コンピュータと教育 67－2
（２００２．１２．１３）

教育目的に応じて観察の抽象度が変更可能な計算機シミュレータECASの計算機構築演習での活用

○矢原潤一* 西田知博** 増澤利光*** 松浦敏雄****

概要

計算機の仕組みを理解するための教育用計算機シミュレータECASを開発した．ECASは任意の抽象度の部品定義が可能であり，その部品を画面上で配置，結線しながら，自由に計算機システムを構築でき，その動作を観察できる．これらの機能をうまく組み合わせることで，計算機アーキテクチャの入門教育から，専門教育まで幅広く利用できる．本研究では，学習者自ら計算機を構築するという，情報専門教育向けのコースウェアを用意し，実際に２人の情報科学科の学生に対して，計算機構築実験を行った結果を報告する．今回の実験では，計算機構築演習でECASを用いることで学生が構築した計算機の動作を視覚的に確認することができ，ECASが計算機構築演習を支援するツールとして利用可能だという見極めができた．

ECAS : Educational Computer Simulator with a Multi-level Observation Mechanism
— in case of constructing computers in laboratory course —

Junichi YAHARA* Tomohiro NISHIDA**
Toshimitsu MASUZAWA*** Toshio MATSUURA****

Abstract

We have developed an educational computer simulator ECAS to understand the architecture of computer systems. Users can define any parts of the computer in arbitrary level of abstraction. They can construct a computer system by placing parts and connecting them. Moreover they can observe how these parts work. Not only experts but beginners can learn about computer architecture by using above functions. In this research, we have prepared an experimental course for students in computer science, in which students construct computer systems by themselves. We describe the the results of an experiment involving two computer science students and discuss about the utility of the system based on these experiments.

1 はじめに

計算機の仕組みや動作を理解するにあたり，可視化した計算機シミュレータを利用することで，理解が容易になることが知られている[1][2]．しかし，どのようなアーキテクチャを，どの程度詳細に見せるかは，一般に教育目標や学習者の理解度によって異なる．初学者に対しては，計算機の中身を抽象度の高いレベルの部品によって表し，全体的な動作を把握させることが重要となる．しかし，さらに踏み込んだ内容を教育する場合は，詳細なレベルの構造が観察できる必要がある．

そこで，さまざまな教育場面で対応できるように，任意の抽象度で動作を観察でき，かつ，シミュレーション対象となる計算機のアーキテクチャ

*大阪大学大学院基礎工学研究科 情報数理系
*大阪学院大学 情報学部
**大阪大学大学院 情報科学研究科
***大阪市立大学 学術情報総合センター
*Graduate School of Engineering Science,Osaka University,
**Faculty of Informatics, Osaka Gakuin University,
***Graduate School of Information Science and Technology,Osaka University
****Media Center, Osaka City University

を任意に定義できる教育用計算機シミュレータECASを作成した[3][4]. ECASでは,学習で必要とする順序回路を部品として定義できる機能をもっている. また,それらをまとめてより抽象度の高い部品を作成することもできる. そして,観察時にはシミュレーションの途中でも部品の抽象度を変更することができ,必要に応じて部品の細部を隠したり見せたりすることができる.

本研究では,情報系学科の専門教育レベルでの計算機構築演習においてECASの有用性を確認することを目的として,計算機構築実験を行った. 実験は大阪大学基礎工学部情報科学科4年生2名を対象に行った. 本稿ではその実験内容と結果について述べていく.

以下,2節では,ECAS上での計算機構築の方法について述べ,3節では,実施した実験の概要と結果について詳述する.

2 ECAS上での計算機の構築

2.1 部品オブジェクト

ECASでは,オブジェクト指向の動作モデルを用いて,計算機を構成する各部品をオブジェクトとして定義する. 各部品はいくつかの入力ポート,出力ポートおよび内部状態を持つ. 各部品は入力メッセージ(時刻付き)を受け付けると,入力と内部状態にしたがって,メッセージを出力する. このとき,部品内での遅延時間を考え,入力メッセージの時刻に定めた時間を加えたものを出力メッセージの時刻として付与する(図1).

