一次元反復回路による計算の高速化とその一般化

DESIGN OF HIGH-SPEED ITERATIVE COMBINATIONAL CIRCUITS AND ITS GENERALIZATION

上林 弥彦
Yahiko KAMBAYASHI
京都大学工学部
Faculty of Engineering, Kyoto University, Kyoto

1. まえがき

非同期論理回路においては，計算時間は入力の組合せによって大きく影響されることがある。出力を利用するためには，計算時間の最も長くかかるような入力に必要な時間だけ待てばよいがこれでは時間的な損失が大きすぎる。Waiteは，非同期の一次元反復回路に対して計算終了信号という考え方を導入した。[2] 最近，Ungerはその結果を改良している。[3] 本論文では，部分的計算終了信号という考え方を導入して，さらに計算時間の面で改良できることを示す。この手法は，さらに一般の非同期回路にも適用することができるので，そのような一般化についても考察した。また，この回路を若干変更すれば高速性を維持したままで，故障の自己検出能力を持たせることができるので，それらの点についても考察した。

一次元反復回路は，同一素子を一次元接続したもので，一つの単純な並列処理システムとしての理論的な興味のほか，実用的な計算機回路の構成に用いられている。本論文では，図1に示すように，信号の方向が一方向であるような一次元一方向反復回路を対象とする。何桁かの算術演算を行う回路は，基本単位のモジュールをこのモデルの形に接続して実現できる。この

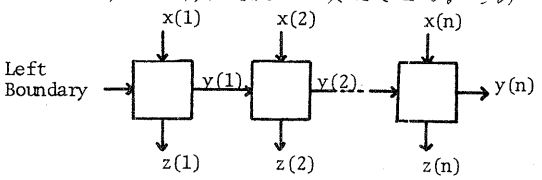


図1 一次元一方向反復回路

ように，このモデルは実用的に重要であるほか有限オートマトンとの対応付けが容易で，理論的に扱い易い特色がある。しかし，このような形式で実現された回路では，モジュールの数をn個とし，個々のモジュールでの最長計算時間をdとすると，入力を加えてからnd単位時間後以前では出力が正しく計算されている保証はない。すなわち，いつもnd単位時間待たなければならないことになる。しかし，たとえば，加算回路で，あるモジュールへの入力（加えられるべき2つの2進数のそれ桁目）が共に0であれば，そのモジュールへの左のモジュールからの入力（キャリー）の値にかかわらず，そのモジュールからのキャリーは0と決ってしまっている。したがって，この右側のモジュールの計算を始めることができる。このような考えを一般化して，計算終了時間を早めたのがWaiteの方法である。Ungerは，この考えを拡張して，対応するオートマトンが同期系列を持つ場合について，計算終了時間をさらに早めることができることを示した。ここでは，部分計算という考え方を導入して，対応するオートマトンが同期系列を持たない場合にも適用できる方法を示した。計算終了信号は，各モジュールにおいて，計算が完全に終了してから出力されるものであるが，部分的計算終了信号は，出力の可能な値の範囲を示すものである。たとえば，モジュール間信号として，{0, 1, 2, 3}の4つの可能性のある場合に，計算の途中で，0か1かのどちらかであることが判ると，この結果を利用することができる。Ungerのモデルでは，ハザードを避け

るため，信号を多線で表現しているので，このような方法に変更しても，ほとんど複雑化しない。このような考えを拡張して，非同期回路を

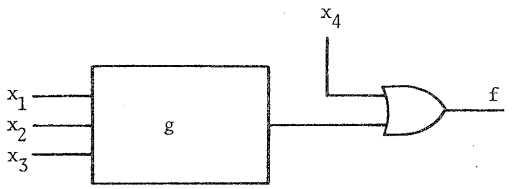


図2　非同期論理回路の例

いくつかのモジュールで構成した場合に，回路の高速化をはかることができる。たとえば，図2の回路は，4変数論理回路を実現しており，$f=g(x_1,x_2,x_3)+x_4$の形になっている。もし，$x_4=1$であれば，gの部分の計算結果を待たないで$f=1$であることが判る。部分的計算結果を用いると，このような性質を生かした回路構成ができ，計算の終了した時点を知ることができる。

この回路のための各モジュールは，入力変数を適当に変換すれば，正関数のみを使って実現できることが示される。したがって，フェイルセイフ論理回路のように，各モジュールを構成する素子（正関数を実現するとする）が，1縮退故障しかしない場合には，出力は正しい値を出すか，正しい値を含む集合を出すこととなる。0縮退故障はこのままでは扱えないが，モジュールの構成をすこし変更すると扱えるようにすることができる。最終的に得られた回路では，0縮退故障と1縮退故障が混在しなければ，出力に影響を与える故障（複数は許される）を回路の出力端子で検出することができる。

