RC-012

系統間交換型誤り検出符号によるフォールトマスク冗長系の一提案

A Proposal of Fault-Mask Redundancy with Error-Detection Code Exchanged between Strings

岩井仁司
Hitoshi Iwai

1. 問題点

フォン･ノイマンは、ある機能を果たすモジュールが故障した場合でも、故障の影響を隠す方式として、三重多数決冗長系 TMR (Triple Modular Redundancy) を考案した[1]．同時に、彼は、この方式では多数決回路が故障した場合、その故障の影響は隠すことができないという問題も指摘していた（「ノイマンの多数決回路問題」と呼ぶ）．その後、ハイブリッド冗長[2]、セルフパージ冗長[3]、シフトアウト冗長[4]などが考案されたが、どれもこの問題を解決していない．近年になっても、文献[5]には「例えば、計算要素を三重化し、その結果の多数決を採って出力とするシステムの場合、多数決を採る機構の故障には耐えられない．したがって、多数決機構の故障率が計算要素に対して十分小さい場合のみ、システム全体を高信頼にすることができる」とあり、状況が変わっていないことが伺える．

本稿提案の方法は、多数決や再構成の機能や回路が一切ないにも関わらず、多数決冗長と同様な高信頼性化が可能である．多数決しないので、多数決回路の故障を心配する必要もない．

フォールトマスクという場合、システムから外部にエラー（誤りデータ）を出さないことだと言われている[1, 6:§1]．しかし、システムから外部に出力する際、伝送エラーや出力デバイスの故障喪失は完全に無くすことができない．このため、外部システムには、出力データの整合性をチェックし、可能であればエラーを訂正し、あるいは冗長関係にある複数のデータ出口からどの出口のデータを採用するか選択する機能が必要になる．これは“フォールトマスク”なシステムが存在するために最低限必要な機能であると考えられる．

ただし、さらに特殊な機能を外部システムに求めると、フォールトマスクという概念があいまいになっていく．例えば、外部システムに“データの多数決を行うこと”を求めると、並列動作三重冗長系は全て、"フォールトマスク”冗長系ということになってしまう．

そこで、本稿では、外部システムとして、次の２つの機能のみを前提とすることにした．

(1) 出力データの整合性チェック (可能であればエラー訂正)
(2) フォールトマスクシステムの複数のデータ出口の選択

本稿では、これら以外の特殊な機能は外部システムに要求しない、という前提でフォールトマスク冗長系を考案するものとする．

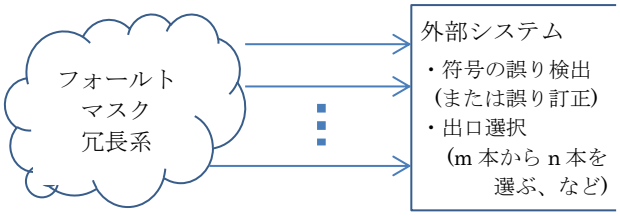


図１　フォールトマスク冗長系の出力先(外部システム)の前提機能

2. 提案のフォールトマスク三重冗長系の説明

2．１ 概要

まずは、フォールトマスク三重冗長系を提案する．多重多数決冗長系の基本は三重系である．四重系以上は、三重系の発展型に過ぎないからである．

本稿提案の方法は、ある機能モジュールのデータの正当性を証明するために、同じ処理を行う他の機能モジュールから誤り検出符号を入手して、当該機能モジュールのデータにも整合するかチェックして問題がなければ、当該機能モジュールのデータとその符号を組みにして、出力するものである．その後、外部システムに符号チェックしてもらうことで、伝送エラーがないか誤り検出するとともに、前記２つの機能モジュールのデータが一致することを認証させようというものである．

三重冗長系において、２つの機能モジュールのデータが一致するということは、その時のデータは三重冗長系の多数決結果と同じになるということである．

ところで、機能モジュールのデータはどれもエラーが無い限り同じなので、各機能モジュールで生成される符号はどれも同じになる．したがって、外部システムからみると、どの機能モジュールがデータの正当性を証明しているのかわからない．

このような問題を回避するために、データには、符号を生成した機能モジュールの ID 番号を付加することを提案する．そして、機能モジュール間で符号を転送する場合は一緒に ID 番号も転送し、データ整合性の検証に利用する．このようにすることで、符号も生成した機能モジュール毎にユニークになる．外部システム側で入力した時に、データに ID 番号も付いてくるので、符号を生成した機能モジュールが特定され、どの機能モジュールがデータの正当性を保証するのか判定できる．

符号は、通信用の誤り検出符号に限らず、暗号の分野で使われている認証符号とか署名データとかメッセージダイジェストとか呼ばれているものでもよい[7]．これらは誤り検出符号と本質的には同じものであるが、本稿で“誤り検出符号”に代表させる理由は、3.2 項で後述する．

2.2 構成と処理手順

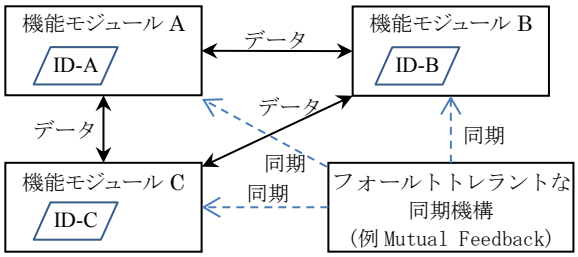
次に詳細について、構成と処理手順について説明する．

まず本稿提案のフォールトマスク三重冗長系に必要なシステム構成から説明する（図２）．

(1) ３つの機能モジュールと、各機能モジュールの出力を同期させるフォールトトレラントな同期機構（たとえば相互帰還法､Mutual Feedback [6: pp.126] [8]）からなること．

