201701664

(19)中華民國智慧財產局

(12)發明說明書公開本 (11)公開編號：TW 201701664 A

(43)公開日： 中華民國 106 (2017) 年 01 月 01 日

(21)申請案號：105116796 (22)申請日： 中華民國 105 (2016) 年 05 月 27 日

(51)Int. Cl.： *H04N19/13 (2014.01)* *H04N19/91 (2014.01)*

(30)優先權：2015/05/29 美國 62/168,503

2016/05/26 美國 15/166,044

(71)申請人：高通公司 (美國) QUALCOMM INCORPORATED (US)

美國

(72)發明人：章立 ZHANG, LI (CN) ；陳建樂 CHEN, JIANLE (CN) ；趙欣 ZHAO, XIN (CN) ；李想 LI, XIANG (CN) ；劉鴻彬 LIU, HONGBIN (CN) ；陳盈 CHEN, YING (CN) ；卡茲維克茲 馬塔 KARCZEWICZ, MARTA (US)

(74)代理人：陳長文

申請實體審查：無 申請專利範圍項數：32 項 圖式數：12 共 94 頁

(54)名稱

進階之算術寫碼器

ADVANCED ARITHMETIC CODER

(57)摘要

一種熵寫碼視訊資料之實例方法，其包括：針對在一上下文自適應性寫碼過程中使用以熵寫碼該視訊資料之一語法元素之一值的複數個上下文中之一上下文判定複數個視窗大小中之一視窗大小；基於該上下文之一機率狀態熵寫碼該語法元素之該值之一位元子；基於該視窗大小及該經寫碼位元子更新該上下文之一機率狀態。該實例方法亦包括基於該上下文之該經更新機率狀態藉由該相同上下文熵寫碼一下一位元子。

An example method of entropy coding video data includes determining a window size of a plurality of window sizes for a context of a plurality of contexts used in a context-adaptive coding process to entropy code a value for a syntax element of the video data; entropy coding, based on a probability state of the context, a bin of the value for the syntax element; updating a probability state of the context based on the window size and the coded bin. The example method also includes entropy coding a next bin with the same context based on the updated probability state of the context.

指定代表圖：

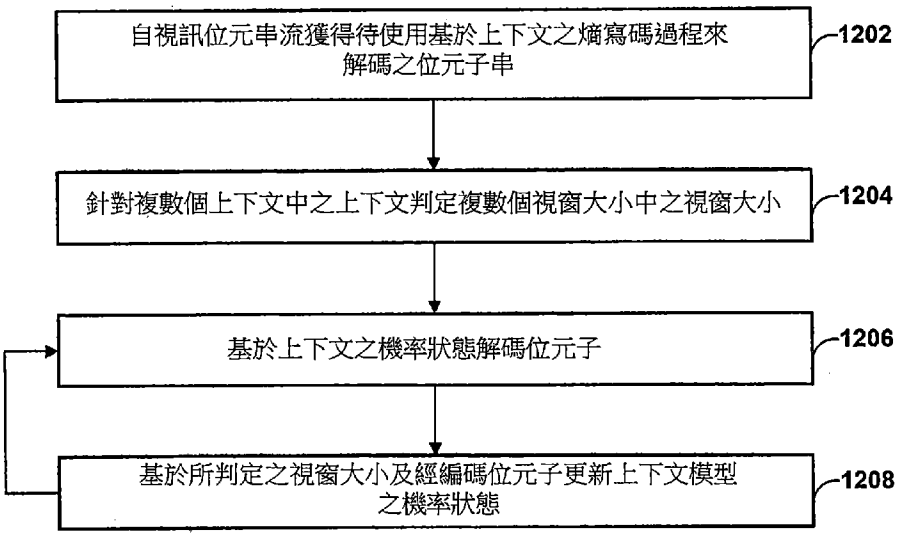


圖12

符號簡單說明：

1202・・・區塊

1204・・・區塊

1206・・・區塊

1208・・・區塊

201701664

發明摘要

※ 申請案號：105116796

※ 申請日：105.5.27

※IPC 分類：H04N 19/13 (2014.01)
H04N 19/91 (2014.01)

【發明名稱】

進階之算術寫碼器

ADVANCED ARITHMETIC CODER

【中文】

一種熵寫碼視訊資料之實例方法，其包括：針對在一上下文自適應性寫碼過程中使用以熵寫碼該視訊資料之一語法元素之一值的複數個上下文中之一上下文判定複數個視窗大小中之一視窗大小；基於該上下文之一機率狀態熵寫碼該語法元素之該值之一位元子；基於該視窗大小及該經寫碼位元子更新該上下文之一機率狀態。該實例方法亦包括基於該上下文之該經更新機率狀態藉由該相同上下文熵寫碼一下一位元子。

【英文】

An example method of entropy coding video data includes determining a window size of a plurality of window sizes for a context of a plurality of contexts used in a context-adaptive coding process to entropy code a value for a syntax element of the video data; entropy coding, based on a probability state of the context, a bin of the value for the syntax element; updating a probability state of the context based on the window size and the coded bin. The example method also includes entropy coding a next bin with the same context based on the updated probability state of the context.

【代表圖】

【本案指定代表圖】：第（12）圖。

【本代表圖之符號簡單說明】：

1202　　區塊

1204　　區塊

1206　　區塊

1208　　區塊

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

(無)

發明專利說明書

（本說明書格式、順序，請勿任意更動）

【發明名稱】

進階之算術寫碼器

ADVANCED ARITHMETIC CODER

本申請案主張2015年5月29日申請之美國臨時申請案第62/168,503號之權益，該申請案之全部內容以引用之方式併入本文中。

【技術領域】

本發明係關於視訊寫碼，且更特定而言係關於用於視訊資料之二進位算術寫碼的技術。

【先前技術】

數位視訊能力可併入至廣泛範圍的器件中，包括數位電視、數位直播系統、無線廣播系統、個人數位助理(PDA)、膝上型或桌上型電腦、數位攝影機、數位記錄器件、數字媒體播放器、視訊遊戲器件、視訊遊戲控制台、蜂巢式或衛星無線電電話、視訊電話會議器件及類似者。數位視訊器件實施視訊壓縮技術(諸如由MPEG-2、MPEG-4、ITU-T H.263、ITU-T H.264/MPEG-4第10部分、進階視訊寫碼(AVC)、高效率視訊寫碼(HEVC)標準所定義的標準及此等標準之擴展中所描述的彼等視訊壓縮技術)以更有效地傳輸、接收及儲存數位視訊資訊。

視訊壓縮技術包括空間預測及/或時間預測以減少或移除視訊序列中所固有之冗餘。對於基於區塊之視訊寫碼，可將視訊圖框或圖塊分割成區塊。可進一步分割每一區塊。框內寫碼(I)圖框或圖塊中

之區塊係使用關於同一圖框或圖塊中之相鄰區塊中之參考樣本的空間預測來編碼。框間寫碼(P或B)圖框或圖塊中之區塊可使用關於同一圖框或圖塊中之相鄰區塊中之參考樣本的空間預測或關於其他參考圖框中之參考樣本的時間預測。空間或時間預測導致待寫碼區塊之預測性區塊。殘餘資料表示待寫碼之原始區塊與預測性區塊之間的像素差。

根據指向形成預測性區塊之參考樣本之區塊的運動向量及指示經寫碼區塊與預測性區塊之間的差異之殘餘資料來編碼經框間寫碼區塊。根據框內寫碼模式及殘餘資料來編碼經框內寫碼區塊。為進行進一步壓縮，可將殘餘資料自像素域變換至變換域，從而導致可接著進行量化之殘餘變換係數。可按特定次序掃描最初配置成二維陣列之經量化之變換係數，以產生變換係數之一維向量以用於熵寫碼。

可利用不同熵寫碼過程來寫碼殘餘變換係數、運動向量資訊、語法元素及其他相關聯資訊。不同熵寫碼及其他資料壓縮過程的實例包括：上下文自適應性可變長度寫碼(CAVLC)、上下文自適應性二進位算術寫碼(CABAC)、機率區間分割熵寫碼(PIPE)、哥倫布(Golomb)寫碼、哥倫布萊斯(Golomb-Rice)寫碼及指數哥倫布寫碼。

【發明內容】

大體而言，本發明描述用於執行視訊寫碼之技術。更特定而言，本發明描述用於藉由不同視窗大小執行基於上下文之熵寫碼的實例技術。在一些實例中，本發明中描述之技術可藉由不同視窗大小啟用CABAC之效能。在其他實例中，本發明中描述之技術可應用於其他熵寫碼器，該等其他熵寫碼器將上下文用於寫碼符號，諸如基於上下文之可變長度寫碼。

在一項實例中，一種用於熵寫碼視訊資料之方法包括針對在上

下文自適應性熵寫碼過程(例如，CABAC或CAVLC過程)中使用以熵寫碼語法元素之值的複數個上下文中之上下文判定複數個視窗大小中之視窗大小。在此實例中，該方法亦包括熵寫碼語法元素之值的位元子，且基於視窗大小及經寫碼位元子更新上下文之機率狀態。在此實例中，該方法亦包括基於上下文之經更新機率狀態藉由相同上下文熵寫碼下一位元子。

在另一實例中，一種用於熵寫碼視訊資料之裝置包括一或多個處理器及記憶體，該記憶體經組態以儲存在上下文自適應性熵寫碼過程中使用以熵寫碼視訊資料之語法元素之值的複數個上下文。在此實例中，該一或多個處理器經組態以針對複數個上下文中之上下文判定視窗大小。在此實例中，該一或多個處理器經進一步組態以：基於上下文之機率狀態熵寫碼語法元素之值的位元子；基於視窗大小及經寫碼位元子更新上下文之機率狀態；及基於上下文之經更新機率狀態基於上下文之經更新機率狀態藉由相同上下文寫碼下一位元子。

在另一實例中，一種用於熵寫碼視訊資料之裝置包括：用於針對在上下文自適應性寫碼過程中使用以熵寫碼視訊資料之語法元素之值的複數個上下文中之上下文判定複數個視窗大小中之視窗大小的構件；用於基於上下文之機率狀態熵寫碼語法元素之值之位元子的構件；用於基於視窗大小及經寫碼位元子更新上下文模型之機率狀態的構件。在此實例中，該裝置亦包括用於基於上下文模型之經更新機率狀態藉由相同上下文熵寫碼下一位元子的構件。

在另一實例中，一種電腦可讀儲存媒體儲存指令，該等指令在執行時使得視訊寫碼器件之一或多個處理器進行以下操作：針對在上下文自適應性寫碼過程中使用以熵寫碼視訊資料之語法元素之值的複數個上下文中之上下文判定複數個視窗大小中的視窗大小；基

於上下文之機率狀態熵寫碼語法元素之值的位元子；基於視窗大小及經寫碼位元子更新上下文之機率狀態；及基於上下文模型之經更新機率狀態藉由相同上下文熵寫碼下一位元子。

在另一實例中，一種電腦可讀儲存媒體儲存視訊資料，該視訊資料在藉由視訊解碼器件處理時使得視訊解碼器件之一或多個處理器進行以下操作：針對使用於上下文自適應性寫碼過程中以熵寫碼語法元素之值之複數個上下文中之上下文判定複數個視窗大小中的視窗大小；基於上下文之機率狀態熵寫碼語法元素之值的位元子；基於視窗大小及經寫碼位元子更新上下文之機率狀態；及基於上下文模型之經更新機率狀態藉由相同上下文熵寫碼下一位元子。

在隨附圖式及以下描述中闡述本發明之一或多個態樣的細節。本發明中描述之技術的其他特徵、目標及優勢將自描述及圖式且自申請專利範圍顯而易見。

【圖式簡單說明】

圖1為繪示實例視訊編碼及解碼系統的方塊圖。

圖2A及圖2B為繪示二進位算術寫碼中之範圍更新過程的概念圖。

圖3為繪示二進位算術寫碼中之輸出過程的概念圖。

圖4為繪示實例視訊編碼器的方塊圖。

圖5為繪示視訊編碼器中之上下文自適應性二進位算術寫碼器的方塊圖。

圖6為繪示實例視訊解碼器的方塊圖。

圖7為繪示視訊解碼器中之上下文自適應性二進位算術寫碼器的方塊圖。

圖8繪示針對給定位元子值使用常規寫碼模式之二進位算術編碼過程。

圖9為繪示基於殘餘四分樹之實例變換方案的概念圖。

圖10為繪示基於係數群之實例係數掃描的概念圖。

圖11為繪示根據本發明之一或多種技術的用於藉由不同視窗大小執行基於上下文之熵編碼之實例過程的流程圖。

圖12為繪示根據本發明之一或多種技術的用於藉由不同視窗大小執行基於上下文之熵解碼之實例過程的流程圖。

【實施方式】

本發明之技術大體上係關於基於區塊之混合視訊寫碼中之熵寫碼模組。此等技術可應用於任何現有視訊編解碼器，諸如HEVC (高效率視訊寫碼)或此等技術可為任何未來視訊寫碼標準或其他專屬或非專屬寫碼技術中之高效寫碼工具。出於實例及解釋之目的，大體上關於HEVC (或ITU-T H.265)及/或ITU-T H.264描述本發明之技術。另外，出於實例及解釋之目的，大體上關於CABAC寫碼器描述本發明之技術，但應理解，本發明之技術可適用於其他基於上下文之熵寫碼器，諸如上下文自適應性可變長度寫碼器。

圖1為繪示根據具有可變視窗大小之CABAC設計可利用寫碼資料之技術的實例視訊編碼及解碼系統10的方塊圖。如圖1中所展示，系統10包括源器件12，其提供稍後將由目的地器件14解碼的經編碼視訊資料。詳言之，源器件12經由電腦可讀媒體16將視訊資料提供至目的地器件14。源器件12及目的地器件14可包含廣泛範圍之器件中的任一者，包括桌上型電腦、筆記型(亦即，膝上型)電腦、平板電腦、機上盒、電話手機(諸如，所謂的「智慧型」手機)、所謂的「智慧型」襯墊、電視、攝影機、顯示器件、數位媒體播放器、視訊遊戲控制台、視訊串流器件或其類似者。在一些情況下，源器件12及目的地器件14可經裝備以用於無線通信。

目的地器件14可經由電腦可讀媒體16接收待解碼的經編碼視訊

資料。電腦可讀媒體16可包含能夠將經編碼視訊資料自源器件12移動至目的地器件14的任何類型之媒體或器件。在一項實例中，電腦可讀媒體16可包含使得源器件12能夠即時將經編碼視訊資料直接傳輸至目的地器件14的通信媒體。可根據通信標準(諸如，無線通信協定)調變經編碼視訊資料，且將其傳輸至目的地器件14。通信媒體可包含任何無線或有線通信媒體，諸如，射頻(RF)頻譜或一或多個實體傳輸線。通信媒體可形成基於封包之網路(諸如，區域網路、廣域網路或諸如網際網路之全球網路)的一部分。通信媒體可包括路由器、交換器、基地台或可用於促進自源器件12至目的地器件14之通信的任何其他設備。

在一些實例中，經編碼資料可自輸出介面22輸出至儲存器件。類似地，可藉由輸入介面自儲存器件存取經編碼資料。儲存器件可包括多種分佈式或本端存取之資料儲存媒體中之任一者，諸如，硬碟機、藍光光碟、DVD、CD-ROM、快閃記憶體、揮發性或非揮發性記憶體或用於儲存經編碼視訊資料之任何其他合適的數位儲存媒體。在另一實例中，儲存器件可對應於檔案伺服器或可儲存由源器件12產生的經編碼視訊之另一中間儲存器件。目的地器件14可經由串流或下載自儲存器件存取所儲存之視訊資料。檔案伺服器可為能夠儲存經編碼視訊資料且將彼經編碼視訊資料傳輸至目的地器件14的任何類型之伺服器。實例檔案伺服器包括網頁伺服器(例如，用於網站)、FTP伺服器、網路附接儲存(NAS)器件或本端磁碟機。目的地器件14可經由任何標準資料連接(包括網際網路連接)存取經編碼視訊資料。此資料連接可包括適於存取儲存於檔案伺服器上之經編碼視訊資料的無線頻道(例如，Wi-Fi連接)、有線連接(例如，DSL、纜線數據機等)，或兩者之組合。來自儲存器件之經編碼視訊資料之傳輸可為串流傳輸、下載傳輸，或其組合。

本發明之技術不必限於無線應用或設定。該等技術可應用於支援多種多媒體應用中之任一者的視訊寫碼，諸如，空中電視廣播、有線電視傳輸、衛星電視傳輸、網際網路串流視訊傳輸(諸如，經由HTTP之動態自適應串流(DASH))、經編碼至資料儲存媒體上之數位視訊、儲存於資料儲存媒體上的數位視訊之解碼或其他應用。在一些實例中，系統10可經組態以支援單向或雙向視訊傳輸從而支援諸如視訊串流、視訊播放、視訊廣播及/或視訊電話之應用。

在圖1之實例中，源器件12包括視訊源18、視訊編碼器20及輸出介面22。目的地器件14包括輸入介面28、視訊解碼器30及顯示器件31。根據本發明，源器件12之視訊編碼器20可經組態以根據增強型CABAC設計將該等技術應用於寫碼資料。在其他實例中，源器件及目的地器件可包括其他組件或配置。舉例而言，源器件12可自外部視訊源18 (諸如，外部攝影機)接收視訊資料。同樣地，目的地器件14可與外部顯示器件介接，而非包括整合式顯示器件。

圖1之所繪示系統10僅為一項實例。可由任何數位視訊編碼及/或解碼器件執行根據增強型CABAC設計用於寫碼資料之技術。儘管本發明之技術一般由視訊編碼器件執行，但該等技術亦可由視訊編碼器/解碼器(通常被稱為「編解碼器」)執行。此外，本發明之技術亦可由視訊預處理器執行。源器件12及目的地器件14僅為源器件12在其中產生經寫碼視訊資料以供傳輸至目的地器件14之此類寫碼器件的實例。在一些實例中，器件12、14可以實質上對稱的方式操作，使得器件12、14中之每一者包括視訊編碼及解碼組件。因此，系統10可支援視訊器件12、14之間的單向或雙向視訊傳播，以用於(例如)視訊串流、視訊播放、視訊廣播或視訊電話。

源器件12之視訊源18可包括視訊俘獲器件，諸如視訊攝影機、含有先前俘獲之視訊的視訊存檔及/或用以自視訊內容提供者接收視

訊的視訊饋入介面。作為另一替代，視訊源18可產生基於電腦圖形之資料作為源視訊，或實況視訊、存檔視訊及電腦產生之視訊的組合。在一些情況下，若視訊源18為視訊攝影機，則源器件12及目的地器件14可形成所謂的攝影機電話或視訊電話。然而，如上文所提及，本發明中描述之技術一般可適用於視訊寫碼，且可適用於無線及/或有線應用。在每一情況下，可由視訊編碼器20編碼所俘獲、經預先俘獲或電腦產生之視訊。可接著藉由輸出介面22將經編碼視訊資訊輸出至電腦可讀媒體16上。

電腦可讀媒體16可包括暫時性媒體，諸如無線廣播或有線網路傳輸，或非暫時性儲存媒體(亦即，非暫時性儲存媒體)，諸如硬碟、快閃驅動器、緊密光碟、數位視訊光碟、藍光光碟或其他電腦可讀媒體。在一些實例中，網路伺服器(未展示)可自源器件12接收經編碼視訊資料，且(例如)經由網路傳輸將經編碼視訊資料提供至目的地器件14。類似地，媒體生產設施(諸如光碟衝壓設施)之計算器件可自源器件12接收經編碼視訊資料且生產含有經編碼視訊資料之光碟。當藉由視訊解碼器件進行處理時，光碟上之經編碼視訊資料可使得視訊解碼器件根據本文所揭示之不同實例解碼視訊資料。因此，在不同實例中，可理解電腦可讀媒體16包括不同形式之一或多個電腦可讀媒體。

目的地器件14之輸入介面28自電腦可讀媒體16接收資訊。電腦可讀媒體16之資訊可包括由視訊編碼器20定義之語法資訊，該語法資訊亦供視訊解碼器30使用，其包括描述區塊及其他經寫碼單元(例如，GOP)之特性及/或處理的語法元素。顯示器件32將經解碼視訊資料顯示給使用者，且可包含多種顯示器件中的任一者，諸如陰極射線管(CRT)、液晶顯示器(LCD)、電漿顯示器、有機發光二極體(OLED)顯示器或另一類型之顯示器件。

視訊編碼器20及視訊解碼器30可根據視訊寫碼標準操作，諸如，高效率視訊寫碼(HEVC)標準，亦被稱作ITU-T H.265。替代地，視訊編碼器20及視訊解碼器30可根據其他專屬或行業標準(諸如，ITU-T H.264標準，替代地被稱作MPEG-4第10部分，進階視訊寫碼(AVC))或此等標準之擴展而操作。然而，本發明之技術並不限於任何特定寫碼標準。視訊寫碼標準之其他實例包括MPEG-2及ITU-T H.263。儘管圖1中未展示，但在一些態樣中，視訊編碼器20及視訊解碼器30可各自與音訊編碼器及解碼器整合，且可包括適當MUX-DEMUX單元或其他硬體及軟體，以處置共同資料串流或單獨資料串流中之音訊及視訊兩者的編碼。若適用，則MUX-DEMUX單元可遵照ITU H.223多工器協定或諸如使用者資料報協定(UDP)之其他協定。

視訊編碼器20及視訊解碼器30各自可實施為多種合適的編碼器電路中的任一者，諸如，一或多個微處理器、數位信號處理器(DSP)、特殊應用積體電路(ASIC)、場可程式化閘陣列(FPGA)、離散邏輯、軟體、硬體、韌體，或其任何組合。當該等技術部分地在軟體中實施時，器件可將用於軟體之指令儲存於合適的非暫時性電腦可讀媒體中，且在硬體中使用一或多個處理器來執行該等指令以執行本發明之技術。視訊編碼器20及視訊解碼器30中之每一者可包括於一或多個編碼器或解碼器中，編碼器或解碼器中的任一者可整合為各別器件中之組合式編碼器/解碼器(編解碼器)的部分。

