(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 600 24 421 T2** 2006.08.03

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 164 493 B1**
(21) Deutsches Aktenzeichen: **600 24 421.0**
(96) Europäisches Aktenzeichen: **00 830 424.8**
(96) Europäischer Anmeldetag: **16.06.2000**
(97) Erstveröffentlichung durch das EPA: **19.12.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **30.11.2005**
(47) Veröffentlichungstag im Patentblatt: **03.08.2006**

(51) Int Cl.[8]: ***G06F 13/364*** (2006.01)

(73) Patentinhaber:
STMicroelectronics S.r.l., Agrate Brianza, Mailand/Milano, IT

(74) Vertreter:
Kehl & Ettmayr, Patentanwälte, 81679 München

(84) Benannte Vertragsstaaten:
DE, FR, GB, IT

(72) Erfinder:
Butta, Pasquale, 98100 Messina, IT; Marty, Pierre, 38180 Seyssins, FR

(54) Bezeichnung: **Arbitrierungsverfahren mit variablen Prioritäten, zum Beispiel für Verbindungsbusse, und entsprechendes System**

Beschreibung

Gebiet der Erfindung

[0001] Die vorliegende Erfindung bezieht sich auf Arbitrierungsverfahren, die im Rahmen von Informationsverarbeitungssystemen implementiert werden.

Beschreibung des Standes der Technik

[0002] Die zunehmende Komplexität hinsichtlich der Architektur von Informationsverarbeitungssystemen und insbesondere der so genannten "eingebetteten" Systeme macht die Optimierung der Zyklen der Übertragung (Transfer) des Datenstroms von den so genannten "Initiatoren" zu den so genannten "Zielen" (Targets) eines Systems immer wichtiger.

[0003] Allgemein ist definitionsgemäß ein Initiator eine beliebige Einheit, die autonom eine Transaktion einleiten kann, mit der sie die Benutzung einer Ressource (zum Beispiel eines dedizierten oder gemeinsam genutzten Busses) beansprucht, um Daten zu lesen oder zu schreiben.

[0004] Definitionsgemäß ist ein Ziel eine beliebige Einheit, von der ausgehend oder mit der ein Initiator Daten lesen oder schreiben kann.

[0005] Ein System, wie zum Beispiel ein eingebettetes System, kann eine große Anzahl von Initiatoreinheiten und auch eine große Anzahl von Zieleinheiten aufweisen.

[0006] Zu dem Zeitpunkt, in dem zwei oder mehr Initiatoren gleichzeitig versuchen, eine Transaktion zu starten, die auf ein identisches Ziel gerichtet ist, ist es notwendig, den Konflikt mittels einer Arbitrierungsfunktion zu lösen. Die Implementierung einer derartigen Funktion kann, insbesondere hinsichtlich der Ausführungsgeschwindigkeit, oft sehr aufwendig sein.

[0007] In Fig. 2 der beiliegenden Zeichnungen ist ein Arbitrierungsvorgang gezeigt, der gemäß einer Lösung des Standes der Technik im Rahmen eines gattungsgemäßen Systems (zum Beispiel des eingebetteten Typs) durchgeführt wird, das in Fig. 1 dargestellt ist, wobei sich diese Figur selbst sowohl auf den Stand der Technik als auch auf die vorliegende Erfindung bezieht.

[0008] Zum Zeitpunkt des Beginns der entsprechenden Transaktion legt jeder Initiator I1, I2, I3 usw. dem so genannten Systemarbitrator A eine Priorität vor, der den Konflikt dadurch löst, dass er den Beginn der Transaktionen denjenigen Initiatoren ermöglicht, welche die höchste Priorität haben, und einen nach dem anderen zu denjenigen mit der niedrigsten Priorität fortschreitet. Datentransaktionen zu einem beliebigen Ziel T werden daher durch den Arbitrator A geregelt, der die möglichen Konflikte mittels der Prioritätsinformation löst, die von den Initiatoren selbst geliefert wird. Eine Transaktion besteht aus dem Transfer eines Datenpakets vom beteiligten Initiator zum Ziel T oder umgekehrt.

[0009] Zum Zeitpunkt des Beginns der Transaktion legt jeder Initiator I1, I2, I3 usw. dem Arbitrator A ein Anfragesignal vor. Der Arbitrator gibt mittels eines Bewilligungssignals, das dem Initiator mit der höchsten Priorität, unter denjenigen, welche gleichzeitig die Transaktion beginnen möchten, zugesendet wird, zu erkennen, dass die Transaktion angenommen wurde. Dann signalisiert er in den nachfolgenden Zyklen die Annahme der Transaktionen an diejenigen Initiatoren, welche niedrigere Prioritäten haben, wenn die Transaktionen mit den höheren Prioritäten abgeschlossen sind.

