ビアスイッチFPGAの部分的再構成における書き換えスイッチ数の最小化

土井 龍太郎[1,2,a)]　劉 載勲[1,b)]　橋本 昌宜[1,c)]

概要：FPGA のエネルギー効率向上を目指して，不揮発ビアスイッチを活用した FPGA の研究開発が進んでいる．ビアスイッチ FPGA ではプログラム変更時に意図しないビアスイッチが書き換えられるスニークパス問題が発生する可能性があったが，問題が回避できるスイッチ書き換え順が常に存在すること，またその書き換え順がスイッチの書き換え状況を表す木構造を用いて求められることが示されている．本稿では，これまで議論されていなかった部分的再構成について，木構造の根ノードを最適に選択することでスニークパスを回避しつつ書き換え回数を最小化する手法を提案する．シミュレーション評価により書き換え回数が最大 77%削減されることを確認した．提案手法はビアスイッチ FPGA の再構成時間の削減および書き換え回数に制約のあるビアスイッチの長寿命化に貢献する．

1. 序論

FPGA (field programmable gate array) が抱えるエネルギー効率が低いという課題 [1,2] の解消を目指し，性能ボトルネックである SRAM (static random access memory) 型スイッチに代えて RRAM (resistive RAM) の一種であるビアスイッチを利用する FPGA の研究・開発が行われている [3,4]．ビアスイッチ FPGA では縦と横に走る配線の交点にビアスイッチを配置したクロスバー回路により配線接続の切り換えを実現する．しかし，スイッチのプログラミングに共用の信号配線を使用するため，回路のプログラミング状態によっては，プログラミング信号が回り込んで意図しないスイッチに与えられるスニークパス問題という現象が生じる．この現象は FPGA の利点である再構成を阻害する重大な問題であるため，対策が不可欠である．

スニークパス問題はクロスバー内のビアスイッチの書き換え順を調整することで常に回避できることが示されており，クロスバー配線の接続状況を表す木構造を用いてそのような書き換え順を求める対策手法が提案されている [5]．文献 [5] はクロスバー内の全てのスイッチがオフである状態を初期状態として想定し，そのクロスバーに対してループのない任意のコンフィギュレーションを書き込むことを実現している．しかし，クロスバーの初期状態を任意のコンフィギュレーションにした場合のスニークパス対策手法は議論されていない．

本稿は，未確立であった新旧のコンフィギュレーションがともに任意である場合において，スニークパス問題を回避する部分的再構成手法を提案する．提案手法では新旧コンフィギュレーションの共通部分の一部をスニークパス回避のために書き換えるが，その数を最低限に抑えることで再構成時に書き換えるスイッチ数の最小化を達成する．提案手法により，ビアスイッチの長寿命化やビアスイッチ FPGA 再構成時間の短縮に貢献できる．

本稿の構成は次の通りである．2 節ではビアスイッチやそれを利用した FPGA の構造や動作を説明する．3 節において，ビアスイッチ FPGA におけるスニークパス問題のメカニズムおよび既存の対策手法について述べる．4 節で提案する部分的再構成手法について詳述する．5 節では提案手法を適用した際のシミュレーション結果を示し，本手法の利点を議論する．最後に 6 節で結論を述べる．

2. ビアスイッチ FPGA

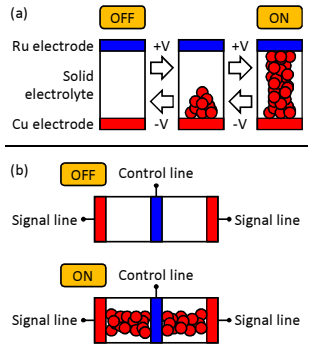
2.1 ビアスイッチ

ビアスイッチは不揮発性を有する書き換え可能な小体積ナノスイッチであり，原子スイッチとバリスタにより構成される．ここではデバイスの構造，動作，特徴を概説する．

原子スイッチは図 1(a) のように銅電極とルテニウム電極の間に固体電解質を挟んだ構造を有し，電極間に電圧を印加することで銅イオンによる架橋の形成・消失が可逆的に繰り返せる素子である．銅電極に正電圧を印加すると電極間に架橋が形成されてオン状態 (低抵抗状態) になり，負電圧を印加すると電極間の架橋が消失してオフ状態 (高抵抗状態) になる．原子スイッチは，電源供給がなくてもオン・オフ状態が維持される不揮発性を有する [6]．信頼性向上のため，図 1(b) のように 2 個の原子スイッチを逆方向に直列接続した CAS (complementary atom switch) が単位スイッチ素子として用いられる．CAS のプログラミング時は信号線と制御線を使い，2 個の原子スイッチを 1 個ずつ書き換える．一方，通常動作時は信号線のみを使用する [7]．

