東京工業大学工学部

学士論文

The Ultrasmall Soft Processor

指導教員 吉瀬 謙二 准教授

平成25年2月

提出者

学科　　情報工学科
学籍番号　09_14645
氏名　　田中 雄一郎

指導教員認定印	
学科長認定印	

平成24年度　学士論文内容梗概

The Ultrasmall Soft Processor

指導教員　　吉瀬 謙二　准教授

情報工学科
09_14645　　田中 雄一郎

ソフトプロセッサとは，Field-Programmable Gate Array（FPGA）を始めとするプログラマブルロジックデバイスをターゲットとして，論理合成で実装することのできるマイクロプロセッサコアである．近年では，ソフトプロセッサがFPGAを使用するシステムにおいて一般的なコンポーネントとなっており，制御及びデータ処理の幅広い機能を実現するために使用されている．小型デバイスにおいてハードウェア容量は限られており，多機能・高性能化においてその容量制限が障害となる．ゆえに，ソフトプロセッサは小さい方が望ましい．

要求を満たし，最小となるソフトプロセッサを設計するにあたり，小さいソフトプロセッサを参考にすることは非常に有意義である．このようなソフトプロセッサの既存研究に，Supersmall Soft Processorが存在する．32ビットのMIPS命令のサブセットをサポートするソフトプロセッサである．この32ビット命令セットアーキテクチャをサポートするソフトプロセッサでは，我々が知る限りでSupersmall Soft Processorは最小のソフトプロセッサである．

本研究ではSupersmall Soft Processorを元に使用面積の削減，並びに速度向上を図る．主要データパスの2ビット化，レジスタ符号化の最適化，マルチプレクサの入力ロジックを含む最適化を施したUltrasmall Soft Processorを提案する．

論理合成の結果から，Ultrasmall Soft Processorは使用ハードウェア量がSupersmall Soft Processor以下となることを示す．また，全ての命令の実行に要するサイクルの平均値から，1.88倍の高速化を達成できることを示す．

Ultrasmall Soft Processorの応用としてメニーコア化を検討する．Ultrasmall Soft Processorの特徴を活かし，使用ハードウェア量が小さくなるようにリング型のネットワーク構成を提案する．また，コア数の変化による使用ハードウェア量の変化について考察を行う．

目　次

第1章

序論

1.1 研究の背景

携帯電話やカーナビには，通信，画像/音声処理，ファイル処理等の非常に多くの機能が盛り込まれている．機能によって最適なCPUのアーキテクチャが異なるため，1つのCPUで集中処理することは現実的ではない．そのため，各々の処理に適したソフトプロセッサを搭載することで，従来では複数のデバイスを用いていたところを1チップに収めることができるようになった．しかし，これら小型デバイスにおいてハードウェア容量は限られている．多機能・高性能化するにあたってその容量制限が障害となる．ゆえに，ソフトプロセッサはより小さい方が望ましい．要求を満たす中で最小となるソフトプロセッサを設計するにあたり，世界最小のソフトプロセッサを参考にすることは非常に有意義である．

既存研究にSupersmall Soft Processor [1]が存在する．一部を除く32ビットのMIPS命令を実行するソフトプロセッサであり，このISAを用いたもの中では調べた限りで最小のソフトプロセッサである．

1.2 研究の目的

本論文ではSupersmall Soft Processorを元に更なる使用面積の削減，並びに速度向上を図る．主要データパスの2ビット化，レジスタ符号化の最適化，マルチプレクサ（MUX）の入力ロジックを含む最適化を施したUltrasmall Soft Processorを提案する．Ultrasmall Soft Processorのハードウェア使用量がSupersmall Soft Processor以下となった時，Ultrasmall Soft Processorは同じISAを用いるソフトプロセッサの中で世界最小であると言える．

以降，本論文ではUltrasmall Soft Processorのことを Ultrasmallと記述することがある．また，既存手法のSupersmall Soft Processorのことを Supersmallと記述することがある．

1.3　本論文の構成

本論文の構成は以下の通りである．まず，2章で関連研究であるSupersmallについて述べ，その特徴と問題点を明確にする．3章ではUltrasmallの概要と採用した手法，並びに最適化について説明を行う．4章にて各手法，最適化を実装し評価を行う．5章ではUltrasmallの応用例としてメニーコア化を挙げる．最後に6章で本論文をまとめる．

第2章

背景と関連研究

関連研究としてSupersmall Soft Processor [1] について述べる．Supersmallは Altera社製のハイエンドFPGAであるStratix IIIをターゲットとしている．Ultrasmallにおいて Xilinx社製のFPGAをターゲットとする．両社のFPGAには相違点が存在し，特にメモリシステムの独自性がSupersmallを移植するにあたって問題となる．本章では研究の背景として，ソフトプロセッサとFPGAについて最初に述べる．次に移植において問題となったStratix IIIの特徴について述べ、Supersmallについて説明する．

2.1　ソフトプロセッサ

ソフトプロセッサとは，Field-Programmable Gate Array（FPGA）を始めとするプログラマブルロジックデバイスをターゲットとして，論理合成で実装することのできるマイクロプロセッサコアである．RTL記述を変更することにより異なる性能のプロセッサを構成することができ，ハードウェア的な変更をする必要がない．また，異なるデバイスをターゲットとした場合に，デバイスに依存しないRTL記述は変更することなく移植できる．

従来のマイコンベースの機器開発が抱える課題として，柔軟性の欠如が挙げられる．大量生産を前提としたASICでは，全てのユーザの要求に応じた最適なデバイスを生み出すことができない．一方，ソフトプロセッサはユーザオリエントなSoCをミニマムオーダー1個から実現することが可能である．そのため，近年ではソフトプロセッサがFPGAを使用するシステムにおいてより一般的なコンポーネントとなっており，制御及びデータ処理の幅広い機能を実装するために使用されている．

一般に利用されているソフトプロセッサには様々な種類が存在する．メーカやコミニュティが提供するRISCソフトプロセッサの一例を表2.1に挙げる．

その他にも，教育・研究用にシンプルで拡張性の高い構成を重視したソフトプロセッサとしてMipsCoreが存在する [2].

表 2.1: ソフトプロセッサ

プロセッサ名	開発元	命令セットアーキテクチャ	参照先
MicroBlaze	Xilinx	32 ビット	[3]
PicoBlaze	Xilinx	8 ビット	[4]
Nios II	Altera	32 ビット	[5]
OpenRISC 1000	OpenCores	32 or 64 ビット	[6]
Plasma	OpenCores	32 ビット，MIPS I	[7]
Cortex-M1	ARM	32 ビット	[8]

2.2 Field-Programmable Gate Array

FPGA とは，製造後に構成を設定できる集積回路である．内部構成には以下のものが含まれる．

- 論理ブロック
- 内部配線
- フラッシュメモリ
- 乗算器
- I/O エレメント
- クロックネットワーク
- JTAG 制御部

FPGA の代表的なベンダとして Xilinx 社と Altera 社が存在する．Supersmall は Altera 社製の Stratix III をターゲットとし，提案する Ultrasmall は Xilinx 社製の FPGA をターゲットする．

Stratix とは Altera 社製ハイエンド FPGA のシリーズ名である．Supersmall がターゲットとする Stratix III は，Stratix II のアーキテクチャをベースに開発されたものである．移植において問題となるのは，Stratix シリーズ当初から内蔵されているメモリ（TriMatrix メモリ）である．TriMatrix メモリはバイトイネーブルサポート機能を備えている [9–11]．この機能は，入力データをマスクして，データの特定のバイト，ニブル，またはビットのみの書き込みを可能にするというものである．Supersmall では，バイト並びにハーフワード単位でのストア命令の処理にバイトイネーブルサポートを利用している．

バイトイネーブルサポートはマスクを行って書き込みを行う機能である．そのため，書き込むデータの下位ビットを任意の位置にシフトする必要がある．実行サイクルを記憶するカウンタを使用しているため，既存のカウンタのみでシフト回数を数えることができない．そのため，このシフトを行うには新たにカウンタを設ける必要がある．バイト，ハーフワード単位でのロード命令においても新たに設けるカウンタを使用するが，このカウンタは他の命令を処理する際には利用されない．つまり，バイト並びにハーフワード単位でのロード・ストア命令の処理にのみ使用されるカウンタが存在する．この処理はSupersmallはStratixシリーズのTriMatrixメモリのアーキテクチャに依存している．実際の利用例については次節にて説明を行う．

Xilinx社製FPGAの1つであるSpartan-3Eにはバイト単位でのイネーブルを決定する機能が実装されていない [12]．また，Spartan-6並びにVirtex-7は同様の機能を持つが [13, 14]，その機能の利用方法がAltera社製のものと異なるため，ソースコードをそのまま利用することができない．

2.3 Supersmall Soft Processor

Supersmallはマルチクロックサイクルの RISC プロセッサである．実行するISAは32ビットMIPS命令であるが，乗除算と非アライメントロード・ストアには対応していない．使用ハードウェア量は，Altera社製のソフトプロセッサであるNios II/eの約45%である．

図2.1にSupersmallのブロック図を示す．これを簡単化したものを図2.2に示す．また，MIPSの命令セットフォーマットを図2.3に示す．Program Counterの下位nビットがInstruction Memoryのアドレスとなっており，リードイネーブルが1になるとメモリから命令を読み出す．命令のrsとrtに応じた値がRegister Fileに読み出され，サイクルに応じて1ビットずつBarrel Shifterへと送られる．Branch Resolution UnitはRegister Fileから送られるデータを常に監視しており，分岐命令に応じてNext Program Counterへの入力を切り替える．Data Memoryにアクセスする場合はBarrel Shifter Aでアドレスを指定し，ストアする際にはBの値を書き込む．ただし，Data Memoryから読み出した32ビットのデータはBarrel Shifter Aへと出力される．このようにメモリを除くユニット間の信号幅を1ビットとすることで，1サイクルに流れる情報量を制限しユニットの構成を簡単化している．また，信号幅が1ビットとなることでマルチプレクサがそれぞれ小さくなり，配線領域の削減が可能となる．

ALU使用時のBarrel Shifterのデータの変化を図2.4に示す．ここでは簡単のため，Barrel Shifterを4ビット幅として6と7の和を求める．また，この時のBarrel Shifter周りのデータフローを図2.5に示す．Barrel Shifter Aは作動時にデータを1ビットずつLSB側へとシフトし，ALUから送られてくるLSBの演算

