A Novel Approach to Fault-Tolerant Logic

YOSHIYASU TAKEFUJI* and MASAHIRO IKEDA*

This paper presents a design scheme that supplies logic circuits with very high reliability by providing redundancy to gate function logic using conventional gates. The basic redundant logic circuits are called Fault-Tolerant Gates (FTGs). The construction and design of FTGs, such as AND, OR, NOT, NAND, NOR and Exclusive OR gates, are described. The improved reliability of these FTGs is evaluated in comparison with conventional gates. An FTG is shown to have a stronger restoring function than any voting circuit. The reliability improvements for the proposed FTGs are discussed for Full Adder and Arithmetic Logic Unit (ALU) applications. The reliability of VLSIs employing FTGs is discussed in terms of the development of ultra reliable computers. The new design scheme not only greatly increases the reliability of logic circuits, but also may improve the yield of VLSIs.

1. Introduction

Hardware redundancy systems allow the formation of fault-tolerant circuits [1].

There are three different redundancy systems: [2].

1. Static
2. Dynamic
3. Hybrid

The same functional units are used in a static redundancy system, and majority logic is applied in these units. Majority logic requires voting circuity (VOTER).** The reliability of such systems never exceeds that of the VOTER. In a dynamic redundancy system, faults are either detected by the detector provided, or corrected by the self-restoring function of a corrector [3]. Extra component like detectors or correctors in fault-tolerant circuits, form bottlenecks in terms of reliability. A hybrid redundancy system combines the static and dynamic redundancy systems and has the same reliability drawbacks.

Because each VOTER or COLLECTOR [4],*** is composed of two or more gates, the reliability of the gates is lower than that of a single gate. Functional logic circuits cannot provide higher reliability than a gate used by these redundancy systems. To achieve reliability higher than a single gate, the reliability of gate function logic, which is the basic component of functional logic circuits, must be raised. Conventionally, the increase of reliability depends upon semiconductor technology.

The design presented here is for a gate function module, which consists of a Fault-Tolerant Gate (FTG) formed from conventional gates; the reliability of the gate function module is higher than that of a single gate. To realize the higher reliability, correct logical operation should occur in an FTG even if inputs fail. This paper presents a design scheme for FTGs that increases the reliability of gate function logic as a basic component of logic circuits. AND, OR, NOT, NAND, NOR, and Exclusive OR FTG designs have been made. The reliability improvement and the application to functional logic circuits are discussed. Results show that a fault-tolerant logic circuit that has greater reliability than a single gate can be easily realized with FTGs. It has been proved that highly reliable fault-tolerant computers also can be realized by a design adopting ultra reliable FTGs.

2. Fault-Tolerant Gates (FTGs)

An FTG is a new logic element that has optimal redundancy of both I/O signal lines and gate function. The concept of an Error Correcting Code (ECC) is taken into account in the FTGs error correction.

2.1 Principles of Fault-Tolerant Gates

It is assumed that an input vector, as well as an output vector, includes both m actual data lines and n redundant data lines. For example, where $m=1$ and $n=4$, two bit errors can be corrected in a five-bit vector, because the number of vectors is two, and the Hamming distance is five.

Generally, up to t bit errors can be corrected by a Hamming distance greater than $2t+1$ [5].

An FTG is a combinational circuit that can produce a correct output vector in logical operation, even if the input vectors include errors.

An example of a NAND-FTG (two input vectors) and its logic symbol are shown in Fig. 1. An example of an error correction in a NAND-FTG (two input vectors) is shown in Fig. 2 and described in Table 1.

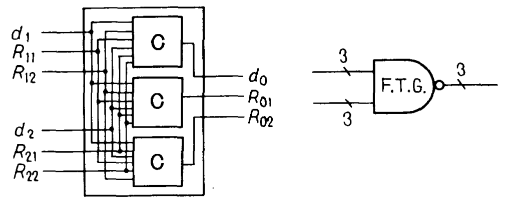
In the example, the input/output vector consists of three bits (one data bit and two redundant bits).

The relation between the data bit and redundant bits is

*Department of Electrical Engineering, Keio University, Yokohama 223, Japan.

**A "Voter" is a decision circuit for majority logic that has odd inputs.

***A "Collector" is a circuit similar to a Voter. Its output is determined by both the masked signals from a detector and signals from a comparator.



