Vol. 21 No. 5 情 報 処 理 May 1980

入選論文

フォールト・トレラント・ゲートの提案†

武藤佳恭†† 池田政弘†††

従来の論理回路（ゲート）を複数個用いることによって論理回路自身に冗長性を与え，ゲートがもつ機能の信頼性を向上させるための設計手法を提案する．この新しい冗長ゲートをフォールト・トレラント・ゲートと名付け，代表的な基本論理回路，AND, OR, NOT, NAND, NOR, Exclusive OR についてその構成法を具体的に示す．それらの信頼性向上を従来のゲートとの比較において論じる．また，従来広く提案されている多数決回路に比べ，フォールト・トレラント・ゲートはより広範囲の故障に対して回復機能を有することを示す．全加算器，算術論理演算器に提案するフォールト・トレラント・ゲートを適用した場合について，信頼性の改善度を明らかにする．またフォールト・トレラント・ゲートによる大規模論理回路の信頼性について論じ，超高信頼性コンピュータが実現可能なことを示す．この高信頼化設計手法は論理回路の信頼性向上はもとより，大規模集積回路素子開発における歩留りの改善にも大きな役割りを果す可能性があることを示す．

1. はじめに

フォールト・トレラント回路【1】を構成するために，ハードウェアの冗長が用いられている．そのハードウェアの冗長方式を大別すると次の3種になる【2】．

i．静的冗長 (static redundancy)

ii．動的冗長 (dynamic redundancy)

iii．ハイブリッド冗長 (hybrid redundancy)

静的冗長方式とは，同一の機能ユニットを多重化し，それらの多数決論理をとる方式である．多数決論理を用いると必ず VOTER* が必要となる．

VOTER を用いた場合，たとえ機能ユニットを多重化しても，VOTER の持つ信頼性以上の信頼性を持つ論理回路は得られない．

動的冗長方式とは，補助の監視診断回路により，故障を検出するか，あるいは自己修復機能回路によって故障部分を修復するか，いずれかの方法で誤りを訂正する方式である【3】．この場合も，監視診断回路もしくは自己修復回路部分がフォールト・トレラント回路の信頼性の隘路となる．

ハイブリッド冗長方式とは，静的冗長方式と動的冗長方式とを混合した方式である．前述の2方式と同様に，VOTER あるいは故障検出訂正回路部分が信頼性の隘路となる．

VOTER あるいは，それに相当する機能の COLLECTOR**【4】といった回路は複数のゲートより構成されているので，それらの信頼性は当然ゲート1個あたりの信頼性よりも必ず低くなる．

上記3方式ともゲート1個あたりの信頼性よりも高い信頼性の論理回路は得られない．それ以上の信頼性を達成するためには，論理回路の基本構成要素であるゲート機能自身の信頼性を上げる必要がある．

従来，ゲート機能自身の信頼性を上げるためには，半導体技術にたよる以外にはなかった．

そこで著者らは，従来のゲートを用いても，なおかつ信頼性の高いゲート機能モジュール，すなわちフォールト・トレラント・ゲート (fault-tolerant gate) を作り出すことを考えた．

高信頼性を得るためには，フォールト・トレラント・ゲートに対して，他のフォールト・トレラント・ゲートで発生した故障を含む入力に対しても正しい論理演算が行える機能をもたせる必要がある．

本論文では，論理回路の基本構成要素であるゲート機能自身の信頼性を向上させるために，従来のゲートを用いることによって構成し得るフォールト・トレラント・ゲートの考え方を提案する．次に代表的なフォールト・トレラント・ゲート，AND, OR, NOT, NAND, NOR, Exclusive OR (以下 XOR と記す) の構成法を具体的に示し，その信頼性の向上に関して検討する．さらに，フォールト・トレラント・ゲートを構成要素とする機能論理回路への応用を考え，その信頼性について検討する．その結果，フォールト・トレラント・ゲートを構成しているゲート1個あたりの信頼性よりも高い信頼性をもつ論理回路が容易に実現

† A Proposal of Fault-Tolerant Gates by YOSHIYASU TAKEFUJI and MASAHIRO IKEDA (Department of Electrical Engineering, Keio University).

†† 慶應義塾大学工学研究科

††† 慶應義塾大学工学部電気工学科

* VOTER とは，奇数本の入力線をもつ2進論理回路で，その出力はどちらか多数の入力値できまる．

** COLLECTOR とは，誤り信号をマスクする検出器と入力信号を相互に比較する回路により出力を決める論理回路で，VOTER と同じ役割を果す．

できることを示す．さらに，このような高信頼性のフォールト・トレラント・ゲートを用いることにより，従来の技術では達成することができなかった高信頼性フォールト・トレラント・コンピュータの設計が可能なことを実証する．

2. フォールト・トレラント・ゲート

フォールト・トレラント・ゲートとは，ゲートの入出力信号線とゲート機能に最適な冗長を持たせた新しい論理素子である．このフォールト・トレラント・ゲートは誤り訂正符号の概念をとり入れたものである．

2.1 フォールト・トレラント・ゲートの原理

論理回路の入力あるいは出力データ線m本に対して冗長線n本を加えたものを入力ベクトルあるいは出力ベクトルと定義する．

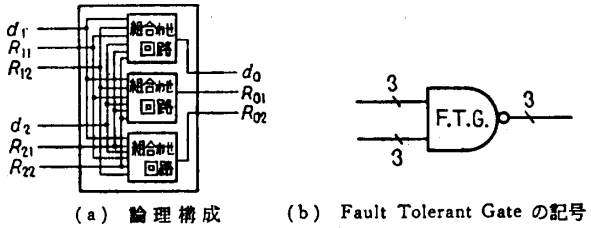
例えばデータを1ビット，冗長を4ビットとすると，ベクトルは5ビット長になる．この時，ベクトルに2ビットまでの誤りを許すことができる．なぜならば，符号数は2個であり，それら2符号間のハミング距離を5つまで離すことができるからである．

一般に，tビットまでの誤りを許すためにはベクトルはそれぞれハミング距離$2t+1$以上離れている必要がある【5】．

フォールト・トレラント・ゲートは，入出力線にこのような冗長を持ったベクトルを用いることによって，入力ベクトルに誤りが含まれていても，その論理演算を行うと正しい出力ベクトルを生成することができる組合わせ論理回路であり，その例を**図1**に示す．

