論文

誤動作自己検知機能を内蔵したマイクロプロセッサの開発

園田　卓*　　森　久直*　　坂巻佳壽美*　　金岡　威*

Design of self-checking microprocessors by coding

Taku SONODA, Hisanao MORI, Kazumi SAKAMAKI and Takeshi KANAOKA

Abstract　Microprocessors are indispensable to control systems such as industrial machines. However standard microprocessors are hardly designed fail-safe, and the need for reliable microprocessors against errors and hardware failures are increasing.
In this paper, the design of self-checking microprocessors which used hamming code as error-detecting and error-correcting code to detect errors in DATA-BUS was proposed. Also, these self-checking microprocessors can be redesigned for user's needs because they are designed by VHDL.

Keywords　error-detecting code, error-correcting code, microprocessor, self check, VHDL

１．はじめに

現在市販されている汎用マイクロプロセッサ（以下ＭＰＵ）には，産業用機器などの重要な制御部分に利用されているにもかかわらず，信頼性向上に関する対策が施されていないものが多い。今日のＭＰＵの利用状況から考えると，高機能であることはともかく，信頼性向上機能の内蔵こそが，制御システム全体の高信頼化と低価格化に必須であると思われる。

そこで，本研究では誤動作の原因の１つである，データ転送過程において発生する誤りに着目し，符号を利用することにより誤りを自己検知する機能を持たせたＭＰＵ（以下モデルＭＰＵ）の開発を行ったので，以下に報告する。

２．モデルＭＰＵの開発

本研究においては，ＭＰＵとメモリとを接続するデータバスに着目をし，ＭＰＵの誤動作を起こす主な原因が，ノイズ等の影響によるデータバス上のデータの変化であり，その変化は一過性のものであるという前提でモデルＭＰＵの開発を行った。当初は，バスとともにＩ／Ｏポートについても符号化の適用を考慮していたが，今回はバスのみへの適用に留めた。それは，バス上ではメモリに書き込んだ（あるいはあらかじめ書き込んでおいた）データやプログラムを再度読み出すことにより，復号化が効果的に適用できるのに対し，Ｉ／Ｏポートではその先に接続される装置によりデータの流れの事情が異なるために，専用Ｉ／Ｏデバイスが必要となってしまうからである。

モデルＭＰＵは，仕様として情報処理技術者試験用の仮想コンピュータであるＣＯＭＥＴを参考にし，ＶＨＤＬ（Very high speed IC Hardware Description Language）を用いて開発を行った。そして，ノイズ等によるデータ誤りへの対策を施していった。データの誤りを検出，または訂正する機能に関しては，データバス（モデルＭＰＵが外部とデータのやりとりをする部分）で扱うデータを符号化することにより，実現するようにした。

また，モデルＭＰＵは，データ誤りが発生する環境の違いなどを意識し，誤り検出後の動作が違う３種類のモデルＭＰＵ－Ａ，Ｂ，Ｃを開発した。

○モデルＭＰＵ－Ａ
- データバス上の１ビットデータ誤りを訂正可能

○モデルＭＰＵ－Ｂ
- データバス上の２ビットデータ誤りを検出可能
- 誤り検出時はデータを再取り込み

○モデルＭＰＵ－Ｃ
- モデルＭＰＵ－Ｂの改良版
- 誤り検出時のデータ再取り込みに回数制限を設け，誤りが無くならないときは，あらかじめＭＰＵに内蔵させておいた別プログラムを起動

2.1　モデルＭＰＵの仕様

モデルＭＰＵは，演算機能等の基本的な機能を持つＭＰＵとして次のような構成で設計した。

*情報システム技術グループ

2.1.1　命令語

命令語の構成は，後述する符号化を考慮して１語１１ビット，命令長を２語長固定とした（図１）。また，命令は２３種類用意した。

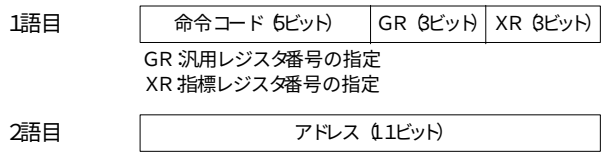


図１　命令語構成

2.1.2　内部構成

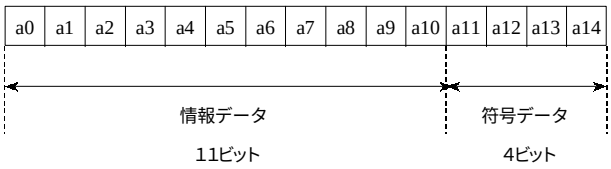
モデルＭＰＵの内部は，主に以下のようなレジスタ群や回路等によって構成した（図２）。

