視覚的シミュレータの開発支援システム†

岡田義広†† 田中譲††

VLSI 設計用の CAD をはじめとして，固定分野を対象としたグラフィカルな対話型シミュレータは，多数開発され発表されている．しかし，種々の分野にわたりシミュレーションが行えるシステムはない．そこで，著者らは，種々の分野にわたり対話型グラフィカルシミュレータを容易に開発できるツールキット・システムとして FES を開発した．FES は，以下の機能を提供する．（1）対象モデルの構成と動作機構を作図により視覚的に定義する機能（ビジュアル合成機能）の提供．（2）対象モデルの状態および動作をグラフィカルに表示する機能（ビジュアルインスペクタ機能）の提供．（3）利用者が基本モデルを定義・登録することにより，システム・ライブラリを拡張しうる機能（MV モデリング）の提供．（4）対話型操作のための統合環境の提供．本論文では，FES が提供する種々の機能と，それらを実現する機構について説明する．MV モデリングについて説明し，それが本システムを開発する上でどのように用いられているかを述べる．

1. はじめに

著者らは，1988年に，待ち行列網に基づいたシミュレーションにより，計算機のシステムレベルにおける性能評価を行える対話型グラフィカルシミュレータ CAB[1] を開発した．CAB は，対象となる計算機システムの構成要素である個々の装置あるいはデバイスをディスプレイ画面上に視覚的に表現し，その視覚化されたグラフィックイメージを画面上で自在に組合せ作図することにより，対象モデルの構成を視覚的に定義でき，種々の構成の対象モデルについて対話的にシミュレーション実験を行うことができた．

今回，このシミュレータの機能を継承し，**図1**に示すようなダイアグラムによる図表現によってその構成が表され，イベント駆動によってその動作が記述できる事物（待ち行列網，ペトリネット，論理回路など）を対象とする，種々のシミュレータの開発支援システムとして FES (Flavor Environment for Simulations) を開発した．開発は，Symbolics 3620 上で Common Lisp[2),3)] およびオブジェクト指向言語 Flavor[3)] を用いて行った．本論文では，FES の構成と機能を述べるとともに，このツールキット・システムがもつ種々の特徴を実現する機構について詳しく述べる．また，FES を用いて，現在既に開発されているシミュレータの例を挙げ，対話型グラフィカルシミュレータの開発支援システムとしての FES の有用性を示す．**図2**が，FES のシステム・ウィンドウの画面ハード・コピーである．

計算機の研究をする上で，性能評価は必要不可欠であり，シミュレータは計算機に係わる研究者，特に計算機アーキテクトにとっては，重要なツールである．計算機のシステムレベルにおける性能評価では，待ち行列網[4)-6)]シミュレーションや，ペトリネット[6),7)]シミュレーションが一般的であるが，これらのシミュレーションに加えてレジスタトランスファレベルや，論理回路のシミュレーションなど，計算機アーキテクトに必要な種々のシミュレータを容易に開発できる環境やツールキット・システムが望まれる．

しかし，従来の計算機上でのシミュレーションでは，対象モデルをシミュレーション用の記述言語あるいはプログラミング言語等を用いてテキスチャルに記述し，実行形式へ変換し，それを実行して結果を得るという手順を踏んでいた．これではモデルの動作の直感的把握と妥当性の検証ができにくく，対象モデルの構成の変更も容易ではない．そこで，シミュレーション対象のモデル定義が容易に行え，対象モデルの状態やシミュレーション結果を直感的に把握でき，対話的にシミュレーション実験が行える環境を持ったシミュレーション・システムが望まれている．

FES は，対話型でグラフィカルなシミュレータを開発するための種々のツール群を提供する．これらのツール群を用いることにより，種々の分野にわたり対話型グラフィカルシミュレータを容易に開発できる．

[FES の特徴]

FES を用いて開発されたシミュレータの特徴として，以下を挙げることができる．

対象モデルの定義は，あらかじめ定義・登録されて

† FES (Flavor Environment for Simulations): A Toolkit System for the Development of Visual Interactive Simulators by YOSHIHIRO OKADA and YUZURU TANAKA (Electrical Engineering Department, Faculty of Engineering, Hokkaido University).

†† 北海道大学工学部電気工学科

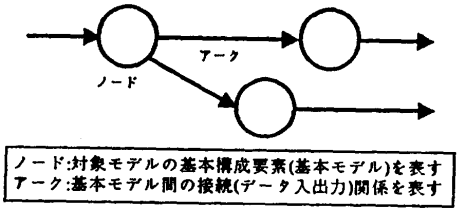


図 1 ダイアグラム
Fig. 1 Diagram.

いる基本モデルを組み合わせることにより行う．基本モデルは，画面上のグラフィックイメージとして視覚化され，そのグラフィックイメージを画面上で自在に組み合せ作図することにより対象モデルを記述することができる．これをビジュアル合成と呼ぶ．これにより，実行形式への変換をすることなく，種々の構成の対象モデルについて対話的にシミュレーション実験が行える．また，基本モデルの状態をグラフィックイメージとしてグラフィカルに表示することもできる．これをビジュアルインスペクタと呼ぶ．これにより，従来のシミュレーションで得られる数値結果だけでは把握できない対象モデルの状態や動作などを直感的に把握できる．基本モデルは，内部動作機構を定義した動作記述部 (MODEL) と視覚的形状を表す視覚表示部 (VIEW) に分けられている．これを MV モデリングと呼ぶ．基本モデルを定義・登録する場合には，VIEW はあらかじめ登録されているものを用いればよく，MODEL のみをテキスチャルに記述すればよい．新たに，異なる分野のシミュレータを開発する場合には，必要となる基本モデルの動作機構のみを，FES の記述言語である Flavor を用いて定義・登録すればよい．シミュレーションの動作方式については，イベント駆動方式で動作するシミュレーションドライバ[5] が用意されている．また，対話型シミュレータに必要な入出力作業のための種々のウィンドウがあらかじめ用意されており，シミュレーションのための統合環境を提供する．

