ビアスイッチFPGA再構成時のスニークパス問題を回避するプログラミング順決定手法

土井 龍太郎[1,2,a)]　劉 載勲[1,b)]　橋本 昌宜[1,c)]

概要：従来 FPGA の性能ボトルネックである SRAM 型スイッチを，不揮発性メモリの一種であるビアスイッチで置換した FPGA に関する研究・開発が行われている．ビアスイッチ FPGA は縦と横に走る信号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替えを実現する．一方，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が生じる．本稿では，FPGA の正常な再構成を阻害するスニークパス問題の発生条件を明らかにし，スニークパス問題を回避するビアスイッチのプログラミング順序の決定手法を提案する．ループのないコンフィギュレーションにおいては常に提案手法が有効であることを示し，シミュレーションによる検証も行う．提案手法により，すべての実用上のコンフィギュレーションにおいてスニークパス問題なく正常な再構成を実現できる．

キーワード：不揮発性ビアスイッチ FPGA，スニークパス回避，クロスバープログラミング，スイッチプログラミング順序

1. 序論

集積回路の微細化に伴う ASIC (application specific integrated circuit) 開発費の高騰を背景とし，FPGA (field programmable gate array) に代表される再構成可能 LSI (large scale integration) への期待が高まっている．しかし，FPGA には面積効率が悪く，信号遅延や消費電力が大きいという課題が存在する [1]．これらの課題は，再構成を実現するために，配線接続を切り換えるスイッチ回路が FPGA 上に大量搭載されていることに起因する．最も一般的な SRAM (static random access memory) 型 FPGA では，スイッチ機能を担うトランスミッションゲートとスイッチのオン・オフ状態を保持する SRAM セルによりスイッチ回路が構成される．しかし，MOS (metal-oxide-semiconductor) トランジスタにより実現されるトランスミッションゲートは抵抗・容量が大きく，6 個のトランジスタを使用する SRAM セルは面積が大きい．そのため，従来のスイッチ回路の使用は性能や面積効率の低下を招く [2]．

従来の FPGA が抱える課題の解消を目指し，性能ボトルネックである SRAM 型スイッチの代わりに 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 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，発生条件や対策の検討が不可欠である．

本稿ではスニークパス問題を引き起こすクロスバー回路のプログラミング状態を明らかにし，スニークパス問題を回避するビアスイッチのプログラミング順序の決定手法を提案する．ループが存在しない任意のクロスバー回路のオン・オフパターンにおいてはスニークパス問題を回避するプログラミング順が必ず存在することを証明し，信号配線の接続状況を木構造で表現することでそのプログラミング順を求めるアルゴリズムを考案する．また，シミュレーション評価により提案手法の有効性を確認する．提案手法により，すべての実用上のコンフィギュレーションにおいてスニークパス問題なく正常な再構成を実現できる．

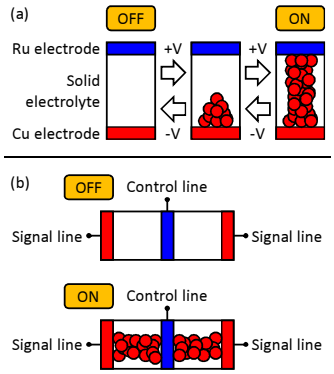
本稿の構成は次の通りである．2 節ではビアスイッチ FPGA の構造やスニークパス問題について説明する．3 節においてスニークパス問題の発生条件を検討し，スニークパス問題を引き起こす回路状況を明らかにする．4 節でスニークパス問題を回避するビアスイッチのプログラミング順の決定手法を提案し，その有効性を 5 節で確認する．最後に 6 節で結論を述べる．

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，特徴について概

[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



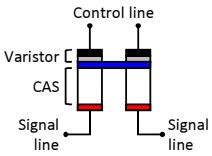


図 2　ビアスイッチの構造

図 1　(a) 原子スイッチおよび (b)CAS の構造と動作

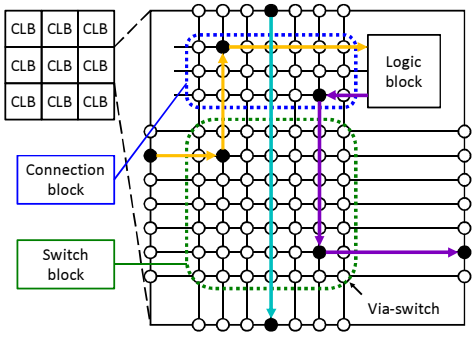


図 3　ビアスイッチ FPGA の構造

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．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 [5]．

