専用プロセッサ設計支援システム（ＳＹＡＲＤＳ）における中間情報の最適化

9M−2

上田 穣　雨坪 孝尚　白井 克彦

早稲田大学 理工学部

1　はじめに

現在、専用VLSIプロセッサの研究・開発はますます多様化してきているが、その設計についてはハードウェアに関する専門的知識がかなり要求される。しかしながら、より高速な計算や処理を実行するという目的から特定のアルゴリズムをハードウェア化したいという要望は一層高まってきており、非専門家であっても容易に設計を行うことの出来る支援システムの登場が熱望されている。

以上述べたようなことを背景として、我々は、汎用性の高い高級言語Pascalをアルゴリズム入力言語として用いた、ハードウェア知識を持たないユーザにも使用可能な専用プロセッサ設計支援システム（ＳＹＡＲＤＳ）の構築を目指している。このシステムは、仕様記述により入力されたアルゴリズムに対し、実行可能なアーキテクチャについてシミュレーションによる解析を行いながら最適化されたプロセッサの設計支援を行うものである。

そこで、本報告では、ＳＹＡＲＤＳの中間言語生成系に焦点を絞り、現在のシステムによって生成される中間情報を解析し、その制御フローを改善することによって、より効率のよい中間情報を再生成するシステムの構築を目指す。これによって生成されるより洗練された中間情報を解析系に供給することにより、それ以降での処理における負担を軽減し、それによってターンアラウンド・タイムの短縮、さらに設計されるＬＳＩの性能の向上をはかろうとするものである。

2　ＳＹＡＲＤＳ全体の概要

ＳＹＡＲＤＳは、大きく分けて中間情報生成系、解析系、合成系の3つの系から構成されている。

中間情報生成系では、Pascalによる仕様記述に対して字句・構文解析、不要なコードを削除するなど様々な解析を実行し、中間情報として3番地文によるフロー情報の作成を行う。

解析系においては、サンプルデータに基づいてシミュレーションを行い、さらにブロック・命令ごとの頻度解析を行って、機能ブロックとして合成できる部分をグラフ構造として取り出し、ユーザに結果を出力する。そこでユーザはこの解析結果から命令の選択を行う。

合成系では、ユーザより、与えられた機能ブロック・命令・レジスタの諸制限の中で割付を行い、システムの出力として、使用した命令・機能ブロック・レジスタの情報とそのフロー情報をハードウェア情報として出力する。

ユーザは、解析系での結果と合成系より得られたコードの再度のシミュレーションにより、ハードウェアの合成を行う。

3　中間情報の改善

ＳＹＡＲＤＳの中間情報生成系では、高位の仕様記述（Pascal）に対して字句・構文解析、不要なコードを削除するなど様々な解析を実行した後に、その出力として中間情報が生成される。これには入出力変数宣言部、変数宣言部、プロセス記述部、変数に関する情報などが含まれるが、ここでは3番地文によるコード（プロセス記述部）の改良に取り組む。

本報告では、特に不用なＧＯＴＯ文の削除とＩＦ文の移動によるプロセス記述部全体のブロック数の削減の2点について述べる。例として、バブルソートの中間情報を用い、上記の2つの改良をした結果を示す。

(1) 不用なＧＯＴＯ文の削除

(a) ＧＯＴＯ文のみから成るブロックの削除

例1に見られるように、1組のＩＦ文ブロックとＧＯＴＯ文ブロックによる無駄なブロック分割を改善する。

例1

```
(B0014                          (B0014
  :                               :
 (36 (IF J = 0 GOTO B0016)))     (36 (IF J <> 0 GOTO B0005)))
(B0015                          (B0016
 (37 (GOTO B0005)))               :
(B0016
  :
```

改良前　　改良後

(b) 次のブロックを指すＧＯＴＯ文の削除

3番地文はＩＦ文、ＧＯＴＯ文によるブロック番号の指定がない限り、次のブロックを逐次的に実行するため、次のブロックを指定するＧＯＴＯ文は不要となるのでこれを削除する。（例2）

例2

```
(B0013                          (B0013
  :                               :
  :                               :
 (33 (A (I) := TEMP))            (33 (A (I) := TEMP)))
 (34 (GOTO B0014)))              (B0014
(B0014                            :
  :
```

改良前　　改良後

Optimization of Intermediate Expression in Special-Purpose Processor Design System(SYARDS)

Yutaka UEDA, Takanao AMATSUBO, Katsuhiko SHIRAI

Department of Electrical Engineering, Waseda University

(2) 条件文（ＩＦ文）の移動によるプロセス記述部全体のブロック数の削減

条件文のみから構成されるブロックに注目して例３、例４のようなＩＦ文の操作を行う。この操作によってステップ数の削減は見込めないが、ブロック数は削減される。このようなブロック数の削減は、解析系における同一ブロック内での並列化に対して有効となる。ただし、この操作は次の点を考慮して実行する必要がある。本システムにおいて基本ブロックに分割する際に用いられているルール、

“条件文（ＩＦ文）は基本ブロックの最後でなければならない”

は守る。その理由は条件文が基本ブロックの途中にあることによってブロック分割の意味がなくなってしまう。さらに解析系での並列化に支障をきたす、制御が容易ではなくなるなどの問題が生じるからである。

