

ロータリー型コンピュータ

身次 茂

熊本テクノポリス財団 電子応用機械技術研究所

〒861-22 熊本県上益城郡益城町田原 2 0 8 1 - 1 0

あらし 演算器をリング状に結合したロータリー型のコンピュータを提案する。ロータリー型コンピュータは、データフローグラフをそのまま実行することをイメージして作られている。また、パイプライン処理方式のコンピュータとして見た場合、無駄な待ち合わせがなく高速である。

ロータリー型コンピュータの演算器を仮想回路で構成する。仮想回路とは、FPGA（フィールドプログラマブルゲートアレイ）のゲート間接続スイッチの1つ1つに対して複数の記憶素子を割り当てたものである。記憶素子をセレクタにより切り替えることで、回路を別の回路に瞬時に変更できる。仮想回路をユーザー定義命令実行回路として用いる方法についても述べる。

和文キーワード ロータリー 演算器 リング 仮想回路 FPGA ユーザー定義命令

Rotary Computer

Shigeru Mithugi

Applied Electronics Research Center, Kumamoto Technopolis Foundation

2081-10 Tabaru Mashiki-machi Kamimashiki-gun Kumamoto 861-22 Japan

Abstract

Rotary computer with arithmetic unit form a ring is proposed. Rotary computer is created in image of executing directly data flow diagram. Rotary computer is faster than pipeline processing. Since there is not useless waiting time.

英文 key words rotary, arithmetic unit, ring, virtual circuit, FPGA, user defined instruction

1. はじめに

コンピュータの性能は、いつの時代も充足されることはなく、常にさらなる性能向上が求められている。最も性能のすぐれたコンピュータは、理論的には図1に示すようなものであろう。無限に広いシリコンチップがあり、その上にプログラムをデータフローの形に展開し、演算の数だけ演算器を用意し、その間を直接接続する。図1のa, bの加算とc, dの減算のように、並列に実行できる演算はすべて並列に演算する。実現性の面で、図1のチップには以下の3つの問題がある。

- (1) 演算器の間を配線で固定すると、プログラム性がなくなる。
- (2) 無限に幅広いチップは実現できない。
- (3) 無限に横長いチップは実現できない。

3つの問題を以下の方法で解決する。

- (1 a) プログラム可能なハードウェアであるフィールド・プログラマブル・ゲートアレイ（以下FPGA）を用いる。
- (2 a) 幅のある一定幅に制限する。これにより、並列実行できる演算の数が制限を受ける。
- (3 a) 無限の長さがあるように見せかけるため、チップ

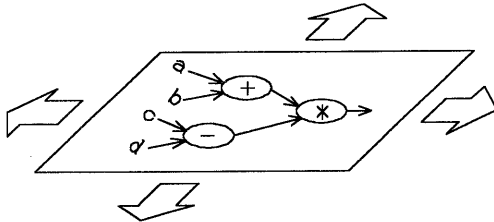


図1 無限の広さのシリコンチップ

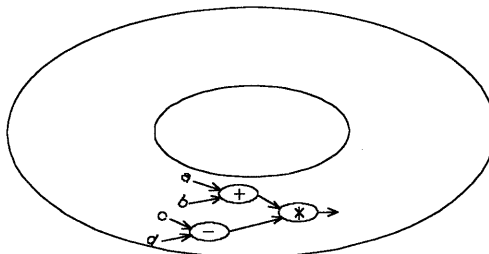


図2 円盤FPGA

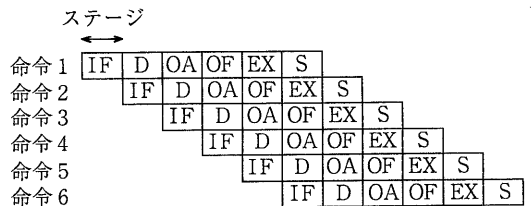
チップをリング状にする（図2参照）。演算がある程度進むと、演算データがまたもとの場所に戻ってくるが、その前にその場所の回路を別のもので書き換えておく。そうすることにより、必要な処理を無限に続けることができる。

