4値VTゲート回路網の合成

4値VTゲート回路網の合成

比嘉広和* 瑞慶覧長定** 島袋勝彦**

Synthesis of Quaternary VT-gate Networks

Hirokazu HIGA*, Chotei ZUKERAN**, and Katsuhiko SHIMABUKURO**

Abstract

Recently one of the most important problems is the pin limitations in the integrated circuits. Multiple-valued logic has attracted for solution of the problem. because for the same amount of infomation transfer, the total number of pins required in the multiple-valued integrated circuit chip is much less than that of an binary integrated. In this paper, we discuss synthesizing quaternary VT-gate networks.(VT-gate has variable threshold) An implicant in the function plays important role when synthesizing VT-gate networks. Because we can reduce the number of VT-gates in the networks by utilizing implicant. We apply a candidate for implicant in eval function and show good result was obtained.

Key Words: VT-gate,Implicant.

1. はじめに

これまでに, 任意の4値2変数関数は樹枝状構造において12個のVTゲートで表せる [1] ことが知られている. しかしほとんどの4値2変数関数は10個以下で合成でき, またゲート数の減少は消費電力の低下や高集積化につながる. 本研究では可変しきい値特性を持つ4値VTゲートを用いて回路網合成プログラムを作成し, より少ないゲート数で回路網合成できるようにする.

任意の4値2変数関数をVTゲートを用いて最適回路網を合成する際に, インプリカント [2] が重要な役割を果たす. そこで, 4値論理関数をある分割パターンで2個の部分関数 (res-a, res-b) に分割し, それらの res-a, res-b を順次2分割していくと res-a, res-b 側に存在する同一定数のセルはインプリカントになりうる. このことより本アルゴリズムではインプリカントの候補といえる同一定数のセル数を評価方法としてとりいれ, 分割パターンを決定していく.

この結果, 過去に作成されたVTゲート樹枝状回路網合成プログラム [2] に比べ大幅なゲート数の減少が得られた.

2. 4値VTゲートの定義

4値論理系における真理値の集合 L を $L=\{0,1,2,3\}$ とすれば, 4値VTゲートの出力 $\mathrm{VT}(a,b;t;x)$ は, 式 (1) のように定義される. そのシンボルを図1に示す.

$$VT(a,b,t;x)=\begin{cases} a: & t\le x\le t+1 \\ b: & \text{otherwise} \end{cases} \tag{1}$$

受理: 1998年12月1日
*大学院理工学研究科 電気電子工学専攻
(Graduate Student, Electrical and Electronic Engineering)
**工学部電気電子工学科
(Dept. of Electrical and Electronic Engineering, Fac. of Eng.)

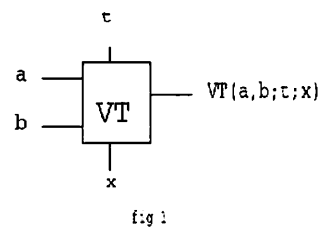


fig 1

表 1 真理値表

t \ x	0	1	2	3
0	a	a	b	b
1	b	a	a	b
2	b	b	a	a
3	b	b	b	a

ただし4値定数 $a,b;x;t\in L$ である. 図1において, x を制御変数, t をしきい値制御変数, a,b を residue function と呼ぶ.

4値VTゲート回路網の合成を行なう上で利用できる定理を以下に示す. 4値VTゲートには, 次のような関係が成立する [3].

$$VT(f(x'),f(x');t;x)=f(x') \tag{2}$$
$$VT(f_1(x'),f_2(x');x;x)=f_1(x') \tag{3}$$
$$VT(b,VT(b,a;x;0);3;x)=VT(a;b;1;x) \tag{4}$$

但し, $f(x'),f_1(x'),f_2(x')$ は任意の4値論理関数とし, $a,b,x,t,\in\{0,1,2,3\}$, とする.
式 (2) ~ (4) はVTゲート回路網の合成において, VTゲートを省くことができることを示している. また任意の4値n変数関数は樹枝状構造において, 高々4^n-1個の VT ゲートで実現できる [3].

3. 4値VTゲートを用いた関数の合成アルゴリズム

3.1 用語定義

本合成法を説明するに当たり, 使用する用語を次に定義する.

