Vol.2016-ARC-219 No.26
Vol.2016-SLDM-175 No.26
Vol.2016-EMB-40 No.26
2016/3/25

再構成可能配線構造検討のためのアーキテクチャ評価環境

山本 浩平[1,2]　森岡 俊樹[1,2]　井上 智哉[1,2]　毛利 真崇[1,2]　密山 幸男[1,2,a)]

概要：再構成可能デバイスはその構造的特徴から配線部が面積効率や動作速度に与える影響が大きくなるため、再構成可能アーキテクチャ開発において、配線構造を慎重に検討することが肝要である。粗粒度再構成可能アレイは、FPGA 等の細粒度型と比較して高い面積効率と電力効率が期待できることが知られている。しかし、面積効率を追及するあまりに配線リソースを削減し過ぎると、マッピング効率が低下して必要なアレイサイズが大きくなり、結果的に面積効率の低下につながる。さらに、ルーティングが複雑になることで、経由スイッチ数に依る配線遅延が大幅に増大する。そこで本稿では、再構成可能配線構造を検討するうえで、高い精度で配線遅延や面積を評価することができる環境を構築した。本評価環境を用いることにより、配線構造と配線リソースをパラメータ化して定義することで、アプリケーションマッピング結果から、配線構造の有効性を定量的に評価することが可能である。さらに、本評価環境により自動性成した RTL ソースを用い、既製のシミュレーションツール上で論理検証を行うことも可能である。

キーワード：再構成可能，配線構造，評価環境，自動配置配線，アーキテクチャ探索

1. はじめに

演算器を搭載する基本要素 (Processing Element: PE) のアレイ構造をもつ粗粒度再構成可能アレイ (Coarse Grained Reconfigurable Array: CGRA) は、ターゲットアプリケーションがアーキテクチャに適合する場合、FPGA に代表される細粒度再構成可能アレイに対して極めて高い性能面積効率を実現が期待できる。再構成可能アーキテクチャは、その構成的特性から、演算部よりも配線部が動作速度ならびに面積に与える影響が大きくなる [1][2]。とりわけ CGRA は、高い面積効率を実現するため、対象とするアプリケーションドメインに適した配線構造および配線リソースの探索が求められる。

配線リソースが少ない場合、配線部のスイッチ数が減少する。そのため、FIR フィルタのような配線が簡易なアプリケーションを実装する場合には、高い面積効率と動作速度を実現することができる。しかし、FFT のように演算器間の配線接続が複雑なアプリケーションを実装する場合には、配線リソースが乏しいと、必要なアレイサイズが大きくなり、結果的に面積効率が大幅に低下する。また、大きく迂回するパスが発生することにより、動作速度が著しく低下することが予測される。

一方、配線リソースが多い場合、配線部のスイッチ数は増加する。しかし、配線の複雑なアプリケーションをマッピングするときには配線リソースが少ない場合に比べて小さいアレイサイズでのマッピングが可能であるため、結果的に面積効率が向上する。また、経由スイッチ数が減ることによって動作速度の向上が見込める。一方、配線の簡易なアプリケーションをマッピングする場合には、配線スイッチの面積・遅延オーバヘッドの大きさが影響し、配線リソースが少ない場合に比べて面積効率と動作速度ともに低下することになる。

また、配線構造が異なると同程度の配線リソースであっても、面積効率や動作速度が大きく異なる。再構成可能アレイの特徴として、アプリケーションマッピング（配置配線）の結果によっても面積や遅延が大きく変わるため、配置配線ツールを用いた実際のマッピング結果から配線構造を定量的に評価することが重要である。

本研究では、配線構造と配線リソースをパラメータ化して定義することで、配線遅延や面積の高精度な評価を行うことができる環境を構築した。さらに、本評価環境は、評価対象アーキテクチャの回路図やアプリケーションマッピング結果の表示機能、ならびに論理検証可能な RTL 記述の自動生成機能を有している。本評価環境により、再構成可能配線構造検討において、アプリケーションマッピング結果にもとづく高精度な定量的評価が可能である。

