設計自動化　５０－４
(1989.12.14)

機能設計支援システムＳＴＬｔｏｏｌｓの図的入力法

小林 一夫　　若林 春夫　　脇村 慶明

NTT交換システム研究所

あらまし　大規模化・構成の複雑化の進むハードウェアの設計を効率化するため、ハードウェアの機能仕様からゲート回路レベルの論理回路とマイクロプログラムを自動生成する機能設計支援システムSTLtoolsを開発した。このシステムでは、機能仕様の計算機入力、解析、保守及び設計者間の情報交換を同一の表現形式に統一するため、次のような特徴を備えた図的な機能記述法を採用した。（１）機能ブロック間の接続図と状態遷移図が記述でき、機能の複雑さ、要求される性能等に応じて使い分けられる。（２）接続図においては、グローバル信号の概念を導入し概要図から詳細図までトップダウン的に記述できる。（３）状態遷移図では、多相クロックを用いた同期動作モデルを導入し、多様な時間表現が可能である。

Schematic Entry for Top-down Hardware Synthesis System : STLtools

Kazuo KOBAYASHI　Haruo WAKABAYASHI　　Yoshiaki WAKIMURA

NTT Communication Switching Laboratories

Musashino-shi, Tokyo 180, Japan.

Abstract

A top-down hardware synthesis system (STLtools) generates automatically both gate-level circuits and micro-codes from a register transfer level (RTL) description. This paper proposes a new schematic notation as the RTL description medium. This notation consists of two types of diagrams:a functional block diagram and a state transition diagram. The functional block diagram allows a hardware designer to describe the RTL structure in the top-down manner using a concept of global signal. The state transition diagram uses a control model with multiple clock-phases. Consequently, the designer can specify various timing conditions of the RTL behavior. The designer can use properly either diagram according to cost and performance requirements of the hardware.

1　まえがき

大規模なハードウェアの設計は大勢の設計者に分担され組織的に行なわれる。そのためには、設計手順の標準化が不可欠であり、設計の過程を仕様（方式）設計、機能設計、論理設計および実装設計の各設計階梯に分けて、各階梯ごとに形式の定められた設計ドキュメントを作っていく。上位の階梯でつくられた設計ドキュメントは、1階位下の階梯の設計仕様となり、更に詳細な設計情報がつけ加えられる。たとえば、機能設計では、方式設計で作られた仕様書などをもとに、その機能を実現するハードウェアの構成と動作を規定する。すなわち、資源間のデータの流れを規定するデータパス回路の設計及び、データ操作の手順を規定する制御回路の設計が行なわれる。そして、前者は機能ブロック図、後者は動作フロー図や状態遷移図などの設計図面が設計ドキュメントとして表わされる。

この設計手法を支援するため、機能設計段階の設計情報を入力として論理設計以降の自動化を図るハードウェア記述言語や論理合成技術が開発されるようになってきた[1]。これらのツールでは、主に設計図面をテキスト形式の記述言語で書き換えて計算機に入力する。これに対して、設計情報の計算機入力・解析・保守、および設計者間の情報交換が同一の記述形式でできるようにして設計の効率化を図るため、図的なハードウェア記述言語を用いた設計支援ツールが検討されるようになってきた。

この図的なハードウェア記述言語は、機能ブロック図のような構造を記述するもの(例えば、[2]-[4])、ペトリネット図形式(例えば、[5]-[7])または状態遷移図形式(例えば、[8]-[11])に基づく制御動作を記述するもの、及び構造と動作を記述するもの(例えば、[12])に大別される。

このうち、ペトリネット図形式は、記述できるハードウェアの仕様と機能を実現する回路とのギャップが大きいため、論理合成やタイミング解析などには、状態遷移図記述か機能ブロック図記述が用いられてきた。しかし、これまでの状態遷移図または機能ブロック図に基づくツールは、大規模なハードウェアの設計を支援する上で、以下のいずれかの課題を抱えている。

1)機能ブロック図を記述するツールでは、上位の階層の図において下位の図に現われる信号を省略できないため、概要図から詳細図に書き下して設計を進めるトップダウン設計手法には不向きである。

