

(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2022年12月1日(01.12.2022)



(10) 国際公開番号

WO 2022/249236 A1

(51) 国際特許分類:
G06F 8/41 (2018.01)

(21) 国際出願番号: PCT/JP2021/019604

(22) 国際出願日: 2021年5月24日(24.05.2021)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人:三菱電機株式会社(MITSUBISHI ELECTRIC CORPORATION) [JP/JP]; 〒1008310 東京都千代田区丸の内二丁目7番3号 Tokyo (JP).

(72) 発明者:矢吹 潤(YABUKI, Jun); 〒1008310 東京都千代田区丸の内二丁目7番3号 三菱電機株式会社内 Tokyo (JP). 東山 知彦(HIGASHIYAMA,

Tomohiko); 〒1008310 東京都千代田区丸の内二丁目7番3号 三菱電機株式会社内 Tokyo (JP). 外山 正勝(TOYAMA, Masakatsu); 〒1008310 東京都千代田区丸の内二丁目7番3号 三菱電機株式会社内 Tokyo (JP).

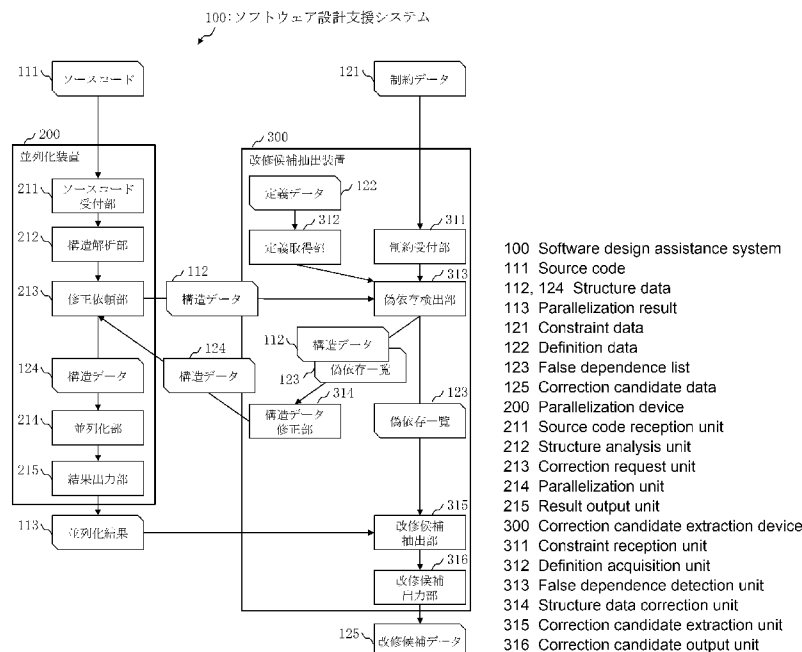
(74) 代理人:弁理士法人クロスボーダー特許事務所(CROSS-BORDER PATENT FIRM); 〒2470056 神奈川県鎌倉市大船二丁目17番10号3階 Kanagawa (JP).

(81) 指定国(表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,

(54) Title: SOFTWARE DESIGN ASSISTANCE SYSTEM, SOFTWARE DESIGN ASSISTANCE METHOD, AND SOFTWARE DESIGN ASSISTANCE PROGRAM

(54) 発明の名称: ソフトウェア設計支援システム、ソフトウェア設計支援方法およびソフトウェア設計支援プログラム

図4



(57) Abstract: According to the present invention, a structure analysis unit (212) analyzes a source code (111) including a plurality of processing blocks to generate structure data (112) on the source code. A false dependence detection unit (313) detects false dependence on the basis of the structure data. A structure data correction unit (314), by correcting the structure data, generates structure data on the source code as corrected structure data (124) when the detected false

HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

- (84) 指定国(表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類：

一 国際調査報告 (条約第21条(3))

dependence is eliminated. A parallelization unit (214) parallelizes the plurality of processing blocks on the basis of the corrected structure data. A result output unit (215) outputs the parallelization result.

(57) 要約：構造解析部（212）は、複数の処理ブロックを含むソースコード（111）を解析して前記ソースコードの構造データ（112）を生成する。偽依存検出部（313）は、前記構造データに基づいて偽依存を検出する。構造データ修正部（314）は、前記構造データを修正することによって、検出された偽依存が解消された場合の前記ソースコードの構造データを修正済みの構造データ（124）として生成する。並列化部（214）は、前記修正済みの構造データに基づいて前記複数の処理ブロックの並列化を実施する。結果出力部（215）は、前記並列化の結果を出力する。

明 細 書

発明の名称：

ソフトウェア設計支援システム、ソフトウェア設計支援方法およびソフトウェア設計支援プログラム

技術分野

[0001] 本開示は、ソフトウェアの並列化を支援するための技術に関するものである。

背景技術

[0002] マルチコアプロセッサを用いた並列実行が性能向上に対して有望である。そのため、ソースコードを解析して自動で並列実行用のソースコードを出力する自動並列化技術が期待される。

ソースコードを解析するためのオープンソースのツールとして、コンパイラフロントエンドであるClangとコンパイラバックエンドであるLLVMといったコンパイラ基盤が知られている。

[0003] 特許文献1には、無視可能なデータ依存関係を示す無視情報から同期必須依存関係を抽出し、同期必須依存関係を有する2つの処理の同時実行を回避しつつ並列化効果を向上させる発明が記載されている。

先行技術文献

特許文献

[0004] 特許文献1：特許第6558310号公報

発明の概要

発明が解決しようとする課題

[0005] レガシーコードと呼ばれる古くから存在するソースコードは、並列化されることを前提とした作りになっていない。また、既存の並列化コンパイラは、ソースコードの実装が正しく行われたことを前提にして、ソースコードを解析する。しかし、本質的には依存性がないデータ依存関係が実装時にソー

スコードに作りこまれる。そのため、本質的には依存性がないデータ依存関係が並列化の阻害要因となってしまう。

[0006] 本開示は、本質的には依存性がないデータ依存関係を考慮してソフトウェアを並列化できるようにすることを目的とする。

課題を解決するための手段

[0007] 本開示のソフトウェア設計支援システムは、
複数の処理ブロックを含むソースコードを解析して前記ソースコードの構造データを生成する構造解析部と、
前記構造データに基づいて、互いに影響を与えないデータ依存関係を偽依存として検出する偽依存検出部と、
前記構造データを修正することによって、検出された偽依存が解消された場合の前記ソースコードの構造データを修正済みの構造データとして生成する構造データ修正部と、
前記修正済みの構造データに基づいて前記複数の処理ブロックの並列化を実施する並列化部と、
前記並列化によって得られて前記複数の処理ブロックのスケジュールを示す並列化結果を出力する結果出力部と、を備える。

発明の効果

[0008] 本開示によれば、本質的には依存性がないデータ依存関係（偽依存）を考慮してソフトウェアを並列化することが可能になる。

図面の簡単な説明

[0009] [図1]実施の形態1におけるソフトウェア設計支援システム100の構成図。
[図2]実施の形態1における並列化装置200の構成図。
[図3]実施の形態1における改修候補抽出装置300の構成図。
[図4]実施の形態1におけるソフトウェア設計支援システム100の機能構成図。
[図5]実施の形態1におけるソフトウェア設計支援方法（並列化）のフローチャート。

[図6]実施の形態1におけるソースコード111の例を示す図。

[図7]実施の形態1における構造データ112の例を示す図。

[図8]実施の形態1における並列化結果113の例を示す図。

[図9]実施の形態1における並列化の前後のスケジュールの例を示す図。

[図10]実施の形態1におけるソフトウェア設計支援方法（偽依存）のフローチャート。

[図11]実施の形態1における偽依存一覧123の例を示す図。

[図12]実施の形態1におけるステップS120のフローチャート。

[図13]実施の形態1における構造データ124の例を示す図。

[図14]実施の形態1におけるステップS130のフローチャート。

[図15]実施の形態1におけるソフトウェア設計支援方法（改修候補）のフローチャート。

[図16]実施の形態1における改修候補データ125の例を示す図。

[図17]実施の形態1におけるステップS160のフローチャート。

[図18]実施の形態1におけるWAR依存の補足図。

[図19]従来技術における並列化の前後のスケジュールの例を示す図。

[図20]実施の形態1における並列化の前後のスケジュールの例を示す図。

[図21]実施の形態1における並列化の前後のスケジュールの例を示す図。

[図22]実施の形態2における偽依存改修装置400の構成図。

[図23]実施の形態2におけるソフトウェア設計支援システム100の機能構成図。

[図24]実施の形態2におけるソフトウェア設計支援方法（改修）のフローチャート。

[図25]実施の形態における並列化装置200のハードウェア構成図。

[図26]実施の形態における改修候補抽出装置300のハードウェア構成図。

[図27]実施の形態における偽依存改修装置400のハードウェア構成図。

発明を実施するための形態

[0010] 実施の形態および図面において、同じ要素または対応する要素には同じ符

号を付している。説明した要素と同じ符号が付された要素の説明は適宜に省略または簡略化する。図中の矢印はデータの流れ又は処理の流れを主に示している。

[0011] 実施の形態 1.

ソフトウェア設計支援システム 100 について、図 1 から図 23 に基づいて説明する。

[0012] ***構成の説明***

図 1 に基づいて、ソフトウェア設計支援システム 100 の構成を説明する。

ソフトウェア設計支援システム 100 は、並列化装置 200 と、改修候補抽出装置 300 と、を備える。但し、ソフトウェア設計支援システム 100 は、1 台の装置または 3 台以上の装置で実現されてもよい。

並列化装置 200 と改修候補抽出装置 300 は、ネットワーク 101 を介して互いにデータを受け渡す。但し、データの受け渡しは人手によって行われてもよい。

[0013] 図 2 に基づいて、並列化装置 200 の構成を説明する。

並列化装置 200 は、プロセッサ 201 とメモリ 202 と補助記憶装置 203 と通信装置 204 と入出力インタフェース 205 といったハードウェアを備えるコンピュータである。これらのハードウェアは、信号線を介して互いに接続されている。

[0014] プロセッサ 201 は、演算処理を行う IC であり、他のハードウェアを制御する。例えば、プロセッサ 201 は、CPU、DSP または GPU である。

IC は、Integrated Circuit の略称である。

CPU は、Central Processing Unit の略称である。

DSP は、Digital Signal Processor の略称である。

GPUは、Graphics Processing Unitの略称である。

[0015] メモリ202は揮発性または不揮発性の記憶装置である。メモリ202は、主記憶装置またはメインメモリとも呼ばれる。例えば、メモリ202はRAMである。メモリ202に記憶されたデータは必要に応じて補助記憶装置203に保存される。

RAMは、Random Access Memoryの略称である。

[0016] 補助記憶装置203は不揮発性の記憶装置である。例えば、補助記憶装置203は、ROM、HDDまたはフラッシュメモリである。補助記憶装置203に記憶されたデータは必要に応じてメモリ202にロードされる。

ROMは、Read Only Memoryの略称である。

HDDは、Hard Disk Driveの略称である。

[0017] 通信装置204はレシーバ及びトランスミッタである。例えば、通信装置204は通信チップまたはNICである。並列化装置200の通信は通信装置204を用いて行われる。

NICは、Network Interface Cardの略称である。

[0018] 入出力インタフェース205は、入力装置および出力装置が接続されるポートである。例えば、入出力インタフェース205はUSB端子であり、入力装置はキーボードおよびマウスであり、出力装置はディスプレイである。並列化装置200の入出力は入出力インタフェース205を用いて行われる。

USBは、Universal Serial Busの略称である。

[0019] 並列化装置200は、ソースコード受付部211と構造解析部212と修正依頼部213と並列化部214と結果出力部215といった要素を備える。これらの要素はソフトウェアで実現される。

[0020] 補助記憶装置203には、ソースコード受付部211と構造解析部212と修正依頼部213と並列化部214と結果出力部215としてコンピュー

タを機能させるための並列化プログラムが記憶されている。並列化プログラムは、メモリ202にロードされて、プロセッサ201によって実行される。

補助記憶装置203には、さらに、OSが記憶されている。OSの少なくとも一部は、メモリ202にロードされて、プロセッサ201によって実行される。

プロセッサ201は、OSを実行しながら、並列化プログラムを実行する。

OSは、Operating Systemの略称である。

[0021] 並列化プログラムの入出力データは記憶部290に記憶される。

メモリ202は記憶部290として機能する。但し、補助記憶装置203、プロセッサ201内のレジスタおよびプロセッサ201内のキャッシュメモリなどの記憶装置が、メモリ202の代わりに、又は、メモリ202と共に、記憶部290として機能してもよい。

[0022] 並列化装置200は、プロセッサ201を代替する複数のプロセッサを備えてもよい。

[0023] 図3に基づいて、改修候補抽出装置300の構成を説明する。

改修候補抽出装置300は、プロセッサ301とメモリ302と補助記憶装置303と通信装置304と入出力インタフェース305といったハードウェアを備えるコンピュータである。これらのハードウェアは、信号線を介して互いに接続されている。

[0024] プロセッサ301は、演算処理を行うICであり、他のハードウェアを制御する。例えば、プロセッサ301はCPU、DSPまたはGPUである。

メモリ302は揮発性または不揮発性の記憶装置である。メモリ302は、主記憶装置またはメインメモリとも呼ばれる。例えば、メモリ302はRAMである。メモリ302に記憶されたデータは必要に応じて補助記憶装置303に保存される。

補助記憶装置303は不揮発性の記憶装置である。例えば、補助記憶装置

303は、ROM、HDDまたはフラッシュメモリである。補助記憶装置303に記憶されたデータは必要に応じてメモリ302にロードされる。

通信装置304はレシーバ及びトランスミッタである。例えば、通信装置304は通信チップまたはNICである。改修候補抽出装置300の通信は通信装置304を用いて行われる。

入出力インタフェース305は、入力装置および出力装置が接続されるポートである。例えば、入出力インタフェース305はUSB端子であり、入力装置はキーボードおよびマウスであり、出力装置はディスプレイである。改修候補抽出装置300の入出力は入出力インタフェース305を用いて行われる。

[0025] 改修候補抽出装置300は、制約受付部311と定義取得部312と偽依存検出部313と構造データ修正部314と改修候補抽出部315と改修候補出力部316といった要素を備える。これらの要素はソフトウェアで実現される。

[0026] 補助記憶装置303には、制約受付部311と定義取得部312と偽依存検出部313と構造データ修正部314と改修候補抽出部315と改修候補出力部316としてコンピュータを機能させるための改修候補抽出プログラムが記憶されている。改修候補抽出プログラムは、メモリ302にロードされて、プロセッサ301によって実行される。

補助記憶装置303には、さらに、OSが記憶されている。OSの少なくとも一部は、メモリ302にロードされて、プロセッサ301によって実行される。

プロセッサ301は、OSを実行しながら、改修候補抽出プログラムを実行する。

[0027] 改修候補抽出プログラムの入出力データは記憶部390に記憶される。

メモリ302は記憶部390として機能する。但し、補助記憶装置303、プロセッサ301内のレジスタおよびプロセッサ301内のキャッシュメモリなどの記憶装置が、メモリ302の代わりに、又は、メモリ302と共に

に、記憶部３９０として機能してもよい。

[0028] 改修候補抽出装置３００は、プロセッサ３０１を代替する複数のプロセッサを備えてもよい。

[0029] 図４に、ソフトウェア設計支援システム１００の機能構成を示す。図中の各矢印はデータの流れ又は処理の流れを示している。

ソフトウェア設計支援システム１００における各要素の動作および各データの内容について後述する。

[0030] ***動作の説明***

ソフトウェア設計支援システム１００の動作の手順はソフトウェア設計支援方法に相当する。また、ソフトウェア設計支援システム１００の動作の手順はソフトウェア設計支援プログラムによる処理の手順に相当する。

