

補償パスを用いない $1 - \frac{1}{2}$ トラックスイッチアレーの再構成について

重井徳貴† 宮島廣美††

† 島根大学 総合理工学部
〒690-8504 松江市西川津町 1060
E-mail: shigei@cis.shimane-u.ac.jp
†† 鹿児島大学 工学部

あらまし

本報告では、2行2列の予備プロセッサをもつ $1 - \frac{1}{2}$ トラックスイッチアレーについて議論している。初め、補償パスに基づくいかなる再構成法も再構成に失敗するような再構成可能な故障パターンが存在することが示される。それから、逐次的ルーティングに基づく再構成手法が与えられる。この手法の時間計算量はプロセッサ数の指数オーダーであるが、再構成可能な全ての故障パターンにおいて再構成に成功することが示される。

キーワード プロセッサアレー, 再構成, 補償パス, ルーティング

Reconfiguration of Processor Arrays Based on Sequential Routing

Noritaka SHIGEI† Hiromi MIYAJIMA††

† Shimane University
Matsue 690-8504, JAPAN
E-mail: shigei@cis.shimane-u.ac.jp
†† Kagoshima University

Abstract

This study considers a processor array model using single-and-half-track switches which has two spare columns and two spare rows. First, it is shown that there exists a reconfigurable fault pattern which any reconfiguration method based on compensation path cannot relieve. Then, a reconfiguration method based on sequential routing is presented. Although the method requires an exponential time complexity, it can relieve any reconfigurable fault pattern.

key words processor array, reconfiguration, compensation path, routing

1 はじめに

一つのシステムを大面積半導体上に一括構築することは、高速化、省電力化、大規模化等の多くの利点を持つが、その実現のためには、必ず発生する欠陥箇所を救済するための何らかの方策を施さなければならない。このことから、相互結合網として格子結合をもつ、プロセッサアレーを対象として、これまでに多くの再構成技術についての研究が行われてきた [1]-[3]。再構成を行うために予備のプロセッサ (PE) やスイッチング回路等の冗長なハードウェアを付加したモデルが幾つか提案されており、これらのモデル上では、故障 PE が発生した場合にも PE 間の結合を切り替えることで、正常な PE のみからなるアレーを得ることができる。また、再構成を行うためには、正常な PE のみをどのように結合したらいいか決めるためのアルゴリズムが必要である。再構成アルゴリズムには、ある故障パターンが与えられたときに、正常な PE のみからなる結合パターンを見つけることが要求される。

$1 - \frac{1}{2}$ トラックスイッチモデルのために多くの再構成アルゴリズムが考えられているが、最も簡単なモデルと考えられるこのモデルの場合でさえ、スイッチング回路を切り替えることで実現できる結合パターンの数は非常に多く複雑である。このことから、ほとんどの再構成アルゴリズムでは結合のパターンを制限し、その中から正常 PE のみからなる結合パターンを探し出す [1]-[2]。これに対し、沼田らは 1 行 1 列の予備 PE を付加したモデルに対し、結合パターンを制限しない、最適な再構成を行なうものを提案した [3]。

本報告では、2 行 2 列の予備 PE をもつ $1 - \frac{1}{2}$ トラックスイッチモデルについて考える。初め、補償バスに基づくいかなる再構成法も再構成に失敗するような、再構成可能な故障パターンが存在することが示される。それから、逐次的ルーティングに基づく再構成手法が与えられる。この手法の時間計算量は PE 数の指数オーダーであるが、再構成可能な全ての故障パターンにおいて再構成に成功することが示される。また、計算機シミュレーションにより、補償バスに基づく手法と逐次ルーティングに基づく手法の再構成率が比較される。

2 準備

2.1 アレーの再構成

以下では変数は特に断ることなく、整数を仮定する。ある正整数 n が与えられるとき、 $[n]$ は集合 $\{0, 1, \dots, n\}$ を表すものとする。

