(19) REPUBLIK ÖSTERREICH Patentamt

(10) Nummer: **AT 412 747 B**

(12) # PATENTSCHRIFT

(21) Anmeldenummer: A 338/2002
(22) Anmeldetag: 05.03.2002
(42) Beginn der Patentdauer: 15.11.2004
(45) Ausgabetag: 27.06.2005

(51) Int. Cl.[7]: **G09C 1/00**
H04L 9/26, G06F 7/58

(56) Entgegenhaltungen:
JP 2000-101567A
JP 2001-016197A

(73) Patentinhaber:
CORDES RENE-MICHAEL MAG.
A-2323 MANNSWÖRTH,
NIEDERÖSTERREICH (AT).

(72) Erfinder:
CORDES RENE-MICHAEL MAG.
MANNSWÖRTH, NIEDERÖSTERREICH (AT).

(54) CODEGENERATOR UND VORRICHTUNG ZUR SYNCHRONEN ODER ASYNCHRONEN SOWIE PERMANENTEN IDENTIFIKATION ODER VER- UND ENDSCHLÜSSELUNG VON DATEN BELIEBIGER LÄNGE

(57) Codegenerator mit einer Mehrzahl von zu einer codeproduzierenden Reihe (R) geschalteten FLIP-FLOP ($FF_{1,2,...n}$), wobei der Ausgang des in der Reihe (R) letzten Flip-Flop (FF_5) mit dem Eingang des in der Reihe (R) ersten Flip-Flop (FF_1) zu einem Kreis zusammengeschlossen ist und Ausgänge und Eingänge der Flip-Flop unter Zwischenschaltung von EXOR-Gattern rekursiv verschaltet sind. Es ist wenigstens ein EXOR-Gatter ($EXOR_{p1}$) vorgesehen, dessen erster Eingang (1) mit dem Ausgang eines in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_1), dessen zweiter Eingang (2) mit dem Ausgang eines weiteren in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_3) und dessen Ausgang (3) mit dem Eingang des in der codeproduzierenden Reihe (R) dem mit dem ersten Eingang (1) des EXOR-Gatters ($EXOR_{p1}$) verbundenen Flip-Flop (FF_1) nachfolgenden Flip-Flop (FF_2) verbunden ist. Der Ausgang eines in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_5) ist mit dem Eingang eines Inverters (INV) und der Ausgang des Inverters (INV) mit dem Eingang eines anderen in der codeproduzierenden Reihe (R) angeordneten Flip-Flop (FF_1) verbunden.

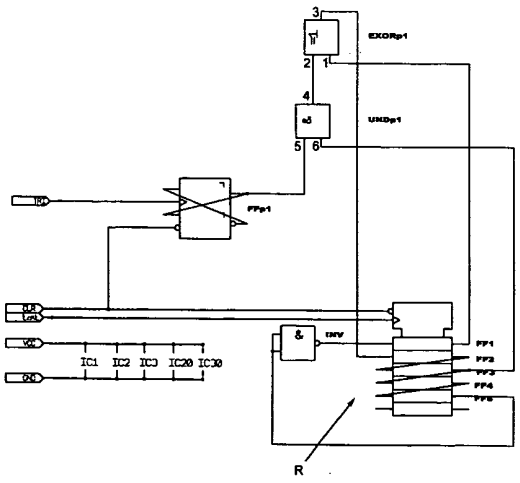


Fig 1

AT 412 747 B

BESCHREIBUNG

Das technische Gebiet auf das sich die beanspruchte Erfindung bezieht

Die beanspruchte Erfindung befaßt sich mit einer Methode und einer Vorrichtung zur Generierung von möglichst vielen unterschiedlichen möglichst langen Codesequenzen, bei niedrigstem Aufwand an Schaltungselementen und unter Verwendung von Bitoperationen, so dass dieser zur simultanen Verschlüsselung von binären Datenströmen hoher Frequenz über lange Zeiträume geeignet ist, wobei sogar der Ausgangscode geheim bleiben soll.

Der bisherigen Stand der Technik

Alle Verschlüsselungsmethoden benützen einen Code, auch wenn die zu verschlüsselnde Information selbst als Code Verwendung findet.

Je besser der Code versteckt ist, desto effektiver ist die Verschlüsselung.

Je länger der Code ist, desto schwerer ist er zu entschlüsseln.

Ein unendlicher Code bräuchte gar nicht versteckt zu werden, da er ja nie ganz bekannt ist.

Funktionell ist jeder Code als unendlich anzusehen, der sich nicht vor dem Ende der zu verschlüsselnden Information wiederholt.

Ein funktionell unendlicher Code hat den Vorteil, dass der Ver- und Entschlüsselungsvorgang selbst denkbar einfach durch eine einzige EXOR - oder EXNOR - Verknüpfung realisiert werden kann.

Ein funktionell unendlicher Code hat den Nachteil, dass er nicht übertragen werden kann. Er muss generiert werden.

Es gibt nun eine einfache Möglichkeit, einen funktionell unendlichen Code zu generieren, indem man die beiden Eingänge eines EXOR - Gatters mit zwei Ausgängen von zu einer Reihe geschalteten Speicherelementen, wie z.b. Flip-Flop bzw. eines Schieberegisters, verbindet und den Ausgang des EXOR - Gatters mit dem Eingang des Schieberegisters rekursiv verschaltet.

Das Ergebnis ist eine Codesequenz, deren maximale Länge

$$L_C = 2^n - 1$$

(L_C = Länge der Codesequenz; n = Anzahl der codegenerierenden in Reihe geschalteten FLIP-FLOP)

Bit beträgt.

Die Probleme die mit der beanspruchten Erfindung gelöst werden sollen

Das Problem ist, dass von dieser Codesequenz leicht auf die Struktur des Generators rückgeschlossen werden kann, so dass sie dann mit einem gleichgebauten Generator nachgeneriert werden kann.

Nachstehende Patente stellen Versuche dar, diese Codesequenzen mit weiteren Prozessen weiter zu verschlüsseln, so dass sie nicht mehr rekonstruierbar sind:

US2001033663 A1 Pseudorandom number generation circuit and data communication system employing the same

WO0105090 A1 A METHOD FOR PROTECTING A PORTABLE CARD

WO9922484 A1 METHOD AND APPARATUS FOR GENERATING ENCRYPTION STREAM CIPHERS

WO9916208 A2 METHOD AND APPARATUS FOR GENERATING ENCRYPTION TREAM CIPHERS

JP10320181 A NON-LINEAR FEEDBACK SHIFT REGISTER CIRCUIT

WO9802990 A1 SPIRAL SCRAMBLING

EP0782069 A1 Pseudorandom number generator

Allen diesen Patenten ist gemein, dass sie Gefahr laufen, dass durch Resonanzeffekte die Länge der produzierten Codes verkürzt wird.