大體而言，根據HEVC，視訊圖框或圖像可劃分成包括明度及色度樣本兩者之樹型區塊或最大寫碼單元(LCU)的序列。位元串流內之語法資料可定義LCU之大小，LCU就像素之數目而言為最大寫碼單元。圖塊包括按寫碼次序之許多連續樹型區塊。視訊圖框或圖像可分割成一或多個圖塊。每一樹型區塊可根據四分樹而拆分成若干寫

碼單元(CU)。大體而言，四分樹資料結構每CU包括一個節點，其中根節點對應於樹型區塊。若將CU拆分成四個子CU，則對應於該CU之節點包括四個葉節點，該四個葉節點中之每一者對應於該等子CU中之一者。

四分樹資料結構中之每一節點可提供對應CU之語法資料。舉例而言，該四分樹中之節點可包括分裂旗標，從而指示是否將對應於該節點之CU分裂成子CU。針對CU之語法元素可經遞歸地定義，且可取決於該CU是否分裂成子CU。若CU未經進一步分裂，則其被稱作葉CU。在本發明中，即使不存在原始葉CU之明顯分裂，葉CU之四個子CU亦將被稱作葉CU。舉例而言，若16×16大小之CU未進一步分裂，則四個8×8子CU亦將被稱作葉CU，儘管16×16 CU從未分裂。

除CU不具有大小區別外，CU具有與H.264標準之巨集區塊類似的用途。舉例而言，可將樹型區塊分裂成四個子節點(亦被稱作子CU)，且每一子節點又可為上代節點且可被分裂成另外四個子節點。被稱作四分樹之葉節點之最終的未分裂子節點包含寫碼節點，該寫碼節點亦被稱作葉CU。與經寫碼位元串流相關聯之語法資料可定義可分裂樹型區塊之最大次數(其被稱作最大CU深度)，且亦可定義該等寫碼節點之最小大小。因此，位元串流亦可定義最小寫碼單元(SCU)。本發明使用術語「區塊」以在HEVC之上下文中指代CU、預測單元(PU)或變換單元(TU)中之任一者，或在其他標準(例如，H.264/AVC中之其巨集區塊及子區塊)之上下文中指代類似資料結構。

CU包括寫碼節點以及與該寫碼節點相關聯之預測單元(PU)及變換單元(TU)。CU之大小大體上對應於寫碼節點之大小，且大體上為正方形形狀。CU之大小可在8×8像素達至具有最大64×64像素或更大像素的樹型區塊之大小的範圍內。每一CU可含有一或多個PU及一或

多個TU。與CU相關聯之語法資料可描述(例如)將CU分割成一或多個PU。分割模式可在CU經跳過或直接模式編碼、框內預測模式編碼抑或框間預測模式編碼之間有區別。PU可分割成非正方形形狀。與CU相關聯之語法資料亦可描述(例如)根據四分樹將CU分割成一或多個TU。TU可為正方形或非正方形(例如，矩形)形狀。

HEVC標準允許根據TU進行變換，該等變換對於不同CU可不同。TU的大小通常係基於針對經分割LCU定義之給定CU內的PU之大小，但可能並非總是此狀況。TU的大小通常與PU相同或比PU小。在一些實例中，可使用被稱為「殘餘四分樹」(RQT)之四分樹結構而將對應於CU之殘餘樣本再分為更小單元。可將RQT之葉節點稱作變換單元(TU)。與TU相關聯之像素差值可經變換以產生可加以量化之變換係數。

葉CU可包括一或多個預測單元(PU)。大體而言，PU表示對應於該對應CU之全部或一部分的空間區域，且可包括用於針對PU擷取及/或產生參考樣本的資料。此外，PU包括與預測有關之資料。舉例而言，當PU經框內模式編碼時，PU之資料可包括於殘餘四分樹(RQT)中，該RQT可包括描述用於對應於該PU之TU的框內預測模式的資料。RQT亦可被稱作變換樹。在一些實例中，可在分葉CU語法而非RQT中傳信框內預測模式。作為另一實例，當PU經框間模式編碼時，PU可包括用於定義PU之運動資訊(諸如一或多個運動向量)的資料。定義PU之運動向量之資料可描述(例如)運動向量之水平分量、運動向量之垂直分量、運動向量之解析度(例如，四分之一像素精度或八分之一像素精度)、運動向量所指向的參考圖像，及/或運動向量之參考圖像清單(例如，清單0、清單1或清單C)。

具有一或多個PU之葉CU亦可包括一或多個變換單元(TU)。如上文所論述，可使用RQT (亦稱作TU四分樹結構)來指定該等變換單

元。舉例而言，分裂旗標可指示葉CU是否分裂成四個變換單元。接著，可將每一變換單元進一步分裂為其他若干子TU。當TU未進一步分裂時，其可被稱為葉TU。大體而言，對於框內寫碼而言，屬於葉CU之所有葉TU共用同一框內預測模式。亦即，一般應用同一框內預測模式來計算葉CU之所有TU之預測值。對於框內寫碼，視訊編碼器可使用框內預測模式將每一葉TU之殘餘值計算為在CU之對應於該TU的部分與原始區塊之間的差。TU不必限於PU的大小。因此，TU可大於或小於PU。對於框內寫碼，PU可與用於同一CU之對應葉TU共置。在一些實例中，葉TU之最大大小可對應於對應葉CU之大小。

此外，葉CU之TU亦可與各別四分樹資料結構(被稱作殘餘四分樹(RQT))相關聯。亦即，葉CU可包括指示該葉CU如何被分割成TU之四分樹。TU四分樹之根節點大體對應於葉CU，而CU四分樹之根節點大體對應於樹型區塊(或LCU)。將RQT之未被分裂的TU稱作葉TU。大體而言，除非另有指示，否則本發明分別使用術語CU及TU來指葉CU及葉TU。

視訊序列通常包括一系列視訊圖框或圖像。圖像群組(GOP)大體上包含一系列視訊圖像中之一或多者。GOP可包括GOP之標頭中、圖像中之一或多者之標頭中或別處的語法資料，該語法資料描述包括於GOP中之圖像的數目。圖像之每一圖塊可包括描述該各別圖塊之編碼模式的圖塊語法資料。視訊編碼器20通常對個別視訊圖塊內之視訊區塊進行操作，以便編碼視訊資料。視訊區塊可對應於CU內之寫碼節點。視訊區塊可具有固定或變化之大小，且可根據指定寫碼標準而在大小方面不同。

作為實例，可針對不同大小之PU執行預測。假定特定CU之大小為2N×2N，則可對2N×2N或N×N之PU大小執行框內預測，且對2N×2N、2N×N、N×2N或N×N之對稱PU大小執行框間預測。亦可針

對2NxnU、2NxnD、nLx2N及nRx2N的PU大小執行框間預測之不對稱分割。在不對稱分割中，CU之一方向未分割，而另一方向分割成25%及75%。CU之對應於25%分割之部分由「n」隨後「上(Up)」、「下(Down)」、「左(Left)」或「右(Right)」之指示來指示。因此，例如，「2N×nU」係指水平地以頂部之2N×0.5N PU及底部之2N×1.5N PU分割之2N×2N CU。

在本發明中，「N×N」與「N乘N」可互換地使用以指視訊區塊在垂直尺寸與水平尺寸方面之像素尺寸，例如，16×16像素或16乘16像素。大體而言，16×16區塊在垂直方向上將具有16個像素(y=16)且在水平方向上將具有16個像素(x=16)。同樣地，N×N區塊通常在垂直方向上具有N個像素且在水平方向上具有N個像素，其中N表示非負整數值。可按列及行來配置區塊中之像素。此外，區塊未必需要在水平方向上與在垂直方向上具有相同數目個像素。舉例而言，區塊可包含N×M個像素，其中M未必等於N。

在使用CU之PU的框內預測性或框間預測性寫碼之後，視訊編碼器20可計算CU之TU的殘餘資料。PU可包含描述在空間域(亦被稱作像素域)中產生預測性像素資料之方法或模式的語法資料，且TU可包含在對殘餘視訊資料應用變換(例如離散餘弦變換(DCT)、整數變換、小波變換或概念上類似的變換)之後變換域中的係數。殘餘資料可對應於未經編碼之圖像之像素與對應於PU之預測值之間的像素差。視訊編碼器20可形成包括表示CU之殘餘資料的經量化變換係數之TU。亦即，視訊編碼器20可計算殘餘資料(以殘餘區塊之形式)、變換殘餘區塊以產生變換係數之區塊，且接著量化變換係數以形成經量化變換係數。視訊編碼器20可形成包括經量化變換係數之TU，以及其他語法資訊(例如，TU之分裂資訊)。

如上文所述，在任何變換以產生變換係數後，視訊編碼器20可

執行變換係數之量化。量化通常指代對變換係數進行量化以可能減少用以表示係數的資料之量，從而提供進一步壓縮之過程。該量化過程可減小與該等係數中之一些或所有相關聯的位元深度。舉例而言，可在量化期間將n位元值降值捨位至m位元值，其中n大於m。

在量化之後，視訊編碼器可掃描變換係數，從而自包括經量化變換係數之二維矩陣產生一維向量。該掃描可經設計以將較高能量(且因此較低頻率)係數置於陣列前部，及將較低能量(且因此較高頻率)係數置於陣列後部。在一些實例中，視訊編碼器20可利用預定義掃描次序來掃描經量化變換係數以產生可經熵編碼之串行化向量。在其他實例中，視訊編碼器20可執行自適應性掃描。在掃描經量化變換係數以形成一維向量之後，視訊編碼器20可(例如)根據本發明中描述之上下文自適應性二進位算術寫碼(CABAC)設計來熵編碼一維向量。視訊編碼器20亦可熵編碼與經編碼視訊資料相關聯的供視訊解碼器30用於解碼視訊資料之語法元素。

大體而言，視訊解碼器30執行儘管與由視訊編碼器20執行之過程互逆但與其實質上類似的過程，以解碼經編碼資料。舉例而言，視訊解碼器30逆量化且反變換所接收TU之係數以再生殘餘區塊。視訊解碼器30使用傳信預測模式(框內預測或框間預測)以形成經預測區塊。接著視訊解碼器30 (在逐像素基礎上)使經預測區塊與殘餘區塊組合以再生原始區塊。可執行額外處理，諸如執行解區塊過程以減少沿區塊邊界之視覺假影。另外，視訊解碼器30可以儘管與視訊編碼器20之CABAC編碼過程互逆但與其實質上類似之方式使用CABAC解碼語法元素。

本發明可大體上指代將某些資訊「傳信」至另一器件(諸如，視訊解碼器30)的視訊編碼器20。然而，應理解，視訊編碼器20可藉由使某些語法元素與視訊資料之不同經編碼部分相關聯來傳信資訊。

亦即，視訊編碼器20可藉由將某些語法元素儲存至視訊資料之不同經編碼部分之標頭來「傳信」資料。在一些情況下，此類語法元素可在由視訊解碼器30接收及解碼之前經編碼及儲存(例如，儲存至儲存器件32)。因此，術語“傳信”可大體上指用於解碼經壓縮視訊資料之語法或其他資料的通信，而不管此通信是即時抑或幾乎即時發生抑或歷時時間跨度而發生，諸如，可能在編碼時在將語法元素儲存至媒體時發生，接著可由解碼器件在將語法元素儲存至此媒體之後的任何時間處擷取語法元素。

以下部分將更詳細描述BAC及CABAC技術。大體而言，BAC為遞歸區間再分程序。BAC在H.264/AVC及H.265/HEVC視訊寫碼標準中之CABAC過程中用於編碼位元子。BAC寫碼器之輸出為表示最終經寫碼機率區間內之機率的值或指標之二進位串流。機率區間係由範圍及下端值指定。範圍為機率區間之擴展。低點為寫碼區間之下限。

將算術寫碼應用於視訊寫碼係描述於D. Marpe、H. Schwarz及T. Wiegand之「Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard」(用於視訊技術之電路及系統，IEEE彙刊，2003年7月，第7期，第13卷，)中。CABAC涉及三個主要功能，即，二進位化、上下文模型化及算術寫碼。二進位化指代將語法元素映射至二進位符號(或「位元子」)之功能。二進位符號亦可被稱作「位元子串」。上下文模型化指代估計不同位元子之機率的功能。算術寫碼指代基於所估計機率將位元子壓縮至位元之後續功能。不同器件及/或其模組(諸如，二進位算術寫碼器)可執行算術寫碼功能。

在HEVC中使用若干不同二進位化過程，包括一元(U)、截斷一元(TU)、k級(kth-order)指數哥倫布(EGk)及固定長度(FL)。不同二進

位化過程之細節描述於V. Sze及M. Budagavi之「High throughput CABAC entropy coding in HEVC」(用於視訊技術之電路及系統上之IEEE彙刊(TCSVT)，2012年12月，第12期，第22卷，第1778至1791頁)中。

CABAC中之每一上下文(亦即，機率模型)由狀態表示。每一狀態(σ)隱含地表示特定符號(例如，位元子)為最不可能符號(LPS)的機率(p_σ)。符號可為LPS或最可能符號(MPS)。符號為二進位，且因此，MPS及LPS可為0或1。機率針對對應上下文經估計且(隱含地)用以使用算術寫碼器來熵寫碼符號。

BAC之過程由取決於待寫碼之上下文及正經寫碼之位元子之值改變其內部值『範圍』及『低點』的狀態機處理。取決於上下文之狀態(亦即，其機率)，將範圍劃分為$rangeMPS_\sigma$ ($state_\sigma$中最可能符號之範圍)及$rangeLPS_\sigma$ ($state_\sigma$中最不可能符號之範圍)。理論上，機率stateσ之$rangeLPS_\sigma$值由乘法運算導出：

$$rangeLPS_\sigma = range \times p_\sigma ,$$

其中p_σ為選擇LPS之機率。當然，MPS之機率為1-p_σ。等效地，$rangeMPS_\sigma$等於範圍減去$rangeLPS_\sigma$。BAC反覆地更新取決於待寫碼之上下文之狀態的範圍、當前範圍以及正經寫碼之位元子(亦即，為等於LPS或MPS之位元子)之值。

圖2A及圖2B展示在位元子*n*處之此過程的實例。在圖2A之實例100中，在位元子n處，位元子2處之範圍包括藉由LPS (p_σ)給定特定上下文狀態(σ)之機率而給定之RangeMPS及RangeLPS。實例100展示當位元子n之值等於MPS時在位元子n+1處之範圍的更新。在此實例中，低點保持相同，但在位元子n+1處之範圍的值縮小至在位元子n處之RangeMPS的值。圖2B之實例102展示當位元子n之值不等於MPS(亦即，等於LPS)時在位元子n+1處之範圍的更新。在此實例中，低

點移動至在位元子n處之RangeLPS的較低範圍值。另外，在位元子n+1處之範圍之值縮小至在位元子n處的RangeLPS之值。

在HEVC中，藉由9個位元表達範圍且藉由10個位元表達低點。存在以足夠精度維持範圍及低點值之再歸一化過程。再歸一化出現在每當範圍小於256時。因此，在再歸一化之後範圍始終等於或大於256。取決於範圍及低點之值，BAC將『0』或『1』輸出至位元串流，或更新內部變數(被稱作BO：突出位元)以保留用於未來輸出。圖3展示取決於範圍之BAC輸出的實例。舉例而言，當範圍及低點高於特定臨限值(例如，512)時，將『1』輸出至位元串流。當範圍及低點低於特定臨限值(例如，512)時，將『0』輸出至位元串流。當範圍及低點在某些臨限值之間時，不輸出任何事物至位元串流。替代地，遞增BO值且編碼下一位元子。

在HEVC之CABAC上下文模型中，存在128種狀態。存在可自0至63之64種可能的LPS機率(由狀態σ表示)。每一MPS可為零或一。因此，128種狀態為64種狀態機率乘以MPS之2個可能值(0或1)。因此，可將機率模型儲存為7位元條目。在每一7位元條目中，可分配6個位元以用於表示機率狀態，且可分配1個位元以用於可適用上下文記憶體中之最可能符號(MPS)。

為減少導出LPS範圍($rangeLPS_\sigma$)之計算，針對所有情況之結果經預先計算且儲存為HEVC中之查找表中的近似值。因此，可在無任何乘法運算之情況下藉由使用簡單表查找來獲得LPS範圍。對於一些器件或應用而言避免乘法運算可為至關重要的，此係由於此操作可引起多種硬體架構中之顯著潛時。

可使用4行預先計算之LPS範圍表而非乘法運算。將範圍劃分為四個片段。可由問題$(range>>6)\&3$導出片段索引。實際上，片段索引係藉由自實際範圍移位及丟棄位元來導出。以下表1展示可能範圍

及其對應索引。

表1-範圍索引

範圍	256-319	320-383	384-447	448-511
(range>>6) & 3	0	1	2	3

LPS範圍表接著具有64個條目(每一機率狀態有一者)乘以4 (每一範圍索引有一者)。每一條目為LPS範圍，亦即，將範圍乘以LPS機率之值。以下表2中展示此表之部分的實例。表2描繪機率狀態9至12。在HEVC之一個提議中，機率狀態可在0至63之範圍內。

表2-RangeLPS

機率狀態(σ)	RangeLPS 索引0	索引1	索引2	索引3
…	…	…	…	…
9	90	110	130	150
10	85	104	123	142
11	81	99	117	135
12	77	94	111	128
…	…	…	…	…

在每一片段(亦即，範圍值)中，每一機率狀態σ之LPS範圍為預定義的。換言之，機率狀態σ之LPS範圍經量化為四個值(亦即，每一範圍索引有一個值)。在給定點處使用之特定的LPS範圍取決於範圍屬於哪一片段。表中所使用之可能的LPS範圍之數目為表行數(亦即，可能的LPS範圍值之數目)與LPS範圍精度之間的折衷。大體而言，更多行產生LPS範圍值之更小量化錯誤，但亦增加對用以儲存表之更多記憶體的需求。更少行增加量化錯誤，但亦減少需要用以儲存表之記憶體。

如上文所述，每一LPS機率狀態具有對應機率。在HEVC中，根據為遞歸方程式之以下方程式(1)針對LPS (最不可能符號)導出64個代表性機率值$p_\sigma \in [0.01875,\ 0.5]$。

$$p_\sigma = \alpha * p_{\sigma-1} \text{ 對於所有 } \sigma = 1, \ldots, 63$$
$$\text{其中} \alpha = \left(\frac{0.01875}{0.5}\right)^{1/63} \quad \textbf{(1)}$$

在上述實例中，該組機率中之所選比例因子$\alpha \approx 0.9492$及基數N = 64兩者表示機率表示之準確度與對快速適應之需要之間的良好折衷。在一些實例中，更接近1之值α可產生具有較高準確度(「穩態行為」)之緩慢適應，而可針對非靜止情況藉由以經降低準確度為代價減少值α來實現更快適應。比例因子α可對應於指示先前經編碼位元子之數目的視窗大小，該等先前經編碼位元子對當前更新具有顯著影響。MPS (最可能符號)之機率等於1減去LPS (最不可能符號)之機率。換言之，可由公式(1-LPS)表示MPS之機率，其中『LPS』表示LPS之機率。因此，可由HEVC中之CABAC表示之機率範圍為[0.01875, 0.98125 (=1-0.01875)]。

由於用於寫碼語法元素之值之位元(或「位元子」)的上下文之機率狀態經更新，CABAC為自適應性的，以便遵循信號統計(亦即，先前經寫碼位元子之值，例如，對於語法元素而言)。更新過程如下。對於給定機率狀態，更新取決於狀態索引及經識別為LPS或MPS之經編碼符號的值。作為更新過程之結果，新的機率狀態經導出，其包括潛在經修改LPS機率估計及(必要時)經修改MPS值。

上下文交換可出現在寫碼每一位元子之後。倘若位元子值等於MPS，則給定狀態索引僅增加1。此用於除當MPS出現在狀態索引62處時之外的所有狀態，其中LPS機率已處於其最小值(或等效地，達到最大MPS機率)。在此情況下，狀態索引保持固定，直至LPS可見或最末位元子值經編碼(將特殊終止狀態用於最末位元子值之特殊情況)。當LPS出現時，藉由以特定量遞減狀態索引來改變狀態索引，如以下方程式中所展示。大體而言，此規則應用於具有以下異常之LPS每次出現時。假定已在具有索引$\sigma=0$之狀態處編碼LPS，該索引對應於等機率情況，則狀態索引保持固定，但將切換MPS值使得LPS及MPS之值將被互換。在所有其他情況下，不管哪一符號已經編

碼，MPS值將未更改。大體而言，視訊寫碼器可根據以下方程式(2)導出新的機率狀態，該方程式展示給定LPS可能性p_{old}與其經更新對應物p_{new}之間的關係。

$$p_{new} = \begin{cases} \max(\alpha * p_{old}, p_{62}), & \text{若MPS出現} \\ \alpha * p_{old} + (1-\alpha), & \text{若LPS出現} \end{cases} \quad (2)$$

為降低複雜度，視訊寫碼器可實施CABAC，使得可藉由各自具有多個條目之至多兩個表實現所有轉換規則。作為一項實例，可藉由各自具有7位元無正負號整數值之128個條目的至多兩個表(例如，以下表3及表4)實現所有轉換規則。作為另一實例，可藉由各自具有6位元無正負號整數值之63條目的至多兩個表(例如，HEVC中之表9-41)實現所有轉換規則。在更新之後，視訊寫碼器可在MPS值經寫碼時將給定狀態索引i定義為新的狀態索引TransIdxMPS[i]，或在LPS值經寫碼時定義為TransIdxLPS[i]。

表3

```
TransIdxMPS[ 128 ] =
{
 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113,
 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 124, 125, 126, 127
};
```

表4

```
TransIdxLPS[ 128 ] =
{
   1, 0, 0, 1, 2, 3, 4, 5, 4, 5, 8, 9, 8, 9, 10, 11,
   12, 13, 14, 15, 16, 17, 18, 19, 18, 19, 22, 23, 22, 23, 24, 25,
   26, 27, 26, 27, 30, 31, 30, 31, 32, 33, 32, 33, 36, 37, 36, 37,
   38, 39, 38, 39, 42, 43, 42, 43, 44, 45, 44, 45, 46, 47, 48, 49,
   48, 49, 50, 51, 52, 53, 52, 53, 54, 55, 54, 55, 56, 57, 58, 59,
   58, 59, 60, 61, 60, 61, 60, 61, 62, 63, 64, 65, 64, 65, 66, 67,
   66, 67, 66, 67, 68, 69, 68, 69, 70, 71, 70, 71, 70, 71, 72, 73,
   72, 73, 72, 73, 74, 75, 74, 75, 74, 75, 76, 77, 76, 77, 126, 127
};
```