(2) 各機能モジュールは、自分の識別子（ID） を認識できること．これは ROM に焼きこんでも､DIP スイッチのような物のステータスを制御回路で読み込めるようにしてもよい．一方で、他の機能モジュールの ID は認識できない方が良い．

(3) 全ての機能モジュールはデータの交換ができるよう通信ラインで相互に接続されていること．



各機能モジュールは､自分の ID が読み込めること

図２ 前提とするシステム構成

次に処理手順について説明する．

(1) 各機能モジュールは、同じ制御周期で同じ処理を行う．各機能モジュールは、自分の出力データと自分の ID の組合せから整合性チェック用符号を生成する．（図 3 (a) フェーズ１）

(2) 生成した符号と各自 ID を、各機能モジュールは、相互に他の機能モジュールと交換する．（図 3 (b) フェーズ 2）

(3) 他の機能モジュールから符号と ID を受信したら、受信した ID と自分の出力データの組み合わせで、受信した符号を整合性チェックする．同じ操作をもう１組の ID と符号にも実施する．符号チェック結果(OK/NG)をメモリに記録する．このチェック結果は、次の周期のフェーズ 2（図 3 (b)） の処理（符号と ID を交換するかどうか）に使う．（図 3 (c) フェーズ３）

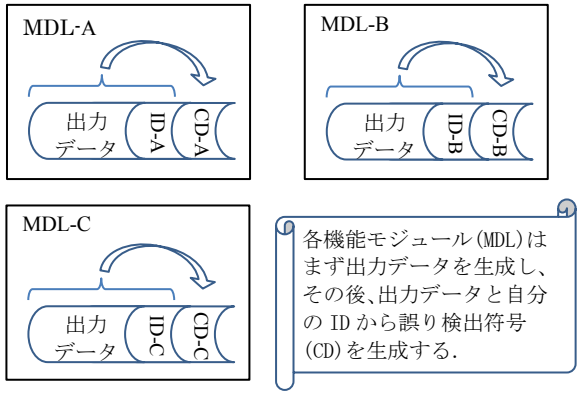
(4) 符号のチェック結果が OK の「ID＋符号」の組合せを選んで出力データと共に外部システムに送信する．OK な符号がなければ、符号や ID なしで送信する．（図 3 (d) フェーズ 4 の左側）

(5) 外部システムでは、符号により、「データ＋ID」の整合性にエラーがなければ､そのデータを採用する．エラーがある場合は、他のデータ出口を選択して、同様に符号チェックする．（図 3 (d) フェーズ 4 の右側）

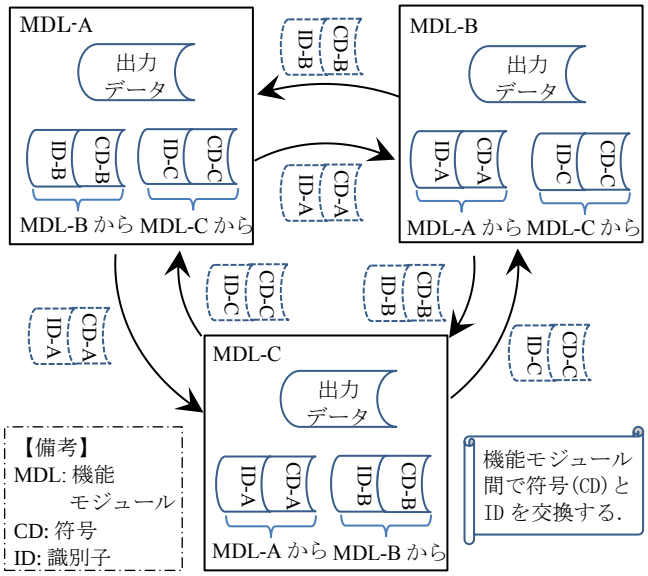
(6) ステップ（1）へ戻って周期的に処理を継続する．

本稿提案の方法で、重要なことは次の２点である．

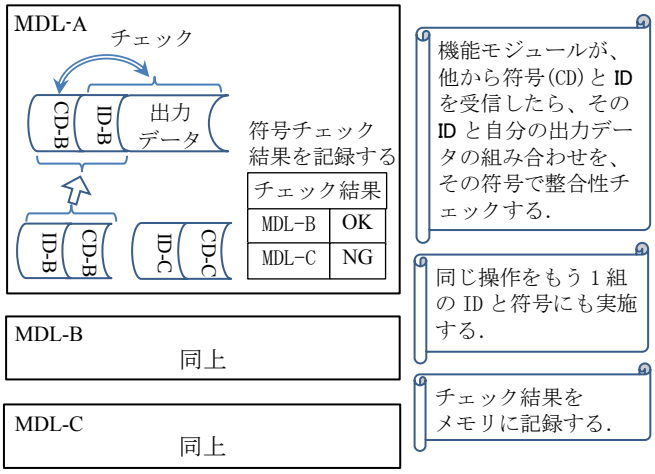
A. 自分で作った符号は、決して自分で使わないこと．
B. データ出力前に、自分で符号をチェックすること．