(3 a) にはまだ問題がある。ある場所の回路が加算器になっていたとし、データが次にそこを通るときは乗算器が必要であるとする。加算器から乗算器にするために、FPGAのプログラマブル接続のためのスイッチデータを1000個書き換えるとする。加算等の時間は10ns程度であるから、スイッチデータの書替もこの時間で行うとすると、書替データの転送量は12.8Gバイト/秒と膨大になり、実現できない。(3 a)の解決策として、

- (3 b) 仮想回路を用いる（仮想回路については本文で説明する）。

以上を具体的に実現したものが、ロータリー型コンピュータである[1]。

現在、コンピュータの高速化手法として、パイプライン処理方式が多く用いられている。1つの命令をいくつかのステージに分け、複数の命令をステージ単位で並列実行する（図3参照）。ステージは、たとえば命令フェッチ、命令デコード、オペランドアドレス計算、オペランドフェッチ、演算実行、結果格納である。パイプライン処理では、各ステージの実行時間をすべて同じにしなければならない。そのため遅いステージがあると、他のステージの実行時間もそれに合わせて遅くなる。演算器をリング状に結合したロータリー型のコンピュータはこの問題を解決する。



- IF : 命令フェッチ
- D : 命令デコード
- OA : オペランドアドレス計算
- OF : オペランドフェッチ
- EX : 演算実行
- S : 結果格納

図3 パイプライン処理

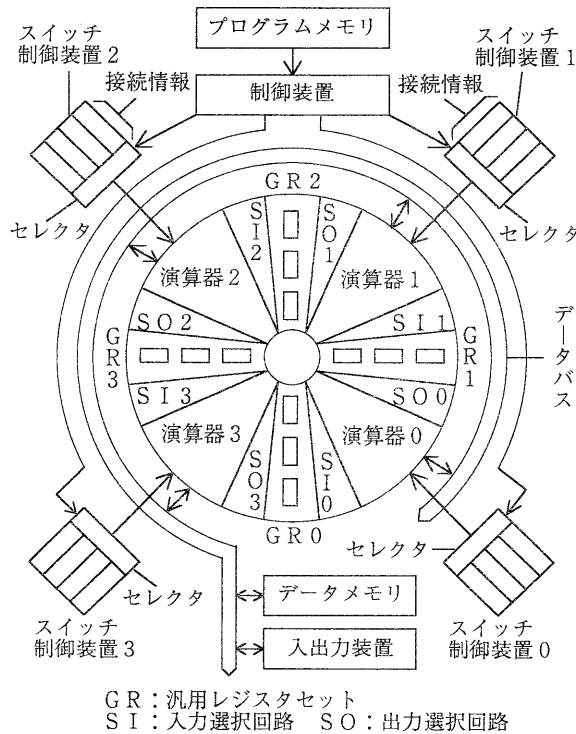
2. 構造

基本構造は、演算器をリング状に結合したものである。演算器が4つの場合を図4に示す。

2.1 仮想回路

演算器を仮想回路で構成する。仮想回路は、フィールド・プログラマブル・ゲートアレイ（以下FPGA、図

5参照）において、ゲート間の接続を変更するスイッチの1つ1つに対して複数の記憶素子を割り当てたものである（図6参照）。記憶素子はセクタを介してスイッチの開閉を制御する。接続スイッチはアナログスイッチであり、記憶素子はSRAMの1セルである。記憶素子群が1つの接続情報つまり回路情報を持つ。回路は加算回路、減算回路、比較回路、ロード回路、ストア回路などである。記憶素子群の切替はセクタで行う。セクタを制御することにより、接続スイッチと記憶素子の対応が一斉に切り替わり、それに応じてFPGA内の結線が切り替わり、回路が他の回路に替わる。これは機能の切替と言ってもよい。切替は高速である（100nS程度）。仮想回路では、複数の機能を始めから回路として用意しておく必要はなく、必要になった時点で必要な回路を素早く準備する。このため、回路素子（ゲートなど）を節約できる。



