社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-SLDM-138 (31)
2009 / 1 / 30

粒度可変論理セルにおける算術演算向け配線アーキテクチャの一検討

佐藤 嘉晃† 趙 謙† 尼崎 太樹†† 飯田 全広†† 末吉 敏則††

† †† 熊本大学 大学院 自然科学研究科 〒 860-8555 熊本市黒髪 2-39-1
E-mail: †{satou,cho}@arch.cs.kumamoto-u.ac.jp, ††{amagasaki,iida,sueyoshi}@cs.kumamoto-u.ac.jp

あらまし リコンフィギャラブル IP (Intellectual Property) を SoC (System on a chip) に搭載することで，専用回路である ASIC (Application Specific Integrated Circuit) の性能を生かしつつチップに柔軟性をもたせることができる．しかしながら，代表的なリコンフィギャラブルロジックデバイスの FPGA (Field Programmable Gate Array) をそのまま IP として用いるだけでは性能面で問題がある．そこで我々はリコンフィギャラブル IP として粒度可変論理セル VGLC (Variable Grain Logic Cell) を提案している．従来の VGLC は汎用的な使用を目的としているのに対し，本稿では算術アプリケーションに特化した配線構造を提案する．データフローグラフより接続構造における特徴量を抽出し，配線構造として用いた場合の評価を行った．結果として，FFT，FIR を対象とした場合，クラスタ内部の論理ブロック数が 4 の場合に平均して最も実装効率が良くなった．また，両演算を同一の配線構造でマッピングを行う場合，それぞれ個別の配線構造と比較して，配線に要するスイッチ数の増加は 33% に抑えられることがわかった．
キーワード リコンフィギャラブルロジックデバイス，配線構造，算術演算，Data Flow Graph

A Study of Routing Architecture on Variable Grain Logic Cell for DSP Application

Guide to the Technical Report and Template

Yoshiaki SATOU†, Quen CHO†, Motoki AMAGASAKI††, Masahiro IIDA††, and Toshinori SUEYOSHI††

† †† Graduate School of Sience and Technology, Kumamoto University, 2-39-1 Kurokami, Kumamoto 860-8555, Japan
E-mail: †{satou,cho}@arch.cs.kumamoto-u.ac.jp, ††{amagasaki,iida,sueyoshi}@cs.kumamoto-u.ac.jp

Abstract A Reconfigurable Logic Device (RLD), which has circuit programmability, is applied to embedded systems as a hardware Intellectual Property (IP) core. However, conventional RLDs,which are commercial Field Programmable Gate Arrays (FPGAs), cannot achieve efficient implementation. Then, we have proposed Variable Grain Logic Cess (VGLC) as a reconfigurable IP core. VGLC is a reconfigurable logic architecture that has both flexibility and high performance. Whereas traditional VGLC assumes general-purpose use, we propose routing architecture specialized in arithmetic applications, in this paper. We extract characteristic in the connection structure from a data flow graph, then think routing architecture using it. As a result, when we target FFT and FIR, four logic block in cluster is improves implementation efficiency most on average. In addition, when both operation are implemented by the same routing architecture, number of swiches increase 33% in comparison with each individual routing architecture.
Key words Reconfigurable Logic Device, Routing Architecture, Arithmetic operation, Data Flow Graph

1. はじめに

近年，携帯電話やディジタルカメラ，車載機器などのディジタル機器は小型化，高性能化が進んでいる．これらは通常，CPU，メモリ，I/O，専用ハードウェアなどを1チップに搭載したSoC（System On a Chip）により実現される．しかし，半導体プロセスが進むにつれてマスク費用は増大し，設計期間も長期化している．そのため，SoCの開発は多くの生産数が見込めない限り利益を得ることが困難となってきた．そこで，再構成可能論理デバイス（Reconfigurable Logic Device，以下RLD）が注目されている．RLDは，アプリケーションに対応してハードウェアを再構成することが可能なため，専用ハードウェアと同様に処理の並列性を活かし，高速に処理を実行することができる．このRLDをリコンフィギャラブルIP（Intellectual Property）として使用することで，プラットフォームに合わせた再構成可能なSoCが実現できる．しかしながら，現在，最も普及しているRLDであるFPGA(Field Programmable Gate Array)をIPとして用いるだけでは専用ハードウェアに対して性能で課題を残す[1]．そこで，我々は新たなリコンフィギャラブルIPとしてVGLC（Variable Grain LogicCell）を提案している[2][3]．