○ＧＲ０～４（汎用レジスタ群：１１ビット×５本）
　演算や指標レジスタ等に用いるレジスタ
○ＰＣ（プログラムカウンタ：１１ビット）
　実行する命令語のアドレスを格納するレジスタ
○ＩＲ（命令レジスタ：２２ビット）
　メモリから取り込んだ命令を格納するレジスタ
○ＦＲ（フラグレジスタ：２ビット）
　演算結果の正負，または，大小を格納するレジスタ
○ＭＡＲ（メモリアドレスレジスタ：１１ビット）
　アドレスバスへ出力するアドレスを指定するレジスタ
○ＭＤＲ（メモリデータレジスタ：１１ビット）
　データバスで入出力するデータを格納するレジスタ
○ＡＬＵ（算術論理演算部）
　算術演算や論理演算を行う回路
○制御回路
　モデルＭＰＵの内部動作を制御する回路
○Ｉ／Ｏポート
　外部の入出力装置とデータの入出力を行う部分
○符号化回路（ハミング符号）
　データの符号化を行う回路
○復号化回路（ハミング符号）
　取り込んだデータの誤りを検出または訂正する回路

2.2　データバスの符号化

データ誤りを検出・訂正するには，データを符号化することによりデータに冗長性を持たせる必要がある。今回は符号としてデータ長が１５ビットで，そのうち１１ビットを情報量として持つ（１５，１１）ハミング符号（図３）を採用し，ハミング符号の符号化・復号化回路を開発してデータバス上に適用した（図４）。

ハミング符号は最小ハミング距離（任意の符号語間における要素の差の数の最小値を示す）が３の符号で，１ビットのデータ誤りの検出・訂正が可能であり，また，訂正能力をなくすことにより，２ビットまでのデータ誤りの検出も可能になる。



$a11 = a0 \oplus a1 \oplus a3 \oplus a4 \oplus a6 \oplus a8 \oplus a10$

$a12 = a0 \oplus a2 \oplus a3 \oplus a5 \oplus a6 \oplus a9 \oplus a10$

$a13 = a1 \oplus a2 \oplus a3 \oplus a7 \oplus a8 \oplus a9 \oplus a10$

$a14 = a4 \oplus a5 \oplus a6 \oplus a7 \oplus a8 \oplus a9 \oplus a10$

(⊕ 排他的論理和)

