I522832

(19)中華民國智慧財產局

(12)發明說明書公告本 **(11)證書號數：TW I522832 B**

(45)公告日： 中華民國 105 (2016) 年 02 月 21 日

(21)申請案號：101134968 (22)申請日： 中華民國 101 (2012) 年 09 月 24 日

(51)Int. Cl.： ***G06F21/00 (2013.01)***

(30)優先權：2011/11/17 英國 1119834.8

(71)申請人：ＡＲＭ股份有限公司 (英國) ARM LIMITED (GB)
英國

(72)發明人：荷斯奈爾馬修詹姆士 HORSNELL, MATTHEW JAMES (GB)；柯蕭丹尼爾 KERSHAW, DANIEL (GB)；格利森斯維特理查羅伊 GRISENTHWAITE, RICHARD ROY (GB)；拜爾斯史都華大衛 BILES, STUART DAVID (GB)

(74)代理人：蔡坤財；李世章

(56)參考文獻：

TW	200844832A	TW	200929062A
US	7877582B2	US	2010/0115232A1
US	2011/0161624A1		

審查人員：施易昉

申請專利範圍項數：26 項 圖式數：3 共 83 頁

(54)名稱

支援密碼學之指令

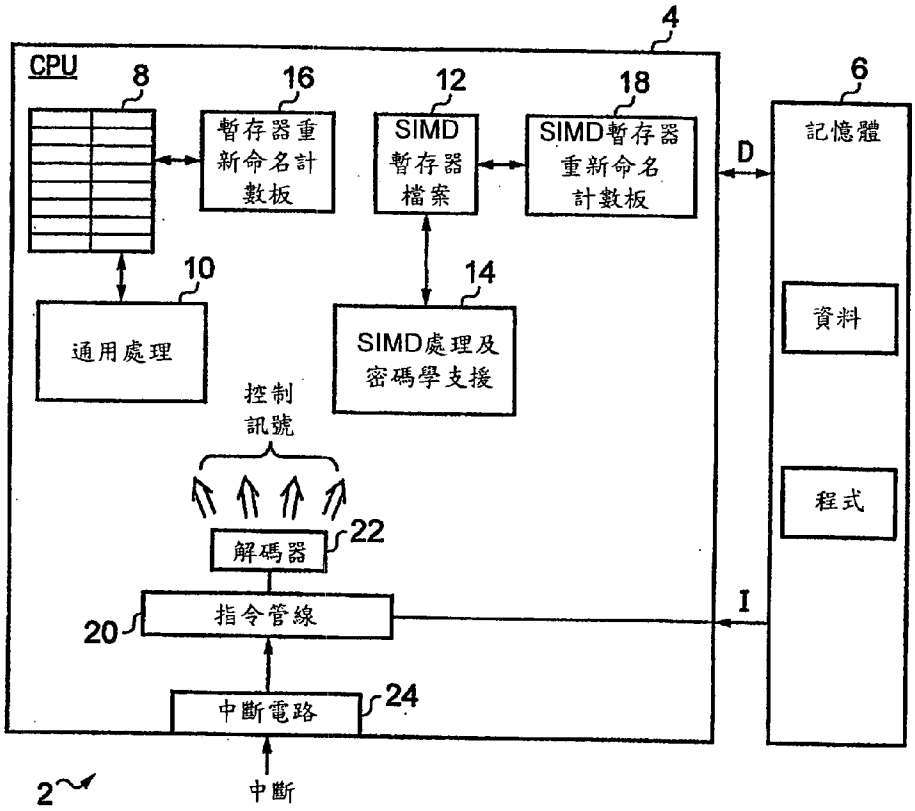
CRYPTOGRAPHIC SUPPORT INSTRUCTIONS

(57)摘要

資料處理系統 2 包括單一指令多重資料暫存器檔案 12 及單一指令多重處理電路 14。單一指令多重資料處理電路 14 支援用於執行散列演算法之部分之密碼學處理指令的執行。運算元儲存在單一指令多重資料暫存器檔案 12 之內。支援密碼學之指令不遵循一般基於通道(lane-based)之處理且產生輸出運算元，在該等輸出運算元中，輸出運算元之不同部分取決於在輸入運算元之內的多個不同元素。

A data processing system 2 includes a single instruction multiple data register file 12 and single instruction multiple processing circuitry 14. The single instruction multiple data processing circuitry 14 supports execution of cryptographic processing instructions for performing parts of a hash algorithm. The operands are stored within the single instruction multiple data register file 12. The cryptographic support instructions do not follow normal lane-based processing and generate output operands in which the different portions of the output operand depend upon multiple different elements within the input operand.

指定代表圖：



第1圖

符號簡單說明：

2・・・資料處理設備

4・・・中央處理單元

6・・・記憶體

8・・・通用暫存器檔案

10・・・通用處理電路

12・・・單一指令多重資料暫存器檔案

14・・・中央處理單元

16・・・通用暫存器重新命名及計數電路

18・・・單一指令多重資料暫存器重新命名及計數電路

20・・・指令管線

22・・・指令解碼器

24・・・中斷電路

I522832

公告本

發明專利說明書

（本說明書格式、順序，請勿任意更動，※記號部分請勿填寫；惟已有申請案號者請填寫）

※ 申請案號：101134968

※ 申請日期：101年9月24日　　※IPC 分類：G06F 21/00 (2013.01)

一、發明名稱：（中文/英文）

支援密碼學之指令/CRYPTOGRAPHIC SUPPORT INSTRUCTIONS

二、中文發明摘要：

資料處理系統2包括單一指令多重資料暫存器檔案12及單一指令多重處理電路14。單一指令多重資料處理電路14支援用於執行散列演算法之部分之密碼學處理指令的執行。運算元儲存在單一指令多重資料暫存器檔案12之內。支援密碼學之指令不遵循一般基於通道（lane-based）之處理且產生輸出運算元，在該等輸出運算元中，輸出運算元之不同部分取決於在輸入運算元之內的多個不同元素。

三、英文發明摘要：

A data processing system 2 includes a single instruction multiple data register file 12 and single instruction multiple processing circuitry 14. The single instruction multiple data processing circuitry 14 supports execution of cryptographic processing instructions for performing parts of a hash algorithm. The operands are stored within the single instruction multiple data register file 12. The cryptographic support instructions do not follow normal lane-based

processing and generate output operands in which the different portions of the output operand depend upon multiple different elements within the input operand.

四、指定代表圖：

(一)本案指定代表圖為：第（ 1 ）圖。

(二)本代表圖之元件符號簡單說明:

2 資料處理設備

4 中央處理單元

6 記憶體

8 通用暫存器檔案

10 通用處理電路

12 單一指令多重資料暫存器檔案

14 中央處理單元

16 通用暫存器重新命名及計數電路

18 單一指令多重資料暫存器重新命名及計數電路

20 指令管線

22 指令解碼器

24 中斷電路

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

無

六、發明說明：

【發明所屬之技術領域】

本發明係關於資料處理系統之領域。更特定言之，本發明係關於在資料處理系統之內提供支援密碼學之指令。

【先前技術】

使用資料處理系統執行密碼學操作是為人們已知的。該等已知密碼學處理操作之實例包括保全散列演算法(Secure Hash Algorithm；SHA)。SHA具有各種不同已知形式，該等形式包括SHA-1、SHA-2、SHA256及SHA512。該等演算法是運算密集的演算法。

支援該等演算法之一已知方法是使用通用處理器，該通用處理器利用該通用處理器之通用暫存器檔案執行通用指令。此方法之一個問題在於在執行該等演算法必須操縱的大量狀態資料（通常可產生160位元及160位元以上之散列值）具有如此結果：操作通常必須被同時對資料的部分操作之單獨程式指令之長序列分離且由該長序列執行，進而導致執行演算法所需之時間量及執行演算法時消耗之能量的不利增加。

另一已知方法是提供專用支援密碼學之處理器，諸如密碼學密碼學共處理器，該密碼學共處理器具有專

用電路，該專用電路用於執行該演算法且通常藉由傳遞待散列之資料之開始的指針，且然後等待接收所得散列值而啟動。此方法之問題在於：由提供專用密碼學硬體產生了額外成本及複雜性。此外，在將專用硬體之操作與裝置之其他操作整合時會產生問題，諸如中斷處理、多任務等等，因為將專用密碼學硬體併入通常在資料處理系統之內提供之機制內以使用資料處理系統處理操作之該等態樣非常困難且複雜。

【發明內容】

自一個態樣而言，本發明提供一種資料處理設備，該資料處理設備包含：

單一指令多重資料暫存器檔案；及

單一指令多重資料處理電路，該單一指令多重資料處理電路經耦接至該單一指令多重資料暫存器檔案且經設置以受單一指令多重資料程式指令的控制，以獨立地對儲存在單獨通道之內的單獨資料元素執行處理操作，該等單獨通道在該單一指令多重資料暫存器檔案之輸入運算元暫存器之內；其中

該單一指令多重資料處理電路經設置以由進一步程式指令控制，以對向量資料值執行進一步處理操作，該向量資料值包含保持在該單一指令多重資料暫

存器檔案之輸入運算元暫存器之內的資料元素序列，以產生儲存在該單一指令多重資料暫存器檔案之輸出運算元暫存器之內的輸出運算元，該輸出運算元具有含有一值的第一部分，該值取決於在該資料元素序列之內的所有資料元素。

本發明技術認識到，許多資料處理系統已具備單一指令多重資料處理機制。該等單一指令多重資料處理機制通常包括單一指令多重資料暫存器檔案，該單一指令多重資料暫存器檔案具有能夠儲存且操縱大資料寬度運算元之大儲存容量，該等大資料寬度運算元通常在單一指令多重資料處理中涉及。在單一指令多重資料處理中，在單一程式指令之控制之下獨立地處理資料的單獨通道是正常的。例如，資料之單獨通道可包含色彩像素值或其他向量值之分量值，該等所有值皆將經受相同處理操作，諸如縮放。本發明技術認識到，單一指令多重資料暫存器檔案之儲存能力可利用進一步程式指令重複使用，該進一步程式指令不遵循單一指令多重資料程式指令之正常形式。特定言之，通道之處理不必為獨立的，且產生之輸出運算元可具有含有一值的第一部分，該值取決於在形成輸入之向量資料值之內的所有資料元素。

在單一指令多重資料程式指令之區域外的單一指令多重暫存器檔案之重複使用可應用於各種領域，諸如資料壓縮及資料密碼術。本技術尤其非常適合於資料密碼術。

在此上下文中，進一步程式指令可經安排以執行迭代處理操作，該迭代處理操作消耗連續的資料字及至少部分中間散列值以產生輸出散列值。散列值產生通常需要操縱大量的資料及暫存器檔案，同時具有儲存及操縱非常長的運算元值之能力。

進一步程式指令之一形式為其中該進一步程式指令具有自該單一指令多重資料暫存器檔案讀取之第一輸入運算元 Qd[127:0]及第二輸入運算元 Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

```
X[127:0]=Qd[127:0];
Y[31:0]=Sn[31:0];
for(I=0 to(2^N-1));
 {
    Index=(I*32);
```

t1[31:0]=OP FUNC(X[63:32], X[95:64], X[127:96]);

Y[31:0]=Y[31:0]+ROL(X[31:1], 5)+T1[31:0]+Vm[Index+31:Index];

