世界最小ソフトプロセッサの設計と応用

田中　雄一郎[†]　笹河良介[†]　吉瀬謙二[†]

Supersmall Soft Processor は，メモリを除く各ユニット間のデータパスを 1 ビットとすることで使用ハードウェア量を最小限に止めたソフトプロセッサである．しかし，データパスの幅を削減したことで動作速度が大きく低下している．本稿では，Supersmall Soft Processor のアーキテクチャを見直し，速度の向上と更なるハードウェア量の削減を図った Ultrasmall Soft Processor を提案する．データパスを 2 ビットとし，レジスタの符号化の最適化と入力ロジックを含む MUX の最適化をおこなう．使用ハードウェア量を Supersmall Soft Processor 以下に抑え，IPC を 88%向上した．

Design of the smallest soft processor and its practical usage

Yuichiroh Tanaka,[†] Ryosuke Sasakawa[†] and Kenji Kise[†]

Supersmall Soft Processor is a soft processor to minimize the amount of hardware with one-bit data path between units except the memory. However, the operating speed is degraded by reducing the width of the data path. In this paper, we propose a Ultrasmall Soft Processor was designed based on Supersmall Soft Processor. Ultrasmall has a 2-bit data path. Some optimizations have been made in the Ultrasmall. We were reduced to less than the amount of hardware Supersmall Soft Processor Using the Ultrasmall. Then, we have greatly improved the IPC.

1. はじめに

Field-Programmable Gate Array(FPGA) とは，購入後に内部構成を書き換えることが出来るロジックデバイスのことである．内部には論理セルが格子状に並んでおり，この論理セルは入力信号に応じて出力する信号を定めた Lookup Table(LUT) と，幾つかのゲートやレジスタなどで構成されている．LUT の定義や配線を変更することにより，ユーザが自身で意図する LSI を実現することが可能となる．カスタマイズ性が高い反面，配線領域などの確保による面積の増大，実装可能な回路規模が小さいことが問題であった．しかし，トランジスタの微細化により近年では FPGA 上でも大規模かつ高性能な回路を実装することが可能となっている．

ソフトプロセッサとは，FPGA に代表されるプログラマブルロジック上で実装することが可能なマイクロプロセッサコアである．Xilinx 社の MicroBlaze[1] や Altera 社の Nios II[2] に代表されるソフトプロセッサは，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いる．

マイクロプロセッサコアを作成する手段の一つとして ASIC(Application Specific Intergrated Circuit) や ASSP(Application Specific Standard Product) が挙げられる．これらを用いた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 や ASSP では，個々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 個から安価に実現することが可能である．

FPGA 上に多数の機能を搭載した後，残り少ない限られた資源で新たな機能を追加したい場合が想定される．特に各機能を統括する制御用プロセッサの実行時間が，実行時間全体に対して非常に小さくなる場合が考えられる．このような場合，制御用プロセッサは動作速度よりもハードウェア資源の使用量に重点が置かれる．

使用ハードウェア量を抑えたソフトプロセッサの研究として，Supersmall Soft Processor[3] が存在する．この既存研究は乗除算と非アライメントロード・ストアを除いているものの，Altera 社の Stratix III[4] を対象として極めてコンパクトに実装されている．32bit のデータを扱い，MIPS-I 命令を実装しているソフトプロセッサの中では我々が知る限りで，最も小さなソフトプロセッサである．

本稿では Supersmall Soft Processor を元に更なる使用ハードウェア量の削減，並びに性能向上を目指す Ultrasmall Soft Processor を提案する．省スペースな

† 東京工業大学
Tokyo Institute of Technology

ソフトプロセッサが求められる環境において，ハードウェア量の削減並びに性能向上を達成することによってより多くのニーズに対応することが可能となる．

Ultrasmall Soft Processor は Supersmall Soft Processor 同様に，乗除算，浮動小数点演算などの一部の命令を除いた MIPS-I 命令セットを実装するプロセッサである．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．ターゲットは Xilinx 社の Spartan-3E，Spartan-6，Virtex-7 としている．したがって，開発するに当たり Supersmall Soft Processor を Xilinx 社の FPGA に実装できるように移植し，その上でハードウェアの削減並びに性能向上を目指す．