N 行 N 列の正常なプロセッシングエレメント (PE) からなる格子結合を得るために、格子点上に配置された $N + 2$ 行 $N + 2$ 列の PE を使用するものとする。ここで、物理アドレスを以下のように定義する。

[定義 2.1.1] 物理アドレスの全集合を $\mathcal{P} = \{(x, y) | x, y \in [N + 1]\}$ とし、座標 (x, y) 上に位置する PE には物理アドレス (x, y) が付される。□

以下では、物理アドレス (x, y) が付された PE のことを単に (x, y) と呼ぶ。また、 $\{(0, y) | y \in [N + 1]\}$ を最も左の列上に位置するものとし、 $\{(x, 0) | x \in [N + 1]\}$ を最も上の行上に位置するものとする。

[定義 2.1.2] 論理アドレスの全集合を $\mathcal{L} = \{[x, y] | x, y \in [N - 1]\}$ とする。論理アドレス $[x, y]$ と $[x', y']$ をもつ二つの PE は、 $|x - x'| + |y - y'| = 1$ であるならば論理的に隣接しているという。□

アレーの再構成は、論理アドレスの物理アドレスへのマッピングであると考えられる。

[定義 2.1.3] ある故障パターン f において、以下の条件を全て満たすような、 $\mathcal{L}$ から $\mathcal{P}$ への単射である写像関数 Ψ が与えられるならば、再構成は成功したという。

- 全ての $l \in \mathcal{L}$ において、 $\Psi(l)$ は正常である。
- 全ての論理的に隣接する PE 間を結合するルーティングが存在する。

□

[定義 2.1.4] ある故障パターン f において、再構成に成功する写像関数 Ψ が存在するならば、 f は再構成可能な故障パターンであるという。□

ここで、定義 2.1.3 の 2 番目の条件は、対象とするプロセッサアレーのスイッチやトラック等のルーティング機構に依存することに注意する。

2.2 $1 - \frac{1}{2}$ トラックスイッチモデル

$1 - \frac{1}{2}$ トラックスイッチモデルは、PE を相互に結合するためのルーティング機構として、リンク、トラック、

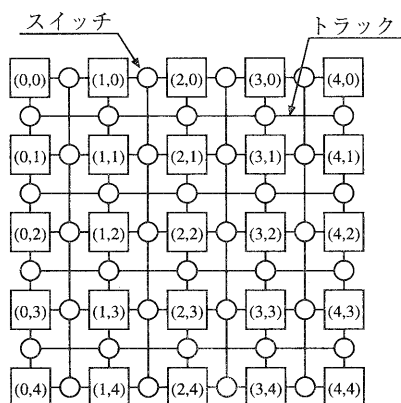


図 1: $1 - \frac{1}{2}$ トラックススイッチモデル ($N = 3$)

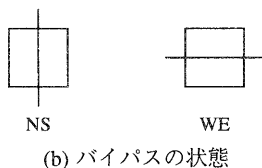
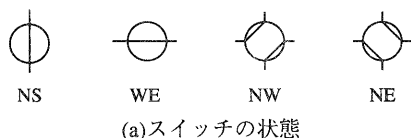


図 2: スイッチとバイパスの状態

スイッチを備える。各 PE は上下左右 4 方向へ伸びるリンクをもち、それらのリンクは隣接する PE 間に存在するスイッチに接続している。また、トラックは各行各列間に 1 本ずつ配置され、各トラック上には $N + 2$ 個のスイッチが存在する (図 1)。スイッチは図 2.(a) のような 4 つの状態を、PE は 2.(b) のような 2 つのバイパスの状態をとることができ、これらを切替えることにより、PE 間の結合関係を変化させることができる。ルーティング機構の構造から以下の補題および定理が成り立つ [3]。

[補題 2.2.1] $1 - \frac{1}{2}$ トラックススイッチモデルで、 (x, y) の上方向のリンクは $\{(x', y - 1) | x' \in [N + 1]\}$ の下方向の