VGLCは論理セル単位で演算粒度が可変であるという特徴をもち，4bitの加減算や2〜4入力の論理演算などの演算機能をホモジニアスな論理セルで実装可能である．文献[4]ではVGLCの配線構造としてFPGAに代表されるアイランドスタイルを想定している．これは汎用目的での用途が想定されているためである．一方で，本稿ではVGLCのもつ算術演算機能に着目し，この演算に特化した配線構造について検討を行う．SoCへの搭載を考えた場合，周辺回路との組合せによりチップとしての用途が特定される．例えば，画像処理や音声処理用途向けのチップでは，リコンフィギャラブルIPに実装されるのは算術演算処理に特化した回路が実装されると考えられる．この場合，配線構造はアイランドスタイルのように高い柔軟性をもつ必要はなく，粗粒度方式にみられるように算術演算に特化した配線構造をもてば良い．本稿では，算術演算の中でもSoCにおけるアクセラレータとして多く用いられる2種類のアプリケーション（FFT，FIR）に特化した配線構造を提案する．以下，第2章ではVGLCでの算術演算実装の問題点について述べる．第3章では提案する算術演算向けの配線構造について述べ，第4章で配線構造のパラメータを変化させた場合の実験結果を述べる．最後に第5章でまとめと今後の課題を述べる．

2. VGLCにおける算術演算実装の問題点

本章では，汎用目的での粒度可変論理セルVGLC[4]における算術演算の実装方法とその問題点について述べる．VGLCは，図1に示すBLE（Basic Logic Element）を基本単位として構成し，論理ブロック単位で演算粒度を切り替えることが可能である．BLEはANDとMUX，EXORから構成され，入力にC，X，Y，Z，AS，出力にT，Sを持つ．メモリ値を変更することで，図1の加減算（ADD/SUB），2入力の論理演算（2RMCF：Reed-Muller Canonical Form），マルチプレクサの3種類の機能を実行できる．また，BLEを複数使用することで，多ビットの加減算器や3入力以上の論理演算を構成可能である．

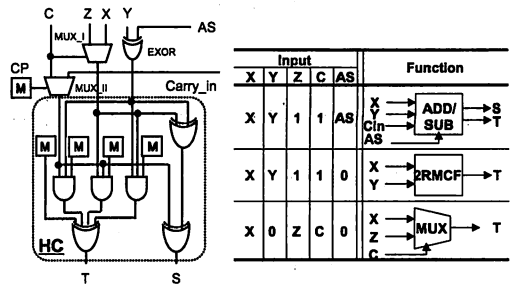

図1 BLEの構成とファンクション

VGLCでは加減算機能と論理演算機能を使用し，加減算や乗算などの四則演算を様々なアルゴリズムで構成することが可能である[5]．しかしながら，算術演算はランダムロジック演算とは異なり，データの流れに規則性がみられるため，アイランドスタイルのような配線構造では配線リソースの無駄が生じる．文献[4]では論理ブロック間の評価だけであり，配線部を考慮すると多数のスイッチを経由するため遅延が大きくなることが予想できる．また，算術演算回路に特化した場合，VGLCの論理演算機能はあまり使用されないと考えられる．実際，粗粒度型のRLDの論理ブロックは一般に，ALUやシフタ，レジスタなどから構成され，論理演算機能はANDやORなどの基本論理演算に限定されている．これより，本稿では算術演算のデータ構造に着目し，その特徴を配線構造に組込むことで，リコンフィギャラブルIPとしての性能改善を目指す．特に，配線部分のオーバヘッドを抑えるために，nビットの加減算や乗算を単一の論理ブロックで実行できるように構成する．これらについては次章で詳述する．

