(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
27. Juni 2013 (27.06.2013)

W I P O **I P C T**

(10) Internationale Veröffentlichungsnummer
WO 2013/091908 AI

(51) **Internationale Patentklassifikation:**
G06F 9/45 (2006.01)

(21) **Internationales Aktenzeichen:** PCT/EP2012/062571

(22) **Internationales Anmeldedatum:**
28. Juni 2012 (28.06.2012)

(25) **Einreichungssprache:** Deutsch

(26) **Veröffentlichungssprache:** Deutsch

(30) **Angaben zur Priorität:**
10 201 1 089 180.3
20. Dezember 201 1 (20. 12.201 1) DE

(71) **Anmelder** *(für alle Bestimmungsstaaten mit Ausnahme von US):* **SIEMENS AKTIENGESELLSCHAFT** [DE/DE]; Wittelsbacherplatz 2, 80333 München (DE). **FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG** [DE/DE]; Schlossplatz 4, 91054 Erlangen (DE).

(72) **Erfinder; und**

(75) **Erfinder/Anmelder** *(nur für US):* **KEMPF, Stefan** [DE/DE]; Benno-Mayer-Str. 6, 90763 Fürth (DE). **VELDEMA, Ronald** [NL/DE]; Schleifweg 9, 91080 Uttenreuth (DE). **PHILIPPSEN, Michael** [DE/DE]; Tucherstr. 29, 90562 Heroldsberg (DE). **WIECZOREK, Michael** [DE/DE]; Hausäckerweg 12, 91056 Erlangen (DE).

(74) **Gemeinsamer Vertreter: SIEMENS AKTIENGESELLSCHAFT;** Postfach 22 16 34, 80506 München (DE).

(81) **Bestimmungsstaaten** *(soweit nicht anders angegeben, für jede verfügbare nationale Schutzrechtsart):* AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

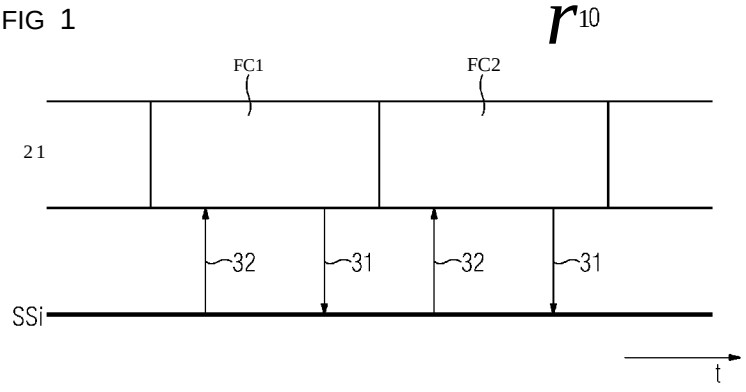
(84) **Bestimmungsstaaten** *(soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, RU, TJ, TM), europäisches (AL, AT, BE, BG, CH, CY,

[Fortsetzung auf der nächsten Seite]

(54) Title: METHOD AND DEVICE FOR INSERTING SYNCHRONIZATION COMMANDS INTO PROGRAM SECTIONS OF A PROGRAM

(54) Bezeichnung : VERFAHREN UND VORRICHTUNG ZUM EINFÜGEN VON SYNCHRONISATIONSBEFEHLEN IN PROGRAMMABSCHNITTE EINES PROGRAMMS

FIG 1



(57) Abstract: A device and a method (100) for inserting synchronization commands (41, 42) into program sections (PAl, PA2) of a program (10, 12) are provided for the synchronization of data. To this end, an enable command (41) is automatically inserted after a write command (31) of a first program section (PAl) to write to a resource (SSi) that is used by the first program section (PAl) and a second program section (PA2) if the write command (31) to write to the resource (SSi) is to be executed by the first program section (PAl) before a read command (32) of the second program section (PA2) to read the resource (SSi); an execution of the enable command (41) places a condition variable (c) into a set State. A wait command (42) to wait for the set State of the condition variable (c) is automatically inserted before the read command (32) to read the resource (SSi), an execution of the wait command (42) causing the second program section (PA2) to wait until the condition variable (c) is set before continuing processing of the second program section (PA2).

(57) Zusammenfassung:

[Fortsetzung auf der nächsten Seite]

WO 2013/091908 A1

CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Veröffentlicht:

— *mit internationalem Recherchenbericht (Artikel 21 Absatz V*

Zur Datensynchronisation wird eine Vorrichtung und ein Verfahren (100) zum Einfügen von Synchronisationsbefehlen (41, 42) in Programmabschnitte (PAl, PA2) eines Programms (10, 12) bereitgestellt. Dazu wird ein Freigabebefehl (41) nach einem Schreibbefehl (31) eines ersten Programmabschnitts (PAl) auf eine Ressource (SSi) automatisiert eingefügt, die von dem ersten Programmabschnitt (PAl) und einem zweiten Programmabschnitt (PA2) gemeinsam genutzt wird, falls der Schreibbefehl (31) auf die Ressource (SSi) von dem ersten Programmabschnitt (PAl) vor einem Lesebefehl (32) des zweiten Programmabschnitts (PA2) auf die Ressource (SSi) auszuführen ist, wobei eine Ausführung des Freigabebefehls (41) eine Bedingungsvariable (c) in einen Gesetztzustand versetzt. Ein Wartebefehl (42) zum Warten auf den Gesetztzustand der Bedingungsvariable (c) wird vor dem Lesebefehl (32) auf die Ressource (SSi) automatisiert eingefügt, wobei eine Ausführung des Wartebefehls (42) den zweiten Programmabschnitt (PA2) dazu veranlasst, mit einer Fortführung der Abarbeitung des zweiten Programmabschnitts (PA2) zu warten, bis die Bedingungsvariable (c) gesetzt ist.

Beschreibung

Verfahren und Vorrichtung zum Einfügen von Synchronisationsbefehlen in Programmabschnitte eines Programms

Die Erfindung betrifft ein Verfahren zum Einfügen von Synchronisationsbefehlen in Programmabschnitte eines Programms. Bei dem Programm kann es sich beispielsweise um ein Programm handeln, das in einer höheren Programmiersprache und/oder in einer Assemblersprache vorliegt und/oder das für eine speicherprogrammierte Steuerung *(PLC = programmable logic Controller)* verwendet wird. Die Synchronisationsbefehle können beispielsweise Freigabe- und Wartebefehle sein.

Außerdem betrifft die Erfindung eine Vorrichtung zum Einfügen von Synchronisationsbefehlen in Programmabschnitte eines Programms .

In einer parallelen Abarbeitung eines sequentiellen Programms muss sichergestellt werden, dass Zugriffe (Schreib- und Lesebefehle) auf Ressourcen (beispielsweise auf Speicherstellen, Aktoren und Sensoren) , die in einem sequentiellen Programm in einer bestimmten Reihenfolge auszuführen sind, auch bei der parallelen Abarbeitung in der gewollten Reihenfolge ausgeführt werden. Andernfalls kann aufgrund fehlender oder fehlerhafter Datensynchronisation eine sogenannte Wettlauf Situation auftreten, die zu unerwünschter Datenänderung und/oder unerwünschter Datennutzung führt.

Aus McCloskey, B . et al .: Autolocker: Synchronization Inference for Atomic Sections, Conf . Record of the 33rd ACM SIGPLAN-SIGACT Symposium of Principles of Programming Languages, POPL '06, S . 346 - 358, ist ein Verfahren bekannt, das im Programmcode Stellen analysiert, die als kritische Abschnitte markiert sind, und automatisch Mutexe einführt, um

einen gegenseitigen Ausschluss zu gewährleisten.

