システムＬＳＩ設計技術　94－14
(2000. 1. 12)

# 通信時間を考慮したスケジューリングと素子配置

中野 正喜　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: {masaki,kazuhito}@ees.saitama-u.ac.jp

あらまし　集積回路製造技術の向上により回路の微細化が進んでいる。微細化によりゲート遅延は減少するが、配線遅延は相対的に増大する傾向にある。LSI 設計において、配線遅延違反による設計の手戻りを低減して設計期間を短縮するためには、上流設計においても配置配線を意識して配線遅延を考慮した設計を行う必要がある。本研究では、与えられた処理アルゴリズムを実行する専用 LSI の設計において、演算の実行時刻決定および素子 (演算器およびレジスタ) 割り当てと配置を同時に行うことで、素子間の正確な配線遅延を考慮しながら演算実行時刻を決定し、高速処理実行速度を達成する手法を提案する。

キーワード　LSI 設計、演算スケジューリング、素子配置、配線遅延

# Scheduling and Placement of Operations Considering Data Communication Time

Masaki Nakano　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**　With the development of deep submicron technology, wire delay on an LSI chip is becoming relatively larger than gate delay. In LSI design, it is necessary to turn around the design if delay violates speed requirement. In order to reduce the turn around, wire delay determined by placement and routing must be considered even in high-level design. In this paper we propose a method to achieve high speed processing for a given processing algorithm by performing scheduling and placement of operations simultaneously. With this method wire delay based on the placement is precisely considered and may be minimized during the scheduling of operations.

**Keywords**　LSI design, operation scheduling, placement, wire delay

# 1 はじめに

集積回路製造技術の向上により回路の微細化が進んでいる。微細化によりゲート遅延は減少するが、配線遅延は相対的に増大する傾向にある [1]。ある0.5$\mu$m デザインルールの集積回路において配線による容量負荷を考慮してシミュレーションによって遅延時間を調べたところ、16 ビット×16 ビット並列乗算器 (非パイプライン) の遅延時間と、同じ乗算器の1辺長と等距離の配線遅延時間の比は約 10 対 1 である。ただし、乗算器のトランジスタサイズの考慮は特に行っていないので、トランジスタサイズ最適化によって乗算器の遅延時間は短縮する可能性がある。一方、ここでは配線の抵抗は考慮しておらず、実際には配線遅延はより大きいと考えられる。今後さらに微細化が進むことにより、配線遅延時間が全遅延時間に占める割合がますます大きくなると考えられる。

ゲートあるいは演算器の遅延時間を考慮して要求クロック周波数を達成できるようにLSI上流設計や論理合成を行うが、配置配線後に得られる実際の配線遅延時間を含めた結果、要求クロック周波数を達成できていない場合がある。その場合には、論理合成やさらには上流設計に戻って特定部分のゲートや演算器の遅延時間を減少し、配置配線をやり直すという設計の手戻りが発生する。これは、上流設計や論理合成において、全遅延時間に占める割合が相対的に小さなゲート遅延時間のみを考慮するだけでは、全遅延時間を制御することが困難であることを意味している。設計の手戻りを低減して設計期間を短縮するためには、上流設計においても配置配線を意識して配線遅延を考慮した設計を行う必要がある。

ディジタル信号処理を代表とする乗算と加算からなる処理アルゴリズムを実行する専用LSIの設計では、演算の実行時刻決定、演算器とレジスタの割り当て、演算器とレジスタの配置、および素子 (演算器とレジスタ) 間配線といった処理が必要である。演算実行時刻決定処理では、演算間データ依存関係による先行制約関係を満足するように演算の実行時刻を決定する。演算間のデータ通信はLSI上の配線を用いて行われるので、配線遅延に由来するデータ通信時間も考慮して演算実行時刻を決定しなければならない。ところが、データ通信時間はどの演算がどの演算器に割り当てられて実行されるか、また演算器がどのように配置されるかに依存しており、配置配線が終了しなければ正確な配線遅延時間、すなわちデータ通信時間は得られない。一方、演算の実行時刻が決定されないと、演算割り当てができず、演算器配置も行うことはできない。このように、演算実行時刻決定と配置配線が相互に影響しており、いずれかを先に行うことは、設計結果の性能低下を招く。

そこで本研究では、乗算と加算からなる処理アルゴリズムを実行する専用LSIの設計において、演算の実行時刻決定および演算器割り当てと演算器配置を同時に行うことで、素子 (演算器およびレジスタ) 間の正確な配線遅延を考慮しながら演算実行時刻を決定し、高速処理実行速度を達成する手法を提案する。

