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

(10) **DE 602 11 921 T2** 2006.10.05

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 425 671 B1**
(21) Deutsches Aktenzeichen: **602 11 921.9**
(86) PCT-Aktenzeichen: **PCT/US02/27920**
(96) Europäisches Aktenzeichen: **02 763 598.6**
(87) PCT-Veröffentlichungs-Nr.: **WO 2003/021453**
(86) PCT-Anmeldetag: **29.08.2002**
(87) Veröffentlichungstag der PCT-Anmeldung: **13.03.2003**
(97) Erstveröffentlichung durch das EPA: **09.06.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **31.05.2006**
(47) Veröffentlichungstag im Patentblatt: **05.10.2006**

(51) Int Cl.[8]: ***G06F 13/00*** (2006.01)

(30) Unionspriorität:
315655 P **29.08.2001** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Analog Devices Inc., Norwood, Mass., US

(74) Vertreter:
Kuhnen & Wacker Patent- und Rechtsanwaltsbüro, 85354 Freising

(72) Erfinder:
TARDIEUX, Jean-Louis, Lexington, MA 02421, US;
Soerensen, Joern, 9600 Aars, DK

(54) Bezeichnung: **ARCHITEKTUR UND SYSTEM VON EINEM GENERISCHEN SERIELLEN PORT**

Beschreibung

Verwandte Anmeldungen

[0001] Diese Anmeldung beansprucht nach Artikel 35 US Code § 119 (e) die Priorität der am 29. August 2001 eingereichten US Patentanmeldung mit dem Titel „Digital Baseband Processor", von Allen, et al. Die oben genannte Patentanmeldung wird hierdurch durch Bezug hierin in ihrer Vollständigkeit aufgenommen.

Gebiet der Erfindung

[0002] Die vorliegende Erfindung betrifft im Allgemeinen programmierbare serielle Anschlüsse und insbesondere programmierbare serielle Hochgeschwindigkeitsanschlüsse.

Hintergrund der Erfindung

[0003] Auf dem Gebiet der elektronischen Schaltungen und in der Datenkommunikation sind viele Anordnungen serieller Anschlüsse bekannt. Anordnungen enthalten z. B. statische serielle Anschlüsse zum Übertragen und Empfangen von Daten eines ausgewählten seriellen Kommunikationsprotokolls (z. B. universale asynchrone Empfangs-Sendevorrichtungen (UART)), sowie konfigurierbare serielle Anschlüsse (z. B. Mikroprozessoren mit softwaregesteuerten seriellen Anschlüssen). Konfigurierbare serielle Anschlüsse bieten die Möglichkeit eine Vielzahl von Protokollen mit einer einzelnen Vorrichtung mit seriellem Anschluss zu bedienen bzw. betreiben. Statische und konfigurierbare serielle Anschlüsse werden für einen breiten Bereich von Anwendungen einschließlich der Kommunikation mit Anzeigevorrichtungen, sowie der Kommunikation mit Modems verwendet, und dienen als ein Universalsystemverbinder (USC = Universal System Connector).

[0004] Es wurden zahlreiche serielle Kommunikationsprotokolle (hierin ebenfalls als Protokolle bezeichnet) veröffentlicht (z. B. um einige zu nennen, UART, I^2C, HC11, IrDa), von welchen jedes spezifische Parameter definiert, nach welchen serielle Daten-Bits zwischen seriellen Anschlüssen kommuniziert werden. Die Parameter, welche ein Protokoll definieren, können Faktoren wie z. B. den Zeitablauf der empfangenen oder gesendeten Bits, elektrische Parameter (z. B. Signalpolarität, Leitungstreibercharakteristiken wie z. B. Open-Source oder Open-Collector Ausgabe Impedanzen, etc.), und logische Definitionen von Bit-Bedeutungen und Sequenzen enthalten.

[0005] Ein Beispiel eines konfigurierbaren seriellen Anschlusses wird durch die Motorola M68HC11 Familie von Mikrocontrollern vorgesehen, welche einen im Stand der Technik als Motorola synchrone serielle Peripherschnittstelle (SPI) bekannten programmierbaren seriellen Anschluss enthält. Solche seriellen Schnittstellen können aus verschiedenen Gründen nachteilhaft sein. Der Prozessor muss z. B. ein Softwareprogramm zum Steuern des seriellen Anschlusses ausführen und sämtliche der über Signalwege zu sendenden Daten-Bits passieren durch den Prozessor, wodurch der Prozessor über die Belastung hinaus, welche dem Prozessor auferlegt wird, während dieser die Aufgabe durchführt, für welche er ansonsten verwendet wird, belastet. Da die serielle Hardware ein leistungsverbrauchendes Teil des Prozessors ist, wird wenn immer der Prozessor ein Softwareprogramm ausführt, zusätzliche Leistung verbraucht, auch falls der serielle Anschluss nicht aktiv ist.

[0006] Als eine Alternative zu Mikrocontroller-gesteuerten programmierbaren seriellen Anschlüssen, sind programmierbare serielle Anschlüsse entwickelt worden, welche eine Zustandsmaschine implementieren, um den Prozessor von der Bürde zu entlasten, viele Aspekte des seriellen Anschlusses steuern zu müssen. Ein Beispiel eines solchen programmierbaren seriellen Anschlusses ist in der am 3. November 2000 eingereichten US Patentanmeldung „Generic Serial Port Architecture and System" von Sorenson mit der Seriennummer 09/706,450 gegeben.

[0007] Fig. 1 ist ein Blockdiagramm eines solchen programmierbaren seriellen Anschlusses **100**. In Fig. 1 empfängt ein Schieberegister **110**, wenn dieses im Sendemodus beschrieben wird, einen parallelen Satz von Daten-Bits auf den Kanälen **102** von einem Pufferspeicher **120** und gibt die Bits als eine serielle Ausgabe auf Kanal **104** (über einen Treiber **180**) unter der Steuerung einer Zustandsmaschine **120** aus. Wenn das Schieberegister **110** im Gegensatz dazu in einem Empfangsmodus betrieben wird, empfängt dieses eine serielle Eingabe auf Kanal **104** und gibt einen parallelen Satz von Daten-Bits auf den Kanälen **102** aus.

[0008] Der Begriff (Zustandsmaschine) ist hierin als eine Vorrichtung definiert, welche einen existierenden Status (z. B. einen Programmzähler und eine Vielzahl von anderen Registern) speichert und nach Empfangen einer Eingabe (z. B. einer Instruktion oder eines Befehls) in einen neuen Status wechselt und/oder eine deter-

ministische Aktion oder eine Ausgabe einleitet, welche in Reaktion auf den existierenden Status und die Eingabe stattfindet. Während Zustandsmaschinen eine arithmetische logische Einheit (ALU) oder andere Schaltungen, welche im herkömmlichen Sinne mit Mikroprozessoren verbunden sind, enthalten können, schließt der Begriff Zustandsmaschine, wie hierin definiert, solche Schaltungen oder Elemente nicht aus.

[0009] In dem programmierbaren seriellen Anschluss **100** werden Instruktionen, welche Regeln zum Implementieren von zwei oder mehreren Protokollen entsprechen, in einem Speicher **130** gespeichert. Unter Verwendung der Instruktionen des Speichers **130** führt die Zustandsmaschine **120** Instruktionen aus, welche einem durch den Controller **150** ausgewähltem Protokolle entsprechen, um eine Ausgabe entsprechend einem angegebenen Protokoll auf Kanal **104** vorzusehen. Ein Bitzähler **170** sieht an die Zustandsmaschine **120** eine numerische Anzahl der gesendeten Bits vor, um es zu erleichtern, eine Ausgabe entsprechend dem ausgewählten Protokoll vorzusehen, da das Verarbeiten eines Bits häufig von der Position in entweder den parallelen Kanälen **102** oder dem seriellen Empfangs- oder Sendebitstream auf Kanal **102** abhängt.

[0010] Eine Ausgabe wird typischerweise durch einen Treiber **180** geleitet, um eine Ausgabe vorzusehen, welche spezifische elektrische Parameter aufweist bzw. mit diesen kompatibel ist. In herkömmlichen, auf Zustandsmaschinen basierenden programmierbaren seriellen Anschlüssen (z. B. der programmierbare serielle Anschluss **100**) erfordert die Ausführung von Instruktionen zum Vorsehen einer Ausgabe entsprechend einem ausgewählten Protokoll, dass die Zustandsmaschine **120** von dem Taktgenerator **160** ein Taktimpuls empfängt und an das Schieberegister **110** ein Taktsignal vorsieht, um die Ausgabe jedes Bits von dem Schieberegister **110** je nachdem an den Kanal **104** oder dem Pufferspeicher **120** zu steuern, und erfordert, dass die Zustandsmaschine eine durch einen Bitzähler **170** vorgesehene Bitanzahl bzw. Zählung beibehält und verarbeitet.

[0011] Das Vorsehen einer Ausgabe entsprechend einem ausgewählten Protokoll erfordert, dass eine Ausgabe zu angegebenen Zeiten vorgesehen wird. In manchen Protokollen muss eine Ausgabe auf Kanal **104** z. B. innerhalb eines bestimmten bzw. angegebenen Zeitraums auf den Empfang eines Zeitablaufsignals (z. B. eine ansteigende Flanke auf einem Kanal **190**) durch den programmierbaren seriellen Anschluss **100** hin erfolgen. Da das Zeitintervall zwischen dem Empfang des Taktsignals und dem Vorsehen der Ausgabe sehr kurz sein kann muss eine Instruktion, um eine Ausgabe zu erreichen, in der Lage sein kurze Ausführungszeiten aufzuweisen, sonst kann der serielle Anschluss eine unzureichende Datenausgabegeschwindigkeit aufweisen und in manchen Fällen kann es sein, dass ein Betrieb von manchen Protokollen nicht möglich ist.

[0012] Fig. 2 ist ein Ablaufdiagramm **200** eines typischen Satzes von Instruktionen für einen herkömmlichen programmierbaren seriellen Anschluss, um eine Standardausgabe (z. B. eine UART-kompatible Ausgabe) zu erreichen. Bei Schritt **205** wartet die Zustandsmaschine auf eine Anzeige, das das Schieberegister mit Daten gefüllt ist (d. h. einem parallelen Datensatz). Bei Schritt **210** lädt die Zustandsmaschine den Treiber mit einem Startzustand (z. B. ein logischer Wert 1 oder 0). Bei Schritt **220** initialisiert die Zustandsmaschine den Bitzähler (z. B. eine Anfangsbitanzahl wird geladen). Bei Schritt **240** identifiziert die Zustandsmaschine die Codezeilen, der Schleife, durch welche Daten von dem Schieberegister verschoben werden. Das erste Daten-Bit wird gesendet und für so viele Taktzyklen wie für das Protokoll notwendig sind, beibehalten, und der Bitzähler wird bei den Schritten **250** bzw. **260** durch die Zustandsmaschine gesenkt. Nachfolgende Daten-Bits werden gesendet und der Bitzähler wird durch die Zustandsmaschine (bei Schritt **270**) gesenkt, bis der Bitzähler Null erreicht. Nachdem alle Daten-Bits gesendet sind, veranlasst die Zustandsmaschine, dass ein Paritäts-Bit bei Schritt **280** gesendet wird. Bei Schritt **290** wird der Treiber schließlich in einen Stop-Zustand versetzt.

[0013] Ein Verfahren zum Erreichen schnellerer Ausführungszeiten ist es, die Taktrate, mit welcher die Zustandsmaschine Instruktionen ausführt, zu erhöhen, so dass eine größere Anzahl von Instruktionen in einem gegebenen Zeitintervall ausgeführt werden (z. B. das Zeitintervall zwischen einem Zeitsteuersignal und dem Beginn der Ausgabe von seriellen Daten-Bits); eine schnellere Taktrate kann jedoch schnellere und teurere elektronische Komponenten erfordern. Eine schnellere Taktrate kann zusätzlich einen erhöhten Leistungsverbrauch erfordern. Es wird dementsprechend ein programmierbarer serieller Anschluss benötigt, welcher in der Lage ist, notwendige Eingabe (z. B. Zeitsteuersignale) zu empfangen und verarbeiten, und notwendige Ausgaben mit einer relativ hohen Geschwindigkeit vorzusehen, während dieser eine relativ geringe Taktgeschwindigkeit beibehält. Es wird darüber hinaus ein programmierbarer serieller Anschluss benötigt, welcher in der Lage ist, entsprechend einer breiten Vielfalt von Protokollen Ausgaben vorzusehen und Eingaben anzunehmen.

[0014] WO02/42919 offenbart eine Ablaufsteuerung, welche Instruktionen basierend auf einer Funktionstaktfrequenz ausführt, um Eingabe/Ausgabefunktionen in einer seriellen peripheren Schnittstelle durchzuführen. Der Funktionstakt weist eine Frequenz auf, welche ein zweifaches des Quelltaktsignals beträgt. Ein Funktions-

taktgenerator erzeugt das Funktionstaktsignal und wählt die zu entschlüsselnde Instruktion aus. Die Instruktion gibt wahlweise die Frequenz des Funktionstaktsignals vor, so dass dieses während der Zeitdauer des Funktionstaktsignals, während welcher die vorbestimmte Instruktion entschlüsselt wird, um eine einzelne Instruktion während des Quelltaktsignals auszuführen, der Frequenz des Quelltaktsignals gleicht.

Zusammenfassung der Erfindung

[0015] Entsprechend einem ersten Aspekt der vorliegenden Erfindung ist ein programmierbarer serieller Anschluss vorgesehen, aufweisend:
ein erstes Schieberegistermodul, welches ein Schieberegister enthält, dass einen Eingabekanal zum Empfangen einer parallelen Eingabe einer ersten Vielzahl von Bits einen ersten Ausgabekanal zum Vorsehen einer seriellen Ausgabe einer zweiten Vielzahl von Bits, sowie ein weiteres Register zum Steuern des Schieberegistermoduls aufweist;
einen Taktgenerator; und
eine erste Zustandsmaschine,
dadurch gekennzeichnet, dass der Taktgenerator ein erstes Taktsignal an das erste Schieberegistermodul vorsieht, und der Taktgenerator, sowie das erste Schieberegistermodul auf Instruktionen von der Zustandsmaschine reagieren, so dass die Zustandsmaschine zumindest eine Instruktion ausgeben kann, um eine Ausgabe der zweiten Vielzahl von Bits von dem Schieberegister ohne eine weitere Steuerung durch die Zustandsmaschine zu bewirken.

[0016] Der Taktgenerator kann ferner einen zweiten Ausgabekanal an ein zweites Schieberegistermodul aufweisen, wobei der zweite Ausgabekanal ein zweites Taktsignal, welches eine vorbestimmte Anzahl von Impulsen aufweist, mit einer vorbestimmten Frequenz in Reaktion auf zumindest eine Instruktion vorsieht. Der Taktgenerator kann ferner optional einen zweiten Ausgabekanal an eine Zustandsmaschine aufweisen, wobei der zweite Ausgabekanal ein zweites Taktsignal an die Zustandsmaschine vorsieht. In manchen Ausführungsformen weist der Taktgenerator ferner ein mit dem zweiten Ausgabekanal gekoppeltes Gate auf, so dass das zweite Taktsignal durch das Gate passiert und wobei das Gate durch die zumindest eine der Instruktionen, welche die vorbestimmte Anzahl von Impulsen anzeigt, und die vorbestimmte Frequenz steuerbar ist. In anderen Ausführungsformen des ersten Aspekts der Erfindung weist der Taktgenerator ferner einen Teiler auf, welcher mit dem zweiten Ausgabekanal gekoppelt ist, um das zweite Taktsignal zu empfangen, sowie mit dem ersten Ausgabekanal gekoppelt ist, um das erste Taktsignal vorzusehen, wodurch das zweite Taktsignal geteilt wird, um das erste Taktsignal zu bilden.

[0017] Die Zustandsmaschine kann konfiguriert und angeordnet sein, um Instruktionen entsprechend einer Vielzahl von seriellen Kommunikationsprotokollen vorzusehen. Jedes der zweiten Vielzahl von Bits kann in Reaktion auf einen Taktimpuls der zweiten Vielzahl von Impulsen ausgegeben werden.

[0018] In manchen Ausführungsformen weist das Schieberegistermodul ferner einen Bitzähler auf, wobei der Bitzähler konfiguriert ist, einen numerischen Zählwert der seriellen Ausgabe der zweiten Vielzahl von Bits beizubehalten. Der Bitzähler wird optional in Reaktion auf einen Taktimpuls der zweiten Vielzahl von Impulsen gesenkt. Jedes der zweiten Vielzahl von Bits kann ein Daten-Bit, ein Paritäts-Bit, oder ein Stop-Bit sein. Jedes der zweiten Vielzahl von Bits wird optional basierend auf dem Bit-Zählwert ausgewählt.

[0019] Das Schieberegistermodul kann ferner einen Paritätsgenerator aufweisen. In manchen Ausführungsformen gibt das Scheiberegistermodul von dem Paritätsgenerator in Reaktion auf einen Taktimpuls der zweiten Vielzahl von Impulsen ein Paritäts-Bit aus. Der programmierbare serielle Anschluss kann ferner einen programmierbaren Treiber aufweisen, welcher mit dem ersten Ausgabekanal gekoppelt ist, um die elektrischen Parameter der seriellen Ausgabe zu steuern. Ein Interrupt-Verarbeitungsmodul kann mit der ersten Zustandsmaschine gekoppelt sein, um einen Interrupt bzw. eine Unterbrechung der Zustandsmaschine zu bewirken.

[0020] Der programmierbare serielle Anschluss kann ferner ein zweites Schieberegistermodul aufweisen, wobei der zweite Taktgenerator mit dem zweiten Schiebereigistermodul gekoppelt ist, um ein zweites Taktsignal vorzusehen, welches eine zweiten Vielzahl von Taktimpulsen aufweist, und wobei das zweite Schieberegistermodul in Reaktion auf das zweite Taktsignal eine zweite serielle Ausgabe vorsieht.