X[63:32]=ROL(X[63:32], 30);

T2[31:0]=Y[31:0];

Y[31:0]=X[127:96];

X[127:0]={X[95:0]:T2[31:0]}

}

Q_{output}[127:0]=X[127:0];

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

迭代程式指令之該形式非常適合於實施 SHA-1 演算法。應瞭解，上文定義之操作可以各種偽代碼形式給出，且上文定義之操作可以各種不同硬體形式實施，熟習該技術領域之技術者將很好地理解此點。特定言之，低電路額外負擔實施可再循環值以執行迭代操作，其中較高效能實施可尋求並聯執行至少部分不

同迭代。

進一步程式指令之另一形式具有自該單另一指令多重資料暫存器檔案讀取之第一輸入運算元Qd[127:0]及第二輸入運算元Sn[31:0]兩者，且該向量資料值包含其中Index為0至2^N之Vm[Index+31:Index]，其中N為正整數，該進一步處理操作產生該輸出運算元Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

```
X[127:0]=Qd[127:0];
Y[31:0]=Sn[31:0];
for (I=0 to (2^N-1));
{
    Index=(I*32);
    T1[31:0]=OP_FUNC(X[63:32], X[95:64], X[127:96]);
Y=Y+ROL(X[31:0],5)+T1[31:0]+Vm[(Index+31):Index];
    X[63:32]=ROL(X[63:32],30);
    T2[31:0]=Y;
    Y=X[127:96];
    X[127:0]={X[95:0]:T2[31:0]};
```

}

$Qd_{output}[127:0]=\{0:Y[31:0]\};$

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

由 OP FUNC 評估之函數之選擇可依賴於在進一步程式指令之內的特定欄位進行，或該選擇可依賴於在處理待散列之資料值之當前輸入區塊期間已執行多少次迭代。

在一些實施例中，單一指令多重資料暫存器檔案可能不具有將所有第一輸入運算元及第二輸入運算元儲存在單個暫存器中之能力，且因此該等輸入運算元可儲存於在單一指令多重資料暫存器檔案之內的單獨暫存器內。在其他實施例中，第一輸入運算元及第二輸入運算元可儲存在共享暫存器之內，且第一輸入運算元及第二輸入運算元可視為單個輸入運算元。

在進一步實施例中，與上述進一步程式指令結合或代替上述進一步程式指令，本發明技術可提供對進一步程式指令之支援，該進一步程式指令具有自該單一

指令多重資料暫存器檔案讀取之第一輸入運算元Qd[127:0]及第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

```
X[127:0]=Qd[127:0];
Y[127:0]=Qn[127:0];
for(I=0 to (2^N-1));
 {
    Index=(I*32);
 TCh[31:0]=Choose(Y[31:0],Y[63:32],Y[95:64]);
 TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);
T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]+
Vm[Index+31:Index];
 X[127:96]=T1[31:0]+X[127:96];
 Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]
 T2[31:0]=Y[127:96];
 Y[127:0]={Y[95:0]:X[127:96]};
 X[127:0]={X[95:0]:T2[31:0]}
 }
```

Q_{output}[127:0]=X[127:0];

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

以類似之方式，進一步程式指令亦可具有一形式，在該形式中，進一步程式指令具有自該單另一指令多重資料暫存器檔案讀取之第一輸入運算元 Qd[127:0]及第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

X[127:0]=Qn[127:0];

Y[127:0]=Qd[127:0];

for(I=0 to (2^N-1));

{

Index=(I*32);

TCh[31:0]=Choose(Y[31:0], Y[63:32], Y[95:64]);

```
TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);
T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]
+Vm[Index+31:Index];
X[127:96]=T1[31:0]+X[127:96];
Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]
T2[31:0]=Y[127:96];
Y[127:0]={Y[95:0]:X[127:96]};
X[127:0]={X[95:0]:T2[31:0]}
}
```

Qd_{output}[127:0]=Y[127:0];

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

上述兩種形式之進一步程式指令特別適合於支援 SHA-224 演算法及 SHA256 演算法。

用於管理進一步程式指令之處理之機制可方便地與單一指令多重資料處理電路結合。用於管理進一步處理指令之處理之機制使用單一指令多重資料指令

暫存器檔案，且當用於管理進一步程式指令之處理（例如中斷處理、排程）之機制與單一指令多重資料處理電路之機制整合時，實施可簡化。

管理可與單一指令多重資料處理電路之程式指令整合之進一步程式指令的處理之態樣包括暫存器重新命名、指令排程、指令發出、指令報廢及指令中斷。單一指令多重資料處理電路通常已包括管理且支援該等操作之電路元件，且進一步程式指令可相對容易地整合至該管理支援中。此舉提供了以下優點：若中斷在產生密碼學散列值中途發生，則正常中斷處理機制可用以服務該中斷且在服務該中斷之後重新啟動或繼續散列計算而具有較少額外管理負擔或複雜性。

對散列演算法之支援係進一步藉由提供旋轉指令增強，該旋轉指令具有輸入運算元 Sm[31:0]且產生具有一值之輸出運算元 Sd[31:0]，該值與 Sm[31:0]右旋轉兩個位元位置給出的值相同。

除產生中間散列值之外還應執行之密碼學散列演算法處理的另一態樣為更新在正在處理的檔案之內的資料元素之排程。應根據散列產生之工作負載平衡該排程更新，以免引入處理量之不利瓶頸。因此，本發明之一些實施例規定該單一指令多重資料處理電

路經設置以由第一排程更新指令控制，該第一排程更新指令具有第一輸入運算元 Sp[127:0]及第二輸入運算元 Sq[127:0]且產生具有一值之輸出運算元 Sr[127:0]，該值與由以下步驟給出之值相同：

T[127:0]={Sp[63:0]:Sq[127:64]}及

Sr[127:0]=T[127:0] XOR Sr[127:0] XOR Sq[127:0].

此外，一些實施例規定該單一指令多重資料處理電路經設置以由第二排程更新指令控制，該第二排程更新指令具有輸入運算元 Ss[127:0]且產生具有一值之輸出運算元 St[127:0]，該值與由以下步驟給出之值相同：

T[127:0]=St[127:0] XOR {32{0}:Ss[127:32]};

St[95:0]={T[94:64]:T[95]:T[62:32]:T[63]:T[30:0]:T[31]};及

St[127:96]=({T[126:96]:T[127]})XOR({T[29:0]:T[31:30]}).

上述兩種形式之程式指令特別適合於支援 SHA-256 及 SHA-224 演算法。

為了幫助支援其他形式之散列演算法之排程產生，如此該單一指令多重資料處理電路之一些實施例經設置以由第一排程更新指令控制，該第一排程更新

指令具有輸入運算元 Sp[127:0]且產生具有一值之輸出運算元 Sq[127:0]，該值與由以下步驟給出之值相同：

T[127:0]={Sp[31:0]:Sq[127:32]};

T[127:0]=VecROR32(T[127:0],7)XOR VecROR32(T[127:0], 18) XOR VecROR32(T[127:0], 3);

及

Sq[127:0]=VecADD32(T[127:0], Sq[127:0]),

其中 VecROR32 (A, B)為在 A 之內的每一 32 位元字單獨右旋轉 B 位元位置，且 VecADD32 (A, B)為在 A 之內的每一 32 位元字單獨增加至在 B 之內的相應 32 位元字。

進一步實施例另外提供該單一指令多重資料處理電路，該單一指令多重資料處理電路經設置以由第一排程更新指令控制，該第一排程更新指令具有第一輸入運算元 Sp[127:0]及第二輸入運算元 Sq[127:0]且產生具有一值之輸出運算元 Sr[127:0]，該值與由以下步驟給出之值相同：

T0[127:0]={Sq[31:0]:Sp[127:32]};

T1[63:0]=Sq[127:64];

T1[63:0]=VecROR32(T1[63:0],17)XOR

VecROR32(T1[63:0], 19) XOR VecROR32(T1[63:0], 10);

T3[63:0]=VecADD32(Sr[63:0], T0[63:0]);

T1[63:0]=VecADD32(T3[63:0], T1[63:0]);

T2[63:0]=VecROR32(T1[63:0],17)XOR VecROR32(T1[63:0], 19) XOR VecROR32(T1[63:0], 10);

T3[63:0]=VecADD32(Sr[127:64], T0[127:64]);及

Sr[127:0]={VecADD32(T3[63:0], T2[63:0]):T1[63:0]},

其中 VecROR32 (A, B)為在 A 之內的每一 32 位元字單獨右旋轉 B 位元位置，且 VecADD32 (A, B)為在 A 之內的每一 32 位元字單獨增加至 B 之內的相應 32 位元字。

上述兩種形式之程式指令特別適合於支援 SHA-256 演算法。

自另一態樣而言，本發明提供資料處理設備，該資料處理設備包含：

單一指令多重資料暫存器檔案構件，用於儲存單一指令多重資料運算元；及

單一指令多重資料處理構件，用於在單一指令多重

資料程式指令之控制下執行處理操作，該單一指令多重資料處理構件經耦接至該單一指令多重資料暫存器檔案構件，且對儲存在該單一指令多重資料暫存器檔案構件之輸入運算元暫存器之內的單獨通道內之單獨資料元素獨立地執行該處理操作；其中

該單一指令多重資料處理構件係由進一步程式指令控制，以對向量資料值執行進一步處理操作，該向量資料值包含保持在該單一指令多重資料暫存器檔案構件之輸入運算元暫存器之內的資料元素序列，以產生儲存在該單一指令多重資料暫存器檔案構件之輸出運算元暫存器之內的輸出運算元，該輸出運算元具有含有一值的第一部分，該值取決於在該資料元素序列之內的所有資料元素。

自本發明之進一步態樣而言，本發明提供一種處理資料之方法，該方法包含以下步驟：

將單一指令多重資料運算元儲存在單一指令多重資料暫存器檔案之內；

在單一指令多重資料程式指令之控制下，獨立地對儲存在該單一指令多重資料暫存器檔案之輸入運算元暫存器之內的單獨通道內的單獨資料元素執行處理操作；及

在進一步程式指令之控制下，對向量資料值執行進一步處理操作，該向量資料值包含保持在該單一指令多重資料暫存器檔案之輸入運算元暫存器之內的資料元素序列，以產生儲存在該單一指令多重資料暫存器檔案之輸出運算元暫存器之內的輸出運算元，該輸出運算元具有含有一值的第一部分，該值取決於在該資料元素序列之內的所有資料元素。

本發明之另一態樣提供虛擬機實施，該虛擬機實施提供在通用電腦上之執行環境，該執行環境許可上文詳述之程式指令如同該等程式指令在上文詳述之資料處理設備上執行一樣執行。在本文中涵蓋本技術之該等虛擬機實施。

【實施方式】

第 1 圖示意圖示形式為中央處理單元 4 之資料處理設備 2，該中央處理單元 4 耦接至記憶體 6，該記憶體 6 儲存待操縱之資料及待執行之程式指令。中央處理單元 4 包括通用暫存器檔案 8、通用處理電路 10、單一指令多重資料暫存器檔案 12 及單一指令多重資料處理電路 14。通用暫存器檔案 8 通常含有低位元寬度通用暫存器（例如 32 位元或 64 位元），諸如具有由 Cambridge, England 之 ARM Limited 生產之處理器

