(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
2012 年 4 月 19 日(19.04.2012)

PCT

(10) 国際公開番号
WO 2012/049728 A1

(51) 国際特許分類:
G06F 9/455 (2006.01) *G06F 11/28* (2006.01)
G06F 9/38 (2006.01)

(21) 国際出願番号: PCT/JP2010/067866

(22) 国際出願日: 2010 年 10 月 12 日(12.10.2010)

(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 (米国を除く全ての指定国について): 富士通株式会社 (FUJITSU LIMITED) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 Kanagawa (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 池 敦(IKE, Atsushi) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 富士通株式会社内 Kanagawa (JP). タシ デビッド(THACH, David) [FR/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 富士通株式会社内 Kanagawa (JP).

(74) 代理人: 渡部 章彦(WATANABE, Akihiko); 〒1130022 東京都文京区千駄木３－５０－１４－３０５ 開明国際特許事務所 Tokyo (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, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

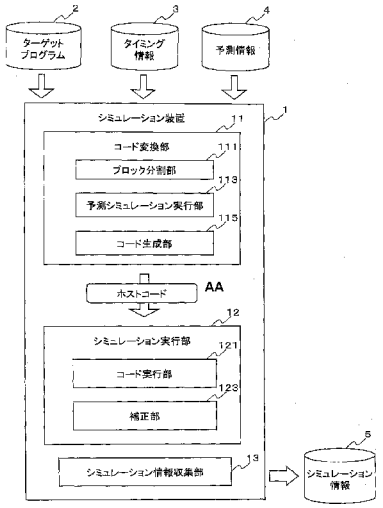
(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ

[続葉有]

(54) **Title**: SIMULATION DEVICE, METHOD, AND PROGRAM

(54) 発明の名称 : シミュレーション装置，方法，およびプログラム

[図1]



1 SIMULATION DEVICE
2 TARGET PROGRAM
3 TIMING INFORMATION
4 PREDICTION INFORMATION
5 SIMULATION INFORMATION
11 CODE CONVERSION UNIT
12 SIMULATION EXECUTION UNIT
13 SIMULATION INFORMATION COLLECTING UNIT
111 BLOCK DIVIDING UNIT
113 PREDICTION SIMULATION EXECUTION UNIT
115 CODE GENERATING UNIT
121 CODE EXECUTION UNIT
123 CORRECTING UNIT
AA HOST CODE

(57) **Abstract**: The present invention makes it possible to execute at high-speed and with high accuracy the performance simulation of a CPU for controlling a pipeline process. A code conversion unit (11) of a simulation device (1), when executing a program of a target CPU, detects an external dependency command affected by the external environment for each divided block, predicts the execution results of the external dependency command, simulates a command execution on the prediction results, and generates on the basis of the simulation results a host code incorporating a code for the performance simulation. A simulation execution unit (12) executes the performance simulation using the host code on the command execution in the prediction results of the program, and if, during execution, the execution results of the external dependency command is different from the settings of the prediction results, the execution time of the command in the prediction results is corrected using the execution time and so forth of the commands which are executed before and after the command. A simulation information collecting unit (13) collects and outputs the performance simulation information.

(57) 要約: パイプライン処理を制御するＣＰＵの性能シミュレーションを，高速かつ高精度で実行できるようにする。シミュレーション装置１のコード変換部１１は，ターゲットＣＰＵのプログラムの実行時に，分割した各ブロックで，外部環境が影響する外部依存命令を検出し，外部依存命令の実行結果を予測し，予測結果での命令実行をシミュレーションし，そのシミュレーション結果をもとに性能シミュレーション用コードを組み込んだホストコードを生成する。シミュレーション実行部１２は，ホストコードを用いてプログラムの予測結果での命令実行について性能シミュレーションを行い，実行中に外部依存命令の実行結果が予測結果の設定と違う場合に，予想結果での命令の実行時間を，その命令前後に実行される命令の実行時間等を用いて補正する。シミュレーション情報収集部１３は，性能シミュレーション情報を収集，出力する。

WO 2012/049728 A1

(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, ML, MR, NE, SN, TD, TG).

添付公開書類:

— 国際調査報告（条約第 21 条(3)）

明　細　書

発明の名称 ： シミュレーション装置，方法，およびプログラム

技術分野

[0001] 本発明は，仮想的にモデル化されたシステム中でのプロセッサの命令実行について，性能または電力のシミュレーション情報を取得する処理技術に関する。

背景技術

[0002] システムが複雑化して，複数プロセッサ（たとえばＣＰＵ）を搭載するマルチコア構成が一般的となっている現況では，各コア（ＣＰＵ）の機能，性能，電力等のシミュレーション処理について，より高い処理速度や処理精度を実現することが要求されている。

[0003] 機能，性能，消費電力のシミュレーションで，評価対象となるターゲットＣＰＵを，ホストＣＰＵで動作させる場合のターゲットＣＰＵの命令コード（ターゲットコード）からホストＣＰＵの命令コード（ホストコード）への変換手法として，インタープリタ方式またはＪＩＴ（Just-in-Time）コンパイラ方式を採用することが知られている。

[0004] ＪＩＴコンパイラ方式によるシミュレーションでは，シミュレーション対象であるターゲットＣＰＵについて，実行中のプログラムに出現するターゲットＣＰＵの命令を，シミュレーションを実行するホストＣＰＵの命令に置き換え，以降では，その置き換えた命令を実行している。そのため，ＪＩＴコンパイラ方式の処理は，インタープリタ方式の処理に比べて高速であり，ＣＰＵの機能シミュレーションでは，特に高速性が求められる場合にＪＩＴコンパイル方式が採用されていた。

[0005] ＪＩＴコンパイラ方式を採用するＣＰＵの性能シミュレーションも提案されている。

[0006] しかし，クロックごとに各ユニットが独立して動作できるようにした上で，次々に命令を投入・並列で実行するパイプライン処理の制御では，処理実

行の度にＣＰＵの内部状態が変化することから，一旦生成したホスト命令を繰り返し利用するというＪＩＴコンパイラ方式の利点が活かせない。

[0007] そのため，一般的に，パイプライン処理やアウトオブオーダ処理を制御するようなＣＰＵに対する性能シミュレーションや電力シミュレーションには，ＪＩＰコンパイラ方式は適さないとされている。

先行技術文献

特許文献

[0008] 非特許文献1：米国特許６，７５１，５８３　Ｂ１

発明の概要

発明が解決しようとする課題

[0009] 上記のように，パイプライン処理やアウトオブオーダ処理を制御するＣＰＵに対する機能，性能，電力のシミュレーションにインタープリタ方式を採用する場合，一般的に，処理速度が非常に遅くなり，現実的な手法として活用できないという問題があった。

[0010] また，昨今の高速なＣＰＵの場合，機能シミュレーションにＪＩＴコンパイラ方式を採用した場合に，なんとか実用的な速度で処理が行える。

[0011] しかし，性能や電力のシミュレーションにＪＩＴコンパイラ方式を採用した場合に，ターゲットＣＰＵではパイプラインの制御により処理の前後の状況は多様であり，内部状態に対応した膨大なシミュレーション用コードの追加とその命令実行が必要なため，処理負担が非常に大きいものとなる。

[0012] さらに，性能シミュレーションで，ターゲットＣＰＵで見込まれる実行遅延に対するタイミングの適応を行うためにも，ホストコードに膨大なシミュレーション用コードを追加する必要がある。例えば，ロード命令（ＬＤ：ｌｏａｄ）のサイクルシミュレーションを実行する際に，この命令によるキャッシュアクセスで，キャッシュミスまたはキャッシュヒットのどちらが生じるか，キャッシュミスの場合に全タイミングを計上し，考慮するべきペナルティサイクルがあるかなどを調べるために，これらの条件記述をホストコー

ドに追加する必要がある。

[0013] しかし，一方で，高い動作性を維持するために，機能コードに追加される性能シミュレーション（サイクルシミュレーション）用のコード量をできる限り抑制する必要がある。

[0014] 本発明は，高速なシミュレーションを行える技術を提供することを目的とする。

課題を解決するための手段

[0015] 本発明の一態様として開示するシミュレーション装置は，パイプライン処理を制御するターゲットプロセッサに対するプログラムの命令実行のシミュレーションを実行するシミュレーション装置であって，１）前記プログラムのコードを所定のブロックに分割し，前記ブロックに含まれる命令のうち，該命令の実行結果が外部環境に依存する外部依存命令の処理の実行結果を予測結果として設定する処理と，前記予測結果を前提とする命令実行の機能シミュレーションを行い，前記ブロックに含まれる命令の実行タイミングを示すタイミング情報を得て，前記機能シミュレーションの結果と前記タイミング情報とをもとに，前記予測結果での外部依存命令の実行時間を算出する処理と，前記機能シミュレーションの結果をもとに，前記予測結果を前提とする命令実行の性能シミュレーションを，前記ターゲットプロセッサを動作させるホストプロセッサに実行させるホストコードを生成する処理とを行うコード変換部と，２）前記ホストプロセッサが前記生成されたホストコードを実行した実行結果において，該ホストコードに含まれる外部依存命令の実行結果が前記予測結果と異なる場合に，該外部依存命令の所定の遅延時間と前記外部依存命令の前後に実行される命令の実行時間とを用いて求めた補正値で，前記予測結果での外部依存命令の実行時間を補正して，前記機能シミュレーションでの該外部依存命令の実行時間とするシミュレーション実行部とを，備える。

[0016] また，本発明の別の一態様として開示するシミュレーション方法は，前記シミュレーション装置が実行する各処理ステップを備えるものである。

[0017] さらに，本発明の別の一態様として開示するシミュレーションプログラムは，コンピュータに，前記シミュレーション方法を実行させるためのものである。

発明の効果

[0018] 開示されたシミュレーション装置によれば，高速にシミュレーションを行うことが可能となる。

図面の簡単な説明

[0019] [図1]本発明の一実施態様として開示するシミュレーション装置の構成例を示す図である。

[図2]ブロックに含まれる命令の例を示す図である。

[図3]タイミング情報の例を示す図である。

[図4]図２に示す命令実行のタイミング例を示す図である。

[図5]サイクルシミュレーション用コードが組み込まれる例を示す図である。

[図6]シミュレーション装置の補正部の処理動作を示す図である。

[図7]シミュレーション装置の補正部のＬＤ命令の実行結果に対する補正例を示す図である。

[図8]シミュレーション装置の補正部のＬＤ命令の実行結果に対する補正例を示す図である。

[図9]補正部１２３のＬＤ命令の実行結果に対する補正例を示す図である。

[図10]シミュレーション装置のコード変換部の概要処理フロー図である。

[図11]シミュレーション装置のシミュレーション実行部の概要処理フロー図である。

[図12]外部依存命令の一例として，ロード（ｌｄ）命令の処理についての予測結果の判定および補正の処理フロー例を示す図である。

[図13]別の実施形態におけるシミュレーション装置の構成例を示す図である。

[図14]シミュレーション装置の電力シミュレーション情報生成部の実施例を示す図である。

発明を実施するための形態

[0020] 図１は，本発明の一実施態様として開示するシミュレーション装置の構成例を示す図である。

[0021] シミュレーション装置１は，パイプライン処理を制御するターゲットＣＰＵにおける命令実行の性能シミュレーションを実行する装置である。

[0022] ターゲットＣＰＵは，シミュレーションの対象となるＣＰＵの制御モデルである。シミュレーション装置１は，ターゲットＣＰＵの命令実行の性能シミュレーションとして各命令のサイクルシミュレーション情報を出力する。

[0023] ここで，ターゲットＣＰＵは，例えばＡＲＭアーキテクチャのＣＰＵである。ホストＣＰＵに相当するシミュレーション装置１は，例えばＸ８６アーキテクチャのＣＰＵを搭載するコンピュータである。

[0024] シミュレーション装置１は，コード変換部１１とシミュレーション実行部１２とシミュレーション情報収集部１３とを有する。

[0025] コード変換部１１は，ターゲットＣＰＵのプログラムの実行時に，ターゲットＣＰＵが実行するプログラムのコード（ターゲットコード）から，シミュレーションを実行するホストＣＰＵのコード（ホストコード）を生成する処理部である。

[0026] コード変換部１１は，ブロック分割部１１１，予測シミュレーション実行部１１３，コード生成部１１５を有する。

[0027] ブロック分割部１１１は，シミュレーション装置１に入力されたプログラムのターゲットコードを，所定のブロックに分割する。分割されるブロック単位は，例えば，一般的なベーシックブロック（分岐から次の分岐前までのコード）単位でよく，または，予め定められた任意のコード単位でよい。

