再構成可能な同期式データフロー計算機に関する一検討

佐々木 浩志† 槻岡 秀朗‡ 庄司 修芳‡ 小林 広明† 中村 維男†

†東北大学大学院情報科学研究科 ‡株式会社フレンドリーシステムズ

†〒980-8579 仙台市青葉区荒巻字青葉01 アーキテクチャ学講座

‡〒980-6108 仙台市青葉区中央アエル8F-810

†022-217-7013 ‡022-261-3693

†{sasa, koba, tadao}@archi.is.tohoku.ac.jp ‡{tsuki, shuho}@friendly-systems.co.jp

あらまし

本報告では、データフローグラフ表現されたアプリケーションに対応してハードウェアを再構成し、各々の計算要素を同期させて計算を行う、同期式データフロー計算機を提案する。アプリケーションとしてJPEGエンコーダをデータフローグラフに実装して、必要となるハードウェア資源を特定した。その結果、メモリアクセス専用の処理要素（ユニット）を取り入れることで、処理を自然な形でデータフローに表現できることが分かった。また、同期式データフロー計算機のアプリケーションを作成するためのソフトウェア開発環境について述べ、提案する計算機によって性能向上の期待できるアプリケーションの持つ性質について考察する。

キーワード 再構成、同期、データフロー、計算機

A Study on a Reconfigurable Synchronous Dataflow Computer

Hiroshi Sasaki†, Hideaki Tsukioka‡, Nobuyoshi Shoji‡, Hiroaki Kobayashi†, Tadao Nakamura†

†Computer Architecture Laboratory, 01 Aramaki Aza Aoba Aobaku, Sendai, 980-8579

‡AER 8F-810, Chuo, Aoba-ku, Sendai, 980-6108

†022-217-7013 ‡022-261-3693

†{sasa, koba, tadao}@archi.is.tohoku.ac.jp ‡{tsuki, shuho}@friendly-systems.co.jp

Abstract

This report proposes a synchronous dataflow computer, which constructs hardware to represent dataflow graphs of applications then processes data in the dataflow fashion. We implemented JPEG encoder on the system and measured the amount of required hardware resources. The experimental results show that computations can naturally be expressed in dataflow graphs using units only for accessing the shared memory. The exploitable features of applications are discussed and a software development environment is also presented.

key words Reconfigurable, Synchronous, Dataflow, Computer

1. はじめに

FPGA 等の再構成可能なハードウェアの発展により、ハードウェアとソフトウェアは互いの領域を侵食し、融合しようとしている[1]。これらの素子技術を用いた再構成可能なコンピュータは、アプリケーションに対応してハードウェアを再構成し、計算を行う。したがって回路を構成するために時間を要するが、アプリケーションの並列性を充分に利用できる可能性がある。汎用プロセッサにおいてソフトウェアとハードウェアのセマンティックギャップのためにアプリケーションの並列性を利用できない問題を、再構成可能なコンピュータによって解決することができれば、大幅な性能向上を行えると期待されている。

再構成可能なコンピュータにおいて、アプリケーションプログラマが意識する処理要素の粒度は、さまざまな選択肢がある[2]。粒度が細かいほどプログラマが利用できる並列性は増加するが、プログラマの負担が増加するだけでなく、設定に必用な情報量が増加してしまう。

本報告では、データフローグラフで表されるアプリケーションをそのままハードウェア上に展開し、ハードウェアの設定後は命令の供給を受けることなく連続的にデータを処理する、同期式データフロー計算機を提案する。2 節ではハードウェア構成について、基本概念と実装の観点から述べる。3 節ではソフトウェア開発環境について述べる。4 節ではアプリケーションの一例として、ベースラインシステムの JPEG デコーダをデータフローグラフで実装する。5 節では実装の結果から、処理を自然な形でデータフローグラフに表現する方法と、同期式データフロー計算機が高性能を達成しうるアプリケーションの性質について考察する。6 節はまとめである。

