# 仮想プリンタデザイナ

尚　幹夫

近年，並行設計，協調設計といった設計効率化のための方法論が重視されるようになり，その設計支援ツールとして，我々は，プリンタシミュレータ（以降，仮想プリンタ）を開発し，運用を行ってきた[1)]。

仮想プリンタにより，キャッシュミス率，バス占有率などの統計データの出力と共に，プリンタ内部のデータ処理の状況が可視化されるため，プロセッサにかかる負荷や，ハードウェアモジュールの動作状況が把握できるようになり，プリンタのデータ処理能力解析に成果を上げた。

一方，プリンタの方式設計は，処理の一部をハードウェアにより高速化し，プロセッサとハードウェアモジュールの並行処理により，データ処理性能を向上させる方式に変わってきた。

しかし，仮想プリンタでそのハードウェアモジュールを動作させるためには，処理手続きをモデル化し，仮想プリンタ上の部品として動作するようにコーディングしなければならない。従来は，ハードウェアモジュールの量も少なく，人手によるコーディングも問題にはならなかったが，ここに来てその量が格段に増え，人手によるコーディングでは，当初の目的である並行設計が困難な状況になってきた。そのため，現在の仮想プリンタの利用が，製品設計後のファームウェアチューニングや，プロセッサの動作周波数やキャッシュ容量の変更などの流用設計の確認程度にとどまっている。

我々は，この問題点を解決するために，人手によるコーディングを削減するツールを試作した。プロセッサ，バス，ハードウェアモジュールなどの各部品の組み合わせを，ブロック図で記述し，仮想プリンタソースコードを生成する仮想プリンタデザイナと，ハードウェアモジュールのモデル化を，状態遷移図を使った入力方法で作成するロジックエディタである。

## 仮想プリンタデザイナ

仮想プリンタは，命令レベルシミュレータの一種である。シミュレータ全体を管理するシミュレータ管理部と，プロセッサ，バス，ロジックなどのハードウェアをソフトウェアに置き換えたモデル，および各モデルの情報を表示するビューワで構成される[2)]。従来は，ハードウェアの方式設計後，LSI設計のためにVerilogによって記述されたソースコードを元に，C++ソースコードを人手で記述していた。仮想プリンタデザイナ（以下，デザイナ）は，Verilogソースコードの代わりに，状態遷移図とブロック図によりハードウェアを記述し，C++ソースコードを自動生成するツールである（図1）。