在一些實例中，視訊寫碼器可藉由單一表TransIdxLPS判定狀態轉換，該表在已觀測到LPS之情況下針對給定狀態索引σ判定新的經更新狀態索引TransIdxLPS [σ]。可藉由使狀態索引僅(飽和)增加固定值1獲得MPS驅動轉換，從而得到經更新狀態索引min(σ+1, 62)。

如上文所論述，上下文模型化提供準確機率估計，該估計為達成較高寫碼效率之促進因子。因此，上下文模型化為自適應性過程。不同上下文可用於不同位元子，且可基於先前寫碼位元子之值更新上下文之機率。具有類似分佈之位元子常常共用同一上下文。可基於語法元素類型、語法元素中之位元子位置(binIdx)、明度/色度資訊，相鄰資訊等選擇每一位元子之上下文。

在寫碼給定圖塊之前，基於一或多個預定義值初始化機率模型。舉例而言，給定由qp表示之輸入量化參數及由initVal表示之預定義值，可根據以下方程式(3)導出機率模型(由狀態及MPS表示)之7位元條目。

```
qp  = Clip3(0, 51, qp);
斜度 = ( initVal >>4)*5 - 45;
偏移 = ((initVal &15)<<3)-16;
initState = min( max( 1, ( ( ( 斜度 * qp ) >> 4 ) + 偏移 ) ), 126 );
MPS = (initState >= 64 );
狀態索引 = ( (mpState? (initState - 64) : (63 - initState)) <<1) + MPS;
```

(3)

經導出狀態索引隱含地包括MPS資訊。更特定而言，當狀態索引為偶數(even)值時，MPS值等於0。相反地，當狀態索引為奇數(odd)值時，MPS值等於1。「initVal」之值在具有8位元精度之範圍[0, 255]內。預定義值「initVal」取決於圖塊。換言之，使用機率模型之三組上下文初始化參數，每一組分別在I圖塊、P圖塊及B圖塊中。以此方式，經組態以執行CABAC之視訊編碼器件經啟用以在三個初始化表之間選擇此等圖塊類型，使得可實現對不同寫碼情境及/或不同類型之視訊內容的更佳擬合。

根據HEVC，另一方法可應用於允許藉由B (或P)圖塊初始化一個P (或B)圖塊。相反地，該方法可應用於允許藉由P圖塊初始化一個B圖塊。相關語法元素描述於下表5(其對應於HEVC之章節7.3.6.1)中，且相關語義及解碼過程在表5之後的下文中描述。

表5

slice_segment_header( ) {	描述符
first_slice_segment_in_pic_flag	u(1)
若 (nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
若(!first_slice_segment_in_pic_flag) {	
若(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
若(!dependent_slice_segment_flag) {	
對於(i = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
若(output_flag_present_flag)	
pic_output_flag	u(1)

若(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
若(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
若(!short_term_ref_pic_set_sps_flag)	
short_term_ref_pic_set(num_short_term_ref_pic_sets)	
否則若(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
若(long_term_ref_pics_present_flag) {	
…	
}	
}	
若(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
若(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
slice_sao_chroma_flag	u(1)
}	
若(slice_type == P || slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
若(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
若(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
若(lists_modification_present_flag && NumPocTotalCurr > 1)	
ref_pic_lists_modification( )	
若(slice_type == B)	
mvd_l1_zero_flag	u(1)
若(cabac_init_present_flag)	
cabac_init_flag	u(1)
若(slice_temporal_mvp_enabled_flag) {	
若(slice_type == B)	
collocated_from_l0_flag	u(1)
若((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) || (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
若((weighted_pred_flag && slice_type == P) || (weighted_bipred_flag && slice_type == B))	
pred_weight_table( )	
five_minus_max_num_merge_cand	ue(v)
}	

...	
byte_alignment( )	
}	

表5之語法元素的語義可定義如下：

cabac_init_present_flag等於1指定cabac_init_flag存在於指代PPS之圖塊標頭中。cabac_init_present_flag等於0指定cabac_init_flag不存在於指代PPS之圖塊標頭中。

如下文描述之解碼過程中所定義，cabac_init_flag指定用於判定上下文變數之初始化過程中所使用之初始化表的方法。當cabac_init_flag不存在時，推斷其等於0。

描述符：

ae(v)：經上下文自適應性算術熵寫碼之語法元素。

b(8)：具有位元串(8個位元)之任何型樣的位元組。

f(n)：使用n個位元書寫(自左至右)之固定型樣位元串，其中首先為左方位元。

se(v)：經帶正負號整數0階指數哥倫布寫碼之語法元素，其中首先為左方位元。

u(n)：使用n個位元之無正負號整數。當n為語法表中之「v」時，位元數目以取決於其他語法元素之值的方式變化。

ue(v)：經無正負號整數0階指數哥倫布寫碼之語法元素，其中首先為左方位元。

HEVC之表9-4為需要初始化的三種初始化類型中的每一者提供上下文索引(ctxIdx)。表9-4進一步包括表號(ctxTable)，該表號包括初始化所需之initValue的值。對於P及B圖塊類型，initType之導出取決於cabac_init_flag語法元素之值。視訊寫碼器可導出變數，如使用由以下偽碼描述之操作導出initType：

若(slice_type　==　I)

initType = 0

否則若(slice_type == P)

initType = cabac_init_flag ? 2 : 1

否則

initType = cabac_init_flag ? 1 : 2

新的算術寫碼器描述於以下文件中：Alshin等人之「Multi-parameter probability up-date for CABAC」(文件：JCTVC-F254，ITU-T SG 16 WP 3及ISO/IEC JTC 1/SC 29/WG 11之JCT-VC，第6次會議：意大利，托里諾，2011年7月14日至22日(在下文中為「JCTVC-F254」))及Alshin等人之「CE1 (subset B): Multi-parameter probability up-date for CABAC」(文件：JCTVC-G764，ITU-T SG 16 WP 3及ISO/IEC JTC 1/SC 29/WG 11之JCT-VC，第7次會議：瑞士，日內瓦，2011年11月21日至30日(在下文中為「JCTVC-G764」))。在JCTVC-F254及JCTV-G764中，將每一機率表示為自1至32767之整數。如此，藉由16位元之精度進行所有計算。JCTVC-F254及JCTV-G764中提出之寫碼器將具有無乘法公式之均勻網目及顯式計算用於機率更新，而非將AVC CABAC中利用之查找表(例如，如上文所論述之TransIdxMPS及TransIdxLPS)及指數網目用於機率。

假設機率P_i由機率索引表示，該機率索引為自0至2^k之整數P_i (例如，其中k等於15) (例如，如以下方程式(4)所展示)。

$$p_i=P_i/2^k \quad (4)$$

隨後用於機率之最常使用的後繼公式在現代算術編解碼器中更新(例如，如由以下方程式(5)所展示)。

$$p_{new} = \alpha y+(1- \alpha)\, p_{old} \quad (5)$$

在方程式(5)中，若當前符號與最可能符號(MPS)匹配，則y等於「零」，否則y等於「一」。此公式(亦即，方程式(5))為最不可能符號

(LPS)之機率提供估計值。類似於以上論述，參數α可對應於指示先前經編碼位元子之數目的視窗大小，該等先前經編碼位元子對當前更新具有顯著影響。

若吾人假定視窗大小(W)為二之冪($W=1/2^M$，M為正整數)，且將方程式(4)中之Pi給定為輸入p_{old}，則可以如下展示之方程式(6)重寫經更新機率索引。

$$P_i = ((2^k) >> M) + P_i - (P_i >> M) \quad (6)$$

在由JCTVC-F254及JCTV-G764提出之一種機率更新模型中，M對於所有上下文而言為固定的且僅一個暫存器用於記錄經更新機率。在一項實例中，M經設定等於6。亦即，視窗大小等於64。機率更新過程可由以下方程式(7)表示。

$$P_{new} = ((2^k) >> 6) + P_i - (P_i >> 6) \quad (7)$$

由JCTVC-F254及JCTV-G764提出之技術的主要想法為藉由不同視窗大小使用若干機率估計(而非僅一個)且將其組合為加權平均值以用於下一位元子機率預測。以下方程式(8)及(9)說明由JCTVC-F254及JCTV-G764提出之技術的實例。方程式(8)中之計算對於每一機率pi而言為獨立的。

$$p_{i\ new} = W_i y + (1 - W_i)\, p_{i\ old} \quad (8)$$

$$p_{new} = \Sigma \beta_i\, p_{i\ new} \quad (9)$$

方程式(8)中之計算對於每一機率p_i而言為獨立的。

在由JCTVC-F254及JCTV-G764提出之方法中，用於機率估計之線性組合由對應$W_0=16$及$W_1=256$ ($W_i = 1/\alpha_i$)之兩個被加數構成，如方程式(10)及方程式(11)中所展示。在方程式(10)及方程式(11)中，若最末寫碼位元子為「1」，則$Y=2^{15}$，且若最末寫碼位元子為「0」，「>>M」為針對M個位元之算術右移位，則Y=0。

$$P_0 = (Y>>4) + P_0 - (P_0>>4) \quad (10)$$

$$P_1 = (Y>>8) + P_1 - (P_0>>8) \quad (11)$$

$$P = (P_0 + P_1+1)>>1 \quad (12)$$

對於短的過渡期，僅具有較快更新速度之短距離預測(亦即，更小視窗大小)為較佳的。但在穩定之後，靠近最佳值機率二之更新模型對於大多數上下文而言為更準確的。JCTVC-F254及JCTV-G764提出在最末初始化以後引入更新計數器。在每一更新之後，計數增加一。直至計數超過某一臨限值，將僅使用如由方程式(10)所定義之短「視窗大小」模型。當計數達到臨限值時，吾人應切換至如上文由方程式(12)所定義之更準確兩種機率更新模型。藉由512×64查找表執行由JCTVC-F254及JCTV-G764提出之範圍計算過程。

根據由JCTVC-F254及JCTV-G764提出之方法，應用不同上下文初始化方法。具體言之，針對如方程式(13)中所展示之每一上下文預定義兩個參數(分別地由asCtxInit[0]及asCtxInit[1]表示)。

```
Int iQPreper = I slice ? 37 : 40;
Int c=asCtxInit[ 0 ]+asCtxInit[ 1 ]*( iQp - iQPreper);          (13)
iP0= min( max( 1, c ), 32767 );
```

對於一種機率更新模型，由具有15位元精度之iP0表示上下文。對於兩種機率更新模型，首先將另一變數iP1設定等於iP0且進一步要求已經寫碼之位元子之數目的計數。在由JCTVC-F254及JCTV-G764提出之方法中，asCtxInit[0]及asCtxInit[1]兩者均儲存於16位元中。

然而，在一些實例中，上述技術(亦即，HEVC之CABAC技術及由JCTVC-F254及JCTV-G764提出之修改)可具有可降低寫碼效率及/或次最佳地利用寫碼器系統資源的一或多個問題。

作為一項實例，在基於算術寫碼器技術(例如，如HEVC或

H.264/AVC中所使用)之上述查找表中，機率更新係基於具有固定視窗大小之固定表(亦即，TransIdxLPS及TransIdxMPS)。固定視窗大小之此使用使得更新速度為固定的。然而，針對給定CTU或圖塊，語法元素出現且需待寫碼之頻率可完全不同。對與針對給定CTU或圖塊以不同頻率出現之語法元素組合的固定更新速度之限制可導致較不頻繁出現之語法元素的所估計機率為次最佳的。舉例而言，對於一個CU，可傳信高達2之inter_pred_idc之值，而一個CU內之變換係數可經若干次寫碼。在此情況下，當對此等語法元素使用相同更新速度時，inter_pred_idc等於1之所估計機率在寫碼一個完整圖塊之後可仍為次最佳的，即使變換係數之機率可變為相對最佳。

作為另一實例，在基於計數器技術(例如，如由JCTVC-F254及JCTV-G764所提出)之上述算術寫碼器中，機率更新速度為固定的，而可能不需要導致針對較不頻繁選擇之語法元素之低效率的高精度(例如，可能機率索引可為[1, 2^{15}-1])。

作為另一實例，在基於計數器技術之算術寫碼器的兩種機率更新模型組件中，兩個狀態參數(機率索引)必須經儲存及更新，該兩個狀態參數可不合需要地限制CABAC過程之輸送量。

作為又一實例，在影像/視訊寫碼系統中，可使用數百個上下文。在由JCTVC-F254及JCTV-G764提出之技術中，每上下文要求32個位元，而對HEVC中之算術寫碼器而言僅8個位元為足夠的。因此，用於在由JCTVC-F254及JCTV-G764提出之技術中的上下文初始化之預定義值的儲存增加300%，對於關於儲存之硬體實施而言該增加可為非所需的。

根據本發明之一或多種技術，視訊寫碼器(例如，視訊編碼器20及/或視訊解碼器30)可將不同視窗大小用於不同上下文。舉例而言，與將固定視窗大小用於所有上下文之HEVC相反，視訊寫碼器可在更

新第一上下文時使用第一視窗大小且在更新第二上下文時使用第二不同視窗大小。在一些實例中，視訊寫碼器可將相對較小視窗大小用於不頻繁使用之上下文，且可將相對較大視窗大小用於頻繁使用之上下文。藉由使用更緊密適合上下文使用之頻率的視窗大小，視訊寫碼器可藉由準確度與將固定視窗大小用於所有上下文之適應速度之間的更有利平衡來更新上下文。以此方式，本發明之技術可改善CABAC之效率，CABAC可使得視訊寫碼器能夠減少編碼視訊資料所需的位元數目。

本發明中所描述之技術可(例如)在視訊編碼器、視訊解碼器或組合式視訊編碼器解碼器(編解碼器)內執行。特定而言，此類技術可在視訊編碼器之熵編碼單元及/或視訊解碼器之熵解碼單元中執行。該等技術可(例如)在CABAC過程內執行，該CABAC過程可經組態以支援視訊寫碼，諸如根據HEVC標準之態樣的視訊寫碼。熵編碼及解碼單元可以互逆或相反方式應用寫碼過程，(例如)以編碼或解碼各種視訊資料中的任一者，諸如與殘餘視訊資料相關聯之經量化變換係數、運動向量資訊、語法元素及可在視訊編碼及/或視訊解碼過程中使用之其他類型之資訊。

圖4為繪示如本發明中所描述之可經組態以利用BAC寫碼技術之視訊編碼器20之實例的方塊圖。將出於說明之目的在HEVC寫碼之上下文中描述視訊編碼器20，但關於其他寫碼標準或方法並不限制本發明。此外，視訊編碼器20可經組態以根據HEVC之範圍擴展實施技術。

視訊編碼器20可對視訊圖塊內之視訊區塊執行框內寫碼及框間寫碼。框內寫碼依賴於空間預測以減少或移除給定視訊圖像內之視訊的空間冗餘。框間寫碼依賴於時間預測或視圖間預測以減少或移除視訊序列之鄰近圖像內之視訊的時間冗餘或減少或移除其他視圖

中之視訊冗餘。

在圖4之實例中，視訊編碼器20包括視訊資料記憶體40、預測處理單元42、參考圖像記憶體64、求和器50、變換處理單元52、量化處理單元54及熵編碼單元56。預測處理單元42又包括運動估計單元44、運動補償單元46及框內預測單元48。為視訊區塊重新建構，視訊編碼器20亦包括逆量化處理單元58、逆變換處理單元60及求和器62。亦可包括解區塊濾波器(圖4中未展示)以便對區塊邊界進行濾波，從而自經重建構視訊中移除區塊效應假影。若需要，解區塊濾波器通常將對求和器62之輸出進行濾波。除解區塊濾波器之外，亦可使用額外迴路濾波器(迴路中或迴路後)。

視訊資料記憶體40可儲存待由視訊編碼器20之組件編碼的視訊資料。儲存於視訊資料記憶體40中之視訊資料可(例如)自視訊源18獲得。 參考圖像記憶體64為儲存供由視訊編碼器20用於編碼視訊資料(例如，以框內寫碼模式或框間寫碼模式，亦被稱作框內預測寫碼模式或框間預測寫碼模式)之參考視訊資料之解碼圖像緩衝器(DPB)的一項實例。視訊資料記憶體40及參考圖像記憶體64可由多種記憶體器件中之任一者形成，諸如動態隨機存取記憶體(DRAM) (包括同步DRAM (SDRAM))、磁阻式RAM (MRAM)、電阻式RAM (RRAM)或其他類型之記憶體器件。視訊資料記憶體40及參考圖像記憶體64可由相同的記憶體器件或單獨記憶體器件提供。在各種實例中，視訊資料記憶體40可與視訊編碼器20之其他組件一起在晶片上，或相對於彼等組件在晶片外。

在編碼過程期間，視訊編碼器20接收待寫碼之視訊圖像或圖塊。圖像或圖塊可劃分為多個視訊區塊。運動估計單元44及運動補償單元46執行所接收視訊區塊相對於一或多個參考圖像中之一或多個區塊的框間預測性寫碼以提供時間壓縮或提供視圖間壓縮。框內

預測單元48可替代地執行所接收視訊區塊相對於與待寫碼區塊相同之圖像或圖塊中之一或多個相鄰區塊的框內預測性寫碼以提供空間壓縮。視訊編碼器20可執行多個寫碼遍次(例如，以選擇用於視訊資料之每一區塊的適當寫碼模式)。

此外，分割單元(未展示)可基於對先前寫碼遍次中之先前分割方案的評估而將視訊資料之區塊分割為子區塊。舉例而言，分割單元可首先將圖像或圖塊分割成LCU，且基於率失真分析(例如，率失真最佳化)將該等LCU中之每一者分割成子CU。預測處理單元42可進一步產生指示將LCU分割為子CU之四分樹資料結構。四分樹之葉節點CU可包括一或多個PU及一或多個TU。

預測處理單元42可(例如)基於錯誤結果選擇框內或框間寫碼模式中之一者，且將所得經框內或框間寫碼區塊提供至求和器50以產生殘餘區塊資料及提供至求和器62以重建構用作參考圖像之經編碼區塊。預測處理單元42亦將語法元素(諸如，運動向量、框內模式指示符、分割資訊及其他此類語法資訊)提供至熵編碼單元56。

運動估計單元44及運動補償單元46可高度整合，但出於概念性目的而單獨說明。由運動估計單元44所執行之運動估計為產生估計視訊區塊之運動的運動向量之程序。舉例而言，運動向量可指示在當前視訊圖像內之視訊區塊的PU相對於在參考圖像(或其他經寫碼單元)內之預測性區塊(其相對於在該當前圖像(或其他經寫碼單元)內正經寫碼之當前區塊)的位移。預測性區塊為就像素差而言被發現緊密地匹配待寫碼區塊之區塊，該像素差可藉由絕對差總和(SAD)、平方差總和(SSD)或其他差度量判定。在一些實例中，視訊編碼器20可計算儲存於參考圖像記憶體64中之參考圖像的子整數像素位置之值。舉例而言，視訊編碼器20可內插該參考圖像之四分之一像素位置、八分之一像素位置或其他分數像素位置之值。因此，運動估計單元

44可相對於全像素位置及分數像素位置執行運動搜尋並輸出具有分數像素精確度之運動向量。

運動估計單元44藉由比較PU之位置與參考圖像之預測性區塊的位置來計算經框間寫碼圖塊中之視訊區塊之PU的運動向量。參考圖像可選自識別儲存於參考圖像記憶體64中之一或多個參考圖像的一或多個參考圖像清單(RPL)。運動估計單元44將所計算之運動向量發送至熵編碼單元56及運動補償單元46。在一些實例中，運動估計單元44可將所選參考圖像之指示發送至熵編碼單元56。

由運動補償單元46執行之運動補償可涉及基於由運動估計單元44判定之運動向量來提取或產生預測性區塊。又，在一些實例中，運動估計單元44及運動補償單元46可在功能上整合。在接收到當前區塊之PU的運動向量後，運動補償單元46即可在參考圖像清單(RPL)中之一者中定位運動向量指向之預測性區塊。如下文所論述，求和器50藉由自正經寫碼之當前區塊的像素值減去預測性區塊之像素值來形成殘餘視訊區塊，從而形成像素差值。大體而言，運動估計單元44相對於明度分量執行運動估計，且運動補償單元46將基於明度分量計算之運動向量用於色度分量與明度分量兩者。預測處理單元42亦可產生與供視訊解碼器30在解碼視訊圖塊之視訊區塊時使用的視訊區塊及視訊圖塊相關聯之語法元素。

作為由運動估計單元44及運動補償單元46執行之框間預測的替代方案，框內預測單元48可對當前區塊進行框內預測，如上文所描述。詳言之，框內預測單元48可判定待用以編碼當前區塊之框內預測模式。在一些實例中，框內預測單元48可(例如)在單獨編碼遍次期間使用不同框內預測模式編碼區塊，且框內預測單元48可自複數個框內預測模式選擇適當之框內預測模式以供使用。

舉例而言，框內預測單元48可針對不同經測試框內預測模式使

用率失真分析計算率失真值，且自該等經測試模式當中選擇具有最佳率失真特性之框內預測模式。率失真分析大體上判定經編碼區塊與原始、未編碼區塊(其經編碼以產生經編碼區塊)之間的失真(或誤差)量，以及用以產生經編碼區塊之位元率(亦即，位元之數目)。框內預測單元48可根據不同經編碼區塊之失真及速率來計算比率以判定哪一框內預測模式展現該區塊之最佳率失真值。在一些實例中，複數個框內預測模式中之每一者可具有可由框內預測單元48傳信(亦即，至視訊解碼器)之對應模式索引。

