システムＬＳＩ設計技術　99－7
(2001. 1. 12)

動的再構成型桁直列演算による離散コサイン変換回路

伊藤 和人

埼玉大学 電気電子システム工学科

〒338-8570 埼玉県浦和市下大久保255

E-mail: kazuhito@ees.saitama-u.ac.jp

あらまし　半導体回路の微細化により，配線を用いたデータ通信時間が相対的に増加しており，データ通信時間が処理速度向上を妨げると予想されている．動的再構成型演算回路を用いれば，近傍の回路を所要の演算器に再構成することでデータ通信時間を低減し，高速な処理を達成できる．本論文では，桁直列演算回路の再構成が容易であることに着目し，動的再構成型桁直列演算によって一次元離散コサイン変換をコンパクトかつ高速に行う回路を提案する．提案回路はLSI試作を行い，回路シミュレーションを用いた遅延時間評価によって提案回路の有効性を確認した．

キーワード　動的再構成，桁直列演算，DCT，スケジューリング，VLSIアーキテクチャ

A Discrete Cosine Transform Circuit with Dynamically Reconfigurable Digit-Serial Computation

Kazuhito Ito

Department of Electrical and Electronic Systems, Saitama University

255 Shimookubo, Urawa, Saiama 338-8570, Japan

E-mail: kazuhito@ees.saitama-u.ac.jp

Abstract　In the era of deep submicron technology, wire delay on an LSI chip is becoming relatively larger than operation delay. Data communication time by wire delay between processing units could be reduced and hence fast processing can be achieved if nearby processing units are dynamically reconfigured into desired operation type and execute operations on the reconfigured units. Based on the simplicity of reconfiguring digit-serial computation, we propose a compact and fast 1-D discrete cosine transfer circuit with dynamically reconfigurable digit-serial computation. An LSI chip is designed and its speed is measured by a circuit simulator. Results show the effectiveness of the proposed circuit.

key words　dynamic reconfiguration, digit-serial, DCT, scheduling, VLSI architecture

1 はじめに

半導体技術の進歩により，LSI上の回路の微細化が進んでいる．LSI回路の遅延は，ゲート回路遅延とゲート間配線遅延の和となるが，ディープサブミクロンプロセスによるLSIでは，全遅延時間に占める配線遅延の割合が増し，動作速度の向上を妨げると予想されている[1, 2]．ゲート間配線遅延時間を考慮して動作速度を向上するには，上流設計において長大な配線を発生しない工夫や配線遅延とゲート遅延を並列化して配線遅延が影響しない工夫などが有効であると考えられる．

ディジタル回路中の信号値を書き換えることで，回路の機能を変更することをディジタル回路の再構成と定義する．ディジタル信号処理などの数値演算処理では，一般に加算や乗算といった複数種類の演算を多数実行する．演算処理速度の向上には複数の演算器を用いた並列処理が有効であるが，処理中は必ずしも演算の並列度は一定ではなく，各時刻で要求する演算器数は処理の進行にしたがって変化する場合がある．ある乗算器Mが行った乗算結果を他のデータとともに加算する場合には，乗算器Mから加算器へ乗算結果のデータ通信が必要となる．加算器と乗算器Mの距離が大きければ，大きなデータ通信時間が必要となり，処理速度を低下させる．しかし，乗算器Mの近傍に演算を実行しない演算器があれば，その演算器を加算器に再構成することでMから加算器へのデータ通信時間を短縮できる[3]．

再構成方式には，静的再構成と動的再構成がある[4]．静的再構成は，回路機能変更の際に回路全体の動作を中断し，再構成が完了したら動作を開始する再構成方式である．一方，動的再構成は，機能変更が必要な回路が一部である場合に，その一部の回路の再構成を他部の動作と並行して行う再構成方式である．再構成を行なう部分が再構成完了後に新たな機能で動作を再開した後は，再構成しなかった部分と協調して動作を続行する[5, 6]．

動的再構成型の回路では，再構成の際に回路全体の動作中断が不要である．また，処理の進行に応じて不要となった機能を必要な機能へ再構成することで回路資源を有効に活用できる可能性がある．さらに，前述のように配線遅延に起因するデータ通信時間を短縮することができる．以上のように，動的再構成型回路は，LSIチップ面積と動作速度の点で従来の非再構成型回路では到達できない最適解の候補を与えると考えられる．動的再構成可能な回路の開発[7, 8]ならびに動的再構成可能回路を用いた処理実装

