(19) österreichisches patentamt

(10) **AT 511 805 A4 2013-03-15**

(12) # Österreichische Patentanmeldung

(21) Anmeldenummer: A 1656/2011
(22) Anmeldetag: 09.11.2011
(43) Veröffentlicht am: 15.03.2013

(51) Int. Cl. : ***G06F 11/16*** (2006.01)
H04L 7/08 (2006.01)

(56) Entgegenhaltungen:
EP 1769356 B1 DE 20121466 U1

(73) Patentanmelder:
FTS COMPUTERTECHNIK GMBH
1040 WIEN (AT)

(72) Erfinder:
KOPETZ HERMANN DR.
BADEN (AT)

(54) **VERFAHREN ZUR OPTIMIERUNG DER WARTEZEIT IN REPLIKA-DETERMINISTISCHEN SYSTEMEN**

(57) Die vorliegende Erfindung legt ein optimales Verfahren offen um in einer Systemarchitektur zu vordefinierten Echtzeitpunkten automatisch einen identischen inneren Zustand in allen korrekten Knotenrechner eines verteilten zeitgesteuerten Computersystems zu realisieren. Durch den Automatismus der inneren Zustandssynchronisation entfällt die Notwendigkeit, die Zustandssynchronisation explicit in der Anwendungssoftware durchzuführen. Damit werden der Entwicklungsaufwand und damit die Kosten, die anfallen, um ein fehlertolerantes Computersystem zu bauen, wesentlich gesenkt.

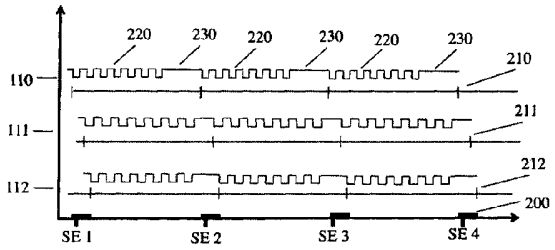

FIG. 2

AT 511 805 A4 2013-03-15

ZUSAMMENFASSUNG

Die vorliegende Erfindung legt ein optimales Verfahren offen um in einer Systemarchitektur zu vordefinierten Echtzeitpunkten automatisch einen identischen inneren Zustand in allen korrekten Knotenrechner eines verteilten zeitgesteuerten Computersystems zu realisieren. Durch den Automatismus der inneren Zustandssynchronisation entfällt die Notwendigkeit, die Zustandssynchronisation explicit in der Anwendungssoftware durchzuführen. Damit werden der Entwicklungsaufwand und damit die Kosten, die anfallen, um ein fehlertolerantes Computersystem zu bauen, wesentlich gesenkt.

Einreichfassung, 9. Nov. 2011

Verfahren zur Optimierung der Wartezeit in *replika-deterministischen* Systemen

Zitierte Literatur

Patente:

[1] US 7,804,890 Navada, et. al. *Method and System for Response Determinism by Synchronization*. (granted Sept. 28, 2010).

[2] US 7,937,618. Dorai et al. *Distributed, Fault-tolerant and highly available computersystem*. (granted May 3, 2011).

[3] US5,317,726 Horst. *Multiple-processor computer system with asynchronous execution of identical code streams*. (granted May 31, 1994).

[4] US6,233,702 Horst et al. *Self-checked, lock step processor pairs*. (granted May 15, 2001).

[5] US 7,908,520 Avizienis, *A. Self-testing and -repairing fault-tolerance infrastructure for computer systems*. (granted March 15, 2011)

Sonstige:

[6] Kopetz, H. (2011). *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Springer Verlag.

[7] Poledna, S. (1995). *Fault-Tolerant Real-Time Systems, The Problem of Replika Determinism*. Springer Verlag.

[8] *TAI—Temps Atomique International*, Wikipedia. 2011.

Technisches Umfeld

Die vorliegende Erfindung beschreibt ein zeitlich optimales Verfahren, um *Replika Determinismus* innerhalb eines verteilten Echtzeitcomputersystems, indem identische Knotenrechner (Replikas) die gleichen Programme abarbeiten, zu realisieren.

