設計自動化 55－9
(1990.12.12)

符号化時間記号シミュレーションに基づく タイミングエラー確率の解析

出口豊　石浦菜岐佐　矢島脩三
京都大学工学部

あらまし　遅延の大きさが不確定で、その最大値と最小値だけが与えられているような論理素子によって構成された非同期回路において、ハザードなどのタイミングエラーが起こる確率を計算する方法を提案する。本手法は以前に提案した「符号化時間記号シミュレーション」の考え方を発展させたものであり、従来再収れんによって不正確にしか求められなかった確率を正確に計算することができる。

Analysis of Timing Error Probability Based on Coded Time-Symbolic Simulation

Yutaka DEGUCHI, Nagisa ISHIURA, Shuzo YAJIMA
Faculty of Engineering, Kyoto University

Abstract　We propose a new technique of calculating timing error probabilities named *probabilistic CTSS*. We are concerned with timing verification of logic circuits consisting of gates whose delay values are uncertain and are specified by their minimum and maximum values. This technique is based on *coded time-symbolic simulation*, which we proposed in our previous work. It calculates accurate timing error probabilities which the conventional method can not calculate because of reconvergences in the circuits.

1 はじめに

近年の集積回路技術の進歩に伴い、より大規模で高機能な論理システムが実現されるようになってきたが、その設計をいかに検証するかということが重要な問題となっている。特にタイミングに関する検証は複雑で困難な場合が多い。大規模な論理回路はタイミング検証の比較的容易な同期式順序回路として設計されることが多い。しかし、通信制御などの周辺論理のように非同期回路として設計される部分も存在している。非同期回路のように、回路の振るまいが微妙なタイミング関係に依存する場合には、製造条件や使用環境の違いに起因する素子遅延のばらつきを考慮する必要がある。論理シミュレーションでは、このような素子遅延の不確定性を最大/最小遅延モデル [1] により解析する。このモデルによるシミュレーションは、高速ではあるが、回路中の再収れんが原因でシミュレーション結果が悲観的になることが知られており、回路中の設計誤りの有無を正確に求めることは難しい。そのため、遅延の不確定性を考慮したタイミング検証に関する研究が数多くなされている。これまでに提案された代表的な検証手法としては、時間記号シミュレーション [3]、素子間の独立性を利用したタイミング検証法 [2]、非決定性オートマトンの比較による検証方法 [4]、符号化時間記号シミュレーション [7] などが挙げられる。

これらの手法は、エラーの有無の判定だけを問題にする検証には有効であるが、実際にはタイミングエラーは歩留まりと深い関連があるので、エラーの存在のみならずエラーの起こる確率を考慮することも重要な問題であると考えられる。遅延のばらつきによるエラー確率の増大は歩留まりの低下となって現れる。一方、遅延のばらつきによるエラーの可能性があっても、その確率が十分小さく歩留まりに与える影響が小さければ、これを無視することができる。本論文では、論理回路におけるタイミングエラー確率の解析について考察する。タイミングエラーの確率を求める一手法としては、信号値確率 (信号値が 1 となる確率) を各時刻、各信号線について近似的に計算する方法が知られている。しかし、この近似計算手法は全ての信号線の信号値確率を独立なものであると仮定して計算を行うため、再収れんが存在する回路では不正確な結果を出力する。最大/最小遅延モデルによるシミュレーションの結果が悲観的であるのに対し、この近似計算では楽観的な (信号値確率を過小評価する) 結果を出力することもありうる。

本稿ではエラー確率を正確に求める手法として、確率的符号化時間記号シミュレーション (probabilistic CTSS) を提案する。この手法では、回路中に再収れんが存在しても正確な結果を得ることができる。本手法は、タイミング検証の一手法である符号化時間記号シミュレーション [7] を基にしている。符号化時間記号シミュレーションでは、全てのゲートの遅延のとりうる全ての値の組合せについてシミュレーションを行うことにより、遅延のばらつきによる回路のふるまいを完全に把握する。そして、類似した計算をまとめて行うために不確定な遅延を持つゲートのとりうる遅延の値の可能性を符号化し、ブール変数を用いて表現する。この遅延の符号化によって、不確定な遅延を持つゲートを含むシミュレーションを通常の記号シミュレーションに帰着させる。確率的符号化時間記号シミュレーションでは、ゲートのとりうる遅延の値の可能性を符号化する際に、遅延の値の分布状況を考慮する。すると、ある信号線のある時刻の信号値確率は、記号シミュレーションの結果得られた論理関数の真理値表密度として求めることができるようになる。この方法では、信号線の依存関係が論理関数の形で保持されているので、回路に再収れんが存在しても正確な結果を得ることができる。また、論理関数の内部表現として共有二分決定グラフ [5] を用いることにより、多くの論理関数を効率良く記憶するとともに、論理関数の真理値表密度を効率良く計算することができる。

