最小テスト集合でテスト可能な加算器について

梶原誠司[†,††] 笹尾勤[†,††]

本論文では，順次桁上げ方式と桁上げ先見方式による2種類の加算器に関して，縮退故障に対する最小テスト集合について述べる．論文の前半では，5つの2入力ゲートで構成された全加算器の構成を2種類示し，それらは2入力ゲートのみで構成される全加算器としてはゲート数最小であること，そのうちの1つは，3パターンですべての単一縮退故障をテストできることを示す．また，4ビットの桁上げ先見加算器を2種類示し，それぞれ10および12個のテストパターンが単一縮退故障に対する最小テスト集合であることを示す．論文の後半では，直列に接続された n ビット加算器について考察する．まず，n ビットの順次桁上げ加算器は，n の値にかかわらず単一縮退故障を全加算器と同じテストパターン数でテスト可能であることを述べる．次に，多重縮退故障に対しても6つのテストパターンでテスト可能で，それが最小のテスト集合であることを示す．さらに，$4m$ ビットの桁上げ先見加算器に対しては，m の値にかかわらず大きさが12以下のテスト集合で単一縮退故障を検出できることを示す．

On the Adders with Minimum Tests

Seiji Kajihara[†,††] and Tsutomu Sasao[†,††]

This paper considers two types of n-bit adders, ripple carry adders and cascaded carry look-ahead adders, with minimum tests for stuck-at fault models. In the first part, we present two types of full adders consisting of five gates, and show their minimality: the adders contains the minimum number of gates among adders consisting of only 2-input gates. We also prove that one of the full adders can be tested by only three test patterns for single stuck-at faults. We also present two types of 4-bit carry look-ahead adders, and prove that the sizes of the minimum tests are 10 and 12 for single stuck-at faults. In the second part, we consider the tests for the cascaded adders and show the followings: Single stuck-at faults in an n-bit ripple carry adder can be detected by the same size of tests as those of a full adder, and six tests are sufficient even for multiple stuck-at faults. For the $4m$-bit cascaded carry look-ahead adders, the sizes of the minimum tests are less than 12 for single stuck-at faults. Note that the sizes of the minimal tests do not depend onthe value of n nor m. These tests are considerably smaller than previously published ones.

1. はじめに

加算器は，多くのLSIの算術演算回路に使われている．加算器の基本ユニットは，全加算器である．**図1**と**図2**にEXORゲートを用いた2種類の全加算器の実現方法を示す．この全加算器を直列に接続することで，**図3**に示す順次桁上げ加算器（ripple carry adder）を構成できる．図3のような回路構成は繰り返し論理アレイ（iterative logic array）と呼ばれている．繰返し論理アレイの用語を用いれば，順次桁上げ加算器はアレイであり，全加算器はセルである．繰返し論理アレイのセルの故障は，故障したセルがその論理的動作を変えるがその故障動作は組合せ回路としてのものであると仮定する．単一セル故障モデル（回路内にたかだか1つの故障セルが存在するモデル）では，任意の n ビットの順次桁上げ加算器が，n の値にかかわらず，8つのテストパターンでテストできる[7]．また，多重セル故障モデルでは，n ビットの順次桁上げ加算器は，11個のテストパターンでテストできる．さらに，各セルが p ビットの直列加算器のテストパターン数についても考察されており，$3 \cdot 2^p - 1$ になる[7]．順次桁上げ加算器は構造が簡単で，n の値が小さいときにはよく使われている．たとえば，INTEL 8080マイクロプロセッサには順次桁上げ加算器が使用されて

† 九州工業大学情報工学部
Department of Computer Science and Electronics, Kyushu Institute of Technology
†† 九州工業大学マイクロ化総合技術センター
Center for Microelectronic Systems, Kyushu Institute of Technology

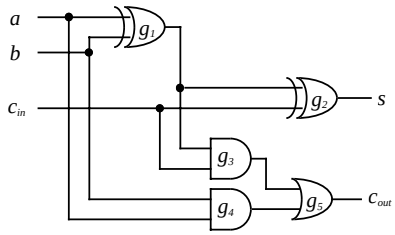

図 1 AND，OR，EXOR ゲートを用いた全加算器
Fig. 1 Full adder using AND, OR, EXOR gates.

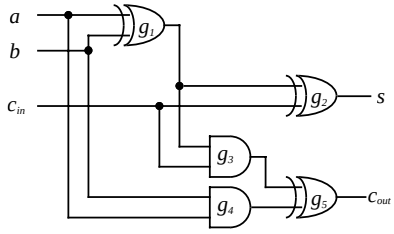

図 2 AND，EXOR ゲートを用いた全加算器
Fig. 2 Full adder using AND, EXOR gates.

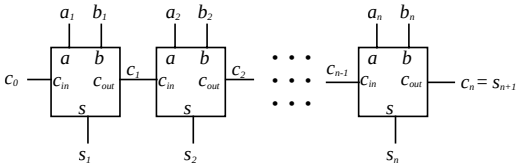

図 3 順次桁上げ加算器
Fig. 3 Ripple carry adder.

