修 士 論 文

３次元パッキングに基づく動的再構成スケジューリング

指導教官　金子峰雄助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

横山 順一

平成 13 年 3 月 31 日

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 本研究の目的

システム LSI は 1 つの LSI で複雑な処理を行えるが，ゲート数が数百万を超え，増加の傾向はさらに強まる一方である．システム規模の増大にともない，大規模アプリケーションの実装が要求されてきている．ソフトウェアによる実現は，多種多様なアプリケーションを柔軟に実装できる点で優れているが，ハードウェア実装に比べて演算速度面で劣る．一方，ハードウェアによる実現では，高速ではあるが，多種多様なアプリケーションに対応するためには膨大なハードウェア資源が必要となる．こうした中で，ハードウェア実現の高速性と，ソフトウェア実現の柔軟性を兼ね備えた動的再構成が注目されつつある．

本研究では，動的再構成可能システムの性能を十分に引き出し，大規模計算処理を限られたハードウェア資源上で高速かつ効率的に実行するための設計論の構築を目的とする．

1.2 本研究の背景

近年，LSI システムの論理を動作中に書き換える動的再構成可能システムの研究が，おもに FPGA(Field Programmable Gate Arrays) を対象として行われている．FPGA の論理規模をはるかに上回る大規模なアプリケーションを，FPGA を用いたシステム上で動的再構成を行うことにより実装し，高速に計算処理を行うことが可能となってきている．データ暗号化標準アルゴリズムの実装等の具体的提案もなされている．

本研究では，与えられたハードウェア資源に対してそれを上回る大規模な計算処理を，動的再構成にて実装するための演算処理データの時間スケジューリング及び，物理空間

への配置の問題について検討を行う．当該問題は，基本的に直方体（計算ブロックの時間的，空間的広がりに対応）の３次元パッキング問題に帰着され，計算ブロックでの実際の計算において先行する計算ブロックの再構成時間を含めた定式化と最適化手法の確立を目指す．

第 2 章

動的再構成について

2.1 FPGA(Field Programmable Gate Array)

FPGA は AND-OR アレイを使わず，GAL(Generic Array Logic) のマクロセルをさらに強化したロジックセルを組み合わせて，ランダム・ロジックを実現する．広義には，FPGA も PLD(Programmable Logic Device) の一種と言えるが，AND-OR アレイを基本とする従来の PLD とは構造が大きく異なる．従来の PLD より設計の自由度が高く，ゲートアレイに近い特長をもつことから，FPGA と呼ばれる．

発表当時は，PLD と FPGA のどちらも集積度が 1000 ゲート程度と低く，スピードも遅かったため，あまり普及しなかった．FPGA は 90 年代に入って次第に普及をはじめ，FPGA 製品を発売するメーカも増えてきた．90 年代には数千ゲートから数万ゲートへ，90 年代後半には数万ゲートから数十万ゲートへと発展してきた．

初期の FPGA のプログラム素子は，アンチフューズと SRAM に二分されていた．最近では，一括消去型の EEPROM であるフラッシュメモリも用いられている．アンチフューズは，消去と再書き込みができないという欠点をもつが，SRAM より配線抵抗や占有面積を小さくできる．そのため，FPGA の中で最も容易に高集積度，高速が得られる．ゲートアレイに近い特長をもつことから，数社がアンチフューズを採用した．SRAM は電気的に消去と再書き込み可能だが，EEPROM のように不揮発性ではなく，電源をオフにすると配線情報が失われる．そのため外部の不揮発メモリに配線情報を記憶しておき，パワーオン時にロードしなければならない．これは欠点でもあるが，現在ではインシステムプログラミング (基板上に装着したデバイスをその場で再プログラミングできる機能) として，逆に SRAM デバイスの利点と考えられることが多くなった．

ゲートアレイはトランジスタ単位で自由にプログラミング可能であり，設計の自由度は

きわめて高い．だが，同じことを FPGA で実現しようとすると，プログラム素子の占有面積や配線の遅延時間が大きくなってしまう．FPGA では，ある大きさの基本ロジックセル (マクロセル) を単位として回路を構成する．基本ロジックセルは数ゲート〜十数ゲートのプログラマブルな機能ブロックで，セル内部では遅延時間の小さい回路を実現できる．それを組み合わせることによって，ランダム・ロジックを実現する．基本ロジックセルのサイズが小さいほど設計の自由度は高いが，セル間の配線が増えるために集積度やスピードが低下する．基本ロジックセルのサイズが大きいほど高速の回路を実現できるが，一部分しか使わないセルが多くなり，ゲートの利用効率が低下する．

2.2 動的再構成

FPGA 内部の計算ブロックの機能や，論理ブロック間の配線パタンを論理構成データとし，ある論理で構成されている FPGA に，別の論理で構成しなおすことが FPGA の再構成機能である．FPGA の再構成機能を利用して専用ハードウェアの高速性と，ソフトウェアの汎用性の両立を兼ね備えたシステムを再構成可能システムという．[1, 2]

動的再構成とはアプリケーションを実行中に，ある論理を実行し終えた FPGA 全体もしくは一部の機能ブロック領域に別の論理を割り当てることであり，SRAM タイプの FPGA は，インシステムプログラミング (基板上に装着したデバイスをその場で再プログラミングできる機能) を利用して，電源を投入したまま再構成できる．このシステムを動的再構成可能システムという．[10],[3],[4]

第 3 章

計算ブロック配置問題

動的再構成システムによる計算実行において，物理的な 2 次元空間への計算ブロックの配置問題と，再構成時刻の関係を考える．FPGA 内に構成される計算ブロックに演算が割り当てられ，どのような条件のもとで計算が実行可能であるかを示す.

問題の入力と出力

問題の入力を，dependence graph $G(V, A)$（図 3.1 参照）とし,

V : 演算の集合と,

A : 演算間のデータの依存関係,

とする. 加えて,

H_F :FPGA 物理サイズの高さ,

W_F :FPGA 物理サイズの幅,

$B = \{b_1, b_2, b_3, \ldots, b_{m-1}, b_m\}$，計算ブロックの種類,

$S = \{s_1, s_2, s_3, \ldots, s_{n-1}, s_n\}$，演算の種類,

$M : V \longrightarrow O$，演算の種類を求める関数,

$t_{rc} : B \longrightarrow Z_+$，再構成に要する時間,

$t_{ex} : O \times B \longrightarrow Z_+$，演算に要する時間,

$h : B \longrightarrow Z_+$，計算ブロック物理サイズの高さ,

$w : B \longrightarrow Z_+$，計算ブロック物理サイズの幅,

とする.

