■6 群（コンピュータ・基礎理論とハードウェア）- 7 編（ディペンダブルコンピューティング）

1 章 ディペンダブルコンピューティングの基礎

（執筆者：）

■概要■

【本章の構成】

■6 群 - 7 編 - 1 章

1-2 冗長化技術に基づく耐故障化設計

（執筆者：伊藤秀男）[2008 年 5 月 受領]

コンピュータシステムの障害に対処する耐故障化設計は，ハードウェアオーバヘッドによる空間冗長とタイムオーバヘッドによる時間冗長の冗長化技術に基づく．比較的冗長度を高めて内部に発生した故障や誤りを外部へ出さない誤りマスク技術と，冗長度を低く抑えて外部へ影響が出る故障や誤りが発生した場合には検出し，再実行により回復する誤り検出・回復技術とがある．ここでは，誤り制御と故障モデルの基本的な定義を述べた後，誤り検出技術と誤りマスク技術を述べる．誤り回復技術や，そのほかの関連技術については，文献 1, 4) などを参考にされたい．

1-2-1 誤り制御と故障モデル

システムが正常な動作をしないとき，システムには障害（failure）があるという．システムの障害を小さく抑える技術を総称して誤り制御という．障害が発生するのは，システム内の構成要素に正しくない出力，すなわち誤り（error）があるためであり，その誤りはその原因である故障（fault，フォールト）があるためである．誤り制御を行うためには，システムに実際に発生する故障や誤りを仮定（モデル化）する必要がある．この仮定する故障や誤りを故障モデル（フォールトモデル）という．一般にシステム内に発生する故障は 1 箇所であり，このような故障モデルを単一故障という．本記事でも，特に断らない限り単一故障を仮定している．

システムの内部に局所的に故障や誤りが発生してもその影響を外部へ出さない技術は誤りマスク技術といわれる．

一方，誤りが発生したらその誤りを検出した後，できるだけ早くシステムの異常を回復し，処理を再開あるいは続行させたい．この場合の誤りの発生を検出する技術を誤り検出ロジックという．また，処理が再開できるためには，誤り発生時点より前のシステム内の状態を保存しておき，その状態へいったん戻して再開することが必要になる．これらの回復に関連する技術（処理）を誤り回復技術（回復処理）という[3)]．

一般に誤りの原因は，雑音などの何らかの要因で発生する一時的な故障，すなわち過渡故障あるいは間欠故障である場合が圧倒的に多い．近年は集積回路の微細化・高速化・大規模化に伴って間欠故障の原因ともいえる指標を表すものとして，シグナルインテグリティが問われている[5)]．また，宇宙からの中性子線が LSI 内の PN 接合部へ照射することによって，そのエネルギーがスパイク状のノイズを発生させ，それが原因でフリップフロップに一時的な誤りをもたらすソフトエラーが問題になってきている[6)]．

このような一時的な誤りをもたらす誤りは上記の回復処理によって回復できることになる．しかし，回復処理を何度試みても同じ誤りが発生して失敗する場合は，永久故障があることになり，修理が必要になる．

1-2-2 誤り検出技術

(1) 誤り検出符号

論理回路やバスあるいは記憶装置などシステムのユニットやモジュール（M と表す）の

出力がとるパターンを符号語と呼び，その集合を符号（C と表す）と呼ぶ．C の任意の要素（符号語）u に対して，M 内の故障モデルによってとりえるパターンからなる集合を S_u（これを誤りパターン集合と呼ぶ）と表す．いま，C の任意の異なる二つの要素 u, v に対する誤りパターン集合を S_u, S_v とするとき，$v \notin S_u$ を満たすならば，C は誤り検出符号であるという．また，$S_u \cap S_v = \phi$ を満たすとき，C は誤り訂正符号という[7]．誤り訂正符号については，本章 1-2-3(1) で述べる．誤り検出符号の符号語と誤りパターン集合との関係を図 1･1 に示す．

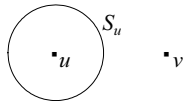


図 1･1　誤り検出符号の符号語 u, v と u の誤りパターン集合 S_u との関係

一般に，誤り検出符号を利用する誤り検出は，図 1･2 に示すように，対象にする回路 M の出力を誤り検出符号 C によって符号化し，検査回路 N_c が，常時 M の出力が符号 C であるかどうかを検査することによって行う．

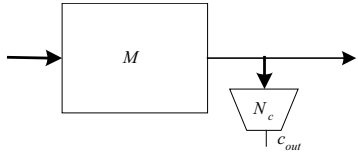


図 1･2　誤り検出符号を利用する誤り検出

誤り検出符号 C は，M の誤りの特性（出方）などに合わせて選択される．そのような符号には，線形符号（単一パリティ符号，チェックサム（check sum）符号，巡回符号），非順序符号（2 線式符号（2-rail code），Berger 符号，m-out-of-n 符号），算術演算符号（剰余符号（residue code），低価格（low-cost）剰余符号，逆剰余符号（inverse residue code），AN 符号）などがある[2, 7, 8)−11)]．

誤り検出符号を利用して誤り検出の徹底化を図った方式にセルフチェッキング（以下，SC と略記）方式がある[12)−14)]．図 1･2 において，M の対象にする故障モデルから得られる故障集合 F_M の任意の故障 f が発生したとき，M の出力が誤るならば必ず C の要素にならないように M を構成する．M のこの特性をフォールトセキュア（FS（Fault Secure））性という．一方，F_M の任意の故障 f を検出できる M への通常入力が少なくとも一つ以上あるとき，M はセルフテスティング（ST（Self-Testing））性を満たすという．FS 性と ST 性をともに満たす回路をトータリーセルフチェッキング（TSC（Totally Self-Checking））回路という．

検査回路 N_c は，M の出力が C の要素であるかどうかを N_c の出力 C_{out} によって正しく判定できる必要がある．N_c のこの特性をコードディスジョイント（CD（Code Disjoint））性という．一般に，SC 方式では，N_c の故障も考慮する必要があるので，N_c 自身も FS 性と ST 性を満たすことが好ましい．このことから，N_c 自身の正常出力も 2 線式符号に符号化されることが多い[13)]．

ただし，M でも N_c でも，仮に ST 性を満たせない故障（一種の冗長故障）が発生しても，各々 FS 性と CD 性を常に満たせるならば十分である．この特性を満たす M，N_c を各々ストロングリーフォールトセキュア（SFS（Strongly Fault Secure））回路[15]，ストロングリーコードディスジョイント（SCD（Strongly Code Disjoint））回路[16]という．

現在まで，ROM，PLA，演算回路，順序回路，各種符号の検査回路，及びプロセッサの試作などに対して各種の符号を利用した多くの SC 構成が提案されてきている[2, 7, 9, 10]．また，完全な SC 回路構成はハードウェアオーバヘッドを大きくすることから，上記の SC の特性を部分的に満たせばよいと考えるパーシャリー（Partially）SC 回路構成[17]なども提案されている．

(2) 二重実行方式

二重実行方式は，一つの処理（クロックごとの動作，命令，スレッド，プロセス，タスクなど）を二重に実行し，それらの結果を比較することによって，誤りを検出する方式である．この場合の “ 二重 ” の意味には以下の 3 通りがある．

(a) 同じハードウェアで 2 回繰り返して実行する（時間冗長）
(b) 異なるハードウェア上で実行する（空間冗長）
(c) 異なるソフトウェアを実行する（ソフトウェア冗長）

更に (b),(c) の組み合せの実行も考えられる．(b) の場合に，(b1) クロックレベルで同期をとるか，(b2) ソフトウェアによって同期をとるかの違いもある．また，実行結果の比較をハードウェア（検査回路）で行うか，ソフトウェアで行うかの違いもある．(a) の場合には，間欠故障は検出できるが，固定故障は検出できない．(b1) はいわゆる二重化であり，更に検査回路が TSC 性を満たす場合はセルフチェッキング方式となる．図 1･3 に二重化方式を示す．

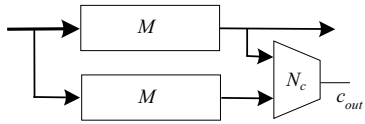


図 1･3　二重化方式

(b2) は一般にマルチプロセッサ構成や分散処理の場合に当たる．多くの高信頼化トランザクション処理システムなどは二重実行方式である[18, 19]．

二重実行方式に関する研究に，マルチプロセッサ（密結合）[20]，マルチコンピュータ（疎結合）[21]，または分散処理環境[22]でのプロセスの割当て方法，二重実行方式の誤り検出システム設計を目標としたマイクロアーキテクチャまたは ASIC のための高位論理合成方法（High Level Synthesis）[23]などがある．

二重実行方式の特別な場合と考えられるものに以下がある．

(イ) オペランドシフト時間冗長方式
(ロ) 交番論理

(イ) のオペランドシフト時間冗長方式は，ALU の演算を通常に行った結果と，オペランドの演算位置を替えて行った結果とを比較して誤りを検出しようとするものである．オペランドを 2 ビットシフトする方式[24]，1 ビット巡回シフトする方式[25]などがある．

(ロ) の交番論理は，論理値 1, 0 を連続する時間系列の (10), (01)（または逆）の 2 ビットに対応づけ，故障があるならば (00) または (11) の結果になることを利用したものである[26]．

次に，1 クロック幅の時間も満たない時間冗長方式の特別な例を示そう．最近のソフトエラー対策の誤り検出方式の例である．ラッチのソフトエラーによる誤りマスクに対しては本章 1-2-3 節のなかで述べるが，ここは論理回路のなかで発生したソフトエラーによる過渡的な誤り（SET: Single Event Transient）波形が運悪くラッチの入力まで伝搬し，ラッチによって捕獲される場合である．一般に SET 波形の幅は 1 クロック幅よりもはるかに狭く，τ 以下であると仮定する．このとき，図 1･4 に示すように，回路 M の出力をラッチ L_0 が捕獲するとき，クロック c_0 を τ 時間だけ遅らせたクロック c_1 で冗長ラッチ L_1 が同じ回路出力を捕獲する．比較器 N_c は L_0 と L_1 の出力が同じであるかどうかを検査する．このとき，もし誤った値の SET が L_0 にラッチされるならば，比較器 N_c によってその誤りを検出することができる[27]．

