C-038

Time-Petri Net を用いた非同期回路のモデル化と階層化設計

Modeling and Hierarchical Design for Asynchronous Circuits based on Time-Petri Net

松山千尋 †　　和崎克己 †
Chihiro Matsuyama　Katsumi Wasaki

1　はじめに

組み合わせ論理回路は，閉路上にラッチを有しない非同期ループが存在する場合，系全体の安定性がゲート素子の遅延時間に依存する [1]．特に，統一されたクロック同期を必要としない非同期・組み合わせ論理回路は，遅延時間を最小化できる利点がある．しかし，閉路上にラッチを有しない非同期ループが存在する場合，系全体の安定性はゲート素子の遅延時間に依存する欠点も有する．非同期回路の安定性解析は一般に難しく，信号点の状態変化に着目したトレース理論を用いた方法などが提案されている．他方，信号点の状態変化（H → L、L → H などのエッジ）をメッセージとして後段のゲートへ順次送る Petri Net でのモデル化がある [2][3]．現実の論理ゲートには遅延時間が存在するので，非同期回路の振る舞いを Time-Petri Net を用いて検査する手法は有用である [4].

本研究では，Time-Petri Net を用いて論理ゲートをモデル化し，各素子を階層的に接続して非同期回路のモデル作成を行った．作成した論理ゲートは 2 入力論理和や 2 入力排他的論理和などの基本的なものであり，それらを用いて作成した組み合わせ非同期論理回路として，半加算器，全加算器，Muller's C-element[5] などの機能演算素子を作成した．更に，作成した Muller's C-element を用いた非同期ループ接続によって Micropipeline(Sutherland, 1989) に基づいた FIFO[6] を構成した．各回路の設計と合成ならびに動作の確認は，著者らによって開発中の Petri Net モデル作成援用ツール "HiPS" によって行った [7].

2　Petri Net

Petri Net(PNs) とは，C. A. Petri によって提唱された，システムのグラフィカルなモデル化手法である [2].

2.1　Time-Petri Net

Time-Petri Net(TPNs) とは，上記で示した Petri Net に，時間の概念を付与したものである．時間に関するパラメータの導入の仕方により，大きく 2 つに分類される．

一方はプレースに入力されたトークンが利用可能になるまでの遅延時間を導入する方法であり，プレース時間ペトリネットと呼ばれる．もう一方はトランジションの発火に要する時間を定めるものであり，トランジション時間ペトリネットと呼ばれ，内部では更にいくつかの手法が提案されている．加えて，このプレース時間ペトリネットとトランジション時間ペトリネットの両方を合わせたモデルも考案されている．本論文では発火遅れ時間を持つトランジション時間ペトリネットを Time-Petri Net と表記し，これを用いて進めていく．

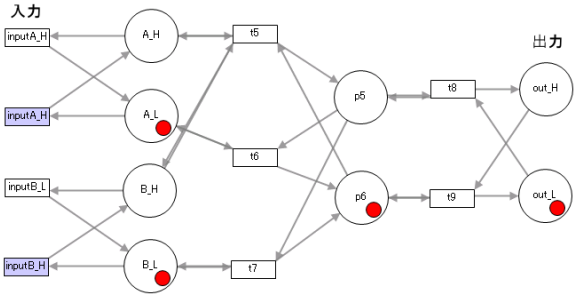

図 1　米田らによる 2 入力論理積ゲートの従来モデル [4]

2.2　Petri Net の階層化

一般に大規模なシステムの設計を行う場合，外部から見た要求仕様（入力と出力の関係，あるいは演算式等）が与えられて，内部の詳細化は設計者に委ねられる場合が多い．Petri Net で対象システムの設計を行う場合，外界とのインターフェース部分が定義されて，内部の機能モジュールは個別に準備あるいは制作し，各々を接続用のトランジション，プレース等を用いて結合する方法が用いられる．Petri Net の階層化手法には様々な方法が提案されている [2]．本稿で提案し，制作ツール [7] で採用した方法は，特にハードウェア回路の合成と接続に適した様式を使用することとした．

