模糊 C-均值影像壓縮硬體電路之高階合成與模擬

林基源* 林灶生+

*國立勤益技術學院電子工程系講師
+國立勤益技術學院電子工程系副教授

摘 要

由於近年來高壓縮率、高效能的資料壓縮超大型積體電路需求與日俱增，因此本篇論文將首先應用力量-定向排程(Force-Directed Scheduling, FDS)演算法,在沒有增加執行時間下,經由平衡操作的同時性(Concurrency)針對著名的模糊 C-均值(Fuzzy c-Mean, FCM)影像壓縮演算法做操作排程,以達到減少功能單元、暫存器,以及匯流排的需求量，並推導其資料路徑(Data path)硬體電路配置。其次再應用資料壓縮向量量化技術配合 VHDL 硬體語言描述(Modeling) 、模擬其硬體設計。向量量化爲一失真壓縮技術,其目的在於建立一編碼字典(Codebook)使得介於訓練向量與編碼字典中之編碼向量(Codevector)的平均誤差最小,經過訓練迭代過程後即可建立包含若干編碼向量之最佳化編碼字典。於是在傳輸過程中,將一影像切割成不同之向量,每一向量則在編碼字典中找出最接近之編碼向量,並以此編碼向量之代碼傳送至接收端,如此即可大大減少傳輸量,進而提升傳輸效率。由研究結果顯示,本篇論文已成功的完成模糊 C-均值影像壓縮演算法之硬體電路配置，並且成功的利用 VHDL 硬體語言描述、模擬了其硬體設計。

關鍵詞：向量量化、影像壓縮、高階合成。

一、前 言

只要一個人去計算一般數位影像所含的資料量便可以了解，即使是只有幾張數位影像都將構成儲存上的問題。另一個需要影像壓縮的原因是傳輸，假設現在我們要透過電話線傳輸一張低解析度 512×512×8 位元/像素×3 顏色的影像。如果我們使用 baud 爲 9600(baud 即位元/秒)的數據機(Modem),那麼傳輸一張影像所需要之時間約爲 11 分鐘。這對於大部份系統都是無法容忍的。因此,本文將針對著名的模糊 C-均值演算法結合影像壓縮向量量化技術進行其硬體電路高階合成與模擬之研究。

模糊集合理論(Fuzzy set)於 1965 年由 Zadeh[10]教授提出後,已被應用於不同之領域中。在過去幾年,亦有不同之策略應用於影像向量量化壓縮技術上[1-9]。向量量化的目的是做編碼字典的設計,使得介於訓練向量與編碼字典中之編碼向量的平均誤差最小。編碼字典設計可以被視爲一種群集技術。在群集技術中,根據介於訓練向量與類別中心(Classes' center)的平均誤差最小,所有訓練向量可被明確的分類。亦即在向量量化技術中,每一影像向量$\{\mathbf{X}_i, i=1,2,\ldots,n\}$與編碼字典中每一編碼向量$\{\hat{\mathbf{X}}_i, i=1,2,\ldots,c\}$相比較,最接近於原向量$\mathbf{X}$之編碼向量將會被選出,即由編碼字典中選出$\hat{\mathbf{X}}_i$使對所有 i=1,2,...,c, $d(\mathbf{X},\hat{\mathbf{X}}_k) \le d(\mathbf{X},\hat{\mathbf{X}}_i)$,其中$d(\mathbf{X},\hat{\mathbf{X}}_i)$表原影像向量與編碼向量$\hat{\mathbf{X}}_i$間之平均誤差,即

$D(\mathbf{X},\hat{\mathbf{X}}_i)=\frac{1}{c}\sum_{i=1}^{c}(\mathbf{X}-\hat{\mathbf{X}}_i)^2$，若平均誤差收斂至一最小值時,則向量量化器可達最佳化。

平均誤差量測最小化被廣泛的使用於以梯度迭代爲過程之 Generalized Lloyd 演算法(GLA)中[1]。根據上次迭代與最接近鄰居法則(Nearest neighbor rule),GLA 在每一次迭代過程中均去更新編碼字典,同樣於 GLA 迭代過程,模糊理論中之模糊 C-均值演算法,則以歸屬函數(Membership function)取代傳統之明確函數(Crisp function),以求在向量量化技術中提高壓縮效能。是以本文首先以各種不同的影像利用 MATLAB 程式語言實驗測試壓縮系統,看是否系統滿足各種不同特性之影像。

而在合成以及最佳化過程中，高階合成主要是將數位電路的演算法抽象行爲說明(Description)轉換成可實現的暫存器轉移階之明確行爲。而資料路徑合成的效能(Performance)和成本(Cost)則與操作排程(Scheduling)及資源配置(Allocation)緊密相關[11]，且在排程與配置也已被證明爲一 NP-hard[12]的問題。是以本文接著應用 FDS 排程演算法[13]，針對所應用的向量量化模糊 C-均值演算法做操作排程，並藉由排程的結果，推得其資料路徑硬體配置。最後再經由 VHDL 硬體語言描述、模擬其硬體設計，以驗證硬體實作之可實現性，以及判定所得結果之 PSNR (Peak Signal to Noise Ratio)值是否爲可接受。

二、向量量化模糊 C-均值演算法

(一)向量量化的定義

來自於實驗的最佳化向量量化器設計,已於 1980 年經由 Linde,Buzo,and Gray[1]等提出,並且利用群集技術廣泛地研究,目前一般稱此演算法爲 LBG 演算法。一個向量量化器可被定義成對映(Mapping)一個 $\lambda\times\lambda$ -維度 $\mathbf{R}^{\lambda\times\lambda}$ 的歐基里得空間(Euclidean space) Q 成一組 $\lambda\times\lambda$ -維度空間的部份集合 $\mathbf{Y}$;亦即, $Q:\ \mathbf{R}^{\lambda\times\lambda}\rightarrow\ \mathbf{Y}$,其中 $\mathbf{Y}=(\hat{\mathbf{X}}_i, i=1,2,,...,c)$ 爲編碼向量的集合,並且 c 是有限部份集合 $\mathbf{Y}$ 的樣本向量數。

向量量化器也可以被視爲兩個功能的組合:一個編碼器,它觀察輸入樣本向量 $\boldsymbol{X}$ 以及根據最接近鄰居法則經由 $\mathbf{Y}(\hat{\mathbf{X}}_i)$ 中獲得編碼向量的位址,一個解碼器,它使用這編碼位址去取得編碼向量 $\hat{\mathbf{X}}_i$ 。

假如一個誤差測量 $d(\mathbf{X},\hat{\mathbf{X}}_i)$ 代表一個輸入樣本向量 $\mathbf{X}$ 經由編碼向量 $\hat{\mathbf{X}}_i$ 重建的誤差,則最好的對映 Q(Mapping Q)是 $d(\mathbf{X},\hat{\mathbf{X}}_i)$ 爲最小,因此,一個簡單的誤差測量即

$$d(\mathbf{X},\hat{\mathbf{X}}_i)=\left|\mathbf{X}-\hat{\mathbf{X}}_i\right|^2=\sum_{j=1}^{\lambda\times\lambda}(\mathbf{X}_j-\hat{\mathbf{X}}_j)^2 \tag{1}$$

當然,設計一個最佳化向量量化器的目標是去獲得一個由 c 個編碼向量組成的量化器,使得它的平均誤差爲最小。也因此,此一平均誤差可定義成

$$D(\mathbf{X},\hat{\mathbf{X}}_i)=\frac{1}{c}\sum_{i=1}^{c}d(\mathbf{X}-\hat{\mathbf{X}}_i) \tag{2}$$

亦即當其平均誤差最小時,我們可得到一個最佳的量化器設計。

(二)模糊 C-均值演算法 (FCM)

