設 計 自 動 化 60－3
（1991．12．12）

パイプライン・プロセッサ向き論理記述言語の考察

清水嗣雄　黒田理香子　影山直洋　庄内　亨
（株）日立製作所

本論文では新しい設計用論理記述言語B^2DL (Block-diagram and Behavior oriented Description Language) の言語仕様について述べる。この言語の設計思想は高性能VLSIや計算機の設計において用いられるパイプライン制御方式を簡潔に表現しうるようにすることにある。この言語はVLSIや計算機の開発の機能設計段階において使用すること、さらに自動論理合成システムへの入力として使用されることを念頭においている。B^2DL の特徴は以下の３点である。

（１）高度なパイプライン計算機の論理設計手法に基づく特別な記述形式を持っていること。
（２）自動論理合成システムに対する特別な記述形式を持っていること。
（３）簡潔な記述形式であること。

Logic Description Language for Pipeline Processors

Tsuguo Shimizu　Rikako Kuroda
Naohiro Kageyama　Toru Shonai
Hitachi, Ltd., Kokubunji , Tokyo 185 , Japan

This Paper describes the specifications for the new hardware description language B^2DL (Block-diagram and Behavior oriented Description Language). The design philosophy of this language is to incorporate hardware designers' knowledge for high performance VLSI and computer design. This language is intended to be used for a functional design in VLSI and computer development, and for input description for automated logic synthesis systems. B^2DL ' s specific features are :

(1) Incorporation of special description formats based on a logic desing methodology for highly pipelined processors,
(2) Incorporation of special description formats for automated logic sysnthesis system, and
(3) Simple description format.

1. 緒言

近年，VLSIプロセッサ・計算機の論理設計自動化のニーズの高まりと同時に，その自動化に必須なハードウエア記述言語あるいは論理記述言語の議論が盛んになってきている．論理設計記述言語の研究は1960年代からおこなわれており，ISPS[1)]，CDL[2)]，DDL[3)]，AHPL[4)]，CONLAN[5)]等多数の発表され，また1980年代に入り，EDIF[6)]，VHDL[7)]，UDL/I[8)]等の設計用記述言語の標準化活動が活発になり，現在も続行されている．これらの言語は多くの目的を持っており，例えば自動論理合成システムへの入力，あるいはシミュレーションシステムのための回路記述，ハードウエアシステムの形式的記述等に使用される．

いづれにしても，上記の要求を満たすために設計記述言語はハードウエアシステムの特性，設計方法，処理系と密接に関連しているべきである．設計手法との関連で言えば，有限状態機械をモデルとする状態遷移記述を取り入れた言語が知られている．

一方，計算機やＶＬＳＩプロセッサの設計の性能を向上される方法として，パイプライン向け制御が広く用いられている．パイプラインアーキテクチャは今後とも高性能実現のために必須なハードウエア方式であり，設計用記述言語としてこれに適した言語形式を取り入れることは極めて重要であると言えよう．

本論文ではパイプラインアーキテクチャを念頭においた設計用論理記述言語 Block-diagram and Behavior oriented Description Language（B^2DL）の設計思想および言語仕様の概要，さらに設計手法に基づく特殊なオペレーションについて述べる．

2. パイプライン型計算機の構造と動作

パイプライン制御とはある一連のデータ処理動作を幾つかのステージに分割し，各ステージを並列に実行することで計算機の高性能を実現する方式である．

パイプライン制御には図１に示すような命令レベルのパイプラインや演算器レベルのパイプライン等細かく分類すれば多様なパイプライン制御方式がある．

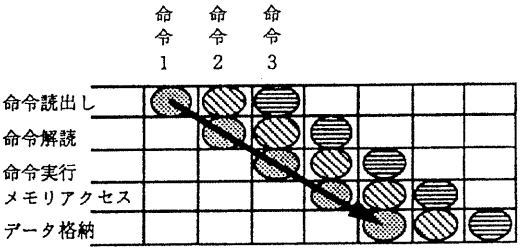
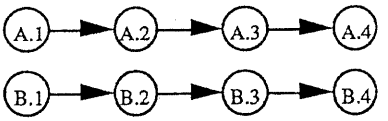
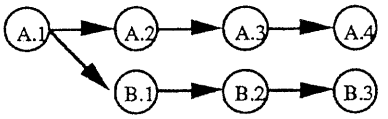