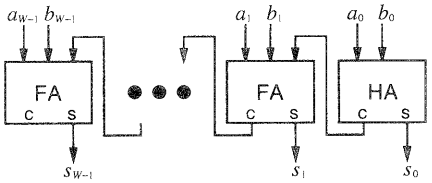

図 1. ビット並列加算回路

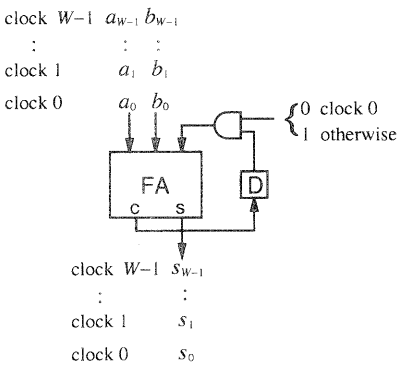

図 2. ビット直列加算回路

[9, 10, 11]が活発に研究されている．

本研究では，桁直列(digit-serial)固定小数点演算[12]を行う乗算回路と加算回路の間の再構成が容易であることに着目し，動的再構成によって演算器間データ通信時間を低減して高い処理速度を実現する離散コサイン変換回路を設計する．

2節では，桁直列演算を紹介し，加算回路と乗算回路の間の再構成方法を述べる．3節では，動的再構成型桁直列演算による8ポイント一次元離散コサイン変換回路を提案し，LSI設計結果についての考察を4節で述べる．

2 桁直列処理と再構成

2.1 桁直列処理

データ語長をWビットとするとき，Wビットの信号を同時に処理する方式をビット並列(bit parallel)処理と呼ぶ．ビット並列加算では，データ語の全ビットを同時(同一クロック時刻)に入力し，加算結果のデータ語の全ビットを同時に出力する．図1にリップルキャリー型ビット並列加算回路を示す．$(a_{W-1}, \ldots, a_1, a_0)$と$(b_{W-1}, \ldots, b_1, b_0)$が2つの$W$ビット入力データであり，$(s_{W-1}, \ldots, s_1, s_0)$は$W$ビットの加算結果である．HAは半加算器，FAは全加算器を意味する．リップルキャリー型加算では，あるビットの処理が下位ビットの処理結果(桁上げ)に依存しており，データ

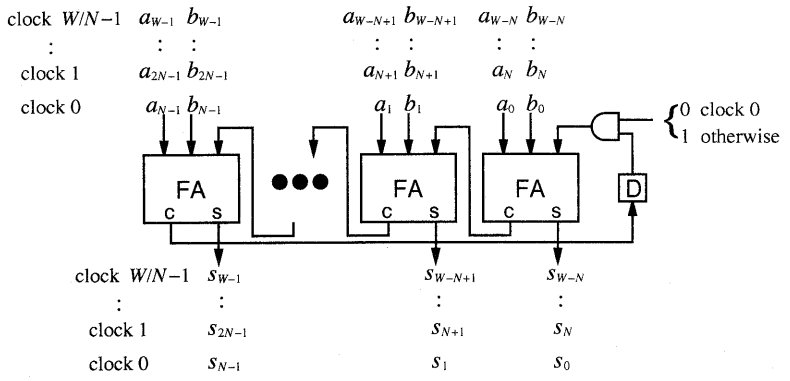

図 3. 桁直列加算回路 (桁サイズ N ビット)

を入力してから加算結果の全ビットの出力が揃うまでの遅延時間が大きい．このように，ビット並列処理は，クロック数は最小の1クロックのみであるが，クロック周期が長くなる傾向がある．

W ビットのデータを処理する際に，1ビットずつ W 回のクロック周期に分割して処理する方式をビット直列 (bit serial) 処理と呼ぶ．図2にリップルキャリー型ビット直列加算回路を示す．2つの被加算データは，最下位ビットから順に1クロックあたり1ビットずつ入力する．ビット直列処理では，1クロックあたり1ビットの処理しか行わないので，あるビットの加算を行った結果発生する桁上げは図中の'D'で表す1ビットレジスタに保存し，次のクロック周期に実行される上位ビットの加算に引き渡す．なお，最下位ビットの加算のときに下位ビットからの桁上げを強制的に0とするための回路が必要であり，図2ではANDゲートで実現している．リップルキャリー型加算回路では，クロック周期は全加算器の遅延時間程度であり極めて短いが，加算を完了するまでに W クロックを要する．

