C-02

フォワーディングを考慮した命令依存距離の抽出
Extraction of Instruction Latency Considering Forwarding

平岡 佑介† 石浦 菜岐佐† 今井 正治††
Yusuke Hiraoka Nagisa Ishiura Masaharu Imai

1 はじめに

コンパイラのパイプラインスケジューリングにおいて, 2 つの命令間にデータ依存関係がある場合には, これらを何サイクルか離さなければならない. 本稿ではこのサイクル数を命令依存距離と呼ぶ. 命令依存距離は, コンパイラの命令スケジューリングだけでなく, サイクル精度の命令レベルシミュレーションを正確に行うためにも必要となる. ISDL [1], Trimaran [2] では命令全ての組み合わせに対して命令依存距離を人手で記述する必要がある. EXPRESSION [3], LISA [4] ではプロセッサ記述から命令依存距離の抽出を行っているが, この際フォワーディングは考慮していない. これに対し本研究では, フォワーディングまで考慮した命令依存距離の抽出手法を提案する. 本稿では, ASIP 開発環境 ASIP-Meister [5] のプロセッサ記述 (サイクル精度の動作記述) 中のフォワーディングを分析し, 正確な命令依存距離を求める手法を示す.

2 命令依存距離

本研究では, 複数の命令スロットを持ち, 最大でスロット数だけの命令を一度に発行できる VLIW 型プロセッサを対象とする. 各命令に対し, その命令が発行されるスロットは一意に決まっているものとし, 各命令はレジスタに対する複数の read と write が行える. 各 read と write は, それぞれ何サイクル目でどのレジスタにアクセスするかの情報を持つ. 以下にその表記を列挙する.

I: 命令の集合
$i \in I$ に対し,
- $i.slot$: 命令 i が使用するスロット
- $i.read$: 命令 i の read の集合
- $i.write$: 命令 i の write の集合

$r \in i.read$ に対し,
- $r.reg$: r が read するレジスタ
- $r.cycle$: r が レジスタを read するサイクル

$w \in i.write$ に対し,
- $w.reg$: w が write するレジスタ
- $w.cycle$: w がレジスタへ write するサイクル

このとき, $i, j \in I$, $w \in i.write, r \in j.read$ に対し, 命令 i と j の w と r に関する命令依存距離 $d_{w,r}(i,j)$ は次式で与えられる.

$$d_{w,r}(i,j) = w.cycle - r.cycle + 1.$$

†関西学院大学 理工学部 情報科学科
††大阪大学 大学院情報科学研究科 情報システム工学専攻

```
micro_operation ADD on RG02ALU
{
  wire [31:0] source0;
  wire [31:0] source1;
  wire [31:0] result;
  stage 2 {
    wire [31:0] tmp0;
    wire [31:0] tmp1;
    tmp0 = GPR.read2(rs0);
    tmp1 = GPR.read3(rs1);
    source0 = FWU2.forward(rs0,tmp0);
    source1 = FWU3.forward(rs1,tmp1);
  };
  stage 3 {
    wire [3:0] flag;
    <result, flag> = ALU1.add(source0,source1);
    null = FWU0.forward2(rd,result);
    null = FWU1.forward2(rd,result);
    null = FWU2.forward2(rd,result);
    null = FWU3.forward2(rd,result);
  };
  stage 4 {
    null = FWU0.forward4(rd,result);
    null = FWU1.forward4(rd,result);
    null = FWU2.forward4(rd,result);
    null = FWU3.forward4(rd,result);
    null = GPR.write1(rd,result);
  };
};
```

図 1: フォワーディングを含む命令の動作記述

3 フォワーディングを考慮した命令依存距離

3.1 フォワーディングの記述とモデル化

図 1 に, フォワーディングを考慮した命令の記述の例を示す. ここでは, ハードウェア資源グループ RG02ALU を用いる ADD 命令の動作をステージ毎に記述している. stage 2 でレジスタファイル GPR の rs0 番目, rs1 番目のデータをそれぞれ source0, source1 に読み込むが, フォワーディングがある場合には, フォワーディングユニット FWU2, FWU3 からデータを読み込む. stage 3 で ALU1.add による演算結果 result を stage 4 で GPR の rd 番目に書き込むが, 演算結果が得られる stage 3 からレジスタ書き込みが行われる stage 4 の間, FWU0, $\cdots$, FWU4 を通じて演算結果をフォワーディングしている.

フォワーディングユニット (FWU) は, read 側に接続されるポート p_{read} と, write 側に接続される複数のポート $p_1, \cdots, p_n$ を持ち, write 側に read 側と同じレジスタの値が 1 つでもフォワードされれば, それを p_{read} に出力する. 複数のポートに同じレジスタの値がフォワードされるときは, $p_1, p_2, \cdots, p_n$ の順に優先される.

$i \in I$, $r \in i.read$, $w \in i.write$ なる r, w に対して, フォワーディングは次のように定式化できる.