[0010] Dieser Vorgang ist in Fig. 2 anhand eines Systems schematisch gezeigt, das drei Initiatoren I1, I2, I3 und ein Ziel T aufweist. Der Arbitrator A, der gemäß dem im Diagramm in Fig. 2 oben gezeigten Taktsignal betrieben wird, löst jegliche Konflikte auf der Grundlage der Prioritäten, die zu den in der Figur gezeigten Zeiten empfangen wurden. All dies erfolgt, indem im Grunde genommen ein Mechanismus zum Vergleich der Ungleichheiten der Reihenfolge angewendet wird, wie er in Fig. 2 dargestellt ist. Hierbei wird angenommen, dass der Initiator I1 eine höhere Priorität als der Initiator I2 hat, der seinerseits eine höhere Priorität als der Initiator I3 hat, so dass Bewilligungen in der Reihenfolge zuerst dem Initiator I1, dann dem Initiator I2 und schließlich dem Initiator I3 gegeben werden.

[0011] Auch wenn der oben beschriebene Mechanismus funktioniert, so hat er doch intrinsische Grenzen hinsichtlich der Ausführungsgeschwindigkeit des Arbitrierungsalgorithmus, der Empfindlichkeit der Ausführungsgeschwindigkeit der Arbitrierung gegenüber der Codierung der Priorität (typischerweise eine generische Anzahl n von Bits) und hinsichtlich einer merklichen Erhöhung der Ausführungsverzögerung mit der Erhöhung der Anzahl von Initiatoren. Aus der EP 552 507 A geht ein Arbitrierungssystem hervor, bei dem relative Prioritätssignale aus Teilmengen von Eingabesignalen abgeleitet werden. Nach der Konfliktlösung wird ein absolutes Prioritätssignal erzeugt.

Aufgaben und Zusammenfassung der Erfindung

[0012] Aufgabe der vorliegenden Erfindung ist es, eine Lösung vorzusehen, welche die oben erwähnten Nachteile überwinden kann.

[0013] Erfindungsgemäß wird diese Aufgabe anhand eines Arbitrierungsverfahrens gelöst, das die spezifisch in den beiliegenden Ansprüchen geforder-

ten Merkmale hat. Die Erfindung bezieht sich auch auf das entsprechende System.

[0014] Das erfindungsgemäße Arbitrierungsverfahren ist im Wesentlichen ein Arbitrierungsmechanismus mit variabler Priorität, der vorteilhafterweise in Hochgeschwindigkeitsanwendungen, wie zum Beispiel in Decodern für HDTV-Signale, verwendet werden kann. Das entsprechende Verbindungsuntersystem, das drei Initiatoren bzw. anfordernde Einheiten (LMI, Aufwärtsschnittstelle und GPx-Schnittstelleneinheiten) und ein Ziel T (die SDRAM-Speicherschnittstelle) umfasst, kann mit einem Taktsignal, zum Beispiel mit 100 MHz, betrieben werden. Die erfindungsgemäße Lösung ermöglicht es, zwischen Prozessen zu unterscheiden, die unterschiedliche Prioritäten haben, die über denselben Initiator auf den externen SDRAM zugreifen können.

[0015] Allgemein kann die erfindungsgemäße Lösung jedoch auch in N × M-Systemen (d.h. mit N Initiatoren und M Zielen) eingesetzt werden, indem für jede Zieladresse ein Arbitrator eingeführt wird. Die Ausführungszeit des Vorgangs verlängert sich nur auf Grund der Decodierung der Zieladresse; das heißt – unter Bezugnahme auf aktuelle Technologien (wie zum Beispiel die 0,25-Mikrometer-Technologie) – mit Werten von wenigen Zehnteln von Nanosekunden für jede Zieladresse.

Kurze Beschreibung der Zeichnungen

[0016] Es folgt eine Beschreibung der vorliegenden Erfindung lediglich als ein nicht einschränkendes Beispiel anhand der beiliegenden Zeichnungen.