全加算器の遅延時間を T_{FA} とすると，リップルキャリー型ビット並列加算ではクロック周期の下限は WT_{FA} となる．加算は1クロックで完了するので，加算に要する時間は WT_{FA} である．大きな W に対しては，リップルキャリー型加算回路がクリティカルパスとなってクロック周期短縮のボトルネックとなることがある．これは，桁上げ先見型などのより遅延時間の小さな加算回路を用いることで解決できるが，加算回路のゲート数は増加する．一方，リップルキャリー型ビット直列加算では1ビットの加算を全加算器で行うので，クロック周期の下限は T_{FA} となる．加算が W クロックで完了するので，周期が T_{FA} のクロックで駆動した場合にビット直列加算に要する時間は WT_{FA} となり，ビット並列加算と同じである．ところが，一般に T_{FA} は極めて短時間であり，周期が T_{FA} のクロックの実現は困難である．低速なクロックを使用する結果，実際にはビット並列加算に比べて低速なビット直列加算しか実現できない．

ビット並列処理とビット直列処理の短所を補う処理方式が桁直列 (digit serial) 処理である [12]．桁直列処理では，1つのデータの W ビットの信号を N ビットの桁 (digit) に分割し，1桁ずつ順に処理する．1回の処理は W/N クロックで完了する．図3に，リップルキャリー型桁直列加算回路を示す．2つの被加算データは，最下位桁から順に1クロックあたり1桁ずつ入力する．1桁分の加算は，データ語長 N ビットのリップルキャリー型ビット並列加算として行うので，クロック周期の下限は NT_{FA} となる．加算は W/N クロックで完了するので，加算に要する時間は WT_{FA} となり，ビット並列加算と同じである．

桁直列処理では，ビット並列処理に比べてクリティカルパスが小さく，より周期の速いクロックで駆動でき，その一方で極端に速いクロックを使用しなくても十分な処理性能を達成することができる．

2.2 桁直列加算回路

データ語長 $W=4$ ビット，桁サイズ $N=2$ ビットの桁直列加算回路を図4に示す．2つの被加算データを，クロック t に最下位桁，クロック $t+1$ に次の桁というように，1クロックあたり1桁 (2ビット) ずつ最下位桁から最上位桁へ順次連続して入力する．被加算データの最下位桁を入力した次のクロック $t+1$ に加算結果の最下位桁が出力され，以降連続するクロックに加算結果が下位桁から順に1桁ずつ出力される．ある桁から次の上位桁への桁上げは，1ビット

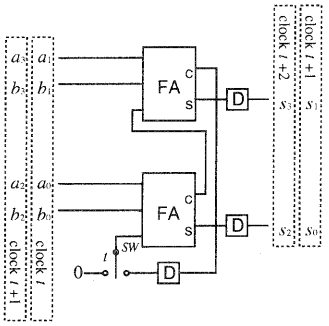

図 4. 桁直列加算回路 (データ 4 ビット，桁サイズ $N = 2$ ビット)

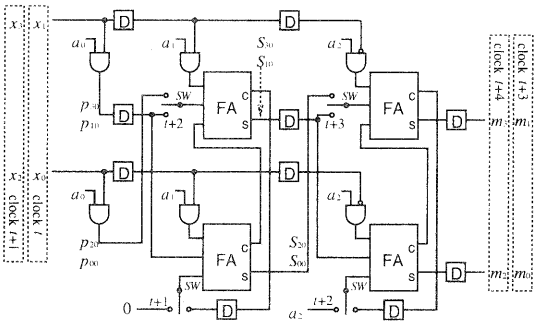

図 5. 桁直列乗算回路 (データ 4 ビット，積定数 3 ビット，桁サイズ $N = 2$ ビット)

レジスタに保存する．最下位桁の加算を行うときには，下位桁からの桁上げを 0 とする．図 4 で SW と示されたスイッチを切り替えることで，1 ビットレジスタに保存された桁上げとデータ 0 とを選択して加算器に入力できる．スイッチの近傍に記されている 't' は，最下位桁が入力されるクロック t においてデータ 0 が選択されるようにスイッチを切り替えることを意味する．他の時刻では，桁上げが選択される．

