



(19)  
 Bundesrepublik Deutschland  
 Deutsches Patent- und Markenamt

(10) DE 10 2008 060 505 A1 2010.06.24

(12)

## Offenlegungsschrift

(21) Aktenzeichen: 10 2008 060 505.0

(22) Anmeldetag: 04.12.2008

(43) Offenlegungstag: 24.06.2010

(51) Int Cl.<sup>8</sup>: **G06F 9/46** (2006.01)  
**G06F 15/16** (2006.01)

(71) Anmelder:  
**Kotorlis, Georgios, 71272 Renningen, DE**

(74) Vertreter:  
**Dreiss Patentanwälte, 70188 Stuttgart**

(72) Erfinder:  
**gleich Anmelder**

(56) Für die Beurteilung der Patentfähigkeit in Betracht  
 gezogene Druckschriften:

**US 69 48 172 B1**

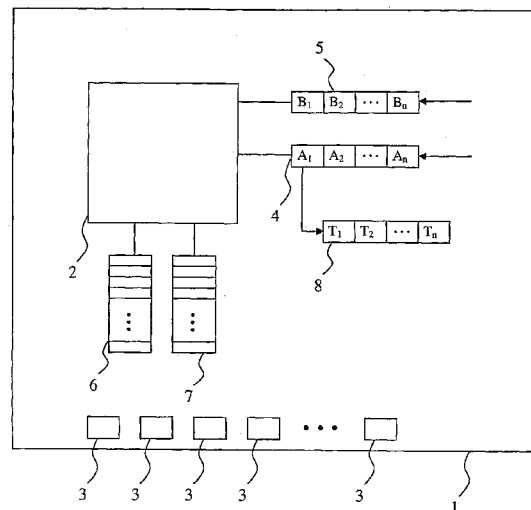
**Ming-fang Wang:A dynamic task scheduling  
 method for multiprocessor  
 system.In:Proceedings of the 1992  
 ACM/SIGAPP symposium on Applied  
 computing,1990,S.840-845**

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gemäß § 44 PatG ist gestellt.

(54) Bezeichnung: **Scheduling in Mehrprozessorsystemen**

(57) Zusammenfassung: Um in einem Mehrprozessorsystem (1) eine effizientere Abarbeitung von Task ( $A_1$ - $A_m$ ,  $B_1$ - $B_m$ ) zu ermöglichen, wird vorgeschlagen, mindestens zwei Tasklisten (4, 5) vorzusehen, wobei parallele Tasks ( $A_1$ - $A_m$ ) in mindestens eine erste Taskliste (4) und parallelisierbare Tasks ( $B_1$ - $B_m$ ) in mindestens eine zweite Taskliste eingetragen werden. Es werden dann in Abhängigkeit von der Anzahl der in der ersten Taskliste (4) eingetragenen parallelen Tasks ( $A_1$ - $A_m$ ) und in Abhängigkeit von der Anzahl der in der zweiten Taskliste (5) eingetragenen parallelisierbaren Tasks ( $B_1$ - $B_m$ ) mindestens eine Prozessorliste (6) und mindestens eine zweite Prozessorliste (7) erstellt beziehungsweise aktualisiert. In mindestens einer ersten Prozessorliste (6) werden die Prozessoren (3) eingetragen, die für eine Bearbeitung der parallelen Tasks ( $A_1$ - $A_m$ ) vorgesehen sind, und in mindestens einer zweiten Prozessorliste (7) werden die Prozessoren (3) eingetragen, die für eine Bearbeitung der parallelisierbaren Tasks ( $B_1$ - $B_m$ ) vorgesehen sind. Die Abarbeitung der parallelen Task ( $A_1$ - $A_m$ ) erfolgt mittels eines präemptiven Scheduling und die Abarbeitung der parallelisierbaren Tasks ( $B_1$ - $B_m$ ) erfolgt mittels eines kooperativen Scheduling.



## Beschreibung

**[0001]** Die Erfindung betrifft ein Verfahren zum Abarbeiten von Tasks in einem Mehrprozessorsystem, wobei mindestens eine Task mindestens eine Teilaufgabe umfasst.

**[0002]** Die Erfindung betrifft ferner ein Mehrprozessorsystem, auf dem eine Mehrzahl von Tasks parallel abarbeitbar ist.

**[0003]** In einem Computersystem werden üblicherweise eine Vielzahl von Aufgaben abgearbeitet. Diese Aufgaben werden beispielsweise als Prozesse oder als Tasks bezeichnet. Eine Aufgabe kann Teilaufgaben umfassen. Je nach der Art der Teilaufgabe kann sich die Teilaufgabe mit der Aufgabe bestimmte Speicherbereiche teilen und wird beispielsweise als Thread bezeichnet. Bezüglich der Bearbeitung einer Teilaufgabe durch einen Prozessor können Teilaufgaben einer Aufgabe ebenfalls als Aufgaben bzw. Tasks betrachtet werden.

**[0004]** Die eigentliche Bearbeitung einer Aufgabe beziehungsweise einer Task erfolgt durch einen Prozessor. Jeder Prozessor kann zu jedem Zeitpunkt grundsätzlich nur eine Task bearbeiten. Ausnahmen hiervon sind so genannte Mehrkernprozessoren, die eine Mehrzahl von Tasks parallel bearbeiten können. Ein Zwei-Kern-Prozessor kann beispielsweise zwei Tasks parallel bearbeiten. Üblicherweise ist die Anzahl der Tasks, die bearbeitet werden sollen, größer als die Anzahl der zur Verfügung stehenden Prozessoren. Es muss deshalb eine Reihenfolge für die Bearbeitung der Tasks bestimmt werden. Dies wird mittels eines so genannten Schedulers erreicht, der die zu bearbeitenden Tasks in einer Liste einreicht und festlegt, welche Task an einen Prozessor zur Bearbeitung weitergereicht wird. Zur Bestimmung der Reihenfolge der Bearbeitung sind eine Vielzahl von so genannten Scheduling-Algorithmen bekannt, beispielsweise das Round-Robin-Scheduling, Priority-Scheduling, Shortest-Job-First-Scheduling, Shortest-Remaining-Time-Scheduling oder Earliest-Deadline-First-Scheduling, um nur einige der bekannten Scheduling-Strategien zu nennen. Den tatsächlichen Zugang einer Task zu einem Prozessor wird durch eine als Dispatcher bezeichnete Funktionseinheit realisiert, die auch als Teil des Schedulers realisiert sein kann.

**[0005]** Eine Vielzahl von Tasks benötigt eine möglichst zeitnahe Bearbeitung, was beispielsweise in Echtzeitsystemen regelmäßig der Fall ist. Hier muss sichergestellt werden, dass diese Tasks möglichst rasch einen Prozessor zugeteilt bekommen und von diesem abgearbeitet werden. Dies wird beispielsweise mittels des so genannten präemptiven Multitasking erreicht, bei dem jeder Task eine bestimmte Bearbeitungszeit zur Verfügung steht. Kann die Task

nicht innerhalb der zur Verfügung stehenden Bearbeitungszeit beendet werden, so wird sie unterbrochen und eine andere Task wird nun von dem Prozessor bearbeitet. Das Unterbrechen und die zu einem späteren Zeitpunkt erfolgende Weiterbearbeitung einer Task erzeugt jedoch einen Overhead, da bei einem Taskwechsel Register neu geladen werden müssen und Speicherbereiche eingelesen beziehungsweise reserviert werden müssen. Ein geringerer Overhead wird mittels des so genannten kooperativen Multitasking erreicht. Dort werden die Tasks ebenfalls in einer bestimmten Reihenfolge bearbeitet, jedoch wird eine Task nicht vor Ablauf einer vorbestimmten Zeit unterbrochen, sondern jede Task wird bis zu deren Ende oder bis zu dem Erreichen eines so genannten Synchronisationspunktes bearbeitet. Ein Synchronisationspunkt wird beispielsweise dann erreicht, wenn eine Task auf eine Ressource zugreifen möchte, die aktuell nicht bereitsteht. Kooperatives Multitasking erzeugt zwar einen geringeren Overhead aufgrund der geringeren Anzahl von Taskwechseln, jedoch ist eine Echtzeitfähigkeit nicht gewährleistet.

**[0006]** Standardmäßig erfolgt die Abarbeitung einer Task linear, so dass die einzelnen Anweisungen innerhalb der Task in linearer Abfolge ausgeführt werden. Diese Linearität kann jedoch durchbrochen werden, falls die Task Teilaufgaben umfasst. Beispielsweise kann eine Berechnung realisierende Task mehrere Teilaufgaben umfassen, die jeweils Zwischenergebnisse berechnen oder eine grafische Oberfläche realisierende Task kann eine Teilaufgabe bzw. einen Thread für die Berechnung eines 3D-Modells umfassen und eine andere Teilaufgabe bzw. einen anderen Thread, der auf eine Benutzereingabe wartet. Eine parallele bzw. quasi-parallele Abarbeitung der Teilaufgaben ist grundsätzlich dann möglich, wenn diese Teilaufgaben unabhängig sind und wenn die Task entsprechend programmiert ist.

**[0007]** Systemarchitekturen, in denen eine parallele beziehungsweise quasi-parallele Abarbeitung von Threads ermöglicht ist, sind so genannte Multitasking-Systeme. In einem Multitasking-System ist es möglich, eine Mehrzahl von Threads einer Task abwechselnd zu bearbeiten, so dass ein Thread, der eine schnelle Reaktion auf ein Ereignis realisieren soll, bevorzugt bearbeitet werden kann. Die parallele Abarbeitung der Threads wird mittels eines präemptiven Schedulers realisiert.