Kurze Beschreibung der Erfindung

Garantiert eine verteilte Systemarchitektur automatisch das replika-deterministische Verhalten der korrekten Knotenrechner zu ausgewählten Echtzeitpunkten, so kann durch den Vergleich von zwei unabhängig errechneten Resultaten zu diesen Echtzeitpunkten ein Fehler in einem Knotenrechner erkannt und bei Vergleich von

drei Resultaten ein Fehler maskiert werden. Entsprechend dem Lehrbuch [6, p.125] wird Determinismus wie folgt definiert: *Determinism is a property of a computation that makes it possible to predict the future result of a computation, given that the initial state and all timed inputs are known.* Replika Determinismus liegt demnach vor, wenn alle korrekten Replikas zu ausgewählten Echtzeitpunkten die gleichen Ergebnisse erreichen [7].

Während das Fortschreiten der Echtzeit TAI durch aufwendige internationale physikalische Verfahren gemessen wird [8], bestimmt der innere Taktgeber eines Computers die Rechenzeit und damit das Resultat einer Berechnung zu einem gegebenen Echtzeitpunkt. Da in verschiedenen Knotenrechner, die über autonome Taktgeber verfügen, das Verhältnis von Echtzeit zu Rechenzeit unterschiedlich ist, sind normalerweise zu einem gegebenen zukünftigen Echtzeitpunkt die Resultate in identischen Knotenrechner eines verteilten Computersystems unterschiedlich, selbst wenn die Berechnungen zum gleichen Echtzeitpunkt mit den gleichen Daten begonnen haben und von der exakt gleichen Software generiert werden.

Die vorliegende Erfindung legt ein optimales Verfahren offen um in einer Systemarchitektur zu vordefinierten Echtzeitpunkten automatisch einen identischen inneren Zustand in allen korrekten Knotenrechner eines verteilten zeitgesteuerten Computersystems zu realisieren. Durch den Automatismus der inneren Zustandssynchronisation entfällt die Notwendigkeit, die Zustandssynchronisation explicit in der Anwendungssoftware durchzuführen. Damit werden der Entwicklungsaufwand und damit die Kosten, die anfallen, um ein fehlertolerantes Computersystem zu bauen, wesentlich gesenkt.

Aufgrund von physikalisch nicht kontrollierbaren Effekten bewegt sich die Anzahl der Schwingungen (und damit der Verarbeitungstakte) eines autonomen Taktgebers mit der nominalen Frequenz f in dem (im Datenblatt des Taktgebers spezifizierten) Intervall $(f \pm \Delta df)$. Die Grundidee der Erfindung besteht nun darin, dass in einem verteilten Computersystem die physikalische Zeit durch periodische signifikante Echtzeitereignissen (SE) in eine Folge von nominal gleich langen Intervallen mit der Länge d_{nom} unterteilt wird und der langsamste Taktgeber des Ensembles von Knotenrechner die Anzahl der Verarbeitungstakte AVT innerhalb eines solchen Echtzeitintervalls bestimmt. Da *a priori* nicht entschieden werden kann, welcher Knotenrechner der langsamste ist, wird angenommen, dass sich die Taktgeber aller korrekten Knotenrechner innerhalb des Intervalls $(f \pm \Delta df)$ bewegen und der langsamste korrekte Taktgeber mindestens so schnell läuft wie $(f - \Delta df)$. Da aufgrund der Synchronisation und Digitalisierung der *lokale Wert* der Länge eines Echtzeitintervalls mit der nominalen Länge d_{nom} schwanken kann [Kop11, p. 62] muss zur Berechnung der Verarbeitungstakte AVT die minimale Intervalllänge wie folgt herangezogen werden $AVT = d_{min} (f - \Delta df)$, wobei $d_{min} = (d_{nom} - 2g)$, wobei g die Granularität der globalen physikalischen Zeit auf der relevanten Abstraktionsebene angibt. Die Granularität der globalen Zeit hängt über die *reasonableness condition* $g > P$ von der Präzision P der Uhrensynchronisation auf der relevanten Abstraktionsebene ab [6, p. 58]. Jeder Knotenrechner zählt die Verarbeitungstakte nach dem Auftreten eines signifikanten Ereignisses (SE) bis der Wert AVT erreicht wird und füllt das Intervall bis zum nächsten signifikanten Ereignis mit *wait-states*.

