(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 364 456**

(51) Int. Cl.:
H04L 9/26 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **03707883 .9**
(96) Fecha de presentación : **05.03.2003**
(97) Número de publicación de la solicitud: **1481509**
(97) Fecha de publicación de la solicitud: **01.12.2004**

(54) Título: **Generador de código y dispositivo para la identificación síncrona o asíncrona y permanente o el encriptado y desencriptado de datos de cualquier longitud.**

(30) Prioridad: **05.03.2002 AT A 338/2002**

(45) Fecha de publicación de la mención BOPI:
02.09.2011

(45) Fecha de la publicación del folleto de la patente:
02.09.2011

(73) Titular/es: **René Cordes**
Raiffeisengasse 3
2323 Mannswörth, AT
Ernst Schobesberger y
M&C CONSULT INVEST & TRADE GmbH

(72) Inventor/es: **Cordes, René**

(74) Agente: **Curell Suñol, Marcelino**

DESCRIPCIÓN

Generador de código y dispositivo para la identificación síncrona o asíncrona y permanente o el encriptado y desencriptado de datos de cualquier longitud.

La presente invención se refiere a un generador de código con una pluralidad de elementos de memoria conectados para formar una serie que genera código, por ejemplo flip-flops (biestables), estando conectada la salida del último elemento de memoria en la serie con la entrada del primer elemento de memoria de la serie para formar un circuito, estando prevista por lo menos una puerta lógica EXOR, cuya primera entrada está conectada con la salida de un elemento de memoria que se encuentra en la serie generadora de código, cuya segunda entrada lo está con la salida de otro elemento de memoria que se encuentra en la serie generadora de código y cuya salida está conectada con la entrada del elemento de memoria posterior al elemento de memoria conectado con la primera entrada de la puerta lógica EXOR en la serie generadora de código. Un generador de código de este tipo se describe en el artículo “PSEUDORANDOM BIT GENERATORS IN STREAM-CIPHER CRYPTOGRAPHY” de Kencheng Zeng, publicado en COMPUTER, IEEE COMPUTER SOCIETY, LONG BEACH., CA, US, US, Vol. 24, Nº 2, 1 de Febrero de 1991 (1991-02-01), páginas 8-17, XP000219462 USSN: 0018-9162. Los generadores de código se utilización para el encriptado y la transmisión de informaciones a través de redes de comunicación. En principio, todos los métodos de encriptado utilizan un código, aunque la propia información que haya que encriptar se utilice como código. Cuanto mejor esté escondido el código utilizado para la encriptación tanto más efectiva será la encriptación. Cuanto más largo sea el código más difícil será de descifrar. Por ejemplo, un código infinito no necesitaría ser escondido dado que no se conoce nunca completamente. Funcionalmente, debe considerarse como infinito cualquier código que no se repite antes del final de la información que hay que encriptar. Un código funcionalmente infinito tiene la ventaja de que el propio proceso de encriptado y desencriptado se puede realizar de manera sencilla mediante una única conexión EXOR ó EXNOR. Un código funcionalmente infinito tiene la desventaja de que no puede ser transmitido; debe ser generado.

Existe una posibilidad sencilla de generar un código funcionalmente infinito, gracias a conectar las dos entradas de una puerta lógica EXOR con dos salidas de elementos de memoria conectados para forma una serie, es decir por ejemplo un registro de desplazamiento, y conectando la salida de la puerta lógica EXOR de manera recursiva con la entrada de registro de desplazamiento.

El resultado es una secuencia de código cuya longitud máxima tiene

$$L_c = 2^n - 1$$

(L_c = longitud de la secuencia de código; n = número de los elementos de memoria generadores de código conectados en serie)

bits.

En este generador de código, es desventajoso el hecho de que a partir de la secuencia de código se puede deducir con facilidad la estructura del generador, de manera que puede ser regenerada con un generador construido de igual manera. Las patentes que se mencionan a continuación representan intentos de continuar encriptando estas secuencias de código mediante otros procesos, de manera que ya no se puedan reconstruir: US 2001033663, WO 01/05090, WO 99/22484, WO 99/16208, JP10320181, WO 98/02990 y EP 0782069. Los generadores de código que se han dado a conocer a partir de estas publicaciones tienen en común que la longitud del código producido es acortada mediante efectos de resonancia. Además, un número de generadores suedoaleatorios, como se describen por ejemplo en los documentos JP 2000-101567, JP 2001-016197, EP 0913964 ó EP 0782069. Estos generadores de código trabajan con variables, calculándose con la ayuda de un algoritmo de conversión matemático no lineal una secuencia de código a partir de estas variables. Esto sucede para la reducción de la posibilidad del cálculo de vuelta con la ayuda de funciones matemáticas superiores y muy superiores. Estos sistemas tienen en común que se sirven de una unidad funcional matemática, la cual tiene una entrada multi-Bit, en la cual está la variable de salida que se encuentra la memoria de código, así como una salida de un-Bit, en la cual se lee una secuencia de código serie, lo cual tiene un efecto desventajoso sobre la velocidad de generación de código máxima que se puede alcanzar. La matemática superior y muy superior tiene como función también encontrar soluciones sencillas para fórmulas compleja, motivo por el cual el riesgo del descubrimiento de una solución sencilla para una función matemático por muy compleja que sea no se puede excluir por completo y no se puede, naturalmente, dejar de evaluar.

La presente invención se refiere a crear un dispositivo para la generación de muchas secuencias de código diferentes, lo más largas posibles, debiendo encontrarse lo suficiente con una complejidad de elementos de conmutación pequeña. El generador de código debe ser adecuado, mediante la utilización de operaciones con bits, para el encriptado simultáneo de flujos de datos binarios de alta frecuencia durante intervalos de tiempo largos, debiendo permanecer secreto incluso el código de salida.

Para la solución de este problema la invención prevé un generador de código según la reivindicación 1. Al mismo

tiempo, son conocidos tanto la estructura del generador de código como también el algoritmo que se hace correr en el mismo. La estructura es de todos modos de tal tipo que está en disposición de generar un número tan grande de códigos diferentes con una longitud de tal manera grande que el descubrimiento del código que se utiliza en ese instante así como la posición generadas actualmente en la secuencia de código es posible únicamente con una probabilidad extremadamente pequeña. El código no se puede regenerar, cuando el generador puede generar tantos códigos diferentes, que a partir de una sección del código individual no se puede concluir acerca de su continuación. El generador genera la secuencia de código al nivel más bajo posible de operaciones-Bit. No se utilizan valores de variable como base para el cálculo de las secuencias de código sino únicamente estados de elementos de memoria individuales, como o por ejemplo flip-flops o registros de desplazamiento conectados para forma una serie. De ello resulta la mayor eficiencia posible en relación con el número de elementos de conmutación utilizados, por un lado, y con respecto a la longitud total de las secuencias de código generadas así como al número de los diferentes códigos que se pueden generar, por la otra. Además, se asegura con ello que el generador de código puede producir la velocidad de producción más alta posible.

Según la invención la secuencia de código generada por el generador de código es modificada gracias a que entre dos elementos de memoria que se encuentran en la serie generadora de código está introducida otra puerta lógica EXOR a cuya primera entrada se conecta la salida de un primer elemento de memoria y que permite alimentar la entrada por la salida de otro elemento de memoria cualquiera que se encuentra en la serie y que, finalmente, alimenta con la salida de la puerta lógica EXOR la entrada del elemento de memoria conectado al primer elemento de memoria en la dirección del flujo de la serie.

