評価版ツール

第1章

DCTのC記述からVerilog RTL記述を合成

――ビヘイビア合成ツール「DesignPrototyper」

古渡 俊明

C言語のビヘイビア・レベル記述からHDLのRTL記述を生成するビヘイビア合成ツール(アーキテクチャ合成ツール)の活用チュートリアルをお届けする．本チュートリアルをとおして，スケジューリング，アロケーションといったビヘイビア合成の基本処理を理解していただきたい．使用するビヘイビア合成ツールは，フューチャーデザインオートメーションの「DesignPrototyper(機能限定版)」，サンプル・デザインはDCT(離散コサイン変換)のC言語記述である．いずれのプログラムも本誌付属のCD-ROMに収録されている．

LSIチップの集積度(搭載ゲート数)は年率50％以上の勢いで伸びています．従来，複数のチップで構成されていたシステムが，プロセッサやメモリなども含めて一つのチップに集約可能となってきました．一方，携帯電話を始めとして，電子機器の製品サイクルは，その機能の豊富さにもかかわらず極端に短くなっています．市場への製品投入時期の遅れが利益の損失につながるという状況にあります．ところが，こうした状況にもかかわらず，LSI設計者の数はそれほど増えていないというのが現実ではないでしょうか．こうした問題を抜本的に改善するためには，以下のような要件を満たす設計手法が必要になります．

- 1人の技術者がカバーできる設計範囲を拡大する(設計の抽象度を引き上げる)．
- ハードウェアとソフトウェアの間でトレードオフ評価を行う(できるだけソフトウェアで実現したい．ただし，消費電力は上げたくない)．
- 既存の設計資産を再利用する(アルゴリズム・レベルやRTL，マスク・レベルなどのIPを利用する)
- 検証速度を1けた以上早くする．
- 既存の設計環境と組み合わせて利用できる環境を構築する．

こうした要求を満足する設計環境として，筆者らはC/C++言語を入力とするビヘイビア合成ツール「DesignPrototyper」を開発しました．このツールには，アーキテクチャ合成とインターフェース合成の機能が含まれています．本チュートリアルでは，本誌付属のCD-ROMに収録した機能限定版のDesignPrototyperとサンプル・デザインを利用して，読者のみなさんにC言語入力によるビヘイビア合成の主要な作業工程を体験していただきます[注1]．

1 ビヘイビア合成とは

ビヘイビア合成はアーキテクチャ合成とも呼ばれます．一般的に動作レベル記述と呼ばれるソフトウェア・プログラムのような逐次記述から，ハードウェアにおける動作の並列性や使用する演算器，レジスタ数などを，設計者が与えた制約条件をもとに自動的に推定します．そして，それらの情報をもとに，動作の実行制御を行う制御回路とデータパス回路を生成し，最適化し，最後にそのRTL記述を出力します．

●ビヘイビア合成を構成する二つの機能

ビヘイビア合成の主要な機能には，スケジューリング

注1：今回のチュートリアルでは，インターフェース合成の機能の説明は割愛する．また，今回のDesignPrototyperは，機能限定版ということで，製品版が本来もついくつかの機能を削除している．詳細はCD-ROM内の`readme`に，現バージョンのバグ情報も含めて書かれているので参照していただきたい．

機能とアロケーション機能があります.

●スケジューリング機能

スケジューリングは，入力記述の各処理（たとえば演算式）をどのサイクルで実行するかを決定する処理です．この実行サイクル数は，設計者によって指定された実行クロック周波数などの制約条件と，使用される演算器の遅延情報などをもとに決定されます．したがって，この処理が終了すると実行サイクル数と各サイクルで実行する演算（使用される演算器も）が確定します.

●アロケーション機能

アロケーションは，スケジューリングで確定した各サイクルで実行する演算器やレジスタを選択し，これらの間を接続する処理です．また，演算器やレジスタのライフ・タイム（レジスタがどのサイクルまで，現在の値を維持する必要があるか）などの解析も行います．さらに，演算器やレジスタの共有処理も行います.

●ビヘイビア合成の作業フロー

実際の作業を始める前に，ビヘイビア合成ツール（DesignPrototyper）を使用した一般的なフローを紹介します．ビヘイビア合成のフローは，以下の(1)〜(6)のようになります.

(1) テキスト・エディタを用いて，入力する動作レベル記述（C言語）を作成します.

(2) 動作レベル記述の動作を確認するテストベンチをC言語で作成します.

(3) Cコンパイラによりコンパイルを行い，動作を確認します.

(4) 動作を確認した後，FDA-Cディレクティブ（Design Prototyperのディレクティブ記述）を用いてハードウェア情報を追加します.

(5) DesignPrototyperを用いてビヘイビア合成を行い，RTL記述を出力します.

(6) 出力されたRTL記述を論理合成し，配置配線へ.

本チュートリアルでは，おもに(4)と(5)の工程を中心に説明を行います.

2 サンプル・デザイン

今回取り上げるサンプル・デザインはDCT（離散コサイン変換）です．このC記述を使用して，ビヘイビア合成の基本的な処理を説明します．また，参考として，出力されたRTL記述の論理合成と配置配線を行った結果も示します．ライブラリとしては，米国Xilinx社のVirtexのものを使用しました.

●DCTとは

DCT（discrete cosine transform）は，JPEG（Joint Photographic Experts Group）やMPEG（Moving Picture Experts Group）などの画像圧縮方式で採用されているアルゴリズムです．画像データの画素レベル情報などを周波数成分に変換します.

圧縮における一般的な問題は，ディジタル的に表現された画像，映像ストリーム，音声などのデータのビット伝送速度を最小化することです．DCTにより空間領域から周波数領域に変換することで，ランレングス・エンコーディングなどのアルゴリズムを用いた圧縮が可能となります．本稿では，静止画像データの各画素に対してDCTを適用します．そこで，2次元空間の変換について簡単に説明します．2次元DCTの演算式は以下のとおりです.

$$F_{uv} = \frac{c(m)c(n)}{4} \times \sum_{m=0}^{N-1}\sum_{n=0}^{N-1}\left[f_{mn} \cos\frac{(2m+1)u\pi}{2N} \cos\frac{(2n+1)v\pi}{2N} \right]$$

u，v：周波数変数

f_{mn}：画像ブロック$N \times N$の(m, n)は画素のグレイ・レベル$(0 \leqq m, n \leqq N-1)$

F_{uv}：周波数成分の係数

$c(m)$，$c(n)$：m，$n = 0$の場合，$1/\sqrt{2}$

m，$n \neq 0$の場合，1

JPEGなどでは，画像を8×8のブロックに細分化し，周波数成分に変換します．本稿のサンプル・デザインでも$N = 8$を使用し，さらにCosBlockを以下のように定義した8×8の行列とします.

$$CosBlock_{un} = round\left(factor * \left(\frac{1}{8} \cos\frac{(2n+1)u\pi}{16} \right) \right)$$

コサイン変換の重要な性質として，以下に示すように二つの演算に分離して計算できることが挙げられます.

$$OutBlock = CosBlock \times InBlock \times CosBlock^{T}$$

InBlock：8×8の画像ブロックのf

OutBlock：周波数変換後の出力行列のF