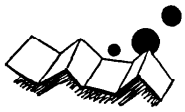
解　説



高信頼化技術の展望†

当　麻　喜　弘††

1. はじめに

コンピュータのみならず，いずれのシステムにおいてもその信頼性を高めることは，いつの時代でも常に要請されている技術的課題であるが，最近は特にこの点に関心が高まっているように思われる．本文ではまず，その背景，最近，高信頼化技術の概要，将来の問題などを述べ，以後の各論の導入的役割を果たしたい．

2. 背　景

コンピュータは，宇宙用，公共社会用，軍用といった特殊ミッションに用いられる場合と，一般の商業ベースの目的に用いられる場合がある．それぞれにおいて高信頼化技術が重要視されるに至った背景をみてみよう．

2.1 特殊ミッションの場合

ミッションの特殊性のため，信頼性に対する要求水準がきわめて高い．以下に若干の例を示そう．

- 木星，土星の探測でよく知られるボイジャーのミッションタイムは 10 年以上である．この間当然のことながら保守は行われない．
- NASA が開発を進めている（実際は SRI と Draper 研究所に依託）航空機制御用のコンピュータでは，10 時間の飛行における故障の発生頻度が 10^{-9} 回以下と定められている．フィットでいえば 0.1 フィット以下である*.
- 航空交通管制用コンピュータは，ダウン後 30 秒以内で自動的に回復するよう要求されている．
- ミニットマンⅢミサイルの故障率は約 4,200 フィットである．この逆数をとると 27 年に相当する．

これらがいかに厳しいものであるかは次のデータをみればすぐ分る．

- MOSメモリの 1 ビット当りの故障率は 0.01 フィット程度である．これを単純に集計しただけでも IBM のメモリモジュールの故障率は約 10^5 フィット，その逆数は約 500 日である．
- かつて信頼性が高いということで注目されたワイヤメモリ（4 k 語×18 ビット）の 4 年後の信頼度は 0.54 であるという報告がある．
- IBM 370/168 ユーザの経験では平均故障間隔時間 (MTBF) が 30〜35 日，平均修理時間 (MTTR) は 1.2〜1.5 時間であった．
- CARY-1 の初期の MTBF は 4 時間，MTTR は 0.41 時間であったという．

したがって，特別な対策を施さない限り前記のような厳しい要求を満たすことはできない．こうして高信頼化技術の重要性が深く認識されるようになった．

2.2 商業ベースの場合

（1） 経 済 性

Eastern Air Lines では予約システムのコンピュータが 1 分間ダウンしただけで 20,000 ドルの損害であるという．また Merrill Linch, Pierce, Fenner & Smith Inc. は，20 分間のコンピュータのダウンで 50,000 ドルにのぼる農作物の取引が失われると見積っている．CRAY-1 のアンアベイラビリティ* を 0.093 とし，システムの購入価格を 20 億円とすれば，約 2 億円が無駄に失われることになる．

このようにコンピュータの信頼性は企業の経済に直接影響するようになってきている．

（2） 社 会 化

企業活動の規模が大きくなり社会化すると，一社のコンピュータのダウンが社会的混乱をひき起す（オンラインバンキングシステムなどその好例である）．したがってコンピュータのダウンは厳しい指弾を受ける

† Introductory Review on Reliable Computing and Fault Tolerance by Yoshihiro TOHMA (Department of Computer Science, Tokyo Institute of Technology).

†† 東京工業大学情報工学科

* フィットは故障率の単位で，10^9 個・時間に 1 つの場合で生じる場合を 1 フィットとする．

* アベイラビリティを 1 から引いた値．

ようになった.

(3) ＬＳＩ化

コンピュータの構成要素が LSI 化すると，故障もしくは欠陥が生じたときの取替単位が大きくなる.

LSI 技術の進歩によって1ビットもしくはゲート当りの故障率はこの数年急激に減少したが，最近この傾向が頭打ちの状態になりつつあるといわれる．集積度の上昇とともに，LSI の中に何らかの高信頼化技術を組み込まねばならないという認識が深まりつつある.

(4) 保　守

若干観点は異なるが，保守の問題が重大化していることをあげておきたい．一般的に人件費が上昇している中で，保守作業の高度化に伴う教育費その他の間接費も増加している．保守コストの低減という意味からも，高信頼化技術が期待されている.

2.3 諸活動・諸機関

以上のような事情を反映して，高信頼化技術に関する会議/シンポジウム，ワークショップなどが頻繁に催されるようになった．またそれらをサポートするいろいろな機関も新しく生まれている.

