公告本

發明專利說明書 571237

(填寫本書件時請先行詳閱申請書後之申請須知，作※記號部分請勿填寫)

※ 申請案號：91123691 ※IPC分類：G06F 9/00

※ 申請日期：91.10.15

壹、發明名稱

(中文) 用以開發系統晶片之開發環境的產生方法及記憶其程式之媒體

(日文) SYSTEM ON CHIPを開発するための開発環境の生成方法及びそのプログラムを記憶した媒体

貳、發明人(共 3 人)

發明人 1 (如發明人超過一人，請填**說明書發明人續頁**)

姓名：(中文) 松本 展

(英文)

住居所地址：(中文) 日本國神奈川縣海老名市柏谷707-1-104

(英文)

國籍：(中文) 日本 (英文) JAPAN

參、申請人(共 1 人)

申請人 1 (如申請人超過一人，請填**說明書申請人續頁**)

姓名或名稱：(中文) 日商東芝股份有限公司

(英文) KABUSHIKI KAISHA TOSHIBA

住居所或營業所地址：(中文) 日本國東京都港區芝浦1丁目1番1號

(英文)

國籍：(中文) 日本 (英文) JAPAN

代表人：(中文) 岡村 正

(英文) TADASHI OKAMURA

發明人 2

姓名：(中文) 大山 隆一郎

(英文)

住居所地址：(中文) 日本國東京都世田谷區奧澤1-60-11

(英文)

國籍：(中文) 日本 (英文) JAPAN

發明人 3

姓名：(中文) 內田 勝也

(英文)

住居所地址：(中文) 日本國東京都大田區田園調布本町44-4-202

(英文)

國籍：(中文) 日本 (英文) JAPAN

捌、聲明事項

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

☑ 本案已向下列國家(地區)申請專利，申請日期及案號資料如下：

[格式請依：申請國家(地區)；申請日期；申請案號　順序註記]

1. 日本；2002年04月26日；特願2002-127381
2.
3.

☑ 主張專利法第二十四條第一項優先權：

[格式請依：受理國家(地區)；日期；案號　順序註記]

1. 日本；2002年04月26日；特願2002-127381
2.
3.
4.
5.
6.
7.
8.
9.
10.

☐ 主張專利法第二十五條之一第一項優先權：

[格式請依：申請日；申請案號　順序註記]

1.
2.
3.

☐ 主張專利法第二十六條微生物：

☐國內微生物 [格式請依：寄存機構；日期；號碼　順序註記]

1.
2.
3.

☐國外微生物 [格式請依：寄存國名；機構；日期；號碼　順序註記]

1.
2.
3.

☐熟習該項技術者易於獲得，不須寄存。

玖、發明說明

(發明說明應敘明：發明所屬之技術領域、先前技術、內容、實施方式及圖式簡單說明)

相關申請案交叉參考

本申請案係以先前於2002年4月26日提出之日本特開2002-127381號公報揭示之申請案為基礎，茲聲請其優先權益，並納入其全部內容以作參考。

發明背景

1. 發明範圍

本發明係關於可利用於例如包含可配置(configurable)處理器之系統晶片之設計開發之開發環境(development environment)，特別係關於產生系統晶片之設計開發環境之方法及記憶其程式之媒體。

2. 相關技藝描述

近年來，對於適用於例如多媒體處理等之LSI(大型積體電路)之需求呈現多樣化之趨勢，另外，此LSI之市場週期也有縮短之現象，因此，一般迫切期望能夠營造可在短期內開發最適於應用程式之LSI之系統晶片之設計開發環境。在此所謂開發環境，係指構建系統晶片之際所需之硬體及系統開發支援工具等之軟體之意。

一般，在LSI內搭載有通用處理器之情形，其硬體之設計成本幾乎等於零。但因此種構成之LSI並非最適合於應用程式之需要，故難以充分發揮應用程式之性能。因此，最近，有人提供可選擇指令及記憶體構成等之可配置處理器。另外，可配置處理器之提供者並提供可指定配置而輸出可邏輯合成之硬體描述之系統。依據此種處理器及系

統，只要指定選擇指令及記憶體大小，即可在短期內開發出最適合於應用程式之構成之處理器。

另一方面，通常在改變指令系統等時，必須連編譯程式及模擬程式等軟體開發工具也一併加以變更，因此，指定配置時，在硬體描述之同時，也可提供產生軟體開發工具之系統。依據此系統，可大幅縮減軟體開發工具之設計所需之勞力及時間。

以往，對於RISC (Reduced Instruction Set Computer：簡化指令系統電腦)磁心，已開發出可配置之DSP (Digital Signal Processor：數位訊號處理器)共處理器。但此DSP共處理器僅可使用於預先準備之ISA (Industry Standard Architecture：工業標準體系)，其他均無法使用。此外，雖有可追加新的指令之可配置處理器，但也只能追加固定型式之指令，因此，難以應付例如VLIW (Very Long Instruction Word：甚長指令字)等體系之需要，使用彈性並不充分。

另外，為描述追加之指令，需使用例如Verilog之類之硬體描述語言(HDL：Hardware Description Language)，因此，無法有效地追加指令。

此外，欲依據用戶所定義之指令產生所需要之開發環境相當困難，因此，一般迫切期望開發可利用充分之彈性，簡單地產生具有高性能之硬體之LSI之開發環境之系統晶片之開發環境的產生方法及記憶其程式之媒體。

發明概述

依據本發明之一技術層面所提供之用以開發系統晶片

之開發環境的產生方法係包含分析所輸入之指令，依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生編譯程式用戶化部、彙編程式用戶化部、及模擬程式產生部，前述配置指定文件包含執行指令之硬體之指定，利用前述編譯程式用戶化部構建用以開發系統晶片之編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式。

依據本發明之另一技術層面所提供之用以產生開發系統晶片之開發環境之電腦可讀取式媒體及程式係包含分析輸入於電腦之指令，依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生編譯程式用戶化部、彙編程式用戶化部、及模擬程式產生部，配置指定文件包含執行指令之硬體之指定，利用前述編譯程式用戶化部構建用以開發系統晶片之編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式。

圖式之簡單說明

圖1係表示適用本發明之系統晶片開發裝置之概略的構成圖。

圖2係表示設定文件產生部之構成圖。

圖3係表示配置指定文件之描述例之圖。

圖4A係表示在全處理器共用之記憶區域之描述例之圖，圖4B係表示所產生之局部記憶變換表之一例之圖，圖4C係表示ISA定義文件之描述例之圖。

圖5係表示系統晶片開發環境產生部之構成圖。

圖6係表示圖5所示RTL產生部之一實施例之構成圖。

圖7係表示RTL樣本之一例之圖。

圖8係表示圖5所示之模擬程式用戶化部之一例之構成圖。

圖9A係表示模擬程式啟動選擇文件之一例之圖，圖9B係表示糾錯程式啟動選擇文件之一例之圖，圖9C係表示機械指令函數說明標題文件之一例之圖。

圖10係表示圖5所示之編譯程式用戶化部43之一實施例之構成圖。

圖11A係表示編譯程式啟動選擇文件之一例之圖，圖11B係表示驗證向量產生用之用戶定義指令文件之一例之圖。

圖12係表示本發明之系統晶片開發環境產生器4之程式構成之構成圖。

圖13係表示配置指定文件之一例之圖。

圖14係表示圖13所示之暫存器定義部之記載例之具體圖。

圖15係表示累加器之定義之一例之圖。

圖16係表示移位總量暫存器之定義之一例之圖。

圖17係表示共處理器定義之一例之圖。

圖18係表示DSP模組用配置指定文件之例之圖。

圖19係表示SIMD指令之定義之一例之圖。

圖20係表示1列所構成之描述例之圖。

圖21係表示1列所構成之描述例之圖。

圖22係表示使用指標之描述例之圖。

圖23係表示SIMD指令庫之例之圖。

圖24係表示由指令庫產生編譯程式等工具之方法之構成圖。

圖25係表示配置指定文件內之指令系統定義部之例之圖。

圖26係表示PTYPE列之描述例之圖。

圖27係表示危險資訊之描述例之圖。

圖28係表示危險資訊與流水線之動作之圖。

圖29係表示編譯程式用之調度資訊之例之圖。

圖30係表示流水線動作描述之例之圖。

圖31係表示RTL產生部之用戶定義模組之高階合成之例之圖。

圖32係表示介面之整合方法之構成圖。

圖33係表示ISA資訊之產生方法中，有關於指令CPXOR3之指令定義之例之圖。

圖34係表示自動產生之ISA資訊之圖。

圖35係表示指令CPXOR3之驗證程式之圖。

圖36係表示模擬程式產生部之概略的動作之圖。

圖37係表示系統晶片開發環境產生系統之概觀之一例之斜視圖。

發明之詳細說明

以下，參照圖示說明本發明之實施形態。

首先，利用圖1至圖10，概略地說明適用於本實施形態之系統晶片開發裝置。

(系統晶片開發裝置之構成)

圖1係表示系統晶片開發裝置1。系統晶片開發裝置1包含設定文件產生部2、用戶定義模組·用戶定義指令記憶部3、系統晶片開發環境產生部4、性能評價部5、結束判定部6、變更項目設定部7、輸出入介面部8及控制部9。

系統晶片開發裝置1連接將各種資訊輸入於裝置1內用之輸入部10、與輸出來自裝置1之各種資訊用之輸出部11。作為輸入部10，例如可使用鍵盤、滑鼠、數字鍵、觸控面板等。作為輸出部11，可適用顯示裝置及印刷裝置等。

圖2至圖13係表示上述系統晶片開發裝置1內之各部之構成。

(設定文件產生部之構成)

圖2係表示設定文件產生部2之構成。設定文件產生部2包含輸入分析部21、局部記憶變換表產生部22、選擇資訊記憶部23、裝置構成記憶部24、超高速緩衝記憶體構成記憶部25及局部記憶變換表記憶部26。設定文件產生部2依據描述系統晶片之設計上之配置之配置指定文件(以下又稱體系資料庫文件)、可變值設定資訊、最優先項目設定資訊、目標性能指定資訊、及由變更項目設定部7供應之變更項目表資訊，產生系統晶片設計上之配置，並加以儲存。前述目標性能指定資訊包含硬體性能與軟體性能之一方或雙方。作為硬體之性能指標，包含面積、頻率、耗電

量；作為軟體之性能指標，包含代碼大小、實效指令數、執行週期數。

前述輸入分析部21係用以分析配置指定文件之內容，並將配置指定文件內所描述之配置之內容分成選擇資訊記憶部23、裝置構成記憶部24、及超高速緩衝記憶體構成記憶部25之各部。

在前述配置指定文件內，描述著選擇指令之有無、裝置之有無、超高速緩衝記憶體之有無、超高速緩衝記憶體之大小、用戶定義指令、用戶定義硬體模組、LSI內部之記憶變換表、多處理器構成(多處理器之數等)等之資訊。又，在用戶定義指令及用戶定義硬體模組之描述部分，利用絕對總線及相對總線等之描述，指定描述用戶定義指令及用戶定義模組之文件名之記憶位置(在本實施形態中，為用戶定義模組•用戶定義指令記憶部3之位置)。

在前述裝置之有無中，裝置中包含指令記憶體、資料記憶體、共處理器，在指令記憶體、資料記憶體之可變項目中，包含大小、位址。在前述記憶變換表中，記憶變換表包含外部記憶體區域、內部記憶體區域、輸出入區域，外部記憶體區域也包含超高速緩衝記憶體存取區域之指定項目，且各記憶體區域也包含等待時間資訊。

又，由配置指定文件指定系統晶片設計上之配置之際，用戶可由設定文件產生部2內之各記憶部23～26所定義之可變值中選擇1個值。將此選擇之值描述於例如圖3所示之配置指定文件內。

即，例如，在以下所示之SIZE描述中，內建資料記憶體之大小可選擇1、2、4、8、16、32 [KB]中之一個。省略此SIZE描述時，其大小表示缺席值之8 [KB]。

DMEM:

SIZE Ptype={1、2、4、8、16、32},default=8, comment="內建資料記憶體之大小";

又，在以下所示之ISA_DEFINE描述中，表示作為用戶定義指令之ISA定義文件名，必須描述任意之文字串。

UCI:

ISA_DEFINE Ptype=string. default="", comment="用戶定義指令之ISA定義文件名";

又，上述配置之設定作業也可利用經由輸出入介面部8之對話式處理來設定。以對話式施行配置之設定時，用戶不必介意配置指定文件之文法等，要設定配置較為容易。因此，配置之設定所需之時間可以縮得比直接在文件中描述之情形更短。另外，全部之配置之設定可利用對話方式加以處理，因此，可防止配置中之1個忘了設定等之錯誤。

又，也可將可變值設定於配置之中。將可變值設定於配置之中時，設定文件產生部2可利用基本設定值、為測定可變值而製作之樣本值、及用戶所設定之值中之一種，自動地導出另一配置之設定值。將可變值設定於配置之中時，用戶可同時獲得對應於配置之可變範圍之多數之開發環境、驗證環境，因此可實現縮短LSI開發週期。

另外，也可在配置之中，設定最優先項目。在配置之中，

設定最優先項目時，設定文件產生部2就最優先項目以外之部分，產生對最優先項目最合適之配置狀態。為了產生此狀態，需使用基本設定值、及依據最優先項目使用樣本值或用戶所指定之設定值。在配置之中，指定最優先項目時，用戶不必辨識系統晶片之構成，即可利用適於優先項目之構成，形成開發環境、驗證環境，並加以評價。

(局部記憶變換表產生部之構成)

局部記憶變換表產生部22係用以統合由配置指定文件指定之整體變換表、與系統晶片內之各個處理器之局部記憶資訊，以產生系統晶片內之每1處理器之局部記憶變換表。前述整體變換表係共通使用於例如圖4A所示之全部處理器之記憶區域。又，作為前述局部記憶資訊，例如包含指令記憶體、資料記憶體、指令超高速緩衝記憶體、資料超高速緩衝記憶體。圖4B係表示所產生之局部記憶變換表之一例之圖。此產生之局部記憶變換表係儲存於局部記憶變換表記憶部26。

在此，處理器之局部記憶區域可利用下列方式產生：即，對各個局部記憶體所預約之整體變換表中之記憶區域，利用反映配置指定文件所指定之記憶體大小之方式所產生。又，事先在整體變換表中指定各處理器之靜區記憶資訊時，也可形成在各處理器之局部記憶變換表含有其他處理器之局部記憶體之靜區記憶體。

選擇資訊記憶部23係依據輸入分析部21對配置指定文件之分析結果，儲存有關內建於系統晶片之處理器之指令

系統內之選擇指令之ON/OFF(開/關)之種別之資訊。

裝置構成記憶部24係依據輸入分析部21對配置指定文件之分析結果，儲存有關內建於系統晶片之裝置之ON/OFF之種別之資訊與其大小之資訊、位址資訊、有關處理器之指令記憶體及資料記憶體之資訊、及有關共處理器等之選擇硬體之資訊。

(超高速緩衝記憶體構成記憶部之構成)

超高速緩衝記憶體構成記憶部25係依據輸入分析部21對配置指定文件之分析結果，儲存有關系統晶片之超高速緩衝記憶體之ON/OFF之種別、超高速緩衝記憶體之型式(直接型、2向型、4向型、n向型)之資訊、及有關其大小之資訊。

(局部記憶變換表記憶部之構成)

局部記憶變換表記憶部26係用以儲存有關局部記憶變換表產生部22所產生之局部記憶變換表之資訊。

(用戶定義模組•用戶定義指令記憶部之構成)

用戶定義模組•用戶定義指令記憶部3係用以儲存有關於內建於用戶定義之硬體模組及系統晶片之處理器之指令系統之用戶定義指令之資訊。在此，有關用戶定義之硬體模組之資訊最好採用RTL(Register Transfer Level：暫存器傳送位準)描述或動作描述，且有關指令之動作之資訊最好以C模型(含C++模型)描述後儲存於記憶部3內。動作描述與C模型描述也可相同。又，有關用戶定義指令之資訊最好描述並儲存於配置指定文件所指定之例如圖13所示

之體系DB文件中。

(系統晶片開發環境產生部之構成)

系統晶片開發環境產生部4如圖5所示，係包含RTL產生部41、模擬程式用戶化部42、編譯程式用戶化部43、彙編程式用戶化部44及驗證向量產生部45。此系統晶片開發環境產生部4在儲存於設定文件產生部2內之配置之全部組合中，產生系統晶片之硬體、驗證環境及開發設計工具。

以下，詳細說明此系統晶片開發環境產生部之內部構成。

(RTL產生部之構成)

RTL產生部41係依據記憶於設定文件產生部2之記憶部內之配置，產生內建於系統晶片之處理器之RTL描述。

圖6係表示RTL產生部41之一實施例。區塊連接部41d係參照儲存於裝置構成記憶部24與超高速緩衝記憶體構成記憶部25內之配置，而由RTL樣本41a、41b中，選擇對應於用戶所設定之配置之樣本。高階合成處理部41e係用以將儲存於用戶定義模組•用戶定義指令記憶部3內之用戶定義指令或用戶定義模組之規格之動作描述施以高階合成而產生RTL描述。另外，連接部41d係利用將前述所選擇之RTL樣本、與來自高階合成處理部41e之RTL描述連接至處理器磁心部之RTL描述41c，以產生處理器之RTL描述。又，記憶部3內之規格為RTL描述時，直接以適合介面訊號之方式連接。

又，在界定多處理器構成之定義時，區塊連接部41d產

生多數處理器描述，並產生以總線將該等描述連接之多處理器之RTL描述。

又，包含於RTL樣本41a、41b內之指令記憶體、資料記憶體、選擇硬體、指令超高速緩衝記憶體及資料超高速緩衝記憶體對用戶可能選擇之記憶體大小之各大小，事先準備有RTL描述。此等RTL描述可依照所指定之配置而選擇地被連接。又，在選擇指令方面，在選擇指令之ON/OFF之全部組合準備有對應於選擇指令之硬體樣本。

圖7係表示RTL樣本之一例之圖。在圖7中，選擇指令單元之RTL樣本針對有關除法選擇指令(DIV)之ON/OFF、及最大最小值選擇指令(MINMAX)之ON/OFF之4種組合，分別準備RTL描述，以作為樣本。此等樣本配合儲存於裝置構成記憶部24與超高速緩衝記憶體構成記憶部25內之配置，而被選擇地連接於磁心RTL描述。又，在共處理器等之選擇硬體方面，在被配置指定為有效時，已定義之RTL描述也被連接於磁心RTL描述。另外，有用戶定義之硬體時，將用戶所描述之硬體描述連接於磁心RTL描述。

