(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7368511号
(P7368511)

(45)発行日　令和5年10月24日(2023.10.24)

(24)登録日　令和5年10月16日(2023.10.16)

(51)国際特許分類		FI		
G06F 9/48 (2006.01)		G06F	9/48	300D
G06F 15/177 (2006.01)		G06F	9/48	350Z
G06F 15/80 (2006.01)		G06F	15/177	B
		G06F	15/80	

請求項の数　17　（全27頁）

(21)出願番号　特願2021-575749(P2021-575749)
(86)(22)出願日　令和3年1月27日(2021.1.27)
(86)国際出願番号　PCT/JP2021/002880
(87)国際公開番号　WO2021/157448
(87)国際公開日　令和3年8月12日(2021.8.12)
審査請求日　令和4年7月12日(2022.7.12)
(31)優先権主張番号　特願2020-16144(P2020-16144)
(32)優先日　令和2年2月3日(2020.2.3)
(33)優先権主張国・地域又は機関
日本国(JP)

(73)特許権者　310021766
株式会社ソニー・インタラクティブエンタテインメント
東京都港区港南１丁目７番１号
(74)代理人　100105924
弁理士　森下 賢樹
(74)代理人　100109047
弁理士　村田 雄祐
(74)代理人　100109081
弁理士　三木 友由
(74)代理人　100134256
弁理士　青木 武司
(72)発明者　大塚 活志
東京都港区港南１丁目７番１号 株式会社ソニー・インタラクティブエンタテイ
最終頁に続く

(54)【発明の名称】　データ処理システム、データ転送装置およびコンテキストスイッチ方法

(57)【特許請求の範囲】

【請求項１】

複数のアプリケーションに関する処理を時分割で実行する処理部と、

前記処理部とメモリ間でデータを転送するデータ転送部と、

を備え、

前記データ転送部は、

前記処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、

前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部から前記メモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行部と、を含み、

前記転送実行部は、前記切替タイミングが検出された場合、前記処理部におけるアプリケーションに関する処理が完了後に前記転送処理を実行することを特徴とするデータ処理システム。

【請求項２】

前記処理部におけるアプリケーションに関する処理は、当該アプリケーションにおける複数の内部処理を含み、

前記転送実行部は、前記処理部における実行対象のアプリケーションの内部処理が完了後に前記転送処理を実行することを特徴とする請求項１に記載のデータ処理システム。

【請求項３】
複数のアプリケーションに関する処理を時分割で実行する処理部と、
前記処理部とメモリ間でデータを転送するデータ転送部と、
を備え、
前記データ転送部は、
前記処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部から前記メモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行部と、を含み、
前記処理部は、ＣＰＵとＧＰＵとを含み、
前記ＣＰＵは、前記ＧＰＵの処理とは非同期に、実行対象の各アプリケーションに関する処理内容を指示するコマンドを、ＧＰＵコマンドバッファにおける各アプリケーションに対応するキューに書き込み、
前記ＧＰＵは、実行対象のアプリケーションに対応するＧＰＵコマンドバッファのキューから前記コマンドを読み出し、
前記ＧＰＵは、実行対象のアプリケーションを切り替える際に、コマンドを読み出すキューを切り替えることを特徴とするデータ処理システム。
【請求項４】
前記メモリに退避されるアプリケーションのコンテキストは、前記処理部において未完了の処理の現在状態を含み、
前記転送実行部は、前記切替タイミングが検出された場合、前記処理部におけるアプリケーションに関する処理が未完了であっても前記転送処理を実行することを特徴とする請求項３に記載のデータ処理システム。
【請求項５】
前記検出部は、垂直帰線期間または水平帰線期間の開始を前記切替タイミングとして検出することを特徴とする請求項１から４のいずれかに記載のデータ処理システム。
【請求項６】
前記転送実行部は、ＤＦＴ（Design For Test）用のスキャンチェーンとメモリＢＩＳＴ（Built In Self-Test）用の回路の少なくとも一方を用いて前記転送処理を実行することを特徴とする請求項１から５のいずれかに記載のデータ処理システム。
【請求項７】
複数のアプリケーションに関する処理を時分割で実行する処理部と、
前記処理部とメモリ間でデータを転送するデータ転送部と、
を備え、
前記データ転送部は、
前記処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部から前記メモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行部と、を含み、
前記処理部は、アプリケーションのコンテキストを保持する機能部を複数含み、
前記転送実行部は、前記処理部が第１の機能部に保持されたアプリケーションのコンテキストを用いて当該アプリケーションに関する処理を実行する間に、前記処理部の第２の機能部に保持された別のアプリケーションのコンテキストを対象として前記転送処理を実行することを特徴とするデータ処理システム。
【請求項８】
複数のアプリケーションに関する処理を時分割で実行する処理部と、

前記処理部とメモリ間でデータを転送するデータ転送部と、
を備え、
前記データ転送部は、
前記処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部から前記メモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行部と、を含み、
前記処理部は、アプリケーションのコンテキストを保持する機能部を複数含み、
前記処理部は、第１の機能部に保持された第１のアプリケーションのコンテキストを用いて前記第１のアプリケーションに関する処理を実行し、第２の機能部に保持された第２のアプリケーションのコンテキストを用いて前記第２のアプリケーションに関する処理を実行し、
前記処理部における実行対象が前記第１のアプリケーションから前記第２のアプリケーションに切り替わる場合、前記処理部は、前記第１のアプリケーションに関する処理のうち前記第２のアプリケーションに関する処理に対して依存関係を有する処理の完了後に、前記第１のアプリケーションに関する処理のうち前記第２のアプリケーションに関する処理に対して依存関係がない処理と、前記第２のアプリケーションに関する処理とを並列実行することを特徴とするデータ処理システム。

【請求項９】
前記転送実行部による転送処理の対象は、前記処理部のステート・マシンに保持されたコンテキストを含むことを特徴とする請求項１から８のいずれかに記載のデータ処理システム。

【請求項１０】
前記転送実行部による転送処理の対象は、前記処理部に保持されるコンテキストであって、ソフトウェアからアクセスできないコンテキストを含むことを特徴とする請求項１から９のいずれかに記載のデータ処理システム。

【請求項１１】
前記コンテキストは、中断された時点のアプリケーションの処理の再現に必要なデータであることを特徴とする請求項１から１０のいずれかに記載のデータ処理システム。

【請求項１２】
複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部からメモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行部と、
を備え、
前記転送実行部は、前記切替タイミングが検出された場合、前記処理部におけるアプリケーションに関する処理が完了後に前記転送処理を実行することを特徴とするデータ転送装置。

【請求項１３】
複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部からメモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行部と、
を備え、

前記処理部は、アプリケーションのコンテキストを保持する機能部を複数含み、
前記転送実行部は、前記処理部が第１の機能部に保持されたアプリケーションのコンテキストを用いて当該アプリケーションに関する処理を実行する間に、前記処理部の第２の機能部に保持された別のアプリケーションのコンテキストを対象として前記転送処理を実行することを特徴とするデータ転送装置。
【請求項１４】
複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出するステップと、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部からメモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行ステップと、
をコンピュータが実行し、
前記転送実行ステップは、前記切替タイミングが検出された場合、前記処理部におけるアプリケーションに関する処理が完了後に前記転送処理を実行することを特徴とするコンテキストスイッチ方法。
【請求項１５】
複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出するステップと、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部からメモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行するステップと、
をコンピュータが実行し、
前記処理部は、ＣＰＵとＧＰＵとを含み、
前記ＣＰＵは、前記ＧＰＵの処理とは非同期に、実行対象の各アプリケーションに関する処理内容を指示するコマンドを、ＧＰＵコマンドバッファにおける各アプリケーションに対応するキューに書き込み、
前記ＧＰＵは、実行対象のアプリケーションに対応するＧＰＵコマンドバッファのキューから前記コマンドを読み出し、
前記ＧＰＵは、実行対象のアプリケーションを切り替える際に、コマンドを読み出すキューを切り替えることを特徴とするコンテキストスイッチ方法。
【請求項１６】
複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出するステップと、
前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部からメモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行する転送実行ステップと、
をコンピュータが実行し、
前記処理部は、アプリケーションのコンテキストを保持する機能部を複数含み、
前記転送実行ステップは、前記処理部が第１の機能部に保持されたアプリケーションのコンテキストを用いて当該アプリケーションに関する処理を実行する間に、前記処理部の第２の機能部に保持された別のアプリケーションのコンテキストを対象として前記転送処理を実行することを特徴とするコンテキストスイッチ方法。
【請求項１７】
複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出するステップと、

前記切替タイミングが検出された場合、前記複数のアプリケーションを管理するソフトウェアの処理を介さずに、前記処理部で実行されていたアプリケーションのコンテキストを前記処理部からメモリへ退避させ、前記処理部で次に実行されるアプリケーションのコンテキストを前記メモリから前記処理部に設定する転送処理を実行するステップと、
をコンピュータが実行し、
前記処理部は、アプリケーションのコンテキストを保持する機能部を複数含み、
前記処理部は、第１の機能部に保持された第１のアプリケーションのコンテキストを用いて前記第１のアプリケーションに関する処理を実行し、第２の機能部に保持された第２のアプリケーションのコンテキストを用いて前記第２のアプリケーションに関する処理を実行し、
前記処理部における実行対象が前記第１のアプリケーションから前記第２のアプリケーションに切り替わる場合、前記処理部は、前記第１のアプリケーションに関する処理のうち前記第２のアプリケーションに関する処理に対して依存関係を有する処理の完了後に、前記第１のアプリケーションに関する処理のうち前記第２のアプリケーションに関する処理に対して依存関係がない処理と、前記第２のアプリケーションに関する処理とを並列実行することを特徴とするコンテキストスイッチ方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本開示はデータ処理技術に関し、特にデータ処理システム、データ転送装置およびコンテキストスイッチ方法に関する。

【背景技術】

【０００２】

複数のアプリケーションに関する処理を時分割で実行する処理部では、実行対象のアプリケーションを切り替える際にコンテキストスイッチを行う必要がある。

【発明の概要】

【発明が解決しようとする課題】

【０００３】

近年、処理部（例えばＧＰＵ等）は、数千以上のレジスタを備えることがあり、また、処理部で実行されるアプリケーションのコンテキストは、数十ＭＢ以上のサイズになることがある。この場合、処理部のコンテキストスイッチに長時間を要する可能性がある。

【０００４】

本開示はこうした状況に鑑みてなされたものであり、１つの目的は、コンテキストスイッチに伴う処理の遅延を抑制することにある。

【課題を解決するための手段】

【０００５】

上記課題を解決するために、本発明のある態様のデータ処理システムは、複数のアプリケーションに関する処理を時分割で実行する処理部と、処理部とメモリ間でデータを転送するデータ転送部と、を備える。データ転送部は、処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、切替タイミングが検出された場合、複数のアプリケーションを管理するソフトウェアの処理を介さずに、処理部で実行されていたアプリケーションのコンテキストを処理部からメモリへ退避させ、処理部で次に実行されるアプリケーションのコンテキストをメモリから処理部に設定する転送処理を実行する転送実行部と、を含む。

【０００６】

本発明の別の態様は、データ転送装置である。この装置は、複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出する検出部と、切替タイミングが検出された場合、複数のアプリケーションを管理するソフトウェアの処理を介さずに、処理部で実行されていたアプリケーションのコンテキストを処理部からメモリへ退避させ、処理部で次に実行されるアプリケーションの

