パーソナルコンピューティング指向の動的再構成可能PCI カード

市川　周一[†]　　島田　俊夫[‡]

[†] 豊橋技術科学大学知識情報工学系
〒441　豊橋市天伯町雲雀ヶ丘 1
Tel: 0532-44-6897, Fax: 0532-44-6873, E-mail: ichikawa@tutkie.tut.ac.jp
[‡] 名古屋大学工学部電子情報学科
Tel: 052-789-3588, Fax: 052-789-3168, E-mail: shimada@nuee.nagoya-u.ac.jp

概要　Field Programmable Gate Array (FPGA)等の再構成可能デバイスをパーソナル・コンピューティングに適用するには多くの検討課題が残されている．個人用途ではコスト面から再構成可能な論理規模が制約され，汎用バスの性能からデータ転送幅や遅延時間が制限される．これらの制限とハードウェア化による利益を定量的に評価するためPCIバス用再構成可能ボードを試作した．ハードウェアの単純化によりコストを抑え，再構成時間を2 ~ 8 msに短縮して小規模応用にも適用可能とする．大規模応用ではボードを複数使用するか実行中に構成を変更する．133 Mhz Pentiumプロセッサの Unixシステムで，転送幅は実測9 ~ 84 MB/sであった．

1. はじめに

専用計算機 (Custom Computing Machine; CCM) による応用プログラムの高速化は，Lispマシンなどで古くから実践されてきた．CCMによる高速化は，専用回路による処理の高速化，専用データパスによるデータ転送幅の向上，複数の演算器やパイプライン化による並列処理などによって得られたものである [1]．しかしCCMは問題に特化したハードウェアを持つため応用範囲が狭く，汎用プロセッサに比べてコストが高いという問題を持つ．マーケットの大きい汎用プロセッサは世代交代が早く性能向上も著しいため，CCMがコストに見合った性能上の優位性を保ち続けることは難しい．このためCCMは，汎用プロセッサで対応できない一部の限られた応用で使われるにとどまっていた．

ところが近年ではField Programmable Gate Array (FPGA)などのプログラム可能な論理デバイスが進歩し，応用に特化した回路が低いコストで使用できる．また書換可能型のFPGAを使用すれば，複数の応用に合わせて実行時に論理を再構成することが可能である．こうしてCCMのコスト面での問題が緩和されたため，FPGAを用いた再構成 可能CCMの研究が多く行われるようになった．

現在研究されているCCMシステムは，大胆に抽象化すれば以下 2 つの両極端の間に分類される．一つは汎用プロセッサに外部から付加するタイプ，もう一つはプロセッサ内部の再構成可能論理で命令セットを変化させるタイプである．前者は多くのばあい並列構成を特徴とし，大規模計算において高性能を発揮する．代表例としてDEC PeRLe-1 [2], SPLASH 2 [3] などがある．後者は粒度の小さな並列性を利用しようとする試みで，ボトムアップに応用固有の並列性を取り出そうとするものである．代表例にはPRISM [4][5] や DPGA [6], DISC [7] などがあるが，現在のところ広く使われるに至っていない．

再構成可能CCMの適用対象は，その性格上広い．例えば汎用機が直接提供していない処理（bit演算や多倍長演算）では専用回路の優位性を生かせるし，時間的制約の厳しい実時間処理でもハードウェア化が有効である．ハードウェアのプロトタイピングや論理シミュレーションでは，FPGAで対象を直接マッピングするエミュレータが以前から商品化されている．このほか多くの応用で優位性が報告されているが，SPLASH2[3]とPeRLe-1[2]での応用例は以下のとおりである．

- テキスト検索，遺伝子データベース検索，指紋照合，画像処理 [3]

・暗号化，物理系のシミュレーション，信号処理，画像処理 [2]

このようにFPGAによる再構成可能CCMは一定の成果を上げてきた．しかし，従来の研究は並列性が明らかな問題に対して大規模な再構成可能論理を適用し高性能を引き出そうとする『性能指向のアプローチ』であった．パーソナル・コンピューティングのような小規模で不規則な応用に対する再構成可能CCMの適用可能性は充分に検討されてきたとは言えない．本研究の目的は，再構成可能CCMを個人用途に使用するための条件と限界を明らかにすることである．

