I837103

(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 I837103 B**

(45)公告日： 中華民國 113 (2024) 年 04 月 01 日

(21)申請案號：107139685 (22)申請日： 中華民國 107 (2018) 年 11 月 08 日

(51)Int. Cl.： ***G06F21/76 (2013.01)*** ***G06F7/575 (2006.01)***

(30)優先權：2018/02/02 英國 1801753.3

2017/11/30 英國 1719998.5

2017/11/09 英國 1718505.9

2017/12/13 英國 1720768.9

2018/04/10 英國 1805948.5

2018/04/20 英國 1806444.4

(71)申請人：安地卡及巴布達商區塊鏈控股有限公司 (安地卡及巴布達) NCHAIN HOLDINGS LIMITED (AG)

安地卡及巴布達

(72)發明人：馬戴爾 席蒙 MADEO, SIMONE (IT)；蒙提林斯基 派翠克 MOTYLINSKI, PATRICK (DK)；文生特 史帝芬 VINCENT, STEPHANE (FR)；柯瓦希 亞歷山特拉 COVACI, ALEXANDRA (RO)

(74)代理人：劉法正；尹重君

(56)參考文獻：

TW M543413U US 5,920,830

US 6,519,754B1 US 7,281,017B2

US 2004/0015739A1

期刊 Kosba, A., et al. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts IEEE Symposium on Security and Privacy(SP) DOI 10.1109/SP.2016.55 2016 pp 839-858

期刊 Parno, B., et al. ,Pinocchio: Nearly Practical Verifiable Computation, IEEE Symposium on Security and Privacy 2013 pages 238-252

審查人員：林文琦

申請專利範圍項數：15 項 圖式數：8 共 45 頁

(54)名稱

電腦實施方法及系統

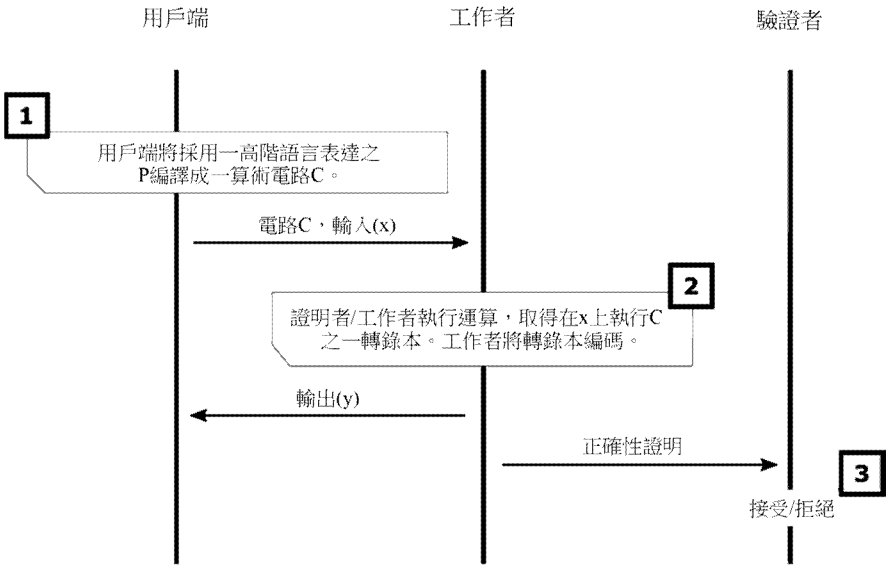
(57)摘要

本發明提供用於將高階原始碼轉換成一算術電路之系統及方法，該算術電路代表該原始碼中表達之功能性。本發明包含用於進行此轉換之一轉譯/解譯組件。在一較佳實施例中，該原始碼係一智慧合約，諸如與一區塊鏈平台有關之合約。舉例而言，本發明之使用可與比特幣網路有關。根據一實施例之方法包含以下步驟：處理一部分高階原始碼(例如：智慧合約)以產生一算術電路。該算術電路包含布置來代表該原始碼中所表達功能性其中至少一些之一或多個算術閘。該處理涉及評估該原始碼中提供之一或多個常數，以產生包括布林及/或算術運算子之一或多個表達式。該算術電

路包含連接至算術閘之n位元導線；其可用於提供一硬體及/或軟體電路。該算術電路可用於產生可在一處理器上執行之一二次程式。

The invention provides systems and methods for converting high level source code into an arithmetic circuit which represents the functionality expressed in the source code. The invention comprises a translation/interpretation component for performing this conversion. In a preferred embodiment, the source code is a smart contract such as those used in relation to a blockchain platform. The invention could be used in relation to the Bitcoin network, for example. A method in accordance with an embodiment comprises the steps of: processing a portion of high level source code (e.g. a smart contract) to generate an arithmetic circuit. The arithmetic circuit comprises one or more arithmetic gates arranged to represent at least some of the functionality expressed in the source code. The processing involves evaluating one or more constants provided in the source code to produce one or more expressions that include Boolean and/or arithmetic operators. The arithmetic circuit comprises n-bit wires connected to arithmetic gates; it can be used to provide a hardware and/or software circuit. The arithmetic circuit can be used to generate a quadratic program which can be executed upon a processor.

指定代表圖：



【圖1】

專利案號：107139685

I837103

公告本

【發明摘要】

【中文發明名稱】

電腦實施方法及系統

【英文發明名稱】

Computer-Implemented Method and System

【中文】

本發明提供用於將高階原始碼轉換成一算術電路之系統及方法，該算術電路代表該原始碼中表達之功能性。本發明包含用於進行此轉換之一轉譯/解譯組件。在一較佳實施例中，該原始碼係一智慧合約，諸如與一區塊鏈平台有關之合約。舉例而言，本發明之使用可與比特幣網路有關。根據一實施例之方法包含以下步驟：處理一部分高階原始碼(例如：智慧合約)以產生一算術電路。該算術電路包含布置來代表該原始碼中所表達功能性其中至少一些之一或多個算術閘。該處理涉及評估該原始碼中提供之一或多個常數，以產生包括布林及/或算術運算子之一或多個表達式。該算術電路包含連接至算術閘之n位元導線；其可用於提供一硬體及/或軟體電路。該算術電路可用於產生可在一處理器上執行之一二次程式。

【英文】

The invention provides systems and methods for converting high level source code into an arithmetic circuit which represents the functionality expressed in the source code. The invention comprises a translation/interpretation component for performing this conversion. In a preferred embodiment, the source code is a smart contract such as those used in relation to a blockchain platform. The invention could be used in relation to the Bitcoin network, for example. A method in accordance with an embodiment comprises the steps of: processing a portion of high level source code (e.g. a smart contract) to generate an arithmetic circuit. The arithmetic circuit comprises one or more arithmetic gates arranged to represent at least some of the functionality expressed in the source code. The processing involves evaluating one or more constants provided in the source code to produce one or more expressions that include Boolean and/or arithmetic operators. The arithmetic circuit comprises n-bit wires connected to arithmetic gates; it can be used to provide a hardware and/or software circuit. The arithmetic circuit can be used to generate a quadratic program which can be executed upon a processor.

【指定代表圖】 圖1

【代表圖之符號簡單說明】

無

【特徵化學式】

無

【發明說明書】

【中文發明名稱】

電腦實施方法及系統

【英文發明名稱】

Computer-Implemented Method and System

【技術領域】

【0001】本發明大致係有關於將高階原始碼轉換成另一形式之工具、技巧及系統之領域，例如解譯器。其亦有關於區塊鏈技術，並且特別的是，其係有關於供建置區塊鏈實施之解決方案之工具。其亦有關於機器可執行技術之改良型產生，舉例如布置來在區塊鏈上執行之智慧合約。

【先前技術】

【0002】在本文件中，「區塊鏈」一詞意指為數種類型之電子、電腦為基之、分散式分類帳中任何一者。這些包括基於共識之區塊鏈及交易鏈技術、許可及未許可之分類帳、共享分類帳以及其變例。其亦包括私用及公開區塊鏈。

【0003】雖然已提出並開發其他區塊鏈實作態樣，區塊鏈技術最廣為人知之應用仍是比特幣分類帳。儘管本揭露中為了方便及說明目的可意指為比特幣之實例，應知，本發明仍不限於與比特幣區塊鏈或相關聯協定之任何特定實作態樣或變例配合使用，並且替代之區塊鏈實作態樣及協定仍落入本發明之範疇內。

【0004】一區塊鏈是一種點對點、電子分類帳，其係

實施成一電腦為基之分散型系統，由諸區塊所構成，該等區塊進而由諸交易所構成。各交易為將區塊鏈系統中諸參與者之間一數位資產之控制轉移編碼、並且包括至少一個輸入及至少一個輸出之一資料結構。各區塊含有前一個區塊之一雜湊，諸區塊與之變為鏈接在一起，以建立所有交易之一永久、不可更改記錄，自其起始以來便已將該等交易寫入至該區塊鏈。

【0005】比特幣區塊鏈之腳本語言稱為指令碼，係屬於堆疊式。可將項目推送到堆疊頂端及從堆疊頂端取出。舉例而言，OP_EQUAL操作將最頂端兩個項目從堆疊取出、進行比較、以及將一結果(例如，若相等則為1，或若不等則為0)推送至堆疊頂端。在本發明一些實施例所運用之一些腳本語言中，可有至少兩個堆疊：一主要堆疊及一替用堆疊。