Petri Net を階層化するにあたって，下位層の Petri Net に上位層の Petri Net の変化を伝え，また下位層の Petri Net 内の変化を上位層の Petri Net に伝える必要がある．入出力の変化を伝える接続用のトランジション，プレースのことをそれぞれ入力ポート，出力ポートという．

3　従来提案の遅延つき論理ゲートモデル

Petri Net で論理ゲートを表現したものとして，米田らによって AND ゲートなどの論理素子のモデルが提案されている [4]．例えば 2 入力の論理積モデルは，以下のような構成になっている．

このモデルは，以下のような構成になっている．

- 2 入力論理積ゲートは入力 2，出力 1 の論理ゲートであるが，このモデルでは High と Low を分けて状態を表しているため，Petri Net の構成としては 4 入力 2 出力である．
- プレース *A_H*，*A_L*，*B_H*，*B_L* がそれぞれの入力の現在の値を示している．*A_H*，*B_H* にトークンがあ

† 信州大学大学院 工学系研究科, Graduate School of Science and Technology, Shinshu University.

れば 1，*A_L*，*B_L* にトークンがあれば 0 が入力されたことを表す．

- プレース *out_H*，*out_L* がそれぞれの出力の現在の値を示している．*out_H* にトークンがあれば 1，*out_L* にトークンがあれば 0 が出力されたことを表す．
- 初期マーキングの状態でトランジション *inputA_L* に刺激があれば発火し，トークンは *inA_L* から *inA_H* に移動する．これは，入力 A が 0 から 1 になったことを示す．
- 同様に，トランジション *inputB_L* に外部から刺激がある場合，プレース *inB_L* にすでにトークンがあり前の状態も 0 のため，トークンは移動しない．

3.1 階層化接続時の問題点

従来モデルを使って実際にゲート素子を接続し，回路のモデルを作成してみると，素子モデル単体で見ていたときには気づかなかった問題が浮上してきた．

主な問題点として，前回の状態と次の入力が同一の場合に見られ，従来モデルが限定下でのみ正しく振舞うことを前提に設計されているために起きる．入力を受けるトランジションは，前回の状態を保持しているプレースと今回の入力を伝えるプレースによって発火する．しかし，前回の状態と同じ入力が来た場合，すでに前回の状態を保持しているプレースにトークンは無く，トランジションが発火することはできないため，入力を伝えるプレースのトークンがその場に残り続けてしまう．

4 非同期回路の新たなペトリネットモデル

前節で挙げた従来モデルの問題点を踏まえて，新たにゲート回路のモデルを提案する．

4.1 概要

従来モデルは，実際の回路と同じようにゲートを接続した際，入力が以前と変わらなければトークンがその場に残り続けてしまい，結果それが回路として不都合な動きをしてしまうことがわかった．そこで，提案するゲート回路のモデルでは，前回と変わらない入力がされた場合もトークンを評価して発火するトランジションを設けることにした．

更に，入力プレースのマーキングと発火するトランジションによって真理値表を表現し，出力プレースに伝えるようにした．また，入力が変化してトランジションが発火した時に，入力の変更を伝えるプレース *input_changed* を用意し，それを真理値表を表すトランジションの入力プレースのひとつとして設定することで，入力の変化が起こる前のトランジションの発火を防いでいる．

4.2 基本的な論理ゲートの例

実際に作成した論理ゲートのモデルの一部を以下に示す．

4.2.1 否定論理

否定論理 NOT は入力された値を反転させて出力する論理演算である．

入力プレースに設定されたプレース *IN0_H* かプレース *IN0_L* のどちらかに入力トークンが入ると，その入力が前回の値と違うかどうかを判定する．前回の入力値

