システムＬＳＩ設計技術　99－3
（2001．1．11）

マルチコンテキストFPGAのための コンテキスト分割アルゴリズムの実例による評価

本多 亮† 北道 淳司‡ 船曳 信生† 東野 輝夫†

大阪大学大学院 基礎工学研究科 / 大阪大学 サイバーメディアセンター / 大阪大学大学院 基礎工学研究科 / 大阪大学大学院 基礎工学研究科

〒 560-8531 豊中市待兼山町 1-3
TEL:(06)6850-6593 TEL:(06)6850-6072 TEL:(06)6850-6591 TEL:(06)6850-6590
† {r-honda,funabiki,higashino}@ics.es.osaka-u.ac.jp
‡ kitamiti@ecs.cmc.osaka-u.ac.jp

あらまし マルチコンテキストFPGAは，1回路素子に対する複数の設定をFPGA内に格納でき，FPGA上の回路構成を高速に変更できる．また，従来型FPGAのための設定データに対する合成において行われる分割・配置・配線の工程に加え，実装する回路をFPGAサイズを上限として分割する問題を考慮する必要がある．この問題に対し，我々は入力回路モデルをEFSMとし，制御部の1状態で行われるべき演算を単位としてグループ化し，回路を分割する方法を提案した．提案解法は二段階の近似解法を組み合わせたアルゴリズムである．第一段階では入力EFSMを深さ優先探索の走査順に状態集合にグループ化して初期解を生成し，第二段階では初期解で得られた分割結果を基に状態集合間で状態の移動を行い，コンテキストを書き換える上で時間の最も余裕がない状態遷移経路を改善する．FPGA上の回路構成の変更はグループ化された制御部状態の演算回路群を単位として行われる．本研究ではこの問題を解く解法を，いくつかの具体例に対して適用し，その結果について報告する．プログラムをハードウェア化した例題に対しアルゴリズムを適用し，対象FPGAの回路容量を変化させ試行を行った結果，改善段階での処理により初期解より平均30％程度解が改善された．

キーワード FPGA，動的再構成構造，マルチコンテキスト，回路分割，組み合わせ最適化問題

An Evaluation of a Context Partitioning Algorithm for Multi-Context FPGAs

Ryo HONDA† Junji KITAMICHI‡ Nobuo FUNABIKI† Teruo HIGASHINO†

Graduate School of Engineering Science, Osaka University / Cybermedia Center, Osaka University / Graduate School of Engineering Science, Osaka University / Graduate School of Engineering Science, Osaka University

1-3, Machikaneyama, Toyonaka, Osaka, 560-8531 Japan
TEL:+81-6-6850-6593 TEL:+81-6-6850-6072 TEL:+81-6-6850-6591 TEL:+81-6-6850-6590
† {r-honda,funabiki,higashino}@ics.es.osaka-u.ac.jp
‡ kitamiti@ecs.cmc.osaka-u.ac.jp

Abstract A multi-context FPGA is a kind of Dynamically Reconfigurable FPGA(DRFPGA) with better reconfiguration time-efficiency than conventional DRFPGA. In synthesis for a multi-context FPGA, we consider the context partitioning probrem. The context partitioning probrem is a step of a multi-context FPGA design flow, in which a given circuit is packed into reconfiguration units, called contexts, of a multi-context FPGA so that the reconfiguration time over head can be small. We have proposed a heuristic algorithm to solve the context partitioning probrem. In this paper, we apply our algorithm to some real examples and evaluate our algorithm.

Key words FPGA, Dynamically Reconfigurable Logic, Multi-Context, Partitioning, Combinatorial Optimization

1 まえがき

FPGA(Field Programmable Gate Array)はユーザが回路動作をプログラムすることが可能なデバイスであり，システム開発時のプロトタイピング用デバイスなどに用いられていた．近年ではFPGA上のアプリケーション動作中に回路の設定情報を再構成する機能を有する動的再構成可能FPGA(Dynamically Reconfigurable FPGA)およびそのアーキテクチャに適した利用法が提案されている[1] [2]．動的な再構成を行うにはその都度FPGA外部から設定情報を書き込む必要があり，この時間が実行の際の大きなオーバヘッドであることが問題となる[3]．これに対しマルチコンテキストFPGAはFPGAの各素子の設定に対し複数の設定をFPGA内部に保持することが可能であり，設定の選択信号により全設定を切替えることで高速に回路構成を変更することができる．マルチコンテキストFPGAのアーキテクチャとしてはさまざまなものが提案されている[4] [5] [6] [7] [8]．我々はマルチコンテキストFPGAに着目し，提案するアーキテクチャ[9]に対して設定データの合成アルゴリズムを提案し，その評価を行っている．マルチコンテキストFPGAに対する回路合成フローでは，従来型FPGAの合成アルゴリズム分割，配置，配線の3工程に加えて，実現する回路をFPGAの容量を上限とする部分回路に分割する問題を考慮する必要があると考える．我々はこの問題を解く二段階の近似アルゴリズムを提案し，アルゴリズミックに生成した例題に対し適用した際の評価を行った[10]．その結果，アルゴリズムの改善段階での処理により初期解より20％程度改善した解を得られることがわかった．本論文では実際的な例題に対しこのアルゴリズムを適用した際の実行結果について報告する．プログラムをハードウェア化した2つの例題に対してコンテキスト分割アルゴリズムを適用した結果，改善段階での処理により初期解より平均30％程度解が改善された．またFPGAの回路容量を変化させ試行した際に初期解から2倍近く改善されたも見られた．