ビアスイッチは図 2 のように CAS の制御端子部分にバ

[1] 大阪大学 大学院情報科学研究科 情報システム工学専攻
[2] 日本学術振興会 特別研究員 DC
[a)] doi.ryutaro@ist.osaka-u.ac.jp
[b)] yu.jaehoon@ist.osaka-u.ac.jp
[c)] hasimoto@ist.osaka-u.ac.jp



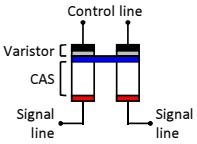


図 2　ビアスイッチの構造

図 1　(a) 原子スイッチおよび (b)CAS の構造と動作

リスタを接続した構造を持つ．信号端子と制御端子の間に閾値以上の電圧 (プログラミング電圧) が印加されるとバリスタが導通し，原子スイッチにプログラミング電流が流れる．一方，印加される電圧が閾値以下 (通常動作時) ならば，非導通のバリスタにより制御線が信号線と切り離され，ビアスイッチは信号端子のみからなる 2 端子素子とみなせる．このような印加電圧に応じて 2 つの機能を提供するバリスタを利用することで，FPGA 上に多数搭載されるビアスイッチの内，狙ったビアスイッチのみにプログラミング信号を与えることができる [4]．なお，ビアスイッチの書き換え可能回数は約 1,000 回である [7]．

2.2　ビアスイッチ FPGA

ビアスイッチ FPGA は CLB (configurable logic block) と呼ばれる単位回路をアレイ状に敷き詰めた構造を有し，各 CLB は縦横のグローバル配線の交点にビアスイッチを配置したクロスバー回路および論理ブロックからなる [3]．クロスバー内のビアスイッチは縦横の信号配線間の接続・非接続を切り換える機能を担い，縦横のグローバル配線のルーティングや論理ブロックの入出力を行う．論理ブロックでは組み合わせ回路や順序回路を構築する．

クロスバーは図 3(a) に示す構造を有する．信号線と制御線はどちらも縦方向と横方向に存在する．図 3(a) では 2x2 のクロスバーにおけるビアスイッチのプログラミングの流れをステップごとに示しており，各ステップで 1 個の原子スイッチがオン状態に書き換えられる．スイッチの書き換えには交差する信号線と制御線の組を使用し，書き換えドライバにより信号線に高電位，制御線に低電位を与える．なお，それ以外の配線はフローティングにする．ステップ 1 と 2 により左下のビアスイッチを構成する 2 個の原子スイッチが正しくオン状態にできることがわかる．

3.　スニークパス問題および既存対策手法

3.1　クロスバーにおけるスニークパス問題

前節で説明したクロスバープログラミングの構造上，クロスバー内ビアスイッチのオン・オフパターンによってスニークパス問題が発生する可能性がある．例えば，3(b) 中の右下のビアスイッチは正常に書き換えられない．既にオン状態になっている左下および左上のビアスイッチを経由してプログラミング信号が右上のビアスイッチに回り込み，意図しない原子スイッチに電圧が印加されるためである．

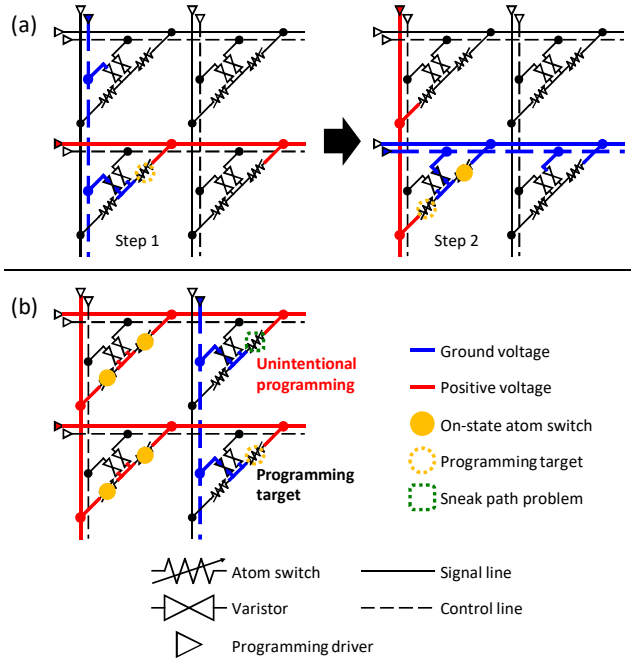


図 3　(a) クロスバーの構造とプログラミング，(b) クロスバーにおけるスニークパス問題

る．このようなプログラミング信号の回り込みにより狙ったスイッチ以外の状態を書き換えてしまう現象をスニークパス問題と呼ぶ．スニークパス問題は FPGA の正常な再構成を阻害するため，その対策の検討が不可欠である．