(a) フェーズ１：整合性チェック符号を生成する

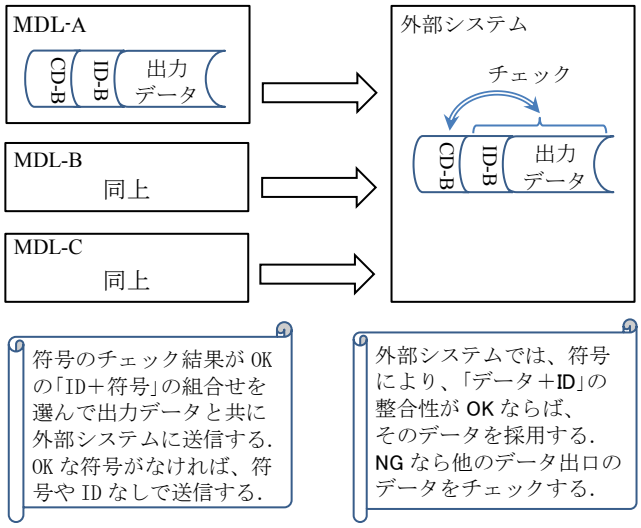


(b) フェーズ２：ID と符号(CD) を相互に交換する



(c) フェーズ３：外来 ID と自データを外来 CD でチェックする

図３ 処理手順の説明（次ページに続く）



(d) フェーズ４：「外来 CD＋外来 ID＋自データ」を外部システムに伝送し CD をチェックしてもらう

図３　処理手順の説明（前ページから続く）

項目 A に関して、注意すべき点は予期しない故障が発生した時に、機能モジュールの出力データに、同じ機能モジュールが生成した符号が付加されて、そのまま外部システムに出力しまうことである．そのような事象は、如何なる故障モードでもあってはならない．したがって、符号が生成する時のアドレス領域と、他の機能モジュールから伝送されてきた符号を格納して、チェックするためのアドレス領域は、異なることが望ましい．

図３では、(a) フェーズ１で ID と CD（符号）を出力データの右側に、(b) フェーズ２で ID と CD（符号）を出力データの左側に示しているのは、回路上のアドレスを分けるべきであるということを模式的に示している．

なお、機能モジュール内の機能フローは、4.1 項の適用例の検討で、別途説明する．

3. 考察

3.1 データ転送時間

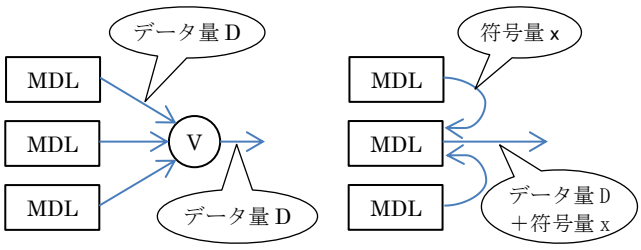
従来の三重多数決冗長系（TMR[1]）では、出力データを機能モジュールから多数決回路に転送して、更に多数決回路から外部システムに出力していた．データ量を D とする時データ転送時間は、(式１) になる （図４(a)）．

TMR のデータ転送時間＝２×D／転送速度　･･･（式１）

本稿提案の系統間交換型誤り検出符号によるフォールトマスク冗長系では、機能モジュール間では、誤り検出コードしか転送しない．機能モジュールから外部に出力する時は、データ＋誤り検出符号を出力する．合計のデータ転送量は、(式２) になる（図４(b)）．

系統間交換型誤り検出符号による冗長系の
データ転送時間＝（D＋2×x）／転送速度 ･･･（式２）

符号のサイズは、一般に元のデータサイズに比べて圧倒的に小さいので、本稿提案の系統間交換型誤り検出符号によるフォールトマスク冗長系は処理時間が、従来の TMR に比べて短くなる．



データ量を D とする時

転送時間＝
2×D／転送速度

(a) 従来の TMR

データ量を D、
符号量を x とする時

転送時間＝(D+2x)／転送速度

(b) 系統間交換型
誤り検出符号による冗長系

図４ データ転送時間の比較

3.2 信頼性と衝突率

系統間交換型誤り検出符号による三重冗長系の信頼性は、機能モジュール１つ当たりの信頼性を R とする時、（式３）となる．

＜系統間交換型誤り検出符号による三重冗長系の信頼性＞

$= R^3 + 3R^2 \times (1-R) = 3R^2 - 2R^3$ ･･･ (式３)

従来の三重多数決冗長系（TMR）では、多数決回路があるので、その信頼性が（式３）に掛かってくる．一方、系統間交換型誤り検出符号による冗長系では、誤り検出符号の衝突率というものがあって、たまたま誤りデータのビットパターンが誤り検出アルゴリズムで整合する場合がある．この衝突が起こった場合は、誤りデータの見逃しが発生する．これは機能モジュールが１つ故障している場合に発生するので、衝突率を考慮した場合のシステムの信頼性は、衝突率を c とする時、の（式４）となる．

＜衝突率を考慮した、
系統間交換型誤り検出符号による三重冗長系の信頼性＞

$= R^3 + 3R^2 \times (1-R) \times (1-c)$
$= R^2 \{ R + 3 \times (1-R) \times (1-c) \}$ ･･･（式４）

（式４）より R が１に近い時、(1－c) はあまり信頼性に効いてこないことがわかる．衝突率 c は、誤り検出符号の種類や符号の誤り検出符号の長さによって決まってくる．チェックサムや CRC では、誤り検出符号の長さが１バイトの時、衝突率 c は 1／256 である．R が１に充分近ければ、誤り検出符号の長さは１バイト程度でも充分かもしれない．一方、もっと小さな衝突率が要求される場合でも、国際的な安全規格 IEC 61508 などの一番厳しい信頼性要求が 10^{-9} 程度であることを考えると、衝突率 c への要求は 10^{-9} より厳しくならない．したがって、誤り検出符号の長さは４バイトあれば充分だと考えられる．

この１～４バイトという長さは、改竄防止のため用いられる認証符号とか署名データの長さというよりも、通信分野で用いられる誤り検出符号の長さである．系統間交換の符号として、認証符号とか署名データでも可能であるが、本稿で”誤り検出符号”という名称を採用する理由は、この符号の長さに由来する．