以下，2章では実装対象とするFPGAについて述べる．3章では対象FPGAの回路合成フローにおけるコンテキスト分割問題を定式化する．4章では3章のコンテキスト分割問題を解く近似アルゴリズムについて述べる．5章では実際的な例題に対し4章の近似アルゴリズムを適用した際の実行結果について述べ，提案解法の評価を行う．

2 マルチコンテキストFPGA

本研究で対象とするマルチコンテキストFPGA[9]の概略を図1に示す．マルチコンテキストFPGAはFPGA外部との入出力を行うI/Oブロック，ロジックブロック，ロジックブロックの周囲のグローバルバスからなる．ロジックブロックでは小規模な演算が実行可能であり，Look Up Table(LUT)とFlipFlop(FF)，およびロジックブロック内で行われる動作の設定を記憶する設定用記憶(Configuration Memory)で構成されている．グローバルバスは任意のI/Oブロックおよびロジックブロック間の入出力信号の送受信を行う．バスの交点にはスイッチブロックが設けられており，交差するバス間の接続の開閉の設定を記憶する設定用記憶が用意され，その出力によりプログラムされた配線構造を電気的に実現する．

ロジックブロックおよびスイッチブロックはそれぞれ設定用記憶を複数持ち，制御信号により1つの設定を選択することにより，回路上の動作を決定する．各設定用記憶群には同一の制御信号(Context Number)が与えられ，信号が切替えられると回路上の全設定が瞬時に変更する．このとき同時に選択される設定群をコンテキストと呼ぶ[1]．

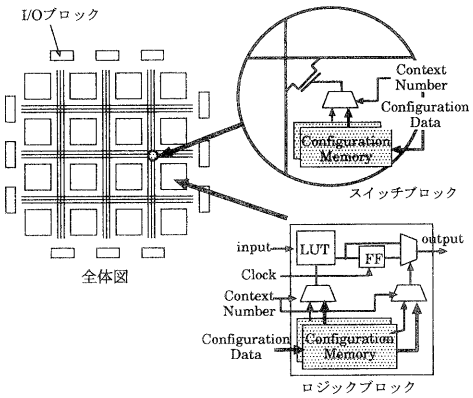

図1: マルチコンテキストFPGAの概略

設定を選択する制御信号が切替えられるとコンテキストが切替えられる．この際，各ロジックブロック内のLUTでの演算および各スイッチブロックにおけるバスの接続の設定が変更されるが，FFに格納されていた値は保持されている．そのためFFに格納された値は，コンテキスト切替えの際にFPGA外に退避させることなく，新たに構成された回路に対する入力として利用することができる．

また各素子に用意されている複数の設定用記憶のうち，制御信号で指定されていない設定用記憶に対して動作と並行して外部から動作の設定情報(Configuration Data)を書き換えることができる(図2)．これによりFPGA内の設定情報を適宜書き換え，コンテキストを

[1] 3章でのコンテキスト分割問題では，図1のアーキテクチャにのみ対応しているというわけではなく，ある程度一般化している．

切替えることで，全設定をFPGA内に保持できないような大規模なアプリケーションを実行することができる．アプリケーション動作におけるクロックと設定情報を書き込むためのクロックの速度には差があるため，制御信号で選択されているコンテキストでの実行が終了し次に実行されるコンテキストに切替えるまでに必要な設定情報の書き込みが完了しない場合がある．この場合設定情報の書き込みが完了するまでアプリケーション動作を停止（ストール）させなければならない．そのためアプリケーション動作と並行して設定情報を書き換えられる機能を利用し，FPGA外部で保持されている設定情報が後の実行に必要であると判明した時点でできるだけ早期に書き込みを開始することでストール時間を短くし，全体の実行を早く完了させることが可能となる．