ビアスイッチは図 2 のように CAS の制御端子部分にバ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 [3]．

ビアスイッチのサイズは 18 F^2, オン抵抗は 200 Ω, 容量は 0.14 fF と非常に小さい [3,6]．これらの特性により，SRAM 型スイッチをビアスイッチに置換することで FPGA の面積効率や性能が飛躍的に向上する．文献 [6] では，SRAM を用いた FPGA と比較してビアスイッチを用いたクロスバー回路の面積が 26 倍小さく，配線の遅延およびエネルギーが共に 90%以上削減されることが報告されている．

2.2　ビアスイッチ FPGA におけるスニークパス問題

ビアスイッチ FPGA は図 3 のように CLB (Configurable Logic Block) を敷き詰めた構造を有し，各 CLB は縦横の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 [6]．クロスバー回路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う．上半分が従来 FPGA のコネクションブロックに相当し，論理ブロックの入出力を行う．下半分がスイッチブロックに相当し，縦横のグローバル配線のルーティングを行う．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．

次に，クロスバー回路におけるビアスイッチのプログラミング方法およびスニークパス問題について説明する．クロスバー回路の構造を図 4 に示す．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 4 では 2x2 のクロスバー回路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．続くステップ (3) と (4) により左上のビアスイッチがオン状態にプログラムされる．しかし，次のステップ (5) における右下のビアスイッチのプログラミングは正常に実行できない．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．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，その発生条件や対策の検討は重要である．

2.3　従来のスニークパス問題対策

ここではスニークパス問題への従来の対策手法とその欠点について述べる．文献 [6] はプログラミング制約を導入することでスニークパス問題の回避が可能であると主張している．このプログラミング制約は同一信号線上に存在するオン状態のビアスイッチ個数に関するもので，縦横いずれかの方向にのみ複数のビアスイッチがオン状態になることを許容する．すなわち，この制約下では図 4 のステップ (5) のように縦横の両方に複数のオン状態ビアスイッチが存在するコンフィギュレーションは禁止される．しかし，この対策手法には明らかな欠点が存在する．プログラミング制約の導入によりビアスイッチ FPGA の一部のコンフィギュレーションが禁止されてしまうため，利用できるコンフィギュレーションパターン数が減少してしまい，ルーティングの柔軟性の低下を招く．

