設 計 自 動 化 77－22
(1995. 10. 20)

複数FPGAによるラピッドシステムプロトタイピング環境
— システム集積化LSIコデザイン開発環境の構築に向けて —

末吉 敏則 †1　　大内 正英 †2　　田中 康一郎 ‡3

† 九州工業大学 情報工学部 知能情報工学科
‡ 九州工業大学 マイクロ化総合技術センター

〒820 福岡県飯塚市川津 680-4

E-mail : [1]sueyoshi@ai.kyutech.ac.jp,[2]oouchi@mickey.ai.kyutech.ac.jp
[3]tanaka@cms.kyutech.ac.jp

あらまし シリコン技術とシステム技術の融合に伴い，統合設計技術を実現するシステム集積化LSIコデザイン開発環境を構築するため，その基盤として書換え可能なFPGAによるラピッドシステムプロトタイピング環境の整備を進めている．FPGAは比較的大規模な回路を実装できる論理素子として注目を集め，最近では単体で2万ゲートを越えるものまで入手できる．しかしながら，システムレベルのプロトタイピングにはなお複数のFPGAを用いた実装にならざるを得ないのが実状である．そこで我々は，書換え可能なFPGAとFPGA間の配線を自由に変更できる配線用デバイスFPIDを利用し，最適なシステム構成を選択するために必要な試行錯誤を容易に行え，かつソフトウェアを含むシステム検証が円滑に行える複数FPGAによるラピッドシステムプロトタイピング環境を構築した．本稿では，このラピッドシステムプロトタイピング環境について説明すると共に，エミュレーションによるシステム検証を実現した適用事例を紹介する．

キーワード ラピッドシステムプロトタイピング，コデザイン，システム集積化LSI，FPGA，配線用デバイス

An Environment for Rapid System Prototyping using Multiple FPGAs
—Toward Constructing of the Co-design Environment for System Integrated LSI Development—

Toshinori Sueyoshi †1　　Masahide Ouchi †2　　Koichiro Tanaka ‡3

† Department of Artificial Intelligence, Kyushu Institute of Technology
‡ Center for Microelectronic Systems, Kyushu Institute of Technology

680-4 Kawazu, Iizuka City, Fukuoka, 820 Japan

E-mail : [1]sueyoshi@ai.kyutech.ac.jp,[2]oouchi@mickey.ai.kyutech.ac.jp
[3]tanaka@cms.kyutech.ac.jp

Abstract As silicon and system technologies harmonize, we are preparing a rapid system prototyping environment using reconfigurable FPGAs(Field Programmable Gate Array) as a base to construct an environment for system integrated LSI co-design. FPGA is a widely known logic device which is capable of realizing its internal circuit by programming. Recent developments in FPGA can accommodate over 20,000 gates in one FPGA device. However, problems arises when implementing a system prototype requiring multiple FPGAs. Here, we utilize the reconfigurable FPGAs and FPIDs (Field Programmable Interconnect Devices) which are reconfigurable interconnecting devices between FPGAs. We are currently developing a rapid system prototyping environment using multiple FPGAs where one can smoothly try and evaluate the design to choose the most suitable system architecture, and further verify the system software. In this paper, we describe the rapid system prototyping environment, and introduce examples whose system verification was realized using emulation.

key words rapid system prototyping, co-design, system integrated LSI, FPGA, interconnecting device

1 はじめに

集積回路技術の進歩により集積度がめざましく向上し，システム自体をLSIに詰め込むシステム集積化LSIが実現できるようになった．さらに，CAD技術の発展やワークステーションの普及により，比較的短期間でのLSI開発が可能になった．こうしたシリコン技術とシステム技術の融合に伴い，コンセプトメーキングから，評価，設計，動作検証，実装，アプリケーション開発までのすべてを効率よく総合的に行えるシステム集積化LSI開発のための統合環境が求められている．これを実現するには，従来のハードウェア/ソフトウェア・コデザイン手法に加え，ラピッドプロトタイピングのための設計支援環境を統合することが不可欠と考えられる．そこで我々は，上述のようなハードウェア/ソフトウェア・コデザイン環境の基盤としてラピッドシステムプロトタイピングを位置付け，書換え可能なFPGA(Field Programmable Gate Array)[6]を利用したシステム集積化LSI開発環境の構築を行っている[1][2]．本稿では，このラピッドシステムプロトタイピングのためのマルチFPGA実装支援環境について紹介する．以下，第2章では本研究の背景および目的を述べ，第3章ではマルチFPGA実装支援環境について説明する．次に，第4章および第5章でマルチFPGA実装支援環境を用いたラピッドシステムプロトタイピングの事例を示し，最後に第6章でまとめを述べる．

