設計自動化　44－1
（１９８８．１０．２８）

仕様設計エキスパートシステムの検討

雪下　充輝　，　中村　行宏
ＮＴＴ情報通信処理研究所

プロセッサ・ハードウェア設計における論理設計より上位の設計支援技術は十分実用になっていない．そこで，上位の設計工程を支援すべく，設計記述のレベルを仕様記述にまで高め，仕様の記述／解析を支援する技術，更には，動作論理を自動合成する技術の研究を進めた．

まず，仕様記述にふさわしい高位の記述モデルを，「処理」，「データ」という対象（オブジェクト）とそれらの「関係」によって表現した．このモデルに基づいた日本語仕様記述を自然言語に制限を加えた形で実現した．

実際のプロトタイプシステムでは，意味解析，正当性検証には可読性，保守性の高さからＥＳ構築支援ツールＫＢＭＳのルール（２５１個）で記述した．また，動作論理合成ではオブジェクトの分類，情報の補完にルール（９６個）を用い他の部分をＬｉｓｐで記述した．

この結果，日本語仕様記述８４行を処理するのに，仕様情報解析に約３０秒，ＳＦＬ合成に約５分程度を要するという実験結果を得ている．修正，再解析を繰り返すことが設計作業の特徴であるが，解析処理は数十秒のオーダであり実用上は一層の高速化が必要である．一方，合成処理は最後に一度実行するだけであるから，数分オーダでも実用上問題がないと考える．　この論文で述べた手法は，ユーザは使い慣れた日本語を用いてハードウェアの設計を行うことができ，筆者らが狙いとしている仕様設計の段階からの支援を進めて行っていく上で有効な方法であると考える．

An investigation of specification design expert system

Mitsuteru YUKISHITA and Yukihiro NAKAMURA

NTT Communications and information Processing Laboratories
1-2356 Take Yokosuka-shi Kanagawa 238-03 Japan

At the present stage of processor hardware design, design support technology for lower levels, such as layout design, has reached a practical state. We have studied and developed an upper level CAD system.

We express those specifications in basic terms with object(or entity) descriptions for processes and data and relation descriptions for the correlation between these entities.

An expert system for computer architecture design has been implemented. With it, the designer is able to use a specification description language in much the same way as a natural language. The system employs 251 rules to analyze the consistency and completeness of the specification. Finally, an RTL behavioral description can be synthesized from the original specification description through a process that employs 96 rules.

1．はじめに

プロセッサ・ハードウェア設計の現状を概観すると，レイアウト設計などの低位レベルの設計支援技術は実用になっているが，論理設計より上位の設計支援技術は十分実用になっていない．そのため，全設計工程の７０％以上を論理設計の上位の工程，すなわち，方式設計，機能・動作設計，論理設計で占めており[1]，これら上位の設計工程段階を支援するツールが強く望まれている．この要望に応えるべく，我々は高位のＣＡＤの研究・開発を進めている．

研究に当たっての一貫した考え方は，⑴従来の接続記述ではなく，上位設計にふさわしい高位の設計記述法を開発し，人間にとって認識しやすい概念記述をそのまま設計データとして扱い可能とする，更に，⑵方式設計の特徴である試行錯誤への対処に適した処理環境を実現する，というものである．そのため，設計という知的活動の中のルーチンワーク的な部分を自動化して，本来の設計の核となる部分に設計者が専念できるＣＡＤシステムの開発を目標とした．

この目標に対し，接続よりも高位の概念である「動作」を直接記述できる動作記述言語ＳＦＬ，会話処理による検証系，ゲート論理回路の自動合成系を研究・開発済みである[2][3][4][5][6][7]．

ＳＦＬをベースとした動作設計システムは，マン・マシンインタフェースの高位化という点で，現在の実用ツールとして最先端に位置するが，まだ，論理設計の専門家でないと十分には使いこなせない．そこで，更に，設計記述のベースを高位化して，仕様記述にまで高めることを次の目標とした．

一般に，プロセッサ・ハードウェアの論理仕様は自然言語で書かれている．そこで，自然言語を設計言語として用い，仕様の記述／解析を支援する技術，更には，動作論理を自動合成する技術の研究を進めた．この自動合成系の実現には，設計ノウハウを知識ベース化するなど知識処理技術の導入が必要であり，エキスパート・システムとしてプロトタイピングを行なった．