的通用暫存器檔案支援之形式之暫存器。單一指令多重資料暫存器檔案 12 通常包括更大的暫存器，且在單一指令多重資料暫存器檔案 12 之內的資料儲存可取決於所利用之暫存器大小說明符以不同方式劃分以形成不同暫存器。單一指令多重資料暫存器檔案 12 之形式可為在由 Cambridge, England 之 ARM Limited 生產之處理器之一些實施中支援的 Neon 暫存器檔案之形式。

通用暫存器重新命名及計數電路(score boarding circuitry)16 與通用暫存器檔案 10 相關聯，且單一指令多重資料暫存器重新命名及計數電路 18 與單一指令多重資料暫存器檔案 12 相關聯。暫存器重新命名及計數自身是將為本技術領域中之工人所熟知的已知技術且將不再於本文中進一步描述。與提供暫存器重新命名及計數用於一般單一指令多重資料處理指令之方式相同，暫存器重新命名及計數可應用於在下文中進一步描述之密碼學處理指令之支援中所利用的暫存器。因此，已提供用於支援暫存器重新命名、指令排程、指令發出、指令報廢及指令中斷之機制可由支援密碼學之程式指令重新使用，且相應地，該等支援密碼學之指令之操作可較好地與中央處理單元 4

之整體操作整合。

程式指令 I 係接收自記憶體 6 且傳遞至指令管線 20。指令解碼器 22 將程式指令解碼且產生控制訊號，該控制訊號控制暫存器檔案 8、12 及處理電路 10、14，以及在中央處理單元 4 之內的其他元件之操作。中斷電路 24 回應於外部產生的中斷訊號 int 以中斷當前由中央處理單元 4 執行之處理，且開始執行為本技術領域中之彼等技術者所熟知的中斷處理碼。應瞭解，中央處理單元 4 通常將包括許多附加電路元件，且為了清楚起見，已將該等附加電路元件從第 1 圖中省略。

第 2 圖示意圖示在散列產生演算法之一形式之內的資料流。待散列之檔案 26 被分割成 64 位元組區塊 28，該 64 位元組區塊 28 經進一步分割成作為一輸入提供至散列演算法 30 之四個 32 位元字的輸入向量。亦在散列操作之開始時提供散列種子值 32。散列操作使用兩個主程式迴路，該兩個主程式迴路分別負責散列更新 34 及排程更新 36。該等主迴路係藉由在單一指令多重資料處理電路 14 中提供支援兩個迴路之專用程式指令來平衡。中間散列值 38 由散列更新迴路 34 產生且作為散列演算法回饋，該散列演算法持續處

理輸入資料之區塊 28。當已處理區塊 28（例如，在 SHA-1 之情況下經歷 80 個散列更新）時，輸出散列值 40 然後係藉由將當前中間散列值 38 添加至該輸出散列值 40 中而更新。重複該方法直至所有檔案 26 已消耗為止。此舉總體上產生檔案 26 之所得散列值。

散列更新迴路 34 將被執行多次，且散列更新迴路 34 自身執行指令，該等指令中之每一指令具有將如下所述之該等指令自身之指令內部迭代。執行排程更新 36 以便平衡散列更新。排程更新可經向量化以改良效能，如將在下文中描述。

第 3 圖示意圖示根據本發明技術之進一步處理指令如何接收包含複數個資料元素之向量資料值 42。然後，支援密碼學之指令對該向量資料值 42 執行處理操作以產生具有第一部分 44 之輸出運算元，該第一部分 44 取決於向量資料值 42 之第一資料元素及在向量資料值之內的兩個或兩個以上進一步資料元素兩者。該行為與典型單一指令多重資料程式指令形成對比，在該等典型單一指令多重資料程式指令中處理操作是基於通道的且在不同通道之內的資料值之間存在有限的互動。

本技術之一實施是一組指令，該組指令將兩個演算

法作為目標，亦即，演算法 SHA-1 及 SHA-256。指令亦對 SHA-224 演算法有利，SHA-224 演算法需要與 SHA-256 相同的操作。SHA 演算法為由國家標準技術局(National Institute of Standards and Technology；NIST)規定之保全散列演算法系列。該等演算法之之規範公開可用。該等演算法通常用於驗證在數位系統之內的資料。

吾人從描述 SHA 演算法之高階操作且包括用於 SHA-1 及 SHA-256 演算法之偽碼開始。

SHA 演算法之高階操作（已知；FIPS 180-4）

演算法中之每一者處理 64 位元組之資料，且產生散列摘要；在 SHA-1 之情況下，資料為 160 位元之長度，且在 SHA-256 之情況下，資料為 256 位元之長度。長度大於 64 位元組之資料串流被分為 64 位元組區塊。在串流或區塊之長度小於 64 位元組之情況下，可按照在聯邦資訊處理標準(Federal Information Processing Standard；FIPS) 180-4 中所規定將區塊填充至 64 位元組。

除非另有說明，否則演算法之以下描述假設一字為 32 位元無符號整數值。假設字由來自雙模二元資料格式形式之 64 位元組之區塊的資料之 4 個相連位元組

組成。

兩個演算法藉由初始化工作散列摘要開始。若資料區塊為給定資料串流中之第一者，則將散列摘要初始化至固定種子值。若區塊為資料串流之延續，則將散列摘要初始化至由先前區塊計算之散列摘要。在 FIPS 180-4 中規定種子值。

演算法使用排程更新操作擴展區塊。對於 SHA-1，此擴展將區塊自初始 16 資料字擴展為 80 資料字；對於 SHA-256，此擴展將區塊自初始 16 資料字擴展為 64 資料字。排程更新操作使用固定邏輯異或（xors）將來自排程之四個字結合，並且將該四個字移位且旋轉，以在擴展排程中產生下一個字。初始 16 個字在擴展排程中保持不變。

然後，擴展排程中之每一字具有添加至該字之鍵值。在 SHA-1 中，存在 4 個鍵常數，每一鍵常數應用於來自擴展區塊之一組 20 個字。在 SHA-256 中，存在 64 個鍵常數，擴展區塊之每一字有一個鍵常數。在 FIPS 180-4 中定義鍵常數。

在已擴展區塊且已添加鍵常數之後，使用散列更新函數處理每一字，該散列更新函數經由一系列固定邏輯異或併入該字，並且將彼字移位且旋轉至散列摘要

中。

最終，在已使用散列更新函數處理來自擴展區塊的每一字之後，將散列摘要添加至先前散列摘要值。

如 FIPS 180-4 中所規定，排程可實施為 80/64 字(SHA-1/SHA-256)之集合或實施為 16 字之循環佇列。

為完整起見，假定循環佇列之 SHA-1 及 SHA-256 之偽碼演算法如下。

SHA-1 演算法偽碼

```
uint32 w[0:15]=16 4-bytes (big-endian) input[];
uint32 wk[0:15]=w[0:15]+k[0:15];
uint32 a:e=io->hashes[0:4];
forround=0:63{
hash_update(round,wk[round]);
w[round]=schedule_update(round,w);
wk[round]=w[round]+k[round];
}
For round=64:79
hash_update(round,w[round]);
io->hashes[0:4]+=a:e;
```

SHA-1 散列更新代碼如下：

```
hash_update(int round, uint32 wk) {
```

```
e+=FN(round,b,c,d)+ROL(a,5)+wk;
b=ROL(b,30);
rotate(a,b,c,d,e)to(e,a,b,c,d)
}
```

其中：

```
if round<20, FN=choose(b,c,d);
else if round <40,FN=parity(b,c,d);
else if round<60,FN=majority(b,c,d);
else FN=Parity(b,c,d);
choose(b,c,d)=(((c^d)&b)^d)
parity(b,c,d)=(b^c^d)
majority(b,c,d)=(b&c)|((b|c)&d)
```

SHA-1 排程更新代碼如下：

```
uint32 schedule_update(int round, uint32 *w){
return
ROR(w[round-3]^w[round-8]^w[round-14]^w[round-16],31);
}
```

SHA-256 演算法偽碼

```
uint32 a:h=io->hashes[0:7];
uint32    w[0:15]=16    4-bytes    (big-endian)
```

```
input[0:63] ;
  uint32 wk[0:15]=w[0:15]+k[0:15];
  for round=0:47{
  hash_update(wk[round]);
  w[round]=schedule_update(round,w);
  wk[round]=w[round]+k[round];
  }
  for round=48:63
  hash_update(wk[round]);
  io->hashes[0:7]+=a:h;
```

SHA-256 散列更新代碼如下：

```
hash_update(uint32 wk){
t=h+Sigma1(e)+Choose(e,f,g)+wk;
d+=t;
h=t+Sigma0(a)+Majority(a,b,c);
rotate(a,b,c,d,e,f,g,h)to(h,a,b,c,d,e,f,g) ;
}
```

其中：

```
Sigma0(x)=ror(x,2)^ror(x,13)^ror(x,22);
Sigma1(x)=ror(x,6)^ror(x,11)^ror(x,25);
Choose(b,c,d)=(((c^d)&b)^d)
```

Majority(b,c,d)=((b&c)|((b|c)&d)

同樣地，SHA-256 排程更新偽碼如下：

```
uint32 schedule_update(int round, uint32 *w){
return
w[round]+sigma1(w[round-2])+w[round-7]+sigma0(w[
round-15]);
}
```

其中：

```
sigma0(x)=ror(x,7)^ror(x,18)^shr(x,3);
sigma1(x)=ror(x,17)^ror(x,19)^shr(x,10);
```

SHA 演算法工作狀態（可源自 FIPS 180-4 規範）

約束經採用以加速 SHA 演算法之方法之 SHA 演算法的一態樣為處理資料區塊所需要之工作狀態量（如先前所述）。單一指令多重資料暫存器檔案保持且操縱此狀態之能力解決該約束。

下表概括對於 SHA-1 及 SHA-256 之狀態要求。

SHA-1 狀態	
始/先前散列摘要	5x32 位元字
工作散列摘要	5x32 位元字
排程	16x32 位元字
鍵常數	4x32 位元字
SHA-256 狀態	
初始/先前散列摘要	8x32 位元字

工作散列摘要	8x32 位元字
排程	16x32 位元字
鍵常數	64x32 位元字

使用 SHA-1 或 SHA-256 演算法中之任一者構建能夠處理資料區塊之專用 SHA 單元（例如，共處理器）需要投資在固定目標狀態中。該狀態無法輕易地由 RISC 微處理器上之其他操作使用。

將 SHA 演算法分成三元形式 RISC 指令

為了避免固定目的狀態，吾人已藉由可於 RISC 微處理器上處理演算法之方法將該等演算法分離，該 RISC 微處理器監視三元指令格式且使用單一指令多重資料處理電路及暫存器檔案。

RISC 三元形式之典型約束為三個暫存器中之僅一個暫存器被定義為目標地。然而，目標地可使用為來源。

吾人使用 SIMD 暫存器以便可處理比使用通用暫存器的每指令處理之資料更多之資料。

藉由監視三元指令格式，指令能夠使用為現代微處理器所常見之重新命名、排程、發出、所得及報廢邏輯。

由於所有狀態及相依性係由指令定義，所以處理亂序執行、中斷及思考之管線機制仍然有效；不需要附

加控制邏輯即可以保持所提議指令之正確執行。

SHA-1 散列更新指令

如先前所述之 SHA-1 散列更新函數將 32 位元字併入 160 位元散列摘要中。函數由固定移位、固定邏輯異或/邏輯和（and）/邏輯或（ors）及固定旋轉組成。

```
hash_update(int round, uint32 wk){
e+=FN(round,b,c,d)+ROL(a,5)+wk;
b=ROL(b,30);
rotate(a,b,c,d,e)to(e,a,b,c,d)
}
where:
if round<20,FN=choose(b,c,d);
else if round<40,FN=parity(b,c,d);
else if round<60,FN=majority(b,c,d);
else FN=Parity(b,c,d);

choose(b,c,d)=(((c^d)&b)^d)
parity(b,c,d)=(b^c^d)
majority(b,c,d)=(b&c)|((b|c)&d)
```