このように，計算の部分的終了の情報を利用する回路構成を用いると，回路の高速化とともに，故障の自己検出能力を持たせることができる。

2．一次元反復回路による計算の高速化

2.1　Ungerの方法

図1に示される一次元一方向反復回路は，有限オートマトンを横方向に展開したものとみなすことができる。ここでX(i)，Y(i)，Z(i)は，時刻iにおける入力，状態，出力に対応することになる。この対応関係から，一次元一方向反復回路は，オートマトンの遷移表を用いて表現する

	x=0	x=1
1	1/0	3/0
2	2/0	3/1
3	2/1	1/0

図3　Ungerの例

	x=0	x=1	y_1 y_2 y_3
0	0/-	0/-	0 0 0
1	1/0	3/0	1 0 0
2	2/0	3/1	0 1 0
3	2/1	1/0	0 0 1 (a)

$Y_1=\bar{x}y_1+xy_3$
$Y_2=\bar{x}y_2+\bar{x}y_3$
$Y_3=xy_1+xy_2$
$Z=\bar{x}y_3+xy_2$
(b)

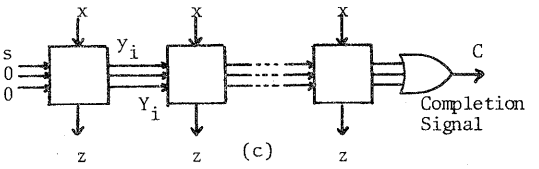


図4　基本的な実現法

ことができる。図3は，Ungerの用いた遷移表を示したものである。状態数が3であるため，状態の表現に重み1の符号を用いた方法が図4に示されている。ここで，(0，0，0)は初期状態で，図4(c)の左端のsを0とすると，すべてのモジュール間の状態をこの状態へリセットできる。計算が終了すれば，右端の出力状態が(0，0，0)ではなくなるので，計算終了信号Cが1となる。このような基本的な方法では，常に左端から右端へ信号が伝ぱんする時間だけ待たなければならない。

図5は，加算器の構成を示したものである。

	ab=00	01	11	10	y_1 y_2
0	1/-	0/-	2/-	0/-	0 0
1	1/0	1/1	2/0	1/1	1 0
2	1/1	2/0	2/1	2/0	0 1

(a)

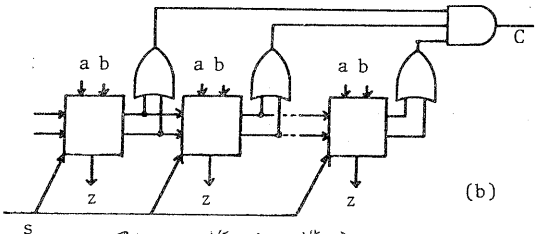


図5　加算器の構成

図5(a)は，遷移表を示しており，この場合，入力(0,0)では，現在の状態がどの状態であっても，次状態は1となる。入力(1,1)でも同じく次状態は2となる。このような列を，定状態列と呼ぶ。定状態列を持つ遷移表に対しては，より短時間で計算終了信号を出すことができる。この例の場合は，計算は左端のモジュールからのみならず，入力が(0,0)または(1,1)であるモジュールからも始めることができる。すべてのモジュール間の状態が(0,0)でなくなれば計算終了であるので，図5(b)のような構成で計算終了信号Cを発生できる。ただし，この場合，リセット信号sは，すべてのモジュールに加える必要がある。

Ungerは，この定状態列の考えを一般化して，定状態列を持たない場合でも同期系列を持てば計算終了時間の計算が短縮されることを示した。

〔定義1〕 オートマトンMがどのような状態にあっても，系列ωを加えると特定の状態に遷移する場合，Mは同期系列ωを持つという。

同期系列を持たないオートマトンも存在するし，逆にいくつかの同期系列を持つオートマトンも存在する。遷移表に定状態列を持つオートマトンは，長さ1の同期系列を持っていることに対応する。Ungerの方法は，長さkの同期系列を持てば，モジュールをk個合せたものを1つのモジュールとすれば，等価的に定状態列を持つオートマトンとなることを利用したものである。

図3の例では，01が同期系列となっており，任意の状態から状態3へ遷移する。図6にこの遷移図を示す。このモジュールを基本単位にすると，必要なモジュール数は半分となるが，1つのセルの構造は複雑になる。図7は，同期系列をより有効に利用した構成で，モジュールの作り方は以前とほぼ同じであるが，$x'=0$，xが1という入力の場合にだけ次状態を3になるようにしている。この方法では，図6の方法と異りモジュール数は減らないが，図6の方法では同期系列が2つのモジュールにまたがる(あるモジュールに(*,0)と加えられ次のモジュールに(1,*)と加えられた場合)ときには利用できなかったがこの方法では利用できる点が有利になっている。

	x_1x_2 00	01	11	10
1	1/00	3/00	1/00	2/01
2	2/00	3/01	1/10	2/11
3	2/10	3/11	3/00	1/00