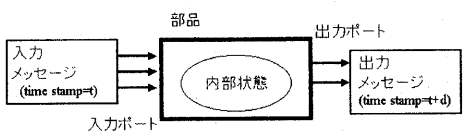

図 1: 部品オブジェクト

また,部品と同様に,部品間をつなぐ信号線もオブジェクトとして定義する. このオブジェクトは信号線とつながっている部品の情報を持ちある部品からメッセージを受け付けると,そのメッセージを信号線がつながっている他の全ての部品へ伝達する.

2.2 部品の定義

シミュレーションエンジンの仕組みに応じた部品を作成するためには,以下の情報が必要となる.

1. 部品の入出力ポート情報
   ポートは識別番号,名前,ポートの位置を情報として持つ.
2. 部品の内部状態の情報
   入力ポートから入ってくるメッセージ以外に,動作に必要な情報を部品内部で保持する.
3. 部品の動作に関する情報
   2.1節で述べたことを踏まえて,各部品は入力ポートからメッセージを受け付けると,入ってきたメッセージの値と現在の内部状態の値に応じて出力するメッセージの値と次の内部状態の値を決定し,遅延時間をメッセージに付与して,メッセージを出力する. 部品ではこれら一連の動作に関わる情報を持つ.

部品の可視化情報としては,部品の画面上でのイメージ,部品パレットに置くボタンのイメージ,表示サポートコンポーネントがある. 表示サポートコンポーネントとは,シミュレーション時に計算機の動作を理解しやすくするためのものであり,部品の内部状態を数値で表示するためのラベルや,部品間の信号のやりとりを強調するための矢印などを用意した.

2.3 部品の作成

部品は必要な情報を記述して作成する. 以下は4節の構築演習で使用しているレジスタの記述例である(図2).

```
⟨ body ⟩
/* 部品名 */
```

```
〈 name 〉 Register16 〈 /name 〉
/* 部品のイメージファイルの場所 */
〈 image 〉 /ImageFiles/ExStep1/register16forstep1.jpg
〈 /image 〉
/* 部品を表すボタンのイメージファイルの場所 */
〈 buttonimage 〉
/ImageFiles/ExStep1/register16forstep1button.jpg
〈 /buttonimage 〉

/* ポート情報 */
〈 ports 〉
〈 port 〉 0 input in 16 174 10 〈 /port 〉
〈 port 〉 1 load in 1 5 10 〈 /port 〉
〈 port 〉 2 send1 in 1 5 40 〈 /port 〉
〈 port 〉 3 send2 in 1 5 70 〈 /port 〉
〈 port 〉 4 output1 out 16 174 40 〈 /port 〉
〈 port 〉 5 output2 out 16 174 70 〈 /port 〉
〈 /ports 〉

/* 状態情報 */
〈 states 〉
〈 state 〉 0 value 16 0 〈 /state 〉
〈 /states 〉

/* 動作定義 */
〈 actions 〉
〈 action 〉
〈 curr 〉
〈 inport 〉 1 1 〈 /inport 〉
〈 /curr 〉
〈 next 〉
〈 nextstate 〉 0 (port[0]) 〈 /nextstate 〉
〈 /next 〉
〈 delay 〉 1 〈 /delay 〉
〈 /action 〉

〈 action 〉
〈 curr 〉
〈 inport 〉 2 1 〈 /inport 〉 /* send1 */
〈 /curr 〉
〈 next 〉
〈 outport 〉 4 (state[0]) 〈 /outport 〉 /* output1 */
〈 /next 〉
〈 delay 〉 1 〈 /delay 〉
〈 /action 〉

〈 action 〉
〈 curr 〉
〈 inport 〉 3 1 〈 /inport 〉 /* send2 */
〈 /curr 〉
〈 next 〉
〈 outport 〉 5 (state[0]) 〈 /outport 〉 /* output2 */
〈 /next 〉
〈 delay 〉 1 〈 /delay 〉
〈 /action 〉
〈 /actions 〉

/* 表示サポートコンポーネント */
〈 supportcomponents 〉
〈 numberlabel 〉 0 27 33 125 38 white black 25 4 16 this
state 0
〈 /numberlabel 〉
〈 /supportcomponents 〉
〈 /body 〉
```

図 2: レジスタの部品記述

部品の定義は, 部品名, 画面イメージ, 部品パレットに置くボタンのイメージ, ポート情報, 状態情報, 動作定義, 表示サポートコンポーネントに分かれる.