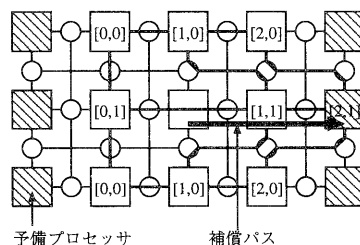


図 3: 補償パス上のシフト

リンクに対してのみ接続可能である。他方向のリンクについても同様。 □

[補題 2.2.2] $\Psi[x, y] = (x', y')$, $\Psi[x + 1, y] = (x'', y'')$ であるならば $x' < x''$ である。 y についても同様。 □

[定理 2.2.1] 写像関数 Ψ に対し実現可能なルーティングが存在するためには、 $\Psi[x, y]$ がとる値は、 $\{(x + i, y + j) | i, j \in [2]\}$ のいずれかでなければならない。 □

2.3 補償パスによる再構成

補償パスによる再構成では、初期状態として、 $N + 2$ 行 $N + 2$ 列の物理アレーのうち、任意の 2 行 2 列のプロセッサが予備として扱われ、予備でない N 行 N 列の PE に対し論理アドレスがマッピングされる¹。論理アドレスのマッピングは、論理的に隣接しているもの同士が、同一行もしくは同一列上で最も近いものになるように行われる。初期状態において予備でない PE が故障している場合、その PE から同一行もしくは同一列に位置する正常な予備の PE に向かって論理アドレスの置換を行うことで、論理アレーからその故障 PE を除外することができる。予備でない故障 PE(x_1, y_1) から正常な予備の PE(x_2, y_2) へ向かう有向パスは補償パスと呼ばれ、 $(x_1, y_1) \rightarrow (x_2, y_2)$ と記す。ただし、パスは上下左右の 4 方向のいずれかであり、 $x_1 = x_2$ もしくは $y_1 = y_2$ でなければならない。ある故障 PE に対し補償パスが与えられると、以下に定義されるような補償パス上のシフトを行うことで、論理アレーからパスの起点となる故障プロセッサを除外することができる。

¹予備の行と列をどれにするかで、再構成の結果は異なるが、本報告では、いずれかの予備配置で再構成に成功するならば、補償パスによる再構成は成功したとみなす。

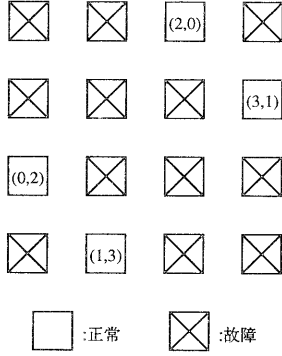


図 4: 補償パスによる再構成法で再構成できない再構成可能な故障パターン

[定義 2.3.1] 補償パス $u_0 \rightarrow u_n$ が与えられ, $u_0 \rightarrow u_n$ 上のプロセッサを u_0 に近い順に $u_0, u_1 \dots u_n$. また, Ψ は, $i \in [n-1]$ において, $\Psi(u_i) = u_i$ となっているとする. このとき, Ψ を, $i \in [n-1]$ において, $\Psi(u_i) = u_{i+1}$ とすることを, 補償パス上の論理的なアドレスの置換といい, 補償パス上のシフトと呼ぶ. $\square$

補償パス上のシフトが行われた状態でのルーティングの様子を図 3 に示す. 補償パス上のシフトは組合せによって, ルーティングができないものが存在することが示されている. したがって, 補償パスによる再構成は, 初期状態における全ての予備でない故障 PE に対し, 実現可能 (ルーティングが可能) な補償パスを与える問題と言い替えることができる.

[定理 2.3.1] 再構成に成功する写像関数が少なくとも 1 つ存在する故障パターンの集合を $\mathcal{R}$ とする. 補償パスに基づく全ての写像関数が再構成に失敗する補償パターン $f \in \mathcal{R}$ が存在する. $\square$

(証明) 図 4 のような, $\mathcal{H} = \{(2,0), (3,1), (0,2), (1,3)\}$ が正常であり, その他が故障している $N=2$ の故障パターン g を考える. g が再構成に成功する写像関数を持ち, 補償パスに基づく全ての写像関数で再構成に失敗することを示そう.