- インプリカントの定義

与えられた関数を1個のVTゲートで2分割し, 分割

された部分関数をさらに2分割する．この一連の操作の繰り返しによってできたM個のセルからなる部分関数が定数となる時，この部分関数をMセルインプリカントと定義し，$m = M - 1$ をMセルインプリカントの重みと呼ぶ．このMセルインプリカントが回路網合成の時に利用されると m 個のゲートが減少できる[2]．

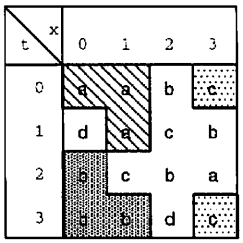


Fig. 2. インプリカントの例

- **代表関数**

ここでより多くの関数の合成を効率的に行なうために代表関数を導入する．

代表関数とは4値論理関数において，数値の代わりに $A, B, C, D \in \{0,1,2,3\}$ でかつ，$A \neq B \neq C \neq D$ となる変数を用いた関数である．この関数を用いることによって最大，$4 \times 3 \times 2 \times 1 = 24$ 通りの関数を1つの代表関数で表すことができる．例えば図3の2つの関数は代表関数で表すと全く同じになる．

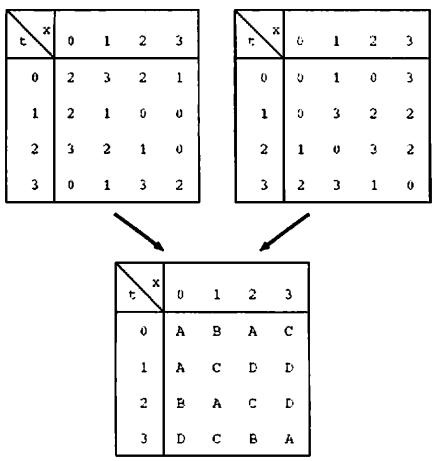


Fig. 3. 代表関数の例

- **VTゲートによる分割パターンの種類**

論理関数の合成では与えられた論理関数を分割VTゲートを用いて2つの部分関数に分解でき，それらの部分関数を順次VTゲートで分解し，分解された部分関数が定数と等しくなるまで繰り返す．そのとき部分関数を分割する分割パターンの種類が多いほど柔軟な回路網合成が可能となり有利である．本プログラムでは100種類の分割パターンを組み込んでおり，それらを表2から表5に示す．分割パターンを表すVTゲートの表現を図4のように定義する．

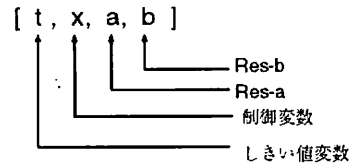


Fig. 4. VTゲートのリスト表現

表2 Nの値を変えることにより得られる分割パターン

N の値	分割パターン
N=x,y	[N,0,a,b] ¦ [[1,x,2,0,N,a,b]]
N=0,1	[[0,x,3,N,y,a,b]]
N=1,3	[[0,y,2,N x,a,b]]
N=0,1,2	[[0,y,3,N],x,a,b]]
N=0,1,3	[N,x,a,b] ¦ [N,y,a,b]
N=0,2,3	[[0,y,1,N],x,a,b]
N=1,2,3	[x,[0,y,0,N],a,b] [[0,y,0,N],x,a,b] [x,[0,y,N,0],a,b] [y[0,x,0,N],a,b] [y,[0,x,N,0],a,b]

表3 値を入れ換えることによって得られる分割パターン

[y,x,a,b]	[[y,0,3,2],x,a,b]
[[1,y,0,1],x,a,b]	[x,[y,0,2,0],a,b]
[x,[y,0,0,1],a,b]	[x,[y,0,3,0],a,b]
[[y,0,0,1],x,a,b]	[[y,0,1,2],x,a,b]
[[3,y,1,0],x,a,b]	[[y,0,0,3],x,a,b]
[[3,y,3,0],x,a,b]	[[1,y,0,3],x,a,b]
[[1,y,1,2],x,a,b]	[[y,0,1,3],x,a,b]

表4　表2と表3の複合パターン

Nの値	分割パターン
N=1，3	[[N,y,0,3],x,a,b] [[N,y,1,2],x,a,b] [[N,y,1,3],x,a,b] [[N,y,2,3],x,a,b] [x,[N,y,0,1],a,b] [x,[N,y,0,2],a,b] [x,[N,y,0,3],a,b]

