設計自動化　68－20
（1993. 10. 29）

代数的手法を用いた順序回路の段階的設計における実現検証

侯 豫榕　杉山 裕二　岡本 卓爾

岡山大学工学部情報工学科

岡山市津島中 3-1-1

あらまし

高信頼性が要求されるハードウェアを対象に，代数的言語を用いた形式的仕様記述方法や，設計された回路等が得られた仕様を満たしているか否かの検証（実現検証）法が提案されている．しかしながら，従来の検証方法は，回路全体を扱っているため，回路規模の増大に伴って検証作業が極端に複雑化するという問題点をもっていた．そこで本論文では，回路のモジュール化に着目し，モジュール毎の実現検証により，回路全体の実現検証が可能となるための十分条件について検討している．まず，仕様（上位）に対して回路等（下位）の間に成立すべき関係を，上位と下位の入出力が常に対応することと定式化し，この定式化のもので上述の十分条件を与えている．この十分条件は，上位と下位で，モジュール間の接続が対応しており，更に入出力や内部状態の対応と状態遷移の対応が存在することを含んである．また，リングアービタを例に用いて，この十分条件を適用した検証例を示している．

和文 キーワード　代数的仕様　順序回路　実現　検証

Implementation Verification for a Sequential Circuit with Algebraic Specification

Yurong HOU,　Yuji SUGIYAMA　and　Takuji OKAMOTO

Faculty of Engineering, Okayama University

3-1-1, Tsushima-naka, Okayama-shi, 700, Japan

Abstract

An algebraic method is proposed for specifying sequential circuits and verifying their implementations. With increasing size, however, the verification becomes increasingly complex. Usually, a large circuit consists of a number of small modules, and an implementation verification of each modules is simpler than that of the entire circuit. In order to decrease the complexty of the implementation verification of the entire circuit, this paper discusses a method of the implementation verification based on implementations of the constituent modules. It presents a sufficient condition under which an implementation of the entire circuit from a collection of those of the constituent modules. An example is also given of an implementation verification using the sufficient conditions.

英文 key words　Algebraic Specification,　Sequential Circuit,　Implemeatation,　Verification

1. まえがき

高信頼性が要求されるハードウェアの開発においては，設計時に混入する誤りをいかにして防ぐかが重要課題の一つになっている．この課題に対して，回路等の仕様を抽象レベルで代数的言語により記述し，数学的な証明を通して，仕様のもつ不完全性，矛盾などを取除くための記述・検証法に関する研究が行なわれている[1,2,3]．代数的言語[4,5]は，抽象レベルの仕様から具体的なゲートレベルの回路まで記述できるという特長をもち，この種の検証に適した言語である．また，言語の意味定義が極めて簡単であるため，検証作業を自動化し易いという側面も併せもっており，ソフトウェアや通信プロトコルを例題に，種々の検証法や検証支援系が得られている[6,7,8]．しかし，従来の検証方法では，回路全体を一括して検証の対象とするため，回路規模が大きくなると，検証作業が極端に複雑化するという傾向がある．特に，設計した回路等が与えられた仕様を満たしているか否かの検証（実現検証）では，仕様と回路等の二つの記述を対象にするため，その影響がより深刻となる．

本論文では，規模の大きな回路は通常モジュール化して設計されるという点に着目して，モジュール毎の実現検証により回路全体の実現検証を行なう方法について検討する．このような検証を行うためには，上位の記述（仕様）と下位の記述（回路等）が満たすべき関係（実現関係）を形式的に定義する必要があるが，ここでは，これを上位と下位とで入出力が常に対応することと定式化する．その際，上位の1回の状態遷移は，一般に，下位の複数回の状態遷移で模倣されるため，入出力の対応は下位の模倣が終った直後にとるものとする．この定義では，下位の模倣途中の状態における入出力については何の制約もないが，定義の簡潔化と一般化のためそのようにしている．従って，このような定義のもとでは，モジュール毎に上位と下位の間の実現関係が証明されたとしても，上述のように，下位の模倣途中の状態については制約がないため，上位と実現関係にある下位のモジュールを（上位と同じように）相互に接続したとき，回路全体として上位の状態遷移を模倣する保証がないことになる．そこで本論文では，そのような場合に回路全体として実現関係が成り立つための十分条件を導く．この十分条件は，上位と下位のモジュール間に，入出力や内部状態の対応および状態遷移の対応などが存在することを含んでいる．

この十分条件を適用した検証例として，リングアービタ[9]の動作の仕様を，抽象化の二つのレベルで代数的に記述し，これらの記述間の実現検証を行なう．リングアービタのような同一モジュールを複数個含む回路に対しては，モジュールの実現検証が少なくて済むため，本論文の実現検証法は特に有効である．

2. 順序回路の代数的仕様とその実現

2.1 モジュール化された順序回路の代数的仕様

モジュール化された順序回路の仕様を代数的に記述する方法が提案されている[1,2]．これは，ブール値等の基本データタイプ，および論理積，論理和，論理否定等の基本関数を前提とし，順序回路の仕様を各モジュールの仕様とモジュール間の接続の仕様に分けて行う方法である．

モジュールの仕様は，前提とする基本データタイプや基本関数，および以下のデータタイプや関数と公理からなる．

(1) モジュールの抽象的状態 (以下，単に状態と呼ぶ）を表すデータタイプ．

(2) 初期状態を表す関数（初期関数）．関数値として初期状態を与える．初期状態における出力値等を指定するパラメータ（初期パラメータ）を引数にもつことができる．

(3) 状態遷移を表す関数（遷移関数）．入力および状態を引数とし，関数値として状態遷移後の状態を与える．状態遷移関数の中には，出力や内部状態の値を全く変えず，時間の経過や入力値の変化を表すための状態遷移関数 $idle$ が含まれる．

(4) モジュールの出力および内部状態を表す関数（成分関数）．状態を引数とし，関数値として出力や内部状態を与える．

(5) 初期状態からの到達可能性を表す関数（$valid$ 関数）．状態を引数とし，その状態が初期状態から到達可能なとき，関数値として真 (1 で表す)，そうでないとき偽 (0 で表す) を返す．

以下では，初期状態，遷移関数，成分関数および $valid$ 関数を，それぞれ，$init$，t（または u），f（または g）および $valid$，またはそれらに添字等をつけた記号で表す．また，$valid(s) \equiv 1$ となる状態 s を $valid$ 状態と呼ぶ．

