回歸分析用 VLSI 머신 設計에 관한 硏究

(A Reserach on the VLSI Machine Design for Regression Analysis)

李 顯 洙*, 武藤佳恭**, 相磯秀夫**
(Hyon Soo Lee, Y. Takefuji and H. Aiso)

要 約

近年, 半導體 技術의 급격한 進步에 따라 高機能 論理回路의 VLSI化가 可能하게 되었다. 이에 따라 數値 處理의 高速化, 광대역 화상처리등을 위한 高機能 回路들의 專用 VLSI 칩의 設計가 硏究되고 있으며, 여러 종류의 소프트웨어 패키지의 VLSI化가 可能하게 되었다.

本 論文에서는 計算機의 回歸分析用 汎用 소프트웨어 패키지(BMD)를 하드웨어化하는 設計 手法을 提案하였다. 이것은 從來의 統計 處理를 소프트웨어에만 의존하기 때문에 處理 速度가 低下되는 것을 하드웨어化함으로써 改善하였다.

設計 알고리즘은 統計 數學의 計算 特徵을 살려 本 시스템을 構成한다. 그 結果 하드웨어化에 의하여 소프트웨어 패키지의 複雜性이 제거되고, 高速 處理함으로써 效率을 向上시켰다.

Abstract

In recent years, the logic circuits of high function have been developed to VLSI by the radical advancement of semi-conductor technologies. Under the above influence, it has become possible to design the special VLSI chips for high speed of numerical value processing, wide-band, image processing, etc. And, the development of the VLSI from various kinds of software package has become quite possible.

This paper is to propose the technical skill of hardware design about general software package (BMD). The decrease of speed of former statistics processing caused by depending on software only is improved by hardware. In regard of design algorithm, the main system will be able to be established by considering of special feature of statistics.

As a result, the complexity of software package is excluded by hardware. And, the efficiency is improved by high speed processing.

I. 序 論

最近, 1개의 칩(chip) 상에 數萬個의 高機能. 論理 디바이스가 VLSI(very large scale integration) 化되고 있다. 이에 따른 영향으로, 커스텀칩의 設計, 즉 신호처리, 광대역 화상처리용의 專用 프로세서의 設計가 容易하게 되고 있으며, 또한 從來의 複雜한 計算 處理에 있어서 소프트웨어的으로 처리해 오던 것을 VLSI에 의한 하드웨어의 實現이 可能하게 되었다[1] 최초로 H.T Kung[2]는 systolic 알고리즘에 따라 數値의

*正會員, **非會員, 慶應義塾大學 電氣工學科
(Faculty of Engineering Keio Univ.)
接受日字 : 1982年 9月 17日

高速 處理를 위한 VLSI 設計를 하였다. 이러한 경향은 제5세대 컴퓨터를 지향하는 것으로써, 컴퓨터 方式 設計上에 전환점을 마련해 주었으며, 계속 이같은 研究가 활발히 진행되고 있다.[3] 그러나 아직 統計 處理用 소프트웨어 패키지가 VLSI化 되어 있지는 않다.

따라서 本 論文에서는, 統計的 回歸 分析을 위한 소프트웨어 패키지(BMD-biomedical computer program)[4], [5]를 대상으로 하여 VLSI化에 의한 專用 머신의 設計를 試圖하였다. 이러한 統計的 回歸 分析은 지금까지는 소프트웨어 패키지를 사용하기 때문에 복잡한 計算 處理를 해야 하고 多量의 데이터 處理에 대한 速度 低下(實時間 處理)를 가져오며, 一般 使用者가 소프트웨어 패키지를 사용할 때의 많은 번거로움으로 인한 利用 效率의 저하, 코스트面등 많은 문제점을 가지고 있다. 이러한 소프트웨어的 處理의 문제점을 VLSI化에 의한 하드웨어로 해결할 수 있었다.

本 하드웨어 設計 알고리즘은 回歸 分析에 쓰이는 공통적인 演算 要素를 모아 小論理回路인 하나의 셀(cell)로 設計하고, 任意의 回歸 分析에서 필요로 하는 각종 식에 對應하는 셀을, 多數 結合하는 設計 사상을 기초로 하드웨어 量에 制限을 받지 않는 VLSI 칩을 設計했다.

以下, 셀의 內部 構成 및 演算器의 하드웨어 설계와 制御裝置 및 全體的인 시스템의 構成을 記述하고, 끝으로 本 方式에 대한 評價를 하였다.

Ⅱ. 回歸 分析을 위한 計算 알고리즘[6], [7]

回歸 分析은 한 개의 從屬 變數와 獨立 變數의 사이에서 데이터 推定 分析을 행하는 것으로, 獨立 變數가 한 개인 경우를 單回歸 分析, 獨立變數가 複數인 경우를 重回歸 分析이라 한다. 이들 두 單·重回歸 分析의 計算 알고리즘은 같으며 필요한 計算 모듈(module)들에 대한 각 式은 표1과 같고, 여기에서는 單回歸 分析의 경우를 例로 들었다. 이들 記述에서 共通的으로 使用되는 計算 알고리즘으로는 加算·減算·乘算·平方根 등으로 整理할 수 있다. 따라서 回歸 分析의 各 計算値의 出力은 이들 5개의 演算에 의하여 얻을 수 있다.