利用以上之連接處理，RTL產生部41產生對應於所設定之配置之處理器之RTL描述。例如，最初構建處理器1時，用戶如果將4 [KB]之指令記憶體、與4 [KB]之資料記憶體追加至處理器磁心，以作為配置之參數指定時，即可自動地獲得處理器1之RTL描述。

又，在構建處理器2時，用戶例如將2 [KB]之指令記憶體、4 [KB]之資料記憶體、2 [KB]之指令超高速緩衝記憶

體、與4[KB]之超高速緩衝記憶體追加至處理器磁心，再將共處理器與若干選擇指令與用戶定義指令之RTL描述追加至處理器磁心，以作為配置之參數指定時，即可獲得處理器2之RTL描述。

又，如果不事先準備多數RTL樣本，而利用使配置所設定之值反映在以參數形態表現可變項目之RTL樣本中時，也可獲得RTL描述。另外，如此參數化之RTL樣本也可設在記憶體、超高速緩衝記憶體等之每1部分模組上，或者也可設置包含記憶體、超高速緩衝記憶體等之部分模組之處理器全體之RTL樣本。另外，也可使用包含全部參數之1個RTL樣本。

(模擬程式產生部之構成)

圖8係表示模擬程式用戶化部42之一例之構成圖。此模擬程式用戶化部42係用以產生執行依據配置之指令動作之模擬程式。

在圖8中，再編譯部42c係利用將儲存於用戶定義模組・用戶定義指令記憶部3內之用戶定義硬體之C模型編入模擬程式樣本42a而再編譯之方式，再構建C模型之模擬程式。又，啟動選擇資訊抽出部42d係參照儲存於設定文件產生部內之前述裝置構成記憶部24與超高速緩衝記憶體構成記憶部25之資料，產生指定啟動時之選擇對象之模擬程式啟動選擇文件(參照圖9A)。

組合處理部42e係利用組合再編譯部42c所供應之再構建之模擬程式與啟動選擇資訊抽出部42d所供應之模擬程

式啟動選擇文件，產生執行依據配置之指令動作之模擬程式。

又，模擬程式最好具備有在通過特定位址時，可輸出糾錯指令之執行結果之手段。以往，在執行途中發生錯誤時，雖有輸出錯誤之判斷錯誤程式等，但在裝置正常執行時，為調查途中經過概況，卻未在應用程式中置入輸出途中結果之指令或利用糾錯程式等停止執行而讀取途中經過概況。針對此問題，如果啟動時在通過指定位址之際，可輸出特定之資訊，則在利用模擬程式執行應用程式之途中，即可調查途中經過概況。例如，可利用下列之指令啟動模擬程式。

sim-load test。hex-printx mem(0x400) at 0x1200

在此，sim係表示系統晶片之模擬程式，-load test。hex係表示載入test。hex之16進制數之文件，-printx mem (0x400) at 0x1200係表示通過0x1200位址時，輸出記憶體之0x400位址之內容之指令。

即，模擬程式係用以記憶此自變數所指定之指示，並在程式計數器每當通過0x1200位址時，以16進制數將0x400位址之內容輸出至控制台。此時，觀測用之代碼並未被置入於應用程式中，因此，模擬程式可在完全不影響指令數及週期數等之統計資訊之情況下，觀測應用程式之執行。且不必施行在特定位址停止執行，然後，經由人手讀出並顯示記憶體之值等之作業，因此，可一面正確取得統計資訊，確認特別希望注目之途中結果，一面施行應用程式之

模擬，故可有效地開發LSI。

又，在對指定以外之區域發生記憶存取動作時，模擬程式最好停止執行。依照設定之配置啟動模擬程式時，可明確地被賦予可使用之記憶體。假使存取未安裝於基板上之區域之記憶體時，會造成例如總線錯誤，其後的動作無法獲得保證。動作無法獲得保證之程式在執行模擬程式時，有必要找出錯誤，並儘早加以修正。因此，模擬程式會指定存在之記憶區域，並禁止未被明確指定之區域之存取動作。另外，模擬程式在此禁止存取區域被存取時，會發出警告，並停止程式之動作。又，模擬程式具有對話模式(糾錯模式)，當轉移至對話模式時，即可分析何處發生錯誤。即，模擬程式在與糾錯程式通信而施行動作時，會中斷模擬動作而轉移至等待糾錯程式之指令之模式，再顯示存取無效區域之錯誤，並早期地確實向用戶傳達程式錯誤之信息。

又，記憶體之解碼電路在構成時，有時會省略掉與上位之位址訊號線之連接。此時，即使在目標執行弄錯位址之程式，也不會成為總線錯誤，但存取動作為寫入動作時，有可能會改寫到原本無意更動之位址之內容，導致其後之執行結果無法如原先之期待。此錯誤之程式在RTL模擬程式中執行時，不能立即發現錯誤。但，如提供正確之變換資訊，使模擬程式僅可執行在正確之位址之存取時，可立即對存取指定範圍以外之位址提出警告，因此，用戶可早期檢測出錯誤之程式，節省無意義之分析時間，故可縮短

開發週期。

又，糾錯程式可參照儲存於裝置構成記憶部24與超高速緩衝記憶體構成記憶部25內之資料，產生如圖9B所示之糾錯程式啟動選擇文件。利用糾錯程式啟動選擇文件，糾錯程式可施行依據配置之動作。

又，糾錯程式最好具有讀出模擬程式之統計資訊之手段，以作為假想的暫存器或變數。糾錯程式通常預先設定所謂$profile之變數，以作為特殊之變數。統計資訊係依照每1區域加以管理，此統計資訊係利用糾錯程式之變數讀出指令print加以顯示，例如為了顯示區域之第3個資訊，可利用print $profile [3]之方式加以輸入。

dbg> print $profile [3]

profile3:Count:7 Inst=123 Cycle=145

dbg>

另外，語言工具最好具有可與位址共同地輸出模擬程式用之糾錯程式用指令之手段，以作為與符號及源列同等之糾錯程式資訊(不反映於目標之機械指令中)。編譯程式中具有利用模擬程式等將途中結果等輸出至標準輸出之擴充功能。接續在#pragma consoleout後之文在目標之機械語言中不被翻譯，而與符號名及列號同樣地，被儲存於文件中，以作為糾錯資訊。當模擬程式讀入附有糾錯資訊之選擇文件時，會記憶定義#pragma文之位址資訊，同時記憶糾錯輸出資訊之printf(“a=%d¥n”, a)。當模擬程式欲執行該位址時，判斷必須輸出糾錯資訊，故將printf文送至分析

程式，以輸出糾錯資訊。

如此，即可在模擬程式執行時輸出糾錯資訊，而不會對目標之機械指令造成影響，既不會中斷模擬程式之執行，也可確認其內容。此時，模擬程式執行時所收集之統計資訊與目標所執行之資訊相同。換另一種表現方式時，等於說目標與模擬程式可使用同一標的碼。因此，不會產生因執行環境之差異而產生之再編譯等之時間，故可相對地縮短開發週期，且因共享單一標的文件，故顯然管理較為容易。

```
func(int b){
float a;
for(I=1;I<20; i++) {
        A=b/I;
#pragma consoleout printf ("a=%d¥n", a);
}
```

(編譯程式用戶化部之構成)

編譯程式用戶化部43係參照儲存於選擇資訊記憶部23與用戶定義模組•用戶定義指令記憶部3內之資料，產生包含機械指令函數說明之圖9C所示之機械指令函數說明標題文件。又，在此所稱機械指令函數說明，由於係由高級語言直接指定處理器固有之指令，故屬於描述處理器固有之指令，以作為高級語言之函數說明。

圖10係表示編譯程式用戶化部43之一實施例之構成圖。在圖10中，機械指令函數說明抽出部43c係參照儲存

於用戶定義模組•用戶定義指令記憶部3內之用戶定義指令，抽出對應之機械指令函數說明。又，結合處理部43d係參照儲存於選擇資訊記憶部23內之資料，而由已定義之樣本43b中選擇對應於有效之選擇指令之機械指令函數說明。另外，結合處理部43d利用將由機械指令函數說明抽出部43c供應之機械指令函數說明結合於前述樣本43b之方式，產生機械指令函數說明標題文件。此機械指令函數說明標題文件中包含因配置而成為有效之選擇指令與對應於用戶定義指令之機械指令函數說明。

因此，用戶可由高級語言程式直接指定配置所指定之選擇指令與用戶定義指令，並加以利用。

另外，編譯程式用戶化部43具有選擇指令抽出部43a。選擇指令抽出部43a參照儲存於選擇資訊記憶部23內之資料，取得可最適於利用之選擇指令之資訊。依據此選擇指令之資訊，編譯程式用戶化部43可產生指定編譯程式啟動時之選擇之編譯程式啟動選擇文件(參照圖11A)。

因此，可將可利用之選擇指令之資訊反映於編譯程式之最適化之中，且因編譯程式之函數庫可使用啟動選擇文件而由源程式再編譯，故可產生編入可利用之選擇指令之程式庫。

(彙編程式用戶化部之構成)

彙編程式用戶化部44係用以編入可利用之選擇指令與用戶定義指令之助記憶符號與指令型式之資訊而再構建彙編程式。依據此彙編程式用戶化部44，可對可利用之全

部指令之組合所組成之彙編程式，產生對應之標的碼。

(驗證向量產生部之構成)

驗證向量產生部45係參照設定文件產生部2內之各記憶部內所指定之配置，產生驗證指定之系統晶片之構成之驗證向量。在此，系統晶片較大時，為了在有限之時間內完成驗證，有必要對由基本之構成變更之部分施以重點式的驗證。因此，為了對指定之選擇指令與指定之大小之超高速緩衝記憶體等施以重點式的驗證，驗證向量最好依據指令系統描述，產生對應於各指令之驗證向量群(圖11B係表示驗證向量產生用之用戶定義指令文件之一例)。

其次，更具體地說明本實施形態。

本實施形態之環境可由描述處理器之指令動作之配置指定文件(體系DB文件)，自動地構建RTL、編譯程式、彙編程式、模擬程式、驗證向量、糾錯程式。此配置指定文件包含指定多數預先定義之體系中之一種之描述(體系型態之指定)。作為在此所稱之體系型態之例，包含以VLIW模態執行(以特定之管線(VLIW槽位)執行)、DSP指令。依據此體系型態之指定，可控制編譯程式之指令調度、及VLIW之並聯化機能，且調整指令系統•模擬程式之性能估計值。利用此環境，用戶可依照應用程式，獨自界定指令之定義。此用戶所定義之指令稱為用戶定義指令。又，可利用高階語言製作程式，並使用模擬程式評價其程式之性能。

另外，可對用戶定義指令，自動地構建機械指令函數之

環境，以作為基本的構成要素。利用機械指令函數，可使用編譯程式所具有之機能，例如指令調度、VLIW之並聯化(指令壓縮)、暫存器分配等。

以下，在各實施形態中具體地加以說明。

(第一實施形態)

圖12係表示本發明之第一實施形態。即圖12係表示追加圖1所示系統晶片開發環境產生器4之指令之實施例之概略構成圖，其構成與圖5所示之構成大致相同。

作為由設定文件產生部2供應之體系DB文件之配置指定文件係被供應至分析程式(parser)。此分析程式係用以分析配置指定文件。指令解釋程式係用以分析用戶指定之指令，並依照分析之指令內容，控制例如RTL產生部、模擬程式產生部、編譯程式用戶化部、彙編程式用戶化部、驗證向量產生部、糾錯程式產生部之執行順序。作為執行方式，可選擇事先登錄一連串之處理內容，而連續地執行此等之處理之成批處理、及對話式處理等。用戶指定之指令係用例如GUI(Graphical User Interface：圖解用戶介面)、或CUI(Character Base User Interface：字符庫用戶介面)等加以輸入。前述分析程式有時也利用指令解釋程式加以啟動。

圖13係表示作為體系DB文件之配置指定文件之一例。在配置指定文件中，例如可描述處理器之指定、暫存器定義、指令定義等。在處理器之指定中，例如可描述處理器之名稱及型式。在圖13中，處理器係以體系型描述其型式

(2_WAY_VLIW)。也就是說，表示在預先定義之體系中，選擇2－WAY之VLIW。

又，指令定義係由指令之助記憶符號、位元組合型式、及動作描述所組成。在此，位元組合型式包含操作碼及操作數之位元資訊。

動作描述除通常之程式之動作描述外，可描述位元分割、位元結合、符號/零擴充、溢出動作等。又，也可使用實現SWAP動作等所需之臨時變數。

又，在圖13中僅記載一部分。

依據第一實施形態，可利用指令解釋程式，依照分析之指令內容，控制RTL產生部、模擬程式產生部、編譯程式用戶化部、彙編程式用戶化部、驗證向量產生部、糾錯程式產生部之執行順序。因此，用戶可任意設定開發環境之產生順序，故可依照應用程式之開發順序任意產生開發環境。

(第二實施形態)

第二實施形態係有關前述暫存器定義部之實施形態。

圖14係具體顯示圖13所示之暫存器定義部之記載例。暫存器定義部中，可指定通用暫存器之個數、編譯程式之暫存器使用法(函數之返回值用暫存器、自變數用暫存器、接續函數調出指令所調出之函數之暫存器、不接續函數調出指令所調出之函數之暫存器、棧指示器用暫存器、整體指示器用暫存器、小型暫存器、0暫存器等)、通用暫存器之別名定義、及控制暫存器定義。暫存器使用法可以省

略，省略時，產生適用缺席規則之使用法之編譯程式。

在圖14中，通用暫存器之數GPR為16個，作為編譯程式之暫存器之使用法，函數之返回值用暫存器RET定義為暫存器0，函數自變數用暫存器ARG定義為暫存器1、2、3、4，零專用暫存器ZERO定義為暫存器12，棧指示字用暫存器SP定義為暫存器15。又，定義連接指示器LP，以作為控制暫存器。

在別名(alias)定義中，可定義暫存器之別名，不需要別名時，可加以省略。利用別名定義，用戶直接描述彙編程式時，可配合自己之喜好加以描述。例如，對應於棧指示器用暫存器SP之暫存器為暫存器15時，可定義SP作為暫存器15之別名。如此一來，在彙編碼內描述SP時，即可將SP辨識為暫存器15。

又暫存器數較少時，有時將一部分之特殊用暫存器，例如小型指示器用暫存器TP(未予圖示)當作通用暫存器加以處理。此時，使用此暫存器作為小型指示器時，描述“TP”，使用此暫存器作為通用暫存器時，則描述暫存器號碼。以此方式描述時，即可利用彙編碼，區別此暫存器到底被使用作為小型指示器，或是使用作為通用暫存器，故可構建維護性較高之彙編碼。

依據上述配置指定文件所描述之資訊，可定製編譯程式、彙編程式、模擬程式、糾錯程式、驗證向量等。即，如圖12所示，暫存器之使用法與暫存器之別名被供應至編譯程式用戶化部，暫存器名與暫存器之別名被供應至RTL

產生部，暫存器名被送交至彙編程式用戶化部、模擬程式用戶化部、驗證向量產生部、糾錯程式產生部。

例如，編譯程式用戶化部係依據暫存器定義部內所定義之通用暫存器與控制暫存器之資訊，產生編譯程式。此編譯程式具有可直接在各暫存器存取之暫存器虛擬變數之定義，用戶可利用此暫存器虛擬變數，以高階語言直接在處理器所具有之各暫存器中存取。

又，在配置指定文件中，也可對累加器及移位總量暫存器界定其定義。

圖15係表示累加器之定義之一例之圖。此例中表示有2個256位元之累加器。

圖16係表示移位總量暫存器之定義之一例之圖。此例中表示有2條移位總量暫存器。

依據上述第二實施形態，用戶可配合配置狀況，變更編譯程式之用法。例如，在應用程式中，頻繁地使用立即值“0”時，可準備經常以“0”作為其值之暫存器，如此即不需要產生形成立即值“0”用之指令，故可同時提高代碼大小及其執行性能。

又，在應用程式中，不太常使用立即值“0”時，不必準備經常以“0”作為其值之暫存器，藉以增加其他用途之暫存器數，故可降低因暫存器數不足所引起之記憶體存取頻度，因此，可同時提高代碼大小及其執行性能。

另外，暫存器定義具有缺席規則，用戶省略暫存器之使用法時，需設定缺席規則之使用法。如此，採用編譯程式

之缺席規則之使用法時，用戶可計測應用程式之性能，參考此計測之性能之結果，可以指定更好的暫存器之使用法。

又，編譯程式具有暫存器虛擬變數之定義，因此，用戶可使用暫存器虛擬變數，在暫存器定義所描述之所有暫存器，在高階語言內施行存取。此暫存器虛擬變數有利於在高階語言內在控制暫存器等特別之暫存器施行存取時使用，也可配合編譯程式之成行裝配機能而加以利用，因此，準備全部之暫存器之暫存器虛擬變數之機能可說相當有效。

另外，暫存器定義部具有別名定義，可配合用戶之喜好，定義暫存器名，因此，可構建方便使用，且維護性高之彙編碼。

(第三實施形態)

其次，說明本發明之第三實施形態，第三實施形態係有關於VLIW之實施形態。

有關於VLIW之指令之動作定義係在作為體系DB文件之配置指定文件內之指令定義部中加以界定。指令定義部中描述指令之規格、操作數(OP)碼、及動作內容。依據指令之規格、及OP碼之資訊，產生定義彙編程式之動作之彙編程式定義文件。且依據動作內容之定義，收集在指令之規格定義中未描述之資訊中之指令之動作所需之資訊，例如未顯現於指令之助記憶符號之資料之讀入及寫入所需之高暫存器及低暫存器或累加器之資訊，將其應用於機械

指令函數定義。因此，可供編譯程式施行更好之暫存器分配與指令調度。

在配置指定文件中也可界定有關共處理器之定義。即，如圖7所示之處理器2一般，可使共處理器連接於例如RISC磁心之類之處理器磁心。作為此共處理器，例如有VLIW(Very Long Instruction Word：甚長指令字)模式之DSP(Digital Signal Processor：數位訊號處理器)。

圖17係表示共處理器之定義之一例。為了與磁心指令作區別，以體系型態具有表示共處理器定義之資訊。即，描述共處理器之名稱、及共處理器之型式。在此描述之後，定義處理器定義與指令定義，以作為共處理器定義之相關資訊。共處理器定義之相關資訊除了上述處理器定義與指令定義以外，尚具有VLIW模式之安裝之有無(無、雙並聯、三並聯)、運算資料寬定義(在共處理器運算途中可獲保證之資料寬度)。

安裝VLIW模式時，在指令系統定義中含有表示共處理器指令應進入VLIW模式之槽之何處之資訊(圖17之V3)。此槽之定義依存於處理器之使用狀況加以決定，係表示在指令並聯化之際應進入哪一槽之指令之資訊。