2. パーソナル・コンピューティングへの適用

2.1. パーソナル・コンピューティングの定義

本研究において『パーソナル・コンピューティング』とは，パーソナル・コンピュータで処理可能な規模の計算をさすこととする．パーソナル・コンピュータの性能は年々向上しているが，それに応じて基本ソフトウェアも応用プログラムも年々大きくなり，パーソナル・システムで扱う問題の規模も大きくなる．再構成可能CCMの必要性が減ずることはない．

汎用マイクロプロセッサ(MPU)の進歩は著しいが，FPGAなどの再構成可能論理もマイクロプロセッサと同じ半導体プロセスの進歩によって集積度と速度が向上する．ある一時期のテクノロジに基づいた評価であるとしても，各要素技術の寄与が定量的に評価されていれば再構成可能CCMの適用可能範囲を明らかにできるはずである．

2.2. パーソナル再構成可能システム

再構成可能システムに必要な要素は，SPLASH2の総括 [8] によれば以下の 3 つである．

(1) 高転送幅入出力
(2) 大容量高転送幅ローカル・メモリ
(3) プログラミング支援

いずれもHigh Performance Computingの分野では自明な要求であるが，これに加えてパーソナル・システムではコスト面から強い制限が加わる．

パーソナル・コンピュータは量産部品で構成することが前提なので，プロセッサやマザーボードを専用化することは考えにくい．汎用バスに再構成可能論理ボードを付加する構成が唯一現実的な解となる．ローカル・メモリは大きいほど良いが，再構成可能論理に大量の高速RAMを付加することは考えにくい．同じコストで主記憶や 2 次キャッシュを増設する方がシステム全体を考えたとき有効かもしれない．再構成可能論理の規模も同じ理由で制約される．性能が最優先される専用機やスーパーコンピュータとは事情が異なる．

以上の点を考慮して，本システムでは以下のような設計方針をとる．

2.2.1. 構成の単純化

ボードに搭載するFPGAはコスト低減のため1個とする．1チップでも数万ゲートの論理が利用可能なので実応用に適用することができる．複数のFPGAを搭載してもチップ間の遅延や転送速度が問題となり動作クロックが低下する可能性がある．複数チップへの設計分割も問題になるが，現行のCADでは自動分割機能を提供していない．

2.2.2. オーバーレイの積極的利用

上述のようにボード上の再構成可能論理規模が制限されているので，実行中に動的再構成を行って設計を書き換える．ソフトウェアの実行フェーズや機能分割という概念を積極的に利用する．必然的に，再構成時間の短縮も検討課題になる．

2.2.3. ボード単位の増設

オーバーレイで性能が不足し複数の再構成可能論理ユニットを必要とする用途では，ボード単位で増設を行う．

2.3. 既存の付加システムとの相違点

再構成可能CCMはプロトタイピングに魅力的な技術であるから，ハードウェア / ソフトウェア協調設計のテストベッドとして幾つかの実現が存在する．[9][10][11]いずれも再構成可能な論理ボードを汎用プロセッサの外部バスに付加し，ソフトウェアの一部をハードウェア化して実行時間の短縮を図ろうとする研究である．

これらのシステムは本システムと類似の構

成を取るが，目的が異なるため設計方針に相違点があり，結果として本研究の目的には適さない．第一の相違点はFPGAが複数搭載され大規模な構成を取ることである．協調設計の実現と評価を行うためには可能な限りハードウェアの制限を減らす必要があることと，並列動作や大規模論理の利用をねらったと考えられる．この結果，付加ボードは大規模かつ複雑になってパーソナル・システムにはコスト面から適用不能になっている．また実行時の動的再構成が考慮されておらず，再構成時間の短縮が図られていないため，オーバーレイのような実行形態は事実上不可能と思われる．

実行中の再構成を積極的に利用している研究としては文献 [12] などがある．FPGAを利用した画像処理プロセッサにおいてPROMに複数の構成データを書き込んでおき，実行時に切り替えながら使って好成績を得ている．本研究の目的の一つは，このような動的再構成を広範囲に適用することである．

動的再構成の利用という意味では，データフローグラフを再構成可能論理の構成データに変換しハードウェアで実行するWASMII [13][14] もあげられる．しかしWASMIIはMPLDというハードウェア構造を前提にデータ駆動による並列処理を行っており，本研究とは目的と適用対象が異なっている．