Dann gibt es noch eine Anzahl von Pseudozufallsgeneratoren so etwa wie in den Patentschriften:

JP 2000-101567 A, JP 2001-016197 A, EP 0913964 A2, EP 0782069 A1 beschrieben), die mit Variablen arbeiten, die mit Hilfe eines mathematischen nichtlinearen Umrechnungsalgorithmus aus diesen Variablen eine Codesequenz errechnen. Dies geschieht zur Herabsetzung der Rückrechenbarkeit mit Hilfe von hohen und höchsten mathematischen Funktionen. Diesen Systemen ist

gemein, dass sie sich einer mathematischen Funktionseinheit bedienen, welche einen multi - Bit - Eingang, an dem die in einem Codespeicher befindliche Ausgangsvariable anliegt, sowie einen ein - Bit - Ausgang haben, aus dem die serielle Codesequenz ausgelesen wird, was sich nachteilig auf die maximal erreichbare Codegenerierungsgeschwindigkeit auswirkt. Auch ist es Gegenstand der hohen und höchsten Mathematik, für komplexe Formeln einfache Lösungen zu finden, weshalb das Risiko der Entdeckung einer einfachen Lösung für eine auch noch so komplexen mathematischen Funktion nie ganz auszuschließen und naturgemäß nicht abwägbar ist.

Lösung der Probleme und Vorteile der beanspruchten Erfindung

Die hier gegenständliche Erfindung geht den entgegengesetzten Weg. Sowohl die Struktur des Codegenerators als auch der in ihm ablaufende Algorithmus sind bekannt. Die Struktur ist allerdings so geartet, dass sie eine derartig hohe Anzahl an unterschiedlichen Codes in einer derartig großen Länge zu generieren im Stande ist, dass die Entdeckung des gerade verwendeten Codes so wie die aktuell produzierte Stelle in der Codesequenz nur mit einer extrem geringen Wahrscheinlichkeit möglich ist. Der Generator generiert die Codesequenz auf der niedrigsten möglichen Ebene von Bit - Operationen. Es werden nicht Variablenwerte als Grundlage für die „Berechnung" der Codesequenzen benutzt, sondern lediglich Zustände einzelner FLIP-FLOPS. Daraus ergibt sich die höchstmögliche Effizienz im Verhältnis der Anzahl der eingesetzten Schaltelemente einerseits und der Gesamtlänge der generierbaren Codesequenzen sowie der Anzahl der generierbaren verschiedenen Codes andererseits. Außerdem ist damit sichergestellt, dass der Codegenerator die höchstmögliche Produktionsgeschwindigkeit leisten kann.

Der erfindungsgemäße Codegenerator mit einer Mehrzahl von zu einer codeproduzierenden Reihe geschalteten FLIP-FLOP, wobei der Ausgang des in der Reihe letzten Flip-Flop mit dem Eingang des in der Reihe ersten Flip-Flop zu einem Kreis zusammengeschlossen ist und Ausgänge und Eingänge der Flip-Flop unter Zwischenschaltung von EXOR-Gattern rekursiv verschaltet sind, ist hierbei dadurch gekennzeichnet, dass wenigstens ein EXOR-Gatter vorgesehen ist, dessen erster Eingang mit dem Ausgang eines in der codeproduzierenden Reihe befindlichen Flip-Flop, dessen zweiter Eingang mit dem Ausgang eines weiteren in der codeproduzierenden Reihe befindlichen Flip-Flop und dessen Ausgang mit dem Eingang des in der codeproduzierenden Reihe dem mit dem ersten Eingang des EXOR-Gatters verbundenen Flip-Flop nachfolgenden Flip-Flop verbunden ist und dass der Ausgang eines in der codeproduzierenden Reihe befindlichen Flip-Flop mit dem Eingang eines Inverters und der Ausgang des Inverters mit dem Eingang eines anderen in der codeproduzierenden Reihe angeordneten Flip-Flop verbunden ist, wobei vorzugsweise in die den zweiten Eingang des wenigstens einen EXOR-Gatters und den Ausgang des weiteren in der codeproduzierenden Reihe befindlichen Flip-Flop verbindende Leitung ein UND-Gatter derart geschalten ist, dass der Ausgang des UND-Gatters mit dem zweiten Eingang des EXOR-Gatters, der erste Eingang des UND-Gatters mit dem Ausgang des weiteren in der codeproduzierenden Reihe befindlichen Flip-Flop und der zweite Eingang des UND-Gatters mit dem Ausgang eines codeprogrammierenden Flip-Flop verbunden ist.

Die Erfindung bedient sich hierbei folgender theoretischer Grundlagen:

a) Damit ein Code nicht nachgeneriert werden kann, muss der Generator so geartet sein, dass er so viele verschiedene Codes erstellen kann, dass von einem Abschnitt des einzelnen Codes nicht auf dessen Fortsetzung geschlossen werden kann.

b) Man kann die von einem Codegenerator erzeugte Codesequenz dadurch verändern, dass man zwischen zwei in der Reihe befindlichen FLIP-FLOP (FF_x, FF_{x+1}, ..., FF_{x+y}) ein weiteres EXOR - Gatter einsetzt, an dessen einen Eingang man den Ausgang des gegen die Flussrichtung vorher gelegenen FLIP-FLOP (FF_x) anschließt und den zweiten Eingang vom Ausgang irgendeines weiteren in der Reihe befindlichen FLIP-FLOP (FF_{x+y}) speisen lässt und schließlich mit dem Ausgang des EXOR -Gatters den Eingang des in Flußrichtung nächsten FLIP-FLOP (FF_{x+y}) speist.

c) Damit ausgehend von einer leeren FLIP-FLOP - Reihe ein Code generiert wird, der eine maximale Länge im Verhältnis zur Anzahl der verwendeten FLIP-FLOP aufweist, muss in der gesamten geschlossenen Reihe von FLIP-FLOP ein einzelner Inverter vorhanden sein.

d) Damit die Länge der produzierten Codes nicht durch Resonanzeffekte verkürzt werden, muss durch eine entsprechende Struktur der Eingliederung der verschiedenen codeverändernden EXOR Gatter (EXORp1,2,...pn) sichergestellt werden, dass zwischen den beiden

FLIP-FLOP (FF_x, und FF_{x+y}), welche sich in der zu einem Kreis geschlossenen Reihe und an denen sich die beiden Eingänge der EXOR - Gatter befinden, jeweils keine solchen Teilstrecken (y) der FLIP-FLOP Reihe existieren, die ein Teil oder ein Vielfaches einer anderen Teilstrecke oder der Gesamtstrecke des Kreises sind. Dies kann am effektivsten dadurch realisiert werden, dass die Anzahl (y) der in diesen Teilstrecken befindlichen FLIP-FLOP, sowie deren Gesamtzahl, Primzahlen entsprechen.

e) Damit der Code programmierbar ist, werden diese EXOR - Gatter in dieser ihrer rekursiven Funktion in Abhängigkeit von einem internen Codespeicherinhalt, beispielsweise mit Hilfe eines codeprogrammierenden FLIP-FLOP (FF_{p1}), an- und abschaltbar gemacht.

f) Damit die Codes mit höchstmöglicher Sicherheit nicht entschlüsselt werden können, wird der interne Codespeicherinhalt generiert, und zwar auf eine solche Weise, die sicherstellt, dass nicht einmal der Anwender den Inhalt des internen Codespeichers kennt.