# 2 ハードウェアモデル

## 2.1 配線遅延

LSIの配線は、半導体基板上に絶縁膜を介して金属細線を敷くことで実現されており、半導体基板と金属配線間に静電容量を形成する [2]。CMOS 回路では、配線の静電容量を充電あるいは放電することによって前段の出力から次段の入力へ信号が伝えられる。この充放電に要する時間が配線遅延時間となる。配線が長いほど静電容量は大きくなって、充電あるいは放電に長時間を要することとなり、配線遅延時間が長くなる。

実際には、配線は容量、抵抗、インダクタンスからなる分布定数線路となり、厳密な遅延時間は配線距離、配線形状などの複雑な関数になると考えられるが、本研究では、配線遅延時間は入出力端子間のマンハッタン距離に比例するものとする。

## 2.2 演算器およびレジスタ

加算器、乗算器は組合せ回路として構成されており、内部にはレジスタを持たないものとする。入力データは演算器外部のレジスタから供給し、演算が完了するまで値を変化させてはならない。また、演算結果の記憶が必要な場合には、演算器外部のレジスタに記憶する。演算器はパイプライン化されておらず、演算が完了するまでは同一演算器上で次の演算を開始することはできない。

以降では、演算器とレジスタの総称を素子とする。素子の形状は長方形であり、長方形の1辺の長さ $d$ はデータビット長によって決まり、全ての素子で共通とする。他辺の長さは、素子の種類によって異なり、レジスタ $a$、加算器 $b$、乗算器 $c$ とする。最小と考えられるレジスタの辺長 $a$ を単位として、加算器、乗算器の辺長 $b$、$c$ は、レジスタの辺長の整数倍であるとする。図1に示す素子形状では、共通辺の長さは $d$、多辺の長さは、$a$:$b$:$c$ の比が 1:3:8 となっている。

素子の入出力端子は共通長の辺上にあり、入力端子と出力端子は互いに他の辺上に位置する。

素子の入力には複数の配線から信号を選択するた

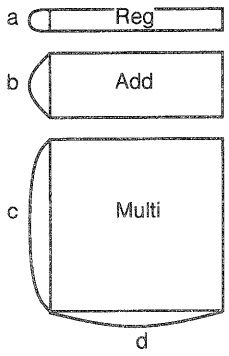


図 1. 素子の形状

めのマルチプレクサが必要となる場合があるので、その配置領域を確保するため、2 つの素子が隣接して配置する場合には、あらかじめレジスタの辺長 $a$ に等しい幅を空けるものとする。

多層配線により、素子内部のゲート間配線と素子間配線に異なる層を用いることで、素子間配線が素子上を通過することを許すものとする。

# 3 素子配置とスケジューリング

## 3.1 設計の方針

高速な処理速度を達成するためには、配線による遅延時間を低減することが重要である。データ通信が必要な素子同士を隣接して配置すれば、配線遅延時間が最短化され、データ通信時間を低減することができる。ところが、データ通信を行うすべての素子を隣接して配置可能とは限らない。そこで、データ通信時間を最短化することが処理速度向上に大きく寄与するクリティカルパスについて、その演算を優先的に近傍に配置する。

図2に示すようなデータフローグラフ (DFG)[3] によって表現された処理アルゴリズムにおいて、ノードは演算を表し、ノード間の有向枝は演算間のデータ通信を表す。DFG 中のパス上の演算は、データ通信が必要な演算の集合であり、ノード数すなわち演算数の多いパスはデータ通信も多い。したがって、演算数が多く演算時間総和の大きいパスがクリティカルパスとなる可能性が高い。

DFG 上のループは、始点と終点が一致したパスである。ループ上の演算を同一演算器あるいは隣接演算器に割り当てることは、データ通信時間最短化の効果が大きいと考えられる。一般に DFG には多数のループが存在するが、演算時間総和の大きなループから順に優先してループ上の演算を同一演算器あるいは隣接演算器に割り当て、演算実行時刻を決定することで処理速度の向上を図る。

なお、複数のループが演算を共有している場合には、あるループの一部の演算が他のループとして演算器割り当てと実行時刻決定が行われることがある。そのため、ループの優先順を考慮して、より優先順位が高い他のループに共通に所属しない演算を演算群と定義する。すなわち、最も優先順位が高いループでは、その全ての演算が 1 つの演算群に含まれる。以降のループについては、まだいずれの演算群にも含まれていない演算が 1 つの演算群となる。

