Varchsyn(2): VHDLからの合成

6N−2

河原林 政道[*1]　浅香 俊治[*1]　水嶋 紀子[*1]　佐藤 克也[*2]　前田 直孝[*1]

[*1]NEC　[*2]NECアイシーマイコンシステム（株）

1　はじめに

デジタルシステムのハードウェア設計においてトップダウン設計は最も重要な役割を担っている。その中でも論理合成は中核となる技術で、近年様々なハードウェア記述言語からの論理合成システムが発表されている。1987年にIEEEで標準化されたハードウェア記述言語VHDLは、全世界的な標準言語になりつつある。論理合成システムVarchsynはVHDLを入力し合成することができる。本稿では、論理合成システムVarchsynにおけるVDHLからの合成技術について報告する。

2　概要

VHDLは動作レベル、RT（レジスタトランスファ）レベル、構造レベルの記述ができる。論理合成用に定めたVHDLのサブセット（RTレベル、構造レベル）に従って記述したハードウェア記述をVarchsynに入力する。VHDLで記述したハードウェアを、組合せ回路部分と記憶素子部分に分類した後、最適化し、ネットリストを生成する。このとき、単純に組合せ回路と記憶素子を分類するだけではなく、VHDLに記述した構造情報を有効に利用して品質の高い回路を合成する。

3　処理フロー

VHDL入力の処理の流れを図1を使って説明する。

『構文意味解析』部において、VHDLシミュレータで十分検証したRTもしくは構造レベルのVHDL記述に対して、構文、意味解析を行ない中間形式ファイルを生成する。次に、『回路データ生成』部において、構造レベルで記述した階層設計の下位階層やテクノロジーライブラリのブロックは、特別な処理を施さず、そのまま内部データ構造に変換する。RTレベルのVHDLに対して、大きく分けて5段階の処理を行なう。各処理段階について説明する。

3.1　演算器共有

同時に実行されない加減算などの機能オペレータ／サブプログラムを共通化する[1]。

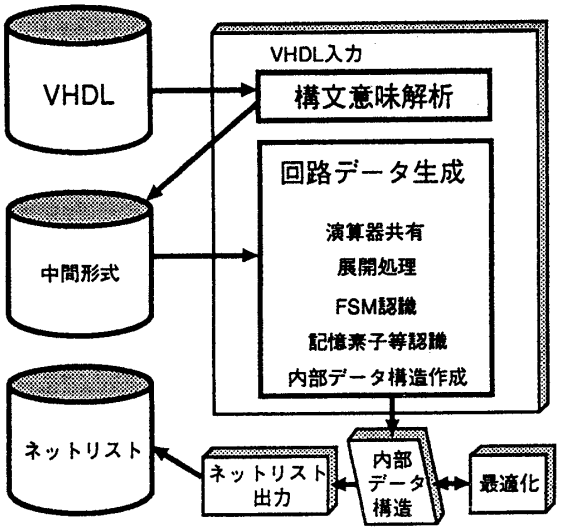

図1. VarchsynにおけるVHDL入力処理フロー

3.2　展開処理

ここでは、ループの展開、オペレータ／サブプログラムの展開を行なう。

ループの展開では、ループ中の記述を繰り返し回数分展開する。繰り返し回数が静的に決定できる場合のみ可能である。

オペレータ／サブプログラム（関数、プロシージャ）の展開では、VHDLのパッケージに記述した定義に従ってVHDLの中間形式に展開する。このオペレータ／サブプログラムの定義は、Varchsynは加算、減算などの算術オペレータ、比較オペレータなどの回路構成を定義したパッケージを提供しており、改めてこれらを定義するパッケージを記述しなくてもオペレータの合成は可能である。勿論、所望の機能を定義して使用してもよい。

オペレータ／サブプログラムを展開せず、最適化時に面積、遅延を考慮しながらハードマクロ（例えば、4ビット加算器など）にマッピングすることも可能である[2]。

3.3　FSM（有限状態機械）の認識

Varchsynは状態遷移記述言語だけでなく、FSM記述規則に従ったVHDL記述を入力、合成可能である。主なFSM記述規則は以下の通り。

1. 1種類の列挙型信号を状態変数として利用
2. シーケンシャル文で記述
3. 入力信号の全ての組合せ条件で状態変数への代入文（即ち状態遷移文）

Varchsyn(2): A Synthesis Technique from VHDL Descriptions
Masamichi KAWARABAYASHI[*1], Toshiharu ASAKA[*1], Noriko MIZUSHIMA[*1], Katsuya SATO[*2], Naotaka MAEDA[*1]
[*1]NEC Corporation, [*2]NEC IC Microcomputer Systems Ltd.