Para que partiendo de una serie de elementos de memoria vacíos se genere un código, el cual presente una longitud máxima en relación con el número de elementos de memoria utilizados, debe haber en la totalidad de la serie cerrada de elementos de memoria un único inversor. Naturalmente la función del inversor puede estar reunida con la función de la puerta lógica EXOR en un elemento de conmutación, por ejemplo con la ayuda de una puerta lógica EXNOR.

Para programar códigos diferentes se hacen conectables y desconectables preferentemente la o las puertas lógicas EXOR, en cuanto a su función recursiva, dependientes de un contenido de memoria de código interno. Con este propósito, se ha perfeccionado la invención de tal manera que la línea que conecta la segunda entrada de la por lo menos una puerta lógica EXOR y la salida del otro elemento de memoria que se encuentra en la serie generadora de código está conectada de tal manera una puerta lógica Y, que la salida de la puerta lógica Y está conectada con la segunda entrada de la puerta lógica EXOR, la primera entrada de la puerta lógica Y lo está con la salida del otro elemento de memoria que se encuentra en la serie generadora de código y la segunda entrada de la puerta lógica Y lo está con la salida de un elemento de memoria que sirve para la programación. El estado del elemento de memoria correspondiente que sirve para la programación determina por consiguiente si la puerta lógica EXOR correspondiente está conectada o desconectada. Como consecuencia un elemento de memoria de este tipo se designa como elemento de memoria programador de código.

Para estructurar el código con mayor riqueza en variantes está previsto preferentemente una pluralidad de puertas lógicas EXOR, cuya primera puerta lógica es alimentada en cada caso por una salida de un elemento de memoria que se encuentra en la serie generadora de código y cuya segunda entrada es alimentada en cada caso por la salida de otro elemento de memoria que se encuentra en la serie generadora de código, que está alejado de un número de elementos de memoria en la dirección del flujo de la serie del elemento de memoria conectado en cada caso con la primera entrada, que corresponde, en cada caso, a un número primo, el cual es mayor que 1 y no es una suma parcial del número total de los elementos de memoria conectados en la serie. De este modo, la longitud del código generado no se acorta por efectos de resonancia. Al mismo tiempo, se asegura, mediante una estructura correspondiente de la integración de las diferentes puertas lógicas EXOR modificadoras de código que entre los dos elementos de memoria, que se encuentra en una serie generadora de código cerrada para dar un circuito y en los cuales se encuentra las dos entradas de la puerta lógica EXOR, no existan en cada caso tramos parciales de este tipo de la serie de elementos de memoria, que sean una parte o un múltiplo de otro tramo parcial o del tramo total del circuito. Esto se puede realizar de la manera más efectiva gracias a que el número de elementos de memoria que se encuentra en estos tramos parciales así como su número total correspondan a números primos.

En un perfeccionamiento preferido de la invención el contenido interno de la memoria de código debe ser generado de tal manera, que ni el usuario conozca el contenido de la memoria interna de código. De este modo, se continua dificultando la descodificación del código. Con este propósito, se ha adoptado la formación preferentemente de tal manera que está previsto un gran número de elementos de memoria programadores de código, asociados, en cada caso, a una puerta lógica Y y a una puerta lógica EXOR y que están conectados en una serie cerrada formando un circuito y estando asignada por lo menos una puerta lógica EXOR, cuya primera entrada está conectada con la salida de un elemento de memoria que se encuentra en la serie programadora de código, cuya segunda entrada lo está con la salida de otro elemento de memoria que se encuentra en la serie programadora de código y cuya salida lo está con la entrada del elemento de memoria que, en la serie programadora de código, viene a continuación del elemento de memoria conectado con la primera entrada de la puesta lógica EXOR. La programación de los estados de las puertas lógicas EXOR modificadoras de código con la ayuda de las puertas lógicas Y es llevada a cabo, por consiguiente, por una serie de elementos de memoria separada, la cual está conectada de forma recursiva, de la

misma manera que es el caso en la serie de elementos de memoria generadora de código, mediante la utilización de por lo menos una puerta lógica EXOR. La programación tiene lugar al mismo tiempo gracias a que la serie de elementos de memoria programadora de código es alimentada con una cadencia de programa pudiendo programarse de forma y manera sencilla un gran número de generadores de código a un código idéntico cuando, como corresponde a una formación preferida, el generador de código presenta por lo menos una conexión para por lo menos un segundo generador de código, estructurado idéntico, de manera que ambos generadores de código pueden ser alimentados, en el mismo instante, con la misma cadencia de programa.

La invención se refiere además a un dispositivo para la emisión y recepción de informaciones encriptadas con por lo menos dos generadores de código, presentando los generadores de código en cada caso por lo menos una conexión para la alimentación simultánea de los elementos de memoria programadores de código de todos los generadores de código conectados con la misma cadencia de programa, de manera que los elementos de memoria programadores de código de todos los generadores de código conectados recorran al mismo tiempo todas las posibles combinaciones de estado y que en caso de separación simultánea de los generadores de código de la cadencia de programa estén dotados con la misma programación.

En un perfeccionamiento preferido según las reivindicaciones 4, 5 y 8, se conectan y desconectan también las diferentes puertas lógicas EXOR programadoras de código de nuevo de una serie de elementos de memoria programadora formada por otros elementos de memoria en una acción influida mediante programa, de manera que en la programación no haya que recorrer linealmente todos los estados de programación posibles y por ello, a partir de la duración del tiempo de programación, no se pueda deducir ni siquiera aproximadamente el estado de los códigos tras su programación.

La invención se explica a continuación con mayor detalle a partir de los ejemplos de formas de realización representados en el dibujo. En éste la Fig. 1 muestra una representación esquemática para una generación de código recursiva programable, la Fig. 2 un ejemplo de conexión total para una unidad funcional constituida por un generador de código, con las cuales se puede crear una conexión encriptada entre dos ordenadores, y la Fig. 3 un ejemplo de conexión total de un generador de código modificado.

La Fig. 1 muestra una representación esquemática en la cual están conectados cinco elementos de memoria, es decir los flip-flops $FF_{1,2,3,4,5}$ conectados entre sí para dar una serie R generadora de código, y una puerta lógica EXOR $EXOR_{p1}$ y una puerta lógica Y Y_{p1} y un inversor INV, y ello de manera que la entrada 2 de la puerta lógica EXOR $EXOR_{p1}$ está conectada con la salida 4 de la puerta lógica Y Y_{p1}, cuya una entrada 5 lo está con una salida de un elemento de memoria FF_{p1} que sirve para la programación y cuya otra entrada 6 lo está con la salida del elemento de memoria FF_3 que se encuentra en la serie generadora de código, y la otra entrada 1 de la puerta lógica EXOR $EXOR_{p1}$ lo está con la salida del elemento de memoria FF_1 que se encuentra en la serie R generadora de código y la salida 3 de la puerta lógica EXOR $EXOR_{p1}$ lo está con la entrada del elemento de memoria FF_2 y la salida del elemento de memoria FF_5 lo está con las entradas del inversor INV y la salida del inversor INV lo está de nuevo con la entrada del siguiente elemento de memoria FF_1, en la dirección del flujo, en la serie – por consiguiente de forma recursiva. Con esta conexión se obtiene generada una secuencia de código, partiendo de una serie R de elementos de memoria $FF_{1,2,3,4,5}$ completamente vacíos. Pasan por lo menos 3 cadencias antes de que el código se repita. Los elementos de conmutación individuales se pueden hacer realidad con componentes usuales en el comercio: por ejemplo, se puede utilizar un IC del tipo 74HC174 para elementos de memoria $FF_{1,2,3,4,5}$ alineados unos con otros, asimismo un IC 74HC08 para la puerta lógica Y Y_{p1}, un IC 74HC386 para la puerta lógica EXOR $EXOR_{p1}$, un IC 74HC00 para el inversor INV y un IC 74HC107 para el componente de memoria FF_{p1}.