視訊編碼器20藉由自正經寫碼之原始視訊區塊減去來自預測處理單元42之預測資料而形成殘餘視訊區塊。求和器50表示執行此減法運算之一或多個組件。

變換處理單元52將變換(諸如離散餘弦變換(DCT)或概念上類似之變換)應用於殘餘區塊，從而產生包含殘餘變換係數值之視訊區塊。變換處理單元52可執行概念上類似於DCT之其他變換。亦可使用小波變換、整數變換、子頻帶變換或其他類型之變換。在任何情況下，變換處理單元52將變換應用於殘餘區塊，從而產生殘餘變換係數區塊。該變換可將殘餘資訊自像素值域轉換至變換域，諸如頻域。

變換處理單元52可將所得變換係數發送至量化處理單元54。量化處理單元54量化變換係數以進一步減小位元率。該量化過程可減小與該等係數中之一些或所有相關聯的位元深度。可藉由調整量化參數來修改量化程度。在一些實例中，量化處理單元54可接著對包括經量化變換係數之矩陣執行掃描。替代地，熵編碼單元56可執行該掃描。

在將變換係數掃描至一維陣列後，熵編碼單元56即可將諸如上下文自適應性可變長度寫碼(CAVLC)、上下文自適應性二進位算術

寫碼(CABAC)、機率區間分割熵寫碼(PIPE)、哥倫布寫碼、哥倫布萊斯寫碼、指數哥倫布寫碼、基於語法之上下文自適應性二進位算術寫碼(SBAC)之熵寫碼或另一熵寫碼方法應用於係數。根據本發明之實例，儘管對各種不同熵寫碼過程進行參考，但熵編碼單元56可經組態以如上文所述執行BAC寫碼。

為了執行CAVLC，熵編碼單元56可選擇用於待傳輸之符號的可變長度碼。可建構VLC中之碼字，使得相對較短碼對應於更可能的符號，而較長碼對應於較不可能的符號。以此方式，相對於(例如)針對待傳輸之每一符號使用相等長度碼字，使用VLC可達成位元節省。

為執行CABAC，熵編碼單元56可選擇適用於特定上下文之上下文，以編碼待傳輸之符號。上下文可係關於(例如)相鄰值是否為非零。熵編碼單元56亦可熵編碼語法元素，諸如代表所選變換之信號。在藉由熵編碼單元56熵寫碼之後，可將所得經編碼視訊傳輸至另一器件(諸如，視訊解碼器30)或加以存檔以供稍後傳輸或擷取。

根據本發明之一或多種技術，熵編碼單元56可在熵編碼供視訊解碼器(諸如，視訊解碼器30)解碼視訊資料時使用之資料(例如，表示為一維二進位向量之語法元素值)時選擇不同視窗大小。下文參考圖5論述熵編碼單元56之一項實例的其他細節。

逆量化處理單元58及逆變換處理單元60分別應用逆量化及逆變換以重建構像素域中之殘餘區塊(例如，供稍後用作參考區塊)。

運動補償單元46亦可將一或多個內插濾波器應用於參考區塊以計算用於運動估計之子整數像素值。求和器62將經重建構殘餘區塊添加至由運動補償單元46產生之運動補償預測區塊，以產生用於儲存於參考圖像記憶體64中之經重建構視訊區塊。經重建構視訊區塊可由運動估計單元44及運動補償單元46用作參考區塊，以對後續視

訊圖像中之區塊進行框間寫碼。在一些實例中，諸如在將當前圖像用作預測當前圖像之參考圖像的情況下，運動補償單元46及/或求和器62可在寫碼當前圖像時以規律時間間隔更新由參考圖像記憶體64儲存之當前圖像的版本。作為一項實例，運動補償單元46及/或求和器62可在寫碼當前圖像之每一區塊之後更新由參考圖像記憶體64儲存之當前圖像的版本。舉例而言，在當前區塊之樣本作為初始化值儲存於參考圖像記憶體64中的情況下，運動補償單元46及/或求和器62可藉由當前區塊之經重建構樣本更新參考圖像記憶體64儲存之當前圖像之當前的樣本。

濾波單元(未展示)可執行各種濾波處理。舉例而言，濾波單元可執行解區塊。亦即，濾波單元可接收形成經重建構視訊之圖塊或圖框的複數個經重建構視訊區塊，且對區塊邊界進行濾波，以自圖塊或圖框移除區塊效應假影。在一項實例中，濾波單元評估視訊區塊之所謂的「邊界強度」。基於視訊區塊之邊界強度，可相對於鄰近視訊區塊之邊緣像素對視訊區塊之邊緣像素進行濾波，使得檢視器更難以感知自一個視訊區塊之變換。

在一些實例中，運動補償單元46及/或求和器62可在濾波對樣本執行濾波(例如，解區塊及/或SAO)之前更新由參考圖像記憶體64儲存之當前圖像的版本。舉例而言，濾波單元可在應用濾波之前等待直至整個圖像經寫碼。以此方式，運動估計單元44可在應用濾波之前將當前圖像用作參考。在一些實例中，濾波單元可在由參考圖像記憶體64儲存之當前圖像之版本經更新時執行濾波。舉例而言，濾波單元可在每一區塊經更新時應用濾波。以此方式，運動估計單元44可在應用濾波之後將當前圖像用作參考。

雖然在本發明中描述該等技術之多個不同態樣及實例，但是該等技術之各種態樣及實例可以一起執行或彼此單獨地執行。換言

之，該等技術不應嚴格地限制於上文所描述的各種態樣及實例，而是可組合使用或一起執行及/或單獨地執行。另外，雖然某些技術可歸於視訊編碼器20之特定單元(例如，框內預測單元48、運動補償單元46或熵編碼單元56)，但是應理解，視訊編碼器20之一或多個其他單元亦可負責執行此類技術。

圖5為根據本發明之技術的可經組態以執行CABAC之實例熵編碼單元56之方塊圖。將語法元素118輸入至熵編碼單元56中。若語法元素已為二進位值語法元素(例如，僅具有值0及1之旗標或其他語法元素)，則可跳過二進位化步驟。若語法元素為非二進位值語法元素(例如，可具有除1或0以外之值的語法元素)，則藉由二進位化器120對非二進位值語法元素進行二進位化。二進位化器120執行非二進位值語法元素至二進位決策序列之映射。此等二進位決策通常被稱作「位元子」。舉例而言，對於變換係數層級，可將層級之值分解為連續位元子，每一位元子指示係數層級之絕對值是否大於某一值。舉例而言，位元子0 (有時被稱作有效旗標)指示變換係數層級之絕對值是否大於0。位元子1指示變換係數層級之絕對值是否大於1，等等。可針對每一非二進位值語法元素產生唯一映射。

由二進位化器120產生之每一位元子饋入至熵編碼單元56之二進位算術寫碼側。亦即，針對一組預定非二進位值語法元素，每一位元子類型(例如，位元子0)在下一位元子類型(例如，位元子1)之前經寫碼。可以常規模式或旁路模式執行寫碼。在旁路模式中，旁路寫碼引擎126使用固定機率模型(例如，使用哥倫布萊斯或指數哥倫布寫碼)執行算術寫碼。旁路模式大體上用於更可預測語法元素。

以常規模式寫碼涉及執行CABAC。在給定先前經寫碼位元子之值則位元子之值之機率為可預測的情況下，常規模式CABAC用於寫碼位元子值。由上下文建模器122判定為LPS之位元子的機率。上下

文建模器122針對上下文輸出位元子值及機率狀態(例如，機率狀態σ，包括LPS之值及LPS出現之機率)。上下文可為用於一系列位元子之初始上下文，或可基於先前經寫碼位元子之經寫碼值判定該上下文。如上文所述，上下文建模器122可基於所接收位元子為MPS抑或LPS而更新狀態。在藉由上下文建模器122判定上下文及機率狀態σ之後，常規寫碼引擎124對位元子值執行BAC。

根據本發明之一或多種技術，與在二進位算術寫碼過程中使用用於更新機率狀態之變數(例如，視窗大小、或比例因子(α)或固定機率更新速度中之一或多者)的相同值相反，熵編碼單元56可針對不同上下文及/或不同語法元素使用不同變數值。舉例而言，上下文建模器122可針對複數個上下文中之上下文判定在二進位算術寫碼過程中用於更新機率狀態之變數的值，且基於所判定之值更新機率狀態。

在一些實例中，由上下文建模器122使用以判定下一機率狀態之視窗大小可取決於上下文製得。舉例而言，上下文建模器122可將不同視窗大小用於不同上下文。作為一項實例，上下文建模器122可判定用於複數個上下文中之第一上下文的第一第一視窗大小，且判定用於複數個上下文中之第二上下文的不同於第一視窗大小之第二視窗大小。

在一些實例中，當取決於更新方法將上述上下文併入至基於計數器之算術寫碼器(諸如，JCTVC-F254及JCTV-G764中)時，視窗大小之值可取決於上下文。另外，除了來自方程式(4)之機率Pi之外，每一上下文可進一步與視窗大小相關聯。

在一些實例中，上下文建模器122可使用可等於2^M之視窗大小W，其中M可為正整數。因此，每一上下文可具有可不同於其他上下文之其自身M值，但某些上下文模型可具有相同M值。

在一些實例中，上下文建模器122可自視窗大小之預定義組判定

視窗大小。某些實例預定義視窗大小為16、32、64及128，但預期其他視窗大小。舉例而言，可預定義一組可能的M值，例如，M可包含性地在4至7之範圍內。在一些實例中，上下文建模器122可產生待在圖塊標頭或參數集中傳信之該組可能的視窗大小之指示(例如，一組可能的M值之指示)，該參數集包括圖像參數集、主動參數集、序列參數集或視訊參數集。

在一些實例中，可預定義與每一上下文相關聯之視窗大小(例如，M之值)。在一些實例中，視窗大小可進一步取決於圖塊類型及/或時間識別符(例如，在HEVC中被稱作temporalId)。在一些實例中，視窗大小可進一步取決於圖像類型(或NAL單元類型)，例如，圖像是否為隨機存取圖像。

在一些實例中，上下文建模器122可使得與每一上下文相關聯的視窗大小(例如，M之值)在位元串流中(諸如，在圖塊標頭/圖像參數集/主動參數集/序列參數集中)傳信。舉例而言，可首先預定義用於每一上下文之預設視窗大小。對於每一各別上下文模型，上下文建模器122可編碼指示預設視窗大小是否用於各別上下文之各別語法元素(例如，旗標)。若預設視窗大小不用於各別上下文，則上下文建模器122可基於預設視窗大小有區別地編碼實際使用之視窗大小。在一些實例中，上下文建模器122可將所有上下文之語法元素(亦即，指示是否使用預設視窗大小)組織在一起，且利用延行長度寫碼對此等語法元素進行寫碼。在一些實例中，上下文建模器122可在寫碼實際使用之視窗大小與預設視窗大小之間的差時利用映射表。舉例而言，在預設M值等於6的情況下，可能的M值為4、5、6及7。可將映射表定義為：

實際M值	4	5	6	7
待寫碼值	0	1	-	2

在一些實例中，上下文建模器122可直接寫碼用於每一上下文之實際視窗大小與預設視窗大小之間的差。舉例而言，在預設M值為4的情況下，上下文建模器122可針對每一上下文寫碼M-4。

在一些實例中，上下文建模器122可寫碼指示用於當前圖塊中之上下文的所有視窗大小相對於先前經寫碼圖塊中之對應上下文是否為繼承(亦即，設定等於)視窗大小的第一語法元素。在一項實例中，可將「先前經解碼圖塊」定義為具有與當前圖塊及/或相同初始化量化參數相同之圖塊類型、或相同圖塊類型及量化參數兩者或相同圖塊類型及時間層兩者的先前經寫碼圖塊。在一些實例中，可要求先前圖塊屬於存在於DPB中之圖像且可作為參考圖像用於當前圖像，詳言之，如在基於HEVC之平台中，可要求先前圖塊屬於參考圖像集(RPS)中之圖像，或甚至為RPS之以下子集中之一者中的圖像：RefPicSetStCurrBefore、RefPicSetStCurrAfter及RefPicSetLtCurr。

在一些實例中，上下文建模器122可寫碼指示預設視窗大小是否用於複數個上下文(例如，在當前圖塊中)之第一語法元素。在預設視窗大小不用於複數個上下文的情況下，上下文建模器122可寫碼指示上下文之視窗大小的第二語法元素。舉例而言，上下文建模器122可寫碼指示上下文之視窗大小與預設視窗大小之間的差的第二語法元素。

在另一實例中，上下文建模器122可(例如)基於來自先前圖塊或圖像之經寫碼資訊導出視窗大小。舉例而言，上下文建模器122可在與一個上下文相關聯之先前圖塊中追蹤經寫碼位元子。對於可能的視窗大小之每一候選者，上下文建模器122可獲得為寫碼此等位元子而消耗之位元，且選擇導致用於寫碼此等位元子之最小位元作為用於此上下文之視窗大小的視窗大小。上下文建模器122可將所選視窗大小用於寫碼後續圖塊/圖像。

在一些實例中，用於判定算術寫碼器中之下一機率狀態或機率更新速度之『視窗大小』可為語法元素特定的，例如，在上下文在不同語法元素當中共用的情況下。舉例而言，當使用上下文編碼語法元素之位元子時，上下文建模器122可基於語法元素判定上下文之視窗大小。作為一項實例，用於在對寫碼單元分裂語法元素及寫碼單元跳過旗標語法元素之位元子進行寫碼時更新上下文之狀態的視窗大小可為相同的，例如，16 (亦即，M=4)。

根據本發明之一或多種技術，上下文建模器122可在熵編碼供視訊解碼器30解碼視訊資料時使用之資料(例如，表示一維向量及/或其他語法元素之語法元素)時自適應地判定不同視窗大小。舉例而言，對於每一上下文，上下文建模器122可藉由不同視窗大小計算寫碼所記錄位元子串之位元，且選擇具有最小位元之一者。在視窗大小係選自視窗大小之預定義集合的情況下，上下文建模器122可針對視窗大小之預定義集合中之各別視窗大小判定用於藉由上下文編碼位元子串之各別位元量，且選擇視窗大小之預定義集合中對應於為上下文之視窗大小的最小位元量之視窗大小。

在一些實例中，上述技術可適用於特定上下文。亦即，上下文之子集可使用經更新『視窗大小』而非預設視窗大小。在一些實例中，上述技術可適用於特定圖塊類型。

返回至圖4，在一些情況下，熵編碼單元56或視訊編碼器20之另一單元可經組態以執行除熵寫碼之外的其他寫碼功能。舉例而言，熵編碼單元56可經組態以判定用於CU及PU之寫碼區塊型樣(CBP)值。又，在一些情況下，熵編碼單元56可執行係數之延行長度寫碼。另外，熵編碼單元56或其他處理單元亦可寫碼其他資料(諸如，量化矩陣之值)。

如上文所論述，逆量化單元58及逆變換處理單元60分別應用逆

量化及逆變換以重建構像素域中之殘餘區塊(例如，供稍後用作參考區塊)。運動補償單元46可藉由將該殘餘區塊添加至參考圖框記憶體64之圖框中之一者的預測性區塊來計算參考區塊。運動補償單元46亦可將一或多個內插濾波器應用於經重建構殘餘區塊以計算用於次整數像素值以用於運動估計。求和器62將經重建構殘餘區塊添加至由運動補償單元46產生之運動補償預測區塊，以產生用於儲存於參考圖框記憶體64中之經重建構視訊區塊。經重建構視訊區塊可由運動估計單元44及運動補償單元46用作參考區塊，以對後續視訊圖框中之區塊進行框間寫碼。

圖6為繪示可實施本發明中所描述之技術的視訊解碼器30之實例的方塊圖。又，將出於說明之目的在HEVC寫碼之上下文中描述視訊解碼器30，但關於其他寫碼標準並不限制本發明。此外，視訊解碼器30可經組態以根據範圍擴展實施技術。

在圖6之實例中，視訊解碼器30可包括視訊資料記憶體69、熵解碼單元70、預測處理單元71、逆量化處理單元76、逆變換處理單元78、求和器80及參考圖像記憶體82。預測處理單元71包括運動補償單元72及框內預測單元74。在一些實例中，視訊解碼器30可執行與關於來自圖4之視訊編碼器20所描述之編碼遍次大體上互逆的解碼遍次。

視訊資料記憶體69可儲存待由視訊解碼器30之組件解碼之視訊資料(諸如，經編碼視訊位元串流)。可(例如)自儲存器件34、自本端視訊源(諸如，攝影機)經由視訊資料之有線或無線網路通信或藉由存取實體資料儲存媒體而獲得儲存於視訊資料記憶體69中之視訊資料。視訊資料記憶體69可形成儲存來自經編碼視訊位元串流之經編碼視訊資料的經寫碼圖像緩衝器(CPB)。

參考圖像記憶體82為儲存供視訊解碼器30 (例如，以框內或框間

寫碼模式)解碼視訊資料時使用之參考視訊資料的經解碼圖像緩衝器(DPB)之一項實例。視訊資料記憶體69及參考圖像記憶體82可由多種記憶體器件中之任一者形成，諸如動態隨機存取記憶體(DRAM) (包括同步DRAM (SDRAM))、磁阻式RAM (MRAM)、電阻式RAM (RRAM)或其他類型之記憶體器件。視訊資料記憶體69及參考圖像記憶體82可由同一記憶體器件或單獨記憶體器件提供。在各種實例中，視訊資料記憶體69可與視訊解碼器30之其他組件一起在晶片上，或相對於彼等組件在晶片外。

在解碼過程期間，視訊解碼器30自視訊編碼器20接收表示經編碼視訊圖塊之視訊區塊及相關聯語法元素的經編碼視訊位元串流。視訊解碼器30之熵解碼單元70熵解碼位元串流以產生經量化係數、運動向量或框內預測模式指示符及其他語法元素。在一些實例中，熵解碼單元70可應用與編碼器所使用之過程大體上互逆的過程。熵解碼單元70對經編碼位元串流執行熵解碼過程，以擷取變換係數之一維陣列。所使用之熵解碼過程取決於視訊編碼器20使用之熵寫碼(例如，CABAC、CAVLC、PIPE或上文所描述的其他過程)。根據本發明中所描述之該等技術，熵解碼單元70可如本發明中所描述應用BAC過程(例如，在CABAC過程內)。編碼器使用之熵寫碼過程中之視窗大小可在經編碼位元串流中傳信或可為預定過程。

熵解碼單元70將運動向量及其他語法元素轉遞至運動補償單元72。視訊解碼器30可接收視訊圖塊層級及/或視訊區塊層級之語法元素。

圖7為根據本發明之技術的可經組態以執行CABAC的實例熵解碼單元70之方塊圖。圖7之熵解碼單元70以與圖5中所描述之熵編碼單元56之彼方式互逆的方式執行CABAC。將來自位元串流218之經寫碼位元輸入至熵解碼單元70中。基於經寫碼位元是使用旁路模式

抑或常規模式經熵寫碼而將其饋入至上下文建模器220或旁路寫碼引擎222。若經寫碼位元以旁路模式寫碼，則旁路解碼引擎將使用哥倫布萊斯或指數哥倫布解碼(例如)以擷取非二進位語法元素之二進位值語法元素或位元子。

若經寫碼位元以常規模式寫碼，則上下文建模器220可判定經寫碼位元之機率模型，且常規解碼引擎224可解碼該等經寫碼位元以產生非二進位值語法元素之位元子(或語法元素自身(在語法元素為二進位值時))。在藉由上下文建模器220判定上下文及機率狀態σ之後，常規解碼引擎224執行BAC以解碼位元子值。換言之，常規解碼引擎224可判定上下文之機率狀態，且基於先前經寫碼位元子及當前範圍解碼位元子值。在解碼位元子之後，上下文建模器220可基於視窗大小及經解碼位元子之值更新上下文之機率狀態。

根據本發明之一或多種技術，與在二進位算術寫碼過程中使用用於更新機率狀態之變數(例如，視窗大小、或比例因子(α)及固定機率更新速度中的一或多者)的相同值相反，熵編碼單元56可針對不同上下文及/或不同語法元素使用不同變數值。舉例而言，上下文建模器220可針對複數個上下文中之上下文判定用於在二進位算術寫碼過程中更新機率狀態之變數的值，且基於所判定之值更新機率狀態。

在一些實例中，由上下文建模器220使用以判定下一機率狀態之視窗大小可取決於上下文製得。舉例而言，上下文建模器220可將不同視窗大小用於不同上下文。作為一項實例，上下文建模器220可判定用於複數個上下文中之第一上下文的第一第一視窗大小，且判定用於複數個上下文中之第二上下文的不同於第一視窗大小之第二視窗大小。

在一些實例中，當取決於更新方法將上述上下文模型併入至基於計數器之算術寫碼器中(諸如，JCTVC-F254及JCTV-G764中)時，

視窗大小之值可取決於上下文。另外，除了來自方程式(4)之機率Pi之外，每一上下文可進一步與視窗大小相關聯。

在一些實例中，上下文建模器220可使用可等於2^M之視窗大小W，其中M可為正整數。因此，每一上下文可具有可不同於其他上下文之其自身M值，但某些上下文可具有相同M值。

在一些實例中，上下文建模器220可自視窗大小之預定義集合判定視窗大小。舉例而言，可預定義一組可能的M值，例如，M可包含性地在4至7之範圍內。在一些實例中，熵解碼單元70可自圖塊標頭或參數集解碼該組可能的視窗大小之指示(例如，一組可能的M值之指示)，該參數集包括圖像參數集、主動參數集、序列參數集或視訊參數集。

在一些實例中，可預定義與每一上下文相關聯之視窗大小(例如，M之值)。在一些實例中，視窗大小可進一步取決於圖塊類型及/或時間識別符(例如，在HEVC中被稱作temporalId)。在一些實例中，視窗大小可進一步取決於圖像類型(或NAL單元類型)，例如，圖像是否為隨機存取圖像。

在一些實例中，熵解碼單元70可自位元串流(諸如，在圖塊標頭/圖像參數集/主動參數集/序列參數集中)解碼與每一上下文相關聯之視窗大小(例如，M之值)。舉例而言，可首先預定義用於每一上下文之預設視窗大小。對於每一各別上下文，熵解碼單元70可解碼指示預設視窗大小是否用於各別上下文之各別語法元素(例如，旗標)。若預設視窗大小不用於各別上下文，則熵解碼單元70可基於預設視窗大小有區別地解碼實際使用之視窗大小。在一些實例中，可將所有上下文之語法元素(亦即，指示是否使用預設視窗大小)組織在一起，且熵解碼單元70可利用延行長度寫碼對此等語法元素進行解碼。在一些實例中，上下文建模器220可在寫碼實際使用之視窗大小