Es entspricht dem Stand der Technik, dass durch die Verlängerung des Echtzeitintervalls bis zu dem ein Resultat vorhanden sein muss der Determinismus in einem System, das den Wert-Determinismus unterstützt, dessen Verhalten aber

zeitlich nicht exakt vorhersehbar ist, auf einer höheren Abstraktionsebene mit einer gewissen Wahrscheinlichkeit wieder hergestellt werden kann [6,p. 130]. Der wesentliche Inhalt der vorliegenden Erfindung ist die Festlegung eines minimal notwendigen und damit optimalen Warteintervalls eines korrekten Knotenrechners. Dieses Warteintervall wird bestimmt von dem Verhältnis der tatsächlichen Frequenzen der Taktgeber des Ensembles zur Echtzeit. Da jede zentrale Echtzeituhr ausfallen kann, muss die Echtzeit über ein verteiltes fehlertolerantes Synchronisationsverfahren, das den Ausfall einer beliebigen Uhr toleriert, aufgebaut werden, so dass es im fehlertoleranten System keinen zentralen Ausfallpunkt gibt [6, p. 69].

Wenn ein großes verteiltes Echtzeitcomputer System hierarchisch aufgebaut ist, so wird die Granularität der globalen Zeit auf den unterschiedlichen Abstraktionsebenen unterschiedlich groß sein. Wenn z.B. in einem *System-on-Chip (SoC)* die IP-cores über ein zeitgesteuertes *Network-on-Chip* kommunizieren, so kann die Granularität der globalen Zeit im NoC zehn Nanosekunden betragen, wogegen in Knotenrechnern, die über ein lokales Echtzeitkommunikationssystem kommunizieren, die Granularität der globalen Zeit eine Mikrosekunde betragen kann.

Das vorliegende Verfahren zur Realisierung von Determinismus in einem verteilten Echtzeitcomputersystem unterscheidet sich von anderen bekannten Verfahren zur Erreichen von Determinismus und Fehlertoleranz durch die Bezugnahme auf die physikalischen Zeit, der Echtzeit, die entweder über einen fehlertoleranten Synchronisationsalgorithmus oder über externe redundante self-checking Synchronisationsnachrichten aufgebaut wird. Wie aus der Literatur bekannt ist, sind zum Aufbau einer fehlertoleranten physikalischen Zeit mindestens vier unabhängige Uhren erforderlich wenn ein beliebiger Fehler in einer Uhr toleriert werden soll [6, p.72]. Wenn eine nicht fehlertolerante externe Zeitreferenz angenommen wird so kann das Auftreten von Fehlern in der physikalischen Referenzzeit nicht ausgeschlossen werden. Ein solcher Zeitfehler ist in [1] kein Problem, da die Zielsetzung von [1] der Aufbau einer Testumgebung und nicht der Aufbau eines fehlertoleranten Systems ist. Das Problem von solchen Zeitfehlern wird in den Architekturen zur Erreichung von Fehlertoleranz [2] und [3] nicht angesprochen. In [4] wird das Problem umgangen indem *self-checking pairs* eingeführt werden. In [5] wird das Problem umgangen indem unabhängige *Monitor Komponenten* die Verarbeitungskomponenten überwachen.

Zusammenfassung

Die vorliegende Erfindung legt ein optimales Verfahren offen um in einer Systemarchitektur zu vordefinierten Echtzeitpunkten automatisch einen identischen inneren Zustand in allen korrekten Knotenrechner eines verteilten zeitgesteuerten Computersystems zu realisieren. Durch den Automatismus der inneren Zustandssynchronisation entfällt die Notwendigkeit, die Zustandssynchronisation explicit in der Anwendungssoftware durchzuführen. Damit werden der Entwicklungsaufwand und damit die Kosten, die anfallen, um ein fehlertolerantes Computersystem zu bauen, wesentlich gesenkt.

Einreichfassung, 9. Nov. 2011

Kurze Beschreibung der Zeichnungen

Die vorliegende Erfindung wird an Hand der Zeichnungen erklärt

Fig. 1 zeigt den Aufbau eines verteilten fehlertoleranten Echtzeitcomputersystems.

Fig. 2 zeigt den zeitlichen Ablauf der Rechenzeit (Takte) bezogen auf das Fortschreiten der Echtzeit auf den unterschiedlichen Knotenrechnern.

Beschreibung einer Realisierung

Das folgende Beispiel zeigt den konkreten Aufbau eines verteilten Echtzeitsystems mit vier Knotenrechnern und den Ablauf der signifikanten Ereignisse auf drei der vier Knotenrechner an Hand der beigefügten Abbildungen.