表-1 年次 FTCS

年	会　議	場　所	ホスト
1971	FTCS-1	Pasadena	JPL
1972	FTCS-2	Newton	Draper Lab.
1973	FTCS-3	Palo Alto	Stanford Univ.
1974	FTCS-4	Urbana	U. Illinois
1975	FTCS-5	Paris	FNIE
1976	FTCS-6	Pittsburgh	CMU
1977	FTCS-7	Los Angeles	USC
1978	FTCS-8	Toulouse	LAAS
1979	FTCS-9	Madison	U. Wisconsin
1980	FTCS-10	Kyoto	
1981	FTCS-11	Portland	IBM
1982	FTCS-12	Santa Monica	UCLA
1983	FTCS-13	Milan or Florence	

最初に組織的な会議が開催されたのは 1962 年の Symposium on Redundancy Technique for Computing Systems (ワシントン) といってよいであろう．その後**表-1** に示すように International Symposium on Fault-Tolerant Computing (以下 FTCS と略す) が 1971 年より毎年開催されるようになった．1980 年に多くの方々のご尽力によりわが国で FTCS-10 が成功裡に開催されたことは記憶に新しい.

FTCS に関する研究活動はアメリカが最も活発で，次いでフランス，日本，イタリア，ドイツ，イギリス，その他となるが，最近は FTCS に 20 カ国内外の参加がある．またワークショップも**表-2** に示すように盛んで，アメリカ，ヨーロッパの各地で開催されている.

1970 年，IEEE Computer Society 内に Fault-Tolerant Computing 技術委員会が設けられた．また 1980 年には IFIP TC-10 内に Reliable Computing and Fault-Tolerance に関するワーキンググループ WG 10・4 の設置が認められた．また IFAC においても，制御用コンピュータの高信頼化が大きな話題となりつつある.

わが国では，1980 年の FTCS 以後，電子通信学会に FTCS 日本国内委員会がおかれている．同学会の電子計算機研究専門委員会，信頼性研究専門委員会は FTCS-10 のホスト委員会である．また，非公式なものであるが，同好の士が集まり，毎年2回程度研究討論会を行っている.

3. 耐故障技術

信頼性に関する厳しい要求を満たすアプローチを大

表-2 研究発表の会議 (予定も含む)

	会　議	場　所	年　月
非定期的	International Conference on Reliable Software	Los Angeles	1975 April
	IFIP Working Conference on Reliable Computing and Fault-Tolerance	London	1979 September
	Workshop on Fault-Tolerant VLSI Design	Santa Monica	1980 April
	Workshop on Validation of Fault-Tolerant Computers and Systems	Skyline Drive	1980 June
	Workshop on Design for Testability	Vail	1981 April
	IFIP WG 10.4	Portland	1981 June
	Workshop on Self-Diagnosis and Fault-Tolerance	Tubingen	1981 July
	Symposium on Reliability in Distributed Software and Database Systems	Pittsburgh	1981 July
	Workshop on the Reliability of Local Area Networks	South Padre Island	1982 February
	International Conference on Fault-Tolerant Systems and Diagnostics*	Katowice	1982 September
定期的	FTCS		
	Design Automation Conference		
	International Test Conference		

* 第2回目と思われるが記録を失ったので正確には分らない.

別すれば，構成要素の品質を高めて故障が生じないようにするもの (Fault Avoidance) と，構成要素は故障するものとはじめから予想しそれに耐える (Fault-Tolerance) 方法を考えておくものという2つがある．前者のアプローチはもちろん大切であり，従来からも長い努力が積み重ねられているが，特に最近新しい研究活動として活発になってきたのは後者のアプローチである．本文でも，以後，後者のアプローチ（耐故障技術とでも呼べばよいであろうか）について主に述べる．

耐故障技術といっても，耐の水準，対象とする故障の範囲などに関していろいろな考え方があるので，この範囲に入る技術は実にさまざまである．

3.1 耐の水準

目的とする処理が一切中断されることなく，外からその生起が全く見えないように故障をマスクしてしまう耐の水準は静的な耐故障といってよいであろう．これに対し，故障の発生によって目的の処理が何らかの意味で中断し，その後再び正しい処理が回復される耐の水準は動的な耐故障といってよいであろう．

正しい処理を行っている状態をアップ (Up) 状態，それ以外をダウン (Down) 状態ということにすれば，ある時間幅でのアップ状態の割合（すなわちアベイラビリティ）を大きくすることが動的耐故障の目的となる．割合を大きくすればよいのであるから，アップ状態の時間 (MTTF) を長くしてもよいし，ダウン状態の時間 (MTTR) を短くしてもよい．静的な耐故障は MTTF→∞ となるか，MTTR→0 となった極限の場合とみることができよう．