與預設視窗大小之間的差時利用映射表。舉例而言，在預設M值等於6的情況下，可能的M值為4、5、6及7。可將映射表定義為：

實際M值	4	5	6	7
待寫碼值	0	1	-	2

在一些實例中，熵解碼單元70可直接解碼用於上下文之實際視窗大小與預設視窗大小之間的差。舉例而言，在預設M值為4的情況下，熵解碼單元70可針對每一上下文解碼M-4之值。

在一些實例中，熵解碼單元70可解碼指示用於當前圖塊中之上下文的所有視窗大小相對於先前經寫碼圖塊中之對應上下文是否為繼承(亦即，設定等於)視窗大小的第一語法元素。在一項實例中，可將「先前經解碼圖塊」定義為具有與當前圖塊及/或相同初始化量化參數相同之圖塊類型、或相同圖塊類型及量化參數兩者或相同圖塊類型及時間層兩者的先前經寫碼圖塊。在一些實例中，可要求先前圖塊屬於存在於DPB中之圖像且可作為參考圖像用於當前圖像，詳言之，如在基於HEVC之平台中，可要求先前圖塊屬於參考圖像集(RPS)中之圖像，或甚至為RPS之以下子集中之一者中的圖像：RefPicSetStCurrBefore、RefPicSetStCurrAfter及RefPicSetLtCurr。

在一些實例中，熵解碼單元70可解碼指示預設視窗大小是否用於複數個上下文(例如，在當前圖塊中)之第一語法元素。在預設視窗大小不用於複數個上下文的情況下，熵解碼單元70可解碼指示上下文之視窗大小的第二語法元素。舉例而言，熵解碼單元70可解碼指示上下文之視窗大小與預設視窗大小之間的差的第二語法元素。

在另一實例中，熵解碼單元70可(例如)基於來自先前圖塊或圖像之經寫碼資訊導出視窗大小。舉例而言，熵解碼單元70可追蹤與經跟蹤之一個上下文相關聯的先前圖塊中之經解碼位元子。對於可能的視窗大小之每一候選者，熵解碼單元70可獲得為寫碼此等位元

子而消耗之位元。熵解碼單元70可選擇導致用於寫碼此等位元子之最小位元作為用於此上下文之視窗大小的視窗大小。熵解碼單元70可將所選視窗大小用於解碼後續圖塊/圖像。

在一些實例中，用於判定算術寫碼器中之下一機率狀態或機率更新速度之『視窗大小』可為語法元素特定的。舉例而言，當使用上下文編碼語法元素之位元子時，上下文建模器220可基於語法元素類型判定上下文之視窗大小。作為一項實例，用於在對寫碼單元分裂語法元素及寫碼單元跳過旗標語法元素之位元子進行寫碼時更新上下文的視窗大小可為相同的，例如，16 (亦即，M=4)。

在一些實例中，上述技術可適用於特定上下文。亦即，上下文之子集可使用經更新『視窗大小』而非預設視窗大小。在一些實例中，上述技術可適用於特定圖塊類型。

在常規解碼引擎224解碼位元子之後，反向二進位化器230可執行將位元子反向轉換為非二進位值語法元素之值的反向映射。

返回至圖6，在一些實例中，熵解碼單元70 (或逆量化單元76)可使用視訊編碼器20之熵編碼單元56 (或量化單元54)所使用之掃描模式呈鏡射的掃描來掃描所接收值。儘管可在逆量化單元76中執行係數之掃描，但出於說明之目的，將描述為由熵解碼單元70執行掃描。另外，儘管為易於說明展示為獨立功能單元，但熵解碼單元70、逆量化單元76及視訊解碼器30之其他單元的結構及功能性可彼此高度整合。

逆量化單元76逆量化(亦即，解量化)位元串流中所提供且由熵解碼單元70解碼之經量化變換係數。逆量化過程可包括習知過程，例如類似於HEVC或由H.264解碼標準所定義之一些實例。逆量化過程可包括將由視訊編碼器20計算之量化參數QP用於CU以判定量化程度，且同樣地，應應用之逆量化的程度。逆量化單元76可在將係數

自一維陣列轉換為二維陣列之前抑或之後逆量化變換係數。

逆變換處理單元78將逆變換應用於經逆量化變換係數。在一些實例中，逆變換處理單元78可基於來自視訊編碼器20之傳信，或藉由自一或多個寫碼特性(諸如區塊大小、寫碼模式或類似者)推斷變換來判定逆變換。在一些實例中，逆變換處理單元78可基於包括當前區塊之LCU的四分樹之根節點處的所傳信變換判定適用於當前區塊之變換。替代地，可針對LCU四分樹中之葉節點CU在TU四分樹之根處傳信變換。在一些實例中，逆變換處理單元78可應用級聯逆變換，其中逆變換處理單元78將兩個或兩個以上逆變換應用於正經解碼之當前區塊的變換係數。

另外，逆變換處理單元可根據本發明之上述技術應用逆變換以產生變換單元分區。

框內預測處理單元74可基於所傳信之框內預測模式及來自當前圖框之先前經解碼區塊的資料產生用於當前圖框之當前區塊的預測資料。基於所擷取之運動預測方向、參考圖框索引及所計算之當前運動向量(例如，根據合併模式自相鄰區塊複製之運動向量)，運動補償單元產生當前部分之運動補償區塊。此等運動補償區塊本質上重建用於產生殘餘資料之預測性區塊。

運動補償單元72可產生運動補償區塊，可能基於內插濾波器執行內插。具有子像素精度之待用於運動估計的內插濾波器之識別符可包括於語法元素中。運動補償單元72可使用如由視訊編碼器20在視訊區塊之編碼期間所使用的內插濾波器，以計算參考區塊之子整數像素的內插值。運動補償單元72可根據所接收語法資訊判定視訊編碼器20使用之內插濾波器，且使用該等內插濾波器以產生預測性區塊。

另外，在HEVC實例中，運動補償單元72及框內預測處理單元74

可使用語法資訊(例如，由四分樹提供)中之一些，以判定用以對經編碼視訊序列之一或多個圖框進行編碼的LCU之大小。運動補償單元72及框內預測處理單元74亦可使用語法資訊，以判定描述經編碼視訊序列之圖框的每一CU如何分裂(且同樣地，子CU如何分裂)的分裂資訊。語法資訊亦可包括指示每一分裂如何經編碼之模式(例如，框內或框間預測，及針對框內預測之框內預測編碼模式)、用於每一經框間編碼PU之一或多個參考圖框(及/或含有參考圖框之識別符的參考清單)及對經編碼視訊序列進行解碼之其他資訊。

求和器80使殘餘區塊與由運動補償單元72或框內預測處理單元74產生之對應預測區塊組合以形成經解碼區塊。必要時，亦可應用解塊濾波器來對經解碼區塊進行濾波以便移除區塊效應假影。接著，將經解碼視訊區塊儲存於參考圖像記憶體82中，參考圖像記憶體提供用於後續運動補償之參考區塊，且亦產生用於呈現於顯示器件(諸如圖1之顯示器件31)上的經解碼視訊。

圖8繪示針對給定位元子值*binVal*使用常規寫碼模式之二進位算術編碼過程。算術編碼引擎之內部狀態照常表徵為兩個量：當前寫碼區間之當前區間範圍R及基礎(較低端點) L。然而，應注意，將此等暫存器儲存於CABAC引擎中(以常規及旁路模式兩者)所需之精度可分別降低達至9個位元及10個位元。在如下之四個基礎步驟序列中執行在具有機率狀態索引δ及MPS之值(δ%2)的上下文中觀測到的給定二進位值*binVal*的編碼。

在第一且主要步驟中，根據給定機率估計再分當前區間。此區間再分過程涉及如圖8中之流程圖之最頂端框中所展示的三個基礎操作。首先，藉由經量化值$Q(R)$使用整個範圍$2^8 \leq R \leq 2^9$至四個小區之等分割來估計當前區間範圍R。而不是明確地使用CABAC引擎中之對應代表性經量化範圍值Q_0、Q_1、Q_2及Q_3，僅由其量化器索引ρ定

址，該量化器索引可藉由移位及位元掩碼操作之組合而有效計算(亦即，根據以下方程式(14))。

$$\rho = (R \gg 6)\ \&\ 3 \qquad \textbf{(14)}$$

接著，將此索引ρ及機率狀態索引δ用作2-D表TabRangeLPS中之條目，以判定(大致)LPS相關之子區間範圍R_{LPS}，如圖8中所展示。在此，針對8位元精度中之$0 \le (\delta \gg 1) \le 63$及$0 \le \rho \le 3$，表TabRangeLPS含有$p_\sigma \cdot Q_\rho$之所有64×4預先計算之產品值。

給定MPS之雙子區間範圍，在編碼過程之第二步驟中選擇對應於給定位元子值*binVal*之子區間。若*binVal*等於MPS值，則選擇較低子區間，以使得L不變(圖8中分枝之右方路徑)；否則，選擇具有等於R_{LPS}之範圍的上部子區間(圖8中之左方分枝)。在常規算術編碼過程之第三步驟中，如上文所述(例如，使用方程式(2))執行機率狀態之更新(圖8中之灰色陰影框)，且最後，第四步驟如Marpe所描述由暫存器L及R之再歸一化構成(圖8中之「RenormE」框)。

可將2-D表TabRangeLPS定義如下：

TabRangeLPS[64][4] =

{

{ 128, 176, 208, 240},

{ 128, 167, 197, 227},

{ 128, 158, 187, 216},

{ 123, 150, 178, 205},

{ 116, 142, 169, 195},

{ 111, 135, 160, 185},

{ 105, 128, 152, 175},

{ 100, 122, 144, 166},

{ 95, 116, 137, 158},

{ 90, 110, 130, 150},

{ 85, 104, 123, 142},

{ 81, 99, 117, 135},

{ 77, 94, 111, 128},

{ 73, 89, 105, 122},

{ 69, 85, 100, 116},

{ 66, 80, 95, 110},

{ 62, 76, 90, 104},

{ 59, 72, 86, 99},

{ 56, 69, 81, 94},

{ 53, 65, 77, 89},

{ 51, 62, 73, 85},

{ 48, 59, 69, 80},

{ 46, 56, 66, 76},

{ 43, 53, 63, 72},

{ 41, 50, 59, 69},

{ 39, 48, 56, 65},

{ 37, 45, 54, 62},

{ 35, 43, 51, 59},

{ 33, 41, 48, 56},

{ 32, 39, 46, 53},

{ 30, 37, 43, 50},

{ 29, 35, 41, 48},

{ 27, 33, 39, 45},

{ 26, 31, 37, 43},

{ 24, 30, 35, 41},

{ 23, 28, 33, 39},
{ 22, 27, 32, 37},
{ 21, 26, 30, 35},
{ 20, 24, 29, 33},
{ 19, 23, 27, 31},
{ 18, 22, 26, 30},
{ 17, 21, 25, 28},
{ 16, 20, 23, 27},
{ 15, 19, 22, 25},
{ 14, 18, 21, 24},
{ 14, 17, 20, 23},
{ 13, 16, 19, 22},
{ 12, 15, 18, 21},
{ 12, 14, 17, 20},
{ 11, 14, 16, 19},
{ 11, 13, 15, 18},
{ 10, 12, 15, 17},
{ 10, 12, 14, 16},
{ 9, 11, 13, 15},
{ 9, 11, 12, 14},
{ 8, 10, 12, 14},
{ 8, 9, 11, 13},
{ 7, 9, 11, 12},
{ 7, 9, 10, 12},
{ 7, 8, 10, 11},
{ 6, 8, 9, 11},

```
    {   6,   7,   9,  10},
    {   6,   7,   8,   9},
    {   2,   2,   2,   2}
};
```

可在HEVC標準之章節9.3.4.3.2.2中發現實例CABAC解碼過程。

圖9為繪示基於殘餘四分樹之變換方案的概念圖。為調適殘餘區塊之不同特性，將使用殘餘四分樹(RQT)之變換寫碼結構應用於HEVC中，該HEVC經簡要地描述於http://www.hhi.fraunhofer.de/departments/video-coding-analytics/research-groups/image-video-coding/hevc-high-efficiency-video-coding/transform-coding-using-the-residual-quadtree-rqt.html。

將每一圖像劃分為寫碼樹型單元(CTU)，該等寫碼樹型單元以光柵掃描次序來寫碼以用於特定圖像塊或圖塊。CTU為方形區塊且表示四分樹(亦即，寫碼樹)之根。CTU大小可在8×8至64×64明度樣本範圍內，但通常使用64×64。每一CTU可進一步分裂成被稱作寫碼單元(CU)之更小方形區塊。在CTU遞歸地分裂成CU之後，將每一CU進一步劃分為PU及TU。基於四分樹方法以遞歸方式進行將CU分割成TU，因此每一CU之剩餘信號藉由樹型結構(即，剩餘四分樹(RQT))來寫碼。RQT允許自4×4達32×32明度樣本之TU大小。圖9展示其中CU包括10個TU (標記有字母a至j)及對應區塊分割之實例。RQT之每一節點實際上為變換單元(TU)。以深度優先樹遍歷次序處理個別TU，該次序在圖式中繪示為字母表次序，該次序之後為具有深度優先遍歷之遞歸Z掃描。四分樹方法使得能夠將調適變換至殘餘信號之變化空間頻率特性。通常，具有更大空間支援之更大變換區塊大小提供更佳頻率解析度。然而，具有更小空間支援之更小變換區塊大小提供更佳空間解析度。(例如)基於率失真最佳化技術藉由編碼器

模式決策選擇兩個(空間及頻率)解析度之間的折衷。率失真最佳化技術針對每一寫碼模式(例如，特定RQT分裂結構)計算寫碼位元及重建構失真之加權總和(亦即，率失真成本)，且選擇具有最小率失真成本之寫碼模式作為最佳模式。

三個參數定義於RQT中：樹之最大深度、最小允許之變換大小及最大允許之變換大小。在HEVC之一些實例中，最小及最大變換大小可在自4×4至32×32樣本範圍內改變，該範圍對應於先前段落中提及之所支援區塊變換。RQT之最大允許深度限制TU之數目。最大深度等於零意謂：若每一所包括TU達到最大允許變換大小(例如，32×32)，則不能更進一步分裂CTU。

所有此等參數與RQT結構互相作用且影響RQT結構。考慮根CTU大小為64×64，最大深度等於零且最大變換大小等於32×32的情況。在此情況下，必須分割CTU至少一次，因為否則的話其將產生不被允許的64×64 TU。RQT參數(亦即，最大RQT深度、最小及最大變換大小)於序列參數集層級處在位元串流中傳輸。考慮RQT深度，可針對經框內及框間寫碼CU指定且傳信不同值。

將四分樹變換應用於框內殘餘區塊及框間殘餘區塊兩者。通常，將相同大小之當前殘餘四分樹分區之DCT-II變換應用於殘餘區塊。然而，若當前殘餘四分樹區塊為4×4且由框內預測產生，則應用上述4×4 DST-VII變換。

在HEVC中，不採用更大的大小變換(例如，64×64變換)，主要因為考慮到其受限的益處及對於相對較小解析度視訊之相對較高的複雜度。

圖10為繪示基於係數群之實例係數掃描的概念圖。無論TU大小，藉由非重疊係數群組(CG)來寫碼變換單元之殘餘，且每一者含有TU之4×4區塊之係數。舉例而言，32×32 TU總共具有64個CG，且

16×16 TU總共具有16個CG。可根據特定預定義掃描次序寫碼TU內部之CG。當寫碼每一CG時，根據用於4×4區塊之特定預定義掃描次序掃描及寫碼當前CG內部之係數。圖10繪示用於含有4個CG之8×8 TU的係數掃描。

將語法元素表定義如下：

7.3.8.11 殘餘寫碼語法

residual_coding(x0, y0, log2TrafoSize, cIdx) {	描述符
若(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize = = 2))	
transform_skip_flag[x0][y0][cIdx]	ae(v)
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
若(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
若(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize − 2)) * (1 << (log2TrafoSize − 2)) − 1	
do {	
若(lastScanPos = = 0) {	
lastScanPos = 16	
lastSubBlock− −	
}	
lastScanPos− −	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} 當((xC != LastSignificantCoeffX) | (yC != LastSignificantCoeffY))	
對於(i = lastSubBlock; i >= 0; i− −) {	
xS = ScanOrder[log2TrafoSize − 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize − 2][scanIdx][i][1]	
inferSbDcSigCoeffFlag = 0	
若((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag[xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
對於(n = (i = = lastSubBlock) ? lastScanPos − 1 : 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
若 (coded_sub_block_flag[xS][yS] && (n > 0 | !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag[xC][yC]	ae(v)
若(sig_coeff_flag[xC][yC])	

inferSbDcSigCoeffFlag = 0	
}	
}	
firstSigScanPos = 16	
lastSigScanPos = −1	
numGreater1Flag = 0	
lastGreater1ScanPos = −1	
對於(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
若(sig_coeff_flag[xC][yC]) {	
若(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
若(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos == −1)	
lastGreater1ScanPos = n	
}	
若(lastSigScanPos == −1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	
signHidden = (lastSigScanPos − firstSigScanPos > 3 && !cu_transquant_bypass_flag)	
若(lastGreater1ScanPos != −1)	
coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)
對於(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
若(sig_coeff_flag[xC][yC] && (!sign_data_hiding_enabled_flag || !signHidden || (n != firstSigScanPos)))	
coeff_sign_flag[n]	ae(v)
}	
numSigCoeff = 0	
sumAbsLevel = 0	
對於(n = 15; n >= 0; n− −) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
若(sig_coeff_flag[xC][yC]) {	
baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
若(baseLevel == ((numSigCoeff < 8) ? ((n == lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])	
若(sign_data_hiding_enabled_flag && signHidden) {	

sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
若((n == firstSigScanPos) && ((sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cIdx][xC][yC] = −TransCoeffLevel[x0][y0][cIdx][xC][yC]	
}	
numSigCoeff++	
}	
}	
}	
}	

對於每一色彩分量，可首先傳信一個旗標以指示當前TU是否具有至少一個非零係數。若存在至少一個非零係數，則接著藉由相對於變換單元之左上角的座標明確地寫碼在TU中之係數掃描次序中的最末有效係數的位置。座標之垂直或水平分量由其前綴及後綴表示，其中前綴藉由截斷萊斯(TR)二進位化且後綴藉由固定長度二進位化。

語義：

last_sig_coeff_x_prefix指定變換區塊內之掃描次序中之最末有效係數之行位置的前綴。last_sig_coeff_x_prefix之值應包含性地在0至(log2TrafoSize << 1)−1的範圍內。

last_sig_coeff_y_prefix指定變換區塊內之掃描次序中之最末有效係數之列位置的前綴。last_sig_coeff_y_prefix之值應包含性地在0至(log2TrafoSize << 1)−1的範圍內。

last_sig_coeff_x_suffix指定變換區塊內之掃描次序中之最末有效係數之行位置的後綴。last_sig_coeff_x_suffix之值應包含性地在0至(1 << ((last_sig_coeff_x_prefix >> 1) -1)) -1的範圍內。

變換區塊LastSignificantCoeffX內之掃描次序中之最末有效係數之行位置經導出如下：

- 若不存在last_sig_coeff_x_suffix，則下文應用：

LastSignificantCoeffX = last_sig_coeff_x_prefix

- 否則的話(存在last_sig_coeff_x_suffix)，下文應用：

LastSignificantCoeffX = (1 << ((last_sig_coeff_x_prefix >> 1) - 1)) * (2 + (last_sig_coeff_x_prefix & 1)) + last_sig_coeff_x_suffix

last_sig_coeff_y_suffix指定變換區塊內之掃描次序中之最末有效係數之列位置的後綴。last_sig_coeff_y_suffix之值應包含性地在0至(1 << ((last_sig_coeff_y_prefix >> 1) -1)) -1的範圍內。

變換區塊LastSignificantCoeffY內之掃描次序中之最末有效係數之列位置經導出如下：

- 若不存在last_sig_coeff_y_suffix，則下文應用：

LastSignificantCoeffY = last_sig_coeff_y_prefix

- 否則的話(存在last_sig_coeff_y_suffix)，下文應用：

LastSignificantCoeffY = (1 << ((last_sig_coeff_y_prefix >> 1) - 1)) * (2 + (last_sig_coeff_y_prefix & 1)) + last_sig_coeff_y_suffix

當scanIdx等於2時，座標被調換如下：

(LastSignificantCoeffX, LastSignificantCoeffY)=Swap(LastSignificantCoeffX, LastSignificantCoeffY)

藉由經寫碼之此位置亦及CG之係數掃描次序，進一步傳信一個旗標以用於除(掃描次序中之)最末CG外之CG，該旗標指示其是否含有非零係數。

CG旗標之上下文建模。當寫碼一個CG是否具有非零係數(亦即，CG旗標(HEVC規格中之**coded_sub_block_flag**)時，利用相鄰CGs之資訊以建置上下文。為更具體，將用於寫碼CG旗標之上下文選擇定義為：

(右方CG可用&&右方CG之旗標等於1)||(下方CG可用&&下方CG之旗標等於1)

在此，右方及下方CG為接近當前CG之兩個相鄰CG。舉例而

言，在圖10中，當寫碼左上方4×4區塊時，將右方CG定義為右上方4×4區塊且將下方CG定義為左下方4×4區塊。

應注意，色度及明度使用上下文之不同組，但藉由同一規則選擇其中之一者。

上下文索引增加之導出的細節可發現於HEVC之9.3.4.2.4中。

在一個CG內之變換係數寫碼。對於可含有非零係數之彼等CG，可針對每一係數根據預定義4×4係數掃描次序進一步寫碼有效旗標(significant_flag)、係數之絕對值(包括coeff_abs_level_greater1_flag、coeff_abs_level_greater2_flag及coeff_abs_level_remaining)及正負號資訊(coeff_sign_flag)。將寫碼變換係數層級分隔成多個掃描遍次。

1)第一位元子寫碼之第一遍次。在此遍次中，除可導出特定變換係數等於0以外，寫碼一個CG內之每一位置處之變換係數的所有第一位元子(或位元子索引0，bin0)。

變數sigCtx取決於相對於當前TU之左上方位置之當前地點、色彩分量索引cIdx、變換區塊大小及語法元素coded_sub_block_flag之先前經解碼位元子。取決於TU大小應用不同規則。上下文索引增加之選擇的實例細節定義於HEVC之9.3.4.2.5中。

2)第二位元子寫碼之第二遍次。在此遍次中應用coeff_abs_level_greater1_flags之寫碼。上下文模型化取決於當前子區塊內之色彩分量索引、當前子區塊掃描索引及當前係數掃描索引。上下文索引增加之選擇的實例細節定義於HEVC之9.3.4.2.6中。

3)第三位元子寫碼之第三遍次。在此遍次中應用coeff_abs_level_greater2_flags之寫碼。上下文模型化類似於coeff_abs_level_greater1_flags所使用之彼上下文模型化。上下文索引增加之選擇的實例細節定義於HEVC之9.3.4.2.7中。

應注意，為改善輸送量，第二及第三遍次可能不會處理CG中之所有係數。以常規模式寫碼CG中之前八個coeff_abs_level_greater1_flags。此後，維持該等值，從而在第五遍次中藉由語法coeff_abs_level_remaining以旁路模式寫碼該等值。類似地，僅寫碼CG中具有大於1之量值的第一係數之coeff_abs_level_greater2_flags。CG之具有大於1之量值的係數的剩餘部分使用coeff_abs_level_remaining寫碼該值。此方法將係數階之常規位元子的數目限制為每CG最多9個：8個用於coeff_abs_level_greater1_flags且1個用於coeff_abs_level_greater2_flags。

4)正負號資訊之第四遍次。在HEVC之一些實例中，在第四掃描遍次中以旁路模式寫碼每一非零係數之正負號。對於每一CG，且取決於準則，當使用正負號資料隱藏(SDH)時，僅省略最末非零係數(在反向掃描次序中)之正負號的編碼。替代地，使用預定義定則將正負號值嵌入於CG之層級之總和的同位檢查中：偶數對應於「+」且奇數對應於「-」。使用SDH之準則為CG之第一非零係數與最末非零係數之間的掃描次序中之距離。若此距離等於或大於4，則SDH經使用。由於此值4對HEVC測試序列提供最大增益，故選擇該值。

5)剩餘位元子之最末遍次。在另一掃描遍次中寫碼剩餘位元子。將係數之*baseLevel*定義為：

$$baseLevel = \text{significant\_flag} + \text{coeff\_abs\_level\_greater1\_flag} + \text{coeff\_abs\_level\_greater2\_flag}$$

其中旗標具有值0或1，且若不存在，則推斷為0。接著，係數之絕對值僅為：

$$\text{absCoeffLevel} = baseLevel + \text{coeff\_abs\_level\_remaining}.$$

在每一CG之開始處將萊斯參數設定成0，且其取決於參數之先前值及當前絕對層級條件性地更新如下：

if absCoeffLevel > 3 × 2m, m= min(4,m + 1).