桁直列加算回路では，スイッチおよび N 段の全加算器からなるパスが最長遅延となる．

2.3 桁直列定数乗算回路

4 ビット被乗算データ (x_3, x_2, x_1, x_0) と 3 ビット積定数 (a_2, a_1, a_0) の桁直列乗算の手順を式 (1) に示す。

$$
\begin{array}{llllll|ll}
 & & & x_3 & x_2 & x_1 & x_0 \\
 & & \times) & & a_2 & a_1 & a_0 \\
\hline
 & & \boxed{p_{30}} & p_{30} & p_{20} & p_{10} & p_{00} & \\
 & +) & p_{31} & p_{21} & p_{11} & p_{01} & & \\
\hline
 & \boxed{S_{30}} & S_{30} & S_{20} & S_{10} & S_{00} & & \\
 & p_{32} & p_{22} & p_{12} & p_{02} & & & \\
+) & & & & a_2 & & & \\
\hline
 & m_3 & m_2 & m_1 & m_0 & & &
\end{array}
\qquad (1)
$$

ここで，p_{ij} は x_i と a_j の積 (論理積) を表す．ただし，積定数の符号を考慮して，p_{i2} は x_i と $\bar{a}_2$ との積を表すものとする．S_{ij} は部分積を表す．最後に a_2 を加えているのは，$a_2 = 1$，すなわち積定数が負である場合に被乗算データの 2 の補数を得るためである．式 (1) 中の縦線は，桁サイズ $N = 2$ としたときに桁の区切りを示す．

式 (1) が示すように，桁直列乗算は基本的には人間が筆算で乗算を行う場合と同じ方式で演算を行う．データが負の場合を考慮して，新しい部分積を計算する際に現在の部分積の符号拡張を行う．式 (1) 中で四角で囲まれた項が符号拡張によって生成されたビットを表し，部分積の最上位ビットを複製することで得られる．

データ語長 $W = 4$ ビット，桁サイズ $N = 2$ ビットの桁直列定数乗算回路を図 5 に示す．被乗算データは，1 クロックあたり 1 桁ずつ最下位桁から最上位桁へ順次連続して入力する．被乗算データは W ビットであるが，積定数のビット数は任意であり，積定数には桁の概念は適用しない．

図において，最上部の全加算器 (FA) の入力部にあるスイッチによって符号拡張を行う．スイッチ切り替えのタイミングは，加算回路の場合と同様である．4 ビット被乗算データと 3 ビット積定数がともに符号付き数の場合，乗算結果は 6 ビットとなるが，被乗算データと同じ 4 ビットのみが出力される．被乗算データの最下位桁を入力する時刻がクロック t のとき，乗算結果の最下位桁はクロック $t+3$ に出力される．すなわち，積定数ビット数に等しいクロック数の遅延が生じる．

桁直列乗算回路では，AND ゲート，スイッチおよび N 段の全加算器からなるパスが最長遅延となる．

2.4 桁直列演算回路と再構成

図 4 に示す加算回路と図 5 に示す乗算回路の比較より，桁直列乗算回路は複数の桁直列加算回路を含むことが分かる．したがって，簡単な配線の繋ぎ替えによって，乗算回路中の加算回路を，純粋な加算を行うために利用することができる．

配線遅延時間を考慮して，被演算データのソースから演算器入力への配線を最短とするため，演算器の入力部に被演算データを記憶するレジスタを置く．すなわち，まず被演算データは演算器入力部のレジスタに記憶され，その後に演算を行う．また，演算結果は演算器出力部のレジスタに記憶する．これに

より，演算遅延と配線遅延を分離し，クロック周期を短縮するとともに，配線遅延を考慮して演算実行時刻を決定するスケジューリングなどの上流設計を行うことが可能となる．

桁サイズ $N = 3$ ビットの再構成可能な桁直列演算回路の基本演算回路を図6に示す．図でMUX1とMUX2はマルチプレクサであり，制御信号によって基本演算回路が加算回路として動作するか，乗算回路の一部として動作するかを切り替える．加算回路として動作する場合には，演算データAとBをそのまま全加算器に入力する．乗算回路の一部として動作する場合には，演算データAと積定数の1ビット a との積，および符号拡張を考慮して左隣の基本演算回路の出力を全加算器に入力する．加算回路の最上位ビットを分担する場合には，演算データAの1の補数をとるとともに，最下位ビットの全加算器の桁上げ入力 q には積定数を入力することで，演算データAの2の補数が得られるようになっている．演算データAの2の補数を得る機能を利用して，加算回路によって減算を実行することもできる．

マルチプレクサの制御信号によって，基本演算回路が加算回路であるか乗算回路の一部であるかを切り替える，すなわち演算回路の再構成ができる．この再構成は基本演算回路ごとに行えるので，再構成を行うタイミングを適切に制御することで，動的な再構成が可能である．

なお，図には示していないが，AとBの位置にはマルチプレクサを置き，いくつかのデータの中から演算に使用するデータを選択できるようにする．マルチプレクサの構成は，この演算回路上で実行するアプリケーション(処理アルゴリズム)に応じて決定する必要がある．

