

解 説**2. 方式・機能・論理設計における CAD****2.4 タイミング検証†**

松 下 浩 明**

1. はじめに

タイミング検証とは、設計された論理回路にタイミングに関する問題があるか否かを調べることである。ここでいうタイミングの問題とは、例えば、フリップフロップのクロック入力端子に幅の狭いパルスが入力されたり、フリップフロップ間をむすぶ組合せ回路の信号伝播時間が予想以上に長かったりすることである。このようなことが発生すると、回路は誤動作したり、期待どおりの速度で動作しなくなる。

タイミング検証は、設計された論理回路に対する検証項目の一つであり、広義には、論理検証の中に含まれる。しかしながら、最近では、論理機能の検証（狭義の論理検証）とタイミングの検証を区別することが多くなっている。その理由として、タイミング検証ツールの発達があげられる。

最初に実用化されたタイミング検証ツールは論理シミュレータである。論理シミュレータは論理機能の検証とタイミングの検証とを同時に行う。この論理シミュレーションによるタイミング検証の方法は今も広く行われているが、この方法でタイミングの問題をすべて調べ上げるのは、タイミングに関するすべての場合をつくすのに多大の計算時間を要するため、事実上、不可能である。

一方、論理回路のタイミングに関する設計誤りをなるべく早い段階で発見するために、網羅的にタイミングの検証を行う必要性が高まり、論理シミュレーションによらない方法がいくつか提案され、実用化されている。これらの方法は、その適用が主に同期式順序回路に限定されるが、実用的計算時間でタイミングに関するすべての場合をつくすという利点をもっている。また、論理機能の検証とタイミングの検証を分離する

ことにより、論理シミュレータの負荷を減らすという効果も副次的に生じている。

本稿では、2.でまず、タイミングの問題にはどのようなものがあるかを述べ、3.で、タイミング検証の基礎となる回路遅延のモデル化について述べる。4.では、今までに実用化されているタイミングの検証手法とその実現法について述べる。

2. タイミング検証の目的

通常の論理回路は、その出力が入力と内部状態で決まる順序回路である。順序回路の機能は状態遷移の順序によって規定されるが、それが実際に正しく動作するためには状態遷移の順序を常に正しく保っておく必要がある。動作中の状態遷移の順序を保証する手段として、順序を時間に関連づける非同期式、同期式と、順序を直接、時間には関連づけない自己同期式がある。自己同期式では、各素子の動作の開始はその素子の前段の素子が出す完了信号（前段の素子の動作が完了したことを示す信号）を受け取ることによって行われ、この機構によって順序が保たれる。タイミング検証の対象となるのは、非同期式、同期式の順序回路である^{2), 3)}。

非同期式順序回路では、状態遷移の順序は各素子の遅延時間によって保たれる。したがって、素子のわずかの遅延時間の変化が回路を誤動作させる原因となるため、大規模な論理回路にはこの方式は用いられない。

同期式順序回路は一つのクロック信号で記憶素子の同期をとることによって、状態遷移の順序を保つものである。通常の論理回路はこの方式である。

ここでは、同期式順序回路である論理回路を中心に、論理回路の設計において生ずる代表的なタイミングの問題にはどのようなものがあるかを分類してみる。

フリップフロップを用いた同期式順序回路の例を

† Timing Verification by Hiroaki MATSUSHITA (Information Systems and Electronics Development Laboratory, Mitsubishi Electric Corp.).