アップ状態の場合，とにかく完全無欠であればそれにこしたことはない．しかし，単に正しいか否かということだけではなく，アップ状態の内容をこまかに眺めそこに性能の尺度を持ち込んだものが Gracefully Degrading もしくは Fail-Soft システムである．アップ状態のとき，ある限度まで性能（パフォーマンス）が低下してもよいとする考え方でシステムを扱う．

これに対し，ダウン状態に注文をつけたものが Fail-Safe システムである．ダウン状態に陥ったとき，少なくとも危険側の出力を出さないことを保証するものが Fail-Safe システムである．Fail-Safe という考え方は鉄道関係の人々の間には早くからあるが，もっと一般に認識されてよいのではなかろうか．たとえばコンピュータネットワークを考えた場合，接続されたコンピュータの故障によってネットワーク全体が汚染されてしまうことがないように，ダウン状態についても何らかの配慮をしておくことが必要であろう．Ether ネットワークのような簡便なネットワークはこのような汚染にきわめて弱いように見える．

いずれにしても，コンピュータシステムにどのような耐の水準を持たせるかは，そのコンピュータシステムが置かれる環境に応じて定めるべき問題である．

3.2 故 障

1981年6月，先に述べた IFIP WG 10・4 の設立委員会で「故障 (Fault)」に関する議論が延々数時間にわたって行われた．常識的な物理故障のほかに，ハードウェアの設計誤り，ソフトウェアのバグ，なども包括して故障を定義したいというので議論がなかなか収まらなかったのである．

処理を正しく行うという目的達成を阻害するものが故障であるととらえ，さらに目的達成までのプロセスに心理的行為までを含めるならば，設計誤りやソフトウェアバグ，さらには操作ミスさえもまさに故障ということができよう．

さらに故障の性質に関しても，自然発生的か意図的か，間欠的か定常的かといったことが問題になる．

耐故障技術もそれぞれの故障の性質に応じていろいろ考案されている．

3.3 諸 技 術

さて，構成要素は故障するという前提に立つから何らかの意味の冗長性が不可欠である．Avizienis は，ハードウェアの冗長性，ソフトウェアの冗長性，時間の冗長性をあげている．以下，諸技術を概観するが，最近注目されている動向に特に重点をおいて述べる．

① ハードウェアの冗長構成

ハードウェアに冗長性を持ち込む場合，部品，機能回路/ユニット，モジュール，システム，などの各水準で行うことが考えられる．また冗長性の形態としては，単純な多重化と多数決原理の組合せ，多重化と再構成機構の組合せ，誤り訂正符号の利用，などがある．歴史的には部品水準の多重化，回路水準の多重化多数決が1960年代に行われたが，1970年代から最近にかけてはモジュール水準の多重化・再構成の形態が一般的になりつつある．誤り訂正符号の利用はメモリにおいて一般的になった．ディスクにはバースト誤り訂正符号であるファイヤ符号，また主メモリでは単一誤り訂正・二重誤り検出符号 (SEC/DED 符号) が用いられている．ごく最近，隣接誤り訂正符号が主メモリ（キャッシュ）に用いられ始めた[1]．

以上はいずれもシステムのアップ状態を長く保つためのものということができよう．

② セルフチェッキング

最近，ハードウェアコストの低下に伴い回路構成を単純に2重化してでもチェックを徹底的に行うという思想で設計されたコンピュータが出現した(UNIVAC 1100/60)．モジュールの多重構成においても再構成のきっかけをなすものは誤りの検出である．チェッキングを重要視する傾向が顕著になりつつある一方，チェッカ自身の故障をも検出できるセルフチェッキングチェッカの研究が現在盛んに行われている．

③ テ ス ト

ダウン状態の時間を短くするためには，なるべく早く修理，回復を行わなければならない．迅速な修理を可能にするためにいろいろな故障検査・診断の技術が研究されている．故障テストを生成する方法としていわゆるDアルゴリズムが広く用いられている．しかしEX-OR ゲートを多く含む回路に対してはDアルゴリズムはあまり能率的ではなく，これを改善するためにごく最近 PODEM といわれるものが提案された[2]．回路によってはD アルゴリズムより35倍も早くテストを生成するという．

一方，Dアルゴリズムのように決定論的なテスト生成を行うのではなく，ランダム入力を用いる簡便なテストが早くから行われているが[3]，故障の検出能力，あるいは逆に無故障であることの保証能力について1970年代後半に疑問が出された[4),5]．反面，入出力間の相関をとれば検出能力は高いとする主張もある[6]．診断データを圧縮し簡便なものとするという観点からCompact Test[7] や Transition Count Test[8] が注目されている．