GR：汎用レジスタセット
SI：入力選択回路 SO：出力選択回路

図4 ロータリー型コンピュータ

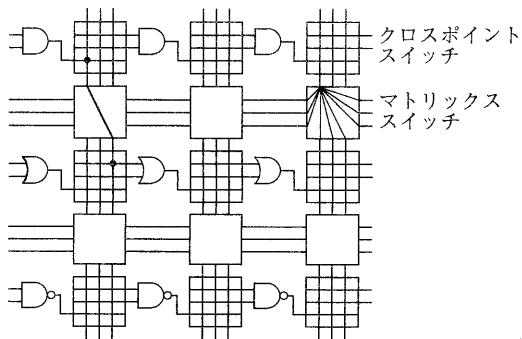


図5 FPGA

2.2 その他の構成

演算器と演算器の間には汎用レジスタ・セットを置く。汎用レジスタ・セットが4つあることになる。汎用レジスタ・セットは、5つの汎用レジスタGRi0～GRi4とフラグレジスタFRi（i=0～3）からなり（図7参照）、GRi1～GRi3はインデックスレジスタとしても用い

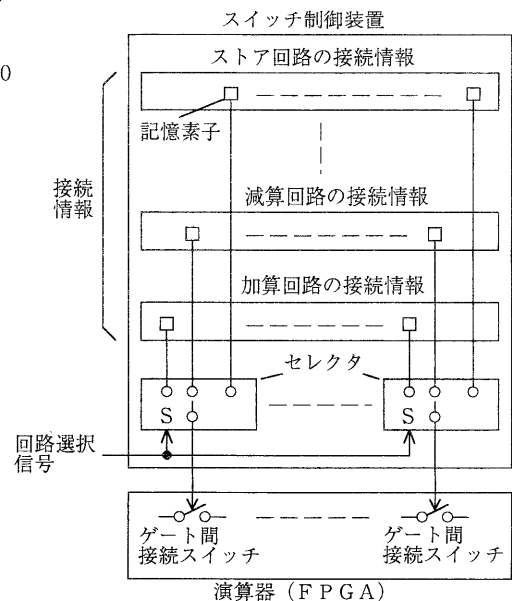


図6 仮想回路

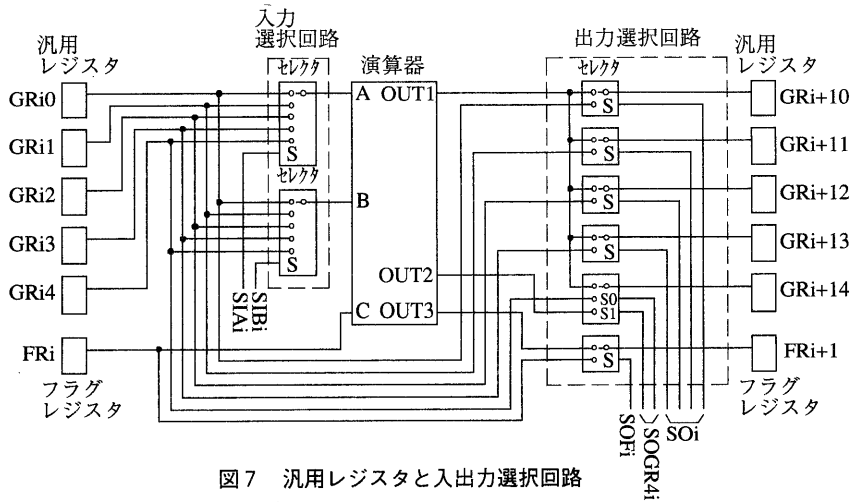


図7 汎用レジスタと入出力選択回路

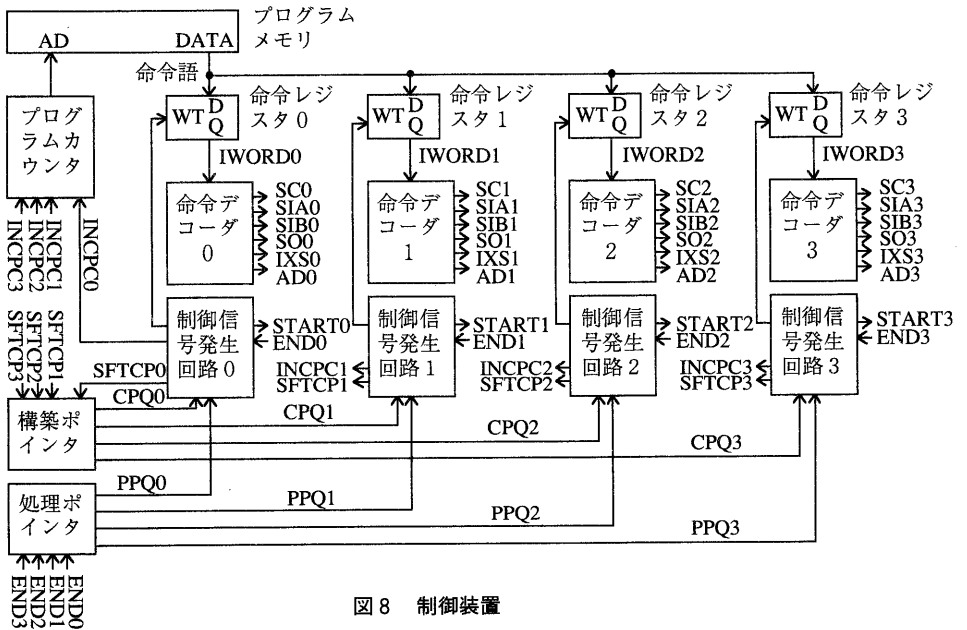


図8 制御装置

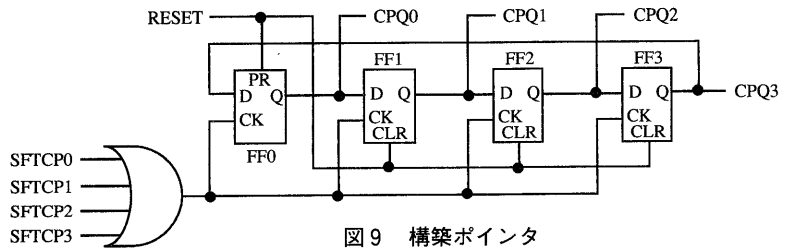


図9 構築ポインタ

る。