$r.fwu$: r が read する FWU
$r.fwcycle$: r が FWU を read するサイクル
$w.fw$: w のフォワーディングの集合

$f \in w.fw$ に対し,

$f.fwu$: f が write する FWU
$f.port$: f が write する FWU のポート
$f.cycle$: f がフォワーディングするサイクル

FWU の write 側のポート p_m に対し, $F(p_m)$ を次のように定義する.

$$F(p_m) = \{(w,k) \mid w \in \bigcup_{i \in I} i.write \text{ s.t. } f \in w.fw \text{ s.t. } f.port = p_m, k = f.cycle\}.$$

例えば図 2 において,

$$F(p_3) = \{(w_1, 4), (w_2, 4)\}$$

であるが, これは w_1, w_2 が共に 4 サイクル目でポート p_3 にフォワーディングを行うことを表す.

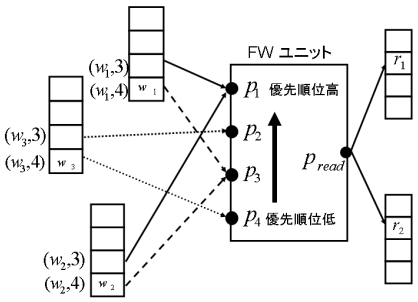

図 2: 複数の命令からのフォワーディング

3.2 正しいフォワーディングの条件と命令依存距離

以上の定式化を行うと, 命令依存距離の抽出に先立ち, フォワーディングの設計, 記述ミスの検証が行える. フォワーディングが正しく行われるためには, 次の条件が成立しなければならない.

1. フォワーディングが valid
$$(w,k) \in F(p_i), (w,l) \in F(p_j), l > k \Rightarrow i > j.$$
これはフォワーディングの優先順位が正しく守られることを意味する.
2. フォワーディングが conflict free
$F(p_m) = \{(w_1,k_1), (w_2,k_2), \cdots, (w_t,k_t)\}$ である p_m に対し, $w_r \in i_r.write$ $(r = 1, 2, \cdots, t)$ とするとき
$$i_1.slot = i_2.slot = \cdots = l_t.slot,$$
$$k_1 = k_2 = \cdots = k_t.$$
これは p_m の要素が, 同じサイクルで同じポートにフォワーディングを行わないことを意味する.
3. フォワーディングが regular
$k < w.cycle$ なる k に対し,
$$(w,k) \in F(p_i) \Rightarrow \exists p_j[(w,k+1) \in F(p_j)].$$
これはフォワーディングに “飛び” がないことを意味する.

この 3 条件が成立するとき, フォワーディングを考慮した命令依存距離 $d_{w,r}(i,j)$ は, 次のように定義できる. $i \in I, w \in i.write$ なる w が FWU u に行うフォワーディングの中で, そのサイクルの最小値 $w.fwcycle$ を
$$w.fwcycle(u) = \min_{f \in w.fw \text{ s.t. } f.fwu=u} f.cycle,$$
とすると,
$$d_{w,r}(i,j) = \begin{cases} w.fwcycle(r.fwu) - r.fwcycle, \\ \quad (\exists f \in w.fw[f.fwu = r.fwu] \text{ のとき}) \\ w.cycle - r.cycle + 1. \\ \quad (\text{それ以外}) \end{cases}$$

4 まとめと今後の課題

本稿では, フォワーディングのモデル化から命令依存距離の抽出法を提案した. 本手法を用いれば, 動作記述から接続関係を抽出し, フォワーディングの設計検証を行った上で命令依存距離を抽出できる.

命令の数が多くなるにつれ命令依存距離のデータ量は膨大になるが, その削減が今後の課題である. また, WAR (write after read), WAW (write after write) ハザードの対応等も課題として挙げられる.

謝辞

本研究に際し, ご助言・ご討論頂きました大阪大学の武内良典助教授, 今井研究室の関係諸氏, および関西学院大学 石浦研究室の諸氏に感謝致します.

参考文献

[1] G. Hadjiyiannis, S. Hanono, and S. Devadas: “ISDL: An Instruction Set Description Language for Retargetability,” in *Proc. DAC*, pp. 299–302, June 1997.

[2] http://www.trimaran.com.

[3] P. Grun, A. Halambi, Nikil D. Dutt, and A. Nicolau: “RTGEN: An Algorithm for Automatic Generation of Reservation Tables from Architectural Descriptions,” in *Proc. ISSS 1999*, pp. 44–50, Nov. 1999.

[4] O. Wahlen, M. Hohenauer, G. Braun, R. Leupers, G. Ascheid, H. Meyr, and X. Nie: “Extraction of Efficient Instruction Schedulers from Cycle-True Processor Models,” in *Proc. SCOPES 2003*, pp. 167–181, Sept. 2003.

[5] K. Okuda, S. Kobayashi, Y. Takeuchi, and M. Imai: “A Simulator Generator Based on Configurable VLIW Model Considering Synthesizable HW Description and SW Tools Generation,” in *Proc. SASIMI 2003*, pp. 152–159, Apr. 2003.