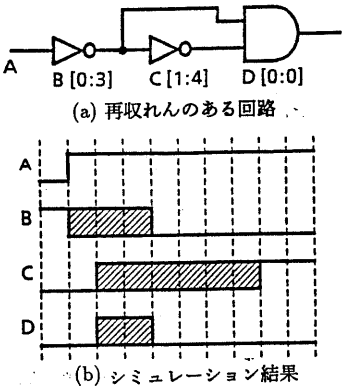


図 1　最大最小遅延モデル

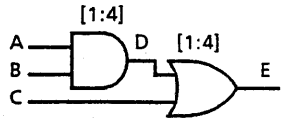
以下、2 章では従来の手法及びその問題点について述べる。3 章では、符号化時間記号シミュレーションの概念について述べる。4 章で確率的符号化時間記号シミュレーションの詳細及びエラー確率の計算方法について述べ、5 章では確率的符号化時間記号シミュレーションの結果の正確さについて議論した後に性能評価の結果について述べる。

2 タイミングエラー確率の近似計算手法

2.1 最大最小遅延モデル

現実の論理素子の遅延の値は製造条件や使用環境の違いによって変化する。従って、非同期回路のように、微妙なタイミングの違いによって回路のふるまいが変化する場合には、素子遅延の不確定性を考慮して検証を行わなければならない。

論理シミュレーションでは、この不確定性を最大/最小遅延モデルを用いて解析する [1]。しかし、このモデルには、回路中に再収れんが存在する場合にはシミュレーション結果が悲観的になる (必要以上の不定値を出力する) という欠点が存在する。図 1(b) は図 1(a) の回路を最大/最小遅延モデルによりシミュレーションした結果である。信号値 D 中の不定値の存在はハザードの出現の可能性を示しているが、実際にはハザードの可能性はない。なぜならば、信号線 B の立ち下がる時刻の方が、信号線 C の立ち上がる時刻より常に早いからである。



t	$P_A[t]$	$P_B[t]$	$P_C[t]$	$P_D[t]$	$P_E[t]$
0	0	1	0	0	0
1	1	1	0	0	0
2	1	1	0	0.125	0
3	1	1	0	0.5	0.016
4	1	1	0	0.875	0.109
5	1	1	0	1	0.344
6	1	1	0	1	0.656
7	1	1	0	1	0.890
8	1	1	0	1	0.984
9	1	1	0	1	1

図2　近似計算手法による信号値確率の計算

2.2　タイミングエラー確率の近似計算手法

タイミングエラーの確率を計算する一つの手法として、信号値確率を計算する方法が知られている。信号値確率は、信号値が1をとる確率と定義される。本節で述べる方法は、近似的な信号値確率を効率良く求めるものである。信号線 a の t 時刻における信号値確率を $P_a[t]$ と表記し、ゲート f の入力信号線を x_1、x_2、$\cdots$、x_n と表記すると、遅延値0のゲート f の出力信号線 y の信号値確率は以下のように計算される。

f が AND ゲートの時　$P_y[t] = P_{x_1}[t] \times \cdots \times P_{x_n}[t]$
f が NOT ゲートの時　$P_y[t] = 1 - P_{x_1}[t]$