**[0008]** Bei präemptivem Scheduling entsteht ein Overhead dadurch, dass bei jedem Wechsel zwischen den Threads ein Kontextwechsel notwendig ist. Der Overhead ist umso größer, je mehr ausführbare Threads aktuell existieren, was zu einer Verringerung der Effizienz führt. Übersteigt die Anzahl der Threads die Anzahl der Prozessoren um ein Vielfaches, beispielsweise um den Faktor 100, so ist ein

sinnvolles Arbeiten kaum mehr möglich, da die Zeitdauer zwischen zwei Prozessorzuteilungen für einen Thread sehr lang sein kann und ein großer Teil der zur Verfügung stehenden Rechenzeit für den Wechsel der Threads benötigt wird.

**[0009]** Zur Vermeidung des Overheads und zur Vermeidung des Aufwands beim Erzeugen von Threads werden so genannte Thread Pools eingesetzt. Hierbei wird von einem Prozess oder einer Task einmalig eine bestimmte Anzahl von so genannten Worker Threads erstellt, so dass nur einmalig der Aufwand für die Erstellung entsteht. Die Anzahl der Worker Threads richtet sich meist nach der Anzahl der zur Verfügung stehenden Prozessoren. Die dann tatsächlich auszuführenden Aufgaben werden dem Scheduler übergeben, die dieser an die verfügbaren Worker Threads verteilt. Die jeweilige Aufgabe wird von dem betreffenden Thread vollständig ausgeführt, bevor eine neue Aufgabe begonnen wird. Da die Threads prozesslokal sind, sich also jeder Prozess seinen eigenen Thread Pool erzeugen muss, kann dennoch bei einer größeren Anzahl von aktuell zu bearbeitenden Prozessen beziehungsweise Tasks eine derart große Anzahl von Threads existieren, dass ein die Effizienz spürbar senkender Overhead existiert. Ferner können in der Regel nur unabhängige Aufgaben durch die Threads durchgeführt werden. Da die Anzahl der Threads fest ist, können blockierende Aufgaben die Anzahl der noch verwendbaren Threads derart reduzieren, dass schließlich kein Thread mehr für die Bearbeitung zur Verfügung steht. Derartige negative Effekte können auch zeitlich begrenzt beispielsweise durch blockierende Systemaufrufe auftreten. Eine bekannte Lösungsmöglichkeit hierfür ist, so genannte Busy-Waits einzusetzen. Blockiert ein Worker Thread, so wird dessen Blockadezustand ständig überprüft (so genanntes Spinning) bis er sich auflöst. Dies ist in solchen Fällen sinnvoll, in denen anzunehmen ist, dass ein aktives Warten weniger Zeit benötigt als ein Wechsel des Threads.

**[0010]** Bestehende Systeme haben folglich den Nachteil, dass aufgrund einer zu geringen Anzahl von parallel zu bearbeitenden Aufgaben in einem Thread Pool eine Abarbeitung eines Problems nicht möglich ist, wenn die Abhängigkeiten zwischen diesen Threads derart sind, dass eine Blockade entsteht. Ein weiterer Nachteil ist, dass durch den übermäßig häufigen Taskwechsel beziehungsweise Threadwechsel ein großer Overhead erzeugt wird.

**[0011]** Aufgabe der vorliegenden Erfindung ist es, eine Möglichkeit vorzuschlagen, die eine effizientere Abarbeitung von Tasks ermöglicht und die eben genannten Nachteile vermeidet oder zumindest deren negativen Einfluss reduziert.

**[0012]** Die Aufgabe wird durch ein Verfahren zum Abarbeiten von Tasks in einem Mehrprozessorsystem

dadurch gelöst, dass zwei Arten von Tasklisten, die beispielsweise in speziellen Speicherbereichen realisiert sind, zur Verfügung stehen. Die Tasklisten der ersten Art werden als erste Tasklisten bezeichnet und die Tasklisten der zweiten Art werden als zweite Tasklisten bezeichnet. Für den Eintrag einer Task in eine Taskliste wird zwischen unterschiedlichen Parallelitätsarten der Tasks unterschieden. Parallele Tasks oder Threads werden in mindestens eine erste Taskliste, also eine Taskliste der ersten Art, alle übrigen, im folgenden als parallelisierbar bezeichnete Tasks beziehungsweise Threads werden in mindestens eine zweite Taskliste, also eine Taskliste der zweiten Art, eingetragen.

**[0013]** Grundsätzlich können eine Mehrzahl von ersten Tasklisten bzw. Tasklisten der ersten Art und/oder eine Mehrzahl von zweiten Tasklisten bzw. Tasklisten der zweiten Art bei dem erfindungsgemäßen Verfahren eingesetzt werden, so dass mittels der ersten Tasklisten z. B. Prioritätslisten für die parallelen Tasks und/oder mittels der zweiten Tasklisten Prioritätslisten für die parallelisierbaren Tasks realisiert werden können. Die Prioritätslisten können von dem Scheduler bei der Auswahl der zu bearbeitenden Task für ein prioritätsbasiertes Scheduling herangezogen werden. Weitere erste und zweite Tasklisten können z. B. die Topologie der Prozessoren oder Fähigkeiten der Prozessoren berücksichtigen, wodurch nochmals verbesserte Zuteilungen ermöglicht werden. Wenn nicht anders erwähnt sollen die Begriffe „erste Taskliste“ und „zweite Taskliste“ im Folgenden stets so verstanden werden, dass diese jeweils eine Mehrzahl von Prioritätslisten oder andere, ein Scheduling unterstützende Listen umfassen können.

**[0014]** Erfindungsgemäß werden die in dem Mehrprozessorsystem vorhandenen Prozessoren der ersten Taskliste beziehungsweise der zweiten Taskliste zugeordnet. Hierbei wird in Abhängigkeit von der Anzahl und/oder Art der in der ersten Taskliste eingetragenen Tasks und in Abhängigkeit von der Anzahl der in der zweiten Taskliste eingetragenen Tasks mindestens eine erste Prozessorliste erstellt beziehungsweise aktualisiert und mindestens eine zweite Prozessorliste erstellt beziehungsweise aktualisiert. Vorteilhafterweise kann eine Mehrzahl von ersten Prozessorlisten für die Bearbeitung von parallelen Tasks und eine Mehrzahl von zweiten Prozessorlisten für die Bearbeitung von parallelisierbaren Tasks vorgesehen sein. Insbesondere kann für jede erste Taskliste eine erste Prozessorliste und für jede zweite Taskliste eine zweite Prozessorliste vorgesehen sein. In der ersten Prozessorliste werden die Prozessoren eingetragen, die für eine Bearbeitung der in der ersten Taskliste eingetragenen parallelen Tasks vorgesehen sind und in der zweiten Prozessorliste werden die Prozessoren eingetragen, die für eine Bearbeitung der in der zweiten Taskliste eingetragenen parallelisierbaren Tasks vorgesehen sind. Die Art einer Task kann bei-

spielsweise bezeichnen, ob diese Task eine besondere Art von Berechnungen durchführt oder ob diese Task eine sehr lange Bearbeitungszeit benötigt.

**[0015]** Werden von einer Task zum Beispiel eine Vielzahl von Gleitkommaoperationen durchgeführt, so kann vorgesehen sein, einen für diese Operationen spezialisierten Prozessor auszuwählen. Benötigt eine Task eine besonders lange Bearbeitungszeit, so kann vorgesehen sein, einen besonders schnellen Prozessor oder eine Mehrzahl von Prozessoren für die Bearbeitung dieser Task vorzusehen und deshalb in die entsprechende Prozessorliste einzutragen.

**[0016]** Die Abarbeitung der in der ersten Liste eingetragenen parallelen Tasks erfolgt mittels der in der ersten Prozessorliste eingetragenen Prozessoren. Die Zuteilung der Tasks erfolgt mittels eines präemptiven Scheduling. Die Abarbeitung der parallelisierbaren Tasks, die in der zweiten Liste eingetragen sind, erfolgt mittels der in der zweiten Prozessorliste eingetragenen Prozessoren, wobei die Zuteilung der Tasks mittels eines kooperativen Scheduling erfolgt.

**[0017]** Erfindungsgemäß wird sichergestellt, dass ein Abarbeiten von parallelen Tasks zeitnah möglich ist, da parallele Tasks erfindungsgemäß durch die in der ersten Prozessorliste eingetragenen Prozessoren unter Anwendung eines präemptiven Scheduling bearbeitet werden. Parallelisierbare Aufgaben, also Tasks, die nicht zwingend parallel und damit zeitnah abgearbeitet werden müssen, werden von Prozessoren aus der zweiten Prozessorliste unter Anwendung eines kooperativen Scheduling abgearbeitet. Dies bedeutet, dass für parallelisierbare Tasks ein geringerer Overhead erreicht wird als für parallele Tasks, eine zeitnahe Bearbeitung für parallelisierbare Tasks jedoch nicht garantiert werden kann. Demgegenüber wird für parallele Tasks eine zeitnahe Bearbeitung aufgrund des präemptiven Scheduling ermöglicht, allerdings auf Kosten eines möglichen Overheads.

**[0018]** Vorzugsweise wird mindestens ein Prozessor für die Abarbeitung der parallelen Tasks reserviert, wenn in der ersten Taskliste eine solche Task eingetragen ist und es wird mindestens ein Prozessor für eine Abarbeitung der parallelisierbaren Tasks reserviert, wenn eine parallelisierbare Task in der zweiten Taskliste eingetragen ist.