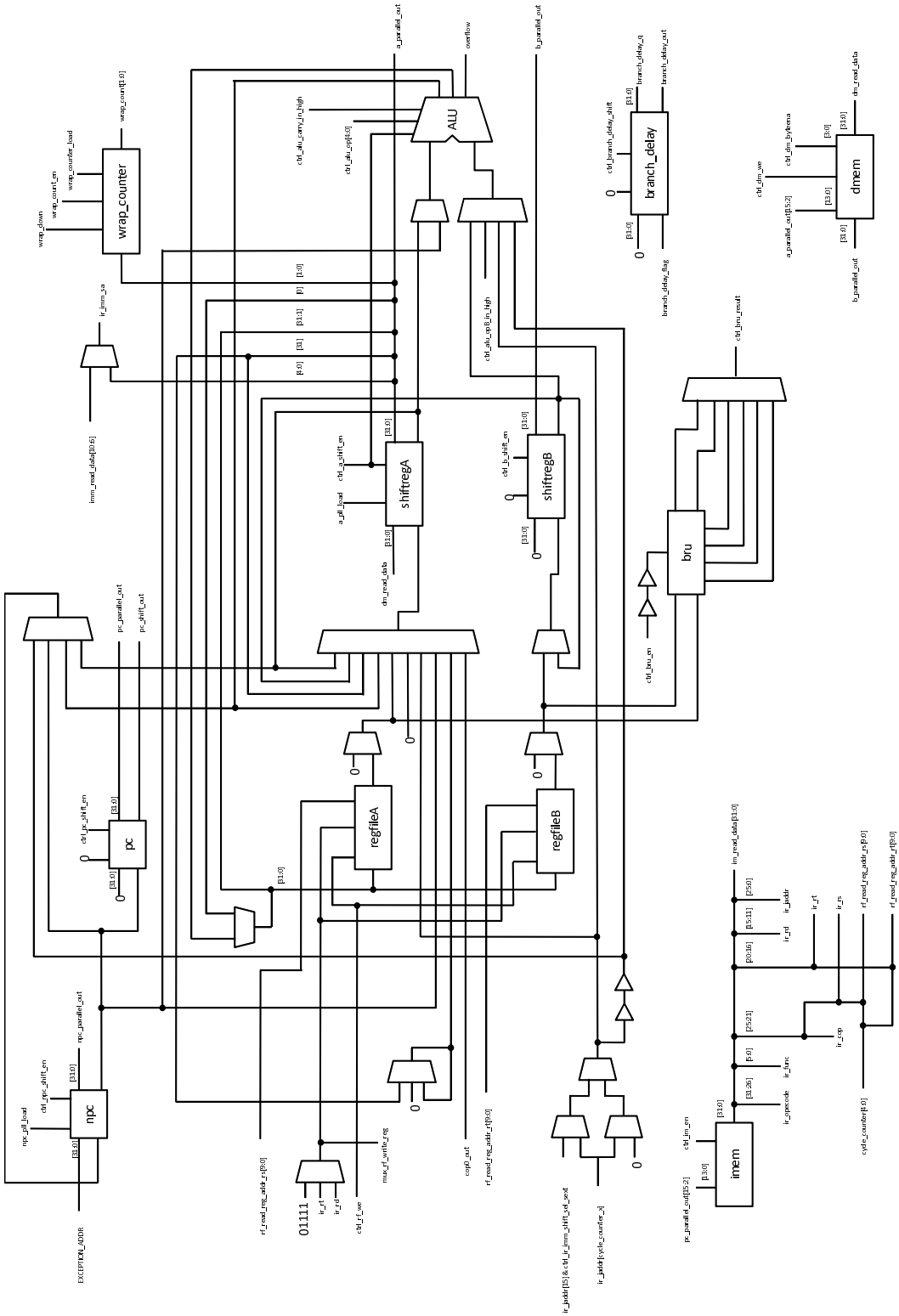

図 2.1: Supersmall のブロック図

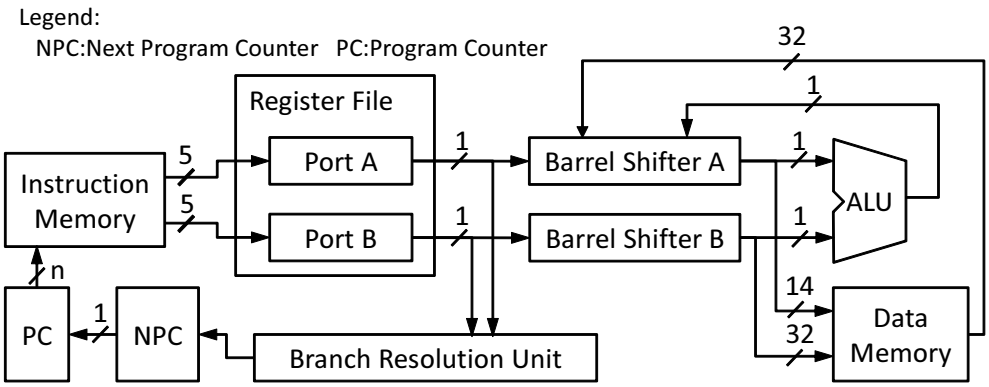

図 **2.2**: Supersmallの簡易ブロック図

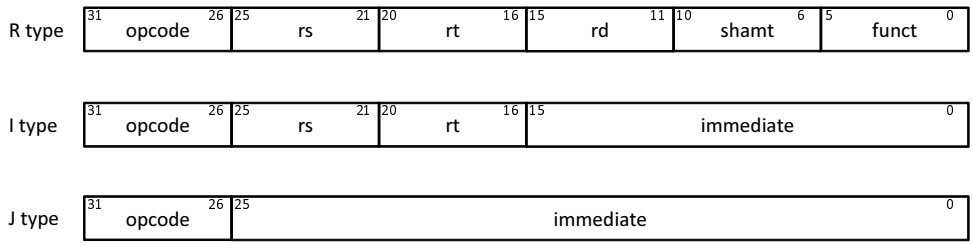

図 **2.3**: MIPS命令セットのフォーマット

結果をMSBへと格納する．これを32サイクル繰り返すことで32ビットの演算を行う．

次に，ロジカルなシフト命令処理時のデータフローを図2.6に示す．右へシフトする場合はMUXからの入力を0とし，シフトするビット数だけBarrel Shifterを作動させればよい．また，左へシフトを行う場合は32サイクル常にBarrel Shifterを作動させる．処理の実行サイクルがシフトしたいビット数となるまでMUXからの入力を0とし，その後LSBへと切り替えればよい．算術シフトにおいては，0の代わりにMSBを入力する．Barrel Shifterの特徴を活かすことによって，ハードウェアを追加することなくシフト命令を実行可能となる．

Supersmallの状態遷移図を図2.7に示す．Supersmallは1サイクルに1ビットしか処理できないため，32ビットの処理を行うには最低32サイクルを要する．図2.7内の太枠で囲われた各状態それにあたる．処理の実行回数を記憶するために，Supersmallは6ビットのカウンタを持つ．カウンタの初期値は0である．そのため，32サイクルで次の状態遷移を行うには，カウンタの下位5ビットの論理積を取って分岐する必要がある．しかし，33サイクル目で状態遷移を行うようにすれば，カウンタの6ビット目を分岐条件に取ればいいため，ロジックが簡単となり要するゲート数が減少する．しかし，処理自体は32サイクルで終

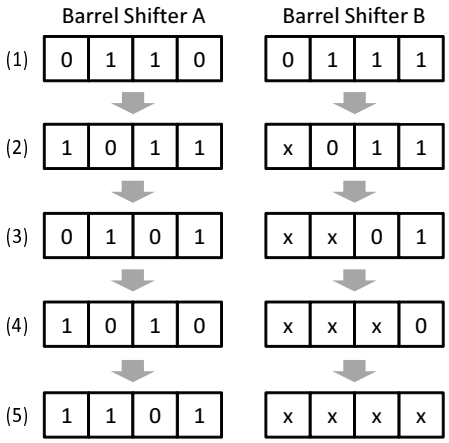

図 2.4: ALU 使用時の Barrel Shifter

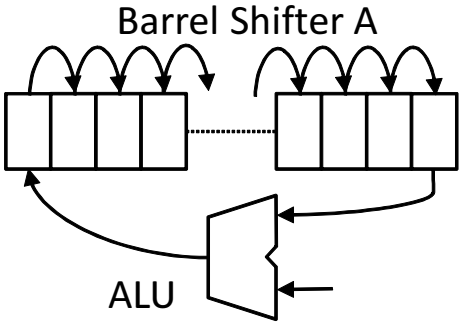

図 2.5: ALU を使用

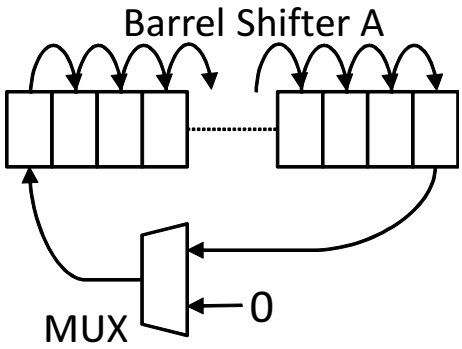

図 2.6: シフトを実行

了しているため，33 サイクル目では何も処理を行わずに状態遷移だけを行う．その様子を図 2.8 に示す．

Supersmall は Nios II/e に対して処理に約 10 倍のサイクル数を要する．

バイト，ハーフワード単位のロード・ストア命令の処理において，Supersmall では Stratix III のメモリシステムに依存したハードウェア記述がなされている．ストア命令を実行する場合，Data Memory 周辺のデータフローは図 2.9 に示す通りである．Barrel Shifter A，B それぞれには rs，rt に応じたデータが Register File から読み出される．次に Barrel Shifter A に即値を加算し，再度 Barrel Shifter A に格納する．メモリシステムは 4 ビットの制御信号を元に，Barrel Shifter A で指定されるアドレスのどの部分を Barrel Shifter B で上書きするかを判断する．また，ロード命令を実行する場合のデータフローは図 2.10 の