図2に示す2入力NANDのフォールト・トレラント・ゲートに誤った入力を含む入力ベクトルを与えた時の論理演算の例を**表1**に示す．ただし，この例では各入出力ベクトルを3ビット（データ1ビットに対して冗長2ビット）とし，そのデータビットと冗長ビットの関係を明らかにするために生成多項式を用いる．いろいろな生成多項式が考えられるが，この場合生成多項式を $G(X)=X^2+X+1$ とする．この生成多項式の選択についてその理由を3章で述べる．この生成多項式を用いると，入出力ベクトルとしては（0 0 0）あるいは（1 1 1）が本来正しいベクトルであるが，1ビットまでの誤りを含む入力ベクトルに対しても正しい出力ベクトル(出力ベクトルは次段のフォールト・トレラント・ゲートの入力ベクトルとなる）が生成される．1ビットの誤りを含む入出力ベクトルとは以下のようなパターンである．

○で囲んだビットは誤りビットを示す．



(a) 論理構成　　(b) Fault Tolerant Gate の記号

図1 2入力NANDフォールト・トレラント・ゲートの構成

Fig. 1 An Example of a NAND-FTG (2 inputs) and its logic symbol.

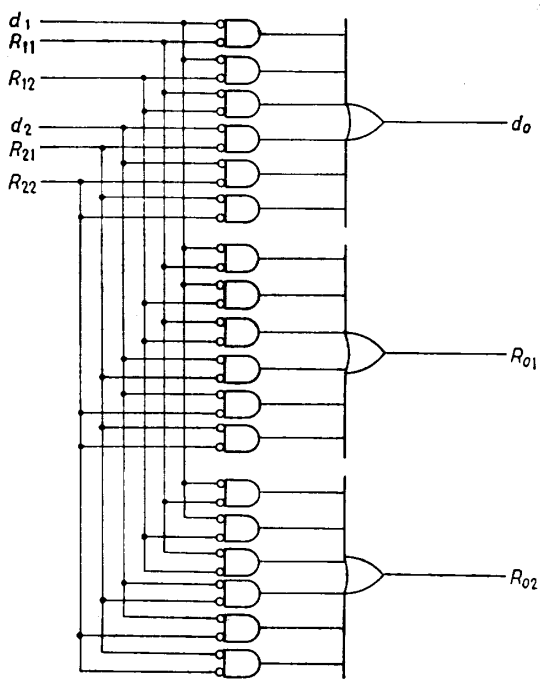


図2 2入力NAND-FTGの論理図

Fig 2 Logic diagram of a NAND-FTG (2 inputs).

表1 2入力NANDの例

Table. 1 An Example of error corrections in a Fault-Tolerant NAND (2 inputs) Gate.

入力ベクトル			入力ベクトル			出力ベクトル		
d_1	R_{11}	R_{12}	d_2	R_{21}	R_{22}	d_0	R_{01}	R_{02}
0	0	0	0	①	0	1	1	1
1	1	⓪	①	0	0	1	1	1
①	0	0	0	①	0	1	1	1
1	1	⓪	⓪	1	1	0	0	0
1	⓪	1	1	⓪	1	0	0	0
	⋮			⋮			⋮	
0	0	①	1	1	⓪	1	1	1

注 1. d_1, d_2 は入力データビット．$R_{11}, R_{12}, R_{21}, R_{22}$ は入力冗長ビット．
2. d_0 は出力データビット．R_{01}, R_{02} は出力冗長ビット．
3. ○で囲まれたビットは誤りビットを示す．

(①0 0)
(0①0)
(0 0①)
} は（0 0 0）ベクトルとみなすことができ，

(⓪1 1)
(1⓪1)
(1 1⓪)
} は（1 1 1）ベクトルとみなせる．

1ビットの誤りを含むこれらの6つの入力ベクトルのどのような組合わせをフォールト・トレラント・ゲートに与えても正しい出力ベクトルを生成できる．

最少のハードウェア量で最大の冗長度を持つフォールト・トレラント・ゲートの作成手順をまとめると以下のようになる．

（i） 各入出力線の冗長度を決める．
（データmビットに対して冗長nビット）

（ii） （i）の冗長度および，フォールト・トレラント・ゲートを構成するゲート数を考えて生成多項式を決定する．

次章に，具体的なフォールト・トレラント・ゲートの設計例を示す．

3. フォールト・トレラント・ゲートの設計例

データ線数と冗長線数の組合わせに関して，フォールト・トレラント・ゲート間のインタフェースを考えるとデータ1ビットごとに冗長ビットをつけた方がフォールト・トレラント・ゲートのハードウェアを簡単にすることができる．そこでデータを1ビット，冗長を2ビットとし，生成多項式を $G(X)=X^2+X+1$ とすると良いことが判明した．以下にこの理由を述べる．

（i） 1つの入力ベクトルに対して，1ビット誤りまで許すためには，データビットと冗長ビットの和は3ビット以上必要である．

（ii） 生成多項式を $G(X)=X^2+X+1$ とするとデータの論理演算および冗長ビットの計算が同じ回路で同時に生成できる．

（iii） データビットと冗長ビットの区別をつける必要がなくなる．

次に具体的に2入力 AND, OR, NOT, NAND, NOR, XOR ゲートに関して，これらの条件（データ1ビット，冗長2ビット，$G(X)=X^2+X+1$）でフォールト・トレラント・ゲートを構成する．説明に際し，論理設計が容易であった NAND フォールト・トレラント・ゲートの設計例を始めに示す．

（1） 2入力 NAND フォールト・トレラント・ゲート

$G(X)=X^2+X+1$ であるから，出力データ・ビットおよび出力冗長ビットの論理演算は等しくなる．

$$d_0=R_{01}=R_{02} \qquad (1)$$

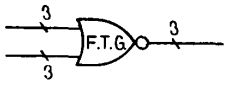
表1の論理演算をブール式で表わすと，次のような論理式となる．

$$d_0=\bar{d}_1\cdot\bar{R}_{11}+\bar{d}_1\cdot\bar{R}_{12}+\bar{R}_{11}\cdot\bar{R}_{12}+\bar{d}_2\cdot\bar{R}_{21}+\bar{d}_2\cdot\bar{R}_{22}+\bar{R}_{21}\cdot\bar{R}_{22} \qquad (2)$$