2.4. パーソナル再構成システムの課題

小規模な再構成システムでは，再構成可能な論理規模が小さいためハードウェア化による利益も限定される．汎用バス経由のデータ転送では，転送速度が制限され遅延時間も大きい．これに対して，ホスト側の汎用MPUは最先端の半導体プロセスを利用して高い周波数で動作し，データの転送幅も広い．定量的評価なしでは，付加プロセッサによる高速化が達成できるか否か自明でない．

汎用MPUに不向きな処理，例えば画像処理や信号処理では再構成可能論理の優位性が発揮できると期待されるが，市場で需要が大きい用途には専用LSIが開発される．再構成可能論理の適用対象は汎用MPUと専用LSIの間のニッチ市場に制限される可能性がある．しかし再構成可能論理には動的適応能力やオーバーレイによる仮想化など独自の能力がある．これらの特徴を生かしてニッチにとどまらない新しい応用を探すべきである．本研究で動的再構成を重視する理由はここにある．

3. 設計と実装

3.1. システム構成

評価システムの仕様は以下の通りである．

プロセッサ	Intel Pentium 133Mhz
マザーボード	Intel Advanced/EV(82430FX)
2次キャッシュ	Pipeline Burst SRAM 256KB
主記憶	Fast Page Mode DRAM 64MB (60ns)
基本ソフトウェア	FreeBSD 2.1.5R

1996年現在でパーソナル・コンピュータとして標準的な仕様を設定した．基本ソフトウェアにはソース・プログラムが提供されているという理由でBSD系unixを選んだ．

このシステムのPCIバスに再構成可能ボードを付加して評価を行っている．PCIバスを選択した理由は，パーソナル・コンピュータからワークステーションまで現在最も標準的に使用されているバスで，近い将来も引き続き使用されると考えられるからである．バスクロックには33 Mhzを供給し，32bit幅で（理論上）最大133MB/sの転送速度を持つ．

3.2. 再構成可能論理ボードの構成

図 1 に，再構成可能論理ボードの構成を示す．

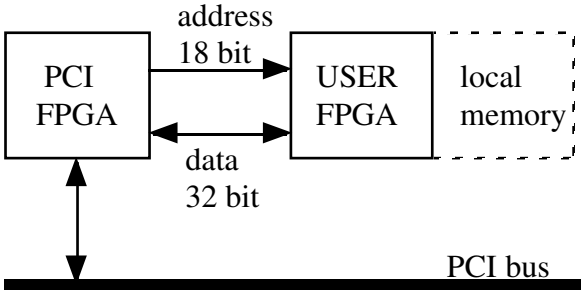

図 1 ：OPERLボードのブロック図

基本的に 2 つのFPGAだけで構成されており，たいへん単純な構成となっている．PCI FPGAはPCIバスへのインタフェース機能を持ち，構成は電源投入時にSROM (serialROM)で設定する．動的再構成の対象となるのはUSER FPGAだけで，プログラムはPCIバス(PCI FPGA)経由

で行う．オプションとしてボード上にローカル・メモリを付加することができる．

FPGAにはルーセント・テクノロジー社のORCA OR2Cシリーズを採用した．パッケージは208pin SQFPで，ボード上への実装にはソケットを使っているためFPGAの容量を任意に変更することができる（4Kゲート～40 Kゲート）．ボードの構成部品点数が非常に少ないので，量産されればコストはかなり安くなると期待される．このボードをOPERLボードとよぶ (Orca PCI Easily Reconfigurable Logic または Orca PErsonal Reconfigurable Logic)．OPERLボード の外観を図 2 に示す．

OPERLボードはPCIのバス・クロックに同期して動作する．本評価システムではPCIに33 Mhzを供給しているので，ボード上の回路を30 nsサイクルでパイプライン化して動作させる．ローカル・メモリには（当面）サイクル・タイム 12 nsのSRAMを搭載し，メモリも含めて 30 nsサイクルで動作させる．

FPGAの再構成時間は容量にほぼ比例して長くなるため，大容量品を実装すると再構成オーバーヘッドが大きくなる．また配置配線等の時間も容量に応じて長くなるため開発コストが増加する．適用対象に応じて適切な容量を選ぶ必要があるが，このようなトレードオフについても今後検討していきたい．

3.3. ユーザ・インタフェース

OPERLボードは通常のPCIバス・デバイ スと同様に，システムによってI/O空間とメモリ空間にマップされる．現在のところI/Oウインドウを256バイト，メモリ・ウインドウを1 MBとしているが，これはPCI FPGAの構成次第で変更可能である．