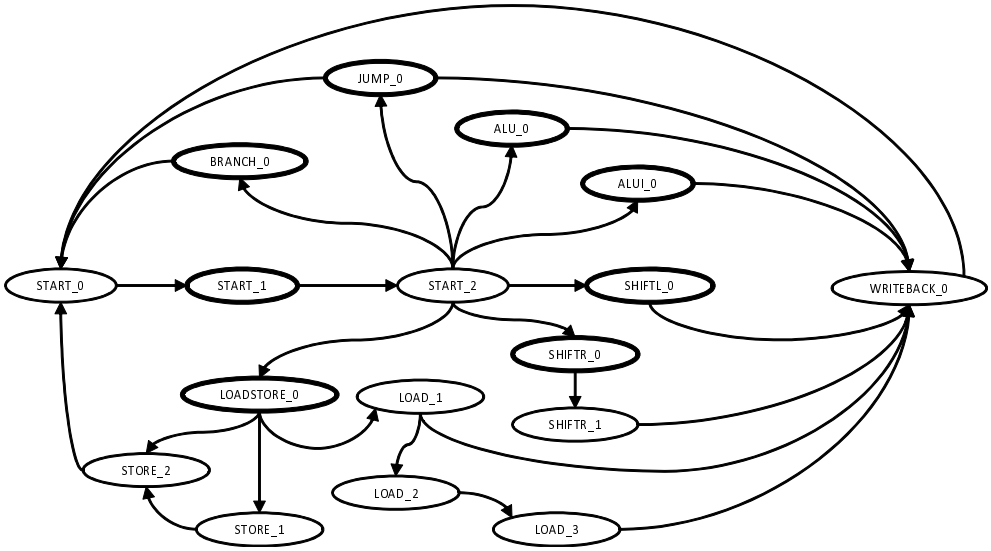

図 **2.7**: Supersmall の状態遷移図

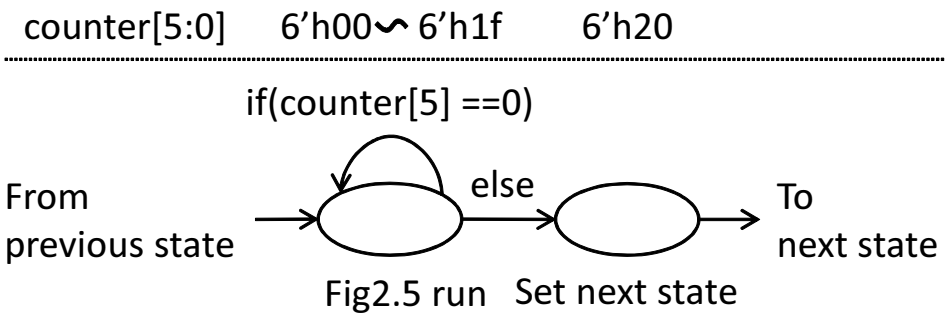

図 **2.8**: Supersmall で ALU を使用する際の状態遷移

通りである．ストア命令同様にアドレスを算出し，Barrel Shifter A にデータを読み出す．

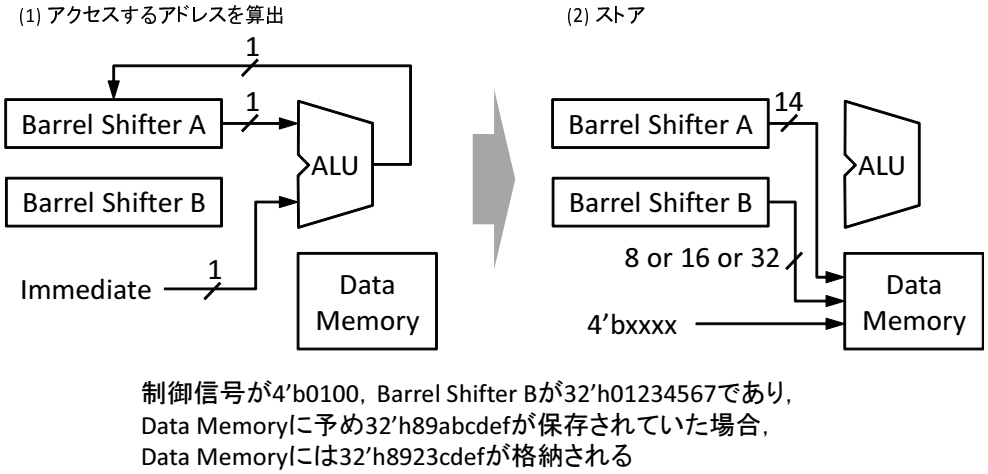

図 **2.9**: Supersmall でストア命令を実行

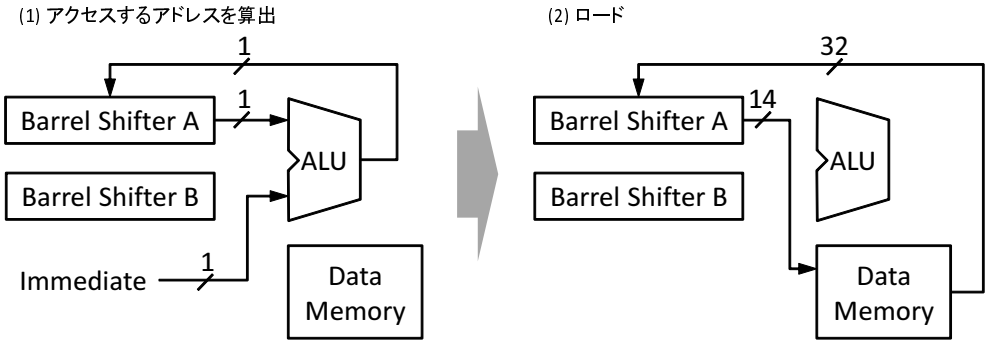

図 **2.10**: Supersmall でロード命令を実行

第3章

Ultrasmall Soft Processorの提案

本章では32ビットのMIPS命令を処理するRISCソフトプロセッサとして，世界最小を目指すUltrasmall Soft Processorを提案する．まず，Supersmallの占有面積の削減と性能向上を目指して，1つのアーキテクチャ的な手法と2つの最適化手法を提案する．これらを組み合わせて，Ultrasmall Soft Processorの全体構成を提案する．

Ultrasmallの前提条件を示す．UltrasmallはXilinx社のFPGAをターゲットとする．FPGAの論理ブロック（Slice）はルックアップテーブル（LUT），全加算器，フリップフロップ（FF）などから構成される．本論文では，安価で広く用いられているSpartan-3Eシリーズ，最先端デバイスを用いている中で安価なSpartan-6シリーズ，最先端デバイスを用いている高性能向けのVirtex-7シリーズをターゲットし，各FPGAにUltrasmallを実装した際に使用するSlice数が最小になることを目指す．また，本章では32ビットの命令セットを扱うソフトプロセッサのみを対象とする．

3.1　2ビット化されたデータパス

ハードウェア量の増加を抑えつつ，プロセッサの性能を向上させることを目的として2ビット化されたデータパスを提案する．図3.1において，太線や太枠となっているものが図2.2からの変更点である．

Supersmallにおいて面積の削減に大きく貢献しているのは，データパスを1ビットとしたことである．これにより，ALUが1ビット幅となり，マルチプレクサ（MUX）が小さくなって配線経路が抑えられる．さらに，実行結果の保存先とシフト処理部を統合することでフリップフロップ（FF）の数を減らしている．

Nios IIやMicroBlazeを始めとする，32ビットの命令を処理するRISCソフトプロセッサのデータパスはいずれも32ビットである．一方，32ビットの命令を処理するSupersmallのデータパスは1ビットである．調査した範囲では，ビット幅がそれら以外であるデータパスを持つソフトプロセッサは存在しない．

仮に，使用ハードウェア量をそのままにSupersmallが採用している1ビット

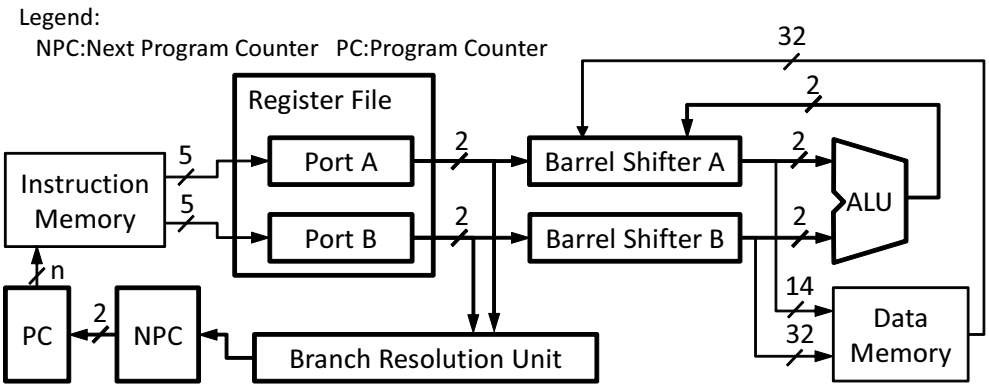

図 3.1: データパスを 2 ビット化

表 3.1: ALU のデータパスを変更

データパス	1bit	2bit	4bit	8bit	16bit
Reg	4	4	5	5	5
LUTs	5	8	15	27	51
Slices	5	3	7	11	18

のデータパスを大きくできれば，Supersmall の問題点であった処理性能が改善される．これに関する予備評価として，同じデバイス上で Supersmall の ALU を 2 ビット，4 ビットとした時のハードウェア量の変化させた結果を表 3.1 にまとめる．この結果から，従来では 1 ビットの ALU が最小と思われていたが，2 ビットの方が使用ハードウェア量が小さくなることがあることが分かった．これより，Ultrasmall ではデータパスを 2 ビットとし，処理性能を高めることにする．

データパスを 2 ビットとした場合，1 ビットの場合と比較して各 MUX は大きくなる．そのため，使用する LUT の数が大きくなると考えられる．しかし，FF に関しては既存のものを使いまわすことができるので，FF の数に大きな変化は見られない．FPGA の各 Slice に含まれる LUT と FF の数は決まっている [11, 12, 15, 16]．Supersmall で必要とされる FF の数が LUT の数に対して過多であった場合，使用されていない LUT を用いることでデータパスの拡張を行える可能性がある．つまり，Slice の使用数を変化させずに処理性能の改善が行える．

図 2.6 に示した様に，Supersmall ではデータの保存先である Barrel Shifter の特徴を活かしてシフトを行う．そのため，Supersmall のデータパスを 2 ビット化するだけでは，図 3.2 に示すように奇数ビット幅のシフトを行えない．ここで

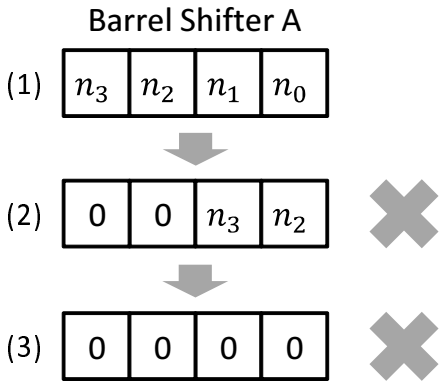

図 3.2: 奇数ビットの非対応

は，簡単のためにBarrel Shifterのビット幅を4ビットとし，3ビットのシフトを行うものとする．一方，Supersmallでは図3.3に示すように，Barrel Shifterを用いて実行することができる．