依據此資訊產生有關共處理器指令之前述機械指令函數定義文件。前述編譯程式用戶化部可依據此機械指令函數定義文件，將用戶所定義之共處理器指令並聯化，且前述彙編程式用戶化部可依據此機械指令函數定義文件，檢查並聯化在使用上是否正確。

槽之資訊被供應至驗證向量產生部，驗證向量產生部依據槽之資訊，產生驗證對並聯化指令之處理器之動作所需之驗證向量。

運算定義寬資訊係表示在指令之動作定義內有臨時變數時，或有符號擴充運算時，模擬程式應以幾個位元來處理該臨時變數或符號擴充後之位元寬之資訊。此寬資訊依存於共處理器之暫存器定義部內所定義之暫存器之位元寬。例如，共處理器之暫存器之運算寬被指定為64位元時，運算定義寬資訊最多可指定至64位元。此運算定義寬資訊被供應至模擬程式用戶化部，並被反映於模擬程式中。

依據上述第三實施形態，用戶可擁有對已定義之全部指令之機械指令函數之定義文件。因此，藉著使用機械指令函數，用戶可獲得編譯程式之暫存器分配、指令調度、VLIW之並聯化機能，故在應用程式之開發上大大有益處。

又，槽定義具有在指令並聯化之際，表示應進入哪一槽之指令之資訊。因此，可組裝於以機械指令函數表現之指令之適正之槽中，或可供彙編程式利用適正之槽，以可促進用戶對應用程式之開發。

另外，可使指令並聯化之際所使用之槽定義之資訊反映於機械指令函數定義中，因此，安裝VLIW模式時，在將用戶定義之共處理器指令並聯化之際，可將表示應進入哪一槽之指令之資訊供應至機械指令函數定義。故編譯程式可對機械指令函數，正確地施行VLIW之並聯化。機械指

令函數因可在高階語言內使用，故可促進用戶對應用程式之開發。

又，運算定義寬資訊可反映於模擬程式，因此，模擬程式可使指令之動作適合於處理器之規格，故可使模擬程式之動作與實際之處理器之動作一致。

另外，槽資訊可反映於彙編程式，因此，在彙編程式中，可檢查用戶直接用手描述之彙編碼內之指令之並聯化部分是否一如規格所示。

又，槽資訊可反映於驗證向量，故可獲得驗證對並聯化之指令之處理器之動作所需之驗證向量。

(第四實施形態)

其次，說明本發明之第四實施形態。第四實施形態係有關DSP之實施形態。

作為以用戶定義模組編入DSP用之機能，配置指定文件具有與其他模組區別用之識別符，在此識別符內描述與DSP相關連之定義。在DSP之指令定義中，可定義任意之位元寬之暫存器。

圖18係表示DSP模組用配置指定文件之例。前頭描述著DSP_NAME之體系區別用識別符，且在暫存器定義部<dsp_register>附加暫存器寬定義REG_WIDTH。在本例之情形，暫存器寬係設定於20 bits，且在指令定義部<dsp_ISA>定義20 bits之指令。

另外，DSP模組用配置指定文件可描述未圖示之固定小數點庫之小數點之位置之資訊。此資訊被供應至編譯程式

用戶化部，並反映於編譯程式之固定小數點庫。

又，在編譯程式之DSP可描述特殊化之機能，此機能例如為複項資料型、X/Y記憶體、循環緩衝器、固定小數點資料型、位元反轉、異種暫存器組之對應中之至少1種。

依據上述第四實施形態，配置指定文件含有處理器為DSP型，硬體之指定為DSP指令型式之描述。另外，在DSP指令型式中，可設定指令之位元寬，因此，用戶可配合應用程式之用途，彈性地設定指令之位元寬。故用戶可設定符合要求之性能之DSP，減少不必要之成本。

又，在DSP指令型式中，可設定暫存器寬，因此，用戶可配合應用程式之用途，彈性地設定暫存器之位元寬，故用戶可設定符合要求之性能之DSP，減少不必要之成本。

另外，編譯程式之固定小數點庫一般係彈性地對應於固定小數點之位置，因此，性能會降低。但在本實施形態中，由於指定固定小數點庫之小數點位址之資訊可反映於編譯程式之固定小數點庫中，故可防止性能之降低。

又，用戶可使用適合編譯程式之DSP之特殊化機能，因此，可縮短應用程式之開發期間，同時提高應用程式之性能。

(第五實施形態)

第五實施形態係關於SIMD (Single Instruction Multiple Data: 單指令多資料)之實施型態。第五實施形態係將SIMD之多數資料之處理彙總成1個而加以描述，由此描述，可產生模擬程式，或對應於多媒體資料型。

即，指令可處理SIMD資料時，在配置指定文件內之指令定義部描述表示處理SIMD資料之資訊。作為此種資訊，例如為指令所處理之資料長、與各指令操作數之資料長。此資訊可指定SIMD資料寬，適用於機械指令函數定義與高階語言之SIMD資料用擴充說明符之產生。因此，用戶可容易對SIMD資料加以處理。以下係表示描述例：

```
CPPACK.B:  CRo,CRq, CRp:
                ...
                Cop: "SIMD=B, Pack=B:H:H":
                ...
```

上述例係表示“CPPACK.B”指令之資料長為8位元。即，表示暫存器為64位元時，為8個並聯，且此暫存器寬與暫存器定義部內之共處理器之暫存器寬相同。除了“B”之外，也可指定“H”(half word：半字，即16位元)、“W”(word，全字，即32位元)。另外，可依“U”之有無，區別無符號、帶符號之資料。例如，描述“BU”時，表示8位元之無符號之資料。

又，指定“Pack=B:H:H”時，表示“CPPACK.B”指令之操作數“CRo, CRq, CRp”分別為帶符號之8位元、帶符號之16位元、帶符號之16位元之資料。

圖19係表示SIMD指令之“CPADD.H”之指令定義之例。定義此指令之共處理器之暫存器例如為64位元寬，此指令之資料長為“SIMD=H”，即，帶符號之16位元。換言之，此指令係以4並聯方式計算16位元寬之資料。又，利用

“PACK=H.H.H”表示此指令之操作數“CRl, CRm, CRn”分別為帶符號之16位元之資料。另外，利用動作定義表示此指令係以4並聯方式計算帶符號之16位元之資料。此等資訊係被供應至編譯程式用戶化部、及模擬程式用戶化部。在編譯程式用戶化部中，將此等資訊反映於編譯程式之高階語言用擴充說明符之產生、機械指令函數之產生、暫存器分配、指令調度機能。又，在模擬程式用戶化部中，將此等資訊反映於模擬之動作、模擬結果之輸出機能。

圖19所示之例係以4列表示利用4並聯方式累加帶符號之16位元之資料之SIMD指令之動作描述。但此動作描述可利用1列加以描述。

以下表示描述之例。

CRl. h=CRm. h+CRn. h; … (1)

在上述例(1)中，“.h”係對應於“PACK=H, H, H”之描述。

圖20係表示上述1列所構成之描述例。

“PACK”之描述例如為“PACK=HU, B, BU”時，上述例(1)係以下列方式描述：

CRl. hu=CRm. b+CRn. bu; …(2)

圖21係表示上述1列所構成之描述例。

另外，使用指標(附標)“i”時，可簡化如下所示之資料排列不一致之描述。

CRl [15:0] = CRm [63:48] + CRn [63:48] ;

CRl [31:16] = CRm [47:32] + CRn [47:32] ;

CRl [47:32] = CRm [31:16] + CRn [31:16] ;

CRl [63:48] = CRm [15:0] + CRn [15:0] ;

簡化之描述例如下所示：

CRl [i] = CRm [3-i] + CR [3-i] ; …(3)

例(3)之情形，指標“i”係取依存於指令之資料寬與共處理器之暫存器之寬之值。“SIMD=H”、共處理器之暫存器之資料寬為64位元時，指標“i”之值為0~3。

圖22係表示使用上述指標之描述例。

依據上述第五實施形態，有關SIMD之資訊被描述於配置指定文件之指令定義部。因此，編譯程式用戶化部係依據有關SIMD指令之資訊，可產生用以產生SIMD指令之機械指令函數之定義文件。機械指令函數因可在高階語言內使用，故可促進用戶對應用程式之開發。

又，可使用在高階語言內使用用以說明SIMD資料之特殊修飾詞，故可促進用戶對應用程式之開發。

另外，模擬程式用戶化部可依據所輸入之有關SIMD指令之資訊，定製模擬程式之動作，故可產生對SIMD指令正確施行動作之模擬程式。

又，模擬程式用戶化部可依據所輸入之有關SIMD指令之資訊，定製模擬程式之結果輸出機能，故可產生對SIMD指令容易分析之模擬程式之輸出結果。

另外，SIMD指令之動作描述可將多數指令彙總成1個指令加以描述，或利用指標加以描述，故可簡化SIMD指令之指令動作之描述，因此，SIMD指令之指令動作之描述對用戶而言，可望成為直覺上容易瞭解之描述。

(第六實施形態)

第六實施形態係關於指令庫之實施形態。

用戶可追加適合配置指定文件規格之指令。本系統晶片之開發環境產生系統可保持已定義之指令，並構建成庫。以下稱之為指令庫。此指令庫係依照指令之種類構成不同之各組。用戶可利用由指令庫選擇必要之指令，以產生特殊化成為應用程式之開發環境。

圖23係表示SIMD指令庫之例。SIMD指令庫中之此等指令具有選擇性，有別於基本指令，事先準備著對此之配置指定文件與RTL。用戶可選擇使用此等指令之全部或特定之資料型之指令。

在圖23中，".xx"中描述助記憶符號"UB, B, UH, H, UW, W"中之一個，以作為指令所使用之資料型。此描述可由用戶選擇。又，如圖23所示，也可省略助記憶符號之描述。助記憶符號固定於"UB, B, UH, H, UW, W"中之一個時，也可以下列方式明示。此例係將助記憶符號固定於"H"。

CPADD. H //2-operand 32bit addition (A<-A+B)

即使在以上述方式描述之情形，變更助記憶符號之描述時，也可對應其變更。

圖24係表示由指令庫產生編譯程式等工具之方法之構成圖。用戶使用指令庫之指令時，例如由多數指令庫ADB1、ADB2，產生符合之中間文件IF1、IF2。此中間文件IF1、IF2被與基本指令之中間文件IF3合併，將此合併之中間文件編入編譯程式等時，即可構建全體之環境。

上述例係由各指令庫分別產生中間文件，並將中間文件合併。但，也可省略中間文件之產生過程，而直接合併由各指令庫選擇之必要指令。

又，為了在不變更指定之指令庫之指令機能之狀態下，獲得最適合於用戶要求之應用程式之調度性能，也可在指令設定分配型式，例如分配槽資訊之機能。

另外，在定義共處理器之際，必定需要例如共處理器與共處理器之間之資料轉送指令。如此，只要設定可使附帶之指令(預約指令)自動地合併於主要之指令，編譯程式即可由高階語言之源位準有效地產生指令庫之指令。或者也可將預約指令明示地描述於指令庫內。

另外，上述例係說明有關使合併之文件反映於編譯程式之情形。但，並不限定於此，也可利用將合併之文件供應至RTL產生部，以選擇安裝而不必變更指定之指令庫之指令機能，例如也可利用尺寸優先方式產生RTL樣本。採用此構成方式時，可維持應用程式之性能而將所欲產生之RTL尺寸控制在最小，故可抑制晶片尺寸之增大，降低成本。

又，在不必變更指定之指令庫之指令機能之狀態下，基於性能優先之考量，也可利用可施行高速動作之電路，產生RTL樣本。採用此構成方式時，可提高縮短應用程式之等待時間等性能。

依據上述第六實施形態，可對應於應用程式之需要，設定描述於指令庫之指令所處理之資料型，故可開發具有適

合於應用程式之指令之處理器。

又，即使已經決定描述於指令庫之指令所處理之資料型，也可再變更其資料型，故可變更為適合於應用程式之指令。

另外，由於可由多數指令庫選擇所需之指令，利用合併此等選擇之指令，以設定一個處理器，因此，可產生更適合於應用程式之性質之處理器。

(第七實施形態)

第七實施形態係說明有關作為體系DB文件之配置指定文件內之高階之動作之描述情形。

配置指定文件內之高階之動作描述可施行大致相當於C語言之「文」之描述。但，"!!"(位元連結)、"[]"(部分位元分離)、邏輯運算符則異於C語言。另外，本實施形態之高階之動作描述具有可在式之左邊書寫位元連結之特徵。又，本實施形態之高階之動作描述不含時鐘之概念，係描述指令執行前之暫存器及記憶體之值、與指令執行後之暫存器及記憶體之值之關係。

以下，列示本實施形態之高階之動作描述之差異與限制：

•資料型不存在。

•有可使用之變數之規則。

•可寫入位元常數。

•!!表示位元連結。

•[msb：lsb]表示部分位元分離。

•and, or, xor, nor, nand表示位元邏輯運算符。

•(Signed), (Unsigned)表示造型運算符。

•在儲存運算中，Mem Word 0等函數可寫在左邊。

(可作為變數使用者)

•操作數變數

•暫存器變數

•臨時變數

(操作數變數)

操作數變數係描述於指令定義之操作數(＜操作數＞)之變數，可施行下列描述：

＜暫存器指定＞

＜(abs)型＞＜(abs)型＞

＜disp型＞＜disp 0型＞

＜cpx＞(實數部)(虛數部)　　…複數

＜int＞　　…整數

＜flt＞　　…浮動小數點

＜立即值型＞

此等之表示暫存器之參數之字母、及指定符之位元寬必須與描述於前面之操作數之指令規格一致。

(暫存器變數)

暫存器變數係操作數變數以外之暫存器，可施行下列描述：

•特定標記(名稱指定、利用數值之索引)

例：PC, SRRO, CCR3

排列指定

例：識別文字串+'['式']'

例：CTR [Imm4]

註1：識別文字串+索引+'['數值']'為部分位元分離。

例：CTRn [4]為部分位元分離。CTR [4]相同於CTR4。也可描述CTR [Rn] [3:0]。

註2：不定標記(參數構成之索引標記、Rm等)分類於操作變數。操作變數以外之不定標記均屬錯誤。

(初級變數)

初級變數係指非操作數變數、暫存器變數之變數。初級變數之名稱可使用下列名稱：

•合乎C語言之變數名之附名稱規則者。

•不與已定義之暫存器名重複者。

•不以操作數變數之各指定符之prefix(詞頭)開頭者。

不以abs-，imm-，Imm-，disp-，code-，target-開頭者。

(常數)：

與Verilog相同，為ss...s'fnn...n。ss...s表示位元數。省略位元數時，為32 bit。f為基數。D：10進制，h：16進制，o：8進制，b：2進制。省略此等描述時，為10進制。nn...n為常數值，常數值不足位元數時，以0遞補上位位元；常數值大於位元數時，可忽略位元數。nn...n之中，不能使用段落字'_'。基數及常數值中也可使用大寫字母。

例：

3'b001：3位元之2進制數

32'Hff：32位元之16進制數(上位位元以0擴充)

'o1234：32位元之16進制數(省略位元數，與32'o1234相同)

1：32位元之10進制數(省略位元數及基數，與32'd1相同)

10'5：10位元之10進制數(省略基數，與10'd5相同)

(補足)

可描述下列事項：

a=b=c+d;

•也可寫入函數調出字。

•可在左邊寫入函數調出字。

MemWord (Rn) =a;

CTR [Imm4] =b;

•連接運算符

!!：位元連結

•部分位元分離

以[msb:lsb]指定

•預約函數名

記憶體存取函數

MemWord

MemDoubleWord

MemByte

MemHWord

MemDefaultSize

ControlBus

•位元擴充函數

SignExtension

ZeroExtension

•條件判定函數

Overflow

(左邊連結)

•位元連結寫在左邊時，以()總括起來。

(HI!!L0) =Rn*Rm;

(SIMD描述)

參照第五實施形態。

(錯誤檢查機能)

顯示錯誤檢查機能之代表性的機能：

(1) 操作數之重複檢查

(2) 依各指令型式所定之暫存器操作數之R0/RW與動作描述之整合性

(3) SIMD描述、Pack描述與動作描述之整合性：無相當於SIMD，Pack所指定之資料寬之動作描述時，輸出警報。具體地依照SIMD/Pack所指定之各資料寬調查動作描述之內容，發現均無以下之資料寬之部分位元參照/代入存在時，輸出警報。資料寬與位元寬之關係如以下所示：

資料寬	位元寬(msb-lsb+1)
B, UB	8
H, UH	16
W, UW	32

A 8, 16, 32

(4) 參照值未定義之臨時變數

在動作描述中，臨時變數最初出現之式在右邊時，輸出警告。

(5) 未參照數值已定義之臨時變數時，輸出警告。

圖25係表示配置指定文件內之指令系統定義部之例之圖。本例係表示LW指令與SWAP指令之定義。在LW指令之動作描述({}所包圍之部分)中，“Rn”為指定暫存器之操作數變數，“disp8”係disp型之操作數變數，“SP”為暫存器變數。“ZeroExtension”及“MemWord”分別為施行位元擴充與記憶體存取之預約函數。又，SWAP指令之“tmp1”與“tmp2”為臨時變數。

依據上述第七實施形態，可在配置指定文件之指令動作描述中使用高階描述，因此，欲追加新的指令較為容易，且容易分析既存之指令，故容易施行指令之改良、維護，促進用戶對應用程式之開發。

(第八實施形態)

第八實施形態係有關流水線描述。

由下列之流水線描述，可產生用以自動產生編譯程式之調度用資訊及危險驗證向量用資訊。

流水線描述包含「流水線型定義」及「危險資訊」2項：

(1) 流水線型定義

有關流水線型定義之格式如下：

PTYPE：＜型名＞：＜資源＞，＜操作＞，＜定時＞，

＜自變數＞[：＜資源＞，＜操作＞，＜定時＞，＜自變數＞....]；

各PTYPE列中，“＜資源＞，＜操作＞，＜定時＞，＜自變數＞”之四個變數最起碼必須要有一個。四個變數可利用“：”斷開而持續描述任意之數。

PTYPE列中之四個變數之意義如下：即，資源表示暫存器等，操作表示讀/寫等，定時表示流水線之階段，自變數表示對操作數(暫存器等)之指標。也就是說，PTYPE列係表示在流水線之階段產生對資源之操作之意。

圖26係表示PTYPE列之描述例之圖。

(2) 危險資訊

危險資訊之格式如下：

HAZARD：＜資源＞：＜動作1＞(型名1)，＜動作2＞(型名2)=＜週期數＞；

表示在資源中，型1之先行指令執行動作1，型2之後續指令執行動作2之情形之失控數。

圖27係表示危險資訊之描述例。遇到此危險資訊時，在某一共處理器之通用暫存器，利用流水線型“pt1”之指令施行寫入，其後，在利用流水線型“pt2”之指令讀出相同暫存器時，必須等待2個週期。

