東京工業大学工学部

学士論文

32 ビット RISC 命令を実行する
世界最小ソフトプロセッサ

指導教員　吉瀬 謙二　准教授

平成 26 年 2 月

提出者

学科　情報工学科

学籍番号　10_17043

氏名　中塚 裕志

指導教員　印	

学科長　認定印	

平成 25 年度　学士論文内容梗概

32 ビット RISC 命令を実行する
世界最小ソフトプロセッサ

指導教員　吉瀬 謙二　准教授
情報工学科
10_17043　中塚 裕志

ソフトプロセッサとは、Field-Programmable Gate Array（FPGA）や、Complex Programmable Logic Device（CPLD）のような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．FPGA において、論理回路を構成するためのロジック・ブロックのハードウェア容量は限られており，ソフトプロセッサの多機能化においてその容量制限が障害となる．ゆえに，ソフトプロセッサでは回路面積は小さい方が望ましく、性能や命令セットをある程度犠牲にしたものでも需要がある．

既存の小さなソフトプロセッサに、Ultrasmall Soft Processor が提案されている．Ultrasmall Soft Processor は、2 ビットシリアルアーキテクチャを採用することで、性能を犠牲にしつつも、回路面積を小さく抑えている．32 ビットの MIPS 命令のサブセットをサポートし、この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世界最小を謳っている．

本研究では、Ultrasmall Soft Processor を元に、Xilinx 社の FPGA である、Spartan-3E のアーキテクチャの特徴を活かした最適化を行うことで、回路面積の削減、ならびに性能向上を図る．一部データパスの 32 ビット化、過剰なリソース・シェアリングの除去、プリミティブ記述による最適化を施した Ultrasmall+ を提案する．

これらの最適化を行った Ultrasmall+ は、Ultrasmall の約 1.64 倍性能向上し、回路面積は約 16% 削減された．これは、Ultrasmall+ が、Spartan-3E 上という条件を持ちながらも、世界最小のソフトプロセッサであることを示している．

目次

第 1 章

序論

1.1 研究の背景

FPGA（Field-Programmable Gate Array）は、設計者が構成を設定できる、再構成可能なゲートアレイである．近年では、SoC（System-on-Chip）や ASIC のプロトタイプ、少量生産のデバイスに実装されることが一般的な使用用途となっている．

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、論理合成で実装可能なマイクロプロセッサコアはソフトプロセッサと呼ばれる．ソフトプロセッサには、MMU（Memory Management Unit）を搭載し、Linux が動作するような大規模なものから、コントローラとしての用途を想定した小規模なものまで様々なものが提案されている．FPGA において、論理回路を構成するためのロジック・ブロックのハードウェア容量は限られており、ソフトプロセッサの多機能化においてその容量制限が障害となる．よって、FPGA に実装するソフトプロセッサは小さいほど好ましく、性能や命令セットをある程度犠牲にしたものでも需要がある．

こうした既存の小さいソフトプロセッサでは、独自 ISA（Instruction Set Archtecture）を持つ 8 ビットプロセッサの PicoBlaze[1] や、32 ビット MIPS ISA を持つソフトプロセッサで世界最小を謳う Ultrasmall Soft Processor[2] などがある．

1.2 研究の目的

本論文では、Ultrasmall Soft Processor を対象に、回路面積削減を主な目的とした最適化を行う．（1）Ultrasmall Soft Processor のデータパスの一部を 2 ビットから 32 ビットに変更することによるマルチプレクサの削減、（2）過剰なリソース・シェアリングの除去、

そして、(3) 一部のモジュールを FPGA プリミティブで記述することによる、回路面積の削減手法を提案する．これら 3 つの手法により，Ultrasmall Soft Processor の回路面積を削減しつつも，プロセッサ性能の向上を図る．Ultrasmall Soft Processor の回路面積を削減するこれら 3 つの提案手法は，Xlinix 社の FPGA である Spartan-3E のアーキテクチャに依存する．

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．

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、研究背景として FPGA、特に Xilinx 社の Spartan-3E のアーキテクチャについて説明する．また、関連研究である Ultrasmall やその他のソフトプロセッサについてその特徴を述べる．3 章では Ultrasmall の回路面積を削減する 3 つの手法について説明を行う．4 章にて各手法を実装し、評価を行う．最後に 5 章で本論文をまとめる．

第 2 章

背景と関連研究

Ultrasmall は、Xilinx 社の FPGA である、Spartan-3E、Spartan-6、Virtex-7 をターゲットとしている．本研究では、Ultrasmall に Xilinx 社の FPGA である Spartan-3E のアーキテクチャに依存した最適化を行う．FPGA の内部構造に着目した最適化を Ultrasmall に適用することで，性能向上を達成しながらも回路面積の削減を行う．

本章では最初に、研究の背景として FPGA、特に Spartan-3E のアーキテクチャについて述べる．次に、関連研究として様々なソフトプロセッサについて述べる．特に、ソフトプロセッサの中でも Ultrasmall Soft Processor は本研究と密接に関係しているため、節を設けて説明する．

2.1 Field-Programmable Gate Array

FPGA とは、設計者が構成を設定できる、再構成可能な集積回路である．FPGA の内部には一般に、以下のものが含まれる．

- I/O ブロック
- スイッチ・マトリクス
- ロジック・ブロック
- BRAM
- 乗算器
- クロックネットワーク
- JTAG 制御部

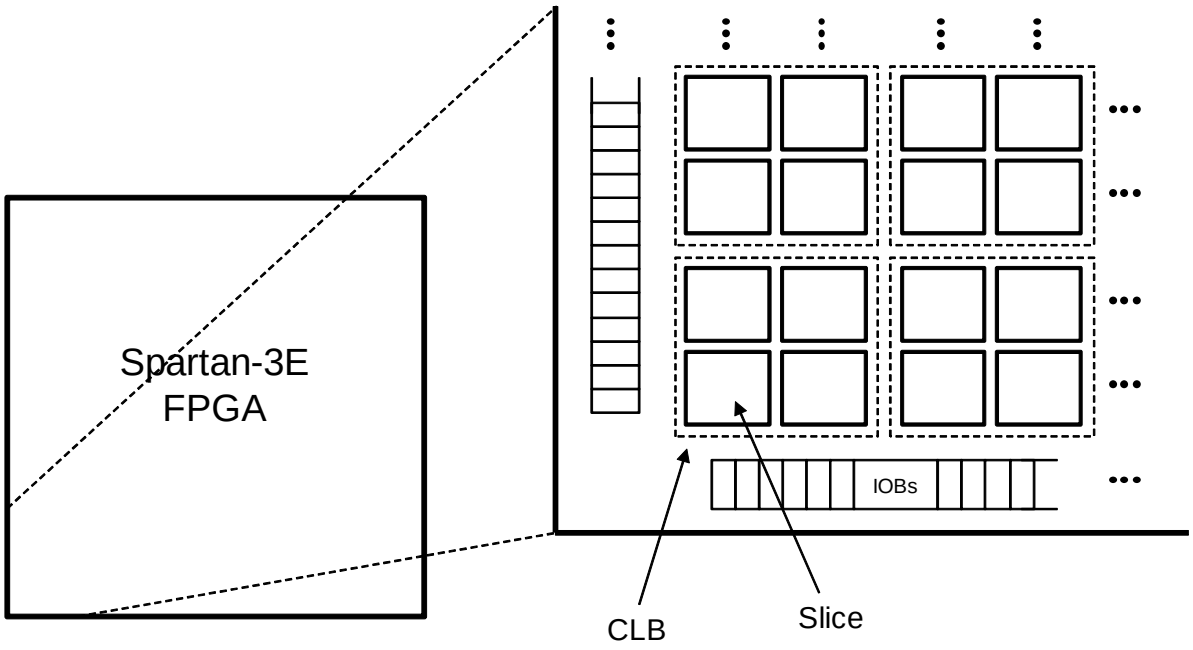

図 2.1　Spartan-3E の CLB

FPGA の代表的なベンダーとして Altera 社、Xilinx 社がある．本研究では、Xilinx 社の FPGA である Spartan-3E をターゲットとし、Ultrasmall の回路面積の削減を主な目的とした最適化を行う．以降、本節では、Spartan-3E のアーキテクチャについて述べる．

Spartan-3E は、CLB（Configurable Logic Block）と呼ばれる同期回路や組み合わせ回路をインプリメントするためのロジック・リソースを持つ [3]．1 つの CLB には 4 つの Slice が含まれ、Spartan-3E 内部では図 2.1 のように規則的に配置されている．

個々の Slice 内部は図 2.2 のようになっており、4 入力 LUT（Look-Up Table）が 2 つ、FF（Flip-Flop）が 2 つ、キャリー・チェイン（MULT_AND、XORCY、MUXCY）、マルチプレクサ（F5MUX、FiMUX）が含まれている．これら LUT、FF、MULT_AND、F5MUX などは FPGA プリミティブと呼ばれ、Verilog-HDL や VHDL などの HDL（Hardware Description Language）から、論理合成ツールが使用するか否かを推論する．