3. 算術演算向け配線構造決定のためのアプローチ

特定の算術演算に特化した配線構造を検討するにあたり，演算間のデータの流れに着目する．具体的には，算術演算中の論理的な接続関係を示すDFG（Data Flow Graph）より，必要となる配線を抽出し，配線構造を検討する．以後，まず最初に論理ブロック構造を示し，その後配線構造の検討を行う．

3.1 演算方式

論理ブロックの構成の前に，演算方式の決定を行う．演算方式にはビットパラレル演算方式とディジットシリアル演算方式とがある．ビットパラレル演算方式はワード単位に1サイクルで演算を行う方式である．一方，ディジットシリアル演算方式は1ワードを複数のビットグループに分割し，グループ毎に逐次的に演算を行う方式である．本稿では，特定の算術演算（FFT，FIR）向けの，限られた配線リソースをもつ構造を目指している．そのため，演算器をコンパクトに実現可能であると

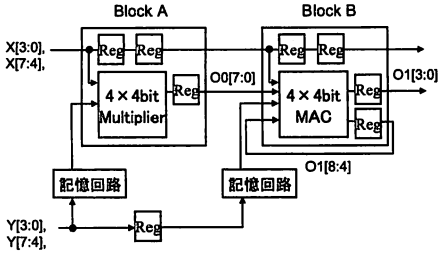

図 2　ディジットシリアルパイプライン乗算器

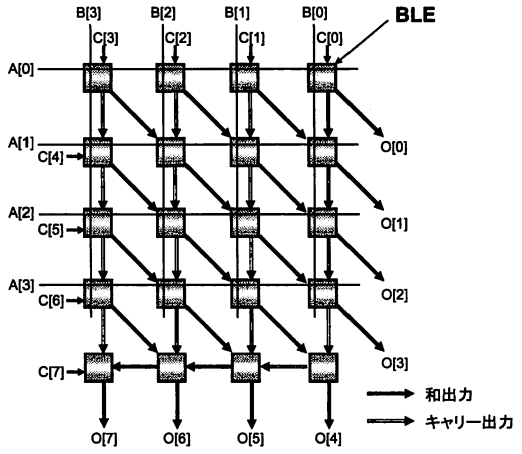

図 3　提案論理ブロック

いう点と，ワード幅変更の影響を受けにくいという点からディジットシリアル演算方式を採用する．

ここで，ディジットシリアル演算器の構成例として，図 2 にディジット幅 4 の 8 ビットディジットシリアル乗算器の構成を示す．X，Y は入力を，O0，O1 は出力を表す．入力 X，Y と出力 O1 はそれぞれ最下位 4 ビットずつディジットシリアルで入力，および出力される．入力 X の転送系路上にはパイプラインレジスタとは別に遅延素子が存在し，部分積の桁合わせを行っている．また，記憶回路はシフトされてくる入力 Y を乗算の実行中，決まった乗算器に供給できるようにするためのもので，Block A には Y[3:0] を Block B には Y[7:4] を供給する．この構成においては，演算のビット幅が n 倍になると，カスケード接続する乗算器数が O(n) で増加する．

3.2　論理ブロック

一般に，粗粒度方式のリコンフィギャラブルロジックでは加減算や乗算，シフト機能を有し，論理演算は AND や OR などの基本論理演算に限定される [6] [7]．そこで，本稿においても加減算や乗算に機能を絞った論理ブロックを提案する．そこで本稿においては，VGLC を構成する BLE を基本単位とし，加減算，乗算に機能を絞った論理ブロックを用いる．図 3 に提案論理ブロック構造を示す．この論理ブロックは 4 つの入力と 1 つの出力をもち，入力 A，B，C は 4 ビット，入力 C，出力 O はそれぞれ 8 ビットである．入力 C は MAC（Multiply and ACcumulation）における加算項である．本論理ブロックでは 4x4 ビット乗算が可能である．また，BLE 間の接続はアレイ型乗算器の接続構成を採用している．提案論理ブロックの機能を以下に挙げる．