3.3 故障した機能モジュールの分離

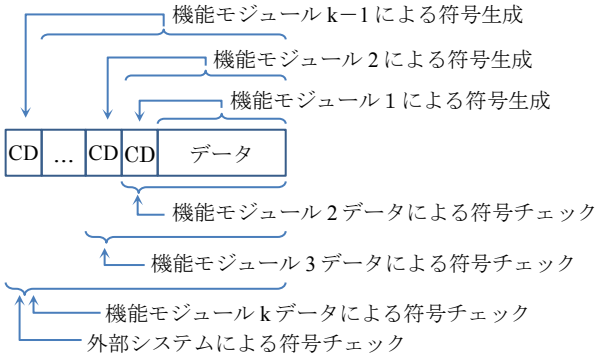
故障した機能モジュールを放っておくと、やがて他の機能モジュールも故障し、正常な機能モジュールも正常な符号を収集できなくなり、システム全体の機能を失ってしまう．したがって、外部のシステム（操作員も含めむ）は、各機能モジュールの故障有無を把握しなければならない．本稿提案の方法であれば、外部システムで各機能モジュールの符号をチェックすれば、その故障有無を把握することができる．

また、複数の機能モジュールが故障した場合、同じ故障モードに陥り、同じ異常な出力や符号を出すかもしれない．この対策のため、各機能モジュールに符号チェック結果テーブルを設け、整合しない機能モジュールを処理から分離することができる（図 3 (c) の符号チェック結果のテーブル）．分離された機能モジュールは、他から符号を集めることができないので、外部システムがそのデータを採用することはない．

3.4 n-フォールトマスク冗長系への拡張

以上は三重冗長系、即ち 1 故障までを想定したシステムである．これを n 故障まで、拡張する可能性について考えてみる．

まず、各機能モジュールが過半数の機能モジュールから認証された証拠を集めて、1 つの誤り検出符号にまとめることが難しい．外部システムが最後に 1 回符号をチェックすることで、過半数の機能モジュールがデータの整合性を保証していることがわかることが望ましいが、そのためには、1 つの機能モジュールの生成するデータに対して、過半数の機能モジュールが順に符号の追加と符号チェックを繰り返して符号の入れ子を作っていくことが考えられる（図５）．このような入れ子構造を作るには、機能モジュール間で通信を繰り返す必要があり、かつ機能モジュールの故障による無応答やデータロス、やり直しなどの例外的なシーケンスにも対応する必要があり、かなり複雑になる．



【備考】 k：機能モジュールの過半数
CD：符号

図５ 入れ子構造の符号

特殊な誤り検出符号を考案して、それを外部システムに解かせることも考えられるが、本稿の前提“外部システムに特殊な処理を要求しない”から外れる．

実用的には、”2 故障が同じ制御周期内で発生する”、という確率は一般に非常に低く無視可能と言われている．したがって、”2 故障同じ制御周期内で発生しない”という前提を付けても実用上問題ない．この論理からは、”2 つの機能モジュールの出力データが一致すれば、それらは正しい”という結論が自明的に言える．

あるいは、ハイブリッド冗長系[2]のように、n-フォールトマスク冗長系への拡張する場合、まず 3 つの機能モジュールだけ動作させ、残りの機能モジュールは待機させておき、故障で機能モジュールが 1 つ分離されるたびに、残りの 2 つの機能モジュールで、待機中の機能モジュールを動作モードに追加していく方法も考えられる．これならば、n 故障にも耐性を持ち、かつ、各機能モジュールは、系統間交換型誤り検出符号を他の機能モジュールから 1 つ集めればよい、ということになる．この場合、必要な機能モジュール数は、(n+2)個となる．

また、三重冗長系から、機能モジュール数を 1 つ減らして、二重冗長系も可能であると考える．この場合、2 個で相互に照合しあうことで符号語の信頼性を高めることができる．さらに、一時故障に対してリトライすることでエラーをマスクする．永久故障に対しては、ゼロ故障耐性となる． 4.1 (2) 項で別途検討する．

4. 適用例の検討

本稿提案の方法は、外部システムが確実に誤り検出符号をチェックすることが前提になっている．少なくとも、三重冗長系の部分に比べて、外部システムの信頼性が著しく低いと、システム全体の信頼性が損なわれる恐れがある．

一般に、出力側の外部システムは当然出力装置が構成要素になるであろう．出力装置は可動部を含むケースが多いだろうから、”外部システムの信頼性が三重冗長系部分に比して低くない”というのは、困難なハードルである．

しかし、それでも、充分有効な適用例があることを指摘したい．以下に、有効な適用例を挙げる．

(1) データと符号を一緒に媒体に記録するケース．
(2) 外部システムがセルフチェッキングシステムになっているケース
(3) 外部システムをループバックや定期的テストなどにより動作を保証するケース
(4) 一般的な分散システムでサーバ側に信頼性が要求されるケース（一般的なインターネット環境以外、特別なハードウェアを用いることに制約があり、リアルタイム性も要求されない場合など）

以下、適用できるケースについて、検討する．

4.1 データと符号を一緒に媒体に記録するケース

ここでは、機能モジュール 3 つのケースと、2 つのケースの適用例を考えてみる．

(1) 機能モジュール 3 つのケース

例えば利用者が IC カードのような持ち運び可能な媒体を持っていて、それに記録する場合を考えてみる．このような媒体はバックアップのような冗長性はないが、媒体から読み出す際に符号チェックすることにより、記録データの正当性を検証できる．通常の誤り検出符号では記録時のエラーしか検出できないが、本稿提案の方法によれば、記録する前の処理過程の誤りの検出ができる．利用者は、媒体からの読み出し時にエラーが出れば、やり直せばよい．