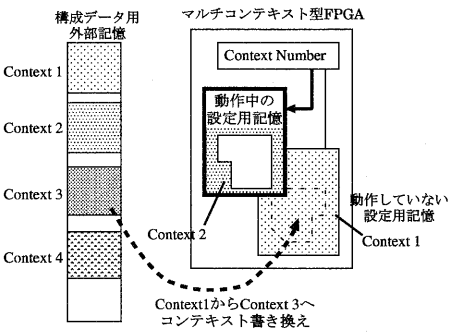

図 2: 設定情報の書き換え

3 コンテキスト分割問題

動的再構成機能を持たない従来型FPGAにおける回路合成では，実装する回路をネットリストとして入力し，分割，配置，配線の工程を経てFPGAに設定される設定情報を出力する．分割工程では入力回路をFPGAの1ロジックブロックで実現可能な大きさの部分回路に分割する．配置工程では分割工程で分割した各部分回路のFPGA上での配置箇所を決定する．配線工程では配置された部分回路間の配線を決定する．マルチコンテキストFPGAにおける回路合成でも従来型FPGAと同様の合成フローを利用できると考えられる．ただしマルチコンテキストFPGAでは上記の3工程に加えて，入力されたアプリケーション回路を1コンテキストで実現可能な大きさの部分回路に分割するコンテキスト分割の工程を考慮する必要がある．コンテキスト分割は分割工程の次の段階で行うものとする．よって本コンテキスト分割問題では，入力回路はロジックブロックを単位とする大きさで表現されている．さらにコンテキスト分割によって分割された各部分回路に対し，個々に配置，配線の工程を実行することで各コンテキストに対する設定情報を得ることができる．

分割，配置，配線の各アルゴリズムは従来型FPGAの合成フローで利用される手法を適用可能であると考えられ，本論文ではコンテキスト分割問題とその解法についてのみ述べる．

3.1 入力

コンテキスト分割問題の入力は，実装するアプリケーションを表現する拡張有限状態機械 (EFSM) およびFPGAのサイズである．(以下Aのサイズとは Aを実装するのに必要なロジックブロック数とする.) EFSMは回路の演算結果を保持するデータレジスタ群と有限状態機械で構成される．入力されるEFSMの詳細は状態機械の状態遷移図，各状態で行われるレジスタ転送や遷移条件判定のための演算回路のサイズ，データレジスタ群のサイズ，状態遷移の各分岐の分岐確率である．各状態の演算回路やデータレジスタのサイズとは，それらがFPGA上で占める領域をロジックブロック数で表現したものである．FPGAのサイズとは，対象とするマルチコンテキストFPGAの1コンテキストで実現可能な回路量のことであり，FPGAが保持する全ロジックブロック数とする．

コンテキスト分割問題の入力としては以下のものが与えられる．

- EFSMの状態 $S = \{s_1, s_2, ..., s_n\}$
- EFSMの初期状態 s_{init} $(s_{init} \in S)$
- 二つの状態間の遷移関係
  $R = \{(s_i, s_j) | 1 \leq i, j \leq n\}$
- 状態遷移の遷移確率
  $P = \{p(s_i, s_j) | (s_i, s_j) \in R,\ 1 \leq i, j \leq n\}$
- 状態 $s_i (\in S)$ における演算回路（状態遷移の条件判定およびレジスタ代入）を実現するのに必要とするロジックブロック数 $size(s_i)$
- EFSMのデータレジスタ群を実現するのに必要とするロジックブロック数 $size(REGs)$
- FPGA全体のロジックブロック数 $Allsize$（定数）

3.2 出力

出力はEFSMの各状態で行う演算をどのコンテキストで実現するか決定し，それらをグループ化した結果である．このときグループ化された1状態集合に含まれる状態の各演算を1コンテキストで実行するものとする．

コンテキスト分割問題の出力として以下のものを得る．

- EFSMの状態集合
  $C = \{C_1, C_2, ..., C_m\}$ $(C_j = \{s_{j_1}, s_{j_2}, ..., s_{j_n}\})$

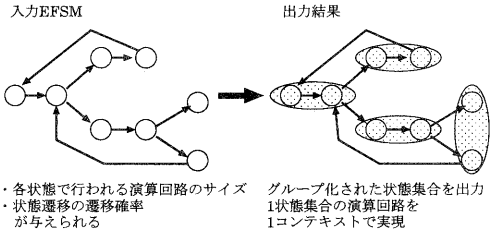

図 3: 入力 EFSM と出力結果

図3は入力EFSMとそれに対するコンテキスト分割の出力結果を示している．入力されたEFSMの各状態の演算を，同一のコンテキストで実現するものごとに，状態を単位として分割した結果が出力となる．

3.3 コンテキスト分割問題における前提

ここでコンテキスト分割問題の定式化における前提について述べる．

（i） コンテキストは静的に分割されるものとし，各コンテキストに割り付けられているEFSMの状態の演算回路の構成をアプリケーション動作中に動的に変更および再構成することはない．すなわちアプリケーション動作中の回路の再構成はコンテキストを交換することでのみ行われる．

