論文 83 - 20 - 4 - 4

32Bit Floating-Point Processor의 設計에 關한 硏究

(A Study on the Design of the 32-Bit Floating-Point Processor)

李 建*, 金 悳 鎭*
(Kun Lee and Duck Jin Kim)

要 約

本 論文에서는 32 bit 浮動 小數點 處理裝置를 IEEE 標準에 따른 데이터 樣式에 맞도록 設計하여 TTL IC로서 構成하였고 이 시스템과 Z-80 마이크로프로세서와 浮動 小數點 4則 演算에 關한 實行時間을 比較해 본 결과 10倍 以上의 時間短縮을 보았다.

制御回路 設計에는 AHPL(A Hardware Programming Language)을 使用하였고 TTL IC로 構成하였으나 演算裝置와 制御裝置를 1칩으로 만들 수 있는 基礎를 이룩하였다. 이것을 조금 더 補完하면 32bit 컴퓨터의 演算裝置로써 使用될 수 있음을 確認하였다.

Abstract

In this paper, a floating-point processor which satisfied the subset of the proposed IEEE standard has been designed and realized by TTL chips. This processor consists of a floating-point arithmetic unit and a control sequencer. AHPL has been used in the design of sequencer. The execution times for the arithmetic operations were measured and compared with other microprocessor. The results had shown faster operations compared to the Z-80 processor. Though this processor was built by TTL chips, it could be fabricated as a one-chip processor.

I. 序 論

마이크로컴퓨터에 있어서는 普通 浮動 小數點 演算을 소프트웨어的으로 處理하기 때문에 演算 時間이 길어지는 短點이 있다. 미니컴퓨터나 小型 컴퓨터에 있어서는 機種에 따라서 소프트웨어的으로 處理하는 것도 있고 演算速度를 높이기 爲하여 浮動 小數點 處理機(floating-point processor)를 使用하는 경우도 있다.

浮動 小數點 演算裝置에 對해서는 Paker[1] 의 硏究를 비롯하여 많은 硏究가 發表된 바 있다. 그리고 最近에는 32bit/64bit 4則演算用 플로우팅-포인트 칩(Intel 8232) 等도 나와 있어서 使用하기 便利하게 되어 있다.[2]

이와같은 既成 프로세서를 利用하면 CPU의 設計를 簡單하게 해준다는 利點도 있으나 設計의 融通性이 不足하고 特히 高速演算을 必要로 하는 경우에는 既成品이 가지는 時間 制限을 解決할 길이 없게 된다.

本 論文은 AHPL(A Hardware Programming Language)을 使用하여 어떠한 設計條件에도 符合될 수 있고, 演算 速度도 既成品보다 빠른 浮動 小數點 4則 演算 處理機의 設計方法을 記述한 것이다.

데이터의 樣式은 IEEE 標準[4]에 準하였고, 設計 알고리즘은 Paker와 같은 方法[1]에 따라 AHPL로 制御順序를 提示하고, TTL을 使用하여 콘트롤 시퀀서(control sequencer), 데이터 패스(bata path), 모우드

*正會員, 高麗大學校 工科大學 電子工學科
(Dept. of Electronic Engineering, Korea Univ.)

接受日字 : 1983年 4月 19日

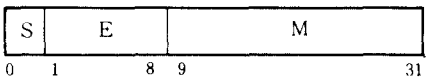
콘트롤 테이블에 依한 게이트 回路網等을 實現시켜서 그 特性을 既存 프로세서와 比較 檢討하려고 한다.

Ⅱ. 浮動 小數點 4則 演算裝置의 設計

浮動 小數點 表記法은 符號, 指數, 및 假數 部分으로 나눌 수 있다.

1. 데이터 樣式의 決定

데이터 樣式은 IEEE 標準[4]에 맞추어 다음과 같이 定하였다.