Aus Naik, M. et AI.: Effective Static Race Detection for Java, Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '06, s. 308 - 319, ist ein weiteres Verfahren bekannt, mit welchem Threads eines Programms analysiert werden und für jede Anweisung untersucht wird, welche Mutexe bei der Anweisung gesperrt sind. Wenn es eine Anweisung gibt, die auf Speicherstellen zugreift, welche von mehreren Threads genutzt werden können, und bei der kein Mutex gesperrt ist, kann bei dieser Anweisung möglicherweise eine Wettlauf Situation entstehen.

Trotz der genannten, bekannten Hilfsmittel, muss ein Programmierer alle kritischen Abschnitte manuell und fehlerfrei erkennen, um eine fehlerfreie Parallelisierung des Programmcodes zu erreichen, mit der eine Abarbeitung des Programmcodes in der gewollten Reihenfolge ausgeführt wird. Die Wettlaufsituationen müssen also bisher vom Programmierer vermieden werden, indem er sogenannte kritische Abschnitte identifiziert, die atomar ausgeführt werden und deshalb auch als 'atomare Blöcke' bezeichnet werden. Der gegenseitige Ausschluss kritischer Abschnitte kann manuell durch Einfügen von Mutexen oder mit einem transaktionalen Speicher realisiert werden. Das manuelle Einfügen von Synchronisationsbefehlen (wie Freigabe- und Wartebefehlen) in parallelisierten Programmcode ist fehleranfällig .

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren bereitzustellen, mit dem eine Fehlerfreiheit der Einfügung von Synchronisationsbefehlen erhöht werden kann.

Erfindungsgemäß wird diese Aufgabe dadurch gelöst, dass ein Verfahren zum Einfügen von Synchronisationsbefehlen in Programmabschnitte eines Programms bereitgestellt wird, das fol-

gende Schritte umfasst:

- automatisiertes Einfügen eines Freigabebefehls nach einem Schreibbefehl eines ersten Programmabschnitts auf eine Ressource, die von dem ersten Programmabschnitt und einem zweiten Programmabschnitt gemeinsam genutzt wird, falls der Schreibbefehl auf die Ressource von dem ersten Programmabschnitt vor einem Lesebefehl des zweiten Programmabschnitts auf die Ressource auszuführen ist, wobei eine Ausführung des Freigabebefehls eine Bedingungsvariable in einen Gesetzt zustand versetzt; und
- automatisiertes Einfügen eines Wartebefehls zum Warten auf den Gesetzt zustand der Bedingungsvariable vor dem Lesebefehl des zweiten Programmabschnitts auf die Ressource, wobei eine Ausführung des Wartebefehls den zweiten Programmabschnitt dazu veranlasst, mit einer Fortführung der Abarbeitung des zweiten Programmabschnitts zu warten, bis die Bedingungsvariable gesetzt ist.

In Bezug auf die Vorrichtung zum Einfügen von Synchronisationsbefehlen in Programmabschnitte eines Programms wird die Aufgabe dadurch gelöst, dass die Vorrichtung dazu vorbereitet ist, einen Freigabebefehl nach einem Schreibbefehl eines ersten Programmabschnitts auf eine Ressource einzufügen, die von dem ersten Programmabschnitt und einem zweiten Programmabschnitt gemeinsam genutzt wird, falls der Schreibbefehl auf die Ressource von dem ersten Programmabschnitt vor einem Lesebefehl des zweiten Programmabschnitts auf die Ressource auszuführen ist, wobei eine Ausführung des Freigabebefehls eine Bedingungsvariable in einen Gesetzt zustand versetzt. Dabei ist die Vorrichtung dazu vorbereitet, einen Wartebefehl zum Warten auf den Gesetzt zustand der Bedingungsvariable vor dem Lesebefehl des zweiten Programmabschnitts auf die Ressource einzufügen, wobei eine Ausführung des Wartebefehls den zweiten Programmabschnitt dazu veranlasst, mit einer Fortführung der Abarbeitung des zweiten Programmabschnitts zu warten,

bis die Bedingungsvariable gesetzt ist.

Durch das erfindungsgemäße Verfahren und die erfindungsgemäße Vorrichtung wird die parallele Version des sequentiellen Programms fehlerfrei synchronisiert. Außerdem ist das Verfahren automatisierbar, so dass es dazu beitragen kann, einen Arbeitszeitaufwand für die Erstellung eines sequentiellen Programms zu verringern.

Es ist zweckmäßig, wenn die Bedingungsvariable anlässlich einer Programminitialisierung sowie bei einem Abarbeiten des zugehörigen Wartebefehls zurückgesetzt wird. Hierdurch kann in Testläufen erkannt werden, wenn ein Freigabebefehl ungewollt von mehreren Wartebefehlen genutzt wird.

Eine bevorzugte Ausführungsform sieht vor, dass das Verfahren einen Schritt zur Erkennung einer Datenabhängigkeit zwischen parallel aufrufbaren Programmabschnitten umfasst.

Ebenso bevorzugt ist, wenn das Verfahren einen Schritt zur Feststellung umfasst, welche Paare von Schreib- und Lesebefehlen in einer vorgegebenen Reihenfolge auszuführen sind und welche Reihenfolge hierbei einzuhalten ist. Durch die vorgenannten beiden Maßnahmen wird das Datensynchronisationsverfahren einer Automatisierung zugänglich.

Es hat auch Vorteile, wenn das Verfahren einen Schritt umfasst, in welchem ein Pfad erkannt wird, der einen Pfad mit einem zu synchronisierenden Schreibbefehl auf eine Ressource überbrückt, und einen Schritt umfasst, in welchem in den überbrückenden Pfad ein Freigabebefehl für diese Ressource eingefügt wird. Hierdurch ist auch dann eine fehlerfreie Datensynchronisation möglich, wenn ein Thread eine Verzweigung enthält, die dazu führt, dass ein Schreibbefehl nur unter bestimmten Umständen durchgeführt wird.

Auch ist es bevorzugt, ressourcenspezifische oder partitionsspezifische Bedingungsvariablen vorzusehen.

Eine Optimierungsoption sieht vor, dass ein Laufzeitbedarf für unterschiedliche Alternativen von Partitionen ermittelt wird und dann die oder eine der Alternativen mit geringstem Laufzeitbedarf genutzt wird. Die beiden vorgenannten Maßnahmen können dazu beitragen, eine Rechenleistung zu verbessern, indem eine Summe von Wartezeiten der Threads verringert wird.

Es kann auch zweckmäßig sein, synchronisationsstellenspezifische Bedingungsvariablen vorzusehen. Hierdurch können bestimmte Formen einer fehlerhaften Wiederverwendung von Bedingungsvariablen von vorneherein ausgeschlossen werden.

Eine weitere Optimierungsoption sieht vor, dass das Verfahren einen Schritt umfasst, in dem eine zweite Bedingungsvariable zusammen mit ihren zugehörigen Freigabe- und Wartebefehlen entfernt wird, wenn jeder Freigabebefehl auf einer ersten Bedingungsvariable jeden Freigabebefehl auf der zweiten Bedingungsvariable postdominiert und jeder Wartebefehl auf der ersten Bedingungsvariable jeden Wartebefehl auf der zweiten Bedingungsvariable dominiert. Hierdurch kann eine Rechenleistung verbessert werden, indem Laufzeiten für Synchronisationsoperationen eingespart werden.