いた[14]．しかしながら，回路の遅延時間の最大値が n の値に比例するという欠点があるため，高速な加算が必要なときには桁上げ先見加算器（carry look-ahead adder）がよく用いられている．**図 4** と**図 5** に 2 種類の 4 ビット桁上げ先見加算器の実現方法を示す．桁上げ先見加算器を構成する各セルの単一縮退故障のテストについては，Becker が $6 \cdot (\log_2 n) - 4$ パターンで十分であることを示している[3]．本論文では，縮退故障に対して最小のテスト集合でテスト可能な順次桁上げ加算器と桁上げ先見加算器の実現方法について考察する．まず，図 1 と図 2 の回路は 2 入力ゲートのみで構成された全加算器の中で最小個のゲートを含んでいることを示す．次に，これらの全加算器の単一縮退故障に対する最小のテスト集合の大きさは，図 1 と図 2 の回路で，それぞれ 5 および 3 になることを示す．どのような 2 入力の論理ゲートに対しても少なくとも 3 つのテストパターンが必要なことから，図 2 の全加算器は最小のパターン数でテストできる全加算器であるといえる．また，図 4 と図 5 の 4 ビット桁上げ先見加算器の単一縮退故障に対する最小テスト集合

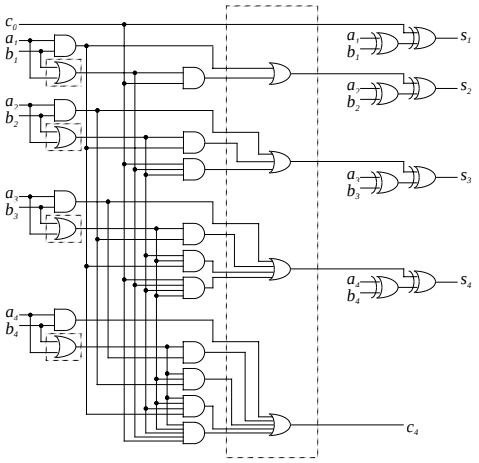

図 4 AND，OR，EXOR ゲートを用いた桁上げ先見加算器
Fig. 4 Carry look-ahead adder using AND, OR, EXOR gates.

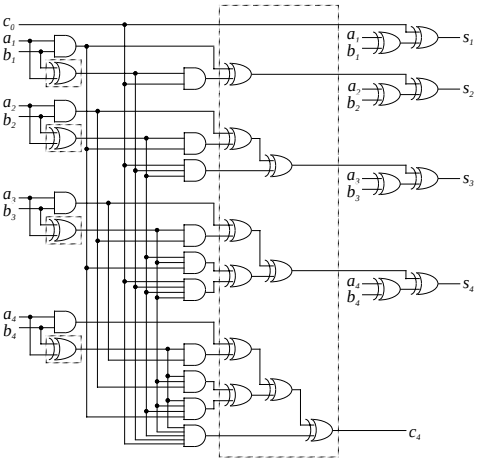

図 5 AND，EXOR ゲートを用いた桁上げ先見加算器
Fig. 5 Carry look-ahead adder using AND, EXOR gates.

の大きさは，それぞれ 10 および 12 であることを示す．さらに，繰返し論理アレイによる加算器のテストについて考察する．n ビットの順次桁上げ加算器の場合には，単一縮退故障は各セルの最小テスト集合と同じ大きさのテスト集合で，多重縮退故障は 6 個のテストパターンでテストできることを示す．$4m$ ビットの直列の桁上げ先見加算器の場合には，単一縮退故障は最小テスト集合の大きさは 12 以下であることも示す．本論文で述べるテストパターン数は，たとえば 8 ビットの加算器に対して文献 7) で必要とするパターン数 $3 \cdot 2^8 - 1 = 767$ と比較して大幅に少ない．

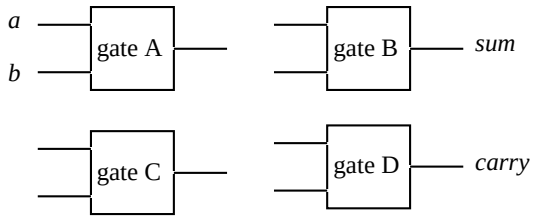

図 6 全加算器の最小性の証明
Fig. 6 Proof of minimality of full adder.

2. 加算器とそのテスト集合

2.1 全 加 算 器

2.1.1 最小の回路構成

まず，図 1 と図 2 に示す 5 つの 2 入力ゲートで構成された 2 種類の全加算器を考える．

定理 2.1 2 入力ゲートだけで構成された全加算器を考えるとき，図 1 と図 2 に示す回路はゲート数最小である．

（証明）4 つ以下の 2 入力ゲートで全加算器を構成できないことを示す．回路出力として sum と carry が必要なので，全加算器は**図 6** のような 4 つの 2 入力ゲートを組み合わせた回路となる．この回路に対して，すべての可能なゲートの種類とゲート間の配線を考えたとしても全加算器は得られないので，2 入力ゲートで全加算器を構成するには少なくとも 5 ゲート必要である．よって，図 1 に示す全加算器はゲート数最小である． （証明終）

図 1 の全加算器は，AND と EXOR ゲートだけの図 2 の構成のものよりよく用いられる[9]．図 1 と図 2 のいずれの回路も EXOR ゲートを使用しているが☆，EXOR ゲートをまったく使用しない場合にはゲートが 6 個以上必要となる．たとえば，8 つの 2 入力ゲートを使った全加算器の構成法が[12]に示されている．次項では，図 1 と図 2 の全加算器のテストパターン数について考察する．

2.1.2 単一縮退故障に対するテスト集合

単一縮退故障の仮定☆☆の下では，図 1 と図 2 の回路とも冗長故障は存在せず，すべての故障は**表 1** と**表 2** に示すテスト集合でそれぞれ検出できる．ただし，表 1 において，$x, y \in \{0, 1\}$ である．

表 1 全加算器（図 1）のテスト集合
Table 1 Tests for the full adder in FIg. 1.