[0017] Die Fig. 1 und Fig. 2, wobei sich die letztere spezifisch auf den Stand der Technik bezieht, wurden schon im Voraus beschrieben; und

[0018] Fig. 3 zeigt in der Form eines Blockdiagramms, die mögliche Architektur eines Arbitrierungssystems, das erfindungsgemäß betrieben wird.

Detaillierte Beschreibung einer bevorzugten Ausführungsform der Erfindung

[0019] Die erfindungsgemäße Lösung ist dazu konstruiert, dass sie in einer Situation betrieben wird, die hinsichtlich der Aspekte allgemeinerer Art der in Fig. 1 dargestellten Lösung entspricht.

[0020] Insbesondere stellt das Blockdiagramm von Fig. 3 die Architektur des Arbitrators A dar, der zum Implementieren der erfindungsgemäßen Lösung in einem eingebetteten System eingesetzt werden kann, bei dem drei Initiatoren I1, I2 und I3, die konstruktionsgemäß auf ein einziges Ziel T zugreifen, vorhanden sind.

[0021] In der Praxis umfasst die Architektur des in Fig. 3 dargestellten Arbitrators drei Stufen, die mit A1, A2 und A3 bezeichnet und dazu konstruiert sind, in einer abgestuften Anordnung drei aufeinanderfolgende Schritte des Arbitrierungsvorgangs durchzuführen.

[0022] Die oben erwähnten Schritte (und folglich die entsprechenden Stufen A1, A2 und A3) können allgemein wie folgt identifiziert werden:

– Vergleich von Prioritäten (Stufe A1);
– Überkreuz-Erzeugung von Anfragen (Stufe A2); und
– Überkreuz-Erzeugung von Bewilligungen (Stufe A3).

[0023] Jede der Stufen A1, A2 und A3 weist ihrerseits drei Module oder Blöcke auf, die im Wesentlichen miteinander identisch sind.

[0024] Die drei Blöcke der Stufe A1, die mit A11, A12 und A13 bezeichnet sind, empfangen als ihre Eingaben die Prioritätssignale oder Daten, die einem entsprechenden Paar von Initiatoren entsprechen; diese sind (in der gezeigten Ausführungsform) in dieser Reihenfolge das Paar I1, I2, das Paar I1, I3 und das Paar I2, I3.

[0025] Die entsprechenden Ausgabesignale werden in dieser Reihenfolge an die Module der Stufe A2, die mit A21, A22 und A23 bezeichnet sind, gesendet. Die letzteren empfangen in der entsprechenden Reihenfolge ebenfalls die von den Initiatoren des Paares I1, I2, des Paares I1, I3 bzw. des Paares I2, I3 erzeugten Anfragesignale.

[0026] Jedes der Module A21, A22 und A23 erzeugt als seine Ausgabe zwei Überkreuz-Anfrage-Signale, die ihrer Bestimmung gemäß an die drei Module der Stufe A3, die in ihrer Reihenfolge mit A31, A32 und A33 bezeichnet sind, zu senden sind.

[0027] Insbesondere wird von den beiden Ausgaben der Module A21, eine an das Modul A31 und die andere an das Modul A32 gesendet. Von den beiden Ausgaben der Module A22 wird eine an das Modul A31 und die andere an das Modul A33 gesendet. Schließlich wird von den beiden Ausgaben des Moduls A23 eine an das Modul A32 und die andere an das Modul A33 gesendet.

[0028] Die Ausgabesignale der Module A31, A32 und A33 stellen die Bewilligungssignale dar, welche die Ergebnisse des Arbitrierungsvorgangs repräsentieren, und werden ihrer Bestimmung gemäß dem Initiator I1, dem Initiator I2 bzw. dem Initiator I3 zugewiesen.

[0029] Im Folgenden wird in der vorliegenden Beschreibung der erfindungsgemäße Arbitrierungsvor-

gang jedoch unter allgemeiner Bezugnahme auf das generische Vorhandensein von N Initiatoren bzw. anfragenden Einheiten beschrieben.

Stufe Nr. 1: Vergleich

[0030] Angenommen, es gibt N anfordernde Einheiten und der i-ten anfordernden Einheit ist die Priorität Pi zugeordnet (zum Beispiel ist P1 die Priorität für die anfordernde Einheit 1), werden die folgenden Operationen parallel durchgeführt.
P1 >= P2 (erzeugt das Signal sel1not2, hoch, wenn das Ergebnis der Operation wahr ist)
P1 >= P3 (erzeugt das Signal sel1not3)
...
P1 >= Pn (erzeugt das Signal sel1notn)
P2 >= P3 (erzeugt das Signal sel2not3)
P2 >= P4 (erzeugt das Signal sel2not4)
...
P2 >= P3 (erzeugt das Signal sel2notn)
...
Pn-1 >= Pn (erzeugt das Signal sel1n-1notn)

[0031] Daher erzeugt allgemein ausgedrückt für jedes Paar anfordernder Einheiten, die allgemein als x und y bezeichnet sind, die Stufe A1 das Signal selxnoty mit einem hohen Pegel, wenn Px >= Py zutrifft.