## 3.2 演算群の選択

優先的に演算器に割り当てて配置する演算群としては、演算時間総和が大きく、データ通信時間を最短化することが処理速度高速化に有効であるものが好ましい。そこで、処理アルゴリズムを表した DFG 中のループに注目し、以下の式によって算出されるループ $L$ のループ下限 $\mathrm{LB}_L$ を求める。

$$\mathrm{LB}_L = \frac{Q_L}{D_L} \tag{1}$$

ここで、$Q_L$ はループ $L$ 上の演算の演算時間総和、$D_L$ は $L$ 上の総遅延数である。ループ下限 $\mathrm{LB}_L$ は、処理アルゴリズムを 1 回実行する間に実行すべき演算の時間を表す。したがって、処理アルゴリズムの繰り返し実行を高速化するには、$\mathrm{LB}_L$ が大きなループ $L$ ほどループ上のデータ通信時間を最短化すべきである。

与えられた DFG からループおよび演算群を選択する手順は以下の通りである。まず、DFG 内でループ下限最大のループを選び、そのループ上の全ての演算を 1 つの演算群とする。この演算群の優先順を 1 とし、$G_1$ と表す。次に、いずれの演算群にも属さない演算を少なくとも 1 つ以上含み、ループ下限最大のループを選ぶ。このループ上の演算のうち、いずれの演算群にも属さない演算を 1 つの演算群 $G_k$ ($k \geq 2$) とする。これを $k$ を 1 ずつ増やしながらすべての演算がいずれかの演算群に属するようになるまで繰り返す。

図 2に示す 5 次ウェーブディジタル楕円フィルタについて、演算時間が、加算 1 単位時間、乗算 2 単位時間であるとして演算群を求めた結果を以下に示す。まず、ループ下限最大のループとして

$$L_1 = \{10, 16, 21, 17, 15, 9, 7, 6, 8, 12, 13, 14, 11\}$$

を選ぶ。このループ $L_1$ 上の 13 個の演算が演算群 $G_1$ に属する。

次に、$G_1$ に属さない演算を少なくとも 1 つ含むループのうち、ループ下限最大のループは

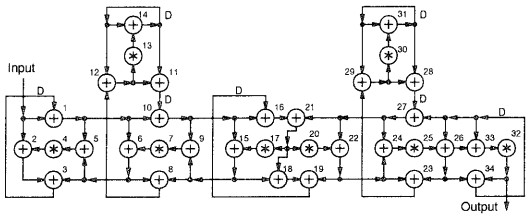


図 2. 5次ウェーブディジタル楕円フィルタ

$L_2 = \{1, 10, 16, 21, 17, 15, 9, 7, 6, 5, 4, 2, 3\}$

である。演算群 $G_1$ に属さない演算 $\{1, 6, 5, 4, 2, 3\}$ が演算群 $G_2$ に属する。

結局、以下のループ

$L_1 = \{10, 16, 21, 17, 15, 9, 7, 6, 8, 12, 13, 14, 11\}$
$L_2 = \{1, 10, 16, 21, 17, 15, 9, 7, 6, 5, 4, 2, 3\}$
$L_3 = \{27, 21, 20, 22, 24, 25, 26, 23, 29, 30, 31, 28\}$
$L_4 = \{27, 21, 20, 22, 24, 25, 26, 33, 32, 34\}$
$L_5 = \{16, 21, 17, 15, 18, 19\}$

が順に選び出され、以下の5つの演算群 $G_1, G_2, \ldots, G_5$ が得られる。先に記された演算群が、より優先順が高い。

$G_1 = \{10, 16, 21, 17, 15, 9, 7, 6, 8, 12, 13, 14, 11\}$
$G_2 = \{1, 6, 5, 4, 2, 3\}$
$G_3 = \{27, 20, 22, 24, 25, 26, 23, 29, 30, 31, 28\}$
$G_4 = \{33, 32, 34\}$
$G_5 = \{18, 19\}$

### 3.3 基本演算セル

3.2節に述べる方法でDFGから選び出した演算群は、演算間データ通信を最短化すべき演算の集合である。したがって、同一演算群に含まれる演算が、隣接する加算器と乗算器の対に割り当てられることによってデータ通信時間が最短化されることが好ましい。