【0006】為了將一交易寫入至區塊鏈，必須對其進行「驗核」。網路節點(礦工)進行工作以確保各交易有效，並且將無效交易從網路拒絕。一節點可具有與其他節點不同之有效性標準。由於區塊鏈中之有效性是以共識為基礎，因此如果大部分節點同意一交易有效，則將該交易視為有效。安裝在節點上之軟體用戶端藉由執行一未動用交易(UTXO)之鎖定及解鎖指令碼，部分地對引用該UTXO之交易進行此驗核工作。如果鎖定及解鎖指令碼之執行評估為成立，並且滿足其他驗核條件(若適用的話)，則由該節點來驗核該交易。已驗核交易係傳播至其他網路節點，

然後一礦工節點可選擇在一區塊鏈中包括該交易。

【0007】因此，為了將一交易寫入至區塊鏈，其必須i)藉由接收交易之第一節點來驗核，如果交易經過驗核，則節點將其轉發至網路中之其他節點；以及ii)將其加入由一礦工所建置之一新區塊；以及iii)受開採，即加入過去交易之公開分類帳。當向區塊鏈加入足以使該交易實際上不可逆之區塊數時，將該交易視為經確認。

【0008】雖然區塊鏈技術在加密貨幣實作態樣之使用方面廣為人知，但數位企業家已開始探索比特幣所基於之密碼編譯保全系統、及可儲存在區塊鏈上之資料兩者之使用以實施新系統。如果區塊鏈可用於不限於加密貨幣領域之自動化任務及程序，那將會非常有利。此類解決方案將能夠利用區塊鏈之效益(例如，一永久性、防竄改事件記錄、分散式處理等)，同時在其應用方面也更加通用。

【0009】一個目前研究領域是使用區塊鏈來實施「智慧合約」。這些是設計旨在使一機器可讀合約或協議之條款自動執行之電腦程式。與採用自然語言撰寫之一傳統合約不同，一智慧合約係一機器可執行程式，其包含可處理輸入以便產生結果之規則，進而可接著依據那些結果進行動作。

【0010】在實施例中，一智慧合約「具有智慧」，概念是，智慧合約之施行及/或執行不需要建立者、或一些其他特定實體。執行或施行不需要人類互動。也就是說，雖然可在智慧合約中以特定步驟對與特定實體之互動進行編

碼，但仍可按其他方式自動執行及自我施行智慧合約。其屬於機器可讀且可執行。在一些實例中，自動執行意指為能夠動用UTXO並且具有一誘因(例如：獎勵)之任何實體。請注意，在此類實例中，能夠動用UTXO之「任何實體」意指為能夠建立解鎖指令碼而不需要證實知悉某秘密之一實體。換句話說，可在不驗證資料來源對一密碼編譯秘密(例如：私用不對稱金鑰、對稱金鑰等)具有存取權之情況下驗核動用交易。另外，在此類實例中，自我施行意指為受令根據限制條件施行動用交易之區塊鏈網路之驗核節點。在一些實例中，「動用」一UTXO意指為建立引用該UTXO並且執行為有效之一動用交易。

【0011】因此，與區塊鏈技術之使用相關之議題令人很感興趣，並非只有加密貨幣支付載具才令人感興趣。這包括物聯網(IoT)裝置、含人工智慧技巧之智慧系統等等之控制。

【0012】然而，儘管程式設計領域中有許多具有通常知識者使用諸如C、C++、Java等高階語言，能夠為區塊鏈建立應用程式之人數仍然相對較少。部分原因出在目前使用操縱主要與替用(alt)堆疊之運算碼(op_code)為區塊鏈以較低階進行寫碼之需求。特別的是，已知用於實施一區塊鏈之智慧合約之程式設計具有困難性且容易出錯，如下列文獻所探索者：Delmolino, K., et al. (2015). Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab、以及Juels, A.,

et al. (2013). The Ring of Gyges: Using Smart Contracts for Crime。

【0013】藉由能夠建立可與一區塊鏈搭配使用之解決方案及應用程式，提供一種改良型區塊鏈及相關聯平台。亦提供一種用於算術閘計算之解決方案。可將算術電路布置成用於執行為一述詞或用於一述詞之執行。本發明能夠建立及編碼一可執行程式(與僅如依據先前技術揭露提供一驗證或證明截然不同)。

【發明內容】

【0014】因此，期望在一或多項態樣中提供改善區塊鏈與分散式運算技術之方法及系統。現已擬出此一改良型解決方案。因此，根據本發明，提供有如隨附申請專利範圍中定義之方法及系統。

【0015】本發明可包括布置來在一分散型系統中提供及促進可程式規劃性之系統及方法。舉例而言，其可以是一加密貨幣系統、一區塊鏈實施之系統及/或一分散式運算系統。從一個觀點來看，其減少程式規劃過程中需要之錯誤、時間、工作量、成本及資源。因此，其為一技術問題提供一技術解決方案。從另一觀點來看，其提供一增強型區塊鏈解決方案，因為錯誤及缺陷減少。其亦保存軟體完整性。

【0016】有助益的是，與已知解譯器及編譯器不同，本發明提供一獨立於架構之解決方案。此外，其不需要使用一虛擬機器(VM)也能夠執行。

【0017】本發明可提供電腦實施之系統及方法，其採用原始碼作為輸入並產生一算術電路$\mathcal{C}$。所產生之電路C可以是HLL原始碼之功能性之一表示型態(與一驗證證明截然不同)。在受執行時，算術電路C可提供可接著受驗證之一運算之一結果。電路C可為機器可執行或經處理以在一機器上執行。原始碼可表達或代表一運算。

【0018】此電路可由將值從一欄位$\mathbb{F}$攜載至加法及乘法閘之「導線」所組成。可將算術電路具體實現為具有導線及邏輯閘之軟體、及/或一實體電路。原始碼可包含或代表採用一HLL或GPL撰寫之功能性或運算$\mathcal{P}$其中一些或全部。可將本發明描述為用於將該運算轉換成一算術電路$\mathcal{C}$之一解譯器。可將算術電路$\mathcal{C}$及一些輸入x供應給一實體以供執行。

【0019】本發明可將原始碼中表達之一運算轉換或轉譯成可(由一機器/處理器)執行之一述詞。可預編譯或預處理該(HLL/GPL)原始碼。

【0020】在一或多項實施例中，可使用一秘密值s來推導出一公開評估金鑰EK及一公開驗證金鑰VK。該方法可包含使用EK及VK對一特定輸入x評估運算之步驟。一輸出y、一或多條電路導線之值以及EK可用於產生一正確性證明π。證明π可儲存在一區塊鏈上。其可儲存在一區塊鏈交易中。該方法可包含在一區塊鏈交易中提供證明之步驟。其可包含將交易提交給區塊鏈及/或將其儲存在區塊鏈上之步驟。

【0021】該方法可包含驗證區塊鏈上所儲存證明之步驟。其可由一或多個當事方驗證。有助益的是，其可由多個當事方驗證，而不需一證明者單獨地與各該當事方互動。相較於先前技術，這提供一更有效率且更快速之解決方案。

【0022】一區塊鏈網路(例如：比特幣)中之所有或一些節點可驗證交易。

【0023】該驗證可使用公開驗證金鑰VK及證明π來進行，以便驗證一智慧合約。

【0024】本發明之一優點在於礦工根據已知、習知區塊鏈協定來驗證交易。驗證程序屬於礦工在網路裡之角色之部分。因此，本發明能夠將節點所做之努力用於技術優勢，因為驗證是作為所需、現有操作之部分來進行。相較於先前技術，這提供一有效率之布置結構。

【0025】在一些實施例中，本發明可將一工作流程之部分從採用一DSL編碼之一智慧合約形成為一二次算術程式(QAP)，如圖1所示。

【0026】如圖2所示，可將一領域特定語言(DSL)智慧合約轉換成一高階語言合約。DSL智慧合約可採用具有精確語義之一正式語言撰寫。DSL智慧合約可包括一條件集。DSL智慧合約之成果可取決於該條件集之滿足。

【0027】一HLL預編譯器(其亦可稱為一預處理器)可將HLL合約引用之外部庫併入以產生一HLL預處理合約。

【0028】HLL預編譯器在圖2中可稱為一C預編譯

器。HLL合約在圖2中可稱為一C語言合約。

【0029】然後，可根據本發明之一實施例將HLL預處理合約(即原始碼)變換成一算術電路。可將此算術電路優化以產生一已精簡算術電路，QAP多項式係推導自該已精簡算術電路。

【0030】預處理轉換可由一軟體程式(圖2中稱為C預編譯器之程式)進行，該軟體程式隨著執行，接收採用一DSL撰寫之一條件集(諸如DSL智慧合約)，並且將DSL軟體碼轉譯成HLL原始碼，諸如HLL合約。這在圖2中可稱為「C語言預處理合約」。其在本文中可簡稱為「原始碼」。HLL合約可以是一HLL程式，諸如一C++程式，其含有在DSL智慧合約中定義之合約。HLL預編譯器可以是一電腦可執行程式，其處理HLL合約及所需外部庫以產生獨立HLL預處理合約(原始碼)。

【0031】外部庫可以是由HLL合約透過調用所利用之預撰寫子程序、函數、類別、容器、值、及/或變數類型之集合。舉例而言，藉由調用外部庫，HLL合約獲得該庫之功能性，而不必實施功能性本身。HLL預處理合約可包括一表達式與運算子集合。該等運算子可包括算術運算子(例如：加法(+)、乘法(*)等)、比較運算子(例如：小於(<)、等式(==)、大於或等於(>=)等)、條件敘述(例如：若則(？，:))、或邏輯運算子(例如：AND(&&)、OR(||)、NOT(!)、XOR(⊕)等)。在一些實施例中，主函數(進入點)具有一預定義之名稱及格式。