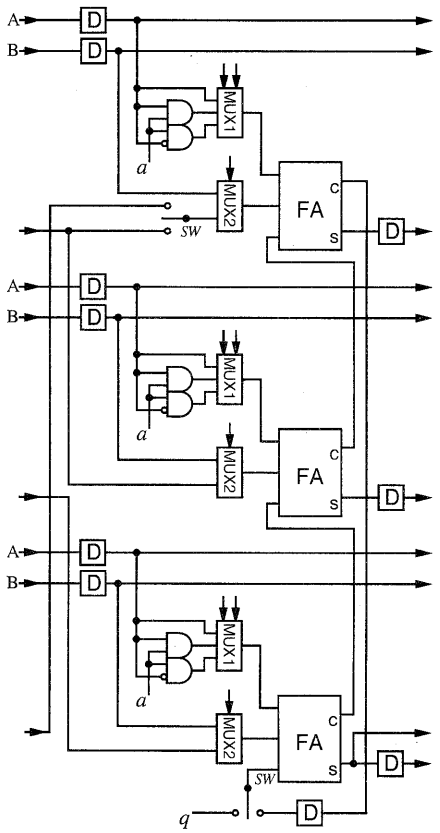

図 6. 再構成型桁直列基本演算回路(桁サイズ $N = 3$ ビット)

3 離散コサイン変換回路

画像符号化において画質を大きく損なうことなくデータ量を低減するデータ変換として，離散コサイン変換(DCT)がJPEGやMPEGなどで用いられている．例えばMPEGでは，画像を8画素×8画素のブロックに分割し，ブロック(予測フレームでは動き補償後のブロック)対して水平方向に8回，垂直方向に8回の8ポイント一次元DCTを施す．高画質動画では，リアルタイム処理実現のため大量のDCTを高速に実行する必要があり，DCT専用回路が開発されている．

図7にDCT演算アルゴリズム[13]を示す。このアルゴリズムは，乗算回数が最少になるように工夫さ

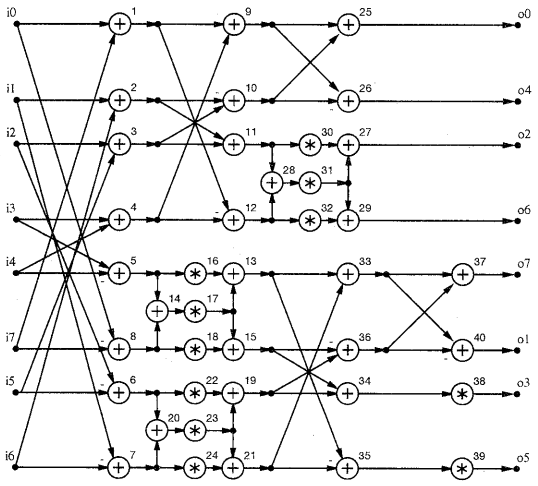

図 7. 離散コサイン変換(DCT)アルゴリズム

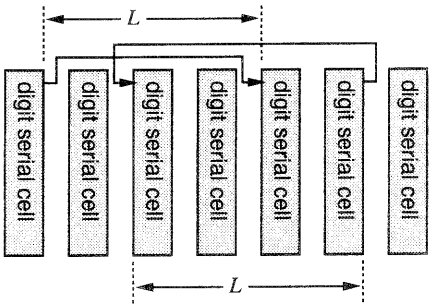

図 8. 1クロック間にデータ通信可能な距離

れている．しかし，演算の実行順序や演算間のデータ依存関係に単純な規則性がない．そのため，ハードウェア実現した場合には，演算器間で頻繁にデータ通信を行う必要があり，データ通信時間が比較的大きな超微細プロセスのLSIには適していないと考えられる．

そこで，動的再構成によってデータ通信時間を低減し，桁直列演算によって高速なクロックを実現して高速にDCTを行う回路を考案する．データ語長 $W = 16$ ビットとし，桁サイズ $N = 4$ ビットとする．