なお、FPGA における回路面積の削減とは、FPGA 上に実装した回路の使用 Slice 数を削減することと同義である．

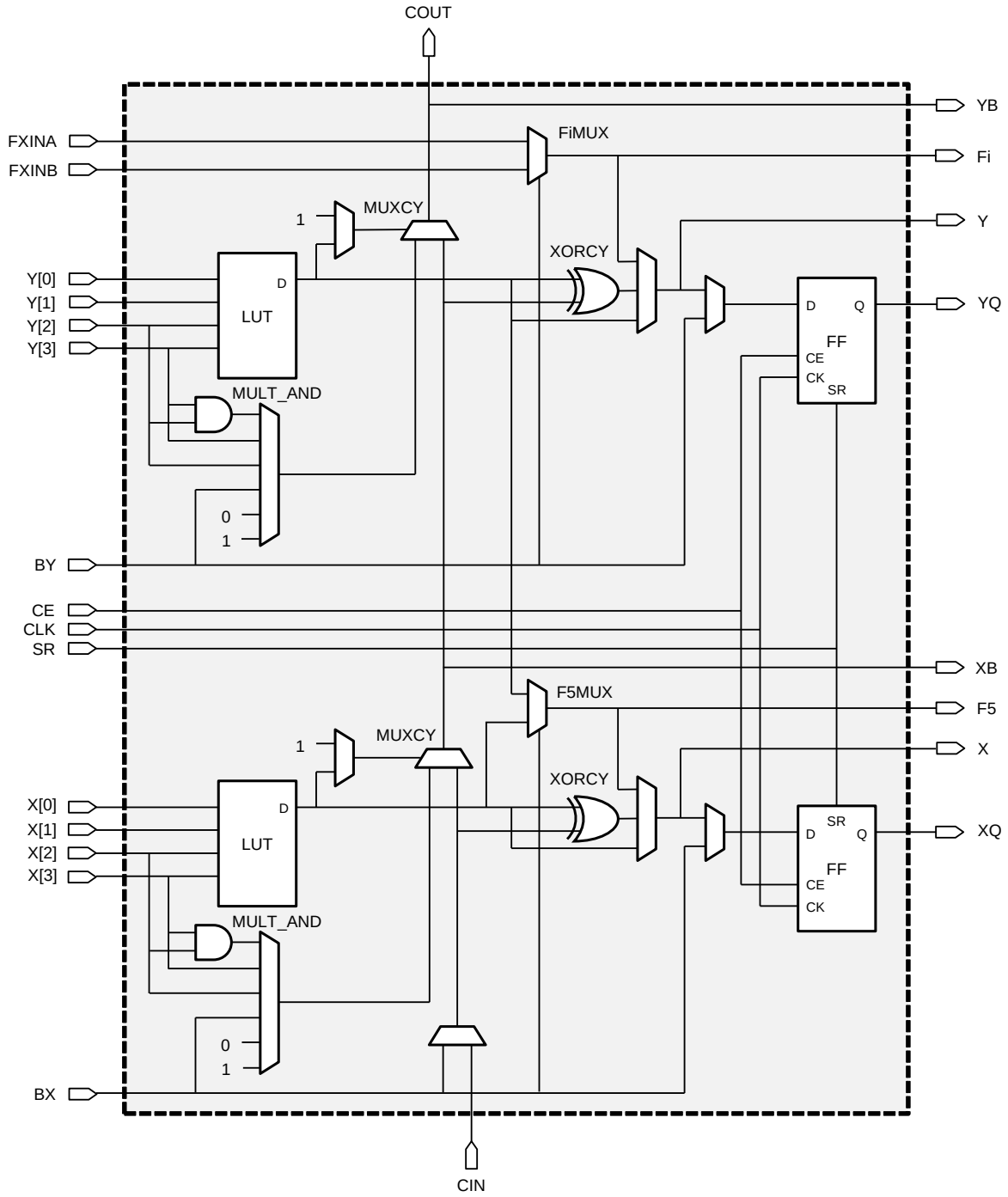

図 2.2 Spartan-3E の Slice

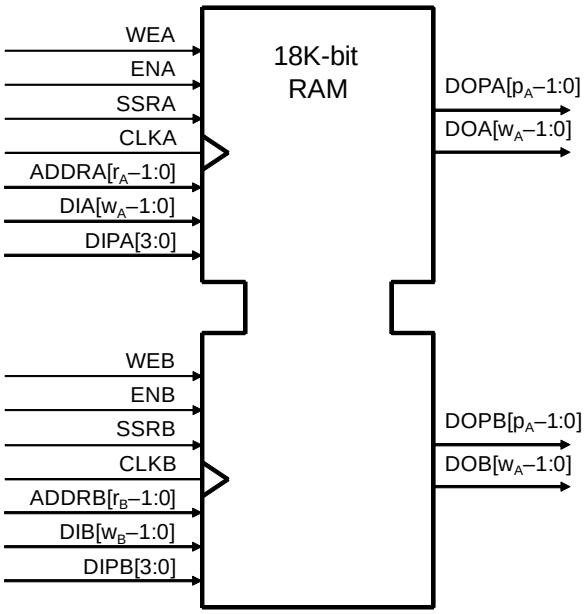

図 2.3　Spartan-3E の BlockRAM

FPGA は、Slice に配置される FF のほか、レジスタファイルやキャッシュなど、比較的大きな記憶域の実装のために、BRAM（BlockRAM）と呼ばれる同期書き込み、同期読み込みのメモリを複数持っている．Spartan-3E の BRAM は、2K ビットのパリティを含む 18K ビットのメモリである [4]．図 2.3 に示されるように、2 つのポートを持ち、それぞれのポート幅は 1、2、4、8、16、32 ビットのいずれかに設定できる．それぞれのポートの出力レジスタにはクロック・イネーブル（ENA、ENB）とリセット（SSRA、SSRB）が用意されている．なお、比較的最近の FPGA にみられる、バイト（8 ビット）単位での書き込みをサポートするような制御信号は、Spartan-3E の BRAM には備わっていない．

2.2　ソフトプロセッサ

ソフトプロセッサとは、FPGA を始めとするプログラマブルロジックデバイスを対象として、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、異なる構成、機能を持ったプロセッサを作成でき、ハードウェア的な変更を加える必要がない．

従来のマイコンベースの機器開発において、柔軟性の欠如が問題として挙げられる．大量生産を前提とした ASIC では、任意のユーザの要求に応じた最適なデバイスを生産することは困難である．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 1 個から実現することが可能である．そのため、近年、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の実装に用いられる．

提案されているソフトプロセッサは、Linux が動作するような大規模なものから、ステートマシンなどのコントローラとして用いられることを想定した小規模なもの、教育・研究用に設計されたものなど様々である．ソフトプロセッサの例と、その特徴を以下に示す．

PicoBlaze

PicoBlaze[1] は、Xiinx 社が自社の FPGA 向けに最適化したソフトプロセッサである．FPGA プリミティブを直接インスタンス化する記述が用いられている．

- 命令セット：独自 ISA
- レジスタのビット数：8 ビット

Yukiyama

Yukiyama[5] は、手動で LSI レイアウト可能であることを目指して開発されたソフトプロセッサである．面積を抑える手段として 1 ビットシリアル演算を用いている．

- 命令セット：独自 ISA
- レジスタのビット数：8 ビット

Leros

Leros[6] は、プロセッサ内部にデータメモリを搭載し、300 以下のロジック・セル [3]（LUT と記憶エレメントを組み合わせたもの）で実現することを目的として開発されたソフトプロセッサである．低価格な FPGA をターゲットとしている．

- 命令セット：独自 ISA
- レジスタのビット数：16 ビット

Supersmall Soft Processor

Supersmall Soft Processor[7] は、主要なデータパスを 1 ビットにすることで、回路面積を抑えたソフトプロセッサである．32 ビットの MIPS 命令を実行する．Altera 社の FPGA をターゲットとしている．

- 命令セット：MIPS I 命令セットのサブセット
- レジスタのビット数：32 ビット

Ultrasmall Soft Processor

Ultrasmall Soft Processor[2] は，Supersmall Soft Processor の主要なデータパスを 1 ビットから 2 ビットに変更し、性能向上させつつも、回路面積の増加を抑えたソフトプロセッサである．アーキテクチャの詳細は次節で説明する．

- 命令セット：MIPS I 命令セットのサブセット
- レジスタのビット数：32 ビット

ZPU

ZPU[8] は、スタックマシン型のアーキテクチャを採用することにより、32 ビットの命令セットを持ちながらも回路面積を抑えたソフトプロセッサである．独自の命令セットではあるが、GCC toolchain が提供されている．

- 命令セット：独自命令セット
- レジスタのビット数：32 ビット

Plasma

Plasma[9] は、非アライメントロード・ストア命令を除く、MIPS I のユーザ・モードの全命令を実装したソフトプロセッサである．2 段、もしくは 3 段パイプラインを選択できるものが、OpenCores にて公開されている [10].

- 命令セット：MIPS I 命令セットのサブセット
- レジスタのビット数：32 ビット

MicroBlaze

MicroBlaze[11] は，Xilinx が自社の FPGA 向けに提供するソフトプロセッサである．3 段、もしくは 5 段のパイプラインのものを選択できる．MMU を持ち，PetaLinux[12] が動作する．

- 命令セット：独自命令セット
- レジスタのビット数：32 ビット

MicroBlaze Micro Controller System (MCS)

MicroBlaze MCS[13] は，MicroBlaze から MMU などの機構を除き、小規模なコントローラとして Xilinx 社が提供しているソフトプロセッサである．3 段のパイプライン構成となっている．

- 命令セット：独自命令セット
- レジスタのビット数：32 ビット

2.3　Ultrasmall Soft Processor

Ultrasmall は、32 ビット MIPS I の命令サブセットを実行するノンパイプラインのソフトプロセッサであり、Xilinx 社の FPGA をターゲットとしている．Ultrasmall では、回路面積を抑えるために、主なデータパスに 2 ビットシリアルアーキテクチャを用いてい

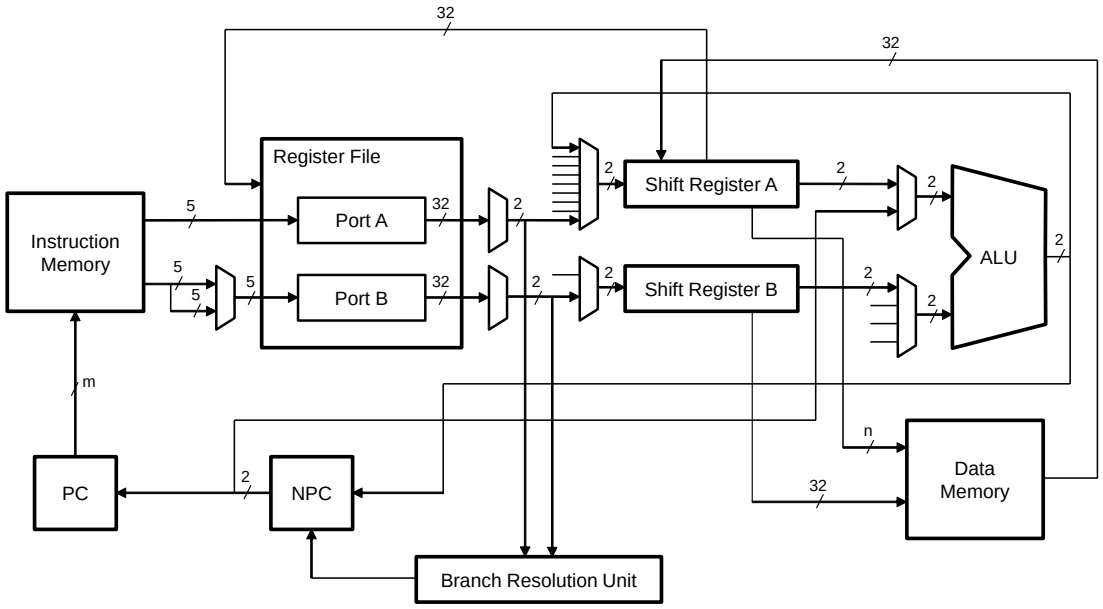

図 2.4 Ultrasmall のデータパス

る. 2 ビットシリアルアーキテクチャとは、レジスタからデータを取り出すときや、ALU（Arithmetic and Logic Unit）で演算するときなど、1 つの処理に 2 ビットずつ複数サイクルをかけて実行する方式である．MIPS I のデータは 32 ビット単位で扱われるため、Ultrasmall の場合、レジスタからデータを取り出すのに 16 サイクル、ALU で演算するのに 16 サイクルを必要とする．このようなアーキテクチャにより、データパスに流れるデータ量が 2 ビットとなるので、加算命令を全加算器 2 つで実現できるなど、ビットパラレルなアーキテクチャに比べ回路面積を大きく削減できる．一方で、2 ビットずつの処理しか行えないので、1 つの処理に複数サイクルを要し、性能が低下するというデメリットも存在する．

Ultrasmall のデータパスを図 2.4 に示す．2 ビットシリアルアーキテクチャを実現するために、Ultrasmall のアーキテクチャで大きな役割を担っているのが Shift Register A、B である．それぞれは 32 ビットの FF で構成され、内部のデータは 2 ビットずつシフトする．ここで、実際の MIPS 命令を用いて、Ultrasmall のデータフローを簡単に説明する．

MIPS 命令の一例とそれに対応する命令フォーマット、命令バイナリを図 2.5 に示す．この MIPS アセンブリ命令は、$3 レジスタと$2 レジスタの値の和をとり、結果を$6 レジスタに格納するものである．

命令は Instruction Memory から、PC（Program Counter）の値をアドレスとして読み

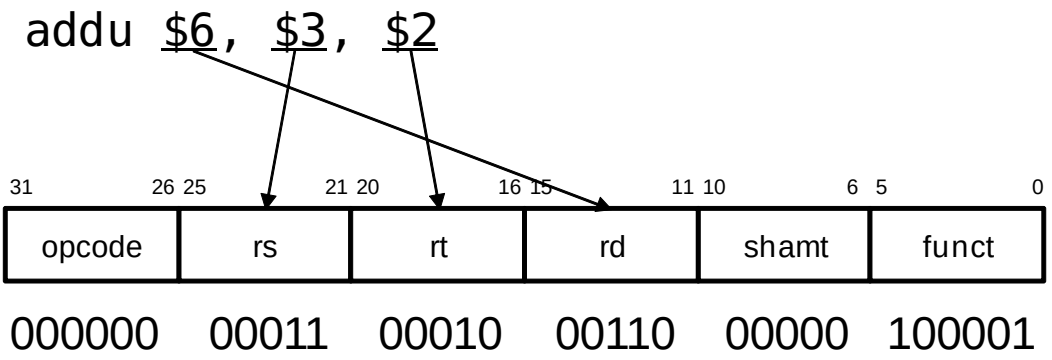

図 2.5　MIPS 命令の一例

出される．命令が読み出されると、Register File の Port A から$3 レジスタ、Port B から$2 レジスタの 32 ビットの値が出力される．Register File から出力された 2 つの 32 ビット値は、マルチプレクサにより 2 ビットが選択され、16 サイクルかけて Shift Register A、B に送られる．この際、Branch Resolution Unit は、Register File から送られる 2 ビットのデータを監視しており、命令に応じて NPC（Next Program Counter）への入力を切り替える．ただ、今回の場合は分岐命令ではないので、Branch Resolution Unit が条件分岐フラグを立てることはない．Shift Register へのデータ転送が終わると、2 つの Shift Register の出力の加算を ALU を用いて行う．Shift Register は、格納されたデータの LSB 側から、2 ビットずつ出力する．ALU での加算も 2 ビットずつ、16 サイクルかけて行われ、結果は、Shift Register A に MSB 側から格納される．Shift Register の出力は格納されたデータの LSB 側から、入力は MSB 側から行われるので、データが破壊されることはない．ALU での加算が終わると、最後に、Shift Register A に格納された 32 ビットの演算結果を$6 レジスタに転送して、addu 命令の実行が完了する．

