システムLSI 検証技術入門

デバイスの記事

第16回

形式的検証ツールの効果的な利用法

――システム・レベル記述どうしの等価性を検証する

藤田昌宏

ここでは，商用の等価性検証ツールを用いて，システム・レベル記述（SystemC）どうしの等価性を判定する．実際の等価性検証では，初期化や記述の読み込みなどの前処理の時間が必要になる．また，ワード・レベル検証を設計全体に適用できるかどうかによっても処理時間が大幅に変わる．　（編集部）

今回と次回は，C言語やRTLのハードウェア記述言語で表現された設計記述どうしの等価性を，商用の等価性検証ツールを使って判定してみます．対象としては，アルゴリズムの表現として記述されたSystemC記述と，同記述に対する計算式の簡単化，並列化，およびパイプライン化を人手で行った設計の間の等価性を検証していきます．

● 順序回路検証におけるレイテンシとスループットを指定

前回も説明したように，順序回路では，フリップフロップなどで実現された内部状態が存在します．出力値は，入力値とその内部状態の値で決定されます．入力が入ってから内部状態に依存した計算を行い，一定時刻後に出力が出てくることになります．また，ハードウェアは，「入力から出力を計算するという処理を無限に繰り返す」というように動作するのが一般的です（リアクティブ・システム）．したがって，二つの順序回路が等価であるか否かを調べるためには，「どのタイミングの入力に対する出力がどのタイミングで出てくるか」という情報を設計者がツールに与えてやる必要があります．通常，等価性検証ツールでは，これらの情報はレイテンシとスループットとして与えます．

前回も示しましたが，レイテンシとスループットは**図1**のようなタイミングの指定に対応しています．スループットは，ある入力が入ってから次の入力が入るまでの間隔をクロック数で指定します．レイテンシは，一つの入力が入り，その入力に対応する出力が出てくるまでの時間であり，これもクロック数で指定します．一つの入力が入ってから，レイテンシ分だけ経過した時刻で出力が等価か否かを調べることになります．

等価性検証では，二つの設計記述の等価性を検証します．一般に，片方は正しい（あるいは，正しいと考えている）設計であり，これは「仕様」となります．これに対して，もう一方の設計が仕様を実現する「実装」であると考えられます．つまり，仕様と等価であることを示すことで，実装の正しさを証明するのです．上のレイテンシとスループットは仕様と実装のそれぞれに対して指定する必要があります．

仕様として，アルゴリズムを表現するために通常のソフ

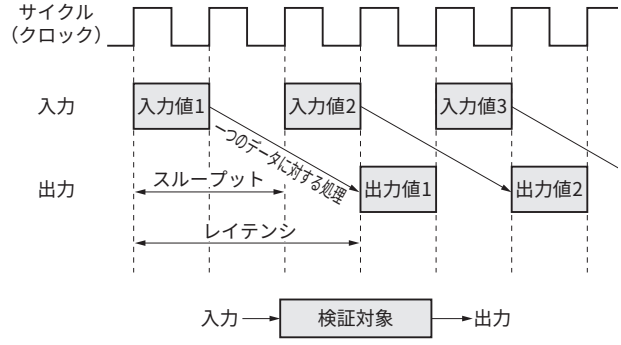

図1　レイテンシとスループット
一つのデータに対する計算時間が「レイテンシ」，入力データ間隔が「スループット」である

KeyWord　等価性検証，システム・レベル記述，SystemC，レイテンシ，スループット，IDCT，SLEC，ビヘイビア合成，パイプライン，リタイミング

トウェアと同じように記述されたものを利用する場合，すべての計算は瞬時に終了する形となり，1時刻で計算が終了し，かつ各入力は順に処理されていくことになります．つまり，レイテンシもスループットも1となります．しかし，より複雑な動作を表現し，部分的な動作の間の時間順序関係があるような記述を仕様とする場合には，実装だけでなく仕様についても，**図1**に示すレイテンシとスループットを指定する必要があります．

SystemCを利用するビヘイビア記述や，より上位のシステム・レベル設計記述では，かならずしもクロックという概念はありません．そのような場合，時刻を進める文としてwait()を利用します．wait()文があると，時刻が一つ進むことを意味し，そのwait()文の前の記述と後の記述では別の時刻の動作を表していることになります．したがって，**図2**に示すようにレイテンシが3であるSystemC記述には，実行のどのパスをたどっても，wait()文が2回実行されるようになっているはずです．