3.2　スニークパス問題の発生条件

文献 [5] は横方向または縦方向の同一信号線上に存在する複数のオン状態ビアスイッチがスニークパスの原因であると示しており，それらをコネクタスイッチ (CS) と呼称している．また，同一線上に存在するオン状態スイッチ数が 1 個の場合は，そのオン状態スイッチを非コネクタスイッチ (NCS) と呼ぶ．横方向または縦方向の CS により縦方向または横方向の複数の信号線が区別不能になり，その後にそれらの配線上に存在するビアスイッチを書き換えようとするとスニークパス問題が発生する．例えば，図 4 左のコンフィギュレーションでは，横方向の CS であるスイッチ E と F が縦方向信号線 SV1 と SV2 を区別不能にする．この状態で SV1 上のスイッチを書き換えるために SV1 にプログラミング信号を与えると，区別不能になっている SV2 にも同じ信号が伝搬してしまう．その結果，SV2 上の同じ位置に存在するスイッチも同時にプログラムされスニークパス問題が発生する．逆に考えると，複数の縦または横の信号線が接続されて区別不能となる前にその信号線上の原子スイッチを書き換えるとスニークパス問題を回避できる．これに着目した既存対策手法について次項で説明する．

なお，ビアスイッチを構成する 2 個の原子スイッチのうち，下側の原子スイッチを書き換える際は縦方向の信号線を使用する (図 3(a) のステップ 2 など) ため，横方向の CS に注意する必要がある．なぜなら，それらの CS により複数の縦方向信号線が接続され区別不能となるためである．一方，上側の原子スイッチのプログラミング時においては縦方向の CS に注意を払う必要がある．

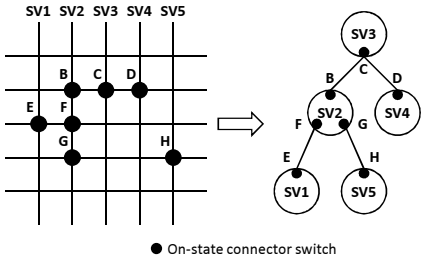


図 4　クロスバーにおける信号線の接続状況を表す接続木

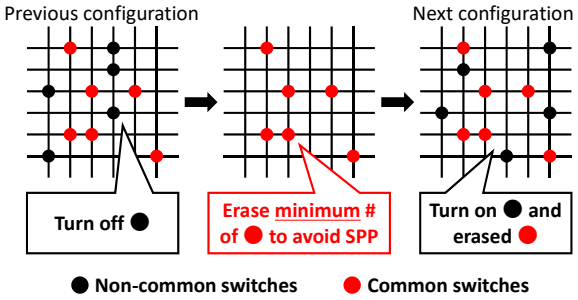


図 5　提案手法のコンセプト

3.3　スニークパス問題への既存対策手法

文献 [5] ではループを含まない任意のコンフィギュレーションを対象として，クロスバーにおけるスニークパスが発生しないビアスイッチのプログラミング順を得る手法が提案されている．この手法は 3 つのステップで構成される．ステップ 1 で対象コンフィギュレーションにおいてオン状態にするビアスイッチの上側原子スイッチをすべて先に書き込む．その後，ステップ 2 で横方向 NCS，ステップ 3 で横方向 CS の下側原子スイッチを書き込む．ステップ 1 と 2 では区別不能になっている信号線が存在しないため，任意のプログラミング順でスニークパス問題なく書き込みが行える．一方，ステップ 3 では CS の書き込みによって区別不能な信号線が生じるため，スニークパス問題の回避のためにプログラミング順の調整が必要となる．