[0021] Das Schieberegistermodul enthält vorteilhafterweise einen Bit-Zähler, welcher konfiguriert ist, einen Zählwert der Anzahl von Bits beizubehalten, welche durch den seriellen Anschluss ausgegeben werden. Der Bit-Zähler kann in Reaktion auf ein Taktimpuls gesenkt werden. In manchen Ausführungsformen weist das Schieberegistermodul ferner einen Paritätsgenerator auf. Das Schieberegistermodul kann in Reaktion auf den

numerischen Zählwert ein Paritäts-Bit vorsehen. Jedes der Vielzahl von Bits, welches der seriellen Ausgabe entspricht, kann ein Daten-Bit, ein Paritäts-Bit, oder ein Stop-Bit sein. Jedes der Vielzahl der Bits, welches der seriellen Ausgabe entspricht, kann optional basierend auf dem numerischen Zählwert ausgewählt werden.

[0022] Entsprechend einem zweiten Aspekt der vorliegenden Erfindung ist ein Verfahren zum Steuern eines Schieberegistermoduls vorgesehen, welches ein erstes Schieberegister aufweist, das eine erste Vielzahl von Bits enthält, um eine serielle Ausgabe entsprechend dem ausgewählten einer Vielzahl von seriellen Kommunikationsprotokollen vorzusehen, wobei das erste Schieberegistermodul mit einem Taktgenerator und einer Zustandsmaschine gekoppelt ist und das Verfahren aufweist.

[0023] Entsprechend dem ausgewählten Protokoll, Auswählen einer durch die Zustandsmaschine unter einer Vielzahl von Instruktionssequenzen auszuführende Instruktionssequenz, wobei jede der Instruktionssequenzen einem Protokoll entspricht; Steuern des Taktgenerators entsprechend einer Instruktion der ausgewählten Instruktionssequenz um eine erste Vielzahl von Taktimpulsen an das Schieberegistermodul vorzusehen, und Ausgeben einer zweiten Vielzahl von Bits, welche der ersten Vielzahl von Bits entsprechen in Reaktion auf die Taktimpulse und Ausführung der Instruktionssequenz ohne weitere Steuerung durch die Zustandsmaschine.

Kurze Beschreibung der Zeichnungen

[0024] Darstellende, nicht begrenzende Ausführungsformen der vorliegenden Erfindung werden anhand von Beispielen mit Bezug auf die begleitenden Zeichnungen beschrieben, in welchen gleiche Bezugszeichen zum Bezeichnen der gleichen Komponenten in unterschiedlichen Figuren verwendet werden. Es zeigt:

[0025] Fig. 1 ist ein Blockdiagramm eines herkömmlichen programmierbaren seriellen Anschlusses;

[0026] Fig. 2 ist ein Ablaufdiagramm eines typischen Instruktionssatzes für einen herkömmlichen programmierbaren seriellen Anschluss, um eine Standardausgabe (z. B. eine UART kompatible Ausgabe) zu erreichen;

[0027] Fig. 3A ist ein funktionelles Blockdiagramm einer ersten beispielhaften Ausführungsform eines programmierbaren seriellen Anschlusses entsprechend von zumindest manchen Aspekten der vorliegenden Erfindung;

[0028] Fig. 3B stellt ein Zeitablaufdiagramm für einen beispielhaften Ausgabe-Bit-Stream eines Schieberegistermoduls dar, welches konfiguriert ist, automatisch ein Paritäts-Bit und ein Stop-Bit auszugeben;

[0029] Fig. 4A ist ein funktionelles Blockdiagramm einer zweiten beispielhaften Ausführungsform eines programmierbaren seriellen Anschlusses entsprechend von zumindest manchen Aspekten der vorliegenden Erfindung;

[0030] Fig. 4B ist ein schematisches Diagramm eines beispielhaften Interrupt-Verarbeitungsmoduls;

[0031] Fig. 5 ist ein schematisches Diagramm einer beispielhaften Ausführungsform eines Transferregister-Schiebemoduls entsprechend von zumindest manchen Aspekten der vorliegenden Erfindung;

[0032] Fig. 6 ist ein schematisches Diagramm eines Beispiels eines Treibers (z. B. der Treiber in Fig. 3A), welcher in einem programmierbaren Anschluss verwendet werden kann;

[0033] Fig. 7 ist ein schematisches Diagramm eines Beispiels einer Ausführungsform eines Taktgenerators zur Verwendung mit zumindest manchen programmierbaren seriellen Anschlüssen entsprechend von Aspekten der vorliegenden Erfindung;

[0034] Fig. 8A stellt eine beispielhafte Taktgeneratorausgabe für einen Taktgenerator dar,. welcher einen Taktimpuls in einem Standardmodus erzeugt;

[0035] Fig. 8B stellt eine Taktgeneratorausgabe für einen Taktgenerator dar, welcher in einem Leistungssparmodus betrieben wird;

[0036] Fig. 9 ist ein Ablaufdiagramm einer beispielhaften Instruktionssequenz für einen programmierbaren seriellen Anschluss entsprechend von manchen Aspekten der vorliegenden Erfindung, um eine Standardausgabe (z. B. eine UART kompatible Ausgabe) zu erreichen;

[0037] Fig. 10A ist eine schematische Darstellung einer geeigneten Speicherordnung, welche zur Verwendung im Rahmen der vorliegenden Erfindung geeignet ist;

[0038] Fig. 10B ist eine schematische Darstellung einer beispielhaften Zustandsmaschinen-Dekodierarchitektur, welche zur Verwendung im Rahmen der in Fig. 10A gezeigten Speicherordnung geeignet ist; und

[0039] Fig. 11A und Fig. 11B sind Tabellen, welche einen beispielhaften Satz von Binärimplementierungen eines Instruktionssatzes darstellen.

Ausführliche Beschreibung

[0040] Fig. 3A ist ein funktionelles Blockdiagramm einer ersten beispielhaften Ausführungsform eines programmierbaren seriellen Anschlusses **300** entsprechend von zumindest manchen Aspekten der vorliegenden Erfindung. Der programmierbare serielle Anschluss **300** überträgt einen parallelen Satz von Daten-Bits, welcher eine Vielzahl von Bits (z. B. ein Byte von Daten) aufweist, auf den Kanälen **302**, und sieht eine serielle Ausgabe einer zweiten Vielzahl von Bits, welche der ersten Vielzahl von Bits entsprechen, auf einem Kanal **304** vor. Der programmierbare serielle Anschluss **300** kann ebenfalls verwendet werden, eine serielle Eingabe auf Kanäle **304** zu empfangen und eine parallele Ausgabe auf den Kanälen **302** vorzusehen. Der programmierbare serielle Anschluss **300** kann des weiteren im Simplex oder Halb-Duplexbetrieb betrieben werden. In der nachfolgenden Erörterung wird der Schwerpunkt hauptsächlich auf den Paralleleingabe/Serielleausgabe-Modus gelegt, da der Serielleingabe/Paralleleausgabe-Modus gewöhnlich ohne weitere Ausführungen ersichtlich ist.

[0041] Der programmierbare serielle Anschluss **300** enthält eine Zustandsmaschine **320**, einen Taktgenerator **360** (hierin ebenfalls als Taktimpulsgenerator bezeichnet), ein Schieberegister **312** und einen Treiber **380**. Ein Controller **350** (herkömmlicher Natur) steuert einige Funktionen des programmierbaren seriellen Anschlusses **300**, z. B. durch Initialisieren der obigen Komponenten, welche den programmierbaren seriellen Anschluss **300** aufweisen, durch Auswählen eines Protokolls, unter welchem die serielle Kommunikation stattfindet, und Füllen des Pufferspeichers **320**.

[0042] Wie nachstehend ausführlicher beschrieben ist, steuert die Zustandsmaschine **320** den Betrieb des Schieberegisters **312** und des Taktgenerators **360** durch Vorsehen von Befehlen an jede von ihnen, einschließlich dem Steuern von Konfigurationsregistern jeder von ihnen. Die Zustandsmaschine **320** führt Instruktionen entsprechend einem durch den Controller **350** ausgewähltem Protokollprogramm durch. Der Controller sieht z. B. eine Programmtextzeilenzahl von einem Satz von Instruktionen vor, welche einem gewähltem Protokollprogramm entsprechend vor, welches in einem Speicher **330** gespeichert ist. Der Speicher kann eine Vielzahl von Instruktionen enthalten, welche einem unterschiedlichen Protokoll entsprechen.

[0043] Ein Schieberegistermodul **312** empfängt eine parallele Eingabe einer Vielzahl von Bits von einem Pufferspeicher **320** auf den Kanälen **302**, und sieht eine serielle Ausgabe, welche der Vielzahl von Bits entspricht, an den Treiber **380** vor. Das Schieberegistermodul **312** enthält ein Schieberegister **310**, um auf dem Kanal **302** empfangene Daten zu seriallisieren bzw. in serieller Reihenfolge zu erstellen, sowie einen Bitzähler **370**. Der Bitzähler **370** ist konfiguriert, eine numerierte Anzahl von den Bits serieller Daten beizubehalten, welche durch das Schieberegistermodul **312** ausgegeben werden.

[0044] Der Taktgenerator **360** ist mit der Zustandsmaschine **320** gekoppelt, und sieht einen oder mehrere Taktimpulse an das Schieberegister **310** in Reaktion auf zumindest eine Instruktion der Zustandsmaschine **320** vor. Die Taktimpulse steuern die Zeitvorgabe des Schiebens von Daten in das und aus dem Schieberegistermodul **312**. Der Taktgenerator **360** weist einen Eingabekanal auf, um zumindest eine Instruktion von der Zustandsmaschine **320** zu empfangen, und einen Ausgabekanal, um Taktimpulse an die Zustandsmaschine **320** vorzusehen. In Reaktion auf z. B. zumindest eine Instruktion, kann der Taktgenerator **360** eine vorbestimmte Anzahl von Taktimpulsen an das Schieberegister **310** mit vorbestimmter Frequenz zu einem vorbestimmten Zeitpunkt (oder nach einer vorbestimmten Verzögerung) vorsehen. In einigen Ausführungsformen sieht die Zustandsmaschine **320** einen einzelnen Befehl an den Taktgenerator **360** vor, um eine vorbestimmte Anzahl von Impulsen mit einer vorbestimmten Frequenz zu erzeugen, um die Ausführungszeitdauer, welche zum Steuern des Taktgenerators notwendig ist, zu verringern. Die Anzahl der durch die Zustandsmaschine durchgeführten Instruktionen, um eine Ausgabe entsprechend einem gewählten Protokoll zu erreichen ist deshalb relativ gesehen geringer und die Gesamtdurchführungszeitdauer zum Erreichen einer Ausgabe ist relativ kurz; der programmierbare serielle Anschluss **300** ist somit in der Lage, Ausgaben vorzusehen und Eingaben entsprechend einer breiten Vielfalt von Protokollen zu akzeptieren. Die Zustandsmaschine **320** und der Taktgenerator **360**

bilden zusammen ein Schieberegistersteuermodul **355** zum Steuern des Schieberegistermoduls **312** aus.

[0045] In Reaktion auf einen von dem Taktgenerator **360** empfangenen Taktimpuls (welchem andere Taktimpulse für andere Zwecke vorangegangen sein können) gibt das Schieberegistermodul **312** ein einzelnes Datenbit an den Treiber **380** aus, und der Bitzähler **370** wird gesenkt oder erhöht (abhängig davon, ob ein Protokoll angibt, dass das Bit mit dem höchsten Stellenwert oder das Bit mit dem geringsten Stellenwert zuerst übertragen werden soll). Das Schieberegister **312** kann derart konfiguriert sein, so dass für ausgewählte Protokolle der erste Taktimpuls, welcher empfangen wird, nachdem der Zähler Null erreicht, automatisch in einer Ausgabe des Paritäts-Bits resultiert, und der zweite empfangene Taktimpuls, nachdem der Zähler Null erreicht in der Ausgabe eines Stop-Bits resultiert.

[0046] Fig. 3B stellt ein Zeitablaufdiagramm für einen beispielhaften Ausgabe-Bitstream **390** von einem Schieberegistermodul **312** dar, welches konfiguriert ist, automatisch ein Paritäts-Bit und ein Stop-Bit auszugeben. Zusätzlich zu dem Ausgabe-Bitstream **390** sind ein entsprechender Bit-Zählwert **392** des Bitzählers **370** (in Fig. 3A oben gezeigt) und ein Taktsignal **394** des Taktgenerators **360** (in Fig. 3A gezeigt) dargestellt. In dem beispielhaften Bitstream **390**, wird angenommen, dass die Daten-Bits an der ansteigenden Flanke des Taktsignals **394** ausgegeben werden. Nach Empfang jeder der jeweiligen Flanken **396a** bis d (entsprechend der Bit-Zählwerte 1 bis 4), wird ein entsprechendes Datenbit ausgegeben. Nach Empfang der ersten ansteigenden Flanke **396e**, wenn der Bit-Zählwert Null beträgt, wird ein Paritäts-Bit ausgegeben; und nach Empfang der zweiten ansteigenden Flanke **396f**, wenn der Bit-Zählwert Null beträgt, wird ein Stop-Bit ausgegeben.

[0047] Nochmals auf Fig. 3A Bezug nehmend ist der Treiber **380** mit dem Kanal **304** gekoppelt, um Leitungssteuer- und Empfangsschaltungen und für das gewählte Protokoll erforderliche Parameter vorzusehen. Der Treiber **380** kann z. B. die Wahl einer gewünschten Leitungstreiber-Schaltungsart, wie z. B. einer Open-Source oder Open-Collector Leitungstreiberschaltung, die Wahl der Polarität eines Ausgabesignals und die Wahl eines Hochimpedanzzustands erlauben. Der Treiber **380** kann zusätzlich die Wahl einer Datenquelle erlauben (z. B. Ausgabe an einem festen logischen Wert (d. h. eins oder zwei), Eingabe/Ausgabe von einer Zustandsmaschine, oder eine Eingabe/Ausgabe von einem Schieberegister). Der Treiber kann ebenfalls Eingabedaten/Ausgabedaten Fehlzuordnungen erfassen. Im Halb-Duplexbetrieb kann der Treiber **380** zusätzlich auf dem Kanal **304** gesendete und empfangene Daten multiplexen. Der Treiber **380** ist vorzugsweise in Reaktion auf ein Protokollauswahlsignal, welches wie gezeigt auf Leitung **381** bereit gestellt wird, programmierbar, um protokollbezogene Auswahlen und Betriebe auszuführen. Das Protokollauswahlsignal kommt direkt oder indirekt von dem Kontrolle **350**. Während der Treiber **380** als den Ausgabekanal **304** aufweisend dargestellt ist, ist es ersichtlich, dass der Treiber **380** einen oder mehrere zusätzliche Ausgaben, wie z. B. eine Taktsignal bzw. Taktsignale vorsehen kann. Der Treiber **380** wird ferner nachstehend mit Bezug auf Fig. 6 beschrieben.

[0048] Fig. 4A ist ein funktionales Blockdiagramm einer zweiten beispielhaften Ausführungsform eines programmierbaren seriellen Anschlusses **400** entsprechend von zumindest einigen Aspekten der vorliegenden Erfindungen. Der programmierbare serielle Anschluss **400** enthält einen Treiber **480**, zwei Zustandsmaschinen **420** und **421**, von welchen jede einen entsprechenden Speicher **430** und **431** aufweist, Schieberegistermodule **412** und **413**, und einen Taktgenerator **460** und **461**. Die Schieberegistermodule **412** und **413** weisen entsprechende Schieberegister **410** und **411**, sowie Bitzähler **470** und **471** auf. Die Schieberegister **410** und **411** empfangen parallele Daten-Bits von Pufferspeichern **422** und **423** und sehen jeweils serielle Ausgaben auf den Kanälen **404** und **405** vor.

[0049] Wie oben steuert ein Controller **450** einige Funktionen des programmierbaren seriellen Anschlusses **400**, z. B. durch Initialisieren jeglicher der obigen Komponenten, welche den programmierbaren seriellen Anschluss **400** aufweisen und durch Auswahl eines Protokolls, unter welchen die Eingaben und Ausgaben verarbeitet werden.

[0050] Der programmierbare serielle Anschluss **400** ist konfiguriert, um ein gleichzeitiges Senden und Empfangen von Daten über die jeweiligen Schieberegistermodule **412** und **413** zu erlauben; der programmierbare serielle Anschluss **400** ist dementsprechend in der Lage, eine Voll-Duplexkommunikation oder Halb-Duplexkommunikation durchzuführen.

[0051] Um die Voll-Duplexkommunikation zu erreichen, arbeiten die Schieberegistermodule **412** und **413** gleichzeitig, Daten zu senden und zu empfangen; und um in Halb-Duplexbetrieb zu arbeiten, senden/empfangen die Schieberegistermodule **412** und **413** Daten in abwechselnden Zeitzyklen.

[0052] In manchen Ausführungsformen enthält der programmierbare serielle Anschluss **400** zwei Ablauf-

bzw. Ereigniszähler (eventcounter) **480** und **481**, welche jeweils den Zustandsmaschinen **420** und **421** zugehörig sind. Die Ablaufzähler **480** und **481** sind Register, welche in Reaktion auf eine Eingabe wie z. B. ein Taktsignal oder ein Signal von den Zustandsmaschinen **420** und **421** erhöht oder vermindert werden. Jeder der Ablaufzähler **480** und **481** ist in der Lage, einen entsprechenden Zählregisterwert vorzusehen. Die Zähler **480** und **481** sind konfigurierbar, in Reaktion auf eine Eingabe von einer Zustandsmaschine oder einer anderen Quelle (z. B. einem Taktgeber) erhöht oder vermindert zu werden. Auf einen Zählregisterwert kann z. B. über die Vergleichs-Instruktion (nachstehend erörtert) zugegriffen werden, und ein Zählregister kann zusätzlich festverdrahtet werden, um eine Ausgabe an eine zugehörige Zustandsmaschine nach dem Auftreten eines Ablaufs bzw. Ereignisses vorzusehen (z. B. der Zähler **480**, **481** wird auf Null vermindert, oder ein Registerüberlauf ist aufgetreten). In einigen Ausführungsformen wird die Ausführung angegebener Instruktionen (eine Verzögerungsinstruktion, ein Wartebefehl oder ein Taktbefehl, welcher den Leistungssparmodus aufruft (jede dieser Instruktionen ist nachstehend beschreiben)) ein Taktsignal **705** und **706** einer Zustandsmaschine (in Fig. 7 sichtbar) durch ein Gate **708**, **709** gesperrt, bis die Zustandsmaschine einer Ausgabe von den Ablaufzähler **480**, **481** empfängt, welche angibt, dass der Zähler **480**, **481** auf Null gesenkt worden ist.