4. クロック信号のエッジ（立ち上がり／立ち下がり）で状態変化、出力値確定
5. 非同期、同期リセット条件が記述可能

Varchsyn は上記規則に従った VHDL 記述を FSM を認識した後、あらかじめ指定した通りに状態コードを割り当てるか、自動的にシミュレーテッドアニーリング法などを使って最適状態コード割り当てを行なうことができる。

3.4 記憶素子等認識

記憶素子と組合せ回路の認識

クロックのエッジ記述の条件で値が変化する信号はエッジトリガータイプの DFF（図 2）、エッジでなくレベルで値が変化する信号はレベルセンシティブタイプのラッチを生成する。その他の部分（トライステートゲートを除く）は全て組合せ回路になる。

「エッジ記述」は信号の立ち上がり／立ち下がりを表現するために利用する。wait 文もしくは if 文の条件式に、信号が変化したことを表すアトリビュート（event もしくは stable と not の組合せ）と、その信号の論理値が '0' もしくは '1' であるかを比較する条件の論理和（and ）で表現する。

```
if (RESET = '1') then
   FF <= '0';
elsif (CLK'event
        and CLK='1') then
   FF <= DATA;
end if ;
```

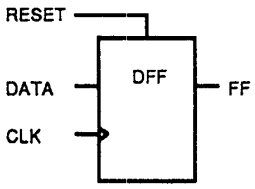

図 2. DFF 記述と回路例

マルチプレクサ認識

if 文、case 文などの条件文から作成する組合せ回路をマルチプレクサ（図 3）と認識する。一般に、組合せ回路からマルチプレクサを自動的に抽出することは困難であるが、ユーザが記述した構造が明らかな VHDL 記述からは容易にマルチプレクサを抽出できる。テクノロジーライブラリのブロックへのマッピングは、最適化段階（後処理）で行なわれ、適当なマルチプレクサブロックが存在しなければ、組合せ回路に変換する。

```
case CTL(1 downto 0) is
   when "00"    =>
        A <= B and  C;
   when "01"    =>
        A <= B or  C;
   when others =>
        A <= B xor  C;
end case ;
```

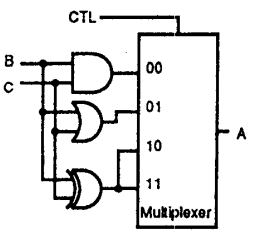

図 3. マルチプレクサ記述と回路例

トライステート認識

信号の型は標準パッケージで定義した論理型（BIT, BIT_VECTOR）に加えて、IEEE で標準化された 9 値の論理値（std_logic_1164）を使用可能である。Varchsyn は 9 値を論理合成で取り扱える論理値である 3 値（'0', '1', 'Z'）に読みかえている。双方向バスなどに出力する信号に対して、特定の条件でハイインピーダンス 'Z' を出力する信号はトライステートゲート（図 4）を生成する。

```
if (CNTL = '1') then
   BUS <= DATA;
else
   BUS <= 'Z';
end if ;
```

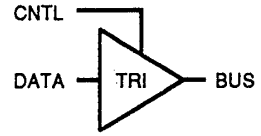

図 4. トライステート記述と回路例

3.5 内部データ構造作成

上記処理を施した後、論理／タイミング最適化しやすい内部データ構造に変換する。内部データ構造に変換した回路データを最適化した後にネットリストとして出力する。

4 論理合成用サブセット

Varchsyn では、論理合成対象となる VHDL のサブセットを規定している。利用できる代表的な項目を以下に列挙する。

- パッケージ
- integer, bit, bit_vector, std_logic (IEEE 1164), enumerate タイプ
- **, abs 以外のオペレータ
- サブプログラム
- シーケンシャル文 (wait, if, case, loop, 代入文など)
- コンカレント文 (process, block, generate など)

ハードウエアに直接マッピング困難な記述（信号代入文の waveform、繰り返し回数が動的に決定するループなど）は使用できない。また検証用の記述（assert など）は、無視する。

5 おわりに

論理合成システム Varchsyn の入力言語の 1 つである VHDL からの合成について報告した。サブセットの範囲内で記述した VHDL の構造情報を有効に利用して品質の高い回路を合成することができた。

今後、論理合成用 VHDL サブセットの制限緩和を進め、1992 年版の VHDL をサポートしてゆく予定である。

参考文献

[1] 伊藤 他. Varchsyn(3): VHDL 記述からの演算シェアリング. 第 46 回情処全大, 1993.

[2] 若林 他. Varchsyn(4): 演算器合成手法. 第 46 回情処全大, 1993.