**[0019]** Umfasst eine parallele Task parallelisierbare Teilaufgaben, so wird dieser Task vorzugsweise eine Mehrzahl von Prozessoren zugewiesen, wobei mindestens zwei Teilaufgaben von unterschiedlichen Prozessoren zumindest teilweise parallel abgearbeitet werden. Grundsätzlich können parallelisierbare Teilaufgaben seriell ausgeführt werden. Eine parallele Ausführung solcher Teilaufgaben ermöglicht jedoch eine beschleunigte Abarbeitung der parallelen Tasks. Folglich wird eine nochmals verbesserte Echt-

zeitfähigkeit realisiert.

**[0020]** Besonders vorteilhaft ist es, wenn die Prozessoren, die einer Task zugeordnet werden, in Abhängigkeit von der Anzahl der parallelisierbaren Teilaufgaben der Task bestimmt werden. Insbesondere kann hierbei die Anzahl der parallelisierbaren Teilaufgaben der in der ersten Liste eingetragenen Tasks auch vorteilhaft bei der Zuteilung der Prozessoren in die erste Prozessorliste und die zweite Prozessorliste herangezogen werden. Sind beispielsweise eine Vielzahl von parallelisierbaren Teilaufgaben bei den in der ersten Liste eingetragenen parallelen Tasks vorhanden, so kann vorgesehen sein, eine größere Anzahl von Prozessoren in die erste Prozessorliste einzutragen und somit für eine Bearbeitung der parallelen Tasks vorzusehen. Die Teilaufgaben einer parallelen Task sind selbst wieder parallelisierbar und können von einem oder mehreren der in der ersten Prozessorliste eingetragenen Prozessoren abgearbeitet werden. Umfasst ein parallelisierbarer Task parallelisierbare Teilaufgaben, so können diese von einem oder mehreren der in der zweiten Prozessorliste eingetragenen Prozessoren abgearbeitet werden.

**[0021]** Eine Einteilung der Prozessoren in die erste Prozessorliste und in die zweite Prozessorliste kann zu fest vorgegebenen Zeitpunkten erfolgen. Vorzugsweise wird eine Aktualisierung der Prozessorlisten jedoch ergänzend oder alternativ dann durchgeführt, wenn eine neu zu bearbeitende Task in die erste Taskliste oder in die zweite Taskliste eingetragen wird. Eine neue Zuteilung der Prozessoren zu den parallelen Tasks und zu den parallelisierbaren Tasks ist vorteilhaft in Ergänzung oder alternativ hierzu auch möglich, wenn eine Task vollständig abgearbeitet worden ist. Vorzugsweise wird also dann eine Reservierung der Prozessoren überprüft bzw. aktualisiert, wenn sich eine Änderung in einer der Tasklisten ergibt.

**[0022]** Gemäß einer bevorzugten Ausführungsform erfolgt eine Zuteilung der Prozessoren zu den Prozessorlisten beziehungsweise eine Veränderung der Prozessorlisten in Abhängigkeit von einer aktuellen Prozessorlast. Wird beispielsweise festgestellt, dass durch die mittels des präemptiven Scheduling betriebenen Prozessoren die Aufgabenlast innerhalb eines Zeitraums nicht signifikant reduziert wird, so kann vorgesehen sein, einen oder mehrere Prozessoren aus der zweiten Prozessorliste in die erste Prozessorliste einzutragen und damit für eine Bearbeitung der parallelen Tasks zu reservieren. Damit kann erreicht werden, dass stets Antwortzeiten für die parallelen Tasks realisierbar sind, die innerhalb vorgegebener Grenzen liegen.

**[0023]** Bei der Einteilung der Prozessoren in die erste Prozessorliste oder in die zweite Prozessorliste werden vorteilhafterweise die Eigenschaften des Prozessors berücksichtigt. Eine Eigenschaft ist bei-

spielsweise die Fähigkeit des Prozessors, von einem Scheduler in der Ausführung einer Task unterbrochen zu werden, z. B. durch einen Zeitgeber. Diese Eigenschaft ist die Voraussetzung für die Realisierung eines präemptiven Scheduling, so dass in die erste Prozessorliste nur derartige Prozessoren eingetragen werden können, die diese Fähigkeit aufweisen.

**[0024]** Eine weitere Eigenschaft eines Prozessors kann eine Spezialisierung auf eine bestimmte Art von Aufgaben beschreiben. Beispielsweise kann ein Prozessor aufgrund seiner Hardware für eine bestimmte Art von Berechnungen besonders geeignet sein. Soll von einer Task eine solche Berechnung durchgeführt werden, so kann vorgesehen sein, diesen Prozessor in die Prozessorliste einzutragen, die der Taskliste zugeordnet ist, in die die Task eingetragen ist. Wird beispielsweise von einer parallelen Task eine Berechnung eines mehrdimensionalen Objekts durchgeführt und umfasst das Mehrprozessorsystem einen hierzu geeigneten Grafikprozessor, so kann vorgesehen sein, diesen Grafikprozessor in die erste Prozessorliste einzutragen.

**[0025]** Eine nochmals weitere Eigenschaft kann die Prozessortopologie sein. Umfasst eine Taskliste eine Mehrzahl ähnlicher Tasks, z. B. Tasks, die einen gemeinsamen Speicherbereich nutzen, und umfasst das Mehrprozessorsystem Netzwerkrechner, so kann vorgesehen sein, dass der bzw. die Prozessoren dieses Netzwerkrechners den ähnlichen Tasks zugeordnet wird bzw. werden. Dadurch kann ein möglicher Kommunikationsweg besonders kurz gehalten werden.

**[0026]** Grundsätzlich kann eine Task seine Parallelisierungsart ändern. So kann eine parallelisierbare Task zu einer parallelen Task werden, wenn diese eine Teilaufgabe startet, die eine zeitnahe Bearbeitung benötigt. Beispielsweise kann eine überwiegend im Hintergrund ablaufende Task eine Teilaufgabe erzeugen, die eine Benutzereingabe anfordert und auf diese reagiert. In diesem Fall wird erfindungsgemäß die vormals parallelisierbare Task von der zweiten Taskliste in die erste Taskliste überführt. Dies ist problemlos stets dann möglich, wenn die Task aktuell nicht von einem Prozess bearbeitet wird. Ist dies jedoch der Fall, so kann vorgesehen sein, dass die vormals parallelisierbare Task unterbrochen wird, in die erste Taskliste aufgenommen wird und – gemäß dem präemptiven Scheduling – zu gegebener Zeit von einem in der ersten Prozessorliste eingetragenen Prozessor weiterbearbeitet wird.

**[0027]** Besonders vorteilhaft ist es, eine Task, deren Zustand von parallelisierbar auf parallel geändert wird und die aktuell auf einem in der zweiten Prozessorliste eingetragenen Prozessor abläuft, nicht zu unterbrechen, sondern den Prozessor, der diese Task

bearbeitet, in die erste Prozessorliste aufzunehmen. Dies bedeutet insbesondere, dass der Prozessor dann nicht mehr in dem kooperativen Modus, sondern in dem präemptiven Modus arbeitet.

**[0028]** Die Aufgabe wird auch durch ein Mehrprozessorsystem, auf dem eine Mehrzahl von Tasks parallel abarbeitbar sind, gelöst, wobei das Mehrprozessorsystem eine erste Taskliste für Einträge von parallel abzuarbeitenden Tasks und eine zweite Taskliste für Einträge von parallelisierbaren Tasks umfasst. Das Mehrprozessorsystem umfasst ferner einen Scheduler zum Eintragen von parallel abzuarbeitenden Tasks in die erste Taskliste und zum Eintragen von parallelisierbaren Tasks in die zweite Taskliste. Der Scheduler kann beispielsweise in Hardware realisiert sein. Insbesondere kann der Scheduler mit einem von dem Betriebssystem zur Verfügung gestellten Scheduler zusammenarbeiten.

**[0029]** Das Mehrprozessorsystem umfasst auch eine erste Prozessorliste, in der für eine Abarbeitung von in der ersten Taskliste eingetragenen parallelen Tasks reservierte Prozessoren eintragbar sind, sowie eine zweite Prozessorliste, in der Prozessoren eintragbar sind, die für eine Abarbeitung von in der zweiten Taskliste eingetragenen parallelisierbaren Tasks reserviert sind.

**[0030]** Das Mehrprozessorsystem umfasst außerdem eine Funktionseinheit zum Erstellen beziehungsweise Aktualisieren der Einträge in der ersten Prozessorliste und der Einträge in der zweiten Prozessorliste in Abhängigkeit von der Anzahl der in der ersten Taskliste eingetragenen Tasks und in Abhängigkeit von der Anzahl der in der zweiten Taskliste eingetragenen Tasks.

**[0031]** Das Mehrprozessorsystem umfasst insbesondere eine Mehrzahl von Prozessoren, wobei mindestens ein Prozessor eine Abarbeitung von Tasks sowohl in einem präemptiven als auch einem kooperativen Modus ermöglicht.

**[0032]** Vorzugsweise ist das Mehrprozessorsystem zur Ausführung des erfindungsgemäßen Verfahrens eingerichtet.

**[0033]** Weitere Merkmale, Anwendungsmöglichkeiten und Vorteile der Erfindung ergeben sich aus der nachfolgenden Beschreibung von Ausführungsbeispielen der Erfindung, die anhand der Zeichnungen erläutert werden. Es zeigen:

**[0034]** Fig. 1 eine schematisierte Darstellung von Komponenten eines Mehrprozessorsystems gemäß einer Ausführungsform;

**[0035]** Fig. 2 ein Ablaufdiagramm zur Darstellung ausgewählter Aspekte des erfindungsgemäßen Ver-

fahrens gemäß einer möglichen Ausführungsform;

**[0036]** Fig. 3a und Fig. 3b ein Ablaufdiagramm zur Darstellung von Aspekten bezüglich der Interaktion zwischen einem Prozessor und dem Scheduler;

**[0037]** Fig. 4 ein Ablaufdiagramm zur Darstellung von möglichen Verfahrensschritten bei der Behandlung von einer Änderung der Parallelitätsart eines Tasks gemäß einer möglichen Ausführungsform.

