設計自動化 39－3
（1987. 10. 15）

動作設計段階でのハードウェア記述に関する基礎的な検討

多喜康朗　パラシオス・アルベルト　角山正博　内藤祥雄
長岡技術科学大学

本稿では，仕様と動作設計の間の論理検証が容易なハードウェア記述言語（ＨＤＬ）の基礎を検討し，さらにそれらを用いた検証方法について検討を行う．

まず論理装置の仕様を記述するための形式的な仕様記述言語バッファを提案する．次にバッファによって記述された仕様と，その仕様に基づいたＨＤＬによる記述との間の論理検証を容易に行うために，論理装置のデータ処理部と制御部を分離して記述し，各々を別々に検証する方法を提案する．またそのためにＨＤＬが有すべきデータ処理部と制御部を記述するための各々の構成要素を示す．

最後に仕様に記述された制御構造とそれに基づいて構成された論理装置の制御部との間で論理検証の基礎について検討する．ここでは仕様の制御構造と制御部の動作を各々入力と出力を持つ２つの有向グラフとして表わし，両者を比較することによって正当性を調べる手法を示す．

A FUNDAMENTAL EXAMINATION OF HARDWARE DESCRIPTIONS AT THE BEHAVIORAL DESIGN LEVEL

Yasuro TAKI, Alberto PALACIOS, Masahiro TSUNOYAMA, Sachio NAITO
The Technological University of Nagaoka
Kamitomioka,Nagaoka-shi,Niigata,940-21 Japan

We examine the fundamentals of a Hardware Description Language (HDL) which facilitates the logical verification between the specification of a system and its behavioral design described in this language. Then,we examine a verification method that applies to this HDL.

First,we present a formal Specification Description Language ,Buffer, for digital systems. Next, we propose a method of verification that contrasts the specification given in Buffer to the HDL description based in that specification. For facilitating the verification , a system is described in two sections(by the HDL) : the Data Proccessing and the Control Sections, and then the process of verification is carried out for each of them , separately. We also show each of the structural elements that a HDL must possess to describe the Data Proccessing and the ControlSections.

Finally,we examine the fundamentals of a logical verification method that applies to the control sequence described in a specification (in Buffer) and the behavior of the Control Section of the digital system described in the HDL. For this purpose, we express the control sequence of the specification (Buffer) and the behavior of the Control Section (HDL) as two directional graphs, and show a method that let us decide their correctness by comparing them.

１．はじめに

近年ＶＬＳＩに代表される論理回路の高集積化に伴い，これらの設計を支援するＣＡＤ技術の確立が益々強く望まれてきている．この中でも特に論理設計の分野においては，装置の大規模化や多様化に伴って増大する設計の誤りを早期に発見し，損失を未然に防止するための技術の開発が要求されている．

論理設計を支援するシステムにおいてはほとんどの場合，設計した回路をハードウェア記述言語（ＨＤＬ）を用いて表現している．

ＨＤＬは，記述するハードウェアのとらえ方によって，幾つかの記述レベルに分けることができる．記述するハードウェアの抽象度の高い順，すなわちトップダウンの設計の流れに従う場合には，次のように３つに分けることができる．

１）論理装置の仕様(要求)を記述する仕様記述レベル
２）論理装置内部の動作を記述する動作記述レベル
３）ゲートの接続関係を表現する接続記述レベル

これらの中で，設計の最初の段階で用いられる装置の仕様または要求は，従来自然言語を用いて記述されてきた．しかし自然言語による記述では，計算機での処理が困難であり，意味表現にもあいまいさが常に存在する．そこで仕様を書くための形式的な仕様記述言語を用いて仕様を記述すれば，計算機での処理が容易になり，仕様の記述の中のあいまいさによる設計の誤りも防ぐことができる．仕様に基づいて論理装置の内部の具体的な動作を記述する言語，つまり動作記述レベルにおけるハードウェアを記述するための言語は，通常ＨＤＬと呼ばれている．そこでここでも，仕様記述言語と区別して，動作を記述するための言語をＨＤＬと呼ぶことにする．

トップダウンによる論理設計における設計の誤りとは，先に述べた３つの記述レベルのうち上位のレベルの記述と異なる下位のレベルの記述が作成されることである．特に今日のように論理装置が大規模化，複雑化するに従って，このような設計の誤りの混入する度合は益々増大してきている．一連の設計作業の中で，仕様に基づいて論理装置の動作を定める設計作業は，設計過程の中では最も機械化が遅れている過程の一つである．これは，設計者の知識を活用して設計を行なうシステムが必要であるにも拘らず，実現が困難なことによるためであり，このため人手に頼る部分が多く，誤りも混入し易いことになる．このような設計の誤りを発見するためには，論理検証が必要である．しかし既に提案されている，異なるレベルでの記述を比較する設計検証方式の大部分は，動作記述レベルの記述及び接続記述レベルの記述の間の検証を行なうことを目的としており，動作記述と仕様の間の検証については，まだ有効な手法が見出されていないのが現状である．

そこで本稿では，仕様と動作記述の間の矛盾を発見する際に検証のし易いＨＤＬを作成することを目的とし，そのためのＨＤＬの構成法の基礎として，動作記述レベルにおけるデータ処理部と制御部の分離について検討する．またそれを用いた論理検証方法についても検討を行なう．