S=符號(sign) bit (0 : 陽數, 1 : 陰數)

E=8 bit 指數部(exponent), 바이어스 128

M=23 bit 假數部(mantissa)+leading bit

따라서 나타내는 값 $X=(-1)^S(2^{E-127})M$

총 bit 수 : 32+(1)

오버 플로우 : 指數部 127 以上, $2^{127}=1.7014\times10^{38}$

언더 플로우 : 指數部 −128以下, $2^{-128}=2.93874\times10^{-38}$

有効 數자리 : 假數部 23 bit, $2^{-23}=1.19209\times10^{-7}$

총 bit수는 32 bit에 데이터 포맷상으로는 나타나지 않지만 하아드웨어 상으로 리이딩 bit(leading bit) 가 늘어가서 32+1 bit가 된다.

또한 오버플로우(overflow)는 10進法으로 換算할 경우 10^{38} 以上이며 언더 플로우(underflow)는 10^{-39} 以下이고 有効數자리는 小數 일곱째자리 까지이다. 指數部分에 128씩을 더한 理由는 모든 指數部分을 陽數化 함으로써 計算過程을 簡便히 하기 위함이고 假數部分에 리이딩 bit를 둔 理由는 演算過程中 發生하는 캐리를 貯藏하기 위해서다.

2. 浮動 小數點 4則 演算의 알고리즘

加算, 減算, 乘算 및 除算의 演算 알고리즘은 다음과 같다.

1) 加算 및 減算 알고리즘

① A, B 두 오퍼랜드(operend)를 2進法 小數點을 中心으로 整列시키기 위하여 指數部分이 같아 질 때까지 指數가 작은쪽을 增加시키면서 假數를 오른쪽으로 移動시킨다.

② 加算일 경우 as를 A 오퍼랜드의 符號, cs를 演算 結果의 符號라고 할 때 假數部分을 더하고 as ⊕ cs=1일 경우 假數를 오른쪽으로 移動시킨 후 指數部分을 增加시킨다.

③ 減算일 경우 假數部分을 뺀후 as ⊕ cs=1일 경우 假數部分을 增加시킨다.

④ 假數가 正規化(normalize)될 때까지 왼쪽으로 移動 小數點 加算과 減算에 대한 알고리즘의 흐름도를 그리면 그림 1과 같다. 이 그림에서 使用한 記號들은 표 1에서 設明하였다.

표 1. 標記된 文字의 意味

Table 1. Meaning of notations.

Notation	Meaning
SIR	Instruction Register Switch
S A	A Operand Switch
S B	B Operand Switch
I R	Instruction Register
a s	A Operand Sign
A M	A Operand Mantissa Register
A E	A Operand Exponent Register
b s	B Operand sign
B M	B Operand Mantissa Register
B E	B Operand Exponent Register
c s	Result Sign
E R 1	A Operand Exponent Register
E R 2	B Operand Exponent Register
M C	Loop Counter
M Q	Multiplier and Quotient Register
e l f	Exponent Link Flag
l f	Mantissa Link Flag
Error	Error sign

2) 乘算 알고리즘

① A, B두 오퍼랜드 중 어느 한 쪽이라도 0일경우 結果는 0이 된다.

② 指数部分을 더한 후 오버플로우 및 언더플로우를 檢査한다.

③ 오퍼랜드가 陰数일 경우 그의 補数를 구한 후 假数들끼리 乘算한다.

④ 結果가 陰数일 경우 假数部分의 2의 補数를 구한 후 正規化 시킨다.

3) 除算 알고리즘

① 被除数가 0일 경우 그 結果가 0이 되고 除数가 0일경우 그 結果는 不定 또는 不能으로 된다.

② 指数部分을 뺀후 오버플로우와 언더플로우를 檢査한 .

③ 오퍼랜드가 陰数일 경우 2의 補数를 구한 후 假数들끼리 除算한다.

④ 結果가 陰数일 경우 假数部分의 2의 補数를 구한후 正規化 시킨다. 浮動 小数点 乘算과 陰算의 알고리즘을 흐름도로 나타내면 그림 2와 같다.