Die Erfindung ist anhand der beigefügten Zeichnungen näher erläutert, in denen zeigen:

FIG 1 im oberen Teil schematisch einen Ablauf eines sequentiellen Programms und im unteren Teil einen Ablauf einer parallelisierten Variante des sequentiellen Programms mit fehlerfreier Zugriffsreihenfolge;

FIG 2 schematisch einen Ablauf einer Ausführungsform eines Verfahrens zur Herstellung einer Datensynchronisation;

FIG 3 schematisch ein Beispiel eines parallelisierten Programms, in dem eine Bedingungsvariable zusammen mit zugehörigen Synchronisationsbefehlen entfernt werden kann;

FIG 4 in sequentieller (d.h. in nicht parallelisierter) Darstellung im oberen Teil schematisch eine erste Programmschleife, die einen Schreibbefehl auf eine Speicherstelle umfasst, und einen darauf folgenden Programmabschnitt, der einen Lesebefehl auf die Speicherstelle umfasst, und in dem unteren Teil schematisch eine zweite Programmschleife, die einen Lesebefehl auf eine Speicherstelle umfasst und die an einen Programmabschnitt anschließt, der einen Schreibbefehl auf die Speicherstelle umfasst.

Die nachfolgend näher geschilderten Ausführungsbeispiele stellen bevorzugte Ausführungsformen der vorliegenden Erfindung dar.

Programme für speicherprogrammierte Steuerungen greifen auf Speicherstellen, Aktoren und Sensoren zu, um technische Prozesse zu beobachten und zu steuern. In zukünftigen Generationen von speicherprogrammierten Steuerungen werden Mehrkernprozessoren zum Einsatz kommen. Um alle Kerne des Prozessors nutzen zu können, müssen Programme für speicherprogrammierte Steuerungen in parallel ablaufende Stränge (Threads) aufgeteilt werden. Dadurch besteht dann jedoch die Möglichkeit, dass die Stränge in nicht vorherbestimmter Reihenfolge auf gemeinsame Speicherstellen, Sensoren oder Aktoren zugreifen. Solche Wettlauf Situationen *(data races)* sollten vermieden werden, da sie zu inkonsistenten Programmzuständen führen

können und auch die Aussagekraft von Testergebnissen beeinträchtigen können. Um die Verständlichkeit der Figurenbeschreibung zu fördern, wird im Folgenden der Begriff 'Speicherstelle' statt 'Ressource' verwendet; dies geschieht jedoch ohne Beschränkung der allgemeineren Offenbarung. Speicherstellen, deren Inhalt während des Programmablaufs mittels Schreibbefehlen verändert wird, können auch als Variablen bezeichnet werden.

Der obere Teil von FIG 1 zeigt einen Ablauf eines sequentiellen Programms 10. Hierbei greifen zwei Programmteile PA1, PA2 nacheinander auf eine selbe i-te Speicherstelle SSi zu. Bei SSi kann es sich beispielsweise um eine externe oder um eine globale Variable handeln (dies ist aber nicht zwingend).

Ohne Beschränkung der Allgemeinheit wird im Folgenden unterstellt, dass die Programme 12 bereits parallelisiert sind (siehe unterer Teil der FIG 1). D.h. im Quelltext der Programme 12 sind bereits Anweisungen enthalten, die mindestens zwei Threads 21, 22 erzeugen. Allerdings wird unterstellt, dass Schreibbefehle 31 und Lesebefehle 32 auf gemeinsam genutzte Daten SSi durch die Parallelisierung des sequentiellen Programms 10 noch nicht synchronisiert wurden. Das heißt, sie sind noch nicht durch Synchronisationsmaßnahmen wie gegenseitigen Ausschluss geschützt.

Eine erweiterte Betrachtung ist möglich, wenn unter Speicherstellen SSi auch andere gedächtnisbehaftete Systemkomponenten subsumiert werden als nur Speicherstellen SSi eines Daten-Speichers im engeren Sinne. Außerdem muss auch nicht zwingend unterstellt werden, dass eine Speicherstelle SSi ihren Dateninhalt zwischen einem Schreibbefehl 31 und einem direkt darauffolgenden Lesebefehl 32 unverändert aufrechterhält. Eine Veränderung des Dateninhalts kann beispielsweise bauartbestimmt sein, wenn (in den Figuren nicht dargestellte) konkur-

rierende Schreibzugriffe aus einem zweiten Bussystem vorgesehen sind, oder wenn ein Zweck des Programmcodes gerade darin besteht, eine ungewollte Veränderung des Dateninhalts einer Speicherstelle SSi zu erkennen. Entsprechend kann beispielsweise auch ein (in den Figuren nicht dargestelltes) Paar aus einem Aktor und einem Sensor für die hier interessierenden Zwecke als Speicherstelle SSi betrachtet werden, wenn der Aktor (beispielsweise als Stellglied eines Regelkreises) dazu geeignet ist, eine (physikalische) Größe zu beeinflussen und der Sensor dazu vorgesehen ist, die (physikalische) Größe (beispielsweise als Regelgröße) zu messen.

Wenn im sequentiellen Code eines Programms 10 ein Programmabschnitt PA2 zeitlich nach einem Programmabschnitt PA1 auszuführen ist und das Programm 12 in parallelisierter Form fehlerfrei ausgeführt werden soll (und wenn keine weiteren Informationen vorliegen) ist grundsätzlich Folgendes sicherzustellen :

- Ein Schreibbefehl 31 des ersten Programmabschnitts PA1 auf eine erste Speicherstelle SSi darf nicht nach dem Schreibbefehl 31 des zweiten Programmabschnitts PA2 auf die Speicherstelle SSi ausgeführt werden, weil die Speicherstelle SSi sonst aus Sicht des zweiten Programmabschnitts PA2 (und auch aus Sicht evtl. weiterer nachfolgender Programmabschnitte) fehlerhaft vorbesetzt sein könnte.
- Der Lesebefehl 32 des zweiten Programmabschnitts PA2 auf die Speicherstelle SSi darf nicht vor dem Schreibbefehl 31 des ersten Programmabschnitts PA1 auf die Speicherstelle SSi ausgeführt werden, weil die Speicherstelle SSi sonst aus Sicht des zweiten Programmabschnitts PA2 (und auch aus Sicht evtl. weiterer nachfolgender Programmabschnitte) fehlerhaft vorbesetzt sein könnte.

Der untere Teil der FIG 1 zeigt einen Ablauf einer parallelisierten Variante 12 des sequentiellen Programms mit fehler-

freier Zugriffsreihenfolge.

Die FIG 2 zeigt einen Ablauf einer Ausführungsform eines Verfahrens zur Herstellung einer Datensynchronisation. Um eine fehlerfreie Zugriffsreihenfolge sicherzustellen, die eine ursprüngliche Semantik des Programms einhält, wird das im Folgenden beschriebene Verfahren 100 vorgeschlagen:

- Mittels einer bekannten Abhängigkeitsanalyse wird in einem ersten Schritt 110 (beispielsweise der in Muchnick, Steven S.: "Advanced Compiler Design & Implementation" beschriebenen Abhängigkeitsanalyse) wird untersucht, welche Datenabhängigkeiten zwischen parallel aufrufbaren Programmabschnitten PA1, PA2 (beispielsweise Programmbausteinen) bestehen .
- Unter Berücksichtigung der aufgefundenen Datenabhängigkeiten wird in einem zweiten Schritt 120 festgestellt, welche Paare von Schreib- und Lesebefehlen 31, 32 in einer vorgegebenen Reihenfolge auszuführen sind und welche Reihenfolge hierbei einzuhalten ist.
- Unter Berücksichtigung der einzuhaltenden Reihenfolge für die Schreib- und Lesebefehle 31, 32 werden in einem dritten Schritt 130 in das zu parallelisierende oder in das bereits parallelisierte Programm 12 Freigabebefehle 41 zum Setzen von Bedingungsvariablen c und Wartebefehle 42 zum Warten auf einen Gesetzt zustand der jeweiligen Bedingungsvariablen c automatisiert eingefügt (Schritte 131 und 132) .