2 背景と目的

我々が理想とするシステム集積化LSI開発環境とは，(1) コンセプト創出や方式設計におけるアイデアの評価を短期間に行え，(2) 最適なアルゴリズムやシステム構成を選択するために必要な試行錯誤が容易に実施でき，かつ (3) その設計データを用いて最終的に実システムとして実現できること．また，(4) ハードウェア設計と並行してソフトウェア開発も同時に行え，(5) ソフトウェアを含むシステム検証が円滑にできること．そのような真の意味での統合設計技術を実現するハードウェア/ソフトウェア・コデザイン環境である[3]．

特に，マイクロプロセッサなどのシステム開発においては，ハードウェアだけでなくソフトウェア開発も並行して行う，ハードウェアとソフトウェアの協調設計が望ましい．しかしながら，ソフトウェア開発環境においては，高級言語で構築されたシミュレーション環境上でアプリケーション開発を行っているため，シミュレーション結果が実際のLSI動作と異なった場合，アプリケーションが利用できない可能性がある．一方，ハードウェア開発環境においては，比較的大規模な回路が実現でき，しかもシミュレーション速度の面で満足できる環境が求められる．

そこで，書換え可能なFPGAによるラピッドプロトタイピング環境をEDA環境に統合することにより，シミュレーションに基づく設計環境で問題となるソフトウェア開発や性能評価におけるシミュレーション速度および評価精度の改善を試みた．さらに，これはハードウェアならびにソフトウェアの仕様を変更する可能性を残したままで実際のソフトウェアを実行できるため，従来は難しかったソフトウェア/ハードウェア化のトレードオフの探求 (図1) や，エミュレーションによる回路検証と並行したソフトウェア開発 (図2) も可能である．

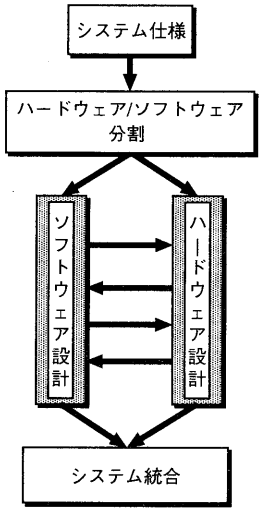

図 1: ソフトウェア/ハードウェア化のトレードオフ

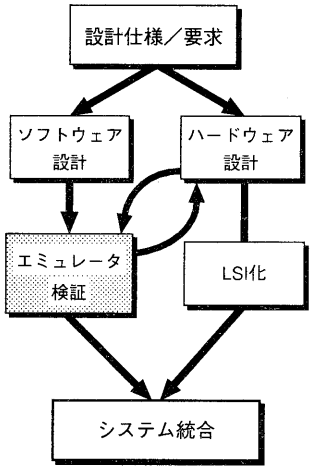

図 2: エミュレーションによるシステム検証 (ソフトウェア開発を含む)

FPGAは比較的大規模な回路を利用者側でプログラム可能な論理デバイスとして急速に高性能化してきており，最近では単体で2万ゲートを越えるものも入手できるようになった．しかしながら，システムレベルのプロトタイピングを行うにはなおFPGA単体では難しく，複数のFPGAによる実装にならざるを得ないのが実状である．そこで我々は，書換え可能なFPGAと，FPGA間の配線を自由にプログラムできる配線用デバイスFPID(Filed Programmable Interconnect Device)[7]を搭載した米国Aptix社のプリント基板FPCB(Field Programmable Circuit Board)[7]を利用して，ラピッドシステムプロトタイピングのためのマルチFPGA実装支援環境の構築を図った[1]．

3 マルチ FPGA 実装支援環境

マルチ FPGA 実装支援環境は，以下に紹介するマルチ FPGA 実装支援ツールおよび実装環境で構成される．

3.1 マルチ FPGA 実装支援ツール

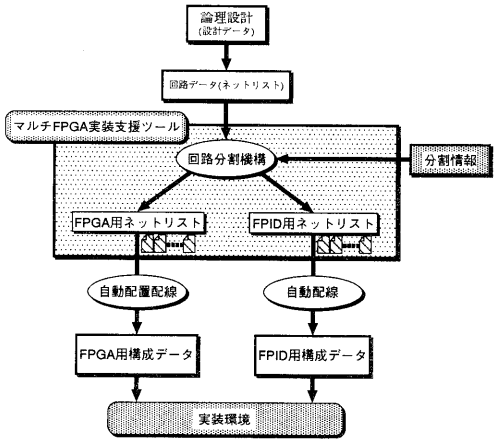

図 3: マルチ FPGA 実装支援環境

システムを設計する際，特定の機能を提供するモジュールを一つの単位として階層設計するため，モジュール単位で回路を分割・実装できることが望ましい．そこで，複数 FPGA による分割実装の場合，各モジュールをどのデバイスに分割するかの情報が必要不可欠となる．マルチ FPGA 実装支援環境では，図 3に示すように回路データと共にこのような情報を分割情報として与えることで，設計者の意図を反映した回路分割が行える．以下に示す情報を分割情報としてテキスト形式でマルチ FPGA 実装支援ツールに与える．

