LC-001

ソフト・マクロ CPU を用いた FPGA 上でのマルチコア並列処理環境の実現と評価

Realization and Evaluation of Multi-Core FPGA Systems using a Soft-Macro CPU

船附 誠弘† Masahiro Funatsuki
山崎 勝弘† Katsuhiro Yamazaki

１．はじめに

情報家電や携帯端末のような様々なアプリケーションを搭載する組み込みシステムで，マルチコアの採用が進んでいる．マルチコア化を進める背景には，分散処理によるリアルタイム処理能力の向上や，部品点数の減少によるコストの削減，さらには省消費電力効果などが挙げられる．また，組み込み機器業界において，再構成可能な FPGA をハードウェアプラットフォームとして導入を進める動きがある．元々処理の一部を専用ハードウェアとして実装するために FPGA が用いられてきたが，近年では，組み込みプラットフォームとしての付加価値を高めるために，プロセッサコアやメモリ，ハードワイヤード高速乗算器，豊富な入出力機能など，ストラクチャード ASIC などと比べても遜色のない進化を遂げてきた．しかも，電子機器メーカは限られた予算や開発におけるリスク，さらには設計期間の短期化など，数多くの問題を抱えており，FPGA はこれらの要件を満たすことができる有力なソリューションとして今後のさらなる発展が期待できる．ソフトプロセッサによるマルチコアシステムの研究は，現在，海外を中心に盛んに行われている[1-2].

本研究では，上記背景を考慮した上で，FPGA 上でのマルチコアシステムを提案し，実装と並列処理計算の評価を行う．マルチコアシステムには Xilinx 社が提供しているソフト・マクロ CPU の MicroBlaze をマスタプロセッサとして用い，HDL で設計したオリジナルのプロセッサがスレーブとして MicroBlaze に複数接続するヘテロジーニアスな構成を採っている．タスクは空いているスレーブ CPU に動的に割り当てることができるため，1 つのタスクを複数のプロセッサで並列処理することができる．本論文では，今回実現したマルチコアシステムの概要と総和計算を用いた並列処理の評価，及び標準のスレーブプロセッサ接続インタフェースの策定と実装，さらに動的タスク割り当てに対応するためのプロセッサ間の可変量通信機構の実装と評価についても述べる．

２．MicroBlaze を用いたマルチコア環境の実現

2.1 マルチコア並列処理システムの構造

本研究で用いたソフト・マクロCPUのMicroBlazeは，Xilinx社が提供するEDK(Embedded Development Kit)の一部として含まれており，EDKのシステムツールを用いることで，組み込みアプリケーションの包括的な設計が可能である．EDKでは用意された通信インタフェースを用いることで，MicroBlazeシステムにオリジナルのハードウェアIPを接続することができ，これを利用してマルチコア環境の構築を行った．図 1にMicroBlazeを用いたマルチコア並列処理システムの構造を示す.

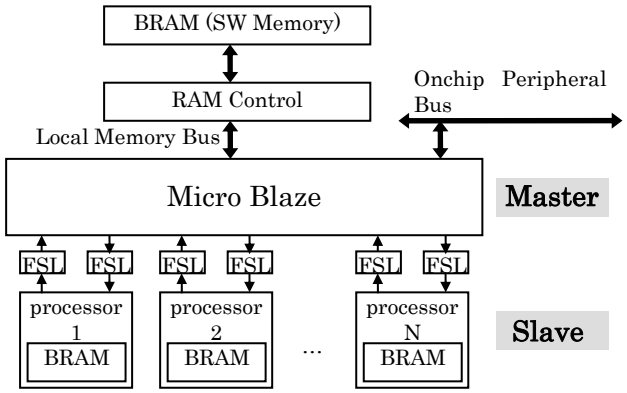

図 1 MicroBlaze を用いたマルチコア並列処理システム

MicroBlaze はシステムにおけるマスタに位置し，FPGA 内部の BRAM に C プログラムの実行コードが格納される．スレーブプロセッサは，FSL と呼ばれる FIFO 型の通信インタフェースで MicroBlaze に直接接続される．FSL は送信用１つと受信用１つを合わせて１組と考え，全部で８組用意されている．MicroBlaze から命令コードや処理データをスレーブのローカルメモリに書き込み，スレーブの処理実行後にローカルメモリからデータを回収することで，複数スレーブプロセッサによる並列処理が行われる．

