# SIDMプリンタのシミュレータ技術

小池　徹　　白坂　光剛

近年、ハードウェアとソフトウェアの並行設計、協調設計といった設計効率のための方法論が重視され、その設計支援ツールとして、ノンインパクトプリンタ開発においては、プリンタシミュレータ[1)]が実用化されてきた。

一方、シリアル・インパクト・ドット・マトリックス（以降、SIDM）プリンタ開発では、その製品開発の性質上、プリンタシミュレータに対する開発側の欲求が少なかったこと、また、プリンタシミュレータの開発自体に多くの費用と期間が必要であったことから実用化が進んでいなかった。

しかし、SIDMプリンタ開発においても、より収益性を高めていくために、ソフトウェアの並行開発を更に加速し、短期間での製品開発を実現することが求められるようになった。そのためには、製品開発を開発要素単位で並行開発させることと、そのことで生じる「試作機台数、および、ICEなどの開発設備（開発コスト）増大の抑制」といった課題を同時に解決する必要があった。

加えて、シミュレータ開発においては、シミュレータ開発支援ツール（仮想プリンタデザイナ[2)]）が開発され、シミュレータの開発が容易に行える環境が整ってきた。

このような環境変化を背景に、前述した課題を解決する手段として、SIDMプリンタシミュレータの実用化（開発・運用）に至った。

本論文では、従来のプリンタシミュレータ機能に加え、SIDMプリンタに適応し、かつ、課題を同時に解決した機能について紹介する。

## プリンタシミュレータの概要

プリンタシミュレータは、CPU、メモリ、ロジック等の電子回路や、センサ、モータ、印字ヘッド等の機構といったハードウェアをソフトウェアに置き換えたモデルで構成される「仮想プリンタ部」と、仮想プリンタの状態を可視化し、ユーザーが直感的に操作できるようにした「グラフィカルユーザーインタフェース部」（以降、GUI部）の2つから構成されている（**図1**）。


図1　プリンタシミュレータの構成

ユーザーは、GUI部を通じて、実際のプリンタに実施するのと同じ操作を行う。

GUI部は、ユーザーが行った操作をスクリプトコマンドとして、仮想プリンタ部へ伝える。

仮想プリンタ部は、スクリプトコマンドを解析・実行することでユーザーの操作をプリンタ動作という形で実現し、その動作の結果としてのプリンタ状態をGUI部へ伝える。

そして、ユーザーはGUI部で表現されるプリンタ動作を確認することで、自らが行った操作の妥当性を検証する（**図2**）。

## SIDMシミュレータの特徴的な機能

本SIDMプリンタシミュレータで開発した特徴的な機能を以下に示す。

**① 行単位での印字結果の可視化**

SIDMプリンタの特徴である「行単位での印字」をリア

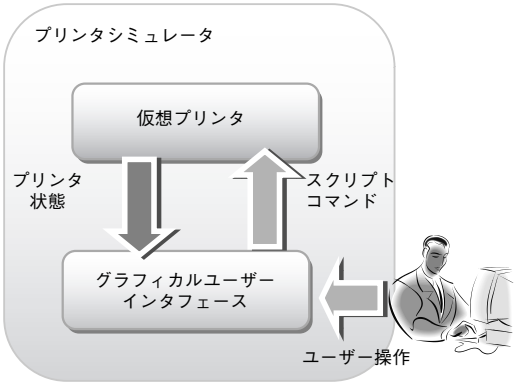


図2　プリンタシミュレータの動作

ルタイムに可視化、および、ページ単位でファイル化する機能。

**② 印字ヘッド電流波形の可視化**

印字ヘッド制御の結果である「印字ヘッド電流波形」を可視化する機能。

**③ OSとのUSB接続**

プリンタドライバ開発をプリンタシミュレータで実施する上で必要となる「OS上からプリンタシミュレータを実デバイスとして認識」させる機能。

以降にて、各機能の詳細を説明する。

## 行単位での印字結果の可視化

SIDMプリンタは印字行単位で印刷を実施する。

SIDMプリンタシミュレータでは、この行単位での印刷動作に合わせリアルタイム、かつ、用紙を使用せずに印字結果を表示する機能を開発した。

印字結果のリアルタイム表示は、用紙ビューワを実装することで実現している。

用紙ビューワは、印字ヘッドピンが着弾した箇所をドットで塗りつぶすことにより印字を表現する。

ドット印字位置は、予め実機にて測定しておいた印字ヘッドピンのドライブ開始から用紙に到達するまでの時間（以降、到達時間）と、シミュレータ内で計算した印字ヘッド移動速度から算出する。