2. ハードウェア構成

2.1. 基本概念

本報告で提案する同期式データフロー計算機は、ユニットと呼ばれる計算要素と、ユニット間の相互結合ネットワークから構成される。ユニット間を適切に結合するようにネットワークを設定し、結合された各ユニットの機能を設定することによって、アプリケーションの同期式データフロー計算機への割り当てが行われる。各ユニットが必要な機能をすべて備えていれば、機能設定の柔軟性は高い。しかし互いに共通する部分の少ない機能を 1 つのユニットに詰め込んで実装すると、ユニット内のハードウェア利用効率が低下してしまう[3]。このため演算用途ごとにいくつかの種類に分けてユニットを実装するアプローチを採用した。

また、ユニット間を結合するネットワークを構成する通信路のバンド幅が大きければ、より少ないクロックサイクル数でデータの受け渡しを行えるが、大きなハードウェア資源が必要になる。さらに、すべてのユニットが完全結合をしていればネットワークの柔軟性は最も高いが、この場合も大きな資源が必用になる。ハードウェア資源の制約を考慮すると、通信路のバンド幅とネットワークの柔軟性はトレードオフの関係にある。本報告で提案する同期式データフロー計算機では、ネットワークの柔軟性を重視する設計方針を採用する。これによって、アプリケーションをハードウェアにマッピングする際のネットワークの制約を緩和する。

2.2. 実装

本報告で提案する同期式データフロー計算機は複数のユニットから構成される。各演算ユニットは 1 本の出力線と、機能に応じて 0 から 3 のいずれかの本数の入力線を持つ。計算機と外部との入出力を行う入出力ユニットは、入力線と出力線をそれぞれ 1 本ずつ持つ。

同期式データフロー計算機のシステム構成を、図 1 に示す。ユニット間の通信は相互結合ネットワークを用いて行われる。各ユニットの機能とネットワークの設定はコネクションレジスタと呼ばれる制御部によって管理される。ユニットの機能は、入出力、メモリアクセス、演算の 3 種類に

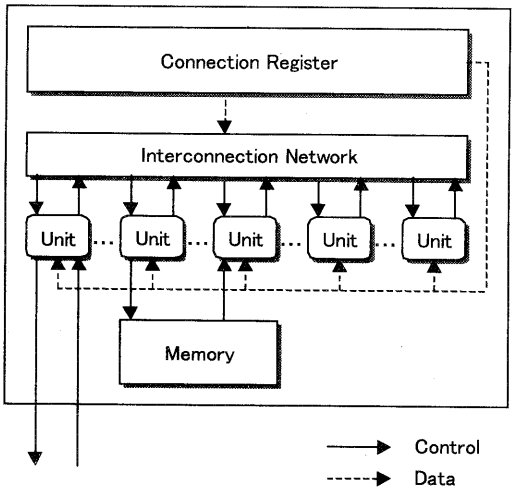

図 1. システム構成図

Fig.1 System Diagram

分類できる。ユニット数が大きくなると、ネットワークの実装に必要なハードウェア量が急激に増加してしまうため、実装では階層化したネットワーク構成などによって、ハードウェアコストを削減する。

ユニットが処理するオペランドの大きさは、16 ビットとする。データは LSB (Least Significant Bit)から 1 ビットずつ出力され、ネットワークを経由して次のユニットに渡される。データの有効性を示すバリッドビットと、データがオーバーフローしているかどうかを示すオーバーフロービットがデータに付加されるため、受け渡しには計 18 クロックサイクルかかることになり、これが 1 演算周期となる。ユニット間のデータの受け渡しを図 2 に示す。

通信路のバンド幅を 1 ビットとして実装することにより、ネットワークに必用な資源量を削減できる。加えて、各ユニットで行われる処理を時間軸方向に展開してパイプライン処理することにより、ユニットの実装に必用なハードウェアコストも削減できる。これにより演算の際に必要となるクロックサイクル数は増加してしまうが、多数のユニットと柔軟性の高いネットワークを実現することができる。