上記 **(2)**～**(5)** の関数の形式的な意味は，以下のいずれかの形の公理により定義する．

(i) $f(init(c)) == exp_1(c)$

(ii) $f(t(s,x)) == exp_2(s,x)$

(iii) $valid(init(c)) == exp_3(c)$

(iv) $valid(t(s,x)) == valid(s)\ \ and\ \ exp_4(s,x)$

ここで，s は状態，c および x は，それぞれ，初期パラメータの並びおよび入力値を表す変数の並びであり，$exp_i(\cdot)$ は，引数の変数，基本関数および成分関数からなる表現式である（以下，同様）．**(ii)** および **(iv)** の形の公理は，それぞれ，状態遷移後の成分関数の値および状態 $t(s,x)$ への到達可能性（すなわち，状態 s で状態遷移 t が起こるか否か）が入力値 x と遷移前の状態 s における成分関数値で決まることを表している．なお，状態と入力により，次に起こる状態遷移は一意に決まらなければならない．すなわち，

$$t_i \neq t_j \Rightarrow valid(t_i(s,x))\ \ and\ \ valid(t_j(s,x)) \equiv 0.$$

任意の状態は，次の **(1)** および **(2)** のみにより定義される状態式で表される．

(1) 初期状態 $init(c)$ と状態を表す変数は状態式である．

(2) s が状態式，t が状態遷移関数，x が入力を表す変数の並びのとき，$t(s,x)$ も状態式である．

以下，状態遷移関数 t と入力（の並び）x に対して，$t(x)$ を遷移と呼び，複数個の遷移が連接されたものを遷移列と呼ぶ．また，状態式 $t_k(t_{k-1}(\cdots t_1(s,x_1),\cdots x_{k-1}),x_k)$ を，遷移列と状態 s との連接 $t_k(x_k)\cdot t_{k-1}(x_{k-1})\cdots t_1(x_1)\cdot s$ で表すことがある．

モジュール間の接続の仕様では，回路全体の状態（全体状態と呼ぶ）を，各モジュールの状態式の並びで表し，与えられた全体状態が，初期状態から到達可能か否かを表す関数 $VALID$ を導入する．各モジュールの動作はモジュールの仕様で規定されているので，$VALID$ 関数の定義の主要な部分は，モジュールの出力がどのモジュールへ伝搬されるかの記述になり，これがモジュール間の接続の仕様に相当する．$VALID$ 関数の定義は以下の形の公理により与えられる．

(v) $VALID(INIT(C)) == \bigwedge_{i=1}^{n} valid(init_i(c_i))\ \ and\ \ exp_5(C)$

(vi) $VALID(T(X)\cdot S) == VALID(S)\ \ and\ \ \bigwedge_{i=1}^{n} valid(t_i(s_i,x_i))\ \ and\ \ exp_6(S,X)$

ただし，nはモジュール数，$C =< c_1, \cdots, c_n >$，$X =< x_1, \cdots, x_n >$，$S =< s_1, \cdots, s_n >$，$INIT(C) =< init_1(c_1), \cdots, init_n(c_n) >$，$T(X) =< t_1(x_1), \cdots, t_n(x_n) >$（全体状態遷移と呼ぶ）であり，$T(X) \cdot S$を

$$T(X) \cdot S =< t_1(x_1) \cdot s_1, \cdots, t_n(x_n) \cdot s_n >$$

と定義する．また，$\bigwedge_{i=1}^{n}$は，$i = 1, \cdots, n$について論理積をとることを表す．

上記公理中の$exp_5(C)$は，各モジュールの初期設定の関係を表す表現式であり，$exp_6(S, X)$は，モジュール間の接続関係を表す$x_i = f_j(s_j)$なる形の等式（接続仕様と呼ぶ）の論理積である．$x_i = f_j(s_j)$のような等式が含まれていれば，モジュールjの出力f_jとモジュールiの入力x_iとが等しい，すなわち接続されていることになる．一つの入力に接続される出力は高々一つと考えることができるので，各x_iに関する接続仕様（等式）は高々一つと限定する．

$s = t_k(x_k) \cdots t_1(x_1) \cdot init(c)$および$s' = t'_{k'}(x'_{k'}) \cdots t'_1(x'_1) \cdot init'(c')$をモジュール$i$と$i'$の状態とする．$k = k'$であり，かつ各$j(j = 1, \cdots, k)$について，$t_j(x_j) \cdots t_1(x_1) \cdot init(c)$の出力と$x'_j$，および$t'_j(x'_j) \cdots t'_1(x'_1) \cdot init'(c')$の出力と$x_j$が$VALID$関数の公理中の接続仕様を満たすとき，単に，$s$と$s'$は接続仕様を満たすという．また，全体状態$S$において，モジュールの状態のすべての組が接続仕様を満たすとき，Sは接続仕様を満たすという．$VALID$関数の公理の形より，$VALID(S) \equiv 1$なる全体状態S（$VALID$全体状態と呼ぶ）は接続仕様を満たす．

2.2 実現の定義

二つの回路が与えられたとき，両者の入出力系列が常に等しいとき，通常，両者は等価であるといわれる．実現関係の基本的な考え方はこれと同じであるが，上位と下位という詳細化特有の関係を考慮する必要がある．その一つとしては，詳細化において，上位の一回の状態遷移を下位では何回かの状態遷移で表すということがある．このため，下位の状態には，上位の状態に対応する状態と上位の遷移を模倣する途中の状態とがある．このため，下位では，どの状態が上位に対応する状態であるかを，状態を引数とする述語ISで表すことにする（$IS(S)$が真のとき，下位の状態Sは上位の状態に対応する）．もう一つとしては，入出力についても詳細化が行われるということがある．例えば，上位の入力xを下位では入力y_1とy_2の連接$y_1 \cdot y_2$で表すことがある．このため，上位と下位の入出力間に対応関係を導入する必要がある．以下，実現の定義を行う前に，状態Sに至るまでの入出力の系列を表す入出力履歴および入出力の対応を定義する．なお，出力を表す成分関数，外部入力を表す変数の指定は別途与えられているとする．

[定義 2.1]（入出力履歴）

各モジュールにおいて，状態sから何回か（0回も含む）の状態遷移で状態s'に到達するとき，$s \leq s'$と書く．そして，任意の状態sに対して，$s_i \leq s$であるようなすべての状態を$s_1 \leq \cdots \leq s_k(= s)$とし，状態$s_i$における入力を$I(s_i)$で表す．このとき，$< I(s_1), \cdots, I(s_k) >$を状態$s$の入力履歴と呼び，$HistI(s)$で表す．そして，状態に関する述語$IS$が与えられたとき，述語$IS$が真になる状態$s_i$のみからなる入力$I(s_i)$の系列を状態$s$の述語$IS$に関する入力履歴と呼び，$HistI_{IS}(s)$で表す．また，出力に関しても，状態$s_i$における出力を$O(s_i)$で表し，出力履歴$HistO(s)$および述語$IS$に関する出力履歴$HistO_{IS}(s)$を同様に定義する．以上の記号の定義域を（モジュール毎の値の並びとして）全体状態にも拡張する．

