第1章

C言語によるハードウェア設計を理解する

動作合成とRTL設計の本質的な違い

若林一敏

本稿は動作合成（Cベース設計）について説明する．動作合成の基本原理や利点を述べた後に，「動作」を記述する動作記述と「構造」を記述するRTL記述の本質的な違いを説明する．また，動作合成ツールの合成する回路の特徴を述べる．さらに，動作合成を実用化する上で必要となる統合的な検証環境について簡単に説明する．動作合成はハードウェアの設計パラダイムを大きく変革する技術である．（筆者）

LSIやFPGA（Field Programmable Gate Array）のハードウェア部分の設計工程は，上流から順に，システム設計，機能設計，論理設計，レイアウト設計からなっています．システム設計では画像処理や顔照合など，実現したいアルゴリズム・機能をハードウェアで行うか，CPU上のソフトウェアで行うかを決定します．機能設計では，メモリや演算器を何個使うか（チップ面積），どのくらいのスループット，レイテンシで回路を動かすか（チップ性能）などを決定します．論理設計では，回路をAND，NANDなどのゲート回路で遅延制約や面積制約を満たすように実現します．レイアウト設計で，そのゲートを，LSI上のどこに置くか決定（配置）し，それぞれを結線（配線）します．

論理設計工程を自動化するツールは論理合成ツールと呼ばれており，1980年代半ばから実用化が始まりました．今ではすっかり定着し，特にFPGAでは必ずというほど利用されているのではないでしょうか．論理設計の一つ上流の機能設計を自動化するツールが，今回取り上げる動作合成ツールです．入力は動作記述と呼ばれ，CやC＋＋，SystemC（ハードウェア用の専用言語）で記述されます．出力はVerilog HDLやVHDLなどのRTL（Register Transfer Level）コードです．図1にツールの流れを示します．

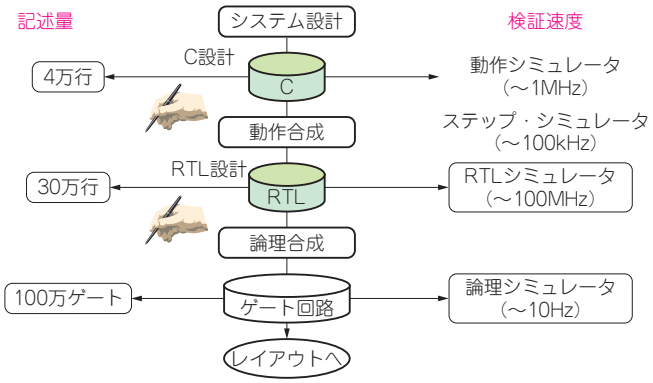

図1　ツールを利用したLSIの設計フロー
100万ゲートの回路を記述した時の行数とシミュレーション速度の一例．

1．動作合成の二つの利点

● RTL設計との量的な違い

設計ツールを導入する際の利点はいろいろあります．代表的なものは設計期間を短縮したい，設計工数を削減したい，バグを少なくしたいなどでしょう．それでは，なぜ設計期間や工数が削減されるのでしょうか．量的な側面からいうと，記述量と検証速度の削減があげられます．

100万ゲートの回路を設計する例を考えます．ゲート・レベル設計では100万個のゲート回路を描く必要があります．HDLによるRTL設計では30万行程度で記述できます．同じ回路を動作レベルで記述すると4万行程度で書くことができます（もちろん，行数は回路や書き方によって大き

Keyword　動作合成，Cベース設計，RTL記述，動作記述，データ・フロー・グラフ，データパス

く異なる）．ソフトウェア設計でアセンブラで書くより，C言語で書く方が少ない行数で書けるのと似ています．

シミュレーションの時間も短縮されます．同様の規模の回路をゲート・レベルでシミュレーションすると数十Hz，RTLでも数百Hzしかでません．動作レベルだと1MHzとか数百kHzの速度がでます．このように記述量，検証時間が短くなるために，設計期間・設計工数が削減でき，設計誤りも少なくなるのです．