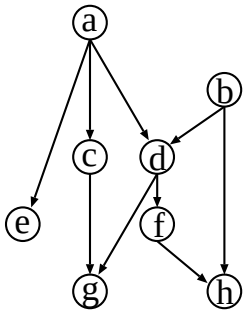

図 3.1: Dependence Graph

出力は，$v \in V$ に対する計算ブロックの，

$t_c : V \longrightarrow Z_+$，構成時刻，

$t_s : V \longrightarrow Z_+$，計算開始時刻，

$x : V \longrightarrow Z_+$，FPGA 上における構成場所の x 座標，

$y : V \longrightarrow Z_+$，FPGA 上における構成場所の y 座標，

とする．（図 3.2 参照）

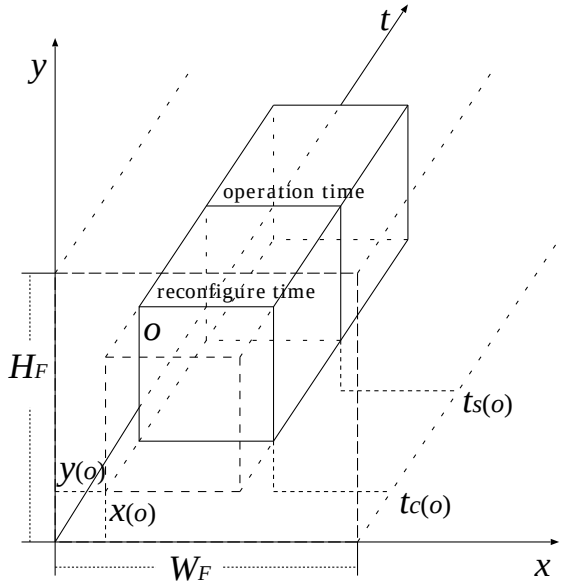

図 3.2: 問題の出力

ここでは，簡単化のため，演算 v から計算ブロックの種類 B を決定問題は取り扱わず，あらかじめ与えられているものとする．

3.1 実行可能条件

任意の 2 つの演算をそれぞれ o と o' とし，出力結果がどのような条件を満たすときに計算が実行可能となるかについて述べる．

計算ブロックの FPGA 上の xy 平面での重なりと種類について

計算ブロックの種類が同じでかつ，FPGA 上の xy 平面において計算ブロック領域の一致するとき，

$M(o) = M(o')$
$\wedge\ x(o) = x(o')$
$\wedge\ y(o) = y(o')$

と表現でき，この条件を O_{esp} とする．図 3.3 に示す．

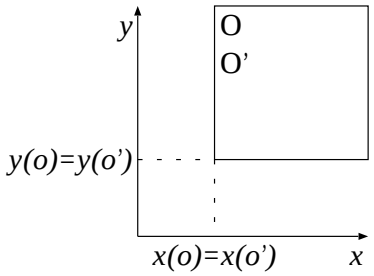

図 3.3: O_{esp}

FPGA 上の xy 平面において計算ブロック領域が重なる条件は，

$x(o) + w(o) \geq x(o') \wedge x(o') + w(o') \geq x(o)$
$\wedge\ y(o) + h(o) \geq y(o') \wedge y(o') + h(o') \geq y(o)$

と表現でき，この条件を O_r とする．図 3.6 に示す．

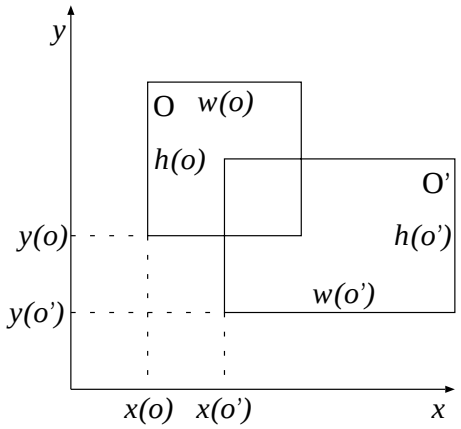

図 3.4: O_r

以上 2 つの条件 O_{esp} と O_r に当てはまらないときは，2 つの演算 o と o' について FPGA 上の xy 平面において重なりがない．

計算ブロックの時間の重なり

演算を演算時間だけとしたときの，計算ブロックの重なりの条件を O_{te} とし以下のように表現できる．

$t_s(o) + t_{ex}(o) > t_s(o')$
$\wedge\ t_s(o') + t_{ex}(o') > t_s(o)$

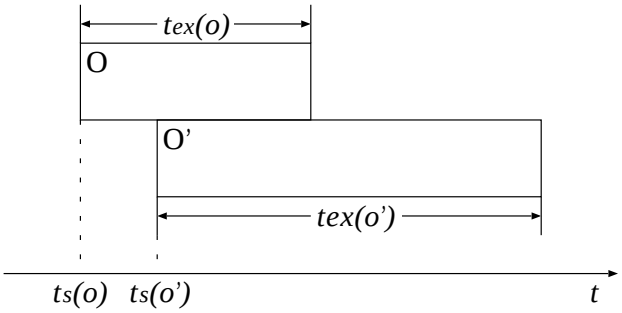

図 3.5: O_{te}

演算を再構成時間を含めた，計算ブロックが重なるときの条件を O_t とし以下のように表現できる．

$t_s(o) + t_{ex}(o) - 1 \geq t_s(o') - t_{rc}(o')$

$\wedge\ t_s(o') + t_{ex}(o') - 1 \geq t_s(o) - t_{rc}(o)$

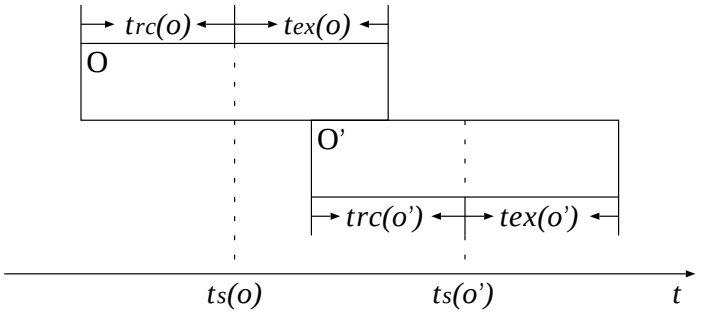

図 3.6: O_t

今まで述べた条件 O_r，O_{esp}，O_t と，O_{te} より，２つの演算 o と o' が実行できない条件を示す.（１）$O_{esp} \wedge O_{te}$，計算ブロックの構成場所が一致し，計算実行時刻に重なりがある,（２）$(O_r \wedge \bar{O}_{esp}) \wedge O_{te}$，計算ブロック構成領域に重なりがあり，計算実行時刻に重なりがある,（３）$(O_r \wedge \bar{O}_{esp}) \wedge (O_t \wedge \bar{O}_{te})$，計算ブロック構成領域に重なりがあり，計算実行時間以外（再構成時間）に重なりがある，の 3 つの場合演算間で衝突があるので演算実行不可能である．

表 4.2 に実行可能条件を示す．任意の演算に対して衝突なく演算の割り当てが可能で，実行可能であるものを ◯ で，実行不可能なものを × で表す．

表 3.1: 演算実行可能条件

	O_{te}	$O_t \wedge \bar{O}_{te}$	$\bar{O}_t$
O_{esp}	×	◯	◯
$O_r \wedge \bar{O}_{esp}$	×	×	◯
$\bar{O}_r$	◯	◯	◯

第 4 章

動的再構成スケジューリング問題

始めに，動的再構成スケジューリング問題の複雑さを明らかにする目的で，計算ブロックの平面上への配置を同時刻に構成可能な計算ブロックの個数に置き換えて理論的な考察を行う．