現状ではPCI FPGAの機能はターゲット・デバイスとしての機能に限定されているため，OPERLからバス上にトランザクションを要求することはできない．全てホスト側から読み書きする形でアクセスを行う．マスタ・デバイスとしての動作を許すと，ユーザが作成した回路から異常なトランザクションが発生した場合にOSがクラッシュする可能性がある．PCI FPGAの構成データを変更すれば，OPERLをPCIマスタ・デバイスにすることは可能であるが，基本ソフトウェアとの整合性を充分検討してから行う必要がある．

OPERLは基本ソフトウェアからは周辺機器の一種として見える．Unixでいうrawdevice（文字型でバッファされないデバイス）であって，デバイス・ドライバによってopen, read,write, mmap,close,ioctl等の機能が提供される．ユーザはsystemcallを使用してOS経由でOPERLにアクセスする．

この他，ユーザ・プログラムから直接入出力命令(in, out)を発行することが可能である．Pentiumのアーキテクチャ[15] では，I/Oポートのアクセスを I/O特権レベルとbitmapで保護しているため，OSでプロセス状態の該当部分を設定すればユーザ・プロセスから入出力命令を発行可能である．本システムではOPERLデバイスをopenするとI/O特権が設定され入出力が可能になる．性能測定やデバイス・ドライバ開発のために付加した機能であるが，低

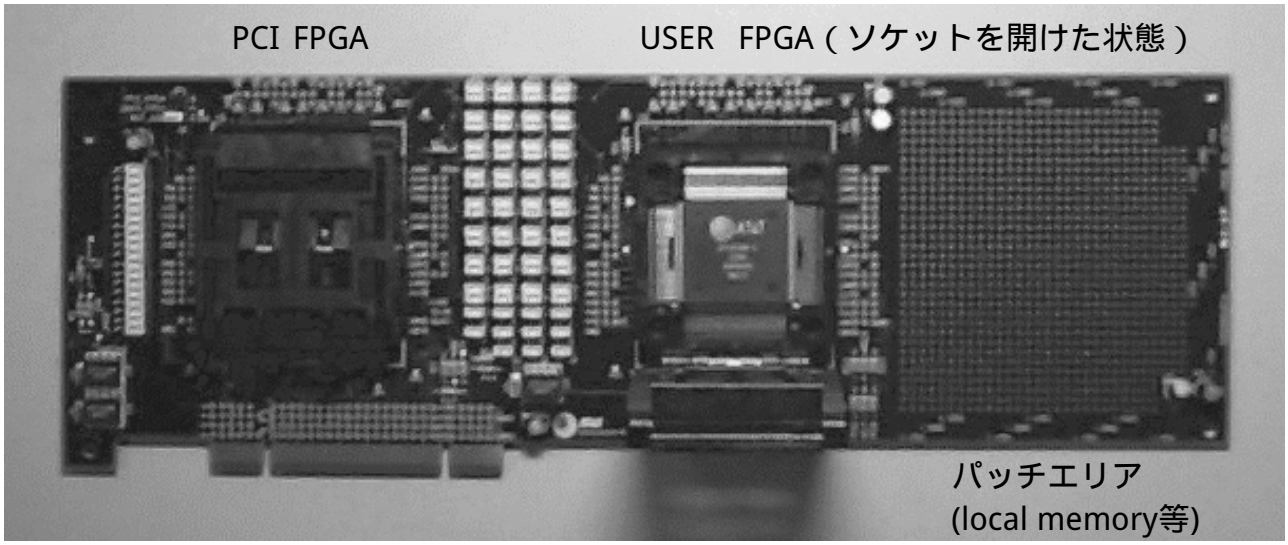

図 2 ：OPERLボード

オーバーヘッドの入出力にも使用できる．

3.4. 設計手法

USER FPGAの設計は以下の方法で行っている．VHDLで記述を行い，論理合成ツール(synopsysFPGA compiler)を用いてedifネットリストを生成する．Cなどのプログラム言語から設計を自動生成することは非常に重要だが，設計自動化の観点から既に多くの研究が行われているため当面本研究では考えない．エンドユーザ向けパーソナル・システムという観点から考えても，設計手法よりパッケージやライブラリの整備が重要になると考えられる．edifネットリストはLucentのツールでP&Rされ，出力された構成データ(bit stream)をonlineでUSER FPGAにダウンロードする．