以降，本稿では 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．

本稿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おこなう．4 章にて各手法，最適化を実装し評価をおこなう．5 章では Ultrasmall の応用例としてマルチコア化を挙げ，その利用について述べる．最後に 6 章で本論文をまとめる．

2. 関 連 研 究

本章では関連研究として Supersmall Soft Processor[3] について述べる．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．Supersmall は命令を処理する間に幾度か状態遷移をおこない，基本的に各状態の実行サイクルは 33 サイクルである．Altera 社製のハイエンド FPGA である Stratix III をターゲットとしており，使用ハードウェア量は同社のソフトプロセッサである Nios II/e の約 2.2 分の 1 である．

図 1 に Supersmall のブロック図を示す．灰色のユニットは実装する対象のデバイス依存であることを示し，この場合 Supersmall のメモリシステムに依存していることを意味する．以降のブロック図においても，デバイス依存のユニットは灰色で示すものとする．

Supersmall は 2^{11} 個の命令を格納出来る Instruction Memory を持ち，Program Counter の下位 11 ビットが Instruction Memory のアドレスとなっている．リードイネーブルが 1 になると Instruction Memory から命令を読み出す．Register File 内に保存されたデータを使用する場合，Register File の各 Port からサイクルに応じて 1 ビットずつ Shift Register へと送られる．

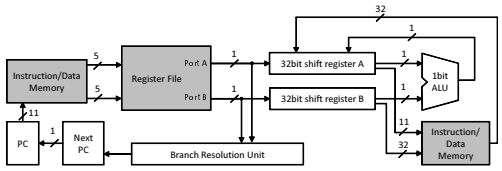

図 1 Supersmall のブロック図

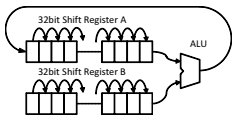

図 2 Supersmall で ALU を使用

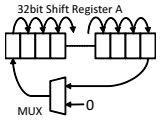

図 3 Supersmall でシフトを実行

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 Shift Register 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 Shift Register A へと読み出される．このようにメモリを除くユニット間のデータパスのビット幅を 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，データパスのビット幅が 1 ビットとなることで MUX が小さくなり，配線領域の削減が可能となる．

ALU 使用時の Shift Register 周りのデータフローを**図 2** に示す．Shift Register A は動作時にデータを 1 ビットずつ LSB 側へとずらし，ALU から送られてくる LSB の演算結果を MSB へと格納する．これを 32 サイクル繰り返すことで 32 ビットの演算をおこなう．

ロジカルなシフト命令処理時のデータフローを**図 3** に示す．右へシフトする場合は MUX からの入力を 0 とし，シフトしたいビット数だけ Shift Register を動作させればよい．また，左へシフトをおこなう場合は 32 サイクル常に Shift Register を動作させる．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．Shift Register 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 は 1 サイクルに 1 ビットしか処理できないため，32 ビットに対して処理をおこなうには最低 32 サイクルを要する．処理の実行回数を記憶する