표 1. 回歸 分析의 計算

Table 1. Computation of regression analysis.

計 算 모 듈	計 算 式
各 變數의 平均値	$\bar{X}=\frac{1}{n}\sum_{i=1}^{n} X_i$, $\bar{Y}=\frac{1}{n}\sum_{i=1}^{n} Y_i$
各 變數의 平方和	$S_{xx}=\sum_{i=1}^{n}(X_i-\bar{X})^2$, $S_{yy}=\sum_{i=1}^{n}(Y_i-\bar{Y})^2$
變數間의 共分散値	$S_{xy}=\sum_{i=1}^{n}(X_i-\bar{X})\cdot Y_i-\bar{Y})$
相關係數	$r=\frac{S_{xy}}{\sqrt{S_{xx}}\cdot\sqrt{S_{yy}}}$
回歸係數	$\hat{b}=\frac{S_{xy}}{S_{xx}}$
定 數 項	$\hat{a}=\bar{Y}-\hat{b}x$
推定回歸値	$\hat{y}=\hat{a}+\hat{b}x$
回歸 變動의 平方和	$S_R=S_{xy}\times\hat{b}$
殘差 平方和	$S_r=S_{yy}-S_R$
不偏分散値	$V_R=S_R$ (回歸不偏分散) $V_r=S_r/n-2$ (殘差不偏分散)
決定係數	$R^2=\frac{\hat{y}}{y}$
重相關係數	$R=\sqrt{\frac{\hat{y}}{y}}$

(但 : "^" 記号는 推定의 意味로 쓰임.)

Ⅲ. 셀(Cell)의 構成 方式

本 論文에서 셀의 構成 方式을 다음과 같이 提案한다.

1) 제1단계의 基本的인 알고리즘은 그림1과 같이 反復法(iterative method)에 의해 構成한다. 왜냐하면 統計 回歸 分析은 反復되는 計算으로 이루어지기 때문이다. (ex. $\sum X_i$, $\sum X_iY_i$ etc.)

그림1은 $\sum_{i=1}^{n} X_i$를 計算하는 모듈로 R_1 (레지스터1)에 데이터가 入力되어지면 R_2 (레지스터 2)에 들어 있는 데이터의 內容과 더해져 그 結果가 R_2에 格納된다. 이와 같은 實行이 X의 데이터數 n만큼 되풀이 되어 최종의 $\sum_{i=1}^{n} X_i$의 計算値가 計算되게 된다. 또 $\sum_{i=1}^{n} Xi$를 하드웨어로써 實現 可能한 다른 方式, 즉 애더(adder)를 많이 使用하여 파이프라인 및 並列 處理에 의한 高

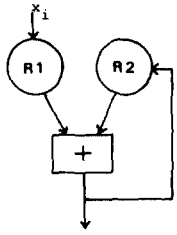


그림 1. 反復法에 의한 $\sum X_i$ 演算

Fig. 1. $\sum X_i$ arithmetic by iterative method.

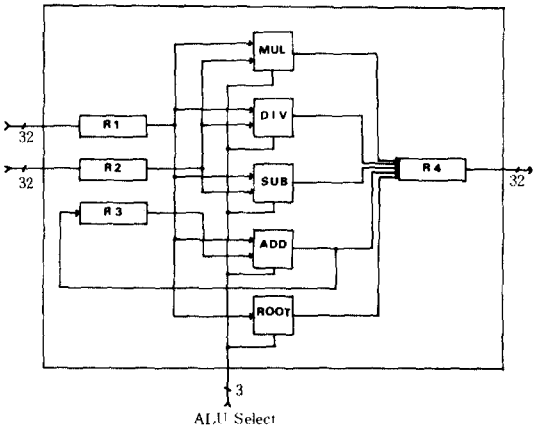


그림 2. 셀의 블럭圖

Fig. 2. Block diagram of a cell.

速 效果를 얻는 方法을 생각할 수 있으나 이러한 方式의 最大 欠點으로는 데이터의 갯수에 따라 애더의 數가 制限을 받는다는 문제를 안고 있다.

따라서 本 論文에서는 反復法을 기초로 II節에서 언급한 5개의 基本 演算 알고리즘을 實現하는 셀을 設計한다.