SHA-1 散列摘要為 160 位元，且因此對整個摘要加上 32 位元字進行之操作不可能為 32 位元之通用三元

RISC 格式，且該等操作將需要很大努力才能實現為64位元通用三元 RISC 格式；將需要更多內務處理才能將 32 位元資料值插入第三 64 位元運算元之高 32 位元中。

為此，該示例性技術將 SHA-1 散列函數映射至一組四個高階 SIMD 指令上；該四個高階 SIMD 指令為 SHA1C、SHA1P、SHA1M 及 SHA1H。

SHA1C Qd, Sn, Vm.4S [OP=C, OP_FUNC=choose]

SHA1P Qd, Sn, Vm.4S [OP=P, OP_FUNC=parity]

SHA1M Qd, Sn, Vm.4S [OP=M, OP_FUNC=majority]

SHA1H Sd, Sn

指令 SHA1C、SHA1P 及 SHA1M 採用三個運算元。Qd 保持摘要散列之前 4 個 32 位元字，其中 Sn 保持第 5 個 32 位元字。第三運算元 Vm 為一向量，該向量在初始實施例中保持四個 32 位元字。如此允許待由指令處理之散列更新函數之 4 個迭代。偽碼定義該等指令之操作如下。應瞭解，根據偽碼定義指令之操作將為本技術領域中之彼等技術者所熟知，且一旦已定義偽碼，則進行（執行）由偽碼定義之指令之電路的實現是常規任務。

```
SHA1<OP>Qd, Sn, Vm.4S
    X=Qd;
    Y=Sn;
    for(i=0 to 3)
    {
    Index=(i*32);
  t1<31:0>=OP_FUNC(X<63:32>,X<95:64>,X<127:96
>);
  Y=Y+ROL(X<31:0>,5)+t1<31:0>+Vm<(index+31):i
ndex>;
    X<63:32>=ROL(X<63:32>,30);
     //Rotate
     t2<31:0>=Y;
       Y=X<127:96>;
       X<127:0>={X<95:0>:t2<31:0>};
    }
  Qd=X;
```

相應地，根據示例性實施例，單一指令多重資料處理電路經設置以由進一步程式指令控制，該進一步程式指令具有自該單一指令多重資料暫存器檔案讀取之第一輸入運算元 Qd[127:0]及第二輸入運算元

Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

```
X[127:0]=Qd[127:0];
Y[31:0]=Sn[31:0];
for(I=0 to (2^N-1));
 {
    Index=(I*32);
    t1[31:0]=OP  FUNC  (X[63:32],  X[95:64],
X[127:96]);

Y[31:0]=Y[31:0]+ROL(X[31:1],5)+T1[31:0]+Vm[Inde
x+31:Index];
    X[63:32]=ROL(X[63:32], 30);
    T2[31:0]=Y[31:0];
    Y[31:0]=X[127:96];
    X[127:0]={X[95:0]:T2[31:0]}
 }
Qd_ouuput[127:0]=X[127:0];
```

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

該等指令之另一實現可涉及用於在 choose ()函數、parity ()函數及 majority ()函數之間選擇之選擇。

SHA1HASHQd, Sn, Vm.4S, #OP // #OP 其中#_1 選擇 C，#_2 選擇 P，#_3 選擇 M。

RISC 指令格式之約束在於僅散列摘要之前 4 個字可由 SHA1C、SHA1P 及 SHA1M 指令返回至 128 位元暫存器 Qd 中。因此，建議指令 SHA1H 返回散列摘要之第 5 個字。

在首次實現中，SHA1H 可實施為：

SHA1H Sd, Sn

Sd=ROR(Sn, 2);

此舉遵循以下監視：在四次迭代後第 5 散列摘要值為 Qd[0]之初始值的旋轉。

SHA1 散列更新指令變體

SHA1C 指令、SHA1P 指令及 SHA1M 指令之變體可由本發明技術之其他變體擴展以允許 Vm.8S 或 Vm.16S 運算元。該等變體包括在本發明技術之內。

如此將允許在單一指令之內處理散列更新函數之8次及16次迭代。亦即，仍將需要Vm.4S變體，因為需要散列更新函數才能每20個迭代之後變化。

作為一實例，該SHA1<OP>Vm.8S變體：

```
SHA1<OP>Qd, Sn, Vm.8S
X=Qd;
Y=Sn;
for(i=0 to 7)
{
Index=(i*32);
t1<31:0>=OP_FUNC(X<63:32>,X<95:64>,X<127:96
>);
Y=Y+ROL(X<31:0>,5)+t1<31:0>+Vm<(index+31):i
ndex>;
X<63:32>=ROL(X<63:32>,30);
//Rotate
t2<31:0>=Y;
Y=X<127:96>;
X<127:0>={X<95:0>:t2<31:0>};
}
Qd=X;
```

操作過 8 次迭代及 16 次迭代（Vm.8S 及 Vm.16S）之變體將另外需要 SHA1C2 指令、SHA1P2 指令及 SHA1M2 指令。該等指令將在 8 次或 16 次迭代之後為散列摘要中之第 5 個字產生適當值。該等新的指令將以與 SHA1C 指令、SHA1P 指令及 SHA1M 指令類似之方式實施，但是在 Qd 暫存器中返回第 5 個散列摘要字，例如：

```
SHA1<OP>2 Qd, Sn, Vm.8S
X=Qd;
Y=Sn;
for(i=0 to 3)
{
Index=(i*32);
t1<31:0>=OP_FUNC(X<63:32>,X<95:64>,X<127:96>);
Y=Y+ROL(X<31:0>,5)+t1<31:0>+Vm<(index+31):index>;
X<63:32>=ROL(X<63:32>, 30);
//Rotate
t2<31:0>=Y;
Y=X<127:96>;
```

X<127:0>={X<95:0>:t2<31:0>};

}

Qd={0:Y<31:0>};

相應地，根據示例性實施例，單一指令多重資料處理電路經設置以由進一步程式指令控制，該進一步程式指令具有自該單一指令多重資料暫存器檔案讀取之第一輸入運算元 Qd[127:0]及第二輸入運算元 Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

X[127:0]=Qd[127:0];

Y[31:0]=Sn[31:0];

for(I=0 to(2^N-1));

{

Index=(I*32);

T1[31:0]=OP_FUNC(X[63:32],X[95:64],X[127:96]);

Y=Y+ROL(X[31:0],5)+T1[31:0]+Vm[(Index+31):Index];

X[63:32]=ROL(X[63:32],30);

T2[31:0]=Y;

Y=X[127:96];

X[127:0]={X[95:0]:T2[31:0]};

}

Qd_{output}[127:0]={0:Y[31:0]};

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

若較寬 SIMD 資料路徑可用，可實現指令之其他變體，該其他變體將整個散列摘要返回成為八位元字（8x32 位元）：

SHA1<OP>Od, Vn.4S

SHA1<OP>Od, Vn.8S

該等指令將處理散列函數之 4 次及 8 次迭代。

SHA1 散列更新微架構選項

對於該等指令之微架構實施存在各種選項：

該等指令之高效能實現可選擇增建一些迭代邏輯且執行更多並行化執行。

微架構可選擇使用多循環級以減少暫時管線狀

態，且因此減少功率消耗。

可以進位儲存形式進行中間架構。

在更廣泛的變體中，在可能需要顯式 SHA1<OP>2 指令之情況下，偵測 SHA1<OP>2 操作何時遵循相應 SHA1<OP>函數是有可能的。在彼等情況下，防止第二計算且僅轉發來自資料路徑之結果應是可能的。如此將在管線中需要一些暫時狀態。

SHA1 排程更新指令

實現自 SHA-1 演算法之加速需要在散列更新函數與排程更新函數之間的平衡。

如先前所述之 SHA-1 排程更新函數將來自資料排程之四個 32 位元字結合為單一所得字，該單一所得字擴展排程，或在循環佇列之情況下，該單一所得字將排程中之字重寫。

排程更新操作由邏輯異或及固定旋轉組成。

```
uint32 schedule_update(int round, uint32*w){
return
ROR(w[round-3]^w[round+8]^w[round-14]^w[round-16],31);
}
```

或在循環佇列形式中：

Void schedule_update(int round, uint32 w[0..15]){

w[round]=ROR(w[round+13$_{mod16}$]^w[round+8$_{mod16}$]^w[round+2$_{mod16}$]^w[round],31);

}

操作需要四個輸入值，該四個輸入值中之一個值具有破壞性。如此不符合通用 32 個三元 RISC 格式。

排程更新指令可由 ARM 高階 SIMD 架構提供。

為了避免記憶體負載及記憶體儲存，吾人選擇實施有效執行在 FIPS 180-4 中描述之循環佇列形式之排程更新的指令。

為了完整起見，吾人包括用於排程更新之向量化方法。

SHA-1 排程更新向量化及替代

如此遵循 w[round]、w[round+1$_{mod16}$] 及 w[round+2$_{mod16}$] 可以平行處理之監視。在計算 w[round+3$_{mod16}$]時具有對 w[round]之相依性，該相依性防止直接路由到四向向量化。

w[round]=ROR(w[round+13]^w[round+8]^w[round+2]^w[round]),31);

w[round+1]=ROR(w[round+14]^w[round+9]^w[round+3]^w[round+1]),31);

w[round+2]=ROR(w[round+15]^w[round+10]^w[round+4]^w[round+2]),31);

w[round+3]=ROR(w[round]^w[round+11]^w[round+5]^w[round+3]),31);

該限制可藉由在 w[round+3_{mod16}]之計算中將零替代為值 w[round]，且藉由用附加邏輯異或及旋轉步驟修復該結果來克服；該情況說明如下。

w[round]=ROR(w[round+13]^w[round+8]^w[round+2]^w[round]),31);

w[round+1]=ROR(w[round+14]^w[round+9]^w[round+3]^w[round+1]),31);

w[round+2]=ROR(w[round+15]^w[round+10]^w[round+4]^w[round+2]),31);

w[round+3]=ROR(0^w[round+11]^w[round+5]^w[round+3]),31);

w[round+3]=w[round+3]^ROR(w[round],31);