まず, 正常な PE 数が 4 であるため写像関数は全単射でなければならないことと, ルーティングの性質より, 以下のことがいえる.

- (2,0) は $[\ast, 0]$ でなければならない.

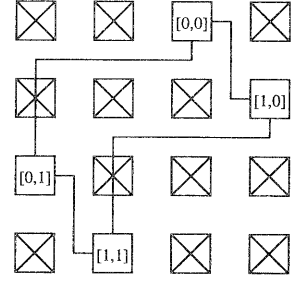


図 5: Ψ_I のルーティング

- (3,1) は $[1, \ast]$ でなければならない.
- (0,2) は $[0, \ast]$ でなければならない.
- (1,3) は $[\ast, 1]$ でなければならない.

ここで, “ $\ast$ ” は 0 と 1 のどちらでもよいことを表す. (2,0) に $[0,0]$ をマッピングする場合をケース I, (2,0) に $[1,0]$ をマッピングする場合をケース II とし, それぞれの場合において再構成に成功する写像関数を Ψ_I , Ψ_{II} とする. Ψ_I と Ψ_{II} が存在し, ユニークに決められることを示す.

ケース I の場合, $\Psi_I[0,0] = (2,0)$ であり, (3,1) が (2,0) の右側に位置する唯一の PE であることから, $\Psi_I[1,0] = (3,1)$ と決まる. また, 写像関数は単射であることから, $\Psi_I[0,1] = (0,2)$, $\Psi_I[1,1] = (1,3)$ と決まる. この写像関数 Ψ_I は図 5 に示されるようにルーティングすることで再構成に成功する. Ψ_I が存在し, ユニークに決められることが示された. ケース II の場合も同様に, $\Psi_{II}[1,0] = (2,0)$, $\Psi_{II}[0,0] = (0,2)$, $\Psi_{II}[0,1] = (1,3)$, $\Psi_{II}[1,1] = (3,1)$ と決まる.

補償パスに基づく全ての写像関数で再構成に失敗することを証明するためには, Ψ_I と Ψ_{II} を実現できるようなスペアラインの配置が存在しないことを示せばよい. 以下では, Ψ_I の場合についてのみ示すが, Ψ_{II} の場合も同様に証明できる.

まず, 第 3 列をスペアラインとすることはできないことを示そう. 第 3 列をスペアラインにしたと仮定する. このとき, $\Psi_I[1,0] = (3,1)$ とするためには, 補償パス $(2,1) \rightarrow (3,1)$ を用いなければならない. これにより, $\Psi_I[0,0] = (2,0)$ とするためには, 補償パス $(0,1) \rightarrow (2,0)$ もしくは $(1,1) \rightarrow (2,0)$ を用いなければ

ならないが、これらはいずれも斜め方向のパスであり、実現できない。よって、第3列目はスペアラインとすることはできない。次に、第2列もスペアラインとすることはできないことを示す。第2列をスペアラインにしたと仮定する。このとき、 $\Psi_I[0,0] = (2,0)$ とするためには、補償パス $(1,0) \rightarrow (2,0)$ を用いなければならない。第1列と第3列はスペアラインにできないことから、第0列がスペアラインに決まる。そして、 $\Psi_I[0,1] = (0,2)$ とするためには、補償パス $(1,2) \rightarrow (0,2)$ を用いなければならない。これにより、 $\Psi_I[1,1] = (1,3)$ とするためには、補償パス $(2,2) \rightarrow (1,3)$ もしくは $(3,2) \rightarrow (1,3)$ を用いなければならないが、これらはいずれも斜め方向のパスであり、実現することはできない。よって、第2列もスペアラインとすることはできない。スペアラインの候補は、第0列と第1列にしばらくは限られるが、これらを同時にスペアラインとすることは、 $(0,2)$ を利用不可能とする。よって、第0列と第1列もスペアラインとできず、どのようなスペアラインの組合せでも、 Ψ_I を与えることはできない。 □