図１　命令レベルのパイプライン

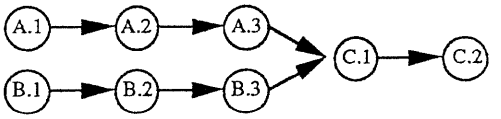
パイプラインプロセッサの動作には以下の特徴がある：ここであるオペレーション（たとえばある命令解読から実行まで，あるいは記憶装置に対するデータの書き込み・読み出し処理など）を実行するに当たって，一連のパイプライン・ステージの連続によってなされる処理をシーケンスと呼ぶことにする．図３(a)の(A.1)～(A.4)が一つのシーケンスを表す．



(a)　パイプラインの平行動作



(b)　パイプラインの分岐動作



(c)　パイプラインの合流動作

図３　パイプライン動作の形態

・パイプライン動作においては，起動条件さえ成立すれば，シーケンスが毎サイクル起動される，すなわち，最大1サイクルのずれで同一オペレーションが同時に実行される．（図1参照）
・一つのオペレーションは通常複数のシーケンスから構成されることが多い（図3(a)）．
・一つのオペレーションの実行において，一つのシーケンス動作が複数のシーケンス動作に分岐したり，あるいは複数のシーケンスが一つのシーケンスに合流したりすることがある（図3(b),(c)）．
・複数のオペレーションにおいて同一シーケンスが実行されることがある．

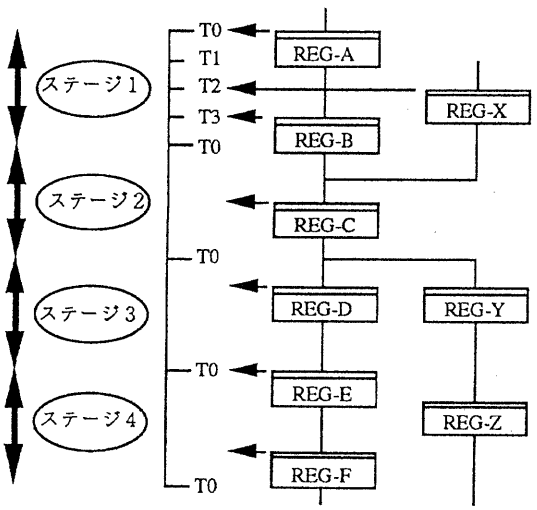


図2　パイプラインの実現方法

パイプライン制御プロセッサの骨格をなすデータパスは以下のハードウエア構成要素からなる：
・レジスタ，メモリ
・機能論理要素(セレクタ，ALU，加算器等)
・入出力端子．
これらのハードウエア構成要素を制御するための制御論理（以下データパス制御論理と称する）には以下に示すものがある．
・レジスタセット論理
・制御フリップフロップ(以下FFと記す)
・レジスタ状態管理論理（ホールド論理）
・メモリ制御論理
・機能論理対応の制御論理

データパス制御論理は，パイプライン制御の特性である，データの並列転送を効率良く実行するため，レジスタを中心とする各ハードウエア構成要素が効率よく動作しうるように構成されなければならない．特にデータパス上の分岐点や合流点において排他制御，優先制御あるいは待ち制御のための制御論理生成が重要である．

論理記述言語の設計に当たっては，これらの制御論理に対応する記述を可能なかぎり簡潔にすること，すなわちパイプライン動作に伴う多様な動作を簡潔に表現することが課題となる．

データパスを構成する要素の中でも重要なものはレジスタである．パイプラインステージは一般にそのプロセッサの基本単位時間であるマシン・サイクルを基本とするが，その実現においては，レジスタをステージの区切りあるいはタイミングの区切りとして用いる．例えば図2に示すように4ステージからなるパイプラインを実現するハードウエア，特にそのデータパスは複数のクロック信号に同期して動作するレジスタによって形成される．このときレジスタは必ずしも想定しているパイプラインステージに1対1ではなく，時間制約等によりきめ細かく設定される．