２．仕様記述言語バッファ

前述したように仕様に基づいて計算機上で効率的に装置の検証を行なえるようにするためには，形式的な仕様記述言語を開発することが必要になる．

さて仕様記述言語はまず，この言語で記述された仕様書を仲介として，仕様書を書いた人の論理装置に対する構想が正確に設計者に伝えられるように，装置に対する要求を正確に記述できることが必要である．更に装置の具体的な実現方法に依存しない仕様であるために，仕様の記述の中で用いられる表現は抽象的でなければならない．このような仕様書を用いることにより，設計者は自由に装置の構成を定め，設計を進めることができるようになる．

我々はこれらの条件を満たした仕様記述言語バッファを開発した．バッファは，論理装置をその実現の仕方ではなく，何をすべきかを中心として記述できること，及び記述形式が自然言語に近く修得し易いこと等の特徴を有している．

バッファの構文とその意味の概要については稿末の付録に述べてある．

３．論理検証の容易なＨＤＬ

３．１　データ処理部と制御部の分離

論理装置は，仕様で定められたデータが入力された時，それらに対して定められた加工を施し，その結果を出力する．このはたらきに着目して，論理装置の動作を分析すると，与えられたデータに対する処理を実行する動作と，それらの定められた処理を定められた順序で実行するように制御する動作の２つに大別することができる．同様にハードウェアは入力されたデータを加工する部分，即ちデータ処理部と，データの入力，出力および加工の順序を指定する部分，即ち制御部に分けることができる．従って，ハードウェアをこれらの２つの部分に分けてＨＤＬで表現し，各々を検証することにすれば，比較的容易に検証を行えるものと考えられる．

そこで論理検証の容易なＨＤＬを作成するために，論理装置を構成するハードウェアを大きくデータ処理部と制御部に分け，更にそれらに含まれるハードウェアをその動作に着目して分類することにする．

３．２　ハードウェア要素の定義

３．２．１　論理装置の定義

まずＨＤＬで記述することができる論理装置を次のように定義する．

［定義１］

Ⅰ）任意の数の入力と出力を持つ．
Ⅱ）全ての信号を０と１の２進数値で表現する．
Ⅲ）システム全体は１つのクロック信号に同期して動作する．
Ⅳ）１個の装置起動入力を持ち，この入力によって論理装置が動作を開始する．
次に論理装置の入力線と出力線を次のように定義する．
［定義２］　論理装置の外部からの任意の数の２値の信号を装置内部に取り込む線を入力線，論理装置の内部の任意の数の２値の信号を外部へ出力する線を出力線という．

３．２．２　データ処理部のハードウェア要素

データ処理部のはたらきは，外部からのデータを取り込む入力，入力されたデータを蓄えておく記憶，データに対する加工，加工されたデータを外部に出す出力に分けられる．データを記憶するためのフリップフロップあるいはレジスタを，ここではまとめて変数として次のように定義する．
［定義３］　以下の機能を有するハードウェアを変数という．
Ⅰ）データ入力，データ出力，クロック入力を持っている．
Ⅱ）クロック入力にパルス信号が加えられると，その時入力されていた値が，以後の出力となる．
またこの変数に新たにデータを記憶させる動作を転送という．またデータを加工するハードウェアを演算装置と呼び，次のように定義する．
［定義４］　以下の機能を有するハードウェアを演算装置という．
Ⅰ）入力と出力を持つ．
Ⅱ）現在の入力の値に対して現在の出力の値が一意に定まる．
この演算装置のはたらきを演算と呼ぶことにし，また論理装置における転送及び演算を処理と呼ぶことにする．なお，この他にハードウェア要素として，これらを結ぶ接続線やバス等が必要である．

データ処理部における入力，処理，出力の全ての動作は，制御部からの指令によって実行される．このようなデータ処理部に指令を与える信号を次のように定義する．
［定義５］　制御部から出力されデータ処理部に入力される指令を制御信号と言う．

３．２．３　制御部のハードウェア要素

制御部は通常仕様で定められた順序に従って，データの処理が実行されるように処理部に指示を与える．論理装置が大規模になるにしたがい，処理の順序も複雑になり，従って論理検証も困難になることが考えられる．そこで論理装置を分割して検証を容易に行うために，先に述べた仕様記述における処理の一単位であるFUNCTION（付録参照）に対応する処理を制御する制御装置を定めることにする．また先に述べたバッファによる仕様記述では，BEIDLE文（付録参照）によってFUNCTIONの実行順序が定められるようになっているが，これに対して制御装置の間では制御装置を起動させる信号の受渡しによって動作の開始と停止を行うことにする．この信号を遷移実行信号と呼ぶことにし，次のように定義する．
［定義６］　制御装置から出力され，制御装置に入力される，制御装置の動作を開始させる信号を遷移実行信号という．