上式の計算の結果は、回路中に再収れんが存在しなければ正確である。

遅延素子 g の出力信号線 y の信号値確率の計算は以下のように計算できる。なお、d_{min} および d_{max} を g の遅延の最大値および最小値、Δ を遅延のとりうる場合の数 $(= d_{max} - d_{min} + 1)$、$p_d$ を遅延の値が d である確率とする。また、x を g の入力信号線とする。

$$P_y[t] = \sum_{d=d_{min}}^{d_{max}} p_d \times P_x[t-d].$$

上式により実際に信号値確率を計算するためには、遅延の値の分布を考慮する必要がある。遅延の値の分布として正規分布を仮定すると、$p_d(d = d_{min}, d_{min}+1, \cdots, d_{max})$ は、以下の値をとる。

$$p_d = \frac{{}_{\Delta-1}C_{d-d_{min}}}{2^{\Delta-1}}.$$

従って、$P_y[t]$ は以下のように計算できる。

$$P_y[t] = \sum_{i=0}^{\Delta-1} \frac{{}_{\Delta-1}C_i}{2^{\Delta-1}} P_x[t - d_{min} - i].$$

図2は1時刻めに信号線 A が立ち上がった際の信号値確率の計算の例である。この例では、回路に再収れんが存在しないので、正確な結果が得られている。

しかし、この近似計算方法には次の二つの重大な問題が存在している。

1) 信号線間の信号値確率の依存関係を把握することができないため、回路中に再収れんが存在する際には、正確な信号値確率を求めることができない。

2) 時刻間での信号値確率の依存関係を把握できないため、正確な歩留まりを計算することができない。

本稿では、4章で正確な信号値確率及びエラー確率を計算する手法を提案し、5章では、近似計算手法による結果の不正確さについて議論する。

3　符号化時間記号シミュレーション - CTSS

本章では、確率的符号化時間記号シミュレーションの基礎となっている符号化時間記号シミュレーション (*coded time-symbolic simulation*) について簡単に述べる。

符号化時間記号シミュレーションでは、遅延の値は未知ではあるが変化しないものと仮定する。また、時間としては通常の論理シミュレーションと同様に離散時間を扱う。すると、上下限を指定された遅延については、そのとりうる場合を数え上げることが可能になる。図1(a)の回路を例にとると、NOT ゲート B の遅延の値は、{0,1,2,3} の4つの値のうちの1つをとり、NOT ゲート C の遅延の値は、{1,2,3,4} の4つの値のうちの1つをとる。よって、16通りの場合、即ち (B のとりうる遅延の値の数) $\times$ (C のとりうる遅延の値の数) 通りの場合についてシミュレーションを行えば、遅延のばらつきによる回路のふるまいを完全に把握できる。シミュレーションしなければならない場合の数は、不確定な遅延を持つゲートの数に対して指数的に増加するため、全ての場合を個別に扱うことは非効率的である。そこで、類似した計算をまとめて行うことで、効率の良いシミュレーションを行うことを考える。図1(a)の回路を再び例にとると、本手法では、ゲート B とゲート C の遅延値をそれぞれ、$delay_B = (b_1, b_0)$ および $delay_C = (c_1, c_0)$ のように符号化する。16通りの信号線の信号値は4つのブール変数による1つの論理関数として表現される。この符号化によって不確定性を持つ遅延を含むシミュレーションを通常の記号シミュレーションに帰着することができる。これが符号化時間記号シミュレーションの基本的な考え方である。

論理回路は一般性を失うことなく、入力信号値から出力信号値を計算する機能を持つ遅延値0の論理ゲートと、出力信号値を遅延値だけ遅らせる機能を持つ遅延ゲートからなるものと考えることができる。信号線 s の t 時刻めの信号値を $s[t]$ と表記する。g の関数を f_g、ゲート g の入力信号線を x_1、x_2、$\cdots$、x_k と表記すると、論理ゲート g の出力信号線 y の信号値は以下のように計算される。

$$y[t] = f_g(x_1[t], x_2[t], \cdots, x_k[t]).$$

遅延ゲートに関しては、遅延の符号化を図3のように考える。ここで、ブール変数 b_1、b_0 は、セレクタの制御入力に相当し、4つの遅延の値のうちの1つを選択する機能を持っている。遅延の符号化を (b_1, b_0) のように行ったとすると、遅延ゲートの出力信号線 y の信号値は以下のようにして計算され