[0028] 図２は，ブロックに含まれる命令の例を示す図である。

[0029] 図２に示すように，あるブロックには，ターゲットコードの３つの命令；（１）“ＬＤ　ｒ１，ｒ２”（ロード）；（２）“ＭＵＬＴ　ｒ３，ｒ４，ｒ５（乗算）”；（３）“ＡＤＤ　ｒ２，ｒ５，　ｒ６（加算）”の命令が含まれ，（１）～（３）の順でターゲットＣＰＵのパイプラインに投入さ

れて実行されるとする。各命令のｒ１～ｒ６は，レジスタ（アドレス）を表す。

[0030] 予測シミュレーション実行部１１３は，タイミング情報３と予測情報４とを得て，入力されたブロックをある実行結果を前提とした条件下で実行する性能シミュレーションを行う処理部である。

[0031] タイミング情報３は，ターゲットコードの各命令について，命令実行時の各処理要素（段階）と使用可能なレジスタとの対応を示す情報と，命令のうち外部依存命令ごとに，実行結果に応じた遅延時間を定めるペナルティ時間（ペナルティサイクル数）とを示す情報である。

[0032] 外部依存命令は，外部環境が関係する処理を行う命令，例えば，ロード命令またはストア命令などのように，命令の実行結果がターゲットＣＰＵ外の外部環境に依存するような処理，例えば，命令キャッシュ，データキャッシュ，ＴＬＢ検索などであったり，さらには，分岐予測，コール／リターンのスタックなどの処理を行う命令である。

[0033] 図３は，タイミング情報３の例を示す図である。

[0034] 図３に示すタイミング情報３では，ＬＤ命令について，ソースレジスタｒｓ１（ｒ１）は１番目の処理要素（ｅ１）で，宛先レジスタｒｄ（ｒ２）は２番目の処理要素（ｅ２）で使用可能であることを表す。

[0035] また，ＭＵＬＴ命令では，第１ソースレジスタｒｓ１（ｒ３）は１番目の処理要素（ｅ１），第２ソースレジスタｒｓ２（ｒ４）は２番目の処理要素（ｅ２），宛先レジスタｒｄ（ｒ５）は３番目の処理要素（ｅ３）で，それぞれ使用可能であることを示す。また，ＡＤＤ命令では，第１ソースレジスタｒｓ１（ｒ２），第２ソースレジスタｒｓ２（ｒ５）は１番目の処理要素（ｅ１），宛先レジスタｒｄ（ｒ６）は２番目の処理要素（ｅ２）で使用可能であることを示す。

[0036] 図４は，図２に示すブロックの各命令の実行タイミング例を示す図である。

[0037] 図３に示すタイミング情報３から，パイプラインに各命令が投入されるタ

イミングは，ＬＤ命令の実行開始がタイミングｔとすると，ＭＵＬＴ命令はタイミングｔ＋１，ＡＤＤ命令はタイミングｔ＋２となる。

[0038] ＡＤＤ命令の第１ソースレジスタ（ｒ２）と第２ソースレジスタ（ｒ５）は，ＬＤ命令とＭＵＬＴ命令で使用されているため，ＡＤＤ命令の開始は，ＬＤ命令とＭＵＬＴ命令の実行完了のタイミングｔ＋４以降となり，２サイクル分の待機時間（２サイクル分のストール）が生じる。

[0039] したがって，図４（Ａ）に示すように，図２に示すブロックをシミュレーションした場合に，ＬＤ命令の実行結果がキャッシュヒットであるケースでは，ブロックの実行時間が６サイクルであることがわかる。

[0040] 図４（Ｂ）は，図２に示すブロックのＬＤ命令の実行結果がキャッシュミスである場合のタイミング例を表す。

[0041] ＬＤ命令の結果がキャッシュミスであると，タイミング情報３に，ペナルティとして，再実行に十分と考えられる任意の時間（ここでは６サイクル分）が設定されているため，このペナルティサイクルが遅延時間として追加される。したがって，２番目の処理要素（ｅ２）の実行は，タイミングｔ＋７に遅延する。ＬＤ命令の次に実行されるＭＵＬＴ命令は，遅延の影響を受けずにそのまま実行されるが，ＡＤＤ命令は，ＬＤ命令の実行完了のタイミングｔ＋８以降となり，４サイクル分の待機時間（４サイクル分のストール）が生じる。

[0042] したがって，図４（Ｂ）に示すように，図２に示すブロックの命令実行をシミュレーションした場合に，ＬＤ命令の実行結果がキャッシュミスであるケースでは，実行時間が１０サイクルとなることがわかる。

[0043] 予測情報４は，ターゲットコードの外部依存命令の処理において，生じる確率が高い実行結果（予測結果）を定めた情報である。予測情報には，例えば，

「命令キャッシュ：予測＝ヒット，

データキャッシュ：予測＝ヒット，

ＴＬＢ検索：予測＝ヒット，

分岐予測：予測＝ヒット，

コール／リターン：予測＝ヒット，…

が定められる。

[0044] 予測シミュレーション実行部１１３は，上記の予測情報４をもとに，入力されたブロックに含まれる外部依存命令の予測結果を設定し，タイミング情報３を参照して，設定した予測結果を前提とする場合（予測ケース）の命令を実行して，命令実行の進み具合をシミュレーションする。予測シミュレーション実行部１１３は，シミュレーション結果として，ブロックに含まれる各命令の実行時間（所要サイクル数）を求める。

[0045] コード生成部１１５は，予測シミュレーション実行部１１３のシミュレーション結果をもとに，処理したブロックに対応するホストコードとして，設定された予測ケースにおける命令実行時の性能シミュレーションを行うためのホストコード（性能シミュレーション用ホストコード）を生成する処理部である。

[0046] コード生成部１１５は，ブロックのターゲットコードをもとに，外部依存命令が予測結果である予測ケースの場合の命令実行を行うホストコードを生成し，さらに，各命令の実行時間を加算して，ブロックの処理時間を計算する処理を行うシミュレーション用コードを組み込む。

[0047] 例えば，コード生成部１１５は，データのＬＤ命令の予測結果として“キャッシュヒット”が設定されている処理については，そのブロック内のＬＤ命令によるキャッシュアクセスが“ヒット”である場合の処理実行をシミュレーションして，この予測ケースでの実行時間を求め，ＬＤ命令によるキャッシュアクセスが“ミス”である場合の実行時間は，予測ケースである“ヒット”時の実行時間の加算／減算を用いた補正計算により求める処理を行うホストコードを生成する。

[0048] シミュレーション実行部１２は，コード生成部１１５が生成したホストコードを実行して，プログラム（ターゲットコード）を実行するターゲットＣＰＵの命令実行の機能および性能シミュレーションを行う処理部である。

[0049] シミュレーション実行部１２は，コード実行部１２１，補正部１２３を有する。

[0050] コード実行部１２１は，ホストコードを用いて，プログラム（ターゲットコード）を実行する処理部である。

[0051] 補正部１２３は，プログラムの実行中に，外部依存命令の実行結果が，設定されていた予測結果と異なる場合（予測外ケース）に，その命令の実行時間を，既に求めた予想ケースでの実行時間を補正して求める処理部である。

[0052] 補正部１２３は，外部依存命令に与えられるペナルティ時間，外部依存命令の前後で実行される命令の実行時間，１つ前の命令の遅延時間などを用いて補正を行う。なお，補正処理の詳細は後述する。

[0053] シミュレーション情報収集部１３は，性能シミュレーションの実行結果として，各命令の実行時間を含むログ情報（シミュレーション情報）５を収集する処理部である。

[0054] 以下に，シミュレーション装置１の処理の流れを説明する。

[0055] 〔コード変換処理〕

（１） シミュレーション装置１のコード変換部１１のブロック分割部１１１は，ターゲットプログラム２のターゲットコードを得て記憶部（図１に図示しない）に保持し，保持したターゲットコードを任意のブロックに分割する（図２参照）。

[0056] （２） 予測シミュレーション実行部１１３は，入力されるターゲットプログラム２に関するタイミング情報３，予測情報４を得て記憶部に保存する。

[0057] そして，予測シミュレーション実行部１１３は，予測情報４をもとに，分割されたブロックの外部依存命令のそれぞれについて予測結果を設定する。例えば，予測シミュレーション実行部１１３は，図２に示すブロックの命令のうち，ＬＤ命令のデータキャッシュの予測結果として「ヒット」を設定する。

[0058] （３） 予測シミュレーション実行部１１３は，ブロックのコードを解釈

して，設定された予測結果を前提とする場合の命令実行をシミュレーションする。すなわち，予測シミュレーション実行部１１３は，図４（Ａ）に示すタイミング例の命令実行をシミュレーションすることになる。

[0059] （４） 次に，コード生成部１１５は，予測ケースのシミュレーション結果をもとに，ターゲットコードからホストコードを生成する。さらに，コード生成部１１５は，ターゲットコードから変換したホストコード（機能コードのみ）に，性能シミュレーション（サイクルシミュレーション）を実行するためのサイクルシミュレーション用コードを組み込む。

[0060] 図５（Ａ）は，ターゲットコードから機能シミュレーションのホストコードが生成される例を示す図，図５（Ｂ）は，機能シミュレーションのホストコードに，サイクルシミュレーション用コードが組み込まれる例を示す図である。

[0061] 図５（Ａ）に示すように，ターゲットコードＩｎｓｔ＿Ａは，ホストコードＨｏｓｔ＿Ｉｎｓｔ＿Ａ０＿ｆｕｎｃ，Ｈｏｓｔ＿Ｉｎｓｔ＿Ａ１＿ｆｕｎｃに変換され，ターゲットコードＩｎｓｔ＿Ｂは，ホストコードＨｏｓｔ＿Ｉｎｓｔ＿Ｂ０＿ｆｕｎｃ，Ｈｏｓｔ＿Ｉｎｓｔ＿Ｂ１＿ｆｕｎｃ，Ｈｏｓｔ＿Ｉｎｓｔ＿Ｂ２＿ｆｕｎｃ，…に変換されて，機能コードのみのホストコードが生成される。

[0062] さらに，機能コードのみのホストコードに，ターゲットコードＩｎｓｔ＿Ａのサイクルシミュレーション用コードＨｏｓｔ＿Ｉｎｓｔ＿Ａ２＿ｃｙｃｌｅ，Ｈｏｓｔ＿Ｉｎｓｔ＿Ａ３＿ｃｙｃｌｅが，ターゲットコードＩｎｓｔ＿Ｂのサイクルシミュレーション用コードＨｏｓｔ＿Ｉｎｓｔ＿Ｂ４＿ｃｙｃｌｅ，Ｈｏｓｔ＿Ｉｎｓｔ＿Ｂ５＿ｃｙｃｌｅが，それぞれ組み込まれる。

[0063] サイクルシミュレーション用コードは，各命令の実行時間（所要サイクル数）を定数化し，各命令の実行時間を合計してブロックの処理時間を求めるコードである。これにより，ブロック実行中の進み具合を示す情報を得ることができる。

[0064] ここで，ホストコードのうち，機能コード，外部依存命令以外の命令についてのサイクルシミュレーション用コードは既知のコードを使用して実施できるので，具体例の説明を省略する。外部依存命令についてのサイクルシミュレーション用コードは，補正処理を呼び出すヘルパー関数として用意される。ヘルパー関数については後述する。

[0065] 〔シミュレーション処理〕

（１） シミュレーション実行部１２のコード実行部１２１は，コード変換部１１が生成したホストコードを用いて，ターゲットプログラム２の性能シミュレーションを行う。

[0066] コード実行部１２１は，ターゲットプログラム２の命令実行をシミュレーションし，各命令の実行時間を得ていく。

[0067] （２） コード実行部１２１は，シミュレーションの実行中に，外部依存命令（例えばＬＤ命令）を検出した場合に，その実行結果が，設定された予測結果と異なっているかを判定し，実行結果が予測結果と違っている場合に，補正部１２３の起動を要求する。例えば，命令「ＬＤ，ｒ１，ｒ２」を検出し，データキャッシュの予測結果（キャッシュヒット）と，実際の実行結果（キャッシュミス）とが異なっていた場合に，補正部１２３が呼び出される。

[0068] （３） 補正部１２３は，呼び出しを受けて起動し，検出された命令「ＬＤ，ｒ１，ｒ２」の実行時間（サイクル数）を補正する。さらに，補正部１２３は，この補正により，次命令の実行タイミングｔ＋ｎも変更する。

