システムＬＳＩ設計技術　94－13
(2000. 1. 12)

リセット端子のないＪＫフリップフロップのゲートレベルシミュレーション

松下 浩明

詫間電波工業高等専門学校
〒769-1192 香川県三豊郡詫間町香田551

E-mail: matusita@di.takuma-ct.ac.jp

あらまし　論理回路を学習するときに用いる解説書の多くは JK フリップフロップの動作原理をリセット端子のないゲート回路図で説明している．論理回路 CAD を用いてリセット端子のない JK フリップフロップをゲートレベルで模擬する場合，フリップフロップの内部ゲートの初期値をすべて0あるいは1に設定して模擬するとフリップフロップは疑似発振する．また，内部ゲートの初期値をすべて不定値xに設定して模擬するとフリップフロップは不定値を出力したまま動かない．このような状況を避けるため，従来の論理回路 CAD の多くは内部ゲートの初期値を手動設定できる機能を備えている．本論文ではリセット端子のない JK フリップフロップの内部ゲートの初期値を自動的に設定するアルゴリズムを新たに提案する．このアルゴリズムにより，リセット端子のない JK フリップフロップのゲートレベルの模擬が手動設定を行わずに可能になった．また，このアルゴリズムを教育用論理回路 CAD システムに組み込み，その有効性を確かめた．

キーワード　論理回路，模擬，ＪＫフリップフロップ，リセット端子，初期値設定

Gate Level Simulation of JK Flip-flops without Reset Terminals

Hiroaki Matsushita

Takuma National College of Technology
551 Kouda,Takuma-cho,Kagawa-ken 769-1192,Japan

E-mail: matusita@di.takuma-ct.ac.jp

Abstract　Most of the textbooks to use in learning digital systems are explaining the operation principle of JK flip-flops by the gate level schematics which don't contain reset terminals. In gate level simulation of JK flip-flops without reset terminals, when all initial values of the inner gates of flip-flops are set to 0 or 1, flip-flops can cause oscillations. When they are set to undefined value x, flip-flops don't work as they output undefined values. To avoid such a situation, as for much of traditional logic circuit CAD's, they are equipped with the function that initial values at inner gates can be manually set. In this paper, we propose a new algorithm which sets initial values at inner gates of JK flip-flops without reset terminals automatically. JK flip-flops without reset terminals became able to be simulated in gate level without doing manual setting by this algorithm. We also incorporated this algorithm into a Educational CAD system and ascertained the effectivity.

Key words　logic circuit, simulation, JK flip-flop, reset terminal, initial values setting

1. はじめに

論理回路の学習において，ＪＫフリップフロップの機能と動作原理を理解することは，それに続く順序回路を理解する上での欠くべからざる学習項目のひとつである．そのため，論理回路に関する解説書はＪＫフリップフロップに関する節を設け，機能と動作原理を詳述している．ただ，多くの解説書はＪＫフリップフロップのゲートレベルの回路図において，リセット端子を付加していない図を載せている[1)~4)]．これは回路図がむやみに煩雑になるのを避けるためである．解説書の大半は文章中で実際のＪＫフリップフロップはリセット端子があることを述べているのではあるが，多くの学習者はＪＫフリップフロップを論理回路ＣＡＤで動作確認しようとするとき，解説書に載っているリセット端子のない回路で確かめようとする傾向にある．

論理回路ＣＡＤを用いてリセット端子のないＪＫフリップフロップをゲートレベルで模擬する場合，フリップフロップの内部ゲートの初期値をすべて '0' あるいは '1' に設定して模擬するとフリップフロップは疑似発振する．また，内部ゲートの初期値をすべて不定値 'x' に設定して模擬するとフリップフロップは不定値を出力したまま動かない．このような状況を避けるため，従来の論理回路ＣＡＤの多くは内部ゲートの初期値を手動設定できる機能を備えている[5)~7)]．

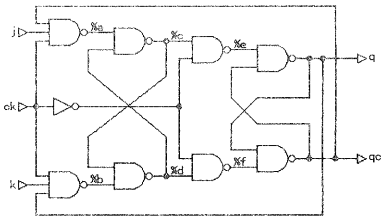
本論文ではリセット端子のないＪＫフリップフロップの内部ゲートの初期値を自動的に設定するアルゴリズムを新たに提案する．このアルゴリズムにより，リセット端子のないＪＫフリップフロップのゲートレベルの模擬が手動設定を行わずに可能になった．また，このアルゴリズムを教育用論理回路ＣＡＤシステムに組み込み，その有効性を確かめた．

