發明專利說明書 200405208

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：92113898

※ 申請日期：92.5.22　　※IPC 分類：H04L 9/38

壹、發明名稱：（中文/英文）

無向量/向量處理器

A SCALAR/VECTOR PROCESSOR

貳、申 請 人：（共 1 人）

姓名或名稱：（中文/英文）

荷蘭商皇家飛利浦電子股份有限公司

KONINKLIJKE PHILIPS ELECTRONICS N. V.

代表人：（中文/英文）

J. L. 凡 德 渥

J. L. VAN DER VEER

住居所或營業所地址：（中文/英文）

荷蘭愛因和文市格羅尼渥街 1 號

GROENEWOUDSEWEG 1 5621 BA EINDHOVEN NETHERLANDS

國　籍：（中文/英文）

荷蘭　THE NETHERLANDS

參、發明人：（共 3 人）

姓　名：（中文/英文）

1. 柯尼利斯 赫曼諾斯 凡 伯克
   CORNELIS HERMANUS VAN BERKEL
2. 派崔克 彼德 伊利莎白 謬偉森
   PATRICK PETER ELIZABETH MEUWISSEN
3. 諾 安金
   NUR ENGIN

住居所地址：（中文/英文）

1. 荷蘭愛因和文市普羅何斯蘭路 6 號
   PROF. HOLSTLAAN 6, 5656 AA EINDHOVEN, THE NETHERLANDS
2. 荷蘭愛因和文市普羅何斯蘭路 6 號
   PROF. HOLSTLAAN 6, 5656 AA EINDHOVEN, THE NETHERLANDS
3. 荷蘭愛因和文市普羅何斯蘭路 6 號
   PROF. HOLSTLAAN 6, 5656 AA EINDHOVEN, THE NETHERLANDS

國　籍：（中文/英文）

1. 荷蘭　THE NETHERLANDS
2. 荷蘭　THE NETHERLANDS
3. 荷蘭　THE NETHERLANDS

肆、聲明事項：

☐ 本案係符合專利法第二十條第一項☐第一款但書或☐第二款但書規定之期間，其日期為： 年 月 日。

☑ 本案申請前已向下列國家（地區）申請專利：

1. 歐洲專利機構；2002 年 05 月 24 日；02077034.3
2.
3.
4.
5.

☑ 主張國際優先權(專利法第二十四條)：

【格式請依：受理國家（地區）；申請日；申請案號數 順序註記】

1. 歐洲專利機構；2002 年 05 月 24 日；02077034.3
2.
3.
4.
5.

☐ 主張國內優先權(專利法第二十五條之一)：

【格式請依：申請日；申請案號數 順序註記】

1.
2.
3.

☐ 主張專利法第二十六條微生物：

☐ 國內微生物 【格式請依：寄存機構；日期；號碼 順序註記】

☐ 國外微生物 【格式請依：寄存國名；機構；日期；號碼 順序註記】

☐ 熟習該項技術者易於獲得，不須寄存。

玖、發明說明：

【發明所屬技術領域】

本發明關係一無向量/向量處理器。

【先前技術】

第三代無線通信標準，如UMTS/FDD、TDD、IS2000，及TD-SCDMA，以非常高頻率操作。3G行動通信標準如UMTS的數據機(收發器)要求大於GSM約100倍以上的數位信號處理功率。希望使用一可程式結構完成該種標準的收發器以便能處理不同標準及能彈性配合新標準。使用傳統DSP技術以傳統頻率運算可能需要30 DSP以提供需要性能。可以了解那樣一種方法比較單標準傳統硬體基準的收發器的方法既不省錢也不減少功率。

一已知增加處理器性能的方法為使用一向量結構。在一向量處理器中，一向量由一個以上的資料元件組成，例如十六個16位元元件。處理器的一功能單元由一指令啟動，平行運算該向量的所有各別資料元件。使用一管線向量處理器，便可進一步增加性能。

向量處理器傳統上主要用於科學處理。原則上，向量處理器也可用於信號處理。不過，傳統向量處理器結構對於非100%向量化的應用無效，由於牽涉到所謂的Amdahl定律。本定律說明向量處理器向量化P處理元件獲得全面增速，如可向量化的部份碼的函數(f)，等於$(1-f+f/P)^{-1}$。表示如果50%的碼可以向量化，便可獲得小於2的總增速(代替理論最大增速32)。因剩餘50%碼不能向量化，所以這部份的碼不能增速。既使90%的碼可以向量化，增速因數仍然小

於8。用於消費者電子應用，特別是行動通信，只有獲得大增速向量處理器的額外成本才能被接受。

【發明內容】

本發明的一目標為提供一處理器結構，更適合於高性能任務，特別是行動通信系統的信號處理。

要達成本目標，一無向量/向量處理器包括複數個功能單元其中至少一功能單元包括一向量部份用於運算至少一向量及一無向量部份用於運算至少一無向量，功能單元的向量部份及無向量部份的配置由無向量部份共同運算，以提供及/或消耗至少一由功能單元的向量部份要求及/或供應的無向量。

本發明人已了解為了突破Amdahls定律，資料非向量化部份也必須以有效方法加以處理。非向量化的部份以無向量處理。通常這些無向量係由向量運算產生及/或消耗。例如：向量的所有元件的和或最大值，選擇第一(或最後)向量元件。在另外的情況中，無向量資料可獨立於向量運算。較佳處理無向量資料，至少處理器的一功能單元不只包括一向量部份也要包括一無向量部份。無向量部份提供及/或消耗一由功能單元的向量部份要求及/或供應的無向量。如此，處理器的無向量部份可準備或進一步處理由向量部份要求或產生的無向量，確保向量部份可更連續流動處理向量。必須注意美國專利5,659,706揭露一無向量/向量處理器具有一分離的無向量處理器部份及一向量處理器部份。配置各處理器部份成為功能單元。不過，在無向量處理器部

份的功能單元及向量處理器部份的功能單元之間並沒有密切共同運算。兩者完全獨立運算。

如申請專利範圍第2及3項所述，功能單元的向量部份及無向量部份分別配置成各自管線。如此增加處理器的向量及無向量部份的性能。

如申請專利範圍第4項所述，管線可獨立配置。如此能達成一最佳的向量管線結構用於原資料處理，同時選擇另外的最佳無向量管線結構以便較佳地消耗/生產向量處理用的無向量。這種配置能力增加性能及簡化處理器的程式處理。並能減少程式碼。

如申請專利範圍第5項所述，至少一管線以一指令接一指令的方式配置。如此，可以進一步增加性能及進一步減少碼。

如申請專利範圍第6項所述，無向量/向量處理器使用具有各功能單元的分離區段的VLIW指令控制。較佳地，VLIW指令包括分離指令用於功能單元的無向量部份及向量部份。如此，兩部份都能最佳地執行其任務。

如申請專利範圍第8項所述，管線也可經VLIW指令配置。這是一種以一指令接一指令方式配置管線的有效方法。

如申請專利範圍第9項所述，管線包括與連接功能單元數目相等的管線路徑。各功能單元連接一路徑。該種路徑，例如，作為功能單元廣播向量(或無向量管線廣播無向量)的路徑。如申請專利範圍第10項所述，功能單元指令指示從向量管線路徑該單元必須消耗一向量(及/或從無向量管

260

線路徑該單元必須消耗一無向量)。或者，一功能單元指令能指示向量管線路徑必須產生一向量輸出(及/或無向量管線路徑必須產生一無向量)，而在下一循環連接該路徑的功能單元根據接收該單元的指令消耗產生的資料。如此，便可獲得完全的管線配置能力，同時保持一合理的網路定址位準即是每一管線只有一路徑。可以了解，由於一路徑及一功能單元之間固定關係，顯示一路徑等於顯示一功能性單元。

如申請專利範圍第11項所述，一偏移單元為一功能單元的例子，其最佳化功能單元無向量部份及向量部份之間共同運算。

為了達成本發明的目標，處理系統包括一無向量處理器及一如申請專利範圍第1項之無向量/向量處理器，其中配置無向量/向量處理器成為無向量處理器的一共處理器及配置無向量處理器以控制無向量/向量處理器，配置無向量/向量處理器的無向量部份用於執行循環內無向量處理及配置無向量處理用於執行不規則，循環外無向量處理。藉由無向量/向量處理器無須處理不規則無向量運算，便可大部份克服Amdahl定律。根據本發明的無向量/向量處理器較理想適合處理循環無向量運算，其中無向量及向量部份之間的密切共同運算確保向量處理儘可能連續。較理想，可以選擇技術(如CMOS技術、運算頻率等)用於兩種處理器，以提供一成本節省的系統。

本發明之這些及其他特徵參考以下所述的具體實施例可

以更明白。

【實施方式】

圖1顯示一較佳結構，其中使用根據本發明的無向量/向量處理器。在本配置中，三個主組件經一匯流排110連接。連接這三個組件的匯流排110為任何適合的匯流排，例如，AMBA高速匯流排(AHB)。主組件為：

- 根據本發明的可程式無向量/向量處理器120，包括功能單元及一區域資料記憶體(稱為向量記憶體，圖1)，
- 一微控制器或DSP子系統130，包括有限晶片上程式及資料記憶體，
- 一介面塊140。

無向量/向量處理器120主要用於規則「重」負載處理，特別內循環內處理。無向量/向量處理器包括向量處理功能。如此，提供執行碼的向量化部份的大量平行性。所有絕大部份的信號處理由無向量/向量處理器的向量部份執行。例如，利用一個32相同處理元件的陣列執行相同指令，提供大量平行性。結合一32字寬記憶體介面達到一前所未有的低成本及一般功率消耗的可程式性能位準。不過，完全發展這樣的平行處理並不一定可行，因為許多演算法並不具有充分正確形式的資料平行性。根據Amdahl定律，在碼直接向量化部份的向量化後，大部份時間耗費在剩餘碼上。剩餘碼可分成四類：

- 位址關係指令(如：增加一指標至一圓形緩衝器，使用模定址)

- 規則無向量運算(即，相對於向量處理器主循環的無向量運算)
- 循環運算
- 不規則無向量運算

這些種類的各碼成分完全根據執行的演算法而定。例如，Golay相關器(用於P-SCH搜尋)要求許多指令關係位址，但是其他演算法如Rake並不需要。不過，所有演算法的一共同性質本發明人已作研究，即是不規則無向量運算的成分非常有限。這性質容許無向量/向量處理器(120)及微控制器或DSP (130)之間的任務分開。

根據本發明的結構克服前三個問題，藉由併入無向量處理功能於無向量/向量處理器120結合向量處理。第四問題可藉由使用一分離微控制器或DSP 130執行不規則任務加以克服及較佳地也控制無向量/向量處理器。在這較佳結構中，無向量/向量處理器120作為一可程式化共同處理器(以下也稱為CVP，共同向量處理器)。無向量/向量處理器120及微控制器130之間的介面處理通信(如經共有記憶體)及同步化(如經共有記憶體及狀態信號)。較佳地，介面為映射記憶體。

介面塊140容許處理器與系統的其餘部份交互作用。在較佳具體實施例中，無向量/向量處理器作為軟體數據機(收發器)使用於2G/3G行動網路。用於該軟體數據機功能，藉由無向量/向量處理器(控制為不規則及資料固定系列通信)或微控制器130(中斷率太高)無法容易地執行控制及連接無

線電。用於該種應用較佳使用專用硬體如具有一主任務的前端處理器以傳遞控制及資料字元至微控制器130控制下的向量記憶體，例如DMA。然後在向量記憶體內的資料由無向量/向量處理器處理。用於一軟體數據機，由無向量/向量處理器執行的接收器功能包括一濾波器、一長型接收器、一頻道估計器、一搜尋器一、解交錯器、一上鏈、一渦輪解碼器、一Viterbi解碼器及一解多工器。由無向量/向量處理器執行的發射器功能包括一多工器、頻道編碼器、一交錯器、一發射器及濾波器。其中，為大家熟知的功能不再詳述。

無向量/向量處理器120為從屬於匯流排110，其中微控制器130及介面塊140(包括一DMA單元)作為主機。所有與CVP通信，如程式、資料、或控制，較佳為映射記憶體。記憶體為一晶片外DRAM及該DRAM也由無向量/向量處理器用作為(解)交錯記憶體。

圖2顯示根據本發明的處理器的主結構。處理器包括一管線向量處理部份210。在圖2的較佳具體實施例中，向量部份包括七個功能單元詳述如下。熟悉本技術者會選擇最佳的特定任務的功能單元。為了支援向量部份的運算，無向量/向量處理器包括一無向量處理部份220配置與向量理部份平行運算。較佳地，無向量處理部份也配置成管線。為了支援向量部份的運算，向量部份至少一功能單元也提供無向量部份的相對部份的功能。例如，偏移功能單元的向量部份功能性偏移一向量，同時由偏移功能單元的無向量

部份供應(或輸送)一無向量組件。如此，偏移功能單元包括向量及無向量兩部份。所以，至少一些功能單元不只具有一向量部份也具有一無向量部份，同時向量部份及無向量部份能由交換無向量資料共同運算。功能單元的向量部份提供原處理功率，而相對無向量部份(即，相同功能單元的無向量部份)藉由供應及/或消耗無向量資料支援向量部份運算。向量部份的向量資料經由向量管線供應。

