高級言語からの機能合成一手法

9N−5

矢野隆則
(株) リコー　中央研究所

1.まえがき

大規模回路の設計は、ハードウェア記述言語ＨＤＬを使ったトップダウン設計が導入されつつある。

筆者らは高級言語によるアルゴリズムレベルの動作記述から、レジスタトランスファーレベルの回路のＶＨＤＬコードを自動合成する機能合成システムを開発している。

その中で本文は特に基本ブロックの範囲を越えた演算処理の並列性を抽出する大局的なスケジューリング方式に付いて報告する。

2.機能合成システムの概要

開発した機能合成システムの処理フローは、以下のようになっている。

<1> 構文・字句解析処理
<2> 最適化処理
　<2-1> スケジューリング処理
　<2-2> アロケーション処理
<3> レジスタトランスファーレベルの
　ＶＨＤＬコードの合成
<4> 論理合成以下の合成処理

図１に示したようなＣ言語（限定版）で動作記述したものを入力すると、図３のような構成のレジスタトランスファーレベルのＶＨＤＬコードを自動合成する。図２に図１の動作記述の入力に対する、ＶＨＤＬコードの一部の出力例を示す。

```
main()
{
Y1 = I1 + I2; /*op1*/
Y2 = Y1 + I3; /*op2*/
Y3 = I4 + 5; /*op3*/
if(Y3 < Y2) /*op4*/
    Y5 = 6 + Y2; /*op5*/
else
    Y5 = Y2 + I5; /*op6*/
Z11 = Y5 + I6; /*op7*/
Z22 = Y1 + 9; /*op8*/
Z33 = Y2 + I7; /*op9*/
Z44 = Y1 + 4; /*op10*/
}
```

図１．動作記述例

```
architecture B of CNTL is
    :
process
    :
  case STATE is
    :
  when ST3 =>
    if(SIG = '1') then
      STATE <= ST4;
    else
      STATE <= ST5;
  when ST4 =>
    STATE <= ST6;
    :
```

図２．VHDLコード出力例

A High Level Synthesis Method using a Programming Language
Takanori Yano
RICOH COMPANY,LTD.

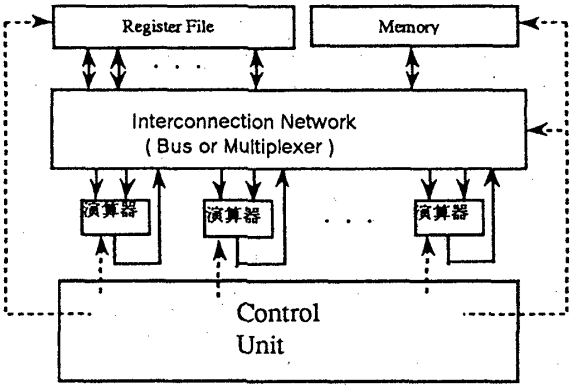

図３.合成される回路のブロック図例

3.大局的なスケジューリング処理方式

提案するスケジューリング方式は、基本ブロックの範囲を越えて演算の並列性を得る方式である。

スケジューリング処理の中では、与えられた動作記述を分析し演算処理の依存関係から演算の並列処理の可能性を抽出した後、その並列処理の可能性を条件として実際に並列に演算処理する部分を決めている。

条件分岐の存在にかかわらず演算処理の並列性を抽出する為に、演算処理の依存関係が明確である部分(分岐部分を含まない範囲）に分割し、その各々の部分で依存関係に基づき並列性を抽出した後に併合するという方式で実現している。

スケジューリング処理は以下のようになっている。

(1). 演算処理の依存関係等初期データの準備
(2). ブロック群への分割
(3). ブロック群毎にそれに属する演算処理のクリティカルパス及び並列処理可能性の抽出
　(3-1). オペレーター依存関係の抽出
　(3-2). ASAP(As Soon As Possible)スケジューリング
　(3-3). ALAP(As Late As Possible)スケジューリング
　(3-4). クリティカルパスの抽出
　(3-5). 各演算処理の並列処理可能性の抽出
(4). 全体レベルでの演算処理のクリティカルパスの抽出
(5). 全体レベルでの演算処理の並列処理可能性の抽出
(6). 各演算処理の実行順位の決定

並列実行可能性はその演算がクリティカルパス上の演算処理のどれと並列処理可能かという形式で抽出される。従って合成される回路のコントロールステップ数はクリティカルパス上の演算処理に必要なステップ数と同じである。

ある演算処理の並列処理可能性は、その演算処理が元々記述されていたブロックに依存する為、それを条

件にして抽出している。例えば、ブロックCにある演算10はブロックB1とブロックB2の両方にある演算で並列実行可能である必要がある。あるブロックの演算がどのブロックにある演算と並列処理可能であるかという条件は予め制御フローを調べ抽出している。

