(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 552 305**

(51) Int. CI.:

G06F 11/10 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **10.08.2012 E 12759498 (4)**

(97) Fecha y número de publicación de la concesión europea: **07.10.2015 EP 2748707**

(54) Título: **Corrección de errores digitales**

(30) Prioridad:

26.08.2011 GB 201114831
09.03.2012 US 201261608694 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
27.11.2015

(73) Titular/es:

OXFORD BROOKES UNIVERSITY (100.0%)
Gipsy Lane Campus, Headington, Oxford
Oxfordshire OX3 0BP, GB

(72) Inventor/es:

POOLAKKAPARAMBIL, MAHESH;
JABIR, ABUSALEH;
MATHEW, JIMSON y
PRADHAN, DHIRAJ K.

(74) Agente/Representante:

ISERN JARA, Jorge

DESCRIPCIÓN

Corrección de errores digitales

La presente invención se refiere a la corrección de errores. Tiene aplicación en el diseño de circuitos tolerantes de errores, por ejemplo, en circuitos para realizar operaciones aritméticas, pero también en otros tipos de circuitos. Algunos aspectos de la invención también tienen aplicación en otras áreas, tales como la corrección de errores en diseños de comunicaciones y memorias.

Los circuitos digitales modernos son cada vez más grandes y complejos y, por lo tanto, son cada vez más susceptibles a errores por una diversidad de razones. Por ejemplo, la escala decreciente de los circuitos y las tensiones más bajas que se usan para representar datos, generalmente aumentan las posibilidades de errores. Los errores pueden producirse, por ejemplo, como resultado de partículas energéticas en los entornos donde se usan los circuitos, causando cambios de bits en el circuito, o simplemente como resultado de errores de fabricación. Además, el bombardeo de los circuitos con partículas energéticas en intentos deliberados de inducir errores en los circuitos puede ser cada vez más una fuente de errores, particularmente en criptografía. Por lo tanto, la detección y/o la corrección de errores en circuitos digitales es cada vez más importante.

Como un ejemplo, se usa aritmética de campos finitos, tal como aritmética sobre los campos de Galois $GF(2^k)$, en numerosas aplicaciones, incluyendo criptografía. En la criptografía, por ejemplo, se desean circuitos tolerantes a fallos, entre otras razones porque es posible aprender información secreta que causa fallos en un circuito mientras que se realizan operaciones criptográficas. Para conseguir circuitos tolerantes a fallos para criptografía, se han propuesto circuitos de multiplicación de campo finito con capacidades de detección y corrección de errores concurrentes. Sin embargo, estos tienen a ser muy grandes, o se corrigen únicamente errores individuales.

MATHEW, Y COL: "Fault-tolerant bit parallel finite field multipliers using LDPC codes", CIRCUITS AND SYSTEMS 2008, ISCAS 2008, IEEE INTERNATIONAL SYMPOSIUM ON, IEEE, PISCATAWAY, NJ, Estados Unidos, 18 de mayo 2008 (2008-05-18), XP031392315, ISBN: 978-1-4244-1683-7 y MAHESH POOLAKKAPARAMBIL: "BCH code based multiple bit error correction in finite field multiplier circuits", QUALITY ELECTRONIC DESIGN (ISQED), 2011, 12TH INTERNATIONAL SYMPOSIUM ON, IEEE, 14 de marzo de 2001 (2011-03-14), XP031869298, ISBN: 978-1-61284-913-3 desvelan circuitos de multiplicador con corrección de error de bit.