FIGURENÜBERSICHT

Fig. 1 Prinzipschaltung für eine programmierbare rekursive Codegeneration

Fig. 2 Schematisch funktionelle Darstellung der Schaltung der Fig. 1

Fig. 3 Gesamtschaltungsbeispiel für eine aus einem Codegenerator aufgebaute Funktionseinheit zum Aufbau einer verschlüsselten Verbindung zwischen zwei Computern.

Fig. 4 Schematisch funktionelle Darstellung der Schaltung der Fig.3

Detaillierte Beschreibung einer Verwirklichung der Erfindung an Hand der Zeichnungen

Wenn man mehrerre FLIP-FLOP $FF_{1,2...n}$ und ein EXOR - Gatter $EXOR_{p2}$ und ein UND - Gatter UND_{p1} und einen Inverter INV in der Weise verschaltet, dass ein Eingang 2 des EXOR - Gatter $EXOR_{p1}$ mit dem Ausgang 4 des UND - Gatter UND_{p1}, dessen einer Eingang 5 mit einem Ausgang eines codeprogrammierenden FLIP-FLOP FF_{p1} und dessen zweiter Eingang 6 mit dem Ausgang des in der Reihe befindlichen FLIP-FLOP FF_3 und der zweite Eingang 1 des EXOR Gatter $EXOR_{p1}$ mit dem Ausgang des FLIP-FLOP FF1 und der Ausgang 3 der EXOR - Gatter $EXOR_{p1}$ mit dem Eingang des FLIP-FLOP - sohin rekursiv - verbunden ist, so erhält man, ausgehend von einer Reihe völlig leerer FLIP-FLOP, eine Codesequenz generiert. (Schaltung: Fig.1, Spannungszustände an einzelnen Pin der im Prototyp verwendeten Integrierten Schaltungen: Fig.4)

Es vergehen mindestens 3 Takte, ehe sich der Code wiederholt.

Wenn zwischen den oben genannten FLIP-FLOP eine Anzahl von weiteren FLIP-FLOP geschaltet werden (wobei eine Anzahl ununterbrochen in Reihe geschalteter FLIP-FLOP auch in Form von Schieberegistern verwirklicht werden können), so verdoppelt sich die Länge des Codes pro weiterem FLIP-FLOP, so dass sich die Länge des Codes wie folgt berechnet (Schaltung: Fig.3, Spannungszustände an einzelnen Pin der im Prototyp verwendeten Integrierten Schaltungen: Fig.4).

$$L_C = 2^n - 1$$

(L_C = Länge der Codesequenz; n = Anzahl der codegenerierenden in Reihe geschalteten FLIP-FLOP)

Wenn diese Einheit mit einem bestimmten Takt betrieben wird gilt für die Dauer des Codes:

$$T_C = \frac{2^n - 1}{f_C}$$

(T_C = Dauer bis sich der Code wiederholt; f_C = Codegenerierungstaktfrequenz)

Mit weniger als 50 FLIP-FLOP bei einer Codegenerierungstaktfrequenz von 384.000 Bit/s läuft der Code länger als ein Jahr, ohne dass sich die Sequenz wiederholt, so dass ein zu verschlüsselndes Signal simultan über einen ebenso langen Zeitraum verschlüsselt, über eine Standleitung übersendet und entschlüsselt werden kann, so dass Übertragungen live über einen ebenso langen Zeitraum möglich sind.

Wenn man nun an mehren Stellen dieser FLIP-FLOP Reihe zwischen einem FLIP-FLOP und dem nächsten in einer Reihe befindlichen FLIP-FLOP ein EXOR - Gatter einfügt und dieses dann mit dem Signal von einem dritten FLIP-FLOP speist, so verändert man jeweils den dadurch erzeug-

ten Code (Schaltung: Fig. 1, Fig. 3, Schematisch funktionelle Darstellung der Schaltung: Fig. 2, Fig. 4).

Wenn sichergestellt ist, dass die verschiedenen codeverändernden EXOR Gatter EXOR1,2,..,pn, deren erster Eingang von einem Ausgang eines FLIP-FLOP $FF_{1,2,3,4}$ (FFx) gespeist wird, ihren zweiten Eingang jeweils vom Ausgang eines FLIP-FLOP $FF_{8,15,20,23}$ (FF x+y, y=2, +3, +5, + 7, +11, +13, +17, +19, +23,....<n) gespeist erhalten, welches eine Anzahl von FLIP-FLOP in Flussrichtung vom erstgenannten FLIP-FLOP $FF_{1,2,3,4}$ entfernt ist, welche jeweils eine unterschiedliche Primzahl entspricht, die größer als 1 aber kein Teilbetrag der Gesamtzahl der in Reihe geschalteten FLIP-FLOP ist. (Schaltung: Fig. 3, Schematisch funktionelle Darstellung der Schaltung: Fig. 4), kommt es bei der Beeinflussung der Codesequenz zu keinen codesequenzverkürzenden Resonanzeffekten.

Wenn man an einen der beiden Eingänge des jeweiligen EXOR - Gatters EXOR1,2...pn den Ausgang eines UND - Gatters UND1,2,...pn dessen einer Eingang am Ausgang des FLIP-FLOP $FF_{8,15,20,23}$ hängt, anschließt, dann kann man dieses EXOR - Gatter in seiner codeverändernden Wirkung über den zweiten Eingang des UND- Gatters UND1,2,...pn an- und abschalten und wenn man daran jeweils ein weiteres FLIP-FLOP FFp1,p2,...,pn anschließt, das An- und Abschalten der codebeinflussenden Wirkung des EXOR - Gatters programmierbar machen (Schaltung: Fig. 3, S Schematisch funktionelle Darstellung der Schaltung: Fig. 4).

