設計自動化　５０－２
(1989. 12. 14)

ハードウェア記述言語の意味定義のための非決定的動作モデルシミュレータ

淡海功二　　安浦寛人　　田丸啓吉

京都大学 工学部 電子工学科

あらまし　現在，国内で標準化作業が進められている機能記述言語 UDL/I の意味定義の基本となる計算モデルとして NES(Nondeterministic Event Sequence) モデルが用いられている．NES モデルは非決定性を記述する動作モデルであり，厳密な意味定義が可能である反面，その記述から動作を類推することは必ずしも容易ではない．そこで我々は，言語設計を支援するために NES シミュレータを試作した．本稿では，主に非決定性の処理を中心に，意味定義における問題点などについて議論する．今回の NES シミュレータの試作において得られた知見は，UDL/I のシミュレータを作成する上で多くの示唆を与えると思われる．

A Nondeterministic Model Simulator for Definition of Semantics of Hardware Description Languages

Koji TANKAI, Hiroto YASUURA and Keikichi TAMARU

Department of Electronics, Faculty of Engineering, Kyoto University
Kyoto 606, Japan

Abstract　A new computation model of hardware behavior, called NES (Nondeterministic Event Sequence) model, were proposed for definition of semantics of hardware description language UDL/I, which is developed in a standardization project in Japan. Since NES model includes nondeterministic operation, it is not always easy to understand intuitively exact behavior of a language description. For helping the work of semantics definition of UDL/I, we developed a NES model simulator, which simulates behavior of the description completely. We also discuss problems on the relation between simulation and nondeterminism.

1 はじめに

近年，ハードウェア記述言語の標準化 [1][2] に関する研究が内外で盛んに行われるにつれ，言語の意味の統一に関する問題 [3] が顕在化してきている．記述言語の厳密な意味定義は，曖昧さを無くし，言語とハードウェアとの明確な対応付けを行う上で非常に重要であるにもかかわらず，現状では，その方法論，即ち意味定義手法は確立しているとは言い難い．

現在，電子協を中心に進められている国内の標準化活動 [2] では，形式的意味定義を与える NES(Nondeterministic Event Sequence) モデル [4] が提案され，その上に標準化言語を構築する手法が採られている．NES モデルは，最も広範に用いられているシミュレーション方式であるイベントドリブン法を一般化した計算モデルであり，非決定的動作が記述できることを特徴としている．しかし非決定性のため，その動作を NES プログラムやシミュレーション結果から直感的に理解することは，一般には困難であることが多い．そのため，NES モデルシミュレータが必要であると考え，試作した．これは，標準化作業の支援になると考えられる．

一方，標準化という立場から離れてみても，今後ハードウェア記述言語はシミュレーションのみならず，論理合成や検証にも利用されると考えられる．論理合成時代の言語は，合成系に対する仕様を記述できることが要求される．しかし一般に，この段階で対象とする回路のすべての動作を規定することは難しく，例えば遅延の幅などに見られるように，不確定性を含んだ記述となる．これを言語の上で厳密に記述するには，何らかの形で非決定性を採り入れる必要性が生じて来る．しかし，このような非決定性の導入は，現在のシミュレーション技術との間にギャップ，つまり不整合を生じる．ここでは，この問題の解決への一つの足がかりとして，非決定性によって発生する全ての場合を完全にシミュレーションする手法に基づき，NES モデルシミュレータを試作した．これにより，非決定的な意味を持つ言語のシミュレーションに関する問題点を明確化する．さらに，この非決定性によって全ての場合を計算するシミュレーションを基準として，決定的アルゴリズムと多値論理を用いた実用的なシミュレータの精度，あるいは精度とスピードのトレードオフなどの一般的な議論が可能なのではないかと考えている．