ソフトウェア設計支援プログラムは、並列化プログラムと改修候補抽出プログラムとを含む。ソフトウェア設計支援プログラムは、光ディスクまたはフラッシュメモリ等の不揮発性の記録媒体にコンピュータ読み取り可能に記録（格納）することができる。

[0031] 図５に基づいて、ソフトウェア設計支援方法（並列化）を説明する。

ソフトウェア設計支援方法（並列化）は、ソースコード１１１の中の複数の処理ブロックを並列化させる方法であり、並列化装置２００によって実行される。

処理ブロックは、並列に実行される最小の処理単位である。処理ブロックの具体例は関数である。

[0032] ステップＳ１０１において、ソースコード受付部２１１は、ソースコード１１１を受け付ける。

例えば、利用者がソースコード１１１を並列化装置２００に入力し、ソースコード受付部２１１は入力されたソースコード１１１を受け付ける。

ソースコード１１１は複数の処理ブロックを含む。

少なくともいずれかの処理ブロック間には、偽依存というデータ依存関係がある。

データ依存関係は、一方の処理ブロックのデータアクセスが他方の処理ブロックのデータアクセスに影響を与える関係である。

偽依存は、偽のデータ依存関係を意味する。つまり、偽依存は、2つの処理ブロックが同じ名前のデータにアクセスするがそれぞれのデータアクセスが互いに影響を与えない関係を意味する。例えば、偽依存は、本質的にはデータ依存関係がないにも関わらず実装時にソースコード111に作りこまれる。

真のデータ依存関係を「真依存」と称する。

[0033] 図6に、ソースコード111の具体例を示す。

ソースコード111は、`func1()`と`func2()`と`func3()`といった関数(処理ブロック)を含んでいる。ソースコード111のファイル名は「`mysample1.c`」である。

ソースコード111において、`func1()`と`func2()`の組と`func2()`と`func3()`の組とのそれぞれにWAR依存がある。WARはWrite After Readの略称である。

WAR依存は、偽依存の具体例であり、データがリードされた後に同じデータがライトされる関係を意味する。

ソースコード111において、`func2()`と`func3()`の組にRAW依存がある。RAWはRead After Writeの略称である。

RAW依存は、真依存の具体例であり、データがライトされた後に同じデータがリードされる関係を意味する。

[0034] 図5に戻り、ステップS102から説明を続ける。

ステップS102において、構造解析部212は、ソースコード111を解析し、ソースコード111の構造データ112を生成する。例えば、構造解析部212は、ソースコード111を入力にして、既存の解析ツールを実行する。既存の解析ツールの具体例はLLVMまたはIRと呼ばれるツールである。

構造データ 1 1 2 は、順序情報 1 1 2 A とアクセス情報 1 1 2 B を含む。

順序情報 1 1 2 A は、ソースコード 1 1 1 の中の複数の処理ブロックの実行順序を特定する情報である。実行順序の特定には、条件分岐および繰り返しが考慮される。

アクセス情報 1 1 2 B は、それぞれの処理ブロックの中のデータアクセスを特定する情報である。

[0035] 図 7 に、ソースコード 1 1 1（図 6 を参照）の構造データ 1 1 2 を示す。

順序情報 1 1 2 A は関数間の呼び出し関係を示している。関数間の呼び出し関係によって、複数の関数の実行順序が特定される。具体的には、順序情報 1 1 2 A の各行は「呼び出し元関数名、呼び出し先関数名、呼び出し行番号」を示している。関数の呼び出し行番号は、ソースコード 1 1 1 において呼び出し先関数が呼び出し元関数を呼び出す命令が記載されている行の番号である。

アクセス情報 1 1 2 B の各行は「変数名、関数名、アクセスタイプ、行番号、ファイル名」を示している。アクセスタイプはリード（r）とライト（w）といったデータアクセスの種類を示す。

[0036] 図 5 に戻り、ステップ S 1 0 3 から説明を続ける。

ステップ S 1 0 3 において、修正依頼部 2 1 3 は、ソースコード 1 1 1 の構造データ 1 1 2 の修正を改修候補抽出装置 3 0 0 に依頼する。

具体的には、修正依頼部 2 1 3 は、ソースコード 1 1 1 の構造データ 1 1 2 を改修候補抽出装置 3 0 0 に受け渡す。

構造データ 1 1 2 の修正により、構造データ 1 1 2 は、偽依存が解消された場合のソースコード 1 1 1 の構造データ 1 2 4 に更新される。

[0037] ステップ S 1 0 4 において、修正依頼部 2 1 3 は、構造データ 1 2 4 を改修候補抽出装置 3 0 0 から受け取る。

構造データ 1 2 4 は、修正済みの構造データ 1 1 2 である。構造データ 1 2 4 の形式は構造データ 1 1 2 の形式と同じである。

[0038] ステップ S 1 0 5 において、並列化部 2 1 4 は、構造データ 1 2 4 に基づ

いて、ソースコード 1 1 1 の中の複数の処理ブロックの並列化を実施する。

例えば、並列化部 2 1 4 は、ソースコード 1 1 1 と構造データ 1 2 4 とを入力にして、既存の並列化ツールを実行する。

[0039] そして、並列化部 2 1 4 は、並列化結果 1 1 3 を生成する。

並列化結果 1 1 3 は、マルチコアプロセッサに対する複数の処理ブロックの割り当てと、複数の処理ブロックのスケジュールと、を示すデータである。

[0040] 図 8 に、ソースコード 1 1 1（図 6 を参照）の並列化結果 1 1 3 を示す。

並列化結果 1 1 3 は、処理ブロックごとに、ブロック名と割り当てコアと開始時刻と終了時刻とを示している。割り当てコアは処理ブロックが割り当てられるプロセッサコアである。開始時刻および終了時刻は処理ブロックのスケジュールを特定する。処理ブロックは、開始時刻から終了時刻まで実行される。

[0041] 図 9 に、ソースコード 1 1 1（図 6 を参照）の並列化の前後における 3 つの処理ブロックの実行タイミングを示す。

並列化前の場合、3 つの処理ブロック（`func 1`、`func 2`、`func 3`）が順に実行される。`func 1` と `func 2` の間に偽依存がある。また、`func 2` と `func 3` の間に真依存がある。

並列化後の場合、偽依存が解消され、2 つのプロセッサコア（コア 0、コア 1）によって 1 つの処理ブロック（`func 1`）と 2 つの処理ブロック（`func 2`、`func 3`）が並列に実行される。

[0042] 図 5 に戻り、ステップ S 1 0 6 を説明する。

ステップ S 1 0 6 において、結果出力部 2 1 5 は、並列化結果 1 1 3 を出力する。

例えば、結果出力部 2 1 5 は、並列化結果 1 1 3 をディスプレイに表示する。また、結果出力部 2 1 5 は、並列化結果 1 1 3 を改修候補抽出装置 3 0 0 に受け渡す。

[0043] 図 1 0 に基づいて、ソフトウェア設計支援方法（偽依存）を説明する。

ソフトウェア設計支援方法（偽依存）は、偽依存が存在するソースコード 111 の構造データ 112 を偽依存が解消された場合のソースコード 111 の構造データ 124 に修正する方法であり、改修候補抽出装置 300 によって実行される。

[0044] ステップ S 111 において、制約受付部 311 は、制約データ 121 を受け付ける。

例えば、利用者が制約データ 121 を改修候補抽出装置 300 に入力し、制約受付部 311 は入力された制約データ 121 を受け付ける。

制約データ 121 は、解消してはいけない偽依存を指定するためのデータである。

[0045] ステップ S 112 において、定義取得部 312 は、定義データ 122 を取得する。

例えば、定義データ 122 は記憶部 390 に予め記憶され、定義取得部 312 は記憶部 390 から定義データ 122 を取得する。

定義データ 122 は、偽依存の定義を示す。

[0046] ステップ S 113 において、偽依存検出部 313 は、構造データ 112 を並列化装置 200 から受け取る。

[0047] ステップ S 120 において、偽依存検出部 313 は、構造データ 112 を解析することによって、ソースコード 111 に存在する 1 つ以上の偽依存を検出する。

このとき、制約データ 121 によって指定される偽依存を除いて、定義データ 122 に示される定義に合う偽依存が検出される。

[0048] そして、偽依存検出部 313 は、偽依存一覧 123 を生成する。

偽依存一覧 123 は、検出された 1 つ以上の偽依存を特定する情報を示すデータである。

[0049] 図 11 に、ソースコード 111（図 6 を参照）の偽依存一覧 123 を示す。

偽依存一覧 123 は、ソースコード 111 に存在する複数の偽依存（WA

R依存)を示している。

具体的には、偽依存一覧123は、処理ブロック組をヘッダとして示し、ヘッダの下にライトアクセスの情報を示している。

例えば、ヘッダ「func1-func2」は、リードアクセスを行う処理ブロック(func1)とライトアクセスを行う処理ブロック(func2)との組を示している。また、ヘッダ「func1-func2」の下で1つ目の行は、変数aについてfunc1とfunc2の間にWAR依存が存在し、mysample1.cの15行目で変数aのライトアクセスが実行されることを示している。

図11において、変数bに関する記載は省略している。

[0050] 図12に基づいて、ステップS120の手順を説明する。

ステップS121において、偽依存検出部313は、アクセス情報112Bから、データを選択する。選択されたデータを「選択データ」と称する。

[0051] ステップS122において、偽依存検出部313は、アクセス情報112Bから、選択データのアクセス情報を選択する。選択されるアクセス情報を「選択アクセス情報」と称する。また、選択アクセス情報が示すデータアクセスを「選択アクセス」と称する。

[0052] ステップS123において、偽依存検出部313は、順序情報112Aを参照し、選択アクセスを実行する処理ブロックの次の処理ブロックを判定する。

次に、偽依存検出部313は、次の処理ブロックによる選択データへのアクセスを示すアクセス情報をアクセス情報112Bから抽出する。抽出されるアクセス情報を「次アクセス情報」と称する。また、次アクセス情報が示すデータアクセスを「次アクセス」と称する。

そして、偽依存検出部313は、選択アクセス情報と次アクセス情報を比較し、選択アクセスと次アクセスが偽依存であるか判定する。

選択アクセス情報のアクセスタイプがリード(r)であり、次アクセス情報のアクセスタイプがライト(w)であれば、選択アクセスと次アクセスは

偽依存（W A R 依存）である。

選択アクセスと次アクセスが偽依存である場合、処理はステップ S 1 2 4 に進む。

選択アクセスと次アクセスが偽依存でない場合、処理はステップ S 1 2 5 に進む。

[0053] ステップ S 1 2 4 において、偽依存検出部 3 1 3 は、選択アクセスと次アクセスの偽依存を特定する情報を偽依存一覧 1 2 3 に登録する。

具体的には、偽依存検出部 3 1 3 は、処理ブロック組（ヘッダ）が偽依存一覧 1 2 3 に記載されていなければ処理ブロック組を偽依存一覧 1 2 3 に記載する。また、偽依存検出部 3 1 3 は、次アクセス情報（ライトアクセス情報）を偽依存一覧 1 2 3 に記載する。

[0054] ステップ S 1 2 5 において、偽依存検出部 3 1 3 は、選択データの未選択のアクセス情報があるか判定する。

選択データの未選択のアクセス情報がある場合、処理はステップ S 1 2 2 に進む。

選択データの未選択のアクセス情報がない場合、処理はステップ S 1 2 6 に進む。

[0055] ステップ S 1 2 6 において、偽依存検出部 3 1 3 は、未選択のデータがあるか判定する。

未選択のデータがある場合、処理はステップ S 1 2 1 に進む。

未選択のデータがない場合、処理は終了する。

[0056] 図 1 0 に戻り、ステップ S 1 3 0 から説明を続ける。

ステップ S 1 3 0 において、構造データ修正部 3 1 4 は、偽依存一覧 1 2 3 に基づいて構造データ 1 1 2 を修正することによって、構造データ 1 2 4 を生成する。

構造データ 1 2 4 は、修正済みの構造データ 1 1 2 である。

[0057] 図 1 3 に、構造データ 1 1 2（図 7 を参照）を修正して得られる構造データ 1 2 4 を示す。

構造データ 1 2 4 は、順序情報 1 2 4 A とアクセス情報 1 2 4 B を含む。

順序情報 1 2 4 A は、順序情報 1 1 2 A（図 7 を参照）と同じである。

アクセス情報 1 2 4 B は、修正済みのアクセス情報 1 1 2 B（図 7 を参照）である。

WAR 依存は、リネームによって解消される。つまり、WAR 依存は、データ名（変数名）を変更することによって解消される。

アクセス情報 1 2 4 B において、func 2 と func 3 のそれぞれのアクセス情報において変数 a および変数 c がリネームされている。

図 1 3 において、変数 b に関する記載は省略している。

[0058] 図 1 4 に基づいて、ステップ S 1 3 0 の手順を説明する。

ステップ S 1 3 1 において、構造データ修正部 3 1 4 は、偽依存一覧 1 2 3 から偽依存を選択する。

[0059] ステップ S 1 3 2 において、構造データ修正部 3 1 4 は、選択された偽依存に対応するアクセス情報を構造データ 1 1 2 から抽出し、抽出されたアクセス情報を選択された偽依存が解消されるように修正する。

具体的には、構造データ修正部 3 1 4 は、選択された偽依存（WAR 依存）のライトアクセスに対応するライトアクセス情報を構造データ 1 1 2 から抽出し、抽出されたアクセス情報の中のデータ名をリネームする。例えば、データ名の末尾が数字であれば、構造データ修正部 3 1 4 は、その数字を 1 増やした値に変更する。また、データ名の末尾が数字でなければ、構造データ修正部 3 1 4 は、データ名の末尾に数字（例えば“1”）を付加する。

[0060] ステップ S 1 3 3 において、構造データ修正部 3 1 4 は、未選択の偽依存があるか判定する。

未選択の偽依存がある場合、処理はステップ S 1 3 1 に進む。

未選択の偽依存がない場合、処理は終了する。

[0061] 図 1 0 に戻り、ステップ S 1 4 1 を説明する。

ステップ S 1 4 1 において、構造データ修正部 3 1 4 は、構造データ 1 2 4 を並列化装置 2 0 0 に受け渡す。

[0062] 図15に基づいて、ソフトウェア設計支援方法（改修候補）を説明する。

ソフトウェア設計支援方法（改修候補）は、偽依存一覧123から改修候補を抽出する方法であり、改修候補抽出装置300によって実行される。

改修候補は、並列化結果113に示されるスケジュール通りに複数の処理ブロックを実行するために改修が必要な偽依存である。

[0063] ステップS151において、改修候補抽出部315は、並列化結果113を並列化装置200から受け取る。

[0064] ステップS160において、改修候補抽出部315は、並列化結果113に示される複数の処理ブロックのスケジュールに基づいて、改修候補となる偽依存を偽依存一覧123から抽出する。

[0065] そして、改修候補抽出部315は、改修候補データ125を生成する。

改修候補データ125は、改修候補となる偽依存についての情報を示すデータである。

[0066] 図16に、改修候補データ125の具体例を示す。

改修候補データ125は、改修候補となる偽依存（WAR依存）と、偽依存の解消方法（リネーム）と、を示している。

[0067] 図17に基づいて、ステップS160の手順を説明する。

ステップS161において、改修候補抽出部315は、並列化結果113から処理ブロック組を選択する。処理ブロック組は2つの処理ブロックである。

処理ブロック組において、開始時刻が早い方の処理ブロックを「先行ブロック」と称し、開始時刻が遅い方の処理ブロックを「後行ブロック」と称する。

[0068] ステップS162において、改修候補抽出部315は、並列化結果113を参照し、先行ブロックの終了前に後行ブロックが開始されるか判定する。

具体的には、改修候補抽出部315は、先行ブロックの終了時刻と後行ブロックの開始時刻を比較する。そして、改修候補抽出部315は、後行ブロックの開始時刻が先行ブロックの終了時刻より先であるか判定する。