次に制御装置への入力としては，論理装置外部からのｋ個の入力信号，データ処理部からのｌ個の信号，自身または他の制御装置からのｈ個の遷移実行信号及びｌ個の装置起動信号がある．これらを各々
XI1,XI2,･･･,XIk（XIi∈ {0,1} ,1≦i≦k），
XJ1,XJ2,･･･,XJl（XJi∈ {0,1} ,1≦i≦l），
C1,C2,･･･,Ch（Ci∈ {0,1} ,1≦i≦h），及び
C0
と表わすことにする．

また制御装置からの出力としては，論理装置外部へのｊ個の出力信号，データ処理部へのｒ個の制御信号及び自身または他の制御装置へのｈ個の遷移実行信号があり，これらを各々
ZO1,ZO2,･･･,ZOj（ZOi∈ {0,1} ,1≦i≦j），
ZP1,ZP2,･･･,ZPr（ZPi∈ {0,1} ,1≦i≦r），及び
C1,C2,･･･,Ch（Ci∈ {0,1} ,1≦i≦h）
と表わすことにする．

制御装置を構成する順序機械Ｍを次のように定義する．
［定義７］　Ｍ＝（Ｘ，Ｑ，Ｚ，δ，ω）
ここで，
Ⅰ）Ｘは次のようなｎ個の入力の集合である．
Ｘ＝｛ｘ1,ｘ2,･･･,ｘn｝，ｘi＝（XI1,XI2,･･･,XIk,XJ1,XJ2,･･･,XJl,C0,C1,C2,･･･Ch）（1≦i≦n）
とする．
Ⅱ）Ｑはｍ個の状態の集合である．これを
Ｑ＝｛ｑ1,ｑ2,･･･,ｑm｝（1≦m）
とする．
Ⅲ）Ｚは次のようなｓ個の出力の集合である．
Ｚ＝｛ｚ1,ｚ2,･･･,ｚs｝，ｚi＝（ZO1,ZO2,･･･,ZOj,ZP1,ZP2,･･･,ZPr,C1,C2,･･･,Ch）（1≦i≦s）
Ⅳ）δは遷移関数であり，δ：Ｘ×Ｑ→Ｑである．
Ⅴ）ωは出力関数であり，ω：Ｘ×Ｑ→Ｚである．

次に順序機械の状態の中で，この遷移実行信号を受け取るACCEPT状態と，遷移実行信号を出力するTRANSFER状態を次のように定義する．但しＸC0,ＸC1,･･･,ＸChは次の条件を満たすＸの部分集合である．
ＸCi＝｛ｘ＝（XI1,･･･,XIk,XJ1,･･･,XJl,C0,･･,Ci,･･,Ch）∈Ｘ｜Ci＝1｝（i∈ {0,1,･･･,h} ）
またＺC1,ＺC2,･･･,ＺChは次の条件を満たすＺの部分

集合である．

$ZCi=\{z=(0,0,\cdots,0,C1,\cdots,Ci,\cdots,Ch)\in Z \mid Ci=1\}\ (i\in\{1,\cdots,h\})$

［定義８］　次の条件を満たす状態$qa\in Q$をACCEPT状態と言う．

ある$XCi\ (i\in\{0,\cdots,h\})$の全ての要素$x\in XCi$に対して

$\delta(x,qa)=qx\ (qx\neq qa, qx\in Q)$

であり，且つ全ての要素$x\in\{X-XCi\}$に対して

$\delta(x,qa)=qa$

である．

［定義９］　次の条件を満たす状態$qt\in Q$をTRANSFER状態と言う．

ある$ZCi\ (i\in\{0,\cdots,h\})$の全ての要素$z\in ZCi$に対して

$\omega(x,qt)=z\ (x\in X)$

$\delta(\varepsilon,qt)=qa$

である．但しqaはACCEPT状態，εは全ての入力を表わす．

これらの定義から明らかなように，ACCEPT状態は遷移実行信号のいずれかが１であるときにのみ他の状態に遷移しうる待機状態であり，TRANSFER状態は同様に，遷移実行信号のいずれかを１にした後必ずACCEPT状態に戻る制御装置間の実行を制御するための状態である．ここではこれらの状態qa，qtは各々$qa=q1, qt=qm$であるものとする．なお順序機械MはTRANSFER状態以外の全ての状態において，遷移実行信号のいずれかが１である出力を出すことはないものとする．また制御装置から出力される信号の中で，制御装置に入力される信号は遷移実行信号以外にはないものとする．

このような順序機械を用いて定義された制御装置は，FUNCTIONで示される処理が行われていない場合には，ACCEPT状態にあり，このときそのFUNCTIONに定められた遷移実行信号が１である入力を受け取ると，状態遷移関数で定められた他の状態へ遷移を開始する．一旦遷移を開始した後では，各々の状態に対して定められた制御信号及び論理装置外部への信号を出力しながら遷移を続ける．そして全ての制御信号及び論理装置外部への信号の出力が終了したなら，TRANSFER状態に遷移し，定められた遷移実行信号を１にして再びACCEPT状態に戻り，待機状態に入ることになる．

また複数の制御装置が，同一の遷移実行信号によって起動することが定められている場合にその信号が１である入力を受け取ったときには，それらの全ての制御装置が同時に動作することになり，これによって，論理装置内部の並行動作を記述することができる．