なお，自然言語によるインタフェースを備えたＣＡＤ，自然言語による仕様から動作論理を自動合成する技術については，これまでにほとんど発表されていない．

2．仕様記述言語

筆者らはＲＴ（ﾚｼﾞｽﾀ・ﾄﾗﾝｽﾌｧ）レベルの動作記述言語ＳＦＬをベースとした設計支援技術を開発し，ハードウェアの動作だけを記述することにより論理回路まで自動で合成可能な設計環境を整えてきた．

設計記述のベースを更に仕様記述にまで高めるため，

⑴仕様記述にふさわしい，高位の記述モデル
⑵記述形式などマン・マシンインタフェース

について以下の研究を進めた．

2.1 基本モデル

ハードウェアの仕様を記述するのに必要な基本モデルについて述べる．筆者らは，仕様は，「処理」と「データ」という対象（オブジェクト）の記述と，それらオブジェクト間の「関係」の記述により基本的には表現したいと考えた．このような簡明なモデルを考えた理由は，限られたタイプのオブジェクトによりプロセッサを構成する要素を表現し，ハードウェア設計のための限られた関係記述によってオブジェクトを結び付けることにより，人間にとって記述が容易で，かつあいまいさの少ない形で仕様を記述しようとしたからである．

このような狙いのもとにプロセッサの仕様を分析し，

⑴オブジェクトとしては，システムの動作の中で操作される情報を記述するための「データ・オブジェクト」，システムの動作の中の個々の処理を記述するための「プロセス・オブジェクト」を用意し，また，
⑵これらオブジェクトを結び付ける関係（リレーション）としては，「構造関係」、「データの流れの関係」、「制御の流れの関係」を用意した．さらに，
⑶関係レベルを関連付ける「上位関係」を用意した．

動作仕様を記述するには、処理の流れの中で、処理の順序関係／実行順序を制御する条件関係の記述が必要になる．具体的には，「事象（～の場合）」，「条件（～の時）」，「時間（～の前／後／同時に）」であるが，これらはオブジェクトどうしを結び付けるものではなく，関係レベルを関連つけているものである．

そこで「上位リレーション」を導入した．表１にオブジェクト，リレーションの関係を示す．

表1 リレーション
Table 1 Relation

リレーション種別	関係	オブジェクト		機能
		ソース	シンク	
構造	構成	プロセス	プロセス	構成上の階層を示す
	〃	データ	データ	データの集合を定義する
	利用	プロセス	プロセス	呼び出し構造を示す
	属性	データ	データ	データの値を定める
データの流れ	参照	プロセス	データ	データを参照する
	生成	プロセス	データ	データを作り出す
	保持	プロセス	データ	データを保持する
	加工	プロセス	データ	データの内容を変更する
				データの値が条件値であるから，条件値の変更と等価
	転送	プロセス	データ	データの転送を行なう
制御の流れ	起動	プロセス	プロセス	他のプロセスを起動し，自分は終了
	発生	プロセス	プロセス	他のプロセスを起動し，自分は動作
	終了	プロセス	プロセス	他のプロセスを終了させる
制御の流れ（上位リレーション）	前後	ニ		リレーション間の二項関係を決める
	条件	ニ		リレーション間の条件関係を定める
	時間	ニ		リレーション間の時間関係を定める

以上のオブジェクト，リレーションを用いればハードウェアの仕様を一通り記述できることをＮＴＴの既開発プロセッサ仕様の解析などにより確認した．

また，仕様設計エキスパートシステムのマン・マシンインタフェースとして，高位な記述にふさわしい日本語を用いることとした．

以下に，オブジェクト，リレーションの詳細を示す．

オブジェクト

・プロセス　：
論理装置内のデータ及び制御の操作のまとまり（処理）を示す．装置内の基本的な個々の操作を示すものから，複数の処理の単位の集合を示すものまで，プロセスで記述する．

・データ　　：
装置内で保持されたり，転送される情報を示す．ハードウェアとの対応としては，記憶要素・端子・配線などが考えられる．

リレーション

・構造　　：
複数の処理の単位（プロセス）をある程度物理的な階層関係でグループとしてまとめる物理的構造，及び，あるプロセスが他のプロセスの機能を利用することにより自己の機能を実現する機能的な構造を示す．

・データの流れ：
データに対する操作及びデータの転送に関する関係によってデータの流れを示す．また，条件のセット，リセットは制御の流れであるが，条件値はデータであり，データの変更に等価であることからここに含める．