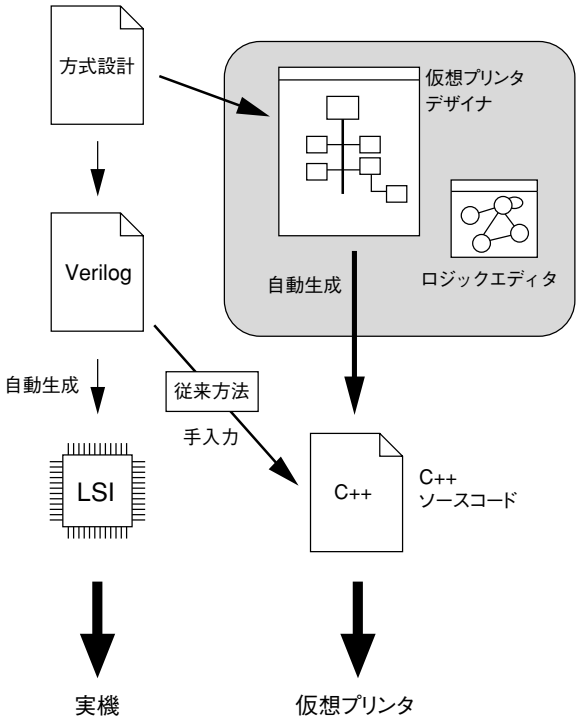


図1　仮想プリンタ作成フロー

デザイナでの入力作業は，ブロック図，メモリマップ，I/O設定などの通常のハードウェア設計と同じ作業を，ウインドウベースで行う。

デザイナのウインドウ構成を，図2に示す。各モデルを配置，配線する作画ウインドウ，作業用のツール選択ウインドウ，メモリマップウインドウであり，各モデルごと

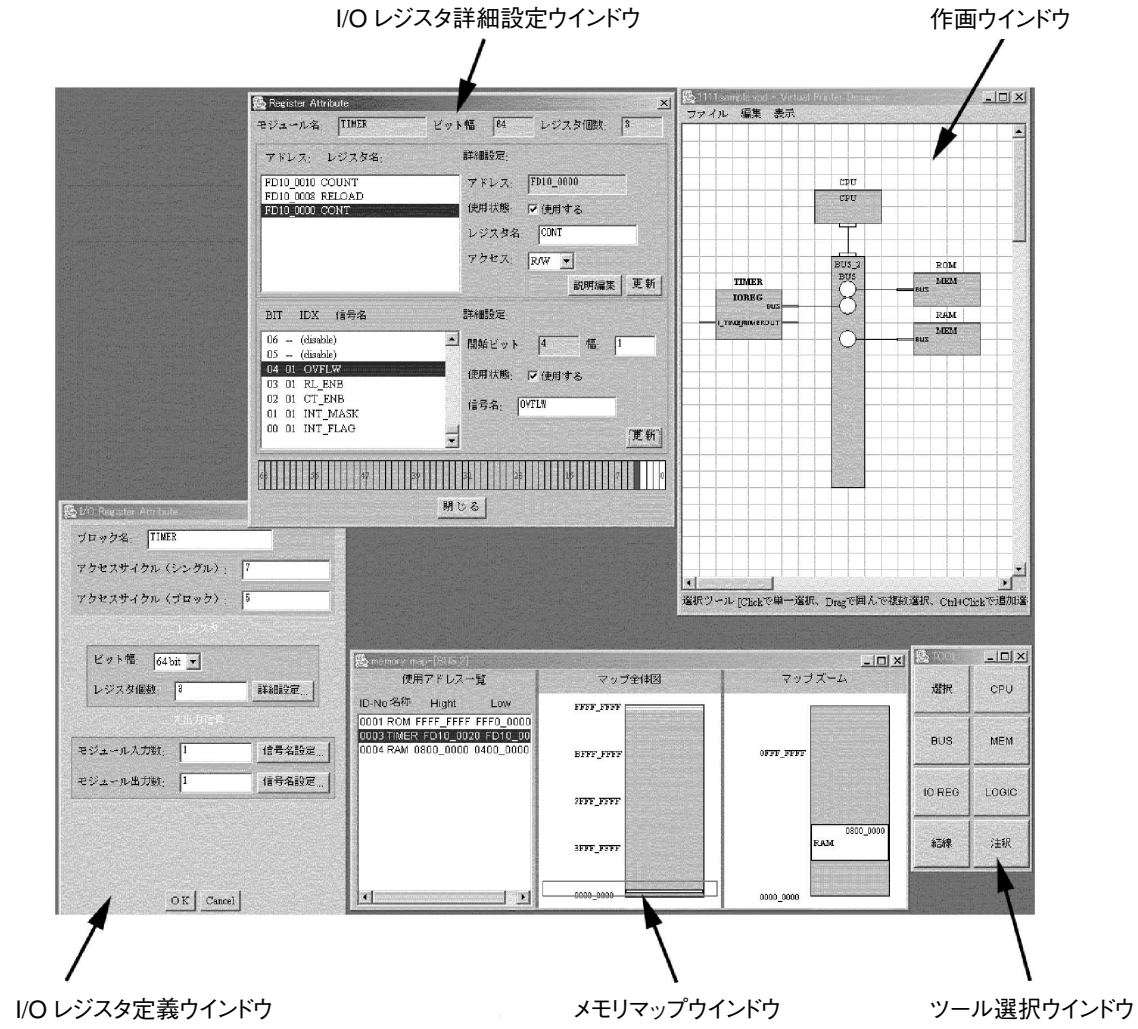


図2　仮想プリンタデザイナ外観

に，それぞれのロジック定義ウインドウを持つ。ロジックとは，ユーザによって新たに作成されるハードウェアモジュールで，ロジック定義ウインドウでは，ハードウェアモジュールの各種パラメータを定義するウインドウである。図2では，ロジック定義ウインドウの拡張であるI/Oレジスタの定義ウインドウと，その詳細設定ウインドウが表示されている。以下では，それぞれのウインドウについて説明する。

### (1) メモリマップウインドウ

メモリマップは，プロセッサバスのアドレス空間を定義するもので，通常，同一バス上に複数のハードウェアモジュールが接続され，プロセッサはこれらのモジュールを，アドレス空間内にいくつかの範囲に分割して管理している。仮想プリンタでは，アドレス情報を元に，そのアドレス空間に接続されているモデルを選択する機能をバスモデルに持たせている。ソースコード生成時には，このウインドウによって定義された内容が，バスモデルクラスから導出された形として，自動生成される。

### (2) ロジック定義ウインドウ

ロジック間の信号入出力ポート数を定義する。基本的に，仮想プリンタで動作するモデルは，プロセッサ，バスを除いて，全てロジックモデルを基本とする。ロジックモデルについては，後述するロジックエディタで詳しく述べる。