[1] 高知工科大学
Kochi University of Technology
[2] 独立行政法人科学技術振興機構, CREST
CREST, JST
[a)] mitsuyama.yukio@kochi-tech.ac.jp

Vol.2016-ARC-219 No.26
Vol.2016-SLDM-175 No.26
Vol.2016-EMB-40 No.26
2016/3/25

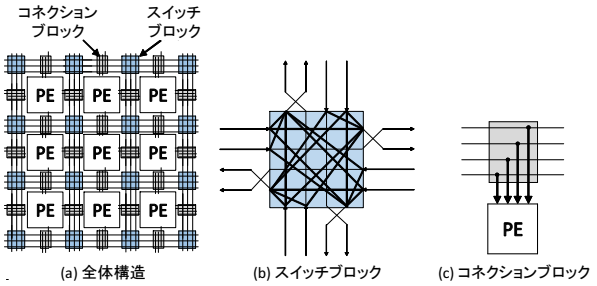

図 1　アイランド型配線構造

2. 配線構造の定義

先行研究 [3] では、DRP[4]、HSRA[5]、RaPiD[6] 等で採用されているアイランド構造（図 1）を対象として、[7] で提案されている片方向のバス型配線構造に基づいて配線リソースをパラメータ化した評価環境を構築している。アイランド型配線構造では、PE 同士を繋ぐ水平方向と垂直方向のバスが、コネクションブロックとスイッチブロックを介して接続される。この評価環境を用いて行った実験結果から、アプリケーションによっては、配線の方向によって使用率が大きくことなることがわかった。しかし、[3] ではスイッチブロックが上下左右対称な構造しか定義できないため、どの方向にも同じ配線リソースを持つことになり、配線パラメータで設定できる構造が限られていた。

そこで本研究では、各方向に対して配線リソースの設定が可能となるよう、配線パラメータの自由度を高めた配線構造の定義を提案する。提案する配線構造は、片方向バス配線の東西南北方向をそれぞれ East（E）、West（W）、South（S）、North（N）で表し、各方向において配線パラメータとして配線長 L と本数 x を定義する。また、配線長が 2（L2）以上のバイパス配線が複数本あるときは、スイッチブロックにおけるセレクタ（MUX）の間隔を設定するためのパラメータも定義する。たとえば、図 2 において、東方向は EL2x2(p0,p0)、西方向は WL2x2(p0,p1) という設定になる。また、隣接する PE 同士を直接接続する隣接配線の有無も定義することができる。隣接配線は、図 3 に示すように、スイッチブロックを経由することなく PE 間を接続することができる配線である。配線のファンアウトが小さくなるようなアプリケーションの場合、この隣接配線を使用して効率よく配線接続が可能になり、スイッチブロックを経由することによる遅延の増加を抑えることができる。この隣接接続の有無を H で定義し、H1 および H0 で隣接配線の有無を指定する。以上の定義に基づき、配線パラメータの設定例として、配線パラメータ「EL2x2(p0,p1), SL1x1, WL1x1, WL2x1(p0), H=1」のスイッチブロック間の接続例を図 4 に示す。