３．３　ハードウェア要素を用いた論理装置の構成

ここでは前節で定義したハードウェア要素に基づく論理装置の構成について検討する．

まずデータ処理部は明らかに３．２．２節で定義したハードウェア要素で全て構成でき，これらのハードウェア要素は制御部からの制御信号に従って，論理装置外部からのデータを入力し，それらを処理し，その結果を論理装置外部へ出力する．

一方制御部は，１つ以上の制御装置から構成されており，これらの全ての制御装置は論理装置が起動される以前にはまだ処理が実行されていないことから，全てACCEPT状態になければならない．ついで論理装置が装置起動信号（CO）を受け取った後に論理装置の最初の処理に対応する制御装置が起動されなければならない．これは仕様中の最初に実行されるFUNCTIONに対応する制御装置のACCEPT状態への遷移実行入力として，装置起動信号を用いることによって実現できる．こうすることにより，論理装置に装置起動信号が与えられると制御部が動作を開始し，仕様で定められた入力，処理，出力を実行することができるようになる．

さて各制御装置への入力となっている信号は全て制御部への入力と考えられるので，装置外部からの入力は，制御部への入力とデータ処理部への入力の２つに分けることができる．そこで，制御部に入力される入力線を外部制御入力線，その信号を外部制御入力と呼び，データ処理部に入力される入力線を外部データ入力線，その信号を外部データ入力と呼ぶことにする．出力についても同様に２つの部分に分けることができ，制御部から出力される出力線を外部制御出力線，その信号を外部制御出力と呼び，データ処理部から出力される出力線を外部データ出力線，その信号を外部データ出力と呼ぶことにする．

装置外部との入出力はこのような各々２つの部分に分割することができるが，装置内部ではこれらの２つの部分をつなぐ接続線が存在する．このうち，制御部からデータ処理部への接続線は先に述べた制御信号を伝送するためのものであり，またデータ処理部から制御部への信号は，制御信号に対する制御のフィードバックと考えることができる．これを制御条件信号と呼ぶことにする．

以上述べた事柄をもとにした論理装置の内部構成を図１に示す．

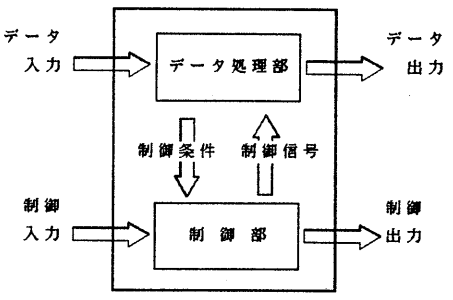

図１　論理装置の構成

4　論理検証の基礎

4．1　制御部の論理検証

前節ではバッファによる仕様記述中の機能のまとまりであるFUNCTIONに対応した制御装置を定義し，これを用いて制御部を記述するようした．本節では，このようにして定められた制御装置が，もとの仕様に記述されている機能を満たしているかどうかを検証する手法の基礎について検討する．検証は次の２段階で行う．まず第１段階では各FUNCTIONに対応する制御装置の動作が，FUNCTIONの中に記述されている機能を満たしているか否かの検証を行う．次いで第２段階では，各制御装置が正しく設計されていることを確認した後で制御部全体が仕様に記述されている機能を満たしているか否かの検証を行う．

なお本稿では，制御部の論理検証方法についてのみ検討を行い，データ処理部の検証については別稿に譲ることにする．

4．2　FUNCTIONに対応する制御装置の検証

ここではFUNCTION内のStatementの並びが表わす制御構造及びそれに対応する制御装置の状態遷移を，各々を入力ラベルと出力ラベルを伴った２つの有向グラフで表現し，このグラフを状態遷移図に持つ２つの順序機械の対応を調べることによって検証を行う．

但しラベルに記入する制御装置への入力及び出力の内，遷移実行信号は各制御装置の実行順序を決定するためにのみ用いられる信号であるため，この検証は後で述べる方法で実行することにし，ここでは考慮しないことにする．また制御信号及び制御条件信号の検証についてもデータ処理部の検証と併せて行う必要があるため，ここでは検証の対象から外す．従って有向グラフの入力ラベルには外部制御入力が，出力ラベルには外部制御出力のみが示されることになる．

4．2．1　FUNCTIONからのグラフの構成

FUNCTIONの機能を記述するStatementの並び（付録参照）に基づく有向グラフを構成するための手続きを示す．但し，”,”で区切られた一連のStatementの並びは並列に実行することを表わしているので，その並びもまとめて１つの”Statement”と呼ぶことにする．

まず外部制御入力は，仕様記述の中のIF文とWHILE文の”条件”（付録参照）を構成する全ての入力線からの信号及びWAIT文の”入力名”で示される全ての入力線からの信号からなる．これらをXg1,･･･,Xgaとし，

