構造化分析手法によるHDLの統一的表現の試み*

3E−1

松田　和浩　　小倉　武†
NTT LSI研究所††

1　はじめに

近年のLSI設計規模の増大、設計の短TAT化の要求から、ハードウエア記述言語による動作レベル記述を起点とする設計手法が用いられるようになってきた。市販のハードウエア記述言語としては、VHDL,Verilog-HDL,UDL/I,SFLなど数種類のものが存在し、使用されている。このため、次の問題が生じている。
(1)テキスト主体の記述であるので、動作の理解が記述者以外には容易ではない。
(2)使用する言語のセマンティックスに依存し、他の言語への変換が困難である。

われわれはこれまで構造化分析ツールを用いたLSI仕様の図的記述手法と構造化分析の結果からハードウエア記述言語-SFL-を自動生成することにより、動的な検証を行う手法について検討してきた[1]。本稿では、構造化分析手法で用いられる図形モデルと各種ハードウエア記述言語との対応規則について述べ、ハードウエア記述言語(HDL)のセマンティックスに依存しない動作記述の可能性について述べる。

2　構造化分析手法による動作記述

本検討では、Hatleyらの手法[2]に基づく構造化分析ツールSoft DA/SA[3]を用いた。Soft DA/SAでは動作の大部分を簡単な図形モデルで表現する。図1はデータ制御フロー図の例であり、データフロー、制御信号フロー、データの加工を意味するプロセス、データと制御信号を蓄積するストア、ならびに制御仕様書とのリンクを意味する制御スペック・バーからなる。データ制御フロー図はデータ制御フロー図を中心としたシステムの静的な表現である。

一方、システムの動的な振る舞いは制御仕様書によって記述される。本検討では一つの図と二つの表を用いて制御仕様書を記述する。図2は状態遷移図であり、状態、イベント、アクションを用いて、システムの動作を表現する。状態遷移図は図3に示したアクション・ロジックとともに用いられる。アクション・ロジックは、状態遷移図に記述されたアクションと起動プロセス、出力信号の関係を表わす。図4はプロセス起動表の例であり、制御信号と起動プロセスの関係を示す。状態遷移図は順序回路の動作表現、プロセス起動表は組み合わせ回路の動作表現と考えることができる。

本検討では、あるマシンサイクルに同期して動作するLSIを記述対象としている。このため、状態は1マシンサイクル内で保持される。ストアへの書き込みは書き込みが起こった次のサイクルで有効となることを仮定している。

3　図形モデルの意味定義とハードウエア記述言語の対応

3.1　ハードウエア記述言語(HDL)の分類

HDLをオートマトン記述の観点から分類すると、UDL/I,SFL[4]等のオートマトン記述の文法を持つ言語とVHDL[5],Verilog-HDL,ISP'等のオートマトンを陽に

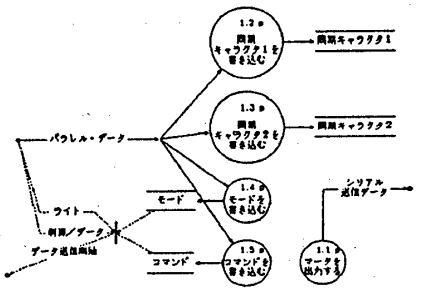

図1:データ制御フロー図の例

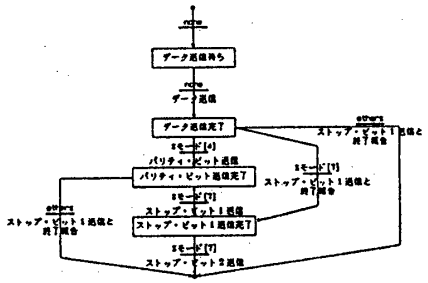

図2:状態遷移図の例

アクション	起動プロセス					コマンド・レディ
	スタート・ビットをPS変換を行なう	パリティを送信する	ストップ・ビット1を送信する	ストップ・ビット2を送信する	データ送信終了	
データ送信	1	1				
パリティ送信		1				
ストップ・ビット1送信			1			
ストップ・ビット1送信と終了報告			1			1
ストップ・ビット2送信				1		1

図3:アクション・ロジックの例

コマンド・レディ	起動プロセス		
モード[1]	パリティを計算する	偶数パリティを送信する	奇数パリティを送信する
0	1	1	
1	1		1

図4:プロセス起動表の例

*An Approach to Common Representation of HDLs using Structured Analysis Method.
†Kazuhiro Matsuda, Takeshi Ogura
††NTT LSI Laboratories