[定義 2.2]（入出力の対応）

同じ基本関数を前提とする二つの仕様をAおよびBとする．Aの各入力$x_i(1 \leq i \leq l)$および各出力$f_j(1 \leq j \leq m)$に対し，$x_i == EXP_{x_i}(Y)$ $(1 \leq i \leq l)$および$f_j == EXP_{f_j}(G)$ $(1 \leq j \leq m)$（但し，YおよびGは，それぞれBの入力および出力の並びであり，EXP_{x_i}およびEXP_{f_i}は引数と基本関数のみからなる式である）を，それぞれ，x_iに関する入力の対応およびf_jに関する出力の対応と呼ぶ．そして，

$$EXP_I(Y) =< EXP_{x_1}(Y), \cdots, EXP_{x_l}(Y) >$$
$$EXP_O(G) =< EXP_{f_1}(G), \cdots, EXP_{f_m}(G) >$$

とおくとき，入力の対応および出力の対応を単にEXP_IおよびEXP_Oで表す．また，入力の対応EXP_Iおよび出力の対応EXP_Oの定義域を，入力履歴または出力履歴$H =< H_1, \cdots, H_k >$に対し，以下のように拡張する．

$$EXP_I(H) =< EXP_I(H_1), \cdots, EXP_I(H_k) >$$
$$EXP_O(H) =< EXP_O(H_1), \cdots, EXP_O(H_k) >$$

[定義 2.3]（実現）

同じ基本関数を前提とする二つの仕様AとB，入力の対応EXP_Iおよび出力の対応EXP_Oに対して，Bの全体状態に関する述語ISが存在して，下記条件 **(1)** および **(2)** が成り立つとき，かつそのときのみ，Bは，$< EXP_I, EXP_O >$のもとでAの実現であるという．

(1) Aのすべての$VALID$全体状態S_Aに対し，Bの$VALID$全体状態S_Bが存在し，以下が成り立つ．

$$HistI(S_A) \equiv EXP_I(HistI_{IS}(S_B)) \quad (2.1)$$
$$HistO(S_A) \equiv EXP_O(HistO_{IS}(S_B)) \quad (2.2)$$

(2) Bのすべての$VALID$全体状態S_Bに対して，次の **(a)** または **(b)** が成り立つ．

- **(a)** $IS(S_B) \equiv 1$のとき，式 (2.1) および (2.2) を満足するAの全体状態S_Aが存在する．
- **(b)** $IS(S_B) \equiv 0$のとき，$S_B \leq S'_B$かつ$IS(S'_B) \equiv 1$であるBの$VALID$全体状態S'_BおよびAの$VALID$全体状態S'_Aが存在し，S'_AとS'_Bに関して式 (2.1) および (2.2) を満たす．

上記条件の **(1)** は，仕様Aの動作を模倣する仕様Bの動作が存在しなければならないことを表し，また条件 **(2)** では，仕様Bの動作は仕様Aの動作を模倣するためのものに限られるということを表している．

3. 実現のための十分条件

実現の検証は，一般に回路規模が大きくなると複雑となる．そこで，モジュールごとの実現検証により，回路全体の実現検証が可能であるための十分条件について考える．

実現の定義では，上位と下位で，成分関数値の中で出力を表すもののみが対応していればよいとした．これは実現という関係を広くとらえる意味で必要な考え方である．しかし一方において，実際の詳細化においては，上位の記述の構造を保存しつつ，それを詳細化したものが下位の記述であることが多いと思われる．したがって，ここでは，上下間で全成分関数値の対応（成分の対応）があるとする．このため以下では，出力の対応EXP_Oの代わりに成分の対応（EXP_Cと書く）を用いる．EXP_Cは，成分関数値の範囲が拡大しただけで，その定義はEXP_Oと基本的に同じである．また，C_AおよびC_Bを，それぞれ，上位モジュールの状態s_Aの全成分関数値の並びおよび下位モジュールの状態s_Bの全成分関数値の並びとし，$C_A = EXP_C(C_B)$が成り立っているとき，$s_A \overset{\overline{EXP_C}}{} s_B$と書く．この$\overline{EXP_C}$の定義域を全体状態に拡張する．また，成分関数値の対応が明らかなとき，$\overline{EXP_C}$を単に$\sim$で表す．

一般に上位の一回の遷移は下位の複数個の遷移列に対応するが，実際の詳細化においては，上下間の遷移の対応を明確に意

識して下位レベルを設計することが多いと思われる．そのため，上下のモジュール間に遷移の対応もあるとする．まず，このモジュール間の遷移の対応の形式的な定義を行う．一般に，上位の一つのモジュールは下位の複数のモジュールに対応することが多いが，以下では議論を簡潔にするため，上下のモジュールの対応は一対一であるとする．

a および b を，それぞれ，A および B のモジュールとし，

$V_a = \{t(x) \mid t(x)$ は $idle$ 以外の a の遷移 $\} \cup \{init_a\}$

$V_b' = \{u(y) \mid u(y)$ は $idle$ 以外の b の遷移 $\}$

$W_b = V_b'^{+} \cup \{init_b\}$

とおく．ここで，$init_a$ および $init_b$ は，それぞれ，a および b の初期状態であり，$V_b'^{+}$ は，V_b' の要素の有限長（但し，1 以上）の系列である．ϕ を V_a から 2^{W_b}（W_b のべき集合）への写像とするとき，ϕ の定義域をつぎのように V_a^{+} へ拡張して用いる．

$$\phi(t_k(x_k)\cdots t_1(x_1)) = \{w_k\cdots w_1 \mid w_i \in \phi(t_i(x_i))(1 \le i \le k)\}$$

[定義 3.1]（遷移の対応）

同じ基本関数を前提とする二つの仕様 A と B，入力の対応 EXP_I，並びに成分の対応 EXP_C が与えられているものとする．A のモジュール a および B のモジュール b に対して，写像 ϕ : $V_a \to 2^{W_b}$ が存在して以下が成り立つとき，$< EXP_I, EXP_C >$ のもとでの a から b へ遷移の対応があるという．また，ϕ を遷移の対応と呼ぶ．

(1) $\phi(init_a) = \{init_b\}$ かつ $init_a \sim init_b$

(2) 任意の $t(x) \in V_a - \{init_a\}$ と任意の $w = u_k(y_k)\cdots u_1(y_1) \in \phi(t(x))$ に対して，次の (i) ～ (iii) が成り立つ．

(i) $x \equiv EXP_I(y_1)$

(ii) $s_a \sim s_b$ ならば，
$valid(t(x)\cdot s_a) \equiv valid(u_1(y_1)\cdot s_b)$

(iii) $s_a \sim s_b$ かつ $valid(t(x)\cdot s_a) \equiv 1$ ならば，
$t(x)\cdot s_a \sim w \cdot s_b$

(3) $u(y)\cdot s_b$ の形（$u(y)$ は b の遷移）の任意の $valid$ 状態に対して，a の状態 s_a' と遷移 $t(x)$，並びに b の状態 s_b' と $\phi(s_a')$ と遷移列 w'，w'' が存在して，
$w''\cdot u(y)\cdot w'\cdot s_b' \in \phi(t(x)\cdot s_a')$，かつ
$valid(w''\cdot u(y)\cdot w'\cdot s_b') \equiv 1$

