PCT WELTORGANISATION FÜR GEISTIGES EIGENTUM
Internationales Büro

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

(51) Internationale Patentklassifikation [7] :
G06F 13/40

A2

(11) Internationale Veröffentlichungsnummer: **WO 00/08566**

(43) Internationales Veröffentlichungsdatum: 17. Februar 2000 (17.02.00)

(21) Internationales Aktenzeichen: PCT/EP99/05679

(22) Internationales Anmeldedatum: 5. August 1999 (05.08.99)

(30) Prioritätsdaten:
98114750.7 5. August 1998 (05.08.98) EP

(71) Anmelder *(für alle Bestimmungsstaaten ausser US)*: SIEMENS AKTIENGESELLSCHAFT [DE/DE]; Wittelsbacherplatz 2, D–80333 München (DE).

(72) Erfinder; und
(75) Erfinder/Anmelder *(nur für US)*: KLOSA, Klaus [DE/DE]; Reisingerstrasse 12, D–80337 München (DE). HOFMANN, Harald [DE/DE]; Keplerstrasse 14, D–90766 Fürth (DE).

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

(81) Bestimmungsstaaten: BR, CN, IN, JP, KR, MX, RU, UA, US, europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Veröffentlicht
Ohne internationalen Recherchenbericht und erneut zu veröffentlichen nach Erhalt des Berichts.

(54) Title: INTERFACE CIRCUIT AND METHOD FOR TRANSFERRING DATA BETWEEN A SERIAL INTERFACE AND A PROCESSOR

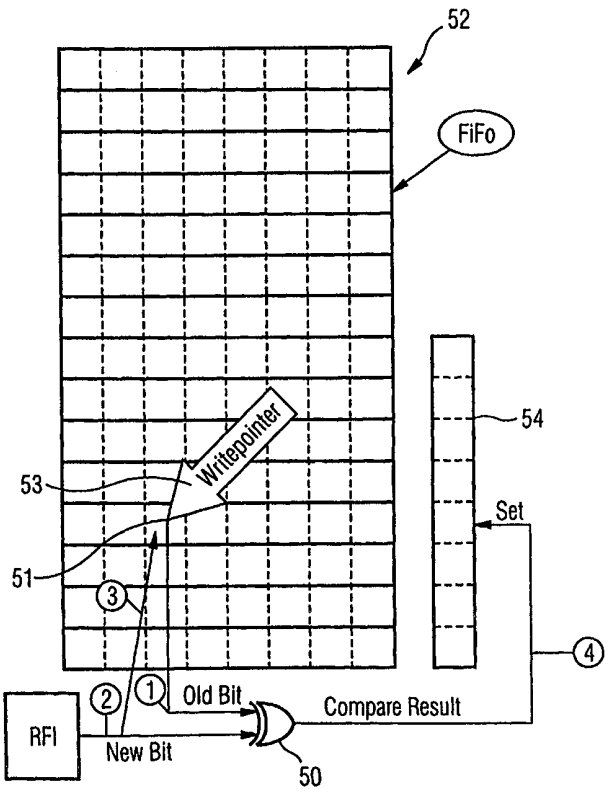
(54) Bezeichnung: INTERFACE–SCHALTUNG UND VERFAHREN ZUR ÜBERTRAGUNG VON DATEN ZWISCHEN EINER SERIELLEN SCHNITTSTELLE UND EINEM PROZESSOR

(57) Abstract

The invention relates to an interface circuit for transferring data from and to a processor via a serial interface, wherein a FIFO memory is disposed between the serial interface and the processor. The invention also relates to an appropriate method for transferring data, wherein said data is received serially bitwise, stored in the memory and read therefrom bytewise by the processor or written bytewise in the memory by the processor and then transmitted bitwise therefrom.

(57) Zusammenfassung

Interface–Schaltung zur Übertragung von Daten über eine serielle Schnittstelle von und zu einem Prozessor, wobei ein First–in–First–out–Speicher zwischen serieller Schnittstelle und Prozessor angeordnet ist, sowie dafür geeignetes Verfahren zur Übertragung von Daten, wobei diese seriell bitweise empfangen und in den Speicher eingelesen werden und vom Prozessor dort byteweise ausgelesen, bzw. vom Prozessor byteweise in den Speicher geschrieben und von dort bitweise gesendet werden können.



LEDIGLICH ZUR INFORMATION

Codes zur Identifizierung von PCT-Vertragsstaaten auf den Kopfbögen der Schriften, die internationale Anmeldungen gemäss dem PCT veröffentlichen.

AL	Albanien	**ES**	Spanien	**LS**	Lesotho	**SI**	Slowenien
AM	Armenien	**FI**	Finnland	**LT**	Litauen	**SK**	Slowakei
AT	Österreich	**FR**	Frankreich	**LU**	Luxemburg	**SN**	Senegal
AU	Australien	**GA**	Gabun	**LV**	Lettland	**SZ**	Swasiland
AZ	Aserbaidschan	**GB**	Vereinigtes Königreich	**MC**	Monaco	**TD**	Tschad
BA	Bosnien-Herzegowina	**GE**	Georgien	**MD**	Republik Moldau	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagaskar	**TJ**	Tadschikistan
BE	Belgien	**GN**	Guinea	**MK**	Die ehemalige jugoslawische Republik Mazedonien	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Griechenland			**TR**	Türkei
BG	Bulgarien	**HU**	Ungarn	**ML**	Mali	**TT**	Trinidad und Tobago
BJ	Benin	**IE**	Irland	**MN**	Mongolei	**UA**	Ukraine
BR	Brasilien	**IL**	Israel	**MR**	Mauretanien	**UG**	Uganda
BY	Belarus	**IS**	Island	**MW**	Malawi	**US**	Vereinigte Staaten von Amerika
CA	Kanada	**IT**	Italien	**MX**	Mexiko		
CF	Zentralafrikanische Republik	**JP**	Japan	**NE**	Niger	**UZ**	Usbekistan
CG	Kongo	**KE**	Kenia	**NL**	Niederlande	**VN**	Vietnam
CH	Schweiz	**KG**	Kirgisistan	**NO**	Norwegen	**YU**	Jugoslawien
CI	Côte d'Ivoire	**KP**	Demokratische Volksrepublik Korea	**NZ**	Neuseeland	**ZW**	Zimbabwe
CM	Kamerun			**PL**	Polen		
CN	China	**KR**	Republik Korea	**PT**	Portugal		
CU	Kuba	**KZ**	Kasachstan	**RO**	Rumänien		
CZ	Tschechische Republik	**LC**	St. Lucia	**RU**	Russische Föderation		
DE	Deutschland	**LI**	Liechtenstein	**SD**	Sudan		
DK	Dänemark	**LK**	Sri Lanka	**SE**	Schweden		
EE	Estland	**LR**	Liberia	**SG**	Singapur		

Beschreibung

Interface-Schaltung und Verfahren zur Übertragung von Daten zwischen einer seriellen Schnittstelle und einem Prozessor.

