システムＬＳＩ設計技術　99－5
（2001．1．11）

FPGAを用いた動的再構成可能システムを対象とするスケジューリング手法

石飛 貴志† 戸川 望†† 柳澤 政生† 大附 辰夫†

† 早稲田大学理工学部電子・情報通信学科
〒169-8555 東京都新宿区大久保3-4-1
E-mail: ishitobi@yanagi.comm.waseda.ac.jp
yanagi@yanagi.comm.waseda.ac.jp
to@ohtsuki.comm.waseda.ac.jp

†† 早稲田大学理工学総合研究センター
〒169-8555 東京都新宿区大久保3-4-1
E-mail: togawa@ohtsuki.comm.waseda.ac.jp

あらまし

近年，システムの動作中にシステムの一部の論理を書き換える動的再構成可能システムが研究されている．FPGAを用いた動的再構成可能システムでは，複数のFPGAを動的に再構成してアプリケーションを実行する場合，FPGAの再構成時間がシステム高速化のボトルネックとなるため，再構成時間を考慮したタスク割り当てが必要となる．本稿では，FPGAを用いた動的再構成可能システムのためのスケジューリング手法を提案する．提案手法は，タスクの実行順序とFPGAの資源を制約としシステムの処理時間の最小化を目的として，タスクの実行開始時刻と割り当てFPGAを決定する．本手法では，タスクの実行開始時刻と再構成時間がシステムの処理時間の遅れにどの程度影響するかを評価関数とし，さらに，評価関数に基づきタスクの割り当て可能候補を徐々に削除することで，各タスクの処理時間の遅れを均一化したスケジューリングを実行する．その結果，再構成の回数を考慮した上でシステムの処理時間の遅れを小さく抑えたスケジューリングが可能となる．計算機実験により手法の有効性を評価した結果を報告する．

キーワード　FPGA，スケジューリング，再構成可能システム，動的再構成，ディジタル信号処理

A Scheduling Algorithm for a Dynamic Reconfigurable System Based on Multiple FPGAs

Takashi ISHITOBI † Nozomu TOGAWA †† Masao YANAGISAWA † Tatsuo OHTSUKI†

† Dept. of Electronics, Information and Communication Engineering, Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan
E-mail: ishitobi@yanagi.comm.waseda.ac.jp
yanagi@yanagi.comm.waseda.ac.jp
to@ohtsuki.comm.waseda.ac.jp

†† Advanced Reserch Center for Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan
E-mail: togawa@ohtsuki.comm.waseda.ac.jp

Abstract

Recently, there has been proposed a dynamically reconfigurable system where a part of the system can be reconfigured in-system. In an FPGA-based dynamically reconfigurable system, a task scheduling algorithm which takes into account a reconfiguration time is required to minimize the runtime of an application running on the system. In this paper, we propose a scheduling algorithm for the dynamic reconfigurable system based on multiple FPGAs. The objective of the algorithm is to minimize the system runtime of the application. A task execution time and a processing FPGA allocation are determined under given FPGA resources and execution order of tasks. In the algorithm, we define a criterion how much the task execution time and the reconfiguration time influence delays of the system runtime of the application and we balance task delays based on the criterion by gradually reducing scheduling candinates of each task. Therefore, we can keep the delays of the system runtime of the application down by taking account of the number of reconfigurations. Experimental results demonstrate the efficiency and effectiveness of the algorithm.

Key Words *FPGA, scheduling, reconfigurable system, dynamic reconfiguration, digital signal processing*

1 まえがき

FPGA（Field Programmable Gate Arrays）はゲートアレイの一種であり，ユーザの手元でディジタル回路をプログラミングできる可変構造デバイスである．FPGAを用いることにより，あたかもソフトウェアのように何度も回路を書き換えることが可能なハードウェアを設計することができ，ディジタル回路の実装・動作確認を柔軟かつ迅速に行うことが可能となる．

FPGAの特徴の一つとして再構成機能が挙げられる．これはFPGA上に実現した回路をシステムの動作中に別の回路に書き換える動的再構成 [2], [13] が可能であることを意味する．したがって，アプリケーションのある機能を実行し終えたFPGAを別の論理に書き換えることにより仮想的に回路規模を増大させ，システムで使用するFPGAのチップ数を減らすことが可能である．処理の流れにおいて必要な機能のみをシステム上に実装することにより，それぞれの処理段階において資源を有効に活用することができる．結局，FPGAの動的再構成機能を取り入れたシステムを構築することで，小規模なシステムで大規模なアプリケーションの処理が可能となる．動的再構成可能システムは，これまで，文献 [3], [5], [6], [12], [15], [16] 等の研究例が報告されている．動的再構成可能システムは，既存のFPGAを用いたシステム [5], [6]，および動的再構成に特化した独自のデバイスを用いたシステム [3], [12], [15], [16] に分類できる．

