論文 86-23-2-7

MC68000 μP의 데이터 處理機能에 관한 試驗알고리즘

(A Test Algorithm for Data Processing Function of MC68000 μP)

金 鍾 勳*, 安 光 善**

(Jong Hoon Kim and Gwang Seon Ahn)

要 約

本 論文에서는 user's manual의 情報를 토대로 MC68000μP의 데이터 處理機能에 관한 효율적인 試驗알고리즘을 제시하였다.

機能試驗을 위한 試驗벡터는 데이터 貯藏 및 傳送機能에 대한 stuck-at, 結合(coupling), 遷移(transition) 故障을 고려하였고 데이터 操作機能은 부울函數를 이용하여 결정하였다.

本 試驗알고리즘은 A) 最少의 試驗處理時間과 最大의 故障點檢率(fault coverage)을 위한 最適試驗命令語의 選定 B) 最少의 試驗模糊性(test ambiguity)을 위한 試驗順序의 결정 C) 試驗處理알고리즘 등으로 構成되어 있다.

Abstract

In this paper, we present an efficient test algorithm for data processing function of MC68000 μP. The test vector for functional testing is determined by stuck-at, coupling and transition fault for data storage and transfer. But for data manipulation it is determined by a boolean function of micro-operation. This test algorithm is composed of 3 parts, choosing optimum test instructions for maximizing fault coverage and minimizing test process time, deciding the test order for minimizing test ambiguity, and processing the actual test.

I. 序 論

一般的으로 μP의 試驗方法은 機能試驗(functional testing)과 構造試驗(structural testing)으로 나누이고 있다.[1] 機能試驗은 μP를 하나의 시스템으로 여겨, 이 自體를 機能的으로 분리한 演算裝置, 記憶裝置, PC(Prgram Counter)등의 機械的인 部分으로 나누어 주어진 試驗벡터로 處理하는 方法이며, 構造試驗은 시스템의 回路에 대한 知識을 가지고 D-알고리즘등을 利用하여 實在回路의 試驗을 하는 것이다.[2,3] μP와 같은 LSI/VLSI回路들은 內部의 複雜性과 內部構造에 대한 産業上의 秘密로 정확한 論理說計圖를 알 수 없기 때문에 構造試驗을 主로 하고 있다.[4,5,6]

μP의 試驗을 위한 일반적인 모델은 크게 데이터處理部와 制御部로 兩分시킬 수 있다.[7] 데이터 處理部

*正會員, 東亞大學校 電算工學科
(Dept. of Comp. Eng., Dong A Univ.)
**正會員, 慶北大學校 電子工學科
(Dept. of Elec. Eng., Kyungpook National Univ.)
接受日字 : 1985年 8月 26日

는 데이터 貯藏要素(버퍼, 레지스터, 화일 등), 데이터 傳送要素(버스, MUX, DEMUX 등), 데이터 操作要素(ALU, 쉬프터, 비교기 등)들로 構成되어 있다. 制御部는 命令語 및 레지스터를 디코오딩하여 命令實行에 필요한 일련의 制御信號를 발생시키는 것으로 이것의 機能은 試驗의 初期에 點檢된다.[8]

本 論文에서는 μP의 制御部를 點檢하여 正常으로 판정된 후 데이터 處理部의 機能故障을 檢出하기 위한 효율적인 試驗方法을 論하였다. 本 硏究의 對象은 MC68000μP이며 이것에 대한 最適試驗命令語의 選定, 試驗順序의 결정, 試驗處理 등에 대한 알고리즘을 제시하였다. 本 論文에서 提案한 方法은 일반적인 μP에 適用할 수 있다.

Ⅱ. 最適試驗命令語

μP의 試驗을 一般化하기 위해 命令語를 分析하여 試驗處理線圖로 記述하고 이를 토대로 最適試驗命令語[9]를 구한다.

1. 試驗處理線圖

試驗을 위한 μP의 記述에는 그래프모델[6,9,12] 및 HDL(Hardware Description Language)[10,11] 등이 提案되었는데 本 硏究에서는 μP의 機能的 構成圖와 命令語을 이용한 試驗處理線圖를 사용하며 이것은 아래의 要素로 構成된다.

i) 命令語에 의해 사용되어지는 記憶素子는 1型 頂點(圓形)으로 나타낸다.

ii) 命令語 實行時 수행되는 마이크로 오퍼레이션은 2型 頂點(矩形)으로 나타낸다.

iii) 1型 頂點이 2型 頂點에 의해 處理될 때는 그 사이에 弧(arc)가 존재한다.

iv) 1型 頂點중 入力위치에 있는 頂點은 소오스(source)라 하고 出力위치에 있는 頂點은 싱크(sink)라 한다.

v) 試驗處理線圖가 한 個의 完全連結 成分으로 構成되어 있으면 單數試驗處理線圖이고 그렇지 않으면 複數試驗處理線圖라 한다.