Die vorliegende Erfindung betrifft eine Interface-Schaltung zur Übertragung von Daten über eine serielle Schnittstelle von und zu einem Prozessor und ein Verfahren zur Übertragung von Daten zwischen einer seriellen Schnittstelle und einem Prozessor.

Die vorliegende Erfindung soll zur seriellen, insbesondere drahtlosen Datenübertragung zu einem beispielsweise auf einer Chipkarte angeordneten Prozessor dienen. Besonders geeignet ist die vorliegende Erfindung für die drahtlose Datenübertragung zwischen einem Kartenleser und einer kontaktlosen Chipkarte mit Prozessor.

In diesen Anwendungsfällen soll der Prozessor entlastet werden, damit ein langsamer getakteter Prozessor verwendet werden kann. Dadurch soll der Stromverbrauch gesenkt werden (bei den heute üblichen Prozessoren ist der Stromverbrauch proportional zur Taktfrequenz). Außerdem soll der Prozessor möglichst oft und möglichst lange in den sogenannten Schlaf- oder Stand by-Modus geschaltet werden, in dem der Prozessor nicht arbeitet und dadurch wesentlich weniger Strom verbraucht.

Gemäß dem gegenwärtigen Stand der Technik erfolgt die Übertragung von Daten zwischen einer seriellen Schnittstelle und einem Prozessor mittels einer parallel-seriell bzw. seriell-parallel-Wandlung über einen sogenannten UART (Universal Asynchronus Receiver Transmitter) -universeller asynchroner Empfänger und Sender. Dieser wird üblicherweise durch ein Schieberegister auf der Seite realisiert, an der beispielsweise der Transceiver einer drahtlosen Übertragung angeschlossen ist. Dieses Schieberegister kann bitweise beschrie-

ben (beim Empfangen) bzw. bitweise gelesen (beim Senden) werden. Der Prozessor muß auf der anderen Seite des UART die Daten parallel und zeitsynchron zu dem verwendeten Datenübertragungsprotokoll von dem UART abholen, bzw. diesem zur Verfügung stellen (sh. Figuren 1 und 2). Dies stellt recht hohe Anforderungen an die Echtzeitfähigkeit des Prozessors. Solche hohen Anforderungen an die Echtzeitfähigkeit des Prozessors stehen im Widerspruch zu der Forderung, daß der Prozessor möglichst wenig Strom verbrauchen soll und möglichst billig sein soll. Gerade bei kontaktlosen Chipkarten sollten Prozessoren verwendet werden können, die möglichst einfach und möglichst langsam getaktet sind, und deshalb sehr wenig Strom verbrauchen und sehr billig sind. Solche Prozessoren können aber die o.g. Echtzeitanforderungen nicht erfüllen.

Man verwendet daher gegenwärtig im Stand der Technik bei kontaktlosen Chipkarten ein Verfahren, bei dem die serielle Information bitweise vom Prozessor vom Anschluß der seriellen Schnittstelle geholt bzw. an den Anschluß der seriellen Schnittstelle geschrieben wird. Diese Aufgabe bindet jedoch sehr viel Rechenzeit des Prozessors, so daß wiederum die Notwendigkeit entsteht, einen schnelleren Prozessor einzusetzen.

Es ist daher Aufgabe der vorliegenden Erfindung, eine Interface-Schaltung zur Übertragung von Daten über eine serielle Schnittstelle von und zu einem Prozessor anzugeben, bei der die Datenübertragung ablaufen kann, ohne daß der Prozessor damit belastet wird. Weiter ist es Aufgabe der Erfindung, ein dafür geeignetes Verfahren zur Übertragung von Daten zwischen einer seriellen Schnittstelle und einem Prozessor anzugeben.

Erfindungsgemäß wird diese Aufgabe durch eine Interface-Schaltung gelöst, bei der ein Speicher für mehrere Bus- oder Prozessor-Wortlängen (z.B. Byte) zwischen serieller Schnittstelle und Prozessor angeordnet ist.

Die Aufgabe wird ebenfalls durch ein Verfahren gelöst, bei dem die Daten seriell bitweise empfangen und in einen Speicher eingelesen und vom Prozessor dort byteweise wieder ausgelesen werden, bzw. vom Prozessor byteweise in den Speicher geschrieben und von dort bitweise gesendet werden.

Es ist dabei besonders bevorzugt, wenn der Speicher Wort- oder bitweise beschrieben und ausgelesen werden kann. Dazu ist es besonders bevorzugt, daß der Speicher einen Schreib und einen Lesepointer besitzt, die jedes Bit oder jedes Wort (z. B. Byte) einzeln adressieren können.

Zur weiteren Stromersparnis ist es vorteilhaft, wenn der Prozessor über einen Stromsparmodus (Sleep- oder Stand by-Mode) mit minimalem Stromverbrauch verfügt.

Es ist weiter vorteilhaft, in dem Speicher einen Vergleicher einzubauen. Dadurch kann der Prozessor weiter entlastet werden.

Vorzugsweise kann der Vergleicher dabei eine einfache Logik umfassen, die automatisch jedes empfangene Bit mit dem Inhalt der Speicherzelle im Speicher vergleicht, auf die das empfangene Bit geschrieben werden soll.

Zur weiteren Vereinfachung der Schaltung kann der Speicher in das CPU-Modul oder in das Empfänger-Modul integriert werden. Eine weitere Vereinfachung der Schaltung ist auch dadurch möglich, daß der Speicher durch RAM-Zellen aus dem normalen RAM im Adreßbereich des Prozessors realisiert wird.

Zur weiteren Entlastung des Prozessors kann zusätzlich zu dem Speicher ein Checksummenmodul vorgesehen werden. Zusätzlich zum Checksummenmodul kann auch noch ein Vergleicher vorgesehen sein, der die Checksumme der empfangenen Daten mit einer erwarteten, vorausberechneten Checksumme vergleicht. Auf diese Weise kann der Prozessor weiter entlastet werden.

Vorzugsweise kann der Speicher in Form einer Ringstruktur realisiert werden, und/oder mit einer Überlaufdetektionsvorrichtung versehen sein, die bei einem drohenden Überlauf des Speichers den Prozessor aktiviert (aufweckt). Dadurch können Datenverluste durch Überschreiben im Speicher vermieden werden.