可以旁路模式寫碼語法元素coeff_abs_level_remaining。另外，HEVC之一些實例針對較小值採用哥倫布萊斯碼且針對較大值切換至指數哥倫布碼。程式碼之間的轉換點通常在一元碼長度等於4時。參數更新過程在觀測到較大值分佈時允許二進位化適應於係數統計。

inter_pred_idc之上下文模型化。inter_pred_idc指定list0、list1或雙向預測是否用於當前預測單元。語法元素已達至兩個位元子，該等位元子兩者皆經CABAC上下文寫碼。將二進位化位元子串定義如下：

inter_pred_idc之值	位元子串	位元子串
	(nPbW + nPbH) != 12	(nPbW + nPbH) != 12
0	00	00
1	01	01
2	1	1

其中nPbW及nPbH分別表示當前明度預測區塊之寬度及高度。

對於每一經框間寫碼圖塊(例如，P圖塊或B圖塊)，上下文選擇係基於以下規則：

- 若(nPbW + nPbH)不等於12，則使用四個上下文寫碼第一位元子且藉由一個上下文寫碼第二位元子。第一位元子之上下文選擇係根據當前CU深度。在HEVC中，CU深度包含地處於0至3之範圍內。

圖11為繪示根據本發明之一或多種技術的用於藉由不同視窗大小執行基於上下文之熵編碼之實例過程的流程圖。圖11之技術可由視訊編碼器(諸如圖1及圖4中所繪示之視訊編碼器20)執行。出於說明的目的，在圖1及圖4之視訊編碼器20之上下文內描述圖11之技術，但具有不同於視訊編碼器20之彼組態的組態的視訊編碼器亦可執行圖11之技術。

視訊編碼器20可獲得待使用基於上下文之熵寫碼來編碼的位元

子串(例如，一維二進位向量) (1102)。舉例而言，視訊編碼器20之熵編碼單元56可藉由二進位化自視訊編碼器20之預測處理單元42接收之語法元素而獲得位元子串。在一些實例中，基於上下文之熵寫碼可包含上下文自適應性二進位算術寫碼(CABAC)。

根據本發明之一或多種技術，視訊編碼器20可針對複數個上下文中之上下文判定複數個視窗大小中之視窗大小(1104)。在一些實例中，視訊編碼器20可基於用於上下文之預定視窗大小判定視窗大小。在一些實例中，視訊編碼器20可藉由分析若干候選視窗大小之寫碼效率判定視窗大小，且將具有最佳寫碼效率之候選視窗大小選為上下文之視窗大小。

舉例而言，對於每一上下文，熵編碼單元56可藉由不同視窗大小計算寫碼所記錄位元子串之位元，且選擇具有最小位元之一者。在一些實例中，可預定義熵編碼單元56使用之不同視窗大小。某些實例預定義視窗大小為16、32、64及128，但預期其他視窗大小。

在一些實例中，熵編碼單元56可編碼指示用於編碼位元串之視窗大小的一或多個語法元素。舉例而言，熵編碼單元56可在當前圖塊之圖塊標頭中編碼指示將預設視窗大小抑或經更新視窗大小用於每一上下文之第一旗標。作為一項實例，在熵編碼單元56使用與除預設視窗大小外之視窗大小相關聯的上下文寫碼當前圖塊之一或多個位元串的情況下，熵編碼單元56可在當前圖塊之圖塊標頭中編碼第一旗標以指示一或多個位元串屬於使用除預設視窗大小外之視窗大小來寫碼的當前圖塊。類似地，在熵編碼單元56使用與預設視窗大小相關聯之上下文寫碼圖塊之所有位元串的情況下，熵編碼單元56可編碼當前圖塊之圖塊標頭中之第一旗標以指示當前圖塊之所有位元串使用預設視窗大小來寫碼。在一些實例中，如下文更詳細描述，第一旗標可被稱為default_updating_speed_flag。

在一些實例中，在熵編碼單元56在當前圖塊之圖塊標頭中編碼指示一或多個位元串屬於使用除預設視窗大小外之視窗大小來寫碼的圖塊之第一旗標的情況下，熵編碼單元56可在當前圖塊之圖塊標頭中編碼指示與用於寫碼當前圖塊之上下文相關聯的視窗大小是否繼承自先前經寫碼圖塊的第二旗標。在一些實例中，先前經寫碼圖塊可為具有與當前圖塊相同之一或多個參數(諸如，相同圖塊類型及相同初始化QP)的最新經寫碼圖塊。在一些實例中，如下文更詳細描述，第一旗標可被稱為inheritance_from_previous_flag。

語法

7.3.6 圖塊片段標頭語法

7.3.6.1 常用圖塊片段標頭語法

slice_segment_header( ) {	描述符
first_slice_segment_in_pic_flag	u(1)
若(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
…	
若(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
對於(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
default_updating_speed_flag	*u(1)*
if(!default_updating_speed) {	
inheritance_from_previous_flag	*u(1)*
if(!inheritance_from_previous_flag) {	
bit_map_run_length_coding ( )	
speed_index_level_coding ( )	
}	
}	
byte_alignment( )	
}	

下文提供上述語法元素之某些實例語義：

default_updating_speed_flag等於1可指定將預設視窗大小用於所有上下文且inheritance_from_previous_flag不存在於圖塊標頭中。

default_updating_speed_flag 等於0可指定inheritance_from_previous_flag存在於圖塊標頭中。

inheritance_from_previous_flag 等於1可指定與上下文相關聯之視窗大小係繼承自具有相同圖塊類型及相同初始化QP之先前經寫碼圖塊。inheritance_from_previous_flag等於0可指定與上下文相關聯之視窗大小在位元串流中傳信，存在bit_map_run_length_coding()及speed_index_level_coding。

在一些實例中，可明確地指定待使用之圖像。若圖像含有多個圖塊，則可使用第一圖塊抑或可明確地指定彼圖像之圖塊的id。在一些實例中，inheritance_from_previous_flag等於1可指定與上下文相關聯之視窗大小係繼承自具有相同圖塊類型之先前經寫碼圖塊。

在一些實例中，在熵編碼單元56編碼第二旗標以指示與用於寫碼當前圖塊之上下文相關聯的視窗大小並非繼承自先前經寫碼圖塊的情況下，熵編碼單元56可寫碼一或多個語法元素以指示與用於寫碼當前圖塊之上下文相關聯的視窗大小。舉例而言，熵編碼單元56可寫碼指示不同視窗大小之使用的一個位元映射，且在位元指示新的視窗大小時寫碼新的視窗大小索引。在一些實例中，熵編碼單元56可如下文所述編碼一或多個語法元素以指示與用於寫碼當前圖塊之上下文相關聯的視窗大小。

7.xxxx 位元映射延行長度寫碼語法

bit_map_run_length_coding ( ) {	描述符
run = 0	
ctxIdx = 0	
當 (ctxIdx < totalCtxNr) {	
run[i]	ue(v)
ctxIdx += run[i]	
若(ctxIdx >= totalCtxNr) {	
break;	
}	
ctxUpdatedSpeedFlag[ctxIdx] = 1	
ctxIdx ++	

}	
}	

替代地，可定義下表：

bit_map_run_length_coding ( ) {	描述符
run = 0	
ctxIdx = 0	
當 (ctxIdx < totalCtxNr) {	
run[i]	ue(v)
ctxIdx += run[i]	
ctxUpdatedSpeedFlag[ctxIdx] = 1	
ctxIdx ++	
}	
}	

7.xxxx 速度索引層級寫碼語法

speed_index_level_coding ( ) {	描述符
對於 (i = 0; i < totalCtxNr; i ++) {	
若 (ctxUpdatedSpeedFlag[i]) {	
ctx_idx_difference[i]	ue(v)
}	
}	
}	

下文提供上述語法元素之某些實例語義：

run[i]可指示使用預設更新速度之連續上下文的數目。

ctxUpdatedSpeedFlag可為具有totalCtxNr條目之陣列。對於每一條目，可在解碼指示每一上下文使用預設機率更新速度之一個圖塊之前將其(亦即，預設視窗大小)設定成0。在一項實例中，預設視窗大小等於64。

在一項實例中，totalCtxNr可表示可用於當前圖塊中之上下文的總數目。在另一實例中，totalCtxNr可表示可用於所有圖塊中之上下文的總數目。在另一實例中，totalCtxNr可表示經預定義之所選上下文的總數目。

ctx_idx_difference可指示視窗大小之索引與預設視窗大小相比之差。

在一項實例中，預設視窗大小可等於64。在一些實例中，諸如

當ctx_idx_difference等於2時，視窗大小可設定成128。在一些實例中，諸如當ctx_idx_difference等於0或1時，視窗大小可設定等於(1<<(ctx_idx_difference+4))。亦即，可支援四個視窗大小(亦即，16、32、64及128)，但預期具有額外或更少視窗大小之實例。在一些實例中，熵編碼單元56可傳信主動參數集中之上述資訊中的一些或所有。

在解碼器側，對於每一圖塊標頭，第一旗標可首先經解碼，該第一旗標可指示將預設視窗大小或經更新視窗大小用於每一上下文。在一些實例中，若第一旗標等於1 (亦即，使用經更新視窗大小)，則第二旗標可進一步經解碼，該第二旗標可指示自先前經寫碼圖像之繼承或經更新視窗大小之額外傳信。若傳信視窗大小為所需的，則可首先傳信位元映射以指示不同視窗大小之使用，且在位元指示新的視窗大小時傳信新的視窗大小索引。

在任何情況下，視訊編碼器20可在視訊位元串流中且基於上下文之機率狀態編碼位元子串中之位元子(1106)。舉例而言，熵編碼單元56可將表示值或指標之二進位串流輸出至上下文之最終經寫碼機率區間內之機率。

視訊編碼器20可基於所判定之視窗大小更新上下文模型之機率狀態(1108)。舉例而言，給定與第i個上下文模型相關聯之所判定視窗大小W_i，熵編碼單元56可根據以下方程式(15)更新第i個上下文模型之機率狀態，其中k可表示機率精度。在一項實例中，k等於15。

$$P_{new}=\begin{cases}(2^k/W_i)+P_{old}-(P_{old}/W_i) & MPS\ (e.g.,1)\\ P_{old}-(P_{old}/W_i) & LPS\ (e.g.,(1-MPS))\end{cases} \quad \textbf{(15)}$$

當W_i等於(1 << M_i)時，由熵編碼單元56執行之機率更新過程可如以下方程式(16)中所展示予以重寫。

$$P_{new}=\begin{cases}P_{old}+((2^k-P_{old})>>M_i) & MPS\ (e.g.,1)\\ P_{old}-(P_{old}>>M_i) & LPS\ (e.g.,(1-MPS))\end{cases} \quad \textbf{(16)}$$

一或多個位元串屬於使用除預設視窗大小外之視窗大小來寫碼的當前圖塊之第一旗標。類似地，在熵解碼單元70使用與預設視窗大小相關聯之上下文解碼圖塊之所有位元串的情況下，熵解碼單元70可自當前圖塊之圖塊標頭解碼指示當前圖塊之所有位元串使用預設視窗大小來寫碼的第一旗標。在一些實例中，如上文更詳細描述，第一旗標可被稱為default_updating_speed_flag。

在一些實例中，在熵編碼單元56在當前圖塊之圖塊標頭中編碼指示一或多個位元串屬於使用除預設視窗大小外之視窗大小來寫碼的圖塊之第一旗標的情況下，熵編碼單元56可在當前圖塊之圖塊標頭中編碼指示與用於寫碼當前圖塊之上下文相關聯的視窗大小是否繼承自先前經寫碼圖塊的第二旗標。在一些實例中，先前經寫碼圖塊可為具有與當前圖塊相同之一或多個參數(諸如，相同圖塊類型及相同初始化QP)的最新經寫碼圖塊。在一些實例中，如上文參考圖11所更詳細描述，第一旗標可被稱為inheritance_from_previous_flag。

在一些實例中，可明確地指定待使用之圖像。若圖像含有多個圖塊，則可使用第一圖塊抑或可明確地指定彼圖像之圖塊的id。在一些實例中，inheritance_from_previous_flag等於1可指定與上下文相關聯之視窗大小係繼承自具有相同圖塊類型之先前經寫碼圖塊。

在一些實例中，在第二旗標指示與用於寫碼當前圖塊之上下文相關聯之視窗大小並非繼承自先前經寫碼圖塊的情況下，熵解碼單元70可解碼指示與用於寫碼當前圖塊之上下文相關聯的視窗大小之一或多個語法元素。舉例而言，熵解碼單元70可解碼指示不同視窗大小之使用的一個位元映射，且在位元指示新的視窗大小時解碼新的視窗大小索引。在一些實例中，熵解碼單元70可如上文參考圖11所描述解碼一或多個語法元素以指示與用於寫碼當前圖塊之上下文相關聯的視窗大小。

在任何情況下，視訊解碼器30可基於上下文之機率狀態解碼位元子串中之位元子(1206)。視訊解碼器30可基於所判定之視窗大小及經解碼位元子更新上下文模型之機率狀態(1208)。舉例而言，給定與第i個上下文模型相關聯之所判定視窗大小W_i，熵解碼單元70可根據以上方程式(15)更新第i個上下文模型之機率狀態。

視訊解碼器30可基於上下文之經更新機率狀態解碼另一位元子(1206)。在一些實例中，經編碼之其他位元子可為位元子串中之第二位元子。

以下編號實例可說明本發明之一或多個態樣：

實例1。 一種用於熵寫碼視訊資料之方法，該方法包含：針對在上下文自適應性熵寫碼過程中使用以熵寫碼視訊資料之語法元素的值之複數個上下文中之上下文判定複數個視窗大小中之視窗大小；基於上下文之機率狀態熵寫碼語法元素之值的位元子；及基於視窗大小及經寫碼位元子更新上下文之機率狀態。

實例2。 如請求項1之方法，其中上下文自適應性熵寫碼過程包含上下文自適應性二進位算術寫碼(CABAC)過程或上下文自適應性可變長度寫碼(CAVLC)過程。

實例3。 如請求項1之方法，其進一步包含：基於經更新機率狀態熵寫碼與同一上下文相關聯之另一位元子。

實例4。 如請求項1之方法，其中上下文為第一上下文，該方法進一步包含：針對複數個上下文中之第二上下文判定複數個視窗大小中之視窗大小，其中第二上下文之視窗大小不同於第一上下文之視窗大小。

實例5。 如請求項4之方法，其中第一上下文之視窗大小及第二上下文之視窗大小不在包括經寫碼位元子之位元串流中傳信。

實例6。 如請求項1之方法，其中複數個視窗大小包含視窗大小

之預定義集合。

實例7。 如請求項6之方法，其中熵寫碼包含熵編碼，且其中判定視窗大小包含：針對視窗大小之預定義集合中之各別視窗大小判定用於熵編碼包括語法元素之位元子值的特定位元子串的位元之各別量；及選擇視窗大小之預定義集合中對應於位元之最小量的視窗大小作為上下文之視窗大小以熵編碼特定位元子串。

實例8。 如請求項1之方法，其進一步包含：寫碼指示預設視窗大小是否用於複數個上下文之第一語法元素。

實例9。 如請求項8之方法，其進一步包含：基於指示預設視窗大小不用於複數個上下文之第一語法元素，寫碼指示上下文之視窗大小的第二語法元素。

實例10。如請求項9之方法，其中為指示上下文之視窗大小，第二語法元素指示上下文之視窗大小與預設視窗大小之間的差。

實例11。如請求項8之方法，其中寫碼第一語法元素包含寫碼包括第一語法元素之當前圖塊之圖塊標頭，其中第一語法元素指示當熵寫碼當前圖塊之位元子時預設視窗大小是否用於複數個上下文。

實例12。如請求項1之方法，其進一步包含：在當前圖塊之圖塊標頭中寫碼指示用於複數個上下文之視窗大小是否繼承自先前經寫碼圖塊之語法元素。

實例13。如請求項1之方法，其中判定上下文之視窗大小包含：基於語法元素類型判定上下文之視窗大小。

實例14。如請求項1之方法，其中熵寫碼包含熵解碼，該方法進一步包含：自經寫碼視訊位元串流解碼指示上下文之視窗大小的一或多個語法元素。

實例15。一種用於熵寫碼視訊資料之裝置，該裝置包含：記憶體，其經組態以儲存在上下文自適應性熵寫碼過程中使用以熵寫碼

視訊資料之語法元素之值的複數個上下文；及一或多個處理器，其經組態以：執行如實例1至14之任何組合之方法。

實例16。如實例15之裝置，其中該裝置包含以下各者中之至少一者：積體電路、微處理器或無線通信器件。

實例17。如實例15至16之任何組合之裝置，其進一步包含經組態以顯示經解碼視訊資料之顯示器。

實例18。如實例15至17之任何組合之裝置，其進一步包含經組態以俘獲視訊資料之攝影機。

實例19。一種用於熵寫碼視訊資料之裝置，該裝置包含：用於執行如實例1至14之任何組合之方法的構件。

實例20。一種儲存指令之電腦可讀儲存媒體，該等指令在執行時使得視訊寫碼器件之一或多個處理器執行如實例1至14之任何組合之方法。

實例21。一種儲存視訊資料之電腦可讀儲存媒體，該視訊資料在藉由視訊解碼器件處理時使得視訊解碼器件之一或多個處理器針對使用於上下文自適應性寫碼過程中以熵寫碼語法元素之值的複數個上下文中之上下文判定複數個視窗大小中的視窗大小；基於上下文之機率狀態熵寫碼語法元素之值的位元子；基於視窗大小及經寫碼位元子更新上下文之機率狀態；及基於上下文模型之經更新機率狀態藉由相同上下文熵寫碼下一位元子。

實例22。如實例21之電腦可讀儲存媒體，其進一步儲存指令，該等指令使得該一或多個處理器執行如實例1至14之任何組合之方法。

在一或多項實例中，所描述之功能可以硬體、軟體、韌體或其任何組合來實施。若以軟體實施，則該等功能可作為一或多個指令或程式碼而在電腦可讀媒體上儲存或傳輸，且由基於硬體之處理單

元執行。電腦可讀媒體可包括電腦可讀儲存媒體，其對應於有形媒體，諸如資料儲存媒體，或包括有助於將電腦程式自一處傳送至另一處(例如，根據通信協定)之任何媒體的通信媒體。以此方式，電腦可讀媒體大體上可對應於(1)非暫時性的有形電腦可讀儲存媒體，或(2)通信媒體，諸如信號或載波。資料儲存媒體可為可由一或多個電腦或一或多個處理器存取以擷取用於實施本發明中所描述之技術的指令、程式碼及/或資料結構的任何可用媒體。電腦程式產品可包括電腦可讀媒體。