遷移の対応と実現との関係について述べると，上記 **(2)** は，$\phi(t(x))$ に属する b の遷移列が a の遷移 $t(x)$ を模倣することを表しており，実現の **(1)** に相当する条件である．また **(3)** は，b の遷移列がいずれかの $\phi(t(x))$ に属する遷移列（即ち，a の動作を模倣する遷移列）の一部であることを表しており，実現の **(2)** に相当する．これらの条件は，実現の条件をより強めたものになっており，従って，a から b へ遷移の対応が存在すれば，b は a の実現である．以下では，他のモジュールの遷移の対応も同じ記号 ϕ で表す．

次に，上位と下位のモジュール間に遷移の対応が存在する場合に，仕様全体として下位が上位の実現となるための十分条件を示す．

[定理]（十分条件）

同じ基本関数を前提とした二つの仕様 A と B，入力の対応 EXP_I，および成分の対応 EXP_C に対して，次の **(1)**～**(4)** が成り立つならば，B は $< EXP_I, EXP_C >$ のもとで A の実現である．

(1) A のモジュール集合から B のモジュール集合への単射 θ が存在し，A の各モジュール a から $\theta(a)$ へは $< EXP_I, EXP_C >$ のもとで遷移の対応がある．この遷移の対応を ϕ で表す．

(2) $t(x)$ を A のモジュールの任意の遷移，w，$w' \in \phi(t(x))$ を異なる任意の遷移列，y_1 と y_1' を $EXP_I(y_1') \equiv EXP_I(y_1)$ なる入力，y_i と $y_i'(2 \le i \le k)$ を異なる入力とする．

(a) w，w' は入力のみが異なる遷移列であり，以下のように表される．
$$w = u_k(y_k)\cdots u_1(y_1)$$
$$w' = u_k(y_k')\cdots u_1(y_1')$$

(b) $valid(w\cdot s) \equiv 1$ なる任意の状態 s，任意の成分関数 g，および各 $i(1 \le i \le k)$ に対して，
$$g(u_i(y_i)\cdots u_1(y_1)\cdot s) \equiv g(u_i(y_i')\cdots u_1(y_1')\cdot s)$$

(3) A において，モジュール a の出力 f とモジュール a' の入力 x が接続されている（即ち，$x = f(s_a)$ が $VALID$ 関数の定義に含まれている）とする．$\theta(a)$ の出力および $\theta(a')$ の入力の並びを，それぞれ，G および Y とし，また，$EXP_x(Y)$ の中に現れる各入力（Y の要素）を，それと接続されている $\theta(a)$ の出力（G の要素）で置き換え得られる表現式を $EXP_x(G)$ と書く．このとき，$EXP_x(G)$ は $EXP_f(G)$ と同じ表現式となる．

(4) $VALID(< t_1(x_1)\cdot s_{a1}, \cdots, t_n(x_n)\cdot s_{an} >) \equiv 1$ を満たす（A の各モジュールの）任意の遷移 $t_1(x_1), \cdots, t_n(x_n)$ および任意の全体状態 $< s_{a1}, \cdots, s_{an} >$ に対して，B の各モジュールの遷移列 $w_1, \cdots, w_n$ が存在して，$s_{ai} \sim s_{bi}(1 \le i \le n)$ ならば，
$$VALID(< w_1\cdot s_{b1}, \cdots, w_n\cdot s_{bn} >) \equiv 1$$
但し，w_i は $\phi(t_i(x_i))$ に属する遷移列に適当に $idle(y)$ なる遷移を挿入して得られる遷移列である（即ち，w_i から遷移関数が $idle$ の遷移を削除した遷移列を w_i' とすると，$w_i' \in \phi(t(x_1))$）

(証明) [I] まず，B の全体状態に関する述語 IS の定義の仕方について述べる．B のモジュール b の状態 s_b に対して，s_b が初期状態または A の遷移の模倣を終えた直後の状態であるか否か（即ち，$s_b \in \phi(s_a)$ なる A のモジュールの状態 s_a が存在するか否か）を表す述語 is_b を定義する．遷移の対応が与えられているので，補助的な成分関数等を必要に応じて導入すれば，is_b を定義することは可能である．B の各モジュール b について is_b が得られると，IS を，それらの論理積として定義する．この IS は，B のすべてのモジュールが，対応する A のモジュールの遷移を模倣し終えた直後の状態にあるとき，かつそのときのみ真となる．

[II] 次に，実現の条件 **(1)** の代りに，より強い条件として，A のすべての $VALID$ 状態 S_A に対して，

$$HistI(S_A) \equiv EXP_I(HistI(S_B)) \quad (3.1)$$

$$HistC(S_A) \equiv EXP_C(HistC(S_B)) \quad (3.2)$$

を満たす B の $VALID$ 状態 S_B が存在することを，A の全体状態の遷移回数に関する帰納法により示す．ここで，$HistC$ は $HistO$ と同様にして定義した全成分関数の履歴である．

定義 3.1**(2)** と上記 [I] の IS の定義の仕方より，初期状態については明らかに上式が成り立つ．

k 回以下の状態遷移で到達する A のすべての全体状態に対して上式が成り立つと仮定する．k+1 回の状態遷移で到達する A の任意の $VALID$ 全体状態 $S_A =< t_1(x_1)\cdot s_{a1}, \cdots, t_n(x_n)\cdot s_{an} >$ を考える．帰納法の仮定より，$S_A' =< s_{a1}, \cdots, s_{an} >$ に対しては，式 (3.1) および (3.2)（S_A および S_B を，それぞれ，S_A' および S_B' で置

き換える）を満たすBの$VALID$全体状態$S'_B =< s_{b1},\cdots,s_{bn} >$が存在する．$s_{ai} \sim s_{bi}(1 \leq i \leq n)$であることと，定理の条件**(4)**，定義3.1**(2)(iii)**より，

$$VALID(w_1 \cdot s_{b1},\cdots,w_n \cdot s_{bn} >) \equiv 1$$

なるw_iが存在して，

$$t_i(x_i) \cdot s_{ai} \sim w_i \cdot s_{bi}$$

このことと，ISの定義の仕方より，$S_B =< w_1 \cdot s_{b1},\cdots,w_n \cdot s_{bn} >$とおくと，式(3.2)が成り立つ．また，式(3.1)も，定義3.1**(2)(i)**と帰納法の仮定より成立する．以上より，$k+1$回の状態遷移の場合にも式(3.1)，(3.2)が成り立ち，従って実現の条件**(1)**が満たされる．