本論文の構成は次のとおりである．2 章で，従来の論理回路ＣＡＤでリセット端子のないＪＫフリップフロップを模擬するときの問題点を述べる．3 章で初期値設定問題を定義し，任意の回路に対する初期値設定問題は $\mathcal{NP}$ 完全であることを示す．4 章でリセット端子のないフリップフロップを含むある回路のクラスを定義し，そのクラスの回路に対し，初期値設定問題を解くアルゴリズムを述べる．5 章で，そのアルゴリズムが組み込まれた教育用ＣＡＤについて述べる．

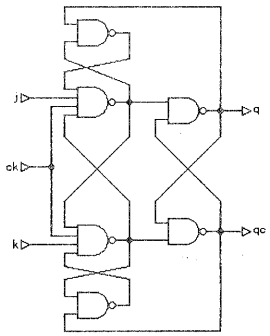
2. ＪＫフロップフロップのゲートレベルシミュレーション

2.1 ＪＫフリップフロップ

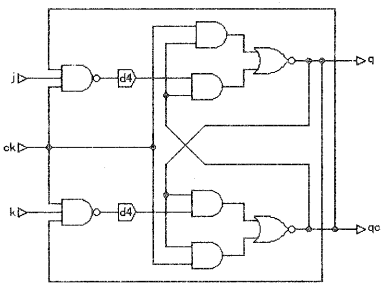
多くの解説書に共通して掲載されているＪＫフリップフロップを**図 1** に示す．タイプ 1 のＪＫフリップフロップはマスタースレーブ型のものである．タイプ 2，タイプ 3 のＪＫフリップフロップはエッジトリガー型のものである．回路上の特徴としてはＤフリップフロップにおいてはフィードバックループが一重であるのに対し，ＪＫフリップフロップではトグル動作を実現するために二重のフィードバックループがある．



(a) type1

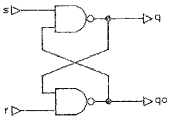


(b) type2

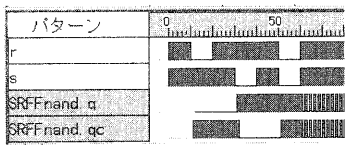


(c) type3

図 1　ＪＫフリップフロップ
Fig. 1　JK flipflop



(a) SR flipflop



(b) Pseudo oscillation

図2 疑似発振
Fig. 2 Pseudo oscillation

2.2 初期値 '0' のゲートレベルシミュレーション

各素子の出力信号線の初期値をすべて '0' または '1' としてJKフリップフロップを模擬すると疑似発振する．疑似発振とは，たすき掛けされた2つの *nand* 素子あるいは *nor* 素子の入力端子に同時に信号変化が起きたとき，あるいはその閉路に外からスパイクが入ったときに発生するものである．この発振現象は実際の回路上では信号の伝播遅延のばらつき等によってほとんど生じないもので，シミュレータ特有のものである．**図2** に *nand* 素子による疑似発振を示す．また，ALDEC社の SUSIE-CAD を用いて，素子の初期値を '0' として，タイプ1のJKフリップフロップを模擬した結果を **図3** （a）に示す．

2.3 初期値 '*x*' のゲートレベルシミュレーション

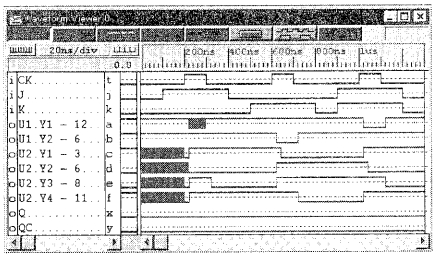
各素子の出力信号の初期値をすべて不定値 '*x*' としてJKフリップフロップを模擬するとフリップフロップは不定値を出力したまま動作しない．SUSIE-CADを用いて，素子の初期値を不定値 '*x*' として，タイプ1のJKフリップフロップを模擬した結果を 図3（b）に示す．