2)状態遷移図を記述するツールでは、遷移の同期に単相クロックを用いていること、制御信号を直接記述することなどから、多相クロックを用いたより詳細な時間条件やデータ転送動作を陽に記述できない。また、装置レベルから論理ブロック（制御回路、データパスなどの機能モジュールの構成要素）レベルまでの機能の階層化に対応する表現手段を持たない。

これらの課題に対処するために、我々は、トップダウン的記述が可能な機能ブロック図と時間条件の記述が可能な状態遷移図とを組み合わせて入力する機能設計支援システムSTLtools (STL:fnctional Structure diagram & state Transition diagram discription Language)を開発した。

本稿では、STLtoolsの設計情報の入力ツールを実現する図的記述法とその試用結果を報告する。

第2章で、システムの中に占める図的入力の位置を示し、第3章で、実現上の課題を整理する。第4章で、状態遷移図の記述法、第5章で機能ブロック図の記述法を述べる。第6章で、STLtoolsの試用結果を元に、これらの図的入力の効果を示す。

2　機能設計支援システムの概要

2．1　システムの構成

STLtoolsでは、機能設計で作られる機能ブロック図と状態遷移図を計算機に直接入力し、そこから下位の設計情報（論理回路、マイクロコード）への展開、記述の検証、機械書き図面の印刷を支援する。このうち、論理回路の合成と論理検証にはLSI-CAD[13]を利用する。

これらを実現するため、図1に示す各種のツールから構成される。

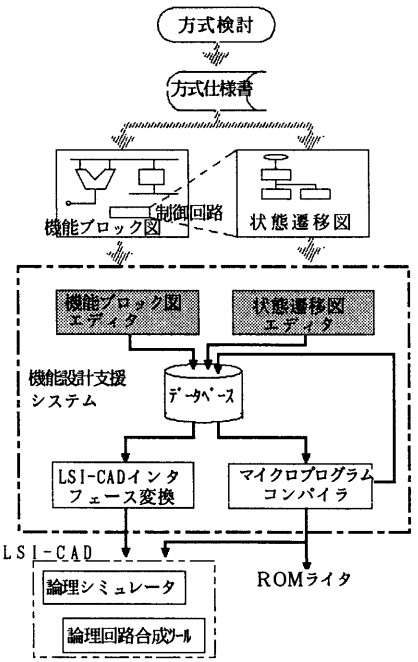


図1　STLtoolsの構成

2．2　設計支援ツール

（1）設計エントリ

設計図面対応に2種類の図面エディタ、状態遷移図エディタSTED(State Transition diagram EDiter)と機能ブロック図エディタHIPACE(HIerarchical PAth Chart Editer)をもつ。それらには、設計入力の容易化、正当性の早期保証を考慮して以下の機能を持たせた。

a)状態遷移図や機能ブロック図を構成する図形の追加、削除などの編集機能のほか、まとまった単位の状態遷移または論理回路を図面として登録し、これを検索、修正する機能を持つ。

b)記述言語の構文チェック、資源の対応チェックを行なう。このほか、連続する状態の同期クロックの一致（同相論理）などの時間設計ルールのチェックを行なう。

c)設計図面の図的記述を内部処理向きに中間言語に変換しデータベースに格納する。

（2）マイクロプログラム合成

STEDで入力された制御論理の機能仕様を、マイクロプログラム方式で実現する場合は、マイクロプログラムコンパイラMIC[14](MIcro Code generater)を用いる。

MICでは、マイクロコードとマイクロプログラムの走行環境（マイクロプログラム制御回路とその制御回路からの信号と被制御回路との接続）を自動生成するために、以下の手法を採った。

a)大規模な処理対象を扱うには、高速なマイクロコード最適化処理を実現する必要がある。ここでは、間接符号化方式*1のマイクロコード生成手順を導入して実用的時間内に語長の最短化を可能とした。

b)マイクロプログラムの走行環境を自動生成するには、マイクロプログラム制御回路の構成の決定および、被制御側の資源の制御点と制御用のマイクロコードの対応付けが必要となる。ここでは、マイクロプログラム制御回路のプロトタイプをもとに、自動生成されたマイクロコードの構成からプロトタイプのレジスタサイズ（語数、語長）を決定する。さらに、被制御側の資源の制御点と制御用のマイクロコードの対応をマイクロプログラム記述から抽出する。これらを機能記述で表現して論理合成ツールを利用して走行環境の自動生成を可能とした。