Bei dem erfindungsgemäßen Verfahren kann zur Entlastung des Prozessors vorzugsweise vorgesehen sein, daß der Schreib- und Lesepointer des Speichers automatisch ohne Tätigwerden des Prozessors gesetzt werden kann.

Eine wesentlich größere Flexibilität der Programmierung bietet ein Verfahren, bei dem der Prozessor die Schreib- und Lesepointer des Speichers frei setzen kann.

Noch größere programmtechnische Flexibilität bietet das Verfahren, bei dem der Prozessor die einzelnen Speicherzellen des Speichers als Bestandteile des eigenen Adreßbereichs behandelt und diese somit wahlfrei lesen und beschreiben kann.

Eine weitere Entlastung des Prozessors kann dadurch erfolgen, daß ein automatischer Datenvergleich vorgesehen wird, wobei die zu erwartenden Daten an der entsprechenden Stelle im Speicher abgespeichert sein müssen, und mittels einer einfachen Logik jedes empfangene Bit mit dem Inhalt der Speicherzelle im Speicher verglichen wird, auf die es geschrieben wird.

Dabei kann vorzugsweise der Schreibpointer des Speichers sowohl die Adressierung des erwarteten Bits, mit dem verglichen werden soll, als auch die Adressierung des empfangenen Bits, das jetzt in den Speicher geschrieben werden soll, liefern. Auf diese Weise ergibt sich eine Vereinfachung der Programmierung und des Schaltungsaufbaus.

Hierbei kann vorzugsweise nach dem Vergleich aller Bits eines Bytes ein diesem Byte zugeordnetes Bit auf 0 gesetzt werden, wenn alle Bits gleich waren, während es sonst auf 1 gesetzt wird. Dabei ist es besonders bevorzugt, wenn auf die zugeordneten Bits von dem Prozessor byteweise zugegriffen werden kann. Durch Ausmaskieren von Vergleichsinformationen kann schnell und einfach eine neu empfangene Bitfolge bewertet werden.

Vorzugsweise kann auch dabei der Prozessor während der Datenübertragung in einen Stromsparmodus (Sleep- oder Stand by-Mode) geschaltet werden. Dabei ist es besonders bevorzugt, daß der Prozessor bei einem drohenden Überlauf des Speichers aktiviert (aufgeweckt) wird.

Anstelle des relativ aufwendigen Vergleichs der einzelnen Bits kann auch ein automatischer Datenvergleich vorgesehen werden, wobei automatisch eine Prüfsumme der empfangenen Daten mit einer im voraus berechneten Prüfsumme der zu erwartenden Daten verglichen wird.

Auf diese Weise kann der Prozessor weiter entlastet werden, indem Vergleiche, z.B. der Seriennummer der kontaktlosen Chipkarte, automatisiert und ohne Unterstützung durch den Prozessor während des Datenempfangs stattfinden können.

Mit der vorliegenden Erfindung können auch Protokolle verarbeitet werden, die die Übertragung von Bruchteilen eines Bytes zulassen (z.B. nur 3 Bit).

Sowohl die Anwendung von Mehrwort-Speichern im seriellen Interface von Chipkarten, als auch die Verwendung solcher automatischer Vergleichsschaltungen und -verfahren in Kombination mit einer CPU sind bisher nicht bekannt gewesen.

Im folgenden wird die Erfindung anhand der in der beiliegenden Zeichnung dargestellten Ausführungsformen näher erläutert. Es zeigen:

Fig. 1 den Stand der Technik mit serieller Übertragung zwischen Empfängermodul und Prozessormodul;

Fig. 2 den Stand der Technik mit Anschluß des Empfängermoduls an den CPU-Bus.

Fig. 3 eine erfindungsgemäße Lösung mit serieller Übertragung der Daten zwischen Empfängermodul und dem Prozessormodul;

Fig. 4 eine erfindungsgemäße Lösung mit Anschluß des Empfängermoduls an den Prozessorbus;

Fig. 5 einen erfindungsgemäßen FiFo-Speicher mit Vergleichsschaltung zur Überprüfung der Richtigkeit erwarteter Daten;

Fig. 6 eine Prinzipdarstellung des First-in-First-out-Speichers;

Fig. 7 einen erfindungsgemäßen Datenvergleich über Bildung einer Prüfsumme; und

Fig. 8 die Bildung der Prüfsumme parallel zum Laden der Daten in den First-in-First-out-Speicher.

Die Figuren 1 und 2 zeigen nochmals die Probleme des Standes der Technik auf. Entweder kann der Empfänger 10 über eine serielle Verbindung 12 mit dem Prozessormodul 14 verbunden sein, auf dem dann eine UART (Universal Asynchronuous Receiver Transmitter = universeller asynchroner Empfänger und Sender)-Schaltung vorgesehen ist. Eine andere Lösung im Stand der Technik sieht ein Empfängermodul 20 vor, welches selbst

die UART-Schaltung trägt, und dann über den Datenbus 22 des Prozessors 24 an diesen angeschlossen ist.

Die Figuren 3 und 4 zeigen demgegenüber die erfindungsgemäße Lösung. In Fig. 3 findet wieder ein Empfängermodul 30 mit serieller Schnittstelle Verwendung. Über die serielle Verbindung 32 steht das Empfängermodul 30 mit dem Prozessormodul 34 in Verbindung. Anstelle der UART-Schaltung ist jedoch auf dem Prozessormodul 34 ein First-in-First-out-Speicher vorgesehen. Statt der UART-Schaltung wird also eine First-in-First-out-Struktur mit beispielsweise 32 Byte verwendet. Dabei werden im Fall des Empfangens von Daten die Daten bitweise in den First-in-First-out-Speicher geschrieben. Sobald ein Byte gefüllt ist, wird automatisch das nächste Byte im First-in-First-out-Speicher bitweise gefüllt, bis der Datenblock zu Ende ist. Das Senden läuft analog zum Empfangen ab, d.h., die Daten werden aus dem First-in-First-out-Speicher bitweise, Byte nach Byte ausgelesen. Sowohl der Empfangs- als auch der Sendevorgang können unabhängig von dem Prozessor (CPU) ablaufen. Vorzugsweise kann sich der Prozessor dabei in einem Sleep- oder Stand by-Mode (einem Stromsparmodus mit vernachlässigbarem Stromverbrauch) befinden. Der Prozessor kann den Inhalt des First-in-First-out-Speichers byteweise sequentiell auslesen bzw. byteweise sequentiell Daten dort einschreiben. Mit Hilfe eines Schreib- und eines Lesepointers ist es möglich, jedes Bit in dem First-in-First-out-Speicher einzeln zu adressieren. Dadurch können auch unvollständige Bytes, beispielsweise nur 3 Bit, gesendet oder empfangen werden. Gleichzeitig sind die Anforderungen an die Echtzeitfähigkeit des Prozessors erheblich geringer als bei der Lösung gemäß dem Stand der Technik mit UART-Schaltung.