可將上述程式碼區塊重新因素化以對具有 4x32 位元之資料路徑大小的 SIMD 架構利用 4 通道向量操作。

SHA-1 排程更新及散列更新平衡

為了使排程更新操作與散列更新操作平衡，可如先

前所述處理排程更新，亦即，使用四向向量化處理排程更新。如此允許單個排程更新為後續散列函數指令產生足夠的 4x32 位元字之資料。

在合理的 SIMD 實施中，與用以執行建議的 SHA-1 散列函數之彼等技術相比，向量化技術將進行更多執行循環以計算排程資料。

對此存在數個原因：

含有元素{round+2,round+3,round+4,round+5}之向量可能將橫跨兩個向量暫存器。

含有元素{round+13,round+14,round+15,0}之向量將需要自一個向量暫存器及零向量中擷取。

SIMD 向量旋轉在例如 ARM 高階 SIMD 之 SIMD 指令集中並不常見。因此向量旋轉需要兩個向量位移及 or 指令。

歸因於 Amdahl 定律，應平衡 SHA-1 演算法之兩個部分，否則較慢部分將限制可達成之加速量。

該監視產生用於加速 SHA-1 排程更新函數之以下 SIMD 指令。

SHA1SU0 Vd.4S, Vn.4S, Vm.4S

T<127:0>=Vn<63:0>:Vd<127:64>

Vd=T XOR Vd XOR Vm

SHA1SU1 Vd.4S, Vn.4S

T<127:0>=Vd XOR {32{0}:Vn<127:32>};

Vd<95:0>=T<94:64>:T<95>:T<62:32>:T<63>:T<30:0>:T<31>;

Vd<127:96>=(T<126:96>:T<127>)XOR(T<29:0>:T<31:30>);

指令假設循環佇列常駐於四個 4x32 位元向量暫存器中。

在指令內部牽引元素之重新排序。如此有效地使元素之重新排序自由，該等元素正好為微架構中之線路。

固定旋轉亦僅為線。

在大部分微架構中，平衡指令且指令可具有極低的循環潛時；該等指令包含串列及佈線中之兩個邏輯異或。

因此，根據示例性實施例，該單一指令多重資料處理電路經設置以由第一排程指令控制，該第一排程指令具有第一輸入運算元 Sp[127:0]及第二輸入運算元 Sq[127:0]且產生具有一值之輸出運算元 Sr[127:0]，該值與由以下步驟給出之值相同：

T[127:0]={Sp[63:0]:Sq[127:64]}及

Sr[127:0]=T[127:0] XOR Sr[127:0] XOR Sq[127:0].

將 SHA-2 演算法作為目標之指令

在針對 SHA-1 演算法提出之指令之論述中所概括的許多特徵可同樣地適用於 SHA-2 演算法。本節將描述針對 SHA-2 演算法提出之指令中之差異。

SHA-2 散列更新指令

由於針對 SHA-1 概括之原因，SHA-2 散列更新函數被由兩個散列更新指令作為目標。

SHA-2 演算法之工作散列摘要為 256 位元或者 512 位元。以下聚焦於演算法 SHA-256 及 SHA-224，該演算法 SHA-256 及 SHA-224 具有 256 位元之工作散列，因為該等演算法包含於本發明之初始實現中。在稍後的章節中論述本發明技術如何應用於 SHA-512、SHA-384、SHA-512/256 及 SHA-512/224。

SHA-256 散列更新指令

SHA-256（及 SHA-224）之工作散列摘要為 256 位元長。在具有暫存器寬度為 128 位元之 SIMD 架構中，對散列摘要之任何操作之結果需要兩個指令；一個指令返回前 4x32 位元字，且第二指令返回剩餘 4x32 位元字。

不同於 SHA-1，SHA-2 散列更新函數是固定的且在

給定數目之迭代之後不變化，因此，吾人僅需要兩個指令。

```
SHA256H Qd, Qn, Vm.4S
    X=Qd;
    Y=Qn;
    for(i=0 to 3)
    {
     index=(i*32);
  tCh<31:0>=Choose(Y<31:0>,Y<63:32>,Y<95:64>)
;
  tMaj<31:0>=Majority(X<31:0>,X<63:32>,X<95:64
>);
  t1<31:0>=Y<127:96>+Sigma1(Y<31:0>)
  +tCh<31:0>+Vm<(index+31):index>;
  X<127:96>=t1<31:0>+X<127:96>;
  Y<127:96>=t1<31:0>+Sigma0(X<31:0>)+tMaj<31:
0>;
  t2<31:0>=Y<127:96>;
  Y<127:0>=Y<95:0>:X<127:96>;
  X<127:0>=X<95:0>:t2<31:0>;
    }
```

```
Qd=X;
```

相應地，根據示例性實施例，單一指令多重資料處理電路經設置以由進一步程式指令控制，該進一步程式指令具有自該單一指令多重資料暫存器檔案讀取之第一輸入運算元 Qd[127:0]及第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

```
X[127:0]=Qd[127:0];
Y[127:0]=Qn[127:0];
for(I=0 to (2^N-1));
 {
    Index=(I*32);
 TCh[31:0]=Choose(Y[31:0],Y[63:32],Y[95:64]);
 TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);
 T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]
+Vm[Index+31:Index];
    X[127:96]=T1[31:0]+X[127:96];
 Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]
    T2[31:0]=Y[127:96];
```

```
        Y[127:0]={Y[95:0]:X[127:96]};
        X[127:0]={X[95:0]:T2[31:0]}
    }
  Qd_output[127:0]=X[127:0];
```

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

```
  SHA256H2    Qd, Qn, Vm.4S
     X=Qn;
     Y=Qd;
     for(i=0 to 3)
     {
      index=(i*32);
  tCh<31:0>=Choose(Y<31:0>,Y<63:32>,Y<95:64>);
  tMaj<31:0>=Majority(X<31:0>,X<63:32>,X<95:64>
);
   t1<31:0>=Y<127:96>+Sigma1(Y<31:0>)
  +tCh<31:0>+Vm<(index+31):index>;
```

X<127:96>=t1<31:0>+X<127:96>;

Y<127:96>=t1<31:0>+Sigma0(X<31:0>)+tMaj<31:0>;

t2<31:0>=Y<127:96>;

Y<127:0>=Y<95:0>:X<127:96>;

X<127:0>=X<95:0>:t2<31:0>;

}

Qd=Y;

相應地，根據示例性實施例，單一指令多重資料處理電路經設置以由進一步程式指令控制，該進一步程式指令具有自該單一指令多重資料暫存器檔案讀取之第一輸入運算元 Qd[127:0]及第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之值：

X[127:0]=Qn[127:0];

Y[127:0]=Qd[127:0];

for(I=0 to (2^N-1));

{

Index=(I*32);

```
TCh[31:0]=Choose(Y[31:0],Y[63:32],Y[95:64]);
TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);
T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]
+Vm[Index+31:Index];
    X[127:96]=T1[31:0]+X[127:96];
Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]
    T2[31:0]=Y[127:96];
    Y[127:0]={Y[95:0]:X[127:96]};
    X[127:0]={X[95:0]:T2[31:0]}
}
```

Qd_{output}[127:0]=Y[127:0];

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

SHA256H 預期散列摘要之前 4x32 位元字在 Qd 中，預期剩餘 4x32 位元字在 Qn 中且 SHA256H 預期排程資料之 4x32 位元字在 Vm.4S 中。

SHA256H2 預期散列摘要之第二 4x32 位元字在 Qd

中，預期前 4x32 位元字在 Qn 中且 SHA256H2 預期排程資料之 4x32 位元字在 Vm.4S 中。

請注意，因為 SHA256H 破壞散列摘要之前 4x32 位元字，所以在執行 SHA256H 之前必須進行複製以便正確值可經傳遞至 Qn 中之 SHA256H2。

SHA-256 散列更新指令變體

如先前對於 SHA-1 散列更新指令所概述，用於較寬向量 SIMD 之 SHA-256 指令之變體可包括以下各者：

SHA256(H|H2)Qd, Qn, Vm.8S

SHA256(H|H2)Qd, Qn, Vm.16S

該等指令將分別處理散列更新函數之 8 次及 16 次迭代。較寬的 SIMD 資料路徑亦可允許：

SHA256HOd, Vm.8S

SHA256HOd, Vm.16S

在 Od 為 256 位元寬暫存器之情況下，不必提供 SHA256H2 操作，整個散列摘要將適合在向量暫存器中。

SHA-256 排程更新

如先前所概述，對於 SHA-1，實現來自 SHA-256 演算法之加速需要在散列更新函數與排程更新函數之間的平衡。

SHA-256 排程更新函數將來自資料排程之四個 32 位元字結合為單一所得字，該單一所得字擴展排程，或在循環佇列之情況下，該單一所得字將排程中之字重寫。

排程更新操作由邏輯異或、固定移位及固定旋轉（已知）組成。

uint32 schedule_update(int round, uint32*w){

return

w[round]+sigma1(w[round-2])+w[round-7]+sigma0(w[round-15]);

}

其中：

sigma0(x)=ror(x,7)^ror(x,18)^shr(x,3);

sigma1(x)=ror(x,17)^ror(x,19)^shr(x,10);

此亦可在循環佇列中表達（已知）：

void schedule_update(int round, uint32*w){

w[round]=sigma1(w[round+14$_{mod16}$])+w[round+9$_{mod16}$]+sigma0(w[round+1$_{mod16}$]);

}

SHA-256 排程更新向量化及替代

SHA-256 排程更新函數亦可以適合於 4 向 SIMD 之

方式被向量化。

w[round]=sigma1(w[round+14])+w[round+9]+sigma0(w[round+1]);

w[round+1]=sigma1(w[round+15])+w[round+10]+sigma0(w[round+2]);

w[round+2]=sigma1(w[round])+w[round+11]+sigma0(w[round+3]);

w[round+3]=sigma1(w[round+1])+w[round+12]+sigma0(w[round+4]);

請注意，存在兩個相依性，亦即 w[round]及w[round+1]。用於 SHA-256 之替代法藉由代入零值且然後決定結果而像以前一樣起作用。該方法說明如下：

w[round]=sigma1(w[round+14])+w[round+9]+sigma0(w[round+1]);

w[round+1]=sigma1(w[round+15])+w[round+10]+sigma0(w[round+2]);

w[round+2]=sigma1(w[0])+w[round+11]+sigma0(w[round+3]);

w[round+3]=sigma1(w[0])+w[round+12]+sigma0(w[round+4]);

w[round+2]+=sigma1(w[round]);

w[round+3]+=sigma1(w[round]);

可將上述程式碼區塊重新因素化以對具有 4x32 位元之資料路徑大小的 SIMD 架構利用 4 通道向量操作。

SHA-256 排程更新及散列更新平衡

為了使排程更新操作與散列更新操作平衡，吾人提出如先前所述處理排程更新，亦即，使用四向向量化處理排程更新。如此允許單個排程更新為後續散列函數指令產生足夠的 4x32 位元字之資料。

在合理的 SIMD 實施中，與用以執行建議的 SHA-1 散列函數之彼等技術相比，向量化技術將進行更多執行循環以計算排程資料。

對此存在數個原因：

含有元素{round+1, round+2, round+3, round+4}及{round+9, round+10, round+11, round+12}之向量將跨越多於一個向量暫存器。

含有{round+14, round+15, 0, 0}之暫存器將需要使用擷取組成。

sigma 操作含有旋轉，且 SIMD 向量旋轉在例如 ARM 高階 SIMD 之 SIMD 指令集中不常見。在該等架

構中之向量旋轉需要兩個向量位移及 OR 指令。

考慮替代之決定亦將需要擷取暫存器。

sigma0 及 sigma1 操作由大約 7 個向量操作組成。

歸因於 Amdahl 定律，需要平衡 SHA-256 演算法之兩部分以免較慢部分限制可達成之加速量。

該等監視產生用於加速 SHA-256 排程更新函數之以下 SIMD 指令。

SHA256SU0 Vd.4S, Vn.4S

T<127:0>=Vn<31:0>:Vd<127:32>

T<127:0>=VecROR32(T, 7) XOR VecROR32(T, 18) XOR VecSHR32(T, 3)