汎用レジスタと演算器の間には、命令に従って汎用レジスタを選択しデータを演算器に送る入力選択回路と、演算結果を命令が指定する汎用レジスタに送り出す出力選択回路を置く(図7参照)。

制御装置はプログラムカウンタと、4つの演算器にそ

れぞれ対応した命令レジスタ0~3, 命令デコーダ0~3, 制御信号発生回路0~3からなる(図8参照)。制御装置はまた、次に回路を構築すべき演算器を指し示す構築ポインタ(図9参照)と、次に処理を行うべき演算器を指し示す処理ポインタ(構築ポインタと同様の構成)を含む。制御信号発生回路には、回路構築が終わったことを示す構築済フラグと、演算処理が終わったことを示す処理済フラグがある(図10参照)。

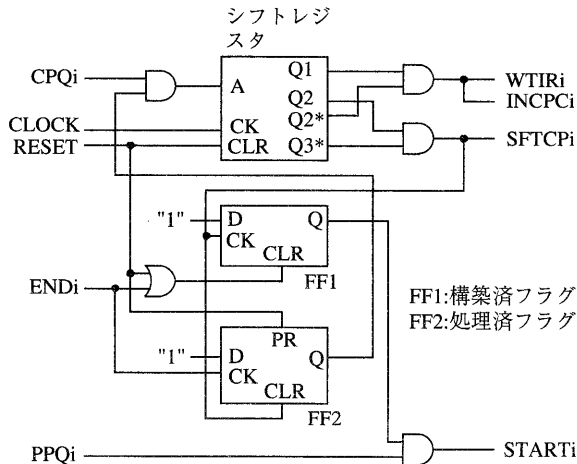


図10 制御信号発生回路

3. 動作

プログラムとデータをそれぞれプログラムメモリとデータメモリに格納する。命令語の構成を図11に示す。プログラムは、データメモリにある2つの数値を足し合わせ、2倍し、結果を再びデータメモリに格納する処理であるとする(図12参照)。