**[0038]** In Fig. 1 ist ein Mehrprozessorsystem **1** dargestellt, das einen Scheduler **2** und eine Mehrzahl von Prozessoren **3** umfasst. Das Mehrprozessorsystem **1** umfasst ferner eine erste Taskliste **4**, in der parallele Tasks  $A_1$ – $A_n$  eingetragen sind und eine zweite Taskliste **5**, in der parallelisierbare Tasks  $B_1$ – $B_m$  eingetragen sind. Die erste Taskliste und die zweite Taskliste können jeweils als eine Mehrzahl von speziellen Listen, z. B.

**[0039]** Prioritätslisten, realisiert sein. Das in Fig. 1 dargestellte Mehrprozessorsystem **1** umfasst ferner eine erste Prozessorliste **6** und eine zweite Prozessorliste **7**. Die Tasklisten **4**, **5** und die Prozessorlisten **6**, **7** werden von dem Scheduler **2** verwaltet.

**[0040]** Für jede Task wird ferner eine Teilaufgabenliste geführt, in welche die bekannten oder während einer Abarbeitung der Task erzeugten Teilaufgaben eingetragen werden. Dies ist in Fig. 1 beispielhaft für eine parallele Task  $A_n$  gezeigt, der eine Teilaufgabenliste **8** zugeordnet ist, in der Teilaufgaben  $T_1$ – $T_q$  eingetragen sind. Während der Abarbeitung einer parallelisierbaren Task werden die parallelisierbaren Teilaufgaben von dem Scheduler wie Tasks behandelt.

**[0041]** In der ersten Prozessorliste **6** sind Prozessoren **3** eingetragen, die für eine Bearbeitung einer parallelen Task  $A_1$ – $A_n$  vorgesehen sind. Die in der ersten Prozessorliste **6** eingetragenen Prozessoren ermöglichen ein präemptives Scheduling und damit eine Unterbrechung einer Task, die aktuell bearbeitet wird.

**[0042]** In der zweiten Prozessorliste **7** sind Prozessoren **3** eingetragen, die für eine Bearbeitung der in der zweiten Taskliste **5** eingetragenen parallelen Tasks  $B_1$ – $B_m$  vorgesehen sind. Diese Prozessoren müssen lediglich eine Bearbeitung der parallelisierbaren Tasks in einem kooperativen Modus ermöglichen.

**[0043]** Der Scheduler **2** ist beispielsweise in Hardware realisiert. Hierzu kann der Scheduler **2** als Mikrocontroller ausgeführt sein, dem mindestens ein Speicherelement zugeordnet ist, wobei das Speicherelement Speicherbereiche umfasst, in denen die Tasklisten **4**, **5** und die Prozessorlisten **6**, **7** abgespeichert sind. Der Scheduler **2** ist ferner mit einem Bus-

system verbunden, das eine Kommunikation mit den Prozessoren **3** ermöglicht. Es kann vorgesehen sein, den Scheduler **2** in dem Mehrprozessorsystem **1** derart anzuordnen, dass dieser mit einem in Software realisierten, beispielsweise in einem Betriebssystem vorhandenen Scheduler interagiert. Hierzu kann sich der erfindungsgemäße Scheduler **2** dem in dem Betriebssystem realisierten Scheduler beispielsweise als eine Vielzahl von Prozessoren oder als nur ein Prozessor darstellen, wobei die tatsächlich in dem Mehrprozessorsystem vorhandenen Prozessoren **3** vor dem in dem Betriebssystem vorhandenen Scheduler verborgen sind. Der erfindungsgemäße Scheduler **2** kann vorteilhafterweise selbst in Software realisiert werden.

**[0044]** Bei dem erfindungsgemäßen Mehrprozessorsystem **1** wird grundsätzlich eine parallelisierbare Task  $B_1$ – $B_m$  von dem zugeteilten Prozessor ohne Unterbrechung vollständig beziehungsweise bis zu einem Synchronisationspunkt abgearbeitet, was einem kooperativen Multitasking entspricht. Mit dem Beginn der Bearbeitung einer Task werden deren Teilaufgaben ebenfalls in die zweite Taskliste **5** eingetragen. Zukünftige Teilaufgaben können ebenfalls in diese Taskliste eingetragen werden und werden damit gleichberechtigt wie Tasks behandelt.

**[0045]** Parallele Tasks  $A_1$ – $A_n$  werden von den ihnen zugeteilten Prozessoren jeweils für eine bestimmte Zeitspanne bearbeitet. Wird die Task innerhalb dieser Zeitspanne nicht beendet, so wird sie unterbrochen und der Prozessor beginnt mit der Bearbeitung einer anderen Task. Dies entspricht einem präemptiven Multitasking. Jeder Prozessor, dem eine Task zugeteilt worden ist, beginnt diese Task und deren Teilaufgaben so lange zu bearbeiten, bis die Task beendet ist oder bis deren Zeitspanne abgelaufen ist. Ist die Zeitspanne abgelaufen, so wird die aktuelle Task unterbrochen und die Task beziehungsweise deren Teilaufgabe wird wieder in die Taskliste eingetragen. Der Prozessor steht dann wieder für die Bearbeitung einer anderen Task zur Verfügung.

**[0046]** Parallelisierbare Teilaufgaben werden ähnlich wie parallelisierbare Tasks  $B_1$ – $B_m$  selbst von den Prozessoren mittels kooperativem Multitasking bearbeitet. Diese Teilaufgaben werden vollständig beziehungsweise bis zu einem Synchronisationspunkt bearbeitet. Teilaufgaben paralleler Tasks  $A_1$ – $A_n$  hingegen werden nach Ablauf der Zeitspanne regelmäßig unterbrochen.

**[0047]** Parallelisierbare Teilaufgaben können sowohl seriell als auch parallel ausgeführt werden. Eine serielle Ausführung ist vorteilhaft, wenn für die Task bereits der optimale Parallelisierungsgrad erreicht worden ist. Dieser wird beispielsweise für rekursive Algorithmen relativ früh erreicht, so dass eine weitere Verfeinerung der Parallelität keinen weiteren Nutzen

bringt. In diesem Fall würde eine weitere Verfeinerung der Parallelität lediglich eine Erhöhung des Overheads und eine unnötige Belegung von Prozessoren bedeuten, die dann nicht für eine zeitnahe Bearbeitung von parallelen Tasks zur Verfügung stünden.

**[0048]** Gemäß einer vorteilhaften Ausführungsform wird eine parallelisierbare Task zunächst linear ausgeführt. Bei jeder parallelisierbaren Teilaufgabe wird durch den Scheduler **2** geprüft, ob diese Teilaufgabe parallel ausgeführt werden kann. Ist dies möglich, so wird sie zu einer Teilaufgabenliste der Task hinzugefügt. Andernfalls wird diese Teilaufgabe seriell ausgeführt. Werden parallelisierbare Tasks zu der Teilaufgabenliste hinzugefügt, so werden diese parallel oder zumindest quasiparallel bearbeitet.

**[0049]** Gemäß einer anderen Ausführungsform wird der lineare Ablauf einer Task beim Erreichen von parallelisierbaren Teilaufgaben unterbrochen. Die Teilaufgaben werden dann in die Teilaufgabenliste eingetragen und von einem oder mehreren Prozessoren bearbeitet. Der lineare Ablauf wird fortgesetzt, wenn die Bearbeitung der Teilaufgaben beendet ist.

**[0050]** Bei der Bearbeitung von Tasks wird ein Synchronisationspunkt erreicht, wenn eine gemeinsam genutzte Ressource bereits durch eine andere Task blockiert ist. In diesem Fall wird auch eine parallelisierbare Task unterbrochen. Eine hierzu eingesetzte so genannte Synchronisationsprimitive, beispielsweise eine Semaphore, hält die unterbrochene Task so lange, bis die blockierende Task die Ressource freigibt. Danach fügt die Synchronisationsprimitive die unterbrochene Task in ihre ursprüngliche Taskliste zur weiteren Bearbeitung wieder hinzu. vorzugsweise wird bei parallelisierbaren Tasks eine vormalig unterbrochene Task, die dann wieder in die Taskliste eingetragen wird, bezüglich der Bearbeitung gegenüber noch nicht gestarteten Tasks bevorzugt. Dies hat den Vorteil, dass das kooperative Konzept besser durchführbar ist, ferner wird dadurch verhindert, dass eine Task, die häufig auf Ressourcen zugreift, eine unverhältnismäßig lange Gesamtbearbeitungszeit benötigt.

**[0051]** Ebenso wie die Synchronisation von Tasks erfolgt, wird auch die Synchronisation von Teilaufgaben untereinander durchgeführt.

**[0052]** Soll ein Prozessor, der aktuell eine Task ausführt, von einer Prozessorliste in die andere Prozessorliste überführt werden, so wird die Bearbeitung dieser Task unterbrochen und die Task wird in deren ursprüngliche Taskliste für eine spätere Bearbeitung hinzugefügt. Der Prozessor kann nun in die andere Prozessorliste verschoben werden.