Die Anzahl der programmierbaren unterschiedlichen Codes berechnet sich wie folgt:

$$N_C = 2^{pn} - 1$$

(N_C = Anzahl der möglichen unterschiedlichen Codes; pn = Anzahl der programmierbaren EXOR - Gatter)

Wenn man nun im Besitz eines identen Codegenerators ist, und an Hand einer bestimmten Anzahl von Bit (N_b) den weiteren Verlauf der Codesequenz erschließen möchte, so hängt die Wahrscheinlichkeit (w), mit der man die richtige Fortsetzung der Codesequenz erkennt, sowohl von der Anzahl der in der Codegenerierung verwendeten FLIP-FLOP (n) als auch jener der programmierbaren Code verändernden EXOR - Gatter (pn) ab. Daraus ergibt sich eine Wahrscheinlichkeit, die dem Code zugrundeliegende Programmierung zu entdecken und sohin den weiteren Verlauf des Codes vorauszusagen von:

$$w = \frac{N_b}{(2^n - 1) * (2^{pn} - 1)}$$

(N_b = Anzahl der beobachteten Bit der Codesequenz; n = Anzahl der codegenerierenden in Reihe geschalteten FLIP-FLOP; pn = Anzahl der programmierbar den Code verändernden EXOR - Gatter)

Rechenbeispiel:

233 ist die 52. Primzahl. Wenn man die 1 nicht nützt und die 233 die Gesamtzahl der in Reihe geschalteten FLIP-FLOP ausdrückt, so befinden sich auf dieser Strecke 50 unterschiedliche FLIP-FLOP, welche sich jeweils in Entfernung von einem Ausgangs FLIP-FLOP befinden, die einer Primzahl entspricht (np = 50). Da jedes rekursive EXOR - Gatter $_{1 - 50}$, jeweils zwischen einem nächsten FLIP-FLOP $_{1 - 50}$ beginnend vom ersten in Reihe eingeschaltet ist, verlängert sich die Gesamtlänge der FLIP-FLOP auf (n = 233 + 50 = 283).

Daraus folgt:

$$w = \frac{N_b}{(2^n - 1) * (2^{pn} - 1)} = \frac{N_b}{(2^{283} - 1) * (2^{50} - 1)}$$

$$w = \frac{N_b}{(1{,}5541351138 * 10^{85} - 1) * (1{,}1258999068 * 10^{15} - 1)} \sim \frac{N_b}{1{,}7498005798 * 10^{100}}$$

Mit anderen Worten man muss die Codesequenz $1{,}7498005798 * 10^{100}$ Taktschritte lang

beobachten, damit man mit der Wahrscheinlichkeit 1 eine bestimmte Sequenz entdeckt. Wenn die Taktfrequenz 384000 Hz beträgt, ergibt dies eine notwendige Beobachtungszeit von 1,4449430312 * 10^{87} Jahren.

Indem man die Programm FLIP-FLOP rekursiv miteinander verschaltet (Schaltung: Fig. 3, Schematisch funktionelle Darstellung der Schaltung: Fig.4), so dass sie innerhalb des Zeitintervalls

$$Tpn = \frac{2^{pn} - 1}{f_p}$$

(T pn = Durchlaufzeit aller möglichen Programmierzustände; pn = Anzahl der Programm FLIP-FLOP; f_p = Programmiertaktfrequenz)

sämtliche mögliche Zustandskombinationen durchlaufen, ergibt sich die Programmierung aus einer bestimmte Zeitspanne, in der die Programm FLIP-FLOP mit einem Programmtakt versorgt werden, so dass durch gleichzeitiges Ein- und Ausschalten des Programmtaktes an zwei identen Codegeneratoren {Einschaltimpuls und Ausschaltimpuls an Pin 12 von IC 10a in Schaltung Fig. 3} diese so durchgeführt werden kann, dass wohl mehrere Codegeneratoren idente Codesequenzen generieren, der Inhalt der Programmierung aber nicht einmal den Programmierern bekannt ist.

Indem man nur jeden 2. Zustand eines Muttercodes einen von zwei Tochtercodes zuweist, kann man aus einem zwei Codes schaffen, welche zwar verwandt, aber nicht ähnlich sind (Schaltung: Fig. 3, Schematisch funktionelle Darstellung der Schaltung: Fig. 4).

Die Codierung des Ausgangssignals erfolgt durch ein EXNOR Gatter, (IC 17a, b in Schaltung Fig. 3) in dessen einen Eingang das zu verschlüsselnde Signal und in dessen zweiten Eingang der Code eingebracht wird, so dass an dessen Ausgang das mit dem Code verschlüsselte Ausgangssignal erscheint.

Die Decodierung des Eingangssignals erfolgt durch ein EXNOR Gatter, in dessen einen Eingang das zu entschlüsselnde Signal und in dessen zweiten Eingang der Code eingebracht wird, so dass an dessen Ausgang das mit dem Code entschlüsselte Ausgangssignal erscheint.

Durch Einsatz von wenigstens zwei solcher ident programmierter Codegeneratoren (Schaltung: Fig. 3, Schematisch funktionelle Darstellung der Schaltung: Fig. 4] kann ein zweiter Inhaber desselben Codegenerators identifiziert werden und in weiterer Folge durch Synchronisation der Codegenerationen mit diesem eine ständige, verschlüsselte Datenverbindung aufgebaut werden, über die permanent und live Daten ausgetauscht werden können.

Da die Schaltung bei der Codegenerierung selbst keine CPU - Zeit in Anspruch nimmt, ist sie unabhängig von jedweder Hand Shake Zeit und daher einzig und allein durch die spezifischen Schaltzeiten, der elektronischen Bauelemente, aus denen sie aufgebaut ist, in ihrer Codeproduktionsgeschwindigkeit begrenzt. (Mit handelsüblichen TTL Bauelementen sind so ohne weiteres Codeproduktionsgeschwindigkeiten im Megaherzbereich realisierbar)

Im nun folgenden Abschnitt wird der innere Ablauf der Schaltung nach Fig. 3 schrittweise dargestellt, wobei für die Funktionseinheit nach Schaltung Fig. 3 die Bezeichnung „Schlüssel" verwendet wird:

Innerer Ablauf der Schaltung		
AKTION	REAKTION	FUNKTION
1.) Der Schlüssel A wird mit seinem Ausgang für Computer und Schlüssel mit dem Eingang für Schlüssel des Schlüssels B verbunden	Sowohl in Schlüssel A als auch in Schlüssel B werden die jeweiligen Kontakteingänge auf LOW gesetzt	Die Produktion von Code wird in Schlüssel A und B eingestellt Schlüssel A wird von seinem eigenen Takt betrieben Der Takt wird von Schlüssel A auf Schlüssel B übertragen