コンテキストをメモリから処理部に設定する転送処理を実行する転送実行部と、を備える。
【０００７】
本発明のさらに別の態様は、コンテキストスイッチ方法である。この方法は、複数のアプリケーションに関する処理を時分割で実行する処理部における実行対象のアプリケーションの切替タイミングを検出するステップと、切替タイミングが検出された場合、複数のアプリケーションを管理するソフトウェアの処理を介さずに、処理部で実行されていたアプリケーションのコンテキストを処理部からメモリへ退避させ、処理部で次に実行されるアプリケーションのコンテキストをメモリから処理部に設定する転送処理を実行するステップと、をコンピュータが実行する。
【０００８】
なお、以上の構成要素の任意の組合せ、本開示の表現を、コンピュータプログラム、コンピュータプログラムを記録した記録媒体などの間で変換したものもまた、本開示の態様として有効である。
【発明の効果】
【０００９】
本開示によれば、コンテキストスイッチに伴う処理の遅延を抑制することができる。
【図面の簡単な説明】
【００１０】
【図１】第１実施例のコンピュータの構成を示す図である。
【図２】処理部とＣＳＤＭＡエンジンの詳細な構成を示すブロック図である。
【図３】処理部におけるビデオ描画処理を時系列に示す図である。
【図４】ビデオタイミングの例を示す図である。
【図５】ＳＯＣの構成を示す図である。
【図６】ＳＯＣの構成を示す図である。
【図７】ＳＯＣの構成を示す図である。
【図８】ＳＯＣの構成を示す図である。
【図９】図９Ａは、第１実施例におけるコンテキストスイッチ時の内部処理の例を示す図であり、図９Ｂは、変形例におけるコンテキストスイッチ時の内部処理の例を示す図である。
【図１０】第２実施例の処理部の構成を示す図である。
【図１１】第３実施例におけるコンテキストスイッチ時の内部処理の例を示す図である。
【図１２】図１２Ａ、図１２Ｂ、図１２Ｃは、コンテキストスイッチにおける処理部とＣＳＤＭＡエンジンの動作を示す図である。
【図１３】第４実施例の処理部の構成を示す図である。
【図１４】第５実施例の処理部の構成を示す図である。
【図１５】図１５Ａと図１５Ｂは、アプリケーションの割当例を示す図である。
【発明を実施するための形態】
【００１１】
実施例では、複数のアプリケーションに関する処理を時分割で実行する処理部であり、言い換えれば、複数のアプリケーションにより時分割で共有される処理部におけるコンテキストスイッチを高速に実行する技術を提案する。実施例の技術は、或るシステムが、サスペンドまたはハイバネーション等、状態を保持したまま一時停止し、その後に、保持した状態から実行を再開する動作にも有効であり、その動作を高速化することができる。
【００１２】
処理部は、ＳＯＣ（System-On-a-Chip）等におけるハードウェア機能ブロックであってもよい。また、処理部は、（１）ＣＰＵ（Central Processing Unit）・ＤＳＰ（Digital Signal Processor）・ＧＰＵ（Graphics Processing Unit）・ＮＰＵ（Network Processing Unit）等のプロセッサであってもよく、（２）ビデオの圧縮および伸長を実行するビデオ・コーデック・ブロックであってもよい。ビデオ・コーデック・ブロックのうちビデオの圧縮を実行するブロックを以下「ビデオエンコーダ」とも呼ぶ。また、処理

部は、（３）ビデオの解像度変換や画質変換、多重化等を実行し、規定のタイミングでピクセルデータを処理するビデオ（ディスプレイ）・パイプライン・ブロック（以下「ビデオパイプライン」とも呼ぶ。）であってもよく、（４）周辺機器への入出力インタフェースブロックであってもよい。

【００１３】

＜背景＞

１つのハードウェア機能ブロック（実施例では「処理部」と呼ぶ。）が、複数のアプリケーション・タスク・スレッド（以下「アプリケーション」と呼ぶ。）を時分割で実行することがある。処理部は、アプリケーションごとに固有の処理を実行するため、実行対象のアプリケーションを指定するデータや、設定・動作を指定するデータ等を保持する。処理部は、処理の開始にあたって、それらのデータをセットアップした上で処理を開始する。処理部は、アプリケーションに関する処理を実行中に、アプリケーション固有のデータや設定、動作状態等を随時生成する。これら個々のアプリケーションに関する処理において必要となるデータや生成されるデータを以下「コンテキスト」と呼ぶ。つまり、コンテキストとは、同じ処理を、再び同じ処理部で実行したときに、同様の結果を得る再現性が確保されるために必要なデータである。

【００１４】

１つの処理部が複数のアプリケーションを時分割で実行する場合、当該処理部における切替処理が必要になる。具体的には、それまで処理部が実行していたアプリケーションのコンテキストを外部に退避させ、次に実行するアプリケーションのコンテキストを処理部に設定することで、時分割によるアプリケーション切替がなされる。この切替を以下「コンテキストスイッチ」と呼ぶ。コンテキストの退避と設定を行う間は、当該処理部の処理を停止する必要がある。処理部の利用効率を高め、また、処理部における処理の遅延を低減する（言い換えればリアルタイム性を高める）ために、コンテキストスイッチを高速化し、処理部の処理停止時間を短縮することが求められる。

【００１５】

一般にコンテキストスイッチは、ソフトウェアが処理部にアクセスし、ソフトウェアによる逐次処理にて、コンテキストの退避と設定をレジスタアクセスを介して実行されることが多い。一方、近年の処理部は高機能化が進み、設定用レジスタ数は数千以上、また、処理中に保持されているコンテキストのデータ量は数十ＭＢ以上になることがある。そのため、ソフトウェアによる逐次処理にてコンテキストスイッチをおこなうと、数十ミリ秒オーダーの時間を要するなど、リアルタイム性を大きく損ねてしまう可能性がある。

【００１６】

ＣＰＵの中には、コンテキストスイッチの高速化を支援するハードウェア機能を持つものがある。しかし、その機能は、ソフトウェアからアクセス可能な一部のレジスタについてコンテキストの退避と設定をハードウェアが自動的に実行するものに過ぎず、内部に保持される全コンテキストの退避と設定をカバーするものではない。そのため、ソフトウェアによる逐次処理にて他のコンテキストの退避と設定を行っている。これは、一般的なｘ８６またはＡＲＭアーキテクチャのＣＰＵにおいて、ＯＳ（Ｗｉｎｄｏｗｓ（登録商標）やＬｉｎｕｘ（登録商標））と複数のアプリケーションが実行される場合等に言える。

【００１７】

＜実施例の概要＞

実施例のデータ処理システムは、処理部のコンテキストスイッチのためのコンテキスト転送を実行する専用機能ブロックを備える。これにより、膨大なコンテキストを利用する処理部とアプリケーションの組み合わせにおいても、コンテキストスイッチを高速に実行する。言い換えれば、コンテキストスイッチに伴う処理の遅延を抑制する。以下、この専用機能ブロックを、「コンテキストスイッチＤＭＡ（Direct Memory Access）エンジン」と呼び、「ＣＳＤＭＡエンジン」と表記する。

【００１８】

ＣＳＤＭＡエンジンは、アプリケーションの切替タイミングを通知されるもしくは自ら

検出すると、（１）メモリ上に保持されている各アプリケーションのコンテキストを選択し、ソフトウェアを介さずに処理部へ転送して設定する。もしくは、（２）処理部に保持されているコンテキストを、ソフトウェアを介さずにメモリへ転送・退避させる。

【００１９】

コンテキストスイッチにおける転送対象は、レジスタ内容と、処理部内メモリの内容と、ステート・マシンの状態ステート（状態遷移の内部ステート）と、オンザフライで実行中のアウトスタンディングな処理の現在状態とを含む。これらの転送対象は、ソフトウェアからアクセス可能なものに限定されず、それ以外に、処理部が内蔵し、コンテキストを保持しているものも含む。つまり、実施例におけるコンテキストスイッチでの転送対象は、通常はソフトウェアからアクセスできないコンテキストも含む。

【００２０】

ＣＳＤＭＡエンジンは、アプリケーションの切替タイミングを通知されるもしくは自ら検出すると、処理部へ動作停止指示を出し、また、オンザフライで実行中の処理、特にアウトスタンディング（out standing）やアウトオブオーダ（out of order）でコミット処理が必要な処理についてその完了を確認次第、コンテキストの退避と設定を開始してもよい。ＣＳＤＭＡエンジンは、コンテキストの退避と設定の完了後に、次のアプリケーションの処理開始を指示してもよい。これら一連の処理は、ＣＳＤＭＡエンジンが実行することに制限されず、最新のコンテキストを保持する処理部とＣＳＤＭＡエンジンとが連携して実行されてもよい。

【００２１】

処理部には、複数のアプリケーション用の複数のキューを設けてもよい。それぞれのキューには、対応するアプリケーションから、処理部に対する指示を示すデータが随時書き込まれてもよい。処理部は、コンテキストスイッチが発生すると、データを読み出すキューを切り替えてもよい。また、或るアプリケーションに対応するキューにデータが書き込まれるタイミングは、そのアプリケーションが処理部を占有している期間に制限されなくてもよい。この場合、コンテキストスイッチ実行中も、各アプリケーションは、処理の指示をキューに積み続けることができ、コンテキストスイッチに伴う処理の停止時間を短縮することができる。また、キューにデータを書き込むＣＰＵ上のソフトウェアが、処理部（例えばＧＰＵ）と非同期に動作する場合にも対応できる。

【００２２】

ＣＳＤＭＡエンジンは、垂直帰線期間または水平帰線期間の開始をトリガとして、コンテキストスイッチを実行してもよい。ビデオ処理システム等では、垂直帰線期間および水平帰線期間に、ＧＰＵやビデオパイプライン、ビデオエンコーダ等を一時停止することができる。そこで、短時間の垂直帰線期間中または水平帰線期間中にコンテキストスイッチを実行することで、コンテキストスイッチのオーバーヘッドを隠蔽することができる。

【００２３】

処理部からコンテキストを退避する際、および、処理部へコンテキストを設定する際に、ＤＦＴ（Design For Test）用のスキャンチェーン、または、メモリＢＩＳＴ（Built-In Self-Test）を経路として用いてもよい。これにより、処理部とメモリとをＣＳＤＭＡエンジンを介して接続する系が、通常のデータ転送に用いるチップ内バス（ファブリック）を介さないため、通常のデータ転送を阻害しない。また、コンテキストスイッチのための専用インタフェースを設ける必要がない。また、処理部内においても、コンテキスト情報を保持する回路にＤＦＴ回路が接続されているため、通常のデータ転送用の系や専用系を用いる必要がなくなる。

【００２４】

処理部の機能ブロックのうち、コンテキストを保持する機能ブロック（例えば回路）のみ、その一部または全てを多重化してもよい。この場合、以下のステップでコンテキストスイッチが実行されてもよい。（１）処理部を一時停止させることなく、多重化された他の回路に対して、コンテキストを予め設定する。（２）処理部の処理を停止させ（直前処理のコミットを待ってもよい）、上記（１）でコンテキストを設定した回路を選択し、次

のアプリケーションの実行を開始する。（３）上記（１）で実行中であったアプリケーションのコンテキストを退避させる。これにより、コンテキストスイッチの処理と、処理部の処理とを時間的にオーバーラップさせ、処理の停止時間を一層削減することができる。

【００２５】

また、処理部の機能ブロックのうち、コンテキストを保持する回路を多重化し、さらに、処理部は、コミット処理を待つための一時停止を行わずに、次のアプリケーションの処理を開始してもよい。ただし、コンテキストスイッチ前のアプリケーションと、コンテキストスイッチ後のアプリケーションとで、同一回路を別設定にて動かす等、両アプリケーションの設定が共通ではないことによる処理の破綻がおこるか否かを判定し、必要な場合は、コミット完了まで一時停止してもよい。さらにまた、一部の機能回路を並列化し、処理破綻を回避しながら、複数アプリケーションを同時実行してもよい。これにより、複数アプリケーションの処理（アウトスタンディングまたはアウトオブオーダの処理であってもよい）を並行して実行でき、コンテキストスイッチによる停止時間を短縮することができる。また、回路利用効率、性能スケーラビリティおよびフレキシビリティを向上することができる。

【００２６】

＜第１実施例＞

図１は、第１実施例のコンピュータ１０の構成を示す。コンピュータ１０は、複数のクライアント装置の要求に応じて、複数のアプリケーション（ゲーム等）を並列処理し、各アプリケーションの処理結果（例えば画像データ）を各クライアント装置に提供するサーバであってもよい。また、コンピュータ１０は、複数のアプリケーションを時分割で並列実行し、各アプリケーションの処理結果（例えば画像データ）を表示装置に表示させるＰＣやゲーム機（ビデオゲームコンソール等）であってもよい。

【００２７】

コンピュータ１０は、ＳＯＣ（System On a Chip）１１とメインメモリ２２とを備える。ＳＯＣ１１は、１つのチップ上にデータ処理システムの機能が実装された集積回路製品である。ＳＯＣ１１は、ＣＰＵ１２、ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８、ビデオタイミングジェネレータ２０を備える。

