# 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

木村　威暁*　　桑子　雅史**　　持木　幸一**

*武蔵工業大学 大学院工学研究科　電気工学専攻　**武蔵工業大学 知識工学部 情報科学科‡

## 1. はじめに

現在、デジタルシステムにおける消費電力の増大などの問題を解消する手法として非同期式論理設計が注目されている[1]。

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。

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。本研究では、STG に対して因果関係を追加することによって CSC conflict を効率的に解消する一手法を提案する。

## 2. STG と CSC 条件

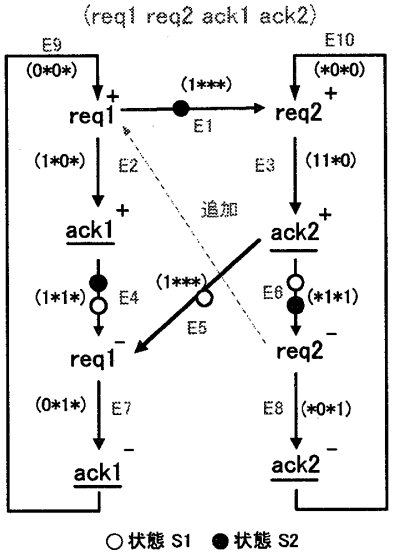


図 1: CSC conflict を起こしている STG の例

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。制御回路への入力信号は、信号名に下線を付す。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。すなわち、STG における枝は状態に対応している。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 1 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 1 に示したように定まる。図 1 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。

STG から回路を生成するための条件である CSC 条件は、

- 全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 1 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 2 つの状態は 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と、因果関係を追加する方法がある。このうち、制御信号を追加する方法[3]は、既に提案されている。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が、その手法は明確には提案されていない。

## 3. 因果関係の追加による CSC conflict を解消する手法

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として、以下を提案する。

1. CSC conflict を起こしている状態を特定する
2. STG 中で因果関係を追加可能な箇所を全て列挙する
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する
4. 3. の組み合わせのうち、効率的に CSC conflict を解消できるものを求める

以上のうち、2. から 4. の詳細を以下に説明する。

---

A method for solving of CSC conflict in designing of asynchronous controller from STGs
†Takaaki Kimura
‡Masashi Kuwako, Kouichi Motiki
†Graduate School of Research Division in Engineering, Musashi Institute of Technology
‡Faculty of knowledge Engineering, Musashi Institute of Technology

### 3.1 STG 中で因果関係を追加可能な箇所の列挙

因果関係を追加する意味のある箇所は、並行して発火する遷移間である。更に、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信号か内部信号の遷移に限られる。図 1 の STG の入力信号は ack1, ack2、出力信号は req1, req2 である。従って、図 1 の STG に因果関係を追加できる箇所は、req1+ → req2-, ack1+ → req2+(-), req1- → req2-, ack1- → req2-, req2- → req1+(-), ack2- → req1+(-)の計 9 箇所となる。

### 3.2 CSC conflict を解消できる組み合わせの列挙

因果関係の追加によって、既存の枝に振られた状態符号がどのような影響を受けるかを考えると、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 に確定する可能性のみである。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 または S' の状態符号が変わり、S と S' の状態符号が一致しなくなり解消することはありえないと言える。因果関係の追加によって 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される可能性しかない。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えば良い。

CSC conflict を起こしている両方の状態に含まれる枝の状態符号を確定させても CSC conflict を起こす状態が先ほどとは異なる箇所に移動するだけで、根本的な解消はできない。従って、S または S' のどちらかにのみ含まれる枝の状態符号を確定させる。

図 1 の STG における S1, S2 を例に考えると、S1, S2 を構成している枝のうち、一方の状態にのみ含まれる枝は E1, E5 である。S1, S2 の状態符号は(1111)である。E1, E5 の状態符号は(1***)である。従って、E1 または E5 のどちらかの状態符号において req2, ack1, ack2 のいずれかの論理値を 0 に確定させるような因果関係の追加をすれば S1, S2 間の CSC conflict が解消する。