[III] 最後に，実現の条件**(2)**を示す．Bの遷移回数に関する帰納法を用いる．初期状態については，実現**(2)(a)**が満たされる．

k回以下の状態遷移で到達するBのすべての全体状態$< s_{b1},\cdots,s_{bn} >$に対して，実現**(2)**を満足するAの全体状態$< s_{a1},\cdots,s_{an} >$が存在すると仮定する．$S_B =< u_1(y_1) \cdot s_{b1},\cdots,u_n(y_n) \cdot s_{bn} >$を$k+1$回の状態遷移で到達する$B$の任意の$VALID$全体状態とする．帰納法の仮定より，$S^*_B =< s_{b1},\cdots,s_{bn} >$については，実現**(2)**を満たす$S^*_B \leq S'_B$が存在する．まず，$S^*_B \neq S'_B$の場合を考える．定義3.1**(3)**，定理の**(2)(a)**より，$S'_B =< w_1 \cdot u_1(y'_1) \cdot s_{b1},\cdots,w_n \cdot u_n(y'_n) \cdot s_{bn} >$と表すことができる．定理の**(2)(b)**より，S'_Bの各y'_iをy_iで置き換えても成分関数の値は変らないので，定義(2)は満たされる．次に，$S^*_B = S'_B$の場合を考える．帰納法の仮定より，$s_{ai} \sim s_{bi}$であり，定義3.1**(2)(ii)**より，

$$valid(t_i(x_i) \cdot s_{ai}) \equiv 1$$

を満たすAの遷移$t_i(x_i)$が存在する．定義3.1**(2)(i)**と定理の**(3)**より，これらのモジュール状態からなる全体状態$< t_1(x_1) \cdot s_{a1},\cdots,t_n(x_n) \cdot s_{an} >$は，接続仕様を満たすので，$VALID$全体状態である．このことと，定理の**(4)**から，Bの$VALID$全体状態$S''_B =< w_1 \cdot s_{b1},\cdots,w_n \cdot s_{bn} >$が存在し，更に，定義3.1**(2)(iii)**と$idle$が成分関数値を変えないことから，$t_i(x_i) \cdot s_{ai} \sim w_i \cdot s_{bi}$が成り立つ．従って，この場合も，$S_B \leq S''_B$に関して実現**(2)**が満たされる．（証明終）

定理の条件**(1)**の遷移の対応の存在は，前にも述べたように，実現のより強い条件である．従って，これを示すことが，モジュールごとの実現検証に相当する．条件**(2)**は，Aの遷移を模倣するBの遷移列において，その最初の入力と状態によりその後の遷移が決まることを意味しており，条件**(3)**は，A，B間の接続仕様の対応関係を規定している．最後の条件**(4)**は，遷移の対応で与えられる各モジュールの遷移列の間で，適当に$idle(y)$なる遷移を挿入することにより，入出力が一致すること，即ち入出力の同期をとることができるという条件である．

この定理より，順序回路全体の実現検証を行うには，定理の条件の**(2)**および**(4)**を満たす遷移の対応を導き，A，B間の接続仕様が対応していること（条件**(3)**）を確認すればよいことがわかる．

4. 十分条件の実現検証への適用例

ここでは，**3.**の十分条件を適用した実現検証例を示す．ここでは簡単のため，同期式のリングアービタを回路例として採用する．アービタは，複数の装置からの資源専有要求を調停し，ただ一つの装置に資源を割り当てる回路であり，リングアービタは，リング上に接続された複数個の同一構造のモジュールからなる．以下，リングアービタを，単に，アービタと呼ぶ．アービタの仕様記述にあたって，通常のブール関数（演算）の定義は，既に得られているものとして省略する．

4.1 上位レベルの仕様

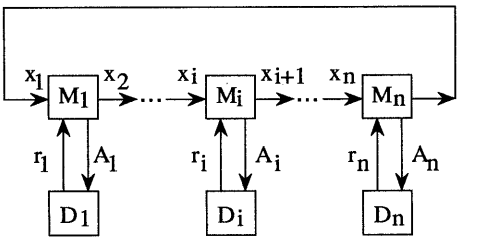

図1　上位レベルのリングアービタの構成

上位レベルとして，図1の構成のアービタを考える．各D_i $(i=1,\cdots,n)$は装置であり，M_iはD_iに対応して設けられたモジュールである．また，r_i（R入力と呼ぶ）は装置からの要求を伝え，A_i（A出力と呼ぶ）は装置への承認を伝える信号線である．リング上には専有権を表す信号（トークンと呼ぶ）がただ一つ存在し，信号線$x_1,\cdots,x_n$により，モジュールからモジュールへと伝えられる．x_iおよびx_{i+1}を，それぞれ，M_iのX入力およびX出力と呼ぶ．

トークンの転送のため，モジュールは次の状態遷移を起こす．

(1) トークンの受信

(2) トークンの送信（X出力を1にする）

(3) X出力のリセット（0にする）

(4) トークン待ち（出力・内部状態は変化しない）

そして，トークンを受信したとき，R入力が0なら直ちにトークンを送信するが，1ならば，要求を承認し（$A_i=1$），R入力が0になった時点でトークンを送信する．要求の承認からR入力が0になるまでの間，モジュールは上記**(4)**の遷移を繰り返すとする．

このようなアービタの仕様を，**2.**で述べた形式に従って，モジュールの仕様と接続の仕様に分けて記述する．まず，モジュールの仕様について述べる．

モジュールの初期状態を$init(c)$（cは初期状態のX出力を指定するブール値）で表し，上記**(1)**~**(4)**の状態遷移を，順に，遷移関数$receive$，$send$，$reset$，$idle$で表す．これら四つの遷移関数は，いずれも，遷移前の状態，X入力およびR入力を引数とし，遷移後の状態を返す．成分関数としては，以下を導入する．X出力およびA出力を表す成分関数を，それぞれ，$outX$および$outA$とし，モジュールにトークンが存在するか否かを表す成分関数を$token$とする．これらの成分関数は，いずれも，状態を引数とする関数であり，与えられた状態におけるそれぞれの値を返す．以上の関数の形式的な意味は，図2公理$A1 \sim A15$で定義される．状態遷移がいつ起こるかは，$valid$関数を用いて記述する．$receive$は，X入力が1で，かつトークンを保持していないとき起こり，$send$は，トークンを保持しており，かつ要求を承認していないか或はR入力が0になったとき起こる．また，$reset$はX出力が1のとき起こり，$idle$は，上記のいずれも起こらないときに起こる．従って，$valid$関数の定義は公理$A16 \sim A20$で与えられる（初期状態に対しては常に真とする）．

次に，接続の仕様について述べる．アービタの全体状態をn個のモジュールの状態の並びで表す．ある時点でモジュールM_iが$send$を起こすと，次の時点でM_iが$reset$，M_{i+1}が$receive$を起こす．また，装置が資源を専有している間は全モジュール