演算の種類の数１，構成できる計算ブロックの数 n のとき

演算の種類が１種類で，同時に構成できる計算ブロックの数が n 個のときは，スケジューリング問題において，再構成が始めの演算開始時に１回だけ必要である．このため，再構成を考慮しない通常の並列スケジューリング問題に帰着される．

演算の種類の数２，構成できる計算ブロックの数１のときの解

演算の種類が２種類で，同時に構成できる計算ブロックの数が１つのときの最適解を述べる．図 4.1 参照．はじめに，問題を定義する．つぎに，最適解を求めるアルゴリズムを述べ，定理の証明を行う

- FPGA 上には，同じ時刻に１つだけ演算ブロックを構成できる．
- 演算の種類の数 $|C| = 2$ とする．
  ここで計算ブロックは，タイプ１と，タイプ２の２種類である．
- S_p を p 回目までの再構成で演算できる，演算の集合とする．
- $p+1$ 回目までの再構成で実行できる演算の集合は，p 回目までの演算の集合を含む．$S_{p+1} \supset S_p$ である．

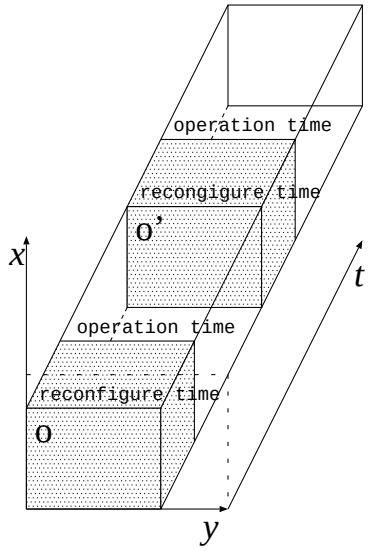

図 4.1: 演算の種類の数２，構成できる計算ブロックの数１

以下に示すアルゴリズムは，現在構成されている計算ブロックで実行可能な演算を全て割り当て，再構成を行い繰り返し演算の割り当て行き，割り当てる演算がなくなるまで行う方法である．

アルゴリズム

ステップ１

演算の種類 $i = 1$ とする．

ステップ２

タイプ i の計算ブロックを構成する．

ステップ３

タイプ i の計算ブロックで割り当て可能な演算を，全て割り当てる．

ステップ４

割り当てる演算が，まだ残っている場合には，計算ブロックを別の種類に再構成し直し，ステップ３へ．全ての演算が終了した場合は，解に S^i の名前を付け，次のステップへ．

ステップ５

$i = 1$ の場合，$i = 2$ としてステップ２へ戻り，スケジューリングを再度行う．$i = 2$ の

場合，計算ブロックのタイプ１から始めたスケジューリングと，計算ブロックのタイプ２から始めたスケジューリングの，割り当てコントロールステップ数を比較し短い方を出力して終了.

定理

このアルゴリズムで求められた解は最適解である.

証明

定理を背理法で証明する.

このアルゴリズムで求められた解 σ を，最適解でないものと仮定する.

最適解の集合の中から，１つ σ' を選び，始めに割り当てられる演算の種類が同じである σ_i と比較する．σ' は最適解の中で，σ と比較して再構成時間も含めもっとも大きいコントロールステップで違いがでるものとする.（ここでは，n 番目まで σ' と，σ の並びが等しく n+1 コントロールステップにて初めて異なる割り当てとなるものとする.）

表 4.1: 解の並び

Control Step			$n-1$	n	$n+1$	$n+2$	...
σ_i	a	b	c	d	e	f	...
σ'	a	b	c	d	f	h	...

σ_i の $n+1$ 番目の要素 e は，n 番目まで最適解 σ' にも含まれていない.（n 番目まで並びが同じだから.）必ず $n+2$ 番目から最後までの間に含まれる.

σ_i から，n 番目の d の次のコントロールステップで e の演算が実行できる.

σ' の $n+1$ 番目に演算 e を入れて，$n+1$ 番目から演算 e の入っていた前の演算までを右に1つシフトしても，σ_i から最後まで実行できる.

よって，最適解 σ' の n 番目の次に σ_i と同じ演算ができるので，σ_i と少なくとも $n+1$ 番目のコントロールステップまで同じ最適解がある.

以上より，矛盾が生じ，このアルゴリズムで求められた解は最適解である．□

4.1 演算の種類の数と計算ブロックの数が大きい問題

計算ブロックの種類の数が１である問題は，再構成を考慮しない通常の並列スケジューリング問題に帰着され，演算器の数が２つの問題が解かれている [9].

同時刻に構成可能な計算ブロック数を１，計算ブロックの種類の数を２とした問題に対しては，最短スケジューリングを求める多項式時間アルゴリズムを導いたが，同時刻に構成可能な計算ブロック数が２以上の問題，計算ブロックの種類の数が３以上の問題の複雑さは今後の課題となっている.

表 4.2: 解かれている問題

	演算の種類の数		
計算ブロックの数	1	2	3
1	○	○	
2	○		
3			

第 5 章

３次元パッキングに基づく解空間構成

5.1　問題の帰着

動的再構成システムでの計算実行は，各演算実行時に FPGA 上に論理回路が形成される．形成される論理回路は，矩形の面積（縦の長さ×横の長さ）を持つものとし，さらに再構成時間と演算時間からなる存在時間を考え３次元の計算ブロックとする.（図 5.1 参照）

動的再構成のスケジューリング問題は，演算実行時に構成される計算ブロックの時間的空間的に制約のあるパッキング問題とみなすことができる.（図 5.2 参照）

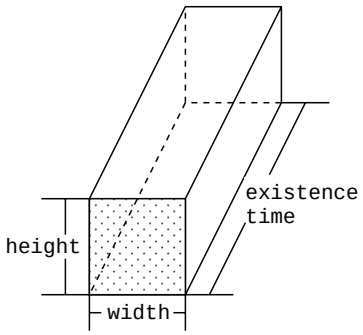

図 5.1: 計算ブロック

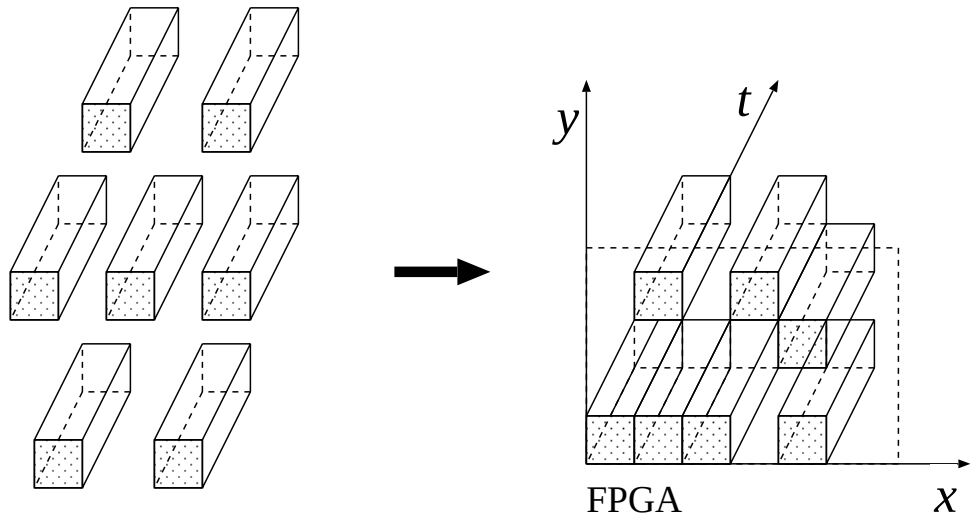

図 5.2: 計算ブロックのパッキング

5.2 ３次元パッキング

与えられたブロックをできるだけ小さい面積の矩形内に配置するという２次元パッキング問題に対し，ブロックが矩形と限定されているときの解の表現方法として，sequence-pair が提案された．ここでは３次元パッキングの表現方法として，sequence-pair を元に提案された sequence-quintuple を用いる．sequence-quintuple は，すべての３次元パッキングを表現することができる．

