Vol.2013-SLDM-160 No.26
Vol.2013-EMB-28 No.26
2013/3/14

世界最小ソフトプロセッサの設計と応用

田中 雄一郎[1]　笹河 良介[2]　佐藤 真平[2]　吉瀬 謙二[2]

概要：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，制御やデータ処理など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，小さいソフトプロセッサの開発が重要である．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．本研究では Supersmall Soft Processor を元に使用面積の削減，並びに性能向上を目指す Ultrasmall Soft Processor を提案する．Supersmall Soft Processor に対し，主要データパスの 2 ビット化，状態遷移の最適化，マルチプレクサの入力ロジックを含む最適化を施す．その結果，Virtex-7 においてハードウェア量の削減を達成し，1.88 倍の IPC を実現した．また，Ultrasmall Soft Processor の応用としてメニーコア化を検討する．

1. はじめに

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．コプロセッサや SoC のソリューションとして，FPGA が導入されつつあり，各機能の制御のために，FPGA 上にソフトプロセッサが組み込まれる機会も増えると考えられる．しかし，実際にソフトプロセッサを組み込む場合に，先に挙げた小型デバイスなど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，ハードウェア量の小さいソフトプロセッサの開発が重要である．

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 を用いたものの中では調べた限りで最小のソフトプロセッサである．

本稿では Supersmall Soft Processor を元に更なる使用面積の削減，並びに性能向上を図る．主要データパスの 2 ビット化，状態遷移の最適化，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 は，性能向上と共に，Supersmall Soft Processor から更なる使用面積を削減することで，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ものを目指す．

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．

2. 関連研究

Supersmall Soft Processor[1] 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．しかし，Ultrasmall は Xilinx 社製の FPGA をターゲットとしており，両社の FPGA には相違点が存在する．特に，Supersmall は Stratix シリーズに依存するメモリシステムを有している．本章では Stratix III のメモリシステムについて述べ，続いて Supersmall について説明する．

2.1 Stratix III

Stratix とは Altera 社製ハイエンド FPGA のシリーズ名である．Xilinx 社製 FPGA との相違点の一つが，Stratix シリーズ当初から内蔵されている 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 [2], [3], [4]．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，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．更に，バイト，ハーフワード単位でのロード・ストア命令のために特殊なカウンタを追加しており，これらのハードウェ

[1] 東京工業大学 工学部情報工学科
Department of Computer Science, Tokyo Institute of Technology
[2] 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

Vol.2013-SLDM-160 No.26
Vol.2013-EMB-28 No.26
2013/3/14

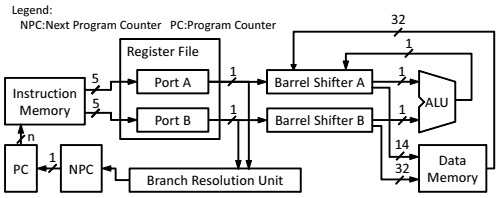

図 1 Supersmall のブロック図

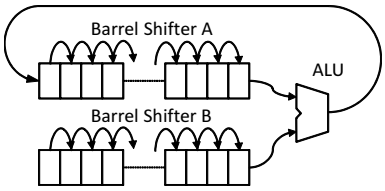

図 2 Supersmall で ALU 使用時のデータフロー

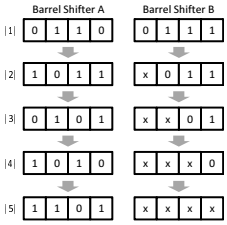

図 3 ALU 使用時の Barrel Shifter

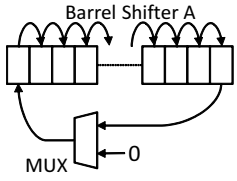

図 4 Supersmall でシフト実行時のデータフロー

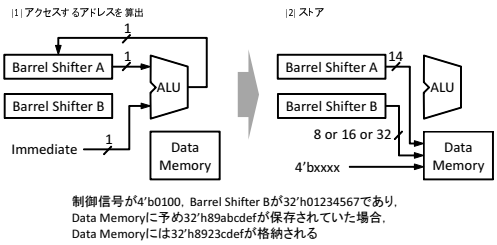

図 5 Supersmall でストア命令を実行

アはメモリシステムに依存したものである．

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 [5]．また，Spartan-6 並びに Virtex-7 は同様の機能を持つ [6], [7] が，その機能の利用方法が Altera 社製のものと異なるため，そのまま Xillinx 社製 FPGA 上に実装することはできない．

