設計自動化 63－2
（1992. 7.17）

高密度データパスコンパイラのレイアウト生成方法

中尾 博臣　北田 修　林越 美智子　岡崎 芳
辻橋 良樹　松本 尚

三菱電機株式会社

あらまし　ALUやシフタ等の機能ブロック間の接続で表現したデータパスの論理図を入力として、高密度なレイアウトを生成する手法について述べる。本手法では、機能ブロックのレイアウトを、アレイコンパイラを用いて生成し、配置配線プログラムがそれらを配置して結線する。アレイコンパイラのセル上配線領域の抽出機能、ソフト／ハード混合セル生成機能を利用して、(1)機能ブロック上を配線領域として用いる、(2)機能ブロック上の配線領域が不足する場合は、機能ブロックを伸ばして必要な配線領域を確保する、ことにより、配線に必要な領域を削減し、高集積なレイアウト生成を可能にした。本手法を用いて、2901データパスを生成した結果、集積度8500トランジスタ/mm^2 （0.8μmルール）のデータパスが生成できた。

A High Density Layout Generation Method for Datapath Compiler

Hiroomi NAKAO　Osamu KITADA　Michiko HAYASHIKOSHI　Kaoru OKAZAKI
Yoshiki TSUJIHASHI　Hisashi MATSUMOTO

Mitsubishi Electric Corporation

4-1 Mizuhara, Itami, 664 Japan

Abstract　We present a method for generating high density datapath layout from a schematic diagram representing the connectivity among functional blocks. We have adopted *array-compiler* for generating the layout of the functional blocks, and a place-and-route program for generating overall layout. Two special functions of the array-compiler, routing area extraction and soft/hard mixed cell generation, enable the place-and-route program to use transistor regions of the functional blocks for routing, and to stretch the regions if needed. This has reduced the area needed for routing drastically, and led to the generation of high density datapath layout. The 2901 datapath layout whose transistor density is 8.5Ktr/mm^2 has been generated by this method.

1．まえがき

近年、開発される大規模LSIの多くは、その内部にデータパスを含んでいる。特に、ディジタル信号処理LSIや、マイクロプロセッサでは、データパスは必須の要素である。これらのLSIに含まれるデータパスは、回路規模が大きい（数万トランジスタ以上）ものが多く、その設計には多くの時間が必要となるため、データパスのレイアウトの自動生成に対する要求は、切実なものになっている。

手設計されるデータパスのレイアウトは、通常、ビットスライス構造をしている。この構造では、1ビット分のデータを処理する行（これをビットスライスと呼ぶ）が、データパスのビット数分だけ縦に並んでいる。各行のレイアウトは、ほぼ同じであり、データパスのレイアウトは規則性を持つ。

データパスのレイアウトを、スタンダードセルやゲートアレイの自動配置配線によって生成しようとする試みがなされている[1,2]。しかし、この方法では、データパスが持つレイアウトの規則性を生かしきれず、人手設計に比べて集積度は悪い。また、バス配線が長くなるため、性能も劣る場合が多い。データパスの性能は、チップ全体の性能に大きな影響を与えるため、高速動作を必要とするLSIのデータパス生成にはこの方法は適用できない。

データパスのレイアウトがビットスライス構造であることを利用し、人手設計に近い集積度のデータパスのレイアウトを生成することを目標にしたツールがいくつか報告されている[3-7]。

[3,4]では、レイアウトが作成済みの基本セル（リーフセル）の接続で表現した、データパスの論理図を入力とする。この際、リーフセルの配置位置やバス配線の位置等のレイアウトに関する情報も同時に指定する。この方法では、リーフセルという基本的な構成要素を用いてデータパスの論理設計を行なうことができるため、設計の自由度は高い。反面、論理図入力とその検証に時間がかかり、また、レイアウト設計の知識があまりない電子機器システムの設計者には使用しにくい。

これに対し、[5-7]では、ALUやシフタ等の機能ブロック間の接続で表現した論理図（または、ネットリスト）を入力とする。機能ブロックは、ビット幅や各種オプションでパラメータ化されている。[3,4]に比べ、設計の自由度は劣るが、機能ブロックレベルでデータパスの設計ができるため、論理図が短時間で入力でき、また電子機器システム設計者でも容易に使用できる。