ステップ 3 のプログラミング順の決定において，クロスバーの縦方向信号線の接続状況を表す接続木を利用する．ここで，縦方向信号線に注目しているのは，前項で述べたように下側原子スイッチの書き換え時は横方向 CS によって区別不能になる縦方向信号線が問題となるためである．図 4 は横方向 CS を対象に構築した接続木の一例である．各ノードが縦方向信号線 1 本と対応する．ルートノードは任意に選択でき，図 4 では縦方向信号線 SV3 がルートに選択されている．対象コンフィギュレーションにおいて 2 本の縦方向信号線が接続される場合，接続木においてはそれらの信号線に対応する 2 個のノード間にエッジが引かれる．各エッジの両端に表示されている黒点は CS を表しており，両端の CS がどちらもオン状態になったときにそのエッジが有効になり 2 本の信号線が接続されて区別不能になることを表現している．

接続木の重要な特徴は，対象コンフィギュレーションのプログラミングにおいて最後に書き込むことができるスイッチが葉ノードとして現れることである．この特徴を利用し，プログラミングの最後に書き込めるスイッチを再帰的に探索することでスニークパス問題を回避するプログラミング順が求まる．1 回の再帰的ステップで葉ノード 1 つが選択され，葉ノード内の親ノードに接続するスイッチが最後に書き込むスイッチとして選ばれ，葉ノードとそれに接続するエッジを除去することで接続木が更新される．最終的に全てのエッジが除去されて全てのノード，すなわち縦方向信号線が区別可能になった時点で再帰的探索は終了する．

4.　提案する部分的再構成手法

4.1　提案手法の概要

本稿はクロスバーのコンフィギュレーションを効率的に変更する部分的再構成手法を提案する．文献 [5] の手法はクロスバー内の全てのビアスイッチがオフ状態であることを初期状態として想定している．そのため，この手法を用いる場合，旧コンフィギュレーションの全てのオン状態スイッチを消去した後に新コンフィギュレーションを書き込むことは可能である．しかし，この方法では新旧コンフィギュレーションで共通のオン状態スイッチがあった場合に不必要な書き換えが発生する可能性がある．不必要な書き換えは再構成時間の増加に直結し，最大書き換え可能回数に制限のあるビアスイッチの寿命も短縮してしまう．これを解決するため，スニークパス問題を回避しつつ書き換えスイッチ数を最小化する部分的再構成手法を提案する．

提案手法は図 5 に示すように，新旧コンフィギュレーションの非共通スイッチの書き換えに加えて，最小限の共通スイッチをスニークパスを回避するために書き換える．表 1 は各ステップにおいて書き換えるスイッチの種類をまとめたものである．なお，各スイッチの種類を表す略語を括弧内に示している．4.2 項および 4.3 項で部分的消去および部分的書き込みの方法をそれぞれ説明する．4.4 項と 4.5 項で書き換えスイッチ数の最小化手法について述べる．

4.2　部分的消去

表 1 のステップ 1 の部分的消去では旧コンフィギュレーション中の非共通ビアスイッチの上側・下側原子スイッチの両方を消去する．書き込み時と同様に区別不能な信号線上のスイッチを消去しようとするとスニークパス問題が発生する．図 6 がその状況を表しており，スニークパスによって区別不能な信号線上の消去対象と同じ位置に存在するスイッチに消去電圧が印加される．一方で，ループなしコンフィギュレーションではスニークパスにより消去電圧が印加されるスイッチは常にオフ状態であることが保証できる．なぜなら，そのスイッチがオン状態であると仮定するとループが形成されるためである．提案手法はループなしのコンフィギュレーションを対象にしているため，プログラミング順に関係なく部分的消去が可能である．

4.3　部分的書き込み

部分的書き込みは 2 つのステップで実現される．新コン

表 1　提案手法の各ステップで書き換えるスイッチ

書き換えステップ		旧コンフィグの非共通スイッチ		新旧コンフィグの共通スイッチ				新コンフィグの非共通スイッチ		
		上側 AS (S_{PU})	下側 AS (S_{PL})	上側 AS (S_{CU})	下側 AS			上側 AS (S_{NU})	下側 AS	
					横方向 CS (S_{CH})	縦方向 CS (S_{CV})	その他 (S_{CO})		横方向 CS (S_{NC})	横方向 NCS (S_{NN})
開始		Proged.	Proged.	Proged.	Proged.	Proged.	Proged.			
部分的消去	ステップ 1	Turn off	Turn off	Proged.	Proged.	Proged.	Proged.			
部分的書き込み	ステップ 2a			Proged.	Proged.	Turn off	Proged.			
	ステップ 2b			Proged.	Proged.		Proged.	Turn on		
	ステップ 3a			Proged.	Proged.		Proged.	Proged.		
	ステップ 3b			Proged.	Turn off		Proged.	Proged.		
	ステップ 3c			Proged.		Turn on	Proged.	Proged.		Turn on
	ステップ 3d			Proged.	Turn on	Proged.	Proged.	Proged.	Turn on	Proged.
終了				Proged.	Proged.	Proged.	Proged.	Proged.	Proged.	Proged.