2.2 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 1 に Supersmall のブロック図を示す．命令に応じた値が Register File から読み出され，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．このようにメモリを除くユニット間のデータパスの幅を 1 ビットとすることで，ユニットの構成を簡単化している．しかし，データパスのビット幅を 1 ビットにしたことで，32 ビットに対して行う 1 つの処理に最低 32 サイクル必要とする点が問題点として挙げられる．

ALU 使用時の Barrel Shifter 周りのデータフローを**図 2** 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．この時の Barrel Shifter のデータの変化を**図 3** 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．(1) の Barrel Shifter A，B の LSB である 0 と 1 が加算され，Barrel Shifter A の MSB に 1 が格納された状態が (2) である．(2) ～ (4) でも同様の処理を行い，(5) では加算結果の 13 が Barrel Shifter A に格納されている．

ロジカルなシフト命令処理時のデータフローを**図 4** 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．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．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．

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** に示す通りである．Barrel Shifter A，B それぞれには命令に応じたデータが Register File から読み出される．次に，Barrel Shifter A に即値を加算してアドレスを算出する．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データのどこに上書きするかを判断する．

Supersmall は 1 サイクルに 1 ビットしか処理できないため，32 ビットの処理を行うには最低 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，32 サイクル目で処理を終了して 33 サイクル目で状態遷移を行う．この 33 サイクル目の処理は状態遷移のみであるため，最適化の余地がある．

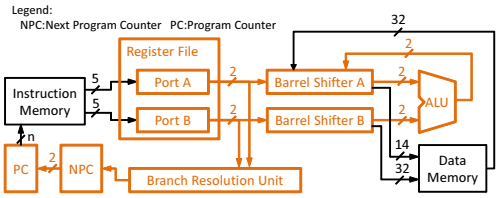

図 6　データパスを 2 ビット化

表 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

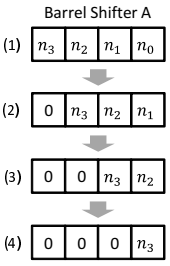

図 7　Supersmall で 3 ビットシフトする

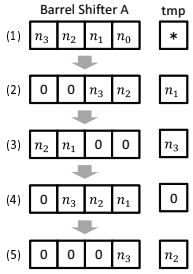

図 8　2 ビット化した後の Supersmall で 3 ビットシフトする

3. 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 は Xilinx 社の FPGA をターゲットとする．本稿では，安価で広く用いられている Spartan-3E シリーズ，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 シリーズの 3 つをターゲットとし，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 データパスの 2 ビット化

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 6** において，橙色となっている部分が図 1 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 のビット幅を 1 ビットから 16 ビットに変化させた時のハードウェア量を**表 1** にまとめる．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分かる．これより，Ultrasmall ではデータパスを 2 ビットとして処理性能を高めることにする．

Supersmall ではデータの保存先である Barrel Shifter の特徴を活かして**図 7** のようにシフトを行う．よって，Supersmall のデータパスを 2 ビット化する場合，そのままでは奇数ビット幅のシフトを行えない．例では，簡単化のために Barrel Shifter を 4 ビット幅として図示する．

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 ビットシフトと 1 ビットずらしの 2 ビットシフトである．**図 8** にサイクル毎の Barrel Shifter の値の変化を示す．$2n+1$ ビットシフトする場合，まず n 回 2 ビットシフトを行う．これは図 8 の（1）～（2）にあたり，その時のデータフローを**図 9** に示す．その後，右シフトなら 17 回，左シフトなら 16 回 1 ビットずらしの 2 ビットシフトを行う．データフローを**図 10** に示す．図 8 では（2）～（5）が 1 ビット分のシフトにあたり，図では全体が 4 ビットなので，3 回 1 ビットずらしの 2 ビットシフトを行っている．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．