本稿においてはこのレジスタの動作，すなわちレジスタ間のデータ転送に着目し，かつレジスタを対象とする制御論理をどのように記述するかを検討する．

3．　パイプライン動作の論理動作記述形式

本章はこのレジスタの動作，すなわちレジスタ間のデータ転送に着目した論理記述言語 B^2DL について述べる．

3．1　B^2DLの設計思想

B^2DL言語を設計する際の基本思想はパイプライン動作に伴う制御論理に関する記述を簡潔に記述可能とすることにある．

これを実現するために

・パイプライン動作に伴う並列，時系列動作を容易に記述可能とする構文を導入する

ことを基本とし，さらに

・パイプライン動作の基本であるシーケンスを操作しやすくするために，データ転送動作記述とシーケンス制御記述の区別

・シーケンス動作参照関数の導入

を行った．一般に行われている階層設計記述の導入，論理機能を表現するための基本関数の充実も言語の基本機能として実現した．

論理設計者はパイプライン計算機の構造や動作を表現するのにブロック図やタイムチャートを頻繁に使用する．ブロック図はハードウエアのデータ構造に関する情報を含む．タイムチャートはハードウエア要素間のデータ転送に関する情報を含む．この情報は目的とするシステムで実行されるオペーレションの実現方法に関する情報を含む．

このようなパイプライン動作（シーケンス）は論理回路中において多くの場合，複数存在すると考えたほうがよい．しかもこれら複数シーケンスは起動条件が成立すれば，他のシーケンスの状態にかかわりなく動作することが条件である．すなわちシーケンスは相互に並列動作するものである．

一つのシーケンスに着目すると，図4に示すように，パイプライン動作の特性から言って，数マシンサイクルにまたがる複数の時系列的なデータ転送動作から構成されることが多い．一方ある時点を取ると，データ転送は一つだけではなく，一般に複数のデータ転送が行われると考えるほうが自然である．

このような観点から，B^2DLではパイプライン動作の基本であるシーケンスを

「SEQUENCE - END」

を基本単位として記述する方法を採った．これをSEQUENCE文と呼ぶ．

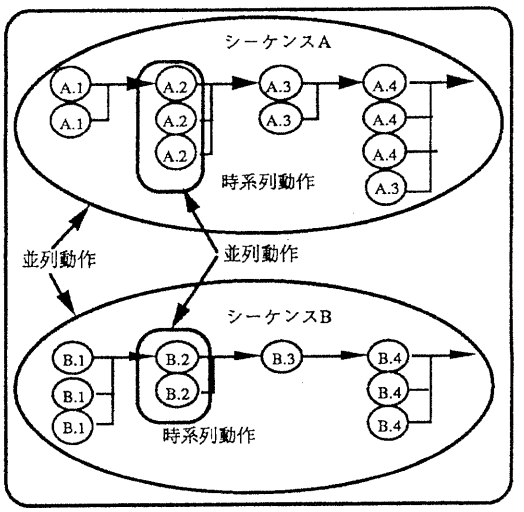


図4　B^2DL言語の基本概念

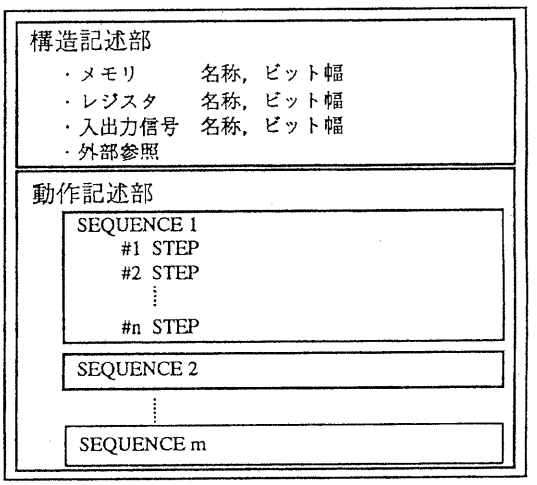


図5　B^2DL言語の基本構成

SEQUENCE文では論理回路の一連の動作を定義する．各SEQUENCE文は付随する名称で区別され，相互に並列に起動，実行されるとみなされる．各SEQUENCE文は起動条件が真の時のみ実行される非手続き型記述方法を採用している．一方，シーケンス内部のデータ転送文等は，記述順に従って実行されることを前提とした手続き型記述となっている．

同一時刻で実行されるべきデータ転送文はまとめて記述可能であり，これを「STEP文」と称する．STEP文で記述されたデータ転送文は全て並列に実行される非手続き型記述である．必要に応じて各STEP文には実行時刻を指定できる．