La invención, que se define en detalle en la reivindicación independiente 1 adjunta, proporciona un circuito corrector de errores, dispuesto para recibir una señal de reloj, que comprende: un componente dispuesto para generar una primera salida a partir de una primera entrada y una segunda entrada; un detector de errores dispuesto para generar un indicador de errores indicativo de si ha detectado o no un error en la primera salida, basándose en la primera salida, la primera entrada y la segunda entrada; un generador de corrección adecuado para generar una salida de corrección después de un primer periodo de tiempo que comienza con un evento de temporización en la señal de reloj, basándose en la primera salida, la primera entrada y la segunda entrada; y un generador de salida dispuesto para generar una salida del circuito corrector de errores después de un segundo periodo de tiempo que comienza con el evento de temporización en la señal de reloj. Si el indicador de errores indica que se ha detectado un error en la primera salida entonces el segundo periodo de tiempo es mayor que el primer periodo de tiempo. De lo contrario, el segundo periodo de tiempo no es mayor que el primer periodo de tiempo. Si el indicador de errores indica que se ha detectado un error en la primera salida, entonces la salida del circuito corrector de errores comprende una combinación de la primera salida y la salida de corrección, con lo que el error detectado en la primera salida se corrige. De otro modo, la salida del circuito corrector de errores corresponde directamente a la primera salida.

Pueden usarse los circuitos de acuerdo con la invención en aplicaciones de comunicaciones, memoria y otras, así como un diseño de circuito tolerante a errores. Pueden usarse para aumentar la velocidad a la que se procesan los datos, por ejemplo, mediante sistemas criptográficos, de comunicaciones o de memoria.

Opcionalmente, el generador de salida comprende un registro de salida que tiene una salida, siendo la salida del registro de salida la salida del circuito corrector de errores, y en el que el generador de salida se dispone para retrasar el registro de salida en la actualización de su salida cuando el indicador de errores indica que se ha detectado un error en la primera salida, causando así que el segundo periodo de tiempo sea mayor que el primer periodo de tiempo.

Opcionalmente, el generador de salida comprende un componente de autorización de salida dispuesto para generar un reloj controlado basándose en la señal de reloj y el indicador de errores, y en el que el registro de salida recibe el reloj controlado en una entrada de reloj del mismo, evitando así que el registro de salida actualice su salida cuando el indicador de error indica que se ha detectado un error en la primera salida.

Por ejemplo, la señal de reloj recibida puede tener un periodo constante que es menor que el segundo periodo de tiempo; el reloj controlado tendrá aproximadamente el mismo periodo que la señal de reloj recibida hasta que se ha

detectado un error en la primera salida, punto en el que al menos un periodo del reloj controlado se extiende para permitir un tiempo adicional para el generador de corrección para generar una salida de corrección apropiada.

Opcionalmente, el circuito comprende adicionalmente un generador de bits de verificación, en el que el generador de bits de verificación se dispone para generar, basándose en la primera entrada y en la segunda entrada, al menos un bit de verificación, y en el que el detector de errores y el generador de corrección se disponen para generar el indicador de errores y la salida de corrección, respectivamente, basándose en la primera salida y dicho al menos un bit de verificación.

Opcionalmente, el detector de errores se dispone para generar el indicador de errores basándose en el primer resultado y el al menos un bit de verificación, siendo el indicador de errores indicativo de si el detector de errores ha detectado uno cualquiera de una pluralidad de errores diferentes que el detector de error se dispone para detectar, comprendiendo la pluralidad de errores diferentes un error en la primera salida y un error en el al menos un bit de verificación.

A diferencia de la técnica anterior, ventajosamente, pueden detectarse los errores en el generador de bits de verificación.

Opcionalmente, el generador de corrección es adecuado para generar la salida de corrección basándose en la primera salida y el al menos un bit de verificación, siendo la salida de corrección adecuada para corregir uno cualquiera de una pluralidad de errores diferentes, comprendiendo la pluralidad de errores diferentes un error en la primera salida y un error en el al menos un bit de verificación.

A diferencia de la técnica anterior, ventajosamente, los errores en el generador de bits de verificación pueden corregirse.

Opcionalmente, el generador de bits de verificación se dispone para generar dicho al menos un bit de verificación directamente a partir de la primera entrada y la segunda entrada, sin generar por separado la primera salida.

Opcionalmente, el generador de corrección se dispone para generar la salida de corrección generando un polinomio de localización de errores y después buscando raíces del polinomio de localización de errores, en el que el generador de corrección busca únicamente raíces correspondientes a la primera salida.