現在，我々は大規模なディジタル信号処理アプリケーションの高速実行を目的とした動的再構成可能システムを構築している [10]．我々の目指すシステムは，既存のFPGA，CADを用いた実現性の高いシステムである．システムは1つの制御ユニット，複数の演算ユニットから構成される．制御ユニットは，システムとホスト計算機とのデータ転送，および各演算ユニットを制御する．各演算ユニットには，SRAMタイプのFPGAが1つ搭載され，FPGA全体の回路構成を一度に書き換える全体再構成が可能である．各演算ユニットでは，機能単位に分割されたアプリケーションを処理する．アプリケーションは，既存のCADを用いてFPGAで実行可能なデータに変換するため，1つの処理の粒度はある程度大きいものとなっている．アプリケーションの処理の流れは，1つの処理を1つのタスクとしたタスクフローグラフで表すことができ，システムの高速実行を目的として各タスクをFPGAに割り当てる場合，動的再構成を考慮したスケジューリングが必要となる．これまでの動的再構成可能システムのためのスケジューリング手法として，文献 [4], [7], [8], [14] が発表されている．しかし，手法 [7], [8] は，FPGAを部分的に書き換える部分再構成可能なFPGAを前提とし，しかも手法 [7] は，粒度の小さいタスク（加算器，乗算器等）を前提としているため，我々の目指す既存のFPGA，CADツールを用いた実現性の高いシステムにはそれらの手法は適用できない．一方，[4] は，粒度が大きいタスクを用い全体再構成を前提とし，ASAPレベル [11] を優先度としたリストスケジューリングに基づく手法であり，資源制約を満足しながら逐次的にタスクをFPGAに割り当てる．資源制約を満足しなくなるとFPGAを再構成して割り当てるため，システムの高速実行にはつながらない．システムの高速実行のためには，FPGAの再構成の回数を考慮する必要がある．

以上の背景より，本稿ではFPGAを用いた動的再構成可能システムのためのスケジューリング手法を提案する．提案手法は，タスクの実行順序とFPGAの資源を制約としシステム全体の処理時間の最小化を目的として，タスクの実行開始時刻と割り当てFPGAを決定する．本スケジューリング問題を実行期限付き0-1判定問題とした問題は，マルチプロセッサ・スケジューリング問題 [9] を多項式時間変換したものであるためNP完全である．したがって，本スケジューリング問題は，NP困難であると予想される．そこで，本手法ではまず，マルチプロセッサ・スケジューリング問題に対する最も有力な対処策と考えられているCP/MISF法 [9] や，文献 [4] に基づき，各タスクのレベルを決定し，レベル毎に分類する．次に，レベル毎に各タスクの評価関数を算出し，各タスクの実行開始時刻と割り当てFPGAを決定する．評価関数は，タスクの実行開始時刻と再構成時間がシステムの処理時間の遅れにどの程度影響するかを表す．評価関数に基づきタスクの割り当て可能候補を徐々に削除することで，各タスクの処理時間の遅れを均一化したスケジューリングを実行する．その結果，再構成の回数を考慮した上でシステムの処理時間を小さく抑えたスケジューリングが可能となる．計算機実験により提案手法の有効性を評価した結果を報告する．

2 準備

2.1 システムモデル

図1に動的再構成可能システムのモデルを示す．システムは，制御ユニット，演算ユニット，メインメモリから構成される．

システムは，ホスト計算機のPCIバスに接続される．制御ユニット，演算ユニット，メインメモリは，バスに並列接続される．制御ユニットは，アプリケーションの実行前に，ホスト計算機からPCIバスを経由してアプリケーションデータを読み込み，メインメモリ上に保持する．制御ユニットは，バスに接続された各演算ユニット，メインメモリを集中制御する．制御ユニットは，各演算ユニットと再構成を制御する専用線で直接接続され，並列的に各FPGAの再構成を可能とする．演算ユニットは，1つのFPGA，演算FPGAでの演算結果を保持するローカルメモリ，演算FPGAのコンフィギュレーションデータ（再構成用の配置・配線データ）を格納するメモリで構成される．演算FPGAでは，アプリケーションの演算器資源が実装され，制御ユニットによって動的再構成される．演算FPGAでの1つの処理に対する演算結果は，メモリに格納される．その際演算結果を，引続き同じ演算ユニットで使用する場合はローカルメモリに格納し，異なる演算ユニットで使用する場合はメインメモリに格納する．演算FPGA–ローカルメモリ間，演算FPGA–メインメモリ間のデータ通信時間は同じとする．

2.2 FPGA

システムで使用する演算FPGAは，すべて同一のものとし，その集合を

$$F = \{f_i | 1 \leq i \leq \mathrm{N_f}\} \tag{1}$$

と書く．演算FPGAは，構成要素としてCLB（Configurable Logic Block），IOB（I/O Block）を持ち，それらを演算FPGAの資源と定義する．演算FPGAの持つCLB，IOBの数をそれぞれ $\mathrm{n_{clb}}$，$\mathrm{n_{iob}}$ と書く．

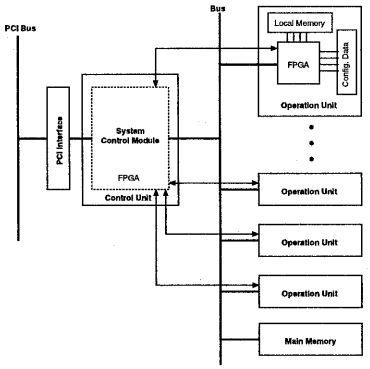

図 1. システムモデル.