**[0053]** In [Fig. 2](#) ist eine mögliche Betriebsweise des

in [Fig. 1](#) dargestellten Mehrprozessorsystems **1** gezeigt. In einem Schritt **101** wird eine neue Task an den Scheduler **2** beispielsweise von einem auf dem Mehrprozessorsystem **1** ablaufenden Betriebssystem oder von einer anderen Task, die aktuell von einem Prozessor **3** ausgeführt wird, übergeben. In einem Schritt **101** wird geprüft, ob die neu übergebene Task eine parallele Task oder eine parallelisierbare Task ist. Ist die neu übergebene Task eine parallele Task, so wird sie in einem Schritt **102** in die erste Taskliste **4** eingetragen. Handelt es sich bei der neu übergebenen Task um eine parallelisierbare Task, so wird diese in einem Schritt **103** in die zweite Taskliste **5** eingetragen.

**[0054]** In einem Schritt **104** werden die Prozessorlisten **6, 7** aktualisiert. Dies bedeutet, dass jeweils die Prozessoren bestimmt werden, die in die erste Prozessorliste beziehungsweise in die zweite Prozessorliste eingetragen werden sollen. Vorzugsweise wird hierzu die Anzahl der in der ersten Taskliste **4** und die Anzahl der in der zweiten Taskliste **5** eingetragenen Tasks herangezogen. Ist beispielsweise aktuell keine parallele Task in der ersten Taskliste eingetragen, so besteht keine Notwendigkeit, einen Prozessor **3** für eine Abarbeitung einer parallelen Task zu reservieren. In diesem Fall kann entschieden werden, dass kein Prozessor in der ersten Prozessorliste eingetragen wird.

**[0055]** Vorteilhafterweise wird bei der Zuteilung der Prozessoren **3** in die Prozessorlisten **6, 7** die Gesamtzahl der Prozessoren berücksichtigt. Ferner kann berücksichtigt werden, wie viele Prozessoren aktuell mit der Bearbeitung einer Task beschäftigt sind beziehungsweise wie viele Prozessoren aktuell keine Task bearbeiten. Befinden sich beispielsweise in der zweiten Prozessorliste Prozessoren, die aktuell keine Task bearbeiten, wohingegen in der ersten Prozessorliste sämtliche Prozessoren mit der Bearbeitung einer Task beschäftigt sind, so kann vorgesehen sein, einen Prozessor oder mehrere Prozessoren von der zweiten Prozessorliste in die erste Prozessorliste zu übertragen und diese somit für eine Bearbeitung von parallelen Tasks zu reservieren. Dies ist insbesondere dann vorteilhaft, wenn eine parallele Task weitere parallele Teilaufgaben umfasst, da diese Teilaufgaben dann ebenfalls parallel bearbeitet werden können, was die Reaktionszeit und damit Echtzeitfähigkeit nochmals verbessert.

**[0056]** Bei der Einteilung der Prozessoren in die Prozessorlisten werden vorzugsweise die Eigenschaften des jeweiligen Prozessors berücksichtigt. Eine solche Eigenschaft kann beispielsweise eine besondere Eignung des Prozessors für die Abarbeitung einer Task, dessen Unterbrechbarkeit oder Prozessortopologie sein.

**[0057]** Gemäß einer bevorzugten Ausführungsform

wird für jede parallele Task, die in der ersten Taskliste **4** eingetragen ist, eine Liste mit parallelisierbaren Teilaufgaben geführt. Die parallele Task selbst muss im Sinne der Gleichzeitigkeit ausgeführt werden. Die Teilaufgaben der parallelen Task können jedoch innerhalb der Ausführung der parallelen Task seriell oder parallel bearbeitet werden. Stehen mehrere Prozessoren für die Bearbeitung der Task zur Verfügung, so können Teilaufgaben ebenfalls parallel bearbeitet werden.

**[0058]** In einem Schritt **105** wird die nächste zu bearbeitende Task ausgewählt. Dies wird sowohl für die in der ersten Taskliste eingetragenen parallelen Tasks als auch für die in der zweiten Taskliste eingetragenen parallelisierbaren Tasks durchgeführt. Hierzu kann eine der bekannten Scheduling-Methoden, beispielsweise das Priority-Scheduling oder das First-in-First-out-Scheduling, herangezogen werden.

**[0059]** Soll eine neue Task bearbeitet werden, so wird diese an den entsprechenden Prozessor übergeben. Für parallelisierbare Tasks wird hierzu ein in der zweiten Prozessorliste **7** eingetragener freier Prozessor gewählt. Soll eine parallele Task ausgeführt werden, so erfolgt dies dann, wenn eine parallele Task beendet ist beziehungsweise deren Zeitscheibe abgelaufen ist und diese deshalb unterbrochen wird.

**[0060]** In einem Schritt **106** wird die Task von dem Prozessor, der dieser Task in dem Schritt **105** zugeteilt worden ist, abgearbeitet. In einem Schritt **107** wird geprüft, ob die Bearbeitung einer Task beendet ist. Ist dies der Fall, so kann erneut eine Aktualisierung der Prozessorlisten **6, 7** vorgenommen werden.

**[0061]** Die Verfahrensschritte sind in **Fig. 2** in einer beispielhaften Reihenfolge und Abhängigkeit gezeigt. Vorteilhafterweise werden einzelne Verfahrensschritte parallel oder unabhängig voneinander ausgeführt. Beispielsweise wird in einer vorteilhaften Ausführungsform nicht in einem Schritt **107** geprüft, ob die Task beendet ist, sondern das Beenden einer Task wird automatisch einer Funktionseinheit, die das Aktualisieren der Prozessorlisten durchführt, mitgeteilt. Auch ein erfolgter Eintrag einer Task in einer Taskliste kann als Ereignis bewertet werden, das eine Aktualisierung der Prozessorlisten startet. Es kann auch vorgesehen sein, eine Aktualisierung der Prozessorlisten zeitabhängig und/oder in Abhängigkeit von einer aktuellen Auslastung der Prozessoren zu aktualisieren.

**[0062]** In **Fig. 3a** sind Verfahrensschritte beispielhaft dargestellt, die eine mögliche Verwaltung der ersten Prozessorliste zeigt. In einem Schritt **200** wird von dem Scheduler eine parallele Task aus der ersten Taskliste ausgewählt. In einem Schritt **201** wird ein Prozessor in der ersten Prozessorliste ausge-

wählt. Dieser Prozessor kann ein freier Prozessor sein, also ein Prozessor, dem aktuell noch keine Task zur Bearbeitung übergeben worden ist. Der ausgewählte Prozessor kann insbesondere auch ein Prozessor sein, der zwar eine Task bearbeitet, diese jedoch unterbricht, da deren Zeitrahmen abgelaufen ist. In einem Schritt **202** wird die ausgewählte Task an den in dem Schritt **201** ausgewählten Prozessor übergeben und es wird in einem Schritt **203** die erste Taskliste aktualisiert.

**[0063]** In einem Schritt **204** wird die Task von dem ausgewählten Prozessor bearbeitet.

**[0064]** In einem Schritt **205** wird geprüft, ob die Task beendet ist. Ist dies der Fall, so wird in einem Schritt **206** der Prozessor, der diese Task bisher bearbeitet hat, als frei markiert und es wird die Prozessorliste in einem Schritt **207** entsprechend aktualisiert. Die Schritte **206** und **207** können von einer dem Scheduler zugeordneten Funktionseinheit ausgeführt werden.

**[0065]** Ergibt die Prüfung in dem Schritt **205** jedoch, dass die Task noch nicht beendet ist, so wird in einem Schritt **208** geprüft, ob der Zeitrahmen für die Bearbeitung der in dem Schritt **202** übergebenen Task abgelaufen ist. Ist dies der Fall, so unterbricht der Prozessor die Bearbeitung der Task in einem Schritt **209** und in einem Schritt **210** wird die erneute Eintragung der unterbrochenen Task in die erste Taskliste veranlasst.

**[0066]** In **Fig. 3b** sind Verfahrensschritte beispielhaft dargestellt, die eine mögliche Verwaltung der zweiten Prozessorliste zeigt. In einem Schritt **300** wird von dem Scheduler eine parallelisierbare Task aus der zweiten Taskliste ausgewählt. In einem Schritt **301** wird ein Prozessor in der zweiten Prozessorliste ausgewählt. Dieser Prozessor kann ein freier Prozessor sein, also ein Prozessor, dem aktuell noch keine Task zur Bearbeitung übergeben worden ist. Der ausgewählte Prozessor kann aber auch ein Prozessor sein, der die Bearbeitung einer Task unterbrochen hat, da ein Synchronisationspunkt erreicht worden ist und deshalb die Bearbeitung nicht fortgesetzt werden kann. In einem Schritt **302** wird die ausgewählte Task an den in dem Schritt **301** ausgewählten Prozessor übergeben und es wird in einem Schritt **303** die erste Taskliste aktualisiert. In einem Schritt **304** wird die Task von dem ausgewählten Prozessor bearbeitet.

**[0067]** In einem Schritt **305** wird geprüft, ob die Task beendet ist. Ist dies der Fall, so wird in einem Schritt **306** der Prozessor, der diese Task bisher bearbeitet hat, als frei markiert und es wird die Prozessorliste in einem Schritt **307** aktualisiert. Die Schritte **306** und **307** können analog zu den Schritten **205** und **206** von einer dem Scheduler zugeordneten Funktionseinheit

ausgeführt werden.

**[0068]** Ergibt die Prüfung in dem Schritt **305** jedoch, dass die Task noch nicht beendet ist, so wird in einem Schritt **308** geprüft, ob die Task einen Synchronisationspunkt erreicht hat und deshalb unterbrochen werden soll. Ist dies der Fall, so unterbricht der Prozessor die Bearbeitung der Task in einem Schritt **309**. Sobald eine Bearbeitung der Task wieder möglich ist, wird in einem Schritt **210** die erneute Eintragung der unterbrochenen Task in die zweite Taskliste veranlasst.