が *idle* を繰り返すので，各モジュールの状態遷移の可能な組合せは

(1) $< \cdots, idle, send, idle, \cdots >$

(2) $< \cdots, idle, reset, receive, idle, \cdots >$

(3) $< idle, \cdots, idle >$

これらについて，接続仕様を記述すると，*A22* ～ *A24* の形の公理が得られる．但し，これらの公理では，$idle(s_j, x_j, r_j)$ を ID_j と略記している．なお，初期状態では，M_1 の X 出力のみを 1 としている（*A21*）．

$$
\begin{aligned}
&A1: \ token(init(c)) == 0;\\
&A2: \ outX(init(c)) == c;\\
&A3: \ outA(init(c)) == 0;\\
\\
&A4: \ token(receive(s,x,r)) == 1;\\
&A5: \ outX(receive(s,x,r)) == 0;\\
&A6: \ outA(receive(s,x,r)) == r;\\
\\
&A7: \ token(send(s,x,r)) == 0;\\
&A8: \ outX(send(s,x,r)) == 1;\\
&A9: \ outA(send(s,x,r)) == 0;\\
\\
&A10: \ token(reset(s,x,r)) == 0;\\
&A11: \ outX(reset(s,x,r)) == 0;\\
&A12: \ outA(reset(s,x,r)) == 0;\\
\\
&A13: \ token(idle(s,x,r)) == token(s);\\
&A14: \ outX(idle(s,x,r)) == outX(s);\\
&A15: \ outA(idle(s,x,r)) == outA(s);\\
\\
&A16: \ valid(init(c)) == 1;\\
&A17: \ valid(receive(s,x,r)) == valid(s) \ \ and\\
&\qquad\qquad x \ \ and \ \ not \ \ token(s);\\
&A18: \ valid(send(s,x,r)) == valid(s) \ \ and\\
&\qquad token(s) \ \ and \ \ \{not \ \ r \ \ or \ \ not \ \ outA(s)\};\\
&A19: \ valid(reset(s,x,r)) == valid(s) \ \ and\\
&\qquad\qquad outX(s) \ \ and \ \ not \ \ x;\\
&A20: \ valid(idle(s,x,r)) == valid(s) \ \ and\\
&\qquad \{(not \ \ outX(s) \ \ and \ \ not \ \ token(s) \ \ and \ \ not \ \ x) \ \ or\\
&\qquad (outA(s) \ \ and \ \ r)\};\\
\\
&A21: \ VALID(< init(c_1), \cdots, init(c_n) >) ==\\
&\qquad\qquad c_1 = 1 \ \ and \ \ \bigwedge_{i=2}^{n} c_i = 0;\\
\\
&A22: \ VALID(< ID_1, \cdots, send(s_i, x_i, r_i), \cdots, ID_n >) ==\\
&\qquad VALID(< s_1, \cdots, s_n >) \ \ and\\
&\qquad \bigwedge_{j=1}^{i-1} valid(ID_j) \ \ and \ \ valid(send(s_i, x_i, r_i)) \ \ and\\
&\qquad \bigwedge_{k=i+1}^{n} valid(ID_k) \ \ and\\
&\qquad x_1 = outX(s_n) \ \ and \ \ \bigwedge_{j=2}^{n} x_j = outX(s_{j-1});\\
\\
&A23: \ VALID(< ID_1, \cdots, reset(s_{i-1}, x_{i-1}, r_{i-1}),\\
&\qquad\qquad receive(s_i, x_i, r_i), \cdots, ID_n >) ==\\
&\qquad VALID(< s_1, \cdots, s_n >) \ \ and\\
&\qquad \bigwedge_{j=1}^{i-2} valid(ID_j) \ \ and \ \ valid(reset(s_{i-1}, x_{i-1}, r_{i-1})) \ \ and\\
&\qquad valid(receive(s_i, x_i, r_i)) \ \ and \ \ \bigwedge_{k=i+1}^{n} valid(ID_k) \ \ and\\
&\qquad x_1 = outX(s_n) \ \ and \ \ \bigwedge_{j=2}^{n} x_j = outX(s_{j-1});\\
\\
&A24: VALID(< ID_1, \cdots, ID_n >) ==\\
&\qquad VALID(< s_1, \cdots, s_n >) \ \ and\\
&\qquad \bigwedge_{j=1}^{n} valid(ID_j) \ \ and\\
&\qquad x_1 = outX(s_n) \ \ and \ \ \bigwedge_{i=2}^{n} x_j = outX(s_{j-1});
\end{aligned}
$$

図 2　上位レベルの仕様記述

4.2　下位レベルの記述

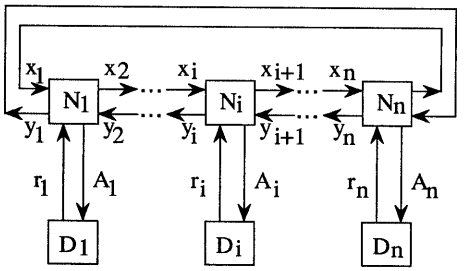

図 3　下位レベルのリングアービタの構成

下位レベルとして，図 3 の構成のアービタを考える．図 1 の構成とは，モジュール間に，トークン受信を知らせるための信号線 y_i の引かれている点が異なる．下位レベルでは，この信号の送受のため，状態遷移の種類が下記のように増えている．信号線 y_i および y_{i+1} の値を，それぞれ，モジュール N_i の Y 出力および Y 入力と呼び，他の信号線の値については，上位レベルで使用した名称を用いる．

(a) X 入力が 1 になることによるトークンの受信．Y 出力を 1 にし，更に R 入力が 1 なら A 出力も 1 にする．

(b) X 入力が 0 になることによる，Y 出力のリセット．

(c) R 入力が 0 または要求を承認していない（$A = 0$）場合のトークンの送信．要求を承認していた（$A = 1$）の場合，A 出力を 0 にする．

(d) Y 入力が 1 になることによる，X 出力のリセット．

(c) を除く状態遷移は入力変化により起こる．従って，これらの状態遷移は，信号変化を考慮すると次のようになる．例えば，N_i で状態遷移 **(a)** が起こるとき，その時点では，N_{i-1} で **(d)** は起こらず，次の時点で **(d)** が起こる．このように **(c)** を除く状態遷移前には，常に，上位レベルの **(4)** の状態遷移と同じ

(e) 内部状態や出力の変化しない状態遷移

が起こる．

このような下位レベルのアービタの仕様を図 4 に示す．この仕様も上位レベルと同様に記述しているので，詳細を省略し，以下，モジュールの仕様については，導入した関数の意味についてのみ述べる．初期状態を $init'(c)$（c の意味は上位レベルに同じ）で表す．状態遷移については，上記の **(a)** を *recX*，**(b)** を *resetY*，**(c)** を *sendX*，**(d)** を *resetX*，**(e)** を *idle* で表す．これらの遷移関数は，遷移前の状態，X 入力，Y 入力および R 入力を引数とする．成分関数については，X 出力，Y 出力および A 出力を，それぞれ，X，Y および A で表し，トークンの存在を表す成分関数を T で表す．

接続の仕様については，上述の状態遷移に関する考察より，全体状態の遷移は次の **(1)** ～ **(5)** に分類されるので，図 4 の *B22* ～ *B27* の形の公理が得られる．