（ii） 1状態の演算回路は必ずいずれかのコンテキストに割り付けられ，複数のコンテキストに重複して割り付けられたり，1状態を分割して別々のコンテキストに割り付けることはない．

（iii） 次に実行されるコンテキストが，以前に一度実行されまだFPGA内の設定記憶に保持されている場合，制御信号を切替えるだけでコンテキストが変更できコンテキストの変更の際に必ず外部から設定情報の書き換えが行われるわけではない．しかしコンテキスト分割を行う際にアプリケーション動作時にFPGA内のコンテキストがどのようにスワップイン・アウトされるかを予測することは困難なので，コンテキストが変更される際には書き換えが起こることを想定して分割を行うこととする．

（iv） 1コンテキストにはできるだけ多くの演算回路を実装する．その際，コンテキスト分割以降に行われる配置・配線が可能であるかどうかについては議論しない．例えば実装対象とするアーキテクチャで，配置・配線が可能であると平均的に見込めるロジックブロック数をFPGAのサイズとする．

（v） データレジスタは常にFPGA内に保持され参照可能であるように，各コンテキストにそのための領域を確保する．

（vi） データレジスタを実現するのに使用するロジックブロックでは，そのLUT部分を各コンテキストでの組み合わせ回路の演算に使用しないものとする．

（vii） EFSMの1状態で行われる演算は，FPGAに与えられる動作クロック1周期の時間内に処理される．

3.4 制約条件

状態 s_i の演算回路のサイズは $size(s_i)$ で与えられている．グループ化された1状態集合に含まれる状態の各演算は1コンテキストで実行する．また，対象FPGAでは各コンテキストにレジスタの位置を固定して配置し，レジスタに接続された論理構造の動作を変更することにより，複数のコンテキスト間で同一のレジスタに対して読み書き可能であるように実装する．求めるべき分割は各状態集合 C_j に対し以下の制約条件式(1)を満たす．

$$\sum_{s_i \in C_j} size(s_i) + size(REGs) \leq Allsize \qquad (1)$$

$\sum_{s_j \in C_i} size(s_j)$ は状態集合 C_i にグループ化された各状態で行われる演算回路を構成するのに必要な総ロジックブロック数である．

3.5 目的条件

目的条件はコンテキスト変更時のオーバヘッドを最小化することである．FPGA 内にすでに保持されているコンテキストへの変更は，設定の選択を行う制御信号を切替えることで瞬時に行うことができる．一方FPGA内に保持されていないコンテキストへ変更する際には外部から設定情報を書き換えた後に制御信号を切替える．このとき制御信号を切替えるまでに外部からの書き換えが完了していない場合，書き換えが完了するまでアプリケーション動作を停止（ストール）させる必要がある．このストール時間が大きなオーバヘッドとなる．マルチコンテキストFPGAでは制御信号で選択されていない設定用記憶に対してアプリケーション動作と並行して外部から設定情報の書き込みが行えるので，次に実行されるコンテキストを早い時点で判定できれば，書き込みを早く開始することができオーバヘッドを軽減することができる．1コンテキスト分の設定情報を書き込む時間は一定とすると，次に実行されるコンテキストを早く判定できた分だけ全体の実行完了を早めることになる．従って次の実行コンテキスト判明からコンテキスト切替えまでの時間を最大化するよう状態のグループ化を行うことが目的条件であるとみなせる．状態遷移の分岐からコンテキスト変更までの状態遷移経路長と分岐確率および各状態の状態

遷移の到達回数を利用し，次の実行コンテキスト判明からコンテキスト切替えまでの状態遷移経路長の期待値を得ると，前提(vii)よりその経路長に対する実行時間が得られる．本研究ではグループ化された各状態集合ごとに，次の実行コンテキスト判明からコンテキスト切替えまでの状態遷移経路長の期待値を求め，各状態集合での和を目的関数とし実行時のコンテキスト先読み時間を近似する．

状態遷移の遷移確率 $P = \{p(s_i, s_j) | 1 \leq i, j \leq n\}$ は問題の入力として与えられている．また状態 s_i における状態遷移の到達回数 $a(s_i)$ は，遷移確率を基に状態遷移図上を走査することで算出する．さらに状態集合 C_i において，状態遷移の分岐からコンテキスト変更までの状態遷移経路を $r_{i \cdot m} = (s_{i \cdot m_1}, s_{i \cdot m_2}, ..., s_{i \cdot m_l})$ とすると，$r_{i \cdot m}$ の遷移経路を状態遷移する確率は

$$prob(r_{i \cdot m}) = a(s_{i \cdot m_1}) \times \prod_j p(s_{i \cdot m_j}, s_{i \cdot m_{j+1}}) \quad (2)$$

