

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2009-76083

(P2009-76083A)

(43) 公開日 平成21年4月9日(2009.4.9)

(51) Int.Cl.	F I	テーマコード (参考)
G06F 9/38 (2006.01)	G06F 9/38 380B	5B013
G06F 9/34 (2006.01)	G06F 9/34 330	5B033

審査請求 有 請求項の数 14 O L (全 113 頁)

(21) 出願番号	特願2008-265785 (P2008-265785)	(71) 出願人	000005223
(22) 出願日	平成20年10月14日 (2008.10.14)		富士通株式会社
(62) 分割の表示	特願2007-180315 (P2007-180315) の分割		神奈川県川崎市中原区上小田中4丁目1番1号
原出願日	平成8年2月13日 (1996.2.13)	(74) 代理人	100099759
(31) 優先権主張番号	08/390,885		弁理士 青木 篤
(32) 優先日	平成7年2月14日 (1995.2.14)	(74) 代理人	100119987
(33) 優先権主張国	米国 (US)		弁理士 伊坪 公一
(31) 優先権主張番号	08/398,299	(74) 代理人	100081330
(32) 優先日	平成7年3月3日 (1995.3.3)		弁理士 樋口 外治
(33) 優先権主張国	米国 (US)	(74) 代理人	100141254
(31) 優先権主張番号	08/472,394		弁理士 榎原 正巳
(32) 優先日	平成7年6月7日 (1995.6.7)	(74) 代理人	100113826
(33) 優先権主張国	米国 (US)		弁理士 倉地 保幸

最終頁に続く

(54) 【発明の名称】 特殊機能を提供する高性能投機的実行プロセッサの構造及び方法

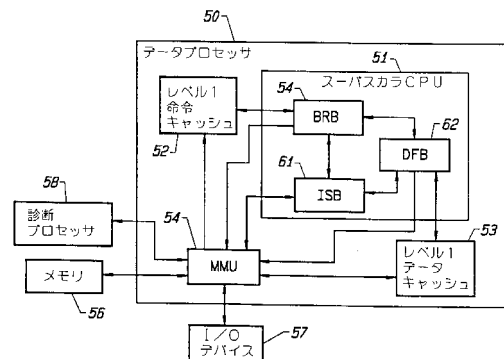
(57) 【要約】

【課題】数多くの特殊なプロセッサ機能及び能力を提供するための構造及び方法を内含する、中央処理ユニット(CPU)といった高性能プロセッサを提供する。

【解決手段】これらの構造及び方法は、(1) 正確な状態を維持しながらロード/ストア命令を含む長待ち時間命令を攻撃的にスケジュールし；(2) 任意の命令境界において正確な状態を維持し回復し；(3) 正確な状態を維持するべく命令状態をトラッキングし；(4) 正確な状態を維持するべく命令をチェックポイントニングし；(5) タイムアウトチェックポイントを新規作成し維持し使用し；(6) 浮動小数点例外をトラッキングし；(7) リネーム可能なトラップスタックを新規作成し維持し使用し；(8) 複数の同時未解決分岐評価のためのウォッチポイントを新規作成し維持し使用し；(9) 正確な状態を維持するために命令状態をトラッキングし；(10) 正確な状態を維持する一方でプロセッサのスループットを増大させる。

【選択図】 図4

図4



**【特許請求の範囲】****【請求項 1】**

命令を格納する命令記憶手段と、

プログラムの順序に従って前記命令記憶手段に格納された命令を発行する命令発行手段と、

前記発行された命令の実行を行う命令実行手段と、

前記プログラム上で定義される論理レジスタと、

前記命令実行手段が前記発行された命令を実行する場合に使用する物理レジスタと、

前記論理レジスタを指定する論理レジスタ番号に対して前記物理レジスタを指定する物理レジスタ番号を対応付けることにより、前記論理レジスタと前記物理レジスタとの対応関係を表す情報を格納するリネームマップ記憶手段と、

チェックポイント命令をデコードする命令デコード手段と、

前記命令発行手段が発行した命令が前記命令デコード手段により、チェックポイント命令であると識別された場合に、前記チェックポイント実行を行うチェックポイント実行手段と、

中央処理ユニットの通常動作を停止させる例外処理を発生させる例外処理発生命令の実行により、前記例外処理が発生したことを検出する例外処理検出手段と、

前記例外処理検出手段により例外処理が検出された場合に、前記例外処理を発生させた命令より後続の前記チェックポイント命令の実行により前記リネームマップ記憶手段に格納された情報を用いて、前記チェックポイント命令を実行した状態に前記論理レジスタの内容を復元するバックアップ手段を有することを特徴とする中央処理ユニット。

**【請求項 2】**

前記論理レジスタは、前記命令の実行対象が格納される論理ソースレジスタと前記命令の実行結果が格納される論理宛先レジスタを含むことを特徴とする請求項 1 記載の中央処理ユニット。

**【請求項 3】**

前記論理レジスタは、前記論理レジスタを識別するための互いに固有の論理レジスタタグを有し、

前記物理レジスタは、前記物理レジスタを識別するための互いに固有の物理レジスタタグを有することを特徴とする請求項 1 又は 2 記載の中央処理ユニット。

**【請求項 4】**

前記中央処理ユニットはさらに、

前記命令デコード手段又は前記例外処理検出手段により例外処理が検出された場合に、前記例外処理を発生させた命令より後続の前記チェックポイント命令の実行により前記リネームマップ記憶手段に格納された情報を用いて、前記チェックポイント命令を実行した状態に前記論理レジスタの内容を復元し、さらに、前記例外処理を発生させた命令まで前記論理レジスタの内容を復元するバックステップ実行手段を有することを特徴とする請求項 1 ~ 3 のいずれか一項に記載の中央処理ユニット。

**【請求項 5】**

前記命令デコード手段又は前記例外処理検出手段により例外処理が検出された場合に、前記例外処理を発生させた命令より後続の前記チェックポイント命令が実行されていない場合には、前記バックアップ手段による前記チェックポイント命令を実行した状態に前記論理レジスタの内容を復元せずに、前記バックステップ実行手段により前記例外処理を発生させた命令まで前記論理レジスタの内容を復元することを特徴とする請求項 4 記載の中央処理ユニット。

**【請求項 6】**

前記チェックポイント命令は、自命令の次に実行する命令のアドレスが自命令のアドレスとは不連続である制御転送命令、条件付命令が実行に際して参照する制御レジスタの値を変更する制御レジスタ値変更命令、及び、トラップを発生させるトラップ命令のいずれかであることを特徴とする請求項 1 ~ 5 のいずれか一項に記載の中央処理ユニット。

10

20

30

40

50

## 【請求項 7】

命令を格納する命令記憶手段と、  
 プログラムの順序に従って前記命令記憶手段に格納された命令を発行する命令発行手段と、  
 前記発行された命令の実行を行う命令実行手段と、  
 前記プログラム上で定義される論理レジスタと、  
 前記命令実行手段が前記発行された命令を実行する場合に使用する物理レジスタと、  
 前記論理レジスタを指定する論理レジスタ番号に対して前記物理レジスタを指定する物理レジスタ番号を対応付けることにより、前記論理レジスタと前記物理レジスタとの対応関係を表す情報を格納するリネームマップ記憶手段と、  
 前記論理レジスタと前記リネームマップ記憶手段の内容を記憶するチェックポイント実行が必要なチェックポイント命令をデコードする命令デコード手段と、  
 前記命令発行手段が発行した命令が前記命令デコード手段により、チェックポイント命令であると識別された場合に、前記チェックポイント実行の代わりに、前記識別されたチェックポイント命令であると識別された命令に先行する全ての命令が完了するまで、前記識別されたチェックポイント命令であると識別された命令の発行及び実行を抑止する命令発行抑止手段と、  
 前記チェックポイント命令の実行後に、中央処理ユニットの通常動作を停止させる例外処理が発生した場合に、前記リネームマップ記憶手段に格納された情報を用いて、前記通常動作の実行に必要な前記論理レジスタの内容を復元するバックアップ手段を有することを特徴とする中央処理ユニット。

10

20

## 【請求項 8】

命令を格納する命令記憶部と前記命令を発行する命令発行部と前記命令を実行する命令実行部とプログラム上で定義される論理レジスタに対応する物理レジスタとを有する中央処理ユニットの命令制御方法において、  
 前記命令記憶部が、命令を格納するステップと、  
 前記命令発行部が、プログラムの順序に従って前記命令記憶部に格納された命令を前記命令実行部に発行するステップと、  
 前記命令実行部が、前記発行された命令を実行するステップと、  
 前記論理レジスタを指定する論理レジスタ番号に対して前記物理レジスタを指定する物理レジスタ番号を対応付けることにより、前記論理レジスタと前記物理レジスタとの対応関係を表す情報をリネームマップ記憶部に格納するステップと、  
 チェックポイント命令をデコードするデコードステップと、  
 中央処理ユニットの通常動作を停止させる例外処理を発生させる例外処理発生命令の実行により、前記例外処理が発生したことを検出する例外処理検出ステップと、  
 前記例外処理が検出された場合に、前記例外処理を発生させた命令より後続の前記チェックポイント命令の実行により前記リネームマップ記憶部に格納された情報を用いて、前記通常動作の実行に必要な前記論理レジスタの内容を復元するバックアップステップを有することを特徴とする命令制御方法。

30

## 【請求項 9】

前記論理レジスタは、前記命令の実行対象が格納される論理ソースレジスタと前記命令の実行結果が格納される論理宛先レジスタを含むことを特徴とする請求項 8 記載の命令制御方法。

40

## 【請求項 10】

前記論理レジスタは、前記論理レジスタを識別するための互いに固有の論理レジスタタグを有し、

前記物理レジスタは、前記物理レジスタを識別するための互いに固有の物理レジスタタグを有することを特徴とする請求項 8 又は 9 記載の命令制御方法。

## 【請求項 11】

前記中央処理ユニットはさらに、

50

例外処理が検出された場合に、前記例外処理を発生させた命令より後続の前記チェックポイント命令の実行により前記リネームマップ記憶手段に格納された情報を用いて、前記チェックポイント命令を実行した状態に前記論理レジスタの内容を復元し、さらに、前記例外処理を発生させた命令まで前記論理レジスタの内容を復元するバックステップ実行ステップを有することを特徴とする請求項 8 ~ 10 のいずれか一項に記載の命令制御方法。

【請求項 12】

前記デコードステップ又は前記例外処理検出ステップにおいて例外処理が検出された場合に、前記例外処理を発生させた命令より後続のチェックポイント命令が実行されていない場合には、前記バックアップステップによる前記チェックポイント命令を実行した状態に前記論理レジスタの内容を復元せずに、前記バックステップ実行ステップにより前記例外処理を発生させた命令まで前記論理レジスタの内容を復元するステップを有することを特徴とする請求項 11 記載の命令制御方法。

10

【請求項 13】

前記チェックポイント命令は、自命令の次に実行する命令のアドレスが自命令のアドレスとは不連続である制御転送命令、条件付命令が実行に際して参照する制御レジスタの値を変更する制御レジスタ値変更命令、及び、トラップを発生させるトラップ命令のいずれかであることを特徴とする請求項 8 ~ 12 のいずれか一項に記載の命令制御方法。

【請求項 14】

命令を格納する命令記憶部と前記命令を発行する命令発行部と前記命令を実行する命令実行部とプログラム上で定義される論理レジスタに対応する物理レジスタとを有する中央処理ユニットの命令制御方法において、

20

前記命令記憶部が、命令を格納するステップと、

前記命令発行部が、プログラムの順序に従って前記命令記憶部に格納された命令を前記命令実行部に発行するステップと、

前記命令実行部が、前記発行された命令を実行するステップと、

前記論理レジスタに対して前記物理レジスタを対応付けることにより、前記論理レジスタと前記物理レジスタとの対応関係を表す情報をリネームマップ記憶部に格納するステップと、

前記論理レジスタと前記リネームマップ記憶部の内容を記憶するチェックポイント実行が必要なチェックポイント命令をデコードするステップと、

30

前記命令発行部が発行した命令が、前記デコードするステップにおいて、チェックポイント命令であると識別された場合に、前記チェックポイント実行の代わりに、前記識別されたチェックポイント命令であると識別された命令に先行する全ての命令が完了するまで、前記識別されたチェックポイント命令であると識別された命令の発行及び実行を抑止するステップと、

前記チェックポイント命令の実行後に、中央処理ユニットの通常動作を停止させる例外処理が発生した場合に、前記リネームマップ記憶部に格納された情報を用いて、前記通常動作の実行に必要な前記論理レジスタの内容を復元するステップを有することを特徴とする命令制御方法。

【発明の詳細な説明】

40

【技術分野】

【0001】

関連出願

この出願は発明者Gene W. Shen et al.によって1995年3月3日に出願された、米国特許出願代理人整理番号第A-60625-1/JAS、米国特許出願第08/398,299号“PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE”の継続であり；該継続出願は同じく発明者Gene W. Shen et al.によって1995年2月14日に出願された、米国特許出願代理人整理番号第A-60625/JAS、米国特許出願第08/390,885号“PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE”の継続である。

50

## 【 0 0 0 2 】

発明者Takeshi Kitaharaによって1995年2月14日に出願された、米国特許出願代理人整理番号第1706号、米国特許出願第08/388,602号“INSTRUCTION FLOW CONTROL CIRCUIT FOR SUPERSCALER MICROPROCESSOR”；発明者Michael Simone及びMichael Shebanowによって1995年2月14日に出願された、米国特許出願代理人整理番号第1693号、米国特許出願第08/388,389号“ADDRESSING METHOD FOR EXECUTING LOAD INSTRUCTIONS OUT OF ORDER WITH RESPECT TO STORE INSTRUCTIONS”；発明者DeForest Tovey,Michael Shebanow,John Gmuenderによって1995年2月14日に出願された、米国特許出願代理人整理番号1707号、米国出願第08/388,606号“METHOD AND APPARATUS FOR EFFICIENTLY WRITING RESULTS TO RENAMED REGISTERS”；及び発明者DeForest Tovey,Michael Shebanow,John Gmuenderによって1995年2月14日に出願された、米国特許出願代理人整理番号第1741号、米国出願第388,364号“METHOD AND APPARATUS FOR COORDINATING THE USE OF PHYSICAL REGISTERS IN A MICROPROCESSOR”はいずれも参考のため本願明細書にその内容を引用する。

10

## 【 0 0 0 3 】

発明者Gene W.Shen,John Szeto,Niteen A.Patkar及びMichael C.Shebanowによって1995年2月14日に出願された、米国特許出願第08/390,885号“PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE”；発明者Gene W.Shen,John Szeto,Niteen A.Patkar及びMichael C.Shebanowによって1995年3月3日に出願された米国特許出願第08/398,299号“PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE”；発明者Chih-Wei David Chang,Kioumars Dawallu,Joel F.Boney,Ming-Ying Li,及びJen-Hong Charles Chenによって1995年3月3日に出願された、米国特許出願第08/397,810号“PARALLEL ACCESS MICRO-TLB TO SPEED UP ADDRESS TRANSLATION”；発明者Leon Kuo-Liang Peng,Yolin Lih,及びChih-Wei David Changによって1995年3月3日に出願された、米国特許出願第08/397,809号“LOOKASIDE BUFFER FOR ADDRESS TRANSLATION IN A COMPUTER SYSTEM”発明者Michael C.Shebanow,Gene W.Shen,Ravi Swami,及びNiteen Patkarによって1995年3月3日に出願された米国特許出願第08/397,893号“RECLAMATION OF PROCESSOR RESOURCES IN A DATA PROCESSOR”；Michael C.Shebanow,John Gmuender,Michael A.Simone,John R.F.S Szeto,Takumi Maruyama,及びDeForest W.Toveyによって1995年3月3日に出願された米国特許出願第08/397,891号“METHOD AND APPARATUS FOR SELECTING INSTRUCTIONS FROM ONES READY TO EXECUTE”；Shalesh Thusoo,Farnad Sajjadian,Jaspal Kohli,及びNiteen Patcarによって1995年3月3日に出願された米国特許出願第08/397,911号“HARDWARE SUPPORT FOR FAST SOFTWARE EMULATION OF UNIMPLEMENTED INSTRUCTIONS” ,Akira Katsuno,Sunil Savkar, 及びMichael C.Shebnowによって1995年3月3日に出願された米国特許出願第08/398,284号“METHOD AND APPARATUS FOR ACCELERATING CONTROL TRANSFER RETURNS” ,Akira Katsuno,Niteen A.Patcar,Sunil Savkar, 及びMichael C.Shebnowによって1995年3月3日に出願された米国特許出願第08/398,066号“METHODS FOR UPDATING FETCH PROGRAM COUNTER”；Sunil Savkarによって1995年3月3日に出願された米国特許出願第08/398,151号“METHOD AND APPARATUS FOR RAPID EXECUTION OF CONTROL TRANSFER INSTRUCTIONS”；Chih-Wei David Chang,Joel Fredrick Boney, 及びJaspal Kohliによって1995年3月3日に出願された米国特許第08/397,910号“METHOD AND APPARATUS FOR PRIORITIZING AND HANDLING ERRORS IN A COMPUTER SYSTEM”；Michael Simoneによって1995年3月3日に出願された米国特許出願第08/397,800号“METHOD AND APPARATUS FOR GENERATING A ZERO BIT STATUS FLAG IN A MICROPROCESSOR”；及びChien Chen及びYizu Luによって1995年3月3日に出願された米国特許出願第08/397,912号“ECC PROTECTED MEMORY ORGANIZATION WITH PIPELINED READ-MODIFY-WRITE ACCESS”もそれぞれ参考のため本願明細書中に引用した。

20

30

40

## 【 0 0 0 4 】

発明者Sunil Savkar,Michael C.Shebanow,Gene W.Shen,及びFarnad Sajjadianによって1995年6月1日に出願された米国特許出願第\_\_\_\_\_号“METHOD AND APPARATUS FOR ROTATING ACTIVE INSTRUCTIONS IN A PARALLEL DATA PROCESSOR”；及び発明者Sunil Sankar,M

50

Michael C. Shebanow, Gene W. Shen, 及び Farnad Sajjadian によって 1995 年 6 月 1 日に出願された米国特許出願第 \_\_\_\_\_ 号 “PROGRAMMABLE INSTRUCTION TRAP SYSTEM AND METHOD” もそれぞれ参考のため本願明細書中に引用した。

【 0 0 0 5 】

発明者 Gene W. Shen, John Szeto, Niteen A. Patcar, 及び Michael C. Shebanow によって 1995 年 6 月 7 日に出願された代理人整理番号第 A -60624/JAS、米国特許出願第 08/487,801 号 “PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE” ; 発明者 Gene W. Shen, John Szeto, Niteen A. Patcar, Michael C. Shebanow, 及び Michael A. Simone によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60622/JAS、米国特許出願第 08/478,025 号 “PROCESSOR STRUCTURE AND METHOD FOR AGGRESSIVELY SCHEDULING LONG LATENCY INSTRUCTIONS INCLUDING LOAD/STORE INSTRUCTIONS WHILE MAINTAINING PRECISE STATE” ; 発明者 Gene W. Shen, John Szeto, Niteen A. Patkar, 及び Michael C. Shebanow によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60623/JAS、米国特許出願第 08/483,958 号 “PROCESSOR AND METHOD FOR MAINTAINING AND RESTORING PRECISE STATE AT ANY INSTRUCTION BOUNDARY” ; 発明者 Gene W. Shen, John Szeto, Niteen A. Patcar, 及び Michael C. Shebanow によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60625-2/JAS、米国特許出願第 08/467,419 号 “PROCESSOR STRUCTURE AND METHOD FOR CHECKPOINTING INSTRUCTIONS TO MAINTAIN PRECISE STATE” ; 発明者 Gene W. Shen, John Szeto, Niteen A. Patcar, 及び Michael C. Shebanow によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60647/JAS、米国特許出願第 08/473,223 号 “PROCESSOR STRUCTURE AND METHOD FOR A TIME-OUT CHECKPOINT” ; 発明者 Gene W. Shen, John Szeto, 及び Michael C. Shebanow によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60648/JAS、米国特許出願第 08/484,795 号 “PROCESSOR STRUCTURE AND METHOD FOR TRACKING FLOATING-POINT EXCEPTIONS” ; 発明者 Hideki Ozone 及び Michael C. Shebanow によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60649/JAS、米国特許出願第 08/472,394 号 “PROCESSOR STRUCTURE AND METHOD FOR RENAMABLE TRAP-STACK” ; 及び発明者 Gene W. Shen, Michael C. Shebanow, Hideki Ozone, 及び Takumi Maruyama によって 1995 年 6 月 7 日に出願された、代理人整理番号第 A -60682/JAS、米国特許出願第 08/482,073 号 “PROCESSOR STRUCTURE AND METHOD FOR WATCHPOINT FOR PLURAL SIMULTANEOUS UNRESOLVED BRANCH EVALUATION” もそれぞれ参考のため本願明細書中に引用した。

【 0 0 0 6 】

技術分野

本発明は投機的追越し (out-of-order) 実行プロセッサにおいて精確な状態 (precise state) を維持しながら、プロセッサの性能を高める装置、システム、及び方法に係わる。

【 背景技術 】

【 0 0 0 7 】

発明の背景

プロセッサにおいて、制御流れ命令 (例えば分岐命令)、メモリのトランザクションに起因する待ち時間 (latency)、及びマルチサイクル演算を必要とする命令は、パイプラインに “バブル” を導入する原因となるため、プロセッサが高い命令実行帯域幅を維持するのを妨げることが多い。投機的追越し命令実行を行うことによって性能を高めることができる。従来、1つの命令からの中間結果を後続の命令に利用できないとき、プロセッサは実行を中断するか、または中間結果が得られるまで “機能停止する”。

【 0 0 0 8 】

2つの技術、即ち、投機的実行と追越し実行によって、この新しいプロセッサは高い実行帯域幅 (execution bandwidth) を維持することができる。投機的実行は、先行の処理ステップから該当分岐を選択するための情報がないままに1つの分岐に遭遇した場合、予測し、この予測に基づいて命令のディスパッチ及び実行を行う公知の技術である。もし予測が誤りであることが後刻判明したら、分岐誤予測回収によって誤予測された命令シーケンスを取消さねばならない。投機的実行によってプロセッサは命令の発行及び実行を継続す

ることができる。公知の予測スキームは性能を高めるため、誤予測実行の頻度を極力少なくする。しかし、投機的マシンにおいて、適正状態の維持は煩雑であり、好ましくないオーバーヘッドを伴なう。追越し実行はメモリ及びマルチサイクル命令待ち時間に留意しない技術である。追越し実行を行うプロセッサにおいては、命令の順序をダイナミックに変更し、順次プログラムとは異なる順序で命令を実行することによって利用可能な命令レベル並列性を見つける。

#### 【 0 0 0 9 】

“ 精確な例外 ” モデルは例外条件のソフトウェア解像度を単純化する重要な要因ではあるが、投機的追越し実行を行うマシンにおいて、精確な例外を維持するのは煩雑な操作である。マシンの状態は一般にプロセッサに特異であり、構造上の状態はすべての制御 / 状態レジスタ、データ・レジスタ、アドレス・レジスタ及びすべての補助メモリの状態を含む。例えばSPARC-V9 Architecture Manualのpp.29 - 30にはSPARC-V9制御 / 状態レジスタが記載されており、ソフトウェア・プログラマーのよく知るところである。マシンの状態はプロセッサに特異な、かつマシンの状態に関する他のすべてを含む構造上の状態のスーパー・セットである。不正命令(faulting instruction)とは例外を発生させる命令である。例外とはプロセッサにその動作を停止させ、処理再開の前にこの例外の原因となった状況を究明するように指示する状況または条件である。例外はエラーとは限らず、例えば、割込みをも含む。実行トラップは、例外から生ずる可能性がある。精確な例外モデルを作成するプロセッサにおいては、不正または例外は構造上の状態を変更しない。不正命令に先立つすべての命令に関しては構造上の状態がすでに変更されているが、不正命令後の命令に関しては構造上の状態は変更されていない。正確な例外モデルがなければ、ソフトウェアが不正命令を識別し、次いで、不正命令を再試行するか、または不正命令をバイパスして次の命令を実行するための再始動点(restart point)を算出しなければならない。

10

20

#### 【 0 0 1 0 】

短パイプライン単一発行マシンにおいて精確な状態を維持する技術は公知である。一般に、短パイプラインマシンは命令の取出し、発行、実行、及び状態を変更されるライトバック段階を含む約4または5段階以下の段階を有する。単一発行方式によれば、パイプライン段階におけるどの命令をフラッシュすべきかを考慮せずにパイプラインをクリアすればよいから、例外または誤予測の場合に回復が簡単になる。これらの公知技術においては、例外が発生した場合、構造上の状態を変更する前に検出される。例外が検出されると、パイプラインから命令がフラッシュされ、構造上の状態が変更されないように、構造上の状態ヘデータ、状態または結果がライトバックされるのを意図的に阻止する。

30

#### 【 0 0 1 1 】

投機的追越しスーパーパスカラ(多重発行)方式では単一発行マシンの場合と比較して、精確な状態を維持することがはるかに困難である。このような投機的追越しマシンの場合には、エラーを発生させる命令が投機的に実行される可能性があり、エラーの場所のあとに現われる構造上の状態の変更を取消す方法及び構造を設ける必要がある。また、例外は多くの場合、プログラムの順序とは異なる順序で検出される。従って、追越しプロセッサは例外を解読し、どの命令を完了(及び構造上の状態を変更)させるべきか、どの命令を取消すべきかを判断できねばならない。

40

#### 【 0 0 1 2 】

図1は、投機的プロセッサにおいて精確な状態を維持するために再順序付けバッファを利用する公知のアプローチを示す。再順序付けバッファは、(結果が得られる)命令実行完了時から実行完了の結果としてマシンの状態が変更されるまでの間に一時遅延を有効に導入する先入れ/先出しスタックによって実現される。メモリへの誤予測分岐命令、実行例外、または同様の状態がライトバックされるのを阻止することによって精確な状態を維持する。誤予測に基づく投機的実行を取消すために状態を復旧するのではなく、命令が投機の域を出ない間、状態の変更を阻止することによって、精確な状態を維持するのである。

#### 【 0 0 1 3 】

50

図2は、命令を実行する前に状態を“チェックポイント”に記憶させ、あとで、記憶されているチェックポイント情報から状態を復旧することにより、投機的プロセッサにおいて精確な状態を維持する公知のアプローチを示す。公知のチェックポイントニングでは、マシンの状態変更を伴う可能性がある、投機的に実行された命令を残らずチェックポイントする。各プロセッサはマシンの状態を決定する一組の状態パラメータ、例えば、すべての制御/状態レジスタ、データレジスタの値などを有する。先のマシン状態を復旧するため、状態を決定するすべてのパラメータを記憶させ、必要に応じて復旧できるようにしなければならない。公知のチェックポイントニング技術では、状態を決定するパラメータのいずれか1つでも変更する可能性がある命令に関してすべての状態決定パラメータを記憶させるのが典型的である。チェックポイントニングの対象となるすべての命令について、公知のチェックポイントニングでは、実行寸前の特定の命令によって変更されそうな状態だけでなく、チェックポイントニングの対象である命令のいずれか1つによって変更される可能性のあるすべての状態情報を記憶させる。例えば、マシンの状態を決定するのに100個の状態パラメータを必要とするマシンの場合、もし命令“X”の実行が1つだけの制御/状態レジスタを変更する可能性があるなら、この命令を実行するには、この命令によって変更されると考えられるパラメータだけでなく、100個の状態パラメータを記憶させる必要がある。

10

20

30

40

50

#### 【0014】

図3は、一連の命令の構造及び内容と公知のチェックポイントとを模式的に示す説明図であり、特定のマシンまたはCPUのための実際のチェックポイント・データを示すものではない。公知のチェックポイントはサイズが一定であるから、各チェックポイントは特定の命令によって実際に変更される状態よりも広くなければならない。公知のチェックポイントニングを利用する回復としては、例えば、不正命令または誤予測に基づいて投機的に実行された命令の直前に記憶されているチェックポイントを利用しての復旧、チェックポイントされている命令の直後にまでプログラム・カウンタをバックアップすること、及びチェックポイントされている命令からの命令再実行などがある。

#### 【0015】

追越し実行を管理し、状況によっては適正状態を維持する手段として、チェックポイントニング、再順序付けバッファ、履歴バッファ、及びフューチャファイルの使用がすでに開示されている。任意の命令境界においてではなく、チェックポイント境界においてマシン状態の復旧を可能にする慣用のチェックポイントニングはHwu及びPattが論述している(第14回コンピュータ・アーキテクチャ年次シンポジウム(1987年6月)における会報pp.18 - 26に発表されたW.Hwu及びY.N.Pattの“Checkpoint Repair for High Performance Out-of-Order Execution Machines”)。パイプラインRISCプロセッサにおける精確な割込みを作成する方法はWang及びEmnettによって報告されている(“Implementing Precise Interruptions in Pipelined RISC Processors”, IEEE Micro, 1993年8月, pp.36 - 43)。同じくパイプライン・プロセッサにおける精確な割込み作成はSmith及びPleszkunによっても報告されている(第12回コンピュータ・アーキテクチャ年次国際シンポジウム会報(1985年6月)、pp. 36 - 44におけるJ.E.Smith及びA.R.Pleszkunの“Implementation of Precise Interrupts in Pipelined Processors”)。高性能スーパースカラ・マイクロプロセッサの設計に関する従来技術の概要はMike Johnsonが記述している(M. Johnson[a.k.a. William Johnson], Superscaler Microprocessor Design, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1991, ISBN 0-13-875634-1)。これらの参考文献の内容は参考のため本願明細書中に引用した。

#### 【発明の開示】

#### 【発明が解決しようとする課題】

#### 【0016】

これらの技術のうち、少なくともいくつかは性能を向上させるが、投機の可能度を制限し、大ざっぱなマシン状態は回復できても命令レベルのマシン状態回復は不可能であり、完全に満足できるものではない。公知の再順序付けバッファ技術は、その良さが投機の可



能度と正比例するから、投機の可能度に限界がある。即ち、取消しの対象となりそうなすべての命令に関して、命令実行の結果を再順序付けバッファに記憶させねばならない。例えば、もしマシンが64個の未決定かつ投機的命令を可能にするなら、再順序付けバッファは結果を記憶するための少なくとも64個の記憶場所を含まねばならない。公知のチェックポイントにおいては、チェックポイントされるチェックポイント数はマシン中の未定命令数よりも少ないのが普通であるが、各チェックポイントに記憶されるデータ量は極めて大きい（各時点におけるマシンの全体的な状態）。チェックポイント記憶条件はプロセッサのチップ基板面積に負担を課する。理想的には、精確な状態を維持するためのスキームは比較的多い同時的に存在する投機的命令に対して線形または低位（例えば対数）関係にスケールできることが望ましい。再順序付けバッファの入口(entry)記憶域はデータを記憶するのに十分な幅を持たねばならず、これらの技術はまたすべての入口までの連想検索を必要とする。これは、非常に大きな再順序付けバッファにとって困難である。

10

**【0017】**

さらに、精確な状態を回復するための従来の方法は一部の環境で不完全である。たとえば、投機的分岐の結果として実行される命令が外部「ダム」デバイスの状態を修正する場合、状態の回復が通常、外部デバイス修正ポイント前のチェックポイント境界での状態回復を伴うため、外部デバイスを修正する命令を含む非不正命令のフォワード再実行を伴う場合である。このような場合、不正命令の再実行だけでは、外部デバイスの状態変化の効果を元通りにせず、状態回復ポイントと不正命令との間で非不正命令を再実行することは、さらに問題を生じる。

20

**【0018】**

従来のチェックポイントニングでは、命令ストリームの限られた数のポイントでマシン状態をセーブし、チェックポイントされたアーキテクチャル状態を復旧して、分岐誤予測または実行エラーから回復する。従来のチェックポイントニングは命令を1つ1つチェックポイントしない。従来のチェックポイントニングは、チェックポイント境界、すなわち、チェックポイントを確立した命令でのみマシン状態を回復できる。フォールトまたは実行エラーが発生すると、チェックポイントされた状態は、周知の技術によって回復され、これによって、チェックポイント後の不正命令を含むすべての命令を事実上「元通り」にする。そして命令は、たとえば単一発行モードで、チェックポイントされた命令から順次前向きに再実行され、不正命令に到達する。

30

**【0019】**

この従来のチェックポイントニング技術では投機的実行が可能だが、多くの点で不利益である。たとえば、これは堅牢な例外回復とならない。間欠的エラーでは、例外作成命令前のチェックポイントへの従来のマシンバックアップと、チェックポイント後の命令の再実行は決定的な行動とならない場合があり、再実行された命令の誤って変更された状態を伝播することによって、マシン状態の混乱を増すことがある。従来のチェックポイントニングとマシンバックアップ手順は、命令の再実行を最小限にしようとするものではない。さらに、ハードウェアの故障やマシンのタイムアウトなど災害的なマシンエラーは、回復したチェックポイント済み命令と例外を生じる命令との間のすべての命令が再実行されると、プロセッサをデッドロックさせることがある。

40

**【0020】**

従来の「再順序づけバッファ」とは、予め定義された大きさの先入れ先出し(FIFO)メモリ構造の再順序づけバッファで投機的命令を管理する。命令が実行を完了したら、実行によるデータ値の結果が再順序付けバッファに書き込まれ、バッファを移動して上位に現れる時、データ値が再順序づけバッファからレジスタファイルに書き込まれる。再順序づけバッファの大きさは、命令実行の完了とアーキテクチャル状態の恒久的修正との間の遅延を効果的に定義する。一旦データ値がレジスタに書き込まれると、これらを元通りにすることはできない。再順序付けバッファの記述項(エントリ)をレジスタファイルの記述項と関連づける手順である「連想探索」について、M. Johnson がSuperscalar Microprocessor Design の40頁以降で説明している。

50

## 【 0 0 2 1 】

再順序付けバッファスキームには少なくとも3つの限界がある。1つ目は、従来の再順序付けバッファスキームでは、命令実行の結果のみが再順序付けバッファにセーブされ、プログラムカウンタ(PC)値はセーブされない。そのため、再順序付けバッファを使った分岐誤予測回復には、PC再構築、命令フェッチ、命令発行の追加ステップが必要となる。その結果、再順序付けバッファを使った分岐誤予測からの従来の回復は遅れる。

## 【 0 0 2 2 】

第2に、再順序付けバッファは一般に限られた命令ミックスの投機的実行しかできない。たとえば、命令発行段階中に検出されたトラップは(発行トラップ)、一般に制御レジスタアップデートを伴うため、再順序付けバッファを使って投機的に実行されないことがある。再順序付けバッファ技術は、制御レジスタアップデートを伴う命令の投機的実行をサポートしない。一定の命令セットアーキテクチャで発行トラップを投機的に入力できないこと(例: Sun Microsystems SPARCアーキテクチャの「スピル/フィル」トラップ)は、大きな性能上の限界となる。

10

## 【 0 0 2 3 】

第3に、再順序付けバッファの大きさは一般に、プロセッサに許された未定命令数の直接線形関数である。たとえば、64個の未定命令の可能なプロセッサでは、64個の記述項を持つ再順序付けバッファが必要となる。プロセッサ内に多数のバッファレジスタを割り当てることは禁止されることがあり、特に大型のアクティブ命令ウィンドウ、すなわち比較的多数の同時未定命令によって、命令レベルの並列性の抽出が改善されるデータフロープロセッサに当てはまる。データフロープロセッサとは、命令実行の順序が、従来の非データフロープロセッサのようにプログラムカウンタのインクリメントによるのではなく、オペランドやデータ利用可能性によって決定するプロセッサである。

20

## 【 0 0 2 4 】

フューチャファイルは、再順序付けバッファの連想探索問題を避けるための再順序付けバッファ技術の修正である。フューチャファイルについては、M. JohnsonのSuperscalar Microprocessor Designの94頁から95頁に説明されている。履歴バッファは、順序無視(out-of-order)完了のパイプライン・スカラー・プロセッサで正確な割り込みを実行するため提案された方法と構造で、M. JohnsonのSuperscalar Microprocessor Designの91頁から92頁に説明されている。

30

## 【 0 0 2 5 】

本書で全体を参照することにより明示的に組み入れるD. WeaverとT. GermondによるThe SPARC Architecture Manual、Version 9、Englewood Cliffs(1994年)では、順序無視投機的実行プロセッサの特定のタイプについて説明している。SPARC V9アーキテクチャには、浮動小数点状態レジスタ(FSR)が必要である。FSRには、FSR\_\_accrued\_\_exception(FSR.aexc)フィールド、FSR\_\_current\_\_exception(FSR.cexc)フィールド、FSR\_\_floating\_\_point\_\_trap\_\_type(FSR.ftt)フィールドの3つのフィールドがあり、浮動小数点例外が発生すると更新されて、トラップハンドリングルーチンが、浮動小数点例外によるトラップに対処するため利用する。これらフィールドのアップデートは、命令がプログラム順序とは異なる順序で実行、完了するため、順序無視実行プロセッサでは難しい。これらフィールドは、命令がプログラム順序で発行、実行されたかのように更新されるか、更新されたように見えなければならないため、これらの例外を追跡してFSRレジスタを正しく更新するためには、装置および対応する方法が必要である。

40

## 【 0 0 2 6 】

命令を投機的に実行できるデータプロセッサでは、分岐方向(取るか取らないか)、あるいは分岐アドレス(目標アドレスまたは分岐命令の次のアドレス)は、解決する前に予測可能である。後に、これら予測が誤っていることがわかれば、プロセッサは以前の状態にバックアップして、正しい分岐ストリームで命令実行を再スタートする。但し、市販のスーパースカラプロセッサの大半は、分岐予測が正しいかどうかをチェックするには、1サ

50

イクルあたり1つの分岐のみ評価可能である。しかし、多重予測分岐は1つのサイクルで評価できることが多い。そのため、他の場合では実行可能な分岐評価が遅延することになる。分岐評価の遅延は、プロセッサの性能に大きな影響を与える。

#### 【0027】

さらに、従来の投機的実行プロセッサでは、トラップが生じると、プロセッサは予測されたすべての分岐が解決してトラップが真実であり投機的ではないことを確かめるまで待たなければならない。トラップが真実であるかどうかをプロセッサが確かめる最も簡単な方法は、トラップを取る前にプロセッサを同期させることである（すなわち、トラップ条件発生前に発行されたすべての命令を実行、完了する）。但し、頻繁に発生するトラップにこれを行うと、プロセッサの性能を落とす。スピル/フィル発行トラップとソフトウェアアトラップ（Tcc命令）がしばしば発生するSPARC-V9アーキテクチャでは特にそれが当てはまる。プロセッサの性能を上げるためにはこの問題を解決しなければならない。

10

#### 【課題を解決するための手段】

#### 【0028】

##### 発明の概要

先行技術の前記問題は、多数の特殊プロセッサ機能および能力を提供する構造および方法を含む高性能プロセッサに関する発明で対処される。これら構造および方法には、（1）精確状態を維持しながらロード/ストア命令を含む長待ち時間(latency)命令の積極的なスケジューリング；あらゆる命令境界での精確状態の維持と回復、（3）精確状態を維持するための命令状態の追跡、（4）精確状態を維持するための命令のチェックポイントニング、（5）タイムアウトチェックポイントの作成、維持および利用、（6）浮動小数点例外の追跡、（7）リネーム可能なトラップスタックの作成、維持および利用、（8）複数の同時未解決分岐評価のためのウォッチポイントの作成、維持および利用、（9）精確状態維持のための命令状態追跡、（10）精確状態を維持しながらのプロセッサループアウト改善のための構造および方法、を限定せずを含む。他の構造機能および能力は下記の開示および添付の図面と請求の範囲で説明される。

20

#### 【0029】

前記問題は、本発明の方法および構造の一側面によって解決されるが、これは、発行時に各命令に独自の識別タグを割り当て、そのタグを最初のアクティブな命令データ構造で記憶場所に関連させ、各命令の命令アクティビティ状態の変化に対応して記憶場所に記憶されたデータを更新し、命令アクティビティ状態に対応して移動する記憶場所に複数のポイントを維持することによって、精確状態の追跡と維持を行うものである。状態情報には、命令発行時に設定され、エラーなしに実行が完了するとクリアされる1つのアクティブビットが含まれる。最後に発行された命令を指すポイント（発行した命令のポイント）、エラーなしに完了し、順序的に早い命令すべてがエラーなしに完了した命令を指すポイント（最後にコミットした命令のポイント）、割り当てられたプロセッサリソースを取り戻した最後の命令（取り戻した命令のポイント）が設定される。これら3つのポイントは、データ構造の1つの命令に関連する各場所のアクティブビットの比較および所定の規則に基づいて、データ構造に沿って後に発行された命令に向かって前進する。命令の例外またはエラー条件は、アクティブビットの変更を妨げるので、ポイントの移動が制御され、これら条件下で妨げられる。

30

40

#### 【0030】

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、ロード/ストア命令を含む、外部メモリ参照命令などの長待ち時間命令を、投機的順序無視実行プロセッサの短待ち時間命令の前に追跡および積極的にスケジューリングしながら、メモリ参照命令を他の命令と区別し、どの命令の予測が終わって投機的結果を有するかを識別し、投機的分岐誤予測や実行例外の懸念なしに実行可能な実行命令を参照するメモリのみスケジューリングすることによって、精確状態を維持する。このようにスケジューリングできる命令は、発行時に各命令に独自の識別タグを割り当て、そのタグをデータ構造の記憶場所に関連させ、各命令の命令アクティビティ状態の変化に対応してデータ構造の記

50

憶場所に記憶されたデータを更新し、命令アクティビティ状態に対応して移動する記憶場所に複数のポインタを維持することによって、追跡する。状態情報には、命令発行時に設定され、エラーなしに実行が完了するとクリアされる1つのアクティブ命令インジケータが含まれる。最後に発行された命令、エラーなしに完了し、順序的に早い命令すべてがエラーなしに完了した命令（最後にコミットした命令）、割り当てられたプロセッサリソースを取り戻した最後の命令（取り戻した命令）を指すポインタが設定される。さらに、データ構造の第2の記憶場所にタグを関連させ、メモリ参照命令のような長待ち時間命令について命令アクティビティ状態変化に応じて第2記憶場所に記憶されたデータを更新し、長待ち時間（メモリ参照等）命令アクティビティ状態に対応して移動する記憶場所に複数のポインタを維持する。状態情報には、長待ち時間命令が発行された時点でクリアされるが、他の命令タイプすべてについて設定され、命令実行がエラーなしに完了した時にクリアされる長待ち時間命令タイプが含まれる。最も早く予測された分岐命令を指す第1ポインタ（予測分岐ポインタ）がデータ構造に設定される。エラーなしに完了し、順序的に前のすべての命令がエラーなしに完了し、アクティブな長待ち時間命令を越えて前進する最後の命令を指す第2ポインタが設定される。発行、コミット、リタイア、予測された分岐と、長待ち時間ポインタは、各場所のアクティブビットとメモリビットの比較および所定の規則に基づき、第1および第2環状データ構造に沿って後に発行された命令に向かって前進する。例外または命令のエラー条件は、アクティブビットとメモリビットの変更を妨げるため、これら条件下ではポインタの移動は妨げられる。1以上のデータ構造を設けてアクティビティ状態と命令タイプデータを記憶することができ、インジケータはデータ構造に記憶されるセットまたはクリアされたビットでもよい。

10

20

#### 【0031】

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、分岐命令や、プログラムカウンタ不連続性を作ったり、制御レジスタ値を修正する副作用のある命令を含む命令タイプの所定セットにのみにチェックポイントを作成し、プログラムカウンタの不連続性を作る各命令にチェックポイントを作成するか、プロセッサを同期させて状態を修正することのある命令を順序通りに実行して、プロセッサを例外から回復するための手段および方法を提供することで、投機的順序無視実行プロセッサの命令境界で正確なマシン状態を回復する方法を提供する。例外条件からの回復には、実行例外が発生した時の命令一連番号、またはサイクル中に1つ以上の実行例外が発生した時の最も早い命令一連番号の決定および記憶と、実行例外後に最も近い前回チェックポイントした命令にプロセッサをバックアップし、チェックポイントが現在発行されている命令と不正命令との間にある場合、チェックポイントした情報からプロセッサ状態を回復し、レジスタリソースを更新し、チェックポイントした命令があれば、ここから最も早い不正命令の直前のポイントまでのバックステップ量でプロセッサのプログラムカウンタをデクリメントすることによるプロセッサのバックステップが含まれる。今後の改良では、従来のチェックポイントに必要な多量の記憶装置は、レジスタデータそのものではなく、ロジカルおよびフィジカルレジスタリネームマップをチェックポイントングすることによって減少する。

30

#### 【0032】

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、プロセッサにチェックポイントング命令を設けてチェックポイントした状態を減らす一方、命令の発行および実行前にCPUで実行された時アーキテクチャル状態を修正するような命令を予め識別し、特定の命令のチェックポイントに必要な状態記憶量と典型的命令ストリームで命令が発生する頻度を含む所定の選択基準に基づき、実行前に特定の命令のアーキテクチャル状態をチェックポイントングすることなしに特別実行モードで実行するため識別された命令の内の1つを予め選ぶことによって、プロセッサの正確状態を維持する。そして、チェックポイントは識別した命令についてのみ形成し、状態を修正するような他の命令はチェックポイントせず、特別な同期モードで実行し、例外が発生した時に、実行結果をプロセッサ状態に書き戻す前にその例外に対処できるようにする。本発明の

40

50

方法および構造の1実施例では、この基準は、修正可能な状態の種類と、実行中に命令によって修正されることのある修正可能状態の量からなり、おそらくは所定の名目命令ストリームで命令の発生する頻度が含まれる。低推定統計頻度を持って現れ、比較的大きなチェックポイント記憶量の必要な命令を同期モードの実行に選択し、チェックポイントしないことが有利である。本発明の構造および方法の1実施例では、プロセッサの同期においては、実行前に命令ストリームでのマシン同期を必要とする同期命令として命令を識別し、未解決の発行済み命令をすべてコミットし、リタイアして、エラーなしに実行を完了して実行結果を状態に書き戻すまで同期命令実行を遅延させ、マシン同期の必要とする各命令を逐次順序通り実行し、状態へのライトバック前に各同期命令実行から生じる例外条件を識別し、同期命令実行中に生じることのある例外条件に対処し、その後のみ実行結果をマシン状態に書き戻すようにして、精確状態を同期命令のチェックポイントニングなしに維持できるようにする。

10

**【0033】**

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、デコードした命令属性にのみ基づいて現在のプロセッサ状態を記憶するためのチェックポイントを形成するのではなく、発行した命令数、経過したクロックサイクル数、および最後にチェックポイントを形成してからの間隔に基づいてチェックポイントを形成する。本発明のチェックポイントニングに基づくタイムアウト条件ベースのチェックポイントニングは、命令の最大数をチェックポイント境界以内に限定するため、例外条件からの回復期間を抑制する。命令ウィンドウのサイズがチェックポイント境界内の命令最大数より大きい限り、例外発生時、プロセッサは命令デコードベースのチェックポイント技術より早くチェックポイントした状態を復旧することができ、プロセッサの命令ウィンドウサイズへの状態回復依存をなくす。本発明のタイムアウトチェックポイント形成は、従来のチェックポイントを実行する構造および方法、または本発明のロジカルおよびフィジカルなレジスタリネームマップチェックポイントニング技術に用いることができる。本発明のタイムアウトチェックポイント形成構造および方法は、従来のプロセッサバックアップ技術、およびプロセッサバックアップとバックステッピングを含む本発明のバックトラッキング技術と共に用いることができる。

20

**【0034】**

前記問題は、本発明の構造および方法の別の側面によって解決されるが、これは、発行ユニット、実行手段、浮動小数点例外ユニット、精確状態ユニット、浮動小数点状態レジスタおよび書き込み手段からなるプログラム制御順序無視実行データプロセッサを提供する。発行ユニットは実行のプログラム制御順序で命令を発行する。発行された命令には浮動小数点命令と非浮動小数点命令が含まれる。実行手段は、少なくとも浮動小数点命令が実行手段によってプログラム制御順序を無視して実行されるよう、発行された命令を実行する。浮動小数点実行ユニットには、記憶要素をはじめとするデータ記憶構造が含まれる。発行された各命令は記憶要素の1つに対応する。各記憶要素は、浮動小数点命令識別フィールドと、浮動小数点トラップタイプフィールドを有する。浮動小数点例外ユニットにも、対応する記憶要素の浮動小数点命令識別フィールドに、発行された各命令に関してデータを書き込み、対応する発行済み命令が浮動小数点命令か否かを示すための第1ロジックが含まれる。これはさらに、実行中に1つ以上の浮動小数点実行例外を生じて、浮動小数点実行トラップの予め定義された複数の種類の対応する1つとなる発行済み浮動小数点命令について、対応する記憶要素の浮動小数点トラップタイプフィールドにデータを書き込み、結果として生じる浮動小数点実行トラップの予め定義された種類の1つを識別する第2のロジックが含まれる。精確状態手段は、実行中に実行例外を生じず、プログラム制御順序でこれに先行するすべての発行済み命令がリタイアした発行済み命令をそれぞれリタイアする。予め定義した実行例外の最初の1つが発行済み命令によって生じた時、実行手段は発行済み命令の実行を継続し、精確状態手段は、リタイアできない発行済み命令に遭遇するまで発行済み命令のリタイアを継続することによって実行トラップの順序付けを行う。リタイアできない発行済み命令は、(a)第1の実行例外を生じさせた発行済み命

30

40

50

令、および(b)第1の実行例外を生じさせた発行済み命令より早く発行されたが、第2の実行例外を第1の実行例外より遅く発生させた発行済み命令の1つである。浮動小数点状態レジスタは浮動小数点トラップタイプフィールドを有する。書き込み手段は、浮動小数点状態レジスタの浮動小数点トラップタイプフィールドにデータを書き込み、リタイアできない命令に対応する記憶要素の浮動小数点識別フィールドのデータが、リタイアできない命令が浮動小数点命令であることを示す時、リタイアできない命令に対応する記憶要素の浮動小数点トラップタイプフィールドのデータによって、浮動小数点実行トラップの種類を識別する。