[関連する研究]

関連する研究としては，VLSI 設計を対象として開発された SIMMER[8], VEGAMES[9], INSIST[10], STEM[11] 等のシステムがある．SIMMER はシミュレーション用言語，VEGAMES は視覚的なインタフェイスを実現したシミュレーション用言語である．INSIST, STEM は，対話型グラフィックスを実現したシミュレータである．

Multicomputer System の性能評価を目的に開発さ

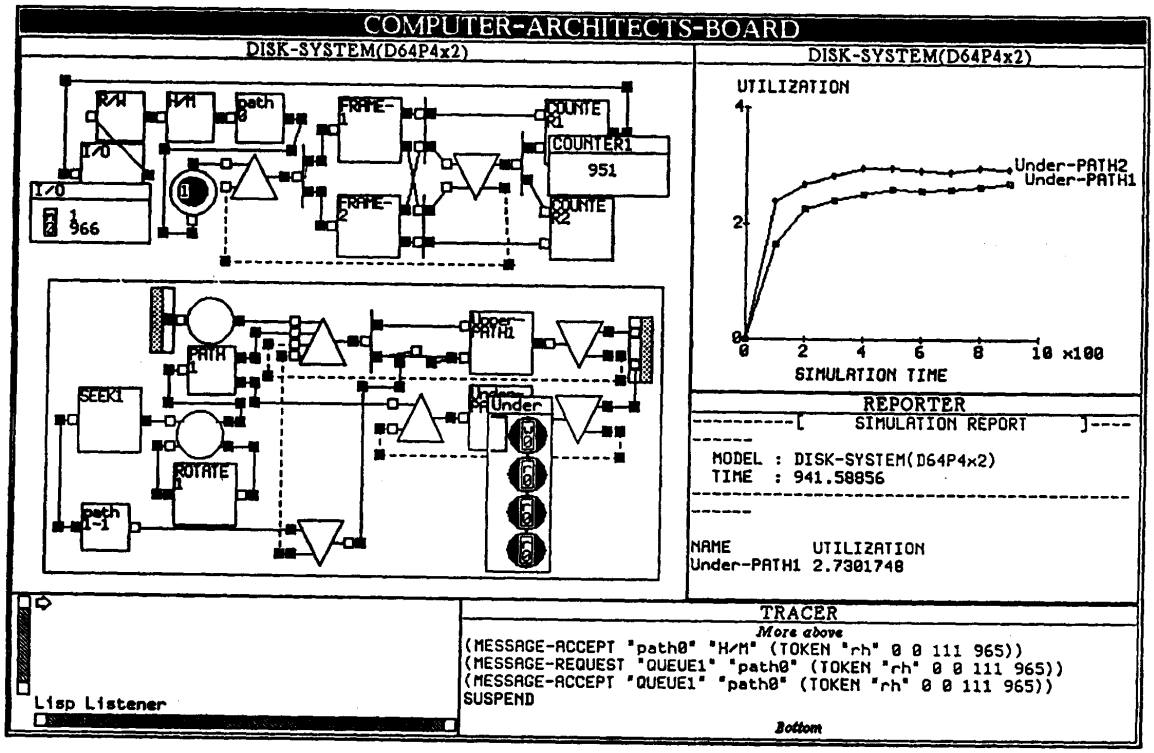


図 2 Simulation-Window の画面ハード・コピー
Fig. 2 A screen hardcopy of the Simulation-Window.

れ，対話型グラフィックスを実現したシミュレータには，PARET[12] 等がある．また，Queueing-Model をグラフィカルに視覚化した待ち行列網シミュレータには，B. Melamed らによって開発されたシステム[13] 等がある．アナログ回路シミュレータ SPICE と接続して，視覚的な操作環境を提供するものに M. G. Walker らの開発したシステム[14]等がある．この外，対象を VLSI とした CAD 用シミュレータは多数開発されている．

一般に，VLSI を対象とした CAD 用シミュレータでは，視覚的インタフェイスや対話的な操作環境を備えたものがほとんどである．しかし，シミュレーションの対象分野を VLSI に固定して開発されたものであり，種々の分野をシミュレーション対象として扱うことはできない．また，種々の対象を扱える対話型でグラフィカルなシミュレータ・システムは著者の知る限り発表されていない．FES は，ダイアグラムによる図表現によってその構成が表され，イベント駆動によってその動作が記述できる事物であれば，シミュレーション対象として扱うことができる．MV モデリングと呼ぶモデリング手法を用い，種々の機能を持つ VIEW をあらかじめ定義・登録しておくことにより，対象モデルの動作機構のみを定義・登録するだけで，対話型グラフィカルシミュレータを容易に開発できる．

以下では，第2章で FES の設計思想として，MV モデリングについて説明する．第3章で FES のシステム構成について説明する．第4章で，FES の応用システムとして開発したシミュレータを例に挙げて FES の種々の機能を解説する．第5章にまとめを述べる．

2. FES の設計思想

一般に，対話型グラフィカルシミュレータは，以下に挙げる機能を有する．

(1)対象モデルの構成を視覚的に定義する機能．(2)対象モデルの状態や動作状況をグラフィカルに表示する機能．(3)シミュレーション結果を種々の形式で表示する機能．(4)対話型操作のための統合環境．これらは，シミュレーションを行う上で必要な入出力作業を，視覚的かつ直接的に行うための機能であり，シミュレーション対象のモデルの動作機構には依存しない．そこで，本システムでは，MV モデリングと呼ぶモデリング手法を用いることにより，シミュレータの機能を，対象モデルの動作機構を定義する部分と視覚的な入出力機能であるビジュアルインタフェイス部分とに分けた．**図3**に示すように，FES は，ビジュアルインタフェイス部分を汎用ツールの形式で提供するものである．これにより，シミュレータの対象モデルの動作機構である MODEL 部のみを定義するだけで，視覚的インタフェイスを備えたシミュレータを容易に開発できる．

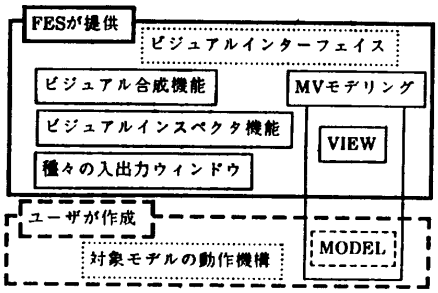


図 3 FES が提供するビジュアル機能
Fig. 3 Visual facilities provided by FES.