図３　（１５，１１）ハミング符号

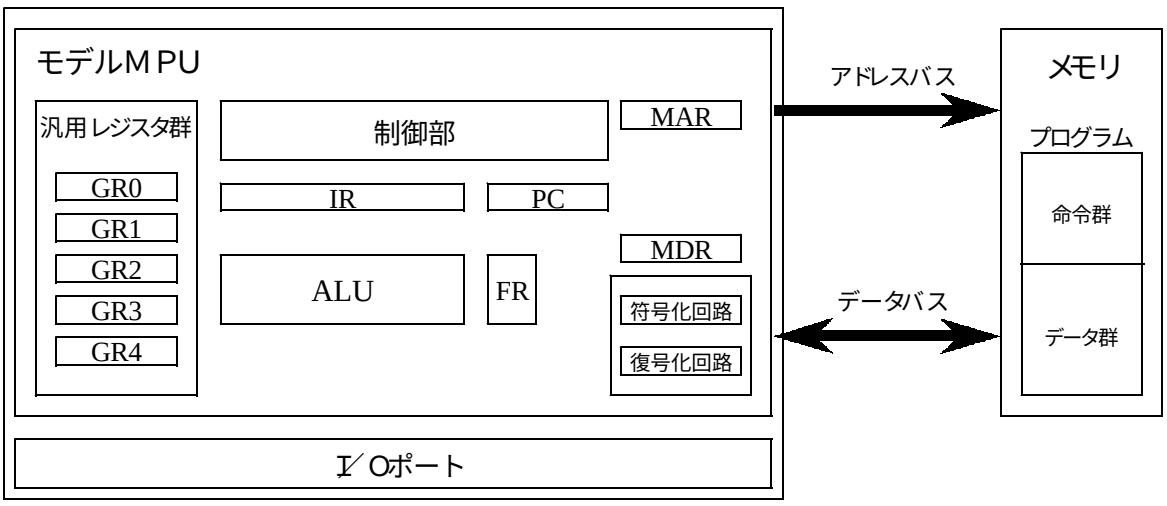


図２　モデルＭＰＵの内部構成

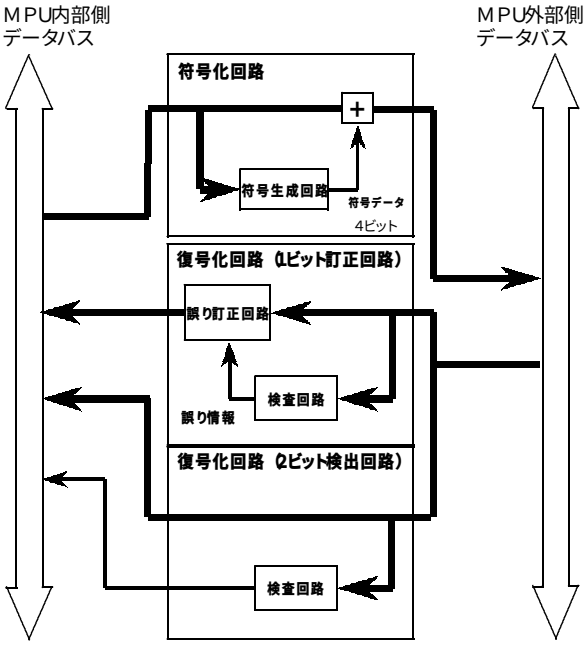


図４　符号・復号化回路のバスへの適用

2.2.1　符号化回路

符号化回路は，モデルＭＰＵ内部の情報データを符号化し，データバス上へ出力する回路である。

情報データ１１ビットを元にして，符号生成回路で符号データ４ビットの生成を行い，情報データと符号データを合わせることでハミング符号語１５ビットを生成する（図５）。

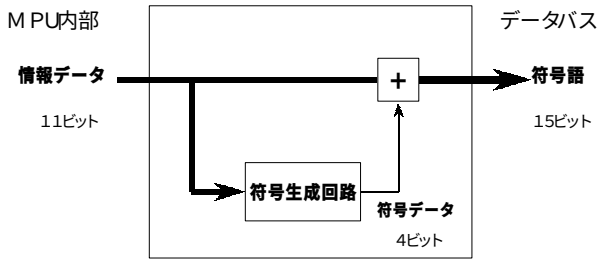


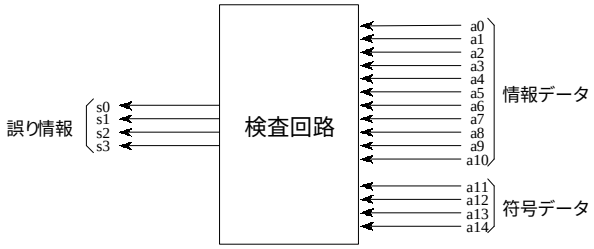
図５　符号化回路

2.2.2　復号化回路

復号化回路は，データバスを通して取り込んだデータの誤りを検出または訂正する回路である。誤りに関する情報を得るために必要な検査回路を使用し，モデルＭＰＵ－Ａで使用する１ビット誤り訂正回路，モデルＭＰＵ－Ｂ，Ｃで使用する２ビット誤り検出回路の２種類を開発した。

(1) 検査回路

復号化回路では，データバスから取り込んだデータ１５ビットからデータの誤り情報を得るために検査回路（図６）を使用する。これにより，１ビットのデータ誤りにおける誤りの位置の特定や，２ビットのデータ誤りの検出が可能となる（表１）。



s0 = a0⊕a1⊕a3⊕a4⊕a6⊕a8⊕a10⊕a11

s1 = a0⊕a2⊕a3⊕a5⊕a6⊕a9⊕a10⊕a12

s2 = a1⊕a2⊕a3⊕a7⊕a8⊕a9⊕a10⊕a13

s3 = a4⊕a5⊕a6⊕a7⊕a8⊕a9⊕a10⊕a14

(⊕ 排他的論理和)