#### 【0035】

前記問題は、本発明の構造および方法の別の側面によって解決されるが、これは、投機的にトラップを取り、トラップから戻るためのデータプロセッサと関連する方法を提供する。データプロセッサは、それぞれ対応するトラップレベルを有する入れ子トラップを取るため所定の数のトラップレベルをサポートする。データプロセッサは、チェックポイント形成手段、チェックポイントへのバックアップ手段、トラップを取る手段、トラップから戻る手段、レジスタおよびトラップスタックユニットからなる。レジスタは、トラップを取る度にデータプロセッサの状態を定義する内容を持つ。トラップスタックユニットは、トラップレベルより数の多いトラップスタック記憶記述項を有するトラップスタックデータ記憶構造を含む。また、トラップレベルの1つに現在マッピングできるトラップスタック記憶記述項の現在利用可能リストを保持するフリーリストユニットを含む。フリーリストユニットは、トラップを取る度に、対応するトラップレベルの1つに現在マッピング可能な次のトラップスタック記憶記述項を識別する。トラップスタックユニットはさらに、トラップを取る度に、現在利用可能なトラップスタック記憶記述項の次の1つにレジスタの内容を書き込む読み取り/書き込みロジックを含む。さらにまた、トラップスタック記憶記述項の1つに各トラップレベルの現在のマッピングを保持するリネームマッピングロジックを含む。リネームマッピングロジックは、トラップを取る度に、トラップスタック記憶記述項の1つへの対応するトラップレベルの古いマッピングを、現在利用可能なトラップスタック記憶記述項の次のものへの対応するトラップレベルの現在のマッピングに置き換える。トラップスタックユニットはまた、現在のマッピングによってトラップレベルの1つに現在マッピングされていないが、トラップレベルの1つにマッピングできないトラップスタック記憶記述項の利用不可リストを保持するリソースリクレームユニットも含む。リソースリクレームユニットは、トラップを取る度に、古いマッピングによって対応するトラップレベルにマッピングされたトラップスタック記憶記述項を利用不可リストに加え、取ったトラップを元通りにできなくなる度に、古いマッピングによって対応するトラップレベルにマッピングされたトラップスタック記憶記述項を利用不可リストから取り除く。フリーリストユニットは、利用不可リストから取り除かれたトラップスタック記憶記述項を、現在利用可能リストに加える。最後に、トラップスタックユニットは、チェックポイント記憶記述項を含むチェックポイント記憶ユニットを含む。形成された各チェックポイントは、対応するチェックポイント記憶記述項を有し、形成された各チェックポイントについて、チェックポイント記憶ユニットがリネームマッピングロジックの現在のマッピングとフリーリストユニットの現在利用可能リストを、対応するチェックポイント記憶記述項に記憶できるようにする。チェックポイントへのバックアップの度に、リネームマッピングロジックは、現在同ロジックが保持するマッピングを、対応するチェックポイント記憶記述項に記憶されたマッピングに置き換え、フリーリストユニットは、現在同ユニットが保持する利用可能リストを、対応するチェックポイント記憶記述項に記憶された利用可能リストに置き換える。

#### 【0036】

前記問題は、本発明の別の側面によって解決されるが、これは、複数の投機的発行・予測された命令の実行結果を同時にモニターするためのデータプロセッサおよび関連する方法を提供する。本発明の構造および方法は、プロセッサ内の1つ以上の実行ユニットからの実行結果信号を受信するため連結されたウォッチポイントデータを記憶するための複数

10

20

30

40

50

のウォッチポイントレジスタを有するウォッチポイントユニットを提供する。投機的に発行・予測された命令の予測された条件データ結果を含むウォッチポイントデータを記憶するウォッチポイントレジスタは、命令が発行される時に割り当てられる。予測された条件データ結果は、投機的に発行された命令の制御フロー転送方向が決まる条件の予測値と、投機的に発行・予測された命令が投機的に発行されたベースを識別する。ウォッチポイントユニットは、実行ユニットからデータフォワードバスで転送される投機的に発行・予測された命令の実行結果信号をモニターし、記憶されたウォッチポイントデータと、実際の既知条件データ結果信号を初めとする実行結果信号および、実際の条件コードが予測された条件コードと一致するか否かを決定する所定の規則に基づき、所定の事象の発生を検出する。投機的に発行・予測された命令の1つについてウォッチポイントレジスタに記憶されたウォッチポイントデータを、投機的に発行された命令に関連するデータフォワードバスで到着した結果信号と比較し、信号が一致するか一致しないかを決定する。一致は、投機的に発行・予測された命令が正しく予測されたことを示し、不一致は、投機的に発行・予測された命令が誤予測だったことを示す。誤予測が識別されると、プロセッサは初期状態に戻り、誤予測に基づいて実行された命令を元通りにする。本発明の側面は、複数の実行ユニットと、複数の予測された分岐またはジャンプ・アンド・リンク命令を初めとする複数の投機的に発行された命令を同時にモニターする能力を含む、実行ユニットからの条件コードデータを同時につかむ構造および方法；実行ユニットからのデータフォワードバスを同時にウォッチすることで、予測された分岐またはジャンプ・アンド・リンクが待っている複数の条件コードデータまたは計算されたジャンプアンド・リンクアドレスをグラブする構造および方法；分岐またはジャンプ・アンド・リンク命令の複数の誤予測信号を同時に生成する構造および方法；1つの共有記憶領域に別の分岐アドレスまたは予測されたジャンプ・アンド・リンクアドレスを記憶する構造および方法；および条件コードタグ比較とグラビングのタスクを分けることで、クリティカルバスをスピードアップするための構造および方法を提供する。本発明の構造および方法の1実施例では、これら複数の側面を有利に組み合わせて全体的なプロセッサ性能を向上させる。

10

20

30

40

50

【発明を実施するための最良の形態】

【0037】

実施例の説明

図4は、データプロセッサ50のトップレベル機能ブロック図である。データプロセッサは、新規の順序無視投機的実行（スーパスカラ）中央処理装置（CPU）51を含む。例示CPUは、SPARC-V9 Architecture Manualに記載されたSPARC-V9命令セットを実行可能である。D.WeaverおよびT.Germondによる「The SPARC Architecture Manual」、Version 9、Englewood Cliffs（1994年）は、参照により本書に明示的に組み込む。但し、当業者は、本書に記載する新規設計および方法がSPARC-V9アーキテクチャに限定されないことを認識するであろう。

【0038】

実際、CPU51は、すべてのデータプロセッサに該当する精確マシン状態を維持しながら、スループットを上げる新規設計および方法を採用している。特に、これは（1）プロセッサリソースおよび状態をモニターし、回復する命令をトラッキングするための新規設計および方法；（2）ロード/ストア命令などの長待ち時間命令を積極的にスケジューリングするための新規設計および方法；（3）マシン状態をチェックポイントニングするための新規設計および方法；（4）例外または誤予測検出後にマシン状態を回復するための新規設計および方法を採用している。

【0039】

本明細書の実施例の説明セクションの概略を便宜上ここに示す。本明細書の見出しは便宜的参照のためにのみ示すもので、本セクションの開示の適用可能性を本発明の特定の側面または実施例に限定すると解釈してはならない。

【0040】

I. CPUオペレーション概要

- II. 命令トラッキング
  - A. 命令フェッチ
  - B. 命令発行
    - 1. 命令発行概要
    - 2. 命令シリアル番号割り当てと命令状態記憶
      - a. 命令発行 - コミット - リクレームユニット (ICRU)
      - b. 命令発行関連ポインタおよび発行段階中のポインタ維持
    - 3. レジスタリネーミング
    - 4. リザベーションステーションへのディスパッチ
  - C. 命令実行および完了 10
  - D. 命令実行段階への PSU 参加
  - E. 命令コミットメント
  - F. 命令リタイアおよびリソース回復
    - 1. 命令コミットおよびリタイア時の RRF への更新とマップリネーム
    - 2. 命令リタイア時の RRF 読み取り
    - 3. ポインタを使った精細状態維持
- III. 積極的長待ち時間 (ロード/ストア) 命令スケジューリング
  - A. メモリ命令状態情報記憶
  - B. 長待ち時間情報のトラッキングおよびスケジューリングのための NMCSN および PBSN ポインタ 20
  - C. NMCSN 前進
  - D. データフローブロック内のロード - ストアユニット (LSU)
    - 1. インレンジメモリ参照 (長待ち時間) 命令の識別
    - 2. 積極的長待ち時間 (ロード/ストア) 命令スケジューリングのための基本的構造
- および方法強化
- IV. チェックポインティング
  - A. チェックポイント割り当てレジスタの構造
  - B. チェックポイント割り当て
  - C. チェックポイントリタイア
  - D. マシンバックアップでのチェックポイント維持 30
  - E. タイムアウトチェックポイント強化
  - F. あらゆる命令境界での精細状態維持と回復
  - G. チェックポイントされたデータ量を減らすため所定の命令についてマシンを同期させる方法
  - H. チェックポイントされたデータ量を減らすためレジスタリネームマップをチェックポインティングする方法
- V. 誤予測および例外からの回復
  - A. 同時複数未解決分岐 / ジャンプ・アンド・リンク命令評価のためのウォッチポイントによる誤予測検出 40
    - 1. ウォッチポイント要素の起動
    - 2. データフォワードバスから直接 CC データのウォッチポイントグラブリング
    - 3. ウォッチポイントオペレーションの例
    - 4. 分岐命令の評価
    - 5. ジャンプ - リンク (JMP L) 命令の評価
  - B. 例外検出
    - 1. 発行トラップ検出
    - 2. 実行トラップ検出
  - C. プロセッサを初期状態にバックトラックすることによる回復
    - 1. チェックポイント境界へのプロセッサバックアップ
    - 2. 誤予測された命令、RED モード、および実行トラップ開始バックアップ 50



### 3. あらゆる命令境界へのプロセッサバックステップ

D. 例外 & 誤予測回復中のプライオリティロジックおよび状態マシンオペレーション

E. トラップスタックによるトラップの取扱

【0041】

#### I. CPUオペレーション概要

CPU51は、レベル1(L1)命令キャッシュ52から命令をフェッチし、レベル1(L1)データキャッシュ53にデータを記憶し、ここからデータを取り出す。L1命令およびデータキャッシュは、CPU51と同じチップ/基板に物理的に配置することもできるが、異なるチップ/基板に組み立てることもできる。

【0042】

CPUはまた、メモリ管理制御ユニット(MMU)54とインタラクトして、データプロセッサ50の全体状態を決定する。特に、MMUは外部メモリ56から命令を取り出し、外部メモリ56にデータを記憶し、ここからデータを取り出し、外部I/Oデバイス57からデータを受取り、ここへデータを出力し、外部診断プロセッサ58、CPU51、キャッシュ52および53に診断情報を提供し、ここから診断情報を受取り、データアドレス翻訳を実行し、メモリ状態情報を記憶・追跡する。

【0043】

図5は、予測されたプログラム制御命令、固定および浮動小数点命令、およびロード/ストア命令を含む一部の典型的な命令タイプに関する、CPU51の新規命令パイプラインの段階(および関連するマシンサイクル)を示す。このパイプラインを実行するため、図4に示すように、CPU51は分岐ブロック(BRB)59、発行ブロック(ISB)61およびデータフローブロック(DFB)62を含む。

【0044】

図4および図5を参照すると、BRB59はフェッチ段階中に命令をフェッチし、これらの命令をISB61に与える。BRB59は、プログラム制御命令の各種タイプについて、ターゲットアドレス予測と分岐取得・不取得予測を行う。そのため、これはフェッチ段階中に投機的に命令をフェッチすることができる。

【0045】

ISB61は、命令が投機的にフェッチされた可能性があるため、発行段階中に予測されたプログラムカウンタ(PC)順でフェッチされた命令を発行する。これはまた、予め定義された間隔に当たる命令と、BRB59が予測を行ったプログラム制御命令を含む予め定義された種類の発行された命令について、発行段階中にCPU51のマシン状態のチェックポイントを形成する。

【0046】

BRB59は、ISB61による発行と同じ段階で、nopと予測されたプログラム制御命令を実行、完了する。但し、DFB62は、必要なリソースが利用可能になり次第、発行された固定小数点および浮動小数点命令を実行、完了する。さらに、DFB62は、必要なリソースが利用可能になった時、実行・完了した時点で例外(例:発行トラップ、実行トラップまたは割り込み)やプログラム制御誤予測の場合に元通りにする必要のないやり方で、発行されたロード/ストア命令を実行のために積極的にスケジューリングする。そのため、一部の固定小数点、浮動小数点、およびロード/ストア命令の発行段階が他のものより早くても、必要なリソースはまだ利用可能になっていないため、実行および完了段階が後になる。さらに、これらはBRB59による初期のプログラム制御予測に基づき、BRB59が投機的にフェッチした可能性があるため、投機的に実行および完了されたかもしれない。言い換えると、DFB62に発行された命令は、予測されたPC順序を無視して実行・完了されることがある。

【0047】

命令がエラー発生(例:発行トラップ、実行トラップまたは誤予測)なしに完了すると、非起動段階でISB61によって非起動にされる。命令は予測されたPC順序を無視して完了することがあるので、誤予測のPC順序を無視して非起動とされることがある。

10

20

30

40

50

## 【 0 0 4 8 】

そこで、非起動命令は、コミット段階に実際のPC順でISB61によってコミットされる。非起動の命令は、先行する発行済み命令をすべてコミットしていなければコミットされないの、真である。その結果、命令がコミットされると、例外または誤予測の場合、元通りにできない。これはすなわち、この点までの実際のPC順序が正しく、命令は実際のPC順でコミットされたことを意味する。さらに、チェックポイントを形成した命令をコミットすると、先行のチェックポイントをリタイアすることができる。

## 【 0 0 4 9 】

命令をコミットすると、リタイア段階に実際のPC順でISB61によってリタイアされる。命令リタイア後、それに割り当てられたリソースはリクレームされ、他の命令が発行された時にそれらに再割り当てすることができる。

10

## 【 0 0 5 0 】

ISB61は、パイプライン中のあるポイントで発生する実行またはプログラム制御の誤予測からの回復を行う。そのため、プログラム制御誤予測が発生すると、ISB61は、不正プログラム制御命令が発行された時、形成したチェックポイントにCPU51をバックアップする。同様に、実行トラップが発生すると、ISB61はまず、不正命令発行後にチェックポイントが形成されていれば、最も早いチェックポイントにCPU51をバックアップし、および/または不正命令にCPU51をバックステップする。チェックポイントへのバックアップおよび/またはバックステップにおいて、CPU51は、不正命令の発行および実行直前に存在した正しいマシン状態に戻る。

20

## 【 0 0 5 1 】

ISB61が、プログラム制御誤予測を生じさせた不正命令にバックアップされていると、BRB59は、正しいプログラムカウンタ値と正しいマシン状態で命令のフェッチを始める。しかし、ISB61が実行トラップを生じた命令にバックアップされているか、および/またはバックステップされていると、BRB59に、適切なトラップ取扱ルーチンのターゲットプログラムカウンタ値が与えられる。そこで、トラップ取扱ルーチンはトラップを処理し、不正命令のプログラムカウンタ値を戻すか、次の命令をフェッチするためBRB59に次のプログラムカウンタ値を戻す。不正命令のプログラムカウンタは、トラップ取扱ルーチンがCPU51に不正命令をフェッチし、その発行と実行を再試行しよう命じると、戻される。しかし、トラップ取扱ルーチンがCPU51に不正命令後に次の命令をフェッチして、不正命令の発行と実行をスキップしよう命じると、次の不正命令のプログラムカウンタが戻される。

30

## 【 0 0 5 2 】

II. 命令トラッキング

前に言及したように、CPU51は、CPU51の命令パイプラインに命令トラッキングのための新規設計および方法を採用している。パイプラインは、この設計および方法を実施するため、図5に示し、前に簡単に述べたように、フェッチ、発行、実行、完了、非起動、コミットおよびリタイアの段階を含む。

## 【 0 0 5 3 】

A. 命令フェッチ

再び図4を参照すると、BRB59はフェッチ段階中に命令をフェッチし、これらをISB61に与える。図6に示すように、BRB59は、命令プリフェッチおよびキャッシュユニット(IPCU)100とフェッチユニット102を含む。

40

## 【 0 0 5 4 】

IPCU100は、レベル1(L1)命令キャッシュ52から1度に4個の命令(INSTs)を取り出すことができる。取り出された命令は、IPCU100の命令レコーダによって、BRB59、ISB61およびDBF62の利用により適したフォーマットにレコードされる。別の実施例では、IPCU100は、1度に4個より多いか少ない命令を取り出して実施することもできる。

## 【 0 0 5 5 】

50

フェッチユニット102は、フェッチプログラムカウンタ(FPC)値、アーキテクチャルプログラムカウンタ(APC)値、および次のアーキテクチャルプログラムカウンタ(NAPC)値を計算するプログラムカウンタ(PC)ロジック106を含む。各マシンサイクルで計算されたFPC、APCおよびNAPCの値はそれぞれ、PCロジックのFPC、APCおよびNAPCレジスタ112から114に記憶される。FPCレジスタの現在のFPC値は、現在のマシンサイクルでフェッチされている命令を指すが、APCおよびNAPCレジスタ113および114の現在のAPCおよびNAPC値は、現在のマシンサイクルでISBが発行可能な最初の命令と、前回のマシンサイクルでフェッチされた次の命令を指す。

#### 【0056】

FPC値に対応して、1度に4個の命令(F\_\_INSTs)がIPCU100によってフェッチされる。別の実施例では、1度に4個より多いか少ない命令をフェッチすることができる。

#### 【0057】

各FPC値について、分岐履歴表104には、FPC値でフェッチした各命令の分岐予測フィールド(BRP)が含まれる。そのため、BRPフィールドはそれぞれFPC値でフェッチした命令の内の1つに対応する。フェッチした命令のいずれかが、SPARC-V9B Pcc、Bicc、BPr、FBfccおよびFBPfcc命令など条件付き分岐命令である場合、これらの対応するBRPフィールドは、分岐を、分岐命令のこの後の発行、実行および完了段階で予測すべきか否かを識別する。分岐履歴表104は、FPC値に対応してフェッチした命令にBRPフィールドを出力する。そこでBRPフィールドは、フェッチユニット102によってフェッチした対応する命令(F\_\_INSTs)に追加され、フェッチ回転レジスタ110が受け取る合成命令を与える。

#### 【0058】

フェッチレジスタには、1度に4個の命令を保持するため4個のラッチがある。このラッチは、予測されたPC順で、ISU200が発行できる次の4個の命令の入った4個のスロット(0-3)を有する発行ウィンドウを定義する。但し、前回のマシン中にフェッチレジスタの保持した命令の一部は、これから説明するような発行上の制約のため、ISU200が発行していないことがある。フェッチされた命令が予測されたPC順で発行されるようにするため、ISU200はフェッチレジスタ制御(FCH\_\_REG\_\_CNTL)信号を生成するが、この信号は、最後のマシンサイクルで発行されなかった命令を該当するラッチに入れるようフェッチレジスタを制御することで、発行ウィンドウ内で予測されたPC順にし、ISU200が予測されたPC順で発行できるようにする。さらに、FCH\_\_REG\_\_CNTL信号に対応して、フェッチレジスタは残りのラッチに、予測されたPC順で次に来るフェッチしたばかりの命令を入れる。

#### 【0059】

別の実施例では、フェッチレジスタは、1度に4個より多いか少ない命令を記憶するよう実施できる。さらに、例示CPU51ではフェッチユニット102に1個のフェッチレジスタとして説明したが、今説明した種類のフェッチレジスタは、デコードを実行するCPUの各ブロックに使うことができる。そして、フェッチレジスタをこれらブロックに設置して、デコードした命令をフェッチレジスタに記憶する前、フェッチサイクル中に初期デコードを行い、残りのデコードを発行段階に行うようにすることができる。

#### 【0060】

### B. 命令発行

#### 1. 命令発行概観

ISB61は、各発行段階中にフェッチした命令を発行する。図7に示すように、ISB61は、発行ユニット(ISU)200、精細状態ユニット(PSU)300、フリーリストユニット700、および制御レジスタファイル800を含む。各発行段階中、ISU200は、BRB59のフェッチレジスタ110から、一度に4個の命令(F\_\_INSTs\_\_BRP)を受け取る。そして、これらをデコードして、その内いくつを発行するか

10

20

30

40

50

を、次に説明するような各種発行上の制約に基づいて決定する。

#### 【0061】

フェッチレジスタ110は、各発行段階中、ISU200に一度に4個の命令を与えるため、ISU200は各発行段階中、最大4個の命令を発行することができる。しかし、各発行段階の発行ウィンドウで予測されたPC順で4個の命令を発行することしかできない。そのため、他の発行上の制約から、現在の発行ウィンドウで命令の内の1つが発行できない場合、この命令の前の発行ウィンドウスロットの命令しか、現在の発行段階では発行できない。別の実施例では、フェッチレジスタ110は、1つの発行段階あたり4個より多いか少ない命令をISU200に与えるよう構築されているため、ISU200は、1つの発行段階あたり4個より多いか少ない命令を発行するよう構築することができる。図8を参照すると、FPU600、FXU601、FXAGU602およびLSU603は、発行された命令について、これらにディスパッチされた命令データを記憶するためのリザベーションステーションあるいは待ち行列を有する。但し、発行段階中、リザベーションステーションの一部は、ディスパッチされた命令データを記憶できるだけの記憶要素または記述項を持たないことがある。そのため、FPU600、FXU601、FXAGU602およびLSU603は、それぞれFPU600、FXU601、FXAGU602およびLSU603のリザベーションステーションに、ディスパッチされた命令データを受け取ることのできる記述項がいくつあるかを示すENTRIES\_\_AVAIL信号を出力する。その結果、ISU200は、ENTRIES\_\_AVAIL信号で示された利用可能な記述項の数に基づき、浮動小数点、固定小数点、およびロード・ストア命令を発行する。

10

20

#### 【0062】

さらに、例示CPU51では、FXU601が、固定小数点計算に関わるプログラム制御、乗算/除算、および制御レジスタ読み取り/書き込み命令を実行するDFB62の唯一の実行ユニットである。そのため、この場合、ISU200はFXU601による実行のためこれらの種類の命令を発行する。別の実施例では、FXAGU602も、固定小数点データに関わるプログラム制御、乗算/除算、および制御レジスタ読み取り/書き込み命令を実行するよう構成することができる。この場合、ISU200も、FXAGU602による実行のためこれらの種類の命令を発行することになる。さらに、実行ユニット600から603はそれぞれ1以上の機能ユニットからなり、各機能ユニットは発行された命令を実行することができる。そのため、発行することのできる命令の数は、各実行ユニット内の機能ユニットの数で決まる。

30

#### 【0063】

図7を参照すると、PSU300は、ISU200による命令発行に発行上の制約を課すこともできる。PSU300はISSUE\_\_KILL信号を出力することができ、これは、フェッチレジスタ110から受け取った命令の内どれを現在の発行段階中に発行してはならないかを識別するものである。後により詳しく説明するように、例外検出後か、次に説明する同期命令の実行、完了、非起動およびリタイア中に、PSU300が不正命令にバックアップまたはバックステップしている時、ISSUE\_\_KILL信号がアサートされる。

40

#### 【0064】

特定の種類の命令については、ISU200は、これら同期命令の内1つを発行する前に、CPU51が同期されることを確かめる。CPU51は、特定の命令に先行するすべての命令が発行、実行、完了、非起動およびリタイアした時、その命令について同期する。そのため、ISU200がフェッチレジスタ110から受け取った命令の内1つが同期命令であると決定すると、その前の発行ウィンドウスロットにある命令を発行し、CPU51が同期したことを示すMACHINE\_\_SYNC信号をPSU300から受け取るまで待つ。これが発生すると、命令は最も早いPC値(すなわちスロット0)で命令の発行ウィンドウのスロットに入り、ISU200はただそれを発行する。例示CPU51は、特定の命令種類について、前に説明したマシン状態同期方法を実施するが、別の技術では

50

、コミットした命令は非投機的であることが保証されるため、前回の命令のコミット後に同期命令の発行ができる。

【0065】

ISU200はまた、SPARC-V9 Architecture Manualに説明するような、命令の発行に影響する発行トラップを検出する。ISU200は、制御レジスタファイル800からレジスタ(CNTRL\_\_REG)フィールドを受取り、フェッチレジスタ110から受け取った命令(F\_\_INSTs\_\_BRPs)をデコードして、発行段階中に一定の種類の発行トラップが発生したか否かを検出する。SPARC-V9アーキテクチャに基づく実施例では、これはSPARC-V9 Architecture Manualに従って行われる。さらに、他の種類の発行トラップはレジスタ制御を必要とせず、ISU200のデコードした命令操作コードに基づき、取得・検出される。

10

【0066】

最も早い発行ウィンドウスロットの命令によって生じた発行トラップのみが取得される。そのため、1以上の発行トラップが検出されると、最早スロットの命令の原因となった発行トラップのスロットより前の発行ウィンドウスロットの命令のみ、ISU200によって発行することができる。さらに、しばしば発生する発行トラップについては、CPU51は、前に述べたような方法で同期し、発行トラップを投機的に取得しないようにする。別の実施例では、発行トラップがスロット0でのみ発生するようCPU51を構成して、ロジックを減らし、簡素化することができる。

【0067】

各発行段階中、ISU200は発行される各命令にシリアル番号を割り当て、これらシリアル番号(SNs)をDFB62にディスパッチする。後でより詳細に説明するように、PSU300は、割り当てられたシリアル番号を使って発行された命令を記録する。さらに、例示CPU51では、発行されたがまだリタイアしていない命令を一度に所定の数だけ記録することができる。しかし、発行された命令をリタイアすると、それらのシリアル番号が利用可能となる。そのため、PSU300は、現在の発行段階にいくつのシリアル番号が利用できるかを示す信号を含むSN\_\_AVAIL信号と、これら利用可能なシリアル番号を含む信号を、ISU200に与える。現在の発行段階中、SN\_\_AVAIL信号が、現在利用可能なシリアル番号の数が発行できる命令の数より少ないことを示すと、シリアル番号を割り当てることのできる最も早いスロットの命令だけが実際に発行される。

20

30

【0068】

やはり発行段階中、ISU200は、ある命令についてCPU51のマシン状態のチェックポイントを形成すべきか否かを決定する。後でより詳細に説明するように、チェックポイントは一定の種類の命令について、所定の発行段階間隔で形成される。そのため、発行する命令にチェックポイントが必要とISU200が決定し、および/またはPSU300が、チェックポイントを形成した最後の発行段階から所定数の発行段階が発生したことをTIMEOUT\_\_CHKPT信号によって示すと、ISU200はDO\_\_CHKPT信号を生成し、PSU300、制御レジスタファイル800、BRB59およびDFB62に、MCPU51のマシン状態のチェックポイントを形成するよう指示する。

40

【0069】

さらに、ISU200は、発行した各命令にチェックポイント番号を割り当て、これらチェックポイント番号(CHKPT\_\_Ns)をDFB62にディスパッチする。そして、チェックポイントを形成した各命令に、新しいチェックポイント番号が割り当てられる一方、チェックポイントを形成しない命令には、前のチェックポイントのチェックポイント番号が割り当てられる。PSU300は、割り当てられたチェックポイント番号を使って形成されたチェックポイントを記録する。

【0070】

割り当てられたシリアル番号同様、PSU300は、形成されたがリタイアしていないチェックポイントを一度に所定の数だけ記録する。但し、チェックポイントがリタイアさ

50

れると、チェックポイント番号が利用可能になるため、P S U 3 0 0 は、現在の発行段階にいくつのチェックポイント番号が利用できるかを示す信号を含む C H K P T \_ \_ A V A I L 信号と、利用可能なチェックポイント番号を含む信号を、I S U 2 0 0 に与える。そのため、C H K P T \_ \_ A V A I L 信号が、現在利用可能な新しいチェックポイントの数が新しいチェックポイントを形成しなければならない命令数より少ないことを示すと、新しいチェックポイント番号を割り当てることのできる最も早い発行ウィンドウスロットの命令だけが実際に発行される。

#### 【 0 0 7 1 】

さらに、C P U 5 1 は、1つの発行段階につき所定の数のチェックポイントを形成するようにのみ構成することができる。そのため、たとえばC P U 5 1 が1つの発行段階につき1つしかチェックポイントを形成できず、チェックポイントを形成する必要のある発行ウィンドウに2個の命令がある場合、I S U 2 0 0 は、現在の発行段階中、最も早いスロットにある命令のみ発行し、別の発行段階の他の命令を発行する。

10

#### 【 0 0 7 2 】

後で説明するように、C P U 5 1 は、プロセッサのスループットを上げるためレジスタリネーミングを採用している。レジスタリネーミングを正しく実施するため、I S U 2 0 0 はレジスタリネーミングフリーリストユニット ( F R E E L I S T ) から R E G S \_ \_ A V A I L 信号を受け取る。図 8 を参照すると、これら信号には、固定小数点レジスタファイルおよびリネームユニット ( F X R F R N ) 6 0 4 のいくつの物理的固定小数点レジスタをリネーミングに利用できるか、浮動小数点レジスタファイルおよびリネームユニット ( F P R F R N ) 6 0 5 のいくつの浮動小数点レジスタをリネーミングに利用できるか、浮動小数点状態および条件コードレジスタファイルおよびリネームユニット ( F S R / C C R F R N ) 6 0 6 のいくつの物理的整数および浮動小数点条件コードレジスタをリネーミングに利用できるかを示す信号が含まれる。図 7 を参照すると、フェッチレジスタ 1 1 0 から受け取った命令のいずれかがレジスタリネーミングを必要とし、物理的レジスタがリネーミングに必要であるが、R E G S \_ \_ A V A I L 信号によって利用できないことが示されると、I S U 2 0 0 はこれらの命令を発行できない。

20

#### 【 0 0 7 3 】

前記発行上の制約は、発行決定の概念を例証するため、例示C P U 5 1 に関連して説明した。さらに、これまで説明した発行上の制約とは異なるが、本書に説明した発行決定の基本的概念から逸脱しない実施も存在できる。そのため、当業者は、発行上の制約が実施依存であり、本書に説明する発明は、これまで説明した発行上の制約に限定されないことを認識するだろう。

30

#### 【 0 0 7 4 】

### 2 . 命令シリアル番号割り当てと命令状態記憶

例示C P U 5 1 は、命令が発行された時点から、命令の実行が完了して究極的にリタイアするまで、命令に関連する独自の識別タグを割り当てる。逐次シリアル番号は、命令識別タグとして便宜上用いられる。シリアル番号は各段階で用いられ、例外または誤予測が発生すると、例外フリー処理中およびC P U 5 1 回復中に実質的にすべてのブロックが用いられる。命令状態情報はC P U 5 1 に記憶され、実行完了信号、実行エラー信号、および分岐命令誤予測信号等の状態の変化に対応して連続的に更新される。I S B 6 1 内の精細状態ユニット ( P S U ) 3 0 0 が、命令シリアル番号タグの割り当てと、他のC P U 5 1 ユニット、特にD F B 6 2 から受け取った信号に対応して命令状態のトラッキングを行う。

40

#### 【 0 0 7 5 】

図 9 を参照すると、精細状態ユニット ( P S U ) 3 0 0 は機能的にI S B 6 1 内にあり、( 1 ) C P U 5 1 での命令の発行、実行、完了およびリタイア状態のトラッキング；( 2 ) 分岐誤予測回復の検出と開始；( 3 ) 命令シリアル番号、チェックポイント、およびウォッチポイントを含む一定のマシンリソースの利用可能性のトラッキング；および( 4 ) 一般例外取扱および制御、を行う。P S U 3 0 0 内のいくつかのユニットは、図 9 に示し、後により詳細に説明するように、この機能性を達成するため実施される。

50

## 【 0 0 7 6 】

PSU300は、発行されたすべての命令について、これらがリタイアされるまでアクティビティ状態情報を維持する発行/コミット/リタイアユニット(ICRU)301を含む。ICRU301は、ISU200が次の命令発行サイクルで用いる4個までの命令シリアル番号(SNs)を識別する信号(SN\_\_AVAIL)をISU200に与える。ISUが命令を発行すると、ISU200は、前のサイクルでISUに与えられた(SN\_\_AVAIL)最大4個のシリアル番号のうちどれがISUによって有効に発行されたかを、前に示したように、どのシリアル番号が割り当てられ、どのシリアル番号が各命令に関連するかを示すISSUE\_\_VALID信号の形でICRUに知らせる。発行上の制約によって、ISUが命令ウィンドウのすべての命令を発行する能力が制約を受けることを思い出してほしい。ICRU301は状態情報と状態情報へのアドレスポイントを記憶・維持し、命令の種類にかかわらず、ISU200が有効に発行した各命令の命令発行、実行、完了、非起動、コミット、リタイアフェーズ状態をトラックする。ISU200はまた、発行された命令の内どれがnopおよび一定のプログラム制御命令に関連し、発行と同じ段階で非起動が可能かを、ISSUE\_\_NOP信号でICRU301に知らせる。ISSUE\_\_NOPは、ポーズや遅延を導入するような、命令ストリームに挿入できる特定の「nop」命令に限定されない。CPU51外のロード/ストアその他命令参照メモリなどの長待ち時間命令を積極的にスケジューリングする際、ISU200は、スロット0-3の命令のどれが長待ち時間かを識別するISSUE\_\_MEM信号の形で、ICRU301にも知らせる。(長待ち時間命令の積極的スケジューリングは、基本的な本発明の構造および方法の強化として別に説明する。)

10

20

## 【 0 0 7 7 】

a. 発行/コミット/リクレームユニット(ICRU)

発行/コミット/リクレームユニット(ICRU)はPSU300内で機能し、n-ビット逐次シリアル番号などの命令タグを割り当て、CPU51内部のメモリで命令状態情報のデータ記憶領域を定義・維持することにより、発行されたすべての命令のアクティビティ状態情報を維持する。図10に、ICRU301のコンポーネントおよびICRUオペレーションに関連する入出力信号の機能ブロック図を示す。ICRU301は機能的に4つの領域で構成される。命令状態情報データ構造308は、命令状態情報の記憶と、データ構造制御ロジック309の信号に対応して状態を更新を行い、データ構造制御ロジック309は、他のCPU51ユニット、特にDFB62からの信号に対応する。ポインタデータ構造310は、発行、完了、リタイア等、CPU51の各種命令段階マイルストーンへの状態ポインタとして働く複数のシリアル番号を個別に記憶するデータ記憶領域を含む。これらシリアル番号ポインタは、後でより詳細に説明する。ポインタ制御ロジック311は、他のCPU51ユニットから受け取ったデータと共に、データ構造308に記憶された状態情報を評価し、シリアル番号ポインタを更新して、カレントCPU51状態を反映させる。ポインタ値は、ICRU301およびPSU300からCPU51の他ユニットへのグローバル出力として与えられる。

30

## 【 0 0 7 8 】

命令状態情報データ構造308は、アクティブ命令リングレジスタ(A-リング)312からなり、オプションでメモリ命令リングレジスタ(M-リング)324からなることもできる。メモリ命令リングレジスタは、積極的ロード/ストア命令スケジューリングに関連して本明細書中の他の部分で説明する。データ構造制御ロジック309は、A-リングセット/クリアロジック313からなり、オプションでM-リングセット/クリアロジック325からなることもできる。ポインタ制御ロジック311は、ISN/NISN初期化および前進ロジック321からなり、次の命令シリアル番号割り当てロジック322と、CSN/RRP前進ロジック323は、オプションでNMCSN前進ロジック326、マシン同期生成ロジック327、バクトラックモードポインタ調整ロジック328からなる。オプション要素は本発明の基本的な構造および方法の強化に関連し、後でより詳細に説明する。

40

50

## 【 0 0 7 9 】

発行 / コミット / リクレームユニットの命令発行段階のオペレーションを、図 10 を参照して説明する。発行された命令に割り当てるため I S U 2 0 0 から送られた一覧番号 ( S N \_ \_ A V A I L ) は、ポインタデータ構造 3 1 0 内のポインタ値に基づき I C R U のシリアル番号割り当てロジックで選ばれる。C P U 5 1 を初期化した時 ( パワーアップ時など )、ポインタデータ構造 3 1 0 内のポインタも初期化され、I C R U はまず、発行ウィンドウの命令について最初のシリアル番号を I S U 2 0 0 に割り当てる ( S N の 0 - 3 等 )。

## 【 0 0 8 0 】

I S U 2 0 0 はデータ構造制御ロジック 3 0 9 に I S S U E \_ \_ V A L I D 信号を送り、前回のサイクルで与えられたシリアル番号の内どれが ( 4 個の S N の内どれが ) I S U 2 0 0 によって有効に発行され、どの命令スロットのものを I C R U に知らせる。セット / クリア A - リング制御ロジック 3 1 3 はこの情報とポインタレジスタ 3 1 0 内のポインタの値を使って、A - リング 3 1 2 に命令状態情報を書き込む。C P U 5 1 の 1 実施例では、I S S U E \_ \_ V A L I D は 4 ビットベクトルで、I S U 2 0 0 が命令シリアル番号 ( S N \_ \_ A V A I L ) のどれを実際を使って命令を実際に発行した命令スロットを識別したかを I C R U に知らせる。そこで I C R U は、S N \_ \_ A V A I L と I S S U E \_ \_ V A L I D に基づきどの S N を使ったかを決定できる。

10

## 【 0 0 8 1 】

アクティブ命令リング 3 1 2 の実施例を、図 1 1 の略図を参照して説明する。図 1 1 のデータ構造は例示なもので、本特許の内容から、このアクティブ命令状態レジスタの実施には様々なデータ構造を利用できることを当業者は理解するであろう。図 1 1 は、6 4 ビットデータ構造として実施したアクティブ命令リング ( A - リング ) 3 1 2 を示す。(メモリ命令リング ( M - リング ) 3 2 4 も略図として示し、長待ち時間命令の積極的スケジューリングの強化に関連して本明細書の他の部分で説明する。) A - リング 3 1 2 内の 6 4 の各アドレス指定可能ロケーションは、C P U 5 1 の 1 つの命令シリアル番号に対応する。各アクティブビット ( A - ビット ) のセット ( 「 1 」 ) またはクリア ( 「 0 」 ) 状態は、命令がアクティブであるか ( 「 1 」 )、インアクティブであるか ( 「 9 」 ) を示す。基本的に、発行されて、エラーなしに実行を完了し誤予測されずに非起動とさえると、命令は始動する。1 つのマシサイクルで発行され効果的に実行完了した一部の命令では、A - ビットが命令発行でクリアされる。A - リング 3 1 2 のアドレス指定可能ロケーションの数は、C P U 内の同時未解決の命令の数を限定する。

20

30

## 【 0 0 8 2 】

例示プロセッサでは、A - リング 3 1 2 は円形、環状あるいは循環データ構造として実施され、ここでポインタは「モジュロ A - リングレジスタ長さ」タイプ演算 (ここでは、モジュロ 6 4 タイプ演算) を使って、データ構造に沿って移動する。すなわち、ポインタが第 1 のデータ記憶場所 ( アドレス指定可能ビット 0 ) から一連の中間記憶場所 ( ロケーション 1 5、1 6 . . . 等 ) を通って最後のデータ記憶場所 ( アドレス指定可能ビット 6 3 ) までインクリメンタルに移動すると、最初のロケーション ( アドレス指定可能ビット 0 ) に戻る。当業者は、本発明の内容から、環状データ構造は有利ではあるが、本発明にとって不可欠ではなく、本発明の特徴を実施するのに他のデータ構造も使えることを理解するであろう。さらに、例示 A - リングは必要な状態情報の維持に 6 4 個のシングルビットのアドレス指定可能ロケーションを持つが、これらおよび追加状態インジケータの記憶に、マルチビットアドレス指定可能ロケーションを設けることもできる。たとえば、ある実施例では、任意回転の 6 4 ビットレジスタを用い、第 2 の実施例では、8 個の 8 ビットレジスタを用いて 6 4 個の A - リング 3 1 2 A - ビットとしている。単純なデコーディング回路が、後に説明するようにビット書き込み ( セットとクリア ) を可能にする。

40

## 【 0 0 8 3 】

I S U 2 0 0 による「命令発行」の結果、命令に命令シリアル番号が割り当てられる。割り当て可能なシリアル番号の数は、A - リング 3 1 2 のアドレス指定可能ロケーション

50



(ビット位置等)の数によって限定される。そのため、各アドレス指定可能ロケーションは独自のシリアル番号に対応する。たとえば、例示64ビット環状A-リングには、64個までの命令シリアル番号を同時に割り当てることができる。A-リングのロケーションを多くか少なくすることによって、これより多くか少ないシリアル番号を割り当てることができる。すべてのシリアル番号を割り当てると、シリアル番号とA-リングのロケーションがその後のマシンサイクルで解放されるまで、それ以降の命令発行をISU200によって機能停止しなければならない。ISU200は、SN AVAIL信号によってSNが利用可能か否かを知らされる。A-リング312の各アドレス指定可能ロケーションは利用可能なシリアル番号の1つに対応し、発行済みシリアル番号ポインタ(ISN)314を発行された各命令について1つ前進させることで、新しく発行された各命令にA-リングに記述項を作る。1つのマシンサイクル中に複数の命令が発行されることがあるため、各サイクルで1個以上のA-ビットを設定し、ISNを各サイクルで1つ以上のA-リングロケーションで前進させることができる。

10

20

30

40

50

#### 【0084】

ISU200による「命令ディスパッチ」は、実行のための(シリアル番号を割り当てられた)命令の起動とDFB62への送信である。ディスパッチされたすべての命令は発行された命令でもあるが、発行された命令のすべてがディスパッチされるわけではない。「ディスパッチ」されたすべての命令は、「1」に相当してセットされたA-リング312中の割り当てシリアル番号に対応し、その命令がマシン中でアクティブであることを示す(1=アクティブ、0=インアクティブまたは非起動)。発行されたがディスパッチされない命令は既知の分岐またはnopタイプの命令に対応し、その命令についてA-ビットが発行時にクリアされる(またはセットされない)。これら既知の分岐とnopタイプの命令は、実行のための実行ユニットを必要とせず、1つの段階で発行、実行、完了することができる。

#### 【0085】

##### b. 発行ステージ間のポインタ関連の命令発行およびポインタメンテナンス

図11は、ポインタ記憶メモリ領域310に記憶され、A-リング312およびM-リング324上のロケーションを指す数個のポインタ(ISN、NISN、CSN、RRP、NMCSN、およびPBSN)も示す。それぞれのポインタは、A-リングのロケーション値0~64の1つを記憶するポインタ記憶ユニット210の構成要素である。ポインタの組み合わせによっては、A-リングまたはM-リングのロケーションを指し示すことがあるが、ロケーションを指し示すことのない組み合わせもある。例えば、NISNはISNと等しくなることはあり得ないが、装置によっては、CSN=ISN=RRPと「同期」する。発行されたシリアル番号ポインタ(ISN)314および次発行のシリアル番号ポインタ(NISN)315は、命令発行状態を保持し追跡すると共に、全体的に他のポインタの利点に制約を加える。コミット(委託)されたシリアル番号ポインタ(CSN)316は、他のポインタには特に規定されていないような命令の実行、完了、および非活動化に続く命令コミットを追跡する。リタイア再生ポインタ(RRP)317は、命令のリタイアを追跡し、リソースの再生を制御する。最速予測分岐命令ポインタ(PBSN)318および非メモリコミットシリアル番号ポインタ(NMCSN)319の2つの補助的なポインタを使用して、予測分岐命令やロード/ストア命令のような待ち時間の長い命令の実行をスケジュールし追跡する。ICRU301はCPU51内の多数の他のユニットに対して、それぞれのISN、NISN、CSN、RRP、およびNMCSNに現在値を与える。ICRUは、以後説明するように、PSU300内のウォッチポイント・ユニット308からPBSN318の値を受け取る。これらのポインタは、実施例のCPUではそれぞれ6ビットベクトルとして実施される。

#### 【0086】

発行されたシリアル番号ポインタ(ISN)314は、常に最後に発行された命令のシリアル番号を指し示している。ICRUによって与えられたシリアル番号の1つが実際にISU200によって割り当てられたとき、命令が発行されたと考えられる。ISNおよ

び N I S N は、 I C R U にそのサイクルの間に実際に発行された命令の数を通知する I S U 2 0 0 からの ISSUE\_VALID 信号に応じてインクリメントされる。 I S N および N I S N がインクリメントされるので、ポインタは効率良く A - リング 3 1 2 の周りに前進する。マシンサイクルごとに I S N が前進できるポジションの最大数は、マシンのピーク発行速度によって決定されるが、ハードウェアを制御する別のソフトウェアによって、この最大数をサイクル当たりの命令の一層小さな所定の値にまで制限できる。例えば、典型的な C P U では、 I S N および N I S N の前進は、マシンサイクル当たり 4 つの命令のシリアル番号 ( A - リングのロケーション ) に制限されている。 C P U が初期化されると、 I S N は初期化されて 0 ( ゼロ ) になり、別の命令が発行されると、 I S N は新たに発行された命令の数によって前進する。

10

#### 【 0 0 8 7 】

次発行のシリアル番号ポインタ ( N I S N ) 3 1 5 は、常に I S N + 1 を指しているため ( モジユロ A - リング長 )、 I S N が 63 の場合、 N I S N はゼロである。これは本質的に発行されるべき次の命令のシリアル番号である ( すべての命令は予測された P C 順に発行される )。独立したポインタの N I S N を提供することは便利であるが、 N I S N は常に I S N よりも 1 だけ大きいため、実施例の中には I S N だけを使用しているものもある。 C P U が初期化されると、 N I S N は初期化されて 1 ( いち ) になり、別の命令が発行されると、 N I S N は新たに発行された命令の数によって前進する。 I S N および N I S N の初期化と前進とは、 I S U 2 0 0 からの ISSUE\_VALID 信号に応じて、 I S N / N I S N 前進論理ユニット 3 2 1 によって実行される。

20

#### 【 0 0 8 8 】

I C R U 3 0 1 は、命令が発行されたときに割り当てられる 4 つの利用可能な命令のシリアル番号を I S U 2 0 0 に与える。適切な 4 つの命令のシリアル番号の決定は、 I S N 3 1 4 および N I S N 3 1 5 の現在値に基づいて、ポインタ制御論理ユニット 3 1 1 内の次命令シリアル番号割当てロジック 3 2 2 の中で行われる。 R R P 3 1 7 は、 C U がフルの場合、すなわちすべてのシリアル番号がすでに割り当てられ再生できない場合、シリアル番号の割当てを制限できる。 I C R U 3 0 1 は、 S N\_AVAIL 信号を I S U 2 0 0 に送ることによってシリアル番号が利用可能になる場合、発行された命令に割り当てられるべき ( 4 つの ) シリアル番号の次のシリーズを I S U 2 0 0 に通知する。

30

#### 【 0 0 8 9 】

C P U 5 1 のある実施例では、 S N\_AVAIL 信号は発行ウィンドウのスロット 0 ~ 3 において最大 4 つの命令に割り当てられるシリアル番号を識別する 4 つの 7 ビット信号 ( S N\_SLO T\_0, S N\_SLO T\_1, S N\_SLO T\_2, および S N\_SLO T\_3 ) から構成する。 6 つのビットはシリアル番号を表し、 7 番目のビットはシリアル番号妥当性ビットである。そのシリアル番号に対する妥当性ビットがセットされていない場合、命令のシリアル番号は C P U 5 1 内では依然としてアクティブであり、 I S U は使用してはならない。 6 4 ビットの A - リング 3 1 2 がサポートするすべての 6 4 ビットの命令が C P U 5 1 の中でアクティブである場合、妥当性ビットはセットされないことがある。シリアル番号が有効な場合は I S U 2 0 0 は 1 つの命令しか発行できないので、 I S U は次の命令を発行する前に、以前発行した命令が実行およびリタイアするまで待機しなければならないことがある。命令の実行およびリタイアについて、以下に説明する。

40

#### 【 0 0 9 0 】

図 1 2 は、 A - リング・ビットセット/クリア・ロジック ( A S C L ) 3 1 3 の構造を含む状態制御論理ユニット 3 0 9 をより詳細に示している。 A S C L 3 1 3 は、セット論理ユニットおよびクリア論理ユニット 3 3 1, 3 3 3 とそれぞれ関連したアドレスデコード論理ユニット 3 3 2, 3 3 4 ならびにセット論理ユニット 3 3 1 およびクリア論理ユニット 3 3 3 から構成する。セット論理ユニット 3 3 1 は、 I S U 2 0 0 から ISSUE\_VALID 信号を、またアドレスデコード論理ユニット 3 3 2 からデコードされた N I S N ポインタアドレスを受け取る。 A - リングのセット論理ユニット 3 3 1 は次に、 I S U によって実際に発行された命令の数に一致する A - ビットを A - リング書込みポート 3 4 2 を介して

50

「1」にセットする。命令が非活性化されるおよび/またはCPUがリセットされると、A - ビットはクリアされる。書き込みポート342の数を選択して、各サイクルごとに活性化(または非活性化)される命令のメーカをサポートする。M - リング324に関して図12で示した他の構造を、長いレイテンシー(待ち時間)命令の積極的なスケジューリングおよび命令のコミットおよびリタイアと共に以下に説明する。

#### 【0091】

ある実施例では、ISSUE\_VALID 信号はISU200から受け取った4ビットのベクトルであり、このベクトルではポジションビット*i*での表明(位置付けられたときに「1」すなわち高い信号レベルにセットされたビット)は、*i*番目の命令発行ウィンドウスロットにおける命令が発行されていることを示している。ICRU301は、前に説明したSN<sub>S</sub> LOT<sub>i</sub> 信号によって、どのシリアル番号がどのスロットに関係しているかを知っている。命令は連続したプログラムの順番に発行されるので、4つの命令発行CPUのISSUE\_VALID 信号に対して可能な状態は5つしかない("0000", "0001", "0011", "0111", および "1111")。ICRUは、「ノップタイプ」の命令信号(ISSUE\_NOP)もISUから受け取る。この信号は、説明したように命令がノップタイプの命令であるとICRUに通知する。CPU51のある実施例では、ISSUE\_NOP 信号は4ビット信号であり、ポジションビット*i*での表明ではスロット内の*i*番目の命令が「ノーオペ」クラスの命令であることを示す。積極的なローカル/ストア命令のスケジューリングを実行する場合、ISSUE\_MEM 信号は4ビット信号であり、ポジションビット*i*での表明ではスロット内の*i*番目の命令がメモリ参照命令であることを示している。

10

20

#### 【0092】

一般的に、ノーオペ命令は分岐命令のような制御転送命令、またはライトASR、ライトFPS、またはライトPIL命令のいずれかである。ライトASR、ライトFPS、およびライトPIL命令は、DFB62ではなく、対応する制御レジスタ・ユニット800内で実行され、これによりNOPタイプの命令として処理される。従って、その対応するISSUE\_NOP 信号は「1」にセットされ、このためA - ビットは「0」にセットされる。Tcc 命令はCCが不明の場合は発行されない。Tcc 命令は、CCが既知で誤りである場合、NOP 命令として発行される。Tcc 命令は、他のケースではISSUE\_NOP=0 で発行される。これ等の命令は、実行するためにDFB62実行ユニットを必要とせず、一般に発行されたときと同じマシンサイクル内に終了する。対応する命令が、「分岐」(br)命令のような「ノップ」クラスの命令の場合、ISSUE\_NOP 信号は「1」にセットされる。ISSUE\_NOP 信号が「1」にセットされると、対応するA - ビットは「0」にセットされる、すなわち命令は即座に非活性化されまたコミット可能となる。(命令コミットおよびコミット時のA - ビットの変更については、以下に説明する。)

30

#### 【0093】

A - リングの64のアクティビティビットは、CPU内に共存する命令の活性を示す。A - ビットは、使用される前のサイクルごとにセットされまたクリアされるので、初期化する必要はない。ISSUE\_VALID, ISSUE\_NOP, およびNISN信号に従って、A - ビットはセットされる。