Alternativ zu einem First-in-First-out-Speicher kann auch ein Last-in-First-out-Speicher (LiFo) verwendet werden. Hierbei wird lediglich die Reihenfolge der empfangenen und gesendeten Daten umgekehrt. Dies kann bei der Programmierung berücksichtigt oder sogar erwünscht sein. Die folgenden Ausführungen

lassen sich selbstverständlich analog für einen Last-in-First-out-Speicher anwenden.

Der First-in-First-out-Speicher ist also bei der vorliegenden Ausführungsform gem. Fig. 3 in das CPU-Modul integriert. Der First-in-First-out-Speicher kann dabei sogar mit RAM-Zellen aus dem normalen RAM-Arbeitsspeicher im Adreßbereich des Prozessors (beispielsweise im internen RAM) realisiert werden. Dadurch sind keine zusätzlichen Speicherschaltungen erforderlich.

Die Figur 4 zeigt eine weitere Ausführungsform der Erfindung. Dabei ist der First-in-First-out-Speicher in das Empfängermodul 40 integriert. Dieses ist sodann über den Prozessordatenbus 42 mit dem Prozessormodul 44 verbunden.

Sowohl bei der Ausführungsform der Erfindung nach Fig. 3 als auch bei der Ausführugsform nach Fig. 4 kann vorgesehen sein, daß der First-in-First-out-Speicher nur sequentiell von dem Prozessor (CPU) beschrieben und gelesen werden kann. Ein wahlfreier Zugriff des Prozessors auf den First-in-First-out-Speicher ist damit ausgeschlossen. Dies trägt erheblich zur Datensicherheit bei, wenn vermieden werden soll, daß eine Anwendung manipulierenderweise die Daten einer anderen Anwendung zu ändern versucht.

Eine wesentlich flexiblere Programmierung ist jedoch möglich, wenn der Prozessor den Schreib- und Lesepointer des First-in-First-out-Speichers frei setzen kann. Damit kann sowohl beeinflußt werden, an welcher Stelle (auf das Bit genau) der Prozessor parallel liest bzw. schreibt, als auch an welcher Stelle (auf das Bit genau) das Drahtlos-Interface seriell schreibt bzw. liest.

Eine weitere Erhöhung der Flexibilität des Zugriffs wird dadurch erlaubt, daß der Prozessor die einzelnen Bits des First-in-First-out-Speichers als Bestandteile seines logischen Adressenbereichs betrachtet und diese wahlfrei und un-

abhängig von den Schreib- bzw. Lesepointern beschreiben bzw. lesen kann. Diese Lösung eignet sich besonders in Verbindung mit der Realisierung des First-in-First-out-Speichers mittels physikalischer RAM-Zellen aus dem normalen Arbeitsspeicher des Prozessors.

Bei jeder der beschriebenen Ausführungsformen des First-in-First-out-Speichers kann zusätzlich eine Vergleichsschaltung im First-in-First-out-Speicher oder eine automatische Checksummenprüfung vorgesehen werden.

Zusätzlich zum First-in-First-out-Speicher kann auch noch eine UART-Schaltung gem. dem Stand der Technik vorgesehen werden, um beispielsweise lange Protokolle (bei einem 32-Byte-First-in-First-out-Speicher beispielsweise 40-Byte-Protokolle) konventionell bearbeiten zu können.

Besonders bevorzugt ist es, den First-in-First-out-Speicher in Form einer Ringstruktur zu organisieren. Dann kann bei einem drohenden Überlauf des First-in-First-out-Speichers der Prozessor aktiviert (aufgeweckt) werden, damit dieser Daten aus dem First-in-First-out-Speicher abarbeitet.

Im Bedarfsfall kann der First-in-First-out-Speicher auch wie eine "klassische" UART-Schaltung des Standes der Technik betrieben werden.

Der First-in-First-out-Speicher kann selbstverständlich auch mit Prozessoren ohne Stromsparmodus realisiert werden.

Zur zusätzlichen Entlastung des Prozessors kann ein automatischer Vergleicher in der Interface-Schaltung realisiert werden. Dadurch wird der Prozessor von der Aufgabe des Vergleichens von empfangenen Daten mit erwarteten Daten (beispielsweise zur Authentifizierung) entlastet werden. Erfindungsgemäß werden diese Aufgaben durch einen First-in-First-out-Speicher mit eingebautem Vergleicher erledigt, wie dies in

Fig. 5 dargestellt ist. Dadurch können die Leistungsanforderungen an den Prozessor und damit der Stromverbrauch weiter gesenkt werden. Ebenfalls ist es eventuell möglich, einen preisgünstigeren Prozessor zu verwenden.

Die in Fig. 5 dargestellte Vergleichsschaltung benötigt die erwarteten Daten, beispielsweise eine Seriennummer oder ein Passwort, im First-in-First-out-Speicher. Eine einfache Logik 50 vergleicht automatisch jedes empfangene Bit (New Bit, 2) mit dem Inhalt der Speicherzelle 41 im First-in-First-out-Speicher 52 auf die es geschrieben werden soll. Somit dient der Schreibpointer 53 sowohl der Adressierung des erwarteten Bits (1), mit dem verglichen werden soll, als auch der Adressierung der Schreibstelle des empfangenen Bits (2), das jetzt in den First-in-First-out-Speicher geschrieben werden soll (→3). Die neuen Bits werden dabei von der Sende-Empfangseinheit RFI (Radio Frequency Interface) geliefert.

Nachdem alle Bits eines Bytes verglichen und in den First-in-First-out-Speicher 52 geschrieben worden sind, wird, wenn alle Bits gleich waren, ein diesem Byte zugeordnetes Bit in einem speziellen Vergleichsregister 54 auf 0 gesetzt, sonst wird dieses Bit auf 1 gesetzt. Die Funktion ist dabei folgende, daß das Register 54 vor Beginn der Vergleichsoperation auf 0 gesetzt wird, und die Logik 50, sobald einmal der Zustand altes Bit ‡ neues Bit eintritt, die entsprechende Speicherzelle des Registers 54 auf 1 setzt. Anschließend wird mit dem folgenden Byte bitweise fortgefahren, bis der Datenblock verarbeitet ist. Auf den Inhalt des Registers 54, also die Vergleichsinformation, kann byteweise vom Prozessor zugegriffen werden. Durch Ausmaskieren von Vergleichsinformationen kann schnell und einfach eine neu empfangene Bitfolge bewertet werden.