Durch Schreib-, Lese, Freigabe- und Wartebefehle werden Schreib-, Lese, Freigabe- beziehungsweise Wartevorgänge ausgelöst. Die Freigabebefehle 41 zum Setzen von Bedingungsvariablen c und die Wartebefehle 42 zum Warten auf einen Gesetztzustand von Bedingungsvariablen c stellen bekannte Synchronisationsprimitive dar. Die Bedingungsvariable c kann beispielsweise mittels eines Flags oder in Gestalt einer Softwareinterrupt-Nachricht realisiert werden. Wartebefehle 42 kön-

nen mit Pollingverfahren (mittels Abfrage einer zugeordneten Bedingungsvariablen c) oder im Interrupt-Verfahren (mittels Empfang einer Nachricht, die von einem Freigabebefehl 41 erzeugt wird) realisiert werden. Um einen möglichst hohen Leistungsgewinn durch die Programmparallelisierung zu erzielen, ist es von Vorteil, den Freigabebefehl 41 an erstmöglicher Stelle und den Wartebefehl 42 an letztmöglicher Stelle einzufügen. Der Freigabebefehl 41 zum Setzen einer Bedingungsvariable c kann beispielsweise unmittelbar nach dem zu schützenden Schreibbefehl 31 eingefügt werden. Der Wartebefehl 42 zum Warten auf einen Gesetzt zustand der Bedingungsvariable c kann beispielsweise unmittelbar vor einem zu schützenden Lesebefehl 32 eingefügt werden.

Jede Bedingungsvariable c ist bei Programminitialisierung zurückzusetzen. Wenn es in einem parallelisierten Programm 12 für eine selbe Speicherstelle SSi mehrere Synchronisations – stellen 5j gibt (siehe FIG 1), an denen für diese Speicherstelle SSi auf eine Freigabe (Gesetzt zustand der Bedingungsvariable c) gewartet werden muss (dies ist der Allgemeinfall), muss die Bedingungsvariable c auch beim Durchlaufen des zugehörigen Wartebefehls 42 zurückgesetzt werden, damit eine zeitlich hinterher durchlaufene Synchronisationsstelle 5j nicht aufgrund der bereits gesetzten Bedingungsvariable c synchronisationslos durchlaufen wird.

Der erste Programmabschnitt PA1 kann eine Verzweigung enthalten, die bewirkt, dass der Pfad mit dem Schreibbefehl 31 nur unter einer bestimmten Randbedingung durchlaufen wird (die mehrere Randbedingungen umfassen kann) und ansonsten ein anderer Pfad durchlaufen wird. In diesem Fall müssen gleichwirkende Bedingungsvariablen c auch in alle diejenigen Programmpfade des ersten Programmabschnitts PA1 eingefügt werden, die durchlaufen werden, ohne dass in ihnen ein Schreibbefehl 31 durchgeführt wird. So wird verhindert, dass der zweite Pro-

grammabschnitt PA2 mit dem Wartebefehl 42 unendlich lange auf eine Freigabe (Gesetztzustand der Bedingungsvariable c) wartet, wenn der erste Programmabschnitt PA1 auf Pfaden durchlaufen wird, in denen der Schreibbefehl 31 nicht durchgeführt wird.

Unabhängig davon kann es ratsam sein, Freigabebefehle 41, Wartebefehle 42 und Bedingungsvariablen c vorzusehen, die für jede Synchronisationsstelle 5j spezifisch sind, und zwar auch dann, wenn an unterschiedlichen Synchronisationsstellen 5j Schreib- und Lesebefehle 31, 32 auf eine selbe Speicherstelle SSi zu synchronisieren sind. Es handelt sich dann also nicht nur um eine speicherstellenspezifische, sondern auch um eine synchronisationsstellenspezifische Synchronisation.

Anstatt einer einzelnen Speicherstelle SSi kann eine speicherstellenspezifische Bedingungsvariable c auch einer Partition (mit anderen Worten Teilmenge, Gruppe oder Freigabegruppe) von Speicherstellen SSi zugeordnet sein. In diesem Fall darf der Freigabebefehl 41 zum Setzen der partitionsspezifischen Bedingungsvariable c erst eingefügt werden, wenn in vorher auszuführenden Programmabschnitten die Schreibbefehle 31 für alle Speicherstellen SSi der Partition abgeschlossen sind. Umgekehrt muss in diesem Fall der Wartebefehl 42 eingefügt werden, bevor in nachher auszuführenden Programmabschnitten PA2 irgendein Lesebefehl 32 für eine der Speicherstellen SSi der Partition ausgeführt wird. Im Extremfall kann die Partition alle Speicherstellen SSi umfassen, die irgendwo im Programmablauf zu synchronisieren sind. In diesem Fall braucht die Bedingungsvariable c nicht speicherstellenspezifisch zu sein.

Es kann vorgesehen sein, dass die Vorrichtung, welche die Freigabebefehle 41 und die Wartebefehle 42 einfügt, den Laufzeitbedarf für unterschiedliche Alternativen von Partitionen

ermittelt und dann diejenige Alternative oder eine der Alternativen mit geringstem Laufzeitbedarf nutzt.

Eine weitere Optimierungsmöglichkeit besteht in einer Verwendung von Dominatoren und Postdominatoren in einem Kontrollflussgraphen zur Erkennung und Beseitigung einer Redundanz in der Datensynchronisation. Die Begriffe 'dominieren' und 'postdominieren' sind wie folgt definiert. Ein erster Befehl A dominiert einen zweiten Befehl B, wenn der erste Befehl A immer garantiert vor dem zweiten Befehl B ausgeführt wird. Ein erster Befehl A postdominiert einen zweiten Befehl B, wenn der erste Befehl A immer garantiert nach dem zweiten Befehl B ausgeführt wird.

Zunächst werden Synchronisationsbefehle (d.h. Freigabe- und Wartebefehle) mit je einer Bedingungsvariable c je gemeinsam genutzter Speicherstelle SSi (Variable) eingefügt. In den so synchronisierten Programmabschnitten werden nun Dominatoren und Postdominatoren berechnet. Postdominiert jeder Freigabebefehl 41 auf einer ersten Bedingungsvariable c jeden Freigabebefehl 41' auf einer zweiten Bedingungsvariable c', und dominiert jeder Wartebefehl 42 auf der ersten Bedingungsvariable c jeden Wartebefehl 42' auf der zweiten Bedingungsvariable c', dann kann die zweite Bedingungsvariable c' zusammen mit ihren zugehörigen Freigabebefehlen 41' und Wartebefehlen 42' entfernt werden.

In dem in der FIG 3 gezeigten Beispiel kann der Lesebefehl 32' erst ausgeführt werden, nachdem der Freigabebefehl 41 auf der ersten Bedingungsvariable c ausgeführt wurde. Der nachfolgende Wartebefehl 42', dessen zugehöriger Freigabebefehl 41' vor dem Freigabebefehl 41 ausgeführt wird, ist also überflüssig (der Freigabebefehl 41 postdominiert Freigabebefehl 41', und der Wartebefehl 42 dominiert den Wartebefehl 42'). Damit kann die zweite Bedingungsvariable c' zusammen mit den

zugehörigen Synchronisationsbefehlen 41', 42' entfernt werden. Da auch die Synchronisationsbefehle 41', 42' Laufzeitkosten verursachen, kann dies zur Verringerung der Rechenzeit beitragen .

In dem oberen Teil der FIG 4 ist eine (nicht parallelisierte) erste Programmschleife 81 dargestellt, die einen Schreibbefehl 31 auf eine Speicherstelle SSi umfasst, und ein darauf folgender Programmabschnitt 84, der einen Lesebefehl 32 auf die Speicherstelle SSi umfasst.