Opcionalmente, el componente dispuesto para generar la primera salida se dispone para generar la primera salida realizando una operación aritmética en la primera y segunda entradas.

Opcionalmente, la operación aritmética es una operación aritmética de campo finito, tal como una multiplicación sobre campos de
Galois $GF(2^k)$.

En cualquiera del primer y segundo aspectos de la invención, la salida puede comprender una pluralidad de bits y el generador de corrección puede disponerse para asignar los bits de salida a un primer conjunto de grupos y realizar una primera etapa de detección de errores en cada uno del primer conjunto de grupos, asignar los bits de salida a un segundo conjunto de grupos y realizar una segunda etapa de detección de errores en cada uno del segundo conjunto de grupos, usar los resultados de la primera y segunda etapas de detección de errores para realizar una etapa de localización de errores para localizar errores en la salida.

El circuito de corrección de error puede disponerse para recibir una salida del sistema que comprende una pluralidad de bits de salida, asignar los bits de salida a un primer conjunto de grupos y realizar una primera etapa de detección de error en cada uno del primer conjunto de grupos, asignar los bits de salida a un segundo conjunto de grupos y realizar una segunda etapa de detección de errores en cada uno del segundo conjunto de grupos, usar los resultados de la primera y segunda etapas de detección de errores para realizar una etapa de localización de error con el fin de localizar errores en la salida, y para generar una salida corregida a partir de la salida recibida y el resultado de la etapa de localización de errores.

Algunos circuitos de acuerdo con la invención tendrán el beneficio de que la corrección de errores puede incorporarse a los circuitos con un gasto de espacio relativamente bajo, ya que la corrección de errores puede realizarse de una manera generalmente eficiente, y puede corregir un número relativamente alto de errores de bit.

Opcionalmente, el primer y segundo conjuntos de grupos se disponen de tal forma que, para cualquiera de los bits de salida, la identidad del grupo, del primer conjunto, del cual es un miembro y la identidad del grupo, del segundo conjunto, del cual es un miembro, identifican únicamente el bit de salida.

Por ejemplo, si los bits de salida se disponen en una tabla rectangular de filas y columnas, el primer conjunto de grupos puede comprender las filas, y el segundo conjunto de grupos puede comprender las columnas, o viceversa. Sin embargo, se apreciará que los bits en la salida pueden asignarse a las posiciones en la tabla en cualquier orden.

El circuito puede comprender un subcircuito dispuesto para realizar cada una de las etapas, por ejemplo, puede incluir uno cualquiera o más de: un subcircuito de asignación, un primer subcircuito de detección de errores, un segundo subcircuito de detección de errores, un subcircuito de localización de errores, y un subcircuito de corrección.

Las etapas de detección de errores, o subcircuitos, pueden disponerse cada uno para determinar el número de errores de bit en el grupo relevante. Por ejemplo, pueden disponerse para generar un código de detección de errores, por ejemplo, un código de paridad, para cada grupo.

Los grupos en el primer conjunto pueden tener todos el mismo tamaño, o pueden ser de tamaños diferentes. Los grupos en el segundo conjunto pueden ser todos del mismo tamaño, o pueden ser de tamaños diferentes. Los grupos del primer conjunto pueden ser del mismo tamaño, o de un tamaño diferente de los grupos en el segundo conjunto.

El sistema puede ser un circuito funcional, tal como un multiplicador, en cuyo caso, la salida puede ser el resultado de una función realizada en una o más entradas. En otros casos, el sistema puede ser un canal por el que se transmite
una comunicación, en cuyo caso, la salida puede ser una comunicación según se recibe del canal. En otros casos, el sistema puede ser un circuito de memoria, en cuyo caso, la salida pueden ser datos recuperados del circuito de memoria.