表5 その他の分割パターン

[y,[3,x,3,1],a,b]	[[1,y,2,0],y,a,b]
[y,[x,0,3,1],a,b]	[[1,y,0,2],x,a,b]
[[y,0,0,2],x,a,b]	[[3,y,0,2],x,a,b]
[[0,x,1,3],y,a,b]	[[0,x,0,3],y,a,b]
[[[1,x,2,0],x,0,2],y,a,b]	

3.2　VTゲート回路網合成アルゴリズム

本プログラムではあらかじめ組み込んである分割パターンを用いて，与えられた論理関数を二つの部分関数に分解し，それらの部分関数を順次分解して，すべての部分関数が don't care と同一定数のみになるまで分割していくことにより回路網合成を行なう．何番目の分割パターンを選ぶかについては，部分関数を分割する際に，点数化してその点数が最大となる分割パターンで分割する．最大の点数となる分割パターンが複数存在した場合それらをすべて試行し，最小のゲート数で合成できたものを解として出力する．メインアルゴリズムのフローチャートを図5に示す．

本プログラムで一番重要となるのは分割ゲートを決定する点数化である．その点数化する手順を次に説明する．

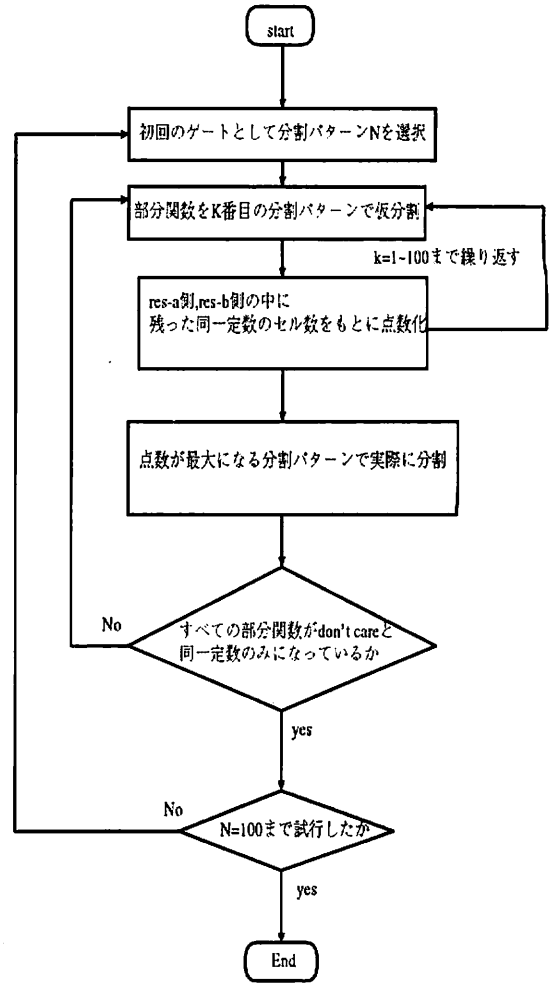


Fig. 5.　メインアルゴリズム

- **点数化**

4値論理関数をある分割パターンで res-a, res-b へと分割する．その時 res-a 側，res-b 側に存在する同一定数のセルは分割を繰り返していくとインプリカントになりうる．これよりインプリカントの候補といえる同一定数のセル数を評価方法としてとりいれ，分割パターンを決定していく．

関数を K 番目の分割パターンで仮分割し，res-a 側の部分関数に N(N は 0，1，2，3) の同一定数が何個存在するかを調べ，評価式 (同一定数の数-1) で計算する．

res-b 側も同様にして計算し，res-a,res-b 両方の点数の和をその K 番目の分割パターンの点数とする．但し，点数が同じになる分割パターンがいくつか存在した場合，より少ないゲート数で分割するために (分割のために使用するゲート数 −1) を点数から引くことによってゲート数が少ない分割パターンが選ばれるようにしている．すなわち res-a 側の点数を S_a，res-b 側の点数を S_b とすると S_a は

$$S_a = \sum_{a=0}^{3}(N_a - 1) \tag{5}$$

と表せる．(S_b も同様) ここで N_a とは 4 値定数 a となっているセルの数を表す．これを用いて，ある分割パターンの点数 S は

$$S = S_a + S_b - g \tag{6}$$

となる. 但し g = (分割のために使用するゲート数 − 1)

このように1から100番目までの分割パターンを同様にして点数化していき, 点数が最大となる分割パターンで実際に関数を分割する.
以下これを繰り返し合成回路網を得る. 以下に例を示す.