- **デバイスのグループ名とパーツタイプの指定**
  マルチ FPGA 実装支援環境では，全ての FPGA に固有のグループ名を付けて取扱う．また，デバイスはパーツタイプにより，その実装可能規模，ピン名，およびピン数が異なるため，実際に使用するデバイスのパーツタイプを指定しなければならない．

- **モジュールのデバイス割付け**
  それぞれのグループに割付けるモジュール名を記述する．

マルチ FPGA 実装支援ツールは，上述の分割情報と，HDL あるいは回路図入力によって設計した回路データとから，各々の FPGA 用ネットリストならびに FPID 用ネットリストを生成し，複数 FPGA への分割実装を可能にする．

3.2 実装環境

プロトタイピング環境は，図 4に示すようにプロトタイピングボード，表示/メモリボード，そしてホスト I/F ボードで構成される．プロトタイピングボードは，書換え可能な FPGA を実装用として 12 個 (別に入出力用として 2 個) 搭載している．実装デバイスとして XC4010

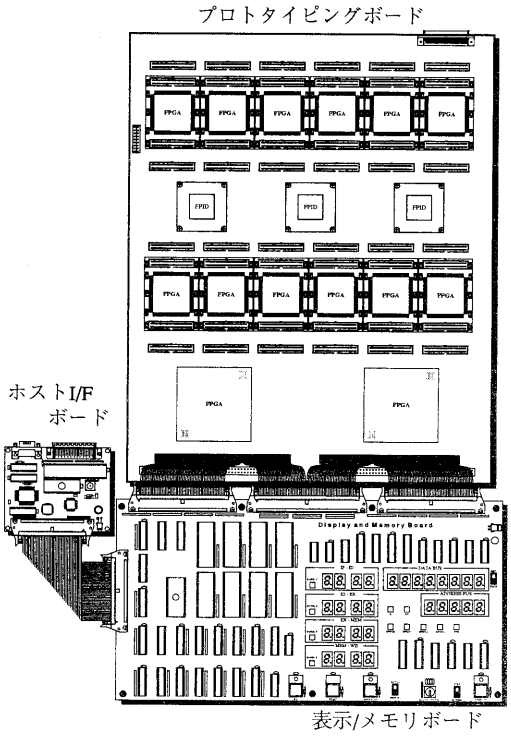

図 4: プロトタイピング環境

や XC4013(Xilinx 社の FPGA[6]) を利用すると全体で約十数万ゲート相当の回路規模まで実装でき，ASIC エミュレータと組み合わせれば数十万ゲート相当の大規模システムにも対応可能である．また，配線用デバイスである 3 個の FPID に構成データをダウンロードすることにより FPGA 間の配線が自由に変更でき，短時間で