La serie mostrada en la Fig. 1 puede ser naturalmente prolongada y puede resultar, por ejemplo, una serie prolongada como está representada en la Fig. 2. Al mismo tiempo se puede realizar un cierto número de elementos de memoria conectados de forma interrumpida en serie también en forma de registro de desplazamiento SRG_1, SRG_2, ... Se dobla la longitud del código por cada elemento de memoria añadido, de manera que la longitud del código se calcula como sigue:

$$L_c = 2^n - 1$$

(L_c = longitud de la secuencia de código; n = número de elementos de memoria conectados en la serie generadora de código).

Cuando esta unidad se hace funcionar con una determinada cadencia se cumple para la duración del código;

$$T_c = \frac{2^n - 1}{f_c}$$

(T_c = duración hasta que el código se repite; f_c = frecuencia de cadencia de generación de código).

Con menos de 50 elementos de memoria para una frecuencia de cadencia de generación de código de 384.000 Bit/s el código corre más de un año sin que se repita la secuencia, de manera que una señal que haya que encriptar puede ser simultáneamente, a lo largo de un intervalo de tiempo igual de largo, encriptada, enviada a través de una línea de estado y desencriptada, de manera que son posibles transmisiones en vivo a lo largo de un intervalo de tiempo igual de largo.

Cuando, para una longitud correspondiente de la serie de elementos de memoria R, se introduce en varios puntos de esta serie de elementos de memoria R, entre un elemento de memoria $FF_{1,2,3,4}$ y el siguiente elemento de memoria $FF_{2,3,4,5}$ que se encuentra en la serie R, una puerta lógica EXOR $EXOR_{p1,p1,p2,p4}$ y se alimenta ésta con la señal de un tercer elemento de memoria $FF_{8,15,20,23}$, entonces se modifica en cada caso el código generado con ello (Fig. 2).

En caso de un gran número de puertas lógicas EXOR $EXOR_{p1,p2,p3,p4}$ modificadoras de código, ver la Fig. 2, debe estar asegurado que las diferentes puertas lógicas EXOR $EXOR_{p1,p2,p3,p4}$ modificadoras de código, cuya primera entrada es alimentada por una salida de un elemento de memoria $FF_{1,2,3,4}$, ven alimentada su segunda entrada en cada caso por la salida de un segundo elemento de memoria $FF_{8,15,20,23}$, el cual está alejado de un número de elementos de memoria en la dirección del flujo del elemento de memoria $FF_{1,2,3,4}$ mencionado en primer lugar, el cual corresponde en cada caso a un número primo diferente, el cual es mayor que 1 pero no es una suma parcial del número total de elementos de memoria conectados en la serie R, de manera que al influir sobre la secuencia de código no se producen efectos de resonancia que acorten el código. Entre los pares de elementos de memoria $FF_{1,8}$; $FF_{2,15}$; $FF_{3,20}$; $FF_{4,23}$ correspondientes hay por lo tanto en cada caso un número de 7, 13, 17 y 19 (números primos) de elementos de memoria.

Cuando en una de las dos entradas 2 de la puerta lógica EXOR $EXOR_{p1}$ ó $EXOR_{p1,p2,p3,p4}$ correspondiente se conecta la salida 4 de una puerta lógica Y Y_{p1} ó $Y_{p1,p2,p3,p4}$, cuya una entrada 6 cuelga a la salida del elemento de memoria FF_3 ó $FF_{8,15,20,23}$, entonces se puede conectar y desconectar esta puerta lógica EXOR $EXOR_{p1}$ ó $EXOR_{p1,p2,p3,p4}$ en cuando a su acción modificadora de código a través de la segunda entrada 5 de la puerta lógica Y Y_{p1} ó $Y_{p1,p2,p3,p4}$ y, cuando se conecta allí en cada caso otro elemento de memoria FF_{p1} ó $FF_{p1,p2,p3,p4}$, se puede hacer programable la conexión y desconexión de la acción modificadora de código de la puerta lógica EXOR $EXOR_{p1}$ ó $EXOR_{p1,p2,p3,p4}$ (Fig. 1 ó Fig. 2). Los elementos de memoria $FF_{p1,p2,p3,p4}$ programadores de código pueden ser conectados entre sí para formar una serie RR. A continuación los propios elementos de memoria $FF_{p1,p2,p2,p4}$ programadores de código pueden ser conectados de manera recursiva con la ayuda de una puerta lógica EXOR $EXOR_{pp1}$.

El número de códigos diferentes programables se calcula como sigue:

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

(N_c = número de posibles códigos diferentes; pn = número de puertas lógicas EXOR $EXOR_{p1,p2,\ldots pn}$) programables.

Cuando se está en posesión de un generador de código idéntico y, sobre la base de un número determinado de Bit, se quiere obtener el desarrollo posterior de la secuencia de código, entonces la probabilidad con la cual se reconoce la continuación correcta de la secuencia de código depende tanto del número de los elementos de memoria $FF_{1,2,\ldots n}$ utilizados en la codificación como también de las puertas lógicas EXOR $EXOR_{p1,p2,\ldots pn}$ modificadoras del código programables. De ello resulta una probabilidad de descubrir la programación en la cual de basa del código y por consiguiente de poder predecir el desarrollo posterior del código:

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

(N_b = número de Bits observados de la secuencia de código; n = número de elementos de memoria $FF_{1,2\ldots n}$ generadores de código conectados en serie; pn = número de las puertas lógicas EXOR $EXOR_{p1,p2,\ldots pn}$ que modifican el código de forma programable).

Ejemplo:

El 233 es el 52. Número primo. Cuando no se utiliza el 1 y el 233 expresa el número total de elementos de memoria conectados en serie, entonces se encuentra en este tramo 50 elementos de memoria distintos, los cuales se encuentran en cada caso a distancia de un elemento de memoria de salida, que corresponde a un número primo (np = 50). Dado que cada puerta lógica $EXOR_{1-50}$ recursiva, está conectada por el primero en la serie empezando en cada caso entre un elemento de memoria$_{1-50}$, la longitud total de los elementos de memoria se prolonga hasta (n = 233 + 50 = 283).

De ello resulta:

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

$$W = \frac{N_b}{(1{,}5541351138 * 10^{85} - 1) * (1{,}1258999068 * 10^{15} - 1)}$$

$$W \sim \frac{N_b}{1{,}7498005798 * 10^{100}}$$

Dicho con otras palabras, hay que observar la secuencia de código 1,7498005798 * 10^{100} pasos de cadencia para descubrir, con una probabilidad de 1, una determinada secuencia. Cuando la frecuencia de cadencia es de 384000 Hz esto da un tiempo de observación necesario de 1,4449430312 * 10^{87} años.

Conectando los elementos de memoria ($FF_{p1,p2,p3,p4,p5,p6}$) programadores de código de forma recursiva entre sí de manera que recorran, durante el intervalo de tiempo

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

(T pn = tiempo de recorrido de todos los estados de programación posibles; pn = número de los elementos de memoria de programa; f_p = frecuencia de cadencia de programación),
todas las combinaciones de esta posibles, resulta la programación de un intervalo de tiempo determinado, en el cual los elementos de memoria programadores de código son alimentados con una cadencia de programa, de manera que mediante la conexión y desconexión simultánea de la cadencia de programa en dos generadores de código idénticos (impulso de conexión e impulso de desconexión en el Pin 12 del IC 10a en el circuito de la Fig. 2) éstas son llevadas a cabo de tal manera que varios generadores de código generan secuencias de código idénticas, si bien el contenido de la programación no es conocido sin embargo ni para los programadores.