$xg=(Xg1,Xg2,\cdots,Xga)\ (Xgi\in\{1,0\},1\leqq i\leqq a)$

と表わすことにする．

次に外部制御出力は，仕様記述の中の全てのSET文の”（出力名）･･･”（付録参照）が示す全ての出力線への信号である．これらをZg1,･･･,Zgbとし，

$zg=(Zg1,Zg2,\cdots,Zgb)\ (Zgi\in\{1,0\},1\leqq i\leqq b)$

と表わすことにする．

これらのｘｇ及びｚｇをそれぞれグラフのエッヂに付ける入力ラベル及び出力ラベルとして用いる．但し，１つのStatementが実行された後で無条件に次のStatementが実行される場合には，それらに対応するノードを結ぶエッヂにラベル”＊”を付ける．またｚｇ＝（0,0,･･･,0）なる値が出力される場合には，出力ラベルｚφを付けるものとする．

［手続き１］

ステップ１：最初のStatementに対してノードｓ1を割り当て，始発ノードとする．そしてi=1としてステップ２へ移る．

ステップ２：ノードｓiに対し，Statementの種類によって次のように次ノード,エッヂ,入力ラベル,出力ラベルを定める．但し出力ラベルについては，次に述べるSET文に対するノードから出て行くエッヂにのみｚφと異なる出力ラベルを付け，その他のStatementに対応するノードから出て行くエッヂの出力ラベルは全てｚφとする．

（１）IF（条件）THEN（副文１）ELSE（副文２）ENDIF

副文１の最初のStatementに対してノードｓi+1を，また副文２の最初のStatementに対して別のノードｓi+2を割り当てる．次に”条件”が信号Xgs,Xgt,･･･,Xgu∈{Xg1,Xg2,･･･,Xga}の論理演算であるｆ（Xgs,Xgt,･･･,Xgu）で表わされているとき，次のような入力の集合Ｘｇ1,Ｘｇ2を構成する．ここでk及びjをそれぞれＸｇ1の要素数,Ｘｇ2の要素数とする．ここでXgs,Xgt,･･･,Xgu以外の信号は任意の値をとりうるものとする．

$Xg1=\{x1i=(Xg1,Xg2,\cdots,Xga)\mid f(Xgs,Xgt,\cdots,Xgu)=1,1\leqq i\leqq k\}$

$xg2\in\{x2l=(Xg1,Xg2,\cdots,Xga)\mid f(Xgs,Xgt,\cdots,Xgu)=0,1\leqq l\leqq j\}$

次いでＸｇ1の要素ｘ11,ｘ12,･･･,ｘ1kを各々ラベルとするk個のエッヂでノードｓiからｓi+1までを結び，Ｘｇ2の要素ｘ21,ｘ22,･･･,ｘ2jをラベルとするj個のエッヂでノードｓiからｓi+2までを結ぶ．同様にして(副文１)及び(副文２)の中の文に対してノード及びエッヂを割り当てる．そして副文１の最後のStatementに対するノードがｓi+p,副文２の最後のStatementに対するノードがｓi+r(i+r＞i+p)である時,ノードｓi+r+1を設け，ｓi+p及びｓi+rからｓi+r+1までを入力ラベル”＊”を持つエッヂで結ぶ．iをi+r+1とする．

（２）WHILE（条件）DO（副文）ENDWHILE

副文の最初のStatementに対してノードｓi+1を割り当てる．IF文と同様にして副文の中のStatementに対してノードを割り当てる．副文の中の最後のStatementに対するノードがｓi+rであるとき，ｓi+rからｓiまでを入力ラベル”＊”を持つエッヂで結ぶと共に，新たにノードｓi+r+1を設ける．次いでIF文と同様にして条件中に含まれる信号に基づいて入力の集合Ｘｇ1,Ｘｇ2を構成し，集合Ｘｇ1の要素をラベルに持つエッヂでノードｓiからｓi+1までを結び，Ｘｇ2の要素をラベルに持つ

エッヂでノードｓiからｓi+r+1までを結ぶ．iをi+r+1とする．

（３） WAIT FOR（入力名）TO GO ON

ＷＨＩＬＥ文の副文がない場合に相当するため，IF文と同様に"入力名"で示される入力線の値が１及び０である場合に対して，入力の集合Ｘｇ1,Ｘｇ2を構成する．次いでノードｓi+1を設け，Ｘｇ1の各々の要素をラベルとしてもつエッヂでｓiからｓi+1までを結び，同様にＸｇ2の各々の要素をラベルに持つエッヂでｓi自身のループを作る．iをi+1とする．