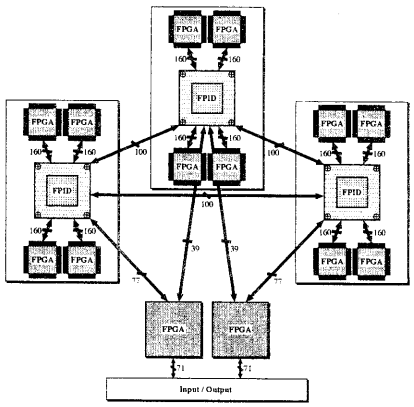

図 5: プロトタイピングボードの構成

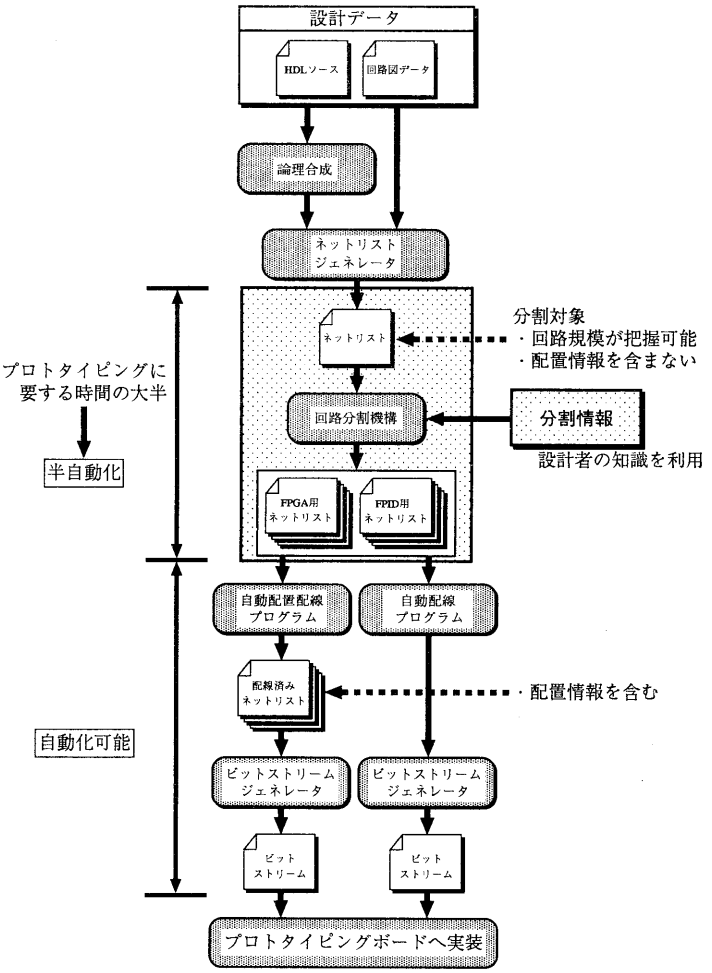

図 6: マルチ FPGA 実装支援環境における設計の流れ

さまざまなシステム構成を実現できる．FPGA および FPID は，図 5 に示すように 4 つの FPGA に対して 1 つの FPID が FPGA 間配線を担当する．各 FPGA と FPID 間は 160 本の配線資源が使用可能で，FPID 間も 100 本の配線資源が利用できる．

また，入出力用の FPGA を介した外部からのデータアクセスが可能で，表示/メモリボードを使用して，プロトタイピングボード上に実装したシステムの内部動作の観測や動作検証が行える．表示/メモリボードにはデュアルポート構成の SRAM を搭載しており，ホスト I/F ボードの接続により各種ホストコンピュータからのプログラムのロードやデータ転送が可能である．

3.3 実装支援環境における設計の流れ

図 6にマルチ FPGA 実装支援環境における設計の流れを示す．本実装支援環境は，設計者が HDL あるいは回路図入力を用いて設計した設計データから，論理合成ツールおよびネットリストジェネレータによりネットリスト形式の回路データを生成する．回路データとともに図 7に示す分割情報を回路分割機構に渡して，ネットリストレベルでの回路分割を行う．ここで，分割の対象をネットリストとすることで，設計手法に依存しない回路分割を可能にした．論理回路の分割により，各々の FPGA 用ネットリストおよび FPGA 間の配線を定義した FPID 用ネットリストを自動生成する．そして，得られた FPGA 用ネットリストを自動配置配線プログラム，ビットストリームジェネレータによってプロトタイピン

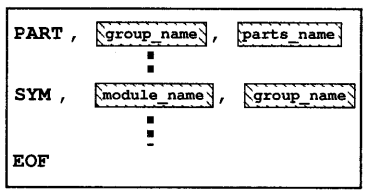

図 7: 分割情報ファイルフォーマット

・各デバイスの端子数
・各デバイスの回路規模
・各デバイスへ実装されているシンボル
・シンボル間のネット数
・各シンボルの回路規模
等々

図 8: 分割状況レポート

グボードに実装可能なフォーマットへ変換する．同様にして FPID 用ネットリストの変換も行い，それぞれのビットストリームデータをプロトタイピングボードに実装して動作検証や評価を行う．

本実装支援ツールは，回路分割時において実装デバイスの物理的制限 (使用ピン数や実装規模など) のため分割情報に従った回路分割が実行できない場合，分割不可能であったことを通知すると同時に，図 8に示すような分割状況レポート (使用ピン数や FPGA 間の接続状況など) を設計者にフィードバックする．設計者はこれらの情報に基づいて，分割情報の変更や論理回路の再設計を行い，再度回路分割を実行する．

4 回路分割実装例 1

本章では，マルチ FPGA 実装支援環境における分割実装例として，32 ビット RISC マイクロプロセッサ DLX-FPGA の回路分割例とその実装結果について示す．

4.1 概要

DLX-FPGA は，実装デバイスとして書換え可能な FPGA を利用した上級コース向き教育用マイクロプロセッサであり，プロセッサ・モデルとして DLX アーキテクチャを採用している．以下に，その特徴を示す．

DLX アーキテクチャの採用： DLX は文献 [5] で紹介される RISC 型のマイクロプロセッサであり，実用化された商用の計算機を平均化したような構成をしている．計算機アーキテクチャ教育における Load/Store アーキテクチャや命令パイプラインなどの高速化手法，パイプライン処理の流れを乱す様々なハザードに対する回避法 (フォワーディング，分岐予測)，またシステムソフトウェア教育における最適化コンパイラ技術といった実用レベルに近いシステム設計教育を支援する教材として有効である．

書換え可能な FPGA の利用： 実装デバイスとして書換え可能な FPGA を利用しており，マイクロプロセッサの設計完了後，短時間で LSI 化でき動作検証が行える．また，書換え回数に制限がないため，デバッギングおよび設計回路の改良・変更が容易である．