Bei Programmschleifen 81, 82 ist zu beachten, dass sie eine Notation für eine Wiederholung des Schleifeninhalts 80, d.h. eine Notation für eine Kette von wiederholten Ausführungen des Schleifeninhalts 80 darstellen. Ein Programmabschnitt 84, der auf eine erste Programmschleife 81 folgt, schließt nicht an jeden Schleifendurchlauf der ersten Programmschleife 81 an, sondern nur an den zuletzt durchgeführten Schleifendurchlauf. Umgekehrt geht ein Programmabschnitt 84, an den eine zweite Programmschleife 82 anschließt, nicht jedem Schleifendurchlauf der Programmschleife 82 voraus, sondern nur dem zuerst durchgeführten Schleifendurchlauf .

Wenn ein Programmabschnitt 84, der auf eine erste Programmschleife 81 folgt, aus einer Speicherstelle SSi liest, die innerhalb der Programmschleife 81 beschrieben wird, muss also eine Ausführung des Programmabschnitts 84, der auf die erste Programmschleife 81 folgt, so lange angehalten werden, bis das Beschreiben der Speicherstelle SSi im letzten Schleifendurchgang abgeschlossen ist. Dazu kann das Setzen der Bedingungsvariablen c (die Ausführung des Freigabebefehls 41) von der Bedingung abhängig gemacht werden, dass es sich um den letzten Schleifendurchgang handelt. Unter Beachtung dieser Maßgabe ist es nicht unbedingt erforderlich, den Freigabebefehl 41 ans Ende 86 der Programmschleife 81 zu setzen. Um an

die Bedingtheit des Freigabebefehls 41 zu erinnern, ist der Freigabebefehl 41 in der FIG 4 gestrichelt gezeichnet.

Es kann jedoch unwirtschaftlich sein, wegen der Bedingtheit des Freigabebefehls 41 die Abbruchbedingung für die erste Programmschleife 81 auszuwerten. Es kann auch sein, dass aus Übersichtlichkeitsgründen eine Überwachung der Abbruchbedingung zu diesem Zweck unerwünscht ist (beispielsweise bei einer while-, for-each- oder forever-Schleife oder wenn die Abbruchentscheidung von einem variablen Abbruchkriterium abhängig ist, dessen Entscheidungsparameter erst zum Ende 86 der erste Programmschleife 81 und nicht schon zum Zeitpunkt des Freigabebefehls 41 vorliegen) . Es gibt auch die Alternative, den Freigabebefehl 41 zum Setzen der Bedingungsvariable c unmittelbar hinter oder an das Ende 86 der vorher auszuführenden Programmschleife 81 einzufügen.

Umgekehrt gibt es auch den Fall, dass ein Schleifeninhalt 80 innerhalb einer (nicht parallelisierten) zweiten Programmschleife 82 aus einer Speicherstelle SSi liest, die von einem vorausgehenden Programmabschnitt 84 beschrieben wurde. In dem unteren Teil der FIG 4 ist eine zweite Programmschleife 82 dargestellt, die einen Lesebefehl 32 auf eine Speicherstelle SSi umfasst und an den vorausgehenden Programmabschnitt 84 anschließt, der einen Schreibbefehl 31 auf die Speicherstelle SSi umfasst. Es muss sichergestellt werden, dass die Bedingungsvariable c bei Ausführung des Wartebefehls 42 zwar gelesen und beachtet wird, aber erst anlässlich einer letztmaligen Ausführung der zweiten Programmschleife 82 gelöscht wird. Dazu kann das Löschen der Bedingungsvariable c von der Bedingung abhängig gemacht werden, dass es sich um den letzten Schleifendurchgang handelt. Dann ist es nicht unbedingt erforderlich, den Wartebefehl 42 an den Beginn 85 der Programmschleife 82 zu setzen. Um an die Bedingtheit des Löschens der Bedingungsvariable c durch den Wartebefehl 42 zu erinnern,

ist der Wartebefehl 42 in der FIG 4 gestrichelt gezeichnet.

Es kann jedoch unwirtschaftlich sein, dafür die Abbruchbedingung für die zweite Programmschleife 82 auszuwerten. Beispielsweise kann aus Übersichtlichkeitsgründen eine Überwachung der Abbruchbedingung zu diesem Zweck unerwünscht sein (beispielsweise bei einer while-, for-each- oder forever-Schleife oder wenn die Abbruchentscheidung von einem variablen Abbruchkriterium abhängig ist, dessen Entscheidungsparameter erst zum Ende 86 der Programmschleife 82 und nicht schon zum Zeitpunkt des Wartebefehls 42 vorliegen) . Außerdem kann es unerwünscht sein, zwischen solchen Wartebefehlen 42 zu unterscheiden, mit denen die zugehörige Bedingungsvariable c zurückgesetzt wird und solchen, mit denen die zugehörige Bedingungsvariable c nicht zurückgesetzt wird. Es gibt auch die Alternative, den Wartebefehl 42 zum Warten auf die Bedingungsvariable c unmittelbar vor oder an den Beginn 85 der nachher auszuführenden Programmschleife 82 einzufügen.

Bezugszeichenliste

10	sequentielles Programm
12	parallelisiertes Programm
21	erster Thread
22	zweiter Thread
31	Schreibbefehl; Schreibzugriff
32	Lesebefehl; Lesezugriff
41	Freigabebefehl
42	Wartebefehl
5j	j-te Synchronisationsstelle
80	Schleifeninhalt
81	erste Schleife
82	zweite Schleife
84	Programmabschnitt
85	Beginn der Programmschleife
86	Ende der Programmschleife
PA1	erster Programmabschnitt
PA2	zweiter Programmabschnitt
SSi	i-te Speicherstelle; i-te Ressource
c	Bedingungsvariable
100	Verfahren
110	Schritt mit Abhängigkeitsanalyse
120	Schritt mit Analyse der Datenabhängigkeiten
130	Schritt des Einfügens von Synchronisationsbefehlen
131	automatisiertes Einfügen eines Freigabebefehls
132	automatisiertes Einfügen eines Wartebefehls

Patentansprüche

1. Verfahren (100) zum Einfügen von Synchronisationsbefehlen (31, 32) in Programmabschnitte (PA1, PA2) eines Programms (10, 12),
dadurch gekennzeichnet, dass
das Verfahren (100) folgende Schritte (131, 132) umfasst:
- automatisiertes Einfügen (131) eines Freigabebefehls (41) nach einem Schreibbefehl (31) eines ersten Programmab-Schnitts (PA1) auf eine Ressource (SSi), die von dem ersten Programmabschnitt (PA1) und einem zweiten Programmabschnitt (PA2) gemeinsam genutzt wird, falls der Schreibbefehl (31) auf die Ressource (SSi) von dem ersten Programmabschnitt (PA1) vor einem Lesebefehl (32) des zweiten Programmabschnitts (PA2) auf die Ressource (SSi) auszuführen ist, wobei eine Ausführung des Freigabebefehls (41) eine Bedingungsvariable (c) in einen Gesetzt zustand versetzt; und
- automatisiertes Einfügen (132) eines Wartebefehls (42) zum Warten auf den Gesetzt zustand der Bedingungsvariable (c) vor dem Lesebefehl (32) des zweiten Programmabschnitts (PA2) auf die Ressource (SSi), wobei eine Ausführung des Wartebefehls (42) den zweiten Programmabschnitt (PA2) dazu veranlasst, mit einer Fortführung der Abarbeitung des zweiten Programmabschnitts (PA2) zu warten, bis die Bedingungsvariable (c) gesetzt ist.