本文で述べるデータパス生成手法では、後者の入力形式を採る。これは、LSI設計者だけでなく、電子機器システム設計者も対象としたデータパスのレイアウト生成システムの開発を目標としているからである。これらの場合、レイアウトは、概ね次の手順で生成される。

(1)ユーザが入力した機能ブロック生成用のパラメータに従って、リーフセルを配置することにより機能ブロックのレイアウトを生成する。

(2)機能ブロックを並べ、それらの間を配線し、データパス全体のレイアウトを生成する。

この場合、レイアウトの集積度を上げるためには、できるだけ配線に要する領域を小さくすることが重要となる。

本文で述べるレイアウト生成手法では、機能ブロック生成用に、セル上配線領域の抽出機能、ソフト／ハード混合マクロ生成機能を持つアレイコンパイラ、データパス全体のレイアウト生成用に、専用の配置配線プログラムを用いている。本手法の特長は、これらのプログラムを用いて、次のようにして、配線領域を削減していることである。

(1)機能ブロック間を結ぶ配線は、アレイコンパイラが抽出した、機能ブロックのトランジスタ上の配線領域を利用する。[5]のように、ビットスライスの間に配線用の領域を設けない。

(2)配線の本数が増え、配線領域が不足する場合は、必要分だけ機能ブロックの高さを伸ばして、配線領域を確保する。これは、ソフト／ハード混合マクロ生成機能を利用している。[6]のように、ビットスライス間に新たに配線領域を設けない。

(3)機能ブロックの端子は、複数の箇所で機能ブロック間を結ぶ配線と接続できるように設計してある[8]。これにより、[7]のような、機能ブロック間の配線領域が不要となる。

本手法を用いたレイアウト生成システムにより、これまでに3種類のデータパスを生成した。その結果、集積度5800～8500トランジスタ/mm^2 （0.8μmルール）のデータパスが生成できた。

本文では、以下2章で本手法によって生成されるデータパスのレイアウト構造について説明する。3章では、本手法によるレイアウト生成システムの概要について述べ、4章、5章では、本手法で用いる主要ツール、アレイコンパイラとデータパス配置配線プログラムについて述べる。6章で適用結果、7章で結論を述べる。

2．レイアウト構造

2.1 データパス全体構造

図1にデータパスのレイアウト構造を示す。データパスは、機能ブロックが横に一列に並んだ構造になっている。機能ブロックは、データ処理部と制御部よりなっている。データ処理部、制御部の内部では、第2層アルミで描かれた電源(vdd)およびグランド(vss)の配線が左右に

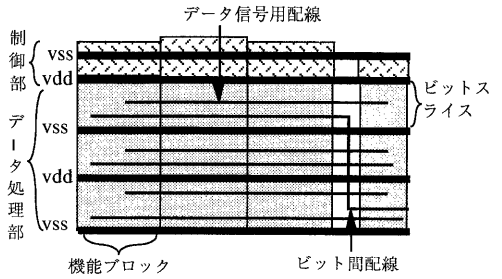
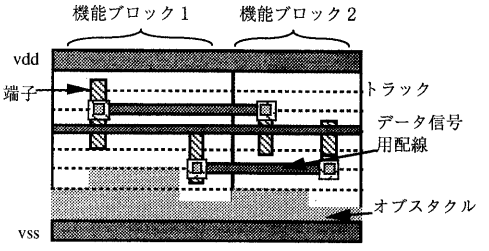
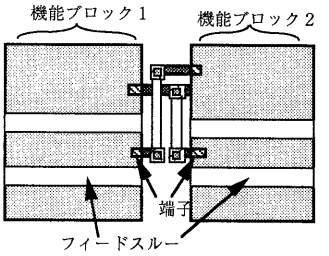


図1　データパスレイアウト全体構造



(a)　セル上配線領域を利用した配線



(b)　機能ブロック間の配線領域を利用した配線

図2　機能ブロック間配線

貫通している。データ処理部の中で、一組のvdd、vssに挟まれた部分が、ビットスライスである。データ処理部中には、ビットスライスが、データパスのビット数分だけ縦に並んでいる。機能ブロックのビットスライスの中のレイアウトは、ビットによって異なる場合がある。また、機能ブロックによっては、レイアウトがないビットスライスが存在する場合もある。機能ブロックのレイアウトは、データ処理部、制御部とも、リーフセルを配置することにより生成される。