そこで，Ultrasmallでは2種類のシフト処理を用意することでこの問題を解決する．2Nビットのシフトと1ビットのシフトである．図3.4にサイクル毎のBarrel Shifterの値の変化を示す．3ビットシフトする場合，まずLSBを無視して2Nビットのシフトを行う．これは図3.4の（1）～（2）にあたり，その時のデータフローを図3.5に示す．これはシフト回数のLSBを無視しているだけで，処理内容はSupersmallと同じである．そのため，アーキテクチャを流用することができる．次に，LSBが1であった場合のみ1ビットのシフトを行い，そうでない場合はシフト処理状態から出る．データフローを図3.6に示す．図3.4では（3）～（5）がこの1ビットのシフトにあたる.（3）において，Barrel Shifterの下位3ビットとtmpを合わせて出力すると要するサイクル数は減少する．しかし，それでは新たにMUXの追加と配線領域の確保を必要とする．そのため，既存のアーキテクチャで実現できるようにこのような構成を採用する．

1ビットのシフトには16ないし17サイクルを要するため，奇数ビットのシフトを行う際にデータパスを2ビット化した恩恵を与ることは難しい．図3.3の例ではSupersmallでは3サイクル必要とするのに対して，2ビット化すると4サイクル必要となる．しかし，偶数ビットのシフトではSueprsmallの半分のサイクルで処理を終えることができる．シフトの状態を2つ設けて分岐させることにより，処理に要する平均サイクル数がSupersmallより小さくなる．また，1ビットのシフトを実装するにあたって追加するフリップフロップはは1つ（図3.5，3.6におけるtmp）である．一方，データパスを2ビット化することによりカウンタが5ビットとなるので，FFの数がSupersmallより大きくなることはない．FFの総数を殆ど変化させることなく，Barrel Shifterの特徴を活かすことが可能である．

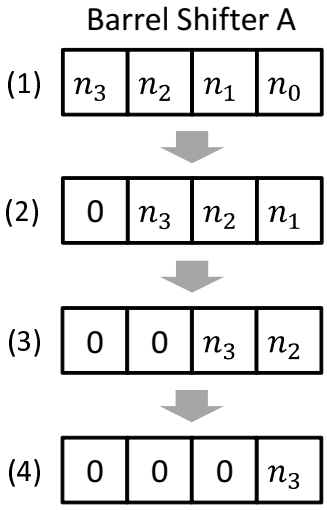

図 **3.3**: Supersmall で 3 ビットシフト

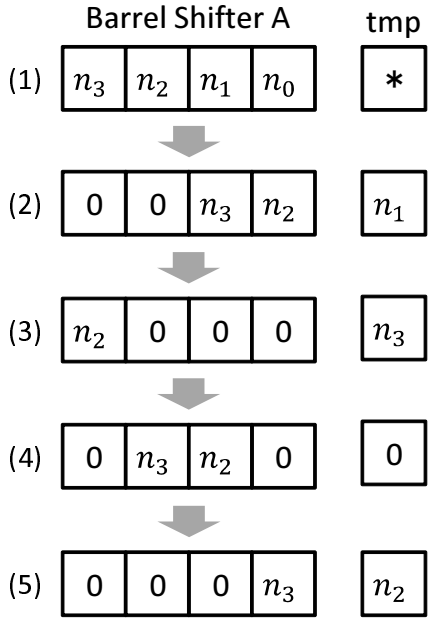

図 **3.4**: Shift 時の Barrel Shifter

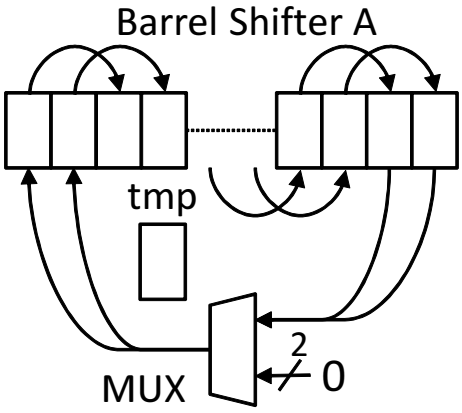

図 **3.5**: 2N ビットのシフト

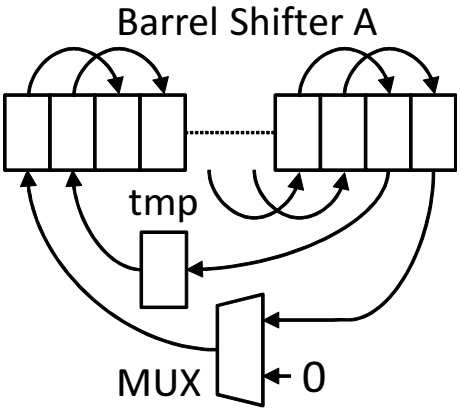

図 **3.6**: 1 ビットのシフト

3.2 レジスタの符号化

2 章で述べたように，Suepersmall は 33 サイクル目では処理を行わずに状態遷移のみを行う．ALU は常に Barrel Shifter の LSB 同士を演算して値を出力している．しかし，Barrel Shifter が動作しなければその値が保存されることはな

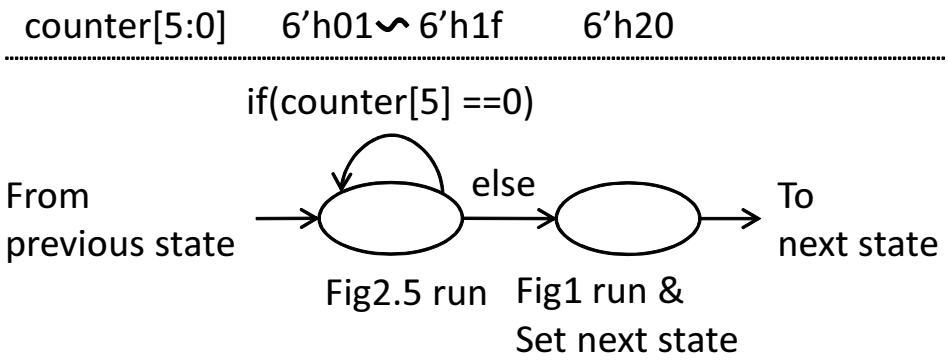

図 **3.7**: カウンタの初期値を1とした場合の状態遷移

く，Barrel Shifter内のデータが破壊されることはない．同様の理由で，Barrel Shifterの動作さえ停止させればデータを破壊することなく待機することができる．

Barrel Shifterには作動させるか否かを決定するイネーブル線が出ている．Supersmallでは，このイネーブル線にカウンタ6ビット目の否定を繋いでいる．そうすることで，33サイクル目ではBarrel Shifterは停止し，状態遷移のみを行うことができる．しかし，これは実行速度を低下させるだけではなく，配置配線を難化させる可能性がある．何故なら，カウンタからイネーブル線までの配線経路を確保しなければならないからである．もし，各状態が32サイクルで処理を終えるのであればイネーブル線は常に1でよく，配線しやすくすることができる．

そこで，カウンタの初期値を1とすることで，下位5ビットの論理積を取らずに32サイクル目で状態遷移を行えるようにする．図3.7にその様子を示す．Ultrasmallではデータパスが2ビットであるため，この手法により図2.7の太枠で囲われた状態の実行サイクルが17サイクルから16サイクルに減る．

3.3　多段構成のマルチプレクサ

UltrasmallのBarrel Shifter Aへの入力は12本あり，その中からMUXで1本選択される．この選択はいかなる状態でも行われる．MUXへの入力本数が2の乗数ではないために，その制御信号を効率よく利用できていない．また，MUXへの入力には数ある状態の中でも1状態でのみ選択されるものも含まれる．このような入力を一度MUXでまとめ，更にそのMUXからの出力と残るBarrel Shifter Aへの入力をMUXでまとめて2段構成とする．

図3.8はMUXに施す最適化の前後の様子を示す．最適化する前では，全ての状態において4ビットの制御信号を出力する必要がある．変更後はほとんどの

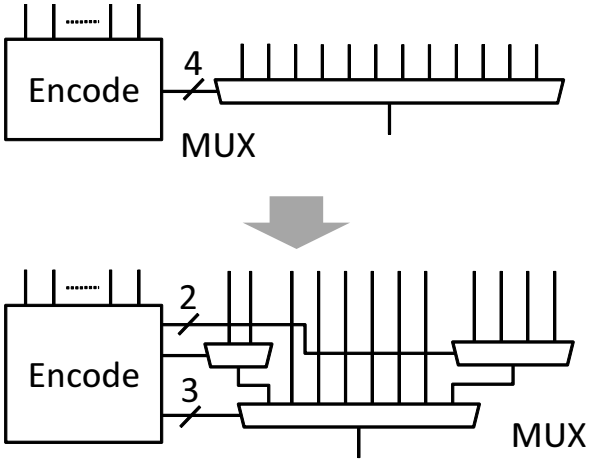

図 **3.8**: MUX による調整

状態において制御信号は3ビット指定するだけでよい．入力をまとめたことにより，その入力を選択するには最大5ビット指定する必要が出てくる．しかし，この入力は使用される頻度が非常に少ないため，全体としては制御信号に要する配線経路が削減されると考えられる．

3.4 デバイス依存のメモリシステム

バイト，並びにハーフワード単位のロード・ストア命令の処理において，SupersmallではStratix IIIのメモリシステムに依存したハードウェア記述が成されている．そのため，UltrasmallではSupersmallのハードウェア構成を変更しない限り処理することができない．そのため，図3.9に示すようにハードウェア構成を一部変更してこれに対応する．変更点は太線並びに太枠で示す．

Ultrasmallのメモリシステムはワード単位で書き込みを行うため，Barrel Shifter Bに書き込むデータをただ用意するだけでは正しく動作しない．そのため，バイト，ハーフワード単位で書き込みを行う場合は，一度書き込み先のデータを読み出す必要がある．しかし，図2.10で示した通り，Supersmallの構成では読み出しを行った際にBarrel Shifter Aに保存された書き込み先のアドレスが破壊されてしまう．そのため，読み出し先をBarrel Shifter Bに変更し，アドレスを破壊することなく，指定部分のみの上書きを可能とする．

図3.10にバイト，ハーフワード単位で書き込みを行う流れを示す．Barrel Shifter Aの値に即値を足し合わせ，書き込み先のアドレスを算出する．Barrel Shifter Bにデータを読み出し，Barrel Shifter BのLSBがMSBに入るよう接続する．このようにすることでBarrel Shifter B内でローテートさせることができる．また，rtに対応したデータは常にRegister File PortBより出力されてい