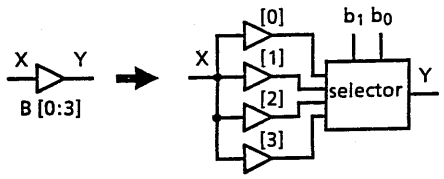


図3　セレクタを用いた不確定な遅延のモデル化

b_1	b_0	$delay_B$	c_1	c_0	$delay_C$
0	0	0	0	0	1
0	1	1	0	1	2
1	0	2	1	0	3
1	1	3	1	1	4

(a) 遅延の値の符号化

t	$A[t]$	$B[t]$	$C[t]$	$D[t]$
0	0	1	0	0
1	1	$b_1 + b_0$	0	0
2	1	b_1	$\overline{b_1 + b_0 + c_1 + c_0}$	0
3	1	$b_1 \cdot b_0$	$\overline{b_1 + c_1 \cdot b_0 \cdot c_0}$	0
4	1	0	$\overline{b_1 + c_1 + b_0 + c_0 \cdot b_1 \cdot c_1}$	0
5	1	0	$\overline{b_1 + c_1 + b_1 + b_0 + \overline{c_1 + c_0} + b_0 + c_0 \cdot b_1 \cdot c_1}$	0
6	1	0	$\overline{b_1 + c_1 + b_1 \cdot c_1}$	0
7	1	0	$\overline{b_1 \cdot b_0 \cdot c_1 \cdot c_0}$	0
8	1	0	1	0

(b) シミュレーション結果

図4　符号化時間記号シミュレーションによるシミュレーション例

る。

$$y[t] = \overline{b_1} \cdot \overline{b_0} \cdot x[t] + \overline{b_1} \cdot b_0 \cdot x[t-1] + b_1 \cdot \overline{b_0} \cdot x[t-2] + b_1 \cdot b_0 \cdot x[t-3].$$

図4は、図1(a)の回路に対してシミュレーションを行った例である。図4(a)は遅延の符号化方法を表し、図4(b)は信号線Aが1時刻めに立ち上がった際のシミュレーション結果、即ち各時刻における各信号線の信号値を表している。2時刻めのCの論理関数$\overline{b_1 + b_0 + c_1 + c_0}$は、$b_1 = b_0 = c_1 = c_0 = 0$、即ち$delay_B = 0$かつ$delay_C = 1$であるときのみ1となり、その他の場合においては0となることを表している。信号線Dの信号値は0のまま変化していないが、これは最大/最小遅延モデルでは求めることができなかった正確な結果である。本手法では信号線の信号値は全て論理関数の形で表現されているので、論理関数の内部表現が効率良いシミュレーションを行うための鍵になる。我々の実現では、共有二分決定グラフ[6]を用いている。

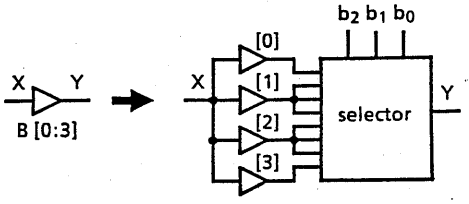


図5　確率的符号化時間記号シミュレーションにおける不確定な遅延のモデル化

4　確率的符号化時間記号シミュレーション

3章で述べてきたように、符号化時間記号シミュレーションはエラーの有無の判定だけを問題にする検証には有効である。しかし、実際には、タイミングエラーは歩留まりと深い関連があるので、エラーの存在のみならずエラーの起こる確率を求めることも重要な問題であると考えられる。遅延のばらつきによるエラー確率の増大は歩留まりの低下となって現れる。一方、遅延のばらつきによるエラーの可能性が存在しても、その確率が十分小さければ、歩留まりに対してほとんど影響を与えないので無視することができる。2.2節で述べたように、タイミングエラーの確率を正確に計算することは難しい問題である。本章では、タイミングエラーの確率を正確に求める新しい手法として、確率的符号化時間記号シミュレーション(probabilistic CTSS)を提案する。