(1), (2)式より，2入力 NAND フォールト・トレラント・ゲートを構成すれば図2に示すようになる．この場合全ゲート数*は21個となる．

他も同様の手順で設計できる．

（2） 2入力 NOR フォールト・トレラント・ゲート



出力データビット d_0 は次のような論理式で表わせる．

$$d_0=R_{01}=R_{02} \qquad (3)$$

$$d_0=(\bar{d}_1\cdot\bar{R}_{11}+\bar{d}_1\cdot\bar{R}_{12}+\bar{R}_{11}\cdot\bar{R}_{12})\cdot(\bar{d}_2\cdot\bar{R}_{21}+\bar{d}_2\cdot\bar{R}_{22}+\bar{R}_{21}\cdot\bar{R}_{22}) \qquad (4)$$

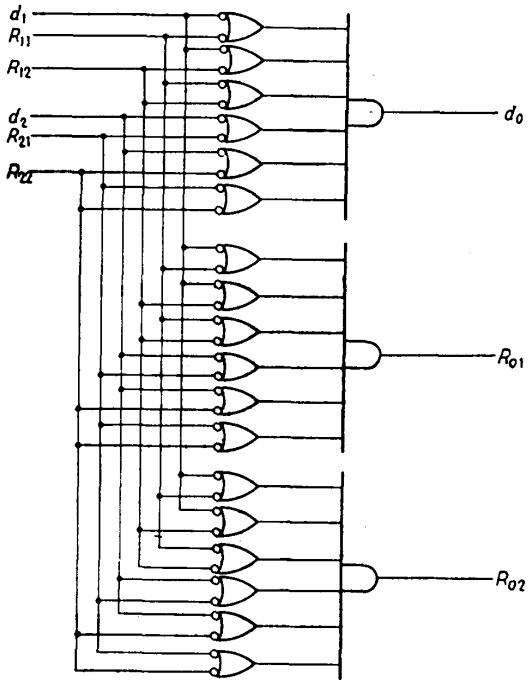


図 3 2入力 NOR-FTG の論理図
Fig. 3 Logic diagram of a NOR-FTG (2 inputs).

* 本論文では個々のゲート (AND, OR, NAND, NOR) の最大入力数を9としている．その理由は，論理回路で用いられるゲートの平均入力数を3とした場合，3入力までのフォールト・トレラント・ゲートは最大9入力までのゲートを用いて実現できるからである．

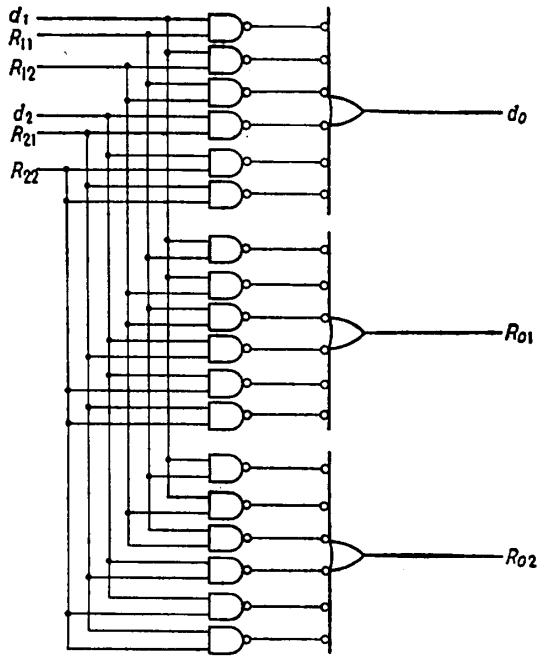


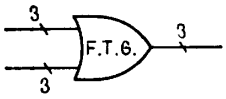
図 4 2入力 OR-FTG の論理図
Fig. 4 Logic diagram of an OR-FTG (2 inputs).

ゲート数を減らすために，さらに（4）式を変形する．

$$d_0=(\bar{d}_1+\bar{R}_{11})\cdot(\bar{d}_1+\bar{R}_{12})\cdot(\bar{R}_{11}+\bar{R}_{12})\cdot(\bar{d}_2+\bar{R}_{21})\cdot(\bar{d}_2+\bar{R}_{22})\cdot(\bar{R}_{21}+\bar{R}_{22}) \quad (5)$$

（3），（5）式より，2入力 NOR フォールト・トレラント・ゲートを構成すれば**図3**に示すようになる．

（3） 2入力 OR フォールト・トレラント・ゲート



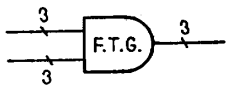
出力データビット d_0 は次のような論理式で表わせる．

$$d_0=R_{01}=R_{02} \quad (6)$$

$$d_0=d_1\cdot R_{11}+d_1\cdot R_{12}+R_{11}\cdot R_{12}+d_2\cdot R_{21}+d_2\cdot R_{22}+R_{21}\cdot R_{22} \quad (7)$$

（6），（7）式より，2入力 OR フォールト・トレラント・ゲートを構成すれば**図4**に示すようになる．

（4） 2入力 AND フォールト・トレラント・ゲート



出力データビット d_0 は次のような論理式で表わせる．

$$d_0=R_{01}=R_{02} \quad (8)$$

$$d_0=(d_1\cdot R_{11}+d_1\cdot R_{12}+R_{11}\cdot R_{12})\cdot(d_2\cdot R_{21}+d_2\cdot R_{22}+R_{21}\cdot R_{22}) \quad (9)$$

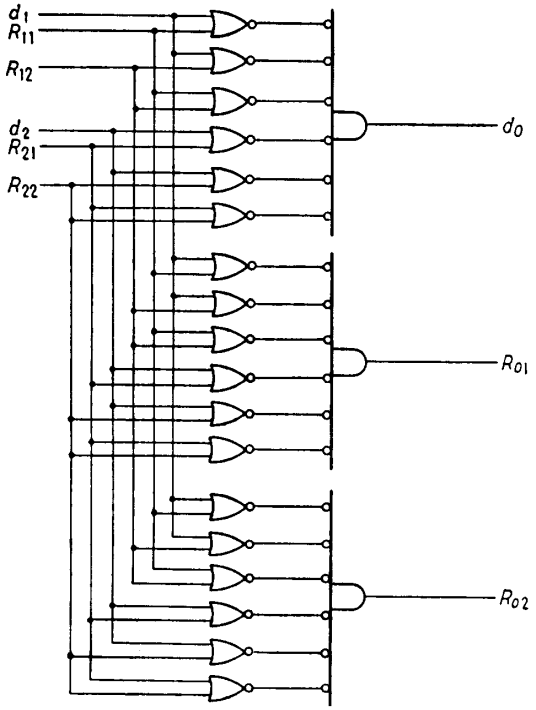


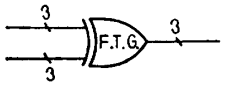
図 5 2入力 AND-FTG の論理図
Fig. 5 Logic diagram of an AND-FTG (2 inputs).