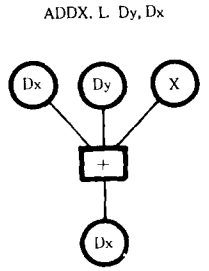
例 1) 試驗處理線圖는 MC68000 μP의 命令語 들을 사용하여 다음과 같이 설명되어 진다.

2. 最適試驗命令語의 選定

最適試驗命令語란 μP內의 모든 構造 및 機能을 포함하는 最少限의 試驗命令語로서 最少의 試驗處理時間과 最大의 故障點檢率(fault coverage)을 동시에 만족하도록 선정된다.

各 命令語에 대한 試驗處理線圖는 그 構造에 따라 클래스 Ci(1≤i≤N, N=1, 2, …, n)로 나눈 후 다음

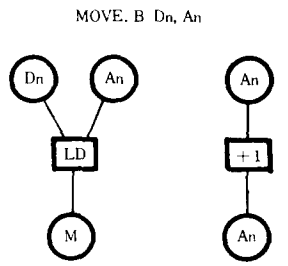
1) 單數試驗處理線圖



확장비트(extend bit) X와 데이터레지스터 Dx 및 Dy의 데이터가 더해져서 그 결과가 Dx에 저장된다.

그림 1. 단수시험처리선도의 예
Fig. 1. Example of simple test execution graph.

2) 複數試驗處理線圖



데이터레지스터 Dn의 내용이 어드레스레지스터 An이 지정하는 외부 데이터메모리에 저장되고 그런뒤 어드레스레지스터 An은 +1 만큼증가 된다.

그림 2. 복수시험처리선도의 예
Fig. 2. Example of multiple test execution graph.

과 같이 세가지 線圖로 분류한다.

i) 必須線圖 : 같은 오퍼레이션 그룹에 속하는 命令語의 線圖중에서 構造的으로나 機能的으로 도미넌트(dominant) 관계에 있는 線圖

ii) 適任線圖 : 같은 오퍼레이션 그룹에 속하는 命令語의 線圖중 機能은 같지만 構造가 다른 線圖중에서 最適試驗命令語의 構成에 적임인 線圖

iii) 除去線圖 : 같은 오퍼레이션 그룹에 속하는 命令語의 線圖중에서 그 構造와 機能이 必須線圖 및 適任線圖들에 완전 포함관계에 있는 線圖

다음 알고리즘 A는 最適의 試驗命令語를 選定하기 위한 것으로 命令語에 대한 試驗處理線圖 記述, 必須線圖 및 適任線圖의 결정 그리고 試驗命令語 결정의 과정을 거친다.

MC68000μP의 總 命令語 數는 레지스터를 별도로 고려하지 않는 경우 1544個이며 알고리즘 A를 수행한 결과 表 1에 보인 바와 같이 65個의 最適試驗命令語가 선정되었다.

Ⅲ. 最適試驗命令語의 試驗順序

```
Algorithm A
// choose optimum test instructions //

Procedure Optimum Test Instruction
  declare  C_I→class of test execution graph
           G_s←structural dominant graph
           G_f→functional dominant graph
           G_e→eligible graph
  begin
    for all instructions do begin
      compose test execution graph of instruction;
      determine C_I of test execution graph;
    end;
    for all test execution graph do begin
      determine G_s by C_I;
      determine G_f in ¬G_s; // ¬G_s is complement of G_s //
      determine G_e in ¬(G_s U G_f);
      G*←G_s U G_f U G_e; // final tert execution graph for test //
      take instructions in G*; // optimum test instructions //
    end;
  end;
```

試驗命令語의 試驗順序는 故障마스크(fault mask)에 의한 試驗의 모호성(test ambiguity)이 最少가 되도록 start small method[6]의 槪念을 도입하여 결정한다. 定義1, 2, 3은 이에 필요하다.