**[0069]** Die in den [Fig. 3a](#) und [Fig. 3b](#) dargestellten Verfahrensschritte dienen zunächst der Darstellung einer möglichen Arbeitsweise des erfindungsgemäßen Mehrprozessorsystems **1**, wobei eine Vielzahl anderer Ausführungsformen vorstellbar ist. Insbesondere kann das Aktualisieren der Taskliste beziehungsweise das Aktualisieren der Prozessorliste zu verschiedenen Zeitpunkt beziehungsweise in Abhängigkeit von unterschiedlichen Ereignissen erfolgen. Eine Realisierung der Verfahrensschritte in einem Mehrprozessorsystem wird ferner zweckmäßigerweise eine ereignisgesteuerte Implementierung wählen. So könnte beispielsweise automatisch eine Nachricht von dem Prozessor an den Scheduler übermittelt werden, sobald der Prozessor die Bearbeitung einer Task beendet hat.

**[0070]** In [Fig. 4](#) sind Verfahrensschritte gezeigt, die gemäß einer Ausführungsform durchlaufen werden können, wenn sich die Parallelitätsart eines Tasks ändert. In einem Schritt **400** wird eine parallelisierbare Task zu einer parallelen Task. Dies ist beispielsweise dann der Fall, wenn eine parallelisierbare Task eine parallele Teilaufgabe startet. In einem Schritt **401** wird geprüft, ob die nun parallele Task aktuell ausgeführt wird. Ist dies nicht der Fall, so wird diese Task in einem Schritt **405** aus der zweiten Taskliste entfernt und in die erste Taskliste eingetragen. Die Task wird dann möglicherweise zusammen mit weiteren in der ersten Taskliste eingetragenen parallelen Tasks in dem präemptiven Modus durch mindestens einen in der ersten Prozessorliste eingetragenen Prozessor bearbeitet.

**[0071]** Ergibt die Prüfung in dem Schritt **401**, dass die vormals parallelisierbare, nun aber parallele Task bereits ausgeführt wird, so wird zu einem Schritt **402** verzweigt. Da die nun parallele Task vormals parallelisierbar war, wird die Task aktuell auf einem Prozessor ausgeführt, der in der zweiten Prozessorliste eingetragen ist und für die Bearbeitung parallelisierbarer Aufgaben reserviert ist. Es wird deshalb in dem Schritt **402** geprüft, ob der Prozessor verschiebbar ist, ob also eine Zuteilung des aktuellen Prozessors in die erste Prozessorliste möglich ist. Dies hätte den Vorteil, dass der Task ohne Unterbrechung auf diesem Prozessor weiterbearbeitet werden könnte. Ist

der Prozessor verschiebbar, so wird er in einem Schritt **404** in die andere Prozessorliste verschoben. Ist der Prozessor jedoch nicht verschiebbar, beispielsweise weil in der zweiten Prozessorliste nur eine sehr geringe Anzahl von Prozessoren eingetragen ist, jedoch in der ersten Prozessorliste eine ausreichende Zahl von Prozessoren eingetragen ist, so wird in einem Schritt **403** die Task unterbrochen und in dem Schritt **405** in die erste Taskliste eingefügt.

**[0072]** Selbstverständlich ist es auch vorstellbar, den Prozessor zusammen mit der ablaufenden Task von der zweiten Prozessorliste in die erste Prozessorliste zu verschieben und als Ausgleich hierfür einen freien Prozessor aus der ersten Prozessorliste in die zweite Prozessorliste zu verschieben. Sollte aktuell kein freier Prozessor verfügbar sein, so könnte der nächste frei werdende Prozessor aus der ersten Prozessorliste in die zweite Prozessorliste verschoben werden.

**[0073]** Ändert sich also die Parallelitätsart einer Tasks, so kann die Task von einer Taskliste in die andere verschoben werden, sofern die Task nicht ausgeführt wird. Wird sie jedoch bereits ausgeführt, so wird die Task zunächst unterbrochen und dann in die andere Liste verschoben. Alternativ hierzu kann der Prozessor zusammen mit der auf diesem ablaufenden Task in die andere Prozessorliste übertragen werden.

## Patentansprüche

1. Verfahren zum Abarbeiten von Tasks ( $A_1-A_m$ ,  $B_1-B_m$ ) in einem Mehrprozessorsystem (**1**), gekennzeichnet durch die folgenden Schritte:
  - a) Einfügen paralleler Tasks ( $A_1-A_m$ ) in mindestens eine erste Taskliste (**4**);
  - b) Einfügen parallelisierbarer Tasks ( $B_1-B_m$ ) in mindestens eine zweite Taskliste (**5**);
  - c) Aktualisieren mindestens einer ersten Prozessorliste (**6**) und Aktualisieren mindestens einer zweiten Prozessorliste (**7**) in Abhängigkeit von der Anzahl und/oder Art der in der mindestens einen ersten Taskliste (**4**) eingetragenen parallelen Tasks ( $A_1-A_m$ ) und in Abhängigkeit von der Anzahl und/oder Art der in der mindestens einen zweiten Taskliste (**5**) eingetragenen parallelisierbaren Tasks ( $B_1-B_m$ ), wobei in mindestens einer ersten Prozessorliste (**6**) Prozessoren (**3**) eingetragen sind, die für eine Bearbeitung der parallelen Tasks ( $A_1-A_m$ ) vorgesehen sind und wobei in mindestens einer zweiten Prozessorliste (**7**) Prozessoren (**3**) eingetragen sind, die für eine Bearbeitung der parallelisierbaren Tasks ( $B_1-B_m$ ) vorgesehen sind;
  - d) Übergeben der parallelen Tasks ( $A_1-A_m$ ) an die in mindestens einer ersten Prozessorliste (**6**) eingetragenen Prozessoren (**3**), wobei die in mindestens einer ersten Prozessorliste (**6**) eingetragenen Prozessoren (**3**) bezüglich der parallelen Tasks ( $A_1-A_m$ ) ein

präemptives Scheduling realisieren;

e) Übergeben der parallelisierbaren Tasks ( $B_1-B_m$ ) an die in mindestens einer zweiten Prozessorliste (7) eingetragenen Prozessoren (3), wobei die in mindestens einer zweiten Prozessorliste (7) eingetragenen Prozessoren (3) bezüglich der parallelisierbaren Tasks ( $B_1-B_m$ ) ein kooperatives Scheduling realisieren.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet dass mindestens ein Prozessor (3) für die Bearbeitung paralleler Tasks ( $A_1-A_m$ ) reserviert wird, falls mindestens eine parallele Task ( $A_1-A_m$ ) in mindestens eine erste Taskliste (4) eingetragen ist, und dass mindestens ein Prozessor (3) für die Bearbeitung parallelisierbarer Tasks ( $B_1-B_m$ ) reserviert wird, falls mindestens eine parallelisierbare Task ( $B_1-B_m$ ) in mindestens eine zweite Taskliste (5) eingetragen ist.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass jeder Task ( $A_1-A_m, B_1-B_m$ ), die mindestens eine Teilaufgabe ( $T_1-T_q$ ) umfasst, eine Teilaufgabenliste (8) zugeordnet ist und die Teilaufgabe ( $T_1-T_q$ ) in die Teilaufgabenliste (8) eingetragen wird.

4. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass mindestens eine parallele Task ( $A_1-A_m$ ) parallelisierbare Teilaufgaben ( $T_1-T_q$ ) umfasst, dieser Task eine Mehrzahl von Prozessoren (3) zugewiesen wird und mindestens zwei Teilaufgaben ( $T_1-T_q$ ) von unterschiedlichen Prozessoren (3) zumindest teilweise parallel abgearbeitet werden.

5. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass die Anzahl und/oder Art der Prozessoren (3), die einer Task ( $A_1-A_m, B_1-B_m$ ) zugeordnet werden, in Abhängigkeit von der Anzahl und/oder Art der Teilaufgaben ( $T_1-T_q$ ) der Task ( $A_1-A_m, B_1-B_m$ ) bestimmt wird.

6. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass bei der Einteilung der Prozessoren in eine der ersten Prozessorlisten und/oder in eine der zweiten Prozessorlisten mindestens eine Eigenschaft des Prozessors berücksichtigt wird, wobei die mindestens eine Eigenschaft insbesondere eine Spezialisierung auf eine bestimmte Art von Aufgaben, eine Prozessortopologie und/oder eine Unterbrechbarkeit bei der Abarbeitung einer Task beschreibt.

7. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass der Schritt c) für jeden neu in eine Taskliste (4, 5) eingefügten Task ( $A_1-A_m, B_1-B_m$ ) und/oder für jede neu hinzugefügte Teilaufgabe ( $T_1-T_q$ ) durchgeführt wird.

8. Verfahren nach einem der vorangehenden An-

sprüche, dadurch gekennzeichnet, dass eine Prozessorlast bestimmt wird und während des Betriebs des Mehrprozessorsystems (1) Prozessoren (3) zwischen den ersten Prozessorlisten (6) und den zweiten Prozessorlisten (7) in Abhängigkeit von der Prozessorlast dynamisch verschoben werden.

9. Verfahren nach einem der vorangehenden Ansprüche, dadurch gekennzeichnet, dass bei Änderung einer parallelisierbaren Task ( $B_1-B_m$ ) in eine parallele Task ( $A_1-A_m$ ) und/oder bei einer Änderung einer parallelen Task ( $A_1-A_m$ ) in eine parallelisierbare Task ( $B_1-B_m$ )

– eine aktuell nicht von einem Prozessor (3) bearbeitete Task ( $A_1-A_m, B_1-B_m$ ) in die jeweils andere Taskliste (4, 5) verschoben wird;

– eine aktuell von einem Prozessor (3) bearbeitete Task ( $A_1-A_m, B_1-B_m$ ) unterbrochen wird und in die jeweils andere Taskliste (4, 5) verschoben wird; oder

– der diese Task ( $A_1-A_m, B_1-B_m$ ) bearbeitende Prozessor (3) in die jeweils andere Prozessorliste (6, 7) verschoben wird, wobei die Bearbeitung der Task ( $A_1-A_m, B_1-B_m$ ) nicht unterbrochen wird.