ドット位置の算出方法は、「到達時間×印字ヘッド移動速度」よりヘッド移動距離を算出し、それに「ドライブ開始時点の位置」を加算することで求める。

| |
|---|
| ドット印字位置＝（到達時間×印字ヘッド移動速度）＋ドライブ開始時の位置 |

なお、印字ヘッドピンのドライブ信号のON/OFFタイミングはハードウェア仕様に基づいて計算する。

また、静的にも印字結果を確認できるように、用紙ごとにデータファイル化し、保存する。

印字ヘッドの位置を追跡して表示することにより、印字した瞬間をリアルタイムで見たり、印字ヘッドの水平移動の動作や用紙の垂直方向の動作を確認したりすることができる（**図3**）。

この機能によって、SIDMプリンタの装置構成に適応した印字開発環境を構築し、かつ、印字コストの低減（紙媒体の削減）を実現する。

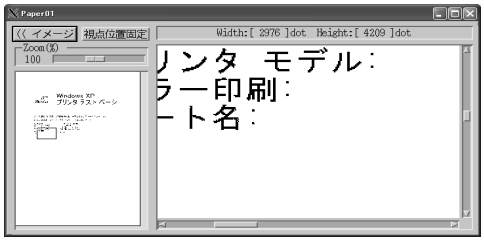


図3　用紙ビューワ外観

## 印字ヘッド電流波形の可視化

印字ヘッドピンのドライブ電流値を検証する作業において、印字パターンを印字行単位、かつ、印字ヘッドピン単位、に分解し、同一時間内にインパクトされるヘッドピンの数（以降、同時ピン数）を数えるという作業が必要である。

SIDMシミュレータでは、ファームウェアが制御している印字ドットデータ、ドットインパクトタイミング、印字ヘッドピンのドライブ時間を基に、印字ヘッドの消費電流値、および、同時ピン数、を生成・可視化し、前記検証に適した機能（印字ヘッド電流波形を可視化する機能）を実装した（**図4**）。

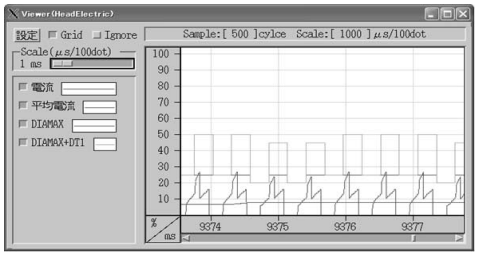


図4　印字ヘッド電流波形ビューワの表示

電流値の算出は、1マイクロ秒もしくは2マイクロ秒ごとに次のような計算式を用いて動的に行っている。

I＝((E/R−α)＊(1−Exp(−t/(L/R)))＋α)＊同時ピン数の係数＊β

I：1ピン当たりの電流
E：印字ヘッドの印加電圧[V]
R：印字ヘッドのコイル巻き線抵抗[Ω]
L：印字ヘッドのコイルインダクタンス[mH]
α：係数
β：印字ヘッド電流に対する電源電流比の係数
t：ドライブ開始から経過した時間[マイクロ秒]

続いて電流値の算出方法について手順を追って説明する。次のような計算とビューワ表示更新を定期的（ユーザー設定により1マイクロ秒もしくは2マイクロ秒ごと）に行う。

**① 同時にインパクトするピン数から「同時ピン数の係数」を決定**

計算時点における同時ピン数から「同時ピン数の係数」を求める。（1ピン→100.0%、2ピン→108.7%等）

**② ドライブしているピンの電流値を算出**

ドライブ開始から経過した時間tを基に、前記した計算式を使用してピンごとの電流値Iを算出する。なお、算出するピンは、計算時点においてドライブ開始しているものだけとする。（ドライブしていないピンは電流値0）