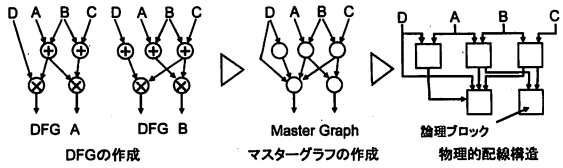

図 4　配線構造決定までの流れ

- MAC 演算　　：A × B+C
- 加減算　　：A+B-C
- ビットシフト　　：A << 1
- 2 入力論理演算

3.3　論理ブロック間の配線構造

本稿では，算術演算の DFG（Data Flow Graph）より配線構造の検討を行う．DFG とはデータの流れを示したグラフであり，演算同士の接続関係を現す．そのため，DFG における接続関係を物理的配線として配線構造にもたせることで，効果的な算術演算の実装が期待できる．以降，DFG 中の演算をノード，演算間のデータの流れをエッジと呼ぶ．図 4 に配線構造決定までの流れを示す．まず，各算術演算の DFG を導出する．その後，DFG のマッチングを行い，各算術演算の接続関係を網羅したグラフを作成する．このグラフのことを本稿ではマスターグラフと定義する．マスターグラフは DFG の接続関係のみを現すため，ノードにおける演算の記述はない．なお，ここでのマッチングとは，各 DFG どうしのノードを対応付けし，DFG 間の異なるエッジ（接続関係）を導出する作業である．マスターグラフのエッジが少ないほど必要な配線数が少なくなる．最後に，得られたグラフをから物理的な配線構造を決定する．このとき，マスターグラフのノードは論理ブロックに 1 対 1 に対応する．

3.3.1　配線構造の階層化

各 DFG のマッチングにおいては，あるノードに対し，どのノードを対応させれば最も少ないエッジの増加でマスターグラフを実現できるかを判定する．これはノード間の対応をとる点で，グラフ同士が同形がどうかを判定するグラフ同型性判定問題に類似している．この問題は，多項式時間で解けるアルゴリズムがまだ知られていない計算困難な問題の一つである [8]．そのため，ノード数が多くなると実時間での判定が困難となる．そこで，DFG を部分グラフに分割し，最低限必要となる配線の導出を行う．DFG を部分グラフ単位に扱うことで，マッチングにおける計算量を減らすことができる．この部分グラフへの分割を複数回行い，DFG を階層化することで，グラフを簡略化する．

図 5 に DFG を部分グラフ単位に分割し，マスターグラフを作成する例を示す．この例では，部分グラフに分割する際の最大ノード数は最初の分割が 3，2 回目では 4 である．図 5 の DFG

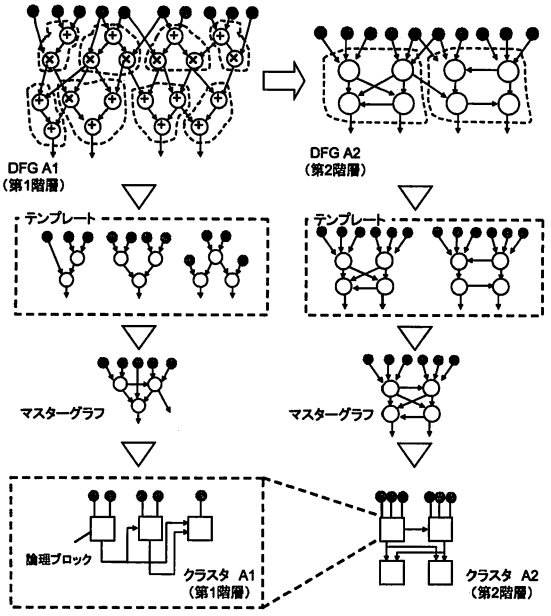

図 5　DFG の階層表現

A1 を図中の破線部分を部分グラフとして，新たなノードで表現したグラフが DFG A2 である．DFG A2 においても同様に，破線部分を部分グラフとして抽出している．DFG A2 の各ノードは DFG A1 の部分グラフの構造を内部にもち，DFG A1 を第 1 階層，DFGA2 を第 2 階層として，階層化されている．