	In			Out	
$test$	a	b	c_{in}	s	c_{out}
t_1	0	0	1	1	0
t_2	0	1	0	1	0
t_3	1	0	0	1	0
t_4	1	1	x	x	1
t_5	y	$\bar{y}$	1	0	1

表 2 全加算器（図 2）のテスト集合
Table 2 Tests for the full adder in FIg. 2.

	In			Out	
$test$	a	b	c_{in}	s	c_{out}
t_1	0	1	1	0	1
t_2	1	0	0	1	0
t_3	1	1	1	1	1

表 3 全加算器（図 1）に対する必要割当
Table 3 Necessary Assignments for the full adder in Fig. 1.

$fault$	a	b	c_{in}	g_3	g_4	g_5
g_1_$g_3/1$	0	0	1	0	0	0
a_$g_4/1$	0	1	0	0	0	0
b_$g_4/1$	1	0	0	0	0	0
$g_4/1$	1	1	x	0	1	1
$g_3/1$	y	$\bar{y}$	1	1	0	1

定理 2.2 表 1 と表 2 に示すテスト集合は，それぞれ図 1 と図 2 の回路の単一縮退故障を検出する最小のテスト集合である．

（証明）図 1 の回路に対して，故障集合 $\{g_1$_$g_3/1, a$_$g_4/1, b$_$g_4/1, g_4/1, g_3/1\}$ を考える．ここで，g_1_$g_3/1$ はゲート g_1 から g_3 への分岐信号線における 1 縮退故障を，$g_4/1$ はゲート g_4 の出力信号線の 1 縮退故障を意味する．**表 3** は，これら 5 つの故障のそれぞれに対して，その故障を検出するために必要な信号値割当（necessary assignment [1]）を示す．表 3 からどの 2 つの故障も同時に検出できないことは明らかで，この 5 つの故障を検出するには少なくとも 5 つのテストパターンが必要となる．表 1 のテスト集合ですべての単一縮退故障を検出できるので，このテスト集合は最小である．図 2 の回路に対しては，2 入力ゲートが含まれるので少なくとも 3 つのテストパターンが必要である．表 2 のテスト集合はすべての単一縮退故障を検出するので，このテスト集合も最小である．（証明終）

図 2 の回路は，最小のテスト集合で単一縮退故障がテストできる最小の全加算器である．他の全加算器で

☆ EXOR ゲートが直接実現できるテクノロジーでは，EXOR ゲートによる性能低下はないと思われる．しかし，CMOS ゲートアレイの場合，2 入力 NAND ゲートと比較して面積は 1.5 倍，遅延時間は 3 倍程度になる例もある．

☆☆ 縮退故障モデルは，ゲートレベルの回路で仮定されるため，特定のテクノロジーで実現した回路では，実際の故障動作を反映できない．特に EXOR ゲートではその乖離が大きくなると考えられ，たとえば，CMOS でトランスファーゲートを用いる場合，縮退故障モデルでトランジスタの故障を表現することは，困難である[17]．

は，4パターン以上のテスト集合が必要となる．

2.2 桁上げ先見加算器

2.2.1 回路構成

桁上げ先見加算器についても，多くの実現方法が存在する．それらの中で，図4に示した構成[14]が一般的に知られている．本論文では，縮退故障のテストに必要なテストパターン数に着目して，さらに2つの実現方法を考える．最初に，外部入力に直接接続するORゲート（$a_i \vee b_i$ を実現するゲート）をすべてEXORゲートに置き換えた回路を考える．その回路も加算器として動作する[16]．次に，図5に示す回路を考える．これは，図4の回路に含まれるすべてのORゲートを2入力EXORゲートに置き換えて得られる回路であり，ANDゲートとEXORゲートのみで構成されている．この回路も以下で示すとおり加算器として動作する．

定理 2.3 図5の回路は，加算器として動作する．
（証明は，付録Aに添付）

上記のようなEXORゲートへの置換のほかに，外部入力に直接接続するORゲートはそのままにして，それ以外のORゲート（図4の破線の大きな四角で囲まれた部分）のみを2入力EXORゲートに置き換えた回路も考えられるが，そのような変換を行った回路は加算器として動作しない．

2.2.2 単一縮退故障に対するテスト集合

桁上げ先見加算器に対して，テスト圧縮機能を用いたテストパターン生成[10]を行い，極小テスト集合の生成と極大独立故障集合の生成を行った．図4の回路に対しては，独立故障集合の大きさは10で，生成されたテストパターンの数も10であった（**表4**）．したがって，得られたテスト集合は，図4の桁上げ先見加算器に対する最小テスト集合である．図5の回路に対しては，独立故障集合の大きさは12で，生成されたテスト集合の大きさも12である（**表5**）．したがって，このテスト集合も，図5の回路に対する最小テスト集合である．図5の回路の最小テスト集合が図4の回路のものより大きい原因の1つに，図5の回路は多入力EXORゲートを用いずに，それを2入力EXORゲートの木構造を用いて実現したことが考えられる．たとえば，2入力EXORゲートの縮退故障のテストには3パターン必要であるが，3入力EXORゲートであれば2パターンで十分である．このように，3入力以上のEXORゲートを用いれば，図5の回路のテストパターン数を12より少なくできる可能性がある．ただし，実際に生じる欠陥を検出する能力は，テストパターンが少なくなる分低下することも考慮が必要である．

表4 桁上げ先見加算器（図4）のテスト集合
Table 4 Tests for the carry look-ahead adder in FIg. 4.