ゲート数を減らすため，さらに（9）式を変形する．

$$d_0=(d_1+R_{11})\cdot(d_1+R_{12})\cdot(R_{11}+R_{12})\cdot(d_2+R_{21})\cdot(d_2+R_{22})\cdot(R_{21}+R_{22}) \quad (10)$$

（8），(10) 式より，2入力 AND フォールト・トレラント・ゲートを構成すれば，**図5**に示すようになる．

（5） XOR フォールト・トレラント・ゲート



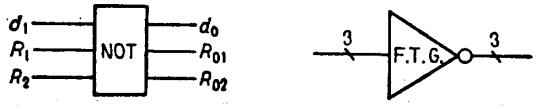
出力データビット d_0 に対してその論理式は次のようになる．

$$d_0=R_{01}=R_{02} \quad (11)$$

$$d_0=(d_1\cdot R_{11}+d_1\cdot R_{12}+R_{11}\cdot R_{12})\oplus(d_2\cdot R_{21}+d_2\cdot R_{22}+R_{21}\cdot R_{22}) \quad (12)$$

(11)，(12) 式より，XOR フォールト・トレラント・ゲートを構成すれば**図6**に示すようになる．

（6） NOT フォールト・トレラント・ゲート



$$d_0=R_{01}=R_{02} \quad (13)$$

$$d_0=\bar{d}_1\cdot\bar{R}_1+\bar{R}_1\cdot\bar{R}_2+\bar{d}_1\cdot\bar{R}_2 \quad (14)$$

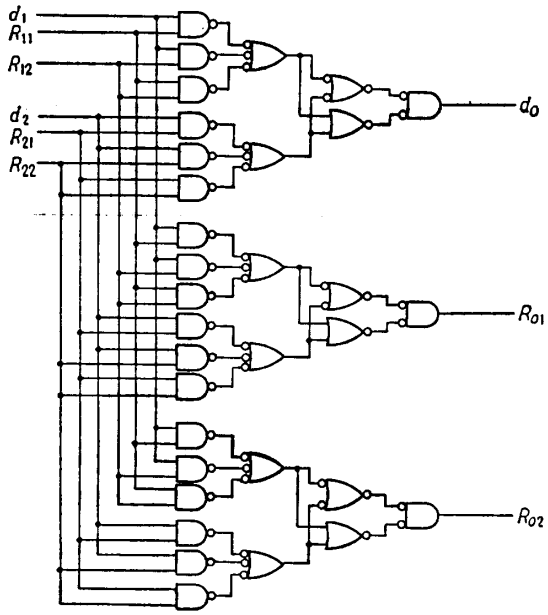


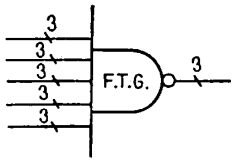
図 6 XOR-FTG の論理図

Fig. 6 Logic diagram of an Exclusive OR-FTG (2 inputs).

(13), (14) 式により, NOT フォールト・トレラント・ゲートの論理図を図7に示す.

(7) 5入力 NAND フォールト・トレラント・ゲート

5入力 NAND フォールト・トレラント・ゲートの d_0 の論理式は(15)および(16)式のようになる.



$$d_0=R_{01}=R_{02} \tag{15}$$

$$\begin{aligned} d_0=&\bar{d}_1\cdot\bar{R}_{11}+\bar{d}_1\cdot\bar{R}_{12}+\bar{R}_{11}\cdot\bar{R}_{12}\\ &+\bar{d}_2\cdot\bar{R}_{21}+\bar{d}_2\cdot\bar{R}_{22}+\bar{R}_{21}\cdot\bar{R}_{22}\\ &+\bar{d}_3\cdot\bar{R}_{31}+\bar{d}_3\cdot\bar{R}_{32}+\bar{R}_{31}\cdot\bar{R}_{32}\\ &+\bar{d}_4\cdot\bar{R}_{41}+\bar{d}_4\cdot\bar{R}_{42}+\bar{R}_{41}\cdot\bar{R}_{42}\\ &+\bar{d}_5\cdot\bar{R}_{51}+\bar{d}_5\cdot\bar{R}_{52}+\bar{R}_{51}\cdot\bar{R}_{52} \end{aligned} \tag{16}$$

(8) その他のフォールト・トレラント・ゲート

次のようなフォールト・トレラント・ゲートの構成ゲート数は2入力 NAND の場合の総ゲート数と同一

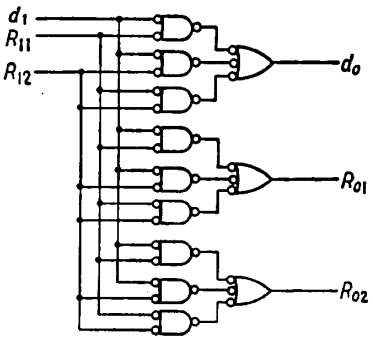
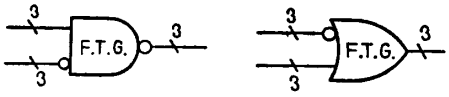


図 7 NOT-FTG の論理図

Fig. 7 Logic diagram of a NOT-FTG.