DFG の部分グラフへの分割はできるだけ最大ノード数で行うとともに，規則的な構成を抜き出す．部分グラフへの分割は Yuanqing Guo らのアルゴリズムを使用する [9]．このアルゴリズムはグラフ中の，ノード数 n 以下の全てのグラフパタンを抽出するもので，得られたグラフパタンより使用するグラフパタンを選択し，DFG を部分グラフに分割する．部分グラフ中の全ての異なる接続関係のグラフをテンプレートと呼び，これらの接続関係を全て表現可能なマスターグラフを作成する．そして，マスターグラフを表現できるよう論理ブロックをグループ化したものをクラスタと定義し，クラスタする論理ブロックの数をクラスタ数と定義する．部分グラフに分割する際の最大ノード数とマスターグラフのノード数およびクラスタ数は全て同じ値である．この例では第 1 階層では 3 つの論理ブロックがクラスタされており，第 2 階層では第 1 階層のクラスタがさらに 4 つクラスタされている．このクラスタ構造はクラスタ A1 の構造を下位層，クラスタ A2 の構造を上位層とした階層構造をもつ．図 5 では第 2 階層までしかないが，さらに階層が続く場合もある．提案する配線構造は多重の階層構造で実現され，階層ごとに異なる配線構造をもつ．図 5 の例では 1 つの DFG しか取り扱ってないが，対象とする算術演算の DFG それぞれにこの処理を行う．そして，得られた同階層のマスターグラフ同士で，マッチングをとり，階層ごとに配線構造の決定を行う．なお，クラスタ化後も論理ブロック間の接続関係は保持しているため，実行時の論理段数は変わらない．クラスタ数や階層数は

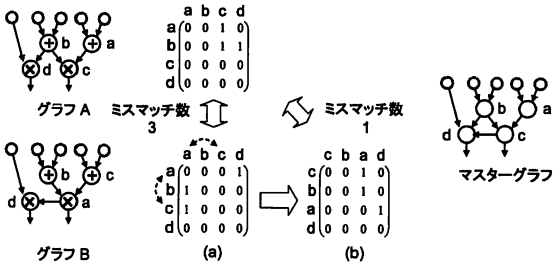

図 6　グラフのマッチング例

対象とする算術演算の DFG 構造によって適する値が異なると考えられる．そのため，それぞれどれぐらいの値がいいのかを検討する必要があり，第 5 章においてクラスタ数の検討を行う．

3.3.2　マッチング

テンプレートまたはマスターグラフ間のマッチングはノードとノード間の接続関係を示した隣接行列を使用する．隣接行列とは，あるノード v とノード w 間にエッジが存在すれば 1 を，存在しない場合は 0 を記述した行列で，隣接行列が一致すればグラフは同形である．図 6 に DFG とその隣接行列を示す．DFG は有向グラフであるため，ノード v からノード w へ向かうエッジがあるときのみ (v,w) 成分が接続を示す 1 となる．ここで図 6 のグラフ A とグラフ B のマッチングについて考える．グラフ A の隣接行列とグラフ B の隣接行列とで異なる行列の成分の数をミスマッチ数と定義する．ミスマッチ数が少なければグラフ同士で多くのエッジが共有できていることを示している．そのため，マスターグラフを作成する際，基準となるグラフに対して新たに追加するエッジも少なくてすむ．この図ではグラフ A とグラフ B の (a) の隣接行列のミスマッチ数は 3 である．これはグラフ A のノード a,b,c,d とグラフ B のノード a,b,c,d をそれぞれ対応させた場合，どちらかのグラフに新たに 3 本のエッジを追加することで両グラフを表現できることを意味する．一方，グラフ B の (a) の隣接行列における a と c を入れ替えた隣接行列 (b) は，グラフ A の隣接行列に対してミスマッチ数が 1 である．つまり，グラフ A のノード a,b,c,d とグラフ B のノード c,b,a,d をそれぞれ対応させた場合は，新たに追加するエッジが 1 本で済むということである．このように，同一のグラフでもノード間の対応のとり方で，エッジの共有数が変わってくる．ノード間の対応のとり方はノード数 n の階乗数存在する．本稿ではその全てを探索し，最もミスマッチ数が少なくなるようにグラフ同士を対応付けている．