C: Combinatorial Circuit
(a) Logic Structure (b) Symbol of a NAND-FTG

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

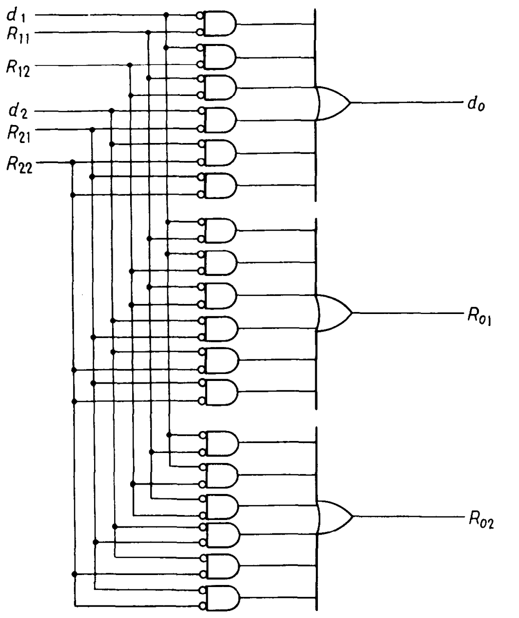


Fig. 2 Logic diagram of a NAND-FTG (2 input vectors).

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

Input Vector			Input Vector			Output Vector		
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

Notes: 1. d_1 and d_2 are input data bits. R_{11}, R_{12}, R_{21} and R_{22} are redundant input bits.
2. d is output data bit. R_{01} and R_{02} are redundant output bits.
3. "○" means error bit.

determined by a generator polynomial. After careful consideration, described in Chapter 3, $G(x)=x^2+x+1$ has been adopted as the generator polynomial. The generator polynomial determines two vectors, (000) and (111), as correct words.

A correct output vector can be produced, even if each input vector includes up to one error bit.

Input/output vectors have the following patterns.

$$\left.\begin{matrix}①\,0\,0\\ 0\,①\,0\\ 0\,0\,①\end{matrix}\right\}\text{Words are regarded as a (000)}$$

vector.

Also

$$\left.\begin{matrix}⓪\,1\,1\\ 1\,⓪\,1\\ 1\,1\,⓪\end{matrix}\right\}\text{Words are regarded as a (111) vector.}$$

If any combination of the above six input vectors, including one-error bit, is the input to an FTG, the FTG will produce a correct output vector.

The design process of FTGs that have minimum hardware and maximum redundancy is as follows:

(i) To determine the redundancy of input/output lines. (m data bits, n redundant bits)
(ii) To determine a generator polynomial taking into consideration the number of gates required to realize an FTG.

In the next chapter, concrete design examples of FTGs are shown.

3. Design Examples of Fault-Tolerant Gates

When considering simple interfaces between FTGs in terms of the combination of the number of data lines and the number of redundancy lines, the combination of one data line with more than one redundancy line makes simpler hardware for an FTG possible than in the case of other combinations.

The simplest combination is one data bit plus two redundancy bits.

Clearly, the use of $G(x)=x^2+x+1$ as a generator polynomial is reasonable. The generator polynomial is adopted for the following reasons:

(i) A one-bit error correction in an input vector needs more than a three-bit vector length.
(ii) A circuit to generate a data bit, and a circuit to generate redundancy bits, have the same structure by the use of $G(x)=x^2+x+1$ as the generator polynomial.
(iii) A data bit and the redundancy bits do not have to be distinguished.

AND, OR, NOT, NAND, NOR and Exclusive OR FTGs are designed with a data bit and two redundancy bits used as a vector, and $G(x)=x^2+x+1$ used as the generator polynomial. A design for a NAND-FTG will be described first because of its simple logical design.

(1) NAND-FTG (two input vectors)

The logical operation of an output data bit is equivalent to that of the output redundancy bits because of the use of $G(x)=x^2+x+1$.

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

The logical operation truth table shown in Table 1 transforms into the following Boolean equation.

$$d_0=\bar{d}_1\cdot\bar{R}_{11}+\bar{d}_1\cdot\bar{R}_{12}+\bar{R}_{11}\cdot\bar{R}_{12}+\bar{d}_2\cdot\bar{R}_{21}+\bar{d}_2\cdot\bar{R}_{22}+\bar{R}_{21}\cdot\bar{R}_{22}, \tag{2}$$

A NAND-FIG (two input vectors) is designed with the use of equations (1) and (2). The number of gates required to realize the FTG is 21.*

(2) NOR-FTG (two input vectors)

The output data bit do is expressed by the Boolean equations:

$$d_0=R_{01}=R_{02}, \tag{3}$$

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

To reduce the number of gates required to realize the FTG, Eq. (4) is transformed into:

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

A NOR-FTG (two input vectors) is designed with the use of Eqs. (3) and (5), as shown in Fig. 3.