[0053] In einigen Ausführungsformen kann der programmierbare serielle Anschluss **400** ein Statusregister **495** und einen Komparator **490** enthalten. Das Statusregister **495** ist in der Lage Daten zu empfangen, welche den Status jeglicher anderer Komponenten des programmierbaren seriellen Anschlusses **400** angeben, z. B. ein Bit kann angeben, dass ein bestimmtes Register voll ist, leer ist, oder dass ein Überfluss vorliegt, oder kann einen Paritäts-Bitfehler angeben). Die Zustandsmaschinen **420** und **421** können bedingte bzw. an Bedingungen geknüpfte Vorgänge basierend z. B. auf jeglichen der Bits des Statusregisters oder eines Zählwerts der Ablaufzähler **480**, **481** durchführen. Ein Komparator **490** kann enthalten sein, um die Durchführung der bedingten Vorgänge durch die Zustandsmaschinen **420** und **421** zu erleichtern. Der Komparator **490** kann z. B. einen Datenwert in einem ausgewählten Register vergleichen.

[0054] Optional kann eine Interrupt-Verarbeitung durch Interrupt-Verarbeitungsmodule **455** und **456** vorgesehen werden. Bezugnehmend auf Fig. 4B ist ein schematisches Diagramm eines beispielhaften Interrupt-Verarbeitungsmoduls **455** dargestellt. Ein Interrupt-Verarbeitungsmodul ist definiert, konfiguriert und angeordnet, um eine Zustandsmaschine entsprechend von zumindest einem ersten Operand selektiv zu unterbrechen. Das Interrupt-Verarbeitungsmodul enthält ein erstes Interrupt-Auswahlregister **457** zum Steuern von Multiplexern **462** und **463**, welche jeweils einen ersten Operand und einen zweiten Operand an ein Bediener- bzw. Anwendermodul **464** vorsehen. Der erste Operand und der zweite Operand können z. B. ausgewählte Bits des Statusregisters **495** sein, oder entweder der erste Operand, oder der zweite Operand kann ein ausgewählter Datenwert sein. Das Anwendermodul **464** führt mit den ausgewählten Operanden einen ausgewählten Vorgang durch und erzeugt eine Ausgabe (z. B. logisch und/oder logisch oder von ihren Werten). Ein Interrupt-Konfigurationsregister **466** kann zum Steuern des Aktivierens eines Interrupts, der Umkehrung bzw. Inversion einer Eingabe oder einer Ausgabe, und ob der Vergleich nach Empfang eines erfassten Niveaus oder einer Flanke durchgeführt wird, verwendet werden. Ein Interrupt-Konfigurationsregister kann ein Interrupt-Freigabe-Bit zum Steuern eines UND Gates **469** enthalten, um zu bestimmen, ob ein Interrupt an einer Zustandsmaschine vorgesehen werden sollte und dadurch einen Interrupt der Zustandsmaschine verursacht.

[0055] Nach Empfang eines Interrupts, beginnt die Zustandsmaschine ein Programm bzw. eine Routine beginnend an einer Adresse (d. h. ein Interupt-Vektor), welche in dem Interrupt-Adressregister **467** angegeben ist. Eine Zustandsmaschine kann optional, nach Empfang eines Interrupts, eine Return-Adresse in einem Return-Register **468** speichern, um der Zustangsmaschine zu erlauben, zu der Programmzeile zurückzukehren, welche die Zustandsmaschine ausgeführt hat, als der Interrupt aufgetreten ist.

[0056] Fig. 5 ist ein schematisches Diagramm einer beispielhaften Ausführungsform eines Transferregisterschiebemoduls **500** entsprechend von zumindest einigen Aspekten der vorliegenden Erfindung. Das Schieberegistermodul **500** ist hierin definiert, zumindest ein Schieberegister **510** zu enthalten, um auf Kanal **502** empfangene Daten in serieller Reihenfolge zu erstellen. Das Schieberegister **500** enthält optional ein Bitzählermodul **570**, Transferlogik **520** und ein Transferkonfigurationsregister **506**. Ein Bitzähler **572** ist konfiguriert, einen numerischen Zählwert der Bits der seriellen Daten, welche durch das Schieberegistermodul **500** ausgegeben werden, beizubehalten.

[0057] Das Schieberegistermodul **500** empfängt eine Eingabe von einem Pufferspeicher (z. B. Pufferspeicher **420** in **Fig.** 4) auf Kanal **502** oder von einer alternativen Quelle wie z. B. einer durch die Zustandsmaschine **420** (in **Fig.** 4 gezeigt) bezeichnete Speicherstelle auf Kanal **505**, welche einen Satz paralleler Daten-Bits aufweist. Die Quelle der parallelen Eingabe wird durch einen Multiplexer **504** bestimmt. Das Schieberegistermodul **500** sieht eine serielle Ausgabe entsprechend dem parallelen Satz von Bits auf Kanal **503** vor. Die Ausgabe

wird z. B. an den Treiber **380** (in Fig. 3A sichtbar) vorgesehen.

[0058] Das Schieberegister **510** erstellt die ausgewählte parallele Eingabe in serieller Reihenfolge. Es ist ersichtlich, dass das hierin definierte Schieberegister **510** ein herkömmliches Schieberegister oder jegliche andere geeignete Struktur zum Erstellen von Daten in serieller Reihenfolge enthält (z. B. ein mit einem Multiplexer gekoppelter Pufferspeicher, um serielle Bits entsprechend einem Satz paralleler EingabeDaten-Bits selektiv auszugeben).

[0059] Die Transferlogik **520** enthält einen Multiplexer **522**, um ein Stop-Bit zu erzeugen und enthält einen Multiplexer **523**, um das Stop-Bit als Ausgabe vorzusehen. Die Transferlogik **520** enthält zusätzlich einen Paritätsgenerator **524** und einen Multiplexer **525** um das auszugebende Paritäts-Bit vorzusehen. Der Multiplexer **522** wählt ein logisches hoch-Pegel oder ein logisches Niedrig-Pegel aus, wie durch ein durch das Transferkonfigurationsregister **506** angegebenes Steuersignal bestimmt wird. Ein Daten-Bit Multiplexer **526** erlaubt dem Schieberegistermodul **500** die Quelle der Daten, welche über den Kanal **503** ausgegeben werden, zu steuern; z. B. in einem gegebenen Protokoll kann ein gegebenes Bit z. B. angegeben werden, von dem Schieberegister **510** sowie der Zustandsmaschine **420** zu kommen oder kann ausgewählt werden, ein logisches hoch-Pegel oder ein logisches Niedrig-Pegel aufzuweisen. Der Paritäts-Bit-Generator **524** empfängt Daten-Bit Werte, welche von dem Daten-Bit Multiplexer **526** ausgegeben werden und berechnet ein Paritäts-Bit.

[0060] Wie oben beschrieben wird das Schieberegistermodul **500** entsprechend von Aspekten der vorliegenden Erfindung in Reaktion auf den Empfang eines Taktimpulses **572** automatisch vermindert. Ein Paritäts-Bit wird automatisch nach Abschluss der Übertragung der Daten-Bits (falls dies ein Protokoll erfordert) ausgegeben und ein Stop-Zustand wird nach Empfang einer vorbestimmten Anzahl von Taktzyklen eingeleitet, nachdem der Bit-Zähler Null erreicht, ohne die Notwendigkeit, dass die Zustandsmaschine weitere Instruktionen ausführt. Die Anzahl der durch die Zustandsmaschine ausgeführten Instruktionen, um eine Ausgabe entsprechend einem ausgewählten Protokoll zu erreichen ist deshalb relativ gesehen geringer und die Gesamtausführungszeitdauer zum Erreichen einer Ausgabe ist relativ kurz; ein programmierbarer serieller Anschluss ist somit in der Lage entsprechend einer breiten Vielfalt von Protokollen Ausgaben vorzusehen und Eingaben zu akzeptieren.

[0061] Um die obigen automatischen Ausgaben zu erreichen weist das Bit-Zähler-Modul **570** einen Bit-Zähler **574**, einen Komparator **572**, einen Zähl-Analysator **576**, einen Weg-Controller **578** und andere Logik wie nachstehend beschrieben auf. Der Weg-Controller **578** empfängt einen Zählwert und steuert das Schieberegister **510**, sowie die Transferlogik **520**, um auf dem Kanal **503** entsprechend einem ausgewählten Protokoll automatisch eine Ausgabe vorzusehen. Das Konfigurationsregister **506** enthält durch den Controller **450** (in **Fig.** 4 oben gezeigt) vorgesehene Daten, um jede der obigen Komponenten entsprechend dem ausgewählten Protokoll anzuordnen und zu steuern.

[0062] Ein Anfangszählwert wird in Abhängigkeit davon, ob eine Eingabe **506a** von dem Konfigurationsregister **506** angibt, daß das ausgewählte Protokoll erfordert, daß das höchstwertige Bit oder das niedrigstwertige Bit zuerst gesendet wird, durch den Multiplexer **580** an den Bitzähler **574** vorgesehen. Falls das höchstwertige Bit zuerst gesendet werden soll, sieht der Multiplexer **580** einen Anfangszählwert vor, welcher der Datengröße gleicht (d.h. der Gesamtanzahl von Daten-Bits in einem gegebenen Satz von Daten-Bits, welche durch den Kanal **502** vorgesehen werden) und der Bitzähler **574** wird auf Null gesenkt und falls das niedrigswertige Bit zuerst gesendet wird, sieht der Multiplexer **580** einen Anfangszählwert vor, welcher gleich Null ist und der Bitzähler **574** wird auf einen Wert erhöht, welcher der Datengröße gleicht. Der Weg-Controller **578** wählt basierend auf dem Zählwert des Bitzählers **574** für jeden Taktimpuls aus, ob ein Datenwert, ein Paritäts-Bit oder ein Stop-Bit gesendet wird.

[0063] Auf den Empfang eines Taktimpulses **572** hin wird der Zählwert des Bit-Zählers **574** basierend auf dem ausgewählten Protokoll erhöht oder gesenkt. Der Weg-Controller **578** empfängt den Zählwert von dem Bitzähler **574**. Der Weg-Controller **578** vergleicht den Zählwert mit der Eingabe 06a von dem Konfigurationsregister **506**, um zu bestimmen, ob die Multiplexer **523**, **526**, **525** konfiguriert werden sollten, ein Datenbit von dem Schieberegister **510**, ein Paritäts-Bit oder ein Stop-Bit auf dem Ausgabekanal **503** vorzusehen. Auf den Empfang eines ersten Null-Zählwerts hin, steuert der Weg-Controller **578** die Multiplexer **523**, **526**, **525** ein niedrigstwertiges Bit, ein Paritäts-Bit oder ein Stop-Bit auszugeben. Auf den Empfang eines zweiten Null-Zählwerts, steuert der Weg-Controller **528** die Multiplexer **523**, **526**, **525** ein Paritäts-Bit vorzusehen, falls die Eingabe **506a** angibt, dass ein Stop-Bit gesendet werden soll. Der Weg-Controller **578** kann den Zählwert an das Schieberegister **510** vorsehen, welches als ein Zeiger auf das auszugebende Daten-Bit verwendet werden kann; dementsprechend wird jedes der Daten-Bits in dem Schieberegister in Reaktion auf einen Taktimpuls

ausgegeben.

[0064] Der Komparator **572** bestimmt, ob die Anzahl der gesendeten Bits der Datengröße entspricht (z. B. falls das niedrigstwertige Bit zuerst gesendet wurde, bestimmt der Komparator **572** ob der Zählwert der Datengröße gleicht. Die Ausgabe des Komparators **572** wird an den Zählanalysator **576** vorgesehen und der Zählanalysator **576** verwendet die Eingabe **506a** in Kombination mit der Ausgabe des Komparators, um den nächsten Wert des Bit-Zählers **574** zu bestimmen. Bis der Komparator **572** angibt, dass eine Anzahl von Bits, welche der Datengröße entsprechen, gesendet worden sind, wird die Bit-Zahl geeigneterweise erhöht (oder gesenkt). Auf den Empfang einer Ausgabe von dem Komparator **572** hin, welche angibt, daß eine Anzahl von Bits, welche der Datengröße entspricht, gesendet worden ist, bestimmt der Zählanalysator **576**, ob ein Paritäts-Bit notwendig ist (z. B. nachdem der Zähler Null erreicht, wird es dem Zähler erlaubt für einen ersten Impuls bei Null zu verbleiben), oder gibt an, dass ein Stop-Bit notwendig ist (z. B. es wird dem Zähler erlaubt nach dem Erreichen von Null für einen zweiten Impuls bei Null zu verbleiben), und ob ein Umlaufs- bzw. zyklischer Modus (circular mode) eingestellt ist (d.h. der Zähler wird auf den Anfangswert zurückgesetzt, nachdem das Paritäts-Bit und das Stop-Bit gesetzt sind).

[0065] Fig. 6 ist ein schematisches Diagramm eines Beispiels eines Treibers **480**, welcher in dem programmierbaren Anschluß verwendet werden kann. In der dargestellten beispielhaften Ausführungsform sind sechs Eingabe/Ausgabe-Treiberschaltungen vorgesehen: eine Datentransfer-Treiberschaltung **601**, eine Takttransfer-Treiberschaltung, eine Datenempfangs-Treiberschaltung **603**, eine Taktempfangs-Treiberschaltung **604**, und zwei konfigurierbare Eingabe/Ausgabe-Schaltungen **605** und **606**. Die konfigurierbaren Eingabe/Ausgabe-Schaltungen **605** und **606** können z. B. Freigabesignale vorsehen, jeweils eines für die Empfangs- und Sendeanschlüsse, oder können zum Empfang eines Signals verwendet werden, welches als ein Slave-Taktsignal verwendet wird (unten beschrieben).

[0066] Die Schaltungen können beliebige herkömmliche Eingabe/Ausgabe-Treiberschaltungen sein. Die Schaltungen können z. B. die Auswahl einer Quelle/Empfänger (z. B. Leistungsversorgung bei logischer Eins oder Null oder eine Eingabe/Ausgabe von einer Zustandsmaschine, oder eine Ausgabe von einem Schieberegister), eine Auswahl der Polarität eines Ausgabesignals, eine Auswahl eines Hoch-Impedanz-Zustands, eine Erfassung einer Daten-Eingabe/Daten-Ausgabe-Fehlzuordnung und eine Paritäts-Bit-Berechnung erlauben. Ein Schalter **610** kann optional enthalten sein, um eine Zuordnung von Eingaben-Ausgaben zu einem der sechs Ausgabe-Pins **621** bis **626** einer integrierten Schaltung zu erlauben, in welcher der programmierbare Datenanschluß angeordnet ist.

[0067] Fig. 7 ist ein schematisches Diagramm eines Beispiels einer Ausführungsform eines Taktgenerators **700** zur Verwendung mit zumindest einigen programmierbaren seriellen Anschlüssen entsprechend von Aspekten der vorliegenden Erfindung. Der Taktgenerator **700** wählt von einer Vielzahl von Taktquellen ein Master-Taktsignal aus; ein Master-Taktsignal kann z. B. von einem System-Taktgeber **702** ausgewählt werden, welches von einem Mikro-Controller (z. B. der Mikro-Controller **450** in **Fig.** 4) angegeben wird, oder einem Hilfstaktgeber **703**, welcher von einer Quelle von Impulsen eingegeben wird, welche zur Verwendung als ein Master-Taktsignal **704** geeignet sind, ausgewählt, und sieht Ausgabe-Taktsignale vor. Solche Ausgabe-Taktsignale enthalten Ausgabe-Taktsignale **705** und **706** an eine erste und eine zweite Zustandsmaschine (z. B. die Zustandsmaschinen **420** und **421** in **Fig.** 4 oben), Ausgabe-Taktsignale **726** und **736** an erste und zweite Schieberegistermodule (z. B. die Schieberegistermodule **412** und **413**), sowie Ausgabe-Taktsignale **727** und **737** an einem Treiber (z. B. der Treiber **480** in **Fig.** 4).

[0068] Ein Taktgeneratormodul **710** einer Zustandsmaschine empfängt das Master-Taktsignal **704** und sieht die Taktsignal-Ausgaben **705** und **706** an die Zustandsmaschinen (z. B. die Zustandsmaschinen **420** und **421** in **Fig.** 4 oben) auf den Ausgabekanälen **705** bzw. **706** vor, um die Instruktionsausführungen durch die Zustandsmaschinen zu steuern. Die Ausgabe-Taktsignale **705** und **706** können durch einen Zustandsmaschinenteiler **712**, welcher mit den Ausgabekanälen **705** und **706** gekoppelt ist, geteilt werden oder relativ zu dem Master-Taktsignal **704** phasenverzögert werden, oder der Zustandsmaschinenteiler **712** kann (abhängig von dem an den Multiplexer **714** angelegten Steuersignal) kurzgeschlossen bzw. umgangen werden, so daß die Taktsignalausgaben **705** und **706** den Master-Taktsignalen **704** gleichen. Die Gates **708** und **709** können mit den Ausgabekanälen **707** bzw. **708** mit den Gate-Ausgabe-Taktsignalen **705** und **706** gekoppelt sein; die Gates **708** und **709** können z. B. durch Steuersignale von den Zustandsmaschinen **420** bzw. **421** resultierend von der Ausführung einer Warte-Instruktion, einer Verzögerungs-Instruktion oder einer Takt-Instruktion gesteuert werden, was einen Leistungssparmodus, wie nachstehend erläutert, aufruft und werden durch die mit "Steuerung" gekennzeichneten Eingaben **708a** und **709a** angegeben.

[0069] Das Zustandsmaschinen-Taktgeneratormodul **710** enthält ein Konfigurationsregister **707**, um die Taktquelle des Mastertakts **704**, den auf den Teiler **712** anzuwendenden Teilungsfaktor und die Phase des Ausgabe-Taktsignals **705** und **706** relativ zu der Taktquelle zu bestimmen. Die Steuereingaben **708a** und **709a** können ebenfalls durch die Inhalte des Konfigurationsregisters in manchen Ausführungsformen eingestellt werden.