2) 제2단계에서 셀의 構成은 그림2와 같다. 셀의 內部에 작은(小) 셀로써 구성된 5개의 演算器와 플로우팅 포인트 레지스터 4개로 構成하고, 이들 演算器의 細部 設計는 32비트 플로우팅 포인트 演算 方式에 따라 超高速 演算 디바이스 MIS 및 LSI로 組合하여 高速 演算을 實行할 수 있도록 設計한다. 또한 이들 5개의 演算 機能을 셀內에서 獨自的으로 遂行하기 위해서 3入力 8出力의 디코더에 의해 선택된다. 이러한 셀을 線型的으로 多數 配列함으로써, 하나의 計算 모듈 値를 구하는 경우에 各 셀에서의 演算器 선택과 셀의 갯수는 이 計算 모듈식의 알고리즘으로 부터 決定되어 實行 處理된다.

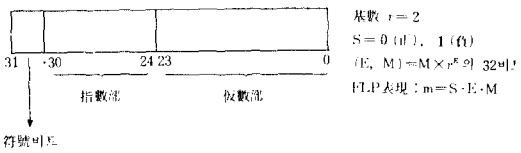
Ⅳ. 演算器의 하드웨어 設計

計算器에 있어서 演算機에 對한 高速化 手法 및 數値의 誤差 방지법등은 끊임없이 硏究되고 있다.

本 論文에서 提案한 그림2의 셀에서 小셀로 구성된 各 演算器는 高速化를 向上시키기 위한 高速 演算 手法[8]을 도입하고 여기에 高性能 素子 MSI 및 LSI를 組合하여 구성하며, 數値 演算 方式은 固定 小數點 方式이 갖고 있는 數値의 精度, 範囲등의 문제를 해결하기 위해서 浮動 小數點 方式을 택하고, 데이터 형식으로는 표2와 같이 1語長 32비트 : 싸인(MSB), 指數部(7비트), 仮數部(24비트), 基數(radix) "2"로 構成하여 設計한다.

표 2. 浮動 小數點 데이터型式

Table 2. Floating point data type.



1. 正規化 浮動 小數點 乘算器[9]

浮動 小數點 方式은 一般 數値를 仮數部와 指數部로 나누어

$$f = M \times r^c \quad (r : 基數)$$

로 表示한다. M은 仮數部이며, c는 指數部로 整數이다. 浮動 小數點 方式에서는 數値(f)를 이같이 하여 計算하며 計算 結果를 다시 一般 數値로 나타내는 것을 正規化라 한다.

一般的으로 乘算式의 表記는 $P = A \times B$로 나타내며 어퍼런드(operand) A는 被乘數, 어퍼런드 B는 乘數이며, 兩 어퍼런드는 指數部(CA, CB), 仮數部(MA, MB), 符號(SA, SB)로 表示하고 다음과 같은 4개의 處理로 나뉘어 實行한다.

스텝 1; 乘算結果 = 0 if A = 0 or B = 0
스텝 2; 指數部의 加算 : CA + (CB − 128)
스텝 3; 仮數部의 乘算 : MA * MB
스텝 4; 正規化를 위한 仮數部의 乘算結果 쉬프트

以上과 같은 處理 알고리즘을 高速化하기 위한 하드웨어 임플리멘테이션은 그림3과 같이 高速 演算 素子를 부가시켜 나타낼 수 있다.

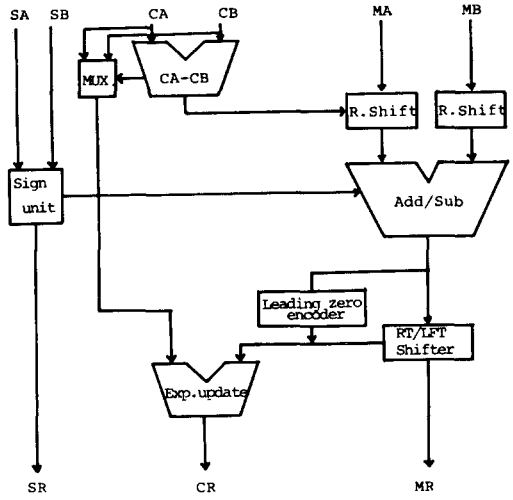


그림 3. 浮動 小數點 乘算器의 블럭圖
Fig. 3. Block diagram of a floating point multiplier.

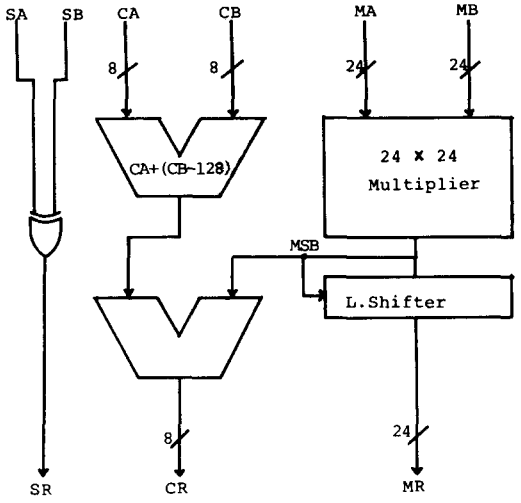


그림 4. 浮動 小數點 加減算의 블럭圖
Fig. 4. Block diagram of a floating point add/sub.