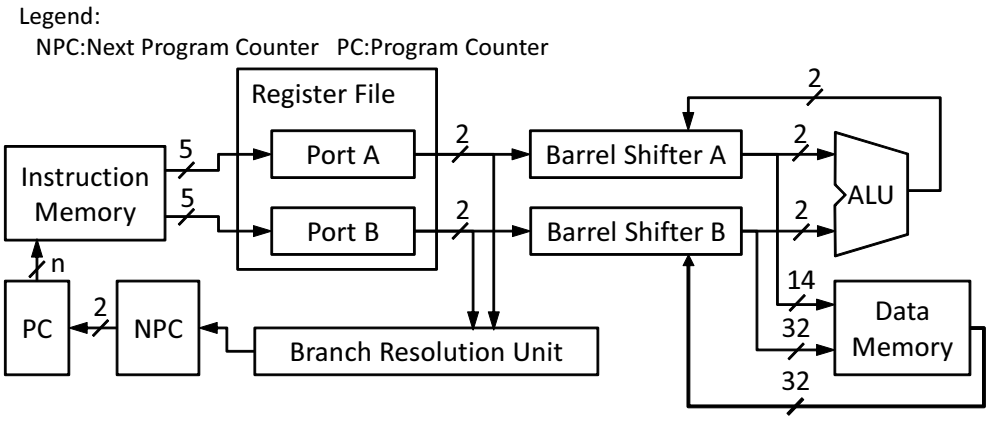

図 **3.9**: 変更後の Ultrasmall のブロック図

る．ゆえに，上書きを行いたい箇所で，入力を Barrel Shifter B から Register File PortB に切り替えればよい．

ロード命令実行時のデータフローを図 3.11 に示す．ただし，図 3.10 と共通の部分については一部省いている．Barrel Shifter B に読み出したデータは，命令に応じた拡張を行いながら Barrel Shifter A へと移す．最後にアライメントを行い，次の状態へと遷移する．

