フロアプランプロシージャインターフェース *FPPI*

5E−4

南光康志　高橋直哉　米澤典剛

日本電気 株式会社

1 はじめに

近年、マルチウィンドウ / マルチプロセスをサポートするEWSの普及が著しい。LSI設計効率化の観点から、このようなEWS上で複数のCADツールをインタラクティブに操作しながら設計を進められる統合的なCAD環境の構築が重要となっている。

複数のツールを統合する場合、問題となるのは各ツール間のデータ交換である。通常、個々のツールは処理効率を考慮した固有のデータベース構造を有しているため、

- ツール間のデータベース形式の変換を行う場合、フォーマット変換処理が大きな時間を占め、インタラクティブな操作性を大きく損なう。
- ツール間で共通なデータベースを設定する場合、データベースの構造が複雑かつ容量が大きくなりすぎる。

等の問題点があった。

我々は、チップレベルのマクロ自動配置、概略配線、面積見積りの機能を持つフロアプランツール[2]と、スタンダードセル自動配置・配線ツール[3]を統合するにあたり、上記の問題点なしにツール間のデータ変換を可能とする手続き型インターフェース(Pocedural Interface)[1]『FPPI』を開発したので、ここに報告する。

2 構成

2.1 概要

本インターフェースによるシステム統合の概要図を図1に示す。本インターフェースではLSIレイアウト設計ツールで普遍的なデータを表現する仮想的なデータモデルを設定し、このデータにアクセスを行う形式で異るツール間のデータ交換を行う。この仮想的なデータモデルを設定したことにより、各ツールのデータベースの形式によらない単一のインターフェース関数によるデータ交換を可能とした。

以下、このデータモデルとインターフェース関数について述べる。

2.2 データモデル

フロアプランツールを含む各種のLSIレイアウト設計用ツールでは、各々のツールのみで必要な一部の情報以外の大部分は同種の情報である。このようなデータとしては

- レイアウトデータの最上位(チップ)[Chip]
- 機能ブロック[block]
- Chip, Blockなどの形状[Shape]
- ネット情報[Net]
- 端子[Pin]
- 配線経路[Wire]
- 各Block固有の属性[Attribute]

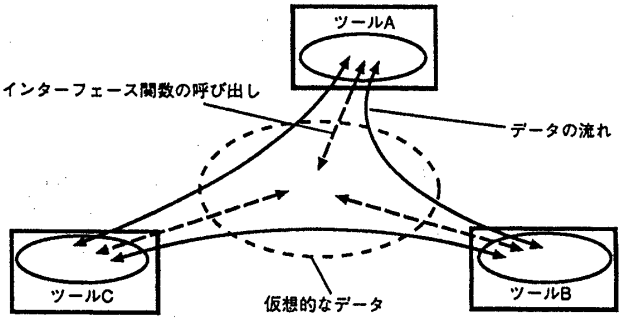

図1: 本インターフェースによるシステム統合

等があるが、これらは互いに参照しあい、全体として複雑な構造になっている。

本インターフェースでは図2のようにデータ構造を一般化し、この上でデータにIDとして項目ごとに一意な正の整数を個別に割当て、これを元にしたインターフェース関数の呼出しによりデータへのアクセスを行なう。

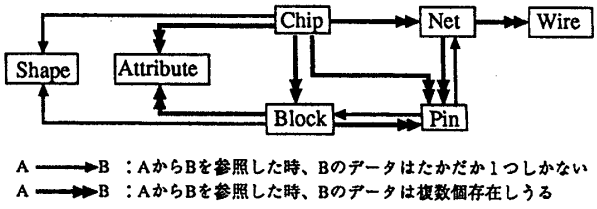

図2: データモデル概要図

2.3 インターフェース関数

インターフェース関数は基本的に次のような形式を持つ。

- データの参照

```
FPPIGet<object><data>(<object_id>)
FPPIGet<object><data>(<object_id>,<previous_data_id>)
```

- データの変更

```
FPPIPut<object><data>(<object_id>)
FPPIPut<object><data>(<object_id>,<previous_data_id>)
```

ここで、<object>はID付けされたデータ、すなわち図2で示したモデルの各項目の名称を表す。これのIDが<object_id>である。図2の矢印で示した参照関係で複数の参照が可能なデータ項目では、全てのデータを得るために一つ前のIDを<previous_data_id>として知らせ、順次データを得る。一番最初は0を<previous_data_id>として上記関数を呼び出し、<previous_data_id>を得る。

以上の関数により、<object>の項目<data>の値を参照 / 変更する。

FPPI: A FloorPlan Procedural Interface

Yasushi NANKOU, Naoya TAKAHASHI, Noritake YONEZAWA

NEC Corporation

3 システム統合

上記のようなインターフェース関数を作成し、自動配置・配線ツール(以下ツールA)とフロアプランツール(以下ツールB)の統合を行った。ここでのデータ交換は以下のように行われる。

3.1 ツールBの起動 / 通信路作成

ツールAはツールBの起動を行い、これとの間に通信路を作成する。これは、daemonとして常駐しているプログラムに対しツールBの起動要求を行ない、要求を受けたdaemonがツールB起動とともににツールAとの間の通信路作成のための情報をツールBに渡すことにより行われる。(図3)