3.1 回路構築動作

制御装置は、プログラムカウンタが指し示す0番地の命令を読み、構築ポインタが指し示す命令

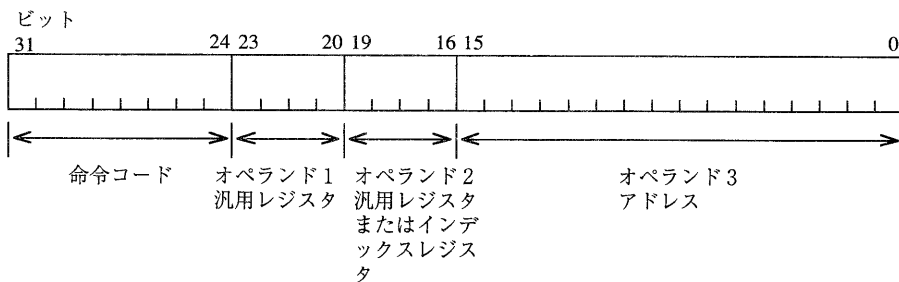


図11 命令語

```
LD GR1, 100
LD GR2, 101
ADD GR1, GR2
LEA GR2, 1
SLA GR1, GR2
ST GR1, 102
```

図12 プログラム

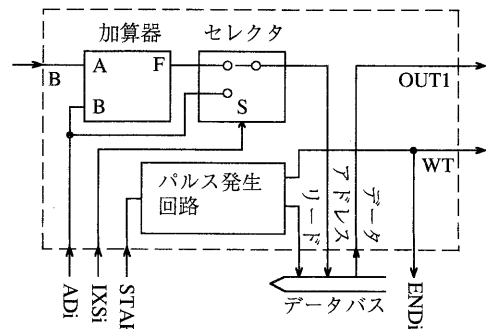


図13 ロード回路

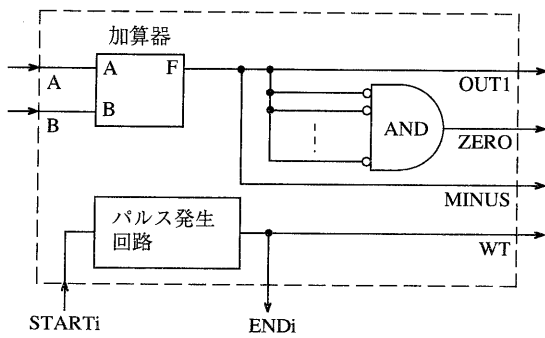


図 14 加算回路

レジスタ 0 に格納する。命令デコーダ 0 で命令を解読し、汎用レジスタを選択する信号 S I A 0, S I B 0, S O 0 を入力選択回路と出力選択回路に送り、命令が指定するオペランドアドレス A D 0 を演算器 0 に送る。また、回路を選択する信号 S C 0 を演算器 0 のスイッチ制御装置 0 に送る。スイッチ制御装置 0 では、セレクタが働き記憶素子群の切替を行う。これにより演算器 0 には命令で指定された回路、ここではロード回路 (図 13 参照) が構築される。演算器 0 の回路切替を待って、制御信号発生回路 0 が演算器 0 に対して処理スタート信号 S T A R T 0 を出す。演算器 0 は処理、ここではデータのロードを始める。

演算器の動作とは独立に、命令読込動作、回路構築動作が続く。制御信号発生回路 0 は演算器 0 の回路構築が終わると、プログラムカウンタを 1, 構築ポインタを 1 に変更する。制御装置はプログラムメモリから次の命令を読み出し、命令レジスタ 1 に格納する。演算器 0 と同様にして、演算器 1 には命令が指定する回路、ここではロード回路を構築する。以下同様にして、演算器 2 に加算回路 (図 14 参照)、演算器 3 にロード実効アドレス回路を構築する。演算器をリング状に結合しているため、演算器 3 の次は演算器 0 となる。演算器 0 では前の処理 (データのロード) がまだ行われているため、演算器 0 の構築動作は行わない。ロード処理が終わって処理済フラグが立つのを待って、演算器 0 の構築を行う。

3. 2 処理動作

処理動作を説明する。制御信号発生回路から処理スタート信号 S T A R T 0 を受け取ると演算器 0 はロード処理を行い、データメモリの 100 番地の数値を汎用レジスタ G R 11 にロードする。他の汎用レジスタ G R 10, G R 12 ~ G R 14 には、前の段の汎用レジスタ G R 00, G R 02 ~ G R 04 の内容がコピーされる。処理が終わると、演算器 0 から処理終了信号 E N D 0 が制御信号発生回路 0 に送られる。処理終了信号 E N D 0 により、処理済フラグがセットされ、処理ポインタが 1 つ進み制御信号発生回路 1 を指す。演算器 1 の構築は終わっているので、制御信号発生回路 1 はすぐに処理スタート信号 S T A R T 1 を発生し演