記述できない言語に分けることができる。後者の場合には、通常、状態変数と制御構文を組み合わせてオートマトンを記述する。オートマトン記述と通常の制御構文を自動的に判別するのは困難であり、例えば、VHDLからUDL/Iへの変換は容易ではない。

以降、Soft DA/SAの図形モデルとHDLの対応について、SFLとVHDLを例に述べる。

3.2 状態遷移図と言語との対応

3.2.1 SFLの場合

SFLの場合にはオートマトンを陽に記述できるため、対応付けは容易である。図1のような最下位層のデータ制御フロー図での対応関係を表1に示す。

表1:図形モデルとSFL記述の対応

図形モデル	対応するSFL記述
状態遷移図	stage
状態	state
状態遷移	goto文
イベント	alt文(条件判断文)の条件部
アクション	アクション・ロジックで示されるプロセスの起動

起動すべきプロセスが状態遷移図を持つ下位のデータ制御フロー図で詳細化されている場合には、プロセスの起動はrelay文によるタスクの遷移で行う。

3.2.2 VHDLの場合

VHDLの場合にはオートマトンを陽に記述できないので、状態変数を定義し、状態に応じた動作をcase〜when文で記述する。また、状態遷移は状態変数への代入で行う。最下位層のデータ制御フロー図との対応関係を表2に示す。

表2 図形モデルとVHDL記述の対応

図形モデル	対応するVHDL記述
状態遷移図	プロセス 状態変数(signalで定義)
状態名	列挙型で定義される状態変数の要素
状態遷移	状態変数への次状態の代入
イベント	if文(条件判断文)の条件部
アクション	アクション・ロジックで示されるプロセスの起動

起動すべきプロセスが状態遷移図を持つ下位のデータ制御フロー図で詳細化されている場合には、オートマトンの起動、停止を信号を用いたハンドシェークによって記述する。

4 変換例

Soft DA/SAを用いて記述したUSART(Intel 8251)[6]の初期化部と送信部を対応規則に従って、それぞれSFL記述、VHDL記述を生成した。記述量は、SFLの場合で約270行、VHDLの場合で約350行であった。図5に記述の一部を示す。VHDLの場合にはシミュレーションのためのクロック・ジェネレータの記述とUSARTとクロック・ジェネレータ間の接続の構造記述を付加している。両者ともシミュレーションを行い、同一の結果が得られることが確認できた。

5 まとめ

構造化分析手法による図的動作記述と各種HDLとの対応について検討を行った。SFLとVHDLを例に対応規則の検討と記述生成の実験を行い、HDLに依存しない形式での動作記述が可能であることを示した。

今後、他のHDLへの対応規則と自動生成についてさらに検討を行う。

```
module usart2(
          ...
  stage initialize{
          ...
    state wait_mode par{
          ...
      goto complete_mode;
    }
    state complete_mode par{
          ...
        goto complete_command;
    }
    state complete_command par{
          ...
        relay transmit_data.run();
        goto complete_command;
          ...
}
```

(a)SFL記述の生成例

```
architecture behavior of usart2 is
          ...
  initialize:
  process
  begin
    wait on clock until initialize_run = '1' and clock = '1';
    case initialize_present_state is
      when wait_mode =>
          ...
        initialize_present_state <= complete_mode;
      when complete_mode =>
          ...
        initialize_present_state <= complete_command;
      when complete_command =>
          ...
        transmit_data_run <= '1';
          ...
        initialize_present_state <= complete_command;
          ...
    end case;
  end process;
          ...
end behavior;
```

(b)VHDL記述の生成例

図5:構造化分析結果からのHDL記述生成例

参考文献

[1]松田,小倉,"LSI仕様記述と仕様検証の一手法について,"情処学会DAシンポジウム'91,論文集,pp.33-36,Aug. 1991.

[2]D.J.Hatley, I.A.Pirbhai,"リアルタイムシステムの構造化分析,"日経BP社,立田監訳,1989.

[3]磯田,山本ほか,"設計情報とコードの一体管理方式に基づくソフトウエア開発支援システム(Soft DA/SA),"NTT R&D, Vol.38, No.11, 1989.

[4]Y.Nakamura,"An Integrated Logic Design Environment Based on Behavioral Description," IEEE Trans., Vol.CAD-6, No.3 1987.

[5]R.Lipset, C.Schaefer and C.Usserry,"Hardware Description and Design," Kluwer Academic Publishers, 1989.

[6]Microcommunications Handbook,Intel,1988.