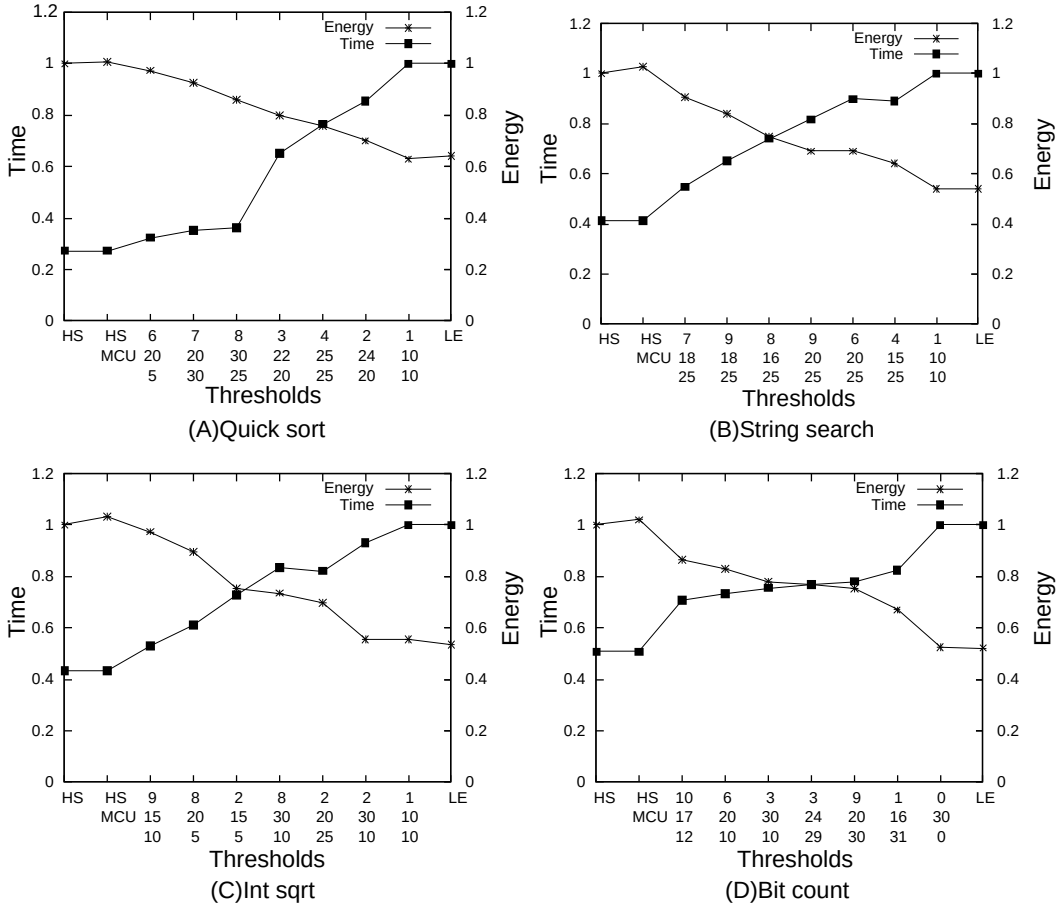
Energy
Time
Time
Energy
Thresholds
HS HS 6 7 8 3 4 2 1 LE
MCU 20 20 30 22 25 24 10
5 30 25 20 25 20 10
(A)Quick sort
HS HS 7 9 8 9 6 4 1 LE
MCU 18 18 16 20 20 15 10
25 25 25 25 25 25 10
(B)String search
HS HS 9 8 2 8 2 2 1 LE
MCU 15 20 15 30 20 30 10
10 5 5 10 25 10 10
(C)Int sqrt
HS HS 10 6 3 3 9 1 0 LE
MCU 17 20 30 24 20 16 30
12 10 10 29 30 31 0
(D)Bit count