図6　図3を長さ2の入力で表現した遷移図

	x=0	x=1	y_1	y_2	y_3
0	0/-	0/-	0	0	0
1	1/0	3/0	1	0	0
2	2/0	3/1	0	1	0
3	2/1	1/0	0	0	1

(a)

$Y_1=s\bar{x}y_1+sxy_3$
$Y_2=s\bar{x}y_2+s\bar{x}y_3$
$Y_3=sxy_1+sxy_2+sx\bar{x}'$
$Z=\bar{x}y_3+xy_2$

(b)

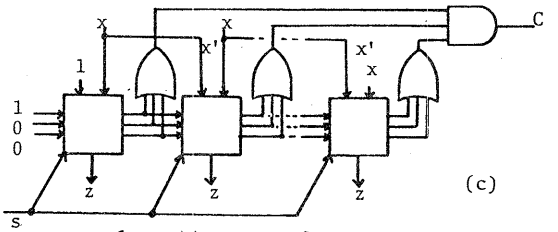


図7　Ungerの方法

2.2　部分的計算結果を用いる方法

Ungerの方法で計算速度は改良されることも多いが，つぎのような点で改良の余地がある。

(1) 対応するオートマトンが同期系列を持たない場合には適用できない。

(2) もし，対応するオートマトンが同期系列を持っていたとしてもそれが長いと一つのモジュールへの入力数が増加する。

(3) 対応するオートマトンがいくつかの同期系列を持つ場合には，これらをすべて利用するためには，モジュール構造が複雑になることがある。

ここで述べる部分的計算結果を用いる方法では，上記の問題を解決することができる。状態が4つあった場合のUngerの表現法は図8(a)に示されている。ここでは，逆に(1,1,1,1)をリセット状態として用いることにする(図8(b))。

state	representation
1	1 0 0 0
2	0 1 0 0
3	0 0 1 0
4	0 0 0 1
Init.	0 0 0 0

(a)

state	representation
1	1 0 0 0
2	0 1 0 0
3	0 0 1 0
4	0 0 0 1
1,2	1 1 0 0
1,3	1 0 1 0
:	:
Init.	1 1 1 1

(b)

図8　状態の表現法

Time	y_1	y_2	y_3	y_4	Possible States
T_1	1	1	1	1	$s_1s_2s_3s_4$
T_2	0	1	1	1	$s_2s_3s_4$
T_3	0	1	0	1	s_2s_4
T_4	0	1	0	0	s_4

図9　可能な状態の表現

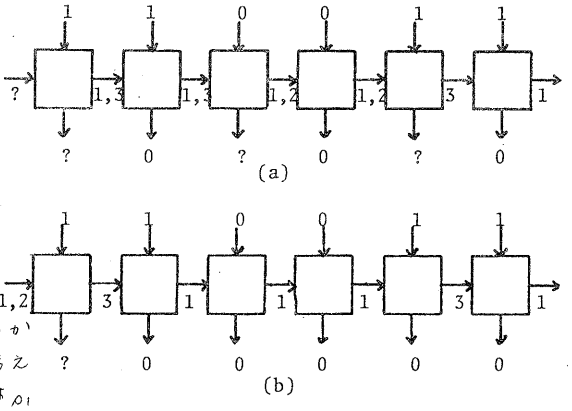


図10　可能な状態

この状態は，A_1，A_2，A_3，A_4のどれであるかが不明であるということを示しているとも考えられる。さらに，たとえば，(1,1,0,0)はρ_1かρ_2のどちらであるか不明であることを示すものとする。このように，図8(b)の表現を用いると，(0,0,0,0)は使われない組合せとなる。図9に，時間とともに信号の変化した場合の解釈が示されている。すなわち，T_1においては状態は全く不明であるが，T_2においては可能な状態はρ_2，ρ_3，ρ_4となっている。時間の経過に伴って可能な状態数は減少し，T_4においてρ_4に確定する。このように，図8(b)の表現を用いた場合でも，各変数は1から0にしか変化しないので，図8(a)の場合と同様にハザードの心配はない。

計算の途中では，このように左側のモジュールより可能な状態集合を示す信号が送られてくる。この状態集合と入力を用いて，可能な次状態を計算する。この方法により，

(1) すべての同期系列が利用できる。

(2) 同期系列を持たない場合にも高速化ができることが多い。

これらの点で，Ungerの方法の改良となっているが，さらに，

(3) 次状態が未定であっても出力が決定する場合があるので，その場合にはそのモジュールより計算終了信号を出すことができ，さらに高速化できる可能性がある。

たとえば，図6によると，入力が00または11と続けば次状態は一意ではないが，出力は共に0となることが判る。このような場合にもこのモジュールは計算終了とみなしうる。