功能單元(FU)係平行運算。各FU能接收及傳送向量資料。許多FU能接收及傳送無向量資料。FU之一為特別FU，稱為指令分配單元(IDU 250)。該單元包括程式記憶體252及回應指令排序及分配指令區段至其本身及其他FU。原則上，各FU具有三部份：控制230、無向量220及向量210。如下列詳述，一些FU的控制部份及無向量部份放空。

根據本發明的無向量/向量處理器以兩種主要方法施加指令階層平行性：

1. 向量處理，即是一單指令作(無向量)資料的向量運算。這種方法也稱為單指令流，多資料流或SIMD，
2. 多功能單元平行處理，各單元作向量運算。這可看作VLIW指令階層平行性的一種(限制)形式，

注意這兩種指令階層平行性的形式為獨立，而其效果為累計。

功能單元(FU)概要

在較佳具體實施例中，CVP包括下列七個特別功能單元。

- 指令分配單元(IDU 250)。IDU包含程式記憶體252，連續

讀取VLIW指令及分配各指令的7個區段至7個功能單元。較佳地，包含一循環單元支援至零額外循環的三巢化階層。在較佳具體實施例中，並不支援分支、跳越、或中斷。初始程式計數器從限制描述符載入，詳述如下。

- 向量記憶體單元(VMU 260)。VMU包含向量記憶體(圖2中未顯示)。在各指令期間從向量記憶體傳送一線或一向量或接收一線至向量記憶體。另外相同指令指定一無向量傳送運算及/或接收運算。VMU是唯一功能單元連接至外界，即是至外部匯流排110。
- 碼產生單元(CGU 262)。CGU特別用於有限欄演算。例如，CGU可用於產生CDMA碼晶片的向量及相關功能，如頻道編碼及CRC。
- ALU-MAC單元(AMU 264)。AMU為特別用於規則整數及固定點演算。支援向量間運算，即是以元件方式執行多向量演算。在一較佳具體實施例中，AMU也提供一些向量內運算，即是在一單向量內執行元件演算。
- 混洗單元(SFU 266)。SFU根據一特定的混洗圖案可重新配置向量的元件。
- 偏左單元(SLU 268)。SLU可向左偏移向量元件一單位，如一字元、一雙字元或一四字元。提供產生的無向量至其無向量部份。根據發出的SLU向量運算形式，所消耗的無向量為零或取自其無向量部份。
- 偏右單元(SRU 270)。SRU與SLU相似，但偏向右邊。另外，具有合併AMU上向量內運算的連續結果的能力。

下表顯示所有FU具有一功能向量部份，但有些並不具有控制部份或無向量部份。

功能單元	控制	無向量	向量
指令分配單元	排序，循環		指令分配
向量記憶體單元	位址計算	無向量i/o	向量i/o
碼產生單元			產生碼向量
ALU-MAC單元	標示	廣播	向量間：ALU、MAC、mul、...
		區段	向量內：加，最大
混洗單元			向量混洗
偏左單元		無向量i/o	向量偏移
偏右單元		無向量i/o	向量偏移

可以了解，特定應用可選擇其他FU。較佳地，在一基本處理器中使用AMU及VMU結合IDU。如果功率消耗重要，捨棄SFU因為該單元比偏移單元消耗較多的功率，而偏移單元可協助執行混洗運算。SFU對Viterbi解碼特別有用。CGU及CGU的特別形式的選擇係根據使用傳統AMU指令不易產生的碼的計算要求，例如，Galois欄計算及產生加密碼。在特定應用中，附加一或更多AMU較有利以便獲得較高的FU平均負載。同樣，也可以加其他專用FU，例如，用於執行一定的位元階層計算。

較佳地，至少一FU為可配置並認為該FU的運算可由儲存在該FU內的參數影響。較佳地，該參數(配置資料)可從向量記憶體讀取。該種配置協助簡化處理器的程式化及減少碼尺寸。

FU間通信

所有功能單元平行運算。在接收指令區段後，各單元輸

入、處理及輸出資料、向量資料及無向量資料(如有)。FU之間通信在無向量部份之間式或在向量部份(FU間通信)之間非常嚴格。即是，所有FU的向量部份除IDU外由一管線連接。在一較佳具體實施例中，該管線根據指令配置。對此，較佳地，由一互連網路連接的FU原則上容許各向量部份在各循環中接收來自任何其他向量部份的一向量。這種特性另外能產生任意的FU管線(IDU除外)。具有向量路徑的六個功能單元在各時脈循環期間能平行輸出一向量及傳送該向量至其他單元。同樣也能接收來自其他單元的向量。網路係近乎完全連接。只有無意義的鏈結被省略。向量路徑的連接性表列如下(!表示一連接)。注意AMU能同時接收兩向量。

FU	源	vmu	cgu	amu	sfu	slu	sru
目標	#輸入						
vmu	1	!	!	!	!	!	!
cgu	1	!		!	!	!	!
amu	2	!	!	!	!	!	!
sfu	1	!	!	!	!	!	!
slu	1	!		!	!	!	!
sru	1	!		!	!	!	!

如圖2所示，網路較佳係由各FU連接作為信號源(以碟片表示)至一網路路徑所形成。並連接至所有其他路徑作為信號槽(以三角形表示)。FU的VLIW指令的部份表示那個路徑必須消耗一向量。在此方法中，該管線根據指令配置。各路徑可傳送一完全向量，如，使用256平行線。同樣，至少一些FU的無向量部份由一分離管線連接。較佳地，該管線

也根據指令配置。FU的無向量部份中的連接網路部份確定至少一FU的無向量部份不能傳送或接收無向量。如此，可以標示較少的管線次序。無向量及向量管線可獨立配置。例如，由功能單元表示相關VLIW部份讀取無向量管線及向量管線。可以了解取代FU具有輸出資料的固定相關路徑，FU可以具有接收資料的固定相關路徑，其中VLIW指令指出資料必須輸出路徑。

FU	源	vmu	cgu	amu	sfu	slu	sru
目標	#輸入						
vmu	1	!		!		!	!
cgu	0						
amu	1	!				!	!
sfu	0						
slu	1	!					!
sru	I	!				!	

如一配置能力例子，一第一VLIW指令造成AMU消耗一由CGU產生的一向量及由VMU產生一向量。下一指令造成SFU消耗VMU的一向量及AMU消耗SFU的一向量。第三指令造成AMU消耗VMU的一向量及SFU消耗AMU的一向量。

不同功能單元的控制部份之中沒有標示連接性。控制部份接收來自IDU的VLIW指令的一區段，更新其狀態及控制各自的無向量及向量部份。

FU內通信

在一FU之內有八個部份間的交互作用(FU內通信)。交互作用為該FU運算的一整體部份。例如SLU及SRU，其中產生及/或消耗的無向量供應至/取自該FU的相對無向量部

269

份。較詳細見FU詳細說明中的部份說明。

指令一般以單循環執行。向量記憶體擁擠產生例外及顯示循環暫停。

資料寬度及資料形式

在一較佳具體實施例中，無向量/向量處理器支援複數個資料寬度及資料形式如圖3所示。記憶體定址的基本單位為一字元，也稱為一單字元。較佳地，資料寬度可以為一單字元(W)、雙字元(DW)或四字元(QW)。一字元的尺寸為W=8位元。一雙字元(2W=16位元)為一對字元，及其位址必須為偶數值。一四字元(4W=32位元)為一對雙字元，且一雙字元位址為四的倍數。單字元對或雙字元可作複數使用。一向量包括P_Q四字元，相當於$2P_Q$雙字元及$4P_Q$單字元。較佳地，P_Q=8，設總向量寬度為256位元及較佳字元尺寸為8位元。

在一較佳具體實施例中，CVP支援下列資料形式：整數及複合整數。

1. 整數(int)的三種尺寸為：字元、雙字元及四字元，即是，整數範圍$[-2^{N-1} \ldots 2^{N-1}-1]$，其中N等於W、2W、或4W。
2. 複合整數形式，即是，一對整數(真實，假設)，圖3所示為int.real。複合整數的兩種尺寸為：雙字元及四字元。

無向量為任何資料形式整數或複合整數的一值。所以，無向量的三種尺寸為：(單)字元、雙字元及四字元。一向量具有一P_Q四字元的固定尺寸。向量可構成下列三種格式之

一：

1. P_Q元件尺寸為四字元，

2. $P_D = 2P_Q$元件尺寸為雙字元，

3. $P_S = 2P_D = 4P_Q$元件尺寸為(單)字元。

向量元件標示範圍為[0 ... $4P_Q$-1]。所以，雙字元具有偶數指數及四字元的指數為四的倍數。圖3顯示資料尺寸及資料形式之間的關係的概要。其結構為完全伸縮P_Q及定義適用任何向量尺寸$P_Q \geq 1$。不過，用於大部份的情況，較佳地，選擇P_Q為2的次方。

在該較佳具體實施例中，P_Q為8，適合的資料路徑寬度及記憶體寬度為32字元。

可以配置或程式化所涉及的數學運算以處理資料形式的變化。例如，4基本倍數(低精密度)可以結合成雙精密度倍數或成為一複合倍數。這種技術在DSP及電路設計中為大家所知因而不作詳細說明。

程式執行

本限制為CVP程式執行及同步化的單元。一有限不可中斷任務的限制由一有限CPV指令系列說明。一限制一般為一連續有效DSP核心的時間片段，需要數打指令用於說明及數百循環用於執行。

CPV程式記憶體包含許多限制程式。根據微控制器130指示那種限制執行及在那個次序。用於這種用途微控制器130能寫出向量記憶體內所謂限制識別符的一鏈結表。各限制識別符表示相對物件碼，後續限制識別符，及該限制完成

後產生的可能信號說明。限制識別符(SD)為一向量記憶體內的結構包括3欄：

- 程式記憶體內物件碼的開始位址；
- 向量記憶體內限後續器位址限制(必須執行的下一限制)；零(nil)，如果沒有後續器；
- 一發訊識別符。

取代限制識別符放置在向量記憶體內，也可放置在CVP程式記憶體內。

在較佳具體實施例中，以下列方式啟動一限制的執行：

1. CVP位於其閒置狀態。微控制器130由寫入其SD位址於一指定VM位址啟動一限制的執行。本特別VM位置必須包含目前有效限制的位址，及包含一零值如果CVP閒置。
2. 完成一限制後，如CVP程式內的一清楚EOS(結束限制)指令所示，假設一後續器在目前SD標示，CVP繼續後續器限制。如果後續器不存在，CVP回到其閒置狀態。

一限制的狀態(有效/完成)可由微控制器130藉由檢查VM內指定「目前限制」位置詢問。一限制完成後，CVP可選擇性提供發訊至其環境。用於一組信號線(至少一輸出線)，可以標示上拉、下拉、或鎖緊其狀態。例如，連接這些信號以中斷微控制器130及介面塊140的輸入。

指令

CVP為一控制指令或一VLIW指令。控制指令為零額外消耗循環初始化，或限制結束。沒有分支、跳越、或副常式。

一VLIW指令分割成數區段，各指令區段標示由相對功能單元執行的運算。該區段可進一步細分成一小部份用於向量部份及無向量部份(如有)。該區段也包括兩部份資訊供網路部份使用以接收資料(一或更多向量部份的向量及一或更多無向量部份的無向量)。

無向量/向量處理器的狀態

CVP的狀態為其功能單元的結合狀態。在該較佳具體實施例中，包括：

- 向量記憶體(VMU的部份)；
- 程式記憶體(IDU的部份)；
- 向量暫存器(所有功能單元)；
- 無向量暫存器(大部份功能單元)；
- 控制暫存器，包括程式計數器及位址偏移暫存器。

程式控制者可見的暫存器以外，CVP的構造一般包括額外暫存器(向量、無向量及控制)用於配置管線及快取。非CVP指令組結構的部份。

一些暫存器(向量、無向量及控制)為所謂配置暫存器。配置暫存器的內容只能從向量記憶體載入，沒有其他方法改變其值。配置暫存器支援功能單元的配置，及一般定義一功能參數。藉由儲存這些「半常數」功能參數於配置暫存器，適當地減少指令寬度及記憶體交通。

下表為所列為CVP狀態成分的概要。

FU	控制路徑				無向量路徑				向量路徑			
	資料		配置		資料		配置		資料		配置	
vmu	偏移	5	位址計算	8					資料記憶體	2048		
cgu					計數器	3	碼	3	狀態	6	遮罩	2

											多項式	2
amu				1	接收	1	區段尺寸	1	暫存器檔案	16		
sfu									暫存器	1	混洗圖案	4
slu					接收	1			暫存器檔案	2		
sru					接收	1			暫存器檔案	2		
idu	程式計數器	1	循環計算	2					程式記憶體	2048		

所有程式控制者可見的暫存器可從向量記憶體載入。所有暫存器，配置暫存器除外，可以儲存在向量記憶體內。在一限制結束儲存CVP暫存器及後來回復，如果同時沒有其他限制在執行，CVP可繼續特別任務。這些儲存及回復運算為選擇性，為部份及必須清楚地程式化。

指令階層執行

一限制的程式係儲存在IDU內部的程式記憶體。IDU控制程式計數器，讀取目前指令及分配指令的6區段至6個相對功能性單元。一般每個時脈循環可發佈一指令。如果在一循環中可執行多向量記憶體存取，該規則唯一例外由VMU的暫停循環產生。如果在單循環中有多重快取失誤便發生記憶體擁擠及相關循環暫停。