以下では，MV モデリングについて詳しく説明する．また，本システムの特徴であるビジュアル合成機能とビジュアルインスペクタ機能について述べ，MV モデリングがどのように用いられているかを述べる．

2.1 MV モデリング

Smalltalk-80[15] のウィンドウ・システムに用いられている MVC (MODEL, VIEW, CONTROLLER) によるモデリング手法は，対象オブジェクトのモデリングを，内部機構を表す MODEL 記述部，形状を表す VIEW 記述部，マウスやキー入力に対する反応を定義する CONTROLLER 記述部の3部分に分けて定義する抽象化手法である．**図4**に示すように，本システムでは，MVC における，C の機能を限定し，V にその機能を含めることによって，M と V の2部分による記述手法を採用した．これを MV モデリングと

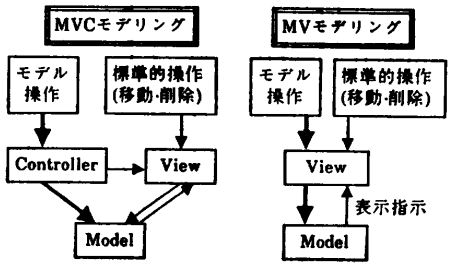


図 4 MVC モデリングと MV モデリング
Fig. 4 The MVC modeling and the MV modeling.

呼ぶことにする．シミュレーションの対象モデルは基本モデルの組合せで構成され，各基本モデルは，視覚的形状を表すVIEWと内部動作機構を表すMODELの対によってモデリングされる．

図4に示すように，MVCモデリングでは，Mに対するユーザ操作は，主にCを介して行われる．C部を他のCに換えることによって，ユーザ操作に対するMの反応を換えることができ，モデルの記述力を高めることができる．しかし，MとVとCの3部分を記述しなければならなく，モデルの記述量が増える．けれども，Cの機能が限定されており，Mに対するユーザ操作が常にVを介して行われるならば，CをVに含めることができ，MとVの2部分によるモデリング手法を用いることができる．

本システムでは，基本モデルに対するユーザ操作を，MODELに対するもの（保持する属性値や結果の表示および変更に限定）と，VIEWに対するもの（移動や複写・削除などに限定）とに分けた．VIEWは，ディスプレイ画面上のグラフィックイメージとして視覚化されており，ユーザ操作はすべて画面上で，VIEWを介して行うものとした．ユーザ操作が，移動や複写などのVIEWに対するものであればVIEWがそれを処理する．属性値の表示や変更などのMODELに対するものであれば，VIEWからMODELへ対応するメッセージが送られ，送られたメッセージにしたがってMODELがそれを処理する．このように，VIEWに対する操作とMODELに対する操作を明確にし，その種類を限定することで，VIEWをあらかじめ定義・登録しておくことができる．

2.2 ビジュアル合成機能

ビジュアル合成機能とは，ディスプレイ画面上に基本モデルの実体が存在するかのごとく，移動や複写・削除などの操作を視覚的に行える環境を提供し，ディスプレイ画面上で作図することにより，対象モデルの構成の定義が行える機能である．これは，ビジュアル・プログラミング[16]の手法であるダイアグラマティック・プログラミングやアイコニック・プログラミングで用いられるインタフェイスと同様の機能を実現したものである．

MVモデリングでは，VIEWはMODELの定義には依存せず，あらかじめ定義・登録しておくことで汎用に使えるものである．そこで，ダイアグラムのノードとアークにあたるVIEW（後述するPorts-VIEWとLine-VIEW）を用意した．これにより，ダイアグラムによる図表現によってその構成が記述できる事物であればシミュレーション対象として扱うことができるようになった．

2.3 ビジュアルインスペクタ機能

ビジュアルインスペクタ機能とは，基本モデルの詳細な状態をディスプレイ画面上のグラフィックイメージとして視覚表示するための機能である．基本モデルの視覚表示部Ports-VIEWは，対象モデルの構成を定義する場合に用いるアイコンでしかない．そこで，基本モデルの状態を表示するための専用の視覚表示部（後述するDetail-VIEW）を設けた．

一般に，グラフィカルな表示機能を実現する場合，そのための処理によるオーバヘッドが生じ，全体の処理速度を低下させる．ビジュアルインスペクタ機能では，Detail-VIEWの表示，非表示を指定することができる．Detail-VIEWが表示されている場合には，対応する基本モデルの状態や動作がグラフィカルに表示される．Detail-VIEWが表示されていない場合には，グラフィカルな表示はされず，シミュレーションの実行速度の低下を避けることができる．

3. FESのシステム構成

FESは，シミュレーションを対話的に行うための統合環境を提供するウィンドウ・システムと視覚的な入出力機能を提供するMVシステムで構成されている．また，イベント駆動方式で動作するシミュレーション・ドライバを用意している．以下，それぞれに関して詳しく説明する．

3.1 ウィンドウ・システム

FESでは，対象モデルの構成の定義，シミュレーション結果のグラフ表示，レポート作成，イベント・トレースの各機能に対応して，以下に挙げる4種類のウィンドウがシステム・ウィンドウ（Simulation-Window）のサブ・ウィンドウとして用意してあり，図5に示すように，シミュレーションのための統合環境を提供する．

（1） Edit-Window：シミュレーションの対象モデルを記述したり，シミュレーションの実行や結果の表示などの指示をするためのウィンドウである．

図6に示したように，VIEWは，Edit-Window内に存在し，Edit-Window内でのマウスやキー入力は，一度Edit-Window のプロセスに渡される．この入力がEdit-Window上の特定のVIEWに対するものであれば，Edit-Windowは，この入力メッセージをそ

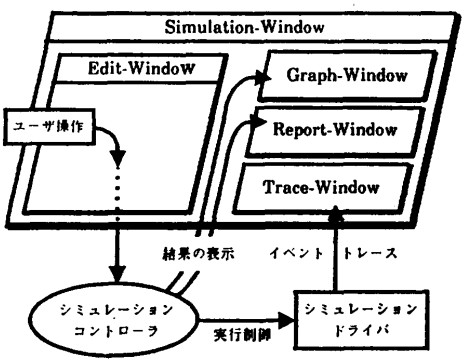


図 5 ウィンドウ間の関係
Fig. 5 The relationship among Windows.