各種類のユニットの種類と、その機能は以下に示すとおりである。

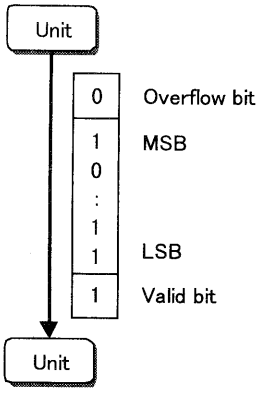

図 2. ユニット間のデータの受け渡し

Fig.2 Data transfer between units

- 算術論理演算ユニット(ALU)
  加算(ADD)、減算(SUB)、同値比較(EQU)、大小比較(EGT)、論理積(AND)、論理和(OR)、排他的論理和(XOR)、否定(NOT)を行う。
- 乗算ユニット
  乗算(MUL)のみを行う。
- シフト演算ユニット
  算術右シフト(SRA)、算術左シフト(SLA)、論理右シフト(SRL)、周期遅延(DLY)を行う。
- フロー制御ユニット
  選択(SEL)、統合(CMB)を行う。「選択」は、2 入力のうちいずれかを出力するかを 3 つ目の入力によって決定する演算である。「統合」は 2 入力のうち、優先される入力のバリッドフラグが真なら優先される入力を出力し、偽なら 2 つ目の入力を出力する演算である。
- 定数生成ユニット(CGU)
  設定された定数(CNS)を出力しつづける。
- メモリ書き込みユニット(MWU)とメモリ読み出しユニット(MRU)
  MWU は、内部メモリのアドレスと値を入力として与えると、指定したアドレスに値を書き込む。MRU は内部メモリのアドレスを入力として与えると、次の周期で値を出力する。メモリそのものをユニットとするのではなく、メモリへのアクセス手段をユニットの機能として提供する。
- 入出力ユニット(IOU)
  外部とのデータのやり取りを行う。

ユニット間のネットワークが完全結合なら最も自由度が高いが、この場合はネットワーク部分のハードウェアコストが高くなってしまう。このため、ユニット間を接続する相互結合ネットワークを実装する際には階層構造をもたせて、各階層内のネットワークを完全結合とすることにより、ハードウェアコストを削減する。 そこで、ユニットの集合がブロックを形成し、ブロックの集合がマクロブロックを形成し、マクロブロックの集合が、同期式データフロー計算機を形成するという階層構造を選択した。この階層構造を図3に示す。

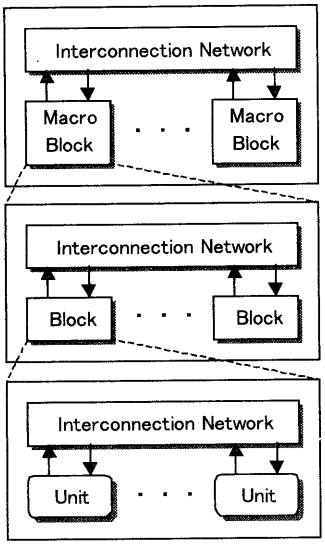

図3. 同期式データフロー計算機の階層構造

Fig.3 Hierarchical structure of a reconfigurable synchronous dataflow computer

現段階でのハードウェア実装では、ブロックはFPGA1個（ALTERA社FLEX10K-3）に対応する。ブロック内には、ALU、MLU、SFU、FCU、CGUがそれぞれ8個ずつ、CGUが16個存在する。マクロブロックはボード1枚に対応する。現在の実装では、IOU、MRU、MWUはマクロブロックに存在し、個数はマクロブロック毎にそれぞれ8個、16個、16個である。実装ではマクロブロックは4個のブロックから構成されている。同期式データフロー計算機は1つ以上のマクロブロックから構成され、ボード（マクロブロック）を増設することでハードウェア資源を増加できる。