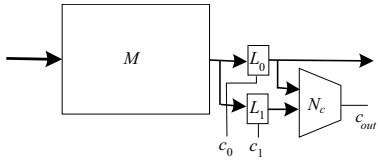


図 1･4　τ 時間冗長ソフトエラー誤り検出方式

(3) コントロールフローチェック方式

コントロールフローチェックはシステムの稼動中に，制御の流れ（プログラムの流れ）などが指定された動作と一致するかをチェックするものである．したがって，演算結果などのデータパス系の故障に対してはあまり効果はない．このことからもフォールトカバレッジには限界がある．代表的なものに，ウォッチドッグタイマ方式[2]，命令レベルチェック方式[28]，順序回路状態遷移レベルチェック方式[29]などがある．

1-2-3 誤りマスク技術

(1) 誤り訂正符号

誤り訂正符号の定義などについては，本章 1-2-2(1) のなかで誤り検出符号を述べるついでに述べた．詳しくは文献 7) を参照されたい．誤り訂正符号の符号語 u, v と誤りパターン集合 S_u, S_v との関係 $S_u \cap S_v = \phi$ を図 1･5 に示す．

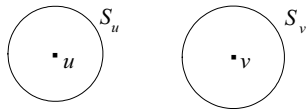


図 1･5　誤り訂正符号の符号語 u, v とその誤りパターン集合 S_u, S_v との関係

計算機の主記憶装置には，1 ビット誤りは訂正し，2 ビット誤りは検出することができる SEC-DED（Single Error Correcting and Double Error Detecting）符号が利用されて

いる．これは符号の H マトリクスが奇数重みベクトルであれば，満たすことができる．例えば，64 ビットの情報ビットに対して，8 ビットの検査ビットを付加することによって達成できる．

16 ビットの情報ビット $(x_0x_1\cdots x_{15})$ の場合に例を示そう．6 ビットの検査ビット $(c_0c_1\cdots c_5)$ をつけ，$w = (x_0x_1\cdots x_{15}\ c_0c_1\cdots c_5)$ を符号語とする．ただし，H マトリクスを

$$H = \begin{bmatrix} 1111111111000000100000 \\ 1111000000111111010000 \\ 1000111000111000001000 \\ 0100100110100110000100 \\ 0010010101010101000010 \\ 0001001011001011000001 \end{bmatrix}$$

とするとき，符号化では，情報ビット $(x_0x_1\cdots x_{15})$ に対して，検査ビットを $(c_0c_1\cdots c_5)$ とするとき，$w = (x_0x_1\cdots x_{15}\ c_0c_1\cdots c_5)$ に対して $Hw^{\mathrm{T}} = 0$ を満たすように，検査ビット $(c_0c_1\cdots c_5)$ をつける．復号の際は復号器により，シンドローム S として

$$S = Hw^{\mathrm{T}}$$

を計算する．このとき S から次のことが分かる．

(a) $S = 0$　誤りがない

(b) $S = h_j$（h_j は H の j 列）w の j 番目に 1 ビット誤りがあり，訂正して出力

(c) $S \neq 0$ （S の重みは偶数）2 ビット誤りがあり，その旨を報告する

ハードディスクや光ディスクなどの外部記憶装置のための符号には巡回符号が利用されている[7)]．

(2) 多重実行方式

多重実行方式のなかでも三重実行方式がよく用いられる．一つの処理を（δ 時間幅，クロック幅，命令，スレッド，プロセス，タスクなど）のレベルで三重に実行し，それらの結果を多数決をとって誤りをマスク（訂正）する方式である[1, 2, 4)]．この場合の “ 三重 ” の意味は，二重実行方式の場合と同様に，以下の 3 通りがある．

(a) 同じハードウェアで 3 回繰り返して実行する（時間冗長）

(b) 三つの異なるハードウェア上で実行する（空間冗長）

(c) 異なるソフトウェアを実行する（ソフトウェア冗長）

更に (a), (b), (c) の組み合せやその変形の実行も考えられる．

まず最初に, 代表的な (b) の方式に属する三重化と多数決回路による TMR（Triple Modular Redundancy）方式を図 1･6 に示す．三つのモジュールはクロックレベルで同期がとられて同じ処理を行い，その出力を多数決回路 V によって出力する．三つのモジュールのなかの一つに誤りがあっても訂正されて出力される．

この場合に，多数決回路 V の誤りは訂正することができない．そこで，多数決回路自体も三重化してシステムを組み上げていく方式がある．

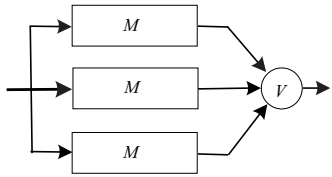


図 1·6 TMR 方式

次は，時間冗長の例を示そう．本章 1-2-2 節のなかで，論理回路のなかで発生したソフトエラーによる過渡的な誤りを検出する τ 時間冗長方式を示した．ここでは，これと同様な手法によって，論理回路のなかで発生したソフトエラーによる過渡的な誤りを訂正（マスク）する 3τ 時間冗長方式を示す[30]．図 1·7 に示すように，回路 M の出力を二重化したラッチへ入力する．片方のラッチへの入力には時間 τ の遅延素子を挿入する．二つのラッチ出力は c 素子と呼ばれる素子へ入力して出力する．この c 素子は図 1·8 に示す回路で，二つの入力が一致するときのみ入力の論理値を反映させ，異なるときはハイインピーダンス出力状態となって，直前に入力が一致していたときの値を継続出力する．このことから，パルス幅が τ より小さいソフトエラーをマスクすることができる．これは，ラッチの 2 倍の空間冗長と 3τ の時間冗長によって誤りが訂正できる例である．文献 30) では，類似の手法で回路 M を二重化することにより，ソフトエラーの過渡的な誤りを訂正する空間冗長方式も示されている．

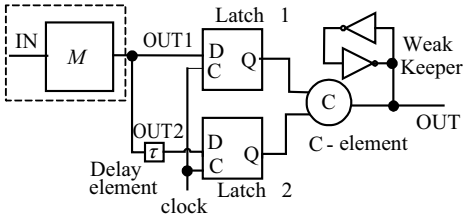


図 1·7 3τ 時間冗長ソフトエラー誤りマスク方式

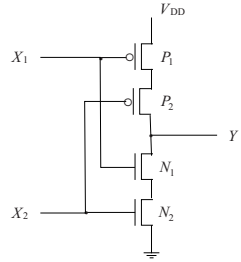


図 1·8 c 素子

ラッチ内部に発生したソフトエラーをマスクできる多くの方式が最近提案されている[27, 31)−33)]．これらは，同じモジュールを多重に並べる単純な多重化方式ではなく，ソフトエラーによって発生した SET を多重方式に近い冗長回路とその電子的な回路動作によってマスクする方式である．これらの方式も空間冗長方式に属する．ラッチ内のどこに誤りが発生してもマスク可能な方式から，一部の箇所に発生した場合はマスク不可能な方式など，マスク率（発生したソフトエラーがマスクされる割合）や回路の複雑さ（オーバヘッド）が様々なものなど多くが提案されている．図 1·9 にはそれらのなかの一つの方式を示す[31]．P_1, P_2, N_1, N_2, I_1, I_2 内にソフトエラー（SET）が一時的に発生しても，Q の値を反転させることはない．いわゆる二重化の両方が反転しない限り Q の値の反転はない．

また，最近の VLSI や SOC の製造後のテストではテスト容易化の目的でスキャン方式を取り入れている場合が多い．そこで，ソフトエラーをマスクしながらも，更にスキャン方式も可能にするラッチ類も提案されている[34)−37)]．

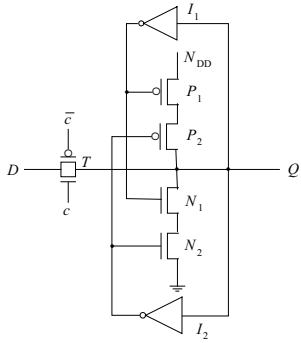


図 1·9　ソフトエラー誤りマスクラッチの例

■参考文献

1) 当麻喜弘編著, “ フォールトトレラントシステム論, ” 電子情報通信学会，1990.
2) 南谷崇, “ フォールトトレラントコンピュータ, ” オーム社，1991.
3) 伊藤秀男，日本信頼性学会編, “ 信頼性ハンドブック（6.1 誤り検出ロジックと回復）, ” pp.299-306, 日科技連出版社，1997.
4) 米田友洋，梶原誠司，土屋達弘, “ ディペンダブルシステム, ” 共立出版，2005.
5) 鈴木五郎, “ システム LSI 設計入門, ” コロナ社，2003.
6) T. Karnik, P. Hazucha, and J. Patel, “Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes,” IEEE Trans., Dependable and Secure Computing, vol.1, no.2, pp.128-143, 2004.
7) E. Fujiwara, “Code Design for Dependable Systems,” Wiley Interscience, 2006.
8) T.R.N. Rao, “Error Coding for Arithmetic Processors,” Academic Press, 1974.
9) John Wakerly, “Error Detecting Codes, Self-Checking Circuits and Applications,” North-Holland, 1978.
10) T.R.N. Rao and E. Fujiwara, “Error-Control Coding for Computer Systems,” Prentice Hall, 1989.
11) 当麻，南谷，藤原, “ フォールトトレラントシステムの構成と設計, ” 槙書店，1991.
12) F.F. Sellers, M.Y. Hsiao, and L.W. Bearnson, “Error Detecting Logic for Digital Computers,” McGraw-Hill, 1968.
13) W.C. Carter and P.R. Schneider, “Design of Dynamically Checked Computers,” Information Processing 68, pp.878-883, Amsterdam, 1969.
14) D.A. Anderson and G. Metze, “Design of Totally Self-Checking Check Circuits for m-out-of-n Codes,” IEEE Trans. on Computer, vol.C-22, no.3, pp.263-269, 1973.
15) J.E. Smith and G. Metze, “Strongly Fault Secure Logic Networks,” IEEE Trans. on Computer, vol.C-27, no.6, pp.491-499, 1978.
16) M. Nicolaidis, I. Jansch, and B. Courtois, “Strongly Code Disjoint Checkers,” FTCS-14, pp.16-21, 1984.
17) John Wakerly, “Partially Self-Checking Circuits and Their Use in Performing Logical Operations,” IEEE Trans. on Computer, vol.c-23, no.7, pp.658-666, 1974.