模糊理論是以一數學架構去定義辨識處理之不確定性,自 1965 年 Zadeh[10]發表模糊集合理論以來,模糊集合理論已被應用於不同之領域,諸如醫學影像分析、影像壓縮、圖訊識別等。根據 Bezdek 敘述,模糊群集問題可描述如下:令 $Z=\{z_1,z_2,\ldots,z_x,\ldots,z_n\}$ 爲一未定義之有限資料集,其中 z_x 爲一 p-維之訓練樣本,而模糊群集 $c_1,\ldots,c_c$ 則是根據歸屬函數矩陣 $U=[u_{x,i}]$ 去區分 Z 而得,其中 $u_{x,i}$ 表示 z_x 歸屬於第 i 個模糊群集之可能性程度。

與一般群集技術一樣,模糊群集技術亦使用最小平方差準則,其模糊目的函數爲

$$J_{FCM}=\frac{1}{2}\sum_{j=1}^{c}\sum_{x=1}^{n}(u_{x,j})^m\left|z_x-w_j\right|^2 \tag{3}$$

其中 "$|\bullet|$" 爲歐基里得(Euclidean)距離。在群集分析上,模糊 C-均值爲一相當知名且功能甚強的技術;首先由 Dunn[14]所介紹,而相關之公式及演算法則由 Bezdek[15]所提出,茲將其敘述如下:

模糊 C-均值演算法

步驟 1: 設定群集個數 c、模糊參數 m ($1\le m\le\infty$) 及歸屬函數矩陣 $U^{(t)}$ 之起始值, 並設定$\varepsilon>0$ 及 $t=0$。

步驟 2: 使用 $U^{(t)}$求得群集中心點矩陣 $W^{(t)}=[w_1,w_2,\ldots,w_c]$,其中

$$w_i=\frac{1}{\sum_{x=1}^{n}\left(u_{x,i}\right)^m}\sum_{x=1}^{n}\left(u_{x,i}\right)^m z_x,\quad \text{對每一個 } i。 \tag{4}$$

步驟 3: 使用 $W^{(t)}$計算歸屬矩陣 $U^{(t+1)}=[u_{x,i}]$

$$u_{x,i}=\left[\sum_{j=1}^{c}\left(\frac{|z_x-w_i|}{|z_x-w_j|}\right)^{2/(m-1)}\right]^{-1},\quad \text{對每一個 } x \text{ 及 } i。 \tag{5}$$

步驟 4: 計算$\Delta=\max(|U^{(t+1)}-U^{(t)}|)$,若$\Delta>\varepsilon$,則重回步驟 2,並設定 t=t+1;否則結束作業。

其中模糊參數 m 爲 1 到∞之預選值,在計算群集中心時,它可減少對雜訊之敏感度;當 $m=1$ 時,演算法變成明確群集演算法。此外,m 可減低小 $u_{x,i}$ 值所產生的影響,m 值愈大則影響愈強。

(三)向量量化模糊 C-均值演算法

在影像壓縮裡,向量量化是一種重要的技術,其向量的來源有很多方式,例如它可以是黑白影像中一個 L×M 的矩形,也可以是彩色影像中的$[R(i,j),G(i,j),B(i,j)]$。而向量量化壓縮中,是把已切割的像素方塊當成訓練樣本向量,而不是個別的純量。亦即,一張維度 $N\times N$ 像素的影像,能夠被切割成 n 個向量(方塊),並且每一個向量含有 $\ell\times\ell$ ($\ell<N$)個像素,其中,向量所含像素個數,我們定義爲其維度。接著,向量量化再依據期望最小平均誤差規範對映(Maps)所有訓練樣本向量成一組代表性樣本(Class center)或編碼向量之集合,我們稱之爲編碼字典。由於編碼字典的大小與所有可能之 n 個向量的個數比起來小很

多,因此我們可以達到影像壓縮之目的。其結果之位元率爲$(\log_2 N_c)/k$ 位元/像素,其中 N_c 代表編碼向量的個數,k 則代表向量之維度。

在圖訊識別應用上,同一群集中之樣本將較不同群集間之樣本相關性高。換句話說,群集中之樣本距離將較不同群集間之樣本距離小,在線性準則分析[16]上,群集間散亂矩陣(Within-class scatter matrix)廣泛的應用於群集分離,且於迭代過程將逐漸使目的函數逐漸減小。

接著下來,我們將影像壓縮的向量量化映射至模糊群集技術上,其構想如 Yang 教授[17-18]及 Karayiannis 教授[19]所提。因此,其群集中心點矩陣及歸屬函數矩陣可被修正成

$$\mathbf{w}_i = \frac{1}{\sum_{x=1}^{n}(u_{x,i})^m} \sum_{x=1}^{n}(u_{x,i})^m \mathbf{z}_x, \text{ 對每一個 } i。 \tag{6}$$

$$u_{x,i} = \left[\sum_{j=1}^{c} \left(\frac{|\mathbf{z}_x - \mathbf{w}_i|}{|\mathbf{z}_x - \mathbf{w}_j|} \right)^{2/(m-1)} \right]^{-1}, \text{ 對每一個 } x \text{ 及 } i。 \tag{7}$$

使用(6)及(7)式進行迭代,可將 n 個訓練樣本向量分類成 c 個編碼向量,其向量量化 FCM 演算法如下所示。

向量量化 FCM 演算法

步驟 1: 輸入一組訓練樣本向量集合 $\mathbf{Z}=\{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n\}$,模糊參數 $m(1\le m<\infty)$, 編碼向量個數 c,且隨機設定歸屬函數矩陣狀態 $U=[u_{x,i}]$。

步驟 2: 使用(6)式計算群集中心點矩陣。

步驟 3: 使用(7)式更新歸屬函數矩陣。

步驟 4: 計算 $\Delta = \max\left(\left|U^{(t+1)} - U^{(t)}\right|\right)$,若$\Delta>\varepsilon$,則跳回步驟 2,否則停止作業。

三、向量量化模糊 C-均值演算法之操作排程與硬體配置

(一)操作排程技術

結構的合成(Architecture synthesis)中,排程(Scheduling)是一個非常重要的問題。然排程必須滿足控制資料流序列(Sequencing)圖形中原始的相依性,並且排程也決定實作結果的操作同時性(Concurrency)。因此,排程技術的選擇將影響實作(Implementation)的面積(Area),及結構單元的高利用度(High utilization)或結構單元的低閒置時間。

本節首先介紹兩個歸屬於排程、配置(Allocation)獨立的操作排程技術: (1) 僅可能快(As Soon As Possible , ASAP)排程演算法及 (2) 僅可能慢(As Late As Possible,ALAP)排程演算法。其中 ASAP 排程演算法爲一種不給任何限制的排程技術,而 ALAP 排程演算法則爲一有限制潛伏時間(Latency-Constrained)的排程技術。有關其排程演算法分別敘述如下:

ASAP Algorithm

```
ASAP(Gs(V,E)){
    Schedule v0 by setting t0s=1;
    repeat{
        Select a vertex vi whose    predecessors are all scheduled;
        Schedule vi by setting ti^s = max_{j:(vj,vi)∈E} tj^s + dj;
        }
        until(vn is scheduled);
        return(t^s);
    }
```

ALAP Algorithm

```
ALAP(Gs(V,E), λ̄){
    Schedule vn by setting tn^L = λ̄ +1;
    repeat{
        Select vertex vi whose successors are all scheduled;
        Schedule vi by setting ti^L = min_{j:(vi,vj)∈E} tj^L - di;
        }
        until(v0 is scheduled);
        return(t^L);
    }
```

其中 t^s 爲開始時間(Start time),它必須滿足控制資料流序列圖形中原始的相依性(dependency)。

t^L 表示經由 ALAP 排程演算法計算的開始時間。

$\overline{\lambda}$ 表示潛伏時間(Latency)的上限。

d_i 或 d_j 代表操作執行延遲時間。

(二)力量-定向排程演算法(FDS)