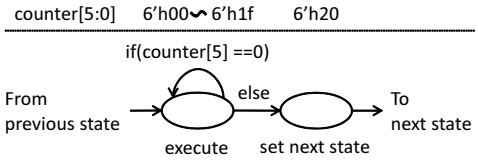

図 4　Supersmall で状態遷移をおこなう

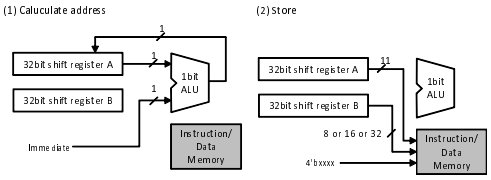

図 5　Supersmall でストア命令を実行

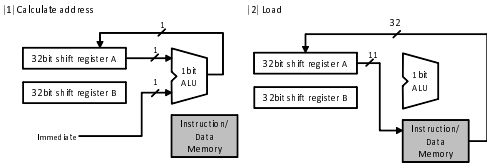

図 6　Supersmall でロード命令を実行

カウンタは初期値が 0 である．そのため，32 サイクル目を検知して次の状態へと遷移するためには，カウンタの下位 5 ビットに対して論理積を取る必要がある．これではロジックが複雑となってしまうため，**図 4** に示すように Supersmall ではカウンタの 6 ビット目を用いて状態遷移の分岐を判断している．罫線上部に処理の実行回数を記憶するカウンタの値を 16 進法で示す．状態遷移図はカウンタの値に対応しており，32 サイクル目までは演算をおこなう．33 サイクル目は次の状態を設定するのみであり，演算をおこなわない．

これら 2 つの理由より，Supersmall は Nios II/e より処理速度が約 10 倍遅い．

バイト，ハーフワード単位の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 5** となる．命令を元に Register File から参照されたデータが，Shift Register A，B それぞれに送られる．次に Shift Register A に即値を加算し，再度 Shift Register A に格納する．これが図 5(1) の書き込み先のアドレス計算である．(2) に示す 4 ビットの制御信号を元に，メモリシステムは Shift Register A で指定されるアドレスのどの部分を Shift Register B で上書きするかを判断する．

バイト，ハーフワード単位のロード命令を実行する場合のデータフローは**図 6** となる．(1) でストア命令と同じようにアドレスを計算し，(2) で Shift Register A へ Data Memory からデータを読み出す．

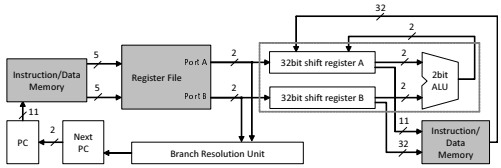

図 7　Ultrasmall のブロック図

3. Ultrasmall Soft Processor の提案

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），フリップフロップ（FF）などから構成される．本稿では，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．

次に列挙する 1 つのアーキテクチャ的な改良手法と 2 つの最適化手法を Ultrasmall のために提案する．

- 主要なデータパスの 2 ビット化
- レジスタの符号化の最適化
- MUX の入力ロジックを含む最適化

これらの手法と最適化を用いる Ultrasmall の構成を**図 7** に示す．Supersmall ではメモリの入出力を除くデータパスのビット幅が 1 ビットであったが，Ultrasmall では 2 ビットとする．

3.1 主要なデータパスの 2 ビット化

メモリを除くユニット間のデータパスを 1 ビット幅としたことで，Supersmall はハードウェア使用量を抑えている．1 ビット幅とすることで MUX が小さくなると共に，配線領域の削減が可能となる．

Supersmall は比較対象として Altra 社が提供するソフトプロセッサ，NiosII を挙げて評価をおこなっており，この NiosII はデータパスが 32 ビット幅であるソフトプロセッサである．しかし，Supersmall の論文ではデータパスが 32 ビット幅ではないソフトプロセッサとの比較はおこなわれておらず，データパスが 1 ビット幅が最適である所以も記されていない．

仮に，使用ハードウェア量を増加させることなくビット幅を大きくすることが可能であれば，動作速度は向上する．データパスのビット幅を 1 ビット以外とした時にハードウェア量がどのように変化するのか，予備評価をおこなう．

Supersmall の論文では，データパスを 1 ビット幅にして ALU のハードウェア量を削減したと述べている．そのため，データパスのビット幅を変更するにあたり，ユニットの中でも ALU は影響を大きく受けると考えられる．図 7 の破線で囲む部分，ALU，32bit

表 1 ALU の入出力ビット幅を変更した際のハードウェア使用量の変化

bit 幅	1bit	2bit	4bit	8bit	16bit
Reg	4	4	5	5	5
LUTs	5	8	15	27	51
Slice	5	3	7	11	18

表 2 シフトレジスタを含めた ALU 部のハードウェア使用量の変化

bit 幅	1bit	2bit	4bit	8bit	16bit
Reg	106	104	102	97	88
LUTs	59	61	65	74	94
Slice	22	22	21	23	30

shift reister A，B をまとめて ALU 部と呼ぶことにする．データパスのビット幅を変更することで ALU と ALU 部がどのように変化するかを予備評価として調べる．