18) 向殿政男編, “ フォールト・トレラント・コンピューティング, ” 第 3 章 秋田雄志, “ フォールトトレラントコンピュータ, ” 丸善, 1989.
19) J. Gray 他著, 渡辺栄一編訳, “ OLTP システム, ” マグロウヒル, 1991.
20) D. Gu, D.J. Rosenkrantz, and S.S. Ravi, “Construction and Analysis of Fault-Secure Multiprocessor Schedules,” FTCS-21, pp.120-127, 1991.
21) J. Kim, H. Lee, and S. Lee, “Process Allocation for Load Distribution in Fault-Tolerant Multicomputers,” FTCS-25, pp.174-183, 1995.
22) C.J. Hou and K.G. Shin, “Replication and Allocation of Task Modules in Distributed Real-Time Systems,” FTCS-24, pp.26-35, 1994.
23) R. Karri and A. Orailoglu, “Optimal Self-Recovering Microarchitecture Synthesis,” FTCS-23, pp.512-521, 1993.
24) J.H. Patel and L. Y. Fung, “Concurrent Error Detection in ALU’s by Recomputing with Shifted Operands,” IEEE Trans. on Computer, vol.C-31, no.7, pp.589-595, 1982.
25) J. Li and E.E. Swartzlander, “Concurrent Error Detection in ALUs by Recomputing with Rotated Operands,” IEEE Int. Work. on DFT in VLSI Systems, pp.109-116, 1992.
26) 山本, 浦野, 渡辺, “ 交番論理系のフェイルセイフ性, ”電子情報通信学会論文誌, vol.54-C, no.12, pp.1079-1086, 1971.
27) M. Nicolaidis, “Time Redundancy-Based Soft-Error Tolerance to Rescue Nanometer Technologies,” Proc. IEEE VLSI Test Symp, pp.86-94, 1999.
28) S.J. Upadhyaya and B. Ramamurthy, “Concurrent Process Monitoring with No Reference Signatures,” IEEE Trans. on Computer, vol.43, no.4, pp.475-480, 1994.
29) G. Noubir, B.Y. Choueiry, “Algebraic Techniques for the Optimization of Control Flow Checking,” FTCS-26, pp.128-137, 1996.
30) S. Mitra, etal., “Combinational Logic Soft-Error Correction,” IEEE ITC2006, pp.824-832, 2006.
31) [Omana 03] M. Omana, D. Rossi, and C. Metra, “Novel Transient Fault Hardened Static Latch,” ITC03, pp.886-892, 2003.
32) Y. Komatsu, Y. Arima, T. Fujimoto, T. Yamashita, and K. Ishibashi, “A Soft-Error Hardened Latch Schemes for SoC in a 90nm Technology and Beyond,” Proceedings of the IEEE Custom Integrated Circuit Conference, pp.324-332, 2004.
33) S. Krishnamohan, and N.R. Mahapatra, “Analysisi and Design of Soft-Error Hardened Latches,” Proceedings of Great Lakes Symposium on VLSI Design, pp.328-331, 2005.
34) S. Mitra, N. Seifert, M. Zhang, Q. Sbi, and K.S. Kim, “Robust System Design with Built-In Soft-Error Resilience,” IEEE Design and Test of Computer, pp.43-52, 2005.
35) A. Goel, S. Bhunia, H. Mahmoodi,and K. Roy, “Low-overhead design of soft-error-tolerant scan flip-flops with enhanced-scan capability,” Proc. of Des. Automation Asia South Pacific Conf., pp.665-670, 2006.
36) M. Fazeli1, A. Patooghy1, S.G. Miremadi, and A. Ejlali, “Redundancy: A Power Efficient SEU-Tolerant Latch Design for Deep Sub-Micron Technologies,” 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’07)0-7695-2855-4/07, 2007.
37) T. Ikeda, K. Namba, and Hideo Ito, “Soft Error Hardened Latch Scheme for Enhanced Scan Based Delay Fault Testing,” 22^{th} IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT2007), (Rome.), pp.282-290, 2007.

■6 群 - 7 編 - 1 章

1-3 安全性

（執筆者：中村英夫）[2009 年 4 月 受領]

1-3-1 安全と安全性

安全に対する定義は，対象や分野にとって様々である．JIS C0508 では「受容できないリスクから免れている状態」としているし，MIL-STD-882（System Safety Program Requirements）と旧 JIS Z8115 では「人間の死傷または資材の損失もしくは損傷を与えるような状態のないこと」と定義している．また，日本学術会議の人間と工学連絡委員会・安全工学専門委員会がまとめた「社会安全への安全工学の役割（平成 12 年 3 月）」のなかでは，安全を「外的事由により心身の安寧が損なわれないでいる状態，及び自己が所有する経済的価値をもつ物品の価値の減少や損失が発生しない状態」としており，同様に安全を一つの状態とみている．一方，2000 年に改訂された JIS Z8115 では「人間の危害または資材の損傷の危険性が，許容可能な水準に押さえられている状態」とし，絶対的な安全という理想から現実的な定義にシフトしている．

安全性の度合いの評価尺度に RISK が使われる．JIS C0508 の機能安全の概念によれば，RISK 解析の結果得られたシステムが本来有する RISK レベルと，受容できる RISK レベルの乖離の度合いを安全度水準（SIL: Safety Integrity Level）といい，製品のライフサイクルプロセスの各フェーズを通して適切な作業とその審査を実施することにより，受容できる RISK レベルまで低減できるとしている．ハードウェアやシステムアーキテクチャの視点からは，RISK を低減するために用いられる技術が安全性技術であり，これに対しソフトウェアは SIL に応じた適切な作業と審査により品質が保証されるということになる．ところで，RISK は，このようなプロセスを通じて低減されるとはいえ完全にはゼロにはできず，残留 RISK の存在を否定できない．ここに，「許容可能な水準に押さえられている状態」とする定義の根拠がある．

一方，「安全性」は，安全が状態であるのに対し，安全を抽象化した概念，性質と見ることができる．安全性には，定量的な意味合いを含むが，使われ方としては，環境・防災，文化・風土，治安まで含めた広義の場合と，物や製品という製造物に限定した狭義の場合がある．安全工学は，後者の安全，すなわち物やシステムに安全性をどうやってつくり込むかということからスタートし，本質的な危険因子の有無や，その影響を限りなく低減することを目標とした手法に体系化されてきた．しかし，実際のシステムでは，ユーザやオペレータとしての人間とのかかわりが問題となったり，故意の妨害者による安全性喪失すら考慮しなければならない状況にあったり，複雑な様相を呈している．

1-3-2 安全の工学的側面と安全文化

製造物の安全は工学的手法により高めることができる．MIL STD-882A（1977）では，安全性の折り込みを組織的に行う方法論を提示した．計画段階での安全性要求仕様の明示と開発側（設計側）で行う予備安全性解析と安全性実施計画書作成，これらを下にした審査に従って，必要な安全性確認試験の義務づけなどの設計方針を定め，検査，受け入れ検査，運用廃棄と至るプロセスを通じた，安全性管理の重要性を指摘している．この思想は，今日の機能安全規格でみられる SLCP（Safety Life Cycle Process）を通しての安全性管理と軌

を一にしている．また，STD ではこの安全性解析に用いる手法として，FTA や FMEA を紹介している．

工学的立場でシステムの安全を確保する手法の一つに，フェイルセイフ設計がある．論理回路のフェイルセイフ設計は，回路の単調性を前提に非対称誤り素子を用いて可能であることが学問的に明らかになっている．しかし，通常のシステムで用いる素子の故障モードは複雑であり，実現する回路機能も単調回路では賄いきれないものが大部分である．この場合の安全性確保は，いかに漏れなく構成素子の故障状態を把握し，それぞれに的確な対応をとるかにかかっている．FMEA や FTA はこのためのツールである．また，RISK 解析や PSA などの手法も，事前に解析を漏れなく実施したかがその後の安全性保障の鍵を握る．この点で，傷害の範囲を限定化する方法や，複雑化をできる限り排除し得る設計は，合理的な方法として説得力をもつ．重要なことは，事前の配慮がどの程度かというカバレージの問題である．しかし，その真の値は神のみぞ知るというところに厄介さがある．

重要なことは，ユーザや設計者はもとより関係する人々（ステークホルダ）間で，RISK コミュニケーションが成り立ち，相互理解の下に展開されることである．ユーザがすべてのシステムに対し一方的に絶対安全を求めるのではなく，製造時になされる RISK マネージメントの内容を理解し，製品の残留 RISK を正しく認識したうえで運用時，使用時に残留 RISK が安全を脅かすことのないよう配慮するという相互のコミュニケーションがライフサイクルを通じた安全性の維持には必要になる．

一方，システムや機械が改良や拡張を重ねて運用されることも多い．機能の改善，生産性向上を意図した工夫や機能増強を意図したこれらの行動は，健全な生産活動の一環でもある．しかし，これらの機能増強や改善が，設計時には想定していなかった新たな RISK を生み出し，重大な事故を引き起こす事例は多い．システムの改良や工夫の目的のみに目を奪われ，その行為が今までのシステムや周囲の装置とのインタフェースにおいて，新たな RISK を生じさせないかどうかということを常に考えるという，安全文化の定着こそ，長い製品のライフサイクルを通じた安全性の確保には極めて重要である．

1-3-3 安全性の評価

信頼性は故障率や MTBF といった指標で数量的に評価できる．これに対し，故障していても安全は確保できるという場合があることからも分かるように，安全性は障害のモードやその影響を吟味する必要があるために，評価は容易ではない．

今日では，個々の不具合事象（threat とか harm といった用語が用いられる）に対し，その致命度と発生頻度をしん酌した指標として RISK が用いられる．RISK は四から五つの等級で評価され，そのレベルが高ければシステムの運用時に要求される RISK レベルまで低減することが求められる．