上記の例では addu 命令を扱ったので、Data Memory を用いなかった．Data Memory にアクセスする場合は、Shift Register A の値でアドレスを指定し、ストアする際には Shift Register B の値を Data Memory へ書き込む．ロード命令の場合は、まず、Shift Register A に Data Memory から読み出した 32 ビットの値を書き込む．その後、命令で指定されたレジスタに Shift Register A の値を書き込む、といったデータフローとなる．

2 ビットシリアルアーキテクチャのほか、Ultrasmall のもう一つの特徴として、リソース・シェアリングを多用していることがあげられる．例えば、次に実行する命令のアドレスである NPC の値の計算は、加減算命令や論理演算命令で用いる ALU で行っている．このほか、2 ビットシリアルアーキテクチャを用いることにより必要となるサイクル・カ

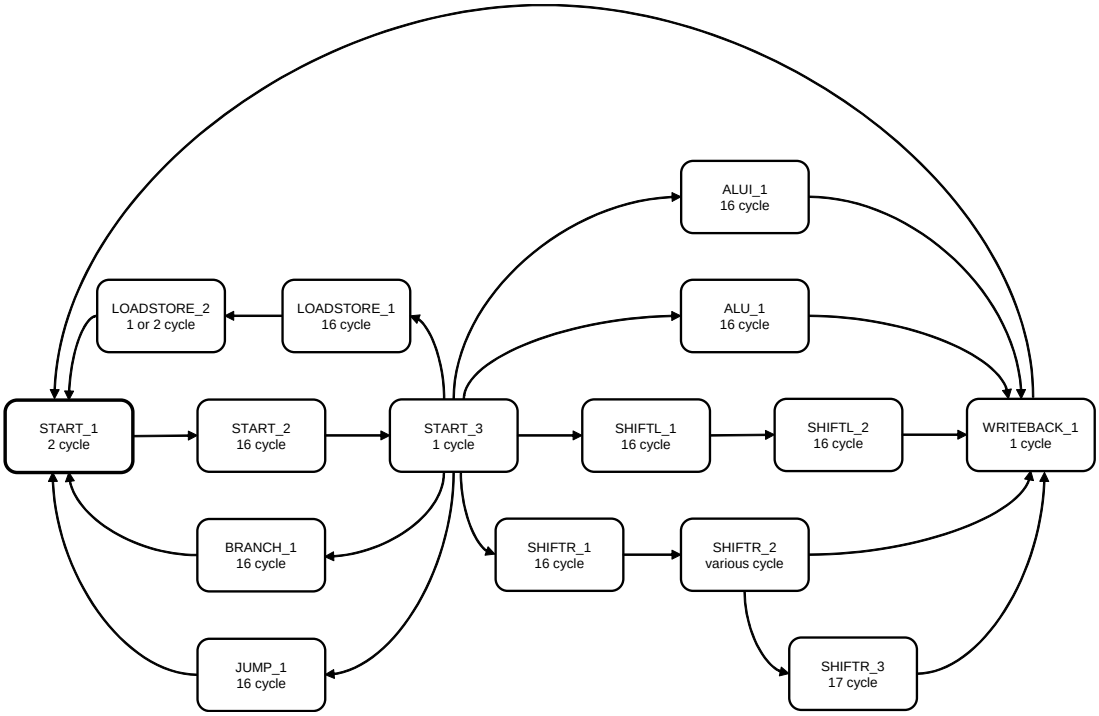

図 2.6 Ultrasmall の状態遷移図

ウンタも、複数のモジュールで共有されている（2 ビットシリアルアーキテクチャ自体も一種のリソース・シェアリングであるといえる）．このように、1 つのモジュールを使い回すことで、回路面積の削減が期待できる．

一方で、1 つのモジュールを複数のモジュールでシェアすることにより、ステートの増加につながることがある．例えば、Ultrasmall は、次の命令アドレスの計算と加減算命令などで用いる ALU を共有していると先ほど述べたが、これは、命令アドレスの計算と加減算命令が同じステートで実行できないことを意味する．2 ビットシリアルアーキテクチャの採用に加え、リソース・シェアリングの多用により、Ultrasmall のステートマシンは複雑なものとなっている．Ultrasmall の状態遷移図を図 2.6 に示す．なお、プロセッサの初期状態でのステートは、START_1 ステートである．また、各状態名の下の値は、必要なサイクル数を示している．

図 2.5 の命令の場合、START_1（命令、レジスタフェッチ）→START_2（PC、NPC の更新など）→START_3（命令デコード）→ALU_1（加算の実行）→WRITEBACK_1（レジスタへのライトバック）と状態を遷移し、命令を実行するのに計 36 サイクルを要する．命令の実行が完了すると START_1 ステートに遷移し、次の命令をフェッチする．

第 3 章

Ultrasmall Soft Processor の回路面積削減と高速化

本章では、Ultrasmall Soft Processor の回路面積を削減し、Cycles per instruction(CPI) を改善する 3 つの手法を提案する．これらの手法は Xilinx 社の FPGA である Spartan-3E のアーキテクチャの特徴を利用している．まず、これらの手法について述べた後、手法を適用し、Spartan-3E 向けに最適化された Ultrasmall Soft Processor の全体構成について述べる．

3.1 一部データパスの 32 ビット化

Ultrasmall は 2 ビットシリアルアーキテクチャを採用しているため、主要なパスは 2 ビット幅で構成されている．本節では、Ultrasmall の Register File から Shift Register A、B へのパスに注目し、このパスの幅を 2 ビットから 32 ビットに変更する．それに伴ない，不要となったマルチプレクサを除くことで，回路面積を削減する手法を提案する．なお、Register File から Shift Register A へのパスの 32 ビット化と、Register File から Shift Register B へのパスを 32 ビット化する手法は、A、B の入力の違いから大きく異なっている．よって、本節では提案手法をこの 2 つに分けて説明する．また、Register File から Shift Register へのパスを 32 ビットにすることで、条件分岐命令を処理する Branch Resolution Unit の接続、ならびにステートマシンに変更を加える必要がある．これについても節を設けて説明する．

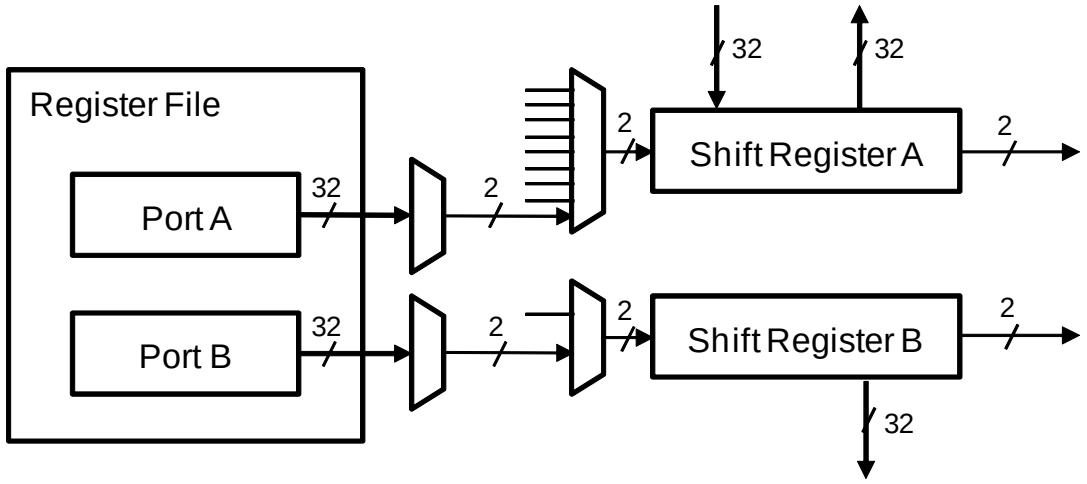

図 3.1　Shift Register へのデータパス

3.1.1　Register File から Shift Register B へのパスの 32 ビット化

Ultrasmall の Register File から Shift Register へのデータパスを図 3.1 に示す．Register File の 32 ビット出力は，まず 32:2 マルチプレクサにより 2 ビットが選択される．さらに、複数のモジュールからの出力を選択するマルチプレクサを通って、Shift Register の入力に入る．Shift Register B への入力は、この 2 ビットシリアル入力のみである．

Ultrasmall を論理合成し、FPGA へ配置・配線すると、Shift Regiseter B の入力の回路は、図 3.2 のように構成される．2 章で、FPGA の 1 つの Slice には、2 つの LUT と 2 つの FF が含まれると述べた．図 3.2 を見ると、2 ビットシリアル入力が接続されているものを除く，すべての Slice で FF は 2 つとも使用されているが、LUT は使用されていない．表 3.1 に、32:2 マルチプレクサと、Shift Register B のリソース使用量を示す．表 3.1 によれば、Shift Register B では 17 個の Slice が使われており、それらの Slice には計 34 個の LUT が含まれている．しかし、LUT はそのうち 4 つしか使用されておらず、多くの LUT が無駄になっている．重要なのは、無駄になっているこれらの LUT を活用しても、使用 Slice 数は増加しないということである．

Register File から Shift Register B へのパスを 32 ビット化することで、32:2 マルチプレクサが必要なくなる．その代わり、Shift Register B の入力に、2 ビットシリアル入力のほか、Register File からの 32 ビットパラレル入力が追加される．

Register File から Shift Register B へのパスを 32 ビット化したときの Shift Register B

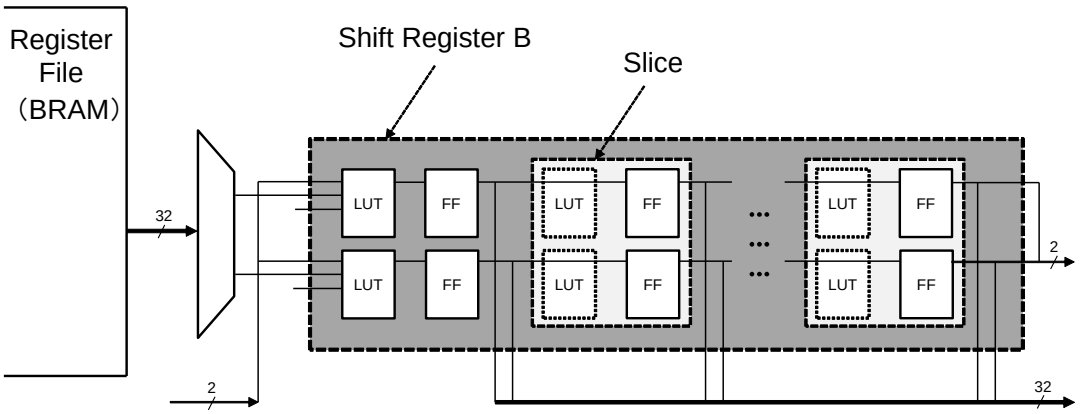

図 3.2　従来の Shift Regsiter B の入力の回路

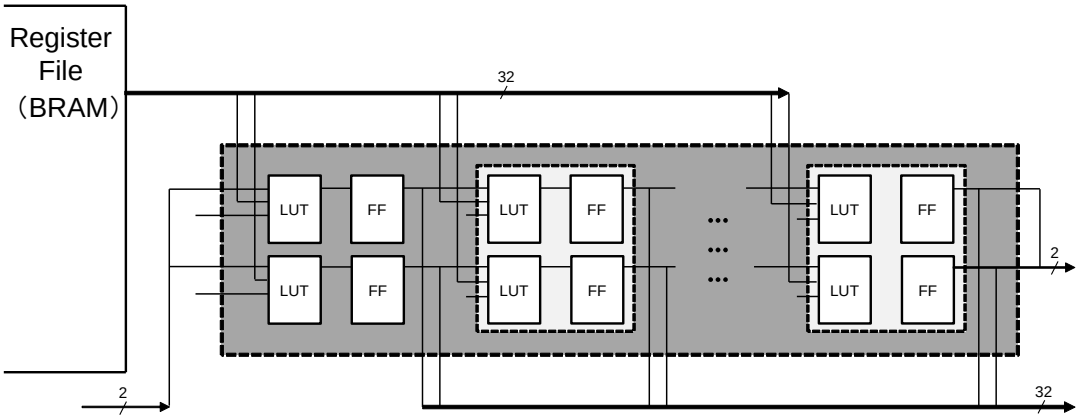

図 3.3　32 ビット化後の Shift Regsiter B の入力の回路

の入力の構成を図 3.3 に示す．パスの 32 ビット化により、図 3.3 では、Shift Register 内部でのシフトと、Register File からの 32 ビットパラレル入力のいずれかを選択する 2:1 マルチプレクサが追加されている．2:1 マルチプレクサは 2 本の被選択信号と 1 本の選択信号の計 3 本の入力を持つ．よって、2:1 マルチプレクサは 1 つの 4 入力 LUT で実装することができ、無駄になっていた LUT を活用することができる．従って、Shift Register B に用いる Slice 数の増加を抑えつつ、32:2 マルチプレクサのロジックを削減できる．本手法を適用した後の 32:2 マルチプレクサと Shift Register B のリソース使用量を表 3.2 に示す．表 3.1 と表 3.2 の合計を比べると、提案手法により使用 LUT 数は増えているが、狙い通り使用 Slice 数が削減されているのがわかる．