力量-定向排程演算法(Force-Directed Scheduling,FDS)[13]是在沒有增加執行時間下,經由平衡操作(Operations)的同時性以達到減少功能單元(Functional unit)、暫存器(Register),以及匯流排(Buses)的需求量。同時性的平衡將有助結構單元的高利用度(High utilization)或結構單元的低閒置時間。

本節將描述力量-定向排程演算法如何去平衡算術(Arithmetic)和邏輯(Logic)操作的同時性,並且假設所有的操作執行在一個控制步階(Control step ; C-step),鏈結(Chaining;置放兩個資料相依(Data-dependent)操作於相同控制步階(C-step))以及多重的控制步階(Multiple c-step operation)是不被考慮。其排程步驟如下所述:

步驟一:時框(Time frame)的決定

首先找尋儘可能快(ASAP)排程,以及儘可能慢(ALAP)排程,然後結合這兩個排

程以決定每一個操作(Operation)的時框。經由操作可能被指定在介於 ASAP 及 ALAP 控制步階中之任一步階,因此可得到一時框圖形。而其中假設一個操作被指定到任何一個可能的控制步階都具有相同的機率,且指定給每一個操作的盒形面積其總機率爲 1,並且延著時框展開。

步驟二:分佈圖(Distribution Graphs, DG's)的建立

對控制資料流程圖形之每一控制步階的每一操作型態計算其總機率,其所得結果之分佈圖將指明相同操作的同時發生率。而每一個分佈圖,在控制步階 i 中,其機率分佈可經由

$$DG(i) = \sum_{Opntype} \Pr ob(Opn, i) \tag{8}$$

得到。其中 Prob(Opn,i) 代表一個操作在控制步階 i 的機率。

步驟三:計算自身力量("Self" Force)

控制資料流圖形中的每一個操作,有一個與它的時框中每一控制步階相關聯的自身力量(Self Force)。自身力量是一種量化,用以反應在所有操作中,當其指定給任一控制步階時,操作同時發生率的影響。當量化數字爲正時,代表操作同時發生率增加,量化數字爲負則反之。

自身力量的計算,非常類似服從(Obey)虎克定律(Hooke's law) : F=Kx 中彈簧受力的情形般。其中 K 代表彈簧係數(Spring's constant),x 代表位移(Displacement),並且 F 代表引起位移的力量。

更明白的說,對一個已知的操作,它的起始時框(Initial time frame)展開控制步階從 t 到 b(t≤b)中,其控制步階 i 的力可經由下式獲得

$$Force(i)=DG(i)*x(i) \tag{9a}$$

其中 DG(i)是目前的機率分佈值(亦即彈簧係數),x(i)爲操作的機率改變量(亦即彈簧的位移)。並且,關於一個操作指定給控制步階 j(t≤j≤b)的全部自身力量可被簡化成

$$\text{Self Foce(j)}=\sum_{i=t}^{b}[Force(i)] \tag{9b}$$

步驟四:計算前任者(Predecessor)及後繼者(Successor)力量

指定一個操作給一個明確的控制步階,往往將影響在控制資料流圖形(CDFG)中鏈結的操作時框。也就是說這樣的變更將波及到前任者及後繼者操作的時框。

經由力的計算,清楚的反應出,控制步階的指定更有效率並可獲致最理想的排程。以上步驟可被扼要說明如下:

重覆直到所有操作被排程:

步驟 1 : 計算時框

1.1 找尋 ASAP 排程。

1.2 找尋 ALAP 排程。

步驟 2 : 使用下式更新分佈圖形(Distribution Graphs)。

$$DG(i)= \sum_{Opntype} \Pr ob(Opn, i)$$

步驟 3：使用

Force(i)=DG(i) * x(i),以及

Self Force(j)= $\sum_{i=t}^{b}[Force(i)]$

對每一個可能的控制步階計算其自身力量。

步驟 4：加前任者和後繼者力量到

自身力量。

步驟 5：以最小的力來排程操作。

結束作業。

(三)向量量化模糊 C-均值演算法操作排程及硬體配置

爲了應用 FDS 演算法對文中所提向量量化 FCM 作操作排程,其相對的硬體描述語言說明(HDL description)及其對應的控制資料流程圖(CDFG)推導,如圖 1 及圖 2 所示。

依前節力量-定向排程演算法所述,我們將可逐步獲得 FCM 演算法之 ASAP 和 ALAP 排程、起始之操作的時框、分佈圖形,以及經 FDS 排程後之操作的時框和分佈圖形。其分別顯示在圖 3 至圖 7。

其次,依 FDS 排程後之操作的時框可獲得圖 8 向量量化 FCM 演算法排程後控制資料流程圖。最後表 1 則列述有關圖 8 向量量化 FCM 演算法排程後之控制資料操作流程的碼序列(Code sequence)。由此,我們即可推得圖 9 所示的向量量化 FCM 演算法之資料路徑硬體結構配置圖。

```
step1: Input m,c,Z,and randomly initialize U=[u_{x,i}]
    while (iter<0) repeat;
step2: Calculate  W_i
    for   i = 1 to c;
        u_sum = 0 , u_sumz_x = 0;
        for   x = 1 to n;
            u_sum = u_sum+pow(u_{x,i},m);
            u_sumz_x = u_sumz_x+z_x*pow(u_{x,i}, m);
            W_i = u_sumz_x/u_sum;
        end;
    end;
step3: Update u_{x,i}
    for i = 1 to c;
        for x =1 to n;
            dil = 0;
            for j = 1 to c;
                dil = dil+pow(pow((z_x-w_j),2), -2/(m-1));
            end;
            u_{x,i} = pow(pow((z_x-w_i),2), -2/(m-1))/dil;
        end;
    end;
step4: Output codebook
```

圖 1 FCM 演算法硬體描述語言說明(HDL description)

四、模擬結果

在向量量化影像壓縮技術中,編碼字典的設計是一個主要的研究領域。而對本篇論文所應用的 FCM 演算法編碼字典設計，其效能(Performance)可先利用 MATLAB 程式語言實驗與傳統 GLA 演算法(Generalized Lloyd Algorithm)[1] 藉由實際影像(Real image)實驗來做比較。其次，再利用 VHDL 硬體語言在 V-System 環境下對其硬體進行配置、描述(Modeling)與模擬。

爲瞭解整個模擬結果是否滿足各種不同特性之影像，文中分別對 Lena、F16、Girl、Pepper,及 Boy-girl 等不同的實際影像做實驗測試。實驗過程中,訓練樣本向量是取自於 256×256 實際影像所切割成的 4096 個 4×4 非重疊 16-維度之向量(方塊),對這些訓練樣本向量資料分別對映成大小爲 64、128,以及 256 個編碼向量之編碼字典以重建影像。而且影像中每一像素以 8-位元灰階值來代表。由此,其峰值訊號雜訊比對 N×N 影像可定義如下:

$$PSNR = 10 \log_{10} \frac{255 \times 255}{\frac{1}{N^2}\sum_{x=1}^{n}\sum_{y=1}^{n}\left(f_{xy} - \hat{f}_{xy}\right)^2}$$