例３

```
(B0005                          (B0005
  (8 (J := 0))                    (8 (J := 0))
  (9 (I := 2)))                   (9 (I := 2))
(B0006                            (10 (IF I = HI GOTO B0011)))
  (10 (IF I = HI GOTO B0011))) (B0007
(B0007                            :
   :                              :
   :                          (B0010
(B0010                            (22 (I := I + 1))
  (22 (I := I + 1))               (23 (IF I <> HI GOTO B0007)))
  (23 (GOTO B0006)))         (B0011
(B0011                            :
   :
```

改良前　　改良後

例４

```
(B0016                          (B0016
  (38 (I := 1)))                  (38 (I := 1))
(B0017                            (39 (IF I = 100 GOTO B0019)))
  (39 (IF I = 100 GOTO B0019))) (B0018
(B0018                            :
   :                              :
   :                              (42 (IF I <> GOTO B0018)))
  (42 (GOTO B0017)))          (B0019
(B0019                            :
   :
```

改良前　　改良後

4　適用例・評価

先のバブルソートの中間情報に対し、構築したシステムによって最適化を行い、中間情報を再生成したものを、例として１００個の数字のデータを与えてシミュレーションを行なった場合の各ブロックの実行回数及び各ステップの実行回数とともに図１に示す。再生成された中間情報の全ブロック数は１３、総ステップ数は３８となる。そして、改良の効果を表すために、先に与えたものと同一のデータを与えてシミュレーションを行なった場合の新旧２つの中間情報をステップの実行回数、ブロックの実行回数、コード上のブロック数及びステップ数という４つの観点から数字として比較した表を表１に示す。この例から上述の中間情報の改良によりブロック・ステップの実行回数の削減が実現されていることがわかる。

```
(PROGRAM BUBBLESORT NIL (IPORT OPORT)
  変数宣言部 略
  ;;                                  各ブロック   各ステップ
  (PROCESS                            の実行回数   の実行回数
    (B0001                                  1
      (1 (I := 1))                                       1
      (2 (IF I = 100 GOTO B0003)))                       1
    (B0002                                 99
      (3 (A (I) := IPORT))                               99
      (4 (I := I + 1))                                   99
      (5 (IF I <> 100 GOTO B0002)))                      99
    (B0003                                  1
      (6 (A (I) := IPORT))                               1
      (7 (HI := 100)))                                   1
    (B0004                                 83
      (8 (J := 0))                                       83
      (9 (I := 2))                                       83
      (10 (IF I = HI GOTO B0008)))                       83
    (B0005                              4,644
      (11 (T0002 := I - 1))                          4,644
      (12 (T0003 := A (T0002)))                      4,644
      (13 (T0004 := A (I)))                          4,644
      (14 (IF T0003 <= T0004 GOTO B0007)))           4,644
    (B0006                              2,526
      (15 (J := I - 1))                              2,526
      (16 (TEMP := A (J)))                           2,526
      (17 (T0007 := A (I)))                          2,526
      (18 (A (J) := T0007))                          2,526
      (19 (A (I) := TEMP)))                          2,526
    (B0007                              4,644
      (20 (I := I + 1))                              4,644
      (21 (IF I <> HI GOTO B0005)))                  4,644
    (B0008                                 83
      (22 (T0008 := I - 1))                              83
      (23 (T0009 := A (T0008)))                          83
      (24 (T0010 := A (I)))                              83
      (25 (IF T0009 <= T0010 GOTO B0010)))               83
    (B0009                                 79
      (26 (J := I - 1))                                  79
      (27 (TEMP := A (J)))                               79
      (28 (T0013 := A (I)))                              79
      (29 (A (J) := T0013))                              79
      (30 (A (I) := TEMP)))                              79
    (B0010                                 83
      (31 (HI := J))                                     83
      (32 (IF J <> 0 GOTO B0004)))                       83
    (B0011                                  1
      (33 (I := 1))                                      1
      (34 (IF I = 100 GOTO B0013)))                      1
    (B0012                                 99
      (35 (OPORT := A (I)))                              99
      (36 (I := I + 1))                                  99
      (37 (IF I <> 100 GOTO B0012)))                     99
    (B0013                                  1
      (38 (OPORT := A (I)))))                            1
  ;;                                  ----------   ----------
  変数情報部 略                        12,344       42,237
```

図１：再生成された中間情報（バブルソート）

表 1: 中間情報の比較（バブルソート）

	改良前の中間情報	操作(1)のみを実行した中間情報	(1)(2)とも実行した中間情報
ステップ数	43	38	38
ブロック数	19	16	13
ステップ実行回数	51,888	47,079	42,237
ブロック実行回数	19,475	17,271	12,344

5　むすび

ＣＡＤシステムＳＹＡＲＤＳにおいて、高位の仕様記述から得られる中間情報の改善とそれによる効果について報告した。今後の展開として入力されたアルゴリズムに対して、より的確な最適化を施してさらに洗練された中間情報を生成し下位システムに供給することを検討している。

参考文献

[1] A.V. エイホ・J.D. ウルマン著、土居　範久訳、『コンパイラ』、培風館