【００２８】

ＣＰＵ１２は、並列処理の対象となる複数のアプリケーションを時分割で実行する。また、ＣＰＵ１２は、並列処理の対象となる複数のアプリケーションを管理するソフトウェア（以下「管理ソフトウェア」とも呼ぶ。）を実行する。管理ソフトウェアは、ＯＳやミドルウェア等、アプリケーションより下のレイヤに位置づけられるソフトウェアであってもよく、実施例では、複数のアプリケーションの実行順序やコンテキストスイッチを管理する。なお、管理ソフトウェアの一部の機能は、ＧＰＵ１４やビデオエンコーダ１６等（後述の処理部４６）により実行されてもよい。

【００２９】

ＧＰＵ１４は、ＣＰＵ１２において時分割で実行される複数のアプリケーションの指示に応じて、複数のアプリケーションに関する画像処理や汎用計算処理を時分割で実行する。例えば、ＧＰＵ１４は、複数のアプリケーションそれぞれの画像データを生成し、生成した画像データを、各アプリケーションに対応するフレームバッファに書き込む。

【００３０】

ビデオエンコーダ１６は、並列処理対象の複数のアプリケーションの画像の圧縮処理を時分割で実行する。例えば、ビデオエンコーダ１６は、複数のアプリケーションに対応する複数のフレームバッファから各アプリケーションの画像データを順次読み出し、読み出した画像データに対する圧縮符号化処理を実行する。

【００３１】

ビデオパイプライン１８は、画像供給元の他ブロック（例えばＧＰＵ１４または不図示のビデオデコーダ等）から供給された画像であり、言い換えれば、並列処理対象の複数のアプリケーションの画像に対する、解像度変換、画質変換または多重化を時分割で実行す

る。
【００３２】
ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８間での画像受け渡しのケースは、以下の３パターンを含む。
（１）ＧＰＵ１４→ビデオパイプライン１８→ビデオエンコーダ１６。
（２）ＧＰＵ１４→ビデオエンコーダ１６。
（３）不図示の他ブロック（ビデオデコーダ等）→ビデオパイプライン１８→ビデオエンコーダ１６。ただし、ビデオパイプライン１８には、ＧＰＵ１４の処理結果も入力される。
すなわち、ビデオエンコーダ１６が、（Ａ）ＣＰＵ１２が生成した画像を直接参照するパターンと、（Ｂ）ビデオパイプライン１８が加工後の画像を参照するパターンのいずれもある。
【００３３】
このように、ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８は、並列処理の対象となる複数のアプリケーションに関する処理を時分割で実行するものであり、以下、ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８を総称して「処理部４６」と呼ぶ。
【００３４】
ビデオタイミングジェネレータ２０は、画像の表示に関する各種タイミング（ビデオタイミングとも言える）を各機器に通知する。実施例では、ビデオタイミングジェネレータ２０は、表示装置の垂直帰線期間（もしくは垂直帰線期間に対応する期間）の開始を通知する信号をＣＰＵ１２、ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８へ送信する。
【００３５】
また、ＳＯＣ１１は、ＣＳＤＭＡエンジン４０、ＣＳＤＭＡエンジン４２、ＣＳＤＭＡエンジン４４（総称する場合「ＣＳＤＭＡエンジン４８」と呼ぶ。）をさらに備える。ＣＳＤＭＡエンジン４８は、コンテキストスイッチのためのコンテキスト転送をＤＭＡにより実行するデータ転送部である。
【００３６】
ＣＳＤＭＡエンジン４０は、ＧＰＵ１４におけるコンテキストスイッチに伴いＧＰＵ１４とメインメモリ２２間でコンテキストを転送する。ＣＳＤＭＡエンジン４２は、ビデオエンコーダ１６におけるコンテキストスイッチに伴いビデオエンコーダ１６とメインメモリ２２間でコンテキストを転送する。ＣＳＤＭＡエンジン４４は、ビデオパイプライン１８におけるコンテキストスイッチに伴いビデオパイプライン１８とメインメモリ２２間でコンテキストを転送する。
【００３７】
パス５０は、ＧＰＵ１４による通常のメモリアクセス用（コマンドの読み込みおよび画像データの格納を含む）のパスである。パス５２は、ＣＰＵ１２で実行されるソフトウェア（例えば管理ソフトウェア）からＧＰＵ１４およびＣＳＤＭＡエンジン４０へのアクセスパス（設定・確認・通知・制御用）である。パス５２は、同様の目的で他ブロックにも接続されている。なお、図中の線の交点に黒丸がある箇所は線が接続されている状態を示し、線の交点に黒丸がない箇所は線が接続されていない状態を示す。パス５４は、メインメモリ２２からＧＰＵ１４へのコンテキスト設定パスである。パス５６は、ＧＰＵ１４からメインメモリ２２へのコンテキスト退避パスである。
【００３８】
パス５８は、ＣＳＤＭＡエンジン４０がＧＰＵ１４の処理状態を監視するパスである。パス６０は、ＣＳＤＭＡエンジン４０がＧＰＵ１４へアプリケーション処理の停止と開始（再開）を指示するパスである。パス６２は、ビデオタイミングジェネレータ２０からＧＰＵ１４およびＣＳＤＭＡエンジン４０へ垂直帰線期間の開始を通知するパスである。パス６２は、同様の目的で他ブロックにも接続されている。ビデオエンコーダ１６とＣＳＤ

ＭＡエンジン４２間の各パス、ビデオパイプライン１８とＣＳＤＭＡエンジン４４間の各パスも同様である。

【００３９】

メインメモリ２２は、ＳＯＣ１１により参照または更新されるデータを記憶する。メインメモリ２２は、ＧＰＵコマンドバッファ３０、ＧＰＵコンテキスト３２、ビデオエンコーダコンテキスト３４、ビデオパイプラインコンテキスト３６を記憶する。

【００４０】

ＧＰＵコンテキスト３２は、ＳＯＣ１１において並列実行される複数のアプリケーション（例えばＡｐｐ　Ａ、Ａｐｐ　Ｂ等）に関するＧＰＵ１４で使用されるコンテキストである。同様に、ビデオエンコーダコンテキスト３４は、ＳＯＣ１１において並列実行される複数のアプリケーションに関するビデオエンコーダ１６で使用されるコンテキストである。ビデオパイプラインコンテキスト３６は、ＳＯＣ１１において並列実行される複数のアプリケーションに関するビデオパイプライン１８で使用されるコンテキストである。

【００４１】

ＧＰＵコマンドバッファ３０は、ＳＯＣ１１で並列処理される複数のアプリケーションに対応する複数のキュー（例えばＡｐｐ　Ａ用Ａｐｐ　Ｂ用等）を含む。ＧＰＵコマンドバッファ３０の各キューには、各アプリケーションに関する処理内容を指示するコマンドがＣＰＵ１２により書き込まれ、蓄積される。言い換えれば、ＣＰＵ１２により実行される各アプリケーションは、自アプリケーションに対応するＧＰＵコマンドバッファ３０のキューに、描画指示等のコマンドを格納する。ＧＰＵ１４は、実行対象のアプリケーションに対応するＧＰＵコマンドバッファ３０のキューからコマンドを読み出す。ＧＰＵ１４は、実行対象のアプリケーションを切り替える際に、コマンドを読み出すキューを切り替える。

【００４２】

このように、ＧＰＵコマンドバッファ３０を設けることで、ＣＰＵ１２は、処理部４６（例えばＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８）と非同期に動作することができる。すなわち、ＣＰＵ１２において並列に（時分割等で）実行される各アプリケーションは、処理部４６が他アプリケーションの処理をしている間も、自アプリケーションに関するコマンドをＧＰＵコマンドバッファ３０に随時格納することができる。言い換えれば、ＧＰＵコマンドバッファ３０に書き込みを行うＣＰＵ１２上のソフトウェアが、処理部４６と非同期に動作する場合にも対応できる。これにより、処理部４６におけるコンテキストスイッチ実行中も、ＣＰＵ１２上の各アプリケーションは、処理の指示をＧＰＵコマンドバッファ３０に積み上げることができるため、処理の遅延を抑制することができる。

【００４３】

なお、実施例のＳＯＣ１１では、ＣＰＵ１２で実行される管理ソフトウェアが、コンテキストスイッチの初期設定を行う。例えば、管理ソフトウェアは、並列処理の対象となる複数のアプリケーションのそれぞれについて、ＧＰＵコンテキスト３２、ビデオエンコーダコンテキスト３４、ビデオパイプラインコンテキスト３６の領域をメインメモリ２２に確保する。また、管理ソフトウェアは、各アプリケーションのコンテキスト格納位置（アドレス等）をＣＳＤＭＡエンジン４８（ＣＳＤＭＡエンジン４０、ＣＳＤＭＡエンジン４２、ＣＳＤＭＡエンジン４４）に通知する。

【００４４】

また、管理ソフトウェアは、並列処理の対象となる複数のアプリケーションの実行順序を、処理部４６とＣＳＤＭＡエンジン４８に通知する。処理部４６とＣＳＤＭＡエンジン４８は、管理ソフトウェアからの通知や設定に基づいて、今回実行対象となるアプリケーションや次回実行対象となるアプリケーションを把握する。

【００４５】

図１には不図示だが、変形例として、ＣＰＵ１２にもコンテキストスイッチのためのＣＳＤＭＡエンジン４８を設けてもよい。この場合、ＣＰＵ１２は、処理部４６と同期して

動作してもよい。

【００４６】

図２は、図１の処理部４６（具体的にはＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８）とＣＳＤＭＡエンジン４８の詳細な構成を示すブロック図である。図２の処理部４６とＣＳＤＭＡエンジン４８の組は、図１のＧＰＵ１４とＣＳＤＭＡエンジン４０の組、ビデオエンコーダ１６とＣＳＤＭＡエンジン４２の組、ビデオパイプライン１８とＣＳＤＭＡエンジン４４の組のそれぞれに対応する。すなわち、図２の処理部４６の構成は、ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８の少なくとも１つに適用可能である。また、図２のＣＳＤＭＡエンジン４８の構成は、ＣＳＤＭＡエンジン４０、ＣＳＤＭＡエンジン４２、ＣＳＤＭＡエンジン４４の少なくとも１つに適用可能である。

【００４７】

本開示のブロック図において示される各ブロックは、ハードウェア的には、コンピュータのＣＰＵ・メモリをはじめとする素子や機械装置で実現でき、ソフトウェア的にはコンピュータプログラム等によって実現されるが、ここでは、それらの連携によって実現される機能ブロックを描いている。これらの機能ブロックはハードウェア、ソフトウェアの組合せによっていろいろなかたちで実現できることは、当業者には理解されるところである。

【００４８】

処理部４６は、第１機能回路７０、第２機能回路７１を備える。第１機能回路７０と第２機能回路７１は、実行対象のアプリケーションに関するデータ処理（例えば画像生成処理や圧縮伸長処理）を実行する。処理部４６は、ＣＰＵ１２上の管理ソフトウェアからの通知や設定等に基づいて、実行対象のアプリケーションを識別し、実行対象のアプリケーションに第１機能回路７０および第２機能回路７１を割り当てる。

【００４９】

処理部４６は、実行対象のアプリケーションに関する各種データをメインメモリ２２から読み出して、第１機能回路７０と第２機能回路７１に入力する。例えば、処理部４６としてのＧＰＵ１４は、実行対象のアプリケーションに対応するＧＰＵコマンドバッファ３０のキューから、当該アプリケーションに関するコマンドを読み出す。また、ＧＰＵ１４は、画像描画に必要な他のデータもメインメモリ２２から読み出す。

【００５０】

第１機能回路７０は、アプリケーションの実行中に更新されるデータであって、処理の再現に必要なデータであるコンテキストを保持する回路である。また、第１機能回路７０は、実行対象のアプリケーションに応じて、コンテキストの入れ替えが必要な回路である。第１機能回路７０は、ステート・マシン７２、レジスタ７３、ワークメモリ７４を含む。ステート・マシン７２は、ステート・レジスタを含み、処理部４６で実行中の各処理の現在状態を保持し、言い換えれば、状態遷移の状況を保持する。レジスタ７３は、設定および処理中のデータや処理の結果を保持する。ワークメモリ７４は、処理部４６のメモリの中で、処理に応じて更新される内部データやデスクリプタおよびマイクロコードを保持する領域である。

【００５１】

第２機能回路７１は、アプリケーションの実行中に更新されるデータであって、処理の再現に必要なデータであるコンテキストを保持しない回路である。また、第２機能回路７１は、初期化が不要な回路もしくは一括でのリセットが可能な回路であり、例えば演算処理等を行う回路である。第２機能回路７１は、ランダムロジック７５、演算器７６、ワークメモリ７７を含む。ランダムロジック７５は、ハードワイヤードによる機能を含み、初期化が不要もしくは一定の初期化が可能なフリップフロップ回路（ラッチ）を含む。演算器７６は、データパスおよび初期化が不要もしくは一定の初期化が可能なフリップフロップ回路（ラッチ）を含む。ワークメモリ７７は、処理部４６のメモリの中でワークメモリ７４以外の領域である。

