1L－2

実行不能パスを考慮したＨＤＬ記述のプログラムスライシング・アルゴリズム

牧川 潔志, 一ノ瀬 茂, 岩井原 瑞穂, 安浦 寛人
九州大学 大学院システム情報科学研究科 情報工学専攻

1　はじめに

ハードウェア記述言語 (HDL) を用いた設計手法の普及に伴ない，ハードウェア記述の設計検証，保守，再利用などの問題はさらに重要さを増すと思われる．ソフトウェア工学の分野では，これらの問題に対し，プロフラムスライシングと呼ばれる手法に基づく研究が多くなされている．プログラムスライシングはプログラム中の文の間の依存関係を解析する手法であり，1982年にWeiserによって提案されたものであり [3]，現在ではテスト，デバッグ，保守など広範囲に応用されている [5]．著者らは標準的なHDLの一つであるVHDLに適用すること研究している．[1, 2].

VHDLにプログラムスライシングを適用した場合，初等的な解析によって抽出されたスライスには，決して実行されない経路 (実行不能経路: infeasible path) が含まれることが多い．スライスからこの実行不能経路を探索し除去することによって，より小さなスライスを得ることは，より効率の良いマイクロプロセッサの動作の解析や，動作の抽出や設計の分割を可能にする．

本稿では，VHDLに対するプログラムスライシングを述べ，VHDLの信号代入において生じる実行不能経路を定式化する．さらに教育用マイクロプロセッサKUE-CHIP2の実行不能経路の例を示す．

2　VHDL記述のプログラムスライシング

プログラムスライシング (以降，スライシング) はプログラム中の注目した動作に関係する記述箇所を抽出し，プログラムを簡約化するソフトウェア工学の技術である [3]．スライシングにより求められるプログラム片をスライスと呼び，注目する動作が保存されている実行可能な部分プログラムであり，プログラムの解析，デバッグなど様々な応用がなされている．

2.1　VHDLの依存関係

VHDLの依存関係を示す．

- **制御依存:** 文 s_1 と s_2 において，s_1 をIFまたはWHILEのフロー制御文とする．s_2 が実行されるか否かが，s_1 の実行結果に直接依存するとき，s_1 から s_2 への**制御依存**があるという．

- **データ依存:** 文 s_1 と s_2，および変数 v について，s_1 で v に代入された値が s_2 に到達するとき，s_1 から s_2 へのデータ**依存**があるという

VHDLはプロセス文の集合からなる．変数は１つのプロセス文内部のみで参照される局所変数である．それに対し，１つのプロセス文内部はIF-THENおよびループなどからなる逐次実行が行なわれる．プロセス間の通信はすべて信号 (signal) によって行なわれる．プロセス文の入れ子も許されるが，フラットなプロセス文の集合に変換することができる．

信号代入はデルタ時間または単位時間の整数倍の遅延を伴なう．信号代入により，信号への新しい値がスケジューリングされ，定められた時間の後，信号値の更新が行なわれる．これをトランザクションという．同一の信号への複数の文から代入も起り得るが，決定関数 (resolution function) によりひとつの値が決定され，それが信号の次の値となる．

プロセス文内の逐次実行は遅延を伴なわず (ゼロ遅延)，かつ信号代入は必ずデルタ時間以上の遅延を伴なうため，ある信号に代入があったとしても，同一プロセス文内でその新しい信号値を参照することはできない．そのときは信号値の古い値が参照される．同一プロセス文が新しい信号値を参照できるのは，WAIT文でプロセスが一旦停止し，しかるべき遅延時間を経過したのちか，そのプロセスが終了してあらたに起動されたときである．

プロセス間の信号による通信の依存関係として，信号依存を導入する (図1参照)．ただし実行不能経路を考慮するため，文献 [2] の定義を簡略化している．

- **信号依存:** p_1 および p_2 をプロセス文とし，sig を信号，s_1 を p_1 中のある信号代入文とする．このとき，sig に対する代入により生じたトランザクションにより，p_2 が活性化されるか，または s_1 で代入された値が p_2 内の文で参照されるとき，s_1 から p_2 への信号依存があるという．

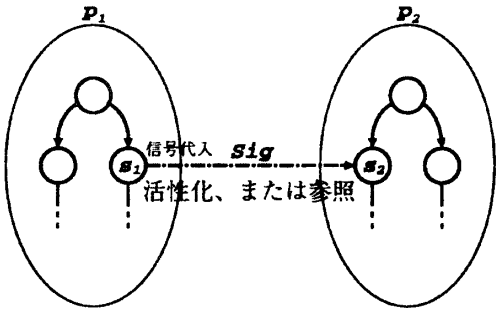

図 1: 信号依存

上記での活性化が起きる場合とは，p_2 の反応リスト (sensitivity list) に sig が含まれている場合と等価である．反応リストに sig が含まれない場合は，p_2 は活性化されない．しかし反応リストに sig がなくても p_2 内で sig が参照されることがある．そのときも s_1 での代入した値が，他の信号で活性化された p_2 での動作に影響を与えうる (ただし同期回路を合成するために，反応リストに参照するすべての信号を列挙した記述がなされること多い)．

また，sig への代入により p_2 が活性化されるが，そのときは sig の値を参照する文へは到達せず，別の信号で p_2 が活性化したときに sig の参照が行なわれることもあり得る．つまり，sig への代入により p_2 は活性化されるが，sig の値が参照されるのは別の信号代入による場合である．

"Program Slicing Algorithm for HDL descriptions considering infeasible paths," K. Makigawa, S. Ichinose, M. Iwaihara, and H. Yasuura, Kyushu University

2.2 スライスの定義

以下にVHDLのスタティックスライスの定義を示す.

