2K−8

論理合成システム SERAPHIM テンプレート化合成手法*

遠藤 真† 高原 厚‡
NTT LSI 研究所§

1 はじめに

LSIの論理設計工数を削減するために、論理合成システムが広く使われてきている。論理設計にRTL言語を用いる理由は、記述の抽象度を上げることにより設計効率を上げるためのほか、製造テクノロジー独立にすることによって記述の再利用を図ることにある。しかしながら、実際の設計では、論理合成システムを用いて所望の回路構成を得られない部分がある場合、例えばデータ転送回路、演算回路、状態遷移回路、テスト用回路等の記述中で、使用ライブラリ中のレジスタのセルやマクロセルを陽に指定して結線するなど、設計者が介入してテクノロジーや回路構成法に依存した記述を用いている。LSIリメーク時の効率化、既存回路再利用促進のためには、特定のASICベンダ、ライブラリ、回路構成法に依存しない記述が望ましいが、現実には市販合成システムの制約から困難である。

論理合成システムSERAPHIMの合成手法は、合成過程にテンプレートを導入し、これを選択、変更することにより、種々の設計手法に応じようとするものである。

本論文では、テンプレート化にあたって検討した合成過程のモデル化と、可読性の高いテンプレート記述形式について報告する。従来合成システムに組み込まれていた合成過程を、テンプレートによってオープン化することにより、柔軟性、拡張性の高いシステムとしている。設計者は、必要に応じて自らテンプレートを記述することにより、所望の回路を合成することができる。

2 合成過程のモデル化

RTL記述からの論理合成は、言語に表現される部品と動作のモデルを、セルやマクロの実部品のネットワークにマッピングする過程として捉えることができる。RTL記述と合成回路の対応を明確にするため、まず単純なRTL動作文(operational statetment)と部品(レジスタ等)をモデル化する。

本モデル化はRTL言語の種別に依存しないが、合成向き言語UDL/I [1]を参考にしている。

次に、これらのマッピング過程をモデル化することにする。

2.1 部品

部品には、動作部品(ファシリティ)として、レジスタ・LATCH・RAM・ROM・ターミナル・定数、構造部品として、モジュール・モジュールピン・サブモジュール・サブモジュールピンがある。

*Logic Synthesis System SERAPHIM
Synthesis Procedure Using Templates

†Makoto Endo
‡Atsushi Takahara
§NTT LSI Laboratories

2.2 RTL動作文

RTL動作文は、合成可能で単純なRTL動作を定義したもので、クロック付き条件代入文や関数参照文等、15の文がある。UDL/I記述は全てRTL動作文に変換可能である。

2.3 マッピング過程

本合成手法で分類したマッピングを挙げる。

テクノロジー独立マッピング　テクノロジー依存マッピング

- ファシリティマッピング
- メモリマッピング
- FSMマッピング
- 定数マッピング
- 関数マッピング
- 組合せ回路マッピング
- RTL動作文マッピング
- 関数モジュール生成

定数マッピングは、定数ファシリティの処理を行なう。組合せ回路マッピングは、従来のテクノロジーマッピングである。

3 マッピング

3.1 ファシリティ・マッピング

ファシリティを信号の入出力ポートをもつ仮想モジュールにマッピングする。RTL言語が仮定している動作に必要なピンと、ユーザが設けたいピンをここで定義する。例えばテスト設計手法に応じて、スキャン用のピン等をこれに付加する。

3.2 FSMマッピング

FSM中のオペレーションステートメントに、状態・タスク等の起動条件、データ転送クロックを付加する。状態・タスクは、最適化、コード割り当ての後にレジスタに割り当てられる。

3.3 関数マッピング

RTL言語の用意するシステム関数及び合成用に設けた内部関数を仮想モジュールにマッピングする。内部関数には、条件信号によりデータをスイッチするPASS関数等がある。

3.4 RTL動作文マッピング

全てのRTL動作文は、単純代入文とブーリアン関数参照文、内部関数参照文(PASS関数)にマッピングされる。

3.5 関数モジュール生成

関数マッピングが仮想モジュールのピンを定義するのに対し、関数の機能を実現するモジュール内部の回路を生成する。スタンダードセル、マクロセルを用いても、ブール式でも記述できる。

このマッピングのテンプレート記述は、モジュール・コンパイラあるいはソフトマクロの記述として用いることができる。

3.6 メモリ・マッピング

仮想モジュールにマッピングされたメモリファシリティ(レジスタ,ラッチ,RAM,ROM)をライブラリ中のセルにマッピングする。リセットが指示されたレジスタは、リセットピン付きのセルにマッピングされる。

4 マッピング・テンプレート記述形式

前章で述べた合成過程の各段階を記述するマッピング・テンプレート記述形式を検討し、C++の operator overloading 機能を用いた記述言語を提案する。

特長は以下の通り。

- 言語解析が不要のため、実現および言語仕様の変更が容易。
- C++の構文（for 文等）を使えるので、拡張性が高い。
- ブーリアン関数を C++の演算子で簡潔に表せる。
- SERAPHIM　の合成系自身が C++で記述されているので、親和性が高い。