構成例を図６に示す．応用プログラムは出力データを生成する．その出力データと機能モジュール ID より、符号(CD)を生成する．符号(CD)と ID を他系統の２つの機能モジュールへ送信する．その結果、１つの機能モジュールには、他の機能モジュールの符号(CD)と ID の組合せが２つ集まる．その組合せの中から、”出力データ＋ID”と整合する符号の組合せを１つ選別する．この”出力データ＋ID＋符号(CD)”組合せを外部システムに出力する．

3 つの機能モジュールの出力データのうち符号が整合するものを１つ選んで、データや符号を加工せずにそのまま IC カードに記録する．

このケースは、データの中身が金額データや個人情報などの場合に適していると考える．

(2) デュアルシステムでデータ保存するケース

大切なデータは、ミラーディスクに記録する．さらに大切なデータは、システムまで二重化する．このような場合に、特に有効である．

一般にハードウェアの故障は、再現性のある永久故障よりも、再現のない一時故障の方が、発生確率が高い．したがって、誤りデータが発生した場合、いきなり FDIR (Failure Detection, Isolation and Recovery) を実行しないで、リトライをして同じ動作を繰り返すことは合理的な解法である．リトライで誤りデータをマスクできる確率は、できない確率より高いので、必ずしも多数決は必要でない．このような場合、機能モジュールは２つでよい（デュアルシステム）．リトライしても、誤りマスクできない場合は、運用を止めて修理を待つ、ということになる．

一般的なミラーディスクは、誤り検出符号により記録時エラーは検出できるが、それより前の処理過程で発生したエラーについては、検出できない．デュアルシステムでは、一般的に処理結果の比較照合が行われるが、誤り検出符号は、出力する機能モジュールが振り直してしまうため、誤動作による見逃しの故障モードの可能性を排除できない．

本稿提案の方法によれば、1 台のディスクのデータを読み出し、誤り検出符号を解くだけで、処理過程の誤りや記録時の誤り混入が無いかまでチェックすることができる．

構成例を図７に示す．応用プログラムは出力データを生成する．その出力データと機能モジュール ID より、符号(CD)を生成する．符号(CD)と ID をもう１つの機能モジュールへ送信する．その機能モジュールから交換で符号(CD)と ID を得る．この符号(CD)と ID の組合せが自らの出力データと整合するかチェックする．チェックの結果、問題なければ、ハードディスクに書き込む．問題ある場合は、どこかでエラーが出たと考えられるので、データを送ってきた機能モジュールに、リトライを要求する．リトライ要求を受けた機能モジュールは、応用プログラムの処理からやり直す．