主メモリ（CI メモリ）についてはパターン依存故障のテスト方法について研究が行われている[9),10]．

順序回路に対するテストは，1960年初めに Seshu らによって提案された帰還路を断ち切り回路を組合せ化するという方法以上にまだ出ていないように見える．状態を判定系列（Locating 系列も基本的な発想は判定系列と同じとみてよいであろう）で確認して行く Hennie の理論は美事であるが結局実用されなかった．

④ Testability Design

LSI の時代を迎えテスト生成問題はきわめて困難な壁に直面している．内部のゲート数が非常に多くなる（たとえば10万～20万ゲート）ので，従来の方法でテストを生成していたのでは，たとえそのためにコンピュータを用いるとしても厖大な時間が必要になる．さらに外部接続のためのピン数の制約もある．テストを容易にするように回路を設計しておく新しいアプローチ (Testability Design) が模索されている．1つの方向として IBM は，テスト時にチップ内のラッチをすべて直列に接続し，テスト入力および結果を直列にスキャンイン・スキャンアウトする方式を打ち出し，大きなセンセイションを与えた (1977年)[11]．もっともこれと同種な着想はそれ以前に NEC から発表されているので NEC の方が先だとする見解もある[12),13]．このほか，PLA を関数とは無関係にテストできるように構成する方法[14)~16]，LSI 中に組み込む Built-In Tester[17]，Autonomous Tester[18),19] などいろいろ新しい提案が行われている．

Testability Design に関連し，Vail でのワークショップで Testability の評価の重要性が指摘されている．

⑤ システム故障診断

システム中のユニットが互いにテストし合い，各ユニットのテスト結果全体をみてどのユニットが故障であるかを定めることができるようにユニット間の接続関係を定めるという問題は Preparata らによってはじめて提起されてから大きな関心を呼んだ[20]．しかし彼らのモデルは誰が（あるいはどこが）全体のテスト結果をみるかという点が不問に付されており，これを大域的なネットワークにそのまま適用することはできない．最近，ユニットはそれぞれのテスト結果を通信し合い，正しいユニットは皆同一の結論を持つように接続するという方法が検討されている[21]．思想的には一歩前進したものといえよう．

⑥ オペレーティングシステム

耐故障動作に OS が大きな役割を果たしていることはいうまでもない．早くから入出力動作に関するリトライが行われているが，先の UNIVAC 1100/60 ではマイクロコードの水準でリトライが行われている．

ダウン状態からアップ状態へシステムを回復する場合，ダウン直前のシステムの状態を復元しなければならない．このためアップ状態中に適当な時点（たとえば30秒ごと）にシステムの状態パラメータを適当なメモリにセイブする．回復するときは，このセイブしたパラメータを読み込んでシステムの状態を以前の時点に戻し，そこから再スタートする．

分散処理の場合，システムの回復はかなり面倒であ

る．自分は故障していなくても他の故障ユニットによって現在のデータが汚染されていることも有り得るので，処理のロールバックを行わなければならない．各ユニット間の交信の記録・抹消をいかに行うべきか，そのために必要となる内部リストの種類と通信プロトコルに関する研究が行われている[22]．分散データベース化が進めば，この種の実用的な方法を見出すことが今後重要となろう．

Non-Stop Computer のうたい文句で有名な Tandem システムでは，1つのプロセサ・メモリモジュール（最高16台まで組込み可）があるプログラムを実行するとき他のモジュールにそのコピーを必ず保存させ，またデータ領域の値をたとえば1トランザクションの処理ごとに相手側に知らせるなどして，実行モジュールが故障しても実質上 Non-Stop でユーザプログラムを処理できるようにしている[23]．

このほか，電子交換機では Audit[24] と呼ばれるチェックプログラムを用意し，テーブル内の矛盾，タイミングエラなどをチェックしている．また Watch-Dog Timer[25] でプログラムの暴走を監視している．

⑦　ソフトウェアの信頼性

システムダウンの原因をハードウェアとソフトウェアに分けると，ソフトウェアによる場合が全体の 65% ほどにも達するといわれている．今後，バグのないソフトウェアをいかに生産するかという点が大きな問題となろう．特にシステム再構成，回復といった通常はあまり生じない異常状態を処理するプログラム中にバグが残っていることが多い．このため適当な故障を故意に挿入し，ソフトウェアが正しく動作することをチェックすることも行われている．