奇数ビットのシフトを行う場合，1 ビットのシフトに 16 もしくは 17 サイクル必要とする．そのため，奇数ビットのシフトを行う場合はデータパスが 1 ビットの時以上にサイクルを必要とするデメリットがある．

3.2 状態遷移の最適化

2 章で述べたように，Supersmall では 32 サイクルかけて処理を行い，33 サイクル目で状態遷移のみを行う．この状態遷移を，処理の終わる 32 サイクル目に同時に行うこと

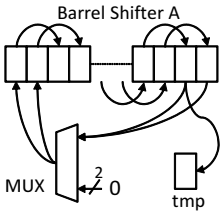

図 9　2 ビットシフト

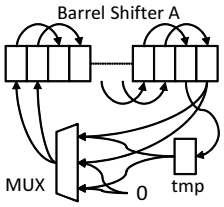

図 10　1 ビットずらしの 2 ビットシフト

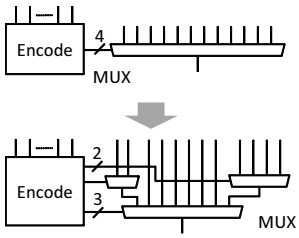

図 11　MUX による調整

で，サイクル数を 1 つ減らすことが可能である．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 32 ビットの処理に要するサイクルが 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，MUX で 12 本の中から 1 本が選択される．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 への入力には滅多に選択されないものも含まれる．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 11 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．MUX を分散させることで LUT と配置配線の最適化が進むと考えられる．

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，Xilinx のデバイスをターゲットとしている Ultrasmall では，バイトイネーブルサポート機能を有する TriMatrix メモリを利用できないので，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 12** 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．変更点は橙色で示す．

Ultrasmall のメモリシステムはワード単位で書き込みを

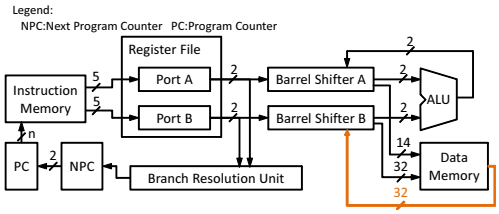

図 12　変更後の Ultrasmall のブロック図

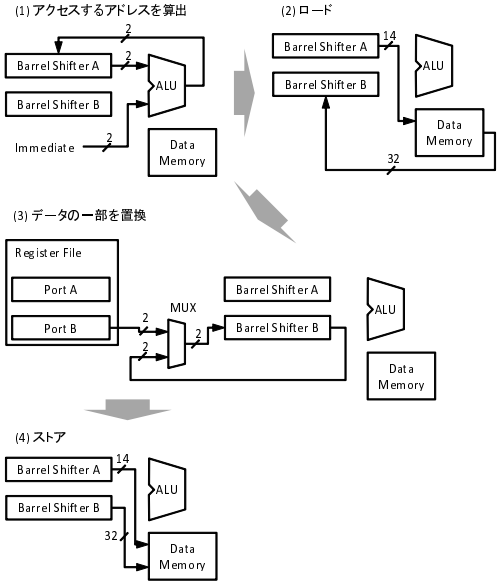

図 13　Ultrasmall でストア命令を実行

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，Supersmall の構成では Barrel Shifter A に読み出したデータを格納する．これでは書き込み先のアドレスが破壊されるため，再度アドレスを算出する必要が出てくる．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 13 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，命令に対応したデータは Register File PortB より常に出力されている．ゆえに，上書きを行いたい箇所で，MUX の入力を Barrel Shifter B から Register File PortB に切り替えればよい．

4.　評価

4.1　ハードウェア量

Supersmall はバイト，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

表 2 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
Proposal（2bit）	211	144	302	4	10
Proposal（State）	205	162	296	5	10
Proposal（2bit & State）	207	141	291	4	10
Ultrasmall	**205**	**141**	**289**	**4**	**10**