等価性検証では，仕様はこれまで正しいと考えられている設計なら何でもよいわけですから，一度等価であると判定された実装は，次の等価性検証では仕様として利用できます．つまり，設計記述を次々に改良し，詳細化していくような設計手法では，**図3**に示すように設計の各ステップの間で次々に等価性を検証していくことで，最終的な設計といちばん最初の仕様が等価であることを示せます．**図3**の上側に示す一連の等価性検証がすべて等価であると判定されれば，最初の仕様記述と最後の設計(実装)記述Nは等価であると言えます．

もちろん，**図3**の下側に示す最初と最後の記述の間の等価性検証を行ってもかまいません．しかし一般に等価性検証では，比較している二つの設計記述の差が小さいほど効率良く検証できます．例えば二つの記述はほぼ同じで，一つの記述ブロックに対してのみ，数式の最適化処理などが施されているだけの差の場合，差のある記述ブロックだけをチェックすればよいので，高速に検証できます．一般に記述の差が少ない場合には，多くの内部のフリップフロップが二つの設計データの間で同じ(あるいは対応する)状態を表していることが多く，その対応がとれる部分については，順序回路ではなく組み合わせ回路の検証手法を適用できます．そのため，等価性検証を大幅に効率化できます．

以下では，米国Calypto Design Systems社の「SLEC」と呼ばれる等価性検証ツールを利用しながら，SystemCによるシステム・レベル設計記述どうしの等価性検証，ならびにリタイミングなどのRTLの設計最適化に関係した等価性検証について見ていきたいと思います．

● システム・レベル記述どうしの等価性を検証

ここでは簡単でわかりやすい例として，画像処理などでよく使われるIDCT(逆離散コサイン変換)を考えます．典型的なソフトウェアとしてのプログラムをほぼそのままSystemC記述で表現したものを**リスト1**に示します(この名まえをIDCT0とする)．これを仕様(出発点)として，個々の算術式レベルの最適化や並列動作の導入，さらにはパイプライン化などを行った記述との等価性検証を実際に行いたいと思います．**リスト1**の記述では，関数idctrowとidctcolが各8回ずつ，異なる引き数値で順に実行されています．関数idctrowとidctcolはそれぞれ数十行の算術式から構成されています．SystemC記述にするために，SC_METHODなどのSCで始まるSystemC用の関数で全体が表現されています．この記述ではwait()文はなく，すべての実行が瞬時に終了するという形になっています．このように1回だけの処理をすべて瞬時に行うような場合，

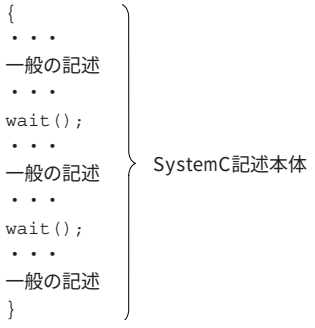

図2　SystemCの場合，時刻が進むことはwait()文で指定する
レイテンシが3の記述．どのパスをたどっても，wait()文が2回実行されるようになっている

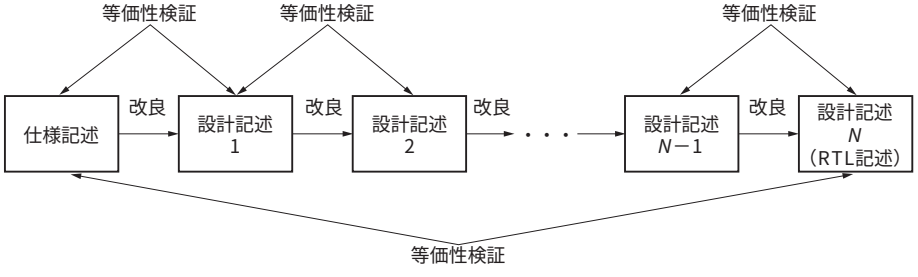

図3　設計の繰り返し改良・詳細化と等価性検証
実際の設計では，設計記述を次々に改良し，詳細化していくことが多い