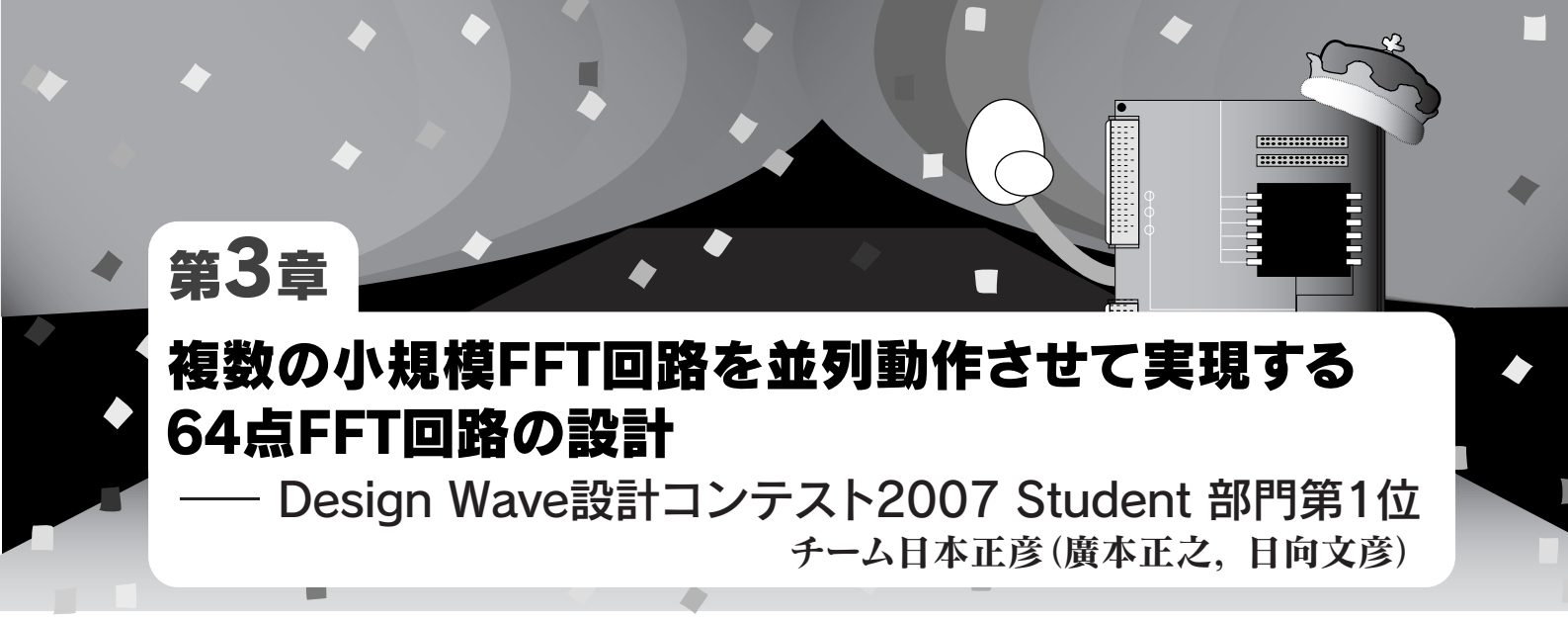

第3章

複数の小規模FFT回路を並列動作させて実現する64点FFT回路の設計

―― Design Wave設計コンテスト2007 Student 部門第1位

チーム日本正彦（廣本正之，日向文彦）

Design Wave 設計コンテスト2007のStudent部門第1位の設計を紹介する．まず，回路規模を小さくすることを目標としてFFT回路を設計した．次に，スループットを上げるために，複数の小規模FFT回路を並列動作させるマルチコア化を行った．任意の並列度のRTLコードを自動生成するために，スクリプト言語を活用した．（編集部）

私たちの所属している研究室では，システムLSIのアーキテクチャや設計方式などについて研究しています．研究の中でHDLを用いて回路を設計したり，FPGAに実装して動作させています．