	In									Out
$test$	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	c_0	c_4
t_1	0	1	1	1	0	0	0	0	1	0
t_2	1	1	1	0	1	1	1	0	1	0
t_3	1	0	0	0	1	0	1	1	1	0
t_4	1	1	0	1	0	1	1	0	0	1
t_5	1	1	1	0	1	0	0	1	0	1
t_6	0	0	0	0	1	1	1	1	1	1
t_7	1	1	0	1	1	1	0	0	1	0
t_8	0	1	1	0	1	0	1	1	0	1
t_9	1	0	0	1	0	0	0	1	1	1
t_{10}	0	0	0	0	1	1	1	1	0	0

表5 桁上げ先見加算器（図5）のテスト集合
Table 5 Tests for the carry look-ahead adder in FIg. 5.

	In									Out
$test$	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	c_0	c_4
t_1	1	1	1	1	0	0	0	0	1	1
t_2	1	0	0	0	1	1	1	1	1	1
t_3	0	1	0	0	1	0	1	1	0	0
t_4	0	1	0	0	1	0	1	0	1	0
t_5	1	1	0	0	1	1	1	1	0	1
t_6	1	0	0	1	1	1	0	0	0	0
t_7	0	1	1	0	1	1	0	0	0	0
t_8	1	1	0	0	1	1	0	1	0	0
t_9	1	1	1	0	0	0	1	0	0	0
t_{10}	1	0	1	0	0	0	0	1	1	0
t_{11}	1	0	1	0	0	1	1	1	1	1
t_{12}	1	0	1	1	1	1	0	1	0	1

ある．

3. 直列接続した加算器の最小テスト集合

3.1 順次桁上げ加算器

3.1.1 単一縮退故障に対するテスト集合

図3の順次桁上げ加算器は，全加算器のセルとし，全加算器の出力 c_{out} と入力 c_{in} を接続したアレイである．ここでは，まず，順次桁上げ加算器の単一縮退故障に対するテスト集合について考察する．アレイとなる順次桁上げ加算器のテスト集合を全加算器のテスト集合に基づいて構成するには，次の条件を満足すればよい．

条件1）各セルに全加算器のテストパターンをすべて印加できること．

条件2）各セルの出力に伝搬してきた故障の影響を観測可能であること．

条件1については，セルの入力 a_i，b_i，c_{i-1} のうち a_i と b_i は外部入力なので，c_{i-1} に必要な値を設定できることを示せばよい．これは次の定理3.4で示す．また条件2については，セルの出力 s_i，c_i のうち s_i

は外部出力なので，c_i に伝搬してきた故障の影響を他のセルの外部出力で観測できることを示せばよい．これは，定理 3.5 で示す．

定理 3.4 図 3 の順次桁上げ加算器を構成する各セル（全加算器）に対して，表 1 および表 2 に示す全加算器のテスト集合を印加できる．

（証明）第 1 番目のセル（a_1 と b_1 を入力とするセル）に対しては，定理は成り立つ．第 i 番目のセルに対しては $(i \geq 2)$，第 i 番目のセルの入力 c_{i-1} の論理値を，第 $i-1$ 番目のセルの入力 a_{i-1}，b_{i-1} により自由に制御できる．よって定理は成り立つ．（証明終）

定理 3.5 全加算器のテスト集合が順次桁上げ加算器の各セルに印加される場合，各セルの故障は外部出力で検出可能である．

（証明）最終段のセル（s_n と c_n を出力とするセル）に対しては，定理は成り立つ．全加算器のテストパターンを第 i 番目のセル $(i < n)$ に印加した場合を考える．s_i に故障の影響が伝搬したなら，そこで検出可能である．また，c_i に伝搬した故障の影響は，第 $i+1$ 番目のセルの出力 s_{i+1} で入力 a_{i+1}，b_{i+1} の値にかかわらず検出できる．よって定理は成り立つ．（証明終）