AS: 原子スイッチ, CS: コネクタスイッチ, NCS: 非コネクタスイッチ, Proged.: 書き込み済み

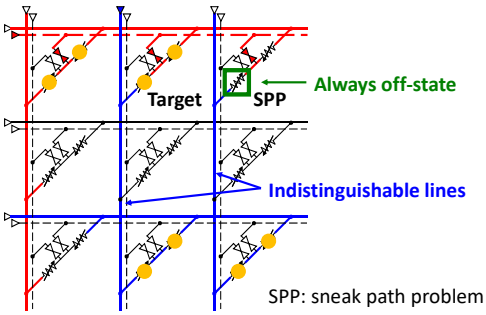


図 6　部分的消去時におけるスニークパス問題

フィギュレーションで新たにオン状態にするビアスイッチの上側原子スイッチ S_{NU} および下側原子スイッチ S_{NC} と S_{NN} をそれぞれステップ 2 および 3 で書き込む．各ステップにおいてスニークパスを回避するため，共通スイッチ S_{CV} と S_{CH} の一部を事前に消去してから書き込みを行う．

ステップ 2 では上側原子スイッチ S_{NU} を書き込むが，3.2 項で説明したように縦方向の CS である S_{CV} に注意する必要がある．S_{NU} と同一の横方向信号線に S_{CV} が存在する場合は，その S_{CV} の下側原子スイッチをステップ 2a で消去する．前項で述べた通り部分的消去はスニークパス問題なく任意のタイミングで行える．このステップで消去された S_{CV} は後のステップ 3 で再度書き込む．ステップ 2a によってプログラミング信号の回り込みを防げるため，続くステップ 2b ではスニークパス問題なく S_{NU} が書き込める．ステップ 2 により，ステップ 3 の開始段階でオン状態にすべきビアスイッチの上側原子スイッチは全てオン状態になっていることが保証できる．

ステップ 3 では新たにオン状態にする横方向の CS と NCS である S_{NC} と S_{NN}，およびステップ 2a で消去した S_{CV} を書き込む．ステップ 3 は 4 つのステップから構成される．ステップ 3a では 3.3 項で述べた接続木のルートノードを任意に選択できる特徴を利用し，ステップ 3b〜3d における書き換えスイッチ数を最小化するようにルートノードを調整する．書き換えスイッチ数の最小化手法については次項以降で説明する．なお，ステップ 3a はルートノードの変更のみであるためスニークパス問題は生じない．次のステップ 3b では S_{CH} の一部，具体的には書き込み対象が存在するノードおよびその子孫ノード内の横方向 CS のうち，各ノードの親に接続する CS を消去する．これにより，書き込み対象以下のノードが接続木から切断され，それらのノードが区別可能になる．ステップ 3b は消去ステップであるためスニークパス問題なく行える．その後，ステップ 3c において書き込み対象のうち横方向 CS 以外のスイッチ，すなわち S_{NN} と S_{CV} を書き込む．ステップ 3b によって書き込み対象ノードは他のノードと切り離されているため，プログラミング信号が他のノードに伝搬することはなく，任意のプログラミング順に対してスニークパス問題は発生しない．最後のステップ 3d で書き込み対象ノードとその子孫ノードを接続木に再接続する．横方向 CS である S_{CH} と S_{NC} を接続木の浅い位置のものから順に書き込んでいくと，常に葉ノードへの書き込みとなるため，3.3 項で説明した通りスニークパスが回避できる．

図 7 にステップ 3 の流れを例示する．ステップ 3a では書き換えスイッチ数を最小化するルートノードとしてノード A が選択される．続くステップ 3b では書き込み対象ノード B およびその子孫ノード C と E 内の親ノードへの CS が消去され，3 つのノードが接続木から切断される．次に，ステップ 3c において区別可能となったノード B 内の赤色で示された対象スイッチを書き込む．そして，ステップ 3d でノード B, C, E という接続木の浅い位置順で各ノード内の CS を書き込むことにより書き込み対象ノード以下の接続を回復する．