・制御の流れ：
タイミングや条件等を考慮にいれない基本的なプロセスとプロセスの間での起動・終了・発生に関する関係，等の制御の流れを示す．

上位リレーション

・制御の流れ：
前述したリレーションの間の実行順序等のタイミングの記述，及び，あるリレーションが条件となり，別のリレーションを実行するような上位の関係．

2.2　日本語仕様記述言語の文法

前述のオブジェクト，リレーションモデルを基本構造とし，記述形式として自然言語（日本語）を採用したものが仕様記述言語である．ただし，対象がハードウェアの仕様記述に限定されること，および，言語の構文解析を容易にすることを考慮し，以下のような記述上の制限を設けた．

制限事項：(1)使用可能な助詞・助動詞の制限など，予約語の設定

(2)１つの動詞に対する構文は１つに制限し，構文解析上の曖昧性を除去

2.2.1　予約語

ハードウェアの仕様表現に限定されることを考慮し，あらかじめ予約語を設定した．予約語は，表２に示す１５３個である．

記述実験により，基本的な記述性を確認している．

表２　予約語リスト
Table 2 List of reserved verb

動詞	成る 定める 保持する 加工する インクリメントする 反転する ANDする 桁ORする なる 生成する する 発生させる		構成する 利用する 送る デコードする インクリメントする ORする 桁ANDする 加える 更新する 参照する なった 終了させる			持つ 用いる 受け取る エンコードする シフトする ORする 桁EORする 引く 生じる 変換する 起動する	
助詞	は へ	が として	を で	から の	に	により	によって
接続	場合 かつ	場合に または	時	時に	前に	後に	同時に
名詞	属性 真	条件 偽	内容	データ	要素	結果	プロセス

注）動詞は「構成し」等の連用形，「構成される」等の受動型を省略

2.2.2　構文

日本語仕様記述においては，人間からみて自然な表現を狙い，オブジェクト，リレーションを“主語”＋“目的語”＋“動詞”の形の単文で表現し，上位リレーションを“単文”＋“接続語”＋“単文”の形の複文で表現する．ただし，文に曖昧性を持たせないために主語は１つに制限する．

また，複文では“単文１”を条件節として，単文中の主語と目　的語の間に割り込ませた形とする．これは，処理の主体である主語が行う処理を一括して記述することを主眼においているためと，構文解析処理を簡単化するためである．

3　仕様解析技術

日本語仕様記述言語で記述された仕様は，パーサ部分である字句解析部，構文解析部，及び，仕様記述からハードウェア構成を解析する意味解析部により解析される．

(1)字句解析部では，入力された漢字かな混じり文の日本語仕様記述のテキストを予約語情報に基づいて単語に切り出す．

(2)　構文解析部では，名詞と予約語の列を，主語・目的語・動詞といった単文構造と，それらを組合せて生成された複文構造を表す情報に変換する．

(3)意味解析部では，単文に対して動詞を中心に主語・目的語の表すオブジェクトのハードウェア構成上での意味付けを行い，複文に対して接続詞を中

心に構文要素である単文の意味付けを行う．加えて，その意味付けの正当性の検証をルールで行い，主語・目的語間の関係からリレーション，複文の構文要素の関係から上位リレーション，を表現する「仕様情報」のリレーションネットワークを生成する．

(4)レポート出力部では，エラーの表示に加えて，構造，データの流れ，制御の流れをユーザの必要時に随時木構造で表示する．

3.1 字句解析部

仕様記述入力が漢字かな混じり文であり，「かな」である助詞の切り出しが容易であることから，助詞に着目する．また，使用される名詞に固有名詞が多いことから，一般的な名詞辞書は持たない．そこで，設計仕様に対応させ登録した助詞・助動詞・接続詞・動詞・少数の名詞を予約語とする．字句解析部は，優先順位の関係から，予約語である助詞→助動詞→接続詞→動詞→名詞の順で検索を行い，最後にどれにも符合しないものを名詞として切り出し，最終的に仕様記述を名詞と予約語の列に変換する．

3.2 構文解析部

構文の解析は，大きく単文における主語・動詞・目的語の関係から導かれるリレーションレベルの意味解析と，複文の構造から導かれる上位リレーションレベルの意味解析とに分けて行う．単文の解析は，動詞と目的語を指示する助詞による構文パターン情報を用いて行い，複文の構造の解析は，接続詞などの文を切り出す予約語の位置とそれによって切り出された単文の組合せの情報を用いて行う．