遷移$x^t$ (x:信号, t:{+,-})、$x^t$と逆方向の遷移を$x^{\bar{t}}$ とする。枝Eにより状態符号の相反を引き起こすとする。できるだけ少ない因果関係の追加によりCSC conflictを解消するために、$x^t$とEが順序関係を持ち、$x^{\bar{t}}$ とEは順序関係を持たないxに着目する。

確定させたい論理値がx=1 であり、$\bar{t}$= '+' とすると、遷移$x^{\bar{t}}$の下流の遷移から、枝Eの始点より上流の遷移へ向けて因果関係を追加すればよい。確定させたい論理値がx=1 であり、t= '-' とすると、枝Eの終点より下流の遷移から、遷移$x^{\bar{t}}$より上流の遷移へ向けて因果関係を追加すればよい。確定させたい論理値がx=0 である場合には '+' と '-' が以上とは逆になる。このとき、

1. $x^{\bar{t}}$の上流(下流)とは、Eの終点(始点)と並行動作をしている範囲とする。Eの上流・下流とは、$x^{\bar{t}}$と並行動作をしている範囲とする
2. 初期トークン(回路の初期状態に対応するトークン)が置かれている枝を越えない範囲とする

の両方を満たしていることとする。1. は、因果関係を追加して意味があるのは並行動作が行われている箇所であるためである。2. は、初期トークンを越えた遷移と因果関係をつけるということは、1 周期前(後)の遷移と因果関係をつけることになり、意味がないためである。

図 1 の枝 E1 は、遷移 req2+、ack2+とは順序関係を持っており、req2-、ack2-とは順序関係をもっていない。そこで、E1 の状態符号のうち、req2 の論理値が 0 に確定するように因果関係を追加する。そのためには、req2-より下流にある遷移(req2-, ack2-)、E1 の始点である req1+より上流の遷移(req1+, req1-)に向けて、因果関係の追加を行えば良い。

### 3.3 効率的に CSC conflict を解消できる因果関係の追加箇所を求める

グラフ表現における動作の並列性をできるだけ高く保つことを考えた場合、最も早く発火する遷移から、最も遅く発火する遷移に向けての因果関係追加であれば、速度低下は最少になると考えられる。

図 1 の場合、req2-と ack2-のうち最も上流であるのは req2-である。req1+と req1-のうち最も下流であるのは req+である。従って、点線で示したように req2- → req1+と因果関係の追加を行う。その結果、req2-と req1+の両方が起こらない限り、枝 E1 にトークンを置かれることは無い。そのため、E1 の状態符号のうち、req2 の論理値が確定する。すると E1 の状態符号が(10**)となり、E6 の状態符号(*1*1)と相反するようになるため、E1, E4, E6 の 3 つの枝に同時にトークンが配置された状態 S2 は存在しなくなる。よって、S1 と S2 の間で起こっていた CSC conflict が解消される。

## 4. まとめ

STG に対して因果関係を追加することによって CSC conflict を解消する一手法を提案した。因果関係を追加する箇所は、CSC conflict を起こしている状態の一方にのみ含まれる枝において、状態符号の相反を引き起こすようなものを選択する。

CSC conflict は因果関係の追加だけでは解消できない場合があり、実際には従来手法である制御信号追加による解消方法を併用する必要がある。因果関係の追加と制御信号の追加を融合させた解消方法を考えることは今後の課題である。

## 参考文献


[1] 南谷 崇 「非同期式マイクロプロセッサの動向」, 情報処理, Vol. 39, No. 3, pp. 181～186, 情報処理学会 (1998)

[2] Moon, C. W., "Synthesis and Verification of Asynchronous Circuits from Graphical Specifications", PhD Thesis, Univ. of California at Berkeley (1992)

[3] Petrify : A Tool for Manipulating Concurrent Specification and Syshtesis of Asynchronous Controllers, IEICE TRANS. INF. & SYST, VOL. E80 -D, No. 3, (1997)