3.1 最長データ通信可能距離

長距離の配線を用いたデータ通信では配線遅延時間が大きいことを考慮して，1クロック周期間にデータを通信できる距離 L を定める．図8に桁直列演算回路と L の関係を示す．図8において'digit serial cell'は，図6に示した基本演算回路を表す．ただし，図6には桁サイズ $N = 3$ ビットの場合を示しているが，DCT回路では $N = 4$ に拡張した基本演算回路を用いる．図6に示すように，各基本演算回路は左端に入力端子があり，右端に出力端子があるものとする．入出力端子の位置を考慮して，1クロック周期間にデータを通信できる距離 L は，右方向へデータ通信する場合は間に3個の基本演算回路を含む距離，左方向へデータ通信する場合は間に2個の基本演算回路を含む距離とする．L 以下の距離のデータ通信はすべて1クロックで実行可能であり，L を越える距離の場合にはデータ通信に2クロック以上を要するとする．

3.2 小数点位置の考慮

図7のDCT演算アルゴリズムでは，絶対値が1以上2未満の積定数が用いられる．そこで，16ビット長の演算データは，整数部が11ビット(符号を含む)，小数部が6ビット，16ビット長の積定数は整数部が2ビット(符号を含む)，小数部が14ビットとする．16ビット×16ビット乗算結果の積は31ビットとなる

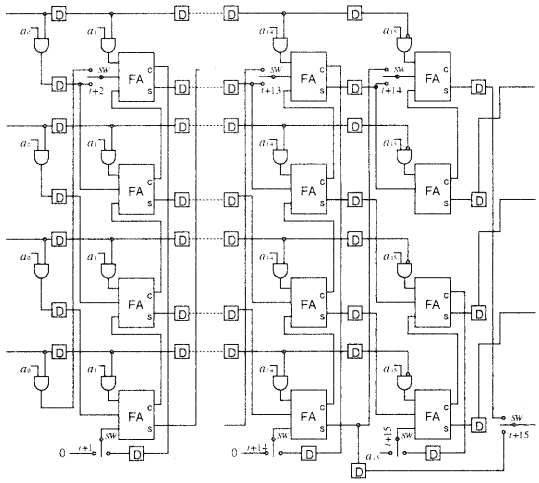

図 9. 固定小数点位置を調整した桁直列乗算回路

が，そのうち桁直列乗算では16ビットのみを出力する．乗算結果の小数点位置を演算データと一致させるには，31ビットの積の最上位ビットを破棄し，最上位ビット側から2番目のビットから17番目のビットまでを桁直列に出力する．これを実現する桁直列乗算回路を図9に示す．

この乗算回路では，被乗算データの最下位桁を入力してから積の最下位桁が出力されるまでの遅延クロック数は16である．また，加算は小数点の位置によらず加算データの最下位桁を入力してから和の最下位桁が出力されるまでの遅延クロック数は1である．

3.3 基本演算回路の配置とスケジュール

3.1節で定義した最長データ通信可能距離と，3.2節で求めた演算回路の遅延クロック数に基づいて，基本演算回路の数を定め，演算実行スケジュールを導出した．基本演算回路の数とDCT処理完了に要する時間はトレードオフとなるが，ここでは基本演算回路の数を34とし，パイプライン処理によって36クロック毎に新しいDCT処理を開始できる構成とした．

演算実行スケジュールを図10に示す．図10において横軸は時刻(クロック)，縦軸は基本演算回路の位置を表す．図10の上から下への向きは，図8の左から右への向きに相当する．

図10中の灰色の長方形は加算実行を表す．$W = 16$, $N = 4$ の桁直列加算は，1個の基本演算回路を用いて4クロックで実行する．すなわち，1回の加算実行は，1個の基本演算回路を4クロック間占有する．また，白い階段状の図形は乗算実行を表す．$W = 16$, $N = 4$ の桁直列乗算は，16個の基本演算回路を用い