表 3.1　従来のリソース使用量

	32:2 マルチプレクサ	Shift Register B	合計
Slice	8	17	25
LUT	16	4	20
Reg	0	32	32

表 3.2　32 ビット化後のリソース使用量

	32:2 マルチプレクサ	Shift Register B	合計
Slice	0	16	16
LUT	0	32	32
Reg	0	32	32

3.1.2　Register File から Shift Register A へのパスの 32 ビット化

Ultrasmall において、Shift Register A は、2 ビットシリアル入力と、Data Memory からの 32 ビットパラレル入力を持つ．Shift Register A の入力のビット幅は、パスを 32 ビット化した Shift Register B と同じであり、FPGA 内での構成もこの 2 つはまったく同じとなる．パスの 32 ビット化された Shift Register B は、Shift Register 内部でのシフトと、Register File からの 32 ビットパラレル入力のいずれかを選択する 2:1 マルチプレクサを持つ．同様に、Shit Register A は、Shift Register 内部でのシフトと、Data Memory からの 32 ビットパラレル入力のいずれかを選択する 2:1 マルチプレクサを持っている．

単純に、3.1.1 節と同様に、Shift Register A の入力のパスを 32 ビット化し、32:2 マルチプレクサを取り除く実装を考えると、Register File からの 32 ビットパラレル入力、Data Memory からの 32 ビットパラレル入力、Shift Register 内部でのシフトのいずれかを選択する 3:1 マルチプレクサが必要となる．この実装前後の Shift Register A のリソース使用量を表 3.3 に示す．表 3.3 を見ると、2:1 マルチプレクサを 3:1 マルチプレクサに変更したことにより、Shift Register A の使用 Slice 数は増加している．これは、3:1 マルチプレクサは通常、構成するのに 4 入力 LUT を 2 つ以上必要とすることに起因する．なぜなら、3:1 マルチプレクサは、3 本の被選択信号と、2 本の選択信号の計 5 本の入力を持つからである．

表 3.3　Shift Register A のリソース使用量の変化

	従来の実装	3:1 マルチプレクサでの実装
Slice	17	32
LUT	32	64
Reg	32	32

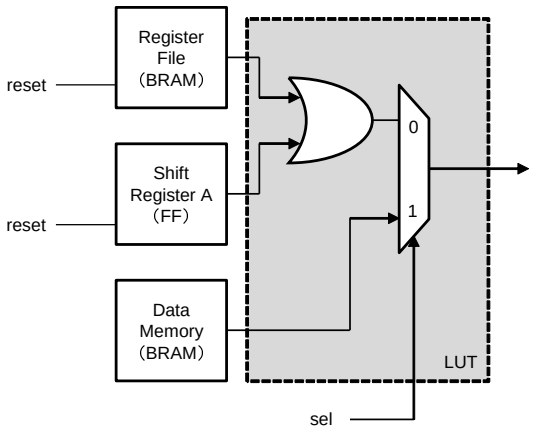

図 3.4　32 ビット化後の Shift Regsiter A の LUT

FF と BRAM の出力レジスタのリセット信号を用いたある特別な方法を用いて、3:1 マルチプレクサを 1 つの 4 入力 LUT で実装する．その特別な方法を用いて 3:1 マルチプレクサを実装した、Shift Register A の LUT を図 3.4 に示す．

図 3.4 の LUT は、Register File の出力と Shift Register の内部シフトの OR をとったものと、Data Memory の出力をマルチプレクサで選択している．Register File の出力を Shift Register A に読み込みたい場合は、Shift Register A の FF をすべてリセットし、マルチプレクサは OR をとったものを出力するよう選択信号を 0 にする．リセットにより内部シフトの信号は 0 となるので、結局、Register File の出力が Shift Register A に読み込まれる．内部シフトをしたい場合は、Register File を構成する BRAM の出力レジスタをリセットすれば、同様に可能である．Data Memory の出力を読み込む場合は、マルチプレクサの選択信号を 1 にすればよい．

このように、FF、BRAM の出力レジスタのリセットを用いることで、3:1 マルチプレクサを 1 つの 4 入力 LUT で実装できる．これにより、3.1.1 節と同様、Shift Register A の使用 Slice 数の増加を抑えつつ、32:2 マルチプレクサを削減することができる．

3.1.3 Branch Resolution Unit の接続とステートマシンの変更

Ultrasmall の Branch Resolution Unit は、Register File から Shift Register へ送られる 2 ビットのデータを監視し、条件分岐のフラグを計算する．この計算は、状態遷移図（図 2.6）における START_2 ステートで、Register File から Shift Register へのデータ転送、PC の更新、NPC の更新と同時に、16 サイクルかけて行われる．しかし、3.1.1 節、3.1.2 節で説明した手法により、Register File から Shift Register A、B へのパスが 32 ビットとなると、このようなフラグの計算は不可能となってしまう．

この問題を解決するため、データパスを図 3.5 のように、Branch Resolution Unit が Shift Register A、B の 2 ビット出力を参照するよう変更する．さらに、ステートマシンにも手を加え、条件分岐のフラグを 16 サイクルかけて計算するステート（BRANCH_0）を新たに追加する．変更した状態遷移図を図 3.6 に示す．

データパスの変更は配線の付け替えのみで実現でき、新たにロジックを消費することは考えにくい．一方で、ステートの追加は、ステートマシンをより複雑にし、制御回路の使用 Slice 数増加が見込まれる．結果としては、32 ビット化により減少した Slice の方が制御回路で増加した Slice より数が多く、全体でみると使用 Slice 数は減少した．実際に減少した数の詳細などは 4 章の評価で述べる．

以降、本論文では、Register File から Shift Register へのパスを 32 ビット化した Ultrasmall のことを Ultrasmall α と記述する．

3.2 過剰なリソース・シェアリングの除去

一部データパスを 32 ビット化した Ultrasmall α は、BRANCH_0 ステートの増加により、条件分岐命令を実行するのに従来の Ultrasmall に比べ 16 サイクル余分に要する．よって、Ultrasmall α は、Ultrasmall に比べ、性能が低下している．本節では、3.1 のアーキテクチャにさらに変更を加え、Ultrasmall α の回路面積を削減しつつも、プロセッサ性能を向上させる手法を提案する．この手法を適用した Ultrasmall α は、Ultrasmall より性能が向上している．

Ultrasmall α の状態遷移図（図 3.6）において、START_2 ステートでは PC の更新と NPC の更新を 16 サイクルかけて行っている．PC の更新は、命令の実行が終わる（START_1 ステートに戻ってくる）までに完了していればよい．よって、16 サイクルを要

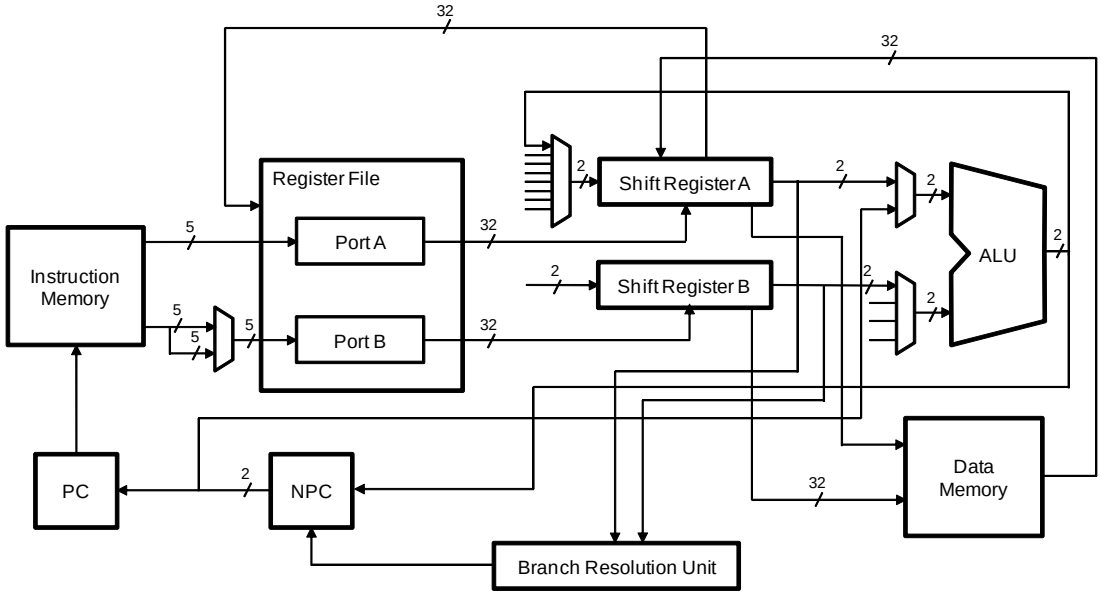

図 3.5　Ultrasmall α のデータパス

する ALU_1、ALUI_1、BRANCH_0、SHIFTL_1、SHIFTR_1、LOADSTORE_1 といったステートでも PC の更新は行うことができる．

2 章で述べたように、Ultrasmall は、加減算命令や論理演算命令で用いる ALU で NPC の値を計算する．Ultrasmall α でもこの方式を踏襲しているが、この場合、PC の更新と同様に、NPC の更新を ALU_1、ALUI_1 や LOADSTORE_1 で行うことはできない．なぜなら、ALU_1、ALUI_1、LOADSTORE_1 では論理演算やアドレス計算に ALU を用いており、同じ ALU で NPC の値を計算しようとすると競合するからである．

本手法では、この ALU のリソース・シェアリングを取り除き、16 サイクルかけて NPC の値を計算する 2 ビット Adder（以降、本論文では NPC Adder と記述する）を Ultrasmall α に追加する．後述するように、この変更では、データパスの回路面積は増加しないことが期待される．また、この NPC Adder の追加により、PC の更新と同様、NPC の更新も START_2 以外のステートで行うことができ、START_2 ステートを取り除くことができる．

Ultrasmall、ならびに Ultrasmall α の ALU への入力には、図 3.7 のように 4 本の 2 ビット信号から 1 本を選ぶマルチプレクサと、2 本の 2 ビット信号から 1 本を選ぶマルチプレクサが接続されている．

NPC Adder を追加すると、これらの 6 本の信号は図 3.8 のように接続される．単純に図 3.7 と図 3.8 のロジックの差分をとると、2 ビットの 4 入力マルチプレクサと NPC