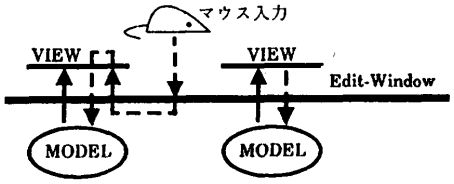


図 6 Edit-Window と VIEW, MODEL の関係
Fig. 6 The relationship among Edit-Window, VIEW and MODEL.

の VIEW に渡す. 各 VIEW には, 一つの MODEL が割り当てられており, VIEW に送られた入力メッセージがその MODEL によって解釈されるものであれば, このメッセージは MODEL に渡される. MODEL と VIEW は互いに他を参照でき, メッセージを送ったり, 必要なデータを渡すことができる.

(2) Graph-Window: シミュレーションによって得られた数値結果を種々のグラフに表示するためのウィンドウである.

(3) Report-Window: 各種数値結果を任意の書式にフォーマットして表示するためのウィンドウである.

(4) Trace-Window: イベント駆動によって動作しているシミュレータのイベント生起過程をトレースし保存するためのウィンドウである.

これらのウィンドウでは, すべての操作をマウスを用いて行え, 対象モデルの構成の定義, シミュレーションの実行, 結果の表示のすべてを対話的に行うことができる. 先に示した図2が, Simulation-Window の画面ハード・コピーである. 左上が Edit-Window, 右上が Graph-Window, 右中が Report-Window, そして, 右下が Trace-Window である.

3.2 MV システム

MV システムは, ビジュアル合成機能, ビジュアルインスペクタ機能および MV モデリングを提供するものである. これらの機能を提供するために, 以下に示す4種類の VIEW があらかじめ用意されている. これらの VIEW が持つスロット (インスタンス変数) の種類と MODEL が持つスロットの種類を図 7 に示す. 図中において使われているスロット名とクラス名の枠線は, そのスロットに対応するオブジェクトがどのクラスに属するかを示すものである.

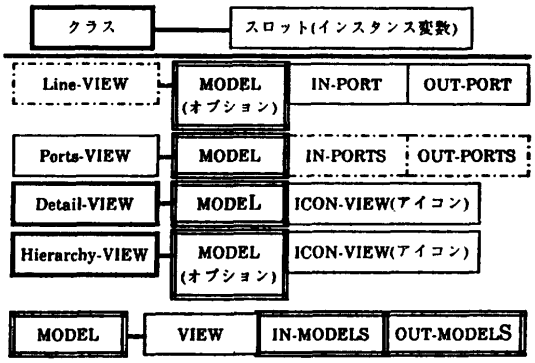


図 7 VIEW と MODEL が持つスロットの例
Fig. 7 Example slots of VIEW and MODEL.

(1) Ports-VIEW & Line-VIEW: 図 8 左にグラフィックイメージが示されているように, それぞれダイアグラムのノードと, アークにあたる VIEW である. Ports-VIEW は任意の数の入出力ポートを持ち, Line-VIEW は一対の入出力ポートを持つ. Ports-VIEW と Line-VIEW の入出力ポートのそれぞれを画面上で接続することで, ダイアグラムを構成することができる. Line-VIEW のスロット IN-PORT, OUT-PORT には, 入出力ポートに接続されている Ports-VIEW がそれぞれ格納される. 同様に, Ports-VIEW のスロット IN-PORTS, OUT-PORTS には, 入出力ポートに接続されている数個の Line-VIEW がリストとしてそれぞれ格納される. Line-VIEW によって接続された二つの Ports-VIEW 間の入出力関係は, それらの Ports-VIEW に割り当てられている MODEL 間の入出力関係となる. Ports-VIEW は, 割り当てられている MODEL を参照するためのスロット MODEL を持つ. 同様に, MODEL は, VIEW を参照するためのスロット VIEW を持ち, スロット IN-MODELS と OUT-MODELS には, その MODEL の入力側, 出力側の MODEL がそれぞれリストとして格納される. 図8右は, 図8左の構成の一部分について, スロットの参照関係を示したものである.

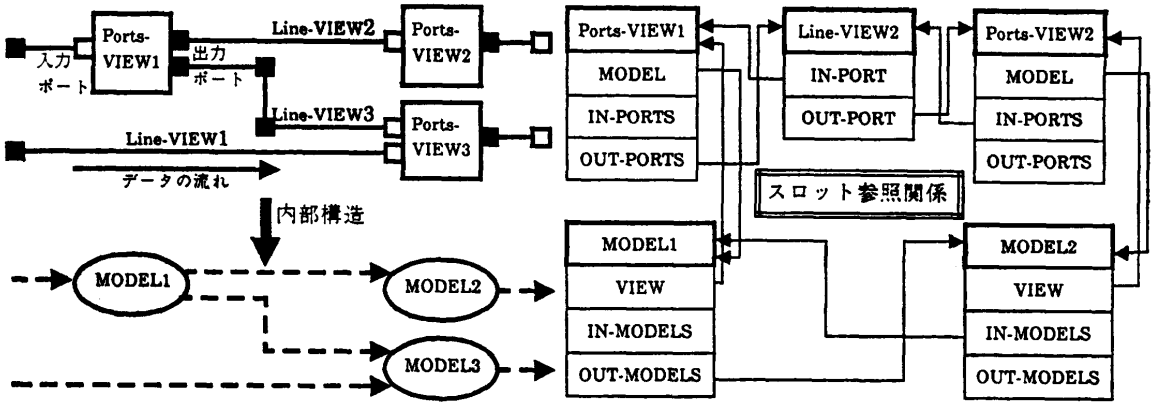


図 8 Ports-VIEW と Line-VIEW のグラフィックイメージおよびそれらの内部構造とスロット参照関係
Fig. 8 Graphical images of Ports-VIEW and Line-VIEW, their internal structures and their internal relationship through their slot reference.