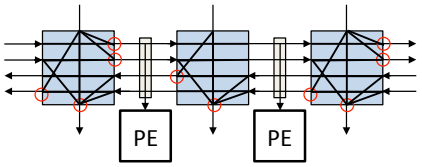

図 2　バイパス配線におけるスイッチブロック内のセレクタ配置間隔設定の例

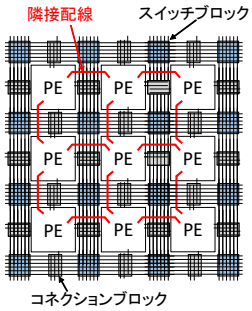

図 3　隣接 PE を直接接続する隣接配線

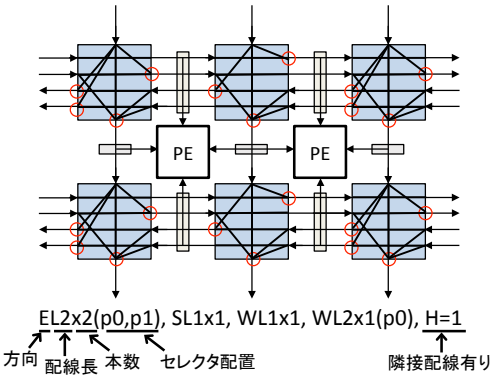

図 4　配線構造例とその配線パラメータ

3. 評価環境の構築

3.1 評価用ツールフロー

構築した評価環境を図 5 に示す。アプリケーションネットリストは、動作合成によって生成したものを使用する。ただし、本稿の評価実験では手動で作成したものを使用している。配線構造の種類と配線パラメータ、アレイサイズから、スクリプトによって配置ツール用にアーキテクチャ定義と遅延定義が自動生成される。配置ツールにより定義したアーキテクチャに対してアプリケーションネットリストを自動配置する。アーキテクチャのパラメータから配線ツールのアーキテクチャ・遅延定義関数を自動生成することで、配線ツール用のアーキテクチャと遅延を定義する。また、スクリプトにより配置ツール用から配線ツール用にアプリケーションネットリストを自動変換する。アーキテクチャ・遅延定義とアプリケーションネットリスト、配置結果から配線ツールによる自動配線を行う。自動配置配線結果から、アプリケーションマッピング時の実装面積やク

Vol.2016-ARC-219 No.26
Vol.2016-SLDM-175 No.26
Vol.2016-EMB-40 No.26
2016/3/25

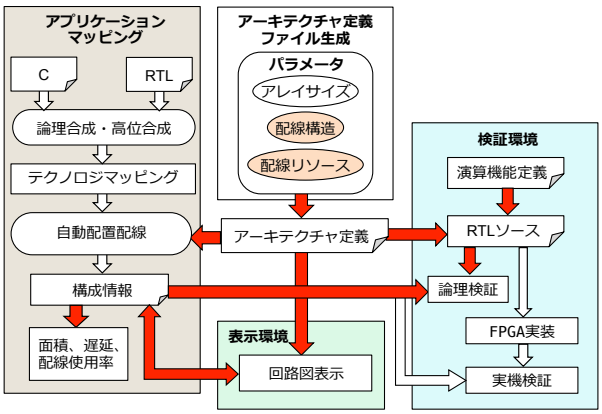

図 5　提案する評価環境の全体フロー

リティカルパス遅延を算出し、配線構造を定量的に評価すること可能である。再構成可能デバイス向けの自動配置配線ツールとして、VPR[8]、SPR[9] などが広く利用されている。本研究では、これらのツールと同様のアルゴリズムを用いて立命館大学が開発した自動配置配線ツールを用いている。また、自動生成したアーキテクチャ定義ファイルで実現される配線構造の回路図を表示することができ、アプリケーションマッピング結果をハイライト表示することも可能である。さらに、論理シミュレーションが可能な RTL ソースを自動生成することもできる。将来的には、その RTL ソースから FPGA プロトタイピングによる実機検証が可能な環境を構築する。

図 5 に示す評価環境の各機能については 、以下で詳しく述べる。

3.2　自動配置ツール

自動配置アルゴリズムは、Simulated Annealing (SA) 法のアルゴリズムを採用しており、アプリケーションマッピング時の総遅延をコストとしている。