5.2.1 Sequence-Pair

矩形のボックス n 個が，高さと幅と共に与えられているとする．sequence-pair は，(Γ_+, Γ_-) のように，すべてのボックスラベルの順列の順序のあるペアである．Γ_* ($*$ は，$+$ もしくは $-$）は，ボックスラベルから１次元配置へのマッピングと見なす．もしボックス b が，Γ_* の k 番目のエレメントならば，$\Gamma_*(k) = b$ もしくは，$\Gamma_*^{-1}(b) = k$ と表す．sequence-pair は，以下に示す復号化ルールでのパッキングのトポロジーのコードとされる．

デコード規則： sequence-pair から２次元トポロジー

sequence-pair，(Γ_+,Γ_-) が与えられているとき，すべてのボックスのペア a と b は，以下のトポロジーによって割り当てられる．

RL-topology

$\Gamma_+^{-1}(a) < \Gamma_+^{-1}(b)$ かつ $\Gamma_-^{-1}(a) < \Gamma_-^{-1}(b) \longrightarrow a$ は，b の左にある．(b は，a の右にある．)

AB-topology

$\Gamma_+^{-1}(a) < \Gamma_+^{-1}(b)$ かつ $\Gamma_-^{-1}(a) > \Gamma_-^{-1}(b) \longrightarrow a$ は，b の上にある．(b は，a の下にある．)

以下の方法で，頂点が重み付けされた directed graph G_{RL} と G_{AB} のペアを作る．

- 頂点の集合は，s, t と n 個のボックスからなる．
- s と t は，それぞれソースとシンクと呼ぶ．
- それらは，混乱がおきないかぎりボックスラベルによって参照される．
- もし sequence-pair が，ボックス a は，ボックス b の左とデコードされるならば，G_{RL} に directed edge(a,b) を割り当てる．
- もし，sequence-pair が，a は，b の上であるならば，G_{AB} に directed edge(a,b) がたされる．
- 最後に，辺 (s, b) と (b, t) は，それぞれのグラフで共通にすべての頂点 b に足される．

頂点 b が，G_{RL} で幅の重みを，G_{AB} で高さの重みを持つが，辺は重みを持たない．これらのグラフが，ループがないことは明白である．s からすべての頂点へのもっとも長いパスは，それぞれのグラフで多項式時間に見つけることができる．ソースからボックス b までの長さを，G_{RL} で $l_H(b)$ と G_{AB} で $l_V(b)$ とする．パッキングは，(x, y) に b を配置することによって実現する．[5] ここで，$x = l_{RL}(b), y = l_{AB}(b)$ である．図 5.3，図 5.4 を参照．

結果として生ずるパッキングは，sequence-pair からデコードされるトポロジーを満たす，最小の幅と最小の高さのパッキングである．

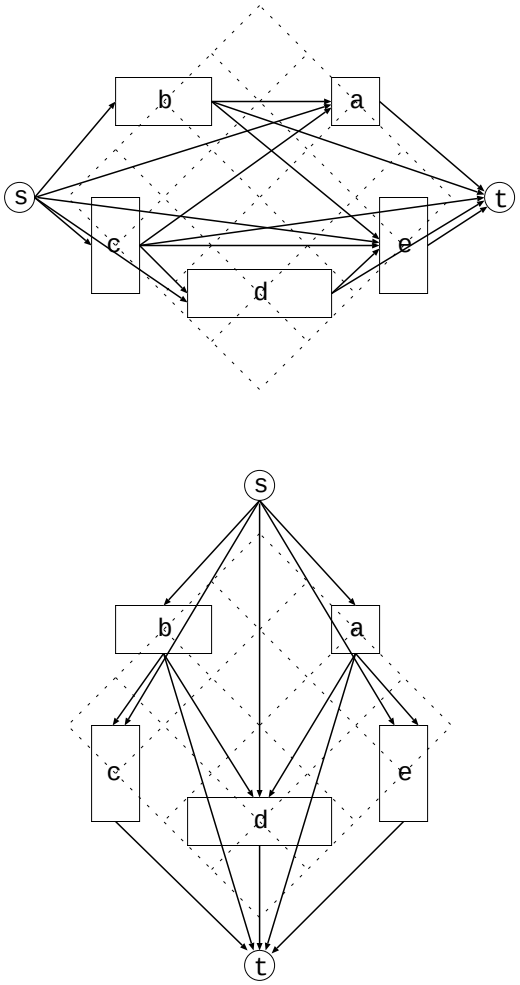

図 5.3: $SP = (\Gamma_+, \Gamma_-) = (bcade, cdbea)$ の G_{RL} と G_{AB}

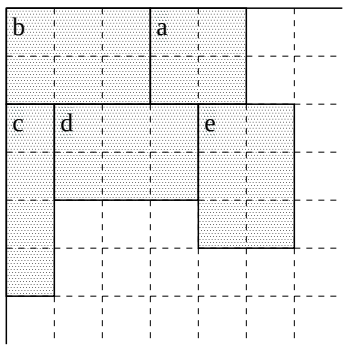

図 5.4: $SP = (\Gamma_+, \Gamma_-) = (bcade, cdbea)$ の 2 次元パッキング

5.2.2 Sequence-Quintuple

sequence-quintuple は，$(\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5)$ で表される．n 個のボックスが与えられているとき，以下のアルゴリズムで３次元パッキングを定める．[6]

アルゴリズム

ステップ 1

Sequence-Pair(Γ_1, Γ_2) により，$RL-topology$ を表すため，左右の制約グラフ G_{RL} を作る．sequence-pair と同じように決定するが，右左だけの関係 $\Gamma_1^{-1}(a) < \Gamma_1^{-1}(b)$ かつ $\Gamma_2^{-1}(a) < \Gamma_2^{-1}(b) \longrightarrow a$ は，b の左にある（b は，a の右にある），だけとする．頂点は，s，t と，対応するボックスの幅の重みでラベル付けされた n 個の頂点とする．辺は，すべてのボックス b に対して，辺 (s, b) と，辺 (b, t) に加え，ボックス a がボックス b の左にあるならば，辺 (a, b) である．