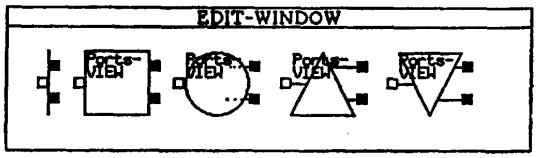


図 9 Ports-VIEW が持つフォームの例
Fig. 9 Example forms of Ports-VIEW.

図 9 に示すように，Ports-VIEW は種々のフォーム（長方形，三角形，円など）を持ち，パラメータを変えることで，表示されるグラフィックイメージを変えることができる．任意のフォームを新たに作成し，表示させることも可能である．対応する MODEL の種類により，表示されるフォームを変えることで，その VIEW に対応する MODEL が何であるかを視覚的に知ることができる．MODEL が割り当てられていない Ports-VIEW は，以下に述べる Detail-VIEW や Hierarchy-VIEW のアイコンとして用いられる．

（2） Detail-VIEW：これは，MODEL の状態をグラフィカルに表示するための VIEW である．この VIEW は，ディスプレイ画面上にグラフィカルな表示が行えるような領域を提供するもので，表示されるグラフィックイメージは，MODEL の内部で定義される．**図 10** に示すように，Detail-VIEW をアイコン化すると Ports-VIEW となり，この Ports-VIEW の入出力が，Detail-VIEW に割り当てられている MODEL の入出力となる．Detail-VIEW のスロット ICON-VIEW には，アイコンである Ports-VIEW が格納される．

（3） Hierarchy-VIEW：対象モデルを階層的に記述するための VIEW であり，この内部で，対象モデルの部分系を記述できる．**図 11** に示すように，Hierarchy-VIEW もアイコン化すると Ports-VIEW となり，一つの構成要素として用いることができる．この Ports-VIEW の入出力は，Hierarchy-VIEW の内部で記述されている部分系の入出力となる．多くの構成要素からなる対象モデルを階層的に記述できる．

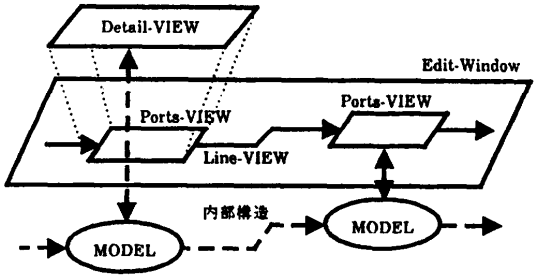


図 10 Detail-VIEW, Ports-VIEW と MODEL の関係
Fig. 10 The relationship among Detail-VIEW, Ports-VIEW and MODEL.

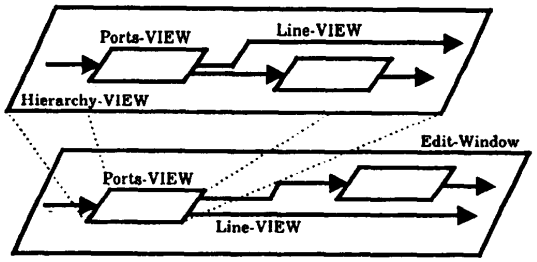


図 11 Hierarchy-VIEW による階層構造
Fig. 11 Hierarchical structuring using Hierarchy-VIEW.

3.3 動作方式

FES では，イベント駆動方式で動作するシミュレーション・ドライバがあらかじめ用意されており，

イベント駆動による動作記述が可能な種々の事物をシミュレーション対象として扱うことができる．イベント駆動方式では，イベントを格納しているリスト（イベントキューと呼ばれる）の内容を変えなければ，シミュレーションの途中で中止や再開が自由にできる．これにより，対話的操作が可能である．

4. FES の応用システム例

FES の応用システムとして，待ち行列網シミュレータ CAB，ペトリネットシミュレータ PAB，および論理回路シミュレータ LAB が，現在までに開発されている．CAB は，計算機システムの性能評価ツールとして開発されたもので，CAB 独自の種々の特徴的な機能を持ち，既に著者らの研究室等で活用されている．本章では，CAB の持つ種々の特徴を述べ，待ち行列網シミュレータとしての CAB の有用性を示すとともに，視覚的シミュレータの開発支援システムとしての FES の有用性を示す．

4.1 待ち行列網シミュレータ：CAB (Computer Architect's Board)

CAB は，待ち行列網シミュレータであり，計算機のシステムレベルにおける性能評価を目的に開発したものである．FES の持つ MV モデリング機能により，待ち行列網を記述するために必要となる基本モデルの動作機構のみを，テキスチャルに定義・登録するだけで開発することができた．先に示した図2が，CAB の画面ハードコピーであり，128 ボリュームから成るキャッシュ・メモリ付きのディスク・システムのシミュレーションを行った例である．

4.2 CAB の特徴

CAB では，FES の特徴であるビジュアル合成機能を用いて，テキスチャルな記述をすることなく対象となる計算機システムを視覚的に記述でき，種々の構成の対象システムについて対話的にシミュレーションを実行することができる．構成要素である基本モデルの動作状況もグラフィカルに表示され，シミュレーション中の対象モデルの動作過程を視覚的に理解できる．さらに，CAB 独自の拡張機能として以下の特徴を持つ．

（1） 状態遷移による動作記述

待ち行列網シミュレーションでは，個々の待ち行列 (Queueing-Model) における窓口 (Server) でのサービス時間と，ジョブ (Token) の到着時間間隔が重要なパラメータである．これらのパラメータの種々の値

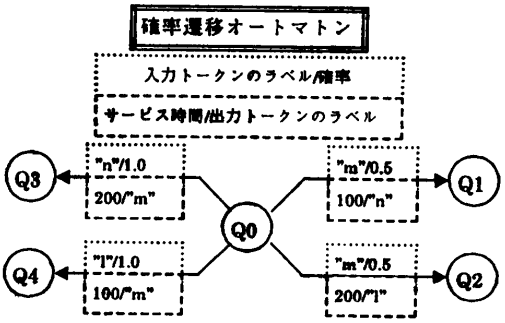


図 12 サーバ動作の状態遷移による表現
Fig. 12 Representation of the Server behavior as a state-transition automaton.