4.　計算機実験

本章では，DFG を部分グラフに分割する際のクラスタ数の検討を行う．本稿で提案する配線構造は図 5 に示すような多層の階層をもつ．ただし，本稿では事前評価として第 1 階層のみを使用した評価を行う．このため，クラスタ間の接続は，各々のアプリケーションに対し最適な接続構造をもつものと仮定する．提案する配線構造は論理ブロック間やクラスタ間に DFG としての接続関係をもつため，演算は最適な接続関係で実行さ

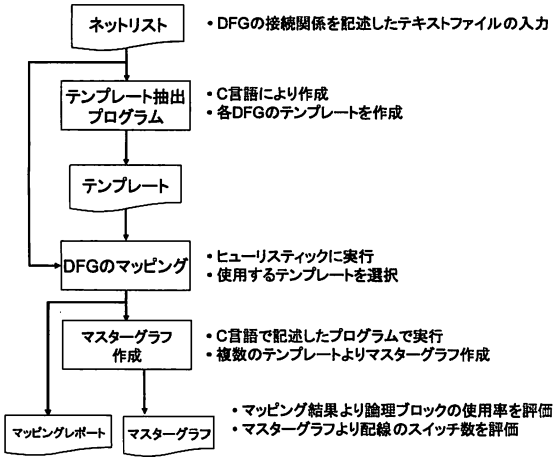

図 7　評価フロー

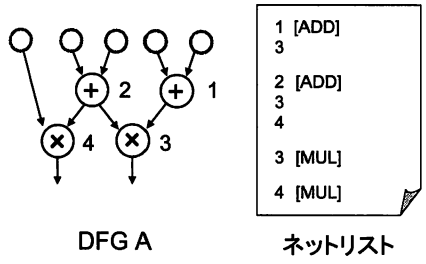

図 8　DFG とネットリスト

れる．そのため，本仮定は提案アプローチに矛盾するものではない．

クラスタ数とはグループ化する論理ブロックの数であり，DFG を部分グラフに分割する際の最大ノード数でもある．このクラスタ数が多い場合，FFT におけるバタフライ演算のように一定の規則性をもつ配線構造を抽出できる可能性が高くなる．ただし，内部の論理ブロック数も増加するため，マッピング時に実装効率をいかに増加させるかが課題となる．一方で，クラスタ数が小さい場合は高い実装効率を得ることができる．しかしながら，クラスタ内でマッピングできる規模が少ないため，効果的な配線構造を抽出できるかどうかはアプリケーション構造に依存する．本実験ではまず最初に複数のアプリケーションを用いて各々に適したクラスタ数および配線テンプレートの算出を行う．続いて，配線テンプレートを配線構造として用いた場合に必要となるスイッチ数を求める．実験はクラスタ数 3，4，5，6 の 4 種類において行う．評価項目としては，使用論理ブロック数，クラスタ内の論理ブロックの使用率，配線に要するスイッチ数とする．評価アプリケーションとしては，入力ビット幅 8，16 の 16 ポイント FFT，16 タップ FIR を用いる．FFT は Cooley-Tukey 型の FFT 回路を対象とし，FIR はシリアルに入力を処理する直線型 FIR を対象とする．アプリケーションの実装はハンドマッピングにより行う．

4.1　評価フロー

図 7 に評価フローを示す．入力のネットリストには，アプリケーションの DFG を記述したテキストファイルを与える．今回はアプリケーション構造を解析し，ハンドマッピングにより作成している．図 8 に DFG とそれに対応するネットリストの記述例を示す．DFG の各ノードに対してはあらかじめ番号を付けておき，この番号を用いてネットリストにおいて接続関係を表す．ネットリストでは，それぞれのノードに対して，演算の種類と接続先のノード番号を記述している．ただし，ノード番号 3 や 4 のように接続先が無い場合は，演算の種類のみの記述である．このネットリストをテンプレート抽出プログラムの入力として与え，4 種類のクラスタ数についてテンプレートの抽出を行う．テンプレート抽出プログラムは DFG 中のクラスタ数 n の全てのテンプレート構造を抽出する．マッピングでは得られたテンプレートの中から，最も効率よくマッピングできるようテンプレートを選択し，DFG のマッピングを行う．この処理についても今回はハンドマッピングにより行っている．得られたマッピング結果より，論理ブロック数と論理ブロックの使用率の評価を行う．マスターグラフの作成は単一のアプリケーションに対するマスターグラフ，もしくは複数のアプリケーションに対するマスターグラフを作成する．