（3）論理回路合成

MICが生成したマイクロプログラム制御回路の機能記述は、レジスタトランスファレベルの中間言語で表わされデータベースに格納される。それは、さらにLSI-CADインタフェース変換LTRAN(design description Language TRANslater)によってLSI-CADの入力形式であるHSL-FX記述[15]に変換され、論理回路合成と論理検証に使われる。このほかに、与えられた機能仕様を布線論理で実現する場合、設計エントリツールから直接LTRANを経由してLSI-CADにつなぐことができる。

LTRANが行う変換には、記述モデルの変換とデータ形式の変換が含まれる。記述モデルの変換では、多相クロックを用いた状態遷移モデル

*1　マイクロコードでは、各種のマイクロ操作信号をフィールドで区切って、各フィールドが意味が他のフィールドの値によって決まる構成方式。コード化される。このときに、フィールドの

を、HSL-FXの論理ブロック間の接続を表す接続記述モデルに変換する。

図2に、状態遷移で表わした機能仕様とその中間言語表現およびHSL-FX変換結果を示す。

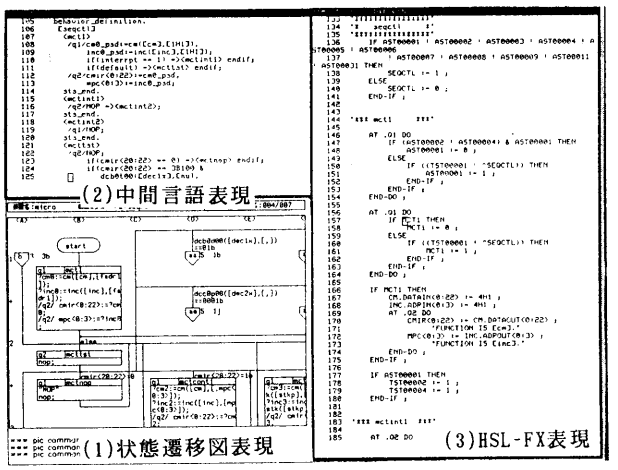


図2　設計エントリツールの表示画面

3　図形入力の基本方針

上記のシステム構成を前提に、設計エントリの図式化を実現するために、以下の方針を採った。

（1）トップダウン設計の支援

大規模なハードウェアの設計では、機能ブロックに分ける段階で全てのインタフェース（各機能ブロック間をつなぐ端子や信号など）を定めるのが困難なため、内部の詳細化を進める過程で必要なインタフェースを決めることが多い。そのため、下位の階層で指定したインタフェースを他の任意の階層で参照可能なグローバル信号の概念を導入する。これにより、下位の階層を詳細化しているときに新たにインタフェースが必要となった場合、そのインタフェースを参照する機能ブロック内で指定するだけで接続を完結させ、上位の階層にさかのぼってその指定を追加する必要がないようにする。

（2）記述形式の選択

大規模な回路の設計では、対象の複雑さと要求される回路規模・遅延時間の要求等に応じて設計情報の記述形式を選択できることが望ましい。

機能ブロック図では、構成単位である機能ブロックとして上位の機能モジュールから論理回路までの任意の詳細さの回路構成を指定できる。記述工数は詳細さに応じて増加するため、単純な回路構成の記述に適する。また、工数はかかっても実現（論理）回路に近いところまで詳細化する必要がある場合に適する。そこでは、転送路を介した資源間の接続を中心に表わして制御信号は陽に記述する必要がないようにする。これに対して、状態遷移図では、時間の経過とそれに対応する動作を記述する。そのため、制御シーケンスの記述や回路の詳細構造を意識しない場合に適する。そこでは、ソースの資源からデスティネーション側の資源へデータが移動する条件（制御）を中心に表わして、資源間の転送路は陽に表現する必要がないようにする。

また、この2種類の記述形式を使い分けて1つのハードウェアを設計できるように、設計結果を自動的に結合できるようにする。

（3）記述能力

計算機処理のため、標準化された記法を持つ。設計者の習得の容易さと記述誤りの回避のため、その記法は簡素にする。大量の設計情報の記述をコンパクトに記述するため、階層化ないしモジュール化を可能とする。多様な設計スタイルに対応できるように、動作の実行時刻の指定にクロック同期信号と非同期的に発生する信号の両者を扱えるようにする。さらに、クロック信号として単相と多相を使用可能とする。