機能ブロックのvdd、vssの位置（y座標）は、配置される全ての機能ブロックで一致するように、プログラムによって合わされる。これをvdd、vssのピッチマッチ、あるいは単にピッチマッチと呼ぶ。従って、vdd、vssは、データパス全体を貫通して水平方向に走ることになり、データパス全体のレイアウトも、ビットスライス構造となる。機能ブロック間を結ぶデータ信号は、ビットスライス内のトランジスタ上の配線が通過できる領域（これをセル上配線領域と呼ぶ）を用いてプログラムにより配線される。機能ブロック同士は、通常隣接配置されるが、2個以上のビットスライスにまたがる配線（例えば、0ビット目のビットスライス内の端子と1ビット目のビットスライス内の端子を結ぶ配線。これをビット間配線と呼ぶ）が存在する場合、機能ブロック間に配線領域を設け、そこを通す。

制御信号は、上から制御部に入力され、機能ブロックの中を縦に走る。制御信号用の配線は、リーフセルの中にあらかじめ書き込まれており、リーフセルを隣接配置することにより、自動的に結線される。

2.2 セル上配線領域を使用した配線

図2(a)にビットスライス内で、2個の機能ブロックのデータ信号の端子を配線した例を示す。各々の機能ブロックのビットスライス内では、左右に貫通する多角形のセル上配線領域が存在する。セル上配線領域以外の、配線が通れない部分をオブスタクルと呼ぶ。機能ブロック間を結ぶデータ信号は、このセル上配線領域を使って配線される。データ信号配線の、横方向の線分（これを幹線と呼ぶ）は、第2層アルミ、縦方向の線分（これを支線と呼ぶ）は、第1層アルミで引かれる。幹線は、必ずトラックと呼ばれる等間隔のグリッド上を通る。支線は、端子の上下にしか存在できない。

データ信号用の端子は、セル上配線領域の内部に存在する。この端子は、同一カラム（x座標）には、一つしか存在せず、セル上配線領域内の全てのトラック上の幹線と接続することができる。これにより、図2(a)に示すように、機能ブロック間を結ぶデータ信号は、直線で配線でき、機能ブロックは隣接して配置できる。使用しなかったトラックはフィードスルーとして、これらの機能ブロック上を通過していく配線が利用できる領域となる。これに対し[7]のような、端子が一点でのみ配線と接続でき、しかも端子やフィードスルーの位置が固定になってるレイアウト構造の場合、機能ブロック間に配線領域が必要となる。これを図2(b)に示す。

2.3 ビットスライスのストレッチ

機能ブロックのvdd、vssをピッチマッチする場合、また、機能ブロック間を結ぶデータ信号用の配線の本数が増えて、セル上配線領域が不足する場合、プログラムは、機能ブロックのビットスライスの高さ（すなわち、vddとvssの間隔）を自動的に拡げる。これを、ビットスライスのストレッチ、あるいは、単にストレッチと呼ぶ。ストレッチは、図3に示すように、トランジスタ領

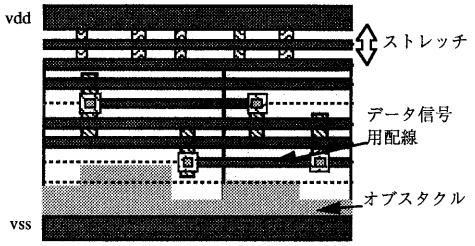


図3　ビットスライスのストレッチ

域と、vddの間を拡げることにより実現する。この際、vddからトランジスタに給電するための配線や、制御信号用の配線も同時に伸ばされる。この機能は、次の利点をもたらす。

(1)セル上配線領域が不足した場合でも、[6]のようにビットスライス間に新たに配線領域を設ける必要がない。これにより、常に隣接するビットスライス間でvdd、vssの配線が共有でき、高集積なレイアウトが生成できる。