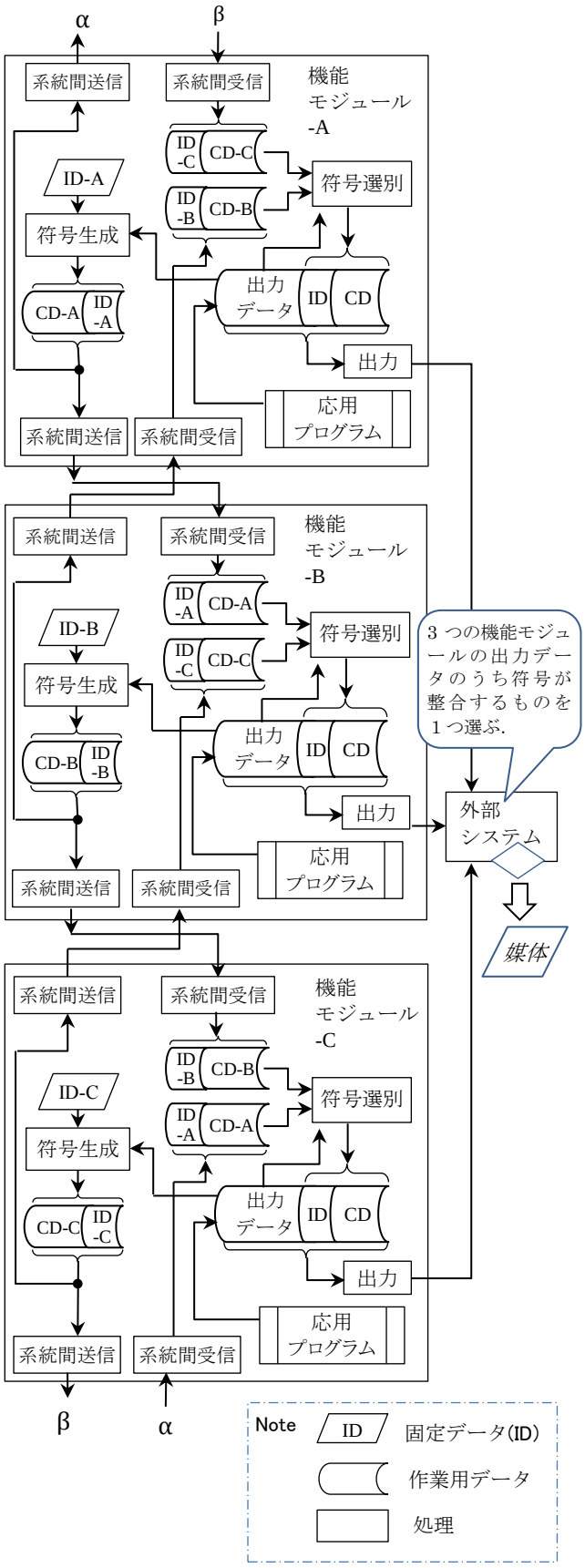


図６ データと符号を媒体に記録する
高信頼性システムの構成例

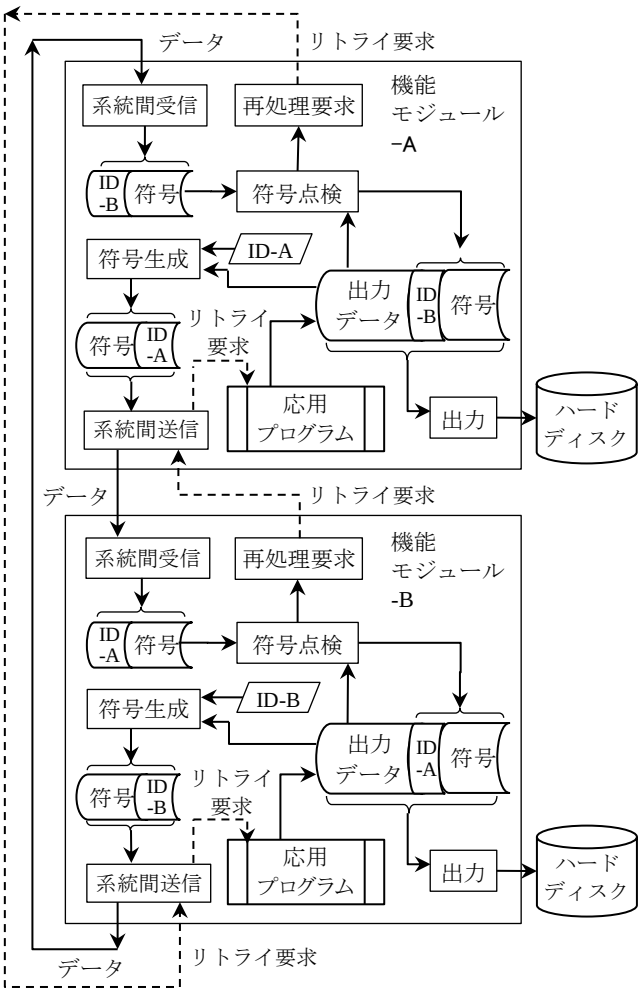


図 7 デュアルシステムによるミラーディスクの構成例

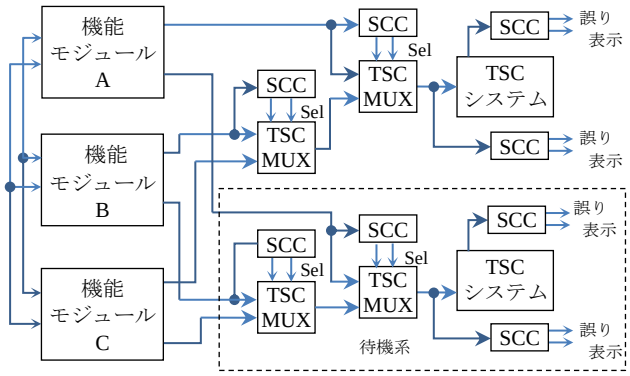
4.2 外部システムがセルフチェッキングシステムになっているケース

外部システムが、セルフチェッキングシステムになっていて、かつ冗長系を構成している場合、全体で 1 故障許容システムを構成することが可能である．セルフチェッキングとは、通常の動作中に、外部からのテストなしに論理回路が自分自身で故障を検出する能力のことである．

セルフチェッキングシステムを構成するためには、次の条件が必要である[9].

(1) 各機能ブロックの出力が、誤り検出符号で符号化されていること．
(2) 動作中に故障が生じると､一定の動作サイクル内に非符号語が出力されるよう機能ブロックを構成すること．
(3) その非符号語を監視するチェッカは機能ブロックのみならず自分自身の故障をも検出する．

本稿提案の方法では、機能モジュールはデータが異常の時、必ず非符号語を出力する（セルフテスティング性[10]. ただし、符号の衝突率は無視可能なほど小さいとする．）この符号語をセルフチェッキングシステムに入力させることにより、システム全体でセルフチェッキングシステムを構成することが可能である．図 8 にセルフチェッキングシステムの構成例を示す．図では、外部システムは、２系統のセルフチェッキングシステムで待機冗長系を構成しており、故障許容性を有している．本稿提案の機能モジュールの出力をセルフチェッキングマルチプレクサ(TSC MUX)に不正な符号語を入力させない構成になっている．機能モジュールが非符号語を出力する場合は、セルフチェキングチェッカー(SCC)で機能モジュールを切り替える。さらに TSC MUX が非符号語を出力する場合は、SCC にて誤り表示する．



Note
・TSC：Totally Self-Checking（トータリセルフチェッキング）
・MUX：Multiplexer（マルチプレクサ）
・SCC：Self-Checking Checker（セルフチェッキングチェッカ）
・Sel：チェッカが出力する２線式マルチプレクサの信号選択信号