マスタではCプログラムのライブラリとしてFSLの通信マクロが用意されており，ブロッキングにより容易に通信の管理ができる．図 2にCプログラムによるスレーブとの通信の例を示す.

```
int INST[14] = {          //スレーブの機械語命令列
  0xd000, 0x4249, 0x4009, 0x8840, 0xe7f0,
  0xd809, 0x0800, 0x76, 0x2f, 0xaa, 0x01,
  0x33, 0x03, 0x0f };
int DATA = 10000;
int ANS;
                    …
//命令・データをスレーブに送信
for(i=0;i<14;i++)  putfsl(INST[i] , 0);
for(i=0;i<1;i++)  putfsl(DATA , 0);
//計算結果をスレーブから回収
for(i=0;i<1;i++)  getfsl(ANS , 0);
```

図 2 Ｃプログラムによるスレーブとの通信

本研究ではマスタのプログラムにスレーブの機械語命令とデータを配列として格納している．マスタ・スレーブ間のデータ通信回数はスレーブ側の HDL のパラメタとして静的に指定されている．マスタはスレーブへの命令とデータの送信を putfsl，計算結果を getfsl で回収する．それぞれのマクロに使用している FSL インタフェースの id を指定することでスレーブを指定でき，命令とデータを複数のス

†立命館大学大学院理工学研究科，Graduate School of Science and Engineering, Ritsumeikan University

レーブに連続して送信することで，プロセッサが並列に処理を行うことができる．

2.2 オリジナルプロセッサとのデータ通信と制御

図 3にFSLインタフェースを用いたオリジナルスレーブプロセッサ接続のデータ通信構造を示す．

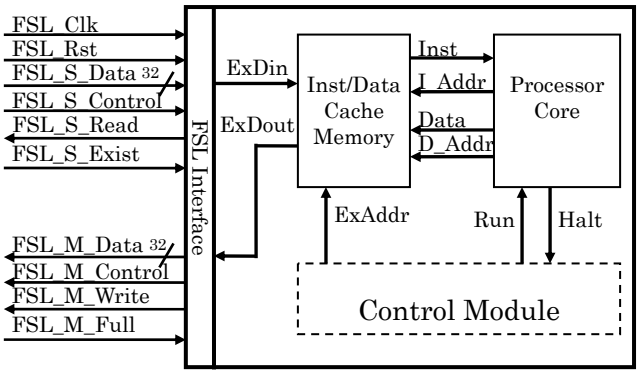

図 3　オリジナルスレーブプロセッサの通信構造

全体の構造は，処理を実行するプロセッサコアモジュール，命令とデータを格納するキャッシュメモリ，及び全体の制御を行う Control Module 部で構成される． Control Module では，FSL を介した命令とデータの入出力管理，メモリへの書き込み・読み出しの制御，及びプロセッサコアへの実行制御を行う．