図10(a)は，同じ図3の例について，左端の状態が不明の場合に入力11001が加えられたときの可能な状態や出力を示したものである。入力が00または11と続くと出力は0と確定している。図10(b)は，左端の状態が1か2のどちらかであると判った場合の可能な状態や出力を示したものである。この場合には，この状態が確定すると，1モジュールに要する計算時間のみで全体の計算が終了する。

このような動作をするモジュールを設計するために，状態の部分集合に対する遷移図を用いる。

〔定義 2〕　オートマトンMに対して，状態の部分集合に対する遷移図は次のように定義される。(1) 遷移図の各節点は，Mの状態の部分集合に対応している。(2) 遷移図の枝で，状態集合S_1に対応する節点から，状態集合S_2に対応する節点へむかう枝は，次の条件が満足されれば，そのときのみ存在する。

$$S_2 = \{\delta(\rho, x) \mid x = \text{fixed}, \rho \in S_1\}$$

ここでδはオートマトンの次状態関数である。この枝には次のようなラベルが付けられる。

$$x / \{\lambda(\rho, x) \mid \rho \in S_1\}$$

ここでλはオートマトンの出力関数である。

図11には，図2のオートマトンの状態の部分集合に対する遷移図が示されている。ここで，たとえば，節点{1,2,3}より節点{1,3}への枝(ラベルが1/0,1)は，状態が未定の場合に入力1が与えられると，出力は未定であるが可能な状態集合は{1,3}となることを示している。節点{1,3}から{1,3}への枝(ラベルが1/0)は，可能な状態集合が{1,3}であって入力1が加えられると，出力は0となり，可

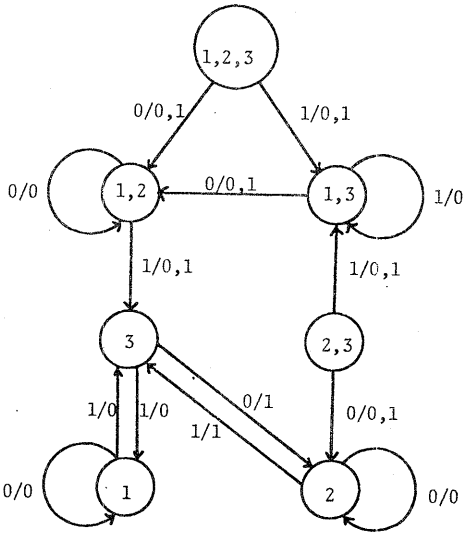
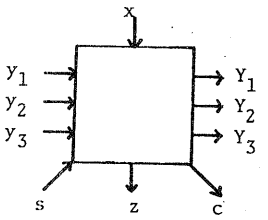


図11　状態の部分集合に対する遷移図



(a)

y_1	y_2	y_3	x	Y_1	Y_2	Y_3	z	c
1	1	1	0	1	1	0	-	0
1	1	1	1	1	0	1	-	0
1	1	0	0	1	1	0	1	1
1	1	0	1	0	0	1	-	0
1	0	1	0	1	1	0	-	0
1	0	1	1	1	0	1	0	1
1	0	0	0	1	0	0	0	1
1	0	0	1	0	0	1	0	1
0	1	0	0	0	1	0	0	1
0	1	0	1	0	0	1	1	1
0	0	1	0	0	1	0	1	1
0	0	1	1	1	0	0	0	1

(b)

$Y_1 = sy_3x + sy_1\bar{x}$
$Y_2 = s\bar{x}y_3 + sy_2\bar{x}$
$Y_3 = sy_1x + sy_2x$
$Z = y_1y_2 + y_2x + y_3\bar{x}$
$C = s\bar{y}_1 + s\bar{y}_2x + s\bar{y}_3\bar{x}$

(c)

能な状態集合がまに{1,3}になることを示している。節点{2,3}へは節点{1,2,3}より到達不能であり、これは、状態2かるのどちらかであるというあいまいさは起り得ないことを示している。