演算 FPGA は，1 つの FPGA で同時に複数の異なる処理をすることが可能である．各処理ごとに CLB，IOB が割り当てられ，FPGA 上のすべての処理が終了したら動的に再構成される．演算 FPGA の再構成は，一度にすべてを書き換える全体再構成とする．再構成に必要な時間は，FPGA ごとに固有の値であり $\mathrm{T_{re}}$ と書く．

2.3 タスクフローグラフ

本システムの入力となるディジタル信号処理アプリケーションは，ある程度の粒度を持つ複数の処理で構成され，それらが逐次的に実行される．この一連の処理はタスクの流れを表現するタスクフローグラフ（TFG）によって表すことができる．TFG は無サイクル有向グラフとする．

TFG $G_{tf}(V,E)$ を考える（図 2）．$G_{tf}(V,E)$ は，1 つの入力節点と 1 つの出力節点を持つ[1]．V を節点集合，E を枝集合とする．節点 $v \in V$ はタスクを表す．各タスク $v \in V$ は演算 FPGA によって実行される．各タスクは，FPGA 上で実装する場合に必要な CLB 数，IOB 数とその処理時間（データ数 × クロックサイクル数）が与えられており，それぞれ $n_{clb}(v)$，$n_{iob}(v)$，$et(v)$ とする．ただし，各タスクの処理時間は，演算 FPGA–メモリ間のデータ通信時間を含むものとする．各タスクの粒度を

$$n_{clb}(v) \leq \mathrm{n_{clb}} \tag{2}$$

$$n_{iob}(v) \leq \mathrm{n_{iob}} \tag{3}$$

と定義する．枝 $e \in E$ は，タスク間の依存関係を表す．枝 $e = (u,v)$ が存在するときは，タスク v はタスク u の後に実行されることを表す．逆にタスク u はタスク v が開始される前に実行が終了されてなければならない．タスク間のデータ通信時間はタスクの処理時間に含める．各タスク $v \in V$ に関し，入枝（出枝）の集合を $E_{in}(v)$（$E_{out}(v)$），入枝（出枝）を介して v と接続している節点集合を $V_{in}(v)$（$V_{out}(v)$）とする．

2.4 スケジューリング

まず，時間を表す単位としてタイムステップという離散的な時間を定義する．**タイムステップ**とは，クロックに同期したステップであり，1 タイムステップは 1 クロック周期において実行される．タイムステップの集合を $TS = \{t | 1 \leq t \leq T_{all}\}$ とする．ここで，T_{all} とは，システムで処理が開始されてからすべてのタスクの処理が終了するまでの時間を表し，演算 FPGA $f_i \in F$ の処理が終了する時間を $t_{all\_end}(f_i)$ とすると，

$$T_{all} = \max_{1 \leq i \leq \mathrm{N_f}} [t_{all\_end}(f_i)] \tag{4}$$

と書ける．

TFG $G_{tf}(V,E)$ の**スケジューリング**とは，$v \in V$ の表すタスクに対し，そのタスクの処理が開始されるタイムステップ $ts(v)$ $(1 \leq ts(v) \leq T_{all})$ を決定することである．同時に，$v \in V$ を実行する FPGA $fe(v) \in F$ を決定することである．タスク v は，FPGA $fe(v)$，タイムステップ $ts(v)$ にそれぞれ割り当てられたと言う．

あるタイムステップ t において，FPGA f_i に割り当てられているタスクの集合を $V_{asd}(f_i,t) \subseteq V$ とするとき，FPGA f_i で使用されている CLB 数（IOB 数）$n_{clb}(f_i,t)$（$n_{iob}(f_i,t)$）を

$$n_{clb}(f_i,t) = \sum_{v \in V_{asd}(f_i,t)} n_{clb}(v) \tag{5}$$

$$n_{iob}(f_i,t) = \sum_{v \in V_{asd}(f_i,t)} n_{iob}(v) \tag{6}$$

と定義する．また，f_i に割り当てられているタスク集合 $V_{asd}(f_i,t) \subseteq V$ の処理が終了するタイムステップ $t_{asd\_end}(f_i,t)$ を

$$t_{asd\_end}(f_i,t) = \max_{v \in V_{asd}(f_i,t)} [ts(v) + et(v)] \tag{7}$$

と定義する．

演算 FPGA が，再構成する場合を考える．FPGA f_i について，再構成が開始されるタイムステップを $tr(f_i)$ とする．再構成している間はタスクの割り当てを行うことができないため，

$$V_{asd}(f_i,t) = \emptyset \ \ (tr(f_i) \leq t \leq tr(f_i) + \mathrm{T_{re}} - 1) \tag{8}$$

と書ける．

スケジューリングに際し，資源制約と実行順序制約を与える．**資源制約**とは，FPGA f_i で使用可能な資源の上限値であり，

$$n_{clb}(f_i,t) \leq \mathrm{n_{clb}} \tag{9}$$

$$n_{iob}(f_i,t) \leq \mathrm{n_{iob}} \tag{10}$$

と書ける．タスクの割り当ては資源制約を満たす場合に行う．

実行順序制約とは，タスクの実行順序を表す制約であり，タスク $u \in V$ の各出枝 $e = (u,v) \in E_o(u)$ に対し，

$$ts(u) + tm(u) - 1 < ts(v) \tag{11}$$

と書ける．

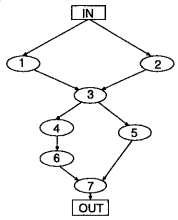

図 2. タスクフローグラフ.