3 逐次的ルーティングに基づく再構成法

3.1 アルゴリズム

ここで与えられるアルゴリズムは、実現可能なルーティングが存在するような写像関数を深さ優先で探索する。写像関数の値は、 $\Psi[0,0], \Psi[1,0], \dots, \Psi[N-1,0], \Psi[0,1], \dots, \Psi[0,N-1], \dots, \Psi[N-1,N-1]$ というような行順列主体の順番で一つ一つ決められる。もしその途中で、ある割り当てに対してルーティングができないような行き止まりの状態になった場合、一つ前の割り当てに戻り、別の割り当てが試みられる。もし $\Psi[N-1,N-1]$ の値が与えられアルゴリズムが終了するならば、望まれるような写像関数が発見されたことになる。

[定義 3.1.1] $x \in [N-1], y \in [N-1]$ である任意の x, y において、行順列主体の順番において論理アドレス $[x, y]$ の次のアドレスを $[x, y]_{next}$ と記す。すなわち、 $x \neq N-1$ のとき、 $[x, y]_{next} = [x+1, y]$ であり、 $x = N-1$ のとき、 $[x, y]_{next} = [0, y+1]$ である。また、 $[x, y]_{next} = [x', y']$ であるとき、 $[x', y']_{prev} = [x, y]$ と

記す。 □

[定義 3.1.2] $x, y \in [N-1]$ である任意の x, y において、論理アドレスの集合 $\mathcal{L}_{[x,y]}$ を $\mathcal{L}_{[x,y]} = \{[i_1, j_1], [i_2, y] | i_1 \in [N-1], j_1 \in [y-1], i_2 \in [x-1]\}$ とする。写像関数 Ψ において、 $\mathcal{L}_{[x,y]}$ に対し写像が定義され、 $\mathcal{L} - \mathcal{L}_{x,y}$ に対し写像が定義されていないとき、単射かつこれを実現可能なルーティングが存在するならば、 Ψ は $[x, y]$ まで正しく定義されたという。 □

[定義 3.1.3] $x, y \in [N-1]$ である任意の x, y において、 Ψ が $[x, y]$ まで正しく定義された状態において、 $\Psi[x, y]_{next} = (x', y')$ と定義することで Ψ が $[x, y]_{next}$ まで正しく定義されるような物理 PE (x', y') を $\Psi[x, y]_{next}$ のマッピング候補といい、マッピング候補の全集合を $\mathcal{C}_{[x,y]_{next}}$ と記す。 □

[定義 3.1.4] $\Psi[x, y] = (x', y')$ であるならば、 $\Psi_x[x, y] = x', \Psi_y[x, y] = y'$ とする。 □

アルゴリズムを手続き FIND_MAPPING として以下に与える。

手続き FIND_MAPPING

初期状態: Σ をアレーの状態を表す変数の集まりとし、PE の状態 (故障, 正常)、バイパス機構の状態、スイッチの状態を保持するものとする。初期状態では、PE の状態には現在の状態が、バイパス機構とスイッチ状態にはルーティングのために使用されていないことを示す特別な状態がセットされているものとする。

Step 1: 正常な PE 数が N^2 より小さいならば、明らかに再構成に成功する写像関数は存在しない。アルゴリズムは終了。

Step 2: ASSIGN(0,0, Σ) を呼び出す。もし、ASSIGN(0,0, Σ) が TRUE を返したら再構成に成功する写像関数が発見された。さもなくば、再構成に成功する写像関数は見つからなかった。 □

手続き ASSIGN(x, y, Σ)

$\Psi[x, y]$ の値を定義し、手続き ASSIGN を再帰的に呼び出す。

Step 1: $\Psi[x, y]$ のマッピング候補の全集合 $\mathcal{C}_{[x,y]}$ を求める。

Step 2: n を $\mathcal{C}_{[x,y]}$ から一つ選び、 $\Psi[x, y] = n, \mathcal{C}_{[x,y]} = \mathcal{C}_{[x,y]} - \{n\}$ とする。 Σ のコピーを作り、 Σ' とする。