(3) OR-FTG (two input vectors)

The output data bit d_0 is expressed by the Boolean equations:

$$d_0=R_{01}=R_{02}, \tag{6}$$

$$d_0=d_1\cdot R_{11}+d_1\cdot R_{12}+R_{11}\cdot R_{12}+d_2\cdot R_{21}+d_2\cdot R_{22}+R_{21}\cdot R_{22}. \tag{7}$$

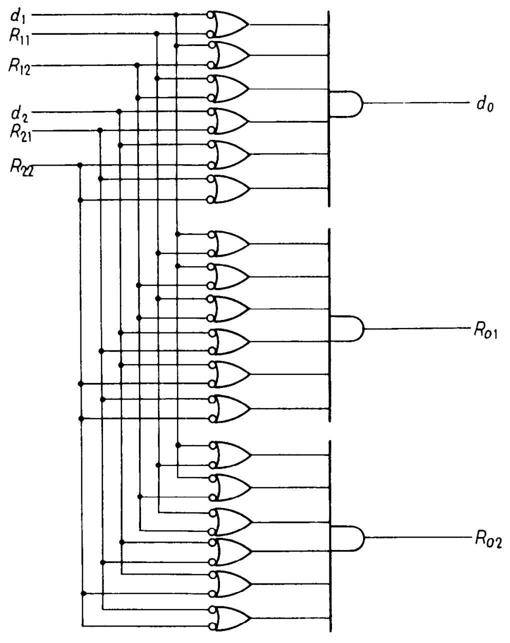


Fig. 3 Logic diagram of a NOR-FTG (2 input vectors).

Because an FTG with three input vectors can be realized with gates that allow up to nine inputs, it is assumed that the average number of inputs for a gate in an LSI is three.

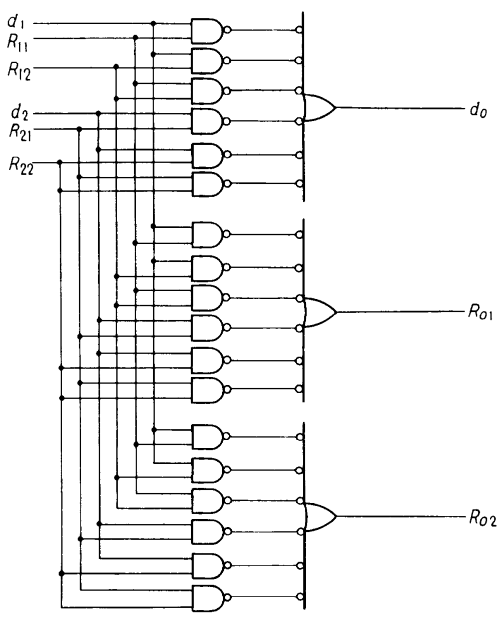


Fig. 4 Logic diagram of an OR-FTG (2 input vectors).

The OR-FTG (two input vectors) design uses Eqs. (6) and (7), as shown in Fig. 4.

(4) AND-FTG (two input vectors)

The output data bit d_0 is expressed by the Boolean equations:

$$d_0=R_{01}=R_{02}, \tag{8}$$

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

To reduce the number of gates required, Eq. (9) is transformed into:

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

An AND-FTG (two input vectors) design uses Eqs. (8) and (10), as shown in Fig. 5.

(5) Exclusive-OR-FTG

The output data bit d_0 is expressed by the Boolean equations:

$$d_0=R_{01}=R_{02}, \tag{11}$$

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

An Exclusive OR-FTG design uses Eqs. (11) and (12), as shown in Fig. 6.

(6) NOT-FTG

$$d_0=R_{01}=R_{02}, \tag{13}$$

$$d_0=\bar{d}_1\cdot\bar{R}_1+\bar{R}_1\cdot\bar{R}_2+\bar{d}_1\cdot\bar{R}_2. \tag{14}$$

*It is assumed that the maximum number of inputs to a normal gate is nine.