3. 演算裝置의 構造

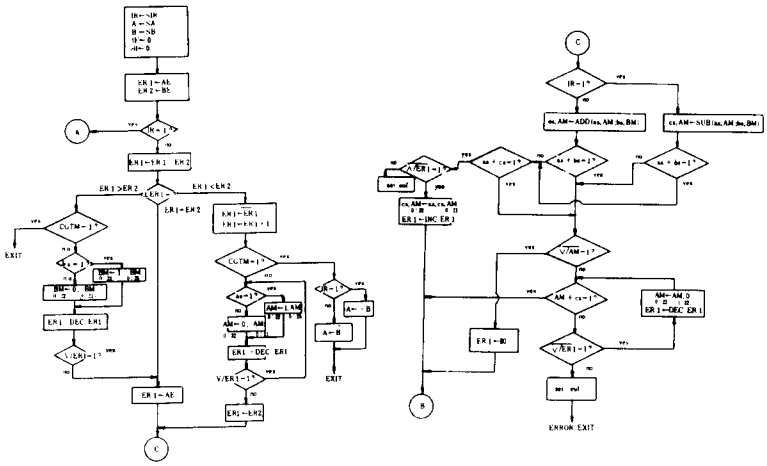


그림 1. 浮動 小數點 加算 및 減算에 관한 알고리즘

Fig. 1. Algorithms of floating-point addition and subtraction.

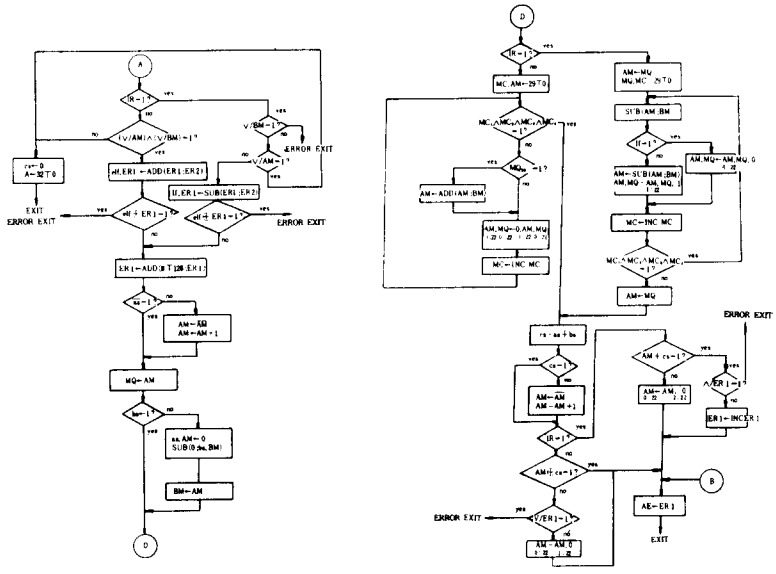


그림 2. 浮動 小數点 乘算및 除算의 알고리즘

Fig. 2. Algorithms of floating-point multiplication and division.

데이터의 入力은 DIP 스위치로 하였고 演算結果의 出力은 LED로 表示하였으며 演算裝置에 使用된 각 유닛트의 名稱과 bit수(괄호내에 表示한 것)는 다음과 같이 定하였다.

32bit 오퍼랜드를 간직하기 위해서 A,B 2개의 32 bit 레지스터가 필요하고 乘算, 除算을 위한 MQ 레지스터와 指數部分을 언패킹(unpacking) 하여 別途로 計算하기 위한 8bit의 ER1, ER2 레지스터가 필요하다.

또한 23bit 假數들의 乘算, 除算을 위하여 23번 루우프를 카운트할 5bit 카운터 MC가 필요하다.

각 레지스터와 ALU등을 結合한 浮動 小數點 演算裝置의 블럭 다이어그램을 그림3과 같이 構成하였다.