先行ブロックの終了前に後行ブロックが開始される場合、処理はステップ S 1 6 3 に進む。

先行ブロックの終了前に後行ブロックが開始されない場合、処理はステップ S 1 6 4 に進む。

[0069] ステップ S 1 6 3 において、改修候補抽出部 3 1 5 は、先行ブロックと後行ブロックの組の偽依存を偽依存一覧 1 2 3 から抽出する。

そして、改修候補抽出部 3 1 5 は、抽出された偽依存の情報を改修候補データ 1 2 5 に登録する。

[0070] ステップ S 1 6 4 において、改修候補抽出部 3 1 5 は、未選択のブロック組があるか判定する。

未選択のブロック組がある場合、処理はステップ S 1 6 1 に進む。

未選択のブロック組がない場合、処理は終了する。

[0071] 図 1 5 に戻り、ステップ S 1 7 1 を説明する。

ステップ S 1 7 1 において、改修候補出力部 3 1 6 は、改修候補データ 1 2 5 を出力する。

例えば、改修候補出力部 3 1 6 は、改修候補データ 1 2 5 をディスプレイに表示する。

[0072] 図 1 8 に基づいて、W A R 依存について補足する。

W A R 依存は、[元コード] のように変数を使いまわすようなプログラムによくみられる。

[元コード] において、変数 `val` に対する処理 (A) の `read` と変数 `val` に対する処理 (B) の `write` との間に W A R 依存が成り立っている。

[並列化修正後コード] のように処理 (A) と処理 (B) の少なくとも一方の変数 `val` を別名の変数 `val 2` にリネームすることで、[元コード] の W A R 依存は解消される。

[0073] ***実施の形態 1 の効果***

実施の形態 1 は、アイデア (1) およびアイデア (2) により、偽依存を

解消して並列化の性能を向上させることができる。

アイデア（１）は、ソースコードに存在する偽依存を抽出して解消することである。ソフトウェア設計支援システム１００は、コンパイラフロントエンドのようなＳ／Ｗ構造解析ツールを用いてソースコードのＳ／Ｗ構造情報を抽出し、そのＳ／Ｗ構造情報をもとに偽依存を抽出する。次に、ソフトウェア設計支援システム１００は、抽出した偽依存が解消されるようにＳ／Ｗ構造情報を修正し、そのＳ／Ｗ構造情報をもとに並列化を行う。そして、ソフトウェア設計支援システム１００は、偽依存が解消された場合の最適な並列化結果を出力する。「Ｓ／Ｗ」はソフトウェアを意味する。

アイデア（２）は、並列化結果が示す並列実行のために必要十分な解消すべき偽依存を抽出し、抽出した偽依存を偽依存改修候補として出力することである。ソフトウェア設計支援システム１００は、アイデア（１）の並列化結果が示す並列実行を行うために必要十分な解消すべき偽依存を抽出し、抽出した偽依存を偽依存改修候補として提示する。偽依存を解消していいかどうかを解析対象としたソースコードの解析だけで判断することは困難であり、Ｓ／Ｗ（解析対象を除く）およびＨ／Ｗによるデータアクセスの有無などを考慮して設計者が判断する必要がある。そのため、偽依存改修候補が提示される。また、アイデア（１）の並列化結果が示す並列実行を行うためには必ずしも全ての偽依存の解消が必要ではない。そのため、偽依存改修候補となる偽依存が必要十分な解消すべき偽依存に絞り込まれる。これにより、設計者に強いる判断の量を減らす効果が得られる。

[0074] 図１９は、従来技術による並列化の前後のスケジュールを示している。

従来技術では、偽依存が真依存と判断される。そのため、細かい粒度での並列化が行われた場合でも図１９に示すような並列化結果が得られる。つまり、ブロックＡとブロックＢの間に偽依存があるため、ブロックＢの実行は、ブロックＡの中の偽依存に関わる処理の実行が終わった後に開始される。なお、細かい粒度での並列化は、タスクのような大きなプログラムの塊を処理単位とした並列化ではなく、関数または命令群のような小さなプログラム

の塊を処理単位とする精密な並列化を意味する。

図20は、実施の形態1による並列化の前後のスケジュールを示している。

実施の形態1では、アイデア(1)により、図20に示すような並列化結果が得られる。つまり、ブロックAとブロックBの間に真依存がないため、ブロックAとブロックBが並列に実行される。また、アイデア(2)により、並列化結果が示す並列実行を行うために解消が必要な偽依存が偽依存改修候補として出力される。図20の場合、並列化結果が示す並列実行を実現するためには、ブロックAとブロックBの間の偽依存を解消する必要がある。そのため、ブロックAとブロックBの間の偽依存が偽依存改修候補として提示される。設計者は、並列化結果および偽依存改修候補を確認し、解消して問題ない偽依存についてソースコードに対して偽依存を解消するための措置を講じる。これにより、偽依存が排除されたソースコードを作成することが可能になる。そして、偽依存が排除されたソースコードを従来の並列化コンパイラの入力にすることで、図20に示すような並列化結果が実現される。また、設計者が解消して問題がある偽依存を並列実行制約として指定し、再度偽依存の解析および並列化結果の出力が行われることで、制約の範囲内で並列化結果および偽依存改修候補が得られる。

[0075] 図21は、実施の形態1による並列化の前後のスケジュールを示している。ブロックAとブロックBの組とブロックBとブロックCの組とのそれぞれに偽依存がある。

実施の形態1では、アイデア(1)により、図20に示すような並列化結果が得られる。つまり、一方のプロセッサコアにブロックAが割り当てられ、他方のプロセッサコアにブロックBとブロックCが割り当てられ、ブロックBとブロックCがブロックAと並列に実行される、という最適なスケジューリングが実現される。また、アイデア(2)により、並列化結果が示す並列実行を行うために解消が必要な偽依存が偽依存改修候補として出力される。図21の場合、並列化結果が示す並列実行を実現するためには、ブロック

AとブロックBの間の偽依存を解消する必要がある。ブロックBとブロックCの間の偽依存を解消する必要はない。そのため、ブロックAとブロックBの間の偽依存が偽依存改修候補として提示される。つまり、2つの偽依存を解消して最適な並列化結果が得られ、最適な並列化結果が示す並列実行を実現するために1つの偽依存の改修が必要である。そのため、1つの偽依存が偽依存改修候補として提示される。

[0076] 従来技術では、偽依存に関わる並列性を抽出できなかった。一方、実施の形態1は、アイデア(1)により、偽依存を解消できる。そのため、より多くの並列性を抽出することができる。また、アイデア(1)だけでは、自動で解消した全ての偽依存についてソースコードを改修して問題ないかを人手で判断しなくてはならない。しかし、実施の形態1は、アイデア(2)により、並列化結果が示す並列実行を行うために必要十分な解消すべき偽依存が抽出される。そのため、ソースコードを改修して問題ないかについて人手による判断の量を削減することができる。

[0077] ***実施の形態1の実施例***

WAR依存以外の偽依存の具体例を説明する。

実施例1

WAW依存は、偽依存の具体例であり、データがライトされた後に同じデータがライトされる関係を意味する。WAWは、Write After Writeの略称である。

偽依存検出部313は、偽依存の一つとしてWAW依存を抽出する(ステップS120)。具体的には、選択アクセス情報のアクセスタイプがライト(w)であり、次アクセス情報のアクセスタイプがライト(w)である場合、偽依存検出部313は、選択アクセスと次アクセスが偽依存(WAW依存)であると判定する(ステップS123)。

構造データ修正部314は、WAW依存が解消されるように、構造データ112のアクセス情報を修正する(ステップS132)。

WAW依存は、WAR依存と同じく、少なくとも一方の変数をリネーム(

複製)することによって解消される。

[0078] ***実施例2***

スコープ変数依存は、偽依存の具体例であり、互いに異なるスコープが決められて同じ名前を有する複数の変数に対するアクセスを意味する。

スコープが決められた変数を「スコープ変数」と称する。スコープ変数の具体例は、`static`変数である。

スコープ変数依存は、ソースコードには依存として存在しないにも関わらず、依存として存在する、と並列化の際に誤って解析される。

変数のスコープを関数内（またはファイル内）に限定し、他の関数（または他のファイル）に設けられる変数と同一の変数名の変数を各関数（または各ファイル）に設けることが可能である。並列化では、これらの変数が同一の変数であると誤って解析される可能性がある。このように、並列化の際に別々の変数が同一の変数であると誤って解析されてしまうケースに対しても、構造データ112を修正することによって、並列化の際に誤解析がなされないようにすることができる。

偽依存検出部313は、偽依存の一つとしてスコープ変数依存を抽出する（ステップS120）。

構造データ修正部314は、スコープ変数依存が解消されるように、構造データ112のアクセス情報を修正する（ステップS132）。

スコープ変数依存は、`WAR`依存と同じく、少なくとも一方の変数をリネーム（複製）することによって解消される。

[0079] ***実施例3***

外部宣言依存は、偽依存の具体例であり、互いに異なる外部ファイルで宣言されて同じ名前を有する複数の変数に対するアクセスを意味する。外部ファイルの具体例はライブラリである。例えば、C言語の外部ファイルでは、変数が`extern`宣言される。

外部宣言依存は、ソースコードには依存として存在しないにも関わらず、依存として存在する、と並列化の際に誤って解析される。

同一の変数名の変数が異なるファイルで使用され各変数が異なる外部ファイルで宣言されている場合、並列化では、これらの変数が同一の変数であると誤って解析される可能性がある。このように、並列化の際に別々の変数が同一の変数であると誤って解析されてしまうケースに対しても、構造データ 1 1 2 を修正することによって、並列化の際に誤解析がなされないようにすることができる。

構造データ修正部 3 1 4 は、外部宣言依存が解消されるように、構造データ 1 1 2 のアクセス情報を修正する（ステップ S 1 3 2）。

外部宣言依存は、W A R 依存と同じく、少なくとも一方の変数をリネーム（複製）することによって解消される。

[0080] 実施の形態 2.

解消すべき偽依存の解消後の並列化結果を得る形態について、主に実施の形態 1 と異なる点を図 2 2 から図 2 4 に基づいて説明する。

[0081] ***構成の説明***

ソフトウェア設計支援システム 1 0 0 は、さらに、偽依存改修装置 4 0 0 を備える。

[0082] 図 2 2 に基づいて、偽依存改修装置 4 0 0 の構成を説明する。

偽依存改修装置 4 0 0 は、プロセッサ 4 0 1 とメモリ 4 0 2 と補助記憶装置 4 0 3 と通信装置 4 0 4 と入出力インタフェース 4 0 5 といったハードウェアを備えるコンピュータである。これらのハードウェアは、信号線を介して互いに接続されている。

[0083] プロセッサ 4 0 1 は、演算処理を行う I C であり、他のハードウェアを制御する。例えば、プロセッサ 4 0 1 は C P U、D S P または G P U である。

メモリ 4 0 2 は揮発性または不揮発性の記憶装置である。メモリ 4 0 2 は、主記憶装置またはメインメモリとも呼ばれる。例えば、メモリ 4 0 2 は R A M である。メモリ 4 0 2 に記憶されたデータは必要に応じて補助記憶装置 4 0 3 に保存される。

補助記憶装置 4 0 3 は不揮発性の記憶装置である。例えば、補助記憶装置

４０３は、ＲＯＭ、ＨＤＤまたはフラッシュメモリである。補助記憶装置４０３に記憶されたデータは必要に応じてメモリ４０２にロードされる。

通信装置４０４はレシーバ及びトランスミッタである。例えば、通信装置４０４は通信チップまたはＮＩＣである。偽依存改修装置４００の通信は通信装置４０４を用いて行われる。

入出力インタフェース４０５は、入力装置および出力装置が接続されるポートである。例えば、入出力インタフェース４０５はＵＳＢ端子であり、入力装置はキーボードおよびマウスであり、出力装置はディスプレイである。偽依存改修装置４００の入出力は入出力インタフェース４０５を用いて行われる。

[0084] 偽依存改修装置４００は、偽依存受付部４１１と偽依存解析部４１２と構造データ再修正部４１３といった要素を備える。これらの要素はソフトウェアで実現される。

[0085] 補助記憶装置４０３には、偽依存受付部４１１と偽依存解析部４１２と構造データ再修正部４１３としてコンピュータを機能させるための偽依存改修プログラムが記憶されている。偽依存改修プログラムは、メモリ４０２にロードされて、プロセッサ４０１によって実行される。

補助記憶装置４０３には、さらに、ＯＳが記憶されている。ＯＳの少なくとも一部は、メモリ４０２にロードされて、プロセッサ４０１によって実行される。

プロセッサ４０１は、ＯＳを実行しながら、偽依存改修プログラムを実行する。

[0086] 偽依存改修プログラムの入出力データは記憶部４９０に記憶される。

メモリ４０２は記憶部４９０として機能する。但し、補助記憶装置４０３、プロセッサ４０１内のレジスタおよびプロセッサ４０１内のキャッシュメモリなどの記憶装置が、メモリ４０２の代わりに、又は、メモリ４０２と共に、記憶部４９０として機能してもよい。

[0087] 偽依存改修装置４００は、プロセッサ４０１を代替する複数のプロセッサ

を備えてもよい。

[0088] 偽依存改修プログラムは、ソフトウェア設計支援プログラムに含まれる。

[0089] 図23に、ソフトウェア設計支援システム100の機能構成を示す。改修候補抽出装置300の機能構成の図示は省略している。

ソフトウェア設計支援システム100における各要素の動作および各データの内容について後述する。

[0090] ***動作の説明***

図24に基づいて、ソフトウェア設計支援方法（改修）を説明する。

ソフトウェア設計支援方法（改修）は、解消すべき偽依存を解消して並列化結果を得る方法であり、偽依存改修装置400と並列化装置200によって実行される。

[0091] ステップS181において、偽依存受付部411は、偽依存データ131を受け付ける。

偽依存データ131は、実施の形態1の改修候補データ125に相当する。具体的には、偽依存データ131は、改修候補データ125に示される偽依存のうち利用者が改修してよいと判断した偽依存について、偽依存と偽依存の解消方法とを示す。

[0092] ステップS182において、偽依存解析部412は、実施の形態1の構造データ112を受け付ける。

[0093] ステップS183において、偽依存解析部412は、偽依存データ131と構造データ112を解析することによって、偽依存一覧132を生成する。

偽依存一覧132の生成方法は、実施の形態1における偽依存一覧123の生成方法と同じである。偽依存一覧132の形式は、実施の形態1における偽依存一覧123の形式と同じである。