3. ソフトウェア開発環境

同期式データフロー計算機のアプリケーションの開発環境は、以下のツールから構成される。

- データフローの言語表記と可視化ツール

  データフローグラフを表記するため、広く用いられているハードウェア記述言語であるVHDLのサブセットを選び、データフロー記述用の言語とした。この言語でデータフローグラフを記述したものをデータフローリストと呼ぶことにする。図4に0の入力回数を数えるデータフローリストの記述例を示す。

```
entity CountZero is
port(i0 : in int;
     o0 : out int);
end CountZero;
architecture dataflowlist of CountZero is
  signal s0, s1, s2, s3 : int;
begin
  cgu0: cns port map(0, s0);
  alu0: equ port map(i0, s0, s1);
  cgu1: cns port map(1, s2);
  fcu0: sel port map(s0, o0, s1, s3);
  alu1: add port map(s2, s3, o0);
end dataflowlist
```

図4. データフローリストの記述例

Fig.4 An example of a dataflow list

データフローグラフには各ユニットの機能とユニット間の結合関係のみを記述する。作成したツールはデータフローリストを読み、データフローグラフを表示する。

- シミュレータ

  データフローグラフをハードウェアに設定した場合のハードウェアの挙動をシミュレートするシミュレータを作成した。シミュレータ

は次の3つのファイルを入力として受け取る。

- ハードウェア設定ファイル
  ユニット間のネットワーク構造、および各ユニットの種類を記述する。
- 実行コードファイル
  ネットワーク構造を設定し、各ユニットの機能を設定する。
- 入力データファイル
  IOU の入力として与えられるデータを、時間軸に沿って記述する。

シミュレータは演算周期毎の各ユニットの状態をシミュレートし、出力を得る。これにより、データフローグラフの処理を実行する際のレイテンシとスループットを得ることができる。また、実行コードファイルのデバッグを行うことができる。

- GUIベースのデータフローエディタおよびシミュレータ

ユニットを並べて各入出力端子を接続することによってデータフローグラフを生成し、そのままシミュレーションを行う機能を持つ。ただしプロトタイプの作成中であり、実装は完了していない。

4. アプリケーション

アプリケーションの例としてベースラインシステムの JPEG[4]のデータフローグラフを作成した。ただし以下の3つの仮定をおいている。(1)実装では除算を行うユニットは存在しないため、量子化に必要な除算はメモリの表を参照する(2)量子化の際に、除数の表および除算結果を得るために内部メモリの値を参照するが、内部メモリが保持する値はすでに設定されているものとした。(3)離散コサイン変換で小数点演算が必要となるが、これに関しては、CGU が固定小数点数を出力できるものとし、MLU が整数と固定小数点数の乗算結果を整数として出力できるものとした。

表 1. JPEG エンコーダの実装に必要なユニット数

Table.1 Number of units for implementation of a JPEG encoder

	ALU	MLU	SFU	FCU	CGU	MRU	MWU
ピクセルシフト	1	0	0	0	1	0	0
離散コサイン変換	71	45	0	0	33	8	8
量子化	1	16	0	1	18	24	0
ジグザグ変換	15	1	0	1	17	9	8
ハフマン符号化	99	0	32	24	104	72	8
合計	187	62	32	26	173	113	24

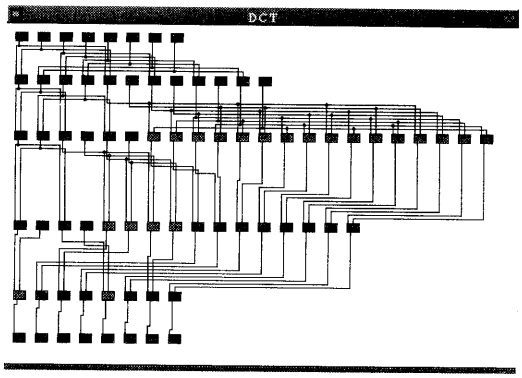

図 5　1 次元 DCT のデータフローグラフ

Fig.5　A dataflow graph of 1-dimentional DCT

JPEG エンコーダの実装に必要なユニット数を表 1 に示す。また、離散コサイン変換の一部である 1 次元離散コサイン変換のデータフローグラフを図 5 に示す。データが与えられてから最初に出力が得られるまでのレイテンシは、612 クロックサイクルである。また演算周期（18 クロックサイクル）毎に 8 ピクセル分のデータを符号化できる。