式中,f_{xy}是原始影像像素灰階值,$\hat{f}_{xy}$爲重建影像像素灰階值,255 則代表影像像素峰值灰階(Peak gray level)。

所有的實驗暨模擬結果如圖 10 及表 2 所示。從 MATLAB 實驗結果得知，著名的模糊 C-均值演算法結合向量量化技術應用於影像壓縮編碼字典設計，能夠產生相當令人滿意的結果。致於其硬體配置 VHDL 模擬結果，其重建品值之 PSNR 值不如 MATLAB 實驗結果，主要是爲了要簡化其硬體設計，於是在整個硬體配置與模擬過程中，其停止規範(Stopping criterion)於 VHDL 描述、模擬過程中是採固定迭代數(本文迭代數取用 iter=40)取代 FCM 中需計算 $\Delta = \max\left|U^{(t+1)} - U^{(t)}\right|$ 來判斷是否小於臨界值 ε (MATLAB 實驗模擬取 0.001)。但依硬體模擬結果比較顯示，簡化硬體設計的結果只犧牲少許的壓縮效能，因此從圖 10 及表 2 知，本論文已成功的應用著名的模糊 C-均值演算法結合影像壓縮向量量化技術來完成其硬體高階合成，並成功的利用 VHDL 硬體語言描述、模擬了其硬體設計。

五、結論與討論

在本篇論文中,一個著名的模糊 C-均值影像壓縮硬體電路之高階合成與模擬被提出。由 MATLAB 程式語言實驗測試知,向量量化模糊 C-均值演算法通常祇花費平均約 26 到 34 個迭代即已達到穩定狀態，且文中所應用的 FCM 演算法不同於傳統的 GLA 演算法,於迭代過程中,模糊 C-均值演算法乃以歸屬函數取代傳統之明確函數(Crisp function),以求在向量量化技術中提高壓縮效能。從 MATLAB 實驗結果亦顯示,本文所應用的 FCM 演算法結合影像壓縮向量量化技術產生之重建影像品質比傳統 GLA 演算法之 PSNR 值平均提升了約 2dB 左右。

且近年來,高壓縮率、高效能的資料壓縮超大型積體電路需求日增,因此本篇論文更針對文中所應用的模糊 C-均值演算法推導其資料路徑(Data path)硬體結構,並利用

VHDL 硬體語言描述(Modeling)、模擬其硬體設計。由模擬結果比較得知,在簡化的硬體設計需求下，其尚可得到優秀的壓縮效能，因此足以顯示本篇論文利用 VHDL 硬體語言結合影像壓縮向量量化技術已成功的描述、模擬了其硬體設計。

然而,向量量化配合模糊理論所找出之解可能陷入一區域最小值。因此,未來本文將繼續研究結合遺傳基因演算法(Genetic algorithm)中近似整體最佳解之特性,以期在影像壓縮研究領域上得到更高之效能。致於實際的影像壓縮超大型積體電路製作更將列爲本研究群下一個研究的目標。

六、參考文獻

[1] Y. Linde, A. Buzo, and R. M. Gray, 1980, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 85-94, Jan., 1980.

[2] R. M. Gray, 1984, " Vector quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4-29.

[3] C. Y. Chang, R. Kwok, and J. C. Curlander, 1988, "Spatial compression of seasat SAR images," *IEEE Trans. on Geoscience and Remote Sensing*, vol.26, pp. 637-685.

[4] A. N. Netravali, 1988, *Digital Pictures: Representation and Compression.* Plenum press, New York.

[5] A. Gersho, and R. M. Gray, 1992, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, Norwell, Ma.

[6] E.A. Riskin, T.Lookabaugh, P.A.Chou, and R. M. Gray, 1990, "Variable rate vector quantization for medical image compression," *IEEE Trans. med. Imaging*, vol. 9, pp. 290-298.

[7] E. Yair, K. Zeger, and A. Gersho, 1992, "Competitive learning and soft competition for vector quantizer design," *IEEE Trans. Signal Processing*, vol.40, pp. 294-309.

[8] K.Zeger, J.Vaisey, and A.Gersho, 1992, "Globally optimal vector quantizer design by stochastic relaxation," *IEEE Trans. Signal Processing*, vol. 40, pp. 310-322.

[9] L. L. H. Andrew, and M. Palaniswami, 1996, " A unified approach to selecting optimal step lengths for adaptive vector quantizers," *IEEE Trans. Commun.*, vol. 44, pp. 434-439.

[10] L. A. Zadeh, 1965, "Fuzzy set," *Inform. Control*, vol. 8, pp. 338-353.

[11] M. C. McFarland, A. C. Parker, and R. Compasano, "Tutorial on high-level synthesis," *in proc. 25th DAC*, June 1988., pp. 330-336.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide for the Theory of NP-Completeness.* San Francisco, CA: Freeman, 1979.

[13] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of ASIC's," *IEEE Trans. Computer-Aided Design,* vol. 8, no. 6, pp.661-678, June 1989.

[14] J. C. Dunn, 1974, "A Fuzzy Relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol.3, no.3, pp. 32-57.

[15] J. C. Bezdek, 1973, " Fuzzy Mathematics in Pattern Classification," *Ph. D. Dissertation, Applied Mathematics*. Cornell University, Ithaca, New York.

[16] K. Fukunaga, 1972, *Introduction to statistical pattern recognition*, New York, Academic.

[17] M. S. Yang, 1993, "On a class of fuzzy classification maximum likehood procedures," *Fuzzy Sets Systems* vol. 57, pp. 365-375.

[18] M. S. Yang, and C. F. Su, 1994, "On parameter estimation for normal mixtures based on fuzzy clustering algorithms," *Fuzzy Sets Systems*, vol. 68, pp.13-28.

[19] N. B. Karayiannis, P. I. Pai,1995, " Fuzzy Vector Quantization Algorithms and Their Application in Image Compression," *IEEE Trans. Image Processing*, vol. 4, No.9, pp.1193-1201.