(2)[6,7]では、レイアウトをビットスライス構造にするために、全ての機能ブロックのビットスライスの高さが同一になるように、リーフセルを設計する必要がある。本機能は、リーフセルを各々最適な大きさに設計しておき、データパスのレイアウト生成時に必要な分だけ伸ばすことを可能にした。これにより、高集積なレイアウトが生成できる他、リーフセルの設計が容易になる。

3．データパスレイアウト生成システム概要

ここでは、本手法によるデータパスレイアウト生成システムの概要を述べる。

3.1 入出力データ

データパスレイアウト生成生成システムの入力は、次の通り。

(1)機能ブロック間の接続を表す論理図。機能ブロックは、本システムが提供するもののみ使用できる。ユーザは、必要なものを選択し、画面上に配置し、結線する。

(2)機能ブロック生成用パラメータ。機能ブロックは、ビット数や各種オプションでパラメータ化されている。このパラメータを入力する。

(3)機能ブロック配置順序。2章で述べたとおり、データパスは、機能ブロックが横に一列に並んだ構造をしている。この配置順序を指定する。

(4)データ信号用の外部端子の配置位置。データパスのデータ信号用の外部端子をどの辺上（上辺、下辺、右辺、左辺）に配置するかを指定する。上辺、下辺に配置する場合は、特定の2つの機能ブロックの間に配置するように指定することもできる。

出力は、データパスのレイアウトである。本稿では触れないが、実際にはこの他に、シミュレーションモデル等の、各種LSI CADツール用のデータ（回路モデルと呼ぶ）も生成する。

3.2 システム構成

図4に、データパスレイアウト生成システムの全体図を示す。図に示すように、本システムは、次の3つのプログラムよりなっている。

(1)ユーザインターフェース

入力データ用の画面の作成や、機能ブロックの論理シンボルの生成を行なう。また、アレイコンパイラや、データパス配置配線プログラム用に入力データを編集す

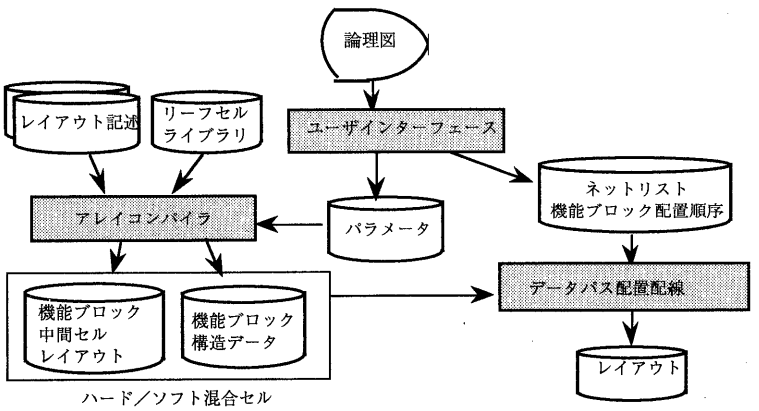


図4　データパスレイアウト生成システム構成図

る。シミュレーションモデル等の回路モデルを生成するプログラム群の実行フローの制御も行なう。その他、ユーザの要求に従って、生成したデータパスの特性値を表示する。

(2)アレイコンパイラ

ユーザインターフェースプログラムより引き渡される機能ブロック生成用のパラメータを入力として、データ処理部および制御部も含んだ機能ブロックのレイアウトを生成する。機能ブロック内部のビットスライスは、電源、グランド配線のピッチマッチや、セル上配線領域の確保のために、データパス全体のレイアウトを生成する際に、ストレッチされる。このため、機能ブロックは、レイアウトが全て固定したセル（これをハードセルと呼ぶ）ではなく、機能ブロックを構成するいくつかのハードセルと、それらの間の相対位置および端子接続要求で表現されたセル（これをハード／ソフト混合セルと呼ぶ）として生成される。機能ブロックのレイアウトは、データパスのレイアウト生成時に、ビットスライスの高さが決定した後で確定する。