藉由實例而非限制的方式，此類電腦可讀儲存媒體可包含RAM、ROM、EEPROM、CD-ROM或其他光碟儲存器、磁碟儲存器或其他磁性儲存器件、快閃記憶體或可用以儲存呈指令或資料結構形式之所要程式碼且可由電腦存取的任何其他媒體。又，將任何連接適當地稱為電腦可讀媒體。舉例而言，若使用同軸纜線、光纜、雙絞線、數位用戶線(DSL)或無線技術(諸如紅外線、無線電及微波)自網站、伺服器或其他遠端源傳輸指令，則同軸纜線、光纜、雙絞線、DSL或無線技術(諸如紅外線、無線電及微波)包括於媒體之定義中。然而，應理解，電腦可讀儲存媒體及資料儲存媒體不包括連接、載波、信號或其他暫時性媒體，而是實際上有關非暫時性有形儲存媒體。如本文所使用，磁碟及光碟包括緊密光碟(CD)、雷射光碟、光學光碟、數位多功能光碟(DVD)、軟性磁碟及藍光光碟，其中磁碟通常以磁性方式再生資料，而光碟用雷射以光學方式再生資料。以上各者之組合亦應包括於電腦可讀媒體之範疇內。

可由一或多個處理器執行指令，該一或多個處理器諸如一或多個數位信號處理器(DSP)、通用微處理器、特殊應用積體電路(ASIC)、場可程式化邏輯陣列(FPGA)或其他等效之整合或離散邏輯電路。因此，如本文中所使用之術語「處理器」可指上述結構或適

用於實施本文中所描述之技術的任何其他結構中之任一者。另外，在一些態樣中，本文中所描述之功能性可提供於經組態用於編碼及解碼的專用硬體及/或軟體模組內，或併入於組合式編解碼器中。又，該等技術可完全實施於一或多個電路或邏輯元件中。

本發明之技術可在廣泛多種器件或裝置中實施，該等器件或裝置包括無線手機、積體電路(IC)或IC之集合(例如，晶片集合)。在本發明中描述各種組件、模組或單元以強調經組態以執行所揭示技術之器件的功能性態樣，但未必需要藉由不同硬體單元來實現。確切地說，如上文所描述，可將各種單元組合於編解碼器硬體單元中，或藉由互操作性硬體單元(包括如上文所描述之一或多個處理器)之集合而結合合適軟體及/或韌體來提供該等單元。

已描述各種實例。此等及其他實例在以下申請專利範圍之範疇內。

【符號說明】

10	視訊編碼及解碼系統
12	源器件
14	目的地器件
16	電腦可讀媒體
18	視訊源
20	視訊編碼器
22	輸出介面
28	輸入介面
30	視訊解碼器
31	顯示器件
32	儲存器件
40	視訊資料記憶體

42	預測處理單元
44	運動估計單元
46	運動補償單元
48	框內預測單元
50	求和器
52	變換處理單元
54	量化處理單元
56	熵編碼單元
58	逆量化處理單元
60	逆變換處理單元
62	求和器
64	參考圖像記憶體
69	視訊資料記憶體
70	熵解碼單元
71	預測處理單元
72	運動補償單元
74	框內預測單元
76	逆量化處理單元
78	逆變換處理單元
80	求和器
82	參考圖像記憶體
100	實例
102	實例
118	語法元素
120	二進位化器
122	上下文建模器

124	常規寫碼引擎
126	旁路寫碼引擎
218	位元串流
220	上下文建模器
222	旁路寫碼引擎
224	常規解碼引擎
230	反向二進位化器
1102	區塊
1104	區塊
1106	區塊
1108	區塊
1202	區塊
1204	區塊
1206	區塊
1208	區塊

申請專利範圍

1. 一種用於熵寫碼視訊資料之方法，該方法包含：

針對在一上下文自適應性熵寫碼過程中使用以熵寫碼該視訊資料之一語法元素之一值的複數個上下文中之一上下文判定複數個視窗大小中之一視窗大小；

基於該上下文之一機率狀態熵寫碼該語法元素之該值之一位元子；及

基於該視窗大小及該經寫碼位元子更新該上下文之該機率狀態。

2. 如請求項1之方法，其中該上下文自適應性熵寫碼過程包含一上下文自適應性二進位算術寫碼(CABAC)過程或一上下文自適應性可變長度寫碼(CAVLC)過程。

3. 如請求項1之方法，其進一步包含：

基於該經更新機率狀態熵寫碼與該同一上下文相關聯之另一位元子。

4. 如請求項1之方法，其中該上下文為一第一上下文，該方法進一步包含：

針對該複數個上下文中之一第二上下文判定該複數個視窗大小中之一視窗大小，其中該第二上下文之該視窗大小不同於該第一上下文之該視窗大小。

5. 如請求項4之方法，其中該第一上下文之該視窗大小及該第二上下文之該視窗大小不在包括該經寫碼位元子之一位元串流中傳信。

6. 如請求項1之方法，其中該複數個視窗大小包含視窗大小之一預定義集合。

7. 如請求項6之方法，其中熵寫碼包含熵編碼，且其中判定該視窗大小包含：

針對視窗大小之該預定義集合中之各別視窗大小判定用於熵編碼包括該語法元素之該等位元子值之一特定位元子串的位元之各別量；及

選擇視窗大小之該預定義集合中對應於位元之該最小量之該視窗大小作為該上下文之該視窗大小，以熵編碼該特定位元子串。

8. 如請求項1之方法，其進一步包含：

寫碼指示一預設視窗大小是否用於該複數個上下文之一第一語法元素。

9. 如請求項8之方法，其進一步包含：

基於該第一語法元素指示該預設視窗大小不用於該複數個上下文，寫碼指示該上下文之該視窗大小的一第二語法元素。

10. 如請求項9之方法，其中為指示該上下文之該視窗大小，該第二語法元素指示該上下文之該視窗大小與該預設視窗大小之間的一差。

11. 如請求項8之方法，其中寫碼該第一語法元素包含寫碼包括該第一語法元素之一當前圖塊之一圖塊標頭，其中該第一語法元素指示當熵寫碼該當前圖塊之位元子時該預設視窗大小是否用於該複數個上下文。

12. 如請求項1之方法，其進一步包含：

在一當前圖塊之一圖塊標頭中寫碼指示該複數個上下文之視窗大小是否繼承自一先前經寫碼圖塊的一語法元素。

13. 如請求項1之方法，其中判定該上下文之該視窗大小包含：

基於該語法元素之一類型判定該上下文之該視窗大小。

14. 如請求項1之方法，其中熵寫碼包含熵解碼，該方法進一步包含：

自一經寫碼視訊位元串流解碼指示該上下文之該視窗大小的一或多個語法元素。

15. 一種用於熵寫碼視訊資料之裝置，該裝置包含：

一記憶體，其經組態以儲存在一上下文自適應性熵寫碼過程中使用以熵寫碼該視訊資料之一語法元素之一值的複數個上下文；及

一或多個處理器，其經組態以：

針對該複數個上下文中之一上下文判定複數個視窗大小中之一視窗大小；

基於上下文模型之一機率狀態熵寫碼該語法元素之該值之一位元子；及

基於該視窗大小及該經寫碼位元子更新該上下文模型之該機率狀態。

16. 如請求項15之裝置，其中該一或多個處理器經進一步組態以：

基於該經更新機率狀態熵寫碼與該同一上下文相關聯之另一位元子。

17. 如請求項16之裝置，其中該上下文模型為一第一上下文，且其中該一或多個處理器經進一步組態以：

針對該複數個上下文中之一第二上下文判定該複數個視窗大小中之一視窗大小，其中該第二上下文之該視窗大小不同於該第一上下文之該視窗大小。

18. 如請求項17之裝置，其中該第一上下文之該視窗大小及該第二上下文之該視窗大小不在包括該經寫碼位元子之一位元串流中傳信。

19. 如請求項16之裝置，其中該複數個視窗大小包含視窗大小之一預定義集合。

20. 如請求項19之裝置，其中，為進行熵寫碼，該一或多個處理器經組態以熵編碼，且其中為判定該視窗大小，該一或多個處理器經組態以：

針對視窗大小之該預定義集合中之各別視窗大小判定用於熵編碼包括該語法元素之該位元子值之一特定位元子串的位元之各別量；及

選擇視窗大小之該預定義集合中對應於位元之該最小量之該視窗大小作為該上下文之該視窗大小，以熵編碼該特定位元子串。

21. 如請求項15之裝置，其中該一或多個處理器經進一步組態以：

寫碼指示一預設視窗大小是否用於該複數個上下文之一第一語法元素。

22. 如請求項21之裝置，其中，基於該第一語法元素指示該預設視窗大小不用於該複數個上下文，該一或多個處理器經進一步組態以：

寫碼指示該上下文之該視窗大小的一第二語法元素。

23. 如請求項22之裝置，其中為指示該上下文之該視窗大小，該第二語法元素指示該上下文之該視窗大小與該預設視窗大小之間的一差。

24. 如請求項21之裝置，其中，為寫碼該第一語法元素，該一或多個處理器經組態以寫碼包括該第一語法元素的一當前圖塊之一圖塊標頭，其中該第一語法元素指示當熵寫碼該當前圖塊之位元子時該預設視窗大小是否用於該複數個上下文模型。

25. 如請求項15之裝置，其中該一或多個處理器經進一步組態以：

在一當前圖塊之一圖塊標頭中寫碼指示該複數個上下文之視窗大小是否繼承自一先前經寫碼圖塊的一語法元素。

26. 如請求項15之裝置，其中，為判定該上下文之該視窗大小，該一或多個處理器經組態以：

基於該語法元素之一類型判定該上下文之該視窗大小。

27. 如請求項15之裝置，其中該裝置包含以下各者中之至少一者：

一積體電路；

一微處理器；或

一無線通信器件。

28. 如請求項27之裝置，其進一步包含一顯示器，該顯示器經組態以顯示經解碼視訊資料。

29. 如27 15之裝置，其進一步包含一攝影機，該攝影機經組態以俘獲該視訊資料。

30. 如請求項15之裝置，其中，為進行熵寫碼，該一或多個處理器經組態以熵解碼該語法元素之該值。

31. 一種用於熵寫碼視訊資料之裝置，該裝置包含：

用於針對在一上下文自適應性熵寫碼過程中使用以熵寫碼該視訊資料之一語法元素之一值的複數個上下文中之一上下文判定複數個視窗大小中之一視窗大小的構件；

用於基於該上下文之一機率狀態熵寫碼該語法元素之該值之一位元子的構件；及

用於基於該視窗大小及該經寫碼位元子更新該上下文之該機率狀態的構件。

32. 一種儲存指令之電腦可讀儲存媒體，該等指令在執行時使得一視訊寫碼器件之一或多個處理器進行以下操作：

針對在一上下文自適應性熵寫碼過程中使用以熵寫碼該視訊

資料之一語法元素之一值的複數個上下文中之一上下文判定複數個視窗大小中之一視窗大小；

基於該上下文之一機率狀態熵寫碼該語法元素之該值之一位元子；及

基於該視窗大小及該經寫碼位元子更新該上下文之該機率狀態。

圖式

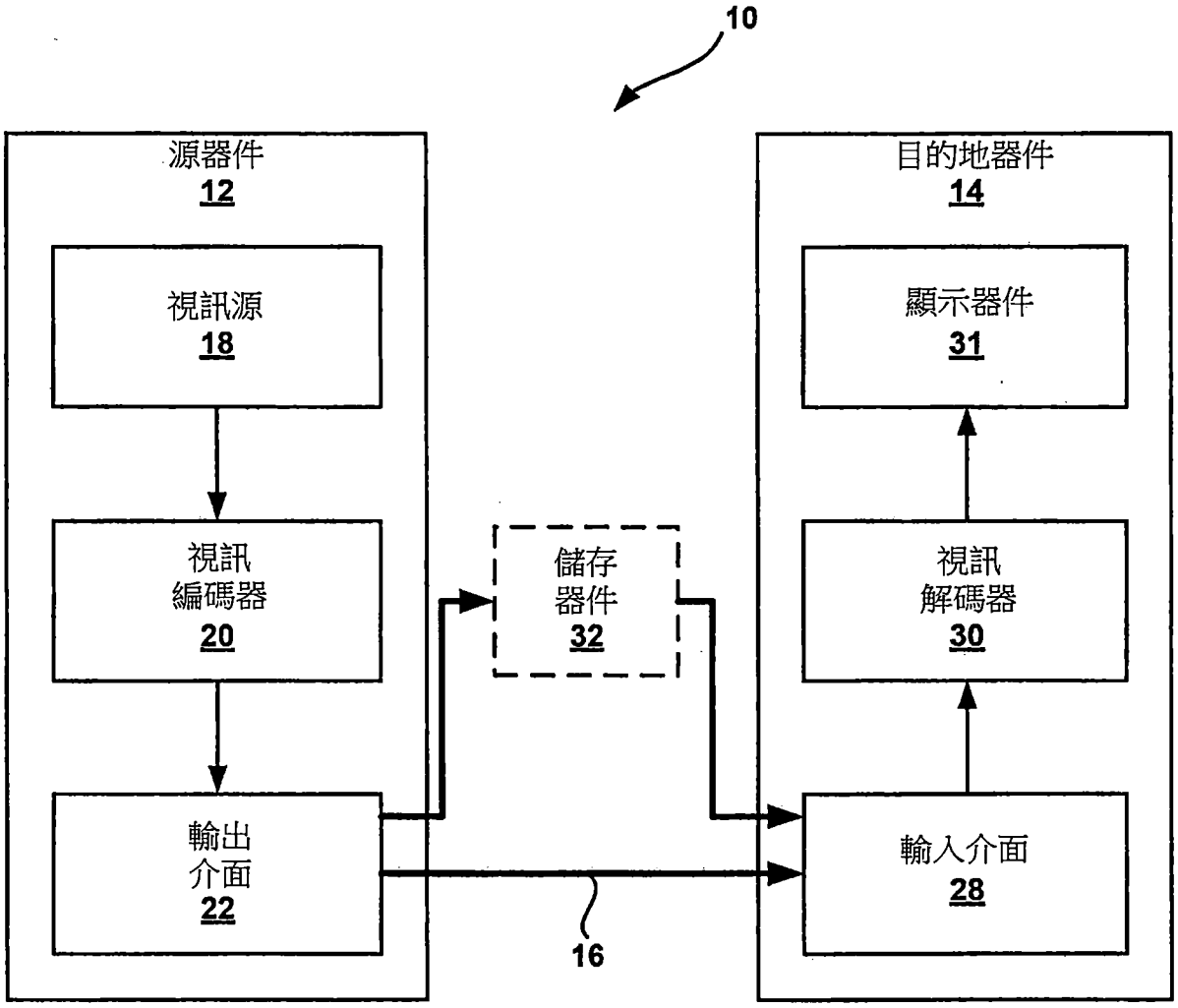

10
源器件
12
視訊源
18
視訊
編碼器
20
輸出
介面
22
儲存
器件
32
16
目的地器件
14
顯示器件
31
視訊
解碼器
30
輸入介面
28

圖1

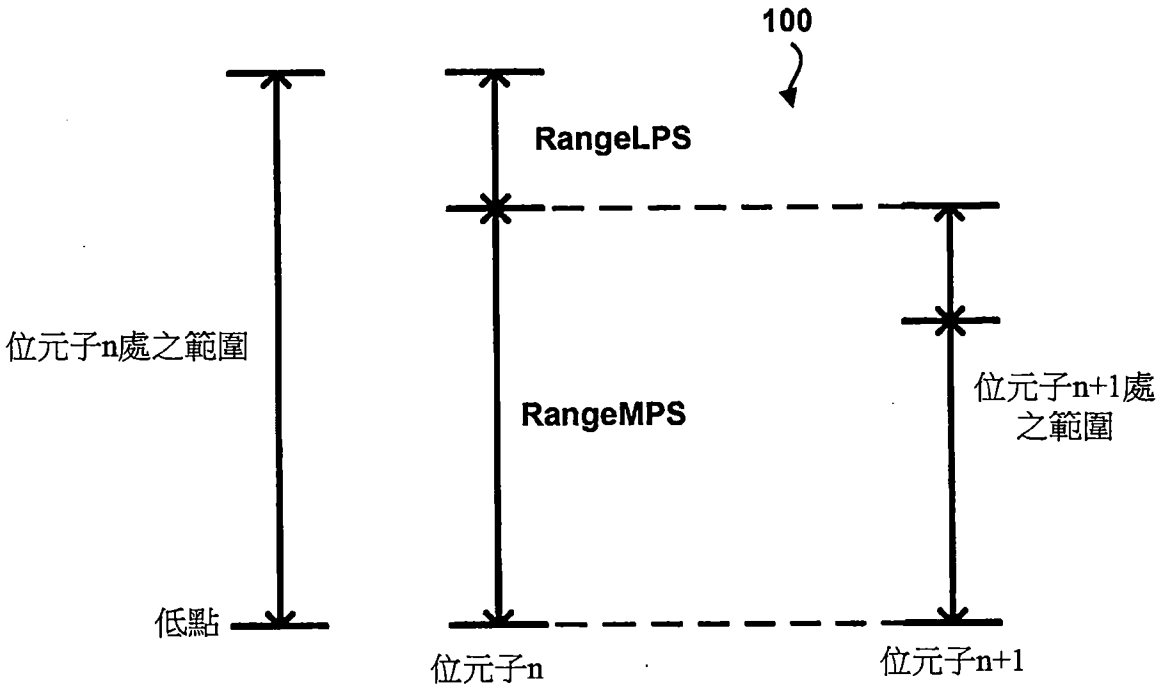

100
RangeLPS
RangeMPS
位元子n處之範圍
位元子n+1處
之範圍
低點
位元子n
位元子n+1

圖2A

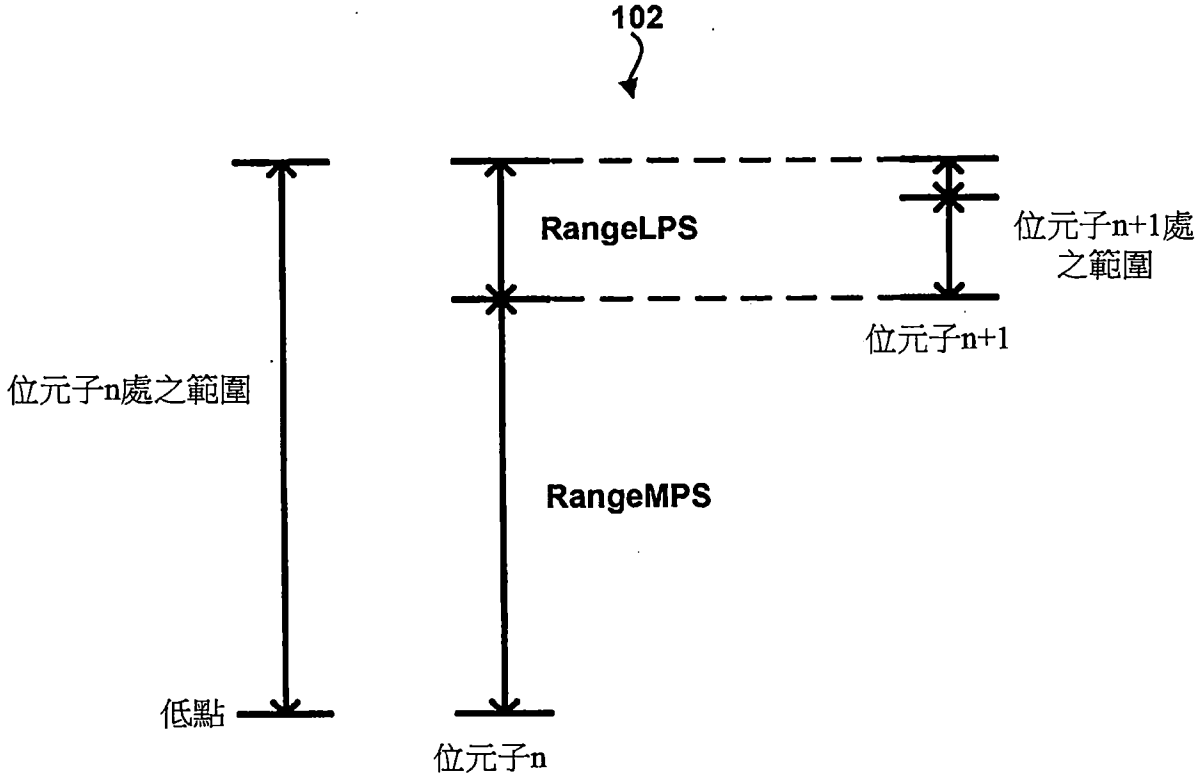

102
RangeLPS
位元子n+1處
之範圍
位元子n+1
位元子n處之範圍
RangeMPS
低點
位元子n

圖2B

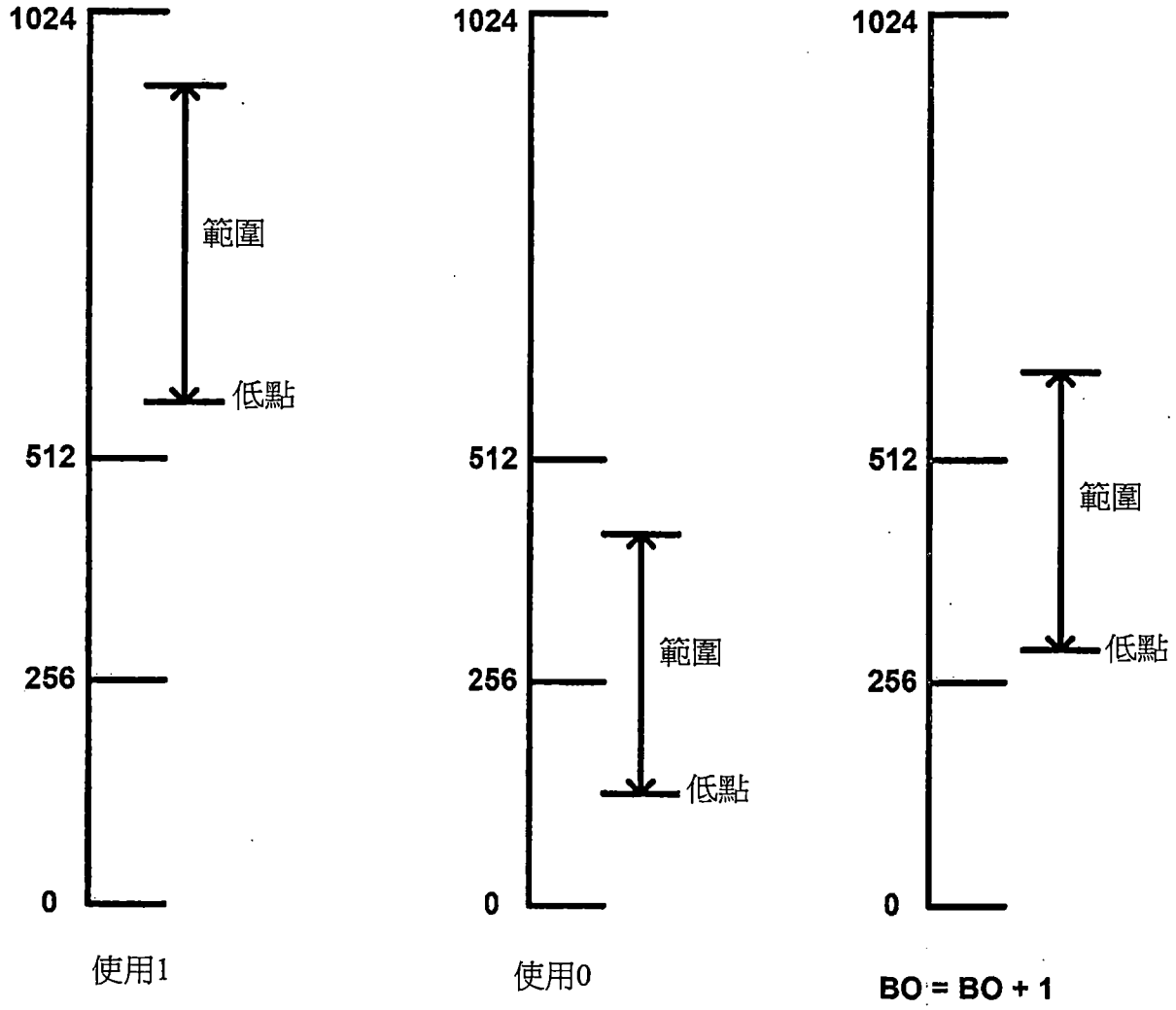

1024
512
256
0
範圍
低點
使用1
1024
512
256
0
範圍
低點
使用0
1024
512
256
0
範圍
低點
BO = BO + 1