また同様に，前後の制約グラフ G_{FR} は，Sequence-Pair(Γ_3, Γ_4) から $\Gamma_3^{-1}(a) < \Gamma_3^{-1}(b)$ かつ $\Gamma_4^{-1}(a) < \Gamma_4^{-1}(b) \longrightarrow a$ は，b の前にある（b は，a の後ろにある），の規則によって作られる．

ステップ 2

グラフ G_{RL} で，ソースからそれぞれの頂点への最長パスの長さを求める．その長さを，対応するボックスの x 座標とする．

同様に，G_{FR} によって y 座標も決定する．

ここですべてのボックスは，ボックスの xy 座標が決定された．2 つのボックスが同じ xy 座標で重なりがあるならば，$xy-overlapping$ という．すなわち，2 つのボックスは，投影された xy 平面で重なり合う．

ステップ 3

以下の手順で上下の制約グラフ G_{AB} を作る．s と t に付随する頂点と辺は，ほかの制約グラフと同じ方法で決定される．ボックス a と b のすべてのペアに対して，必要十分条件として a と b は $xy-overlapping$ で，かつ，$\Gamma_5^{-1}(a) < \Gamma_5^{-1}(b)$ であるならば，a から b のエッジが足される．

ステップ 4

z 座標は，グラフ G_{AB} のもっとも長いパスによって決定する．

すべての sequence-quintuple に対して，このアルゴリズムは，唯一かつ最適化された３次元パッキングを導く．

5.3　パッキングのコード化

直方体の３次元パッキング解空間の表現として提案されている sequence-quintuple を基礎とし，先行制約グラフにて指定された演算間の先行関係と計算ブロックの再構成を反映した直方体の時間軸方向の伸び縮みを考慮した解表現（解のコード化）を提案する．sequence-quintuple は，５つの計算ブロック名順列を使うものであり，Γ_1 と Γ_2 の２つの順列にて計算ブロックの FPGA 上の X 座標を，Γ_3 と Γ_4 の２つにて Y 座標をそれぞれ算出し，先行制約グラフのトポロジカル順序に限定した Γ_5 は，$x-y$ 平面上での重なりのある計算ブロック a と b に対し時間方向での順番を付け，計算ブロックの再構成時刻，演算の実行時刻を算出する．

定理

Γ_5 が与えられたグラフ G のトポロジカル順序を満足するとき，sequence-quintuple に動的再構成スケジューリング問題の最適解が少なくとも１つは含まれている．

証明

任意のスケジューリングの解 a と同じか評価の良い解 A を作る，

計算ブロックの左右の関係について計算ブロックを左につめられるだけ詰める．

計算ブロック（再構成が必要なく連続している計算ブロックは１つとする.）と壁 (s) を頂点とする．ある計算ブロック β が左に詰めたときにはじめて当たるブロックを α とする．２つ以上の計算ブロックに同時に当たるときには，適当に１つのブロックを選び計算ブロック間に枝 (α,β) をつける．また，壁に当たるときには，計算ブロックと壁の間に枝 (s,β) をつける．この操作を繰り返し行い s を根とする（外向）木を作る．作られた木の頂点ラベルを壁 s からの距離により付け直す.（図 5.5）

sequence-quintuple の順列 Γ_1，と Γ_2 を作る．を，s から開始し，隣接関係にある頂点の中から辞書順で最も前にくるものを選択し，頂点名を列挙する．選択された頂点に隣接関係のある頂点の中から辞書順でもっとも前にくるものを選び頂点名を列挙する．この操作を繰り返すことにより Γ_1 を作る．図 5.5 の例の場合，始めに s と隣接関係のある頂点 a,b, と c の中から辞書順で a を選択し列挙する．次に,a と隣接関係のある d と e を辞書順に列

挙する．繰り返すことにより $\Gamma_1 = (a, d, e, b, c, g, h, i)$ を得る．

s から開始し，隣接関係にある頂点の中から辞書順で最も後にくるものを選択し，頂点名を列挙する．選択された頂点に隣接関係のある頂点の中から辞書順でもっとも後にくるもの選び頂点名を列挙する．この操作を繰り返すことにより Γ_2 を作る．例の場合，始めに s と隣接関係のある頂点 a,b, と c の中から辞書の逆順で c を選択し列挙する．次に,c と隣接関係のある g,h と i を辞書の逆順に列挙する．繰り返すことにより $\Gamma_2 = (c, i, h, g, b, f, a, e, d)$ を得る．

ここで求められた順列 Γ_1，と Γ_2 から，x 座標を決定する．このことは，始めのアルゴリズムで解を求めたことと同じである．

同様に上下関係についても行う．

Γ_5 で表される時間軸方向については，時刻 0 の方向に計算ブロックが他の計算ブロックに当たるまで詰める．他の計算ブロックに当たるときは，データの依存関係があるか，または 2 つのブロックの衝突である．与えられた Dependence Graph のデータ依存関係に，すべての衝突する計算ブロックの対 α と β についての枝 (α,β) をつけ加える．有効サイクルは，計算ブロック β が時刻 0 方向詰め α に衝突し，また α が時刻 0 方向詰め β に衝突することが無いので存在しない．よって Γ_5 は，トポロジカル順序を満たしている．

この方法で構成した，$\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, \Gamma_5$ からデコードして得られる再構成スケジューリング A の横，縦と時間の大きさは，a のそれぞれの大きさより大きくない．よって，その中に最適解が少なくとも 1 つ以上存在する．□

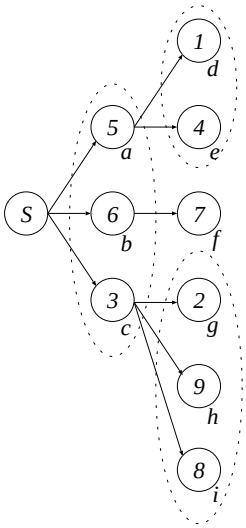

図 5.5: 外向木

5.4 Simulated Annealing法による探索

Simulated Annealing法は，注目している解に対し適当に隣接解を１つ求めて比較し，条件を満たしていればこれを注目している解に置き換えていく操作を繰り返すことで，よい解を求めて解空間を探索する手法である．

解空間内に，同一の解（冗長解）がたくさん存在すると，その解ばかりを多数回も探索することになり，効率的に探索することができなくなってしまう．[7],[8]

到達可能性

Simulated Annealing法では任意の１つの初期解から探索を始めるため，任意の許容解から任意の許容解まで，問題サイズの多項式回数の隣接解移動操作により到達できることが望まれる．ある初期解からはじめると（非許容解に拒まれるなどの原因などで）最適解に到達できない場合などは，Simulated Annealing法の特徴が失われてしまう．

5.4.1 $feasible$の定義

Γ_5が与えられたグラフGのトポロジカル順序を満足するときに，Sequence Quintupleは$feasible$な解である．

5.4.2 隣接解の定義

１．$\Gamma_i(i = 1, 2, 3, 4)$から任意の順列１つを選び，その中の任意の２つのラベルを選択し交換する．

２．Γ_5の中から任意のラベルAを選びAと交換してもトポロジカル順序を満足するようなラベルの候補を列挙する．その候補の中から１つ選びAと交換する．

定理

任意の$feasible$なSequence Quintupleから，$feasible$な解への移動だけで，任意の$feasible$なSequence Quintupleへ到達可能である．