여기에서, 24 × 24멀티플레이어의 細部 構成은 8 × 8 멀티플레이어(67558 : 實行 速度 125ns)들과 1k × 4 PROM₈(6353 : 50ns), 애더(74S182 : 85ns)들로 構成하고 正規化를 위한 래프트 쉬프터는 PAL10H8 (40ns)에 따라 構成한다. 따라서 乘算 實行 時間을 다음과 같이 算出할 수 있다.

$$T_{total} = T_{multiplier} + T_{left\ shift}$$
$$= (125 + 50 + 85) + 40$$
$$= 300\,(ns)$$

2. 正規化 浮動 小數點 加減算

浮動 小數點 加減算 알고리즘[8]은 다음과 같다.

스텝 1 ; 두 개의 指數部 어퍼런드는 仮數部의 쉬프팅에 의해 處理하고, 쉬프트數는 指數의 差로 이 差만큼 仮數部를 右로 쉬프트한다.

스텝 2 ; 두 仮數部 어퍼런드의 加減算

스텝 3 ; 加減算한 結果를 正規化하기 위해 리딩 제로 디텍트(leading zero detect)한 位置만큼 仮數部를 左로 쉬프트한다.

스텝 4 ; 仮數部의 쉬프트한 數만큼 指數部의 數에 더한다.

以上에 대한 하드웨어 블록圖는 그림 4와 같다.

여기에서 라이트 쉬프터(right shifter)는 4비트 쉬프터(25S10)들과 512 × 8 PROM들로 構成되고 라이트 쉬프트하는 實行 速度는 90ns, 仮數部의 加算/減算은 74S381(實行速度 : 60ns), 래프트 쉬프터 (25S10, 6348 : 實行速度 90ns), (PAL16L8 (80ns)이다.

따라서 總實行 時間을 計算하면,

$$T_{total} = T_{right\ shift} + T_{add/sub} + T_{leading\ zero\ detect} + T_{left\ shift}$$
$$= 90 + 60 + 80 + 90$$
$$= 320\,(ns)$$

로 된다.

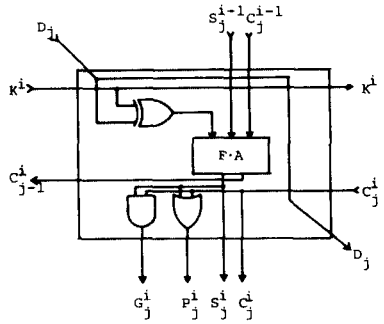
3. 浮動 小數點 셀 配列 除算器

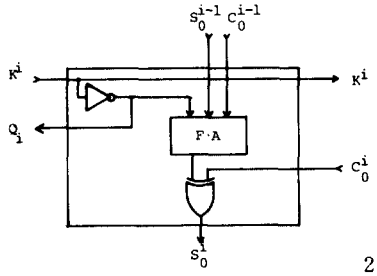
除算式의 表記는 P = A/B로, A를 被除數, B를 除數라 하며 構成 알고리즘은 다음과 같다.

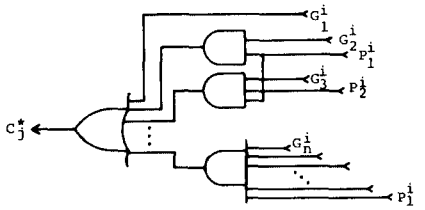
1) 仮數部의 除算은 Cappa-Hamacher[10]가 提案한 행올림 保存先見型 셀 配列 方式에 따라 構成한다.

2) 指數部의 計算은 두 어퍼런드의 指數部差 CA-(CS − 128)로 각각 並列 實行된다. 이 셀 配列 除算器는 그림 5(a)에서의 3종류의 셀을 多數 合成하여 構成하고 필요한 셀의 갯수는 얻어지는 商의 비트數에 比例한다. 여기에서는 商의 비트數를 12비트로 그림 5(b)와 같이 構成한다. 위의 그림과 같이 構成되어진 3종류의 셀의 機能 및 論理式은 다음과 같이 表現할 수 있다.

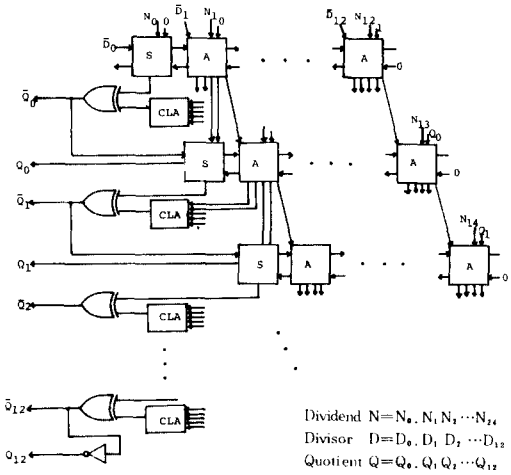
3) A셀 : 3入力 8出力을 가지는 制御入力을 포함하여 행올림 保存先見型 全加算器이며, 그의 論理式은