【0032】算術電路可以是一變數集合之一有向無循環圖DAG。具有一為零之入度之DAG之每個節點可以是代表一變數(例如：x_i)之一輸入閘，並且DAG之每個其他節點可以是一和閘(+)或一積閘(×)。每個閘(節點)可具有為一之一外度，因此下層圖可以是一有向樹。算術電路可具有兩個複雜度度量：尺寸及深度。在一些實例中，一算術電路之一「尺寸」可基於算術電路內之若干閘。在一些實例中，算術電路之「深度」可基於算術電路內最長有向路徑之長度。

【0033】在產生算術電路之後，可將其精簡。

【0034】根據本發明之電腦實施之方法可包含處理一部分原始碼以產生一(可執行之)算術電路之步驟。原始碼可採用一高階程式設計語言撰寫；以及算術電路可包含一或多個算術閘，其係布置來代表原始碼中表達之一些功能性或運算。

【0035】這與先前技術中已知之編譯器及解譯器形成對比，其不將HLL原始碼轉換成一算術電路。

【0036】可在一處理器上執行算術電路。可在一區塊鏈交易中提供該電路。

【0037】這與先前技術揭露形成對比，其教示一運算或邏輯電路之驗證，及/或導致一(零知識)證明之輸出。舉例而言，*Pinocchio*系統產生一零知識證明用於驗證一般運算，請參閱“Communications of the ACM”; Vol. 59, No 2; 2016 Parno B, et al；“Pinocchio: Nearly Practical

Verifiable Computations"; 103-112 (Parno)。因此，諸如*Pinocchio*之先前技術驗證系統可搭配本發明使用以達到驗證之目的，本發明提供超越先前技術之功能性及結果。

【0038】因此，本發明提供代表及/或提供一運算之一機器可執行輸出。接著可對此進行驗證。這與例如*Pinocchio*系統形成對比，其編譯一驗證指令碼，以便*驗核*布置來進行運算之一(單獨)程式，從而包含一兩部分程序。反而，本發明使用一單部分方法，亦即將高階*運算*本身編譯成一算術電路。因此，本發明使用與此類先前技術非常不同之一方法來解決了一不同技術問題。

【0039】原始碼較佳為一智慧合約。可如上述預處理原始碼。

【0040】該處理步驟較佳為包含評估原始碼中所提供一或多個常數之子步驟。這可提供包含布林及/或算術運算子之一或多個表達式。

【0041】該方法可更包含以下步驟：使用該算術電路提供一硬體及/或軟體電路。該算術電路可包含連接至算術閘之n位元導線。該算術電路係獨立於架構。也就是說，其未布置成用於配合一特定硬體或軟體架構或平台操作或使用。其不需要使用一虛擬機器。這與具有技術特定性或需要使用一VM之先前技術編譯器及解譯器形成對比。

【0042】該方法可更包含以下步驟：預處理該原始碼以判定一或多個常數。該預處理可包含以下步驟中一或多者：

移除評論；將標頭宣告從標頭檔案匯入至原始檔案；

將多個原始檔案合併；

解決或評估指示詞及巨集。

【0043】該方法可更包含以下步驟：偵檢該原始碼中宣告之所有全域變數，其中一全域變數係有關於供執行之一函數、一結構或類別、一常數及/或一進入點。

【0044】該方法可更包含以下步驟:產生一符號(識別符)表以使該原始碼中提供之各符號(即識別符)與該原始碼中提供之宣告資訊相關聯。該表中之該等符號可以是全域及/或局部符號。

【0045】該方法可更包含以下步驟：進行該原始碼之一逐行評估，導致一算術及/或邏輯表達式，其將一或多個輸出變數表達為施用於一或多個輸入變數之邏輯及/或算術運算之一組合。

【0046】這可更包含以下步驟：類型解碼；表達式解碼；表達式評估；以及/或為該功能性所需之資料結構分配記憶體。

【0047】該方法可更包含以下步驟：將該表達式之該等算術及/或邏輯運算映射至算術閘。該映射步驟可包含以下子步驟：進行一導線擴張；以及/或進行一導線壓縮。

【0048】該方法可更包含以下步驟：使用該算術電路產生包含一多項式集合之一二次程式，該等多項式提供該電路之一描述。該方法可更包含使用一或多個輸入將該二次程式提供給一實體以執行該二次程式之步驟。

【0049】本發明亦可提供一種布置來實施或執行上述方法步驟中任何一者之系統。該系統可包含布置來進行原始碼之處理之一轉譯或轉換組件(即一解譯器/編譯器)。雖然用於產生(零知識)證明之編譯器在所屬技術領域中屬於已知(例如：如上文參考之「Pinocchio: Nearly Practical Verifiable Computations」(Parno))，這些並未依據本發明編譯成算術電路。本發明產生一答案，而不是該答案之一證明或驗證。

【0050】一種電腦實施之系統，其包含：

一處理器；以及

包括可執行指令之記憶體，該等可執行指令隨著藉由該處理器執行而令該系統進行電腦實施之方法之任何實施例、本發明之一或多項實施例/如本文中所述或訴求之方法。該等指令可包含用於提供一轉譯或轉換組件(即一解譯器/編譯器)之指令，該轉譯或轉換組件係布置來進行該原始碼之處理。

【圖式簡單說明】

【0051】一種上有儲存可執行指令之非暫時性電腦可讀儲存媒體，該等可執行指令隨著藉由一電腦系統之一處理器執行而令該電腦系統至少進行本發明之一或多項實施例/如本文中所述或訴求之方法。

本發明之這些及其他態樣將經由本文中所述之實施例而顯而易見，並且參照該實施例來闡明。本發明之一實施例現將僅以舉例方式、並且參照附圖作說明，其中：

【0052】圖1繪示本發明之一說明性使用及實施例中涉及之可驗證運算及動作者之一協定。這些動作者為：用戶端、工作者(亦稱為「證明者」)及驗證者。

【0053】圖2根據本發明之一實施例，展示從DSL合約到二次運算程式(QAP)之轉譯程序。

【0054】圖3根據本發明之一實施例，展示一算術電路之一實例。

【0055】圖4根據本發明之一實施例，展示含有一電路表示型態之一封包(標頭+主文)之一高階描述。

【0056】圖5依據下文所述之實例，展示代表「檢查變數<a>是否為偶數」敘述之一4位元導線擴張器之一實作態樣。

【0057】圖6根據本發明之一說明性實作態樣，展示一構建塊，其中該構建塊實施一條件敘述。

【0058】圖7繪示一常數產生器模組可如何根據本發明之一實施例負責建立由算術電路使用之常數。在使用之實例中，產生三個常數(C_1、C_2及C_3)加上預設值一(1)及零(0)。

【0059】圖8係一示意圖，其繪示可實施各項實施例之一運算環境。

【實施方式】

概述

【0060】現在，我們提供可如何根據一項實施例將本發明付諸工作實踐之一例示。在這項實例中，我們說明一

解譯器之一可能實作態樣，其係布置來將一高階語言契約(例如：C/C++語言)轉換成包含算術閘之一電路。然而，本發明可布置來轉譯其他HLL語言。可使用特定結構或*構建塊*來促進此轉換。在一或多項實施例中，此表示型態可視為用於建構能夠提供一分散式可驗證運算之一綜合管線之第一步驟。

【0061】這項實例中呈現之構建塊非意欲成為本發明之一實施例所處理之所有可能高階語言構造之一徹底囊括清單。此外，可提供所呈現實例之替用實作態樣。這些都落入所述技術領域中具有通常知識者之範疇內。

【0062】我們現在提供本發明之一說明性實施例。然而，值得注意的是，這是可使用本發明之一應用之一實例。所屬技術領域中具有通常知識者將理解的是，本發明可有助益地用在其他環境及應用中。本發明並不受限於與智慧合約或金融工具有關之使用。

本發明之說明性實施例及例示性使用案例

【0063】對於我們的實例，請思考允許使用者使用一領域特定語言(DSL)為金融工具產生合約之一協定。合約一旦產生，其執行便可外包給不受信賴之當事方(稱為「工作者」或「證明者」)，而其正確性則可進行公開驗證。該協定利用密碼編譯基元，其確保：

- *完備性*，亦即如果正確遵循協定，則誠實之驗證者將確信輸出之有效性；

- *健全性*，亦即沒有行騙之證明者可說服誠實之驗證者關於輸出之真確性；
- *零知識*，亦即沒有行騙之驗證者得知輸出之有效性除外之任何事。

【0064】該協定之首要效益是：

- 由於參與者之間未請求通訊，因此防止了中間人攻擊。
- 由於使用區塊鏈技術，惡意節點很難竄改資料。
- 避開諸如受信賴硬體裝置等受信賴第三方
- 合約驗核並未暗指軟體碼重新執行。不是網路中之每個節點都複製運算。反而，最誠實執行之證明儲存在公開區塊鏈中，並且僅用於驗核目的。

【0065】此一系統將能夠處理與各種類型之任務及產品對應之各種類型之智慧合約，並且不受限於金融應用或使用。(比特幣)區塊鏈由於其分散型及分散式本質的關係，為二或更多個當事方達成協議而提供一頗為適合之環境。

【0066】此一系統需要在一分散型加密貨幣系統中提供及促進可程式規劃性。然而，所屬技術領域中認知智慧合約規劃係一容易出錯之程序。請參照Delmolino, K., et al. (2015). *Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab*、以及Juels, A., et al. (2013). *The Ring of Gyges: Using Smart Contracts for Crime*。