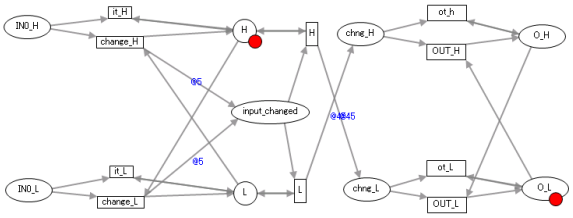

図 2　提案する NOT ゲートのモデル

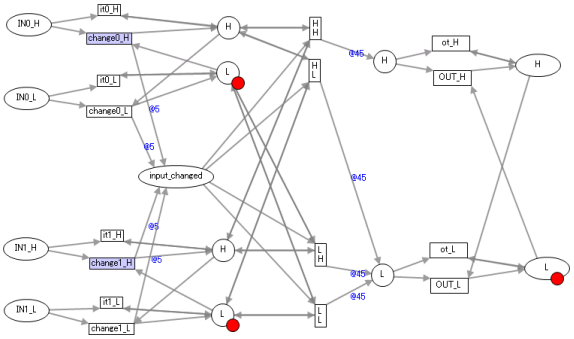

図 3　提案する 2 入力論理積ゲートのモデル

と同じであれば出力を変える必要が無いため，トランジション *it_H* またはトランジション *it_L* が発火し，トークンを消化して終了する．

前回の入力値と異なればトランジション *change_H* またはトランジション *change_L* が発火し，プレース *H* またはプレース *L*，そしてプレース *input_changed* にトークンが出る．トランジション *L* またはトランジション *H* が発火すると，プレース *chng_H* またはプレース *chng_L* にトークンが入り，更に出力ポートに設定されたトランジション *OUT_H* またはトランジション *OUT_L* が発火して出力の変化を伝え，プレース *O_H* またはプレース *O_L* にトークンが出て終了する．

4.2.2 2 入力論理和

入力プレースに設定されたプレース *IN0_H* または *IN0_L*，プレース *IN1_H* または *IN1_L* に入力トークンが入ると，その入力が前回の値と違うかどうかを判定する．前回の入力値と同じであれば出力を変える必要が無いため，トランジション *it0_H* または *it_L*，トランジション *it1_H* または *it1_L* が発火して終了する．

前回の入力値と異なれば，トランジション *change0_H* または *change0_L*，トランジション *change1_H* または *change1_L* が発火し，それぞれ今回の入力値を保持するプレースとプレース *input_changed* にトークンが出る．このとき，プレース *input_changed* には入力の値が異なった分だけトークンが入る．

その後，AND ゲートの真理値表として表されたトランジション *HH*，*HL*，*LH*，*LL* が入力の値によって発火し，トークンが移動する．トランジション *OUT_H* または *OUT_L* が発火して AND2 の出力の変化を伝え，プレース *H* または *L* にトークンが収まって今回の出力の値を保持する．出力が 2 つとも変化した場合は，プレース *input_changed* にトークンが 2 つ入っているはずである．そのため，一度トランジション *OUT_H* または *OUT_L* が発火した後はトランジション *ot_H* または *ot_L* によってトークンが評価される．

