4K−5

プロセッサのパイプライン制御に関する動作レベル設計／検証支援

岩下洋哲　中田恒夫　広瀬文保
(株)富士通研究所

1　はじめに

プロセッサの初期設計およびその論理検証を支援するシステムについて報告する．我々の考える設計／検証支援（図1）では，目的の機構を検証するためのテストプログラムと目的の機構を実現するVHDL動作記述を仕様記述から自動生成する．設計者は仕様記述あるいは生成された動作記述をもとに設計を詳細化するが，そこで発生する設計誤りの検証は，設計者の記述と自動生成された動作記述の両方にテストプログラムを与え，実行結果を比較することによって達成できる．本論文では，基本的なパイプライン制御機構に対して実現した，テストプログラムおよび動作記述の自動生成について報告する．

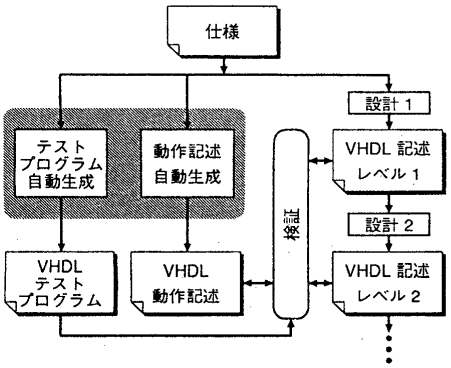

図1　設計／検証支援

2　支援モデル

本論文では，以下のような基本的なパイプライン構造を持つプロセッサを対象とする．

- すべての命令を同じステージ構成で処理する．
- すべてのステージを1クロックで完了する．
- 命令の実行順序は逐次実行の場合と同じ．
- 割り込み，例外処理は発生しない．

Behavioral Design and Test Assistance for Pipelined Processors
Hiroaki Iwashita, Tsuneo Nakata, and Fumiyasu Hirose
Fujitsu Laboratories Ltd.

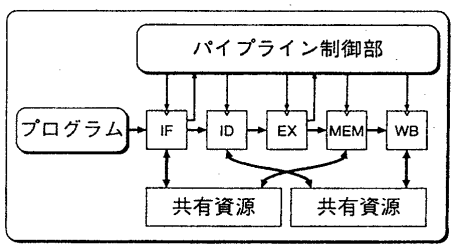

図2　プロセッサのモデル

プロセッサのモデルを図2に示す．パイプライン制御部は，入力として命令グループとオペランド，プロセッサの内部信号，クロック信号，出力としてクロックサイクル毎に各ステージの実行／停止を制御する信号を持つ機能ブロックであると考える．なお，命令グループとは，パイプライン制御上等価な命令をまとめたものである．

ここでは，構造／データ／制御ハザードに対するインターロック機構を考える．ハザードを発生する命令パターンをすべて列挙し，VHDL記述を生成するために，仕様記述は以下の情報を含む．

- 命令グループ，ステージの構成，パイプライン制御部の入力ポートに関する宣言
- 複数のステージに共有されており，ハザードの原因となり得るハードウェア資源（共有資源）
- 資源の占有，データの読み出しおよび書き込みなどの，共有資源に対する動作
- 共有資源を使用するステージ
- 共有資源に対する動作が行なわれるための，ステージの状態（命令グループ，内部信号値，共有資源のインデックスに関する条件）

また，以下の条件を満たすようにパイプラインを制御する．

- 実行結果が逐次実行の場合と同じである．
- 最少のクロック数で実行が完了する．

この仮定に基づき，パイプライン制御部の動作を仕様から一意に決定する．

3 テストプログラム／動作記述自動生成

ここでは，ハザードが発生するようなプロセッサの内部状態に関する条件をハザードパターンと呼び，その集合 H を次のように定義する．

$$H = \left\{ (s_1, q_1, s_2, q_2) \middle| \begin{array}{l} \text{ステージ } s_1 \text{ が状態 } q_1 \text{ を持つ} \\ \text{とき，ステージ } s_2 \text{ は状態 } q_2 \text{ を} \\ \text{持つことができない．} \end{array} \right\}$$