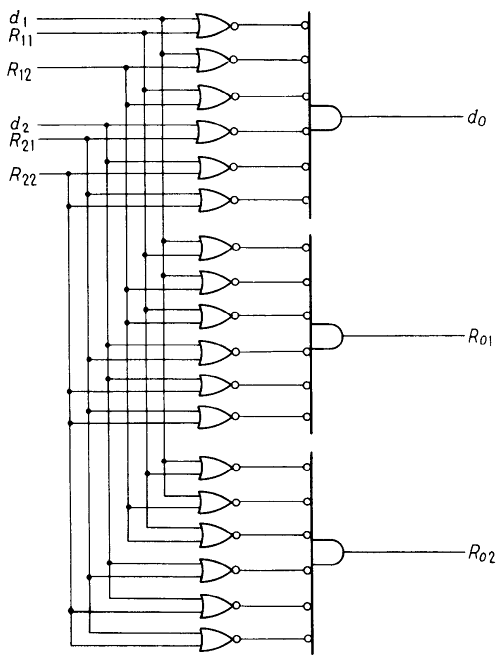


Fig. 5 Logic diagram of an AND-FTG (2 input vectors).

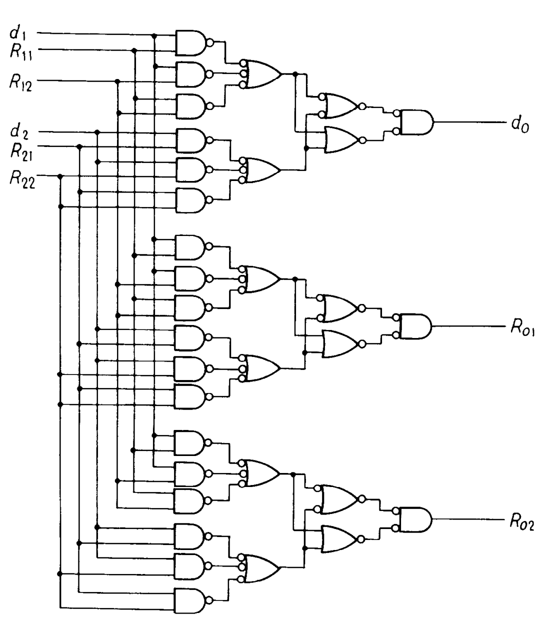


Fig. 6 Logic diagram of an Exclusive OR-FTG (2 input vectors).

A NOT-FTG design uses Eqs. (13) and (14), as shown in Fig. 7.

(7) NAND-FTG (five input vectors)

The d_0 for the NAND-FTG (five input vectors) is expressed by Boolean Eqs. (15) and (16):

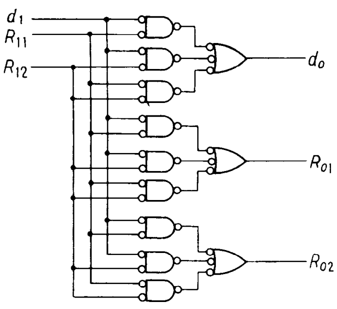
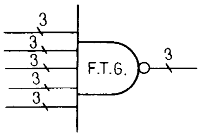


Fig. 7 Logic diagram of a NOT-FTG

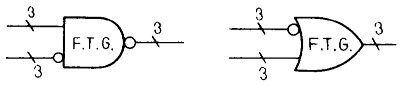


$$d_0=R_{01}=R_{02}, \tag{15}$$

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

(8) Other FTGs

The number of required gates to realize each of the following FTGs (two input vectors) is equal to the number of gates in a NAND-FTG (two input vectors). Boolean equations are: ·



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

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

The number of gates required to realize FTGs from (1) to (6) are compared in Table 2.

A close relationship exists between the number of gates and reliability. Generally, a greater number of gates decreases reliability. Increasing the number of gates for an FTG increases the possibility of a failure propagating into the FTGs in succeeding stages. However, the probability that an FTG can survive is greater than the probability of normal gate failure, because of its built-in restoring function. The reliability of FTGs will be discussed in the next chapter.

4. FTG Reliability

Is it possible to make an FTG that has greater reliability than the individual gates used? The answer to this question will be provided after investigating the reliability of such FTGs as a NAND-FTG (two input

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

	AND-FTG	OR-FTG	NOT-FTG	NAND-FTG	NOR-FTG	XOR-FTG
Logic circuit	2 input vectors	2 input vectors		2 input vectors	2 input vectors	2 input vectors
No. of gates	$3\times7=21$	$3\times7=21$	$3\times4=12$	$3\times7=21$	$3\times7=21$	$3\times11=33$

vectors), an Exclusive OR-FTG (two input vectors), and a NOT-FTG.