40

#### 【0094】

分岐命令は発行時にクリアされるA - ビットを備えているので、追加のステップが実行されて、CSN316が投機的に実行された分岐命令を越えて前進しないようにする(以後の説明を参照のこと)。典型的な実施例では、分岐命令が発行されるときに、分岐条件コードがウォッチポイント・ユニット304に記憶される。ウォッチポイント・ユニット304は、命令の実行をモニタし、投機的に発行された分岐命令が終了するとき、ウォッチポイントは実行結果を必要な分岐条件コードと比較して、投機的な発行を確認する。条件コードを比較することによって、投機的な実行が正確に予測されたかについての決定が行われる。投機的な実行が正確でない場合、(バックアップおよび/またはバックステップのような)CPU51の回復手段が起動されてその実行を取り消し、これによりA - ビットがクリアされているにもかかわらず、CSNが誤予測された分岐を過ぎて前進しない

50

ようにする。そうしないと回復を妨げる可能性がある。誤って発行された命令シーケンスを命令コミットすることを避けるために、マシン回復手段を十分早く起動するための特別な注意を取らなければならない。評価に先立って、保留の命令自体が誤って予測された命令シーケンスがコミットされないようにする。ウォッチポイントおよびCPU51の回復手段については、この明細書の他の場所でさらに詳細に記載する。

【0095】

命令の状態を追跡し詳細な状態を維持するための命令タグの使用法を含む本発明による方法の実施例の態様を、図13～図15のフローチャートで説明する。図13は、状態を追跡し詳細な状態を維持するための命令タグの使用法の実施例の概略フローチャートである。図14は、命令の状態を追跡して詳細な状態を維持するための本発明による方法の実施例を示す概略フローチャートである。図15は、本発明の実施例によるアクティブ命令リングおよびメモリ命令リングに状態情報を書き込みまた維持する方法のフローチャートである。

10

【0096】

### 3. レジスタのリネーミング

レジスタの依存性を取り除いてISU200がマシンサイクル当たり一層多くの命令を発行できるようにするために、CPU51は発行段階の間にレジスタのリネーミングを実行する。これは、1994年10月11日にシェバナウ (Shebanow) 等に授与された米国特許第5,355,457号に記載されている方法、および/または「マイクロプロセサの物理レジスタの使用を調整するための方法と装置 (METHOD AND APPARATUS FOR COORDINATING THE USE OF PHYSICAL REGISTERS IN A MICROPROCESSOR)」という名称の同時継続出願番号第08/388,364号に記載されている方法、および/またはジョンソン (Johnson) の論文「スーパースカラー・マイクロプロセサの設計 (Superscalar Microprocessor Design)」のページ48～55に記載されている方法と同じ方法で実行できる。

20

【0097】

レジスタのリネーミングを実行するために図8を参照すると、DFB62には、FXRFRN604, FPRFRN605, およびFSR/CCRFRN606のCCRFRN610が含まれている。前に簡単に説明したように、FXRFRN604には固定小数点データを記憶するための物理固定の小数点レジスタが含まれ、FPRFRN605には浮動小数点データを記憶するための物理浮動の小数点レジスタが含まれ、CCRFRN610には整数と浮動小数点条件のコードデータを記憶するための物理整数 (固定小数点) および浮動小数点条件のコードレジスタ (XICCおよびFCGS) が含まれている。FXRFRN604, FPRFRN605, およびCCRFRN610は、それぞれ図16に示すように構成できる。

30

【0098】

図16に示すように、レジスタファイルおよびリネームユニット (RFRN) 612には、物理レジスタを含む物理レジスタファイル614が含まれる。RFRNは、物理レジスタファイル614内の物理レジスタへの命令によって規定される論理レジスタおよび/またはアーキテクト・レジスタをマッピングするリネームマッピング・ロジック613も含む。例えば、SPARC-V9命令では、FXRFRN604のリネームマッピング論理は、80の論理128の物理変換固定小数点レジスタに構築された32のマッピングを実行でき、32のアーキテクトまたは論理的な単精度および倍精度の浮動小数点レジスタは、2組の64の単精度 (32ビット) のリネームされた浮動小数点レジスタ (奇数および偶数) にマッピングされ、5つのアーキテクトまたは論理条件コードレジスタ (xicc, fcc0, fcc1, fcc2, および fcc3) は32のリネームされた条件コードレジスタにマッピングする。アーキテクト論理変換マッピングは次の事実に基づいている、すなわちSPARC-V9の固定小数点命令は、図17に示す制御レジスタ・ファイル800のCWPレジスタに記憶されたカレントウィンドウ・ポインタ (CWP) に従って、論理レジスタにマッピングするアーキテクトレジスタを規定する。

40

【0099】

50

RFRNの制御ロジック613は、各発行段階の間にBRB59のフェッチレジスタから命令(F\_INSTS\_BRP)を受け取る。これに回答して、制御ロジック613は命令をデコードし、どの命令がこの特定のRFRNのレジスタファイル614からのソースおよび/または宛先として物理レジスタを必要とするかを求める。

【0100】

リネームマッピング・ロジック615は、先ず始めに、レジスタファイルの中で現在マッピングされた物理レジスタへの命令によって指定されたそれぞれの論理またはアーキテクト・ソースレジスタをマッピングし、TAGS\_R信号の中で適当な実行ユニット(FPU 600, FXU 601, FXAGU 602, または LSU 603)のリザベーションステーションへの対応する物理レジスタタグを与える。それぞれの物理レジスタタグに対して、リネームマッピング・ロジックは、マッピングされた物理ソースレジスタ内のデータがまだソースとして利用可能かどうかを識別するデータ有効(DV)ビットを有する。DVビットは、リネームマッピング・ロジック615によってDV\_R信号の中の適当な実行ユニットのリザベーションステーションに与えられる。さらに、データがDVビットによって示されるように利用可能な場合、それはDATA\_R信号内のマッピングされた物理ソースレジスタによって提供される。

10

【0101】

それぞれのRFRN(すなわち、FXRFRN604, FPRFRN605, および CRFRN610)に対して、FREELISTユニット700は、RFRNのレジスタファイル614のフリーな物理レジスタのすべてのリストを含む。フリーすなわち利用可能なレジスタとは、リタイアしていない命令によって現在使用されていないレジスタのことである。これらの利用可能な物理レジスタは、各サイクルごとにFREELISTユニット700からのFREE\_REGS信号によって識別される。

20

【0102】

従って、論理またはアーキテクト・ソースレジスタがマッピングされた後、リネームマッピング・ロジック615は次にFREE\_REGS信号によって識別された利用可能なすなわちフリーな物理レジスタに対する命令が指定したそれぞれの論理またはアーキテクト宛先レジスタをマッピングし、対応するDVビットをセットしてそのレジスタ内のデータはまだ利用可能でないことを示す。リネームマッピング・ロジック615は次に、新しくマッピングされた物理宛先レジスタに対して物理レジスタタグを与え、また対応するDVビットをTAGS\_R信号およびDV\_R信号の適当な実行ユニットのリザベーションステーションに与える。

30

【0103】

さらにリネーム・ロジック615は、PSU300の再生フォーマットユニット(RRF)302に旧論理またはアーキテクトされたレジスタマッピングの物理レジスタマッピング変換を提供する。前にマッピングした物理宛先レジスタに対する論理すなわちアーキテクトされたレジスタタグをLOG\_OPHYS\_TAGS信号のRRF302に送ることによって、リネームマッピング・ロジック615はそれを行う。それぞれのRFRN612からのLOG\_OPHYS\_TAGS信号は、これらの信号がその特定のRFRNから来ることを示すためのフィールドを含んでいる。

【0104】

RRF302をさらに詳細に図18に示す。このRRFはデータ記憶構造体366を含んでいる。最大64の発行されたがまだリタイアしていない命令にシリアル番号を割り当てることができる典型的なCPU51では、データ記憶構造体366は64のアドレス可能な記憶素子またはエントリを含んでいる。各記憶素子は、命令のシリアル番号の1つに対応している。さらに、データ記憶構造体366の各記憶素子は、FP論理旧物理変換マップフィールドまたはFX論理旧物理変換マップフィールドを、またFC論理旧物理変換マップフィールドまたはXIC論理旧物理変換マップフィールドを含んでいる。各種の構造体はRRF302を実行するために使用できる。ある典型的なRRFは、64レジスタラ29ビットの4つのバンクのインターリーブされたRAMとして実行され、そこではRRFへの書込みサイクルの間、フィールドはバックまたはバンドルされ、ボックス

40

50

ステップの間の読み出しでアンパックまたはアンバンドルされる。

【 0 1 0 5 】

R R F の制御ロジック 3 6 5 は、D F B 6 2 から LOG\_OPHYS\_TAGS 信号を受け取り、F X R F R N 5 0 4 からの浮動小数点レジスタタグ、F P R F R N 6 0 5 からの固定小数点レジスタタグ、および C C R F R N 6 1 0 からの X I C C タグおよび F C C タグをどれが含まれているかを決定する。制御ロジック 3 6 5 は、P S U 3 0 0 の I C R U 3 0 1 から N I S N ポインタを、また I S U 2 0 0 から ISSUE\_VALID 信号も受け取る。N I S N のポインタと ISSUE\_VALID 信号に応答して、制御ロジック 3 6 5 は、LOG\_OPHYS\_TAGS 信号を受け取る発行された命令のシリアル番号を決定する。

【 0 1 0 6 】

典型的な C P U 5 1 では、命令は浮動小数点と F C C レジスタとを同時にまたは固定小数点と X I C C レジスタとを同時に変更できるにすぎないが、固定小数点、浮動小数点、X I C C、および F C C レジスタを同時に変更できない。従って、固定小数点または浮動小数点の論理および旧物理レジスタタグを LOG\_OPHYS\_TAGS 信号の中で受け取るそれぞれの命令に対して、制御ロジック 3 6 5 はこれらのタグ (FP\_TAGS または FX\_TAGS) をシリアル番号を付けて発行された命令に対応する記憶素子の F P 論理物理変換マップフィールドまたは F X 論理物理変換マップフィールドに書き込む。同様に、F C C または X I C C の論理および旧物理レジスタタグを受け取るそれぞれの命令に対して、制御ロジック 3 6 5 は F C C 論理旧物理変換タグまたは X I C C 論理旧物理変換タグを、シリアル番号を付けて発行された命令に対応する記憶素子の F C C 論理物理変換マップフィールドに書き込む。従って、命令が浮動小数点と F C C レジスタとの両方または固定小数点と X I C C レジスタとの両方を必要とする場合、この命令の LOG\_OPHYS\_TAGS 信号に含まれた論理および旧物理レジスタタグは同じ記憶素子に書き込まれる。しかしながら、R R F 3 0 2 は固定小数点、浮動小数点、X I C C、および F C C レジスタを同時に変更する命令に対する論理旧物理変換マッピングを記憶するように構成できることは、当業者は理解されよう。

【 0 1 0 7 】

従って、R R F 3 0 2 は、必要な場合、以前の論理物理変換マッピングまたは関係を再構成できる連続的な履歴台帳として機能する。例えば、SN="X" に対応する命令を取り消すことが必要になる場合、R R F の中で SN="X" をインデックスとして用いて、論理および旧物理レジスタタグを識別する。次に、これ等のタグは適当な R F R N に渡され、SN="X" 命令を実行する直前に存在した物理レジスタマッピングに対する論理レジスタマッピングを回復し、この新たにマッピングされた物理レジスタはフリーリストに戻される。これによりレジスタのリネーミングを効果的に逆にでき、レジスタの状態は命令を実行する直前に存在した状態に戻される。数個の命令を取り消す必要がある場合、同じ手順が逆にしたステップごとに、I S N から始めて SN="X" までデクリメントして加えられる。このプロセスは詳細に後述する。

【 0 1 0 8 】

図 1 9 は、論理ソースレジスタ L 4 1 および L 4 2、および宛先レジスタ L 4 7 を備えた加算命令用のレジスタリネーミングの例である。前述に従って説明すると、論理ソースレジスタ L 4 1 および L 4 2 はリネームマッピング・ロジック 6 1 5 によって物理レジスタ P 5 0 および P 5 2 にマッピングされる。次に、論理宛先レジスタ L 4 7 は、F R E E L I S T ユニット 7 0 0 が提供したフリーな物理レジスタ P 9 9 に再マッピングされ、論理宛先レジスタ L 4 7 用の論理レジスタタグおよび以前マッピングされた (旧) 物理レジスタ P 7 8 が加算命令に割り当てられたシリアル番号 (SN=13) を用いてマッピングするため R R F に与えられる。この例から明らかなように、各「物理」レジスタタグは、(1) フリーリスト 7 0 0、(2) F X R F R N 6 0 4、F P R F R N 6 0 5、または C C R F R N 6 1 0 のリネームマッピング・ロジック、(3) R R F 3 0 2 の 1 つにまた 1 つだけに現れる。

【 0 1 0 9 】

4 . リザベーションステーションへのディスパッチ

10

20

30

40

50

図 8 をさらに参照すると、発行段階の間に、F P U 6 0 0、F X U 6 0 1、F X A G U 6 0 2、および L S U 6 0 3 のリザベーションステーションは S R B 5 9 から F\_INSTS\_BRP 命令を受け取る。これに応答して、それぞれのリザベーションステーションは、これ等の命令からオペコードと即値データとを抽出する。

#### 【 0 1 1 0 】

それぞれの F P U 6 0 0、F X U 6 0 1、F X A G U 6 0 2、および L S U 6 0 3 のリザベーションステーションは、発行段階の間に発行された命令に割り当てられたシリアル番号およびチェックポイント番号を受け取る。これ等の番号は I S B 6 2 から受け取ったシリアル番号および CHKPT\_N 信号が提供する。

#### 【 0 1 1 1 】

浮動小数点データを含む数学上の命令およびリード/ライト命令を実行するために、F P U 6 0 0 のリザベーションステーションは、発行段階の間に、F S R 6 0 7 の F P R F R N 6 0 5 および C C R F R N 6 1 0 から、使用可能なデータ、データ有効ビット、および F P 物理レジスタタグおよび C C 物理レジスタタグ ( F P \_ D A T A \_ R, F P \_ D V \_ R, F P \_ T A G S \_ R, C C \_ D A T A \_ R, C C \_ D V \_ R, C C \_ T A G S \_ R ) を受け取ることができる。同様に、F X U 6 0 1 は、固定小数点を含む数学上の命令、プログラム制御命令、およびリード/ライト命令を実行するために、発行段階の間に、F X R F R N 6 0 4 および C C R F R N 6 1 0 から、I S B 2 D F B バスに対する制御レジスタ・ファイル 8 0 0 からのデータ、使用可能データ、データ有効ビット、および F X 物理レジスタタグおよび C C 物理レジスタタグを受け取ることができる。ロード/ストア命令および固定小数点非乗算/除算の演算命令用のアドレス発生動作を実行するために、F X A G U 6 0 2 は、F X R F R N 6 0 4 および/または C C R F R N 6 1 0 からデータならびに F X 物理レジスタタグおよび C C 物理レジスタタグ ( F X \_ D A T A \_ R, F X \_ D V \_ R, F X \_ T A G S \_ R, C C \_ D A T A \_ R, C C \_ D V \_ R, C C T A G S \_ R ) を受け取ることができる。しかしながら、前に説明したように、プログラム制御、乗算/除算、および制御レジスタのリード/ライト命令を実行するために、F X A G U 6 0 2 は F X U 6 0 1 と同様に構成することができ、この場合 F X A G U 6 0 2 は F X R F R N 6 0 4 および C C R F R N 6 1 0 から適当な使用可能なデータおよび物理レジスタタグも受け取ることができる。さらに、ロード/ストア命令を実行するために、L S U 6 0 3 は発行段階の間に F P R F R N 6 0 5 および/または F X R F R N 6 0 4 および F S R / C C R F R N 6 0 6 の F S R レジスタの内容から、使用可能なデータならびに F P 物理レジスタタグおよび F X 物理レジスタタグ ( F X \_ D A T A \_ R, F X \_ D V \_ R, F X \_ T A G S \_ R, F P \_ D V \_ R, F P \_ T A G S \_ R ) を受け取ることができる。

#### 【 0 1 1 2 】

そして、リザベーションステーションのエントリが ENTRIES\_AVAIL 信号によって識別されたように使用可能な発行されたそれぞれの命令に対して、そのリザベーションステーションは識別されたエントリに抽出されたオペコード、シリアル番号、チェックポイント番号、物理宛先レジスタタグおよび/またはソースレジスタタグ、およびその命令に対して使用可能なデータを配置する。さらに、受け取ったデータ有効ビットによって識別されたように、データがすでに使用可能であるそれぞれの物理ソースレジスタタグについては、リザベーションステーションはリザベーションステーション内の別のデータ有効ビットをセットして、レジスタタグに関係するデータが使用可能であることを示す。また、命令が即値データを含む場合は、そのデータは即座に使用可能でありまたリザベーションステーションのエントリに記憶されるので、データ有効ビットもセットされる。

#### 【 0 1 1 3 】

### C. 命令の実行と完了

再度図 6 を参照する。発行段階の間に、P C ロジック 1 0 6 はフェッチレジスタ 1 1 0 からフェッチ命令 ( F\_INSTS\_BRP ) を受け取り、それをデコードする。P C ロジック 1 0 6 は I S U 2 0 0 から ISSUE\_VALID 信号も受け取り、実際に発行されたどの命令をデコードしているかを求める。1 つのマシンサイクル内に発行され、実行と完了が予測できるプログラム制御命令については、この P C ロジックはプログラムの流れを予測し、その予測

10

20

30

40

50

に基づいて F P C、A P C、および N A P C の値を計算する。その他のすべての命令については、P C ロジックは現在のマシンサイクルに対する F P C 値に基づいて次のマシンサイクル用の F P C 値と、ISSUE\_VALID 信号が示すように現在のマシンサイクルの間にいくつの命令が発行されるかを計算する。これらの命令についても同様に、P C ロジックは以前のマシンサイクルの A P C 値に基づいて A P C 値および N A P C 値を計算し、また現在のマシンサイクル内に ISSUED\_VALID 信号が示すいくつの命令が発行されるかを計算する。

【 0 1 1 4 】

例えば、SPARC-V9 BPcc, Bicc, BPr, FBfcc および FBPfcc 命令のように、発行された命令に条件付き分岐命令がある場合、P C ロジック 1 0 6 は分岐を取るように予測するかまたは取らないように予測するかを決定するために、これ等の命令に対応する B R P ファイルをデコードする。分岐を取るように予測した場合、P C ロジック 1 0 6 は命令から置換値を抽出し、その値を使用して次のマシンサイクル用の新しい F P C 値、A P C 値、および N A P C 値を計算する。分岐を取らない場合は、F P C 値、A P C 値、および N A P C 値は、プログラムの流れに変更なく標準的に計算される。

10

【 0 1 1 5 】

しかし、発行された命令の中に、SPARC-V9 JMPL[rd=0] 命令のようなリターンタイプのジャンプおよびリンク命令がある場合、P C ロジック 1 0 6 はリターン予測スタック 1 0 5 から予測された J M P L ターゲット P C ( P \_ J M P L \_ P C ) 値をポップオフするために P O P 信号を出力する。P C ロジック 1 0 6 はこのターゲット P C 値を使用して、新しい F P C 値、A P C 値、および N A P C 値形成する。サブルーチンから復帰するために JMPL[r d=0] 命令を使用するが、P C ロジック 1 0 6 は、SPARC-V9 JMPL[rd=15] 命令のような C A L L 命令すなわちコールタイプのジャンプおよびリンク命令が発行されたと判断したときはいつでも、P U S H 信号を出力してリターン予測スタック 1 0 5 に P \_ J M P L \_ P C 信号を送る。リターン予測スタック 1 0 5 に送られたこの P \_ J M P L \_ P C 信号は、8 だけインクリメントする命令用の A P C 値である。さらに、C A L L 命令すなわち JMPL[rd=15] 命令が発行されるときはいつでも、P \_ J M P L \_ P C 信号がリターン予測スタック 1 0 5 に送られるので、P C ロジック 1 0 6 は R E T U R N 命令が発行されるときは常に、P \_ J M P L \_ P C 信号がリターン予測スタック 1 0 5 からポップオフされることも確認しなければならない。

20

【 0 1 1 6 】

さらに、発行された命令には、SPARC-V9 BPA, BA, FBA および FBPA 命令のように無条件常時分岐命令が含まれることがある。この場合、P C ロジック 1 0 6 はこれらの命令から置換値を抽出し、新しい F P C 値、A P C 値、および N A P C 値を計算するために使用する。

30

【 0 1 1 7 】

P C ロジック 1 0 6 は、条件分岐命令、常時分岐命令、および JMPL[rd=0] 命令に対して、D F B 6 2 からの補足データを待つ必要がないので、これ等のタイプの命令は同じ段階で発行、実行、また完了することができる。さらに、発行された条件分岐が実施されるまたは実施されないことが予測され、また予測されたターゲット F P C 値が JMPL[rd=0] 命令について計算されるので、続いてフェッチされた命令は投機的にフェッチされこのため投機的に実行される。

40

【 0 1 1 8 】

さらに、発行された命令の中には、コールタイプの JMPL[rd=15] 命令のような C A L L 命令および R E T U R N 命令、ならびに JMPL[rd=0] 命令以外の J M P L 命令のように他の種類のプログラム制御命令がある。この場合、P C ロジック 1 0 6 は、F P C 値を計算し実行段階の間にその値を D F B 2 B R B バスに乗せるために D F B 6 2 を待つ。図 8 を参照すると、F P C 値は固定小数点データであるので、F X U 6 0 1 は F P C 値を計算する。そして、完了段階の間に、P C ロジック 1 0 6 はこの値を取り込み、新しい F P C 値、A P C 値、および N A P C 値を形成する。

【 0 1 1 9 】

後でより詳細に説明するように、発行されたいくつかの命令が SPARC-V9 DONE 命令また

50



はRETRY 命令のようなトラップハンドリング・ルーチン命令からリターンするとき、PC ロジック 106 はトラップ PC (TPC) 値またはトラップ NPC (TNPC) 値を、図 7 に示す制御レジスタファイル 800 が提供したRD\_TPC\_TNPC 信号から受け取る。図 17 を参照すると、制御レジスタファイル 800 のトラップスタック 815 は、ISU200 がRETRY 命令またはDONE命令が発行されたことを示すRETRY\_DONE\_IS 信号を出力するとき、これ等の信号を提供する。PC ロジック 106 は次にトラップスタック 814 から受け取った TPC 値またはTNPC値を使用して、新しいFPC値、APC値、およびNAPC値を形成する。

#### 【0120】

他の発行された命令については、PC ロジック 106 は発行段階の間に前述のPC値およびNPC 値をインクリメントして、次のフェッチ段階用の新しいPC値およびNPC 値発生する。

10

#### 【0121】

図 8 を参照する。発行段階の間に発行された命令は、必要なすべてのデータが対応するリザベーションステーションのエントリに配置された場合のみ、FPU600、FXU601、FXAGU602、およびLSU603を実行できる。他の命令よりも後で発行された命令用のデータがより早く使用可能になるので、これ等の命令は以前発行されたよりも前に実行され完了されることがある。

#### 【0122】

前述したように、物理ソースレジスタ用のデータ有効ビットが使用可能と指示しない場合は、物理ソースレジスタのソースデータは使用できないことを、リザベーションステーションは知っている。従って、リザベーションステーションは、データが使用可能な場合、FXRFN604、FPRFN605、およびCCRFN610から転送された、適当な使用可能なデータ、データ有効ビット、および物理レジスタタグ(FX\_DATA\_F, FX\_DV\_F, FX\_TAGS\_F, FP\_DATA\_F, FP\_DV\_F, FP\_TAGS\_F, CC\_DATA\_F, CC\_DV\_F, CC\_TAGS\_F)を受け取りモニタする。リザベーションステーションは、対応する転送された物理レジスタタグがリザベーションステーションのエントリに記憶された物理レジスタタグに適合する場合は、特定のレジスタ用に意図された転送データを受け取る。

20

#### 【0123】

すべてのデータ依存性がある命令に適合した場合(すなわち、すべての必要データが使用可能になる場合)、その特定の命令用にリザベーションステーションのエントリに記憶された命令を用いて、実行ユニットは次にその命令を実行する。実行ユニット600~603が実行する命令のタイプについては、リザベーションステーションの記憶エントリに関して前に説明してある。

30

#### 【0124】

生成したデータがその命令用のエントリのリザベーションステーションに記憶された物理宛先レジスタタグによって識別された物理レジスタに記憶されたとき、実行された命令は完了する。完了する間に、実行ユニット600~603はERR\_STAT信号のエラーおよび完了状態情報もPSU300に送る。

#### 【0125】

さらに、図 7 を参照すると、ISU200 はフェッチレジスタ 100 から受け取った命令をデコードして、発行された命令のいずれかが図 17 に示す制御レジスタ 801~814 のリード/ライト命令であるかどうかを決定する。制御レジスタ 801~814 はCPU51の全体的な管理のもとで使用され、SPARC-V9のアーキテクチャマニュアルの中で説明されたタイプの特権付きのステートレジスタ、特権なしのステートレジスタ、および補助ステートレジスタを含むことができる。

40

#### 【0126】

SPARC-V9 RDPR 命令およびRDASR 命令のような発行された制御レジスタのリード命令については、レジスタ 801~814 の 1 つを識別して読み込まれることになっていることを示すRD/WR\_CNTRL 信号をISU200は出力する。その応答として、制御レジスタ・フ

50

ファイル 8 0 0 の制御ロジック 8 1 6 は、RD/WR\_CNTRL 信号が識別した制御レジスタの内容 (RD\_CNTRL\_REG) をウォッチポイントユニット 3 0 4 に出力する。

【 0 1 2 7 】

後述するように、CPU 5 1 は制御レジスタ 8 0 1 ~ 8 1 4 のリード命令に対してチェックポイントおよび対応するウォッチポイントを作るので、ISU 2 0 0 は DO\_WATCHPT 信号および DO\_CHKPT 信号を発生する。DO\_CHKPT 信号は、チェックポイントを作成しこのチェックポイントに対するチェックポイント番号を含むべきであることを示している。DO\_WATCHPT 信号は、DO\_CHKPT 信号が提供したチェックポイント番号にあるウォッチポイントユニット 3 0 4 によって形成されるべきであることを示している。ウォッチポイントユニット 3 0 4 は、それぞれが 1 6 のチェックポイント番号の 1 つに対応する 1 6 のアドレス可能な記憶素子を有する。従って、DO\_WATCHPT 信号および DO\_CHKPT 信号に反応して、ウォッチポイントユニット 3 0 4 は、DO\_CHKPT 信号の中で提供されたチェックポイント番号に対応する記憶素子の中で読み取られる制御レジスタの内容を記憶する。

10

【 0 1 2 8 】

リード制御レジスタ命令は、さらに実行を続けるために、FXU 6 0 0 のリザベーションステーションにも提供される。従って、命令のチェックポイント番号はその命令用のリザベーションステーションのエントリの中に記憶される。そして、FXU 6 0 1 がリード制御レジスタ命令を実行するとき、それは命令のチェックポイント番号をウォッチポイントユニット 3 0 4 に出力する。その応答で、ウォッチポイントユニットは FXU 6 0 0 より前に読み取られた制御レジスタのデータを出力する。FXU 6 0 0 は次にそれを提供し、また固定小数点レジスタに記憶するために対応する物理宛先レジスタタグを FXRFN 6 0 4 に与える。従って、CPU 5 1 は制御レジスタの読み込みを行うために同期される必要はない。

20

【 0 1 2 9 】

制御レジスタ (SPARC-V9 WRPR および WRASR のような) に書き込み動作を行うためには、FXU 6 0 1 はリザベーションステーションにその命令用に記憶された命令データを実行してデータを DFB 2 B R B バスに出力する。図 1 7 に示すように、そのデータは次に制御レジスタ・ファイル 8 0 0 に送られる。ISU 2 0 0 は、制御レジスタへの書き込み命令が発行されまたどのレジスタにこの書き込み命令が行われるかを示す RD/WR\_CNTRL 信号を発生しているので、RD/WR/UPDATE ロジック 8 1 6 はデータを適切に制御レジスタに書き込むことができる。

30

【 0 1 3 0 】

前述したように、典型的な CPU 5 1 は固定小数点用にレジスタウィンドウを使用している。レジスタウィンドウの動作は、SPARC-V9 のアーキテクチャマニュアルに詳細に記載されている。従って、ISU 2 0 0 はいつレジスタウィンドウの制御命令が発行されたかを決定する。そのように行う際に、ISU 2 0 0 はウィンドウ制御命令が発行されたことを示すと共に特定の種類の行うべきレジスタウィンドウ動作を識別する WIN\_CNTRL 信号を発生する。これ等の信号は、図 1 7 に示すように、制御レジスタ・ファイル 8 0 0 に渡される。その応答として、RD/WR/UPDATE ロジック 8 1 6 は次に、WIN\_CNTRL 信号が示すレジスタウィンドウの更新動作を実行する。

40

【 0 1 3 1 】

D . 命令実行段階における PSU の参加

ひとたび命令が DFB 6 2 内の適当な実行ユニットにディスパッチされると、ICRU 3 0 1 は命令が完了するのを待ち、それぞれの命令のシリアル番号についてエラーおよび状態情報が DFB 6 2 および PSU 3 0 0 の間の実行データ転送バスをわたって到来することをモニタする。実行段階における PSU の参加は、実行状態情報を受信することに限定されている。命令がエラーなしで実行を完了し誤った予測がなかった場合、この命令は非活性化される。しかしながら、実行の例外が発生するかまたは投機的な実行が誤って予測された場合、この命令は非活性化されないかまたは CPU 5 1 は回復手順を初期化する。A - リングの状態情報は更新されて、正確または不正確な投機的実行および例外的また

50

は例外なしの実行を反映する。命令が正しく予測され実行が例外なしに予測された場合、ICRU301はシリアル番号に関連するA-リングのビットをクリアしてその命令を非活性化する。

#### 【0132】

典型的なCPU51においては、DFB62は浮動小数点機能ユニット(FPU)600、固定小数点機能ユニット(FXU)601、固定小数点/アドレス発生機能ユニット(FXAGU)602、およびロード/ストア機能ユニット(LSU)603、ならびにFPRFRN605、FXRFRN604、CCRFRN606、およびFSR607を含む。典型的なCPUでは、それぞれのDFB機能ユニットはマシンサイクル当たり2つの命令まで処理できるので、命令の実行完了はマシンサイクル当たり最大8つの命令に(ハードウェアの制約によって)制限される。CPUのある実施例では、FPU600は2つの命令を同時に処理できるが、1つの命令の結果しか出力できない。そのためその実施例では、命令の完了は7つの命令に制限されている。しかしながら、2つの出力を実行できない理由はない。図8はDFB62の中の主な機能ユニットと、DFB62およびISB61間を通過する主な信号の機能的ブロック図を示す。

10

#### 【0133】

ディスパッチされた命令がDFB62の中で実行を完了すると、命令識別エラーおよび他の状態情報(ERR\_STAT)は、実行データ転送バス上のICRU301(および他のユニット)を含むPSU300に同報通信される。データと状態は各命令に対して受け取られるが、順序を乱して受け取られることがある。各命令用の実行状態情報信号には、例えば、実行中にエラーが発生したかどうか、エラーのタイプ、投機的な実行が正しく予測された場合の評価用の条件コード、シリアル番号、命令のチェックポイント(マシンが故障した場合およびバックアップを必要とする場合、マシンがバックアップするチェックポイント)、および命令には制御レジスタに対するリードまたはライト命令が含まれているかどうかの情報を含むことができる。

20

#### 【0134】

同報通信命令のシリアル番号は、図12のアドレスデコード・ロジック334によってA-リング312にインデックスするために使用される。A-リング上のA-リングライトポート341, 342(およびCSN/RRP前進ロジック323用のリードポート343)の数は、プロセッサのピーク実行帯域幅に一致するように選択するのが好ましい。特定の命令がエラーの原因になる場合、A-リングクリアロジック333はこの特定の命令のシリアル番号に相当するA-ビットに「0」を書き込み、この命令は以後非活性化されたと考えられる。しかしながら、命令がエラーを伴って終了した場合、A-リングの対応するアクティブビットはクリアされないため、この命令はなおアクティブと考えられ、例外条件またはエラー条件から回復するために、プロセッサを回復させる別のステップが用いられる。結局、CSNは、エラー処理命令に先行するロケーションのあるスロットを指すISNに追い付くことになる。CSNはそのA-ビットがまだセットされているため、エラー処理命令のシリアル番号を通過できない。そして、実行トラップが行われるとき、エラー処理命令のA-ビットは「1」で上書きされ、CSNおよびRRPが前進することを可能にする。

30

40

#### 【0135】

トラップのタイプを識別するテイクントラップ(TT)フィールドはトラップスタックからポップオフされデコードされて、このトラップが実行トラップ(etrapp)であるかどうかを決定する。この実行トラップは次に「DONE」命令または「RETRY」命令を発行することによって処理される。実行トラップをうまく処理できた場合、トラップハンドラによってETRAPP\_CLEAR信号が現れ、A-ビットをクリアする。同様のロジックがM-リング324のクリアM-ビットに送られる。

#### 【0136】

データ信号および状態信号は、それぞれの実行発行スロットにおける命令のシリアル番号を提供すると共に、どの例外が一時的に早いかを定めることを複数の命令が誤るような

50

場合に、P S U 3 0 0 が例外処理のプライオリティを決定するために使用される。( 例外の優先順位については、プライオリティロジックおよびステートマシン P L S M 3 0 7 に関連してさらに説明する。 ) 典型的なプロセサにおいては、シリアル番号はエラーおよび状態情報よりも 1 サイクル早く到来するため、エラー情報の到来の前にアドレスデコード・ロジックユニット 3 3 4 がシリアル番号をデコードすることが可能であり、このためアドレスデコード・ロジックの中で一層の遅延を招くことなく、ビットをクリアできる場合 ( エラーがない場合 ) またはビットをクリアできない場合 ( エラーがある場合 ) がある。エラーまたは他の例外が検出された場合、この明細書の中で後述するように、このエラーまたは例外を処理するための適切なステップが実施される。

【 0 1 3 7 】

#### E . 命令コミット

これまで説明した構造と方法は、典型的な実施例における A - リング 3 1 2 と 6 4 の命令のアドレス可能なロケーションと同数の命令の発行および実行状態を追跡する構造および方法を提供している。C P U のリソースを回復する構造と方法が欠けていても、C P U 5 1 によって命令に割り当てられた、例えばシリアル番号および関連する A - ビットを含むリソースは、例え命令が完了し非活性化されていても、その命令に対して割り当てられたままであろう。A - リングのデータ構造に関連する付加的なポインタを付け加えて、また C P U 5 1 内の関連する命令の状態の変化に応じてそのポインタを移動させることによって、特に非活性化と命令をコミットすることができ、またリソースをリタイアさせて別の命令によって後で使用するために再生できるようになる。この方法では、リソースは新たに発行された命令が連続的に使用できるようになる。

【 0 1 3 8 】

コミットシリアル番号ポインタ ( C S N ) 3 1 6 は、最新のコミットされた命令を指し示す。「コミットされた命令」とは、エラーなしで実行が完了した命令と定義されるため、それは発行から実行を通して進行し、取り消される必要があるまた取り消されることがないかなる投機的に発行された予測中央転送 ( ブランチ ) 命令よりも前に発行されている。コミットされた命令を取り消す必要はない。定義により、コミット命令シリアル番号 ( C S N ) ポインタ 3 1 6 より前のすべての命令は、エラーなしで実行が完了しており、取り消される必要はない。C S N がマシンサイクルごとに進む量は、マシンの最大命令コミット速度を設定する。完了しておりかつ非活性化された命令は、A - リングにおけるこれ等の命令用のシリアル番号に向かってまたは超えて C S N が前進するまでコミットされるとは考えられない。

【 0 1 3 9 】

I S N 3 1 4 について前述したように、C S N 3 1 6 が前進できる最大のポジション数は、ソフトウェア、ハードウェア、または他の制御によってサイクル当たりの命令のより小さな最大数に制限できる。例えば、I S N の前進をマシンサイクル当たり 4 つの命令に制限する発明のある実施例では、C S N はハードウェアに課せられた制限によってサイクル当たり 8 つに制限される。さらに、C S N は I S N に等しくなることもあるが、A - リングの周りでは I S N を超えて前進することはできない。( A - リングの構造のため、C S N は厳密には数的に I S N より小さく、等しく、または大きくなることができる。 ) C P U が初期化される時、C S N は 0 ( ゼロ ) に初期化され、次に命令の状態の変化および、以後説明するように、所定の C S N 前進ルールに従って前進する。

【 0 1 4 0 】

アクティブビットはアクティブ命令リングの中でクリアされるので、C S N は所定のルールに従って前進して、実行が完了している命令を「コミット」する。一般に、C S N のポインタは、すでに実行を完了した命令より遅れることがある ( すなわち、非活性化される ) 。C S N / R R P 前進ロジックユニット 3 2 3 は、C S N ポインタおよび R R P ポインタのポジションを前進させることに責任を負う。各マシンサイクルで、C S N / R R P ロジックユニット 3 2 3 は I S N を通らずに、完了した非活性化された命令 ( A - リングにおいて、アクティビティビット = 「 0 」 ) を超えて C S N を前進させようとする。据置

10

20

30

40

50

きトラップが生じた場合を除いて、CSNはA - リング312のA - ビットのセットされた条件またはクリアされた条件および実行された（例えば、ソフトウェア、ファームウェア、および/またはハードウェア）所定のルールに基づく場合のみ前進する。CSN/R RPロジックユニットはCSN用の適切なシリアル番号のアドレスを決定するために、A - リング自体の状態を問い合わせ、ISNを注意し続ける必要があるだけである。据置きトラップが得られてうまく処理された場合、PLSM307はDEFERRED\_TRAP信号を、CSNを1つのシリアル番号のロケーションだけ前進させるCSN/R RP前進ロジック制御・ユニット323に送り、このためCSNと後のRRPが例外の原因となる命令を通過して前進できるようになる。本発明のある実施例では、CSNの前進は所定の最大命令コミット速度（マシンサイクル当たり8命令）によって制限される。この速度はCSNポインタを移動させるために必要なハードウェアを制限する希望に基づいて選択される。

10

#### 【0141】

原則として、CSNはISNと等しくなるまで前進できる。ISN = CSNの場合、CPUは空である、すなわちすべての発行された命令はコミットされている。実際、ロジックを単純化できまた必要なハードウェアを少なくできるので、CSNの最大の前進量を（マシンサイクル当たり（4命令発行マシンについては）8つに）制限することは有利である。例えば、マシンサイクル当たり4命令発行CPUでは、CSNの前進量はサイクル当たり8つのシリアル番号に制限されている。CSNの前進量を支配するルールは、以下のように要約できる。（1）CSNの前進はISNまでである（両端を含む）。（2）CSNの前進は1マシンサイクル当たり8命令のシリアル番号以下である。（3）前進はまだコミットできないアクティブ命令に相当するA - リングの最初の「1」までである（両端を除く）。さらに厳密にいうと、CSNは一般に次の関係に基づいて前進する（下記の例外に関する修正を参照のこと）。すなわち、 $CSN = \min(CSN+4, ISN, slot\_A)$ 、ここで、 $A(CSN+1), \dots, A(slot\_A)$  はすべて「0」であり、 $A(slot\_A+1) = 1$  である。この式で「min」は最小関数であり、「slot\_A」はA - リング312のA - ビットのロケーションのことであり、また $A(slot\_A)$ はそのA - ビットロケーションのビットの状態（例えば、「1」または「0」）である。据置きトラップが削除されまた未完成の浮動小数点オペレーションの例外またはデータブレークポイントの例外が検出されたように受け取られたとき、CSNを前進させるルールは修正される。未完成の浮動小数点の例外およびデータブレークポイントの例外は、DFB62からの状態信号（ERR\_STAT）を実行することによって識別される。これ等の例外が発生しトラップが処理されると、CSNは1だけ前進する、すなわち $CSN = CSN + 1$ となり、CSNは例外の原因となった命令を通過して前進する。PSU300は未完成の浮動小数点オペレーションおよびデータブレークポイントに特に関心を持っており、A - ビットが処理されたトラップに対してクリアされない場合、CPUは満杯となり機能停止するため、未完成の浮動小数点オペレーションおよびデータブレークポイントの両方はポインタの特別の処理を必要とすることがある。

20

30

#### 【0142】

CSNの前進を「 $\max(8, ISN, \text{the next A-bit} = 1)$ 」に制限することによって、CSNを適切なポジションに移動させるための単純なハードウェアの回路を実現できる。例えば、A - リング312は8つの8ビットレジスタとして実現できる。これ等8つの8ビットレジスタのA - リングのリードポート343において、CSNの周囲のレジスタが読み込まれ、これ等の読み込まれたレジスタの内容は2つの別個の8ビットバスに送られる。これ等のバスを単一の16ビットバスに連結して、次に0ビットから8ビットまでシフトすることによって、古いCSNを前進させるための8ビット関心ウィンドウを形成する（CSNの最大の前進量は8）。「最初の1を見つける」回路は、CSNが前進する量を見つけることになる。

40

#### 【0143】

ICRU301はまたコミットシリアル番号（CSN）信号および発行転送番号（ISN）をチェックポイントユニット303に送ることによって、現在のコミットシリアル番号（CSN）316のチェックポイントユニット303に通知し、CSNより小さいシリ

50

アル番号を有する任意の割り当てられたチェックポイントを解放できるようにする。厳密な数学的な順序（すなわち、シリアル番号）ではなく円形の A - リング上のロケーションに比例するもので、CSN は円形の A - リング上のロケーションに比例する。チェックポイントについては、この明細書の各所でより詳細に説明している。CPU 51 のある実施例では、本願に記載した CSN ならびに他のポイントは、アドレス可能な A - リングのビットロケーションと一致する命令のシリアル番号をエンコードする 6 ビットのベクタである。

【0144】

#### F. 命令のリタイアメントとリソースの回復

CPU の割り当て可能なリソースはプロバイダによって回復される。補助ポイントは、命令がリタイアするときに解放されるリソースに対して命令のリタイアメントおよびリソースの再生を追跡および制御するために提供される。リソース再生ポイント (RRP) 317 は、最新のリタイアした命令を指し示し、CSN を追跡する。「リタイアした命令」とは、実行が完了し、CPU 51 によって非活性化され、またコミットされた命令のことであり、次の命令の発行の間に、次の使用のために再生された関連するマシンのすべてのリソースを有している。再生されたマシンのリソースには、リネームされたおよび/または再マッピングされた論理レジスタ、アーキテクチャレジスタ、および物理レジスタが含まれる。これ等のレジスタは、命令のリタイア時に開放されこのため次の命令が発行される間に再割り当てに使用可能とされた命令、命令のシリアル番号 (SN)、チェックポイント、およびウォッチポイントに割り当てられている。マシンサイクルごとに RRP が前進する量は、そのマシンの最大命令リタイアメント速度を設定する。CPU が初期化されると、RRP は 0 (ゼロ) に初期化され、命令の状態の変化および後述する所定の RRP の前進ルールに従って前進する。チェックポイントとウォッチポイントの割り当ておよび割り当て解除については、以下に詳細に説明する。

【0145】

命令がコミットされると、リソース再生ポイント (RRP) 317 は、命令をリタイアするために CSN 316 の後ろに前進するため、マシンのリソースを解放することができまた再利用のために再生できる。それぞれのマシンサイクルごとに、A - リングの A - ビットの状態と CSN を超えない所定の RRP の前進ルールとに基づいて、RRP 317 はコミットされた命令を超えて前進しようとする。非コミット命令に割り当てられたマシンのリソースは、たとえ非活性化された場合でも、命令が取り消される必要がないことがはっきりするまで開放されないため、RRP は CSN を通過できない。非活性化された命令はまだ投機的であるので、命令の完了 (非活性化) だけでは命令コミットまたはリタイアメントに対して十分ではない。コミットされた命令のみがリタイアできる。RRP 317 はレジスタのリソースを再生できる次の命令を指し示している。A - リング 312、CSN 316、および CSN / RRP 前進ロジック 328 の中で実行できる所定のルールのセットされた条件またはクリアされた条件のみに基づいて、RRP 317 は前進する。

【0146】

ISN および CSN の場合のように、RRP がそれぞれのマシンサイクルごとに前進できるポジションの最大数は、他のソフトウェア、ハードウェア、または他の制御によって、サイクル当たりの命令のより小さな最大数に制限される。本発明のある実施例では、RRP の前進量は、RRP のポイントを移動させるに必要なハードウェアを制限したいという希望に基づいて選択された所定の最大命令リタイアメント速度によって制限される。例えば、典型的なサイクル当たり 4 発行タイプの CPU の実施例では、RRP の前進とこれによるリソースの回復は、マシンサイクル当たり最大 4 命令に制限される。これは典型的なマシンの最大命令発行速度に等しいため、CPU 51 の性能を制限するものではない。RRP を通常の発行速度よりも遅く前進するように制限すべきではない。さらに厳密に言うと、RRP は次の関係、すなわち、「 $RRP + \min(RRP + 4, CSN)$ 」に基づいて前進する。原則的に、この方法でリソースの回復を制限することは、本質的なものではなく、RRP は CSN まで前進できる。ハードウェア設計のトレードオフのために、典型的な CPU 51

10

20

30

40

50

に制限が課せられた。CSNを所定の命令数だけ前進させるに必要なハードウェアは、RRPを同じ命令数だけ進めるに必要なハードウェアよりも大規模になる。従って、ハードウェアを希望のレベルまで小さくするために、性能に有害な影響がほとんどまたは全くないようにして、RRPをCSNよりも遅く前進させる。

#### 【0147】

##### 1. 命令のコミットと命令のリタイアメント時のRRFとリネームマップの更新

図18を参照する。リソースリカバリは命令のシリアル番号を開放するので、そのシリアル番号を再び使用することができ、その結果、いくつかのレジスタを開放することになり、そのレジスタはフリーリストユニット700内のフリーリストに戻すことができる。命令がコミットされると、この命令はICRU301が発生したCOMMIT\_VALID信号によって識別される。このCOMMIT\_VALID信号に応答して、RRF302の記憶ユニット366のコミットされた命令と一致するエントリは、再利用のため制御ロジック365によって開放される。

10

#### 【0148】

##### 2. 命令のリタイアメント時のRRFの読取り

命令がリタイアするとき(ICRU301の説明を参照のこと)およびマシンのバックアップ時(バックトラックの説明を参照のこと)に、RRFは読み取られる。ある命令がリタイアする場合、RRPポインタが示すように、制御ロジック365は記憶ユニット366を制御して、フリーリストユニット700に、FREE\_TAG信号の形式で、リタイアした命令に一致する物理レジスタタグを出力する。フリーリストユニットは次に、この物理レジスタタグを空きの物理レジスタタグのリストの中に入れる。

20

#### 【0149】

##### 3. ポインタを使用する精確なステートメンテナンス

A-リング312およびその関連するロジックは、発行された各命令の状態に関する正確なマシンサイクルの現在の情報を提供する。ISNとCSNのポインタは、発行されたアクティブな命令の状態を追跡する。CSNが特定の命令に向かって前進する場合のみ、CPU51は、その命令をコミットできまたフリーなリソースをその命令の実行と関連付けることができると確信することができ、これにより精確な状態を保存できる。ISNはエントリを通過するとき、発行された命令にA-ビット(A=1)をセットするA-リングの周りを移動する。CSNは、コミットするために完了した命令(A=0)を捜すISNに続く。NMCSNは、完了した(非活性化された)メモリ命令(M=0)を捜すISNに続く。RRPはA-リング312の周りのCSNの後に従い、コミットされた命令のリソースを再生する。積極的なロード/ストア命令のスケジューリングを説明する場合、その説明にはどのようにNMCSNが完了した(非活性化された)メモリ命令(M=0)を捜すISNに続くかが記載されることになる。

30

#### 【0150】

CSNはA-ビットがセットされている命令へは進まない。(NMCSNは、M-ビットがメモリ参照命令の発行時にクリアされるので、メモリ参照命令を通過して前進する。)正確な状態を維持できるポインタを移動させるための他のルールは、以下の通りである。すなわち、(1)CSNはISNを通過して前進できない(ISN CSN)。(2)CSNはNMCSNを通過して前進できない(NMCSN CSN)。(3)CSNはPBSNを通過して前進できない(PBSN CSN)。(4)NMCSNはISNへ前進できない(ISN NMCSN)。(5)PBSNはISNへ前進できない(ISN PBSN)。(6)RRPはCSNを通過して前進できない(CSN RRP)。(7)NISNは常にISN+1である(加算はモジュロリング長である)。(8)ISNはRRPまたはCSNに追い付くことはできない(最大限のマシン条件を示す)。好ましい実行方法において任意に実行できる他のルールとして、(9)CSNおよびNMCSNは1クロックサイクルで最大8前進できる。(10)RRPはマシンの1クロックサイクルで最大4前進できる。記号「」は、より大きいまたは等しい関係は、厳密に数学的に等しいまたは等しくないということではなく、A-リング構造の周りのラップに関して決定されることを示す。

40

50

【0151】

ISNポインタ、CSNポインタ、およびRRPポインタに関して、次のように観察できる。すなわち、第1に、CSNとISNとの間のシリアル番号を持つ命令は、マシン内で未決着のアクティブな命令に相当する。これ等の命令は投機的であり、取り消す必要がある。第2に、CSNとRRPとの間の命令はすでにコミットされたリタイアしていない命令に相当するもので、RRPによるリタイアメントを単に待っている。これ等のコミットされた命令は取り消されることはない。第3に、CSNはまだ発行されていない命令の結果をコミットできないので、ISNを通り越すことはできない。第4に、定義によりCSN = ISN = RRPの場合に「シンク」信号が到着するので、マシンで発行されたすべての (ISN) 命令はコミットされ (CSN) リタイアされる (RRP)。第5に、ISNは概念的に円形のA - リングにおいてRRPを超えることはできず、またRRPとISNとの間のシリアル番号は命令の発行に使用できる。ISN = RRP - 1の場合 (モジュロA - リング長)、すべての命令のシリアル番号は割り当てられ、そしてRRPが前進して別の命令が発行される前にマシンのリソースを再利用するために再生できるまで、マシンはストール (待機) しなければならない。無論、CPU51はストールの前に発行された命令の実行、非活性化、完了、およびコミットを継続する。最後に、3つのポインタ間の相対的な関係は、RRP CSN ISNであり、ここで「」のオペレータは円形のA - リング (モジュロA - リング長) 内の順序付けを示すものであり、厳密な数学的な関係を意味するものではない。図20は命令の発行、非活性化、コミット、およびリタイアメントを含む精確な状態を維持するための、本発明による方法の実施例の概略フローチャートである。