なお，クロスバーの対称性より，ステップ 2 と 3 を入れ替えて下側原子スイッチを先に書き込んだ後に上側原子スイッチを書き込むことも可能である．この場合，上側原子スイッチのプログラミング順を同様に求めることになる．新旧コンフィギュレーションのパターンに依存し，上下のどちら側の原子スイッチを先に書き込むかによって書き換えスイッチ数が変化する．そのため，両方を評価してより少ない書き換えスイッチ数のものを選択する．

4.4　書き換えスイッチ数の最小化

本項では部分的再構成時における書き換えスイッチ数の最小化手法について議論する．提案手法のステップ 1 と 2 における書き換えスイッチ数は，コンフィギュレーション

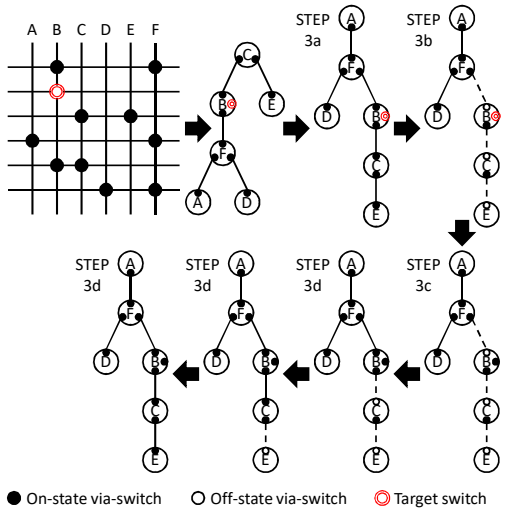


図 7　提案する部分的書き込み手法

が与えられると固定値となる．一方，ステップ 3a においてはルートノードの選択に自由度が存在し，ルートの選択によって木構造が変化する．それに伴って書き込み対象ノードの子孫ノード数も変動し，ステップ 3b で消去する S_{CH} の数が変化する．したがって，ステップ 3b で消去する S_{CH} の数を最小化するようにルートノードを選択することで，部分的再構成全体の書き換えスイッチ数を最小化できる．

新たにオン状態にする対象スイッチが 1 つの場合，書き込み対象に先立って事前に消去する S_{CH} の数は式 (1) で与えられる．ステップ 3b では書き込み対象ノードとその子孫ノード内で親に接続する S_{CH} を消去するが，ステップ 2a によって一部の S_{CH} が既にオフ状態である場合があるため式 (1) となる．

$$\begin{aligned}&(\text{書き込み対象が 1 つの場合に事前消去する } S_{CH} \text{ の数})\\ &= (\text{対象ノードと子孫ノード内で親に接続する } S_{CH} \text{ の数})\\ &\quad - (\text{右辺第 1 項のうち既にオフ状態の } S_{CH} \text{ の数}) \quad (1)\end{aligned}$$

一方，複数の書き込み対象が存在する場合，事前消去する S_{CH} の総数は必ずしも各書き込み対象に対する式 (1) の総和にはならず，以下で説明する書き込み対象スイッチ同士の支配関係を利用することでさらに削減できる．ある書き込み対象スイッチへのステップ 3 の適用時にその対象ノードや子孫ノードに他の書き込み対象が存在する場合，後者のスイッチは前者と同時に書き込める．なぜなら，ステップ 3b によって対象ノードおよび子孫ノードは接続木から切り離され，区別可能になっているためである．このような状況に対して前者のスイッチが後者を支配していると定義する．例えば，図 8 左の接続木では，ノード E がルートの場合にスイッチ b がスイッチ a と f を支配する．ステップ 3b により書き込み対象ノード B と子孫ノード A, C, D, F が接続木から切断されるため，ステップ 3c においてスイッチ b だけでなくスイッチ a と f も書き込むことができる．以降の段落でスイッチの支配関係を活用し，書き換えスイッチの総数を最小化する最適なルートノードおよび一度に書き込むスイッチ群の導出手法を提案する．

本稿ではこの最適化問題をコスト最小化被覆問題にモデル化する．まず，各書き込み対象スイッチについて，ルートノードを変更しながら，式 (1) による事前消去する S_{CH} の数 (コスト) の計算，および書き込み対象ノードと子孫ノード内に存在する支配可能なスイッチの列挙を行う．なお，ルートノードの変更については全通りを評価する必要はなく，注目している書き込み対象ノードに繋がる各エッジが親へのエッジになった場合と対象ノード自身がルートになった場合のみを評価すればよい．なぜなら，親へのエッジが決まった時点で先祖ノードのどれを選択しても事前消去する S_{CH} の数や支配可能なスイッチが変化しないためである．上記の手続きにより，支配関係によって同時に書き込むことができるスイッチ群とそのスイッチ群を書き込むために必要な事前消去 S_{CH} 数の組が複数得られる．それらの組み合わせから，全ての書き込み対象を被覆し，かつ合計の事前消去 S_{CH} 数が最小になる集合を求める．