** 三菱電機(株)情報電子研究所

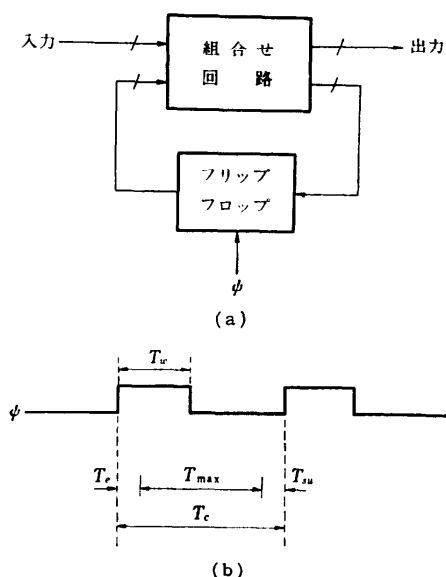


図-1 フリップフロップを用いた同期式順序回路

図-1 に示す。図-1 に示した順序回路が正しく動作するためには次のようなタイミングの制約が満たされていなければならない。以下では、フリップフロップはポジティブエッジトリガード型であると仮定して説明する。

(1) クロックの周期と組合せ回路の最大遅延との間のタイミング制約²⁾

図-1 において、フリップフロップは、クロック信号ψの電位が低電位から高電位になるときにデータを取り込み、フリップフロップ内の状態を定める。この状態は、再びクロック信号ψの電位が低電位から高電位に遷移するまで保持される。フリップフロップがデータを取り込み、フリップフロップ内の状態を定める時間、すなわち、フリップフロップの遅延時間を T_f とする。また、クロックの周期を T_c とする。フリップフロップに保持された状態はフリップフロップの出力端子から、組合せ回路をへて、次の周期のフリップフロップの状態になる。フリップフロップがデータを取り込む前のある時間、フリップフロップのデータ入力信号は安定状態になければならない。これをフリップフロップのセットアップ時間といい、 T_{fu} で表す。このとき、図-1 の順序回路が正しく動作するた

めには、次の不等式が成り立つ必要がある。

$$T_c + T_{max} + T_{fu} < T_c$$

ここで T_{max} は組合せ回路の最大遅延時間である。

(2) クロック信号に関するタイミング制約²⁾

●クロック信号のパルス幅

図-1 の順序回路において、フリップフロップが正常動作するためには、クロックが高電位である時間、すなわち、クロックのパルス幅 T_w はある定められた値より大きくなければならない。

●クロック信号のハザード (hazard)

図-1 の順序回路では、クロック信号のハザードに注意する必要がある。ハザードとは、組合せ回路の出力が定常値になる前に過渡的に定常値と異なる値になることである。クロック信号ψはそのままの形でフリップフロップに入力されず、図-2 のように、ある制御回路で制御される場合がある。制御信号をつくる組合せ回路にハザードが発生すると、フリップフロップが本来起動すべきでないときに起動してデータを取り込み、誤動作する可能性がある。