[0070] Ein Takttransfergeneratormodul **720** empfängt von dem Zustandsmaschinen-Taktgeneratormodul **710** eine Signalausgabe **705a**, sowie alternative Takteingaben (z. B. ein asynchrones Slave-Taktsignal **721**, ein logisches Hoch-Signal, und ein logisches Niedrig-Signal) und sieht Taktsignal-Ausgaben **726** und **727** (welche eine Vielzahl von Taktimpulsen aufweisen) an ein Transferschieberegister (**412** in **Fig.** 4) bzw. einen Treiber (z. B. der Treiber **480** in **Fig.** 4) vor. Ein Transferteiler **722** und ein Transferteiler **724** teilen die Nicht-Gate-gesteuerte Ausgabe **705a** des Zustandsmaschinen-Taktgeneratormoduls **710** und sind mit dem Transferschieberegister gekoppelt, um eine Taktsignalausgabe **726** vorzusehen. Der Multiplexer **725** wählt von dem Slave-Taktsignal **721** ein logisches Hoch-Level und ein logisches Niedrig-Level aus; und ein Multiplexer **745** wählt zwischen der Ausgabe des Multiplexers **725** und der Ausgabe des Transferteilers **722** aus, um eine Taktsignalausgabe **726** vorzusehen. Der Multiplexer **723** wählt zwischen der Ausgabe des Transferteilers **724** und der Ausgabe des Multiplexers **745** aus, um die Taktsignalausgabe **727** vorzusehen. Durch eine geeignete Konfiguration des Multiplexers **723** können das Transferschieberegister und der Transfertreiber entsprechend durch das gleiche Taktsignal angesteuert werden.

[0071] Ein Takttransfer-Konfigurationsregister **728** steuert die Teilungsfaktoren der Teiler **722** und **724** sowie die Startpolarität und Stop-Polarität der Ausgaben, ob der Taktgenerator im Leistungssparmodus betrieben wird (d.h. ob der Taktgenerator der Zustandsmaschine während dem Takttransferbetrieb abgeschaltet ist), und ob der Betrieb des Taktgenerators der Zustandsmaschine an dem Ende des Betriebs oder einen Taktzyklus früher gestartet wird (aus Gründen, die nachstehend mit Bezug auf die Fig. 8A und Fig. 8B näher erläutert sind). Ein Takttransfer-Teilerregister **729** steuert den Taktzyklus des Takttransfersignals. Das Takttransfer-Teilerregister kann z. B. ein Hoch-Pegel-Teilungsverhältnis und ein Niedrig-Pegel-Teilungsverhältnis enthalten, um die Anzahl von Zyklen zu bestimmen, in welchen die Taktsignale **726** und **727** einen Hoch-Pegel, sowie einen Niedrig-Pegel aufweisen, und somit der Taktzyklus bestimmt wird. Dem Durchschnitts-Fachmann ist die Implementierung solcher Hoch-Pegel-, sowie Gering-Pegel-Teilungsverhältnissen ersichtlich; deshalb sind weitere Details hierin nicht enthalten.

[0072] Ein Taktempfangs-Generatormodul **730** empfängt eine Signalausgabe **705a** von dem Zustansmaschinen-Taktgeneratormodul **710**, sowie alternative Takteingaben (z. B. ein asynchrones Slave-Taktsiganl **731**, ein logisches Hoch-Pegel-Signal und ein logisches Niedrig-Pegel-Signal) und sieht die Taktsignalausgaben **736** und **737** an ein Empfangsschieberegister (**413** in **Fig.** 4) bzw. einen Treiber (z. B. Treiber **480** in **Fig.** 4) vor. Ein Empfangsteiler **723** und ein Empfangsteiler **734** teilen die nicht-Gate-gesteuerte Ausgabe **705a** des Zustandsmaschinen-Taktgeneratormoduls **710**. Ein Multiplexer **735** wählt zwischen dem Slave-Taktsignal **731** und einem logischen Hoch-Pegel sowie einem logischen Niedrig-Pegel aus, und ein Multiplexer **746** wird zwischen der Ausgabe des Multiplexers **735** und der Ausgabe des Transferteileer **732** aus, um eine Taktsignalausgabe **736** vorzusehen. Ein Multiplexer **733** wählt zwischen einer Ausgabe des Transferteilers **734** und der Ausgabe des Multiplexers **746** aus, um eine Taktsignal-Ausgabe **737** vorzusehen. Durch eine geeignete Konfiguration bzw. einen geeigneten Aufbau des Multiplexers **433** können das Transferschieberegister und der Transfertreiber entsprechend durch das gleiche Taktsignal angesteuert werden.

[0073] Ein Taktempfangs-Konfigurationsregister **738** steuert den Teilungsfaktor der Teiler **732** und **734**, sowie die Startpolarität als auch die Stop-Polarität der Ausgaben. Das Taktempfangs-Teilerregister **737** steuert den Taktzyklus des Takttransfersignals. Das Taktempfangs-Teilerregister **737** kann ein Hoch-Pegel-Teilungsverhältnis und ein Niedrig-Pegel-Teilungsverhältnis enthalten, um die Anzahl der Zyklen zu bestimmen, in welchen die Taktsignale **736** und **737** einen Hoch-Pegel, sowie einen Niedrig-Pegel aufweisen und somit der Taktzyklus bestimmt wird. Dem Durchschnittsfachmann ist die Implementierung solcher Hoch-Pegel- und Niedrig-Pegel-Teilungsverhältnisse ersichtlich; weitere Details sind deshalb hierin nicht enthalten.

[0074] Es ist ersichtlich, daß durch die Verwendung des Taktgenerators **700** ein programmierbarer serieller Anschluß, welcher eine erste Zustandsmaschine und eine zweite Zustandsmaschine aufweist (z. B. der programmierbare serielle Anschluß **400** in **Fig.** 4), in einem Vollduplex-Betrieb betrieben werden kann. Der Taktgenerator **700**, welcher eine erste Zustandsmaschine und eine zweite Zustandsmaschine aufweist, kann alternativ in einem Halb-Duplex-Betrieb betrieben werden, wobei das Takttransfersignal und das Empfangssignal ausgebildet sind, so dass Transfers und ein Empfang in abwechselnden Taktintervallen auftreten.

[0075] Fig. 8A und Fig. 8B sind Zeitablaufdiagramme zweier beispielhafter Taktoptionen. Jedes Zeitablaufdiagramm stellt einen Master-Takt **892**, einen Schieberegister-Steuertakt **804** oder **814** (d.h. ein Takttransfersignal oder ein Taktempfangssignal) und ein entsprechendes Zustandsmaschinen-Taktsignal **806**, **816** dar. Zusätzlich wird eine Anzeige **810** oder **820** des Taktzyklus, während welchem die Taktinstruktionen ausgeführt werden, zusammen mit einer Anzeige **805** oder **815** des Taktzyklus, während welchem die Ausführung der Instruktionen auf die Taktinstruktion folgend (ebenfalls auf die "nächste Instruktion" bezeichnet) gezeigt.

[0076] Fig. 8A stellt eine beispielhafte Taktgeneratorausgabe eines Taktgenerators dar, welcher einen Taktimpuls in einem Standardmodus erzeugt. In einem Standardmodus wird ein Zustandsmaschinen-Taktsignal **806** während dem Zeitraum, wenn das Schieberegister-Taktsteuersignal **804** erzeugt wird, erzeugt. Die der Taktinstruktion folgende Instruktion wird dementsprechend während dem Master-Taktzyklus **805** unmittelbar auf den Taktzyklus **810** folgend ausgeführt, während welchem die Taktinstruktion ausgeführt wird.

[0077] Die beispielhaften Zeitablaufdiagramme entsprechen einem Taktgenerator, welcher ein Konfigurationsregister aufweist, das konfiguriert ist, einen AUS-Zustand **821** von eins und einen Startzustand **823** von Null zu erreichen. Zusätzlich wird durch Auswahl der niedrig-Pegel-und-hoch-Pegel-Teilungsfaktoren ein Tastzyklus von einem Viertel erreicht, so dass zwei von acht Zyklen einen hoch-Pegel aufweisen.

[0078] Fig. 8B stellt eine Taktgeneratorausgabe eines Taktgenerators dar, welcher in einem Leistungssparmodus betrieben wird. In dem Leistungssparmodus wird das Zustandsmaschinen-Taktsignal **816** ausgesetzt, während das Schieberegister-Taktsteuersignal **814** erzeugt wird. Der Leistungssparmodus ermöglicht eine Senkung des Leistungsverbrauchs. Der Leistungssparmodus kann z. B. verwendet werden, falls ein ausgewähltes Protokoll nicht erfordert, dass Instruktionen ausgeführt werden, währende ein Schieberegister-Steuertakt erzeugt wird (d. h. während Bits von einem entsprechenden Schieberegister ausgegeben werden).

[0079] Der Leistungssparmodus wird durch Verwenden der Gates **708**, **709** (in Fig. 7 gezeigt) erreicht, um eine Ausgabe eines Zustandsmaschinen-Taktsignals zu blockieren, während ein entsprechendes Schieberegistermodul eine Ausgabe vorsieht. Während der Ausführung einer Taktinstruktion blockieren die Gates **708** und/oder **709** die Signale **705** bzw. **706**, falls die Steuerregister **728** und **738** in einem Leistungssparmodus konfiguriert sind, und nach Vollendung der geeigneten Anzahl von Zyklen wird das Blockieren des Zustandsmaschinen-Taktsignals durch die Gates **708**, **709** aufgehoben.

[0080] In Fig. 8B sieht das Schieberegistersteuersignal **816** einen Schieberegistersteuertakt **814** vor, welcher zwei Zeiteinheiten bzw. Intervalle (wie durch den Bereich **825** angezeigt) dauert. Das Taktsignal weist einen Tastzyklus von 33 Prozent, einen AUS-Zustand **822** von Null und einen Start-Zustand **824** von eins auf.

[0081] Da das Dekodieren und die Ausführung einer Instruktion zwei Taktzyklen erfordert, tritt die Ausführung der nächsten Instruktion in dem zweiten Taktzyklus **815** nach dem Ende **830** des Schieberegistersteuertaktsignals **814** auf (das Dekodieren der nächsten Instruktion tritt während dem ersten Taktzyklus **831** auf das Ende **830** hin folgend auf). In manchen Ausführungsformen kann ein Gate **708** und/oder **709** dementsprechend gesteuert sein, so dass dieses bzw. dieser das Blockieren des Zustandsmaschinen-Taktsignals **816** einen Zyklus vor dem Ende **830** der Schieberegistertaktausgabe beendet bzw. beenden. Dies ermöglicht das Ausführen der nächsten Instruktion in dem Taktzyklus, welcher unmittelbar auf die Beendigung der Schieberegistertakterzeugung folgt.

[0082] Fig. 9 ist ein Ablaufdiagramm **900** einer beispielhaften Sequenz von Instruktionen für einen programmierbaren seriellen Anschluss entsprechend von einigen Aspekten der vorliegenden Erfindung, um eine Standardausgabe (z. B. eine UART kompatible Ausgabe) zu erreichen. Bei Schritt **905** wartet die Zustandsmaschine auf eine Eingabe von dem Schieberegister, dass das Schieberegister mit Daten gefüllt ist. Bei Schritt **910** lädt die Zustandsmaschine den Treiber mit einem Start-Zustand. Bei Schritt **920** wird der Bit-Zähler initialisiert. Bei Schritt **930** lädt die Zustandsmaschine den Bit-Zähler in das Schieberegister. Das erste Daten-Bit wird gesendet und der logische Wert wird für so viele Taktzyklen wie für das Protokoll nötig sind, bei Schritt **950** gehalten. Bei Schritt **960** wird ein Taktbefehl an den Taktgenerator gesendet, welcher die Anzahl von Impulsen und den Teilungsfaktor für den Teiler definiert. Der Treiber wird schließlich bei Schritt **990** in einen Stop-Zustand versetzt.

[0083] Im Gegensatz zu dem Ablaufdiagramm des Programms für einen herkömmlichen programmierbaren seriellen Anschluss (oben mit Bezug auf Fig. 2 beschrieben), ist es ersichtlich, dass die Ausführungszeit, welche notwendig ist; um eine gegebene Datenausgabe zu erreichen, entsprechend dem hierin gezeigten Verfahren und der Vorrichtung bedeutend verringert wird, da eine verringerte Anzahl von Instruktionen durch die Zu-

standsmaschine ausgeführt werden muss, um eine ausgewählte Ausgabe vorzusehen. In Fig. 2 war z. B. erforderlich, dass die Zustandsmaschine für jedes auszugebende Daten-Bit einen Befehl an das Schieberegister sendet (Schritt **270** in Fig. 2). In Fig. 9 dagegen wird ein einzelner Befehl an den Taktgenerator gesendet (z. B. Taktgenerator **360** in **Fig.** 3), um eine vorbestimmte Anzahl von Impulsen mit einer durch eine Teilungsrate bestimmten Geschwindigkeit zu erzeugen. Da der Taktgenerator mit dem Schieberegister gekoppelt ist, gibt das Schieberegister in Reaktion auf die Taktimpulse. des Taktgenerators Daten-Bits aus, wodurch die Zustandsmaschine von der Erfordernis befreit wird, dem Schieberegister zu befehlen, jede Daten-Bit-Ausgabe vorzusehen. In Fig. 2 senkt die Zustandsmaschine ebenfalls den Bit-Zähler (Schritt **260**), um eine numerische Anzahl von ausgegebenen Daten-Bits beizubehalten. In Fig. 9 ist es dagegen nicht erforderlich, dass die Zustandsmaschine eine Instruktion ausführt, einen Zähler zu senken, da das Schieberegister einen Bit-Zähler aufweist, welcher in Reaktion auf den Empfang eines Taktimpulses von dem Taktgenerator automatisch gesenkt wird. Das Schieberegister ist des weiteren angeordnet, ein Paritäts-Bit auszugeben (falls das Protokoll dies erfordert) und auf den Empfang von Taktzyklen hin, nachdem die Bits gesendet worden sind, in einen Stop-Zustand zu gehen, ohne dass es notwendig ist, dass die Zustandsmaschine weitere Instruktionen ausführt.

[0084] Die folgende Liste von Instruktionen ist ein beispielhafter Instruktionssatz, welcher durch die Zustandsmaschinen **410** und **411** (in **Fig.** 4 gezeigt) ausgeführt werden soll. Die Fig. 10A und Fig. 10B sind Tabellen, welche einen beispielhaften Satz binärer Implementierungen jeder der Instruktionen auf der Liste darstellen. Die Liste enthält eine funktionelle Beschreibung, als auch eine Erklärung der in einer entsprechenden Binärimplementierung enthaltenen Bits für die entsprechenden in den Fig. 10A und Fig. 10B dargestellten Binärimplementierungen **1000**.

[0085] Da die Binärimplementierungen in den Fig. 10A und Fig. 10B zur Implementierung mit der Dekodier- und Ausführungsarchitektur, in welcher ausgewählte Instruktionen parallel ausgeführt werden können (nachstehend mit Bezug auf Fig. 10A beschrieben) geeignet ist, entsprechen einige Instruktionen der nachfolgenden Liste zwei Binärimplementierungen, jeweils eine in Fig. 10A (zur Verwendung an den Bitstellen 0 bis 7) und Fig. 10B (zur Verwendung an den Bitstellen 15 bis 8).

[0086] Bezugnehmend auf die Fig. 10A und Fig. 10B weist jede Instruktion einen Betriebscode **1004** (durch logische Werte von eins und Null in den Binärimplementierungen **1000** angegeben), sowie ein oder mehrere Datenfelder und/oder Adressfelder auf.

[0087] Die nachfolgende Liste von Instruktionen enthält fünf Typen von Instruktionen: Konfigurationsinstruktionen, Betriebssteuerinstruktionen, Ablaufsteuerinstruktionen, Zeitablaufsteuerinstruktionen, Taktsteuerinstruktionen, sowie Bedingungsinstruktionen. Programmierbare serielle Bit-Anschlüsse können unter Verwendung von Zustandsmaschinen implementiert werden, welche ein bekanntes Abruf-, Dekodier-, sowie Ausführungsschema aufweisen. Die aufgezählten Instruktionen können z. B. sequentiell ausgeführt werden. In manchen Ausführungsformen werden die Instruktionen wie nachstehend beschrieben parallel ausgeführt.

Konfigurationsinstruktionen	kurze Beschreibung
Load	lädt Daten in ein angegebenes Register

[0088] Bezugnehmend auf Fig. 10A (**1002**) stellen die d Werte die zu ladenden Werte dar und die i Werte geben eine zu ladende Registeradresse an.

Dual Bit Load	lädt zwei ausgewählte Daten-Bits an den Treiber oder das Treiberkonfigurationsregister.

[0089] Bezugnehmend auf die Fig. 10A und Fig. 10B (**1004a**, **1004b**), geben die i Werte die Bits an, in welche die Werte geladen werden sollen, und die v Werte stellen die zu ladenden Werte dar.

Mask	ermöglicht das Setzen/Zurücksetzen ausgewählter Bits eines ausgewählten Registers.

[0090] Bezugnehmend auf Fig. 10A (**1006**) bilden die m Werte die Maske und die i Werte schildern die Adresse eines auszublendenden Registers dar.

Map	wird in Kombination mit Instruktionen verwendet, welche eine endliche Anzahl von Adress-Bits aufweisen, um die Anzahl von Bits zu erhöhen, die unter Verwendung der Instruktion in einem gegebenen Register zugreifbar sind. Eine ausgewählte Instruktion, welche ein drei-Bit Adressfeld aufweist kann z. B. unter Verwendung der Map Instruktion unter mehr als acht Bits auswählen; die Map Instruktion wählt einen acht-Bit Vektor von innerhalb eines Registers aus, welches mehr als insgesamt acht Bits aufweist und das drei-Bit Adressfeld wählt ein Bit innerhalb des acht-Bit Vektors aus. Map kann z. B. bei der Bedingungs-Ausführungsinstruktion (nachstehend erläutert) verwendet werden, um unter 55 Bits des Statusregisters **495** (in Fig. 4A gezeigt) auszuwählen, trotz der Tatsache, dass in manchen Ausführungsformen das Bedingungs-Ausführungsregister lediglich ein drei-Bit Adressfeld aufweist.

[0091] Bezugnehmend auf Fig. 10B (**1008**) geben die i Werte einen acht-Bit Vektor an.

Extend	ein Extend Befehl wird in Kombination mit einer anderen Instruktion (z. B. einem Dual Bit Load oder einem Trigger) zum Vorsehen eines vergrößerten Adressfelds verwendet.

[0092] Bezugnehmend auf Fig. 10A (**1010**) stellen die i Werte zusätzliche Adressbits dar.