また，ステージの集合を S，ステージ s が共有資源 c を占有する状態の集合を X_{sc}，ステージ s が共有資源 c をリードする状態の集合を R_{sc}，ステージ s が共有資源 c にライトする状態の集合を W_{sc} とし，s_1 が s_2 より後であることを $s_2 < s_1$，s_1 が s_2 より後または両者が同じステージであることを $s_2 \leq s_1$ と表す．共有資源 c における構造ハザードのパターンを求める手続きを図3 (a)，代表的なデータハザードである RAW ハザードのパターンを求める手続きを図3 (b) に示す．WAR ハザード，WAW ハザードについても同様に求めることができる．制御ハザードはプログラムカウンタにおける RAW ハザードとして取り扱う．

```
foreach q1, q2 (q1 ∈ X_{s1c} ; q2 ∈ X_{s2c} ; s1, s2 ∈ S ; s2 < s1) {
  H ← H ∪ {(s1, q1, s2, q2)};
}
```

(a) 構造ハザードのパターンを求める手続き

```
foreach q1, q2 (q1 ∈ W_{s1c} ; q2 ∈ R_{s2c} ; s1, s2 ∈ S ; s2 < s1) {
  foreach s (s ∈ S ; s2 < s ≤ s1) {
    H ← H ∪ {(s, q1, s2, q2)};
  }
}
```

(b) RAW ハザードのパターンを求める手続き

図3　ハザードパターンを求める手続き

テストプログラムは，列挙されたハザードパターン (s_1, q_1, s_2, q_2) を引き起こす命令列である．それぞれのハザードパターンを次のようなコード列に変換することにより，テストプログラムを生成する．

1. 状態 q_1 を実現するコード
2. s_1，s_2 の間のステージ数に等しい数の，パイプラインの流れを乱さない命令列（NOP）
3. 状態 q_2 を実現するコード
4. ステージ総数 − 1 個の，パイプラインの流れを乱さない命令列（NOP）

パイプライン制御部の動作記述も，列挙されたハザードパターンに基づいて生成する．それぞれのハザードパターン (s_1, q_1, s_2, q_2) は，以下の場合に次のクロックサイクルでのステージ s_2 の処理を止めなければならないことを示している．

- s_1 の前のステージが状態 q_1 を持ち，s_2 の前のステージが状態 q_2 を持つ場合
- 構造ハザードのパターンでは，上に加えて，次のクロックサイクルでステージ s_1 が実行される場合

パイプライン制御部は，状態を記憶する内部ステージと制御信号を求める組合せ論理で構成する．入力ポートからの信号は対応する内部ステージに送られ，外部のパイプラインと同期して制御部内を流れる．それらの信号から各ハザードパターンに対応する状態が検出され，制御信号が求められる．

4 実験結果

我々は，テストプログラム，動作記述，およびテスト環境を自動生成する実験システムを約 2,500 行の Perl プログラムで実現した．実験は SPARCstation2（28 MIPS）上で行なった．

DLX[1] の基本的なパイプラインに対し，ハザードパターンの列挙に 1.0 秒，テストプログラムの生成に 2.5 秒，その他の VHDL 記述の生成に 2.0 秒の CPU 時間を消費した．列挙されたハザードパターンは 26 通り，生成されたテストプログラムは 187 の命令コードから成り，出力された VHDL 記述の量は，合計 728 行であった．

5 まとめ

基本的なパイプライン制御部に対するテストプログラム自動生成および動作記述自動生成を実現した．テストプログラムはハザードを起こす命令パターンを網羅するものであり，動作記述は任意の命令列に対するインターロック機構の動作を実現するものである．

今後，我々は SPARC などの現実のプロセッサに対してより広く実用可能な設計／検証支援を目指し，対象モデルの拡大を進めていく方針である．

参考文献

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 1990.

[2] D. C. Lee and D. P. Siewiorek, "Functional Test Generation for Pipelined Computer Implementations," *FTCS21*, pp. 60–67, 1991.

[3] H. Iwashita, T. Nakata, and F. Hirose, "Behavioral Design and Test Assistance for Pipelined Processors," To appear in *Asian Test Symposium '92*, 1992.