アレイコンパイラは、データパス配置配線プログラムが、機能ブロック間を結ぶデータ信号を配線する際に使用する、機能ブロック上の配線領域の抽出を行なう。

(3)データパス配置配線プログラム

機能ブロックを配置し、それらの間のデータ信号を配線して、データパスのレイアウトを生成する。先にも述べた通り、機能ブロックは、ハード／ソフト混合セルで表現されており、機能ブロック内のハードセルの絶対位置は決定していない。配置配線プログラムは、機能ブロック間の電源、グランド配線の位置が一致すること、および、データ信号用配線に必要な領域を確保することを考慮して、各機能ブロックのハードセルの位置を決める。その後、機能ブロック内のハードセル間の配線、および機能ブロック間のデータ信号の配線を行ない、データパス全体のレイアウトを生成する。

4．アレイコンパイラによる機能ブロック生成

ここではまず、アレイコンパイラの概要を述べ、データパス生成で使用している、ハード／ソフト混合セル生成機能と、セル上配線領域抽出機能について説明する。また、それらを用いて、機能ブロックをいかに生成するかについて述べる。

4.1 アレイコンパイラ概要

アレイコンパイラは、基本的にはタイル方式により、レイアウトを生成する。これは、あらかじめ設計してあるリーフセルを、手続き言語で記述したリーフセルの配置規則に従って配置することにより、レイアウトを生成する方法である。この配置規則は、生成対象としているセルのレイアウトを記述しているものであり、ここではこれをレイアウト記述と呼ぶ。本アレイコンパイラでは、レイアウトを記述するのに、リーフセルの配置規則だけでなく、セル内部の配線パターンの記述や、セル内配線用の自動配線プログラムを起動する記述も可能となっている。

レイアウト記述では、レイアウトを階層的に記述できる。この階層の深さに制限はない。例えば、RAMを記述する際に、まずメモリセルを配置したメモリ部やバッファセルを配置したバッファ部を記述し、次にこれらを配置してRAMのレイアウトを記述する。これらの中間階層のセルを中間セルと呼ぶ。これに対して最上層のセル（この例では、RAM）を特にモジュールと呼ぶ。

アレイコンパイラは、レイアウト記述、リーフセルおよびビット数等のパラメータを入力することにより、所望のモジュールのレイアウトを生成する。

4.2 ハード／ソフト混合セル生成機能

アレイコンパイラは、モジュールを、レイアウトが全て固定したハードセルとしてだけではなく、ハード／ソフト混合セルとして生成することができる。これは、中間セルをレイアウトの確定したハードセルとして生成し、モジュールを中間セル間の相対位置と、端子の接続要求で表現したものである。図５に、ハードセルと、ハード／ソフト混合セルの概念図を示す。この機能は、モジュールがより上位階層のセルやチップの中に配置される場合、外部の状況によりモジュールの大きさや形状を変えたい場合に使用する。

ハード／ソフト混合セルを生成するためには、レイアウト記述で以下のように記述する。

(1)ハードセルとして生成したい中間セルのレイアウト記述では、リーフセル、中間セルを隣接配置するように記述する。また、配線パターン、自動配線プログラムの起動命令を記述する。

(2)最上層のモジュールの記述では、レイアウトを中間セル、リーフセルの相対配置で記述する。相対配置とは、新しく配置する中間セル、リーフセルを、既に配置してある中間セル、リーフセルに対して、どの方向（上、下、右、左）に配置するかのみを指定する配置である。互いに隣接しているのか、どのくらいの距離をおいて配置されるかについては、何も指定しない。相対配置は、中間セル、リーフセルが並んだセル列（またはセル行）同士でも記述できる。