2. Verfahren (100) nach Ansprüche 1,
dadurch gekennzeichnet, dass
das die Bedingungsvariable (c) anlässlich einer Programminitialisierung sowie bei einem Abarbeiten des zugehörigen Wartebefehls (42) zurückgesetzt wird.

3. Verfahren (100) nach Anspruch 1 oder 2,
dadurch gekennzeichnet, dass

das Verfahren (100) einen Schritt (110) zur Erkennung einer Datenabhängigkeit zwischen parallel aufrufbaren Programmabschnitten (PA1, PA2) umfasst.

4. Verfahren (100) nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass
das Verfahren (100) einen Schritt (120) zur Feststellung umfasst, welche Paare von Schreib- und Lesebefehlen (31, 32) in einer vorgegebenen Reihenfolge auszuführen sind und welche Reihenfolge hierbei einzuhalten ist.

5. Verfahren (100) nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, dass
das Verfahren (100) einen Schritt umfasst, in welchem ein Pfad erkannt wird, der einen Pfad mit einem zu synchronisierenden Schreibbefehl (31) auf eine Ressource (SSi) überbrückt, und einen Schritt umfasst, in welchem in den überbrückenden Pfad ein Freigabebefehl (41) für diese Ressource (SSi) eingefügt wird.

6. Verfahren (100) nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, dass
ressourcenspezifische oder partionsspezifische Bedingungsvariablen (c) vorgesehen sind.

7. Verfahren (100) nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass
ein Laufzeitbedarf für unterschiedliche Alternativen von Partitionen ermittelt wird und dann die oder eine der Alternativen mit geringstem Laufzeitbedarf genutzt wird.

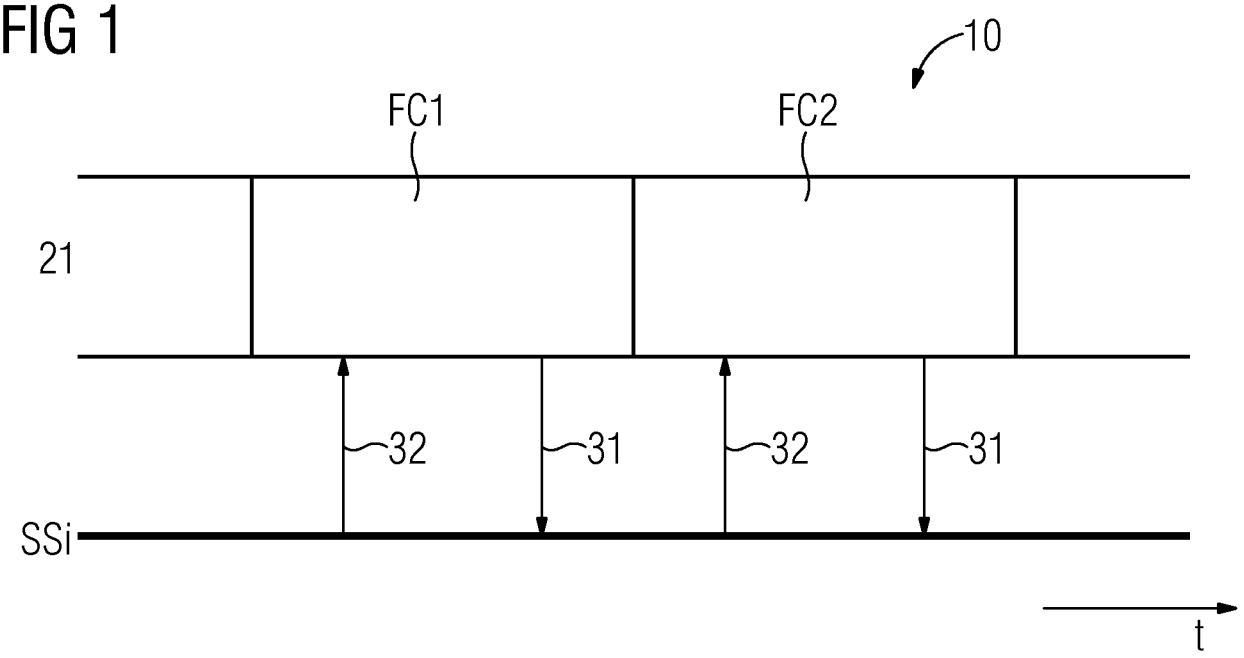
8. Verfahren (100) nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, dass
synchronisationsstellenspezifische Bedingungsvariablen (c) vorgesehen sind.

9. Verfahren (100) nach einem der Ansprüche 1 bis 8, dadurch gekennzeichnet, dass
das Verfahren (100) einen Schritt umfasst, in dem eine zweite Bedingungsvariable (c') zusammen mit ihren zugehörigen Freigabebefehlen (41') und Wartebefehlen (42') entfernt wird, wenn jeder Freigabebefehl (41) auf einer ersten Bedingungsvariable (c) jeden Freigabebefehl (41') auf der zweiten Bedingungsvariable (c') postdominiert und jeder Wartebefehl (42) auf der ersten Bedingungsvariable (c) jeden Wartebefehl (42') auf der zweiten Bedingungsvariable (c') dominiert.

10. Vorrichtung (100) zum Einfügen von Synchronisationsbefehlen in Programmabschnitte (PA1, PA2) eines Programms (100), dadurch gekennzeichnet,

- dass die Vorrichtung dazu vorbereitet ist, einen Freigabebefehl (41) nach einem Schreibbefehl (31) eines ersten Programmabschnitts (PA1) auf eine Ressource (SSi) einzufügen, die von dem ersten Programmabschnitt (PA1) und einem zweiten Programmabschnitt (PA2) gemeinsam genutzt wird, falls der Schreibbefehl (31) auf die Ressource (SSi) von dem ersten Programmabschnitt (PA1) vor einem Lesebefehl (32) des zweiten Programmabschnitts (PA2) auf die Ressource (SSi) auszuführen ist, wobei eine Ausführung des Freigabebefehls (41) eine Bedingungsvariable (c) in einen Gesetztzustand versetzt; und
- dass die Vorrichtung dazu vorbereitet ist, einen Wartebefehl (42) zum Warten auf den Gesetztzustand der Bedingungsvariable (c) vor dem Lesebefehl (32) des zweiten Programmabschnitts (PA2) auf die Ressource (SSi) einzufügen, wobei eine Ausführung des Wartebefehls (42) den zweiten Programmabschnitt (PA2) dazu veranlasst, mit einer Fortführung der Abarbeitung des zweiten Programmabschnitts (PA2) zu warten, bis die Bedingungsvariable (c) gesetzt ist.