10

20

【0152】

III . 積極的な長レイテンシ (長待ち時間) (ロード / ストア) 命令のスケジューリング

命令のコミットとリタイアメントを追跡し予定する各種のポインタを使用する本発明による方法の説明は、すべての命令のタイプに関係するものであり、現在の投機的な順序なし (out-of-order) のプロセサでは多くの利点を提供する。多くの現在のプロセサは、ロード命令およびストア命令のみが外部メモリを参照し、一方他のすべての命令またはオペレーションはプロセサ内のレジスタを参照するような「ロード - ストア」アーキテクチャとして設計されてきたことを認識することによって、さらに別の利点を得ることができる。

30

【0153】

実際に、ロード命令とストア命令は、この明細書ではレイテンシ (待ち時間) の長い命令と呼ぶより一般的なクラスの命令に属しており、このレイテンシの長い命令は完了するには数個のマシンサイクルを必要とし、ロード命令とストア命令が2つのタイプである外部メモリ参照命令を含む。内部レジスタを参照する命令は、一般に完了するためにはより少ないまたは単に1つのマシンサイクルしか必要としない。レイテンシの長い命令も、割り込みなしに複数のマシンサイクルで実行するように設計された命令である「原子」命令を含んでいる。

40

【0154】

レイテンシとは、命令が発行されてからその命令の実行が完了するまでの間の経過時間のことである。ロード - ストアアーキテクチャに対する命令のセットは好ましい、というのはそれがレイテンシの長いメモリ参照 (メモリへのロード - ストア参照) を、CPU51の内部レジスタを参照するレイテンシの短い論理演算または算術演算 (加算、乗算、除算、論理比較、など) からデカップリングするためである。さらに、ロード - ストアタイプのアーキテクチャで設計された最も新しいCPU51は、このタイプの命令セットから発生した増大するレジスタへの要求事項を支援できる多数の汎用目的のレジスタを有している。

【0155】

ロード命令とストア命令は両方ともアーキテクチャの状態を修正する。順序なしに実行

50



するプロセサでは、精確な状態を危うくすることなく高いロード/ストア帯域幅を維持するために、ロード命令とストア命令とを効果的に予定できることが重要である。精確な状態を維持しながらレイテンシの長い命令を積極的に予定する、命令の状態を追跡するための基本的な構造と方法への改良が提供される。P S U 3 0 0 の I C R U 3 0 1 内の構造と方法は、性能を改良するために後で説明するが、D F B 6 2 の、特にロード/ストアユニット(L S U) 6 0 3 内の構造と方法に協力する。

【0156】

#### A. メモリ命令状態情報記憶

本発明による構造と方法は、長レイテンシ(長待ち時間)命令を区別する他の信号と、長レイテンシ命令状態情報を記憶する記憶領域と、命令状態を追跡するポインタを供給する。第21図は、メモリ参照命令(例えばロード/ストア命令)に代表される長レイテンシ命令状態情報を記憶するメモリ命令リング(Mリング)324と前述の能動命令リング311との関係を示す概念図である。Mリング324は一般にAリング311と同じ構造と全体特性を持っているが、Mリング状態ビットはAリング状態ビットとは異なる組の規則に従ってセットされる。特にMリング324は、命令と命令形式が発行時に能動(ビットが“1”にセットされる)または非能動(ビットが“0”にクリアされる)として処理される規則を設定する。

【0157】

第10図に示すようにISU200が命令を発行する時には、前述のISSUE\_NOP及びISSUE\_VALIDと共にメモリ参照命令と非メモリ参照命令とを区別する命令形式信号を供給する。ロード命令、ストア命令、原子命令順序及び他のメモリ参照命令を含め、積極的にスケジューラされることによって利益を生む特定クラスの長レイテンシ命令を示すようにISSUE\_MEMを発生させることもできる。

【0158】

CPU51の一種の実施例では、ISSUE\_MEMはビットでのアサーション(“1”)がi番目の命令がメモリ関連であることを示す4ビットベクトル(各ビットは命令発行ウィンドウ中で発行可能な命令の1個に対応)となっている。ISSUE\_MEMは、第12図に示すようにMリング324にビットをセットするためにMリングセットロジック7335によって用いられる。Aリング312内にはアドレスデコードロジック338, 336が設けられている。ISSUE\_MEMは発行された命令が記憶動作に関係するか否かを示す。ISU200からのISSUE\_MEMがビットiでアサートされて同じ命令スロットに対応するISSUE\_VALIDがアサートされると、メモリ命令リング(Mリング)324はメモリ命令リング(Mリング)324にはセットされない。命令がメモリに関連している時にMビットをセットしないことにより、CSNがクリアされたAリングビットへ進むように非メモリコミテッド命令シリアル番号(NMCSN)ポインタ(後述)がクリアされたMビットへ進み、非メモリ参照命令によって進むことを防止されるように、発行時にメモリ関連命令を有効に「非活性化」することができる。ISSUE\_VALIDが“1”の時、即ちメモリ関連であろうとなかろうと発行された各命令に対しては、Mビットは必ず“0”または“1”として書かれなければならない。記憶動作のMビットをセットしないことによりNMCSNを進めるために記憶関連動作を「無視」することが容易になる。

【0159】

命令がエラーなく完了した時には、Aリング用にクリアされた時と同じようにMビットはMリングクリアロジック337によってクリアされる。発行時にクリアされたMビットを持っているメモリ参照命令は、命令の非能動化後もクリアされたままである。Aリング312と同じように、Mリングビットは実行トラップのためにクリアされる。Mリング324状態情報はメモリ関連命令を追跡し積極的にスケジューラするのに用いられる。メモリ参照命令の発行状態、コミテッド状態、リタイヤ状態を含むすべての状態は、Aリング311とそれに関連するポインタによって維持される。

【0160】

AリングとMリングの前述の類似性からして、当業者はAリングとMリングの機能はリ

10

20

30

40

50

ングのアドレス可能な場所が複数のビットを含んでいる単一の円形リングのような単一のデータ構造内で発効できることにお気づきであろう。このような複数ビットのリングでは、命令活性（能動または非能動）と命令形式（メモリ、分岐、NOP(ノップ)など)の両方共複数ビットにコード化できる。更に、現在別のAリングとMリングのデータ構造に供給されているのと同じ情報を記憶するために、他のコード化スケジュールが提供できよう。以上の説明からして、当業者は、マルチビット書式サポートしNMCSNを収容するためのリングセットとクリヤロジックの変更の仕方、マルチビット円形リング用のCSN/RRPロジックの変更の仕方、能動と非能動との差異化とメモリ命令の差異化のためにセットには“1”クリヤには“0”とする以外に異なるロジック状態または記号を使用できることを御理解されることであろう。

10

【0161】

#### B. 長レイテンシ命令を追跡しスケジュールするNMCSN ポインタとPBSNポインタ

Mリング324にMビットをセットする構造と方法については以上に説明した。本発明の構造と方法では、長レイテンシ命令、特に外部メモリを参照するロード/ストア命令をCPU51が有効且つ積極的にスケジュールすることができるようにするため、Mリング324に関連して予測分岐命令シリアル番号(PBSN)318とポインタと非メモリコミットドシリアル番号(NMCSN)ポインタ319が供給使用される。NMCSNとPBSNは前述のように両方共ICRU301内のポインタレジスタ313に記憶される。

【0162】

NMCSN319は最後の「コミットド」命令を支持する(前述)が、命令発行時にメモリ参照命令用のMビットがクリヤされるので、NMCSNを進めるためにすべてのメモリアクセス命令は有効に無視され、従ってNMCSNは更に高速に進む。ポインタCSN316が進むだけで命令が実際にコミットされる点に御注目頂きたい。後述するようにNMCSNはロード/ストアユニット603によってだけ使用される。命令がエラーなく完了し、それ以前のすべての命令がエラーなく完了している場合に命令をコミットすることができる。メモリ参照命令が未だ能動であってもNMCSNはメモリ参照命令を追い越して進むことができる。この方法はメモリを参照しない他の長レイテンシ命令形式を積極的にスケジュールするのにも使用できる。CPUが初期設定されるとNMCSNは0(ゼロ)に初期設定される。

20

【0163】

PBSN318は一番初期の(一番古い)未解決予測分岐命令のシリアル番号である。この実施例ではPBSNはウォッチポイントユニット304によって決定される。(ウォッチポイントユニット304については本明細書の他の箇所に詳述。)未解決分岐命令とは、発行され実行されたことが確認される以前に理論上発行され、実行開始され実行完了されている可能性のある命令(及びその命令から生じる命令順序)のことを言う。換言すれば、命令が発行された時には処理の流れの中にその命令を実行する条件が未だ生じてはいないのである。本発明の構造及び方法では、未解決予測分岐命令が存在しない時にはPBSNは必ずISNと等しくなる。PBSNはNMCSN進行ロジックでNMCSN進行のパリヤとして用いられる。NMCSNはポインタPBSNまたはISNを追い越して進むことはできず、メモリ命令が存在しない情況では、ポインタNMCSNはCSNに等しい。CPUが初期設定されるとPBSNは0(ゼロ)に初期設定される。

30

40

【0164】

#### C. NMCSNの進行

NMCSN319はCSN316と対になってメモリ関連動作を行う。CSNを進めるAリングに対応してNMCSNを進めるMリングがある。Mリングはラップの周囲のMビット64個から成る。即ち、典型的には64ビットのMリングで、Mビット#63に後続するビットはMビット#0である。Mビット=0とは、メモリ関連命令、例えばロード/ストア命令のような長レイテンシ命令のことである。

【0165】

例示したCPUでは、NMCSNポインタを進める規則は次の示すようにCSNを進める規則に類似している。(1)NMCSNをISNまで進める(ISNを除く)。(2)NMCSNを1サイクル

50

で 8 個以下の命令シリアル番号以下進める。(3) Mリング内の最初の“1”まで進める(この“1”を除く)。

【0166】

浮動小数点ユニット(FPU)で「未完浮動小数点動作」例外が生ずるかまたはロード/ストアユニット(LSU)から「データブレイクポイント例外」が検出された時にはNMCSNを進める規則が変更される。これら2種類の場合には、前述したCSNと同様にNMCSNはシリアル番号1個だけ進められる。即ち、これらの状態ではCSN CSN + 1とNMCSN NMCSN + 1になる。PSU300は特別のポインタ操作を必要とする未完浮動小数点動作とデータブレイクポイント動作に特に関係している。

【0167】

MリングのインプリメンテーションはAリングのインプリメンテーションに似ている。例示したCPUではNMCSNは事実上maximum(8, ISN、次のMビット=1)までしか進めないで、CSNと同様にNMCSNを適当な位置へ進めるには簡単なハードウェア回路構成の設置計画をすればよい。(maximum(x, y, z)という表現は、パラメータ値またはx, y及びzという表現の中から最大値を選択するという意味である。)Mリング324は8個の8ビットレジスタとして設置される。これら8個の8ビットレジスタのMリング読取り口346では包囲しているNMCSNのレジスタが読取られ、読取り時のこれらのレジスタの内容が2本の別々の8ビットバスに送られる。これらのバスを連結して1本の16ビットバスとし、次にこのバスを0ビットから8ビットへ移動させて古いNMCSNを進める問題の8ビットウィンドウを作る。「最初の1を発見する」回路がNMCSNの進む量を与える。

【0168】

NMCSNとPBSNの保守を正しく行えば、精確な状態を曖昧化しない実行のために選択し得るロード/ストア命令またはその他のメモリ参照命令を簡単に決定することができる。精確な状態を曖昧化しない実行のために選択し得るロード/ストア命令は、実行に「インレンジ」だと考えられる。(1)例外を発生する命令がアーキテクチャ状態を変えず、(2)故障命令以前のすべての命令がアーキテクチャ状態への変更を完了し、(3)故障命令以後のすべての命令がアーキテクチャ状態を変えていない時に精確な状態が維持されることをもう一度述べておく。LSU603はまた、長レイテンシ命令のスケジュールに使用するため、ICPU301からのNMCSN319とウォッチポイントユニット304からのPBSN318を受信する。

【0169】

D. データフローブロック内のロード/ストアユニット(LSU)

本発明の特徴のいくつかの設定は一般的な実行ユニットによって得られるものであるが、長レイテンシ命令の積極的スケジュールが利益を生み出すのは状態スケジュールに長レイテンシをサポートする特定の構造が存在する場合である。例えば、ロード/ストア命令と、他のメモリ参照命令を積極的にスケジュールする構造が提供される。局所的な記憶動作に参加しない他の実行ユニットにこの種のテストを設置する方法は、以上の開示からして当業者には自明であろう。

【0170】

LSUは、SPARC-V9アーキテクチャマニュアルに定義されているリラックスメモリモデル(RMO)とトータルストアオーダリング(TSO)の両モードをサポートする。LSUは固定小数点と浮動小数点両方のロード/ストア命令に対して責任を持つ。LSUは各々がキャッシュへのサイクルである2個までのリクエストを作ることができる。精確な状態を維持するために必要な命令の順序立ては、プロセッサとキャッシュチップとの間で1組のプロトコル信号によって行われる。ISNは12項目のリザーブステーションを内蔵している。LSUとデータキャッシュの間には分割されたトランザクションをサポートする3段階のパイプラインがある。第1段階では命令のアドレスと操作符号とシリアル番号がデータキャッシュへ送られる。理論上の実行に使用されるいくつかの制御ビットも送られる。第2段階ではISUからデータキャッシュへとストア命令のデータが送られる。第2段階ではまた、データキャッシュが次のサイクルで完了する命令のシリアル番号と有効ビットをLSUへ返送す

10

20

30

40

50

る。第3段階ではデータキャッシュが状態とロードデータを返送する。キャッシュミスの場合は、データキャッシュはパイプラインスロットが使用されていない間にデータを返送するか、そのデータのためにパイプラインスロットを開く信号をアサートする。

#### 【0171】

第22図は選択兼実行制御ロジックユニット609と、種々の命令及びデータ関連情報を記憶する命令/データ待ち行列(IDQ)610と、ISU200, FPU600, FXU601, FXAGU602からの信号を復号する復号&フィールド分離ロジック(SSFSL)/620を含むロード/ストアユニット(LSU)の機能ブロック図である。詳述すると、IDQ610はインレンジビットフィールド611と、操作符号フィールド612と、命令シリアル番号(SN)フィールド613と、チェックポイント番号614と、有効性ビットフィールド615と、アトリビュート(ATR)フィールド616と、レジスタタグ(TAGS)フィールド617と、タグ有効フィールド(TV)618と、アドレスフィールド619と、アドレス(AV)フィールド620と、キャッシュデータフィールド621を含んでいる。データキャッシュ52はIDQ610とアドレス及びデータを送受信する。

10

#### 【0172】

LSU603が命令を受信すると、デコード・フィールド分離ロジック622は命令の直接データフィールドにアドレスを含めるか或いはFXAGU602がアドレスを計算しなければならないかを決定する。デコードロジック622がアドレスはFXAGU602によって計算されなければならないと決めると、アドレス有効ビットをクリアする。しかし、命令の内部に直接データがあるという意味でアドレスが命令の内部に含まれると決めると、アドレス有効ビットをセ

20

#### 【0173】

ットして、その命令用のアドレスデータが準備されて利用でき、その命令を実行することができることを指示する。タグ信号がデータが有効か否かを指示するように、FPUとFXUから来るタグも同様に操作される。命令がリザーブステーションまたは待ち行列610に記憶されると、発行された有効信号が実際に実行されたことを示すまで有効信号がクリアされる(アサートされない)。

30

#### 【0174】

LSU603が命令を実行すると、データキャッシュ52へ直接データを送る。LSU603がデータキャッシュ52からのデータを検索している時には、本発明の設置計画のデータは選択兼実行制御ロジックに入っており、このロジックはタグセンドを含む信号とデータが有効であることを示すデータ有効ビットを送出し、また図示したようにデータを送出するので、組合わせ信号は出て来ない。第22図にはFP\_TAGS\_DV\_F信号とFP\_DATA\_F信号が示されている。

#### 【0175】

選択兼実行制御ロジックユニット(SECLU)609はLSUで実行する命令の選択に責任がある。このユニットはロード、ストア、アトミック動作などの順序立て強制を取り扱う。例えば、アドレス「X」への若いロードはアドレス「X」への古いストアを追い越すことはできない。この型のロードはストアのようなアトリビュートアレイでは「強い」とマーク付けされる。このユニットはストアの「順序立て」を開始し、すべての先行するロードとストアが完了するまでストアを開始することはできない。最後に、ストアのシリアル番号はインレンジでなければならない。SECLU609は、有効適格(即ちキャッシュによって現在処理されていない)であってすべての従属性を満し、且つ「弱い」動作(例えばロード)が「インレンジ」であってしかも前のキャッシュミスによって強制されていない命令だけをマスクするロジックを含んでいる。マスクされた命令は次に、マスクロジックロジック準備を満す命令の組から一番古い命令を選択するプリシードンスマトリックス623によって処理される。

40

#### 【0176】

50

SECLU609もメモリアクセスの制御に責任がある。このユニットはキャッシュにコマンドを送り、アクセス完了時にキャッシュ状態情報を受信する。また、キャッシュからの「ロード」データに適当な目的地レジスタタグを貼り、アクセスのアトリビュートのトラックをアトリビュートアレイ616に保存する。これらのアトリビュートはインレンジで、強/弱な(一般にストアとアトミックは「強く」、ロードは「弱い」)理論上のアクセスで、キャッシュへ送るに適格なものを含んでいる。

#### 【0177】

SSFSL620は命令操作符号(オペコード)と、命令シリアル番号と命令チェックポイントとレジスタタグを含む命令パケットをISU200から受信し、情報をデコードし、情報を記憶するためにIDQ610で正しいスロットを選択し、情報を適当なフィールドに分離する。SNに関連する各シリアル番号項目とデータは待ち行列中の空のスロットに記憶される。

10

#### 【0178】

SECLUはポインタ値(ISN, NISN, PBSN及びNMCSN)もPSU300から受信する。CSN, PBSN及びNMCSNは、命令シリアル番号がインレンジであるかの決定に特に関係する。SECLU609はアクセスが「インレンジ」であるか、即ち、シリアル番号が前述のようにCSNより大きくNMCSN以下であるかを定めるロジックを含んでいる。インレンジ情報はアトリビュートアレイ616へ進められてSECLU内のインレンジファイル611に記憶される。SECLUはIDQに記憶されている項目をデータ返送状態及び/または与えられたアクセス用のキャッシュからのデータとマッチングするロジックも含んでいる。IDQ内のすべての有効項目はそのサイクル中インレンジであるかを見るためにモニタされ、アトリビュートアレイは必要に応じて更新される。待ち行列項目が一旦インレンジになると、再使用されるまではインレンジのままである。

20

#### 【0179】

チェックポイントフィールド614は、待ち行列スライス中のアクセスに対応するチェックポイント番号を含んでいる。各サイクル毎に新しいチェックポイント番号がISB61から到着する。有効ビット待ち行列615は、メモリアクセス命令が実際に発行された場合にセットされる有効性ビットを保持する。この情報は命令発行サイクル中にISB61から来る。ISBによってアクセスが殺された場合にはビットをリセットすることができる。アドレスフィールド618はすべてのキャッシュアクセス用のアドレスを保持する。DFB62アドレス発生ユニット(FXAGU)602からアドレス待ち行列にアドレスが到着する前に、アドレス番号がIDQアドレスマッチロジックに到着する。制御レジスタの内容に応じてアトリビュートアレイ616にアトリビュートをセットしてもよい。例えばプログラムオーダビットをセットすると、アトリビュートセクションには「強い」ビットがセットされる。

30

#### 【0180】

##### 1. インレンジメモリ参照(長レイテンシ)命令の識別

前述のように、SECLU609はICRU301からのCSNとNMCSN及びウォッチポイントユニット304からのPBSNに応じて命令が「インレンジ」であるかを定めるロジックを含んでいる。このロジックはCSNとPBSN(CSN第1窓PBSN)の間のシリアル番号を持つ第1窓にある命令が分岐の誤予測に関係なく実行できるか識別する(但し、「」の記号は厳密な算術順序よりもメモリ命令リング内で相対順序を示す)。第2のCSNとNMCSN(CSNインレンジNMCSNPBSN)の間のシリアル番号を持つ多分小型のウィンドウの命令は、分岐誤予測または実行例外とは無関係に実行するようにスケジュールされる。積極的な実行をスケジュールするためにDFB62内のLSU200に識別することができる「インレンジ」となっているのはメモリ参照命令である。第23図は厳密な状態を維持しつつロード/ストア命令を含む長レイテンシ命令を積極的にスケジュールする本発明の方法の実施例を概略的に示すフローチャートである。

40

#### 【0181】

##### 2. ベーシック構造への強化と長レイテンシ(ロード/ストア)命令の積極的スケジュール

本発明のベーシックな方法へと強化する際には、ロード/ストア命令とメモリを参照す

50

る他の命令を更に積極的にスケジュールできるようにしてNMCSNを進め性能を改善するために「非故障」として知られている所定の命令を無視するように規則を設定することができる。

【0182】

#### IV. チェックポイント法

チェックポイントは既知の瞬間におけるマシン状態の「スナップショット」である。チェックポイントは、誤予測分岐または実行例外が生じた場合に理論上の命令順序を元に戻すために前のマシン状態を素速く回復するための方法と構造を提供する。本発明による典型的なプロセッサでは、命令発行サイクルの間にチェックポイントが作られる。本発明の方法では、チェックポイントの回復に基づくマシン状態の回復はマシンの「バックアップ」と定義される。

10

【0183】

本発明ではチェックポイントは数クラスの命令用に創られる。即ち、(1)分岐のような予測プログラム制御転送命令と(2)制御レジスタ値を変更するサイドエフェクトを持っているかも知れない命令である。

【0184】

制御転送命令(分岐のような)は予測されプログラムカウンタ(PC)不連続を生ずるかも知れない。本発明の構造と方法を設定するCPU51では、プログラム制御命令(分岐またはリターン型のジャンプ及びリンク)が誤予測された場合に素速くマシンのバックアップを行うために、各予測プログラム制御命令に対してプロセッサがチェックポイントを作る。これらの状況でAPCとNAPCがセーブされないと、誤予測されたプログラム制御命令の再発行と再実行せずに正しいFPCとAPCとNAPCを再構成することは困難である。このようなプログラム制御命令の各々に対してチェックポイントが作られているので、予測できてしかも同時に未解決のままのプログラム制御命令の数は、マシン内に許容されるチェックポイントの最大数によって制限される。予測プログラム制御命令は前述したSPARC-V9 B Pr, FBcc, FBpcc, Bcc, BPcc及びJMPL [rd = 0]命令のような命令とすることができる。

20

【0185】

プログラムフローを変える他の型の命令もチェックポイントで調べることができる。これらの命令にはCALL, JMPL, TCC, RETURN, DONE及びRETRYのようなSPARC-V9命令が含まれる。

30

【0186】

制御レジスタ値を変更するサイドエフェクトを持つことができる命令を含むマシン状態を変更する命令に対してもチェックポイントが創られる。ある制御レジスタをチェックポイントすることによってこの制御レジスタを理論上変更することができる。これらはSPARC-V9 WRPR及びWRASR命令のような制御レジスタファイル800内の制御レジスタへの書き込みを含み、SPARC-V9 SAVED, RESTORED及びFLUSHW命令のようなレジスタウィンドウ制御命令も含む。しかし、性能を失わずにチェックポイントされる状態の量を少なくするためにすべての制御レジスタがチェックポイントされる訳ではない。

【0187】

更に、例示CPU51では、前述したように、制御レジスタのデータは後刻実行のためにFXU601で利用できるようにウォッチポイント304に記憶されるので、制御レジスタファイル800内の制御レジスタの読取りにはチェックポイントが必要である。これらはSPARC-V9 RDPR及びRDASR命令のような命令とすることができる。

40

【0188】

更に、発行トラップを頻繁に起す命令もチェックポイントを必要とすることがある。これらには、SPARC-V9 SAVE及びRESTORE命令のようなウィンドウ制御命令と、例えばSPARC-V9 LDD/LDDA及びSTD/STDA命令とすることができる設置計画されない命令が含まれる。

【0189】

トラップも理論上エンターされうるが、そのように発行した時にはチェックポイントさ

50

れる。従って、後に述べるように実行トラップ(etrp) または発行トラップを生ずる命令に先行して発行される命令にはチェックポイントを作ることができる。更に、理論上トラップが取られた場合に復帰が行えるように、トラップ操作ルーチンの前にチェックポイントが作られる。

#### 【0190】

理論的に制御することができないサイドエフェクトを持つ命令、即ち、例外を生ずることが稀で多量のチェックポイント情報を必要とする命令に対しては、効率上の理由から命令のチェックポイントを行わず、その代り発行前にマシンを強制的に同期化するように決定を行うことができる。即ち、理論的に制御できないサイドエフェクトを持つ命令を発行する前に、CPU をマシン同期とし、すべてのペンディングな命令がコミットしリタイヤされた (ISN = CSN = RRP) とするのである。

10

#### 【0191】

どの命令をマシン同期命令とし、どの命令をチェックポイントするかを選択する時には重要な性能 / 設計の引き渡しが行われる。同期のための性能低下は、チェックポイントされたマシン状態を記憶するために要するチップ領域を含め、この種の命令の理論上の操作に関係する追加のロジックの複雑性とプロセッサ領域と比較して評価される。

#### 【0192】

多くの異なる型の命令がレジスタ中のデータ値を変更する。従来のチェックポイント法ではレジスタ状態を回復するためにレジスタデータ値をチェックポイントする。対照的に、本発明の方法をマシン設置する時には、実際のレジスタデータ値をチェックポイントせず、代わりにレジスタ再命名マップをチェックポイントすることによってCPU51 が以前のレジスタ状態を回復することができる。CPU51 に記憶されているチェックポイントされた状態情報の量を大幅に減少できるので、「レジスタデータ値」ではなくて「レジスタ再命名マップ」をチェックポイントするのが有利である。

20

#### 【0193】

例示したマシンでは、コンカレント命令のサポートリング64により、16個までのチェックポイントを任意の時に利用できる。実施例ではISU200は発行サイクルにつき最大1個のチェックポイントの割り当てに制限されている(ハードウェアの考慮により)。しかし一般的には、1サイクルで1個または任意の複数個のチェックポイントを割り当てることができる。例示したCPU51 では、同一のマシンサイクル中に発行できる命令の型と組み合わせを制限するように所定の規則が設置されている。例えば例示したCPU では、1個のマシンサイクル中1個の制御転送命令だけを発行でき、従ってそのサイクルに状態をセーブするに要するチェックポイントは1個だけである。このチェックポイントの制限により、サイクル中に最大数(例えば4個)の命令を発行できない場合(発行ウィンドウで待期中の命令の型による)がある。更に多数または少数のチェックポイントを供給することもできるが、その数はマシンの中で共時的に目立つチェックポイントを必要とするナンバー命令のようなファクターに基づいて選択される。

30

#### 【0194】

実施例では、能動命令リング312 が64個もの命令を同時にサポートする。後に説明するように、本発明の構造と方法は、ある種の命令型式のチェックポイントを行い、ISU200からのTIMEOUT\_CHKPNNT信号に応じて所定の命令サイクルタイムアウト間隔で任意の命令型式のチェックポイントを任意に行う。従って、サポートされるチェックポイントの数は能動命令リング312 によってサポートされる命令の数より大きくないことが必要で、この数は一般に命令の数より小さい。例示のCPU51 では16個のチェックポイントを割り当てることができ、これらのチェックポイントの各々はチェックポイントを生じた命令がリタイヤされた後に回復されるマシン資源になっている。

40

#### 【0195】

例示のCPU51 では、チップのルートを減らし性能を向上させるため、DO\_CHKPT信号に応じてチェックポイントデータ(マシン状態)はチェックポイント記憶ユニット内のCPU51 全体に局所的にセーブされる。しかし、チェックポイントされたデータを記憶消去された

50

チップにできることは自明である。

【0196】

チェックポイント記憶ユニットの各々は16個のアドレス可能記憶素子または項目を持っている。各記憶項目は後述する16個のチェックポイント番号の中の1個に対応する。従って第6図を参照すると、DO\_CHKPT信号に応じてAPCとNAPCレジスタ113及び114 (CHKPT\_PC及びCHKPT\_NPC)の内容はPCロジック106によってPCチェックポイント記憶ユニット107に記憶される。同様に、制御レジスタの内容 (CHKPT\_CNTRL\_REG)は、第17図に示すように制御ロジック816により制御レジスタチェックポイント記憶ユニット817に記憶される。更に第16図に示すように、FXRFRN604とCCRFRN610と、FPRFRN605の各々の再命名マップを表わすデータはチェックポイント記憶ユニット616に記憶され、フリーリストロジック701のフリーリスト (CHKPT\_FREELIST)を表わすデータはフリーリストチェックポイント記憶ユニット702に記憶される。そして後述するようにトラップスタックユニット815の或る種の内容もチェックポイントされる。

10

【0197】

ISU200はチェックポイントを作る時機の決定に責任がある。BRB59から受信した命令をデコードして、どれかがチェックポイントを要する前述の型であるか決定する。そして後述するように、所定数のマシンサイクルが行われた後にチェックポイントを作るようにPSU300がTIMEOUT\_CHKPT信号によって指示した時、チェックポイントを作る時機を決定する。この結果、ISU200がチェックポイントを作ることを指示し、そのチェックポイントのチェックポイント番号を指定するDO\_CHKPT信号を発生する。

20

【0198】

第24図はPSU300内にあるチェックポイントユニット303の機能ブロック図である。ISU200によってDO\_CHKPT信号を主張すると、1個または2個以上の命令に対してチェックポイントを作るようチェックポイント (CKPT) 303に指示し、命令スロット0-3で発行されたどの命令がチェックポイントを要求したかを示す。少なくとも1個のフリーチェックポイントレジスタが利用できなければチェックポイントを割り当てることができないから、DKPT303はISU200へCHKPT\_AVAIL信号を送らなければならない。CHKPTは少なくとも個のチェックポイントを割り当てることができることをISUに知らせ、チェックポイント番号が各命令と関係づけでき、ISU200によって作られたチェックポイント番号 (CHKPTs)に合体でき、DFB62へ進められるようにチェックポイント番号をISUに知らせる。これらの信号はISU200によるDO\_CHKPT信号の発行に先行する。チェックポイント番号を割り当てることができない場合は、チェックポイント資源が利用できるようになるまでISUによる信号発行を待機させる。簡単に言えば、実行例外または誤予測から回復させるのに必要な場合だけ記憶されているチェックポイントを使用し、そうでない場合はチェックポイントの割り当てを解除し記憶スペースを解放して再利用するのである。(チェックポイントを含むマシン資源の割り当てと割り当て解除については後に詳述する。)後述する本発明のベーシック構造と方法へ強化するにも、所定数のマシンサイクルまたは所定数の命令が発行され、チェックポイントを作ることなく終結した時にデコードされた命令型式とは無関係にタイムアウトチェックポイントが作られる。

30

【0199】

チェックポイントユニット303はチェックポイントのトラックを維持し、ISU200が指示した時にチェックポイントを割り当て、必要でなくなった時にチェックポイントを解放する。コミットドシリアル番号がチェックポイントのシリアル番号(CSN>SN)  $i + 1$ よりも大きくなるとチェックポイント  $i$ をリタイヤさせることができるが、これは本発明の方法と構造によればマシン状態を任意の命令の境界に再記憶させることができるので、コミットされた命令よりも順序的に前にある命令までマシンを逆追跡する必要がないからである。PSU300はシステムの他の構成要素にもいつチェックポイントを割り当て解放するかを知らせるので、これらの構成要素はリタイヤされたチェックポイントに使用されていた局所データ記憶領域を再使用することができる。チェックポイントユニット303はまた、いくつかのチェックポイントが自由に割り当てできるかをISU200に知らせる。チェックポイン

40

50



トの使用については、後にバックアップ兼バックステップユニットに関連して詳述する。

#### 【0200】

チェックポイント303 は5種の主要機能を持っている。第1に、ISU200からのDO\_CHKPT信号に応じて、チェックポイントレジスタユニット352内のチェックポイントデータレジスタ構造(CHKPNT\_REG\_i)のリストの保守によってISU200の要請する通りにチェックポイントの割り当てを制御する。制御転送信号が発行されると、DO\_CHKPTは1個または2個以上のチェックポイントが作られるべきことと、それぞれのチェックポイント番号の位置(例えば0-15)を指定する。本発明の構造と方法を強化する際には、後にベーシックなチェックポイント法への強化において述べるように、TIMEOUT\_CHKPTがチェックポイント303によって発生されISU200へ送られて命令型式とは比較的無関係に所定のタイミングでチェックポイントを作る。このタイムアウトチェックポイントは、チェックポイントが作られなかった命令の効果を元に戻そうとするサイクルの最大数を決めて制限するために供給される。(チェックポイントできる命令型式が同じ命令ウィンドウから発行される時には、タイムアウトチェックポイントの形成は延期されるかキャンセルされる。)

10

#### 【0201】

第2に、チェックポイントは或る与えられたチェックポイント(n)を追い越して進んだCSNに基づくチェックポイントと仮に次に位置しているチェックポイント(n+1)をリタイヤ(クリヤ)させる。特定のチェックポイント番号を過ぎたCSNを決めるメカニズムは、CSNをチェックポイントレジスタに記憶されているシリアル番号と比較することである。チェックポイントレジスタに記憶されているシリアル番号とは、チェックポイント形成した命令の次の命令のシリアル番号である。CSNがチェックポイントnとn+1を追い越して進んだ場合にだけ、例えばICRU301からのISNとCSNに応じてチェックポイントnをリタイヤさせることができる。

20

#### 【0202】

第3に、チェックポイント303は、ISUによってチェックポイントが割り当てできるか否かとチェックポイントの位置番号を含め、ISU200へ送られる信号CHKPT\_AVAILによって割り当てられる次のチェックポイントは何かをISU200に知らせる。第4に、チェックポイント303は割り当てられたチェックポイントに対応する命令のシリアル番号を記憶し、他のユニットによって要請された時に、ICRU301に対してチェックポイントに関連するSNとSN+1の両方を供給する。第5に、顕著な有効チェックポイントの中のどれを殺すかをバックトラックユニット305内のバックアップユニット402に示す命令キルベクトル(KILL\_VEC)を発生し、信号CHKPNT\_SNとCHKPNT\_SN\_INCによってICRU301に対し、マシンのバックアップに関連してどのチェックポイントのリタイヤさせられるかを知らせることによって実行マシンのバックアップに参加する。(マシンのバックアップは後に詳述。)

30

#### 【0203】

##### A. チェックポイント割り当てレジスタ

チェックポイント割り付けレジスタ352の構造は表1に概略的に示してある。最初の6ビットは状態ビットである。ビットVはチェックポイントの有効性(割り付け)を示す。ビットCP(チェックポイントが追い越されている)はCSNがこのチェックポイントのSNを追い越したかを示し、QはCSN前進トラックを維持する5種の状態のうちの1つを示し、GT(より大きい)はCSNが割り付け時のチェックポイントのシリアル番号よりも大きいか否かを示す。ビットGTはCSNがいつこのチェックポイントを追い越したかを決定する。固有の機能性を得るためにビットVが「1」にセットされビットCP「0」にセットされるチェックポイント位置0以外では、ブーツアップ時にビットVとCPをゼロにセットすべきである。TSNフィールドはチェックポイントされたシリアル番号に後続している次のシリアル番号を示す。この設置計画では、チェックポイント番号はセーブされず、位置「i」用のチェックポイントデータ構造にVビットをセットするとチェックポイント「i」が割り付けられたことが示される。チェックポイントレジスタ(CHKPNT\_REG\_i)に可能な状態は表2に説明してある。例示したCPUでは、チェックポイントの割り付け中、ビットQは「000」か「001」にリセットされる。ビットQはAリング312の周囲のCSNの進行に基づ

40

50

く状態を変える。表3にQビットの状態と遷移を示す。

【0204】

【表1】

表1. 典型的なチェックポイント割り付けレジスタ (CHKPNT\_REG\_i) の構成

i	1-ビット	1-ビット	3-ビット	1-ビット	6-ビット
0	V	CP	Q	GT	NSN
1	V	CP	Q	GT	NSN
⋮	⋮	⋮	⋮	⋮	⋮
14	V	CP	Q	GT	NSN
15	V	CP	Q	GT	NSN

10

【0205】

【表2】

表2. 典型的なチェックポイントレジスタ (CHKPNT\_REG\_i) に可能な状態

V	CP	GT	説明
0	X	X	チェックポイントが割り付けられていない。チェックポイントが能動でない場合、CPは常にゼロにセットされる。
1	0	0	チェックポイントが割り付けられている。割り付け時にはCSNはこのチェックポイントのシリアル番号より小さく、CSNはまだチェックポイントを追い越して進んでいない。
1	0	1	チェックポイントが割り付けられている。割り付け時にはCSNがチェックポイントのシリアル番号より大きく、CSNはまだチェックポイントを追い越し進んでいない。
1	1	X	チェックポイントが割り付けられている。CSNはチェックポイントを追い越している。GTは有効でない(注意不要)。次のチェックポイントとも「11X」状態ならば、このチェックポイントをリタイヤさせることができる。

20

30

40

【0206】

## 【表 3】

表 3. 典型的な Q ビットの状態と遷移

Q の状態	状態の説明
000	チェックポイント割り付け中 $0 < CSN < 32$ ならば、現在形成されているチェックポイントの初期状態。
001	チェックポイントの割り付け中 $31 < CSN < 64$ 用に現在形成されているチェックポイントの初期状態。
010	割り付け以後、63/0 境界を横切って CSN が進んでしまったチェックポイントの状態。
011	割り付け以後 CSN が 63/0 境界を横切って 2 回進んでしまったチェックポイントの状態。
1XX	CSN が 63/0 境界を横切って 2 回進んでしまい、CSN が現在 32 よりも小さいチェックポイントの状態。X は注意不要を示す。

10

## 【 0 2 0 7 】

## B. チェックポイントの割り付け

20

命令 SN 用のチェックポイントを割り付ける時には、チェックポイント割り付けロジックユニット 351 が、チェックポイント「1」が有効でリタイヤされていないことを示すために CHKPNT\_REG<sub>i</sub> で V ビットを 1 にセットし、CP ビットをゼロにセットする信号をチェックポイントレジスタとコントロールユニット 352 へ送る。割り付けられるチェックポイントは、ISU に送られて割り付けに使用できる次のチェックポイント番号を知らせる信号と、チェックポイントの形成を可能にさせる信号を含みチェックポイントを作る命令のシリアル番号を識別する ISU からの DO\_CHKPNT とによって CAL351 内で決められる。チェックポイント割り付けの時に CSN の位置を示すため、ビット Q は表 3 に示すように書き込まれる。この情報はリングの周囲（例えばロケーション 63 から逆進してロケーション 0 まで）からの CSN の遷移を検出するために用いられる。割り付けられたチェックポイントは、チェックポイントのシリアル番号を識別し駆動信号としても働く信号 DO\_CHKPNT によって決定される。更に、チェックポイントに対応するシリアル番号はチェックポイント割り付けロジックユニット 351 によって計算されるが、1 だけ大きくされたシリアル番号 (NSN = SN + 1) だけがレジスタに記憶される。シリアル番号 NSN は 2 種の異なる条件で用いられる。第 1 に、NSN は ICRU301 へのデータバスで大きくされたチェックポイントのシリアル番号になる。第 2 に、NSN はチェックポイントに後続する命令に故障命令があるか否かを決定するのに用いられる。SN ではなくて NSN (SN + 1) をセーブすることによってロジックを節約し、チェックポイントの後のシリアル番号を識別する信号の発生を高速化するのである。この代りに、大きくされたシリアル番号信号を発生するために設けられているレジスタとロジックに SN そのものを記憶することができる。

30

40

## 【 0 2 0 8 】

次のようなチェックポイント割り付け法が典型的である。チェックポイント形成信号が主張される (DO\_CHKPNT) と、V ビットフィールドが「1」にセットされ (CHKPNT\_REG<sub>i</sub> [V\_\_フィールド] = 1)、CP ビットフィールドがクリアされる (CHKPNT\_REG<sub>i</sub> [CP\_\_フィールド] = 0)。次にチェックポイントが形成された命令のシリアル番号がコード化された 4 ビットのチェックポイントロケーションベクトルから引き出されてチェックポイントのシリアル番号フィールドに記憶される (CHKPNT\_REG<sub>i</sub> [SN\_\_フィールド])。例えば U51 の実施例では、「0000」はチェックポイントロケーション NISN を示し、「0001」は NISN + 1 を示し、「0011」は NISN + 2 を示し、「0111」は NISN + 3 を示し、「1111」は NISN + 4 を示す。

50

## 【 0 2 0 9 】

C . チェックポイントのリタイヤ

チェックポイントはチェックポイントクリヤ/リタイヤメントロジックユニット353でクリヤまたはリタイヤされる。チェックポイント303はサイクル毎にISU200によって要請される数だけのチェックポイントを必要とする。現在のチェックポイント(n)と次のチェックポイント(n+1)を通るCSNに応じてレジスタ352のVビットフィールドをクリヤすることによってチェックポイントがリタイヤされる。チェックポイント割り付けロジックユニット353は、チェックポイントレジスタ352でリタイヤされるチェックポイントを指示するために用いられるOLDEST\_ACTIVE\_CHECKPOINTと呼ばれるポインタを維持する。チェックポイントをリタイヤさせる手続きは次の通りである。チェックポイント(CKPSN)に記憶されているシリアル番号とユニットされたシリアル番号(CSN)ポインタ(例えばCKPSN\_CSN)との差を計算することによってリタイヤロジック353に「CSN\_PASSED」と呼ばれる信号が発生する。CSNはICRU301に維持され、各サイクルでチェックポイントユニット303へ送られる。CSNがある与えられたチェックポイント(n)と現在その次にあるチェックポイント(n+1)を追い越して進んだ時か、チェックポイントのシリアル番号を追い越したマシンバックアップがある場合だけ、チェックポイントをリタイヤさせることができる。

10

## 【 0 2 1 0 】

本発明の構造と方法の一実施例では、CSNをすべてのチェックポイントのシリアル番号と同時に比較するロジックを供給することにより、CSNがチェックポイントを追い越して進んでいることが決められる。比較の際には、CSNがチェックポイントのシリアル番号を追い越したか否かを決定するのに単に試験よりも大きいことだけでは不十分な場合があるので、Aリング312の特性の周辺のラップも考慮する(モジュロ・リング長)。

20

## 【 0 2 1 1 】

D . マシンバックアップに関するチェックポイントの保守

次のマシンサイクルを開始するためにKILL\_VEC信号を用意するようチェックポイントユニット303に知らせるWILL\_BACKUP信号とBACKUP\_CHECKPOINT信号をバックトラップ305から受信すると、KILL\_VECはコミットされている命令を殺す。殺される命令は、チェックポイントの命令のシリアル番号を指定するBACKUP\_CHECKPOINT信号と最後のコミット信号から引き出される。CSNとバックアップチェックポイントの間にあるすべての命令が殺される。

30

## 【 0 2 1 2 】

CPU51の一種の実施例では、BACKUP\_CHECKPOINTはバックトラック305内のバックアップユニット402によって発生される信号で、状態を回復するためにどのチェックポイントをバックアップするかCPU51に知らせる。ISNとバックアップチェックポイントの間にある命令のどれを殺すかを指示するため、命令キルベクトル信号(KILL\_VEC)もバックトラックによって発生される。例えば、命令キルベクトルは殺される命令を識別するホットビットを持つマルチホット信号とすることができる。この代りに他の命令型式を使用することもできる。

40

## 【 0 2 1 3 】

チェックポイント303は、バックアップ中にICRU301がISUを調整することができるように、チェックポイントのシリアル番号を識別するチェックポイントレジスタ情報をICRU301に供給する。チェックポイントユニット303は、未完浮動小数点例外またはデータ区切り点例外が検出された時(CHKPNT\_SN\_INC)、ISNを調整するためにICRUが使用するシリアル番号も供給する。

## 【 0 2 1 4 】

E . タイムアウトチェックポイントの強化

従来のチェックポイント技術では、命令のデコードされたアトリビュートに基づいてチェックポイントが作られる。連続するチェックポイント(チェックポイント境界のついた)の間の命令の数が制限されないように、チェックポイントは或る種の命令のアトリビュ

50

ートに対してだけ作られる。命令実行例外が起こった時、故障命令以前のCPU51の状態を回復するためにCPU51は多数のマシサイクルを費さなければならないかも知れない。従って、故障命令に達する前にCPU51が命令を再実行するか元に戻すために費す時間は、チェックポイントの境界内にある命令ウィンドウのサイズと命令の数の関数になる。ここでの命令ウィンドウのサイズとは、CPU51内で任意の時間に目立つことを許される命令の最大数のことである。(例示したCPU51では、任意の時間に64個までの命令が目立つことができる。)

#### 【0215】

本発明の構造と方法に強化する場合、チェックポイントはデコードされた命令のアトリビュート(だけ)に基づいて作られるのではなく、むしろCPU51内のタイムアウト状態によって作られる。タイムアウト状態はチェックポイントを作らずに経過した命令発行サイクルの数に基づいている。例えば、最近発行された命令が従来チェックポイントを作られない型であればチェックポイントは形成されていない。タイムアウトチェックポイントは命令の最大数をチェックポイントの境界内に制限する。命令ウィンドウのサイズがチェックポイント境界内にある命令の最大数よりも大きい限り、例外が生じた場合に、プロセッサは従来の命令デコードに基づくチェックポイント技術よりも速くチェックポイントされた状態を回復することができる。更に、本発明のチェックポイントの強化により、CPU51の状態回復の命令ウィンドウサイズへの依存性が除去される。

10

#### 【0216】

タイムアウトチェックポイントは従来のチェックポイントを設置する構造と方法でも使用することができる。更に、タイムアウトチェックポイントは、本発明のバックアップ及びバックステップ構造と方法と同様、従来のプロセッサバックアップ技術と共用することができる。

20

#### 【0217】

タイムアウトカウンタ355は、そのマシサイクル中に少なくとも1個の命令が発行されたことを示すINSTRA\_ISSUED信号をISU200から受信して命令発行サイクルの数をカウントする。チェックポイント(命令デコードに基づくチェックポイントまたは以前のタイムアウトチェックポイント)を作ることなく経過した命令発行サイクルの数が所定のカウンタ数を越えると、タイムアウト状態が生じたのでチェックポイントを形成すべしというTIMEOUT\_CHKPNT信号をISUに送る。この時ISU200はチェックポイントを形成すべしと示すDO\_CHKPT信号を発生する。前述したように、発行された信号がチェックポイントを必要とする型であるとISU200が決定した時にもこの信号を発生させることができる。チェックポイントが実際に形成された場合、そしてこのような場合だけ、ISUからのDO\_CHKPNT信号によってカウンタ356がセットされる。チェックポイント境界内に許される命令の最大数は、カウンタの最大命令発行率(マシサイクル毎に発行される命令の最大数)とタイムアウト値によって決められる。

30

#### 【0218】

タイムアウトチェックポイント上の変化が有利であろう。例えば、タイムアウトチェックポイントと同一サイクルに正当に発行される命令デコードが正当である場合、命令デコードに基づくチェックポイントが優先的に作られてタイムアウトカウンタをリセットするように働く。更に、この時すべてのチェックポイントが割り付けられてタイムアウトチェックポイントが正当になった場合には、命令デコード型のチェックポイントのために厳密な状態を維持するために要するようなマシ停止を行わずに、所定の規則(例えば1サイクル延期)に従ってタイムアウトチェックポイントを遅らせることもできる。

40

#### 【0219】

#### F. 任意の命令境界での厳密状態の維持及び回復

本発明の方法によれば、任意の命令境界で厳密状態を維持することができ、その命令境界に状態を回復するために要する情報量を最小にすることもできる。ここで言う「命令境界」とは、命令が発行される直前(または直後)にある連続する命令の組の中の点に対応する。各命令が命令境界を作る。

50

【0220】

従来の命令のチェックポイントではプロセッサ内のチェックポイント境界（即ち、チェックポイントされた命令でだけ）厳密状態を維持することができるが、従来のチェックポイント法は各命令で用いられ、そのように用いられた場合でもサイクル毎に複数命令を発行するマシンに対してはサイクル毎に同数のチェックポイントが必要であるため、CPU51の性能を改善するよりも阻害するであろう。本発明では、サイクル毎に1個の命令だけがチェックポイントを要するように種々の命令の発行規則が設定され、バックステップ法と構造がより細かい命令レベルの回復を行う。命令のチェックポイントは理論上の命令順序によって不正に変更されるかも知れない状態のようなアーキテクチャ（マシン）状態を保護し後に記憶するために命令のチェックポイントが使用される。

10

【0221】

チェックポイント境界に厳密状態を維持する従来のチェックポイント法が知られている。しかし、マシン状態を捕捉し後に回復する方法としてだけ従来のチェックポイント法を使用すると、理論上順序外の実行中、間隔の狭い命令境界に厳密状態を維持することができない。

【0222】

G. チェックポイントされるデータ量を減らすよう所定命令に対してマシンを同期させる方法

チェックポイントは、命令型式とは無関係な固定サイズまたはチェックポイントが作られているオペランドである。例えば、200個の状態値を変えることができる命令を持つマシン(CPU)では、命令自体がせいぜい8個の値を変更できる場合でも、形成された各チェックポイントは100個の状態値を記憶しなければならない。本発明の実施例では、特定の命令の各々によって変更される状態の量と典型的な処理において特定命令の各々が発生する統計上の頻度を考慮して、すべてのチェックポイントのチェックポイントされたデータの量を減らすように構造と方法が設定される。

20

【0223】

この設計技術を用いると、最適なチェックポイントサイズを決めてマシンに設定することができる。第1に、各命令によって変更できる状態が決定される。第2に、ある代表的な名目上の処理作業に対して各特定命令が発生する統計的な頻度が決められる。各命令用の状態記憶要件が検討される。設計の目標は、稀にしか発行されず頻繁に発生する命令よりも比較的多量のチェックポイントを必要とする命令を識別することである。チェックポイントする命令を決めるにはリニヤまたはノンリニヤな最適化技術を用いることができ、チェックポイントされた状態を減らすために最大チェックポイント記憶に基づく非算術的な規則を用いることもできる。単純化した例を次に説明する。

30

【0224】

チェックポイントされる状態が減少している単純化した例  
命令の組の中のどれかの命令によって変更できる状態

【数1】

状態1	状態2	状態3	状態4	状態5	状態6	状態7	状態8	状態9	状態10
-----	-----	-----	-----	-----	-----	-----	-----	-----	------

40

【0225】

命令Aによって変更(mod)できる状態

【数 2】

mod	-	-	mod	-	-	-	-	-	-
-----	---	---	-----	---	---	---	---	---	---

【0 2 2 6】

命令 B によって変更(mod) できる状態

【数 3】

10

mod	-	-	-	-	-	-	-	-	-
-----	---	---	---	---	---	---	---	---	---

【0 2 2 7】

命令 C によって変更(mod) できる状態

【数 4】

-	mod	mod	-	mod	mod	mod	-	mod	mod
---	-----	-----	---	-----	-----	-----	---	-----	-----