本システムでは再構成可能論理を1 chipに限定したため，設計の分割など難しい問題はない．遅延の検証もチップ内で完結するため，既存のツールで問題なく設計できる．現在のFPGAでは，単独チップで使用する限り33 Mhz 程度で動作する設計は難しくない．

4. 基礎的な評価結果

4.1. 現状と測定環境

OPERLシステムは実機で稼働しており，実応用での評価を進めている．本節ではシステムの基礎的性能評価結果を示すが，それらの測定はmulti-usermulti-processでネットワークに接続するという使用条件下で行われたものである．PCIバスにはOPERLの他にSCSIコントローラとビデオボードが接続されX window等が動作している．OPERLがPCIを独占しているわけではない．

4.2. 実行モデル

本システムでは，ホスト計算機からOPERLにデータを転送し計算結果を回収するというモデルで計算が行われる．従って応用の実行時間を見積もるには，最低限以下のパラメータを事前に評価する必要がある．

(1) データの転送遅延と転送レート
(2) 論理の構成時間

4.3. 転送速度と遅延時間

データ転送には既に述べたように 3 つの方法が用意されている．それぞれ特徴があり性能も異なるので，応用に適した方法を選ぶ必要がある．

4.3.1. User I/O

所定のファイルをincludeしてinline関数を呼び出せば，コンパイラが入出力命令を挿入する．システムが介在しないためオーバーヘッドは小さいが，I/O命令ではバースト転送が行えないため，一度に転送できるのは4 byteだけである．レジスタの読み書き等に適している．

	遅延時間	換算転送レート
入力 inl	0.42μsec	(9.0MB/s)
出力 outl	0.29μsec	(13.MB/s)

上の測定結果は，PCIバス上のトランザクションに要した時間をそのまま示していると考えられる．出力のほうが速いのは，入力が「アドレスを送ってデータを受け取る」のに対して，出力は「アドレスとデータを送る」という一方通行で終了するからである．

4.3.2. mmap

mmapは，物理アドレス上に存在するOPERLのメモリ・ウインドウをユーザのメモリ空間にマップする機能である．マップされたあとは，ユーザ・モードで通常の配列のようにアクセスできる．ただしコンパイラの最適化との関係でプログラム上の注意を要する場合もある．

	アドレス	遅延時間	換算転送レート
読み出し	連続	0.36μsec	(11.MB/s)
	ランダム	0.39μsec	(9.7MB/s)
書き込み	連続	0.046μsec	(84.MB/s)
	ランダム	0.15μsec	(26.MB/s)

ここで書き込みと読み出しの転送レートが大きく異なるのは，書き込み時にバースト転送が行われるためと考えられる．本システムのPCI-Hostブリッジ(82437FX) はwrite bufferを持っており，連続アドレスへの書き込みをバースト転送に変換する．読み出しの転送レートがinl命令とほぼ等しいことからも，読み出

し時にはバースト転送が行われていないことが推測できる．

4.3.3. read/write

readはデバイスからメモリへの転送処理，writeはメモリからデバイスへの転送処理である．実際には，物理空間にマップされているOPERLのメモリ・ウインドウとユーザのメモリ空間との間のコピー操作になる．OPERLは現在ターゲット専用デバイスなので，ホスト側で逐一データを転送する必要がある (いわゆるPIO)．writeは連続アドレスへの書き込みなので，ここでもバースト転送により転送レートが向上している．

表に示した遅延時間は，システム・コールの呼び出し毎に生じるオーバーヘッド時間である．read/writeの場合，データ転送でシステム・コールを行う度に表中の遅延時間が加算される．

	転送レート	遅延時間
read	9.7MB/s	3.5μsec
write	42.MB/s	5.5μsec

4.4. 動的再構成に関する検討

4.4.1. 再構成方法

USER FPGAの動的再構成はPCI FPGAの制御で行うので，構成データはPCI FPGAに対して送らねばならない．PCIバス規格書にはI/Oアクセス，メモリ・アクセスの他にコンフィグ・アクセスという方法が定義されていて，これがPCIインタフェース (OPERLではPCI FPGA) に対するアクセス方法として規定されている．コンフィグ・レジスタはPCI-Hostブリッジによって I/O空間にマップされるので，結局入出力命令を使うことになる．本研究でも，コンフィグ・レジスタに対してI/O命令を発行することで動的再構成を行った．