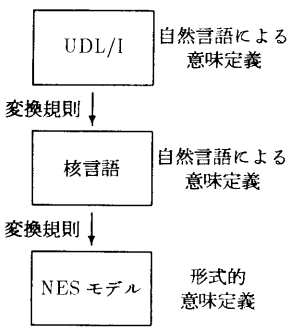


図 1　意味定義の手法

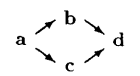
2 NES モデルと意味定義

2.1 意味定義手法

まず，NES モデルの位置づけとして，意味定義の手法を図 1 に示す．現在，国内で標準化の進められているハードウェア記述言語 UDL/I は，核言語に変換されて意味付けが行われる．核言語は，組合せ論理に関する記述と記憶素子に関する記述からなり，代入の衝突の解決法などが非決定的に決められている．さらに核言語は NES モデルに変換され，数学的に厳密な意味付けがなされる．UDL/I と核言語の対応付けは，最終的にはマニュアルなど自然言語によって与えられるが，この数学的基盤は構文木間の変換規則として形式的に与えられる．核言語と NES モデルの対応は，NES プログラムと呼ばれる抽象機械 (NES モデル) 上のプログラムとして与えられる．すなわち，核言語の各記述に対応する NES プログラムが，核言語の意味を与えることになる．NES モデルシミュレータは，NES モデルの上で NES プログラムの動作をシミュレーションするものである．

2.2 NES モデルの定義

基本的に NES モデル [4] は，従来から広く用いられてきたイベントドリブン法を一般化した考えに沿って構成されている．信号値の時系列は，イベントの一次元系列によってモデル化される．そのイベントの順序は，時間的発生順序を表している．また，各イベント間に同時の概念はない．つまり，一般に用いられる同時は，それに対応する半順序関係を満たす全ての全順序関係の集合によって表される．図 2(a) の半順序関係の場合，(b) の 2 つの全順序関係の集合として表される．信号線はプレースと呼ばれるが，物理的なプレース以外に仮想的な信号



(a) グラフによる表現

{abcd, acbd}

(b) 集合による表現

図 2　半順序関係の表現法

線として，時刻変化を与える @t，信号値系列の開始，終了を表わす @sys が記憶のないプレースとして加えられる．イベントはそのプレースの値の変化であり，プレースと値の2つ組 (@sys, start), (@t, ns), (a, 1), ... のように表わされる．NES プログラムは，5つ組 B = {Pm, Pn, M, V, R} で表わされる．Pm は記憶のあるプレースの集合，Pn は記憶のないプレースの集合，M は内部記憶 (内部変数) の集合，V は値の集合，R は動作規則の集合である．動作規則は，発火条件とアクションの集合から成り，アクションは内部条件の更新と出力イベント列から成る．この発火条件が真である時にそれに続くアクションが実行される．そのシミュレーションは，以下のように行われる．

NES モデルにあるイベント系列が入力として与えられると，NES モデルは入力イベントを逐次走査しながら，そのとき読み込んだイベントと現在の内部状態によって，NES プログラムに基づき出力と次の内部状態を決定する．出力イベント列は，そのとき読み込んだイベントの直後に挿入される．この動作は，

1. ヘッドの位置のイベントを読む．
2. NES プログラム中の実行可能なアクションを全て実行する．
   - 内部記憶の更新
   - 出力イベント列を生成，イベント系列に挿入

   ただし，これらは非決定的に行われる．
3. ヘッドを1つ右に動かす．

の繰り返しである．この様子を図3に示す．図中の右の状態は，左の状態に2つのイベント列 $\alpha\beta$ が挿入された直後の状態である．

3 NES モデルシミュレータ

3.1 NES モデルのシミュレーション

NES モデルは 3.3節で述べるように非決定的に動作が定義される．非決定的な動作のシミュレーションは，一般には木を探索空間とする探索問題に帰着される [5]．つまり NES モデルの1つの動作は，非決定的選択により1つのパスを選び1つの出力系列を生成する．しかし可能な系列は一般に複数であり，NES モデルではこの全ての系列をもってその動作の意味と定義する．我々はこのような観点から，可能な系列すべてを計算し，出力するシミュレータを試作した．つまり，前節で述べた NES モデルの動作アルゴリズムより得られる出力系列を，可能な全てについて，集合として出力する．