- **例1** 表6の4値2変数関数を分割パターン $[0, x, a, b]$ で分割した時の点数化

$$S_a = (1-1)+(3-1)+(2-1)+(2-1) = 4$$

$$S_b = (2-1)+(1-1)+(1-1)+(4-1) = 4$$

これよりこの分割パターンの点数は式 (6) より

$$S = 4+4-(1-1) = 8$$

となる.

VTゲート回路網合成の際に, 初段の分割ゲートの決定が重要な要素となる. 初段の分割ゲートが適切でないと最適な回路網合成が不可能となる. しかし適切な初段の分割ゲートが必ずしも最大の点数を獲得するとは限らなかったので本アルゴリズムでは点数化を用いて適切な初段の分割ゲートを決定しなかった. そのため初段の分割ゲートはすべてのパターンを試行し, 2段目以降を点数化によって決定してその中で全体のゲート数が最小となる解を最適解として出力する.

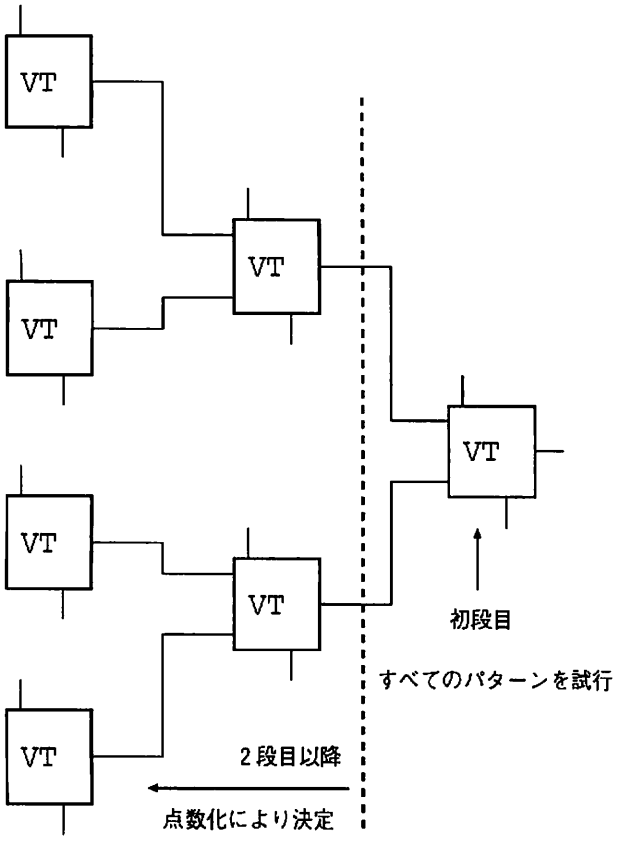
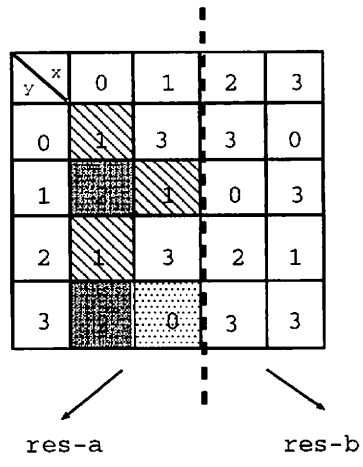


Fig. 6. VTゲート樹枝状回路網

表6 真理値表

y \ x	0	1	2	3
0	1	3	3	0
1	2	1	0	3
2	1	3	2	1
3	2	0	3	3



4. シミュレーション結果

乱数で発生させた一万個の関数を合成プログラムで処理した結果を図7に示す. またここで使用した計算機のCPUは pentium(133MHz) である.