(a)



(b)

그림 5. (a) 1. A셀, 2. S셀, 3. CLA셀
(b) 24-비트被除數, 12-비트 除數

Fig. 5. (a) 1. A cell, 2. S cell, 3. CLA cell.
(b) 24-bit dividend, 12-bit divisor.

$$S_j^i = S_j^{i-1} \oplus C_j^{i-1} \oplus (D_j \oplus C_i) \quad (1)$$

$$C_{j-1}^i = (D_j \oplus K^i)(S_j^{i-1}) + S_j^{i-1} \cdot C_j^{i-1} \quad (2)$$

로 나타낸다. 그리고 행올림 先見型에 使用되는 행올림과 轉播의 機能을 實現시키기 위해 $G_j^i = C_j^i \cdot S_j^i$ (행올림 기능), $P_j^i = C_j^i + S_j^i$ (轉播機能)의 論理回路가 付加된다.

4) S셀 : 各 行의 左端의 비트는 符號 비트로 使用되어져 仮數部의 符號비트 S_0^i를 作成하고 論理式은

$$S_0^i = (S_0^{i-1} \oplus C_0^{i-1} \oplus \bar{k}^i) \oplus C_0^i$$

로 表記한다.

5) CLA셀 : 행올림 先見型 셀의 出力 C_j^i는 $C_j^i = G_1^i + P_1^i G_2^i + P_1^i P_2^i G_3^i + \cdots + P_1^i P_2^i + \cdots + P_{n-2}^i P_{n-1}^i$의 論理式으로 나타낸다.

一般的으로 n비트로써 構成되는 各 셀 數와 게이트 數는,

- A 셀 : n(n + 1)개
- S 셀 : n + 1개
- CLA 셀 : n + 1개
- XOR 게이트 : n + 1개
- 反轉 게이트 : 1개

와 같다. 따라서 12비트로 택할 때 商의 數는 A셀 156개, S셀 13개, CLA셀 13개, XOR 게이트 13개, 反転 게이트 1개로써 그림 5 (b)와 같이 構成된다. 이때 實行에 소요되는 時間을 算出하면 A셀은 (1), (2)式의 出力 S_j^i, C_{j-1}^i가 나오기까지 3△(△ : 게이트의 딜레이 타임)가 지연되고 G_j^i와 P_j^i에서는 3△ + △ = 4△가 지연된다. 또 각 셀은 행올림 出力 信號 C_j^i의 지연 시간 3△와 (3)式의 同一行 左端 S_0^i에 필요한 시간은 3△로 되기 때문에 全體的인 시간은 3△ + 3△ = 6△가 된다. △行은 商의 비트 $\bar{Q}_i$의 生成에 요하는 시간으로 XOR 게이트의 지연 시간 3△에 A셀과 S셀로부터의 G와 P의 算出 時間 4△(10≦n (bit) ≦64)를 더한 總時間은,

△行 = 3△ + 8△ = 11△ ……(a)

로 된다. 全體의 除算 實行 時間은, 各 行마다 지연시간의 n+1 倍가 됨에 따라

△配列除算器 = (n + 1)△行 + △(反転 게이트 時間)

이 式에 (a)식을 代入하면 (11n + 12)△가 된다. 따라서, 商 12비트의 出力을 얻기 위한 總除算 實行 時間은 (11 × 12 + 12)△ = 144△가 된다.

4. 浮動 小數點 셀 配列 平方根器

一般的인 平方根 알고리즘[11] 은 다음과 같다.

우선, $Q = \sqrt{A} = 0 \cdot q_1 q_2 \cdots\cdots q_n$ (3)

$A = Q^2 = 0 \cdot a_1 a_2 \cdots\cdots a_{2n-1} a_{2n}$ (4)

라고 하면

스텝 1;(2式의 A를 小數點의 位置로 부터 a_1a_2, a_3a_4, $\cdots$, $a_{2n-1}a_{2n}$로 分割

스텝 2;初期値 : k = 1, Ro = a_1a_2, Do = 0.01

스텝 3; $R_k = R_{k-1} - D_{k-1}$

1) $q_k = 1$ if $R_k > 0$

$R_k \leftarrow R_k \cdot a_{2k+1}\, a_{2k+2}$

$D_k \leftarrow q_1 q_2 \cdots\cdots q_k 01$

$R_{k+1} \leftarrow R_k - D_k$

2) $q_k = 0$ if $R_k < 0$

$R_k \leftarrow R_k \cdot a_{2k+1}\ a_{2k+2}$

$D_k \leftarrow q_1 q_2 \cdots\cdots q_k 11$

$R_{k+1} \leftarrow R_k + D_k$

3) $q_k = 1$ if $R_k = 0$

$Q = 0 \cdot q_1 q_2 \cdots\cdots q_k 00\cdots\cdots 0$

一般的으로, k번째의 스텝에 있어 平方根 비트를 q_k라고 하면 다음과 같은 操作을 행하게 된다.