自動配置ツール用のアーキテクチャ定義ファイルと遅延定義ファイルの PE の記述フォーマットを説明するために、図 6 に PE の構成例を示す。図 6 の PE を、配置ツール用のアーキテクチャ定義ファイルおよび遅延定義ファイルに記述した例を図 7 に示す。配置ツール用のアーキテクチャ定義ファイルは、Verilog-HDL に準拠したフォーマットである Verilog ネットリスト形式で記述され、アーキテクチャを階層的に記述できる。しかし、assign 文や always 文は使用できず、モジュール呼び出しのみによってアーキテクチャを定義する。配置ツールでは座標の概念がないため、座標情報を含んだインスタンス名をスクリプトにより生成する。配置ツールの遅延定義ファイルは、Standard Delay Format (SDF) に準拠し、各要素の入出力ポートの遅延と要素内部での遅延を設定可能である。

図 8 に示すアプリケーションのデータフローグラフ (Data Flow Graph: DFG) を定義する配置ツール用のネットリスト

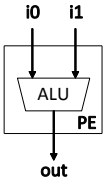

図 6　PE の構成例

```
module TOP_ARRAY( /*入出力信号*/ );
  PE PE_R00C00( /*入出力信号*/ );   //0行0列
  PE PE_R00C01( /*入出力信号*/ );   //0行1列
    :
endmodule

module PE( i0, i1, out );
  input i0, i1;
  output out;
  ALU ALU( .a(i0), .b(i1), .x(out) );
endmodule
```

(a) アーキテクチャ定義

```
(CELLTYPE "ALU")
  (PORT a 0.1)
  (PORT b 0.1)
  (IOPATH a x 0.1)
  (IOPATH b x 0.1)
  (DEVICE x1 1)
```

(b) 遅延定義

図 7　アーキテクチャ・遅延定義のフォーマット（配置）

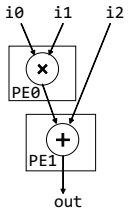

図 8　アプリケーションの例

```
module application ( i0, i1, i2, out );
  input i0, i1, i2;
  output out;
  wire net0;
  PE MLT ( .IN0(i0), .IN1(i1), OUT0(net0) );
  PE ADD ( .IN0(i2), .IN1(net0), OUT0(out) );
endmodule
```

図 9　アプリケーションネットリストのフォーマット（配置）

記述を 図 9 に示す。配置ツール用アプリケーションネットリストも、アーキテクチャ定義と同様のフォーマットで記述される。

3.3　自動配線ツール

自動配線ツールは negotiation ベースのアルゴリズムで、配置ツールと同様に総遅延をコストとする。自動配線ツールには、配置ツールのようなアーキテクチャ定義ファイルや遅延定義ファイルは存在せず、rrginit 関数によってアーキテクチャ定義と遅延定義を行う。図 6 の PE のアーキテクチャ定義と遅延定義を行う rrginit 関数の記述を図 10 に示す。rrginit 関数は、構造体 ws により配線を定義し、putsw 関数により各配線をスイッチで接続することでアーキテクチャを定義する。配線定義の構造体 ws のメンバを表 1 に示す。putsw 関数には、配線の種類、PE の x 座標、PE の y 座標、PE 内の識別番号を接続元の配線、接続先の配線の順に引数として渡す。

図 8 に示したアプリケーションの配線ツール用のネット

Vol.2016-ARC-219 No.26
Vol.2016-SLDM-175 No.26
Vol.2016-EMB-40 No.26
2016/3/25

```
void rrginit(){

  ws[0].type = input;                // i0の定義
  ws[0].x= 0;  ws[0].y = 0;
  ws[0].i= 0;  ws[0].w = 1;

  ws[1].type = input;                // i1の定義
  ws[1].x= 0;  ws[1].y = 0;
  ws[1].i= 1;  ws[1].w = 1;

  ws[2].type = output;               // outの定義
  ws[2].x= 0;  ws[2].y = 0;
  ws[2].i= 0;  ws[2].w = 10;

  putsw(input,0,0,0,output,0,0,0); // i0->out
  putsw(input,0,0,1,output,0,0,0); // i1->out

}
```