そこで、図3に示すように予め素子配置を固定した基本演算セルを用意して、その演算器に演算群の演算を割り当てる。基本演算セルでは、中央のレジスタから出力されたデータが加算器あるいは乗算器に入力され、演算結果が元のレジスタに戻るとしている。レジスタ出力から演算器入力までのデータ通信時間および演算器出力からレジスタ入力までのデータ通信時間も考慮して、データが正しくレジスタに読み込まれるようにクロック周期を選ぶとともに、レジスタのデータ読み込みタイミングを制御する。

レジスタを出力したデータを用いた演算結果が元のレジスタに戻るので、演算器とレジスタが隣接していることが望ましい。そこで、基本演算セルでは、レジスタが乗算器と加算器の双方に隣接できるように、中央にレジスタを配置した構造となっている。

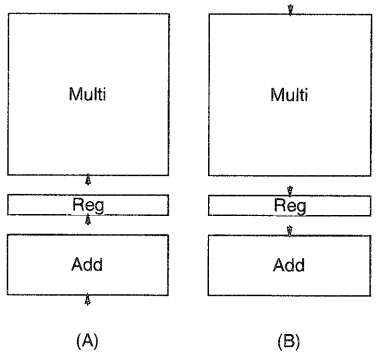


図 3. 基本演算セル

データ通信時間最短化のため、基本演算セル内ではレジスタ入力と一方の演算器の出力、レジスタ出力と他方の演算器の入力が隣接する向きに素子を配置する。したがって基本演算セルには、データ通信の方向によって図3に示す2種類が考えられる。

演算群によっては、演算の種類がすべて加算のみあるいは乗算のみとなるが、その場合には基本演算セルは使用せず、加算器とレジスタあるいは乗算器とレジスタのみを使用して演算を割り当てる。

### 3.4 演算実行時刻決定およびレジスタ割り当て

演算群の演算を基本演算セルあるいはその他の演算器に割り当てることでデータ通信時間が確定するので、as soon as possible (ASAP) スケジューリング [4] によって、演算実行時刻を決定する。もしも現在想定している処理アルゴリズムの繰り返しクロック周期数 $T$ では、全ての演算間先行制約を満足するように演算実行時刻を決定できない場合には、$T$ を1だけ増やして演算器割り当て、演算器配置、演算実行時刻決定を最初からやり直す。

このとき、演算 $i$ から $j$ に先行制約関係があり、演算 $j$ の実行可能時刻範囲 (スケジューリング・レンジ [3]) に余裕があるにも関わらず演算間先行制約を満足するように演算 $i$ の実行時刻を決定できない場合は、最初から設計をやり直したとしても再度同じ状況になることが予想される。したがってこの場合には、次の演算実行時刻決定の際に $j$ の実行時刻を強制的に1クロック周期だけ遅らせる。

ある演算結果を使用する演算が遅れて実行されるなど、長時間保持が必要なデータは基本演算セル中央のレジスタには保持できない。その場合には別途

レジスタを用意し、データ通信時間を考慮して基本演算セルの近傍に配置する。

### 3.5 素子配置ルール

以上述べた点を考慮して、素子配置を簡単化するため以下のルールを用いる。

- 基本演算セルを新たに配置する際には、すでに配置されている基本演算セルと同一の列には配置しない。ここで列とは、基本演算セル内で素子が並んでいる方向であり、図3では上下方向となる。すなわち、図3の上下方向には複数の基本演算セルを配置しない。

- 演算群 $G$ 内の演算が、すでに基本演算セル $C'$ に割り当て済みの演算群 $G'$ 内の演算との間にデータ通信を行う場合に、演算群 $G$ を割り当てる基本演算セル $C$ を配置する際には、$G-G'$ 間データ通信時間が最短になるように基本演算セルを2種類のうちから選択し、$C$ を列方向に適宜反転するとともに $C'$ に対して最適な相対位置に $C$ を配置する。

  例えば、基本演算セル $C'$ に割り当てられた演算群 $G'$ 内の加算との間にデータ通信を行う演算を含む演算群 $G$ が、基本演算セル $C$ に割り当てられるとする。このとき図4に示すように基本演算セル $C$ を列方向に反転するとともに $C'$ と $C$ の位置関係を調節して加算間データ通信時間が最短になるようにする。