Para que de la duración de la programación no se pueda deducir, tampoco aproximadamente, la programación puede tener lugar en dos etapas. Para ello, se puede añadir otro nivel de programación gracias a que la propia puerta lógica EXOR $EXOR_{pp1}$ programadora de código, mediante la interconexión de una puerta lógica Y Y_{pp1}, sea conectada con una serie de elementos de memoria RRR y, por consiguiente, sea hecha programable, utilizándose de nuevo una puerta lógica EXOR $EXOR_{ppp1}$ para la conexión recursiva de la serie RRR (Fig. 3).

En el primer nivel de programación se programa el programador, de manera que explora un segmento de los posibles estados de programación, la cual representa entonces el punto de partida para la programación que viene a continuación, en la cual pueden ser explorados todos los puntos de un segmento de este tipo.

Partiendo del ejemplo de cálculo expuesto más arriba se garantiza con ello que los (2^{283}-1) * (2^{50}-1) estados diferentes son estructurados en 2^{50}-1 segmentos diferentes, de los cuales uno es escogido en la primera fase de programación. Este proceso de elección tiene lugar en como máximo 2^{ppn}-1 pasos (ppn = número de números primos, que están contenidos en un número de números primos (50) utilizados en una programación, es decir 16). Esto significa que deben tener lugar como máximo 2^{16} pasos antes de que la totalidad de los segmentos hayan sido localizados. Para una frecuencia de cadencia de programación de 1 MHz este proceso se ha concluido en 0,065 segundos. Un intervalo de tiempo, el cual es recorrido en cada programación, ya que está en el tiempo de reacción del ser humano, por lo cual está garantizado que del tiempo de programación que ha transcurrido realmente no se puedan extraer conclusiones sobre la programación de los códigos.

Asignando únicamente cada 2º estado de un código madre a dos códigos hija se puede generar a partir de un código dos, los cuales están emparentados pero no son iguales.

La codificación de la señal de salida tiene lugar mediante una puerta lógica EXNOR (IC 17b, c Fig. 2) en una de cuyas entradas se introduce la señal que hay que encriptar y en cuya segunda entrada de introduce el código, de manera que en su salida aparece la señal de salida encriptada con el código.

La descodificación de la señal de entrada tiene lugar mediante una puerta lógica EXNOR, en una de cuyas entradas

se introduce la señal que hay que descodificar y en cuya segunda entrada se introduce el código, de manera que en su salida aparece la señal de salida descodificada con el código.

Mediante la utilización de por lo menos dos generadores de código programables idénticos puede ser identificado un segundo poseedor del mismo generador de código y, como consecuencia, mediante sincronización de las generaciones de código, se puede establecer con éste una conexión de datos encriptada continua, a través de la cual se pueden intercambiar datos de manera permanente y en vivo.

Dado que la conexión durante la generación de código no consume tiempo de CPU es independiente de cualquier tiempo de Hand-Shake y está limitada por ello única y exclusivamente por los tiempos de conmutación específicos de los componentes electrónicos con los que está construida en cuanto a su velocidad de generación de código. Con componentes TTL usuales en el comercio se puede realizar de esta manera, sin más, velocidades de generación de código en el margen de los Megahercios.

En la sección que viene ahora a continuación se representa, paso a paso, el desarrollo interno de la conexión según la Fig. 2, utilizándose para la unidad funcional según la conexión de la Fig. 2 la designación de “llave”:

Desarrollo interno de la conmutación		
ACCIÓN	REACCIÓN	FUNCIÓN
1.) La llave A se conecta con su salida para ordenador y llave con la entrada para llave de la llave B	Tanto en la llave A como también en la llave B se ponen las entradas de contacto correspondiente a LOW	La producción de código es ajustada en las llaves A y B. La llave A es accionada por su propia cadencia. La candencia es transmitida de la llave A a la llave B
2.) La tecla de cierre (cierre ABIERTO / cierre CERRADO) de la llave B es accionada por primera vez	La señal cierre ABIERTO es conducida desde la llave A hacia la llave B. En ambas llaves se deriva una señal CLR	Todos los registros de desplazamiento en ambas llaves son borrados
3.)	La cadencia es almacenada en la parte de programación tanto de la llave A como también de la llave B	La programación se desarrolla de manera síncrona dentada en ambas llaves, si bien sin transmisión directa del contenido del programa de la llave A a la llave B
4.) La tecla de cierre de la llave A es accionada nuevamente	La cadencia en la parte de programación tanto de la llave A como también de la llave B ya no se almacena	Parara síncrona de la programación
5.) Las dos llaves son separadas	Tanto en la llave A como también en la llave B se ponen las entradas de contacto a HIGH	La producción de código puede iniciarse ahora de manera síncrona en la llave A y la B

En la siguiente sección, se representa el desarrollo, paso a paso, de una conexión encriptada entre dos ordenadores con dos unidades funcionales según la conexión de la Fig. 2:

Vista general de los posibles desarrollos			
MODO	Modo de identificación	Modo de sincronización	Modo síncrono
CÓDIGO	X	X	X
DIRECCIÓN		X	X
TIEMPO			X
identificar	X		
sincronizar	X	X	
descodificar	X	X	X

Modo de identificación					
(A)	(B)	FASE	ACCIONES	LLAVE (A)	LLAVE (B)
-x	-x	programación	La llave (A) es conectada con su salida ordenador/ llave con la entrada de llave de la llave (B)	La llave (A) es programada y se le asigna el código C1 como código de emisión y el código C2 como código de recepción	La llave (B) es programada y se le asigna el código C2 como código de emisión y el código C1 como código de recepción
0	0	separación	La llave (A) es extraída de la llave (B)	La llave (A) guarda silencio	La llave (B) guarda silencio
0	0	Activación de la llave (A)		La llave (A) es conectada con un ordenador	
0	0			Se escribe un E-Mail en el ordenador de la llave (A)	
0 + 1000 + E-Mail	0	Llamada del paquete de código (A, 1000 + E-Mail)		El E-Mail es codificado (en vivo) con la ayuda de C1, anteponiéndose una secuencia de 1000 Bit de código vacío, incluido el tiempo de formación del código, sin codificar al E-Mail	
0 + 1000 + E-Mail	0	Transmisión de datos		El E-Mail es enviado a la Homepage del ordenador de la llave (A) y es mostrado allí	
0 + 1000 + E-Mail	0	Activación de la llave (B)			La llave (B) es conectada con un ordenador
0 + 1000 + E-Mail	0 + 1000	Llamada del paquete de código (B, 1000)			Una secuencia de 1000 Bit del código C1 es leída en la llave (B) y transferida a al buscador

(continuación)

Modo de identificación					
(A)	(B)	FASE	ACCIONES	LLAVE (A)	LLAVE (B)
0 + 1000 + E-Mail	0 + 1000				Cuando la Homepage es descubierta por la llave (A) se abre el E-Mail
0 + 1000 + E-Mail	0 + 1000 + E-Mail	Llamada del paquete de código (B, E-Mail)			La llave (B) descodifica (se forma asíncrona) con la ayuda de C1 la señal de la llave (A)
0 + 1000 + E-Mail	A	Flujo libre de código (B)			La llave (B) genera ahora código constantemente
0 + 1000 + E-Mail	A				La llave (B) sabe ahora donde puede encontrar la llave (A) y puede establecer conexión con ella para sincronizarse con ella.
Modo de sincronización					
B	0				El E-Mail es llamado por el servidor de la llave (B).
C	0 + 1000	Llamada del paquete de código (B, 1000)			La llave (B) compara la secuencia de 1000 Bit de la llave (A) con su C1 y desplaza ésta hasta que existe sincronía (reconocible por los 1000 Bit vacíos) descodifica (de manera síncrona) con la ayuda de C1 la señal de la llave (A). Resultado: vacía
D	0 + 1000 + E-Mail	Llamada del paquete de código (B, E-Mail)			La llave (B) descodifica (de manera sincronizada) con la ayuda de C1 el contenido del mensaje de (A).