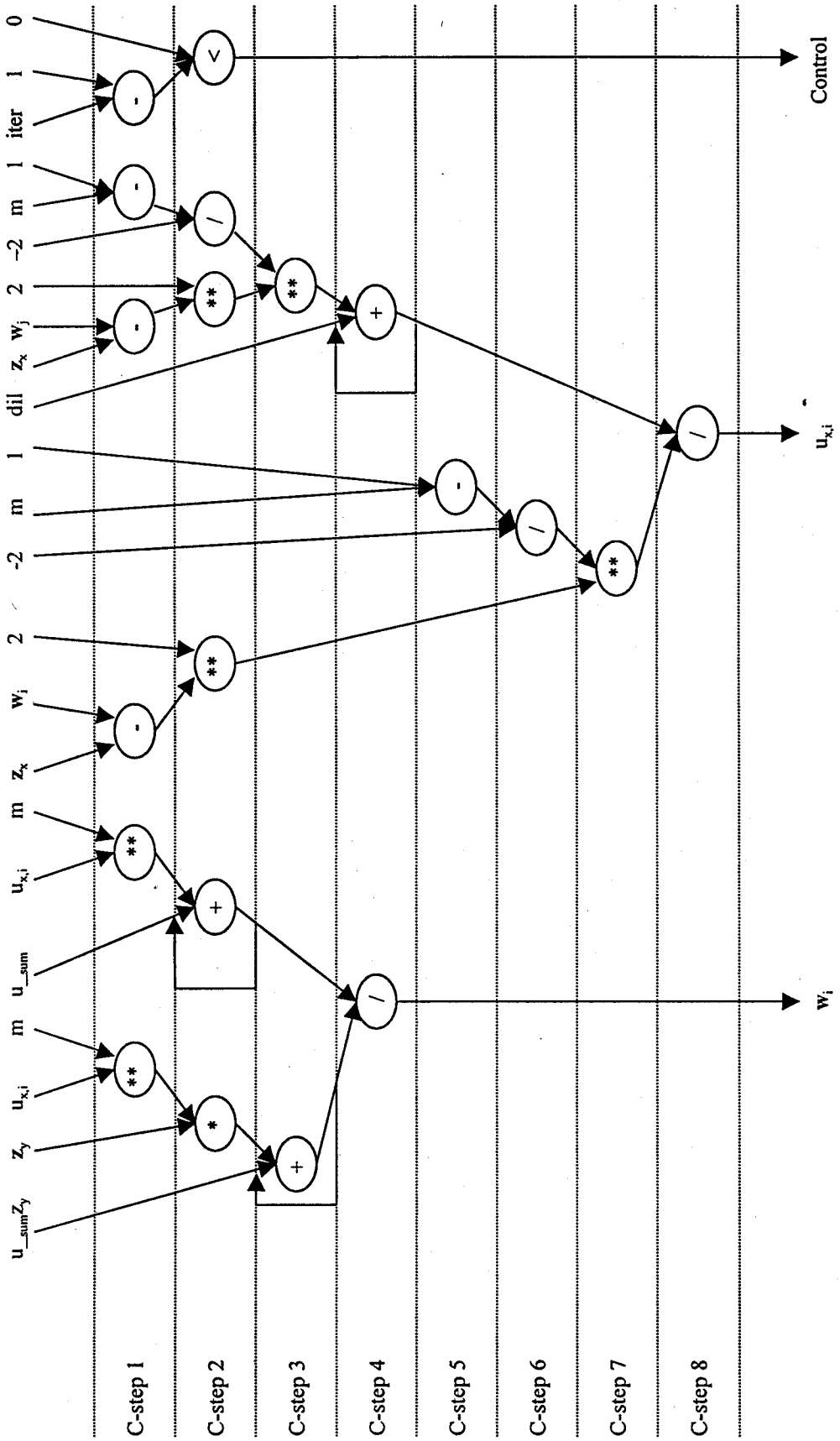

圖 2 FCM 控制資料流程圖

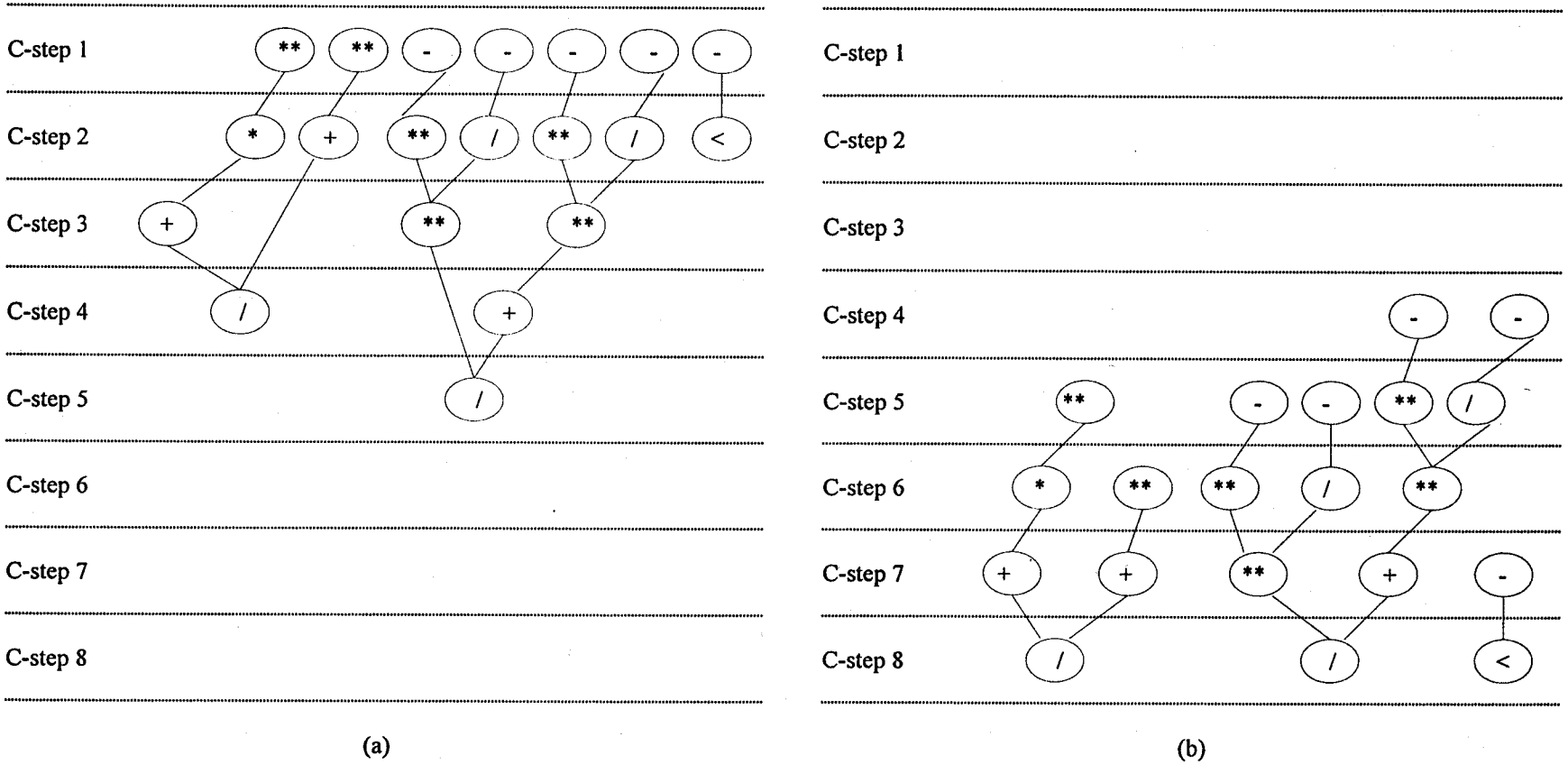

圖 3(a) FCM 之 ASAP 及(b)ALAP 排程

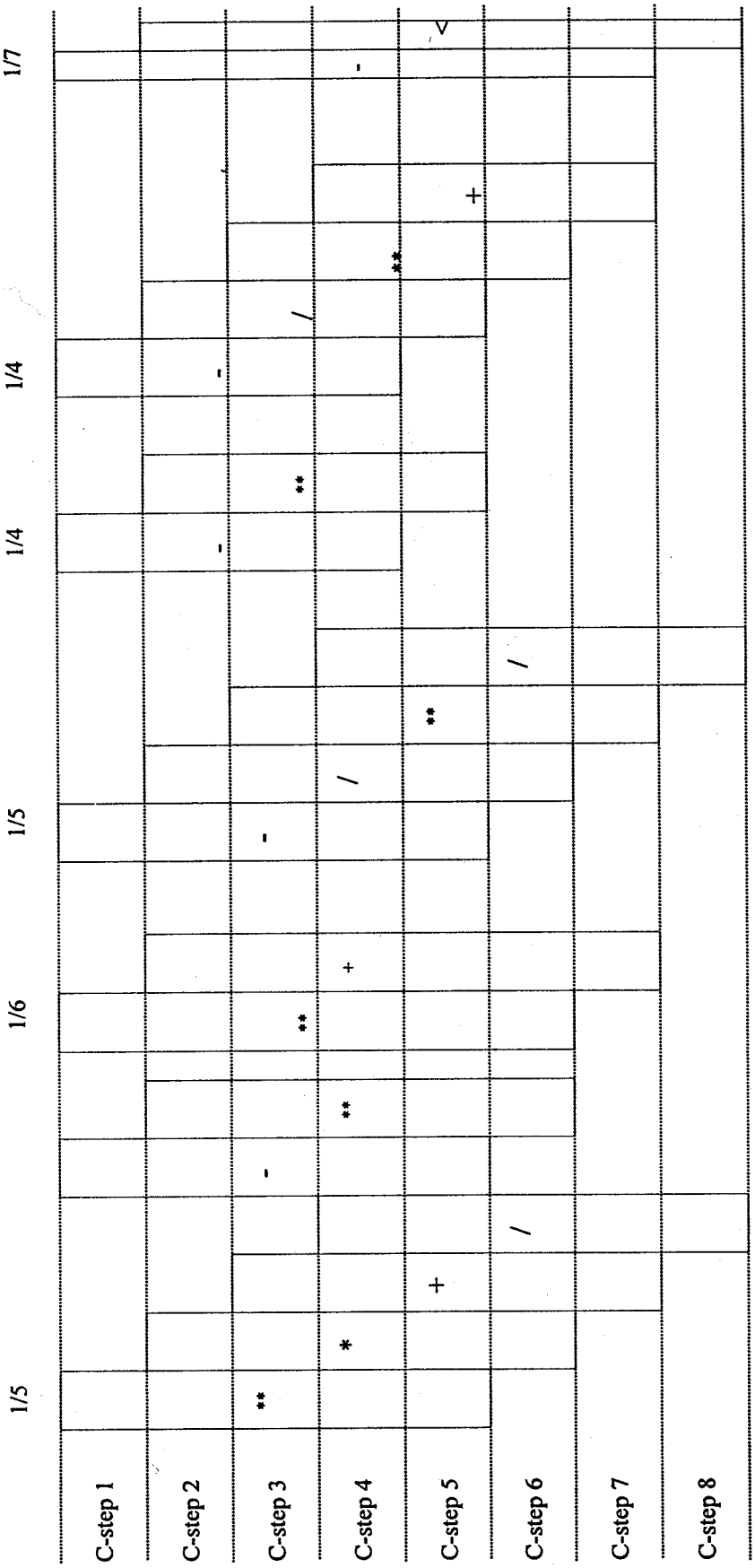

圖 4 FCM 起始之操作的時框

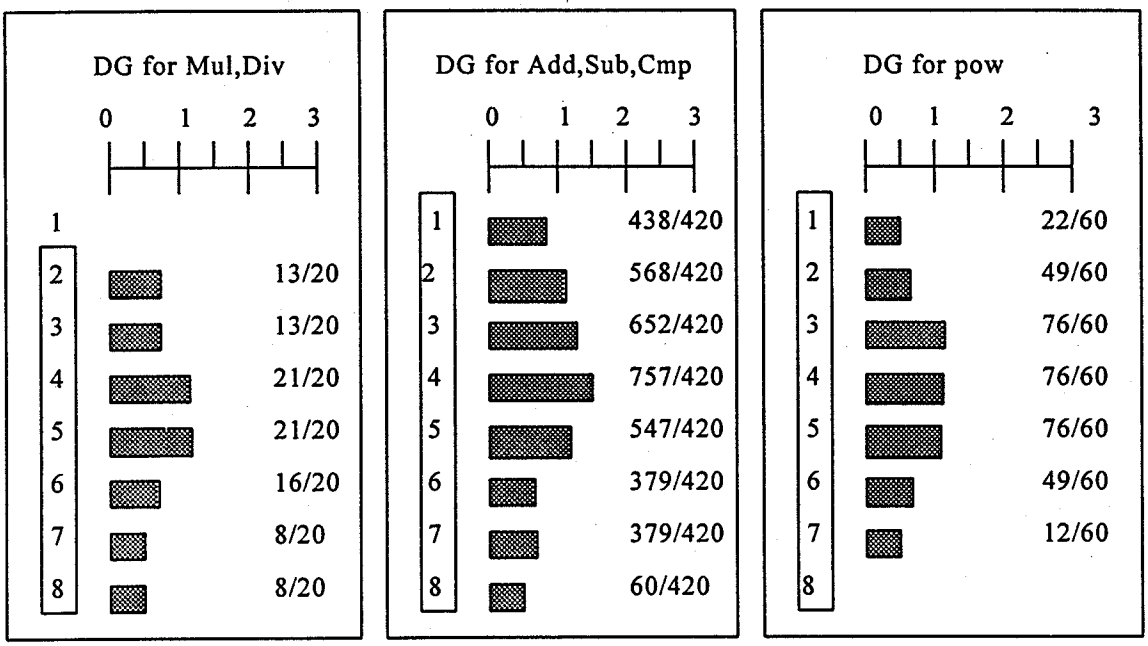

圖 5 FCM 起始之分佈圖形

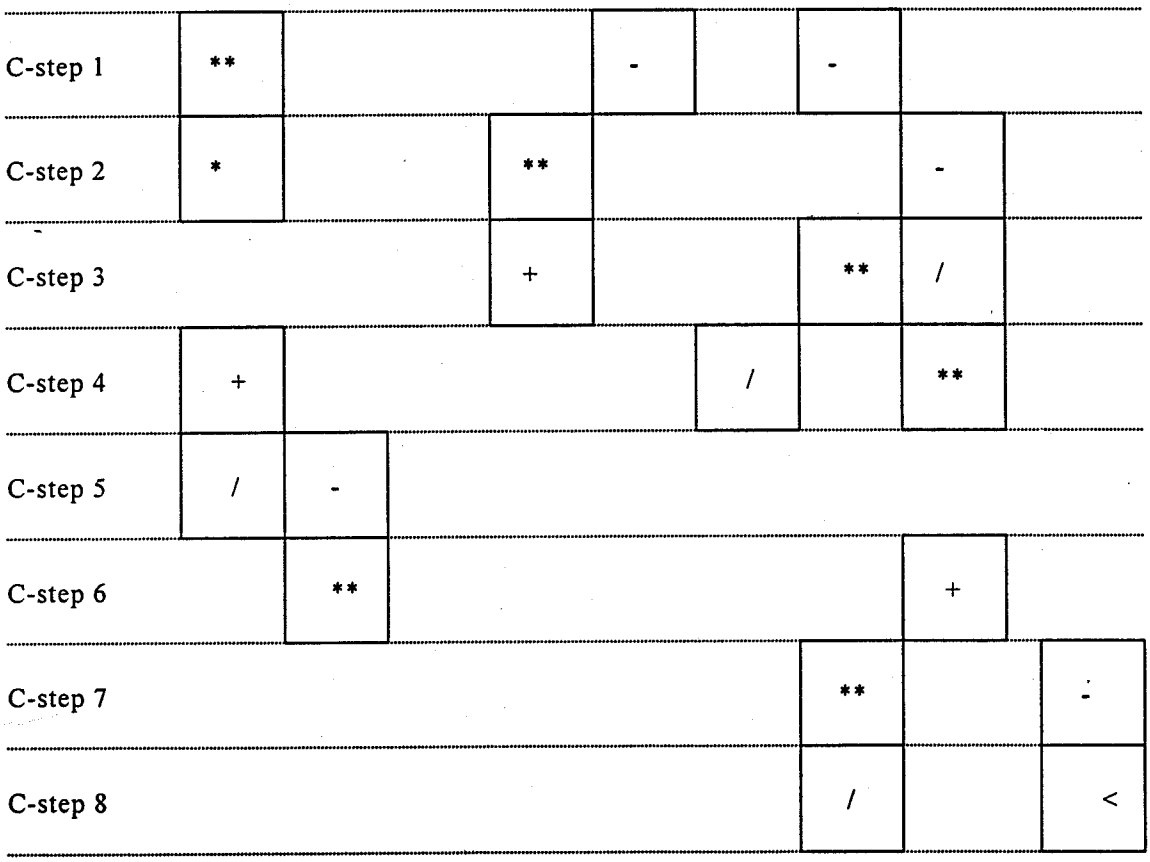