ただ，RISK は必ずしも安全性の指標としてのみ用いられるわけではない．システムによってはセキュリティが重視されたりリアルタイム性が重視されたりする場合もある．この場合にも RISK が安全性ではない次元での評価指標に用いられる．例えば，情報漏えいなどが大きな問題になるようなシステムでは，それぞれの不具合事象がどの程度情報漏えいに結びつくか，そしてその頻度はどうであるかといったかたちで RISK 評価が行われる．この場合の RISK はセキュリティに対する指標値である．

1-3-4 安全性の重要な側面

システムや装置などの製造物を前提として展開される工学的なアプローチは，安全性向上に有効かつ見通しのもてる方法論でもある．しかし，現実の社会にはこのような工学的アプローチのみでは対応できないものが多い．セイフティネットの整備が整わない段階での規制緩和が，多くの失業者を生みだした．弱肉強食の格差社会のなかで社会的保障や公的扶助すらも受けられない人々にあっては，生活そのものの安全，安心すら実感として抱けない状況におかれている．

このようなことを考えると，安全は機械安全や工学的アプローチのほかに，社会学，経済学，心理学，法学，そして行政学などとのかかわりを必然的に必要としている．この分野横断的な課題として総合的な研究が待たれている．

■6 群 - 7 編 - 1 章

1-4 システム例

1-4-1 フォールトトレラントコンピュータ

（執筆者：金川信康）[2009 年 4 月 受領]

ディペンダブルコンピュータ及びその実現手段であるフォールトトレラントコンピュータには様々な信頼性特性が要求され，安全性重視型，信頼度重視型，アベイラビリティ重視型，長寿命重視型などに分類される．これらの重視される信頼性特性は，コンピュータの用途によって異なる．

例えば，航空，宇宙（有人），交通，鉄道などの分野に適用されるシステムはシステムの誤動作が人命にかかわるおそれがあるため信頼度，安全性が重視され，特にフォールト検出カバレッジを高めることに重点を置いて設計されている．

更に，宇宙で使用される電子機器には宇宙線などの影響でソフトエラーと呼ばれる一過性のエラーが頻発するほか，温度，振動など過酷な環境下にさらされるうえ，打ち上げ後は修理が不可能であるため，長寿命であることが求められている．

また，金融，証券などの分野に適用されるシステムはシステムの誤動作が社会の混乱につながり，生産システムなどの分野に適用されるシステムはシステムの誤動作が経済的損失につながるため，高いアベイラビリティ（可用性）が求められている．

これらの様々な信頼性特性は，システムがいかに高い確率で，高い比率で，長時間正常に動作するかを表す第 1 の尺度（信頼度，アベイラビリティ，寿命）と，どれだけ確実に異常を検出できるか，影響を回避できるかを表す第 2 の尺度（安全性，検出カバレッジ）とに分けられる．

これらの尺度によりディペンダブルシステムを分類すると，図 1･10 に示すように第 1 の尺度が高いことが求められるシステム（高アベイラビリティシステム，長寿命システム），そして第 2 の尺度が高いことが求められるシステム（高安全システム），そして第 1，第 2 の尺度がともに高いことが求められるシステムとに分類される．

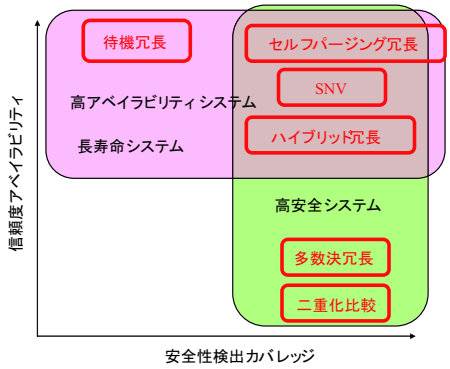


図 1･10　高信頼システムの分類

当然ながら，これらの要求される信頼性特性ごとに異なる高信頼化手法が用いられる．第 1 の尺度が高いことが求められるシステムでは主として待機冗長が用いられ，第 2 の尺度が高いことが求められるシステムでは主として二重化比較や多数決が用いられる．更に第 1，第 2 の尺度がともに高いことが求められるシステムでは多数決と待機冗長の特徴を組み合わせたハイブリッド冗長や，セルフパージング冗長[1]，SNV 方式[2]などが用いられる．

(1) 宇宙用コンピュータ

先に述べたように宇宙で使用される電子機器には，高い信頼性，長寿命が求められるだけでなく打ち上げコストの削減，ペイロードの増加のために小型軽量，低消費電力であることが望まれる．したがって，限られた冗長資源を使っていかに効率的に信頼性を高めるかが課題である．

宇宙用コンピュータとしてはジェット推進研究所（JPL）で開発された STAR（Self-Testing And Repairing）コンピュータ[3]が古典的存在で，その設計思想はその後の多くのシステムに影響を与えている．

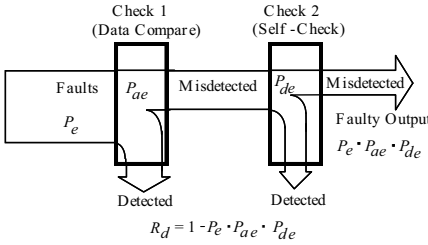


図 1･11　検出漏れの確率

自己チェック	データ照合	R_d	グレースケール
正　常	一　致	$1 - P_e \cdot P_{ae} \cdot P_{de}$	
誤り検出	一　致	$1 - P_e \cdot P_{de}$	
正　常	不一致	$1 - P_e \cdot P_{ae}$	
誤り検出	不一致	――	

図 1･12　データの信頼度

筆者らが開発した SNV（Stepwise Negotiating Voting）方式[2]では，図 1･11 に示すように誤り検出漏れの確率に着目して，冗長化したコンピュータそれぞれの信頼度を図 1･12 に示すように検査結果から推定し，最も信頼度の高いコンピュータの出力を選択する．以上により，限られたハードウェアで，より高い信頼性を得ることが可能となった．図 1･13 は SNV 方式を実現するためのシステム構成である．冗長化されたそれぞれサブシステムでデータが処理され，処理結果は検査結果とともにサブシステム間で交換される．ここのサブシステムでは，図 1･11，1･12 に基づきそれぞれの処理結果の信頼度を推定し，最も信頼度が高いと推定される結果を選択機能つき多数決回路 MV（Modified Voter）で最終出力として選択する．

この方式によるコンピュータ（図 1･14）は 1991 年に打ち上げられた宇宙科学研究所（現（独）宇宙航空研究開発機構　宇宙科学研究本部）の衛星「ひてん」に搭載され，3 年半にわたる期間，正常動作を続けてその任務を完遂した[4]．

(2) 一般産業用システム

1980 年代末，米国ではタンデム社[5, 6]，ストラタス社[7]などが商用の無停止型コンピュータを商品化し，大きな成功を収め，「フォールトトレラントコンピュータ」という言葉も市民権を得はじめていた．国内でもコンピュータシステムの大規模化，グローバル化に伴い，電力などの分野を中心に，システムの 24 時間無停止連続運転やオンライン拡張が求められ

てきた．こうしたなかで筆者らは 1991 年，無停止連続運転とオンライン拡張を可能とする小型・高速無停止 TPR（Triple Processor check Redundancy）方式を開発し，HITAC FT-6100（図 1･15）[8]として量産開始した．

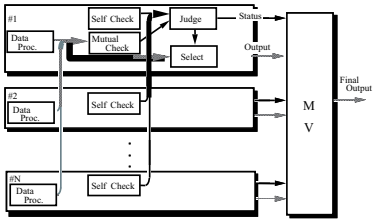


図 1･13 SNV **方式のためのシステム構成**

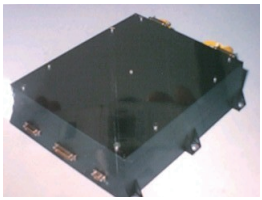


図 1･14 「ひてん」搭載コンピュータの外観

図 1･15 HITAC FT-6100 **の外観**

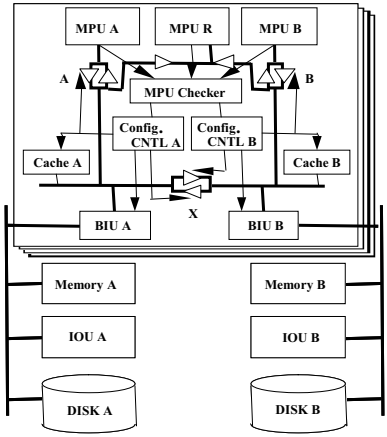
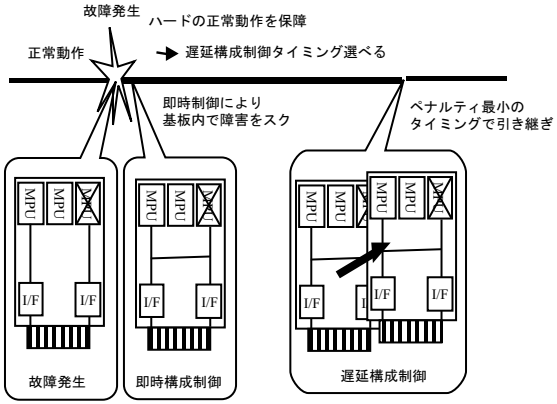


図 1･16 TPR **アーキテクチャ**

この方式では，図 1･16 に示すように小型化と高速のために CPU（Central Processing Unit）ボード内に高密度実装された三つのプロセッサが同一の処理を実行し，一つのプロセッサに障害が発生しても図 1･17 に示すようにこれを瞬時に切り離し，残り二つのプロセッサで処理を継続する（即時構成制御）．また CPU ボードのマルチ化により，障害 CPU ボードのジョブをほかの CPU ボードに移動し，オンラインで障害 CPU ボードを保守・交換できるようにした（遅延構成制御）[9]．このように即時構成制御と遅延構成制御を組み合わせ，CPU を一つのボードに高密度実装することにより，小型化，高速プロセッサによる処理性能の向上とフォールトカバレッジの向上とを両立させることができる．ここで培われたディペンダブル技術は日立クリエイティブサーバ 3500 のフォールトトレラントモデルである 3500/FT（図 1･18）にも継承されている．