- レジスタを配置する際には、レジスタに記憶するデータを生成する演算が割り当てられている演算器と同じ列に配置する。

### 3.6 アルゴリズム

素子配置と演算実行時刻決定アルゴリズムを以下に示す。

1. 処理アルゴリズムの最短繰り返し周期 $T$ を求める。

2. 優先順が最も高い未処理演算群 $G$ を選ぶ。未処理演算群が1つもなければ終了する。

3. $G$ に属する演算を配置済みの演算器に割り当てて実行可能か調べる。配置済みの演算器に割り当て可能な場合は、割り当てを行い、ステップ7へ進む。

4. 演算群 $G$ の演算種類が1種類(加算のみあるいは乗算のみ)の場合は次のステップへ進む。さもなければ、新たに基本演算セル $C$ を用意し、演算群 $G$ を割り当てる。基本演算セル $C$ を3.5節のルールにしたがって配置する。

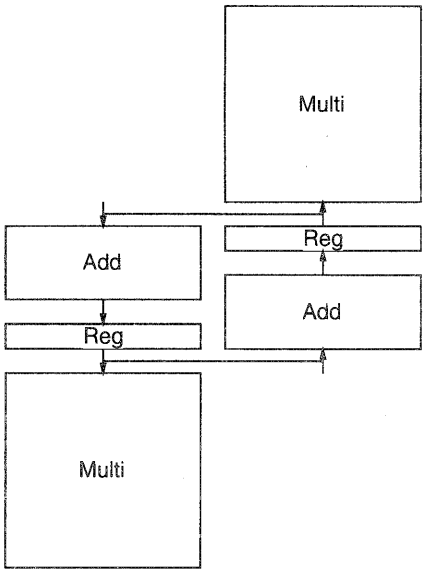


図 4. 加算間データ通信がある基本演算セルの配置

5. 演算群 $G$ の演算種類が1種類の場合は、その演算種類の演算器を用意し、データ通信時間が最短となる位置に配置する。この際に、既配置の基本演算セルと同一の列に配置することを許す。

6. 演算群 $G$ に属する演算の実行時刻を決定する。このとき、すでに処理済みの演算群に属する演算の実行時刻および演算間データ通信時間を考慮する。処理済みの演算から演算群 $G$ に属する演算へのデータ通信時間が長い場合には、必要に応じてクロック周期を割り当てて演算実行時刻を決定する。

7. 演算群 $G$ の演算の演算結果データを保持するため、データ保持期間をレジスタに割り当てる。演算群が基本演算セルに割り当てられる場合は、その中央のレジスタに割り当てる。データを2クロック周期以上保持する必要がある場合、および演算群 $G$ が加算器のみあるいは乗算器のみに割り当てられる場合には、データ通信時間を考慮してすでに配置済みのレジスタに割り当てるか、あるいは新たにレジスタを用意して適当な位置に配置し、そこにデータ保持期間を割り当てる。

表 1 演算時間および配線遅延時間

| | |
|---|---|
| 加算 | 30 単位時間 |
| 乗算 | 100 単位時間 |
| レジスタ | 7 単位時間 |
| 配線遅延 | 7 単位時間 |

8. 通信時間を考慮した結果、現在の繰り返し周期 $T$ ではすべての演算先行制約を満足するように演算実行時刻を決定できない場合は、繰り返し周期を 1 クロック周期だけ増やし、演算割り当てと素子配置をすべて破棄してステップ 2 へ戻る。

## 4 実験結果

提案する手法を用いて、処理アルゴリズムを高速に実行する回路の設計を行った。演算時間と配線遅延時間は、表 1に示すように仮定した。ここで、配線遅延時間は、レジスタの 1 辺 $a$ に等しい距離の遅延時間である。また素子の大きさは、$a:b:c:d=1:3:8:8$ とした。レジスタはデータ出力およびデータ入力の合計が 7 単位時間とする。この仮定により、基本演算セルについて、レジスタを出力したデータが乗算を経由してレジスタの入力に到達するまでを 2 クロック周期以内に行うことができる。

まず、図 2に示す 5 次ウェーブディジタル楕円フィルタ (WEF) について設計を行った結果を図 5, 6, 7 に示す。例えば図 5は、演算 1 が加算器 A3 において時刻 0 に実行されることを表している。また、図 6 は、加算 1 の結果を時刻 1 から時刻 11 までレジスタ R8 に記憶することを表している。図 5に示されるように、クロック周期 100 単位時間で、繰り返し周期は 16 クロック周期である。したがって、乗算器 4 個、加算器 4 個、レジスタ 11 個を用いて、処理アルゴリズムを 1600 単位時間ごとに繰り返し実行することができる。