因為沒有資料關係控制，IDU至其他功能單元的交通為單向。這樣大幅簡化CVP指令的管線執行。絕大部份這種管線配置程式控制者看不見。例如，一指令的源暫存器可作為先前指令的目標暫存器。唯一可見管線效果為相關的「昂貴」資源，如向量路徑乘法。一些運算具有多時脈循環的潛伏期。另外，少數的運算也具有多時脈循環的初始化期間(如果一運算具有一初始化期間n循環，則兩個該種運算必須隔離n-1循環的時間)。

較佳功能單元的詳細說明

CVP的各功能單元可分割成控制、無向量及向量部份。根據Moore機器模型模造這些部份，包括五元件：輸入、輸出、狀態、下一狀態功能及輸出功能。

Moore機器的狀態由現有記憶體及/或暫存器決定。用於各功能單元，提供一表定義所有容許轉移包括相對保護。保護為發生真轉移的需要條件。轉移定義Moore機器的下一狀態功能及輸出功能。為了能了解轉移表中的實際資料形式的摘要，使用下列規定：

- P表示處理元件數。根據資料尺寸，P估計為：P_S (WORD資料元件)、P_D (DWORD資料元件)或P_Q (QWORD資料元件)；
- 除非另有說明，使用C型語法；
- 使用方括號以選擇向量內的一元件。例如：v[p]表示向量v的元件p；
- 複合值x的真實及假設部份分別由Re(x)及Im(x)表示；
- 使用括號<及>表示一複合對。例如<re, im>表示複合數re＋j·im；
- 使用運算符號∀(「全部」)以表示已經執行一範圍的向量元件的運算。運算符號並不含任何順序(即，範圍內的所有元件可以平行處理)。例如：$\forall_{P:\ 0\le p<P}\{v[p]=0\}$表示所有向量v的元件設定為0。注意用來表示範圍的假變數(例如p)不具有功能意義；
- 使用運算符號^(「及」)以分隔可以平行執行的運算。換

言之：不像C隔離符號「；」，運算符號^不含系列執行的運算；

- 「if-then-else」運算符號來自C: "*cond ? exp1 : exp2*"如果cond為真，估計為exp1，否則為exp2。

指令分配單元

圖4顯示指令分配單元(IDU 400)的方塊圖。IDU具有下列功能：

- 包含程式記憶體410；
- 分配運算420至其他功能單元；
- 說明限制識別符及控制由微控制器130發佈的核心執行。

對於後者維持程式計數器430及支援零額外消耗循環包括三巢階層。

指令分配單元(IDU)可發佈5種指令形式之一的指令：

1. 正常VLIW指令(NORMAL)；
2. 零額外消耗循環初始化指令(LOOP)；
3. 限制結束指令(EOS)；
4. 副常式呼叫指令(CALL)；
5. 副常式返回指令(RETURN)。

instruction = (NORMAL, commands) | (IDU_cmd, paddr, count)

commands = (VMU_cmd, CGU_cmd, AMU_cmd, SFU_cmd, SLU_cmd, SRU_cmd)

IDU_cmd = LOOP | EOS | CALL | RETURN

paddr = {0, ..., 65535}

count = {1, ..., 65536}

輸入/輸出為：

輸出	說明
cmd_vmu	VMU命令
cmd_cgu	CGU命令
cmd_amu	AMU命令
cmd_sfu	SFU命令
cmd_slu	SLU命令
cmd_sru	SRU命令

IDU向量部份包含CVP程式記憶體410：

名稱	說明
pmem[2048]	程式記憶體：2048指令各含(約) 128位元。

各循環(除非CVP由VMU暫停)，由程式計數器(PC 430)從程式記憶體位置擷取一CVP指令。本指令可以為5種指令之一：

1. 正常VLIW指令：分配指令的命令欄內編碼的命令至其他功能單元；
2. 一IDU循環初始化指令(LOOP)：根據該指令的PADDR及COUNT欄設定循環控制暫存器。循環本體(由LOOP之後指令，直至及包括PADDR欄標示的指令組成)必須包括至少一指令。注意1指令的循環本體係自動承認為「特別情況」，及類似處理R.E.A.L. DSP的一重複指令。巢化循環容許具有相同的結束位址。實際完成程式記憶體範圍外的結束位址的狀態並未定義，保留額外位址位元供將來擴充程式記憶體。循環開始位址係自動設定為LOOP指令之後第一指令的位址。沒有運算的指令分配至所有其他功能單元：

3. IDU限制結束指令(EOS)：如果在目前限制識別符表示，產生一或更多信號以表示限制完成。然後，估計下一限制指標。如果為NILL，則CVP進入閒置模式，否則載入下一限制識別符，及在相對限制的程式計數器(PC)執行的初始化後啟動。
4. IDU副常式呼叫指令(CALL)，支援副常式呼叫的最小形式。副常式呼叫機構支援巢化的三階層，及只儲存返回位址。儲存不被破壞的暫存器內容為程式控制者的責任，程式控制者可選擇呼叫規則所使用(呼叫者儲存或被呼叫者儲存)。PADDR欄包含被呼叫的副常式的第一指令的位址及因而可以直接載入程式計數器。儲存在返回位址堆疊的返回位址為CALL指令之後指令的位址。
5. IDU副常式返回指令(RETURN)，將程式計數器送回至相對CALL指令之後的指令(如上)。

IDU控制部份包含CVP程式計數器(PC)。包含暫存器以啟動零消耗循環及副常式呼叫，兩者支援巢化的三階層：

名稱	#位元	說明
pc	11	CVP程式計數器
count0 count1 count2	16	循環計數暫存器(巢化3階層)
end_addr0 end_addr1 end_addr2	11	循環結束位址(巢化3階層)
ret_addr0 ret_addr1 ret_addr2	11	副常式返回位址(巢化3階層)

向量記憶體單元

圖5A顯示向量記憶體單元(VMU 500)的方塊圖。VMU包含及控制向量記憶體510，提供一巨大資料帶寬至其他功能單元。較理想，實際向量記憶體510係根據一單埠SRAM。因為埋入SRAM為P_S*W寬並非一般現有，實際向量記憶體由一或更多的寬隨機存取記憶體(RAM)的堆平行配置而形成。在該較佳具體實施例中，一向量不需要在記憶體內的向量邊界上對齊。如此，由P_S字元組成的向量具有一任意的記憶體位址。一記憶體線具有相同尺寸，但其開始位址由多重P_S定義(用於線存取，忽略該位址的最低有效位元$^2\log P_S$)。容許向量任意對齊(一般對齊最小字元的界線)，記憶體可以獲得較好利用，具有較少的空位置。以下作較詳細說明，採取量測以容許無向量/向量處理器讀取/寫入各別向量，其中向量儲存在兩連續的實體記憶體線內。較理想，無向量資料儲存在用來儲存向量資料的相同記憶體。在那種系統中，無向量可與相對的向量混合。對於記憶體的成本效益及最佳存取時間，較理想，記憶體只容許讀取及寫入全部向量線。如此，邏輯上實體記憶體由線組成，各線的尺寸為一向量。為了支援讀取及寫入無向量使用額外硬體(線內無向量部份的線快取530及支援540)以存取無向量型式向量寬實體記憶體。假設Nr無向量讀取埠及Nw無向量寫入埠為現成，快取530具有至少一組Nr＋Nw向量寬暫存器。對每個無向量讀取埠，快取中的相對暫存器連接(向量寬)實體記憶體510的讀取埠。支援硬體540包括一解多

工器以選擇暫存器的相關無向量資料。解多工器由暫存器的一些無向量控制如該位址的最低有效位元所標示(如，使用256位元向量具有32個8位元字元，無向量由五個最低有效位元表示)。解多工器為大家熟知不再詳細說明。對於每個無向量寫入埠，快取530中的相對暫存器連接Nw輸入的向量寬多工器以選擇送回實際記憶體的寫入快取線。如果一VMU指令要求寫回多條快取線，依順序完成，暫停所有其他功能單元直到完成所有寫入。存取不同寫入埠，但指令相同，則不容許存取實體記憶體內的同一線。假設存取連續無向量的空間區域(如，屬於一處理循環的連續無向量實質上連續儲存在實體記憶體510)，實體記憶體510載入及儲存暫存器的存取頻率大部份可低於無向量對暫存器的存取頻率。較理想，程式控制者看不見向量記憶體周圍的快取。不論使用快取模仿具有單埠SRAM的多埠向量記憶體，程式控制者可假設一相干向量記憶體。因為各暫存器能包含實體記憶體中可能相同資料的複本，必須自動維持相干性取代相干性必須由程式控制者保護。對此，執行一位址衝突檢查，即是，寫入一線位址至一暫存器，同時該線也儲存在另一暫存器內。這樣的檢查，對各暫存器儲存該暫存器內已儲存的線的線位址(位址最明顯部份)已屬充分。如果偵測到一可能的衝突，便可採取改正措施。例如，一讀取暫存器在一具有相同線的暫存器產生一寫入運算後立刻標示為無效。除非再從記憶體讀取暫存器否則不能進一步使用(在寫入暫存器首先寫回至記憶體後)。或者，在產生一

寫入至該寫入暫存器後，一寫入暫存器的內容可複製至所有具有相同線的讀取暫存器。第三可能性為共享讀取及寫入埠之間的暫存器。後面的方法需要額外向量寬多工器，增加成本，但提供一性能優點。也可以採取相同的相干性檢查及量測用於向量讀取，而(部份)向量儲存在具有寫入埠的暫存器。較理想，一線讀取或寫入實體記憶體於一單時脈循環執行利用一單存取實體記憶體510。

因為實體記憶體只能在線邊界上存取，向量傳送運算需要一對齊單元。對齊單元由兩線快取組成，包含由要求向量隔開的兩線。如果存取連續向量，只有一新線必須從實體記憶體擷取，因為另一線仍在一線快取之內。形成要求向量的兩快取線的部份結合一由多工器組成的網路，然後儲存在一向量寬管線暫存器內。從這管線暫存器，在VMU廣播匯流排上傳輸該值。

向量記憶體單元能支援高達四個同時發生的單VMU指令的「子運算」：

1. 傳送一向量、或傳送一線、或接收一線自/至VM位置；
2. 傳送一無向量自一VM位置；
3. 接收一無向量至一VM位置；
4. 修改一位址計算單元的狀態/輸出。

VMU_cmd = (vopc, aid_v, ainc_v, sopc, aid_s, ainc_s, size, srcv, aid_r, ainc_r, aopc, aid_a, imm_aadr)

vopc = NOP | SENDL | SENDV | RCVL_CGU | RCVL_AMU | RCVL_SFU | RCVL_SLU |

RCVL_SRU

Aid_v = {0, ..., 7}

Ainc_v = NOP | INC

sopc = NOP | SEND

aid_s = {0, ..., 7}

ainc_s = NOP | INC

size = WORD | DWORD | QWORD

srcv = NONE | VMU | AMU | SLU | SRU

aid_r = {0, ..., 7}

ainc_r = NOP | INC

aopc = NOP | IMM | LDBASE | LDOFFS | LDINCR | LDBOUND

aid_a = {0, ..., 7}

imm_addr = {0.0, ..., 524288.31} | {-262144.0, ..., 262143.31}

VMU指令根據子運算數及位址系列的連續性使用不同的時脈循環數。

VMU輸入/輸出為：

輸入	說明
Cmd	VMU指令
rcv_amu	AMU向量接收匯流排
rcv_cgu	CGU向量接收匯流排
rcv_sfu	SFU向量接收匯流排
rcv_slu	SLU向量接收匯流排
rcv_sru	SRU向量接收匯流排
s_rcv_amu	AMU無向量接收匯流排
s_rcv_slu	SLU無向量接收匯流排
s-rcv_sru	SRU無向量接收匯流排

輸出	說明
Snd	VMU向量結果
s_snd	VMU無向量結果

另外有兩無向量埠(一傳送，一接收)連接外部匯流排。使用CVP指令記憶體存取同步化記憶體為微控制器130的任務。

VMU向量部份包含實體向量記憶體510：

名稱	說明
mem[4096][32]	向量記憶體：4096線各為32字元

注意向量子運算不能存取無向量記憶體。所以，向量子運算忽略最有效位址位元。VMU的向量部份支援七個在指令的VOPC欄內編碼的子運算：向量傳送(SENDV)，線傳送(SENDL)，及五個線接收子運算(RCVL_CGU、RCVL_AMU、RCVL_SFU、RCVL_SLU及RCVL_SRU)。接收源的功能單元在相對線接收子運算中清楚編碼。各子運算的讀取位址或寫入位址由相對位址計算單元標示。AINC_V欄為所有向量子運算之間共有。傳遞至AID_V欄中編碼ACU。AINC_V欄顯示是否受影響的位址計算單元必須執行一增加後運算。

保護	轉移
vopc＝NOP	無
vopc＝SENDL	snd＝mem.line[acu[aid_v].out]
vopc＝SENDV	snd＝mem.vector[acu[aid_v].out]
vopc＝RCVL_CGU	mem.line[acu[aid_v].out]＝rcv_cgu
vopc＝RCVL_AMU	mem.line[acu[aid_v].out]＝rcv_amu
vopc＝RCVL_SFU	mem.line[acu[aid_v].out]＝rcv_sfu
vopc＝RCVL_SLU	mem.line[acu[aid_v].out]＝rcv_slu
vopc＝RCVL_SRU	mem.line[acu[aid_v].out]＝rcv_sru