[1] 与えられた TFG に複数の入出力節点がある場合には，ダミー節点を入出力節点に付加することによって，1 入力 1 出力の TFG に変換する．

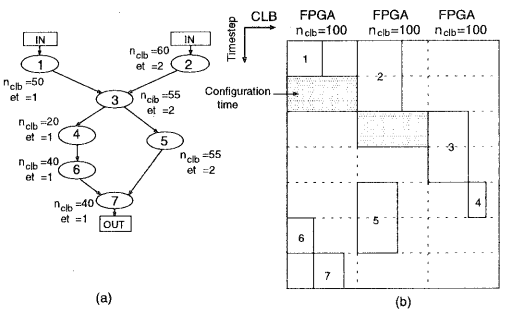

図 3. スケジューリング結果（$N_f = 3$ とする）. (a) 入力 TFG. (b) スケジューリング結果.

2.5 問題の定式化

以上の準備のもと，本稿で扱うスケジューリング問題を定義する．

定義 1 FPGAを用いた動的再構成可能システムを対象とするスケジューリング問題とは

(1) タスクフローグラフ $G_{tf}(V, E)$

(2) FPGA F

(3) FPGA 1 つ当たりの使用可能な CLB 数 n_{clb}，IOB 数 n_{iob}

(4) 再構成時間 T_{re}

が与えられたとき，資源制約・実行順序制約のもとに，

(a) 各タスク v の実行開始タイムステップ $ts(v)$

(b) 各タスク v を実行する FPGA $fe(v)$

をシステム全体の処理時間 T_{all} を最小化することを目的として決定することである．

図 3にスケジューリング結果の例を示す．

3 スケジューリングアルゴリズム

アルゴリズムに先立ち，用語を定義する．

ASAP レベル: TFG $G_{tf}(V, E)$ の各タスク $v \in V$ は，ASAP レベル $ASAP_{lv}(v)$ を持つ．$ASAP_{lv}(v)$ を

$$ASAP_{lv}(v) = \max_{u \in V_{in}(v)} [ASAP_{lv}(u)] + 1 \qquad (12)$$

と定義する．ただし，$V_{in}(v) = \emptyset$ の時，$ASAP_{lv}(v) = 1$ とする．

割り当て可能なタイムステップ: タスク v，FPGA f_i を考えるとき，タスク v が実行順序制約を満足し，かつ，FPGA f_i が再構成中でないタイムステップを割り当て可能なタイムステップと呼ぶ．タスク v が実行順序制約を満足するタイムステップの最小値は，ASAP スケジューリング [11] を実行して算出でき，$t_{asap}(v)$ とする．タスク v を f_i に割り当て可能なタイムステップ全体の集合を $TS_a(v, f_i)$ と定義する．

割り当て期限 $t_{alap}(v)$: ALAP スケジューリング [11] を実行して算出される各タスク $v \in V$ のタイムステップ．

割り当て時間違反: タスク v を割り当てる時に生じる違反を数値（後述）で表したものである．割り当て期限より前に割り当てる場合負の値，後に割り当てる場合正の値をとる．

割り当て方 k: タスク v の割り当て方法は 2 種類あり，$k = 0$ の割り当てとは FPGA を再構成しないで割り当てることであり，$k = 1$ の割り当てとは FPGA を再構成してから割り当てることを言う．

次に，本スケジューリング問題を実行期限付き 0-1 判定問題とした問題（以下 A）が NP 完全であることを示す．$A \in NP$ は明らかである．NP 完全問題であるマルチプロセッサ・スケジューリング問題[2]（以下 B）が問題 A に多項式時間で変換可能であることを示すには，B の任意の個別問題 B_i に対し，A のある個別問題 $f(B_i)$ を構成できればよい．今，問題 B は，

- $n_{clb}(v) = n_{clb}$，for all $v \in V$
- $n_{iob}(v) = n_{iob}$，for all $v \in V$
- $T_{re} = 0$

なる問題 A（以下 $f(B)$）と等価である．つまり，B における B_i に対する答えが yes のとき，かつそのときに限り，A における $f(B_i)$ に対する答えが yes になる．問題 B は問題 A に多項式時間で変換可能であることが言えるため，問題 A は NP 完全である．したがって，本スケジューリング問題は NP 困難であると考えられる．

本稿で提案するスケジューリングアルゴリズムは，ASAP レベルを用いた手法 [4] を基本とする．手法 [4] は，マルチプロセッサ・スケジューリング問題の最適に近い解を高速に得ることで知られる CP/MISF 法 [9] を，動的再構成を考慮したスケジューリングに適用した手法である．しかし，この手法は資源制約を満足しながら逐次的にタスクを割り当て，資源制約を満足しなくなると再構成して割り当てるため，再構成の回数を考慮しているとは言えない．提案手法は，レベル毎に各タスクの評価関数を算出し，再構成しないで割り当てるか再構成して割り当てるかを決定するため，再構成の回数を考慮したスケジューリングが可能となる．提案手法は，以下の処理に基づく．

(1) TFG $G_{tf}(V, E)$ 内の各タスクの ASAP レベルを算出する．

(2) ASAP レベルが同じタスクに対し，それぞれ評価関数を算出しスケジューリングする．

(3) すべての ASAP レベルに対し，(2) の処理を繰り返す．

以下，まず，処理 (2) における評価関数を定義し，続いて処理 (1)–(3) に基づいたアルゴリズムを提案する．

3.1 評価関数

ASAP レベル j に属する未割り当てタスク $v_j \in V_j$，FPGA f_i，割り当て方 k，ASAP レベル j 毎に与えられる定数 α_j が与えられるとき，評価関数 $c(v_j, f_i, k, \alpha_j)$ を

$$c(v_j, f_i, k, \alpha_j) = g(v_j, f_i, k) + \alpha_j \cdot h(v_j, f_i, k) \qquad (13)$$