ALU の入出力を 2 の乗数となるビット幅とした際，**表 1** に示すようにハードウェア使用量が変化する．変更した ALU に適応するようにシフトレジスタを変更し，論理合成した際の使用量を**表 2** に示す．論理合成には Xilinx 社の ISE(version 14.2) を用いる．ターゲットは Spartan-6 XC6SLX16 とする．

ALU のみで比較をおこなった場合，ビット幅を 2 ビットとした場合が最も小さくなる．ALU 部で比較をおこなった場合，16 ビット幅とした場合を除き Slice 数に大きな変化はない．動作速度を大きく向上することを目的として，データパスのビット幅を大きくすることを提案する．

Supersmall のデータパスを大きくすると、一部の命令で正しく動作しなくなる．2 ビット幅とした場合の解決策を説明する．図 3 に示した様に，Supersmall ではデータの保存先である Shifter Register の特徴を活かしてシフト命令を実行する．そのため，1 サイクルあたりのシフト量を 1 ビットから 2 ビットへ変更するだけでは，奇数ビット分のシフトをおこなうことはできない．

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．2 ビットのシフトと Shift Register に 1 ビットのレジスタを加えて 33 ビット幅とした状態でおこなう，2 ビットのローテートである．

$2n+1$ ビットシフトする場合，まず 2 ビットのシフトを n 回おこなう．これは**図 8** の（1）～（2）である．その後，右シフトなら 17 回，左シフトなら 16 回 2 ビットのローテートをおこなう．これは図 8 では（2）～（5）であり，この操作が 1 ビット分のシフトである．図では全体が 4 ビットなので，3 回 2 ビットのローテートをおこなうことで 1 ビット分のシフトとなっている．

この操作のため，偶数ビットのシフトと比べて奇数ビットのシフトの実行サイクルは 16 ないし 17 サイクル大きくなる．しかし，レジスタファイルからの読み出しに必要な実行サイクルが半分となっているため，シフト命令自体の実行サイクルはデータパスが 1 ビット幅の時よりも小さい．これは任意のビット数のシフトにおいても成り立つ．

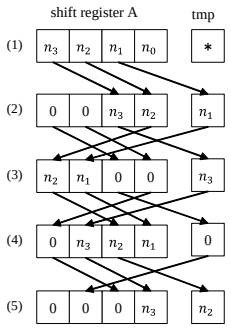

図 8 長さ 4 ビットの Shift Register で 3 ビット右へ論理シフト

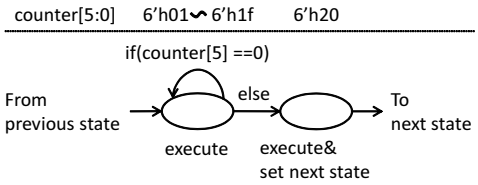

図 9 カウンタの初期値を 1 とした場合の状態遷移

2 ビットより大きいビット幅で実装する場合は，1 ビット分のシフトを繰り返すことにより対応することができる．しかし，実行回数を記憶するカウンタが必要となるためにスライス数が増加する．そのため，データパスのビット幅を 2 ビットとして，ハードウェア使用量を抑えつつ速度の向上を図る．

3.2 レジスタの符号化の最適化

Supersmall では，カウンタの 6 ビット目を用いて状態遷移をおこなっている．カウンタの初期値は 0 であるため，33 サイクル目では Shift Register をストールさせ，演算をおこなわずに状態遷移のみをおこなっている．状態遷移に余計に 1 サイクルを要するため，このカウンタの初期値は非効率的である．

そこで，**図 9** に示すようにカウンタの初期値を 1 とする．32 サイクル目で演算をおこないつつ，並行して次の状態遷移先を設定する．Ultrasmall ではデータパスが 2 ビットであるため，この手法により実行サイクルが 1 状態あたり 17 サイクルから 16 サイクルへと減る．さらに，状態遷移のために Shift Register をストールさせる必要がなくなったため，それらの制御線がなくなる．