4. 制御順序

앞서 說明한 알고리즘을 基礎로 AHPL을 使用하여 制御順序를 作成하였다. 쓰여진 각 段階는 設計者에 의해서 이미 設定된 하아드웨어 素子들의 機能을 나타낸다.

```
Declare : Switches,  SA(32)
                     SB(32)
                     SIR(2)
          Registers, AE+AM(32)
          (74198)    BE+BM(32)
                     ER1(8)
                     ER2(8)
                     MQ(24)
          (74181)    EALU(8)
                     ALU(24)
          (74161)    MC(5)
                     elf(1)
                     If(1)
          Clock,     4 MHz
          L. E. D.   AM(23)  AE(8)  as(1)
                     BM(23)  BE(8)  bs(1)
                     MC(5) Error(1) cs(1)
                     elf(1)  If(1)
```

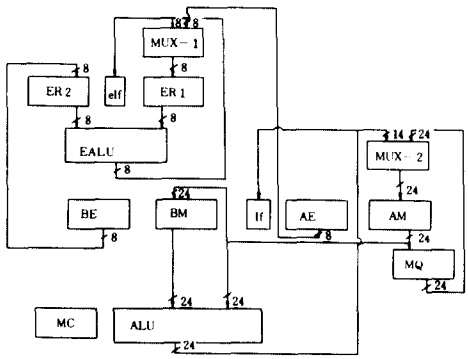


그림 3. 浮動 小數点 하드웨어의 블럭 다이어그램
Fig. 3. Block diagram of the floating-point arithmetic unit.

前述했던 알고리즘을 基礎로 하여 制御順序를 AHPL로서 表現하면 다음과 같다.

5. 콘트롤 시퀀서

앞서 設明된 制御順序를 基礎로 로직게이트와 플립플롭으로 콘트롤 시퀀서 (control sequencer)를 構成하였다. 動作되기 전 플립플롭은 클리어 狀態를 維持하다가 스타터에 의하여 만들어진 펄스가 論理條件에 따라 各各의 플립플롭을 지나면서 信号를 發生시킨다. 이 信号로서 데이타 패스를 이루는 모든 칩들을 制御한다.

例로서 AM 레지스터에 대한 스텝33의 타이밍 다이어그램을 그려보면 그림4와 같다.