但し、偽依存一覧132には、偽依存データ131に示される各偽依存についてのみ、各偽依存を特定する情報が登録される。

[0094] ステップS184において、構造データ再修正部413は、偽依存一覧1

32に基づいて構造データ112を修正することによって、構造データ133を生成する。

構造データ133は、再修正済みの構造データ112である。

構造データ133の生成方法は、実施の形態1における構造データ124の生成方法と同じである。

[0095] ステップS185において、構造データ再修正部413は、構造データ133を並列化装置200に受け渡す。

修正依頼部213は、偽依存改修装置400から構造データ133を受け取る。

[0096] ステップS186において、並列化部214は、構造データ133に基づいて並列化を実施し、並列化結果114を生成する。

ステップS186は、実施の形態1のステップS105に相当する。

[0097] ステップS187において、結果出力部215は、並列化結果114を出力する。

ステップS187は、実施の形態1のステップS106に相当する。

[0098] ***実施の形態2の効果***

実施の形態2は、実施の形態1の改修候補データ125に示される偽依存のみが解消された場合の並列化結果114を得ることができる。

実施の形態1では、利用者による偽依存の解消可否の判断がなされずに、検出された全ての偽依存が解消された場合の並列化結果113が得られる。

一方、実施の形態2では、利用者によって解消可能と判断された偽依存が解消された場合の並列化結果114を得ることが可能である。

[0099] ***実施の形態の補足***

図25に基づいて、並列化装置200のハードウェア構成を説明する。

並列化装置200は処理回路209を備える。

処理回路209は、ソースコード受付部211と構造解析部212と修正依頼部213と並列化部214と結果出力部215とを実現するハードウェアである。

処理回路209は、専用のハードウェアであってもよいし、メモリ202に格納されるプログラムを実行するプロセッサ201であってもよい。

[0100] 処理回路209が専用のハードウェアである場合、処理回路209は、例えば、単回路、複合回路、プログラム化したプロセッサ、並列プログラム化したプロセッサ、ASIC、FPGAまたはこれらの組み合わせである。

ASICは、Application Specific Integrated Circuitの略称である。

FPGAは、Field Programmable Gate Arrayの略称である。

[0101] 並列化装置200は、処理回路209を代替する複数の処理回路を備えてもよい。

[0102] 処理回路209において、一部の機能が専用のハードウェアで実現されて、残りの機能がソフトウェアまたはファームウェアで実現されてもよい。

[0103] このように、並列化装置200の機能はハードウェア、ソフトウェア、ファームウェアまたはこれらの組み合わせで実現することができる。

[0104] 図26に基づいて、改修候補抽出装置300のハードウェア構成を説明する。

改修候補抽出装置300は処理回路309を備える。

処理回路309は、制約受付部311と定義取得部312と偽依存検出部313と構造データ修正部314と改修候補抽出部315と改修候補出力部316とを実現するハードウェアである。

処理回路309は、専用のハードウェアであってもよいし、メモリ302に格納されるプログラムを実行するプロセッサ301であってもよい。

[0105] 処理回路309が専用のハードウェアである場合、処理回路309は、例えば、単回路、複合回路、プログラム化したプロセッサ、並列プログラム化したプロセッサ、ASIC、FPGAまたはこれらの組み合わせである。

[0106] 改修候補抽出装置300は、処理回路309を代替する複数の処理回路を備えてもよい。

[0107] 処理回路309において、一部の機能が専用のハードウェアで実現されて、残りの機能がソフトウェアまたはファームウェアで実現されてもよい。

[0108] このように、改修候補抽出装置300の機能はハードウェア、ソフトウェア、ファームウェアまたはこれらの組み合わせで実現することができる。

[0109] 図27に基づいて、偽依存改修装置400のハードウェア構成を説明する。

偽依存改修装置400は処理回路409を備える。

処理回路409は、偽依存受付部411と偽依存解析部412と構造データ再修正部413とを実現するハードウェアである。

処理回路409は、専用のハードウェアであってもよいし、メモリ402に格納されるプログラムを実行するプロセッサ401であってもよい。

[0110] 処理回路409が専用のハードウェアである場合、処理回路409は、例えば、単回路、複合回路、プログラム化したプロセッサ、並列プログラム化したプロセッサ、ASIC、FPGAまたはこれらの組み合わせである。

[0111] 偽依存改修装置400は、処理回路409を代替する複数の処理回路を備えてもよい。

[0112] 処理回路409において、一部の機能が専用のハードウェアで実現されて、残りの機能がソフトウェアまたはファームウェアで実現されてもよい。

[0113] このように、偽依存改修装置400の機能はハードウェア、ソフトウェア、ファームウェアまたはこれらの組み合わせで実現することができる。

[0114] 各実施の形態は、好ましい形態の例示であり、本開示の技術的範囲を制限することを意図するものではない。各実施の形態は、部分的に実施してもよいし、他の形態と組み合わせて実施してもよい。フローチャート等を用いて説明した手順は、適宜に変更してもよい。

[0115] ソフトウェア設計支援システム100の要素である「部」は、「処理」、「工程」、「回路」または「サーキットリ」と読み替えてもよい。

符号の説明

[0116] 100 ソフトウェア設計支援システム、101 ネットワーク、111

ソースコード、１１２ 構造データ、１１２Ａ 順序情報、１１２Ｂ アクセス情報、１１３ 並列化結果、１２１ 制約データ、１２２ 定義データ、１２３ 偽依存一覧、１２４ 構造データ、１２４Ａ 順序情報、１２４Ｂ アクセス情報、１２５ 改修候補データ、２００ 並列化装置、２０１ プロセッサ、２０２ メモリ、２０３ 補助記憶装置、２０４ 通信装置、２０５ 入出力インタフェース、２０９ 処理回路、２１１ ソースコード受付部、２１２ 構造解析部、２１３ 修正依頼部、２１４ 並列化部、２１５ 結果出力部、２９０ 記憶部、３００ 改修候補抽出装置、３０１ プロセッサ、３０２ メモリ、３０３ 補助記憶装置、３０４ 通信装置、３０５ 入出力インタフェース、３０９ 処理回路、３１１ 制約受付部、３１２ 定義取得部、３１３ 偽依存検出部、３１４ 構造データ修正部、３１５ 改修候補抽出部、３１６ 改修候補出力部、３９０ 記憶部、４００ 偽依存改修装置、４０１ プロセッサ、４０２ メモリ、４０３ 補助記憶装置、４０４ 通信装置、４０５ 入出力インタフェース、４０９ 処理回路、４１１ 偽依存受付部、４１２ 偽依存解析部、４１３ 構造データ再修正部。

請求の範囲

- [請求項1] 複数の処理ブロックを含むソースコードを解析して前記ソースコードの構造データを生成する構造解析部と、
前記構造データに基づいて、互いに影響を与えないデータ依存関係を偽依存として検出する偽依存検出部と、
前記構造データを修正することによって、検出された偽依存が解消された場合の前記ソースコードの構造データを修正済みの構造データとして生成する構造データ修正部と、
前記修正済みの構造データに基づいて前記複数の処理ブロックの並列化を実施する並列化部と、
前記並列化によって得られて前記複数の処理ブロックのスケジュールを示す並列化結果を出力する結果出力部と、
を備えるソフトウェア設計支援システム。
- [請求項2] 前記ソフトウェア設計支援システムは、
解消してはいけない偽依存を指定する制約データを受け付ける制約受付部を備え、
前記偽依存検出部は、前記制約データに示される偽依存以外の偽依存を検出する
請求項1に記載のソフトウェア設計支援システム。
- [請求項3] 前記偽依存検出部は、データがリードされた後に同じデータがライトされるWAR依存と、データがライトされた後に同じデータがライトされるWAW依存と、互いに異なるスコープが決められて同じ名前を有する複数の変数に対するアクセスを意味するスコープ変数依存と、互いに異なる外部ファイルで宣言されて同じ名前を有する複数の変数に対するアクセスを意味する外部宣言依存との少なくともいずれか、を前記偽依存として検出する
請求項1または請求項2に記載のソフトウェア設計支援システム。
- [請求項4] 前記ソフトウェア設計支援システムは、

検出された偽依存の一覧から、前記並列化結果に示される前記スケジュールの通りに前記複数の処理ブロックを実行するために改修が必要な偽依存を改修候補として抽出する改修候補抽出部と、

前記改修候補となる偽依存についての情報を示す改修候補データを出力する改修候補出力部と、を備える

請求項 1 から請求項 3 のいずれか 1 項に記載のソフトウェア設計支援システム。

[請求項5] 前記改修候補抽出部は、先行ブロックの終了前に後行ブロックが開始される場合に前記先行ブロックと前記後行ブロックの組の偽依存を前記改修候補として抽出する

請求項 4 に記載のソフトウェア設計支援システム。

[請求項6] 前記ソフトウェア設計支援システムは、

前記構造データを再修正することによって、前記改修候補となる前記偽依存が解消された場合の前記ソースコードの構造データを再修正済みの構造データとして生成する構造データ再修正部を備え、

前記並列化部は、前記再修正済みの構造データに基づいて前記複数の処理ブロックの並列化を実施し、

前記結果出力部は、前記並列化によって得られて前記複数の処理ブロックのスケジュールを示す並列化結果を出力する

請求項 4 または請求項 5 に記載のソフトウェア設計支援システム。

[請求項7] 複数の処理ブロックを含むソースコードを解析して前記ソースコードの構造データを生成し、

前記構造データに基づいて、前記ソースコードに存在する互いに影響を与えないデータ依存関係を偽依存として検出し、

前記構造データを修正することによって、検出された偽依存が解消された場合の前記ソースコードの構造データを修正済みの構造データとして生成し、

前記修正済みの構造データに基づいて前記複数の処理ブロックの並

列化を実施し、

前記並列化によって得られて前記複数の処理ブロックのスケジュールを示す並列化結果を出力する

ソフトウェア設計支援方法。

[請求項8]

複数の処理ブロックを含むソースコードを解析して前記ソースコードの構造データを生成する構造解析処理と、

前記構造データに基づいて、互いに影響を与えないデータ依存関係を偽依存として検出する偽依存検出処理と、

前記構造データを修正することによって、検出された偽依存が解消された場合の前記ソースコードの構造データを修正済みの構造データとして生成する構造データ修正処理と、

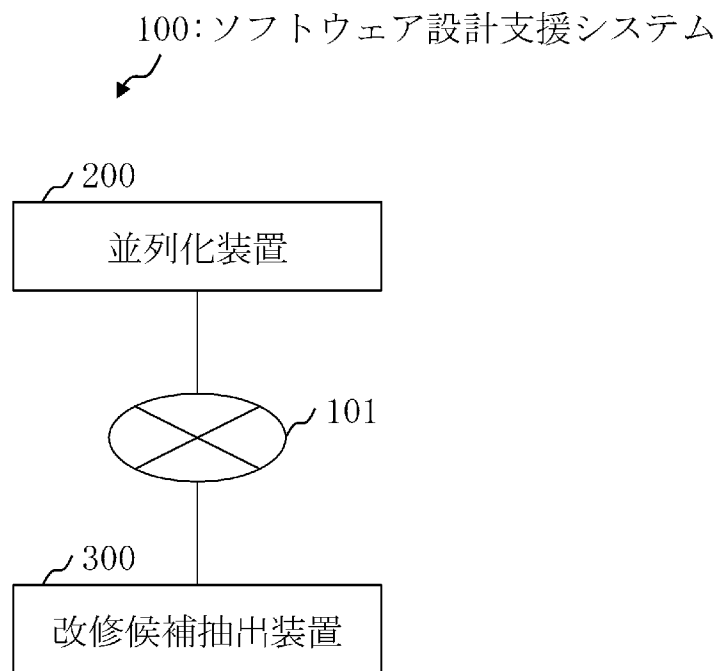
前記修正済みの構造データに基づいて前記複数の処理ブロックの並列化を実施する並列化処理と、

前記並列化によって得られて前記複数の処理ブロックのスケジュールを示す並列化結果を出力する結果出力処理として、

コンピュータを機能させるためのソフトウェア設計支援プログラム。

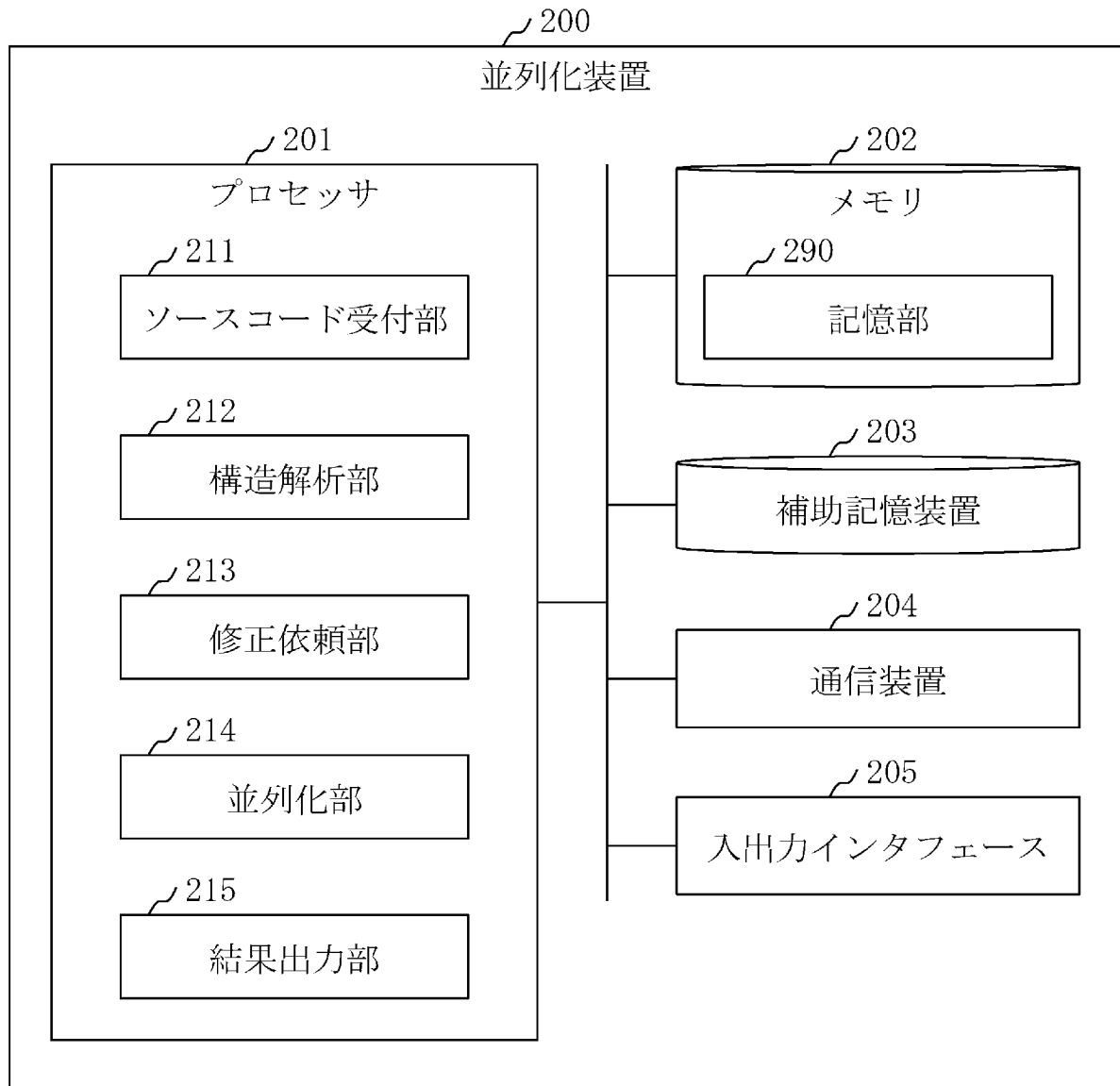
[図1]