【００５２】

ＣＳＤＭＡエンジン４８は、検出部８０、監視部８２、指示部８４、転送実行部８６を含む。検出部８０は、処理部４６における実行対象のアプリケーションの切替タイミング（以下「Ａｐｐ切替タイミング」とも呼ぶ。）を検出する。実施例の検出部８０は、垂直帰線期間の開始タイミングをＡｐｐ切替タイミングとして検出し、具体的には、ビデオタイミングジェネレータ２０から垂直帰線期間の開始タイミングが通知されたことをもってＡｐｐ切替タイミングであると判断する。

【００５３】

変形例として、検出部８０は、水平帰線期間の開始タイミングをＡｐｐ切替タイミングとして検出してもよい。具体的には、検出部８０は、ビデオタイミングジェネレータ２０から水平帰線期間の開始タイミングが通知されたことをもってＡｐｐ切替タイミングであると判断してもよい。別の変形例として、ＣＰＵ１２上の管理ソフトウェアが、Ａｐｐ切替タイミングを処理部４６およびＣＳＤＭＡエンジン４８に通知してもよい。検出部８０は、ＣＰＵ１２上の管理ソフトウェアからの通知をもとにＡｐｐ切替タイミングを検出してもよい。

【００５４】

監視部８２は、Ａｐｐ切替タイミングが検出された場合、処理部４６におけるアプリケーションに関する処理の実行状態を監視する。処理部４６におけるアプリケーションに関する処理は小さい粒度の複数の内部処理を含み、複数の内部処理が並列して実行される。監視部８２は、個々の内部処理の実行状態（例えば個々の内部処理が完了したか否か）を確認する。

【００５５】

指示部８４は、アプリケーションに関する処理の停止と開始（再開）を処理部４６に指示する。指示部８４は、Ａｐｐ切替タイミングが検出された場合に、アプリケーションに関する処理の停止を処理部４６に指示してもよい。処理部４６は、ＣＳＤＭＡエンジン４８の指示部８４からの指示をもとにＡｐｐ切替タイミングを検出してもよく、ＣＰＵ１２上の管理ソフトウェアまたはビデオタイミングジェネレータ２０からの通知をもとにＡｐｐ切替タイミングを検出してもよい。なお、監視部８２と指示部８４はオプショナルな構成であり、ＣＳＤＭＡエンジン４８は、監視部８２と指示部８４の一方または両方を含まない構成であってもよい。

【００５６】

転送実行部８６は、Ａｐｐ切替タイミングが検出された場合、コンテキスト転送処理を実行する。具体的には、転送実行部８６は、コンテキスト転送処理として、複数のアプリケーションを管理するソフトウェア（実施例ではＣＰＵ１２上の管理ソフトウェア）の処理を介さずに、処理部４６で実行されていたアプリケーションのコンテキストを処理部４６（第１機能回路７０）からメインメモリ２２（当該アプリケーションのコンテキスト記憶領域）へ退避させる。

【００５７】

また、転送実行部８６は、コンテキスト転送処理として、複数のアプリケーションを管理するソフトウェア（実施例ではＣＰＵ１２上の管理ソフトウェア）の処理を介さずに、処理部４６で次に実行されるアプリケーションのコンテキストをメインメモリ２２（当該アプリケーションのコンテキスト記憶領域）から読み出す。転送実行部８６は、読み出したコンテキストを処理部４６（第１機能回路７０）に設定する。指示部８４は、転送実行部８６によるコンテキスト設定処理の完了を検出した場合、アプリケーションに関する処理の開始（再開）を処理部４６に指示してもよい。

【００５８】

転送実行部８６による転送の対象となるコンテキストは、処理部４６におけるコンテキストスイッチに伴い中断された時点のアプリケーションの処理（上述の個々の内部処理）の再現に必要なデータである。実施例では、転送実行部８６による転送の対象となるコンテキストは、処理部４６の第１機能回路７０に保持されたコンテキストの全てであって、ソフトウェア（例えばＣＰＵ１２上のソフトウェアおよび処理部４６上のソフトウェア）

からアクセスできないコンテキストを含む。ソフトウェアからアクセスできないコンテキストは、例えば、処理部４６のステート・マシン７２に保持されるコンテキストであってもよく、アプリケーションの処理に含まれる個々の内部処理の状態遷移の状況を示すコンテキストであってもよい。

【００５９】

図３は、処理部４６（例えばＧＰＵ１４）におけるビデオ描画処理を時系列に示す。同図は、４つのアプリケーション（Ａｐｐ　Ａ、Ａｐｐ　Ｂ等）に関するビデオ描画処理を時分割で実行する例を示している。本実施例において、各アプリケーションは、６０ｆｐｓ（frames per second）で画像を生成する必要がある。処理部４６は、４つのアプリケーションの画像を各々１／２４０秒で生成することにより、１つのアプリケーションあたり６０ｆｐｓのフレームレートを実現する。

【００６０】

しかし、各アプリケーションの処理時間は描画内容に基づいて常に変動しており、必ずしも１／２４０秒以内で処理を完了できるとは限らない。このようなとき、既述したように、実施例では、垂直帰線期間の開始タイミング（図３のタイミング９０）がＡｐｐ切替タイミング（言い換えれば画像書き込み先のフレームバッファを切り替えるフリップタイミング）として検出される。ここで、画像の生成がまだ完了していない場合も、次のアプリケーションへ処理部４６をあけわたすことで、各アプリケーションが処理部４６を使用できる時間を均等にできる。なお、タイミング９０までに画像の生成ができなかった場合、コマ落ちとなり、該当アプリケーションの前画像が繰り返し後段処理に渡される。

【００６１】

図３の期間９２は、垂直帰線期間に相当し、ＣＳＤＭＡエンジン４８を用いることで、処理の再現に必要な全コンテキストの退避・設定をこの期間以内で高速に実行する。このようにして、矢印で示す画像生成期間とコンテキストスイッチを含む期間９２（垂直帰線期間）との合計が１／２４０秒以内になるように制御される。

【００６２】

なお、アプリケーションが垂直帰線期間の開始タイミングより早く画像の生成を完了できたとき、アプリケーションはフレームバッファを切り替えるフリップの実行が可能であることをＣＰＵ１２上の管理ソフトウェアへ通知してもよい。そして、ＣＰＵ１２上の管理ソフトウェアは、垂直帰線期間の開始タイミングに代わり、Ａｐｐ切替タイミングを自ら処理部４６へ通知してもよい。これにより該当アプリケーションの後段処理や、後続のアプリケーションは前倒しで処理を開始し、遅延の削減や、処理部の利用効率を高めることができる。

【００６３】

なお、変形例では、水平帰線期間の開始タイミングがＡｐｐ切替タイミングとして検出されてもよく、水平帰線期間内にコンテキストスイッチを行ってもよい。

【００６４】

図４は、ビデオタイミングの例を示す。同図は、垂直帰線期間９４（Ｖｂｌａｎｋ）と水平帰線期間９６（Ｈｂｌａｎｋ）を示している。なお、サーバがビデオを描画してそのビデオをクライアントへ配信する場合、ビデオタイミングを使用する外部ディスプレイがサーバに接続されないことがある。その場合も、従来のアプリケーションとの互換維持のため、ビデオタイミングが生成されることがあり、実施例ではそのビデオタイミングを利用してＡｐｐ切替タイミングを検出する。

【００６５】

垂直帰線期間および水平帰線期間には、ＧＰＵ１４やビデオエンコーダ１６、ビデオパイプライン１８が処理を一時停止することもできる。実施例では、短時間の垂直帰線期間または水平帰線期間の中でコンテキストスイッチを実行することで、コンテキストスイッチのオーバーヘッドを隠蔽することができる。

【００６６】

次に、コンテキストの転送経路について説明する。

図５は、ＳＯＣ１１の構成を示す。ＳＯＣ１１内の各回路は、通常、バスファブリック１００およびメモリインタフェース１０２を介してメインメモリ２２にアクセスする。ＳＯＣ１１（すなわち半導体チップ）には、機能が正しく動作するか否かをテストするためのＤＦＴ（Design For Test）機能（ＤＦＴ制御回路１０４等）が内蔵されている。また、ＢＩＳＴ（Built-In Self Test）は、自己診断機能を持ち、ロジック回路用とメモリ回路用とが設けられる。スキャンチェーン１０６は、ＳＯＣ１１内の各回路内部へ網羅的に接続され、ＢＩＳＴやチップ外部の指示に基づいて各回路をテストするために使用される。スキャンチェーン１０６を用いることで、ＳＯＣ１１内部の各状態値の設定および読み出しができる。

【００６７】

図６も、ＳＯＣ１１の構成を示す。同図に示すように、ＣＳＤＭＡエンジン４８は、通常のデータ転送用のバス（バスファブリック１００）と、パス１１４とを使用して、コンテキストの退避および設定を行うことができる。パス１１４は、ソフトウェアからはアクセスできないコンテキストを保持する回路にアクセスするためのパスである。しかし、この構成では、コンテキストの転送中に、他の回路（ＣＰＵ１２等）とバスファブリック１００を共有するため、通常のデータ転送が阻害されてしまうというデメリットがある。

【００６８】

図７も、ＳＯＣ１１の構成を示す。同図に示すように、ＣＳＤＭＡエンジン４８は、コンテキスト転送用の専用バス（専用経路１０８）と、パス１１４とを使用して、コンテキストの退避および設定を行うこともできる。しかし、この構成では、専用経路１０８を新たにＳＯＣ１１に実装する必要があり、ＳＯＣ１１のコストが増大するというデメリットがある。

【００６９】

図８も、ＳＯＣ１１の構成を示す。同図に示すように、実施例のＣＳＤＭＡエンジン４８は、ＤＦＴ用の回路（スキャンチェーン１０６を含む）およびＢＩＳＴ用の回路（ＢＩＳＴ回路１１０）を使用して、コンテキストの退避および設定を行う。例えば、ＣＳＤＭＡエンジン４８は、スキャンチェーン１０６を介して第１機能回路７０から今回実行されたプリケーションに関するコンテキストを読み出し、読み出したコンテキストをＢＩＳＴ回路１１０を介してメインメモリ２２に保存する。また、ＣＳＤＭＡエンジン４８は、次回実行されるアプリケーションに関するコンテキストをＢＩＳＴ回路１１０を介してメインメモリ２２から読み出し、読み出したコンテキストをスキャンチェーン１０６を介して第１機能回路７０に設定する。

【００７０】

このようにスキャンチェーン１０６を使用することで、ソフトウェアがアクセスするパスがない内部回路のコンテキストにもアクセスすることができ、処理の再現に必要なコンテキストの退避と設定が可能になる。また、バスファブリック１００、専用経路１０８、パス１１４のいずれも新たに設ける必要がなく、低コストでコンテキストの転送を実現できる。さらにまた、コンテキストの転送によりＳＯＣ１１における通常のデータ転送を阻害してしまうことを回避できる。

【００７１】

次に、コンテキストスイッチの開始タイミングについて説明する。

ＣＳＤＭＡエンジン４８の転送実行部８６は、処理部４６の第１機能回路７０（ステート・マシン７２等）に保持されたアプリケーションのコンテキストをメインメモリ２２に退避させる。処理部４６からメインメモリ２２に退避されるアプリケーションのコンテキスト、および、メインメモリ２２から処理部４６に設定されるアプリケーションコンテキストは、ステート・マシン７２等に保持されたデータであって、コンテキストスイッチ開始時点で処理部４６において未完了の処理（小さい粒度の内部処理）の現在状態を示すデータを含む。

【００７２】

処理部４６の第１機能回路７０および第２機能回路７１は、Ａｐｐ切替タイミングが検

出された場合、それまで実行対象であったアプリケーションの内部処理の状態にかかわらず、言い換えれば、内部処理が未完了であっても、即時に内部処理を停止する。ＣＳＤＭＡエンジン４８の転送実行部８６は、Ａｐｐ切替タイミングが検出された場合、それまで実行対象であったアプリケーションの内部処理の状態にかかわらず、言い換えれば、内部処理が未完了であっても、コンテキストの転送処理を開始する。