3.5 Ultrasmall Soft Processor のアーキテクチャ

これらの手法と最適化を施した後の状態遷移図を図 3.12 に示す．また，ブロック図を図 3.13 に示す．

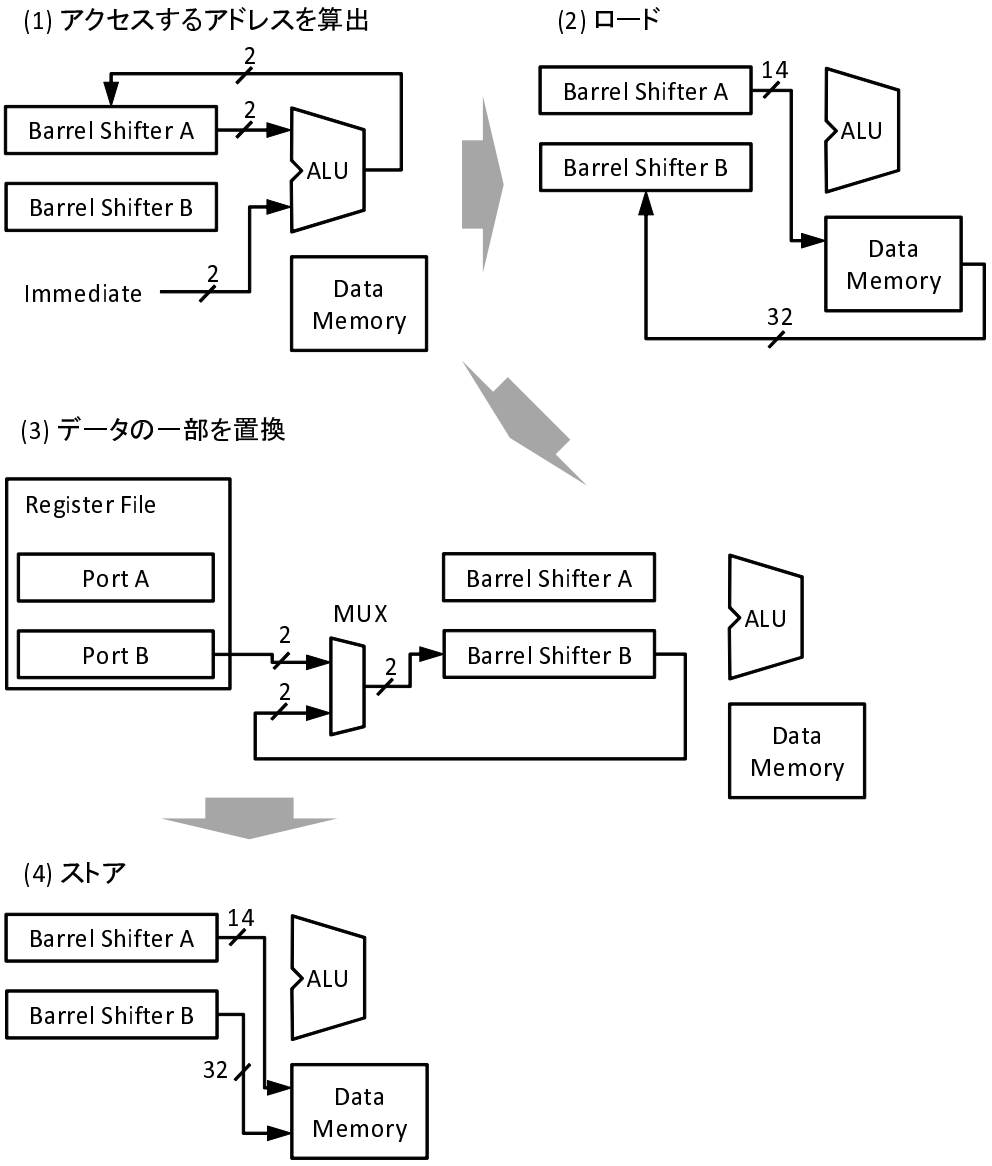

図 **3.10**: Ultrasmall でストア命令を実行

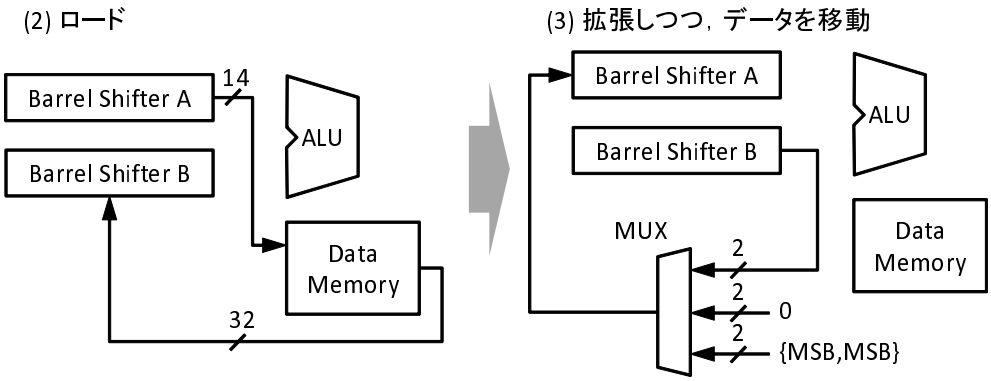

図 **3.11**: Ultrasmall でロード命令を実行

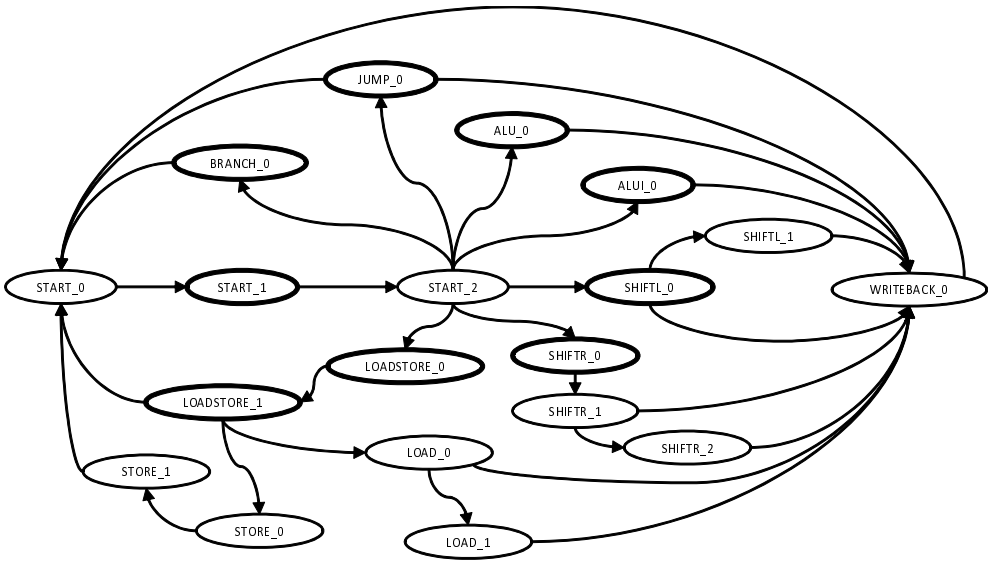

図 **3.12**: Ultrasmall の状態遷移図

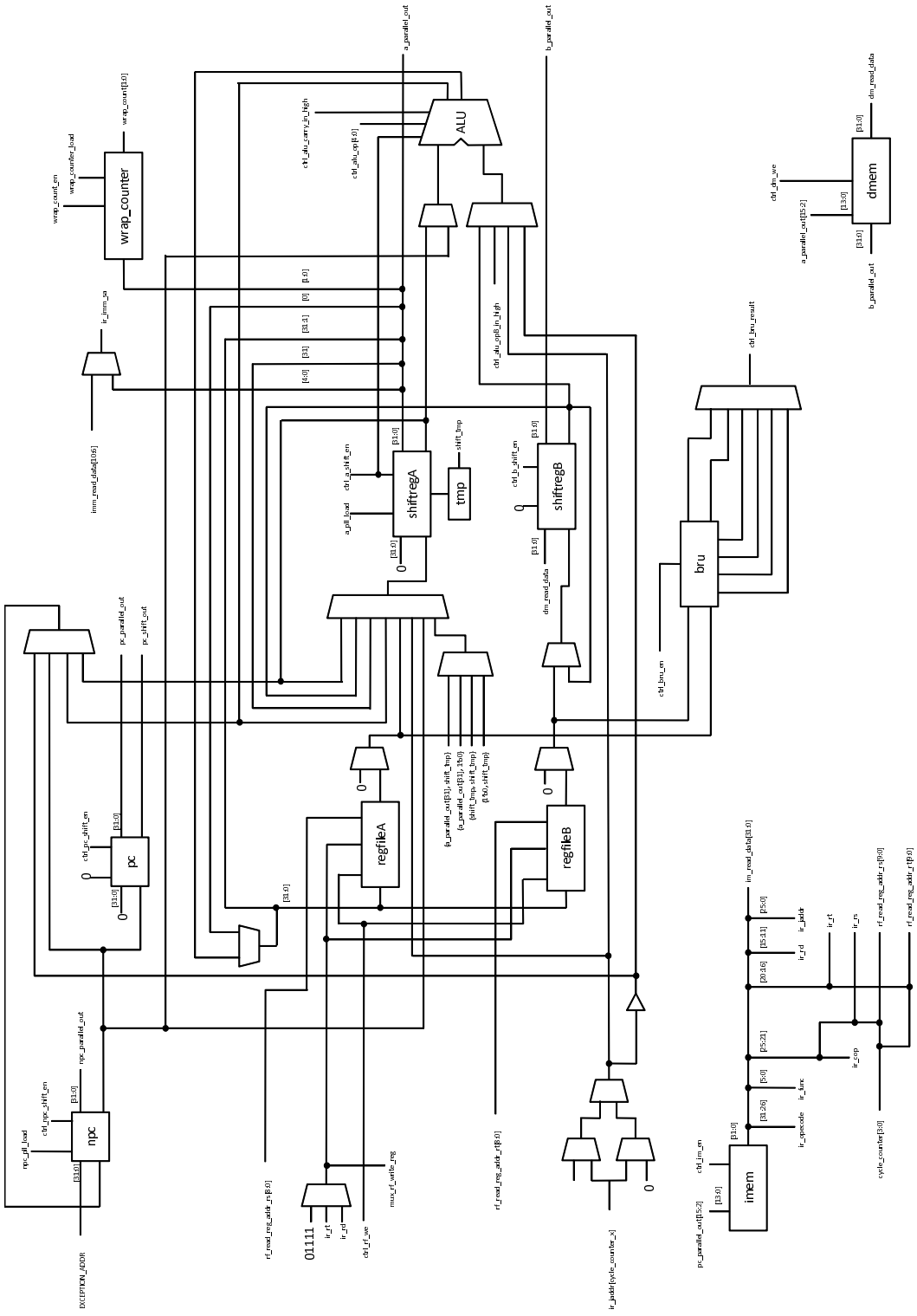

図 3.13: Ultrasmall のブロック図

第4章

評価

4.1 ハードウェア量

Supersmall は Altera 社製の FPGA である Stratix III をターゲットとしており，バイト，ハーフワード単位のロード・ストア命令に関してそのメモリシステムに依存したハードウェア記述がなされている．Ultrasmall は Xilinx 社製の FPGA をターゲットとするため，双方からこれらの命令処理部を除いて比較を行う．また，簡単のために例外処理部も除く．

ツールは Xilinx 社製 ISE（version 14.2）を使用し，Optimization Goal を Area に，Optimization Effort を High と設定して評価を行う．

表 4.1 に以下の各手法と最適化を実装した場合と Supersmall との比較を示す．図 4.1，4.2，4.3 は表 4.1 をグラフにしたものである．

- Supersmall（2bit）……データパスの 2 ビット化
- Supersmall（Encode）……符号化の最適化
- Supersmall（2bit & Encode）……2 ビット化し符号化を最適化したもの
- Ultrasmall……Supersmall（2bit & Encode）に MUX の変更を行ったもの

4.2 性能向上

表 4.2 に 4.1 節で考察を行った各手法，最適化を施した場合の CPI をそれぞれ示す．また，そのグラフを図 4.4 に示す．本来，CPI の測定はアプリの命令の使用頻度にもとづいた重み付き平均である．しかし，今回は簡単のために全ての命令に対して CPI を計算し，その算術平均をとった．

レジスタの符号化の最適化により 1.8 サイクル，データパスの 2 ビット化により 31.5 サイクル減少する．これらを採用した Ultrasmall では CPI が 38.1 サイクルとなり，Supersmall の 71.6 サイクルと比べて 47%削減された．これは，動作

表 4.1: Supersmall と各手法並びに最適化との比較

Spartan-3E

	Slices	Reg	LUTs	LUTRAM	BRAM
Supersmall	**205**	164	300	8	10
Supersmall（2bit）	211	144	302	4	10
Supersmall（Encode）	205	162	296	5	10
Supersmall（2bit & Encode）	207	141	291	4	10
Ultrasmall	**205**	**141**	**289**	**4**	**10**

Spartan-6

	Slices	Reg	LUTs	LUTRAM	BRAM
Supersmall	**140**	152	205	4	10
Supersmall（2bit）	142	144	201	2	10
Supersmall（Encode）	131	149	205	1	10
Supersmall（2bit & Encode）	142	142	203	2	10
Ultrasmall	**139**	**142**	**203**	**2**	**10**

Virtex-7

	Slices	Reg	LUTs	LUTRAM	BRAM
Supersmall	**163**	152	200	4	6
Supersmall（2bit）	153	144	202	2	6
Supersmall（Encode）	154	149	194	1	6
Supersmall（2bit & Encode）	158	142	202	2	6
Ultrasmall	**157**	**142**	**200**	**2**	**6**

表 4.2: 各ソフトプロセッサの CPI

	Supersmall	2bit	Encode	Fusion	**Ultrasmall**
CPI	71.6	40.1	69.8	38.1	**38.1**

周波数が等しいとき，Ultrasmall が Supersmall の 1.88 倍の性能をもつことを意味する．

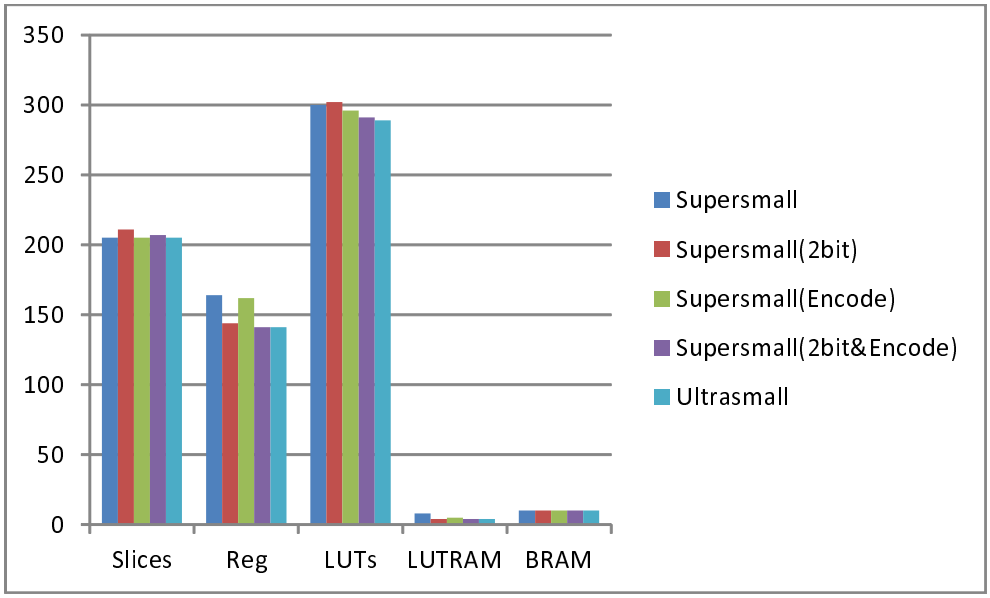

図 **4.1**: Spartan-3E 上での比較

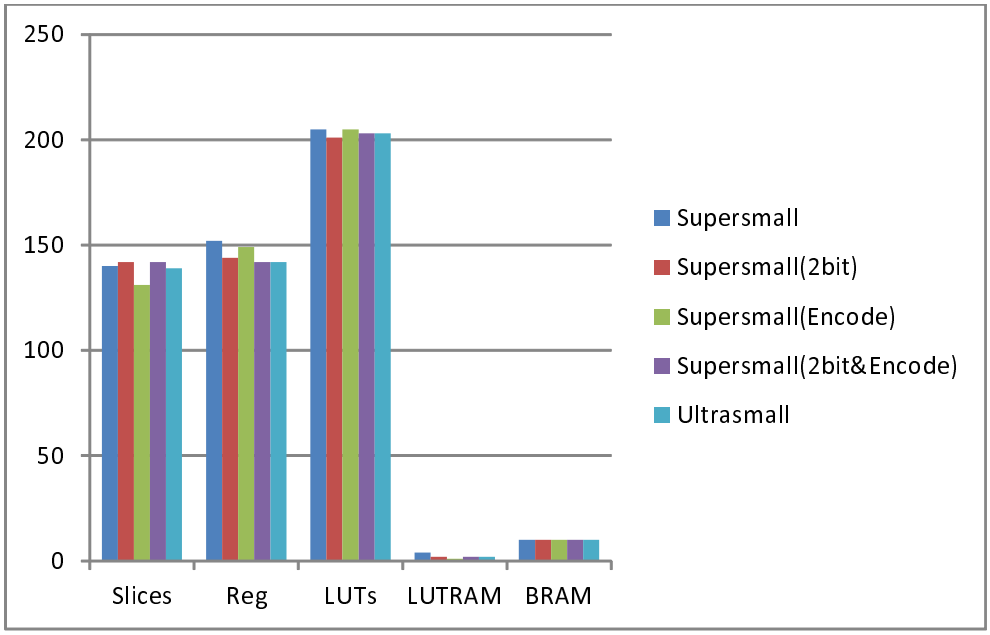

図 **4.2**: Spartan-6 上での比較

4.3 Ultrasmall Soft Processor

メモリシステムに依存した命令を実装した Ultrasmall の各状態と必要サイクルを図 4.5 に示す．また，命令それぞれが通る状態も合わせて示す．図 4.5 で用いている状態名は Supersmall に準拠している．アーキテクチャを変更したた