更に国内では，ソフトウェアのバグに起因する障害回避にも考慮した SURE SYSTEM 2000 が開発されている[10, 11]．



即時構成制御：ハードによる単純で応処置的な構成制御
ハードの動作を保障する．

遅延構成制御：タスク引き継ぎなどの高度で複雑な構成制御
ソフトで実現しハード簡略化

図 1·17　即時構成制御と遅延構成制御

図 1·18　3500/FT **の外観**

■参考文献

1) J. Losq, “A highly efficient redundancy scheme: self-pursing redundancy,” IEEE Trans. On Comput., vol.C-22, no.3, pp.290-297, 1976.
2) 金川, 外, “新しい多数決方式によるフォールトトレラントコンピュータシステム, ”電子情報通信学会論文誌, vol.J72-DI, no.2, pp.109-116, 1990.
3) A. Avizienis et al., “The STAR (Self-Testing And Repairing) computer: an investigation of the theory and practice of fault-tolerant computer design,” IEEE Trans. On Comput., vol.C-20, no.11, pp.1312-1321, 1971.
4) T. Takano, et al., “In-orbit experiment on the fault-tolerant space computer aboard the satellite “Hiten”,” IEEE Trans on Reliability, vol.45, no.4, pp.624-631, 1996.
5) J.A. Katzman, “A Fault-Tolerant Computing System,” The 11-th Hawaii International Conference on System Science, 1978.
6) J.F. Bartlett, ”A ’NonStop’ Operating System, ”The 11-th Hawaii International Conference on System Science, 1978.
7) O. Serlin, ”Fault-Tolerant Systems in Commertial Applications, ”IEEE Comput., pp.19-30, 1984.
8) コンピュータ博物館, 日本のコンピュータ, UNIX サーバ, (日立) HITAC FT-6100 http://museum.ipsj.or.jp/computer/unix/0008.html
9) “基板内フォールトマスキング方式によるフォールトトレラントコンピュータの高速化と透過性, ”電気学会論文誌, vol.114-D, no.9, pp.903-909, 1994.
10) コンピュータ博物館, 日本のコンピュータ:オペレーティング・システム ... http://museum.ipsj.or.jp/computer/os/fujitsu/0014.html
11) 黒羽法男, 他, “耐故障性を持った商用の並列システム　SURE SYSTEM　2000 の OS“ SXO ”, 情報処理, vol.36, no.8, pp.758-763, 1995.

1-4-2 フェイルセイフシステム

（執筆者：中村英夫）[2009 年 3 月 受領]

(1) フェイルセイフの概念

人間がかかわる機器やシステムにおいては，万一故障や障害が発生しても，安全な状態は維持することが望まれる．システムが取り得る入力誤りや，構成要素の任意の故障に対し，システムとして安全側の状態が維持される性質をフェイルセイフ（Fail-safe）といい，そのための技術をフェイルセイフ技術という．

一方，安全側といってもシステムの使命や機能，状態によって多様である．例えば，鉄道においては，一律に停止が安全側とされてきたが，西日本トンネルの火災後の実験を経て，トンネル内の列車火災時には，トンネルを脱出することが安全ということで，指導が変更された．航空機も，飛行を継続することが安全とされる．このように，安全側が異なるため，このためのフェイルセイフ技術も業界やシステムにより，多様な意味でとられている．例えば，飛行中の航空機におけるフェイルセイフは故障をマスクしたり予備系を備えたりして達成される．一方，産業機械においては，万一のときにはシステムを停止させることで，安全を確保するが，このためのフェイルセイフ技術は必ずしも，多重系ではない．

このような産業界における技術の多様性を反映し，表 1･1 に示すようにフェイルセイフの定義にもばらつきが見られる．このため，フェイルセイフという用語すら，規格などで使用することは慎重になされている．

表 1･1　フェイルセイフの多様な定義

朝日現代用語 ：知恵蔵 1994	構造の部材の一つに疲労亀裂などが生じても，直ちに全体の致命的破壊に進まないようにする設計の考え方．実際には，多くの部材に加重を分担させたり，二重構造を用いる．いずれも適切な点検で欠陥を発見・修理するのが前提
電子通信ハンドブック（電子情報通信学会）	素子に故障が生じても誤動作したり危険な出力を生じない
電子用語辞典（日刊工業新聞社）	何か故障が起きたとき，大事故にならないよう電気回路あるいはシーケンス回路またはシステムを配慮してあること
JIS (W 0108 −1976)	一部の部材が壊れても構造全体は安全で，その設計荷重の特定の比率の荷重を一定期間は受けもつことができる構造
JIS (Z 8115－1981)	アイテムに故障が生じても安全性が保持されるように配慮してある設計
朝日新聞社説 (1987.6.20)	一つの機能に故障や不具合があっても，ほかの装置が作動し全体の安全が保てる仕組み
読売新聞社説 (1985.9.15)	多重安全構造
フリー百科事典『ウィキペディア（Wikipedia）』（2009. 3.30）	何らかの装置，システムにおいて，誤操作，誤動作による障害が発生した場合，常に安全側に制御すること．またはそうなるような設計手法で信頼性設計の一つ．これは装置やシステムは必ず故障する，あるいはユーザは必ず誤操作をするということを前提にしたものである

(2) フェイルセイフ論理

しかし，論理回路におけるフェイルセイフの仕組みについては学問的に明確に定義され，フェイルセイフ論理として確立している．フェイルセイフ論理は，我が国の研究者が鉄道の保安装置の解析を行うなかで，1960 年代半ばに明らかにしたものである[1)]．

その考え方を，事例をもって説明する．図 1･19 に示す踏切制御の概念図において危険な事態は「列車が来ても踏切が遮断されない」，「列車通過中に遮断棹が上がる」といったケースである．踏切では列車接近を検知すると警報し，一定時間後に遮断する．進入側センサの無検知や，時間計測回路 DELAY の 1 側故障は無遮断につながるし，これらの条件を判断するリレー（A）の 1 側故障も許されない．一方，進出側センサが列車有りと誤検知すると通過前に遮断棹が上がりかねず，これまた危険である．このため，進入側の列車センサには 0 側（列車あり）にしか誤らない（0 側非対称誤り）センサを，また進出側には逆に列車なしとしか誤らないセンサをそれぞれ用いて安全を保っている．このように，フェイルセイフ論理系では，非対称な誤り特性が重要な意味をもつ．

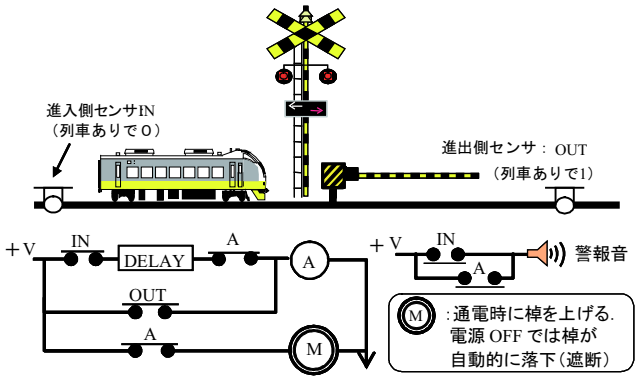


図 1･19　踏切制御システムの概念（Simplified concept of a level crossing system）

次に，回路の特性としての「フェイルセイフ論理回路」を考察する．回路出力の 0 側誤りは安全と仮定しよう．例えば，図 1･20 において出力段から安全側を割り当てていく．出力端の AND 素子は 0 側非対称誤り素子を使えばよい．このようにして，入力まで安全側を割り当てていくと，A には 1 側非対称誤り特性を有した入力を，B,C には 0 側非対称誤り特性をもつ入力を使い，回路素子には非対称誤り素子を使うことにより，回路としてのフェイルセイフ性が達成できることが分かる．

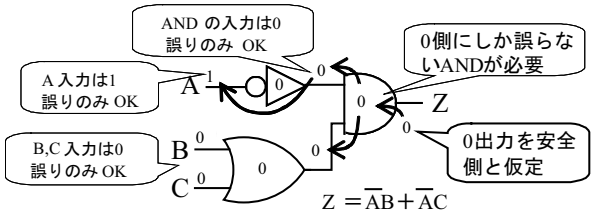


図 1･20　フェイルセイフな回路（An example of Fail-safe circuits）

一方，図 1･21 に示した回路は，排他的論理和を実現する回路である．出力から安全側状態を割り当てていくと， NOT 素子からは「1 に誤れ」，AND 素子からは「0 に誤れ」というそれぞれ矛盾した要求が入力になされるので，対応できない．事実，入力 A, B の故障は本来 0 出力のときに危険側の 1 に誤らせる可能性があり，フェイルセイフにできない．

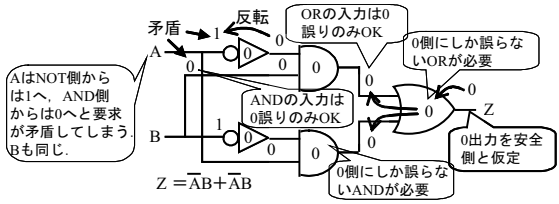


図 1･21　フェイルセイフでない回路（An example of non Fail-safe circuits）

これらの根本的な違いはどこに起因するのであろうか．回路を表現した論理式において，それぞれの入力変数が負論理，もしくは正論理の一方のみで構成される回路と，そうでない回路がある．前者の回路をユネイト（Unate）な回路と呼んでいるが，「回路がユネイトなら非対称誤り論理素子を用いてたかだか一重系でフェイルセイフ回路が構成できる」．この知見を基に，フェイルセイフ論理の研究は広がり[2)−4)]，「ユネイトでない回路のフェイルセイフ化」や，「対称誤り素子によるフェイルセイフ回路の構成方法」[3)]，「非対称誤り論理素子の開発」[4)]などの研究成果が相次いで発表された．

ところで，ユネイトな回路の論理式で，負論理の入力変数の否定形を，あらためて一つの入力とすると，すべての入力変数は正論理となり，回路も否定素子を含まずに構成できる．このような論理関数を「正論理関数」と呼ぶ．正論理関数で表現される回路は典型的なユネイト回路であるが，この正論理関数を基本に，フェイルセイフの定義を行う．

まず，入力集合をベクトルとして表現し，ベクトル間に大小関係を定義する．

ベクトル，$A=(a_1,a_2,\cdots,a_n)$，$B=(b_1,b_2,\cdots,b_n)$ が存在し，任意の要素間に，$a_i \le b_i$ が成り立つとき，$A \le B$ とする．