（４）TAKE（入力名）･･･,MAKE（Ａ）BE EQUAL TO（Ｂ）

ノードｓi+1を設け，ｓiからｓi+1を"＊"を持つエッヂで結ぶ．iをi+1とする．

（５） SET（出力名）･･･

出力名が示す出力線がZgs,Zgt,･･･,Zgu∈｛Zg1,Zg2,･･･,Zgb｝である時，出力を

ｚｇ＝（Zg1,･･,Zgs,Zgt,･･,Zgu･･,Zgb），Zgs=Zgt=,･･･,=Zgu=1，Zgi=0（１≦i≦b,i∉｛s,t,･･･,u｝）

とする．次いでノードｓi+1を設け，ｓiからｓi+1を入力ラベル"＊"及び出力ラベルｚｇを持つエッヂで結ぶ．iをi+1とする．

ステップ３：実行委譲部に到ったならノードqiを実行委譲部に割り当て，手続きを終り，そうでないときにはステップ２に戻る．

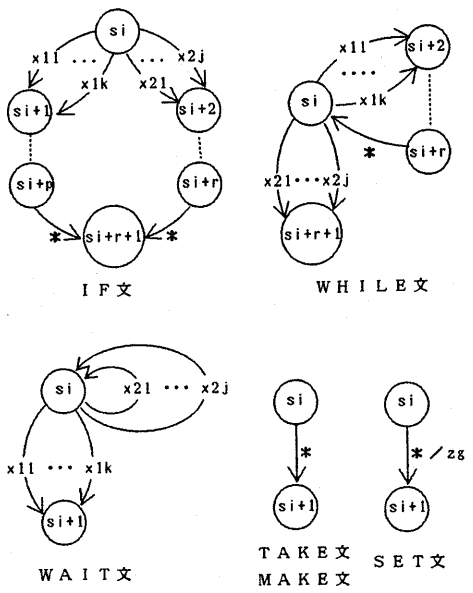

図２　バッファからのグラフの構成
（記入無き出力は"ｚφ"）

４．２．２　制御装置からのグラフ化

制御装置の状態の集合ＱにおいてACCEPT状態及びTRANSFER状態は，各制御装置の動作順序を定めるためにのみ用いられる状態であり，これらの制御装置間の検証については後で述べるため，ここではＱからACCEPT状態を除き，TRANSFER状態からの遷移及び出力も除いてグラフを作成することにする．さらにここで検証の対象としない入力と出力を除いて，制御装置Ｍから新たに制御装置Ｍ'を次のようにして構成する．

Ⅰ）入力Ｘから制御条件信号及び遷移実行信号を除き，

Ｘ'＝｛ｘ1,ｘ2,･･･,ｘn'｝,ｘi＝（XI1,XI2,･･･,XIk）（1≦i≦n'）

とする．

Ⅱ）状態の集合ＱからACCEPT状態ｑa(＝ｑ1)を除き，

Ｑ'＝｛Ｑ－｛ｑa｝｝＝｛ｑ2,･･･,ｑｍ｝

とする．この時，ｑaから遷移する唯一の次状態をあらためて始発状態とする．

Ⅲ）出力Ｚから制御信号及び遷移実行信号を除き，

Ｚ'＝｛ｚ1,ｚ2,･･･,ｚs'｝,ｚi＝（ZO1,ZO2,･･･,ZOj）（1≦i≦s'）

とする．

Ⅳ）δ'：Ｘ'×Ｑ'→Ｑ'

Ⅴ）ω'：Ｘ'×Ｑ'→Ｚ'

このようにして構成された制御装置Ｍ'に基づいて次の手続きにより，グラフを作成する．

［手続き２］

ステップ１：状態ｑ2,･･･,ｑｍ(∈Ｑ')に対し，それぞれ１つずつノードｑ2,･･･,ｑｍを割り当てる．このとき始発状態に対応するノードを始発ノードとする．i=1としてステップ２へ移る．

ステップ２：ｑiに対しｑj∈Ｑ'，ｘ∈Ｘ'，ｚ∈Ｚ'であって，

δ'（ｘ,ｑi）＝ｑj

ω'（ｘ,ｑi）＝ｚ

であるとき，ノードｑiからｑjを入力ラベルｘ，出力ラベルｚを持つエッヂで結ぶ．但しこの場合，任意のｘ∈Ｘ'に対し遷移先ｑj及び出力ｚが唯一に定まる場合には，ノードｑiからｑjを入力ラベル"＊"，出力ラベルｚを持つエッヂ１本だけで結ぶ．また出力がｚ＝（0,･･･,0）である場合には，出力ラベルを"ｚφ"とする．

ステップ３：iの値がm-1であるなら手続きを終了し，そうでないならiの値を１だけ増してステップ２へ戻る．

４．２．３　FUNCTIONと制御装置の間の検証

前節で述べた手続きに従いFUNCTIONから作成された有向グラフをグラフＲ，そのFUNCTIONに対応する制御装置から作成されたグラフをグラフＰと呼ぶことにする．

まずグラフのノードについて次のように定義する．

［定義１０］　ノードから出て行くエッヂの入力ラベ

ルが"＊"且つ出力ラベルが"ｚφ"である時，そのノードを無効ノードと呼ぶ．
このようなノードは仕様と動作記述の間で対応づけることができず検証には不用のものであるため，次のような手法でグラフＲとＰから削除することにする．
［無効ノードの削除方法］
グラフＲ及びグラフＰにおいて，ノードｒ（ｒはノードｓｉまたはｑｊ）が無効ノードであるなら，ｒに入るエッヂをｒの次の遷移先のノードｒ'に入るエッヂとし，ノードｒ及びｒから出て行くエッヂの両者をグラフから削除する．この時ｒが始発ノードなら，ｒ'をあらためて始発ノードとする．
この方法を繰り返すことによりグラフＲ及びグラフＰから無効ノードを全て削除することができる．

次に２つのグラフのノードについて次のように定義する．
［定義１１］　２つのグラフの中の各々１つずつのノードについて，各々のノードから出て行くエッヂの数が等しく且つそれらのエッヂの全ての入力ラベルと出力ラベルの組がお互いに等しいとき，この２つのノードは等価であると言う．