となる．前提(ii)より C_i において次に実行するコンテキストが判明してからコンテキスト切替えまでの状態遷移経路長の期待値は式(2)によって重み付けを行うことで以下の式(3)のように表される．

$$obj(C_i) = \frac{\sum_m (prob(r_{i \cdot m}) \times |r_{i \cdot m}|)}{\sum_m prob(r_{i \cdot m})} \quad (3)$$

各コンテキストにおいて式(3)の値を算出し，それらの和を目的関数値とする．目的関数値は以下の式(4)で表される．

$$\sum_{C_j \in C} obj(C_j) \quad (4)$$

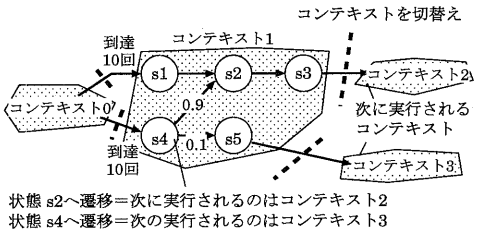

図4: コンテキスト1における目的関数値の算出

図4でコンテキスト1における目的関数値を算出する場合を考える．コンテキスト1の次に実行されるコンテキストの候補はコンテキスト2およびコンテキスト3である．状態s1へ状態遷移が行われた場合，その時点でコンテキスト1の次にコンテキスト2が実行されると判明し，コンテキスト2に切替えるまでの遷移経路の長さは3となる．状態s4へ状態遷移した場合，さらにs4から状態遷移の分岐が存在するため，この分岐でどちらに状態遷移が行われるかが判明した時点で次に実行されるコンテキストが決定される．s4からs2へ遷移した場合はコンテキスト2に切替えるまでの遷移経路の長さは2となり，s4からs5へ遷移した場合はコンテキスト3に切替えるまでの遷移経路の長さは1とみなす．また，ここではs4からs2へ遷移する確率が0.9，s4からs5へ遷移する確率が0.1と与えられるものとする．さらに各状態の状態遷移の到達回数が分岐確率を基にして与えられ，ここでは状態s1に10回，状態s4に10回，状態遷移が到達するものとする．このとき式(3)より，コンテキスト1の目的関数値は以下の式(5)で求められる．

$$\begin{aligned} & obj(C_1) \\ &= \frac{10 \times 3 + (10 \times 0.9) \times 2 + (10 \times 0.1) \times 1}{10 + 10 \times 0.9 + 10 \times 0.1} \\ &= 2.45 \end{aligned} \quad (5)$$

このような計算を各コンテキストについて行い，得られた値の和を目的関数値とする．

4 コンテキスト分割アルゴリズム

本章ではコンテキスト分割問題に対する近似アルゴリズムを述べる．このアルゴリズムでは前処理として，状態遷移図上の各分岐における分岐確率を基に各状態の状態遷移の到達回数を計算する．またコンテキスト分割の工程は二段階の処理で構成され，第一段階では初期解を生成し，第二段階では初期解を基に解の改良を行う．

このアルゴリズムは実装対象とするFPGAのロジックブロックの論理規模やバスの配線領域などの内部構造を意識せずに適用することができる．

コンテキスト分割の第一段階ではGreedyアルゴリズムを用い，入力されたEFSMとFPGAサイズに対するコンテキスト分割の初期解を生成する．EFSMの状態遷移図上を初期状態から深さ優先で走査し，走査した状態順にコンテキストの空き容量が許す限り同一の状態集合にグループ化する．新たにグループ化しようとした状態の演算回路のサイズがコンテキストの空き容量を上回った時は，新たな状態集合を用意しそこに状態をグループ化する．また分岐のある状態からは，分岐確率の高い方の状態を先に走査する．これは実行頻度の高い遷移経路上の状態は同じコンテキスト内に割り付けることが目的関数を良くすると考えたためである．EFSMの状態遷移図上の全状態に対して以上の手順で状態のグループ化を行い，その結果を初期解とする．

コンテキスト分割の第二段階では，第一段階で得られた初期解を基に以下の処理によって分割結果の改善を行う．

(I) 分割された各状態集合から1つを選択する．状態遷移の分岐からコンテキスト切替えまでの状態

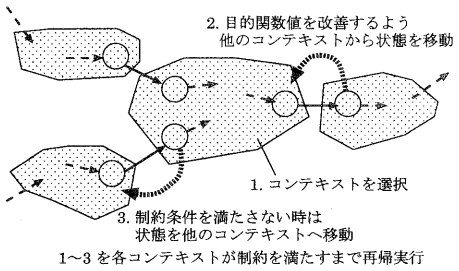

図 5: 第二段階での再帰実行の流れ

遷移経路が短い状態集合が優先的に選択される．

（II） 選択された状態集合の分岐からコンテキスト切替えまでの状態遷移経路が長くなるよう他の状態集合から 1 状態を移動させる．移動させる候補が複数ある場合は，遷移確率の高い遷移先にある状態を選択する．