$R_{k+1} \leftarrow R_k \cdot a_{2k+1}\, a_{2k+2} - q_1 q_2 \cdots q_k 01 (q_k = 1$의 경우)

$R_{k+1} \leftarrow R_k \cdot a_{2k+1}\, a_{2k+2} + q_1 q_2 \cdots q_k 11 (q_k = 0$의 경우)

여기에서, 追加 操作 "·"은 全段의 나머지를 2비트 左로 쉬프트하고 右端에 새로운 값의 쌍을 補充함에 따라 實現된다.

本 論文에서 하드웨어는 그림 6 (a)의 기본 CAS 셀(制御 可能 加減算 셀)을 多數 合成하여 그림 6(b)와 같이 構成한다. 이것은 8비트 平方根器의 경우의 例이다.

2n비트數의 平方根을 구하기 위해서는 CAS 셀이 n(n + 1)개가 필요하게 된다. 따라서 仮數部 24비트의 平方根을 구하기 위해서는 156개의 셀로 構成할 수 있으며 總時間은 156 × 3△가 된다(CAS 셀의 지연시간).

위에서 계산은 仮數部의 整數 알고리즘이며, 指數部의 計算 알고리즘은

$$\sqrt{0 \cdot q_1 \cdots q_k \times 2^k} = \underbrace{\sqrt{0 \cdot 8_1 \cdots 8_k}}_{\text{仮數部}} \times \underbrace{(2n)^{\frac{1}{2}}}_{\text{指數部}}$$

위의 式에서, 指數 n의 數가 짝수일 경우는 그대로 "2"로 나누고(論理的으로 1 비트 쉬프트), 홀수인 경우는 1 비트 쉬프트시킴으로써 짝수로 되어 간단하게 處理된다.

實行의 흐름은 그림 7과 같다. 따라서, 總實行 時間은 指數가 짝수일 경우는 156 × 3△가 되고, 홀수인 경우는 156 × 3△ + δ_T (δ_T : 1 비트 쉬프트 時間)로 된다.

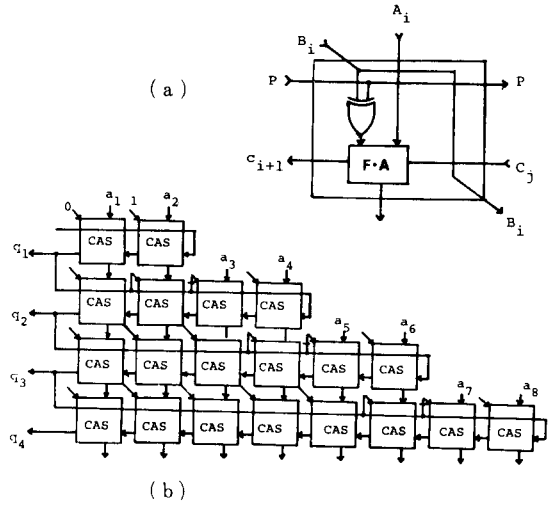


그림 6. (a) CAS(controlled add-subtract) 셀
(b) 예 : 8-비트 平方根 셀 어레이

Fig. 6. (a) CAS(controlled add-subtract) cell.
(b) Example : Cell array for 8-bit square root.

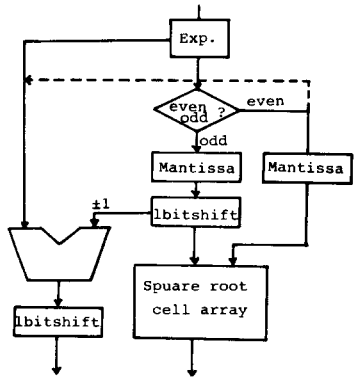


그림 7. 浮動 小數點 平方根 實行 플로우 차아트

Fig. 7. Flow chart of a floating point square root.

V. 시스템 構成

本 論文에서, 回歸 分析 시스템의 概念圖는 그림 8과 같다.

이 回歸 分析 시스템은 셀 配列 構成으로 하고, 블록 안에 있는 각각의 分析 모듈식은 本稿에서 提案한 셀

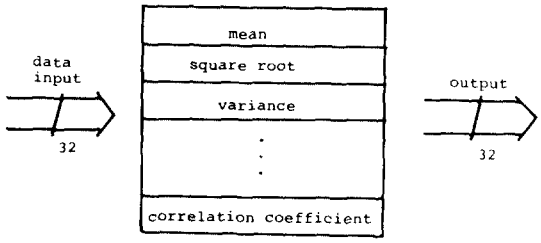


그림 8. 回歸 分析의 블럭圖

Fig. 8. Block diagram of regression analysis.