研究室ではほぼ毎年，修士1回生が本コンテストに参加しています．そこで今年も先輩方に続き，沖縄における発表会を，さらにはコンテストでの優勝を目指したいと考えました．コンテストへの参加を通して，与えられた条件，期間内でハードウェアを設計する経験が得られると思い，課題の64点FFT回路の設計にチャレンジしました．

まず，回路規模の大半を占めるメモリを重点的に削減し，コンパクトなFFT回路を作りました．これを元に，スループットを拡張できるマルチコアのフレームワークを設計しました．ハードウェアの設計やFPGAへの実装，ボード上での組み込みシステムの実現などの貴重な経験ができました．

1. 設計方針を考える

まず，今回の設計で何をアピール・ポイントにするのか，何を狙ったアーキテクチャにするのかという所から考え始めました．

考えられるアプローチとしては，とにかく高いスループットを目指す，逆に回路面積をできるだけ小さくする，あるいはそのバランスの取れた設計を目指す，などがあります．通常の設計では，システムの要求仕様に応じてある程度方向性を決めることができます．しかし今回はコンテストということで自由度が高く，設計方針の決定には非常に頭を悩ませました．

当初はスループットもそこそこ高く，回路規模もそれなりに小さくしたバランスの良い設計を目指そうと思っていました．性能と面積のトレードオフを考慮した，王道とも言える設計方針です．しかし，これだと優れた回路はできますが，あまりおもしろくありません．せっかくコンテストに応募するので何か「オモロイ」ことをやろうと思い，ユニークな特徴を持たせた設計にしようと思いました．

● とにかく小さく

高いスループットか小さい面積か，どちらかにターゲットを絞って攻めてみるのがおもしろいのではないかと考えました．

ここで，FFTの応用例について簡単に考えてみました．設計仕様書にも書かれていた通り，64点FFTがよく用いられるのはOFDM（orthogonal frequency division multiplexing；直交周波数分割多重）方式などの無線通信用途です．このような用途においてはスループットも重要ですが，リソースの限られたモバイル機器上で動作させるため，小

KeyWord FFT，マルチコア，RADIX-4，バタフライ演算，Design Compiler，FPGA，ML403，Virtex-4FX，スペクトラム・アナライザ

規模かつ低消費電力であることが求められます．そこで今回は，とにかく回路規模の小さなコンパクトなFFT回路を目指してみることにしました．

● まずは設計仕様書の回路を合成してみる

回路規模を削減すると言っても，演算器を小さくする，RAMやROMを減らすなど，複数の方法が考えられます．設計仕様書を読むだけではどこに着目すれば最も効果的に回路を小さくできるのか分からないため，まずは設計仕様書通りに回路を記述し，その合成結果を見て作戦を立てようと思いました．また，設計仕様書通りの実装例を作っておくことで，最終的に自分の設計と比較対象をすることもできます．

設計仕様書に従いRADIX-4のバタフライ演算器とシフト・レジスタを用いて設計した回路を**図1**に示します．各ステージはパイプライン動作し，シリアルに入力されるデータを絶え間なく処理することが可能です．リオーダ処理には64点データが格納可能なメモリを2セット用い，こちらも連続的にデータを流せるようにしました．回転因子についてはあらかじめ計算しておいた三角関数値を格納したROMを用いました．

この回路を米国Synopsys社のDesign Compilerを用いて合成し，遅延時間および回路面積の評価を行った結果を**表1**に示します．合成に用いたライブラリは台湾UMC（United Microelectronics Corp.）の0.18 μm プロセスです．遅延時間の単位はns，回路規模は2入力NAND ゲートに換算した場合のゲート数としています．**表1**にはFFT回路全体の合成結果のほかに，組み合わせ回路の中で大部分を占めるバタフライ演算器と回転因子乗算器についても，個別に合成した結果を示しています．演算器の合計規模は約9,000ゲートであり，FFT回路全体の約15%にしか過ぎ