3. 初期値設定問題

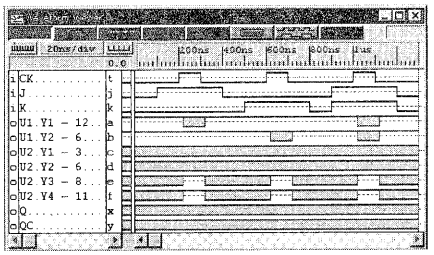
3.1 回路のグラフモデル

回路 c を有向2部グラフ $(E, S, T, type)$ でモデル化する☆．ここで E, S は集合である．E, S の要素をそれぞれ素子，信号線とよぶ．T は $(E \times S) \cup (S \times E)$ の部分集合である．T の要素を端子とよぶ．$type$ は E から $\{not, and, nand, or, nor\}$ への関数である．

☆ グラフ理論の用語の中に回路という用語があるが，本論文では，回路をグラフ理論における用語ではなく，論理回路を表す用語として用いる[13]．



(a) initial values are set to 0



(b) initial values are set to x

図3 SUSIE-CAD による模擬
Fig. 3 Simulation by SUSIE-CAD .

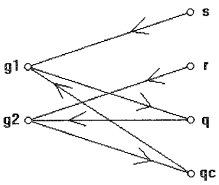


図4 回路のグラフモデル
Fig. 4 Graph model of circuit

$type(e)$ を素子 e の機能とよぶ．

端子 t が $t \in S \times E$ のとき，t を入力端子とよぶ．$t \in E \times S$ のとき，t を出力端子とよぶ．素子は1個の出力端子と接続しており，また，1個以上の入力端子と接続している．信号線は1個の出力端子と接続しており，また，0個以上の入力端子と接続している．

端子 t に素子 e が接続しているとき，t は e の端子であるという．素子 e の出力端子につながる信号線を e 出力信号線とよび，$out(e)$ で表す．入力端子につながる信号線を e の入力信号線とよぶ．

図2 (a) の回路のグラフモデルを**図4** に示す．

3.2 初期値設定問題

val は S から { '0', '1', '*x*'} への関数とする．$val[s]$

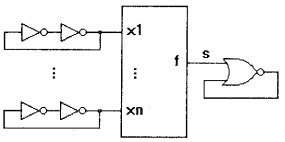


図5　充足判定問題に対する回路
Fig. 5　Circuit for satisfiability problem

を信号線 s の信号値とよび，val を信号値関数とよぶ．

e を素子とし，e の入力信号線を $s_1,\ldots,s_k$ とする．また，e の機能は f であるとする．そのとき，

$$val[out(e)] = f(val[s_1],\ldots,val[s_k])$$

であるならば，信号値関数 val は素子 e に対し，正常であるという．また，そのとき，素子 e は正常であるという．$val[out(e)]$ が 'x' でないならば，素子 e は確定しているという．

$s_1,\ldots,s_m$ を回路 c の一次入力 (*primary input*) の信号線とする．回路 c と信号値 $v_1,\ldots,v_m$ が与えられたとき，以下の条件を満たす信号値関数 val の1つを求める問題を回路 c の初期値設定問題とよぶ．

（a）c のすべての素子は確定しており，かつ正常である．

（b）$val[s_i] = v_i (i = 1,\ldots,m)$.

定理1　初期値設定問題は $\mathcal{NP}$ 完全である．

証明　$\mathcal{NP}$ 完全の定義は文献 8) による．論理関数の充足判定問題は $\mathcal{NP}$ 完全である．ここでは，論理関数の充足判定問題を初期値設定問題に帰着させることにより，初期値設定問題が $\mathcal{NP}$ 完全であることを証明する．

論理関数 $f(x_1,\ldots,x_n)$ を回路で表現する．さらに，**図5** に示すように，論理関数 f の入力 $x_1,\ldots,x_n$ に2個のインバータの閉路を付加し，出力に1個の2入力 *nor* 素子を付加する．

f の出力につながる信号線を s とすると val が *nor* 素子に対し，正常であるためには $val[s]$ は1でなければならない．したがって，val が回路中のすべての素子に対し，正常であるならば，入力 $val[x_1],\ldots,val[x_n]$ に対し，論理関数 f は充足する．そうでないならば，入力 $val[x_1],\ldots,val[x_n]$ に対し，論理関数 f は充足しない．　□