Betriebssteuerinstruktion	
Trigger	ein Trigger Befehl ermöglicht es einer Zustandsmaschine einen angegebenen fest programmierten Ablauf durch Setzen bzw. Einstellen ausgewählter Bits einer Ausgabe einer Zustandsmaschine zu erreichen. Der Ablauf kann direkt oder indirekt (z. B. über ein Register, welches fest programmiert ist, einen angegebenen Ablauf zu erreichen) erreicht werden.

[0093] Bezugnehmend auf die Fig. 10A und Fig. 10B (**1012a**, **1012b**), bezeichnet i die Adresse eines fest programmierten Registers, und die bi Werte geben ein spezifisches Bit innerhalb des Registers an, welches einem bestimmten Ablauf entspricht.

Ablaufsteuerinstruktionen	
Jump Absolute	Springen zu einer absoluten Zeile eines Instruktionssatzes.

[0094] Bezugnehmend auf Fig. 10A (**1014**), stellt der a Wert die Bestimmungsadresse des Sprungs dar.

Jump Short Relative	ein relativer Sprung, welcher auf einen Sprung einer ausgewählten Anzahl von Zeilen begrenzt ist (z. B. 32 Zeilen vor und 16 Zeilen zurück).

[0095] Bezugnehmend auf die Fig. 10A und Fig. 10B (**1016a**, **1016b**), stellen die a Werte die Anzahl der zu überpringenden Programmlinien dar.

Call Absolute	Springen an eine absolute Adresse und Speichern einer Rücksprungadresse in einem designierten Register.

[0096] Bezugnehmend auf Fig. 10A (**1018**), stellen die a Werte die Bestimmungsadresse des Sprungs dar.

Ret	Rücksprung an Adresse, welche in einem designierten Register während eines Call absolut gespeichert wurde, wie als **1020a** und **1020b** in den Figuren 10A bzw. 10B dargestellt ist.
Software Reset	setzt eine Zustandsmaschine zurück. In manchen Ausführungsformen kann der Speicherinhalt bewahrt werden.

[0097] Als **1022a** und **1022b** in den Fig. 10A bzw. Fig. 10B dargestellt.

Loop	führt eine Schleife unter Verwendung einer angegebenen Startadresse und einer angegebenen Endadresse aus. Die Instruktionen zwischen der Startadresse und der Endadresse werden mehrmals, wie durch ein ausgewähltes Register angegeben, ausgeführt.

[0098] Bezugnehmend auf Fig. 10A (**1024**), gibt der o1 Wert eine Startadresse und der o2 Wert eine Endadresse für eine Schleife an. Die Anzahl der Wiederholungen ist durch ein getrenntes Schleifenzählerregister festgelegt.

Null	ein „Auffüller", welcher verwendet wird, um sechzehn-Bit Instruktionen in Ausführungsformen, welche sechzehn-Bit Abrufe aufweisen, ausrichtet. Resultiert in keiner Ausführung. (Aspekte der Null-Instruktion sind nachstehend mit Bezug auf Fig. 10A ausführlicher beschrieben).

[0099] Als **1026** in Fig. 10B dargestellt.

Zeitablaufsteuerinstruktionen	
Delay	verzögert die Ausführung einer nächsten Instruktion für eine ausgewählte Anzahl von Taktzyklen. Wie oben mit Bezug auf Fig. 4A beschrieben ist, kann eine Verzögerung unter Verwendung von den Event-Zählern **480**, **481** ausgeführt werden.

[0100] Bezugnehmend auf die Fig. 10A und Fig. 10B (**1028a**, **1028b**), stellen die d Werte eine Gesamtverzögerungslänge bzw. Dauer dar.

Long Delay	Long Delay ist ein Vorgang, der Delay gleicht, außer dass eine Dauer der Verzögerung unter Verwendung eines Zeigers zu einem Register anstatt innerhalb der Instruktion selbst ausgewählt wird. Es kann dementsprechend eine längere Verzögerungsdauer angegeben werden.

[0101] Bezugnehmend auf Fig. 10A und Fig. 10B (**1030a**, **1030b**), stellen die i Werte einen Zeiger zu einem Register dar, welches eine Gesamtverzögerungslänge aufweist.

Wait	wartet bis eine angegebene Bedingung war ist (z. B. eine unter Verwendung des Statusregisters **495** und des Komparators **490** (siehe Fig. 4A) angegebene Bedingung). Der Takt der Zustandsmaschine kann während der Ausführung einer Wait Instruktion unter Verwendung der Gates **708**, **709** gate gesteuert werden.

[0102] Bezugnehmend auf Fig. 10A (**1032**), geben die c1 Werte eine erste zu testende Bedingung an (z. B. eine Flankenerfassung, Pufferspeicher voll oder Pufferspeicher leer), und die c2 Werte stellen eine zweite Bedingung dar. Die v1 und v2 Werte sind Werte, welche auf die erste Bedingung bzw. die zweite Bedingung getestet werden. Die mm Werte wählen die für einen oder beide der ersten Bedingung und der zweiten Bedingung zu machende Wertung aus (z. B. eine Bewertung kann sowohl Bedingung c1 gleich v1 und c2 gleich v2 enthalten).

Taktsteuerinstruktionen

Clock	weist einen ausgewählten Taktteiler an, eine ausgewählte Anzahl von Taktzyklen in einem Standard- oder Leistungssparmodus in Abhängigkeit von dem Taktkonfigurationsregister auszugeben.

[0103] Bezugnehmend auf Fig. 10A (**1034**), wählen die cd Werte eine Taktauswahl aus, und die d Werte stellen eine Anzahl von auszugebenden Impulsen dar.

Logische Instruktionen

Conditional Execution	die Ausführung einer angegebenen Instruktion wird an eine ausgewählte Bedingung geknüpft.

[0104] Bezugnehmend auf Fig. 10A (**1040**), stellen die i Werte das Bit innerhalb eines acht-Bit Vektors des Statusregister dar, welches den Operand bildet, und der v Wert stellt die Bedingung (1 = wahr und 0 = falsch) dar. Das zweite Byte, welches typischerweise als das erste Byte einer sechzehn-Byte Instruktion verwendet wird, ist die auszuführende Instruktion, falls die Bedingung wahr ist.

Compare Data	Vergleich von Daten in einem angegebenen Register mit einem angegebenen Datenwert. Der Vergleich kann einen Vergleich basierend auf zumindest den folgenden Operatoren enthalten: kleiner als, größer als, gleich, etc.

[0105] Bezugnehmend auf Fig. 10A (**1036**), geben die d Werte das Register an, welches als der erste Operand agiert. Die d Werte geben die Daten an, welche den zweiten Operand bilden, und die tc Werte stellen den Vergleichstyp dar (Daten = Register, Daten zwei Register, Daten neun Register).

Compare Registers	Vergleich von Daten in einem angegebenen Register mit Daten in einem anderen angegebenen Register.

[0106] Bezugnehmend auf Fig. 10A (**1038**), geben die i Werte das Registerpaar an, welches den ersten Operand und den zweiten Operand bildet, und die tc Werte stellen den Vergleichstyp dar (Daten = Register, Daten zwei Register, Daten neuen Register).

[0107] Konfigurationsregisterinhalte und Instruktionen zum Implementieren eines Protokolls können direkt manuell erzeugt werden oder von einer Eingabe höherer Ebene unter Verwendung von geeigneten Werkzeugen übersetzt werden. Eine Instruktionssequenz zur Verwendung für programmierbare serielle Anschlüsse entsprechend von Aspekten der vorliegenden Erfindung kann in einer geeigneten Speicherorganisation bzw. Ordnung angeordnet werden. Fig. 11A ist eine schematische Darstellung einer geeigneten Speicherordnung **1100**, welche zur Verwendung mit der vorliegenden Erfindung geeignet ist; die Speicherordnung **1100** ermöglicht die Verwendung eines Instruktionssatzes einschließlich einer Kombination von acht-Bit und sechzehn-Bit Instruktionen (wie oben mit Bezug auf die Fig. 10A und Fig. 10B erläutert), und ermöglicht, dass ein sechzehn-Bit Abruf bei jedem Taktzyklus auftritt, ohne dass partielle Instruktionen während einem gegebenen Abruf

abgerufen werden. Die beispielhafte Speicherordnung **1100** weist die Speicherzeilen **1102**, **1104**, **1106**, sowie **1108** auf; jede Speicherzeile ist in sechzehn-Bit Segmente unterteilt.

[0108] In manchen Ausführungsformen ist eine erste acht-Bit Instruktion **1102a** in den acht-Bit Speichersegmenten **1120** angeordnet, welche bei der Stelle **15** beginnen, wobei das höchstwertige Bit an Stelle **15** angeordnet ist, und eine zweite acht-Bit Instruktion **1102b** ist in einem nächsten acht-Bit Speichersegment **1135** (d. h. beginnend bei Stelle **7** der Zeile **1102**) angeordnet. Die zweite Speicherzeile **1104** wird von einer sechzehn-Bit Instruktion **1104a** besetzt.

[0109] Ähnlich der ersten Zeile **1102**, weist die Zeile **1104** eine erste acht-Bit Instruktion **1106a** auf, welche in den acht-Bit Speichersegmenten **1130** beginnend an Stelle **15** angeordnet ist, wobei das höchstwertige Bit an Stelle **15** angeordnet ist. Da die nächste Instruktion jedoch eine sechzehn-Bit Instruktion **1108a** ist, wird die zweite Instruktion **1106b** in Zeile **1106** ausgewählt, eine Null-Instruktion (oben beschrieben) zu sein, um sechzehn-Bit Abrufe zu vermeiden, welche partielle Instruktionen enthalten (d. h. die Hälte einer sechzehn-Bit Instruktion). Ein Übersetzer zur Verwendung in einer solchen Architektur fügt entsprechend vorzugsweise eine acht-Bit Null-Instruktion **1106b** in die acht-Bit Speichersegmente **1135** der Zeile **1106** ein. Die Null-Instruktion ist vorzugsweise eine nicht-ausgeführte Instruktion (d. h. diese ist einfach eine Platzhalterinstruktion).

[0110] Fig. 11B ist eine schematische Darstellung einer beispielhaften Zustandsmaschinen Dekodier- und Ausführungsarchitektur **1150**, welche zur Verwendung in der Speicherordnung **1100** der Fig. 11A geeignet ist. Die Dekodier- und Ausführungsarchitektur **1150** enthält einen Vor-Dekoder **1160** und zwei Dekoder **1170**, sowie **1175**.

[0111] Wie oben erwähnt ruft die zwei-Stufen Dekodier- und Ausführungsarchitektur **1150** sechzehn Instruktionsbits wie oben beschrieben bei jedem Taktzyklus ab. Der Vor-Dekoder **1160** prüft unter Verwendung verwandter Techniken die Vorgangs- bzw. Ablaufcodes von Instruktionen, welche einer Speicherzeile **1102**, **1104** sowie **1106** (in Fig. 11A oben sichtbar) entsprechen. Der Vor-Dekoder **1160** bestimmt, ob die sechzehn Bits eine sechzehn-Bit Instruktion, zwei acht-Bit Instruktionen, welche parallel ausgeführt werden sollen, oder zwei acht-Bit Instruktionen, welche serielle bzw. in Reihe ausgeführt werden sollen, aufweist.

[0112] Die Präsenz einer in Fig. 10B dargestellten Instruktion, welche an einer Speicherstelle **1130** angeordnet ist (z. B. wie durch Identifizieren ihres Vorgangscodes bestimmt), gibt an, dass die acht-Bit Instruktionen an den Stellen **1130** und **1135** parallel ausgeführt werden sollen. Die Präsenz von anderen Instruktionen gibt die Präsenz einer sechzehn-Bit Instruktion oder zwei acht-Bit Instruktionen an, welche parallel ausgeführt werden sollen.

[0113] Im Falle, dass eine bestimmte Speicherzeile **1102**, **1104**, **1106** (in Fig. 11A sichtbar) eine sechzehn-Bit Instruktion enthält, werden sechzehn Bits an den Dekoder **1170** vorgesehen; im Falle dass eine bestimmte Speicherzeile **1102**, **1104**, **1106** zwei acht-Bit Instruktionen enthält, welche seriell ausgeführt werden sollen, werden acht Bits, die der ersten Instruktion entsprechen, in einem ersten Zyklus und acht Bits in dem nächsten Zyklus vorgesehen, so dass die erste Instruktion in einem ersten Taktzyklus und die zweite Instruktion in dem folgenden Taktzyklus ausgeführt wird; und im Falle, dass einen bestimmte Speicherzeile **1102**, **1104**, **1106** zwei acht-Bit Instruktionen enthält, welche parallel ausgeführt werden sollen, empfängt der Dekoder **1170** die erste Instruktion und der Dekoder **1175** die zweite Instruktion in einem ersten Taktzyklus.

[0114] Da somit die erfinderischen Konzepte einer Anzahl beispielhafter Auführungsformen beschrieben worden sind, ist es dem Fachmann ersichtlich, dass die Erfindung auf verschiedene Arten implementiert werden kann, und dass Modifikationen und Verbesserungen sich solchen Personen leicht erschließen. Die gegebenen Beispiele sind somit nicht als begrenzend beabsichtigt. Die Erfindung ist lediglich, wie erforderlich, durch die folgenden Ansprüche und deren Äquivalente begrenzt. Es ist ebenfalls ersichtlich, dass die Verwendung der Begriffe (enthalten), oder (aufweisen) die entsprechend nachstehend aufgeführten Gegenstände und deren Äquivalente als auch zusätzliche Gegenstände, welche vor, nach, oder zwischen den aufgeführten Gegenständen genannt sind, umfassen.

Patentansprüche

1. Programmierbarer serieller Anschluß, aufweisend:
ein erstes Schieberegistermodul (**312**) einschließlich einem Schieberegister, aufweisend einen Eingabekanal zum Empfangen einer parallelen Eingabe einer ersten Vielzahl von Bits und einen ersten Ausgabekanal zum Vorsehen einer seriellen Ausgabe einer zweiten Vielzahl von Bits, und ein weiteres Register zum Steuern des

Schieberegistermoduls;
einen Taktgenerator; und
eine erste Zustandsmaschine (**320**),
dadurch gekennzeichnet, daß der Taktgenerator (**360**) ein erstes Taktsignal an das erste Schieberegistermodul vorsieht, und der Taktgenerator, sowie das erste Schieberegistermodul auf Instruktionen von der Zustandsmaschine reagiert, so daß die Zustandsmaschine zumindest eine Instruktion ausgeben kann, um eine Ausgabe der zweiten Vielzahl von Bits von dem Schieberegister ohne weitere Steuerung von der Zustandsmaschine zu verursachen.

2. Programmierbarer serieller Anschluß nach Anspruch 1, wobei der Taktgenerator einen zweiten Ausgabekanal an die erste Zustandsmaschine aufweist, und der Kanal ein zweites Taktsignal an die erste Zustandsmaschine vorsieht, wobei das Signal eine zweite Vielzahl von Taktimpulsen aufweist.

3. Programmierbarer serieller Anschluß nach den Ansprüchen 1 oder 2, wobei der Taktgenerator mit einer ersten Zustandsmaschine durch ein Gate gekoppelt ist, welches durch zumindest eine der Instruktionen steuerbar ist.

4. Programmierbarer serieller Anschluß nach Anspruch 3, wobei das Gate steuerbar ist, um das zweite Taktsignal zu blockieren, während das Schieberegister die serielle Ausgabe vorsieht.

5. Programmierbarer serieller Anschluß nach Anspruch 2, ferner einen Teiler aufweisend, welcher mit dem zweiten Ausgabekanal gekoppelt ist, um das zweite Taktsignal zu empfangen, und mit dem ersten Ausgabekanal gekoppelt ist, um das erste Taktsignal vorzusehen, wobei das zweite Taktsignal geteilt wird, um das erste Taktsignal zu bilden.

6. Programmierbarer serieller Anschluß nach einem der vorhergehenden Ansprüche, wobei die Zustandsmaschine konfiguriert und angeordnet ist, um Instruktionen entsprechend einer Vielzahl von seriellen Kommunikationsprotokollen vorzusehen.

7. Programmierbarer serieller Anschluß nach Anspruch 2 oder einem der Ansprüche 3 bis 6 sofern dieser von Anspruch 2 abhängig ist, wobei jedes der zweiten Vielzahl von Bits in Reaktion auf einen Taktimpuls der zweiten Vielzahl von Impulsen ausgegeben wird.

8. Programmierbarer serieller Anschluß nach Anspruch 2 oder einem der Ansprüche 3 bis 7, sofern dieser von Anspruch 2 abhängig ist, wobei das Schieberegistermodul ferner einen Bit-Zähler aufweist, und der Bit-Zähler konfiguriert ist, einen numerischen Zählwert der seriellen Ausgabe der zweiten Vielzahl von Bits beizubehalten.

9. Programmierbarer serieller Anschluß nach Anspruch 8, wobei der Bit-Zähler in Reaktion auf einen Taktimpuls der – zweiten Vielzahl von Impulsen vermindert wird.

10. Programmierbarer serieller Anschluß nach Anspruch 9, wobei jedes der zweiten Vielzahl von Bits entweder ein Daten-Bit, ein Paritäts-Bit, oder ein Stop-Bit ist.

11. Programmierbarer serieller Anschluß nach Anspruch 10, wobei jedes der zweiten Vielzahl von Bits basierend auf dem Bit-Zählwert ausgewählt wird.

12. Programmierbarer serieller Anschluß nach einem der vorhergehenden Ansprüche, wobei das erste Schieberegistermodul ferner einen Paritäts-Generator aufweist.

13. Programmierbarer serieller Anschluß nach Anspruch 12, wobei das erste Schieberegistermodul ein Paritäts-Bit von dem Paritäts-Generator in Reaktion auf einen Taktimpuls der zweiten Vielzahl von Impulsen ausgibt.

14. Programmierbarer serieller Anschluß nach einem der vorhergehenden Ansprüche, ferner einen programmierbaren Treiber aufweisend, welcher mit dem ersten Ausgabekanal gekoppelt ist, um die elektrischen Parameter der seriellen Ausgabe zu steuern.

15. Programmierbarer serieller Anschluß nach einem der vorhergehenden Ansprüche, ferner ein Interrupt-Verarbeitungsmodul aufweisend, welches mit der ersten Zustandsmaschine gekoppelt ist, um einen Inter-

rupt der Zustandsmaschine zu verursachen.