[0032] Zum Beispiel:

1. bei 2 anfordernden Einheiten: P1 >= P2 (1 Komparator)
2. bei 3 anfordernden Einheiten: P1 >= P2, P1 >= P3, P2 >= P3 (2 Komparatoren)
3. bei 4 anfordernden Einheiten: P1 >= P2, P1 >= P3, P1 >= P4, P2 >= P3, P2 >= P4, P3 >= P4 (6 Komparatoren)
4. bei 6 anfordernden Einheiten: P1 >= P2, P1 >= P3, P1 >= P4, P1 >= P5, P1 >= P6, P2 >= P3, P2 >= P4, P2 >= P5, P2 >= P6, P3 >= P4, P3 >= P5, P3 >= P6, P4 >= P5, P4 >= P6, P5 >= P6 (insgesamt 15 Komparatoren)
5. bei n anfordernden Einheiten: (insgesamt = 1 + 2 + 3 + 4 + ... n – 2 + n – 1, was mit y = n – 1 insgesamt bedeutet, dass es (y/2*(y + 1)) Komparatoren gibt.

[0033] Vorzugsweise werden alle Vergleiche gleichzeitig durchgeführt und hängt ihre Komplexität von der Prioritätsauswahl ab. Zum Beispiel kann bei einem HDTV-Decoder die Prioritätsauswahl von 0 bis 15 reichen.

Stufe Nr. 2: Überkreuz-Erzeugung von Anfragen

[0034] Diese Stufe verwendet die von den Initiatoren kommenden Anfragen und die von den Komparatoren der übergeordneten Stufe kommenden Ergebnisse, um so Zwischenparameter zu berechnen, die für die Erzeugung der Bewilligungen nützlich sind.

[0035] Insbesondere wird die folgende Funktion implementiert: req_arb (selxnoty, req_x, req_y, req_x_y, req_y_x), wobei req_x (bzw. req_y) die Anfrage ist, die vom Initiator x (bzw. y) kommt, während req_x_y und req_y_x die entsprechenden Ausgaben der folgenden Funktionen sind:
req_x_y = req_y AND (selxnoty OR NOT req_y), and
req_y_x = req_y AND (NOT selxnoty) OR (NOT req_x)).

[0036] In der Praxis entspricht req_arb einem Modul mit drei Eingaben (INPUTS) und zwei Ausgaben (OUTPUTS):
req_arb (INPUTS: selxnoty, reqx, reqy
OUTPUTS: reqx_y, reqy_x)

[0037] Hierdurch wird eine Arbitrierungsfunktion zwischen den Anfragenpaaren reqx, reqy durchgeführt (in dem in Fig. 3 gezeigten Beispiel: req1, req2 oder req1, req3 oder aber req2, req3), wobei das Ergebnis in einem One-Hot-Coding-Verfahren codiert wird, d.h. es gibt n Ergebnisse, die unter der Verwendung von n binären Zahlen darstellbar sind, im vorliegenden Fall 01 bzw. 10.

[0038] Das Ergebnis einer solchen Operation hängt von den folgenden Faktoren ab:

i) Priorität von reqx bezüglich reqy: reqx hat dabei eine höhere Priorität als reqy, wenn selxnoty auf einem hohen logischen Pegel ist, und umgekehrt, wenn es bei einem niedrigen logischen Pegel ist;
ii) Anwesenheit der Anfragen: Die Entscheidung wird beeinflusst durch die tatsächliche Anwesenheit von Anfragen; es könnte nämlich in der Situation, in der selxnoty auf einem hohen logischen Pegel ist, reqy bedient werden (wobei das nur ein Zwischenergebnis wäre), wenn kein reqx vorliegt.