20

【0 2 2 8】

命令 D によって変更(mod) できる状態

【数 5】

-	-	-	-	-	-	-	mod	-	-
---	---	---	---	---	---	---	-----	---	---

30

【0 2 2 9】

命令 A , B , C 及び D の各々がチェックポイントされる場合には、各チェックポイントは状態 1 ~ 10 のすべてを記憶しなければならない。しかし命令 C は稀にしか実行されないため稀に行われるこの命令の実行中マシンを同期させることによって CPU51 のスローダウンが必要である場合には、命令 C のチェックポイントを行わないことによってチェックポイントされる状態を 10 状態から 3 状態、即ち、状態 1 と状態 4 と状態 8 に減らす。

【0 2 3 0】

マシンを同期させると、ペンディング命令がコミットしリタイヤされるまでマシンの同期を要する命令の発行が遅れ、次に同期を要する命令が順次連続して実行される。マシンが同期モードで作動している時には、状態への逆書き込みが行われる前に例外条件が識別されるので、厳密状態を維持するために命令のチェックポイントをする必要はない。

40

【0 2 3 1】

H . チェックポイントされるデータ量を減らすようにレジスタリネームマップをチェックポイントする方法

更に、誤予測分岐、故障、実行エラーその他の処理中の異状からマシン状態が回復できるようにチェックポイントが発生された時の全機状態が記憶されるため、従来のチェックポイントは多量の情報を含んでいる。従来のチェックポイント法では変化する状態だけを変更する試みは行われていない。

【0 2 3 2】

例えば、64 個の命令を並行して発行することができる従来のチェックポイント法を CPU5

50

1 内の各命令に適用すると、各命令に対して1個、つまり64個までのチェックポイントを維持しなければならない、4発行のマシンでは、マシンサイクル毎に4個ものチェックポイントを作らなければならないことになる。従来のチェックポイントに記憶される状態の特定の型と量はCPU51のアーキテクチャと命令の組によって異なるが、従来の各チェックポイントは任意の命令によって変更できる各状態用の状態情報を含んでいる。即ち、命令が状態を変更し得るか否かに拘らず、従来のチェックポイントは各変更可能状態用の状態情報を含んでいる。従来のチェックポイント法では、チェックポイント毎に変化する状態だけではなく、マシン状態を回復するに必要なすべてのマシン状態が各チェックポイントに対して書き換えられる。

### 【0233】

本発明の方法では、レジスタがリネームされリネームマップがレジスタリネームマップに記憶されている。レジスタのリネームについては他の箇所で説明する。レジスタ名のリネームは処理に利用できるアーキテクチャレジスタの数を効果的に増加するのに役立つもっと多数の物理レジスタにアーキテクチャレジスタをマッピングする方法である。本発明の方法では、データ値そのものではなくてリネームされたレジスタをチェックポイントとすることによって、新しいチェックポイント法に必要な記憶の量を減らす。前述した通り、以前に発行されたすべての命令が実行されコミットされ、その資源がリタイヤされる(ISN = CSN = RRP)ように、厳密状態を維持するために多量のチェックポイントされたデータを必要とする或る種の所定の命令はマシン同期を起させる。同期型の命令故障が誤予測されると、状態は変更されず、命令の実行結果が状態を変更することを許される前に、故障が処理されるかまたは代りに命令通路が追跡される。

### 【0234】

マシン状態が素速く回復されるように、コントロールレジスタを変更できる非同期を要求するすべての命令はチェックポイントされる。命令によって変更されたすべての状態は、前述のようにチェックポイントされてCPU51内に局所的に記憶される。更に、コントロールレジスタを変更しない命令を実行する状態はCPU51をバックステップすることによって記憶できる。第26図は、タイムアウトチェックポイントを形成する他のチェックポイント法に適用できる任意のチェックポイント強化法を含む本発明のチェックポイント法の実施例の概略的フローチャートである。

### 【0235】

#### V. 誤予測及び例外からの回復

前に言及したように、第5図の命令パイプラインでは、プログラムコントロールの誤予測と例外が時折生ずる。この時にはエラーを検出して誤予測または例外を生じさせた命令の直前の状態にCPU51を戻すように操作しなければならない。

### 【0236】

#### A. 同時的な複数の分岐/ジャンプアンドリンク命令の評価のためのウォッチポイントによる誤予測の検出

命令を理論上で実行できるプロセッサに対しては、命令を解決する前に分岐方向(採用または非採用)または分岐アドレス(目標アドレスまたは分岐命令の次のアドレス)を予測することができる。後にこれらの予測が誤っていたと判明すると、CPU51は以前の状態に戻り正しい分岐ストリームで命令の実行を開始する。従来のプロセッサでは、1サイクルでの評価のためにしばしば複数の分岐が用意される。しかし、従来のプロセッサはサイクル毎に1個だけの分岐予測を評価する。従って、評価のために用意はされたが選択されなかった分岐評価は遅延されなければならない。分岐評価が遅延するとCPU51の性能が低下し、分岐をできるだけ早い時期に評価することによって性能を著しく改良することが出来る。

### 【0237】

本発明のウォッチポイントの構造と方法は従来技術よりもいくつかの点で有利である。第1に、複数の予測分岐またはジャンプアンドリンク命令を同時にモニタまたは監視することができる。第2に、複数の実行ユニットからのデータ拡大バスを同時にモニタまたは

10

20

30

40

50



監視して予測分岐またはジャンプアンドリンク命令が待っている複数のデータまたは計算されたジャンプアンドリンク (JMPL) アドレスをつかむことができる。第3に、分岐またはジャンプアンドリンク命令の複数の誤予測信号を発生することができる。第4に、単一の共有記憶領域内に交代の分岐アドレスまたは予測ジャンプアンドリンクアドレスを記憶することができる。第5に、ウォッチポイントによって誤予測が検出された時に命令を取り出すための正しい分岐アドレスまたはジャンプアンドリンクアドレスを送ることができる。

#### 【0238】

第27図を参照すると、ウォッチポイント(WPT) 304 は発行された各チェックポイント済みのコントロール転送命令の実行及び予測状態のモニタに責任がある。従って、WPT304は、分岐、無効化分岐と、BrXcc、条件BrIcc、条件BrFcc、JMPL rt (1) (rd=0) 用の条件情報とコントロールレジスタ(CCR及びFSR レジスタを除く)への読取り/書き込みを含むジャンプ・リンク命令を含んでいるDO\_WTCHPT 信号と共にDO\_CHKPNP 信号を受信する。条件コード情報は後に、命令順序が正しく予測されたかそれとも元に戻す必要があるかを決めるのにWPT304によって使用される。WPT304については本明細書の他の箇所で詳細に論議する。

10

#### 【0239】

ウォッチポイントユニット304 は、DFB62 からのデータ拡大バスによって到来する実行完了信号のモニタまたは監視と、例えば分岐誤予測のような所定の事象が検出された時をCPU51 に知らせることに責任を持ち。ウォッチポイント304 内のウォッチポイントレジスタ491 は多目的である。これらのレジスタは分岐予測とJMPL予測をチェックし、本来の予測が正しくない時にバックアップ後に命令を参照するために正しい分岐/JMPLアドレスをBRB59へ送る。これらのレジスタはデータ拡大分配バスにあって数個のユニット(FXU, FPU, LSU 及びFXAGU)からの結果をモニタするので「ウォッチポイント」レジスタと呼ばれる。16個のウォッチポイントレジスタがあり、各チェックポイントに対して1個となっている。

20

#### 【0240】

整数実行ユニット(FXU)と、整数及びアドレス発生実行ユニット(FXAGU)と浮動小数点実行ユニット(FPU)とを持つCPUに関連して例示したウォッチポイントユニット304を説明する。この説明からして、設定する実行ユニットの数を大きくしたり、小さくしたり、変えたりすることができることは当業者には自明であろう。特に、例示したCPU51の実行ユニットをマッチさせるために、FXUとFXAGUが追加されるであろう。

30

#### 【0241】

第27図はウォッチポイントユニット304内にある主な機能ブロックの機能ブロック図である。BRB59内のフェッチユニット(FETCH)は命令を取り出してISSUE200に与え、命令のデコードに基づいて分岐とJMPL'sを予測する。ISSUE200はウォッチポイント304とFSR/CCRFRNへDO\_CHKPNP 信号を送り、ウォッチポイント304へDO\_CHKPNP 信号を送る。FSR/CCRFRNは条件コードのリネーム(CCリネーミング)に責任がある。DO\_CHKPNPはチェックポイントの形成を開始するMAKE\_CHKPNP 信号と、形成されるチェックポイント番号を識別するNEXT\_CHKPNP 信号とを含む。DO\_WTCHPT 信号は分岐またはJMPL信号が発行され分岐の型を識別したことと、発行された命令中の条件フィールドと、条件コード(CC)タグと、発行された命令中の次の交代のプログラムカウンタアドレス/予測ジャンプアンドリンク命令アドレスをウォッチポイント304へ知らせる。FETCHはまた、DO\_WTCHPT 信号内のWR\_ANPC 信号という形で分岐/JMPLアドレスをウォッチポイント304へ知らせる。分岐またはJMPL或いは他のコントロール転送命令(分岐/JMPL)が発行されると、ISSUE200は分岐/JMPL命令発行済み信号(DO\_WTCHPT)を送るが、これは信号JMPLを含む発行済みの各予測分岐またはJMPL用の信号である。WP\_ANPCは目標RAMに書き込まれ、PCロジック106によって計算された目標FPCを示す。ANPCは予測分岐命令とJMPL命令の命令デコード中に決められる。JMPL命令に対しては、DO\_WTCHPT 信号内のウォッチポイントへ送られたWP\_ANPCが予測目標FPCになる。予測分岐命令に対しては、DO\_WTCHPT 信号内のウォッチポイント

40

50

へ送られたWP\_ANPC が交代のFPC、つまり交代の分岐方向のFPCになる。誤予測が生ずると、FPC、APC及びNAPCを計算するバックアップの間、ウォッチポイント内のANPC出力ロジックからの出力されるRD\_ANPCが使用される。リネーム600 (FSR / CCRFRN606 と呼ばれるの条件コード (CC) 部分はCCレジスタのリネーミングを管理し、命令発行後にウォッチポイント304へCCリネーミング信号を送る。CCリネーミング信号は次に説明するようにCC\_TAG\_RENAMED\_Cと、CC\_TAG\_Cと、CC\_DV\_Cと、CC\_DADA\_Cを含んでいる。

#### 【0242】

実行ユニット (FXU601, FXAGU 及びFPU600) の各々はデータ拡大バスによって条件コードタグ (CC\_TAG\_F) と条件コードデータ有効 (CC\_DV\_F) と条件コードデータ (CC\_DATA\_F) をウォッチポイント304とFSR / CCRFRN606の両方へ送る。FXUからのタグとデータ有効とXCCデータ信号とICCデータ信号は、FXU\_XICC\_TAGS\_Fと、FXU\_XICC\_DV\_Fと、FXU\_XC  
C\_DATA\_FとFXU\_ICC\_DATA\_Fを含んでいる。FXAGUからの信号は、FXAGU\_XICC\_TAGS\_Fと、FXUAG\_XICC\_DV\_Fと、FXAGU\_XCC\_DATA\_FとFXAGU\_ICC\_DATA\_Fを含んでいる。FPUからの信号は、FPU\_FCC\_TAGS\_Fと、FPU\_FCC\_DV\_Fと、FPU\_FCC\_DATA\_Fを含んでいる。XCC, ICC及びFCC条件コードの例と書式は例えばSPARC-V9に説明してある。

#### 【0243】

第28図を参照すると、ウォッチポイント304は、実行ユニット (FXU, FPU及びFXAGU) からのCCデータ拡大バスから直接、またはFSR / CCRFRNから到来するリネームされたCC信号から条件コードをつかむCC\_\_グラッピングロジック1000と、分岐またはJMPL命令がデコードされる度毎にBRB59から到来する予測条件コードデータを含むウォッチポイント情報の記憶に責任を持つウォッチポイント記憶素子1002及び関連する読取り / 書き込みコントロールロジックと、実行ユニットから到来する (直接またはリネームユニットを経由して) 計算された条件コードまたは実際の条件コードの比較を行ってこれらの条件コードを個々のウォッチポイント記憶素子1002に記憶されている予測条件コードと比較することに責任を持つ評価ロジック (Evalロジック) とを含む。目標RAMとその関連するコントロール比較ロジック1003は、次に詳細に説明するように分岐またはJMPL誤予測からの回復に参加する。CCグラッピングロジック1000と、Evalロジック1001とWP\_\_素子10002は、CPU51内に割り付けできる16個のウォッチポイントの各々に対して1個、つまり16個の別々の構造を含んでいる。唯1個だけの目標RAM1003データ記憶構造が供給される。

#### 【0244】

16個のウォッチポイントの各々に対して数個のデータ素子が記憶される。各ウォッチポイントに対して記憶されるデータは、ウォッチポイント活性化時にセットされるウォッチポイント有効ビット (WP\_VALID[i] = 1) と、JMPL命令から分岐命令を区別する命令型式データ (WP\_JMPL [i] = 1はJMPLがウォッチポイントを必要としていることを示し、WP\_JMPL [i] = 0は予測分岐用にウォッチポイントが形成されていることを示す) と、命令中の条件フィールド (WP\_COND [i]) と、WP\_XCC, WP\_ICC及びWP\_FCC用の記憶装置とを含む。一種の典型的な実施例では、これらのデータ素子はラッチに記憶されている。CCタグはCC\_TAG\_ARRAYに記憶され、「ANPC」(交代用の次のプログラムカウンタアドレス) / 予測JMPL (ジャンプアンドリンク命令) アドレスは目標RAMに記憶されている。有効はウォッチポイントロジックによって書き込まれ、分岐の型、COND, CC - タグ及びANPC / JMPLアドレスはISU200から受信するDO\_WTCHPT信号から引き出される。DO\_WTCHPTに応じて各分岐 / JMPL命令に対してウォッチポイント記憶素子491が活性化され、フィールド選択書き込みロジックがDO\_WTCHPT信号をデコードし、ウォッチポイントに分岐型式と、CONDフィールドとANPC / 予測JMPLアドレスとを書き込む。

#### 【0245】

有効ビットを「1」にセットすることによってウォッチポイント項目 (エントリ) を活性化させることができるが、すべてのチェックポイント素子とそのサイクルで既に割り付けられてしまっている場合にはウォッチポイント (及びチェックポイントを必要とする命令の発行は延期される。複数マルチポイント素子を同時に能動 (有効) とすることができ、有効な (VALIDフィールドセット) すべてのウォッチポイント項目は各サイクルで同時に

10

20

30

40

50

評価することができ、誤予測信号を発生することができる。データ記憶に割り付けされるチップ領域を減らすため、ウォッチポイント304 は分岐命令用のANPCの個かJMPL命令用の予測ターゲットアドレスどちらかを同一の記憶位置に記憶することができる。割り付けられた多目的ウォッチポイントレジスタの実際の使われ方は、ISU200からのDO\_WTCHPT 信号の内容、即ち、ウォッチポイント項目が予測分岐用として作られているかまたは予測JMPL命令用として作られているかによる。ウォッチポイントレジスタは自ら割り付けを解いて分岐となり、ジャンプ予測が評価する。

【0246】

条件分岐命令の場合のように命令によって誤予測が生ずる可能性のある時には、ウォッチポイントユニット304 によって必ずウォッチポイント素子が割り付けされる。各ウォッチポイント素子是对应するチェックポイント素子を持っているが、逆は必ずしも真ではない。例えば条件分岐のように命令によって誤予測が生ずる可能性がある場合には必ずウォッチポイント素子が割り付けされるので、チェックポイント是对应する能動ウォッチポイント項目を持たなくてもよい。チェックポイントは誤予測を生ずる可能性のある命令（「SAVE命令のような命令または周期的な「タイムアウト」チェックポイントが分岐命令以外の命令用に作られた時）以外の命令に対しても割り付けできるので、すべてのチェックポイントがウォッチポイントを持っている訳ではない。ウォッチポイントはウォッチポイントユニット304 内の複数のウォッチポイント素子432 のうちの1個に記憶される。チェックポイント情報を記憶するために1個のチェックポイントレジスタが設けられる。しかし、実際のチェックポイント済みマシン状態は前述のようにCPU51 の全体に亘って局所的に記憶される。表4 は一組の典型的な命令用としてシリアル番号によってチェックポイント中に記憶された情報の型と対応するチェックポイントの例を示す。

【0247】

条件コードグラビングロジックユニット492 は、DFB62 内の実行ユニットからのデータ拡大バスに現われる時の条件コード（CC）を捕捉することと、命令が発行された時に予測され現在ウォッチポイント素子491 に記憶されている条件コードと捕捉された条件コードを比較する評価ロジックユニット493 ヘルート指定することに責任を持つ。捕捉された条件コードが予測条件コードと合う場合は、命令が正しく予測されたので元に戻される必要はない。しかし、グラビングユニットによって捕捉された条件コードが予測条件コードに合わない場合には、命令が誤予測されたので元へ戻さなければならない。

【0248】

【表4】

表4. ウォッチポイントとチェックポイント記憶の単純化した例

No.	チェックポイント	ウォッチポイント
0	SN=33, BRZ、マシン状態	BRZ, CC レジスタがCCレジスタタグを待つ。
1	SN=41, ADD、マシン状態	-ウォッチポイント不要
2	SN=44, JMPL、マシン状態	JMPL、レジスタタグ (jmpI アドレス計算の結果)
3	SN=47, SAVE、マシン状態	-ウォッチポイント不要
4	--フリーチェックポイント	--フリーチェックポイント
----	----	----
14	SN=28, FBGE、マシン状態	FBGE, FCC がレジスタタグを待っていた。
15	--フリーチェックポイント	--フリーチェックポイント

10

20

30

40

50

## 【 0 2 4 9 】

実行ユニットからの条件コードの捕捉と比較は下記のものを含む数種のファクタによって複雑になる。即ち、命令順序から外れる実行、各実行ユニット内の命令実行待ち行列、異なる命令型式の実行に要するマシンサイクルが変数であること、同一の命令型式でも実行資源には実行データを待つ必要に関する変動性である。上記のファクタとその他のファクタは一般に、命令の結果と状態が実際にデータ拡大バスにいつ現れるかに関して不確実性を生む。従って、本発明は命令が発行された時に同一のマシンサイクル中または任意の後のサイクル中にバスに現れる可能性のある条件コードデータを捕捉する構造及び方法を含む。また、本発明はリネームされたレジスタ用の条件コードを捕捉する構造及び方法を提供する。レジスタ依存性を除くため、例示したCPU51では条件コードのリネーミングを含めたレジスタのリネーミングが行われる。従って、ウォッチポイントユニット304はCCデータとタグを直接データ拡大バスから受信し、その後CCRFRN内でCCレジスタが更新されてから、FSR / CCRFRN606からCCデータとタグを受信する。

10

## 【 0 2 5 0 】

第27図ではDFB62に内に2個の固定小数点実行ユニット(FXUとFXAGU)と1個の浮動小数点実行ユニット(FPU)が示されているが、更に多数または少数を設置することができ、これらは任意の型であってよい。各実行ユニットはCCデータ - 拡大バスによってCCRFRN606とウォッチポイント304に接続される。CCデータ - 拡大バスは単位実行ユニット(FXU, FXAGU及びFPU)によって計算されたばかりのCC - データの結果を搬送する。CC - データはCCRFRNを更新するために使われる。ウォッチポイント304はCC - データ拡大 - バスをモニタまたは監視し、後のサイクルでCCRFRNからCC - データを貰うのを待つのではなく、バスから直接にCC - データをつかむ。このようにバスから直接にCC - データをつかみ取ることによって分岐評価が速くなる。

20

## 【 0 2 5 1 】

ウォッチポイント304は分岐誤予測を検出すると、PSU300内のPLSM307へ誤予測信号WP\_MISPRED\_VECを送る。PSU300は前述したようにCPU51の主要状態を追跡制御する。PSUがウォッチポイントから誤予測信号WP\_MISPRED\_VECを貰うと、PSU300内のPLSM307が誤予測と実行例外のうちで一番初期のものを選択し、本明細書の他の箇所に説明するように逆追跡手続きを開始する。

## 【 0 2 5 2 】

第25図はウォッチポイント304及びISBと、SFR / CCRFRNと、DFB, BRB内の実行ユニット(FXU, FXAGU及びFPU)と、PSU内のPLSMへのインターフェースのより詳細な機能ブロック図である。これらの主要な機能ブロックを次に説明する。

30

## 【 0 2 5 3 】

#### 1. ウォッチポイント要素の活動化

ウォッチポイント要素及び制御論理ユニット(WP - 要素)1002は、WP活動化 / 非活動化制御論理2012及び16の要素の各々がXCC, ICC, FCC, COND, JMPL / 分岐及び妥当性インジエータに関する情報を記憶している16のウォッチポイント要素セット2013を含んでいる。分岐 / JMPLが発行され予測された時点で、予測済み分岐、取消し分岐及びジャンプ - リンク命令についての条件コード情報を内含するISU200によってDO\_WTCHPT信号がアサートされる。「分岐を予測する」というのは、「分岐の方向(とられたか又はとられていないか)を予測する」ことを意味し、「JMPLを予測する」というのは、JMPLがつねにプログラムのフローを変更する(つねにとられる)ことから、「JMPL標的アドレスを予測する」ことを意味する。分岐 / JMPL命令を発行する場合、典型的CPU51は、チェックポイント番号により識別されたチェックポイントを作成する。ひとたび作成されると、チェックポイントはCPU51が1つのチェックポイント場所と次のチェックポイント場所の間に存在する命令のための全ての命令実行を完了してしまうまで、活動状態にとどまる。プロセッサは一度に多数の活動中のチェックポイントをもつことができる。チェックポイントの形成は発行された分岐 / JMPL命令に制限されず、例えば1つのトラップをとるときといったようなその他のケースで形成される可能性もある。

40

50

## 【 0 2 5 4 】

各々のウォッチポイント要素2012は、1つのチェックポイントエントリに対応する。典型的な CPU51は16のチェックポイントエントリを有し、従ってウォッチポイント304 は16のウォッチポイント要素をもつ。ウォッチポイント活動化 / 非活動化論理 (WADL) 2012は、ISU200からのDO\_CHKPN T 及びDO\_WTCHPNT信号に応えてウォッチポイント要素を活動化させることを担当している。DO\_CHKPN T は、チェックポイント形成を開始させるMAKE\_CHKPN T 信号及び、形成されるべきチェックポイント番号を識別するNEXT\_CHKPN T 信号を含んでいる。WADL2012は同様に、予測済み分岐 / JMPL命令が解決された時点でウォッチポイント要素を非活動化することをも担当している。ISU2000が分岐命令を発行すると、ISU200は、チェックポイントが形成されるべきであるということを表示するMAKE\_CHKPN T 信号をアサートし、1つのチェックポイント番号がPSU300によって割当てられNEXT\_CHKPN T により指定される。図50に示されているように、1つのウォッチポイント要素を活動化するためには、NEXT\_CHKPN T がデコーダ434 によって復号され、ANDゲート435 内でJSU200からのDO\_WTCHPT から誘導されたDO\_PREDICT信号との論理積がとられる。この出力はDO\_PREDICT\_VEC [i]と呼ばれる。信号DO\_PREDICT\_VEC [i]はWP\_VALID [i]をセットする。なおここで「[i]」は16のウォッチポイント番号のうちの1つを表示する。図50は同様に、ウォッチポイント出力論理2102の一部を含む、WP\_ACTIVE\_VEC 及びWP\_MISPRED\_VEC論理を結びつけられたウォッチポイントユニットの特定の実施形態についての付加的な構造的詳細をも示している。

10

## 【 0 2 5 5 】

ISU200は、分岐 / JMPL命令が発行された時点でウォッチポイントに対して複数の信号を送る。これらの信号は、XCCに左右される分岐として、ICCに左右される分岐として、FCCに左右される分岐として、ジャンプ - リンク命令 (JMPL) として、その命令を識別し、その分岐について条件コードがファイルされる (COND)。これらの信号はウォッチポイント304 により受信され、DO\_CHKPN T の成分信号としてNEXT\_CHKPN T によって指定されたウォッチポイント要素の中に記憶される。ウォッチポイント要素「i」がウォッチポイント活動化制御論理 (WACL) 2012によって割当てられると、これらの信号はそれぞれWP\_XCC [i], WP\_ICC[i], WP\_FCC[i], WP\_JMPL [i] 及びWP\_COND[i]レジスタの中に保たれる。各々のウォッチポイント要素は、それが非活動化され次にもう1つの分岐 / JMPL命令によって再度活動化されるまで、これらのレジスタ値を保持する。

20

30

## 【 0 2 5 6 】

典型的な CPU51は、各サイクル毎に複数のチェックポイントを作成することができ、又1サイクルにつき複数の分岐 / JMPL命令を発行することができる。従って、複数のウォッチポイント要素を活動化させることができる。例えば、CPU51が同じサイクル内で2つのチェックポイントを作成し2つの分岐を発行できる場合、DO\_WTCHPT, XCC, ICC, FCC, JMPL, COND, DO\_CHKPN T 信号が2セット又はそれと同等の情報内容が必要とされる。

## 【 0 2 5 7 】

条件コード書式及びフィールド定義及びJMPL (ジャンプ & リンク) 命令の書式及び定義は、本明細書中ですでに記録されている通り、SPARCV9アーキテクチャマニュアル内に提供されている。浮動小数点条件コードはビット - 0 及びビット - 1 を含む2ビット書式をもつ、整数条件コードは、XCCについてビット7 - 4 (それぞれN, Z, V, C) に、又ICCについてはビット3 - 0 (それぞれN, Z, V, C) として提供されている。分岐が発行され「とられた」ものと予測された時点で、COND値は分岐命令の条件フィールドデータと同じである。分岐が送信され「とられていない」と予測された時点で、CONDフィールドデータのビット - 3 が逆転され、残りのビットは変更されず、これらはCOND値となる。

40

## 【 0 2 5 8 】

図28は、CC - 捕捉論理1000, Eval (評価) 論理1001, WO - 要素記憶及び制御ユニット1002及び標的 ram1003を示す。ウォッチポイント304 の内部構造のより詳しい表示を示す。図29は、評価準備完了 (EVAL\_READY) 及び評価条件コード (EVAL\_CC) 論理を含むCC - セレクト論理1006、アレイ晚期一致論理1004、現行一致論理1005を含む、ウォッチポイント30

50

4 の異なる概略的部分を示す。

【 0 2 5 9 】

図30は、分岐予測及び評価のタイミング図を示す。ウォッチポイントは、各々予測された分岐/JMPL情報を保持し予測の正しさを評価することのできる複数のウォッチポイント要素を有する。サイクル1では、分岐が発行される。サイクル2では、複数のウォッチポイント要素の1つ(要素「i」と仮定する)が活動化される。その後、WP\_ACTIVE [i]がアサートされる。サイクル5では、FXUユニットは、その分岐を左右しウォッチポイント要素が待っているCC-データを復帰させている。サイクル5では、ウォッチポイント要素iはCCデータを捕捉する。サイクル6では、ウォッチポイント要素がCC-データを捕捉していることからWP\_ACTIVE\_VEC [i] はアサート解除され、EVAL\_READY [i]がアサートされる。EVAL\_READY [i]がCC評価をバリッドする。予測が正しい場合には、EVAL\_TRUE[i]がアサートされる。そうでない場合、EVAL\_READY [i]はアサートされない。この例では、タイミングダイアグラムは、予測が正しくないことをアサートする。次にWP\_MISPRED\_VECがアサートされ、PSU300内のPLSMへ送られる。PLSMは全てのウォッチポイント要素(典型的実施形態では16の要素)からのWP\_MISPRED\_VEC [i]信号及び実行ユニットからの実行トラップ信号の中から優先順位をとり、最も早期の事象を決定する。サイクル7では、PSU300は、CPU51を以前の状態までバックアップさせるDO\_BACKUP 信号をアサートする。DO\_BACKUP 信号は、CPUにバックアップを実施するよう知らせるMAKE\_BACKUP 及び、そこまでバックアップが作成されるべきチェックポイント番号を識別するBACKUP\_CHKPNNT 信号を含んでいる。図29は、複数のウォッチポイント要素のうちの一つに付随するウォッチポイントユニットの特定の実施形態についての付加的な構造的詳細を示している。

10

20

【 0 2 6 0 】

## 2. データ順方向送りバスから直接のCCデータのウォッチポイント捕捉

CC-捕捉(グラビング)論理1000は、データ順方向送りバスから条件コードデータ有効及び条件コードタグを受理しこれらを現行のマシサイクル中に ISBから受理した条件コードタグと一致させようとするCCタグ現行一致論理2001を含んでいる。典型的な論理回路が図31にCCタグ現行一致論理の一実施形態について示されている。

【 0 2 6 1 】

FSR / CCRFRN606 は、以下の信号をウォッチポイントに送る: CC\_RENAMED, CC\_TAG\_C, CC\_TAG\_RENAMED\_C, CC\_DV\_C及びCC\_DATA。CC\_RENAMEDは、CCが現行のマシサイクル内で発行された命令によって修正された場合にアサートされる。CC\_TAG\_Cは、リネーム済みCC物理タグを識別するものの、現行のマシサイクル内で発生するCC修正は除外される。CC\_TAG\_RENAMED\_Cも同様に、リネーム済みCC物理タグを意味するが、現行サイクル中に発生するCC修正は内含される。CC\_DV\_C は、CC\_DATA が、先行サイクルによって修正されたCCデータを有するものの現行サイクル内で行なわれたCC修正を除外する場合に、アサートされる。従って、CC\_VD\_C = 1である場合、先行サイクルによって行なわれた全てのCC修正はCC\_DATA 内に反映される。その上、CC\_DV\_C = 0である場合、CC\_DATA 値は最新のものではなく、FSR / CCRFRNは、最新のCCデータを捕捉する前に先行するサイクルによって発行された命令の完了を持っている。FSR / CCRFRN606 によってウォッチポイントに送られる信号は、表6に記されている。

30

40

【 0 2 6 2 】

## 【表 5】

表 6 FSR/CCRFRN606は以下の信号をウォッチポイントに送る

信号	内容説明
CC_RENAMED	現行マシンサイクル内で発行された命令によりCCが修正された時点で表明される。条件コードセレクト論理によって使用される。
CC_TAG_C	リネーム済みCC物理タグを識別するが、現行のマシンサイクル内で発生するCC修正を除外する。現行の一致について使用される。
CC_TAG_RENAMED_C	リネーム済みCC物理タグを識別するが現行サイクル内で発生するCC修正を内含する。アレイ内に書込まれ、アレイ一致のために使用される。
CC_DV_C	CC_DATA が先行サイクルによって修正されたCCデータを有する場合に表明されるが、現行サイクル内で行なわれたCC修正を除外する。条件コードセレクト論理により用いられる。
CC_DATA	CC_DV_C = 1である場合、先行サイクルによって行なわれた全てのCC修正は、CC_DATA [7 : 0] 内に反映される。 CC_DV_C = 0である場合、CC_DATA 値は最新のものではなく、FSR/CCRFRNは、最新CCデータを捕捉する前に先行サイクルによって発行された命令の完了を待っている。条件コードセレクト論理によって使用される。

10

20

30

## 【 0 2 6 3 】

図31では、比較回路2006, 2008及び2010は各々、リネーム済みCC物理タグを識別するCC\_TAG\_C信号を受理するが、現行のマシンサイクル中に発生するCC修正は除外する。比較回路の各々は同様に、データ順方向送りバスから直接条件コードタグを受理する。すなわちFPUからFPU\_FCC\_TAG\_Fを、FXUからFXU\_XICC\_TAG\_FをそしてFXAGUからFXAGU\_XICC\_TAG\_Fを受理し、ここでFXAGU\_XICC\_TAG\_F及びFXU\_XICC\_TAG\_Fは両方共整数条件コードのXCC及びICCの両方の部分を内含する。比較の出力はANDゲート2007, 2009及び2011に送られ、制御信号として役立つデータ有効信号(FPU\_FCC\_DV\_F, FXU\_XICC\_DV\_F又はFXAGU\_XICC\_DV\_F)との論理積がとられる。データ有効信号がアサートされたならば、そのとき、比較回路によって決定された一致は有効であり、CC現行一致信号(FPU\_CURR\_MATCH, FXU\_CURR\_MATCH 又は FXAGU\_CURR\_MATCH)がアサートされる。

40

## 【 0 2 6 4 】

図31に示されているように、CCリネーム済みタグアレイ一致論理2002は、各々の実行ユニット(FPU, FXU, FXAGU)についてAND論理機能を実行するための論理回路2016及び比較回路2015 16セットから成る1つのセットを含んでいる。現行一致論理2001はデータ順方向送りバスからの信号をCC\_TAG\_Cと比較する一方、アレイ一致論理ユニット2002は、リネーム済みCC物理タグを識別するものの現行サイクル中に発生するCC修正を含むCC\_TAG\_RENAMED\_Cとデータ順方向送りバスからの信号の各々を同時に比較する。CC\_TAG\_RENAMED\_Cは、DO\_PREDICT\_VECと現行マシンサイクル内で発行された命令によりCCが修正された時点で

50

アサートされているCC\_RENAMEDとの論理積をとることにより、生成された書込みバリッド信号の制御下で、16のCCタグレイ2003記憶要素のうちの中へ書き込まれる。サイクル毎に、CC\_TAG\_RENAMED\_Cの記憶された値は、何らかの一致を識別する目的で、FPUからのFPU\_FCC\_TAG\_F、FXUからのFXU\_XICC\_TAG及びFXAGUからのFXAGU\_XICC\_TAG\_Fと比較される。比較回路2016の出力は、制御信号として役立つデータ有効信号(FPU\_FCC\_DV\_F、FXU\_XICC\_DV\_F、又はFXAGU\_XICC\_DV\_F)と論理積がとられる。データ有効信号がアサートされると、そのとき比較回路によって決定された一致は有効であり、CCアレイ一致信号(FPU\_ARRAY\_MATCH [i]、FXU\_ARRAY\_MATCH [i]及びFXAGU\_ARRAY\_MATCH [i])がアサートされる。従って、CC - 捕捉論理は、複数の未解決分岐評価を同時に監視することができる。

10

## 【0265】

図31は、CC捕捉論理を示す。CC\_TAG\_ARRAYは16のレジスタを含み、各レジスタは16のチェックポイントのうちの一つに対応し、CC\_TAG\_Cビット数と同じラッチ数をもつ(ここで我々は4ビットラッチを仮定している。1つの分岐を発行し予測する場合、CC\_TAG\_RENAMED\_Cは、DO\_WTCHPTから誘導されたDO\_PREDICT\_VEC [i]により指定されるCC\_TAG\_ARRAYの場所内に書き込まれる。CC\_RENAMEDは、AND論理ゲートといった書込み有効化及びアドレスセクタ論理2004内で書込み有効化信号として役立つ。

## 【0266】

この記述においては、プロセッサは2つの固定小数点実行ユニットすなわちFXU及びFXAGUそして1つの浮動小数点実行ユニットFPUをもつものとして仮定されている。典型的な実施形態においては、FXUのみがJMPL命令を実行することが仮定されている。各々のユニットはCC - データ順方向送りバスを有する。FXU CCデータ順方向送りバスは、FXU\_CC\_TAG\_VALID、FXU\_CC\_TAG及びFXU\_CC\_DATAを含む。FXUが、CCを修正するようなその実行を終わろうとしているとき、FXU\_CC\_TAG\_VALIDがアサートされ、FXU\_CC\_TAGはそのリネームされたCC物理タグを有し、FXU\_CC\_DATAはCCの結果データを有する。同じことはFXAGU及びFPUにもあてはまる。FSR/CCRFRNユニット内の条件コード(CC)データは次のサイクル内でこれらのCCデータ順方向送りバスを用いて更新される。ウォッチポイントは同様に、ウォッチポイント要素がCCデータを待機している場合にデータ順方向送りバスからのCCデータを捕捉する。CC\_TAG\_CはFXU\_CC\_TAGと比較され、その一致信号はFXU\_CC\_TAG\_VALIDとの論理和がとられ、出力信号はFXU\_CC\_CURR\_MATCHと呼ばれる。同様にCC\_TAG\_ARRAY内のレジスタの内容は、FXU\_CC\_TAGと比較され、その一致信号は、FXU\_CC\_TAG\_VALIDとの論理積がとられ、出力信号はFXU\_CC\_ARRAY\_MATCH [i]と呼ばれる。FXAGU及びFPUについては、FXUと同じ一致機能が存在する。ここで3つの命令発行シーケンス例を基準にして、捕捉論理のオペレーションについて記述する。

20

30

## 【0267】

CC捕捉論理1000は同様に、そのウォッチポイント要素内の分岐を左右し、アサートされた時点でCCデータ信号(例えばCC\_DATA [7:4])を選択し同様に次のサイクル内での評価のためにそのデータ信号をラッチングする条件コードに基づいて信号を生成することをも担当するCCセレクト論理2005(各ウォッチポイントに1つずつの16層)をも含んでいる。条件コードセレクト論理2005の1実施形態のための典型的論理回路は、図32の中に提供されている。このCCセレクト論理2005のオペレーションについては、以下の3つの例に関連して記述されている。

40

## 【0268】

評価論理1001は、図25に示されている通り、評価準備完了論理(EVAL\_READY) 2100、評価真論理(EVAL\_TRUE) 2101及びウォッチポイント出力論理(WP出力論理) 2102を内含する。図33は、CCセレクト論理2005からの1組のセレクト信号(例えば、SEL\_BR\_XCC [i])及びAND論理ゲート2106の出力からの有効化信号(EVAL\_ENABLE)を受理するEVAL\_READY論理2100に付随する付加的な構造的詳細を示している。EVAL\_ENABLEは、WP\_ACTIVE\_VEC [i]、WP\_JMPL、FXAGU\_JMPL及びFXAGU\_CHKPTに基づいてアサートされる。EVAL\_READY論理2100は、評価が準備完了していることを表わすi番目のウォッチポイントのためのEVAL\_R

50



EADY [i]信号を出力する。

【 0 2 6 9 】

図34は、CC\_EVAL 論理2105を含むEVAL\_TRUE 論理2101に付随する付加的な論理的詳細を示す。EVAL\_TRUE 論理は、分岐命名又はJMPL命令が適正に予測されたか否かを決定することを担当する。CC\_EVAL2105 は、CCセレクト論理2005からのEVAL\_CC[i]及び、 ISBから受理されウォッチポイント要素2013の中に記憶されたWP\_COND[i], WP\_XCC [i], WP\_ICC[i]及びWP\_FCC [i]を受理し、SPARC-V9マニュアル内に記述された規則を用いた比較に基づいて2つの信号を評価する。これらが一致する場合、分岐真信号(BR\_TRUE[i])がその i 番目のウォッチポイントについて生成される。EVAL\_TRUE 論理2101は、同様にJMPL一致論理2201によるJMPL\_MATCH出力 (図35参照) を、 ISBから受理されウォッチポイント要素2013内に記憶されたWP\_JMPL[i]と比較する。JMPL\_MATCHがWP\_JMPL[i]と一致する場合、JMPLは適正に予測されており、JMPL\_TRUE[i]がアサートされる。BR\_TRUE[i]とJMPL\_TRUE[i]の論理和がとられ、EVAL\_TRUE[i]信号を生成する。単一の命令のみについて1つのウォッチポイントが形成され、従って、BR\_TRUE[i]又はJMPL\_TRUE[i]の1つのみアサートされることになる。

10

【 0 2 7 0 】

WP出力論理2102が図50に示されている。PSU300からのEVAL\_READY [i], EVAL\_TRUE [j], WP\_VALID [i]及びKILL\_VEC [i]は、WP出力論理に入力され、これがWP\_ACTIVE\_VEC [i]及びWP\_MISPRED\_VEC [i]信号を出力する。

【 0 2 7 1 】

図35は、TARGET\_RAM及びJMPL評価論理に付随する付加的な構造的詳細を示している。JMPL評価論理は、rd = 0とした復帰タイプのJMPL命令を評価した。ANPC出力論理2103は、正しい計算値である FXU\_DATA の値を、本書で記述されている優先順位決定スキーマ及び記憶されたWP\_ANPC の比較に基づいて PSUPLSMによって誤予測(MISPRED) がアサートされているか否かに基づき BRBフェッチユニットまで送るため、RD\_ANPC を選択する。

20

【 0 2 7 2 】

3 . ウォッチポイントオペレーション例

a . ウォッチポイントオペレーション - 例 1 :

表7は、サイクルX, X + 1、及びX + 2の中で実行された、例1のための命名のリストである。サイクルXでは、4つの命令が発行され、そのうちの1つ(addcc)がCCを修正する。このとき、リネームされた - CC物理タグ # 7が割当てられる。サイクルX + 1では、ここでも4つの命令が発行され、そのうちのいずれもCCを修正せず、addccが実行されてサイクルX + 1内でCCデータを復帰させる。サイクルX + 2では addccによって修正されたCCデータはFSR/CCRFRN内にある。サイクルX + 2では4つの命令が発行され、そのうちのいずれも分岐前にCCを修正せず、その後、分岐命令 (br, xcc)が発行される。この場合、サイクルX + 2では、現行サイクル内でCCを修正する命令が全く発行されないことからCC\_RENAMED = 0である。その上、最も晩期のCCタグが # 7でありこれがサイクルXで割当てられることから、CC\_TAG\_C = CC\_TAG\_RENAMED\_C = 7である。最後に、最も晩期のCCがFSR/CCRFRN内にあることから、CC\_DV\_C = 1である。

30

【 0 2 7 3 】

1つの分岐を発行しているとき、FSR/CCRFRN内で分岐を左右するCC\_DATA が有効である。従って、CC\_DV\_C は分岐の発行と同じサイクルでアサートされる。分岐が XCCにより左右されると仮定してみよう。そのとき SEL\_BR\_XCC (i) がアサートされ (図32参照)、信号はCC\_TDATA [7 : 4]を選択し、データは次のサイクルにおいてEVAL\_CC[i]内でラッチされる (信号CC\_DATA [7 : 4]は XCC\_DATA\_C 及び ICC\_DATA\_C であるか又は FCC\_DATA\_C である)。又EVAL\_READY [i]は次のサイクルにおいてアサートされる (図33参照)。CC評価は、準備完了態となる。

40

【 0 2 7 4 】

b . ウォッチポイントユニットオペレーション - 例 2 :

表8は、サイクルX及びX + 1内で実行された例2についての命令リストである。サイ

50

クルXでは、4つの命令が発行され、そのうちの1つ(addcc)がCCを修正する。その後、リネーム済みCC物理タグ#7が割当てられる。サイクルX+1では、ここでも4つの命令が発行され、そのうちのいずれも分岐前にCCを修正せず、そのうちの1つが分岐命令である。この場合、サイクルX+1では、現行サイクルでCCを修正するような命令は全く発行されていないため、CC\_RENAMED = 0である。又、最も晩期のCCタグが#7であり、それが先行サイクルで割当てられているため、CC\_TAG\_C = CC\_TAG\_RENAMED\_C = 7である。ただし、addccによって修正される最も晩期のCCデータはFSR/CCRFRNにおいてCCレジスタ内に書込まれないことからCC\_DV\_C = 0である。

【0275】

(サイクルX+1において)1つの分岐を発行する場合、サイクルXで発行されている命令「addcc」はCCを修正し結果データはFSR/CCRFRNにおいてCCレジスタ内に書込まれないことから、FSR/CCRFRN内でその分岐を左右するCC\_DATAは有効でない(CC\_VD\_C = 0)。我々はFXU実行ユニットによりサイクルX+1で「addcc」が実行されつつあり、FXUがサイクルX+1でそのデータ順方向送り母線上でCCデータを復帰させている、と仮定している。その後、サイクルX+1でFXU\_CC\_CURR\_MATCH(図31参照)がアサートされる。サイクルX+2でSEL\_FXU\_XCC[i](図32参照)がアサートされ(図32参照)、信号はFXU\_CC\_DATAを選択し、データはEVAL\_CC[i]内でラッチされる。又、サイクルX+2でEVAL\_READY[i]がアサートされる(図33を参照)。次にCC評価が準備完了状態となる。

【0276】

FXU\_CC\_CURR\_MATCH比較のためにはCC\_TAG\_RENAMED\_CではなくCC\_TAG\_Cが使用されることに留意されたい。分岐が発行されるのと同じサイクル内で、CCを修正するような命令が発行される場合、CC\_TAG\_Cは最も晩期のCC物理タグではない。しかし、CC\_RENAMEDがアサートされ信号はFXU\_CC\_CURR\_MATCHを抑制する(図32参照)。このことはすなわち、分岐が発行されるのと同じサイクルにてCCを修正する命令が全く無い場合には、FXU\_CC\_CURR\_MATCHを使用することができる、ということの意味している。FXU\_CC\_TAG\_CURR\_MATCHのためにCC\_TAG\_C[3:0]を用いることは、CC\_TAG\_CがCC\_TAG\_RENAMED\_Cよりも早い信号であることから、経路遅延を減少させる一助となる。

【0277】

c. ウォッチポイントユニットオペレーション - 例3 :

表9は、サイクルX及びX+1で実行される例3についての命令リストである。サイクルXでは、4つの命令が発行され、(addcc)の1つがCCを修正する。このときリネーム済み-CC物理タグ#7が割当てられる。サイクルX+1では、ここでも4つの命令が発行され、そのうちの1つ(subcc)が分岐前にCCを修正し、それらのうちの1つは分岐命令である。リネームされたCC物理タグ#8は命令subccに割当てられる。この場合、サイクルX+1では、現行サイクル内でCCを修正する命令が発行されているため、CC\_RENAMED = 1である。先行サイクルによる最も晩期のCCタグが#7であることからCC\_TAG\_C = 7であり、現行サイクルでCCを修正する命令が発行されておりリネームされたCC物理タグ#8が割当てられることから、CC\_TAG\_RENAMED\_C = 8である。同様に、subccによって修正されている最も晩期のCCデータがFSR/CCRFRN内でCCレジスタ内に書込まれていることから、CC\_DV\_C = 0である。

【0278】

(サイクルX+1において)1つの分岐を発行する場合、サイクルXで発行されている命令「addcc」はCCを修正し結果データはFSR/CCRFRNにおいてCCレジスタ内に書込まれないことから、FSR/CCRFRN内でその分岐を左右するCC\_DATAは有効でない(CC\_DV\_C = 0)。その上、命令「subcc」はサイクルX+1で発行され、その後CC\_RENAMEDがアサートされる。我々は、FXU実行ユニットによりサイクルX+2で「subcc」が実行されつつあり、FXUがサイクルX+2でそのデータ順方向送り母線上でCCデータを復帰させている、と仮定している。その後、サイクルX+2でFXU\_CC\_ARRAY\_MATCH[i](図31参照)がアサートされる。サイクルX+3でSEL\_FXU\_XCC[i](図32参照)がアサートされ(図32参照)、信号はFXU\_CC\_DATAを選択し、データはEVAL\_CC[i]内でラッチされる。又サイクルX

10

20

30

40

50

+ 3 で EVAL\_READY [i] がアサートされる (図33を参照)。次にCC評価が準備完了状態となる。この例では16のウォッチポイント要素が存在し、各々の要素は、いずれかのサイクルにおいて各分岐評価について並行なCCを捕捉することができる。

【 0 2 7 9 】

【表 6】

表 7 例 1 についてのFSR/CCRFRNからの命令及び信号

サイクル	命令	FSR/CCRFRN信号
X	move mul addcc <#7 div	
X+1	move move move (#7 実行済み) move	
X+2	mul move move br, xcc	CC_RENAMED=0 CC_TAG_C=7 CC_TAG_RENAMED_C=7 CC_DV_C=1

10

20

【 0 2 8 0 】

【表 7】

表 8 例 2 についてのFSR/CCRFRNからの命令及び信号

サイクル	命令	FSR/CCRFRN信号
X	move mul addcc <#7 div	
X+1	mul (#7 実行済み) move move br, xcc	CC_RENAMED=0 CC_TAG_C=7 CC_TAG_RENAMED_C=7 CC_DV_C=0

30

40

【 0 2 8 1 】

## 【表 8】

表9 例3についてのFSR/CCRFRNからの命令及び信号

サイクル	命令	FSR/CCRFRN信号
X	move mul addcc <#7 div	
X+1	mul move subcc <#8 br, xcc	CC_RENAMED=1 CC_TAG_C=7 CC_TAG_RENAMED_C=8 CC_DV_C=0

10

## 【0282】

## 4. 分岐命令の評価

分岐が発行されてから1サイクル後に、WP\_VALID [i]がアサートされる。WP\_VALID [i] = 1でありEVAL\_READY [i] = 0そしてKILL\_VEC [i] = 0である場合、WP\_ACTIVE\_VEC [i]がアサートされる(図29参照)。「WP\_ACTIVE[i] = 1」は、チェックポイントiを用いる分岐がすでに発行されているもののまだ解決されていないことを意味する。値が0となったならば、それは分岐が解決された(完了した)ことを意味する。WP\_ACTIVE\_VECはICRU 301に送られ、どの分岐/JMPL命令が完了されるかを決定するために用いられる。

20

## 【0283】

分岐評価が準備完了状態になった時点で、EVAL\_READY [i]がアサートされEVAL\_CC[i]は、捕捉されたCC値である。このとき、EVAL\_CC[i]及びWP\_COND[i]はEVAL\_TRUE LOGIC2101内でCC\_EVAL論理中へフィードされ(図34参照)、分岐は評価される。分岐を発行するときに作成された予測が正しい場合、BR\_TRUE[i]がアサートされる。そうでない場合、これはアサートされない。1つの分岐を評価するとき、分岐タイプを知るために、WP\_XCC [i], WP\_ICC[i], WP\_FCC[i]信号も同様に使用される。BR\_TRUE[i]はJMPL\_TRUE[i](以下で説明する)と論理和がとられ、EVAL\_TRUE[i]が生成される。WP\_VALID [i] = 1でEVAL\_READY [i] = 1でEVAL\_TRUE[i] = 0でKILL\_VEC [i] = 0である場合、WP\_MISPRED\_VEC [i]はアサートされる(図29参照)。「WP\_MISPRED\_VEC [i] = 1」というのは、以前に作成された分岐予測がまちがっていることを意味する。PSUはこの信号及び実行エラー信号を受理し、最も早期の誤予測又は実行エラー条件を選択し、次にBACKUP信号をアサートして以前の状態までバックアップする。

30

## 【0284】

PSUは、バックアップが発生しバックアップ時点の後にとられたチェックポイントをキルする必要がある場合に、KILL\_VEC [i]信号をアサートする。KILL\_VEC [i]がアサートされた時点で、反応するウォッチポイント要素も同様にキルされる。「キルされる」というのは、WP\_ACTIVE\_VEC [i]及びWP\_MISPRED\_VEC [i]が抑制されることを意味する。WP\_ACTIVE\_VEC及びWP\_MISPRED\_VECは、WP\_MISPRED\_VEC (0 - 16)を同じベクトル内でコード化できるように、マルチホットであり得る。このことはすなわち、一度に最高16のウォッチポイント要素(この例では)が活動中であり得又一度に誤予測を検出できるということを意味している。