Vd=VecADD32(T,Vd)

SHA256SU1 Vd.4S, Vn.4S, Vm.4S

T0<127:0>=Vm<31:0>:Vn<127:32>

T1<63:0>=Vm<127:64>

T1<63:0>=VecROR32(T1<63:0>,17)XOR VecROR32(T1<63:0>,19) XOR VecSHR32(T1<63:0>,10)

T3<63:0>=VecADD32(Vd<63:0>,T0<63:0>)

T1<63:0>=VecADD32(T3<63:0>,T1<63:0>)

T2<63:0>=VecROR32(T1<63:0>,17)XOR

VecROR32(T1<63:0>,19) XOR VecSHR32(T1<63:0>,10)

T3<63:0>=VecADD32(Vd<127:64>,T0<127:64>)

Vd=VecADD32(T3<63:0>,T2<63:0>):T1<63:0>

指令假設循環佇列常駐於四個 4x32 位元向量暫存器中。指令不排除使用排程擴展。

在指令內部牽引元素之重新排序及擷取。然後，微架構能夠選擇以將該等及固定移位與旋轉實施為接線。

在大部分微架構中，指令可具有低循環潛時。

因此，根據示例性實施例，單一指令多重資料處理電路經設置以由第一排程指令控制，該第一排程指令具有輸入運算元 Sp[127:0]且產生具有一值之輸出運算元 Sq[127:0]，該值與由以下步驟給出之值相同：

T[127:0]={Sp[31:0]:Sq[127:32]};

T[127:0]=VecROR32(T[127:0],7)XOR VecROR32(T[127:0], 18) XOR VecROR32(T[127:0], 3);

及

Sq[127:0]=VecADD32(T[127:0], Sq[127:0]),

其中 VecROR32 (A, B)為在 A 之內的每一 32 位元字單獨右旋轉 B 位元位置，且 VecADD32 (A, B)為在 A

之內的每一32位元字單獨增加至在B之內的相應32位元字。

因此，根據示例性實施例，單一指令多重資料處理電路經設置以由第二排程指令控制，該第二排程指令具有第一輸入運算元 Sp[127:0]及第二輸入運算元 Sq[127:0]且產生具有一值之輸出運算元 Sr[127:0]，該值與由以下步驟給出之值相同：

T0[127:0]={Sq[31:0]:Sp[127:32]};

T1[63:0]=Sq[127:64];

T1[63:0]=VecROR32(T1[63:0],17)XOR VecROR32(T1[63:0], 19) XOR VecROR32(T1[63:0], 10);

T3[63:0]=VecADD32(Sr[63:0], T0[63:0]);

T1[63:0]=VecADD32(T3[63:0],T1[63:0]);

T2[63:0]=VecROR32(T1[63:0],17)XOR VecROR32(T1[63:0], 19) XOR VecROR32(T1[63:0], 10);

T3[63:0]=VecADD32(Sr[127:64], T0[127:64]);及

Sr[127:0]={VecADD32(T3[63:0], T2[63:0]):T1[63:0]},

其中 VecROR32 (A, B)為在A之內的每一32位元字

單獨右旋轉 B 位元位置，且 VecADD32 (A, B)為在 A 之內的每一 32 位元字單獨增加至 B 之內的相應 32 位元字。

在 **SHA-256** 與 **SHA-512** 之間的差異

SHA-512 演算法非常類似於 SHA-256 演算法。在描述對 SHA-256 之支援的章節中概述之相同方法可同樣地應用於 SHA-512，僅存在以下小的不同：

輸入資料分成 128 位元組之區塊，且以雙模二元資料格式處理為 16x64 位元字。

SHA-512 對 8x64 位元字有效

SHA-512 需要散列函數之 80 次迭代。

散列函數及排程更新對 64 位元字有效，且散列函數及排程更新含有不同固定移位、旋轉，及邏輯異或。

為了簡便起見，吾人忽略 SHA-512 演算法。

將 **SHA-512** 、 **SHA-384** 、 **SHA-512/256** 及 **SHA-512/256** 演算法作為目標之指令。

將 SHA-256 作為目標之指令的目的與對於 SHA-512 演算法之目的相同。吾人列出將 SHA-512 作為目標之該等指令的可能實現。

在 SIMD 暫存器為 128 位元，且假定每散列 4 次迭代及排程指令之情況下：

SHA512H{Qd, Qd+1}, {Qn, Qn+1}, {Vm.2D, Vm+1.2D}

SHA512H2{Qd, Qd+1}, {Qn, Qn+1}, {Vm.2D, Vm+1.2D}

SHA512SU0{Vd.2D, Vd+1.2D}, {Vn.2D, Vn+1.2D}

SHA512SU1{Vd.2D, Vd+1.2D}, {Vn.2D, Vn+1.2D}, {Vm.2D, Vm+1.2D}

請注意，上述指令有可能需要暫存器鎖定(register pinning)；在微架構之內指定一暫存器且暗示第二暫存器。指令將不再屬於典型 RISC 三元格式，然而，該等類型操作存在優先，例如，在載入/儲存 ARM 有限公司之指令的 Neon 中。

在較廣泛 SIMD 暫存器可用之情況下，指令之可能的變體包括：

SHA512H Od, On, Vm.4D

SHA512H2 Od, On, Vm.4D

SHA512SU0 Vd.4D, Vn.4D

SHA512SU1 Vd.4D, Vn.4D, Vm.4D

該等指令之變體亦處理用於散列之迭代及排程更新操作，但是歸因於較廣泛 SIMD 暫存器符合三元 RISC 格式。

使用如 FIPS 180-4 中所述之截斷，該等指令可同樣地以 SHA-384、SHA-512/256 及 SHA-512/224 為目標。

儘管已在本文中參看隨附圖式詳細地描述本發明之說明性實施例，但熟習此項技術者應理解，本發明不限於彼等精確實施例，且在不脫離由附加申請專利範圍界定之本發明之範疇及精神的情況下，可在其中實現各種變化及修改。

【圖式簡單說明】

第 1 圖示意圖示包括單一指令多重資料暫存器檔案及單一指令多重資料處理電路之資料處理設備，該單一指令多重資料處理電路包括對執行密碼學處理指令之支援；

第 2 圖示意圖示在散列演算法之一示例性形式之內的資料流；及

第 3 圖示意圖示進一步處理指令如何不遵循與單一指令多重資料處理電路相關聯之一般基於通道之處理。

【主要元件符號說明】

2	資料處理設備	4	中央處理單元
	記憶體	8	通用暫存器檔案
10	通用處理電路	12	單一指令多重資料暫存器檔案
14	中央處理單元	16	通用暫存器重新命

名及計數電路

單一指令多重資料

18 暫存器重新命名及計數電路

20 指令管線

22 指令解碼器

24 中斷電路

26 檔案

28 區塊

30 散列演算法

32 散列種子值

34 散列更新

36 排程更新

38 中間散列值

40 輸出散列值

42 向量資料值

44 第一部分

104年9月3日修正替換頁(本)

七、申請專利範圍：

1. 一種資料處理設備，該資料處理設備包含：

一單一指令多重資料暫存器檔案；及

單一指令多重資料處理電路，該單一指令多重資料處理電路經耦接至該單一指令多重資料暫存器檔案且經設置以由一單一指令多重資料程式指令控制，以獨立地對儲存在單獨通道之內的單獨資料元素執行一處理操作，該等單獨通道在該單一指令多重資料暫存器檔案之一輸入運算元暫存器之內；其中

該單一指令多重資料處理電路經設置以由一進一步的程式指令控制，以對一向量資料值執行一進一步的處理操作，該向量資料值包含保持在該單一指令多重資料暫存器檔案之一輸入運算元暫存器之內的一資料元素序列，以產生儲存在該單一指令多重資料暫存器檔案之一輸出運算元暫存器之內的一輸出運算元，該輸出運算元具有含有一值的一第一部分，該值取決於在該資料元素序列之內的所有資料元素，其中該進一步的程式指令為一密碼學程式指令，該密碼學程式指令經操作以依賴於形成該向量資料值之資料之複數個字而產生作為該輸出運算元之一輸出散列值。

2. 如請求項 1 所述之資料處理設備，其中該進一步的程式指令執行一迭代處理操作，該迭代處理操作消耗連續之資料之字及至少部分的中間散列值以產生該輸出散列值。

3. 如請求項 1 所述之資料處理設備，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之一值，該等以下步驟為：

```
X[127:0]=Qd[127:0];
Y[31:0]=Sn[31:0];
for(I=0 to (2^N-1));
   {
   Index=(I*32);
   t1[31:0]=OP FUNC (X[63:32], X[95:64], X[127:96]);
   Y[31:0]=Y[31:0]+ROL(X[31:1],5)+T1[31:0]+Vm[Inde
   x+31:Index];
```

X[63:32]=ROL(X[63:32],30);

T2[31:0]=Y[31:0];

Y[31:0]=X[127:96];

X[127:0]={X[95:0]:T2[31:0]}

Qd_{output}[127:0]=X[127:0];

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

4. 如請求項 1 所述之資料處理設備，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之一值，該等以下步驟為：

X[127:0]=Qd[127:0];

```
Y[31:0]=Sn[31:0];
for(I=0 to(2^N-1));
   {
   Index=(I*32);
   T1[31:0]=OP_FUNC(X[63:32],X[95:64],X[127:96]);
   Y=Y+ROL(X[31:0],5)+T1[31:0]+Vm[(Index+31):Inde
   x];
   X[63:32]=ROL(X[63:32],30);
   T2[31:0]=Y;
   Y=X[127:96];
   X[127:0]={X[95:0]:T2[31:0]};
   }
Qd_output[127:0]={0:Y[31:0]};
```

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

5. 如請求項 3 所述之資料處理設備，其中該進一步的程式指令包括一欄位，該欄位選擇以下各者中之一者作

為 OP FUNC (B, C, D)，該以下各者為：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D)。

6. 如請求項 3 所述之資料處理設備，其中該第一輸入運算元 Qd[127:0]及該第二輸入運算元 Sn[31:0]係讀取自於該單一指令多重資料暫存器檔案之內的單獨暫存器。

7. 如請求項 3 所述之資料處理設備，其中該第一輸入運算元 Qd[127:0]及該第二輸入運算元 Sn[31:0]係讀取自於該單一指令多重資料暫存器檔案之內的一共享暫存器。

8. 如請求項 1 所述之資料處理設備，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元

Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之一值，該等以下步驟為：

```
X[127:0]=Qd[127:0];
Y[127:0]=Qn[127:0];
for(I=0 to(2^N-1));
   {
   Index=(I*32);
   TCh[31:0]=Choose(Y[31:0], Y[63:32], Y[95:64]);
   TMaj[31:0]=Majority(X[31:0], Y[63:32], Y[95:64]);
   T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]+Vm
   [Index+31:Index];
   X[127:96]=T1[31:0]+X[127:96];
   Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]
   T2[31:0]=Y[127:96];
   Y[127:0]={Y[95:0]:X[127:96]};
   X[127:0]={X[95:0]:T2[31:0]}
   }
```