16. Programmierbarer serieller Anschluß nach einem der vorhergehenden Ansprüche, ferner ein zweites Schieberegistermodul aufweisend, wobei der Taktgenerator mit dem zweiten Schieberegistermodul gekoppelt ist, um ein zweites Taktsignal vorzusehen, welches eine zweite Vielzahl von Taktimpulsen aufweist, und das zweite Schieberegistermodul in Reaktion auf das zweite Taktsignal eine zweite serielle Ausgabe vorsieht.

17. Verfahren zum Steuern eines Schieberegistermoduls, aufweisend ein erstes Schieberegister, welches eine erste Vielzahl von Bits enthält, um eine serielle Ausgabe entsprechend einem ausgewählten einer Vielzahl von seriellen Kommunikationsprotokollen vorzusehen, wobei das erste Schieberegistermodul mit einem Taktgenerator und einer Zustandsmaschine gekoppelt ist, und das Verfahren aufweist:
Auswählen einer Instruktionssequenz von einer Vielzahl von Instruktionssequenzen entsprechend dem ausgewählten Protokoll, welche durch die Zustandsmaschine ausgeführt werden soll, wobei jede der Instruktionssequenzen einem Protokoll entspricht;
Steuern des Taktgenerators entsprechend einer Instruktion der ausgewählten Instruktionssequenz, um eine erste Vielzahl von Taktimpulsen an das Schieberegistermodul vorzusehen, und
Ausgeben einer zweiten Vielzahl von Bits, welche der ersten Vielzahl von Bits entsprechen, in Reaktion auf die Taktimpulse und Ausführen der Instruktionssequenz ohne weitere Steuerung durch die Zustandsmaschine

18. Verfahren zum Steuern eines Schieberegistermoduls nach Anspruch 17, ferner einen Vorgang des Beibehaltens eines Zählwerts der zweiten Vielzahl von Bits aufweisend.

19. Verfahren zum Steuern eines Schieberegistermoduls nach Anspruch 18, ferner einen Vorgang des selektiven Ausgebens eines Paritäts-Bits in Reaktion auf den Zählwert aufweisend.

20. Verfahren zum Steuern eines Schieberegisters nach den Ansprüchen 17 bis 19, ferner einen Vorgang des Steuerns des Taktgenerators zum Vorsehen einer zweiten Vielzahl von Taktimpulsen zum Steuern der Ausführung der Instruktionssequenz durch die Zustandsmaschine aufweisend.

21. Verfahren zum Steuern eines Schieberegisters nach Anspruch 20, ferner einen Vorgang des Blockierens der zweiten Vielzahl von Taktimpulsen aufweisend, wobei das Ausführen der Vielzahl von Instruktionen während dem Ausgeben der zweiten Vielzahl von Bits gestoppt wird.

22. Verfahren zum Steuern eines Schieberegistermoduls nach einem der Ansprüche 17 bis 21, ferner den Vorgang des Steuerns des Taktgenerators zum Vorsehen einer dritten Vielzahl von Taktimpulsen an ein zweites Schieberegistermodul, welches eine dritte Vielzahl von Bits enthält, und das Ausgeben einer vierten Vielzahl von Bits in Reaktion auf die Taktimpulse entsprechend der dritten Vielzahl von Bits aufweisend.

23. Verfahren zum Steuern eines Schieberegisters nach Anspruch 17, wobei das Steuern des Taktgenerators entsprechend einer Instruktion das Angeben der Anzahl von Impulsen und der Taktrate enthält.

Es folgen 15 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1
STAND DER TECHNIK

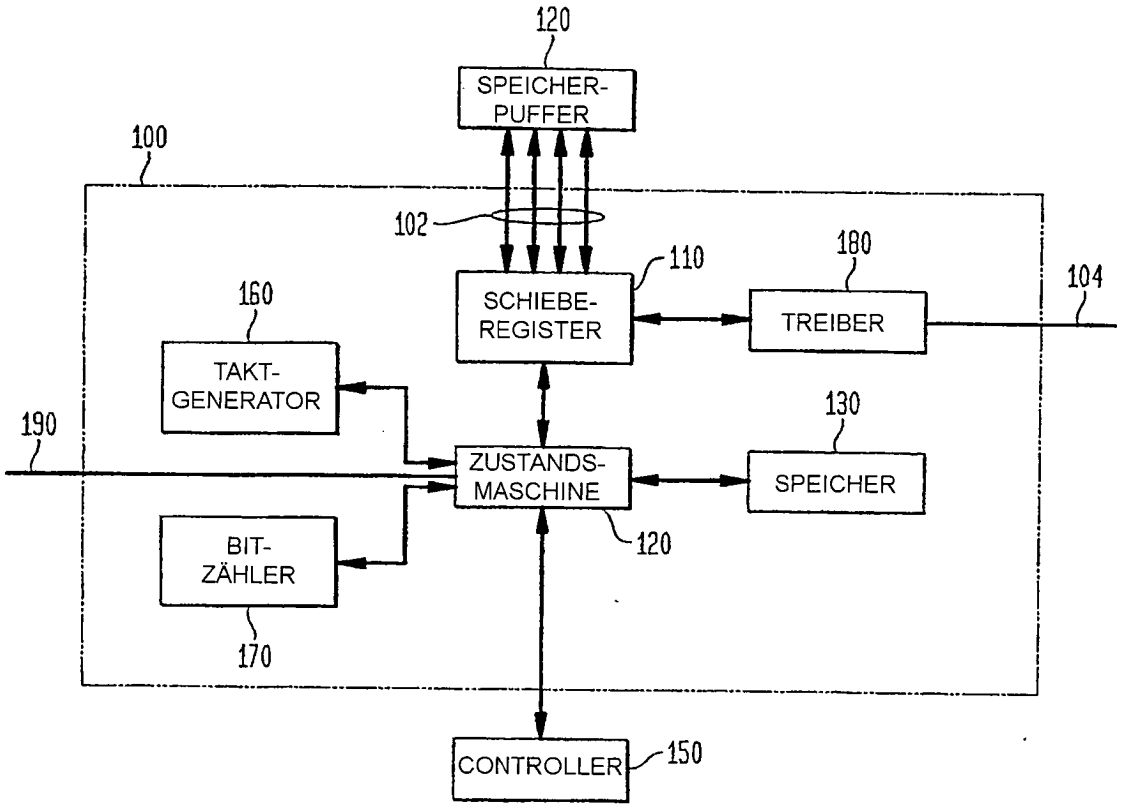


FIG. 2
STAND DER TECHNIK

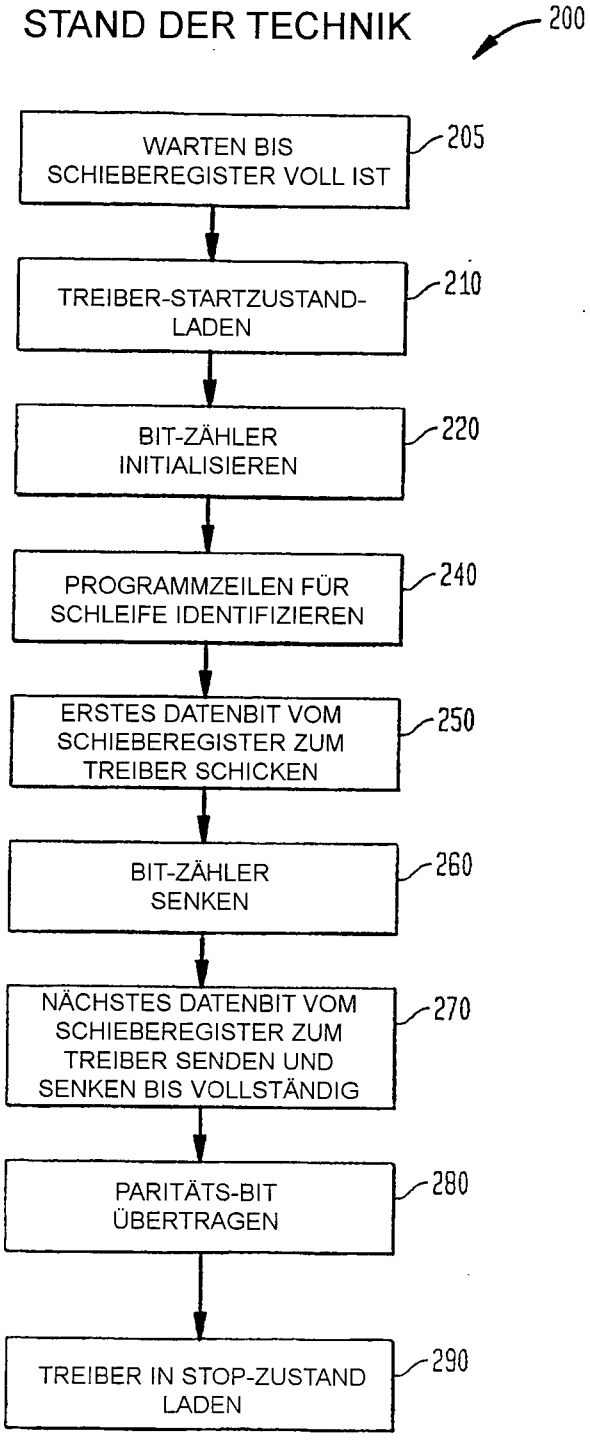


FIG. 3A

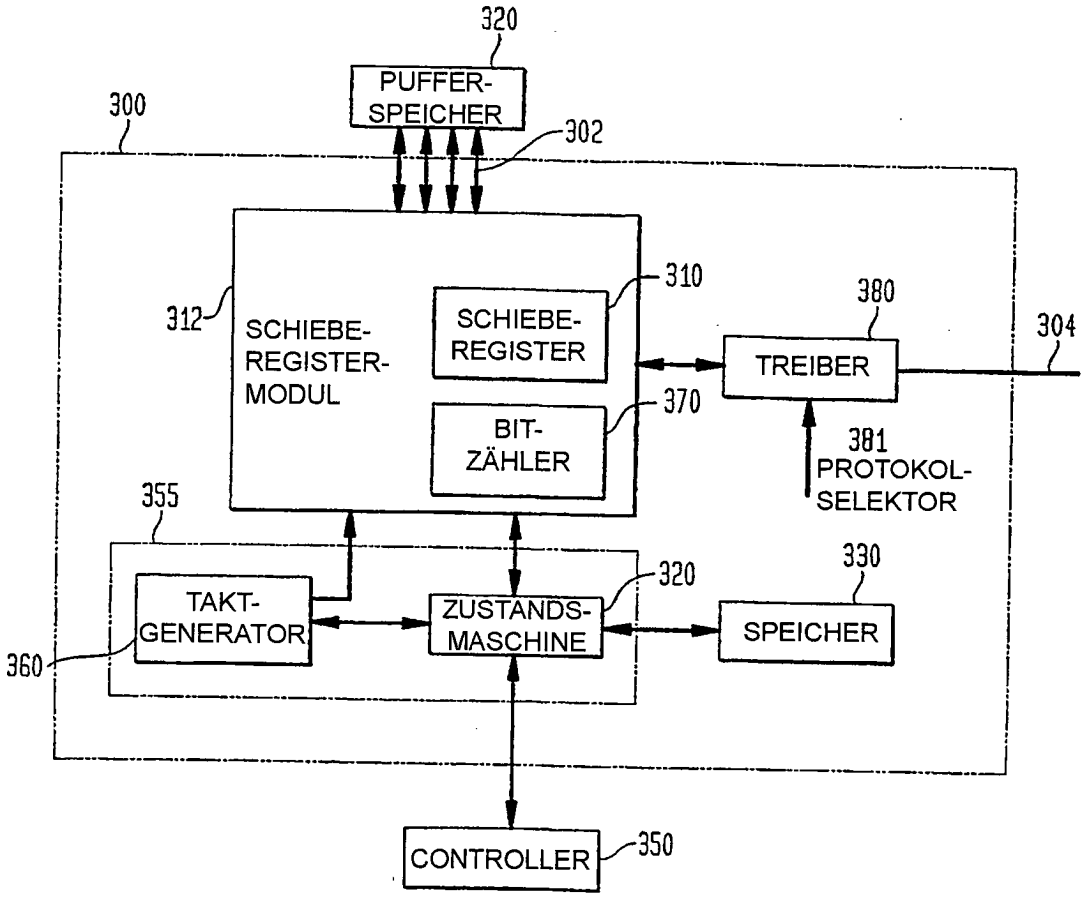
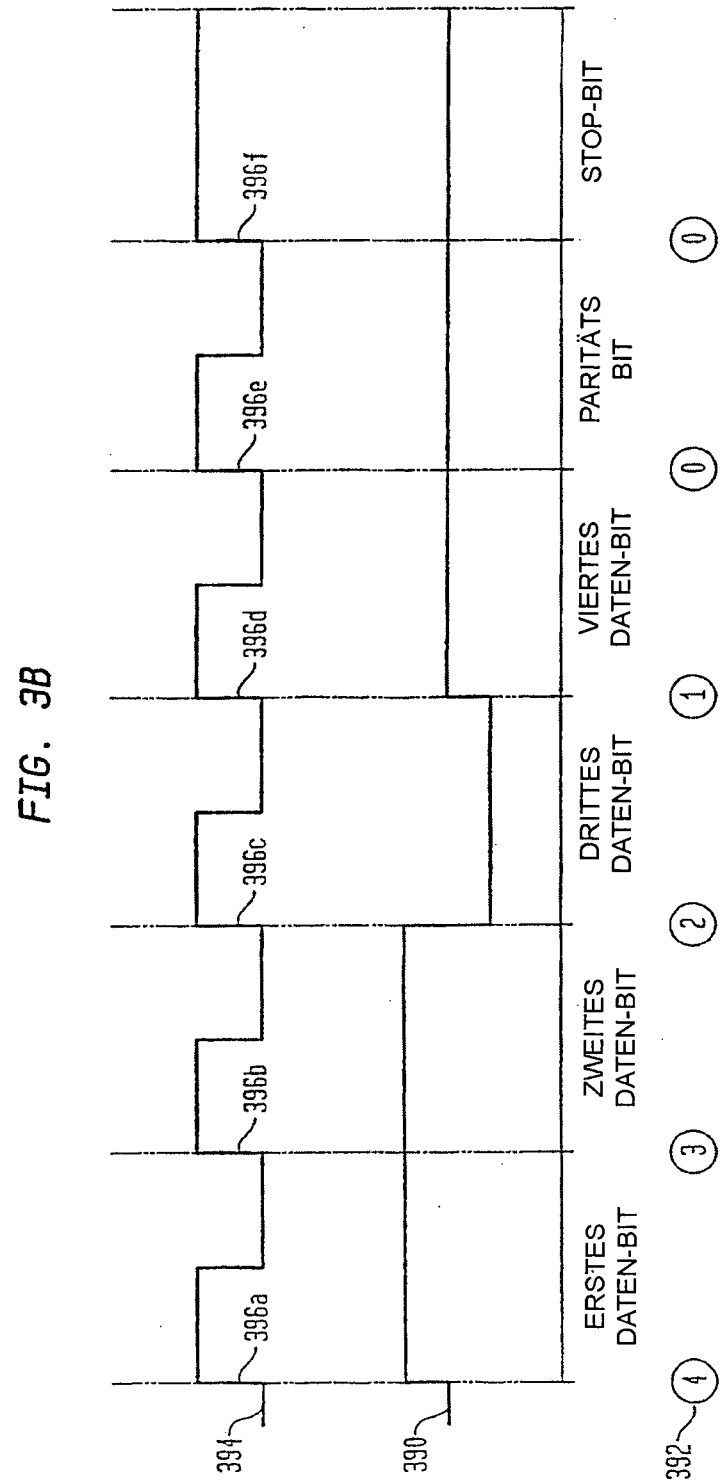

320
PUFFER-SPEICHER
300
302
312
SCHIEBE-REGISTER-MODUL
SCHIEBE-REGISTER
310
380
304
TREIBER
BIT-ZÄHLER
370
381
PROTOKOL-SELEKTOR
355
330
320
TAKT-GENERATOR
ZUSTANDS-MASCHINE
SPEICHER
360
CONTROLLER
350

FIG. 3B


394
390
392
396a
396b
396c
396d
396e
396f
ERSTES DATEN-BIT
ZWEITES DATEN-BIT
DRITTES DATEN-BIT
VIERTES DATEN-BIT
PARITÄTS BIT
STOP-BIT
4
3
2
1
0
0