(continuación)

Modo de identificación					
(A)	(B)	FASE	ACCIONES	LLAVE (A)	LLAVE (B)
E	E	Flujo libre de código (A, B)			La llave (B) sabe ahora dónde se encuentra la llave (A) sobre el eje de tiempo, de manera que puede emitir ahora de forma síncrona código en ella y puede cambiar de esta manera al modo síncrono
Modo síncrono					
E + 1000	E + 1000			La llave (A) descodifica (en vivo) con la ayuda de C2 la señal de la llave (B). Resultado: vacía	La llave (B) descodifica (en vivo) con la ayuda de C1 la señal de la llave (A). Resultado: vacía
F + duración del mensaje	F + duración del mensaje	Transmisión de datos	Se almacena un contenido en la llave (A)	La llave (A) descodifica (en vivo) con la ayuda de C1 el contenido del mensaje	La llave (B) descodifica (en vivo) con la ayuda de C1 el contenido del mensaje de (A)
G + duración del mensaje	G + duración del mensaje		Se almacena un contenido en la llave (B)	La llave (A) descodifica (en vivo) con la ayuda de C2 el contenido del mensaje de la llave (B)	La llave (B) codifica (en vivo) con la ayuda de C2 el contenido del mensaje

REIVINDICACIONES

1. Generador de código con una pluralidad de elementos de memoria ($FF_{1, 2, ...n}$) conectados para formar una serie (R) que genera código, por ejemplo, flip-flops, estando conectada la salida del último elemento de memoria (FF_5) en la serie (R) con la entrada del primer elemento de memoria (FF_1) de la serie (R) para formar un circuito, estando prevista por lo menos una puerta lógica EXOR ($EXOR_{p1}$), cuya primera entrada (1) está conectada con la salida de un elemento de memoria (FF_1) que se encuentra en la serie (R) generadora de código, cuya segunda entrada (2) lo está con la salida de otro elemento de memoria (FF_3) que se encuentra en la serie (R) generadora de código y cuya salida (3) está conectada con la entrada del elemento de memoria (FF_2) posterior al elemento de memoria (FF_1) conectado con la primera entrada (1) de la puerta lógica EXOR ($EXOR_{p1}$) en la serie (R) generadora de código, caracterizado porque las salidas y por lo menos una entrada de los elementos de memoria están conectadas de manera recursiva con interconexión de por lo menos una puerta lógica EXOR ($EXOR_{p1}$), en la línea que conecta la segunda entrada (2) de dicha por lo menos una puerta lógica EXOR ($EXOR_{p1}$) y la salida del otro elemento de memoria (FF_3) que se encuentra en la serie (R) generadora de código está conectada de tal manera una puerta lógica Y (Y_{p1}), que la salida (4) de la puerta lógica Y (Y_{01}) está conectada con la segunda entrada (2) de la puerta lógica EXOR ($EXOR_{p1}$), la primera entrada (6) de la puerta lógica Y (Y_{p1}) lo está con la salida del otro elemento de memoria (FF_3) que se encuentra en la serie (R) generadora de código y la segunda entrada (5) de la puerta lógica Y (Yp_1) lo está con la salida de un elemento de memoria (FF_{p1}) programado mediante código y porque la salida de un elemento de memoria (FF_5) que se encuentra en la serie (R) generadora de código está conectada con la entrada de un inversor (INV) y la salida del inversor (INV) lo está con la entrada de otro elemento de memoria (FF_1) dispuesto en la serie (R) generadora de código.

2. Generador de código según la reivindicación 1, caracterizado porque está prevista una pluralidad de puertas lógicas EXOR ($EXOR_{p1, p2, p3, p4}$), cuya primera entrada es alimentada en cada caso por un elemento de memoria ($FF_{1, 2, 3, 4}$) que se encuentra en la salida de una serie (R) generadora de código y cuya segunda entrada es alimentada, en cada caso, por la salida de otro elemento de memoria ($FF_{8, 15, 20, 23}$) que se encuentra en la serie (R) generadora de código, el cual está alejado, en cada caso, en un número de elementos de memoria en la dirección de circulación de la serie (R) de elemento de memoria ($FF_{1, 2, 3, 4}$) conectado, en cada caso, con la primera entrada, el cual corresponde, en cada caso, a un número primo diferente, que es mayor que 1 y no es una suma parcial del número total de elementos de memoria ($FF_{1, 2, ...n}$) conectados en la serie (R).

3. Generador de código según la reivindicación 1 ó 2, caracterizado porque está previsto una pluralidad de elementos de memoria ($FF_{p1, p2, p3, p4, ...pn}$) programadores de código, asignados en cada caso a una puerta lógica Y ($Y_{p1, p2, p3, p4}$) y a una puerta lógica EXOR ($EXOR_{p1, p2, p3, p4}$) y que están conectados en una serie (RR) cerrada formando un circuito y estando asociada por lo menos una puerta lógica EXOR ($EXOR_{pp1}$), cuya primera entrada está conectada con la salida de un elemento de memoria (FF_{p6}) que se encuentra en la serie (RR) programadora de código, cuya segunda entrada lo está con la salida de otro elemento de memoria (FF_{p5}) que se encuentra en la serie (RR) programadora de código y cuya salida lo está con la entrada del elemento de memoria (FF_{p1}) que, en la serie (RR) programadora de código, viene a continuación del elemento de memoria (FF_{p6}) conectado con la primera entrada de la puesta lógica EXOR ($EXOR_{pp1}$).

4. Generador de código según la reivindicación 3, caracterizado porque en la línea que conecta la segunda entrada de dicha por lo menos una puerta lógica EXOR ($EXOR_{pp1}$) y la salida del otro elemento de memoria (FF_{p3}) que se encuentra en la serie (RR) programadora de código está conectada de tal manera una puerta lógica Y (Y_{pp1}) que la salida de la puerta lógica Y (Y_{pp1}) está conectada con la segunda entrada de la puerta lógica EXOR ($EXOR_{pp1}$), la primera entrada de la puerta lógica Y (Y_{pp1}) lo está con la salida del otro elemento de memoria (FF_{p3}) que se encuentra en la serie (RR) programadora de código y la segunda entrada de la puerta lógica Y (Y_{pp1}) lo está con la salida de un elemento de memoria (FF_{pp5}) que sirve para la programación de la serie (RR) programadora de código.

5. Generador de código según la reivindicación 4, caracterizado porque está prevista una pluralidad de elementos de memoria ($FF_{pp1, pp2, pp3, pp4, ...ppn}$) que sirven para la programación de la serie (RR) programadora de código, asociados, en cada caso, a una puerta lógica Y (Y_{pp1}) y a una puerta lógica EXOR ($EXOR_{pp1}$) y que están conectados en una serie (RRR) que forma un circuito y que está prevista por lo menos una puerta lógica EXOR ($EXOR_{ppp1}$), cuya primera entrada está conectada con la salida de un elemento de memoria (FF_{pp1}) que se encuentra en la serie (RRR), cuya segunda entrada está conectada con la salida de otro elemento de memoria (FF_{pp3}) que se encuentra en la serie (RRR) y cuya salida está conectada con la entrada del elemento de memoria (FF_{pp2}) que viene a continuación del elemento de memoria (FF_{pp1}) conectada, en la serie (RRR), con la primera entrada de la puerta lógica EXOR ($EXOR_{ppp1}$).