図６　検査回路

表１　検査回路における誤り情報

<s0:s3>	1ビット誤りの位置	2ビット誤りの有無
0000	無し	無し
1100	a0	有り
1010	a1	
0110	a2	
1110	a3	
1001	a4	
0101	a5	
1101	a6	
0011	a7	
1011	a8	
0111	a9	
1111	a10	
1000	a11	
0100	a12	
0010	a13	
0001	a14	

(2) １ビット誤り訂正回路

１ビット誤り訂正回路では，検査回路で得られた誤り情報に基づいて，データバスから取り込んだデータの訂正を行う。そして，訂正したデータをモデルＭＰＵ内部へ入力する（図７）。

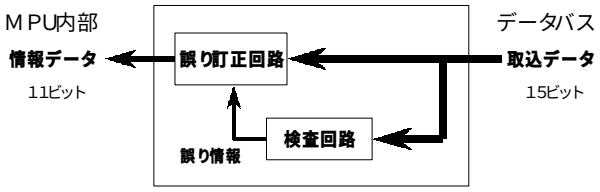


図７　１ビット誤り訂正回路

(3) ２ビット誤り検出回路

２ビット誤り検出回路では，データバスから取り込んだデータを未訂正のまま，検査回路で得られた誤り情報

と共にモデルＭＰＵ内部へ入力する（図８）。検出したデータ誤りに対する処理については，モデルＭＰＵの制御部で実行される。

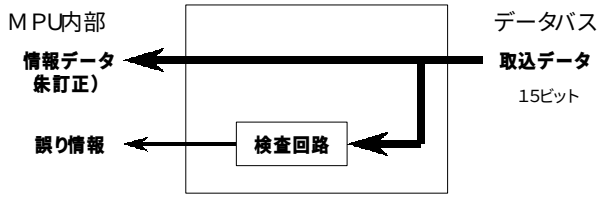


図８　２ビット誤り検出回路

2.3　制御部

モデルＭＰＵの動作は制御部によって制御しており，大きく分けると命令取り出し（fetch），命令の解読（decode），命令の実行（execute）という３つの基本動作の繰り返しによる制御となっている。

今回開発したモデルＭＰＵは，その種類によって違う動作をするように，以下のような設計を行った（図９）。また，誤り検出時の動作を簡略化するため，データバスを利用したデータ取り込みについては，すべて命令取り出し動作の段階で行うよう設計をした。

(1) モデルＭＰＵ－Ａ

命令取り出し動作で誤りがあった場合，復号化回路により自動的にデータが訂正されるため，制御部は３つの基本動作のみで構成している。

(2) モデルＭＰＵ－Ｂ

命令取り出し動作において取り込んだデータの誤りを検出した場合は，命令取り出し動作を再び実行する。この動作は，データの誤りを検出しなくなるまで繰り返され，誤りを検出しなくなった後は，通常の動作に戻る。

(3) モデルＭＰＵ－Ｃ

命令取り出し動作において取り込んだデータの誤りを検出した場合は，モデルＭＰＵ－Ｂと同じように命令取り出し動作を再び実行する。さらに，同じデータの取り込みにおけるデータの誤りが，ある一定回数まで続く場合は，あらかじめ内蔵させておいた別プログラムが起動する。

３．モデルＭＰＵの動作検証

3.1　シミュレーションによる回路検証

ＶＨＤＬで設計したモデルＭＰＵの動作検証を行うため，まず最初に，ツール上のシミュレーションによる検証を行った。その中で，誤り検出と訂正の要となる部分である符号化回路と復合化回路におけるシミュレーション結果について，以下に示す。