[0069] 補正部１２３は，外部依存命令の実行結果が予測結果と異なる度に，命令の実行時間を補正する。ここで，予測ケースでの外部依存命令の実行時間は既に定数化されているため，補正部１２３は，予測外ケースでの外部依存命令の実行時間を，その命令に対するペナルティ時間，前後に実行される命令の実行時間，前に処理された命令の遅延時間等の値を単に加算または減算して計算することができる。

[0070] 図６は，補正部１２３の処理動作を示す図である。

[0071] 補正部１２３は，ヘルパー関数モジュールとして実施される。

[0072] 本実施形態では，例えば，ＬＤ命令のキャッシュの実行結果ごとにシミュレーションを行う従来の関数「ｃａｃｈｅ＿ｌｄ（ａｄｄｒｅｓｓ）」の代わりに，ヘルパー関数「ｃａｃｈｅ＿ｌｄ（ａｄｄｒｅｓｓ，ｒｅｐ＿ｄｅｌａｙ，ｐｒｅ＿ｄｅｌａｙ）」がホストコードに組み込まれることにより，実現している。

[0073] ヘルパー関数の“ｒｅｐ＿ｄｅｌａｙ”は，このロード（ｌｄ）命令の返り値を使用する次の命令の実行までに，ペナルティ時間のうち遅延時間として処理されなかった時間（猶予時間）である。“ｐｒｅ＿ｄｅｌａｙ”は，１つ前の命令から受ける遅延時間である。“－１”は，前の命令に遅延がないことを示す。“ｒｅｐ＿ｄｅｌａｙ”と“ｐｒｅ＿ｄｅｌａｙ”は，性能シミュレーション結果とタイミング情報３との静的分析処理の結果から得られる時間情報である。

[0074] 図６に示す動作例では，補正部１２３は，現タイミングｃｕｒｒｅｎｔ＿ｔｉｍｅと１つ前のｌｄ命令の実行タイミングｐｒｅｌｄ＿ｔｉｍｅとの差が，１つ前のｌｄ命令の遅延時間分ｐｒｅ＿ｄｅｌａｙを超えているときは，１つ前のｌｄ命令の実行タイミングｐｒｅｌｄ＿ｔｉｍｅと現タイミングｃｕｒｒｅｎｔ＿ｔｉｍｅまでの時間で遅延時間ｐｒｅ＿ｄｅｌａｙを調整して有効遅延時間ａｖａｉｌ＿ｄｅｌａｙを求める。

[0075] 次に，補正部１２３は，実行結果がキャッシュミスであれば，予測結果の誤りであり，有効遅延時間ａｖａｉｌ＿ｄｅｌａｙにキャッシュミス時のペナルティ時間ｃａｃｈｅ＿ｍｉｓｓ＿ｌａｔｅｎｃｙを加算して，猶予時間ｒｅｐ＿ｄｅｌａｙをもとに，ＬＤ命令の実行時間を補正する。

[0076] 図７～図９は，補正部１２３のＬＤ命令の実行結果に対する補正例を示す図である。

[0077] 図７は，１つのキャッシュ処理が実行されるケースで１つのキャッシュミスが生じた場合の補正例を説明するための図である。

[0078] 図７の例では，以下の３命令のシミュレーションが実行される。

「ｌｄ　［ｒ１］，ｒ２　：［ｒ１］→ｒ２；
ｍｕｌｔ　ｒ３，ｒ４，ｒ５　：ｒ３＊ｒ４→ｒ５；
ａｄｄ　ｒ２，ｒ５，ｒ６　：ｒ２＋ｒ５→ｒ６」

図７（Ａ）は，予測結果が「キャッシュヒット」の場合の命令実行タイミングのチャート例を示す図である。この予測ケースにおいて，３番目に実行されるａｄｄ命令に，２サイクルストールが生じている。

[0079] 図７（Ｂ）は，予測結果と異なる「キャッシュミス」の場合の命令実行タイミングのチャート例を示す図である。この予測ミスのケースでは，ｌｄ命令の実行結果がキャッシュミスであると，ペナルティサイクル（６サイクル）分の遅延が生じる。そのため，ｍｕｌｔ命令は，遅延の影響を受けずに実行されるが，ａｄｄ命令の実行は，ｌｄ命令の完了を待つため，４サイクル分遅延することになる。

[0080] 図７（Ｃ）は，補正部１２３による補正後の命令実行タイミングチャートの例を示す図である。

[0081] 補正部１２３は，ｌｄ命令の実行結果がキャッシュミスであるので（予測結果のミス），残りの実行時間（２－１＝１サイクル）に所定のキャッシュミス時のペナルティ時間（６サイクル）を加算して有効遅延時間（７サイクル）とする。有効遅延時間は，最大の遅延時間となる。

[0082] さらに，補正部１２３は，次のｍｕｌｔ命令の実行時間（３サイクル）を得て，次命令の実行時間が遅延時間を超過しないと判定して，有効遅延時間から次命令の実行時間を差し引いた時間（７－３＝４サイクル）を，ｌｄ命令の遅延が生じた実行時間（遅延時間）とする。

[0083] また，補正部１２３は，有効遅延時間から上記の遅延時間を差し引いた時間（３サイクル）を猶予時間とする。猶予時間は，ペナルティとしての遅延が猶予された時間である。

[0084] 補正部１２３は，ヘルパー関数ｃａｃｈｅ＿ｌｄ（ａｄｄｒ，ｒｅｐ＿ｄｅｌａｙ，ｐｒｅ＿ｄｅｌａｙ）で，猶予時間ｒｅｐ＿ｄｅｌａｙ＝３，前命令の遅延時間ｐｒｅ＿ｄｅｌａｙ＝－１（遅延なし）を返す。

[0085] この補正により，ｌｄ命令の実行時間は，実行された時間と遅延時間を加算した実行時間（１＋４＝５サイクル）となり，実行完了のタイミングt_1から，後続のｍｕｌｔ命令，ａｄｄ命令の実行時間が計算される。

[0086] すなわち，補正したｌｄ命令の実行時間（５サイクル）に，予測シミュレーション実行部１１３の処理結果（予測結果による予測シミュレーションの結果）で求められていたｍｕｌｔ命令とａｄｄ命令の各々の実行時間（３サイクル，３サイクル）を単純に加算するだけで，このブロックの実行時間（サイクル数）を得ることができる。

[0087] よって，実行結果が予測と異なる命令の実行時間のみを加算または減算による補正処理を行って，その他の命令については，予測結果にもとづくシミュレーション時に求められた実行時間を加算するだけで，高精度に，キャッシュミス時のシミュレーションの実行サイクル数をも求めることができる。

[0088] 図７（Ｄ）は，シミュレーション装置１の処理と比較するために，従来技術によるキャッシュミス時のサイクル数を単純な加算により求めた場合の誤差の大きさを示す図である。図７（Ｄ）の場合には，ｌｄ命令の遅延時間をそのまま加算しているため，実際には，ｌｄ命令の実行中に実行が完了するｍｕｌｔ命令の実行タイミングのずれによる誤差が生じていることがわかる。

[0089] 図８は，２つのキャッシュ処理が実行されるケースで２つのキャッシュミスが生じた場合の補正例を説明するための図である。

[0090] 図８の例では，以下の５命令のシミュレーションが実行される。

「ｌｄ　［ｒ１］，ｒ２　：［ｒ１］→ｒ２；
　ｌｄ　［ｒ３］，ｒ４　：［ｒ３］→ｒ４；
　ｍｕｌｔ　ｒ５，ｒ６，ｒ７　：ｒ５＊ｒ６→ｒ７；
　ａｄｄ　ｒ２，ｒ４，ｒ２　：ｒ２＋ｒ４→ｒ２；
　ａｄｄ　ｒ２，ｒ７，ｒ２　：ｒ２＋ｒ７→ｒ２」

図８（Ａ）は，２つのキャッシュ処理での予測結果が「キャッシュヒット」の場合の命令実行タイミングのチャート例を示す図である。この予測ケー

スでは，２つのｌｄ命令が，２サイクル分（通常の１サイクル＋付加した１サイクル）をあけて実行されるものとする。

[0091] 図８（Ｂ）は，２つのキャッシュ処理の両方が予測結果と異なる「キャッシュミス」の場合の命令実行タイミングのチャート例を示す図である。この予測ミスのケースでは，２つのｌｄ命令のそれぞれでキャッシュミスがあり，ペナルティサイクル（６サイクル）分の遅延が生じる。しかし，２つのｌｄ命令の遅延時間は重なる時間があり，ｍｕｌｔ命令も，遅延の影響を受けずに実行され，２つのａｄｄ命令の実行が２つめのｌｄ命令の完了まで遅延することになる。

[0092] 図８（Ｃ）は，補正部１２３による補正後の命令実行タイミングチャートの例を示す図である。

[0093] 補正部１２３は，図７を用いて説明したように，タイミングt_0において，１つめのｌｄ命令の遅延時間を補正し，ヘルパー関数ｃａｃｈｅ＿ｌｄ（ａｄｄｒ，３，－１）を返す。

[0094] 次に，現タイミングt_1において，補正部１２３は，２つめのｌｄ命令の実行結果がキャッシュミスであるので（予測結果のミス），このｌｄ命令の残りの実行時間にペナルティサイクル（６）を追加して有効遅延時間（１＋６＝７サイクル）とする。

[0095] 補正部１２３は，有効遅延時間から，現タイミングt_1までに消費した遅延時間（＜現タイミングt_1－前命令の実行タイミングt_0＞－設定された間隔）を差し引いて，現タイミングt_1から超過した有効遅延時間を求め（７－（６－２）＝３サイクル），この超過した有効遅延時間を，２つめのｌｄ命令の実行時間とする。

[0096] さらに，補正部１２３は，超過した有効遅延時間から本来の実行時間を差し引いて（３－１＝２サイクル），前命令の遅延時間とする。

[0097] また，補正部１２３は，有効遅延時間から，現タイミングt_1までに消費した遅延時間と現タイミングt_1で超過した有効遅延時間との合計を差し引いて（７－（３＋３）＝１サイクル），猶予時間とする。

[0098] 補正部１２３は，タイミングt_1において，２つめのｌｄ命令の遅延時間を補正した後，ヘルパー関数ｃａｃｈｅ＿ｌｄ（ａｄｄｒ，２，１）を返す。

[0099] この補正により，現タイミングt_1に補正値（３サイクル）を付加したタイミングがｌｄ命令の実行完了のタイミングとなり，そのタイミングから，以降のｍｕｌｔ命令，ａｄｄ命令の実行時間が加算されていくことになる。

[0100] 図８（Ｄ）は，シミュレーション装置１の処理と比較するために，従来技術によるキャッシュミス時のサイクル数を単純な加算により求めた場合の誤差の大きさを示す図である。図８（Ｄ）の場合には，２つのｌｄ命令それぞれに与えられたペナルティにもとづく遅延時間をそのまま加算しているため，大きな誤差（９サイクル）が生じていることがわかる。図８（Ｃ）に示す補正部１２３の処理でも，図８（Ｂ）に示すように正しくシミュレーションされる場合に比べて誤差（１サイクル）があるが，従来手法に比べて，非常に高精度に求められることがわかる。

[0101] 図９は，２つのキャッシュ処理が実行されるケースで１つのキャッシュミスが生じた場合の補正例を説明するための図である。図９の例では，図８で示す説明例と同様の５つの命令のシミュレーションが実行される。

[0102] 図９（Ａ）は，２つのキャッシュ処理での予測結果が「キャッシュヒット」の場合の命令実行タイミングのチャート例を示す図である。この予測ケースでは，図８（Ａ）の場合と同様に，２つのｌｄ命令が，２サイクル分（通常の１サイクル＋付加した１サイクル）をあけて実行されるものとする。

[0103] 図９（Ｂ）は，１つめのｌｄ命令が予測結果と異なる「キャッシュミス」となり，２つめのｌｄ命令の結果が予測結果（キャッシュヒット）である場合の命令実行タイミングのチャート例を示す図である。この予測ミスのケースでは，２つのｌｄ命令のそれぞれにペナルティサイクル（６サイクル）分の遅延が生じる。しかし，２つのｌｄ命令の遅延時間は重なる時間があり，ｍｕｌｔ命令も，遅延の影響を受けずに実行され，２つのａｄｄ命令の実行が２つめのｌｄ命令の完了まで遅延することになる。