在模擬程式上執行產生危險之指令串之際之失控數決定於危險資訊、與各指令所指定之流水線定義。

例如，圖28係表示指令“cinst1”為流水線型“pt1”，指令“cinst2”為流水線型“pt2”之情形之動作。

在圖27所示之危險資訊之情形，在某一共處理器之通用暫存器，利用流水線型“pt1”之指令施行寫入，其後，在利用流水線型“pt2”之指令讀出相同暫存器時，必須等待2個週期。在圖28所示之例之情形，在階段“T”中，以指令“cinst1”在共處理器之通用暫存器“$cr1”施行寫入，接著，等待2個週期後，準備以指令“cinst2”施行同一暫存器之讀出。在此種情形下，在模擬程式中，指令“cinst2”會失控1個週期。

(編譯程式用之調度資訊)

圖29係表示編譯程式用之調度資訊之例之圖。

圖29所示之例係表示在PTYPE1之指令後面之PTYPE2之指令失控3個週期，在PTYPE2之指令後面之PTYPE3之指令失控4個週期之情形。依據此資訊，編譯程式應極力地施行調度，以便在PTYPE1與PTYPE2之間空出3個指令，在PTYPE1與PTYPE3之間空出4個指令。

(危險驗證向量產生用之流水線動作描述)

圖30係表示流水線動作描述之例之圖。

此動作描述係由流水線定義與明示性的失控之指定所組成。有關此描述及由此產生驗證向量之詳細說明在此予以省略。

依據上述第八實施形態，配置指定文件可包含流水線定義之描述。因此，可構建流水線，且在流水線定義中，可包含流水線型之定義、及危險資訊。

又，配置指定文件在有關流水線方面，可包含編譯程式

用之調度資訊，因此，編譯程式可依據調度資訊施行調度動作。

另外，配置指定文件在有關流水線方面，可包含危險驗證向量產生用之描述，因此，編譯程式可確實施行流水線之危險驗證。

(第九實施形態)

第九實施形態係有關RTL之合成之實施形態。

圖31係表示圖6所示之RTL產生部之用戶定義模組之高階合成之情形。即，圖31所示之用戶定製指令之高階合成除了可施行通常之硬體資源分配與調度之外，尚可施行磁心部及控制總線與記憶體之介面(I/F)之整合及產生。

圖32係表示用戶定製指令之合成中之介面之整合方法。可利用圖31所示之高階合成產生指令本體之RTL。此時，指令本體部分與磁心部之間需另行產生介面電路。磁心部與介面電路之間之連接係利用輸出入之暫存器場之訊號、16位元之操作數訊號、失控訊號加以表現。指令本體係以由此介面電路被調出之型式加以描述。指令本體與介面電路之間之連接係利用module(模組)之自變數加以表現。

依據上述第九實施形態，可利用通常之硬體資源分配與調度、及磁心部及控制總線與記憶體之介面之整合，施行用戶定製指令之高階合成。

(第十實施形態)

第十實施形態係關於驗證向量產生之情形之實施形態。

茲就由前述配置指定文件產生體系驗證程式(AVP：Architecture Verification Program)之產生用資料之方法加以說明。

(AVP產生所需之資料)

在此，特別將驗證處理器之指令單體之機能用之驗證程式稱為AVP。為了產生AVP，需使用ISA資訊。ISA資訊中需要以下所述之資料：

•助記憶符號

•操作數排列

•操作數資訊

操作數之種類(暫存器/立即值)

操作數之位元寬

操作數為暫存器時，參照/代入之有無

(ISA資訊之產生方法)

茲說明有關由體系資料庫文件產生ISA資訊之方法。暫存器定義部中，按照每1暫存器種類定義著暫存器之數、暫存器之位元寬等。指令定義部由助記憶符號、操作數排列、及指令動作定義部等所構成。指令動作定義部係利用描述表現於前述操作數排列之操作數、未表現於暫存器排列之暫存器、使用臨時變數、常數等之文之方式定義指令動作。

圖33係表示有關例如指令CPXOR3之指令定義。指令CPXOR3係表示有必要描述CRo、CRq、CRp，以作為暫存器操作數。暫存器名之接頭語因暫存器之種類而異，以

“CR”開頭之暫存器表示共處理器之通用暫存器。以‘{’與‘}’圍成之部分係指令動作描述部。在本例之情形，可知本指令係將計算2個共處理器之通用暫存器CRp、CRq之各位元之“異”值之結果代入CRo之指令。

如前所述，指令定義部具有助記憶符號之資訊及操作數排列之資訊，此等之資訊直接成為ISA資訊之助記憶符號之資訊及操作數排列之資訊。操作數排列中記載有操作數名之表。操作數名中，Imm＊，Imm＊係立即值，其他為暫存器。使用此規則，可由操作數排列取得各操作數之種類。

各操作數之資訊利用以下方式取得。操作數為暫存器時，由暫存器定義資訊取得暫存器之位元寬。操作數為立即值時，由操作數碼取得立即值之位元寬。具體而言，操作數碼之文字串中，‘i’之個數為立即值之位元寬。

操作數為暫存器時，操作數之參照/代入之有無係利用分析指令動作描述之語法之方式所取得。

以程式安裝上述方法時，可由圖33所示之指令定義部、及圖14所示之暫存器資訊，自動地形成圖34所示之ISA資訊。

(資料之使用方法)

ISA資訊係用以產生彙編程式所描述之驗證程式之資訊。首先，助記憶符號及操作數排列係以產生驗證程式中之檢查對象指令碼之目的被使用。各操作數之操作數資訊係為下列目的而被使用。

操作數為立即值操作數時，使用位元寬之資訊產生檢查

用之立即值資料。操作數為暫存器操作數，其暫存器之值被參照時，產生由位元寬之資訊等施加至暫存器之值，再由此產生之值產生設定於暫存器之代碼。產生此值之方法有下列幾種。

第一種方法係在可設定之位元寬(含符號之數值)中隨機產生數值之方法。

第二種方法係求出可設定輸入之區域(有2個輸入時，為2維平面上之區域)，將該區域依次細分，而以此細分後之區域之重心座標作為輸入值之方法。

利用第二種方法，可確實選擇區域之點，且利用重點式地以交界上及交界附近之點為輸入資料而產生數值時，可施行交界條件之檢查。

有關產生由ISA資訊驗證各個指令動作用之驗證程式之方法之詳細內容，礙於說明上篇幅的關係，在此省略其說明。

操作數為暫存器操作數，且將數值代入該暫存器時，可產生將代入於暫存器之指令之運算結果輸出至記錄文件之代碼，可由圖34所示之ISA資訊自動產生圖35所示之驗證程式。

依據上述第十實施形態，驗證向量產生部可由前述配置指定文件之動作指令產生處理器之驗證向量，故可確實驗證處理器之動作，而有助於用戶對LSI之開發。

(第十一實施形態)

第十一實施形態係關於模擬程式之產生之實施形態。

茲說明由配置指定文件產生模擬程式之方法。

圖36係表示前述模擬程式產生部之概略的動作。模擬程式產生部係依據配置指定文件與預先準備之C++模型樣本產生模擬程式。即，模擬程式產生部利用編入C++模型樣本所需之資訊而產生模擬程式之主要部、模擬程式之解碼表及指令定義部。主要部係依據配置指定文件之標體部與暫存器說明部而產生，解碼表與指令定義部係依據配置指定文件之ISA部而產生。

在此，所謂解碼表，係指將程式中之代碼部之位元組合形態變換成指令串之變換表。又，所謂C++模型樣本，係指用以產生模擬程式之C++源碼之樣本，模擬程式產生部將所需之資訊代入此樣本時，即可產生C++模型。

主要部中儲存著可利用之指令格式、通用暫存器、控制暫存器之數及大小、指令間危險性等之類之模擬程式全體之資訊。

解碼表儲存著配置指定文件之ISA部之各指令定義之操作碼部。模擬程式之指令碼係利用比較此表之操作碼部分與指令之操作碼部分，以識別指令之方式而產生。

指令定義部主要係利用將配置指定文件之動作描述變換成與此相等效之動作之C++描述之方式而產生。又，在指令定義部中也被附加流水線之型式、追蹤輸出之格式、立即值操作數之符號擴充之有無、在共處理器之暫存器與共處理器之暫存器間轉送資料時之轉送方向等之資訊。

附加轉送方向之資訊之方法如下所述。

轉送方向之資訊係被定義為對各指令定義之屬性之一。在模擬程式產生部之共處理器模型文件(=mkcop之輸入)中，係以下列方式被描述。

```
{name          =>'ICMOV',
cname          =>'CMOV_1',
instType       =>'t64c_a1',
instSubType    =>'core_cop', #<=由共處理器通用暫存器轉送至磁心通用暫存器
regsbit        =>'true',
opcodeA        =>'0xf',
opcodeB        =>'0x0',
opcodeC        =>'0x0',
opcodeD        =>'0x1',
func=>'
#動作描述
',
},
```

動作描述基本上係將變數變換成C++變數，並以1比1方式，將運算符變換成與此等效之C++運算符而產生。暫存器及記憶體之讀寫及預約函數係利用調出預先準備之模擬程式之API函數而產生。

有關不在C++中之一部分之表現方面，需在施行對C++變換之前，變換動作描述。例如，記載於式之左邊之位元聯結係以以下方式變換。

(變換前)

(A!!B) =Expression;

(變換後)

TMP=Expression;

B=TMP [(B最上位•B最下位•1)：0]；

A=TMP [(A最上位+B最上位•1)：(B最上位•B最下位)]；

(TMP為臨時變數)

又，在SIMD標記方面，係以在第五實施形態所說明之方式展開。

將數值被其指令動作改寫暫存器輸出至追蹤輸出，有無寫入於暫存器可從分析該暫存器是否出現於指令之動作描述之左邊部加以判定。暫存器間之資料轉送也同樣地將出現於左邊部之暫存器判斷為寫入用暫存器，將出現於右邊部之暫存器判斷為讀出用暫存器。並視為資料由讀出之暫存器轉送至寫入之暫存器。

依據上述第十一實施形態，可依照配置指定文件之描述，產生必要之模擬程式，因此，用戶可模擬應用程式，並在短時間瞭解應用程式之性能。故可縮短開發系統晶片之期間。

(第十二實施形態)

第十二實施形態係關於糾錯環境之構建之實施形態。

糾錯環境構建用之對應項目有下列3項。

(1) 暫存器定義：

糾錯程式必須要有瞭解作為可顯示及改寫之暫存器有

何種暫存器之資訊。此資訊係由配置指定文件之暫存器定義部所產生。可顯示之暫存器係被定義之全部暫存器。可改寫之暫存器或暫存器之場(暫存器之一部分)在圖14所示之暫存器定義中，例如係被描述為“rw”之暫存器。被描述為“r”之暫存器無法利用糾錯程式加以改寫。

(2) 逆彙編程式：

糾錯程式係對執行之程式施行逆彙編程式，並加以標記起來。逆彙編程式係利用與模擬程式之解碼器部相同之方法，辨識指令，並顯示其助記憶符號。

(3) 糾錯監視器：

糾錯監視器係在糾錯開始時，用以設定暫存器之初始值。由配置指定文件之暫存器定義部獲得預備設定此初始值之暫存器(存在之暫存器)之資訊。

依據上述第十二實施形態，可由配置指定文件構建糾錯環境，因此，用戶可獲得開發系統晶片所需之糾錯環境，故用戶可對應用程式施行糾錯，並縮短開發應用程式之期間。

又，本發明並不限定於上述各實施形態。

圖37係表示應用上述各實施形態之系統晶片開發環境產生裝置之概觀。此系統晶片開發環境產生裝置50包含所謂通用電腦、工作站、PC (Personal Computer：個人電腦)、NC (Network Computer：網路電腦)等。系統晶片開發環境產生裝置50具有未圖示之硬碟裝置50a。

另外，系統晶片開發環境產生裝置50具有例如軟碟機52

及光碟機54。軟碟機52中裝有軟碟53，光碟機54中裝有光碟55。

又，連接於系統晶片開發環境產生裝置50之驅動裝置57例如為記憶卡58之讀寫器、或卡式磁帶59之讀寫器。利用此驅動裝置57存取記憶卡58、或卡式磁帶59。

前述指令解釋程式、RTL產生部、模擬程式用戶化部、編譯程式用戶化部、彙編程式用戶化部、驗證向量產生部、及糾錯程式產生部等之程式、及配置指定文件等文件係被記憶於軟碟53、光碟55、記憶卡58、或卡式磁帶59等之記憶媒體。利用讀出此等程式及文件，將其安裝於前述硬碟裝置50a，即可執行上述之動作，且也可使用上述之傳送媒體，作為記錄媒體。

有鑑於精通此技術領域者可輕易地對本發明之實施形態加以模仿或變更，獲取附加利益。因此，從廣義而言，本發明之內容不應僅限定於上述特殊細節及代表性之實施形態，從而在不背離其精神或一般發明概念下，如所附申請專利範圍等闡述之要旨之範圍內，當然可作種種適當之變更，不待贅言。

圖式代表符號說明

元件編號	中文
1	系統晶片開發裝置
2	設定文件產生部
3	用戶定義模組·用戶定義指令記憶部
4	系統晶片開發環境產生部

5	性能評價部
6	結束判定部
7	變更項目設定部
8	輸出入介面部
9	控制部
10	輸入部
11	輸出部
21	輸入分析部
22	局部記憶變換表產生部
23	選擇資訊記憶部
24	裝置構成記憶部
25	超高速緩衝記憶體構成記憶部
26	局部記憶變換表記憶部
41	RTL產生部
42	模擬程式用戶化部
43	編譯程式用戶化部
44	彙編程式用戶化部
45	驗證向量產生部
41a	RTL樣本
41b	RTL樣本
41c	RTL描述
41d	連接部
41e	高階合成處理部
42a	模擬程式樣本部
42b	模擬程式樣本

42c	再編譯部
42d	啟動選擇資訊抽出部
42e	組合處理部
43a	選擇指令資訊抽出部
43b	已定義之樣本
43c	機械指令函數說明抽出部
43d	結合處理部
ADB1	多數指令庫
ADB2	多數指令庫
IF1	指令庫之中間文件
IF2	指令庫之中間文件
IF3	基本指令之中間文件
50	系統晶片開發環境產生裝置
52	軟碟機
53	軟碟
54	光碟機
55	光碟
56	鍵盤
57	驅動裝置
58	記憶卡
59	卡式磁帶
50a	硬碟裝置

肆、中文發明摘要

本發明之用以開發系統晶片之開發環境的產生方法係包含由配置指定文件構建編譯程式之編譯程式用戶化部、構建彙編程式之彙編程式用戶化部、及構建模擬程式之模擬程式產生部，配置指定文件包含執行指令之硬體之指定。

伍、日文發明摘要

System on chip 開発環境生成方法は、コンフィグレーション指定ファイルからコンパイラを構築するコンパイラカスタマイズ部、アセンブラを構築するアセンブラカスタマイズ部、及びシミュレータを構築するシミュレータ生成部を含み、コンフィグレーション指定ファイルは、命令を実行するハードウェアの指定を含む。

陸、(一)、本案指定代表圖為：第_____圖

(二)、本代表圖之元件代表符號簡單說明：

柒、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

拾、申請專利範圍

1. 一種用以開發系統晶片之開發環境的產生方法，其係包含：

分析所輸入之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生編譯程式用戶化部、彙編程式用戶化部、及模擬程式產生部，前述配置指定文件包含執行指令之硬體之指定，利用前述編譯程式用戶化部構建用以開發系統晶片之編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

2. 如申請專利範圍第1項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含定義暫存器之構成之暫存器定義者。

3. 如申請專利範圍第2項之用以開發系統晶片之開發環境的產生方法，其中前述暫存器定義係包含定義前述編譯程式之暫存器之使用方法之資訊者。

4. 如申請專利範圍第2項之用以開發系統晶片之開發環境的產生方法，其中前述暫存器定義係包含定義前述編譯程式之暫存器之使用方法之資訊，此資訊被省略時，利用前述編譯程式用戶化部設定編譯程式之缺席規則之暫存器之使用方法者。

5. 如申請專利範圍第4項之用以開發系統晶片之開發環境的產生方法，其中前述編譯程式用戶化部係依據前

述暫存器定義之資訊，定義可直接在各暫存器存取之暫存器虛擬變數者。

6. 如申請專利範圍第2項之用以開發系統晶片之開發環境的產生方法，其中前述暫存器定義係包含暫存器之別名定義，前述暫存器之別名定義係被供應至前述彙編程式用戶化部者。

7. 如申請專利範圍第1項之用以開發系統晶片之開發環境的產生方法，其中前述開發環境進一步包含RTL產生部、驗證向量產生部、糾錯程式產生部中至少一種者。

8. 如申請專利範圍第7項之用以開發系統晶片之開發環境的產生方法，其中前述暫存器定義包含暫存器名，此暫存器名被供應至前述編譯程式用戶化部、彙編程式用戶化部、模擬程式用戶化部、RTL產生部、驗證向量產生部、及糾錯程式產生部者。

9. 如申請專利範圍第2項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含指令之規格、操作數碼、動作內容者。

10. 如申請專利範圍第2項之用以開發系統晶片之開發環境的產生方法，其中前述編譯程式用戶化部係依據前述指令定義部之資訊產生機械指令函數定義者。

11. 如申請專利範圍第1項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件之處理器係VLIW型，硬體之指定係包含VLIW之指令型式之指定、及VLIW之槽之指定中一方者。

12. 如申請專利範圍第11項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含槽定義，其係在將指令並聯化之際，用以區別置於何槽之指令者。

13. 如申請專利範圍第11項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含槽定義，其係使將指令並聯化之際所使用之槽資訊反映於機械指令函數定義者。

14. 如申請專利範圍第11項之用以開發系統晶片之開發環境的產生方法，其中依據處理器之規格指定指令動作描述內之臨時變數及符號擴充後之值之位元寬，將其資訊供應至模擬程式用戶化部，使其反映於前述模擬程式者。

15. 如申請專利範圍第13項之用以開發系統晶片之開發環境的產生方法，其中將前述槽資訊供應至彙編程式用戶化部，使其反映於前述彙編程式者。

16. 如申請專利範圍第15項之用以開發系統晶片之開發環境的產生方法，其中將前述槽資訊供應至前述驗證向量產生部，使其反映於前述驗證向量者。

17. 一種用以開發系統晶片之開發環境的產生方法，其係包含：

分析所輸入之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生編譯程式用戶化部、彙編程

式用戶化部、及模擬程式產生部，前述配置指定文件之處理器為DSP型，硬體之指定包含DSP指令型式之描述，利用編譯程式用戶化部構建編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

18. 如申請專利範圍第17項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含指定任意位元寬之指令之資訊者。