[0039] Daraus folgt:

– Die Ausgabe reqx_y ist dann auf einem hohen logischen Pegel (reqx wird mit Sicherheit gegenüber reqy bevorzugt), wenn das Signal reqx auf einem hohen logischen Pegel ist (Anwesenheit der Anfrage) und wenn gleichzeitig reqx eine höhere Priorität als reqy hat (selxnoty auf einem hohen logischen Pegel ist) oder wenn reqy auf einem niedrigen logischen Pegel ist (Abwesenheit einer Anfrage);
– Die Ausgabe reqy_x ist dann auf einem hohen logischen Pegel (reqy wird mit Sicherheit gegenüber reqx bevorzugt), wenn das Signal reqy auf einem hohen logischen Pegel ist (Anwesenheit der Anfrage) und gleichzeitig, wenn reqy die höhere Priorität als reqx hat (selxnoty auf einem niedrigen logischen Pegel ist) oder wenn reqx auf einem niedrigen logischen Pegel ist (Abwesenheit einer Anfrage).

[0040] Insgesamt werden y/2*(y + 1) req_arb-Module benötigt.

Stufe Nr. 3: Überkreuz-Erzeugung von Bewilligungen

[0041] Nachdem die Anfragen umgeordnet wurden, besteht die abschließende Operation aus der Erzeugung der Bewilligungen (Grants).

[0042] Für jede i-te anfordernde Einheit wird die Bewilligung grant_i einfach dadurch erhalten, dass unter allen Signalen req_i_x die logische UND-Operation (AND) durchgeführt wird, mit x von 1 bis N, wobei der Fall von x = i ausgeschlossen ist.

Beispiele:

a) 3 anfordernde Einheiten

Stufe 1:

[0043] Parallele Berechnung von
P1 >= P2, P1 >= P3, P2 >= P3

Stufe 2:

[0044] Parallele Berechnung von
req_arb (P1 >= P2, req1, req2, req1_2, req2_1)
req_arb (P1 >= P3, req1, req3, req1_3, req3_1)
req_arb (P2 >= P3, req2, req3, req2_3, req3_2)

Stufe 3:

grant1 = req1_2 AND req1_3
grant2 = req2_1 AND req2_3
grant3 = req3_1 AND req3_2

b) 4 anfordernde Einheiten

Stufe 1:

[0045] Parallele Berechnung von
P1 >= P2, P1 >= P3, P1 >= P4, P2 >= P3, P2 >= P4, P3 >= P4

Stufe 2:

[0046] Parallele Berechnung von
req_arb (P1 >= P2, req1, req2, req1_2, req2_1)
req_arb (P1 >= P3, req1, req3, req1_3, req3_1)
req_arb (P1 >= P4, req1, req4, req1_4, req4_1)
req_arb (P2 >= P3, req2, req3, req2_3, req3_2)
req_arb (P2 >= P4, req2, req4, req2_4, req4_2)
req_arb (P3 >= P4, req3, req4, req3_4, req4_3)

Stufe 3:

grant1 = req1_2 AND req1_3 AND req1_4
grant2 = req2_1 AND req2_3 AND req2_4
grant3 = req3_1 AND req3_2 AND req3_4
grant4 = req4_1 AND req4_2 AND req4_3

[0047] Zusammengefasst ermöglicht die erfindungsgemäße Lösung eine schnellere Ausführung des Arbitrierungsalgorithmus mit variabler Priorität. All dies in einer Situation, in der die Ausführungszeit gegenüber der Codierung der Priorität nicht sehr empfindlich ist. Außerdem wird die Ausführungszeit der Arbitrierung von der Anzahl von Initiatoren nicht stark beeinflusst. Die Implementierung ist modular, wobei die sie auszeichnenden Elemente (siehe Darstellung von **Fig. 3**) zum Implementieren von Arbitratoren mit einer generischen Anzahl n von Wahlmöglichkeiten repliziert werden können.

[0048] Zum Beispiel hat sich bei vom vorliegenden Anmelder durchgeführten Experimenten herausgestellt, dass im Falle eines eingebetteten Systems mit 9 Initiatoren in 0,25-Mikrometer-Technik die Erzeugungszeit der Bewilligungen gegenüber den Anfragen 3 Nanosekunden betrug.

[0049] Selbstverständlich können ohne Beeinträchtigung des Prinzips der vorliegenden Erfindung die Konstruktionseinzelheiten und die Ausführungsformen gegenüber dem hier Beschriebenen und Veranschaulichten weit abweichen, ohne dass dadurch vom Umfang der vorliegenden Erfindung abgewichen wird, wie er in den beiliegenden Ansprüchen definiert ist.