(1) NAND-FTG (two input vectors)

It is assumed that R is the possibility that an FTG can produce a recoverable output vector, even if each input vector to the FTG includes up to one errorneous bit; that P is the possibility of a gate failure; and that Pd is the possibility of an error propagating into suceeding FTGs. V is the possibility that two input vectors are recoverable. Each input vector can tolerate up to one erroneous bit, so V is the summation of V_0, V_1, and V_2, where V_0 is the possibility that two input vectors include no errors, V_1 is the possibility that two input vectors include only one erroneous bit, and V_2 is the possibility that each input vector includes one erroneous bit. V is expressed by:

$$\begin{aligned} V&=V_0+V_1+V_2\\ &=(1-Pd)^6+6(1-Pd)^5\cdot Pd+9(1-Pd)^4Pd^2. \end{aligned}$$

On the other hand, the possibility that there is no failure in all 21 gates of the FTG is $(1-P)^{21}$. The possibility that only one gate will fail in all 21 gates is ${}_{21}C_1\cdot(1-P)^{20}\cdot P$. The possibility that the FTG can produce a recoverable output vector even if n gates $(2\leq n\leq 7)$ fail in all 21 gates is

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

Thus, R, for the possibility that an output vector is recoverable, is:

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

Pd, the possibility of a failure propagating into the succeeding FTGs, is nearly determined by the number of gates required for the FTG. Because propagated failures are corrected by an FTG, the possibility of failure propagating into succeeding FTGs can be neglected when the following equation for Pd is taken into account:

$$\begin{aligned} Pd_i=&1-(1-P)^7\cdot[(1-Pd_{i-1})^6+6\cdot(1-Pd_{i-1})^5\\ &\times Pd_{i-1}+9\cdot(1-Pd_{i-1})^4\cdot Pd_{i-1}^2], \end{aligned}$$

where

Pd_{i-1} is the possibility of a failure propagating into the succeeding FTGs (two input vectors). Generally, Pd_{i-1} and P are so small $(1\gg Pd_{i-1}>0,\ 1\gg P>0)$ that only Pd_{i-1} and P need be considered, with Pd_{i-1}^2, $Pd_{i-1}^3,\cdots$, Pd_{i-1}^6 and $P^2P^3,\cdots$, P^7 being neglected. In this case, Pd_i is $Pd_i\fallingdotseq 7\cdot P$. In order to estimate the possibility R, that an FTG can survive, the possibility of propagation failure Pd is assumed to be:

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

where, N is the total number of gates for the FTG. The total number of gates is shown in Table 2.

N of a NAND-FTG (2 input vectors) is 21.

N of an Exclusive OR-FTG (2 input vectors) is 33.

N of a NOT-FTG is 12.

To pessimistically compare the reliability R of a NAND-FTG with that of a normal gate, Pd in R of a normal gate is evaluated as zero. The comparison is shown in Table 3.

(2) Exclusion OR-FTG

R of the possibility that an Exclusive OR-FTG can survive is estimated in the same manner as for the NAND-FTG:

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

Where, $Pd=11P$.

The R of a normal gate is compared with that of an FTG in Table 4. The Exclusive OR gate is assumed to be composed of three gates.

Table 3 Reliability comparison between a NAND-FTG (2 input vectors) and a normal NAND gate.

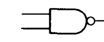
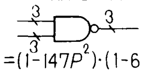
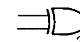
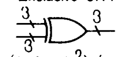
Probability of a Gate Failure P	Normal NAND gate  $R=(1-P)\cdot(1-Pd)^2$	NAND-FTG  $R=(1-147P^2)\cdot(1-6Pd^2)$
10^{-2}	$0.\underline{99}$	0.956
10^{-3}	$0.\underline{999}$	$0.\underline{999}56$
10^{-4}	$0.\underline{9999}$	$0.\underline{9999}9559$
10^{-5}	$0.\underline{99999}$	$0.\underline{99999}99559$
10^{-6}	$0.\underline{999999}$	$0.\underline{999999}999559$
10^{-7}	$0.\underline{9999999}$	$0.\underline{9999999}9999559$
10^{-8}	$0.\underline{99999999}$	$0.\underline{99999999}99999559$
10^{-9}	$0.\underline{999999999}$	$0.\underline{999999999}999999559$

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

Probability of a Gate Failure P	Normal Exclusive OR  $R=(1-P)^3(1-Pd)^2$	Exclusive OR-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	0.9999999999999989
10^{-9}	0.999999997	0.999999999999999989

Pd for the normal gate is pessimistically regarded to be zero.