4.4.2. 再構成時間

応用の実行中に小規模なFPGAを何度も書き換えて使用するためには，再構成時間の短縮が必須条件である．再構成時間を制限する要素は大きく分けて以下の 2 つである．どちらか遅いほうに制限される．

(1) デバイスの構成データ入力幅とサイクル時間
OR2Cシリーズでは，スレーブ・パラレル・モードで 8 bit幅×10 Mhz = 10 MB/s の転送速度をサポートしている[16] ．
(2) 構成データを転送するパスの転送レート
OPERLへの書き込みレートは前節で示したように11 ~ 42MB/sであるから，今回の実装でPCI経由のデータ転送がネックになることはない．

実際にOPERLボードの再構成時間を測定した結果は以下の通りであった．測定対象は 3 品種で，2C08 (8Kゲート相当), 2C15 (15Kゲート相当), 2C40 (40Kゲート相当) である．

品種	usr(ms)	sys(ms)
2C08	1.9	0.4
2C15	3.6	0.8
2C40	7.7	1.9

表中，ユーザ時間(usr)はメモリ上に存在する構成データをOPERLボードに転送する時間である．構成データがファイルシステムに存在する場合はread等のシステムコールでメモリ上に読み込む必要があるが，システム時間(sys)はこれに要する時間の実測値である．ただし測定時には同じデータを繰り返し読んでいるため，OSによるキャッシュが働いてバッファ・キャッシュからの転送だけが行われていると考えられる (ディスクのアクセス時間が含まれないベストケース) ．

表からわかるとおり2 ~ 8 msという短時間で再構成できるため，OPERLボードは小規模な応用プログラムの実行にも適用可能である．

4.4.3. 構成データ転送に関する問題

現在使用中のFPGA (OR2C) では，構成データを汎用バス(PCI)上で転送してもネックにならなかった．しかし新型のFPGAではPCIバス上の転送レートが問題になる可能性がある．例えば Xilinx の XC6200 シリーズでは再構成時間を1 ms以下にできると予想されるが，このときの構成データ転送レートは 100MB/s に達する [17] . このレートは PCI のピーク転送速度に近く，PCIバス上の転送がネックになりうる．

4.4.4. 構成データの圧縮効果

構成データの転送時間を短縮するために，構成データが圧縮できないか検討した．ホス

ト側で構成データを圧縮しOPERL側で伸長すれば，圧縮率次第で転送のボトルネックを緩和または解消できる．

FPGAの内部は規則的な構造を持つため構成データにも規則性が存在する．またFPGAのゲートや配線の多くが使われずに残るため，「未使用」という状態が多く表現されているはずである．実際にいくつかの構成データで調査した結果，構成データに含まれている0と1の比率には大きな片寄りがあった．

そこでいくつかの構成データを例にとって圧縮方法を検討してみた．取り上げた構成データは以下の5つである．

(1) pci: PCI FPGAのnetlistをOR2C15(15 K gate相当) で実現したもの．
(2) pci8: pciと同じnetlistをOR2C08(8 K gate相当) で実現したもの．
(3) pci6: pciと同じnetlistをOR2C06(6 K gate相当) で実現したもの．
(4) accum:OR2C15に32 bitのaccumulatorを1つだけ置いたもの．
(5) acc-c: accumのレイアウトをベンダが提供する圧縮形式[16]で表現したもの．同じ内容のフレームが連続している場合，データ部分の転送が省略される．

pci, pci8, pci6からは，同じ複雑さの設計で論理の使用率が上がった場合の影響を読み取ることができる．accumとacc-cは，レイアウトまで全く同じで構成データの形式だけが異なるものである．各々の設計のPFU使用率（再構成可能論理の使用率）と，構成データの全ビット中に含まれる0の比率を図3に示す．疎な設計から密な設計まで含まれており，いずれも共通に0の比率が高いことがわかる．