である. すなわち d_0 の論理式は (17), (18) 式のようになる.



$$d_0=R_{01}=R_{02} \tag{17}$$

$$\begin{aligned} d_0=&\bar{d}_1\cdot\bar{R}_{11}+\bar{d}_1\cdot\bar{R}_{12}+\bar{R}_{11}\cdot\bar{R}_{12}\\ &+d_2\cdot R_{21}+d_2\cdot R_{22}+R_{21}\cdot R_{22} \end{aligned} \tag{18}$$

(1)から(6)までのフォールト・トレラント・ゲートについて, ゲート数に関する比較を行った. その結果を表2に示す.

ゲート数は信頼性にも密接な関係があり, 一般的にゲート数が増すと信頼性は低下する. すなわち, フォールト・トレラント・ゲートを構成するゲート数が増加すると, 次段に接続されるフォールト・トレラント・ゲートへの伝播故障確率が増大する.

次章にこれらのフォールト・トレラント・ゲートの信頼性に関して定量的に論じる.

4. フォールト・トレラント・ゲートの信頼性

"従来のゲートを用いて, そのゲート以上の信頼性を持つフォールト・トレラント・ゲートを果して構成できるか?" という最も重要な課題に関しては, 前章の2入力 NAND, 2入力 XOR, および NOT に関する信頼性をそれぞれ定量的に求めてから検討する.

(1) 2入力 NAND フォールト・トレラント・ゲート

表 2 フォールト・トレラント・ゲートと構成ゲート数

Table 2 The number of gates required for realizing an FTG.

論理回路	2入力 AND-FTG	2入力 OR-FTG	NOT-FTG	2入力 NAND-FTG	2入力 NOR-FTG	2入力 XOR-FTG
ゲート数(個)	3×7=21	3×7=21	3×4=12	3×7=21	3×7=21	3×11=33

誤りを含むベクトルが入力されてもその出力が正しい論理演算結果になる．すなわち復帰可能な出力になる確率を R，フォールト・トレラント・ゲートを構成するゲート1個の故障確率を P，フォールト・トレラント・ゲートの1つの入力に誤りが混入してくる確率（伝播故障確率）を Pd とする．

2つの入力ベクトル各々に関して誤りを1ビット訂正できるから，入力側の復帰可能な入力ベクトルの確率 V は，2つの入力ベクトルが正しい確率 V_0，2つの入力ベクトルに1個だけ誤りが混入する確率 V_1，ならびに2つの入力ベクトルに各々1個ずつの誤りが混入する確率 V_2 をすべて加えたものになる．すなわち，

$$V=V_0+V_1+V_2$$
$$=(1-Pd)^6+6(1-Pd)^5\cdot Pd+9(1-Pd)^4\cdot Pd^2$$

一方，フォールト・トレラント・ゲートを構成している全ゲートが正しく動作する確率は $(1-P)^{21}$ である．また全ゲート中1個だけ故障する確率は，${}_{21}C_1\cdot(1-P)^{20}\cdot P$ であり，全ゲート中 n 個$(2\leq n\leq 7)$のゲートが故障しても，復帰可能な出力ベクトルを生成する確率は，

$$3\cdot {}_7C_n(1-P)^{21-n}\cdot P^n,$$

となる．

すなわち復帰可能な出力になる確率 R は，

$$R=\{(1-Pd)^6+6\cdot(1-Pd)^5Pd+9(1-Pd)^4\cdot Pd^2\}$$
$$\cdot\{(1-P)^{21}+{}_{21}C_1(1-P)^{20}\cdot P$$
$$+3\cdot\sum_{n=2}^{7}{}_7C_n\cdot(1-P)^{21-n}\cdot P^n\}$$
$$\fallingdotseq(1-147P^2+1568P^3-8967P^4+\cdots)$$
$$\cdot(1-6Pd^2+4Pd^3-9Pd^4-12Pd^5+4Pd^6)$$

である．

次段への伝播故障確率 Pd はフォールト・トレラント・ゲートを構成するゲートの故障確率によりほぼ決定される．なぜならば，前段からの伝播故障はその段のフォールト・トレラント・ゲートにより訂正され，その故障が次段へ伝播される確率は無視できる程小さくなるからである．その理由は，例えば2入力NANDフォールト・トレラント・ゲートから後段のフォールト・トレラント・ゲートへ伝播される故障確率 (Pd_i) は次式により表わされる．

$$Pd_i=1-(1-P)^7\cdot[(1-Pd_{i-1})^6+6\cdot(1-Pd_{i-1})^5$$
$$\cdot Pd_{i-1}+9\cdot(1-Pd_{i-1})^4\cdot Pd^2{}_{i-1}]$$

ここで Pd_{i-1} は2入力フォールト・トレラント・ゲートへの伝播故障確率である．

一般に Pd_{i-1} および P は小さい $(1\gg Pd_{i-1}>0, 1\gg P>0)$ ので Pd_{i-1} および P のみを考慮し，$Pd^2{}_{i-1}$, $Pd^3{}_{i-1}$, …, $Pd^6{}_{i-1}$ および $P^2, P^3, \cdots, P^7$ を無視した場合，$Pd_i\fallingdotseq 7\cdot P$ となる．

表3 2入力 NAND-FTG の信頼性

Table 3 Reliability comparison between a NAND-FTG (2 inputs) and an usual NAND gate.

ゲート1個の故障確率 P	通常のNANDゲート $R=(1-P)\cdot(1-Pd)^2$	2入力NAND-FTG $R=(1-147P^2)\cdot(1-6Pd^2)$
10^{-2}	0.99	0.956
10^{-3}	0.999	0.99956
10^{-4}	0.9999	0.99999559
10^{-5}	0.99999	0.9999999559
10^{-6}	0.999999	0.999999999559
10^{-7}	0.9999999	0.99999999999559
10^{-8}	0.99999999	0.9999999999999559
10^{-9}	0.999999999	0.999999999999999559

フォールト・トレラント・ゲートの出力ベクトルが復帰可能となる確率 R について具体的に計算してみよう．計算するにあたって，次段への伝播故障確率 Pd を

$$Pd=\frac{N}{3}\cdot P$$

と仮定する．

ただし，N はフォールト・トレラント・ゲートを構成する総ゲート数である．表2に示すように2入力AND，2入力OR，2入力NAND，2入力NORのフォールト・トレラント・ゲートはそれぞれ構成ゲート数 N は21個である．またNOTフォールト・トレラント・ゲートでは $N=12$，XORフォールト・トレラント・ゲートでは $N=33$ である．比較対照の通常ゲートの信頼性計算においては，最悪の場合を比較評価するため $Pd=0$ として計算すると結果は表3のようになる．