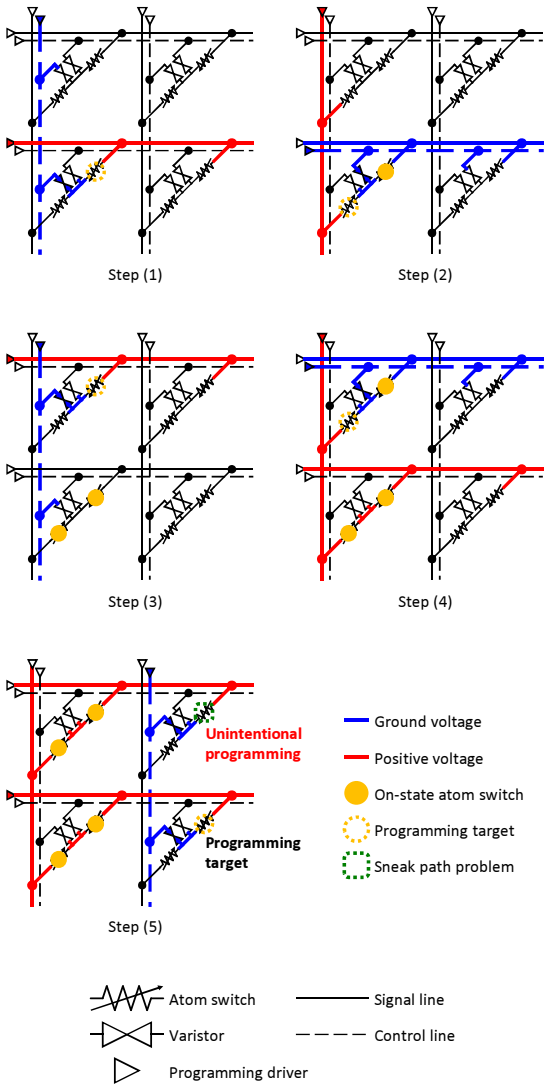


図 4　クロスバー回路の構造とビアスイッチのプログラミング

3.　スニークパス問題の発生条件

スニークパス問題へのより効率的な対策を検討するため，まず本節ではスニークパス問題を引き起こすクロスバー回路のプログラミング状態を明らかにする．本稿で提案するスニークパス問題の回避手法は 4 節で述べる．2.2 節で説明したように，クロスバー回路の交点に存在する原子スイッチは信号線に高電位，制御線に低電位が与えられたときにオン状態になる．そのような交点が複数存在する場合にスニークパス問題が発生する．本稿では信号線に与えられた信号線に与えられたプログラミング信号の折れ曲がり回数に着目し，スニークパス問題を起こす回路状況を 2 種類に分類する．それらの回路状況を図 5 の条件 (a) と (b) に示す．条件 (a) は信号線に与えたプログラミング信号が 2 回以上曲がるケースで，条件 (b) は折れ曲がり回数が 1 回か 0 回のケースである．以降では各条件の詳細について考察する．なお，本節では原子スイッチのオン状態への書き換え時についてのみ議論するが，オフ状態への書き換えは信号線・制御線に与える電位が逆になることを除き同一であるため同様の議論が可能である．

条件 (a) では横方向の同一信号線 SH2 に存在する複数の

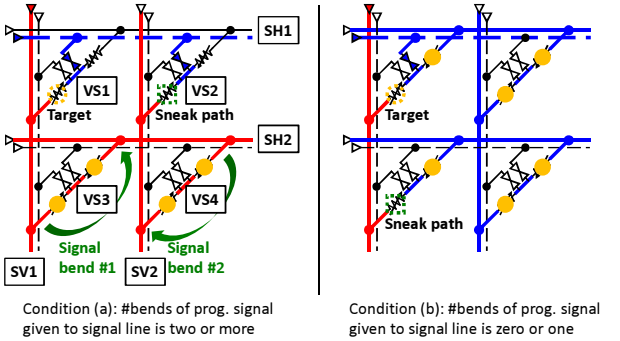


図 5　2 種類のスニークパス問題の発生条件

オン状態ビアスイッチ VS3 と VS4 により，2 本の縦方向信号線 SV1 と SV2 が接続されている．このとき，信号線 SV1 と SV2 のいずれかにプログラミング信号が与えられた場合，もう一方の信号線にも同じ信号が伝搬するため，プログラミング時において SV1 と SV2 は区別不能である．したがって，信号線 SV1 上のビアスイッチ VS1 の下側原子スイッチをオン状態にしようとすると信号線 SV2 上の同じ位置に存在する原子スイッチ，すなわちビアスイッチ VS2 の下側原子スイッチも同時にプログラムされる．まとめると，スニークパス問題は既に区別不能になっている縦方向または横方向の信号線上のスイッチをプログラムする際に発生する．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．例えば，条件 (a) ではビアスイッチ VS1 の書き換えをビアスイッチ VS3 と VS4 の書き換え前に実行することでスニークパス問題が回避できる．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 4 のステップ (2) など) ため，横方向の同一信号線上に存在する 2 個以上のオン状態ビアスイッチに注意する必要がある．なぜなら，それらのビアスイッチ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同一信号線上に存在する複数のオン状態ビアスイッチに注意を払う必要がある．

次に，信号線に印加したプログラミング信号の折れ曲がり回数が 1 回以下である条件 (b) について議論する．この場合では，制御線に与えたプログラミング信号が回り込み，スニークパス問題が発生する．しかし，このような回路状況になるのはループのあるコンフィギュレーションのプログラミング時のみである．例えば，図 5 の条件 (b) の例では 4 個のビアスイッチがすべてオンになるコンフィギュレーションでのプログラミング状況を示しており，ビアスイッチ VS1 と VS3 の上側のスイッチがオン状態になっている．条件 (b) はループを構成する原子スイッチのうち，最後の 2 個のスイッチを書き換える際にのみ成立し，制御線に与えたグラウンド電位がループ構造のため書き換え対象でないほうの原子スイッチに回り込む．