定義 1 : 클래스 Ci속의 命令語들은 다음 파라미터에 의해 副클래스 $C_I^{\alpha}(1 \leq \alpha \leq G,\ G=1, 2, \cdots, g)$로 나뉘어진다.

i) 複合度 k : 複數試驗處理線圖의 完全連結成分의 數

ii) 深度 p : 試驗處理線圖의 最大層(layer)의 數

定義 2 : 副클래스 C_I^{α}속의 命令語들은 다음 파라미터의 최대값에 의해 블록 B_i로 나뉘어진다.

i) 命令度 r : 記憶素子에 試驗벡터를 發生시킬 수 있는 容易性 : 한 記憶素子에 情報가 도달하기 위해 命令度 i-1의 記憶素子를 통과해야 한다면 이 記憶素子의 命令度는 i(i>1)이다.

ii) 觀察度 q : 記憶素子로 부터 試驗되어 나온 결과를 觀察할 수 있는 容易性 : 한 記憶素子에 들어있는

표 1. MC68000μP의 데이터 처리 機能試驗을 위한 最適試驗命令語

Table 1. Optimal test instructions for MC68000μP data processing function testing.

operation type	testingorder	essential graph instruction	erigible graph instruction
	4	SWAP D1	
data-movement	6	EXG D3, D4	
operation	7	EXG D5, A0	
	8	EXG A1, A2	
	23	UNLK A0	
	41	LINK A1, #⟨displacement⟩	MOVEQ.L #data, D0
	3		MOVEA.L D6, A5
	9		MOVEP.L A0@(d), D6
	20		MOVEM.L A2@+, ⟨reg.L⟩
	22		LEA A0@(d, D0), A2
	43		MOVEM.L ⟨reg.L⟩, A2@(d, D2)
	49		MOVE.W SR, A4@(d, D4)
	50		PEA PC@(d, D7)
	57		
integer-arithmetic	5	EXT.L D2	
operation	52	CMPM.L A6@+, A0@+	ADDA.L #data, A6
	14		DIVS XXX.W, D5
	19		TAS A1@
	21		MULU A3@(d, D3), D0
	48		CLR.L A5@(d, D5)
	51		ADDI.L #data, A6@(d, A0)
	58		SUBQ.L #data, A0@(d, A1)
	59		NEGX.L A3@(d, A4)
	62		
logical-operation	37		OR.L XXX.L, D7

operation type	testingorder	essential graph instruction	erigible graph instruction
	39		NOT. L A4@ (d)
	60		EOR. L D1, A1@ (d, A2)
shift-rotate operation	13		LSL. L # data, D7
	15		ASR. L D0, D1
	38		ROR. L XXX. L
	61		ROXL. L A2@ (d, A3)
bit-manipulation operation	64	BCHG. L D2, A4@ (d, A5)	
	53	BSET. L # data, A1@+	BCLR. L # data, XXX. L
	63		
binary-coded decimal D operation	18		SBCD D3, D4
	54		NBCD A2@-
	55		ABCD A3@-, A4@-
program-control operation	33	BSR. W (label)	
	35	RTR	
	17		SCS D2
	24		DBCC D7
	25		DBEQ D0
	26		DBGE D1
	27		DBHI D2
	28		DBLS D3
	29		DBMI D4
	30		DBPL D5
	31		DBVC D6
	32		SCF XXX. W
	34		SGT A3@
	40		SLE A5@-
	42		SLT A6@+
	45		JSR PC@(d)
	46		SNE XXX. L
	47		ST A1@ (d)
	56		SVS A5@ (d, D6)
system control operation	1	RESET	
	11	MOVE A3, USP	
	10	MOVE USP, A4	
	16	STOP # XXX	
	36	RTE	
	65	CHK A5@ (d, A6), D3	
	12		ANDI. W # data, SR
	44		MOVE PC@ (d, D1), CCR
no-operation	2	NOP	

情報가 觀察度 j－1의 記憶素子를 통과하여야 外部에서 觀察할 수 있다면 이 記憶素子의 觀察度는 j(j>1)이다.

μP의 外部素子는 命令度 및 觀察度가 각각 1이다. 그리고 即値(immediate value)는 命令度를 0으로 한다.

定義 3 : 블록 B_i속의 命令語들은 다음 파라미터의 최대값에 의해 副블록 $B_j^{\beta}(1 \leq \beta \leq B,\ B=1, 2, \cdots, b)$로

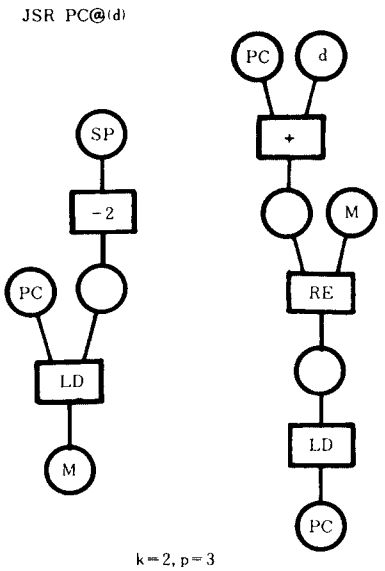


그림 3. 시험처리선도의 복합도k와 심도p
Fig. 3. Complexity k and depth p of test execution graph.