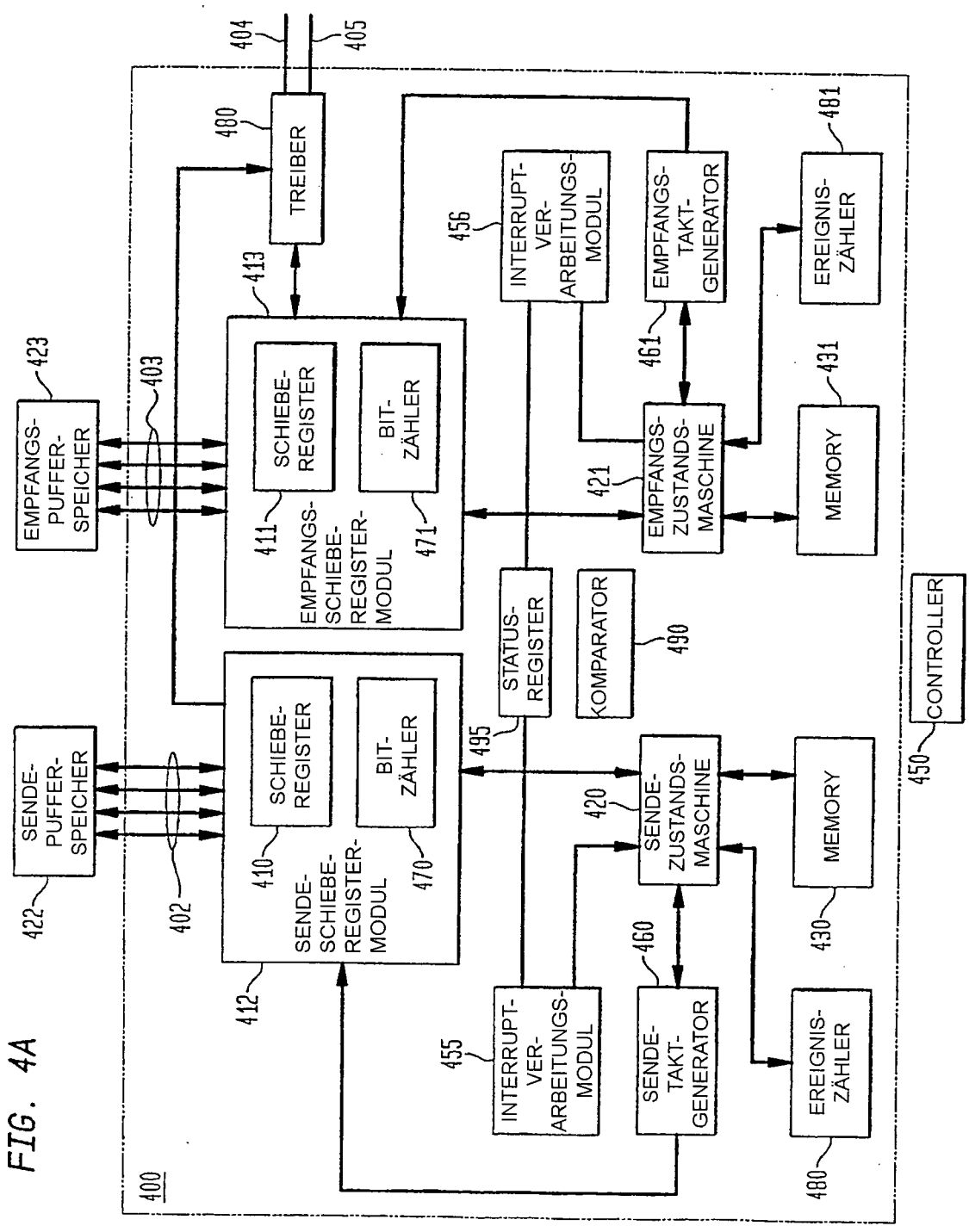

400
404
405
480
TREIBER
413
SCHIEBE-REGISTER
BIT-ZÄHLER
411
471
EMPFANGS-SCHIEBE-REGISTER-MODUL
456
INTERRUPT-VER-ARBEITUNGS-MODUL
EMPFANGS-TAKT-GENERATOR
461
EREIGNIS-ZÄHLER
481
423
EMPFANGS-PUFFER-SPEICHER
403
421
EMPFANGS-ZUSTANDS-MASCHINE
MEMORY
431
STATUS-REGISTER
KOMPARATOR
490
CONTROLLER
450
495
SENDE-PUFFER-SPEICHER
422
402
410
470
SENDE-SCHIEBE-REGISTER-MODUL
412
420
SENDE-ZUSTANDS-MASCHINE
MEMORY
430
455
INTERRUPT-VER-ARBEITUNGS-MODUL
460
SENDE-TAKT-GENERATOR
EREIGNIS-ZÄHLER
480

FIG. 4A

FIG. 4B

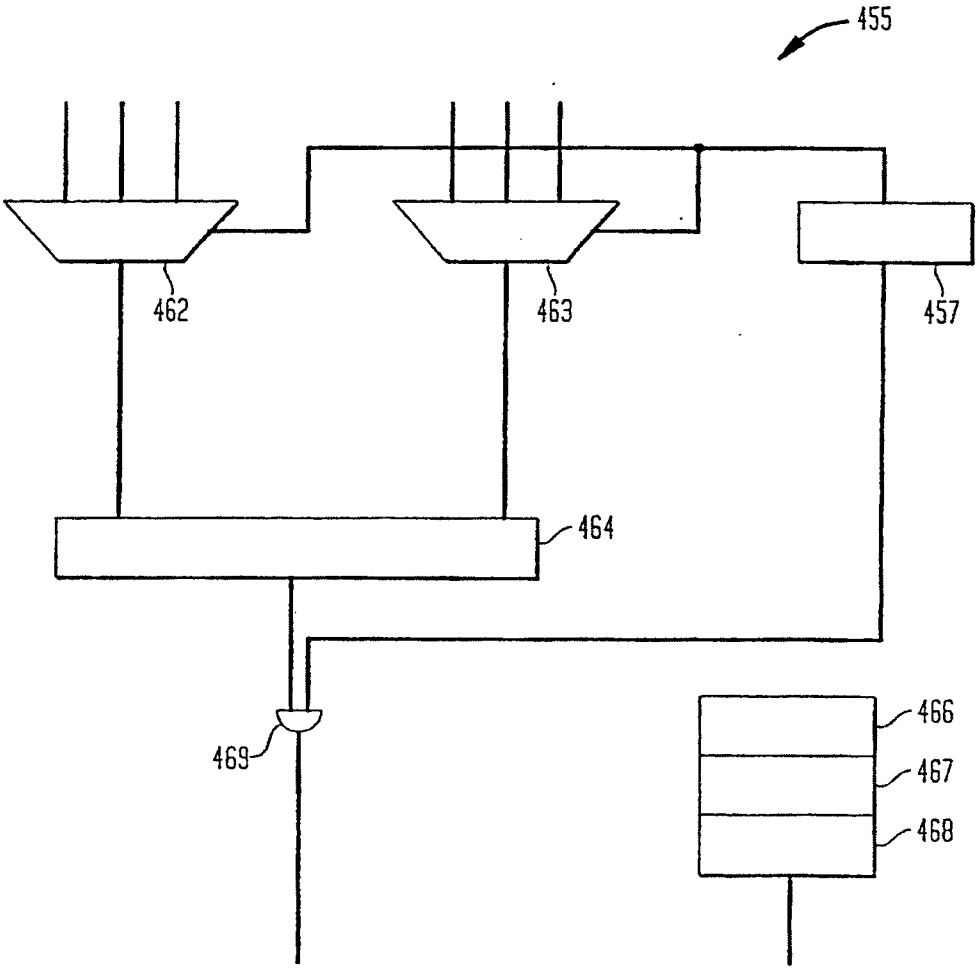


FIG. 5

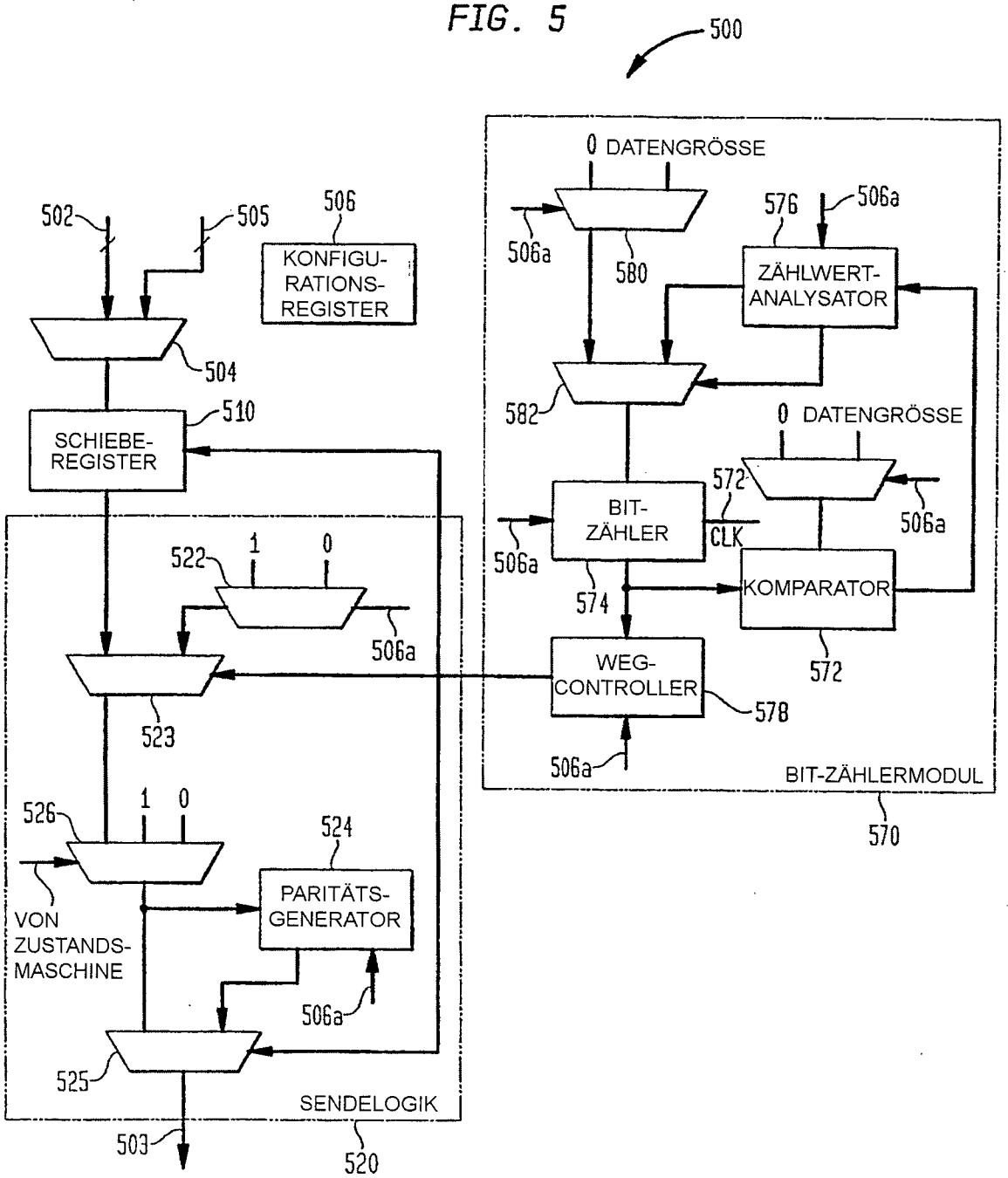

500
0 DATENGRÖSSE
506a
576
506a
506a
580
ZÄHLWERT-
ANALYSATOR
582
0 DATENGRÖSSE
572
506a
BIT-
ZÄHLER
CLK
506a
KOMPARATOR
574
572
WEG-
CONTROLLER
578
506a
BIT-ZÄHLERMODUL
570
502
505
506
KONFIGU-
RATIONS-
REGISTER
504
SCHIEBE-
REGISTER
510
522
1
0
506a
523
526
1
0
524
PARITÄTS-
GENERATOR
VON
ZUSTANDS-
MASCHINE
506a
525
SENDELOGIK
503
520

FIG. 6

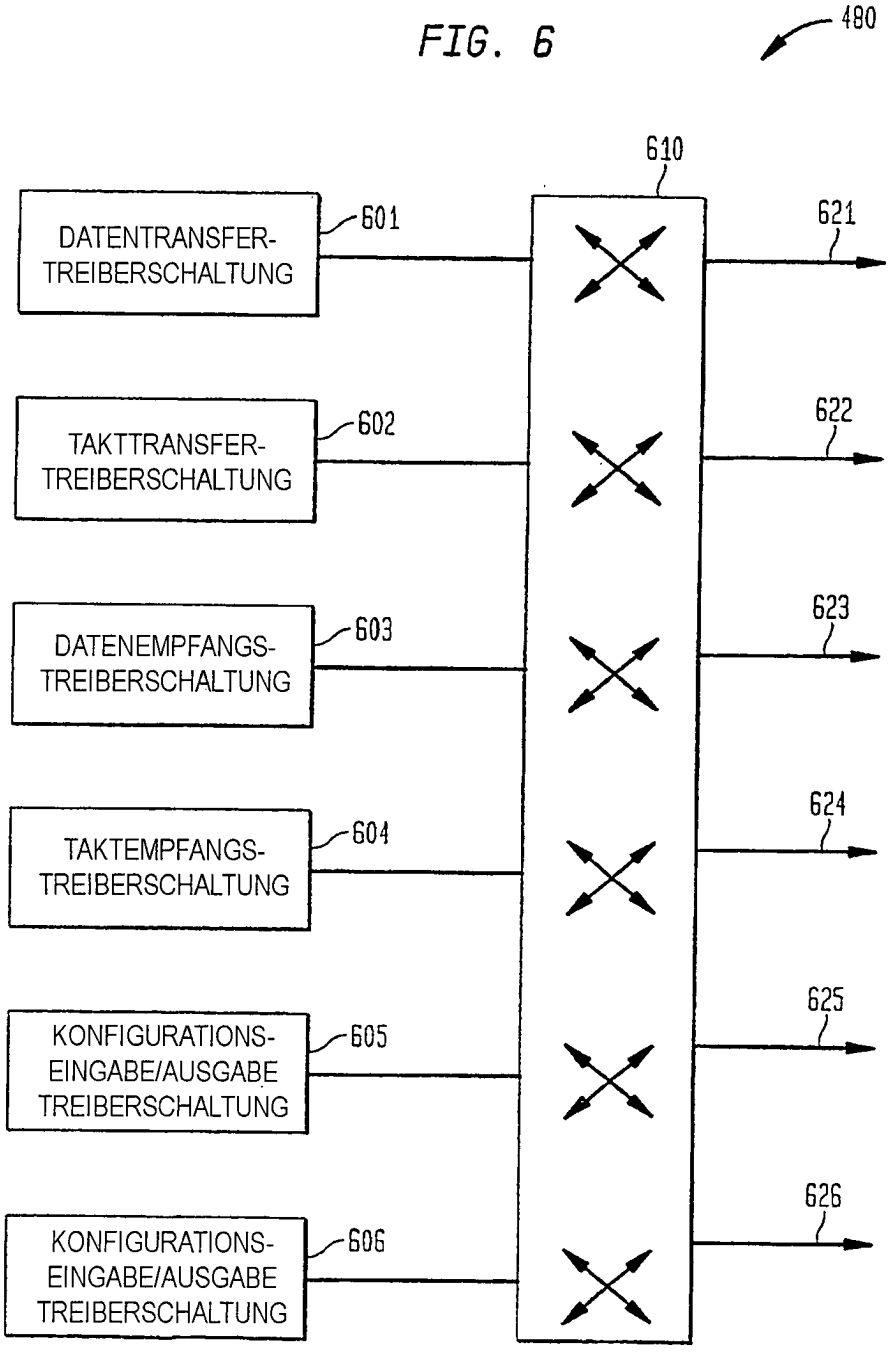

480
610
DATENTRANSFER-TREIBERSCHALTUNG
601
621
TAKTTRANSFER-TREIBERSCHALTUNG
602
622
DATENEMPFANGS-TREIBERSCHALTUNG
603
623
TAKTEMPFANGS-TREIBERSCHALTUNG
604
624
KONFIGURATIONS-EINGABE/AUSGABE TREIBERSCHALTUNG
605
625
KONFIGURATIONS-EINGABE/AUSGABE TREIBERSCHALTUNG
606
626

FIG. 7

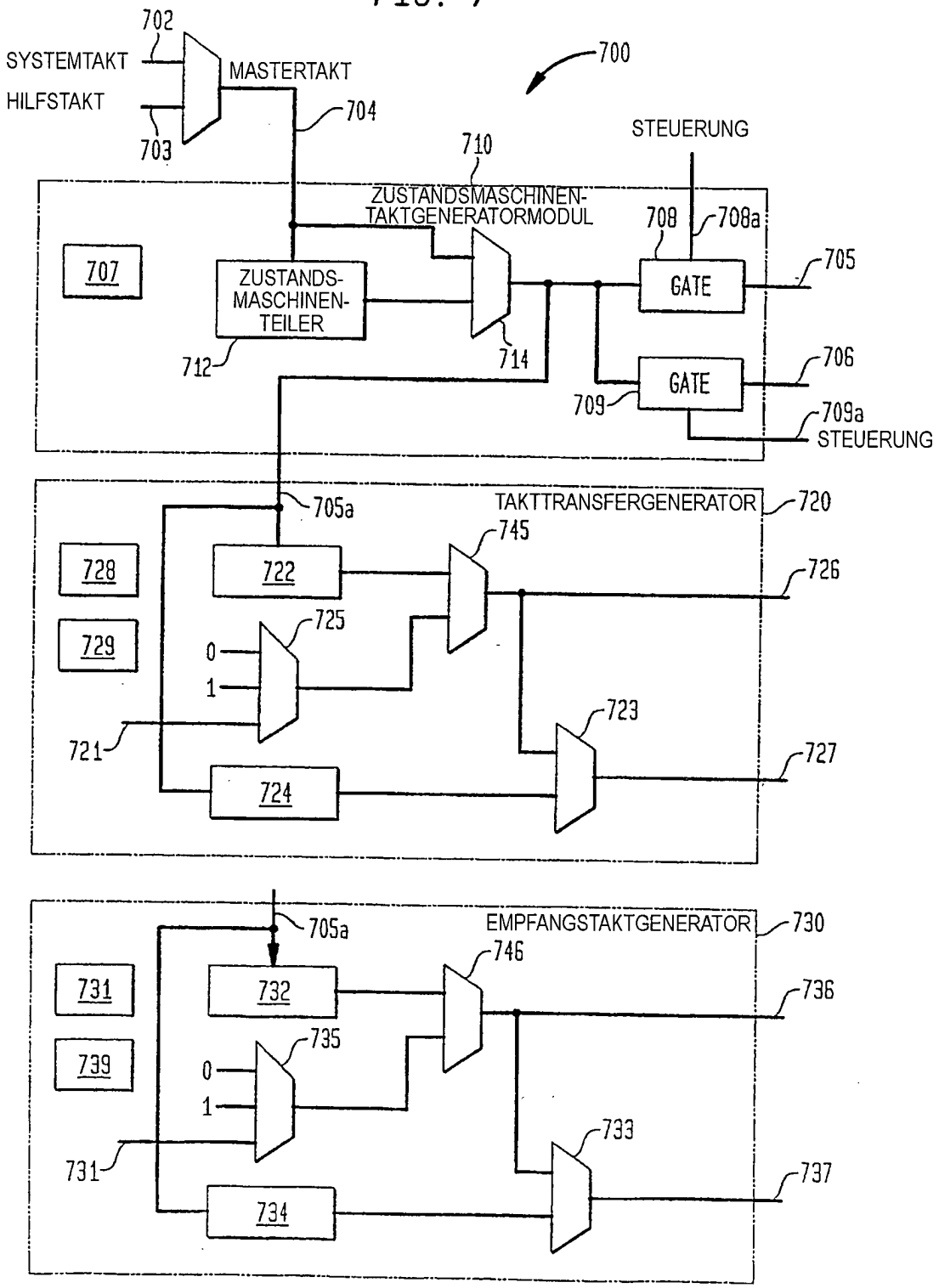

702
SYSTEMTAKT
HILFSTAKT
703
MASTERTAKT
704
700
STEUERUNG
710
ZUSTANDSMASCHINEN-TAKTGENERATORMODUL
708
708a
707
ZUSTANDS-MASCHINEN-TEILER
712
714
GATE
705
GATE
706
709
709a
STEUERUNG
705a
TAKTTRANSFERGENERATOR
720
745
728
722
726
729
725
0
1
721
723
724
727
705a
EMPFANGSTAKTGENERATOR
730
746
731
732
736
739
735
0
1
731
733
734
737