6. 모우드 콘트롤 테이블

完成된 콘트롤 시퀀서에서 나오는 信号를 使用하여

```
1.  IR←SIR ; A←SA ; B←SB ; R←0 ; elf←0 ;;
2.  cs←as ;;
3.  ER1←AE ; ER2←BE ;;
4.  IR/50 ;
5.  ER1←SUB(ER1, ER2) ;;
6.  ←(ER1∧∨/ER1, ER1∧∨/ER1/7, 15, 13) ;
7.  ←CGTM, ER1/8 ; →Exit
8.  bs/10
9.  BM0/22←0, BM1/21←11
10. BM0/22←1, BM1/21 ;;
11. ER1←DEC ER1 ;;
12. →∧/ER1/8 ;
13. ER1←AE ;;
14. →32 ;
15. ER1←ER1 ;;
16. ER1←ER1+1 ;;
17. →CGTM, ER1/26 ;
18. IR/23 ;
19. cs, AM←bs, BM ;;
20. ER2←BE ;;
21. ER1←ER2 ;;
22. AE←ER1 ;; →Exit
23. as, AM←0 ;;
24. cs, AM←SUB(0 ; bs, BM) ;;
25. →20 ;
26. as/29
27. AM1/22←0, AM0/21→29
28. AM1/22←1, AM0/21
29. ER1←DEC ER1 ;;
30. →(∨/ER1)/26
31. ER1←ER2 ;;
32. IR/35 ;
33. cs, AM←ADD(as, AM ; bs, BM) ;;
34. (as⊕bs, as⊕bs)/(42, 37) ;
35. cs, AM→SUB(as, AM ; bs, BM) ;;
36. →(as⊕bs)/42
37. →(as⊕cs)/42
38. (∧/ER1)/41
39. →105 ;
40. cs, AM←as, cs, AM ;
    ER1←INC ER1 ;;
41. →104
42. →(∨/AM)/45
43. →ER1→NC ;;
44. →104 ;
45. →(AM⊕cs)/104
46. →(∨/ER1)/48
47. →105 ;
48. AM←AM1/22, 0 ; ER1←DEC ER1 ;;
49. →45 ;
50. IR/56
51. →(∨/AM)∧(∧/BM)/53
52. →Exit
53. elf, ER1←ADD(ER1 ; ER2) ;;
54. elf⊕ER1/105
55. →60
56. →(∨/BM)/105
57. →(∨/AM)/52
58. elf, ER1←SUB(ER1 ; ER2) ;;
59. (elf⊕ER1)/105
60. ER2←0 ;;
61. ER2←1, ER2 ;;
62. ER1←ADD(ER1 ; ER2) ;;
63. →65/66 ;
64. AM←AM ;;
65. AM←AM+1 ;;
66. MQ←AM ;;
67. →5 /71
68. as, AM←0 ;;
69. AM←SUB(0 ; bs, BM) ;;
70. BM←AM ;;
71. IR/79 ;
72. MC, AM←2T0 ;;
73. (MC4∧MC2∧MC3∧MC4)/90 ;
74. MQ23/76 ;
75. AM←ADD(AM ; BM) ;;
76. AM1/22, MQ1/22←0, AM0/21, MQ0/21 ;;
77. MC←INC MC ;;
78. →73
79. AM←MQ ;;
80. MQ, MC←2T0 ;;
81. R←SUB(AM ; BM) ;;
82. R/86
83. AM←SHR'AM ; BM ;;
84. AM0/22, MQ←AM1/22, MQ, 1 ;;
85. →87 ;
86. AM, MQ←AM1/22, MQ, 0 ;;
87. MC←INC MC ;;
88. →(MC4∧MC2∧MC3∧MC4)/81
89. AM←MQ ;;
90. cs←as⊕bs ;;
91. →∨/94
92. AM←AM ;;
93. AM←AM+1 ;;
94. IR/99
95. →AM⊕cs/104
96. →∨/ER1/105
97. AM0/22←AM0/22, 0 ;
    ER1←DEC ER1 ;;
98. →104 ;
99. AM⊕cs/102
100. AM0/22←AM1/22, 0 ;;
101. →104
102. (∧/ER1)/111 ;
103. ER1←INC ER1 ;;
104. AE←ER1 ;;
     →Exit
105. Error←1 ;;
     →Exit
```

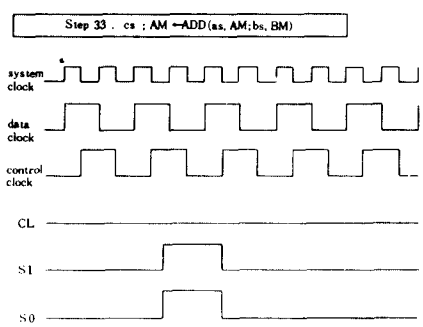


그림 4. AM 레지스터의 타이밍 다이어그램
Fig. 4. Timing diagram of AM register.

AM 레지스터에 관한 모우드 콘트롤 테이블(mode control table)을 作成하면 표2와 같다.

Ⅲ. 實驗 및 檢討

사진1에서 보는 바와 같은 브레드보오드에서 完成된 시스템으로 A, B 두 오퍼랜드가 陽数및 陰数일 경우 각각의 4則 演算을 해 보았고 이 實驗値와 計算機 fx-2200으로 計算한 값을 표3에서 比較하였다.

