第3章

動作合成とC/C++/SystemC/SystemVerilogの協調検証

Cynthesizerの活用事例

渋谷貴利，内海功朗，森 義一

ソフトウェアのどの部分をどのような思想でハードウェア化するのか，事例をもとに解説する．具体的には米国Forte Design Systems社の動作合成ツール「Cynthesizer」を利用し，C言語で書かれたソフトウェアの一部をVerilog HDLなどのハードウェア記述言語に変換する．その後，SystemVerilogのDPI機能を利用してハードウェアとソフトウェアの協調検証環境を構築する．（編集部）

ソフトウェア（アルゴリズム）をハードウェア化することは以前から行われてきましたが，最近の設計技術の発展により，容易かつ迅速に実現できるようになってきました．ハードウェア化による利点として，

- 大幅な性能向上
- 低消費電力化（低クロック化）
- CPU処理負担の軽減

などが期待できます．

プロセッサの動作周波数向上は難しくなってきましたが，SOCに求められる機能と処理能力は増える一方です．最近ではマルチコア化などが話題になっていますが，CPU処理を大幅に軽減し，かつ性能向上を実現する方法にハードウェア化があります．この方法は，動作周波数の向上が難しくなった今日，集積度の向上は今後も継続するため，有効なアプローチです．また，ハードウェア化の結果，動作周波数を落とすことで消費電力の低減を実現できます．低消費電力化は21世紀のLSIが目指すべき目標であり，エコ時代への対応やモバイル機器の電池駆動時間の向上などの利点をもたらします．

● これまでは大変だったハードウェア化設計

このように利点の多いハードウェア化ですが，これまではソフトウェアを解析してRTL（Register Transfer Level）の設計をほとんど人手だけで行ってきました．また，ソフトウェアと連携して動作するハードウェアの場合は，FPGA化を実現した段階で初めてハードウェアと制御ソフトウェアを組み合わせてデバッグを行ってきました．

このような方法ではハードウェア化に時間を要し，投入する工数も極めて大きなものになりますし，開発期間も長期化することになります．また，ソフトウェアの特徴であるソース・コード変更の容易性による機能・性能の改善に対してハードウェアは簡単に追随できず，結果としてハードウェア化を行うのは特定の大きな市場向けの製品に限定されていました．

性能向上を見込めるハードウェア化が上述のような理由で限定されることは非常に残念なことです．しかし最近の設計技術の進展は，これらの問題の多くを改善できるようになりました．

1．ハードウェア化のための設計手法

ハードウェア化をより容易に実現する設計技術が利用可能になってきたということに触れました．ここではその手法について説明します．

Keyword 協調検証，ハードウェア化，動作合成，プロファイリング，SystemC，Cynthesizer，SystemVerilog，顔認識エンジン，Face Sensing Engine

● 動作合成技術がハードウェア化を推進

最も重要な手法として動作合成技術[(1)]を挙げます．動作合成技術が登場した際に多くの注目を浴びましたが，一方で設計者の期待の大きさとツールの能力とのギャップによって一時的なブームは去りました．それからツールの改善・改良が着実に進んだこともあり，いよいよ本格的な普及期を迎えようとしています．

動作合成技術はソフトウェアのハードウェア化という点で一番肝心な，ソフトウェアを解析してデータ・パスとその制御回路（ステート・マシン）を設計するという最も手のかかる部分をツール側が実行します．

● ハードウェア/ソフトウェア協調検証を容易に実現

一方，設計とともに検証技術も重要です．ハードウェアとソフトウェアを組み合わせて検証を行う協調検証はこれまで，RTL 設計後に FPGA 化を行ってから，ハードウェアとソフトウェアを組み合わせて，それぞれのデバッグを行ってきました．このような方法は両者の問題の切り分けを必要とするとともに，デバッグの効率性の問題からデバッグ期間が長期化することになります．

この問題を解決するためにはシミュレーション段階で両者を組み合わせて実行することです．協調検証を専門に実行する EDA ツールもありますが，SystemVerilog が持つ機能である DPI（Direct Programming Interface）を用いて，ハードウェアとソフトウェアを組み合わせた検証をシミュレータ単体で実行することが可能になりました．この方法を採用することで，シミュレーション段階で両者のインターフェースを確認し，それぞれのデバッグを進めることができます．詳しくは後述します．

2. ハードウェア化の方針と設計の流れ

今回，筆者らが採用したハードウェア化の方針は以下のようなものです．

(1) ハードウェアとソフトウェアの協調設計とする．
(2) ソフトウェアの関数単位でハードウェアを実現する．
(3) ソフトウェア機能を一切変更しない．

以下，その理由を説明します．

● 必ず実行され負荷の大きな部分をハードウェア化する

複雑かつ規模の大きなソフトウェアをすべてハードウェア化すると，多大な労力を必要とします．一方，ソフトウェアの中身を見ると例外的な処理への対応などが，かなりの部分を占めています．そこで，ソフトウェアの中で必ず実行され，かつ負荷の大きな部分に着目してハードウェア化を行うことで，開発の規模を限定でき，結果として開発が容易になります．

関数単位のハードウェア化については，ソフトウェアと同じ構成とすることで分割単位やパラメータの類がハードウェアとソフトウェアで共通化できます．この結果，性能よりもハードウェア規模の小ささを追求する場合は，ハードウェア化した関数を使わずにソフトウェアに戻せば要求に応えることができます．ハードウェアもモジュール化して開発しておくことで，さまざまな要求に応えることが可能になります．

● ソフトウェアの機能は下げない

従来の人手によるハードウェア設計では，開発を進めるために処理方式そのものを簡略化する，ハードウェア量を削減するためにビット精度を犠牲にする，というようなことが行われていました．動作合成を使用することで，基本的にはそのままソフトウェア機能をハードウェア化できます．また，ハードウェア規模の問題はもはや大きな問題ではなくなってきています．優れたソフトウェアの機能を変えることはソフトウェアの価値をも下げてしまうものと考えられます．筆者らはソフトウェアに最大の価値を置き，処理に関する変更は行いませんでした．

● 設計の流れは分析，実装，検証

ソフトウェアのハードウェア化を行う場合に，どのような手順を踏むでしょうか．**図1** に設計の流れを示します．筆者らは工程を大きく三つに分けました．分析，実装，検証です．この工程は RTL コードをシミュレーション段階で確認するところまでとしています．実際にはこの後，RTL コードから合成された回路を FPGA で動作させ，実機に近い環境で各種の評価を行います．

最初にハードウェア化を行う対象を決定するための分析を行います．ここではソフトウェアを実行した結果からプロファイリングを採取し，ハードウェア化する部分を決定します．

次にハードウェア化する関数を決定後，それを動作合成して RTL コードを生成します．最初に対象となるプログ