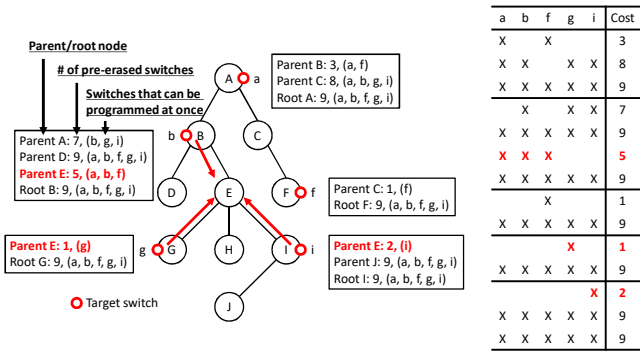


図 8　書き換えスイッチ数の最小化手法

図 8 は前述の被覆問題を例示したものである．書き込み対象スイッチ a に注目すると，2 本のエッジがあるためノード B が親の場合とノード C が親の場合，それに加えてノード A 自身がルートの場合の合計 3 つの場合について評価する．例えばノード B が親の場合はステップ 3b でノード A, C, F を切り離すため，事前消去 S_{CH} 数は 3 であり，支配関係によって一度に書き込めるスイッチは a と f である．同様の評価を全ての書き込み対象について行うと，図 8 右に示す表が作成できる．表中の各行が支配関係によって同時に書き込むことができるスイッチ群とそのスイッチ群を書き込むために必要な事前消去 S_{CH} 数の組に対応する．この表より，全ての書き込み対象を被覆しつつコストを 8 に最小化する赤字で示した 3 つの組の集合が求まる．この例ではノード E または H をルートに選択し，書き込み対象をスイッチ a, b, f，スイッチ g，スイッチ i の 3 つのスイッチ群に分け，各スイッチ群ごとにステップ 3 を適用する場合が最適解となる．

4.5　ルートノード選択における計算複雑度の削減

被覆問題を解くことで書き換えスイッチ数を最小化する最適なルートノードが選択できることを前項で述べた．一方で，被覆問題が NP 困難であることは周知の事実である．本項では被覆問題を解くことなく最適なルートノードを効率的に選択する手法を提案する．

被覆問題の解としては，一度に書き込める支配・被支配スイッチ群および支配スイッチから見た親ノードの方向の組の集合が得られる．このとき，この集合の各組における親ノードの方向を同時に満足するルートノードが必ず 1 つ

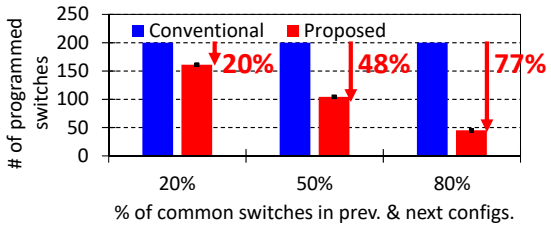


図 9 提案手法と従来手法における書き換えスイッチ数の比較

以上存在するという接続木の特徴を確認した．例えば，図 8 の例では左図に赤字で示した 3 つの組の集合が最適解となる．各組によってスイッチ b, g, i が支配スイッチであり，赤矢印で示した通りいずれの組においてもノード E が親ノードであることがわかる．この場合，全ての赤矢印を同時に満たすルートノードとしてノード E または H が存在する．このように，被覆問題の最適解として得られる集合から決定されるルートノードは必ず 1 つに集約できるという特徴を接続木は有する．紙面の都合上，この特徴が常に成立することについての詳細な証明は省略するが，以下の式 (2) に示す流れで証明可能である．

$$
\begin{aligned}
&\text{支配スイッチは他のスイッチから支配されない} \\
&\Rightarrow \text{複数の支配スイッチは互いに支配関係にない} \\
&\Leftrightarrow \text{複数の支配スイッチは互いに子孫関係にない} \\
&\Rightarrow \text{複数の支配スイッチはルートノードを共有できる}
\end{aligned}
\tag{2}
$$