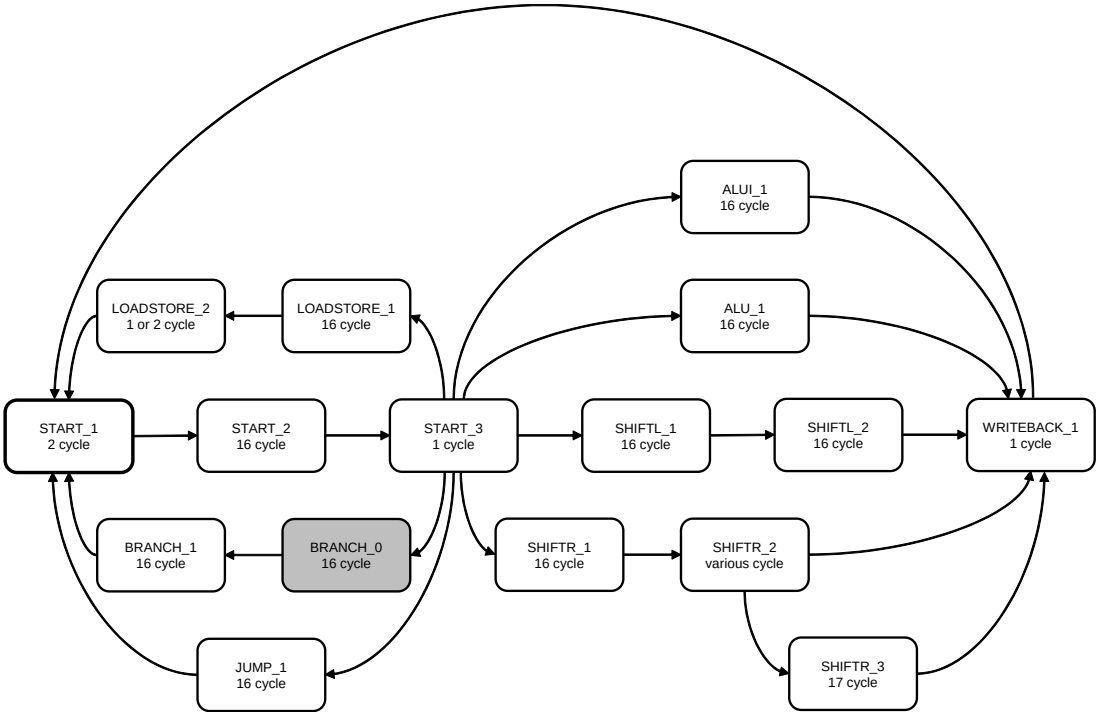

図 3.6 Ultrasmall α の状態遷移図

Adder が異なっている．4:1 マルチプレクサは、3:1 マルチプレクサと同様、1 ビットあたり 2 つの LUT、つまり、1 つの Slice を使用する．よって、2 ビットの 4 入力マルチプレクサの実装には 2 つの Slice を使用する．一方で、3.3 節で詳しく説明するように、NPC Adder もキャリー・チェインを用いることで 2 つの Slice で実装することができる．よって、NPC Adder の追加では、データパスの回路面積は増加しないことが期待される．

しかし、実際には、NPC Adder を追加した Ultrasmall α を論理合成し、FPGA へ MAP すると、データパスの回路面積は増加した．これは、論理合成ツールが、NPC Adder を予期した回路とは違う回路を生成したことに起因する．なお、NPC Adder の追加によりデータパスの回路面積は増加したが、増加した Slice の数以上に、START_2 ステートの除去により、制御回路の使用 Slice 数が減少しており、全体で見ると使用 Slice 数は削減されている．

NPC Adder を追加し、START_2 ステートを除去した Ultrasmall α の状態遷移図を図 3.9 に示す．Ultrasmall に比べ、START_2 ステートが除去されたため、条件分岐命令を除くすべての命令で、実行に必要なサイクル数が 16 サイクル減少している．条件分岐命令については、16 サイクルを要する BRANCH_0 ステートの追加により、Ultrasmall と同

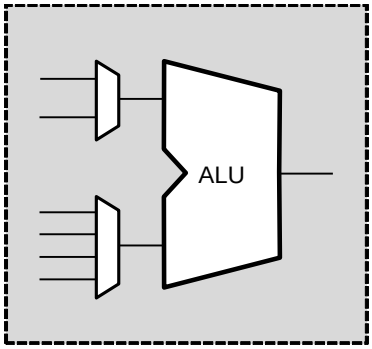

図 3.7　従来の ALU（パスはすべて 2 ビット幅）

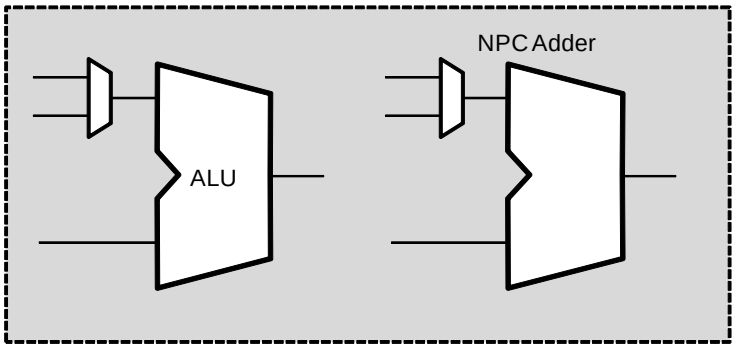

図 3.8　NPC Adder を追加した ALU（パスはすべて 2 ビット幅）

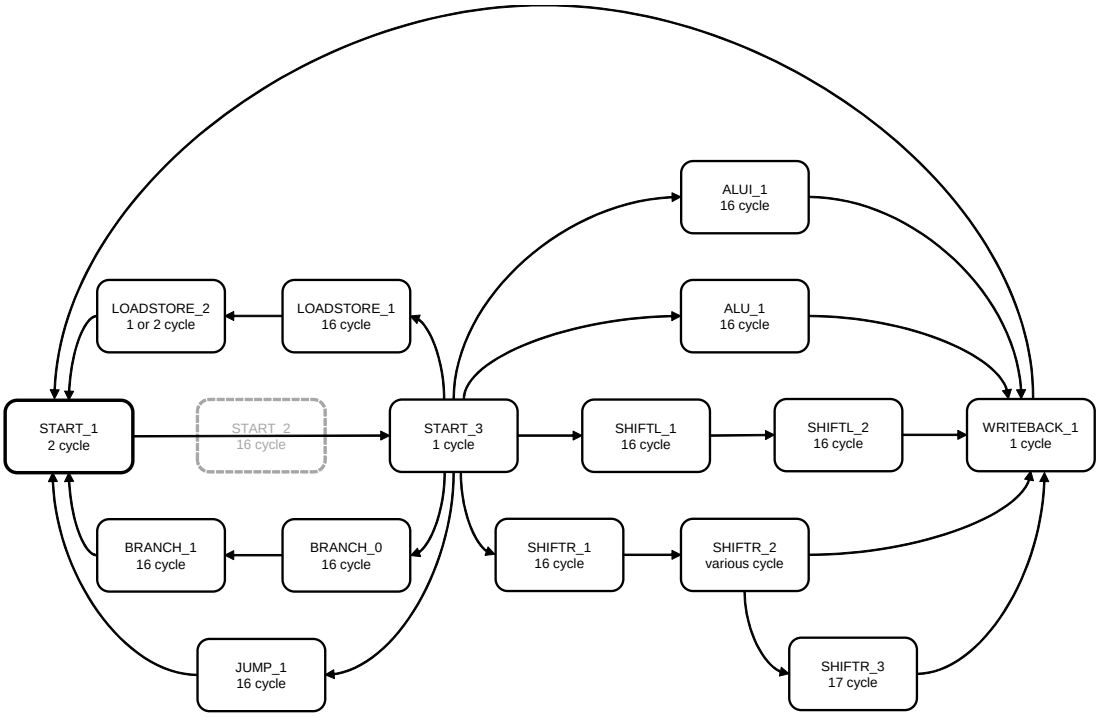

図 3.9　Ultrasmall β の状態遷移図

じサイクル数で実行する．このことから、本手法を適用した Ultrasmall α は、Ultrasmall に比べ、1 命令の実行に必要なサイクル数が減少していおり、性能が向上している．

以降、本論文では、NPC Adder を追加し、回路面積の削減と高速化をした Ultrasmall α を Ultasmall β と記述する．

```
1  module npc_adder
2      (clk,
3       reset,
4       npc_adder_in_0,
5       npc_adder_in_1,
6       npc_adder_carry_in,
7       npc_adder_out);
8
9      input        clk;
10     input        reset;
11     input        sel;
12     input  [1:0] npc_adder_in_0;
13     input  [1:0] npc_adder_in_1;
14     input        npc_adder_carry_in;
15     output [1:0] npc_adder_out;
16
17     reg          carry;
18
19     wire       npc_adder_carry;
20     wire [1:0] npc_adder_in_2 = (sel) ? npc_adder_in_1 : 2'b0;
21     wire       carry_in = npc_adder_carry_in | carry;
22
23     always @(posedge clk) begin
24         if(reset)
25             carry <= 1'b0;
26         else
27             carry <= npc_adder_carry;
28     end
29
30     assign {npc_adder_carry, npc_adder_out} =
31         npc_adder_in_0 + npc_adder_in_2 + carry_in;
32
33 endmodule
```

図 3.10　NPC Adder の Verilog HDL ソースコード

3.3 プリミティブ記述

3.2 節で、Ultrasmall α に追加した NPC Adder のソースコードを図 3.10 に示す．NPC Adder は、キャリーを伝搬するための FF を 1 つ持っている．この FF により、2 ビットずつ、16 サイクルかけて 32 ビットの加算をすることが可能となる．この回路は、Spartan-3E のキャリー・チェインを用いて 2 つの Slice で実装できる（図 3.11）．

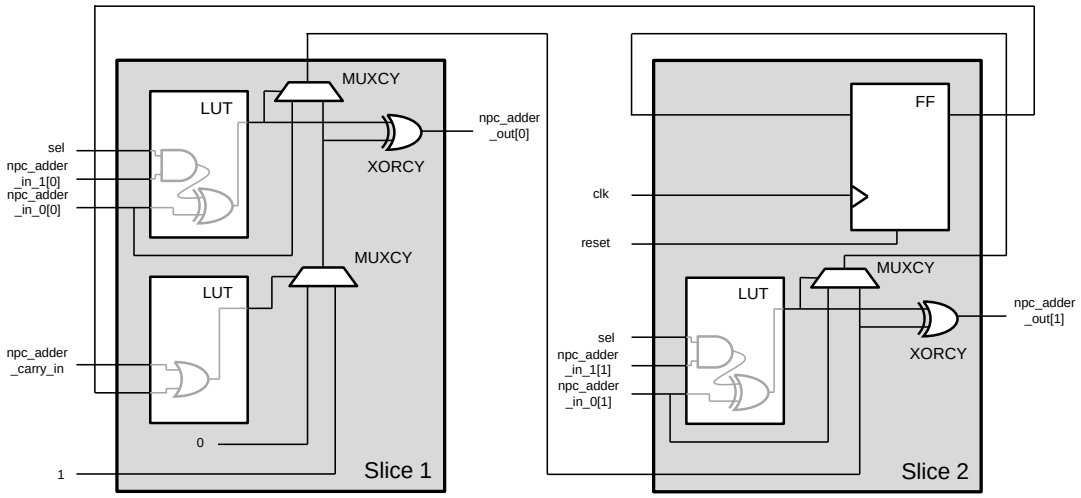

図 3.11　2 つの Slice で構成された NPC Adder

しかし、Ultrasmall β をエリア優先で論理合成すると、NPC Adder は 6 個の Slice を消費し、予期していた回路よりも大きくなっている．これは、論理合成ツールが、Slice 内部にあるキャリー・チェインをうまく扱えていないことに起因する．本節では、プリミティブ記述と呼ばれる、FPGA プリミティブを直接インスタンス化する方法を用いて、Ultrasmall β の回路面積を削減する手法を提案する．

プリミティブ記述の例として、4:1 マルチプレクサを図 3.12 に示す．特に RLOC 制約 [14] などで指定しなくとも、図 3.12 のようにプリミティブ記述を用いて FPGA プリミティブをインスタンス化すると、配置・配線ツールはこれらのプリミティブを 1 つの Slice 中に配置する [15].

