ISSN 2186-5647

2-18

診断容易化のためのレジスタバインディング手法の一考察

日大生産工（院）　○久保倉 修司　　日大生産工　細川 利典
明治大・情報コミュニケーション　　山崎 浩二
京都産業大・コンピュータ理工　　吉村 正義

1. まえがき

近年の超大規模集積回路(Very Large Scale Integrated circuits：VLSI)の大規模化，高集積化に伴い，VLSIの異常動作が発生した場合に物理的な原因を特定する故障解析[1]を行う．故障解析は，歩留まりの向上のために重要であり，電子顕微鏡などを用いて故障VLSI内部を観測するため，多大なコストを要する．そのため，故障VLSI内部に存在する可能性のある故障を事前に絞り込んでおく故障診断[2]が，故障解析コストの低減のために重要となる．故障診断の実施により，故障VLSI内部の故障個所を推定した被疑故障集合が得られ，被疑故障集合の要素数が少ないことが診断分解能の向上へとつながる．

診断分解能を向上させる手法として２種類の手法が提案されている．１つ目は，高診断分解能を指向したテスト生成法[3][4]である．故障診断では，テスト集合を用いて故障を推定するため，テスト集合の診断品質が重要となる．文献[3]では，テスト集合に診断用テストパターンを追加する手法が提案されている．しかしながら，診断パターンを追加することによるテストパターン数の増大が課題となっている．それゆえ，文献[4]ではテスト集合に存在する診断分解能向上に比較的寄与していないテストパターンを，故障検出効率を維持しながら診断分解能を向上させるテストパターンに置換する手法が提案されている．

診断分解能を向上させる２つ目の手法として，レジスタ転送レベル(Register Transfer Level：RTL)における故障診断容易化設計(Design for Diagnosability：DFD)手法[5][6]が提案されている．文献[5]では，抽象度の高いレジスタ転送レベル(Resigter Transfar Level：RTL)に着目したDFD手法が提案されている．RTL回路はデータパスとコントローラから成り，データパスからコントローラへの状態信号とコントローラからデータパスへの制御信号が互いに接続されている．コントローラは有限状態機械(Finite State Machine：FSM)で設計されていると仮定し，状態遷移時に制御信号がデータパスに供給される．それらの制御信号値には多数のドントケア(X)が含まれており，設計の自由度が存在する．一般に，論理合成時には面積小化や低消費電力を指向してXに論理値を割当てるが，文献[6]では各状態遷移においてデータパス中のハードウェア要素ペアが識別可能であることが，診断分解能の向上に重要であると考え，各状態遷移のドントケアに対して識別可能故障ペア推定数を最大化するように論理値を割当てる手法が提案されている．

抽象度の高い高位合成(High-Level synthesis：HLS)においてDFDを考慮することは，RTLでのX割当てなどの回路構造を変更するDFD手法と異なり，回路の構造自体を変更することができるため，RTLより高い設計の自由度が存在する．

本論文では，スキャン設計回路を対象としたHLSにおける診断容易化のためのレジスタバインディング手法の一考察として，１つのベンチマーク回路に対して，スケジューリングアルゴリズムやバインディングアルゴリズムを変更して様々なRTL回路を生成し，それぞれについて故障ペア推定数の割合[6]を求め，診断分解能向上に寄与している回路特性の評価を行い，診断容易化レジスタバインディングアルゴリズムの考察を行う．

本論文の構成は以下のとおりである．第2章では，代表的なHLSのアルゴリズムの概要を述べる．第3章では解析方法と解析結果を示し，第4章では解析結果から診断容易な回路の特徴を考察し，診断容易な回路を生成するレジスタバインディングアルゴリズムを考察する．

2. 代表的な高位合成アルゴリズム

高位合成を最適化するうえでスケジューリングのアルゴリズムは重要となる．本章では，代表的なスケジューリング方法として，2.1節ではASAP(As Soon As Possible：ASAP) / ALAP(As Late As Possible：ALAP)スケジューリングの概要について述べ，2.2節ではフォースディレクティッドスケジューリング(Force Derected Scheduling：FDS)[7]の概要について例題DFGを用いて述べる．2.3節ではバインディングアルゴリズムであるレフトエッジアルゴリズム(Left edge algorithm)[8]の概要について説明する．

2.1. ASAP/ALAPスケジューリング

DFGの演算器の個数などの制約なしにスケジューリングを考える場合，考慮すべきことはDFGの演算順序の依存関係のみとなる．このようなスケジューリングとして，各演算操作を可能な限り早いコントロールステップに割当てるASAPスケジューリングと，各演算を可能な限り遅いコントロールステップに割当てるALAPスケジューリングがある．図1に例題DFGにALAPスケジューリングとASAPスケジューリングを適用したスケジュール済みDFGを示す．