Fig. 1 zeigt ein Strukturdiagramm des verteilten Echtzeitsystems. Die vier Knotenrechner **110**, **111**, **112** und **113** sind über jeweils einen Kommunikationskanals mit den beiden Sternkopplern **120** und **121** verbunden. In Fig. 1 ist verbindet der Kommunikationskanal **100** den Knotenrechner **110** mit dem Sternkoppler **120** und der Kommunikationskanal **101** den Knotenrechner **110** mit dem Sternkoppler **121** (analog bei den anderen Knotenrechnern). Zwei Kommunikationskanäle pro Knotenrechner und zwei unabhängige Sternkoppler sind in einem fehlertoleranten System erforderlich, um den Ausfall eines Kommunikationskanals oder eines Sternkoppler tolerieren zu können. In einem minimalen System müssen mindestens vier Knotenrechner vorhanden sein, um eine fehlertolerante globale Zeit aufbauen zu können.

Fig. 2 zeigt den zeitlichen Ablauf der signifikanten Ereignisse auf den Knotenrechnern **110**, **111** und **112**, die eine Triade eines fehlertoleranten TMR (Triple Modular Redundant) Systems bilden [6, p.155-157). Auf der Abszisse von Fig. 2 sind die globalen Echtzeitticks der signifikanten Synchronisationsereignisse SE1, SE2, SE3 und SE4 durch dicke senkrechte Striche dargestellt. In den Knotenrechnern finden die entsprechenden lokalen Synchronisationsereignisse SE **210**, **211** und **212** innerhalb der Präzision **200** der globalen Uhrensynchronisation statt. In jedem Knotenrechnern befindet sich ein autonome Taktgeber, der sich im Intervall ($f \pm \Delta df$) bewegt, wobei f die nominale Frequenz eines Taktgebers ist und Δdf die maximale Abweichung eines noch korrekten Taktgebers von dieser nominalen Frequenz angibt. Die Anzahl der Verarbeitungstakte (AVT) zwischen zwei aufeinanderfolgenden SEs mit dem nominalen Abstand d_{nom} bestimmt sich durch $AVT = d_{min}(f - \Delta df)$, wo $d_{min} = (d_{nom} - 2g)$ die minimale Dauer zwischen zwei aufeinanderfolgenden SEs angibt. Der nominale Abstandswert d_{nom} ist um 2g, der Granularität der globalen physikalischen Zeit, zu reduzieren, um den Synchronisationsfehler und den Digitalisierungsfehler zu kompensieren. Im konkreten Beispiel der Fig. 2 wird angenommen dass innerhalb des Abstands d_{nom} auf allen korrekten Rechnern sieben Takte **220** ablaufen können. Nach Abarbeitung dieser sieben Takte gehen die Rechner erfindungsgemäß in den Wartezustand bis das nächste lokale signifikante Synchronisationsereignis auftritt. Beim Knotenrechner mit dem schnellsten Taktgeber, d.i. in Fig. 2 der Knotenrechner **110** ist dieses Warteintervall **230** am längsten, wogegen im Knotenrechner **111**, dem Knotenrechner mit dem langsamsten Taktgeber, dieses Warteintervall am kürzesten ist.

Die lokalen Synchronisationsereignisse SE **210, 211** und **212** können entweder durch eine fehlertolerante interne Uhrensynchronisation oder durch eine fehlertolerante externe Uhrensynchronisation hergestellt werden. In beiden Fällen werden diese Ereignisse in korrekten Knotenrechner innerhalb der Präzision der globalen Zeit auftreten.

Wenn nun die Knotenrechnern **110, 111** und **112**, als *IP-Cores* in einem *System–on-Chip (SoC)* realisiert werden und diese Knotenrechner über ein zeitgesteuertes *Network-on-Chip (NoC)* kommunizieren, so ist zu berücksichtigen, dass das NoC von einem einzigen Taktgeber getrieben wird, wobei sich die Sendezeitpunkte des NoC vom Makrotick innerhalb des SoC ableiten. Um den zeitlichen Gleichlauf mit anderen SoCs eines Ensembles zu erreichen sind diese SoC internen Makroticks mit den externen Synchronisationssignalen zu synchronisieren. Dies geschieht indem die Anzahl Z_{macro} der SoC internen Makroticks zwischen den äquidistanten externen Synchronisationssignalen statisch festgelegt wird und der Abstand d_{ext} zwischen zwei externen Synchronisationssignale durch Z_{macro} geteilt wird, um das Intervall d_{int} zwischen zwei internen Macroticks zu bestimmen. Diese internen Macroticks müssen dann durch eine entsprechende Logik innerhalb des SoCs von einem Taktgeber des SoCs abgeleitet werden. Wenn nicht alle lokalen IP Cores vom gleichen Taktgeber getrieben werden, so ist innerhalb eines SoCs zwischen einem IP Core und dem NoC die gleiche beschriebene Vorgangsweise wie zwischen Knotenrechnern einzuhalten.