プリミティブ記述を用いることで、論理合成ツールの最適化に頼ることなく、設計者が思い描いたとおりの回路を FPGA 上に実装可能である．論理合成ツールは、現在でも、キャリー・チェインとマルチプレクサの使い方が不得手なところがある [16]．よって、キャリー演算や多数のマルチプレクサを含むモジュールについては、論理合成ツールの最適化が不十分であれば、より最適な回路をプリミティブ記述することで回路面積の削減が期待できる．Ultrasmall β の NPC Adder のほか、データパス中の多段構成となっていたマルチプレクサを手動で最適化し、プリミティブ記述を用いることにより回路面積を削減した．

以降、本論文では、一部にプリミティブ記述を用い、回路面積を削減した Ultrasmall β を、Ultrasmall+ と記述する．

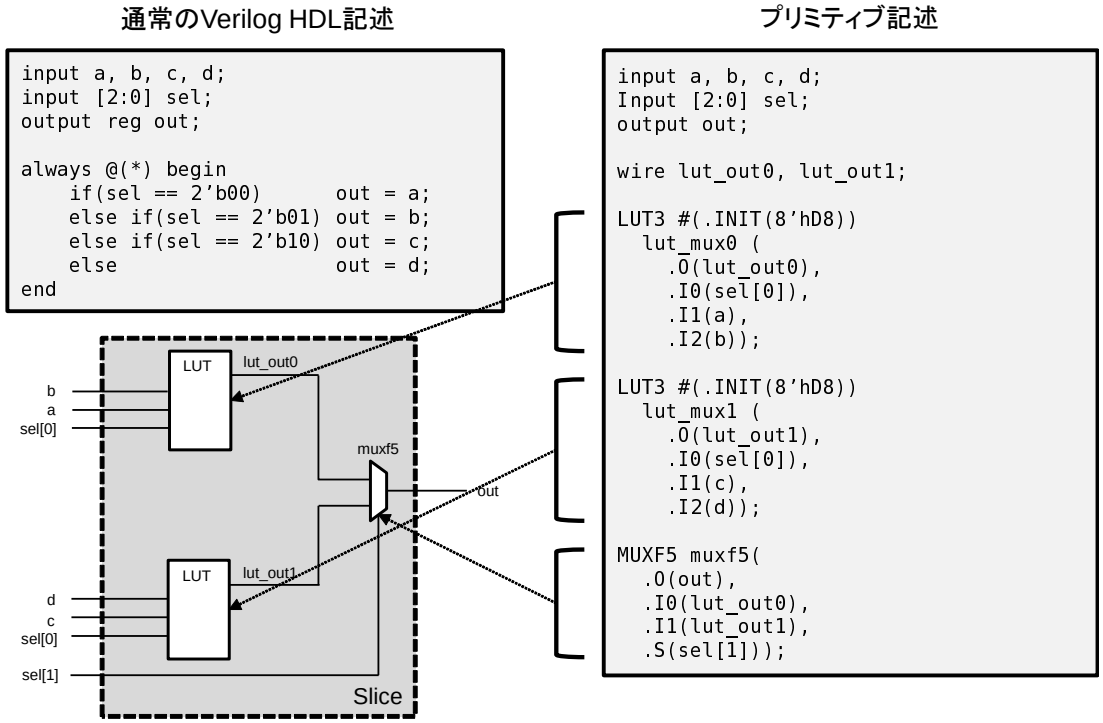

図 3.12　4:1 マルチプレクサのプリミティブ記述の例

3.4　Ultrasmall+ のアーキテクチャ

Ultrasmall+ のデータパスを図 3.13 に示す．図 3.13 のデータパスでは、Ultrasmall のデータパス（図 2.4）と比べ、Register File から Shift Register へのパスの 32 ビット化、NPC Adder の追加がなされている．Ultrasmall+ の状態遷移図は，Ultrasmall β と同じもの（図 3.9）となるので，ここでは省略する．

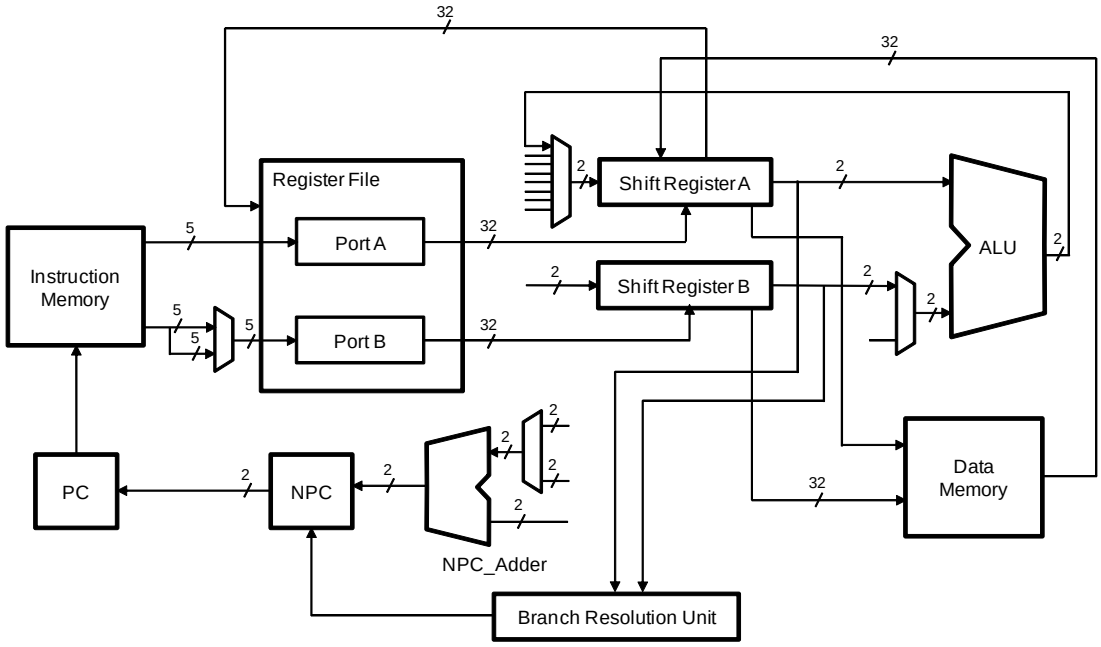

図 3.13　Ultrasmall+ のデータパス

第 4 章

評価

本章では、3.1 節、3.2 節、3.3 節の手法を適用し、最終的に提案した、Ultrasmall+ が Ultrasmall に比べ、面積、性能面で優位であることを示す．また、Ultrasmall+ と 2.2 節で述べたソフトプロセッサのリソース使用量を比較し、評価する．

4.1 評価環境

論理合成、MAP、配置・配線を行うソフトウェアはすべて、Xilinx 社の提供する ISE WebPACK[17]（ver. 14.7）を使用した．3 章で述べた Ultrasmall α、Ultrasmall β、Ultrasmall+ は Xilinx 社の Spartan-3E FPGA をターゲットとしている．ソフトウェアのターゲット・デバイスには Spartan-3E ファミリーの XC3S500E の VQ100 パッケージを用い、Speed Grade は-5 とした．また、本論文では回路面積削減が主なテーマであるため、最適化オプションもエリア優先で最適化を行うよう、「Optimization Goal」を「Area」に、「Optimization Effort」を「High」として評価を行った．

本章では、Ultrasmall と、各手法を適用した Ultrasmall α、Ultrasmall β、Ultrasmall+ の回路面積、動作周波数のほか、CPI についても評価している．Ultrasmall、Ultrasmall α、Ultrasmall β、Ultrasmall+ はノンパイプラインのソフトプロセッサであり、一部のシフト命令を除き、1 命令の実行に要するサイクル数は、命令ごとに一定である．シフト命令のうち、sra、srav といった右シフトを行うものは、シフト量に応じて要する実行サイクル数が変化する．

MIPS プロセッサのソフトウェアシュミレータである SimMips[18]（ver. 0.6.7）のソースコードを一部変更し、実行命令数だけでなく、Ultrasmall、Ultrasmall α、Ultrasmall β、Ultrasmall+ で要する総サイクル数も計算するようにした．この変更した SimMips を

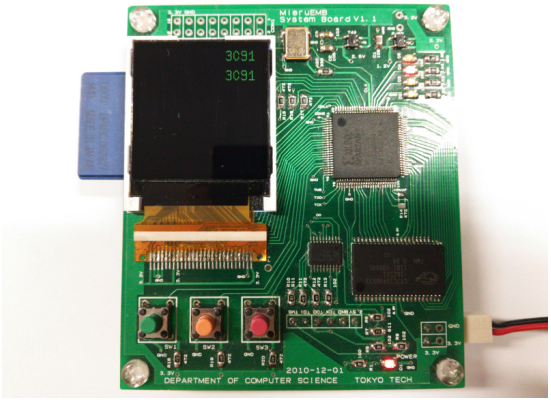

図 4.1　MieruEMB FPGA ボード

用いて、対象のアプリケーションをシミュレートし、実行命令数と必要総サイクル数から、各々の CPI を計算している．なお、Linux 上に構築した MIPS クロスコンパイル環境の gcc のバージョンは 4.3.6、コンパイルオプションは-O2 である．

4.2　動作確認

提案した Ultrasmall+ の動作確認を、東京工業大学の吉瀬研究室で開発された教育用途向けの FPGA ボード、MieruEMB[19] を用いて行った．MieruEMB は FPGA に XC3S500E を搭載し、そのほか、512KB の SRAM、128 × 128 ドットの LCD、SD カードスロットなどを備えている（図 4.1）．

また、動作確認用のアプリケーションには、Stanford Integer Benchmark[20][21] の一部アプリケーションを用いた．Ultrasmall+ を実装した MieruEMB 上で、Stanford Integer Benchmark に含まれる「Bubble Sort」「Permutations」「Eight Queens」「Quick Sort」「Towers of Hanoi」の 5 つのアプリケーションを実行し、LCD に正しく結果が出力されることを確認した．

4.3　ハードウェア量

表 4.1 に、以下の各手法を実装した場合と Ultrasmall とのリソース使用量の比較を示す．図 4.2 は、表 4.1 のうち、使用 Slice 数をグラフにしたものである．

Ultrasmall α

表 4.1 Ultrasmall と各手法を適用したもののリソース使用量の比較

	Slice	LUT	LUT RAM	Slice Reg	BRAM
Ultrasmall	163	244	4	141	9
Ultrasmall α	149	244	4	142	9
Ultrasmall β	143	243	4	132	9
Ultrasmall+	**137**	235	4	132	9

表 4.2 Ultrasmall+ とその他のソフトプロセッサのリソース使用量の比較

	Slice	LUT	LUT RAM	Slice Reg	BRAM
Plasma	1307	2557	256	239	0
MicroBlaze MCS	542	973	301	215	8
ZPU	367	719	0	165	8
Supersmall	164	253	8	164	9
Ultrasmall	163	244	4	141	9
Ultrasmall+	**137**	235	4	132	9

Ultrasmall の一部データパスを 32 ビット化したもの．

Ultrasmall β

Ultrasmall α の一部リソース・シェアリングを除去したもの．

Ultrasmall+

Ultrasmall β の一部モジュールをプリミティブ記述にしたもの．

最終的に提案した Ultrasmall+ と、2.2 節で例として示したソフトプロセッサのうち、レジスタのビット幅が 32 ビットのもののリソース使用量の比較を表 4.2 に示す．また、表 4.2 のうち、使用 Slice 数をグラフにしたものを図 4.3 に示す．なお、MicroBlaze は、MMU を搭載し、Linux が動作するなど Ultrasmall と趣旨が異なるため、MicroBlaze から MMU などを取り除いた MicroBlaze MCS を比較対象としている．また、データメモリを内部に持つソフトプロセッサは、データメモリサイズを 16KB として比較している．

Supersmall Soft Processor（以降、Supersmall）は Altera 社の 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．本研究では、Xilinx 社の