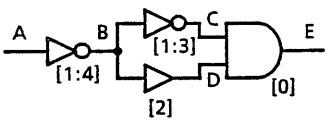
4.1　確率的符号化時間記号シミュレーションの概念

確率的符号化時間記号シミュレーションは、正確な信号値確率を求めることによってタイミングエラー確率の解析を行う。

符号化時間記号シミュレーションでは、図3のようにセレクタにより遅延の値を選択する回路を用いて遅延のモデル化を行っていた。このモデル化では1つの遅延の出力を1つのセレクタのデータ入力に割り当てている。確率的符号化時間記号シミュレーションにおける遅延のモデル化では、図5のように1つの遅延の出力を複数のデータ入力の割り当てることを許す。データ入力の重複は遅延値のばらつきに対応する。図5の例は、遅延値0、1、2、3における重複度は、それぞれ1、3、3、1であり、その遅延値をとる確率はそれぞれ1/8、3/8、3/8、1/8であることを表現している。このような回路に対して記号シミュレーションを行うと、得られる論理関数の真理値表密度は信号値確率を表すようになる。確率的符号化時間記号シミュレーションは、信号値を表現する論理関数により、回路中の再収れんによる信号線の依存関係に関する情報を保持しているので、回路中に再収れんが存在しても正確な結果を得ることができる。

4.2　遅延の符号化および論理関数の表現

遅延のとりうる場合の数をΔ $(= d_{max} - d_{min} + 1)$と表記する。遅延の値の分布として正規分布を仮定すると、i番め



(a) 再収れんのある回路

b_2	b_1	b_0	$delay_B$
0	0	0	1
0	0	1	2
0	1	0	2
1	0	0	2
0	1	1	3
1	0	1	3
1	1	0	3
1	1	1	4

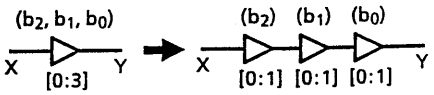
c_1	c_0	$delay_C$
0	0	1
0	1	2
1	0	2
1	1	3

(b) 遅延の値の符号化

t	$A[t]$	$B[t]$	$C[t]$	$D[t]$	$E[t]$	den
0	0	1	0	1	0	0
1	1	1	0	1	0	0
2	1	$b_2+b_1+b_0$	0	1	0	0
3	1	$b_2\cdot b_1$ $+b_2\cdot b_0$ $+b_1\cdot b_0$	$\overline{b_2+b_1+b_0+c_1+c_0}$	1	$\overline{b_2+b_1+b_0+c_1+c_0}$	0.031
4	1	$b_2\cdot b_1\cdot b_0$	$\overline{b_2+b_1+b_0}\cdot\overline{c_1\cdot c_0}$ $+\overline{b_2\cdot b_1+b_2\cdot b_0}\cdot\overline{c_1\cdot c_0}$ $+\overline{b_1\cdot b_0}\cdot\overline{c_1\cdot c_0}$	$b_2+b_1+b_0$	$\overline{c_1+c_0}\cdot\overline{b_2\cdot b_1+b_2\cdot b_0}$ $\overline{c_1+c_0}\cdot\overline{b_1\cdot b_0}$	0.094
5	1	0	$\overline{b_2+b_1+b_0}$ $+\overline{b_2\cdot b_1+b_0}\cdot\overline{c_1+c_0}$ $+\overline{b_2\cdot b_1+b_2\cdot b_0}\cdot\overline{c_1\cdot c_0}$ $+\overline{b_1\cdot b_0}\cdot\overline{c_1\cdot c_0}$	$b_2\cdot b_1$ $+b_2\cdot b_0$ $+b_1\cdot b_0$	$\overline{c_1+c_0}\cdot(b_2\cdot b_1+b_2\cdot b_0)$ $\overline{c_1+c_0}\cdot b_1\cdot b_0$	0.094
6	1	0	$\overline{b_2\cdot b_1\cdot b_0\cdot c_1\cdot c_0}$ $+\overline{b_1\cdot b_0\cdot c_1\cdot c_0}$ $+\overline{b_0\cdot c_1\cdot c_0}+\overline{c_1+c_2}$	$b_2\cdot b_1\cdot b_0$	$\overline{c_1+c_0}\cdot b_2\cdot b_1\cdot b_0$	0.031
7	1	0	$\overline{b_2\cdot b_1\cdot b_0\cdot c_1\cdot c_0}$	0	0	0
8	1	0	1	0	0	0
$yield$					c_1+c_2	0.75

(c) シミュレーション結果

図6　確率的符号化時間記号シミュレーションによる
シミュレーション例



(a) 遅延素子　　(b) 遅延素子の連結

図7　不確定な遅延のモデル化

の遅延の値に関する重複度は ${}_{\Delta-1}C_i$ となる。重複度の合計は $2^{\Delta-1}$ となるので遅延の符号化には $\Delta-1$ 個の変数を要する。