[0104] 図９（Ｃ）は，補正部１２３による補正後の命令実行タイミングチャート

の例を示す図である。

[0105] 補正部１２３は，図７を用いて説明したように，タイミングt_0において，１つめのｌｄ命令の遅延時間を補正し，ヘルパー関数ｃａｃｈｅ＿ｌｄ（ａｄｄｒ，３，－１）を返す。

[0106] 次に，現タイミングt_1において，補正部１２３は，２つめのｌｄ命令の実行結果がキャッシュヒットであるので（予測結果），このｌｄ命令の実行開始から現タイミングt_1までの時間＜t_1-t_0－設定された間隔（６－０－２＝４サイクル）＞が，このｌｄ命令の実行時間（２サイクル）より大きいかを判断する。

[0107] 補正部１２３は，２つめのｌｄ命令の実行開始から現タイミングt_1までの時間が，このｌｄ命令の実行時間（２サイクル）より大きいので，現タイミングt_1を，次のｍｕｌｔ命令の実行タイミングとする。

[0108] そして，補正部１２３は，２つめのｌｄ命令の実行完了から現タイミングｔ１までの時間を（２サイクル），次の命令に対する遅延時間として扱い，前命令の遅延時間ｐｒｅ＿ｄｅｌａｙ＝２とする。また，補正部１２３は，１つめのｌｄ命令の有効遅延時間から，現タイミングt_1までに消費した遅延時間と現タイミングt_1で超過した有効遅延時間との合計を差し引いて（７－（６＋０）＝１サイクル），猶予時間ｒｅｐ＿ｄｅｌａｙ＝１とし，ヘルパー関数ｃａｃｈｅ＿ｌｄ（ａｄｄｒ，１，２）を返す。

[0109] 図９（Ｄ）は，シミュレーション装置１の処理と比較するために，従来技術によるキャッシュミス時のサイクル数を単純な加算により求めた場合の誤差の大きさを示す図である。図９（Ｄ）の場合には，１つめのｌｄ命令のペナルティによる遅延時間をそのまま加算しているため誤差が生じていることがわかる。

[0110] 図１０は，シミュレーション装置１のコード変換部１１の概要処理フロー図である。

[0111] シミュレーション装置１において，コード変換部１１のブロック分割部１１１は，ターゲットプログラムのコード（ターゲットオード）を所定の単位

のブロックに分割して入力する（ステップＳ１）。

[0112] 予測シミュレーション実行部１１３は，ブロックの命令を分析して，外部依存命令を検出して（ステップＳ２），検出した全ての命令について，予測情報４をもとに，確率が高い実行結果を予測ケースとして決定する（ステップＳ３）。

[0113] さらに，予測シミュレーション実行部１１３は，タイミング情報３を参照して，ブロックの各命令について予測結果として設定された実行結果を前提とする性能シミュレーションを実行する（ステップＳ４）。

[0114] コード生成部１１５は，シミュレーション結果をもとに，シミュレーション実行部１２が実行する性能シミュレーション用ホストコードを生成する（ステップＳ５）。

[0115] 以上のステップＳ１～Ｓ５の処理により，設定された実行結果の場合（予測ケース）での機能コードに，ターゲットＣＰＵの性能をシミュレーションするコードが組み込まれたホストコードが出力される。

[0116] 図１１は，シミュレーション装置１のシミュレーション実行部１２の概要処理フロー図である。

[0117] シミュレーション装置１において，シミュレーション実行部１２のコード実行部１２１は，コード生成部１１５が生成したホストコードを実行し，性能シミュレーションを行う（ステップＳ１０）。コード実行部１２１は，実行中に外部依存命令を検出すると（ステップＳ１１），その命令の実行結果が予測結果として設定されたものと同じであるかを判定する（ステップＳ１２）。外部依存命令の実行結果が設定された予測結果と同じではない場合にのみ（ステップＳ１２のＮ），補正部１２３が呼び出され，補正部１２３は，その外部依存命令の実行時間を補正する（ステップＳ１３）。

[0118] そして，シミュレーション情報収集部１３は，ターゲットプログラムに相当するホストコード全てのシミュレーション処理についてのシミュレーション情報５を出力する（ステップＳ１４）。

[0119] 以上のステップＳ１０～Ｓ１４の処理ステップにより，ターゲットプログ

ラム２を実行するターゲットＣＰＵのシミュレーション情報（サイクルシミュレーション情報）５が出力される。

[0120] 図１２は，外部依存命令の一例として，ロード（ｌｄ）命令の処理についての予測結果の判定および補正の処理フロー例を示す図である。

[0121] コード実行部１２１は，処理中のブロックの命令から，外部依存命令を検出すると，補正部１２３に相当するヘルパー関数を呼び出す（ステップＳ２０）。

[0122] コード実行部１２１は，ｌｄ命令で，キャッシュアクセスが要求されているかを判定し（ステップＳ２１），キャッシュアクセスが要求されていれば（ステップＳ２１のＹ），キャッシュアクセスをシミュレーションする（ステップＳ２２）。キャッシュアクセスの結果が“キャッシュミス”であれば（ステップＳ２３の“ミス”），補正部１２３は，ｌｄ命令の実行時間（サイクル数）の補正を行い（ステップＳ２４），補正された実行時間（サイクル数）を出力する（ステップＳ２５）。

[0123] ステップＳ２１で，キャッシュアクセスが要求されていない場合（ステップＳ２１のＮ），または，要求されたキャッシュアクセスが“キャッシュヒット”であれば（ステップＳ２３の“ヒット”），補正部１２３は，未補正の予測された実行時間（サイクル数）を出力する（ステップＳ２６）。

[0124] 図１３は，別の実施形態におけるシミュレーション装置１の構成例を示す図である。

[0125] シミュレーション装置１は，図１に示す構成例に，さらに，電力シミュレーション情報生成部１５を備える。

[0126] 電力シミュレーション情報生成部１５は，電力情報６を得て，シミュレーション情報収集部１３が出力したシミュレーション情報５をもとに，ブロックの実行時の消費電力を計算し，電力シミュレーション情報７として出力する処理部である。

[0127] 図１４は，電力シミュレーション情報生成部１５を，性能シミュレーション用のホストコードに組み込む関数（電力シミュレーション関数）として実

施した場合の例を示す図である。

[0128] 電力情報６として，ＬＤ命令，ＭＵＬＴ命令，ＡＤＤ命令の１実行当たりの消費電力が，それぞれ４ｕ［Ｗ］，０．５ｕ［Ｗ］，０．３ｕ［Ｗ］と設定されている。

[0129] 電力シミュレーション関数Ｈｏｓｔ＿Ｉｎｓｔ＿Ａ－Ｃ＿ｐｏｗｅｒは，シミュレーションで実行された各命令の実行回数にもとづいて，電力を計算する。

[0130] 次に，シミュレーション装置１のハードウェア構成例を説明する。

[0131] シミュレーション装置１は，演算装置（ＣＰＵ），一時記憶装置（ＤＲＡＭ，フラッシュメモリ等），永続性記憶装置（ＨＤＤ，フラッシュメモリ等），およびネットワークＮとのネットワークインターフェースを有するコンピュータＰＣと，入力装置（キーボード，マウス等）と出力装置（ディスプレイ，プリンタ等）とによって実施することができる。

[0132] また，シミュレーション装置１は，コンピュータＰＣが実行可能なプログラムによって実施することができる。この場合に，シミュレーション装置１が有すべき機能の処理内容を記述したプログラムが提供される。提供されたプログラムをコンピュータＰＣが実行することによって，上記で説明したシミュレーション装置１の処理機能がコンピュータＰＣ上で実現される。

[0133] なお，コンピュータＰＣは，可搬型記録媒体から直接プログラムを読み取り，そのプログラムに従った処理を実行することもできる。また，コンピュータＰＣは，サーバコンピュータからプログラムが転送されるごとに，逐次，受け取ったプログラムに従った処理を実行することもできる。

[0134] さらに，このプログラムは，コンピュータＰＣで読み取り可能な記録媒体に記録しておくことができる。

[0135] 以上説明したように，シミュレーション装置１によれば，パイプライン処理を制御するＣＰＵの命令実行の性能シミュレーションを高速に行うことが可能となる。

[0136] シミュレーション装置１は，ＪＩＴコンパイラ方式のように，ターゲット

ＣＰＵのプログラムのコード変換処理において，コード変換部１１が，分割したブロックごとに，外部依存命令の実行結果を予測した場合（予測ケース）での実行について機能シミュレーションを行い，タイミング情報３をもとに各命令の実行時間を定数化しておく。

[0137] シミュレーション装置１のコード変換部１１では，予測ケース以外の実行についての性能シミュレーション情報（サイクルシミュレーション情報）を得るためのシミュレーション実行コードをホストコードに組み込む代わりに，予測ケースでの命令の実行時間の補正処理を行うコードを組み込む。

[0138] また，シミュレーション装置１のシミュレーション実行部１２では，予測ケースの場合の命令実行について性能シミュレーションを行い，予測外ケースでのシミュレーションの代わりに，その命令に対する遅延時間，前後に実行される命令の実行時間等を用いて，予測ケースでの実行時間を補正して，外部依存命令の実行時間を得る。

[0139] そのため，従来に比べて，機能コードに追加するシミュレーション用コード量を少なくすることができ，性能シミュレーション処理の負荷増大を抑制しつつ，高速に行うことが可能となる。

[0140] また，シミュレーション装置１では，予測ケースを前提として機能シミュレーションを行うことにより，１つのケース（予測ケース）における各命令の実行時間を，静的分析処理により得ておき，予測ケース以外のケースにおける命令の実行時間は，予測ケースでの実行時間を，ペナルティ時間，前後に実行される命令の実行時間，前の命令の遅延時間等の加算または減算により補正する。

[0141] シミュレーション装置１によれば，ＣＰＵを含むシステムの性能，電力のシミュレーションを高速に実施することが可能となり，従来では現実レベルでの実行が困難であった，大規模なシステム全体の性能，電力の評価や解析，予測などを容易に行えるようになる。

[0142] さらに，シミュレーション装置１によれば，電力情報と性能シミュレーション情報とをもとに，プログラムの消費電力シミュレーションを，高速かつ

高精度に行うことが可能となる。

[0143] よって，各ケースでの性能シミュレーションを行う必要がなく，性能シミュレーション処理の負荷増大を抑制しつつ，高精度に行うという効果を奏する。

符号の説明

[0144] １　シミュレーション装置

１１　コード変換部

１１１　ブロック分割部

１１３　予測シミュレーション実行部

１１５　コード生成部

１２　シミュレーション実行部

１２１　コード実行部

１２３　補正部

１３　シミュレーション情報収集部

１５　電力シミュレーション情報生成部

２　ターゲットプログラム

３　タイミング情報

４　予測情報

５　シミュレーション情報

６　電力情報

７　電力シミュレーション情報

請求の範囲

[請求項1] パイプライン処理を制御するターゲットプロセッサに対するプログラムの命令実行のシミュレーションを実行するシミュレーション装置であって，

前記プログラムのコードを所定のブロックに分割し，前記ブロックに含まれる命令のうち，該命令の実行結果が外部環境に依存する外部依存命令の処理の実行結果を予測結果として設定する処理と，前記予測結果を前提とする命令実行の機能シミュレーションを行い，前記ブロックに含まれる命令の実行タイミングを示すタイミング情報を得て，前記機能シミュレーションの結果と前記タイミング情報とをもとに，前記予測結果での外部依存命令の実行時間を算出する処理と，前記機能シミュレーションの結果をもとに，前記予測結果を前提とする命令実行の性能シミュレーションを，前記ターゲットプロセッサを動作させるホストプロセッサに実行させるホストコードを生成する処理とを行うコード変換部と，

前記ホストプロセッサが前記生成されたホストコードを実行した実行結果において，該ホストコードに含まれる外部依存命令の実行結果が前記予測結果と異なる場合に，該外部依存命令の所定の遅延時間と前記外部依存命令の前後に実行される命令の実行時間とを用いて求めた補正値で，前記予測結果での外部依存命令の実行時間を補正して，前記機能シミュレーションでの該外部依存命令の実行時間とするシミュレーション実行部とを，備える

ことを特徴とするシミュレーション装置。

[請求項2] 前記シミュレーション実行部は，前記外部依存命令の次に実行される次命令の実行時間が，前記外部依存命令に付加される遅延時間を超えない場合に，前記次命令の実行時間を前記補正値として前記外部依存命令の遅延時間から減算する処理を行う