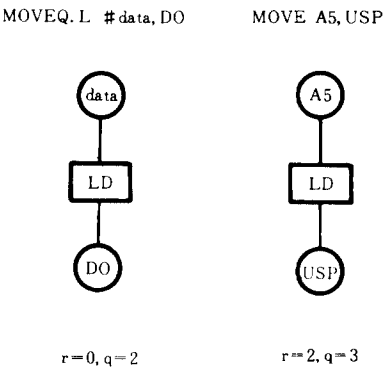


그림 4. 시험처리선도의 명령도 r및 관찰도g q
Fig. 4. Commandability r and observability q of test execution graph.

나뉘어진다.

i) 記憶素子 크기 s : 記憶素子의 비트 크기로서 바이트(byte)의 倍數로 주어진다.

ii) 오퍼레이션 크기 w : 오퍼레이션의 비트 크기로서 바이트(byte)의 倍數로 주어진다. 단 unsized operation은 0으로 한다.

例 4) ADDA.L #data, A6 : s=4, ω=4
ANDI.W # data, SR : s=2, n=2

다음 알고리즘 B는 定義1, 2, 3의 順으로 最適試驗命令語를 세분하여 그 試驗順序를 결정하기 위한 것이다. 알고리즘 遂行結果를 表1에 보였다.

```
Algorithm B
// determine the test order of optimum test instructions //

Procedure Sort (d, A, B)
// subroutine for sorting of instructions //
 begin
   sort the group for A; // indrementing order for A //
   sort the group of the same A for B; // incrementing order for B //
   let the sorted group to X1, X2 ··· Xd;
 end;
Procedure Test Order
 declare  Ci→class of test execution graph
          Ciα→sub-class of test execution graph
          Bj→block of test execution graph
          Bjβ→sub-block of test execution graph
          K→complexity of test execution graph
          P→depth of test execution graph
          r→commandability of source                    k p
          q→observability of sink
          m→bit size of memory
          n→bit size of operation
 begin
   for i=1 to 16do begin
     take all test instructions in Ci;
     determine k and p of all test instructions in Ci;
     group instructions of the same k and p;
     call Sort(e, k, p);
     forα=1 to e do begin
       for all instructions in Ciα
         determine r and q;
         group instructions of the same r and q;
         call Sort(l, r, q);
         for j=1 to l do begin
           for all instructions in Bj;
             determine m and n;
             group instructions of the same m and n;
             call Sort(s, m, n);
             for β=1 to s do begin
               take the Bjβ;
             end;
           end;
         end;
       end;
     end;
   end;
 end;
```

Ⅳ. 機能故障 모델과 試驗벡터

μP의 機能은 命令語 디코오딩 및 레지스터 디코오딩의 制御部 機能과 데이터 貯藏, 데이터 操作, 데이터 傳送등의 데이터 處理部 機能으로 대별되며[12] 이에 대한 試驗을 구분 실시한다.

1. 機能故障 모델

試驗處理線圖를 통해 구해진 表1의 最適試驗命令語을 對象으로 하여 데이터 處理 機能故障 모델을 제

시하고 이들에 대한 試驗벡터를 결정한다.

1) 데이터 貯藏機能에 대한 故障

MC68000 μP에는 데이터 레지스터(D0-D7)와 어드레스 레지스터(A0-A6) 그리고 USP, SSP 및 SR등이 내장되어 있으며 이들에는 다음과 같은 故障이 발생할 수 있다.

i) stuck-at 1 혹은 stuck-at 0

ii) 0→1→0 혹은 1→0→1의 遷移(transition) 불능

iii) 두개 以上의 셀(cell)상호간의 結合(coupling)

2) 데이터 傳送機能에 대한 故障

試驗對象이 되는 傳送路는 命令語를 實行하는 동안에 데이터가 흐르는 論理的 傳送路이며 이들에는 다음과 같은 故障이 발생할 수 있다.

i) stuck-at 1 혹은 stuck-at 0

ii) 두개 以上의 傳送路 상호간의 結合(coupling)