圖 6 FCM 經 FDS 排程後之操作的時框

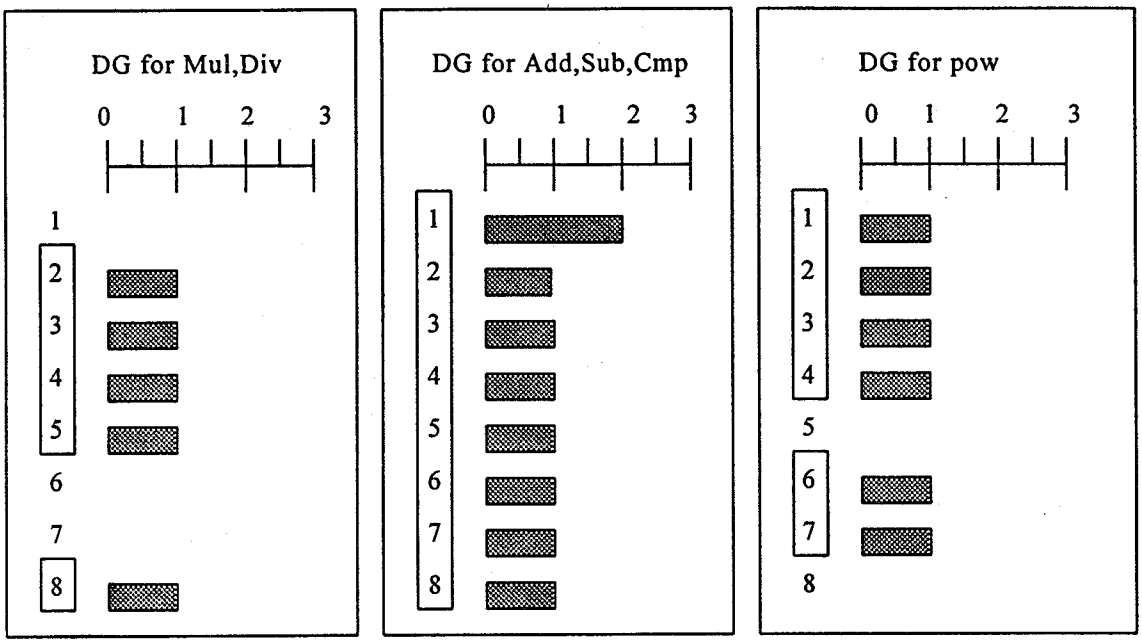

圖 7 FCM 經 FDS 排程後之分佈圖形

表 1 FCM 操作的碼序列(Code sequence)

1	$V_1=U_{x,i}**m$	FU_1;	$V_2=m-1$	FU_2;	$V_3=z_x-w_j$	FU_3;
2	$V_4=z_x*V_1$	FU_4;	$V_5=U_{x,i}**m$	FU_1;	$V_6=m-1$	FU_2;
3	$V_7=u\_sum+V_5$	FU_2;	$V_8=V_3**2$	FU_1;	$V_9=-2/V_6$	FU_4;
4	$V_{10}=u\_sum z_x+V_4$	FU_2;	$V_{11}=-2/V_2$	FU_4;	$V_{12}= V_8**V_9$	FU_1;
5	$V_{13}=V_{10}/V_7$	FU_4;	$V_{14}= z_x-w_i$	FU_2;		
6	$V_{15}= V_{14}**2$	FU_1;	$V_{16}=dil+V_{12}$	FU_2;		
7	$V_{17}= V_{16}**V_{11}$	FU_1;	$V_{18}=iter-1$	FU_2;		
8	$V_{19}=V_{17}/V_{15}$	FU_4;	$V_{20}=V_{18}<0$	FU_2;		