An Analysis of Register Binding Methods for Synthesis for Diagnosability

Shuji Kubokura, Toshinori Hosokawa, Koji Yamazaki and Masayosi Yoshimura

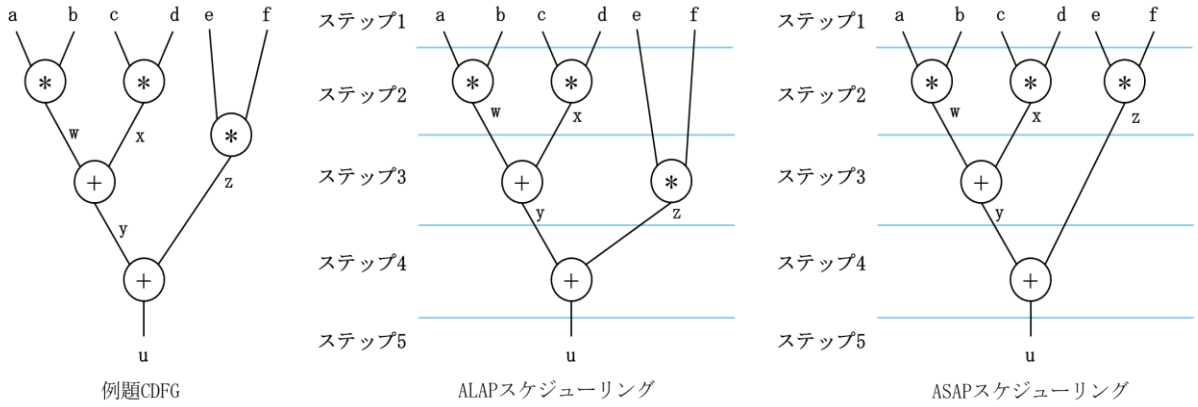

図 1 ALAP スケジューリング / ASAP スケジューリング

ALAPスケジューリングとASAPスケジューリングを行うことで各演算の移動度[7]を求めることができる．移動度は以下の式で求められる．

$$\boldsymbol{move(op_i) = ALAP(op_i) - ASAP(op_i) + 1}$$

この式は，ある演算op_iがALAPでスケジューリングされているコントロールステップを$ALAP(op_i)$，ASAPでスケジューリングされているコントロールステップを$ASAP(op_i)$としている．移動度$move(op_i)$は$ALAP(op_i) - ASAP(op_i)$で求めることができ，eとfを入力としている乗算について移動度を求めると，$3 - 2 + 1 = 2$となる．

各コントロールステップの演算の個数に着目すると，ステップ１において乗算数がALAPスケジューリングでは2つ，ASAPスケジューリングでは3つと，演算器の必要個数に差があることがわかる．このことから，スケジューリングの方法により，演算が割当られるコントロールステップや演算器の個数が異なることがわかる．

```
Call ALAP(V);
Call ASAP(V);
while there exists O_i such that E_i ≠ L_i do
  MaxGain = -∞
  for each O_i, E_i ≠ L_i do
    for each j, E_i ≤ j ≤ L_i do
      S_work = SCHEDULE_OP(S_current, O_i, S_j);
      ADJUST_DISTRIBUTION(S_work, O_i, S_j);
      if COST(S_current) - COST(S_work) > MaxGain then
        MaxGain = COST(S_current) – COST(S_work);
        BestOp = O_i; BestStep = S_j;
      end if
    endfor
  endfor
  S_current = SCHEDULE_OP(S_current, BestOp, BestStep);
  ADJUST_DISTRIBUTION(S_current, BestOp, BestStep);
endwhile
```

図 2 FDS アルゴリズム

2.2. フォースディレクティッドスケジューリング（FDS）

ASAPスケジューリングとALAPスケジューリングより求められる移動度を利用するものとして，FDSがある．FDSは，各演算操作が移動度内の各時刻に等確率でスケジュールされると仮定し，演算確率の最大値がなるべく小さくなるようにスケジューリングする手法である．図2にFDSのアルゴリズムを示す．FDSは以下のStep1からStep8の操作で構成される．

Step1 ：ASAPスケジューリングとALAPスケジューリングを実施
Step2 ：Step1で得られた結果から，各演算の移動度を取得
Step3 ：移動度1の演算を各コントロールステップに割当てる
Step4 ：各コントロールステップに対して各演算の演算確率を計算
Step5 ：コントロールステップに割当られていない演算操作がある場合，Step6へ進む．そうでない場合は，終了する
Step6 ：コントロールステップごとに演算確率の合計値を計算
Step7 ：演算確率が最大の演算を演算確率の合計値が最小のコントロールステップにStep4へ戻る