各モジュールは次の方法で設計される。

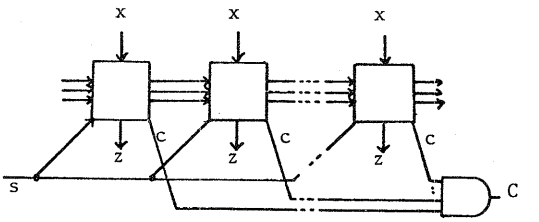
(1) 状態の部分集合に対する遷移図を求める。

(2) すべての状態よりなる集合に対応する節点より到達不能な節点はすべて除く。

(3) 図8(b)に示す方法で各節点に対する状態変数を割当てる。

(4) 対応する真理値表を作り、論理関数の形を求める。ここで、出力が一意に決まる場合については終了信号cを1にする。

図12(a)は、図2の遷移図にこの方法を適用する場合の、モジュールの入出力を示している。図11で節点{2,3}を除いたものに対応する真理値表は図12(b)に示されている。図12(c)は、結果として得られた論理式である。図7の結果と比べても、複雑さの差はあまりない。このモジュールでは、計算終了信号が次状態とは別に出力されており(状態が決定しなくても出力が決定するとそのモジュールの計算は終了とするため)、全体の構成は図12(d)のようにな



(d)

図12　部分的な計算結果を用いる方法

る。モジュールの出力と計算終了信号Cは別々に計算されるので、Cの方が遅く出力されるような工夫が必要である。

3. 部分的計算結果を利用した非同期回路の高速化

前節の方法は、一次元一方向反復回路だけではなく、より一般の非同期回路の高速化にも利用できる。

図12の方法では、状態にのみ図8(b)の表現を用いていたが、出力もこの形式にすることができる。すなわち、未定:(1,1)，出力1:(1,0)，出力0:(0,1)である。このようにすると次のような利点がある。

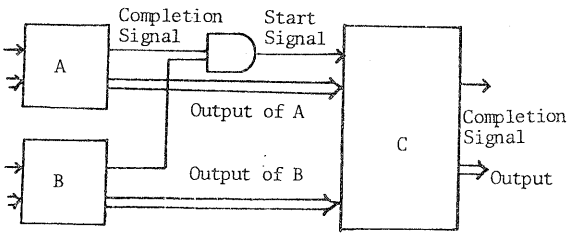


図13 計算終了信号を用いる方法

(1) 別の計算終了信号は必要とせず、図12の場合のように、Cの方が遅く出るといった条件を考えなくても良い。

(2) 図12の構成では、すべての終了信号の論理積をとり、それが1のときに初めてすべての出力が利用される。これは、図13のように、回路モジュールA、BがCへの入力となっているときに、A, Bの計算が終了してからCが働らくという形式になっている。実際には、たとえばBが先に計算終了し、その出力がAの出力に関係なくCの出力を決めることもありうるので計算の終了は別々に扱った方がよい。すなわち、あるモジュールの出力が未定(1,1)であっても、他のモジュールの出力により、この一次元反復回路の出力を利用する回路の計算を部分的に実行できる。

このような考え方を一般化すると、図14のような回路構成で非同期回路の高速計算が実現できる。ここで、各モジュールは、基本素子によって構成されており、各モジュールの入出力は図8(b)の形式の多線論理で表わされているものとする。この多線論理を部分集合の表現可能な多線論理という。この方式では、可能な出力の集合を常に示している。たとえば、AもBも出力値は確定していなくても、Aの出力は0か2のどちらかで、Bの出力は1か2のどちらかであることが判ると Cの出力が決まってしまうという場合も考えられ、図13の構成より効率はかなり良くなる可能性がある。図14のSはリセット信号で、各モジュールの出力はすべて11……となる。(出力にOR素子を入れて実現)

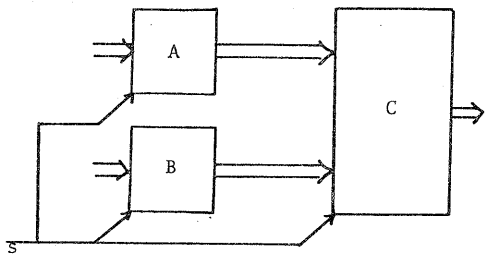


図14 部分集合の表現可能な多線論理による方法

あるモジュールの出力が4値の場合、2値を2つ組合せるか、直接4値にするか等といった問題がある。この場合、両方とも4本の信号線を必要とするが、2値のものを2つ組合せると表現できる場合の数は9通り(3×3)にすぎないが、4値を直接表わすと15通り(2^4-1)となる。すなわち、図15に示すように、2値の組合せでは可能なすべての出力の部分集合を表わす能力はない点に注意が必要である。

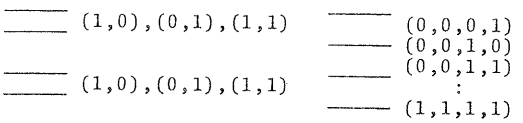


図15 2組の2線論理と4線論理

[定義 3] 論理関数$f(X)$が正関数であるといわれるのは、$f(X)$の表現法の中に、否定の論理変数($\bar{x}_i$)を使わない論理表現のある場合である。論理関数$f(X)$が負関数であるといわれるのは、$\bar{f}(X)$が正関数の場合である。