注意運算設定為傳送(或接收)動作，不是載入(或儲存)動作包括一目的地(或源)。後者由運算在其他功能單元標示。一線傳送功能性等於一具有相同位址的向量傳送。線傳送子運算一般用來配置功能單元，或回復各暫存器中一任務的狀態。藉由採用一線傳送的特別模式，連續向量傳送(「向量流」)的存取時間經有效使用快取可以獲得最佳結果。

VMU的無向量子運算係在指令的SOPC欄內編碼的。只支援一個子運算：無向量傳送(SEND)。讀取位址由由AID_S欄標示的位址計算單元標示。本指令的AINC_S欄顯示是否本位址計算單元必須執行一增加後運算。無向量子運算的運算元尺寸(WORD，DWORD或QWORD)係由指令的SIZE欄決定。

保護	轉移
sopc＝NOP	無
sopc＝SEND && size＝WORD	S_snd＝mem.word[acu[aid_s].out]
sopc＝SEND && size＝DWORD	S_snd＝mem.dword[acu[aid_s].out]
sopc＝SEND && size＝QWORD	S_snd＝mem.qword[acu[aid_s].out]

VMU的無向量接收子運算係在指令的SRCV欄內編碼。如果其值為無，便不執行無向量接收。否則，指令的SRCV欄決定使用那個功能單元作為無向量接收源。寫入位址由AID_R欄標示的位址計算單元標示。本指令的AINC_R欄顯示是否本位址計算單元必須執行一增加後運算。無向量接收子運算的運算元尺寸(WORD，DWORD或QWORD)係由源無向量尺寸決定。

保護	轉移
scrv＝NONE	無
srcv＝VMU	mem.scalar[acu[aid_r].out]＝s_rcv_vmu
srcv＝AMU	mem.scalar[acu[aid_r].out]＝s_rcv_amu
srcv＝SLU	mem.scalar[acu[aid_r].out]＝s_rcv_slu
srcv＝SRU	mem.scalar[acu[aid_r].out]＝s_rcv_sru

傳送及接收子運算可合併成一無向量移動運算，從一VM位置至另外位置。各存取的位址係由相對位址計算單元標示。

VMU控制部份550主要為一組的位址計算單元(ACU)或位址產生單元(AGU)以支援定址模式如傳統DSPS。該種單元執行每指令一或更多位址計算不使用處理器的主資料路徑。例如，一無向量的位址可以在各無向量讀取存取後增加。如此容許位址計算與資料計算演算平行發生，改善處理器的性能。根據支援定址模式組，如一ACU需要存取一些暫存器。例如，相對定址，即是，相對一所謂的基本位址定址，需要一基本暫存器base。

- 對基本位址的偏移儲存在一偏移暫存器offs
- 預/後增加偏移值儲存在一增加暫存器incr
- 對一位址模定址儲存在一界限暫存器bound

使用這組定址模式，以下所述便可成立。假設一偏移暫存器offs。在各記憶體在base＋offs存取(讀取或寫入)後，根據offs := (offs＋incr)模bound更新暫存器offs。如此，offs改變頻繁(每次存取後)，而儲存在base，incr，及bound的值則少有變化。一般後面三個暫存器在一程式循環前完成初始化。在其餘暫存器中，假設暫存器為ACU的一部份。暫

存器的初始化稱為「ACU配置」。較理想，無向量/向量處理器包括複數個ACU。在較佳具體實施例中，無向量/向量處理器包括八個ACU。各ACU的配置需要幾個時脈循環。如此，ACU配置所需的時間可成為瓶頸因為受到更多的Amdahl定律的限制。為克服配置延遲，在一較佳具體實施例中，至少兩個屬於一ACU的暫存器可以在一單運算中配置。這可由映射所有ACU暫存器於一單向量及使用專用負載及儲存指令從向量記憶體至ACU記憶體而達成。較理想，ACU的整組相對暫存器可在一時脈循環的單運算中配置。如果記憶體寬度容許一個以上ACU的暫存器可以在一運算中配置較為有利，詳細說明如下。

假設向量記憶體包括2^L線，一無向量或向量位置需要$L+^2\log 4P_Q$位元。例如，$P_Q=8$及$L=12$，則為17位元。為了儘量避免位址計算過長指令及避免分離指令，VMU的控制部份維持一些位址計算單元如上述。各位置計算單元(ACU)由一些位址暫存器及相關增量運算組成。ACU 520另外支援圓形緩衝器。較理想，VMU控制部份包含8個ACU，各ACU可以彈性分配至任何VMU子運算。一限制為各ACU只用於一VMU子運算，ACU子運算除外，即是AID_V≠AID_S≠AID_R。

VMU控制部份支援一子運算，係在該VMU指令的AOPC欄內編碼。支援一子運算以設定一ACU的輸出至一直接位址值(IMM)及四子運算以載入一直接位址至ACU暫存器之一(LDBASE、LDOFFS、LDINCR、及LDBOUND)。該相對

直接位址在IMM_ADDR欄內編碼。AID_A欄標示那個ACU受AOPC子運算影響；VMU指令的AOPC欄及IMM_ADDR欄直接傳遞至特別ACU，及所有其他ACU的AOPC欄設定為無運算(NOP)。

較理想，各ACU包含四位址暫存器：

名稱	#位元	說明
base	24無符號	位址基本暫存器。
offs	24無符號	偏離基本位址。
incr	24有符號	增量值(-bound＜incr＜bound)。
bound	24無符號	上界線。

較佳位址範圍及形式(有符號/無符號)也在表中顯示。在本配置中，四ACU暫存器需要4*24＝96位元。如先前所述，較理想，一向量為256位元寬。在那種情況下，較理想，增加ACU配置速度甚至映射多ACU的暫存器至一向量。在本例中，兩組的ACU暫存器映射至一暫存器。這種情況也在圖5B中顯示。圖中顯示一向量580及四字元邊界。也顯示兩組ACU暫存器590及595，各相對於不同ACU。在本例中，ACU暫存器為24位元及因而不合該向量的一標準資料尺寸之一。為了能容易地經向量記憶體存取各ACU暫存器，載入/儲存ACU暫存器至記憶體的特別指令確保各ACU暫存器在字元邊界上對齊(在本例中，24位元暫存器對齊四字邊界)。熟悉本技術者能根據ACU暫存器尺寸及向量尺寸定義最佳映射。例如，使用16位元ACU暫存器及256位元向量便能映射四組ACU暫存器至一向量。在本指令中需要標示儲存/載入的ACU數。分離或合併指令係用來載入單ACU或一

組ACU暫存器。

位址計算單元(ACU)在一單ACU運算中能支援兩「子運算」:

1. 一後增量子運算;

2. 一直接位址操作子運算。

ACU_cmd = (ainc, aopc, imm_addr)

ainc = NOP | INC

aopc = NOP | IMM | LDBASE | LDOFFS | LDINCR | LDBOUND

imm_addr = {0.0, ..., 524288.31} | {-262144.0, ..., 262143.31}

後增量子運算係在指令的AINC欄內編碼的。只支援一個子運算:後增量(INC)。使用本子運算以避免過詳盡位址計算指令。

保護	轉移
ainc=NOP	無
ainc=INC	offs=(offs+incr) mod bound

直接位址操作子運算係在指令的AOPC欄內編碼的。支援一子運算以輸出一直接位址值(IMM)及四子運算以載入一直接位址至ACU暫存器之一(LDBASE、LDOFFS、LDINCR及LDBOUND)。直接位址在指令的IMM_ADDR欄內編碼。

保護	轉移
aopc=NOP	out=base+offs
aopc=IMM	out=imm_addr
aopc=LDBASE	out=base+offs；base=imm_addr
aopc=LDOFFS	out=base+offs；offs=imm_addr
aopc=LDINCR	out=base+offs；incr=imm_addr
aopc=LDBOUND	out=base+offs；bound=imm_addr

ACU輸入/輸出為：

輸入	說明
cmd	ACU命令(詳細見指令格式)

輸出	說明
out	ACU位址(線位址＋線內無向量)

碼產生單元

圖6顯示CGU(碼產生單元600)的方塊圖。CGU的任務為產生複合符號的碼系列，用長度為P_D或P_S的向量格式化。在一較佳具體實施例中，碼系列係根據(w-)CDMA的需要設定及能構成作為兩系列的產品，加密碼及通道碼。用於該種應用，CGU由一加密碼產生器(SCG 610)及一通道碼產生器(CCG 620)組成。加密碼由一可配置產生器產生。通道碼產生器也可用查找表來取代。該表存放在向量記憶體內。CGU具有的功能有限，如只支援UMTS-FDD下鏈碼產生。碼產生單元(CGU)支援兩子運算：

1. 一加密碼產生器子運算；
2. 一通道碼產生器子運算。

```
CGU_cmd   = (scram_opc, scram_reg, chan_opc, chan_reg,
             chan_config)
scram_opc = OFF | NOP | LOAD_CONFIG | CONFIG |
             LOAD_STATE | SAVE_STATE | LOAD_REGS_X |
             LOAD_REGS_Y | SAVE_REGS_X | SAVE_REGS_Y
             | STEP_1 | STEP_P_D | STEP_1_X | STEP_P_D_X
scram_reg = {cgus0, cgus1}
```

chan_opc = OFF | NOP | CONFIG | LOAD_STATE | SAVE_STATE | STEP_1 | STEP_P_D

chan_reg = {cguc0, cguc1}

chan_config = 0, ..., ^{2}log(SF_{MAX})-1

輸入/輸出為：

輸入	說明
Cmd	CGU指令(詳細見指令格式)
rcv_vmu	VMU向量接收匯流排

輸出	說明
Snd	CGU向量結果

CGU向量部份包含下列暫存器(檔案)：

名稱	#位元	說明
poly_x	18	LFSR X的多項式
mask_x	18	LFSR X的遮罩
cstate_x	18	LFSR X的目前狀態
state_x0，state_x1	18	LFSR X的另外狀態
poly_y	18	LFSR Y的多項式
mask_y	18	LFSR Y的遮罩
cstate_y	18	LFSR Y的目前狀態
state_y0，state_y1	18	LFSR Y的另外狀態
Ccounter	9	目前計數器暫存器
counter0，counter1	9	另外計數器暫存器
ccode_no	9	目前碼數
code_no0，code_no1	9	另外碼數暫存器

CGU指令的SCRAM_OPC欄標示下列SCG向量子運算之一：

- 關閉加密碼產生器(CGUS_OFF)；
- 無運算(CGUS_NOP)；

- 從vmu載入配置向量(CGUS_LOAD_CONFIG)；
- 用配置向量配置LFSR X及LFSR Y (CGUS_CONFIG)；
- 從SCRAM_REG欄標示的內暫存器載入LFSR X及LFSR Y狀態(CGUS_LOAD_STATE)；
- 儲存LFSR X及LFSR Y狀態於SCRAM_REG欄標示的內暫存器(CGUS_SAVE_STATE)；
- 從VMU載入整個LFSR X內暫存器檔案(CGUS_LOAD_REGS_X)；
- 從VMU載入整個LFSR Y內暫存器檔案(CGUS_LOAD_REGS_Y)；
- 儲存整個LFSR X內暫存器檔案於VMU (CGUS_SAVE_REGS_X)；
- 儲存整個LFSR Y內暫存器檔案於VMU (CGUS_SAVE_REGS_Y)；
- LFSR X及LFSR Y前進一單步(CGUS_STEP_1)；
- LFSR X及LFSR Y前進P_D步(CGUS_STEF_P_D)；
- LFSR X前進一單步(CGUS_STEP_1_X)；
- LFSR X前進P_D步(CGUS_STEP_P_D_X)。

CGU指令的CHAN_OPC欄標示下列CCG向量子運算之一：

- 關閉通道碼產生器(CGUC_OFF)；
- 無運算(CGUC_NOP)；
- 用CHAN_CONFIG欄中標示的碼數配置該通道碼產生器(CGUC_CONFIG)；
- 從CHAN_REG欄標示的內暫存器載入OVSF狀態(CGUC_LOAD_STATE)；
- 儲存OVSF狀態於CHAN_REG欄標示的內暫存器(CGUC_SAVE_STATE)；

- OVSF計數器前進一單步(CGUC_STEP_1)；
- OVSF計數器前進P_D步(CGUC_STEP_P_D)。

ALU-MAC單元

圖7顯示ALU-MAC單元(AMU)的方塊圖。AMU為機器的心臟，用來執行實際信號運算，而其他單元則只傳送運算元及造成各量格式化的形式。AMU也包含相當大向量暫存器檔案(較理想，16向量暫存器)。累積器向量需要兩(正向)向量暫存器，加一所謂擴充向量，含有擴充精密度位元。如此，結合擴充暫存器檔案，AMU暫存器檔案，較理想，也支援8累積器向量暫存器。在一較佳具體實施例中，AMU向量部份包含五獨立內部處理單元：