4　状態遷移図記述支援

ここでは、STL/A(STLの状態遷移[Automaton]図対応部)の特徴である、多相クロックと階層記述を中心に、前記の課題の実現法を述べる。

4．1　記述モデル

一般の状態遷移では、唯一の状態が活性となって次々に遷移していく中で、1つの状態に属する動作は該当の状態が活性である時間に有効となる。これに対して、STL/Aでは動作と時間

の関係を以下のように拡張して多相クロックを扱えるようした。

すなわち、動作を時間に依存するものと時刻に依存するものとに分け、それぞれの有効時刻と有効時間を規定する。まず、時間に依存する動作(信号送出、機能回路のイネーブル、遷移)は、状態の同期クロックを含め該当の状態が活性中に生起するクロックの立ち上がり時刻に有効となり1マシンサイクル持続する。一方、時刻に依存する動作(エッジトリガの記憶回路への書き込み)は、状態の同期クロックを除く該当の状態が活性中に生起するクロックの立ち上がり時刻に有効とする。

図3に概念図を示す。この動作と時間の関係は、図4に示す回路モデルでハードウェア化できる。なお、マイクロプログラムの場合は文献[14]に示されているので、ここでは布線論理の場合の実現法を述べている。

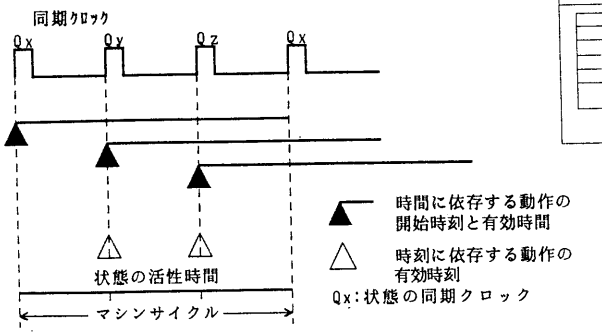


図3　動作の有効条件

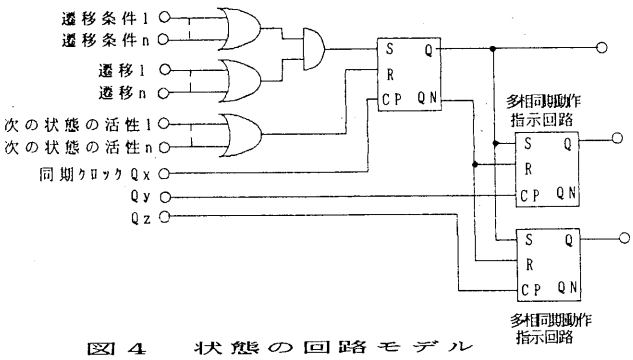


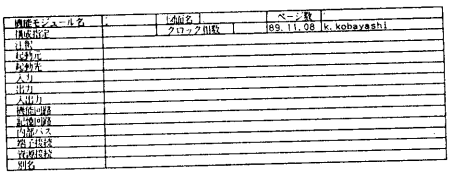
図4　状態の回路モデル

4．2　図記号の構成

状態遷移を図として表現する場合の最小限の構成要素は、『状態』と『遷移』およびそれらに付随する『状態の固有名』、『状態で実行される動作』、『遷移条件』である。この他に、『機能ブロック図とのインタフェースをとるための資源の構成情報』、『多相クロック動作を陽に表現するための状態と動作の同期時刻指定』、『階層化ないしモジュール化指定』をもうけた。表1に示す図記号を用いてこれらを記述する。

表1　ＳＴＬ(状態遷移図)の図記号

図記号	意　味
同期ｸﾛｯｸ　状態名 動作 ＜状態＞	ステージ内の1状態を表す。 (1)"同期ｸﾛｯｸ"は状態の同期時刻 (2)"状態名"は状態の固有名 (3)"動作"は状態内の複数の動作の指定 (4)状態内の動作は以下の構文で表される。 ●動作: /動作の実行時刻/ ﾃﾞｽﾃｨﾈｰｼｮﾝ:=ｿｰｽ; または、/動作の実行時刻/ 制御信号<-1; ●演算: 演算回路名([機能],[ｵﾍﾟﾗﾝﾄﾞ])
ﾏｸﾛ名 ＜マクロ＞	ステージ内のサブステージを表す。 (1)"マクロ名"はマクロの固有名
資源名 ＜多分岐＞	遷移条件の一括判断を表す。 (1)"資源名"は判断対象
詳細は下図 ＜インターフェース＞	ステージの管理情報と構造記述を表す。 ●管理情報:機能ﾓｼﾞｭｰﾙ名、図面名、ﾍﾟｰｼﾞ数 構成指定、ｸﾛｯｸ相数、作成年月日、作成者 ●構造記述:起動元、起動先、入力、出力、入出力、 機能回路、記憶回路、内部ﾊﾞｽ、端子接続、 資源接続、別名