図1



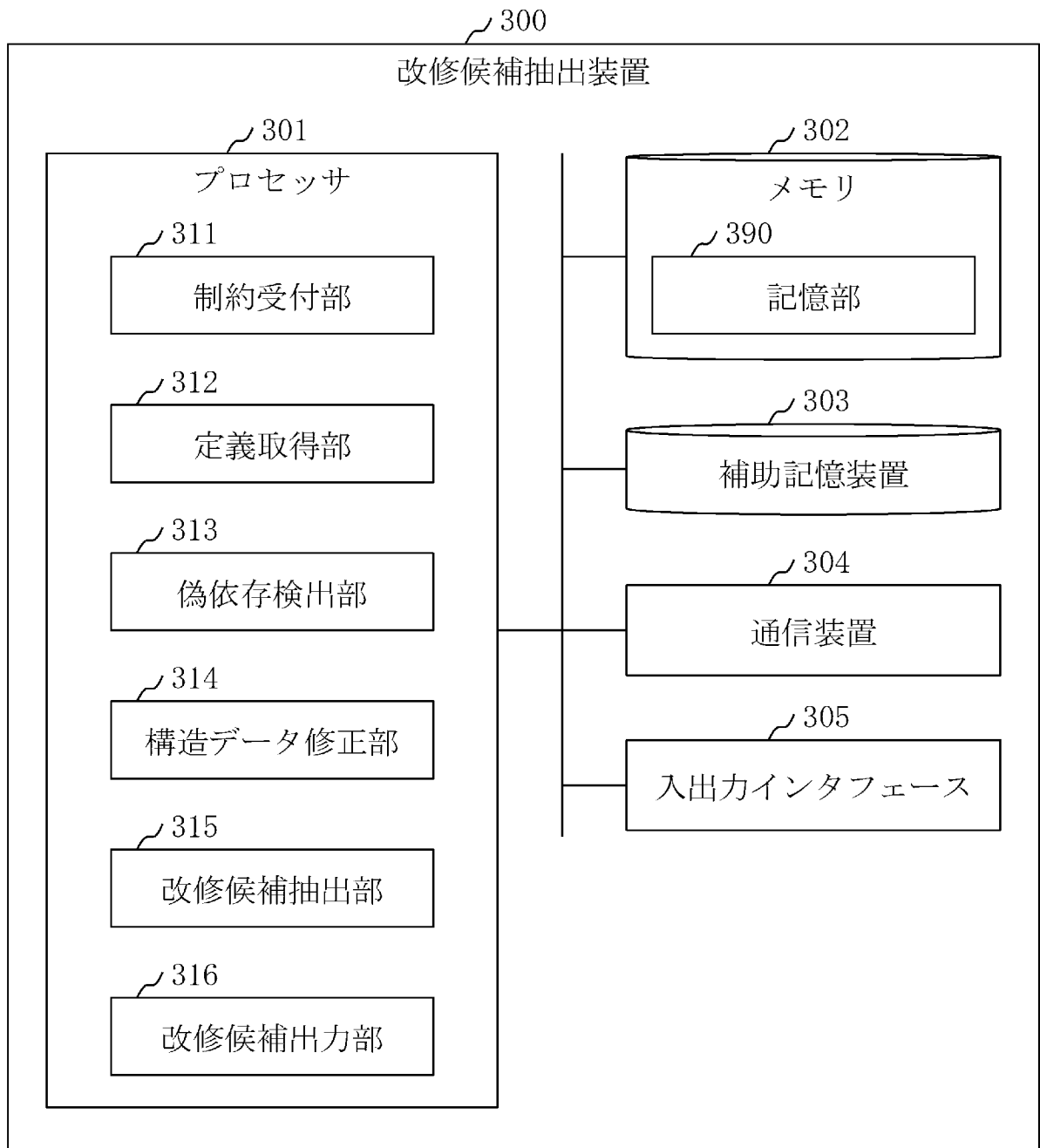
[図2]

図2



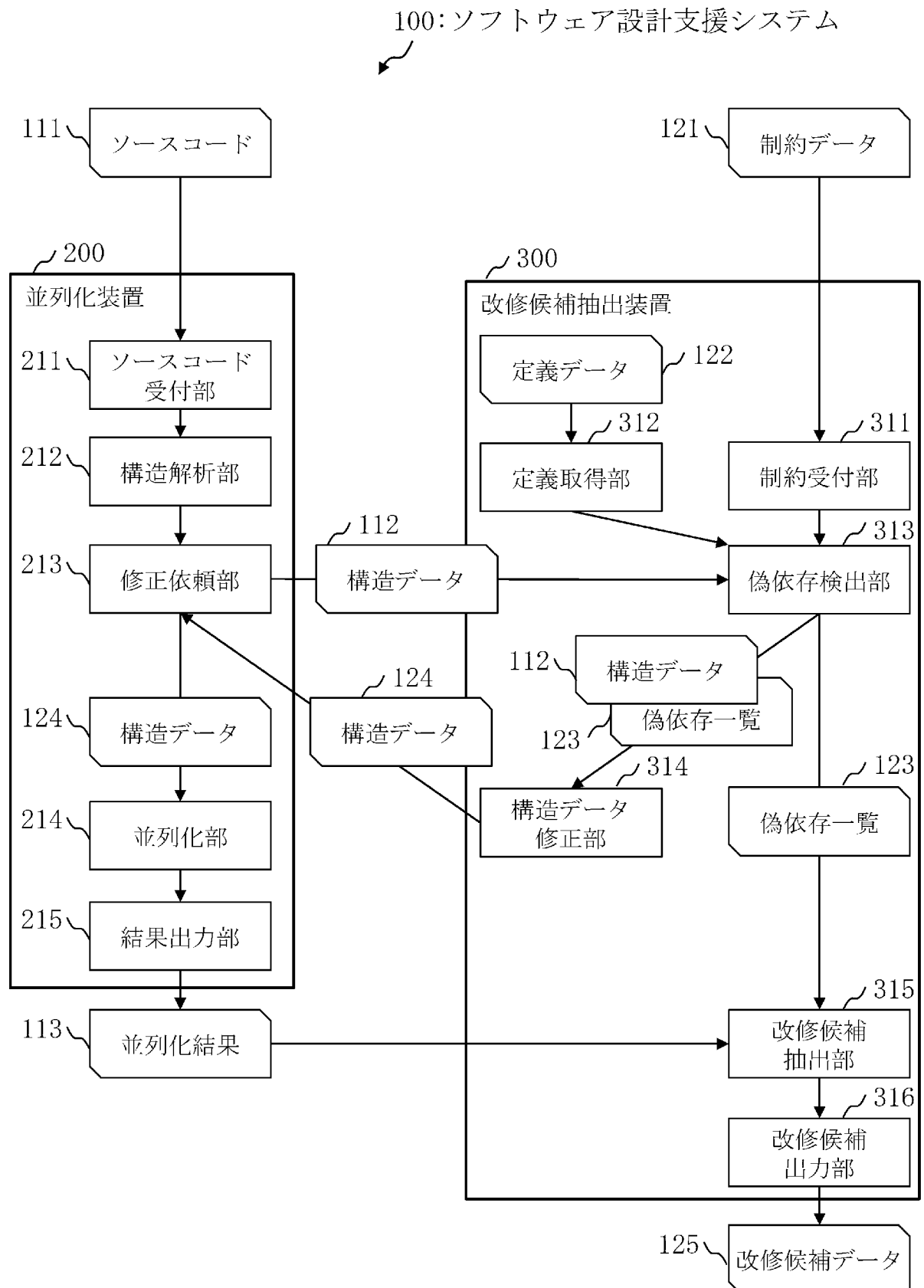
[図3]

図3



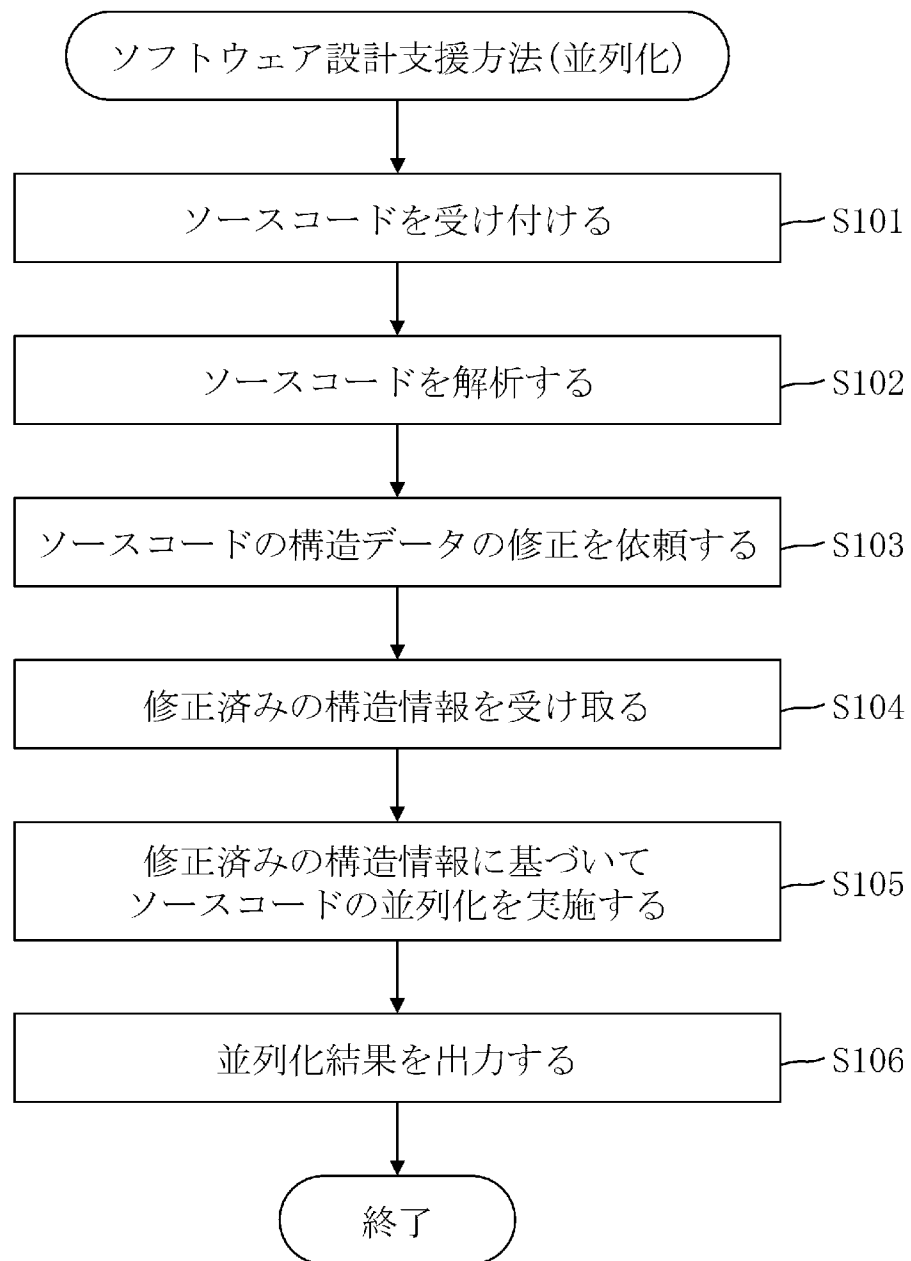
[図4]

図4



[図5]

図5



[図6]

図6

111: ソースコード (mysample1.c)



1	int result1, result2, result3;
2	int a = 0;
3	int b = 0;
4	int c = 0;
5	
6	
7	void func1(){
8	a = 1;
9	b = a;
10	c = b;
11	result1 = c;
12	}
13	
14	void func2(){
15	a = 2;
16	b = a;
17	c = b;
18	result2 = c;
19	}
20	
21	void func3(){
22	a = c;
23	b = a;
24	c = b;
25	result3 = c;
26	}
27	
28	int main(){
29	
30	func1();
31	func2();
32	func3();
33	
34	return (result1 + result2 + result3);
35	}

[図7]

図7

112: 構造データ

112A: 順序情報

```
main,func1,30  
main,func2,31  
main,func3,32
```

112B: アクセス情報

```
a,func3,r,23,mysample1.c  
a,func3,w,22,mysample1.c  
a,func2,r,16,mysample1.c  
a,func2,w,15,mysample1.c  
a,func1,r,9,mysample1.c  
a,func1,w,8,mysample1.c  
b,func3,r,24,mysample1.c  
b,func3,w,23,mysample1.c  
b,func2,r,17,mysample1.c  
b,func2,w,16,mysample1.c  
b,func1,r,10,mysample1.c  
b,func1,w,9,mysample1.c  
c,func3,r,25,mysample1.c  
c,func3,w,24,mysample1.c  
c,func3,r,22,mysample1.c  
c,func2,r,18,mysample1.c  
c,func2,w,17,mysample1.c  
c,func1,r,11,mysample1.c  
c,func1,w,10,mysample1.c  
result1,main,r,34,mysample1.c  
result1,func1,w,11,mysample1.c  
result2,main,r,34,mysample1.c  
result2,func2,w,18,mysample1.c  
result3,main,r,34,mysample1.c  
result3,func3,w,25,mysample1.c
```

[図8]

図8

113:並列化結果

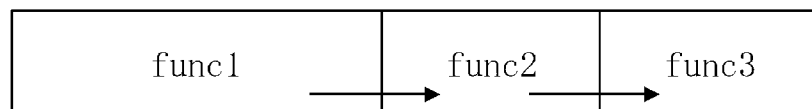


ブロック名	割り当てコア	開始時刻	終了時刻
func1	0	0	8
func2	1	0	5
func3	1	5	10

[図9]

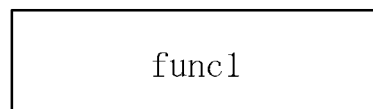
図9

並列化前

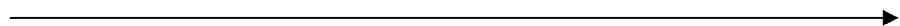
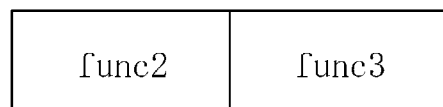


並列化後

コア0

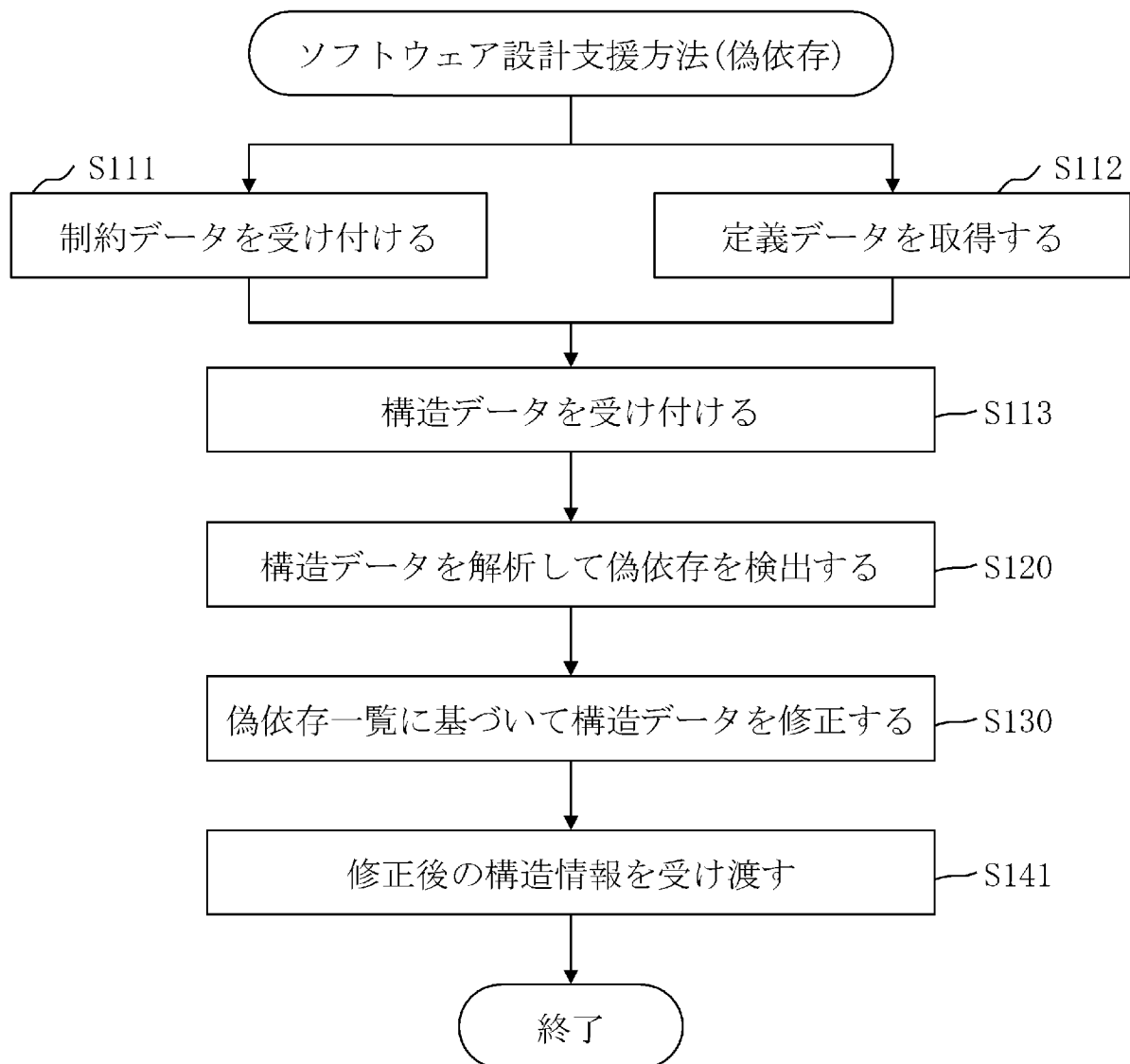


コア1



[図10]