無効ノードが全て削除されたグラフについて，グラフＲに存在するノードと等価なノードがグラフＰに存在し，且つそれがグラフＲの順序通りに並んでいることを確認することによって，仕様中のFUNCTIONとそれに対応する制御装置の検証を行う方法を次に示す．なお，２つのグラフのノードについては，等価であることを確認したノードを確認ノード，そうでないノードを未確認ノードと呼ぶ．但し，グラフＲの始発ノードがｓｉ，グラフＰの始発ノードがｑｊであるものとする．
［手続き３］
ステップ１：ｓｉとｑｊが等価であるならばこれらを確認ノードとし，もし等価でないならば，仕様とハードウェア記述の間で不一致が存在したものとして手続きを終了する．
ステップ２：ｓｉが未確認ノードへ入るエッヂを持つならステップ４へ移り，そうでないならステップ３へ移る．
ステップ３：グラフＲの中で未確認ノードへ入るエッヂを持つ確認ノードがｓｉ'，グラフＰの中でｓｉ'と等価なノードがｑｊ'であるとすると，ｉの値をｉ',ｊの値をｊ'としてステップ４へ移る．もしこのときグラフＲに未確認ノードが存在しないなら，ハードウェア記述は仕様を満たしているものとして手続きを終了する．
ステップ４：ｓｉ及びｑｊから等しい入力ラベルと出力ラベルを持つエッヂが入る未確認ノードがそれぞれｓｉ",ｑｊ"であるなら，ｉの値をｉ"，ｊの値をｊ"としてステップ１に戻る．

４．３　FUNCTIONと制御装置の実行順序の検証

ここではまず１つのFUNCTION及び１つの制御装置を各々１つのノードに対応させ，それらのFUNCTIONの実行順序及び制御装置の動作順序を，入力ラベルを持つエッヂで表わすことによってグラフを作成する．次にこれらの２つのグラフを比較することによって両者の間の検証を行なう．

バッファで記述された仕様がn（≧1）個のFUNCTIONの並びから成っているとき，設計されたハードウェア記述の検証を行うための手続きを次に示す．但し制御装置ＭiはFUNCTIONi（1≦i≦n）に対応する制御装置である．
α）FUNCTIONの並びからグラフへの変換
FUNCTIONの実行順序はその中の実行委譲部に記述されており，この実行順序は４．２．１節で示した入力ｘｇ＝（Xg1,Xg2,･･･,Xga）（Xgi∈｛1,0｝,1≦i≦a）によって制御される．そこでここでも同じ入力を用いて，グラフの入力ラベルを表わすことにする．
［手続き４］
ステップ１：FUNCTION1からFUNCTIONnに対し，それぞれ対応するノードΛ1,･･･,Λnを割り当てる．i＝1として次のステップ２に移る．
ステップ２：FUNCTIONiの実行委譲部の記述が，
①BEIDLE s,t,･･･,u（s,t,･･･,u∈｛1,･･･,n｝）
ならば，ノードΛiからノードΛs,Λt,･･･,Λuを入力ラベル"＊"を持つエッヂで結ぶ．
②IF（条件）THEN BEIDLE s1,t1,･･･,u1 ELSE BEIDLE s2,t2,･･･,u2（s1,･･･,u1,s2,･･･,u2∈｛1,･･･,n｝）
ならば，４．２．１節のIF文と同様にして入力の集合Ｘｇ1,Ｘｇ2を求める．次にこれらの入力の集合について，Ｘｇ1の要素数kだけのラベルｘ11,ｘ12,･･,ｘ1kをそれぞれ持つk個のエッヂでノードΛiからΛs1,Λt1,･･･,Λu1までを結ぶ．次いでＸｇ2の要素数jだけのラベルｘ21,ｘ22,･･･,ｘ2jをそれぞれ持つj個のエッヂでノードΛiからΛs2,Λt2,･･･,Λu2までを結ぶ．
ステップ３：iの値がnであるなら手続きを終了し，そうでないならiの値を１だけ増してステップ２に戻る．
β）制御装置の並びからのグラフへの変換
各制御装置のACCEPT状態及びTRANSFER状態における遷移実行信号の受渡しを調べることにより，制御装置の動作順序を知ることができる．具体的なグラフへの変換方法を次に述べる．

なおここでは，３．２．３節で示した入力Ｘの部分集合ＸＣ1,ＸＣ2,･･,ＸＣh及び出力Ｚの部分集合ＺＣ1,ＺＣ2,･･･,ＺＣhを用いる．
［手続き５］
ステップ１：全ての制御装置Ｍ1,Ｍ2,･･･,Ｍnに対し，それぞれ対応するノードΣ1,Σ2,･･･,Σnを記入する．i＝１として次のステップ２に移る．
ステップ２：制御装置Ｍiにおいて，
ω（ｘ,ｑｔ）＝ｚ（ｘ∈Ｘ,ｚ∈ＺＣr(r∈｛1,･･･,h｝)）
であり，且つ制御装置Ｍjにおいて，全てのｘＣr∈ＸＣrに対し，