テンプレートでは基本的に、仮想モジュールのピン定義、RTL 動作文要素定義、回路を表現する。

4.1 総称名

テンプレート中では、テクノロジー依存性（ライブラリ依存性）を低減させるため、同一機能のセル及びピンに対して同一名「総称名」で記述する。

セル・ライブラリには、セル名と「総称名」を登録する。

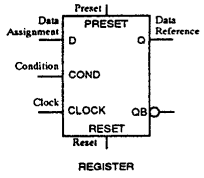
5 テンプレート記述例

以下にレジスタに対するクロック付き条件代入文(UDL/I 記述)と加算器のマッピング例を挙げる。

```
IF .cond THEN AT RISE(.clk) DO
        reg := .data ; END_DO ; END_IF;
```

5.1 ファシリティマッピング

レジスタのマッピング例を示す。




```
Pins("D",      INPUT, ASGN);
Pins("PRESET",INPUT, PRESET);
Pins("RESET", INPUT, RESET);
Pins("COND",  INPUT, COND);
Pins("CLOCK", CLOCK, CLOCK);
Pins("Q","QB",OUTPUT,REF);
```

5.2 RTL 動作文マッピング

クロック付き条件代入文のマッピング例を示す。

```
DESTINATION(CLOCK)= CLOCK();
DESTINATION(ASGN) =  PASS(SOURCE(),CONDITION());
DESTINATION(COND) = CONDITION();
```

5.3 メモリ・マッピング

レジスタを D 型 FF にマッピングする例を示す。JK 型 FF を使う等、回路構成ごとにテンプレートを用意し、切替えて使用できる。

```
SUBMODULE* reg = SUBMODULE(LIB,1,"FDRP");
SUBMODULE(reg).PIN("CLOCK") =  XPIN("CLOCK");
SUBMODULE(reg).PIN("D") =
 XPIN("D")|(~XPIN("COND")&SUBMODULE(reg).PIN("Q"));
XPIN("Q")       = SUBMODULE(reg).PIN("Q");
XPIN("QB")      = SUBMODULE(reg).PIN("QB");
SUBMODULE(reg).PIN("PRESET")    = ~XPIN("PRESET");
SUBMODULE(reg).PIN("RESET")     = ~XPIN("RESET");
```

5.4 関数モジュール生成

1 ビット全加算器(FADD)を用いて、ビット幅(N)可変のリップルキャリ加算器を生成する例を示す。

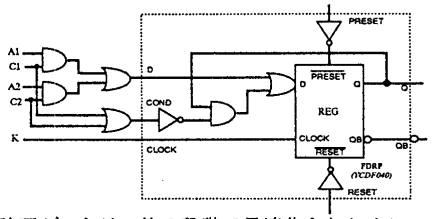
```
int N = BitWidth();
SUBMODULE* fadd = NewSUBMODULE(FUNC,N,"FADD");
SUBMODULE(fadd).PIN("IN1")  =  XPIN("IN1");
SUBMODULE(fadd).PIN("IN2")  =  XPIN("IN2");
SUBMODULE("OUT")  =  SUBMODULE(fadd).PIN("OUT");
SUBMODULE(fadd,0).PIN("CI")  =  XPIN("CI");
for ( int i=1; i<N; i++ ){
  SUBMODULE(fadd,i).PIN("CI") =
    SUBMODULE(fadd,i-1).PIN("CO");
};
```

5.5 合成例

テンプレートは、RTL 動作文ごとに局所的処理できるように記述される。複数信号の衝突は、UDL/I の OFFSTATE と同様の概念で解決される。

UDL/I 記述と上記テンプレートによる合成回路例を示す。

```
AT RISE(K) DO IF C1 THEN REG := A1 OFFSTATE 0 ;
        END_IF; END_DO;
AT RISE(K) DO IF C2 THEN REG := A2 OFFSTATE 0 ;
        END_IF; END_DO;
```



論理ゲートは、後の段階で最適化されセルにマッピングされる。

6 まとめ

多様な設計手法に対応するため、合成過程をオープンにしたテンプレート化合成手法を提案した。本手法の特長は、

- 可読性の高いテンプレート記述言語により、合成過程のメンテナンス及カスタマイズが容易
- RTL 言語の合成モデル変更に柔軟に対処可能
- 演算器等のモジュール・コンパイラ機能
- テスト設計手法に応じた回路構成を容易に組み込み可能
- 総称名の導入によるライブラリ依存性の低減

マッピング・テンプレート記述言語は拡張性のある反面、C++の operator overloading 機能により、記述性が限定される。独自の言語解析部を持つことを含め、改善の余地がある。

なお、本論文で述べた合成手法に基づいた論理合成システム SERAPHIM の基本部分は、SYNTHESISKIT として、日本電子工業振興協会 UDL/I 標準化委員会に開示する予定である。

参考文献

[1] UDL/I 標準化委員会: UDL/I 言語仕様, 第 1.0m 版 (May 1992).