上記字句解析，構文解析の処理系はＬｉｓｐによって作成している．

3.3 意味解析部

意味付け処理は，予め用意した動詞をキーワードとした構文パターンと，単文中の動詞，主語，及び，目的語とのパターンマッチ処理により各オブジェクトの意味付けを行う．処理の基本がパターンマッチ処理であることから，エキスパートシステム構築支援ツールＫＢＭＳ[8]を用いて，If-then形式のプロダクションルールで処理を記述した．

また，正当性検証処理では，日本語仕様記述されたハードウェアの正当性を検証する際に使用する検証項目の可読性，保守性を高めるため，検証知識をルールで記述した．

ＫＢＭＳでは事実に関する知識はフレーム[8][9]，推論を行うための経験的な知識をルール，により表現している．以下に，解析処理について具体例を用いて説明する．

3.3.1 解析処理

解析処理は，(1)単文からリレーションの生成，(2)複文から上位リレーションの生成を行うという２段階からなっている．

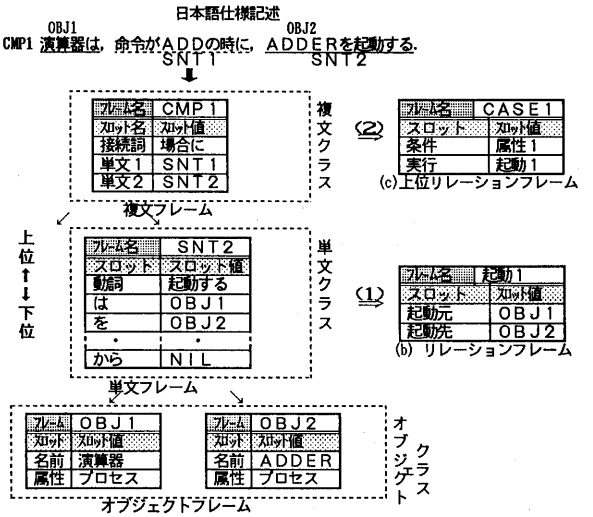

図1 解析処理におけるインスタンスフレーム
Fig. 1 Example of instance frames made by analysis

(1) 単文からリレーションの生成（図１(1)）

２．１節の基本モデルで述べたが，ハードウェアの構成を記述する上で必須なものとして，構造，データの流れ，制御の流れがある．それらのリレーションを明確化しているのは単文中の動詞である．したがって，具体的なルールは，図２のように単文フレーム内の動詞，助詞スロットの値を条件部に持ち，検出した構文フレームからリレーションフレームを生成する実行部を持つ．

```
(frame (単文 ?s
          (verb 起動する)
  条件部   (は   ?cr)
          (から ?cd) ))
-->
(create-instance '構成  nil
                 'caller ?cr
  実行部         'called ?cd)
```

図2 リレーションフレーム生成ルール
Fig. 2 A Rule for making relation instance frames

図２の生成ルールによって，単文構文フレーム群中で動詞に「起動する」を持ち，助詞として「は」，「を」を持つものが存在したとき，「は」スロットに代入されている主語?crをcaller（起動元），「を」スロットに代入されている目的語?cdをcalled(起動先）のオブジェクトとする「起動関係」クラスのインスタンスを生成する．生成されたリレーションフレームを図１.(b)に示す．

上述のルールにより，単文から生成されるリレーションレベルのフレームが生成される．

⑵複文から上位リレーションの生成（図１⑵）

単文フレームからリレーションのフレームを生成後，複文の解析を行う．単文レベルの解析では動詞に着目しているが，複文レベルでは接続詞に着目して処理を進める．処理手順は単文レベルと同様にルールで行う．対象とするフレームが複文クラスのフレームである点が異なるが，基本的な考え方は同様であるため，詳細は省略する．

3.3.2 正当性検証

前節において知識ベース（ＫＢＭＳ）のルールにより一括生成したリレーションネットワークを表現するフレームに対して検証を行う．

検証項目として，構造に関するもの，データの流れに関するもの，制御の流れに関するものがあり，それぞれのリレーションフレームに関するルールによって検証を行う．

これらのルールの検証項目一覧を表３に示す．

表3　検証ルール項目
Table 3 Items of checking rule

構造	２５ルール
再帰的構造	構造を示す関係がループを生じている
ﾌﾟﾛｾｽのｸﾞﾙｰﾌﾟ化	プロセスが２つ以上のプロセスの構成要素となっている
データの流れ	６４ルール
存在	未生成で存在する
保持と更新	保持されているが更新されていないﾃﾞｰﾀがある，またはその逆
プロセスとの関係	プロセスによって操作されていない
更新とﾃﾞｰﾀの流れ	更新されたデータが利用されていない
ﾌﾟﾛｾｽとﾃﾞｰﾀの入出力	データを受け取るが，ほかのデータに渡さないプロセスがある
名称の一意性	または，ほかにデータを渡すがそのデータを受けていない
	プロセス間を移動するデータの名称が一意でない
制御の流れ	２８ルール
再帰	プロセスの起動関係でループを生じている
起動の欠如	起動されていないものを終了する
プロセスの孤立	プロセスの起動／終了のつながりから孤立している
上位の制御の流れ	３２ルール
非条件節の存在	構造の記述を複文の条件節に記述する
再帰	時間的なループを生じている