3. 回路遅延の取扱い

タイミング検証を実効のあるものにするためには、設計された論理回路の信号伝播の遅延をいかに正確に

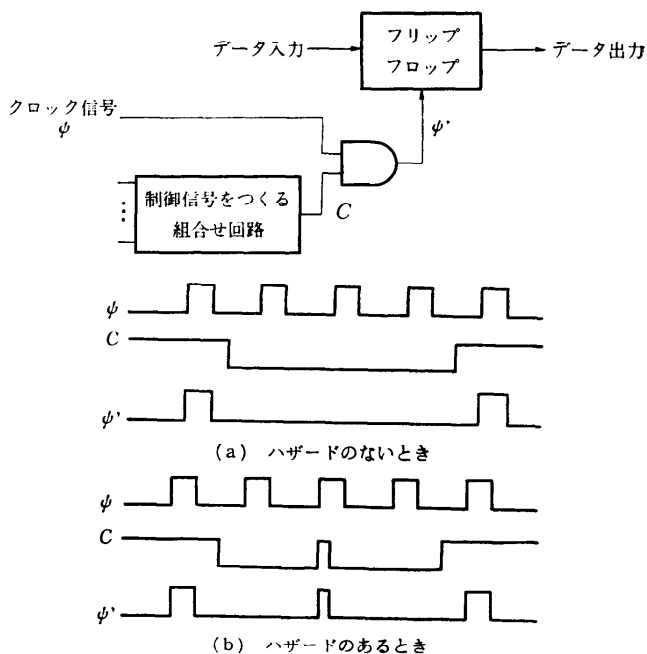


図-2 クロック信号のハザード

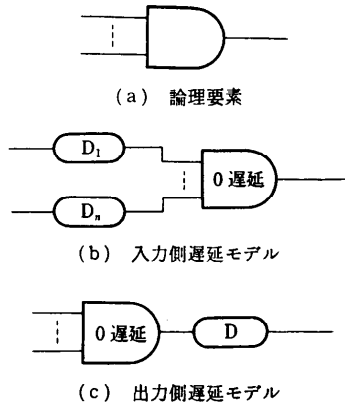


図-3 論理要素の遅延のモデル

モデル化するかが鍵になる。ここでは、回路遅延のモデル化について述べる。

3.1 遅延モデル

(1) 論理素子の遅延³⁾

AND ゲートとか、OR ゲートとかの論理素子の遅延は、一般に、図-3 に示すような入力側遅延モデルまたは出力側遅延モデルでモデル化することができる。ここで、遅延 $D_1, \dots, D_n$, D は論理素子内部の遅延と論理素子に接続している信号の配線抵抗、配線容量とで決まるものである。また、論理素子の遅延は論理素子の出力信号が低電位から高電位に遷移するとき（立上り）と高電位から低電位に遷移するとき（立下り）では異なるのが普通であるので、立上りと立下りによって区別する必要がある。

(2) パスの遅延

信号がいくつかの論理素子を伝わっていくときの遅延は、各論理素子の遅延の和として表すことができる。なお、パスの遅延においても、信号の立上りと立下りを考慮する必要がある。例えば、図-4 のパスの立上り遅延（パスの出力信号が立上るときの遅延）は 33 単位時間（以下、遅延の単位はすべて単位時間とし、混乱が生じない限り、単位を省略する）であり、立下り遅延（パスの出力信号が立下るときの遅延）は 31 である。パスの立上り遅延、立下り遅延のうち、小さい方をパスの最小遅延、大きい方を最大遅延とよぶことにする。

(3) 組合せ回路の遅延^{4), 5)}

組合せ回路の遅延を次のように定義する。組合せ回路の各入力に信号値 $(S_1, \dots, S_n)$ が割り当てられていて、その内のある入力が S_i から S_i' に遷移したとき、その信号は組合せ回路内を伝播

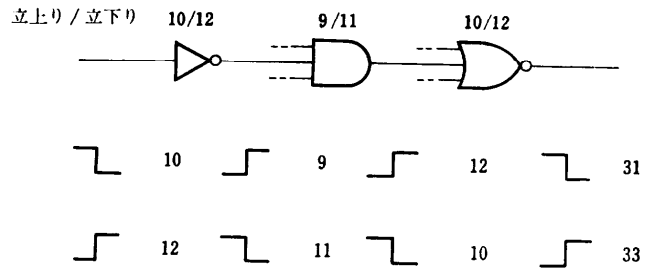


図-4 パスの遅延

し、組合せ回路の出力まで到達する。組合せ回路の最小遅延、最大遅延とは、 $(S_1, \dots, S_n)$ と (S_i, S_i') のあらゆる組合せに対して、最初に信号が出力に到達するまでの時間、および、最後に出力が安定するまでの時間である。組合せ回路の最小遅延、最大遅延をそれぞれ $T_{\min}$, $T_{\max}$ で表す。

図-5 において、各論理素子の遅延が 1 であるとき、この組合せ回路の最小遅延、最大遅延はそれぞれ 2, 4 である。この例において、パス $u \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow x$ のパス遅延が 5 であるのに組合せ回路の最大遅延がそれよりも小さいことに注意されたい。これは、パス $u \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow x$ に信号を伝播させるような入力信号の組が存在しないからである。