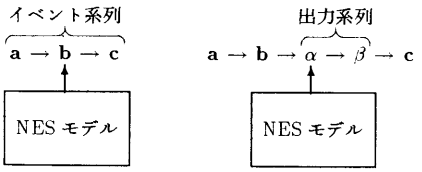


図3　NES モデルの動作

これより NES モデルシミュレータの入力は，動作モデルの記述プログラムと入力イベント系列であり，出力はイベント系列の集合となる．

3.2 NES モデルシミュレータ

NES モデルシミュレータの試作目的は，単に NES モデルプログラムの動作確認を行うためだけでなく，マニュアルを作成するときの検証ツール，言語設計者のためのツールとしての利用も考えている．さらに発展として，NES モデルシミュレータを使って，既存のシミュレーション技法の精度やスピードについて議論することも考えている．但し現段階では，扱える回路規模やスピードよりも，あらゆる非決定的な動作を完全にシミュレートし，可能な系列をすべて出力することに重点を置いている．

コーディングは，NES モデルの今後の変更や拡張の可能性，データ構造，データ領域の管理，式の再評価などの観点から，Lisp を用いて行った．これにより，NES プログラムの構文解析，イベント系列の操作，関数の外部登録などが容易に実現できる．また，内部表現の変換部をシミュレータから完全に分離することで，今後の NES モデルの変更などに対しても容易に対応できる．

データ構造は，イベント系列，NES プログラムについてはリストを用いている．リスト構造は，イベント駆動方式においてイベントの削除・挿入が容易で，特に NES モデルにおいてはイベント系列は一次元であり，整合がよく，扱い易い．NES プログラムは，Lisp による式の動的な評価のために，S-式に変換される．NES プログラム中の変数は，参照の高速化のために配列に格納している．

非決定性による探索木の探索法としては，深さ優先探索や幅優先探索などが考えられるが，記憶容量の面やプログラムの容易さなどから深さ優先探索の方が有利である [6]．これはいわゆるバックトラックを行うことであり，ここでは再帰呼び出しによって実現している．

3.3 非決定的動作のシミュレーション

NESモデルにおける非決定性には，次のものがある．

1. 成立した発火条件に続くアクションが複数である
2. 複数の発火条件が満たされた場合は，
   (a) 同じ内部変数に異なる値の更新がある
   (b) アクションにイベント列を持つものが複数である

シミュレータは，イベントが読み込まれるごとに以下の処理を行う．

1. 全ての動作規則の発火条件の真偽を調べる
2. 成立したものが1つの時
   (a) それに続くアクションが1つならば，それを実行
   (b) 複数であれば，非決定的に順次実行
3. 成立したものが複数の時
   (a) 各動作規則からアクションを1つずつ選ぶ全ての場合の組合わせ，つまり組み合わせ $a_1 \sim a_n$ の集合Aを生成
   (b) $1 \sim n$ について，a_i の全ての内部変数更新の値を計算し，同一変数に異なる値を割り当てる更新は，各変数から1つずつ更新を選ぶ全ての組み合わせ $u_1 \sim u_m$ の集合Uを生成する．残りの更新の集合は d とする．a_i の出力イベント列に関しては，各イベント列を全てシャフル演算した結果のイベント列 $o_1 \sim o_l$ の集合をOとする．
   (c) $n = m = l = 1$ ならば非決定性は生じない．それ以外は，$a_1 \sim a_n$ について，$1 \leq j \leq m$, $1 \leq k \leq l$ なる全ての j, k の組み合わせについて，u_j, d, o_k を実行する．

NESモデルシミュレータは各節点において，イベントを読み込む．そしてそのイベントと内部状態から，上記のアルゴリズムにより，次の探索枝を非決定的に選択する．このような状態遷移による木の探索がNESモデルのシミュレーションということも出来る．

```
D=(Pm,Pn,M,V,R), Pm={a,b,y} , Pn={@s,@t} ,
M={state,M,T,min_delay,max_delay} ,
V={start,end,s0,s1,ns,PI} ,
R={
  /* initialize */
  if place=@s & val(@s)=start
      then T:=0, state:=s0, y:=0
           min_delay:=1, max_delay:=2 end,

  /* event on input places */
  if (place=a | place=b) & val(state)=s0
         & val(y) != nand(val(a),val(b))
      then M:=nand(val(a),val(b)),
           state:=s1, T:=0 end,

  /* input event before output */
  if (place=a | place=b) & val(state)=s1
         & val(y) = nand(val(a),val(b))
      then state:=s0 end,

  /* event on time place "@t" */
  if place=@t & val(@t)=ns & val(state)=s1
      then T:=val(T)+1 end,

  /* output */
  if place!=@s & val(state)=s1
         & !((place=a | place=b)
         & val(y) = nand(val(a),val(b)))
         & (place=@t & val(T)+1=val(min_delay)
         | val(T)>=val(min_delay)
         & val(T)<=val(max_delay))
      then state:=s0 output (y,val(M))
      or   /* do nothing */ end,

  /* failure */
  if val(state)=s1 & (place=@s
         | place=@t & val(T)>=val(max_delay))
      then Failure end
}
```

