1C－1　並列モジュール記述を可能とした高位記述言語による専用プロセッサ設計支援システム(SYARDS)の構築

上田 穣　吉田 裕　樋渡 仁　白井 克彦

早稲田大学 理工学部

1　はじめに

最近のCAD分野では、アーキテクチャのCodesignが注目されつつある。これはアーキテクチャのどの部分をソフトウェアで、またどの部分をハードウェアで実現するかを考察しながら、両者の利点を生かしつつ設計を進めていく手法である。

これまで当研究室で提案されてきた設計支援システム(SYARDS)もアーキテクチャ実現に必要なハードに関する情報とそれを制御するソフトを同時に生成していることから、ある意味ではCodesignを行っているとも言うことができる。

しかし、設計対象のアーキテクチャが今後さらに大規模化することが考えられることから、ユーザがある程度まとまった処理を1つのモジュールとして定義し、複数のそれぞれモジュールで構成されたアルゴリズムを記述することにより、ユーザがハードウェアであるいはソフトウェアで実現すべき部分を高位の段階である程度イメージ可能となるようなアルゴリズム入力用記述言語の仕様拡張は今後有効になってくると思われる。

そこで本稿では、上記の目的を達成するために、C言語を基礎とした並列モジュール記述が可能なアルゴリズム入力用記述言語の開発、その入力システムの構築を試みる。

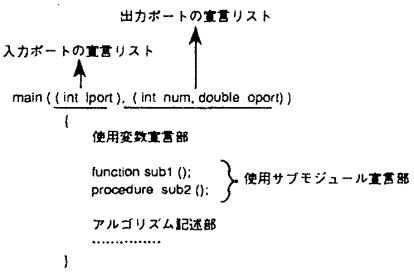
2　アルゴリズム記述言語

LSIの自動設計を行うための仕様記述用に、C言語を基礎としたハードウェア設計仕様記述言語を開発した。ここでは、その特徴的な点のみを述べる。

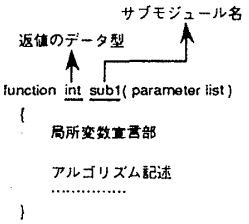
1. 変数宣言とデータ型
   変数のビット幅に関しては、ハードウェアに依存する部分であるから高位で供給するよりも、下位で供給する方がより正確なハードウェアに関する情報を的確に供給できると考えられるので、各データ型のビット幅を設定せず、初期段階での変数宣言は単に変数の属するデータ型のグループを明確にするものとしている。[1]

2. モジュールの記述
   このハードウェア設計記述言語ではメインルーチン部をメインモジュール、サブルーチン部をサブモジュールという名称で取り扱う。以下ではそれぞれのモジュールの仕様記述を述べる。

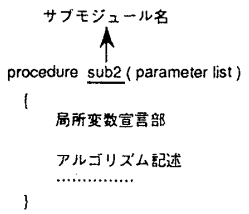
   - メインモジュール
     メインモジュールの引数部分に入力ポートとその型のリスト、及び、出力ポートとその型のリストを付加記述する。また変数宣言部の後に、使用するサブモジュールを宣言させる。



   - サブモジュール
     返値ありのサブモジュールの記述を以下のように定義する。



     返値なしのサブモジュールの記述を以下のように定義する。



Establishment of Special-Purpose Processor Design System(SYARDS) with High-level Language enable to parallel module description

Yutaka UEDA, Yutaka YOSHIDA, Jin HIWATASHI and Katsuhiko SHIRAI

Department of Electrical Engineering, Waseda University

さらに本言語ではサブモジュールの並列記述を採用している。それは次のような記述により実行される。

```
parallel
  {
    sub-module1 ( parameter1 );
    sub-module2 ( parameter2 );
  }
```

} 並列実行されるサブモジュール群

これはサブモジュールにのみ適用され、その他のステイトメントには適用されない。その理由はステイトメント単位の並列性は下位の解析系システムによって自動的に検出・調査されるからである。