と定義する．ここで，

- $g(v_j, f_i, k)$ ： 割り当て時間違反を表す関数．
- $h(v_j, f_i, k)$ ： その他の時間違反を表す関数．

とする．評価関数 $c(v_j, f_i, k, \alpha_j)$ は $2|V_j|N_f$ 個存在する．評価関数の詳細を以下に示す．ただし，評価関数を算出する際，資源制約を満たさない場合は，

[2] 実行時間が任意でありノンプリエンプティブなプロセスグラフを m 台のプロセッサ上で期限 T までにスケジューリング可能かどうかを判定する問題．

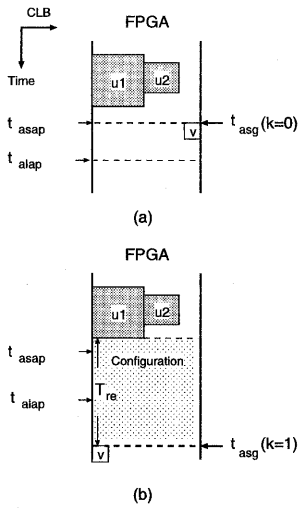

図 4. t_{asg} の決定. (a)$k=0$ の時. (b)$k=1$ の時.

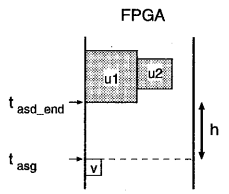

図 5. FPGA が動作していない時間.

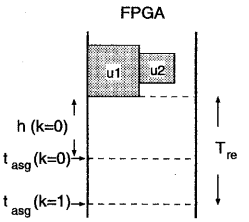

図 6. 定数 α_j の決定.

$$c(v_j, f_i, k, \alpha_j) = +\infty \tag{14}$$

とする.

3.1.1 割当て時間違反 $g(v_j, f_i, k)$

タスク v_j を, FPGA f_i に割り当て方 k で割り当てる時に生じる割り当て時間違反 $g(v_j, f_i, k)$ を

$$g(v_j, f_i, k) = t_{asg}(v_j, f_i, k) - t_{alap}(v_j) \tag{15}$$

と定義する. $t_{asg}(v_j, f_i, k)$ は, v_j を f_i に 2 つの割り当て種類 $(k = 0, 1)$ で割り当てる時のタイムステップとし,

・$k = 0$ の時:

$$t_{asg}(v_j, f_i, k) = \min_{t_a \in TS_a(v_j, f_i)} [t_a] \tag{16}$$

・$k = 1$ の時:

$$t_{asg}(v_j, f_i, k) = \max[t_{asap}(v_j), t_{asd\_end}(f_i, t_{asap}(v_j)) + \mathrm{T_{re}} + 1] \tag{17}$$

とする. 図 4に詳細を示す. 図中のタスク v, $FPGA$, t_{asap}, t_{alap} はそれぞれ v_j, f_i, $t_{asap}(v)$, $t_{alap}(v)$ を表す. $k = 0$ の時, $t_{asg}(v_j, f_i, k_{=0})$ は, v_j, f_i に対する割り当て可能なタイムステップの最小値となる. $k = 1$ の時, $t_{asg}(v_j, f_i, k_{=1})$ は, f_i を再構成すると仮定した時の, v_j, f_i に対する割り当て可能なタイムステップの最小値となる.

3.1.2 その他の時間違反 $h(v_j, f_i, k)$

その他の違反として, FPGA f_i が動作していない時間, つまりタスクが割り当てられていないタイムステップ数を割り当て時間違反と仮定する (図 5の h). 純粋な割り当て時間違反ではないので, 定数 α_j を掛けることで時間違反の尺度を変えることができる. 以下, 関数 $h(v_j, f_i, k)$ を,

・$k = 0$ の時:

$$h(v_j, f_i, k) = t_{asg}(v_j, f_i, k) - (t_{asd\_end}(f_i, t_{asap}(v_j)) + 1) \tag{18}$$

・$k = 1$ の時:

$$h(v_j, f_i, k) = t_{asg}(v_j, f_i, k) - \{t_{asd\_end}(f_i, t_{asap}(v_j) + \mathrm{T_{re}} + 1\} \tag{19}$$

と定義する. ただし,

$$h(v_j, f_i, k) = 0, \quad \text{if } V_{asd}(f_i, t_{asg}(v_j, f_i, k)) = \emptyset \tag{20}$$

とする.

3.1.3 定数 α_j の決定

α_j は各 ASAP レベル j 毎の定数でユーザが任意に決定する値である. 値の与え方により, $k = 0, 1$ どちらの割り当てを優先するかを決定できる.

$k = 0$ の時, 評価関数 c は最大で $\alpha_j \cdot h(v_j, f_i, k_{=0})$ をとり得る. $k = 1$ の時, 評価関数 c は最小で $\mathrm{T_{re}} - h(v_j, f_i, k_{=0})$ をとり得る. このとき, 最大値と最小値が一致する点を考える.

$$\alpha_j \cdot h(v_j, f_i, k_{=0}) = \mathrm{T_{re}} - h(v_j, f_i, k_{=0}) \tag{21}$$

よって,

$$\frac{h(v_j, f_i, k_{=0})}{\mathrm{T_{re}}} = \frac{1}{\alpha_j + 1} \tag{22}$$

$h(v_j, f_i, k_{=0})/\mathrm{T_{re}}$ は, 図 6に示すように, FPGA が動作していない時間の再構成時間に対する割合を表している. つまり, α_j により, FPGA が動作していない時間をどのくらいまで許容できるかをある程度決定できる. $k = 1$ の割り当てを優先させたい場合は, α_j を大きくすればよい. ASAP レベルが最大のレベルでは, 当然 α_j は小さい方がよい.