上記の接続木の特徴を利用することで，各ノードがルートである場合の書き換えスイッチ数を順番に評価し，その値が最小となるものを選択するのみで最適なルートノードを得ることができる．この手法により，被覆問題を解く場合と比較して計算複雑度を削減することができる．アルゴリズムとしては各ノードをルートにして深さ優先探索を行うことで実装可能であり，接続木のノード数は最大でもクロスバーの縦方向信号線の数 N_{ver} であるため，計算複雑度は最悪でも $O({N_{\mathrm{ver}}}^2)$ である．

5. シミュレーション評価結果

本項では提案手法による書き換えスイッチ数の削減効果について議論する．提案手法と従来手法における書き換えスイッチ数の比較結果を図 9 に示す．ここで，従来手法とは文献 [5] の手法を用いて旧コンフィギュレーションのオン状態ビアスイッチを全て消去した後，新コンフィギュレーションをクロスバーに書き込むものを指す．この評価ではループなしの新旧コンフィギュレーションをランダムに生成し，スニークパスを回避するプログラミング順の導出を行った．クロスバーのサイズは 100x100 であり，そのうち 0.5%のビアスイッチがオン状態であるコンフィギュレーションを対象とした．また，新旧コンフィギュレーションの共通スイッチの割合を 20%, 50%, 80%と変化させ，それぞれについて 10,000 サンプルのシミュレーションを実施した．

図 9 より，従来手法における書き換えスイッチ数は 200 で固定されていることがわかる．これは 100 × 100 ビアスイッチ × 0.5% × 2 原子スイッチ × 2 コンフィギュレーション (新旧) = 200 に一致している．一方，提案手法により書き換えスイッチ数が 20%〜77%削減されていることが確認できる．書き換えスイッチ数の 77%の削減によってビアスイッチ FPGA 全体の再構成可能回数は 4.3 倍になり，再構成時間は 77%削減される．

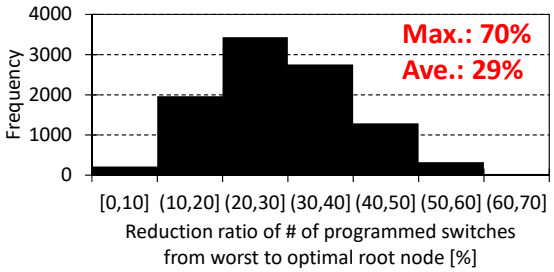


図 10 最悪ケースと比較した最適なルートノード選択による書き換えスイッチ数の削減率のヒストグラム

次に，提案手法における最適なルートノードの選択が与える影響について議論する．ここでは，最適なルートノードを選択した場合と最悪のルートノードを選択した場合の書き換えスイッチ数を比較する．最悪ルートノードの場合と比較した最適ルートノードの場合における書き換えスイッチ数の削減率のヒストグラムを図 10 に示す．この評価では，100x100 のクロスバーに対して 10,000 サンプルのシミュレーションを実行した．旧コンフィギュレーションにおいて 1%のスイッチがオン状態であり，新コンフィギュレーションではそこからさらに 0.1%のスイッチが新たにオン状態になる場合を評価した．最適なルートノード選択による書き換えスイッチ数の削減率は最大で 70%，平均で 29%と高い．このことから，提案するルートノード選択手法が重要な役割を果たしていることが確認できる．

6. 結論

本稿ではビアスイッチ FPGA のクロスバーにおける部分的再構成手法を提案した．提案手法は接続木のルートノードを適切に選択することで，スニークパス問題を回避しつつ再構成時の書き換えスイッチ数を最小化する．シミュレーション評価により，提案手法を用いると書き換えスイッチ数を最大 77%削減できることを確認した．提案手法はビアスイッチの長寿命化やビアスイッチ FPGA の再構成の高速化に貢献する．

謝辞 本研究は，JST，CREST，JPMJCR1432 の支援，および JSPS 科研費，JP17J10008 の助成を受けたものである．

参考文献

[1] I. Kuon, et al., IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., 2007.
[2] M. Lin, et al., IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., 2007.
[3] H. Ochi, et al., IEEE Trans. VLSI Syst., 2018.
[4] N. Banno, et al., IEEE Trans. Electron Devices, in press.
[5] R. Doi, et al., in ICCAD, 2018.
[6] K. Okamoto, et al., in IEDM, 2011.
[7] M. Miyamura, et al., in ISQED, 2014.