3) 데이터 操作機能에 대한 故障

데이터 操作機能이란 命令語의 마이크로 오퍼레이션에 해당하며 이 機能에 대한 故障은 ALU, 인터럽트回路 그리고 SP, IR 및 PC의 인크리멘팅(incrementing) 및 디크리멘팅(decrementing)에 사용되는 回路와 指標어드레싱(indexed addressing) 및 相對어드레싱(relative addressing)과 같은 여러가지 어드레싱 모드속에 들어있는 오퍼랜드의 어드레스를 계산하는 데 사용되는 回路등의 故障이 原因이 된다. 이들은 機能이 아주 다양하기 때문에 特定의 故障모델을 제시하기 곤란하므로 各 마이크로 오퍼레션의 機能에 대한 모든 가능한 試驗벡터를 구한다.

2. 試驗벡터

제시된 故障모델의 機能故障을 檢出하기 위한 試驗벡터를 결정한다. 데이터 貯藏 및 傳送機能故障의 檢出에 필요한 試驗벡터는 대부분 공통되므로 試驗의 편의상 동시에 취급하고 데이터 操作機能에 대한 試驗벡터는 各 마이크로 오퍼레이션을 부울函數로 표현한 후 이를 토대로 구한다.

1) 데이터 貯藏 및 傳送機能에 대한 試驗벡터

命令語의 데이터 貯藏 및 傳送機能 故障을 檢出하기 위한 試驗벡터 V_s는 다음과 같다.

$$\underbrace{00\cdots0}_{w\ bit};\ \underbrace{00\cdots0}_{w/2}\ \underbrace{11\cdots1}_{w/2};\ \underbrace{00\cdots0}_{w/4}\ \underbrace{11\cdots1}_{w/4}\ \underbrace{00\cdots0}_{w/4}\ \underbrace{11\cdots1}_{w/4};\ \cdots;\ \underbrace{0101\cdots01}_{w\ bit}$$

및 補의 값 그리고

$$\underbrace{00\cdots001}_{w\ bit};\ \underbrace{00\cdots010}_{w\ bit};\ \underbrace{00\cdots100}_{w\ bit};\cdots;\ \underbrace{100\cdots00}_{w\ bit}$$

및 補의 값

試驗벡터 V_s중에서 00…0 및 11…1은 故障모델 **1**) i) 및 2) i)을 檢出하기 위한 것이다. 그리고 1) ii)의 故障은 試驗벡터 V_s의 組合에서 모든 비트가 最少限 한번씩은 0→1→0 및 1→1→0로 遷移하도록 주어지므로 檢出할 수 있다. 또한 1) iii) 및 2) ii)의 故障은 雙의 結合인 경우 00…011…1 ;…; 0101…01 및 補의 값에 의해 檢出되고 個個의 結合인 경우 00…001 ;…; 100…00 및 補의 값에 의해 檢出된다.

試驗命令語의 오퍼레이션이 데이터移動(Ri←R_j)인 경우 소오스 Ri를 試驗벡터 V_s로써 初期化시키면 Ri와 R_j의 데이터 貯藏機能과 傳送機能을 동시에 點檢할 수 있다. 그러나 데이터操作(R_i←R_k 및 R_i○R_j→R_k, 단 ○은 manipulation operator 임)인 경우에는 各 소오스 R_i, R_j 및 싱크 R_k의 데이터 貯藏機能과 各 소오스-ALU 및 ALU-싱크사이의 데이터 傳送機能을 點檢하기 위한 試驗벡터가 별도로 요구된다.

例 5) 다음 試驗命令語의 데이터 貯藏 및 傳送機能을 點檢하기 위한 試驗벡터를 구한다.

i) MOVEA. L D6, A5

[D6 ; A5]=[V_s; dd…d]

단 d는 don't care 임

ii) OR. L XXX. L, D7

$$[XXX.\ L; D7]=\begin{bmatrix} V_s & ;00000000 \\ 00000000; & V_s \end{bmatrix}$$

2) 데이터 操作機能에 대한 試驗벡터 마이크로 오퍼레이션 機能故障 檢出에 필요한 試驗벡터 V_M을 구하기 위해 이들 機能을 다음과 같이 부울函數로 표현한

$$Z=F(X, Y, \cdots)\equiv z_1=f_1(X, Y, \cdots)$$
$$z_2=f_2(X, Y, \cdots)$$
$$\vdots$$
$$z_P=f_P(X, Y, \cdots)$$