B2DLでは，このように設計対象の動作状態に応じて記述を容易にすべく，

非手続き型記述

→ 時系列的動作

→ 非手続き型記述

このような一種の入れ子形式となっている．

3．2 言語仕様

B^{2}DL言語では論理設計での基本的ドキュメントであるブロック図およびタイムチャートに記述されている論理情報を各々構造記述部分，動作記述部分に記述する（図５）．本節においてB2DLの構文の概要を示す．

（1） 構造記述

構造記述部ではハードウエア構成要素，例えばレジスタ，メモリ，入出力信号端子等の名称，データ幅等を記述する．特に組み合わせ論理に対し，ブール式表現，真理値表表現により構造記述の簡便化を可能としている．

（2） 動作記述

動作記述はSEQUENCEの集合である．

SEQUENCEは以下に述べる項目からなっている．

シーケンス名：：SEQUENCEの識別名称．これはB2DL記述の中でユニークでなければならない．

起動条件：SEQUENCEに記述されているこの起動条件が満足されればSEQUENCEが起動される．

各SEQUENCEはデータ転送を基本とするSTEP文の集合である．STEP文は以下の要素からなる．

ステップ番号：データ転送が実行される順序を表している．いる．

クロック：データ転送動作の実行時間を表す．

```
MEMORY :メモリの属性を記述する
    (名称，ワード構成，メモリ容量，アドレス，データ)
REGISTER :レジスタの属性を記述する
    (名称，ワード構成等)
INPUT :入力信号の属性を記述する
    (名称，ワード構成，他の入力信号との関連)
OUTPUT :出力信号の属性を記述する
    (名称，ワード構成，他の出力信号との関連)
SIGNAL :内部信号の属性を記述
    (名称，ワード構成等)
CONNECT :ブール式形式で論理ネットワークを記述する
ARRAY :真理値表形式で組み合わせ論理を記述する
EXBLOCK :外部で定義されたBLOCKを指定する
    (名称，入出力関係)
EXMODULE 外部で定義されたMODULEを指定する
    (名称，入出力関係)
```

図６ 構造記述部の概要

```
1．シーケンス文
   SEQUENCE: <シーケンス名>;
     '<<'<起動条件>'>>'<ステップ文>{<ステップ文>}
   END;

   <シーケンス名> ::= <単純名>
   <起動条件> ::= <１ビット幅単純式>| CALLED

2．ステップ文
     #<ステップ番号>:<クロック>[<ステップ記述文>]
     <ステップ記述文>::=<データ転送文>|<条件付転送文>

3．単純転送文
   <転送先>:=<式>[<転送制御>]

   <転送制御>::=   HOLD FOR(<サイクル数>)
                 | HOLD UNTIL(<ビット幅単純式>)
                 | HOLD WHILE(<ビット幅単純式>)

4．条件付転送文
   IF <条件> THEN <データ転送文>{<データ転送文>}
     [ELSE <データ転送文>{<データ転送文>}]
   END-IF;
```

図７ 動作記述部の概要

B^{2}DLには２種類の動作記述方法がある．

(i) データ転送動作：これは一つのハードウエア要素から他のハードウエア要素へのデータ転送を記述するものである．転送先要素は通常はレジスタである．ただし転送元は任意の要素である．この記述は自動生成シ

ステムにおいて制御論理を生成する際の基本的な情報となる．

(ii) 制御動作：データ転送の流れを制御するための制御ステートメントが用意されている．これに属するものはSTART文とGOTO文である．これらの文はある条件のもとで他のSEQUENCEを起動したり，他のSEQUENCE やSTEPへ分岐することを可能とする．（もしくは他のSEQUENCEの中のSTEPへ分岐することもありうる．）これらの制御動作文により複雑な並列動作が容易にか記述可能となり，オペレーション間の同期を取ることを容易に記述できる．またいくつかのオペレーションにおいてその動作記述の中に共通する部分があるとき，これらの共通部分は一つのモジュールとして，あたかもプログラミング言語のサブルーチンのように表現することが可能である．

（3）関数

B^2DLにおいては２種類の関数，すなわち基本関数と拡張関数があり，これらにより簡潔な記述が可能となる．関数の一部を以下に示す．

表１　関数

種類		タイプ	
基本関数	演算器	ABS	
		ADD1	SUB1
		ADD	SUB
		CMP	EQU
	論理関数	NOT	
		AND	OR
		XOR	
		NAND	NOR
		DEC	ENC
		SEL	
拡張関数		HOLD	
		BUSY	
		EXEC	