図 4にControl Moduleによるプロセッサ間通信の状態遷移を示す．

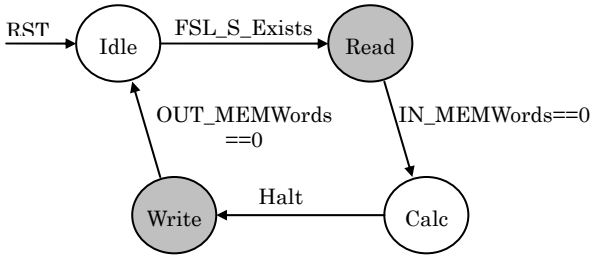

図 4　プロセッサ間通信の状態遷移

システムの起動直後に待機状態である Idle に遷移する．そして MicroBlaze からデータの送信要求を受け取ると Read に遷移し，キャッシュメモリに命令とデータが書き込まれる．FSL への Read 回数から書き込むキャッシュメモリのアドレスが決まり，規定数の通信が終わると実行状態である Calc に遷移する．Control Module で生成される Run 信号に伴ってプロセッサが動作する．プログラムの実行が終了するとプロセッサコアが Halt 信号を出力する．Control Module が Halt を検知すると，キャッシュメモリの内容を MicroBlaze へ転送する Write に遷移し，プロセッサコアでの実行結果を FSL を介して MicroBlaze が受け取ることができる．MicroBlaze への送信がすべて終われば待機状態である Idle に遷移し，再び MicroBlaze からのデータ送信要求を待つ．このような仕組みから，各スレーブプロセッサはマスタから動的にタスクを受け取って処理を行うことができる．

3. 総和計算を用いた並列処理の評価

図 1で示したマルチコアシステム上で，1 からNまでの総和演算を用いた並列処理計算の評価を行った．スレーブプロセッサには本研究室の設計資産である教育用マルチサイクルアーキテクチャのKUE-CHIP2[3]と，RISCのシングルサイクルアーキテクチャSOAR[4]を，それぞれ 32bitのデータ幅に拡張して実装した． 表 1，表 2にそれぞれKUE-CHIP2，SOAR実装時の所要クロックサイクル数を示す．ここで，所要クロックサイクル数は，マスタからスレーブへの命令とデータの送信，スレーブでの部分和の計算，スレーブからマスタへの部分和の送信，及びマスタでの総和計算の全てのクロック数の合計である．また，マスタ（MicroBlaze）でも部分和の計算を行った場合の結果を表 3に示す．

本システムでは最大 8 個までスレーブが接続できるが，FSL の 1 つをクロックサイクル数の測定ハードウェアに接続するため，7 個までの並列処理を行っている．システムクロック周波数は KUE-CHIP2 実装時は 50MHz，SOAR 実装時は 25MHz である．

表 1　KUE-CHIP2 による所要クロックサイクル数

N の値 スレーブ数	5000	20000	40000
1	55167 (1)	220167 (1)	440167 (1)
2	27815 (1.98)	110315 (1.99)	220315 (1.99)
3	18789 (2.93)	73789 (2.98)	147126 (2.99)
4	14361 (3.84)	55611 (3.95)	110611 (3.97)
5	11759 (4.69)	44759 (4.91)	88759 (4.95)
6	10070 (5.47)	37570 (5.86)	74233 (5.92)
7	8909 (6.19)	32482 (6.77)	63909 (6.88)

括弧内は速度向上比

表 2　SOAR による所要クロックサイクル数

N の値 スレーブ数	5000	20000	40000
1	15091 (1)	60091 (1)	120091 (1)
2	7680 (1.37)	30180 (1.99)	60180 (1.99)
3	5297 (1.99)	20297 (2.96)	40298 (2.98)
4	4146 (2.55)	15396 (3.90)	30396 (3.95)
5	3493 (3.03)	12493 (4.80)	24493 (4.90)
6	3089 (3.42)	10589 (5.67)	20588 (5.83)
7	2829 (3.74)	9258 (6.49)	17829 (6.73)

括弧内は速度向上比

表 3　MicroBlaze との実行結果の比較

N の値 実行プロセッサ	5000	20000	40000
MicroBlaze のみ	30003 (1)	120003 (1)	240003 (1)
KUE-CHIP2×7	8909 (3.36)	32482 (3.69)	63909 (4.38)
KUE-CHIP2×7 + MicroBlaze	7931 (3.78)	28556 (4.20)	56056 (4.99)
SOAR×7	2829 (10.06)	9258 (12.96)	17829(15.70)
SOAR×7 + MicroBlaze	5297 (5.66)	20297 (5.91)	40298 (6.95)

白枠内はクロックサイクル数(速度向上比)