3.3 MUX の入力ロジックを含む最適化

Ultrasmall の Shift Register A への入力は 12 本あり，常にその中から MUX で 1 本選択される．MUX

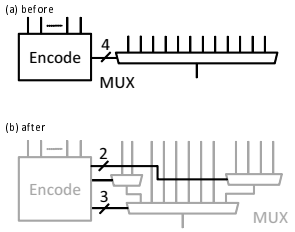

図 10 MUX を 2 段に変更

への入力本数が 2 の乗数ではないために，資源を効率よく利用できていない．また，MUX への入力に使用頻度が極端に少ないものが存在する．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 Shift Register A への入力を MUX でまとめて 2 段構成とする．まとめた MUX への制御信号はほとんどの状況下においてドントケアでよく，全体としての制御信号を減らすことを図る．

図 10 は MUX に施す最適化の前後の様子を示す．変更前では，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，まとめられた入力のいずれかを用いる場合には最大 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が削減されると考えられる．

4. 評　　価

4.1 使用ハードウェア量と考察

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．我々は 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おこなう．また，簡単のために例外処理部も除く．ツールは Xilinx 社製 ISE（ver14.2）を使用し，Optimization Goal を Area に，Optimization Effort を High と設定して評価をおこなう．

表 3 に各手法と最適化を実装した場合と Supersmall との比較を示す．データパスの 2 ビット化，符号化の最適化，2 ビット化と符号化の最適化を併せたものがそれぞれ，2bit，Encode，Fusion である．また，Fusion に MUX の最適化をおこなったものが Ultrasmall となる．

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 の量を Supersmall と比較すると，Reg は常に Supersmall を下回っている．一方，LUTs を見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．このことから，2bit の構成では LUTs が最小化に影響を及ぼしたと考えられる．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．ISE の自動配置配線への依存が大きいため詳しいことは不明である．

表 3 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
2bit	211	144	302	4	10
Encode	205	162	296	5	10
Fusion	207	141	291	4	10
Ultrasmall	**205**	**141**	**289**	**4**	**10**

Spartan-6

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

Virtex-7

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

Encode では Shift Register の制御線を減らすと共に，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 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．これは制御線が少なくなったことで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 Fusion である．2bit と同じように，Virtex-7 を除く 2 つのデバイスでは Supersmall より Slice が大きくなる．しかし，Spartan-3E 並びに Spartan-6 において Reg と LUTs は Supersmall のそれを下回っている．これは，Fusion が配置配線に適した構成をしていないために Supersmall より大きくなったと考えられる．

Ultrasmall は Fusion 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 Fusion 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，Fusion と比較して配置配線に適した構成であることが分かる．

全てのデバイスで 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，我々が知る限り

表 4　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**

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．

4.2　CPI から見た性能向上

表 4 に 4.1 節で考察をおこなった各手法，最適化を施した場合の CPI をそれぞれ示す．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が等しい確率で用いられるものとして CPI を計算する．

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 サイクルよりも小さい．これは，動作周波数が等しいとき，Ultrasmall の処理速度が Supersmall の 1.88 倍であることを意味する．したがって，Ultrasmall は Supersmall と比べて性能を大きく改善するものである．

5.　マルチコア化

Ultrasmall の応用例としてマルチコア化を考える．Supersmall が 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たため，4 章の Ultrasmall では該当命令の処理部を除外している．この処理部を実装するために，ハードウェアを一部変更してこれらの命令に対応する．また，ソフトプロセッサ同士を繋ぐだけでは、マルチ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 に施した以下のハードウェア変更，並びに追加について述べる．

- 除外した命令の実装
- ネットワーク構成と追加ハードウェア

5.1　除外した命令の実装

Supersmall と比較するために 4 章ではバイト，ハーフワード単位のロード･ストア命令の処理部分を除外した．しかし，実用化にあたってこれらの命令は無視できるものではない．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をおこなっているため，Ultrasmall ではそのハードウェア構成をそのまま採用することができない．そのため，**図 11** 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．Instruction/Data Memory からの読み出し先を Shift Register A から B へと変更している．