図 10　アーキテクチャ・遅延定義関数（配線）

表 1　配線定義のための構造体変数のメンバ

メンバ	概要
type	配線の種類（入力や出力）
x	PE の X 座標
y	PE の Y 座標
i	PE 内の 配線識別番号
w	遅延コスト

```
// 配線名 PE(source) fan-out数 PE(target1) PE(target2) ...
   i0 input 1 PE0.i0
   i1 input 1 PE0.i1
   i2 input 1 PE1.i1
   net0 PE0.out 1 PE1.in0
   out PE1.out 1 output
```

図 11　アプリケーションネットリストのフォーマット（配線）

リストの記述を図 11 に示す。配線識別番号 i は、PE 内に同じ種類の配線が複数存在する場合にそれらを識別するために定義される。

3.4　面積・遅延算出

本稿では、これまでは配線間の接続をスイッチとして説明してきたが、実際には MUX により実現する。MUX は入力数に応じて面積と遅延が決まる。アレイサイズと配線パラメータから、PE 内の MUX 数と各 MUX の面積と遅延が分かる。PE 内の MUX 数と各 MUX の面積からアレイ全体の面積を算出する。さらに、各 MUX の遅延と配置配線結果から得られるクリティカルパスの経由 MUX 数から、クリティカルパス遅延を算出する。また同様に、配置配線結果から各方向の配線使用率を算出する。

3.5　回路図表示

回路図表示機能として、図 7 に示すアーキテクチャ定義ファイルから、配線構造の回路図を表示することが可能である。処理の流れとしてはまず、アーキテクチャ定義ファイルを Ruby スクリプトで解析し、アーキテクチャ定義記述から回路表示に必要な情報を抽出する。この情報を、表示用プログラムに対応した形式のファイルとして自動生成する。表示用プログラムは、開発プラットフォームの依存性をなくすため、Java を用いた。起動時にはアレイの全体構成を表示する。各スイッチブロックや PE を選択することで、各内部の回路構成を表示することができる。

さらに、自動配置配線で得られた構成情報を取り込むことで、図 12 に示すように、マッピングに使用される配線リソースがハイライトされるようになっている。

この画面上で、使用配線の変更操作によって構成情報を編集できる機能を現在開発中である。

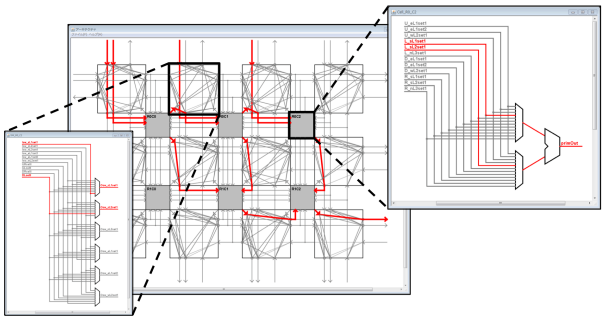

図 12　回路図表示

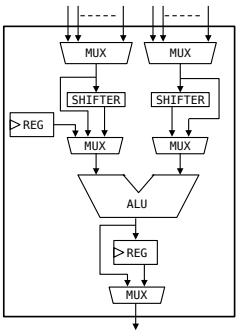

図 13　評価実験に用いた PE の構成

3.6　論理検証環境

図 7 に示すアーキテクチャ定義ファイルは、入出力宣言やモジュール呼び出しの記述しかなく、PE 内部の演算機能に関する情報がない。そこで、アーキテクチャ定義ファイルに PE の演算機能を定義する Verilog-HDL 記述を追加した RTL ソースを自動生成するプログラムを作成した。この自動生成した再構成可能アーキテクチャの RTL ソースと、アプリケーションマッピング結果の構成情報を用いて、一般的なシミュレーションツールを用いた論理検証を行うことが可能となっている。