FIG 1


10
FC1
FC2
21
32
31
32
31
SSi
t

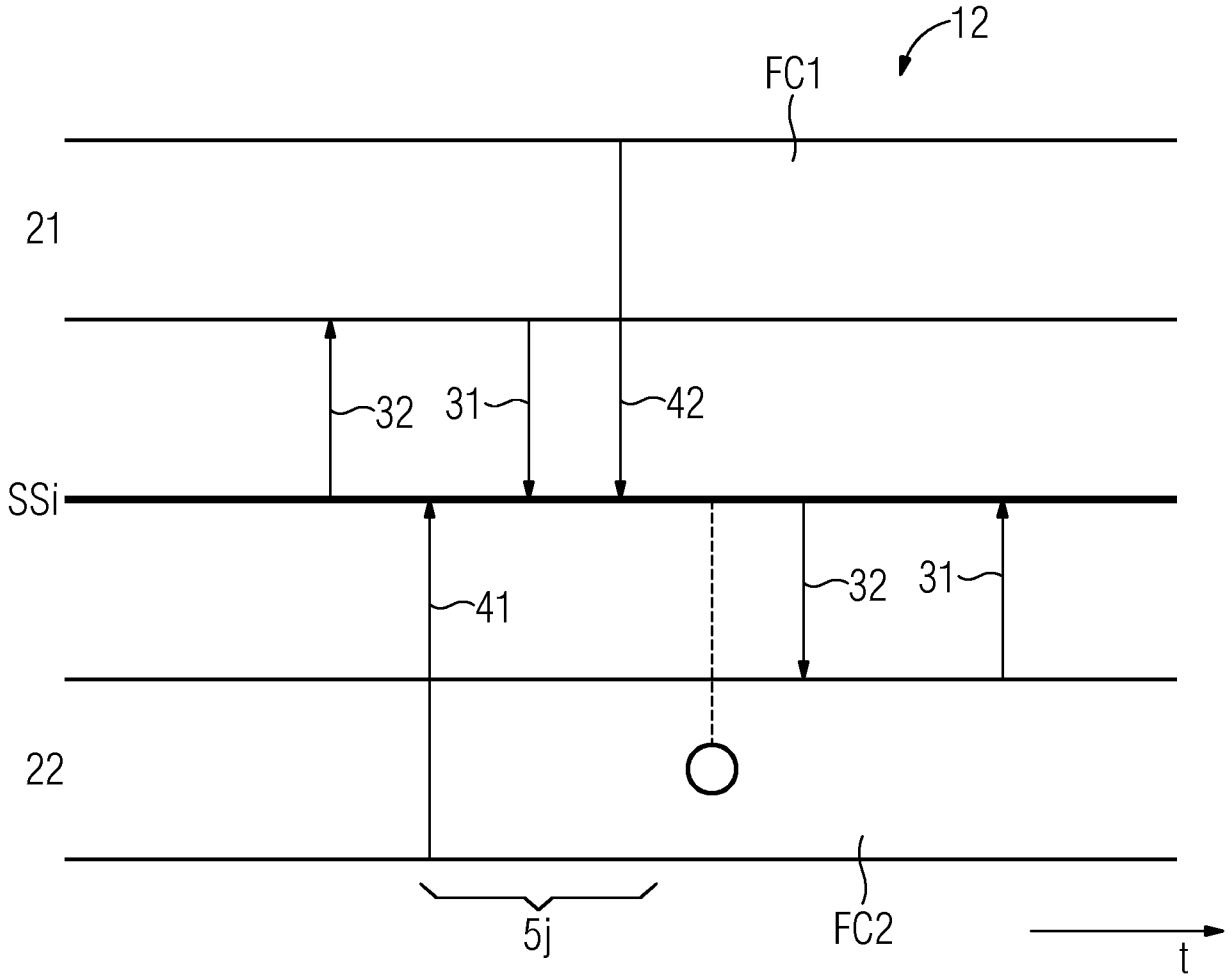

12
FC1
21
32
31
42
SSi
41
32
31
22
5j
FC2
t

FIG 2

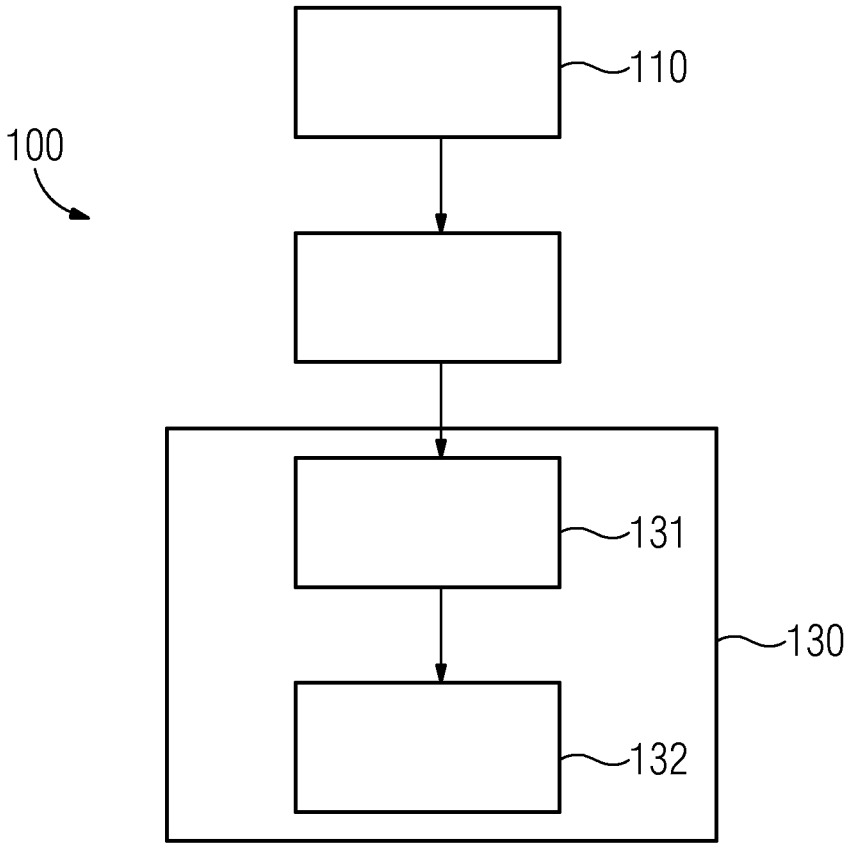


FIG 3

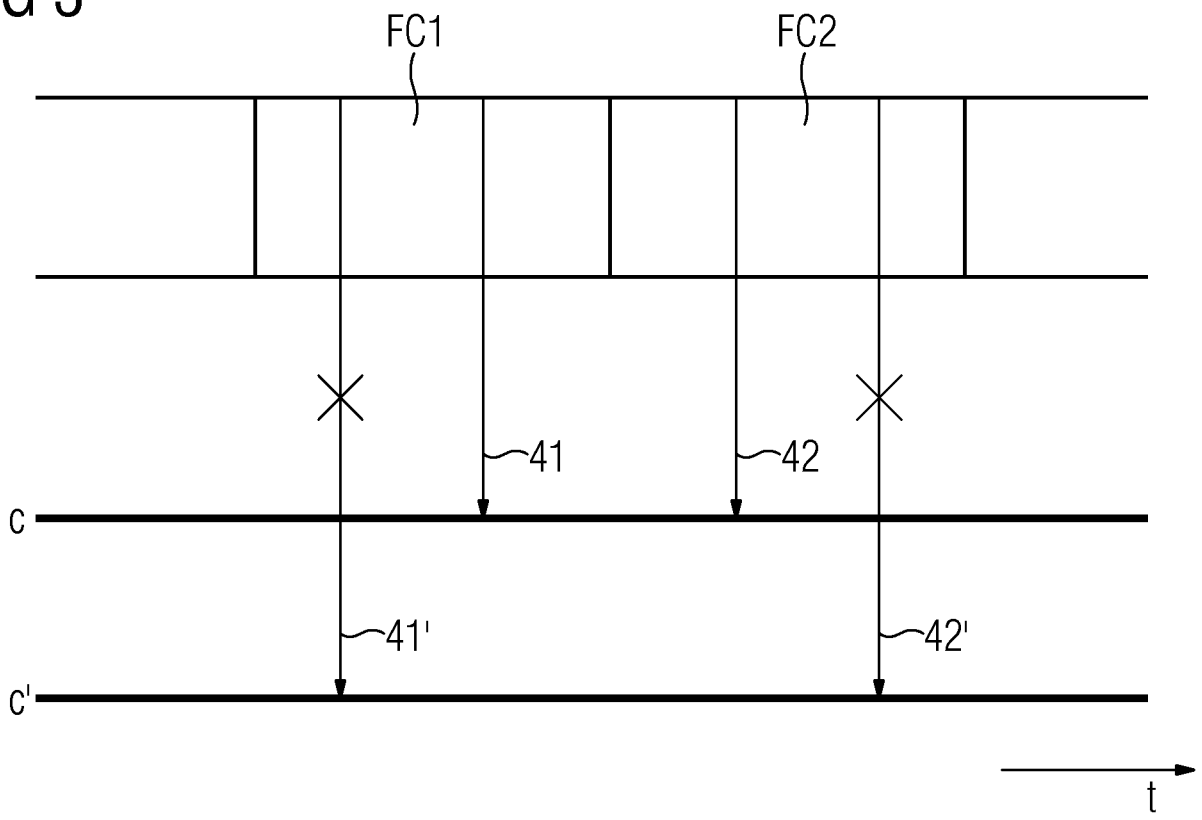
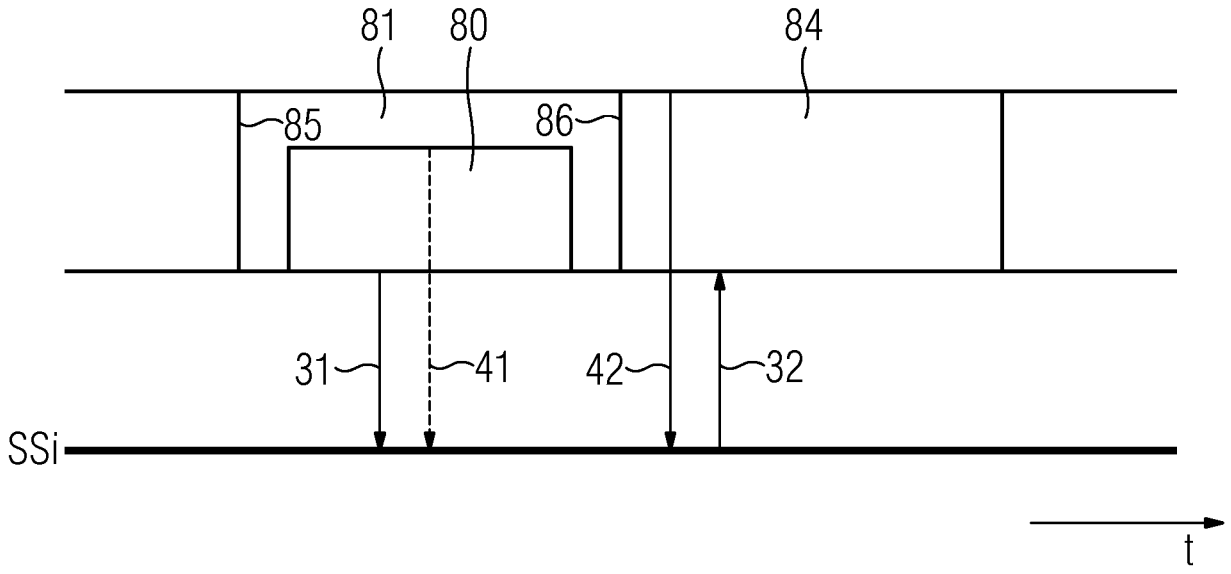
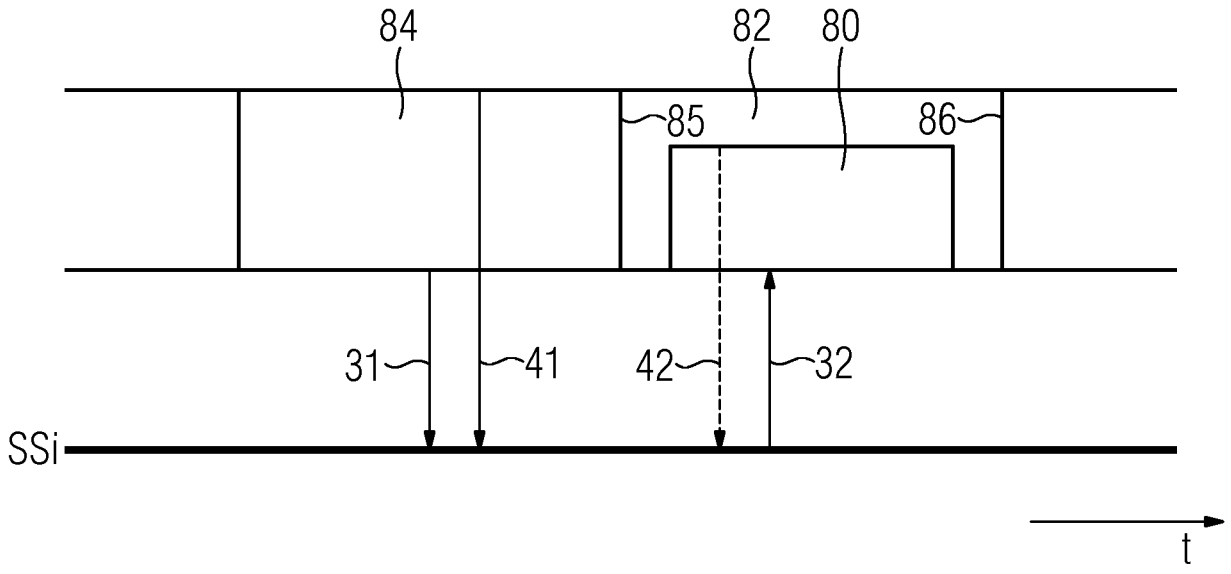


FIG 4





INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2012/062571

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/45
ADD.

According to International Patent Classification (IPC) orto both national Classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (Classification System followed by Classification Symbols)
G06F

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
X	US 2005/108695 A1 (LI LONG [CN] ET AL) 19 May 2005 (2005-05-19) abstract; figures 5,7,10,16 paragraphs [0001] - [0002], [0029], [0032], [0034] - [0036], [0039], [0041] - [0042], [0051], [0069] -----	1-10

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

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general State of the art which is not considered to be of particular relevance

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

"L" documentwhich may throw doubts on priority claim(s) orwhich is cited to establish the publication date of another citation or other special reason (as specified)

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 October 2012	06/11/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Steinmetz, Christof

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2012/062571

Patent document cited in search report	Publioation date	Patent family member(s)		Publioation date
US 2005108695 AI	19-05-2005	AT	453894 T	15-01-2010
		CN	1906578 A	31-01-2007
		EP	1683010 A2	26-07-2006
		US	2005108695 AI	19-05-2005
		WO	2005050445 A2	02-06-2005