図4　NANDゲートのNESプログラム

4 実行例と性能評価

1 ～ 2nsの不確定な遅延(慣性遅延)をもったNANDゲートのNESプログラムを図4に示す．入力線はaとb，出力線はyであり，最大/最小遅延が表現されている．これはまた，イベント間の順序の非決定性も含んだ記述である．このNESプログラムに対する入力イベント系列の例と，実行結果を図5に示す．また，このプログラムに対し，入力イベント系列を変えてシミュレートした時の実行時間を表1に示す．他の回路素子のNESプログラムでも，傾向としては同じような結果が得られた．これより，計算時間に関し，以下のような事が言える．

葉の数	出力系列数	入力系列長	run time (sec)	real time (sec)
7	5	11	1.25	2.30
11	8	14	1.90	4.63
26	19	16	4.97	14.3
84	59	19	12.8	42.5
182	129	23	29.7	99.4
261	185	27	47.0	163
862	611	33	191	641
2847	2018	39	666	2350

表1: NANDゲートモデルの実行時間

```
input sequence:
(@s,start)(@t,ns)(a,1)(b,0)(@t,ns)(@t,ns)(b,1)(@t,ns)(@t,ns)(@t,ns)(@s,end)

>nes nand.nes nand.event
KCl (Kyoto Common Lisp) September 30, 1987
Loading init.lsp
(@s,start)(@t,ns)(a,1)(b,0)(@t,ns)(y,1)(@t,ns)(b,1)(@t,ns)(y,0)(@t,ns)(@t,ns)(@s,end)
(@s,start)(@t,ns)(a,1)(b,0)(@t,ns)(y,1)(@t,ns)(b,1)(@t,ns)(@t,ns)(y,0)(@t,ns)(@s,end)
(@s,start)(@t,ns)(a,1)(b,0)(@t,ns)(@t,ns)(y,1)(b,1)(@t,ns)(y,0)(@t,ns)(@t,ns)(@s,end)
(@s,start)(@t,ns)(a,1)(b,0)(@t,ns)(@t,ns)(y,1)(b,1)(@t,ns)(@t,ns)(y,0)(@t,ns)(@s,end)
(@s,start)(@t,ns)(a,1)(b,0)(@t,ns)(@t,ns)(b,1)(@t,ns)(@t,ns)(@t,ns)(@s,end)

5 event sequences have been obtained.
the cases of reaching "FAILURE":  2
real time :  3.367 secs
run time  :  1.250 secs
Bye.
```

図5　入力と実行結果

計算時間は，同一モデルに関する限りでは，探索木の節点数を n とすると，$O(n)$ 程度と考えられる．しかし n は入力そのものではなく，入力である NES モデルプログラムや入力イベント系列の記述量と複雑度に関係した変数であるので，定式化は容易ではない．あえて入力系列長 l(入力イベント系列に含まれるイベントの数) について注目すると，最悪の場合，$O(2^l)$ であると考えられる．

当初の NES モデルの目的が言語の各文の表す動作，すなわち基本的回路動作に対する意味定義であることを考えると，表1の結果は，この目的のためには現実的な時間内での処理が可能であることを示していると言える．