4. 初期値設定アルゴリズム

前章では，任意の回路に対する初期値設定問題は $\mathcal{NP}$ 完全であることを示した．リセット端子のないＪＫフリップフロップを含むある限定された回路のクラスに対して，初期値設定問題を解くアルゴリズムを示す．

4.1 単流動的な素子

信号値関数 val は素子 e に対し，正常であるとする．$val[out(e)]$ が 'x' であるとき，e は流動的であるという．

e は流動的な素子であるとする．e の入力信号線のうち，その値が 'x' である入力信号線が1つしかないとき，e を単流動的であるといい，その入力信号線を $float(e)$ で表す．

素子の機能が *not*, *nand* または *nor* のとき，その素子を反転素子とよぶ．素子の機能が *and* または *or* のとき，その素子を非反転素子とよぶ．

補題1　素子 e は単流動的であるとする．

（1）素子 e が反転素子であって，それが正常に確定されたならば，

$$val[out(e)] = \overline{val[float(e)]}$$

である．

（2）素子 e が非反転素子であって，それが正常に確定されたならば，

$$val[out(e)] = val[float(e)]$$

である．

証明　（1）素子 e は単流動的であるとする．e の機能が *not* であるときは明らかに成立する．

e の機能が *nand* であるとき，$val[out(e)]$ が 'x' であることより，$float(e)$ 以外の e の入力信号線の信号値は '1' である．よって，確定化後，

$$val[out(e)] = \overline{val[float(e)]}$$

となる．

e の機能が *nor* のときも同様な議論で成立する．

（2）素子 e は単流動的であるとする．e の機能が *and* であるとき，$val[out(e)]$ が 'x' であることより，$float(e)$ 以外の e の入力信号線の信号値は '1' である．よって，確定化後，

$$val[out(e)] = val[float(e)]$$

となる．

e の機能が *or* のときも同様な議論で成立する．　□

4.2 偶反転閉路

偶数個の反転素子と0個以上の非反転素子からなる閉路を偶反転閉路とよぶ．

補題2　偶反転閉路上の反転素子を辺の向きの順に $e_1,\ldots,e_{2n}$ とし，反転素子 e_i と e_{i+1} 間の非反転素子を $e_i^1,\ldots,e_i^{m_i}$ とする．そのとき，$v = val[out(e_1)]$ とすれば，以下が成立する．

（1） $val[out(e_i)] = \begin{cases} v, & i \text{ が奇数のとき} \\ \overline{v}, & i \text{ が偶数のとき} \end{cases}$

$(i = 1, \ldots, 2n)$

（2） $val[out(e_i^j)] = val[out(e_i)]$

$(i = 1, \ldots, 2n;\ j = 1, \ldots, m_i)$

証明　補題 4 を繰り返し適用することにより，結果を得ることができる．□

4.3 確定化操作

$s_1, \ldots, s_m$ を回路 c の信号線とする．回路 c と信号値 $v_1, \ldots, v_m$ が与えられたとき，信号線 $s_i (i = 1, \ldots, m)$ の信号値を v_i に変更する．次に，信号値が変更した信号線を入力信号線に持つすべての素子を模擬する．その結果，素子の出力信号線の信号値が変更される可能性があるが，その変更が行われなくなるまで模擬を繰り返す．そのとき，信号値が '0' または '1' に設定された信号線を回路 c から削除し，さらに信号線を削除した結果どの信号線ともつながらなくなった素子も削除する．このようにして回路 c の部分グラフをつくる操作を信号値 $v_1, \ldots, v_m$ に対する確定化操作とよぶ．

図 6 に確定化操作のアルゴリズムを示す．

4.4 ソース成分

確定化操作によって構成された回路 c の部分グラフにおいて，強成分（極大な強連結成分）の中でその強成分外からその強成分に入る信号線がないとき，その強成分をソース成分とよぶ．回路 c のすべてのソース成分が偶反転閉路であるとき，回路 c を偶反転回路とよぶ．