3.2 アルゴリズム

図 7にアルゴリズムを示す. まず, 与えられた TFG 内の各タスクの ASAP レベルを算出し, レベル毎にタスクを分類する (Step1). 次に, レベルの昇順に以下の操作を繰り返す (Step2–7). レベル j の未割り当ての各タスク $v_j \in V$ に対し, 評価関数を算出する. 評価関数の大きい割り当てを, 割り当て可能候補から徐々に削除していくことで最終的なスケジューリング結果を得

Step 1. 各タスクのASAPレベルを決定する．

Step 2. ASAPレベルを一つ増やす．ASAPレベルが存在しなければ終了．

Step 3. ASAPレベル j におけるタスク $v_j \in V_j$ の $c(v_j, f_i, k, \alpha_j)$ を算出する．

Step 4. $c(v_j, f_i, k, \alpha_j) = +\infty$ をすべて選択する．割り当て方を k とした場合の v_j の f_i への割り当てを不可能とする．

Step 5. $c(v_j, f_i, k, \alpha_j)$ の最大値を選択する．割り当て方を k とした場合の v_j の f_i への割り当てを不可能とする．

Step 6. Step5で選択した v_j の割り当て可能候補がただ1つになったならば，v_j を割り当てる．そうでなければ，Step5へ．

Step 7. 未割り当てタスクが存在するならば $c(v_j, f_i, k, \alpha_j)$ を更新し，Step4へ．そうでなければ，Step2へ．

図 7. スケジューリングアルゴリズム．

る．割り当て可能候補を徐々に削除することで，レベル内の各タスクの時間違反を均一化でき，各レベルの時間違反をできるだけ小さく抑えることができる．これにより，全体の処理時間の最小化につながると考えられる．以下，Step3–7における割り当て可能候補の削除方法を示す．

割り当て可能候補の削除

まず，ASAPレベル j の未割り当てタスク $v_j \in V_j$ に対し，評価関数 $c(v_j, f_i, k, \alpha_j)$ を算出する（Step3）．続いて $c(v_j, f_i, k, \alpha_j) = +\infty$ なる割り当てを選択し，割り当て方を k とした場合の v_j の f_i へ割り当て不可能とする（Step4）．評価関数 $c(v_j, f_i, k, \alpha_j)$ の値が大きい割り当てから順に，同様に割り当て不可能とする（Step5）．評価関数が等しい場合は，

$$\sum_{v \in V_{asd}(f_i, t_{asg}(v_j, f_i, k))} n_{clb}(v)$$

が小さい方を割り当て不可能とする[3]．あるタスク $u \in V_j$ の割り当て可能候補が1つになったら，$ts(u)$，$fe(u)$ が

$$ts(u) = t_{asg}(u, f_j, k) \tag{23}$$

$$fe(u) = f_j \tag{24}$$

と決定される（Step6）．未割り当てタスク $v_j \in V_j$ に対し，評価関数 $c(v_j, fe(u), k, \alpha_j)$ を更新する（Step7）．$V_j = \emptyset$ となるまで，以上の操作を繰り返す．図8に割り当て可能候補の削除例を示す．図8では，3つのタスク，2つのFPGAに対し評価関数が8つ求められている．まず，評価関数が $+\infty$ なる割り当てを削除する．続いて評価関数が大きいものから順に30，25を削除する．評価関数が20なるタスク v_2 の割り当て可能候補がただ1つになったため，v_2 を f_1 に割り当て方 $k = 1$ で割り当てる．評価関数を更新し，同様の操作を実行する．

3.3 計算複雑度

入力として与えられるTFG $G_{tf}(V, E)$，FPGA F に対し，$n = |V|$，$m = |F|$ として，計算複雑度を見積もる．図7のアルゴリズムで，最も時間を要するのは，step2–7の反復である．時間複雑度が最悪となるASAPレベル数が1のときを考える．評価関数 $c(v_j, f_i, k, \alpha_j)$ を算出する手間は $O(mn)$ である．

[3] 大きい方に割り当てることで資源使用率がより高くなると考えられる．

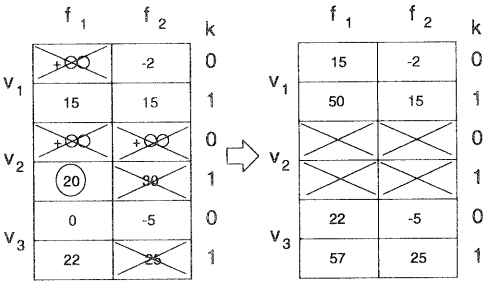

図 8. 割り当て可能候補の削除．

$c(v_j, f_i, k, \alpha_j)$ の最大値を探索・削除し，割り当て可能候補がただ1つとなるタスクを決定する手間は最悪で $O(m^2n^2)$ である．評価関数の更新にかかる手間は $O(n)$ である．以上より，タスク1つの割り当てを決定する手間は $O(m^2n^2)$ である．未割り当てタスクについて同様の操作をするので，時間複雑度は $O(m^2n^3)$ となる．一方，提案手法の空間複雑度は，$O(mn)$ である．