(3) NOT-FTG

R of a NOT-FTG is:

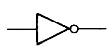
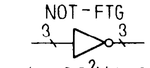
$$\begin{aligned}R&=\{(1-Pd)^3+3\cdot(1-Pd)^2\cdot Pd)\}\\&\quad\times\left\{(1-P)^{12}+3\cdot\sum_{n=1}^{4}\cdot{}_4C_n\cdot(1-P)^{12-n}\cdot P^n\right\}\\&=(1-48P^2+272P^3-780P^4+\cdots)\\&\quad\times(1-3Pd^2+2Pd^3).\end{aligned}$$

The evaluation of R is shown in Table 5. The advantageous reliability of FTGs is shown in comparisons of Table 3 and 5. If the probability of gate failure, P, is greater than 10^{-2}, the tables indicated that the reliability of the FTG is lower than that of the single gate. It is generally assumed that failures of electrical elements depend on the Poisson distribution [6]. Applying this assumption to the failure of a gate, probability P of a gate's failure is expressed by the equation:

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

Where, λ is the failure rate of the gate. The critical probability of $P=10^{-2}$ has been investigated using this equation. The probability of a normal gate failure is from 30 FIT to 300 FIT, assuming $\lambda=10^{-7}$ (=100 FIT) (failures/hour) and time T is $T=10^5$ (hour) to satisfy $P=10^{-2}$. This indicates that 1% of gates will fail at time $T=10^5$ (hour), after testing all gates to determine that there is no error in any gate at time $T=0$. However, when the probability of gate failure is less than 10^{-2}, the reliability of an FTG gate configuration is significantly increased. For example, when using gates, the reliability of which is 1 FIT ($\lambda=10^{-9}$), to make an FTG, the probability of FTG fatal failure is $P=10^{-6}$ at $T=10^5$ (hour), and the probability of gate failure is $P=10^{-4}$. When the probability of gate failure is less than 10^{-4} (i.e., i. mission duration time T is sufficiently short to satisfy this or, ii. the failure rate of a gate is small), the reliability of the FTG consisting of the gates is further improved. For example, when the probability of a gate is $P=10^{-7}$, the reliability of an FTG composed of the gates is $P=10^{-12}$. This means that a gate module can be made that has a failure rate of 10^{-3} FIT; on the other hand, that of the primitive gate is 100 FIT for the assumption $T=1$.

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

Probability of a Gate Failure P	Normal NOT gate  $R=(1-P)(1-Pd)$	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	0.99999999999999999

5. Relation between Reliability and Redundancy

To reduce the number of input-output signal lines, the ratio of redundancy lines to actual data lines must be improved.

Considering three NAND gates (each NAND gate has two inputs), three redundancy bits per three data bits and employed for each vector, the redundancy bits are determined by $G(x)=x^3+x+1$ as the generator polynomial, and the amount of hardware to realize the three NAND redundant gates increases above that required for three NAND-FTGs, in which two redundancy bits per data bit are employed for each vector. Thus, the reliability in the case where three redundancy bits per three data bits are employed as a vector cannot be improved. Where three redundancy bits per four data bits are employed as a vector for four NAND gates (two input vectors), much more hardware is needed than in the above cases. The greater amount of hardware in such cases decreases reliability, in general, because only a one-error bit correction capability per vector is in the hardware. Thus, multiple error-bit correction must be provided for each vector to increase reliability. For example, when four redundancy bits per data bit are provided for each vector, up to two bit errors per vector can be corrected for an FTG. To demonstrate the fault tolerance of an FTG, as shown in Fig. 2, the FTG is compared with both an existing voting (Fig. 8) and the improved voting circuit (Fig. 9) for the classification of fault-occurrences in logic circuits.

Fault-occurrences are classified as follows:

Level 0:	No faults occur in a logic circuit.
Level 1:	A new fault occurs in a logic circuit.
Level 2:	A fault is propagated to a logic circuit.
Level 3:	A new fault occurs in a logic circuit, and a fault is propagated to a logic circuit.
Level 4:	A fault is propagated to each input vector of a logic circuit. An input vector consists of more than one input.
Level 5:	A fault occurs in a logic circuit, and a fault is propagated to each input of the logic circuit.
Level 6:	More than one fault occurs in a logic circuit, or more than one fault is propagated to each input of the logic circuit.

The voting circuit can tolerate conditions up to Level 2. The improved voting circuit can tolerate conditions up to Level 3. The FTG proposed in this paper can tolerate conditions up to Level 5, owing to efficient error correction. The complexity is the same as the improved voting circuit, and delay is one stage less. The voting circuit has 30% fewer gates than the FTG, but also has one extra gate delay.