4.2　実験結果と考察

まず最初に DFG のマッピング方針について述べる．マッピングに使用するテンプレート群はクラスタ数 n のすべての配線パタンが含まれている．そのため，その中よりできるだけ同一のパタンでマッピングを実行できるものを選択する．また，実装効率の観点よりマッピング時にはできるだけ最大ノード数となるよう配慮する．マッピングは全てのテンプレートを使用した場合を検討し，最も効率のよいものを選択する．マッピングにおける参考として，表 1 に各クラスタ数における FFT とバタフライ演算器の総テンプレート数を示す．バタフライ演算器とは FFT を構成する演算回路である．このように FFT がバタフライ演算で構成されることに着目すると，テンプレート数が半減する．そのため，本実験ではバタフライ演算単位でマッピングを行うこととする．なお，8，16 の両ビット幅で同様の結果が得られ，ビット幅が変わってもテンプレート数に変化がなかった．

表 1　FFT とバタフライ演算器の総テンプレート数

ノード数	FFT	バタフライ演算器
3	4	2
4	10	5
5	21	13
6	43	28

各クラスタ数でマッピングした場合の論理ブロックの使用率を図 9 に，使用論理ブロック数を図 10 に示す．図 9 より，FIR の実装はほぼ 100% の使用効率を達成していることがわかる．これは論理ブロックが 1 列にカスケード接続される簡易な構成のためである．また，シンプルな構成であるため，クラスタ数が変化しても，実装効率に大きな影響は無い．一方，FFT は

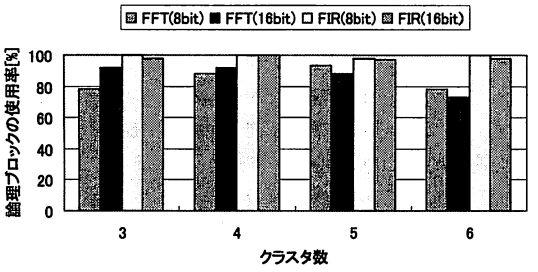

図 9 各クラスタ数での論理ブロック使用率

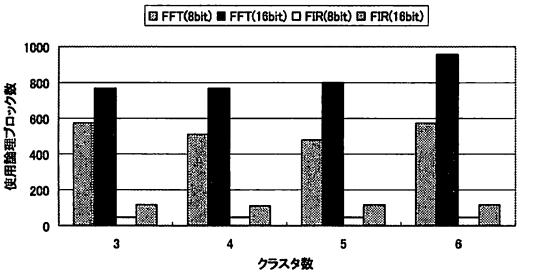

図 10 各クラスタ数での論理ブロック使用数

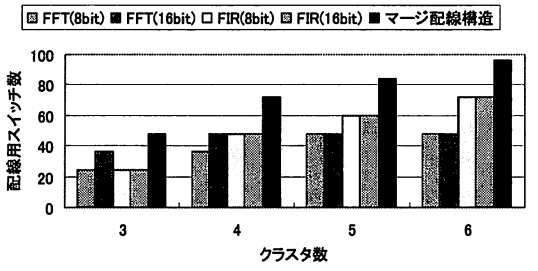

図 11 各クラスタでの配線用スイッチ数

FIR に比べ，論理ブロックの使用率が低下している．これは FIR に比べて構成が複雑であるため，実装において内部の論理ブロックを使い切れないからである．FFT に関してはクラスタ数が 3，4，5 の場合に比べ，クラスタ数が 6 の場合は使用率が低下する．使用論理ブロック数についても，クラスタ数が 6 の場合は多くの論理ブロック数を必要とする．ビット幅の変化による，使用率への影響はあまり大きくないといえる．