Step 3: 手続き Route(x, y, Σ') を呼び出し、ルーティングを行い、 Σ' に反映させる。手続き Route が FALSE を返してきたときは、FALSE を返す (ルーティ

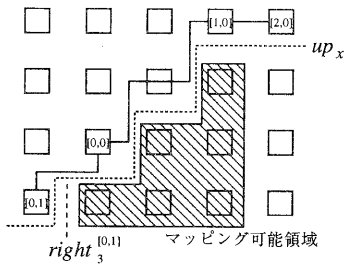


図 6: マッピング可能領域

ングができなかった場合に, FALSE が返される).

Step 4: $x = N - 1$ かつ $y = N - 1$ ならば, TRUE を返す.

Step 5: $x = N - 1$ ならば, $\text{ASSIGN}(0, y + 1, \Sigma')$ を呼び出す. $\text{ASSIGN}(0, y + 1, \Sigma')$ が TRUE を返したら, TRUE を返す.

Step 6: $\text{ASSIGN}(x + 1, y, \Sigma')$ を呼び出す. $\text{ASSIGN}(x + 1, y, \Sigma')$ が TRUE を返したら, TRUE を返す.

Step 7: $C_{[x,y]} = \phi$ ならば, FALSE を返す. さもなくば, Σ' を捨て, Step1 へ. $\square$

3.2 写像関数のマッピング候補

以下の定義は, 図 6 に示されるように, マッピング可能な PE 群を特定するために用いられる.

[定義 3.2.1] 第 x 列において, マッピングもしくは NW 状態の PE のうち最も下側に位置するものの行位置を up_x と記す. $\square$

[定義 3.2.2] Ψ が $[x, y]$ まで正しく定義されているとき, $right_{y'}^{[x,y]}$ を以下のように定義する. $y' \in [\Psi_y[x, y]] - [\Psi_y[x, y - 1]]$ のとき, マッピングもしくは NS 状態の PE のうち最も右側に位置するものの列位置. $y' \in [y + 2] - [\Psi_y[x, y]]$ のとき, $\Psi_x[x, y]$. $\square$

写像関数のマッピング候補について, 以下の補題が成り立つ.

[補題 3.2.1] Ψ が $[x, y]_{prev}$ まで正しく定義されているとき, $\Psi[x, y]$ のマッピング候補の集合 $C_{[x,y]}$ は, $x \neq 0, y \neq 0$ のとき $C_{[x,y]} \in \{(x + i, y + j) | i, j \in [2], x + i > right_{y+j}^{[x,y]}, y + j > up_{x+i}, \text{近傍のスイッチが NS や EW でないかつ } (x + i, y + j) \text{ は正常}\}$ であ

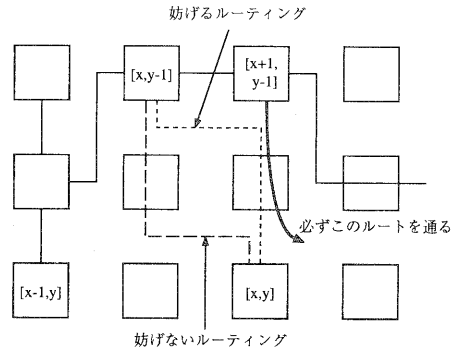


図 7: 妨げるルーティングと妨げないルーティング

る. $\square$
(証明) 補題 2.2.1 より, $[x, y]$ がマッピングされる PE は, $[x - 1, y]$ がマッピングされたものよりも右側に位置し, かつ $[x, y - 1]$ がマッピングされたものよりも下側に位置しなければならない. 近傍のスイッチが NS や EW であるとその PE は明らかに利用不可能である. 論理的な第 $y - 1$ 行上の水平方向のルーティングよりも上にマッピングすることはできない. 論理的な第 $x - 1$ 列上の垂直方向のルーティングよりも左にマッピングすることはできない. $\square$

3.3 ルーティング

アルゴリズムに示されるように, 各論理アドレスのマッピングが行われた直後に, そのマッピングに関するルーティングが逐次的に行われる. これらのルーティングは以降のマッピングとルーティングの選択肢を最大とするように行われる. すなわち, ここで行われるルーティングで再構成できないマッピングは, 他のようなルーティングを用いたとしても再構成失敗となる.