図 5.12: 評価結果

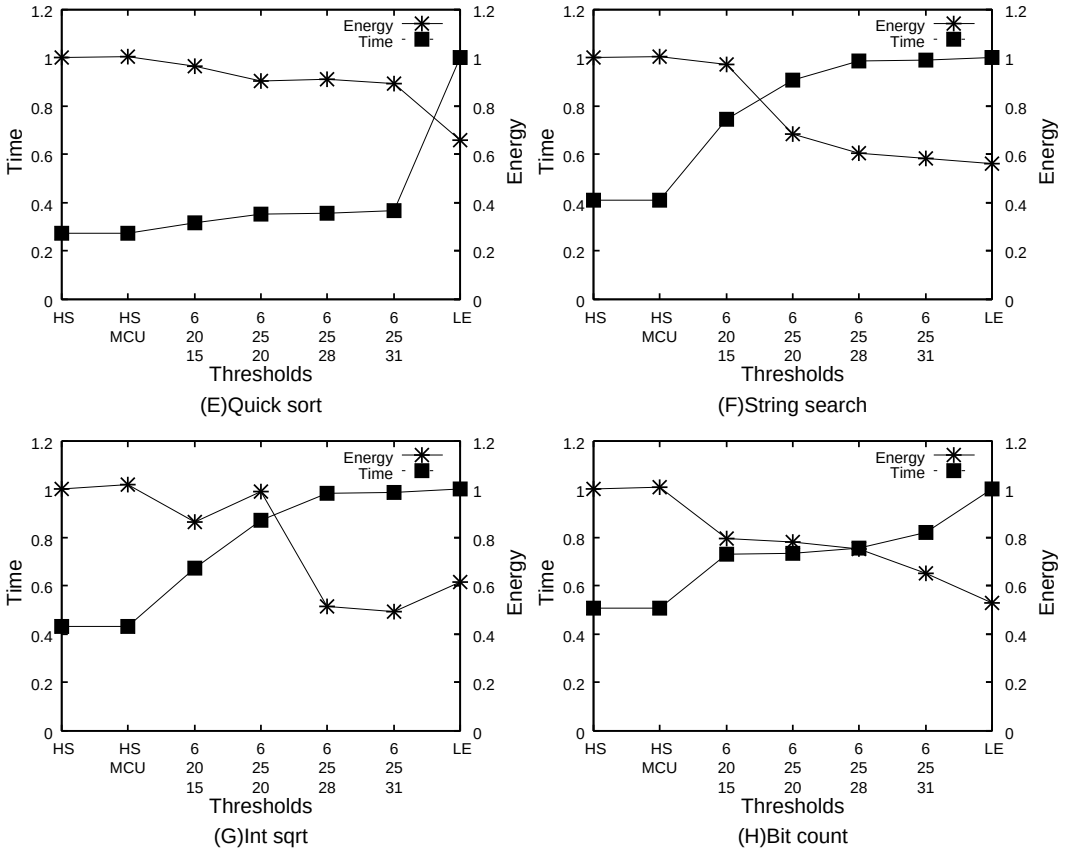
Energy
Time
Time
Energy
Thresholds
HS
HS MCU
6 20 15
6 25 20
6 25 28
6 25 31
LE
(E)Quick sort
(F)String search
(G)Int sqrt
(H)Bit count