（III） 状態の移動により状態集合の構成が変化しコンテキストに割り付けされる回路量が変化するので制約条件の判定を行う．制約条件を満たさずFPGA サイズを越えた場合，他の状態集合へ状態をさらに移動させる．状態を移動させた先の状態集合においても再帰的に制約条件の判定，状態の移動を行う．この再帰処理は各状態集合が制約条件を満たした時点で収束する (図 5)．また，この再帰処理は無限ループに陥ってしまう可能性があるので一定回移動を行った状態集合には，以降その試行中には状態の移動を行わないものとする．すべての状態集合で一定回の移動が行われた時，制約条件を満たさない場合は新たな状態集合を設け，そこに状態を移動させる．

（IV） 得られた分割結果に対し目的関数値の算出を行う．(I)〜(III) の処理を行う以前の分割結果における目的関数値と比較し改善が見られた場合は，得られた分割結果を新たな解とし，再び (I) の処理を行う．改善が見られなかった場合は，元の分割結果に戻し，(I) の処理を行う．(I) の処理では前回選択したものと別の候補を選択し実行する．

これら (I)〜(IV) の処理を繰返し，改善が見られなくなった時点で第二段階の処理は終了し，そのとき得られた分割結果がアルゴリズムの出力となる．(I)〜(IV) の処理を繰り返す回数は与えられた入力例題の規模に応じて変更する必要がある．分割された状態集合間をまたぐような状態遷移の数だけ (I)〜(IV) の処理の繰返しが行われれば十分であると考えられる．

5 コンテキスト分割アルゴリズムの例題への適用

本章では実際的な例題に対し 4 章で述べたコンテキスト分割アルゴリズムを適用し，その結果について報告する．今回試行を行った例題は，基数ソートにより入力された要素を昇順に並び替えるプログラムおよびエラトステネスのアルゴリズムを用いて素数の列挙を行うプログラムである．これらのプログラムを回路記述言語 SFL[11] を用いて記述し，回路合成ツール PARTHENON および MAX+PLUS II[12] を利用して，ハードウェア化した際のデータレジスタ群および状態遷移図内の各状態で行う演算回路の占めるロジックブロック数を得た．また PARTHENON システムの SFL 動作シミュレータである SECONDS を利用し，SFL で記述されたハードウェアの動作をシミュレートすることで状態遷移図の遷移の各分岐における分岐確率を得た．

5.1 ソート

ソートプログラムは 256 個の要素を基数ソートを用いて昇順に並び替える．基数ソートは，ソートされる要素の特定の桁にのみ注目し並び替えるという操作を下位の桁から順に行う．このプログラムでは各要素の定義域は 0〜65535（16 ビット）とし，下位から 8 ビットごとにソートを行う．状態遷移図で表すと状態数 80，状態遷移に分岐のある状態数は 8 で実現される．またデータレジスタ群で 216 個，各状態で用いる演算回路は 784 個のロジックブロックを使用する．

この例題に対しコンテキスト分割アルゴリズムを適用し，実装対象とする FPGA のサイズを変化させ試行した際の結果を表 1 に示す．アルゴリズムの改善段階での (I)〜(IV) の処理の繰返しは 100 回として実行した．項目 “FPGA サイズ”は 1 コンテキストで実現可能な回路量をロジックブロックの数で表したものである．EFSM のデータレジスタは各コンテキストにそのための領域が確保されていると仮定している（前提 (v)）ため，他の組み合わせ回路のために使用できるのは FPGA サイズからデータレジスタの占める領域を引いた分となり，その領域のサイズは表の第 1 列の () に示された値となる．“Greedy”は 4 章で説明したコンテキスト分割アルゴリズムの第一段階を適用した際に得られる初期解の目的関数値を示している．“Greedy+改善”は 4 章のアルゴリズムを改善段階も含めて適用した際に得られた解の目的関数値を示す．“コンテキスト数”は入力された EFSM がコンテキスト分割アルゴリズムによっていくつのコンテキストに分割されたかを示す．各試行において “Greedy”，“Greedy+改善”ともに得られた分割結果のコンテキスト数は同じであった．FPGA サイズが 350，400，450 として試行を行っ

た時に改善段階で解の改善が見られた．

表 1: ソート例題に適用した際の目的関数値

FPGA サイズ	Greedy	Greedy +改善	コンテキスト数
350(134)	17.33	20.19	7
400(184)	13.85	29.81	5
450(234)	12.31	13.06	4
500(284)	20.79	20.79	3
550(334)	11.15	11.15	3