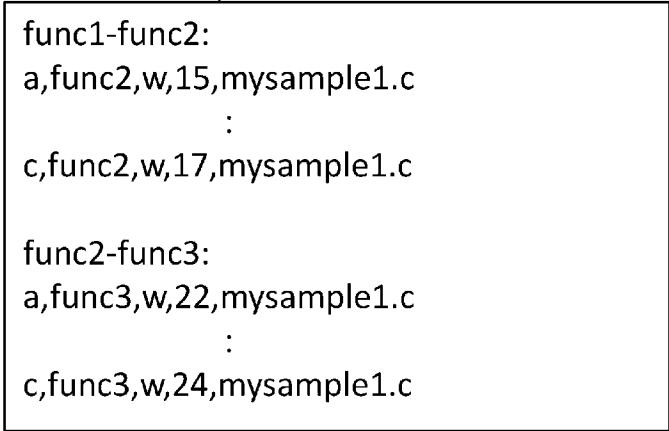
図10



[図11]

図11

123: 偽依存一覧

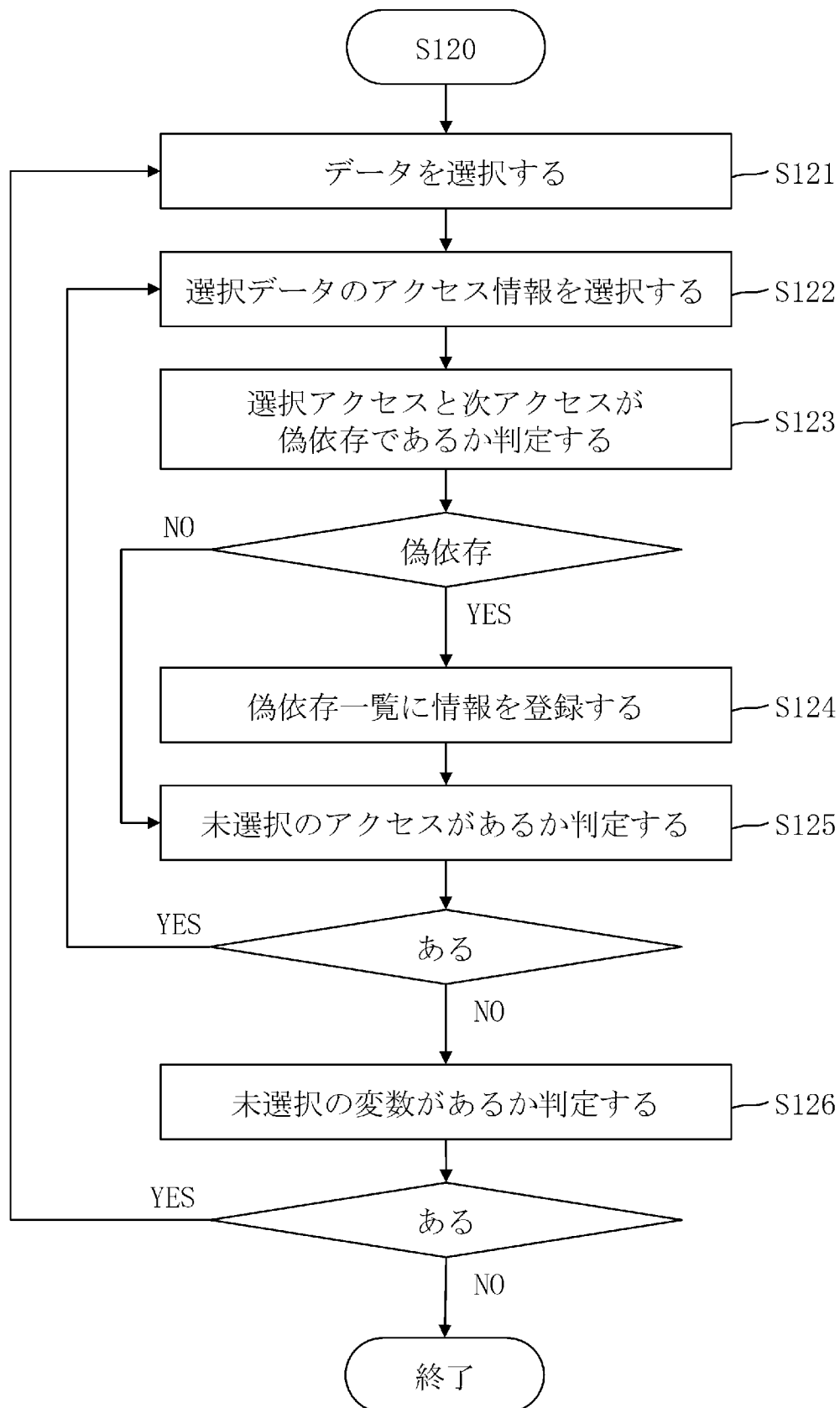


```
func1-func2:
a,func2,w,15,mysample1.c
:
c,func2,w,17,mysample1.c

func2-func3:
a,func3,w,22,mysample1.c
:
c,func3,w,24,mysample1.c
```

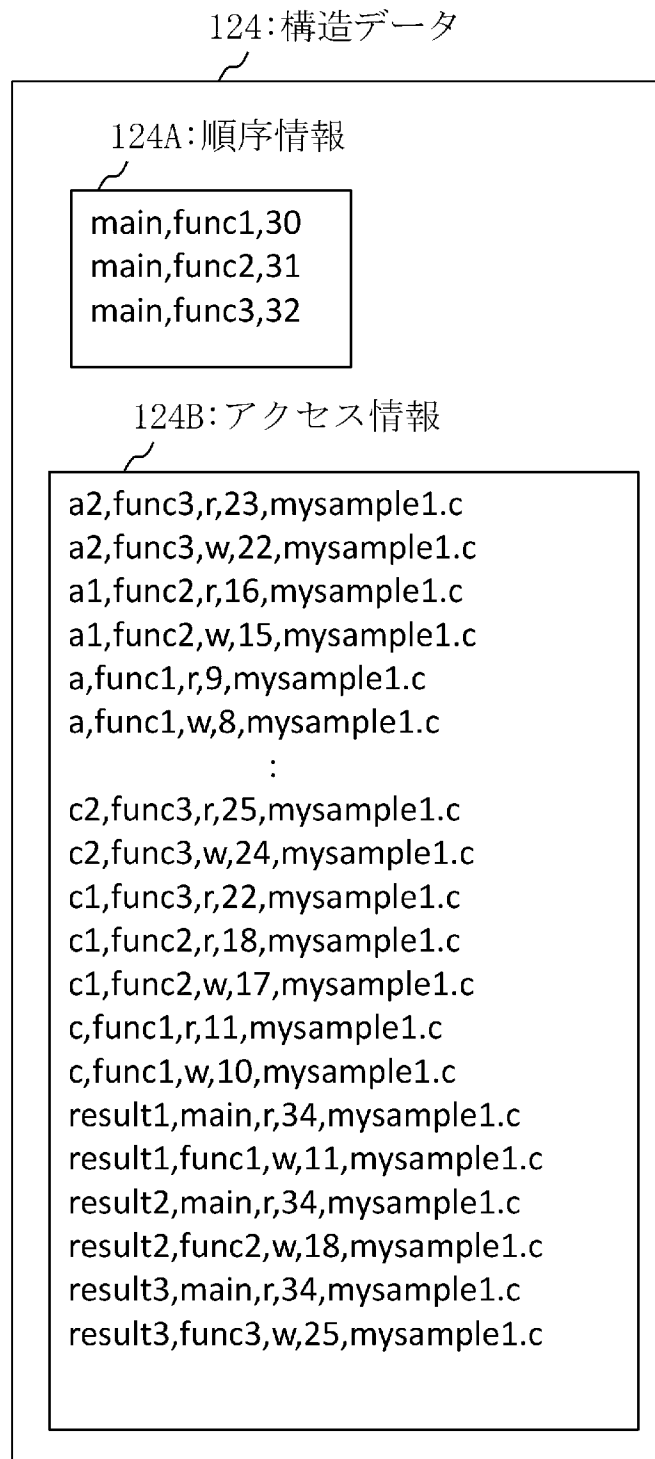
[図12]

図12



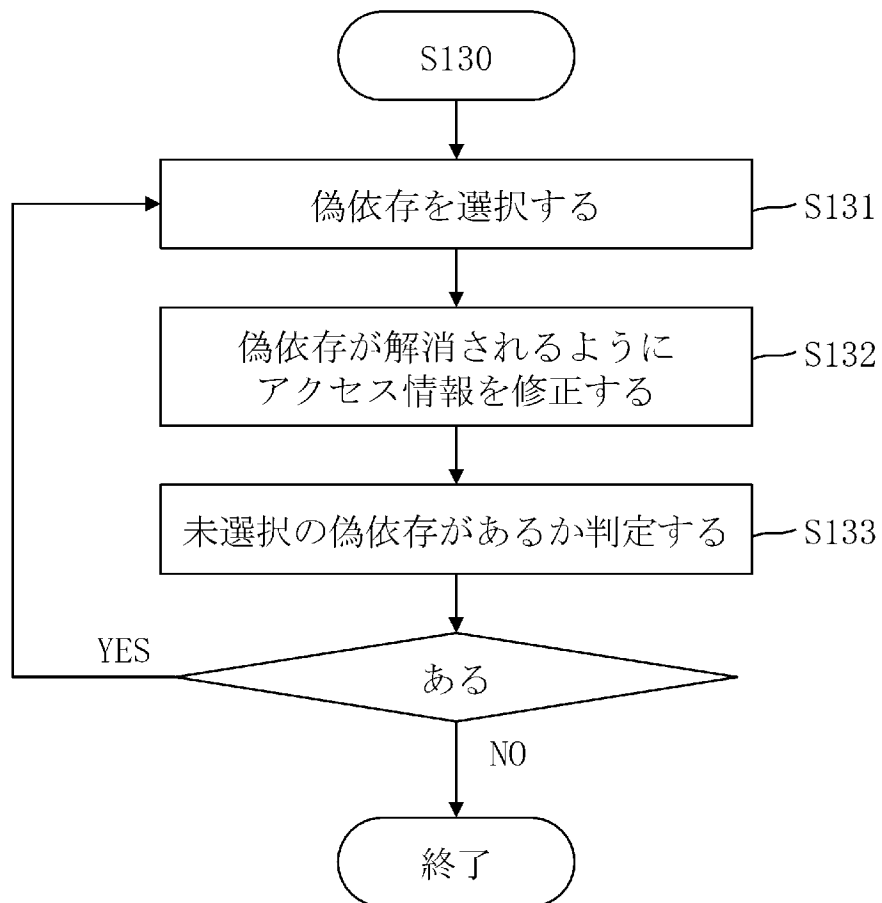
[図13]

図13



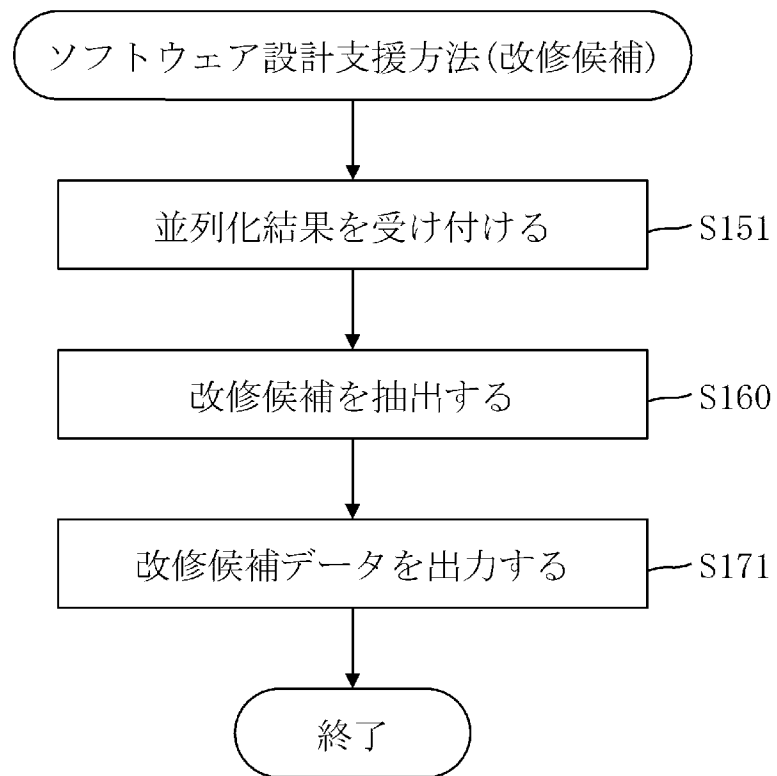
[図14]

図14



[図15]

図15



[図16]