10. Mehrprozessorsystem (1), auf dem eine Mehrzahl von Tasks ( $A_1-A_m, B_1-B_m$ ) parallel abarbeitbar sind, umfassend:

– mindestens eine erste Taskliste (4) für Einträge von parallelen Tasks ( $A_1-A_m$ );

– mindestens eine zweite Taskliste (5) für Einträge von parallelisierbaren Tasks ( $B_1-B_m$ );

– einen Scheduler (2) zum Eintragen von parallelen Tasks ( $A_1-A_m$ ) in mindestens eine erste Taskliste (4) und zum Eintragen von parallelisierbaren Tasks (5) in mindestens eine zweite Taskliste (5);

– mindestens eine erste Prozessorliste (6) in der ein Eintrag für jeden Prozessor (3) vorhanden ist, der für die Abarbeitung der parallelen Tasks ( $A_1-A_m$ ) vorgesehen sind;

– mindestens eine zweite Prozessorliste (7) in der ein Eintrag für jeden Prozessor (3) vorhanden ist, der für die Abarbeitung der parallelisierbaren Tasks ( $B_1-B_m$ ) vorgesehen ist;

– eine Funktionseinheit zum Erstellen bzw. Aktualisieren der Einträge in den ersten Prozessorlisten (6) und in den zweiten Prozessorlisten (7) in Abhängigkeit von der Anzahl der in der ersten Taskliste (4) oder den ersten Tasklisten eingetragenen parallelen Tasks ( $A_1-A_m$ ) und in Abhängigkeit von der Anzahl der in der zweiten Taskliste (5) oder den zweiten Tasklisten eingetragenen parallelisierbaren Tasks ( $B_1-B_m$ );

– eine Mehrzahl von Prozessoren (3), wobei mindestens ein Prozessor eine Abarbeitung von Tasks sowohl in einem präemptiven als auch in einem kooperativen Modus ermöglicht, und wobei jeder in einer der ersten Prozessorlisten (6) eingetragene Prozessor die diesem zugeteilte parallele Task ( $A_1-A_m$ ) präemptiv abarbeitet und jeder in einer der zweiten Prozessorlisten (7) eingetragene Prozessor (3) die

diesem zugeteilte parallelisierbare Task ( $B_1$ - $B_m$ ) kooperativ abarbeitet.

11. Mehrprozessorsystem (1) nach Anspruch 10, dadurch gekennzeichnet, dass das Mehrprozessorsystem (1) zur Durchführung des erfindungsgemäßen Verfahrens nach einem der Ansprüche 1 bis 9 hergerichtet ist.

12. Mehrprozessorsystem nach Anspruch 10 oder 11, dadurch gekennzeichnet, dass der Scheduler (2) in Hardware realisiert ist.