19. 如申請專利範圍第17項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含定義任意位元寬之暫存器之資訊者。

20. 如申請專利範圍第17項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含指定固定小數點庫之小數點位置之資訊，利用前述編譯程式用戶化部將此資訊供應至編譯程式之固定小數點庫者。

21. 如申請專利範圍第17項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件之處理器為DSP型時，包含對應於前述編譯程式之DSP之機能者。

22. 如申請專利範圍第17項之用以開發系統晶片之開發環境的產生方法，其中對應於前述編譯程式之DSP之機能係包含複項資料型、X/Y記憶體、循環緩衝器、固定小數點資料型、位元反轉、異種暫存器組中之至少1種者。

23. 一種用以開發系統晶片之開發環境的產生方法，其係

包含：

分析所輸入之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生編譯程式用戶化部、彙編程式用戶化部、及模擬程式產生部，前述配置指定文件之處理器為SIMD型，硬體之指定包含SIMD指令型式之描述，利用前述編譯程式用戶化部構建編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

24. 如申請專利範圍第23項之用以開發系統晶片之開發環境的產生方法，其中前述編譯程式用戶化部係依據有關輸入之SIMD指令之資訊，產生機械指令函數定義者。

25. 如申請專利範圍第23項之用以開發系統晶片之開發環境的產生方法，其中前述編譯程式用戶化部係依據有關輸入之SIMD指令之資訊，產生特殊資料型者。

26. 如申請專利範圍第23項之用以開發系統晶片之開發環境的產生方法，其中前述模擬程式用戶化部係依據有關輸入之SIMD指令之資訊，使用戶執行模擬程式之動作者。

27. 如申請專利範圍第23項之用以開發系統晶片之開發環境的產生方法，其中前述模擬程式用戶化部係依據有關輸入之SIMD指令之資訊，使用戶執行模擬程式之結果輸出機能者。

28. 如申請專利範圍第23項之用以開發系統晶片之開發環

境的產生方法，其中前述SIMD指令之動作描述係將多數指令彙總成1個指令而加以描述者。

29. 如申請專利範圍第23項之用以開發系統晶片之開發環境的產生方法，其中前述SIMD指令之動作描述係利用指標描述者。

30. 一種用以開發系統晶片之開發環境的產生方法，其係包含：

由含有已定義之多數指令之多數指令庫選擇必要之指令，

合併此等選擇之指令而產生適應於應用程式之開發環境，且前述開發環境係包含構建編譯程式之編譯程式用戶化部、構建彙編程式之彙編程式用戶化部、及構建模擬程式之模擬程式產生部者。

31. 如申請專利範圍第30項之用以開發系統晶片之開發環境的產生方法，其中進一步包含由前述多數指令庫選擇必要之指令後，在前述各指令庫產生中間庫，並合併此等中間庫者。

32. 如申請專利範圍第30項之用以開發系統晶片之開發環境的產生方法，其中進一步包含由前述多數指令庫選擇必要之指令後，在前述各指令庫產生中間庫，並合併此等中間庫與基本指令之中間庫者。

33. 如申請專利範圍第30項之用以開發系統晶片之開發環境的產生方法，其中前述指令庫內之指令係可選擇前述指令所使用之資料型者。

34. 如申請專利範圍第30項之用以開發系統晶片之開發環境的產生方法，其中前述指令庫內之指令係可變更指令所使用之資料型者。

35. 一種用以開發系統晶片之開發環境的產生方法，其係包含：

分析所輸入之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生系統晶片之開發環境，

前述開發環境包含構建編譯程式之編譯程式用戶化部、構建彙編程式之彙編程式用戶化部、及構建模擬程式之模擬程式產生部，

前述配置指定文件包含高階之描述者。

36. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述高階之描述係至少包含對位元連結、部分位元分離、函數型式、符號之造型運算符、臨時變數、溢出之動作描述者。

37. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述高階之描述亦可在式之左邊描述位元連結者。

38. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件係包含流水線定義之描述者。

39. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述流水線定義之描述係包含流

水線型之定義者。

40. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述流水線定義之描述係包含危險資訊者。

41. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件在有關流水線方面，包含編譯程式用之調度資訊者。

42. 如申請專利範圍第35項之用以開發系統晶片之開發環境的產生方法，其中前述配置指定文件在有關流水線方面，包含產生危險驗證向量用之描述者。

43. 一種用以開發系統晶片之開發環境的產生方法，其係包含：

分析所輸入之指令，

按照前述分析之指令，由記載於描述系統晶片之配置之配置指定文件之動作指令產生RTL產生部，且前述RTL產生部包含施行硬體資源分配與調度之調度部、整合磁心部及控制總線與記憶體之介面之介面整合部、及將C語言變化為Verilog之變換部者。

44. 如申請專利範圍第43項之用以開發系統晶片之開發環境的產生方法，其中前述介面整合部係由前述動作指令施行硬體資源分配與調度，並整合磁心部及控制總線與記憶體之介面，以產生RTL者。

45. 一種用以開發系統晶片之開發環境的產生方法，其係包含：

分析所輸入之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生系統晶片之開發環境，

前述開發環境包含產生驗證向量之驗證向量產生部，前述驗證向量產生部由前述配置指定文件之動作指令產生處理器之驗證向量者。

46. 如申請專利範圍第45項之用以開發系統晶片之開發環境的產生方法，其中前述驗證向量產生部係由前述配置指定文件之暫存器資訊產生ISA資訊者。

47. 一種電腦可讀取式媒體，其記錄用以產生開發系統晶片之開發環境之程式，該程式係包含：

分析輸入至電腦之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生編譯程式用戶化部、彙編程式用戶化部、及模擬程式產生部，配置指定文件包含執行指令之硬體之指定，利用前述編譯程式用戶化部構建用以開發系統晶片之編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

48. 如申請專利範圍第47項之電腦可讀取式媒體，其中前述配置指定文件係包含定義暫存器之構成之暫存器定義者。

49. 如申請專利範圍第47項之電腦可讀取式媒體，其中前述暫存器定義係包含定義前述編譯程式之暫存器之使

用方法之資訊者。

50. 如申請專利範圍第48項之電腦可讀取式媒體，其中前述暫存器定義係包含定義前述編譯程式之暫存器之使用方法之資訊，此資訊被省略時，利用前述編譯程式用戶化部設定編譯程式之缺席規則之暫存器之使用方法者。

51. 如申請專利範圍第50項之電腦可讀取式媒體，其中前述編譯程式用戶化部係依據前述暫存器定義之資訊，定義可直接在各暫存器存取之暫存器虛擬變數者。

52. 如申請專利範圍第48項之電腦可讀取式媒體，其中前述暫存器定義係包含暫存器之別名定義，前述暫存器之別名定義係被供應至前述彙編程式用戶化部者。

53. 如申請專利範圍第47項之電腦可讀取式媒體，其中前述開發環境進一步包含RTL產生部、驗證向量產生部、糾錯程式產生部者。

54. 如申請專利範圍第53項之電腦可讀取式媒體，其中前述暫存器定義包含暫存器名，此暫存器名被供應至前述編譯程式用戶化部、彙編程式用戶化部、模擬程式用戶化部、RTL產生部、驗證向量產生部、及糾錯程式產生部者。

55. 如申請專利範圍第48項之電腦可讀取式媒體，其中前述配置指定文件係包含指令之規格、操作數碼、動作內容者。

56. 如申請專利範圍第48項之電腦可讀取式媒體，其中前

述編譯程式用戶化部係依據前述指令定義部之資訊產生機械指令函數定義者。

57. 如申請專利範圍第48項之電腦可讀取式媒體，其中前述配置指定文件之處理器係VLIW型，硬體之指定係包含VLIW之指令型式之指定、及VLIW之槽之指定中一方者。

58. 如申請專利範圍第57項之電腦可讀取式媒體，其中前述配置指定文件係包含槽定義，其係在將指令並聯化之際，用以區別置於何槽之指令者。

59. 如申請專利範圍第57項之電腦可讀取式媒體，其中前述配置指定文件係包含槽定義，其係使將指令並聯化之際所使用之槽資訊反映於機械指令函數定義者。

60. 如申請專利範圍第57項之電腦可讀取式媒體，其中依據處理器之規格指定指令動作描述內之臨時變數及符號擴充後之值之位元寬，將其資訊供應至模擬程式用戶化部，使其反映於前述模擬程式者。

61. 如申請專利範圍第59項之電腦可讀取式媒體，其中將前述槽資訊供應至彙編程式用戶化部，使其反映於前述彙編程式者。

62. 如申請專利範圍第59項之電腦可讀取式媒體，其中將前述槽資訊供應至前述驗證向量產生部，使其反映於前述驗證向量者。

63. 一種電腦可讀取式媒體，其記錄用以產生開發系統晶片之開發環境之程式，該程式係包含：

分析輸入於電腦之指令，

依據按照前述分析之指令描述系統晶片之配置之配置指定文件之資訊，產生構建編譯程式之編譯程式用戶化部、構建彙編程式之彙編程式用戶化部、及構建模擬程式之模擬程式產生部，前述配置指定文件之處理器為DSP型，硬體之指定包含DSP指令型式之描述，利用前述編譯程式用戶化部構建編譯程式，利用彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

64. 如申請專利範圍第63項之電腦可讀取式媒體，其中前述配置指定文件係包含指定任意位元寬之指令之資訊者。

65. 如申請專利範圍第63項之電腦可讀取式媒體，其中前述配置指定文件係包含定義任意位元寬之暫存器之資訊者。

66. 如申請專利範圍第63項之電腦可讀取式媒體，其中前述配置指定文件係包含指定固定小數點庫之小數點位置之資訊，利用前述編譯程式用戶化部將此資訊供應至編譯程式之固定小數點庫者。

67. 如申請專利範圍第63項之電腦可讀取式媒體，其中前述配置指定文件之處理器為DSP型時，包含對應於前述編譯程式之DSP之機能者。

68. 如申請專利範圍第63項之電腦可讀取式媒體，其中對應於前述編譯程式之DSP之機能係包含複項資料型、

X/Y記憶體、循環緩衝器、固定小數點資料型、位元反轉、異種暫存器組中之至少1種者。

69. 一種電腦可讀取式媒體，其記錄用以產生開發系統晶片之開發環境之程式，該程式係包含：

分析輸入至電腦之指令，

按照前述分析之指令，依據配置指定文件之資訊，產生編譯程式用戶化部、彙編程式用戶化部、及模擬程式產生部，前述配置指定文件之前述處理器為SIMD型，硬體之指定包含SIMD指令型式之描述，利用前述編譯程式用戶化部構建編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

70. 如申請專利範圍第69項之電腦可讀取式媒體，其中前述編譯程式用戶化部係依據有關輸入之SIMD指令之資訊，產生機械指令函數定義者。

71. 如申請專利範圍第69項之電腦可讀取式媒體，其中前述編譯程式用戶化部係依據有關輸入之SIMD指令之資訊，產生特殊資料型者。

72. 如申請專利範圍第69項之電腦可讀取式媒體，其中前述模擬程式用戶化部係依據有關輸入之SIMD指令之資訊，使用戶執行模擬程式之動作者。

73. 如申請專利範圍第69項之電腦可讀取式媒體，其中前述模擬程式用戶化部係依據有關輸入之SIMD指令之資訊，使用戶執行模擬程式之結果輸出機能者。

74. 如申請專利範圍第69項之電腦可讀取式媒體，其中前述SIMD指令之動作描述係將多數指令彙總成1個指令而加以描述者。

75. 如申請專利範圍第69項之電腦可讀取式媒體，其中前述SIMD指令之動作描述係利用指標描述者。

76. 一種電腦可讀取式媒體，其記錄用以產生開發系統晶片之開發環境之程式，該程式係包含：

使電腦由含有已定義之多數指令之多數指令庫選擇必要之指令，

合併此等選擇之指令而產生適應於應用程式之開發環境，且前述開發環境係包含構建編譯程式之編譯程式用戶化部、構建彙編程式之彙編程式用戶化部、及構建模擬程式之模擬程式產生部者。

77. 如申請專利範圍第76項之電腦可讀取式媒體，其中進一步包含由前述多數指令庫選擇必要之指令後，在前述各指令庫產生中間庫，並合併此等中間庫者。

78. 如申請專利範圍第76項之電腦可讀取式媒體，其中進一步包含由前述多數指令庫選擇必要之指令後，在前述各指令庫產生中間庫，並合併此等中間庫與基本指令之中間庫者。

79. 如申請專利範圍第76項之電腦可讀取式媒體，其中前述指令庫內之指令係可選擇前述指令所使用之資料型者。

80. 如申請專利範圍第77項之電腦可讀取式媒體，其中前

述指令庫內之指令係可變更指令所使用之資料型者。

81. 一種電腦可讀取式媒體，其記錄用以產生開發系統晶片之開發環境之程式，該程式係包含：

分析輸入至電腦之指令，

按照前述分析之指令，依據配置指定文件之資訊，產生構建編譯程式之編譯程式用戶化部、構建彙編程式之彙編程式用戶化部、及構建模擬程式之模擬程式產生部，前述配置指定文件包含描述系統晶片之配置之高階之描述，利用前述編譯程式用戶化部構建編譯程式，利用前述彙編程式用戶化部構建彙編程式，利用前述模擬程式產生部構建模擬程式者。

82. 如申請專利範圍第81項之電腦可讀取式媒體，其中前述高階之描述係至少包含對位元連結、部分位元分離、函數型式、符號之造型運算符、臨時變數、溢出之動作描述者。

83. 如申請專利範圍第81項之電腦可讀取式媒體，其中前述高階之描述亦可在式之左邊描述位元連結者。

84. 如申請專利範圍第81項之電腦可讀取式媒體，其中前述配置指定文件係包含流水線定義之描述者。

85. 如申請專利範圍第81項之電腦可讀取式媒體，其中前述流水線定義之描述係包含流水線型之定義者。

86. 如申請專利範圍第81項之電腦可讀取式媒體，其中前述流水線定義之描述係包含危險資訊者。

87. 如申請專利範圍第81項之電腦可讀取式媒體，其中前

述配置指定文件在有關流水線方面，包含編譯程式用之調度資訊者。

88. 如申請專利範圍第81項之電腦可讀取式媒體，其中前述配置指定文件在有關流水線方面，包含產生危險驗證向量用之描述者。

89. 一種電腦可讀取式媒體，其記錄用以產生開發系統晶片之開發環境之程式，該程式係包含：

分析輸入至電腦之指令，

按照前述分析之指令，由記載於描述系統晶片之配置之配置指定文件之動作指令產生包含調度部、介面整合部、及變換部之RTL產生部，利用前述調度部施行硬體資源之分配與調度，利用介面整合部整合磁心部及控制總線與記憶體之介面，利用前述變換部將C語言變化為Verilog者。

90. 如申請專利範圍第89項之電腦可讀取式媒體，其中前述介面整合部係由前述動作指令施行硬體資源分配與調度，並整合磁心部及控制總線與記憶體之介面，以產生RTL者。

拾壹、圖式

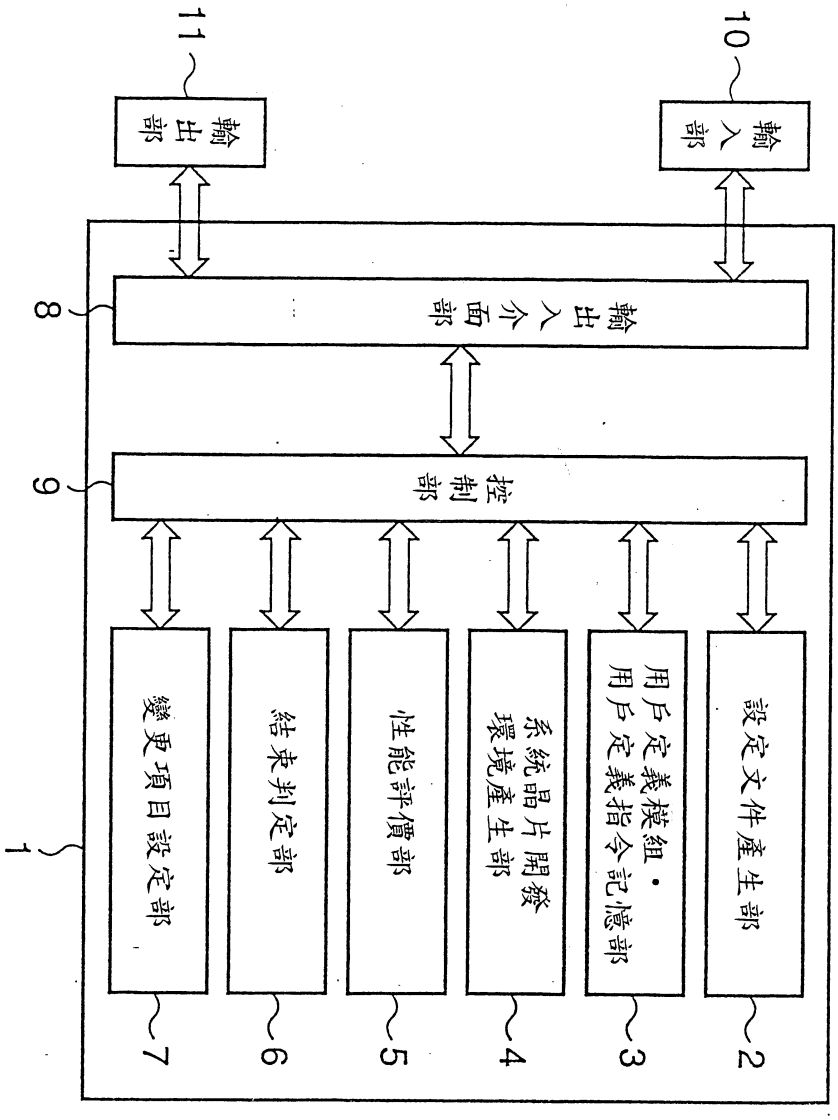

輸入部
10
輸出部
11
輸出入介面部
8
控制部
9
設定文件產生部
2
用户定義模組・用户定義指令記憶部
3
系統晶片開發環境產生部
4
性能評價部
5
結束判定部
6
變更項目設定部
7
1

