計算機アーキテクチャ　98－19
設　計　自　動　化　65－19
(1993. 1. 22)

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２０８１－１０

あらまし　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 witch 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。最も性能のでるコンピュータは、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。

（１ａ）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。
（２ａ）幅を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。
（３ａ）無限の長さがあるように見せかけるため、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。

（３ａ）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。（３ａ）の解決策として、

（３ｂ）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あわせて遅くなる。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。

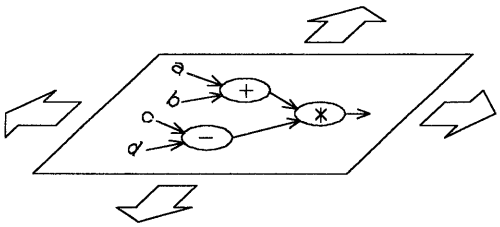


図１　無限の広さのシリコンチップ

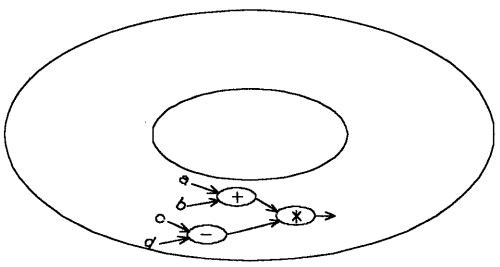


図２　円盤ＦＰＧＡ

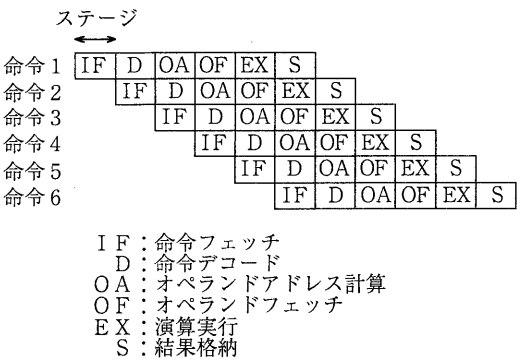


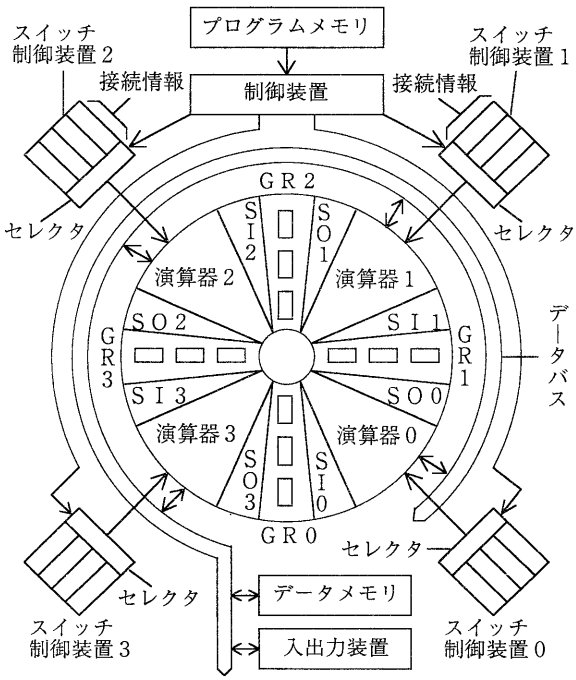
図３　パイプライン処理

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）。記憶素子はセレ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レクタで行う。セレ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