【0067】因此，能夠使用DSL使智慧合約更易於由程式設計師撰寫及閱讀會有助益，從而在程式規劃過程中減少錯誤、縮減時間、精力、成本及資源。理想情況下，非專業程式設計師會有能力撰寫合約而無需實施任何密碼術。反而，一編譯器/解譯器會自動將原始碼編譯成使用者與區塊鏈之間的一密碼編譯協定。這些屬於本發明解決之技術問題。用於驗證及產生證明之先前技術技巧未提供此類優點。

【0068】此框架使用現代化密碼編譯可驗證運算(請參照 Gennaro, R., et al. (2013). *Quadratic Span Programs and Succint NIZKs without PCPs*)，並且確保功能評估正確：由於使用可驗證運算，取得敏感資訊之一敵手將無法操縱結果。此模型利用區塊鏈技術儲存正確性證明，並且將「正確建構」密碼術方法與智慧合約組合。

【0069】在這項實例中，我們聚焦於能夠將一高階語言契約(例如：C/C++語言)轉換成包含算術閘之一電路之一轉譯組件之一實作態樣。所產生之電路係HLL源始碼之一機器可執行表示型態(與用於驗證目的之證明截然不同)。特定結構或*構建塊*係用於促進此轉換。在我們之實例中，本發明可用於在建構一管線時提供第一步驟，該管線係布置來實施一分散式可驗證運算系統。然而，請注意，我們再次指出本發明不受限於此使用案例，並且可用於在一更廣泛應用範圍及背景中產生功效。

【0070】為了描述一說明性實作態樣，我們為可驗證

運算提供一框架之概述，然後介紹根據本發明之智慧合約之電路表示型態。

說明性使用：可驗證之運算：框架

【0071】**問題敘述**。一用戶端向一不受信賴證明者(*工作者*)發送一運算P及輸入x之規格。工作者運算一輸出y並將其回傳給用戶端。如果$y = P(x)$，則一正確證明者應該能夠產生一正確性憑證，並且說服任何人(不僅僅是用戶端)相信y之正確性。否則，驗證者應該有高機率拒絕y。

【0072】對於驗證者，該協定應該比本機執行$P(x)$更便宜，或者該協定應該處理驗證者無法自己執行之運算P。此外，不需要關於工作者行為正確性之假設。

【0073】**資料存取**。系統中存在兩個相異之分散型資料庫：(i) *DHT* - 鏈外資料係儲存在DHT上。資料在節點間充分隨機化，並且經複製以確保高可用性；以及(ii)*公開分類帳* - 正確執行之證明係儲存在區塊鏈中並且可進行稽核。

【0074】**協定**。圖1所示係協定所要求之步驟。一運算P係由一電路C表示。用戶端供應輸入x，以及工作者在輸入x上執行電路C，並且聲稱輸出為y。預期證明者取得$\{C, x, y\}$之一有效轉錄本。$\{C,x,y\}$之一有效轉錄本係對電路導線之一值指派，使得：

- 指派給輸入導線之值為x之值；
- 中間值對應於C中各閘之正確操作；

- 指派給輸出導線之值為y。

【0075】如果聲稱之輸出不正確，即$y \neq P(x)$，則$\{C, x, y\}$之一有效轉錄本不存在。

【0076】設置階段涉及採用具有一精確語義之一正式語言撰寫合約。根據本發明，解譯器將原始碼認作輸入，並且產生由將值從欄位$\mathbb{F}$攜載出來、並且連接至加法及乘法閘之諸導線所組成之一算術電路C。在一些實施例中，可利用算術電路優化技巧，諸如英國專利申請案第1718505.9號中所述之技巧，以便減少判定智慧合約之一成果期間必要之所需資源。

【0077】系統從電路C產生一二次程式Q，亦即Q含有一多項式集合，其提供原始電路C之一完整描述。接著，產生待由所有證明者及驗證者使用之公開參數。

【0078】一公開評估金鑰EK及公開驗證金鑰VK係使用由用戶端所選擇之一秘密值s來推導。工作者使用這些公開資訊來評估對一特定輸入x之運算。輸出y、內部電路導線之值、及EK係用於產生正確性證明π。證明π可儲存在區塊鏈上，並且由多個當事方驗證，而不需要證明者與這些實體中之各者單獨地互動。

【0079】所有比特幣節點都可使用公開驗證金鑰VK及證明π來驗核支付交易，從而驗核合約。

智慧合約及電路表示型態

【0080】雖然需要領域特定語言(DSL)才能實施一智

慧合約，例如Actulus建模語言(AML)、數位資產建模語言(DAML)、金融產品標記語言(FpML)，為了便於說明，我們在本文中說明一更通用語言之使用，其能夠提供類型、運算子及構造(諸如高階語言(HLL)、C)之一更高範圍。然而，本發明可布置來使用專屬工具將不同DSL語言轉換成C (或另一HLL)。

【0081】在本文所使用之實例中，解譯器處理採用C撰寫之原始碼。本發明可適用於與其他高階語言(亦可稱為「通用語言」(GPL))配合工作。通用程式設計語言之實例包括Ada、ALGOL、組合語言、BASIC、Boo、C、C++、C#、Clojure、COBOL、Crystal、D、Dart、Elixir、Erlang、F#、Fortran、Go、Harbour、Haskell、Idris、Java、JavaScript、Julia、Lisp、Lua、Modula-2、NPL、Oberon、Objective-C、Pascal、Perl、PHP、Pike、PL/I、Python、Ring、RPG、Ruby、Rust、Scala、Simula、Swift、以及Tcl。

【0082】圖2展示用於將一高階語言轉換成一邏輯電路之一綜合管線。根據本發明，我們聚焦於圖2中虛線方塊中彰顯之模組。

【0083】

- 含有合約之高階C程式與所需外部庫係連結在一起，以施作*獨立*預處理合約。在此階段，C預編譯器負責檢查是否全部所需資源都可用。亦評估預處理器指示詞。

- 評估常數表達式並註冊所有符號。成果是諸如加法(+)、乘法(*)、比較(<)、等式(==)、條件敘述(?, :)及邏輯運算子(及、或、否、互斥或)等類C語言運算子之一*表達式*集合。我們要求主函數具有一預定義之名稱及格式。算術電路(請參照圖3)係藉由以連接至例如加法及乘法等基本算術閘之*n*位元導線代表符號所建置。
- 二次算術程式(QAP)中之多項式係依據其在算術電路根處之評估來定義，如以下文獻所介紹：Gennaro, R., et al (2013) *Quadratic Span Programs and Succinct NIZKs Without PCPs*。

【0084】圖3展示一算術電路之一實例。各導線來自於一欄位F，並且所有操作係透過欄位F進行。此電路運算 $y = x_1 + x_2 \cdot x_3 \cdot (x_3 + x_4)$。由於電路之最終輸出係一總和，因此需要一附加乘法閘(乘以常數一)。

C語言解譯器

【0085】根據本發明之一項說明性實施例，我們現在說明一解譯器，其能夠辨識為C程式設計語言定義之一指令子集，包括：預處理器指示詞、條件、算術及按位元布林運算子、全域函數。如所屬技術領域中具有通常知識者輕易理解的是，對於陣列(array)及結構(struct)之支援亦可不設置有額外邏輯。

【0086】解譯器使用下文標題為「產生算術基元」之

章節中介紹之乘法、加法及特定構建塊將表達式增強成算術閘語言。各導線將具有一指定位元寬度。以1位元寬度來說明，導線代表一二進位變數。

用於在一區塊鏈上實施之類C語言程式之算術增強

【0087】我們現在詳細描述用於建構一算術電路之程序，該算術電路代表由一C原始碼表達之功能性。在該程式碼之各階段，一非預期行為(例如，遺漏符號、語法錯誤或運算子不明)將導致程式執行隨著一適當代碼錯誤立即終止。

預處理

【0088】如圖2所示，一智慧合約可由多個檔案及庫所組成。協定之第一步驟涉及建立*單一原始檔案*，其含有實施合約所需之全指令集。個別子步驟可列示如下：

- 將所有評論移除。
- 將標頭宣告從標頭檔案匯入至原始檔案。
- 將全部原始檔案合併。
- 解決或評估預處理器C指示詞及巨集。其包括#define指示詞及條件式#ifdef指示詞。

【0089】此步驟結束時，必須知道所有預處理器常數之實際值。原始碼中使用之變數之值必須***僅***取決於合約之輸入之值。此外，原始碼中*進入點*之宣告必須具有以下語法：

```
void contract(inputType *in, outputType *out)
```

【0090】類型inputType及outputType係由使用者定義。以下原始碼方塊展示一智慧合約之一簡單實例，其含有介於兩個無符號整數輸入之間的單一加總運算。

```
struct inputType { unsigned int i1; unsigned int i2; };
struct outputType { unsigned int o; };

void contract(struct inputType *in, struct outputType *out) {
    unsigned int val = in->i1 + in->i2;
    out->o = val;
}
```

【0091】輸出可由不同類型之變數表示，端視特定合約而定。在上面之例子中，單一輸出係單純地連接至算術運算之結果。

整數及實數

【0092】在此說明性實施例中，為了簡單起見，我們假設僅介於諸(帶正負號或無符號)整數之間的運算可用。如果移除此假設，則必須延伸電路構建塊(請參照「產生算術基元」章節)。因此，必須將介於諸實數之間的運算轉換成介於諸整數之間的運算。我們來思考一合約之以下部分：