図 7 に偶反転回路のソース成分を構成する素子の 1 つを見つけるアルゴリズムを示す．

補題 3　アルゴリズム *Find* は偶反転回路に対し，ソース成分を構成する素子の 1 つを見つける．

証明　回路は偶反転回路と仮定する．ソース成分にはソース成分外から入る信号線がないから，ソース成分に属する素子はすべて単流動素子である．アルゴリズム *Find* では単流動素子を逆トレースすることにより，単流動素子が閉路上にあるか否か調べている．したがって，アルゴリズム *Find* は偶反転回路に対し，ソース成分を構成する素子の 1 つを見つける．□

4.5 初期値設定アルゴリズム

図 8 に，偶反転回路に対し，初期値設定を行うアルゴリズムを示す．

アルゴリズム A の入力は回路 c と回路 c の一次入力の信号線 $s_1, \ldots, s_m$ およびそれらの信号線の信号値 $v_1, \ldots, v_m$ である．このアルゴリズムはまず，確定化操作を行う．次に回路のソース成分を構成する素子の 1 つを見つけ，その流動入力信号線の信号値を '0' にする．そして再び確定化操作を行う．確定化操作とソース成分の検出を繰り返すことにより，回路の各信号線の初期値を設定していく．

```
global var
  c:   LogicCircuit;
  val: SignalLinesSet → {'0','1','x'};
  sim: ElementsSet → {'yes','no'};
procedure Fix(
  入力 s_1, ..., s_m: SignalLine;
  入力 v_1, ..., v_m: SignalValue);
begin
  E := φ;
  for i := 1 to m do begin
    val[s_i] := v_i;
    信号線 s_i を入力信号線とする素子でかつ
    sim フラグが 'yes' である素子の集合を
    E_i とする;
    E := E ∪ E_i
  end;
  while E ≠ φ do begin
    E の要素のひとつ e を幅優先探索に
    よって取り出す;
    OneSim(e, E)
  end
end

procedure OneSim(
  入力   e: LogicElement;
  入出力 E: LogicElementSet);
begin
  e の機能を f とし，e の入力信号線の値を
  v_1, ..., v_k とする;
  v := f(v_1, ..., v_k);
  if v ≠ val[out(e)] then begin
    val[out(e)] := v;
    out(e) を入力信号線とする素子でかつ
    sim フラグが 'yes' である素子の集合を
    E_e とする;
    E := E ∪ E_e
  end
end
```

図 6　確定化操作
Fig. 6　Fixing signal values

```
global var
  c:   LogicCircuit;
  val: SignalLinesSet → {'0','1','x'};
function Find(
  出力 e: LogicElement) : Boolean;
var
  reach: ElementsSet → {'yes','no'};
begin
  c の単流動素子からなる集合を E_f とする;
  for each e1 ∈ E_f do begin
    for each e2 ∈ E_f do
      reach[e2] := 'no';
    e := e1;
a1: reach[e] := 'yes';
    e := float(e) を出力信号線とする素子;
    if e が単流動素子でない then
      go to a2;
    else if reach[e] = 'yes' then
      return true;
    else
      go to a1;
a2: ;
  end;
  return false
end
```

図7 ソース成分に属する素子の検出
Fig. 7 Finding source element

```
global var
  c:   LogicCircuit;
  val: SignalLinesSet → {'0','1','x'};
  sim: ElementsSet → {'yes','no'};
procedure A(
  入力 s1,...,sm: SignalLine;
  入力 v1,...,vm: SignalValue);
begin
  S を c の全信号線の集合とする;
  for each s ∈ S do
    val[s] := 'x';
  E を c の全素子の集合とする;
  for each e ∈ E do
    sim[e] := 'yes';
  Fix(s1,...,sm, v1,...,vm);   .....(a)
  出力信号線の値が 'x' でないすべての素子の
  sim フラグを 'no' に変更する;
  while Find(e) do begin
    e の流動入力信号線を s とする;
    Fix(s,'0');                .....(b)
    出力信号線の値が 'x' でないすべての素子の
    sim フラグを 'no' に変更する;
  end
end
```