4．3　記述法と実現回路

前述の図記号を用いた階層と多相クロックの同期動作は、以下のように回路に変換される。

(1)階層記述

STL/Aでは、階層記述にマクロを用いる。マクロで指定された機能は、独立した1つの回路として合成する。その場合、他の閉じた状態遷移の集合(ステージ)と同様に、状態を表わす制御レジスタの集合として合成できるが、マクロとマクロを呼び出している元のステージとの間の遷移を制御する回路が必要となる。すなわち、『マクロへの遷移』、『元のステージでの遷移の待

ち合わせ』、『マクロからの復帰』を制御する必要がある。この制御回路の生成は以下のようにして実現する。

①マクロへの遷移については、マクロの先頭の状態に対応する制御レジスタの入力条件として”マクロ呼出しの直前の状態の活性”を用いる。これにより、マクロの呼出しがあると、該当のマクロが遷移を開始することができる。

②元のステージでの遷移の待ち合わせとマクロからの戻りは対で実現する。すなわち、マクロ側の最後尾の状態に対応する制御レジスタが活性になると、その活性信号を受けて元のステージの遷移が復活する回路構成をとる。

（２）遅延

単相クロック式の状態遷移では、記憶回路へのデータ受渡し動作が自状態内で完結することが前提である。しかし、多相クロック式では、１マシンサイクル内に複数の状態が活性となるので、同期クロックの異なる状態間でのデータの受渡しが必要となる。そこで、仮想資源という概念を導入し、これをデータ受渡しの媒体とすることで多相クロック式のデータ受渡しを記述する。この仮想資源は、実回路としてインプリメントせず、クロックレベルで動作の遅延を陽に表現する手段である。

図５に概念図を示す。図６の記述例では、同期クロックの異なる２つの状態間のデータ受渡しを表わしている。状態mct1において、機能名memで指定される機能を持つ機能回路cmの入力fadr1に対する出力は、まず、仮想資源*Pcm0*に転送される（頭文字Pは仮想資源を指定する）。次の状態mcttstでは、*Pcm0*を介して機能回路cmの出力が資源cmir<0:22>に転送される。

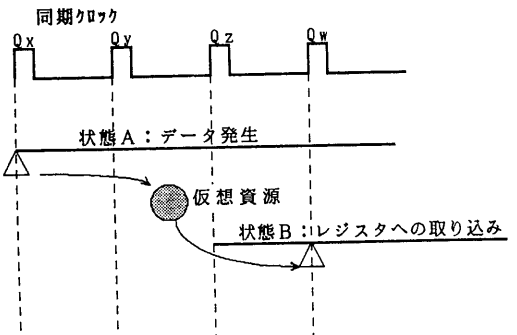


図５　動作の遅延

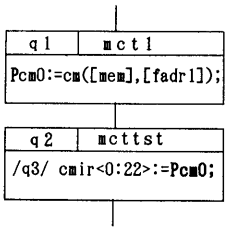


図６　遅延の記述例

5　機能ブロック図記述支援

STL/B（機能ブロック[Block]図対応部）を特徴付ける、グローバル信号の実現法、状態遷移とのリンク法に付いて以下に説明する。

5．1　グローバル信号の実現法

グローバル信号を実現する方法は、信号の参照可能範囲を全階層とする方法と、特定の階層内に制限する方法とに分けられる。前者では、分担設計時に信号名の重複を避けるため、信号名の一括管理が必要になる。その制約は、できるだけ機能ブロック毎に閉じて設計を進めたい分担設計には適さない。そこで、階層間にわたって参照される信号を参照範囲の大小によって２つに分け、信号名の管理を簡便化する以下の方法を採った。