「檢查員工之平均薪資是否大於3.25萬美元。」

【0093】此敘述需要一除法(除以*N*名員工)才能運算平均值。然而，可將此敘述轉換成介於諸整數之間的以下

表達式：

$$\sum_{i=1}^{N} s_i > 32500 \cdot N$$

【0094】其中s_i代表第i名員工之薪資。

建立全域符號表

【0095】在電腦科學中，一*符號表*係一種資料結構，由一編譯器/解譯器用於使原始碼中之各識別符(*符號*)與其宣告相關資訊產生關聯。在此第二步驟中，解譯器偵檢原始檔案中宣告之所有全域符號：

- 函數
- 結構(或類別)
- 請注意：類別(OOP語言，諸如C++)之使用需要附加邏輯才能檢查*public*、*protected*及*private*區段之範疇。
- 常數
- (亦允許全域變數，但不建議。*contract(...)*函數之範疇應視為一獨立黑盒子。其行為不應該取決於外部變數)。

【0096】對於這些符號中之各者，建置一局部符號階層，代表該等符號之識別符之內部宣告。此階段結束時，可直接定址表中之各全域符號(名稱、類型及值)以供進一步處理。

偵檢合約之進入點

【0097】該等全域符號之一必須是合約之進入點(「預處理」章節中之*Contract*函數)。針對期望語法檢查其參數之名稱、數量及類型。附加邏輯可包括檢查合約內之所有輸入結構是否都已用、以及所有輸出結構是否都連結至合約之一些部分。

逐行評估

【0098】各軟體碼行係獨立進行分析。局部符號代表識別符之內部宣告，並且係包括在全域符號表之階層中。更詳細地說，此階段負責以下任務：

- *類型解碼*，包括結構與陣列、基本類型(布林、整數等)及指標之宣告。
- *表達式解碼*，例如一元或二元運算、常數、識別符、資料結構及函數呼叫。
- *表達式評估*，即不取決於輸入值之(數值)表達式評估。
- *記憶體分配*，即用於合約功能性所需資料結構之暫時儲存分配。

【0099】此階段從一*空間*(即已用記憶體)及*時間*(即運算子優先順序)觀點連結算術表達式之所有敘述。因此，將各輸出變數表達為施用於輸入變數之邏輯與算術運算之組合。

建立顯式算術表達式

【0100】使用「逐行」章節中定義之資料結構，將一通用算術/邏輯表達式*e*收合，以便根據以下語法採用顯式形式表示：

$OP_N (OP_{N-1} (\ldots (OP_1 (OP_0)) \ldots))$

【0101】根據此語法，將一任意運算子OP_{i+1}施用於運算子OP_i後之*e*。

【0102】舉例來說，給定以下軟體碼：

```
void contract(struct inputType *in, struct outputType *out) {
    unsigned int val = in->i1 + in->i2 + in->i1 + in->i2;
    val = (val > 15) ? 44 + 6 : in->i1 * in->i2 * in->i2;
    out->o = val;
}
```

【0103】可將顯式表達式*e*表達為：

out_0 = (?(< 15 (ADD (ADD (ADD in_0 in_1) in_0) in_1))

50 (MUL (MUL in_0 in_1) in_1))

【0104】如「產生算術基元」章節中之說明，表達式*e*係用於建立用以代表合約功能性所需之算術基元。

產生算術基元

【0105】在此階段，解譯器已準備好在用於產生表達式*e*之運算與電路上實施這些功能性所需之結構之間進行一對一映射。

【0106】建立電路所需之一重要參數係*位元寬度*

n_{bit}，即用於代表一帶正負號(或無符號)整數之位元數量。不同電腦架構之特徵在於不同之n_{bit}值。如果一用戶端未知悉一工作者之一較佳位元寬度值，則其值將連同附加資訊予以任意選擇並且在電路之標頭中指定，如圖4所示。(正如針對一特定目標架構進行編譯，知悉位元寬度值可導致電路之實施及執行更有效率)。

【0107】*版本(version)*欄位提供與用於在電路中建立一特定構建塊之特定演算法有關之重要資訊。在我們的說明性實作態樣中，我們選擇帶正負號整數之二的補數二進位表示型態。

加法及乘法運算

【0108】加法及乘法運算係一對一映射到電路中之加法及乘法閘。給定兩個n位元導線輸入，一附加導線輸出需要$n+1$位元，並且一乘法導線輸出需要$2n$位元。舉例來說，可將介於兩條$n_{位元}$導線a與b之間的一乘法表示為：

$$\mathrm{MUL}\ [id_a\ id_b]\ \mathrm{TO}\ [id_c]$$

(電路中之每個算術或布林導線x可藉由一值id_x來單義地識別。至於二進位變數，我們從值零開始計數。結果c將自動表示在$2n_{bit}$位元上。

布林運算

【0109】全布林閘集合可使用算術閘來運算。給定兩個布林值a及b，以下等式有效：

- $\mathrm{AND}(a, b) = ab$
- $\mathrm{NAND}(a, b) = 1 - ab$
- $\mathrm{OR}(a, b) = 1 - (1 - a)(1 - b)$
- $\mathrm{NOR}(a, b) = (1 - a)(1 - b)$
- $\mathrm{XOR}(a, b) = (1 - a)b + (1 - b)a$

【0110】除了XOR運算子，各布林閘僅需要一次乘法。所有算術運算都是在1位元寬導線上進行。

【0111】n位元寬輸入上之按位元布林運算需要n個1位元乘法(對於AND)或加法(對於OR)。從最低有效輸出位元開始，接著將各元素乘以二並加到下一個元素，以建置所產生之n位元整數值(請參照「導線壓縮」章節)。

導線擴張

【0112】導線擴張通常用於將一算術導線a轉譯成一n_a位元輸出導線，其中n_a係可由a表達之最大值之以2為基數之對數。舉例來說，我們來思考一合約之以下部分：

「檢查變數<a>是否為偶數。」

【0113】如果$n_a = 4$，則使用如圖5所示之導線擴張來實施此敘述。假設a_0代表a之最低有效位元，則該敘述之輸出等於a_0本身。此電路構建塊可表達為：

$$\mathrm{EXPAND}\ [id_a]\ \mathrm{TO}\ [id_{a3}\ id_{a2}\ id_{a1}\ id_{a0}]$$

【0114】更仔細觀察該電路，清楚可知的是，進一步處理僅需要a之一個位元，而其餘1位元導線則可予以移除。解譯器可僅產生合約其餘部分中使用之個別1位元導

線。圖5展示代表「檢查變數<a>是否為偶數」敘述之一4位元導線擴張器之一實作態樣。

【0115】解譯器為優化之導線擴張器施用一特定語法：

$$\text{EXPAND } [id_a] \text{ TO } [0 \text{ -> } id_{a0}]$$

【0116】也就是說，僅採用最低有效之1位元導線(即編號為零之識別符)，並且向其指派識別符id_{a0}。n_a越大，優化可越有效。(在此上下文中，我們將*優化*定義為節省空間之可能性，該空間是用來儲存或傳輸用於代表算術電路之低階指示詞。

導線壓縮

【0117】導線壓縮係用於將1位元導線a_i組合回到一n_a位元輸出導線：

$$a = \sum_{i=0}^{n_a-1} 2^i a_i$$

【0118】此構建塊由常數之加法與乘法所組成，因此QAP多項式之尺寸不受影響(請參照Genaro R., et al (2013) *Quadratic Span Programs and Succinct NIZKs Without PCPs*)。假設$n_a = 256$且識別符在$[id_{a0}, id_{a255}]$範圍內連序，則代表此構建塊之一優化方法如下：

$$\text{COMPRESS } [id_{a0} : id_{a255}] \text{ TO } [id_a]$$

【0119】然後，藉由id_a識別所產生之導線a。

否定運算

【0120】比較兩個變數需要否定運算，因為可將這兩個變數之差與值零作比較。可將否定n_{bit}位元導線實施成乘以常數-1。此常數必須表示為：

$$-1_{n_{bit}} \triangleq \sum_{i=0}^{n_{bit}-1} 2^i$$

等於零運算

【0121】可將一n_{bit}位元導線a之構建塊實施如下：

- n_{bit}位元上之導線擴張($a_0, \ldots, a_{nbit-1}$)；
- 否定各1位元導線($a_i \rightarrow b_i$)；
- 乘以所產生之b_i條導線：$c = \prod_{i=0}^{n_{bit}-1} b_i$

【0122】因此，當且僅當$a = 0$時，1位元變數$c = 1$。

與零運算作比較

【0123】可使用簡單之方程式規則將一「*大於*」運算變換成一「*低於*」運算。在這二的補數表示型態中，此運算對應於檢查兩個帶正負號整數之差為正或負(或在「*低於或等於*」運算之情況下，檢查是否等於零)。差值$c = a - b$之正負號之判別式係採用二進位表示型態由最高有效位元x給定：負數之特徵在於$x = 1$，而正數之特徵在於$x = 0$：

EXPAND [id_c] TO [n_{bit} - 1 -> x]

【0124】取決於比較類型(正之於負)，需要否定二進位值x。

條件敘述

【0125】一高階語言中之一條件敘述可採用以下形式來表達：

$$\text{IF } (S_c)\ S_a \text{ ELSE } S_b$$

【0126】由於敘述S_c取決於合約之輸入，因此S_a與S_b兩分支都必須實施在電路中。邏輯流程如圖6所示。取決於敘述S_c之(二進位)輸出，將會執行敘述S_a或敘述S_b。二進位運算x + 1係用於否定x。