我々は先に，整数乗除算命令および浮動小数点命令をサポートしないサブセット版 DLX-FPGA マイクロプロセッサが換算ゲート数にして約 28,000 ゲート (VHDL による設計) であり，当時の FPGA のゲート規模を考慮すると単一の FPGA では実装不可能であることを報告した [4]．そこで，回路設計に自由度を持たせることを考慮し，マルチ FPGA 実装支援環境を用いた複数 FPGA による分割実装を行った．

以下，整数乗除算命令および浮動小数点命令をサポートしない DLX-FPGA を 4 つの FPGA に分割実装した場合の結果について報告する．

4.2 分割実装

設計手法には，回路図入力およびハードウェア記述言語 (VHDL を利用) を使用し，FPGA への**分割を考慮した (回路図による) 設計**と，**分割を考慮しない (VHDL による) 設計**を行い，回路分割・実装を行った．DLX-FPGA を 4 つの FPGA に分割した時の機能分割案を図 9に示す．

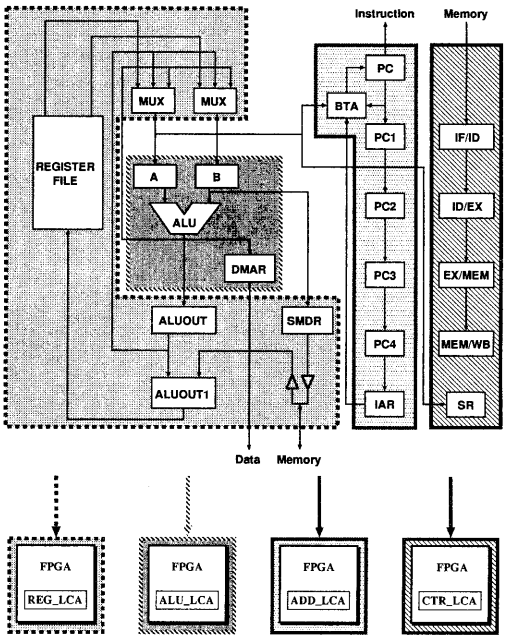

図 9: DLX-FPGA における機能分割例

```
PART, REG_LCA, 4010p208
PART, ALU_LCA, 4010p208
PART, ADD_LCA, 4010p208
PART, CTR_LCA, 4010p208
SYM, REGISTER_FILE, REG_LCA
SYM, ALU, ALU_LCA
SYM, PC, ADD_LCA
SYM, IR, CTR_LCA
      ⋮
EOF
```

図 10: DLX-FPGA の分割情報ファイル例

図 10に，分割案 (図 9) から作成した DLX-FPGA 用分割情報ファイルの一部を示す．DLX-FPGA を 4 つの FPGA (REG_LCA, ALU_LCA, ADD_LCA, CTR_LCA) に分割実装することを指示している．

4.3 実装結果

回路分割の最大の問題点は，分割に伴う入出力端子数の増加である．図 4に示すように，各 FPGA 間および

FPID 間の配線資源は限定されており，実装デバイスとしての FPGA には，160 本の入出力端子を割付けることしかできない．そこで，回路の分割結果については特に入出力端子数に着目して，物理的制限を越えない範囲での回路分割を実現した．

- **回路図入力による設計**

回路図入力による設計では，あらかじめ回路を 4 つの FPGA に分割実装することを意識した設計を行った．そのため，マルチ FPGA 実装支援ツールを用いることで，分割情報 (図 10) に従った回路分割が 1 回の試行で問題なく実現できた．回路図入力による設計データの回路分割結果を表 1に示す．各 FPGA に割当てられた入出力端子数は入出力端子数の制限の 160 本より少なく，全て実装可能であることが分かる．なお項目中の Input，Output はそれぞれ入出力端子であり，Input(B),Output(B),In-Out(B) は，双方向端子であるが，入力ライン，出力ラインおよび双方向ラインとして使用される端子を示している．これらは回路分割によって新たに生成されたラインである．一方，External は回路分割前から定義されていた入出力ラインを示している．

表 1: 回路分割後の入出力端子数 (回路図版)

Component \ Line	Input	Output	Input(B)	Output(B)	In-Out(B)	External	Total
REG_LCA	23	32	32	32	0	32	151
ALU_LCA	45	4	32	32	0	15	128
ADD_LCA	46	6	0	32	32	17	133
CTR_LCA	9	44	0	0	32	46	131