証明

任意の$feasible$な順列$A = (a_1, a_2, \cdots, a_n)$から，ある$feasible$な順列$B = (b_1, b_2, \cdots, b_n)$へ$feasible$な解の移動だけで到達可能であることを証明する．

はじめに入力のグラフ G において outdegree が 0 のラベル $b_n = a_i$ に注目する．
b_n の outdegree が 0 であるのは，B が $feasible$ であるためである．もし，b_n の outdegree が 0 でないならば B は，トポロジカル順序が満足されないので矛盾が生じる．
a_i は outdegree が 0 であるため，a_{i+1} は，トポロジカル順序を満足して a_i と交換できる候補であり，かつ隣接解 2 より交換可能である．
つまり，$(a_1, a_2, \cdots, a_i, a_{i+1}, \cdots, a_n)$ から $(a_1, a_2, \cdots, a_{i+1}, a_i, \cdots, a_n)$ への a_i の移動である．
よって，$(a_1, a_2, \cdots, a_{i+1}, a_i, \cdots, a_n)$ は $feasible$ である．
同様の手順を繰り返し，$(a_1, a_2, \cdots, a_{i-1}, a_{i+1}, \cdots, b_n)$ を得る．

次に, $(a_1, a_2, \cdots, a_j, \cdots, a_k, a_{k+1}, \cdots, a_n)$ と $(b_1, b_2, \cdots, b_k, b_{k+1}, \cdots, b_n)$ は，$(a_{k+1}, \cdots, a_n)$ と $(b_{k+1}, \cdots, b_n)$ の並びは同じであると仮定する．
ここで，$b_k (= a_j)$ は，outdegree が 0 であるか，$(b_{k+1}, \cdots, b_n)$ のどれかにしか枝が存在しない．(B はトポロジカル順序を満足しているためである．)

今,a_j は outdegree が 0 であるか $(a_{k+1}, \cdots, a_n)$ の部分集合にしか枝が存在しない．よって a_{j+1} は，トポロジカル順序を満足して a_j と交換できる候補であり，かつ隣接解 2 より交換可能である．このとき，$(a_1, a_2, \cdots, a_{j+1}, a_j, \cdots, a_k, a_{k+1}, \cdots, a_n)$ は $feasible$ である．

同様の手順を繰り返し，
$(a_1, a_2, \cdots, a_{j-1}, a_{j+1}, \cdots, a_k, a_j, a_{k+1}, \cdots, a_n)$ から $(a_1, a_2, \cdots, a_{j-1}, a_{j+1}, \cdots, a_k, b_k, b_{k+1}, \cdots, b_n)$ を得る.
帰納法により A から B へ到達可能である．□

第 6 章

計算機実験

6.1 実験モデル

Simulated Annealing 法を用いて解空間を探索するプログラムをC言語を用いて実装した．入力は，Dependence Graph と横の大きさ (x)，縦の大きさ (y) と，演算ステップ数 (t) で表される計算ブロックの大きさとする．出力は，計算ブロックの 3 次元パッキング後に $x \times y \times t$ で求められるパッケージの大きさとする．評価を出力で得られたパッケージの大きさとする．実験は，入力 Dependence Graph に楕円フィルタとカウンタの例を用いた．それぞれの入力パラメータを以下に示す．

楕円フィルタ

入力に用いた楕円フィルタの Dependence Graph を図 6.1 に示す．図 6.1 中の記号＋は加算器，＊は乗算器を示す．

計算ブロック数は合計３４．演算の種類は加算器と乗算器の２種類であり，それぞれの構成される計算ブロック数は，加算器２６，乗算器８とする．再構成に必要な時間は加算器と乗算器ともに１とする．計算ブロックの大きさの x，y と，t を，表 6.1 に示す．データは，$a1$ から $a7$ の７種類である．

また，パッキングのとき x，y で表される平面と，t で表す時間方向とで行っているが，計算ブロックの y 軸方向の配置を考えず，x と t 方向だけを用いてパッキングを行った２次元パッキングの実験も行った．そのときの，計算ブロックの大きさを，表 6.2 に示す．データは，$b1$ から $b5$ の５種類である．