組合せ回路の遅延を求める問題の計算量について論じた文献は見当たらないが、大きな計算量を要する問題であることが予想される。多くのタイミング検証プログラムは上で定義した遅延の代わりに、次に定義する疑似遅延で組合せ回路の遅延をモデル化している。

組合せ回路の疑似遅延を次のように定義する。組合せ回路のすべてのパスの最小遅延のうちで最小のものを組合せ回路の最小疑似遅延と定義し、 $T_{p, \min}$ で表す。組合せ回路のすべてのパスの最大遅延のうちで最大のものを組合せ回路の最大疑似遅延と定義し、 $T_{p, \max}$ で表す。そのとき、

$$T_{p, \min} \leq T_{\min} \leq T_{\max} \leq T_{p, \max}$$

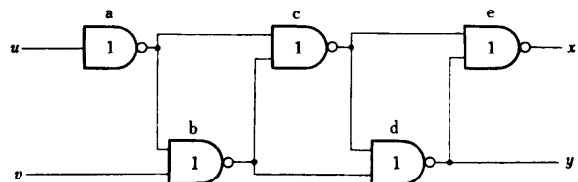


図-5 組合せ回路の遅延

の関係が常に成立する。すなわち、最小疑似遅延、最大疑似遅延は組合せ回路の最小遅延、最大遅延の下界と上界を与える。

組合せ回路内のすべての論理素子の遅延が1であるとき、組合せ回路の最大疑似遅延は段数とよばれる。

3.2 遅延のばらつき

3.1 で述べた遅延モデルは、遅延の値に不確定要素がないものである。しかしながら、遅延は製造上の理由や、論理回路が実際に動作する環境等さまざまな理由でばらつきをもっている。タイミング検証ではこのばらつきも考慮することが大切である⁴。

論理素子や配線の遅延のばらつきをモデル化するものとして次の二つが知られている。

●最大-最小遅延モデル²⁾

これは遅延のばらつきを最大遅延と最小遅延の組で表すものである。信号伝播の遅延計算が加算だけで可能であるため、論理シミュレータにおいてこのモデルがよく用いられる。このモデルはばらつきを実際のものより大きく見積りすぎる傾向がある。

●平均-標準偏差遅延モデル¹⁷⁾

これは遅延のばらつきを平均遅延と標準偏差の組で表すものである。信号伝播の遅延計算には乗算が必要であるが、最大-最小遅延モデルより実際の遅延のばらつきをよく近似している。

4. 検 証 手 法

タイミング検証手法として、これまで大別して、シミュレーションによる方法とシミュレーションは行わずに論理回路を解析する方法とが提案され、実用化されている。

4.1 シミュレーションによる方法

(1) 論理シミュレータによる方法⁹⁾⁻¹³⁾

この方法は論理シミュレータで論理回路をなるべく詳細にシミュレーションし、シミュレーションの過程でタイミングの検証を行うものである。

この方法の利点は、同期式順序回路、非同期式順序回路の双方に適用可能であるということである。

しかしながら、この方法では、タイミングに関するすべての場合をつくすには多大の計算時間を要するため、事実上、すべての場合をつくせないという短所がある。

(2) 安定/変化状態をもつシミュレータによる方法

同期式順序回路では各周期において信号が変化する時刻は常に一定しているということに着目し、通常の論理シミュレータがもっている信号状態 0, 1, R (立上り状態), F (立下り状態), U (不定状態) のほかに S (安定状態), C (変化状態) を用いてシミュレーションし、タイミングの検証を行う。

この方法は、その適用範囲が同期回路に限られるが、実用的計算時間でタイミングに関するすべての場合をつくすことができるという長所がある。

この方法を用いたものに SCALD タイミングベリファイ等がある¹⁴⁾。

4.2 シミュレーションによらない方法

シミュレーションによらずに論理回路を解析する方法は、主に、同期式順序回路に適用されるものであって、解析の中心は組合せ回路の遅延である。この方法はパスを一つ一つ列挙して解析する方法と組合せ回路全体を PERT の手法を用いて解析する方法に分けることができる。