また，電子部品のみならず，エンジン，用紙などのモデルについても，このロジックモデルを基本に導出することで，仮想プリンタの構成部品として動作する。

ロジック内部の動作については，後述するロジックエディタを使って作成する。

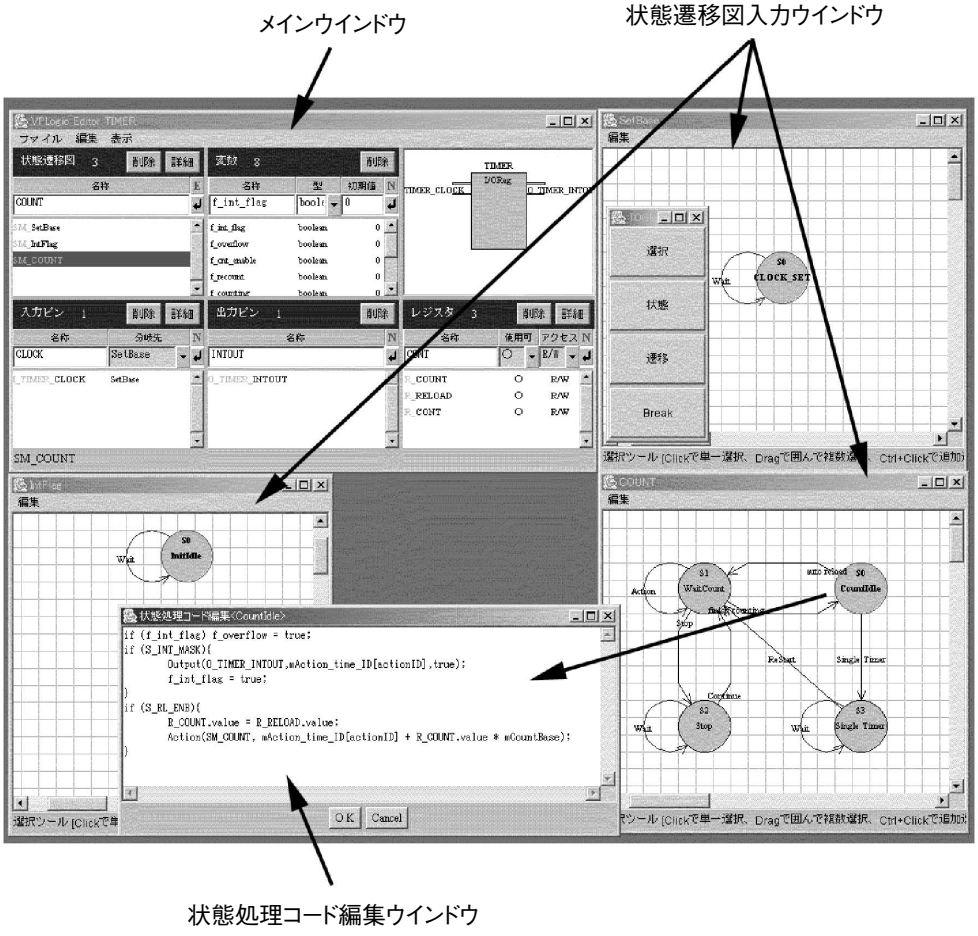


図3　ロジックエディタ外観

**(3) I/Oレジスタ定義ウインドウ**

バスに接続されるロジックは，必ずプロセッサからアクセスできるレジスタを持つ。I/Oレジスタ定義ウインドウでは，ロジック定義に加え，レジスタのビット幅，アクセス時間，個数を定義する。

## ロジックエディタ

仮想プリンタは，シミュレータ管理部と，ハードウェアモデルから構成されているが，プロセッサ，バス，ロジックというハードウェア基本クラスをシミュレータ管理部で同期をとりながら動作するようになっている。シミュレータ管理部は，これらハードウェア基本クラスとの間で信号の受け渡しを行っているため，新規ハードウェアモデルは，これらの基本クラスから導出して作成する事で，仮想プリンタの構成部品として動作可能となる。

プロセッサ，バスについては，ほとんど使うモデルが決まっており，ユーザが新規に作成する事はない。そこで，ロジック基本クラスを元に，新規ハードウェアモデルを作成するツールを別途用意した。ロジック基本クラスには，仮想プリンタで動作するために必要な最低限のコードが既に定義されており，ユーザが定義するロジックは，この定義済みのコードを暗黙のうちに使う。ユーザは，ロジックエディタ入力時，ハードウェアモデル内部の動作についてだけ記述すればよく，仮想プリンタのモデルであることを意識する必要はない。モデル内部の動作については，全て状態遷移図で記述できるように，ステートマシンとして定義した。