Pentium III 600MHz, 256MB RAM

5.2 素数列挙

素数列挙プログラムはエラトステネスのアルゴリズムを用い 0〜1024 の範囲内の全素数を列挙する．状態遷移図で表すと状態数 37，状態遷移に分岐のある状態数は 4 で実現される．またデータレジスタ群で 118 個，各状態で用いる演算回路は 360 個のロジックブロックを使用する．

この例題に対しコンテキスト分割アルゴリズムを適用し，実装対象とする FPGA のサイズを変化させ試行した際の実行結果について表 2 に示す．アルゴリズムの改善段階での (I)〜(IV) の処理の繰返しは 100 回として実行した．FPGA サイズを 250，300 として試行を行った時に改善段階で解の改善が見られた．

表 2: 素数列挙の例題に適用した際の目的関数値

FPGA サイズ	Greedy	Greedy +改善	コンテキスト数
200 (82)	18.12	18.12	5
250(132)	15.45	27.53	4
300(182)	3.52	16.55	3
350(232)	9.47	9.47	2
400(282)	6.75	6.75	2

Pentium III 600MHz, 256MB RAM

5.3 アルゴリズム適用についての考察

実際の例題においては，一定の繰り返し処理を行うような状態遷移のループが状態遷移図内に多く見られる．このようなループとなる状態遷移は繰返し実行されるため，そのループ経路上を状態遷移している間はできるだけコンテキストの入れ替えが少ない方が望ましい．実際に使用されるコンテキストに格納可能なようにできるだけ少ない状態集合で分割すべきである．またループ経路上のすべての状態を 1 つの状態集合でグループ化することが可能であれば，FPGA に実装し動作させた際，その状態集合で構成されるコンテキストが実行された後，長期間コンテキストの変更を行わずにアプリケーションの実行を行うことができる．よってコンテキスト分割を行う際に，ループ経路上にある全状態が 1 状態集合でグループ化された場合はその状態集合の構成は保存しておき，他の状態集合の再グループ化により解の改善を行うべきであると考えられる．例えば図 6 の分割においてアルゴリズムの改善段階の (III) の処理が行われ，コンテキスト 2 の状態 s5 を移動させるとする．状態 s5 を移動させる候補としてはコンテキスト 1 とコンテキスト 3 があり，遷移確率によっていずれに移動させるかを決定する．遷移確率によりコンテキスト 1 に移されると決定し，さらに状態 s5 を移動させたことによってコンテキスト 1 から状態 s0 がコンテキスト 0 に移されたとする．このときループ経路上にある状態を 1 状態集合でグループ化した構成が変更されてしまう．5.1，5.2 の実行結果で初期解から改善されなかったものは，図 6 のコンテキスト 1 のような状態集合の構成が改善段階で変更されてしまい，全体の解を改善するように状態の移動が行われない．

この対策の 1 つとして目的関数の精度を向上させるために目的関数を変更することを考える．図 6 においてコンテキスト 1 について注目すると，状態 s0 から s4 の処理を繰り返した後，s4 の分岐からコンテキスト 2 へと状態遷移が行われる．次に実行するコンテキストが判明するのは，コンテキスト 0 から状態 s0 へ状態遷移が行われた時点である．今回適用した目的関数の算出方法ではこの場合のコンテキスト 1 における目的関数値は，s0 から s4 の遷移経路の長さから 5 と求められる．一方，FPGA 上での実際の動作では一定回繰返し処理が行われるため，次に実行されるコンテキストが判明してからコンテキストを変更するまでの時間は s0 から s4 の状態遷移の実行時間にループ回数を乗じた長さとなり，アプリケーション動作をストールさせずにより多くの設定情報を書き込んでおくことが可能である．たとえばループを 100 回繰り返した後，s4 からコンテキスト 2 へ状態遷移する場合，次に実行されるコンテキストが判明してからコンテキストを変更するまでの状態遷移経路の長さは 500 となる．ループの平均繰返し回数は，入力で与えられた分岐確率から推測できる．この算出方法を採用すると，ループ経路上の全状態を 1 状態集合にグループ化することに重みをつけ，またアプリケーション動作と並行して設定情報を書き換えることが可能な時間をより正確に近似できると思われる．

FPGA 上で動作させる際のコンテキスト変更を制御する回路の機能を考える．この制御回路は EFSM の状態遷移および FPGA 内に保持されているコンテキストを監視し，FPGA の設定記憶を選択する制御信号の切替えと外部からの設定情報の書き換えの指示を行う．現在実行中のコンテキストに割り付けられた状態から他のコンテキストに割り付けられた状態へ状態遷移が行われる時に，FPGA の設定記憶を選択する制御