手続き $\text{Route}(x, y, \Sigma)$

Step 1: $y \neq 0$ であるならば, 垂直方向のルーティングすなわち $\Psi[x, y]$ と $\Psi[x, y - 1]$ 間のルーティングを以下の三つの場合ごとに行う.

- $\Psi_x[x, y] = \Psi_x[x, y - 1]$ の時: $\Psi[x, y]$ と $\Psi[x, y - 1]$ 間を縦一直線に結合する.

- $\Psi_x[x, y] > \Psi_x[x, y-1]$ の時: $\Psi[x+1, y-1]$ や $\Psi[x+2, y-1]$ の垂直方向のルーティングを妨げない限り, 上よりルーティングを行い, そのようなルーティングが不可能な場合は FALSE を返す. ここでいう「 $\Psi[x+1, y-1]$ や $\Psi[x+2, y-1]$ の垂直方向のルーティングを妨げない」とは, $\Psi[x+1, y-1]$ や $\Psi[x+2, y-1]$ から下方向のリンクを $\Psi[x, y]$ よりも右側に取り出せる状態のことをいう (図 7).
- $\Psi_x[x, y] < \Psi_x[x, y-1]$ の時: できるだけ上よりルーティングを行う.

Step 2: $x \neq 0$ であるならば, 水平方向のルーティングすなわち $\Psi[x, y]$ と $\Psi[x-1, y]$ 間のルーティングをできるだけ上に行い, TRUE を返す. □

上記のような垂直方向ルーティングが, 以降のマッピングとルーティングの選択肢を狭めないことを示す. [補題 3.3.1] Ψ が $[x, y]_{prev}$ まで正しく定義された状態において, $[x, y]$ がマッピングされる PE がマッピング候補の中から選ばれたとする. $\Psi_x[x, y] = \Psi_x[x, y-1]$ であるならば, $\Psi[x, y]$ と $\Psi[x, y-1]$ を縦一直線に結合するようなルーティングは常に実現可能であり, このようなルーティングが以降のマッピングやルーティングの選択肢を狭めることはない. □

(証明) 補題 2.2.2 より, $\Psi[x, y]$ と $\Psi[x, y-1]$ の間にマッピングやバイパスされたプロセッサは存在しない. よって, このようなルーティングは常に実現可能.

仮に右に凸なルーティングを行ったとすると, 凸な領域の PE をマッピングの候補として使用できなくなり, 以降のマッピングやルーティングの選択肢は明らかに狭まる. 仮に左に凸なルーティングを行ったとしても凸な領域の PE をマッピングの候補として使用することは補題 2.2.2 より不可能であり, 以降のマッピングやルーティングの選択肢が広がることはない. □

[補題 3.3.2] Ψ が $[x, y]_{prev}$ まで正しく定義された状態において, $[x, y]$ がマッピングされる PE がマッピング候補の中から選ばれたとする. $\Psi_x[x, y] > \Psi_x[x, y-1]$ であるとき, $\Psi[x+1, y-1]$ や $\Psi[x+2, y-1]$ の垂直方向のルーティングを妨げないかぎり, 上よりルーティングを行うことは, 以降のマッピングやルーティングの選択肢を狭めることはない. □

(証明) $\Psi[x+1, y-1]$ や $\Psi[x+2, y-1]$ の垂直方向のルーティングを妨げると, 明らかに Ψ を $[x+1, y]$ や $\Psi[x+2, y]$ まで正しく定義できなくなる. また, 下より