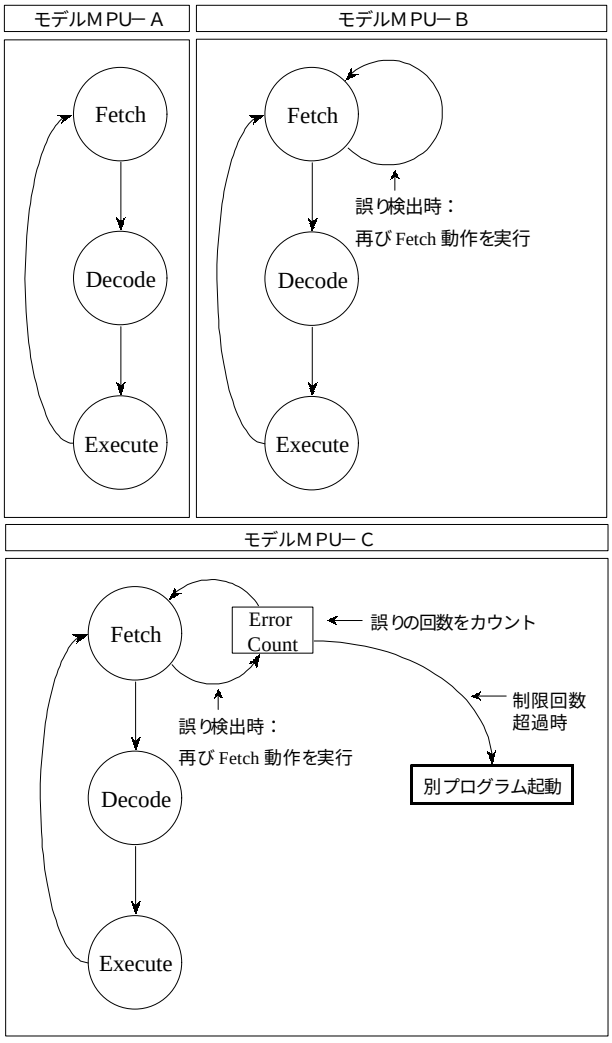


図９　制御部の動作

(1) １ビット誤り訂正回路

図10は，符号化回路と１ビット誤り訂正回路をシミュレートしたものである。この図において，i_n は元になる情報データ，C_s は i_n を符号化したデータ，e_rr は意図的に付加する誤り，E_s は C_s に誤り e_rr が付加されたデータ，ou_t は E_s を訂正したデータとして示している。この結果より，200～600(ns)に発生している１ビット誤りについては，正常にデータの訂正を行っていることが確認できる。しかし，600～800(ns)に発生している２ビット誤りに対しては誤ったデータの訂正を行っており，元のデータと違うデータになってしまっている。