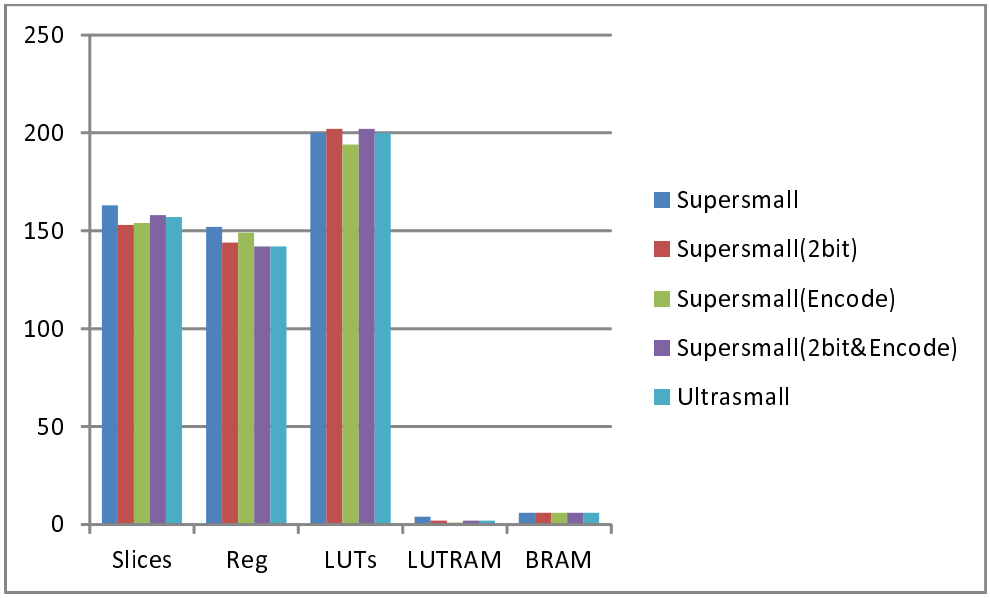

図 **4.3**: Virtex-7 上での比較

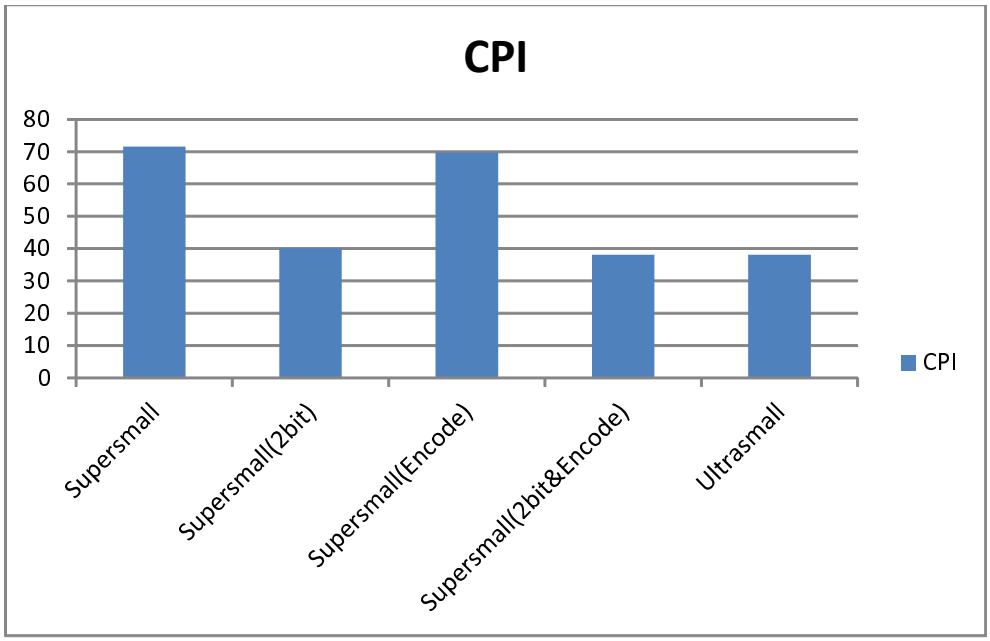

図 **4.4**: CPI の比較

め，ワード単位のロード命令が 16 サイクル遅くなる．そのため，表 4.2 で示した CPI は 41.3 サイクルに増加する．しかし，Supersmall も対応命令数が増加したために，CPI は 75.3 サイクルとなる．そのため，Ultrasmall は除外した命令の実装の有無に関わらず，Supersmall の 1.8 倍以上の速度で処理を実行できる．

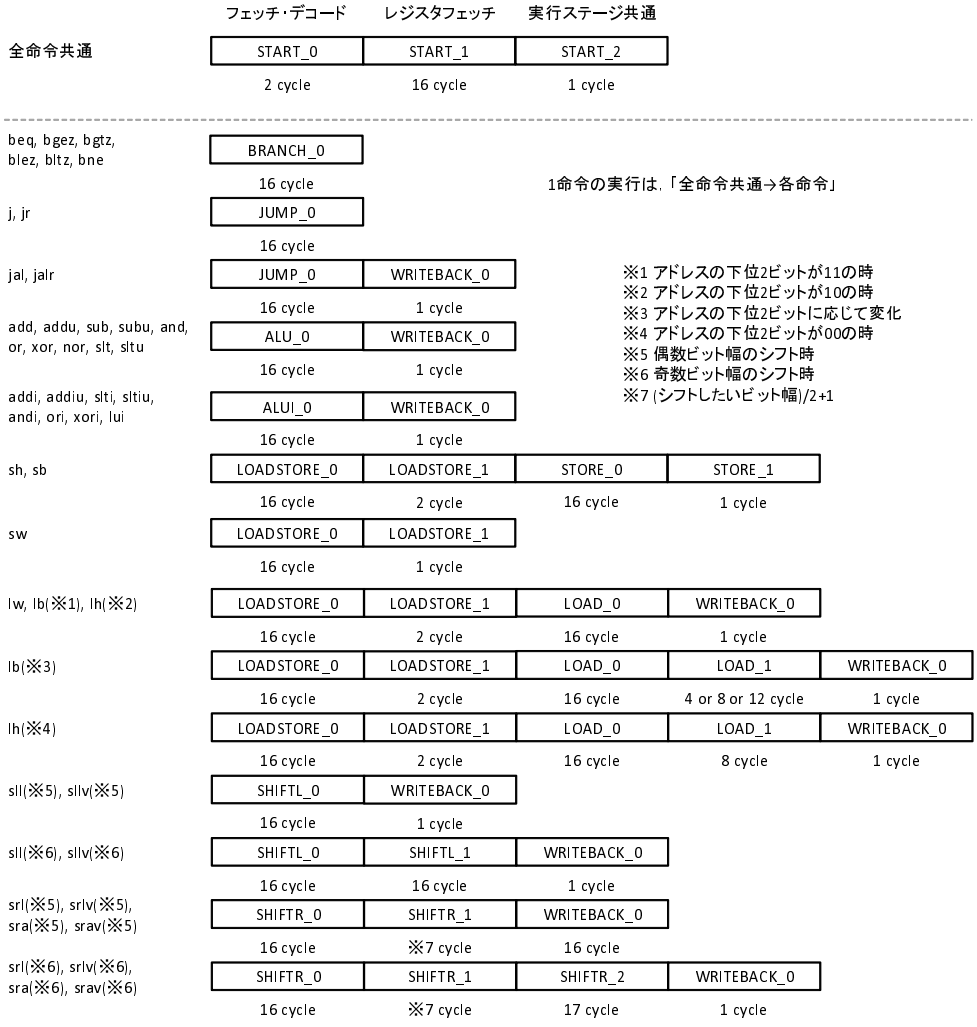

図 4.5: 各命令の状態遷移と必要サイクル数

4.4 考察

表 4.1 において，Supersmall（2bit）は Virtex-7 以外のデバイスにおいて Supersmall よりもハードウェア量が大きくなる．Reg と LUTs を比較してみると，Reg は常に Supersmall を下回っている．一方，LUTs を見てみるとデバイスによって Supersmall よりも大きくなったり小さくなったりしているため，今回の構成では LUTs が最小化に影響を及ぼしたと考える．また，Spartan-6 においては Reg，LUTs の双方共に Supersmall よりも下回っているにも関わらず，使用している Slice は上回る．これは Supersmall（2bit）が配置配線しづらい構成をしていたために，結果的に Supersmall よりも大きくなったと考えられる．ゆえに，配置配線を考慮に入れた構成を行うことも最小化を行うにあたって重要であることが分かる．予想に反して Reg の総数が大きく減少しているのは，Next

Program Counterが分散RAMをによって形成されるようになったからである．

Supersmall（Encode）では配置配線を簡単化すると共に，無駄なサイクルの削減を図った．どのデバイスにおいてもSupersmall（Encode）の使用Slice数はSupersmall以下である．RegとLUTsの総数においてSupersmallとの差が大きく開いていないにも関わらず，Spartan-6やVirtex-7においてSlice数に隔たりが生じている．これは配置配線がより簡単となったために，配線経路が削減されたと考えられる．

2bitとEncodeを組み合わせて用いたものがSupersmall（2bit & Encode）である．Supersmall（2bit）と同じように，Virtex-7を除く2つのデバイスではSupersmallより大きくなる．しかし，Spartan-3E並びにSpartan-6においてRegとLUTsはSupersmallのそれを下回っており，Supersmall（2bit & Encode）も配置配線に適した構成をしていなかったためにSupersmallより大きくなったと考えられる．

UltrasmallはSupersmall（2bit & Encode）にMUXの最適化を施したものである．Regの数はいずれのデバイスにおいてもSupersmall（2bit & Encode）と一致しており，LUTsのみSpartan-3EとVirtex-7上で減少している．Spartan-6上ではその数に変化はないもののSlice数は減少しており，Supersmall（2bit & Encode）と比較して配置配線に適した構成であることが分かる．また，全てのデバイス上でSlice数が減少したことで，Ultrasmallは使用ハードウェア量がSupersmall以下のソフトプロセッサとなった．32ビットMIPS命令を実行するソフトプロセッサにおいて，調査した限りで最も小さいものはSupersmall Soft Processorである．ゆえに，Ultrasmall Soft Processorは世界最小のソフトプロセッサである．

第5章

Ultrasmall Soft Processorの応用

Ultrasmallの応用例としてメニーコア化を考える．ソフトプロセッサ同士を繋ぐだけではメニーコアとして動作しないため，ハードウェアを追加してネットワークを構築する必要がある．本章ではUltrasmallに施したハードウェア変更，並びに追加について述べる．

5.1　メニーコアプロセッサ

Ultrasmallは占有面積が最小となるように設計を行ったソフトプロセッサである．その特徴を最大限活かすため，ネットワークも極力小さくなるように設計する．

構成するネットワークを図5.1に示す．ルータとUltrasmallをセットで1ノードとし，これをリング状に配置する．バレルシフタをループ状に配置したような構造をしており，1サイクルに1ビットずつノードに対して入力並びに出力が行われる．既定のサイクル（シフタのビット数と等しいサイクル数）毎に正しいデータが送られてくることになるので，受信する場合はReceive BufferにFlow Bufferからコピーして Flow Bufferの中を空にする．送信を行う場合は，既定サイクル時にFlow Bufferが空であることを確認して，Send Bufferのデータを Flow Bufferへと移す．ノードにはそれぞれIDがハードウェア的に与えられており，送られてくるデータのID部を自身のものと比較することで送受信の判断を行う．