40

## 【0285】

分岐を発行し予測するとき、FETCHがWR\_ANPCを通してウォッチポイントに、分岐標的アドレス又は分岐の次のアドレスを送る。予測が「とられた」場合、WR\_ANPCが、分岐の次のアドレス(とられていないアドレス)である。予測が「とられていない」場合、WR\_A

50

NPC は分岐標的アドレス（とられたアドレス）である。予測が誤っていることが判明した場合、ウォッチポイントは、適正な分岐経路から命令を再度フェッチするためRD\_ANPC を介してアドレスを FETCHまで送り戻す。WR\_ANPC 値はTARGET\_RAM内に記憶される（図35参照）。TARGET\_RAMは16のエン트리をもち、各々のエントリは各々のチェックポイントに対応し、1つのWR\_ANPC アドレスを記憶することができる。この RAMの書込み有効化信号及び書込みアドレスは、MAKE\_CHKPNNT 及びNEXT\_CHKPNNT である。MAKE\_CHKPNNT は、チェックポイントを作成するときアサートされる。BACKUP信号がアサートされた時点で、TARGET\_RAMの読取りアドレスのためにBACKUP\_CHKPNNT が用いられる。BACKUP\_CHKPNNT は、CPU51がそれまでバックアップするチェックポイントを指定する。又TARGET\_RAMの読取りデータがRD\_ANPC を介して FETCHまで送られる。

10

【0286】

#### 5. JMPL-LINK(JMPL) 命令の評価

JMPL命令を発行し予測するとき、FETCHは、ウォッチポイントに対しWR\_ANPC を介して予測されたJMPL標的アドレスを送る。ウォッチポイントはアドレスを保ち、実行ユニットにより計算上の正しいアドレスとこのアドレスを比較することによって適正さを評価する。予測が誤っていたことが判明した場合、ウォッチポイントは、正しいアドレスから命令を再度フェッチするためRD\_ANPC を介して FETCHに計算上の正しいアドレスを送る。予測されたJMPL標的アドレスは分岐の場合と同じ方法でTARGET\_RAM内に記憶される。

【0287】

我々は、FXAGU実行ユニットのみがJMPL標的アドレスを計算するものと想定している。FXAGUがJMPL命令の実行を終えた時点で、FXAGU\_JMPL がアサートされ、FXAGU\_CHKPNNT はJMPLのチェックポイント番号を指定し、FXAGU\_DATA は、計算上の正しいJMPL標的アドレスを内含する。このとき、FXAGU\_CHKPNNT によって指標付けされたTARGET\_RAM内の予測されたJMPLアドレスが読みとられ、読取られたデータはFXAGU\_DATA と比較されることになる。比較結果はラッチされ、ラッチ済み信号はJMPL\_MATCHである（図35）。予測上のアドレスと計算上のアドレスが同じである場合、JMPL\_MATCHがアサートされる。このとき、JMPL\_MATCHはWP\_JMPL[i]との論理値がとられ、出力はEVAL\_TRUE[i]となる（図34）。FXAGU\_JMPL がアサートされてから1サイクル後に、EVAL\_READY [i]もアサートされる（図33）。

20

【0288】

分岐命令の場合と同じ要領で、JMPLについてWP\_ACTIVE\_VEC [i]及びWP\_MISPRED\_VEC [i]が生成される。WP\_MISPRED\_VEC [i]がアサートされてから1サイクル後に、PSU300は、ウォッチポイント304に対する2つの信号すなわちBACKUP\_CHKPNNT 及びMISPRED（ただしバックアップがJMPLの誤予測のせいである場合のみと同数のBACKUPを送る。MISPRED がアサートされた場合、ウォッチポイントは、FETCHに対し正しいJMPLアドレスを含むラッチされたFXAGU\_DATA 信号を2回送る（図35参照）。TARGET\_RAMは次の2つの目的のために使用される；その1つは分岐の代替的アドレスを記憶するためそしてもう1つは予測されたJMPL標的アドレスを記憶するためである。これら2つの用途は排他的であり、従って、この実施のためには1つのTARGET\_RAMだけで充分である。

30

【0289】

発明力ある構造及び方法は、（1）複数の予測済み分岐又はジャンプ&リンク命令のウォッチング（監視）を同時に開始するための構造及び方法；（2）実行ユニットからのデータ順方向送り母線を同時にウォッチングすることにより、予測済み分岐又はジャンプ&リンク命令が待っている複数の条件コードデータ又は計算上のジャンプ&リンクアドレスを捕捉するための構造及び方法；（3）単独又は（2）と組合わせた形で、分岐又はジャンプ&リンク命令の複数の誤予測信号を同時に生成できるようになるための構造及び方法；（4）単独又は（2）と組合わせた形で、上記2つのケースで共有される1つの記憶装置内に代替分岐アドレス又は予測されたジャンプ&リンクアドレスを記憶するための構造及び方法；（5）単独又は（4）と組合わせた形で、誤予測が発生した場合に命令フェッチのため正しい分岐アドレス又はジャンプ&リンクアドレスを送るための構造及び方法；

40

50

そして(6)CCデータを捕捉するためのタグ比較を2つの部分すなわちデータ順方向送り母線タグと比較した先行サイクル内のCCタグ及びデータ順方向送り母線タグと比較した現行発行ウィンドウ内の現行リネーム済みCCタグという2つの部分に分割することによって、クリティカルパスをスピードアップするための構造及び方法(単独又は(2)と組合わせた形)、を提供する。後者の場合、リネーム済みCCタグのラッチされた信号を比較のために用いることができ、従ってこれはタイミングクリティカルではない。図36は、複数の同時未解決分岐評価のための発明力あるウォッチポイント方法の一実施形態の概略的流れ図である。

【0290】

## B. 例外検出

例外は、発行又は実行中に発生する可能性がある。かくして例外には発行トラップと実行トラップが含まれる。以下では、発行トラップと実行トラップの検出について記述する。

【0291】

### 1. 発行トラップの検出

ISU200は同様に、命令の発行を実施する発行トラップを検出する。これらの発行トラップは、同期化又は非同期化発行トラップであり得る。当業者であれば、同期化及び非同期化とされる発行トラップのタイプは、さまざまな基準に基づいて選択され得、設計者の選択の対象となりうるということを認識することだろう。

【0292】

例えば、非同期化発行トラップは、標準的には、往々にして発生し、かつ前述の要領でのCPU51のそれに対する同期化がプロセッサの速度を低下させることになるような発行トラップである。従って、このような種類のトラップは、機械的に同期化されエントリを受けるとはならない。これらの発行トラップとしては、SPARC-V9アーキテクチャマニュアル中に記述されているfill\_normal, fill\_other, spill\_normal, spill\_other, clean, window, trap\_instruction( つねに (TA%go tsimm7) 命令の中間にあるトラップ)、unimplemented\_ldd, unimplemented\_std 及び illegal\_instructionが含まれていると考えられる。

【0293】

非同期化発行トラップは、標準的に、往々にして発生せず、かつそれについてCPU51を同期化することによってプロセッサの速度が著しく影響を受けないような発行トラップである。これには、SPARC-V9 instruction\_access\_error, instruction\_access\_exception, privileged\_opcode, privileged\_action, fp\_exception\_other, fp\_disabled, emulation\_trap、及びtrap\_instruction(条件付きトラップ(TCC)命令)発行トラップといったような発行トラップが含まれると考えられる。

【0294】

これらの発行トラップの或るタイプのものが発行段で発生したか否かを決定するために、ISU200は制御レジスタ(CNTR-REG)フィールドを制御レジスタファイル800から受理する。図17に示されているように、制御レジスタファイル800は、クリーンウィンドウ(CLEANWIN)レジスタ801、回復可能ウィンドウ(CANRESTORE)レジスタ802、セーブ可能ウィンドウ(CANSAVE)レジスタ803、ウィンドウ状態(WSTATE)レジスタ804及びプロセッサ状態(PSTATE)レジスタ805を含むSPARC-V9特権的レジスタを収納している。これは又、TICKレジスタ806及び浮動小数点レジスタ状況(FPRS)レジスタ807を含むSPARC-V9非特権的レジスタをも収納している。CNTRL\_REGフィールドは、SPARC-V9アーキテクチャマニュアル内に記述されCLEANWIN, CANRESTORE, CANSAVE, WSTATE, TIC、及びFPRSレジスタ801~807により提供されているCLEANWIN, CANRESTORE, CANSAVE, WSTATE\_NORMAL, WSTATE\_OTHER, PSTATE\_PEF, PSTATE\_PRIV, TICK\_NPT、及びFPRS\_FEFフィールドを含んでいる。

【0295】

ここで図7に戻ると、ISU200は、FR\_INSTs\_BRPs 命令を復号し、SPARC-V9発行トラップ

10

20

30

40

50

のいずれかが発生したか否かを決定するべくSPARC-Vアーキテクチャマニュアルに従って CNTRL\_REGフィールドを利用する。最も早期の発行ウィンドウスロット内で命令により引き起こされた発行トラップのみが取られることになる。従って、単数又は複数の発行トラップが検出された時点で、最も早期のスロット内にある発行トラップ誘発命令のスロットに先立つ発行ウィンドウスロット内の一部の命令のみがISU200によって発行され得る。発行トラップ誘発命令及び発行トラップ誘発命令のスロットの後の一部のFR\_INST\_BRP\_0-3命令は、(trap\_instruction)発行トラップをひき起こしたTA又はTcc命令が発行されるということを除いて、発行されない。

【0296】

発行トラップが起こった時点で、ISU200は、トラップが発生したことを表わすISSUE\_TRAP信号をPSU300に対して出力し、前述の発行トラップのうちのいずれが発生したかを識別する。ISU200により非同期化発行トラップが検出された場合、その検出時点で直ちにISU200によりISSUE\_TRAP信号が出力されることになる。しかしながら、同期化発行トラップがISU200により検出された場合、発行トラップ誘発命令がスロット0内にくるまでISU200によってISSUE\_TRAP信号が出力されることはなく、CPU51は前述の要領で同期化される。非同期化発行トラップをひき起こすTcc命令の場合、Tcc命令は、CPU51が同期化された時点で初めて発行され得る。

【0297】

## 2. 実行トラップの検出

図8を参照すると、発行済み命令の実行中に、FPU600、FXU601及びLSU603によって検出されるエラーが発生する可能性がある。前述した通り、FPU600、FXU601及びLSU603は、命令の実行中にエラーが発生したか否かを表示するERR\_STAT信号をISB61のPSU200に出力する。LSU及びFXUからの実行エラーは、PSU300により容易にとり扱うことができる。しかし、次に記述するように、浮動小数点例外は特殊な取り扱いを必要とする。

【0298】

図8を参照すると、前述の通り、FPU600は、資源利用可能性に基づいて予測上の及び/又は実際のPC順序外で浮動点命令をout-of-order実行する。浮動小数点命令の実行中、FPU600は実行トラップ(すなわちエラー)を検出することができる。しかしながら、FPU600は予測上の及び/又は実際のPC順序外で命令をout-of-order実行し得ることから、それらがひき起こす実行トラップも又予測上及び/又は実際のPC順序外であり得る。浮動小数点実行トラップをあたかもそれらが実際のPC順に発生したかのごとく適切に取扱い検出するために、PSU300は、図37に示されているように浮動小数点例外(FPEXCEP)リング又はユニット306を含んでいる。

【0299】

図38に示されている通り、FPEXCEPリング306は64の記憶要素又はエントリを伴うデータ記憶構造900を含んでいる。各々の記憶エントリは64の命令通し番号のうちの1つに対応し、命令識別(FP)フィールド、浮動小数点トラップタイプ(FTT)フィールド及び現行浮動小数点例外タイプ(CEXC)フィールドを有する。図37に示されている通り、データ記憶構造はFFFPフィールドを記憶するための64のアドレス可能な記憶要素をもつFPフィールドリング、FTTフィールドを記憶するための64のアドレス可能な記憶要素を有するFTTフィールドリングそしてCEXCフィールドを記憶するための64のアドレス可能な記憶要素を有するCEXCフィールドリングを形成する。FP、FTT及びCEXCフィールドリングのアドレス可能な記憶要素の各々は命令通し番号に対応する。

【0300】

FPビットは、対応する命令がSPARC-V9浮動小数点タイプの命令であるか否かを表示するべくセット(「1」)又はクリア(「0」)される。FPEXCEPリング306内の浮動小数点命令としてFP-ビットにより識別された命令については、FTTフィールドは、発生したSPARC-V9浮動小数点トラップのタイプを識別する。SPARC-V9アーキテクチャマニュアル内に表示されている通り、これにはいかなるトラップも含まれず、IEEE\_754例外、未実施\_\_FPOP、未終了\_\_FPOP、シーケンス-エラー、ハードウェア-エラー、及び無効\_\_fp\_\_レ

10

20

30

40

50

ジスタエラーが含まれている。当業者であればわかるように、CPU51はこれらのトラップタイプのいくつかを特定、検出及び取扱いすることなく実施でき、その場合、FPEXCEPリング306のFTTフィールドは、所要記憶空間が少なくすむように、FPEXCEPリング306のFTTフィールドをより小さなものにすることができる。その上、IEEE\_754\_例外が発生したことをFTTフィールドが表示している浮動小数点命令については、CEXCフィールドは、IEEE規準754-1985及びSPARC-V9アーキテクチャマニュアルに従ってまさに発生したIEEE\_754\_例外の単数又は複数のタイプを識別する。かくして、CEXCフィールドは、1つのオペランドが不適正であるか否かを示すためのビット(nvc)、オーバーフローが発生したか否かを表わすためのビット(ofc)、アンダーフローが発生したか否かを表わすためのビット(ufc)、ゼロ除算が発生したか否かを表わすためのビット(dzc)、及び不正確な結果が発生したか否かを表わすためのビット(nxc)を含んでいる。しかしながら、浮動小数点トラップタイプのその他のタイプ又は浮動小数点トラップ無しのタイプを結果としてもたらず浮動小数点命令については、CEXCフィールドは、IEEE\_754\_例外のいずれのタイプも発生しなかったことを表示する。

10

#### 【0301】

前述し図11にも示されているA-リング312及びM-リング324と同様に、FPEXCEPリング306は、図37に示されているように「モジュールFPEXCEP-リングレジスタ長」算術演算(ここではモジュール64算術演算)を用いてデータ構造に沿ってポインタが移動させられる循環リング又はラップアラウンドデータ構造として実施される。当業者であれば、循環データ構造が有利であるものの、これは、本発明にとって不可欠なことではなく、本発明の特徴を実施するその他のデータ構造を用いることも可能であるということがわかるだろう。

20

#### 【0302】

ここで再び図38を参照すると、各々の発行段の間、ISU200は、フェッチされたFR\_INST、BRP\_0-3命令のどれが発行されたかを示す前述のISSUE\_VALID信号を出力する。同時にこれは、発行された命令のうちのどれが浮動小数点命令であることを識別するEP\_ISSUE信号を出力する。これらの信号は、FP書込み(WR)論理901及びFTT書込み(WR)論理902により受理される。さらに、FPWR論理901及びFTTWR論理902はICRU301から、前述のNISNポインタを受理する。

#### 【0303】

FPWR論理901及びFTTWR902はNISNポインタ及びISSUED\_VALID信号を用いて、ISU200により発行されたばかりの命令のためのFTTフィールド及びFRビットに対応するFPEXCEPリング内の場所をアドレスする。浮動少数点命令であるものとしてFP\_ISSUEによって識別された命令の各々について、FRWR論理901は、それが浮動小数点命令であることを表わすために対応するFPビットをセットする。又、浮動小数点命令でないものとしてFP\_ISSUEによって識別された発行済み命令については、FPWR論理901は、それが浮動小数点命令でないことを表示するために対応するFPビットをクリアする。

30

#### 【0304】

図8を参照すると、前述の通り、FPU600、FXU601、FXAGU602及びLSU603は発行済み命令を実行し、実行が完了した時点でERR\_STAT信号を出力する。図39に示されているように、ERR\_STAT信号は、FPU600によって実行され完了した浮動小数点命令についての通し番号、チェックポイント番号及びエラー状況フィールドを内含するFP\_ERR\_STAT信号を含む。エラー状況フィールドは、実行済みの浮動小数点命令の結果をもたらされる浮動小数点トラップタイプ又は1つももたらされない場合には浮動小数点トラップ無しを、そして浮動小数点命令により引き起こされた現行の浮動小数点例外又は例も引き起こされなかった場合には現行浮動小数点命令無しを表示する。

40

#### 【0305】

その上、図39を参照すると、FSR/CCRFRN606は、SPARC-V9浮動小数点状況レジスタ(FSR)を内含する。FSRは、上述のIEEE\_754\_例外の各々について1つのマスキングビットを含むトラップ有効化マスク(TEM)フィールドを含み、これらの例外のうちの単数又は複

50



数のものがマスキングされうようになっている。TEMフィールドはFPU602に提供される。IEEE\_754 \_\_例外が起こると、FR\_ERR\_STAT 信号は、この例外がマスキングされるべきタイプのものであることを TEMフィールドが表示した場合には、1つの例外が発生したことを PLSM307に対して表示しない。しかしながら、FP\_ERR\_STAT 信号はそれでも FPEXCEPユニットに対して、このようなIEEE\_754 \_\_例外が発生した（すなわちFTT）ことと同時にそれがIEEE\_754 \_\_例外のどのタイプであるか（すなわちCEXC）を表示する。

【0306】

完了済み浮動小数点の各々について、FTTWR論理902 は、FP\_ERR\_STAT 信号により識別された FTTデータを FPEXCEPリング306 の対応する FTTフィールド内に書込むため、FP\_ERR\_STAT 信号内に提供された通し番号を用いる。その上、CEXC書込み（WR）論理903 は、EP\_ERR\_STAT 信号によって識別されたCEXCデータを FPEXCEPリング306 の対応するCEXCフィールド内に書込むために、この通し番号を用いる。

10

【0307】

読取り及び累積例外計算（RD and AEXC compute）論理904 は、PSU300の ICPU301から RRPポインタ及びCOMMIT\_VALID信号を受理する。RRPポインタは、前のマシンサイクル内で退去させられた最後の命令をポイントし、COMMIT\_VALID信号は RRPポインタが前のマシンサイクル内をどれほど遠くまで前進するか（すなわち退去させられた命令の数）を表示する。

【0308】

各々のマシンサイクル中、RD及びAEXC計算論理904 は RRPポインタ及びCOMMIT\_VALID信号を用いて、最後のマシンサイクル内で退去させられた各命令について FPEXCEPリング306 内のFPビット及び FTT及びCEXCフィールドを読みとる。その後、浮動小数点命令（FPビットにより識別されたような）である退去済み命令について、RD及びAEXC計算論理904 は、そのCEXCフィールドを論理和してAEXCフィールドを計算する。要約したとおり、マスキングされていないIEEE\_754 \_\_例外をひき起こす浮動小数点命令は退去されておらず PLSM307による実行トラップ順序決定を結果としてもたらずことになることから、AEXCフィールドは、最後のマシンサイクル内で退去させられた浮動小数点命令の全てによって累積されたマスキング済みIEEE\_754 \_\_例外を識別する。

20

【0309】

さらに、RD及びAEXC計算論理は、読取りFPビットから、現行のマシンサイクルの中で退去させられた命令のうちのいずれが、退去されるべき最も晩期の浮動小数点命令であるかを決定する。現行のマシンサイクル内で単数又は複数の浮動小数点命令が退去させられた場合、RD及びAEXC計算論理904 は、計算上のAEXCフィールドを含む信号及び図39内の FSRのfsr.aexrフィールド内に書き込むべきであることを表示する信号を含むWR\_CEXC\_AEXC\_FTT信号を出力する。その上、WR\_CEXC\_AEXC\_FTT信号は、同様に、ほとんどの現行の退去済み浮動小数点命令のためのFTT 及びCEXCデータを含む信号及び図39内の FSRのfsrftt及びfsr.cexcフィールドの中にこのデータを書き込むべきであることを表示する信号をも含んでいる。この状況下で、FTTデータは、いくつかの理由で浮動小数点トラップ無しタイプを表示する。まず第1に、実行例外をひき起こさない浮動小数点命令は、退去される可能性があり、又浮動小数点トラップ無しタイプを表示する FPEXCEPリング内の対応する FTTフィールドを有することになる。第2に、IEEE\_754 \_\_例外トラップを表示する FTTフィールドを有するもののFSR607の TEMフィールドによってマスキングされた単数又は複数の対応するIEEE\_754 \_\_例外をひき起こした浮動小数点命令は、これらの実行例外のいずれかが発生したことについて PLSM307が警告を受けていないことから、退去される可能性がある。

30

40

【0310】

前述のとおり、PLSM307は同様にFR\_ERR\_STAT 信号も受理する。これらの信号が、例外トラップが発生したことを表示した時点で、PLSM307により実行トラップ配列決定プロセスが開始され、こうして CPU51はトラップ誘発命令に先立つ命令において同期化されることになる（すなわち RRP=CSN=ISN）。

50

## 【 0 3 1 1 】

CPU51が同期化された時点で、PLSM307はFPEXCEP\_CHP信号を生成する。FPEXCEP\_CHK信号に応じて、RD及びAEXC計算論理902は、RRPポインタを用いて、RRP+1にある命令のためのFPEXCEPリング306内のFP、FTT、及びCEXCフィールドを読みとる。

## 【 0 3 1 2 】

読取りFTTフィールドが、浮動小数点トラップ無しを表示した場合、RRP+1にある命令は、浮動小数点例外をひき起こす原因ではなかったことになる。この命令では、いくつかの他の種類の例外が発生した可能性もあることから、RD及びAEXC論理904は、fsr.ftt、fsr.cexc及びfsr.aexcフィールド内に例も書き込まれるべきではないことを表示する信号を内含するように、WR\_CEXC\_AEXC\_FTT信号を出力する。

10

## 【 0 3 1 3 】

しかしながら、読取りFTTフィールドが浮動小数点トラップ発生を表示した場合、RRP+1での命令が、浮動小数点例外の原因であったことになる。このとき、RD及びAEXC論理904は、RRP+1での命令のためのFPEXCEPリング306から読みとられたFTTフィールドを含む信号及び図39内のFSR607のfsr.fttフィールド内に読みとられるべきであることを表示する信号を含むように、WR\_CEXC\_AEXC\_FTT信号を出力する。かくして、浮動小数点命令によってひき起こされたトラップタイプは、浮動小数点実行トラップ取扱いルーチンがトラップを適切に取扱うべく記憶FSR(STFSR)命令でFSRにアクセスできるような形で、FSRのfsr.fttフィールド内に含まれている。

## 【 0 3 1 4 】

さらに、読取りFTTフィールドがIEE\_754\_\_例外が発生したことを表示した場合、RD及びAEXC論理904によって出力されたWR\_CEXC\_AEXC\_FTT信号は、RRP+1での命令のためのFPEXCEPリング306から読みとられたCEXCフィールドを含む信号及び図39内のFSR607のfsr.cexcフィールド内に読みとられるべきであることを表示する信号を含む。その結果、浮動小数点命令によってひき起こされた現行の例外タイプは、ここでトラップ取扱いルーチンが記憶FSR(STFSR)でFSR607のこのフィールドにアクセスしトラップを適切に取扱うことができるような形で、FSRのfsr.cexcフィールド内に含まれる。

20

## 【 0 3 1 5 】

図37を参照すると、この場合、浮動小数点命令は予測された及び/又は実際のPC順外でFPU600によりout-of-order実行されることから、RRP+1における命令は、実行トラップ順序決定プロセスをPLSM307に開始させた障害発生命令ではなかった可能性がある。換言すると、浮動小数点命令は、より早期に発行された(すなわちより早期の通し番号をもつ)もののより晚期に実行されたもう1つの浮動小数点命令によってひき起こされた実行トラップよりも早く検出された実行トラップを結果としてもたらした可能性がある。しかしながら、より晚期に発行されたもののより早期に実行された命令は、より早期の命令が1つの例外を結果としてもたらした場合にマシンサイクルに先立って順序決定しながらPLSM307による実行トラップ順序決定の開始をひき起こし、その後PLSM307はその処理へと切替わることになる。かくして、実行トラップ順序決定はそれでも、あたかもその実行トラップが実際のPC順で発生したかのごとく、より晚期に検出された実行トラップがより早期に検出された実行トラップに代って実際に取扱われるという結果をもたらす。

30

40

## 【 0 3 1 6 】

FPEXCEPユニットについてSPARC-V9アーキテクチャという状況下で記述してきたが、当業者であれば、浮動小数点命令の機械的実行が関与するようなあらゆるアーキテクチャのためにこれを実施することができる、ということも理解できるだろう。

## 【 0 3 1 7 】

### C. プロセッサをより早期の状態にバックトラッキングすることによる回復

ウォッチポイント304が誤予測命令を検出した時点、又はDFB62内で実行例外が発生し、PSU300内の実行例外(etrap)をトリガーした時点で、CPU「バックトラック」が開始される。プロセッサをバックトラッキングすることには、以下でより詳細に説明する通り、プロセッサバックアップ及び/又はプロセッサバックステップが含まれる可能性がある。

50

1つのマシンサイクル中、1つ以上の誤予測及び実行例外が発生し得る。典型的な CPU51 においては、全ての機械的に実行された制御転送命令（例えば、予測された分岐命令）がチェックポイントされる。このような誤予測された命令はつねに「チェックポイント境界」に存在する。チェックポイント済みの誤予測命令からの回復は、発行時点における誤予測命令に対応するチェックポイントの中に記憶されたマシン状態を回復することによって達成できる。実行例外は、誤予測とは異なり、全ての命令がチェックポイントされるわけではないことから、チェックポイント済み命令に対応しないかもしれない。一般に、実行例外を生成する命令が、あらゆる「命令境界」に存在し得る。1つのチェックポイント境界は、機械的に発行された制御転送命令についてのみ1つの命令境界に対応する。機械的に発行された命令の全てがチェックポイント形成を要求するわけではなく、むしろPC不連続性を作り出すような機械的制御転送命令だけである（ただし、制御転送命令がマシン「同期化」命令として選択され、本書に記述のとおり選択的にチェックポイントされる場合は除く）ということをお願いしたい。従って CPU51がバックトラックされる要領は、回復を開始した条件及び誤予測又は例外がチェックポイントされた命令に対応しているか否かによって左右される。

10

20

30

40

50

#### 【0318】

本書で使用される「バックアップ」という語は、チェックポイントの中に以前に記憶されたマシン状態を回復することによってチェックポイント境界においてマシン状態を回復し、次に CPU51命令の発行及び実行を再開するべく適切な行動をとるための構造及び方法のことを言う。本書で使用されているように、バックアップには、チェックポイントされた命令から順方向の命令のその後の in-order再発行及び再実行が関与していてもいなくてもよい。ここで使用される「バックステップ」という語は、可能な場合にはバックアップと合わせてあらゆる命令境界でマシン状態を回復し次に CPU51命令発行及び実行を再開するべく適切な行動をとるための構造及び方法のことを言う。

#### 【0319】

実行トラップ (etraps) に遭遇した場合、マシンは、その精確な状態を保存するため障害発生命令までバックトラックする。チェックポイント境界上に存在する誤予測と異なり、etrapsは、障害発生命令に達するべくバックステップ、バックアップ又はその両方の組み合わせを必要とする。オペレーションの実行の必要性以外にはバックステップ又はバックアップオペレーションの順序付け又は頻度にはいかなる制約条件もない。合わせて用いられた場合、マシンのバックアップ及びバックステップは、該予測及び実行例外からの効率の良い回復方法を提供し、しかも或る種の状況下では、有利にも、例外をひき起こす命令又はチェックポイント済み命令と障害発生命令の間の命令の再実行を強制することなく、これを行なうことができる。

#### 【0320】

プロセッサのフローの中に変化があった場合（例えば、Bcc, FBcc, jumpI及び同様の命令）にはつねに、チェックポイントが割振られる。典型的な CPU51は、割当てのために最高16のチェックポイントが利用可能な状態にある。バックアップはマシン状態をチェックポイント境界へ回復するために行なわれる。あらゆる命令境界にマシン状態を回復するためにバックステップが行なわれる。典型的なマシンは、利用可能なハードウェアによりリサイクルあたり最高4つの命令をバックステップすることに制限される。バックアップはマシン状態を回復するための粗いが迅速なバックトラック機構を提供し、バックステップは、細かい命令境界においてマシン状態を回復するための機構を提供する。マシンは、バックアップ又はバックステップのあらゆる組合せを通して、より早期の状態までバックトラックすることができる。

#### 【0321】

発明力あるバックアップ手順と合わせて発明力あるバックステップ手順を用いて精確な状態回復を実現するためには、2つの規則を遵守しなければならない。まず第1に、命令ストリーム内で、アーキテクチャ制御レジスタを修正し得る命令に遭遇した場合、CPU51が、命令の実行に先出ち同期化されなくてはならないか、又はマシンがその命令をチェッ

クポイントしなくてはならない。この第1の規則が遵守された場合、バックステッピングは、いかなるアーキテクチャ制御レジスタの修正も更新も決して要求しない。第2に、プログラムカウンタの不連続性を作り出す可能性のある全ての命令は、チェックポイントされなくてはならない。この第2の規則が遵守された場合、バックステッピングには、PCをバックステップ数だけ減分し、より早期の対応するプログラムカウンタで割当てられるにつれてマシン資源（RRF302内に維持された資源を含む）を回復することが関与する。

#### 【0322】

PSU300内に機械的に位置設定されたバックトラックユニット（BKTR）305が、適正なCPU51の機能及び回復に必要とされるようなマシンバックアップ及びバックステップを実施することを担当する。図40に示されている通り、BKTR305は、マシンバックアップ、単数又は複数のマシンバックステップ又はバックアップとバックステップの組み合わせのいずれか誤予測及び例外からのCPU51の回復にとって適しているかを決定することを担当するバックトラック制御ユニット（BTCN）401を含んでいる。BTCN401は、DFB62内の各々との実行ユニット及びウォッチポイント304からSN、エラー及び状況情報を受理する。バックアップユニット（BKUP）402は、マシンバックアップを開始し順序決定することを担当し、バックステップユニット（BKST）403はマシンバックステップを開始し順序決定することを担当する。CPU51は、命令発行を停止し、例外又は誤予測をひき起こす命令に適する通りバックアップ及びバックステップを用いてバックトラックしなくてはならない。バックステップを用いてCPU61をバックアップするため及びマシンをバックステップするための構造及び方法について以下でさらに詳しく記述する。

#### 【0323】

##### 1. チェックポイント境界へのプロセッサのバックアップ

マシンバックアップは、(1) PSU300内のPLSM307がRED Modeへのエントリを試みたとき、又は(2) ウォッチポイント304が誤予測命令（mispredicted instruction）を検出したときに起動される。また、(3) DFB62から障害命令（faulting instruction）（実行例外（execution exception））、即ちe-トラップが検出されたときは、マシンのバックアップが起動される場合もあり、そうでない場合もある。バックアップ手続きは、バックアップのために実行されたチェックポイント命令（checkpointed instruction）の選択を含めて、バックアップを起動する理由によって異なる。実行例外の原因となる障害命令がチェックポイント境界に一致しない場合には、バックステップがバックアップに続いて行われる。

#### 【0324】

各マシンサイクル毎に、バックアップユニット（BKUP）402は、データフォワードバスに関する命令の実行状態を示す情報ERR\_\_STATをDFB62から受信すると共に、ウォッチポイント304からは誤予測命令に関する情報を受信する。これらの信号は、命令のシリアル番号（SN）、必要に応じてマシンのバックアップを命ずるチェックポイント、エラー発生の有無に関する指示、及びエラー形式に関する指示（例えば、据置（deferred）、データ中断、浮動小数点動作等）を識別する。FXAGUが乗算又は除算処理を行わない場合には、その実行ユニットからのERR\_\_STAT信号は、一般にエラー情報は与えず、ERR\_\_STAT信号中の命令のシリアル番号SN及びチェックポイント番号に関する情報を与える。チェックポイントは、発せられる全ての命令に対して形成されるものではないが、各命令にはその発生時点で、バックアップチェックポイントに割り当てられる。このチェックポイントは、命令発生以前にPSU300内のICRU301によって割り当てられ、命令発生時にはISU200によって命令と組み合わせられ、ISU200によって演算コード（op-code）及び命令のシリアル番号と共にDFB62に送られ、実行ユニットの待ち行列の一つに記憶される（例えば、バックアップチェックポイントは、チェックポイントフィールド614に記憶される。図22参照）。従って、命令実行が完了すると、シリアル番号及びチェックポイント番号は直ちに、実行ユニット（例えば、FXU、FPU、FXAGU、又はLSU）内で利用可能な状態となる。

#### 【0325】

## 2. 誤予測命令、RED Mode、及び実行トラップ起動によるバックアップ

誤予測命令がウオッチポイント304によって検出されたとき、又はRED Modeへのエントリが行われたときは何時でも、BKUP402によってバックアップサイクルが起動され、実行例外からの回復が行われる。この実行例外からの回復はバックステップだけ、又はバックアップだけ、或いはバックアップと1以上のバックステップとの組合せを必要とする。BKUP402はチェックポイント・ベクトル・ロジックユニットを含み、このユニットは各サイクル毎に、DFB62及びウオッチポイント304から受け取る命令の実行状態に関する情報を評価し、どの命令が誤って予測されたか、又はどの命令が実行例外を起こしたかを決定する。典型的なCPUでは、ロジックユニット404は、DFB62又はウオッチポイント304から受けた各チェックポイントに関して“OR”即ち論理和を取る。最早チェックポイント選択ロジック405は、どのチェックポイントが最早かを決定し、後の障害命令に対応するe-トラップが、既に強制終了させたロジック404によって識別されたチェックポイントへのバックアップを起動しないようにする。BKUP402は、どのチェックポイント番号をバックアップに使うかを決定する際に、誤予測及び実行例外を考慮する。

10

20

30

40

50

### 【0326】

マシンは、誤予測命令が最早チェックポイントであるときには、誤予測命令チェックポイント間で戻される。RED Mode起動のバックアップに関しては、マシンはコミットしたシリアル番号(CSN)に最も近く、それよりは大きい(A-リング循環の意味で)チェックポイントへ戻される。また、実行例外に関しては、チェックポイントの選択はより複雑であって、実行例外の性質に依存する。

### 【0327】

実行例外(e-トラップ)起動のバックアップの場合、据置及び非据置トラップによって、異なったバックアップチェックポイント番号の計算が必要になる。通常非据置トラップに関しては、もしISNがチェックポイントn+1を越えて移動したとき、チェックポイントnの命令iが障害命令であれば、マシンはISNからチェックポイントn+1、即ち障害命令の後のチェックポイントまで戻される。もし、チェックポイントn+1がアクティブでもなく又は有効なチェックポイントでもなければ、チェックポイントを要求する命令は発せられず、又は時間切れチェックポイントも形成されないから、バックアップは行われない。その代わりに、障害命令がチェックポイント直後のシリアル番号スロットになれば、ISNから障害命令に後退するバックステップだけを使う回復が適当である。例えば、チェックポイントmがシリアル番号SN=10を持つように割り当てられ、その時スロットSN=11にある命令が障害命令であれば、チェックポイントm+1へバックアップの代わりに、チェックポイントmへのバックアップが起動される。即ち、マシンは障害命令以前のチェックポイントまで戻される。

### 【0328】

実行されたチェックポイント命令が投機的分岐又はその他の制御転送命令に対応している場合には、実行されたチェックポイント命令と次の命令との間に、プログラムカウンタの不連続点が存在することになる。典型的なバックステップ動作は、プログラムカウンタの不連続点を越えて実行されることはなく、単一プログラム命令のシリアル番号差があるだけでも拘わらず使用することは出来ない。この状態では、バックアップ動作によるのみ、プログラムカウンタとマシン状態を適切に回復することができる。

### 【0329】

据置トラップは、非据置トラップとは異なるバックアップ・デスティネーション計算を必要とする。例えば、チェックポイントmがシリアル番号SN=10を持つように割り当てられ、その時SN=11にある命令が障害命令であって、据置トラップを発生すれば、ISNはトラッピング命令に先立つ命令よりはむしろトラッピング命令を指定するから、マシンはチェックポイントmの代わりにチェックポイントm+1(もし在れば)まで戻される。もし、その時チェックポイントm+1がなければ、バックアップは起動されない。その代わりに、正確な状態は、マシンのバックステップだけを用いて回復される。表10は

、チェックポイント番号及び障害命令のシリアル番号に関する幾つかを条件として行った据置及び非据置トラップに対するバックアップ作用の概略を示す表である。

【0330】

バックトラック305が、マシンバックアップによって影響を受けるCPU51にバックアップが生じていることを知らせる信号と、この影響を受けるCPU51の各要素にバックアップに使用するチェックポイント番号を知らせる4ビットのデータ信号とから成る信号(DO\_\_BAKUP)をアサートしたとき、プロセッサの状態は実行されたチェックポイントのシリアル番号に於いて回復される。実行されたチェックポイントの状態は、マシンに局部的に記憶されるから、CPU51の各々は、DO\_\_BAKUP信号に応答して影響を受けたユニット内の記憶領域から状態を“回復”しなければならない。

10

【0331】

【表9】

表10. チェックポイント番号及び障害命令のシリアル番号を条件として行った据置及び非据置トラップに対するバックアップ作用

トラップは据置か 又は非据置か?	トラッピング命令 はチェックポイント の直後にSNを持 っているか?	チェックポイン トn+1は有効 か? (チェック ポイントnは有 効と仮定する)	取られたバックアップ アクション
非据置	NO	NO	バックアップなし
非据置	NO	YES	n+1にバックアップ
非据置	YES	NO	nにバックアップ
非据置	YES	YES	nにバックアップ
据置	NO	NO	バックアップなし
据置	NO	YES	n+1にバックアップ
据置	YES	NO	バックアップなし
据置	YES	YES	n+1にバックアップ

20

30

【0332】

実行されたチェックポイントに於ける状態の記憶、保持、回復を受け持つCPU51のユニット、同様にバックアップ動作を順序づけるCPU51のユニット(特にPSU300内のユニット)は、DO\_\_BAKUP信号を受信し、かつそれに応答する。RRF302はDO\_\_BAKSTEP信号を受信するが、DO\_\_BAKUP信号は受信しないことに注意されたい。これは、RRFがCPU51をバックアップによってではなく、ステップ毎に状態を回復するのに直接必要とされるからである。

【0333】

更に明確に言えば、図6に於いて、チェックポイント記憶ユニット106は、DO\_\_BAKUP信号によって特定されるチェックポイント番号に対応するエントリ時に、其処に記憶された実行されたチェックポイントのAPC及びNAPCの値を、CHKPT\_\_PC及びCHKPT\_\_NPC信号の形で出力する。これらの値はAPC及びNAPCのレジスタ112、113に記憶される。同様に、制御レジスタチェックポイント記憶ユニット817は、DO\_\_BAKUP信号によって特定されるチェックポイントに関して実行された制御レジスタの内容を、CHKPT\_\_CONTROL\_\_REG信号の形で出力し、これら内容は図17に図示の制御レジスタファイル800中の対応する制御レジスタNORD/WR更新ロジックによって記憶される。更に、図16に於いて、FXRFRN604、CCRFRN601、及びFPRFRN605各々のチェックポイント記憶ユニット616は、

40

50

DO\_\_BACKUP信号によって特定されるチェックポイントに関して実行されたチェックポイントのリネームマップを、CHKPT\_\_MAPの形の信号で出力する。この実行されたチェックポイントのリネームマップは、制御ロジック613によってリネームマッピングロジック615に戻して記憶される。更に、フリーリストユニット700は、FXRF RN604、CCRF RN601、及びFPRFRN605各々の物理的レジスタの実行されたチェックポイントのフリーリストを、CHKPT\_\_FREELIST信号の形で出力し、この信号はフリーリストロジック701に記憶される。

#### 【0334】

典型的なCPU51では、e-トラップがまだ続いているときでも、バックアップは起動される。e-トラップ例外処理は多重サイクルを必要とするから、マシンが同期(ISN=CSN=RRP)され、トラップを取って処理する以前に、一つ以上のe-トラップをDFB62がアサートすることは可能である。典型的なCPU51の一実施例では、スロット0及び1の各々に関するDFB62からのエラー及び状態信号(ERR\_\_STAT)の各々は、1ビットエラー信号、6ビットのシリアル番号信号(FX\_\_SN、FP\_\_SN、LS\_\_SN、及びFXAG\_\_SN)、及び各命令毎に適当なバックアップチェックポイントを計算するのに使用される16ビットの1-ホット・チェックポイント・ベクトル(FXDF\_\_CHKPNT、FPDF\_\_CHKPNT、FXDF\_\_CHKPNT、LSD F\_\_CHKPNT)からなっている。更に、誤予測命令に関する情報信号WP\_\_MISPREDは1-ホットベクトルであって、其処でのビットiのアサートは、i番目のチェックポイントが誤予測命令に対応し、バックアップを必要としたことを示す。チェックポイントベクトルロジックユニット404は、スロット0及び1の各々からの16ビットの1-ホットチェックポイントベクトルの全てについて、累積的に論理和を取る。この動作はマシンがベクトルをトラップテーブルにするまで続けられる。論理和を取ったチェックポイントベクトルは、最早チェックポイントを決める最早チェックポイント選択ロジック405に供給される。たとえ、後の命令に対応して実行例外が生じても、前のバックアップによって既に強制終了したチェックポイントへのバックアップは起動されない。典型的なCPU51では、ウォッチポイント誤予測信号(WP\_\_MISPRED)は16ビットの1-ホットベクトルで、其処でのビットiのアサートは、i番目のチェックポイントが誤予測命令に対応し、バックアップを必要としたことを示す。

#### 【0335】

### 3. 何れかの命令境界へのプロセッサのバックステップ

マシンバックステップはDFB62に於ける障害命令から発生された実行例外(e-トラップ)の結果としてのみ起動される。マシンバックステップはマシンバックアップとの組合せで起こることが多いが、バックアップを起動しなくても起動される。

#### 【0336】

バックステップユニット(BKST)403はバックステップの実施を受け持っている。マシンの“バックステップ”はDFB62に於ける障害命令から発生するe-トラップの結果として起こるだけである。バックステップが実施されると、2つの事象がマシンに起こる。第1に、未決命令が強制終了されるため、資源は回復されて利用可能になる。これら資源にはシリアル番号、チェックポイント、ウォッチポイント、及び割り当てられた物理的レジスタ及び論理レジスタが含まれる。第2に、ISNはバックステップ動作中減分するが、CSN及びRRPを下回って減分することはない。障害命令は依然としてそのアクティブビットをA-リングセット中に有していて、それによってCSN及びRRPの送り(advancement)を抑えている。障害命令がエラーを起こさずに首尾良く実行されたときだけ、その命令に対応するA-リングビットがクリアされる。(据置トラップを上手く扱った時に、A-ビット(及びB-ビット)はクリアとなって、CSN(及びNMCSN)の送り及び資源の最終回復を許容することを除く)。

#### 【0337】

マシンバックステップ動作は、バックステップ始動制御信号(BACKSTEP\_\_LAUNCH)及びバックステップに使用したい命令のバックステップシリアル番号を識別す

10

20

30

40

50

る P L S M 3 0 7 からのバックステップシリアル番号識別信号 ( B A C K S T E P \_ \_ T O \_ \_ S N ) によって B K S T 4 0 3 に於いて開始される。これに応じて、 B K S T 4 0 3 は、バックトラック 3 0 5 がバックステップモードを開始していることを I C R U 3 0 1 及び他の C P U 5 1 2 知らせるバックステップ通知信号 ( D O \_ \_ B A C K S T E P ) を発生することによってバックステップ手続きを順次実施する。 B K S T 4 0 3 は、バックステップを終えた命令番号のカウントを識別し、 P L S M 3 0 7 によるタスクを果たすために起動することが可能な周辺制御装置として見る事ができる信号を発生する。 B K S T 7 0 3 は P L S M の要求に応じてタスクを開始し、タスクを果たし終えた時には、タスクの終了を P L S M に指示して、次の要求に待つ待機する。

#### 【 0 3 3 8 】

B K S T 4 0 3 は 2 つの条件形式に関するバックステップを扱う。その第 1 の形式は非据置 e - トラップである。非据置 e - トラップは、据置トラップ信号が P L S M 3 0 7 によってアサートされず ( I S A \_ \_ D E F E R R E D \_ \_ T R A P = 0 )、かつバックステップ開始制御信号が、据置トラップではない e - トラップを示す信号 ( B A C K S T E P \_ \_ L A U N C H = 1 ) であるとアサートされている場合に起動される。この場合、バックステップ量はバックステップされるシリアル番号 ( B A C K S T E P \_ \_ S N ) と次ぎに発せられるシリアル番号 ( N I S N ) との差を決める B K S T 4 0 3 で計算される。発せられた命令のシリアル番号 ( I S N ) ではなく、次ぎに発せられるシリアル番号 ( N I S N ) を使用することによって、マシンは障害命令直前の命令まで戻る。このバックステップ量は、 C P U 5 1 内の記憶領域に保存され、障害命令の前の命令に到達するまで減分される。

#### 【 0 3 3 9 】

第 2 のバックステップ条件は据置 e - トラップを含んでいる。据置 e - トラップは、据置トラップ信号が P L S M 3 0 7 によってアサートされてる信号であって ( I S A \_ \_ D E F E R R E D \_ \_ T R A P = 1 )、かつバックステップ開始制御信号が、据置トラップである e - トラップを示す信号 ( B A C K S T E P \_ \_ L A U N C H = 1 ) であることをアサートしている場合に起動される。この場合、バックステップ量はバックステップされるシリアル番号 ( B A C K S T E P \_ \_ S N ) と、発せられるシリアル番号 ( N I S N ) との差を決める B K S T 4 0 3 で計算される。此処でマシンは障害命令の前の命令ではなく、障害命令にバックステップされる。計算されたバックステップ量は、通常の非据置バックステップより小さい。

#### 【 0 3 4 0 】

典型的な C P U 5 1 では、 4 つの命令の内の最大のもので、各マシンサイクルに関してバックステップされる。従って、 4 つまでの命令の多重バックステップはそれぞれ行き先 (バックステップの) 命令に到達する。一般に、シングルサイクルに関してバックステップ命令の数を限定する必要はない。典型的な実施例に於けるもう一歩進めた限定は、 P C をバックステップ量だけ I S N (又は N I S N ) から減じても、適当な状態回復とはならないから、バックステップ方法はプログラムカウンタの不連続の原因となる命令に関するバックステップをサポートするものではない言うことである。

#### 【 0 3 4 1 】

“ I D L E ” 及び “ B A C K S T E P ” とする 2 つの状態からなる簡単なマシン状態は、バックステップを順次実施するのに使用される。 I D L E 状態にある間の “ B A C K S T E P \_ \_ L A U N C H ” アサートは、 B A C K S T E P 状態への遷移となる。また、 B A C K S T E P 状態にある間に D O \_ \_ B A C K S T E P がアサートされると、 2 つの終了条件が評価される。これら 2 つの条件の何れかが、 B A C K S T E P を終了させ、 I D L E に戻す。

#### 【 0 3 4 2 】

バックステップ中に於ける C P U 5 1 の状態回復には、マシンレジスタ資源を更新すること、及びプログラムカウンタをバックステップ量だけ減分することだけが含まれる。上記のように、構成上の制御レジスタを修正する命令は、マシンを同期するか、チェックポ

10

20

30

40

50



イントを実行するから、バックステップ中に構成制御レジスタの回復を必要としない。典型的なCPU51では、以下のマシン状態はバックステップ中に更新される。即ち、プログラムカウンタ(PC)、次のPC、RRFに記憶された情報、及びリネームマップに記憶された情報が更新される。トラップPC、トラップRRF、及びBRB59の特権レジスタを含むその他の制御レジスタは、バックステップ動作によって修正も更新もされない。マシンの資源は、レジスタリクレームフォーマットファイル(RRF)302に記憶されたロジカル-ツウ-オールド-フィジカルマッピングに関連するレジスタファイルから回復される。バックステップ通知信号(DO\_\_BACKSTEP)がアサートされると、ICRU301、RRF302、及び他のCPU512は、バックステップモードが起動されていることが通知される。バックステップに於いて、各バックステップサイクル中に回復及び記憶されるRRF302アイテム数は、バックステップカウント信号によって示されるバックステップされる命令の数に等しい。

10

## 【0343】

従って、これらの状態の記憶、保持、及び回復を果たすCPU51は、バックステップ動作を順次果たすCPU51(特にPSU300内の)と共に、BRB59、ISB61、及びDFR62内のユニットを含めて、バックステップ信号を受信し、それに応答する。RRF320は、CPU51をバックステップさせて、バックアップではなくステップ毎にそれを回復することだけに直接関与するから、DO\_\_BACKSTEP信号を受信するがDO\_\_BACKUP信号は受信しない。

20

## 【0344】

レジスタリクレームフォーマットユニット302は、バックステップをアサートした信号(DO\_\_BACKSTEP)に応答し、リクレームされて回復されるRRFアイテムの数は、DO\_\_BACKUP信号内のバックアップステップカウント数に等しい。RRFは、ロジカル-ツウ-オールド-フィジカルマップ、即ちソース又は行き先データ値を保持するロジカル-ツウ-オールド-フィジカルレジスタの割当を含んでいる。RRF302の命令に割り当てられたレジスタ資源、又はレジスタマップの何れもCSN又はRRPを進めることによってフリーリスト700へは自由に戻れないから、レジスタの実際のデータは妨害されない。RRFは、バックステップした命令をもと実行したきと同じ関係で、フィジカルレジスタ(オールドフィジカルレジスタ)をロジカルレジスタを再度組み合わせる。レジスタマッピングを一つずつ回復することによって、各命令のレジスタ状態が回復される。要するに、レジスタ資源から見れば、マシンはバックステップ動作中後退する。図18はRRFノブロック図である。図14は、ロジカル-フィジカルマッピングが、レジスタリネームファイルFXFRN、CCFRN、FCCFRN及びFPFRNに回復される仕方を示す図である。

30

## 【0345】

RRFの典型的な実施例では、幾つかの情報(LOG\_\_OPHYS\_\_TAGS)を含むロジカル-オールドフィジカルレジスタ情報は、データ保存構造又はユニット366の何れかに保存のため、読み/書き制御ユニット365にまとめられている。回復のためRRFをレジスタリネームマップに読み出すとき、RRFデータは回復のためにもとのフォーマットに戻される。本願の教示から当業者は、種々の保存フォーマット及びデータのまとめ/解き方の手順をRRFデータに適用して保存領域を最小化し得ることが分かるだろう。

40

## 【0346】

特に、バックステップが行われているとき、読み/書きロジック365は、保存ユニット366を制御して、DO\_\_BACKST信号によって識別される命令に対応するロジカル-オールドフィジカルレジスタ・タグ・マッピングをLOG\_\_OPHYS\_\_TAGS\_\_BKST信号の形で出力する。図8、16、18、及び39に於いて、ロジカル-オールドフィジカルレジスタ・タグ・マッピングは、FXRFRN640、FPRFRN606、及びCCRFRN610に復元される。図16に示すように、DO\_\_BACKST信号に応答する制御ロジック613は、これらのマッピングをリネームマッピングロジック6