5　考察

5.1　NES のシミュレーションアルゴリズム

現在の NES モデルシミュレータは，探索木をしらみつぶしに見ている．しかし，探索木の節点の中には同じ状態を持つものがあり，これを合併してシミュレーションを簡単化できる可能性がある．言い換えれば，探索木をグラフに変換することになる．これにより，NES モデルのシミュレーションの高速化やシミュレータの記憶容量の軽減が期待できる．そして扱える問題の範囲が広くなる．ここでは，同一状態ではイベント系列のポインタが同一イベントを指していることに着目し，ある特定のイベントごとに入力イベント系列を分割して，部分シミュレーションを行う手法を考える．例えば，入力イベント系列が

abcdefgcbdhc …

で，イベント **c** に着目するとすれば，系列を **ab**，**cdefg**，**cbdh**，**c** …，の集合に分割する．そして各イベント系列についてシミュレーションが終了するごとに，同じ状態の節点をまとめる．

節点の照合に費やされる計算回数は，その時の枝別れの数を n とすると，最悪で $n(n-1)/2$ 回，最小で $n-1$ 回となる．通常，回路設計者は非決定的な枝別れがある時点 (例えばクロック) で収束する設計を行うであろうから，適切にイベントを切り分ければ，効率よく簡単化できる．注目するイベントは，非決定性が収束していると思われるイベント，つまり同期設計ならばクロック，非同期設計ならばある時間ごとの時刻イベントが適当であろう．この手法は，現在の NES モデルシミュレータで大きな変更なく実現できる．さらに，一般に非決定性を扱う機能シミュレータに応用できるのではないかと考えている．

5.2　精度とスピード

論理 / 機能レベルのシミュレータは，古くから実用的なものが作られ，そのシミュレーション手法は様々である．シミュレーションの方式やスピードについては多くの研究がなされているが，精度 [7] について述べられているものはあまり見あたらない．我々は，NES モデルによる記述を基準として，シミュレーションの精度について論じることを考えている．ここでは，いくつかの例を挙げ，簡単な考察を加える．

論理 / 機能シミュレーションには，論理とタイミングの検証という2つの目的がある．論理検証は，通常のシミュレーション以外にも記号シミュレーション，形式的推論などの手法があり，技術的にもかなり固まっている．しかしタイミング検証に関しては未解決の問題も多く，実際の設計現場においても最も大きな問題となっている．そこでここでは，主にタイミングの問題について考える．

タイミングの問題は回路素子の遅延に起因するため，遅延のモデル化が最大の鍵である．遅延モデルには，

- 標準遅延モデル
- 立ち上がり / 立ち下がり遅延
- 単純・慣性遅延

- 入力側・出力側遅延

などや，これらの組合せが用いられる．さらに，遅延のばらつきには，

- 最大 / 最小遅延
- 平均 / 標準偏差遅延
- 動的・静的遅延

などがある．最大 / 最小遅延には，両端の値だけを使うものや不定値 x を用いる多値論理などがある．動的遅延はシミュレーション中つねにばらつきを示すモデルであり，静的遅延はシミュレーションの初めに遅延が決定される．本稿の 4章で示した NAND ゲートは動的遅延であるが，1番目のアクションを

```
then delay = 1
or   delay = 2 end,
```

5番目の条件の 3 ～ 6 行目を

```
place=@t & val(T)+1=val(delay) | place!=@t
& val(T)=val(delay)
```

6番目の条件の 2 行目を

```
| place=@t & val(T)>=val(delay))
```

のように変更することで静的遅延が表現できる．このように NES モデルの振舞いは，その記述によって微妙に変化する．つまり，NES モデルによる意味付け即ち回路素子のモデル化であるといえる．

次に具体的な例について考える．図 6 の 2 ビットのセレクタ (NAND ゲートの遅延 21 ～ 30，インバータの遅延 16 ～ 25: 単位時間) を例題として採り挙げ，慣性遅延でモデル化する．この回路は，ハザードを発生する場合がある．入力を図 7 とし，以下の 3 つの手法でシミュレーションした場合と NES モデル (各ゲートは，4章のプログラムで表される) によるものとを比較を，表 2に示す．