δ（xCr,qa）＝qx（qx≠qa,qx∈Q）
であるならば，xの中の制御入力信号XI1,XI2,･･･,XIkの並びからなるベクトルを入力ラベルとして持つエッヂでΣiからΣjを結ぶ．
ステップ３：iの値がnであるなら手続きを終了し，そうでないならiの値を１だけ増してステップ２に戻る．
γ）両グラフの比較
FUNCTIONの並びから作成したグラフをグラフR'，制御装置の並びから作成したグラフをグラフP'とする．FUNCTIONと制御装置は一意に対応しているので，グラフR'とグラフP'のノードは容易に対応づけることができる．また両グラフには，論理装置の制御入力値が入力ラベルとして示されているため，グラフR'の入力ラベルに対して，グラフP'にも同じ入力ラベルが存在しなければならないことになる．
従って，グラフR'のノードとそれに対応するグラフP'のノードにおいて，それらのノードから発する同じ入力ラベルを持つエッヂが各々，対応するノードに入るとき，ハードウェア記述が仕様を満たしているということができる．逆にこれを満たさないノードの対が１組でもあるなら，仕様とハードウェア記述の間で不一致が存在することになる．

５．むすび
本稿では仕様記述言語バッファとの間で容易に論理検証を行うためのＨＤＬの基礎について検討を行った．ＨＤＬを用いて動作記述レベルにおいて論理装置を記述する場合には，装置をデータ処理部と制御部に分割して記述し，各々を別々に検証することにより容易に論理装置の検証を行うことができるものと考えている．次に具体的な検証方法の基礎として，仕様の制御構造と制御部の動作をそれぞれ有向グラフとして表わすことにより，制御部の検証を行う方法を示した．
今後はＨＤＬの作成及びデータ処理部の検証方法の検討を進め，ここで示した制御部の検証と併せて用いることにより，検証の容易な新しいＨＤＬとそのＨＤＬを用いた論理装置の検証方法を提案する予定である．

［参考文献］
(1)"論理設計ＣＡＤに関する調査報告書"
昭和６１年３月,日本電子工業振興協会
(2)多喜康朗,Albert･Palacios,角山正博,内藤祥雄
"設計の支援と検証に適したハードウェア記述言語"
昭和６１年度電子通信学会信越支部大会
(3)当麻喜弘,山崎克典,内藤祥雄,鰯谷昌司　共訳
"コンピュータの構成と設計Ⅰ,Ⅱ" サイエンス社

［付　録］
バッファによって記述される仕様書は，次のように宣言部と処理記述部とに大別される．
SYSTEM（システム名）.（宣言部）（処理記述部）END.
宣言部は，処理記述部で用いられる処理要素を宣言するための部分である．データを入力すると共に入力されたデータを蓄えておく処理要素，装置の内部にデータを蓄える処理要素，及び処理されたデータを外部に出力する処理要素などを宣言できる．
処理記述部は，宣言部で宣言された処理要素を用いて一まとまりの機能あるいは仕事を表わすFUNCTIONの並びからなっている．このようなFUNCTIONは，Statementの並びと実行委譲部からなっている．FUNCTIONの構成を次に示す．
FUNCTION(数字)(Statement)･･(実行委譲部)END(数字)
Statementの並びはFUNCTIONの機能を記述しており，次の６種類の文からなる．
＊IF（条件）THEN（副文１）ELSE（副文２）ENDIF
条件が満たされているならば副文１を実行し，条件が満たされていないなら副文２を実行する．但し副文１，副文２はそれぞれ１つ以上のStatementの並びである．
＊WHILE （条件） DO （副文） ENDWHILE
条件が満たされている間，副文を繰り返し実行する．
＊WAIT FOR （入力名） TO GO ON
入力名で示される処理要素の入力の値が１になるまで，次のStatementの実行を停止する．
＊TAKE （入力名）･･･
入力名で示される処理要素からデータを内部へ取り込み，格納する．
＊MAKE （Ａ） BE EQUAL TO （Ｂ）
Ａで示される処理要素の値をＢに等しくする．
＊SET （出力名）･･･
出力名で示される処理要素の値を１にする．
これらのStatementは";"で区切られていればシーケンシャルに実行し，TAKE,MAKE,SET文に限り",,"で区切られていれば並列に実行する．
Statementの並びの後には，そのFUNCTIONで表わされる仕事が終了した後に次に動作するべきFUNCTIONを示す実行委譲部が続く．各FUNCTIONの実行委譲部の中で次に実行すべきFUNCTIONを示すことにより，装置が実行すべき仕事の流れを表わすことができる．実行委譲部の内容を示す．
NEXT BEIDLE（数字）,･･･
この文はFUNCTIONの実行を終了したなら，数字で示されるFUNCTIONの実行を次に開始するということを示している．もし複数の数字が並記してある場合は，それらの数字で示されるFUNCTIONが並列実行されることを意味する．また定められた条件に従って実行されるFUNCTIONを選択したい場合は，
NEXT IF （条件） THEN BEIDLE（数字１） ELSE BEIDLE（数字２） ENDIF
と記述する．これは，条件が満たされているときは数字１で示されるFUNCTIONを実行し，満たされていない場合には数字２で示されるFUNCTIONが実行されることを表わしている．