以上の議論より，ループを含むコンフィギュレーションにおいてはスニークパス問題は不可避であることがわかる．一方で，そのようなコンフィギュレーションは実用上使用されない．ループありの信号ルーティングは，ループなしのものに比べて配線抵抗・容量が増加し，遅延や電力

を悪化させるのみでメリットがないためである．したがって，条件 (b) については考慮する必要がない．以降の節では条件 (a) を考慮したスニークパス問題の対策手法を提案する．

4. 提案するスニークパス回避手法

本節では 3 節での議論に基づき，スニークパス問題の回避手法を提案する．提案手法により，クロスバー回路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が得られる．対象とするコンフィギュレーションはループを含まない任意のコンフィギュレーションである．

4.1 提案手法の概要

提案手法は 2 段階のステップで構成される．ステップ (1) では，対象とする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換える．例えば，10 個のビアスイッチをオン状態にするコンフィギュレーションでは，ステップ (1) で上側の原子スイッチ 10 個がすべてプログラムされる．ビアスイッチを構成する 2 個の原子スイッチが両方オン状態になってはじめて縦横の信号線がそのビアスイッチによって接続される．したがって，ステップ (1) においてはいずれの信号線も他の信号線に接続されることはなく，信号の回り込みは発生しないため，このステップではスニークパス問題は発生しない．よって，ステップ (1) におけるプログラミング順は任意である．

続くステップ (2) では，ステップ (1) で書き換えた原子スイッチに対応する下側の原子スイッチをすべてオン状態にプログラムする．このステップにおいては，上側の原子スイッチは既にオフ状態のため，下側の原子スイッチをプログラムする度に縦横の信号線が接続される．したがって，3 節で議論したスニークパス問題の発生条件を考慮してプログラミング順を決定する必要がある．3 節で述べたように，下側の原子スイッチを書き換える際は縦方向の信号線にプログラミング信号を与えるため，横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に注意する必要がある．縦方向の同一信号線上に存在する 2 個以上のオン状態ビアスイッチは問題にならない．プログラミング順の決定手法については 4.2 節で詳述する．

なお，クロスバーの構造およびプログラミング方法は対称性を有するため，上記のステップ (1) と (2) は逆にすることが可能である．その場合は上側原子スイッチのプログラミング順を以降で詳説する提案手法と同じように決定すればよい．

4.2 接続木を用いたプログラミング順の決定手法

ここでは，クロスバー回路におけるスニークパス問題を回避するプログラミング順をどのように決定するかについて説明する．以降では図 6 に示す 5x5 クロスバー回路のコンフィギュレーションを例として使用する．

4.1 節で述べたように，スニークパスの回避のためには横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に注意が必要となる．このようなビアスイッチは複数の縦方向信号線を接続し，区別不能とするためである．そこで，コネクタスイッチと非コネクタスイッチという 2 種類のカテゴリを定義し，各ビアスイッチを分類する．横方向の同一信号線上に 2 個以上のオン状態ビアスイッチが存在する場合，それらのスイッチはコネクタスイッチに分類される．一方，横方向信号線に存在するスイッチが 1 個のみの場合，そのスイッチは非コネクタスイッチに分類される．例えば，図 6 のビアスイッチ B〜H はコネクタスイッチであり，ビアスイッチ A と I は非コネクタスイッチである．コネクタスイッチの組により 2 本の縦方向信号線が接続される．例えば，ビアスイッチ E と F は縦方向信号線 SV1 と SV2 を接続する．それに対し，非コネクタスイッチは縦方向信号線同士を接続することはない．