(3)相対配置した中間セル、リーフセルの端子間、またはセル列やセル行を構成する中間セル、リーフセルの端子間の結線要求を記述する。

4.3 セル上配線領域抽出

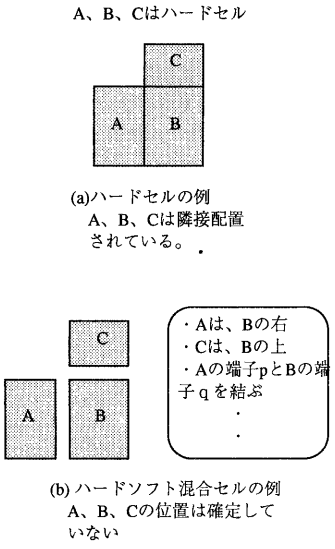


図5　ハードセルとハード／ソフト混合セルの例

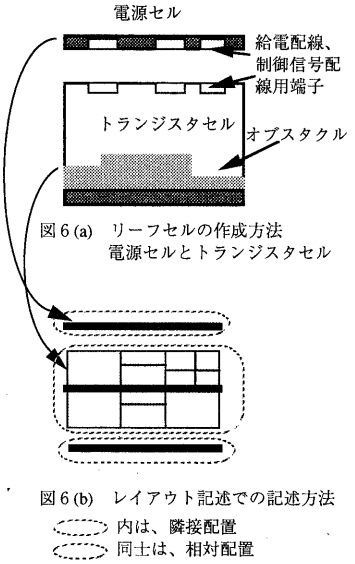


図6　機能ブロックの生成方法

アレイコンパイラは、セル上配線領域の抽出機能を持つ。これは、モジュールを含む上位セルや、チップ全体の配線の際に、モジュールの上の配線が通れる領域を配線領域として利用することにより、集積度を上げることを目的とした機能である。セル上配線領域を抽出するために、アレイコンパイラは、リーフセルのオブスタクルを、リーフセルの配置座標に従って階層的に合成する。合成されたオブスタクル以外の領域が、モジュールのセル上配線領域である。この手法は、生成したモジュールのレイアウトから配線領域を抽出する手法より高速である[9]。

4.4 機能ブロック生成方法

機能ブロックを生成するためには、機能ブロックのリーフセルとレイアウト記述を作成する必要がある。これらは、次のようにする。

(1)データ処理部、および制御部を構成するリーフセルを作成する。データ処理部を構成するリーフセルを作成する際には、トランジスタまたはグランド配線（vss）を含むセルと、電源配線（vdd）のみを含むセルを作成する。トランジスタまたはvssを含むリーフセルをトランジスタセル、vddを含むリーフセルを電源セルと呼ぶ。図6(a)にトランジスタセルと電源セルの例を示す。トランジスタセルでは、第2層アルミはvssに近い位置でのみ使用し、その位置をオブスタクルとする。電源セルは、全面がオブスタクルである。vddからトランジスタへの給電配線や、縦に走っている制御信号用の配線が存在する位置には、これらの配線用の端子を定義しておく。

(2)レイアウト記述で、これらのリーフセルの配置位置を記述する。データ処理部の記述では、まずトランジスタセルだけから構成される中間セル（トランジスタ中間セルと呼ぶ）、電源セルだけから構成される中間セル（電源中間セルと呼ぶ）を隣接配置で記述する。制御部は、制御部を構成するリーフセルを隣接配置して中間セルとする（これを制御中間セルと呼ぶ）。トランジスタ中間セル、電源中間セル、制御中間セルの位置関係は、相対配置で記述する。同時に、電源セルからの給電配線や、制御信号の配線用の端子間の結線要求を記述する。制御中間セルとトランジスタ中間セルを隣接配置させて、新たな中間セルにしてもよい。

アレイコンパイラを実行すると、各ビットスライスのトランジスタ中間セル、電源中間セル、制御中間セルがハードセルとして生成される。これらの相対位置、端子接続要求がファイルとして生成される。また、各リーフセルのオブスタクルは合成され、機能ブロックのオブスタクルが生成される。オブスタクル以外の領域は、セル上配線領域として、次のデータパス配置配線プログラムで使用される。

5．データパス配置配線プログラム

データパス配置配線プログラムは、次の手順でデータパスのレイアウトを生成する。

(1)機能ブロックを構成する中間セルの仮配置

ユーザが入力した機能ブロックの配置順序、および機能ブロックジェネレータが生成した機能ブロック内部の中間セルの相対位置より、各機能ブロックの中間セルを仮配置する（図７）。この時、各機能ブロックのvdd、vssの位置（ｙ座標）が一致するように配置する。トランジスタ中間セルと電源中間セルの距離は、この段階では、まだ決定されていない。

(2)ビット間配線の経路決定

ビット間配線は、機能ブロックと機能ブロックの間に配線領域（ビット間配線領域と呼ぶ）を設けて、そこを通す（図７）。ここでは、全てのビット間配線について、どのビット間配線領域を通すかを決定する。これは、次のようにする。