スレーブ数が 1 の時を基準にした場合，KUE-CHIP2，SOAR いずれの場合でもスレーブ数に比例した速度向上が得られていることがわかる．また N の値の比較から，計算量が多いほど，全実行時間に占める通信の割合が相対的に小さくなり，より理想的な速度向上が得られた．

一方で，表 3のようにMicroBlazeのみと比較すると，KUE-CHIP2，SOARそれぞれ 7 スレーブで実行した方がMicroBlazeよりも速く処理できることがわかる．KUE-CHIP2 については，MicroBlazeも加えて 8 等分の負荷分散で処理するとより多くの速度向上が得られるが，SOARの場合はむしろ悪化している．これはSOARがシングルサイクルのため，パイプラインアーキテクチャのMicroBlazeがSOARと同じシステムクロック周波数ではスループットが相対的に低くなっていることを示している．従って，MicroBlazeを最大限に活用するためには，マスタとスレーブで別系統のクロック供給が必要であると考えられる．

4. FPGA マルチコア環境の機能拡張と検討

4.1 標準接続インタフェースの策定と実装

前章のマルチコアシステムではスレーブのプロセッサにKUE-CHIP2，SOARを用いた．両者の大きな違いとして，KUE-CHIP2 は命令とデータのメモリアクセスのデータパスを共有しており，SOARは命令，データそれぞれが別のデータパスを持つ．そのため，図 3で挙げたメモリ・プロセッサコア間の通信プロトコルやControl Moduleの状態遷移に違いが生じた．そこで様々なアーキテクチャのプロセッサを，通信構造を変更することなく接続できるようにすることを目的に，標準となるプロセッサインタフェースの策定を行った．その際，本研究室で開発しているプロセッサデバッガ[5]のインタフェースを用いた．図 5にプロセッサデバッガのインタフェースを示す．

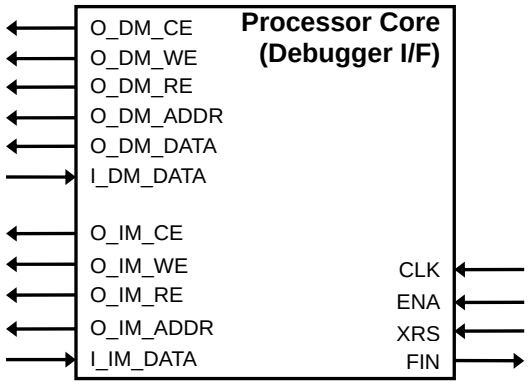

図 5　標準インタフェースに基づくプロセッサ入出力

プロセッサデバッガは本来プロセッサ設計教育において，FPGA実装時に学習者が実行結果や内部状態を自由に観測するために開発されたツールである．図 5の入出力に準拠した場合に，シングルサイクルからパイプライン，スーパスカラまで様々なアーキテクチャを開発することが可能である．本研究ではこの特徴を利用し，図 5の入出力に対応したメモリインタフェースやプロセッサ実行制御部の設計を行った．メモリアクセスは命令，データそれぞれ独立し，プロセッサコアはENAがアクティブの間，処理の実行を行い，実行終了時にパルス状のFIN信号を出力する．標準インタフェースに基づいたプロセッサコアの実装を行い，MicroBlazeとの接続動作検証を行った．表 4に各プロセッサコアの実装結果を示す．

表 4　標準インタフェースによる実装結果

processor	Architecture	Slices	MAX freq.(MHz)
SOAR 16bit	Single Cycle	775	34.8
SOAR 32bit	Single Cycle	1000	33.3
MONI 16bit	Multi Cycle	1032	76.9

実装には SOAR と MONI[6]の 2 つの命令セットを用い，SOAR においてはデータビット幅の異なるプロセッサコアを，MONI についてはマルチサイクル方式と，異なるアーキテクチャの実装をすることができた．インタフェースを標準化することで，プロセッサコア周りの通信構造は基本的に修正を加えずに実装ができた．すなわち、標準インタフェースを提供することにより，ユーザは異なるプロセッサ毎にコントロールモジュールを設計する手間がなくなり，プロセッサコアの設計に注力することができる．