Patentansprüche

1. Verfahren zur Arbitrierung zwischen einer Anzahl von n Einheiten (I1, I2, ..., In), welche Zugriff auf mindestens eine Resource (T) durch Ausdrücken jeweiliger Anfragen (req_x) und jeweiliger Prioritäten (P1, P2, ..., Pn) ersuchen, wobei der Zugriff der Einheiten zu der mindestens einen Resource (T) gemäß Bewilligungen (grant1, grant2, ...) reguliert wird, welche mittels des Arbitrierungsverfahrens identifiziert werden,
dadurch gekennzeichnet, daß es die folgenden Operationen aufweist:
– Vergleichen der jeweiligen Prioritäten (P1, ..., Pn) miteinander, wobei für jedes Paar der Einheiten, welches im allgemeinen eine Einheit x und eine Einheit y mit zugehörigen Prioritäten Px und Py aufweist, ein Auswahlsignal (selxnoty) mit einem hohen Niveau, wenn das Ergebnis der Operation Px >= Py wahr ist;
– Erzeugen zugehöriger Überkreuz-Anfrage-Signale für die Paare der Einheiten (I1, I2, ..., In)
req_arb(selxnoty, req_x, req_y, req_x_y; req_y_x)
wobei
selxnoty das obengenannte Auswahlsignal ist;
req_x das von der Einheit x kommende Anfragesignal ist;
req_y das von der Einheit y kommende Anfragesignal ist; und
req_x_y und req_y_x jeweils die Ausgabe der Funktion
req_x UND (selxnoty ODER NICHT req_y)
und der Funktion

req_y UND ((NOT selxnoty) ODER (NICHT req_x)) ist; und
– Erzeugen der Bewilligung für die i-te Einheit als ein logisches Produkt (UND) aller der vorgenannten Überkreuz-Anfrage-Signale req_i_x, mit x von 1 bis n, ausgenommen den Fall x = i.

2. Verfahren gemäß Anspruch 1, dadurch gekennzeichnet, daß die Operation des Vergleichs zwischen den jeweiligen Prioritäten für alle Paare der Einheiten parallel ausgeführt wird.

3. Verfahren gemäß Anspruch 1 oder Anspruch 2, dadurch gekennzeichnet, daß die Überkreuz-Anfrage-Signale für alle Paare der Einheiten parallel erzeugt werden.

4. System, aufweisend eine Anzahl von n Einheiten (I1, I2, ..., In), welche dazu ausgelegt sind, Zugriff auf mindestens eine Resource (T) durch Ausdrücken jeweiliger Anfragen (req_x) und jeweiliger Prioritäten (P1, P2, ..., Pn) zu ersuchen, wobei der Zugriff der Einheiten zu der mindestens einen Resource (T) gemäß Bewilligungen (grant1, grant2, ...) reguliert wird, welche von einem Arbitrierer (A) erzeugt werden, dadurch gekennzeichnet, daß der Arbitrierer (A) folgendes aufweist:
– eine erste, Vergleich-Stufe (A1) zum Vergleichen der jeweiligen Prioritäten (P1, ..., Pn) miteinander, wobei für jedes Paar der Einheiten, welches im allgemeinen eine Einheit x und eine Einheit y mit zugehörigen Prioritäten Px und Py aufweist, ein Auswahlsignal (selxnoty) mit einem hohen Niveau erzeugt wird, wenn das Ergebnis der Operation Px >= Py wahr ist;
– eine zweite, Überkreuz-Anfrage-Erzeugungs-Stufe (A2) zum Erzeugen zugehöriger Überkreuz-Anfrage-Signale für die Paare der Einheiten (I1, I2, ..., In)
req_arb(selxnoty, req_x, req_y, req_x_y, req_y_x)
wobei
selxnoty das obengenannte Auswahlsignal ist;
req_x das von der Einheit x kommende Anfragesignal ist;
req_y das von der Einheit y kommende Anfragesignal ist; und
req_x_y und req_y_x jeweils die Ausgabe der Funktion
req_x UND (selxnoty ODER NICHT req_y)
und der Funktion
req_y UND ((NOT selxnoty) ODER (NICHT req_x)) ist; und
– eine dritte, Bewilligung-Erzeugungs-Stufe (A3) zum Erzeugen der Bewilligung für die i-te Einheit als ein logisches Produkt (UND) aller der vorgenannten Überkreuz-Anfrage-Signale req_i_x, mit x von 1 bis n, ausgenommen den Fall x = i.

