柔軟性のある乗算器モジュールジェネレータの開発

5K−5

大江　良一*　　坂手　将人*　　萩　哲也**
*(株)富士通研究所　　**富士通(株)

1．はじめに

乗算器やROM/RAMなどのように規則構造を持つモジュールレイアウトの自動生成は、これまでにもASICの設計によく用いられ、高機能化や短納期化に大きく役立ってきた。しかしながら従来の自動生成は、あらかじめ作成されている専用セルを規則的に並べるという方法が一般的で、与えられた入力に対して1つのレイアウト結果しか得られず、モジュール形状や端子位置の変更などを容易に行うことはできなかった。そのため、今後さらに高集積化・高速化・多様化が進むASICに対応するには、回路の構成からレイアウトに至るまで、もっとユーザの意図が反映できる柔軟性のあるモジュールの自動生成が必要となってきている。

本モジュールジェネレータでは、回路の接続と各セルの配置は設計者が記述し、配線はすべて自動で行う方式により、こうした問題の解決を図った。この方式では設計者自身の記述により、回路構成の変更や自動配線のトライアル、さらには端子位置／モジュール形状の決定などを容易に行うことができる。

このような方式を用いたモジュールジェネレータの最初のターゲットとして、任意ビットの乗算器を自動生成する乗算器モジュールジェネレータの開発を行った。自動配線を用いたことにより、従来規則化が難しく自動生成には用いられていなかったWallace Tree方式の採用など回路構成の最適化や配置／配線のトライアルが可能となり、高機能な乗算器モジュールが自動生成できた。

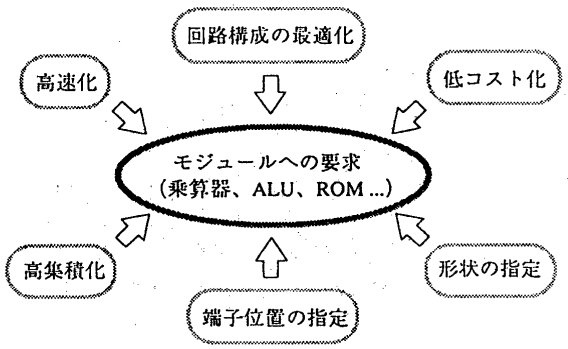


図1．ユーザの要求の多様化

A Flexible Module Generator for Multiplier
Ryoichi OHE *, Masato SAKATE *, Tetsuya Hagi **
* FUJITSU LABORATORIES LTD., ** FUJITSU LIMITED

2．モジュールジェネレータの基本方式

本モジュールジェネレータは、マニュアル設計に近いレイアウトの自動生成をめざして開発を進めている。マニュアル設計の場合、設計者はまず図1に示すようなさまざまな要求のうち、特に何に重点を置くかにより試行錯誤を繰り返し、それぞれのトレードオフを考えながら最適なレイアウトを作成している。そこでこのようなマニュアル設計での方法を考慮し、従来までのように完全に自動生成するばかりでなく、マニュアル設計で行われているようなトライアルが可能な設計環境の構築をめざした。そのため、モジュールの記述を行うアクセス関数や配線順序などを考慮できる自動配線を用意し、さらにディスプレイへの表示や外部データへの変換などを可能とした。これらの関数はすべてC言語で作成されており、設計者はこれらの関数を用いてC言語でプログラミングすることによりモジュールを作成する。また、自動配線についても、配線順序などの指定を外部記述できるようにしているため、レイアウトのトライアルやクリティカルパスの優先配線などが可能である。

基本構成を図2に示す。本モジュールジェネレータでは、レイアウトばかりでなく論理回路図を作成することも前提としている。これらは基本的にどちらも配置／配線という同じ処理であるため、読み込むデータと配置位置を変えるだけで同じ自動配線により生成することができる。そのため生成手順としては、まずネットリストを記述した後、各セルのシンボルの配置位置を記述して自動配線により論理回路図を作成し、続いて同様にセルパターンの配置位置を記述してレイアウトを生成する。

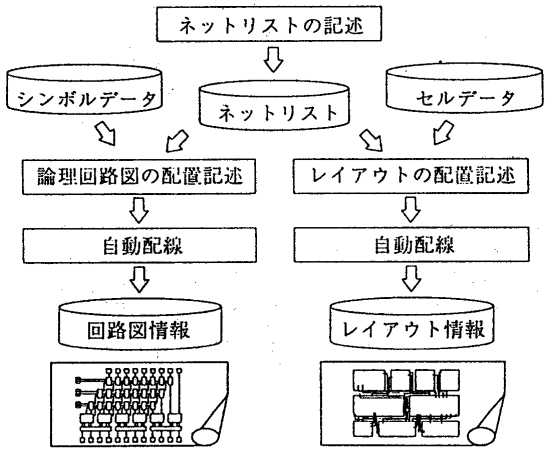


図2．モジュールジェネレータの基本構成

3．処理概要

（1）ネットリストの記述

モジュール内の各セルの接続関係について記述する。具体的には、まず使用されるセルすべてについて、そのセルのライブラリ名と任意の固有名を記述し、さらにセル間の接続をセルの固有名と端子名により記述する。