- ALU單元710，處理算術及邏輯形式的向量間運算：加法(ADD)，減法(SUB)，位元方式及(AND)，位元方式或(OR)，位元方式互斥(XOR)，及Hamming距離計算(HAMM)；
- MAC單元720，處理乘法及累積乘法形式的向量間運算：數值(MAGN)，乘法(MUL)，及累積乘法(MAC)；
- (C)I-ADD單元730，處理(相關)內加向量運算：內加(IADD)，內加具有雙精密度結果(IADDD)，及內加之前的向量間相關(CIA)；
- I-MAX/MIN單元740，處理向量內最大及最小運算：內最大(IMAX)，及內最小(IMIN)；及
- 定標單元，用來提供具有整數計算的彈性固定點支撐。

可以了解根據應用有些子單元並不需要。同樣，可以添加其他子單元。例如，如果添加支援外部收發器計算，較

理想，AMU包含兩額外處理單元：

- acs單元，處理Viterbi的加-比-選運算(acs)；
- abl單元，處理渦輪解碼需要的αβΛ運算(abl)。

使用那個處理單元AMU指令內並未清楚編碼，但從vopc欄可導出詳細。這表示初始時間間隔並不妨礙執行，只要發佈後續的向量指令至不同處理單元。

ALU_MAC單元(AMU)在一單AMU運算中能支援高達4「子運算」：

1. 執行ALU(算術及邏輯)，MAC(累積乘法)，(C)I-ADD((相關)內加)，或I-MAX/MIN(內最大或內最小)形式的向量計算；
2. 接收一向量值至AMU向量暫存器檔案；
3. 接收一第二向量值至AMU向量暫存器檔案；
4. 接收一無向量值至AMU廣播暫存器(BCST)；

AMU_cmd =(vopc, type, src1, src2, src34x, rcv1, dst1, rcv2, dst2, srcv)

vopc = NOP | SND | ADD | SUB | ABS | NEG | MAX | MIN | CONJ | MAGN | IADD | DIADD | IMAX | IMIN | CIA | AND | OR | XOR | MUL | MAC | SGNX

type = cvp_int | cvp_dint | cvp_cint | cvp__qint | cvp_cdint

src1 = {amu0, ..., amu15}

src2 = {amu0, ..., amu15} | {bcst}

src34x = src3 ∈ {amu0, ..., amu15} | acc ∈ {acc0, ..., acc7} |

293

$$Iseg \in \{1, 2, 3, ..., {}^{2}\log P_S\}$$

rev1 = NONE | VMU | CGU | AMU | SFU | SLU | SRU

dst1 = {amu0, ..., amu15}

rcv2 = NONE | VMU | CGU | AMU | SFU | SLU | SRU

dst2 = {amu0, ..., amu15}；dst2≠dst1

srcv = NONE | VMU | SLU | SRU

輸入/輸出為：

輸入	說明
Cmd	AMU命令(詳細見指令格式)
rcv_vmu	VMU向量接收匯流排
rcv_cgu	CGU向量接收匯流排
rcv_sfu	SFU向量接收匯流排
rcv_slu	SLU向量接收匯流排
rcv_sru	SRU向量接收匯流排
s_rcv_vmu	VMU無向量接收匯流排(廣播單元)
s_rcv_slu	SLU無向量接收匯流排(廣播單元)
s_rcv_sru	SRU無向量接收匯流排(廣播單元)

輸出	說明
Snd	AMU向量結果
s_snd	AMU無向量結果

AMU向量部份包含下列暫存器檔案：

名稱	#位元	說明
amu0，...，amu15	P_S×8	一般暫存器檔案(RF_{AMU})
acc0, ..., acc7	P_S×20	累積器暫存器檔案。映射累積器暫存器至兩個一般暫存器(從RF_{AMU})，加擴充暫存器(從RF_{EXT})包含擴充精密度位元。

各累積器暫存器需要兩AMU暫存器810，820，加一擴充暫存器830如圖8所示。例如，暫存器ACC3佔有暫存器

AMU6、AMU7及EXT3。注意擴充暫存器檔案只用來儲存擴充累積器值的精密度部份。所以，不能自行存取，及因而程式控制者看不見。

向量子運算係在AMU指令的VOPC欄內編碼。支援4種的子運算：

1. 算術及邏輯形式的向量間運算：傳送一暫存器值(SND)，加法(ADD)，減法(SUB)，絕對值(ABS)，負值(NEG)，最大(MAX)，最小(MIN)，位元方式及(AND)，位元方式或(OR)，位元方式互斥(XOR)，及信號擴充(SGNX)；
2. 乘法及累積乘法形式的向量間運算：數值(MAGN)，乘法(MUL)，及累積乘法(MAC)；
3. (相關)內加向量運算：內加(IADD)，內加具有雙精密度結果(DIADD)，及內加之前的向量間相關(CIA)；
4. 內最大(IMAX)，及內最小(IMIN)運算；

使用的資料形式在AMU指令的TYPE欄中編碼。SRC1及SCR2欄表示那個暫存器必須分別用作向量子運算的源1及源2。注意SRC2資料可來自無向量廣播暫存器BCST，取代來自標準暫存器檔案RF_{AMU}。如此，相同無向量值傳送至各處理元件。同樣，SRC34X欄表示那個暫存器必須作為需要3向量源的子運算的源3 (以SRc3表示)。如果是乘法(累積)子運算，SRC34X欄表示那個累積器暫存器(ACC表示)必須使用(作為源及目的地)。如果是向量內運算(IADD，DIADD，IMAX，IMIN，及CIA)，SRC34X欄包含基數2對數(以LSEG表示)的區段尺寸：$^{2}\log(SEG)$。最小區段尺寸為2 (SRC34X

＝1)。上限根據資料尺寸：尺寸WORD的運算元P_S (LSEG＝5)，尺寸DWORD的運算元P_D (LSEG＝4)，及尺寸QWORD的運算元P_Q (LSEG＝3)。

保護	區段尺寸(seg)
lseg＝0	seg＝segment (見Error! Reference source not found.)
lseg≠0	seg＝2^{lseg}

向量內運算結果為一備用向量其中只計算各區段的位置0。目標暫存器的其他位置設定為0。連結偏右單元，混洗單元可用來記錄備用向量的內容。

保護	轉移
vopc＝NOP	無
vopc＝SND	$\forall_{p:\ 0\le p<P}$ {snd[p]＝src1[p]}
vopc＝ADD	$\forall_{p:\ 0\le p<P}$ {snd[p]＝src1[p]＋src2[p] }
vopc＝SUB	$\forall_{p:\ 0\le p<P}$ {snd[p]＝src1[p]－src2[p]}
vopc＝CONJ	$\forall_{p:\ 0\le p<P}$ {snd[p]＝<Re(src1[p]), -Im(src1[p])>}
vopc＝ABS	$\forall_{p:\ 0\le p<P}$ {snd[p]＝| src1[p] |}
vopc＝NEG	$\forall_{p:\ 0\le p<P}$ {snd[p]＝-src1[p]}
vopc＝MAX	$\forall_{p:\ 0\le p<P}$ {snd[p]＝max(src1[p], src2[p])}
vopc＝MIN	$\forall_{p:\ 0\le p<P}$ {snd[p]＝min(src1[p], src2[p])}
vopc＝MAGN	$\forall_{p:\ 0\le p<P}$ {snd[p]＝Re(src1[p])^2＋Im(src1[p])^2}
vopc＝IADD	$\forall_{s:\ 0\le s<P}$ {snd[s]＝(s mod seg＝0) ? $\sum_{p=s}^{s+seg-1}$ src1[p]:0}
vopc＝DIADD	$\forall_{s:\ 0\le s<P}$ {snd[s]＝(s mod seg＝0) ? $\sum_{p=s}^{s+seg-1}$ src1[p]:0}
vopc＝IMAX	$\forall_{s:\ 0\le s<P}$ {snd[s]＝(s mod seg＝0) ? $MAX_{p=s}^{s+seg\ 1}$src1[p]:0}
vopc＝IMIN	$\forall_{s:\ 0\le s<P}$ {snd[s]＝(s mod seg＝0) ? $MIN_{p=s}^{s+seg\ 1}$$src1[p]$:0}
vopc＝CIA	$\forall_{s:\ 0\le s<P}$ {snd[s]＝(s mod seg＝0) ? $\sum_{p=s}^{a+seg-1}$ src1[p] src2[p]:0}
vopc＝AND	$\forall_{p:\ 0\le p<P}$ {snd[p]＝src1[p] & src2[p]}
vopc＝OR	$\forall_{p:\ 0\le p<P}$ {snd[p]＝src1[p] | src2[p]}
vopc＝XOR	$\forall_{p:\ 0\le p<P}$ {snd[p]＝src1[p] ^ src2[p]}

vopc＝MUL	$\forall_{p:\,0\le p<P}$ {acc[p]＝src1[p] × src2[p]}
vopc＝MAC	$\forall_{p:\,0\le p<P}$ {acc[p] += src1[p] × src2[p]}
vopc＝SGNX	$\forall_{p:\,0\le p<P/2}$ {snd[2p]＝src1[2p]}
*結果具有兩倍源精密度。	

大部份向量運算支援所有資料形式，下列除外：

- CONJ、MAGN：只支援複合資料形式；
- ABS、MAX、MIN、IMAX、IMIN：不支援複合資料形式；
- DLADD及SGNX：不支援四字元大小的資料形式；
- CIA：只支援複合單精密度整數形式；
- AND、OR、XOR：只支援非複合整數；
- NON及SND：加以忽略。

向量子運算以外，AMU向量部份也可以接收來自任何功能單元的兩向量值至區域暫存器檔案(RF_{AMU})。接收子運算在RCV1及RCV2欄編碼，及相對RF_{AMU}目的地暫存器分別在DST1及DST2欄編碼。

保護	轉移
rcv1＝NONE	無
rcv1＝VMU	dst1＝rcv_vmu
rcv1＝CGU	dst1＝rcv_cgu
rcv1＝AMU	dst1＝snd
rcv1＝SFU	dst1＝rcv_sfu
rcv1＝SLU	dst1＝rcv_slu
rcv1＝SRU	dst1＝rcv_sru

保護	轉移
rcv2＝NONE	無
rcv2＝VMU	dst2＝rcv_vmu
rcv2＝CGU	dst2＝rcv_cgu

rcv2＝AMU	dst2＝snd
rcv2＝SFU	dst2＝rcv_sfu
rcv2＝SLU	dst2＝rcv_slu
rcv2＝SRU	dst2＝rcv_sru

AMU無向量部份包含下列暫存器：

名稱	#位元	說明
bcst	32	無向量廣播暫存器。

AMU無向量部份也接收一來自VMU，SLU或SRU的無向量值至廣播暫存器(BCST)。相對接收子運算在SRCV欄編碼。

保護	轉移
srcv＝NONE	無
srcv＝VMU	bcst＝s_rcv_vmu
srcv＝SLU	bcst＝s_rcv_slu
srcv＝SRU	bcst＝s_rcv_sru

整個向量複製的BCST暫存器的內容可用來作為另外SRC2向量運算(取代正常AMU暫存器)：

保護	轉移
src2＝BCST	$\forall_{P:\ 0\le p<P}=\{src2[p]=bcst\}$

AMU控制部份包含一暫存器。

名稱	#位元	說明
segment	$\lceil {}^2\log({}^2\log P_S)\rceil$	向量內運算的區段尺寸。

混洗單元

圖9顯示混洗單元900的方塊圖。向量混洗功能的基本理念為標示各(向量)目標元件910的一(向量)源元件920。源元件係在專用配置暫存器930中標示。混洗可用單字元粒度標示(注意這也包括所有可能的雙字元及四字元混洗圖案)。一完全混洗可利用CMOS十字桿940完成。其成本約與目標數

及源數的積成比例。在許多情況下實現混洗帶寬不需要為$4P_Q$字元。所以，在本較佳具體實施例中只支援半混洗。

混洗單元(SFU)支援兩同時發生的子運算：

1. 一配置或半混洗向量運算；

2. 一向量接收。

SFU_cmd = (vopc, cfg, rcv)

vopc = NOP | CONF | ODD | EVEN | DUPL0 | DUPL1

cfg = {sfuc0, ..., sfuc1}

rcv = NONE | VMU | CGU | AMU | SFU | SLU | SRU

輸入/輸出為：

輸入	說明
cmd	SFU命令(詳細見指令格式)。
rcv_vmu	VMU向量接收匯流排
rcv_amu	AMU向量接收匯流排
rcv_cgu	CGU向量接收匯流排
rcv_slu	SLU向量接收匯流排
rcv_sru	SRU向量接收匯流排

輸出	說明
snd	SFU向量結果

SFU向量部份包含下列暫存器檔案：

名稱	#位元	說明
sfu	P_S×8	普通向量暫存器
sfuc0, ..., sfuc1	P_S×8	向量配置暫存器(RF_{CFG})

注意：另外也有一絕對混洗目標暫存器，但不直接由程式控制者存取。各種形式的混洗結果自動寫入。

SFU支援兩種向量子運算，在SFU指令的VOPC欄標示：

1. 接收一混洗圖案RF_{CFG}(CONF)；

2. 根據RF_{CFG}的圖案在暫存器SFU混洗向量及結果儲存於(絕對)混洗目標暫存器的影響部份。可能混洗奇數目標元件(ODD)，偶數目標元件(EVEN)，或複製源元件至奇數及偶數目標元件(DUPL0及DUPL1)。