Innerer Ablauf der Schaltung		
AKTION	REAKTION	FUNKTION
2.) Die Riegeltaste (Riegel AUF/Riegel ZU) des Schlüssel B wird erstmals betätigt	Das Riegel AUF - Signal wird von Schlüssel A nach Schlüssel B weitergeleitet In beiden Schlüsseln wird ein CLR - Signal abgeleitet	Sämtliche Schieberegister in den beiden Schlüsseln werden gelöscht
3.)	Der Takt wird in den Programmierteil sowohl von Schlüssel A als auch von Schlüssel B eingespeist	Die Programmierung läuft verzahnt synchron in beiden Schlüsseln, jedoch ohne direkte Übertragung des Programminhaltes von Schlüssel A nach Schlüssel B, ab.
4.) Die Riegeltaste des Schlüssel A wird abermals betätigt	Der Takt wird in den Programmierteil sowohl von Schlüssel A als auch von Schlüssel B nicht mehr eingespeist	Synchrones Stopp der Programmierung
5.) Die beiden Schlüssel werden getrennt	Sowohl in Schlüssel A als auch in Schlüssel B werden die jeweiligen Kontakteingänge auf HIGH gesetzt	Die Produktion von Code kann nun sowohl in Schlüssel A und B synchron gestartet werden.

Im nun folgenden Abschnitt wird der schrittweise Ablauf des Aufbaues einer verschlüsselten Verbindung zwischen zwei Computern mit zwei Funktionseinheiten nach Schaltung Fig. 3 dargestellt:

Übersicht der möglichen Abläufe			
MODUS	**Identifikationsmodus**	**Synchronisationsmodus**	**Synchronmodus**
CODE	X	X	X
ADRESSE		X	X
ZEIT			X
Identifizieren	X		
Synchronisieren	X	X	
Decodieren	X	X	X

Identifikationsmodus					
(A)	(B)	PHASE	HANDLUNGEN	SCHLÜSSEL (A)	SCHLÜSSEL (B)
-x	-x	Programmierung	Schlüssel (A) wird mit seinem Computer/Schlüssel Ausgang mit dem Schlüsseleingang von Schlüssel (B) verbunden	Schlüssel (A) wird programmiert und wird ihm der Code C1 als Sendecode und der Code C2 als Empfangscode zugewiesen	Schlüssel (B) wird programmiert und wird ihm der Code C2 als Sendecode und der Code C1 als Empfangscode zugewiesen

Identifikationsmodus					
(A)	(B)	PHASE	HANDLUNGEN	SCHLÜSSEL (A)	SCHLÜSSEL (B)
0	0	Trennung	Schlüssel (A) wird aus Schlüssel (B) herausgezogen	Schlüssel (A) schweigt	Schlüssel (B) schweigt
0	0	Schlüsselaktivierung (A)		Schlüssel (A) wird mit einem Computer verbunden	
0	0			Eine E-Mail wird auf dem Computer von Schlüssel (A) geschrieben	
0 + 1000 + E-Mail	0	Codepaketabruf (A, 1000 + E-Mail)		Die E-Mail wird (live) mit Hilfe von C1 codiert, wobei eine Sequenz von 1000 Bit leeren Codes samt der Codeentstehungszeit uncodiert dem E-Mail vorangestellt wird	
0 + 1000 + E-Mail	0	Datenübertragung		Die E-Mail wird an die Homepage des Computers von Schlüssel (A) gesandt und dort ausgestellt	
0 + 1000 + E-Mail	0	Schlüsselaktivierung (B)			Schlüssel (B) wird mit einem Computer verbunden
0 + 1000 + E-Mail	0 + 1000	Codepaketabruf (B, 1000)			Eine Sequenz von 1000 Bit des Code C1 wird aus Schlüssel (B) ausgelesen und an eine Suchmaschine transferiert
0 + 1000 + E-Mail	0 + 1000				Wenn die Homepage von Schlüssel (A) entdeckt ist wird die E-Mail abgerufen
0 + 1000 + E - Mail	0 + 1000 + E-Mail	Codepaketabruf (B, E- Mail)			Schlüssel (B) decodiert (asynchron) mit Hilfe von C1 das Signal von Schlüssel (A)
0 + 1000 + E - Mail	A	Codefreifluss (B)			Schlüssel (B) generiert nunmehr laufend Code

Identifikationsmodus					
(A)	(B)	PHASE	HANDLUNGEN	SCHLÜSSEL (A)	SCHLÜSSEL (B)
0 + 1000 + E-Mail	A				Schlüssel (B) weiß jetzt wo Schlüssel (A) zu finden ist und kann mit ihm Verbindung aufnehmen um sich mit ihm zu synchronisieren.
Synchronisationsmodus					
B	0				Die E-Mail wird vom Server des Schlüssel (B) abgerufen
C	0 + 1000	Codepaketabruf (B, 1000)			Schlüssel (B) vergleicht die 1000 Bit Sequenz von Schlüssel (A) mit seinem C1 und verschiebt diese so lange bis Synchronität besteht (erkennbar an den leeren 1000 Bit) decodiert (synchron) mit Hilfe von C1 das Signal von Schlüssel (A) Ergebnis: leer
D	0 + 1000 + E-Mail	Codepaketabruf (B, E- Mail)			Schlüssel (B) decodiert (synchronisiert) mit Hilfe von C1 den Inhalt der Botschaft von (A)
E	E	Codefreifluss (A, B)			Schlüssel (B) weiß jetzt wo sich Schlüssel (A) auf der Zeitachse befindet, so dass er nunmehr synchron mit ihm Code aussenden kann und solcherart in den Synchronmodus wechseln kann.
Synchronmodus					
E + 1000	E + 1000			Schlüssel (A) decodiert (live) mit Hilfe von C2 das Signal von Schlüssel (B) Ergebnis: leer	Schlüssel (B) decodiert (live) mit Hilfe von C1 das Signal von Schlüssel (A) Ergebnis: leer

Synchronmodus					
F + Botschaftsdauer	F + Botschaftsdauer	Datenübertragung	Ein Inhalt wird in Schlüssel (A) eingespeist	Schlüssel (A) codiert (live) mit Hilfe von C1 den Inhalt der Botschaft	Schlüssel (B) decodiert (live) mit Hilfe von C1 den Inhalt der Botschaft von (A)
G + Botschaftsdauer	G + Botschaftsdauer		Ein Inhalt wird in Schlüssel (B) eingespeist	Schlüssel (A) decodiert (live) mit Hilfe von C2 den Inhalt der Botschaft von Schlüssel (B)	Schlüssel (B) codiert (live) mit Hilfe von C2 den Inhalt der Botschaft

PATENTANSPRÜCHE:

1. Codegenerator mit einer Mehrzahl von zu einer codeproduzierenden Reihe (R) geschalteten FLIP-FLOP ($FF_{1,2,...n}$), wobei der Ausgang des in der Reihe (R) letzten Flip-Flop (FF_5) mit dem Eingang des in der Reihe (R) ersten Flip-Flop (FF_1) zu einem Kreis zusammengeschlossen ist und Ausgänge und Eingänge der Flip-Flop unter Zwischenschaltung von EXOR-Gattern rekursiv verschaltet sind, **dadurch gekennzeichnet**, dass wenigstens ein EXOR-Gatter ($EXOR_{p1}$) vorgesehen ist, dessen erster Eingang (1) mit dem Ausgang eines in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_1), dessen zweiter Eingang (2) mit dem Ausgang eines weiteren in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_3) und dessen Ausgang (3) mit dem Eingang des in der codeproduzierenden Reihe (R) dem mit dem ersten Eingang (1) des EXOR-Gatters ($EXOR_{p1}$) verbundenen Flip-Flop (FF_1) nachfolgenden Flip-Flop (FF_2) verbunden ist und dass der Ausgang eines in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_5) mit dem Eingang eines Inverters (INV) und der Ausgang des Inverters (INV) mit dem Eingang eines anderen in der codeproduzierenden Reihe (R) angeordneten Flip-Flop (FF_1) verbunden ist, wobei vorzugsweise in die den zweiten Eingang (2) des wenigstens einen EXOR-Gatters ($EXOR_{p1}$) und den Ausgang des weiteren in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_3) verbindende Leitung ein UND-Gatter (UND_{p1}) derart geschalten ist, dass der Ausgang (4) des UND-Gatters (UND_{p1}) mit dem zweiten Eingang (2) des EXOR-Gatters ($EXOR_{p1}$), der erste Eingang (6) des UND-Gatters (UND_{p1}) mit dem Ausgang des weiteren in der codeproduzierenden Reihe (R) befindlichen Flip-Flop (FF_3) und der zweite Eingang (5) des UND-Gatters (UND_{p1}) mit dem Ausgang eines codeprogrammierenden Flip-Flop (FF_{p1}) verbunden ist.
2. Codegenerator nach Anspruch 1, **dadurch gekennzeichnet**, dass eine Mehrzahl von EXOR-Gattern ($EXOR_{p1,p2,p3,p4}$) vorgesehen ist, deren erster Eingang jeweils von einem Ausgang eines in der codeproduzierenden Reihe (R) befindlichen Flip-Flop ($FF_{1,2,3,4}$) gespeist wird und deren zweiter Eingang jeweils vom Ausgang eines weiteren in der codeproduzierenden Reihe (R) befindlichen Flip-Flop ($FF_{8,15,20,23}$) gespeist wird, welches eine Anzahl von Flip-Flop in Flussrichtung der Reihe (R) von dem jeweils mit dem ersten Eingang verbundenen Flip-Flop ($FF_{1,2,3,4}$) entfernt ist, welche jeweils einer unterschiedlichen Primzahl entspricht, die größer als 1 und kein Teilbetrag der Gesamtzahl der in Reihe (R) geschalteten Flip-Flop (FF1,2,...n) ist.
3. Codegenerator nach Anspruch 1 oder 2, **dadurch gekennzeichnet**, dass er wenigstens einen Anschluss für wenigstens einen zweiten, identisch aufgebauten Codegenerator aufweist, sodass beide Codegeneratoren zur gleichen Zeit mit dem selben Programmtakt versorgt werden können.
4. Einrichtung zum Senden und Empfangen von verschlüsselten Informationen mit wenigstens zwei Codegeneratoren nach einem der Ansprüche 1 bis 3, **dadurch gekennzeichnet**, dass die Codegeneratoren jeweils wenigstens einen Anschluss für die gleichzeitige

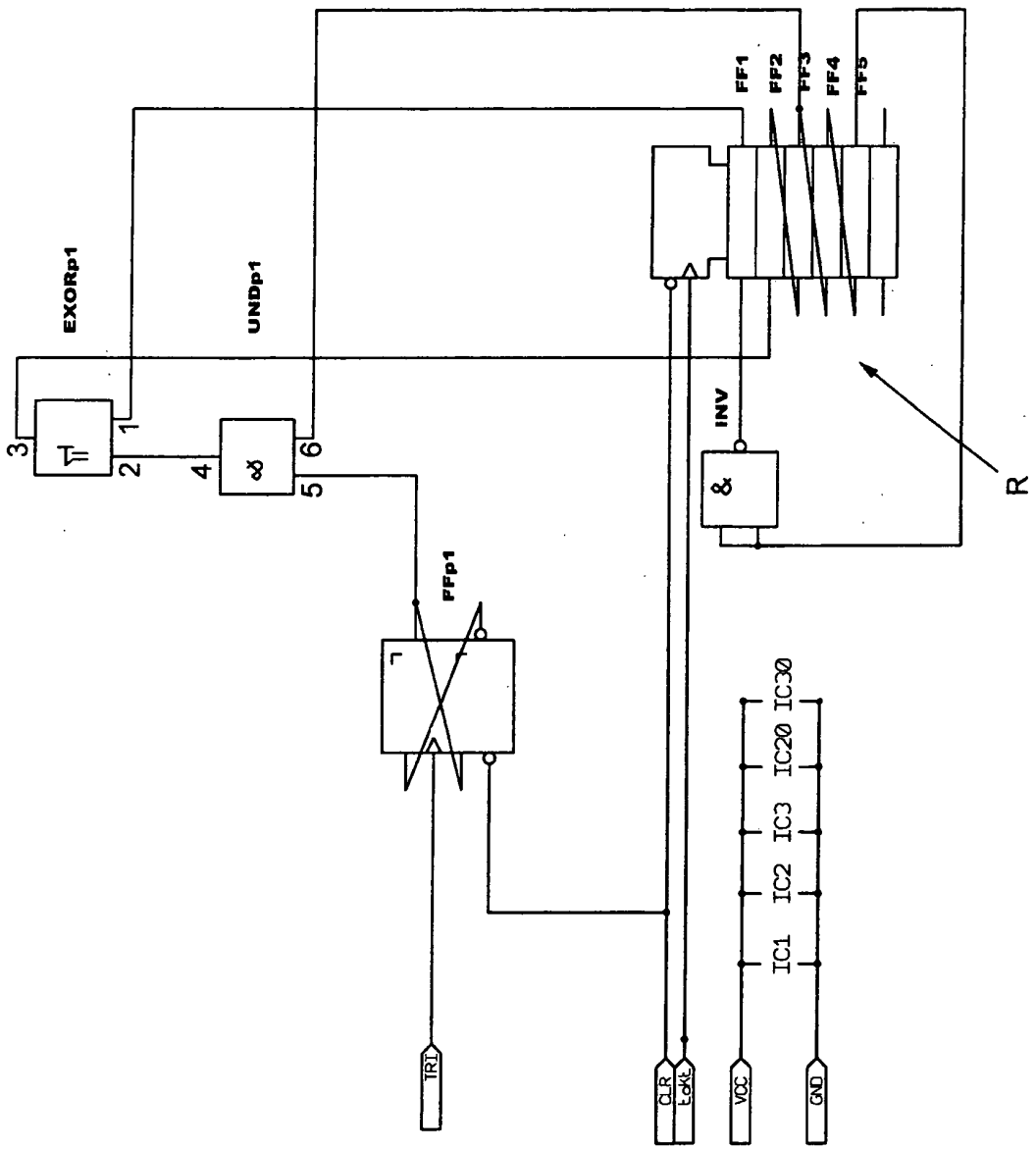
Versorgung der codeprogrammierenden Flip-Flop (FF_{p1}) aller zusammengeschlossenen Codegeneratoren mit demselben Programmtakt aufweisen, sodass die codeprogrammierenden Flip-Flop (FF_{p1}) aller zusammengeschlossenen Codegeneratoren gleichzeitig sämtliche mögliche Zustandskombinationen durchlaufen und bei gleichzeitigem Trennen der Codegeneratoren von dem Programmtakt mit derselben Programmierung versehen sind.