簡単な例では、AND, ORが正関数、NAND, NORが負関数である。$x_1x_2+x_3$も正関数の例である。

[定義 4] 入力ベクトル$X_1=(x_{11},\cdots,x_{1n})$と$X_2=(x_{21},\cdots,x_{2n})$において、$x_{1i}\geqq x_{2i}$ $(i=1,2,\cdots,n)$が成立すれば$X_1\geqq X_2$であるという。ここで、$x_{1i}\geqq x_{2i}$は、$1\geqq 1$, $1\geqq 0$, $0\geqq 0$ の関係である。

[性質 1] 論理関数$f(X)$が正関数であるための必要十分条件は、$X_i>X_j$であるようなすべての入力の組合せに対して、$f(X_i)\geqq f(X_j)$が成立することである。

[定義 5] 論理関数$f(X)$の双対関数$f^d(X)$は、つぎの形で定義される。

$$f^d(X)=\bar{f}(\bar{X})$$

ここで、$X=(x_1,\cdots,x_n)$に対し$\bar{X}=(\bar{x}_1,\cdots,\bar{x}_n)$である。

図8(b)に示される、部分集合の表現可能な多線論理を用いる場合、図14の各モジュールには次の性質が仮定される。

[定義 6] 非同期論理回路を図14の形式で実現した場合、モジュールが次の性質を持つとき、そのモジュールは部分的計算結果を保存するという。

(1) モジュールの入出力はともに部分集合の表現可能な多値論理である。

(2) モジュールの入出力関係を $Y=f(X)$ で表わす。このとき $X_i > X_j$ なら $f(X_i) \geqq f(X_j)$。

(2)の条件は、入力の不確実性が減少したのに、出力の不確実性が増加することがないという仮定である。

[定理 1] 部分的計算結果を保存するモジュールは、正関数による実現が可能である。

(証明) ある出力 z_k に注目する。$z_k = f_k(X)$。定義6により、任意の $X_i > X_j$ に対し

$$f_k(X_i) \geqq f_k(X_j).$$

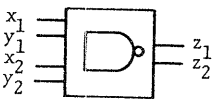
可能な入力は、制約を受けているため、f_k は不完全指定であるが、未指定の部分を適当な値になるように指定して性質1を満足させることができるため、f_k は正関数にできる。すべての出力に対して同じことが言える。 (証明終)

多値論理における値の置換は、出力線の入れ換えによって実現できるので、このようなモジュールを組合せて任意の関数が実現できる。各モジュールにおいて、入力は1が0に変化する方向にしか変化しないので、ハザードの起らない構成が可能である。

とくに、2値の場合は、出力の論理関数として、論理関数 f とその双対関数 f^d を用いることができ、従来より知られている2重系と類似してくる。まず簡単な例として、2入力NAND素子を部分的計算結果を保存するモジュールで構成することを考えてみる。入力は、(x_1, y_1) および (x_2, y_2) であり、出力は (z_1, z_2) である。

x_1	y_1	x_2	y_2	z_1	z_2
0	1	*	*	1	0
*	*	0	1	1	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

(a)



(b)

図16 2入力NANDモジュール

あるとし、(1,0), (0,1), (1,1) でそれぞれ、1, 0, 「1か0か不明」を表現するものを表現するものとする。入力のうち1つでも0があれば出力は1となり、2つとも1なら出力は0となり、一方が不明で他方が0でなければ不明になることから、図16に示す真理値表が得られる。この真理値表より、

$$z_1 = y_1 + y_2$$
$$z_2 = x_1 \cdot x_2$$

という表現が得られ、定理1に示したように共に正関数である。より一般的には次の定理を得ることができる。

[定理 2] 入出力がすべて2線論理(2値の表現)であり、部分的計算結果を保存するモジュールで正関数 f を実現するためには、入力 (x_i, y_i) $(i=1, \cdots, n)$ および出力 (z_1, z_2) の間に次の関係を与えればよい。

$$z_1 = f(x_1, x_2, \cdots, x_n)$$
$$z_2 = f^d(y_1, y_2, \cdots, y_n)$$

負関数 $\bar{f}$ を実現する場合は z_1 と z_2 を入れ換えるとよい。

(証明) (1) 入力が未定値を含まない場合:
この場合はすべての i $(1 \leqq i \leqq n)$ について $x_i = \bar{y}_i$ が成立する。

$$z_2 = f^d(y_1, y_2, \cdots, y_n) = \bar{f}(\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_n)$$
$$= \bar{f}(x_1, x_2, \cdots, x_n)$$