電源が投入されるとまず初期化を行う．ノードは初期化モードと通常のルータモードの2種類のモードを持つ．初期化モードでは，Slaveノードの Instruction Memoryへ書き込むデータをMasterノードからネットワーク上を1周させる．この際，SlaveノードはIDの比較を行わずにReceive Bufferへデータをコピーする．そして，次のSlaveノードへとデータを転送する．初期化を終えるとルータモードへと切り替わり，以降Receive Bufferへコピーした際には Flow Bufferを空とする．

初期化を終えると，Masterノードはタスクを分割し各Slaveノードへと配る．

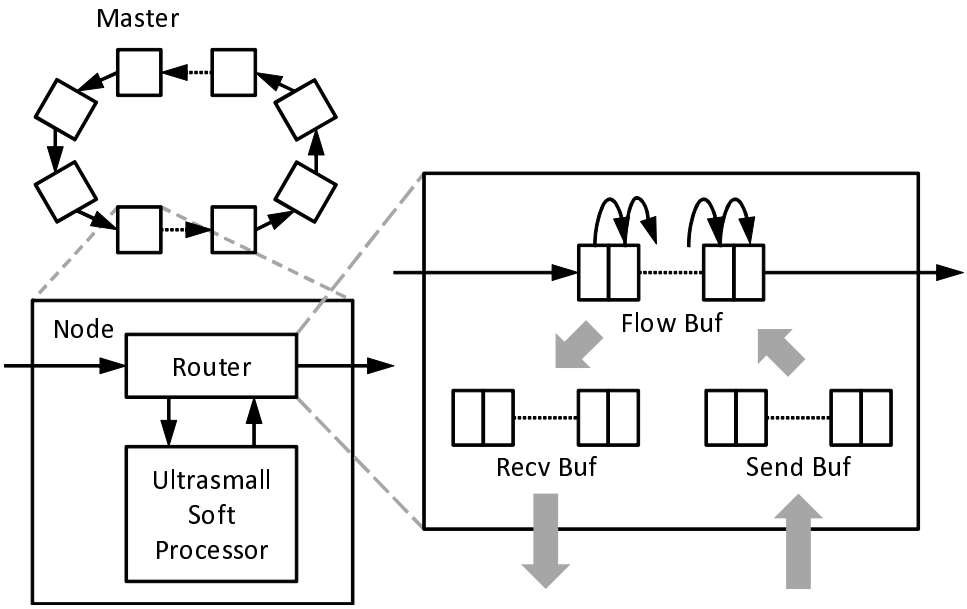

図 **5.1**: マルチコア化に向けたネットワーク構成

各 Slave ノードは処理を終えると Master ノードへ演算結果を送信する．このような体系をとることによって，Master ノードを書き換えるだけで様々な並列プログラムを実行することが可能となる．また，ネットワークの構成も簡単となるため，非常に多くのソフトプロセッサを載せたメニーコアを実装することが可能となる．

まだ少数のコアでの実装例であるが，我々はネットワーク経由によるマルチコアの初期化と Barrier による同期がシミュレーション上で正しく動作することを確認した．

ノード数の変化によるハードウェア規模の推移を表 5.1 に示す．対象としたのは Virtex-7 であり，ツールは 4 章と同様の設定で論理合成を行う．

1 コアでの合成結果を見ると，ノードの約 3 分の 1 をルータが占めていることが分かった．ルータは 3 種類のバッファを持つために，多くのスライスを使用したと考えられる．また，表 5.1 のコア間の使用 Slice 数を比較すると，32 コアまではコア数が 2 倍になっても使用 Slice 数は 2 倍にならない．しかし，64 コアからは Slice 数が 2 倍以上となっている．ハードウェアの構成要素の内，コア数が増えるにつれて使用率が最も高くなるのは BRAM で，128 コアでは 49%となる．このことから，BRAM の使用率が高くなったために配置配線が困難になり，Slice の使用数が大きくなったと考えられる．

表 5.1: コア数を変化させての論理合成結果

Core	Slices	Reg	LUTs	LUTRAM	BRAM
1	255	298	488	2	6
2	570	587	943	4	12
4	1092	1213	1909	8	24
8	2051	2417	3809	16	48
16	4139	4813	7597	32	96
32	8095	9617	15140	64	192
64	16210	19273	30507	128	384
128	34708	38537	63234	256	768

第6章

結論

本論文では，ソフトプロセッサの最小化を目的として，占有面積の削減並びに速度向上を実現する手法と最適化を提案した．また，その応用としてメニーコア化を検討した．

Supersmall Soft Processorは，使用マルチプレクサの縮小化と配線領域の削減のために主要データパスが1ビット幅であった．これは，32ビット幅に対して，処理速度を犠牲に使用ハードウェア量を小さくするものである．提案するUltrasmall Soft Processorはこのデータパスを2ビット幅とし，使用ハードウェア量をSupersmallに比べて殆ど変化させることなく処理速度を大幅に向上した．また，Supersmall Soft Processorはレジスタの符号化，マルチプレクサの最適化が不十分であった．改良することにより，Ultrasmall Soft ProcessorはSupersmall Soft Processorの使用ハードウェア量以下に抑え，1.88倍の処理速度を実現した．Ultrasmall Soft Processorは，32ビットMIPS命令で動作するソフトプロセッサの中で世界最小である．

各手法並びに最適化の実装結果を比較してみると，一部ではレジスタ，LUT共にSupersmallよりも少ないにも関わらずSlice数は大きくなるという事例が生じた．この原因は配置配線による差であると考え，配線の易化も考慮に入れてUltrasmallを実装した．しかし，これはISEの出力結果からの判断によるものであり，実配線を元に判断したものではない．そのため，この配置配線の考察は不十分である．

また，Ultrasmall Soft Processorの応用例としてメニーコア化を提案した．少数コアではあるが，ネットワークを経由しての初期化とBarrierによる同期を実装し，シミュレーション上で正常動作を確認した．実装するコア数を2の乗数として論理合成を行った．64コアまでは，コア数が2倍となっても使用ハードウェア量は2倍以下となる．しかし，64コアを境に使用ハードウェア量が2倍以上となる．これは，ハードウェアリソースが枯渇してきたために配置配線が難化したことが原因と考えられ，メニーコア化していく過程においては問題となる．

謝辞

研究を進めるにあたり，適切なご指導を賜りました指導教員の吉瀬謙二准教授に深く感謝いたします．永塚智之先輩には本論文を書くにあたり，数多くの助言を戴きました．メニーコア化にあたっては，佐藤真平先輩並び笹河良介先輩に様々なご迷惑をおかけしました．おかげさまで研究が意図する形となり，深く感謝しています．また，吉瀬研究室の皆様には多くの助言をして戴きました．煮詰まっていないか適度に声をかけてくださる先輩方，息抜きにと運動やゲームに誘ってくださる先輩方と多くの先輩方に助けられました．人材だけでなく，物資の面でも非常に恵まれた環境で研究することができたと思います．最後に，地元を遠く離れて暮らす息子にいつも温かく接し，支援してくれる両親にも感謝したいです．ありがとう．

参考文献

[1] Robinson, J. and Vafaee, S. and Scobbie, J. and Ritche, M. and Rose, J.,
"The supersmall soft processor,"
Programmable Logic Conference (SPL), 2010 VI Southern, pp.3–8, 2010

[2] Naoki Fujieda, Takefumi Miyoshi, and Kenji Kise,
"SimMips: A MIPS System Simulator,"
Workshop on Computer Architecture Education(WCAE) held in conjunction with MICRO-42, pp. 32-39, December 2009

[3] MicroBlaze Processor Reference Guide,
http://japan.xilinx.com/support/documentation/sw_manuals/xilinx14_1/mb_ref_guide.pdf
Xilinx, 2013年1月31日

[4] PicoBlaze 8-bit Embedded Microcontroller User Guide,
http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf
Xilinx, 2013年1月31日

[5] Nios II Processor Reference Handbook,
http://www.altera.co.jp/literature/hb/nios2/n2cpu_nii5v1.pdf
Altera, 2013年1月31日

[6] OR1K:Community portal,
http://opencores.org/or1k/OR1K:Community_Portal
OpenCores, 2013年1月31日

[7] Plasma - most MIPS I(TM) opcodes :: Opcodes ,
http://opencores.org/project,plasma,opcodes
OpenCores, 2013年1月31日

[8] ARMv6-M Architecture Reference Manual,
https://silver.arm.com/download/ARM_and_AMBA_Architecture/AR585-DA-70000-r0p0-00rel0/DDI0419C_arm_architecture_v6m_reference_manual.pdf
ARM, 2013年1月31日

[9] Stratix Device Handbook,
http://www.altera.co.jp/literature/hb/stx/stratix_handbook.pdf,
Altera, 2013 年 1 月 31 日

[10] Stratix II Device Handbook,
http://www.altera.co.jp/literature/hb/stx2/stratix2_handbook.pdf,
Altera, 2013 年 1 月 31 日

[11] Stratix III Device Handbook,
http://www.altera.co.jp/literature/hb/stx3/stratix3_handbook.pdf,
Altera, 2013 年 1 月 31 日

[12] Spartan-3E FPGA ファミリ：データシート（全モジュール）,
http://japan.xilinx.com/support/documentation/data_sheets/ds312.pdf,
Xilinx, 2013 年 1 月 31 日

[13] Spartan-6 FPGA Block RAM Resources User Guide,
http://www.xilinx.com/support/documentation/user_guides/ug383.pdf,
Xilinx, 2013 年 1 月 31 日

[14] 7 Series FPGAs Memory Resources User Guide,
http://www.xilinx.com/support/documentation/user_guides/
ug473_7Series_Memory_Resources.pdf,
Xilinx, 2013 年 1 月 31 日

[15] Spartan-6 FPGA Configurable Logic Block User Guide,
http://www.xilinx.com/support/documentation/user_guides/ug384.pdf,
Xilinx, 2013 年 1 月 31 日

[16] 7 Series FPGAs Configurable Logic Block User Guide,
http://www.xilinx.com/support/documentation/user_guides/
ug474_7Series_CLB.pdf,
Xilinx, 2013 年 1 月 31 日