符号語の割り当てについては、図6(b)のように重みが i の符号語を i 番めの遅延値に割り当てる。この符号化方法は、図7のように、遅延の最小値が0、最大値が N である遅延ゲートを、遅延の値が0か1である遅延ゲートが N 個つながった回路に置き換えてシミュレーションすることを意味している。即ち、図7(a)の回路の確率的符号化時間記号シミュレーションによるシミュレーション結果は、図7(b)の回路の符号化時間記号シミュレーションによるシミュレーション結果に等しくなる。この符号化方法は、符号化時間記号シミュレーションにおいて共有二分決定グラフが少ない記憶量で表現されていることを考えると、もっとも良い符号化であると思われる。また、我々が使用している共有二分決定グラフに基づく論理関数処理プログラム[5]は、論理関数の真理値表密度を簡単に求めることができるので、信号値確率を効率良く算出することができる。

図6はシミュレーションの例である。図6(c)は1時刻に A が立ち上がった時のシミュレーション結果である。表中の欄 y が論理関数を、den が真理値表密度、即ち信号値確率を表している。3時刻めから6時刻めまでの信号値確率は、ハザードが発生する可能性があることを示している。

4.3　エラー確率の解析

前節のシミュレーション結果によれば、回路に出力の3時刻めから6時刻めまでの信号値確率はそれぞれ0.031、0.094、0.094、0.031である。これらの確率は互いに依存関係を持っているので、ハザードが生じる確率をこれらの確率から単純に計算することはできない。しかし、論理関数 $E[3]$、$E[4]$、$E[5]$、$E[6]$ は、これらの確率の間に存在する依存関係に関する情報を保持しているので、これらの論理関数からハザードが生じる確率を求めることができる。符号化時間記号シミュレーションの結果解析手法として以前に提案した、有限オートマトンの記号シミュレーションによりシミュレーション結果と設計者の期待値とを照合する結果解析手法[7]を、確率的符号化時間記号シミュレーションの確率解析に応用する。

まず、設計者の要求を正則表現で記述し、それをもとにして設計者の要求を満たす時に限り1を出力するような有限オートマトンを作成する。そして、確率的符号化時間記号シミュレーションの結果として出力された、各時刻の回路の出力信号線の論理関数をこの有限オートマトンに入力する。シミュレーションが終了した時点でオートマトンから出力される論理関数の真理値表密度が、回路が正常に動作する確率を示している。即ち、得られた論理関数の真理値表密度が1であることは、その回路は素子の遅延の値に依存せず正常に動作することを意味し、真理値表密度が1でない場合には、遅延の値の組合せによっては誤動作の可能性があることを意味している。

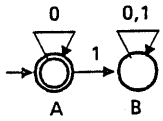


図8　仕様を表現するオートマトン

前節の例についてこの解析手法を適用する。ハザードが生じないという仕様を表現したオートマトンは図8のようになる。この2状態のオートマトンから、次のような遷移関数および出力 ok を持った順序機械を作成できる。ここで、y は状態変数を e は回路の出力信号値、即ち順序機械の入力を表している。

$$(y', ok) = (y + e, \bar{y})$$

この順序機械を回路とともにシミュレーションすることにより、図7(c)の $yield$ の欄に見られる論理関数を得ることができる。この論理関数の真理値表密度がハザードが起こらない正確な確率である。

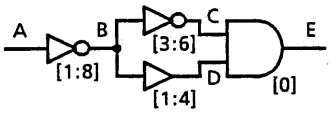
5　考察および性能評価

5.1　シミュレーション結果の正確さについて

本節では、2.2節で述べた近似計算手法が不正確な結果を出力するのに対して、確率的符号化時間記号シミュレーションが正確な結果を得ることができる3つの例を示す。まず、近似計算手法によって求められる信号値確率が悲観的である例を、次に近似計算手法によって求められる信号値確率が楽観的である例を示す。最後に近似計算手法によって計算されたエラー確率の誤差が大きくなる例について示す。