部品のポート情報は port タグを用いて記述する. この例では 6 つのポートを定義している. 例えば, 〈 port 〉 0 input in 16 174 10 〈 /port 〉 という記述は, 番号 0, 名前は input, 入力ポート, 16bit の信号を扱う, ポートの位置は部品画像の左上端を起点として, x 方向に 174,y 方向に 10, という情報を示している.

内部状態の情報は state タグを用いて記述する. この例では 1 つの内部状態を持っている. 〈 state 〉 0 value 16 0 〈 /state 〉 この記述は, 状態番号 0, 名前は value,16bit の値を扱う, 初期値は 0 という情報を示している.

部品の動作は action タグを用いて記述する. ECAS では, 各ポートへの入力とそのときの内部状態より, 出力するメッセージと次の内部状態, 遅延時間を定めることで 1 つの動作を定義する. action タグ 1 つにつき, この 1 つの動作を記述できる.

この部品例では 3 つの動作を定義している. はじめの 1 つを示す.

```
〈 action 〉
〈 curr 〉
〈 inport 〉 1 1 〈 /inport 〉
〈 /curr 〉
〈 next 〉
〈 nextstate 〉 0 (port[0]) 〈 /nextstate 〉
〈 /next 〉
〈 delay 〉 1 〈 /delay 〉
〈 /action 〉
```

この記述では,

> 「『1 番の入力ポートにメッセージが到着し, その値が 1』 であるなら『0 番の状態 (状態名 value) に 0 番ポートの値を格納し, 遅延時間を加えてタイムスタンプを更新』」

という動作を定義している.

動作定義の記述では, 前の行にある動作から優先的に動作条件の判別を行う. curr タグ内にある動作条件と 1 つでもマッチしない場合は,

next タグ内にある出力や状態の更新を行うことなく, 次の action タグを評価する. そして, どの動作定義の動作条件ともマッチしなければ, 次のメッセージが届くまでは何もしない.

表示サポートコンポーネントは supportcomponent タグを用いて記述する. 部品の内部状態の値の表示は,numberlabel タグで記述する. レジスタでは内部状態番号 0(状態名 value) の値を画面上に表示するよう定義している (図 3). その他の観察機能も,supportcomponent タグの中で定義できる.

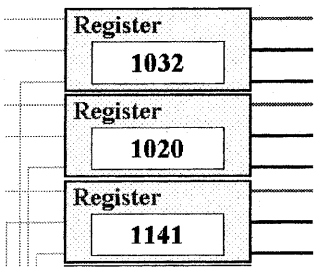

図 3: 内部状態の表示

2.4 計算機の構築

シミュレーション対象の計算機は, 画面上に部品を配置し, それら部品間を線で結ぶことによって構築する. ECAS では, 図 4 で示すような部品パレットを用意しており, そこから部品を選択することによって画面上に配置する. 部品は, 先ほど述べた方法でユーザが作成してパレットに登録してもよいし, システムが標準で提供しているものを利用して構築してもよい.

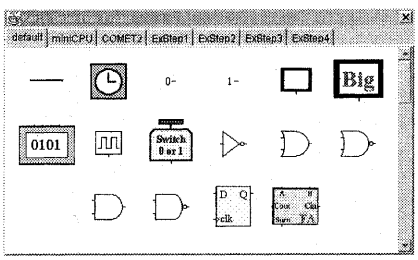

図 4: 部品パレット

ECAS では, 計算機を構築している途中でも, 部分的なシミュレーションを行い, 動作を確認しながら作業を進めることができる. 学生が計算機の構築演習を行う場合などは, この機能を積極的に利用し, 段階的に演習を進めていくことが可能となる.

3 ECAS を用いた計算機構築実験

ECAS を用いて, 部品の設計から行う計算機構築演習を実施し, ECAS の有用性を検証するための実験を行った.

3.1 実験の概要

大阪大学基礎工学部情報科学科の 4 年生 2 名 (男女 1 名ずつ) に対して, 用意した実験指導書をもとに,ECAS を用いて計算機の構築演習を行った. 演習を行った 2 名は, 学部 3 年次で簡単な CPU を設計した経験がある.

実験の準備として, はじめに実験指導書と ECAS でシミュレーション可能なサンプル計算機 (CPUExStep1,CPUExStep4), 各課題で作成した計算機の動作を確認するためのサンプルプログラムを配布した.