Das beschriebene innovative Verfahren zur Realisierung von Determinismus lässt sich auf der Ebene der Hardware oder des Betriebssystems eines Knotenrechners autonom implementieren. Dadurch wird die Entwicklung und Validierung der Anwendungssoftware wesentlich vereinfacht, was den wirtschaftlichen Wert dieser Erfindung unterstreicht.

Patentansprüche

1. Verfahren zur Realisierung von replika-deterministischen Verhalten in einem verteilten zeitgesteuerten Computersystem, bestehend aus einer Anzahl von autonomen Knotenrechnern die über eine gemeinsame fehlertolerante Echtzeitbasis mit der Präzision P verfügen und die über ein zeitgesteuertes Echtzeitkommunikationssystem verbunden sind, wo jeder Knotenrechner mit einem eigenen Taktgeber der nominalen Frequenz f ausgestattet ist, und wo sich die tatsächlichen Frequenzen der Taktgeber der Knotenrechner im Intervall ($f \pm \Delta df$) bewegen, **dadurch gekennzeichnet dass** ein definierter innerer Zustand eines Knotenrechner zum Zeitpunkt des Auftretens von periodischen signifikanten Ereignissen (SE) mit dem nominalen Echtzeitabstand d_{nom} dadurch gewährleistet wird, dass die Anzahl der Verarbeitungstakte (AVT) zwischen zwei aufeinanderfolgenden SEs bestimmt ist durch $AVT = d_{min}(f - \Delta df)$, wo $d_{min} = (d_{nom} - 2g)$, wobei g die Granularität der globalen physikalischen Zeit angibt, und wo jeder Knotenrechner nach dem Auftreten eines SE einen Zähler initialisiert und die Verarbeitungstakte in diesem Zähler zählt und wenn dieser Zähler den Wert AVT erreicht eine Folge von *wait-states* bis zum Auftreten des nächsten signifikanten Ereignisses SE einfügt.
2. Verfahren nach Anspruch 1 **dadurch gekennzeichnet, dass** die periodisch eintreffenden Nachrichten zur externen Uhrensynchronisation signifikante Ereignisse darstellen.
3. Verfahren nach Anspruch 1 **dadurch gekennzeichnet, dass** die signifikanten Ereignisse durch eine fehlertolerante interne Uhrensynchronisation gebildet werden.
4. Verfahren nach einem oder mehreren der Ansprüche 1 bis 3 **dadurch gekennzeichnet dass** jeder interne Makrotick der globalen Zeit innerhalb eines Knotenrechners ein signifikantes Ereignis SE darstellt.
5. Verfahren nach einem oder mehreren der Ansprüche 1 bis 3 **dadurch gekennzeichnet dass** ein Knotenrechner als ein IP-Core in einem System-on-Chip (SoC) realisiert ist.
6. Verfahren nach einem oder mehreren der Ansprüche 1 bis 3 **dadurch gekennzeichnet dass** das innerhalb eines SoC ein zeitgesteuertes Network on Chip realisiert ist, welches das Signal zum Senden einer Nachricht vom Auftreten eines spezifizierten internen Macroticks ableitet.
7. Apparat **dadurch gekennzeichnet dass** der Apparat einen oder mehrere der in den Ansprüchen 1 bis 7 offen gelegten Verfahrensschritte realisiert.