● RTL設計との質的違い

動作合成のもう一つの利点は，質的な側面です．従来，「これはハードウェア向きのアルゴリズムではないね」とか「こんな複雑な処理はハードウェア向きではないのでCPUを使おう」とかいう会話がされていました．C言語などで書かれたアルゴリズムは逐次的な動作（CPUは機械語を順に逐次に処理していく）を示しており，これを並列に動作するハードウェアの構造にマッピングするのが，従来のRTL設計です．複雑なアルゴリズムをミスなくRTLに変換するのは非常に困難です．動作合成はアルゴリズムのC記述から回路の構造を記述したRTL記述を合成します．RTL設計ではハードウェア化をちゅうちょするようなアルゴリズムでも，動作合成を用いれば容易にハードウェア化できるのです．

ハードウェア設計というと，ブロック図とかゲート回路図を思い浮かべる人も多いと思います．従来のハードウェア設計は，レジスタや演算器，カウンタなどの部品をどのように結合して所望の動作をさせるかと考えます．つまり，回路の「構造」を考えます．これらの部品を動かすのは制御回路で，ステート・マシンで実現したり，カウンタや微分回路などを利用して制御信号を生成したりします．この「構造」記述された回路は，基本的に並列に動作します．対して，動作レベルのC言語設計では，ハードウェアの構造は記述しないで，回路のアルゴリズムや（逐次的）動作を記述します．

簡単な例としてソート回路を考えます．RTL設計では「入力データを1クロックに二つずつ読み，比較器に同時に入力し，その出力に応じて，入力データを二つのレジスタのどちらに入れる」などと考えて，構造を決めていきます．動作レベルのC記述では，ソートのアルゴリズム（動作）のみを記述します．アルゴリズムや動作の記述は，時間的に逐次に記述されます．この記述を並列化し，構造や制御回路を生成するのは動作合成ツールに任せることになります．

この「構造設計」と「動作設計」の違いを理解することが，動作合成をうまく使いこなす重要なポイントにつながります．

2. RTL記述と動作記述

動作合成を理解するときに注意してほしいのは，動作レベル記述とRTL記述は記述の抽象度のレベルが違うのであって，言語の差（C言語とVHDL，Verilog HDL）ではないということです．C言語で，動作記述だけでなく，RTLやゲート・レベル記述も書けます．Verilog HDLやVHDLは主にRTL言語として開発されましたが，動作レベルやゲート・レベルの記述能力は持っています．

● RTLは構造を記述する

図2を用いて，動作記述とRTL記述の違いを説明します．まず，左側の記述，“X=（a+b）+c; Y=d+e;”をRTL記述だと考えます．RTLは回路の構造を示しており，すべての記述が1クロックで実行（毎クロック実行・評価）されます．従って，右上の回路を示しています．RTLに“+”が三つあるので，回路でも加算器が三つ使われて，1クロックでX，Yの値が並列に計算され，計算結果が同時に代入されます．RTLを入力とする論理合成ツールはいつも右上の回路を合成します（加算器のさまざまなゲート回路構成を出力し，ゲート・レベルの遅延，面積のトレードオフを考えてくれる）．RTL記述はブロック図レベルのハードウェア「構造」を規定しています．

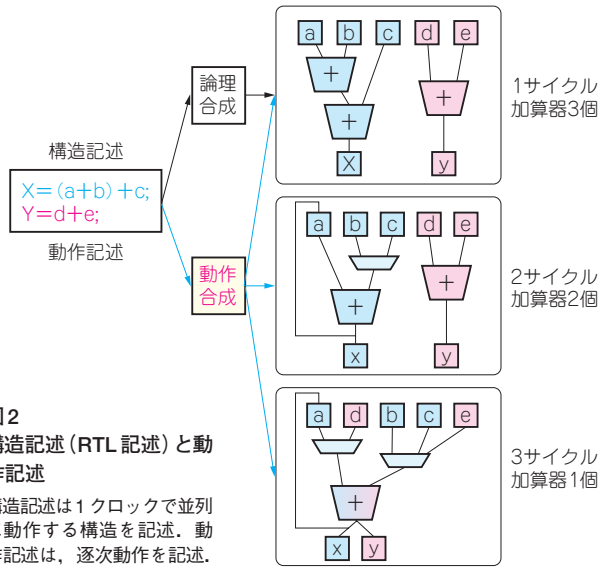

図2
構造記述（RTL記述）と動作記述
構造記述は1クロックで並列に動作する構造を記述．動作記述は，逐次動作を記述．