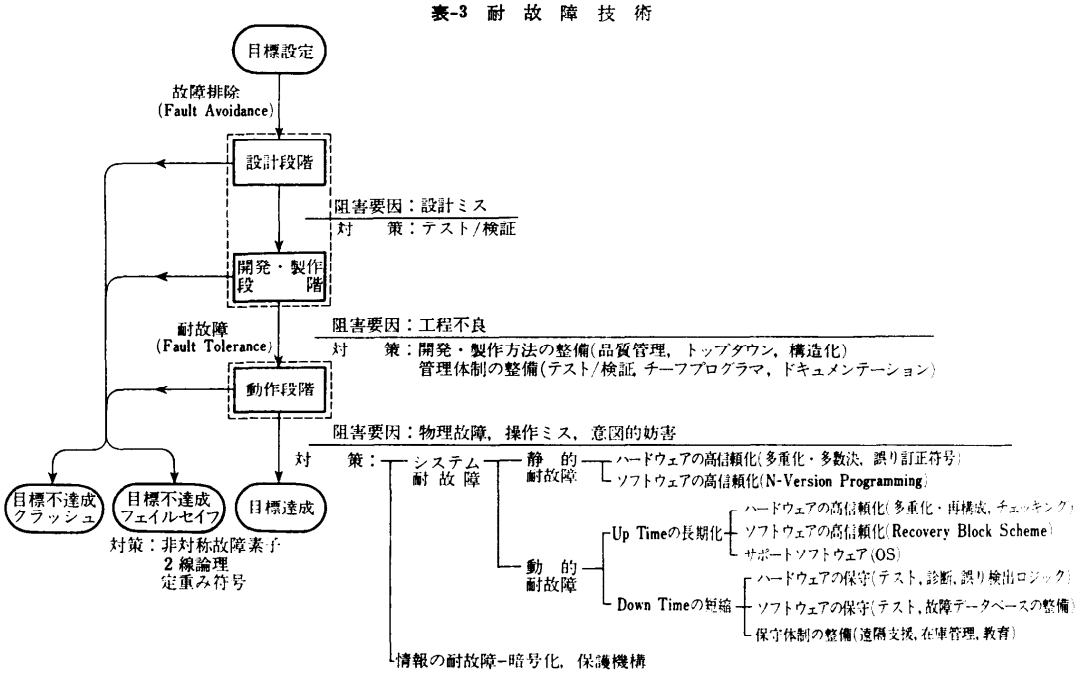
ソフトウェアそのものの正しさを Assertion を挿入してチェックする，いわばセルフチェッキングソフトウェア[26] が研究されている．ソフトウェアの記述体系，構造，開発体制，ドキュメンテーション，テストなどをきちんと整備することによってソフトウェアの信頼性を高めるという考え方が主なようである．しかしソフトウェアエンジニアリングの課題は多い．

Univ. Newcastle Upon Tyne や UCLA で，ソフトウェアバグによって処理が中断してもシステムはアップ状態を続ける Recovery Block Scheme[27] や N-Version Programming[28] の研究が行われている．

⑧　フェイルセイフシステム

フェイルセイフシステムに関する理論は 1960 年代末から 1970 年代半ばにかけて主にわが国の研究者によって発展され，世界をリードした分野の1つである[29]．非対称的な故障素子の研究，組合せ回路の構成

表-3　耐　故　障　技　術



論，順序回路の設計法など多岐にわたっている．現在フェイルセイフ性を具備した鉄道用制御装置がわが国の国鉄はじめ，西独，スウェーデン，イギリス，フランスなどで研究されている．

⑨ 耐故障システム

特に耐故障設計を施されたシステムとして有名なものをあげておこう．まずベルの NO. 1 ESS[30),31)], JPL の STAR[32)], SCCM[33)], UC バークレーの PRIME[34)], ARPA ネットワーク用 IMP として開発された Pluribus[35)]，カーネギメロン大学の C. mmp[36)], Cm*[37)], C. vmp[38)], Draper 研究所の FTMP[39)], SRI の SIFT[40)] などがある．わが国でも前記の国鉄のシステム(SIPSS)のほかにハードウェア多数決回路を用いたものが東北大で[41)]，SIFT に相当するものが東工大で[42),43)]試みられている．予備を設けておくといった一般的なシステムは，広く実用に供されている[44)]．

以上，いろいろな諸技術を思いつくままに並べ立てたが，これらを耐の水準，故障の範囲といった観点から整理すると**表-3** のようになろう．

4. 評　価

前章で述べた諸技術は耐故障を直接支援するものであるが，評価は間接的な支援技術である．間接的だが評価はきわめて重要であることを強調しておきたい．評価によって我々は具体的な設計指針を得ることができる．

4.1 故 障 率

システムの信頼度評価に関する理論的研究は実に長い歴史をもち，立派な成書も多い．しかし詳細な理論的検討とは別に実際上の評価を行う場合，構成要素の故障率を一定とする事が多かった．これに対して，最近，カーネギメロン大学では自作のコンピュータ複合体 (C. mmp や Cm*) の使用経験から，過渡的故障発生の時間間隔がワイブル分布に非常によく合致すると報告している[45)]．コンピュータの負荷によって故障率が変るという指摘もある[46)]．