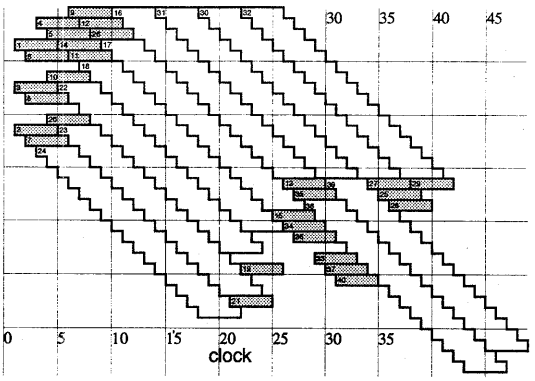

図 10. 動的再構成型桁直列演算による DCT 実行スケジュール

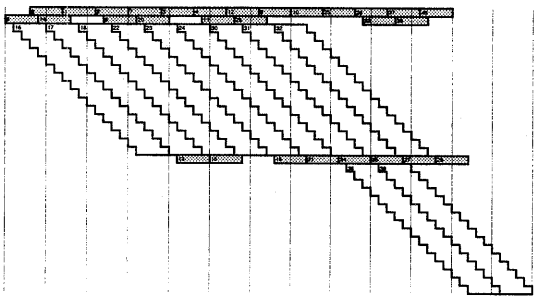

図 11. 再構成のない桁直列演算による DCT 実行スケジュールの一例

るが，1 回の乗算実行が各基本演算回路を占有する時間は 4 クロックのみである．

図 10 の演算実行スケジュールは，全てのデータ通信について 1 クロック間にデータが転送される距離が最長データ通信可能距離 L 以下となっている．また，例えば上端 (図 8 では左端) の基本演算回路は，再構成によって加算回路となったり，乗算回路の一部となったりする．そして，再構成は，他の部分での演算と並行して実施される動的再構成となっている．

比較として，再構成を行わない桁直列演算による DCT の演算実行スケジュールを図 11 に示す．動的再構成を利用する場合と比べて，基本演算回路 (ただし再構成機能なし) は多く，また DCT 完了に要するクロック数も多い．

4 LSI 設計結果

前節で設計した DCT 回路を CMOS 1.2μm ルール，2 層ポリシリコン (うち 1 層のみ使用)，2 層金属配線プロセスの LSI に試作した．図 12 にレイアウトを示す．縦約 2.95mm，横約 4.29mm，面積 12.66mm^2 となっている．この回路は，DCT の演算を行う部分のみであり，スイッチやマルチプレクサの制御信号を生成する回路は含んでいない．

Hspice 回路シミュレータを用いて遅延時間を評価したところ，基本演算回路の最長遅延時間は約 43nS，最長データ通信時間は約 24nS である．データ通信時間には，図 6 の A 点，B 点におけるマルチプレクサの遅延時間も含んでいる．演算とデータ通信を切り分け，演算を行うクロックとデータ通信を行うクロックを別にしたことで，クリティカルパスを短縮できたことが分かる．

配線遅延の大きさを評価するため，作為的にデータ通信距離を延長して遅延時間を調べたところ，データ通信距離を 10 倍程度まで延ばしてもデータ通信時間はほとんど変化しなかった．これは，1.2μm プロセスでは，配線遅延が演算遅延に比べて極めて小さいためであると考えられる．回路シミュレーションでは，配線負荷は静電容量のみであり，静電容量を MOS トランジスタで充放電する遅延モデルを用いているが，より微細なプロセスでは，配線の抵抗も考慮した遅延モデルがより現実的であり，配線遅延が相対的に増大し，提案回路の有効性が顕著になると予想される．

なお，図 12 より，最長データ通信可能距離 L を基本演算回路 3 列分とした仮定は，桁サイズ $N = 4$ ビットの場合には悲観的過ぎることが分かる．図 12 のレイアウトに従えば，基本演算回路の高さと同程度の距離のデータ通信が 1 クロック周期内に行えるはずである．基本演算回路の高さと同程度の距離は，基本演算回路 16 列分程度に相当する．この距離に基づいて基本演算回路の配置と演算実行スケジュールを決定すべきである．

5 まとめ

桁直列演算は再構成が容易であることに着目し，動的再構成型桁直列演算によって離散コサイン変換を実行する回路を設計した．動的再構成によって桁直列演算の基本演算回路を有効に利用するとともに，長大なデータ通信に要する遅延時間を演算遅延時間と分離し，コンパクトかつ高速な回路を実現できることを確認した．しかし，LSI 試作で用いたプロセスが 1.2μm であるため，動的再構成によるデータ通信時間の低減を確認することはできなかった．