図4に示すデータフローは図1の動作記述の演算処理の依存関係を示す。クリティカルパス上の演算処理は演算1, 2, 5, 6, 7である。

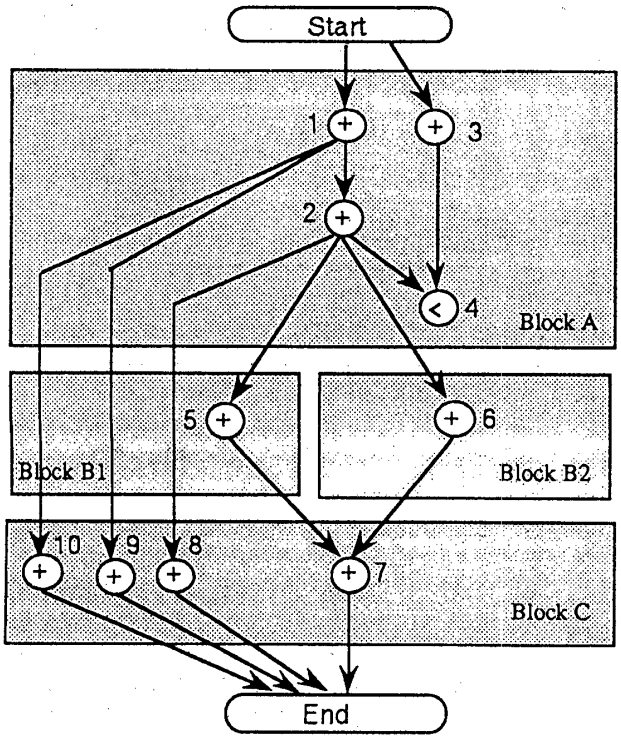

図4. データーフロー

この例では、ブロックCはブロックA、ブロックB1とB2の両方と可能であるということを予め求めておき、その条件を満たすように並列性を抽出する。例えば、Z44 = Y1 + 4; に対応する演算10の並列処理可能性は、(3)の処理段階で、ブロック群A, B1, Cの処理で演算2, 演算5, 演算7と並列処理可能であり、ブロック群A, B2, Cの処理で演算2, 演算6, 演算7と並列処理可能であることが求められる。(5)の処理段階の併合の際には、所属ブロックの条件を満たす演算2, 演算5と演算6, 演算7を演算10と並列処理可能であるとする。

最後の(6)では、並列処理可能性に基づき必要な演算器を少なくするような優先配置ルールに従って順次各演算処理の実行順位を確定している。

4. スケジューリング結果の検討

図1の動作記述に対する本スケジューリング処理の結果を図5に示す。図中[ ]の中に同時に処理される演算を列挙している。

演算処理 Z44 = Y1 + 4; のように、元々ひとつのブロックで演算処理するように記述されていた演算が、分岐先の両方で実行される様にスケジューリングされる例である。

```
main()
{
    [Y1 = I1 + I2;Y3 = I4 + 5;] /* 基本ブロックA */
    [Y2 = Y1 + I3;Z22 = Y1 + 9;] /* 基本ブロックA */
    if(Y3 < Y2)
        [Y5 = 6 + Y2;Z44 = Y1 + 4;] /* 基本ブロックB1 */
    else
        [Y5 = Y2 + I5;Z44 = Y1 + 4;] /* 基本ブロックB2 */
    [Z11 = Y5 + I6;Z33 = Y2 + I7;] /* 基本ブロックC */
}
```

図5. 本方式のスケジューリング処理結果

この様に本方式によると、実行時の分岐条件にかかわらず高いレベルの並列度が得られ、演算器の共有利用度を高めることができる。

図1の動作記述を与え、スケジューリング処理を行った場合以下の様に合成される。

（a）. 本スケジューリング処理を行った場合
　演算器2個　マルチ・デマルチプレクサ6個
　レジスタ／ラッチ類19個
　コントロールステップ総数6ステップ
（b）. 従来のスケジューリング処理を行った場合
　演算器2個　マルチ・デマルチプレクサ6個
　レジスタ／ラッチ類19個
　コントロールステップ総数7ステップ
（c）. スケジューリング処理を行わない場合
　演算器1個　マルチ・デマルチプレクサ3個
　レジスタ／ラッチ類19個
　コントロールステップ総数10ステップ

(b)のスケジューリングは、従来一般的に行われている方式で、基本ブロック毎に独立に並列性の抽出を行う処理方式である。

本方式で合成される回路は、コントロールステップが動作記述で示された演算処理のクリティカルパスのステップに一致するようにスケジューリングしているのでコントロールステップ総数が一番少ない。

5. 結論

提案した大局的なスケジューリング方式を用いたＶＨＤＬコードを自動合成する機能合成システムのプロトタイプシステムを開発した。面積・スピードの性能面で優れた回路を合成する見通しが得られた。

現在、複雑な動作記述（プログラム）が与えられた場合の、処理の有効性の検証および処理スピードの改善を進め、実用化に向けて改良デバッグを行っている。

参考文献

[1] Flamel:A High-Level HardwareCompiler; HowardTrickey; IEEE Trans. on CAD, Vol.CAD-6, No.2, March 1987
[2] HAL: A Multi-Paradigm Approach to Automatic Data Path Synthesis; Carleton Univ.; P.G. Paulin ,J.P. Knight, E.F.Girczyc; Proc. 23rd DAC, pp.503-509, 1986
[3] Forced-Directed Scheduling for the Behavioral Synthesis of ASIC's; Paulin P.G. Knight J.P; IEEE Trans. CAD IC&System,8, 6, pp.661-679, June 1989