第11回

論理回路のプロパティ検証技術（3）

――プロパティ検証の実際

藤田昌宏

リファレンスとなるプロパティ（仕様）を正しく表現することは，実はそれほど容易ではない．今回は，誤ったプロパティの例を示しながら，仕様のあいまいさの問題について説明する．また，時相論理式から状態遷移表現を生成するフリーのツールを紹介する．時相論理式を自分で作成し，対応する状態遷移表現（Verilog HDL記述）を生成して，それを調べるという作業を何度か繰り返すと，時相論理でかなり自由に仕様が書けるようになるという．（編集部）

前回（本誌2004年10月号，pp.136-141）までに，プロパティ検証（property checking）の基礎として，プロパティの表現法やその状態遷移表現への変換について考え，また，検証の例も示しました．その中で，プロパティ検証では，検証しようとしているプロパティそのものを正しく表現することが，実はそれほど簡単ではないということについて説明しました．

今回は，このプロパティを正しく表現するために注意するべきことや，利用できる関連技術について説明していきたいと思います．とくに，時相論理でプロパティを記述するうえで重要な概念であるeventualityについて，詳しく説明していきます．

● 記述したプロパティの正しさをどう確認するのか？

まず，前回の終盤の説明について復習したいと思います．いま，1ビット・カウンタとして，入力が2回‘1’になると，出力が1回‘1’になるという仕様のカウンタ（ただし，後述するようにこの仕様そのものが「あいまい」なのだが…）を

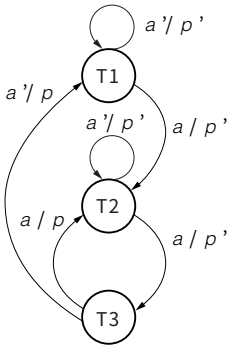

図1　1ビット・カウンタの設計例

入力信号*a*が2回‘1’になると，出力信号*p*が‘1’になる．内部でフリップフロップを一つ使って信号*a*が‘1’になった回数を数えるということで，ここでは1ビット・カウンタと呼ぶ．

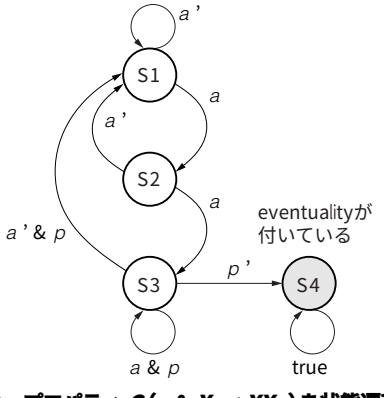

図2　プロパティ *G*（*a* & *Xa*→*XXp*）を状態遷移に展開

1ビット・カウンタの設計に対する仕様として，「今，信号*a*が‘1’で，かつ，次のクロックでも‘1’であると，その次のクロックで信号*p*が‘1’になる」を考える．時相論理の記述は *G*（*a* & *Xa*→*XXp*）となる．また，それを状態遷移に展開すると図のようになる．

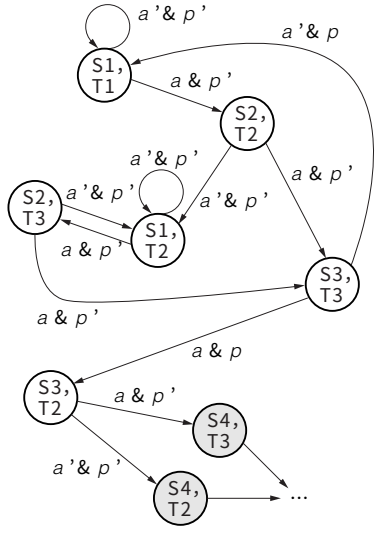

図3　図1とプロパティ *G*（*a* & *Xa*→*XXp*）の否定（図2）の積に対応する状態遷移図

この図には状態S4が現れている．図1の設計は，1ビット・カウンタとして正しいように見えるが，実際にはプロパティを満たしていないことになる．実はこれは，カウンタの動作の詳細をどう考えるかという問題であり，検証に利用しているプロパティがかならずしもそれに沿ったものではないことが原因である．