についてシミュレーションすることで，網全体の性能評価を行いボトル・ネックを見つけることができる．しかし，窓口でのサービスの内容については考慮されないため，サービスの内容を細かく記述できない．そこで，CAB では Server を確率遷移オートマトンとしても記述できるようにし，Server に内部状態を持たせ，Queueing-Model への入力である Token にラベルを与え，図 12 に示すように，このラベルの種類によって，Server での処理時間を変えたり，出力する Token のラベルの種類を決めることができるようにした．これにより，通常の Queueing-Model では表現できない細かな動作のモデリングも可能となった．

（2） 対象システム記述における簡便性

CAB では，シミュレーションの対象となる計算機システムの動作を簡便に記述できるよう，後述するいくつかのモデルをあらかじめ用意している．対象となる計算機システムは，待ち行列網モデルで表される．Token のフローに関して種々の形態の制御が行えるように，いくつかのフロー制御用のモデルを用意した．例えば，ペトリネットにおけるプレースとトランジションと同様の機能を持つモデルを用意した．これにより，要求駆動型の処理 (Token) の流れも実現でき，計算機システムの動作記述における柔軟性が増した．

4.3 待ち行列網の記述用モデル

待ち行列網シミュレーションにより計算機システムの性能評価を行うために，あらかじめ定義・登録されている基本モデルには以下に挙げるものがある．(図 13 および図2参照)

（1） TOKEN：ジョブ (処理) を表すモデルである．処理の種類を示すラベルと優先順位を示すプライオリティ・ナンバを要素として持つ．また，フロー制

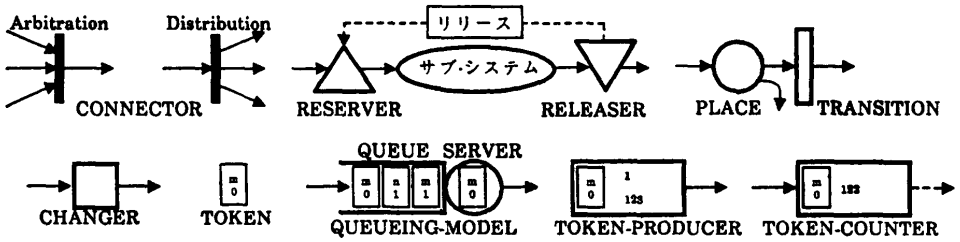


図 13 待ち行列網システムの記述用モデル
Fig. 13 Description models for Queueing-model network systems.

御のための種々の要素（アドレス，パス番号など）を持つ．図 13 に示すようなグラフィックイメージで表示される．

（2） CONNECTOR：多入力，多出力であり，TOKEN のフロー制御のためのモデルである．乱数方式や，ラウンドロビン方式により，出力先を決めることができるだけでなく，TOKEN の持つ要素（ラベル，アドレス，パス番号など）の値に従って，出力先を決めることもできる．

（3） RESERVER & RELEASER：リソース使用の排他制御をするためのモデルである．TOKEN が RESERVER を通過すると，この RESERVER に対応する RELEASER をその TOKEN が通過するまで，その RESERVER はロックされ，別の TOKEN を通過させることができない．

（4） PLACE & TRANSITION：これもフロー制御のためのモデルであり，同期および排他制御の記述に用いられる．PLACE は，待ち行列あるいはバッファとして機能し，その動作は，ペトリネットにおけるプレースと同様であり，排他制御に用いることができる．PLACE の内部には，待ち行列（バッファ）の長さがグラフィカルに表示される．TRANSITION も，ペトリネットにおけるトランジションと同様の動作をし，同期制御に用いることができる．

（5） CHANGER：TOKEN が持つ種々の要素（ラベル，アドレス，パス番号など）の値を強制的に変更するためのモデルである．

（6） QUEUEING-MODEL：サービスをする窓口 (SERVER) と待ち行列 (QUEUE) を表すモデルである．前述したように，TOKEN を入出力とした確率遷移オートマトンとしてサービスすることもできる．

（7） TOKEN-PRODUCER：TOKEN をある時間間隔（確率分布）で生成し，システムに送り出すモデルである．

（8） TOKEN-COUNTER：システムで処理を終えた TOKEN の回収を行ったり，TOKEN の通過量を計りシミュレーションの終了制御を行うモデルである．

以上の基本モデルのうち，(2), (3), (4), (5)は，Ports-VIEW に割り当てられる MODEL として，その動作機構のみがテキスチャルに定義されている．図 13 に示すように，モデルの種類によって Ports-VIEW のフォームを変えており，Ports-VIEW に割り当てられている MODEL が何であるかをディスプレイ画面上で識別できる．

また，(6)，(7)，(8)は，Detail-VIEW に割り当てられる MODEL として定義されており，図 13 に示すようなグラフィックイメージで表示される．QUEUEING-MODEL では，QUEUE と SERVER の内部に TOKEN のグラフィックイメージが表示され，待ち行列の長さやサービスの状態が視覚的に把握できる．TOKEN-PRODUCER では，システムに送り出した TOKEN の種類と数がグラフィカルに表示される．TOKEN-COUNTER では，通過した TOKEN の数がグラフィカルに表示される．シミュレーション中には，これらのグラフィックイメージがリアルタイムで変化し，対象システムの動作状況を，アニメーションを見るごとく視覚的に把握できる．