5. System gemäß Anspruch 4, dadurch gekennzeichnet, daß die erste Stufe (A1) dafür konfiguriert ist, die Operation des Vergleichs zwischen den jeweiligen Prioritäten für alle Paare der Einheiten parallel durchzuführen.

6. System gemäß Anspruch 4 oder Anspruch 5, dadurch gekennzeichnet, daß die zweite Stufe (A2) dafür konfiguriert ist, die Überkreuz-Anfrage-Signale für alle Paare der Einheiten parallel zu erzeugen.

7. System gemäß einem der Ansprüche 4–6, dadurch gekennzeichnet, daß die mindestens eine Resource (T) aus einem Bus besteht.

8. System gemäß einem der Ansprüche 4–7, dadurch gekennzeichnet, daß die Einheiten der Anzahl (I1, I2, ..., In) dafür konfiguriert sind, einen Datenaustausch mit der Resource selbst als Transaktion zu der mindestens einen Resource (T) auszuführen.

9. System gemäß einem der Ansprüche 4–8, dadurch gekennzeichnet, daß das System ein System vom Typ eingebettetes System (Embedded System) ist.

Es folgen 2 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

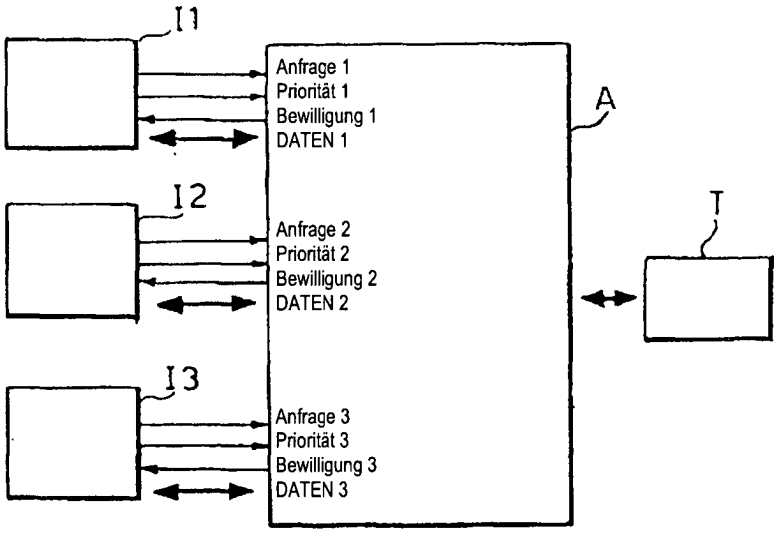


FIG. 2

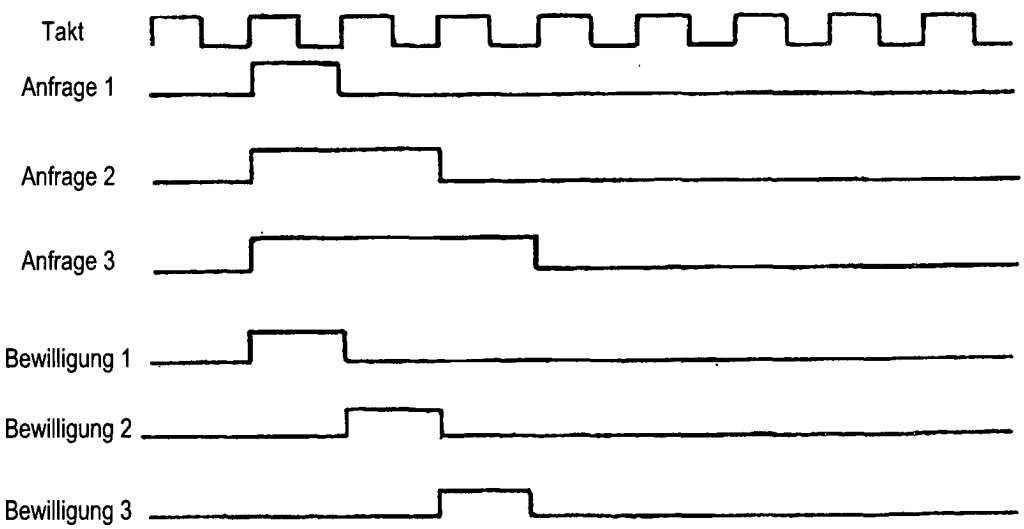
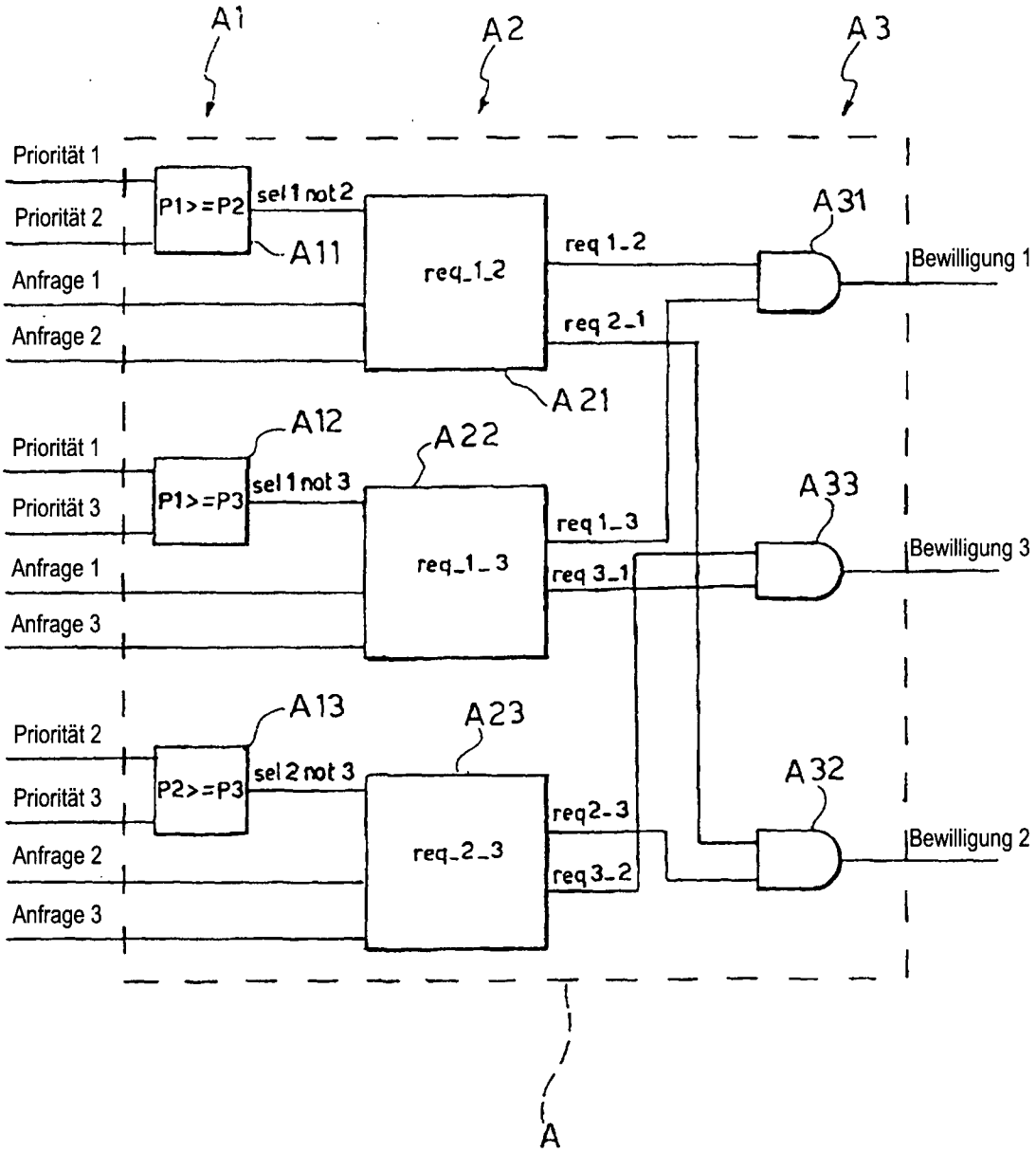


FIG. 3


A1
A2
A3
Priorität 1
Priorität 2
Anfrage 1
Anfrage 2
P1>=P2
sel 1 not 2
A11
req_1_2
req 1_2
req 2_1
A21
A31
Bewilligung 1
Priorität 1
Priorität 3
Anfrage 1
Anfrage 3
A12
P1>=P3
sel 1 not 3
A22
req_1_3
req 1_3
req 3_1
A33
Bewilligung 3
Priorität 2
Priorität 3
Anfrage 2
Anfrage 3
A13
P2>=P3
sel 2 not 3
A23
req_2_3
req2-3
req3-2
A32
Bewilligung 2
A