図1に示すDFGを用いると，ASAPやALAPは図1と同様にスケジューリングされる．Step2で各乗算操作の演算確率を計算したものを図3に示す．Step3で移動度１となる乗算操作はコントロールステップに割当てられる．そのため，2つの乗算はステップ2へ割当てられる．Step4にて，コントロールステップに割当てられていない乗算が存在しているため，Step5のステップごとの演算確率の合計値を計算すと，ステップ2の合計値は2.5，ステップ3の合計値は0.5となる．Step6により，残りの乗算はステップ3へ割当てられる．例題DFGにFDSアルゴ

リズムを適用すると，ALAPスケジューリングで得られたスケジュール済みDFGと同じ結果が得られる．

2.3. レフトエッジアルゴリズム

レジスタバインディングとは，スケジュール済みDFGにおいて，各内部変数をレジスタに割当てる操作のことである．各コントロールステップの境界では，計算された演算結果を保持し，以降の演算でその値が利用され終わるまでその値を保持する必要がある．その値を保持するコントロールステップを各内部変数に割当てられるレジスタのライフタイムという．図4にALAPスケジューリングされたスケジュール済みDFGにおける各内部変数のレジスタのライフタイムを示す．

各内部変数において，レジスタのライフタイムが重複しなければ，レジスタを共有することができる．レジスタのライフタイムを使ったバインディングアルゴリズムとして，レフトエッジアルゴリズムがあり，このアリゴリズムはレジスタ数が最小となることを指向している．レフトエッジアルゴリズムは以下の手順で行われる．

Step1 ：各内部変数のライフタイムの開始コントロールステップで変数集合Lを昇順ソート
Step2 ：使用されているレジスタ数iを0に初期化
Step3 ：Lが空集合であれば処理を終了する
Step4 ：レジスタiに割り当てられた変数集合Sを空集合に初期化し，S中にある変数のライフタイムの最大値rを0に初期化
Step5 ：Lを順に走査したとき，ライフタイムの開始コントロールステップがrより大きい変数sがある限り，Step5-1からStep5-3を繰り返す
Step5-1 ：$S = S \cup \{s\}$
Step5-2 ：sの終了コントロールステップを rに代入
Step5-3 ：Lからsを削除
Step6 ：iをインクリメント
Step7 ：S中のすべての変数をi番目のレジスタに割当て，Step3へ戻る

ALAPスケジューリングされたスケジュール済みDFGへレフトエッジアルゴリズムを適用し，レジスタバインディングを行うと，レジスタ数は，R0:{a, w, y, u}，R1:{b, x}，R2:{c, z}，R3:{d}，R4:{e}，R5:{f}の6個となる．

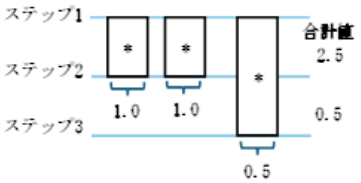

図 3 乗算における各ステップ演算確率合計値

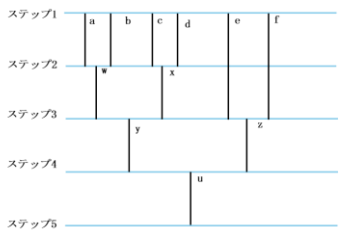

図 4 レジスタのライフタイム

3. 解析

本章では，診断容易な回路の特徴の解析方法および解析結果を述べる．3.1節では，解析方法について述べ，3.2節では解析方法に基づいた解析結果を示す．

3.1. 解析方法

診断容易な回路を解析するにあたり，ベンチマーク回路であるex2[9]について，レジスタ数，演算器数を調べるために，ALAPおよびASAP，FDSスケジューリングを行ったところ，2種類のスケジュール済みDFGが得られた．それら2種類のスケジュール済みDFGにそれぞれレフトエッジバインディングを行ない，RTL回路を生成した．生成したRTL回路のレジスタ数，演算器数を表.1に示す．

表 1 ex2 で得られた RTL 回路の特徴

ex2	レジスタ数	演算器数	
		乗算器	減算器
ALAP/FDS	5	2	1
ASAP	5	3	1

得られた特徴をもとに，レジスタ数を5個，乗算器を2個，減算器を1個の制約で，ALAP/FDSで得られたスケジュール済みDFGに対して考慮されるすべてのバインディング結果を列挙することで診断容易な回路の特性を解析した．

解析にあたり，RTL回路のみでは厳密な診断分解能を算出することは困難である．そのため，診断分解能を算出する際，文献[6]で提案されている識別可能故障ペア推定数を用い，以下の式で近似的に診断分解能を算出した．

$$DC(\%) = \frac{\textit{識別可能故障ペア推定数} * 100}{\textit{全ハードウェア要素が識別可能な時の識別可能故障ペア推定数の和}}$$

また，各ハードウェア要素から到達可能な観測点数および制御点数について解析を行い，ハードウェア要素の平均合計テスト可能回数およびハードウェア要素数を解析し，回路面積の増加についても解析し，診断容易な回路の特性を解析した．

3.2 解析結果

ベンチマーク回路ex2[9]において，制約のもとで得られたRTLデータパス回路は合計で4800種類であった．

その中から上位3つ，中位3つ，下位3つを抽出した解析結果を表.2に示す.