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

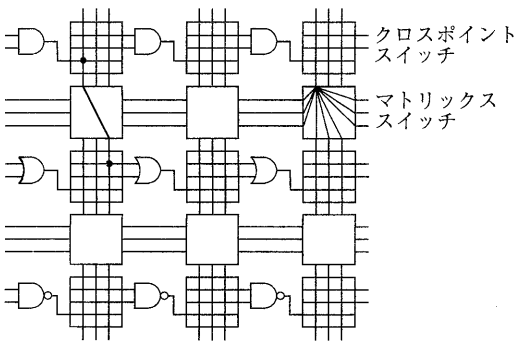


図５　ＦＰＧＡ

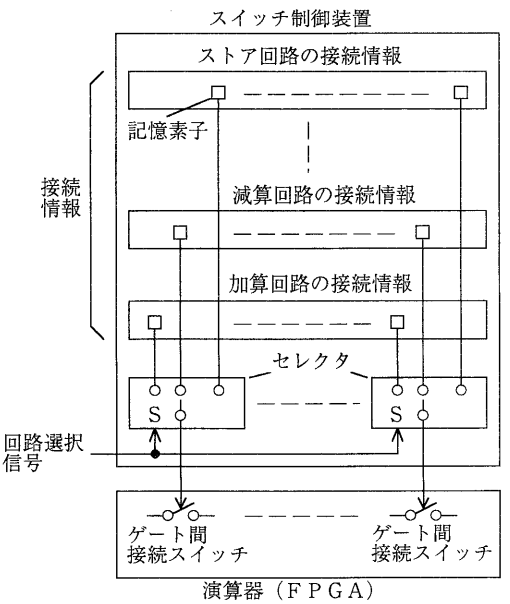


図６　仮想回路

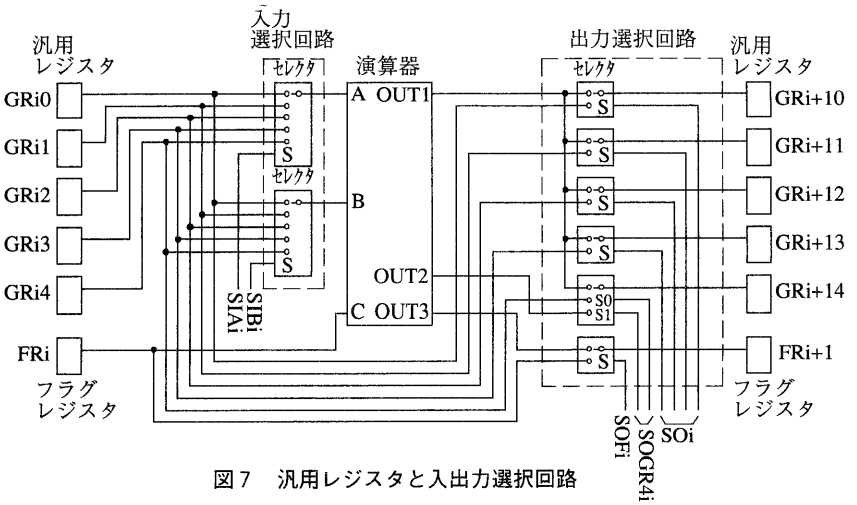


図7　汎用レジスタと入出力選択回路

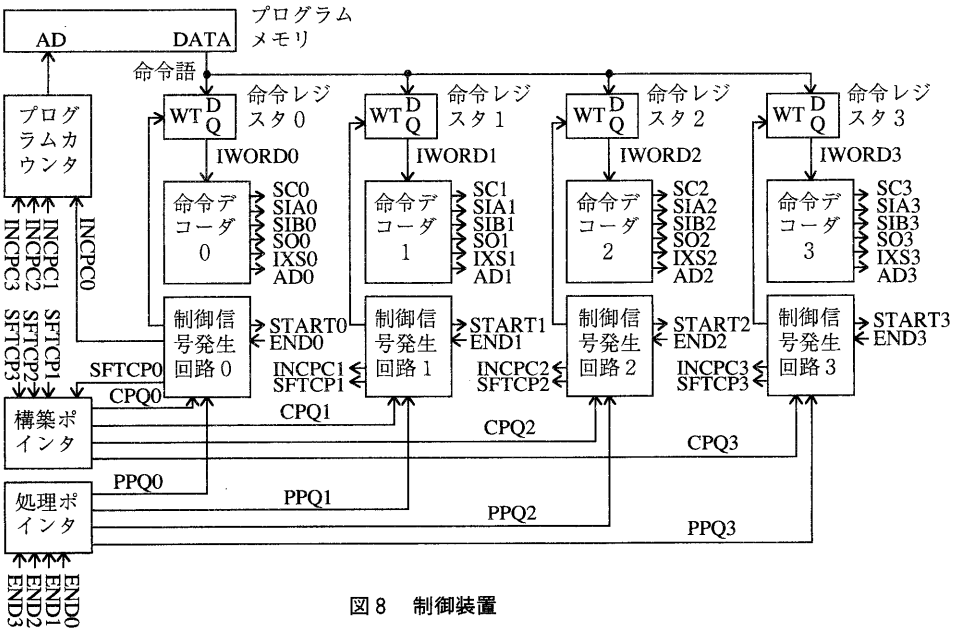


図8　制御装置

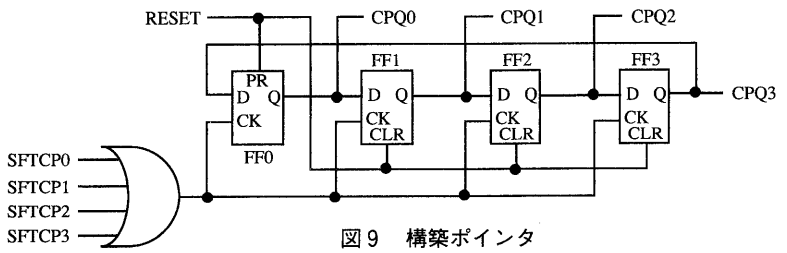


図9　構築ポインタ

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）。

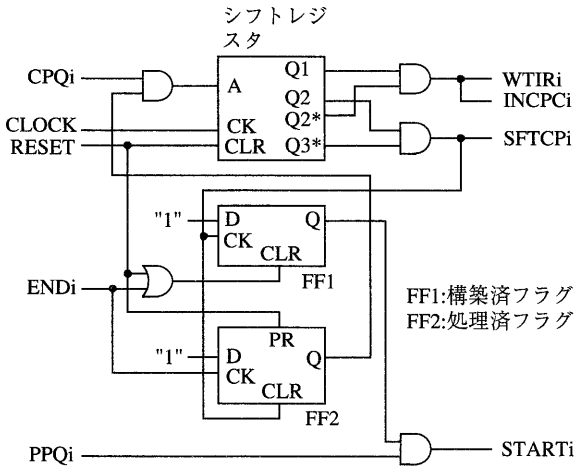


図１０　制御信号発生回路

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

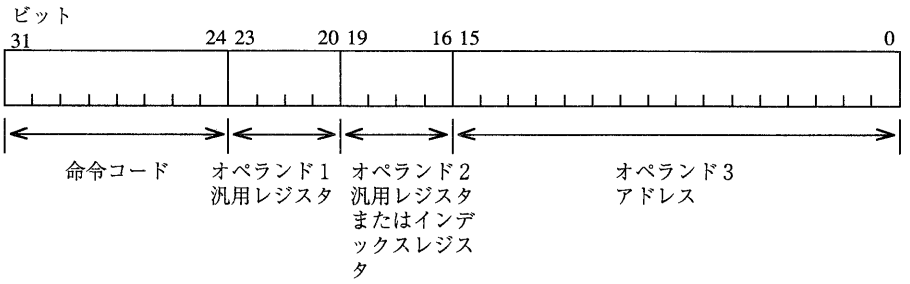


図１１　命令語

```
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