50

15に復元する。

【0347】

更に練られたバックステップ計画は追加情報を維持することによって実施される。しかし、本願発明による方法では、プログラムカウンタの値は基本ブロック内の命令に対応する。即ちバックステップされる命令の間にはプログラムカウンタの中には不連続が存在しないから、方法は簡素化される。バックステップユニット403は単に実行された減入れのシリアル番号からのバックステップ数、又はバックアップなしにバックステップを用いられたときの現在のISNからバックステップ数を信号で合図する。もし、プログラムの不連続があれば、それらプログラムカウンタの減分は、必ずしも適当な回復とはならない。しかし、本願に含まれている教示にから当業者は、より複雑なバックステップ技術を用

10

【0348】

2つのバックトラック機構、即ちマシンバックアップ及びマシンバックステップは、投機的命令の続発を取り消す効果的な方法及び構成でプリサイス・ステート・ユニット(Precise State Unit)を提供する。マシンバックアップは、プリサイス・ステート・ユニットがCPU51をアクティブチェックポイント境界に対応する状態にすることを可能にする。マシンバックステップは、プリサイス・ステート・ユニットがCPU51をチェックポイント境界の間の特定の命令境界にバックトラックすることを可能にする。バックアップは、マシン状態回復のための粗いけれども早い方法及び構成を提供し、バックステップはマシン状態回復のための正確だが遅い方法及び構成を提供する。図41は何れの命令境界に於いても、正確な状態を保持し、回復するための本発明による実施例の流れ図を示す。図42はロジカル-フィジカルマッピングを、レジスタリネームファイルに復元する方法を示す図である。

20

【0349】

図43は2つのバックステップを伴うマシンバックアップの例を示す図である。バックステップは一度の幾つかの命令を、単一ステップ又は多重ステップとして実行できる。この例では、CPU51は命令障害が検出された時点で、ポインタISNに対応する命令を既に発している。マシンは障害命令の後の最も近いチェックポイントにマシンを戻すことによって応答する。このもっと近いチェックポイントとISNとの間の命令は強制終了させられる。4つの命令の量による第1のバックステップは、それらの命令を発している間、4つの命令(第1バックステップ参照)に関わる全ての資源を再生することによって、マシンを障害命令に更に近づける。次いで、3つの命令量による第2(及び最終)バックステップは、それら残りの命令(第2バックステップ領域参照)に関わる全ての資源を再生することによって、障害命令の直前にマシンを持ってくる。

30

【0350】

チェックポイントまでの戻りは早いですが、チェックポイントを回復するのに多くのマシン資源が必要になる。誤予測命令は全て、チェックポイントが作られる要因となるから、状態はワンステップで誤予測命令に戻される。バックステップはバックアップより遅いが、本発明のバックステップ手続きはマシン状態を、障害命令前のチェックポイントに於ける状態にではなく、特定命令の実行点に在ったときの状態に戻す。そして、本発明のバックステップ構成及び方法によって、マシンは従来の方法によるように、チェックポイントの前のチェックポイントへ行き、それから障害命令にバックステップするのではなく、障害命令の後のチェックポイントに戻ることが出来る。

40

【0351】

D. 例外及び誤予測回復時に於ける優先ロジック及び状態マシン動作

図44はCPU51内に機能的に置かれた典型的な優先ロジック及び状態マシン(PLSM)307を示す。PLMSは幾つかの役割を持っている。第1に、未処理又は並行例外群の最早の例外を識別する役割を果たす。例外条件には、RED Mode、発行トラップ(i-traps)、実行トラップ(e-traps)、割り込み、及び誤予測が含まれる。第2には、マシンサイクル中に到着する例外及び誤予測を、相互間で、また並行

50

処理される例外及び処理待ちのものとの間の優先順位を決める役割をは果たす。第3に、形式の異なる例外及び誤予測間の可能な状態遷移を決定する役割を果たす。第4には、Backtracking (Backupそして/又はBackstep)、REDモード処理を起動する制御信号を発する役割を果たす。

#### 【0352】

命令の発行又は実行中に起こる種々の形式の例外について、PLSM307による例外及び誤予測の優先順位付け及び扱いがよく分かるように説明する。REDトラップは、種々の条件下で起こり、全てのトラップの中で最も高い優先順位を有する非同期トラップである。RED Modeは、SPARC V-9アーキテクチャマニュアルに説明されている。i-trapsは命令発行時にISU200によって発生される発行トラップである。これらは、何れのレジスタスpill/フィル(spill/fill)トラップ、何れのトラップ命令(TA、Tcc)、命令アクセスエラー、違法命令、特権演算コード等を含んでいる。一般に、発行トラップは、取り扱いに関して低い優先順位にある割り込みに較べれば、高い優先順位にある。信号を送られるその他の全てのトラップ(誤予測、e-トラップ又はredトラップ)は、一時的には発行命令よりは早い命令に向けられる。

10

#### 【0353】

e-trapsはデータフォワードパスの一つから信号を送られる実行トラップである。実行トラップは、SN順に他の実行トラップに対して優先順位を付けなければならない。これは、実行トラップは必ずしもチェックポイント境界で発生するとは限らず、また障害命令に達するにはバックステップが必要になる場合があるからである。もし、マシンが障害命令に達するためにバックアップを実施すれば、何れかの新たな例外がより早い命令に向けられる。もし、実行トラップが順次実施されていれば、何れか新たな誤予測がより高位の優先順位となる。これは、誤予測命令は常にそれが利用し得るチェックポイントを有しているからである。もし、PSU300が必ずしも何れかの例外を実施していなければ、実行トラップ及び誤予測は、それぞれ他に対して優先順位が付けられていなければならない。割り込みは、非同期トラップのもう一つの形である。割り込みは他の全ての例外より下位の優先順位にある。

20

#### 【0354】

投機的に発した転送制御命令が誤予測であることをウォッチポイントが検出したとき、ウォッチポイントユニット304は誤予測に信号を送る。もし、何れかの非RED例外が順次実行されていれば、誤予測は常にチェックポイント境界上にあつて、一時的に早い命令に対して起こるから、優先順位を付ける必要はない。もし、例外が実施されていなければ、誤予測は割り込み及び発行トラップより高位の優先順位を持つが、実行トラップに対しては優先順位を付けなければならない。

30

#### 【0355】

最早シリアル番号選択ロジックユニット(ESNSL)481は、DFB62の各実行ユニットから、実行完了した命令のシリアル番号及び実行中に例外が起こったかどうかに関する情報、及びもし起こっていたら、その例外の形に関する情報を含む状態情報を受けると共に、ウォッチングユニット304からはそのサイクル中に起きた何れかの誤予測を示す信号(WP\_\_MISPREDICT)を受け、更に現在のISNを受け、また更にESNSL481からは、最終マシンサイクル(CURR\_\_EARLIEST\_\_SN)からの新たな例外情報を受け取る前の現マシンサイクルで処理された例外のSNを示す帰還信号を受け取る。各実行ユニットからの信号はERR\_\_STATを含んでいる。例えば、もし64ロケーションA-リング312の命令SN=10及びSN56に対して例外が生じれば、最早例外はISNの値を知ることなしには決めることは出来ない。もし、ISN=40なら、SN=56が最早例外となる。しかし、もしISN=6なら、SN=10が最早例外となる。

40

#### 【0356】

ESNSLU481は、最早例外及び誤予測を優先順位及び実行スイッチングロジック(PESL)ユニット482に知らせて、PESLが最早SN基準及び例外の形に基づい

50

て例外及び誤予測処理に優先順位を付けることが出来るようにする。また、PESLはISU200から発行トラップ信号（ISU\_\_ITRAP）及び割り込み信号（ISU\_\_INTERRUPT\_\_PEND）を受信すると共に、Backtrackがチェックポイント情報の比較に基づいて誤予測が実行トラップより早く生じたことを決定する場合に、Backtrackユニット305からMISPREDICTED\_\_EARLIER信号を受信する。また、PESLは状態マシン及び制御ロジック（SMCL）483から状態信号を受信する。SMCLは状態マシン484及び例外制御信号発生ロジック484を含み、現在処理している例外（例えば、実行トラップ、発行トラップ、RED等）があればその形を識別する。例外の状態は状態マシン484に記憶される。PESL482はこれらの入力を現在及び未決の例外の優先順位を決めるのに使用し、現在実施中のマシンサイクルのための新たな例外に形を識別する信号（NEW\_\_EXCEPTION\_\_TYPE）を発生する。

10

## 【0357】

例外制御信号発生ロジック484は、DPU51内の他のユニットにCPUの状態を知らせると共に、場合によっては、状態に応じて他のCPUユニットにアクションを取らせる信号を発生する役割を持っている。更に明確に言えば、ECSIGはTAKE\_\_TRAP信号を発生し、それをトラップが行われる制御部分及びトラップの種類を含むトラップスタックユニットに送る。このトラップスタックユニットはDEFERRED\_\_TRAP\_\_INCを、据置トラップが起きたかどうかを示すICRU301に送り、例外を越えて一つだけCSN及びNMCSNを進ませる。また、ECSIGはBacktrackに対して、どの命令SNにバックステップするか（BACKSTEP\_\_TO\_\_SN）、REDモード例外があるか、そしてREDモード例外の場合にはどの命令にバックアップするか（BACKUP\_\_RED）を告げる幾つの信号をバックトラックユニット305に送る。また、ECSIG484は命令強制終了信号をDFB62（KILL\_\_ALL）及びISU200（ISSUE\_\_KILL）に送り、DFB62に対しては、実行待ち行列中の命令の実行を強制終了するように指示し、ISUに対しては、そのマシンサイクル中に扱われている例外又は誤予測に適当な命令の発行を中止するように指示する。

20

## 【0358】

PLSM307はトラップを感知して5つの異なる形、即ち割り込み、レッドトラップ、発行トラップ、実行トラップ、誤予測に区分する。どの例外が現在順次処理されているかによって、検知される他の例外は高い優先順位を持っているか、そうでない場合もある。もし、新たな例外が高い優先順位を持っていれば、PESL482はSMCL483に対し現在の例外を順次処理することから新たな高い優先順位の例外又は誤予測にスイッチするように指示する。状態マシン484に於けるこれらの状態遷移は、現在の例外の形、即ち新たなエラーがウオッチポイント、ISU、又は6つのデータフォワードバスの何れから報告されたか、及び新たな例外が現在のものより一時的に早いかどうかによる。

30

## 【0359】

可能な状態遷移は以下のようにして起こる。RED - Noトラップ形は状態REDからの遷移を強制する。ITRAP - 3トラップ形（誤予測、実行トラップ、REDOアラート）は他の一つの状態への遷移を強制できるが、割り込みは遷移を強制しない。ETRAP - Redアラート及び誤予測は遷移を強制し、常に高い優先順位を有し、他の実行トラップは優先順位化を必要とし、割り込みは低い優先順位を有し、マシンはマシン同期のためバックアップ又は待機状態の何れかにあり、発行トラップを発する命令は発せられないから、発行トラップは不可能である。INTERRUPT - 全トラップ形は高い優先順位を有し、遷移を強制することが出来る。MISPREDICT - 3トラップ形（誤予測、事項とラップ、レッドアラート）は高い優先順位を有し、優先順位付けを必要とせず、張り込みは低い優先順位を有し、命令発行は誤予測処理ちゅう強制終了させられているので、発行トラップは起こらない。IDLE - Redトラップは最高優先順位を有し、同時誤予測及び実行トラップが、チェックポイントの比較に基づいて誤予測が早く起きたか、実行トラップが早かったかの決定が行われるBACKUPへの遷移に帰着する4つの形の

40

50

つを伴う。Backtrackは優先順位より早く決定が出来るから、優先順位化に基づいたチェックポイントに対する据置は好ましい。しかし、優先順位化はPSLMに於いて実行が可能である。最後に、発行トラップ、次いで割り込みが順次処理される。

【0360】

#### E. トラップ・スタックによるトラップの処理

先に示した如く、トラップは図5の命令パイプライン内で幾度も生じ、且つ、トラップ処理ルーチンにより処理されねばならない(即ち、受入れられねばならない)。図17を参照すると、トラップを受入れるべき例外をPSU 300が既述の手法で検知すると、PLSM 307はTAKE\_TRAP信号を生成し、これにより、制御レジスタファイル800にトラップを受入れさせると共に、受入れられるべきトラップのタイプを識別する。これに応じ、リード/ライト/更新ロジック816はTBAレジスタ812から対応トラップベクトル(TRAP\_VEC)信号を読み出してこれをBRB 59に送る。図6を参照すると、PCロジック106はTRAP\_VEC信号を使用して新たなFPC、APCおよびNAPCを算出し、これにより、対応トラップ処理ルーチンの命令を取り出して出力する。

【0361】

トラップ処理ルーチン内で生じたトラップがSPARC-V9アーキテクチャにより特定された如きネスト手法(即ち、受入れられたトラップの内部でトラップが受入れられる手法)で処理される様に、制御レジスタファイル800は図17に示されたトラップスタックユニット815を含んでいる。このトラップスタックユニット815は図45に更に詳細に示されているが、該ユニットは、記憶エレメントもしくは記憶エントリを備えたデータ記憶構造であるトラップスタック820を含んでいる。

【0362】

記憶エントリの各々は4個のフィールドを有しており、これらのフィールドはSPARC-V9アーキテクチャマニュアルによれば、トラップ・プログラム・カウンタ(TPC)、トラップ・ネクスト・プログラム・カウンタ(TNPC)、トラップ・ステート(TSTATE)、トラップ・タイプ(TT)の各フィールドである。TPCおよびTNPCフィールドは、トラップが受入れられる時点の、BRB 59のAPCレジスタ113およびNAPCレジスタ114内の夫々のAPC値およびNAPC値を含んでいる。SPARC-V9アーキテクチャマニュアルに記載された様に、TSTATEフィールドは、トラップが受入れられた時点で図39に示されたCCRFRN 610内のXICCレジスタの内容と、図17に示されたASIレジスタ808、CWPレジスタ809およびPSTATEレジスタ805の内容とを含んでいる。TTフィールドは、受入れられたトラップのタイプを識別する。

【0363】

図45に再度戻ると、トラップが受入れられたとき、現在のPC、NPC、TSTATEおよびTTの各値は、トラップスタック820の記憶エントリの中のひとつの記憶エントリの対応フィールド内に記憶されている。受入れられたトラップのネスト数を表示するトラップレベル(TL)は次にインクリメントされると共に、トラップ処理ルーチンは受入れられたトラップの処理を開始する。しかし乍ら、もしトラップ処理ルーチンの間にトラップが生じたときには、この第2のトラップの時点におけるPC、NPC、TSTATEおよびTTの各値は、トラップスタック820の記憶エントリの中のもうひとつの記憶エントリの対応フィールド内に記憶される。次に、第2トラップを処理する為に、第2のトラップ処理ルーチンが呼び出される。第2トラップ処理ルーチンの終わりにてはDONEもしくはRETRY命令が実行されて、第2トラップに対する記憶エントリ内に記憶されたPC、NPC、TSTATEおよびTTの各値が対応レジスタ内に書込まれると共にトラップレベルがデクリメントされる。この結果、CPU 51は、第2のトラップを受入れた時点の状態に戻ると共に、第1トラップ処理ルーチンが第1トラップの処理を再開する。その後、第1トラップ処理ルーチンのDONEもしくはRETRY命令が実行されたとき、CPU 51は第1トラップを受入れた最初の状態に戻る。

【0364】

以上から、SPARC-V9アーキテクチャにより必要とされる如く、トラップスタックユニット815は多数のトラップレベルを含むことにより多段階にネストされたトラップの処理をサポートすることは明らかである。但し、SPARC-V9アーキテクチャが4段階のトラップレ

ベルをサポートすべく4個の記憶エントリを備えたトラップスタックを指定してはいるが、上記トラップスタック820は4段の所要のトラップレベルをサポートすべく8個の記憶エントリ0~7を含んでいる。付加的な4個の記憶エントリは、レジスタまたは記憶エレメントの名称変更の概念が以下に記述する如くトラップスタックユニット815まで拡張され、トラップが投機的に受取られ且つそこから戻れるようになる。更に、前述の如く、記憶エントリ数がトラップレベル数より大きい限り、本発明は任意の個数のトラップレベルにて実施され得る。

**【0365】**

トラップスタックユニット815はフリーリストロジック821を含み、該ロジックは、各トラップと連携されたTPC、TNPC、TSTATEおよびTTの値を記憶すべく現在使用可能なトラップスタック820の記憶エントリの全てのリストを記憶している。図46を参照するに、フリーリストレジスタ822は、各マシンサイクルの間に、8ビットのFREELISTベクトル信号形態とされた現在使用可能な記憶エントリのリストを記憶する。このFREELIST信号の各ビットは、トラップスタック820の記憶エントリのひとつに対応すると共に、その記憶エントリが現在使用可能であるか否かを識別するものである。FREELISTベクトルにより使用可能であると識別された最初の記憶エントリに対し、ファーストワンエンコーダ823は次にそのベクトルをWR\_ENTRY信号にエンコードするが、該信号は、この記憶エントリを識別すると共に、受入れるべき次のトラップに対するTPC、TNPC、TSTATEおよびTTの各値を書込むためのポインタの役割を果たすものである。

10

**【0366】**

FREELISTロジック821はまた、オールゼロ検出器836も含んでいる。それは、FREELIST信号がオールゼロビットを含むことによりトラップスタック820内の記憶エントリはいずれも利用できないことを示す時点を決する。もしこの場合が生ずれば、それはNO\_ENTRY\_AVAIL信号を提示する。

20

**【0367】**

図45を参照すると、PSU 300によりトラップを受入れるべきことが決定され乍らもNO\_ENTRY\_AVAIL信号は利用できる記憶エレメントが存在しないことを示しているとき、それが出力したTAKE\_TRAP信号は、NO\_ENTRY\_AVAIL信号により記憶エレメントが利用可能であることが示されるまで、トラップを受入れるべきことを示さない。更に、PSU 300はISSUE\_KILL信号およびFETCH\_SHUTDOWN信号を発生し、ISU 200による命令の出力とBRB 59による命令のフェッチとを停止するが、これは、NO\_ENTRY\_AVAIL信号により記憶エントリが使用可能であることが示されるまで継続される。これによれば、直ちに明らかとなる如く、CPU 51を同期することによりチェックポイントが解除され且つトラップスタック820内の記憶エントリが回復されて再利用出来るようになる、という効果が得られる。

30

**【0368】**

しかし乍ら、NO\_ENTRY\_AVAIL信号が発せられないときにPSU 300は制御レジスタファイル800に対してTAKE\_TRAP信号を出力するが、該TAKE\_TRAP信号は前述の如く、トラップを受入れるべきことを示す信号とトラップのタイプを識別する信号とを含むものである。これに応じ、制御レジスタ・リード/ライト/更新ロジック816は、ASIレジスタ808、CWPレジスタ809およびPSTATEレジスタ805の内容を読み出してトラップスタックユニット815のトラップスタック820に供与する。

40

**【0369】**

これと同時に、トラップスタック820は、該トラップスタック820のTPCおよびTNPCフィールドに書込まれるべき現在のAPCおよびNAPC値を含むWR\_TPC\_TNPC信号を受信する。また、後述する様に、トラップスタック820は、利用できる様になったときに図8のFSR/CCRFRN 610内のXICCレジスタのXICCデータを受信する。

**【0370】**

TAKE\_TRAP信号に応じ、トラップスタックRD/WR制御ロジック822はTAKE\_TRAP信号により含まれたトラップタイプ(TT)フィールドを抽出すると共に、それをトラップスタック820に供与する。次に、RD/WRロジック822は、受信したAPC、NAPC、ASI、CWP、PS

50

TATEおよびTTの各値を、WR\_ENTRY信号により指示された記憶エレメントの適切なフィールド内に直ちに書込む。更に、後述の如く、XICCデータは同一の記憶エレメントのTSTATEフィールド内に書込まれるが、これは、論理的なXICCレジスタの内容が利用できるようになったときのWR\_ENTRY\_XICC信号に応じて行われる。

【 0 3 7 1 】

図 1 7 を参照すると、制御レジスタRD / WR / UPDATEロジック816は次にTLレジスタ811からの旧TL値を受信してそれをインクリメントする。それは次に新たなTL値をTLレジスタ811に書き戻す。

【 0 3 7 2 】

図 4 5 に示された如く、トラップ・スタック・リネーム・マッピング(RENAME MAP)・ロジック824 は、新たなTL値、および、TAKE\_TRAP およびWR\_ENTRY信号を受信する。図 4 7 を参照すると、RENAME MAPロジック824 は、夫々が4個のトラップレベルのひとつに対応するTL1-4(トラップレベル1-4)書込マルチプレクサ826 ~ 829 およびTL1-4 レジスタ830 ~ 833 を含んでいる。新たなTL値およびTL\_TRAP 信号に応じ、RENAME MAPロジック824 の制御回路825 はWR\_MUX\_CNTRL信号によりTL1-4 書込マルチプレクサ826 ~ 829 を制御し、これにより、WR\_ENTRY信号は新たなTL値により識別されるトラップレベルに対応するレジスタ内に記憶され、且つ、その他のレジスタは以前のマシンサイクルにおいて記憶したのと同じの値を再記憶する。

【 0 3 7 3 】

従って、レジスタのひとつに記憶されたWR\_ENTRY信号により識別される記憶エントリは、RENAME MAPロジック824 内の新たなTL値により識別される現在のトラップレベルにマッピングされる。更に、他のレジスタ内に記憶された信号により識別される記憶エントリは、以前のマシンサイクルにおけるのと同様にしてマッピングが維持される。

【 0 3 7 4 】

図 4 6 を参照すると、FREELIST 821のファーストワンエンコーダ823 はTAKE\_TRAP 信号も受信する。TAKE\_TRAP 信号がトラップを受入れるべきことを示すとき、ファーストワンエンコーダ823 はOR回路835 に対し、WR\_ENTRY信号に対応するビットが“0”とされた8ビット信号を送り返す。OR回路835 はまた、(図 4 5 に示された)RRF ロジック837 からの8ビットFREE\_ENTRY信号を受信するが、そのビットは、トラップスタック820 の記憶エントリの中で前回マシンサイクルで回復されて現在使用可能となった記憶エントリを識別するものである。バックアップの間、OR回路835 はAND 回路836 から8ビットのBACKUP\_FREELIST 信号を受信するが、該信号のビットは、バックアップされたチェックポイントが作られた時点において使用可能であった記憶エントリを識別するものである。ファーストワンエンコーダ823 から受信した信号のビットは、FREE\_ENTRYおよびBACKUP\_FREELIST 信号と論理和が取られており、且つ、最終的なFREE\_LIST 信号はレジスタ822 に供与される。従って、バックアップを除けば、次のマシンサイクルにおけるFREELIST信号は、WR\_ENTRY信号内で“0”に設定されたビットに対応する記憶エレメントはもう使用出来ないことを示している。

【 0 3 7 5 】

図 7 を再度参照すると、PSU 300 はTAKE\_TRAP 信号を生成すると同時に、CHKPT\_REQ 信号により、ISU 200 に対し、トラップ処理ルーチンから最初に発せられた命令に対してチェックポイントを割当ててることを要求する。これに応じ、ISU 200は、制御レジスタファイル800 を含め、CPU 51内の種々のブロックおよびユニットにDO\_CHKPT信号を出力する。

【 0 3 7 6 】

図 4 7 に戻り、RENAME\_MAP\_1-4信号はTL1-4 レジスタ830 ~ 833により出力されることから、夫々、トラップレベル1 ~ 4 に対応している。更に、これらの信号はTL1-4 レジスタ830 ~ 833 により出力されることから、それらはトラップレベル1 ~ 4 に対する記憶エントリの現在のマッピングを与えるものである。

【 0 3 7 7 】

しかし乍ら、前述の如く、トラップが受け入れられたとき、RENAME MAPロジック824 は新

10

20

30

40

50

たに利用し得る記憶エントリを新たなトラップレベル(TL)にマッピングする。但し、このトラップレベルに対する別の記憶エントリの旧マッピングは、前述の如くバックアップが生ずるときには記録しておかねばならない。従って、図48を参照すると、PRF ロジック837は、自身がRENAME MAPロジック824から受信したRENAME\_MAP\_1-4信号からの旧マッピングを、それらが不要となるまで保存しておく。

**【0378】**

トラップが受入れられたとき、マルチプレクサ843は、新たなトラップに対するTL(即ち、旧TLは1だけインクリメントされる)に対応するRENAME\_MAP信号を出力する。換言すると、このマルチプレクサは、次のマシンサイクルにおいて新たなマッピングを反映するRENAME\_MAP信号のみを選択する。この信号は次に、デコーダ842により、記憶エントリ内のいずれが新たなマッピングにより新たなトラップレベルに対して置換えられているのかを示す8ビット信号にデコードされる。ここで、TAKE\_TRAP信号はトラップが受入れられつつあることを示すことから、デコードされた信号は、RRF ロジック837のRRF記憶ユニット839に対してAND回路841を介して供与される。しかし乍ら、何らのトラップも受入れられていない場合、AND回路841はRRF記憶ユニット839に対して全てのビットが“0”に設定された8ビット信号を供与する。

**【0379】**

RRF記憶ユニット839は、16個の記憶エレメントすなわちエントリを含んでいる。各記憶エントリは、前述の16個のチェックポイント番号のひとつに対応している。従って、DO\_CHKPT信号によりチェックポイントが形成されるべきことが示されたとき、RRF RD/WR制御ロジック840はAND回路841から受信したデータを、DO\_CHKPT信号により識別されたチェックポイント番号に対応する記憶エントリ内に書込んでいる。

**【0380】**

従って、RRF 837は、トラップスタックユニット820の各記憶エントリの不稼働リストを保持するが、該記憶エントリは、RENAME MAPロジック824の現在のマッピングによりトラップレベルの内のひとつに現在マッピングされてはいないがトラップレベルのひとつにマッピングする為には依然として使用され得ないものである、と言うのも、トラップレベルに対するその旧マッピングはチェックポイントをバックアップする場合には再記録する必要があり得るからである。この場合、記憶エントリは不稼働リスト内に保持されるが、これは、形成されるべきチェックポイントを引起こしたトラップのトラップレベルに対して記憶エントリがマッピングされた時点において形成されたチェックポイントが解除されるまで続くことになる。

**【0381】**

RRF RD/WR制御ロジック840はまた、PSU 300から16ビットのCHKPT\_CLR信号も受信する。各ビットは、チェックポイント番号のひとつに対応すると共に、対応するチェックポイントが前述の手法で解除(即ちクリア)されたか否かを示している。この信号に応じ、PRF RD/WRロジックは、RRF記憶ユニット839の対応記憶エントリからデータを読み出す。CHKPT\_CLR信号はクリアされた多数のチェックポイントを識別し得ることから、対応する記憶エントリから読み出されたデータの各ビットは論理和が取られると共に結合されてFREE\_ENTRY信号を形成し、それは8ビットのFREE\_ENTRY信号として図46のFREELISTロジック821へ送られる。

**【0382】**

トラップが受入れられたときにCHKPT\_CLR信号により識別されたチェックポイントが形成された場合、FREE\_ENTRY信号により識別された記憶エントリは回復されて現在は再び使用可能となっている、と言うのも、対応するチェックポイントは解除されているからである。その結果、このトラップはもはや行い得ないものではない。従って、バックアップが生じなかったと仮定すれば、FREE\_ENTRY信号の各ビットは次に、ファーストワンエンコーダ823により供与された信号の対応ビットと論理和が取られ、次のマシンサイクルに対するFREELIST信号が提供される。

**【0383】**



また、トラップが受入れられたときにCHKPT\_CLR 信号により識別されたチェックポイントが形成されなかった場合、対応する記憶エレメントから読み出されたデータのビットは全て“0”であり、従って、FREELIST信号の生成に関して何らの影響を有さない。

【0384】

図45に示された様に、トラップスタックユニット815はまた、トラップ・スタック・チェックポイント記憶ユニット845も含んでいる。この記憶ユニット845は図48に更に詳細に示す様にデータ記憶構造846を含むが、該構造は、夫々が5個のフィールドを有する16個のアドレッシング可能な記憶エレメントすなわちエントリを有している。

【0385】

この記憶ユニットは、FREELISTロジック821からのFREELIST信号およびRENAME\_MAPロジック824からのRENAME\_MAP信号を受信する。而して、チェックポイントを形成すべきことがDO\_CHKPT信号により示された場合、トラップスタック記憶RD/WR制御ロジック847は、FREELIST信号およびRENAME\_MAP信号を、DO\_CHKPT信号により識別されたチェックポイント番号に対応する記憶エントリの適切なフィールド内に書込む。現在のFREELIST信号およびRENAME\_MAP信号はチェックポイントが形成されたときに記憶されていることから、トラップレベルに対するトラップ記憶エントリの現在のマッピング、および、利用し得るトラップスタック記憶エントリの現在のリストには、チェックポイントが割当てられている。

【0386】

チェックポイントに対するバックアップが為されつつあることがDO\_BACKUP信号により表されるとき、RD/WR制御ロジック847は、DO\_BACKUP信号により識別されたチェックポイント番号に対応する記憶エレメントから、チェックポイントが割当てられたFREELIST信号およびRENAME\_MAP信号を読み出す。この点、FREELIST信号はBACKUP\_FREELIST信号として読み出される一方、RENAME\_MAP信号はBACKUP\_MAP信号として読み出される。

【0387】

図46に示された如く、フリーリストロジック821はBACKUP\_FREELIST信号およびDO\_BACKUP信号を受信する。DO\_BACKUP信号はバックアップが生じていることを表すことから、AND回路838はBACKUP\_FREELISTをOR回路835に供与する。前述の如く、ファーストワンエンコーダ823から受信された信号の各ビットは、FREE\_ENTRY信号およびBACKUP\_FREELIST信号の対応ビットと論理和が取られ、FREELIST信号を生成している。

【0388】

図47を参照するに、RENAME\_MAPロジック824はBACKUP\_MAP信号およびDO\_BACKUP信号を受信する。DO\_BACKUP信号はバックアップが生じていることを表すことから、制御回路825が生成するWR\_MUX\_CNTRL信号は、TL1-4レジスタ830~833の夫々に対するBACKUP\_MAP\_1-4信号を供与する。その結果、チェックポイントが割当てられたときに存在したトラップレベルに対するトラップスタック記憶エレメントのマッピングは、元通りの状態にされている。

【0389】

前述の如く、CPU51は投機的に(speculatively)命令を発し且つ実行する。トラップスタックユニット815は、バックアップの場合に以前のトラップレベルを回復する機構を含んでいることから、CPU51はトラップを投機的に受取るとともにトラップから投機的に戻ることができる。

【0390】

トラップを投機的に受入れると共にトラップから投機的に戻るというCPUの能力は、図51に示されている。図示された如く、最初に、実行手順はトラップレベル0に在り、且つ、トラップレベル1~4はトラップスタックエントリ1~4へ夫々マッピングされている。最初のトラップが受入れられるとき、トラップレベルはトラップレベル1にインクリメントされると共に、チェックポイント1が形成され、チェックポイント1における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ(checkpointed)、且つ、トラップレベル1の新たなマッピングが記憶エントリ5に対して作られる。新たなチェックポイントはチェックポイント2で作られるが、これは予期されたプログラム

10

20

30

40

50

制御命令などの命令に対するものであり、且つ、チェックポイント2における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。第2のトラップが受入れられると共にトラップレベルはトラップレベル2にインクリメントされ、チェックポイント3が形成され、チェックポイント3における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ、且つ、トラップレベル2の新たなマッピングが記憶エントリ6に対して作られる。トラップレベル2におけるトラップに対するトラップ処理においてdoneもしくはretry命令が実行されたとき、トラップレベルはトラップレベル1にデクリメントされると共にチェックポイント4が形成され、かつ、チェックポイント4における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。トラップレベル1におけるトラップに対するトラップ処理に戻った後、このトラップハンドラの命令に対してチェックポイント5においてチェックポイントが作られ、且つ、チェックポイント5における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。更なるトラップが受入れられたとき、トラップレベルはトラップレベル2にインクリメントされ、チェックポイント6が形成され、チェックポイント6における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ、且つ、記憶エントリ7に対するトラップレベル2の新たなマッピングが作られる。そして更なるトラップが受入れられたとき、トラップレベルはトラップレベル3にインクリメントされると共にチェックポイント7が形成され、チェックポイント7における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ、且つ、記憶エントリ0へのトラップレベル3の新たなマッピングが作られる。トラップレベル3におけるトラップに対するトラップハンドラ内でdoneまたはretry命令が実行されたとき、トラップレベルはトラップレベル2にデクリメントされると共にチェックポイント8が形成され、且つ、チェックポイント8における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。その後、チェックポイント2が形成されたプログラム制御命令が誤って予測されたと判断されたのであれば、チェックポイント2へのバックアップが生じ、チェックポイント2における記憶エントリマッピングへのトラップレベルが回復される。従って、チェックポイント3および7が形成されたトラップは投機的に受入れ/戻りが行われ、一方、チェックポイント6が形成されたトラップは投機的に受入れられている。

10

20

**【0391】**

30

前述の如く、トラップスタックのTSTATEフィールドはXICCフィールドを含み、該XICCフィールドは、トラップが受入れられたとき、図8のFSR/CCRFRN 606内の論理的XICCレジスタとしてマッピングされた物理レジスタの内容(すなわちXICCデータ)を保持している。固定小数点命令はPC命令を受けずにFXU 601 およびFXAGU 602 により実行され得ることから、これらの内容はトラップが生じた時点で利用され得ない。従って、後の時点にて正しいXICCデータを得る機構が無ければ、PSU 300 はXICCデータがトラップを受入れるべく利用出来るようになるまでまたねばならない。

**【0392】**

しかし乍ら、図45を参照すると、トラップスタックユニット815 は斯かる機構、即ち、XICC捕捉ロジック823 を含んでいる。このXICC捕捉ロジック823 は、FSR/CCRFRN 606から、CC\_TAGS\_C、CC\_DV\_C、CC\_TAGS\_F およびCC\_DV\_F 信号を受信し、一方、トラップスタック820 はCC\_DATA\_C およびCC\_DATA\_F 信号を受信する。

40

**【0393】**

図52を参照すると、CC\_DATA\_C 信号はXICC\_DATA\_C 信号を含むと共に、CC\_DV\_C 信号はXICC\_DV\_C 信号を含んでいる。可能であれば、XICC\_DATA\_C 信号は、トラップが受入れられたときのマシンサイクルにおける論理XICCレジスタ(即ち、図39に示されたCCRFRN 610による論理XICCレジスタとしてマッピングされた物理レジスタ)の現在の内容を含んでいる。XICC\_DV\_C 信号は、XICC\_DATA\_C 信号の内容が有効か否か(即ち、論理XICCレジスタとして現在マッピングされている物理レジスタの内容が依然として利用し得るか否か)を表している。

50

## 【0394】

トラップが受入れられた時点において論理XICCレジスタとしてマッピングされた物理レジスタの内容は、その時点で既に利用し得る可能性もある。これが生じた場合、XICC\_DV\_C信号は、XICC\_DATA\_C信号の内容が有効であることを示し、且つ、PSU 300から受入れたTAKE\_TRAP信号は、トラップが受入れられつつあることを示す。これに応じ、XICC書込ロジック825は、WR\_ENTRY信号により識別されたエントリに対応するWR1\_ENTRY\_XICC信号を提供する。これが生じた場合、トラップスタックユニット815のRD/WR制御ロジック822は、XICC\_DATA\_C信号の内容を、WR1\_ENTRY\_XICC信号により識別されたトラップスタック820のエントリのTSTATEフィールド内に書込む。

## 【0395】

しかし乍ら、XICC\_DV\_C信号が、トラップが受入れられた時点のXICC\_DATA\_C信号の内容が有効でないことを示すときは、XICC\_DATA\_C信号の内容はトラップスタック820に書込まれない。この場合、トラップスタックユニット815は、トラップが受入れられた時点で論理XICCレジスタとしてマッピングされた物理レジスタの内容が利用できるようになるまで待たねばならない。図53を参照すると、これを行うために、XICC捕捉ロジック823は、現在整合ロジック826および後期整合配列ロジック827を含んでいる。

## 【0396】

依然として図53を参照するに、CC\_TAG\_C信号はXICC\_TAG\_C信号を含むが、該XICC\_TAG\_C信号は、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの物理レジスタ・タグを含んでいる。更に、CC\_TAG\_F信号は、FXU 601およびFXAGU 602により夫々発信されたFXU\_XICC\_TAG\_F信号およびFXAGU\_XICC\_TAG\_F信号を含むが、これは、FXU 601およびFXAGU 602が、論理XICCレジスタを変更する固定小数点命令を実行するときに行われる。これらの信号は、実行されたこれらの固定小数点命令に対するものであると共にCC\_DATA\_F信号のFXU\_XICC\_DATA\_F信号およびFXAGU\_XICC\_DATA\_F信号が書込まれるべき論理XICCレジスタとしてマッピングされた物理レジスタの物理レジスタ・タグを含むものである。FXU\_XICC\_DATA\_F信号およびFXAGU\_XICC\_DATA\_F信号は夫々、FXU 601およびFXAGU 602により発信されたものである。

## 【0397】

CC\_DV\_F信号は、論理XICCレジスタを変更する固定小数点命令をFXU 601およびFXAGU 602が実行するとき該FXU 601およびFXAGU 602により夫々発信されたFXU\_XICC\_TAG\_F信号およびFXAGU\_XICC\_TAG\_F信号を含んでいる。これらの信号は、実行された固定小数点命令に対する論理XICCレジスタとしてマッピングされた物理レジスタに対してFXU\_XICC\_DATA\_F信号およびFXAGU\_XICC\_DATA\_F信号の内容が有効か否か（即ち、利用できるか否か）を表している。

## 【0398】

既述の如く、実際のおよび/または予期されたプログラム命令に基づいて命令は実行され得ることから、発信されたFXU\_XICC\_DATA\_F信号およびFXAGU\_XICC\_DATA\_F信号は最終的に、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタに書込まれるべき内容を含むことになる。これは、トラップが受入れられるのと同じのマシンサイクル（即ち、現在のマシンサイクル）もしくは後期のマシンサイクルにて生ずる。

## 【0399】

これが同一のマシンサイクルで生じる場合、XICC\_TAG\_C信号は、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタのタグを現在の的に識別する。XICC\_TAG\_C信号により識別されたタグは、現在整合ロジック826の比較ロジック850および比較ロジック851により、FXU\_XICC\_TAG\_F信号およびFXAGU\_XICC\_TAG\_F信号と比較される。もし、整合が生じ、且つ、対応するFXU\_XICC\_DV\_F信号またはFXAGU\_XICC\_DV\_F信号が、対応するFXU\_XICC\_DATA\_F信号またはFXAGU\_XICC\_DATA\_F信号の内容が有効であることを示すのであれば、現在整合ロジック826は、対応するFXU\_XICC\_CURR\_MATCH信号またはFXAGU\_XICC\_CURR\_MATCH信号を出力する。この信号は、トラップが受入れられたと

10

20

30

40

50

きに論理XICCレジスタとしてマッピングされた物理レジスタの内容が、トラップが受け入れられたときと同一のマシサイクル内で利用し得る様になったことを表す。

#### 【0400】

図52に戻り、このマシサイクルの間、TAKE\_TRAP信号は依然としてトラップが受け入れられつつあることを表し、且つ、WR\_ENTRY信号は依然としてトラップのトラップレベルにマッピングされたエントリを識別している。従って、FXU\_XICC\_CURR\_MATCH信号またはFXAGU\_XICC\_CURR\_MATCH信号に応じ、XICC書込ロジック825は対応するWR2\_ENTRY\_XICC信号またはWR3\_ENTRY\_XICC信号を生成する。この信号は、WR\_ENTRY信号により識別されたエントリに対応する。次に、トラップスタックユニット815のRD/WR制御ロジック822は、FXU\_XICC\_DATA\_F信号またはFXAGU\_XICC\_DATA\_F信号の内容を、対応するWR2\_ENTRY\_XICC信号またはWR3\_ENTRY\_XICC信号により識別されるトラップスタック820のエントリのTSTATEフィールドのXICCフィールド内に、書込む。

10

#### 【0401】

図53に戻ると、既述の如く、トラップが受け入れられたときのマシサイクルの後のマシサイクルにおいて、発信されたFXU\_XICC\_DATA\_F信号およびFXAGU\_XICC\_DATA\_F信号は、トラップが受け入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタに書込まれるべき内容を含んでいる可能性がある。この状況に備えるべく、XICC捕捉ロジック823は、タグ配列記憶ユニット854を含むタグ配列記憶ユニット827を含んでいる。

#### 【0402】

記憶ユニット854は、8個の記憶エントリを含んでおり、各エントリは、トラップスタック820の各エントリのひとつに対応している。従って、トラップが受け入れられつつあることがTAKE\_TRAP信号により示される都度、RD/WRロジック855は、そのときのXICC\_TAG\_C信号により識別されるタグを、WR\_ENTRY信号により識別されるトラップスタック820のエントリに対応する記憶ユニット854のエントリに、書込む。

20

#### 【0403】

比較配列ロジック852および853は、夫々、FXU\_XICC\_TAG\_F信号およびFXAGU\_XICC\_TAG\_F信号により識別されるタグを、記憶ユニット854内に記憶されたタグの各々と、比較する。その結果、もし、整合が生じ、且つ、対応するFXU\_XICC\_DATA\_F信号またはFXAGU\_XICC\_DATA\_F信号の内容が有効であることが対応するFXU\_XICC\_DV\_F信号またはFXAGU\_XICC\_DV\_F信号により表されたのであれば、対応する比較配列ロジック852または853は、対応するFXU\_XICC\_DATA\_F信号またはFXAGU\_XICC\_DATA\_F信号の内容が書込まれるべきトラップスタック820のエントリを識別する対応FXU\_XICC\_LATE\_MATCH信号または対応FXAGU\_XICC\_LATE\_MATCH信号を、出力する。

30

#### 【0404】

図54に戻ると、XICC捕捉ロジックは、XICC用待機ロジック828を含んでいる。又、レジスタ856はWAIT\_FOR\_XICC\_VEC信号を記憶する。このWAIT\_FOR\_XICC\_VEC信号の各ビットは、トラップスタック820の各エントリのひとつに対応すると共に、対応するトラップが受け入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が利用可能になるのを対応エントリが待機しているときに提示される。従って、WAIT\_FOR\_XICC\_VECベクトル信号は、対応するトラップが受け入れられたときに論理XICCレジスタとしてマッピングされた種々の物理レジスタの内容が利用可能になるのを現在待っているトラップスタック820の各エントリの全てのリストを提供するものである。

40

#### 【0405】

TAKE\_TRAP信号により表されるトラップが受け入れられると共に、XICC\_DV\_C信号が“論理XICCレジスタとしてマッピングされた物理レジスタの内容を利用することができない”ことを示し、且つ、FXU\_XICC\_CURR\_MATCH信号およびFXAGU\_XICC\_CURR\_MATCH信号のいずれもまたこれを示さない、という状態が生じる都度、WR\_ENTRY信号は、後期のマシサイクルにおいてこれらの内容が利用し得る様になるのを待っているエントリを、識別する。これに応じ、このエントリは、XICC用待機ロジック828により、対応トラップが受け入れら

50

れたときに論理XICCレジスタとしてマッピングされた種々の物理レジスタの内容が利用し得るようになるのを現在待っているトラップスタック820のエントリのリストであってWAIT\_FOR\_XICC\_VECベクトル信号により与えられるリスト、に対して加えられる。

【0406】

しかし乍ら、前述の如く、TAKE\_TRAP信号により示されるごとくトラップが受入れられたとき、論理XICCレジスタとしてマッピングされた物理レジスタがこの時点で利用し得ることがXICC\_DV\_C信号により示されるか、あるいは、FXU\_XICC\_CURR\_MATCH信号またはFXAGU\_XICC\_CURR\_MATCH信号がこのことを示す可能性もある。これが生じる都度、WR\_ENTRY信号により識別されるエントリはXICCロジック828により、WAIT\_FOR\_XICC\_VECベクトル信号により与えられたリストに加えられない。

10

【0407】

更に、前述の如く、対応するトラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が後期のマシサイクルにおいて利用し得る様になったときは常に、FXU\_XICC\_LATE\_MATCH信号またはFXAGU\_XICC\_LATE\_MATCH信号は、これらの内容が書込まれるべきトラップスタック820のエントリを識別する。図52に戻ると、この時点でWAIT\_FOR\_XICC\_VECベクトル信号は依然としてこのエントリをリストすることから、XICC書込ロジック825は、このエントリを識別する対応WR2\_ENTRY\_XICC信号または対応WR3\_ENTRY\_XICC信号を生成する。その後、トラップスタックユニット815のRD/WR制御ロジック822は、FXU\_XICC\_DATA\_F信号またはFXAGU\_XICC\_DATA\_F信号の内容(即ち、対応トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容)を、対応するWR2\_ENTRY\_XICC信号またはWR3\_ENTRY\_XICC信号により識別されたトラップスタック820のエントリのTSTATEフィールドのXICCフィールド内に、書込む。次に、図54に戻ると、このエントリは、WAIT\_FOR\_XICC\_VECベクトル信号により与えられたリストから、XICC待機用ロジック828により除去される。

20

【0408】

従って、受入れられ乍らも未だ戻っていないトラップの各々に対し、XICC捕捉ロジックは、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が利用し得るようになる時点、および、トラップスタック820内のどのエントリ内にこれらの内容が書込まれるべきなのかを、決定することが出来る。その結果、これらの内容はいずれかのトラップが受入れられる時点では未だ利用できない可能性があるとしても、ネストしたトラップを受入れることが可能となる。換言すると、XICC捕捉ロジック823は、対応トラップステート(TSTATE)データの全てが利用可能になる以前にトラップが受入れられることを許容する機構である。

30

【0409】

しかし乍ら、トラップ受取から戻ると同時に、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容は、トラップスタック820の対応エントリ内に既に書込まれていなければならない。これは、RETRYもしくはDONE命令を行うためには、これらの内容が、現在において論理XICCレジスタとしてマッピングされた物理レジスタ内に書き戻されるべきことが必要である、という事実に基づいたものである。

40

【0410】

図54に示される如く、これが生ずるのを確実にすべく、XICC用待機ロジック828はマルチプレクサ857を含んでいる。現在受入れられるつつあるトラップのトラップレベルに現在マッピングされているトラップスタック820のエントリを識別するRD\_ENTRY信号に同じ、上記マルチプレクサは、WAIT\_FOR\_XICC\_VECベクトル信号のビットの内でのこのエントリに対応するビットを、WAIT\_FOR\_XICC信号として出力する。従って、この信号は、現在受入れられつつあるトラップのトラップレベルに現在マッピングされているトラップスタック820のエントリが、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容を待っているか否か、を表す。

【0411】

図45を参照すると、XICC捕捉ロジック823はISU 200に対してWAIT\_FOR\_XICC信号を

50

出力する。もしこの信号が提示されれば、ISU 200 は、この信号が提示されなくなるまで、RETRY 命令またはDONE命令を発しない。

【0412】

ISU 200 が最後にRETRY 命令またはDONE命令を発したとき、該ISU 200 は、DONE命令が発せられたか否かを示す信号とRETRY 命令が発せられたか否かを示す信号とを含むDONE\_RETRY\_IS 信号を、出力する。

【0413】

図47を参照すると、読込マルチプレクサ835 はRD\_ENTRY信号として、現在のTL値に対応するRENAME\_MAP信号を出力する。更に図45を参照すると、トラップスタックユニット815 のRD/WR 制御ロジック822 は、DONE命令またはRETRY 命令が既に発せられたことがDONE\_RETRY\_IS 信号により示されたときRD\_ENTRY信号により識別される記憶エレメントから、TPC、TNPC、ASI、CWP、PSTATEおよびXICCの各値を、読み出す。

10

【0414】

図17を参照すると、ASI、CWP およびPSTATEの各値は、DONE\_RETRY\_IS 信号に応じ、制御レジスタファイル800 のRD/WR/UPDATEロジック816 により対応レジスタ内に書込まれる。

【0415】

更に、図45に示される様に、TPC およびTNPCの各値はRD\_TPC\_\_TNPC信号を以てBRB 59 に付与される。その結果、BRB 59はこの値を使用してPCおよびNPC の各値を形成して、次の命令セットをフェッチする。

20

【0416】

最後に、XICC値はRD\_XICC 信号を以てDFB 62に供与される。図8を参照すると、DONE\_RETRY\_IS 信号に応じ、CCRFRN 610は論理XICCレジスタを物理レジスタにマッピングする。従って、CCRFRN 610はトラップスタックユニット815 からXICCデータを受けたとき、それを新たにマッピングされた物理レジスタ内に記憶する。

【0417】

更に、当業者であれば、FXU 601 およびFXAGU 602 の各々が、論理XICCレジスタを変更する固定小数点命令を実行し得る1個以上の実行ユニットを有し得ることを理解し得よう。この場合、XICC捕捉ロジック823 は、トラップが受入れられるときに論理XICCレジスタとしてマッピングされた物理レジスタの内容を以てトラップスタック820 を適切に更新すべく既述したロジック、の複製ロジックを有することになる。

30

【0418】

これに加え、当業者であれば、XICC捕捉ロジック823 の概念を他の型式のレジスタまで拡張し得ることを理解し得ようが、斯かる他の型式のレジスタとは、その内容が、トラップが受入れられた時点では利用する必要は無いがトラップスタックには書込まれる必要があるレジスタ、および、レジスタ名称変更(即ち、論理レジスタに対する物理レジスタのマッピング)が既に行われたレジスタ等である。

【0419】

本明細書にて言及した公報および特許出願の全てを参照により援用するが、これは、個々の公報もしくは特許出願を特定的にかつ個別的に参照により援用したのと同程度の程度まで援用する。

40

【0420】

本発明を十分に説明してきたが、当業者であれば、添付の請求の範囲の精神および範囲から逸脱すること無しに多くの変更および修正を為し得ることは理解し得よう。

【図面の簡単な説明】

【0421】

【図1】図1は、投機的プロセッサの精確状態を維持するために再順序付けバッファを用いる従来のアプローチを示す。

【図2】図2は、命令を実行する前にチェックポイントに状態を記憶し、後に記憶したチェックポイント情報から状態を復元することによって、投機的プロセッサに精確状態を維

50

持する従来のアプローチを示す。

【図3】図3は、図解のために略図で命令の例示シーケンスと従来のチェックポイントの構造と内容を示すもので、特定のマシンの実際のチェックポイントデータを表すものではない。

【図4】図4は、中央処理装置を含む本発明のデータプロセッサの実施例のトップレベル機能ブロック図である。

【図5】図5は、一部典型的命令タイプに関するCPUの新規命令パイプラインの実施例の段階である。

【図6】図6は、命令のプリフェッチとキャッシュユニットおよびフェッチユニットを含む本発明のCPUの例示分岐ブロック(BRB)を示す。

【図7】図7は、発行ユニット(ISU)、精確状態ユニット(PSU)、フリーリストユニット、制御レジスタファイルを含む例示発行ブロック(ISB)を示す。

【図8】図8は、発行ユニットがレジスタリネーミングフリーリストユニットから受取り、本発明のレジスタリネーミングの実施に用いる例示信号を示す。

【図9】図9は、精確状態ユニット(PSU)の実施例を示す。

【図10】図10は、発行/コミット/リクレームユニット(ICRU)の実施例のコンポーネントと、その動作に関連する入出力信号の機能ブロック図である。

【図11】図11は、命令状態情報とデータ構造に記憶される第1の例示データセットに関連する各種ポインタを記憶するためのアクティブな命令データ構造とメモリ命令データ構造の実施例の略図である。