因為中間結果儲存在(絕對)混洗目標暫存器，完全混洗可由後續執行ODD及EVEN子運算而完成。如果是配置子運算(VOPC＝CONF)，指令的CFG欄標示那個配置暫存器必須載入。否則標示那個配置暫存器用作混洗圖案。

保護	轉移
vopc＝NOP	無
vopc＝CONF	$\forall_{P:\ 0\le p<P}$ {cfg[p]＝rcv_rmu[p]}
vopc＝ODD	$\forall_{q:\ 0\le q<P/2}$ {p＝2q；if (cfg[p]≠-1) snd[p]＝sfu[cfg[p]]}
vopc＝EVEN	$\forall_{q:\ 0\le q<P/2}$ {p＝2q+1；if (cfg[p]≠-1) snd[p]＝sfu[cfg[p]]}
vopc＝DUPL0	$\forall_{q:\ 0\le q<P/2}$ {p＝2q；if (cfg[p]≠-1) snd[p]＝snd[p+1]＝sfu[cfg[p]]}
vopc＝DUPL1	$\forall_{q:\ 0\le q<P/2}$ {p＝2q+1；if (cfg[p]≠-1) snd[p-1]＝snd[p]＝sfu[cfg[p]]}

向量運算以外，SFU也支援一向量接收運算(至區域SFU暫存器)，並在指令的RCV欄編碼。注意藉由結合向量子運算CONF及一向量接收子運算，可能同時接收一混洗圖案(來自VMU)及一新混洗源(來自任何其他功能單元)。

保護	轉移
rcv＝NONE	無
rcv＝VMU	sfu＝rcv_vmu
rcv＝CGU	sfu＝rcv_cgu
rcv＝AMU	sfu＝rcv_amu
rcv＝SFU	sfu＝snd

rcv＝SLU	sfu＝rcv_slu
rcv＝SRU	sfu＝rcv_sru

注意：結合偏移運算，混洗運算可用來有效記錄來自向量內ALU運算結果的備用向量元件。

偏左單元

圖10顯示偏左單元(SLU 1000)的方塊圖。SLU維持兩向量暫存器能各左偏移一無向量步驟，即三無向量尺寸(WORD，DWORD或QWORD)之一。右邊附加0或一無向量值於區域無向量接收暫存器。向量可接收自或傳送至任何功能單元。偏左單元(SLU)支援三同時發生的子運算：

1. 一向量偏左或傳送運算；
2. 一向量接收運算；
3. 一無向量接收運算。

SLU_cmd ＝ (vopc, src, size, rcv, dst, srcv)

vopc ＝ NOP | SND | SHIFT0 | SHIFTS

src ＝ {slu0, slu1}

size ＝ WORD | DWORD | QWORD

rcv ＝ NONE | VMU | CGU | AMU | SFU | SLU | SRU

dst ＝ {slu0, slu1}

srcv ＝ NONE | VMU | AMU | SLU | SRU

輸入/輸出為：

輸入	說明
cmd	SLU命令(詳細見指令格式)。
rcv_vmu	VMU向量接收匯流排
rcv_amu	AMU向量接收匯流排
rcv_cgu	CGU向量接收匯流排

rcv_sfu	SFU向量接收匯流排
rcv_sru	SRU向量接收匯流排
s_rcv_vmu	VMU無向量接收匯流排
s_rcv_amu	AMU無向量接收匯流排
s_rcv_sru	SRU無向量接收匯流排

輸出	說明
snd	SLU向量結果
s_snd	SLU無向量結果

偏左單元(SLU)的向量部份包含2普通向量暫存器1010：

名稱	#位元	說明
slu0, ..., slu1	$P_S \times 8$	普通向量暫存器檔案file RF_{SLU}。

由SLU支援在指令的VOPC欄編碼的向量運算為：

- no-op，即是，不移動目前向量(NOP)，
- 傳送，廣播選擇源暫存器的內容(SND)，
- 偏左，附加0 (SHIFT0)，
- 偏左，附加來自無向量接收暫存器的無向量(SHIFTS)。

指令的SRC欄決定那個SLU暫存器偏移。如果是SHIFT0運算，指令的SIZE欄決定無向量步驟的尺寸。如果是SHIFTS運算，由無向量接收暫存器SSLU(因而，忽略SIZE欄)的無向量決定無向量步驟的尺寸。

保護	轉移
vopc＝NOP	無
vopc＝SHIFT0	$\forall_{p:\ 0 \le p < P-1} \{snd[p] = src[p+1]\} \wedge snd[P-1] = 0$
vopc＝SHIFTS	$\forall_{p:\ 0 \le p < P-1} \{snd[p] = src[p+1]\} \wedge snd[P-1] = sslu$

向量運算以外，SLU也支援一向量接收運算(至區域SLU暫存器1010之一)，並在指令的RCV欄編碼。DST欄標示接

收向量寫入那個區域暫存器。

保護	轉移
rcv＝NONE	無
rcv＝VMU	dst＝rcv_vmu
rcv＝CGU	dst＝rcv_cgu
rcv＝AMU	dst＝rcv_amu
rcv＝SFU	dst＝rcv_sfu
rcv＝SLU	dst＝snd
rcv＝SRU	dst＝rcv_sru

偏左單元(SLU)的無向量部份包含一無向量接收暫存器1020：

名稱	#位元	說明
sslu	32 (+2)	無向量接收暫存器(包括，尺寸)。

偏移SLU的無向量經S_SND埠傳送至所有功能單元。另外，SLU也支援一無向量接收運算(至無向量接收暫存器SSLU 1020)，並在指令的SRCV欄編碼。SSLU暫存器的無向量值可在下一SHIFTS向量子運算中偏移自右邊的向量。

保護	轉移
srcv＝NONE	無
srcv＝VMU	sslu＝s_rcv_vmu
srcv＝AMU	sslu＝s_rcv_amu
srcv＝SLU	sslu＝s_snd
srcv＝SRU	sslu＝s_rcv_sru

偏右單元

圖11顯示偏右單元(SRU 1100)的方塊圖。SRU與SLU相似，但用來合併AMU向量內運算的備用結果的特別接收模式(IAMU)除外。偏右單元(SRU)支援三同時發生的子運算：

- 一向量偏右或傳送運算；
- 一向量接收運算；
- 一無向量接收運算。

SRU cmd = (vopc, src, size, rcv, dst, srcv)

vopc = NOP | SND | SHIFT0 | SHIFTS

src = {sru0, sru1}

size = WORD | DWORD | QWORD

rcv = NONE | VMU | CGU | AMU | IAMU | SFU | SLU | SRU

dst = {sru0, sru1}

srcv = NONE | VMU | AMU | SLU | SRU

輸入/輸出為：

輸入	說明
cmd	SRU命令(詳細見指令格式)。
rcv_vmu	VMU向量接收匯流排
rcv_amu	AMU向量接收匯流排
rcv_cgu	CGU向量接收匯流排
rcv_sfu	SFU向量接收匯流排
rcv_slu	SLU向量接收匯流排
s_rcv_vmu	VMU無向量接收匯流排
s_rcv_amu	AMU無向量接收匯流排
s_rcv_slu	SLU無向量接收匯流排

輸出	說明
snd	SRU向量結果
s_snd	SRU無向量結果

偏右單元(SRU)的向量部份包含2普通向量暫存器1110：

名稱	#位元	說明
sru0, ..., sru1	P_S×8	普通向量暫存器檔案RF_{SRU}。

由SRU支援在指令的VOPC欄編碼的向量運算為：

- no-op，即是，不移動目前向量(NOP)，
- 傳送，廣播選擇源暫存器的內容(SND)，
- 偏右，附加0 (SHIFT0)，
- 偏右，附加來自無向量接收暫存器的無向量(SHIFTS)。

指令的SRC欄決定那個SRU暫存器偏移。如果是SHIFT0運算，指令的SIZE欄決定無向量步驟的尺寸。如果是SHIFTS運算，由無向量接收暫存器SSRU(因而，忽略SIZE欄)的無向量決定無向量步驟的尺寸。

保護	轉移
vopc＝NOP	無
vopc＝SHIFT0	$\forall_{p:\,1\le p<P}$ {snd[p]＝src[p-1]} ^ snd[0]＝0
vopc＝SHIFTS	$\forall_{p:\,1\le p<P}$ {snd[p]＝src[p-1]} ^ snd[0]＝ssru

向量運算以外，SRU也支援一向量接收運算(至區域SRU暫存器之一)，並在指令的RCV欄編碼。DST欄標示接收向量寫入那個區域暫存器。

保護	轉移
rcv＝NONE	無
rcv＝VMU	dst＝rcv_vmu
rcv＝CGU	dst＝rcv_cgu
rcv＝AMU	dst＝rcv_amu
rcv＝IAMU	dst＝snd | rcv_amu
rcv＝SFU	dst＝rcv_sfu
rcv＝SLU	dst＝rcv_slu
rcv＝SRU	dst＝snd

偏右單元(SRU)的無向量部份包含一無向量接收暫存器1120：

名稱	#位元	說明
ssru	32 (+2)	無向量接收暫存器(包括，尺寸)。

偏移SRU的無向量經S_SND埠傳送至所有功能單元。另外，SRU也支援一無向量接收運算(至無向量接收暫存器SSRU 1120)，並在指令的SRCV欄編碼。SSRU暫存器的無向量值可在下一SHIFTS向量子運算中偏移自左邊的向量。

保護	轉移
srcv＝NONE	無
srcv＝VMU	ssru＝s_rcv_vmu
srcv＝AMU	ssru＝s_rcv_amu
srcv＝SLU	ssru＝s_rcv_slu
srcv＝SRU	ssru＝s_snd

應注意，上述的具體實施例係用以解說本發明而非限制本發明，及熟悉本技術者能設計很多替代的具體實施例，而不致脫離隨附的申請專利範圍的範疇。在申請專利範圍中，任何置於括號之間的參考符號不應視為限制該申請專利範圍。用語「包括」及「包含」並不排除那些在申請專利範圍所列出之外的元件或步驟。

【圖式簡單說明】

圖式中：

圖1顯示一較佳結構，其中使用根據本發明的無向量/向量處理器；

圖2顯示根據本發明的無向量/向量處理器的主結構；

圖3顯示支援資料寬度及資料形式；

圖4顯示指令分配單元的方塊圖；

圖5A顯示向量記憶體單元的方塊圖；

圖5B顯示ACU的映射組登記一向量暫存器；

圖6顯示碼產生單元的方塊圖；

圖7顯示ALU-MAC單元的方塊圖；

圖8顯示一累積暫存器的結構；

圖9顯示一混洗單元的方塊圖；

圖10顯示偏左單元的方塊圖；及

圖11顯示偏右單元的方塊圖。

【圖式代表符號說明】

110 匯流排

120 無向量/向量處理器

130 DSP或微控制器

140 介面

210 向量處理部份

220 無向量處理部份

230 控制

250 指令分配單元

252 程式記憶體

260 向量記憶體單元

262 碼產生單元

264 ALU-MAC單元

266 混洗單元

268 偏左單元

270 偏右單元

400 指令分配單元

410 程式記憶體

420 運算

430 程式計數器

500 向量記憶體單元

510 向量記憶體

520 ACU記憶體

530 線快取

540 支援硬體

550 VUM控制部份

580 向量

590 ACU暫存器

595 ACU暫存器

600 碼產生單元

610 加密碼產生器

620 通道碼產生器

700 ALU-MAC單元

710 ALU單元

720 MAC單元

730 (C)I-ADD單元

740 I-MAX/MIN單元

810 AMU暫存器

820 AMU暫存器

830 擴充暫存器

900 混洗單元

910 向量目標單元

920 向量源單元

930 配置暫存器

940 CMOS十字桿

1000 偏左單元

1010 向量接收暫存器

1020 無向量接收暫存器

1100 偏右單元

1110 向量接收暫存器

1120 無向量接收暫存器

伍、中文發明摘要：

本發明揭示一種無向量/向量處理器包括複數個功能單元252、260、262、264、266、268、270。至少一功能單元包括一向量部份210用於運算至少一向量及一無向量部份220用於運算至少一無向量。功能單元的向量部份及無向量部份的配置由無向量部份共同運算以提供及/或消耗至少一由功能單元的向量部份要求及/或供應的無向量。

陸、英文發明摘要：

A scalar/vector processor includes a plurality of functional units 252, 260, 262, 264, 266, 268, 270. At least one of the functional units includes a vector section 210 for operating on at least one vector and a scalar section 220 for operating on at least one scalar. The vector section and scalar section of the functional unit co-operate by the scalar section being arranged to provide and/or consume at least one scalar required by and/or supplied by the vector section of the functional unit.