上記の項目をもとにして，リレーション毎に数個，合計で１５０余りのルールにより検証を行う．

具体的なルールとして，構造に関するチェックルールを用いて説明する．

「ＡはＢから構成される」という記述が存在し，上位構成物＝Ａ，下位構成物＝Ｂなるリレーションフレームが存在する場合に，別のところで，

「ＢはＡから構成される」という記述が存在し，上位構成物＝Ｂ，下位構成物＝Ａなるリレーションフレームが生成されると，ハードウェアの構成上矛盾が生じる．このような矛盾を検出するルールを図３のように記述することにより，再帰的な構成を検出可能である．

```
(Frame (構成 ?p1 (上位 ?u) (下位 ?l) ))
(Frame (構成 ?p2 (上位 ?l) (下位 ?u) ))
-->
(call (format t "~%構成関係で矛盾があります
              ~A ~A" ?p1 ?p2) )
```

図3　検証ルール例
Fig. 3 Example of check rule

3.4 レポート機能

仕様記述結果（仕様レベルの設計データ）に対し，前述の検証機能によりエラーをレポートするのは当然として，関係ネットワークを解析することにより，設計者の意図通りの仕様になっているかをシステムが報告してくれれば非常に有効である．ここでは，「構造」，「データの流れ」，「制御の流れ」を木構造で表示するコマンドを用意した．実験例で示す単純計算機についてそれぞれのコマンドによるレポート例を図４に示す．このように出力したレポートから仕様の全体像を把握することができる．その結果，必要であれば，記述を変更し，再度，解析処理をやり直す．

以上の操作により，各種誤りのチェックが設計の進度に合わせて行える．

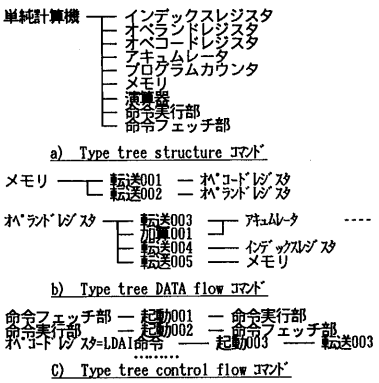

図４　レポートの実行例
FIg.4 Example of report executing

４．動作論理合成技術

仕様設計が固まると，次に詳細な設計データを作成する必要がある．筆者等は，すでに，ＲＴレベルの設計言語ＳＦＬとＳＦＬからゲートレベルの論理回路を自動合成するシンセサイザの開発に成功している．そこで，仕様レベルから論理回路レベルまでの体系的な設計支援技術の構築を狙い，仕様記述からＲＴＬ動作記述へ論理合成技術について考えた．

仕様記述では，前述したように処理，データ，リレーションにより表現している．そこで，これより，詳細なＳＦＬ動作論理を合成するため，以下に示すような方法を検討した．

⑴まず，仕様解析結果（オブジェクト×リレーション解析）により，オブジェクトを分類して，ＳＦＬの概念への対応付けを行う（処理にはモジュール，ステージ（ＳＦＬ用語：後述），レジスタ，組合せ回路，データには端子，記憶素子，リレーションには信号線，デコーダ出力，が対応する）．

⑵次に，モジュール，端子などに応じて，あらかじ

め用意したＳＦＬの記述形式を持つクラス・フレームに対し，仕様記述に応じてインスタンス・フレームを生成する．ここで，クラス・フレームはＳＦＬの各記述項目をスロットとして持っており，仕様解析により得られたデータをもとにシステムがスロット値を与えていく．

⑶仕様記述で陽に記述されていないスロット値を知識により補充する．

⑷仕様記述段階では明記されていないが，動作記述で詳細化されるべき項目として，①クロック分割，②状態割付，③組合せ回路の機能などがある．これらを自動合成するには，最も高度な設計ノウハウを知識ベース化する必要があるが，今回は，その第一歩としてシステムから設計者に問い合わせを行う会話処理形態として試作した．