上述したこれらのモデルを用いることで，TOKEN のフロー制御を柔軟に表現でき，種々の機構を持つ対象システムを記述できる．MV モデリングにより，対象となる計算機システムを構成している装置あるいはデバイスの動作機構のみをプログラムにより定義し，新たな MODEL として登録することができ，その固有の動作を細かく記述することも可能である．また，Hierarchy-VIEW を用いて，対象システムの部分システムをこの VIEW の内部で記述することで，階層的に構成の定義を行うことができる．

図 13 に示した各モデルは，それぞれプロパティおよび結果を保持している．**図 14** に示すように，プロ

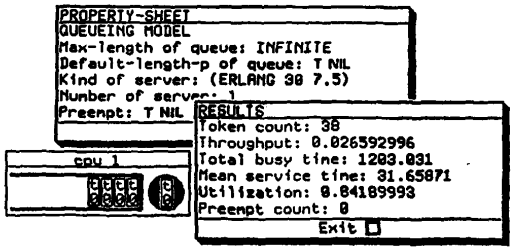


図 14 QUEUEING-MODEL のメニューシート
Fig. 14 Menu sheets of QUEUEING-MODEL.

パティの設定や結果の表示は，専用のウィンドウ（プロパティシートおよびリザルトシート）を開いて，メニューを選択することにより対話的に行うことができる．

4.4 性能について

図 15 は，最も簡単な待ち行列網モデルである M/M/1[4)~6)] モデルを，CAB を用いて記述した画面ハードコピーである．TOKEN-PRODUCER と QUEUEING-MODEL，および TOKEN-COUNTER がそれぞれ 1 個ずつの合計 3 個のモデルで構成されている．10,000 個の TOKEN を発生させてシミュレーションした場合の時間を示す．

図 15 は，Detail-VIEW の表示が行われているものである．シミュレーション時間は，約 8 分 26 秒であった．Detail-VIEW の表示が行われていない場合のシミュレーション時間は，約 4 分 15 秒であった．1 個の TOKEN が三つのモデルの間で処理される平均時間は，グラフィカルな表示をした場合 0.05 秒，グラフィカルな表示をしない場合 0.025 秒である．したがって，十分に実用に耐える処理速度であると思われる．

また，本シミュレータは，FES を用いることで，基本モデルの動作機構のみをテキスチャルに定義するだけで開発できた．参考文献 1)ですでに開発されていた旧 CAB と本 CAB のプログラミング量を比較する意味で，概算ではあるが，ソース・プログラムのテキスト量を**表 1** に示す．

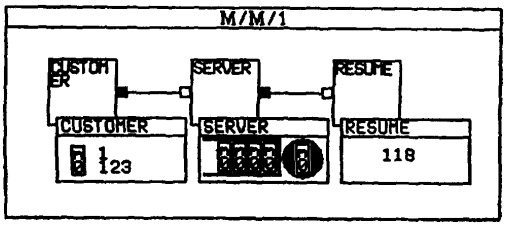


図 15 CAB を用いて M/M/1 モデルを記述した例
Fig. 15 An example description of an M/M/1 model in CAB.

表 1 システムごとのソースプログラムのテキスト量
Table 1 The amount of text in the source program of each system.

・旧 CAB	5,500 行
・FES (共通部分)	7,500 行
・CAB (基本モデルの定義部分)	3,400 行
・LAB (基本モデルの定義部分)	1,300 行
・PAB (基本モデルの定義部分)	700 行

本 CAB は，旧 CAB よりも多くの基本モデルが用意されており，対象システムのモデル記述力が高い．しかし，FES を用いることで，旧 CAB に比べて格段に少ないプログラミング量で開発できた．

4.5 その他の応用システム

FES を用いて開発されたシミュレータには，CAB のほかに，ペトリネットシミュレータ：PAB (Petri-net Analyzer Board) と論理回路シミュレータ：LAB (Logical-circuit Analyzer Board) がある．いずれのシミュレータも FES の提供するビジュアル合成機能やビジュアルインスペクタ機能を特徴として持つ．また，MV モデリングにより，基本モデルの動作機構のみをテキスチャルに定義・登録するだけで開発できた (表 1 参照)．

5. おわりに

本論文では，著者らが開発した FES シミュレーションシステムの仕様および機能について述べた．MV モデリングについて説明し，FES の特徴であるビジュアル合成機能とビジュアルインスペクタ機能の実現に，それがどのように用いられているかを示した．また，FES を用いて開発されたシミュレータの例を挙げ，FES の有用性を示した．特に，ビジュアル合成による対象モデルの構成の視覚的定義機能と，ビジュアルインスペクタによる対象モデルの状態・動作のグラフィカル表示機能が，本システムの応用例として開発したシミュレータにどう活かされているかを示した．

図 16 に示すのは，LAB の画面ハードコピーである．M.G. Walker らの開発したシステム[14)]のように，FES の持つビジュアル合成機能を用いることで，既存の論理回路シミュレータに対する回路図エディタとして，LAB を用いることができると思われる．また，論理回路シミュレータをはじめとする既存のシミュレータについても，本論文で説明した FES の機能や考え方を採用することで，機能的に等価なものを，