拡張関数は論理合成システムを前提とした関数であり，また合成される論理はパイプラインアーキテクチャを前提としている．拡張関数の機能は以下のようである．

(a) EXEC 文：指定シ-ケンスの指定ステップが実行されるクロック周期の間は1，他の時は0を出力する．ステップの指定は複数可能である．

(b) BUSY 文：シ-ケンス内で用いられているレジスタが有効なデ-タを保持しているか否かを出力する．有効なデ-タを保持しているときは，論理値1を出力し，保持していないときは論理値0を出力する．

(c) ＨＯＬＤ：この関数はレジスタに対するホールド論理を生成することを要請するものである．自動論理生成システムは目的とするハードウエア要素に関連したデータ構造と制御論理を生成する．

（4）動作記述の意味

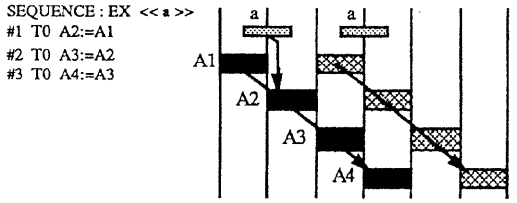
SEQUENCE文およびSTEP文による動作記述の意味をタイムチャートを用いて説明する．

(a) SEQUENCE文はその起動条件（図８(1)）が成立すれば，それに含まれる一連のSTEP文が，基本的にはステップ番号順に実行される．この例ではレジスタA1→A2→A3→A4へ順次データが転送される．

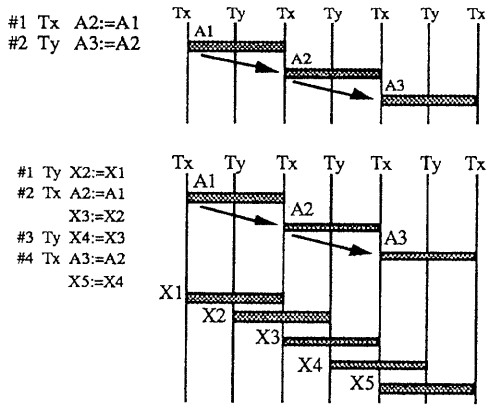
(b) STEP文に現われるデータ転送文の基本は単純転送文である．STEP文中のクロックは指定された転送動作，図８(2)上段の例ではレジスタA1からレジスタA2への転送，すなわちレジスタA2へのデータセットが完了する時点を表している．

(c) 図８(3)に転送文のうち，条件付きの例を示した．この例ではレジスタA1からのデータ転送先が条件ａによって変わる場合を示しているが，then 節あるいは else 節において動作制御文（START 文や GOTO 文）を書くことも可能である．また本稿では詳述しなかったが，複数条件付き（いわゆる CASE 文に相当する）条件文も用意されている．

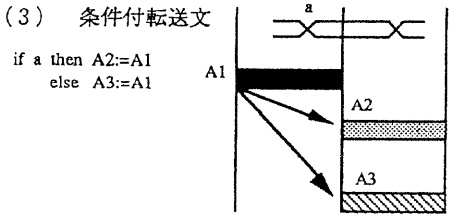
（1） シーケンス文



（2） 単純転送文



（3） 条件付転送文



（4） 転送文に付随する制御（HOLD FOR)

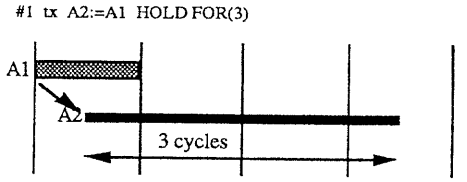


図8 動作記述の意味

(d) データ転送文には転送されたデータの転送先でのデータ保持条件に関する記述が可能である．図8(4)ではHOLD FORを使用した例を示している．本例ではデータをレジスタA1からA2に転送後，レジスタA2においてデータを3サイクル保持することを示している．したがって論理合成システムにおいてはレジスタA2のための制御論理として，計時論理（カウンタなど）およびデータの追突防止論理を合成することが必要である．

4． 記述例

図9示すデータパス構造とタイムチャートに示される論理を対象とする簡単な記述例を図10に示す．図9に示すデータパスはアクセル時間2サイクルのメモリ（MEM）に対する論理を示している．レジスタREGAに対し毎サイクル，アクセス要求が発生しうるものとする．もし，要求1と要求2が連続した時間に発生したとすると，タイムチャートに示すように，要求2のデータはレジスタREGAにおいて待たなければならない．