HIEZU 4 BLATT ZEICHNUNGEN

ÖSTERREICHISCHES PATENTAMT
Ausgegeben am: **27.06.2005**
Blatt: **1**

Patentschrift Nr.: **AT 412 747 B**
Int. Cl. [7]: **G09C 1/00,** H04L 9/26, G06F 7/58

Fig 1



ÖSTERREICHISCHES PATENTAMT
Ausgegeben am: **27.06.2005**
Blatt: **2**

Patentschrift Nr.: **AT 412 747 B**
Int. Cl. [7]: **G09C 1/00,** H04L 9/26, G06F 7/58

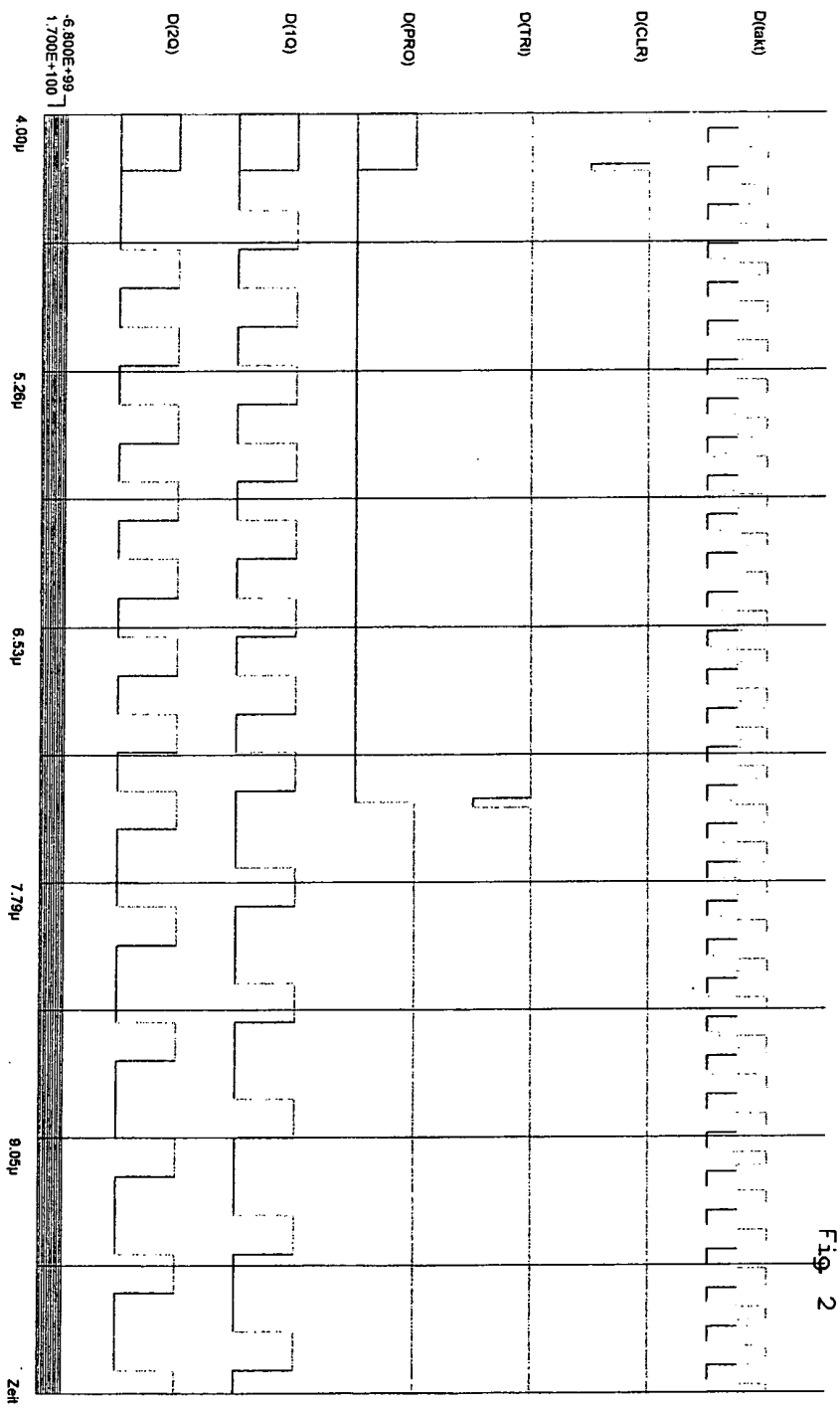


Fig. 2

ÖSTERREICHISCHES PATENTAMT
Ausgegeben am: **27.06.2005**
Blatt: **3**

Patentschrift Nr.: **AT 412 747 B**
Int. Cl. [7]: **G09C 1/00,** H04L 9/26, G06F 7/58