표 2. AM 레지스터의 모우드 콘트롤 테이블

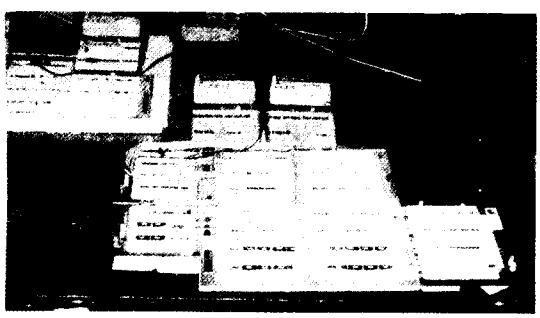
Table 2. Mode control table of AM register.

Action	Control Signal	Clear	S 1	S 0
A←SA	1	H	H	H
AM←bs, BM	19	H	H	H
as, AM← 0	23, 68, 72	L	X	X
cs, AM←SUB(0 ; bs, BM)	24, 69	H	H	H
AM 0 : 22←0, AM 0 : 21	27, 76	H	L	H
cs, AM←ADD(as, AM ; bs, BM)	33, 75	H	H	H
cs, AM←SUB(as, AM ; bs, BM)	35, 83	H	H	H
cs, AM←as, cs, AM	40	H	L	H
AM 0 : 22←AM 1 : 22 0	48, 84, 86, 97, 100	H	H	L
AM←$\overline{AM}$	64, 92	H	H	H
AM←AM + 1	65, 93	H	H	H
AM←M Q	79, 89	H	H	H
all others		H	L	L

$CL = \overline{23 + 68 + 72}$

$S1 = 1 + 19 + 24 + 69 + 33 + 75 + 35 + 83 + 48 + 84 + 86 + 97 + 100 + 64 + 92 + 65 + 93 + 80$

$S0 = 1 + 19 + 24 + 69 + 27 + 76 + 33 + 75 + 35 + 83 + 40 + 64 + 92 + 65 + 93 + 79 + 89$



사 진 1. 브레드보오드상에 완성된 시스템

Photo 1. Completed system on breadboards.

표 3. 實驗値와 計算値와의 比較

Table 3. Comparison of experimental and calculated values.

i) A=0.0537071 B=3.9982457

4 칙연산 \ 값	실 험 치	계 산 치
A D D	4.0519523	4.0519528
S U B	−3.9445385	−3.9445386
M U L	0.2147342	0.2147341
D I V	0.0134326	0.0134326

ii) A=−3.4372558 B=0.0624725

4 칙연산 \ 값	실 험 치	계 산 치
A D D	−3.3747835	−3.3747833
S U B	−3.4997282	−3.4997283
M U L	−0.2147342	−0.2147339
D I V	−55.020236	−55.020301

iii) A=0.4218479 B=−6.8775296

4 칙연산 \ 값	실 험 치	계 산 치
A D D	−6.4556837	−6.4556817
S U B	7.2993784	7.2993775
M U L	32.665275	32.665283
D I V	0.1726480	0.1726479

iv) A=−2.3747832 B=−13.75509

4 칙연산 \ 값	실 험 치	계 산 치
A D D	−16.129840	−16.129842
S U B	11.380270	11.380275
M U L	32.665275	32.665283
D I V	0.1726480	0.1726479

위 표 3에서 보는 바와 같이 小數 일곱째 자리에서 誤差가 나는 理由는 計算機도 有效數字가 7자리이기 때문이며, 이런 誤差는 指數가 작을수록 줄어드는 것을 알 수 있다.

표 4와 표 5는 各各 完成된 시스템과 Z-80 어셈블리 언어를 使用한 프로그램의 浮動 小數点 4則 演算의 實行時間을 나타낸다.

표 4. 製作된 演算裝置의 4則演算 實行時間

Table 4. Execution times of arithmetic operations on the designed processor.

4 칙연산	T states	Time (μ sec)
A D D	55 T	13.75
S U B	55 T	13.75
M U L	64 T	16
D I V	145 T	36.25

클럭 주파수 : 4MHz

設計된 演算裝置와 INTEL 8232 프로세서의 4則演算 時間을 比較하면 표 6과 같다. 이 표에서는 두 프

표 5. Z-80 마이크로프로세서에서의 4則演算 實行時間

Table 5. Execution time of arithmetic operations on Z-80 microprocessor.