Ultrasmall のメモリシステムはワード単位で書き込みをおこなうため，バイト，ハーフワード単位で書き込みをおこなう際には，書き込まない部分のデータも

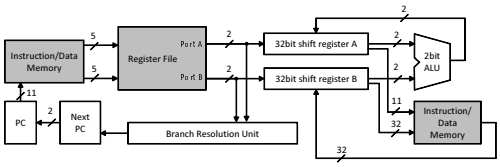

図 11　バイト，ハーフワード単位のロード・ストア命令に対応化した Ultrasmall のブロック図

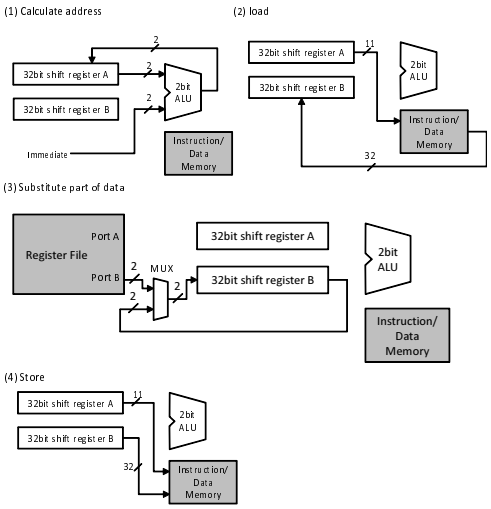

図 12　Ultrasmall でストア命令を実行

用意する必要がある．そのため，バイト，ハーフワード単位で書き込みをおこなう場合は，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 5 で示した通り，Supersmall の構成では読み出しをおこなった際に Shift Register 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 Shift Register 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 12 にバイト，ハーフワード単位で書き込みをおこなう際の流れを示す．(1) で Shift Register A の値に即値を足し合わせて書き込み先のアドレスを算出し，(2) で Shift Register B にデータを読み出す．図 12 の (3) に示すように，Shift Register B への入力は Shift Register B か Register File か選択できる．書き込むデータは常に Register File PortB より出力されているので，Shift Register B のデータを上書きする場合は Register File B を選択する．上書きをしない箇所では，Shift Register B を選択して Shift Register B 内でのローテートとする．必要に応じて Shift Register B への入力を切り替えながら上記の処理を 32 サイクルおこなうことで，データを部分的に変更することが可能になる．(4) で Shift Register A に保持していたアドレスにワード単位で書き込みをおこなう．

ロード命令実行時のデータフローを**図 13** に示す．アドレスの算出については図 12 と同じため，省略す

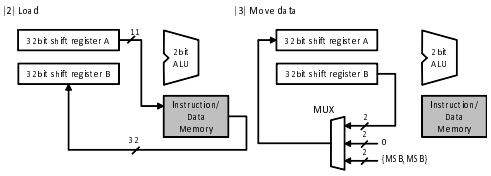

図 13　Ultrasmall でロード命令を実行

表 5　MicroBlaze

	Reg	LUTs	BRAM
MicroBlaze(Minimum)	546	276	1

表 6　アーキテクチャを変更した Ultrasmall

	Slices	Reg	LUTs	LUTRAM	BRAM
Ultrasmall	171	147	278	2	6

る．(2) ではストア命令実行時と同様にデータを Shift Register B へ読み出す．(3) に示すように Shift Regiter A にデータを移しながら命令に応じて符号拡張を行う．最後にアライメントをおこない，次の状態へと遷移する．

Xilnx 社が提供するソフトプロセッサに MicroBlaze が存在する．これは Ultrasmall と同じ 32 ビットの命令を処理する RISC プロセッサであり，処理性能は Ultrasmall を超える．MicroBlaze のハードウェア構成を**表 5** に，アーキテクチャ変更後の Ultrasmall を論理合成した結果を**表 6** に示す．