Einreichfassung, 9. Nov. 2011

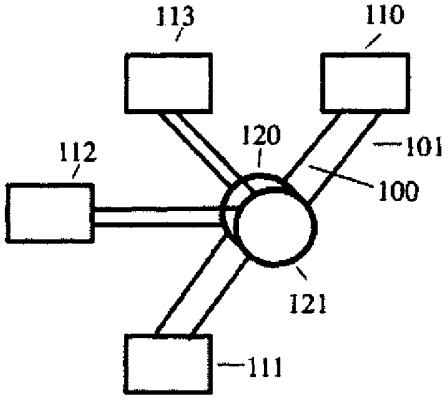
113
110
120
101
112
100
121
111

FIG. 1

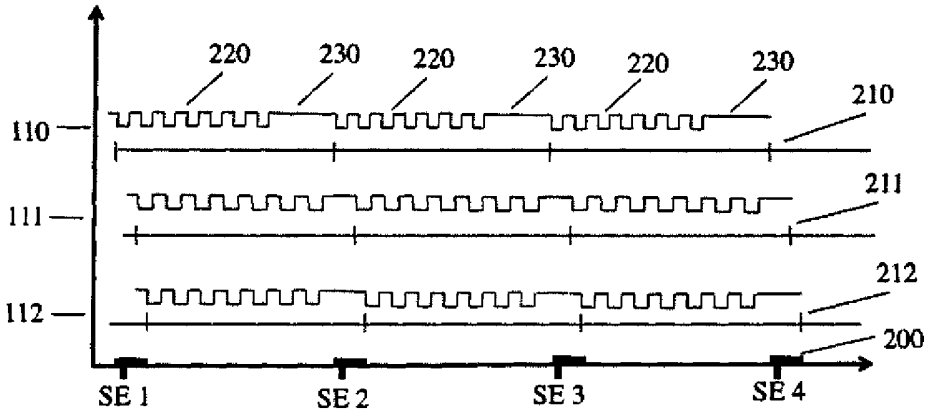
220
230
220
230
220
230
210
110
211
111
212
112
200
SE 1
SE 2
SE 3
SE 4

FIG. 2

Patentansprüche

1. Verfahren zur Realisierung von replika-deterministischen Verhalten in einem verteilten zeitgesteuerten Computersystem, bestehend aus einer Anzahl von autonomen Knotenrechnern die über eine gemeinsame fehlertolerante Echtzeitbasis mit der Präzision P verfügen und die über ein zeitgesteuertes Echtzeitkommunikationssystem verbunden sind, wo jeder Knotenrechner mit einem eigenen Taktgeber der nominalen Frequenz f ausgestattet ist, und wo sich die tatsächlichen Frequenzen der Taktgeber der Knotenrechner im Intervall (f ± Δdf) bewegen, **dadurch gekennzeichnet dass** ein definierter innerer Zustand eines Knotenrechner zum Zeitpunkt des Auftretens von periodischen signifikanten Ereignissen SE mit dem nominalen Echtzeitabstand d_{nom} dadurch gewährleistet wird, dass die Anzahl der Verarbeitungstakte (AVT) zwischen zwei aufeinanderfolgenden signifikanten Ereignissen SEs bestimmt ist durch $AVT = d_{min}(f - \Delta df)$, wo $d_{min} = (d_{nom} - 2g)$, wobei g die Granularität der globalen physikalischen Zeit angibt, und wo jeder Knotenrechner nach dem Auftreten eines signifikanten Ereignisses SE einen Zähler initialisiert und die Verarbeitungstakte in diesem Zähler zählt und wenn dieser Zähler den Wert AVT erreicht eine Folge von *Warteintervallen* bis zum Auftreten des nächsten signifikanten Ereignisses SE einfügt.
2. Verfahren nach Anspruch 1 **dadurch gekennzeichnet, dass** die periodisch eintreffenden Nachrichten zur externen Uhrensynchronisation signifikante Ereignisse darstellen.
3. Verfahren nach Anspruch 1 **dadurch gekennzeichnet, dass** die signifikanten Ereignisse durch eine fehlertolerante interne Uhrensynchronisation gebildet werden.
4. Verfahren nach einem oder mehreren der Ansprüche 1 bis 3 **dadurch gekennzeichnet dass** jeder interne Makrotick der globalen Zeit innerhalb eines Knotenrechners ein signifikantes Ereignis SE darstellt.
5. Verfahren nach einem oder mehreren der Ansprüche 1 bis 3 **dadurch gekennzeichnet dass** ein Knotenrechner als ein IP-Core in einem System-on-Chip SoC realisiert ist.
6. Verfahren nach Anspruch 4 und 5, **dadurch gekennzeichnet dass** das innerhalb eines System-on-Chip SoC ein zeitgesteuertes Network on Chip realisiert ist, welches das Signal zum Senden einer Nachricht vom Auftreten eines spezifizierten internen Macroticks ableitet.
7. Apparat **dadurch gekennzeichnet dass** der Apparat ein Verfahren nach einem oder mehreren der Ansprüche 1 bis 6 durchführt.