Diese Vergleichsschaltung kann übrigens auch in Verbindung mit einer UART-Schaltung gem. dem Stand der Technik verwendet

werden. Auf diese Weise kann auch ein automatischer Vergleicher ohne First-in-First-out-Speicher realisiert werden.

Die prinzipielle Funktion eines First-in-First-out-Speichers ist in der Figur 6 dargestellt. Der First-in-First-out-Speicher besteht aus einer Reihe von Speicherzellen, die zyklisch durch einen Lese- und einen Schreibpointer adressiert werden. Beim Schreiben in den First-in-First-out-Speicher wird der zu schreibende Wert an die Stelle geschrieben, an die der Schreibpointer zeigt und der Schreibpointer wird um eine Stelle inkrementiert. Beim Lesen wird geprüft, daß der Schreibpointer ungleich dem Lesepointer ist. Sodann wird der Wert der Speicherzelle ausgelesen, auf die der Lesepointer gezeigt hat, der Lesepointer wird inkrementiert. Wenn Schreibpointer gleich Lesepointer ist, ist der First-in-First-out-Speicher leer. Dies wird durch eine entsprechende Differenzschaltung 62 festgestellt. Dann wird entweder der Wert 0 oder die Mitteilung zurückgegeben, daß ein Lesen nicht möglich ist. Da die Differenzschaltung 62 also stets die Anzahl der Bits im First-in-First-out-Speicher angibt, kann diese Schaltung auch verwendet werden, um vor einem Überlauf des First-in-First-out-Speichers den Prozessor zu aktivieren und die Abarbeitung von Daten zu veranlassen, damit wieder Platz im First-in-First-out-Speicher geschaffen wird.

Eine weitere erfindungsgemäße Möglichkeit für den Datenvergleich besteht in der automatischen Erstellung einer Prüfsumme (Checksummer) der empfangenen Daten, beispielsweise nach dem CRC-Verfahren, die mit einer im voraus (beispielsweise während der Initialisierung) berechneten Prüfsumme der zu erwartenden Daten verglichen wird. Dieses Verfahren ist jedoch nicht so effizient, wie der in Fig. 5 beschriebene Vergleicher im First-in-First-out-Speicher, da die Checksummen der empfangenen Daten und der erwarteten Daten vom Prozessor verglichen werden müssen. Dies belastet also wiederum den Prozessor. Außerdem darf höchstens ein Bitfehler pro Datenblock vorhanden sein, damit dieser sicher erkannt wird. Darüber

hinaus muß eine Logik zwischen Daten und Befehlen unterscheiden, da sonst die Prüfsumme der empfangenen Daten unter Umständen über andere Befehle gebildet wird, als die Prüfsumme der erwarteten Daten. Dies ist im einzelnen in den Figuren 7 und 8 dargestellt:

Aus den erwarteten Daten wird die CRC-Prüfsumme berechnet. Diese besteht aus zwei Bytes. Es werden sodann aus den empfangenen Daten ebenfalls nach dem CRC-Verfahren die Prüfsummen gebildet. Es müssen dann lediglich 2 Byte-lange CRC-Prüfsummen verglichen werden, und nicht die ganzen Daten. Der Vergleich dieser kurzen CRC-Prüfsummen aus zwei Byte ist wesentlich schneller als der Vergleich der gesamten Daten. Die Prüfsumme der zu erwartenden Daten kann bereits lange im voraus errechnet werden. Wie in Fig. 8 dargestellt besteht auch die Möglichkeit, die empfangenen Daten gleichzeitig in den First-in-First-out-Speicher und in eine Checksummenlogik einzugeben. Diese erzeugt in Echtzeit die Checksummen, so daß der Prozessor in Echtzeit nur noch den wenig Rechenzeit bedürftigen Checksummenvergleich durchführen muß.

Auf diese Weise kann durch Hinzufügen eines First-in-First-out-Speichers, sowie ggf. eines Vegleichers oder eines Prüfsummengenerators, der automatisch eine Prüfsumme über die empfangenen Daten bildet, der Prozessor erheblich entlastet werden, im Vergleich zu dem üblichen UART-Konzept. Dadurch kann die Frequenz und damit der Stromverbrauch des Prozessors niedriger sein. Niedrigerer Stromverbrauch bedeutet bei einer kontaktlosen Chipkarte insbesondere größere Reichweite.

Besonders bevorzugt ist es dabei, einen Prozessor zu verwenden der während des Sendens, Empfangens, oder solange er nicht beschäftigt ist, in einen Stromsparmodus versetzt werden kann. Der Prozessor kann dann beispielsweise, wenn er mit der Vorbereitung für das Senden/Empfangen fertig ist, in den Stromsparmodus geschaltet werden, bis das Senden/Empfangen beendet ist. Die sonst vom Prozessor benötigte Energie kann

dann entweder gespart werden oder sie steht der Sende- bzw. Empfangshardware zur Verfügung. Außerdem können während des Sende- bzw. Empfangsbetriebs keine Versorgungsspikes von dem Prozessor auf das drahtlos übertragene Signal durchschlagen. Geringerer Stromverbrauch und günstigere Energieverteilung auf dem Kartenchip sowie ein günstigeres Signalverhalten ergeben eine größere Reichweite.

Darüber hinaus kann die Sendefunktion auch zeitgesteuert ausgelöst werden, während sich der Prozessor im Stromsparmodus befindet. Ebenso kann eine "Autoreceive-Funktion", d.h. eine automatische Empfangsfunktion realisiert werden. Dabei kann während des Stromsparmodus des Prozessors nach dem Senden automatisch in den Empfangsbetrieb umgeschaltet werden, ohne den Prozessor aktivieren zu müssen.

Patentansprüche

1. Interface-Schaltung zur Übertragung von Daten über eine serielle Schnittstelle von und zu einem Prozessor (CPU), d a d u r c h g e k e n n z e i c h n e t, daß lediglich ein Speicher (FiFo) für mehrere Bus- oder Prozessor-Wortlängen (z. B. Byte) zwischen serieller Schnittstelle und Prozessor (CPU) angeordnet ist.

2. Schaltung nach Anspruch 1, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) Wort- oder bitweise beschrieben und ausgelesen werden kann.

3. Schaltung nach Anspruch 2, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) einen Schreib- und einen Lesepointer besitzt, die jedes Bit oder jedes Wort (z.B. Byte) einzeln adressieren können.

4. Schaltung nach Anspruch 1, 2 oder 3, d a d u r c h g e k e n n z e i c h n e t, daß der Prozessor (CPU) über einen Stromsparmodus (Sleepmode) mit minimalem Stromverbrauch verfügt.

5. Schaltung nach Anspruch 1, 2, 3 oder 4, d a d u r c h g e k e n n z e i c h n e t, daß in dem Speicher (FiFo) ein Vergleicher eingebaut ist.