図16

125:改修候補データ

■並列化結果が示す並列実行を実施するために必要十分な
解消すべき偽依存

関数func1と関数func2の間に変数a, cに関するWAR依存が
あります

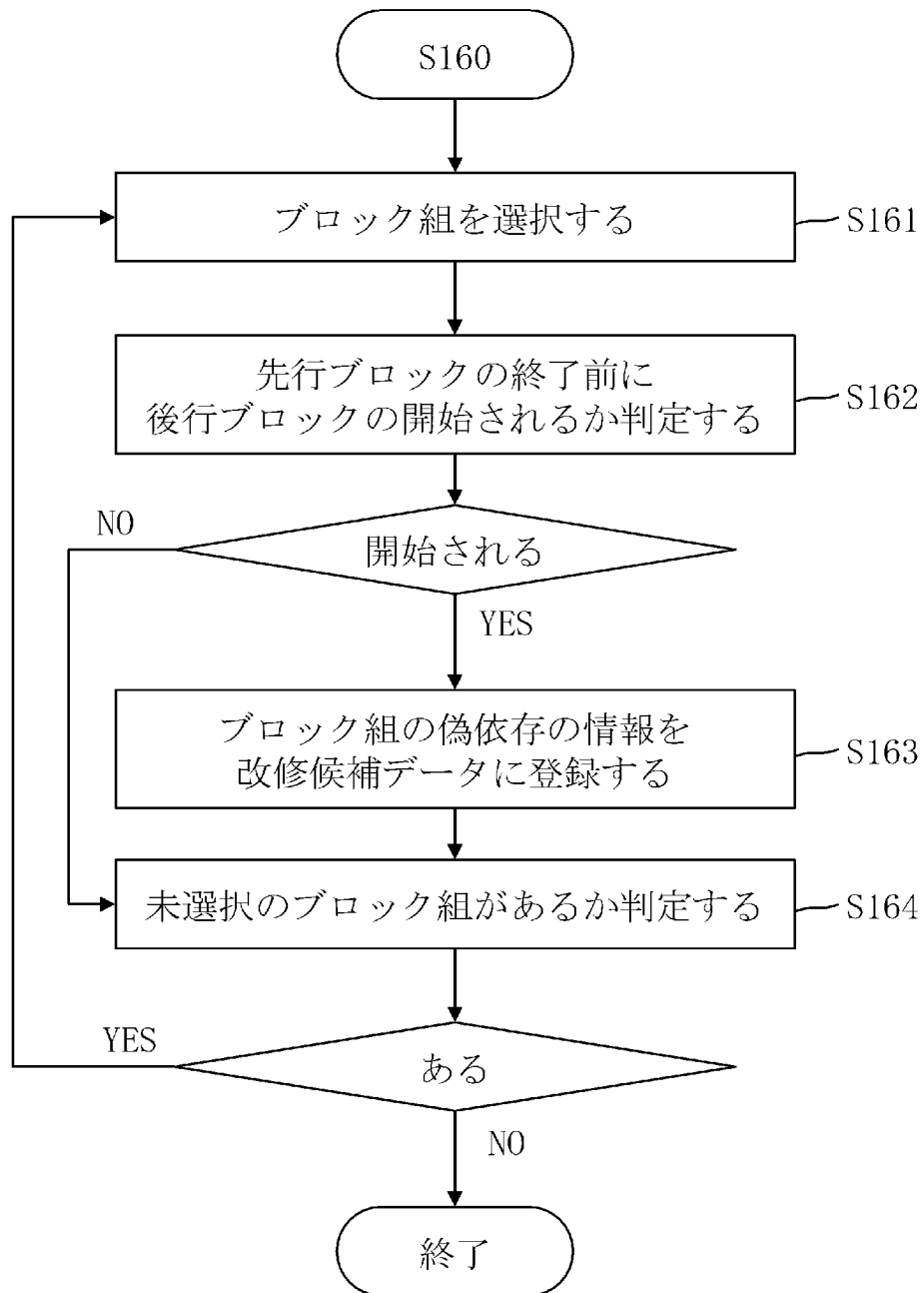
■偽依存の解消方法

ソースコード中の

```
a, func3, r, 23, mysample1.c  
a, func3, w, 22, mysample1.c  
のaをa1に修正してください。  
c, func3, r, 25, mysample1.c  
c, func3, w, 24, mysample1.c  
c, func3, r, 22, mysample1.c  
c, func2, r, 18, mysample1.c  
c, func2, w, 17, mysample1.c  
のcをc1に修正してください。
```

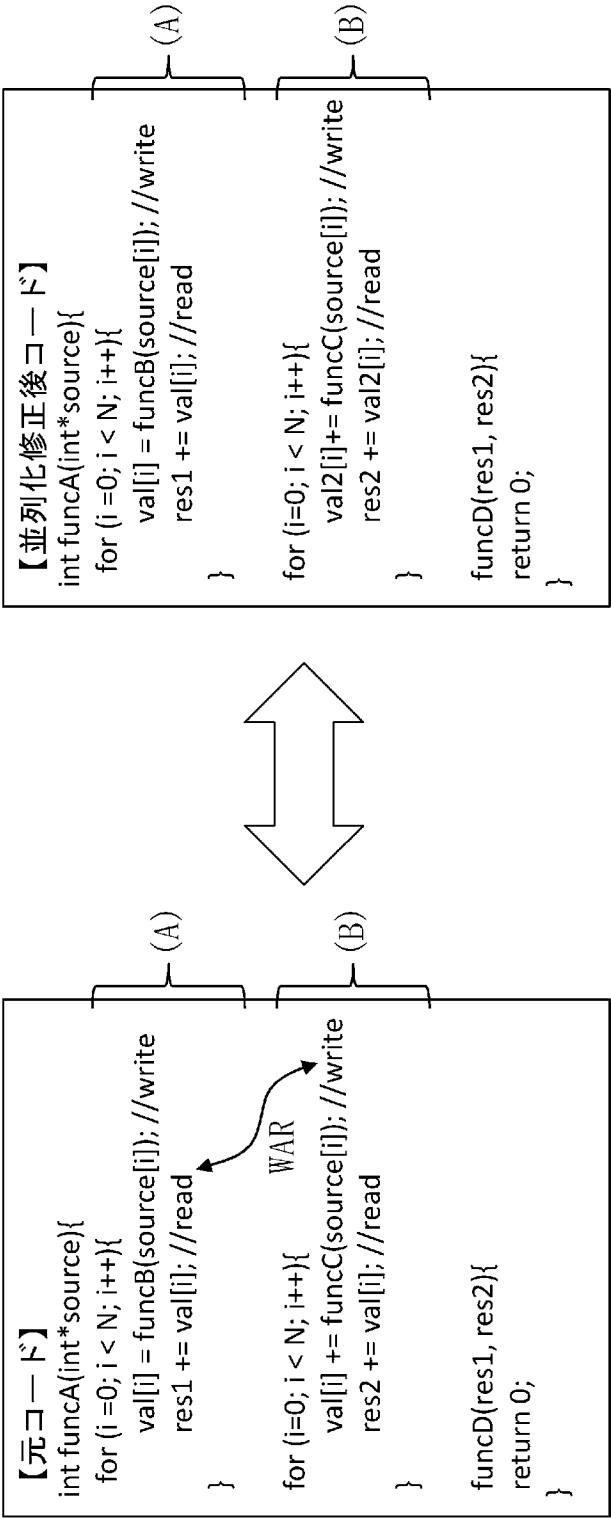
[図17]

図17



[図18]

図18

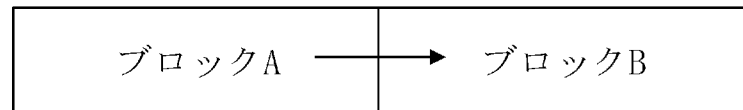


[図19]

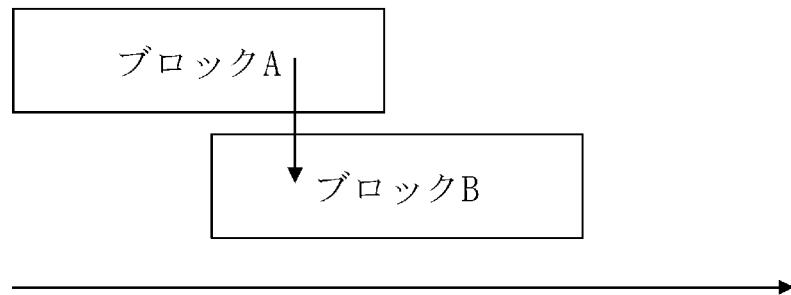
図19

従来技術

並列化前



並列化後

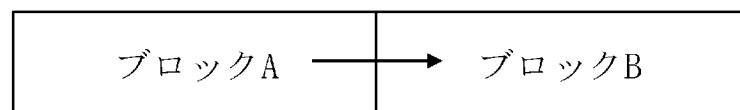


[図20]

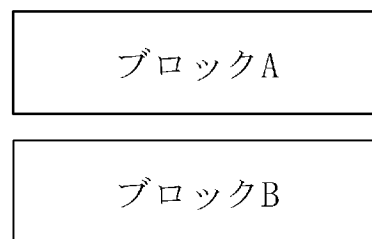
図20

実施の形態1

並列化前



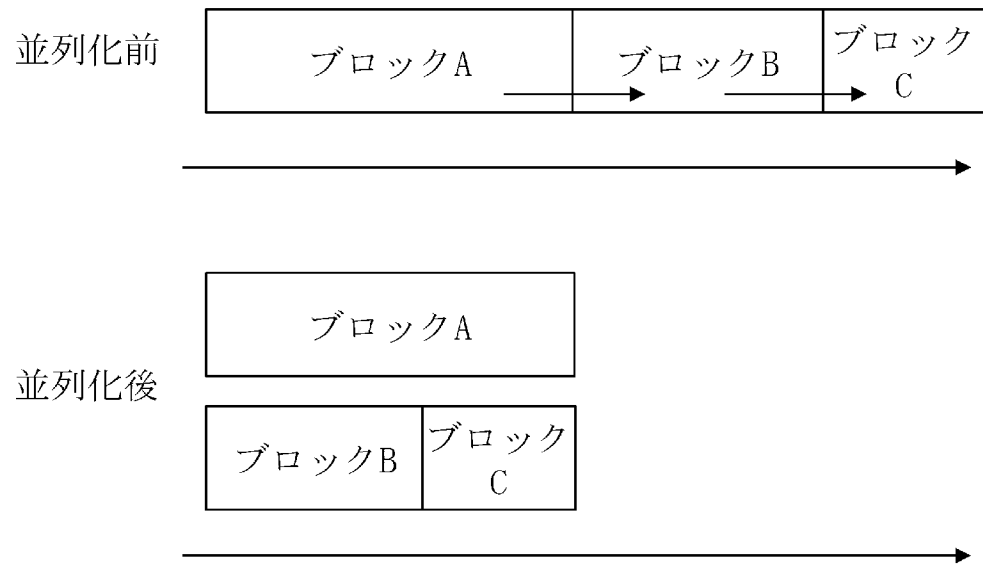
並列化後



[図21]

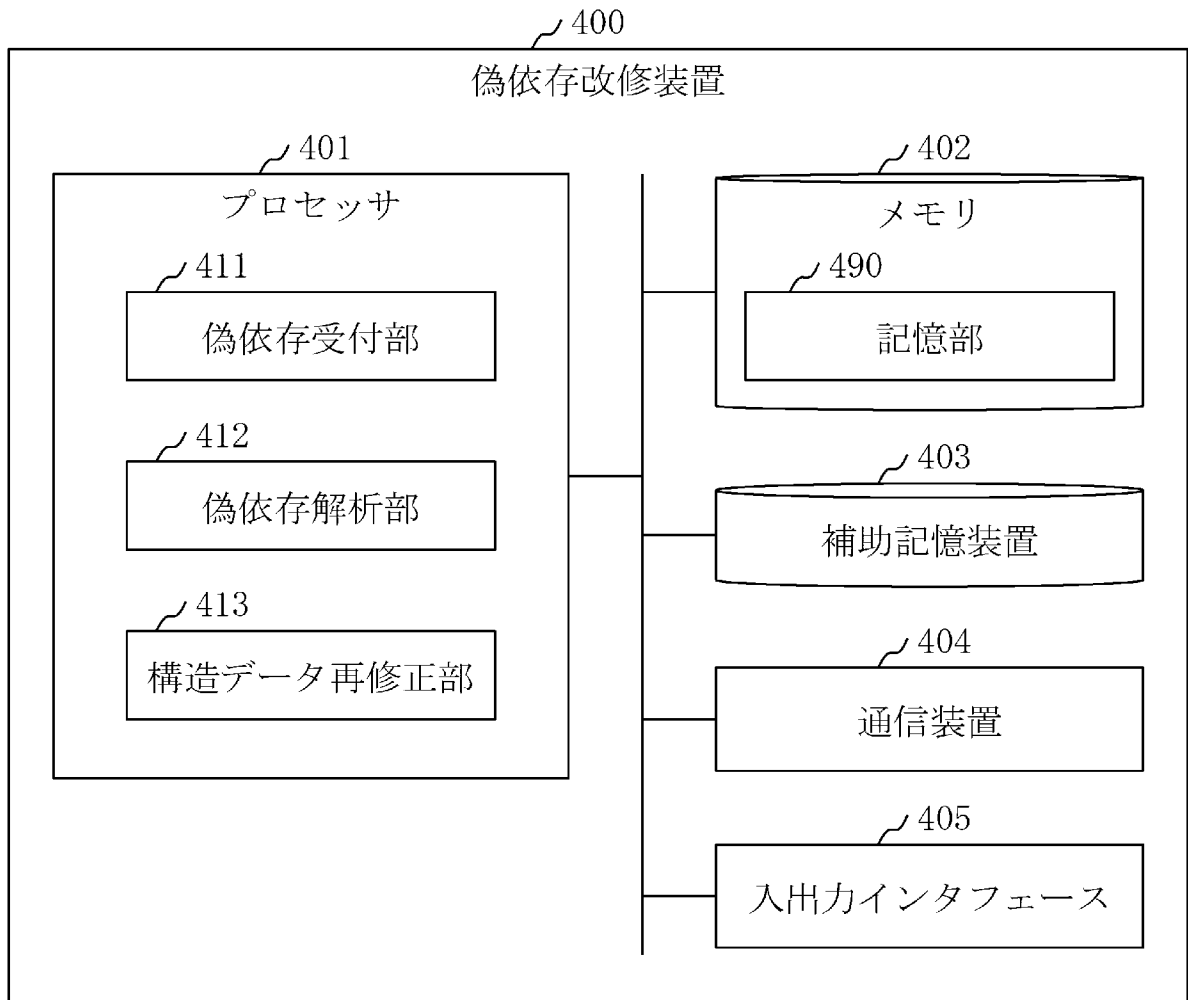
図21

実施の形態1



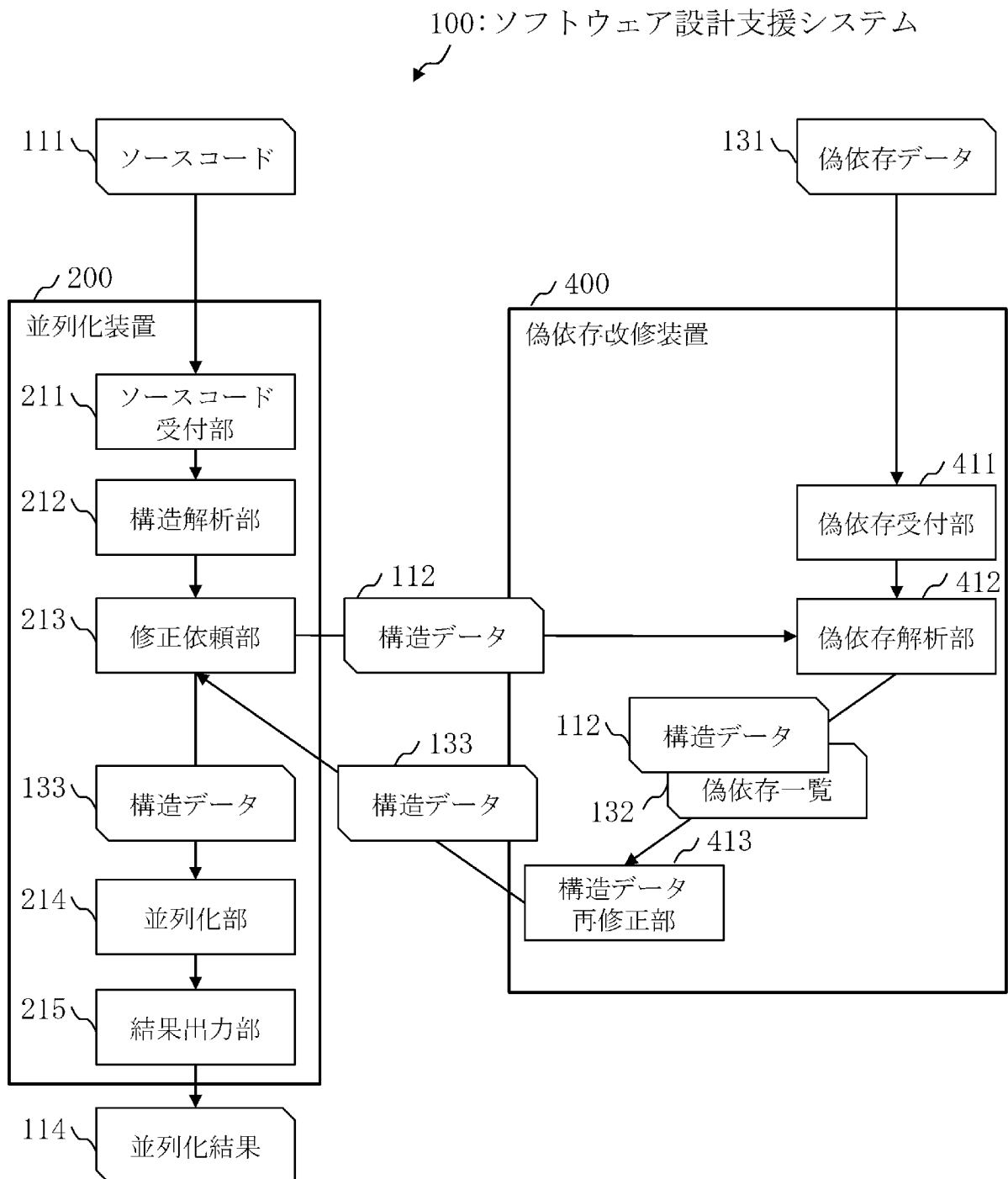
[図22]