(1) パス列挙法

パス列挙法は、まず、組合せ回路のすべてのパスを列挙し、各パスの遅延を求め、その結果に基づいて、組合せ回路の遅延を求めるものである。この方法の利点は、パスを列挙した時点で、回路の遅延の計算において考慮にいれなくてもよいパスを取り除くことができる点である。

例えば、図-5 の例において、この組合せ回路の入力 u, v から出力 x, y に至るパスは、

$$\begin{aligned} u \rightarrow a \rightarrow c \rightarrow e \rightarrow x \\ u \rightarrow a \rightarrow b \rightarrow c \rightarrow e \rightarrow x \\ \dots\dots \\ v \rightarrow b \rightarrow d \rightarrow y \end{aligned}$$

の13個あるが、ここで、パス $u \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow x$ を回路遅延の計算で考慮しなくてよいことがあらかじめ分かっているならば、組合せ回路の最小、最大遅延は2, 4であると計算される。

この方法を用いたタイミング検証ツールに GRASP 等がある¹⁵⁾。この方法の短所は、最悪の場合、指数関数的な計算時間を要することである。

(2) クリティカルパス法

PERT (Project Evaluation Review Technique) は1950年代の後半にプロジェクト管理の手法として提案されたものであるが、これを組合せ回路の遅延の

* 遅延のばらつきによって起きるタイミングの問題の代表例にクロック信号の位相ずれ、すなわち、クロックスキュー (clock skew) がある。これはクロック信号を分配する配線の遅延がばらつくことにより起こるものである。

計算に適用したのがここで述べる方法である*。

この方法で求められる遅延は3.1の(3)で述べた疑似遅延である。この方法では、実用的計算時間で遅延が求まること、クリティカルパス（遅延の最も大きいパス）が求まること、回路の遅延が期待値より大きいときどの論理素子の遅延がネックになっているかを示す指標であるスラック (slack) 情報¹⁷⁾を計算できること等の利点がある。

この方法を用いたものに、NELTAS, クリティカルパス遅延検査システム, TA 等がある¹⁶⁾⁻²²⁾。

5. お わ り に

本稿では、同期式順序回路を中心に、論理回路において起きるタイミングの問題とその検証方法について述べた。LSI の設計においては、タイミング検証は論理設計が終わった後以外に、LSI 内のレイアウト設計が終わった後にも行われる。このとき、レイアウト結果に基づいて各配線遅延を予測する遅延計算処理はタイミング検証を実効のあるものにするために重要なものであるが、本稿では、紙面の都合で述べるのを控えた⁶⁾⁻⁸⁾。

今後の課題の一つは検証の適用範囲を拡大することである。ここ数年来、論理シミュレーションによらないタイミング検証法が多く発表されている。これらの方法はいくつかのタイミングの問題にたいしては網羅的に検証可能であるという利点を有している一方、適用範囲が限定されている。

また、タイミング検証と関連した課題として、リタイミング (retiming) の問題がある²³⁾。リタイミングとは、同期式順序回路において回路構造を変更することによりクロック周期を短くすることである。論理装置の内部で使用されている LSI を n 倍の速度の LSI で置き換えることによって、論理装置を高速化しようとする場合、そのままでは通常 n 倍高速な論理装置を得ることはできないが、リタイミングにより、 n 倍に近い速度をもつ論理装置を構成し得る可能性がある。

* この方法に分類したすべてのタイミング検証プログラムが最初から PERT にもとづいて作成されたわけではない。むしろ PERT とは独立に考案されたものが多く、結果として、PERT で用いられた手法と同様なものになっている。