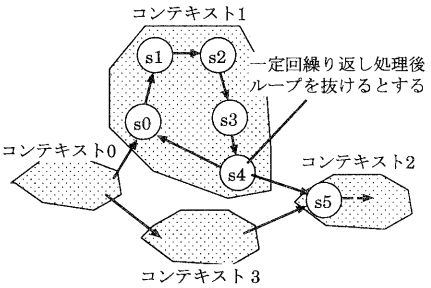

図 6: 状態遷移のループ

信号を切替える．また後に実行されるコンテキストは現在の状態遷移，状態遷移図，コンテキストに割り付けられた状態の集合で判断し必要に応じて書き換えの指示を行う．図6ではコンテキスト1の次には必ずコンテキスト2が実行される．そのためコンテキスト0を実行中，ある状態遷移において，次にコンテキスト1が実行されると判明した場合その時点で同時にコンテキスト1の次にコンテキスト2が実行されるとわかり，コンテキスト変更が複数回行われた後まで予測できる．一方，今回適用したアルゴリズムでは，実行中のコンテキストの次に実行されるコンテキストのみ早く判明できるように分割を行った．そのため実際の動作時には，目的関数値による予測よりも長期間アプリケーション動作と並行して設定情報の書き込みが行える．より正確な目的関数の設定とそれをアルゴリズムに反映させることが今後の課題である．

6 まとめ

マルチコンテキスト FPGA のためのコンテキスト分割アルゴリズムについて述べ，ハードウェア化されたアルゴリズム例題に対し実装対象の FPGA のサイズを変化させコンテキスト分割アルゴリズムを適用した．実際的な例題では，繰返し処理を行う状態遷移のループが状態遷移図に多く見られ，このようなループ経路上の状態はできるだけ少ない状態集合で分割されることが望ましい．今回適用したコンテキスト分割アルゴリズムではこの点を特に考慮していないため，改善段階で初期解より改善されないような試行が見られた．これに対してループの繰返し回数を考慮した目的関数値に変更することで改善が見込めると考えられる．

今後の課題としては，例題をコンテキスト分割した結果を利用して FPGA に実装した際の回路の実行速度の評価，より大規模な実用例題に対する適用，および現在の前提条件のいくつかを取り除き目的関数値をより正確なものとしアルゴリズムに反映させることを考えている．

参考文献

[1] 末吉敏則: Reconfigurable Computing System の現状と課題 -Computer Evolution へ向けて-, 信学技報 VLD, Vol. 96, No. 426, pp. 111–118 (1996).

[2] 末吉敏則: リコンフィギャラブルロジック, 電子情報通信学会誌, Vol. 81, No. 11, pp. 1100–1106 (1998).

[3] 凌暁萍, 天野英晴: データ駆動型制御機構付き MPLD を用いた並列処理マシン WASMII の仮想化, 情報処理学会論文誌, Vol. 35, No. 4, pp. 646–657 (1994).

[4] Chang, D. and Marek-Sadowska, M.: Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs, *IEEE Transactions on Computers*, Vol. 48, No. 6, pp. 565–578 (1999).

[5] 山口晃由, 橘山智訓, 大熊繁: Reconfigurable Computing System の暗号処理への適用, 情処研報 SLDM, Vol. 99, No. 101, pp. 25–30 (1999).

[6] 宇野正樹, 柴田裕一郎, 天野英晴, 古田浩一朗, 藤井太郎, 本村真人: 動的な再構成可能デバイスを用いた仮想ハードウェアシステム, 信学技報 VLD, Vol. 99, No. 530, pp. 53–60 (2000).

[7] 羽切崇, 戸川望, 柳澤政生, 大附辰夫: FPGA を用いた動的再構成システムと暗号化アルゴリズムへの応用, 信学技報 VLD, Vol. 99, No. 658, pp. 7–14 (2000).

[8] 岡部淳, 坂井修一, 田中英彦: 細粒度再構成可能なリコンフィギュラブルシステムの提案, 情報処理学会第 61 回全国大会, Vol. 1 (2000).

[9] 糸将之, 北道淳司, 船曳信生: 動的再構成可能 FPGA の設計とそれへの並列アルゴリズムの実装, 情処研報 DA, Vol. 98, No. 43, pp. 1–8 (1998).

[10] 本多亮, 北道淳司, 船曳信生, 東野輝夫: マルチコンテキスト FPGA におけるコンテキスト分割問題とその一解法, 情報処理学会第 61 回全国大会, Vol. 1, pp. 125–126 (2000).

[11] 日本電信電話株式会社: PARTHENON HOME PAGE, http://www.kecl.ntt.co.jp/parthenon/index_j.htm.

[12] Altera Corporation: ALTERA Homepage, http://www.altera.com.