- **VHDL による設計**

回路分割を意識せず設計した DLX-FPGA の設計データと図 10に示す分割情報とを，マルチ FPGA 実装支援ツールに与え回路分割を実行した．回路分割実行後の各 FPGA における入出力端子数を表 2に示す．ALU_LCA で示される FPGA のみ実装可能で，その他の FPGA(REG_LCA，ADD_LCA，CTR_LCA) は，入出力端子数が制限を大幅に上回り，実装不可能であることが分かる．本実装支援環境は，この分割結果より再設計を支援するために，分割状況レポートとして表 3 に示すような各デバイス間の結線情報を詳細に報告する．DLX-FPGA は 32 ビット・マイクロプロセッサであるため，各 FPGA 間の大部分が 32 本を単位とする配線により接続されると予想できる．表 3の結果を図式化すると，図 11 に示すように各 FPGA 間が多数の単方向ラインで接続されていることが明らかになった．これらのデータパスは全て同時に使用されない．そこで，同時に使用する必要がないデータパスのグルーピングを行い，図 12に示すような 3 つのパスを利用して回路を再設計した．

再設計後の設計データを再度マルチ FPGA 実装支援ツールを用いて回路分割した結果，表 4に示す結果が得られた．各 FPGA の入出力端子数が制限以内に抑えられていることが読みとれる．また，図 5では，再設計後

表 2: 回路分割後の入出力端子数 (VHDL 版)

Component \ Line	Input	Output	Input(B)	Output(B)	In-Out(B)	External	Total
REG_LCA	192	96	0	0	0	32	320
ALU_LCA	74	35	0	0	0	15	124
ADD_LCA	113	70	0	0	0	17	200
CTR_LCA	40	115	0	0	0	42	197

表 3: デバイス間接続情報 (VHDL 版)

(a)単方向ライン

Output	Input	Number	Output	Input	Number
REG	ALU	32	ADD	REG	65
REG	ALU,ADD	32	ADD	REG,ALU,CTR	2
REG	ADD,CTR	32	ADD	CTR	3
ALU	REG	32	CTR	REG	58
ALU	REG,CTR	2	CTR	REG,ADD	33
ALU	CTR	1	CTR	ALU	8
			CTR	ADD	16

の設計データが双方向ラインを用いた設計をし，それを反映した回路分割が実現されたことを示している．

分割状況レポートから回路構成を検討し，その結果，単方向ラインで形成されていたデータパスを双方向ラインに置き換えることで，設計者の意図した回路分割が実現できた．

表 4: 再設計後の入出力端子数 (VHDL 版)

Component \ Line	Input	Output	Input(B)	Output(B)	In-Out(B)	External	Total
REG_LCA	25	32	32	32	0	32	153
ALU_LCA	44	3	32	32	0	15	126
ADD_LCA	50	5	0	32	32	17	136
CTR_LCA	8	49	0	0	32	42	131

マルチ FPGA 実装支援ツールにより 4 つの FPGA に回路分割された DLX-FPGA を，プロトタイピングボード上に実装し，いくつかのアプリケーション (ハノイの塔，総和計算，ダイクストラ法による最短路問題など) を実行した．ハノイの塔を解くアプリケーションプログラムでは，データ用メモリにストアした演算結果を，ホスト計算機がホスト I/F ボードを介してアクセスする形態を採った．ホスト計算機ではその演算結果から円盤の動きをグラフィカルに表現し，正しい動作が行われていることを視覚的に示した．また，DLX-FPGA の動作クロックを変化させた場合でも，プロセッサ動作に同期した動作結果が得られた．このように，回路の動作検証を含めたソフトウェア開発も容易に行えることを確認した．

なお，整数乗除算命令および浮動小数点命令をサポートした 10 万ゲート相当のフルセット版 DLX-FPGA についても同様の手順で，設計手法に依存することなく回路分割を実現した [2].

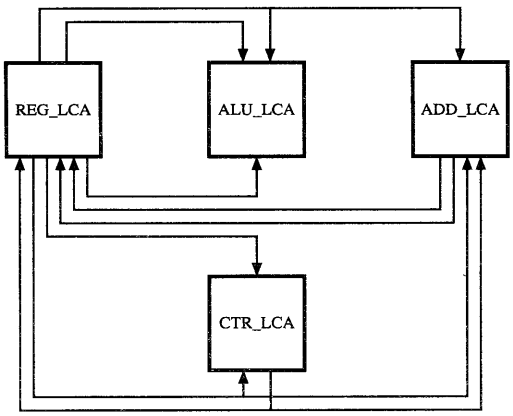

図 11: DLX-FPGA のデータパス

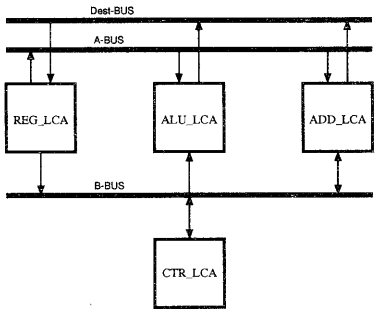

図 12: 再設計後の DLX-FPGA のデータパス

5 分割実装例 2

ここでは，設計データを修正することなく分割情報の変更により分割を実現した例として，バックプロパゲーション・アルゴリズムに基づく文字認識のためのニューラルネットワーク・エミュレータの分割実装例を示す．

