排他的な処理を含むSTGに対するCSC Conflictの因果関係追加による解消手法について

赤池 大介* 桑子 雅史** 新家 稔央** 横山 孝典**

*東京都市大学 大学院工学研究科 情報工学専攻 **東京都市大学 知識工学部 情報科学科‡

1. はじめに

近年，同期式回路は高集積化・高速化している．しかし，それにともない消費電力の増大が問題になっている．そこで，クロック信号を使わないため消費電力が少ない非同期式回路が注目されている．非同期式制御回路の制御仕様を表記する手法としてSTG(Signal Transition Graph)がある．STGは，制御信号の遷移の因果関係を有向グラフで表現したものであり，設計者はSTGを記述し，それを元に非同期式制御回路を設計する[1]．

回路を生成できるためにSTGが満たさなければならない必要十分条件として，CSC(Complete State Coding)条件[2]が知られている．設計者が記述したSTGがこの条件を満たしていない場合，回路を生成するためにはSTGに何らかの追加仕様を加えてCSC conflictを解消する必要がある．従来研究[3]では，排他的な処理が含まれないSTGに対象を限定しているが，一般に制御回路では条件分岐が頻繁に現れ，その箇所は排他的な処理となっている．従って，そのようなSTGに対する手法が求められており，本研究では，従来の手法を改良した排他的な処理が含まれるSTGにも対応できる手法を提案する．

2. STGとCSC条件

STGの一例を図1に示す．STGでは，制御信号xの0→1遷移をx+と表し，1→0遷移をx-と表す．遷移間の因果関係を有向枝で表す．回路が現在どの状態にあるのかは，有向枝上にトークンを置くことで表す．トークンは枝の上には一つのみである．すなわち，STGにおける枝は状態に対応している．信号が遷移するためには，信号遷移の入力枝すべてにトークンが置かれている必要がある．STGにおける排他処理への分岐および排他処理からの合流はプレース（○）で表す．プレースへの入力遷移が複数ある場合には，それらの遷移の一つが発火することでプレースにトークンが乗る．プレースからの出力遷移が複数ある場合には，プレースにトークンが乗ると，それら出力遷移のうち，いずれか一つが発火する．図1の場合E1にあるトークンがA+の発火によりP1に移動する．次にB+かC+のうちいずれか一方が発火し，トークンが発火した遷移の出力枝へ移動し，P2に到達しG-が発火する．

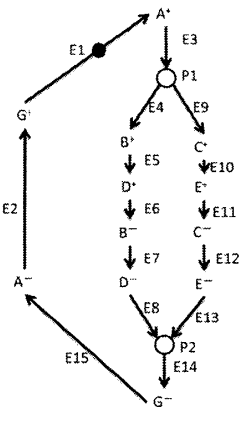

図1：STGの例

STGのある一状態の状態符号は，その状態においてトークンが置かれている枝全部の状態符号の合成となる．例えば，図2の枝E1の状態符号を考える．E1にトークンが置かれた状態で他のトークンがどのように移動するかを考慮すると，制御信号req1は1であり，req1以外の制御信号は不定である．そのため，E1の状態符号は(1***)である．同様にして，E2からE9の状態符号も図2に示したように定まる．図2において，「○」で表したトークン配置である状態S1の状態符号は(1*1*)+(1***)+(*1*1) = (1111)である．

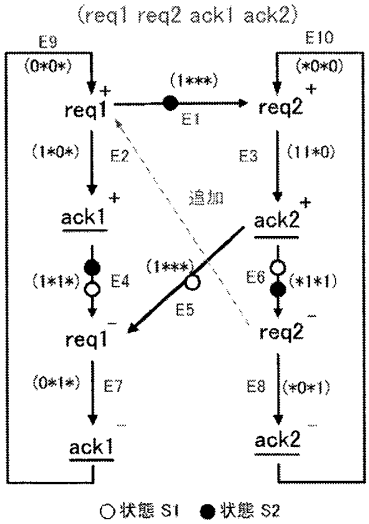

図2：CSC conflictを起こしている STGの例

STGから回路を生成するための条件であるCSC条件は，

- すべての到達可能状態が異なる2進コードを持つ
- 2つ以上の状態が同一の2進コードを持つ場合，それらの状態において遷移可能な内部信号と出力信号は等しい

のどちらかを満たすことである．CSC条件を満たしていないSTGは「CSC conflictが生じている」と言われる．設計者が記述したSTGはCSC conflictが生じていることが多い．

図2において，「●」で表したトークン配置である状態S2の状態符号は（1111)である．S1とS2は異なる状態であるのに同じ状態符号となっており，このSTGはCSC conflictを起こしている．

CSC conflictを解消する方法として，制御信号を追加する方法[4]と，因果関係を追加する方法[3]がある．このうち，因果関係を追加する方法は，STGから生成された回路の回路量が少なくなる傾向にある．

3. 従来手法と問題点

因果関係の追加によってCSC conflictを解消するための大まかな手順を以下に示す。

1. CSC conflictを起こしている状態を特定する
   STGのすべての状態を列挙し状態符号が重複しているか調べる．
2. STG中で因果関係を追加可能な箇所をすべて列挙する
   STGは非同期式制御回路の設計仕様を表記したものなので，因果関係を追加する際，有向枝の終点は出力

A method for solving of CSC conflict in designing of asynchronous controller from STGs including exclusive operation

†Daisuke Akaike

‡Masashi Kuwako, Niinomi Toshihiro, Takanori Yokoyama

†Graduate School of Engineering,
Tokyo City University Graduate Division

‡Department of Computer Science,
Faculty of knowledge Engineering, Tokyo City University