設計することを考えます．この仕様に対する設計例を**図1**に示します．

図1の動作を初期状態から追ってみると，設計に誤りはないように思えます．合計2回，入力aの値が‘1’になる（**図1**ではaで表現している．aが‘0’であることは，a’で表現している）と，出力pが‘1’になっています．これに対して，プロパティとして，前回と同じように時相論理式（$G(a \,\&\, Xa \rightarrow XXp)$）を考えます．これは，信号$a$が2回連続して‘1’になると，その次の時刻に信号$p$が‘1’にならなければならないということを示しており，満たされなければならないプロパティであると言えます．さて，検証には，まずこの式の否定，

$$(G(a \,\&\, Xa \rightarrow XXp))' = F(a \,\&\, Xa \rightarrow XXp)'$$

を計算し，それを状態遷移に展開します．すると**図2**が得られます．そして，**図1**と**図2**の状態遷移の積を計算して，**図3**が得られます．**図2**において状態S4に到達すると，検証しているプロパティの否定が満たされたことになります．

状態S4にはeventualityが付いており，時相論理式を満たすには，かならずこの状態にたどり着く必要があることを示しています．このeventualityについては，後で詳しく説明します．

さて，**図3**を見てください．**図1**の設計が正しい場合，**図2**が満たされてはならないので，状態S4は結果の状態遷移図である**図3**には決して現れないはずです．しかし，現実には現れています．つまり，仕様の否定が満たされているということであり，設計は仕様を満たしていないことになります．しかし，直感的には，**図1**の設計は正しいように見えます．また，「信号aが2回連続して‘1’になると，その次の時刻に信号pが‘1’になる」というプロパティも正しいように思えます．これは，いったいどういうことなのでしょう？

図1の設計は，その状態遷移をたどってみても1ビット・カウンタとして正しいように見えますが，**図3**では状態S4が現れており，実際に反例（命題が正しくないことを示す例）を生成することもできます．実は，これはカウンタの動作の詳細をどう考えるかという問題であり，検証しようとしているプロパティがかならずしもそれに沿ったものではないことが原因です．

例えば，信号aが連続して3回‘1’になった場合，1ビット・カウンタはどのように動作するべきでしょうか？**図1**のカウンタでは，出力pを1回‘1’にするだけです．しかし，今考えているプロパティでは，信号aが2回連続して‘1’になると，出力pを次の時刻に‘1’にすることになっています．信号aが3回連続して‘1’になると，2回連続して‘1’になることが2回起こったことにもなるため，信号pが2回連続して‘1’にならないといけないことになっているのです．これが正しいプロパティかそうでないかは，1ビット・カウンタの動作の詳細をどう考えるかによります．

なお，これは，プロパティ検証の結果，「設計が正しくない」となった場合に生成される反例の一つであり，一般に，こうした検討が反例を解析することに相当します．

実際，1ビット・カウンタの仕様を「入力が2回‘1’になると，出力が1回‘1’になる」と日本語で述べた時点で，実は不明瞭なことを言っているのです．すなわち，入力が2回‘1’になって，その次の時刻に出力を‘1’にするときの入力の値もカウントするのかしないのか，明確ではありません．あるいは，上述の例のように入力が続けて3回‘1’になったとき，2回‘1’になったことが2回起きたと見るのか，そう見ないのかなどは，どちらとも解釈できます．通常のカウンタの概念では，3回続けて‘1’になっても，最初の2回でカウンタが出力を‘1’にしているので，入力が2回‘1’になったことが2回起きたとは考えないのが普通です．そのように考えると，検証しようとしていたプロパティがまちがっていたことになります．

● 自然言語やタイミング図の記述にはあいまいさがある

しかし，ここでよく考えてください．今の例では，「時相論理で仕様を書いたので，まちがったプロパティを記述した」とも言えますが，逆に言えば，「もとの日本語で表現した仕様があいまいで，厳密な仕様になっていない」とも考えられます．つまり，時相論理のようなあいまい性のない表現法を使って記述しないと，仕様を厳密に定義できないとも言えます．

また，日本語だけで仕様を記述しているからいけないのであって，タイミング・チャートなどの図形表現を併用して示せばよいと思われるかもしれません．しかし実際には，タイミング・チャートで表現しても，状況はそれほど改善されません．1ビット・カウンタの仕様として，単純に**図4**のようなタイミング・チャートで表現しても，もとの日本語以上の情報はありません．入力が3回続けて‘1’になった場合の動作については，何も触れられていないからです．