여기서 $X=x_1, x_2, \cdots x_n$

$Y=y_1, y_2, \cdots, y_n$

$Z=z_1, z_2, \cdots z_P$, $p\geq n$

윗式에서 오퍼랜드의 i번째 비트와 결과 Z의 i번째 비트의 관계를 나타내기 위해 부울함수 f(x)를 다음과 같이 전개시킨다.

$$f_i(X, Y, \cdots)=\sum_j \tilde{x}_i\cdots\tilde{y}_i\cdots\cdots C_i^j$$

여기서 $\tilde{x}_i$: x_i 혹은 $\bar{x}_i$

$\tilde{y}_i$: y_i 혹은 $\bar{y}_i$

C_i^j : 변수 x_k, y_k의 부울式(단, k≠i)

윗式을 이용하여 다음과 같이 各 마이크로 오퍼레이션을 표현한다.

-complement : $Z=\bar{X}$

$\forall$ i, $Z_i=\bar{x}_i\cdot 1+x_i\cdot 0$

- AND : $Z = X \cdot Y$

$\forall i,\ Z_i = x_i \cdot y_i \cdot 1 + \bar{x}_i \cdot y_i \cdot 0 + x_i \cdot \bar{y}_i \cdot 0 + \bar{x}_i \cdot \bar{y}_i \cdot 0$

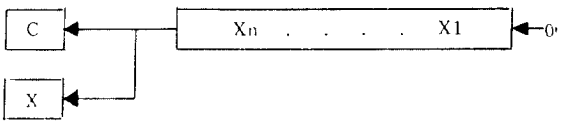
$C_i^0 = C_i^1 = C_i^2 = 0,\ C_i^3 = 1$

- OR : $Z = X + Y$

$\forall i,\ Z_i = \bar{x}_i \cdot \bar{y}_i \cdot 0 + \bar{x}_i \cdot y_i \cdot 1 + x_i \cdot \bar{y}_i \cdot 1 + x_i \cdot y_i \cdot 1$

$C_i^0 = 0,\ C_i^1 = C_i^2 = C_i^3 = 1$

- logical shift left

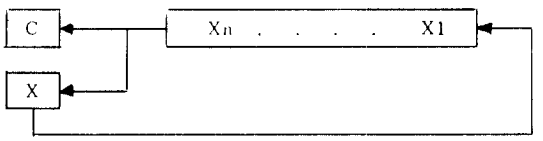


$\forall i \in [2, n]\ Z_i = \bar{x}_i \cdot \bar{x}_{i-1} \cdot 0 + \bar{x}_i \cdot x_{i-1} \cdot 1 + x_i \cdot \bar{x}_{i-1} \cdot 0 + x_i \cdot x_{i-1} \cdot 1$

$Z_1 = 0$

$Z_{n+1} = C = X = x_n$

- rotate left with Extend

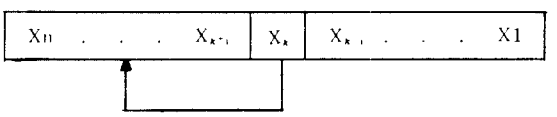


$\forall i \in [2, n]\ Z_i = \bar{x}_i \cdot \bar{x}_{i-1} \cdot 0 + \bar{x}_i \cdot x_{i-1} \cdot 1 + x_i \cdot \bar{x}_{i-1} \cdot 0 + x_i \cdot x_{i-1} \cdot 1$

$Z_1 = X$

$C = X = x_n$

- sign extend



$\forall i \in [1, k]\ Z_i = x_i$

$\forall i \in [k+1, n]\ Z_j = \bar{x}_j \cdot \bar{x}_k \cdot 0 + \bar{x}_j \cdot x_k \cdot 1 + x_j \cdot \bar{x}_k \cdot 0 + x_j \cdot x_k \cdot 1$ j

- add with carry

$\forall i \in [1, n]\ Z_i = \bar{x}_i \cdot \bar{y}_i \cdot C_i + \bar{x}_i \cdot y_i \cdot \bar{C}_i + x_i \cdot \bar{y}_i \cdot \bar{C}_i + x_i \cdot y_i \cdot C_i$

$C_i = x_{i-1} \cdot y_{i-1} + C_{i-1}(x_{i-1} + y_{i-1})$

- subtract with borrow

$\forall i \in [1, n]\ Z_i = \bar{x}_i \cdot \bar{y}_i \cdot C_i + \bar{x}_i \cdot y_i \cdot \bar{C}_i + x_i \cdot \bar{y}_i \cdot \bar{C}_i + x_i \cdot y_i \cdot C_i$

$C_i = \bar{x}_{i-1} \cdot y_{i-1} + C_{i-1}(\bar{x}_{i-1} + y_{i-1})$

마이크로 오퍼레이션이 부울함수로 표현되면 다음과 같은 관점하에서 시험벡터를 결정한다.

i) 오퍼랜드의 i번째 비트의 모든 가능 組合을 취하여 변수 x_i, y_i의 積을 試驗한다.

ii) 모든 부울式 C_i^j에 眞(1) 및 爲(0)를 대입하여 C_i를 試驗한다.