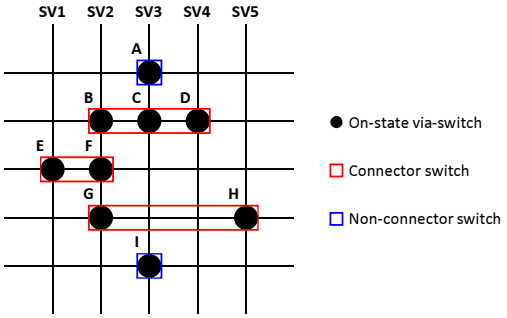


図 6 ループのないコンフィギュレーション例およびコネクタスイッチ・非コネクタスイッチの定義

3 節での議論より，既に接続されて区別不能となった縦方向信号線上に存在する下側原子スイッチをプログラムする際にスニークパス問題が生じる．そのため，スニークパス問題の回避のためにはすべての非コネクタスイッチをコネクタスイッチよりも先に書き換える必要がある．そうすることで複数の縦方向信号線が接続される前にすべての非コネクタスイッチのプログラミングが完了するため，非コネクタスイッチの書き換え時におけるスニークパス問題を回避できる．複数個の非コネクタスイッチが存在する場合のプログラミング順は任意である．

次に，残るコネクタスイッチのプログラミング順を決定する．コネクタスイッチのプログラミングにおいてはプログラミング順によりスニークパスの有無が変化するため，慎重に順番を決定する必要がある．例えば，図 6 のスイッチ E と F の後に B や G をプログラムするとスニークパス問題が発生する．スイッチ E と F により縦方向信号線 SV1 と SV2 が接続されるためである．

本稿ではコネクタスイッチのプログラミング順決定のために，クロスバー回路内の縦方向信号線の接続状況を表す接続木を利用する．図 7 は図 6 のコンフィギュレーションにおけるコネクタスイッチ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 7 では縦信号線 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コネクタスイッチを表しており，両端のコネクタスイッチ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 個のノード，2 本の信号線が接続されることを表現している．接続木の定

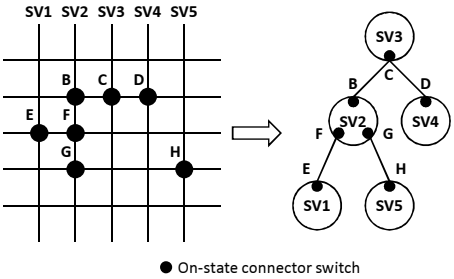


図 7 図 6 のコネクタスイッチを対象とする接続木の例

義より，本稿で対象としているループのないコンフィギュレーションにおいては必ず接続関係を木構造で表現できる．コンフィギュレーションにループが存在するときのみ接続木は木構造にならず，閉路を含むグラフ構造となる．

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換える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．接続木の葉ノードと葉ノードでないノードそれぞれをプログラミングの最後に書き換えた場合に何が起こるか，図 8 を用いて説明する．プログラミングの最後では書き換え対象以外のオンにするべきスイッチは既にすべてオン状態である．図 8(a) では葉ノード SV1 内のコネクタスイッチがプログラミング対象であり，その他のスイッチはすべてオン状態になっている．この場合，ノード SV1 と SV2 はまだ接続されておらず，そのため信号線 SV1 に与えたプログラミング信号は他の縦信号線には伝搬しない．すなわち，スニークパス問題なく葉ノード SV1 内のコネクタスイッチを書き換えることができる．図 8(b) は図 8(a) に相当する回路図であり，この図からも信号線 SV1 に印加した信号が他の縦信号線に伝搬していないことが確認できる．一方，図 8(c) ではプログラミングの最後に葉ノードではないノード SV2 内のコネクタスイッチをプログラムする状況を示している．この場合，既にオン状態であるスイッチを経由してプログラミング信号が他の縦方向信号線に到達し，スニークパス問題が発生することがわかる．図 8(d) に示した回路図からもその様子が確認でき，区別不能となった縦信号線の書き換え対象と同じ位置に存在する原子スイッチが意図せずプログラムされることがわかる．

本稿はスニークパス問題を回避するコネクタスイッチのプログラミング順を決定するため，プログラミングの最後にオン状態にできるスイッチを再帰的に探索する手法を提案する．図 9 に再帰的探索の流れを示す．ここで，各ノードにおいては親ノードに接続するコネクタスイッチと子ノードに接続するコネクタスイッチの 2 種類存在する．例えば，ノード SV2 ではスイッチ B が親ノードに接続するコネクタスイッチであり，スイッチ F と G が子ノードに接続するコネクタスイッチである．各ノードにおいては親ノードに接続するコネクタスイッチに先立って子ノードに接続するすべてのスイッチをプログラムする必要がある．さもなければ，図 8(c) に示したように，子ノードに接続するスイッチのプログラミング時にオン状態スイッチを経由して信号が伝搬し，スニークパス問題が起こる．