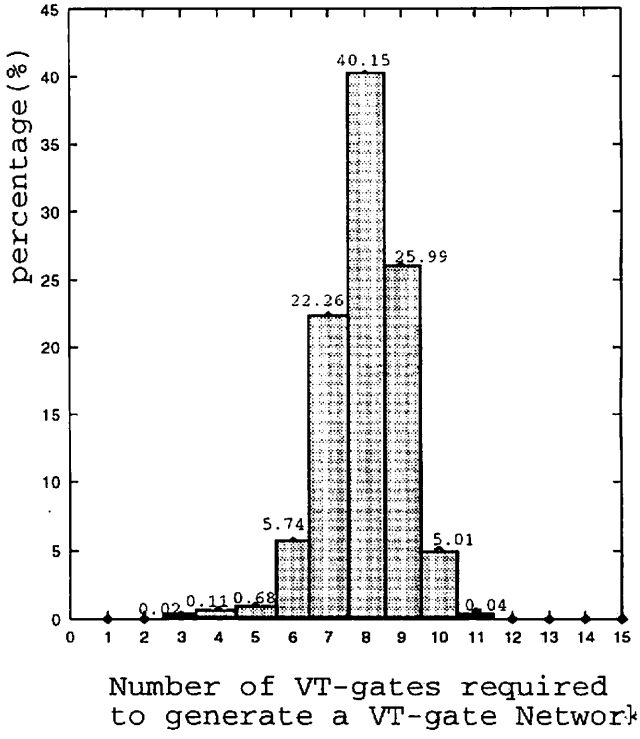


Fig. 7. シミュレーション結果

このシミュレーションの結果, すべての関数がVTゲート11個以下で合成できた. その中で11個のVTゲートを必要とする関数が4個 (一万個中) あり, 本アルゴリズムがVTゲート回路網の合成に対して有効であることが確認できた. その11個のVTゲートを必要とする関数を調べると, すべて10個のVTゲートで合成できることがわかった. そのうち合成の困難な関数とその回路網を表7と図8に示す. この表7の関数がVTゲート樹枝状回路網においてもっとも簡単化が難しい関数の一つであると考え

られ, これを調べることによってVTゲートの性質が明らかになると思われる.

また, 関数を分割する初段の分割ゲートに関しては本アルゴリズムでは最適なゲートを見つけることができなかった. そのため初段の分割ゲートは１００種類のゲートすべてを試行し, その中でゲート数が最少となる解を採用している. この結果, 計算時間が1個当り約１０秒かかった. 今後の課題として初段の分割ゲートを求めるために新たな評価方法を検討する必要がある.

表7　簡単化が困難と思われる関数

t \ x	0	1	2	3
0	A	B	C	D
1	B	C	D	D
2	A	B	A	C
3	D	A	D	B

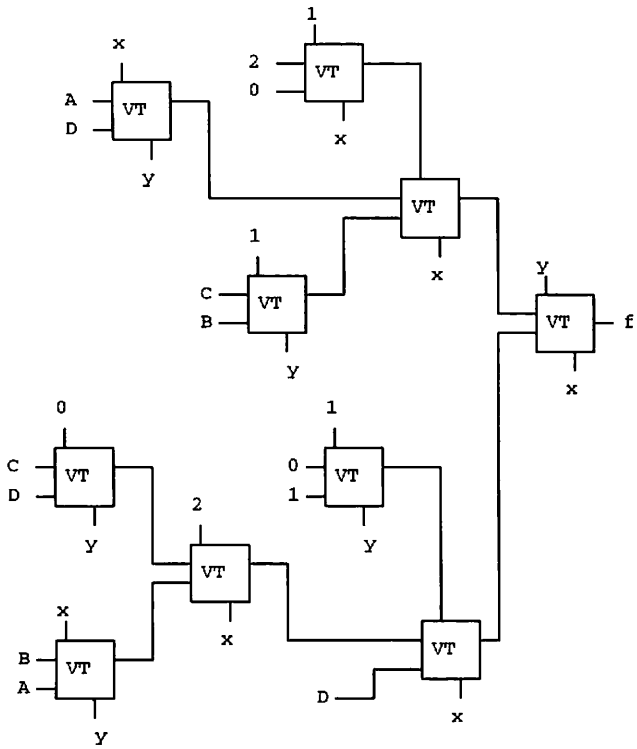


Fig. 8. 表7の関数の合成例

参考文献

[1] 瑞慶覧、陳、島袋、安富祖.「4値論理関数の合成に要するVTゲートの上限」. 多値技報 *vol1,MVL-96,No1*, pages 14–23, 1 1996.

[2] 陳丁.「4値VTゲート回路網の合成に関する研究」. 平成8年度琉球大学研究生論文.

[3] 山城哲也.「4値VTゲート回路網に関する基礎的研究」. 平成5年度琉球大学修士論文.