4 計算機実験結果

提案手法を，SUN Ultra Sparc 166Mhz 上に C 言語で実装し，計算機実験によって評価した．評価関数内の α_j は，レベル j が最大の場合のみ $\alpha_j = 0$ とし，それ以外は $\alpha_j = 0.5$ とした．タスクフローグラフとして，階層符号化（階層数3．図9），サブバンド符号化（図10）と，タスクグラフジェネレータ TGFF[1] で生成したタスクフローグラフ I–VI を用いた．タスク1つ当たりの平均処理時間は 10–20 タイムステップ，平均 CLB 数は 185–235，平均 IOB 数は 20–30 とした．

表 1に，階層符号化，サブバンド符号化を入力とし，ASAP レベル法 [4] と提案手法を比較した実験結果を示す．演算 FPGA は，Xilinx XC4020E（784CLBs，224IOBs，再構成時間 $30ms$）を想定した．表 2–4に，TGFF で生成したタスクフローグラフを入力とした実験結果を示す．使用する演算 FPGA の CLB 数，IOB 数をそれぞれ 500，100 とし，再構成時間は 15，30 タイムステップの 2 種類とした．表 2–3では，小規模な TFG に対し，分枝限定法で 1 日以内に最適解を求めることができた入力パラメータについてその結果を示し，比較する．表中，♯tasks はタスク数，♯FPGAs は演算 FPGA 数，♯res は再構成回数，Diff は最適解との差を百分率で示した値を表す．

実験結果から，提案手法は，処理時間について従来の手法と比較して同等あるいはそれ以上の良い結果が得られることがわかる．従来手法と処理時間が同じ結果に対し，再構成回数が本手法の方が多くなっている場合があるが，これは FPGA の動作していない時間を割り当て時間違反としたことで，動作していない時間が後に再構成時間に置き換わったためと考えられる．表 2では，小規模な入力に対し，従来手法では最適解との差が最大 30% とばらつきがあったが，提案手法は 5% 程度と安定した解が得られることがわかった．本手法は，レベル毎にタスクを分類し，各レベルの時間違反をできるだけ小さくすることで全体の処理時間の最小化を目指しているため，大規模な入力に対しても，最適解に近い解を安定して得られる可能性が高いと考えられる．しかも，

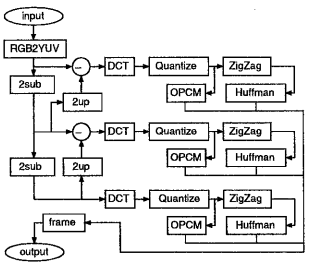

図 9. 階層符号化（階層数 3）.

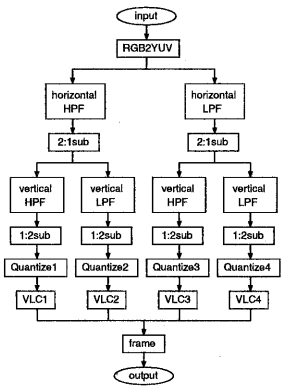

図 10. サブバンド符号化.

実行時間は 1 秒以下であり，高速にスケジューリング結果を求めることが可能である.

5 むすび

本稿では，FPGA を用いた動的再構成可能システムのためのスケジューリング手法を提案した．いくつかの入力に対して本手法を適用した結果，従来のスケジューリング手法と比較して同等あるいはそれ以上の良い結果を高速に得ることができた．しかも，実行時間は 1 秒以下である．また，提案手法は，アルゴリズムが簡潔であるため，拡張性が高いという特徴を持つと言える.

今後の課題として，本手法を現在我々が提案している動的再構成可能システム [10] のスケジューラとして組み込むことが挙げられる．本システムでは，最適なシステム構成を得るために複数のシステム候補を評価する高速なスケジューラが必要であり，本手法は十分な速度を持っていると考えられる．その際，分岐処理や反復処理を含むアプリケーションのタスクフローグラフをスケジューリングするために，条件分岐，反復処理にも対応する必要がある.

謝辞

本研究に関し，有用な議論，討論をいただいた本学香西伸治氏，陳暁梅氏に感謝いたします．本研究の一部は，文部省科学研究費補助金（基盤研究 C(2)，課題番号 10650345 および奨励研究 (A)，課題番号 12750369）の援助を受けた.

参考文献

[1] R. P. Dick, D. L. Rhodes and W. Wolf, "TGFF: Task graphs for free," in *Proc. of IEEE Int. Workshop on Hardware-Software Codesign*, pp. 97–101, 1998.

[2] J. G. Eldredge and B. L. Hutchings, "Density enhancement of a neural network using FPGAs and run-time reconfiguration," in *Proc. of IEEE Workshop on FPGAs for Costum Computing Machines*, pp. 180–188, 1994.

[3] T. Fujii, K. Furuta, M. Motomura, M. Nomura, M. Mizuno, K. Anjo, K. Wakabayashi, Y. Hirota, Y. Nakazawa, H. Ito and M. Yamashita, "A dynamically reconfigurable logic engine with a multi-context/multi-mode unified-cell architecture," in *ISSCC Digest of Technical Papers*, 21. 3, 1999.

[4] K. M. GajjalaPurna and D. Bhatia, "Temporal partitioning and scheduling for reconfigurable computing," in *Proc. of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 329–330, 1998.