4.2 信頼度・性能評価

従来のシステムの信頼度のみに着目した評価では不十分で，システムの性能と絡めて評価すべきであるという新しい概念をはじめて導入したのは Beaudry である (1978年)[47)]．たとえば，2台のモジュールから成る待機予備システムとアップ状態のときは2台ともタスクを実行する Gracefully Degrading システムを比較するとき，評価尺度としてシステム全体の信頼度，平均寿命 MTTF をとるといずれに関しても待機予備システムの方がよいという結論を得る．しかし時刻 t から始めて処理量 T (命令ステップ数でも処理トランザクション数でもよい) を無故障で実行する確率 (これを彼女は Computation Reliability と呼んでいる)，あるいはスタートしてからダウンに至るまでの平均の処理量などで評価すると，時刻 t またはシステムオーバヘッドによっては Gracefully Degrading システムの方がよい場合があるというのである．

これに続いて，処理量はシステムの性能だけでなく，その時の要求の多少によっても影響されるはずゆえ，システムの故障に関する状態，その時の性能，要求の発生率，処理率などのパラメータを考慮してスループットの期待値その他を評価すべきであるという指摘が行われた[48)]．思想的には，信頼度のみの評価の水準から進歩したというべきであろう．

これと近い考え方であるが，Performability なる尺度が提案されている[49)]．システムの処理のレベルをいくつかの段階に格付けし，故障によってシステムの性能が劣化したときどの段階の処理が可能であるかを調べる．こうして確率過程としての処理段階の格付けの期待値 (すなわち Performability) を求めるのである．

4.3 信頼度・コスト評価

信頼度を向上するために冗長設計を行えばコストが上昇するのは当然である．特殊ミッションの場合は信頼度が至上命令ともいえるから，コストの増加は止むを得ぬものと容認されよう．しかし商業ベースの場合は信頼度とコストのトレードオフをはからねばならない．トレードオフの興味ある考え方を以下に紹介する．

コンピュータの故障によってミッションが不可能になるための損害の期待値 L を評価する．一方，コンピュータの信頼度 R を要求された水準に保つよう設計するとしたときのコスト C を求める．$\Delta C/\Delta R = -\Delta L/\Delta R$ となるように設計点を選ぶ[50)]．

コストを考える場合，今後は保守コストをも含めるべきであろう．モジュールの多重構成においては定期保守を行うことが可能であるから，保守コストの算定が容易となろう[51)]．このことから，与えられた信頼度水準の下で，保守コストを含めたシステムコストという観点から，モジュールの多重構成に関する設計指針が得られるのではないかと思う．

4.4 耐故障設計

これまで，単なる信頼度の評価，信頼度と性能，信頼度とコストを関連づけた評価が行われてきた．しかし耐故障設計のゴールは，信頼度，性能，コストの三者を関連づけた総合的な評価によって正当化されるものでなければならないであろう．

5. 将来の問題

耐故障技術はコンピュータのアーキテクチャ，構成要素，使用形態などと無関係ではあり得ない．前章までの各所で問題点を指摘してきたが，ここで改めてまとめると以下のようになろう．

まず，LSI 化が大きな問題である．プロセステクノロジは進歩したが，Testability の確保，故障の発生にもかかわらず活き続けるサバイバル機能の確保に関して何を組み込むべきか，方向性すら見出されていないようにみえる．

ジョセフソン素子にもとづく新しい回路構成，新しいアーキテクチャの採用によってこれまで営々と築いてきたテスト，診断等の技術はすべて崩壊してしまうかもしれない．

1,000 ステップの命令に 0.25～10 個のバグが存在するといわれるソフトウェアの信頼性を向上することも緊急な課題である．ソフトウェアの生産体制を整備してバグが入らないようにするアプローチは Fault Avoidance といえる．一方，入力データによっては最終結果に誤りが発生せずバグの存在を見落とすこともあるので，バグの存在を顕在化するような方法をインプリメントしておくことも重要であろう．

分散化，大域ネットワーク化に伴い，ネットワークあるいは全体のシステムの回復の問題はさらに検討する必要があるのではなかろうか．ネットワークにおけるフェイルセイフの概念はもっと認識されてよいのではないかと思う．

コンピュータの信頼度を高めた場合，それを実際に確認する手段がないという SRI の指摘は示唆に富んでいる．設計の検証問題が今後重要な課題となろう．

最後に，自然発生的な故障に対する耐性ばかりではなく，意図的故障——すなわち妨害，不正操作などに対する耐性，防護策に関するわが国の研究はきわめて不十分であることを指摘しておきたい．