図8 初期値設定アルゴリズム
Fig. 8 Initial values setting algorithm

図9 に，タイプ1のJKフリップフロップにアルゴリズム A を適用したときの処理の概要を示す．

補題4 アルゴリズム Fix により偶反転閉路を確定したとき，各素子は正常である．

証明 e_a をソース成分に属する素子とする．$float(e_a)$ が '0' に確定し，それを起点にして確定化操作が行われたとする．$float(e_a)$ を出力信号線とする素子を e_b とする．アルゴリズム $OneSim$ は出力信号線の値が 'x' である素子に対しては出力信号線を正常に確定していくので，確定化操作後，e_b 以外の素子はすべて正常である．

e_b が正常であるかどうかを調べてみる．確定化操作前に信号値が 'x' である e_b の入力信号線を s_b とする．e_b が反転素子ならば，補題 1 により，確定化操作後，s_b の値は '1' である．$out(e_b)$ の値は '0' であるから，素子 e_b は正常である．

e_b が非反転素子ならば，補題 1 により，確定化操作後，s_b の値は '0' である．$out(e_b)$ の値は '0' であるから，素子 e_b は正常である． □

アルゴリズム A において，ステップ（a）では信号値を $v_1,\ldots,v_m$ に，ステップ（b）では信号値を '0' に確定したが，これをすべての組合せの信号値に対し確定化操作を行い，最終的にすべての素子を確定することができるとき，そのような回路のクラスを $\mathcal{C}$ で表す．

定理2 アルゴリズム A はクラス $\mathcal{C}$ の回路 c に対し，正常な初期値設定を $\mathcal{O}(n^2)$ で行う．ここで，n は回路 c の素子の総数である．

証明 補題 3 と補題 4 を繰り返し適用することにより，アルゴリズム A は偶反転回路に対し，正常な初期値設定を行うことがわかる．

回路 c の素子の総数を n とする．アルゴリズム Fix の実行時間は $\mathcal{O}(n_f)$ である．ここで n_f はアルゴリズム Fix により確定された素子の個数である．また，アルゴリズム $Find$ の実行時間は $\mathcal{O}(n)$ である．よって，アルゴリズム A 全体では $\mathcal{O}(n^2)$ 時間を要する． □

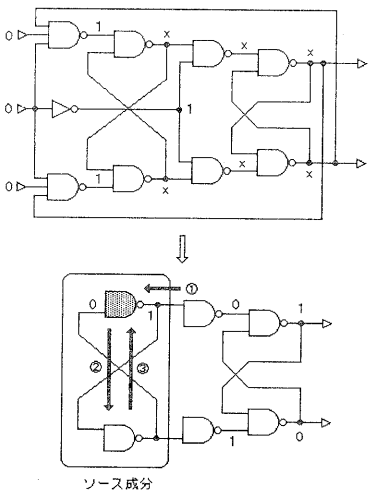


図9 アルゴリズム A の適用例
Fig. 9 An Example which applied algorithm A

タイプ1，2，3のＪＫフリップフロップはクラス C に含まれるので以下の系が成立する．

系1 アルゴリズム A はタイプ1，2，3のＪＫフリップフロップに対し，正常な初期値設定を行う．

5. 教育用 CAD への組み込み

5.1 源内 CAD の概要

源内 CAD は教材用に単純化されたディジタルシステムの設計検証を支援する教育用 CAD システムである．源内 CAD は回路図エディタ，波形エディタ，シンボルエディタ，階層展開プログラム，論理シミュレータなどからなり，以下の特徴を持つ．

（1）階層化設計のサポート

回路を1つのシンボルでブラックボックス化する階層化のみならず，信号線接続シンボルによる信号線の階層化も実現している．

（2）検証レベルの複数化

模擬可能な素子として，素子素子やフリップフロップなどのほか，データ幅が可変なレジスタなどの機能素子を用意している．論理素子の模擬結果は波形表示し，機能素子の模擬結果は文字表示することにより，論理レベルのみならず機能レベルの検証も可能にしている．

源内ＣＡＤは詫間電波高専において，組合せ回路，順序回路，簡単な計算機などの設計演習で使用されている．また，フリーソフトとして，公開している[14]．

5.2 ＪＫフリップフロップの模擬

本論文で提案した初期値設定アルゴリズムを源内ＣＡＤに組み入れた．**図10** はタイプ1のＪＫフリップフロップの模擬結果である．源内ＣＡＤでは，自動的に初期値設定された信号値と通常の信号値を区別するため，自動的に初期値設定された信号値は '0'，'1' のかわりに新たな信号値 'x0'，'x1' を導入している．図10において，濃い色で '1' を，淡い色で 'x1' を表している．