Es folgen 5 Blatt Zeichnungen

Anhängende Zeichnungen

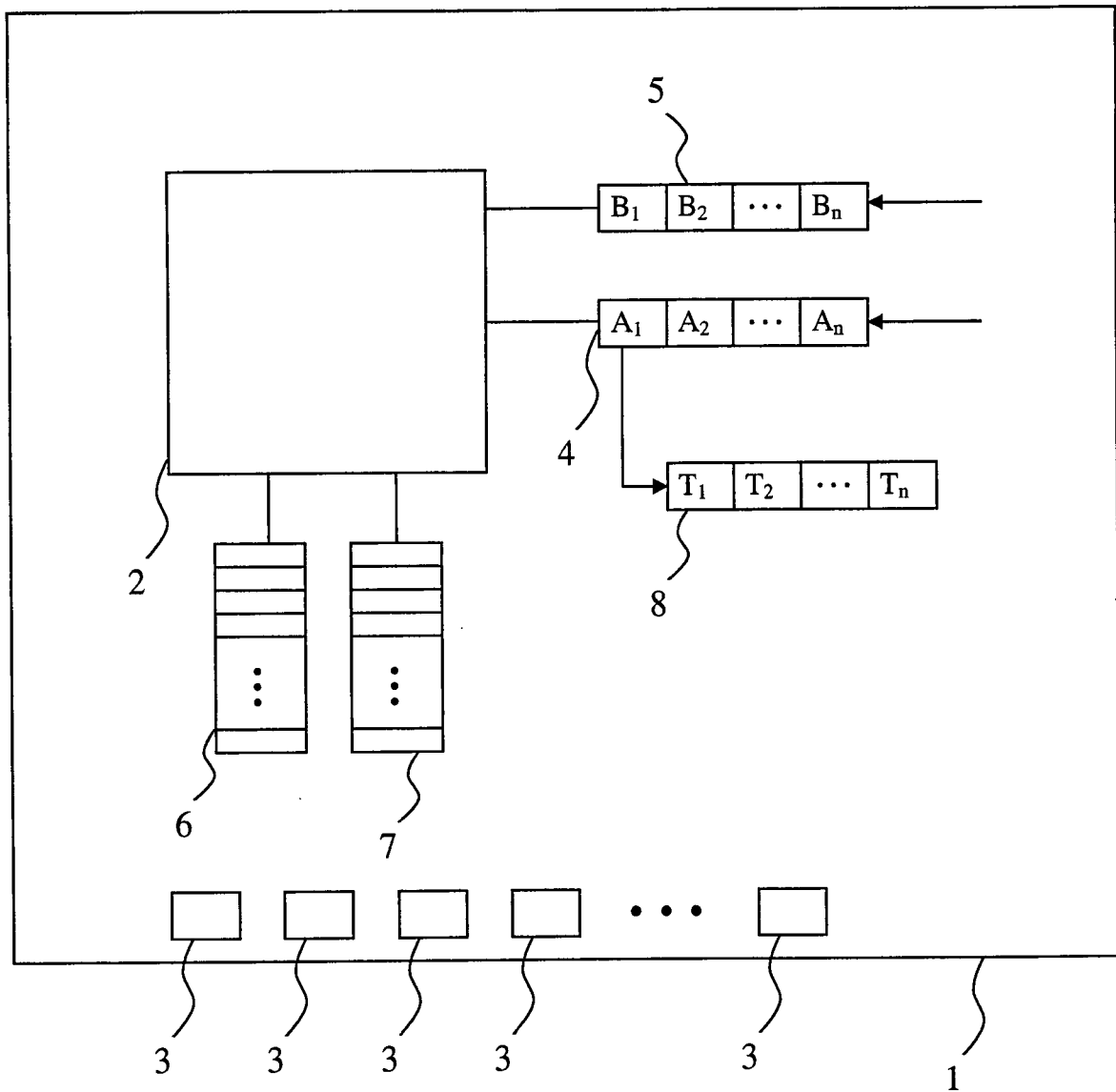


Fig. 1

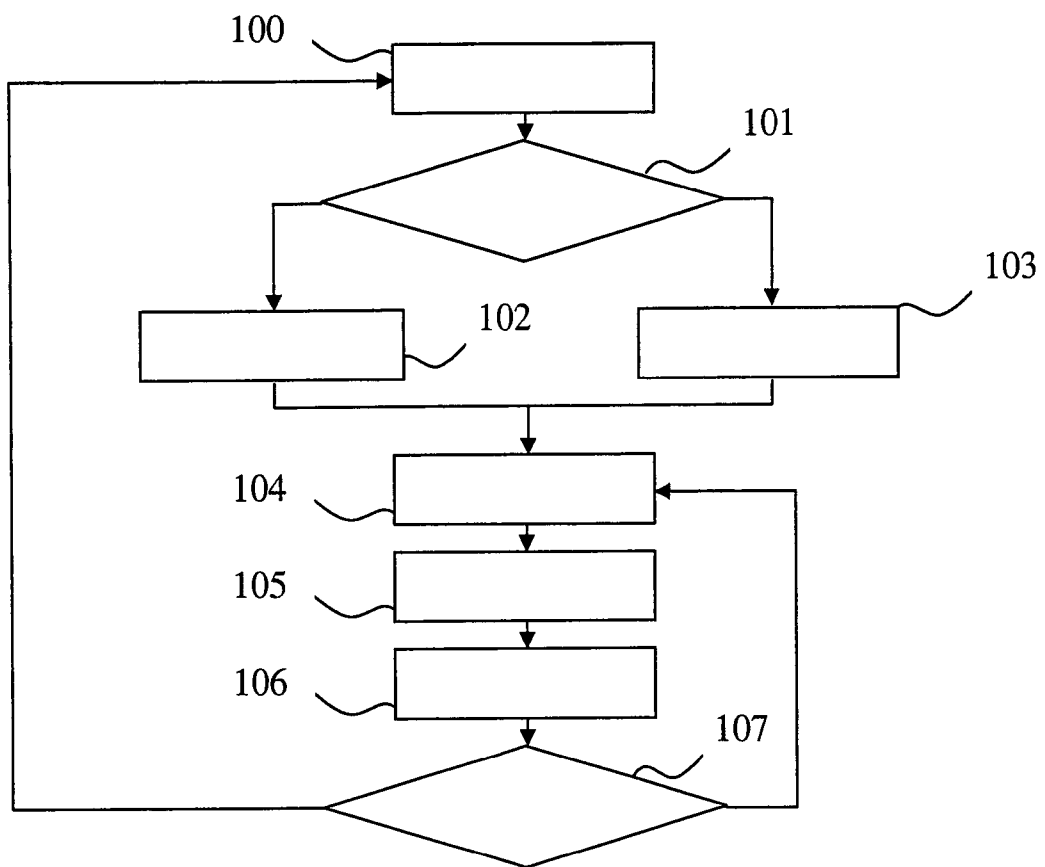


Fig. 2

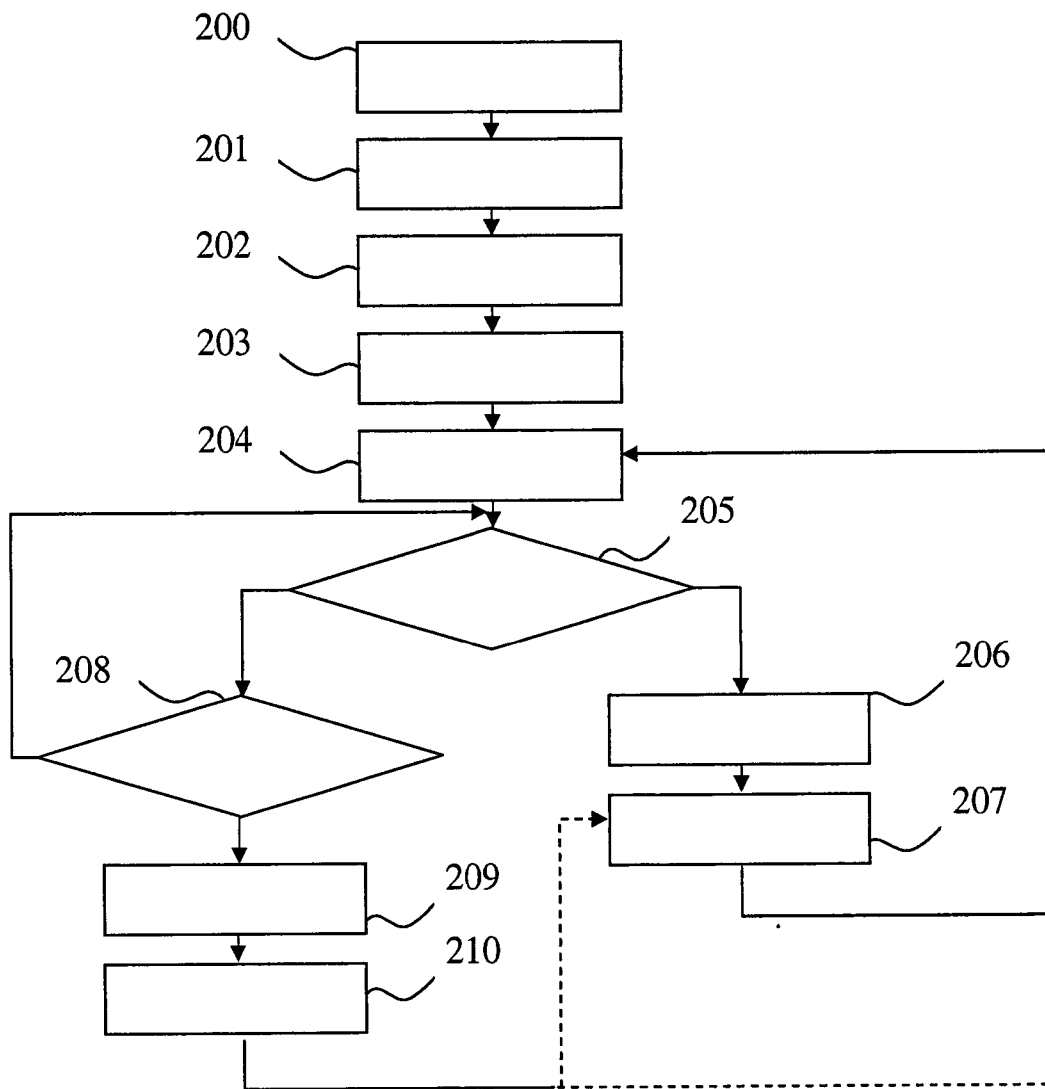


Fig. 3a

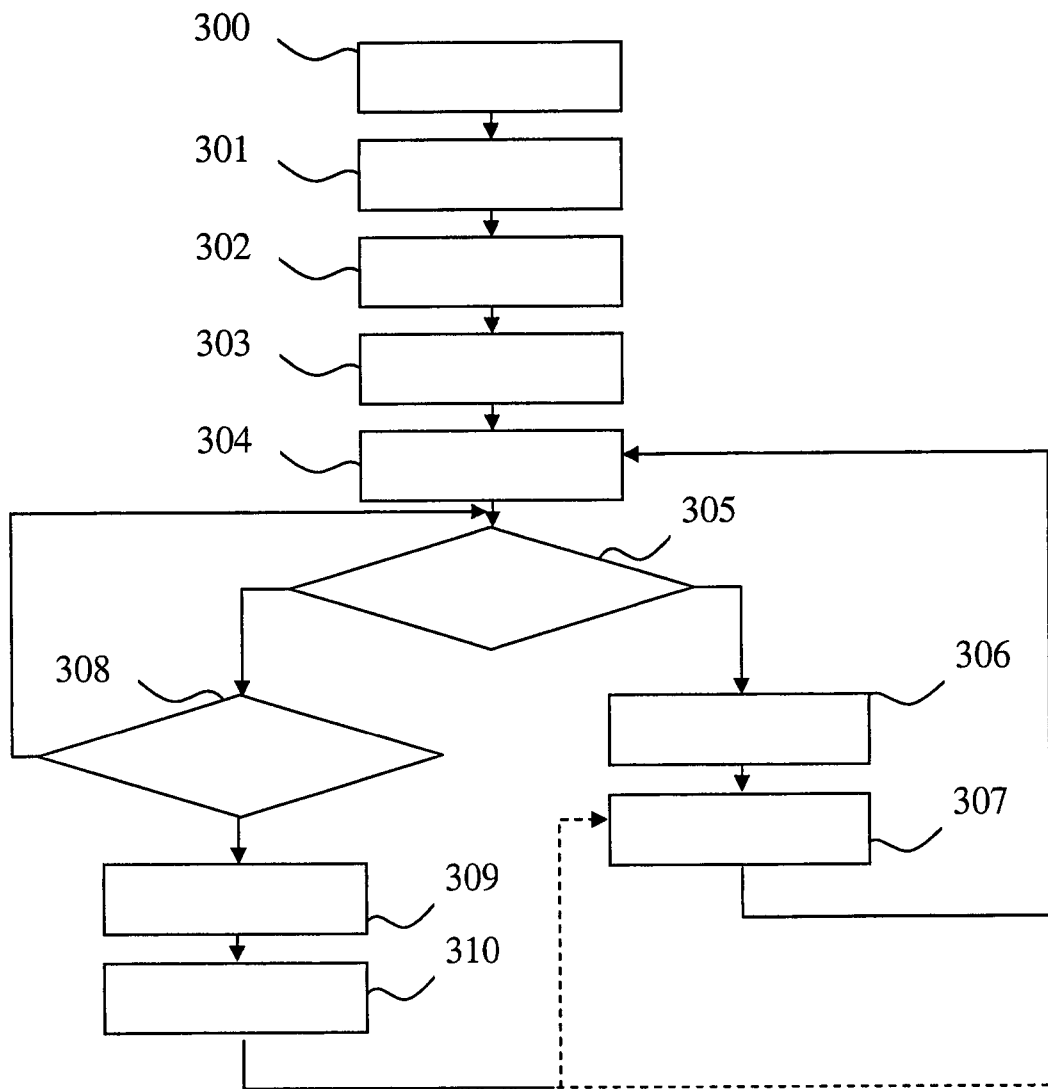
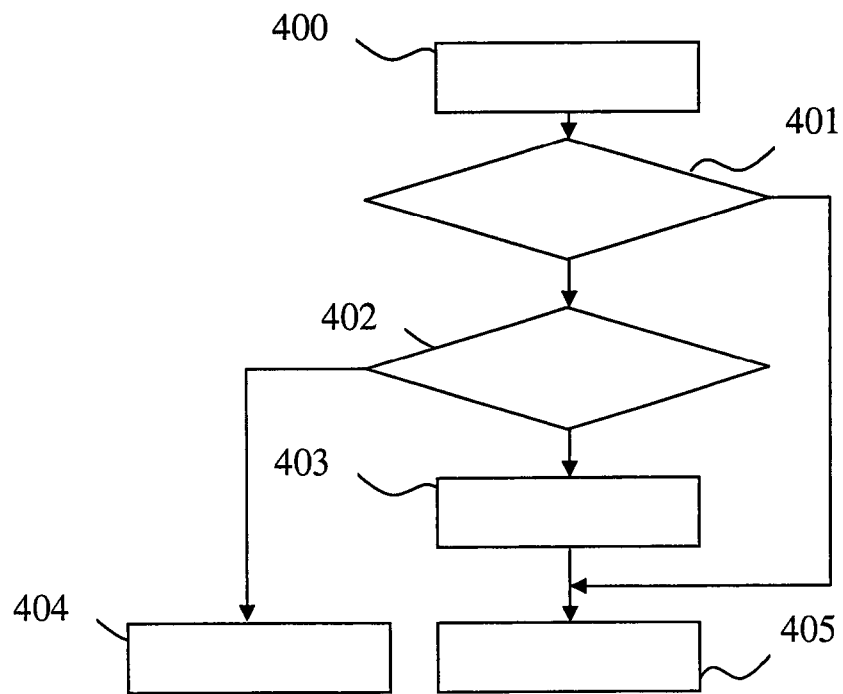


Fig. 3b



**Fig. 4**