図 8 セルフチェッキングシステムの構成例

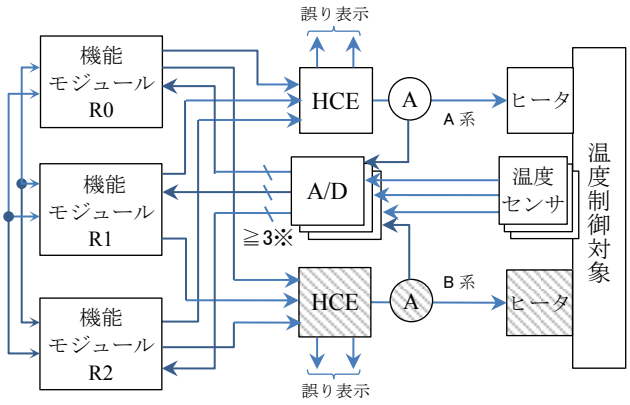
4.3 外部システムをループバックや定期的テストなどにより動作を保証するケース

これは、外部システムが冗長系を構成しており、ループバックや定期的テストなどの方法により動作が検証されるケースである．例として、有人宇宙船のヒータ制御システムを採りあげてみよう[11, 12]．宇宙船に搭載される化学エンジンは、噴射口や配管、バルブ、タンクから構成されている．この内、配管とバルブは宇宙の厳しい温度環境に特に弱い部分である．地球近傍の宇宙では高温側は太陽光によって 130℃以上、低温側は暗黒宇宙との熱結合によって -160℃以下にもなると言われている．配管・バルブは、低温になり過ぎると燃料の凍結による破裂や異常燃焼、高温になり過ぎると、爆発などの危険がある．これらは宇宙船の人命に関わる重大事象である．このため、高温側対策として配管やバルブを断熱材で覆うこと、低温側対策として電気ヒータで加熱することが一般的に行われている．この場合ヒータ制御に求められている要求は、故障があっても、低温時にはヒータ加熱を止めないこと、逆に高温時には決してヒータ加熱をしないことにより、所定の温度範囲に温度制御対象を収めることである．故障耐性が要求されている．

図 9 に 1 故障耐性のヒータ制御システムの構成例を示す．図について、処理の流れを説明する．

(S1) 機能モジュールは、系統間交換型誤り検出符号付きのデータ（=ヒータ制御コマンド．以下コマンド）をヒータ制御装置（Heater Control Equipment, 以下 HCE）に出力する．機能モジュールは３台あり HCE は３台の機能モジュールからコマンドを入力する．

(S2) HCE は、符号をチェックし、整合性に問題のないコマンドを 1 つ任意に選択して、ヒータに電力を供給する．
(S3) ヒータの近傍には温度センサが 3 系統付いていて、温度を計測し、A/D 変換によりデータ化して機能モジュールに送る．
(S4) 機能モジュールは、温度データによりヒータ制御コマンドを生成して、(S1) に戻って制御を繰り返す．



Note
・機能モジュール Ri：周期的に互いに位相をずらして意図的に非符号語を出力する機能モジュール．これにより、外部システムによる機能モジュール入力ポート切替を起こさせ、切替機能の健全性を確認する．
・HCE：Heater Control Equipment（ヒータ制御装置）
・A：Ampere Meter（電流センサ）
・A/D：Analogue to Digital Converter（AD 変換器）
・※：機能モジュールにて温度センサ 3 系統を多数決するために、A/D→機能モジュールの信号ラインは、機能モジュール 3 台にそれぞれ独立した 3 系統のラインが必要である．電流センサ 2 系統も互いに独立したラインが必要になる．

図 9　1 故障耐性のヒータ制御システムの構成例

続いて、部位ごとの FDIR について説明する．

(F1) HCE とヒータの FDIR：ヒータの断線や HCE の故障検知のためヒータ電流を検知する電流センサを使用する[11,12]．この場合、機器が正常ならばコマンドでヒータ On の時は電流有り、ヒータ Off の時は電流無しなので、電流センサから返ってくるデータ（電流有／無）は、ヒータ制御コマンドと同じ内容になる．したがって、機能モジュールでは、出力したコマンドと異なるデータが電流センサから返ってくる場合は、経路に故障があるということになり、HCE やヒータを切り替える．

(F2) 温度センサと A/D 変換器の FDIR：温度センサは 3 個以上用意し、機能モジュール側でそのデータを多数決取捨する．A/D 変換器も、温度センサの系統ごとに別々に用意し、1 故障で系統間にエラーが伝搬しないように構成する．

(F3) 機能モジュールの FDIR：機能モジュールは、系統間交換型誤り検出符号を出力する．系統間交換型誤り検出符号を入力する HCE は、データとの整合性をチェックし、整合性のあるデータを選択する．HCE のこの選択機能が故障していないか点検するため、機能モジュールは、定期的に他の機能モジュールと重ならないように位相をずらして、非符号語を出力する．HCE で非符号語を検出した場合は、機能モジュールの再選択が行われる．

HCE が非符号語を検出すべきタイミングで、検出しなかった場合は、HCE に潜在的な故障があるということがわかる．

この例では、機能モジュール（出力：コマンド）→ HCE（出力：ヒータ電流）→ 電流センサ（出力：アナログ H/L）→ A/D 変換器（出力：ディジタル 1/0）→ 機能モジュールと、同じ情報が形態を変えながら、ループバックしている．このループバックにより、ヒータ制御機能の健全性を保証することができる．

さらに、HCE の誤り検出符号のチェック機能により、非符号語を出力する機能モジュールを切り替えることにより故障マスクを実現している．この機能は異常が発生しないと働かない機能なので、ループバックで検証することができない．このため、定期的に意図的に機能モジュールから非符号語を発することで、健全性を検証することができる．

このループバックと定期的テストは、セルフチェッキングシステムの有するフォールトセキュア性とセルフテスティング性に対応する性質を実現している．

以上のような設計により、本稿提案の系統間交換型誤り検出符号を適用して、フォールトトレラントシステムを構築できると考える．

4.4 一般的な分散システムでサーバ側に信頼性が要求されるケース

一般的なインターネット環境以外の特別なハードウェアを用いることに制約があり、リアルタイム性も要求されない場合などの利用が想定される．

例えば、外部システム側とはクライアント PC であり、文字通り他者（顧客や他社）の設備であり、そこで故障が発生した発生したとしても、大きな問題にならないが、機能モジュールで構成されるサーバ側（自社設備）が原因で障害が発生することは避けたい場合がある．