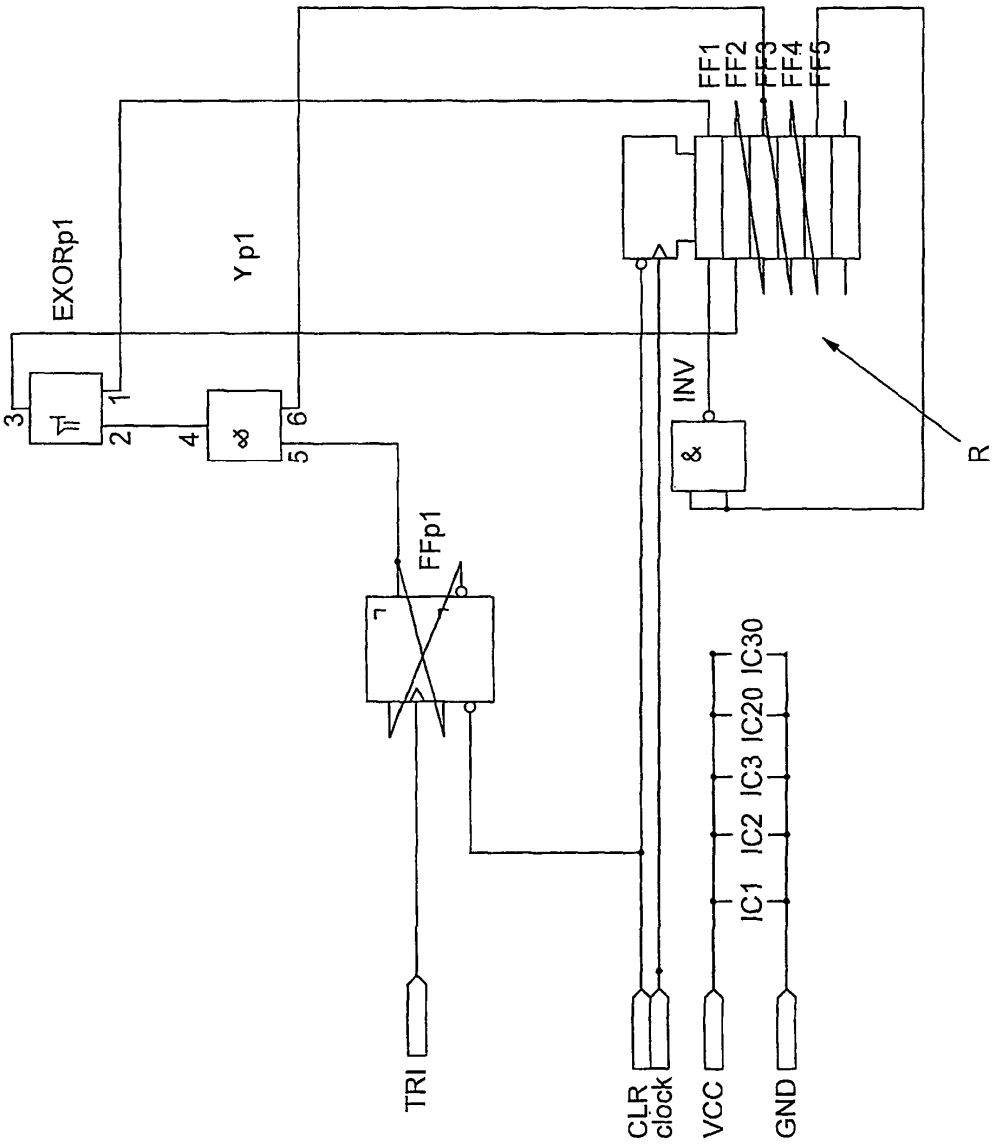
6. Generador de código según una de las reivindicaciones 1 a 5, caracterizado porque presenta por lo menos una conexión para por lo menos un segundo generador de código, estructurado idéntico, de manera que ambos generadores de código puedan ser alimentados al mismo tiempo con la misma cadencia de programa.

7. Dispositivo para la emisión y recepción de informaciones encriptadas con por lo menos dos generadores de código según una de las reivindicaciones 1 a 6, caracterizado porque los generadores de código presentan, en cada caso, por lo menos una conexión para la alimentación simultánea de los elementos de memoria ($FF_{p1, p2, p3, p4}$)

programadores de código de todos los generadores de código conectados con la misma cadencia de programa, de manera que los elementos de memoria ($FF_{p1,\ p2,\ p3,\ ...pn}$) programadores de código de todos los generadores de código conectados recorran al mismo tiempo todas las posibles combinaciones de estado y que en caso de separación simultánea de los generadores de código de la cadencia de programa estén provistos de la misma programación.

8. Dispositivo según la reivindicación 7, caracterizado porque los generadores de código presentan, en cada caso, dos conexiones para la alimentación simultánea de los elementos de memoria ($FF_{p1,\ p2,\ p3,\ ...pn}$) programadores de código y de los elementos de memoria ($FF_{pp1,\ pp2,\ pp3,\ ...ppn}$) que sirven para la programación de los elementos de memoria ($FF_{p1,\ p2,\ p3,\ ...pn}$) programadores de código de todos los generadores de código conectados con dos cadencias de programa que corren de manera independiente, recorriendo los elementos de memoria ($FF_{pp1,\ pp2,\ pp3,\ ...ppn}$) que sirven para la programación de los elementos de memoria ($FF_{p1,\ p1,\ p3,\ p4}$) programadores de código todas las posibles combinaciones de estado por lo menos una vez y los elementos de memoria ($FF_{p1,\ p2,\ p3,\ ...pn}$) programadores de código de todos los programadores conectados recorren simultáneamente un número determinado de todas las posibles combinaciones de estado y, tras la separación simultánea de los generadores de código de las cadencias de programa, todos los generadores de código están provistos de la misma programación.