以降では，提案する再帰的探索を用いたプログラミング順の決定フローを図 9 を使ってステップごとに説明する．

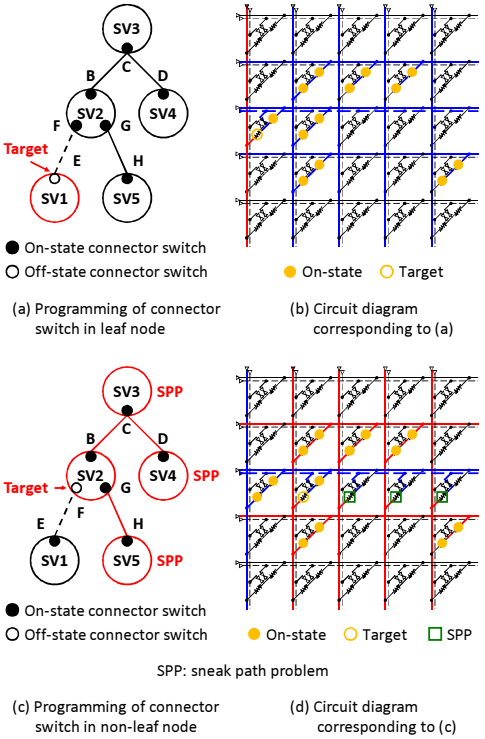


図 8 プログラミングの最後における接続木の葉ノード・非葉ノードの書き換え

先に議論した通り，プログラミングの最後に接続木の葉ノードのみがスニークパス問題なく書き換えることができる．そこで，ノード SV1 内のスイッチ E が最後にプログラムされるスイッチとして選択され，ノード SV1 およびノード SV1・SV2 間のエッジが接続木から削除される．更新された接続木に対して再び探索を行うと，最後に書き換えるスイッチとしてスイッチ H が選択される．ノード SV1 と SV5 が削除されるとノード SV2 は新たな葉ノードとなるため，SV2 内の親ノード SV3 に接続するスイッチ B をプログラムできる．このように 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換える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．最終的にルートノード以外のノードがすべて削除された時点で再帰的探索は終了する．なお，探索終了後に，図 9 のスイッチ C，F，G などの各ノードにおいて子ノードに接続するスイッチが残っている．これらのスイッチについては先述したように，再帰的探索で選択されたスイッチより先にプログラムする限り任意の順番で書き換えられる．

対象コンフィギュレーションによっては複数の探索木が構築されることがある．この場合，各接続木は他の木と接続していないため，プログラミング信号が他の木に伝搬することはない．したがって，各接続木を独立に取り扱い，提案手法によりスニークパス問題を回避するプログラミング順を求めることができる．

ここまで，ビアスイッチをオン状態にするプログラミング順について議論した．ビアスイッチをオフ状態にする際は信号線と制御線に与える電位が逆になる以外は同一の動作であるため，提案手法で決定したプログラミング順を逆

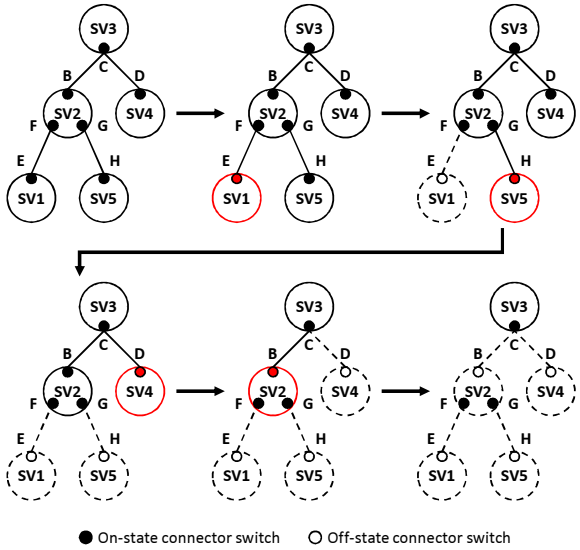


図 9　プログラミングの最後に書き換えられるスイッチの再帰的探索

転させることでスニークパス問題なくすべてのビアスイッチをオフ状態にできる．本節で述べたフローは任意の接続木，つまりループを含まない任意のコンフィギュレーションに対して適用可能である．