①ビット間配線でつなぐ端子を持つ機能ブロックのうち、最も左に配置される機能ブロック、最も右に配置される機能ブロック、およびそれらの間に配置されている機能ブロックを左から順にBi、Bi+1、・・、Bjとする。

②i≦k＜jを満たすすべてのkにつき、BkとBk+1の間にビット間配線を通した場合の配線長を見積もる。配線長は、ビット間配線で結ばれる全ての端子を結ぶスタイナー木を作成し、その長さとする。

③配線長が最小になるkを探す。BkとBk+1の間にビット間配線領域を設け、ここにビット間配線を通す。

(3)ビット間配線領域内部での配線位置の決定

ビット間配線領域では、ビット間配線の縦線分は、等間隔で設けられた縦トラックの上を通る。ここでは、１つのビット間配線領域を複数のビット間配線が通る場合、各々のビット間配線の縦線分が、どの縦トラック上を通るかを決定する。あるビット間配線が結ぶ最も上側のビットをi、最も下側のビットをjとすると、このビット間配線の縦線分は、iの下側にあるvdd、またはvssから、jの上側にあるvdd、またはvssまで引かれるものとする。必要な縦トラック数が最小になるように、left-edge法[10]で各々の縦線分が通る縦トラックを決定する。

(4)ビットスライス内部の配線経路の決定

ビット間配線の位置が決まれば、ビットスライス内部の配線が可能となる。配線は、図８に示すように、各々のビットスライス毎に行なう。これは、チャネル配線の問題になる。チャネル配線のアルゴリズムは、[11]を基本として、オブスタクルを扱えるように改良したものを用いている。

(5)ビットスライスのストレッチ

ビットスライス内部の配線経路が決定すると、各ビットスライスにおける必要な配線領域が判る。この値に従い、必要ならばそのビットのトランジスタセルと電源セルの間を離して配置し、それらの間を、機能ブロックジェネレータが出力した結線要求の通りに配線する。これがビットスライスのストレッチである。トランジスタセルと電源セルの間の配線は、全て縦方向に伸びた直線となる。これにより、データパスのレイアウトが生成できる。

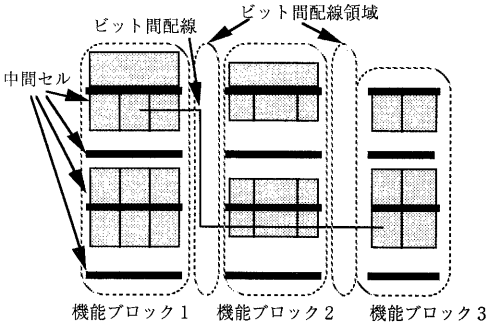


図７　機能ブロック構成セルの仮配置とビット間配線

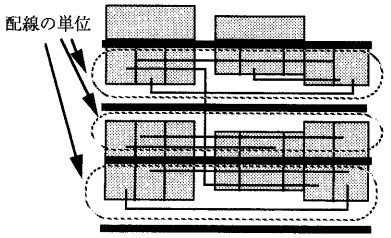
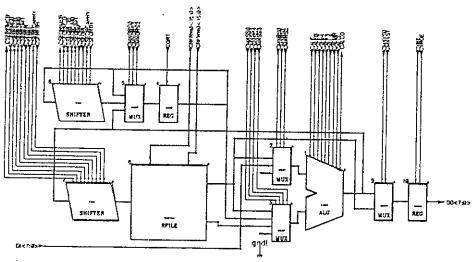