したがって、$z_1 = f$, $z_2 = \bar{f}$ となり出力は f の値によって (1,0) か (0,1) のどちらかとなる。

(2) 入力の一部に未定値のある場合: 一般性を失わず、$(x_1, y_1), \cdots, (x_j, y_j)$ が未定値であると仮定できる。このとき、

$$x_i = y_i = 1 \quad (1 \leqq i \leqq j),$$
$$x_i = \bar{y}_i \quad (j+1 \leqq i \leqq n).$$

したがって、

$$z_1 = f(1, 1, \cdots, 1, x_{j+1}, \cdots, x_n)$$
$$z_2 = f^d(1, 1, \cdots, 1, y_{j+1}, \cdots, y_n)$$
$$= \bar{f}(0, 0, \cdots, 0, x_{j+1}, \cdots, x_n)$$

f は正関数であるから、性質1により、常に

$$z_1 \geqq \bar{z}_2$$

この条件に対して次の3つの場合がある。

(2-1) $\mathfrak{z}_1=0$, $\mathfrak{z}_2=1$ の場合:

$$\mathfrak{f}(1,\cdots,1,x_{j+1},\cdots,x_n)=\mathfrak{f}(0,\cdots,0,x_{j+1},\cdots,x_n)=0$$

したがって、未定値がどの値に決っても $\mathfrak{f}$ の値は0となる(性質1による)。出力($\mathfrak{z}_1$, $\mathfrak{z}_2$)は(0,1)で0を表わしているので正しい。

(2-2) $\mathfrak{z}_1=1$, $\mathfrak{z}_2=0$ の場合:

上記と同様に、未定値がどの値に決っても $\mathfrak{f}$ の値は1となる。出力も(1,0)で1を表わしている。

(2-3) $\mathfrak{z}_1=1$, $\mathfrak{z}_2=1$ の場合:

$$\mathfrak{f}(1,\cdots,1,x_{j+1},\cdots,x_n)=1,$$
$$\mathfrak{f}(0,\cdots,0,x_{j+1},\cdots,x_n)=0,$$

となっているので、未定の部分の値によっては1にも0にもなりうる。出力も(1,1)で未定を表わしており、正しく対応している。

以上、すべての可能な場合についてこの方法で $\mathfrak{f}$ が実現できることが示された。$\mathfrak{F}$ の実現についても明らかである。 (証明終)

論理回路の信頼性の向上のために、従来から知られている $\mathfrak{f}$ と $\mathfrak{f}^d$ を用いた2重系と異るのは、計算の高速化に利用している点である。

[定理 3] 部分集合の表現可能な多線論理(線の数を m とする)と一般の2値論理の信号線の利用度の比率は次のようになる。

$$k=\frac{m}{\log_2 m}$$

この定理は、この構成による冗長度を示しており、$m=2$ で $k=2$, $m=4$ でも $k=2$, $m=8$ で $k=2.7$ 位となる。

4. 部分的計算結果を利用した非同期回路の故障検出性

前節で示したように、入出力がすべて部分集合の表現が可能な多線論理であるようなモジュールの内部は正関数で実現できる(定理1)。入力の一部にそのような条件を満足しないものがあれば、変換回路を付けることによりこの条件を満足させうる。

[定理 4] (フェイルセイフ実現) 入出力がすべて部分集合の表現が可能な多線論理であり内部がすべて正関数であるようなモジュールを用いて論理回路を構成した場合、各モジュール内の素子の故障は1縮退故障しかないと仮定すれば、回路は故障(多重も含む)によって、正しい答を含むような答を出すようになる。

(証明) 素子の1縮退故障は出力の1を増加させるか影響しないかのどちらかで、出力は部分集合の表現可能な多線論理で実現されていることから明らか。 (証明終)

すなわち、計算が最も長くかかる入力の組合せだけの時間の経過後も、出力値が一意に確定しなければ、出力に影響する1縮退故障があったことが判る。このようなことが起ると、入力がすべて一意的であるのに出力が一意的に確定していないようなモジュールを見付けて検査することになる。多重故障の場合は、1つのモジュールを直してから同じ操作を繰り返すとよい。

このように素子の故障が1縮退故障であれば出力の不確定性と共存できるが、0縮退故障では、誤った答を出す可能性がある。

(例) 出力が $\mathfrak{z}_1$ と $\mathfrak{z}_2$ であるモジュールを考える。故障がなければ(1,1)⟶(1,0)と変化するものとする。0縮退故障は、正関数の場合出力値を1から0に変化させる可能性があるので、0縮退故障が $\mathfrak{z}_1$ を計算する回路で起ったとすると(0,1)⟶(0,0)と変化する。このため、(0,1)のときに確定値として、まちがった出力が回路より出る可能性がある。