4칙연산	T states	Time (μsec)
ADD	1704 T	852
SUB	1704 T	852
MUL	2019 T	1009.5
DIV	2281 T	1140.5

데이터 포맷 : 16 bit
사용언어 : Z-80 assembly language
Clock 주파수 : 2 MHz

표 6. 設計된 처리기와 INTEL 8232 프로세서의 처리시간 비교

Table 6. Comparisons of execution times between designed and Intel 8232 processors.

4칙연산	設計된 프로세서	Intel 8232 프로세서	時間差
ADD	55 T	58 T	3 T
SUB	55 T	56 T	1 T
MUL	64 T	198 T	134 T
DIV	145 T	228 T	83 T

로세서에 4 MHz의 같은 클럭 주파수를 사용했을 때의 클럭 사이클의 數 T로 나타냈다.

위의 표에서 보는 바와 같이 같은 클럭 周波数를 使用했을 경우에도 本 論文의 프로세서의 演算速度가 빠르며 乘算의 경우에는 約 3倍나 빠르다.

또, Intel 8232는 最大 使用 클럭 周波数가 4MHz로 制限되어 있으나 本 프로세서는 TTL로 構成되어 있으므로 約 10MHz까지 動作이 可能하다. 本 프로세서는 5 V의 單一 電源을 使用하는데 比하여 Intel 8232는 5 V와 12V 2個의 電源을 必要로 한다. 또 Intel 8232는 8bit CPU와 인터페이스를 容易하게 하기 위하여 8bit의 外部 버스를 使用하고 있으나 本 프로세서는 32bit CPU의 演算裝置로 設計되어 있으므로 유니트間 data 伝送速度가 3倍만큼 빠르게 된다.

誤差의 処理에 있어서는 8232는 round to even方法을 쓰는 대신 本 프로세서에서는 라운드 다운 方式을 使用하였다.

Ⅳ. 結 論

마이크로컴퓨터에 利用할 수 있는 32bit 浮動小数点 処理裝置를 提案된 IEEE 標準에 따른 데이타 様式으로 AHPL을 使用하여 設計하였으며 콘트롤시퀀서, 데이터 패스, 게이팅回路를 TTL로서 構成하였다.

또한, 完成된 시스템과, Z-80 어셈블리 言語를 使用한 프로그램과의 浮動 小数点 4則 演算의 実行時間을 比較하여 10倍 以上의 時間短縮을 보았고, 使用 클럭 주파수는 4 MHz로서, 앞으로 32bit 컴퓨터의 演算裝置로서 使用될 수 있음을 確認하였다.

既存 Intel 8232 浮動 小数点 프로세서와의 演算時間 比較에 있어서도 本 프로세서가 빠른 것이 立証되었고 유니트間 데이타 伝送 時間도 約 3倍 빠르다. 또 클럭 周波数로 8232보다 높은 周波数를 使用할 수 있어 보다 融通性이 있음을 알 수 있다.

參 考 文 献

[1] Yakup Paker, "A binary floating-point register," *IEEE Trans. Computer*, vol. C-20, no. 1, pp. 7-11, Jan. 1971.

[2] J.F. Palmer, "The INTEL standard for floating-point arithmetic," *IEEE COMSAC Conference*, Chicago, Nov. 1977.

[3] Intel, *The 8086 Family User's Manual Numerics Supplement.*, July 1980.

[4] IEEE Computer Society Microprocessor Standards Committee Task P.754, "A Proposed Standard for Binary Floating-Point Arithmetic," *Computer* 14. no. 3, pp. 51-62, Mar. 1981.

[5] Fredrick. J. Hill, *Digital Systems, Hardware Organization and Design*: Wiley, New York pp. 701, 1978.