信号か内部信号の遷移に限られる．

3. 因果関係を追加可能な箇所のうち、CSC conflict を解消できる組み合わせを列挙し追加する
　因果関係の追加による既存の枝に振られた状態符号への影響は，論理値が 0 または 1 に確定している信号の論理値が反転することはない．不定値となっていた信号の論理値が 0 または 1 に確定するのみである．すなわち，CSC conflict を起こしている状態を S，S’とすると，因果関係の追加によって CSC conflict を解消する場合，S，S’を構成する枝の状態符号中の不定値が確定することにより，S または S’が消滅することによって解消する．このことより，因果関係の追加は，S(または S’)を構成する枝のどれかの状態符号が，S(または S’)の状態符号と相反を起こすように行う．図 2 の場合 req2-から req1+への因果関係を追加することにより E1 の状態符号が(10**)となり状態 S2 の状態符号と相反し S2 が消滅する．

従来手法は排他的な処理を含まない STG に対象を限定している．排他的な処理を含む STG にこの手法をそのまま適用すると，誤った STG となる危険がある． 図 3 の STG において，G+から D+へ因果関係 E1 を追加した仮定する．

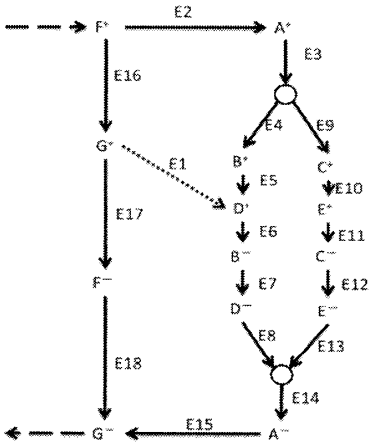

図 3:排他的な処理を含む STG への因果関係の追加

F+が発火することにより A+，G+が発火し，B+または C+が発火する．B+が発火した場合，E1 と E5 にトークンが乗るので，D+が発火し，続いて B−，D−，A−が発火する．さらに G+の発火により F−が発火，E18，E15 にトークンが乗り G−が発火する．
一方、A+，G+が発火し， C+が発火する場合，C+，E+，C−，E−，A−と発火する．G+の発火により F−が発火し，G−が発火する．
この場合，E1 にはトークンが残っている．すると，次回の排他的な処理で，B+が発火した場合，すでに E1 にトークンが乗っているので，G+が発火しなくとも D+が発火してしまい誤った動作となる．また，C+が発火した場合 E1 に複数のトークンが乗ってしまい STG のルールとしても正しくない．

4. 提案手法

問題点を解消するためには，排他的な処理との間に因果関係を追加する際には，相補的な処理からも併せて因果関係を追加することにより STG のルールに反しない追加が可能である．

図 4 の STG において G+から D+へ因果関係を追加したいとする．
まずプレース P5 を追加する．G+から P5，P5 から D+への因果関係を追加する．次に，G+の相補的な処理である I+，J+，I−，J−のうちいずれかから，P5 へ因果関係を追加する．図 4 では I+から P5 へ追加している．さらに，D+の相補的な処理である C+，E+，C−，E−のうちから従来手法手順 2 の条件に従って P5 からの因果関係を追加する．
図 4 では P5 から E−へ因果関係を追加している．

この追加処理を行った STG の動作は，F+，A+が発火し P1，P3 へトークンが移動する．I+，G+のうちいずれかが発火する．並行して B+，C+のうちいずれかが発火する．I+，G+のいずれかが発火しても，P5 にトークンが移動するので，B+，C+のどちらが発火したとしても P5 のトークンにより D+もしくは E−が発火することができる．

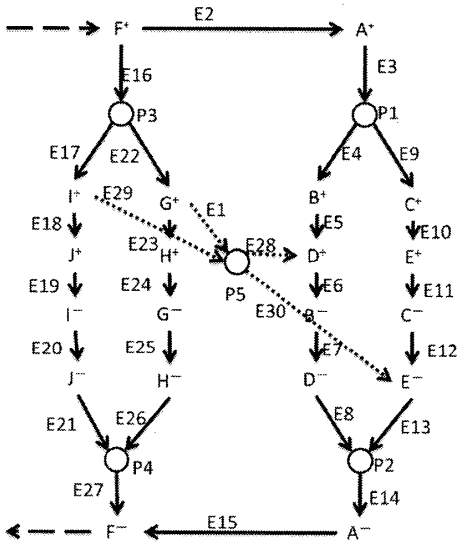

図 4: 提案手法

5. まとめ

排他的な処理を含む STG に因果関係を追加することによって CSC conflict を解消する手法を提案した．排他的な処理以外にその処理と相補的な処理にも因果関係を追加することにより STG のルールに反することなく因果関係の追加を行うことができる．本研究では CSC conflict を解消した STG から生成される回路の性能を考慮していないので，回路性能を考慮した手法が今後の課題である．

参考文献

[1] T. A. Chu, “Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications”, Ph. D. Thesis, Massachusetts Institute of Technology, 1987
[2] C. W. Moon, “Synthesis and Verification of Asynchronous Circuits from Graphical Specifications”, Ph. D. Thesis, Univ. of California at Berkeley, 1992
[3] 木村 威暁, “STG からの非同期式制御回路設計における CSC conflict の一解決手法 ”, 情報処理学会, 第 70 回全国大会講演論文集, 2P-7, 2008
[4] J. Cortadella, M. Kishinevsky, A. Kondratyev, L.Lavagno, A. Yakovelv, “Petrify : A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers”, IEICE TRANS.INF. & SYST, VOL. E80-D, No3, 1997