Qd_{output}[127:0]=X[127:0];

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR

ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

9. 如請求項 1 所述之資料處理設備，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟給出之值相同之一值，該等以下步驟為：

```
X[127:0]=Qn[127:0];
Y[127:0]=Qd[127:0];
for(I=0 to (2^N-1));
   {
   Index=(I*32);
   TCh[31:0]=Choose(Y[31:0],Y[63:32],Y[95:64]);
   TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);
   T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]+Vm
   [Index+31:Index];
```

X[127:96]=T1[31:0]+X[127:96];

Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]

T2[31:0]=Y[127:96];

Y[127:0]={Y[95:0]:X[127:96]};

X[127:0]={X[95:0]:T2[31:0]}

}

Qd_{output}[127:0]=Y[127:0];

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

10.如請求項 8 所述之資料處理設備，其中該第一輸入運算元 Qd[127:0]及該第二輸入運算元 Qn[127:0]係讀取自於該單一指令多重資料暫存器檔案之內的單獨暫存器。

11.如請求項 8 所述之資料處理設備，其中該第一輸入運算元 Qd[127:0]及該第二輸入運算元 Qn[127:0]係讀取

自於該單一指令多重資料暫存器檔案之內的一共享暫存器。

12. 如請求項 1 所述之資料處理設備，其中該單一指令多重資料處理電路利用用於管理該進一步的程式指令及該單一指令多重資料程式指令之處理之共用機制。

13.如請求項 12 所述之資料處理設備，其中該管理處理之步驟包括如下列之管理步驟之一或更多者：

暫存器重新命名；

指令排程；

指令發出；

指令報廢；及

指令中斷。

14.如請求項 3 所述之資料處理設備，其中該單一指令多重資料處理電路經設置以由一旋轉指令控制，該旋轉指令具有一輸入運算元 Sm[31:0]且產生具有一值之一輸出運算元 Sd[31:0]，該值與 Sm[31:0]右旋轉兩個位元位置所給出的一值相同。

15.如請求項3所述之資料處理設備，其中該單一指令多重資料處理電路經設置以由一第一排程更新指令控制，該第一排程更新指令具有一第一輸入運算元Sp[127:0]及一第二輸入運算元 Sq[127:0]且產生具有一值之一輸出運算元 Sr[127:0]，該值與經由以下步驟所給出的一值相同，該等以下步驟為：

T[127:0]={Sp[63:0]:Sq[127:64]}及

Sr[127:0]=T[127:0]XOR Sr[127:0]XOR Sq[127:0]。

16.如請求項13所述之資料處理設備，其中該單一指令多重資料處理電路經設置以由一第二排程更新指令控制，該第二排程更新指令具有一輸入運算元Ss[127:0]且產生具有一值之一輸出運算元St[127:0]，該值與經由以下步驟所給出的一值相同，該等以下步驟為：

T[127:0]=St[127:0]XOR{32{0}:Ss[127:32]};

St[95:0]={T[94:64]:T[95]:T[62:32]:T[63]:T[30:0]:T[31]};及

St[127:96]=({T[126:96]:T[127]})XOR

({T[29:0]:T[31:30]})。

17.如請求項 8 所述之資料處理設備，其中該單一指令多重資料處理電路經設置以由一第一排程更新指令控制，該第一排程更新指令具有一輸入運算元 Sp[127:0]且產生具有一值之一輸出運算元 Sq[127:0]，該值與經由以下步驟所給出的一值相同，該等以下步驟為：

T[127:0]={Sp[31:0]:Sq[127:32]};

T[127:0]=VecROR32(T[127:0],7)XOR VecROR32(T[127:0],18) XOR VecROR32(T[127:0], 3);

及

Sq[127:0]=VecADD32(T[127:0], Sq[127:0]),

其中 VecROR32 (A, B)為在 A 之內的每一 32 位元字單獨右旋轉 B 位元位置，且 VecADD32 (A, B)為在 A 之內的每一 32 位元字單獨增加至在 B 之內的一相應 32 位元字。

18.如請求項 17 所述之資料處理設備，其中該單一指令多重資料處理電路經設置以由一第二排程更新指令控制，該第二排程更新指令具有一第一輸入運算元 Sp[127:0]及一第二輸入運算元 Sq[127:0]且產生具有一值之一輸出運算元 Sr[127:0]，該值與經由以下步驟

所給出的一值相同，該等以下步驟為：

T0[127:0]={Sq[31:0]:Sp[127:32]};

T1[63:0]=Sq[127:64];

T1[63:0]=VecROR32(T1[63:0],17)XOR VecROR32(T1[63:0],19) XOR VecROR32(T1[63:0],10);

T3[63:0]=VecADD32(Sr[63:0], T0[63:0]);

T1[63:0]=VecADD32(T3[63:0], T1[63:0]);

T2[63:0]=VecROR32(T1[63:0],17)XOR VecROR32(T1[63:0], 19) XOR VecROR32(T1[63:0],10);

T3[63:0]=VecADD32(Sr[127:64], T0[127:64]);及

Sr[127:0]={VecADD32(T3[63:0], T2[63:0]):T1[63:0]},

其中 VecROR32 (A, B)為在 A 之內的每一 32 位元字單獨右旋轉 B 位元位置，且 VecADD32 (A, B)為在 A 之內的每一 32 位元字單獨增加至在 B 之內的一相應 32 位元字。

19.如請求項 1 所述之資料處理設備，該資料處理設備進一步包含與該單一指令多重資料暫存器檔案分離之一通用暫存器檔案，該通用暫存器檔案具有通用暫存

器，該等通用暫存器具有比在該單一指令多重資料暫存器檔案之內的暫存器之位元寬度低之位元寬度，且通用處理電路經耦接至該通用暫存器檔案，且該通用處理電路經設置以由一通用處理指令控制以對儲存在該等通用暫存器中之一者之內的一輸入運算元執行一處理操作。

20.一種資料處理設備，該資料處理設備包含：

單一指令多重資料暫存器檔案構件，用於儲存單一指令多重資料運算元；及

單一指令多重資料處理構件，用於在一單一指令多重資料程式指令之控制下執行一處理操作，該單一指令多重資料處理構件經耦接至該單一指令多重資料暫存器檔案構件，且對儲存在該單一指令多重資料暫存器檔案構件之一輸入運算元暫存器之內的單獨通道內之單獨資料元素獨立地執行該處理操作；其中

該單一指令多重資料處理構件由一進一步的程式指令控制，以對一向量資料值執行一進一步的處理操作，該向量資料值包含保持在該單一指令多重資料暫存器檔案構件之一輸入運算元暫存器之內的一資料元素序列，以產生儲存在該單一指令多重資料暫存器檔案

構件之一輸出運算元暫存器之內的一輸出運算元，該輸出運算元具有含有一值的一第一部分，該值取決於在該資料元素序列之內的所有資料元素，其中該進一步的程式指令為一密碼學程式指令，該密碼學程式指令經操作以依賴於形成該向量資料值之資料之複數個字而產生作為該輸出運算元之一輸出散列值。

21.一種處理資料之方法，該方法包含以下步驟：

將單一指令多重資料運算元儲存在一單一指令多重資料暫存器檔案之內；

在一單一指令多重資料程式指令之控制下，獨立地對儲存在該單一指令多重資料暫存器檔案之一輸入運算元暫存器之內的單獨通道內的單獨資料元素執行一處理操作；及

在一進一步的程式指令之控制下，對一向量資料值執行一進一步的處理操作，該向量資料值包含保持在該單一指令多重資料暫存器檔案之一輸入運算元暫存器之內的一資料元素序列，以產生儲存在該單一指令多重資料暫存器檔案之一輸出運算元暫存器之內的一輸出運算元，該輸出運算元具有含有一值的一第一部分，該值取決於在該資料元素序列之內的所有資料元

素，其中該進一步的程式指令為一密碼學程式指令，該密碼學程式指令經操作以依賴於用於該向量資料值之資料之複數個字而產生作為該輸出運算元之一輸出散列值。

22.如請求項 21 所述之方法，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟所給出之值相同之一值，該等以下步驟為：

```
X[127:0]=Qd[127:0];
Y[31:0]=Sn[31:0];
for(I=0 to (2^N-1));
   {
   Index=(I*32);
   t1[31:0]=OP FUNC (X[63:32], X[95:64], X[127:96]);
   Y[31:0]=Y[31:0]+ROL(X[31:1],5)+T1[31:0]+Vm[Inde
   x+31:Index];
```

X[63:32]=ROL(X[63:32],30);

T2[31:0]=Y[31:0];

Y[31:0]=X[127:96];

X[127:0]={X[95:0]:T2[31:0]}

}

Qd_{output}[127:0]=X[127:0];

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

23.如請求項 21 所述之方法，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Sn[31:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟所給出之值相同之一值，該等以下步驟為：

X[127:0]=Qd[127:0];

```
Y[31:0]=Sn[31:0];
for(I=0 to (2^N-1));
   {
   Index=(I*32);
   T1[31:0]=OP_FUNC(X[63:32],X[95:64],X[127:96]);
   Y=Y+ROL(X[31:0],5)+T1[31:0]+Vm[(Index+31):Inde
   x];
   X[63:32]=ROL(X[63:32],30);
   T2[31:0]=Y;
   Y=X[127:96];
   X[127:0]={X[95:0]:T2[31:0]};
   }
Qd_output[127:0]={0:Y[31:0]};
```

其中 OP FUNC (B, C, D)為以下各者中之一者：

(((C XOR D) AND B) XOR D);

(B XOR C XOR D);及

(B AND C) OR ((B OR C) AND D);及

ROL (P, Q)為值 P 左旋轉 Q 位元位置。

24.如請求項 21 所述之方法，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第

一輸入運算元 Qd[127:0]及一第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元 Qd_{output}[127:0]，以具有與由以下步驟所給出之值相同之一值，該等以下步驟為：

```
X[127:0]=Qd[127:0];
Y[127:0]=Qn[127:0];
for(I=0 to (2^N-1));
   {
   Index=(I*32);
   TCh[31:0]=Choose(Y[31:0],Y[63:32],Y[95:64]);
   TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);
   T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]+Vm
   [Index+31:Index];
    X[127:96]=T1[31:0]+X[127:96];
    Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]
    T2[31:0]=Y[127:96];
    Y[127:0]={Y[95:0]:X[127:96]};
    X[127:0]={X[95:0]:T2[31:0]}
    }
```

$Qd_{output}[127:0]=X[127:0];$

其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

25.如請求項 21 所述之方法，其中該進一步的程式指令具有自該單一指令多重資料暫存器檔案讀取之一第一輸入運算元 Qd[127:0]及一第二輸入運算元 Qn[127:0]兩者，且該向量資料值包含其中 Index 為 0 至 2^N 之 Vm[Index+31:Index]，其中 N 為一正整數，該進一步的處理操作產生該輸出運算元 $Qd_{output}[127:0]$，以具有與由以下步驟給出之值相同之一值，該等以下步驟為：

X[127:0]=Qn[127:0];

Y[127:0]=Qd[127:0];

for(I=0 to (2^N-1));

{

Index=(I*32);

TCh[31:0]=Choose(Y[31:0],Y[63:32],Y[95:64]);

TMaj[31:0]=Majority(X[31:0],Y[63:32],Y[95:64]);