図11 （a）の回路は6進カウンタであり，タイプ1のＪＫフリップフロップを3個用いている．（b）にその模擬結果を示す．

6. おわりに

本論文では，従来の論理回路ＣＡＤでリセット端子のないＪＫフリップフロップを模擬するときの問題点を述べ，その問題を解決するために初期値設定アルゴリズムを新たに提案した．このアルゴリズムにより，解説書に記載されているどのタイプのＪＫフリップフロップも内部ゲートの初期値を手動設定することなしに模擬可能になった．

謝辞 熱心にご議論いただいた詫間電波高専國井洋臣教授，村上純一助教授に謹んで感謝の意を表する．

参考文献

1) 角田秀夫: フリップフロップ回路と計数回路の設計, 東京電機大学出版局 (1975).
2) 中川圭介: 計算機の論理設計, 近代科学社 (1984).
3) 村上国男，石川勉: 論理回路入門，共立出版 (1996).
4) 村田裕: 順序論理回路, 共立出版 (1998).
5) デザインウェーブ企画室編: PC によるディジタル回路の設計とシミュレーション, CQ 出版社 (1996).
6) 松下浩明: ディジタルシステムの設計演習のための教育用 CAD システム, 電子情報通信学会論文誌 A, Vol. J77-A, No. 3, pp. 506–517 (1994).
7) 深山正幸，北川章夫，秋田純一，鈴木正國: ＨＤＬによるＶＬＳＩ設計，共立出版 (1999).
8) Aho, A.V., Hopcroft, J.E. and Ullman, J.D.: *The design and analysis of computer algorithm*, Addison-Wesley Reading Mass. (1974).
9) Breuer, M. A.: *Design Automation of Digital Systems*, Prentice-Hall (1973). (林孝雄訳：ディジタル計算機の自動設計，産業図書 (1973)).
10) Bening, L.: Developments in Computer Simulation of Gate Level Physical Logic, *Proc. 16th DA Conf.*, pp. 561–567 (1979).

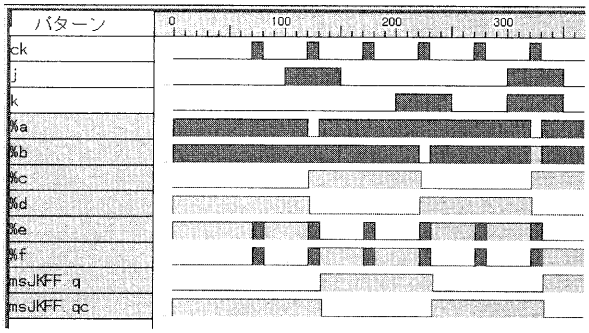
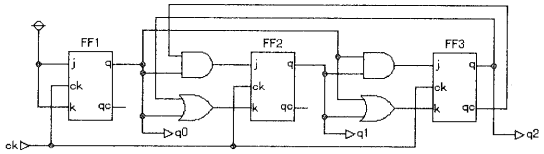
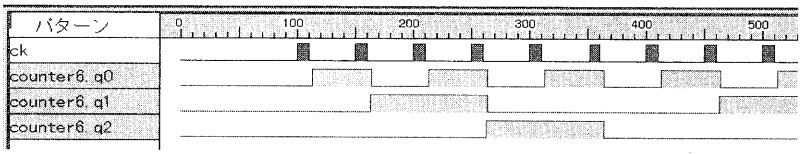


図10 タイプ1のJKフリップフロップの模擬
Fig. 10 Simulation of JK flipflop type 1



(a) counter



(b) Simulation

図11 6進カウンタの模擬
Fig. 11 Simulation of counter

11) 樹下行三編: 論理装置のCAD, 情報処理学会 (1981).
12) Smith, M.J.S.: *Application - Specific Integrated Circuits*, Addison-Wesley (1997).
13) Behzad M., Chartrand G., Lesniak-Foster L.: *Graphs & Digraphs*, Prindle, Weber & Schmidt (1979). (秋山仁, 西関隆夫訳: グラフとダイグラフの理論, 共立出版 (1981)).
14) 松下浩明: 源内CAD, http://www.di.takuma-ct.ac.jp/ matusita/GuenCAD/index.html.