図10の記述例では2通りの方法でこの動作を記述している．記述例1では，複数のメモリアクセス要求が競合しない場合（SEQUENCE : FETCH）と競合する場合（SEQUENCE : OVER_RUN）に分けて記述している．この例においては要求の競合をあるSEQUENCE : FETCH 実行常態を拡張関数EXECを用いることにより調べている．

記述例2では，動作の実行常態ではなく，レジスタREGAとREGBが同時にビジーになるか，と条件を拡張関数BUSYを用いて調べており，この結果を用いることによって，記述例1の場合より簡潔な記述となっている．

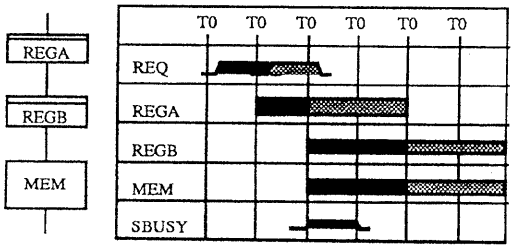


図9 記述例の対象

(1) 記述例1

```
CONNECT : SBUSY=EXEC(OVER_RUN:1)

SEQUENCE : FETCH ; << ^EXEC(FETCH:1) & REQ >>
  #1 : T0  REGA := DATA ;
  #2 : T0  REGB := REGA ;
  #3 : T0  ;

SEQUENCE OVER_RUN ; << EXEC(FETCH:1) & REQ >>
  #1 : T0  REGA := DATA ;
  #2 : T0 ;
  #3 : T0  REGB := REGA ;
  #4 : T0  ;
```

(2) 記述例2

```
CONNECT : SBUSY=BUSY(REGA) & BUSY(REGB) ;

SEQUENCE : FETCH ; << ^SBUSY & REQ >>
  #1 : T0  REGA := DATA ;
  #2 : T0  REGB := REGA HOLD_FOR(2);
```

図10　記述例

5．結言

新しい論理記述言語B^{2}DLの設計思想，言語仕様の概要について述べた．

本言語は大型計算機やマイクロプロセッサに必須なパイプラインアーキテクチャの論理合成を念頭においた論理合成向き記述言語である．これを実現するため，（1）パイプライン動作に伴う並列，時系列動作を容易に記述可能とする構文の導入，（2）パイプライン動作の基本であるシーケンスを操作しやすくするための，データ転送動作記述とシーケンス制御記述の区別，（3）シーケンス動作参照関数の導入，を行っている．

この言語ではブロック図やタイムチャートなどの設計ドキュメントから容易にB2DLへの変換が可能であるという特長を持っている．

本言語によれば，パイプライン動作に必要な制御論理を簡潔に記述可能であるため，論理仕様を記述するのに必要なステートメント数が非常に少なくてすむ．

論理記述言語としては多様な設計手法に対応しうることが必要である．本言語では状態遷移を記述することは可能ではあるが，必ずしも簡潔に記述できない．パイプライン記述と状態遷移記述との整合性が今後の課題である．

参考文献

1) M.Barbacci, 'Instruction Set Processor Specifications(ISPS) : The Notation and Its Applications', IEEE Trans. on Comp[., Vol. C-30, No.1, JAN., 1981, pp24-40
2) Y.Chu, "An ALGOL Like Computer Design Language', Communication of the ACM, Vol.8, No.10, Aug., pp607-615
3) J.R.Duley et al., 'A Digital System Design Language (DDL)' , IEEE Trans. on Comp., Vol.C-17, No.9, pp950-961
4) F.J.Hill et al., 'Digital Systems : Hardware Organization and Design' John Wiley &Sons
5) R.Piloty et al., 'CONLAN Report', Lecture Notes in Computer Science 151, Springer-Verlag, 1983
6) "EDIF specifications version 110" 1985
7) IEEE Standard VHDL Language, Reference Manual, IEEE, 1988
8) LSI設計用記述言語仕様，日本電子工業振興協会，LSI設計用記述言語標準化委員会
9) T.Shimizu et al., " A Logic Synthesis Algorithm for The Design of a High Performance Processor", Int. Symp. Circuits and Systems proc., 1985, pp.407-410
10) T.Shimizu et al., "High Level Logic Synthesis Algorithm with global Minimization Porocess" Int. Conf. Computer Design 1986, pp.15-19