また，このほかの論理ゲートについても，入力ポート

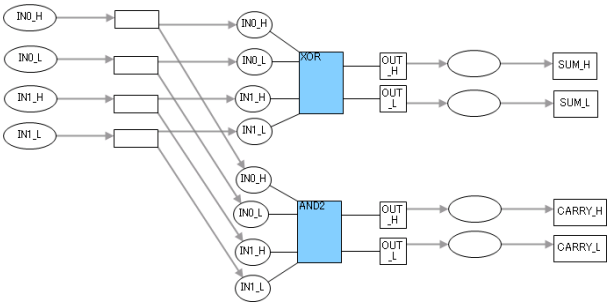

図 4　XOR2 と AND2 を使用した半加算器（HA）

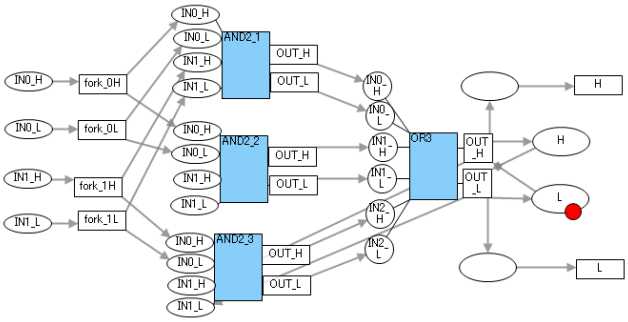

図 5　2 入力論理積 3 つと 3 入力論理和で構成された C-element

の数や真理値表を表すトランジションに変更を加えるだけで容易に作成することが可能となった．

5　設計例

実際に設計したゲート回路を利用して階層的に合成した機能素子について説明する．

5.1　半加算器（HA）

半加算器は，1 ビットと 1 ビットの加算を行う回路である．入力は 2 つ，出力はそのビット出力 (Sum) と桁上がり (Carry) の 2 つである．半加算器を論理ゲートで構成するにはいくつかの方法があるが，ここでは XOR2 と AND2 を各 1 個で構成した．

図 4 は，XOR2 と AND2 を使用した半加算器の Petri Net モデルである．XOR2 と AND2 は階層化されている．入力用プレース *IN0_oH*, *IN0_oL*, *IN1_oH*, *IN1_oL* にトークンが到着すると，入力値を変化させたい刺激であるとみなして，トランジション *fork_IN0_H*，*fork_IN0_L*，*fork_IN1_H*，*fork_IN1_L* にて入力を 2 つに分岐させて階層化された各モジュールへの入力とし，それぞれの演算結果を出力する．

5.2　Muller's C-element

Muller's C-element は非同期回路の構成要素として用いられる，David E. Muller によって考案された論理回路である [5].

2 入力論理積素子を 3 つと 3 入力論理和素子で構成されるこの論理回路は，2 入力 1 出力の論理回路になる．2 つの入力値が同じであればその値を出力し，異なる場合の出力は変化しないといった，データラッチのような振る舞いをする．

図 5 は，2 入力論理積を 3 つと，3 入力論理和 1 つで構成された Muller's C-element である．入力用プレース

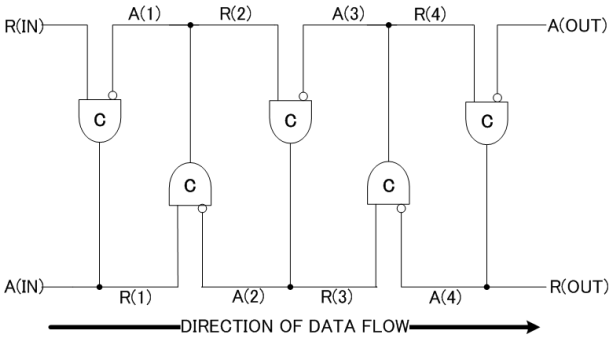

図 6　Micropipeline の回路構成（段数=3）[6]

IN0_H, *IN0_L*, *IN1_H*, *IN1_L* にトークンが入ると，入力が入ってきたとみなす．プレース *IN0_H*, *IN0_L*, *IN1_H*, *IN1_L* への入力は，トランジション *fork_0H*，*fork_0L*，*fork_1H*，*fork_1L* によって 2 つに分岐され，階層化された 2 入力論理積 *AND2_1*，*AND2_2*，*AND2_3* への入力とされる．また，*AND2_2* と *AND2_3* への入力の一部には，出力を入れなければならない．今回，この Muller's C-element のモデルは，最初に一度リセットを掛け，更に出力が *L* であると仮定することにした．つまり，実行するとまず各素子にてリセットが掛かり，初めの一回は発火が保障されている各トランジションが発火する．3 入力論理和はリセットすることによってトランジション *OUT_L* が発火し，*AND2_2* と *AND2_3* への入力になる．