ことを特徴とする請求項１に記載のシミュレーション装置。

[請求項3] 前記ホストコードの命令セットの各命令の１実行当たりの消費電力量を定めた電力情報を得て，前記電力情報と前記機能シミュレーションの結果とをもとに，前記ブロックの実行時の電力シミュレーション情報を求める電力シミュレーション情報生成部を備える

ことを特徴とする請求項１または請求項２に記載のシミュレーション装置。

[請求項4] パイプライン処理を制御するターゲットプロセッサに対するプログラムの命令実行のシミュレーションを実行するシミュレーション方法であって，

ホストＣＰＵが，

前記プログラムのコードを所定のブロックに分割し，前記ブロックに含まれる命令のうち，該命令の実行結果が外部環境に依存する外部依存命令の処理の実行結果を予測結果として設定する処理過程と，

前記予測結果を前提とする命令実行の機能シミュレーションを行い，前記ブロックに含まれる命令の実行タイミングを示すタイミング情報を得て，前記機能シミュレーション結果と前記タイミング情報とをもとに，前記予測結果での外部依存命令の実行時間を算出する処理過程と，

前記機能シミュレーション結果をもとに，前記予測結果を前提とする命令実行の性能シミュレーションを，前記ターゲットプロセッサを動作させるホストプロセッサに実行させるホストコードを生成する処理過程と，

前記ホストプロセッサが前記生成されたホストコードを実行した実行結果において，該ホストコードに含まれる外部依存命令の実行結果が前記予測結果と異なる場合に，該外部依存命令の所定の遅延時間と前記外部依存命令の前後に実行される命令の実行時間とを用いて求めた補正値で，前記予測結果での外部依存命令の実行時間を補正して，前記機能シミュレーションでの該外部依存命令の実行時間とする処理

過程とを，実行する

ことを特徴とするシミュレーション方法。

[請求項5] パイプライン処理を制御するターゲットプロセッサに対するプログラムの命令実行のシミュレーションを実行させるためのシミュレーションプログラムであって，

ホストＣＰＵに，

前記プログラムのコードを所定のブロックに分割し，前記ブロックに含まれる命令のうち，実行結果が外部環境に依存する外部依存命令の処理の実行結果を予測結果として設定する処理と，

前記予測結果を前提とする命令実行の機能シミュレーションを行い，前記ブロックに含まれる命令の実行タイミングを示すタイミング情報を得て，前記機能シミュレーション結果と前記タイミング情報とをもとに，前記予測結果での外部依存命令の実行時間を算出する処理と，

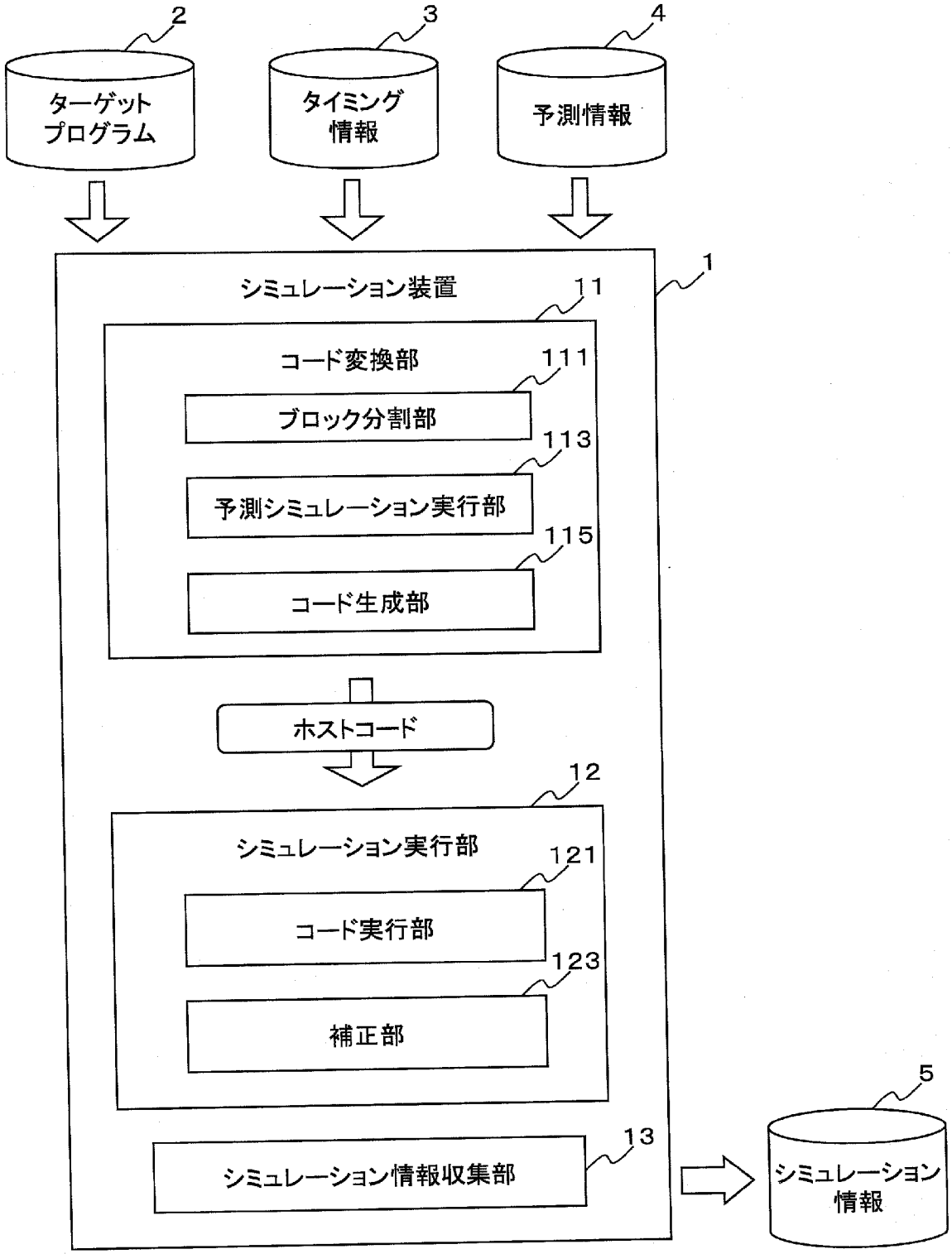
前記機能シミュレーション結果をもとに，前記予測結果を前提とする命令実行の性能シミュレーションを，前記ターゲットプロセッサを動作させるホストプロセッサに実行させるホストコードを生成する処理と，

前記ホストプロセッサが前記生成されたホストコードを実行した実行結果において，該ホストコードに含まれる外部依存命令の実行結果が前記予測結果と異なる場合に，該外部依存命令の所定の遅延時間と前記外部依存命令の前後に実行される命令の実行時間とを用いて求めた補正値で，前記予測結果での外部依存命令の実行時間を補正して，前記機能シミュレーションでの該外部依存命令の実行時間とする処理とを，実行させる

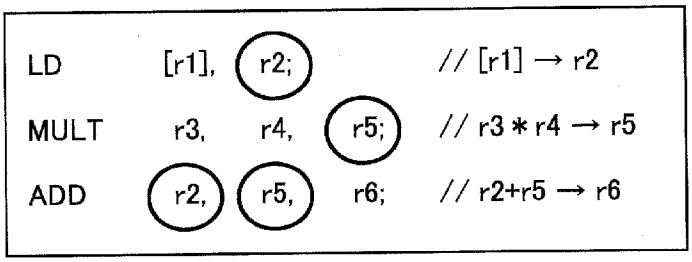
ことを特徴とするシミュレーションプログラム。

WO 2012/049728 PCT/JP2010/067866

[図1]



[図2]

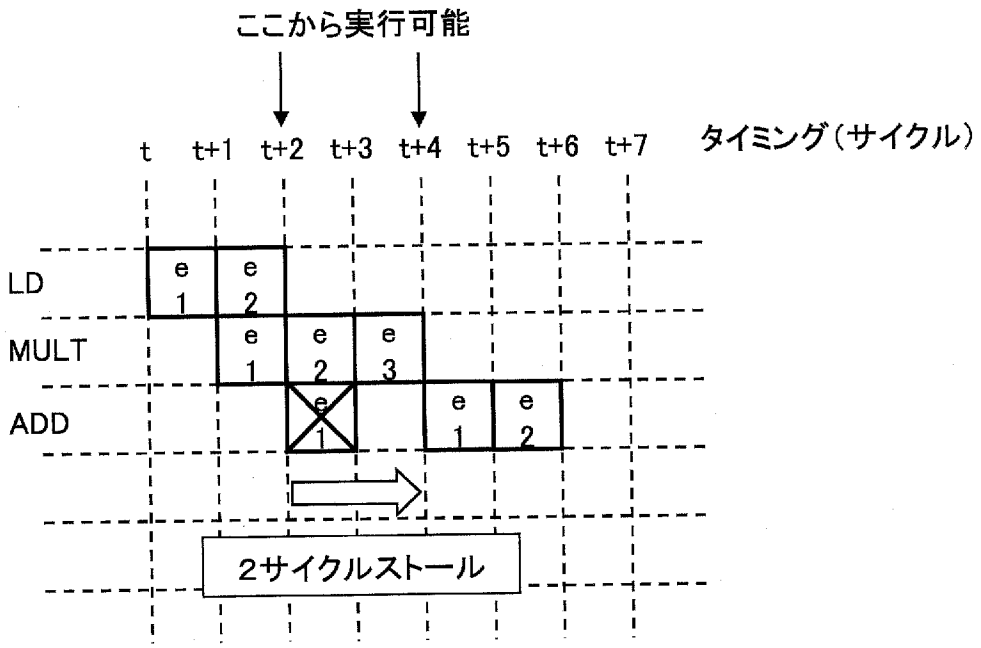


[図3]

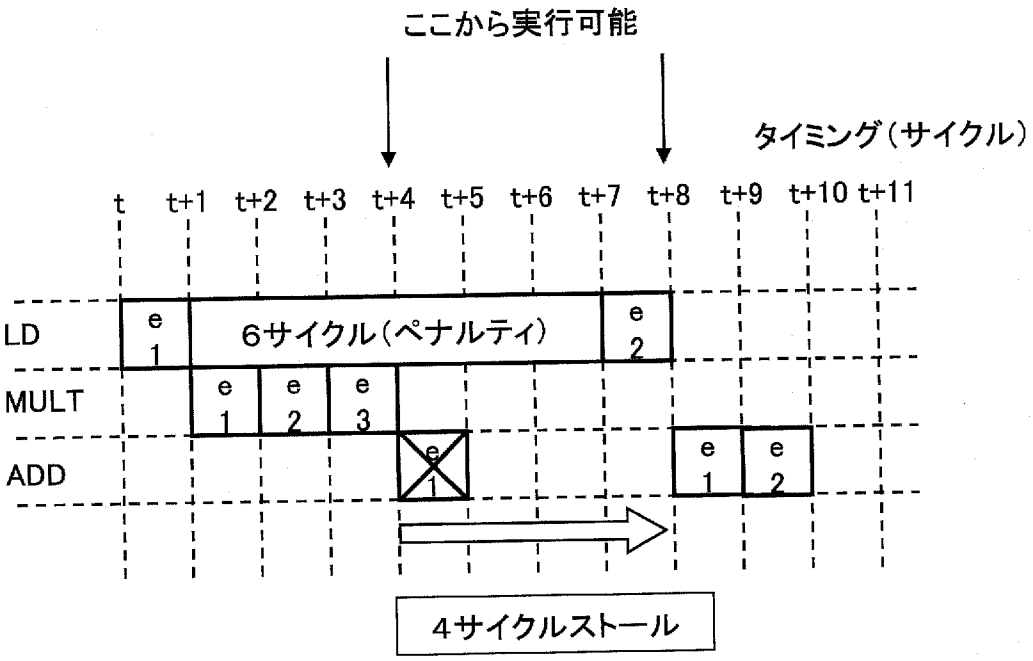
命令	ソースレジスタ	宛先レジスタ	ペナルティ
LD	rs1:e1	rd:e2	キャッシュミス：6
MULT	rs1:e1, rs2:e1	rd:e3	−
ADD	rs1:e1, rs2:e1	rd:e2	−

[図4]

(A)