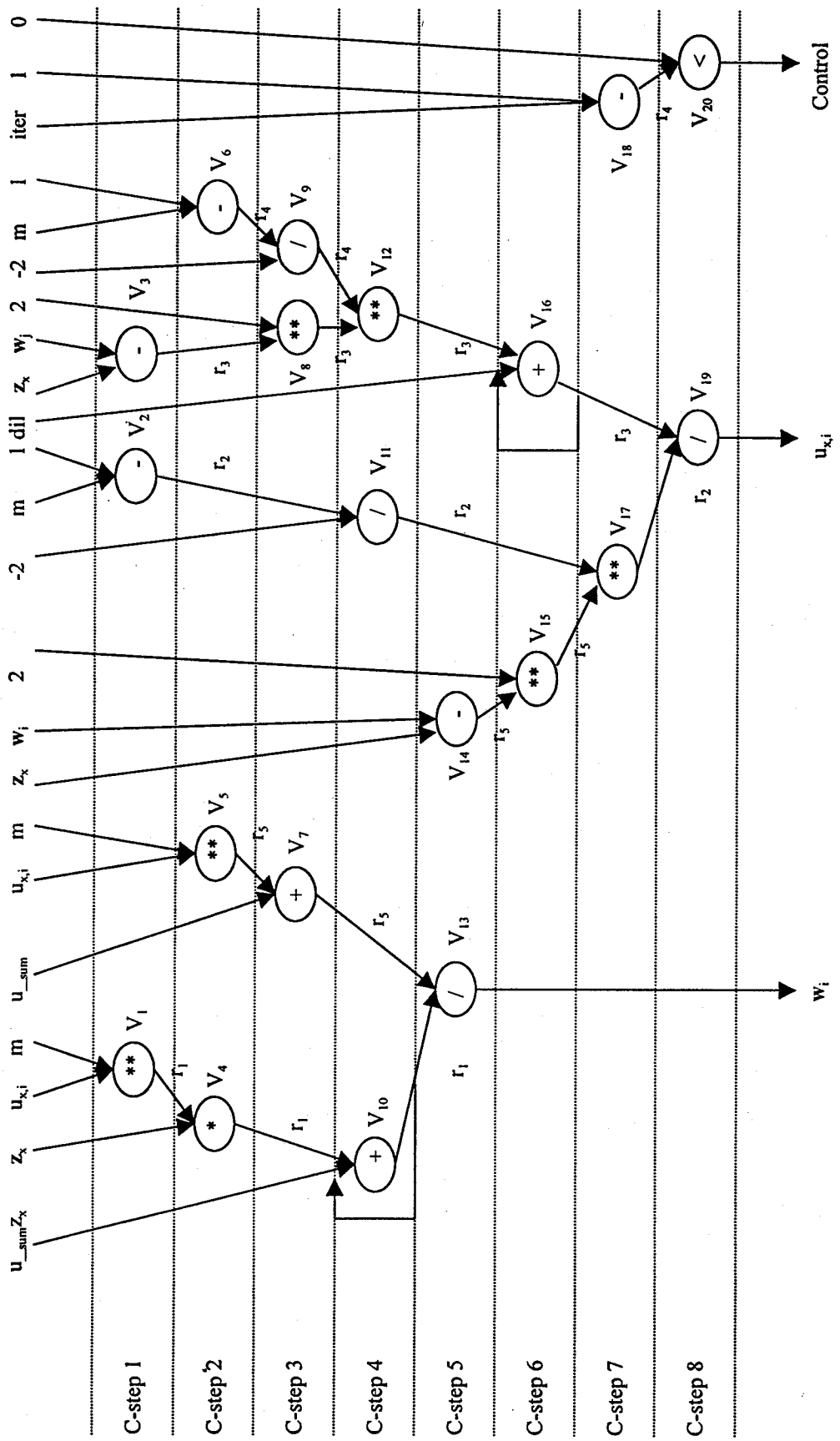

圖 8 FCM 經 FDS 排程後之控制資料流程圖

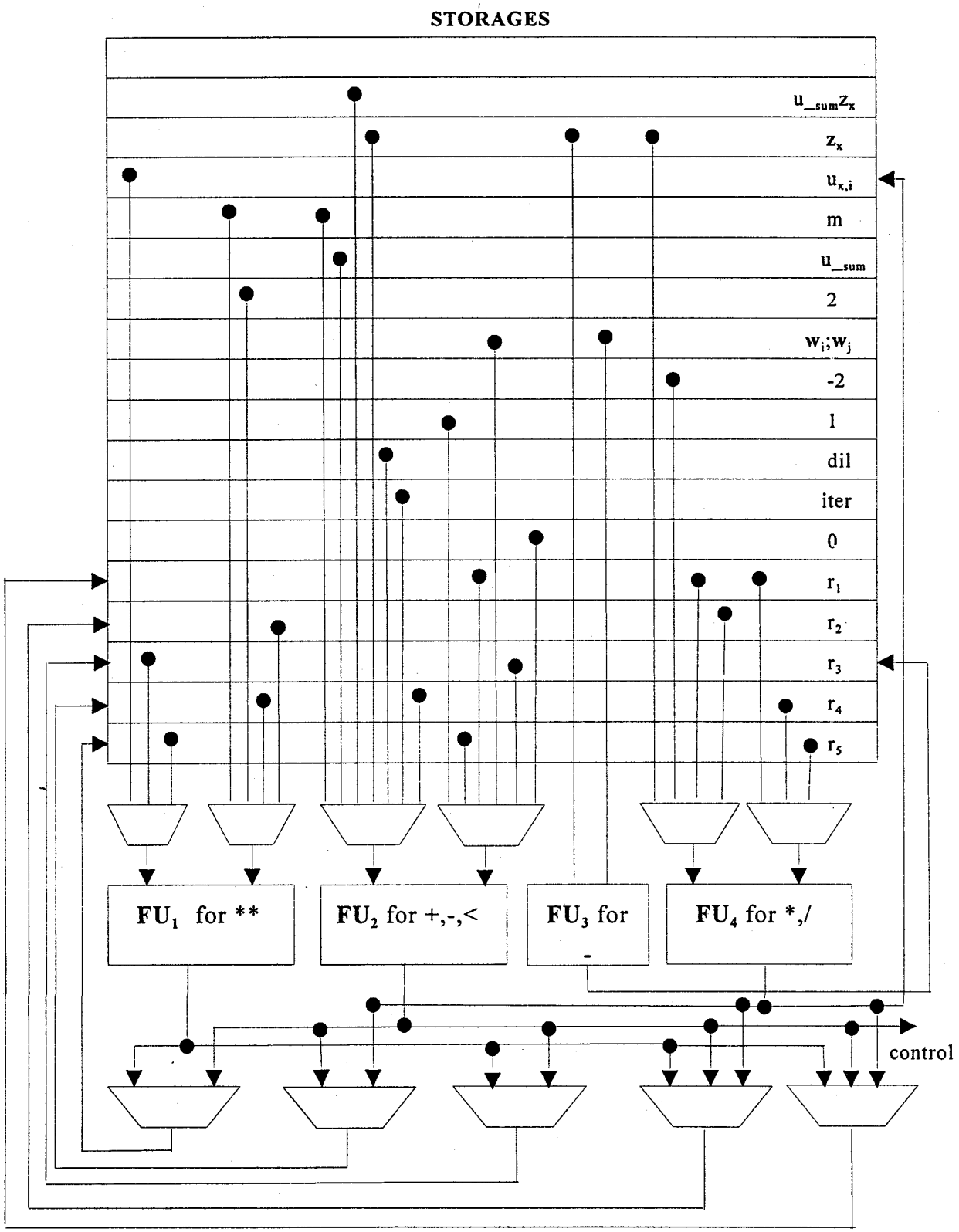

圖 9 FCM 資料路徑硬體結構配置圖

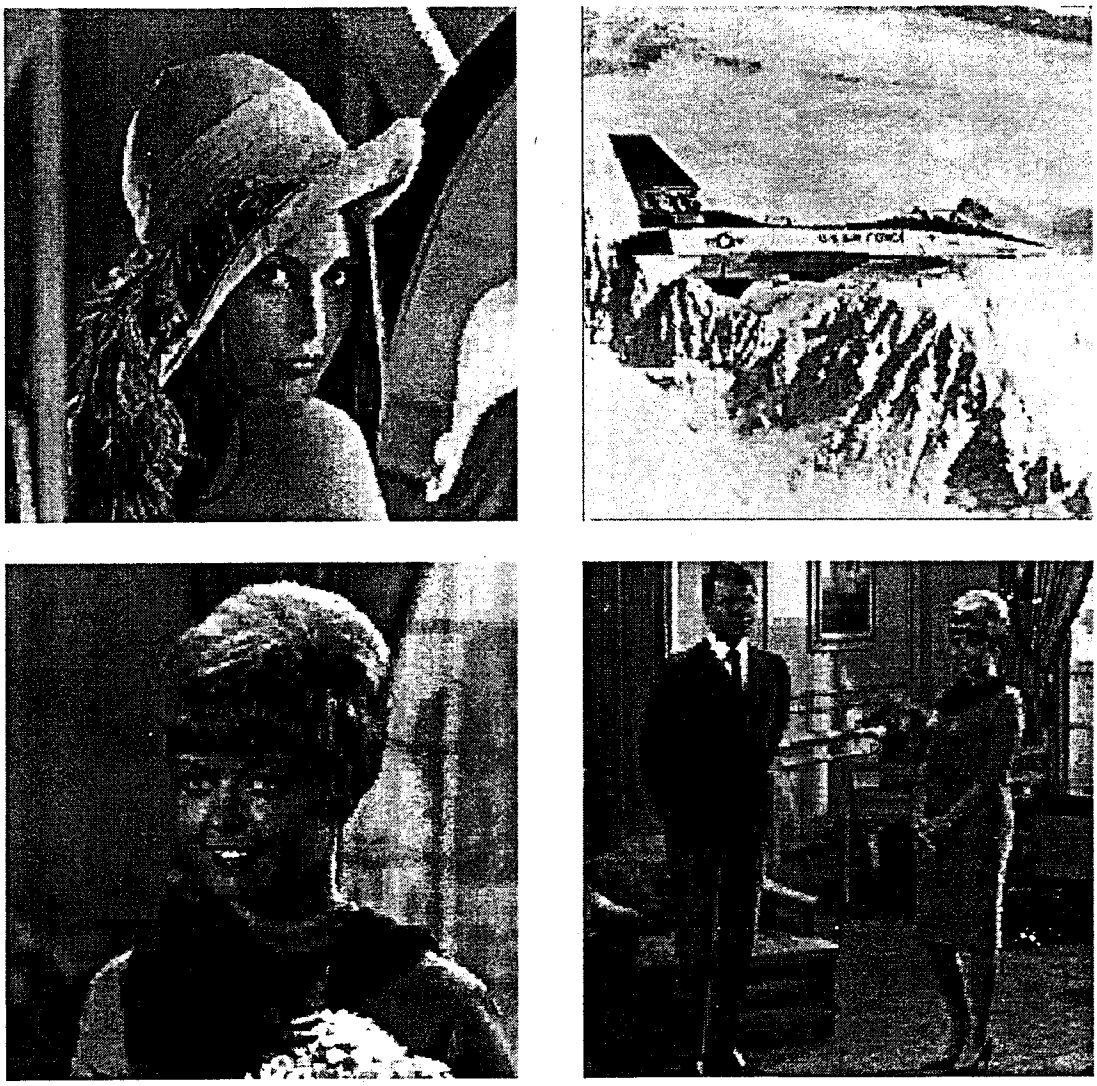

圖 10 VHDL 於編碼字典大小 c=256 時對各種測試影像模擬重建所得之結果影像

表 2 各種影像暨大小編碼字典實驗、模擬重建所得結果影像之 PSNR 值

Codebook Size	64	128	256
Lena	26.555	27.799	29.169
F16	25.632	26.821	28.322
Girl	29.454	30.521	31.745
Pepper	26.145	27.577	29.263
Boy-girl	30.198	31.542	33.123

(a) MATLAB 實驗結果 ($\varepsilon = 0.001$)

Codebook Size	64	128	256
Lena	26.447	27.520	28.679
F16	25.446	26.481	27.822
Girl	29.101	30.230	31.228
Pepper	26.077	27.432	28.785
Boy-girl	29.891	31.017	31.981

(b) VHDL 模擬結果 (iter = 40)