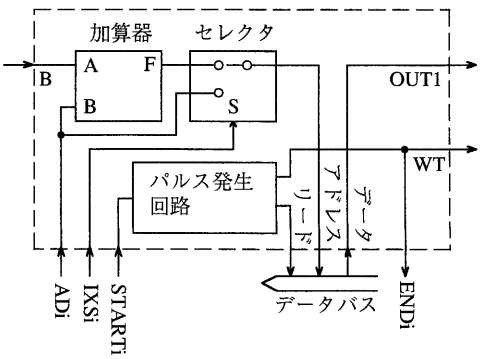


図１３　ロード回路

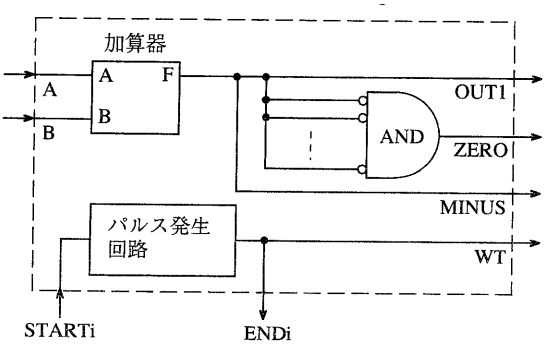


図１４　加算回路

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ＳＩＡ０，ＳＩＢ０，ＳＯ０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ＡＤ０を演算器０に送る。また、回路を選択する信号ＳＣ０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ＳＴＡＲＴ０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。ロード処理が終わり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ＳＴＡＲＴ０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ＧＲ１１にロードする。他の汎用レジスタＧＲ１０，ＧＲ１２～ＧＲ１４には、前の段の汎用レジスタＧＲ００，ＧＲ０２～ＧＲ０４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ＥＮＤ０が制御信号発生回路０に送られる。処理終了信号ＥＮＤ０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ＳＴＡＲＴ１を発生し演