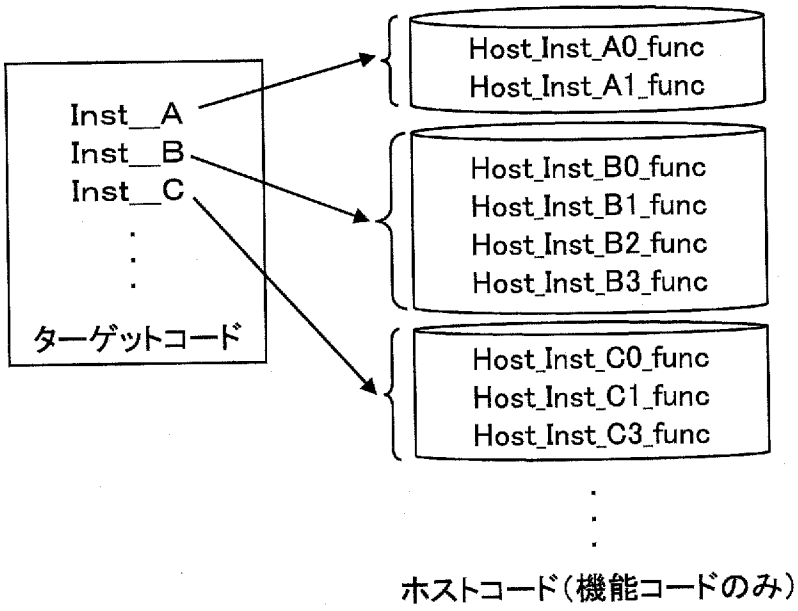

ここから実行可能
t t+1 t+2 t+3 t+4 t+5 t+6 t+7
タイミング(サイクル)
LD
MULT
ADD
e1
e2
e3
2サイクルストール

(B)

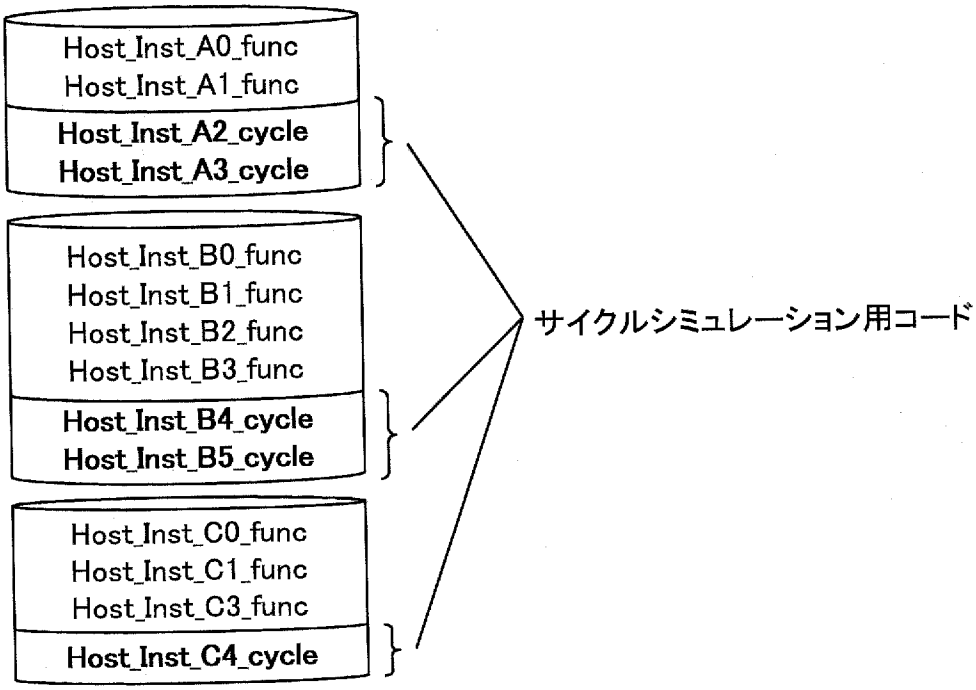

ここから実行可能
タイミング(サイクル)
t t+1 t+2 t+3 t+4 t+5 t+6 t+7 t+8 t+9 t+10 t+11
LD
MULT
ADD
e1
e2
e3
6サイクル(ペナルティ)
4サイクルストール

[図5]

(A)


Inst__A
Inst__B
Inst__C
ターゲットコード
Host_Inst_A0_func
Host_Inst_A1_func
Host_Inst_B0_func
Host_Inst_B1_func
Host_Inst_B2_func
Host_Inst_B3_func
Host_Inst_C0_func
Host_Inst_C1_func
Host_Inst_C3_func
ホストコード(機能コードのみ)

(B)

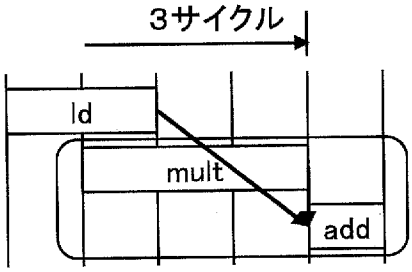

Host_Inst_A0_func
Host_Inst_A1_func
Host_Inst_A2_cycle
Host_Inst_A3_cycle
Host_Inst_B0_func
Host_Inst_B1_func
Host_Inst_B2_func
Host_Inst_B3_func
Host_Inst_B4_cycle
Host_Inst_B5_cycle
Host_Inst_C0_func
Host_Inst_C1_func
Host_Inst_C3_func
Host_Inst_C4_cycle
サイクルシミュレーション用コード

[図6]

```
cache_ld( address, rep_delay, pre_delay ) {
        avail_delay = 0;
        if(pre_delay < current_time - preld_time)
                avail_delay = pre_delay - current_time + preld_time;
        cache_lookup( address );
        if(cache_hit) {
                cache_update_onhit( address );
        } else {
                cache_update_onmiss( address );
                avail_delay += cache_miss_latency
                if(rep_delay < avail_delay)
                        avail_delay -= avail_delay - rep_delay;
        }
        preld_time = current_time;
}
```

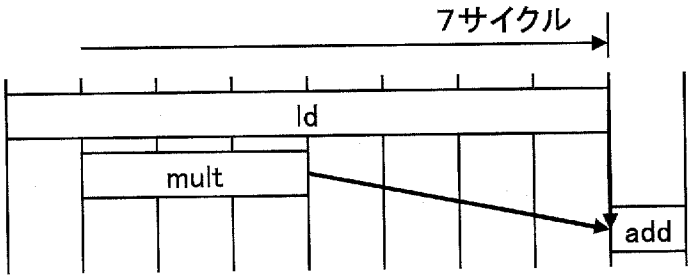
[図7]

(A)

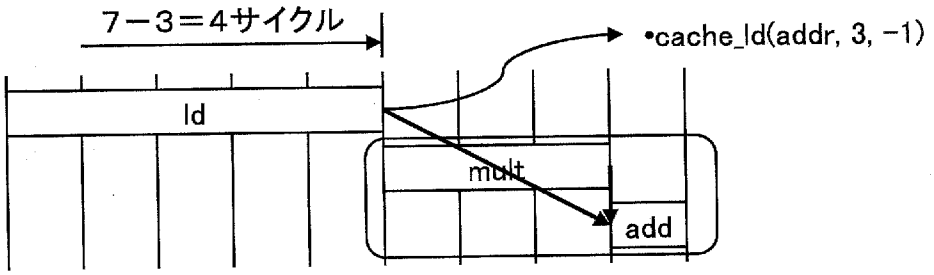



```
ld    [r1], r2     ; [r1]→r2
mult   r3, r4, r5   ; r3*r4→r5
add   r2, r5, r6    ; r2+r5→r6
```

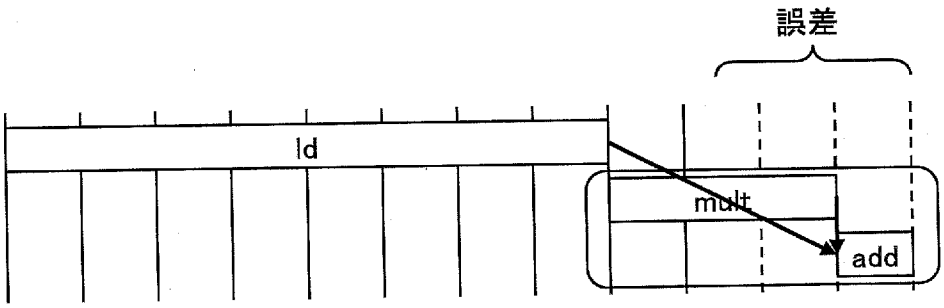
(B)



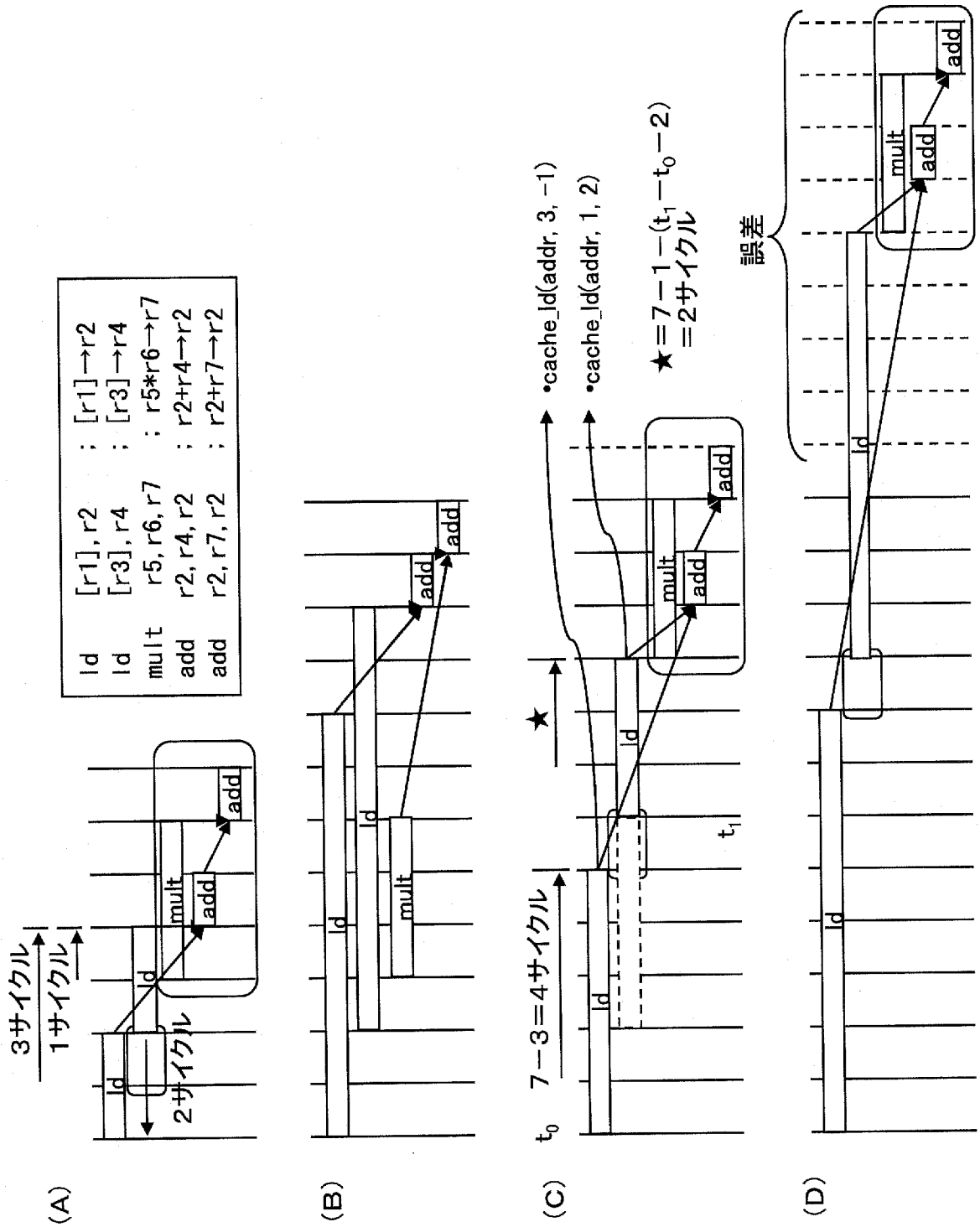
(C)