対応命令を追加するにあたって状態数が増えたために，Ultrasmall が使用する Slice の数が増加した．対応命令数が異なるため性能に基づいた比較をおこなうことはできないが，MicroBlaze に対して Ultrasmall が非常に小さいことが Reg の比較からよく分かる．BRAM に関しては，MicroBlaze および Ultrasmall 共に命令，データ用のメモリとして利用しており，設定しているメモリの量は Ultrasmall の方が大きい．よって，表中での BRAM は Ultrasmall の方が多くなっている．

バイト，ハーフワード単位のロード・ストア命令の対応化をおこなった Ultrasmall の各状態と必要サイクルを**図 14** に示す．Commem to all instructions の 3 状態を経た後，命令ごとに対応した状態へと遷移する．図 14 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 で示した CPI は 41.3 サイクルに増加する．しかし，Supersmall も該当命令を含めると 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．

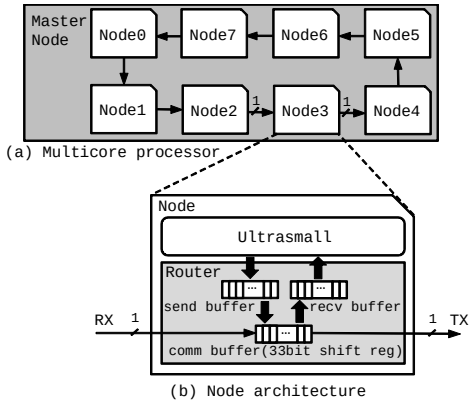

図 15　マルチコア化に向けたネットワーク構成

5.2　ネットワーク構成と追加ハードウェア

Ultrasmall は占有面積が最小となるように設計をおこな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 15** に示す．ルータと Ultrasmall をセットで 1 ノードとして図 15 中の (a) に示すように，これをリング状に配置する．ノードは 1 サイクルに 1 ビット出力をおこない，1 サイクルに 1 ビット入力を受け付ける．

既定のサイクル（comm buffer 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ので，受信する場合は recv buffer に comm buffer からコピーして comm buffer の中を空にする．送信をおこなう場合は，既定サイクル時に comm buffer が空であることを確認して，send buffer のデータを comm 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受信の判断をおこなう．

電源が投入されるとまず初期化をおこなう．始めに Master ノードのみがプログラムを持っており，Slave ノードにプログラムを配布する必要がある．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，プログラムを Master ノードがネットワーク上を 1 周させる．この際，Slave ノードは ID の比較をおこなわずに recv buffer へデータをコピーし，Instruction Memory に書き込んでいく．またこの時，comm buffer を空にせずそのまま次のノードへ送る．プログラムを全て配布し初期化を終えると，ルータモードへと切り替わる．

以上のように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マルチ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