いま，関数 $f(X)$ について，二つの入力ベクトル A,B が，$A \le B$ であるとき，

$f(A) \le f(B)$ であれば，$f(X)$ を単調増加関数

$f(A) \ge f(B)$ であれば，$f(X)$ を単調減少関数

という．

次に，回路入力の誤りを定義する．

入力ベクトル $A=(a_1,a_2,\cdots,a_n)$ において，誤った入力を A^*, $A^*=(a_1^*,a_2^*,\cdots,a_n^*)$ とする．フェイルセイフ論理回路の安全側を 0 と仮定すると，任意の入力誤りが安全側であるということは，$A \ge A^*$ が成立することにほかならない．一方，任意の故障を含む回路を $f^*(X)$ とすると，この回路 $f(X)$ がフェイルセイフであるということは，$f(A) \ge f(A^*)$ がまず成立せねばならず，回路は単調増加関数であることが要求される．もちろん，正論理関数は単調増加関数である．そして，フェイルセイフであるためには回路の故障に対しても $f(A) \ge f^*(A)$ であることはもとより，$f(A) \ge f^*(A^*)$ も成立せねばならない．

この関係を満足するフェイルセイフ論理回路の構成方法を一般化すると，前述した「回路がユネイトなら非対称誤り論理素子を用いてたかだか一重系でフェイルセイフ回路が構成できる」が導出できる．

その後の研究により，ユネイトでない回路に対しては，1 を (1,0)，0 を (0,1) という 2 ビットで表現する二線論理系が提案された．二線論理系の NOT は線を交差させればよい．したがって，ユネイトでない論理回路（AND,OR,NOT で表現できる）も，二線論理系で実現するなら，入力要素の正論理展開で構成されるため，フェイルセイフにできることが分かった．

更に，二線論理系では (1,0)，(0,1) という正規な情報のほかに (1,1) や (0,0) という誤り情報が定義できる．同時故障がないという前提は必要であるが，誤り情報が検出されたときに出力を安全側に制御すれば，非対称誤り論理素子でなく汎用の素子でフェイルセイフ回路が構成できる．この考え方は，正規な 1,0 のほかに誤り状態値を定義した 3 値フェイルセイフ論理系に発展する[3]．更に，二線論理系は情報を 2 ビットで表現したが，n ビットの符号語中に必ず k ビットの 1 をもつような符号を利用した k-out-of-n 符号語によるフェイルセイフ論理回路も考えられる．

(3) フェイルセイフコンピュータ

非対称誤り素子によるフェイルセイフコンピュータの開発は，フェイルセイフ論理の創成期 1970 年ごろにチャレンジされ成功している[5]．しかし，汎用コンピュータの急成長もあり，経済性の隘路（あいろ）から普及するには至らなかった．

今日のシステムがコンピュータや通信技術を要素として構築される以上，フェイルセイフコンピュータの実現は重要な課題であった．

安全に関係する応用分野に使えるフェイルセイフコンピュータの開発は 1980 年代に入り相次いで成果をあげた．ここでは，故障やコンピュータの異常を確実に検出することがまず第一とされる．更に，対応する応用分野により，別系に切り替えて機能を維持したり，安全側に停止させたりする処置が取られている．その方法論にもフェイルセイフ論理ではない様々な工学的なアプローチが見られるが，その多くは，複数の MPU の処理を照合し，不一致を早期に検知して安全側に制御する機構の組込みが一般的である．そのなかには，符号語と非符号語をもとに高信頼化を図ったフォルトトレランスの概念[6]に符合するものもある．また，今日では一つの FPGA のなかに二つの MPU とフェイルセイフな照合部を組み込んだフェイルセイフ RISC プロセッサも実用化されている[7]．

(4) システムのフェイルセイフ

ハード的には解が得られたとはいえシステムのフェイルセイフには課題も多い．ソフトウェアの誤りはコンピュータ制御では致命的である．様々な高信頼化プログラミングへのチャレンジがなされた．決定的な方法論が定まらないなか，国際規格（IEC61508）がソフトウェアを含むシステムの安全性設計に一つの方向性を示した．

IEC61508 では，システム開発の上流工程で RISK 解析を求めている．RISK 解析では，システムが有する様々なイベントに対し，致命度や生起頻度をもとに RISK 評価を行うことになるが，その作業が漏れなく実施されるよう，FTA や ETA あるいは FMEA などを行うことを推奨している．根源的な RISK レベルが明らかになると，実用時に望まれるリスクレベルまで，何らかの安全技術を適用して RISK を低減することになるが，この当初の RISK レベルと受け入れ時の RISK レベルのかい離の度合いが安全度水準（SIL:Safety Integrity

Level）の根拠となる．安全度水準が定まると，その水準に応じて配慮されたつくり方が，システムのライフサイクルの各フェーズでなされていることを第三者に認証してもらうことで安全性を立証しようという考え方が，規格の基本的精神である．したがって，IEC61508 はプロセス規格とも呼ばれ，現時点ではソフトウェアの妥当性を評価する唯一の方法論ともいえる．もちろん，フェイルセイフシステムの妥当性検証や評価にも応用できる．

しかし，実際のシステムは，人間とのかかわりも多いほか，ネットワークなどを介して他システムと結合され次第に肥大化していく状況にある．このようななかでは，当初設計時に想定されたシステムの環境を越える利用がなされたり，新たな RISK もつくり込まれたりするため，システムのフェイルセイフ性確保には厄介な状況がある．異なった使い方や，システムの環境が変わる際には，ユーザを含め常に故障や障害の影響の分析，そしてそれらへの的確な対処などを漏れなく行う安全文化の醸成が必要になる．この面での産業界における今後の展開が注目される．

■参考文献

1) 渡辺，高橋, “ フェイルセイフ形論理系の一般法, ” 信学全大，vol.72，Nov. 1965.
2) H. Mine and Y. Koga,“ Basic properties and a construction method for a fail-safe logical systems, ” IEEE Trans. Electronic Computer, vol.EC-16, no.3, pp.282-289, June. 1967.
3) 当麻, 大山, 坂井, “ 対称誤りを考慮したフェイルセイフ順序回路の一構成法, ”信学論（D）, vol.55-D, no.3, pp.202-209, Mar. 1972.
4) 土屋, “ 3 値 C 型フェイルセイフ論理回路, ” 計自学論, vol.6, no.1, pp.81-88, Feb. 1967.
5) 奥村, “ 鉄道信号用電子連動装置のフェイルセイフ構成に関する研究, ” 鉄研報告，no.1002, 1976.
6) W.C. Carter, et al.,“ Cost effectiveness of self-checking computer design, ” Dig. Paper, vol. FTCS-7, pp117-123, June. 1977.
7) 高橋, 中村, 星野, 三枝, 平, “ フェイルセイフプロセッサのシステム LSI 化, ”信学技報 FTS2001-75, 2001.

1-4-3 組込みシステムへの適用例

（執筆者：金川信康）[2009 年 4 月 受領]

（1）チップ内冗長化

ムーアの法則[1)]は提唱されて以来，様々な技術的限界から何度も限界説が出てきたが，半導体の微細化は留まるどころかむしろ数々の技術革新により加速されてきている．半導体プロセスサイズが 0.1 μ m(100 nm) を切るようになると，μ m であった単位がいつしか nm になり図 1･22 に示すように年を追うごとに 90 nm，70 nm，45 nm と微細化が進んでいる．このような微細化により，臨界電荷量（データの反転を引き起こすために必要な電荷量）減少，電源電圧の低下によりソフトエラー（シングルイベントアップセット）と呼ばれるデータエラーが発生しやすくなってきている[2)]．またそれと同時に，1 チップ内に収納できる論理規模が増大し，システム全体を一つのチップで構成できるどころか，一つのチップに複数のシステムを構成できるところまできている．マイクロプロセッサでは，一つのチップ内に複数のプロセッサコアを構成するマルチコアプロセッサが登場している．

高信頼化技術の分野では，一つのチップ内にもたせた複数のプロセッサに同一の処理をさせ，冗長系を構成させることにより異常を検出しようとするチップ内冗長化が検討されている[3)]．

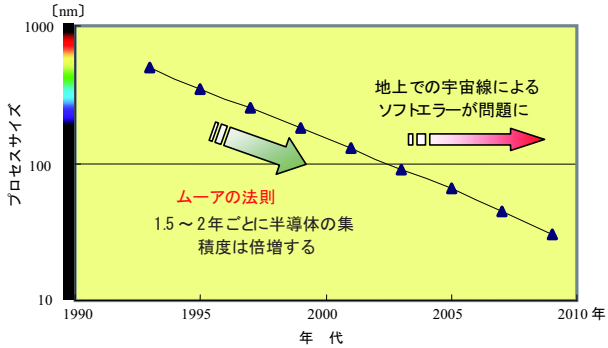


図 1·22　半導体プロセスの微細化の流れ

こうした流れのなかで筆者らは，まずプロセッサの設計資産を異なるプロセスに移植するためのシンセサイザブル（ソフト IP）コア化技術を確立し，この技術を活用して二重化したプロセッサ，比較器などをゲートアレイ上に実装した FUJINE（図 1·23）を 1999 年に試作した[4]．更に，2006 年には浮動小数点演算機能やキャッシュメモリ，2 プロセッサによる並列処理などの機能を追加した FS-CPU（フェイルセイフ CPU）（図 1·24）を開発し[5]，鉄道信号システムの標準部品として種々の製品に適用を推進中である．

図 1·23　チップ内冗長化プロセッサ (FUJINE) **外観**

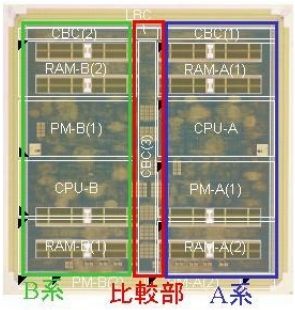


図 1·24　チップ内冗長化プロセッサ (FS-CPU) **外観**

(2) 新しい応用分野：X-by-Wire

X-by-Wire とは航空機をコンピュータで制御しようとする Fly-by-Wire を自動車に適用した技術であり，自動車の制御を電子化することで，より複雑な制御を可能とし，車両運動の安定化などによる安全性，運転特性の向上などを図るものである[6]．ブレーキを制御する Brake-by-Wire，ステアリングを制御する Steer-by-Wire などがある．