4.2 プロセッサ間データ通信の可変量化

提案したマルチコアシステムはHW/SWの協調設計システム[7]をベースにしており，当初はFSLによる通信量をHDLのパラメタで静的に指定していた．そのため，単一のスレーブに対して動的に複数のタスクを割り当てた場合に通信効率が悪くなる．そこで，従来の固定量のデータ通信方法から，タスク割り当て時にマスタが自由に通信量を指定できる可変量データ通信方式を検討し，通信構造の改良を行った．図 6に可変量通信を適用したときの状態遷移を示す．

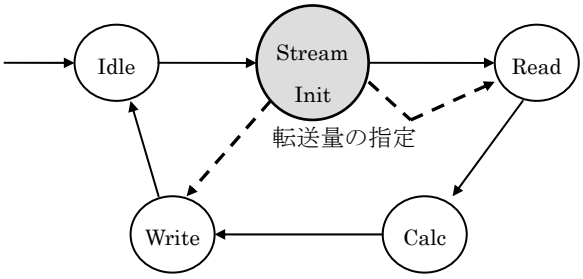

図 6　可変量通信適用後の状態遷移

Idle と Read の間に転送量情報をマスタプログラムから受け取る状態 Stream Init を追加した．マスタから受け取ったデータがそのままレジスタに保存され，保持された内容が Read , Write フェーズで参照されて指定回数分のデータ転送を行う．マスタからのデータのみで制御を行うデータ駆動方式で，FSL の通信を除く外部からの制御線は無い．実現した可変量通信の実装と評価を KUE-CHIP2 を用いて行った．表 5 では 1 つのスレーブに対して 3 つのプログラムを順次割り当て，固定量通信と比較を行っている．

固定量通信の場合，通信量の最も多いバブルソートが基準になり，他のプログラムは空データの通信をするため非常に効率が悪い．一方で可変量通信では，各プログラムの転送に 2 ワード分の転送量情報を送らなければならないが，転送量が必要最低限に調節されているため，固定量通信の

約半分の転送量に削減できた．また，所要クロックサイクル数も約 200 削減している．このことから動的タスク割り当てにおける可変量通信の有効性を示すことができる．

表 5　固定量通信と可変量通信の比較

データ転送量(ワード)	固定量通信	可変量通信
1 から 10 までの和	64(R:56,W:8)	17(R:14,W:1,SI:2)
最大公約数(16 進 60,40)	64(R:56,W:8)	21(R:18,W:1,SI:2)
バブルソート(8 ワード)	64(R:56,W:8)	66(R:56,W:8,SI:2)
データ転送量合計	192	104
所要クロックサイクル数	2452	2256

(R: Read , W: Write , SI: Stream Init の転送量)

4.3　FSL を用いた Many-Core 構造の検討

MicroBlazeに直接接続できるFSLは 8 組であり，1 スレーブに 1 組のFSLを割り当てると最大 8 スレーブしか接続ができない．そこで，より多くのプロセッサコア接続を実現するために，1 組のFSLに複数のプロセッサコアを接続することを検討し，図 7のような構造を提案する．