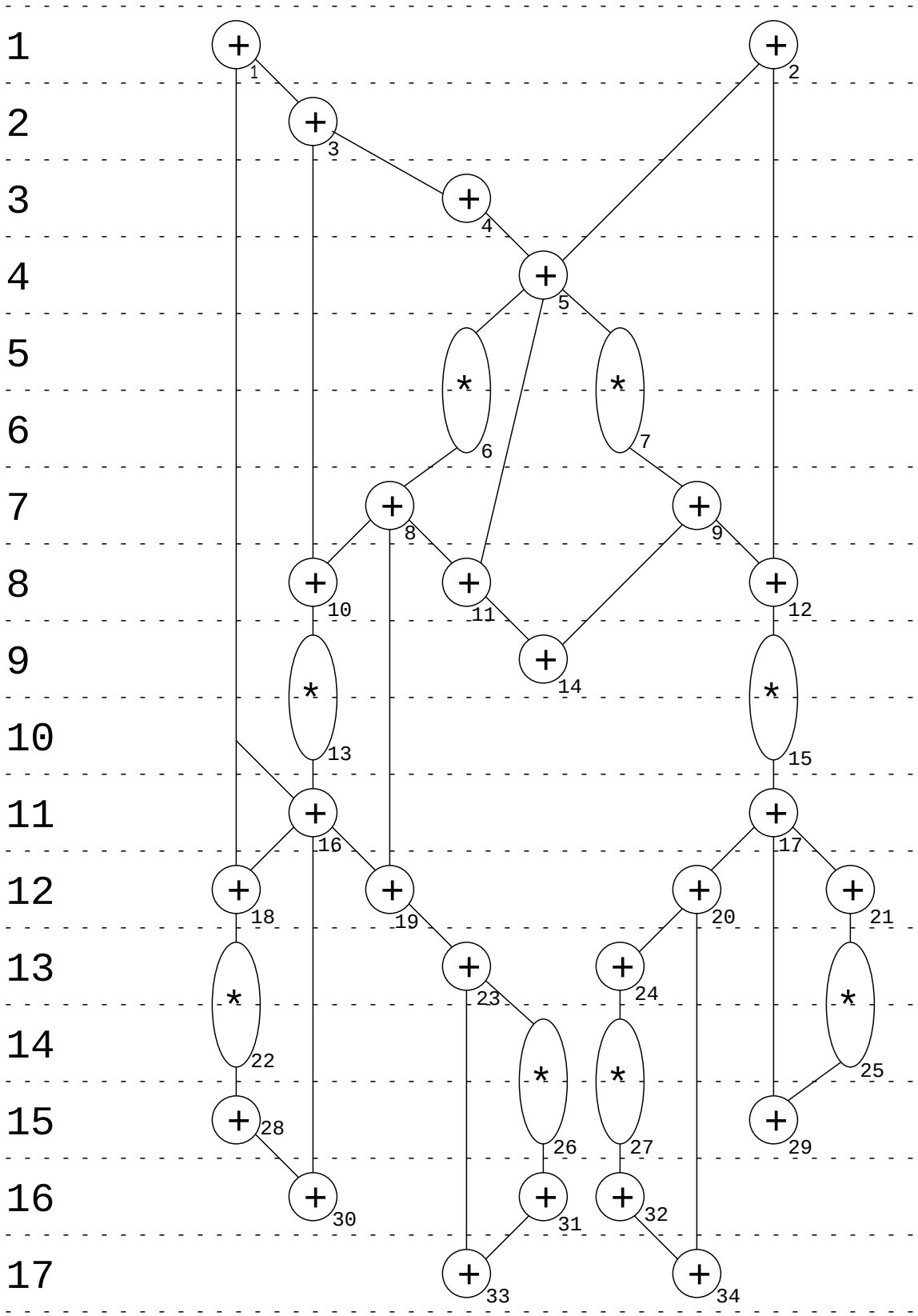

図 6.1: 楕円フィルタの Dependence Graph

表 6.1: 楕円フィルタ入力データ 1

	加算器			乗算器		
データ名	x	y	t	x	y	t
$a1$	1	1	1	1	1	1
$a2$	1	1	1	1	1	2
$a3$	1	1	1	1	1	4
$a4$	1	1	1	1	2	4
$a5$	1	1	1	4	4	1
$a6$	1	1	1	4	4	2
$a7$	1	1	1	4	4	4

表 6.2: 楕円フィルタ入力データ 2

	加算器		乗算器	
データ名	x	t	x	t
$b1$	1	1	1	1
$b2$	1	1	1	2
$b3$	1	1	1	4
$b4$	1	1	4	2
$b5$	1	1	4	4

カウンタ

計算ブロック数は合計４３．演算の種類は３種類であり，それぞれの構成される計算ブロック数は，演算タイプ１を２０，演算タイプ２を８，演算タイプ３を１５．再構成に必要な時間はすべて１とする．計算ブロックの大きさを表6.3に示す．また楕円フィルタの実験と同様に２次元パッキングの実験も行った．そのときの，計算ブロックの大きさを表6.4に示す．

表 6.3: カウンタ入力データ１

	タイプ１			タイプ２			タイプ３		
データ名	x	y	t	x	y	t	x	y	t
$c1$	1	1	1	1	1	1	1	1	1

表 6.4: カウンタ入力データ２

	タイプ１		タイプ２		タイプ３	
データ名	x	t	x	t	x	t
$d1$	1	1	1	1	1	1

6.2 実験結果

楕円フィルタの入力に対しての結果を，表 6.5 と表 6.6 に示す．また，カウンタの入力に対しての結果を，表 6.7 と表 6.8 に示す．結果は，計算ブロックをパッキングした後のパッケージの横の大きさを x，縦の大きさを y，演算ステップ数と，の $x \times y \times$演算ステップ数で表されるパッケージの大きさを示す．

表 6.5: 楕円フィルタ入力データ 1 の結果

データ名	x	y	t	$x \times y \times t$
$a1$	3	4	15	180
$a2$	3	3	18	162
$a3$	3	3	24	216
$a4$	3	4	24	288
$a5$	5	7	16	560
$a6$	5	6	22	660
$a7$	6	8	27	1296

表 6.6: 楕円フィルタ入力データ 2 の結果

データ名	x	t	$x \times t$
$b1$	4	16	64
$b2$	4	20	80
$b3$	3	24	87
$b4$	7	22	154
$b5$	8	33	264

表 6.7: カウンタ入力データ 1 の結果

データ名	x	y	t	$x \times y \times t$
$c1$	4	5	7	140

表 6.8: カウンタ入力データ 2 の結果

データ名	x	t	$x \times t$
$d1$	5	12	60

実験により得られた計算ブロックのパッキング図を示す．楕円フィルタ入力データ b1 に対する 2 次元パッキングの結果を図 6.2 に示す．

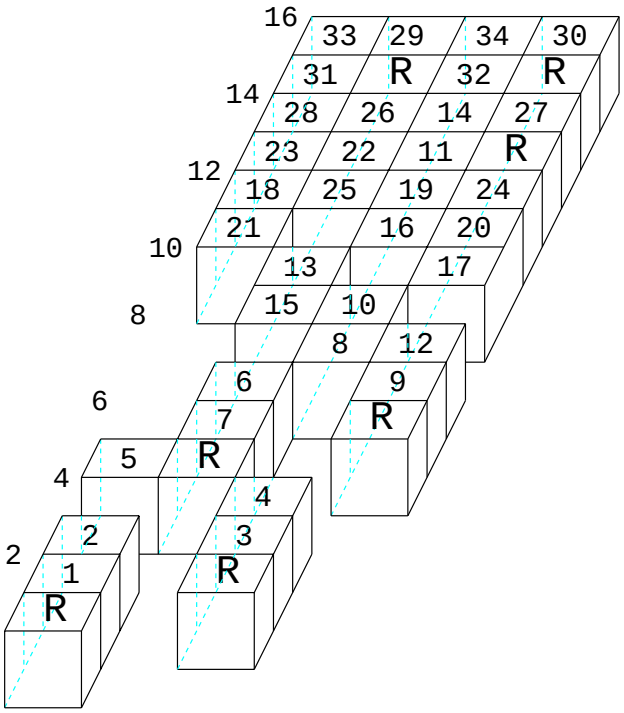

図 6.2: 入力データ b1 のパッキング図

カウンタ入力データ d1 に対する 2 次元パッキングの結果図 6.3 に示す.

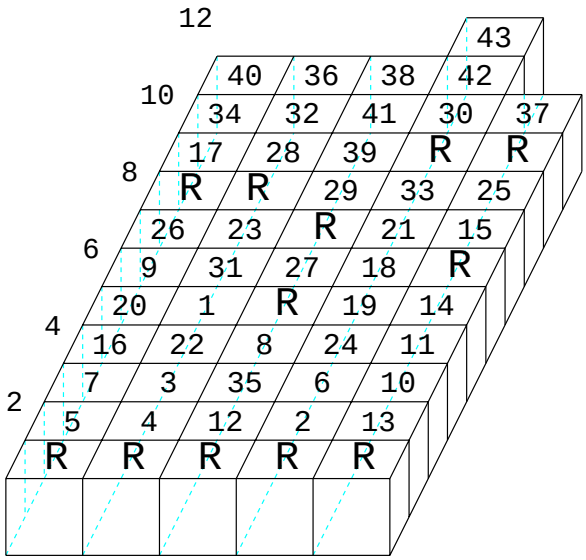

図 6.3: 入力データ d1 のパッキング図

6.3 実験の考察

実験により，冗長な解の多さから Simulated Annealing 法のような探索的な手法では，たどり着きにくい最適解があることが明らかになった．最適解のパッキング形状が時間軸方向に 1 列に並ぶなどの特殊な場合である．Simulated Annealing 法は，解空間を探索するに従い値の悪い解が受け入れられにくくなることから，解の評価がパッキング後の $x \times y \times t$ では，図 6.4 の step1 から step3 へたどり着きにくいためである．