產生常數

【0127】常數值不取決於電路之輸入導線。我們使用形式為*mul-by-const-c*之專屬一元乘法閘，提供以下附加電路系統以產生合約所需之常數值：

- 常數零係藉由將一輸入導線乘以零來運算。(合約必須具有至少一個輸入。因此，具有識別符1之輸入(例如：圖7中之in_1)可用於產生常數零)。
- 常數一係藉由將常數零加一來運算。
- 任何附加常數c_i係藉由在常數一上使用*mul-by-const-ci*來運算。

【0128】由於電路總是加常數零及一，因此***k***個任意常數之實作態樣需要**k+2**個閘。此程序如圖7所示。根據本發明之一實施例提供之一常數產生器模組負責建立由算術電路使用之常數。在實例中，產生三個常數(C_1、C_2及C_3)

加上預設值一(1)及零(0)。

【0129】常數具有如二的補數標準所指定之一已知位元寬度。

【0130】現請參照圖8，提供有一運算裝置2600之一說明性、簡化方塊圖，其可用於實踐本揭露之至少一項實施例。在各項實施例中，運算裝置2600可用於實施以上所示及所述系統中任何一者。舉例而言，可為了當作一資料伺服器、一網頁伺服器、一可攜式運算裝置、一個人電腦、或任何電子運算裝置使用而組配運算裝置2600。如圖8所示，運算裝置2600可包括具有一或多個層級之快取記憶體及一記憶體控制器之一或多個處理器(集體標示為2602)，可將其組配成與包括主記憶體2608及永續性儲存器2610之一儲存子系統2606通訊。主記憶體2608可包括動態隨機存取記憶體(DRAM) 2618及唯讀記憶體(ROM) 2620，如所示。儲存子系統2606及快取記憶體2602可用於儲存資訊，諸如與本揭露中所述交易及區塊相關聯之細節。(多個)處理器2602可用於提供如本揭露所述任何實施例之步驟或功能性。

【0131】(多個)處理器2602亦可與一或多個使用者介面輸入裝置2612、一或多個使用者介面輸出裝置2614及網路介面子系統2616通訊。

【0132】一匯流排子系統2604可提供一種用於使運算裝置2600之各種組件及子系統能夠如希望彼此通訊之機制。雖然將匯流排子系統2604示意性展示為單一匯流

排，匯流排子系統之替代實施例仍可利用多條匯流排。

【0133】網路介面子系統2616可向其他運算裝置及網路提供一介面。網路介面子系統2616可當作用於從出自運算裝置2600之其他系統接收資料及將資料傳送至該等其他系統之一介面。舉例而言，網路介面子系統2616可使一資料技術人員能夠將裝置連接至一網路，使得該資料技術人員在一遠距位置(諸如一資料中心)時，可有能力將資料傳送至該裝置及從該裝置接收資料。

【0134】使用者介面輸入裝置2612可包括一或多個使用者輸入裝置，諸如鍵盤；指標裝置，諸如一整合式滑鼠、軌跡球、觸控板、或圖形輸入板；一掃描器；一條碼掃描器；一併入顯示器之觸控螢幕；諸如語音辨識系統、麥克風之音訊輸入裝置；以及其他類型之輸入裝置。一般而言，「輸入裝置」一詞之使用係意欲包括用於將資訊輸入至運算裝置2600之所有可能類型之裝置及機制。

【0135】一或多個使用者介面輸出裝置2614可包括一顯示子系統、一列印機、或諸如音訊輸出裝置等非視覺化顯示器。顯示子系統可以是一陰極射線管(CRT)、諸如液晶顯示器(LCD)之一平板裝置、發光二極體(LED)顯示器、或一投射或其他顯示裝置。一般而言，「輸出裝置」一詞之使用係意欲包括用於將資訊從運算裝置2600輸出之所有可能類型之裝置及機制。一或多個使用者介面輸出裝置2614舉例而言，可用於呈現使用者介面以在可能適當的情況下促進使用者與進行所述程序及其中變體之應用程

式互動。

【0136】儲存子系統2606可提供用於儲存基本程式設計及資料構造之一電腦可讀儲存媒體，該等基本程式設計及資料構造可提供本揭露之至少一項實施例之功能性。該等應用程式(程式、程式碼模組、指令)在受一或多個處理器執行時，可以提供本揭露之一或多項實施例之功能性，並且可予以儲存在儲存子系統2606中。這些應用程式模組或指令可由一或多個處理器2602執行。儲存子系統2606可另外提供用於儲存根據本揭露所用資料之一儲存庫。舉例而言，主記憶體2608及快取記憶體2602可為程式及資料提供依電性儲存。永續性儲存器2610可為程式及資料提供永續(非依電性)儲存，並且可包括快閃記憶體、一或多個固態驅動機、一或多個磁性硬碟機、具有相關聯可移除式媒體之一或多個軟碟機、具有相關聯可移除式媒體之一或多個光學驅動機(例如：CD-ROM或DVD或藍光)驅動機、以及其他相似之儲存媒體。此類程式及資料可包括用於實行如本揭露所述一或多項實施例之步驟之程式、以及與如本揭露所述交易及區塊相關聯之資料。

【0137】運算裝置2600可呈各種類型，包括一可攜式電腦裝置、平板電腦、一工作站、或下面所述之任何其他裝置。另外，運算裝置2600可包括可透過一或多個連接埠(例如：USB、一耳機插孔、Lightning連接器等)連接至運算裝置2600之另一裝置。可連接至運算裝置2600之裝置可包括組配來接受光纖連接器之複數個連接埠。因此，此裝

置可組配成將光學信號轉換成電氣信號，可透過將裝置連接至運算裝置2600之連接埠傳送該電氣信號以供處理。由於電腦及網路之本質不斷變化，為了說明裝置之較佳實施例，圖8所示運算裝置2600之說明僅意欲作為一特定實例。許多其他組態有可能比圖8所示系統具有更多或更少組件。

【0138】應知上述實施例說明而不是限制本發明，並且所屬技術領域中具有通常知識者將能夠設計許多替代實施例而不脫離如隨附申請專利範圍所定義之本發明之範疇。在請求項中，置放於括號內的任何參照符號不得視為限制請求項。「包含」一詞及其變體、及類似者不排除存在任何請求項或本說明書中整體所列者外之元件或步驟。在本說明書中，「包含」意味著「包括或由以下所組成」，並且「包含」之變體意味著「包括或由以下所組成」。元件之單數參照不排除此類元件之複數參照，反之亦然。本發明可藉助於包含數個相異元件之硬體、及藉由一適當程式規劃之電腦來實施。在列舉數個構件之裝置請求項中，這些構件中有數個可藉由相同硬體項目來具體實現。在互不相同之附屬項中明載某些量測的唯一事實不在於指出這些量測之一組合無法用於產生利益。

【0139】特此以參考方式併入本文中包括公布、專利申請、及專利在內所引用之所有參照，其程度猶如將各參照個別且具體地指為要以參考方式併入，並且在本文中將其完整提出。這包括以下編號之英國專利申請案：

GB1719998.5、GB 1718505.9、GB 1720768.9

【符號說明】

【0140】

2600...運算裝置

2602...處理器

2604...匯流排子系統

2606...儲存子系統

2608...主記憶體

2610...永續性儲存器

2612...使用者介面輸入裝置

2614...使用者介面輸出裝置

2616...網路介面子系統

2618...動態隨機存取記憶體

2620...唯讀記憶體

【發明申請專利範圍】

【請求項1】 一種電腦實施之方法，其包含以下步驟：

處理一部分原始碼以產生一算術電路，其中：

該原始碼係採用一高階程式設計語言撰寫，並且進一步地其中該原始碼代表一智慧合約；以及

該算術電路包含布置來代表該原始碼中所表達功能性其中一些或全部之一或多個算術閘，其中該算術電路係獨立於架構。

【請求項2】 如請求項1之方法，其中：

該算術電路係該原始碼之一機器可執行版本，並且係布置來運算一結果。

【請求項3】 如請求項1或2之方法，其中該處理步驟包含：

評估該原始碼中提供之一或多個常數，以提供包含布林及/或算術運算子之一或多個表達式。

【請求項4】 如請求項1或2之方法，且其更包含以下步驟：

使用該算術電路提供一硬體及/或軟體電路。

【請求項5】 如請求項1或2之方法，其中：

該算術電路包含連接至算術閘之n位元導線。

【請求項6】 如請求項1或2之方法，其中該方法更包含預處理該原始碼以判定一或多個常數，該預處理包含以下步驟中一或多者：

移除評論；

將標頭宣告從標頭檔案匯入至原始檔案；

將多個原始檔案合併；

解決或評估指示詞及巨集。

【請求項7】 如請求項1或2之方法，其中該方法更包含以下步驟：偵檢該原始碼中宣告之所有全域變數，其中一全域變數係有關於供執行之一函數、一結構或類別、一常數及/或一進入點。

【請求項8】 如請求項1或2之方法，其中該方法更包含以下步驟：

產生一符號表以使該原始碼中提供之各符號(即識別符)與該原始碼中提供之宣告資訊相關聯，該表中之該等符號係全域及/或局部符號。

【請求項9】 如請求項1或2之方法，其中該方法更包含以下步驟：進行該原始碼之一逐行評估，導致一算術及/或邏輯表達式，其將一或多個輸出變數表達為施用於一或多個輸入變數之邏輯及/或算術運算之一組合。