【図12】図12は、A-リングビットセット/クリアロジックの一部構造的コンポーネントを含む例示的状态制御ロジックユニットを示す。

【図13】図13は、状態を追跡し精確状態を維持するため命令タグを使用する本発明の方法の実施例のフローチャートである。

【図14】図14は、精確状態の維持に命令状態を追跡する本発明の方法の実施例の模式的フローチャートである。

【図15A】図15は、精確状態の維持に命令状態を追跡する本発明の方法の実施例に従って、アクティブな命令リングとメモリ命令リングに状態情報を書き込む方法のフローチャート図である。

【図15B】図15は、精確状態の維持に命令状態を追跡する本発明の方法の実施例に従って、アクティブな命令リングとメモリ命令リングに状態情報を書き込む方法のフローチャート図である。

【図16】図16は、フィジカルレジスタを含むフィジカルレジスタファイルの入ったレジスタファイルおよびリネームユニット(RFRN)の実施例を示す。

【図17】図17は、レジスタリネーミングと関連する制御レジスタファイルの実施例を示す。

【図18】図18は、リソースリクレームファイル(RRF)の実施例を示す。

【図19】図19は、本発明のレジスタリネーミング方法を示す。

【図20】図20は、命令発行、非起動、コミットおよびリタイアを含む精確状態の維持のための本発明の方法の実施例のフローチャートである。

【図21】図21は、命令状態情報とデータ構造に記憶された第2の例示データセットに関連する各種ポインタを記憶するためのアクティブ命令データ構造およびメモリ命令データ構造の実施例の略図である。

【図22】図22は、CPUのデータフォワードブック内のロード/ストアユニット(LSU)の機能ブロック図である。

【図23】図23は、精確状態を維持しながら、ロード/ストア命令を含む長待ち時間命令を積極的にスケジューリングするための本発明の方法の実施例のフローチャートである。

【図24】図24は、チェックポイントユニットの機能ブロック図である。

【図25A】図25は、ウォッチポイントユニットの機能ブロック図である。

10

20

30

40

50

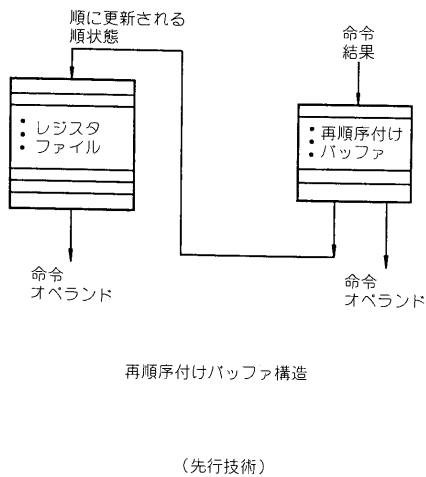
- 【図 2 5 B】図 2 5 は、ウォッチポイントユニットの機能ブロック図である。
- 【図 2 6】図 2 6 は、タイムアウトチェックポイントを形成するためのオプションのチェックポイント強化を含む本発明のチェックポイント方法の実施例のフローチャートである。
- 【図 2 7】図 2 7 は、ウォッチポイントユニットの実施例内の主要機能ブロックの機能ブロック図である。
- 【図 2 8】図 2 8 は、条件コードグラビングロジック、評価ロジック、ウォッチポイント要素記憶および制御ユニット、およびターゲット R A M の特定の実施例を示すウォッチポイントユニットの実施例である。
- 【図 2 9】図 2 9 は、カレントマッチングロジック、アレーマッチングロジック、および評価レディおよび評価条件コードロジックを含む条件コードセレクトロジックの入ったウォッチポイントの実施例の一部を示す。 10
- 【図 3 0】図 3 0 は、分岐予測および評価に関連する例示タイミングチャートを示す。
- 【図 3 1】図 3 1 は、カレントマッチロジックおよびアレーマッチロジックの実施例に関連する構造的詳細を示す。
- 【図 3 2 A】図 3 2 は、条件コードセレクトロジックの実施例に関連する構造的詳細を示す。
- 【図 3 2 B 1】図 3 2 は、条件コードセレクトロジックの実施例に関連する構造的詳細を示す。
- 【図 3 2 B 2】図 3 2 は、条件コードセレクトロジックの実施例に関連する構造的詳細を示す。 20
- 【図 3 3】図 3 3 は、評価レディロジックの実施例に関連する構造的詳細を示す。
- 【図 3 4】図 3 4 は、評価ツールロジックの実施例に関連する構造的詳細を示す。
- 【図 3 5】図 3 5 は、ターゲット R A M およびジャンプ・リンク命令評価ロジックの実施例に関連する構造的詳細を示す。
- 【図 3 6 A】図 3 6 は、複数同時未解決分岐評価のための本発明のウォッチポイント方法の実施例のフローチャートである。
- 【図 3 6 B】図 3 6 は、複数同時未解決分岐評価のための本発明のウォッチポイント方法の実施例のフローチャートである。
- 【図 3 7】図 3 7 は、浮動小数点例外 ( F P E X C E P ) データ構造の実施例の略図である。 30
- 【図 3 8】図 3 8 は、例示浮動小数点例外リングデータ構造の信号インターフェースを示す。
- 【図 3 9】図 3 9 は、R D および A E X C ロジックを含む浮動小数点例外ユニットを示す。
- 【図 4 0】図 4 0 は、バックアップおよびバックステップコンポーネントを含むバックトラックユニットの実施例の図である。
- 【図 4 1】図 4 1 は、バックアップおよびバックステップ手順を含む命令境界で精確状態を維持および回復するための本発明の方法の実施例のフローチャートである。
- 【図 4 2 A】図 4 2 A は、レジスタリネームファイルにロジカルからフィジカルマッピングを復旧する方法を示す。 40
- 【図 4 2 B】図 4 2 B は、レジスタリネームファイルにロジカルからフィジカルマッピングを復旧する方法を示す。
- 【図 4 3】図 4 3 は、チェックポイント境界への 1 つのマシンバックアップの後の命令境界への 2 つのマシンバックステップを含むマシンバックトラックの例である。
- 【図 4 4】図 4 4 は、例示プライオリティロジックおよび状態マシン ( P L S M ) を示す。
- 【図 4 5】図 4 5 は、例示トラップスタックユニットを示す。
- 【図 4 6】図 4 6 は、トラップスタックの記憶要素リストを記憶するためのフリーリストロジックの実施例を示す。 50



- 【図47】図47は、例示リネームマッピングロジックを示す。
- 【図48】図48は、例示トラップスタックRRFを示す。
- 【図49】図49は、トラップスタックチェックポイント記憶ユニットを示す。
- 【図50】図50は、ウォッチポイントWP\_\_ACTIVE\_\_VECおよびWP\_\_MISPREDロジックを示す。
- 【図51】図51は、トラップスタックバックアップ例を示す。
- 【図52】図52は、XICC書き込みロジックを示す。
- 【図53】図53は、XICCグラビングロジックおよびアレーレーターマッチロジックを示す。
- 【図54】図54は、ウェイト・フォー条件コード待ちロジックを示す。

【図1】

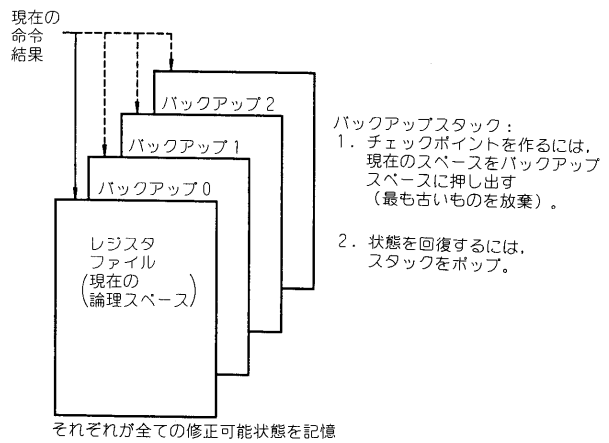
図1



【図2】

図2

チェックポイントの時点で利用不可能な結果は、適切なバックアップスペースに書き込まれ、順状態を完了



- バックアップスタック：
1. チェックポイントを作るには、現在のスペースをバックアップスペースに押し出す（最も古いものを放棄）。
  2. 状態を回復するには、スタックをポップ。

状態のチェックポイント記憶と復旧

(先行技術)

【 図 3 】

従来のチェックポイントレジスタスキームでのチェックポイント状態

図3

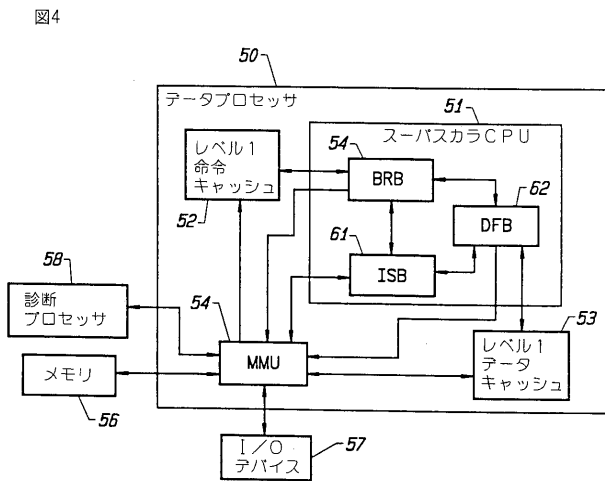
命令Aによって修正(mod) されうる状態:									
mod	-	-	-	mod	-	-	-	-	-
命令Bによって修正(mod) されうる状態:									
mod	-	-	-	-	-	-	-	-	-
命令Cによって修正(mod) されうる状態:									
-	mod	mod	-	-	mod	mod	mod	-	mod
命令Dによって修正(mod) されうる状態:									
-	-	-	-	-	-	-	mod	-	-

チェックポイントされた状態は、命令セットにおける命令のいずれか1つによって修正されうる全ての状態からなる:

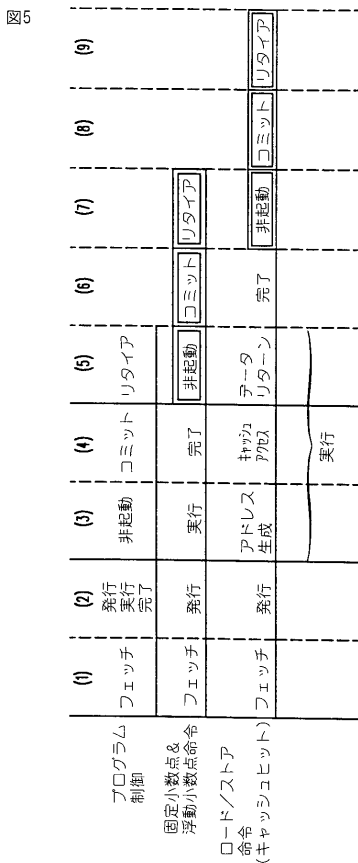
状態1	状態2	状態3	状態4	状態5	状態6	状態7	状態8	状態9	状態10
-----	-----	-----	-----	-----	-----	-----	-----	-----	------

従来のチェックポイントレジスタは、実行可能な命令の内のいずれか1つによって修正される全ての状態を記憶する。

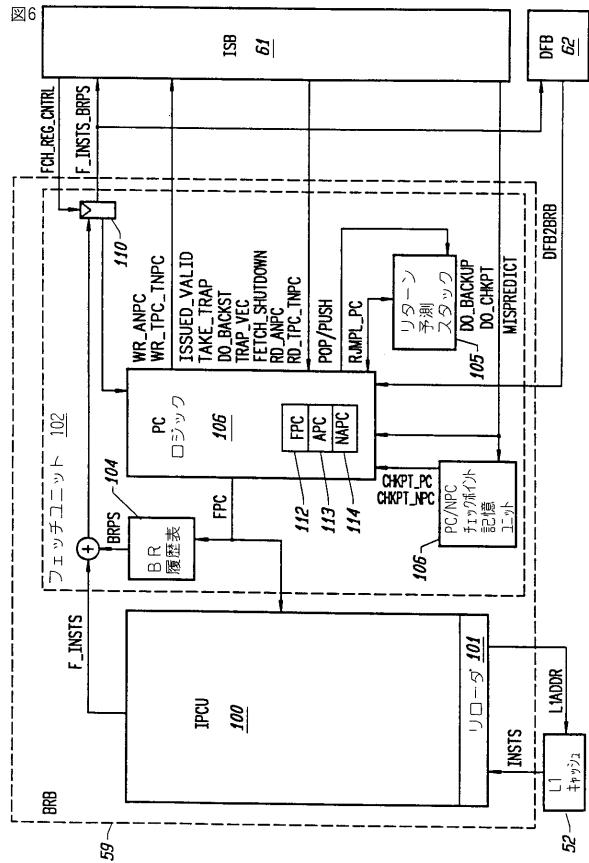
【 図 4 】



【 図 5 】



【 図 6 】

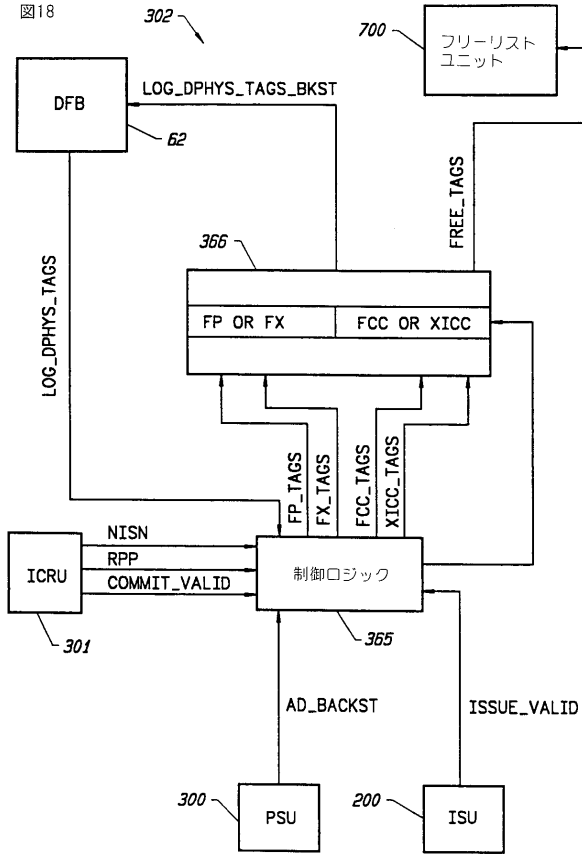




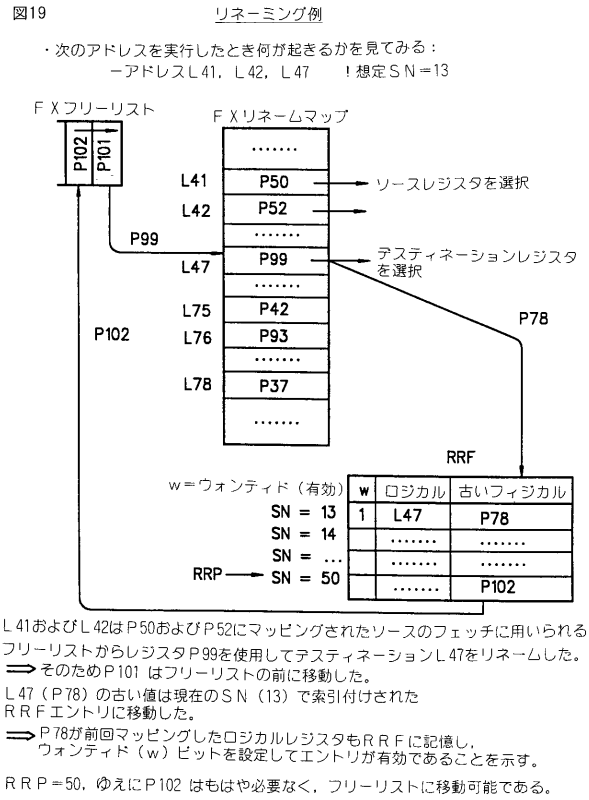




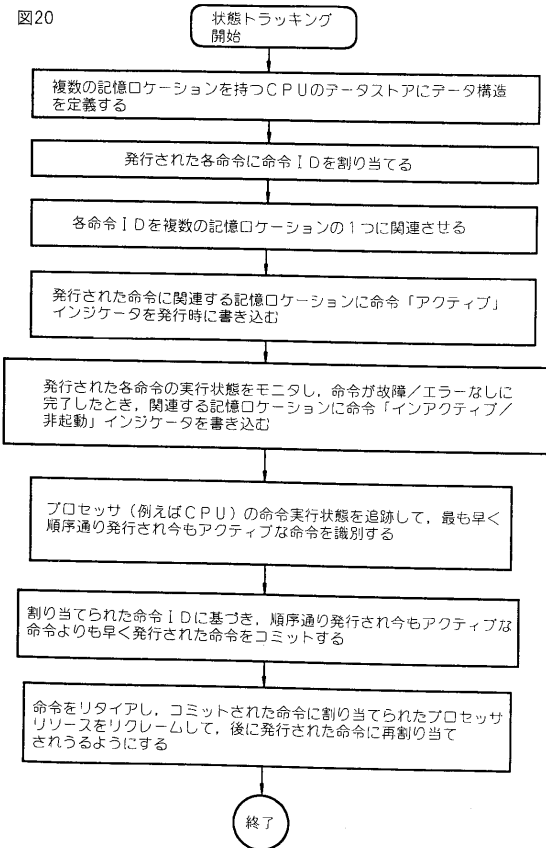
【 図 1 8 】



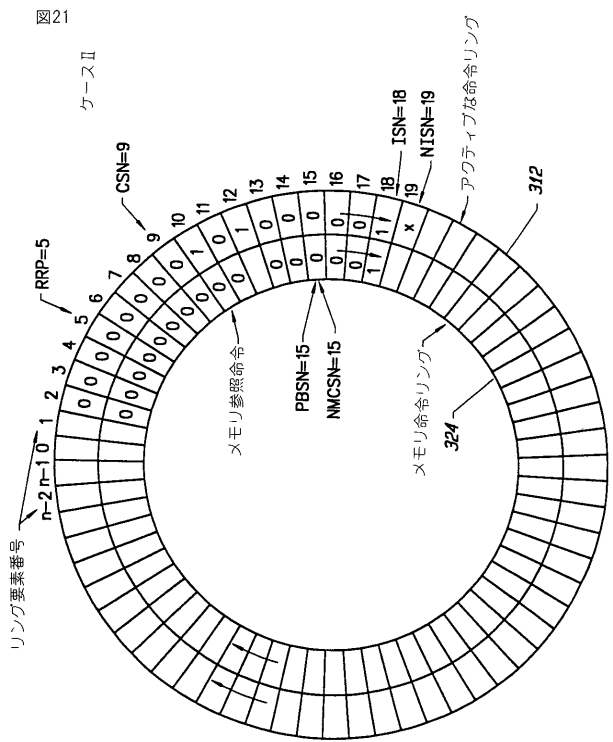
【 図 1 9 】



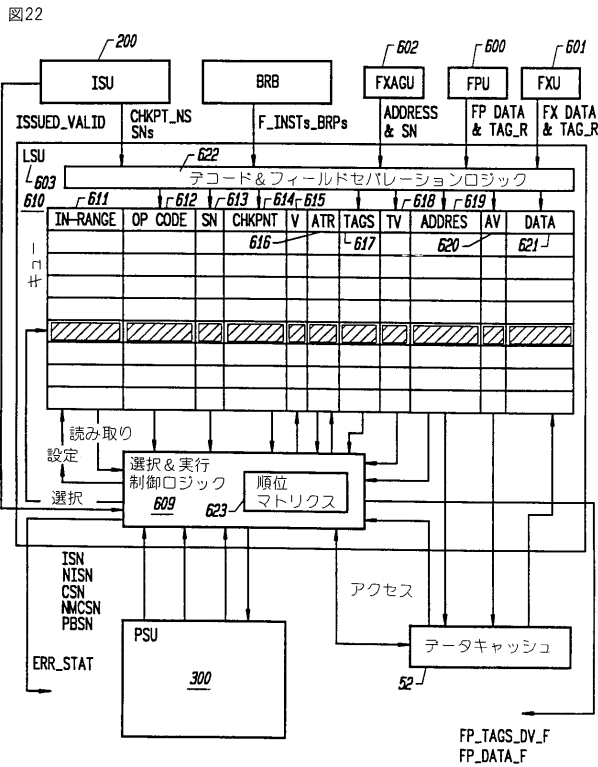
【 図 2 0 】



【 図 2 1 】

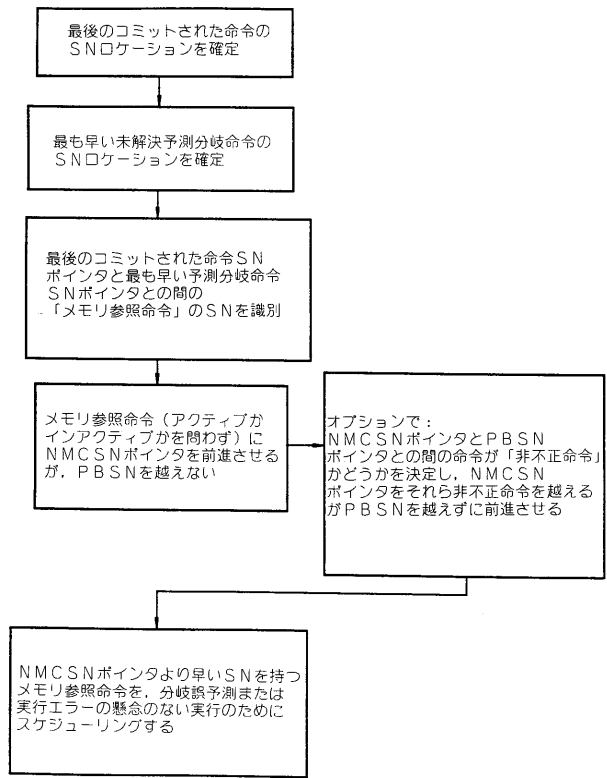


【 図 2 2 】



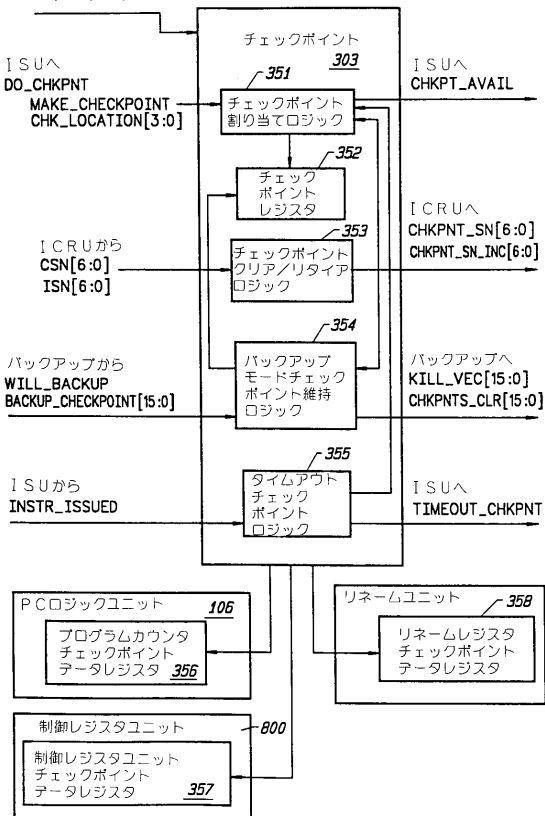
【 図 2 3 】

図23 ロード/ストア命令の積極的スケジューリング



【 図 2 4 】

図24 一般の CLK, SC, SI, SE, SO



【 図 2 5 A 】

図25A

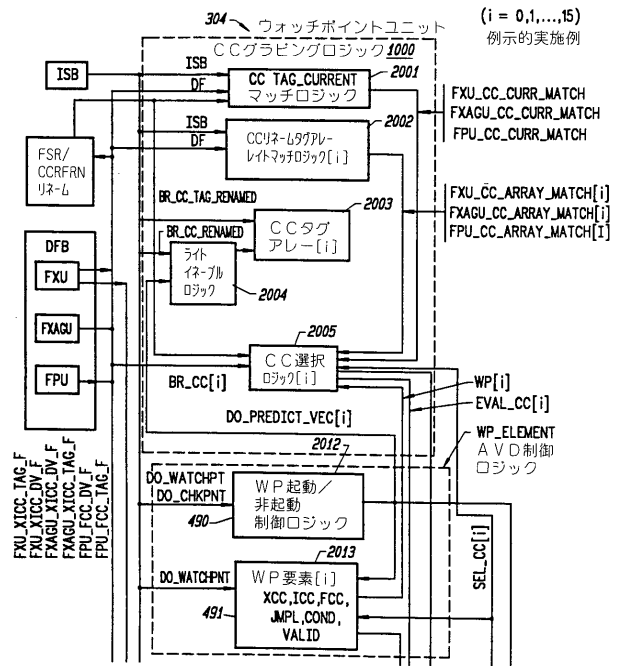
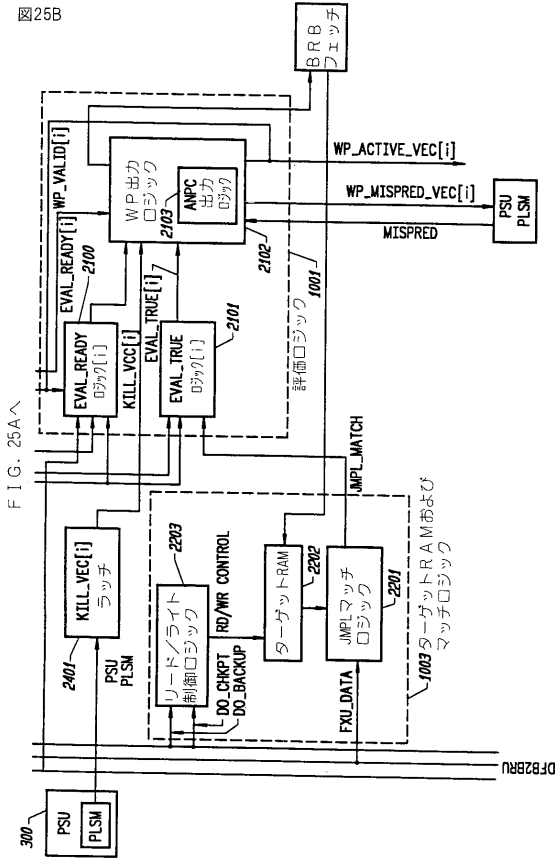


FIG. 25Bへ

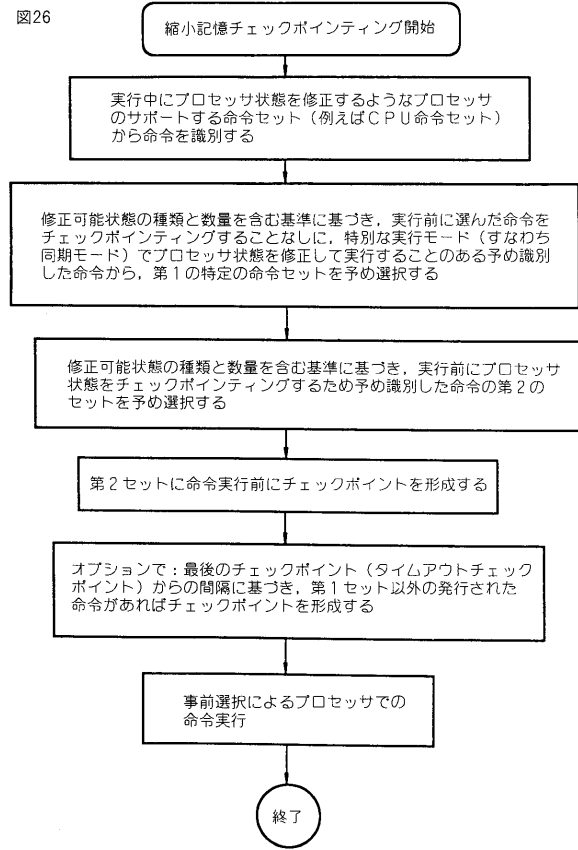
【 図 2 5 B 】

図25B



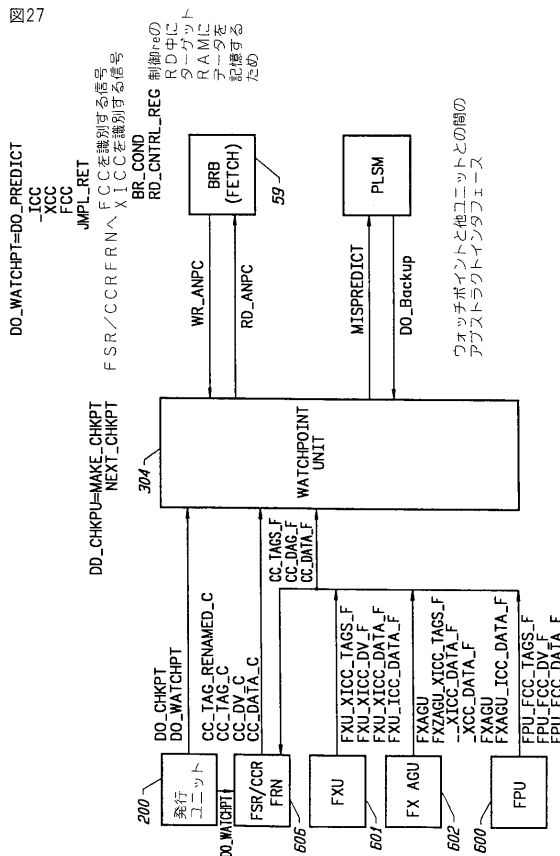
【 図 2 6 】

図26



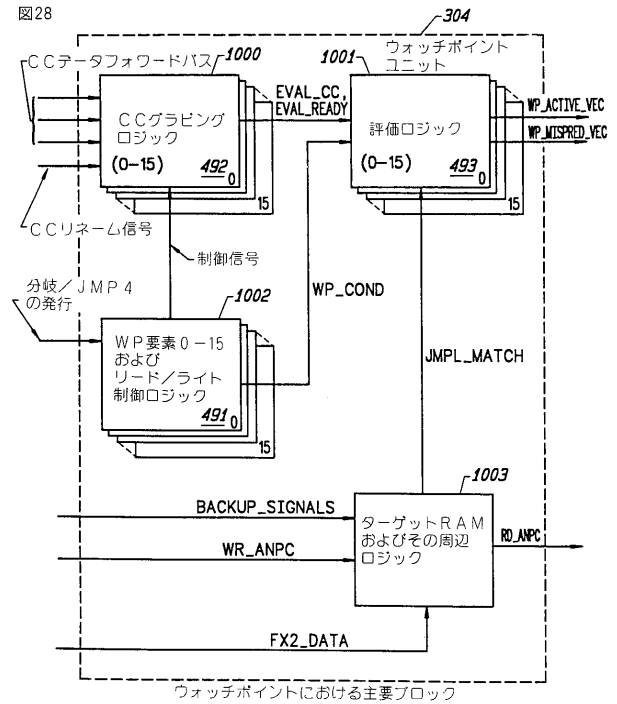
【 図 2 7 】

図27



【 図 2 8 】

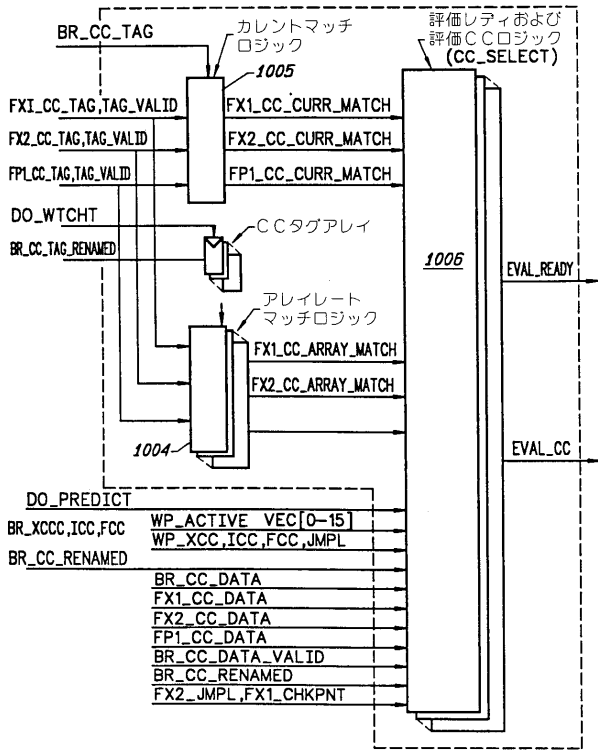
図28





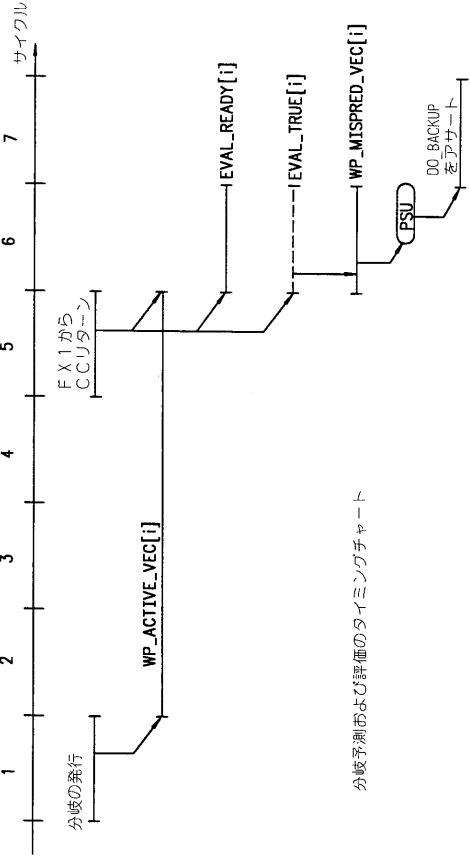
【 図 2 9 】

図29



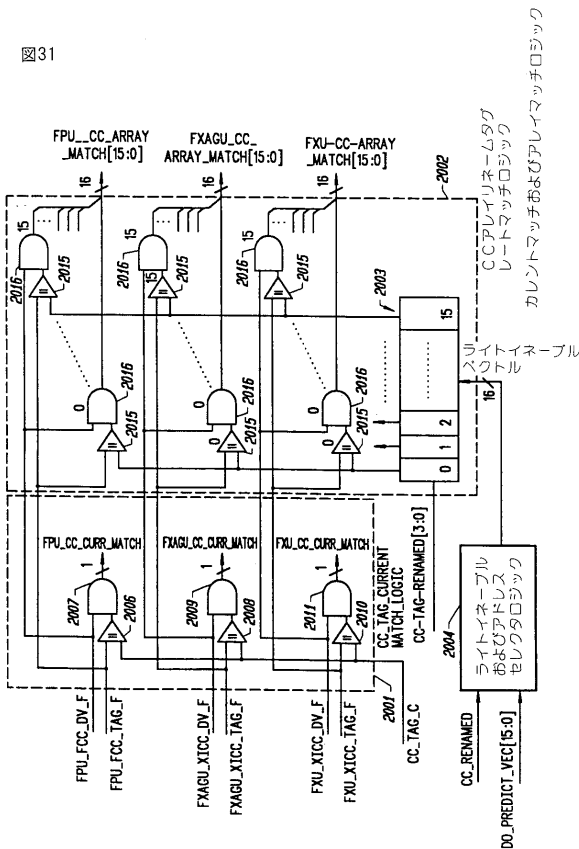
【 図 3 0 】

図30



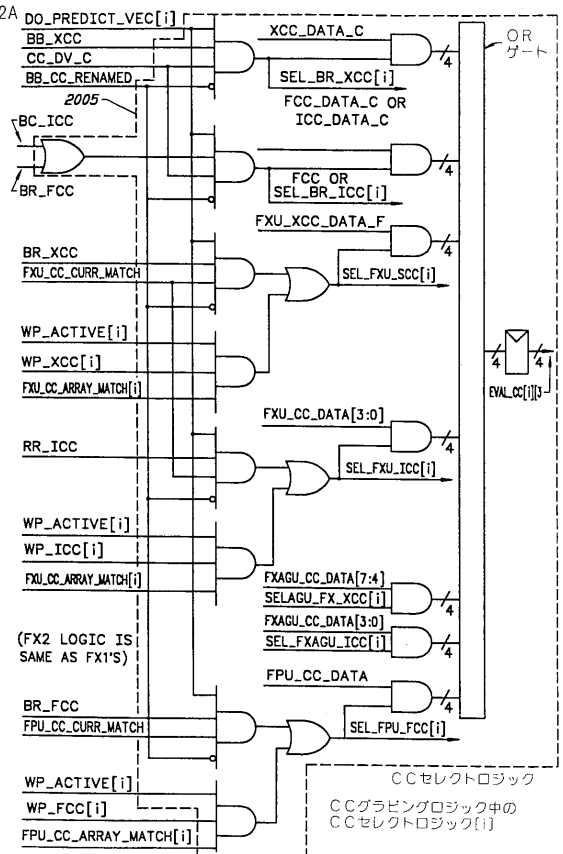
【 図 3 1 】

図31

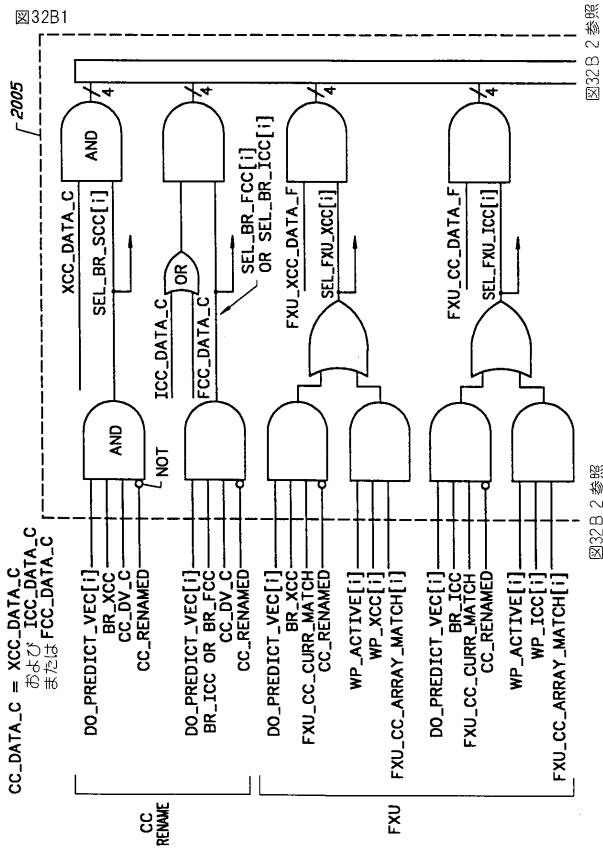


【 図 3 2 A 】

図32A



【 図 3 2 B 1 】



【 図 3 2 B 2 】

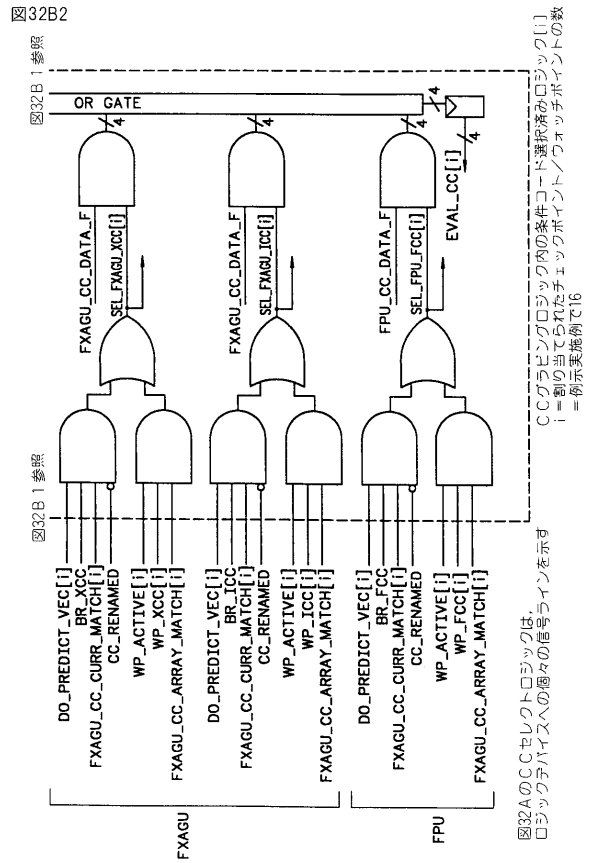
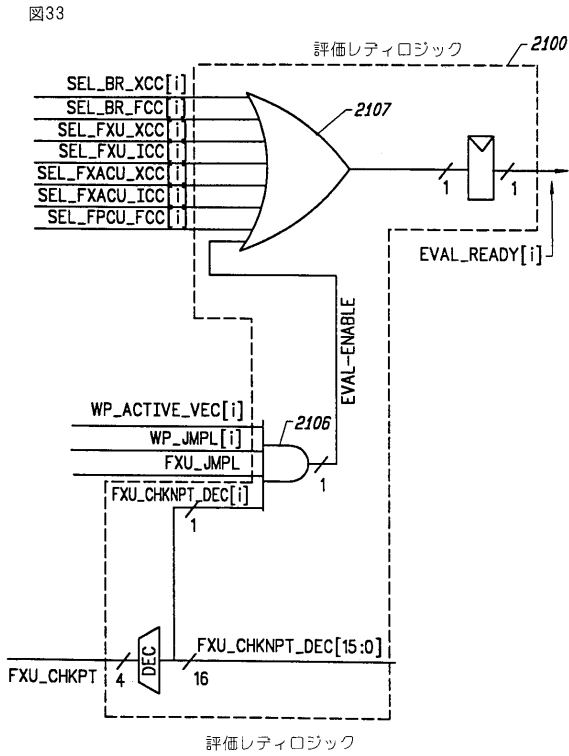
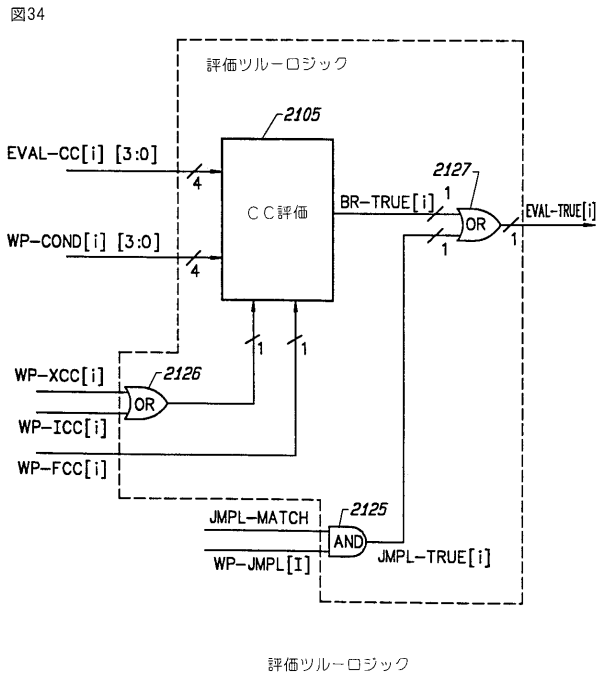


図32AのCCレトリフトロジックは、ロジックテーブルへの個々の信号ラインを示す  
 i = 割の当てられたチェクポイント/ウォッチポイントの数  
 = 例示実施例で16

【 図 3 3 】



【 図 3 4 】

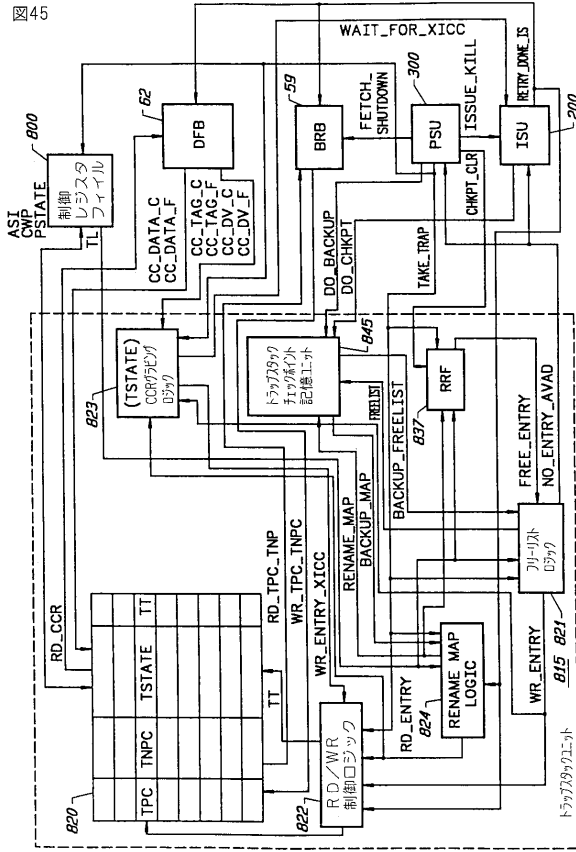




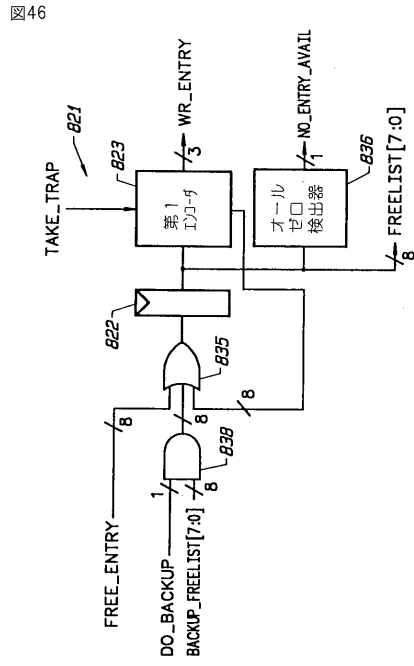




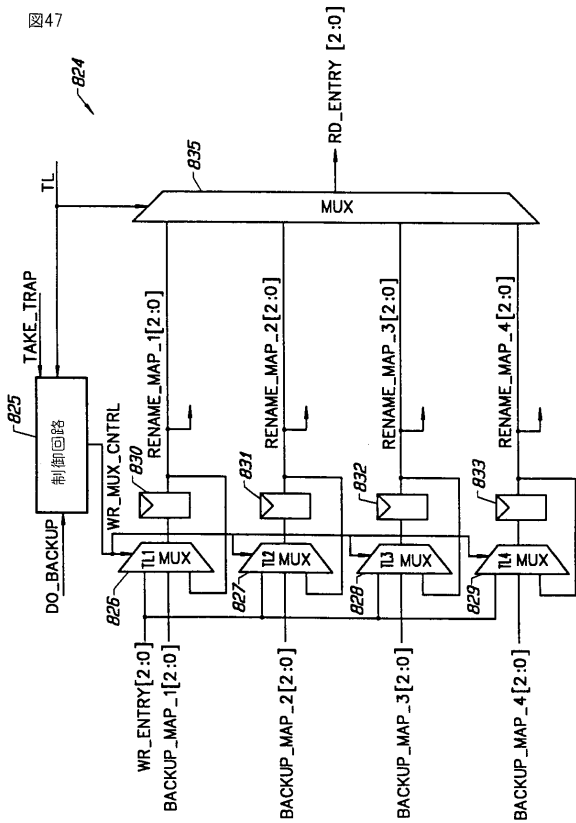
【 図 4 5 】



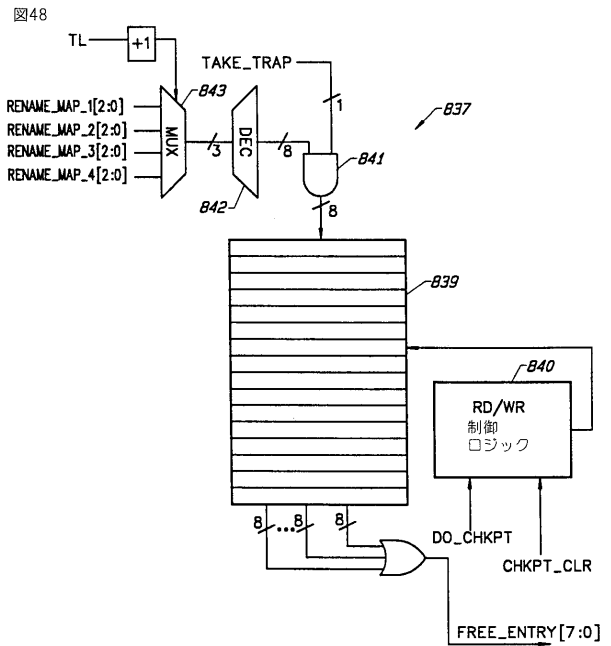
【 図 4 6 】



【 図 4 7 】



【 図 4 8 】









## フロントページの続き

- (31)優先権主張番号 08/473,223  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (31)優先権主張番号 08/476,419  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (31)優先権主張番号 08/478,025  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (31)優先権主張番号 08/482,073  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (31)優先権主張番号 08/483,958  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (31)優先権主張番号 08/484,795  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (31)優先権主張番号 08/487,801  
 (32)優先日 平成7年6月7日(1995.6.7)  
 (33)優先権主張国 米国(US)
- (74)代理人 100114177  
 弁理士 小林 龍
- (72)発明者 シェン, ジーン ダブリュ.  
 アメリカ合衆国, カリフォルニア 94043, マウンテン ビュー, セントラル アベニュー 1  
 81エー
- (72)発明者 スゼエトー, ジョン  
 アメリカ合衆国, カリフォルニア 94610, オークランド, イースト サーティーフォース  
 ストリート 1217
- (72)発明者 パトカー, ニッティーン エー.  
 アメリカ合衆国, カリフォルニア 94087, サニーバール, カーリスル ウェイ 160
- (72)発明者 シェパノウ, マイケル シー.  
 アメリカ合衆国, テキサス 75075, プラノ, グレンウィック ドライブ 1920
- (72)発明者 大曾根 秀樹  
 アメリカ合衆国, カリフォルニア 95130, サン ホセ, グリムズビー コート 2318
- (72)発明者 シモーネ, マイケル エー.  
 アメリカ合衆国, カリフォルニア 95050, サンタ クララ, ワシントン ストリート #8  
 680
- (72)発明者 丸山 拓巳  
 アメリカ合衆国, カリフォルニア 95032, ロス ガトス, イースト メイン ストリート  
 #3020
- Fターム(参考) 5B013 CC03 CC08 CC10 EE02  
 5B033 BE00 DD03 FA16