【００７３】

図９Ａは、第１実施例におけるコンテキストスイッチ時の内部処理の例を示す。同図は、横方向が時間経過軸であり、１つの矢印は、１つのアプリケーションに関する処理に含まれる小さな粒度の内部処理を示す。各矢印の左端が内部処理の開始タイミングを示し、右端が内部処理の終了タイミングを示している。

【００７４】

図９Ａに示すように、実施例の処理部４６は、コンテキストスイッチ実行指示を受け付けた時点（すなわちＡｐｐ切替タイミングを検出した時点）で、即時に全ての内部処理を停止し、コンテキストスイッチを開始する。また、その時点で未完了（すなわちオンザフライ・アウトスタンディング）の内部処理（図中の破線）も即時に中断し、その時点のコンテキストをメインメモリ２２に保存する。なお、図中の一点鎖線で示す内部処理は、コンテキストスイッチ実行指示の受付後に開始予定のものであるため、開始されない。ＣＳＤＭＡエンジン４８の転送実行部８６は、コンテキストスイッチ実行指示を受け付けた時点（すなわちＡｐｐ切替タイミングを検出した時点）で、即時にコンテキストの転送処理（退避および設定）を開始する。

【００７５】

実施例のＳＯＣ１１では、処理部４６における各内部処理の状態であり、すなわち、ソフトウェアからは確認できないデータ（例えばステート・マシン７２に保持されるデータ等）を含むコンテキストを退避させる。これにより、コンテキストスイッチ実行指示受付時点で実行途中だった内部処理は、次回の実行時に、その状態を復元して再開することができる。したがって、実施例のＳＯＣ１１では、コンテキストスイッチ実行指示の受付時点（Ａｐｐ切替タイミングの検出時点）で即時にコンテキストスイッチを開始でき、処理の遅延を抑制することができる。

【００７６】

変形例として、ＣＳＤＭＡエンジン４８の転送実行部８６は、Ａｐｐ切替タイミングが検出された場合、処理部４６におけるアプリケーションの内部処理が完了したことが監視部８２により確認されるまで待機し、処理部４６におけるアプリケーションの内部処理が完了したことが確認された後にコンテキストの転送処理を開始してもよい。

【００７７】

図９Ｂは、変形例におけるコンテキストスイッチ時の内部処理の例を示す。同図でも、コンテキストスイッチ実行指示時に開始済だが未完了（すなわちオンザフライ・アウトスタンディング）の内部処理を破線で示している。本変形例では、コンテキストスイッチ実行指示時に開始済だが未完了の内部処理が終了するまで待ってからコンテキストスイッチを開始する。

【００７８】

コンテキストスイッチ実行指示時に開始済だが未完了の処理を中断させずに完了させることにより、コンテキストスイッチ対象ブロック外との処理連携が求められる場合や、リアルタイム性が求められる場合において、処理破綻（不整合の発生等）を防止できる。また、転送対象となるコンテキストのデータ量を低減することができる。

【００７９】

＜第２実施例＞

本実施例に関して、これまでの実施例と相違する点を中心に以下説明し、共通する点の説明を省略する。本変形例の構成要素のうちこれまでの実施例の構成要素と同一または対応する構成要素には同一の符号を付して説明する。

【００８０】

図１０は、第２実施例の処理部４６の構成を示す。図１０の処理部４６とＣＳＤＭＡエンジン４８の組は、図１のＧＰＵ１４とＣＳＤＭＡエンジン４０の組、ビデオエンコーダ１６とＣＳＤＭＡエンジン４２の組、ビデオパイプライン１８とＣＳＤＭＡエンジン４４の組のそれぞれに対応する。すなわち、図１０の処理部４６の構成は、ＧＰＵ１４、ビデオエンコーダ１６、ビデオパイプライン１８の少なくとも１つに適用可能である。また、図１０のＣＳＤＭＡエンジン４８の構成は、ＣＳＤＭＡエンジン４０、ＣＳＤＭＡエンジン４２、ＣＳＤＭＡエンジン４４の少なくとも１つに適用可能である。

【００８１】

第２実施例の処理部４６は、アプリケーションのコンテキストを保持する第１機能回路７０を複数含む。図１０の処理部４６は、２つの第１機能回路７０（第１機能回路７０ａと第１機能回路７０ｂ）を備えるが、３つ以上の第１機能回路７０を備えてもよい。

【００８２】

第２実施例のＣＳＤＭＡエンジン４８の転送実行部８６は、処理部４６が或る第１機能回路７０に保持されたアプリケーションのコンテキストを用いて当該アプリケーションの処理を実行する間に、別の第１機能回路７０に保持された別のアプリケーションのコンテキストを対象としてコンテキスト転送処理を実行する。

【００８３】

例えば、処理部４６が第１機能回路７０ａに保持されたＡｐｐ　Ａのコンテキストを用いてＡｐｐ　Ａの処理を実行する間に、転送実行部８６は、（１）第１機能回路７０ｂに保持されたＡｐｐ　Ｄ（すなわち前回実行したＡｐｐ）のコンテキストをメインメモリ２２に退避させる。それとともに、転送実行部８６は、（２）メインメモリ２２に保持されたＡｐｐ　Ｂ（すなわち次に実行対象となるＡｐｐ）のコンテキストを読み出して、第１機能回路７０ｂに設定する。

【００８４】

処理部４６は、第１機能回路７０ａに保持されたアプリケーションのコンテキストを用いて当該アプリケーションの処理を実行中にＡｐｐ切替タイミングを検出すると、即時に当該アプリケーションの処理を中止する。そして、処理部４６は、第１機能回路７０ｂに保持された別のアプリケーションのコンテキストを用いて当該別のアプリケーションの処理を開始する。これにより、コンテキストスイッチに伴う待機時間（すなわちコンテキストの退避と設定を待つ時間）をほぼゼロにすることができる。

【００８５】

＜第３実施例＞

本実施例に関して、これまでの実施例と相違する点を中心に以下説明し、共通する点の説明を省略する。本変形例の構成要素のうちこれまでの実施例の構成要素と同一または対応する構成要素には同一の符号を付して説明する。

【００８６】

第３実施例の処理部４６の構成は、図１０に示した第２実施例の処理部４６の構成と同じである。処理部４６は、第１機能回路７０ａに保持された第１のアプリケーション（例えばＡｐｐ　Ａ）のコンテキストを用いて第１のアプリケーションに関する処理を実行する。また、処理部４６は、第１機能回路７０ｂに保持された第２のアプリケーション（例えばＡｐｐ　Ｂ）のコンテキストを用いて第２のアプリケーションに関する処理を実行する。

【００８７】

処理部４６は、Ａｐｐ切替タイミングを検出すると、ＣＰＵ１２上の管理ソフトウェアによる事前設定等をもとに、次に実行すべきアプリケーションを識別する。ここでは、処理部４６における実行対象が第１のアプリケーション（例えばＡｐｐ　Ａ）から第２のアプリケーション（例えばＡｐｐ　Ｂ）に切り替わることとする。処理部４６は、第１のアプリケーションに関する処理のうち第２のアプリケーションに関する処理と混流（言い換えれば並列実行または同時実行）不可能な処理が完了後に、第１のアプリケーションに関する処理のうち第２のアプリケーションに関する処理と混流可能な処理と、第２のアプリ

ケーションに関する処理とを並列実行する。
【００８８】
図１１は、第３実施例におけるコンテキストスイッチ時の内部処理の例を示す。同図は、上記第１のアプリケーション（切替前のアプリケーション）の内部処理を示す。同図の破線および二点鎖線は、コンテキストスイッチ実行指示時に開始済だが未完了（オンザフライ・アウトスタンディング）の内部処理を示す。このうち破線は、第２のアプリケーションに関する処理と混流不可能な内部処理を示し、二点鎖線は、第２のアプリケーションに関する処理と混流可能な内部処理を示している。
【００８９】
混流可能な処理は、例えばデータパスや演算器を使用する内部処理であって、時間的に前後の処理内容に対して依存関係がなく、かつ、空間的に回路として周辺に接続された機能の処理内容に対して依存関係がない内部処理である。混流不可能な処理は、例えば、上記の依存関係を有する内部処理であり、また、第２のアプリケーションに関する処理と並列実行すると破綻する可能性がある内部処理である。処理部４６は、各内部処理が、他のアプリケーションの内部処理と混流可能か否かを示すデータを予め記憶してもよく、また、ＣＰＵ１２上の管理ソフトウェアがそのデータを処理部４６に予め格納してもよい。
【００９０】
処理部４６は、コンテキストスイッチ実行指示を受け付けても、第１のアプリケーションに関するオンザフライ・アウトスタンディングの内部処理を継続する。処理部４６は、オンザフライ・アウトスタンディングの内部処理のうち第２のアプリケーション（言い換えれば他のアプリケーション）と混流不可能な処理（図１１の破線）が終了するまで待ってから、コンテキストスイッチを開始する。言い換えれば、処理部４６は、オンザフライ・アウトスタンディングの内部処理のうち第２のアプリケーションと混流不可能な処理が全て終了した場合、第２のアプリケーションと混流可能な処理（図１１の二点鎖線）が終了したか否かにかかわらず、コンテキストスイッチを開始する。
【００９１】
コンテキストスイッチ実行後、処理部４６（例えば第１機能回路７０ａ）は、第１のアプリケーションに関するオンザフライ・アウトスタンディングの内部処理のうち第２のアプリケーションと混流可能な内部処理を継続して実行する。それと並行して、処理部４６（例えば第１機能回路７０ｂ）は、第２のアプリケーションに関する内部処理を実行する。混流可能な内部処理が全て終了すると、第１のアプリケーションのコンテキストを保持する回路（例えば第１機能回路７０ａ）には実行結果の状態が記録される。
【００９２】
ＣＳＤＭＡエンジン４８の転送実行部８６は、第１のアプリケーションに関するオンザフライ・アウトスタンディングの内部処理が全て完了したことが監視部８２により検知されると、第１のアプリケーションのコンテキストを保持する回路（例えば第１機能回路７０ａ）からメインメモリ２２に第１のアプリケーションのコンテキストを退避させる。また、転送実行部８６は、処理部４６において次に実行予定の第３のアプリケーションのコンテキストをメインメモリ２２から上記回路（例えば第１機能回路７０ａ）に設定する。
【００９３】
第３実施例のＳＯＣ１１によると、第１のアプリケーションに関する処理の破綻を防止しつつ、第２のアプリケーションに関する処理を早期に開始することができ、処理部４６における処理の遅延を抑制することができる。
【００９４】
図１２Ａ、図１２Ｂ、図１２Ｃは、コンテキストスイッチにおける処理部４６とＣＳＤＭＡエンジン４８の動作を示す。図１２Ａは、第１実施例の処理部４６の動作（即時コンテキストスイッチ）を示している。処理部４６は、前Ａｐｐ処理と次Ａｐｐ処理を実行し、ＣＳＤＭＡエンジン４８は、前Ａｐｐのコンテキスト退避処理と次Ａｐｐのコンテキスト設定処理を実行する。図１２Ｂは、第１実施例の変形例に記載した処理部４６の動作（全ての内部処理完了後のコンテキストスイッチ）を示している。前Ａｐｐ処理の終了を待

つ時間だけ、コンテキストの退避処理の開始が遅れることになる。

【００９５】

第２実施例で記載したようにコンテキストを保持する第１機能回路７０が多重化される場合、図１２Ａと図１２Ｂのいずれに示すコンテキストスイッチ方法でも、一方のコンテキスト保持回路を用いてアプリケーションを実行中に、他方のコンテキスト保持回路に次に実行予定のアプリケーションのコンテキストを事前に設定することができる。これにより、コンテキストの退避と設定の完了を待つ時間（図１２Ａと図１２Ｂの期間１１２）をほぼゼロにすることができる。

【００９６】

図１２Ｃは、第３実施例の処理部４６の動作（混流不可能な内部処理完了後のコンテキストスイッチ）を示している。図１２Ｃの「コンテキストスイッチ開始・終了」は、次Ａｐｐを実行するために、参照先をそれまでとは異なるコンテキスト保持回路に切り替えることを示している。例えば、参照先を第１機能回路７０ａから第１機能回路７０ｂに切り替えることを示している。既述したように、前Ａｐｐの混流可能な内部処理が終了すると、ＣＳＤＭＡエンジン４８は、前Ａｐｐのコンテキスト保持回路（例えば第１機能回路７０ａ）からメインメモリ２２へ前Ａｐｐのコンテキストを退避させる。