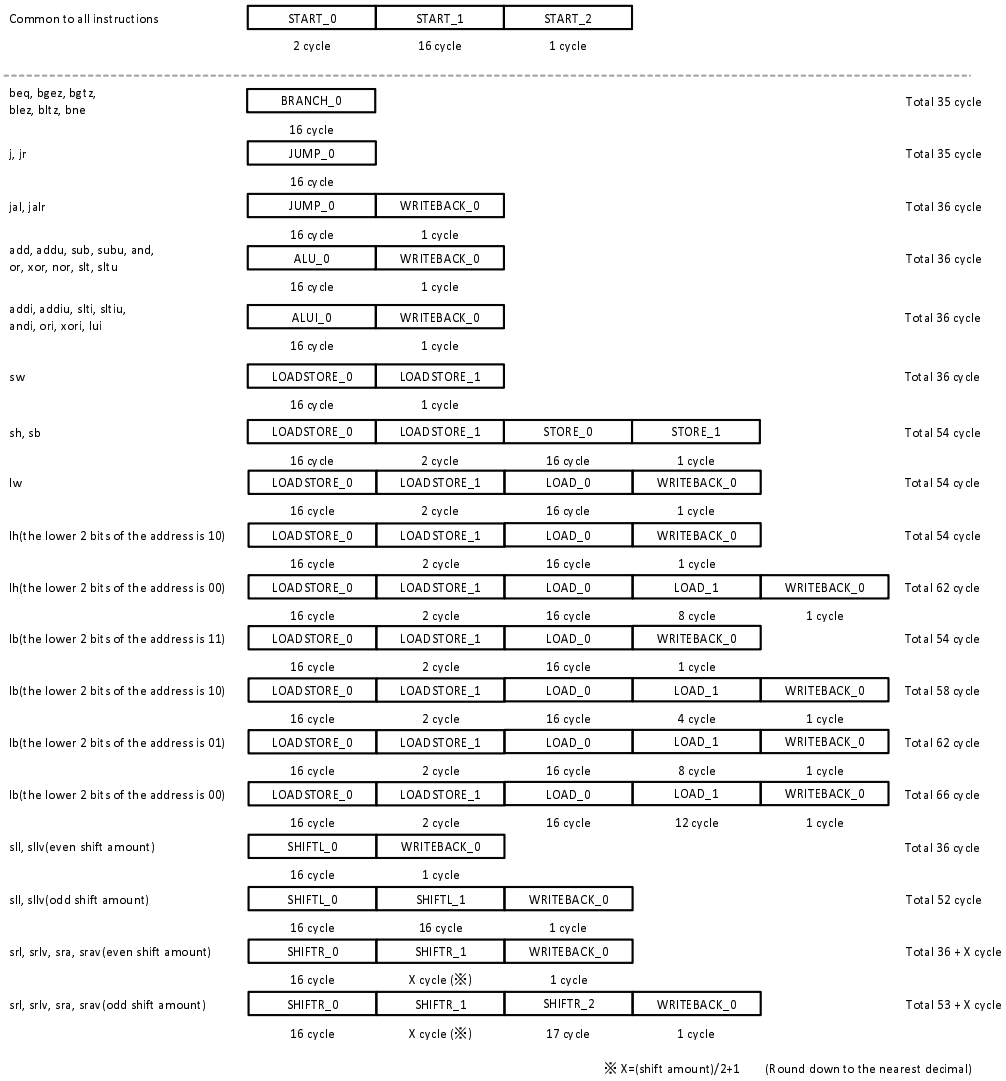

図 14 各命令の状態遷移と必要サイクル数

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 7** 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おこなう．

表 7 の 1 コアと表 6 を比較すると，ノードの約 3 分の 1 をルータが占めていることが分かる．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．表 7 より，ノード数の増加に対してほぼ比例するハードウェア量の増加に抑えられていることが分かる．

表 7 各コア数におけるハードウェア量

Core	Slices	Reg	LUTs	LUTRAM	BRAM
1	255	298	488	2	6
2	570	587	943	4	12
4	1,092	1,213	1,909	8	24
8	2,051	2,417	3,809	16	48
16	4,139	4,813	7,597	32	96
32	8,095	9,617	15,140	64	192
64	16,210	19,273	30,507	128	384
128	34,708	38,537	63,234	256	768

6. おわりに

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．これらの手法と最適化を施した 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．また，既存研究である Supersmall Soft Processor の 1.8 倍以上の処

理性能を持つ．

各手法並びに最適化の実装結果を比較すると，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 ISE の自動配置配線による差であると考えられ，この自動配置配線に対してはより深い考察が必要であると考えられる．

また，Ultrasmall Soft Processor の応用例としてマルチコア化を提案した．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，正常動作を確認した．

今後の課題として，実配線を元にした配置配線の最適化によるハードウェア量の更なる削減が挙げられる．また，応用例としては，コア数を増やしてメニーコアによる台数効果の検証が挙げられる．

本稿を執筆するにあたり，多くのご支援賜りました豊橋技術科学大学市川研究室の藤枝先生，吉瀬研究室の佐藤さんに心より深くお礼申し上げます．

参 考 文 献

1) Xilinx: LogiCORE IP MicroBlaze Micro Controller System (v1.1) (2012).
2) Altera: Nios II Processor Reference Handbook (ver 11.0.0) (2011).
3) Robinson, J., Vafaee, S., Scobbie, J., Ritche, M. and Rose, J.: The supersmall soft processor, *Programmable Logic Conference (SPL), 2010 VI Southern*, pp. 3 –8 (2010).
4) Altera: Stratix III Device Handbook (2011).