図 5.13: 追加評価結果

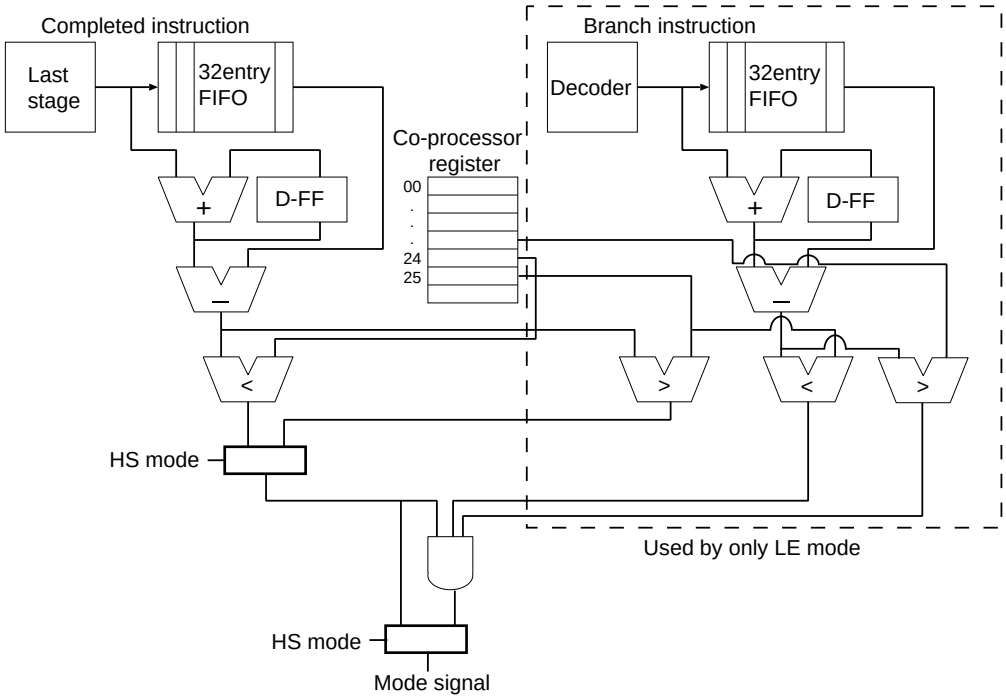

図 5.14: 改良手法のブロック図

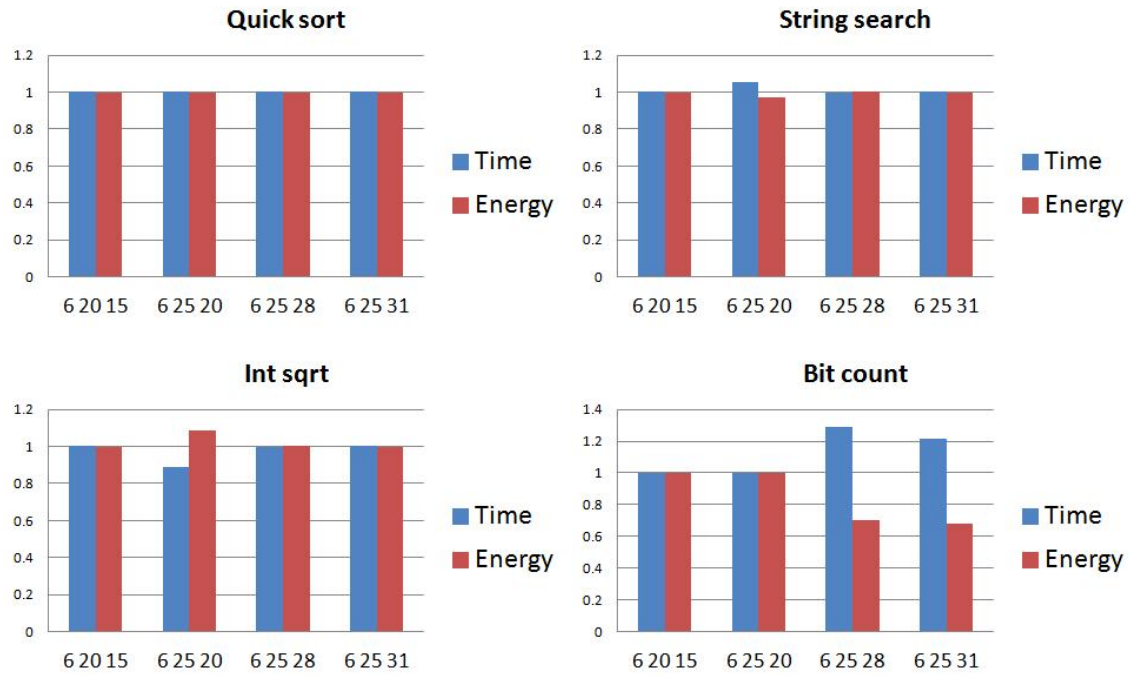

図 5.15: 改良手法の評価結果

6　試作チップの評価環境

本章では，まず，試作 VSP チップの評価に用いた現状の LSI テスター Power Medusa とその問題点について述べる．その後，LSI テスター Power Medusa の改良案の提案を行う．図 6.16 に評価環境のブロック図を示す．