定理 3.4 と定理 3.5 では，順次桁上げ加算器の各セルに全加算器のテスト集合を印加でき，そのセルの故障が検出できることを述べた．次は，アレイとしてのテストパターンを考える．第 i 番目のセルの出力 c_i は，第 $i+1$ 番目のセルの入力でもある．したがって，第 i 番目のセルに印加したテストパターンに対する出力値が，第 $i+1$ 番目のセルの c_i におけるテストパターンと一致していれば，異なるセルを同時にテスト可能になる．まず，図 1 の全加算器で構成されるアレイについて考える．この全加算器に対するテスト集合は表 1 に示したが，このうち t_2，t_3，t_5 については，c_{in} の値と c_{out} の値が同じなので，各セルに同時にテストパターンを印加できる．しかしながら，t_1 については，c_{in} の値と c_{out} の値が異なるので，第 i 番目のセルと第 $i+1$ 番目のセルには同時に t_1 を印加できない．そこで，t_4 において，x の値を 0 に決定する．これにより，第 i 番目のセルに t_1 を印加したときには第 $i+1$ 番目のセルに t_4 を印加でき，逆に第 i 番目のセルに t_4 を印加したときには第 $i+1$ 番目のセルに t_1 を印加できる．このように，表 1 のテスト集合の t_4 における x の値を 0 にすることで，図 1 の全加算器に基づく順次桁上げ加算器は 5 つのテストパターンでテスト可能であり，このテスト集合は最小である．表 1 のテスト集合には t_5 にドントケア y が残っているが，y の値を 0 としたとき，順次桁上げ加算器全体のテスト集合は**表 6** のようになる．同様にして，図 2 の全加算器に基づく順次桁上げ加算器のテスト集合を考える．表 2 の全加算器のテスト集合において，c_{in} の値と c_{out} の値はすべて同じなので，すべてのセルに同じテストパターンを同時に印加できる．したがって，図 2 の全加算器に基づく順次桁上げ加算器のテスト集合は 3 パターンとなり，これはこの回路（アレイ）における最小のテスト集合である．

表 6 順次桁上げ加算器の最小テスト集合
Table 6 Minimum tests for the ripple carry adder.

$test$	a_1	b_1	c_0	a_2	b_2	a_3	b_3	a_4	b_4	・	・	・
T_1	0	0	1	1	1	0	0	1	1	・	・	・
T_2	0	1	0	0	1	0	1	0	1	・	・	・
T_3	1	0	0	1	0	1	0	1	0	・	・	・
T_4	1	1	0	0	0	1	1	0	0	・	・	・
T_5	0	1	1	0	1	0	1	0	1	・	・	・

3.1.2 多重縮退故障に対するテスト集合

本節では，順次桁上げ加算器の多重縮退故障に対するテスト集合について考察する．多重縮退故障を仮定するときは，縮退故障を回路内の分岐枝の信号線と外部入力にだけ考えればよく[6]．故障を仮定するこれらの信号線をチェックポイントという，すべての多重縮退故障はチェックポイントにおける縮退故障の組合せで表現できる．図 1 と図 2 の全加算器のどちらも 11 個のチェックポイントを持ち，それらは，外部入力 a，b，c_{in} とゲート g_1，g_2，g_3，g_4 の分岐枝である．順次桁上げ加算器において多重縮退故障を扱う場合，各セルのチェックポイントの故障を仮定することになる．順次桁上げ加算器の多重故障を考える前に，まず全加算器における多重故障を考える．全加算器における多重故障は，表 1 または表 2 の単一故障と同じテスト集合で検出できる．たとえば，入力 a，b，c_{in} から出力 s までの経路上に存在するチェックポイントの故障を含む多重故障は出力 s で検出できる．また，ゲート g_3，g_4 のファンインの故障を含む多重故障は出力 c_{out} で検出できる．このように全加算器では，単一縮退故障のテスト集合がそのまま多重故障のテスト集合となる．しかしながら，順次桁上げ加算器では，表 6 の単一縮退故障のテスト集合では検出できない多重故障が存在する．たとえば，多重故障 $(a_2/1, b_2/0, g_1\_g_3/1, a_1\_g_4/0, b_1\_g_4/1)$ は，表 6 のテスト集合で検出できない．表 6 のテスト集合を構成するために，表 1 のドントケア y の値に 0 を設定したが，仮にこの値を 1 としてもこの多重故障は検出できない．したがって，順次桁上げ加算器の多重故障は 5 つのテストパターンではすべて検出できないことにな

る．表6のテスト集合に1つのテストパターンをつけ加え，**表7**のようなテスト集合とすることで，多重故障をすべて検出できるようになる．これは，[8]の故障解析手法の考え方を用いて証明できる．

定理 3.6 表7に示す6つのテストパターンは，図1の全加算器を用いた順次桁上げ加算器の多重故障をすべて検出する．

（証明は，付録Bに添付）

5つのテストパターンではすべての多重故障を検出できないことから，表7のテスト集合は最小である．また，同じテスト集合で図2の全加算器を用いた順次桁上げ加算器の多重故障もすべて検出できるが，このテスト集合が回路にとって最小か否かは明らかでない．