比較として、配線遅延を考慮せずに演算器数を最少化するように演算実行時刻決定および演算器とレジスタの割り当てを行った結果に基づいて、演算器割り当てを変更せずに最適な素子配置を行った結果を図 8, 9, 10に示す。素子数は、乗算器 2 個、加算器 3 個、レジスタ 11 個である。素子配置を行った結果、クロック周期は同じ 100 単位時間であるが、データ通信のために余分なクロック周期を必要とし、繰り返し周期は 20 クロック周期、処理アルゴリズムの繰り返しは 2000 単位時間ごととなっている。

提案手法では、素子数が必ずしも最少ではないが、与えられた処理アルゴリズムを高速に実行する回路

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 | 11 | 10 | 16 | 21 | | | 15 | 9 | 18 | 19 | 6 | 8 | 12 | | | 14 |
| A2 | 28 | 27 | | | | | 22 | 24 | | | 26 | 23 | 29 | | | 31 |
| A3 | 1 | | | | | | | | | | | 5 | | | 2 | 3 |
| A4 | | | | | | | | | | | | 33 | | | 34 | |
| M1 | | | | | 17 | 17 | | | 7 | 7 | | | | 13 | 13 | |
| M2 | | | | | 20 | 20 | | | 25 | 25 | | | | 30 | 30 | |
| M3 | | | | | | | | | | | | | 4 | 4 | | |
| M4 | | | | | | | | | | | | | 32 | 32 | | |

図 5. WEF の提案手法による演算実行時刻および演算器への割り当て結果

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R1 | 14 | 11 | 10 | 16 | 21 | 21 | 17 | 15 | 9 | 9 | 7 | 6 | 8 | 12 | 12 | 13 |
| R2 | | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| R3 | 12 | 12 | | 10 | 10 | 10 | 10 | 10 | 22 | 22 | 22 | 22 | | | | 12 |
| R4 | 31 | 28 | 27 | 27 | | | 20 | 22 | 24 | 24 | 25 | 26 | 23 | 29 | 29 | 30 |
| R5 | 19 | 19 | 19 | | 27 | 27 | 27 | 27 | 15 | 18 | 19 | 19 | 19 | 19 | 19 | 19 |
| R6 | 29 | | | | | | 21 | 21 | 21 | 15 | 15 | 15 | | | | 29 |
| R7 | | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| R8 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 | 5 | 4 | 2 |
| R9 | | | | | | | | | | | | | 6 | 6 | 6 | 6 |
| R10 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 33 | 33 | 32 | 34 |
| R11 | | | | | | | | | | | | 26 | 26 | 26 | 26 | |

図 6. WEF レジスタ割り当て結果

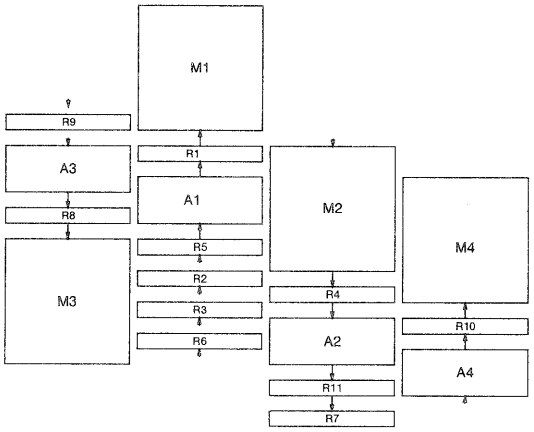


図 7. WEF 素子配置結果

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A1 | 11 | 10 | 16 | | 21 | | | | 15 | 9 | | | 6 | 8 | | 12 | | | | 14 |
| A2 | 31 | 28 | 27 | | | | | | | 22 | 24 | | | 26 | | 23 | 29 | 33 | | |
| A3 | 1 | 34 | | | | | | | | | | 18 | 19 | 5 | | | | 2 | | 3 |
| M1 | 32 | | | | | | 17 | 17 | | | 7 | 7 | | | | | 13 | 13 | | 32 |
| M2 | | | | | | | | 20 | 20 | | | 25 | 25 | | | 4 | 4 | 30 | 30 | |