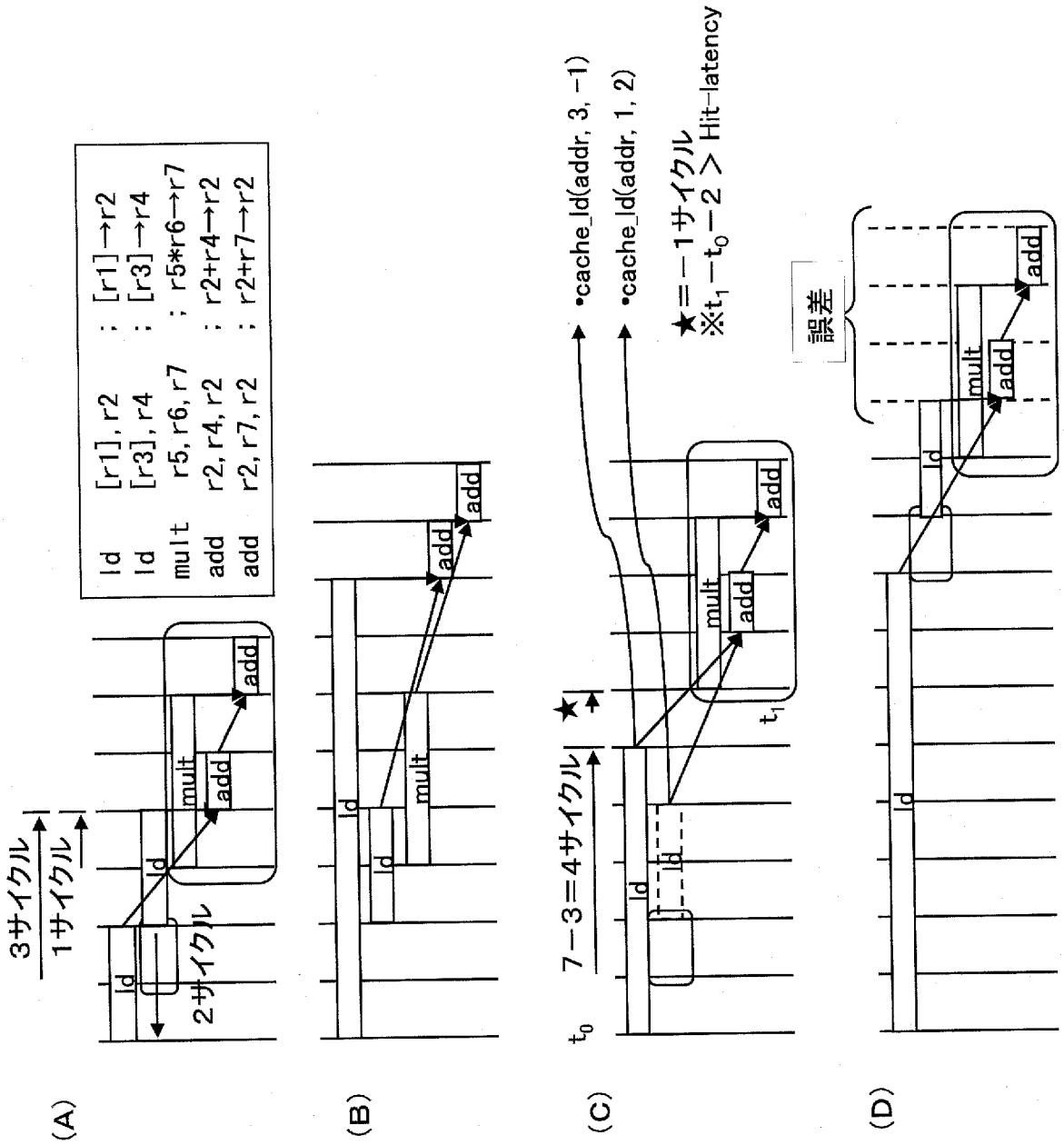
(D)



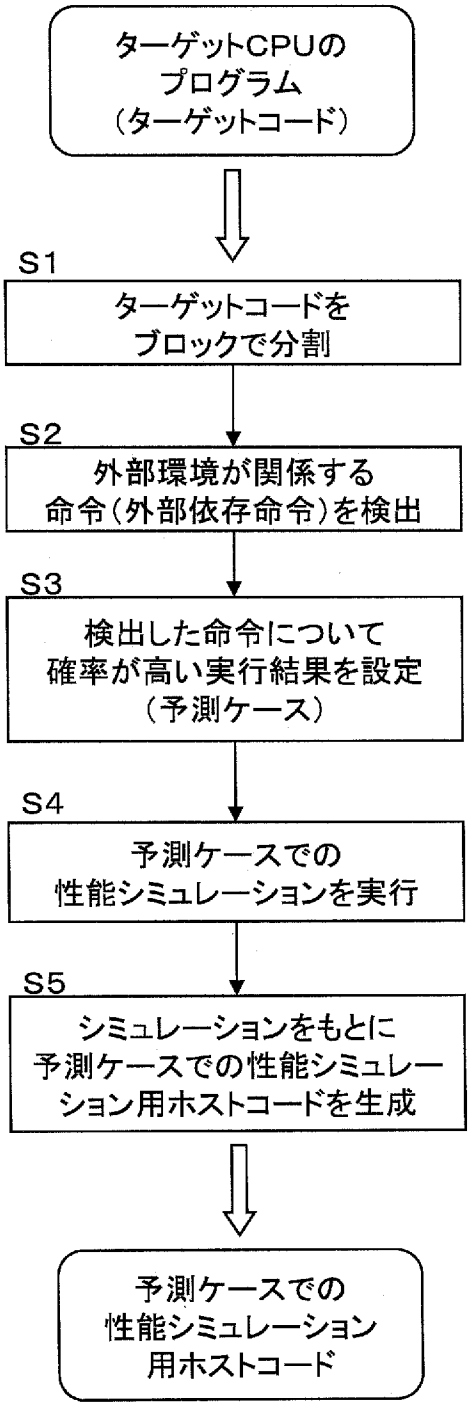
[図8]


ld [r1], r2 ; [r1]→r2
ld [r3], r4 ; [r3]→r4
mult r5, r6, r7 ; r5*r6→r7
add r2, r4, r2 ; r2+r4→r2
add r2, r7, r2 ; r2+r7→r2
3サイクル
1サイクル
2サイクル
ld
mult
add
(A)
(B)
(C)
(D)
•cache_ld(addr, 3, −1)
•cache_ld(addr, 1, 2)
★=7−1−(t1−t0−2)
=2サイクル
7−3=4サイクル
t0
t1
★
誤差

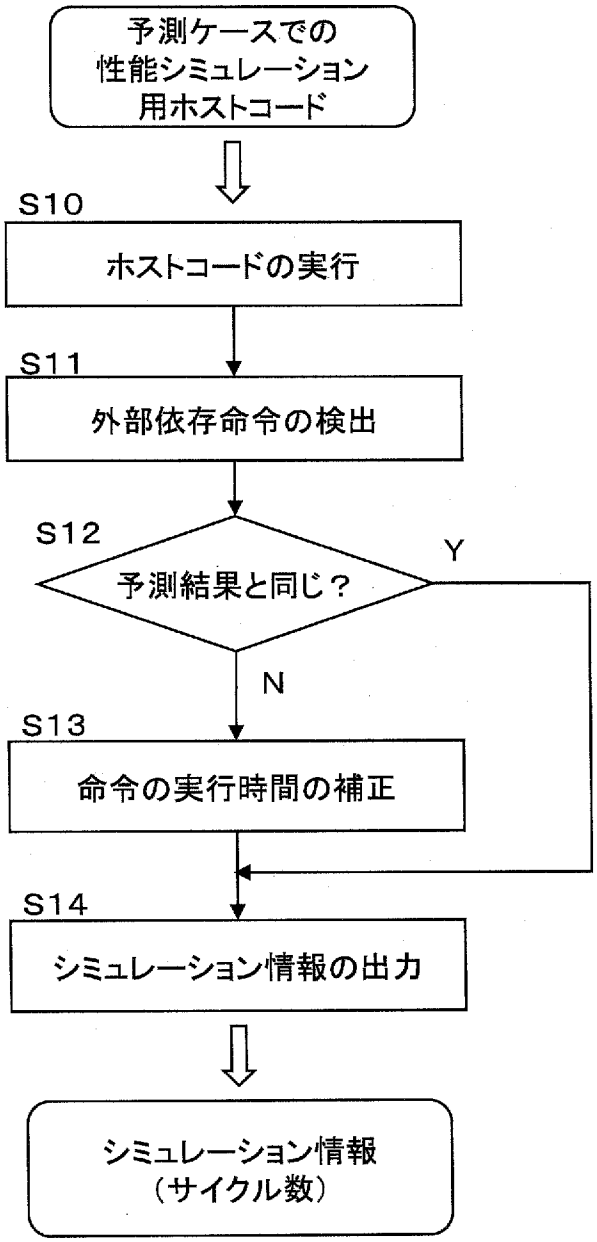
[図9]


ld [r1], r2 ; [r1]→r2
ld [r3], r4 ; [r3]→r4
mult r5, r6, r7 ; r5*r6→r7
add r2, r4, r2 ; r2+r4→r2
add r2, r7, r2 ; r2+r7→r2
3サイクル
1サイクル
2サイクル
ld
mult
add
(A)
(B)
(C)
(D)
•cache_ld(addr, 3, −1)
•cache_ld(addr, 1, 2)
★=−1サイクル
※t1−t0−2 > Hit-latency
7−3=4サイクル
t0
t1
誤差

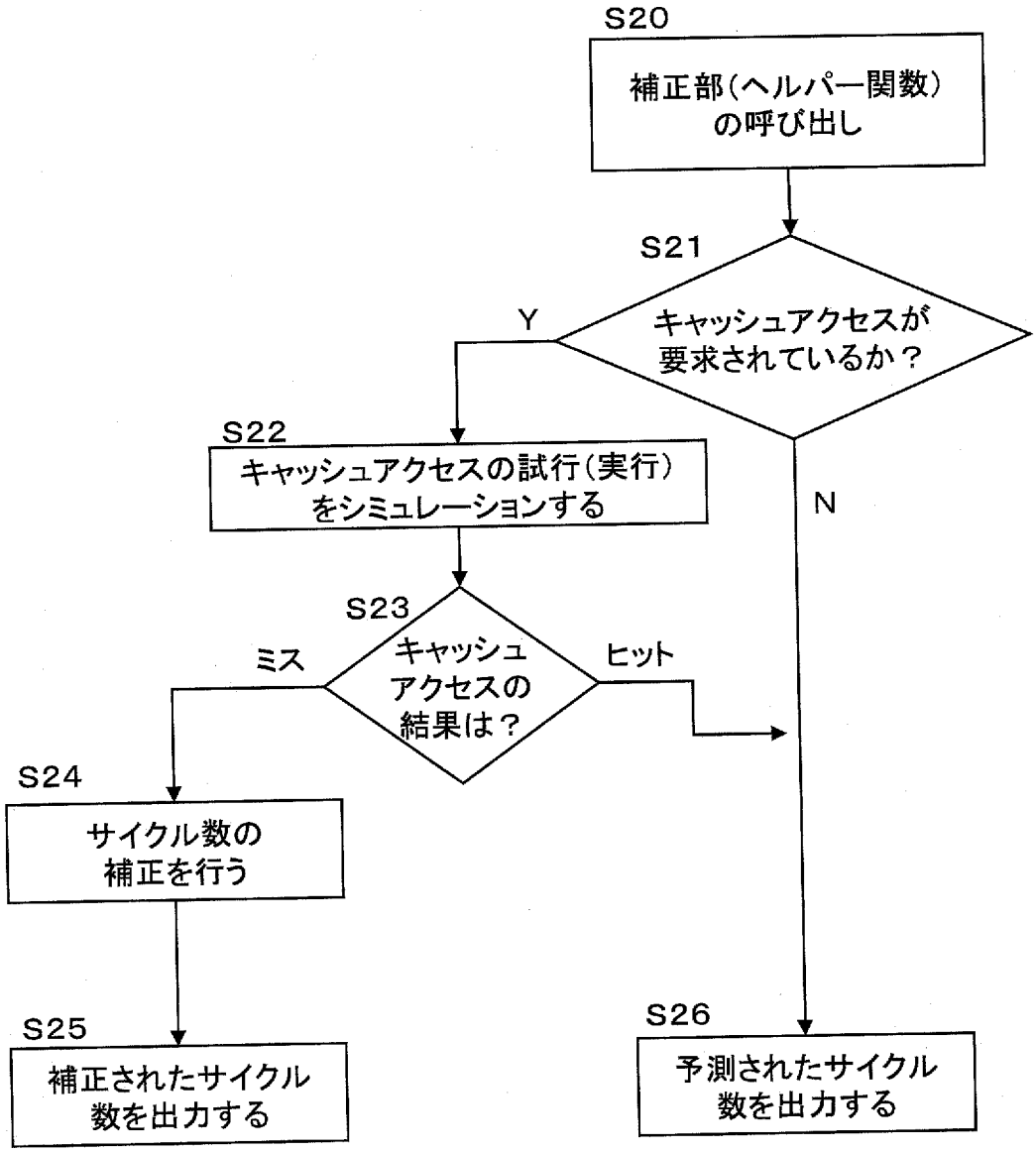
[図10]