（2） XORフォールト・トレラント・ゲート

（1）と同様の手順で復帰可能な出力になる確率 R を求めれば次式のようになる．

$$R=\{(1-Pd)^6+6\cdot(1-Pd)^5\cdot Pd+9\cdot(1-Pd)^4\cdot Pd^2\}$$
$$\cdot\{(1-P)^{33}+3\cdot\sum_{n=1}^{11}\cdot{}_{11}C_n\cdot(1-P)^{33-n}\cdot P^n\}$$
$$\fallingdotseq(1-363P^2+6292P^3-59895P^4+\cdots)$$
$$\cdot(1-6Pd^2+4Pd^3+9Pd^4-12Pd^5+4Pd^6)$$

表 4 2入力 XOR-FTG の信頼性

Table 4 Reliability comparison between an Exclusive OR-FTG and an usual Exclusive OR gate.

ゲート1個の故障確率 P	通常のXORゲート $R=(1-P)^3(1-Pd)^2$	2入力 XOR-FTG $R=(1-363P^2)\cdot(1-6Pd^2)$
10^{-2}	0.97	0.891
10^{-3}	0.997	0.99891
10^{-4}	0.9997	0.99998911
10^{-5}	0.99997	0.9999998911
10^{-6}	0.999997	0.999999998911
10^{-7}	0.9999997	0.99999999998911
10^{-8}	0.99999997	0.99999999999989
10^{-9}	0.999999997	0.999999999999989

表 5 NOT-FTG の信頼性

Table 5 Reliability comparison between a NOT-FTG and an usual NOT gate.

ゲート1個の故障確率 P	通常のNOTゲート $R=(1-P)(1-Pd)$	NOT-FTG $R=(1-48P^2)(1-3Pd^2)$
10^{-2}	0.99	0.9904
10^{-3}	0.999	0.99990
10^{-4}	0.9999	0.99999904
10^{-5}	0.99999	0.9999999904
10^{-6}	0.999999	0.999999999904
10^{-7}	0.9999999	0.99999999999904
10^{-8}	0.99999999	0.999999999999990
10^{-9}	0.999999999	0.9999999999999999

$$Pd=11P$$

とみなして，Rの計算結果を表4に示す．

なお比較対照の通常の XOR ゲートは3ゲートより構成されるものとし，その伝播故障確率 Pd は0とする．このときその結果を表4に示す．

(3) NOT フォールト・トレラント・ゲート

この場合は，

$$R=\{(1-Pd)^3+3\cdot(1-Pd)^2\cdot Pd)\}\cdot\{(1-P)^{12}+3\cdot\sum_{n=1}^{4}{}_4C_n\cdot(1-P)^{12-n}\cdot P^n\}=(1-48P^2+272P^3-780P^4+\cdots)\cdot(1-3Pd^2+2Pd^3)$$

となり，表5に通常の NOT との比較を示す．

4章(1)～(3)の計算に注意してみると，フォールト・トレラント・ゲートの信頼性に関して興味ある性質をあげることができる．つまりゲートの故障確率が極めて悪い場合（例えば故障確率 $P=10^{-2}$），それらのゲートによって構成されるフォールト・トレラント・ゲートの故障確率は使用されるゲートの故障確率よりも悪くなることを計算結果は示している．

ところで，電気的構成要素の故障は一般にポアソン分布に従うものと仮定される【6】．そこでゲートの故障に対してもこの仮定を適用してみる．ゲート1個の故障確率を P，ゲート1個の故障率を λ とすると，次式で P は表わされる．

$$P=1-e^{-\lambda\cdot T}$$

そこでこの故障確率式を用いて，前述した故障確率 $P=10^{-2}$ について調べてみる．実際使用されるゲートの故障率は通常 30～300 FIT【3】程度なので $\lambda=10^{-7}$（故障数／時間）と仮定した場合，ゲートの故障確率 $P=10^{-2}$ を満たす時間 T は，$T=10^5$（時間）となる．すなわち $T=0$ におけるゲートはすべて完全な状態であったとすると，$T=10^5$ 時間後にそのゲートを検査したとき 100 個のうち1個は故障していることになる．

ところが逆に故障確率が小さい場合 $(0<P\ll10^{-2})$ には，フォールト・トレラント・ゲートの信頼性は著しく向上する．例えば 1 FIT ($\lambda=10^{-9}$) のゲートを用いてフォールト・トレラント・ゲートを構成し，$T=10^5$ 時間後にそれを検査したとすると，通常のゲート1個の故障確率 $P=10^{-4}$ からフォールト・トレラント・ゲートの故障確率 $P=10^{-6}$ に改善される．さらにゲートの故障確率が 10^{-4} より小さい（例えば，【1】検査時間 T が短いか，【2】ゲート故障率が低い）場合には，それらを用いて構成されるフォールト・トレラント・ゲートの信頼性はさらに確実に向上する．一例として，ゲートの故障確率を $P=10^{-7}$ としたとき，それを用いたフォールト・トレラント・ゲートの故障確率は $P=10^{-12}$ と大幅に改善される．これはあたかも 10^{-3} FIT の故障率のゲートを入手したことに他ならない．他方，時間 $T=1$ と仮定したとき，通常のゲート1個の故障率は 100 FIT である．

5. 信頼性と冗長度に関する考察

入出力ビット数を減らすために，データビット数に

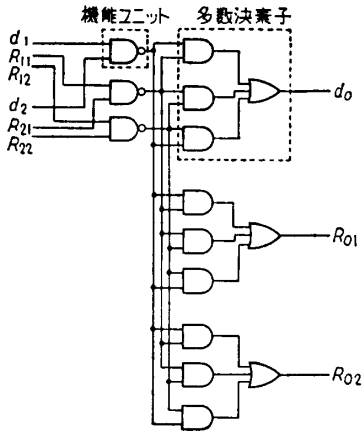


図 8 多数決方式の2入力 NAND ゲート
Fig. 8 An Example of a Voting NAND (2 inputs).