拾、申請專利範圍：

1. 一種無向量/向量處理器，包括複數個功能單元，至少一功能單元包括一向量部份用於運算至少一向量及一無向量部份用於運算至少一無向量，該功能單元的向量部份及無向量部份由無向量部份共同運算的配置用來提供及/或消耗至少一由功能單元的向量部份要求及/或供應的無向量。
2. 如申請專利範圍第1項之無向量/向量處理器，其中複數個功能單元包括一各別向量部份，該向量部份位於一第一管線內。
3. 如申請專利範圍第1項之無向量/向量處理器，其中複數個功能單元包括一各別向量部份，該無向量部份位於一第二管線內。
4. 如申請專利範圍第2及3項之無向量/向量處理器，其中第一及第二管線係獨立配置。
5. 如申請專利範圍第2、3或4項之無向量/向量處理器，其中至少一管線根據處理器指令接指令的方式配置。
6. 如申請專利範圍第1項之無向量/向量處理器，其中該無向量/向量處理器由一VLIW指令控制，包括分離的各功能單元的指令區段。
7. 如申請專利範圍第6項之無向量/向量處理器，其中一用於具有向量部份及無向量部份的功能單元的指令區段包括各別指令用於功能單元的向量部份及無向量部份。
8. 如申請專利範圍第5及6項之無向量/向量處理器，其中

VLIW指令包括第一及/或第二管線的配置。

9. 如申請專利範圍第7及8項之無向量/向量處理器，其中第一管線包括連結第一管線的各向量部份的一各別連接管線路徑用於傳送一向量及/或第二管線包括連結第二管線的各無向量部份的一各別連接管線路徑用於傳送一無向量。

10. 如申請專利範圍第8及9項之無向量/向量處理器，其中用於一功能單元的向量部份的指令標示一管線路徑連接一不同功能單元用於傳送一向量及/或用於一功能單元的無向量部份的指令標示一管線路徑連接一不同功能單元用於傳送一無向量。

11. 如申請專利範圍第1項之無向量/向量處理器，其中至少一功能單元為一偏移單元，運算偏移單元的向量部份以偏移一向量超過至少一無向量，運算偏移單元的無向量部份以供應及/或接收來自向量部份至少一偏入或偏出該向量的無向量。

12. 一種處理系統，包括一如申請專利範圍第1項之無向量/向量處理器，其中配置無向量/向量處理器成為無向量處理器的共同處理器及配置該無向量處理器以控制該無向量/向量處理器，配置該無向量/向量處理器的該無向量部份用於執行循環內無向量處理及配置該無向量處理用於執行不規則，循環外無向量處理。

拾壹、圖式：

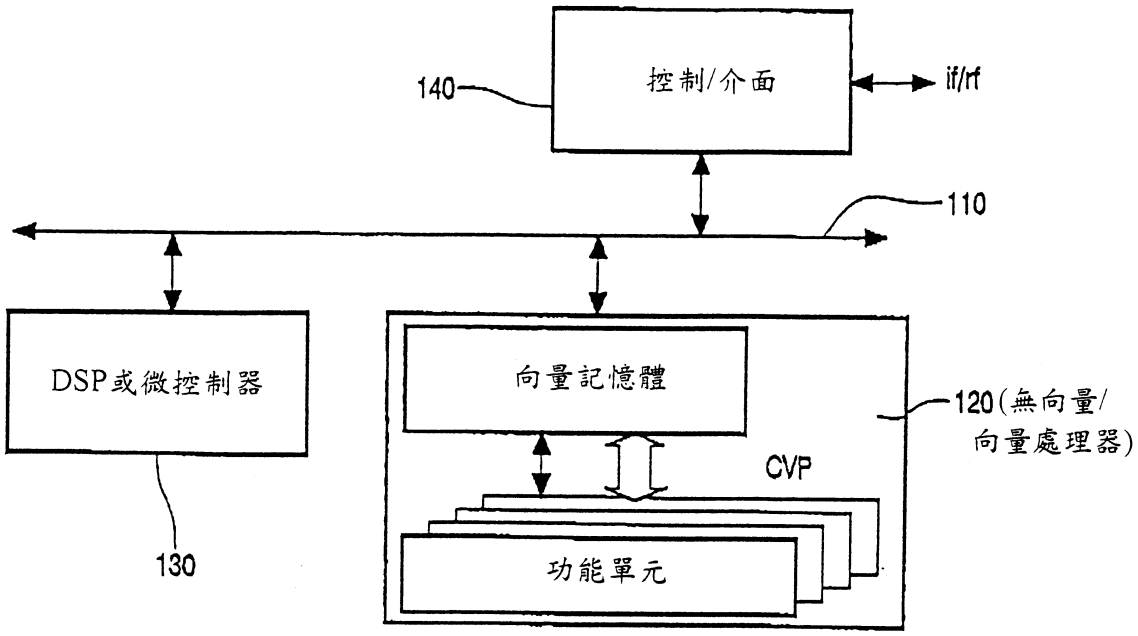
140
控制/介面
if/rf
110
DSP或微控制器
向量記憶體
120(無向量/
向量處理器)
CVP
功能單元
130

圖1

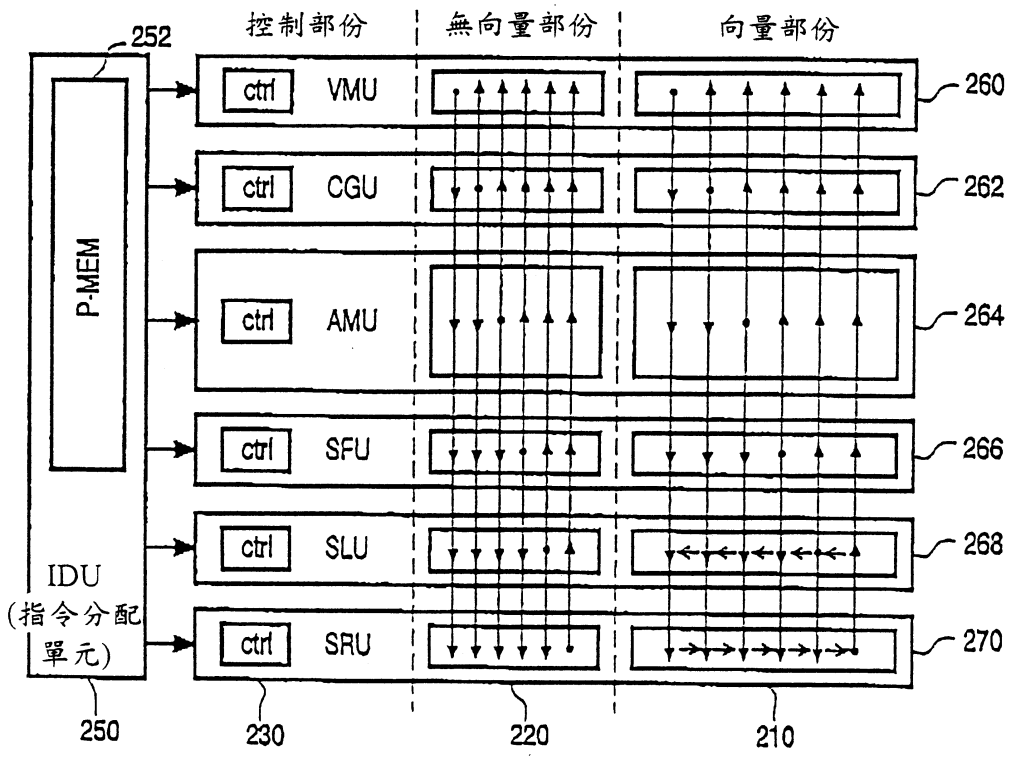
252
控制部份
無向量部份
向量部份
P-MEM
ctrl
VMU
CGU
AMU
SFU
SLU
SRU
260
262
264
266
268
270
IDU
(指令分配
單元)
250
230
220
210

圖2

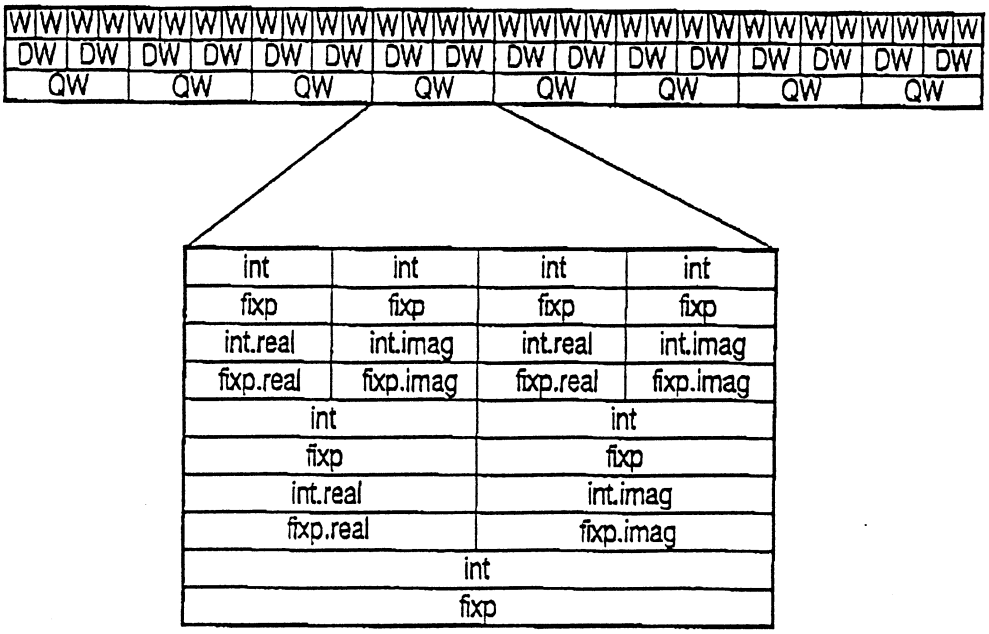
W W W W W W W W W W W W W W W W W W W W W W W W W W W W
DW DW DW DW DW DW DW DW DW DW DW DW DW DW DW DW
QW QW QW QW QW QW QW QW
int int int int
fixp fixp fixp fixp
int.real int.imag int.real int.imag
fixp.real fixp.imag fixp.real fixp.imag
int int
fixp fixp
int.real int.imag
fixp.real fixp.imag
int
fixp

圖3

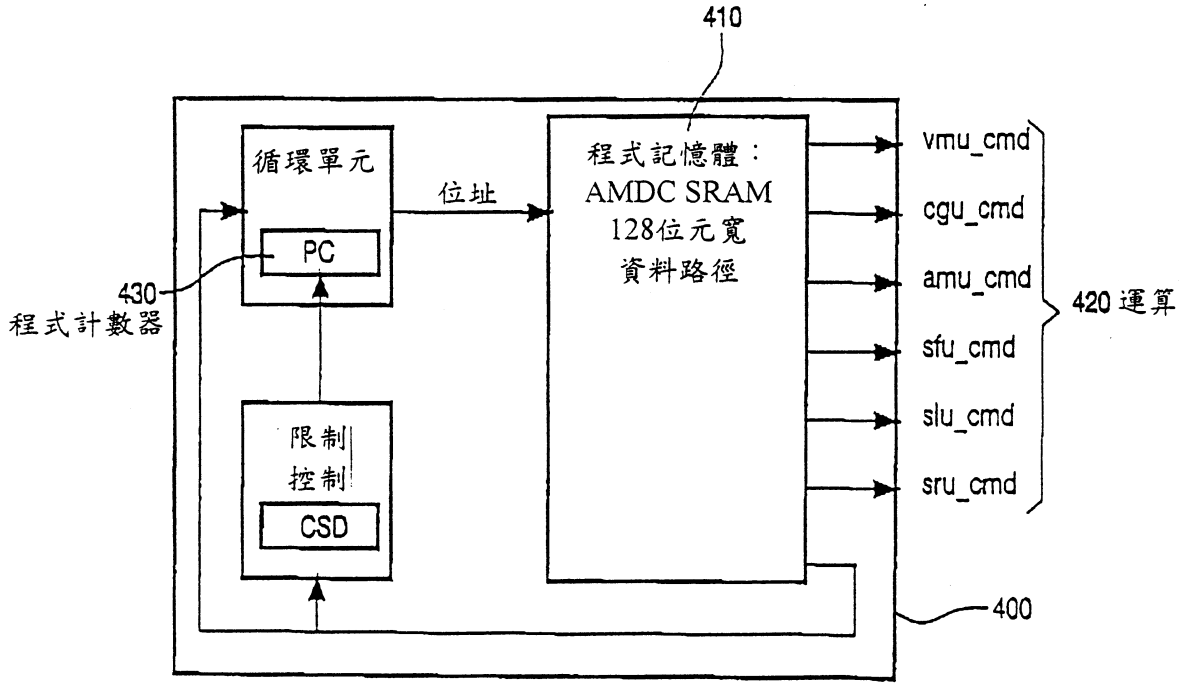
410
循環單元
PC
位址
程式記憶體：
AMDC SRAM
128位元寬
資料路徑
vmu_cmd
cgu_cmd
amu_cmd
sfu_cmd
slu_cmd
sru_cmd
420 運算
430
程式計數器
限制
控制
CSD
400