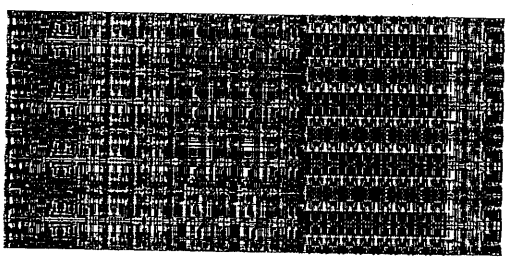
図８　セル上のデータ配線

６．適用結果

表１にデータパスの生成結果を示す。図中で2901は2901データパスと同機能のデータパス、DP1は、マイクロプロセッサ、DP2はディジタル信号処理LSI用のデータパスである。また、ストレッチ数は、セル上配線領域が不足したことによるビットスライスのストレッチが、トラック数にして何本分起こったかを、全てのビットスライスについて合計した値である。DP1は、バス配線の数が多く、ストレッチ数が多くなったこと、集積度の低いシフタを多く使用していることが、2901より集積度が劣っている理由である。DP2は、17個の機能ブロックのうち、13個が14ビット幅、4個が10ビット幅になっている。また、ビット間配線本数も多く、これらが2901より集積度が劣っている原因である。過去に手設計されたマイクロプロセッサ用のデータパスの集積度は、5000トランジスタ/mm^2 （0.8μmルール）であった。このことを考慮すると、本手法で人手設計なみまたはそれ以上の集積度のレイアウトが生成できるといえる。図９(a)に2901の入力となった論理接続図を示す。図９(b)に生成したデータパスのレイアウトを示す。



(a) 論理図（入力）



(b) レイアウト（出力）

図9 2901データパス生成例

7．結論

高密度なデータパスのレイアウトを生成する手法について述べた。本手法では、機能ブロックのレイアウトを、セル上配線領域の抽出機能、ソフト／ハード混合セル生成機能を持つアレイコンパイラを使用して生成する。データパス全体のレイアウトは、専用の配置配線プログラムを用いて生成する。この際、(1)機能ブロック間を結ぶデータ信号の配線を、機能ブロック上の配線領域を用いて行なうことにより、ビットスライス間の配線領域を不要としたこと、(2)機能ブロック上の配線領域が不足する場合にも、ビットスライスをストレッチすることにより、ビットスライス間の配線領域を不要としたこと、(3)データ信号用の端子を、複数の点でデータ信号配線と接続可能としたことにより、機能ブロック間の配線領域を不要としたこと、により高密度なレイアウト生成を可能とした。また、ストレッチ機能は、ビットスライスの高さを機能ブロック全体で揃える必要をなくし、リーフセルの作成を容易にするとともに、データパスの高密度化にも貢献した。

本手法を用いて生成したデータパスは、トランジスタ密度5800～8500トランジスタ／mm^2（0.8μmルール）を達成し、人手と同程度の集積度のレイアウトが生成できることが分かった。

表1 データパス生成結果

	2901	DP1	DP2
機能ブロック数	10	11	17
ストレッチ数	0	32	4
ビット間配線数	0	1	24
集積度 (ﾄﾗﾝｼﾞｽﾀ／mm^2)	8500	5800	6560

参考文献

[1] J.Rowson et al., "A Datapath Compiler for Standard Cells and Gate Arrays," Proc. of Custom Integrated Circuits Conference, pp149-152, 1987.

[2] T.Hattori et al., "A Method of Dataflow Oriented Cell Initial Placement," International Workshop on Layout Synthesis, No.9.1, 1990.

[3] N.Matsumoto et al, "Datapath Generator Based on Gate-Level Symbolic Layout," Proc. of 27th Design Automation Conference, pp388-393, 1990.

[4] D.Curry, "Schematic Specification of Datapath Layout," Proc. of International Conference on Computer Design, pp28-34, 1989.

[5] R.Gordon et al., "An N-Bus Datapath Compiler for IC Design," Proc. of Custom Integrated Circuits Conference, No.23.3, 1989.

[6]H.Cai et al., "A Data Path Layout Assembler for High Performance DSP Circuits," Proc. of 27th Design Automation Conference, pp306-311, 1990.

[7]今橋 他、「レジスタトランスファーレベル回路図を用いたデータパスレイアウト設計手法」, VLD91-131, pp23-29, 1992.

[8] Y.Tsujihashi et al., "A High Density Data Path Generator with Stretchable Cells," Proc. of Custom Integrated Circuits Conference, No 11.3, 1992.

[9]塚本 他、「タイル方式により生成したモジュールのセル上配線通過可能領域抽出手法」、情報処理学会第41回全国大会、1990.

[10] A.Hashimoto et al., "Wire routing by optimizing channel assignment," Proc.of 8th Design Automation Conference, pp214-224, 1971.

[11] T.Yoshimura, "An Efficient Channel Router," Proc.of 21st Design Automation Conference, pp38-44, 1984.