スタティックスライスVHDL記述のスタティックスライスの判定基準とは2項のタップル $(s, Sig/V)$ である. この s は記述中の文で, Sig/V は s で使用される信号, あるいは変数の集合である. 与えられているスタティックスライスの判定基準 $(s, Sig/V)$ 上のVHDL記述のスタティックスライス $SS(s, Sig/V)$ とは, s の実行の開始や終了, および s 中の実行結果に影響を与える記述中の文の集合である.

2.3 信号代入に関する実行不能経路

信号を介したプロセス間の依存関係について, データ依存, 制御依存および信号依存を単に推移的に接続した経路では, 実際には起り得ない依存の経路が含まれることが多い. これを**実行不能経路 (infeasible path)** と呼ぶことにする.

実行不能経路が起り得る状況を分類してみる. 信号代入は, 他のプロセスを活性化するという制御依存的な性質と, 代入された値を他のプロセスが参照するというデータ依存的な性質の両面を持つ. 両者について, 実行不能経路が起り得ると考えられる.

p_1 および p_2 をプロセス文とし, p_1 のある信号代入文 s_1 により, 信号 sig に値が代入されるものとする. s_2 を p_2 のある文とする.

1. **信号代入の制御依存的な性質に関する実行不能経路:** s_1 による sig への信号代入により, p_2 が活性化されるが, s_2 に制御が到達することが起り得ない (そのような入力列が存在しない).

2. **信号代入のデータ依存的な性質に関する実行不能経路:** s_1 により sig へ値が設定されるが, s_2 で sig を参照するときには常に他の信号代入によって sig の値が更新されており, s_1 で代入された値が s_2 で参照されることは起こり得ない.

本稿では, 以下において信号代入の制御依存的な性質に関する実行不能経路について考察する.

制御依存的な性質に関する実行不能経路が起きる例として, 信号代入が起きるための条件と, 注目する文に到達する条件が同時に成立しない場合がある. ただし, ここでいう同時とは, VHDLの実行モデルにおける有限個のデルタサイクル以内という意味である. よって, 証明する方法としては次の2つが考えられる.

- 代入側の到達条件, および参照側の到達条件を求める
- 両条件の論理積が恒偽であることを示す
  解法アルゴリズムとしては,
  - −論理式の充足可能性問題 (SAT) による解法
  - −BDDによる解法

  が考えられる.

3 マイクロプロセッサ記述における実行不能経路

実行不能経路を教育用マイクロプロセッサKUE-CHIP2において検出する. KUE-CHIP2は京都大学で開発された8ビットマイクロプロセッサであり, 教育を目的として設計されているため, 簡素なアーキテクチャを持っている.

今回はKUE-CHIP2の仕様書 [4] を使用して, 実行不能経路を求めることを試みた.

信号代入の制御依存的な性質に関する実行不能経路を証明するために, 代入側の到達条件および参照側の到達条件を求め, 両条件の論理積が恒偽であるかを調べた.

入力バス DBi に注目し, そこにデータを出力するのはアキュームレータ ACC, インデックスレジスタ IX, 内部メモリ IMEM の3つであり, それ上のデータをラッチするのは演算器 ALU, 命令レジスタ IR, プログラムカウンタ PC の3つであるので, 考えられる経路は $3 \times 3 = 9$ 通りである (図2参照).

上記の経路のうち, 実行不能となる経路は (1) ACC → DBi → PC, (2) ACC → DBi → IR, (3) IX → DBi → PC, (4) IX → DBi → IR, の4つであった. また実行不能でない経路は (5) ACC → DBi → ALU, (6) IX → DBi → ALU, (7) IMEM → DBi → PC, (8) IMEM → DBi → IR, (9) IMEM → DBi → ALU の4つであった.

この結果により, PC および IR レジスタ周辺のスライスを計算する際は, DBi からの ACC および IX からの影響を無視することができる.

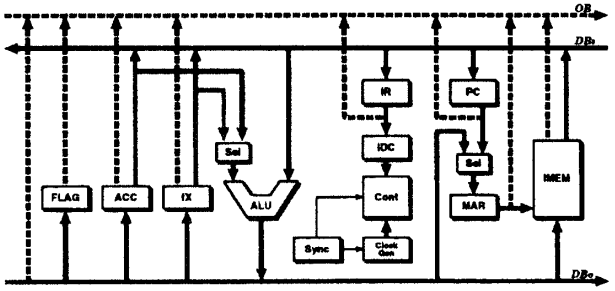

図 2: KUE-CHIP2 の構造

4 おわりに

本稿では, プログラムスライシングをハードウェア記述言語に適用する際に必要となる実行不能経路について検討し, 定式化した. また実際のマイクロプロセッサ記述における実行不能経路の例を示し, それらがスライスの計算に利用できることを示した.

参考文献

[1]一ノ瀬茂, 野村 雅也, 岩井原瑞穂, 安浦寛人, "VHDL 記述のプログラムスライシング — VHDL における依存関係とそのグラフ表現 —," 第9回回路とシステム軽井沢ワークショップ, pp.377–382, 1996年4月.

[2]Iwaihara, M., M. Nomura, S. Ichinose and H. Yasuura, "Program Slicing on VHDL Descriptions and Its Applications, " *Proc. 3rd Asian-Pacific Conf. Hardware Description Languages*, pp. 132–139, Jan. 1996.

[3]Weiser, M., "Program Slicing," *IEEE Trans. Software Eng.*, Vol. SE-10, No. 4, July 1984.

[4]神原 弘之, 越智 裕之, 澤田 宏, 浜口 清治, 岡田 和久, 上嶋 明, 安浦 寛人, "KUE-CHIP2 設計ドキュメント Version 1.10", 1993年1月.

[5]下村隆夫, プログラムスライシング技術と応用, 共立出版,1995年7月.