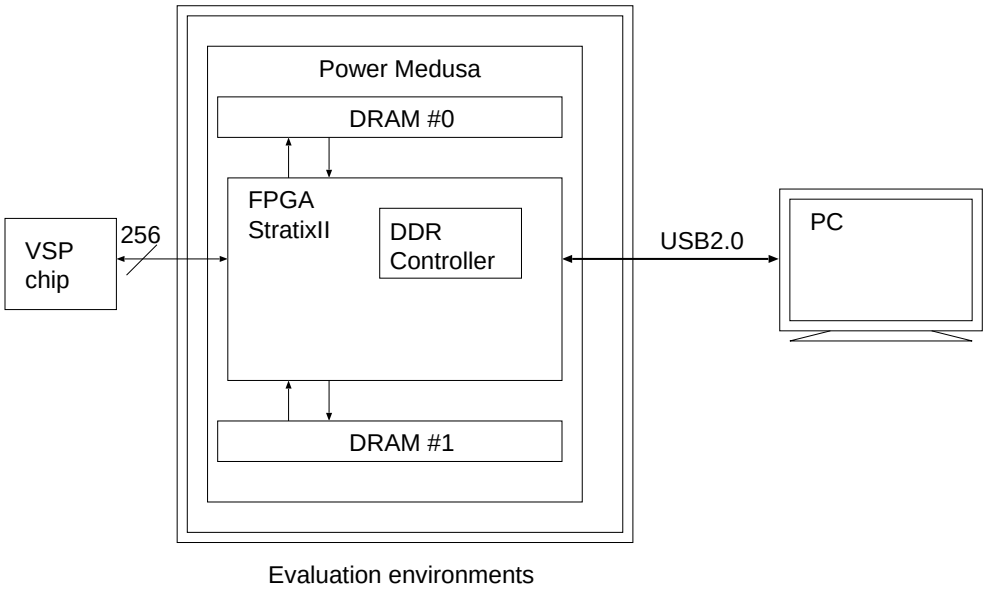

図 6.16: 評価環境のブロック図

6.1　現状の評価環境について

試作チップの評価に用いた LSI テスター Power Medusa は，任意のハードウェアの評価・検証用の汎用テスターである．その汎用性のため，数十万サイクル程度の評価しか行うことができない．現状の LSI テスター Power Medusa を用いて試作チップの電力評価を行うためには，まずコンピュータで評価を行いたいベンチマークプログラムを実行させ，テストパターンを作成する．そして，作成したテストパターン全てを一度コンピュータから送信することでテスター内部の DRAM #0 に格納する．その後，サイクル毎に DRAM #0 から FPGA を介して，試作チップの各ピンにテストパターンを送ることで，チップを動作させている．チップの出力結果は，FPGA を介して DRAM #1 に送信され，チップがプログラ

ムの実行を完了した後でコンピュータに送られる．プログラムを実行するためには，試作チップの 256 ピンにサイクル毎の信号値を保存する必要があり，プログラムの実行サイクル数が増加すると，テストパターンのファイル容量が膨大なものとなってしまう．そのため，大容量 DRAM を用いても数十万サイクル程度の評価しか行うことができない．数十万命令程度の評価では，実際のアプリケーションの様な大規模なプログラムと傾向が異なる可能性がある．そこで，次節では，LSI テスター Power Medusa の改良を提案する．

6.2 評価環境の改良案について

本節では，実際のアプリケーションの様な大規模なプログラムを実行するために，LSI テスター Power Medusa の改良案を提案する．LSI テスター Power Medusa では，数十万命令程度のベンチマークしか実行できない．数十万命令程度の評価では，実際のアプリケーションの様な大規模なプログラムと傾向が異なる可能性がある．そこで，テスター内の DRAM を試作チップのメインメモリの様に扱い，DRAM 内部を動的に書き換えることが可能な様にテスター内の FPGA を改良することで，大規模ベンチマークの実行が可能になると考えられる．具体的には，Power Medusa 内の一方の DRAM を試作チップの Instruction Memory，もう一方の DRAM を Data Memory として扱い，DRAM にはテストパターンではなく，プログラムそのものを送信して実行を行う．この実装を行うことにより，大規模ベンチマークの実行が可能になると考えられる．

提案評価環境を用いて，チップの評価を行う手順を以下に示す．テストパターンではなく，プログラムそのものをコンピュータから，instructon memory である DRAM #0，data memory である DRAM #1 に送信する．その後，チップと DRAM #0，DRAM #1 で FPGA を介して通信を行いプログラムを実行する．チップがプログラムの実行を完了した後，DRAM #2 の検証用データがコンピュータに送られる．

評価環境の改良の提案を行うにあたって今回提案する点を以下に示す．

IO 装置の利用:

- LSI テスター Power Medusa には，入力装置として，テンキー，ディップスイッチ，ロータリースイッチ，出力装置として LED が備えられているが，試作チップの検証には用いられていない．そこで，ディッ

プスイッチを試作チップのリセット，クロックの進行に，LED をプログラムカウンタなどの試作チップのプログラム実行時の情報の表示などに利用できる．

モード表示機能の追加:

- LSI テスター Power Medusa に備えられている LED を試作チップのモード信号と接続し，プログラム実行時の VSP のモードを LED を用いて可視化する．これにより，試作チップのモードが視覚的に検証可能となる．