6. Schaltung nach Anspruch 5, d a d u r c h g e k e n n z e i c h n e t, daß der Vergleicher eine einfache Logik (50) umfaßt, die automatisch jedes empfangene Bit (2) mit dem Inhalt der Speicherzelle (1) im Speicher (FiFo) vergleicht, auf die das empfangene Bit (2) geschrieben werden soll.

7. Schaltung nach einem der Ansprüche 1 bis 6, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) in das CPU-Modul (34) integriert ist.

8. Schaltung nach einem der Ansprüche 1 bis 6, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) in das Empfängermodul (40) integriert ist.

9. Schaltung nach einem der Ansprüche 1 bis 6, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) durch RAM-Zellen aus dem normalen RAM im Adressbereich des Prozessors (CPU) realisiert ist.

10. Schaltung nach einem der Ansprüche 1 bis 9, d a d u r c h g e k e n n z e i c h n e t, daß zusätzlich zum Speicher (FiFo) ein Checksummenmodul vorgesehen ist.

11. Schaltung nach Anspruch 10, d a d u r c h g e k e n n z e i c h n e t, daß zusätzlich zum Checksummenmodul ein Vergleicher vorgesehen ist, der die Checksumme der empfangenen Daten mit einer erwarteten vorausberechneten Checksumme vergleicht.

12. Schaltung nach einem der Ansprüche 1 bis 11, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) in Form einer Ringstruktur realisiert ist.

13. Schaltung nach einem der Ansprüche 1 bis 12, d a d u r c h gekennzeichnet, daß der Speicher (FiFo) mit einer Überlaufdetektionsvorrichtung versehen ist, die bei einem drohenden Überlauf des Speichers (FiFo) den Prozessor (CPU) aktiviert (aufweckt).

14. Verfahren zur Übertragung von Daten zwischen einer seriellen Schnittstelle und einem Prozessor, d a d u r c h g e k e n n z e i c h n e t, daß die Daten seriell bitweise empfangen und in einen Speicher (FiFo) eingelesen und vom Prozessor (CPU) dort byteweise wieder ausgelesen werden, bzw.

vom Prozessor (CPU) byteweise in den Speicher (FiFo) geschrieben und von dort bitweise gesendet werden.

15. Verfahren nach Anspruch 14, d a d u r c h g e k e n n z e i c h n e t, daß der Speicher (FiFo) vom Prozessor (CPU) nur sequentiell gelesen und beschrieben werden kann, indem Schreib- und Lesepointer des Speichers (FiFo) automatisch ohne Belastung des Prozessors (CPU) gesetzt werden.

16. Verfahren nach Anspruch 14, d a d u r c h g e k e n n z e i c h n e t, daß der Prozessor (CPU) Schreib- und Lesepointer des Speichers (FiFo) frei setzen kann.

17. Verfahren nach Anspruch 14 oder 16, d a d u r c h g e k e n n z e i c h n e t, daß der Prozessor (CPU) die einzelnen Speicherzellen des Speichers (FiFo) als Bestandteile des eigenen Adreßbereichs behandelt und diese somit wahlfrei lesen und beschreiben kann.

18. Verfahren nach einem der Ansprüche 14 bis 17, d a d u r c h g e k e n n z e i c h n e t, daß ein automatischer Datenvergleich vorgesehen wird, wobei die zu erwartenden Daten an der entsprechenden Stelle im Speicher (FiFo) abgespeichert sein müssen, und mittels einer einfachen Logik (50) jedes empfangene Bit (2) mit dem Inhalt der Speicherzelle (51) im Speicher (FiFo) verglichen wird, auf die es geschrieben wird.

19. Verfahren nach Anspruch 18, d a d u r c h g e k e n n z e i c h n e t, daß der Schreibpointer (53) des Speichers (FiFo) sowohl die Adressierung des erwarteten Bits, mit dem verglichen werden soll, als auch die Adressierung des empfangenen Bits (2), das jetzt in den Speicher geschrieben werden soll, liefert.

20. Verfahren nach Anspruch 18 oder 19, d a d u r c h g e k e n n z e i c h n e t, daß nach dem Vergleich aller Bits

eines Byte ein diesem Byte zugeordnetes Bit (4) auf 0 gesetzt wird, wenn alle Bits gleich waren, während es sonst auf 1 gesetzt wird.

21. Verfahren nach Anspruch 20, d a d u r c h g e k e n n z e i c h n e t, daß auf die zugeordneten Bits von dem Prozessor (CPU) byteweise zugegriffen werden kann.

22. Verfahren nach einem der Ansprüche 14 bis 21, d a d u r c h g e k e n n z e i c h n e t, daß der Prozessor (CPU) während der Datenübertragung in einen Stromsparmodus (Sleepmode) geschaltet wird.

23. Verfahren nach Anspruch 22, d a d u r c h g e k e n n z e i c h n e t, daß der Prozessor (CPU) bei einem drohenden Überlauf des Speichers (FiFo) aktiviert (aufgeweckt) wird.

24. Verfahren nach einem der Ansprüche 14 bis 17, d a d u r c h g e k e n n z e i c h n e t, daß ein automatischer Datenvergleich vorgesehen ist, wobei automatisch eine Prüfsumme der empfangenen Daten mit einer im voraus berechneten Prüfsumme der zu erwartenden Daten verglichen wird.

25. Verfahren nach einem der Ansprüche 22 bis 24, d a d u r c h g e k e n n z e i c h n e t, daß das Senden von Daten aus dem Speicher (FiFo) zeitgesteuert eingeleitet werden kann, ohne daß der Prozessor (CPU) dabei aktiviert (aufge-weckt) werden muß.

26. Verfahren nach einem der Ansprüche 22 bis 25, d a d u r c h g e k e n n z e i c h n e t, daß nach dem Senden von Daten automatisch in den Empfangsbetrieb umgeschaltet werden kann, ohne daß der Prozessor (CPU) dabei aktiviert (aufgeweckt) werden muß.