圖4

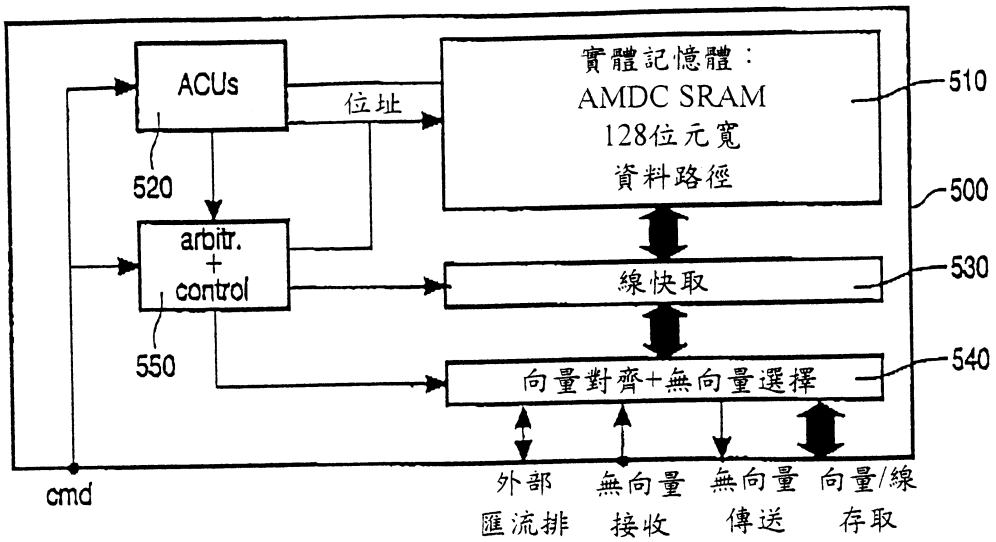
ACUs
位址
實體記憶體：
AMDC SRAM
128位元寬
資料路徑
510
520
500
arbitr.
+
control
線快取
530
550
向量對齊+無向量選擇
540
cmd
外部
匯流排
(32b)
無向量
接收
無向量
傳送
向量/線
存取

圖5A

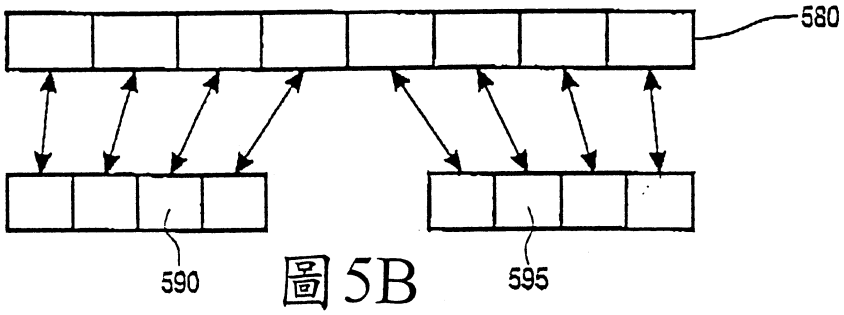
580
590
595

圖5B

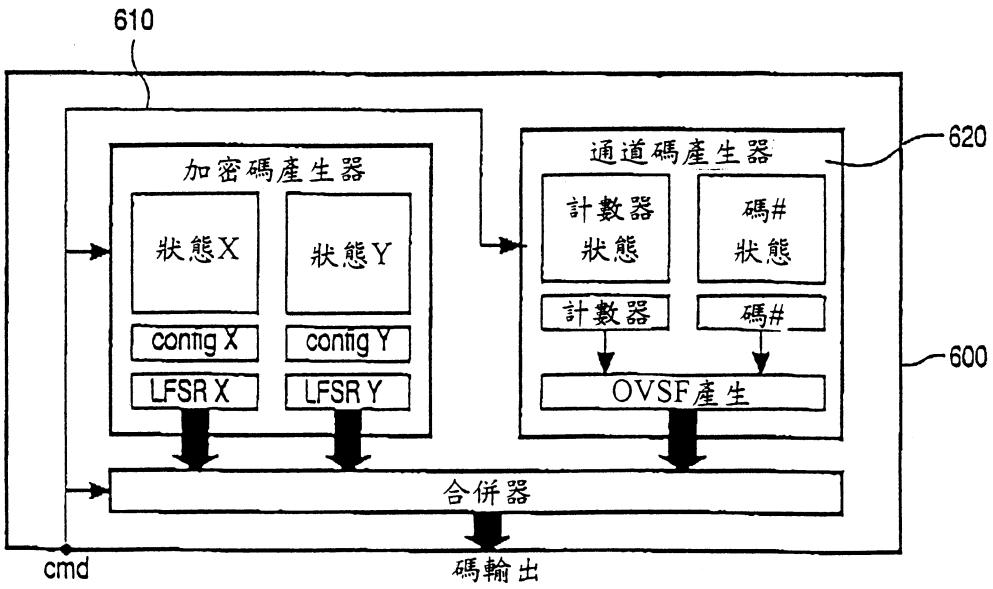
610
加密碼產生器
狀態X
狀態Y
config X
config Y
LFSR X
LFSR Y
通道碼產生器
620
計數器
狀態
碼#
狀態
計數器
碼#
OVSF產生
600
合併器
cmd
碼輸出

圖6

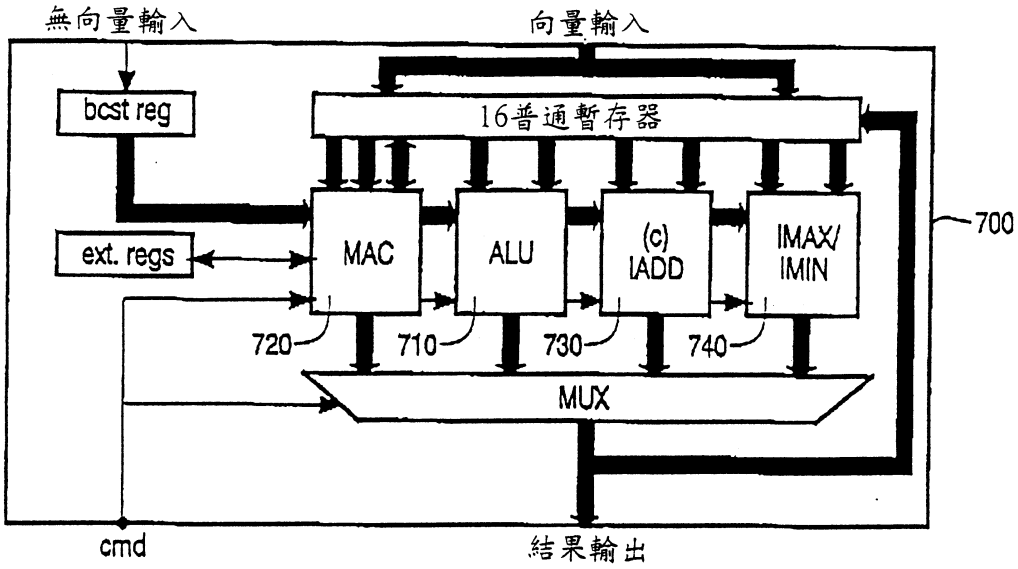
無向量輸入
向量輸入
bcst reg
16普通暫存器
ext. regs
MAC
ALU
(c)
IADD
IMAX/
IMIN
700
720
710
730
740
MUX
cmd
結果輸出

圖7

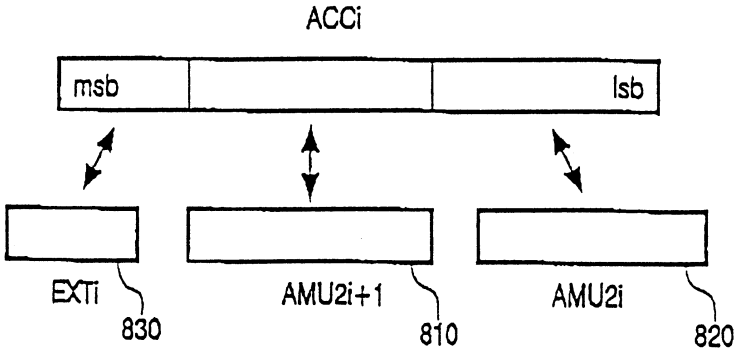
ACCi
msb
lsb
EXTi
830
AMU2i+1
810
AMU2i
820

圖8

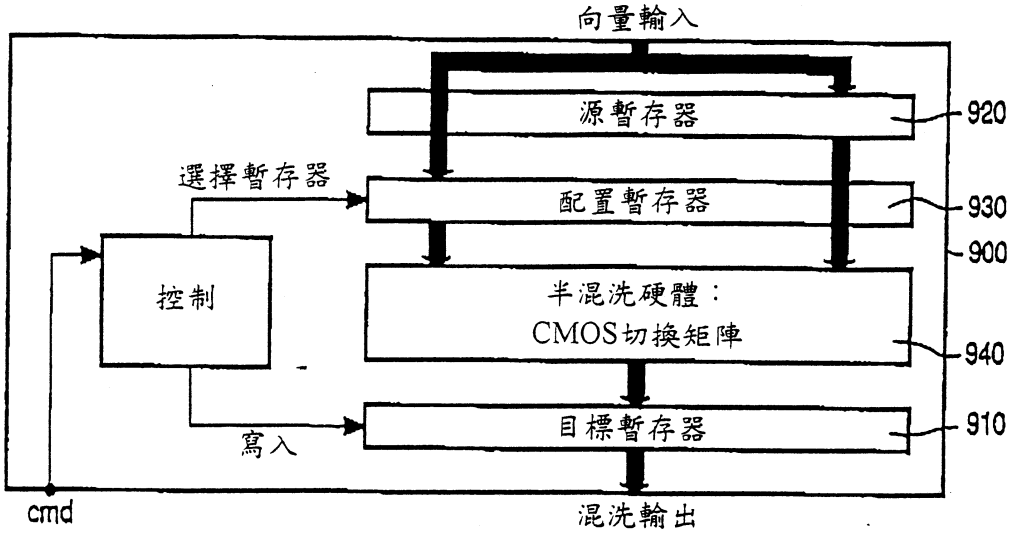
向量輸入
源暫存器
920
選擇暫存器
配置暫存器
930
900
控制
半混洗硬體：
CMOS切換矩陣
940
目標暫存器
910
寫入
cmd
混洗輸出

圖9

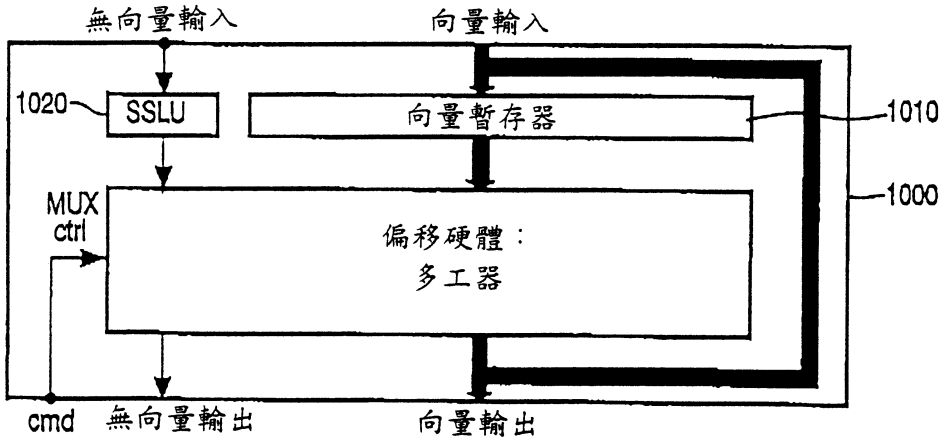
無向量輸入
向量輸入
1020
SSLU
向量暫存器
1010
1000
MUX
ctrl
偏移硬體：
多工器
cmd
無向量輸出
向量輸出

圖10

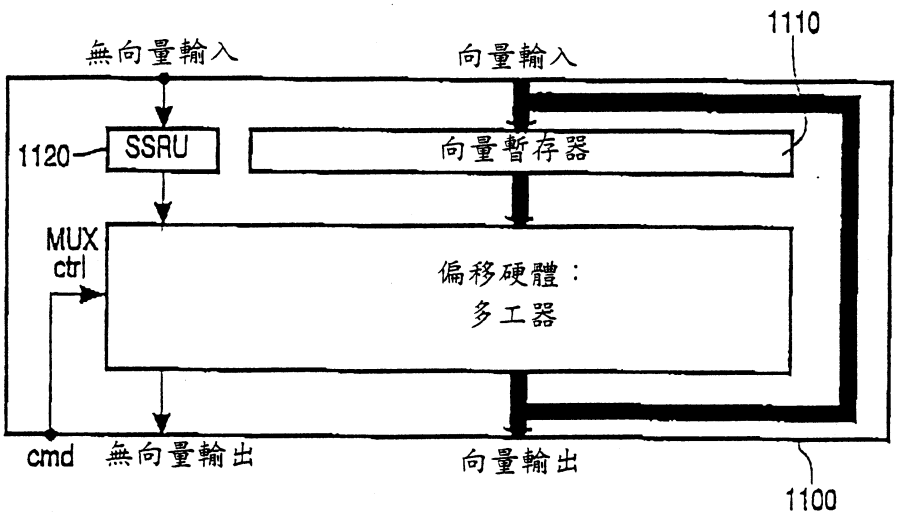
1110
無向量輸入
向量輸入
1120
SSRU
向量暫存器
MUX
ctrl
偏移硬體：
多工器
cmd
無向量輸出
向量輸出
1100

圖11

柒、指定代表圖：

(一)本案指定代表圖為：第（ 2 ）圖。

(二)本代表圖之元件代表符號簡單說明：

210　向量處理部份

220　無向量處理部份

230　控制

250　指令分配單元

252　程式記憶體

260　向量記憶體單元

262　碼產生單元

264　ALU-MAC 單元

266　混洗單元

268　偏左單元

270　偏右單元

捌、本案若有化學式時，請揭示最能顯示發明特徵的化學式：