(1) $< \cdots, idle, recX, idle, \cdots >$

(2) $< \cdots, idle, resetY, idle, \cdots >$

(3) $< \cdots, idle, sendX, idle, \cdots >$

(4) $< \cdots, idle, resetX, idle, \cdots >$

(5) $< idle, \cdots, idle >$

なお，これらの公理でも，$idle(s_j, x_j, r_j)$ を ID_j と略記し，初期状態では，N_1 の X 出力のみを 1 としている．

$B1: \ X(init'(c)) == c;$
$B2: \ Y(init'(c)) == 0;$
$B3: \ A(init'(c)) == 0;$
$B4: \ T(init'(c)) == 0;$

$B5: \ X(recX(s,x,y,r)) == 0;$
$B6: \ Y(recX(s,x,y,r)) == 1;$
$B7: \ A(recX(s,x,y,r)) == r;$
$B8: \ T(recX(s,x,y,r)) == 1;$

$B9: \ X(resetY(s,x,y,r)) == X(s);$
$B10: \ Y(resetY(s,x,y,r)) == 0;$
$B11: \ A(resetY(s,x,y,r)) == A(s);$
$B12: \ T(resetY(s,x,y,r)) == T(s);$

$B13: \ X(sendX(s,x,y,r)) == 1;$
$B14: \ Y(sendX(s,x,y,r)) == 0;$
$B15: \ A(sendX(s,x,y,r)) == 0;$
$B16: \ T(sendX(s,x,y,r)) == 0;$

$B17: \ X(resetX(s,x,y,r)) == 0;$
$B18: \ Y(resetX(s,x,y,r)) == 0;$
$B19: \ A(resetX(s,x,y,r)) == 0;$
$B20: \ T(resetX(s,x,y,r)) == T(s);$

$B21: \ X(idle(s,x,y,r)) == X(s);$
$B22: \ Y(idle(s,x,y,r)) == Y(s);$
$B23: \ A(idle(s,x,y,r)) == A(s);$
$B24: \ T(idle(s,x,y,r)) == T(s);$

$B25: \ valid(init'(c)) == 1;$
$B26: \ valid(recX(s,x,y,r)) == valid(s) \ and$
$\quad x \ and \ not \ Y(s);$
$B27: \ valid(resetY(s,x,y,r)) == valid(s) \ and$
$\quad not \ x \ and \ r \ and \ A(s) \ and \ Y(s);$
$B28: \ valid(sendX(s,x,y,r)) == valid(s) \ and \ not \ x \ and$
$\quad \{(not \ r \ and \ A(s)) \ or \ (not \ A(s) \ and \ Y(s))\};$
$B29: \ valid(resetX(s,x,y,r)) == valid(s) \ and$
$\quad not \ x \ and \ y \ and \ X(s);$
$B30: \ valid(idle(s,x,r)) == not \ \{valid(recX(s,x,y,r)) \ or$
$valid(resetY(s,x,y,r)) \ or$
$valid(sendX(s,x,y,r)) \ or$
$valid(resetX(s,x,y,r))\};$

$B31: \ VALID(< init'(c_1), \cdots, init'(c_n) >) ==$
$c_1 = 1 \ and \ \bigwedge_{j=2}^{n} c_j = 0;$

$B32: \ VALID(< ID_1, \cdots, recX(s_i,x_i,y_i,r_i), \cdots, ID_n) >) ==$
$VALID(< s_1, \cdots, s_n >) \ and$
$\bigwedge_{j=1}^{i-1} valid(ID_j) \ and \ valid(recX(s_i,x_i,y_i,r_i)) \ and$
$\bigwedge_{k=i+1}^{n} valid(ID_k) \ and$
$x_1 = X(s_n) \ and \ \bigwedge_{j=2}^{n} x_j = X(s_{j-1})$
$\bigwedge_{j=1}^{n-1} y_j = Y(s_{j+1}) \ and \ y_n = Y(s_1);$

$B33: \ VALID(< ID_1, \cdots, resetY(s_i,x_i,y_i,r_i), \cdots, ID_n >) ==$
$VALID(< s_1, \cdots, s_n >) \ and$
$\bigwedge_{j=1}^{i-1} valid(ID_j) \ and \ valid(resetY(s_i,x_i,y_i,r_i)) \ and$
$\bigwedge_{k=i+1}^{n} valid(ID_k) \ and$
$x_1 = X(s_n) \ and \ \bigwedge_{j=2}^{n} x_j = X(s_{j-1})$
$\bigwedge_{j=1}^{n-1} y_j = Y(s_{j+1}) \ and \ y_n = Y(s_1);$

$B34: \ VALID(< ID_1, \cdots, sendX(s_i,x_i,y_i,r_i), \cdots, ID_n >) ==$
$VALID(< s_1, \cdots, s_n >) \ and$
$\bigwedge_{j=1}^{i-1} valid(ID_j) \ and \ valid(sendX(s_i,x_i,y_i,r_i)) \ and$
$\bigwedge_{k=i+1}^{n} valid(ID_k) \ and$
$x_1 = X(s_n) \ and \ \bigwedge_{j=2}^{n} x_j = X(s_{j-1})$
$\bigwedge_{j=1}^{n-1} y_j = Y(s_{j+1}) \ and \ y_n = Y(s_1);$

$B35: \ VALID(< ID_1, \cdots, resetX(s_i,x_i,y_i,r_i), \cdots, ID_n >) ==$
$VALID(< s_1, \cdots, s_n >) \ and$
$\bigwedge_{j=1}^{i-1} valid(ID_j) \ and \ valid(resetX(s_i,x_i,y_i,r_i)) \ and$
$\bigwedge_{k=i+1}^{n} valid(ID_k) \ and$
$x_1 = X(s_n) \ and \ \bigwedge_{j=2}^{n} x_j = X(s_{j-1})$
$\bigwedge_{j=1}^{n-1} y_j = Y(s_{j+1}) \ and \ y_n = Y(s_1);$

$B36: \ VALID(< ID_1), \cdots, ID_n) >) ==$
$VALID(< s_1, \cdots, s_n >) \ and$
$\bigwedge_{j=1}^{n} valid(ID_j) \ and$
$x_1 = X(s_n) \ and \ \bigwedge_{j=2}^{n} x_j = X(s_{j-1})$
$\bigwedge_{j=1}^{n-1} y_j = Y(s_{j+1}) \ and \ y_n = Y(s_1);$

図 4　下位レベルの仕様記述

5. 実現検証

5.1 モジュールの対応 θ と遷移の対応 ϕ

上位のモジュール集合から下位のモジュール集合への写像 θ を

$$\theta(M_i) = N_i$$

と定義し，X 入力同士および R 入力同士がそのまま対応する入力の対応を EXP_I とする．(B の Y 入力は A のいずれの入力にも対応しない．) また，成分関数値の対応 EXP_C および遷移の対応 ϕ を，それぞれ，表 1 および表 2 のように定義する．