ここでグローバル変数の使用は許可していない。またモジュールの並列記述部では複数のサブモジュールが同時に同一変数にアクセスすることはエラーとして処理する。

3 システムの処理概要

本研究において構築したシステムの処理概要を図1に示す。ユーザは実現しようとするアルゴリズムを先述の言語を用いて記述する。ここで外部とのデータのやりとりをする入出力ポート部に、サンプルデータを取込み、また処理したデータをシミュレーション結果として出力する関数群を使用する。こうして記述したCアルゴリズムを汎用のCコンパイラを用いてコンパイルし実行することにより、記述されたアルゴリズムが意図した処理を行っているかを確認する。ますます大規模かつ複雑になる処理アルゴリズムに対しては、中間情報にまで変換するだけであってもその過程は複雑で時間を要することが考えられる。そこでより入力言語に近いレベルでその記述内容を検証して、記述の正確な動作を確認しておくことは、その後の設計過程の負担を軽減するばかりでなく設計時間の短縮にもつながると考えられる。

次にこのプログラムにおいて、先述の入出力関数用に付加された記述を本来のポートの記述へ変換し、記述検証を行うために追加記述されたヘッダファイルの読込部分削除を行なう。ここでユーザが並列に動作させたいと考えるハードウェア・モジュールを選択して先の並列実行文を追加記述することとなる。こうして得られる仕様記述が中間情報生成系の入力となる。

中間情報生成系ではまず入力された仕様記述に対して字句解析、構文解析を行い、Lisp形式の中間表現に変換する。これを3番地文と呼ばれる中間表現にさらに変換する。そして制御フロー解析を行うとともに、並列に実行されるサブモジュールのセットを検出する。これらを中間情報として出力する。

次に生成された制御フロー情報の冗長性の改善を行う。C言語のような高級言語における曖昧性によってやむを得ず生じてしまう制御フロー情報の冗長な部分を解消することにより次段階のデータフロー解析、コード最適化での負担を軽減することが可能となる。

この後フローグラフをもとにして基本ブロック間のデータフロー解析を実行する。この解析結果として、変数のデータフロー情報が生成される。これに基づき、定数の畳み込み、ループ不変式の検出・移動など通常のソフトウェアコンパイラが実行するのと同様なコードの最適化を行なう。

以上のような過程を経て、アルゴリズム記述は中間情報に変換され、下位合成システムに供給される。

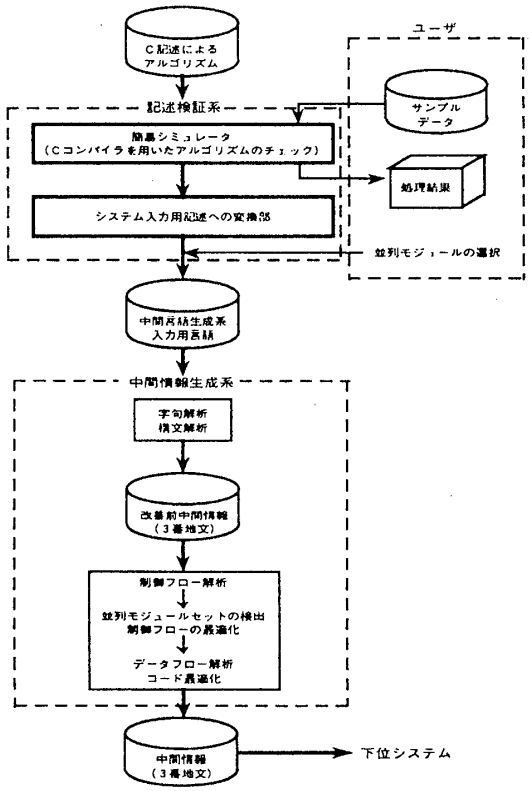


図 1: システム概要

4 むすび

今回は、C言語を基礎とした並列モジュール記述が可能なハードウェア記述言語の提案とその入力システムのSYARDSへの接続を試みた。今後の展開として、生成された中間情報による下位でのシミュレーションとその評価、その結果をフィードバックさせての並列部分の評価・検討などが考えられる。

参考文献

[1] 雨坪, 上田, 吉田, 白井:“ビット幅を考慮した大規模システム処理系の設計手法について,” 情報処理学会設計自動化研究会,67-2(1993-6)