5. 考察

JPEG エンコーダのデータフロー表現例では、データの並び替えと表の参照にメモリを利用することによって処理を簡単に表現できる。たとえば離散コサイン変換 (DCT) では、1次元DCTを行った後に内部メモリを用いてデータの並び順を変換し、さらに 1 次元 DCT を行う。並び順変換では、空間方向の並びと時間方向の並びを、それぞれ 8 入力と 8 周期ごとに交換する。量子化では、メモリ内に設定された表の値によってデータの除算を行っている。ジグザグ変換も空間的に 2 次元に並んだデータを時間方向に配列させる並び替えの一

種であり、メモリを用いて実装した。

以上で述べた処理はCGUやFCU等に遅延機能を組み合わせることによっても表現できる。しかし表が大きい場合や、並び替えるために保持しなければならないデータ量が多い場合は、多数のユニットを消費してしまうことになる。このため、メモリにアクセスするユニット（MRUおよびMWU）は、演算処理を簡単にデータフローグラフとして表現するための重要な手段である。

次に、一般にアプリケーションをデータフロー表現するときの単位であるユニット機能について考察する。アプリケーションで多用される処理を、複数のユニット機能の組み合わせではなく、新しい機能として実現すれば、効率的なデータフロー表現が可能になる。このためユニット機能を新たに追加することによって、特定の処理を効率的に表現できると考えられる。

汎用プロセッサを用いた場合に比べて、同期式データフロー計算機による処理に向いているアプリケーションの特徴は以下のようになる。

- 常に一定のスループットが要求される
- レイテンシに対する要求が厳しくない
- 条件分岐が多い

データフローグラフが非常に大きいものであったとしても、同期式データフロー計算機のハードウェアに割り当てることができれば、一定のスループットを得ることができる。その反面、レイテンシはデータフローグラフをハードウェアに割り当てたときに入力から出力までに通るユニットの数と各ユニットの処理時間で決まってしまう。アルゴリズムそのものを変更することによってレイテンシを改善できる可能性もあるが、これは非常に困難である。このため、レイテンシに対する要求が厳しくなく、大量のハードウェア資源をつぎ込んでも一定のスループットを満たす必要のあるアプリケーションに向いている。

条件分岐は、汎用プロセッサでは命令パイプラインを乱して性能を低下させる原因となる。しかし同期式データフロー計算機では、ハードウェア資源が許す限り、すべての条件に対応したデータパスをハードウェア上に展開できる。その上で分岐先のパスで処理されたデータを、フロー制御ユニットを用いて選択的に次の処理に渡せるため、高速化を図ることが可能である。

6. おわりに

本報告ではデータフローグラフをハードウェアに展開し処理を行う同期式データフロー計算機を提案した。また、ベースラインJPEGデコーダを実装して必要となるハードウェア資源を特定した。その結果、メモリにアクセスできるユニットを用いることで、表の参照やデータの並び替え等の処理を、自然な形でデータフローグラフ表現に付加できることが分かった。また、同期式データフロー計算機によって性能向上の期待できるアプリケーションのもつ性質について考察した。今後の課題としては、データフローグラフをハードウェアに効率的に割り付けるための手法の開発、およびユニットが小数点数を扱う場合の手法の開発などが挙げられる。

謝辞

本研究は宮城県創造技術研究開発事業として、宮城県のご支援をいただきました。ここに感謝の意を表します。

参考文献

[1] G. De Micheli, M. Sami. "Hardware/Software Co-Design," Kluwer Academic Publishers.

[2] William H. Mangione-Smith et al., "Seeking Solutions in Configurable Computing," IEEE Computer, Dec. 1997, pp. 38-43."

[3] G. D. Micheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill, 1994.

[4] ケイワーク,「JPEG 概念から C++による実装まで」, ソフトバンク, 1998年