実験で学生が取り組む課題は 5 つあり, 課題 1,4 では, 配布したサンプル計算機を用いて動作を確認する. 課題 2,3,5 では学生がみずから計算機を作成する.

指導書は 17 ページからなる構成で, 主に実験での課題内容を述べている. 指導書では, 例えば課題 1 ではサンプル計算機 CPUExStep1 の仕様やその構成方法について解説している. 学生は指導書を読んで CPUExStep1 のアーキテクチャや部品の定義方法や ECAS の使用方法を学習でき, 課題 2 以降で各自が計算機を作成するための準備を行えるようになっている. なお指導書の付録として, サンプル計算機 CPUExStep1 を構成している部品 10 種類のうち,4 種類の部品定義を掲載した.

以下,5 つの課題について内容を述べる.

課題 1

サンプル計算機CPUExStep1(図5)について，仕様などを参考にしながら各部品のはたらきを理解し，実際にCPUExStep1をECASで動作させ，正しく動くことを確認せよ．

また，計算機を構成するすべての部品について，配布した部品定義より，部品の動作定義をはじめとする部品の記述方法を理解し，各部品の動作を把握せよ．

CPUExStep1 の仕様

1word 16bit

1 命令の長さ 1word

アドレス長 10bit

レジスタ R0,R1,R2,R3 の 4 つ

命令の種類 4 種類 (Load, Store, 加算, 減算)

各命令のコードを以下に示す．

命令	コード
LD reg,adr	0000
ST reg,adr	0100
ADD reg1,reg2	1000
SUB reg1,reg2	1100

1 命令のビット割り当て

15 .. 12	11 .. 10	9 .. 0
命令	レジスタ番号	アドレス

演算命令では以下のような割当となる．

15..12	11..10	9..8	7..0
命令	reg1	reg2	don't care

CPUExStep1 を構成する部品

レジスタ
プログラムカウンタ
インストラクションレジスタ
ALU
メモリアドレスレジスタ
メモリデータレジスタ
ゲートコントローラ
メモリ
デマルチプレクサ
クロック

図 5: CPUExStep1 の仕様

課題 2

CPUExStep1を拡張したCPUExStep2(図6)をECASを用いて作成せよ．

CPUExStep2 の仕様

フラグ

演算結果が 0 のとき値が 1，それ以外のとき値が 0 となる Z フラグを追加．

命令の種類

CPUExStep1 の命令を以下のように拡張する．

命令	意味
ADD reg1,reg2	reg1 ← reg1 + reg2 (Z)
SUB reg1,reg2	reg2 ← reg1 - reg2 (Z)

(Z) は Z フラグの値が変わり得ることを示す．

また以下の命令を追加する．

命令	意味
JMP adr	アドレス adr に飛ぶ
JNZ adr	Z=0 のときアドレス adr に飛ぶ
JPZ adr	Z=1 のときアドレス adr に飛ぶ

図 6: CPUExStep2 の仕様

課題 3

CPUExStep2を拡張したCPUExStep3(図7)を作成せよ．

CPUExStep3 の仕様

命令の種類

以下の命令を追加する．

命令	意味
LAD reg,val	値 val をレジスタ reg に格納
AND reg1,reg2	reg1 ← reg1 AND reg2(Z)
OR reg1,reg2	reg1 ← reg1 OR reg2(Z)
XOR reg1,reg2	reg1 ← reg1 XOR reg2(Z)

図 7: CPUExStep3 の仕様

課題 4

サンプル計算機 CPUExStep4 に (図 8) 対して，課題 1 と同様に，ECAS 上でサンプルプログラムを用いて CPUExStep4 を動作させ，その動きを確認せよ．

CPUExStep4 の仕様

1word 16bit

1 命令 1word または 2word

アドレス長 16bit

レジスタ CPUExStep1 と同じ

命令の種類 CPUExStep1 と同じ

1 命令のビット割り当て

15 .. 8	7 .. 4	3 .. 0
命令	reg1	reg2

メモリアクセスの必要がある命令は 2word 命令となり, 2word 目にはアドレスが格納される.

図 8: CPUExStep4 の仕様

課題 5

CPUExStep4 を拡張し,CPUExStep5(図 9) を作成せよ.

CPUExStep5 の仕様

命令の種類

CPUExStep4 の命令を以下のように拡張する.

