5P-03　FPGAを指向した高速な配置配線プログラムの作成

宮下 朗
丸山 勉、星野 力
筑波大学 機能工学系

1　はじめに

近年、再構成可能なデバイスである Field Programmable Gate Array(FPGA) の高集積化、高速化に伴い FPGA アクセラレータの研究が盛んに行われている [1]。FPGA アクセラレータとは、既成のシステムに FPGA を Coprocessor として付加することで高性能化を図るシステムである。

FPGA アクセラレータを用いたアプリケーション開発には FPGA 上に構成されるハードウェアを設計する必要がある。このプロセスを自動化するため、C 言語プログラムをハードウェア記述言語 (HDL) に変換するツール (CtoHDL コンパイラ) が提案されている [2]。

ここで、HDL を実際に FPGA に書き込まれる回路データへと変換するため、通常はハードウェア設計用 CAD ツールのレイアウトプログラムを用いる。このプログラムは、ランダムロジックを対象としており、作成される回路の規模をできるだけ小さくするように、様々な最適化手法により配置配線を行っている [3]。そのため、大規模な回路の場合、変換に数時間から数十時間を要する。しかし、FPGA アクセラレータでは、用いる FPGA デバイスのサイズが予め決まっており、作成される回路はそのデバイス内に収まる大きさであればよい。さらに、C 言語で用いられる様々なアルゴリズムは、強いデータ依存関係を持つ。従って、その処理をハードウェア化した場合、データフローグラフは比較的単純なものになる。

そこで、本研究ではこれらの特徴を利用し、ロジックを順次配置配線することで、CtoHDL コンパイラ用の極めて実行時間の短い回路レイアウトツールの作成を目指す。対象とする FPGA は Xilinx 社の Virtex シリーズとし、その回路データ作成には JBits を用いる。

A High Speed Placement and Routing Method for FPGA
Akira Miyashita, Tsutomu Maruyama, Tsutomu Hoshino
(Institute Mechanics and Systems University of Tsukuba)

2　FPGA Architecture

FPGA はデバイス内に碁盤の目状に配置された Logic block と Routing switch、さらに縦方向及び横方向に連続して配置された Routing resource から構成されている。Logic block は様々な Logic やステートマシンを構成することができる。デバイスの左右にはメモリが用意されており、さらに、Logic block をメモリとして用いることも可能である。

ユーザはこれらの状態を指定した Configuration bitstream を FPGA に書き込むことにより、様々な回路を実装することができる。

3　CtoHDL コンパイラ

CtoHDL コンパイラは C 言語プログラムを HDL に変換するプログラムである。作成される回路は以下のような特徴を持つ。

- 回路全体のデータフローが極めて単純であり、レジスタで区切られたいくつかのステージに分割することができる。
- 限られた数種類の Logic(adder,comparator,...,etc) で構成されている。
- ループなどより発生する複数のステージを跨ぐフィードバックパスが存在し、その所在は明示的に示される。
- メモリアクセスが多数存在する。

4　システムの構成

図 1に本ツールが組み込まれるシステムの概要を示す。

まず、CtoHDL コンパイラにより C 言語ソースが HDL ソースに変換される。本ツールはその HDL ソースに対して配置配線処理を施し、JBits を用いて FPGA の Configuration Bitstream を出力する。

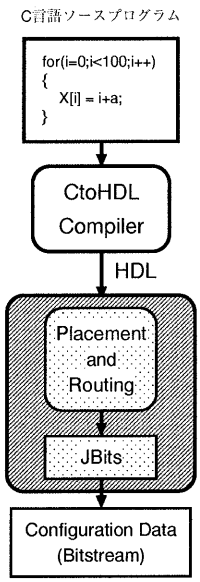

図 1: システムの概要

5 配置配線

現在の CAD ツールの実行時間の長さは、ランダムロジックを対象とした配置と配線の最適化にともなう繰り返し計算に起因する。一方、Section 1で述べたように、FPGA アクセラレータでは予めデバイスのサイズが決まっており、作成される回路の規模はその中に収まる大きさであれば良い。即ち、必ずしも最小である必要はない。さらに、Section 3で述べたように、CtoHDL コンパイラが出力する回路は、限られた数種類の Logic より構成されるステージ単位に分割可能であり、そのデータフローグラフは枝分かれが存在しない極めて単純なものである。

そこで、本研究ではこれらの特徴を利用した次のような配置と配線を行う。

まず、回路全体の遅延のクリティカルパスは、デバイスの左右に位置するメモリにアクセスするパスとなることが予想される。このことから、ロジックから各メモリへのパスができるだけ短くなるよう、例えば1つのステージで異なった2つのメモリへとアクセスする場合、それらのメモリができるだけ近い位置になるようメモリ割り当てを行う。

次に、割り当てられたメモリの位置とデータフローグラフに従い、連続する各ステージができるだけ隣接するように、大まかなステージの配置位置を決定し、これに従い一番最後のステージより Logic を順々に配置していく。

各ステージごとが配置が決定した後、必要な入出力間の配線を行う。この配線には Maze router を用いる。配線はパスの長さが長い順、即ち、メモリへのアクセス、フィードバックパス、隣接するステージ及び同一ステージ内の配線の順で行う。より長いパスを先に配線することで、優先的に Routing resource を使用させ、その長さをできるだけ短くする。この時、フィードバックパスの配線に関しては、入力先の Logic の位置が確定していない。そこで、各ステージの Logic のサイズから大まかな配置位置を推定し配線を行う。その後、入力先の Logic の配置位置が決定した後に残りの配線を行う。また、もし配線が不可能なケースが発生した場合、その原因は配置時にステージ内の各 Logic 間の距離が近すぎたため、十分な配線スペースが用意されていなかったものと考えられる。従って、各 Logic 間に十分なスペースを確保した Logic 配置に変更し、改めて配線を行う。

6 おわりに

本論文では実行時間の短縮に重点を置いた CtoHDL コンパイラ用配置配線プログラムの作成について述べた。

この配置配線は C 言語を回路化したとき、その回路は限られた数種類の Logic より構成され、そのデータフローグラフが単純であるという特徴を利用することで、順次、Logic を配置配線していくものである。

参考文献

[1] Ferran Lisa, Faustino Cuadrado, Dolores Rexachs and Jordi Carrabina, A Reconfigurable Coprocessor for a PCI-based Real Time Computer Vision System, FPL 1997

[2] Tsutomu Maruyama, Tsutomu Hoshino, "A C to HDL compiler for pipeline processing on FPGA", FCCM 2000

[3] Vaughn Betz, Jonathan Rose, Alexander Marquardt, "Architecture and CAD for Deep-submicron FPGAs", 1999, Kluwer Academic Publishers