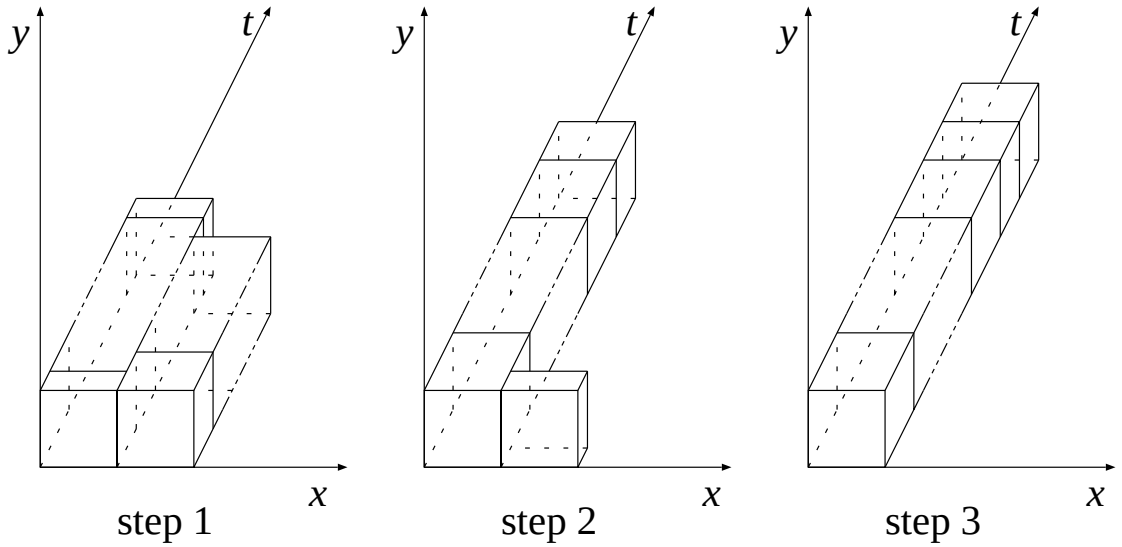

図 6.4: たどり着きにくい解

第 7 章

まとめ

動的再構成スケジューリング問題の複雑さを明らかにする目的で，計算ブロックの平面上への配置を同時に実装可能な計算ブロックの数に置き換えて理論的な考察を行い，実装可能な計算ブロック数を１，計算ブロックの種類の数を２とした問題に対して最短スケジュールを求めるアルゴリズムを導いた．計算ブロックの種類の数が１である時は，再構成を考慮しない通常の並列スケジューリング問題に帰着される．同時に実装可能な計算ブロック数が２以上の問題，計算ブロックの種類の数が３以上の問題の複雑さは今後の課題となっている．

また，計算ブロックの平面上での構成位置決定を含めた動的再構成スケジューリング問題に対して，確率的解空間探索に基づく手法を提案した．各計算ブロックをその平面的な広がりと時間的生存期間よりなる３次元直方体と見なし，それらを FPGA の平面的広がりと時間軸からなる３次元空間へパッキングする問題としてとらえることとした．次いで直方体の３次元パッキング解空間の表現として提案されている sequence-quintuple を基礎とし，先行制約グラフにて指定された演算間の先行関係と計算ブロックの再構成を反映した直方体の時間軸方向の伸び縮みを考慮した解表現（解のコード化）を提案した．これは，５つの計算ブロック名順列を使うものであり，その中の２つの順列にて FPGA 上の x 座標を，他の２つにて y 座標をそれぞれ算出し，先行制約グラフのトポロジカル順序に限定した第５の計算ブロック名順列と計算ブロック間の $x-y$ 平面上での重なりから，計算ブロックの再構成時刻，演算の実行時刻を算出するものとなっている．

実験結果より FPGA の xy 平面上における計算ブロックの配置を，１次元的な x 方向だけとしてパッキングする２次元パッキングの実験から良質の解が得られた．このことから，３次元パッキング空間の探索においても隣接解の定義に制約を加えることなどを行

い，時間方向への重なりの機会を増やすことで良質の解が得られることが予想される．

今後の課題として，本手法を基礎により良質の解を得るための解評価手法，隣接解生成手法の検討が必要である．

また実際の計算実行では，各計算ブロック間でのデータ受け渡しが必要である．計算ブロックから演算後に出力されるデータを，レジスタブロックを構成することで保持する機構など，データ受け渡し手法の開発が今後の課題となっている．

謝辞

本研究を行うにあたり，日頃から温かく御指導をしていただきました金子峰雄助教授ならびに田湯智助手に深く感謝の意を表します．また，有益な御助言や御検討いただきました金子研究室の皆様に感謝いたします．

参考文献

[1] 羽切　崇，戸川　望，柳沢政生，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 ”，電子情報通信学会技術研究報告，VLD99 Vol.99, No.658，PP7-14.

[2] 石飛　貴志，戸川　望，柳沢政生，大附辰夫，”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 ”，電子情報通信学会技術研究報告，VLD2000 Vol.100，No.531，PP33-40.

[3] Douglas Chang and Malgorzata Marek-Sadouska, ”Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs,” *IEEE Transaction on Computer*, Vol.48, No.6, pp565-578, 1999.

[4] Karthikeya M. Gajjala Purna and Dinesh Bhatia, ”Temporal Partitioning and Scheduling Data Flow Graphs for Reconfigurable Computer” *IEEE Transaction on Computer*, Vol.48, No.6, pp579-590, 1999.

[5] Hiroshi Murata, Kunihiro Fujiyoshi, Shigetoshi Nakatake and Yoji Kajitani, ”VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair” *IEEE Transaction on Computer-Aided Design of Integrated Circuits and System*, Vol.15, No.12, pp1518-1524, December 1996.

[6] Hiroyuki Yamazaki, Keishi Sakanushi, Shigetoshi Nakatake and Yoji Kajitani, ”The 3D-Packing by Meta Data Structure and Packing Heuristics” *IEICE Transaction Fundamentals*, Vol.E83-A, NO.4 April 2000

[7] 藤吉　邦洋, 大村　智一, 井尻　堅大, ”Simulated Annealing 法探索に適した Sequence-Pair によるパッキング解空間 ”，電子情報通信学会技術研究報告，VLD99 Vol.99, No.659，PP9-16.

[8] 大村　智一，藤吉　邦洋，”Sequence-Pair を用いたパッキングにおける矩形回転による面積最小化／－局所的なスライス構造の利用－ ”，電子情報通信学会技術研究報告，VLD99 Vol.99，No.659，PP17-24.

[9] Hesham El-Rewini, T. G. Lewis, Hesham H. Ali, Task Scheduling in Parallel and Distributed Systems (Prentice Hall Series in Innovative Technology)

[10] Eduardo Sanchez, Moshe Sipper, Jacques-Olivier Haenni, Jean-Luc Beuchat, Andre Stauffer and Andres Perez-Uribe, ”Static and Dynamic Configurable System,” *IEEE Transaction on Computer*, Vol.48, No.6, pp556-564, 1999.