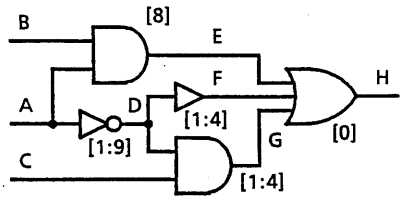
5.1.1　近似計算手法では悲観的な信号値確率を出力する例

図9は、近似計算手法では信号値確率を過大評価してしまう例である。表中の欄 $conv.$ が近似計算手法によって求めた信号値確率を、$PCTSS$ が確率的符号化時間記号シミュレーションによって求めた正確な信号値確率を表している。近似計算手法で計算された回路出力の信号値確率は、確率的符号化時間記号シミュレーションで求めることのできる正確な値に比べて、約10倍となっている。この悲観性は、再収れんによって生じる信号線 C と信号線 D の間の依存関係を考慮できないことに原因がある。この悲観性は、最大/最小遅延モデルによるシミュレーション結果の悲観性と密接な関係にある。



t	A	E	
		$Conv.$	$PCTSS$
0	1	0	0
1	1	0	0
2	1	0	0
3	0	0	0
4	0	0	0
5	0	0.001	0
6	0	0.009	0.001
7	0	0.034	0.003
8	0	0.065	0.004
9	0	0.065	0.004
10	0	0.034	0.003
11	0	0.009	0.001
12	0	0.001	0
13	0	0	0
$yield$		0.799	0.968

図 9　近似計算手法による悲観的な解析結果



t	A	B	C	H	
				$Conv.$	$PCTSS$
0	0	1	1	1	1
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	0.999
5	1	1	1	1	0.993
6	1	1	1	0.995	0.964
7	1	1	1	0.962	0.882
8	1	1	1	0.850	0.719
9	1	1	1	0.624	0.494
10	1	1	1	1	1

図 10　近似計算手法による楽観的な解析結果

5.1.2　近似計算手法では楽観的な信号値確率を出力する例

信号値確率を過大評価することは、望まれることではないが、安全ではある。[1]しかし、近似計算手法による解析では、信号値確率を過小評価してしまう場合が存在する。図 10 の例では、回路の出力の 9 時刻めの信号値確率は、確率的符号化時間記号シミュレーションでは 0.624 と正確に計算されるのに対して、近似計算手法では 0.494 という、正確な値に比べて小さい値が出力される。この楽観性も、信号線 E と F の信号値確率が独立であるという仮定で計算をしていることに原因がある。確率的符号化時間記号シミュレーションでは、論理関数によって各信号線の信号値確率間に存在する依存関係を正確に把握することができるので、正確な信号値確率を求めることができる。

5.1.3　エラー確率の計算

エラー確率の計算においては、近似計算手法によって求めた値の誤差はさらに大きくなる。確率的符号化時間記号シミュレーションでは信号線間の依存関係のみならず、同一信号線中の時刻間の依存関係も正確に把握できるので、4.3 節で述べた、オートマトンの記号シミュレーションに基づく解析手法を用いることにより正確なエラー確率を求めることができる。一方、近似計算手法においては、異なる時刻間の依存関係を把握することは非常に困難であるので、近似計算手法によって求めたエラー確率は悲観的にならざるを得ない。5.1 節の図 9 は、近似計算手法による値の誤差が大きい例である。$yield$ の欄は、回路の出力信号線 E にハザードが起こらない確率を表している。確率的符号化時間記号シミュレーションでは、99% という値を計算するのに対して、近似計算手法では、各時刻において出力信号値が 0 となる確率の積とみなして計算するので、79% と計算される。

[1]本節の例はすべて信号値 1 が出力されることがエラーと考えられる例であるため、信号値確率の過大評価は悲観的、過小評価は楽観的となる。

5.2　性能評価

プログラムを Sun3/60 上の C 言語で実現し、性能評価を行った。

実験結果を表 1 に示す。$circuit$ 欄の addern は、n ビット桁上げ加算器を、multn は、n ビット配列型乗算器を表し、N は回路のゲート数を表している。$CTSS$ の欄は符号化時間記号シミュレーションによる結果を、$PCTSS$ は、確率的符号化時間記号シミュレーションによる結果を表している。ノード数の欄はシミュレーションに要した共有二分決定グラフのノード数を表し、CPU 時間はシミュレーションに要した CPU 時間 (単位は秒) を表している。遅延は全ゲートについて最小値を 1、最大値を Δ とした。初期値をあるパターンに設定した後、1 時刻めに全ての信号値が反転した際のシミュレーションを、回路中の信号値の変化が存在しなくなる時刻まで行った。共有二分決定グラフのノード数は 100000 ノード (約 10MB) を上限とし、それ以上のノードが必要な場合にはシミュレーションを途中で打ち切った。(表中の "-"で示す)。