以下にこれらの処理について具体的に述べるが，まず最初に，論理の合成先であるＳＦＬ動作記述について，概要を述べる．

4.1 動作記述言語ＳＦＬ

ＳＦＬ記述のモデルは，上述の仕様記述における機能記述レベルのモデルではなく，内部がどういう仕掛になっているかを表現する作りの記述のモデルである．

①ハードウェアの枠を表すモジュール（サブモジュール含む），
②動作主体を含み，動作を独立に制御できるステージ，
③データを表すデータ端子，
④制御の流れを表す指示端子，
⑤動作主体を含まず，ただ使われるだけの存在で，ブラックボックス的扱いが可能な機能回路，メモリ，レジスタ，

等のハードウェア要素から成る．このモデルで動作主体であるステージが客体である機能回路，端子，指示端子を使う手順，及び，他のステージへタスクの遷移等を表現する．このように，並列に独立動作可能なステージの動作を記述することによりハードウェアを漏れなく記述することが可能である．

4.2 オブジェクト分類

リレーションネットワークにおけるオブジェクトがハードウェア要素のどれに対応するかを分類する．

仕様記述のオブジェクトとＳＦＬのハードウェア要素との対応は表４のようになっており，対応付けルールは，構造／データの流れ／制御の流れに関するものを基本とし，これらの組合せによってプロセスの分類を行なっている．表５にルールの分類を示す．

このルールは，各ハードウェア要素が持つ特徴（他プロセスへの制御の関係，構造のレベル及び下位に持つプロセスの種類等）に注目したものである．

表４　仕様記述とＳＦＬ記述との対応
Table 4 Relations between Entities and Elements

仕様記述のオブジェクト	ＳＦＬの記述要素
プロセス	モジュール，レジスタ 機能回路，機能名など
データ	端子 内部バス　など
制御	デコーダ， 指示端子　など

表５　ルールの分類
Table 5 Classification of rules

プロセス	プロセスが機能回路に対応するか プロセスがレジスタに対応するか プロセスがモジュールに対応するか プロセスがステージに対応するか
構造	オブジェクトが下位のオブジェクトを持つか オブジェクトが構造の最下位であるか
データ	データがプロセスから外へ流れるか データがプロセスの中へ流れるか
制御	制御がデータの操作に影響するか 制御がプロセスから影響を受けるか

機能回路を例に分類の考え方を以下に示す．

機能回路の特徴は，以下の４つである．

①データを持つ，②データを加工する
③プロセスにより利用される
④プロセスを制御しない

従ってリレーションネットワーク内で，機能回路であるプロセスは，

①保持関係の保持の主体，
②加工関係の加工の主体，
③利用関係の利用の客体，

というリレーションを備えており，

④利用関係の利用の主体，

というリレーションを備えていないというパターンが定まる．

つまり，あるプロセスに関して①②③のリレーションがあり，④のリレーションがないならば，それを機能回路と同定する．

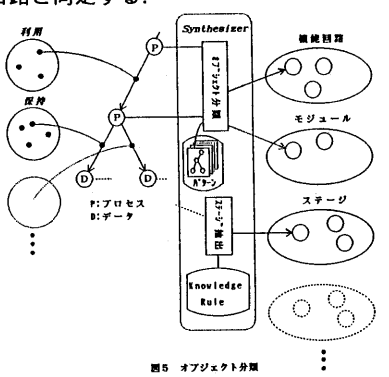

図５　オブジェクト分類
Fig.5 Classification of Entities

そこでこのようにリレーションのパターンが定まっていることに着目して，プロセスとハードウェア要素との対応付けを行う．（図５）

```
演算器は，ADDとANDから成り，
        命令セットによって利用される．
演算器は，演算結果を更新する．
ANDXは，演算器とｲﾝﾃﾞｯｸｽﾚｼﾞｽﾀとｱｷｭﾑﾚｰﾀと
                        ﾒﾓﾘｲﾝﾀﾌｪｰｽを利用し，
        ｲﾝﾃﾞｯｸｽﾚｼﾞｽﾀの内容をﾒﾓﾘｲﾝﾀﾌｪｰｽへ送り，
        ｱｷｭﾑﾚｰﾀの内容とﾒﾓﾘﾃﾞｰﾀを演算器に送り，
        演算器の結果をｱｷｭｰﾑﾚｰﾀへ送る．
ﾒﾓﾘﾃﾞｰﾀは，ｱｷｭﾑﾚｰﾀとｲﾝﾃﾞｯｸｽﾚｼﾞｽﾀと演算器に送られ，
        bit長が８である．
演算結果はbit長が８である．
```