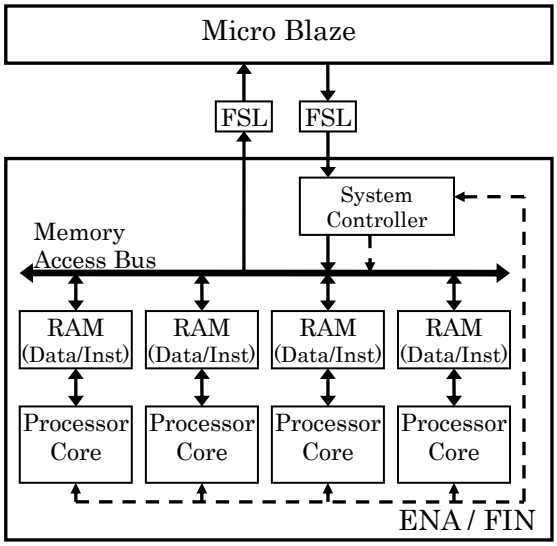

図 7　FSL を用いた Many-Core 構造

図 7 ではコア数を 4 つにした場合を示している．全体の制御はSystem Controllerが担当し，メモリアクセスバスの制御や各プロセッサの実行制御を行う．制御方法は図 4 や図 6が複数重なった状態遷移で実現される．図 7 ではコア数が 4 つであり，FSLを 8 組用いると 32 コアのスレーブを持つことになる．もちろんFSL一組で接続できるコア数には際限が無く，デバイス容量に応じた数のコア接続が期待できる．すなわち，命令メモリを共有して仮想的なSIMDを構成したり，可変量通信で用いたデータ駆動制御によってプロセッサの並列度を動的に指定したり，さらには複数種のプロセッサを待機させて選択実行する等，様々な可能性が期待できる．

5. おわりに

本論文では，MicroBlaze をマスタとした FPGA 上でのマルチコアシステムの提案を行い，1 から N までの総和計算について 7 つのスレーブプロセッサで均等に負荷分散して並列に実行し，プロセッサ数に比例した速度向上が得られることを示した．また，標準となるインタフェースを策定し，複数の異なるアーキテクチャのプロセッサコアの実装が確認できた．このことから通信構造を変更することなく，スレーブアーキテクチャを自由に差し替えることができた．すなわち，MicroBlaze では FSL を 8 組使用できるので，複数のプロセッサコアを容易に接続できること，及び標準インタフェースの導入により，異なるアーキテクチャのプロセッサに容易に差し替えられることを実験的に示した．さらに，プロセッサ間通信機構を改良し，マスタのプログラムから動的にデータ転送量を指定することが可能になった．このことから，FPGA マルチコア並列処理システム上で，高い性能を出せる可能性があることを示した．

今回は総和計算のみでマルチコア並列処理システムの可能性を示したが，今後，画像処理などの実用的な問題で，本システムの有効性を示すことが必要であろう．また，メディア処理専用プロセッサの接続なども行って，より汎用性と機能性の高い通信構造について検討したい．

参考文献

[1] Y.Jin, N.Satish, K.Ravindran, K.Keutzer, ”An Automated Exploration Framework for FPGA-Based Soft Multiprocessor Systems ”, International Conference on Hardware Software Codesign (Algorithms and Methodologies for New Architectures) , 2005.

[2] J.Ou, V.K.Prasanna, ” Design Space Exploration Using Arithmetic-Level Hardware–Software Cosimulation for Configurable Multiprocessor Platforms ”, ACM Transactions on Embedded Computing Systems (TECS), vol.5, issue2, pp.355-382, 2006.

[3] 船附誠弘, 山崎勝弘, 小柳滋：教育用マイクロプロセッサの設計と FPGA 上での検証,第 4 回情報科学技術フォーラム(FIT2005),C-001,2005.

[4] 難波翔一朗,山崎勝弘,小柳滋：マイクロプロセッサの設計と検証に基づいたハード/ソフト・コラーニングシステムの拡張,第 4 回情報科学技術フォーラム(FIT2005),LC-001,2005.

[5] 難波,中村,Tuan,山崎,小柳：ハード/ソフト協調学習のための命令セット定義ツールとプロセッサデバッガの開発,第 5 回情報科学技術フォーラム(FIT2006),N-009,2006.

[6] 大八木,池田,山崎,小柳：ハード/ソフト・コラーニングシステムにおけるアーキテクチャ選択可能なプロセッサシミュレータの設計,情報処理学会第 66 回全国大会論文集,5T-6,2004.

[7] 船附誠弘, 山崎勝弘, 小柳滋：Handel-C による SHA-1 の設計とハードウェア/ソフトウェア最適分割の検討,第 5 回情報科学技術フォーラム(FIT2006),C-002,2006.

[8] Avnet Japan: http://www.jp.avnet.com