命令	意味
ADD reg1,reg2	reg1 ← reg1 + reg2 (Z)
SUB reg1,reg2	reg2 ← reg1 - reg2 (Z)

(Z) は Z フラグの値が変わり得ることを示す.

また以下の命令を追加する.

命令	意味
JMP adr	アドレス adr に飛ぶ
JNZ adr	Z=0 のときアドレス adr に飛ぶ
JPZ adr	Z=1 のときアドレス adr に飛ぶ

図 9: CPUExStep5 の仕様

3.2 実験経過

実験全体で必要とした時間は約 7 時間であった (図 10). 7 時間のうち, 学生がはじめて ECAS で計算機を構築する課題 2 では約 3 時間かかっているが, 課題 3,5 では必要とした時間がかなり短くなっており, 計算機の構築方法の理解を深めていることが伺える. 以下, 各課題ごとに実験結果を述べる.

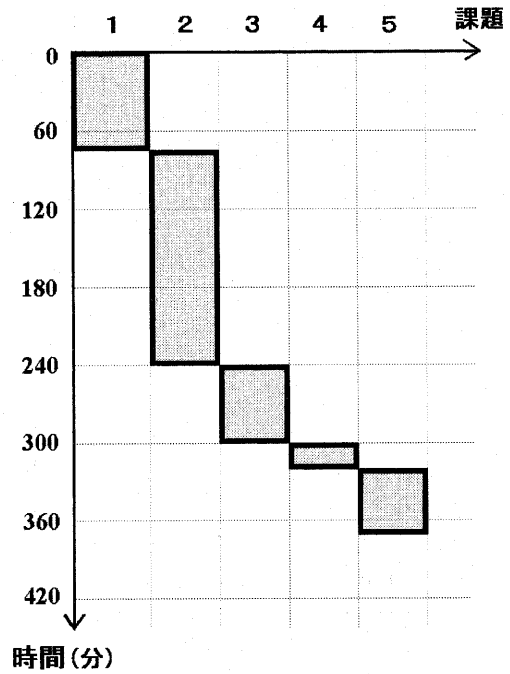

図 10: 実験経過

課題 1

課題に要した時間は 75 分だった. 実施事項を以下に示す.

1. 学生に配布したチュートリアルを参考に ECAS の使用方法を説明.
2. ECAS を用いて, サンプル計算機を構成する各部品の働きと部品間の配線のつながり, 命令セットを説明.
3. 部品定義の記述方法について説明.
4. 学生が ECAS で計算機を動作させ, 各部品が記述通り動作していることを確認.

2. について, 被験者の 1 人はいくつかの部品のはたらきをすぐには理解できなかったようだが, 3. と 4. で部品の動作定義記述をみながら ECAS で計算機を動作させることで理解できたようである.

課題 2

課題に要した時間は 1 人は 2 時間 30 分, もう 1 人は 3 時間であった.

計算機を作成するにあたって,2 人ともはじめ何をすべきか 5 分ほど考えていたようだが, どの部品に改良を加えるべきなのかを決めることができなかった. そこでアドバイスとして, はじめに制御部品の状態遷移図を書き換えることで, どのような制御信号が必要となり, どの部品の動作を改良すべきなのかわかる, ということを学生に教えた.

それからは, 制御部品の状態遷移を記述, 各部品の動作定義を記述, 画面上で回路図を構築, 動作確認という一連の作業を 3,4 回繰り返し,CPUExStep2 を完成させることができた.

CPUExStep2 の動作がうまくいかなかった理由としては,

- ALU の動作修正を忘れていた (Z フラグ値を計算していなかった)
- Z フラグの値の更新を忘れていた, あるいは更新のタイミングを間違った
- 分岐命令実行時に, プログラムカウンタへロード制御信号を出すのを忘れていた

といった点があげられた.

課題 3

課題に要した時間は 2 人とも約 1 時間であった.

課題 2 で計算機の作成にかなり慣れた様子で, 一度計算機を作成した後で修正する時間が, 課題 2 のそれよりも大幅に少なかった.

課題 4

課題に要した時間は 20 分であった.

CPUExStep1 からの主な拡張点は, アドレスの 16bit への拡張とそれにともなう 2word 命令の導入であり, 制御部品の状態遷移図が大きく変わっているが, 2 人とも状態遷移図の説明を聞いて, 計算機の動作がどう変わったか理解できたようである.