1. 標準遅延 (最大値を使う) モデル
2. 最大 / 最小遅延 (不定値 x を使う)
3. 静的かつ同種のゲートは同じ遅延と仮定したモデル

	出力系列数	節点数	ハザード検出
1	1	10^2	不可
2	2^{45}	10^2 強	可
3	127	10^4	一部可
NES	56029	10^6	可

表 2: NES モデルと他手法の比較

最大 / 最小遅延の出力を図 7 に伴わせて示す．これらの出力を，NES による信号値系列のモデルの上で出力系列の数を考えたもの，即ち，各々の出力がその意味として含む全ての場合を考えたものが，表の出力系列数である．節点数は読み込んだイベントの数に相当し，これが計算時間を決定する．

標準遅延では，明らかにハザードは検出できない．通常，複数のパスを伝搬してきた信号は，その遅延の差が大きいときにハザードが生じる可能性がある．ゆえにこの手法は簡単ながら精度が良いとはいえない．最大 / 最小遅延は，比較的少ない計算量で意味上では多くの系列が得られ，広い範囲をカバーしているという面で，広く一般に用いられているのももっともである．しかし，悲観的すぎるという多くの指摘があるように，NES モデルでは表していない非常に多くの場合を含んでいる．この例はハザードを生じるが，実際にはハザードの生じない回路でもハザードを検出するという欠点がある．精度が良いとはいえないが，コストの割には有用である．3 の場合は比較的現実的に考え得るモデルであり，ある程度非決定性を採り入れている．NES モデルは正確ではあるが，この数ゲートの回路でも膨大な計算を要する．逆にいえば，精度を上げるにはある程度のコストは必要であ

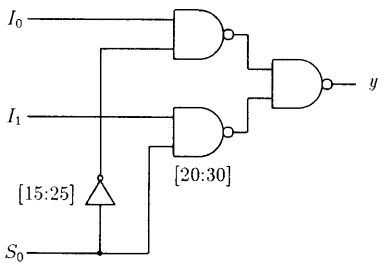


図 6　2 ビットのセレクタ

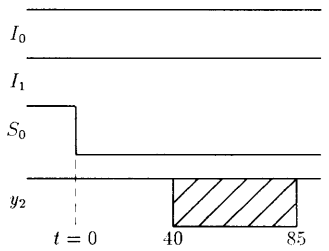


図 7　セレクタの入力と最大 / 最小遅延の出力

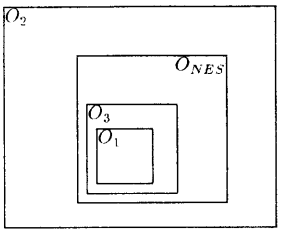


図8　各出力系列集合の包含関係

る．これらの包含関係は図8（Oの添字が各番号）に示すように，

$O_1 \subset O_3 \subset O_{NES} \subset O_2$

となり，一般に用いられている標準遅延や最大/最小遅延の性質をよく表している．

このような議論を行う場合も，NESモデルによる回路素子のモデル化，すなわちNESプログラムをどう記述するかが大きな問題である．

6　おわりに

ハードウェア記述言語の意味付けのための，NESモデルシミュレータの試作を行い，NESモデルを基礎としたシミュレーションの精度について議論した．今後も非決定性のシミュレーションアルゴリズムと，シミュレータの精度などについて研究する方針である．最後に，日頃御討論頂く京都大学情報工学教室石浦菜岐佐氏，並びに田丸研究室の諸氏に感謝致します．

参考文献

[1] M.Shadad,et al.,"VHSIC Hardware Description Language", *IEEE Computer*,18(1985),94

[2] 唐津修,"LSI設計用言語標準化委員会の活動について", 電子工業月報, vol.30,No.9,pp.47～55(Sep.1988)

[3] 安浦寛人,"LSI設計用記述言語の標準化における 意味の統一について", 信学技報,VLD88-96(Feb.1989)

[4] 石浦菜岐佐, 矢島脩三,"ハードウェア記述言語の 意味付けのための非決定的な動作モデル", 信学技報,VLD89-3(Apr.1989)

[5] 木村泉, 米沢明憲: 算法表現論, 岩波講座情報科学 12, 岩波書店 (1982)

[6] 西川, 三宮, 茨木: 最適化, 岩波講座情報科学 19, 岩波書店 (1982)

[7] E.Ulrich, D.Herbert, "Speed and Accuracy in Digital Network Simulation Based on Structural Modelling", 19th Design Automation Conference, IEEE 1982