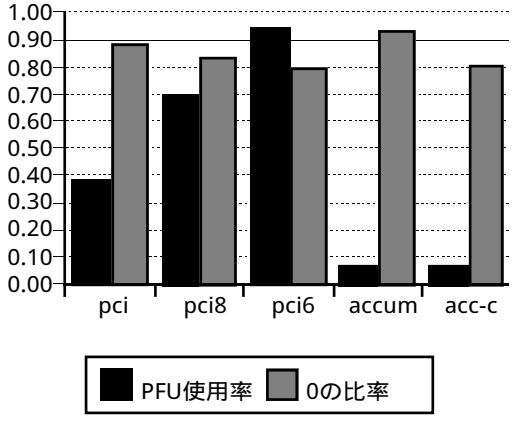

図3．構成データの特徴

acc-cのサイズはaccumの39%であった．これはaccumが非常に使用率の低い構成であるため，空のフレームが連続するからと考えられる．使用されているフレームの内容が前後で偶然一致することは確率上ほとんどありえないので，使用率の低い設計以外ではこの圧縮手法は効果がない．たとえばpciでは全てのフレームが異なるため全く圧縮されなかった．他の一般的な圧縮方法を考える必要がある．

以上5つの構成データを幾つかの手法で圧縮し効果を調べた．OPERL側のハードウェアで実時間伸長することが前提であるから，圧縮率が良くても複雑な方法を採用することはできない．アドホックな手法としてrun-length圧縮などを試みたが，FPGAの構成データでは同じデータが連続する可能性が低いため良い結果を得ることはできなかった．

理論的かつ定量的な比較を行うため，入力モデルとして無記憶情報源を仮定し8 bit単位の静的FV符号化を行うことにする．入力の統計から，0および2のベキ(8 bit中1 bitだけが1)を圧縮すると効果が上がることがわかった．そこで統計をもとにハードウェア化が容易な実現例として以下の符号化手法を評価した．

(1) 00000000 → 0
(2) 2のベキ → 10xxx
　　xxx = 1のビット位置 (0 ~ 7)
(3) それ以外 → 11yyyyyyyy
　　yyyyyyyy = 8 bitリテラル

圧縮前の構成データ長を1とし，情報理論上の圧縮限界(ideal)，ハフマン符号化(huff)，提案手法(power)の評価結果を図4に示す．

アドホック圧縮法の例としてrun-length法の結果もあわせて示した．零以外が連続することは統計上ほとんどないので，零以外をrun-length圧縮すると結果は悪化する．したがって図4では，8 bit単位で零だけをrun-length圧縮する手法 (rl-zero)の評価結果を示す．

図4より，小規模なハードウェアで実現可能な圧縮法(power)でも，情報理論上の限界(ideal)やハフマン符号化(huff)に近い圧縮率が得られ，run-length圧縮よりも高い圧縮率を示すことがわかる．

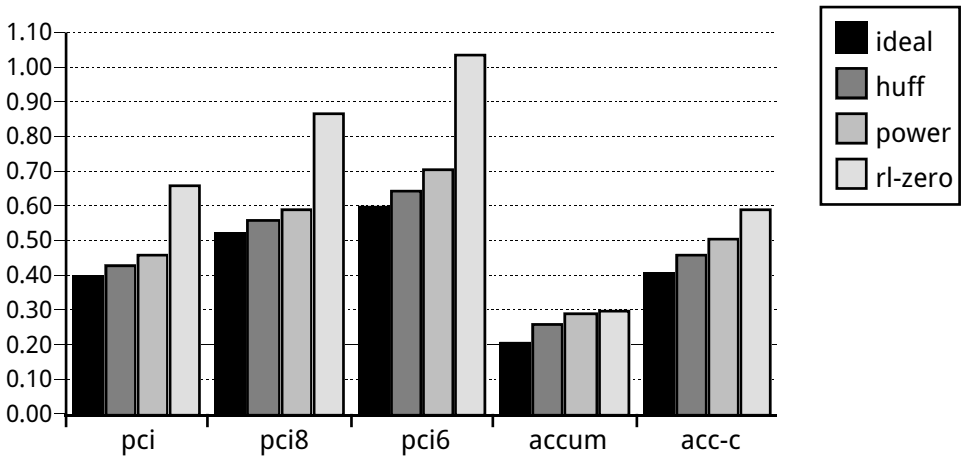

図４．構成データの圧縮率

5. おわりに

システムを構成する要素技術の評価がほぼ終了したので，今後は実応用での評価をすすめる．また，実応用からのフィードバックをふまえて要素性能向上のための改良を続ける必要がある．現在，TSPなどの最適化問題，符号化／暗号化を含む数論上の問題，オペレーティング・システムの部品や通信プロトコルのハードウェア化，ライセンスや課金手法などの応用について検討を進めている．