의 配列로 構成하여, 全體的으로는 多段 機能 셀 配列 시스템이 된다. 이것들의 各 段의 機能 모듈은 獨立的으로 構成되며, 入力 데이터가 同時에 各 段의 계산 모듈에 들어감에 따라 셀이 活性化되어, 各 段에서는 셀을 통해서 하나의 處理 테스크를 順次的으로 實行하여 간다. 이에 따른 하나의 具體的인 構成例는 그림9와 같으며, XY의 共分散의 경우 이 計算 모듈식($Sxy = \Sigma(x_i - \bar{x}) \cdot (y_i - \bar{y})$)에 따라 알고리즘은 다음과 같이 5단계로 나누어 構成된다.

1) 셀 노드 1 : 平均値를 구하기 위한 x, y 데이터의 合算(Σ)
2) 셀 노드 2 : 合算(Σ)値를 데이터數 n으로 除算(平均値 出力)
3) 셀 노드 3 : 各 데이터로 부터 平均値를 減算
4) 셀 노드 4 : 노드 3으로 부터 나오는 두 變數에 대한 結果値의 乘算
5) 셀 노드 5 : 노드 4의 出力値에 대한 데이터數(n)만큼 合算(Σ)

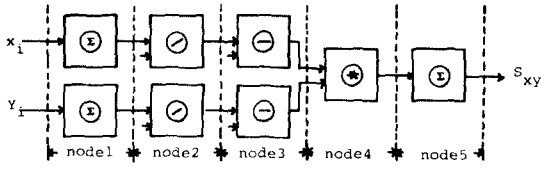


그림 9. 共分散을 위한 셀 어레이

Fig. 9. Cell array for multiple-variance.

이와 같은 構成 方法으로, 셀 配列을 回歸 分析의 各 計算 모듈식에 따라 그림10과 같이 構成하였다.

그리고 셀간의 데이터 轉送 制御에 대하여는 그림11과 같이 래취 래지스터, 3-스테이트 게이트, 콘트롤 로직, SR 플립 플립로 構成하며 데이터는 非同期로 轉送되어 데코더에 의해 실래트(select) 되어진 하

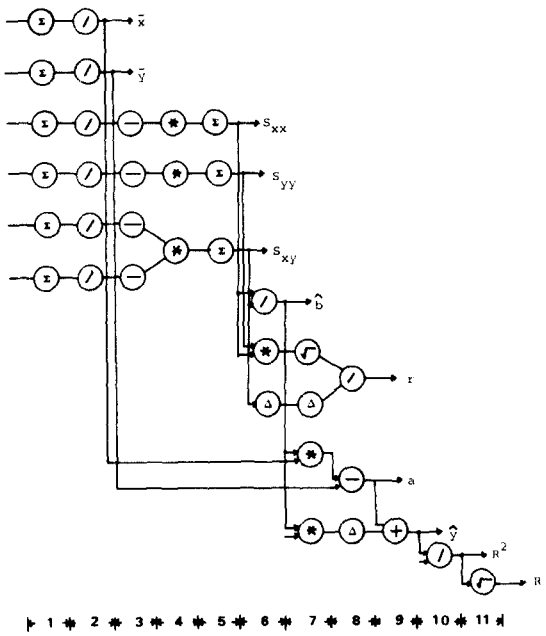


그림10. 各 計算 모듈을 위한 셀 어레이

Fig. 10. Cell array for each computation module.

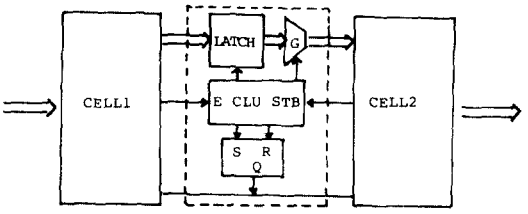


그림11. 셀間의 데이터 轉送 制御

Fig. 11. Data transmission control between cell.

나의 演算器 出力 라이트 시그날(write signal)을 E에, 하나의 셀이 데이터를 라이팅(writing)하면 要求되는 플래그(flag)가 셋트되어 이 信號에 따라 셀의 實行을 開始하고 實行이 終了되면 STB 시그날을 보내 플래그를 리셋트한다. 이때 셀은 플래그가 리셋트 되어진 것을 確認하면 새로운 데이터를 보낸다. 이와 같이 플래그 狀態를 確認하면서 動作 進行하게 된다.

이렇게 하여 構成한 시스템은 既存의 計算機上에 접속되어 回歸 分析을 위한 專用 VLSI 머신으로 그 機能을 수행할 수 있다.

VI. 評　　價

本 論文에서 設計한 머신에 대한 速度 評價値와 실제의 BMD 소프트웨어 패키지로 回歸 分析을 했을 때 實行 時間과의 比較檢討 및 하드웨어 아키텍처上의 擴張性 및 柔軟性 등에 관해서 論한다.

우선, V節에서 設計한 各 演算器의 1語(32비트) 單位 데이터를 處理하는 시간을 整理하면 표 3과 같다.