著者らは，自律分散を適用した X-by-Wire による横滑り防止装置を試作し，机上検討を進めている．図 1·25 に示すように，一般産業分野に自律分散を適用することにより，演算部をほかと代替することができ，演算部の高信頼化が可能となる．更に自動車や航空機など

の制御対象が大域的である分野に適用すれば，制御対象に密接に結びついている I/O 部の高信頼化を図ることができる．例えば，4 輪のうち 1 輪のブレーキが故障した場合も，ほかの 3 輪のブレーキを用いて制動することが可能となる．この場合，故障した側全体の制動力が不足し，いわゆる片効き状態となり車体が回転しようとするが，これを横滑り防止装置のセンサで検知することで防止することができる．別の見方をすれば，元来制御対象に備わっている冗長性を利用して，更なる冗長化をせずに I/O 部の高信頼化を図ることが可能となる．

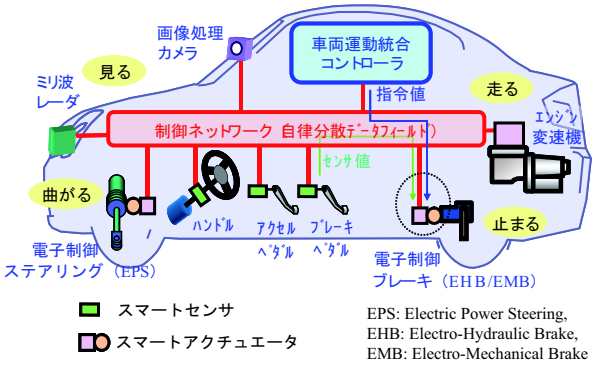


図 1·25 X-by-Wire **システム**

また自動車への応用で第一の問題になるのはコストであるが，宇宙，航空，鉄道分野に比べてけた違いに大きな量産規模をもつことが分かる．したがって，特に LSI 化による量産効果により，コストダウンを図ることが期待できる．そこで，チップ内冗長化，セルフチェッキング／フェイルセイフ技術，最適クロックダイバーシティなどの LSI 技術と親和性の高いディペンダブル技術が重要となる．

■参考文献

1) Moore, G.E., “Cramming more components onto integrated circuits,” Electronics Magazine, vol.38, no.8, pp.114-117, 1965.
2) T.J. O’Gorman, et al., “Field testing for cosmicray soft error in semiconductor memories,” IBM J. of R D, vol.40, no.1, pp.41-50, 1996.
3) N. Kanekawa, et al., “Self-Checking and Fail-Safe LSIs by Intra-Chip Redundancy,” Proc. FTCS-26, pp.426-430, 1996.
4) K. Shimamura, et al., “A Fail-Safe Microprocessor Using Dual Synthesizable Processor Cores,” AP-ASIC, pp.46-49, 1999.
5) Shimamura, et al., “A Single-Chip Fail-Safe Microprocessor with Memory Data Comparison Feature,” PRDC 2006, pp.359-368, 2006.
6) 植木, 外,“ 安全走行支援システムを支える自動車運動制御技術, ”日立評論, vol.86, no.5, pp.379-384, May. 2004.

■6 群 - 7 編 - 1 章

1-5 ディペンダビリティの新展開

（執筆者：南谷　崇）[2009 年 3 月 受領]

1-5-1 背　景

2001 年 9 月 11 日に米国で起きた同時多発テロ以降，「これまでは想像にすぎなかったことが今は現実であり，これまでは思いも及ばなかったことが今は想像できるようになった」との認識が世界に広がった．その結果，電力網，都市交通網，物流網，金融網，企業基幹網など，社会の重要インフラへの関心が高まり，ネットワーク化された情報システムの安全性，信頼性の保障が各国の政策レベルにおける最重要かつ喫緊の課題となっている．欧州と米国ではネットワーク化情報システムのディペンダビリティとセキュリティが重要な科学技術政策上の課題として取り上げられており，欧州連合の EC（European Commission）と米国の NSF（National Science Foundation）, DHS（Department of Homeland Security）が連携して今後投資すべき重要な研究領域を政策担当者と研究者が協議するための非公開会議 EU-US Summit Series: Workshop on System Dependability and Security が 2006 年 11 月にアイルランド・ダブリンで，2007 年 4 月にイリノイ州アバナ・シャンペーンでそれぞれ開催された．我が国においても科学技術振興機構・研究開発戦略センターが，2006 年 12 月に戦略プロポーザル「ニューディペンダビリティ宣言―情報化社会の安全と信頼を担保する情報技術体系の構築」を，続いて 2007 年 12 月に戦略プロポーザル「情報社会のディペンダビリティ―情報技術の目指すべき目標理念」を，それぞれ政策提言として発行した．また日本学術会議情報学委員会に設置されたセキュリティ・ディペンダビリティ分科会では 2008 年 7 月に提言「安全・安心を実現する情報社会基盤の普及に向けて」を公表した．

1-5-2 情報社会のディペンダビリティ

情報社会のディペンダビリティとは，情報社会が直面する様々なリスク要因の存在にもかかわらず，人と組織が社会インフラや情報環境から提供されるサービスの安全性と信頼性に揺るぎない確信をもち，その良質なサービスに依拠して安寧な生活と十全な活動を展開できる，という情報社会の属性をいう．人と組織のあらゆる活動が情報システム・ネットワークに依拠する情報社会では，ディペンダビリティは最高の価値であり，情報技術の研究開発が目指すべき普遍的な目標理念であるといえる．

これまでの情報技術の研究開発は，ムーアの法則に象徴されるように，ひたすら高速化，大容量化，高集積化，高機能化，低電力化という「性能向上」をシステム開発の目標として追求してきた．その結果，情報システムは社会のなかに深くかつ複雑に浸透し，人と組織のあらゆる活動がネットワーク化された情報システムに依拠する社会，すなわち情報社会が出現した．情報社会では，情報システムと社会システムの間に明確な境界線を引くことはできず，システムの開発・実装に当たっては情報技術と社会技術とが一体となって機能する．このため人間と社会と技術が複雑に絡み合う多種多様なシステムやパラダイムが次々と出現し，それらの無秩序な融合，拡張が地球規模で進行している．

こうした状況で今後の成熟した情報社会を展望すると，今後情報技術の研究開発が目指すべき方向は，従来のような「性能向上」の追求ではなく，「情報社会のディペンダビリティ」の追求であることが分かる．「情報社会をよりディペンダブルにする情報技術」，あるいは「情

報システムをよりディペンダブルにする情報技術」はいまだ必ずしも市場では広く認知されるに至っていないが，近い将来，情報社会の根幹を支える最高の付加価値を有する技術となり，情報技術体系の新たな発展・進化の駆動力になるとともに，国際競争力強化の源泉を創出することが予想される．

実際，今日の情報社会は以下に示すようにディペンダビリティを阻害する様々なリスク要因に直面している．

(a) ブラックボックス化

人と社会のすべての活動が依存しているにもかかわらず，情報システムの全容を誰も把握できない．国の重要インフラの相互依存性を含めて，ネットワーク化された情報システムで今何が起きているのか，将来何が起き得るのか，誰も把握できない．

(b) システムの複雑化・巨大化

システム仕様の不明確化，設計ミス，操作ミス，経年劣化に加えて，複雑に絡み合ったオープンシステムの相互作用，人間とシステムの予期せぬ相互作用が潜在している．このため大規模で複雑化する情報システムの 「設計・実現の正しさ」,「運用・保全の確かさ」に確信がもてない状況になっている．

(c) VLSI の微細化

ハードウェア中枢である VLSI の微細化が進み，リーク電流，発熱，プロセスパラメータ変動，性能変動，ソフトエラー，クロストーク，IR ドロップなど，これまで見られなかった物理現象が，VLSI をハードウェア基盤として構築されている巨大な情報システムの安全性，信頼性に対する深刻なリスク要因になりつつある．

(d) 情報量の爆発的増加

グローバルなネットワーク上で毎日大量に生産され，蓄積され，加工され，流通し続ける巨大な情報量は，人間の処理能力をはるかに超え，その大部分は人間に直接アクセスされることが永久にない．提供される情報の大部分は，検索され，要約され，優先順位づけられた結果である．この情報量の爆発が「情報の安全性，信頼性」に対する脅威であり，情報社会のディペンダビリティ／セキュリティに対する深刻な阻害要因である．

(e) サービス利用の多様化

日々進化し，変貌し続けるネットワーク化システムには無邪気な初心者ユーザも悪意の練達ユーザも等しくアクセス可能であり，また多様なサービスが生まれている．そこに悪意か過失かにかかわらず人為的フォールトの生ずる可能性がユビキタスに存在し，日常的にシステムを障害，不正侵入，情報漏えいの危険にさらしている．更に世界規模でサイバーテロの脅威が常に存在する．

(f) システム要素の経年劣化

社会の至る所に埋め込まれた膨大な数のハードウェア要素の物理的な経年劣化に加えて，その上に構築された巨大システムの動作環境変化やシステム更新によって，関連するソフトウェア要素の論理的な経年劣化が潜在的に進行し，その相互依存関係を十分に把握できない状況が生まれている．

(g) 責任所在の不明確化

自律的に発達してきたグローバルなインターネットは，互いに独立に設計された巨大な数の「システムのシステム」であり，明確に定義されないまま拡大を続けている．その結果，

ネットワーク化システムで起きた事象，あるいは将来起き得る事象に対する責任の所在が明らかでない．

こうした状況で，従来別々の研究コミュニティを形成してきたディペンダビリティとセキュリティを融合する必要性が世界的な共通認識になりつつある．

ディペンダビリティ（Dependability）とは「提供するサービスが良質で信頼でき，人間と社会の活動が安心してそれに依拠できる」という情報システムの属性である．自然現象，経年劣化，設計ミス，操作ミス，システム不整合など，予測不能で偶発的に生ずる物理的，人為的な障害原因（フォールト）の存在を前提として，情報システムの可用性（availability），信頼性（reliability），安全性（safety），完全性（integrity），保全性（maintainability）を総合した概念として定義される．国際規格 IEC 615081 に定められている電子情報システムの機能安全（functional safety）の概念も包含する．また情報システムの属性概念から拡張し，広く社会システム，情報社会の属性を表す概念としても用いられる．