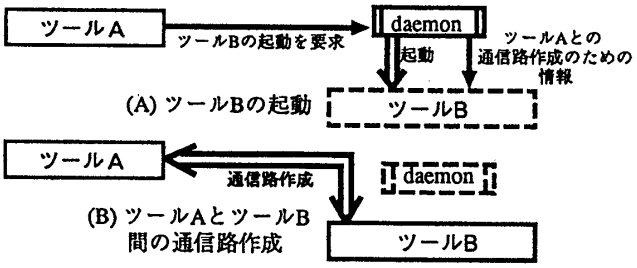

図 3: ツールAからのツールBの起動

3.2 ツール間のデータ転送

通信路が確立されると、まずツールAがツールBへデータ構造への入口となるIDを送信する。ツールB側では受信したIDを元にデータ構造をたどり、必要なデータを得る。データ送受信のイメージとしては図4(A)に示したような、仮想的なデータベースに対する参照 / 変更によって行なわれる形式であるが、その実現方法は図4(B)および以下に示すようなものである。

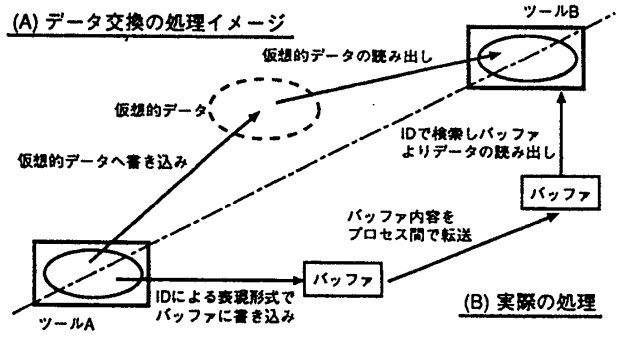

図 4: データ交換

1) ツールAは要求されたIDに対応するデータを自身のデータベース内から検索し、これをバッファに格納する。バッファの内容は図5のような、図2に示した構造をIDで表現した形式になっており、データ要求を受けたID下のデータだけを保持している。バッファにデータを格納後、ツールAはツールBへそのバッファ内容を送信する。

2) ツールBは、必要なデータをバッファから読み出す。読み出したデータがIDである場合、そのID下のデータを得る必要があればそのIDについてデータをツールAへ再度要求して、バッファ内容を更新し、そこからデータを読み出す。

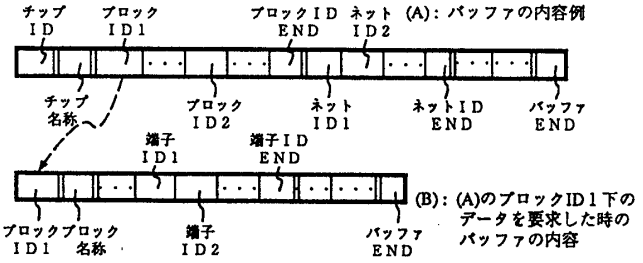

図 5: バッファ内容

以上を繰り返し、ツールAからツールBへ必要なデータを転送する。ツールBでの処理終了後これを逆方向に行い、ツールAはツールBでの処理結果を得る。

なお、本インターフェースでツール間で作成する通信路はUNIXのsocketインターフェースを用いているので、ネットワーク透過なデータ交換が可能である。

図6にこのシステムの表示画面例を示す。

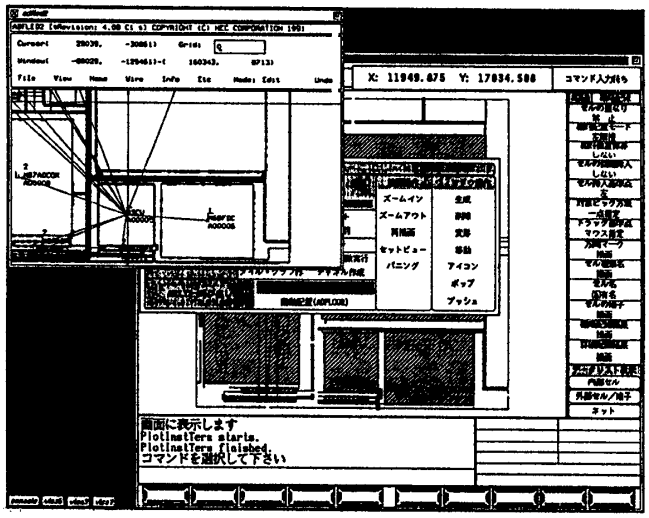

図 6: 表示画面例

4 おわりに

本インターフェースによりツールのインタラクティブな操作性を損なわず、かつ巨大な共通データベースを設定せずに、異なる機能をもつツールの統合環境が実現できた。

また、このインターフェースはネットワーク透過なので各ツールを異るEWS上で実行し、負荷を分散させることも可能である。

今後の課題としては、一方のツールでのデータ変更が即座に他方のツールのデータベースに反映されるようなインクリメンタルな環境のためのインターフェースの開発が必要である。

参考文献

[1] CAD Framework Initiative, Inc.:*"A Demonstration of CAD Standardization at the 1990 Design Automation Conference PROJECT PLAN Version 1.0"*, Nov.1990

[2] N.Yonezawa, et al.:*"A VLSI Floorplanner Based on 'Ballon' Expansion"*, Proc. EDAC, March,1990 pp.257-261

[3] 村越他:"スタンダードセル統合化レイアウトシステムの開発 ", 情報処理学会第43回全国大会講演論文集 (1991)