例 6) 試驗命令語 ADDA. L # data, A6의 Add 오퍼레이션에 대한 試驗벡터를 구한다.

Add 오퍼레이션의 부울函數에서 변수 x_i, y_i의 積에 대한 가능한 비트組合은 (0,0), (0,1), (1,0), (1,1)이다. 그리고 부울式 C_i의 값이 眞 및 爲가 되는 경우는 다음과 같으므로 이들 중 $x_{i-1} = y_{i-1} = 0$ 및 $x_{i-1} = y_{i-1} = 1$을 선택한다.

$$C_i = 0 \begin{cases} x_{i-1} = y_{i-1} = 0 \\ x_{i-1} \cdot y_{i-1} = 0 \text{ 및 } C_{i-1} = 0 \end{cases}$$

$$C_i = 1 \begin{cases} x_{i-1} = y_{i-1} = 1 \\ x_{i-1} = C_{i-1} = 1 \\ y_{i-1} = C_{i-1} = 1 \end{cases}$$

그러므로 Add 오퍼레이션의 i번째 비트에 대한 試驗벡터는 다음과 같이 8個의 비트形態로 나타난다.

$x_{31} \cdots x_{i+1}$	x_i	$x_{i-1} \cdots x_1 x_0$	;	$y_{31} \cdots y_{i+1}$	y_i	$y_{i-1} \cdots y_1 y_0$
	0	0			0	0
	0	1			0	1
	0	1			1	1
	0	0			1	0
d	1	0 d	;	d	0	0 d
	1	1			0	1
	1	1			1	1
	1	0			1	0

여기서 d는 don't care이며 이것을 全 오퍼레이션 비트까지 확장시켜 試驗벡터를 구하면 다음과 같다.

[# data ; A6] = {00000000 ; 00000000
55555555 ; 55555555
55555555 ; FFFFFFFF
00000000 ; AAAAAAAA
FFFFFFFF ; 55555555
AAAAAAAA ; 00000000
AAAAAAAA ; AAAAAAAA
FFFFFFFF ; FFFFFFFF
AAAAAAAA ; FFFFFFFF
00000000 ; 55555555
55555555 ; 00000000
FFFFFFFF ; AAAAAAAA}

V. 試驗處理

선정된 最適試驗命令語에 대한 데이터 處理機能試驗은 데이터 貯藏機能, 데이터 傳送機能, 데이터 操作機能을 동시에 점검할 수 있도록 이들 各各의 故障檢出에 필요한 試驗벡터의 合集合이 최종적인 試驗벡터가 되어 各 試驗命令語의 소오스를 初期化시킨다. 따라서 試驗벡터가 U個인 試驗命令語인 경우 試驗處理를 U번

반복해야 한다.

最適試驗命令語 상호간에 共通의 貯藏要素나 오퍼레이션 機能을 포함하는 경우에는 이미 點檢된 부분을 그 다음 試驗命令語의 試驗벡터에서 제외시킨다. 最適試驗命令語에 대한 試驗處理알고리즘은 다음과 같다.

```
Algorithm C
//process of test for optimum test instructions//
Procedure Test Processing
  declare Vs→test vector for data storage function and transfer function
          Vm→test vector for data manipulation function
          S(I)→source of instruction
          D(I)→sink of instruction
  begin
a:  for all test instruction do begin
      take instruction;//according to increasing test order//
      determine Vs;
      determine Vm;
c:    V*←VsU Vm;//final testvector for test instruction//
      for all V*do begin
        initialize S(I) and D(I) with V*;
d:      ececute instruction;
e:      compare D(I) with expected value;
        if result≠expected value then goto error;
      end;
      write "GOOD"
    end;
error; write "ERROR"; stop
  end;
```

定理 1 : 알고리즘 C는 制御部의 디코오딩故障이 없는 狀態下에서 데이터 處理機能故障을 檢出한다.