機能モジュールは 1 台のパソコンであり、二重、三重の冗長化を図ったとしても、大したコストにならない．系統間誤り検出符号として、TCP/IP のチェックサムの領域を使えば、外部システム（クライアント PC）側は、二重系（あるいは三重系）を意識することなく、高信頼のデータを受信することができる．

4.5 適用ができないケース

逆に適用できないケースは、次のケースなどである．
(1) 対象システムの出力先が、誤り検出符号を処理できないケース
(2) 出力機器などを含む外部システムの信頼性が低く、FDIR などの機構もないケース．

5. 適用例の信頼性についての考察

以上の適用例のうち、4.1 節～4.3 節の適用例のシステム全体の信頼性は、機能モジュールを三重系とする時、機能モジュール 1 台の信頼性を R、外部システムの信頼性を R_E とすると（式 5）となる．

＜系統間交換型誤り検出符号三重冗長系の信頼性：
外部システムは単系＞
$=(3R^2-2R^3)\times R_E$ ・・・・・・・・ (式５)

さらに、外部システムが二重冗長系の場合は,（式６）となる。

＜系統間交換型誤り検出符号三重冗長系の信頼性：
外部システムは二重系＞
$=(3R^2-2R^3)\times(2R_E-R_E^2)$ ・・・・ (式６)

一方、TMR の信頼性は、多数決回路の信頼性を R_V とする時,（式７）で与えられることが知られている.

＜TMR の信頼性＞
$=(3R^2-2R^3)\times R_V$ ・・・・ (式７)

TMR の文献 [1, 5, 6] では、多数決回路の信頼性 R_V が低い場合、システム全体の信頼性を上げるためには、多数決回路を多重化しても効果はなく、唯一の解決は多数決回路自体の信頼性 R_V を上げていくしかないと言われている. では、4.1 節～4.3 節の適用例ではどうか. 4.1 節～4.3 節の適用例では、外部システムが誤動作したことを検証することができる. 誤動作した場合は、外部システムを冗長系に切り替えて動作させればよい. 即ち、外部システムを冗長化することで、システム全体の信頼性（稼働率）を上げることができる. 外部システムの信頼性が低い場合には、外部システムの多重度を上げていけば良いのである.

即ち,（式５）は、形の上では（式７）と同じでも、全く異なる結論を得ることができる.

6. まとめ

提案の方法は、下流の外部システムに誤り検出符号をチェックする機能があれば適用することができる.

三重冗長系を構成する場合でも、多数決をしてるわけではないので、多数決冗長系ではない. 系統間交換型誤り検出符号冗長とでも言うべき方法であると考えられる.

既知の TMR に比べてデータ伝送量が少ない、システム中の全ての部位の全ての故障モードをマスクできる、などのメリットがある. また、外部システムの信頼性があまり高くない場合でも、次のケースでは有用であることがわかった.

(1) データと符号を一緒に媒体に記録するケース.
(2) 外部システムがセルフチェッキングシステムになっているケース
(3) 外部システムをループバックや定期的テストなどにより動作を保証するケース
(4) 分散システムでサーバ側に信頼性が要求されるケース

特に (1)～(3) のケースでは、外部システムの信頼性が低い場合には、外部システムを冗長系にすれば、システム全体の信頼性を上げることができる. もちろん、TMR と同じように機能モジュールの信頼性が低い場合（但し信頼性 0.5 以上）には、機能モジュールの多重度を増やせばよい. 機能モジュールと外部システムの冗長数により、システム全体の信頼性や稼働率は、いかようにでも調整することができる.

特別なハードウェアを必要としない、という特長も合わせて考えると、今後普及が期待される.

文　献

[1] J. Von Neumann: “Probabilistic Logics and the synthesis of reliable organisms from unreliable components,” Automata Studies, Ann. of Math. Studies , no. 34, C. E. Shannon and J. McCarthy, Eds. , Princeton University Press, pp. 43-98, 1956
[2] A. D. Ingle and D. P. Siewiorek: “A reliability model for various switch designs in hybrid redundancy”, IEEE Trans. Compt., Vol.C-25, No.2, pp.115-133 (Feb. 1976).
[3] J. Losq: “A highly efficient redundancy scheme: self-purging redundancy”, IEEE Trans. Compt., Vol.C-25, No.6, pp.569-578 (June 1976).
[4] De Sousa, Paulo T.; Mathur, Francis P., "Sift-Out Modular Redundancy," *Computers, IEEE Transactions on*, vol.C-27, no.7, pp.624, 627, July 1978
[5] 米田友洋, “無駄と冗長”, 電子情報通信学会情報システムソサイエティ誌19(2) 19-20., 2014年8月
[6] 南谷崇: “フォールトトレラントコンピュータ”, オーム社, 1991
[7] 岩井仁司:”認証整合性符号によるフォールトマスク三重冗長系の一提案”, 信学技報 IEICE Technical Report DC2014-71 (2014-12)
[8] D. Davies and J.F. Wakerly: “Synchronization and matching in redundant systems”, Trans. Compt., Vol.C-27, No.6, pp531-539 (June 1978)
[9] W.C.Carter and P.R.Schneider: “Design of dynamically checked computers”, IFIP68, pp.878-883 (Aug.1968)
[10] D.A.Andersomn and G.Metze: “Design of totally self-checking circuit for m-out-of-n codes”, IEEE Trans. Compt., Vol.C-22, No.3, pp.263-269 (Mar.1973)
[11] 深津敦: “HTV の安全解析の現状”, 宇宙開発委員会 安全部会資料 p19, JAXA, Aug.5.2005
[12] 岩井仁司, 樋渡美香, 桐谷浩太郎, 原田基之：“HTV システム管理機能の CBCS 要求と安全化設計”, 第 54 回宇宙科学技術連合講演会 2G13, Nov.2010