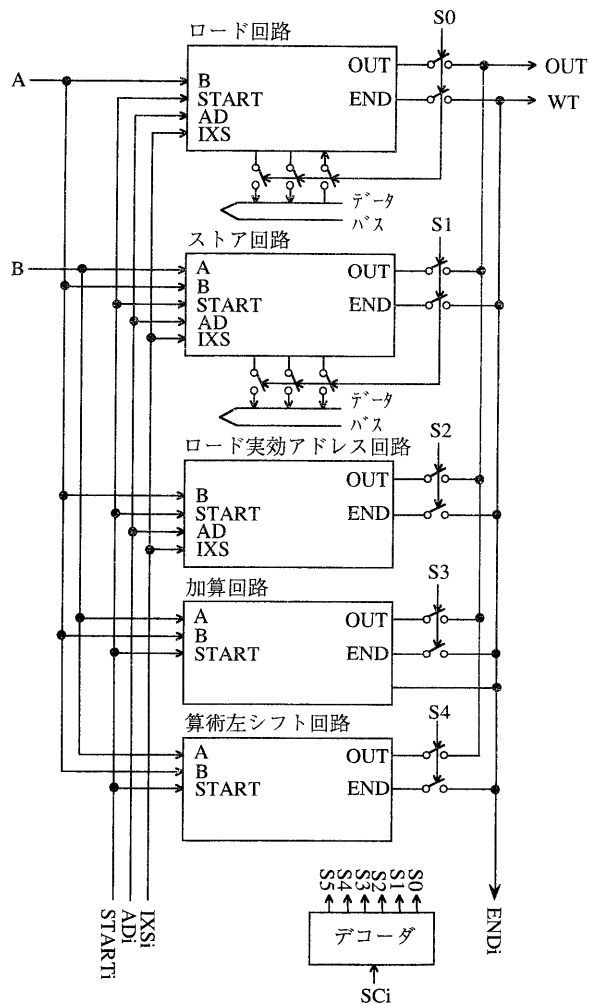


図 15 ゲートアレイによる演算器

算器1に送る。演算器1でもロード処理を行い、データメモリの101番地の数値を汎用レジスタGR22にロードする。このとき他の汎用レジスタには、前の段の汎用レジスタの内容がコピーされる。処理ポイントが1つ進み制御信号発生回路2を指す。演算器2では加算処理を行い、汎用レジスタGR21と汎用レジスタGR22の数値を足し合わせ、結果を汎用レジスタGR31に格納する。演算器3ではロード実効アドレス処理を行い、汎用レジスタGR02に左シフトの回数1を格納する。ここで次の演算器0での処理に移るのであるが、ここでもし演算器0の回路構築、つまりロード命令回路から算術左シフト命令回路への切替が済んでいなければ、演算器0での処理スタートは行わない。演算器0の回路構築が終わり構築済フラグが立つのを待って処理を始める。

上記の処理が行われている間に回路構築が進み、演算器0に算術左シフト回路、演算器1にストア回路が構築される。

演算器0で算術左シフト処理を行い、汎用レジスタGR01の数値を2倍する。演算器1でストア処理を行い、汎用レジスタGR11に格納されている演算結果をデータメモリの102番地にストアする。これで一連の処理が終わる。

これまでの動作から分かるように、回路構築と処理が追いついて進むように進むが、互いに相手を追い越すことはない。回路構築が処理よりも十分速ければ、処理は待たされることなく進み、最も処理効率が良い。

4. 課題とその対策

ロータリー型コンピュータの性能のネックは仮想回路である。仮想回路およびFPGAは、接続をプログラマブルとするためにアナログスイッチを用いている。アナログスイッチはオン抵抗が数10～数100オームあるため、信号の遅延を生じる。ディスクリートのアナログスイッチで実験した結果では、オン抵抗50オームで遅延時間2.5nSである。LSI内部では浮遊容量が小さいため、この遅延はもっと小さいと思われるが、いずれにせよ、ゲート間を配線のみで接続するのに比べると遅延は大きい。オン抵抗ゼロ、あるいはゼロに近いアナログスイッチは超電導や単一電子