T1[31:0]=Y[127:96]+Sigma1(Y[31:0])+TCh[31:0]+Vm

[Index+31:Index];

X[127:96]=T1[31:0]+X[127:96];

Y[127:96]=T1[31:0]+Sigma0(X[31:0])+TMaj[31:0]

T2[31:0]=Y[127:96];

Y[127:0]={Y[95:0]:X[127:96]};

X[127:0]={X[95:0]:T2[31:0]}

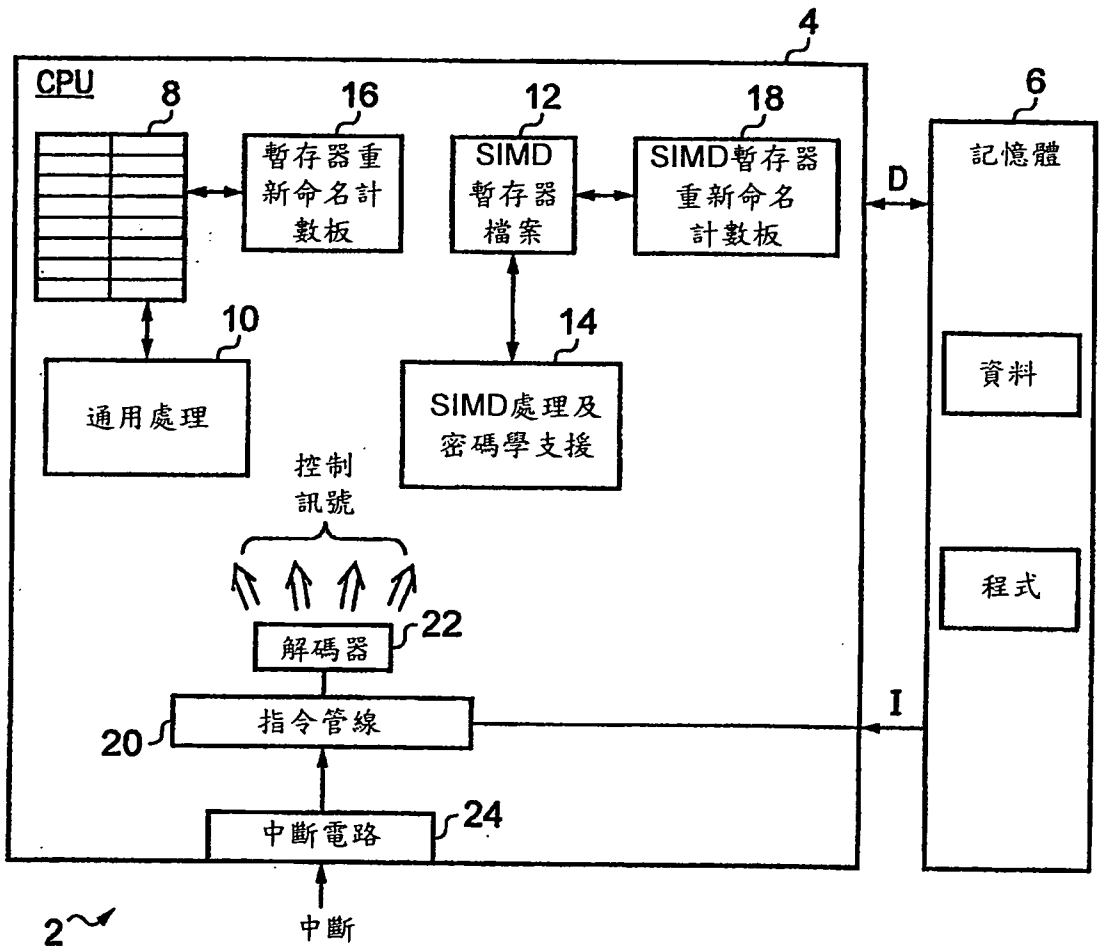
}

Qd_{output}[127:0]=Y[127:0];

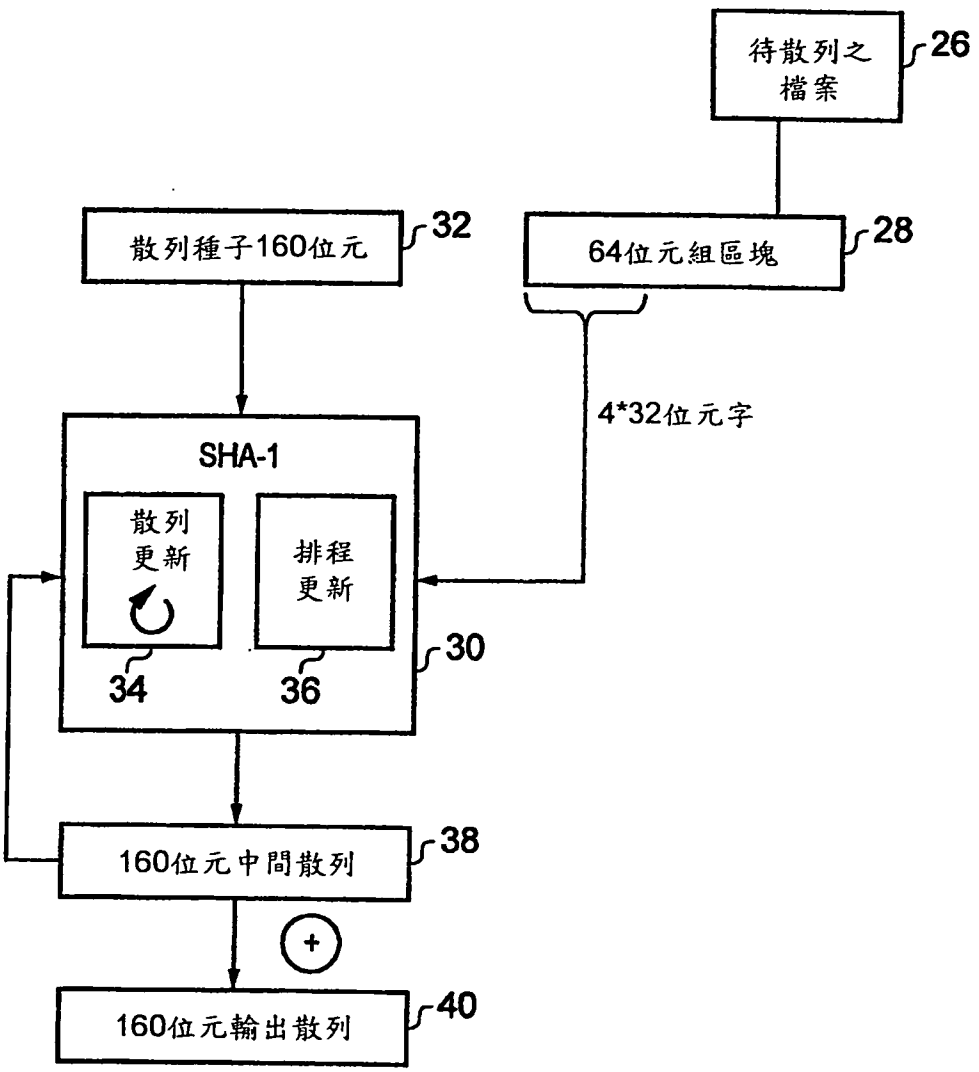
其中 Choose (B, C, D)為(((C XOR D) AND B) XOR D)，Majority (B, C, D)為((B AND C) OR ((B OR C) AND D))，Sigma0(B)為(ROR(B,2) XOR ROR(B, 13) XOR ROR(B, 22))，Sigma1(B)為(ROR(B,6) XOR ROR(B, 11) XOR ROR (B, 25))且 ROR (P, Q)為值 P 右旋轉 Q 位元位置。

26.一種儲存於一非暫態電腦可讀取儲存媒體上之電腦程式，當該電腦程式經實施於一電腦上時，該電腦程式提供對應於如請求項 1 所述之資料處理設備之一虛

擬機器執行環境。

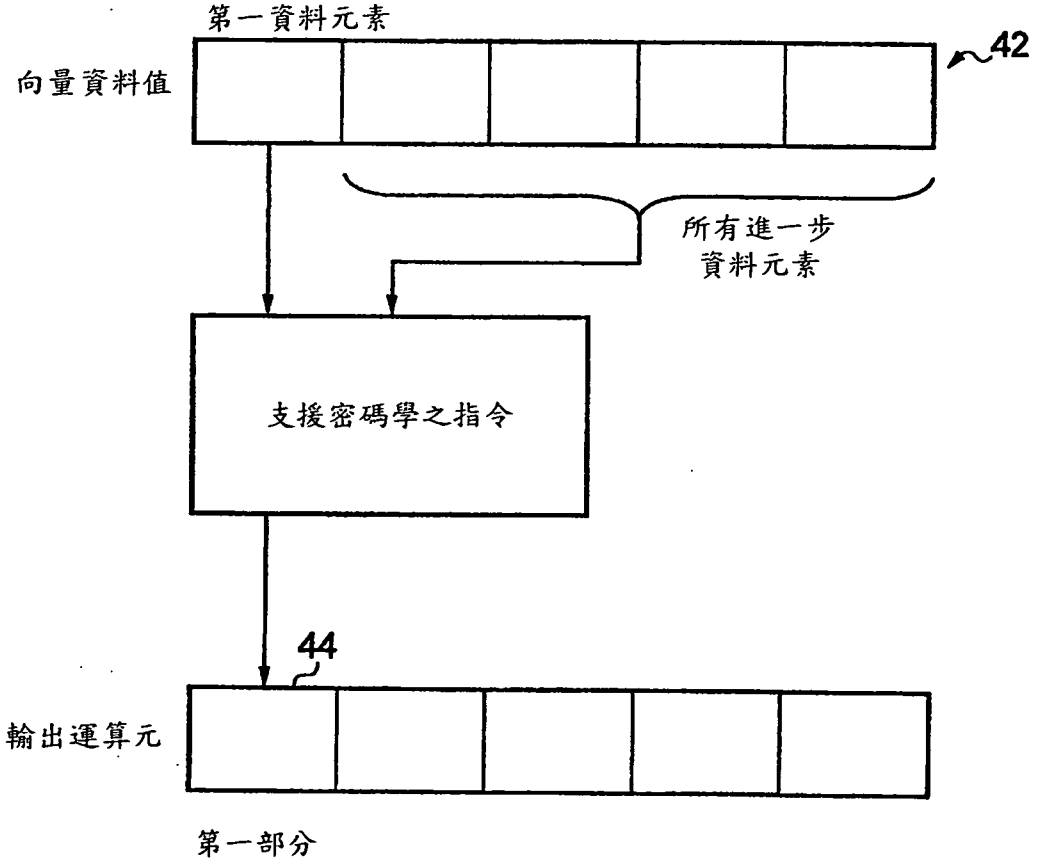

4
CPU
8
16
暫存器重新命名計數板
12
SIMD暫存器檔案
18
SIMD暫存器重新命名計數板
D
6
記憶體
10
通用處理
14
SIMD處理及密碼學支援
資料
控制訊號
程式
22
解碼器
20
指令管線
I
24
中斷電路
2
中斷

第1圖


待散列之檔案
26
散列種子160位元
32
64位元組區塊
28
4*32位元字
SHA-1
散列更新
排程更新
34
36
30
160位元中間散列
38
+
160位元輸出散列
40

第2圖

第一部分產生



第3圖