【請求項10】 如請求項9之方法，其中該逐行評估包含以下子步驟：

類型解碼；

表達式解碼；

表達式評估；及/或

為該功能性所需之資料結構分配記憶體。

【請求項11】 如請求項9之方法，且其更包含以下步

驟：

將該表達式之該等算術及/或邏輯運算映射至算術閘。

【請求項12】 如請求項11之方法，其中該映射步驟包含以下子步驟：

進行一導線擴張；及/或

進行一導線壓縮。

【請求項13】 如請求項1或2之方法，且其更包含以下步驟：

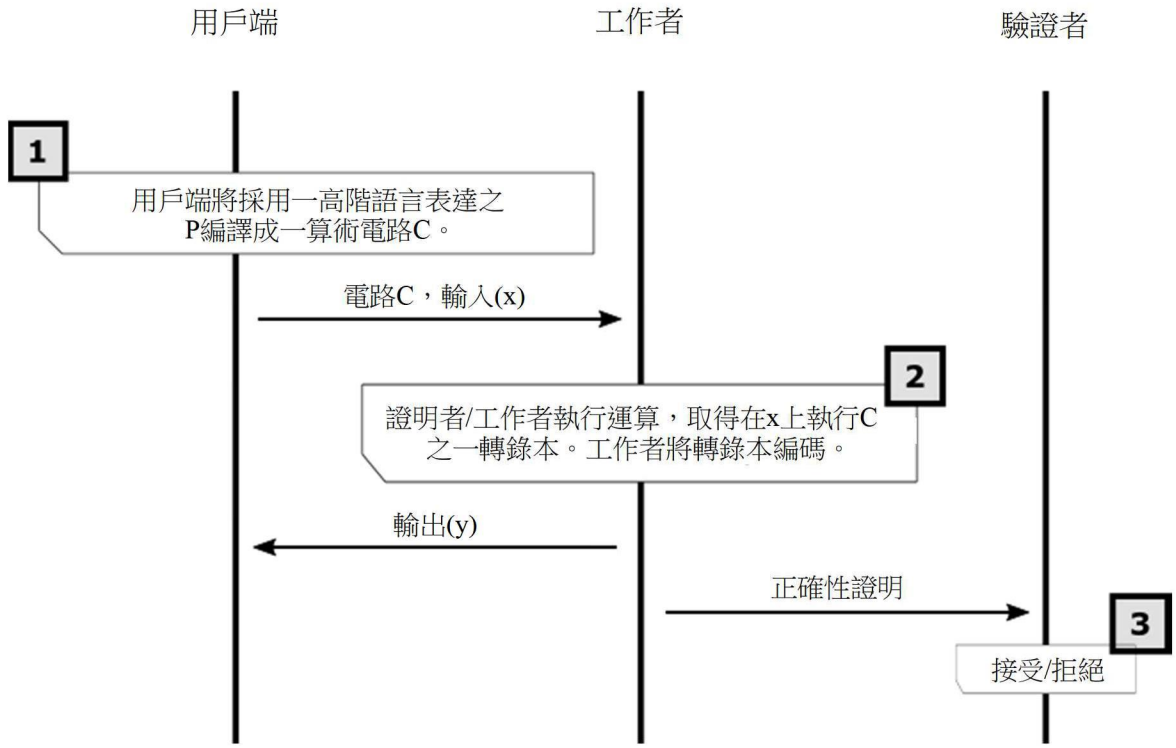
使用該算術電路產生包含一多項式集合之一二次程式，該等多項式提供該電路之一描述。

【請求項14】 如請求項13之方法，且其更包含以下步驟：

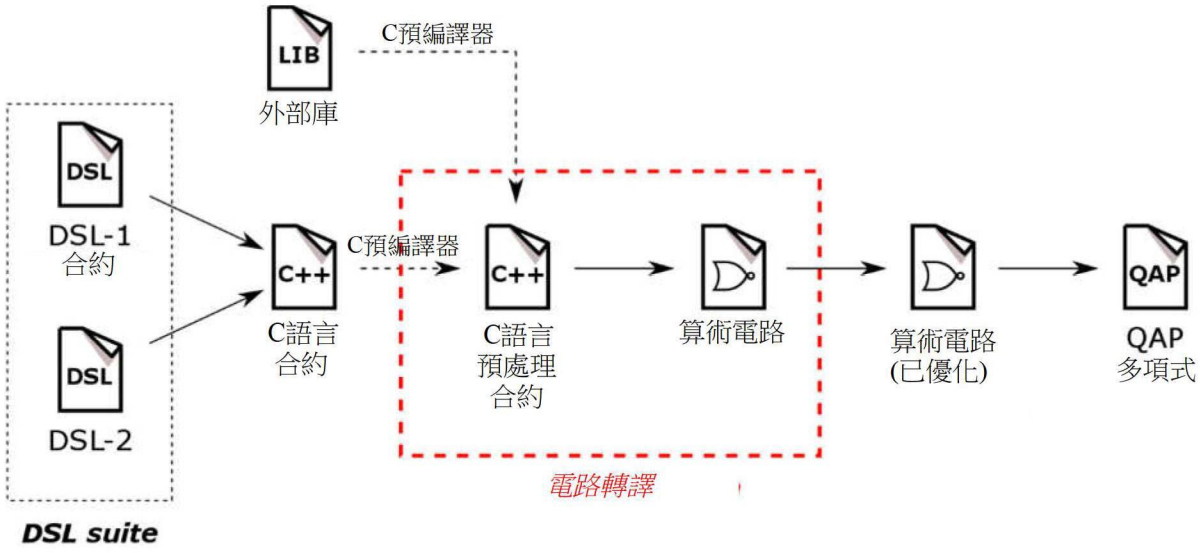
使用一或多個輸入將該二次程式提供給一實體以執行該二次程式。

【請求項15】 一種電腦實施之系統，其係布置來進行前述請求項1至14中任一項之步驟，較佳的是，其中該系統包含布置來進行該原始碼之處理之一解譯器。

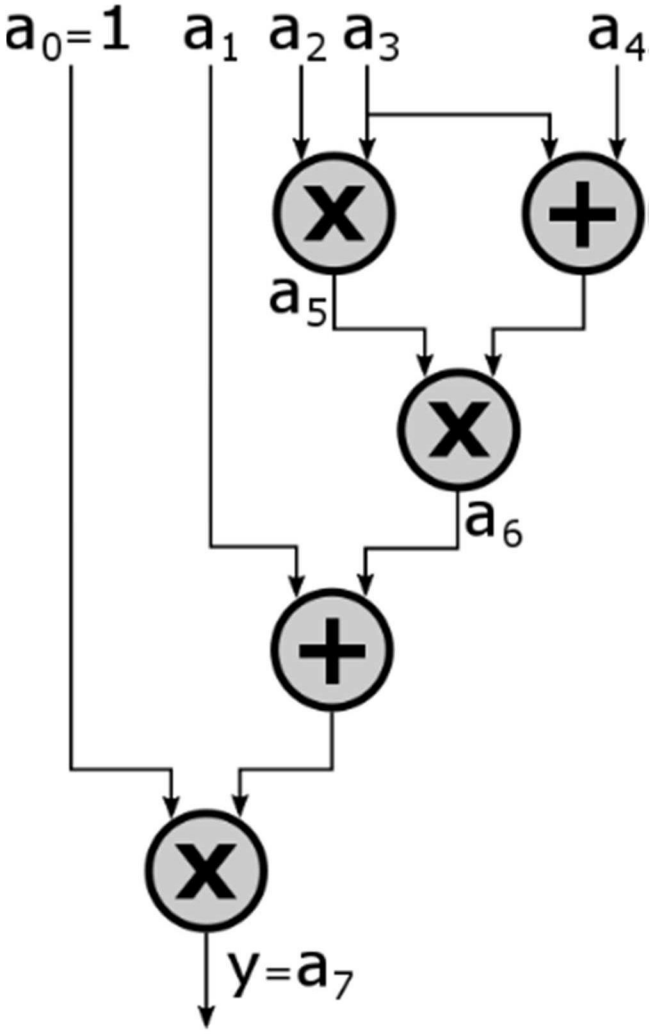
【發明圖式】


用戶端
工作者
驗證者
1
用戶端將採用一高階語言表達之
P編譯成一算術電路C。
電路C，輸入(x)
2
證明者/工作者執行運算，取得在x上執行C
之一轉錄本。工作者將轉錄本編碼。
輸出(y)
正確性證明
3
接受/拒絕

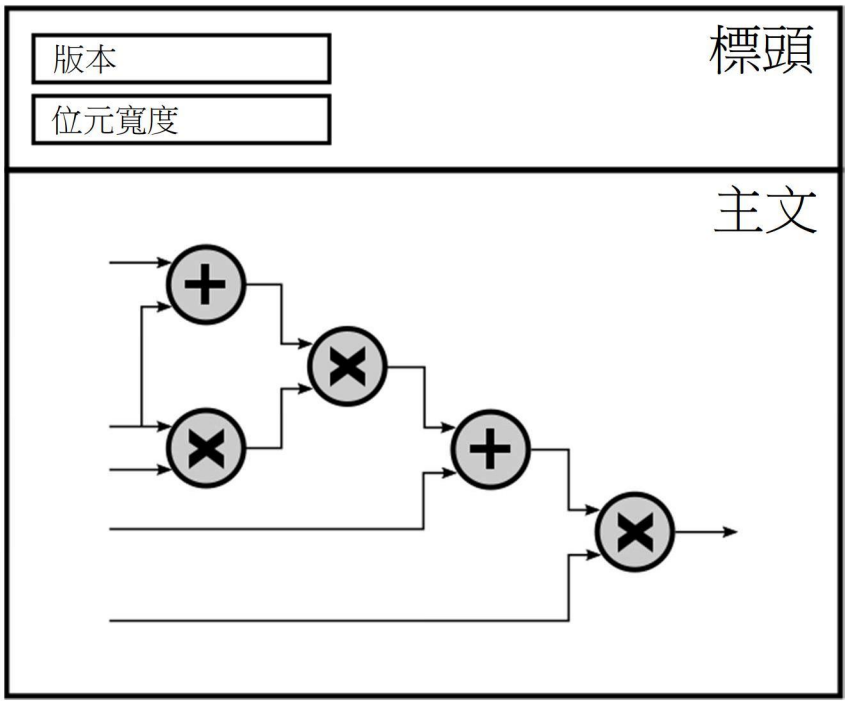
【圖1】


LIB
外部庫
C預編譯器
DSL
DSL-1
合約
DSL
DSL-2
DSL suite
C++
C語言
合約
C預編譯器
C++
C語言
預處理
合約
算術電路
算術電路
(已優化)
QAP
QAP
多項式
電路轉譯