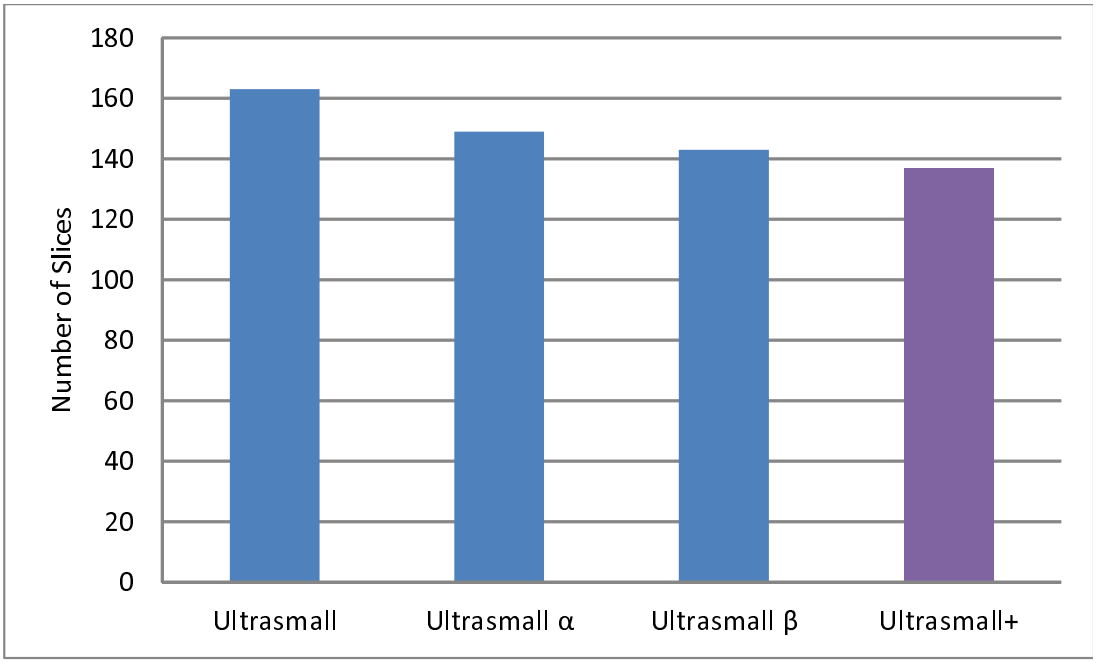

図 4.2　Ultrasmall と各手法を適用したもののリソース使用量の比較

Spartan-3E をターゲットとするため、Supersmall からこれらの命令処理部を除いて比較を行い、Ultrasmall+ では実装されていない例外処理部も除いている．

表 4.1 を見ると、使用 Slice 数が Ultrasmall、Ultrasmall α、Ultrasmall β、Ultrasmall+ と、徐々に小さくなっており、提案した 3 つの手法が有効であることがわかる．最終的に提案した Ultrasmall+ は、Ultrasmall に比べ回路面積を約 16% 削減している．また、その他のソフトプロセッサとのリソース使用量の比較（表 4.2）を見ても、使用 Slice 数は Ultrasmall+ が最も小さい．Ultrasmall は、32 ビット RISC 命令を実行するソフトプロセッサにおいて、世界最小のソフトプロセッサであった．従って、Ultrasamll+ は、Spartan-3E 上という条件を持ちながらも、世界最小のソフトプロセッサである．Ultrasmall+ を XC3S500E に配置・配線し、FPGA Editor で使用リソースを色分けしたものを図 4.4 に示す．図 4.4 中の赤い部分が使用 Slice であり、FPGA の内部でも比較的小規模に実装できていることがわかる．

4.4　動作周波数

表 4.3 に、各手法を実装した場合と Ultrasmall との動作周波数の比較を示す．この動作周波数は、FPGA に MAP した後の値ではなく、論理合成した直後の値を示している．図

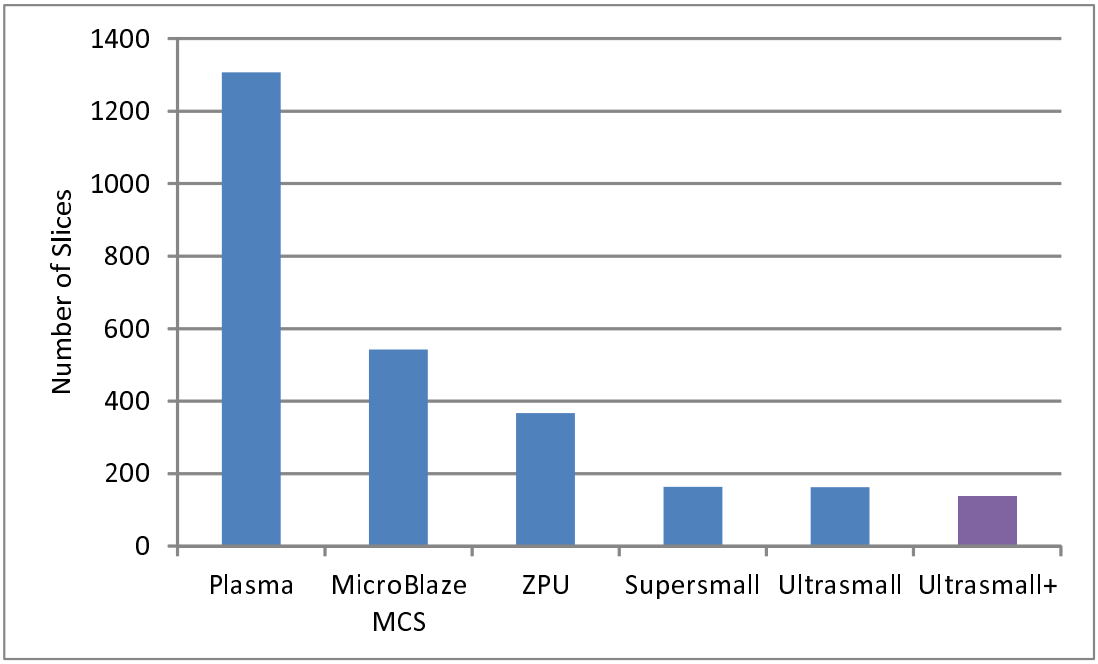

図 4.3　Ultrasmall+ とその他のソフトプロセッサのリソース使用量の比較

表 4.3　Ultrasmall と各手法を適用したものの動作周波数の比較

	動作周波数（MHz）
Ultrasmall	64.54
Ultrasmall α	64.72
Ultrasmall β	65.57
Ultrasmall+	**65.01**

表 4.4　Ultrasmall+ とその他のソフトプロセッサの動作周波数の比較

	動作周波数（MHz）
Plasma	20.41
MicroBlaze MCS	112.27
ZPU	113.71
Supersmall	70.37
Ultrasmall	64.54
Ultrasmall+	**65.01**

4.5 は、表 4.3 をグラフにしたものである．

また、Ultrasmall+ と、その他のソフトプロセッサとの動作周波数の比較を表 4.4 に、表 4.4 をグラフにしたものを図 4.6 に示す．ソフトプロセッサの構成などの条件は、4.3 節と同じである．

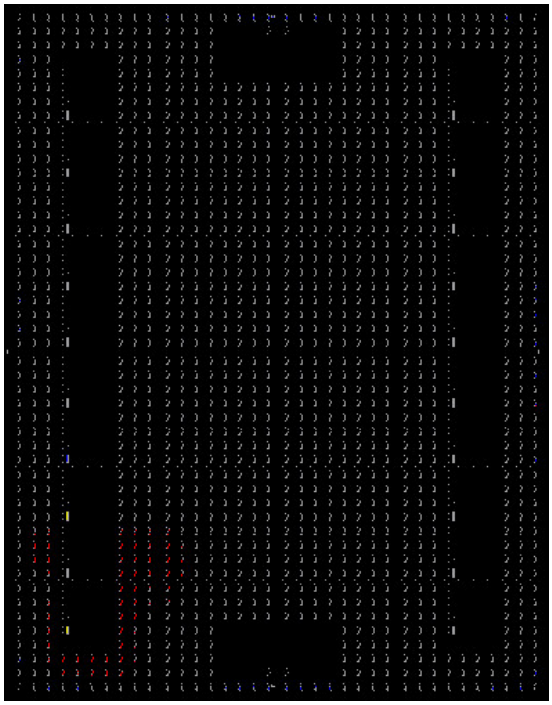

図 4.4　Ultrasmall+ を FPGA に実装したときのリソースの使用状況

4.5　Cycles per instruction

Stanford Integer Benchmark には、8 つのプログラムが含まれている．そのうち、「Bubble Sort」「Permutations」「Eight Queens」「Quick Sort」「Towers of Hanoi」の 5 つのプログラムを、ソースコードを変更した SimMips 上でシミュレートし、CPI を計測した．残りの 3 つのプログラム、「Integer Matrix Multiplication」は、Ultrasmall+ が実行できない乗算命令を含むため、「Puzzle」は使用メモリサイズが Ultrasmall+ のデータメモリサイズ（最大 16KB）を超えてしまったため、「Tree Sort」は stdlib.h の malloc 関数を用いていたため、使用していない．

計測した Cycles per instruction（CPI）を表 4.5 に示す．また、表 4.5 をグラフにしたものを図 4.7 に示す．なお、表 4.5、図 4.7 では、「Bubble Sort」を「Bubble」、「Permutations」を「Perm」、「Eight Queens」を「Queens」、「Quick Sort」を「Quick」、「Towers of Hanoi」を「Towers」としている．また、「average」は、5 つのプログラムの実行に要する総サイクル数を、全実行命令数で割った商としている．

動作周波数（表 4.3）を見ると、Ultrasmall、Ultrasmall α、Ultrasmall β、Ultrasmall+

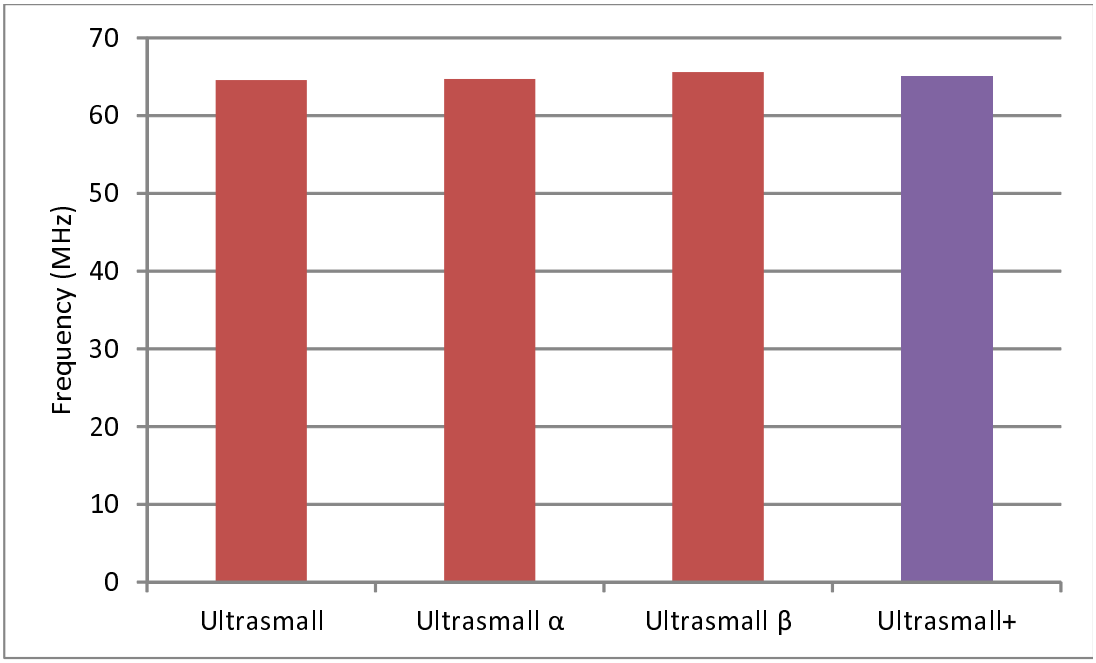

図 4.5 Ultrasmall と各手法を適用したものの動作周波数の比較

と、すべて 65MHz 前後であり、ほとんど変化していない．これに対し、CPI（表 4.5）の値は大きく異なっている．Ultrasmall α が、すべてのアプリケーションで Ultrasmall より大きな CPI となっているのは、条件分岐命令の 16cycle のステートが増加したことによる．Ultrasmall β の CPI は、Ultrasmall α の CPI よりもちょうど 16 だけ小さくなっている．これは、リソース・シェアリングの除去により、NPC の計算のためのステート（16cycle）が必要なくなくなり、このステートが除かれたことによる．Ultrasmall+ は、Ultrasmall β とアーキテクチャが同じであるため、CPI も等しい．

結果、Ultrasmall+ の CPI は従来の Ultrasmall に比べ、average の値でみて CPI が約 14 小さくなっている．MIPS（million instruction per second）値を、動作周波数（表 4.3）と、CPI（表 4.5）の average の値から計算すると、Ultrasmall が 1.76MIPS、Ultrasmall+ が 2.90MIPS となる．これは、提案する Ultrasmall+ が、Ultrasmall の約 1.64 倍高速なソフトプロセッサであることを示す．

4.6 Spartan-6 での評価

提案した手法は、すべて、Spartan-3E を対象としていた．表 4.6 は、従来の Ultrasmall と、これら手法の適用した Ultrasmall α、Ultrasmall β、Ultrasmall+ を Spartan-6 を