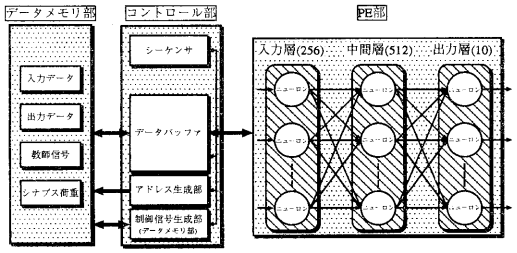

図 13: ニューラルネットワーク・エミュレータの構成

ニューラルネットワークは，演算の高並列性を内包しており，同時に複数の演算を実行することで処理の高速化を図ることができる．LSI 化を前提としたニューラルネットワーク・エミュレータを設計する場合には，アプリケーションを処理するために要求される演算器構成やネットワーク形態を早期に決定し，直ちにプロトタイプ

表 5: 再設計後のデバイス間接続情報 (VHDL 版)

(a)単方向ライン

Output		Input	Number
REG	→	ALU_ADD	32
ALU	→	CTR	3
ADD	→	REG_ALU_CTR	2
ADD	→	CTR	3
CTR	→	REG	22
CTR	→	ALU	9
CTR	→	ADD	17
CTR	→	REG_ALU_ADD	1

(b)双方向ライン

In-Out	Output	Input	Number
	ALU_ADD	REG	32
ADD_CTR	REG	ALU	32

の設計・実装を行える環境を提供する必要がある．そこで，マルチ FPGA 実装支援ツールを用いて，ニューラルネットワーク・エミュレータを複数の FPGA へ分割実装した．

5.1 概要

ニューラルネットワーク・エミュレータは，図 13に示すようにコントロール部，データメモリ部，プロセッサ要素 (PE) 部で構成する．コントロール部，データメモリ部，PE 部を複数 FPGA へプロトタイプ実装し，データメモリ部を，FPCB 上の入出力用 FPGA を介して接続される表示/メモリボード上で実現する．各構成要素の詳細を以下に示す．

データメモリ部： 各種のデータを並列にアクセスするために複数の独立なメモリで構成する．今回実現したバックプロパゲーション・アルゴリズムでは，入出力データ用，シナプス荷重用，教師信号用，修正誤差用があり，他にも演算の途中結果を収めておくメモリを用意する．

コントロール部： データメモリ部や PE 部の制御，アドレス生成の他に，メモリ部と PE 部間においてデータの入出力を行う．データ入出力の際，必要に応じてビット幅の変更や算術シフト演算などのデータ加工を行う．なお，システムの統括を行うシーケンサも含む．

PE 部： 実際に演算を実行する部分である．バックプロパゲーション・アルゴリズムでは主として積和演算を行うため，乗算器および加減算器を多数備える構成を採る．ソフトウェア・シミュレーションにより決定したシステム構成に必要な演算量を考慮し，乗算器数と加減算器数は 1:4 の割合で用意している．演算器の機能を制御する信号は，初期化時に与えた状態を基にクロック単位で次の演算器の制御信号として伝達する方法を利用した．これにより，リセットおよびクロック以外に大域的制御信号を必要としない演算器を構成した．

アプリケーションとして，数字の 0～9 の合計 10 文字を扱う簡単な文字認識を行った．文字のサンプルデータは 16 × 16 ドットで，各々2 値で表現されている．ソフトウェア・シミュレーションの結果より，エミュレータの構成は，階層ネットワークを 3 層とし，各階層でのニューロン数は，入力層を 256 個，中間層を 512 個，そして出力層を 10 個とした．また，演算器で取り扱うデータは主として 4 ビット長とし，演算器の小型化により多数の演算器を集積できるよう考慮した．さらに，

文字の学習を，各文字に対して10回，合計100回の学習を行うことで，十分な認識率を得られることがシミュレーション結果から確認できた．そこで，この決定仕様に基づく論理設計を行った．

5.2 実装結果

ニューラルネットワーク・エミュレータをVHDLで設計し，マルチFPGA実装支援ツールを用いて回路分割を行った．設計の段階では複数のFPGAへの分割は意識せずにモジュール単位での階層設計を行う．論理合成の後，11個のFPGA(1個当たり約1万ゲート相当)に分割するように記述した分割情報を与え，マルチFPGA実装支援ツールを用いて回路分割を行った．新たに生成された分割状況レポートをもとに図式化したデータパスを図14に示す．太い線で示されたものは双方向ラインで，VHDLソースの記述が反映された回路分割が実現されていることが分かる．なお，この例ではPMA，PMBで識別される4個のFPGAには各々16個の加減算器を，PIOで識別される2個のFPGAには各々8個の乗算器が実装されている．回路分割後の各FPGAの入出力端子数を表6に示す．この結果から，すべてのFPGAにおいて入出力端子数は物理的制限である160本以内に収まり，11個のFPGA(約10万ゲート相当)に回路分割できたことがわかる．なお，分割の際には，VHDLソースを修正することなく，実装支援ツールに与える分割情報の変更のみで回路分割を実現した．