図 8. 演算器数最少の演算実行時刻および演算器への割り当て結果

が得られていることが分かる。

別の例として、提案手法を図 11に示す 1 次元 8 ポ

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R1 | 14 | 11 | 10 | 16 | 16 | 21 | 21 | 21 | 17 | 15 | 9 | 9 | 7 | 6 | 8 | 8 | 12 | 12 | 13 | 13 |
| R2 | 30 | 31 | 28 | 27 | 27 | 27 | 27 | | | 20 | 22 | 24 | 24 | 25 | 26 | 26 | 23 | 29 | 29 | 30 |
| R3 | 3 | 1 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 34 | 5 | 5 | 5 | | 2 | 2 |
| R4 | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 34 | 34 | 34 | 34 | 33 | 33 |
| R5 | 12 | | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 15 | 15 | 15 | 15 | 6 | 6 | 6 | 6 | 12 | 12 |
| R6 | | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| R7 | 29 | 29 | | | | | | 27 | 27 | 27 | 27 | 22 | 22 | 22 | 22 | 22 | | | | 29 |
| R8 | 31 | | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 | 31 |
| R9 | | 32 | | | | | | | 21 | 21 | 21 | 21 | 18 | | | | | 4 | 6 | 6 |
| R10 | 19 | 19 | 19 | | | | | | | | | | | 19 | 19 | 19 | 19 | 19 | 19 | 19 |
| R11 | 26 | 26 | | | | | | | | | | | | | | | | 26 | 26 | 26 |

図 9. 図8に対するレジスタ割り当て結果

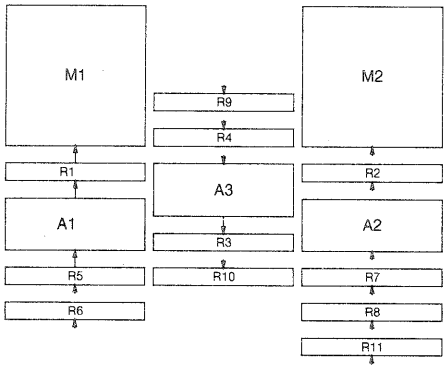


図 10. 図8に対する素子配置結果

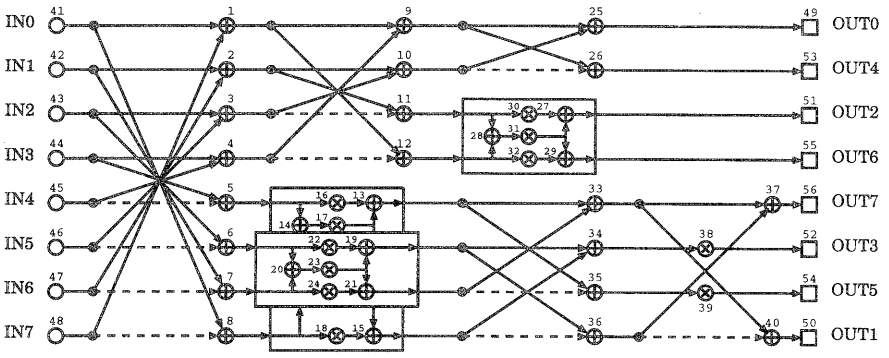


図 11. 1次元8ポイント DCT

イント離散コサイン変換 (DCT)[5] に適用した結果を図12, 14, 13に示す。本来はDCTは繰り返し処理ではないので、入出力レイテンシを最短化することを目的としている。素子数は、乗算器7個、加算器7個、レジスタ14個であり、クロック周期は100単位時間、入出力レイテンシは900単位時間である。

比較として、データ通信時間を考慮せずに入出力レイテンシを最短化し、かつ演算器数最少のスケジューリング結果に基づいて、演算器割り当てを変更せずにデータ通信時間最短化配置を行った。素子数は、乗算器6個、加算器5個、レジスタ11個であり、クロック周期は100単位時間、入出力レイテンシは1400単位時間となり、提案手法によってよりレイテンシの小さい設計結果が得られることが分かる。

以上の結果をまとめて表2に示す。

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| A1 | 5 | 14 | | | 13 | | **35** | **36** | **40** |
| A2 | 6 | 20 | | | 19 | | 34 | 33 | **37** |
| A3 | 7 | 1 | 12 | 28 | 21 | | | | 27 |
| **A4** | 8 | | | | 15 | | | | |
| **A5** | 2 | 11 | | | | | | | 29 |
| **A6** | 3 | 10 | | | 26 | | | | |
| **A7** | 4 | | | 9 | 25 | | | | |
| M1 | | | 17 | 17 | | | | **39** | **39** |
| M2 | | | 23 | 23 | | | | **38** | **38** |
| M3 | | | 16 | 16 | | | | | |
| M4 | | 18 | 18 | | | | | | |
| M5 | | | 22 | 22 | 32 | 32 | 31 | 31 | |
| M6 | | 24 | 24 | | | | | | |
| M7 | | | 30 | 30 | | | | | |

