6M−4 ハードウェアシミュレーションシステムALHARD ビジュアルインタフェースの拡張

小島泰三、杉本 明、阿部 茂
三菱電機株式会社 中央研究所

1 はじめに

計算機ハードウェアの開発には、性能評価や検証のため様々な角度からのシミュレーションが必要である。筆者らは設計者自身によるレジスタトランスファレベルのシミュレータ作成支援を目的として、ハードウェア記述言語 Alhard の設計及びシミュレーションシステムの開発を行った[1]。Alhard システムは、ハードウェアシミュレータ作成のための問題向き言語として、C言語を拡張した言語部分と実行支援環境から構成される。C言語の拡張では、様々な動作モデルを構成するための基本機能や、性能評価や検証のための枠組みの提供に重点を置き、ビット幅付き整数型、オブジェクト指向機能、同期動作記述のためのデーモン機能の導入を行っている。

Alhard システムでは、グラフィック画面上にハードウェアブロック間のデータの流れを示すことにより、直感的にシミュレーション実行状態を把握できるビジュアルインタフェースが提供されている。今回、ビジュアルインタフェースの構築をより容易にし、またより柔軟なユーザインタフェースを提供する目的で画面編集機能の付加を行なった。

以下では、まず本システムにおけるビジュアルインタフェースと従来のシステムの問題点を示す。次にこの問題を解決するために新たに付加した画面編集機能とその実現方法について述べる。

2 ビジュアルインタフェースの画面編集

図1にビジュアルインタフェース画面例を示す。Alhard システムでは、ハードウェア記述とは別に画面上のアイコンの配置を定義した画面記述を用意することにより、図のようなビジュアルインタフェースを生成することができる。

ビジュアルインタフェースでは、ユーザは画面上のアイコンを操作することにより、シミュレーションを行なう。シミュレーション進行に従い、ハードウェアの内部状態が変わると対応する画面上のアイコンが更新される。このため、ハードウェアの状態を直感的に把握することが可能となり、実際のマイクロプログラムのデバッグに適用した結果、約3倍の生産性の向上が確認された。

本システムでは、ハードウェア記述とシミュレーションのためのユーザインタフェースの実現が完全に独立して行なえることを特長の1つとしている。図2に従来のシステムにおける画面記述例を示す。従来の Alhard システムでは、C言語を用いて画面の記述を行なっていた。画面記述はハードウェア記述とは完全に独立しており、後に述べるデーモンを使用して実行時に両者の結合を行なった。

ところが、ハードウェア記述とは異なる言語を用いるため、設計者は2種類の記述言語を扱う必要がある。また、記述による定義

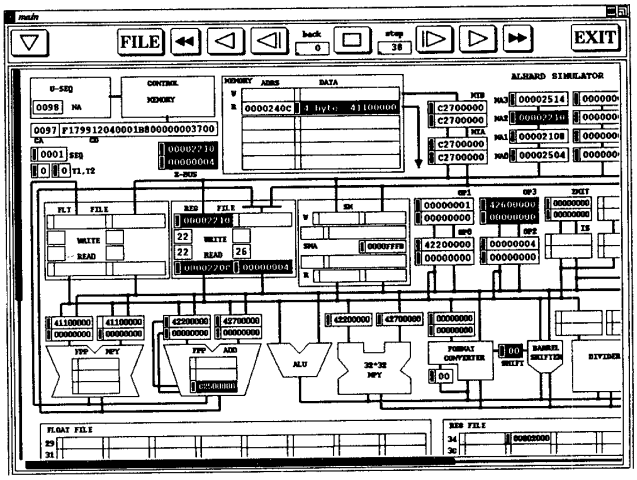

図1: ビジュアルインタフェースの画面例

```
1: areg=data_box(self,8,MARK,std,DEV(AREG));
2: breg=data_box(self,8,MARK,std,DEV(BREG));
3: align(breg,TopLeft,pt_down(BottomLeft(areg),30));
4: v_data_line(self,BottomCenter(areg),Top(breg),
5:              width,ARROW,DEV(AREG),GET);
```

図2: C言語による画面記述例

を用いると、各アイコン間の詳細な関係を定義できるという利点があるものの、ハードウェア設計者自らが複雑な画面イメージを記述することは困難であるという問題が生じた。さらに、C言語を用いて画面記述をコンパイルする方式を採用していたため、画面構成を変更するには、シミュレーションを終了し、シミュレータ自体を再構成する必要があった。このため次に述べる画面編集機能をビジュアルインタフェースに付加した。

図3に、画面編集時の例を示す。画面編集機能はシミュレーション時のモードの1つとして実現した。このためシミュレーション時に必要に応じてアイコンを付加し、対象とするハードウェアリソースを指定することにより、ハードウェアの内部状態を目的に応じてモニタすることが可能となっている。

編集機能としては、通常のドローイングツールのようにカット、ペースト、ドラッギング、グループ化、整列、配置などの他に制約を用いたドラッギングを実現した。また、対応するハードウェアリソースや文字フォントなどのアイコン属性はプロパティシートを用いて操作する。なお、画面定義はファイルにテキスト形式で保存するので、従来のように通常のテキストエディタを用いる画面記述を行なうこともでき、また、アイコン定義をライブラリ化することにより、複雑な画面を容易に作成することが可能である。

A Visual Interface for Hardware Simulation
Taizo KOJIMA, Akira SUGIMOTO, Shigeru ABE
MITSUBISHI ELECTRIC CORPORATION