5. シミュレーション評価

5.1 提案手法の有効性検証

前節で，ループを含まない任意のコンフィギュレーションに対してスニークパス問題を回避するプログラミング順が提案手法により発見可能であることを示した．ここでは，提案手法の有効性をシミュレーション評価により確認する．提案手法を実装し，得られたプログラミング順にしたがって各プログラミングステップにおいてスニークパス問題の有無を検証した．小規模クロスバー回路に対する網羅的評価および大規模クロスバー回路に対するモンテカルロ評価を行った．網羅的シミュレーションでは 2x2，3x3，4x4，5x5 の 4 種類のクロスバー回路の全通りのコンフィギュレーションに対して提案手法を適用し，その有効性を検証した．一方，モンテカルロシミュレーションにおいては 100x100 のクロスバー回路を対象にランダムで 10,000 通りのコンフィギュレーションを生成し，各コンフィギュレーションで提案手法によるスニークパス問題回避を検証した．期待通り，すべての評価において，ループのないコンフィギュレーションで提案手法によりスニークパス問題が解決できることを確認できた．

5.2 提案手法の利点

次に，提案手法の利点について述べる．2.3 節で説明したように，従来はビアスイッチ FPGA の一部のコンフィギュレーションを禁止するプログラミング制約を導入することでスニークパスに対策していた．したがって，利用可能なコンフィギュレーションパターン数は減少し，ルーティングの柔軟性が悪化する．一方，本稿の提案手法は本質的にスニークパス問題回避が不可能なループを含むコンフィギュレーションを除き，すべてのコンフィギュレーションに対してスニークパスを回避するプログラミング順

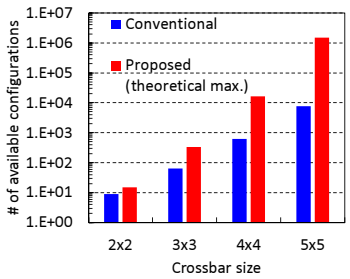


図 10　提案手法と従来手法における利用可能コンフィギュレーション数の比較

を発見可能である．図 10 は小規模クロスバー回路における利用可能コンフィギュレーション数の比較結果である．この評価において，従来手法は複数のオン状態ビアスイッチを含む信号線が縦横の両方向に存在するコンフィギュレーションを禁止する．グラフより，従来手法と比較して提案手法により使用可能なコンフィギュレーション数が大幅に増加することが確認できる．5x5 という比較的小さいクロスバー回路においても，利用可能数が 2 桁以上増加している．また，クロスバー回路のサイズ増大に伴って利用可能コンフィギュレーション数の増加率が高まっていることもわかる．すなわち，提案手法によって新たに対応可能になるコンフィギュレーション数は実用的な規模のクロスバー回路において非常に大きくなると考えられる．

提案手法はクロスバー回路の構造の変更やハードウェアのオーバヘッドなく適用可能であるため，大きな欠点は存在しない．スニークパスを回避するプログラミング順の決定のためにある程度の計算は必要であるが，提案したアルゴリズムにより効率的に計算が可能である．

6. 結論

本稿ではビアスイッチ FPGA においてスニークパス問題を引き起こすクロスバーの回路状況を考察し，ループを含むコンフィギュレーションでは本質的にスニークパス問題が回避できないことを明らかにした．一方，ループのないコンフィギュレーションにおいては常にスニークパスを回避するビアスイッチのプログラミング順が存在することを示し，そのような順番を求めるアルゴリズムを提案した．また，提案手法の有効性をシミュレーション評価により確認した．提案手法の適用により，実用上すべてのコンフィギュレーションにおいてスニークパス問題を解決可能である．今後の課題として計算複雑度の評価が挙げられる．

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] N. Banno, et al., IEDM, 2015.
[4] N. Banno, et al., IEDM, 2016.
[5] M. Miyamura, et al., ISQED, 2014.
[6] H. Ochi, et al., IEEE Trans. VLSI Syst., 2018.