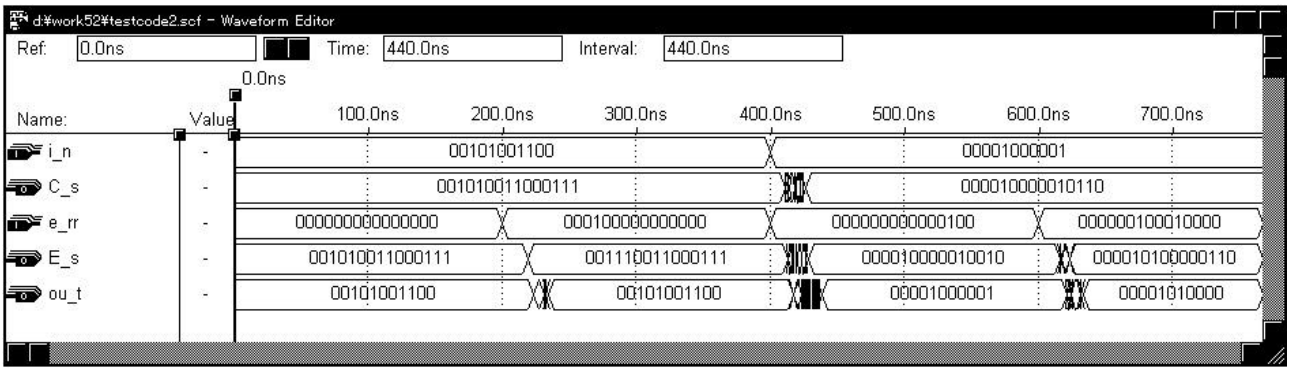


図１０　１ビット誤り訂正回路シミュレーション結果

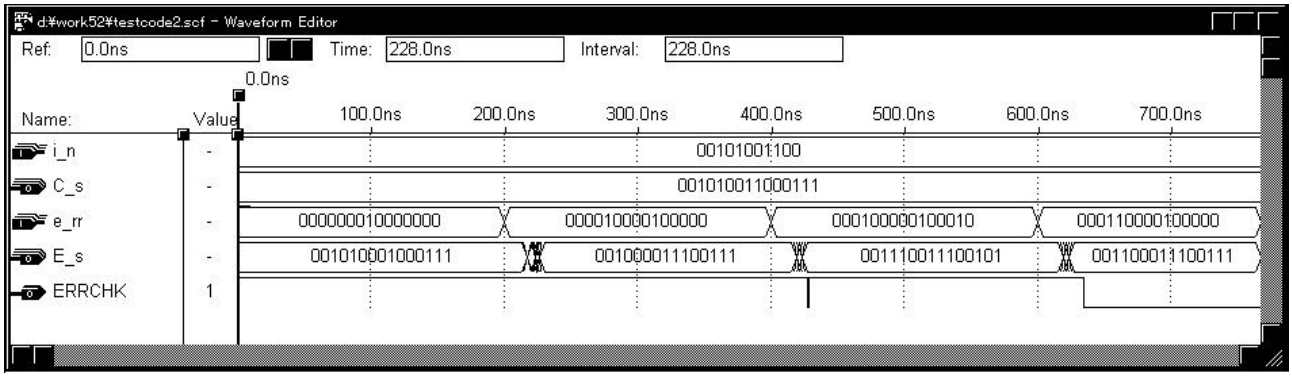


図１１　２ビット誤り検出回路シミュレーション結果

(2) ２ビット誤り検出回路

図１１は，符号化回路と２ビット誤り検出回路をシミュレートしたもので，ERRCHK が，E_s における誤りの検出を示している。この結果から，0～200(ns)に発生している１ビット誤りや，200～400(ns)に発生している２ビット誤りに対しては，誤りの検出を正常に行っていることが確認できる。また，３ビット誤りに対しては，400～600(ns)のように誤りの検出を行っていたり，600～800(ns)のように検出を行っていなかったりすることが確認できる。

3.2　評価用ボードによる回路検証

ツール上でのシミュレーションの結果，期待通りの動作を確認できたため，次に実機を使ったモデルＭＰＵの動作検証を行った。

ＶＨＤＬで設計・記述した回路は，論理合成ツール，及び，配置配線ツールを使ってコンパイルすることにより，ＦＰＧＡ（Field Programmable Gate Array）やＣＰＬＤ（Complex Programmable Logic Device）といった回路の書き込みができるＩＣに，その回路の内容を書き込むことが可能である。また，書き込みを繰り返して行えるＩＣが存在するため，ＶＨＤＬで設計した回路は実機での検証が比較的簡単に行える。

動作検証では，開発したモデルＭＰＵをＣＰＬＤに書き込み，その外部にメモリと入出力装置（ボタンスイッチ，ＬＥＤ×２）を接続した評価用ボードを製作して，テストを行った（図１２，図１３）。

テストには２桁１０進数のアップ・ダウンカウンタのプログラムをメモリに格納して使用し，入出力装置による実動作の確認を行った。また，データバス部分に誤りを発生させるため，データバスのデータをビット毎に自由に反転できるようにした。

(1) モデルＭＰＵ－Ａ

データバス上に１ビット誤りを発生させた場合は，データの修正を行って，問題なく動作した。しかし，２ビット以上の誤りに対しては，予想した通り誤動作を起こした。これは，検査回路から得られる誤り情報が，実際の誤りとは異なる１ビット誤りとして解釈され，訂正されてしまうためである。