圖3

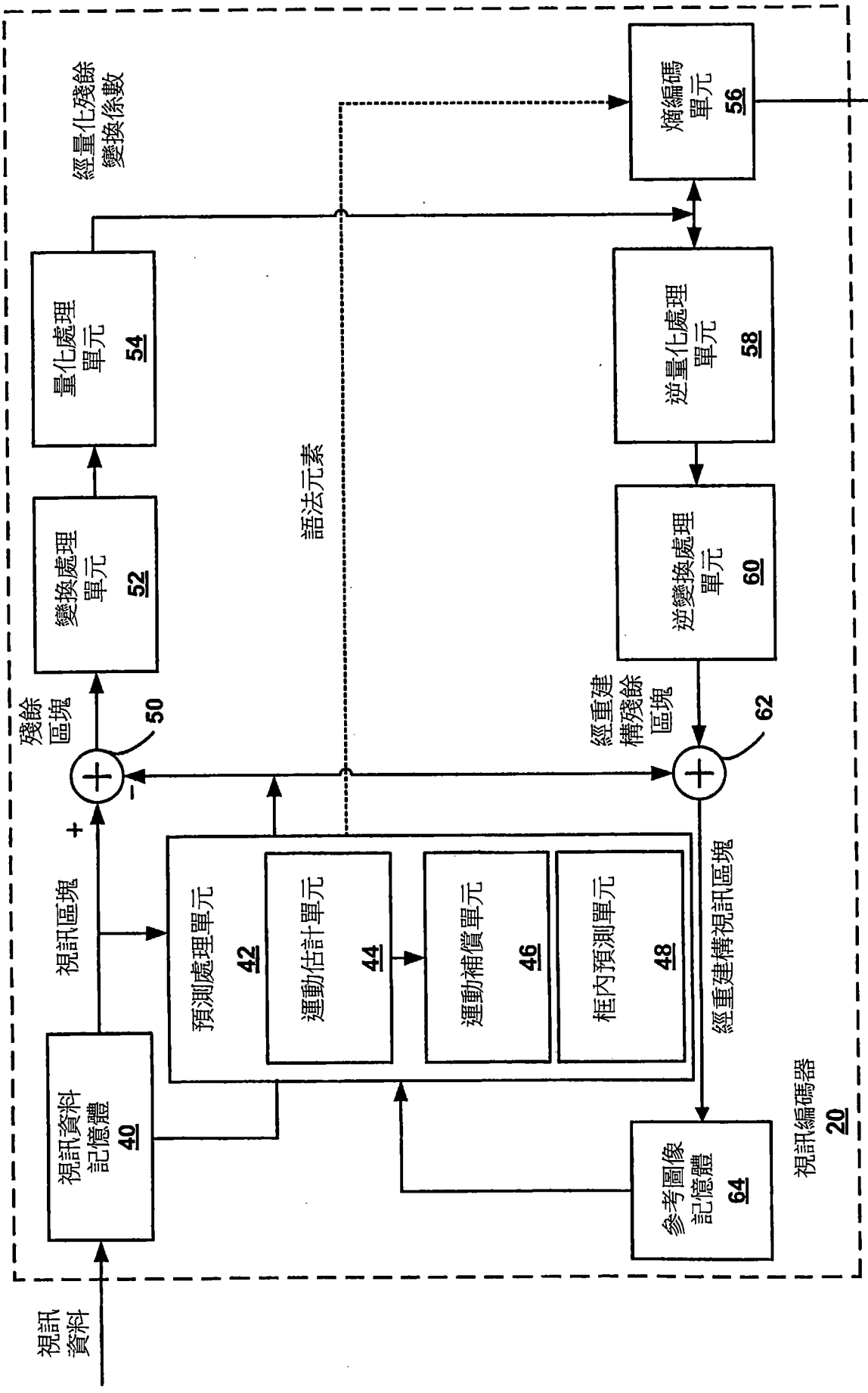

視訊資料
視訊資料記憶體 40
視訊區塊
預測處理單元 42
運動估計單元 44
運動補償單元 46
框內預測單元 48
+
-
殘餘區塊
50
變換處理單元 52
量化處理單元 54
經量化殘餘變換係數
語法元素
熵編碼單元 56
逆量化處理單元 58
逆變換處理單元 60
經重建構殘餘區塊
62
經重建構視訊區塊
參考圖像記憶體 64
視訊編碼器 20

圖4

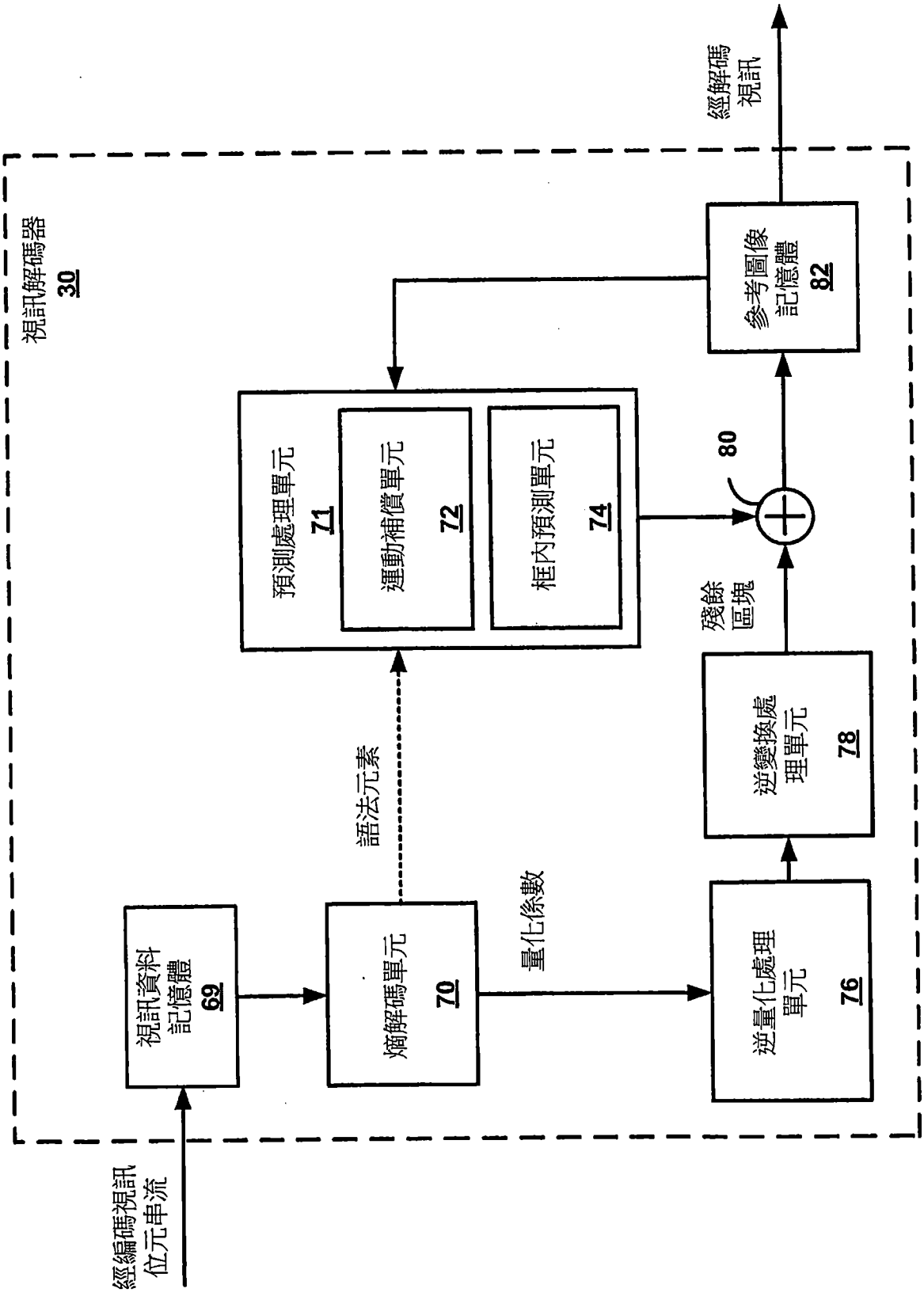
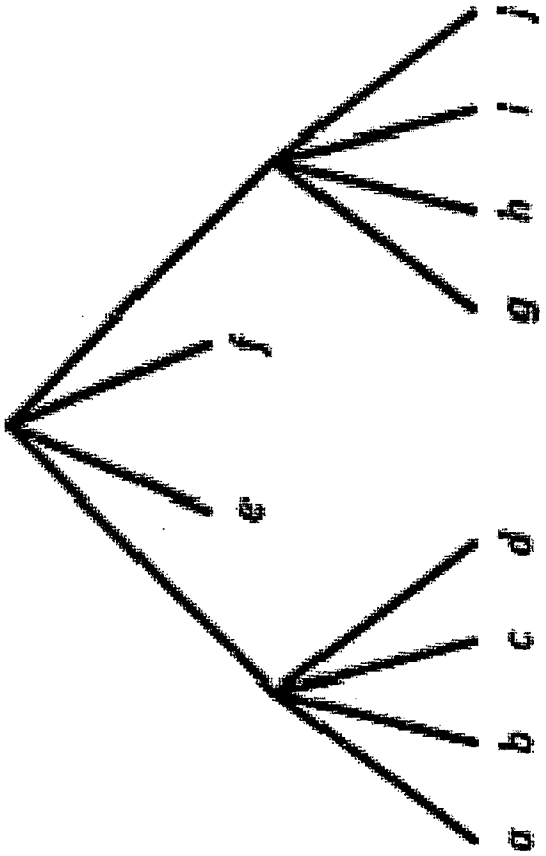

經編碼視訊位元串流
視訊解碼器 30
視訊資料記憶體 69
熵解碼單元 70
語法元素
量化係數
預測處理單元 71
運動補償單元 72
框內預測單元 74
逆量化處理單元 76
逆變換處理單元 78
殘餘區塊
80
參考圖像記憶體 82
經解碼視訊

圖6



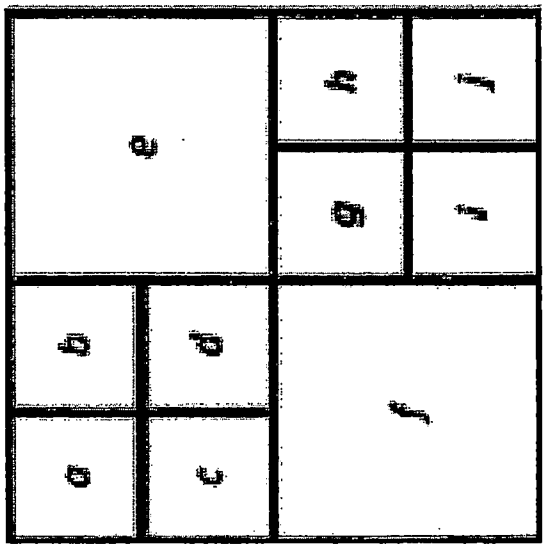


圖9

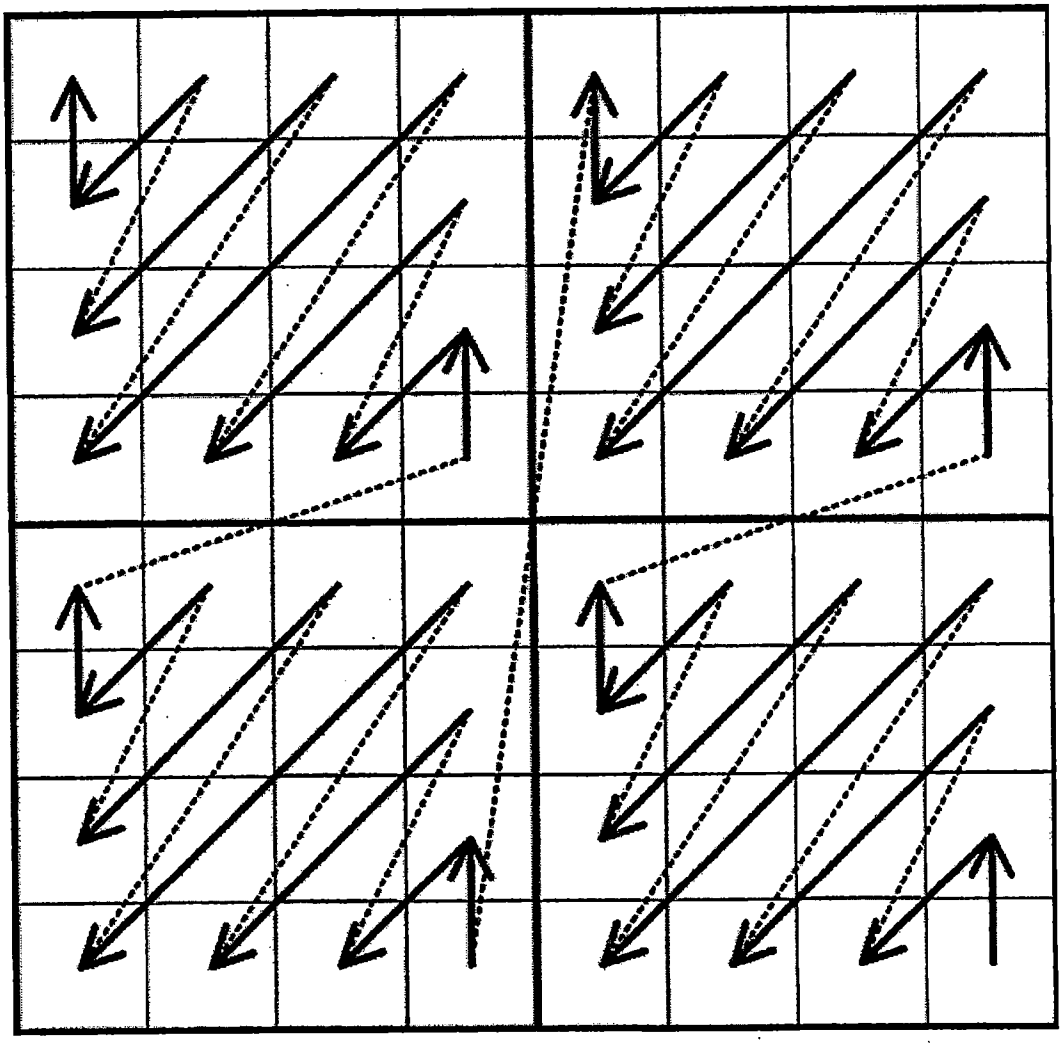


圖10

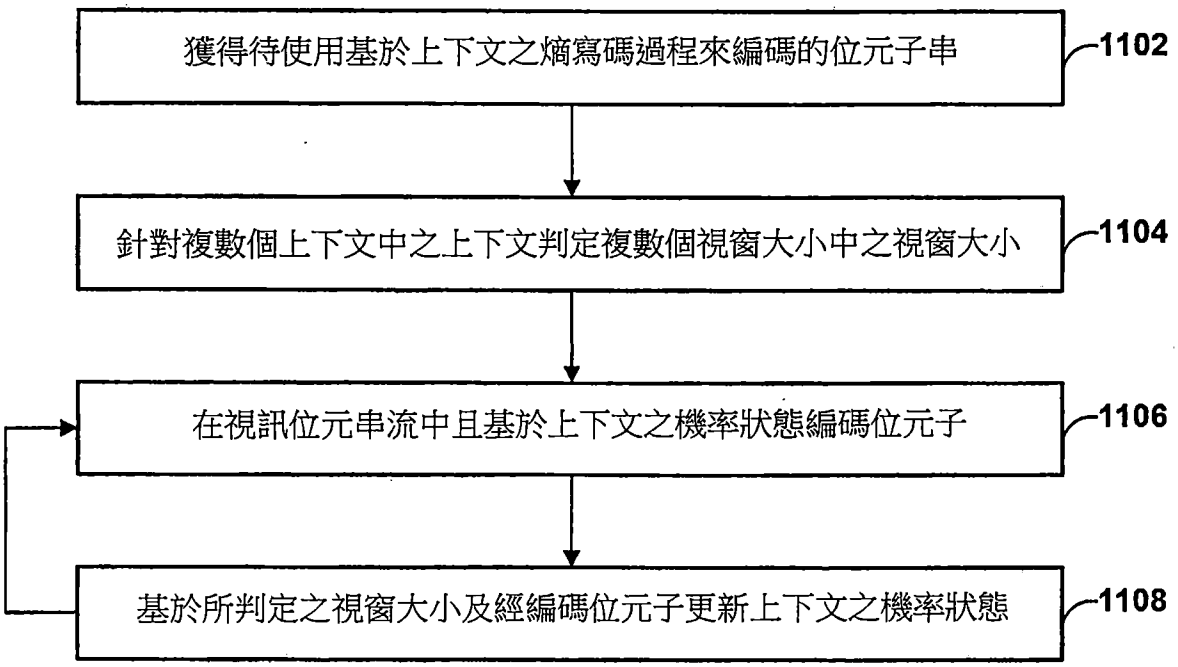

獲得待使用基於上下文之熵寫碼過程來編碼的位元子串
1102
針對複數個上下文中之上下文判定複數個視窗大小中之視窗大小
1104
在視訊位元串流中且基於上下文之機率狀態編碼位元子
1106
基於所判定之視窗大小及經編碼位元子更新上下文之機率狀態
1108

圖11

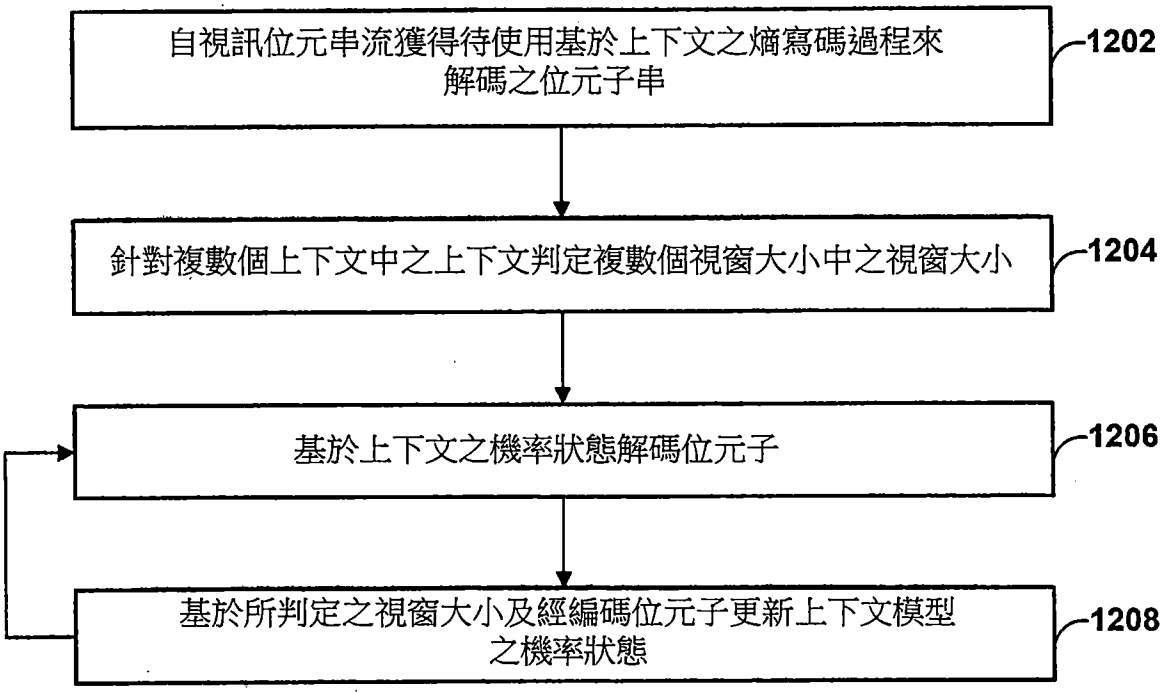

自視訊位元串流獲得待使用基於上下文之熵寫碼過程來解碼之位元子串
1202
針對複數個上下文中之上下文判定複數個視窗大小中之視窗大小
1204
基於上下文之機率狀態解碼位元子
1206
基於所判定之視窗大小及經編碼位元子更新上下文模型之機率狀態
1208

圖12