桁直列演算では，スイッチやマルチプレクサの切り替えのため多くの制御信号を必要とするが，制御信号の通信時間を考慮して制御信号を生成する回路

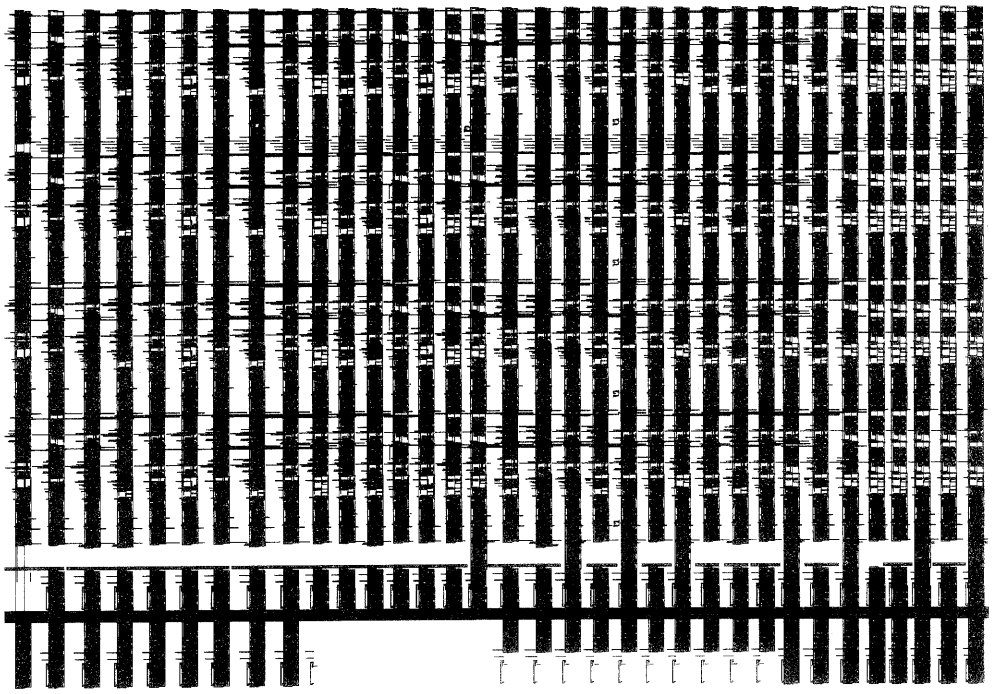

図 12. 1.2μm プロセスによる DCT 回路

を考案する必要がある．また，与えられた処理アルゴリズムに対して最適な基本演算回路の配置と演算実行スケジュールを得る手法の自動化，より微細なプロセスにおける提案回路の評価などが今後の課題である．

謝辞 本研究の一部は，文部省科学研究補助金: 奨励研究 (A) 課題番号 11750303 による．本チップ試作は東京大学大規模集積システム設計教育研究センターを通しオンセミコンダクター (株)、日本モトローラ (株)、大日本印刷 (株)、および京セラ (株) の協力で行われたものである。

参考文献

[1] 山品, "サブクオータミクロン時代の LSI 設計技術," **信学技法**, vol. VLD95-136, pp. 53–60, 1996.

[2] T. Sakurai, "Outline of System LSI: An Introduction to System LSI's — Applications and Issues," *Journal of IEICE*, vol. 81, pp. 1083–1086, Nov. 1998.

[3] K. Ito, "A Scheduling and Allocation Method to Reduce Data Transfer Time by Dynamic Reconfiguration," in *Proc. ASPDAC 2000*, Yokohama, pp. 323–328, 2000.

[4] T. Sueyoshi, "Reconfiurable Logic," *Journal of IEICE*, vol. 81, pp. 1100–1106, 1998.

[5] P. Lysaght and J. Stockwood, "A Simulation Tool for Dynamically Reconfigurable Field-Programmable Gate Arrays," *IEEE Trans. VLSI Systems*, vol. 4, pp. 381–390, Sept. 1996.

[6] 末吉, "Reconfigurable Computing System の現状と課題 — Computer Evolution へ向けて," **信学技報**, vol. VLD96-79, pp. 111–118, 1996.

[7] XILINX, *XC6200 Field Programmable Gate Arrays Data Sheet*, 1997. http://www.xilinx.com.

[8] M. Ito, J. Kitamichi, and N. Funabiki, "A Design of Dynamically Reconfigurable FPGA and An Implementation of Parallel Algorithm on it," *IPSJ SIG Notes*, vol. 98DA88, pp. 1–8, 1998.

[9] 鎌田, 伊藤, "動的再構成による高速信号処理," **信学技報**, vol. CAS98–102, pp. 33–40, 1999.

[10] 上田他, "動的再構成可能回路を用いた新しい電磁粒子シミュレーション回路の設計," **情処学設計自動化研報**, vol. 98DA88, pp. 1–8, 1998.

[11] 中根他, "自律再構成可能 FPGA におけるメッセージ自己ルーティングのためのセルラー・アルゴリズム," in **第 *11* 回回路とシステム *(軽井沢)* ワークショップ**, pp. 199–204, 1998.

[12] K. K. Parhi, "A Systematic Approach for Design of Digit-Serial Processing Architectures," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 358–375, Apr. 1991.

[13] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical Fast 1-D DCT Algorithms with 11 Multiplications," in *Proc. IEEE ICASSP*, pp. 988–991, 1989.