4.　評価実験

4.1　実験条件

本評価実験で対象としている CGRA の PE の構成を図 13 に示す。PE は、入出力 MUX、ALU、シフタのほか、定数保持用レジスタとパイプライン用レジスタを持つ。2 入力 1 出力 ALU は、加減算と乗算を選択できる。アレイサイズ（PE 数）は、後述する評価用アプリケーションの規模から 8×8 とする。

Vol.2016-ARC-219 No.26
Vol.2016-SLDM-175 No.26
Vol.2016-EMB-40 No.26
2016/3/25

表 2　配線パラメータとチップ面積

	配線リソース	配線パラメータ	面積 $[mm^2]$
(1)	少	EL2x2, SL2x4, WL2x2, H1	0.464
(2)	中	NL2x2, EL2x2, SL2x4, WL2x2, H1	0.510
(3)	多	NL2x4, EL2x4, SL2x8, WL2x4, H1	0.818

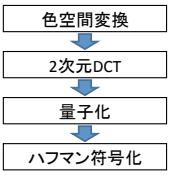

図 14　JPEG エンコーダの処理フロー

評価用アプリケーションとして、JPEG エンコーダ [10] を用い、JPEG エンコーダに含まれる複数の処理を時分割実行することを考える。JPEG エンコーダおよびその各処理のマッピングについては、4.2〜4.5 章で述べる。

アレイ上へのアプリケーションマッピングでは、アレイ北側のポートからデータを入力し、南側のポートから出力する。SA の初期温度を 50 度、減少刻み温度を 0.1 度とし、各配線パラメータとアプリケーションに対して配置配線を 30 回行い、その中で最もクリティカルパス遅延が小さくマッピングできた時の配置配線結果を採用し、各方向の配線使用率を評価する。

本実験では、3 段階の配線リソースで評価を行うため、3 種類の配線パラメータを用意した。また、各配線パラメータで定義される配線構造について、65nm プロセスで設計したときのチップ面積を表 2 に示す。表 2 では省略しているが、セレクタ配置は等間隔交互になるように定義した。

4.2　JPEG エンコーダ

JPEG エンコーダは、8×8 画素のブロック単位で図 14 に示す処理を行うことにより、画像の圧縮を行う [10]。まず、各画素について、3 原色で表現された RGB 信号を、輝度成分と色差成分で表現する YCbCr へ色空間変換を行う。次にブロック単位で 2 次元 DCT を行うことで、各画素値の周波数データである DCT 係数へ変換する。2 次元 DCT は、1 次元 DCT を 2 回施すことによって実行する。その後、量子化を行い、DCT 係数の精度を削減する。最後にハフマン符号化を行い、出現確率の高いビットの並びを短い符号で表現することでさらにデータ量を削減する。

本評価実験では、ハフマン符号化は CGRA には不向きであると判断し、細粒度アレイもしくはプロセッサで行うものとして、今回のマッピング対象外とした。また、2 次元 DCT において、1 次元 DCT を行った後の転置処理ならびにジグザグスキャンについては、メモリアクセス時のアドレスをあらかじめメモリに格納しておくことにより実現するものとした。

表 2 示す、3 段階の配線リソースを定義する配線パラメータ「EL2x2, SL2x4, WL2x2, H1」、「NL2x2, EL2x2, SL2x4, WL2x2, H1」、「NL2x4, EL2x4, SL2x8, WL2x4, H1」を用いて、JPEG エンコーダの各処理をマッピングし、クリティカルパス遅延と配線使用率を自動算出し、評価を行った。

表 3　評価用アプリケーションのマッピングに必要な演算器数

処理	演算器数
色空間変換	7
量子化	2
1 次元 DCT	46

```
Y = (66R + 129G + 25B + 4096) >> 8
Cb = (−38R − 74G + 112B +32768) >> 8
Cr = (112R − 94R − 18B + 32768) >> 8
```