Fig. 1


FF1
FF2
FF3
FF4
FF5
EXORp1
Yp1
INV
R
FFp1
TRI
CLR
clock
VCC
GND
IC1
IC2
IC3
IC20
IC30

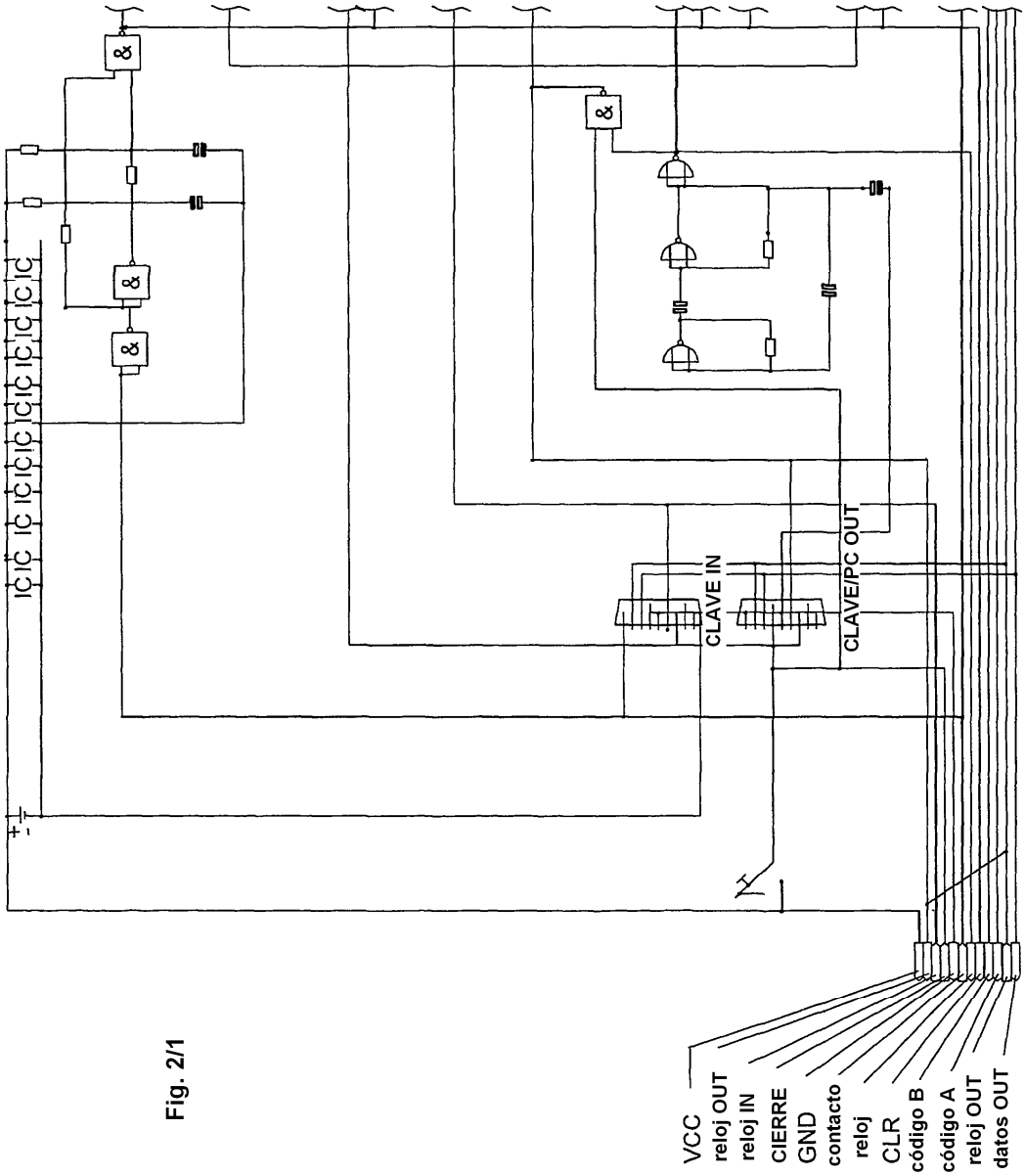
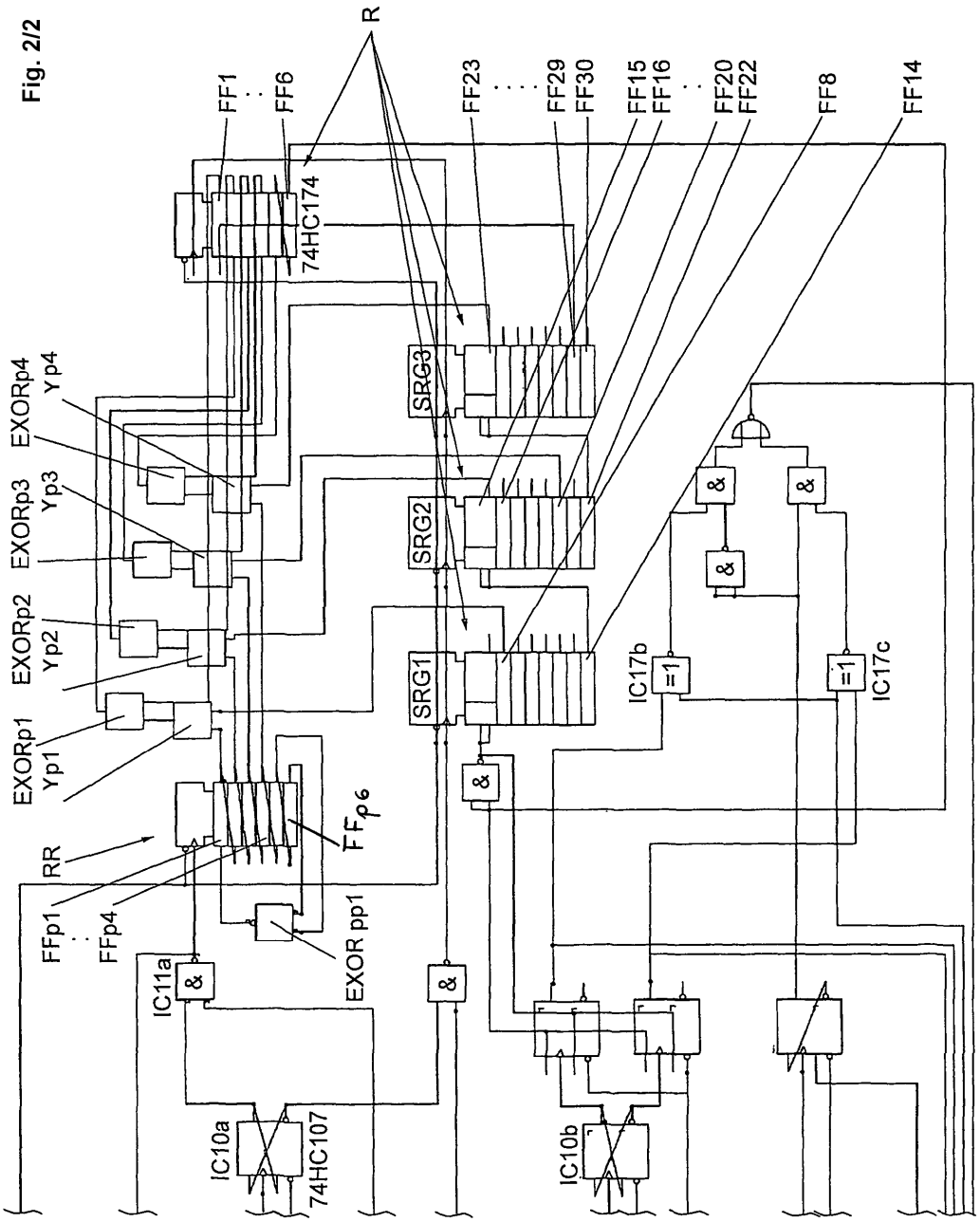

&
&
&
&
IC IC IC IC IC IC IC IC IC IC IC IC
+
-
CLAVE IN
CLAVE/PC OUT
VCC
reloj OUT
reloj IN
CIERRE
GND
contacto
reloj
CLR
código B
código A
reloj OUT
datos OUT

Fig. 2/1

Fig. 2/2


R
FF1
FF6
FF23
FF29
FF30
FF15
FF16
FF20
FF22
FF8
FF14
74HC174
SRG3
SRG2
SRG1
EXORp4
Yp4
EXORp3
Yp3
EXORp2
Yp2
EXORp1
Yp1
RR
FFp1
FFp4
EXOR pp1
IC11a
IC10a
74HC107
IC10b
IC17b
IC17c
=1
&