トランジスタ[2]で実現できるかもしれない。しかし、ここでは既存の技術での解決方法を考える。

4.1 ゲートアレイによる実現

ロータリー型コンピュータは仮想回路を用いないと実現できないわけではなく、図15に示すように、ゲートアレイを用いて演算器を組めば実現できる。仮想回路のように必要なときに必要な回路をダイナミックに作り出すのではなく、最初から複数の演算器を並べておき、それらの出力をセレクトで切り替える。

4.2 ユーザー定義命令

さらに、仮想回路の利点を生かすために、上記演算器に並列に仮想回路を置く(図16)。仮想回路を付け加える目的は、ユーザー定義命令を仮想回路で実行することにある。特定のアプリケーションで数多く用いる演算を一命令で実行できるようにすることができる。例えば、 $(a+b) * (c-d)$ を数多く計算するプログラムがあったとする。一般には、この演算は既存命令の組み合わせで数ステップを使って計算する。仮想回路のプログラム性を利用すれば、この演算を一命令で実行する演算回路を作り出すことができる。仮想回路は、ユーザー定義命令を複数種類持つことができる。それらを切り替えて実行する様子は、仮想回路を用いたロータリー型コンピュータで説明したとおりである。

このようなハイブリッド構成のロータリー型コン

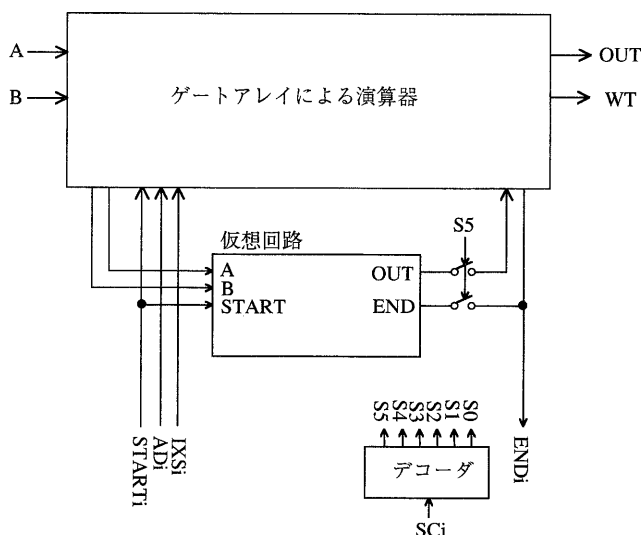


図16 ユーザー定義命令実行回路

コンピュータが現在の技術での実用的な答である。パイプライン方式のコンピュータでは、1つの命令の演算の長さが他の命令のステージに影響するため、ユーザー定義命令は実現しにくい。その点、ロータリー型コンピュータではそのような干渉がないため、容易にユーザー定義命令実行機構をハードウェアで実現できる。

92年8月

[3] 大見 忠弘, "シリコンテクノロジーの展望", 電子情報通信学会誌 Vol. 75 No. 11, p. 1207-1215, 1992年11月

4.3 ソフトハードウェアロジックによる実現

仮想回路と同じようにデジタル回路にプログラム性を持たせ、リアルタイムに機能を変更できるものとして、ソフトハードウェアロジックがある [3]。仮想回路をソフトハードウェアロジックで置き換えることもできる。

5. まとめ

演算器をリング状に結合したロータリー型のコンピュータを提案した。このコンピュータは、

- (1) 命令取り出し動作、解読動作と命令実行動作をまったく独立に行う。
- (2) パイプラインのステージの実行時間に相当するものは、その演算の実行時間のみによって決まり、命令取り出しや解読の時間の一番遅いものに従う必要はない。
- (3) 命令に応じて実行ステージの演算時間が大幅に変化しても、回路構築動作、処理動作それぞれが待ち合わせを行う。たとえば、加算命令と除算命令では実行時間が大幅に異なる。
- (4) 演算器を普通のゲートアレイで構成してもよい。
- (5) 仮想回路をユーザー定義命令実行機構として用いることができる。

ロータリー型コンピュータはパイプライン処理方式の実行ステージだけを抜き出し、リング状に並べたものとも言える。

参考文献

- [1] 身次 茂, "ロータリー型コンピュータと仮想回路", 情報処理学会第44回全国大会6D-6, pp. 109-110, 1992年3月
- [2] K. K. リカレフ, T. クラーソン, "シングル・エレクトロニクス素子", 日経サイエンス Vol. 22 No. 8, pp. 92-100, 19