技術に対する依存が大きくなればなるほど，技術的な障害に対するサバイバル技術に深い配慮と必要な投資を行うという思想的風土を定着させる必要がある．

参 考 文 献

1) 日経エレクトロニクス，1981 年 10-26 号，pp. 186-189 (1981).
2) Goel, P.: An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits, Digest of Papers, FTCS-10, pp. 145-151 (1980).
3) Breuer, M. A.: A random and an algorithmic technique for fault detection test generation for sequential circuits, IEEE Trans. Comput., C-20, 11, pp. 1364-1370 (1971).
4) Shedletsky, J. J. and McCluskey, E. J.: The Error Latency of a Fault in a Combinational Digital Circuit, Digest of Papers, FTCS-5, pp. 210-214 (1975).
5) Wakimura, Y. and Tohma, Y.: An Evaluation of Random Input Test for the Verification of Circuit Realizations, ACM-SIGDA Newsletter, 9, 1, pp. 2-4 (1979).
6) Ohteru, S., Kato, T., Hashimoto, S., Watabe, K. and Minegishi, K.: Digital Circuit Test System using Statistical Method, Digest of Papers, FTCS-10, pp. 179-181 (1980).
7) Parker, K. P.: Compact Testing: Testing with Compressed Data, Digest of papers, FTCS-6, pp. 93-98 (1976).
8) Hayes, J. P.: Transition Count Testing of Combinational Logic Circuits, IEEE Trans. Comput., C-25, 6, pp. 613-620 (1976).
9) Suk, D. S. and Reddy, S. M.: An algorithm to detect a class of pattern-sensitive faults in semiconductor random access memories, Digest of Papers, FTCS-9, pp. 219-226 (1979).
10) 岸 政七: Waltzing Pattern を用いた IC メモリ素子試験，電子通信学会論文誌，60-D，12，pp. 1031-1038 (1977).
11) Eichelberger, E. B. and Williams, T. W.: A logic design structure for LSI testability, Proc. 14th Design Automation Conference, pp. 462-468 (1977).
12) 小林 亮，松江繁樹，柴 宏: FLT に適したフリップフロップ回路，電子通信学会全国大会，No. 892 (1968).
13) Funatsu, S., Wakatsuki, N. and Arima, T.: Test Generation Systems in Japan, Proc. 12th Design Automation Conference, pp. 114-122 (1975).
14) Hong, S. J. and Ostapko, D. L.: FITPLA: A Programmable Logic Array for Function Independent Testing, Digest of Papers, FTCS-10, pp. 131-136 (1980).
15) Fujiwara, H., Kinoshita, K. and Ozaki. H.: Universal Test Sets for Programmable Logic