この表から、加算器などの性質の良い回路においては、回路規模の大きさにかかわらず、確率的符号化時間記号シミュレーションは、実行時間、必要な記憶量ともに符号化時間記号シミュレーションの 2 倍弱しか必要としないことがわかる。

遅延の幅の大きさに対しては、必要な記憶量は遅延符号化の際に要した変数の数にほぼ比例していることがわかる。実行時間は、符号化時間記号シミュレーションと比較して長時

表1　実験結果

回路	N	Δ	CTSS ノード数	CTSS CPU(秒)	PCTSS ノード数	PCTSS CPU(秒)
adder1	6	4	12	0.2	23	0.9
adder2	12	4	56	0.6	89	1.0
adder4	24	4	293	2.3	445	3.9
adder8	48	4	1794	17.9	2698	22.6
adder16	96	4	12703	137.4	18118	246.2
mult2	16	4	286	0.7	463	1.3
mult4	88	4	-	-	-	-
adder4	24	8	988	12.9	2332	28.3
adder4	24	16	2779	74.3	10498	1650.3

間かかっているが、イベント駆動方式を採用するなど、実現上の工夫で短縮は可能であると思われる。

6　おわりに

本稿ではエラー確率を正確に求める手法として、確率的符号化時間記号シミュレーションを提案した。本手法は、符号化時間記号シミュレーションを基にしており、再収れんの存在にかかわらず正確な結果を得ることができる。確率的符号化時間記号シミュレーションは、信号線間および時刻間の依存関係を論理関数の形で保持することにより正確な結果を得る。また、オートマトンの記号シミュレーションによって、正確なエラー確率を効率良く計算することができる。

本手法を大規模な回路にそのまま適応することは、計算量ならびに記憶量の面から考えて困難であると思われる。今後の課題としては近似記号シミュレーションを導入し、近似計算手法と本手法を組み合わせることで、効率の良いシミュレーションを行うことが挙げられる。また、正規分布でない遅延の値の分布への適応や、動的遅延モデル(遅延の値がシミュレーション中に変化することを可能とするモデル)への適応なども今後の課題としたい。

謝辞

共有二分決定グラフに基づく論理関数処理プログラムを提供して頂いた湊真一氏(現在NTT)に感謝します。また、平石裕実助教授、高木直史博士、澤田宏氏をはじめ、御討論頂いた矢島研究室の諸氏に感謝します。

参考文献

[1] M. A. Breuer and A. D. Friedman: Diagnosis & Reliable Design of Digital Systems, Computer Science Press, (1976).

[2] T. Yoneda, K. Nakade and Y. Tohma: A Fast Timing Verification Method Based on the Independence of Units Proc. FTCS-19, pp. 134-141, (1989).

[3] N. Ishiura, M. Takahashi and S. Yajima: Time-Symbolic Simulation for Accurate Timing Verification of Asynchronous Behavior of Logic Circuits, Proc. 26th DAC, pp. 497–502, (1989).

[4] E. Cerny, P. Rioux and C. Berthet: Comparison of Specification and Implementation for Asynchronous Circuits with Arbitrary Delays, Proc. IFIP International Workshop on Applied Formal Methods for Correct VLSI Design, pp. 704–720, (1989).

[5] S. Minato, N. Ishiura and S. Yajima: Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation, Proc. 27th DAC, pp. 52–57, (1990).

[6] R. Bryant: Graph-Based Algorithms for Boolean Function Manipulation, IEEE Trans. Comput., vol. C-35, no. 8, pp. 677-691, (1986).

[7] N. Ishiura, Y. Deguchi and S. Yajima: Coded Time-Symbolic Simulation Using Shared Binary Decision Diagram, Proc. 27th DAC, pp. 130–135, (1990).