【００９７】

＜第４実施例＞

本実施例に関して、これまでの実施例と相違する点を中心に以下説明し、共通する点の説明を省略する。本変形例の構成要素のうちこれまでの実施例の構成要素と同一または対応する構成要素には同一の符号を付して説明する。

【００９８】

図１３は、第４実施例の処理部４６の構成を示す。第４実施例の処理部４６は、図１０に示した第３実施例の構成に加えて、複数のアプリケーション間で混流不可能な内部処理を実行する機能ブロックを複数備える。複数のアプリケーション間で混流不可能な内部処理を実行する機能ブロックは、例えば、（１）「時間的に前後の処理内容や、空間的に回路として周辺に接続された機能の処理内容」に対して依存関係がある処理を実行する回路であってもよい。または、（２）各アプリケーションの別設定を前提とした内部処理が投入されると、両アプリケーションの設定が共通でないことによる内処理の破綻が起こる回路であってもよい。

【００９９】

図１３に示す処理部４６は、複数のアプリケーション間で混流不可能な内部処理を実行する機能ブロックの例として、ランダムロジック７５を複数備える。図１３の処理部４６は、２つのランダムロジック７５（ランダムロジック７５ａとランダムロジック７５ｂ）を備えるが、３つ以上のランダムロジック７５を備えてもよい。

【０１００】

Ａｐｐ切替タイミングが検出され、処理部４６における処理対象を第１のアプリケーション（例えばＡｐｐ　Ａ）から第２のアプリケーション（例えばＡｐｐ　Ｂ）に切り替えるべき際、処理部４６は、第１のアプリケーションに関するオンザフライ・アウトスタンディングの内部処理のうち第２のアプリケーションに関する処理と混流不可能な内部処理を、一方のランダムロジック７５（例えばランダムロジック７５ａ）を用いて継続実行する。また、処理部４６は、上記混流不可能な内部処理の完了を待たず、即時に、他方のランダムロジック７５（例えばランダムロジック７５ｂ）を用いて、第２のアプリケーションに関する処理を開始する。

【０１０１】

第４実施例のＳＯＣ１１によると、第１のアプリケーションに関するオンザフライ・アウトスタンディングの内部処理のうち第２のアプリケーションに関する処理と混流不可能な内部処理の完了を待つ必要がない。これにより、コンテキストスイッチ実行指示時に、第２のアプリケーションに関する処理を早期に開始することができ、コンテキストスイッチを一層高速に実現できる。

【０１０２】
第４実施例の変形例を説明する。処理部４６は、複数の多重化された機能回路に対して複数のアプリケーションを同時に割り当ててもよく、処理部４６は、複数のアプリケーションの内部処理を並列実行してもよい。ここで、機能回路の多重化が２重であり、並列処理対象のアプリケーションが４つ（例えばＡｐｐ　Ａ～Ｄ）の場合、選択された２つのアプリケーション（例えばＡｐｐ　ＡとＡｐｐ　Ｂ）を２つの多重化された機能回路に割り当ててもよい。コンテキストスイッチ時には、残りの２つのアプリケーション（例えばＡｐｐ　ＣとＡｐｐ　Ｄ）を新たな実行対象として、それまで実行した２つのアプリケーション（例えばＡｐｐ　ＡとＡｐｐ　Ｂ）と入れ替えてもよい。
【０１０３】
例えば、図１３に示す構成において、Ａｐｐ　Ａの内部処理とＡｐｐ　Ｂの内部処理を並列実行する場合、処理部４６は、Ａｐｐ　Ａのコンテキストを第１機能回路７０ａに保持させ、Ａｐｐ　Ｂのコンテキストを第１機能回路７０ｂに保持させてもよい。また、処理部４６は、Ａｐｐ　Ａに関する内部処理のうちＡｐｐ　Ｂに関する内部処理と混流不可能な内部処理をランダムロジック７５ａに割り当て、Ａｐｐ　Ｂに関する内部処理のうちＡｐｐ　Ａに関する内部処理と混流不可能な内部処理をランダムロジック７５ｂに割り当ててもよい。また、処理部４６は、Ａｐｐ　Ａに関する内部処理のうちＡｐｐ　Ｂに関する内部処理と混流可能な内部処理と、Ａｐｐ　Ｂに関する内部処理のうちＡｐｐ　Ａに関する内部処理と混流可能な内部処理の両方を共通の演算器７６とワークメモリ７７に割り当ててもよい。
【０１０４】
この変形例の構成によると、処理部４６が複数のアプリケーションを時分割に実行するのではなく同時に実行するため、処理の遅延を一層抑制できる。また、コンテキストスイッチの発生回数も抑制できる。また、処理部４６が有する回路のアクティブ率（言い換えれば活性化率）が向上し、データ処理の性能を高めることができる。
【０１０５】
＜第５実施例＞
本実施例に関して、これまでの実施例と相違する点を中心に以下説明し、共通する点の説明を省略する。本変形例の構成要素のうちこれまでの実施例の構成要素と同一または対応する構成要素には同一の符号を付して説明する。
【０１０６】
図１４は、第５実施例の処理部４６の構成を示す。第５実施例の処理部４６では、コンテキストを保持する第１機能回路７０と、コンテキストを保持しない第２機能回路７１の両方を多重化する。図１４の処理部４６は、４つの第１機能回路７０と４つの第２機能回路７１を備えるが、第１機能回路７０と第２機能回路７１の多重化は４重に制限されない。
【０１０７】
本実施例の構成において、各アプリケーションの性能要件が異なる場合、処理部４６は、１つのアプリケーションに対して何個の第１機能回路７０および第２機能回路７１を割り当てるかをコンテキストスイッチのたびに決定する。そして、処理部４６は、実行対象となる各アプリケーションに１つ以上の第１機能回路７０と１つ以上の第２機能回路７１を割り当てる。各アプリケーションの性能要件、または、各アプリケーションが必要とする第１機能回路７０と第２機能回路７１の個数は、ＣＰＵ１２上の管理ソフトウェアが処理部４６に設定してもよい。
【０１０８】
図１５Ａと図１５Ｂは、アプリケーションの割当例を示す。図１５Ａに示すように、処理部４６は、４つのアプリケーション（例えばＡｐｐ　Ａ～Ｄ）のそれぞれを、１つの第１機能回路７０と１つの第２機能回路７１の組に割り当ててもよい。また、Ａｐｐ　Ｅに関するデータ処理に、Ａｐｐ　Ａに関するデータ処理の４倍の性能が必要である場合、図１５Ｂに示すように、処理部４６は、Ａｐｐ　Ｅに対して、４つの第２機能回路７１と１つの第１機能回路７０を割り当ててもよい。Ａｐｐ　Ｅの実行時、３つの第１機能回路７０は未使用であってもよい。

【０１０９】
処理部４６は、図１５Ａに示す割当での処理と、図１５Ｂに示す割当での処理とを時分割で実行してもよい。この場合、第１実施例から第４実施例で説明したコンテキストスイッチ方法のいずれかを適用することで、図１５Ａの状態と図１５Ｂの状態との間での遷移を高速に実現できる。
【０１１０】
なお、多重化した第１機能回路７０の間で、現在稼働中か否か（コンテキストを保持するＡｐｐが実行中か否か）、また、どのようなコンテキストか（Ａｐｐの識別情報等）を共有するインタフェースを設けてもよい。また、処理部４６に接続されたＣＳＤＭＡエンジン４８が、処理部４６に代わって、複数の第１機能回路７０を統括し、各第１機能回路７０にどのＡｐｐのコンテキストが保持されるかを管理してもよい。
【０１１１】
また、図１５Ａの状態と図１５Ｂの状態とを切り替える場合、全ての第１機能回路７０のコンテキストスイッチを同時に実行する必要がある。このとき、第１実施例から第４実施例で説明したコンテキストスイッチ方法のいずれかを各Ａｐｐについて実行してもよい。そして、全てのＡｐｐが揃ったタイミング（例えば全てのＡｐｐの内部処理が完了したタイミング）で、コンテキストスイッチを実行してもよい。なお、図１５Ａに示す状態でＡｐｐ　Ａのみを別のＡｐｐ（例えばＡｐｐ　Ｆ）に切り替える場合で、かつ、Ａｐｐ　Ｆの性能要件がＡｐｐ　Ａの性能要件以下であれば、Ａｐｐ　Ａに対応する第１機能回路７０と第２機能回路７１のみにおいてコンテキストスイッチを実行してもよい。
【０１１２】
なお、第１機能回路７０および第２機能回路７１には、図１５Ａ、図１５Ｂに示した態様以外のアプリケーション割当の態様が可能であることはもちろんである。例えば、処理部４６では、２つまたは３つの第２機能回路７１を占有するアプリケーションが、他のアプリケーションと同時に実行されてもよい。
【０１１３】
上記の第２実施例～第５実施例では、コンテキストを保持する第１機能回路７０を多重化した。この構成によると、アプリケーションの処理中にオーバーラップして、コンテキストの退避と設定が可能になる。これにより、ＣＳＤＭＡエンジン４８およびデータを入出力する系（例えばバスファブリック１００、スキャンチェーン１０６、専用経路１０８、パス１１４）の転送能力を簡易化できる。
【０１１４】
例えば、図３に示したビデオ描画処理において、１／２４０秒毎（＝期間４．２ミリ秒）に各アプリケーションが動作するとき、垂直帰線期間が０．５ミリ秒、各アプリケーションの処理時間が３．７ミリ秒であったとする。このとき、多重化を行わない構成では、コンテキストスイッチを０．５ミリ秒未満で実行する必要がある。一方、多重化を行った可能性では、３．７ミリ秒未満で実行すればよい。ここで、コンテキストのデータ量が５０ＭＢであったとき、垂直帰線期間内に転送するには、１００ＧＢ／秒以上の転送能力が必要になる。一方、アプリケーション処理時間内の転送であれば、１３．５ＧＢ／秒以上の転送能力があればよい。
【０１１５】
以上、本開示を第１実施例～第５実施例をもとに説明した。これらの実施例は例示であり、各構成要素あるいは各処理プロセスの組合せにいろいろな変形例が可能なこと、またそうした変形例も本開示の範囲にあることは当業者に理解されるところである。
【０１１６】
上記実施例ではコンテキストの退避先をメインメモリ２２としたが、コンテキストの退避先となるメモリ（言い換えればコンテキスト保存するメモリ）は、メインメモリ２２とは異なるメモリであってもよい。また、コンテキストの退避先となるメモリのデータ保持特性は、揮発性であってもよく、不揮発性であってもよい。
【０１１７】

コンテキストの退避先となるメモリが不揮発性である場合、サスペンド・ハイバネーションなどの、処理部４６の一時停止・再開において当該メモリへの電源供給がカットされても、当該メモリに退避されたコンテキストは保持される。これにより、上記実施例に記載の高速なコンテキストスイッチを一層有効に利用できる。

【０１１８】

ＳＯＣ１１において、未使用の処理部４６に対する電源供給を個別にカットして低電力化を図る場合に、処理完了の区切りを待つ必要があったり、または、コンテキストの退避・設定の時間が長いと実用に耐えない可能性がある。上記実施例に記載の高速コンテキストスイッチを用いると、低電力化を図る場合にも間欠動作が可能となり、電力効率を高めることができる。

【０１１９】

また、上記実施例では、ＣＰＵ１２、処理部４６（ＧＰＵ１４、ビデオエンコーダ１６等）、ＣＳＤＭＡエンジン４８は、１つのハードウェア（ＳＯＣ１１）上に実装された。変形例として、これらの機能ブロックは、複数のハードウェア上に分散して実装されてもよい。例えば、各実施例のＣＳＤＭＡエンジン４８は、処理部４６が実装された装置から独立したデータ転送装置として実現されてもよい。

【０１２０】

上述した実施例および変形例の任意の組み合わせもまた本開示の実施の形態として有用である。組み合わせによって生じる新たな実施の形態は、組み合わされる実施例および変形例それぞれの効果をあわせもつ。また、請求項に記載の各構成要件が果たすべき機能は、実施例および変形例において示された各構成要素の単体もしくはそれらの連携によって実現されることも当業者には理解されるところである。