図 15　RGB から YCbCr への変換式

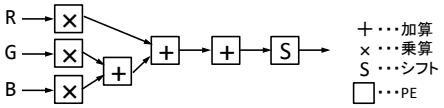

図 16　RGB-YCbCr 変換のアプリケーションネットリスト

表 4　実験結果（RGB-YCbCr 変換処理）

	配線リソース	最大遅延 [ns]	配線使用率 N	E	S	W
(1)	少	4.432	0/0	27/81	93/162	22/81
(2)	中	4.522	2/81	16/81	86/162	184/81
(3)	多	3.688	6/162	30/162	79/324	31/162

JPEG エンコーダにおける各処理のマッピングに必要な演算器数を表 3 に示す。1 ブロックの画素データに対して必要な各処理の実行回数とアレイサイズ（演算器数）から、各処理のマッピングにおいて適切な並列度を考慮する。

4.3　色空間変換処理のマッピング

RGB から YCbCr への色空間変換は、図 15 に示す式を用いた。色空間変換処理には 7 個の PE を使用するため、アレイの PE 数を考慮して 8 並列でマッピングすることを考える。すなわち、マッピングに必要な PE 数は合計 56 個となる。色空間変換処理のアプリケーションネットリストを図 16 に示す。また、評価結果を表 4 に示す。

4.4　1 次元 DCT のマッピング

1 次元 DCT は Chen のアルゴリズム [11] を用いる。1 次元 DCT のマッピングには 46 個の PE を使用するため、アレイの PE 数を考慮し DCT 処理単位の並列は行わない。1 次元 DCT のアプリケーションネットリストを図 17 に示す。また、配置配線結果を表 5 に示す。

4.5　量子化処理のマッピング

量子化処理は 2 個の PE で実現できる、アレイの PE 数を考慮し 22 並列でマッピングすることを考える。すなわち、マッピングに必要な PE 数は合計 44 個となる。アプリ

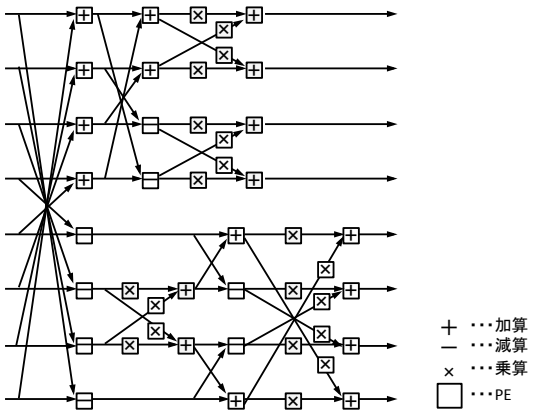

図 17 1 次元 DCT のアプリケーションネットリスト

表 5 実験結果（1 次元 DCT）

	配線リソース	最大遅延 [ns]	配線使用率 N	E	S	W
(1)	少	3.290	0/0	18/81	45/162	21/81
(2)	中	2.647	4/81	10/81	45/162	14/81
(3)	多	3.071	2/162	16/162	42/324	15/162

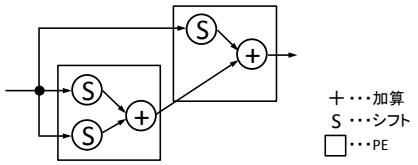

図 18 量子化処理のアプリケーションネットリスト

表 6 実験結果（量子化処理）

	配線リソース	最大遅延 [ns]	配線使用率 N	E	S	W
(1)	少	4.432	0/0	24/81	124/162	32/81
(2)	中	3.265	6/81	16/81	121/162	14/81
(3)	多	2.982	2/162	8/162	128/324	15/162

ケーションネットリストを図 18 に示す。また、評価結果を表 6 に示す。

4.6 考察

実験結果から、配線リソースの差によってチップ面積が大きく異なるが、配線リソースが多い場合はクリティカルパス遅延の大幅な削減が可能であることがわかった。しかし一部の結果において、配線リソースとクリティカルパスのトレードオフが逆転するケースが見られた。これは、配線リソースだけでなく、各方向の配線数比率やバイパス配線数などによるものと考えられる。