3.2 $4m$ ビット桁上げ先見加算器の単一縮退故障に対するテスト集合

図4と図5の4ビット桁上げ先見加算器を各セルとして直列に m 個接続することで，$4m$ ビットの桁上げ先見加算器を実現できる．本節では，$4m$ ビット桁上げ先見加算器の単一縮退故障に対するテスト集合について述べる．まず，図4の桁上げ先見加算器をセルとするアレイを考える．2.2.2 項で述べたように，この桁上げ先見加算器に対する最小テスト集合の大きさは10である．このテスト集合を $4m$ ビット桁上げ先見加算器の各セルに適用する場合を考える．定理 3.4 と定理 3.5 で述べたのと同様にして，各セルに4ビット桁上げ先見加算器のテストパターンを印加可能で，各セルの出力で検出される故障は，全体の加算器でも検出できる．ここで問題となるのは，異なるセルに同時にテストパターンを印加できるかである．同時に印加可能にするためには，第 i 番目のセルに印加したテストパターンに対する c_{out} の出力値が，第 $i+1$ 番目のセルの c_{in} におけるテストパターンと一致していなければならない．つまり，c_{in} において論理値1となるテストパターンの数と c_{out} において論理値1となるテストパターンの数が同数でなければならない．表4のテスト集合では，c_{in} において論理値1となるテストパターンは6個（t_1，t_2，t_3，t_6，t_7 と t_9）であり，c_{out} において論理値1となるテストパターンは5個（t_4，t_5，t_6，t_8 と t_9）である．したがって，すべてのセルにこれらのテストパターンを10パターンで印加することはできない．そこで，c_{in} において論理値0となり，c_{out} において論理値1となるテストパターンを1つ追加する．そうすれば，c_{in} と c_{out} における論理値1の数が同数となり，表4のテスト集合を各セルに印加可能となる．結果として，たとえば**表8**に示すような11個のテストパターンが，$4m$ ビット桁上げ先見加算器の単一縮退故障に対するテスト集合となる．このテスト集合が最小であるか否かは不明である．次に図5の桁上げ先見加算器をセルとするアレイを考える．この桁上げ先見加算器に対する最小テスト集合は表5に示すようなものである．このテスト集合の c_{in} と c_{out} における論理値1の数は同数であるため，12パターンで $4m$ ビット桁上げ先見加算器のテスト集合を作成でき，これは回路に対して最小のテスト集合となる．

表7 順次桁上げ加算器の多重故障用最小テスト集合
Table 7 Minimum tests for multiple faults of the ripple carry adder.

$test$	a_1	b_1	c_0	a_2	b_2	a_3	b_3	a_4	b_4	・	・	・
T_1	0	0	1	1	1	0	0	1	1	・	・	・
T_2	0	1	0	0	1	0	1	0	1	・	・	・
T_3	1	0	0	1	0	1	0	1	0	・	・	・
T_4	1	1	0	0	0	1	1	0	0	・	・	・
T_5	0	1	1	0	1	0	1	0	1	・	・	・
T_6	1	1	1	1	1	1	1	1	1	・	・	・

4. ま と め

本論文では，順次桁上げ加算器と直列接続の桁上げ先見加算器の最小テスト集合について述べた．順次桁

表8 $4m$ ビット桁上げ先見加算器のテスト集合
Table 8 Tests for the 4m-bit carry look-ahead adder.

$test$	a_1	a_2	a_3	a_4	b_1	b_2	b_3	b_4	c_0	a_5	a_6	a_7	a_8	b_5	b_6	b_7	b_8
T_1	0	1	1	1	0	0	0	0	1	1	1	0	1	0	1	1	0
T_2	1	1	1	0	1	1	1	0	1	1	1	1	0	1	0	0	1
T_3	1	0	0	0	1	0	1	1	1	0	1	1	0	1	0	1	1
T_4	1	1	0	1	0	1	1	0	0	0	1	1	1	0	0	0	0
T_5	1	1	1	0	1	0	0	1	0	1	1	1	0	1	1	1	0
T_6	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1
T_7	1	1	0	1	1	1	0	0	1	0	1	1	0	1	0	1	1
T_8	0	1	1	0	1	0	1	1	0	1	0	0	0	1	0	1	1
T_9	1	0	0	1	0	0	0	1	1	1	0	0	1	0	0	0	1
T_{10}	0	0	0	0	1	1	1	1	0	0	0	0	0	1	1	1	1
T_{11}	0	1	1	0	1	0	1	1	0	1	1	0	1	1	1	0	0

上げ加算器については，3パターンで単一縮退故障がテストできる構成法を示した．これは，これまでに知られているどの加算器よりも小さいテスト集合でテストできるものであり，EXORゲートをうまく利用したことによる効果である．したがって，縮退故障に関しては，EXORゲートの利用はテストパターン数の削減に効果があると期待できる．また，4ビット桁上げ先見加算器の最小テスト集合の考察結果より，n ビットの順次桁上げ加算器と単一および多重縮退故障の最小テスト集合を示した．本論文では述べなかったが，冗長2進加算器についても本論文と同様のアプローチで単一縮退故障に対する最小のテスト集合が求められているが，これはまた別の機会に報告する．