2 入力論理積での演算結果を入力として，3 入力論理和が演算を行い，その出力結果が Muller's C-element の出力となる．

5.3　Micropipeline

5.3.1　概要

Micropipeline は，1989 年に Ivan E. Sutherland によって考案された FIFO 動作を行う非同期回路のモデルであり，片方の入力を反転した 2 入力の Muller's Celement を複数個連結して構成される [6]．図 6 に，Micropipeline の回路構成（段数=3）を示す．回路中には非同期ループが多数存在しており，通常の状態遷移による解析は容易でない．

Micropipeline は，片側が否定論理入力であるような Muller's C-element を入力と出力が互いにループ状になるように多段連結することで構成される．つまり，Micropipeline は，Muller's C-element を連結した段数分だけデータラッチを有するような FIFO バッファとして振舞う．

パイプライン動作のモードと概要を以下に述べる：(1)C-element の出力が交互に High ならば FIFO バッファは FULL であり，全ての C-element の出力が Low ならば FIFO バッファは EMPTY である．(2) 図 6 に示す入力線 R(IN) からのポジティブエッジトリガ（L → H）の信号変化を一度受けると，図中向かって左側の（入力線 R(IN) に近い側の）C-element から逐次トリガ刺激を伝達し，最後には一番右の C-element の出力が H になる．この状態が，FIFO バッファに有効データを一個格納した状態である．(3) 入力線 A(OUT) からのポジティブエッジトリガ（L → H）の信号変化を受けると，R(IN) 動作とは逆方向に向かって C-element 間のトリガ刺激伝達が進行し，最も古く格納したデータを読み出して（無

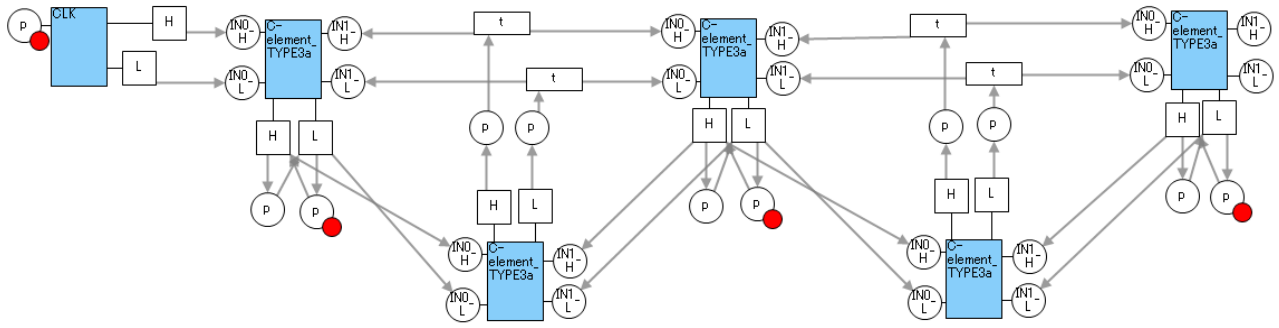

図 7　Micropipeline で構成された FIFO のネットモデル（段数=3）

効データとして），その後に続く有効データを待ち行列の先頭方向へ押し出す．

5.3.2　作成した FIFO の階層化ペトリネットモデル

図 7 に，5 個の C-element を用いて Micropipeline で構成された FIFO のネットモデル（段数=3）を示す．モデル全体のデザインエントリと合成は，HiPS ペトリネットツールによって行った [7].