図22



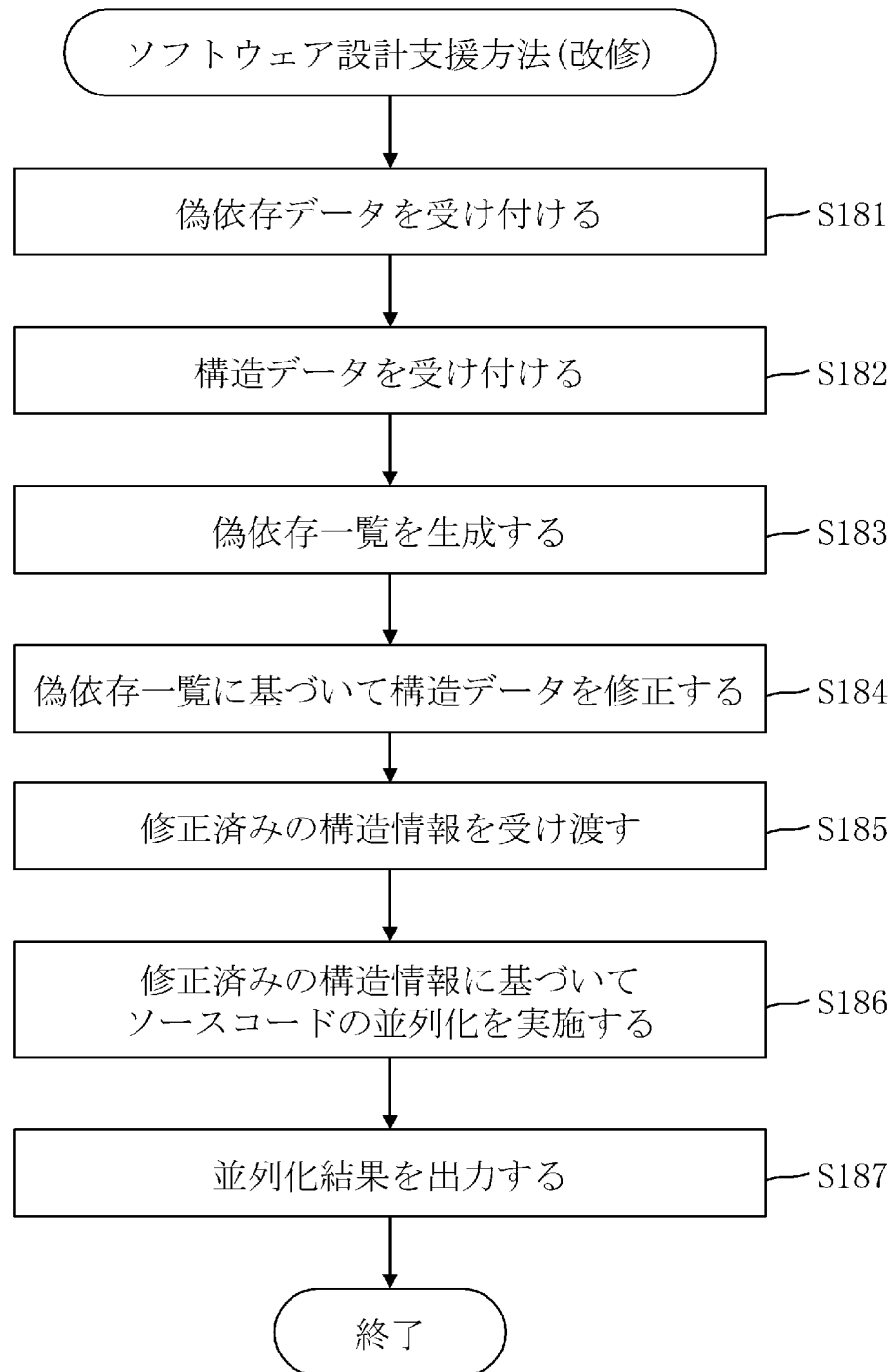
[図23]

図23



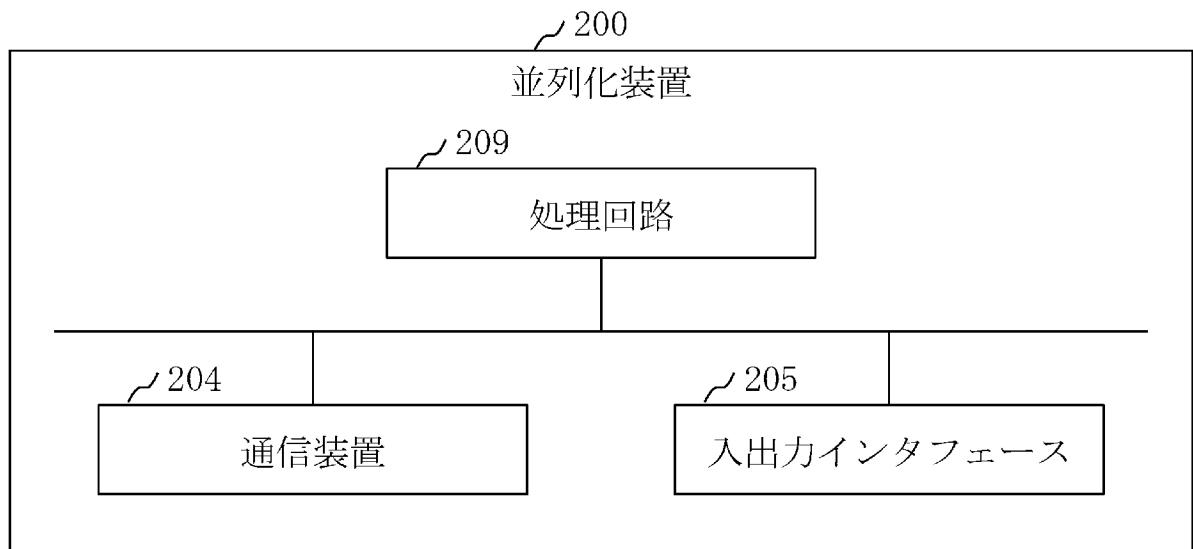
[図24]

図24



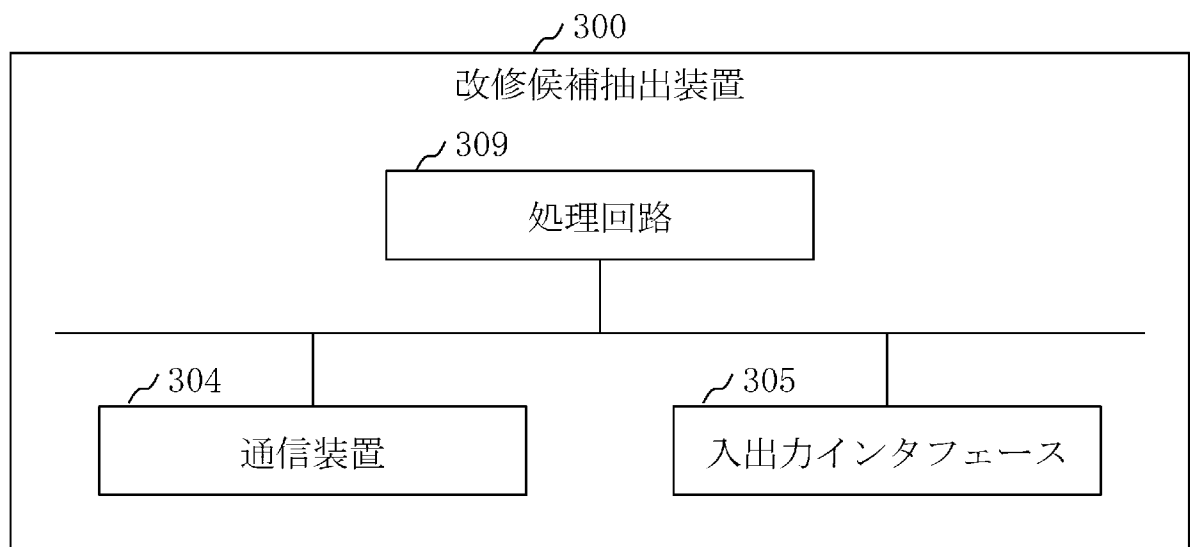
[図25]

図25



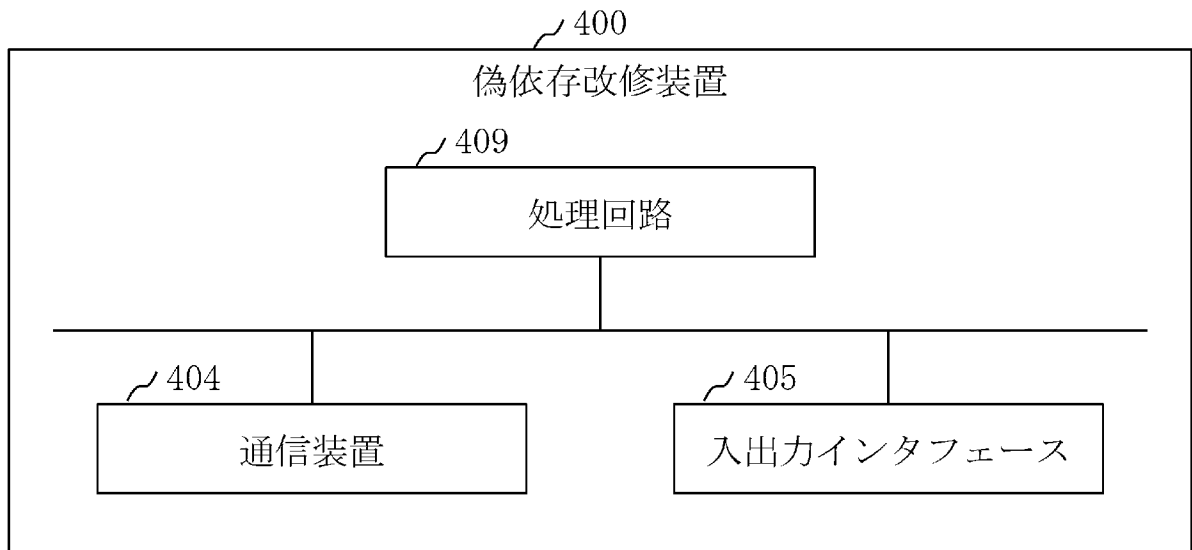
[図26]

図26



[図27]

図27



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2021/019604

A. CLASSIFICATION OF SUBJECT MATTER

G06F 8/41(2018.01)i

FI: G06F8/41 130

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F8/41

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Published examined utility model applications of Japan 1922-1996

Published unexamined utility model applications of Japan 1971-2021

Registered utility model specifications of Japan 1996-2021

Published registered utility model applications of Japan 1994-2021

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	JP 9-106351 A (HITACHI LTD) 22 April 1997 (1997-04-22) paragraphs [0002]-[0023]	1-3, 7, 8 4-6
A	JP 2009-175882 A (FUJITSU LTD) 06 August 2009 (2009-08-06) entire text, all drawings	1-8
A	WO 2012/004972 A1 (PANASONIC CORPORATION) 12 January 2012 (2012-01-12) entire text, all drawings	1-8
A	JP 2016-143378 A (DENSO CORP) 08 August 2016 (2016-08-08) entire text, all drawings	1-8

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

28 July 2021 (28.07.2021)

Date of mailing of the international search report

10 August 2021 (10.08.2021)

Name and mailing address of the ISA/

Japan Patent Office

3-4-3, Kasumigaseki, Chiyoda-ku,

Tokyo 100-8915, Japan

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/JP2021/019604

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
JP 9-106351 A	22 Apr. 1997	(Family: none)	
JP 2009-175882 A	06 Aug. 2009	(Family: none)	
WO 2012/004972 A1	12 Jan. 2012	US 2012/0167036 A1 entire text, all drawings CN 102483701 A entire text, all drawings	
JP 2016-143378 A	08 Aug. 2016	DE 102016201612 A1 entire text, all drawings	

A. 発明の属する分野の分類（国際特許分類（IPC）） G06F 8/41(2018.01)i FI: G06F8/41 130		
B. 調査を行った分野		
調査を行った最小限資料（国際特許分類（IPC）） G06F8/41		
最小限資料以外の資料で調査を行った分野に含まれるもの 日本国実用新案公報 1922-1996年 日本国公開実用新案公報 1971-2021年 日本国実用新案登録公報 1996-2021年 日本国登録実用新案公報 1994-2021年		
国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）		
C. 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求項の番号
X	JP 9-106351 A（株式会社日立製作所）22.04.1997（1997-04-22） 段落[0002]-[0023]	1-3, 7, 8
A		4-6
A	JP 2009-175882 A（富士通株式会社）06.08.2009（2009-08-06） 全文、全図	1-8
A	WO 2012/004972 A1（パナソニック株式会社）12.01.2012（2012-01-12） 全文、全図	1-8
A	JP 2016-143378 A（株式会社デンソー）08.08.2016（2016-08-08） 全文、全図	1-8
☐ C欄の続きにも文献が列挙されている。 ☒ パテントファミリーに関する別紙を参照。		
* 引用文献のカテゴリー “A” 特に関連のある文献ではなく、一般的技術水準を示すもの “E” 国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの “L” 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す） “O” 口頭による開示、使用、展示等に言及する文献 “P” 国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献 “T” 国際出願日又は優先日後に公表された文献であって出願と抵触するものではなく、発明の原理又は理論の理解のために引用するもの “X” 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの “Y” 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの “&” 同一パテントファミリー文献		
国際調査を完了した日 28.07.2021		国際調査報告の発送日 10.08.2021
名称及びあて先 日本国特許庁(ISA/JP) 〒100-8915 日本国 東京都千代田区霞が関三丁目4番3号		権限のある職員（特許庁審査官） 松崎 孝大 5B 4060 電話番号 03-3581-1101 内線 3545

国際調査報告
 パテントファミリーに関する情報

国際出願番号

PCT/JP2021/019604

引用文献			公表日	パテントファミリー文献	公表日
JP	9-106351	A	22.04.1997	(ファミリーなし)	
JP	2009-175882	A	06.08.2009	(ファミリーなし)	
WO	2012/004972	A1	12.01.2012	US 2012/0167036	A1
				全文、全図	
				CN 102483701	A
				全文、全図	
JP	2016-143378	A	08.08.2016	DE 102016201612	A1
				全文、全図	