図6　仕様記述の例
Fig. 6 Example of a specification

ここで，図６の仕様記述の例で演算器に着目すると，演算器は命令セットから利用され（③），メモリデータを送られ（①），演算結果を更新し（②），かつ④に示した記述がないので，機能回路に該当している．

4.3　フレーム生成

プロセスの分類によって認識されたハードウェア要素（モジュール・機能回路・ステージ）に対し，ＳＦＬ記述を合成するために必要な情報を取り出す必要がある．

そこで，その記述形式を枠組みとして予め用意してＳＦＬ動作記述を引き出すこととした．つまり，記述形式に対応するフレームを用意しておき，リレーションネットワークから得られる情報をフレームの持つスロットに対応させ，最終的にＳＦＬ記述を生成する．

図６の例の場合には，演算器は機能回路であったので，図７.aに示す機能回路のＳＦＬ記述形式に対応するフレーム（クラス：図７.bのcircuit_class）に対応付けたインスタンスが生成される．

```
<機能回路種定義部>::=circuit_class 機能回路名 {
                    〈入力端子定義部〉
                    〈出力端子定義部〉
                    〈機能定義部〉 }
```

(a)　記述形式

```
circuit_conponent
 term     :端子に対応          ── terminal
 input    :入力データに対応          bitlength :ビット長に対応
 output   :出力データに対応          flow      :データの流れに対応
 instruct:機能に対応                 :
 reg      :レジスタに対応       └─ instruction
 mem      :メモリに対応                instruct  :機能の記述
                                       argument  :引数
                                       :
```

(b) フレーム

図７　ＳＦＬ記述形式とフレーム（機能回路の例）
Fig. 7 SFL Description style and frame

4.3.1　スロット補充

分類されたプロセスに対応するフレームのスロットを埋めるに当たり，仕様記述では端子のように陽に記述されていないが，ＳＦＬでは明示されなければならないものがある．

‘端子’の値を得るためには，データ・制御の流れがどのプロセス間の流れであるか，流れの向きはどちらであるか，データの送出元での値が記述されている場合上位リレーションが存在して条件となり得るか，等に着目した「知識」を用いて，データや条件に分類して種々の端子に対応付ける．

図６の例において，演算器に関係するデータの流れは以下の通りである，

①“ANDXはｱｷｭﾑﾚｰﾀの内容とﾒﾓﾘﾃﾞｰﾀを演算器に送る”，
②“ANDXは演算器の結果をｱｷｭｰﾑﾚｰﾀへ送る”，
③“ﾒﾓﾘﾃﾞｰﾀは，演算器に送られ，bit長が８である”，
④“演算結果は，bit長が８である”．

ここで①は演算器に対するデータの入力，②はデータの出力，③，④は各bit長を示す．これにより図７に示したフレーム[terminal]のスロット[bit-length, flow]が埋まり，演算器に対応するフレーム[circuit_conponent]の端子を示すスロット[term]と方向性を示す[input]が埋まる．

4.3.2　動作手順入力

仕様記述では，動作の順序関係は部分的な二項関係だけが記述され，ステージ内の全体の順序は記述されていない．例えば，Ａ→Ｂ，Ｃ→Ｄ（前→後）なる二項関係が記述されていた場合，全体の関係はＡ→Ｂ→Ｃ→Ｄ，Ｃ→Ａ→Ｂ→Ｄ，等，どれが正しいかは判断できない．したがって，知識で全体の順序を決定することは困難である．

また，ステージは複数の「状態」をとることができるが，仕様記述からは，どの動作をステージとして設計すべきか，または，あるステージの１つの「状態」として設計すべきかは，陽には判別できない．ここをいかに設計するかは，並列制御に関する高度な設計知識が必要である．何を知識ベース化するかを明らかにするため，今回の実験システムでは，設計者に問い合わせを行い会話的に情報収集を行うこととした．

以上により，ＳＦＬ記述生成に必要な詳細情報を全て獲得する．そして生成されたフレーム（インスタンス）を展開することにより，ＳＦＬを生成する．

５．仕様設計ＥＳの構成

以上の研究に基づき，仕様記述エキスパートシステム（ＥＳ）のプロトタイプを試作した．本ＥＳは仕様解析技術を実現する仕様情報抽出部と動作論理合成技術を実現する設計情報合成部から構成する．