図3はロジックエディタの外観であり，ハードウェアモデル間の入出力信号をやりとりする入力ピン，出力ピン，内部動作を定義するための状態遷移図，および内部変数を追加するメインウインドウ，状態遷移図を入力する状態遷移図入力ウインドウ，各状態の内部処理を定義する状態処理コード編集ウインドウを示している。

一つのハードウェアモデルの内部には，複数の独立したステートマシンが定義できる。ステートマシン内部を定義するには，ステートマシン名称を登録し，詳細ボタ

ンで状態遷移図入力ウインドウが表れ，作画により記述する。ステートマシンの各状態での処理については，状態遷移図入力ウインドウで，各ステート記号をダブルクリックする事により，状態処理コード編集ウインドウが表れ，C++ソースコードで記述する。また，実際のハードウェアと同じように，内部信号状態を表す信号を内部変数として用意し，ソースコードに変換する際，そのままクラス変数としてオブジェクト内部に保持できるようにした。

仮想プリンタでは，各モデルの同期と高速化のために，一つのステートマシンが待ち状態に入った場合，他のステートマシンの処理に移るようになっている。したがって，処理が移る時点の状態を保持しておき，次の処理が回ってきたときに，前の状態から処理を再開できるようにする必要がある。この仕組みを実現するために，状態遷移記号として，「入力待ち」と「先行処理」を用意した。

「入力待ち」は，特定の条件が成立する間，同一状態にとどまる記号で，現在の状態を保持したまま，ステートマシンから抜ける。入力ピンや，レジスタアクセスなどによる条件変化待ちに対応する。

「先行処理」は，高速化のための手段で，ステートマシンが外部に影響されない範囲でシミュレーション時間を先取りする。先取りした時間をシミュレータ管理部に届ける事で，シミュレーション時間が登録された時間に到達した時，ステートマシンを再開させる。この仕組みにより，ステートのクロックごとによる実行を減らし，シミュレーション速度の低下を防ぐ事ができる。

## 仮想プリンタ生成

デザイナによる仮想プリンタ作成は，作画ウインドウに構成要素のブロックを配置し，それらを線でつなぐ事で実現する。プロセッサ，バス，メモリなどの汎用的に利用できる部品はデザイナに登録済みであり，その中から選択して配置する。ユーザが独自のハードウェアモジュールを追加する場合，ロジックエディタによって作成されたモデルをデザイナに取り込む。

作画ウィンドウで設計された情報は，仮想プリンタ生成メニューコマンドによって，新規モデルの生成，モデル間接族情報の登録コードが埋め込まれた仮想プリンタ全体のC++ソースコードに変換される。

## 評　価

ハードウェアモデルの一例として，タイマーモジュールを作成した。タイマーモジュールで作成したステートマシンCOUNTは，4つの状態を持つ。モジュールの機能をモデル化するには，この各状態での処理内容および，状態遷移条件を定義すればよい。ここで定義するコードは，ロジックエディタが生成するC++ソースコードにそのままコピーされるようになっているため，C++文法に従った記述とし，それぞれの状態記号や遷移記号をダブルクリックする事で編集ウインドウが開き，処理内容，遷移条件を記述する。各状態と，遷移記号は，人間が読みやすいように，それぞれに名称を与える事ができる。

このようにして作成したハードウェアモジュールは，コード生成メニューコマンドにより，C++ソースコードへと変換される。ここで作成されたタイマーモジュールは，ロジック基本クラスから導出されたコードになっており，そのまま仮想プリンタにおいて動作することが確認できた。

## あ と が き

ハードウェア基本クラスと状態遷移図を用いる事で，人手によるコード入力を大幅に削減できた。また，ハードウェアの具体設計でも使われる状態遷移図がそのまま使えるため，モジュール動作の視認性も高い。今後はロジックエディタを使ってモデルを記述していく事で，仮想プリンタ開発を加速させる事が期待される。

現状では，状態内部の処理や，状態遷移条件の記述にC++ソースコードをそのまま記述しているが，これもさらにアイコン化やマクロ化により簡略化したい。 ◆◆

## ■参考文献

1）尚幹夫，松代信人：プリンタシミュレータ･･･仮想プリンタ，沖電気研究開発178号，Vol.65 No.2，p.83–86，1998年5月
2）尚幹夫，松代信人：組込機器用ファームウェア最適化のためのプロファイラ型シミュレータ　情報処理学会　第54回（平成9年前期）全国大会　講演論文集（1），p.247, 248，1997年

## ●筆者紹介

尚幹夫：Mikio Takashi.株式会社沖データ NIP事業本部 画像開発グループ チームリーダ