**③ 全てのピンの電流値を加算**

全てのピンの電流値を加算し、印字ヘッド電流値を算出する。

**④ ビューワ表示更新**

算出された印字ヘッド電流値を、同時ピン数とともにヘッド電流波形ビューワに表示する。

なお、印字ヘッド電流波形ビューワの表示／非表示は設定により変更可能であり、不要な時は非表示にしてシミュレート速度を上げることができる。また、計算式に使用するE、R、L、α、βのハードウェア特性を示す各種パラメータ値はシミュレータ上で自由に設定できる。

この機能によって、複雑な印字パターンごとの使用電流量の算出が容易になり、ハードウェア開発（印字ヘッド、電源など）の検証効率を向上する。

## OSとのUSB接続

SIDMシミュレータをWindows*1) PC上のUSBデバイスとして認識させる機能を実装した（**図5**）。

本機能は、以下に示す2つの技術を用いて実現している。
①仮想デバイスを実USBデバイスとして認識させる技術
②サーバ上のUSBデバイスをクライアント上のデバイスとして認識させる技術

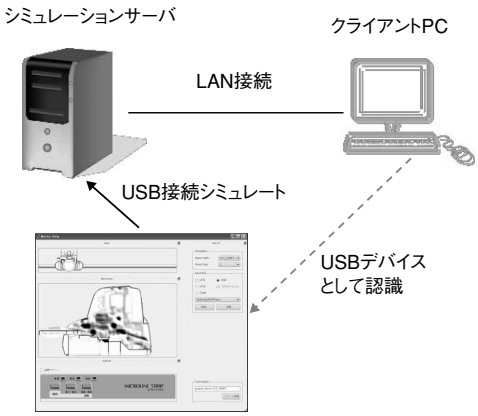


図5　USB接続シミュレートの概略図

初めに、仮想デバイスを実USBデバイスとして認識させる技術であるが、Linux*2) 上で仮想デバイスを実USBデバイスとして認識させるための仮想HCDという既存のデバイスドライバを用いている。仮想HCDとSIDMシミュレータを接続するため、以下のことを行っている。

**（1）SIDMシミュレータ動作**

ユーザーがGUI部上からUSB接続を指示すると、仮想HCDのライブラリで用意されている接続用の関数を呼び出し、仮想HCDと接続する。

**（2）仮想HCD動作**

接続関数が呼ばれると、接続元とやりとりを行ってデバイス情報を取得する。続いて取得したデバイス情報をバスドライバに通知する。これによりバスドライバはSIDMシミュレータをUSBデバイスとして認識する（**図6**）。

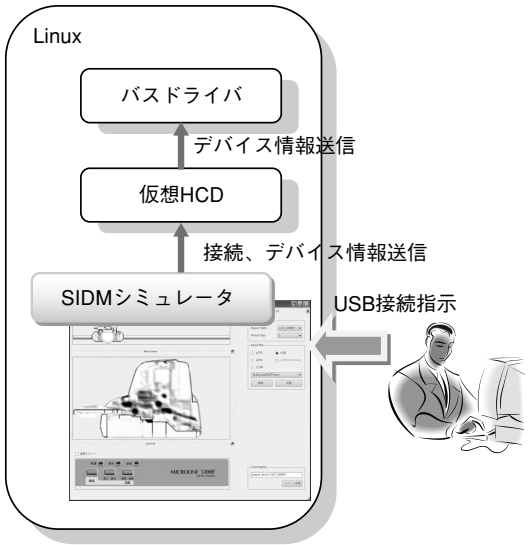


図6　仮想USBデバイスの接続動作

*1）Windowsは、米国Microsoft Corporationの米国およびその他の国における登録商標です。
*2）Linux は、Linus Torvaldsの米国およびその他の国における登録商標または商標です。

続いて、サーバ上のUSBデバイスをクライアント上のデバイスとして認識させる技術であるが、周知の技術であるUSB/IPというLAN経由でUSB接続を実現するためのアプリケーションを用いている。これはサーバ、クライアントそれぞれに実行ファイルを設け、通信を行う仕組みになっている。以下、サーバ、クライアントそれぞれの動作について説明する。