FIG. 8A

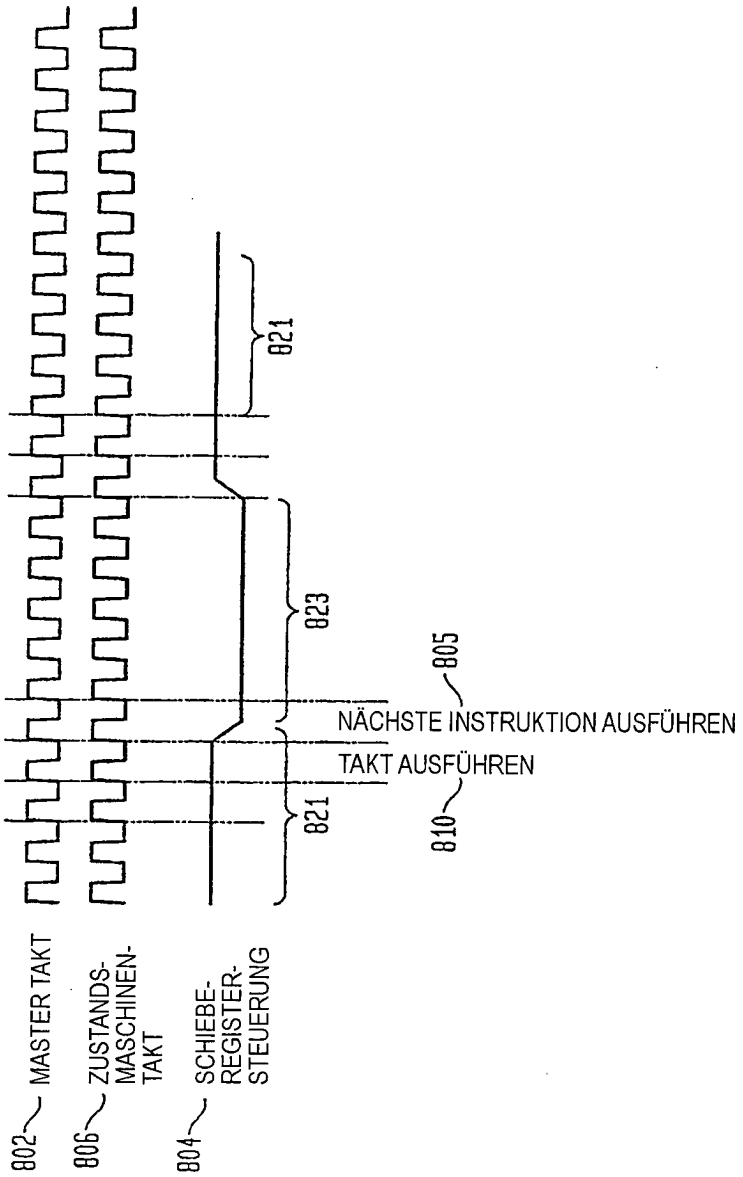

MASTER TAKT
ZUSTANDS-MASCHINEN-TAKT
SCHIEBE-REGISTER-STEUERUNG
802
806
804
821
823
821
805
NÄCHSTE INSTRUKTION AUSFÜHREN
TAKT AUSFÜHREN
810

FIG. 8B

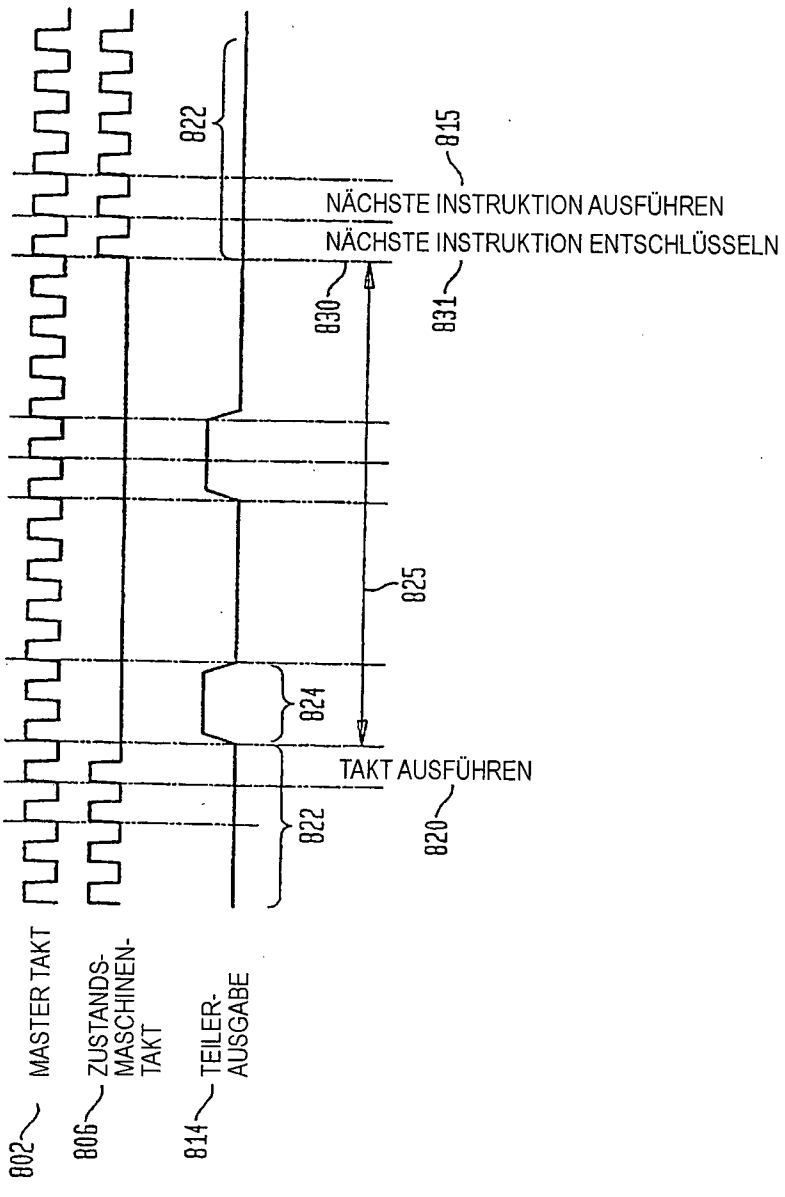

802
MASTER TAKT
806
ZUSTANDS-MASCHINEN-TAKT
814
TEILER-AUSGABE
822
824
825
830
831
815
NÄCHSTE INSTRUKTION AUSFÜHREN
NÄCHSTE INSTRUKTION ENTSCHLÜSSELN
TAKT AUSFÜHREN
820

FIG. 9

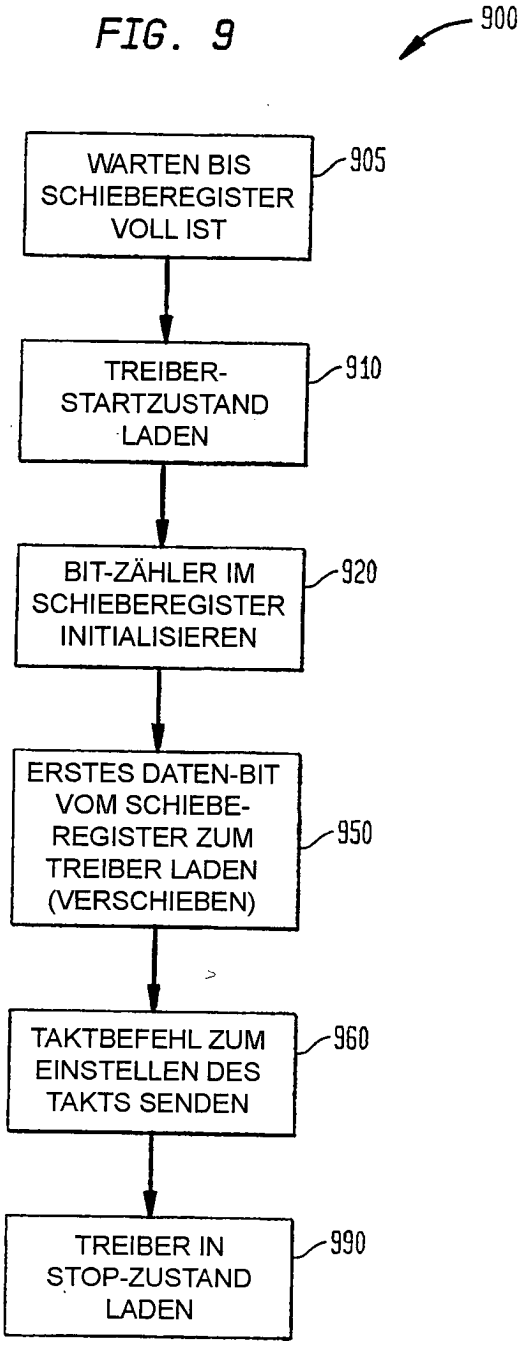
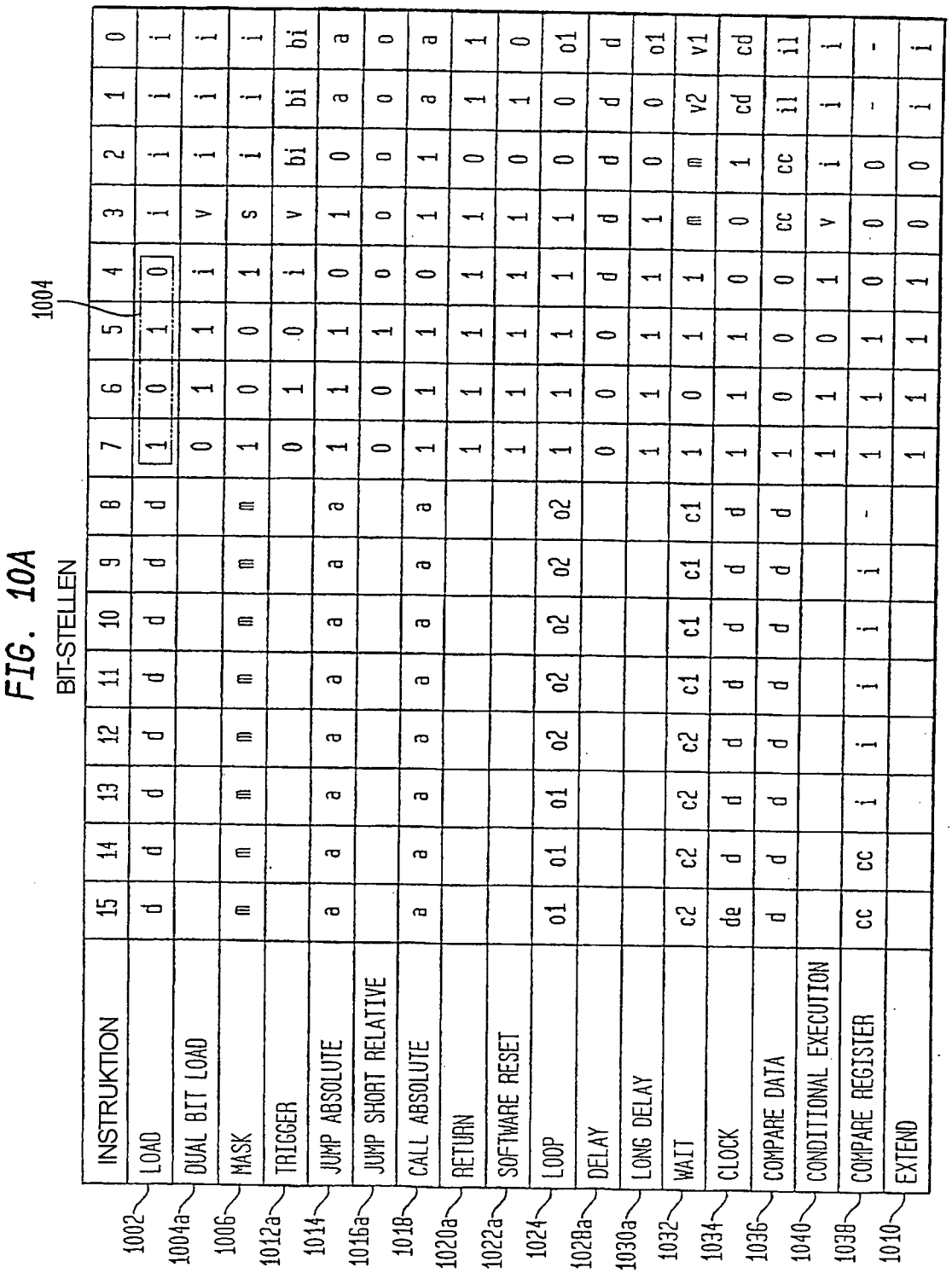

900
WARTEN BIS SCHIEBEREGISTER VOLL IST
905
TREIBER-STARTZUSTAND LADEN
910
BIT-ZÄHLER IM SCHIEBEREGISTER INITIALISIEREN
920
ERSTES DATEN-BIT VOM SCHIEBE-REGISTER ZUM TREIBER LADEN (VERSCHIEBEN)
950
TAKTBEFEHL ZUM EINSTELLEN DES TAKTS SENDEN
960
TREIBER IN STOP-ZUSTAND LADEN
990

FIG. 10A



	INSTRUKTION	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1002	LOAD	d	d	d	d	d	d	d	d	1	0	1	0	i	i	i	i
1004a	DUAL BIT LOAD									0	1	1	i	v	i	i	i
1006	MASK	m	m	m	m	m	m	m	m	1	0	0	1	s	i	i	i
1012a	TRIGGER									0	1	0	i	v	bi	bi	bi
1014	JUMP ABSOLUTE	a	a	a	a	a	a	a	a	1	1	1	0	1	0	a	a
1016a	JUMP SHORT RELATIVE									0	0	1	o	o	o	o	o
1018	CALL ABSOLUTE	a	a	a	a	a	a	a	a	1	1	1	0	1	1	a	a
1020a	RETURN									1	1	1	1	1	0	1	1
1022a	SOFTWARE RESET									1	1	1	1	1	0	1	0
1024	LOOP	o1	o1	o1	o2	o2	o2	o2	o2	1	1	1	1	1	0	0	o1
1028a	DELAY									0	0	0	d	d	d	d	d
1030a	LONG DELAY									1	1	1	1	1	0	0	o1
1032	WAIT	c2	c2	c2	c2	c1	c1	c1	c1	1	0	1	1	m	m	v2	v1
1034	CLOCK	de	d	d	d	d	d	d	d	1	1	1	0	0	1	cd	cd
1036	COMPARE DATA	d	d	d	d	d	d	d	d	1	0	0	0	cc	cc	il	il
1040	CONDITIONAL EXECUTION									1	1	0	1	v	i	i	i
1038	COMPARE REGISTER	cc	cc	i	i	i	i	i	-	1	1	1	0	0	0	-	-
1010	EXTEND									1	1	1	1	0	0	i	i

FIG. 10B

BIT-STELLEN

	INSTRUKTION \FORMAT	15	14	13	12	11	10	9	8
1016b	JUMP SHORT RELATIVE	0	0	o	o	o	o	o	o
1030b	LONG DELAY	0	1	0	1	0	0	i	i
1020b	RETURN	0	1	0	1	1	0	0	0
1022b	SOFTWARE RESET	0	1	0	1	1	0	0	1
1028b	DELAY	0	1	1	d	d	d	d	d
1012b	TRIGGER	1	0	=	i	v	bi,	bi	bi
1004b	DUAL BIT LOAD	1	1	0	1	i	i	i	i
1008	MAP	1	1	1	0	-	-	-	-
1026	NULL	1	1	1	1	1	1	-	-

FIG. 11A

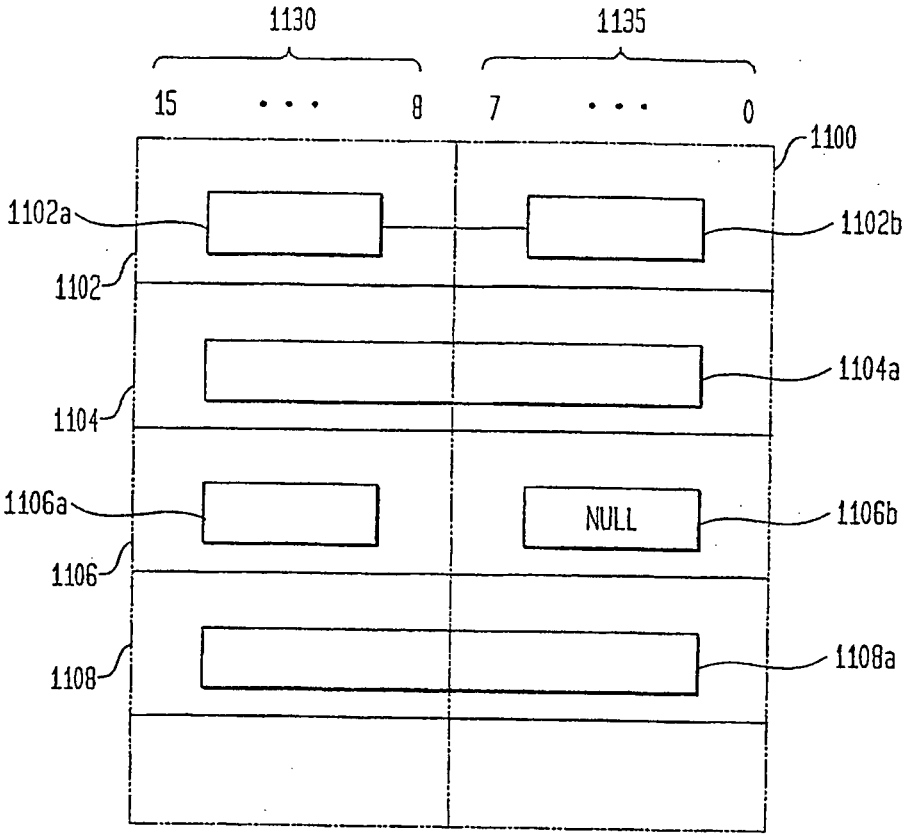


FIG. 11B