図 7 に示したモデルにおいて，階層化されている Muller's C-element は，下位設計時，既に片方の入力は否定論理として作成してある．また，今回のモデルでは，初期状態を全て出力が Low の状態，つまり待ち行列内の有効データ数がゼロ（EMPTY）の状態で開始することを前提とし，作成した．そのため，下層の Muller's C-element で初期マーキングを設定し，モデルの駆動時にトークンが評価されることによって，全ての出力が Low の状態で開始できるよう設計した．一般には，有効データ数がゼロ以外のケースで駆動を開始する場面も想定されるが，その場合においても，初期マーキングをケース毎に自由に設計ツール上で設定することによって，途中の場面からの駆動再開や，一時的にデータラッチの値が異常になった場合からの回復過程などを，容易に解析することが可能となった．

6　まとめと今後の課題

Time-Petri Net を用いて各種の論理ゲート（プリミティブ）をモデル化し，各素子を階層的に接続して半加算器，全加算器，Muller's C-element などの機能演算素子を作成した．次に，作成した Muller's C-element を用いた非同期ループ接続によって Micropipeline に基づいた FIFO モデルを構成した．

本稿で提案した階層化 Petri Net の手法については，作成された既存の論理ゲートモデルの問題点を明らかにし，改良を行うことで進めた．既存の論理ゲートモデルは前回の入力の値と今回の入力の値が同じである場合にトランジションが発火せず，トークンが消費されないままその場に残り続けてしまう問題があった．新たに提案する論理ゲートモデルは，その問題を回避するべく入力に変化があったかどうかを確認し，変化が無ければトークンを消費してしまうトランジションを設けることで，この問題を回避した．

本提案のペトリネットによる階層化手法が，実際の非同期ループを有する組み合わせ論理回路の階層合成法として有意であることを確認した．更に，各素子の設計と合成ならびに動作の確認は，ネット作成援用ツール "HiPS" によって行い，記述能力の評価を行った．

今後の課題として，これまで作成した論理ゲートや組み合わせ論理回路を用いて，規模の比較的大きな演算回路やシーケンサを作成し，その要求仕様の検証等を HiPS ツール上の解析器で実施することが挙げられる．その目的のため，援用ツール HiPS への各種ネット解析アルゴリズム（deadlock 解析，有界性解析，インバリアント解析など）の実装を行う．ツール上でのインタラクティブ・シミュレーションやタイムライン解析を向上させるため，入出力インターフェースを中心とした種々のリアクティブモデルとしての要素を追加する．他方，今回作成した論理ゲートや，これから作成していく論理回路のモデルを，HiPS 上で使えるライブラリとしていくことも視野に入れる．加えて，上流・下流の各種ツールや言語処理系（例：LOTOS，VHDL など）との接続も考えていきたい．

参考文献

[1] Janusz A. Brzozowski, Carl-Johan H. Seger : "Asynchronous Circuits", Springer-Verlag, ISBN 0-387-94420-6 (1995).

[2] Tadao. Murata : "Petri Nets : Properties, Analysis and Applications", Proc.IEEE, Vol.77, No.4, pp.541-580 (1989).

[3] Kurt Jensen : "Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use, Vol.1, Basic Concepts, 2nd ed.", Springer-Verlag (1996).

[4] Tomohiro Yoneda, Bin Zhou, Bernd-Holger Schlingloff : "Verification of Bounded Delay Asynchronous Circuits with Timed Traces", Proc. of Int'l Conf. on Algebraic Methodology and Software Technology, Springer, LNCS-1548, pp.59-73 (1999).

[5] David E. Muller and W S. Bartky, "A Theory of Asynchronous Circuits", Proc. Int'l Symp. Theory of Switching, Part 1, Harvard Univ. Press, pp.204-243(1959).

[6] Ivan E. Sutherland : "MICROPIPELINES", Communications of the ACM volume32, pp720-728 (1989).

[7] 野村達雄 : "階層化可能なペトリネットシミュレータ (HiPS) の開発", 信州大学工学部卒業論文 (2009).