圖 1

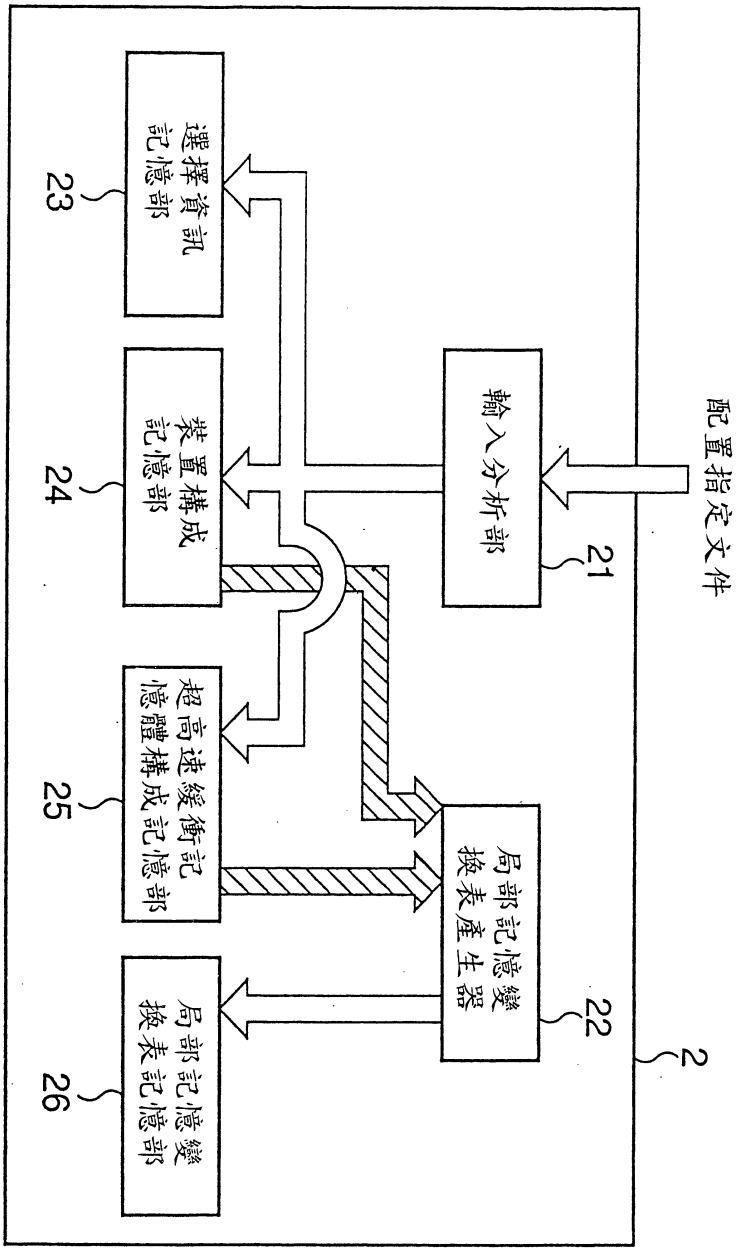

配置指定文件
輸入分析部
21
局部記憶變換表產生器
22
2
選擇資訊記憶部
23
裝置構成記憶部
24
超高速緩衝記憶體構成記憶部
25
局部記憶變換表記憶部
26

圖 2

```
//configuration data
CHIP_NAME="SimpleChip1"; //晶片名
FREQUENCY=200;           //頻率、單位、MHz。

//處理器1
P_MODULE[Processor1]{
        P_ENGINE{
            //處理器磁心之規格
            CORE{
                ID=0;  // 磁心ID
                // 選擇指令
                DIV=ON;    //除法指令
                MINMAX=ON;  //最大/最小值指令
                BIT=ON;     //位元操作指令
            };
            IMEM{
                SIZE=4;     //KB
            };
            DMEM{
                SIZE=2;     //KB
            BIU{  //總線介面單元
                BUS_SIZE=64;
            };
            INTC{  //中斷控制器
                CHANNEL_BITW=16; //未確定
                LEVEL=15;  //未確定
            };
            DSU{  //糾錯單元
                INST_ADDR_BREAK_CHANNEL=1;
                DATA_ADDR_BREAK_CHANNEL=1;
            };
        };
};
//處理器2
P_MODULE[Processor2]{
        P_ENGINE{
            CORE{
                ID=1;  // 磁心ID
                // 選擇指令
                DIV=ON;    //除法指令
                MINMAX=OFF;  //最大/最小值指令
                BIT=OFF;     //位元操作指令
            };
            IMEM{  // 指令RAM
                SIZE=8;     //KB
            };
            DMEM{ // 資料RAM
                SIZE=20;     //KB
            };
            ICACHA{  // 指令超高速緩衝記憶體
                SIZE=4;     //KB
            };
            DCACHE{  // 資料超高速緩衝記憶體
                SIZE=8;     //KB
            };
        };
        //共處理器
        COPRO[Cop1]{
            IS_VLIW=YES ;  //VUW型共處理器
            VLIW_BTW=32 ; //指令長32位元
            ISA_DEFINE="cop1.isa" ;  //ISA定義、別的文件
            RTL     ="cop1.v"          //RTL描述、別的文件
        };
//用戶定義模組
UCI{
            ISA_DEFINE="uci1.uci" ; //ISA定義、別的文件
            RTL="uci1.v" ;  //RTL描述、別的文件
            SIM="ucimode1.c" ;       //硬體c模型、別的文件
};
//整體變換表
GLOBAL_MAP="schip1.map";  //指定別的文件

//end of file
```

圖 3

圖式簡單

圖 4A

```
//start : size : name : cache(opt) : shadow_original_start(opt) : scope : type : read_write(opt)
0X0000_0000 : 0X0020_0000 : RAM1 :          : : global:memory : ro ;
0X0080_0000 : 0X0080_0000 : RAM2 : Cache    : : global:memory : rw ;
0X8080_0000 : 0X0080_0000 : RAM3 :          : : global:memory : rw ;
```

圖 4B

```
:RAM1      :0X00000000 : 0X00200000 :          :global : ro::       memory:
:lmem0     :0X00200000 : 0X00002000 :          : local : rw::         lmem:   In  mm
:lmem1     :0X00202000 : 0X00002000 :          : local : rw::         lmem:   In  mm
:dmem0     :0X00208000 : 0X00004000 :          : local : rw::         dmem:   In  mm
:dmem1     :0X0020c000 : 0X00004000 :          : local : rw::         dmem:   In  mm
:lcahe_dat :0X00300000 : 0X00004000 :          : local : rw:: icache_data:   In  mm
:lcahe_tag :0X00310000 : 0X00004000 :          : local : rw::  icache-tag:   In  mm
:RAM2      :0X00800000 : 0X00800000 : cache  :global : rw::       memory:
:RAM3      :0X00808000 : 0X00800000 :          :global : rw::       memory:
```

圖 4C

```
說明#指令之助記憶符號、操作碼與自變數
void  xor(int_reg_modify,int_reg_src) ;
{
        code16="0000_0010_0000_0000" ;
}
short  xon(int_reg_src,signed  char_Imm) ;
{
        code16="0000_0011_iiii_iiii" ;
}
```


571237

圖式繪頁

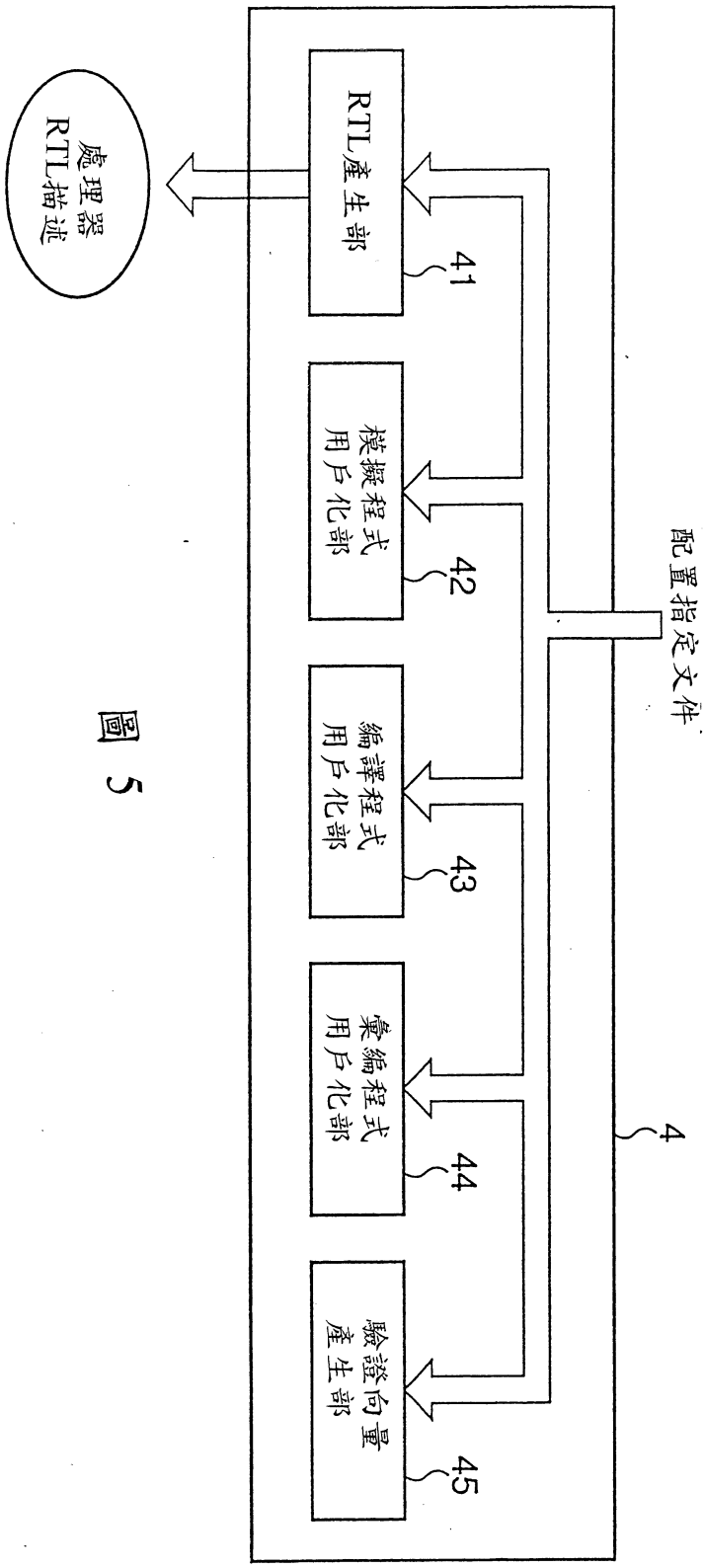
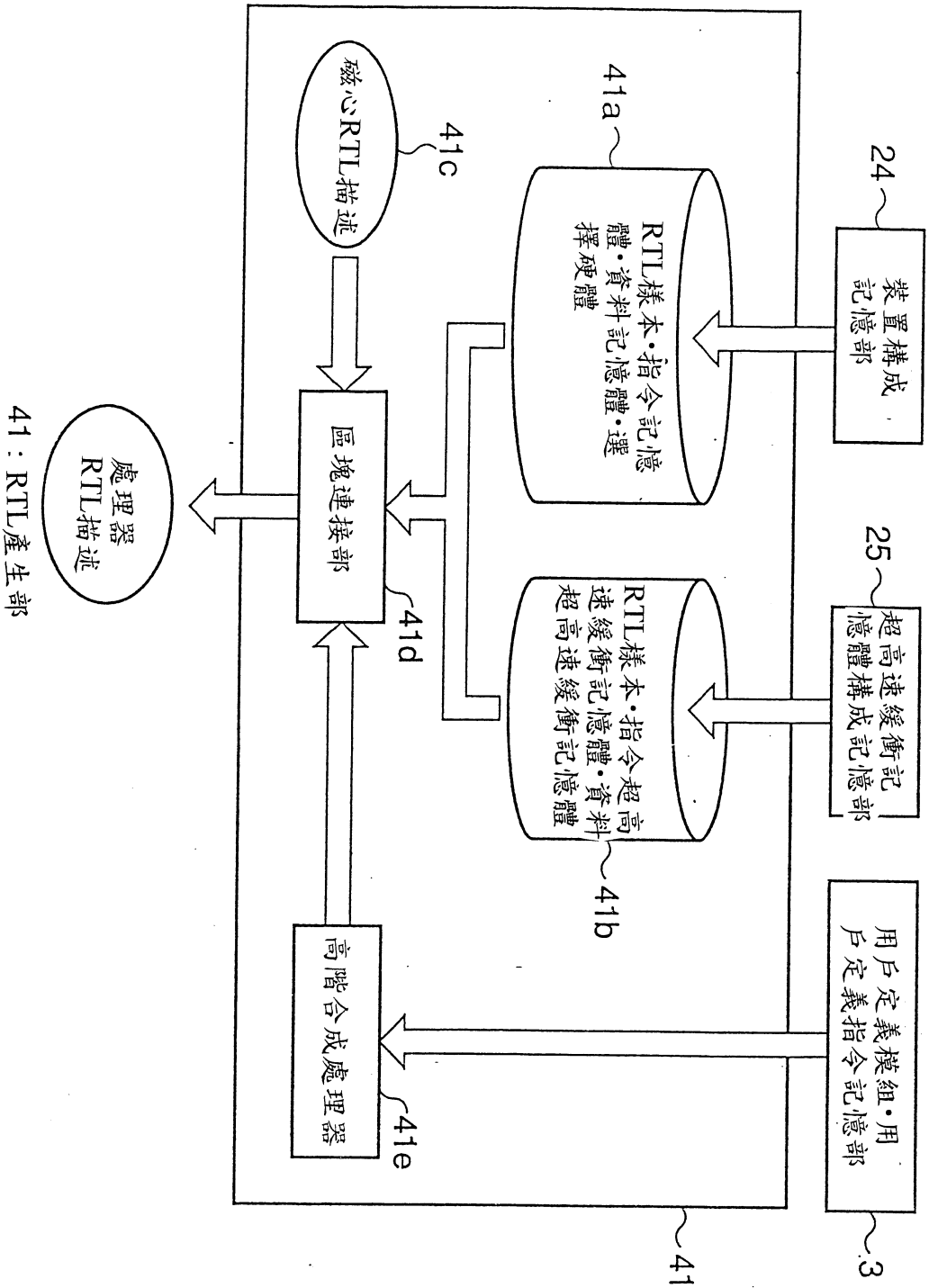

處理器
RTL描述
RTL產生部
41
模擬程式
用户化部
42
編譯程式
用户化部
43
彙編程式
用户化部
44
驗證向量
產生部
45
配置指定文件
4

圖 5


24
裝置構成記憶部
25
超高速緩衝記憶體構成記憶部
用戶定義模組・用戶定義指令記憶部
3
41a
RTL樣本・指令記憶體・資料記憶體・選擇硬體
41b
RTL樣本・指令超高速緩衝記憶體・資料超高速緩衝記憶體
41
41c
磁心RTL描述
區塊連接部
41d
高階合成處理器
41e
處理器RTL描述

41：RTL產生部

圖 6

圖式續頁

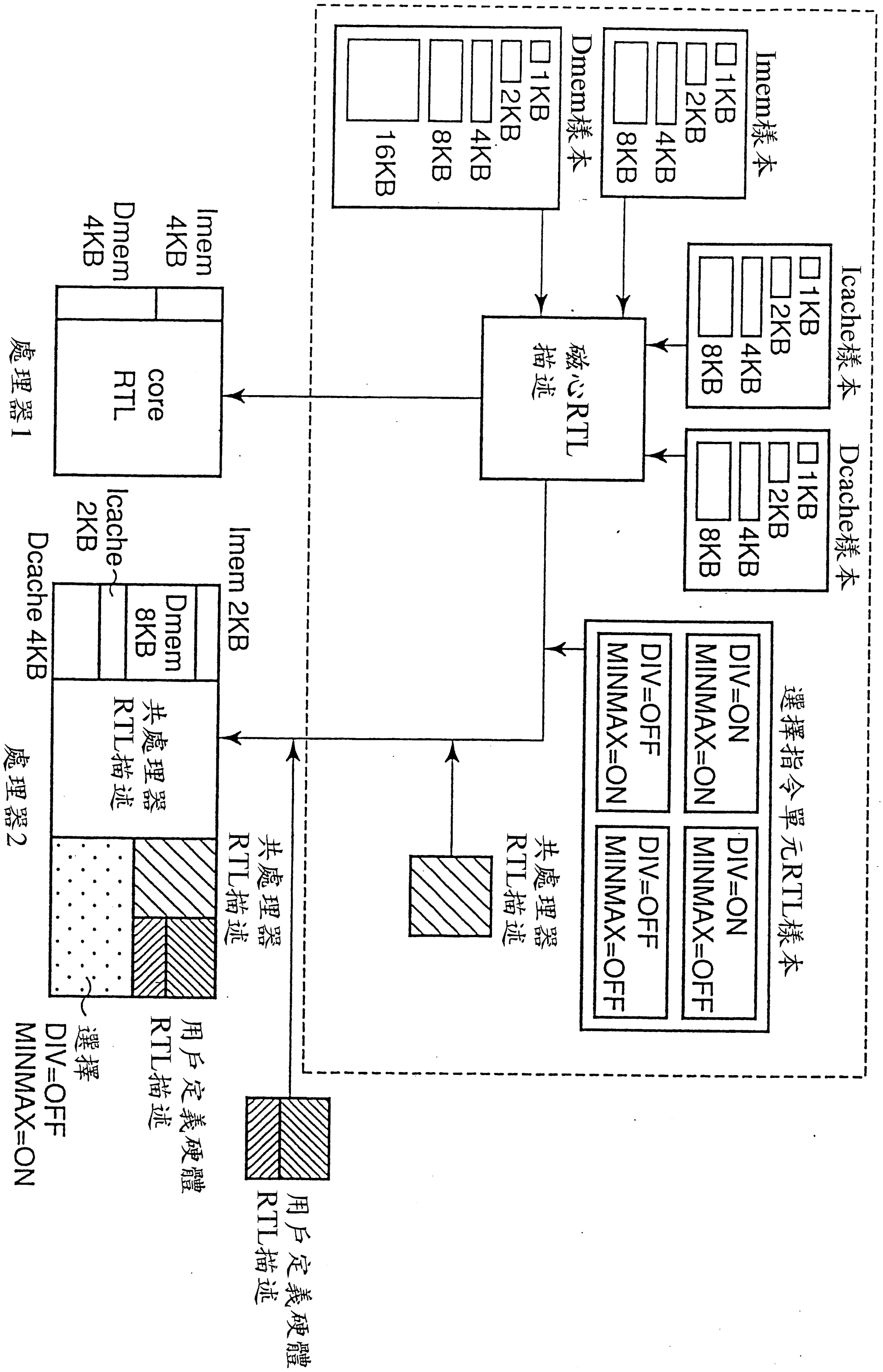

RTL樣本
Imem樣本
1KB
2KB
4KB
8KB
Dmem樣本
1KB
2KB
4KB
8KB
16KB
Icache樣本
1KB
2KB
4KB
8KB
Dcache樣本
1KB
2KB
4KB
8KB
選擇指令單元RTL樣本
DIV=ON
MINMAX=ON
DIV=ON
MINMAX=OFF
DIV=OFF
MINMAX=ON
DIV=OFF
MINMAX=OFF
磁心RTL描述
共處理器RTL描述
Imem 4KB
Dmem 4KB
core RTL
處理器1
Imem 2KB
Dmem 8KB
Icache 2KB
Dcache 4KB
共處理器RTL描述
處理器2
共處理器RTL描述
用戶定義硬體RTL描述
選擇
DIV=OFF
MINMAX=ON
用戶定義硬體RTL描述

圖 7.