證明 : 단계 a의 最適試驗命令語는 MC 68000μP 의 모든 레지스터, 어드레싱 모드 및 마이크로 오퍼레이션을 포함하고 있으므로 μP內의 하드웨어 故障은 이들 命令語의 實行時에 반드시 그 故障效果가 나타난다. 단계 b에서는 알고리즘 B에서 定해진 試驗順序대로 命令語를 遂行시키므로 故障의 겹침에 의한 故障마스크가 발생하지 않는다. 단계 c에서는 데이터 貯藏, 데이터 傳送 및 데이터 操作에 대한 機能故障을 모두 고려한 試驗벡터를 결정하였다. 그러므로 制御部의 디코오딩 機能故障이 없다고 가정할 때 단계 d에서 命令語의 實行時 I_i/I_k故障[8]에 의한 故障마스크가 없으므로 이들 데이터 處理部의 故障이 존재한다면 結果가 틀리게 된다. 따라서 단계 e에서 命令語의 實行結果와 기대값이 다를것이므로 故障이 檢出된다. Q. E. D

Ⅳ. 結 論

MC 68000μP의 데이터 處理機能에 관한 효율적인 試驗알고리즘을 제시하였다. 本 알고리즘은 User's manual의 情報를 이용하였으며 最適試驗命令語의 選定알고리즘, 試驗順序알고리즘, 試驗處理알고리즘등으로 構成되었다. 最適試驗命令語를 選定하기 위하여 알고리즘 A을 적용하였으며 그 結果 總 命令語 1544個중 65個의 最適試驗命令語를 구하였다. 그런뒤 故障마스크에 의한 試驗의 模糊性을 最少로 하기 위해 最適試驗命令語에 대한 試驗順序를 결정하였다. 알고리즘 B는 이를 위한 것이다.

機能試驗을 위한 試驗벡터는 데이터 處理部의 stuck-at, 結合, 遷移故障등을 고려하여 결정하였고 마이크로 오퍼레이션 機能은 부울函數를 이용함으로서 試驗벡터를 體系的으로 구할 수 있었다. 이것은 試驗벡터의 自動的 生成(automatic test vector generation)을 可能하게 한다. 알고리즘 C는 이들 試驗벡터를 사용하여 最適試驗命令語를 시험하는 試驗處理알고리즘이다.

本 試驗알고리즘은 특히 試驗의 效率性을 강조하고 있으므로 命令語 디코오딩機能의 試驗알고리즘[8]과 함께 짧은 時間동안에 칩(chip)에 대한 機能試驗을 할수 있다. 따라서 製作중의 칩에 대한 試驗은 물론 本 칩을 사용하는 SSM-16등의 시스템의 시험도 시스템의 여유시간(idle time) 동안에 이룰 수 있다.

參 考 文 獻

[1] W. Barraclogh, "Techniques for testing the microcomputer family," *Proc. IEEE, vol. 64, pp. 943-950, 1976.*

[2] B. Courtois, "On line oriented functional testing of control sections of intergrated CPUs,' *Proc. Euromicro 7,* pp. 221-231, 1981.

[3] 김충기, "MOS technology development," 전자기술연구소 연구보고서 pp. 243- 325, 1983.

[4] M. Annaratone, "An approach to functional testing of microprocessor," *Proc. FTCS-12,* pp. 158-164, 1982.

[5] C. Robach, "Application oriented microprocessor test method," *Proc. FTCS-10,* pp. 121-125, 1980.

[6] R. Chantal, "Microprocessor functional testing," *IEEE Test Conference,* pp. 433-443, 1980.

[7] S. Thatte, "A methodology for functional level testing of microprocessors," *Proc. FTCS-8,* pp. 90-95, 1978.

[8] 김종훈, 안광선, "MC68000μp의 명령어 디코오딩기능에 관한 시험알고리즘," 대한전자공학회지, vol. 22, 1985.

[9] 김종훈, 안광선, "AHPL로 기술된 디지탈시스템에 대한 시험벡터의 생성 알고리즘," 대한전자공학회 추계 학술발표논문집, pp. 300－302, 1984.

[10] Y. Levendel, "Test generation algorithms for computer hardware description languages," *IEEE Trans. on Computer*, pp. 577-588, 1982.

[11] 안광선, "마이크로프로세서 데이터 처리시험을 위한 최적시험명령어," 대한전자공학회지, vol. 21, pp. 57－61, 1984.

[12] S. Thatte, "Test generation for microprocessors," *IEEE Trans. on Computer*, pp. 429-440, 1980.