[5] 羽切崇, 戸川望, 柳澤政生, 大附辰夫, "FPGA を用いた動的再構成可能システムと暗号化アルゴリズムへの応用," 信学技報, VLD99-109, pp. 7–14, 2000.

[6] 長谷川洋平, 戸川望, 柳澤政生, 大附辰夫, "FPGA を用いた動的再構成可能システムとその応用," 信学技報, VLD98-143, pp. 17–24, 1999.

[7] K. Ito, "A scheduling and allocation method to reduce data transfer time by dynamic reconfiguration," in *Proc. of ASP-DAC 2000*, pp. 323–328, 2000.

[8] B. Jeong, S. Yoo, S. Lee and K. Choi, "Hardwara-software cosynthesis for run-time incrementally reconfigurable FPGAs," in *Proc. of ASP-DAC 2000*, pp. 169–174, 2000.

[9] H. Kasahara and S. Narita, "Practical multiprocessor scheduling algorithms for efficient parallel processing," *IEEE Trans. Computers*, vol. c-33, no. 11, pp. 1023–1029, 1984.

[10] 香西伸治, 戸川望, 柳澤政生, 大附辰夫, "パラメータ付けされた動的再構成可能システムとその応用," 信学技報, VLD2000, Jan. 2001.

[11] M. T. -C. Lee, *High-Level Test Synthesis of Digital VLSI Circuits*, Artech House Publishers, 1997.

[12] X. P. Ling and H. Amano, "WASMII:A data driven computer on a virtual hardware," in *Proc. of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 33–42, 1993.

[13] W. Luk, N. Shirazi and P. Y. K. Cheung, "Modeling and opimizing run-time reconfiguration systems," in *Proc. of IEEE Workshop on FPGAs for Costum Computing Machines*, pp. 167–176, 1996.

[14] P. Merino, M. Jacome and J. C. López, "A methodology for task based partitioning and scheduling of dynamically reconfigurable systems," in *Proc. of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 324–325, 1998.

[15] K. Nagami, K. Oguri, T. Shiozawa, H. Ito and R. Konishi, "Plastic cell architecture: Towards reconfigurable computing for general-purpose," in *Proc. of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 68–77, 1998.

[16] Xilinx, *XC6200 Field Programmable Gate Arrays*, 1997.

表 1. 実験結果 1（再構成時間 = 30）.

			提案手法			[4]		
TFG	♯tasks	♯fpgas	処理時間	♯res	CPU time[s]	処理時間	♯res	CPU time[s]
階層		2	450	10	0.01	480	11	0.01
符号化	22	3	350	9	0.01	365	9	0.01
		4	285	7	0.02	290	9	0.01
サブ		2	350	10	0.01	350	10	0.01
バンド	21	3	260	11	0.01	260	9	0.01
符号化		4	190	8	0.02	200	8	0.01

♯tasks: タスク数, ♯fpgas: 演算 FPGA 数, ♯res: 再構成回数

表 2. 実験結果 2（再構成時間 = 15）.

			提案手法			[4]			最適解
TFG	♯tasks	♯fpgas	処理時間	♯res	Diff[%]	処理時間	♯res	Diff[%]	処理時間
I	10	2	66	3	1.5	78	3	20.0	65
		3	53	2	3.9	53	2	3.9	51
		4	51	1	0	51	1	0	51
II	15	2	91	4	7.1	92	4	8.2	85
		3	67	5	6.3	67	3	6.3	63
III	15	2	168	7	0	168	7	0	168
		3	143	7	0	143	6	0	143
IV	20	2	122	7	0.83	128	6	5.8	121
		3	92	6	8.2	92	5	8.2	85

♯tasks: タスク数, ♯fpgas: 演算 FPGA 数, ♯res: 再構成回数, Diff: 最適解との差を百分率で表した値

表 3. 実験結果 3（再構成時間 = 30）.

			提案手法			[4]			最適解
TFG	♯tasks	♯fpgas	処理時間	♯rec	Diff[%]	処理時間	♯rec	Diff[%]	処理時間
I	10	2	81	2	0	108	3	33.3	81
		3	68	2	4.6	68	2	4.6	65
		4	51	1	0	65	1	27.5	51
II	15	2	121	4	5.2	122	4	6.1	115
		3	82	3	5.1	82	3	5.1	78
		4	76	4	5.3	76	2	5.3	72
III	15	2	223	7	7.7	223	7	7.7	207
		3	157	6	3.3	157	6	3.3	152
IV	20	2	167	6	0.60	173	6	4.2	166
		3	124	6	7.8	124	5	7.8	115

♯tasks: タスク数, ♯fpgas: 演算 FPGA 数, ♯res: 再構成回数, Diff: 最適解との差を百分率で表した値

表 4. 実験結果 4（再構成時間 = 30）.

			提案手法			[4]		
TFG	♯tasks	♯fpgas	処理時間	♯res	CPU time[s]	処理時間	♯res	CPU time[s]
V	30	2	287	11	0.02	296	11	0.01
		4	157	10	0.04	168	9	0.01
		6	124	8	0.05	129	7	0.02
VI	50	4	635	47	0.09	650	47	0.02
		8	323	43	0.32	338	43	0.02
		12	211	38	0.78	238	39	0.03

♯tasks: タスク数, ♯fpgas: 演算 FPGA 数, ♯res: 再構成回数