チップ供給クロックの変更:

- 提案評価環境では，DRAM を試作チップの内部 RAM として扱うので，試作チップから DRAM に要求を送りデータを取得する．既存のクロックを用いてチップを動作させると，DRAM がボトルネックとなってしまい正しく動作しない．そこで，DRAM がチップにデータを送る準備ができたときのみチップのクロックを動作させるようにチップ供給クロックの変更をすることで解決する．

既存資源の流用:

- LSI テスター Power Medusa 内には既にテストパターンを用いて試作チップを評価する回路が内蔵されている．提案評価環境を実装する際には，既存の DRAM コントローラなどを利用することで，回路を一から設計するよりも，設計期間を短縮できると考えられる．

7　結論

本論文では，試作を行ったVSPチップを用いて詳細な評価を行った．

試作VSPチップを用いて評価，解析を行った結果，試作チップが正常に動作することを確認し，プログラムごとに動的に閾値を割当てることにより，より効率的な実行が行えるであろうことが分かった．また，LEモードにおいて，HSモードと比較して，平均44%の消費エネルギーを削減することができた．コントローラ自体のハードウェア量と電力評価を行った結果，コントローラのハードウェア量はプロセッサ全体の1%程度，消費エネルギーは平均で2.1%程度と，十分に小さい値であると考えられる．更に，モード切換コントローラの改良手法により，消費エネルギーを2.6%削減することができた．また，本研究では，試作チップの評価環境である汎用LSIテスターPower Medusaの改良を提案した．

今後の展望としては，提案評価環境を実際に構築することや，より電力効率の高いモード切換コントローラの開発，最適な閾値の動的な決定手法の提案が考えられる．

謝辞

本研究を行うにあたり，多くの助言をいただきました近藤利夫教授，並びにご指導，ご助言くださいました佐々木敬泰助教に深く感謝いたします．また，様々な局面にてお世話になりました計算機アーキテクチャ研究室の皆様にも心より感謝いたします.

参考文献

[1] Hong I., et.al., “On-line scheduling of hard real-time tasks on variable voltage processor”, Proc. of Int. Conf. on Computer-Aided Design, pp. 653-656, November, 1998.

[2] Pouwelse J., et.al., “Dynamic voltage scaling on a low-power microprocessor”, Proc. of 7th ACM Int. Conf. on Mobile Computing and Networking (Mobicom), pp. 251-259, July, 2001.

[3] Tomoyuki Nakabayashi, et.al., “VLSI implementation of Variable Stages Pipeline Processor using Fine-Grain Pipeline Depth Controller”, ITC-CSCC2012, July, 2012.

[4] 野村 和正，佐々木 敬泰，大野 和彦，近藤 利夫，“可変パイプライン段数プロセッサのためのメモリアクセスに着目した細粒度なモード切換えコントローラ “, 信学技報, vol.109, no. 319, CPSY2009-45, pp. 13-18, 2009.

[5] Jun YAO, Shinobu MIWA, Hajime SHIMADA, Members, and Shinji TOMITA, “A Dynamic Control Mechanism for Pipeline Stage Unification by Identifying Program Phases“, IEICE TRANS. INF. & SYST., Vol. E91-D, pp.1010-1022, APRIL 2008.

[6] 嶋田 創，安藤 秀樹，島田 俊夫，“低消費電力化のためのパイプライン”，情報処理学会研究報告，ARC, 2001.

[7] 嶋田 創，安藤 秀樹，島田 俊夫，“パイプラインステージ統合：将来のモバイルプロセッサのための消費エネルギー削減技術”，先進的計算基盤システムシンポジウム SACSI2003, 2003.

[8] Jun YAO, Hajime SHIMADA, Yasuhiko NAKASHIMA, Shin-ichro MORI, Shinji TOMITA " An EDP Study on the Optimal Pipeline Depth for Pipeline Stage Unification Adoption,", Information Processing Society of Japan (IPSJ), 2006.

[9] A.S. Dhodapkar and J.E. Smith, "Managing multi-configuration hardware via dynamic working set analysis," Proc. 29th Annual International Symposium on Computer Architecture, pp.233-244,IEEE Computer Society, 2002.

[10] Koppanalil, J., et.al., "A Case for Dynamic Pipeline Scaling", Proc. of Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems 2002, pp. 1-8, 2002.

[11] MiBench. http://www.eecs.umich.edu/mibench/