参 考 文 献

1) Akers, S.B. and Krishnamurthy, B.: On the Application of Test Counting to VLSI Testing, Technical Report, No.CR-85-12, Computer Research Laboratory, Tektronix Laboratories (1985).
2) Batek, M.J., and Hayes, J.P.: Test-set Preserving Logic Transformations, *Proc. 29th ACM/IEEE Design Automation Conference*, pp.454–458 (1992).
3) Becker, B.: Efficient Testing of Optimal Time Adders, *IEEE Trans. Comput.*, Vol.37, No.9, pp.1113–21 (1988).
4) Becker, B., Drechsler and Molitor, P.R.: On the Generation of Area-time Optimal Testable Adders, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, Vol.14, No.9, pp.1049–66 (1995).
5) Becker, B., Drechsler, R., Krieger, R. and Reddy, S.M.: A Fast Optimal Robust Path Delay Fault Testable Adder, *European Design and Test Conference ED&TC 96*, pp.491–498 (1996).
6) Bossen, D.C. and Hong, S.J.: Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks, *IEEE Trans. Comp.*, Vol.C-20, pp.1252–1257 (1971).
7) Cheng, W.-T. and Patel, J.H.: A Minimum Test set for Multiple Fault Detection in Ripple carry Adders, *IEEE Trans. Comput.*, Vol.C-36, No.7, p.891-5 (1987).
8) Cox, H. and Rajski, J.: A Method of Fault Analysis for Test Generation and Fault Diagnosis, *IEEE Trans. CAD.*, Vol.7, pp.813–833 (1988).
9) Hill, F.J. and Peterson, G.R.: *Introduction to Switching Theory and Logical Design, Third Edition*, p.180, John Wiley & Sons (1981).
10) Kajihara, S., Pomeranz, I., Kinoshita K. and Reddy, S.M.: Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.14, No.12, pp.1496–1504 (1995).
11) Kautz, W.H.: Testing for Faults in Cellular Logic Arrays, *8th Annu. Sympo. Switching and Automata Theory*, pp.161–174 (1967).
12) Kohavi, Z.: *Switching and Automata Theory*, p.116, McGraw-Hill (1978).
13) Liu, T-K., Hohlin, K.R., Shiau, L-E. and Muroga, S.: Optimal One-bit Full Adders with Different Types of Gates, *IEEE Trans. Comput.*, Vol.C-23, No.1, pp.63–70 (1974).
14) Muroga, S.: *Logic Design and Switching Theory*, pp.518–520, Wiley-Interscience Publication (1979).
15) 高木直史：加算回路のアルゴリズム—桁上げ伝搬との戦い，情報処理，Vol.37, No.1, pp.80–85 (1996).
16) Tinder, R.F.: *Digital Engineering Design*, p.274, Prentice-Hall (1991).
17) 梶原誠司，板崎徳禎，樹下行三：組合せ回路のスタックオープン故障に対するテストパターン生成について，電子情報通信学会論文誌（D-I），Vol.J73-D-I, No.12, pp.997–1005 (1990).

付録 A（定理 2.3 の証明）

s_i と c_i をそれぞれ第 i 番目のセルの出力とし，c_0 を桁上げ入力とする．加算の定義から以下の関係が成り立つ：

$$s_i = (a_i \oplus b_i) \oplus c_{i-1}$$
$$c_i = a_i b_i \vee (a_i \vee b_i) c_{i-1}$$
$$= a_i b_i \oplus (a_i \oplus b_i) c_{i-1}$$

c_i の式において $\vee$ 演算は $\oplus$ 演算に置き換えることができる．ここで，

$$E_i = a_i \oplus b_i,$$
$$G_i = a_i b_i$$

とする．E_i と G_i を用いれば，s_i と c_i は以下のように書ける：

$$s_i = E_i \oplus c_{i-1},$$
$$c_i = G_i \oplus E_i c_{i-1}.$$

以上から次の式が成り立つ：

$$s_1 = E_1 \oplus c_0,$$
$$c_1 = G_1 \oplus E_1 c_0,$$
$$s_2 = E_2 \oplus c_1$$
$$= E_2 \oplus G_1 \oplus E_1 c_0,$$

表 9 テストパターン（表 5）に対して取りうる値
Table 9 Possible values for the tests in Table 5.

a_1	b_1	c_0	$a_1 \oplus b_1$	a_2	b_2	g_3	g_4	g_5	g_1	g_2
001101	010111	100011	011010	101001	110011	000101	000010	000111	011010	011101
000000	000000		111111	000000	000000	000000	100011	000010	101001	·
111111	111111			111111	111111	010111		100011	110011	·
						001101		010111	010110	·
								110111	001100	
								001111	000000	
								101111	111111	

表 10 正当化できないものを削除した値
Table 10 Possible values after removing unjustifiable ones.

a_1	b_1	c_0	$a_1 \oplus b_1$	a_2	b_2	g_3	g_4	g_5	g_1	g_2
001101	010111	100011	011010	101001	110011	000101	000010	000111	011010	011101
000000	000000		111111	000000	000000	000000	100011	000010	101001	
111111	111111			111111	111111	010111		100011	110011	
						001101		010111	010110	
								110111	001100	
								001111	000000	
								101111	111111	

$$
\begin{aligned}
c_2 &= G_2 \oplus E_2 c_1 \\
&= G_2 \oplus E_2(G_1 \oplus E_1 c_0) \\
&= G_2 \oplus E_2 G_1 \oplus E_2 E_1 c_0, \\
s_3 &= E_3 \oplus c_2 \\
&= E_3 \oplus G_2 \oplus E_2 c_1 \\
&= E_3 \oplus G_2 \oplus E_2(G_1 \oplus E_1 c_0) \\
&= E_3 \oplus G_2 \oplus E_2 G_1 \oplus E_2 E_1 c_0, \\
c_3 &= G_3 \oplus E_3 c_2 \\
&= G_3 \oplus E_3(G_2 \oplus E_2 G_1 \oplus E_2 E_1 c_0) \\
&= G_3 \oplus E_3 G_2 \oplus E_3 E_2 G_1 \oplus E_3 E_2 E_1 c_0, \\
s_4 &= E_4 \oplus c_3 \\
&= E_4 \oplus G_3 \oplus E_3 G_2 \oplus E_3 E_2 G_1 \oplus E_3 E_2 E_1 c_0, \\
c_4 &= G_4 \oplus E_4 c_3 \\
&= G_4 \oplus E_4(G_3 \oplus E_3 G_2 \oplus E_3 E_2 G_1 \oplus \\
&\quad E_3 E_2 E_1 c_0) \\
&= G_4 \oplus E_4 G_3 \oplus E_4 E_3 G_2 \oplus E_4 E_3 E_2 G_1 \oplus \\
&\quad E_4 E_3 E_2 E_1 c_0.
\end{aligned}
$$