【産業上の利用可能性】

【０１２１】

本開示の技術は、データを処理するシステムまたは装置に適用することができる。

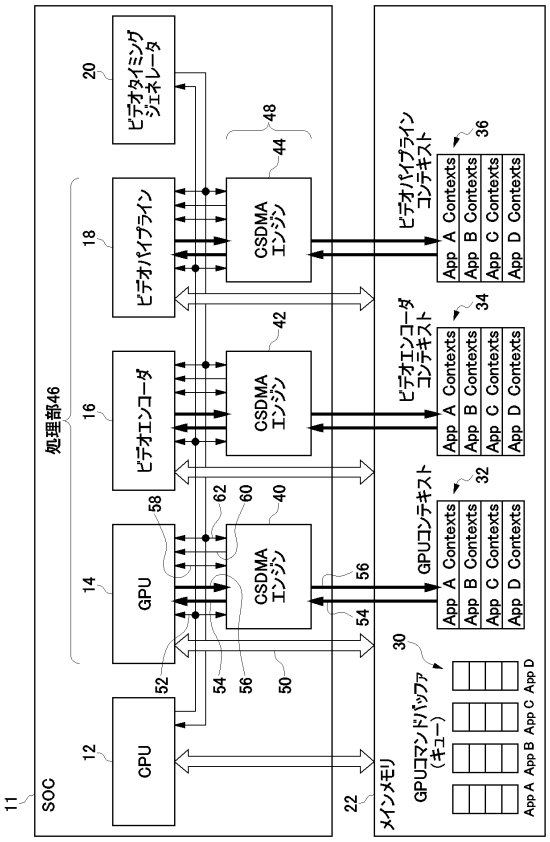
【符号の説明】

【０１２２】

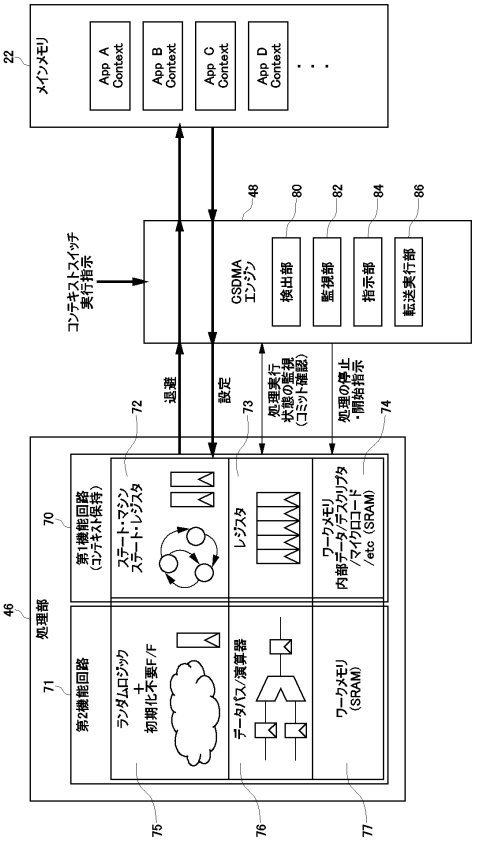
１０　コンピュータ、　１１　ＳＯＣ、　４６　処理部、　４８　ＣＳＤＭＡエンジン、８０　検出部、　８６　転送実行部、　１０６　スキャンチェーン。

【図面】

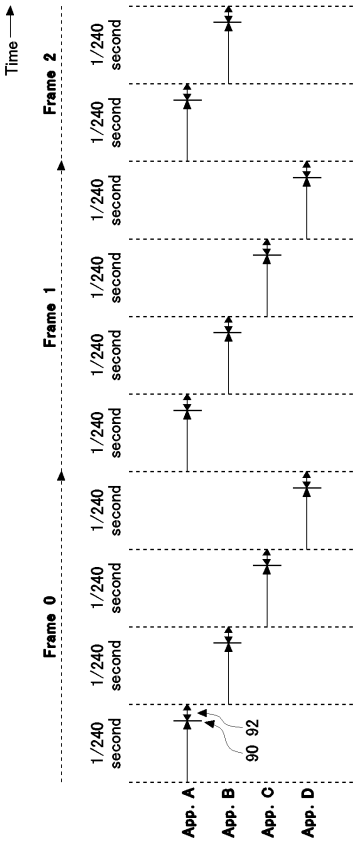
【図１】


SOC
処理部46
GPU
CPU
ビデオエンコーダ
ビデオパイプライン
ビデオタイミングジェネレータ
CSDMAエンジン
GPUコンテキスト
ビデオエンコーダコンテキスト
ビデオパイプラインコンテキスト
App A Contexts
App B Contexts
App C Contexts
App D Contexts
GPUコマンドバッファ（キュー）
App A App B App C App D
メインメモリ
コンピュータ 10

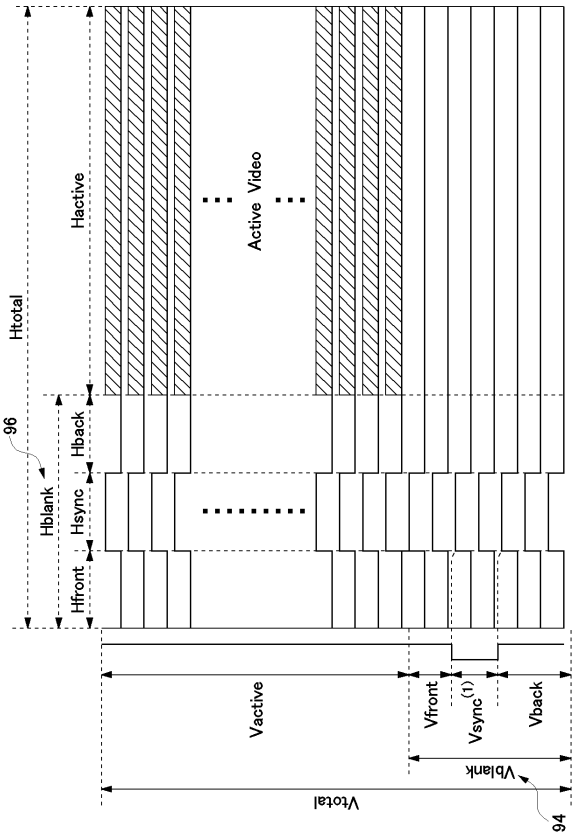
【図２】


メインメモリ
App A Context
App B Context
App C Context
App D Context
コンテキストスイッチ実行指示
CSDMAエンジン
検出部
監視部
指示部
転送実行部
退避
設定
処理実行状態の定期(コミット確認)
処理の停止・開始指示
処理部
第1機能回路（コンテキスト保持）
ステート・マシン ステート・レジスタ
第2機能回路
ランダムロジック＋初期化不要F/F
レジスタ
データパス/演算器
ワークメモリ（SRAM）
ワークメモリ 内部データ/テスクリプタ/マイクロコード /etc（SRAM）

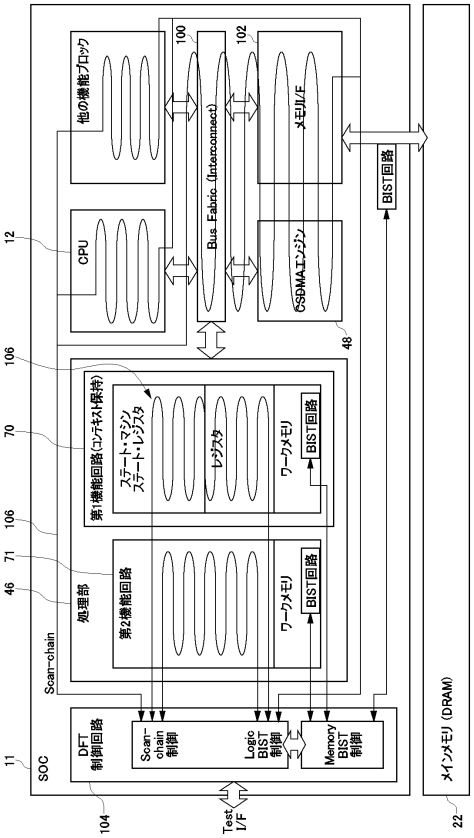
【図３】


Time
Frame 0
Frame 1
Frame 2
1/240 second
App. A
App. B
App. C
App. D

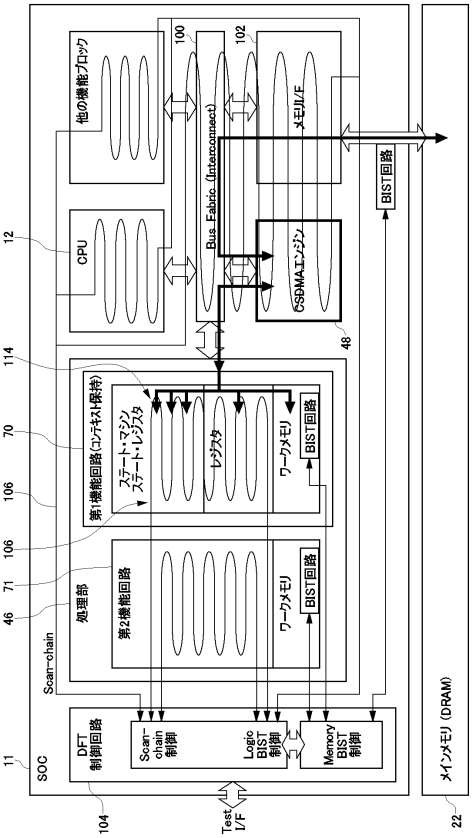
【図４】


Htotal
Hactive
Active Video
Hblank
Hback
Hsync
Hfront
Vactive
Vfront
Vsync
Vback
Vblank
Vtotal

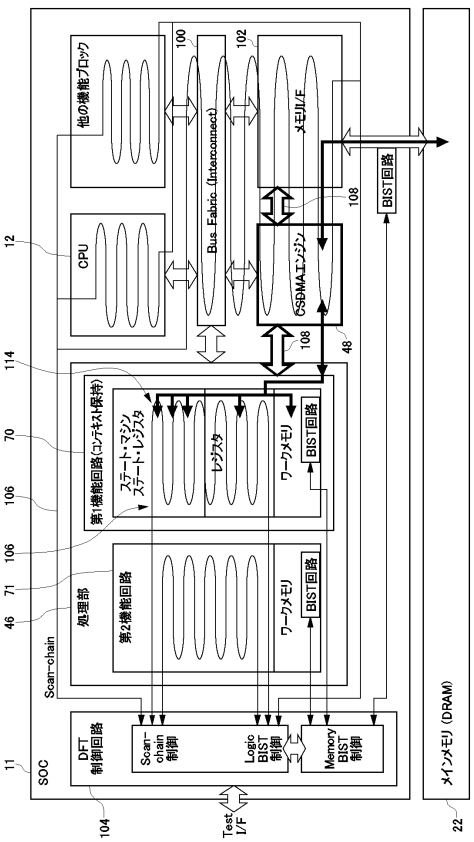
【図５】


他の機能ブロック
CPU
Bus Fabric (Interconnect)
メモリI/F
DSDMAエンジン
BIST回路
第1機能回路(コンテキスト保持)
ステート・マシン
ステート・レジスタ
レジスタ
ワークメモリ
処理部
第2機能回路
Scan-chain
SOC
DFT
制御回路
Scan-
chain
制御
Logic
BIST
制御
Memory
BIST
制御
Test
I/F
メインメモリ (DRAM)
100
102
12
48
106
70
71
46
11
104
22

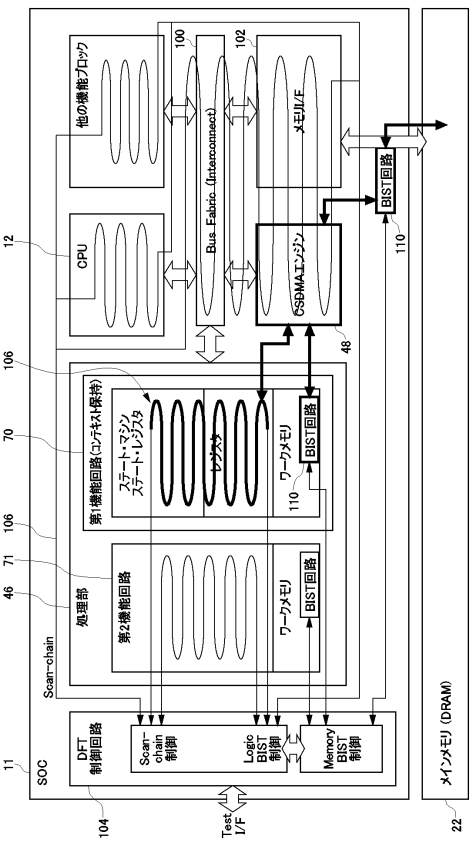
【図６】


他の機能ブロック
CPU
Bus Fabric (Interconnect)
メモリI/F
DSDMAエンジン
BIST回路
第1機能回路(コンテキスト保持)
ステート・マシン
ステート・レジスタ
レジスタ
ワークメモリ
処理部
第2機能回路
Scan-chain
SOC
DFT
制御回路
Scan-
chain
制御
Logic
BIST
制御
Memory
BIST
制御
Test
I/F
メインメモリ (DRAM)
100
102
12
48
114
106
70
71
46
11
104
22