３，４章をまとめると，仕様情報抽出部では入力された日本語を論理装置の仕様として，その構造・

データの流れ・制御の流れを解析・抽出する．次に，仕様記述の意味的矛盾検出を行うため，モデルに沿ってハードウェアを表現するリレーションネットワークを生成する．このリレーションネットワークにおいて，矛盾の無いものを最終的に仕様情報のリレーションネットワークとして出力し，設計情報合成部に渡す．

次の設計情報合成部では，リレーションネットワークから設計情報を抽出し，ＲＴＬ（レジスタ・トランスファレベル）動作記述言語ＳＦＬによる動作論理を合成する．

６．　実験例

単純計算機を対象に，上記プロトタイプを用いて，仕様解析，動作論理合成の実験を実施した．

日本語仕様記述８４行を処理するのに，仕様情報解析に約３０秒，ＳＦＬ合成に約５分程度を要するという実験結果を得ている．情報解析処理でのルールは，対象とする仕様記述が単文であれば動詞，複文であれば接続詞をキーワードとして処理を進めていることに着目して，キーワード毎にルールのグループ化を行なっている．加えて，ルール内でのキーワードとのパタンマッチを条件項目の先頭に置くなど，リートネットワークの特性を考慮したルールを記述している効果がでていると考える．

また，修正，再解析を繰り返すことが設計作業の特徴であるが，解析処理は数十秒のオーダであり実用上は一層の高速化が必要である．一方，合成処理は最後に一度実行するだけであるから，数分オーダでも実用上問題がないと考える．

７．おわりに

本論文では，ハードウェア要素に関する知識及びＳＦＬ動作記述に関する知識を知識ベース化し，日本語による仕様記述を解析・評価し，次に機能・動作論理を合成する方法について述べた．

仕様情報抽出での意味解析，正当性検証には可読性，保守性の高さからＥＳ構築支援ツールＫＢＭＳのルール（２５１個）を用いた．

また，ＳＦＬ合成では，仕様記述の基本要素であるオブジェクトの分類，ＳＦＬのみで必要な情報の補完にルール（９６個）を用いた．

この論文で述べた手法は，ユーザは使い慣れた日本語を用いてハードウェアの設計を行うことができ，筆者らが狙いとしている仕様設計の段階からの支援を進めて行っていく上で有効な方法であると考える．

今後の課題として，より良いマン・マシンインタフェースの実現を目指して，

①仕様抽出処理での矛盾検出時の的確なユーザへの助言（どこがどの様な理由で矛盾しているか指示等），

②従来の設計を効率よく利用するためのライブラリィの構成法，

③動作論理合成の自動化率拡大のための設計知識の抽出法，

について，引続き検討を進めて行く．

謝辞：

日頃のご指導，ご支援に対し，ＮＴＴ情報通信処理研究所　知識処理研究部　村上国男部長に感謝致します．

参考文献：

[1] M.Breuer, A.Friedman, and A.Iosupovicz, "A survey of the state of the art of design automation," Computer, Oct.1981, pp.58-75.

[2] Y.Nakamura, K.Oguri, H.Nakanishi, and R.Nomura, "An RTL Behavioral Description Based Logic Design CAD System with Synthesis Capability," Proc.7th International Conference on Computer Hardware Description Languages and their Applications, 1985, pp.64-78.

[3] K.Oguri, Y.Nakamura, and R.Nomura, "Evaluation of Behavior Description Based CAD System Used in Prolog Machine Logic Design," Proc. of the International Conference on Computer Aided Design (ICCAD-86), Nov. 1986, pp.116-119.

[4] Y.Nakamura, "An Integrated Logic Design Environment based on Behavioral Description," IEEE Trans. CAD, Vol. CAD-6, No.3, 1987, pp.322-336.

[5] Y.Nakamura and K.Oguri, "An RTL Logic Design Aid for Parallel Control VLSI Processors," Proc. of the 4th International Conference on VLSI (IFIP VLSI 87), 1987, pp.13-28

[6]中村，小栗，野村，「ハードウェア動作記述言語ＳＦＬ」　情処第２９回全国大会　1984

[7]仲西，打橋，小栗，中村，「ハードウェア向き仕様記述システム」　情処第３２回全国大会　1986

[8]服部，清水，土屋，桑原，和佐野，「知識ベース管理システム（ＫＢＭＳ）」　情報処理学会，知識工学と人工知能 41-6 1985