표 3. 各 演算 유니트의 實行 速度
(△ : 게이트의 지연 시간)

Table 3. Execution speed of each arithmetic unit.
(△ : delay time of a gate)

데이타 샘플數	BMD 패키지 (FACOM M-180)	本 머신
50	350 ms	38·98 μs
100	490 ms	70·98 μs
150	650 ms	102· 9 μs

그림10에 나타낸 全體 實行 테스크는 全 스텝11로 實行되므로 分析에 要하는 全 實行 時間은,

$$T_{total} = T_1 + T_2 + \cdots\cdots + T_{11}$$

이 된다. 이 式에 표3에서 얻은 演算器들의 單位 實行 時間을 각각 代入하면 다음과 같은 一段式을 얻을 수 있다.

$$T_{total} = 640(n+1) + 920 + 1244\triangle$$

보통 게이트 딜레이 시간은 4ns이며 여러데이터 갯수에 대한 全 實行 時間을 얻을 수 있다.

n =데이터 샘플數
單位時間=ns
△=게이트 딜레이 시간

한편, BMD 소프트웨어 패키지를 使用한 FACOM-180上에 데이터 50개, 100개, 150개의 경우 서브프로그램을 움직여 각 데이터 갯수에 대한 CPU 實行 時間의 結果를 얻고 이 結果値와 앞서 얻은 本 方式의 實行 時間과 比較한 結果는 표 4와 같다.

표 4. 實行 時間의 比較

Table 4. Comparison of execution time.

演 算 유 니 트	實行 時間 (單位 : ns)
멀티플리케이션	300
에디션 서브트랙션	320
디 비 젼	144△
스 퀘 어 루 트	156△

표 4에 나타낸 바와 같이 本 머신은 BMD 소프트웨어 패키지로 實行할 때 보다 高速 處理의 性能面에서 뛰어남을 나타내고 있다. 그리고 本 머신은 既存의 計算機에 직접 연결하여 動作이 이루어지기 때문에 단지 入力 데이터를 넣음으로써 回歸 分析 結果의 出力을 얻을 수 있는 特徵을 갖는다. 따라서 BMD 소프트웨어 패키지를 使用할 때의 一般 使用者의 이용 부담율과 복잡성을 크게 감소시키는 큰 利點이 있다. 그러나 本 머신의 演算器 하드웨어 設計에 있어서는 回歸 分析의 高速 演算을 目標로 한 것이며 하드웨어의 설계 영역을 더욱 확장시키는 문제는 今後 계속 硏究가 필요한 것으로 생각된다.

Ⅶ. 結 論

本 論文의 成果는 다음과 같다.

1) 從來의 回歸 分析을 소프트웨어 패키지에 의해서 處理함으로써 速度가 극히 저하되는 결점을 VLSI의 하드웨어 構成으로써 解決하였다.
2) 시스템의 設計方式은 단순한 셀을 規則的으로 配列함으로써 設計를 容易하게 하였다.
3) 소프트웨어 패키지를 使用할 때에 비하여 使用者의 이용 부담율을 대폭 감소시켰다.
4) 本 論文에서 얻어진 한 개의 專用 칩으로써 從來의 汎用 計算機에 직접 연결하면 쉽게 使用할 수 있다.
5) 特殊 目的 專用 칩의 設計 可能性 및 소프트웨어 패키지의 하드웨어化 手法을 提示함으로써 관련 연구에 參考가 될 것으로 기대된다.

參 考 文 獻

[1] Carver Mead and Lynn Conway, *Introduction to VLSI System.* Chapter 8, Addison-Wesley, 1980.
[2] M.J Foster and H.T Kung, "The design of special purpose VLSI chips," *IEEE Trans. Comp.*, pp. 26-40, Jan. 1980.
[3] 相磯秀夫, "將來のコンピュータ" 電子通信學會, vol. 62, no. 11, 1979.
[4] W.R Schucany and D. Minton, "A survey of statistical package," *ACM Computing Surveys,* vol. 4, no.2, pp.65-79, June 1972.
[5] FACOM. BMD package Manual.

[6] 小林龍 ·"相關·回歸分析入門" 日科技連.

[7] 高根芳雄, 柳井晴夫, "多變量解析法", 朝倉書店.

[8] Kai Hwang, 掘越彌譯, "コンピュータの高速演算方式, 近代科學社.

[9] William J. Stenzel, William J. Kubitz and Gilles H. Garcia, "A compact high-speed parallel multiplication scheme," *IEEE Trans. Comp.*, vol.10, pp.948-957, Oct. 1977.

[10] Maurus Cappa and V. Carl Hamacher, "An augmented iterative array for high-speed binary division," *IEEE Trans. Comp.*, vol. C-22, no.2, pp. 172-175, Feb. 1973.

[11] J.C Majithia, "Cellular array for extraction of square and square roots of binary number," *IEEE Trans. Comp.*, vol. C-21, pp. 1023-1024, Sept. 1972.