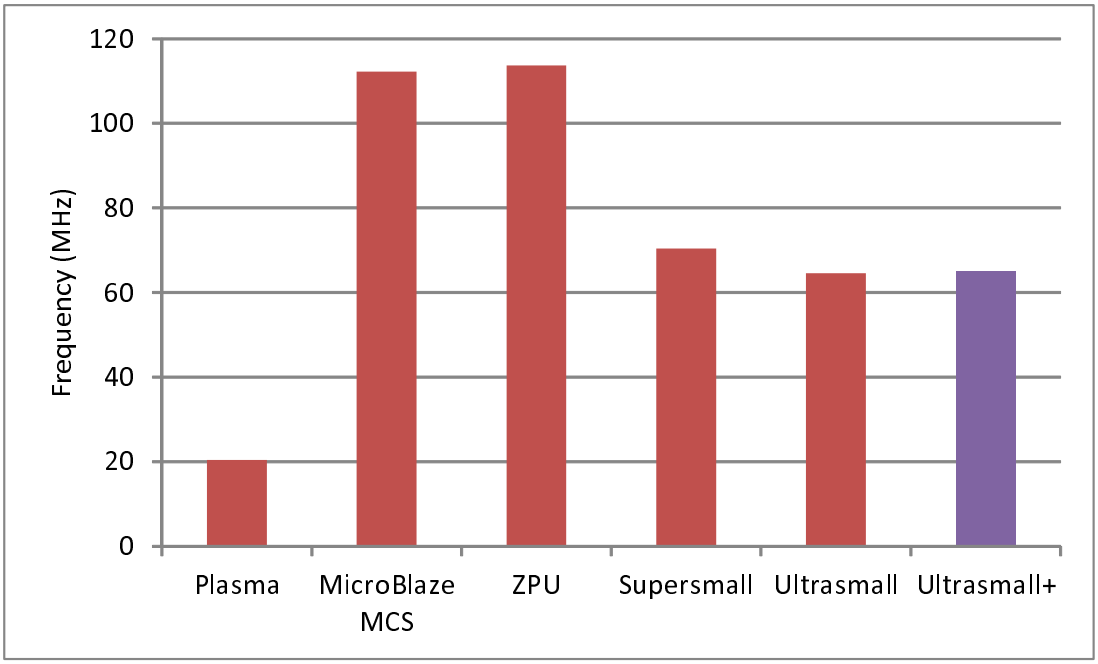

図 4.6　Ultrasmall+ とその他のソフトプロセッサの動作周波数の比較

表 4.5　各手法と Ultrasmall の CPI の比較

	Bubble	Perm	Queens	Quick	Towers	average
Ultrasmall	36.22	36.51	36.23	36.38	37.05	36.51
Ultrasmall α	39.76	37.41	38.59	39.21	38.14	38.38
Ultrasmall β	23.76	21.41	22.59	23.21	22.14	22.38
Ultrasmall+	23.76	21.41	22.59	23.21	22.14	22.38

ターゲットとして論理合成したときのリソース使用量の比較である．なお、ターゲットデバイスには、Spartan-6 ファミリーの XC6SLX16 の CSG324 パッケージを用いた．Spartan-3E では、1 つの Slice に 2 つの 4 入力 LUT と、2 つの FF が配置されていたのに対し、Spartan-6 は、1 つの Slice に 4 つの 6 入力 LUT と、4 つの FF が配置されている [22].

表 4.6 を見ると、Ultrasmall の一部データパスを 32 ビット化した Ultrasmall α は、従来の Ultrasmall に比べ使用 Slice 数が増加した．これは、データパスを 32 ビット化する際、4 入力 LUT 向けの最適化を行ったことによるものと考えられる．また、プリミティブ記述による最適化を行った、Ultrasmall+ も、Ultrasmall β に比べ使用 Slice 数が増加し

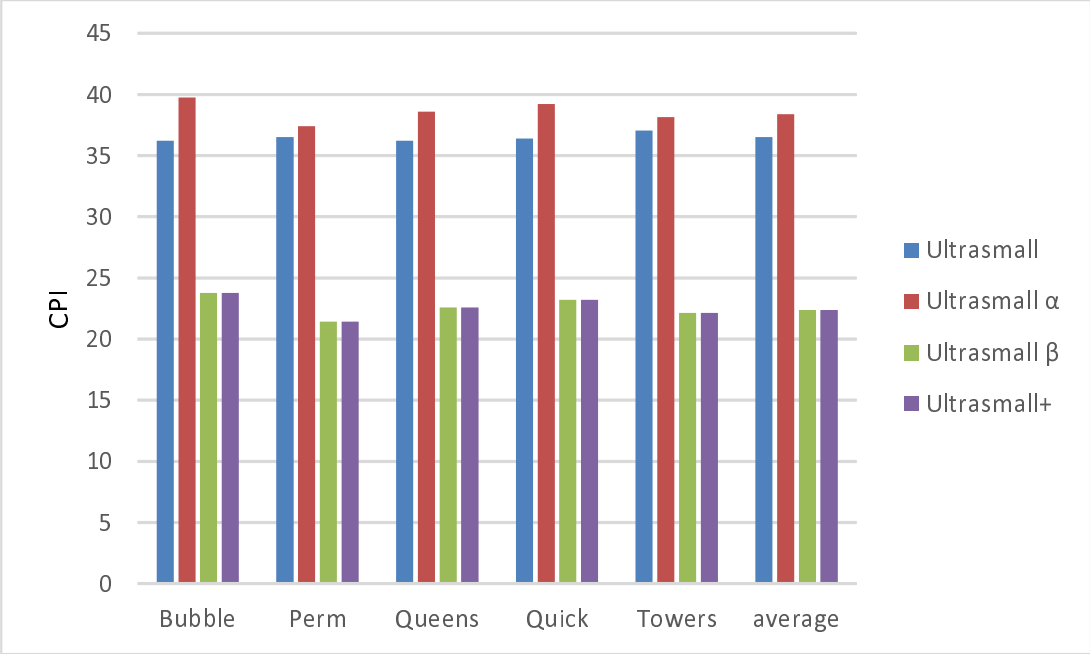

図 4.7 各手法と Ultrasmall の CPI の比較

表 4.6 Spartan-6 上でのリソース使用量の比較

	Slice	LUT	LUT RAM	Slice Reg	BRAM
Ultrasmall	74	185	2	141	9
Ultrasmall α	100	207	2	142	9
Ultrasmall β	99	218	2	142	9
Ultrasmall+	**122**	211	2	142	9

ている．これも、4 入力 LUT 向けの最適化に加え、Spartan-3E の FPGA プリミティブ配置に最適化された記述を行ったことによるものと考えられる．

本論文で提案した手法は、Spartan-3E のアーキテクチャ向けに最適化されたものであるので、Spartan-6 のようなその他のデバイス上では、提案した手法を適用した Ultrasmall+ は、従来の Ultrasmall より大きくなってしまった．これについては、6 入力 LUT 向けのアーキテクチャを新たに考案するなどして、3 章と同様に、一部データパスの 32 ビット化などの面積削減手法を提案できると考えられる．

第 5 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性能向上を実現する手法と最適化を提案した．

Ultrasmall Soft Processor は、全加算器などのロジックの削減のために主要データパスを 2 ビットとする、2 ビットシリアルアーキテクチャを採用していた．これは、32 ビット幅のビットパラレルなプロセッサに対して、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+ は、従来の Ultrasmall に、Xilinx 社の FPGA である Spartan-3E のアーキテクチャに依存した 3 つの最適化を施したものである．Ultrasmall Soft Processor のデータパスの一部を 2 ビットから 32 ビットに変更することによるマルチプレクサの削減、過剰なリソース・シェアリングの除去、一部のモジュールを FPGA プリミティブで記述することにより、Ultrasmall+ は、Ultrasmall に比べ、回路面積を削減しつつも性能を向上させた．

これら 3 つの手法を適用した Ultrasmall+ は、Ultrasmall の回路面積を約 16% 削減つつも、約 1.64 倍の性能向上を実現した．Ultrasmall Soft Processor は、32 ビット RISC 命令を実行する世界最小のソフトプロセッサであった．ゆえに、提案する Ultrasamll+ は、Spartan-3E 上というデバイス依存の条件を持ちながらも、世界最小のソフトプロセッサである．

今後の課題として、FPGA の論理ブロックの構造だけではなく、スイッチ・マトリクスを意識した最適化による Ultrasmall+ のさらなる面積削減手法の提案や、Spartan-6 など、Spartan-3E 以外のデバイス向けの Ultrasmall+ の最適化手法の提案が挙げられる．

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．また、Ultrasmall Soft Processor の開発に多大な貢献をしていただいた田中先輩に感謝いたします．高前田先輩には、研究内容について数多くの有用な意見をいただきました．また、佐藤先輩、小林先輩には、本論文を書くにあたり数多くの助言をいただきました．おかげさまで、なんとか研究が形のあるものとなり、深く感謝しています．吉瀬研究室の皆様には多くの助言をいただき、様々なご迷惑をおかけしました．事ある度に呑みに誘うと、すぐに釣れる池田先輩にももちろん感謝しています．辛くなったとき、負けたとき、いつも美味しいお酒と美味しいオツマミを格安で提供してくれる一軒め酒場雪が谷大塚店とはこれからもお付き合いしていきたいです．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] Xilinx,Inc., PicoBlaze 8-bit Microcontroller,
http://www.xilinx.com/products/intellectual-property/picoblaze.htm

[2] Yuichiro Tanaka, Shimpei Sato, and Kenji Kise, "The Ultrasmall Soft Processor," *Highly Efficient Accelerators and Reconfgurable Technologies (HEART), 2013*

[3] Xilinx,Inc., "Xilinx DS312 Spartan-3E FPGA Family Data Sheet,"
http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[4] Xilinx,Inc., Xilinx UG331 Spartan-3 Generation FPGA User Guide,
http://www.xilinx.com/support/documentation/user_guides/ug331.pdf

[5] 渡辺 優一、山本 泰輔、吉田 雄揮、谷川 一哉、弘中 哲夫、"小規模再構成可能デバイス用ソフトコア・マイクロプロセッサ" リコンフィギャラブルシステム研究会（*RECONF*）, *2013*, pp.39–44

[6] Schoeberl, M., "Leros: A Tiny Microcontroller for FPGAs," *Field Programmable Logic and Applications (FPL), 2011*, pp.10–14

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

[8] Zandrahimi, M., Zarandi, H.R., Rohani, A., "An analysis of fault effects and propagations in ZPU: The world's smallest 32 bit CPU," *Quality Electronic Design (ASQED), 2010*, pp.308–313

[9] Steve, R., Plasma CPU,
http://plasmacpu.no-ip.org:8080/cpu.htm

[10] OpenCores., Plasma - most MIPS I(TM) opcodes,
http://opencores.org/project,plasma

[11] Xilinx,Inc., MicroBlaze Soft Processor Core,
http://www.xilinx.com/tools/microblaze.htm

[12] Xilinx,Inc., PetaLinux Software Development Kit,
http://www.xilinx.com/tools/petalinux-sdk.htm

[13] Xilinx,Inc., MicroBlaze Micro Controller System (MCS),
http://www.xilinx.com/tools/mb_mcs.htm

[14] Xilinx,Inc., Xilinx UG625 Constraints Guide,
http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_5/cgd.pdf

[15] Singh, S., "The RLOC is dead - long live the RLOC," *International Symposium on Field-Programmable Gate Arrays (FPGA '11)*, pp.185-188, 2011

[16] Ehliar, A., "Optimizing Xilinx designs through primitive instantiation," *FPGA-world '10*, pp.20-27, 2010

[17] Xilinx,Inc., ISE WebPack Design Software,
http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html

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

[19] 吉瀬 謙二、"シンプルな計算機システムと組込みシステムの作り方と使い方 〜そこから見えてきた今時のハードとソフトの学び方〜、" 組み込みシステム シンポジウム（*ESS*）、 *2013*、招待公演

[20] Hennessy, J., and Nye, P. "Stanford Integer Benchmarks," *Stanford University*, 1988

[21] Weicker, R.P., "An Overview of Common Benchmarks," *IEEE Computer, Vol.23, No.12*, pp.66-75, 1990

[22] Xilinx,Inc., Xilinx UG384 Spartan-6 FPGA Configurable Logic Block User Guide,
http://www.xilinx.com/support/documentation/user_guides/ug384.pdf