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2012/062571

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
INV. G06F9/45
ADD.

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter MindestprUfstoff (Klassifikationssystem und Klassifikationssymbole)
G06F

Recherchierte, aber nicht zum MindestprUfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	US 2005/108695 AI (LI LONG [CN] ET AL) 19. Mai 2005 (2005-05-19) Zusammenfassung; Abbildungen 5,7,10,16 Absätze [0001] - [0002], [0029], [0032], [0034] - [0036], [0039], [0041] - [0042] , [0051] , [0069] -----	1-10

☐ Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen ☒ Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :

"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist

"E" frühere Anmeldung oder Patent, die bzw. das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist

"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)

"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht

"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist

"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist

"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden

"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist

"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
23. Oktober 2012	06/11/2012

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Steinmetz, Christof

Formblatt PCT/ISA/210 (Blatt 2) (April 2005)

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/EP2012/062571

Im Recherchenbericht angeführtes Patentdokument	Datum der Veröffentlichung	Mitglied(er) der Patentfamilie	Datum der Veröffentlichung
US 2005108695 AI	19-05-2005	AT 453894 T	15-01-2010
		CN 1906578 A	31-01-2007
		EP 1683010 A2	26-07-2006
		US 2005108695 AI	19-05-2005
		WO 2005050445 A2	02-06-2005