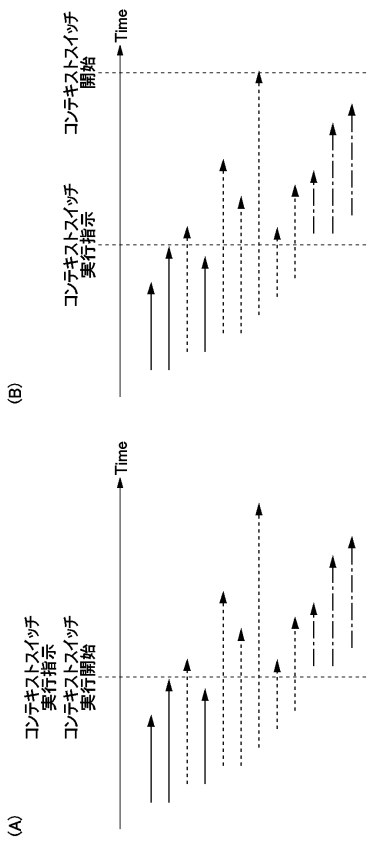
【図７】


他の機能ブロック
CPU
Bus Fabric (Interconnect)
メモリI/F
DSDMAエンジン
BIST回路
108
第1機能回路(コンテキスト保持)
ステート・マシン
ステート・レジスタ
レジスタ
ワークメモリ
処理部
第2機能回路
Scan-chain
SOC
DFT
制御回路
Scan-
chain
制御
Logic
BIST
制御
Memory
BIST
制御
Test
I/F
メインメモリ (DRAM)
100
102
12
48
114
106
70
71
46
11
104
22

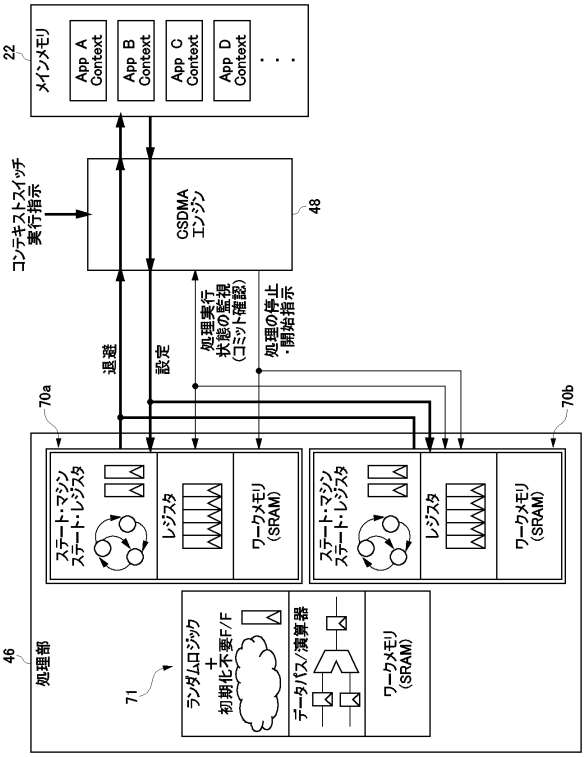
【図８】


他の機能ブロック
CPU
Bus Fabric (Interconnect)
メモリI/F
DSDMAエンジン
BIST回路
110
第1機能回路(コンテキスト保持)
ステート・マシン
ステート・レジスタ
レジスタ
ワークメモリ
処理部
第2機能回路
Scan-chain
SOC
DFT
制御回路
Scan-
chain
制御
Logic
BIST
制御
Memory
BIST
制御
Test
I/F
メインメモリ (DRAM)
100
102
12
48
106
70
71
46
11
104
22

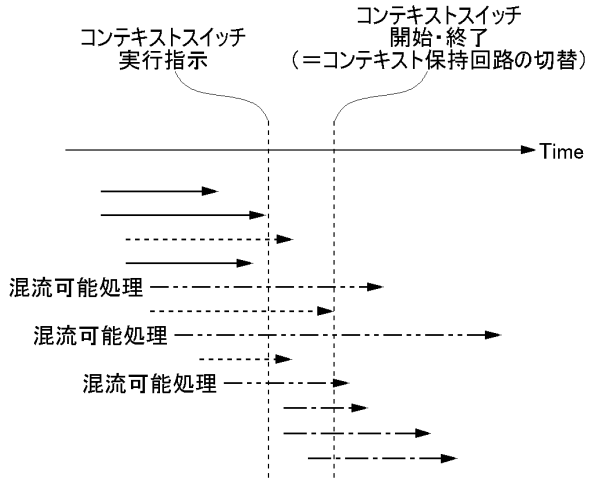
【図９】



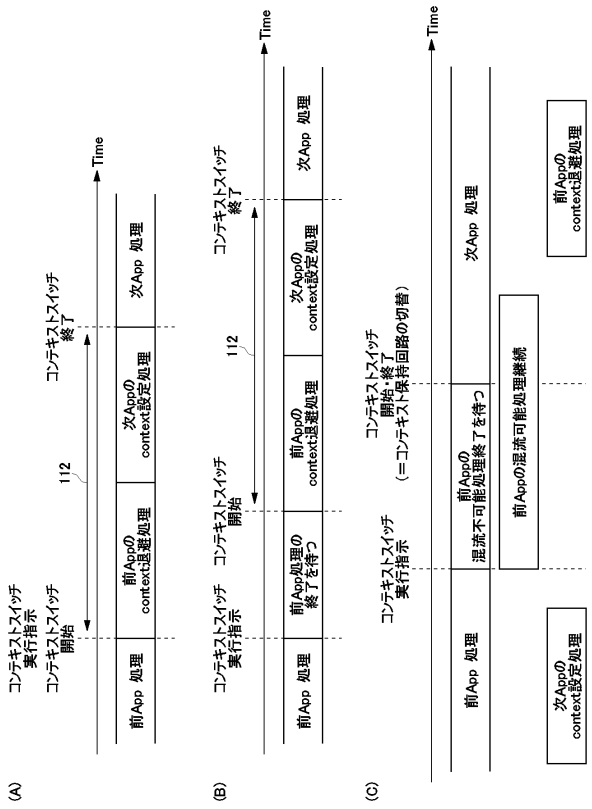
【図１０】



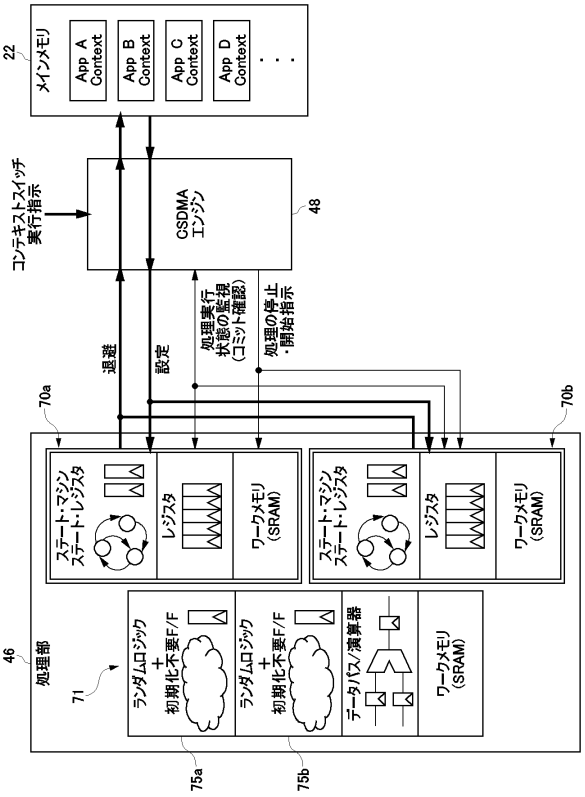
【図１１】



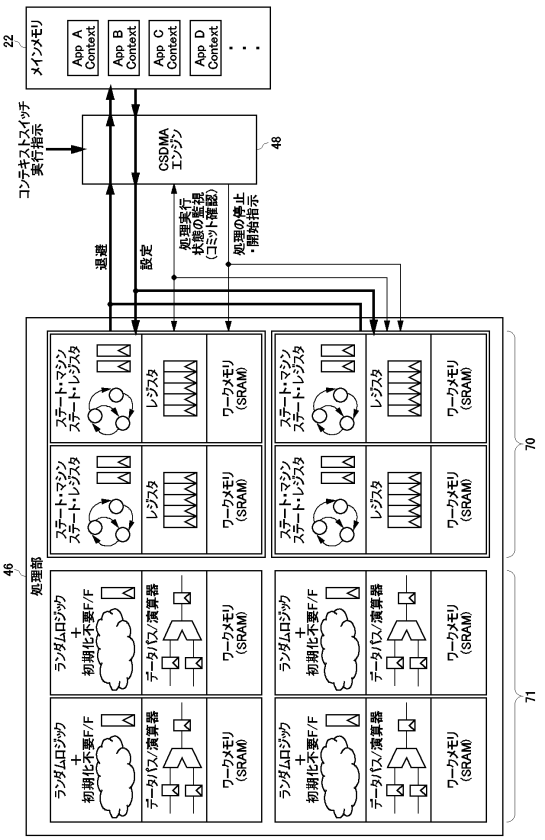
【図１２】



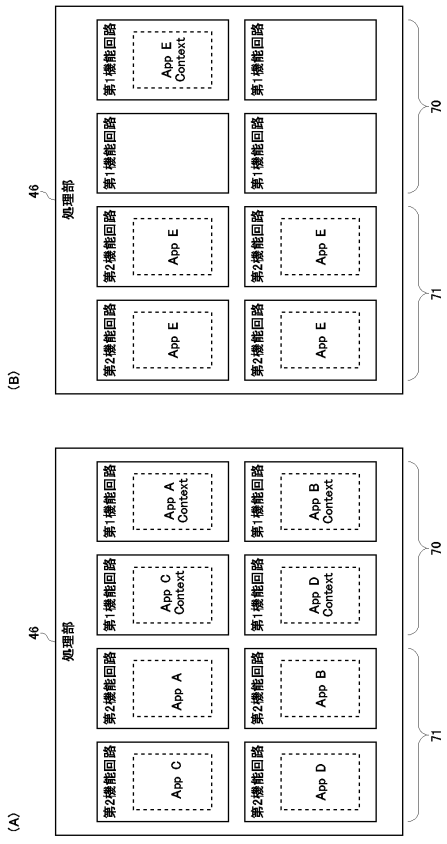
【図１３】


22
メインメモリ
App A Context
App B Context
App C Context
App D Context
コンテキストスイッチ実行指示
CSDMAエンジン
48
退避
設定
処理実行状態の監視（コミット確認）
処理の停止・開始指示
70a
70b
ステート・マシン ステート・レジスタ
レジスタ
ワークメモリ（SRAM）
46
処理部
71
ランダムロジック＋初期化不要F/F
データパス/演算器
75a
75b

【図１４】


22
メインメモリ
App A Context
App B Context
App C Context
App D Context
コンテキストスイッチ実行指示
CSDMAエンジン
48
退避
設定
処理実行状態の監視（コミット確認）
処理の停止・開始指示
ステート・マシン ステート・レジスタ
レジスタ
ワークメモリ（SRAM）
46
処理部
70
71
ランダムロジック＋初期化不要F/F
データパス/演算器

【図１５】


(A)
(B)
46
処理部
70
71
第1機能回路
第2機能回路
App A Context
App B Context
App C Context
App D Context
App E Context
App A
App B
App C
App D
App E

フロントページの続き

ンメント内
審査官　田中 幸雄
(56)参考文献　特開２００３－２７１３９９（ＪＰ，Ａ）
特開平６－８３６１４（ＪＰ，Ａ）
特開平６－１８０６５３（ＪＰ，Ａ）
特開２００４－１８５６０２（ＪＰ，Ａ）
特開２００６－３５１０１３（ＪＰ，Ａ）
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４８
Ｇ０６Ｆ　１５／１７７
Ｇ０６Ｆ　１５／８０