一方セキュリティ（Security）とは「情報をその生産者，運用者，利用者（あるいは社会）が合意した意図のとおりに利用できることを保障する」という情報システムの属性である．悪意による意図的な不正アクセス，不正侵入の存在を前提として，情報システムの可用性（availability），完全性（integrity），機密性（confidentiality）を総合した概念として定義される．

ディペンダビリティとセキュリティはもともと，可用性と完全性の概念を共有するが，情報社会の安全信頼保障を考えるとき，以下の理由でディペンダビリティとセキュリティの技術を融合させることが重要である．すなわち，従来のセキュリティ研究は「秘密情報（鍵）は適切に管理されている．情報システムは正しく実現されている」という仮定の上に成り立っているが，システム操作ミスによる秘密情報漏えいの可能性，仕様ミス，設計ミス，実装ミスによる脆弱性発生の可能性は情報システムに常に存在しているため，セキュリティ確保にはディペンダビリティ技術が必要である．一方，従来のディペンダビリティ研究は「フォールト（物理的，人為的）の発生は偶発的である」という基本的仮定の上に成り立っているが，実際には悪意ある人為フォールト（侵入，改ざん）に偶発性はないので，ディペンダビリティ確保にはセキュリティ技術が必要である．このため，上述の戦略プロポーザル「ニューディペンダビリティ宣言―情報化社会の安全と信頼を担保する情報技術体系の構築」では，ディペンダビリティとセキュリティの技術概念を社会システムのユーザ視点で総合する新しい概念を表す用語としてニューディペンダビリティが用いられている．

なお，フェイルセイフ（Fail-safe）は「システムに障害が生ずる場合，それは必ず安全な（許容可能な）障害である」，あるいは「提供するサービスに障害が生ずる場合，必ず安全な状態でサービスを停止する」という情報システム，社会システムの属性である．例えば，鉄道踏切の遮断機に障害が発生した場合，開いたままの状態で機能を停止すると危険であるが，必ず閉じた状態で機能停止になれば安全な障害であるといえる．システム障害の発生を回避できない事態になった場合の「最後の手段」を提供するシステム設計思想である．

1-5-3 研究分野

情報社会が，様々なフォールトや不正侵入の発生リスクの存在にもかかわらず，社会インフラ，情報環境から提供されるサービスの安全性と信頼性に確信をもつことができる「安全

信頼保障」を確立するためには，以下の四つの階層におけるディペンダビリティ設計が必要である．

(1) 社会基盤となるネットワーク化情報システム

情報システムでディペンダビリティ設計とシステム保全アーキテクチャが適切に実現されていれば，例えフォールトや不正侵入が発生しても，それがシステム障害となって現れることを回避できる．しかし，ディペンダビリティ設計が不適切，不完全な場合には，上位階層とのインタフェースにシステム障害となって現れる．

(2) ネットワーク化情報システムを活用して構築される社会の重要インフラ

下位階層の情報システムの障害は，上位階層である重要インフラにフォールトを生じさせることになる．また，重要インフラ自身もフォールト発生や不正侵入の可能性を内在させる．この場合，重要インフラのディペンダビリティ設計とインフラ保全アーキテクチャが適切に実現されていれば，これらのフォールト発生にかかわらず，重要インフラの障害を回避できる．しかし，ディペンダビリティ設計が不適切，不完全な場合には，上位階層とのインタフェースにシステム障害となって現れる．

(3) 重要インフラを活用して提供されるサービス・情報

重要インフラの障害は，それによって提供されるサービスあるいは情報にフォールトを生じさせる．また，サービス・情報自身にもフォールト発生や不正アクセスの可能性が内在する．この場合，サービス・情報のディペンダビリティ設計とサービス保全アーキテクチャが適切に実現されていれば，これらのフォールト発生にかかわらず，サービス・情報の障害を回避できる．しかし，ディペンダビリティ設計が不適切，不完全な場合には，上位階層とのインタフェースにシステム障害となって現れる．

(4) サービスを受ける人と組織が形成する情報社会

情報社会の階層で人や組織が受けるサービス・情報にひとたび障害が起きると，これを回避する手段はなく，社会は大きな被害を受ける可能性がある．この場合，障害のために期待するサービスが停止しても，必ず「安全側」でのサービス停止状態になることを保証する「フェイルセイフ」な社会システムが構築されていれば，被害を最小限度に留めることができる．また，適切な回復プラットフォームが構築されていれば，被害を最小限度に留めつつ，社会の活動や事業を要求されるレベルまで早期に回復させることができる．このようなフェイルセイフな社会システムが実現されていれば，社会の「安全信頼保障」に確信をもつことができる．

「情報社会のディペンダビリティ」を実現し，社会の安全信頼保障を恒久的に確立するためには，上に述べた四つの階層のそれぞれにおいて，

i. ディペンダビリティを恒久的に保証するアーキテクチャ
ii. ライフサイクル・リスクを想定した設計・保全技術
iii. ディペンダビリティの定量的評価技術

の総合的な基盤研究開発を戦略的かつ永続的に推進することが必要である．

具体的には，推進すべき研究開発課題は以下の 4 階層 12 課題に分類される．

(1) 情報システム

i. フォールトや不正アクセスの存在にかかわらず，実世界と相互作用するネットワーク化情報システムのディペンダビリティを恒久的に保証するシステム・アーキテクチャ

の研究開発

ii. 情報システムのライフサイクル・リスクを想定した一貫性のあるシステム仕様定義技術，設計技術，保全技術の研究開発

iii. ユーザ視点から情報システムのディペンダビリティを定量的に評価し，経済価値へマッピングする評価技術の研究開発

(2) 重要インフラ

i. フォールトや不正アクセスの存在にかかわらず，相互に依存し合う多様な社会重要インフラのディペンダビリティを恒久的に保証するインフラ・アーキテクチャの研究開発

ii. 重要インフラのライフサイクル・リスクを想定した一貫性のあるインフラ仕様定義技術，設計技術と防衛・保全技術の研究開発

iii. ユーザ視点から重要インフラのディペンダビリティを定量的に評価し，経済価値へマッピングする評価技術の研究開発

(3) サービス・情報

i. フォールトや不正アクセスの存在にかかわらず，ネットワーク化情報システム，社会重要インフラが提供するサービスと情報のディペンダビリティを恒久的に保証するサービス・アーキテクチャの研究開発

ii. サービス・情報のライフサイクル・リスクを想定した一貫性のあるサービス仕様定義技術，設計技術，保全技術の研究開発

iii. ユーザ視点からサービス・ディペンダビリティを定量的に評価し，経済価値へマッピングする評価技術の研究開発

(4) 情報社会

i. システム障害，サイバー攻撃，情報漏えい，自然災害などの発生に際して，被害を最小限度に留めるフェイルセイフな社会システム・アーキテクチャの研究開発

ii. 情報社会が将来にわたって直面するリスクを想定し，ディペンダビリティの実現を促進する一貫性のあるフェイルセイフな社会システムの設計・保全技術とその運用支援ツールの研究開発

iii. 情報社会のディペンダビリティを定量的に評価し，恒久的な安全信頼保障を科学的方法で確立する情報社会技術・システムの研究開発，政策提言，実装支援と人材育成

上記の 4 階層にわたる課題は従来の情報技術，情報システムの範囲内では解決が困難であり，社会科学，数理科学，認知科学，生物学などが融合する以下のような領域の研究の一層の推進が必要である．

1) 巨大データの取り扱い

複雑なシステム（自然，生命，精神，社会，経済，情報，工学）に関与する膨大で多様な要素から絶え間なく生産される巨大な量のデータの計測，処理可能形式への編集と構造化，データの分析と結果の統合，システムのモデリングと検証，システムモデルに基づく巨大量の計算と結果の可視化，データの信頼性保証，巨大データからの有用な知識の発見，データ

に基づく対象の制御と将来事象の予測など，巨大データの取り扱いに関する研究領域.

2) 人間の心理・行動の理解

不確定性を有するシステム（情報，経済，社会，政治，教育）の要素である人間の心理・行動の計測，心理・行動のメカニズムとダイナミクスの解明，人間を要素として含むシステムのモデリング，シミュレーションと検証，人間を含むシステムの制御と予測，教育と創造，システム評価方法など，人間の心理・行動の理解とそのシステム応用に関する研究領域.

3) 進化・変異・劣化への対応

自然，人間，社会，人工物，情報が時間の経過とともに変化する進化過程，変異過程，劣化過程の計測，要因分析，進化・変異・劣化のモデリングと検証，変化に応じてシステムを最適状態に保つ適応制御，生涯を通じた環境変化を考慮した最適システム設計，将来事象の予測，変異状態からの回復，システムの若返りなど，システムの進化・変異・劣化の理解と制御に関する研究領域.

4) システムの複雑性克服

マクロからミクロまで巨大なスケールギャップ，多層性，相互依存性，不確定性を有するシステム（情報，経済，社会，生物，自然）の振る舞いの理解と記述，抽象化と階層化によるモジュール分割とシステムモデリング，多層統合シミュレーションと形式的検証，多層統合アーキテクチャとシステム設計，実現，評価，リスク管理，システム保全，システム信頼性保証等，複雑性を克服するシステム設計に関する研究領域.

5) ユーザ視点のサービス

情報社会を前提に人間と社会に対して提供されるサービスの価値を計測し，結果を可視化させ，サービス品質を定量的に評価する技術，ユーザ視点からサービスを分析し，モデル化し，最適化し，実現する技術，サービスを効果的に創造し，普及させ，定着させる技術などに関する研究領域.

■参考文献

1) 戦略プロポーザル,“ニューディペンダビリティ宣言－情報化社会の安全と信頼を担保する情報技術体系の構築,”科学技術振興機構・研究開発戦略センター，Dec. 2006.
2) 戦略プロポーザル,“情報社会のディペンダビリティ,”科学技術振興機構・研究開発戦略センター，Dec. 2007.
3) “新興・融合分野研究検討報告書,”科学技術振興機構・研究開発戦略センター，Feb. 2009.