圖式

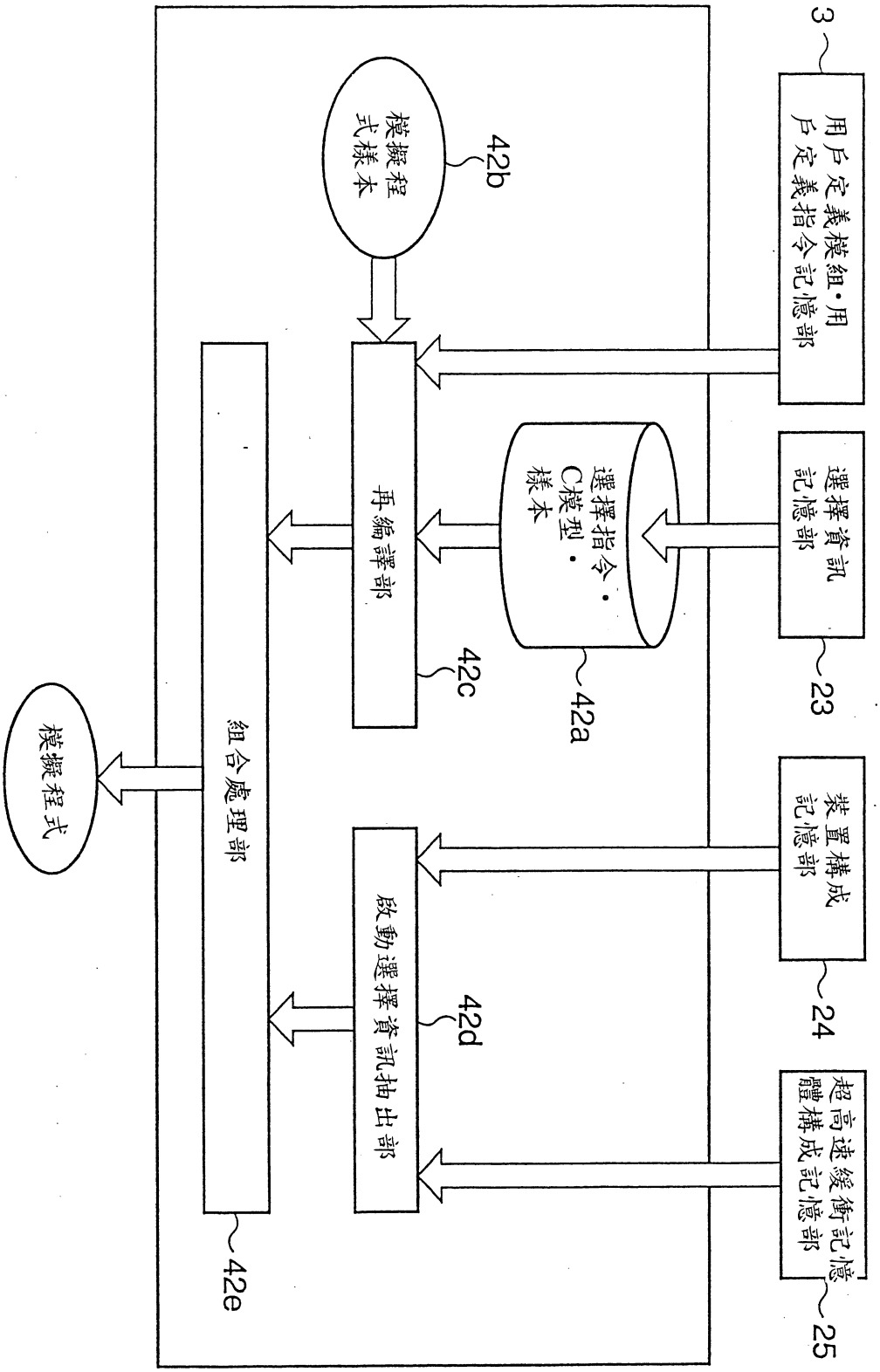

3
用户定義模組·用户定義指令記憶部
42b
模擬程式樣本
選擇資訊記憶部
23
選擇指令·C模型·樣本
42a
再編譯部
42c
裝置構成記憶部
24
放動選擇資訊抽出部
42d
超高速緩衝記憶體構成記憶部
25
組合處理部
42e
模擬程式

圖 8

圖 9A

```
-opt_minmax=ON
-opt_div=ON
-opt_bit=OFF
-rom=0x00000000,0x00200000
-imem=0x0020000,0x00300800
-dmem=0x00300000,0x00400000
-icache=0x00400000,0x007f0000
-dcache=0x007f0000,0x00800000
-ram=0x00800000,0x01000000
-ram-shadow=0x80800000,0x81000000
```

圖 9B

```
-rom=0x00000000,0x00200000
-imem=0x0020000,0x00300800
-dmem=0x00300000,0x00400000
-icache=0x00400000,0x007f0000
-dcache=0x007f0000,0x00800000
-ram=0x00800000,0x01000000
-ram-shadow=0x80800000,0x81000000
```

圖 9C

```
//選擇指令
void_asm  min(int_reg_modify,int_reg_src) ;
void_asm  max(int_reg_modify,int_reg_src) ;
//用戶定義指令
void_asm  xor(int_reg_modify,int_reg_src) ;
short_asm  xori(int_reg_src,signed  char_imm) ;
```

圖式簡單說明

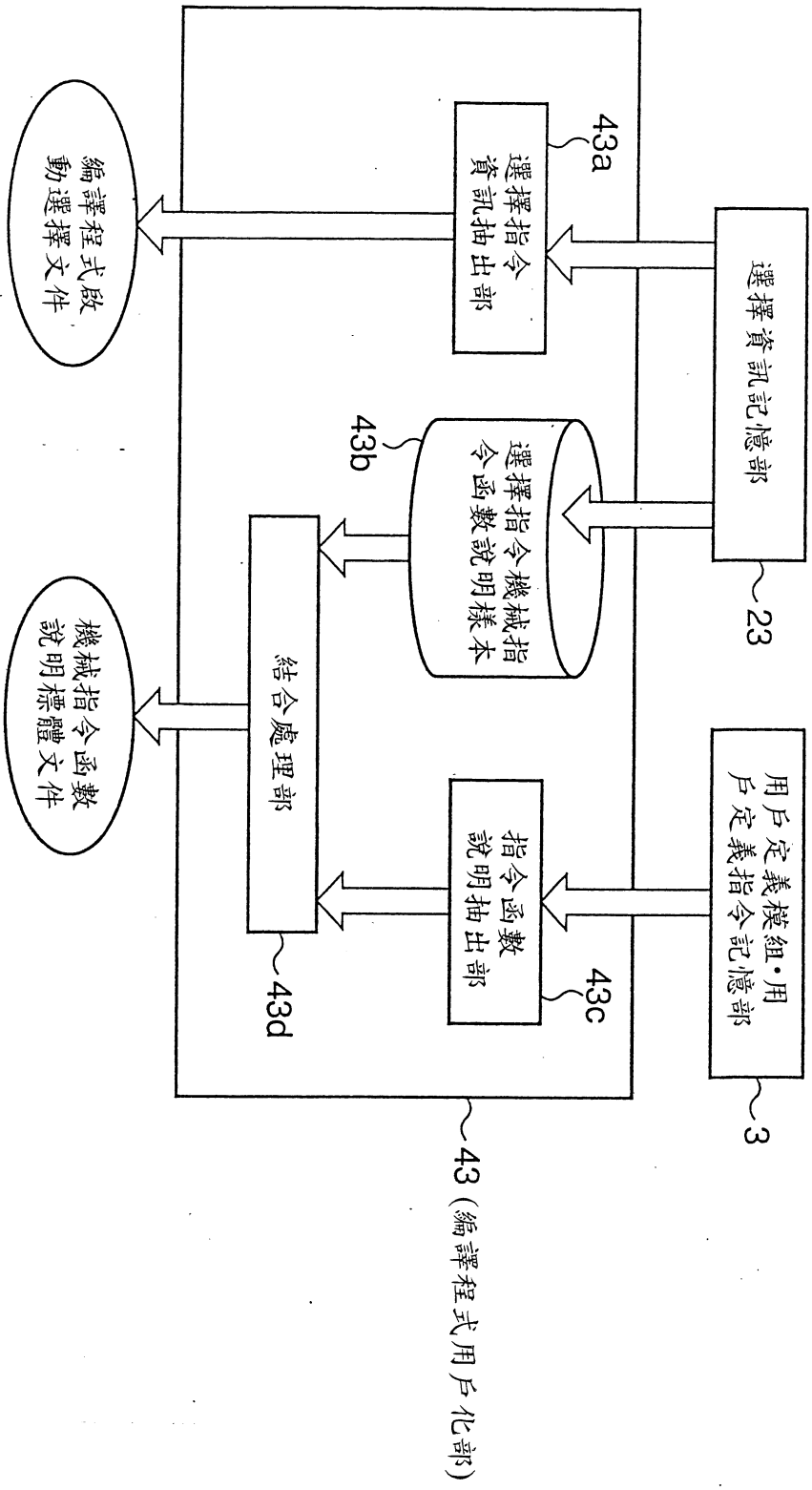

選擇資訊記憶部
23
43a
選擇指令資訊抽出部
編譯程式啟動選擇文件
43b
選擇指令機械指令函數說明樣本
結合處理部
機械指令函數說明標體文件
43d
用戶定義模組・用戶定義指令記憶部
3
指令函數說明抽出部
43c
43（編譯程式用戶化部）

圖 10

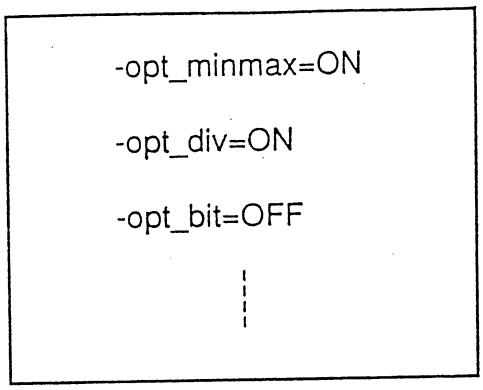


圖 11A

```
XOR       Rn  Rm      {(Rn,32,U,1,1),(Rm,32,U,1,0)}
XOR1  Rn, Rm, Imm8    {(Rn,32,U,1,1),(Rm,32,U,1,0),(Imm8,U,1,0)}

#XOR  Rn、Rm之部分在彙編程式之格式、
#接著連接操作數之定義

#(Rn, 32, U, 1, 1)中，Rn表示操作數、32位元數
#U表示符號之有無(U表示unsigned，S表示signed)，其次之1表示
#Rn為源操作數，最後之1表示
#Rn為目標操作數。
```

圖 11B

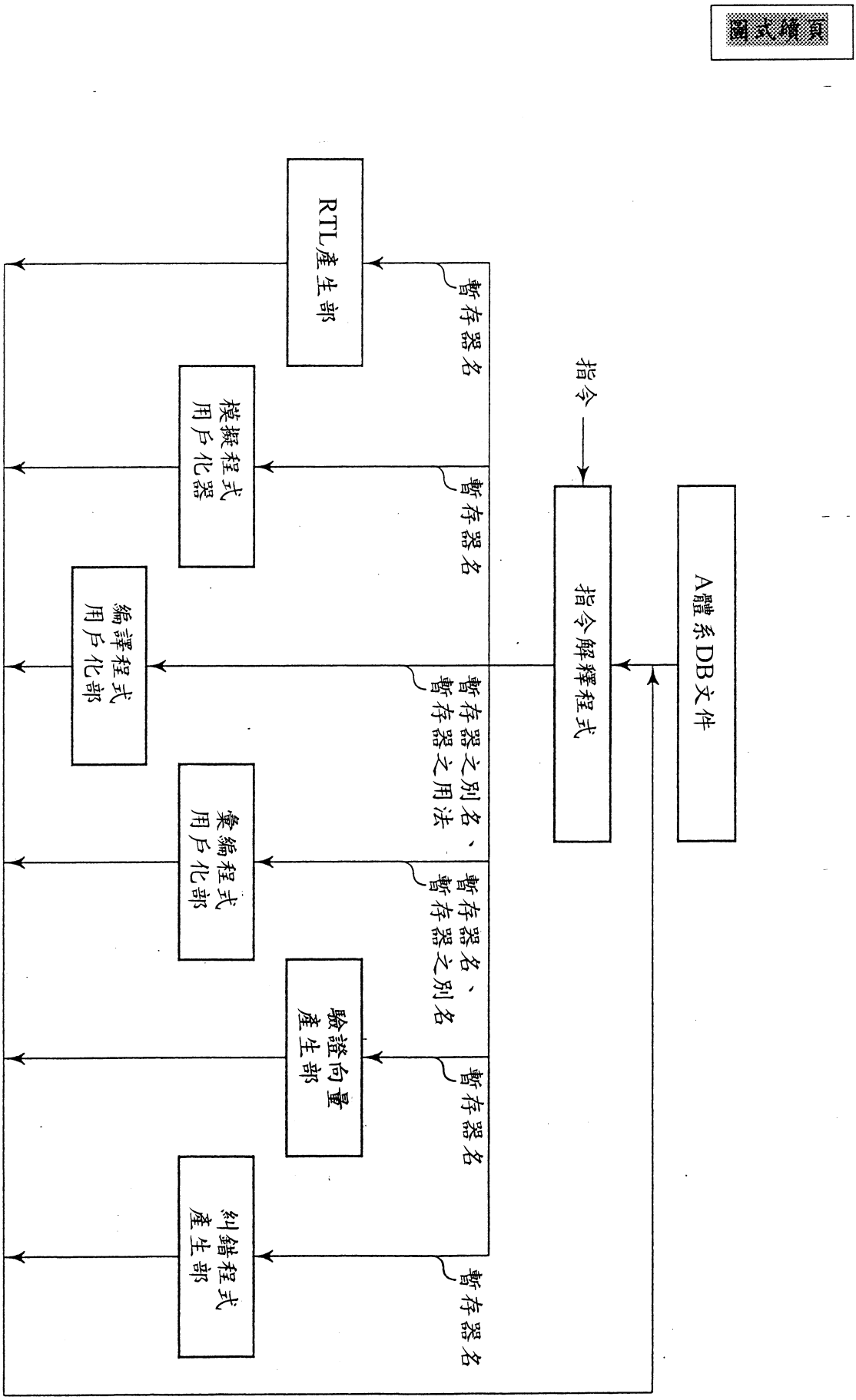

A體系DB文件
指令
指令解釋程式
RTL產生部
暫存器名
模擬程式用户化器
暫存器名
暫存器之別名、暫存器之用法
暫存器名
編譯程式用户化部
象編程式用户化部
暫存器名、暫存器之別名
驗證向量產生部
暫存器名
糾錯程式產生部
暫存器名

圖 12

```
//處理器之指定

Pro_NAME: Audio VLIW Processor;
Pro_TYPE: 2_WAY_VLIW;

//暫存器之定義
<pro_register>
CR:     8: 32:
CCR: 0:  csar: "Audio Copro Shift-Amount Register": {
         csar[5:0]:rw:X:"Shift Amount":
};
//指令定義
<pro_ISA>
CAND: CRq, Crp:
          1111_0000_0111 0000_qqqq_pppp_0101:
          [ CRq=CRq and CRp ; ] :
          Cop: :
          C3:
          ;

//動作描述

(HI!!L0)=Rn*Rm;
[msb:lsb]
```

圖 13

```
<register>
//The number of global purpose registers
GPR:16;
CR: 32: 64:
//Registers usage of compiler
RET: 0;        // The register for return value is 0
ARG: 1,2,3,4;// The register for parameters are 1,2,3,4
ZERO: 12;    // The register of which value is always 0 is 12
...
SP: 15;        // The register for stack pointer is 15
//Alias
alias sp R15;// sp is same as r15
//Definitions of control/special registers
CTR: 1: LP: "Link Pointer" : {
      LP[31 : 1] : rw: x"Return Address" ;
      LTOM[0] : rw:X:"LP Toggle Operation Mode" ;
} ;
...
```

圖 14

```
// 32bit*8
ACC: 2: 256;
```

圖 15

```
// Shift Amount Register
CSAR: 2;
```

圖 16

```
COP_NAME: Audio VLIW Coprocessor;
COP_TYPE: 2_WAY_VLIW;

//暫存器之定義
<cop_register>
CR:     8: 32:
CCR: 0:  csar: "Audio Copro Shift-Amount Register": {
         csar[5:0]:rw:X:"Shift Amount";
};
//指令定義
<cop_ISA>
CAND: CRq, Crp:
            1111_0000_0111 0000_qqqq_pppp_0101:
            [ CRq=CRq and CRp ; ]:
            Cop: :
            V3:
            chadd CRl,CRm,CRn:
            CRl,h=CRm. h+CRn. hi :
            Cop: :
            V1:
            ;
```

圖 17

```
DSP_NAME: TEST_DSP_MODULE

<dsp_register>
REG_WIDTH:20;
.
.
.
<dsp_ISA>
dmul: drq, drp:
        1111_0000_qqqq_pppp:
        { dhi = [39. . 20]drg * drp; dlo = [19. . 0]drg * drp; }:
        dsp: :
.
.
.
```

圖 18

```
//SIMD指令

CPADD. H: CRl, CRm, CRn:
        00111_11111_mmmmm_nnnnn:
        {
          CRl[15:0] = CRm[15:0] + CRn[15:0] ;
          CRl[31:16] = CRm[31:16] + CRn[31:16] ;
          CRl[47:32] = CRm[47:32] + CRn[47:32] ;
          CRl[63:48] = CRm[63:48] + CRn[63:48] ;
        } :
        Cop: "PACK=H,H,H"
        POs:
        ;
```

圖 19

```
//SIMD指令
CPADD2. H:  CRl, CRm, CRn:
        00111_11111_mmmmm_nnnnn:
        {
            CRl.h=CRm. h+CRn. h;
        }:
        Cop: "PACK=H,H,H"
        POs:
        ;
```

圖 20

```
//SIMD指令
CPADD3. H:  CRl, CRm, CRn:
        00111_11111_mmmmm_nnnnn:
        {
            CRl.hu=CRm. b+CRn. bu;
        }:
        Cop: "PACK=HU,B,BU"
        POs:
        ;
```