Spartan-6

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

Virtex-7

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

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 2 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．

- Proposal（2bit）……データパスの 2 ビット化
- Proposal（State）……状態遷移の最適化
- Proposal（2bit & State）……2 ビット化し状態遷移を最適化したもの
- Ultrasmall……Proposal（2bit & State）に MUX の変更を行ったもの

論理合成した結果，Spartan-3E 上では Supersmall と Ultrasmall は共に使用 Slice 数が 205 となる．Spartan-6 上では Supersmall は 140，Ultrasmall は 139 となる．Virtex-7 上では Supersmall は 163，Ultrasmall は 157 となる．

4.2 性能向上

表 3 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．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，その算術平均をとる．状態遷移の最適化により 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%削減される．

表 3 各ソフトプロセッサの CPI

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

4.3 考察

表 2 において，Proposal（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 Proposal（2bit）が配置配線が困難な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．

Proposal（State）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る．どのデバイスにおいても Proposal（State）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 State を組み合わせて用いたものが Proposal（2bit & State）である．Proposal（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，Proposal（2bit & State）も配置配線に適していなかったと考えられる．

Ultrasmall は Proposal（2bit & State）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 Proposal（2bit & State）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，Proposal（2bit & State）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る．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．Ultrasmall Soft Processor の使用ハードウェア量は，Supersmall Soft Processor 以下である．ゆえに，Ultrasmall Soft Processor は世界最小のソフトプロセッサである．

5. Ultrasmall Soft Processor の応用

Ultrasmall の応用例としてメニーコア化を考える．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．

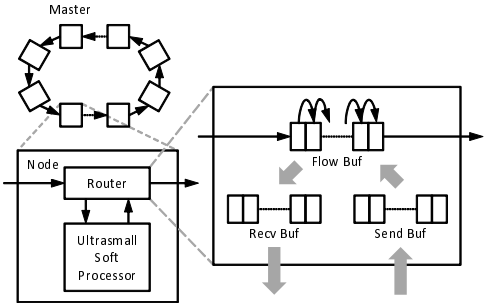

図 14　マルチコア化に向けたネットワーク構成

表 4　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

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 14** 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．Barrel Shifter のように 1 ビットずつシフトするバッファを使い，常に 1 サイクルに 1 ビットずつ隣のルータへデータを送信する．既定のサイクル（バッファのビット数と等しいサイクル数）毎に正しいデータが揃うことになるので，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受信の判断を行う．

まだ少数のコアでの実装例であるが，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 4** 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．

表 4 のコア間の使用 Slice 数を比較すると，2 コアから 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，1-2 コア間と 64 コアからは Slice 数が 2 倍以上となっている．1-2 コア間で 2 倍以上となっているのは，コアの構成がパッキングに適していなかったためと考えられる．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，64 コア以上では 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．

6.　まとめ

本稿では，ソフトプロセッサの最小化を目的として，既存研究である Supersmall Soft Processor に対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．また，その応用としてメニーコア化を検討した．

Supersmall Soft Processor は，ユニットの簡単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．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向上した．また，Supersmall Soft Processor は状態遷移，並びにマルチプレクサの最適化が不十分であった．Ultrasmall Soft Processor は Supersmall Soft Processor よりも使用ハードウェア量を抑え，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 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．

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] Stratix Device Handbook, Altera, 2013 年 1 月 31 日
[3] Stratix II Device Handbook, Altera, 2013 年 1 月 31 日
[4] Stratix III Device Handbook, Altera, 2013 年 1 月 31 日
[5] Spartan-3E FPGA ファミリ：データシート（全モジュール），Xilinx, 2013 年 1 月 31 日
[6] Spartan-6 FPGA Block RAM Resources User Guide, Xilinx, 2013 年 1 月 31 日
[7] 7 Series FPGAs Memory Resources User Guide, Xilinx, 2013 年 1 月 31 日