課題 5

課題に要した時間は, それぞれ 45 分,55 分であった. 今までの課題への取り組みで作業にはすっかり慣れており, 2 人とも 1 回で正しい動作の計算機を作成することができた (図 11).

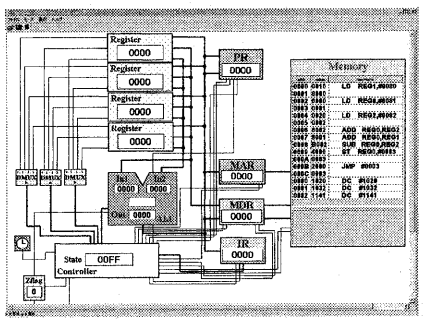

図 11: 学生が構築した CPUExStep5

3.3 事後アンケート

実験後に実験内容の理解のしやすさやシミュレータの使いやすさ, 感想を中心とするアンケートを提出してもらった.

a. 実験内容の理解のしやすさ

ECAS を利用して実験内容が理解しやすかったかどうかについては,

- 1 クロック毎にシミュレーション実行できる機能が便利だった. どの部品に対してどういう信号が出ているかを 1 ステップずつ確認できるので, それぞれの部品がどんな働きをしているのかが理解しやすかった
- 計算機の動作を修正する際に, 動作を目でを見ながら確認できるのがいいと思う
- 学部 3 年次の実験で使用した波形を見てで

動作を確認するシミュレータよりも, ずっとわかりやすいし, 馴染みやすい

などの感想を得た. ただ, 今回の実験対象の学生は学部3年次に1度CPUの設計を行っているため, まったくCPUの設計を行ってない学生に対して同様の実験を行うとどのような結果になるのかが疑問, という声も上がった. ちなみに今回の被験者は, 学部3年次に行った実験で制御部品の状態遷移の理解に約10時間必要としたとのことなので, 制御部品の仕組みの理解を中心として, ECASを用いてどの程度理解を助けられるのか興味深い.

b. シミュレータの使いやすさ

シミュレータの使いやすさについての感想には,

- (回路図を) 書くことはお絵書き感覚で面白かった
- 部品間の配線をする際に部品の配線できる場所を表示してくれるのがよかった

などの肯定的な感想を頂いたが, 同時に,

- 部品を自分で定義する際に, 自動的にポートの位置を決定してほしい
- サンプル計算機の回路図を表示したまま他のウィンドウで計算機を作成したい

のような意見もあり, 計算機の構築機能に関して改良の余地があると思われる.

c. その他感想

その他の感想として,

- ユーザ部品の定義方法について, 〈 action 〉タグの記述の際に, ポートや状態を番号で参照するのは見にくい
- 課題1と課題2は順番が逆の方がよかった

などが挙げられた. これらについても, 計算機の構築機能に関する課題として今後改良すべきだと考えている.

また, 今回実験で利用した部品以外にさまざまな部品を用意していたので, それらの部品も使って何か作ってみたかったという感想もあった.

4 おわりに

本稿では, 教育用計算機シミュレータECASを, 計算機構築実験で試用した結果について報告した.

今回の演習は少人数かつ計算機の構築経験がある学生を対象としての実験であったが, 今後は構築経験のない学生に対してもECASを用いた演習を実施し, その有効性を調べていきたい. また, はじめに論理ゲートを用いた低いレベルの部品から構築し, それを徐々に抽象化して, 最終的に抽象部品を用いて計算機を構築するといった段階的な計算機の構築演習にも取り組み, 教育効果を調べていきたい.

参考文献

[1] 前田他：“計算機構造の教育支援システムとその評価”, 信学技報,ET89-149,pp.43–48(1989).

[2] 吉岡他：“画面上で計算機の構築が可能な計算機アーキテクチャ教育用シミュレータGCS”,CAI学会論文誌,Vol.8,No.2,pp.80–89(1991).

[3] 西田他：“目的に応じて観察の抽象度が変更可能な計算機シミュレータECASを用いた教育とその評価”, 情報処理学会 情報教育シンポジウム SSS2002 論文集,pp.245–252(2002).

[4] 矢原他: ”目的に応じて観察レベルを変更できる教育用計算機シミュレータ”, 情報処理学会 平成13年度前期全国大会 3T-03,pp.4-243-4-244(2001).