これは図 4 の OR ゲートを図 5 のように EXOR ゲートに置き換え可能なことを意味する． （証明終）

付録 B（定理 3.6 の証明）

まず，入力 a_1，b_1，c_0 から出力 s_1 までの経路上に存在するチェックポイント上の縮退故障を考える．表 7 のテストパターン (T_5, T_6) がチェックポイント a_1，a_1_g_1，g_1_g_2 の 0 縮退故障と 1 縮退故障を含む多重故障を検出する．これは，もしこれらのチェックポイント上に故障が存在するならば，外部出力 s_1 に正常時の論理値 $(0, 1)$ を出力できないことから，明らかである．同様にして，テストパターン (T_3, T_4) がチェックポイント b_1，b_1_g_1，c_0，c_0_g_2 の 0 縮退故障と 1 縮退故障を含む多重故障を検出する．したがって，表 7 のテストパターンは少なくとも入力 a_1，b_1，c_0 と出力 s_1 の間にあるチェックポイント上の故障は検出できる．次に，最初の段のセルの残りのチェックポイントの故障の検出のために，**図 7** に示す．部分回路を考える．もし，この部分回路のすべての多重故障が表 7 のテストパターンで検出可能なら，最初の段のセルの多重故障はすべて検出可能となる．テストパターン (T_2, T_5) で，チェックポイントの故障 $c_0/0$，$c_0/1$，$(a_1 \oplus b_1)/0$ が検出できる．ここで，表 7 のテストパターンに対して，残りのチェックポイントの故障を考慮した場合に正常回路と故障回路のどちらかでとる可能性のある各内部信号線の論理値を計算する（**表 9**）．これは，文献 8) の故障解析手法における前方伝搬操作と同様の操作である．たとえば，テストパターン $(T_1, T_2, T_3, T_4, T_5, T_6)$ に対して，a_0 における論理値は 001101，000000，または 111111 のいずれかになる．また，g_4 における論理値は，a_0 と b_0 においてと

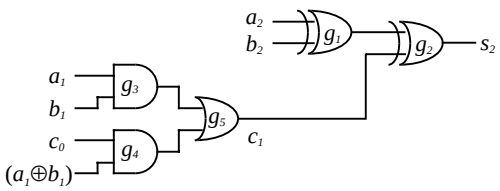

図 7 順次桁上げ加算器の部分回路
Fig. 7 Sub-circuit of the ripple carry adder.

りうる論理値のすべての組合せに対して，AND 演算を行って得られる．ここで，正常な論理回路の s_2 における出力は 011101 をとらなければならないことを利用して，表 9 の値の中から，s_2 で正常出力をもたらすことができない値を取り除く．これは，文献 8) の故障解析手法における後方含意操作と同様の操作である．表 9 のとりえない信号値を下線を付して**表 10** に示すが，結果として，すべてのチェックポイントにおいて，表 7 のテストパターンに対して 000000 および 111111 をとる可能性がないことが分かる．したがって，最初の段のセルの多重故障はすべて検出される．2 段目以降のセルの多重故障に対しては，定理 3.4 と定理 3.5 と同様の考え方により，再帰的に検出可能なことを証明でき，定理は成立する． （証明終）

(平成 12 年 9 月 26 日受付)
(平成 13 年 2 月 1 日採録)

梶原 誠司（正会員）

1965 年生．1987 年広島大学総合科学部総合科学科卒業．1989 年広島大学大学院工学研究科情報工学専攻博士課程前期修了．1992 年大阪大学大学院工学研究科応用物理学専攻博士後期課程修了．博士（工学）．同年大阪大学工学部応用物理学科助手．1996 年九州工業大学情報工学部助教授．2001 年同大学マイクロ化総合技術センター助教授（兼任），現在に至る．1997 年〜1998 年大阪大学大学院工学研究科助教授（併任）．論理回路のテスト生成，テスト容易化設計等の研究に従事．電子情報通信学会，IEEE 各会員．

笹尾 勤（正会員）

昭和 25 年生．昭和 47 年大阪大学工学部電子工学科卒業．昭和 52 年同大学院博士課程修了．同年同大学工学部助手．昭和 63 年九州工業大学情報工学部助教授，平成 5 年同大学情報工学部教授，平成 12 年同大学マイクロ化総合技術センターセンター長併任，現在に至る．工学博士．この間，昭和 57 年 2 月より 1 年間 IBM ワトソン研究所客員研究員，平成 2 年 1 月より 3 カ月米国海軍大学院教授．昭和 54 年丹羽記念賞，昭和 62 年および平成 7 年 ISMVL 論文賞．論理設計，スイッチング理論，多値論理等の研究に従事．著書「論理設計：スイッチング回路理論」（近代科学社），Logic Syntheis and Optimization (1993)，Representations of Discrete Functions (1996)，Switching Thoery for Logic Synthesis (1999)（Kluwer）等．電子情報通信学会，IEEE フェロー．