(2) モデルＭＰＵ－Ｂ

データバス上に１ビット誤り，２ビット誤りを発生させた場合は再度データ取り込みを行い，誤りが無くなる

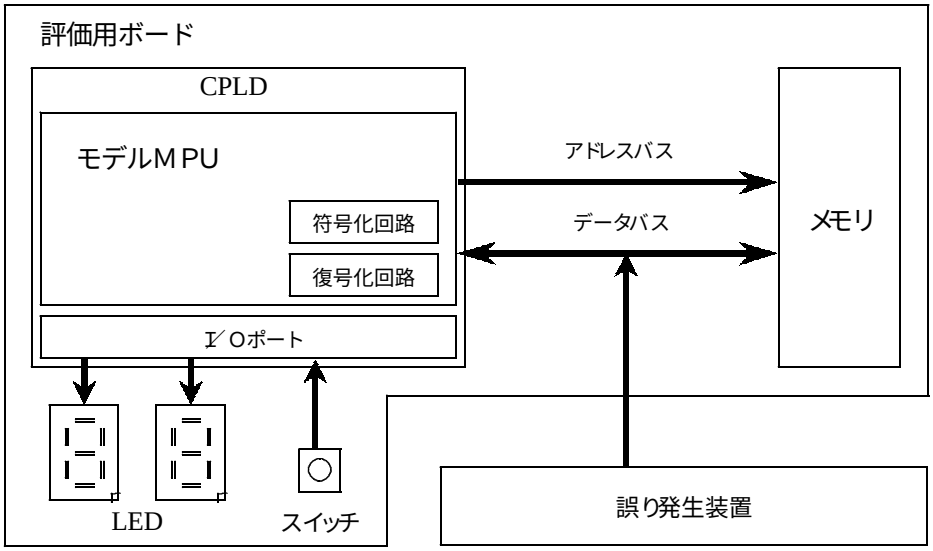


図１２　評価用ボードの構成

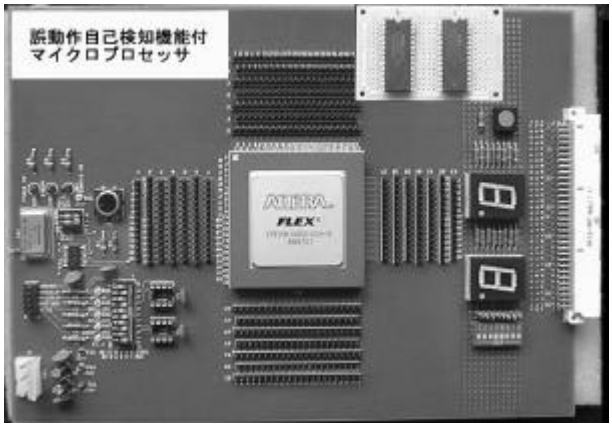


図１３　評価用ボード

と通常動作に戻ることが確認できた。さらに，３ビット以上の誤りに対しても同じように動作をすることが多いことも確認できた。これは３ビット以上の誤りについても誤りの発生を検出できるケースが多いためで，具体的には３ビット誤り，４ビット誤りに対して約９割を検出する。ただし，データバスの配線異常などで誤りが発生し続けた場合には，いつまでもデータ取り込みを繰り返すため，見かけ上は停止状態と同じになってしまう。

(3) モデルＭＰＵ－Ｃ

基本的にはモデルＭＰＵ－Ｂと同じ動作をしたが，長時間誤りを発生させた場合はモデルＭＰＵに内蔵した別プログラムが起動することが確認できた。これによりデータバスの配線異常などの継続的な誤り発生に対しても対処が可能となる。また別プログラムについては，ＶＨＤＬでユーザが自分で設定できる。

４．ま　と　め

これまで，デバイス自身の信頼性に頼ることが多かった制御用ＭＰＵの高信頼化対策に対し，根本的な対策の一方法として符号の冗長化に着目して本研究を実施し，極めて有効であることを確信した。今回は，ハミング符号を採用して誤り検出・訂正を実施したが，能力的にはまだ十分であるとはいえない。しかし，誤り検出・訂正に使用できる符号は他にも多くのものが開発されており，今回の開発手法を応用しそれらを適材適所で利用することによって，さらなる信頼性向上を実現させることが可能になると考えられる。

本研究成果は，中小企業へ技術移転をすることにより，それぞれの目的にあった信頼性向上機能を持つ制御システムの開発を可能とすることができる。また，ＣＰＬＤへ書き込んだ内容は，読み出して他のＣＰＬＤに複写することを不可能とすることができるため，知的財産権を保護することができ，中小企業の利用に対し有効であると思われる。

参 考 文 献

１）向殿政男編：フォールト・トレラント・コンピューティング，P18-23，丸善（1989）.
２）藤原秀雄：コンピュータの設計とテスト，P90-94，工学図書（1990）.
３）宮川洋，岩垂好裕，今井秀樹：符号理論，P40-43，昭晃堂（1973）.
４）南谷崇，河村俊明：セルフチェッキング・プロセッサの一構成法，電子通信学会論文誌，Vol.J68-D No12,P2015-2026（1985）.

（原稿受付　平成11年８月９日）