どこかのモジュールで、このモジュールの出力に影響するような0縮退故障があった場合で、モジュールの出力が最終的にはすべて0となる場合にこのような誤りが起る。このような場合、出力が誤ったという情報をあとで伝えて、先の出力を無効にすることを考える。すなわち、ある変数の多線表現がすべて0であるような入力が加えられると、出力もすべて0とするような操作を考える。

さきに述べたNAND素子では出力は次のようになる。

$$\mathfrak{z}_1=y_1y_2+x_1y_2+y_1y_2$$
$$\mathfrak{z}_2=x_1x_2$$

通常の動作の場合は、$\mathfrak{z}_1=y_1+y_2$ と同じ動作をするが、$(x_1,y_1)=(0,0)$ または $(x_2,y_2)=(0,0)$ の場合は、$(\mathfrak{z}_1,\mathfrak{z}_2)=(0,0)$ となる。このあらたに得られた関数も正関数であるため先に述べた性質を有している。

一般的には、次の定理がある。

[定理 5] (自己診断実現) 入出力がすべて部分集合の表現が可能な多線論理であり、内部がすべて正関数であるようなモジュールを用いて、出力に影響を与える多重の1縮退故障や多重の0縮退故障(1縮退故障と0縮退故障は混ざらないものとする)を出力側で検出できるような非同期回路の構成法がある。

(証明) 各モジュールを、正関数のままで、多線論理のすべての信号が0となるような入力が存在するとき出力もすべて0となるように構成できれば、0縮退故障を出力にまで伝播できる。図17にそのような変更方法を示している。たとえば、$(x_i, y_i) = (0,0)$となれば、それを検出して出力のAND素子に0を入力して出力をすべて0としている。 (証明終)

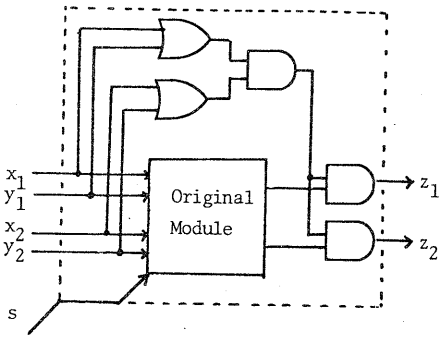


図17 0縮退故障の伝播

この方法によって、1縮退故障も0縮退故障も発見できることになる。しかしながら、この回路は高速計算のために用いるものであるから、一時的に出力値が確定すると、あとで(0,…,0)に変化する前にリセットしてしまうおそれがある。このため次のような方法が考えられる。

(1) 0縮退故障に関しては、ときどきテスト入力を入れて調べることにする。

(2) モジュール構成を変更し、出力がすべて0となった場合に限りリセット不能とする。

(2)の方法によれば、必ず故障の発見は可能となる。故障モジュールの発見も出力がすべて0のモジュールをたどってゆけば可能である。しかし、あらたに付加した部分の故障に対する対処が必要となる。図18に、出力がすべて0となった場合にリセット不能とするような回路の例を示す。(図17の回路とまとめることもできる)。

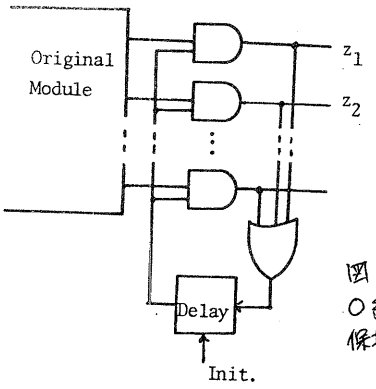


図18
0縮退故障の保持

5. むすび

本論文では、多線論理を利用して、回路の高速化をはかるとともに、故障の自己検出の能力を持たせるための方法について考察した。本論文の内容に関係して次のような問題が考えられる。

(1) 入力パターンによって計算時間が異るので、入力パターンの生起確率を与えた場合に、計算時間の平均値を最小にするような回路構成は可能か。

(2) 0縮退故障に対するより効果的な方法があるか。

(3) MOSのような素子を基本に用いると負関数が基本になる。高速性を問題にする場合は、2段実現が可能であるが、出力が否定の形で出るので、それに応じた構成法の変更が必要である。

謝辞 本論文について熱心に御検討いただいた矢島脩三教授、稲垣耕作氏、岩間一雄氏、安浦寛人氏に深謝する。

文献

(1) F.C. Hennie, "Iterative Arrays of Logical Circuits", The Technology Press of MIT, 1961.
(2) W.H. Waite, "The Production of Completion Signals by Asynchronous Iterative Networks", IRE, TEC, vol. EC-13, pp. 83-86, April 1964.
(3) S.H. Unger, "The Generation of Completion Signals in Iterative Combinational Circuits", IEEE, TC, vol. C-26, pp. 13-18, Jan. 1977.