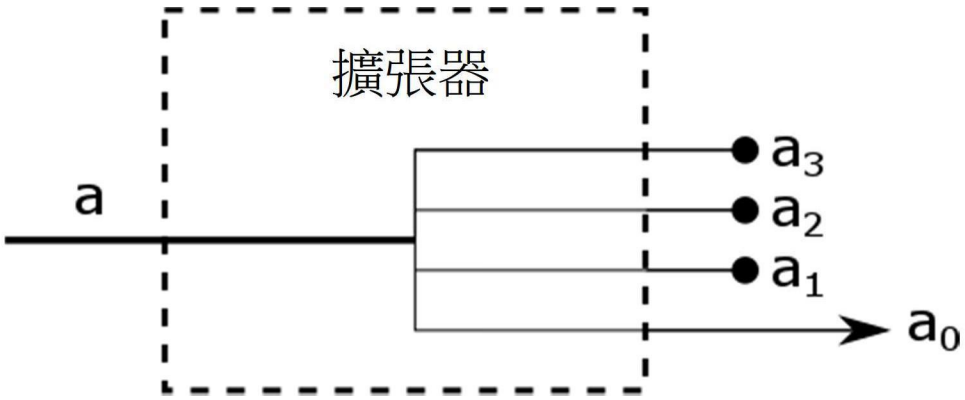
【圖2】



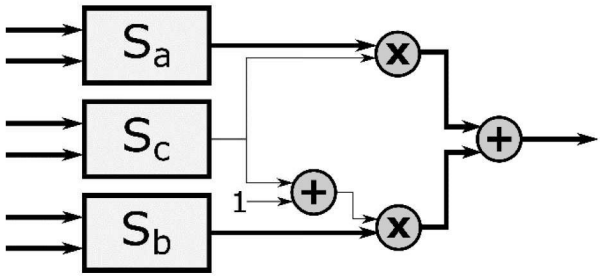
【圖3】



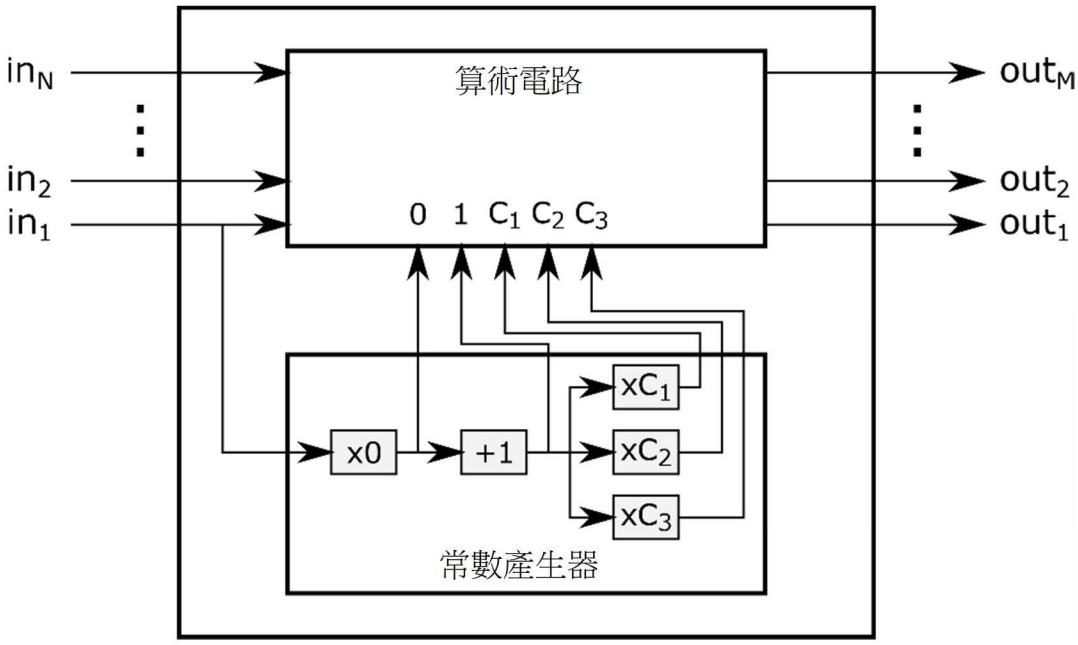
【圖4】



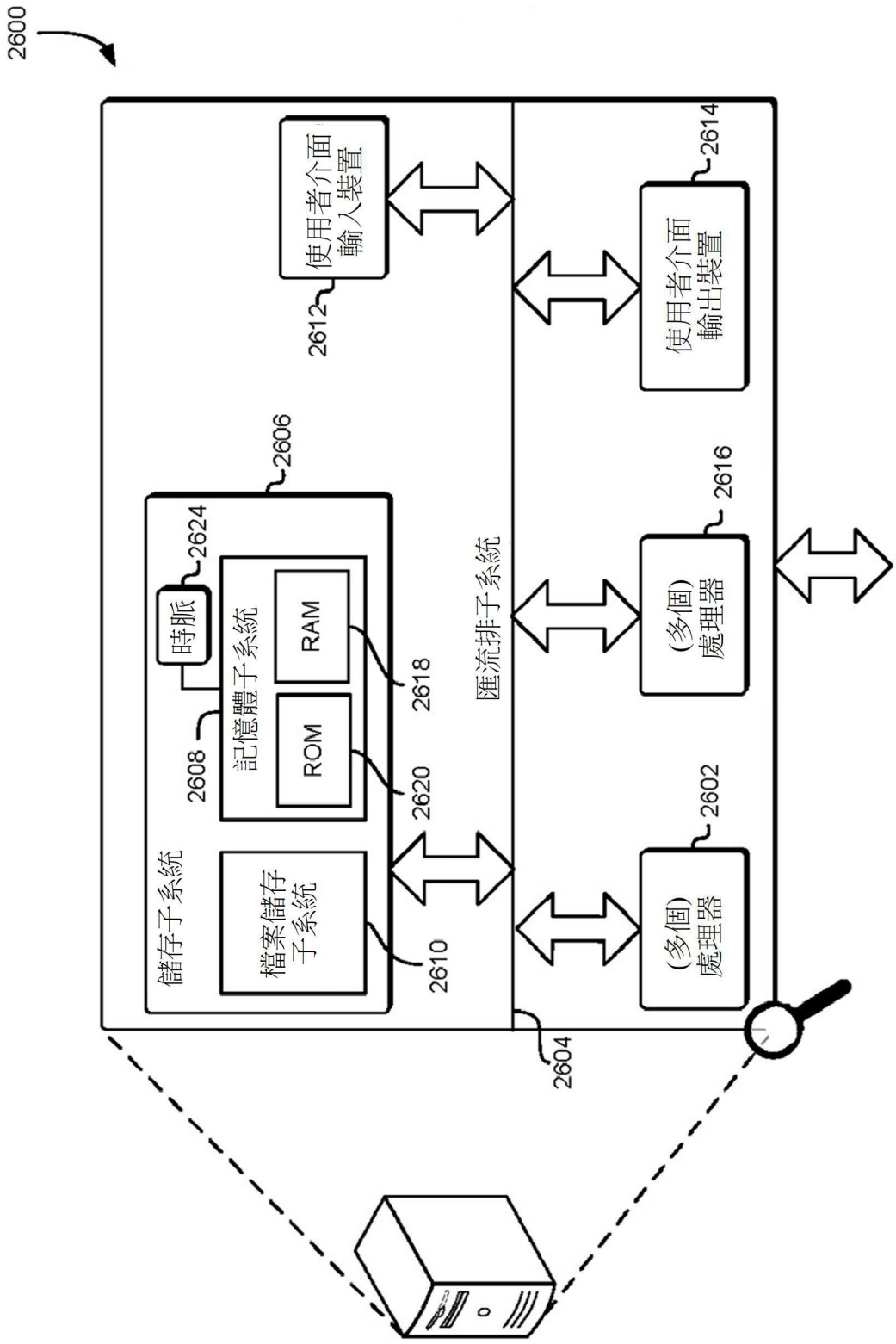
【圖5】


Sa
Sc
Sb
1

【圖6】


inN
in2
in1
算術電路
0 1 C1 C2 C3
outM
out2
out1
xC1
x0
+1
xC2
xC3
常數產生器

【圖7】


2600
使用者介面輸入裝置
2612
2614
使用者介面輸出裝置
2606
2624
時脈
記憶體子系統
RAM
ROM
2608
2618
2620
匯流排子系統
2616
(多個)處理器
儲存子系統
檔案儲存子系統
2610
2602
(多個)處理器
2604

【圖8】