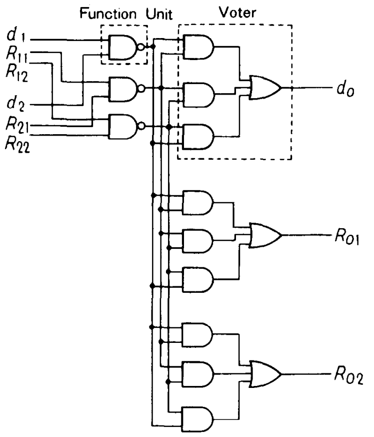


Fig. 8 An Example of a Voting NAND (2 input vectors).

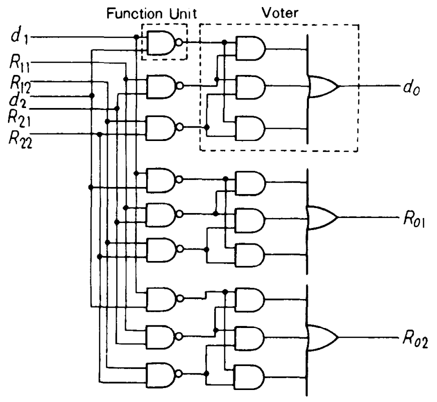


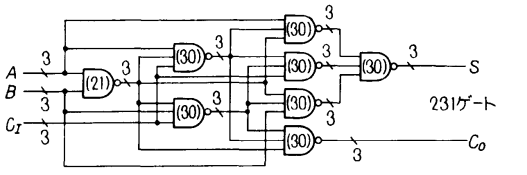
Fig. 9 An Example of the improved Voting NAND (2 input vectors).

6. Application of FTGs

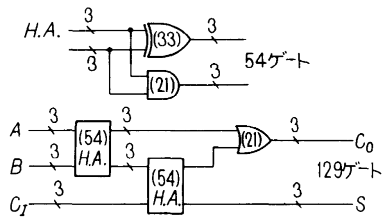
A fault-tolerant, Full-Adder design is proposed using the FTGs. The number of gates for the Full Adder and its reliability will be discussed. Fig. 10 shows arrangements for Full Adders. Circuits (b) and (c) each have fewer gates than (a). It is not easy to make fault-tolerant functional logic (FTL) with a minimum amount of hardware. The use of the FTG shown in Chapter 3–(8). makes it possible to reduce the number of gates required for an FTL like Full Adder. Circuit (a) shown in Fig. 10 consists of 231 gates, circuit (b) 129 gates, and circuit (c) 129 gates.

The reliability of Full Adder (c) will be discussed. It is assumed that the probability of no error and one bit error in S is R_s, and in C_0 is R_{c0}. Assuming there are no errors in all input vectors (A, B, Ci), R_s and R_{c_0} are expected to be:

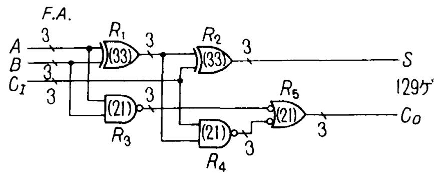
$$R_1=(1-363P^2+6292P^3-59895P^4+\cdots),$$
$$R_2=(1-363P^2+6292P^3-59895P^4+\cdots)$$
$$\times\{(1-P_1)^3+3(1-P_1)^2\cdot P_1\}(\therefore R_1=1-P_1),$$
$$R_3=(1-147P^2+1568P^3-8967P^4+\cdots),$$
$$R_4=(1-147P^2+1568P^3-8967P^4+\cdots)$$
$$\times\{(1-P_1)^3+3(1-P_1)^2\cdot P_1\}.$$



(a) Structure (1)



(b) Structure (2)



(c) Structure (3)

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

Assuming, $R_3 = 1 - P_3$ and $R_4 = 1 - P_4$:

$$\begin{aligned} R_5 = {} & (1 - 147P^2 + 1568P^3 - 8967P^4 + \cdots) \\ & \times \{(1-P_3)^3 \cdot (1-P_4)^3 + 3(1-P_3)^3 \cdot (1-P_4)^2 \cdot P_4 \\ & + 3(1-P_4)^3 \cdot (1-P_3)^2 \cdot P_3 \\ & + 9(1-P_3)^2 \cdot (1-P_4)^2 \cdot P_3 \cdot P_4\}, \end{aligned}$$

$$R_S = R_1 \cdot R_2 \fallingdotseq 1 - 2P_1,$$

$$R_{C_0} = R_1 \cdot R_3 \cdot R_4 \cdot R_5 \fallingdotseq 1 - 3P_3 - 2P_1.$$

Thus, when assuming the failure rate per gate is 10 FIT and time $T = 1$, R_s and R_{c_0} are:

$$R_S = 0.9999999999999274$$

$$R_{C_0} = 0.9999999999998833.$$

The high reliability of the Full Adder results from the error-correcting capability of up to one bit error per input vector of each FTG.