図 12. 8ポイント DCT の演算実行時刻および演算器への割り当て結果

## 5 まとめ

本稿では、配線遅延に由来するデータ通信時間を考慮して、与えられた処理アルゴリズムの実行速度を最大にするスケジューリングと素子配置の一手法を提案した。

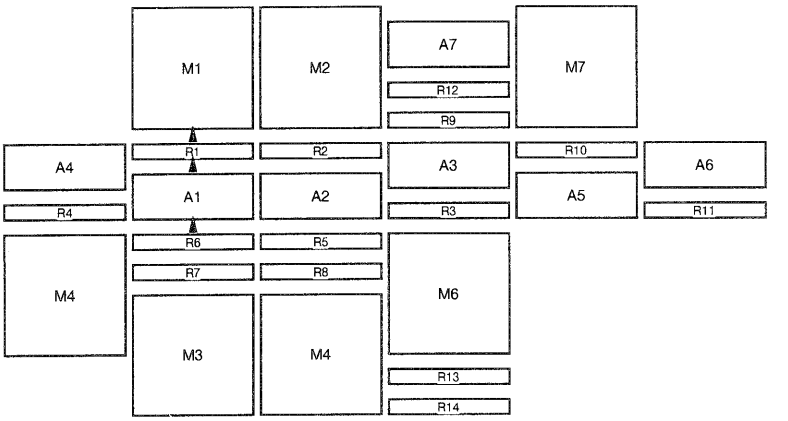


図 13. 8 ポイント DCT の素子配置結果

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| R1 | | 5 | 14 | 14 | 17 | 13 | 13 | 35 | 35 |
| R2 | | 6 | 20 | 20 | 23 | 19 | 19 | 34 | 34 |
| R3 | | 7 | 7 | 24 | 24 | 21 | 21 | 21 | |
| R4 | | 8 | 8 | 18 | 18 | 15 | 15 | 15 | |
| R5 | | | | | | | | 13 | 33 |
| R6 | | | | | | | | 19 | 36 |
| R7 | | | 5 | 5 | 16 | | | | |
| R8 | | | 6 | 6 | 22 | | | | |
| R9 | | | 1 | 12 | 28 | 28 | 28 | | |
| R10 | | 2 | 11 | 30 | 30 | 30 | 30 | | |
| R11 | | 3 | 10 | 10 | 10 | | | | |
| R12 | | 4 | 4 | 4 | 9 | | | | |
| R13 | | | | | | | | 31 | 31 |
| R14 | | | | | | 32 | 32 | 32 | 32 |

図 14. 8 ポイント DCT のレジスタ割り当て結果

5 次ウェーブディジタル楕円フィルタと 1 次元 8 ポイント DCT の例において、本手法のスケジューリングと配置を用いることで、データ通信時間を考慮しない場合に比べて実行速度の高速な設計結果が得られており、本手法の有効性が確認できた。

今後の課題としては、手法の効率化、素子数の最少化を考慮したスケジューリング、演算器割り当て、および素子配置の手法の考案と、パイプライン化された演算器への対応などが上げられる。

表 2 実験結果

| 処理アルゴリズム | 5 次ウェーブフィルタ | 8 ポイント DCT |
|---|---|---|
| 乗算数 | 8 | 11 |
| 加算数 | 26 | 29 |
| 枝数 | 58 | 69 |
| 提案手法による結果 | | |
| 繰り返し周期 | 1600 | 900 |
| 素子数 | 乗算 4, 加算 4 レジスタ 11 | 乗算 7, 加算 7 レジスタ 14 |
| 演算器最少化に基づく結果 | | |
| 繰り返し周期 | 2000 | 1400 |
| 素子数 | 乗算 2, 加算 3 レジスタ 10 | 乗算 6, 加算 5 レジスタ 11 |

## 参考文献


[1] 桜井, "総論—システム LSI のアプリケーションとシステム LSI の課題—," 信学誌, vol. 81, pp. 1083–1086, Nov. 1998.

[2] 國枝, 集積回路設計入門. コロナ社, 1996.

[3] S. M. Heemstra de Groot, S. H. Gerez, and O. E. Herrmann, "Range-Chart-Guided Iterative Data-Flow Graph Scheduling," *IEEE Trans. Circuits Syst.-I: Fund. Theory & Appl.*, vol. CAS-39, pp. 351–364, May 1992.

[4] M. C. McFarland, A. C. Parker, and R. Camposano, "The High-Level Synthesis of Digital Systems," *Proc. of the IEEE*, vol. 78, pp. 301–318, Feb. 1990.

[5] 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.