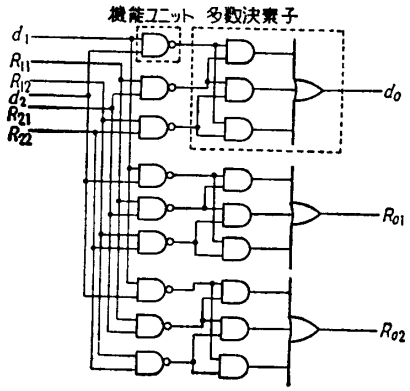


図 9 多数決方式改良型の2入力 NAND ゲート
Fig. 9 An Example of the improved Voting NAND (2 inputs).

対する冗長ビット数を減らした場合の信頼性に関して検討してみる．データ3ビットで冗長3ビットの3つの2入力 NAND ゲートを生成多項式 $G(X)=X^3+X+1$ を使って構成すると，ハードウェア量が増大し，前章で示したデータ1ビット冗長2ビットのフォールト・トレラント・ゲートを3個用いる回路に比較して，信頼性が改善できないことがわかる．

さらにデータ4ビットで冗長3ビットの4つの2入力 NAND ゲートを構成すると，データ3ビット冗長3ビットと比較して，ハードウェア量がさらに増加し，信頼性も低下する．逆にデータ1ビットに対し冗長4ビットのように冗長を増やし，1つの入力ベクトルの誤り訂正能力を2ビットまで許してフォールト・トレラント・ゲートを構成すると，信頼性はデータ1ビット冗長2ビットに比較してさらに増加する．

従来の冗長方式による2入力 NAND 回路と本方式で提案した2入力 NAND 回路の故障耐力 (fault-tolerancy) を検討してみよう．比較する回路は以下の3つである．

ⅰ) 多数決方式の2入力 NAND 回路
ⅱ) ⅰ)の多数決方式の改良型
ⅲ) 図2に示すフォールト・トレラント・ゲート回路

ⅰ)およびⅱ)は従来の技術をゲートに応用した回路であり，その論理図を図8および図9にそれぞれ示す．比較の基準を与えるために故障の発生状況に関してレベル付けを試みた．

0レベル：故障が全く存在しない．
1レベル：回路中に新しい1個の故障が発生している．
2レベル：回路の入力に1個だけ故障が混入している．
3レベル：回路中に新しい1個の故障が発生しかつ回路の入力に1個だけ故障が混入している．
4レベル：回路の各入力ベクトルについて，各々1個ずつの故障が混入している．
5レベル：回路中に新しい1個の故障が発生しかつ回路の各入力ベクトルについて，各々1個ずつの故障が混入している．
6レベル：5レベル以上の故障の混入または故障が発生している．

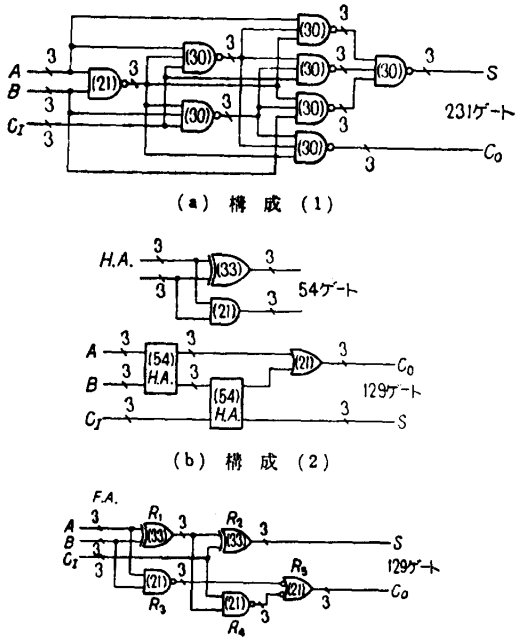
図8の多数決回路では2レベルまで復帰可能であり，図9の改良型は3レベルまで復帰可能である．本論文で提案する回路では5レベルまで復帰可能である．このような故障耐力は2つの入力ベクトル各々に対し1ビットまでの誤りを含んだ論理演算を行っても正しい出力ベクトルを生成できることに起因している．

6. フォールト・トレラント・ゲートの応用

簡単な例として，フォールト・トレラント・ゲートを用いた全加算器の構成例を示し，さらにその使用ゲート数および信頼性に関して検討する．

全加算器を構成するにあたり，多くの回路構成が考えられるが，ここでは図10に3つの異なる構成の全加算器を示す．この中でゲート数が最少で構成されているのは，図10(b)および(c)である．

一般に，ゲート総数を最少にする構成は一意には決まらない．3章(8)で示したようなフォールト・トレラント・ゲートを使用すれば，全加算器などの機能論理回路を構成する場合，全体のゲート総数を減らすことは可能である．



(a) 構 成 (1)

(b) 構 成 (2)

(c) 構 成 (3)

( ) 内はゲート数.

図 10 全加算器の構成例

Fig. 10 Examples of a Fault-tolerant Full Adder.

図10(a)の回路の使用ゲート数は231ゲート,
図10(b)の回路の使用ゲート数は129ゲート,
図10(c)の回路の使用ゲート数は129ゲートである.

次に図10(c)の全加算器に関して信頼性を検討してみる.

出力SおよびC_0が復帰可能な値をとる確率をそれぞれR_S, R_{C_0}とする.

ここで,入力A, B, C_Iはすべて正常であると仮定し,R_SおよびR_{C_0}を求めてみる.

$$R_1=(1-363P^2+6292P^3-59895P^4+\cdots)$$
$$R_2=(1-363P^2+6292P^3-59895P^4+\cdots)$$
$$\cdot\{(1-P_1)^3+3(1-P_1)^2\cdot P_1\}\quad(\because R_1=1-P_1)$$
$$R_3=(1-147P^2+1568P^3-8967P^4+\cdots)$$
$$R_4=(1-147P^2+1568P^3-8967P^4+\cdots)$$
$$\cdot\{(1-P_1)^3+3(1-P_1)^2\cdot P_1\}$$

さらに,$R_3=1-P_3, R_4=1-P_4$とすると,

$$R_5=(1-147P^2+1568P^3-8967P^4+\cdots)$$
$$\cdot\{(1-P_3)^3\cdot(1-P_4)^3+3(1-P_3)^3\cdot(1-P_4)^2\cdot P_4$$
$$+3(1-P_4)^3\cdot(1-P_3)^2\cdot P_3$$
$$+9(1-P_3)^2\cdot(1-P_4)^2\cdot P_3\cdot P_4\}$$
$$R_S=R_1\cdot R_2\fallingdotseq 1-2P_1$$
$$R_{C0}=R_1\cdot R_3\cdot R_4\cdot R_5\fallingdotseq 1-3P_3-2P_1$$

となる.