本研究の開始時にはパーソナル指向の再構成可能付加ボードが入手できなかったため，評価用プラットホームとしてOPERLボードを自作する必要があった．しかし昨今では米国のベンチャー企業などからOPERLボードと似た用途の付加ボードが入手可能になっている[18][19]．今後の研究では，OPERLと共に市販のボードを上手に利用して実応用での評価を進める．それと同時に，機能の固定された市販品ではできない研究（例えば構成データの圧縮転送や構成データの符号化に関する研究等）を行う道具として積極的にOPERLを利用していく予定である．

類似の市販品の登場は，OPERLの設計目標が妥当であったことの傍証であって喜ばしいことであるとも言える．

謝辞

本研究の一部は，財団法人　堀情報科学振興財団の研究助成（研究課題『Field Programmable Gate Arrayを用いたLSI試作と設計教育』）によるものである．

文献

[1] 杉本大一郎（編）：“専用計算機によるシミュレーション”，朝倉書店，東京　(1994).
[2] J. E. Vuillemin et al.: "Programmable Active Memories Reconfigurable Systems Come of Age", IEEE Trans. on VLSI systems, Vol. 4, no. 1, pp. 56-69 (1996).
[3] D. A. Buell, J. M. Arnold, and W. J. Kleinfelder Ed.: "SPLASH 2: FPGAs in a Custom Computing Machine", IEEE Computer Society Press, Los Alamitos, CA (1996).
[4] P. M. Athanas, H. F. Silverman: "Processor Reconfiguration Through Instruction-Set Metamorphosis", IEEE Computer, vol. 26, no. 3, pp. 11-18 (1993).
[5] M. Wazlowski et al.: "PRISM-II Compiler and Architecture" Proc. IEEE workshop on FPGAs for Custom Computing Machines, pp. 9-16 (1993).
[6] A. DeHon: "DPGA-Coupled Microprocessors: Commodity ICs for the Early 21st Century", Proc. IEEE workshop on FPGAs for Custom Computing Machines, pp. 31-39, Napa, CA. (1994).
[7] M. J. Wirthlin, B. L. Hutchings: "A Dynamic Instruction Set Computer", Proc. IEEE symp. on FPGAs for Custom Computing Machines, pp. 99-107, Napa, CA. (1995).

[8] D. A. Buell and J. M. Arnold: "The Promise and the Problems" In "SPLASH 2: FPGAs in a Custom Computing Machine", D. A. Buell, J. M. Arnold, and W. J. Kleinfelder Ed., IEEE Computer Society Press, Los Alamitos, CA, pp. 166-178 (1996).
[9] D. E. Thomas, J. K. Adams, H. Schmit: "A Model and Methodology for Hardware-Software Codesign", IEEE Design and Test of Computers, Vol. 10, no. 3, pp. 6-15 (1993).
[10] 伊藤康史，平尾誠，木村晋二，渡辺勝正："汎用コプロセッサGPCP-SSの実現と評価"，情報処理学会研究報告 95-DA-77, pp. 169-176 (1995).
[11] 斎藤正伸，多田好克："計算資源としてのFPGAとソフトウェアのインタフェースに関する考察"，第 7 回パルテノン研究会予稿集，pp. 28-36 (1995).
[12] B. Schoner, C. Jones, J. Villasenor: "Issues in Wireless Video Coding using Run-time-reconfigurable FPGAs", Proc. IEEE symp. on FPGAs for Custom Computing Machines, pp. 85-89, Napa, CA. (1995).
[13] 凌暁萍，天野英晴："データ駆動型制御機構付きMPLDを用いた並列処理マシンWASMIIの仮想化"，情報処理学会論文誌，vol. 35, no. 4, pp. 646-657 (1994).
[14] 凌暁萍，天野英晴："WASMII: データ駆動型制御機構をもつMPLD"，電子情報通信学会論文誌 D-I, Vol. J77-J-I, no. 4, pp. 309-317 (1994).
[15] Intel Corp.: "Pentium Processor Family Developer's Manual", Intel (1995).
[16] Lucent Technologies Inc.: "FPGA Data Book", Tokyo (1996).
[17] Xilinx, Inc.: "The Programmable Logic Data Book", San Jose, CA. (1996).
[18] Annapolis Micro Systems, Inc., http://www.annapmicro.com/
[19] Virtual Computer Corp., http://www.vcc.com