**(1) サーバ動作**

USB/IPがサーバに接続されているUSBデバイスの一覧を取得する。続いて各USBデバイスにバスNo.を付加する。

**(2) クライアント動作**

ユーザーがUSB/IPを起動し、接続するUSBデバイスのバスNo.を入力する。USB/IPはサーバ側USB/IPと通信を行い、バスNo.で指定されたデバイスのデバイス情報を得る。

サーバ側USB/IPではUSBデバイスから情報を取得し、取得したデバイス情報をクライアントへ送信する。

USB/IPがデバイス情報を得ると、PnPマネージャとデバイスマネージャにデバイス情報の通知を行い、それを受けてクライアントはUSBデバイスとして認識し、接続が完了する。

以上、仮想デバイスをサーバ上の実USBデバイスとして認識させる技術と、サーバ上のUSBデバイスをLAN接続されたWindows PC上のUSBデバイスとして認識させる技術を組み合わせることで、SIDMシミュレータをWindows PC上の実USBデバイスとして使用できるようになった。

この機能により、OSを経由しプリンタと接続しながら開発する必要があったプリンタドライバ開発においてもシミュレータによる開発を可能とする（**図7**）。

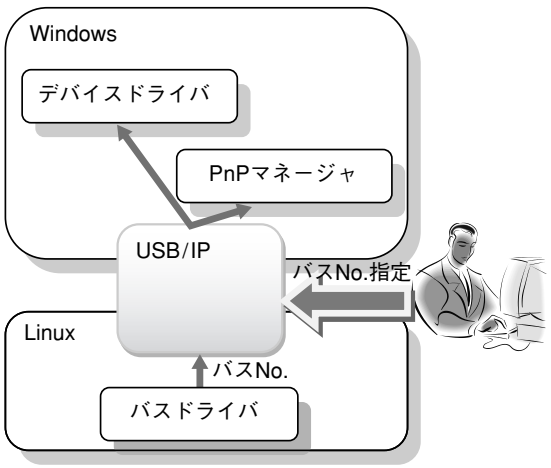


図7　LAN経由USBデバイスの接続動作

＊3）In-Circuit Emulatorは、米国インテル社の登録商標です。

## ま と め

これまで述べてきた機能を実装したプリンタシミュレータを実用化（開発・運用）したことで、SIDMプリンタ開発においても、次に示す製品開発が可能となった。

- ドライバ開発を含め、ソフトウェアの開発要素のほとんどを本シミュレータで実施できる。
- 1システム当り6〜8ユーザーの使用が可能なため、設備費用の抑制が行え、低コストで必要数の開発環境を構築できる。
- 開発過程での印字コストの低減、印字ヘッド・電源の検証効率の向上ができる。

これにより、開発コストの増加を抑制し、かつ、開発要素単位での並行開発を実施することが可能となった。

◆◆

## ■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 No2，p.83–86，1998年5月
2）尚幹夫：仮想プリンタデザイナ，沖テクニカルレビュー194号，Vol.70 No2，p.74–77，2003年4月

## ●筆者紹介

小池 徹：Toru Koike. 株式会社沖データシステムズ SIDM技術センタ ソフトウェア開発技術部 General Manager
白坂光剛：Mitsuyoshi Shirasaka. 株式会社沖データシステムズ SIDM技術センタ ソフトウェア開発技術部

## TIPS【基本用語解説】

**ICE：In-Circuit Emulator®＊3)**
マイクロコンピュータシステム（マイコン基板）を開発する際に使うデバッガ。

**USB：Universal Serial Bus（ユニバーサル・シリアル・バス）**
コンピュータに周辺機器を接続するためのシリアルバス規格の1つである。

**仮想HCD**
仮想USBデバイスをOS上に認識（ホットプラグ等の処理）させ、URB（USB Request Block）の生成および展開を行いOSと情報のやり取りを行うためのドライバ。

**USB/IP**
LinuxおよびWindows（クライアントのみ）上で動作する仮想USBバスドライバ。
Linux上に接続されているUSBデバイスを自仮想USBバスドライバへフック（デバイスバインダ）し、URBを取得してTCP/IPにカプセル化し、クライアントとの通信を行う。