ここで,ゲート1個の故障率λを10 FITとし$T=1$(時間)とした場合のR_S, R_{C_0}を実際に計算してみると,

$$R_S=0.9999999999999274\ (約\ 10^{-4}\ \mathrm{FIT})$$
$$R_{C_0}=0.9999999999996833$$

となる.

このようにフォールト・トレラント・ゲートを用いて高い信頼性が得られる要因は,各フォールト・トレラント・ゲートの1つの入力ベクトルに対して1ビットまでの誤り訂正能力を持たせたからである.すなわち4章で示したように,あるフォールト・トレラント・ゲートの前段で起こった故障は次段のフォールト・トレラント・ゲートの故障伝播確率にはほとんど影響を与えないからである.フォールト・トレラント・ゲートの故障確率は前段を構成しているゲートの故障確率にのみ依存するだけである.

例えば前段が2入力NANDフォールト・トレラント・ゲートの場合,故障伝播確率(Pd)は$Pd\fallingdotseq 7P$となる.

等しい入力数のフォールト・トレラント・ゲートをN個用いて機能論理回路を構成する場合,すべての個所の故障伝播確率は等しいと仮定したとき,その回路全体の信頼性はR^Nである.ただしRは1つのフォールト・トレラント・ゲートの信頼性である.

しかしながら,フォールト・トレラント・ゲートを用いて今まで述べたような機能論理回路を構成する場合,次の2点を考慮する必要がある.

(1) XORを除いてフォールト・トレラント・ゲートの伝播遅延時間は通常ゲートの2段分となる.

(2) 2入力フォールト・トレラント・ゲートの出力数(fanout)が通常の1/6になる.

(2)に関しては,フォールト・トレラント・ゲートの入力段あるいは出力段に増幅器を設ければ出力数による制限は軽減される.増幅器を設けることによる信頼性低下は,たかだか故障伝播確率Pdが$Pd=N/3\cdot P$から$Pd=N/3\cdot P+P_a$になるにすぎない.ただしNを総ゲート数,Pをゲート1個の故障確率およびP_aを増幅器の故障確率とする.

7. フォールト・トレラント・コンピュータへの発展

複雑な論理演算を伴う論理回路として算術論理演算器が考えられる.算術論理演算器を高信頼化するには,3章で示したフォールト・トレラント・ゲートで

構成した方が，信頼性およびハードウェア量を考えると有利である．従来の4ビット算術演算器をフォールト・トレラント・ゲートで構成した場合，ハードウェア量で二十数倍，ゲート1個の信頼性を 10 FIT および時間Tを $T=1$ とするとその算術論理演算器の信頼性は 0.002 FIT 程度となる．

別の考え方として，個々のゲートではなく回路そのものに次数が高く符号冗長度の小さい生成多項式を適用する方法があるが，算術論理演算器の場合，冗長ハードウェア量が多いわりには誤り訂正能力が弱いので高い信頼性は得ることができない．

ところで，フォールト・トレラント・ゲートの考え方を IC メモリにも適用でき，ハードウェア量および信頼性に関して現在検討中である．

今後フォールト・トレラント・ゲートの手法をデータ転送路に適応し，その考え方を二次記憶装置にも発展させる予定である．

10 FIT のゲートを用いて，百万個のフォールト・トレラント・ゲートにより大規模な論理回路を構成したとすると，時間Tを $T=1$ とした場合，その故障率は数十 FIT となる．

このようにフォールト・トレラント・ゲートを用いることによって，超高信頼性コンピュータの実現が可能になる．

8. おわりに

通常のゲートを複数個用いてフォールト・トレラント・ゲートを構成すると，それを構成しているゲート1個あたりの信頼性より高い信頼性のゲートが実現できる．

フォールト・トレラント・ゲートで論理回路を構成すると二十数倍のハードウェア量が必要となるが，その反面信頼性は飛躍的に向上する．例えば 100 FIT 程度のゲートで構成されたフォールト・トレラント・ゲートを千個用いた回路でさえも，その全体の故障率は数 FIT に改善できる．

従来のゲートでは，故障した場合交換する以外に適当な手段はないが，フォールト・トレラント・ゲートでは，その故障の伝播を積極的に阻止し正しい動作を続けようとする．この動作は，フォールト・トレラント・ゲート中に準備された誤り訂正能力によるものである．このような考え方は，自己修復機能をもつ論理素子に根ざしており，いわば人間の細胞に相当する画期的なものであると考えられる．

したがって，フォールト・トレラント・ゲートを用いることにより，超高信頼性コンピュータ実現への新たなアプローチが可能となったといえる．

最後に，本研究の機会を与えられ，ご指導いただいた慶応義塾大学工学部相磯秀夫教授に厚く御礼申しあげます．そして終始貴重なご助言を惜しまなかった同研究室博士課程中川徹氏に深く感謝いたします．

参 考 文 献

1) 当麻喜弘：フォールト・トレラントとその関連技術，電子通信学会誌，Vol. 59, No. 4, pp. 359-368 (1976).
2) STEPHEN Y. H. SU. et al.: An Overview of fault-tolerant digital system architecture, Proc. NCC, 46, pp. 19-26 (1977).
3) 藤木正也，塩見 弘：エレクトロニクスにおける信頼性，電子通信学会編，コロナ社 (1978).
4) DeSOUSA, P. T. et al.: Modular redundancy without voters decreases complexity of restoring organ, Proc. NCC, 46, pp. 801-806 (1977).
5) 宮川 洋，岩垂好裕，今井秀樹：符号理論，昭晃堂 (1973).
6) ALGIRDAS AVIZIENIS: Fault-tolerance: The Survival Attribute of Digital Systems, Proc. The IEEE, Vol. 66, No. 10, pp. 1109-1125 (1978).

(昭和54年8月31日受付)