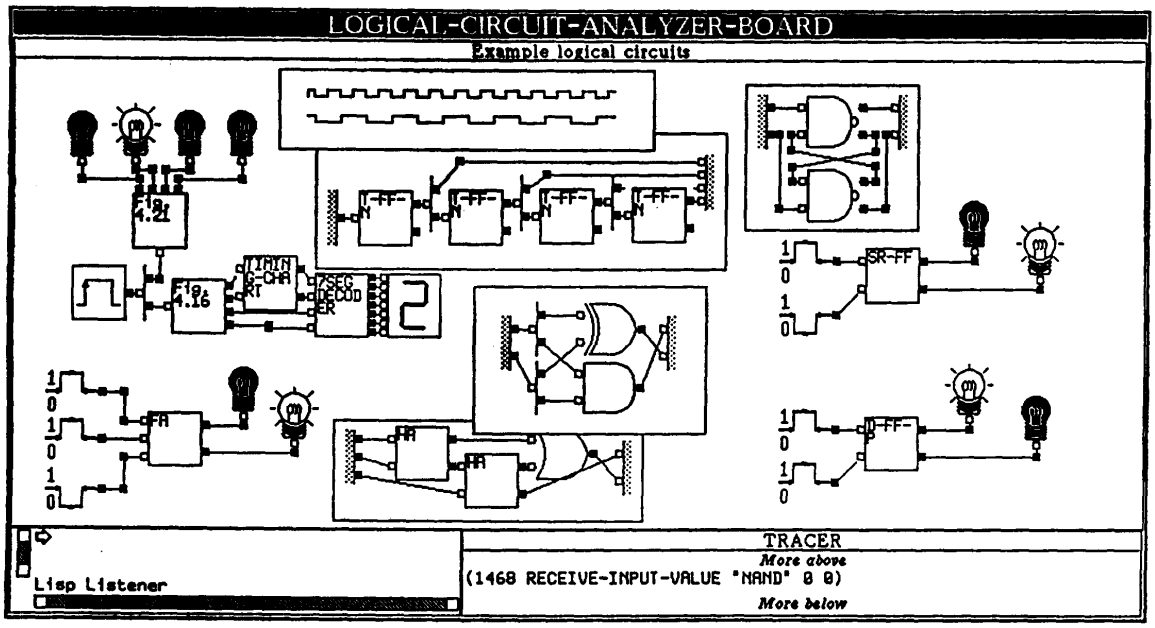


図 16 LAB を用いて記述した論理回路システムの例
Fig. 16 Example logical circuits described in LAB.

比較的容易に開発できるようになると思われる.

FES が提供するシミュレーションドライバは, イベント駆動方式で動作するもののみであるので, FES を用いて開発可能なシミュレータの対象分野は制限される. 他の方式で動作するシミュレーションドライバを新たに用意することで, より広範な分野をシミュレーション対象として扱えると思われる.

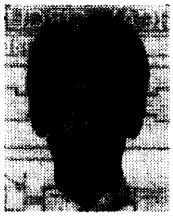
現在 CAB は, 計算機システムの性能評価ツールとして実際に活用されている. 種々の機能を持つ基本モデルを提供しており, これらを用いることで, きめ細かなモデリングを行うことができ, 共有メモリ・アーキテクチャや, ディスクキャッシュ・システムの性能評価において有効に活用されている. これらの結果についても早期に発表したいと考えている.

参 考 文 献

1) 岡田義広, 田中 譲: 対話型ビジュアル・シミュレータ CAB, 情報処理学会計算機アーキテクチャ研究会報告, 77-5, pp. 35-42 (1989).
2) 湯浅太一, 萩谷昌己(共著): Common Lisp 入門, 岩波コンピュータサイエンス (1987).
3) Bromley, H.: *LISP LORE: A Guide to Programing the Lisp Machine*, Kluwer Academic Publishers (1986).
4) Gelenbe, E. and Mitrani, I. (共著), 秋丸春夫, 橋田 温(監訳): 計算機システムの解析と設計, オーム社 (1988).
5) Macdougall, M. H.: *Simulating Computer Systems: Techniques and Tools, Computer Systems Series*, The MIT Press (1987).
6) Marsan, M. A., Balbo, G. and Conte, G.: *Performance Model of Multiproceessor Systems, Computer Systems Series*, The MIT Press (1986).
7) W. ライシッヒ(著), 長谷川健介, 高橋宏治(訳): ペトリネット理論入門: 並列同時進行の表現と解析, シュプリンガー・フェアラーク東京(株) (1988).
8) Lathrop, R. H. and Kirk, R. S.: An Extensible Object-Oriented Mixed-Model Functional Simulation System, *22nd IEEE Design Automation Conference*, pp. 630-636 (1985).
9) 杉本 明, 阿部 茂: オブジェクト指向言語 VEGAMES による構造レベル・ハードウェアのモデル化, コンピュータソフトウェア, Vol. 3, No. 3, pp. 71-85 (1986).
10) Meulen, P. S.: INSIST: Interactive Simulation in SmallTalk, *OOPSLA '87 Proc.*, pp. 366-376 (1987).
11) Girczyc, E. F. and Ly, T.: STEM: An IC Design Environment Based on the Smalltalk Model-View-Controller Construct, *24th IEEE Design Automation Conference*, pp. 757-763 (1987).
12) Nichols, K. M. and Edmark, J. T.: Modeling Multicomputer Systems with PARET, *IEEE Computer*, pp. 39-48 (1988).
13) Melamed, B. and Morris, R. J.: Visual Simulation: The Performance Analysis Workstation, *IEEE Computer*, pp. 87-94 (1985).

14) Walker, M.G. and McGregor, J.: Computer-Aided Engineering for Analog Circuit Design, *IEEE Computer*, pp. 100-108 (1986).
15) 梅村恭司(著), 竹内郁雄 (監修): Smalltalk-80 入門, ソフトウェアライブラリ=4, サイエンス社 (1986).
16) Shu, N.C.: *Visual Programming*, Van Nostrand Reinhold (1988).

(平成2年10月11日受付)
(平成3年 3 月 4 日採録)



岡田 義広 (正会員)

昭和39年生. 昭和63年北海道大学工学部電気工学科卒業. 平成2年同大学院修士課程修了. 現在北海道大学大学院工学研究科博士後期課程電気工学専攻在学中. コンピュータアーキテクチャ, データベースマシンの研究に従事.

田中 譲 (正会員)

昭和25年生. 昭和47年京都大学電気工学科卒業. 昭和49年京都大学電子工学専攻修士課程修了. 工学博士. 現在, 北海道大学電気工学科教授. データベースマシン, データベース理論, メディア・ベース, 論理型プログラミング等の研究に従事. 主たる著書,「コンピュータ・アーキテクチャ」(オーム社, 共著). IEEE, ソフトウェア科学会, 人工知能学会各会員.