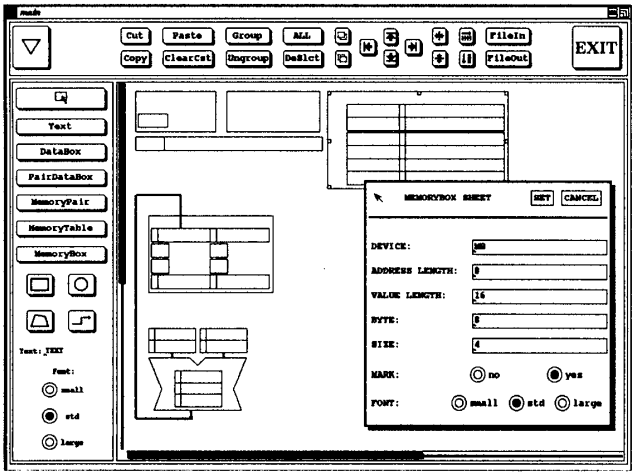

図 3: アイコン編集画面例

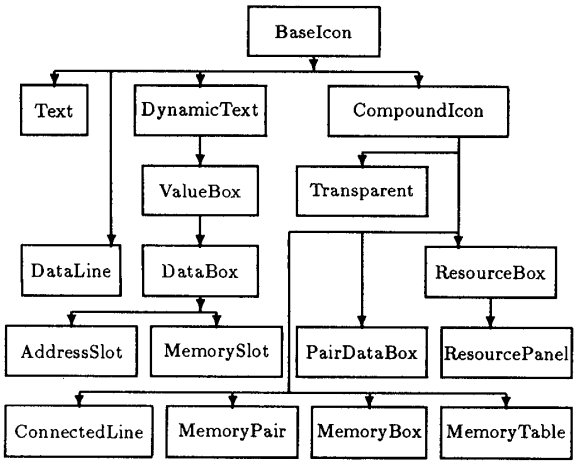

図 4: アイコンの継承関係

3 実現方式

Alhard システムではオブジェクト指向とデーモン機能を使用してビジュアルインタフェースの実現を行なっている。以下では本システムにおける視覚的オブジェクトの構成と、デーモン機能の働きについて説明する。

3.1 クラス継承階層

画面上に表示されるアイコンはオブジェクトとして実現されている。図 4 に本システムで用いたクラスの継承関係を示す。

BaseIcon は全てのアイコンクラスの基底となるクラスであり、矩形領域を持つ。この矩形領域内にアイコンの描画が行なわれる。CompoundIcon は他のアイコンの親となれるアイコンであり、子アイコンは親アイコンの矩形領域によりクリッピングされる。また Transparent は CompoundIcon のサブクラスであり、アイコンのグループ化に使用する。

なお対話型システムの実装の容易さから、メッセージ通信により実行時にメソッドを決定する方式を採用した。

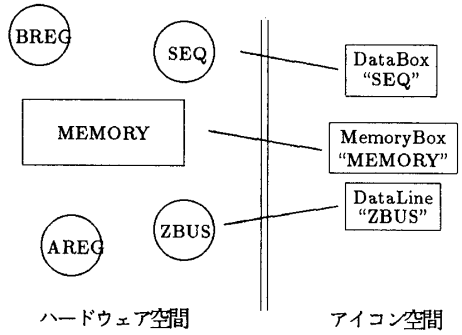

図 5: デーモンによるオブジェクトの対応

3.2 ハードウェアリソースとの対応

ハードウェア記述ではオブジェクト指向を採用し、各ハードウェアリソースはオブジェクトとして実現される。また上記のようにアイコンもオブジェクトとして実現される。しかしながら、ハードウェア記述ではコンパイル時に静的にメソッドを決定する方式を採用しており、アイコンオブジェクトとは異なった構造となっている。これら異なる実装によるオブジェクト間を関係付けるためにデーモン機能を利用した。デーモン機能は、イベントに対して手続きを登録し、これがイベント発生により自動的に起動される機構である。シミュレーション進行により、レジスタなどのハードウェアリソースの参照更新が発生するとデーモンが起動し、対応するアイコンオブジェクトに通知が行なわれ画面更新が行なわれる。

図 5にハードウェアオブジェクトとアイコンオブジェクトの関係を示す。図において左側はハードウェアオブジェクト、右側はアイコンオブジェクトの空間である。ハードウェアオブジェクトとアイコンオブジェクトの対応は、ハードウェアリソース名を用いて実行時にデーモンの設定解除を行なうことによる。

本システムでは、このようにデーモン機能を用いてハードウェアオブジェクトとアイコンオブジェクトとの対応付けを行なっているため、ハードウェア記述とは完全に分離した形でユーザインタフェースを実装し、またシミュレーション時に編集可能なビジュアルインタフェースを提供することが可能となった。

4 おわりに

本稿では Alhard シミュレーションシステムにおけるビジュアルインタフェースの拡張について述べた。ビジュアルインタフェースのソフトウェア量は、C 言語による記述が約 10000 行であり、このうち画面編集機能に関する部分は約 6000 行であった。またこの他、X-Window 上での実現のために Toolkit を作成した。

必要に応じて画面上にアイコンを作成し、並列に動作する動きを視覚的に表示するという方式は、一般的なプログラムにおけるユーザインタフェース構築やデバッギング方法としても有効である。現在、アイコンインタフェース記述用の問題向き言語の設計を行なっており、これを用いて、ビジュアルインタフェース機能を一般化し、ハードウェアシミュレータ以外の問題にも適用してゆきたいと考えている。

References

[1] 杉本, 小島, 阿部, 鶴, 加藤：ハードウェア動作記述言語 ALHARD(1 ～ 3),情処 37 全大(1988).