表1　設計仕様書に従ったアーキテクチャの合成結果

	遅延時間［ns］	面積［ゲート］
FFT全体	4.98	62111
バタフライ演算器（ステージ1）	2.80	808
バタフライ演算器（ステージ2）	2.49	1061
バタフライ演算器（ステージ3）	2.75	1240
回転因子乗算器（ステージ1）	4.09	2928
回転因子乗算器（ステージ2）	4.29	3251

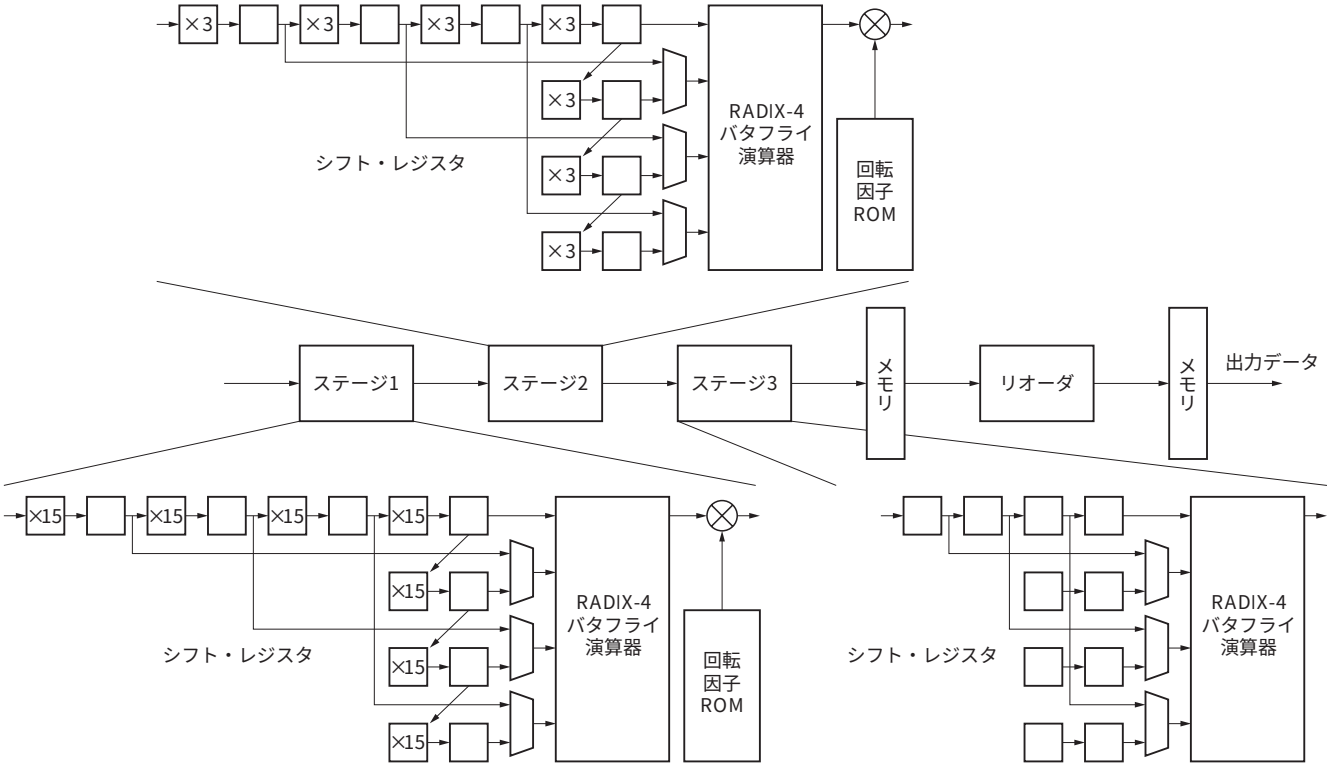

図1　設計仕様書に従ったアーキテクチャ

各ステージはパイプライン動作し，シリアルに入力されるデータを絶え間なく処理することが可能．リオーダ処理には64点データが格納可能なメモリを2セット用い，こちらも連続的にデータを流せるようにした．回転因子についてはあらかじめ計算しておいた三角関数値を格納したROMを用いた．