ターゲットCPUの
プログラム
(ターゲットコード)
S1
ターゲットコードを
ブロックで分割
S2
外部環境が関係する
命令(外部依存命令)を検出
S3
検出した命令について
確率が高い実行結果を設定
(予測ケース)
S4
予測ケースでの
性能シミュレーションを実行
S5
シミュレーションをもとに
予測ケースでの性能シミュレー
ション用ホストコードを生成
予測ケースでの
性能シミュレーション
用ホストコード

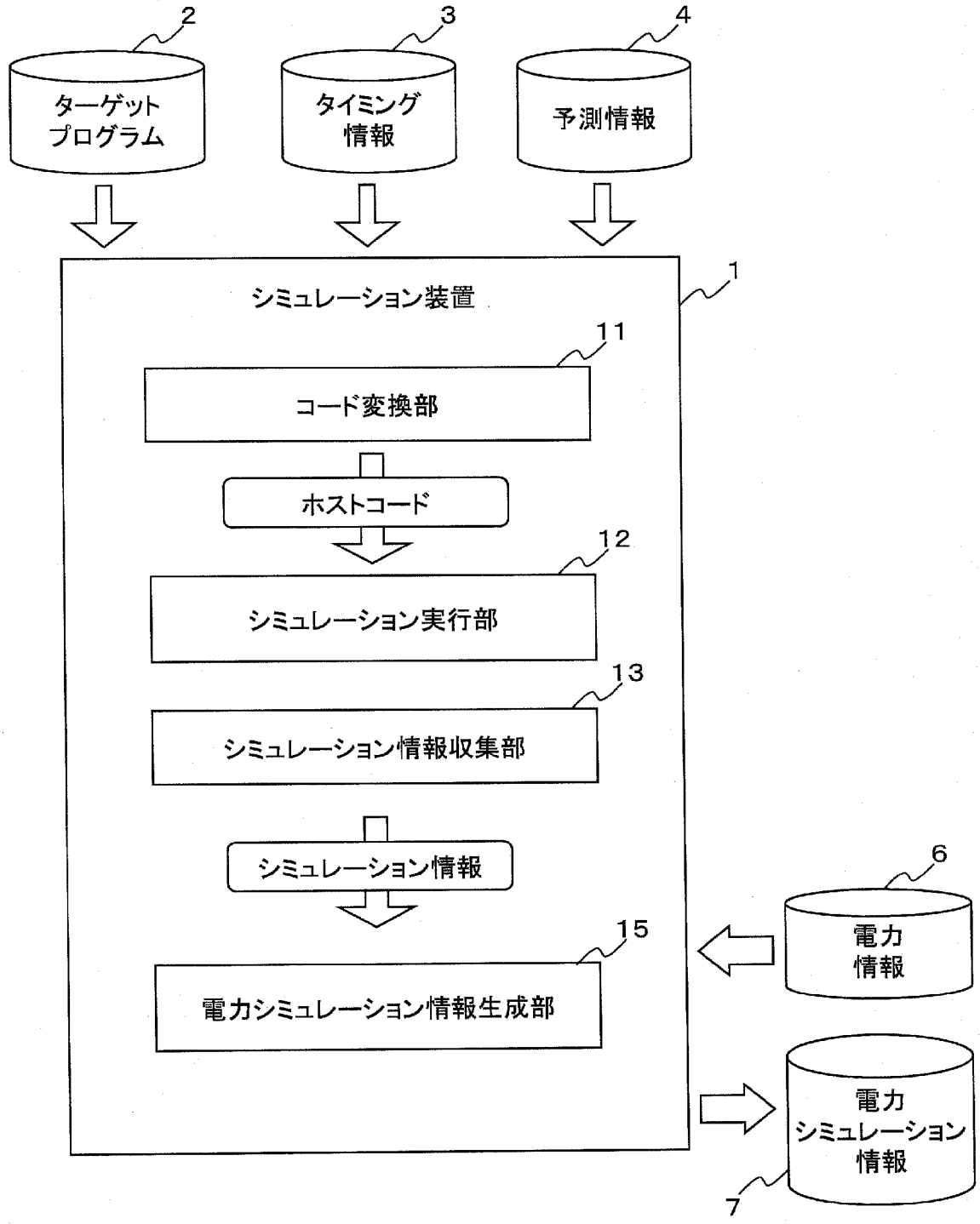
[図11]


予測ケースでの
性能シミュレーション
用ホストコード
S10
ホストコードの実行
S11
外部依存命令の検出
S12
予測結果と同じ？
Y
N
S13
命令の実行時間の補正
S14
シミュレーション情報の出力
シミュレーション情報
（サイクル数）

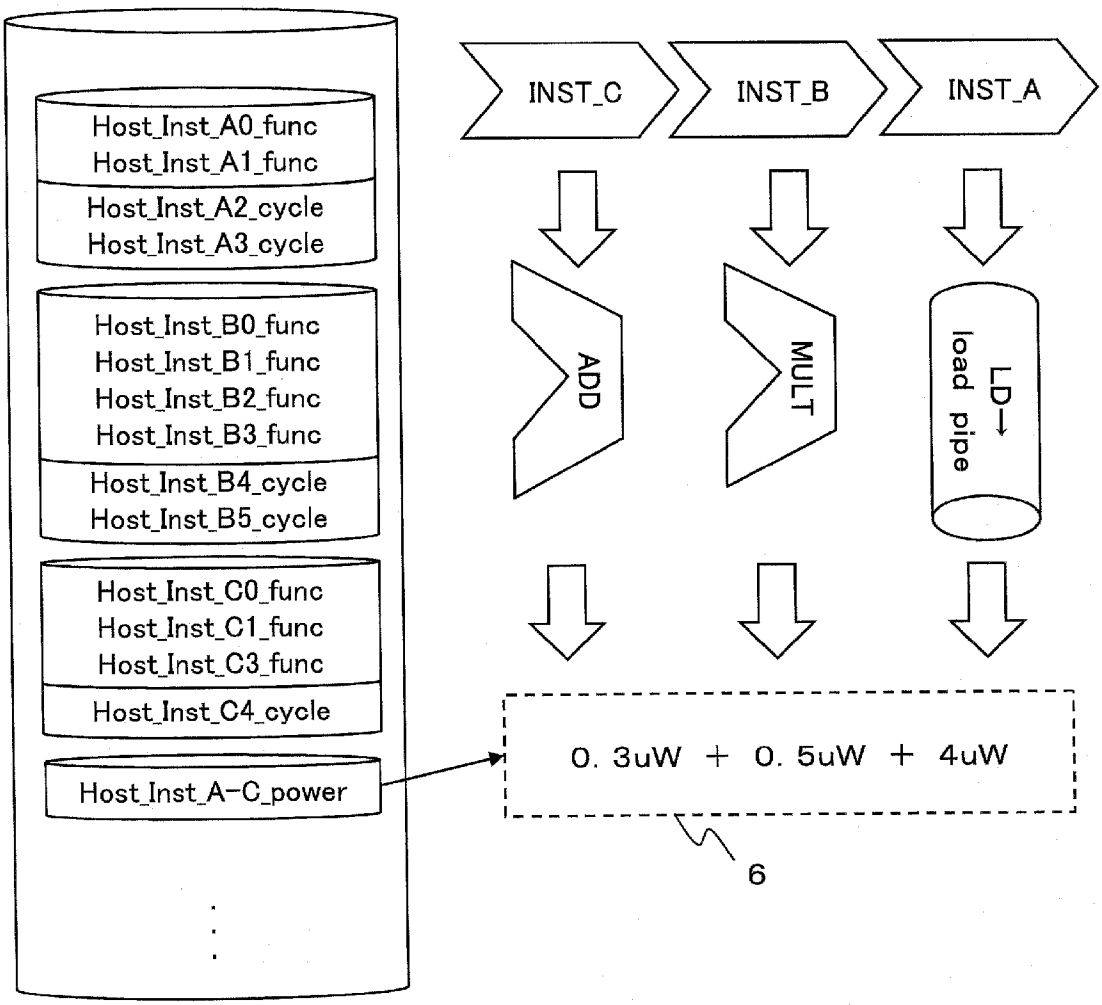
[図12]


S20
補正部(ヘルパー関数)の呼び出し
S21
キャッシュアクセスが要求されているか？
Y
N
S22
キャッシュアクセスの試行(実行)をシミュレーションする
S23
キャッシュアクセスの結果は？
ミス
ヒット
S24
サイクル数の補正を行う
S25
補正されたサイクル数を出力する
S26
予測されたサイクル数を出力する

[図13]


2
3
4
ターゲット
プログラム
タイミング
情報
予測情報
1
シミュレーション装置
11
コード変換部
ホストコード
12
シミュレーション実行部
13
シミュレーション情報収集部
シミュレーション情報
6
電力
情報
15
電力シミュレーション情報生成部
電力
シミュレーション
情報
7

[図14]


Host_Inst_A0_func
Host_Inst_A1_func
Host_Inst_A2_cycle
Host_Inst_A3_cycle
Host_Inst_B0_func
Host_Inst_B1_func
Host_Inst_B2_func
Host_Inst_B3_func
Host_Inst_B4_cycle
Host_Inst_B5_cycle
Host_Inst_C0_func
Host_Inst_C1_func
Host_Inst_C3_func
Host_Inst_C4_cycle
Host_Inst_A-C_power
INST_C
INST_B
INST_A
ADD
MULT
LD→
load pipe
0.3uW + 0.5uW + 4uW
6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2010/067866

A. CLASSIFICATION OF SUBJECT MATTER
G06F9/455(2006.01)i, *G06F9/38*(2006.01)i, *G06F11/28*(2006.01)i

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)
G06F9/455, G06F9/38, G06F11/28

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

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2010
Kokai Jitsuyo Shinan Koho	1971-2010	Toroku Jitsuyo Shinan Koho	1994-2010

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.
A	JP 2000-222245 A (Toshiba Corp.), 11 August 2000 (11.08.2000), paragraphs [0030] to [0054], [0068] to [0072] (Family: none)	1-5
A	JP 2004-227204 A (Hitachi, Ltd.), 12 August 2004 (12.08.2004), paragraphs [0017] to [0019] (Family: none)	1-5
A	JP 2006-23852 A (Semiconductor Technology Academic Research Center), 26 January 2006 (26.01.2006), paragraphs [0107] to [0125] (Family: none)	1-5

☒ 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	Date of mailing of the international search report
28 October, 2010 (28.10.10)	09 November, 2010 (09.11.10)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2010/067866

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 10-254944 A (Toshiba Corp.), 25 September 1998 (25.09.1998), paragraphs [0055] to [0101] & US 6096089 A	3

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１０／０６７８６６

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G06F9/455(2006.01)i, G06F9/38(2006.01)i, G06F11/28(2006.01)i

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））

Int.Cl.　G06F9/455, G06F9/38, G06F11/28

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	１９７１－２０１０年
日本国実用新案登録公報	１９９６－２０１０年
日本国登録実用新案公報	１９９４－２０１０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
A	JP 2000-222245 A (株式会社東芝) 2000.08.11, 第３０～５４，６８～７２段落（ファミリーなし）	1-5
A	JP 2004-227204 A (株式会社日立製作所) 2004.08.12, 第１７～１９段落（ファミリーなし）	1-5
A	JP 2006-23852 A (株式会社半導体理工学研究センター) 2006.01.26, 第１０７～１２５段落（ファミリーなし）	1-5

☑　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矛盾するものではなく、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	国際調査報告の発送日
２８．１０．２０１０	０９．１１．２０１０

国際調査機関の名称及びあて先

日本国特許庁（ＩＳＡ／ＪＰ）

郵便番号１００－８９１５

東京都千代田区霞が関三丁目４番３号

特許庁審査官（権限のある職員）　５Ｂ　９２８８

坂庭　剛史

電話番号　０３－３５８１－１１０１　内線　３５４５

Ｃ（続き）．　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
A	JP 10-254944 A (株式会社東芝) 1998.09.25, 第５５～１０１段落 & US 6096089 A	3