As described in Chapter 4, a fault occurring in an FTG seldom affects the probability of a propagation failure to succeeding FTGs. Thus, the probability that an FTG cannot survive depends almost completely on only the probability of FTG failure, and less on the probability of propagation failure from FTG in preceding states. For example, the probability of propagation failure to/from NAND-FTG (two input vectors), is $Pd = 7 \cdot P$, where P is the probability of conventional gate failure. Assuming the probabilities of propagation failure in any FTG node are equal, the reliability of the total system is R^N, where R is the probability that an FTG can survive, and N is the number of FTGs. However, the following two points have to be considered in designing an FTL:

(1) The propagation delay time of an FTG (two stage- gate delay time except for an Exclusive OR-FTG)
(2) Fan-out limitation

When using $G(x) = x^2 + x + 1$ as the generator polynomial, fan-out is limited to 1/6.

Relevant to (2), the fan-out limitation can be lightened by the use of higher-gain amplifiers in I/O stages.

The reduction in reliability caused by the use of such amplifiers is $Pd = NP/3 + Pa$, where N is the number of gates, P is the probability of gate failure, and Pa is the probability of amplifier failure.

7. Fault-Tolerant Computers

An Arithmetic Logic Unit (ALU) is a complicated logic circuit. In order to produce a Fault-Tolerant ALU, a design scheme using FTGs is reasonable in terms of both reliability and the amount of hardware required. When making a 4-bit Fault-Tolerant ALU with FTGs, the amount of hardware needed is about 20 times that needed for a normal 4-bit ALU. Assuming 10 FIT to failure rate per gate, and $T = 1$, the failure rate of the ALU is 0.002 FIT.

By contrast with the proposed design, a design that directly applies error correction to an ALU by the use of a less redundant code, such as three redundancy data bits per four data bits, cannot obtain such high reliability. The fault-tolerancy of the ALU is lower than that of the proposed ALU that is composed only of FTGs because of the increased amount of redundancy hardware. FTGs can also be used in memory systems; the reliability and amount of hardware required for this usage is now under investigation. In the future, FTGs will be used for data buses and peripheral storage units. A very large scale integration circuit of a million FTGs employing normal gates whose failure rate is assumed to be 10 FIT and time $T = 1$, will have a failure rate of less than 100 FIT. Ultra reliable computers can thus be realized by the use of FTGs.

8. Conclusion

Ultra reliable gate function logic can be realized by the use of FTGs composed of gates that have a reliability greater than that of the individual gates used. A circuit composed only of FTGs requires over 20 times the amount of hardware that a normal circuit requires, but its reliability is significantly improved. For example, if 100 FTGs, each composed of normal gates with a failure rate of 100 FIT per gate, are employed to realize an FTL, the failure rate of the total circuit can be improved to a few FIT. If a gate failure occurs, the failed gate has to be removed from a nonredundant circuit to achieve recovery. However, even if gate failure occurs in an FTG, the FTG can prevent the failed signal from spreading, and will continue to operate normally. The recovery action comes from the error-correction capability of the FTG.

This capability seems to resemble the recovery action of a neuron in the human brain; i.e., like a self-restoring organ.

Thus, ultra reliable computers can be realized by the use of FTGs.

Acknowledgements

We wish to thank Professor Hideo Aiso at Keio University and Messrs. T. Nakagawa, S. Sugihara, Y. Adachi, J. Tsuji and J. Miyazaki for their advice.

We would like to acknowledge our colleagues at Keio University for their enthusiastic interest and suggestions.

References

1. TOHMA, Y. Fault-Tolerant Computing and the Associated Technologies, *Journal of the IECE of Japan.* **59**, 4 (1976) 359–368.
2. Y. H. SU., Stephen, et al. An Overview of Fault-Tolerant digital system architecture, *Proc. NCC*, **46** (1977) 19–26.
3. FUJIKI, M. et al. Reliability in Electronics, Korona press in 1978.
4. P. T., DeSOUSA, et al. Modular redundancy without voters decreases complexity of restoring organ, *Proc. NCC*, **46** (1977) 801–806.
5. MIYAZAWA, Y. et al. Coding theory, Shokodo press in 1973.
6. ALGIRDAS AVIZIENIS, Fault-tolerance: The Survival Attribute of Digital Systems, *Proc. The IEEE*, **66**, 10 (1978) 1109–1125.