圖 21

```
//SIMD指令
CPADD3. H:  CRl, CRm, CRn:
        00111_11111_mmmmm_nnnnn:
        {
            CRl [i]=CRm[3-i]+CR[3-i];
        }:
        Cop: "SIMD=B,PACK=H,H,H"
        POs:
        ;
```

圖 22

```
//SIMD指令庫

CPADD.xx //2-operand 32bit Addition (A<-A+B)
CPADD3.xx //3-operand 32bit Addition (A<-B+C)
CPADD1.xx //2-operand 32bit Addition Immediate (A<-A+SignExt (Immediate))
CPADD13.xx //3-operand 32bit Addition Immediate (A<-A+SignExt (Immediate))
.
.
.
CPSLL3.xx //3-operand Shift Left Logical Immediate (A<-B≪Immediate(Unsigned))
```

Xx	助記憶符號
Unsigned Byte	UB
Signed Byte	B
Unsigned Halfword	UH
Signed Halfword	H
Unsigned Word	UW
Signed Word	W

圖 23

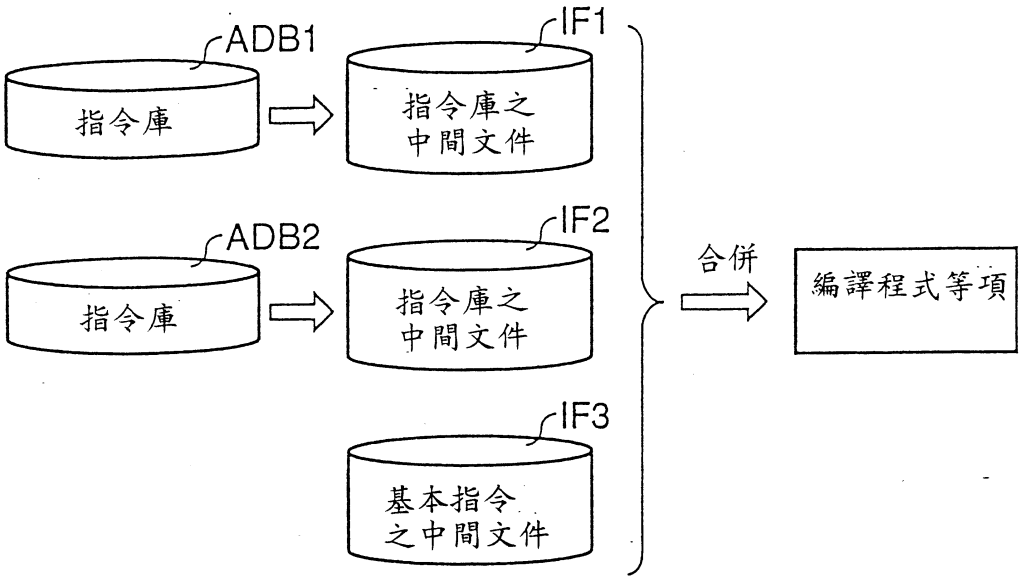


圖 24

```
LW:          Rn.disp8. align4(SP) :
             0100_nnnn_dddd_ddII :
             {Rn = MemWord(ZeroExtension(disp8[7. . 2]||0)+SP) ; }:
             Load:
             ;


SWAP:        (Rn), (Rm):
             1111_nnnn_mmmm_0010_0000_0000_0000_0001
             {tmp1 = MemWord(Rn[31. . 2]||00);
              tmp2 = MemWord(Rm[31. . 2]||00);
              MemWord(Rn[31. . 2]||00) = tmp2;
              MemWord(Rm[31. . 2]||00) = tmp1; }:
             Uci:
             ;
```

圖 25

```
PTYPE : pt1: cr, w, t, opd1: cr, r, opd2;
PTYPE : pt2: cr, r, t, opd1;
```

圖 26

```
HAZARD:cr:Write(pt1), Read(pt2)=2;
```

圖 27

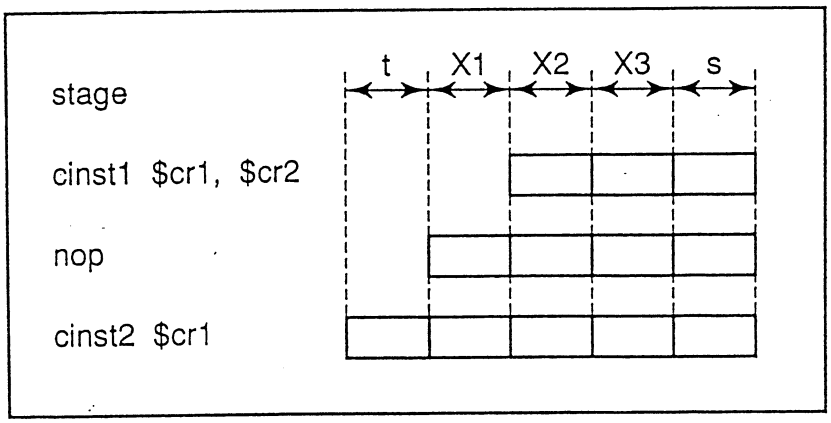


圖 28

```
InterlockPTYPE1:PTYPE2 3:PTYPE3 4;
```

圖 29

```
//core  pipe

      ...   ...

//copro  pipe
pipeline:cp0nop ={FCop () check () available (), Dcop (), T (), X0 (), S ()
pipeline:cp0move4={FCop (), Dcop (), T () check (CPR) available (GPR), S () } ;
pipeline:cp0alu0={FCop (), Dcop (), T () check (CPR), X0 () available (CPR), S () } ;
pipeline:cp0alu3={FCop (), Dcop (), T () check (CSAR,ACC),
                  X0 () available (CBCR), S () } ;
pipeline:cp0mac3={FCop (), Dcop () check (CPR), T () check (ACC), X0 (),
                   X1 () available (ACC), S () } ;

Stall : T->D0;
Stall : D0->FCop;
```

圖 30

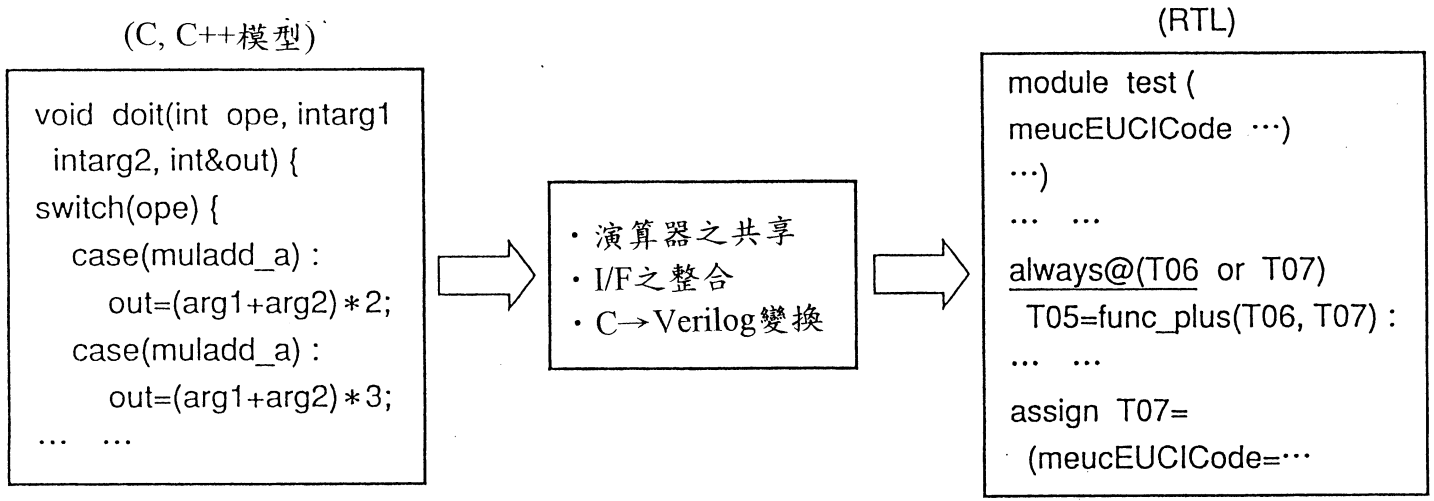

(C, C++模型)
void doit(int ope, intarg1
intarg2, int&out) {
switch(ope) {
case(muladd_a) :
out=(arg1+arg2)*2;
case(muladd_a) :
out=(arg1+arg2)*3;
… …
・演算器之共享
・I/F之整合
・C→Verilog變換
(RTL)
module test (
meucEUCICode …)
…)
… …
always@(T06 or T07)
T05=func_plus(T06, T07) :
… …
assign T07=
(meucEUCICode=…

圖 31

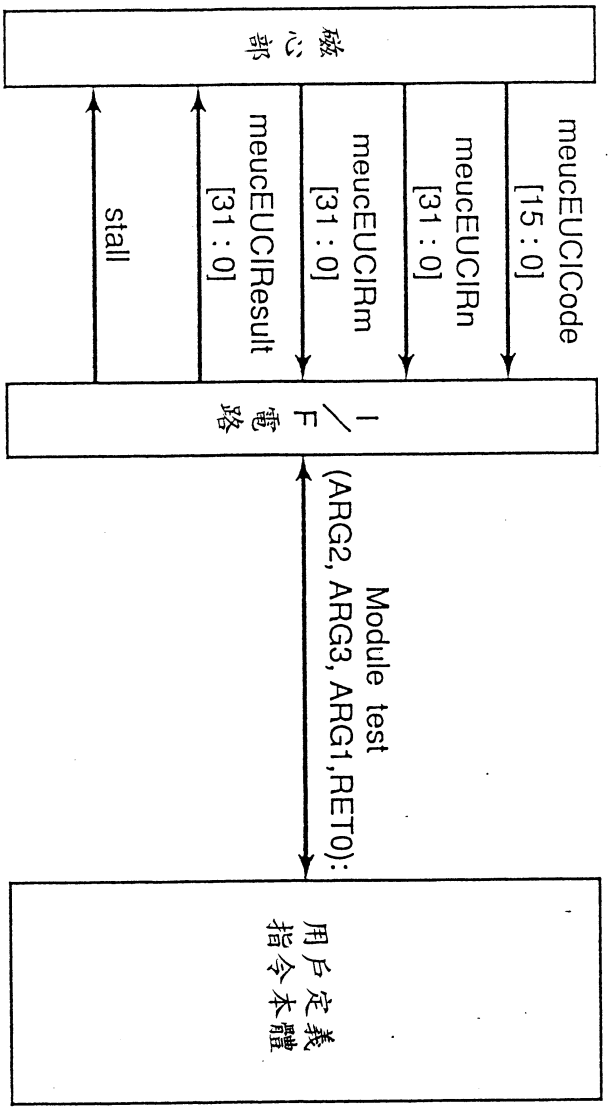

磁心部
meucEUCICode [15 : 0]
meucEUCIRn [31 : 0]
meucEUCIRm [31 : 0]
meucEUCIResult [31 : 0]
stall
I／F電路
Module test (ARG2, ARG3, ARG1,RET0):
用户定義指令本體

圖 32

```
//指令定義

CPXOR3: CRo, CRq, CRp:   //CRo  <-cp(CRq, CRp)
          01011_qqqqq_ppppp_ooooo:
          {
             CRo = CRp  xor  CRq;
          } :
          Cop: :
          POs:
          ;
```

圖 33

```
//ISA資訊

CPXOR3          //助記憶符號
CRo, CRq, CRp   //操作數排列
POS             //流水線型
{
  (CRo_0, CPR, 32, DEST),    //   操作數資訊
  (CRq_0, CPR, 32, SRC),     //   操作數資訊
  (CRp_0, CPR, 32, SRC),     //   操作數資訊
}
```

圖 34

圖式續頁

```
//指令CPXOR3之驗證程式

# mvac0:
        __PutMark
        movh    $1, %hi (0x0)              ¥
        add3    $1, $1, %lo (0x0)           |
        cmovh   $c22, $1                    | 以0x0將輸出暫存器
        movh    $1, %hi (0x0)               | ($c22:64bit)初期化
        add3    $1, $1, %lo (0x0)           |
        cmov    $c22, $1                   /
        movh    $1, %hi (-487809850)       ¥
        add3    $1, $1, %lo (-487809850)    |
        cmovh   $c0, $1                     | 將檢查用資料設定於
        movh    $1, %hi (-942338075)        | 輸入暫存器($c0:64bit)
        add3    $1, $1, %lo (-942338075)    |
        cmov    $c0, $1                    /
        movh    $1, %hi (-920359754)       ¥
        add3    $1, $1, %lo (-920359754)    |
        cmovh   $c15, $1                    | 將檢查資料設定於
        movh    $1, %hi (472324452)         | 暫存器($c15:64bit)
        add3    $1, $1, %lo (472324452)     |
        cmov    $c15, $1                   /
        mov     $1, 0x0
        +CPXOR3 $c22, $c0, $c15            ＞執行檢查對象之指令
        cmovh   $1, $c22                   ¥
        stcb    $1, LOGDISP                 | 將指令之運算結果
        cmov    $1, $c22                    | 輸出至記錄文件
        stcb    $1, LOGDISP                /
        . . .
```

圖 35

圖式頁

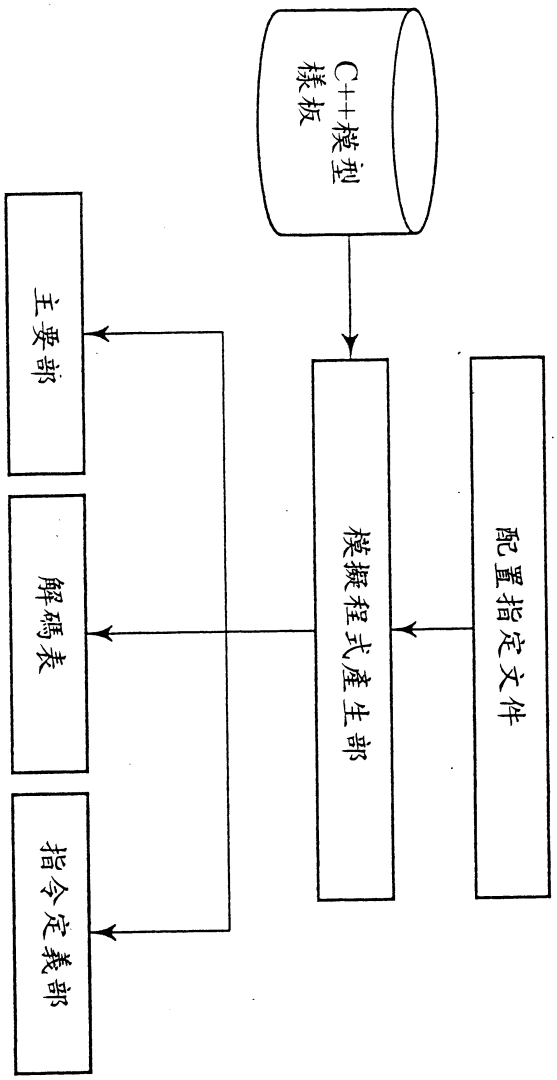


圖 36

圖式修正

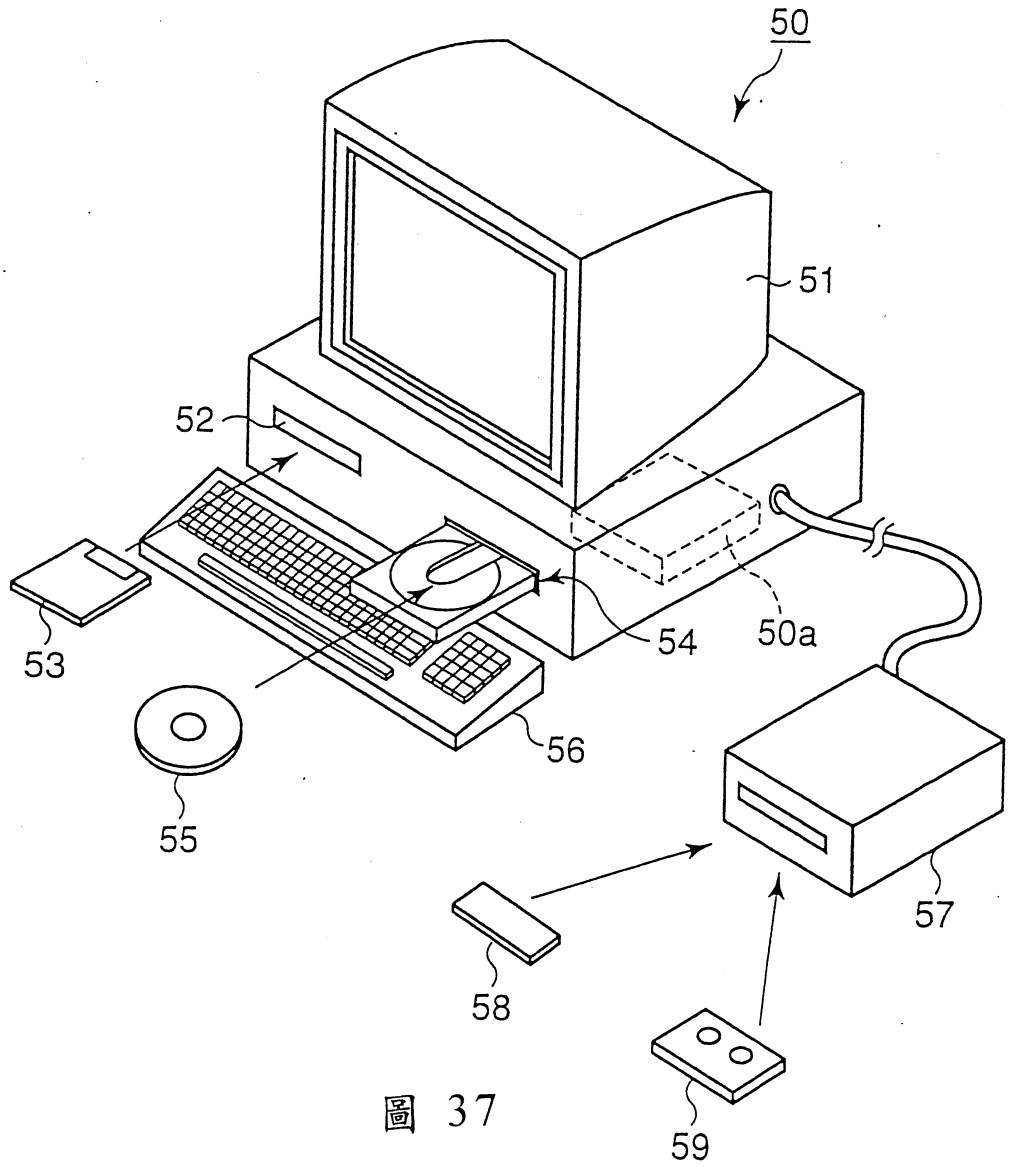


圖 37