FIG 1
"Stand der Technik"

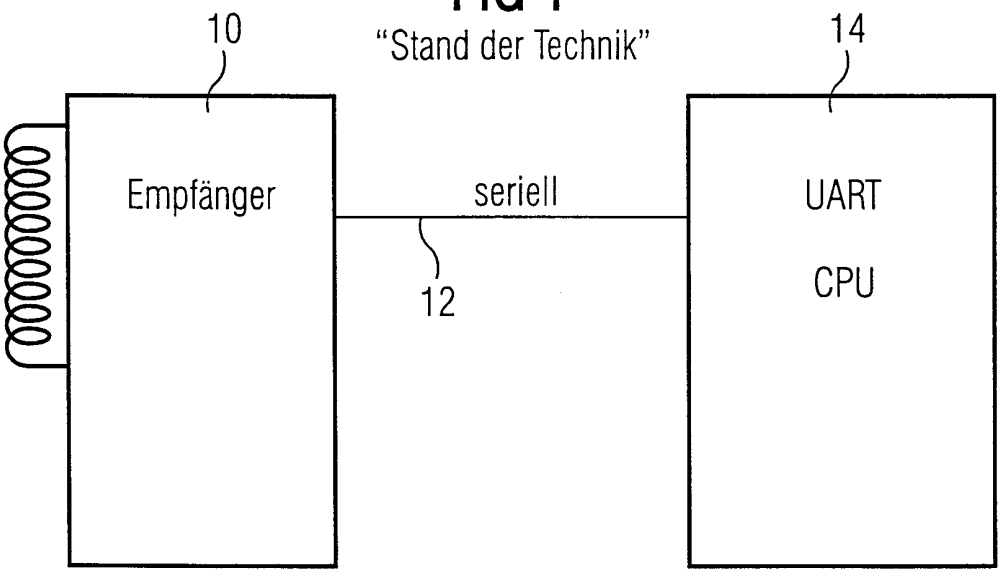


FIG 2
"Stand der Technik"