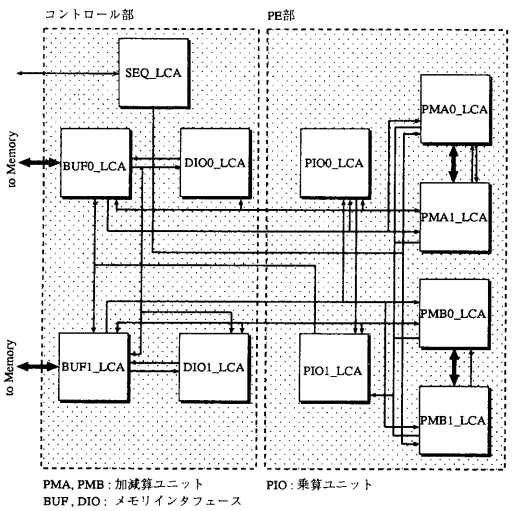

図14: エミュレータのデータパス

6 おわりに

本稿では，システム集積化LSIコデザイン開発環境の構築に向けて，その基盤となるラピッドシステムプロトタイピングのためのマルチFPGA実装支援環境について述べ，2種類の回路分割例を紹介した．最初の教育用32ビットRISCマイクロプロセッサDLX-FPGAの事例では分割情報の変更を行わず，分割状況レポートを参考に回路の再設計を行い，回路分割を実現した．そして，次のニューラルネットワーク・エミュレータの事例では，設計データ(VHDLソース)には手を加えずに，分割情報の修正により回路を分割した．本環境を利用して，回路の動作検証および実装した回路で動作するアプリケーション・ソフトウェアの開発が円滑に行えることを確認した．なお，ニューラルネットワーク・エミュレータの分割事例では，本実装支援ツールとソフトウェア・シミュレーションを組合わせることで，アプリケーションを処理するために必要とされる最適な回路構成が短時間で決定でき，直ちにプロトタイプの設計・実装が行える環境が構築できた．

表6: 回路分割後の入出力端子数

Component \ Line	Input	Output	Input(B)	Output(B)	In-Out(B)	External	Total
SEQ_LCA	0	7	0	0	0	2	9
BUF0_LCA	58	72	24	0	0	0	154
BUF1_LCA	58	72	24	0	0	0	154
DIO0_LCA	26	24	0	0	0	73	123
DIO1_LCA	26	24	0	0	0	73	123
PIO0_LCA	34	31	0	0	0	0	65
PIO1_LCA	30	87	0	0	0	0	117
PMA0_LCA	41	7	0	24	0	0	72
PMA1_LCA	67	26	0	24	0	0	117
PMB0_LCA	46	3	0	24	0	0	73
PMB1_LCA	50	7	0	24	0	0	81

ラピッドシステムプロトタイピングに関する今後の課題としては，回路構成の決定から設計，回路の分割および実装，動作検証，そしてアプリケーション・ソフトウェア開発までの一貫したシステム開発が，設計者とマルチFPGA実装支援環境間においてより円滑に行えるようグラフィカルユーザインタフェース(GUI)面での充実を図る予定である．

謝辞

日頃御討論頂く，本学マイクロ化総合技術センターの久我守弘講師，本学情報工学研究科の奥村勝氏，井上弘士氏，山中圭氏に感謝いたします．

参考文献

[1] 末吉，大内："システムラピッドプロトタイピングへのFPGA応用例，" 電子情報通信学会総合大会講演論文集，SA-3-3，pp.447-448，Mar. 1995.

[2] 井上，中垣，大内，久我，末吉："教育用RISC型マイクロプロセッサDLX-FPGAとそのラピッドシステムプロトタイピング，" 信学技CPSY95-20，Apr. 1995.

[3] 末吉："計算機アーキテクチャ設計のためのCADツールの在り方，" 情報処理学会 合同研究会 (95-ARC-102, 95-DA-73) パネル討論用メモ，Jan. 1995.

[4] 中垣，井上，久我，末吉："上級コース向き教育用マイクロプロセッサDLX-FPGAの設計と実現，" 信学技法CPSY94-57，1994.

[5] Hennessy, J.L., and Patterson, D.A.,: Computer Architecture : A Quantitative Approach, Morgan Kaufmann Publishers, Inc., 1990.

[6] The Programmable Logic Data Book, Xilinx, Inc. (1994).

[7] Programmable Interconnect Data Book, Aptix Corp. (1993).