(1)全階層共通のグローバル信号と、特定の階層以下で有効なセミグローバル信号を用意する。

(2)グローバル信号は、信号名の頭にグローバル信号の指定記号をつけて、<ｸﾞﾛｰﾊﾞﾙ信号指定子><信号名>の形式で指定する。セミグローバル信号は、有効範囲を指定する名称をグローバル信号の形式の頭につけて、<有効範囲識別子><ｸﾞﾛｰﾊﾞﾙ信号指定子><信号名>の形式で指定する。<有効範囲識別子>は、セミグローバル信号が有効な最上位の階層で指定する。

(3)階層の木構造に属さないマクロ内では全階層共通のグローバル信号のみを許す。

図７に記述例と階層を展開した例を示す。同図では、図面F_1にグローバル信号#y、図面F_3とF_4にセミグローバル信号f#yが使われている。ただし、"#"をグローバル信号指定子、"f"を有効範囲識別子としている。fは図面F_2で指定されており、F_2を詳細化した図面内で

有効であることが規定される。

このように、有効範囲識別子をもうけたことにより、以下のようにトップダウン設計向きの記法が実現できた。

すなわち、セミグローバル信号が必要となった図面内で、信号名に有効範囲識別子を付けるだけで指定できるため、予め上位の階層で全信号を指定する必要がない。1つのセミグローバル信号が指定されている階層内に、更に別の有効範囲のセミグローバル信号を指定する場合、新たに有効範囲識別子を定義すれば対応する信号が一意に定めることができる。詳細化の過程で信号の有効範囲の変更が生じた場合、有効範囲識別子の定義図面を変更するだけでよいので、有効範囲の変更が容易に行える。

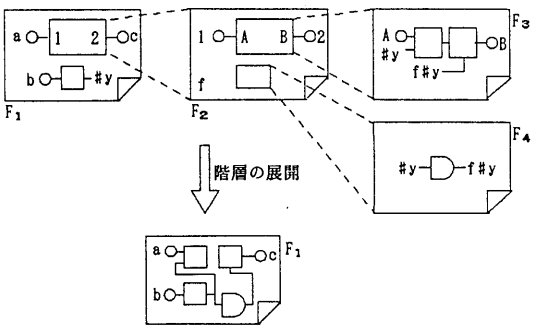


図7　グローバル信号

5．2　状態遷移とのリンク法

機能ブロック図で定義される資源間の接続には、原則として接続を制御する信号を表さない。一方、状態遷移図では、資源間の接続はデータ転送のソースとデスティネーションとして間接的に指定される。この2種類の図面で定義された資源間の接続は、次のようにして回路に合成される（図8参照）。

まず、機能ブロック図で定義されたデータパス回路の部分を論理回路に合成するときに、接続を持つ資源間をマルチプレクサでつなぎ、マルチプレクサの制御信号にグローバル信号を生成する。このグローバル信号には、対となる資源毎に固有の名称を付与する。一方、状態遷移図で定義された制御回路では、論理回路に合成するときに、各状態を表わす記憶回路の出力に先と同様にしてグローバル信号を生成する。この後、データパス回路と制御回路を統合して、1つの機能モジュールを構成する回路とする。この統合時に、転送のソースとデスティネーションに対応して固有に付与された信号同士がつながれ、制御回路からの制御信号が転送路を制御する回路が作られる。