にルーティングしたとすると, 水平方向のルーティング位置が下がり, それにより以降のマッピングやルーティングの選択肢を狭める可能性がある. よって, 上記のルーティングを行うことは, 以降のマッピングやルーティングの選択肢を狭めることはない. □

[補題 3.3.3] Ψ が $[x, y]_{prev}$ まで正しく定義された状態において, $[x, y]$ がマッピングされる PE がマッピング候補の中から選ばれたとする. $\Psi_x[x, y] < \Psi_x[x, y-1]$ であるならば, できるだけ上よりルーティングを行うことは, 以降のマッピングやルーティングの選択肢を狭めることはない. □

(証明) 仮に下よりルーティングを行ったとすると, マッピング候補は増えること無しに必ず減ることより, 明らか. □

[補題 3.3.4] Ψ が $[x, y]_{prev}$ まで正しく定義された状態において, $[x, y]$ がマッピングされる PE がマッピング候補の中から選ばれたとする. $\Psi[x, y]_{prev}$ と $\Psi[x, y]$ 間のルーティングをできるだけ上に行うことは, 以降のマッピングやルーティングの選択肢を狭めることはない. □

(証明) 仮に上よりルーティングを行ったとすると, マッピング候補は増えること無しに必ず減ることより, 明らか. □

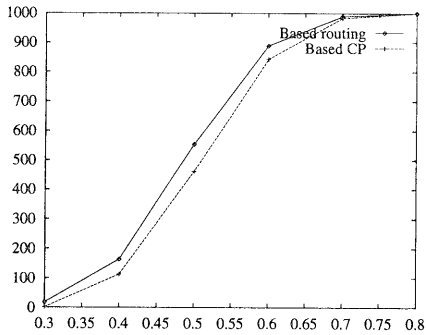
上記 4 つの補題より, 以下の定理が成り立つ.

[定理 3.3.1] 手続き Route がルーティングに失敗するマッピングは, いかなる方法を用いたとしてもルーティングできない. □

3.4 シミュレーション

補償パスによる再構成法と逐次的ルーティングによる再構成法の再構成率を比較するために計算機シミュレーションを行ってみた. 補償パスによる再構成法では, 予備 PE 配置の全ての組合せに対し, 再構成を行った. PE が正常な確率におうじて故障パターンをランダムに 1000 個発生させ, 同じパターンに対し 2 つの手法を適用した結果を図 8 に示す. $N = 4$ であり, 横軸は PE 歩留まり, 縦軸は成功した回数である.

最大で 10% 程度, 提案手法が補償パスによる再構成法よりも再構成率が高いことがわかる.

図 8: シミュレーション結果 ($N = 4$)

4 むすび

本報告では、2行2列の予備 PE をもつ $1 - \frac{1}{2}$ トラックスイッチモデルについていくつかの点について検討を行った。まず、補償パスにもとづく再構成法では再構成不可能な故障パターンが存在することを示した。そして、再構成可能な場合には必ず再構成できる最適な再構成法として、逐次のルーティングにもとづく再構成法を提案した。計算機シミュレーションでは、 $N = 4$ の場合、最大で約 10% 程度、補償パスによる手法よりも再構成率が高いことが確認できた。

今回提案した手法は、PE のマッピングの決定を網羅的に行っており、そのため本手法の時間計算量は $O(9^{N \times N})$ であると考えられる。より効率の良い手法の考案については今後の課題としたい。

参考文献

- [1] S.Y.Kung, S.N.Jean, and C.W.Chang, "Fault-tolerant array processors using single-track switches," IEEE Trans. Comput., vol.38, no.4, pp.501-514.
- [2] V.P. Roychowdhury, J. Bruck, and T. Kailath, "Efficient algorithms for reconfiguration in VLSI/WSI arrays," IEEE Trans. Comput., vol.39, no.4, pp.480-489, April 1990.
- [3] 沼田一成, 堀口進, "グラフを用いた $1-1/2$ 型格子結合ネットワークの最適構成法," 信学技報, FIIS95,