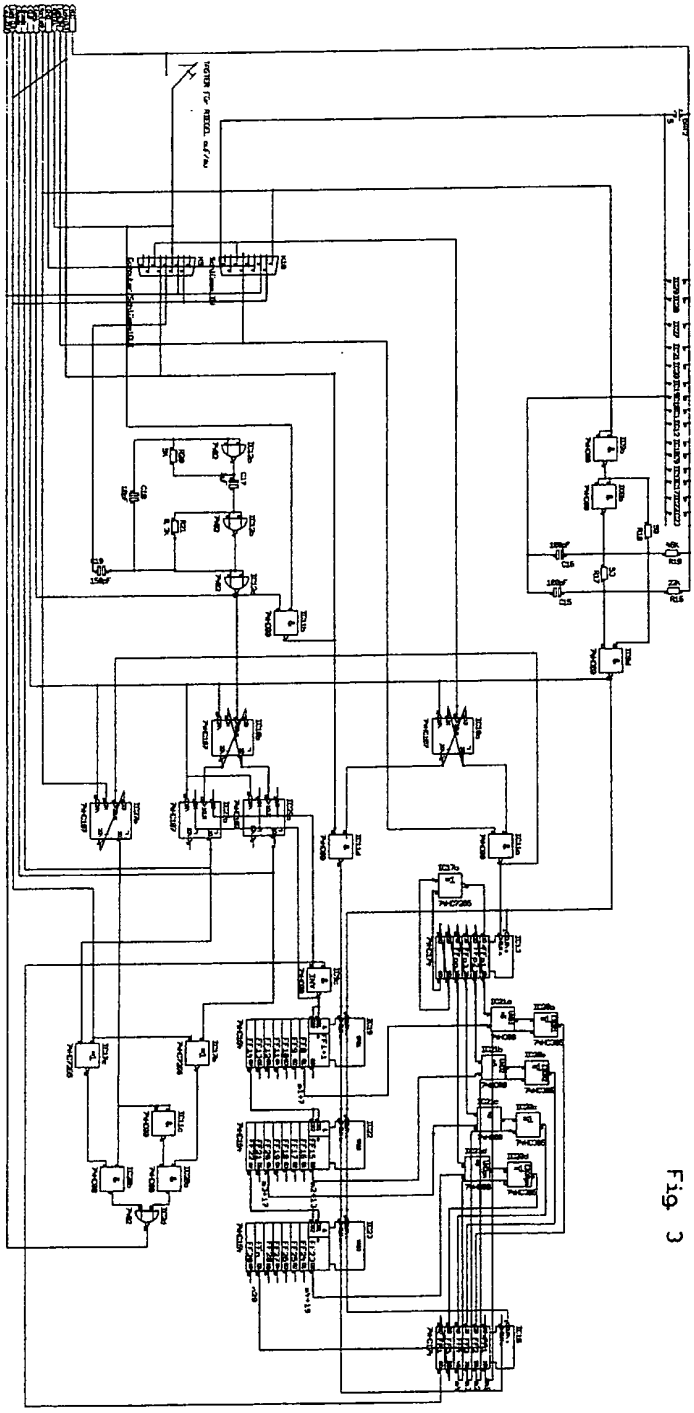


Fig. 3

ÖSTERREICHISCHES PATENTAMT
Ausgegeben am: **27.06.2005**
Blatt: **4**

Patentschrift Nr.: **AT 412 747 B**
Int. Cl. [7]: **G09C 1/00,** H04L 9/26, G06F 7/58

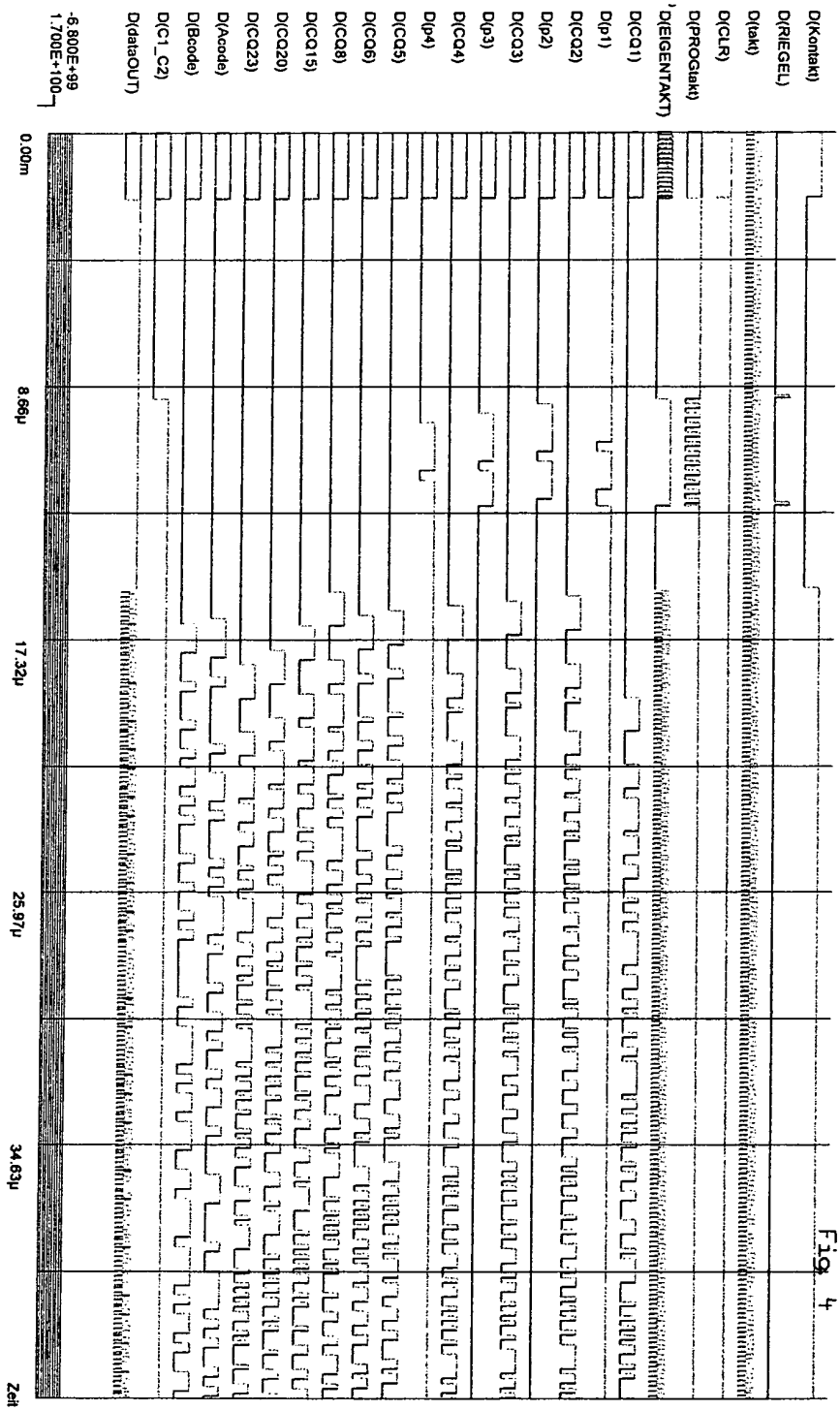


Fig. 4