5. まとめ

先行研究で課題となっていた配線構造の自由度向上を実現する配線パラメータを提案し、自動配置配線向けのアーキテクチャ定義ファイルの自動生成環境を構築した。さらに、配線パラメータ（アーキテクチャ定義ファイル）で定義される評価対象配線アーキテクチャの回路図表示機能、ならびにアプリケーションマッピング結果の表示機能を実現した。また、シミュレーションツールを用いた論理検証が可能な RTL ソースの自動生成機能も実現した。

本評価環境を用いて、配線リソースの異なる 3 種類の配線構造に対して、JPEG エンコーダにおける色空間変換、DCT、量子化の 3 処理をマッピングした。これにより、様々な再構成可能配線構造において、面積、クリティカルパス遅延、配線使用率を評価できることを示した。

本評価環境を用いて得られる情報を基に、適切に配線パラメータを変更し評価を繰り返すことで、ターゲットアプリケーションドメインに対し、処理速度や面積効率などの最適化要件を満たす配線構造の探索が可能であると考える。

謝辞 本研究は一部、越智裕之教授（立命館大学/JST-CREST）の開発による配置配線ツールを使用して行った。

参考文献

[1] W. Mark Freg and S. Kaptanoglu, “Designing efficient input inter-connect blocks for LUT clusters using counting and entropy,” in Proc. *International Symposium on Field-Programmable Gate Arrays(FPGA)*, pp.23-32, Feb. 2007.

[2] Mike Sheng and Jonathan Rose, “Mixing Buffers and Pass Transistors in FPGA Routing Architectures,” in Proc. *International Symposium on Field-Programmable Gate Arrays(FPGA)* pp. 75-84, Feb. 2001.

[3] 森岡俊樹, 山本浩平, 密山幸男, “再構成可能配線構造検討のための性能評価環境の構築,” 情報処理学会研究報告, 2015-SLDM-170, 2015 年 3 月.

[4] M. Motomura, “A Dynamically Reconfigurable Processor Architecture,” Microprocessor Forum, Oct. 2002.

[5] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon “HSRA: high-speed, hierarchical synchronous reconfigurable array,” in Proc. *International Symposium on Field-Programmable Gate Arrays(FPGA)*, pp. 125-134, Feb. 1999.

[6] C. Ebeling, D. C. Cronquist, and Paul Franklin, “RaPiD - Reconfigurable Pipelined Datapath,” in Proc. *International Conference on Field Programmable Logic and Applications (FPL)*, pp 126-135, Sep. 1996.

[7] G. Lemieux, E. Lee, M. Tom, and A. Yu “Directional and single-driver wires in FPGA interconnect,” in Proc. *International Conference on Field Programmable Technology(ICFPT)*, pp. 41-48, .Dec. 2004.

[8] V. Betz and J. Rose “VPR: A New Packing, Placement and Routing Tool for FPGA Research,” in Proc. *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 213-222, Sept. 1997.

[9] S. Friedman, A. Carroll, B. V. Essen, B. Ylvisaker, C. Ebeling, and S. Hauck “SPR: an architecture-adaptive CGRA mapping tool,” in Proc. *International Symposium on Field-Programmable Gate Arrays(FPGA)*, pp. 191-200, Feb. 2009.

[10] 古島直道, 渡邊誠也, 名古屋彰, “動的再構成可能プロセッサへの JPEG エンコーダの実装と評価,” 電子情報通信学会技術研究報告, RECONF2008-24, pp. 7-12, 2008 年 9 月.

[11] W. H. Chen, C. H. Smith, and S. C. Fralick “A Fast Computational Algorithm for the Discrete Cosine Transform,” IEEE Transactions on Communication, vol. 25, pp. 1024-1009, Sept. 1977.