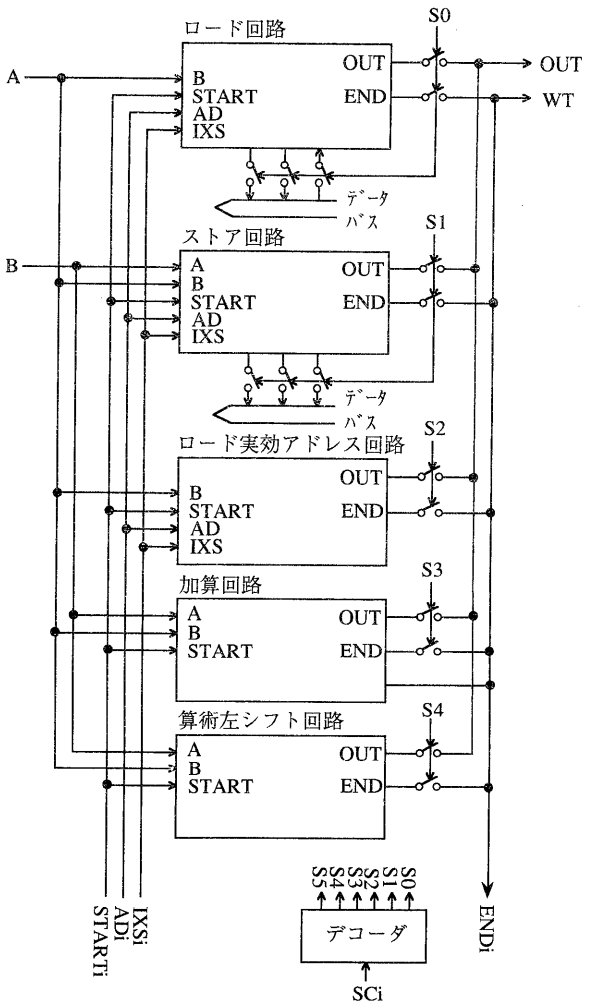


図１５　ゲートアレイによる演算器

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。処理ポインタが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かけっこをするように進むが、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タで切り替える。

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

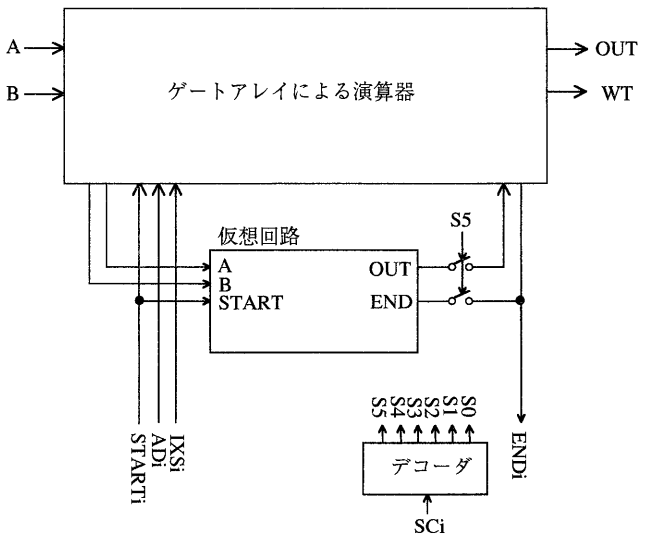


図１６　ユーザー定義命令実行回路

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。

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．２２Ｎｏ．８，ｐｐ．９２－１００，１９９２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．７５Ｎｏ．１１，ｐｐ．１２０７－１２１５，１９９２年１１月