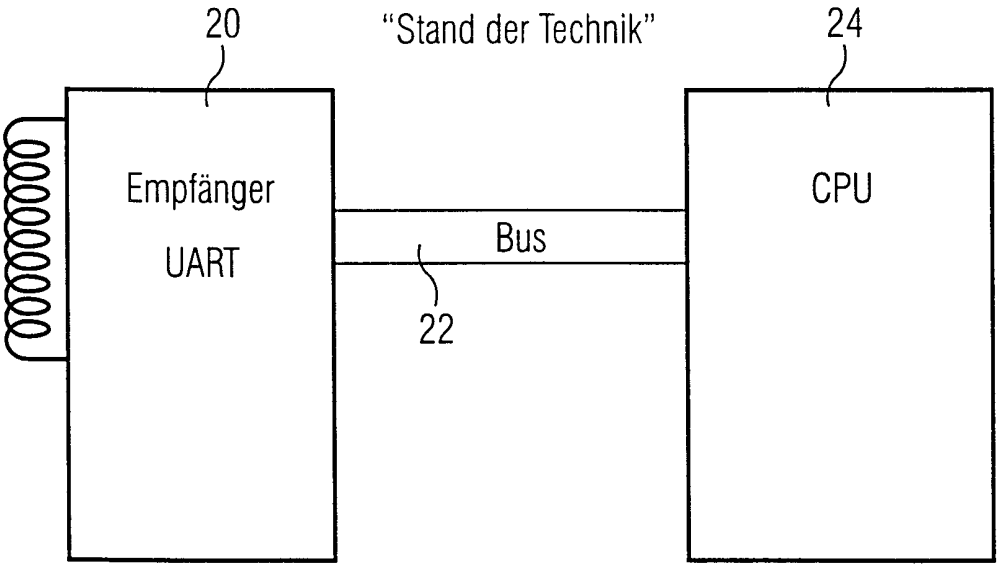


FIG 3

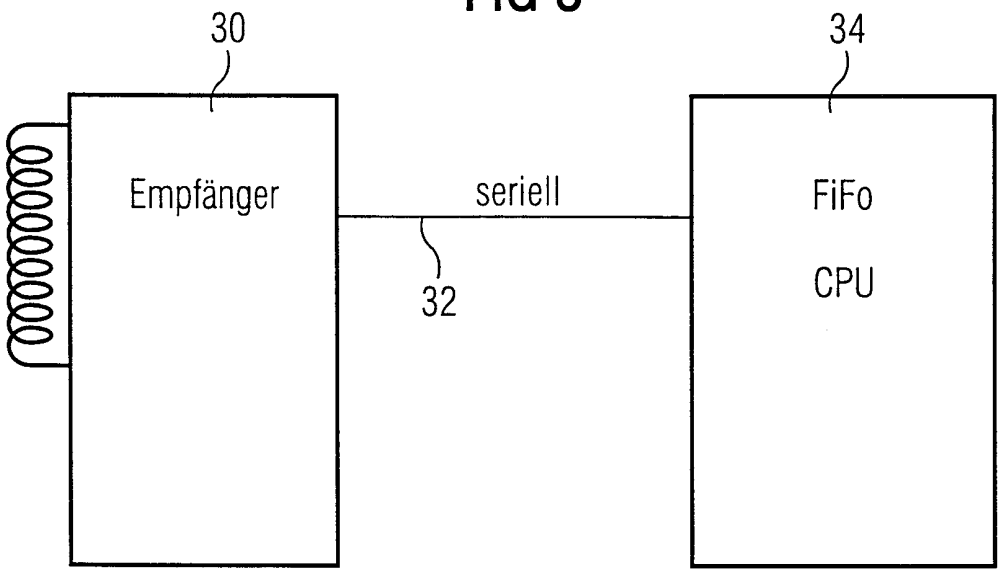

30
34
Empfänger
seriell
FiFo
CPU
32

FIG 4

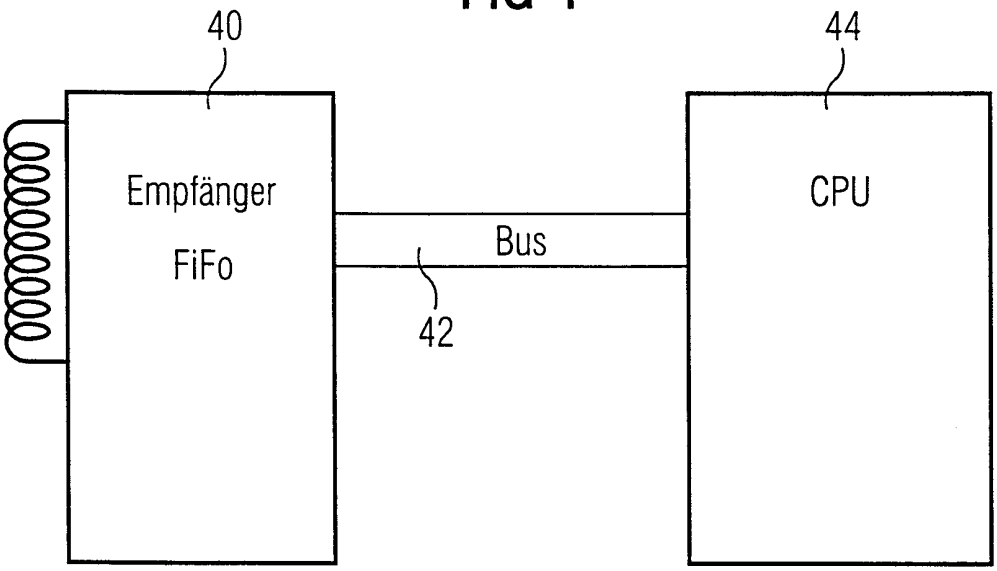

40
44
Empfänger
FiFo
Bus
CPU
42

FIG 5

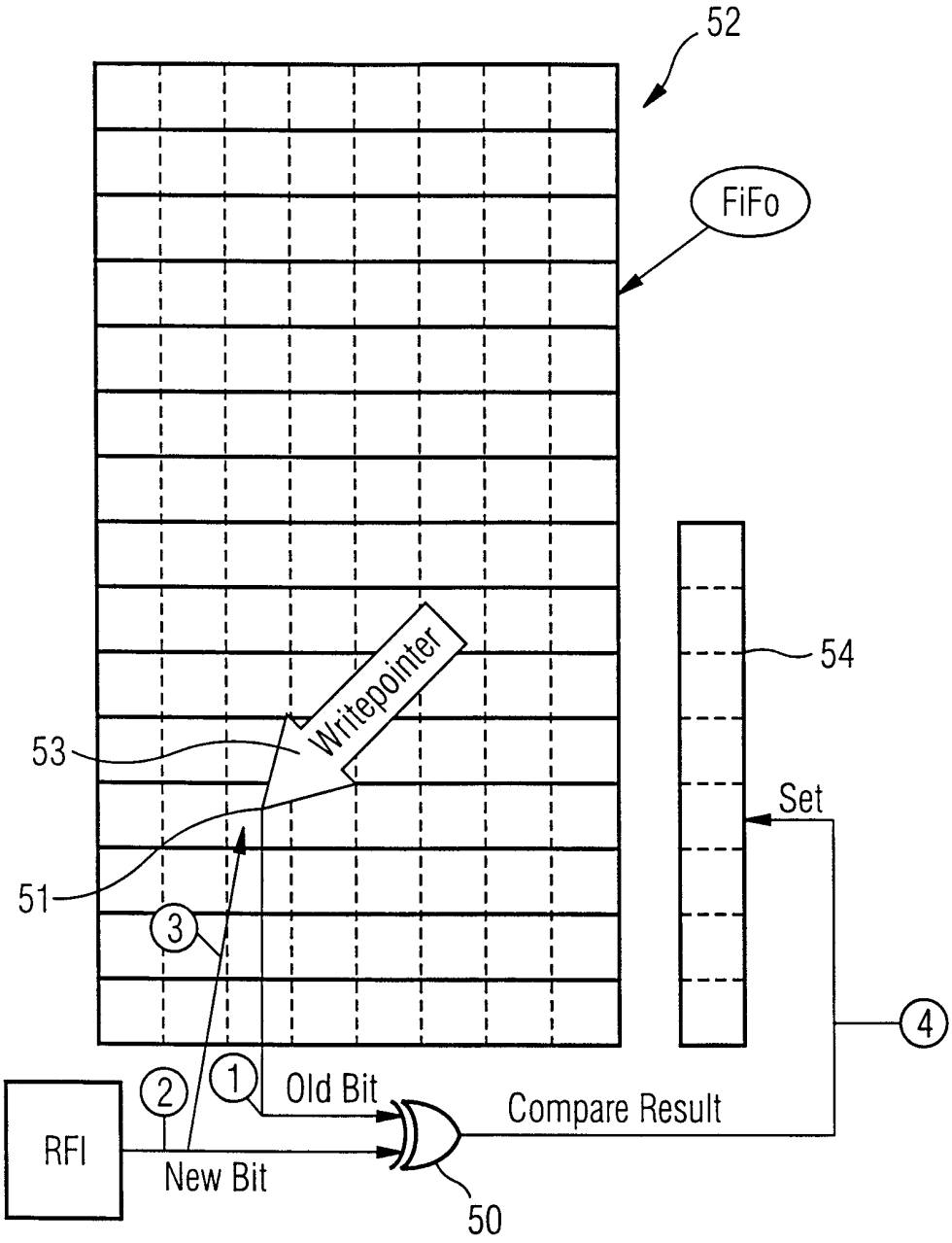

52
FiFo
Writepointer
53
54
Set
51
3
4
2
1
Old Bit
Compare Result
RFI
New Bit
50

FIG 6

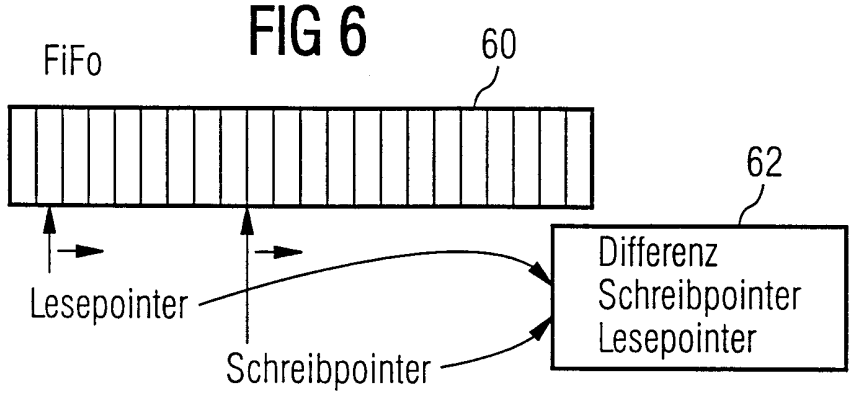


FIG 7

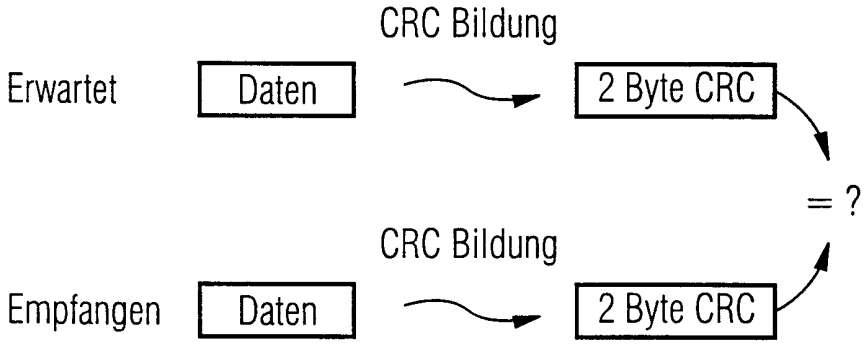


FIG 8