（2）配置の記述

使用されるすべてのセルについて配置座標を記述する。座標値の指定は、絶対座標でも、すでに配置されているセルからの相対座標でもよい。また、外部端子もセルとして扱っているため、任意の位置に端子を置くことができる。

（3）自動配線

ネットリストと各セルの配置座標が設定されていれば、配線はすべて自動で行われる。配線には3次元の迷路法を用いているため、指定により何層配線でも可能である。また、配線ルール／配線順序／配線領域などを外部データとして記述できるため、配線トライアルが容易である。

4．乗算器への適用

このモジュールジェネレータの方式を実際に任意ビットの乗算器の自動生成に適用した。被乗数および乗数のビット数を与えることにより、乗算器のネットリスト／論理回路図／レイアウトが自動生成できる。

生成される乗算器の構成を図3に示す。従来規則化が難しく自動生成には用いることができなかった Wallace Tree 方式【1】を採用することにより、特にビット数の大きい乗算器の高速化が実現できた。また乗算器の場合、ビット数によって最適な回路構成が異なるため、今回トライアルとして、ユーザの指定により最終段加算器を Carry Select Adder【2】で構成することも可能とした。これにより、演算速度重視か面積の最小化重視か、といったユーザの要求をもとに回路構成やレイアウト構成が選択でき、より高機能なモジュールの自動生成が可能となった。

実際に生成した乗算器の論理回路図とレイアウト図を図4に示す。0.8 μm ルールのスタンダードセルを用い、7種類の標準セルをそのまま使用した。この乗算器モジュールジェネレータは、C言語のプログラム約400行、外部記述データ約40行から成っている。処理時間は、16 bit ×16 bit の乗算器の場合、S4/2 の CPU Time 約12分であった。

5．まとめ

設計者の記述により生成を行うモジュールジェネレータを開発した。ネットリストおよび配置位置の記述と自動配線により、論理回路図とレイアウトを生成することができる。設計者自身で記述を行うため、回路構成の最適化や自動配線のトライアルなどが可能である。この方式を任意ビットの乗算器モジュールの自動生成に適用し、高速・高密度な乗算器モジュールを短時間で生成することができた。

今後は、さらに他のモジュールへの適用や、トライアル環境の整備、配置の自動化などについて検討を進めていく。

参考文献

【1】Wallace, C. S., "A Suggestion for a Fast Multiplier", IEEE Trans Electronic Computers, Vol, EC-13, FEB. 1964

【2】坂手他 "112 ビット高速加算回路の設計" 電子情報通信学会集積回路研究会 VLD91-45, 1991

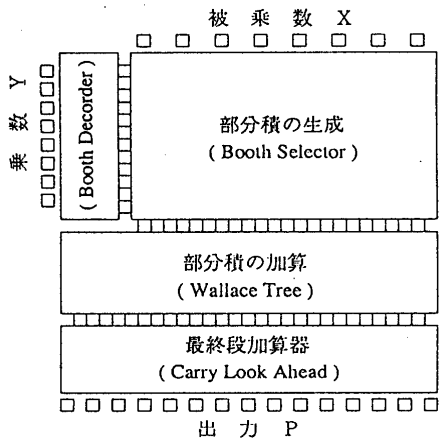
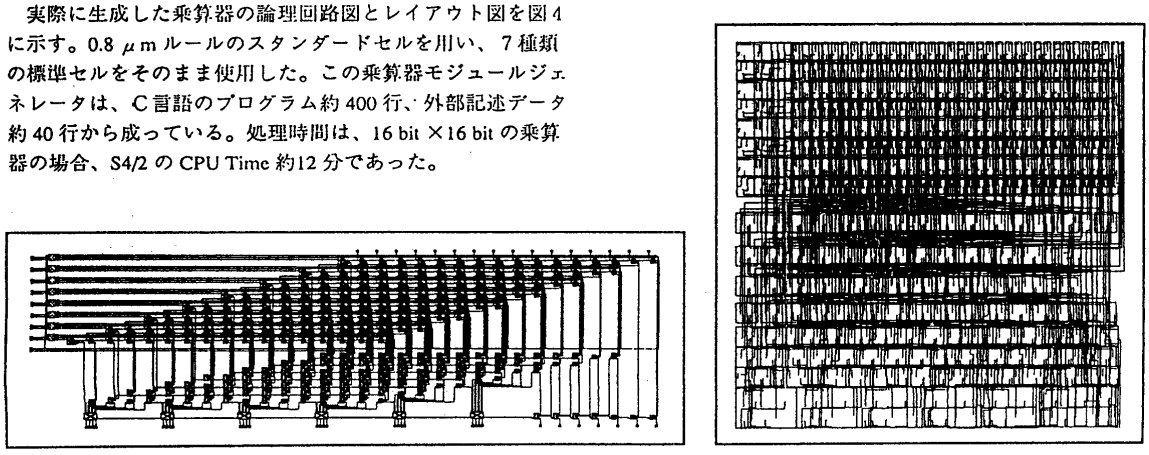


図3．乗算器の構成



(a) 論理回路図　　(b) レイアウト図

図4．16 bit ×16 bit 乗算器の生成例