各ハードウェア要素が持つ観測点数の平均と分解能の関係について着目すると，ほとんどの回路について各ハードウェア要素が持つ観測点数が多いほど分解能が高くなっている傾向がある結果が得られた．また，各ハードウェアが持つ制御点数の平均と分解能の関係に着目すると，各ハードウェア要素から到達可能な制御点数が多いほど分解能が高いという結果が得られた.

4. 考察

本章は得られた解析結果をもとに考察を行う．4.1節では，診断容易な回路の特性について考察し，4.2節では診断容易な回路を生成するレジスタバインディングアルゴリズムの考察を行う.

4.1. 診断容易な回路の特性

表.2にある回路特性を解析するために，それぞれのレジスタバインディングを調査した結果，各ハードウェア要素からの観測点数および制御点数が多いという特徴を持つRTL回路が診断容易な回路である可能性が高いという結果が得られた．そのため，HLSにおいて分解能を向上させるためには，下記の特徴をもつRTL回路を生成するアルゴリズムを適用する必要があると考えられる.

・各ハードウェア要素からの観測点数が多い
・各ハードウェア要素からの制御点数が多い

4.2. レジスタバインディングアルゴリズムの一考察

解析結果から，各ハードウェア要素から可能な限り多くのレジスタへ到達できるようにレジスタバインディングを行うことで，診断容易なRTL回路を生成することができることが考えられる．今後の課題として，全ハードウェア要素が識別可能な時の識別可能故障ペア推定数の和の算出が正確でないため，改善を行う必要がある．また，解析結果に基づき演算操作の結果を可能な限り多くのレジスタへ伝搬できるようなレジスタバインディングを提案することや制御経路についての考察が課題となる.

参考文献

1) 藤原 秀雄，“ディジタルシステムの設計とテスト”，工学図書株式会社，(2004).
2) 高松雄三，佐藤康夫，高橋寛，樋上喜信，山崎浩二，“論理回路の故障診断法―外部出力応答に基づく故障箇所指摘法の発展―”，電子情報通信学会論文誌 D，94(1)，(2011)，pp.266-279.
3) 千田 裕弥，細川 利典，山崎 浩二，“故障診断分解能向上のための複数故障ペア識別パターン生成法”，信学技報，vol.122，no.DC-393，DC2022-83，(2023)，pp.6-11.
4) 青野 竜弥，細川 利典，吉村 正義，山崎 浩二，“完全故障検出効率と完全診断分解能を両立するテストパターン生成法”，信学技報，vol.124，no.148，DC2024-17，(2024)，pp.5-10.
5) 大塚 裕衣，千田 裕弥，徐 浩豊，細川 利典，山崎 浩二，“識別可能ハードウェア要素ペア数最大化のためのコントローラの制御信号のドントケア割当て法”，信学技報，vol.123，no.146，DC2023-12，(2023)，pp.25-30.
6) 大塚 裕衣，細川 利典，吉村 正義，山崎 浩二，“レジスタ転送レベルにおける識別可能故障ペア推定数最大化のための状態遷移のドントケア割当て手法”，DAシンポジウム2024論文集，(2024)，pp.147-154
7) Paulin, Pierre G., and John P. Knight., “Force-directed scheduling for the behavioral synthesis of ASICs.″, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 8.6, (1989), pp.661-679.
8) Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, Steve Y-L Lin, “High-Level Synthesis: Introduction to Chip and System Design”, Kluwer Academic Publishers, (1992).
9) M.T.-C. Lee, “High-Level Test Synthesis of Digital VLSI Circuits”, Artech House Publishers, (1997).

表 2. 診断容易な RTL 回路の解析結果

Name	*Ave_obs*	*Score*	*Num_Pair*	*DC(%)*	*Num_testable*		*port*	*Ave_reg*
					All	*Ave*		
Top1	5	119321551	116701597	102	409	3.97	103	4
Top2	5	188956801	116481848	162	391	3.95	99	4
Top3	5	118851582	116412906	102	397	3.93	101	4
Mid1	3	117435003	115059781	102	321	3.31	97	2
Mid2	4	111699220	109195135	102	356	3.91	91	3
Mid3	4	111596467	109127857	102	375	4.03	93	3
Worst3	3	97239365	94808460	102	276	3.68	75	2
Worst2	2	96707429	94281228	102	256	3.41	75	1
Worst1	2	94286226	91800553	102	278	3.86	72	2