Arrays, ibid, pp. 137-142 (1980).
16) Sato, T. and Tohma, Y.: A New Configuration of PLA with Function Independent Test, FTC 研究会資料 (1982).
17) Koenemann, B., Mucha, J. and Zwiehoff, G.: Built-In Logic Block Observation Techniques, Digest of Papers, 1979 Test Conference, pp. 37-41 (1979).
18) Eiki, H., Inagaki, K. and Yajima, S.: Autonomous Testing and its Application to Testable Design of Logic Circuits, Digest of Papers, FTCS-10, pp. 173-178 (1980).
19) McCluskey, E. J. and Bozorgui-Nesbat, S.: Design for Autonomous Test, IEEE Trans. Comput., C-30, 11, pp. 866-875 (1982).
20) Preparata, F. P., Metze, G. and Chen, R. T.: On the Connection Assignment Problem of Diagnosable Systems. IEEE Trans. Electron. Comput., EC-16, 6, pp. 848-854 (1967).
21) Kuhl, J. G. and Reddy, S. M.: Fault Diagnosis in Fully Distributed Systems, Digest of Papers, FTCS-11, pp. 100-105 (1981).
22) Wood, W. G.: A Decentralized Recovery Control Protocol, Digest of Papers, FTCS-11, pp. 159-164 (1981).
23) Bartlett, J. F.: A Non-Stop Operating System, Proc. Hawaii International Conference on System Sciences, pp. 103-117 (1978).
24) Connet, J. R., Pasternak, E. J. and Wagner, B. D.: Software Defenses in Real-Time Control Systems, Digest of Papers, FTCS-2, pp. 94-99 (1972).
25) Ramamoorthy, C. V., Cheung, R. C. and Kim, K. H.: Reliability and Integrity of Large Computer Programs, Computer Systems Reliability, pp. 617-709, Infortech Information (1974).
26) Metze, G. and Mili, A.: Self-Checking Programs: An Axiomisation of Program Validation by Executable Assertions, Digest of Papers, FTCS-11, pp. 118-120 (1981).
27) Anderson, T. and Lee, P. A.: Fault-Tolerance, Principle and Practice, Prentice Hall, pp. 251-272 (1981).
28) Chen, L. and Avizienis, A.: N-Version Programming: A Fault-Tolerance Approach to Reliability of Software Operation, Digest of Papers, FTCS-8, pp. 3-9 (1978).
29) まとめたものとして次が参考になろう．当麻喜弘：フェイルセイフシステムの理論，電子通信学会技術報告，R77-6 (1977).
30) Downing, R. W., Nowak, J. S. and Tuomenoksa, L. S.: No. 1 ESS Maintenance Plan, BSTJ, pp. 1961-2019 (1964).
31) Toy, W. N.: Fault-Tolerant Design of Local ESS Processors, Proc. IEEE, 66, 10, pp. 1126-1145 (1978).
32) Avizienis, A., Gilley, G. C., Mathur, F. P., Rennels, D. A., Rohr, J. A. and Rubin, D. K.: The STAR (Self-Testing and Repairing) Computer: An Investigation of the Theory and Practice of Fault-Tolerant Computer Design, IEEE Trans. Comput., C-20, 11, pp. 1312-1321 (1971).
33) Rennels, D. A., Avizienis, A. and Ercegovac, M.: A Study of Standard Building Blocks for the Design of Fault-Tolerant Distributed Computer Systems, Digest of Papers, FTCS-8, pp. 144-149 (1978).
34) Baskin, H. B., Borgerson, B. R. and Roberts, R.: PRIME-A modular architecture for terminal-oriented systems, Proc. SJCC, 40, pp. 431-437 (1972).
35) Ornstein, S. M., Crowther, W. R., Kraley, M. F., Bressler, R. D., Michel, A. and Heart, F. E.: Pluribus-A reliable multiprocessor, Proc. NCC, 44, pp. 551-559 (1975).
36) Wulf, W. A. and Bell, C. G.: C. mmp- a multi-mini-processor, Proc. FJCC, 41, pp. 765-777 (1972).
37) Swan, R. J., Fuller, S. H. and Siewiorek, D. P.: Cm*: a modular, multi-microprocessor, Proc. NCC, 46, pp. 637-644 (1977).
38) Siewiorek, D. P., Kini, V., Mashburn, H., McConnel, S. and Tsao, M.: A Case Study of C. mmp, Cm*, and C. vmp: Part I and Part II, Proc. IEEE, 66, 10, pp. 1178-1220 (1978).
39) Hopkins, A. L., Smith, T. B., Ⅲ, and Lala, J. H.: FTMP-A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft, ibid, pp. 1221-1239 (1978).
40) Wensley, J. H., Lamport, L., Goldberg, J., Green, M. W., Levitt, K. N., Melliar-Smith, P. M., Shostak, R. E. and Weinstock, C. B.: SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control, ibid, pp. 1240-1255 (1978).
41) 亀山充隆，樋口龍雄：TMR によるフォルトトレラントマイクロコンピュータシステムの一構成法，電子通信学会技術報告，EMCJ 78-57 (1979).
42) 古屋 清，木村秀明，当麻喜弘：SAFE システムのハードウェア，情報処理学会全国大会，No. 5 J-8 (1981).
43) 木村秀明，古屋 清，当麻喜弘：SAFE システムのオペレーティングシステム，ibid, No. 5 J-9 (1981).
44) 電子通信学会誌，59，4 (1976).
45) McConnel, S. R., Siewiorek, D. P. and Tsao, M. M.: The measurement and analysis of

transient errors in digital computer systems, Digest of Papers, FTCS-9, pp. 67-70 (1979).
46) Castillo, X. and Sieworek, D. P.: Workload, Performance, and Reliability of Digital Computing Systems, Digest of Papers, FTCS-11, pp. 84-85 (1981).
47) Beaudry, M. D.: Performance-Related Reliability Measures for Computing Systems, IEEE Trans. Comput., C-27, 6, pp. 540-547 (1978).
48) Gay, F. A. and Ketelsen, M. L.: Performance Evaluation for Gracefully Degrading Systems, Digest of Papers, FTCS-9, pp. 51-58 (1979).
49) Meyer, J. F.: On Evaluating the Performability of Degradable Computing Systems, Digest of Papers, FTCS-8, pp. 44-49 (1978).
50) Hecht, H.: Figure of Merit for Fault-Tolerant Space Computers, Digest of Papers, FTCS-2, pp. 189-192 (1972).
51) Oda, Y., Tohma, Y. and Furuya, K.: Reliability and Performance Evaluation of Self-Reconfigurable Systems with Periodic Maintenance, Digest of Papers, FTCS-11, pp. 142-147 (1981).

(昭和56年12月15日受付)