表 1　成分の対応

上位 A	下位 B
$token(s_a)$	$T(s_b)$
$outX(s_a)$	$X(s_b)$
$outA(s_a)$	$A(s_b)$

表 2　遷移の対応

$t(x)$	$\phi(t(x))$
$init(c)$	$init'(c)$
$receive(1,r)$	$resetY(0,y',r') \cdot recX(1,y,r)$
$send(x,r)$	$sendX(x,y,r)$
$reset(0,r)$	$resetX(0,1,r)$

表 2 では，$valid$ 関数を真にするような遷移のみを挙げている．また，引数の変数は，その変数の各値について対応関係があることを表す．例えば，$receive(1,0)$ は，

$$\{resetY(0,y',r') \cdot recX(1,y,0) \mid 0 \le y,y',r' \le 1\}$$

と対応することを表している．以下，この ϕ が定義 3.1 (遷移の対応) の条件を満足していることを示す．

まず，**(1)** については，公理 $A1 \sim A3$, $B1 \sim B3$ と表 1 より明らかである．次に **(2)** について考える．**(i)** は表 2 より，**(iii)** は公理 $A4 \sim A15$, $B4 \sim B24$ より，また，**(ii)** は，$send$ 以外の場合，公理 $A16$, $A17$, $A19$, $B25$, $B26$, $B29$ より明らかである．$send$ の場合，公理 $A18$ と $B28$ を比べると，後者には条件として"$not \ Y(s)$"が加わってる．ここで，公理から，A と B の各遷

移（*idle* を除く）の可能なシーケンスを調べると，A は *receive*, *send*, *reset* の順に，また Bは *recX*, *resetY*, *sendX*, *resetX* の順に遷移が起こり，かつこの順だけであることが分かる．従って，*send* についても **(ii)** が成り立つことが分かる．最後に **(3)** について調べる．遷移の対応より，Bの *recX* と *resetY* 以外は A の遷移に単独で対応しており，**(3)** を満たすことは明らかである．*recX* と *resetY* の場合も，上述のシーケンスに関する考察より *recX*, *resetY* の順にのみ遷移が起こることと，*receive* に対応する遷移列が *resetY* · *recX* であることから，**(3)** を満たす．

以上より，ここで定義したϕは遷移の対応である．

5.2 定理の適用

まず，$EXP_I, EXP_C, \theta, \phi$として，**5.1** で定義したものを用いれば，条件の **(1)** は満たされる．また **(3)** も，X入出力の接続の仕方は A と Bで一致しているため（公理 $A22 \sim A24, B32 \sim B36$ 参照），満たされることは明らかである．

次に **(2)** について考える．**(a)** は，ϕの定義より明らかである．**(b)** については，*recX* 以外の遷移後の成分関数値は入力に依存せず，*recX* も，R入力に依存して A 出力の値が変るだけである．従って，A の遷移 *receive*(x,r), *send*(x,r), *reset*(x,r) の入力値 x, rを 0，1 のいずれかに固定し，それぞれに対応する Bの遷移列の集合を考えるとその集合に属する遷移列は **(b)** を満たす．

最後に **(4)** について考える．*VALID* 関数を 1 にする A の全体状態の遷移には，次の三つのタイプのものがある（簡単のため遷移関数のみを記す）．

(a1) $< \cdots, idle, reset, receive, idle, \cdots >$

(a2) $< \cdots, idle, idle, send, idle, \cdots >$

(a3) $< \cdots, idle, idle, idle, idle, \cdots >$

これらのそれぞれに，以下の遷移列を対応付けると，**(4)** の満たされることが分かる．

(b1) $< \cdots, idle, idle, receiveX, idle, \cdots >$
$< \cdots, idle, resetX, idle, idle, \cdots >$
$< \cdots, idle, idle, resetY, idle, \cdots >$

(b2) $< \cdots, idle, idle, sendX, idle, \cdots >$

(b3) $< \cdots, idle, idle, idle, idle, \cdots >$

以上より，**5.1** で定義した $EXP_I, EXP_C, \theta, \phi$は定理の条件を満足し，従って，4.2 の仕様 Bは，$< EXP_I, EXP_C >$のもとで 4.1 の仕様 A の実現である．

6. あとがき

本論文では，順序回路を対象に，モジュール化された代数的仕様間の実現関係の定義を行い，モジュールごとの実現検証により順序回路全体の実現関係が成り立つための十分条件を示した．また，リングアービタを対象に，その十分条件の適用例を示した．

一般に，実現検証では，下位の仕様で定義されているすべての動作（入力と成分関数値のすべての組合せに対する動作）を調べる必要があるので，モジュールごとに行なうことによる実現検証の手間の削減効果は大きい．なかでも，リングアービタのように同一モジュールで構成される順序回路では，モジュールの実現検証を 1 組のモジュールについてのみ行えばよく，その効果は特に大きい．

本論文の十分条件では，上下間の入出力や遷移の対応が与えられるものとしており，また，下位のモジュール間で同期をとるための *idle* の挿入の仕方が分かっているものとしている．下位の記述が上位を詳細化して得られたものである場合，これらは詳細化の段階で明確に意識されており，それらを与えることが書き手の負担となることは少ないと思われる．

本論文では，議論を簡明にするために，上下の仕様でモジュールが 1 対 1 に対応するという制限を設けている．この制限を解除することは困難なことではないが，それに伴う条件が新たに必要となる．この条件を明らかにすることと，十分条件判定手順の自動化が今後の課題として残されている．

参考文献

[1] 杉山，北道，谷口:“代数的手法を用いた順序回路の記述法とその詳細化について”，信学技報，Vol.88，No.55，COMP88-7，pp.61-70(1988-05).

[2] Yurong HOU, Atsushi OHNISHI, Yuji SUGIYAMA and Takuji OKAMOTO: “An Algebraic Specification of a Daisy Chain Arbiter”, IEICE Trans. Inf. & Syst., VOL.E75-D, 6, pp.778-783(1992-11).

[3] 菰口，侯，杉山:“モジュール化された順序回路の代数的仕様の実現検証について”，平成 4 年度 電気・情報関連学会中国支部第 43 回連合大会，p.369(1992-10).

[4] 杉山，谷口，嵩:“基底代数を前提とする代数的仕様”，信学論，J64-D，4，pp.324-331(1981).

[5] 嵩，谷口，杉山，関:“代数的言語 ASL*-意味定義を中心に-”，信学論，J69-D，7，pp.1066-1074(1986-7).

[6] 東野，森，谷口，嵩:“HDLC 手順の代数的記述”信学論（D），J66-D，7，pp.773-780(1983).

[7] 川原林，太田，大山口，稲垣:“代数的仕様記述に基づいたソフトウェアの正当性証明システム”，信学論（D），J66-D，6，pp.691-698(June 1983).

[8] 東野，工藤，縄田，杉山，谷口:“代数的仕様検証支援系及びそれを用いた検証例”，信学論（D），J67-D，4，pp.472-479(1984).

[9] 岡本，藤原:“非同期式リングアービタの一設計法”，信学論，J64-D,9,pp.846-853(1981-9).