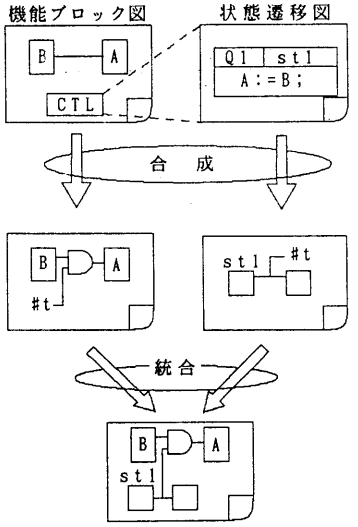


図8　機能ブロック図と状態遷移図のリンク

6　評価

6．1　図的入力の効果

設計仕様を図的入力する主な効果には、以下がある。

(1)可視性の効果：動作の並列性、動作間のつながりなどが直感的に理解しやすいため、設計情報の解析、保守および設計者間の情報交換の容易化が図れる。

(2)共用化の効果：設計図面と計算機入力用の情報が同一であるため、記述工数の削減とバグ混入の排除が図れる。

この内、定量化が可能な『共用化の効果』を10種類の制御回路をもとに評価した。

ここでは、記述工数は記述量に比例すると仮定する。図的入力の記述量にはテキスト形式で表現される中間言語の行数を用いる。設計図面

をあらためて計算機入力用の記述言語で書き換える従来手法の場合の記述量としては、LTRANにより変換されたHSL-FXの行数を用いる。

ＳＴＬを用いた場合の記述量は、１０種類の試行例の合計で765行であった。一方、従来手法のように設計図面の作成と計算機入力を行う場合には、両者の記述工数が必要となる。設計図面の作成工数は本手法の場合と等価とすると765行分の工数となり、計算機入力用は2202行分の工数であった。この結果、本手法によれば約1/4(=765/(765+2202))の工数で済むことがわかる。

6．2　ＳＴＬの記述範囲

ＳＴＬでは、組合せ回路についてはゲートレベルの回路は記述しない。したがって、ＳＴＬで記述した仕様を論理シミュレーションとＬＳＩのレイアウト設計につなぐには、ゲートレベルの回路をあらためて定義する必要がある。

通信処理用のプロセッサの一部の機能について評価した結果、ゲート換算で９割の回路がＳＴＬの記述対象であった。

7　あとがき

制御回路の設計支援を目的とした状態遷移図記述支援ツールSTEDとデータパス回路の記述支援ツールHIPACEを持つ機能設計支援システムを構築した。本稿では、この２種類の図的な設計エントリの実現法を報告した。従来の図的設計エントリツールに比べて、大規模ハードウェアの設計をトップダウン的に進められるように、STEDには、多様な時間条件の記述能力と階層記述の能力を、HIPACEには、グローバル信号の概念の導入、状態遷移図とのリンク等の特徴を備えた。

現在インプリメントしたツールは、設計入力の図式化を図ったものであるが、出力側の図式化すなわち、変換結果を論理回路図として出力し設計者の解析、評価を容易化することが今後の課題である。

「謝辞」　本システムの開発に当たり、ご指導、ご助言をいただいた、交換システム研究所　浜田リーダおよび関係者各位に深謝致します。

「参考文献」

(1)M.C.McFarland, et al.:"Tutorial on High-Level Synthesis";25th DA Conf. pp.330 (1988)

(2)来山、下出、横尾他：機能図展開システムについて　情処技報　DA46-8 (1989)

(3)中村、飯島、今井他：機能図入力システムにおける回路図面生成　情処全大　2R-4 (1986)

(4)鈴木、薄井、国友他：VLSI用階層論理設計システム　情処全大　4H-7　(1985)

(5)P.J.Drongowski et al. :"A graphical hardware design language" ;25th DA Conf. pp. 108 (1988)

(6)G.Estrin:"SARA in the design room";Proc. of the 1985 ACM CS Conf. pp.1 (1985)

(7)C.U.Smith et al.:"An Architecture Design and Assessment System for Software/Hardware Codesign"; 22nd DAC pp.417 (1985)

(8)G.ODAWARA et al.:"A Symbolic Functional Description Language"; 21st DA Conf. pp. 73 (1984)

(9)D.Drusinsky et al.:"Using Statecharts for Hardware Description";ICCD'87 pp.162 (1987)

(10)U.G.BAITINGER et al.:"The MEGA System :an Automated Methodology for Semi-custum VLSI Chip Design"; ICCAD'85 pp.97 (1985)

(11)H.Soderstrom:"STDL:a Graphical Behaviour Language"; ICCAD'85 pp.353 (1985)

(12)深沢、山岸、関根：図形入力による状態系設計支援システムの開発構想　信学技報 VLD89-22 (1989)

(13)O.Karatsu,T.Hoshino:An Integrated Design Automation System for VLSI Circuits; IEEE Design and Test of Computer, Vol. 2 pp.17 (1985)

(14)小林、脇村、岡田：論理設計自動化アプローチにもとづくマイクロコードジェネレータ　信学技報　CAS88-109 (1988)

(15)T.Hoshino,O.Karatsu,T.Nakashima:"HSL-FX:A Unified Language for VLSI Design";CHDL85, (1985)