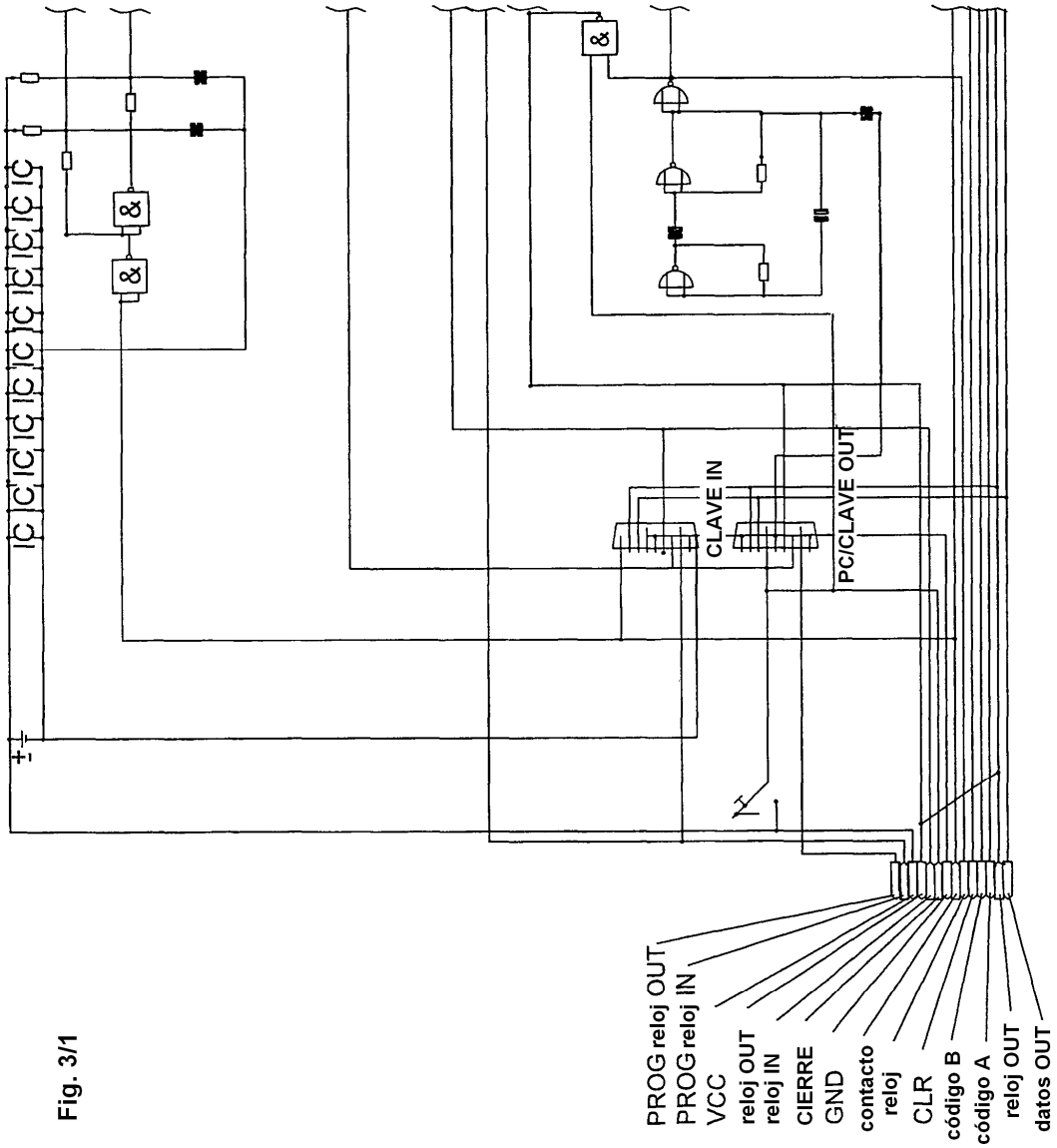
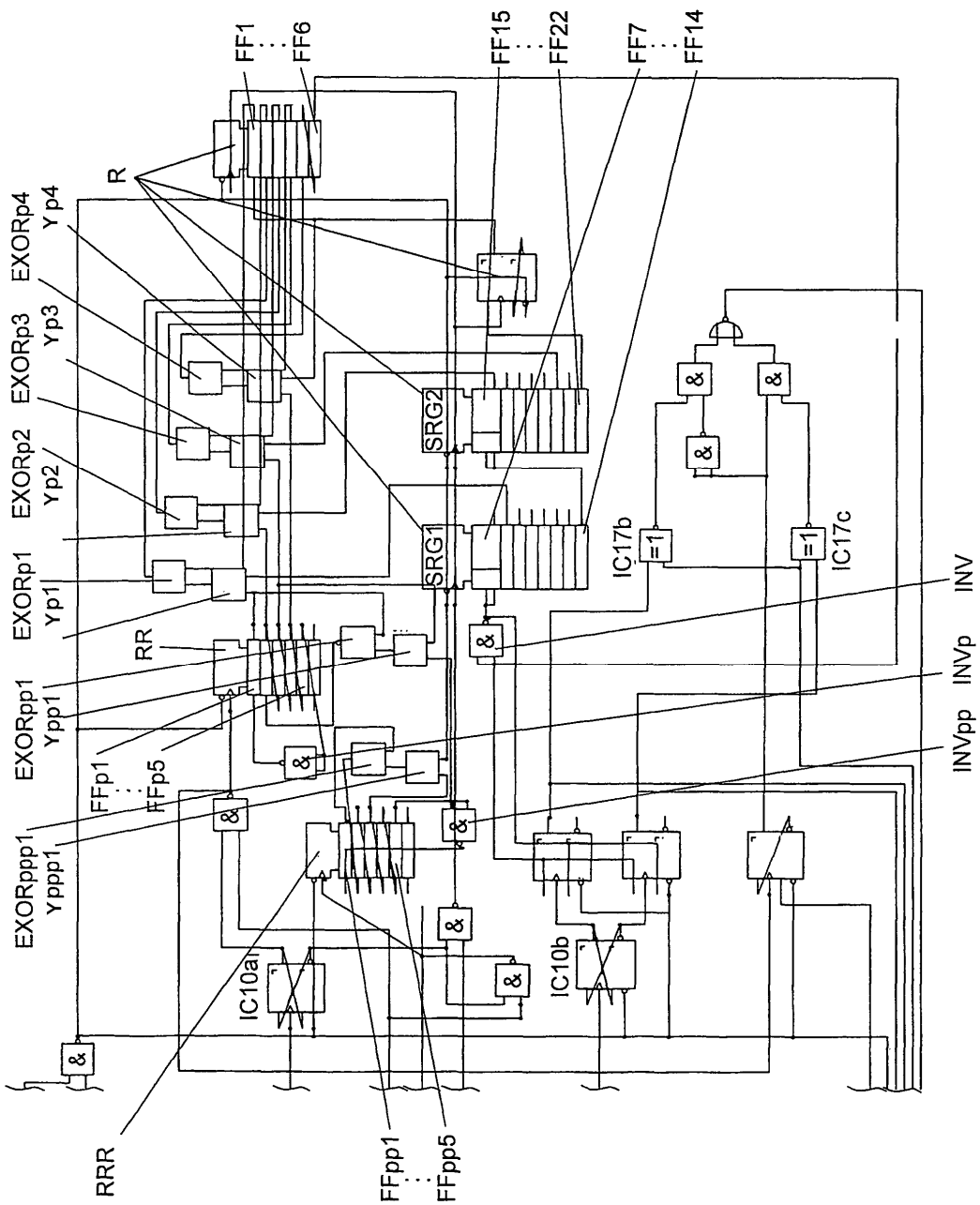

&
&
&
IC IC IC IC IC IC IC IC IC IC
CLAVE IN
PC/CLAVE OUT
PROG reloj OUT
PROG reloj IN
VCC
reloj OUT
reloj IN
CIERRE
GND
contacto reloj
CLR
código B
código A
reloj OUT
datos OUT

Fig. 3/1

Fig. 3/2


FF1
FF6
FF15
FF22
FF7
FF14
R
EXORp4
Yp4
EXORp3
Yp3
EXORp2
Yp2
EXORp1
Yp1
RR
EXORpp1
Ypp1
FFp1
FFp5
EXORppp1
Yppp1
SRG2
SRG1
IC17b
=1
IC17c
INV
INVp
INVpp
IC10a
IC10b
RRR
FFpp1
FFpp5
&