次に，各アプリケーションにおける配線のスイッチ数とマージ配線構造のマスターグラフにおける配線のスイッチ数を図 11 に示す．ここでのマージ配線構造のマスターグラフとは，4 種類の演算回路の接続関係をマッチングにより合成したもので，これらの演算に対応した配線構造をもつ．図 11 より，各演算回路のスイッチ数に対するマージ配線構造のスイッチ数の増加幅はクラスタ数によらず，ほぼ一定である．マスターグラフを作成する際は，できるだけ配線を共有するようにしており，できなかった場合に，新たな配線として追加される．そのため，クラスタ数によらず，共有できない配線数がほぼ同じであることを示している．また，同一クラスタ数における各演算回路のスイッチ数に大きな違いはない．これより，今回対象とした演算回路をクラスタ単位に扱った場合，その接続構成が類似し，両演算を扱える配線構造が少量の配線で実現できるといえる．

5. まとめと今後の課題

本稿では算術演算向けの配線構造を DFG から検討し，多層の階層をもつ配線構造の提案を行った．今回はその事前評価として，最下層におけるクラスタ数を 3，4，5，6 の 4 種類の場合で実験を行い，最適なクラスタ数とクラスタ数の違いによる実装効率の変化を調査した．結果として，入力ビット幅 8，16 の FFT と FIR に関してはクラスタ数が 3，4，5 の場合に良い結果が得られ，論理ブロックの平均使用効率に関してはクラスタ数 4 が最も良かった．また，FFT と FIR の接続関係が類似していることより，少量の配線の追加で，両演算に対応する配線構造が実現できるという結果を得た．

今後はクラスタ数 7 以上の場合の評価や FFT，FIR 以外のアプリケーションの実装評価を行い，アプリケーションとクラスタ数の関係を調査する．また，今回は 1 階層での評価であったため，多層の階層構造での評価を行う必要がある．

文　献

[1] I. Kuon, J. Rose：“Measuring the Gap between FPGAs and ASICs ”, Proc.FPGA, pp.21-30, Feb. 2006.

[2] 尼崎太樹，中山英明，山口良一，松山和憲，飯田全広，末吉敏則：“粒度可変構造をもつ再構成論理セルアーキテクチャ”，電子情報通信学会論文誌 vol.J90-D no.6, pp. 1346-1356, Jun.2007.

[3] M. Amagasaki, R. Yamaguchi, K. Matsuyama, M. Iida and T. Sueyoshi, “A variable grain logic cell architecture for reconfigurable logic cores”, In Proc. International Conference on Field Programmable Logic and Applications (FPL07), pp.550-553, Aug. 2007.

[4] K. Inoue, K. Matuyama, Y. Satou M. Koga, M. Amagasaki, , M. iida, T. Sueyoshi：“A Npvel Cluster-based Logic Block with Variable Grain Logic Cells”, Proc.14th IFIP International Conference on Very Large Scale Integration (VLSI-SoC2008), pp.198-203, Nise, France, Oct. 2008.

[5] 佐藤嘉晃，尼崎太樹，山口良一，飯田全広，末吉敏則：“粒度可変構造論理セル向け算術演算回路の実現”，電子情報通信学会技術研究報告 RECONF2007-7, Vol.107, No.41, pp. 37-42, May 2007.

[6] Reiner Hartenstein：“A Decade of of Reconfigurable Computing : a Visionary Retrospective”, Proc.of the conference on Design, automation and test in Europe, pp.642-649, 2001.

[7] 天野英晴：“動的リコンフィギャラブルシステムの最近の動向”電子情報通信学会誌，pp.478–483, May. 2008.

[8] 福田和真，中森眞理雄：“グラフ同型性問題の解決における完全マッチング問題の利用の効果”，電子情報通信学会論文誌 vol.J85-D-I, No.10, pp. 994-999, Oct 2002.

[9] Yuanqing Guo, Gerard J.M. Smit, Hajo Broersma, Paul M. Heysters：“A graph covering algorithm for a coarse grain reconfigurable system”, Proc.he 2003 ACM SIGPLAN conference on Language, compiler, and tool support for embedded systems, pp.199-208, San Diego, CA, July. 2003.