参 考 文 献

- 1) Ruehli, A.E. and Ditlow, G.S.: Circuit Analysis, Logic Simulation, and Design Verification for VLSI, Proc. IEEE, Vol. 71, No. 1, pp. 34-48 (1983).
- 2) 渡辺 誠: 超 LSI 設計, p. 351, 企画センター, 東京 (1983).
- 3) Mead, C. and Conway, L. (菅野, 穉訳): 超 LSI システム入門, 培風館, 東京 (1981).
- 4) Muroga, S. (室賀, 笹尾訳): 論理設計とスイッチング理論, p. 367, 共立出版, 東京 (1982).
- 5) Lai, H.C. and Muroga, S.: Minimum Parallel Binary Adders with NOR(NAND) Gates, IEEE Trans. Comput., Vol. C-28, No. 9, pp. 648-659 (1979).
- 6) Koppel, A., Shah, S. and Puri, P.: A High Performance Delay Calculation Software System for MOSFET Logic Chips, 15th DAC, pp. 405-417 (1978).
- 7) Putatunda, R.: AUTO-DELAY: A Program for Automatic Calculation of Delay in LSI/VLSI Chips, 19th DAC, pp. 616-621 (1982).
- 8) Tokuda, T., Okazaki, K., Sakashita, K., Ohkura, I. and Enomoto, T.: Delay-Time Modeling for ED MOS Logic LSI, IEEE Trans. Computer-Aided Design, Vol. CAD-2, No. 3, pp. 129-134 (1983).
- 9) Evans, D.J.: Accurate Simulation of Flip-Flop Timing Characteristics, 15th DAC, pp. 398-404 (1978).
- 10) 荻原, 松下, 村井: 順序回路のタイミングベリファイア (SIMCHK), 情報処理学会研究会報告, 電子装置設計技術 10, pp. 1-10 (1981).
- 11) Hirakawa, K., Shirai, N. and Muraoka, M.: Logic Simulation for LSI, 19th DAC, pp. 755-761 (1982).
- 12) Agrawal, V.D.: Synchronous Path Analysis in MOS Circuit Simulator, 19th DAC, pp. 629-635 (1982).
- 13) 吉田, 西本, 久保, 千葉: MOS LSI 用会話型論理シミュレーションの一手法, 電子通信学会研究会報告, CAS 83-193, pp. 49-54 (1983).
- 14) McWilliams, T.M.: Verification of Timing Constraints on Digital Systems, J. Digital Systems, Vol. 5, No. 4, pp. 401-427 (1981).
- 15) Wold, M.A.: Design Verification and Performance Analysis, 15th DAC, pp. 264-270 (1978).
- 16) Kamikawai, R., Yamada, M., Chiba, T., Furumaya, K. and Tsuchiya, Y.: A Critical Path Delay Check System, 18th DAC, pp. 118-123 (1981).
- 17) Hitchcock, R.B. Sr., Smith, G.L. and Cheng, D.D.: Timing Analysis of Computer Hardware, IBM J. Res. Dev., Vol. 26, No. 1, pp. 100-105 (1982).
- 18) Bening, L.C., Lane, T.A. and Smith, J.E.: Development in Logic Network Path Delay Analysis, 19th DAC, pp. 605-615 (1982).

- 19) Nomura, M., Sato, S., Takano, N., Aoyama, T. and Yamada, A.: Timing Verification Based on Delay Time Hierarchical Nature, 19th DAC, pp. 622-628 (1982).
- 20) Shelly, J.H. and Tryon, D.R.: Statistical Techniques of Timing Verification, 20th DAC, pp. 396-402 (1983).
- 21) Tamura, E., Ogawa, K. and Nakano, T.: Path Delay Analysis for Hierarchical Building Block Layout System, 20th DAC, pp. 403-410 (1983).
- 22) Jouppi, N.P.: Timing Analysis for n MOS VLSI, 20th DAC, pp. 411-418 (1983).
- 23) Leiserson, C.E., Rose, F.M. and Saxe, J.B.: Optimizing Synchronous Circuit by Retiming, 3rd Caltech Conference on Very Large Scale Integration, pp. 87-116 (1983).

* DAC: Proceedings of Design Automation Conference

(昭和59年6月4日受付)