El circuito puede comprende adicionalmente, en cualquier combinación, una cualquiera o más características de una o más de las realizaciones preferidas de la invención, que se describirán ahora, únicamente a modo de ejemplo, con referencia a los dibujos adjuntos, en los que:

La figura 1 es un diagrama de bloques esquemático de un circuito, de acuerdo con una primera realización, para computar multiplicaciones de campo finito y que tiene capacidades de corrección de errores concurrentes;

la figura 2 es un diagrama de bloques esquemático que muestra componentes del bloque del generador de corrección de la figura 1;

la figura 3 es un diagrama de bloques esquemático de un circuito, de acuerdo con una segunda realización, para computar multiplicaciones de campo finito y que tiene capacidades de corrección de errores concurrentes; y

la figura 4 es un diagrama de bloques esquemático que muestra componentes del subcircuito de detección de errores de la figura 3;

la figura 5 es una línea temporal que muestra algunos periodos de tiempo representativos de los retrasos de propagación respectivos a través de diferentes componentes del circuito de la figura 3;

la figura 6 es un diagrama de temporización que muestra algunas de las señales del circuito de la figura 3;

la figura 7 es un diagrama de bloques esquemático de un circuito de corrección de errores de acuerdo con una tercera realización de la invención;

la figura 8 es una tabla que muestra bits de verificación generados en el circuito de la figura 7;

las figuras 9a, 9b, 9c y 9d son tablas que muestran algunos ejemplos de combinaciones de errores que pueden corregirse usando el circuito de la figura 7;

la figura 10 es una tabla que muestra, para una modificación de la realización de la figura 7, algunos ejemplos de combinaciones de errores que pueden corregirse;

la figura 11 es una tabla que muestra algunos ejemplos de combinaciones de errores que pueden detectarse en una realización adicional de la invención;

la figura 12 es un gráfico que muestra el área de circuitos de multiplicador de diferentes tamaños usando tecnologías de 180 nm y 90 nm;

la figura 13 es un gráfico que muestra el área de circuitos de detección y corrección de errores para multiplicadores de diferentes tamaños usando tecnologías de 180 nm y 90 nm que usan códigos de Hamming para la corrección de múltiples errores en cada fila de la tabla de bits;

la figura 14 es un gráfico que muestra el área de circuitos de detección y corrección de error para multiplicadores de diferentes tamaños usando tecnologías de 180 nm y 90 nm usando códigos BCH; y

las figuras 15 y 16 muestran el consumo de energía de los circuitos de multiplicador con la detección y corrección de errores de las figuras 13 y 14.

Haciendo referencia a la figura 1, en la primera realización de la invención, un circuito 100 incluye un subcircuito de multiplicación de campos finitos 105 con dos entradas paralelas 105a, 105b y una salida paralela 105c. El subcircuito de multiplicación 105 se dispone para generar un producto C en la salida 105c. El producto C es el resultado de multiplicar los dos operandos A,B, recibidos en las dos entradas paralelas 105a, 105b, por un campo de Galois $GF(2^k)$ del cual los dos operandos son elementos. Las dos entradas paralelas 105a, 105b y la salida paralela 105c tienen k bits de ancho, es decir, cada una consiste en k bits.

En otras realizaciones, el subcircuito de multiplicación de campos finitos 105 puede reemplazarse por circuitos para realizar otra aritmética de campo finito, tal como operaciones de inversión o exponenciación multiplicativas (por ejemplo, cuadratura). En algunas realizaciones, el subcircuito de multiplicación de campos finitos 105 puede reemplazarse por circuitos para realizar una aritmética distinta de la aritmética de campo finito, tal como una aritmética binaria de complemento a dos, por ejemplo.

En la primera realización, el circuito 100 también incluye un generador de bits de verificación 110 con dos entradas paralelas de *k* bits 110a, 110b y una salida paralela de n *k* bits 110c, que puede ser una salida de un único bit en algunas realizaciones.

El generador de bits de verificación 110 se dispone para recibir los mismos dos operandos A,B que se reciben en el subcircuito de multiplicación 105, y para generar una palabra de paridad P en la salida paralela 110c. En algunas realizaciones, la palabra de paridad P puede reemplazarse por un bit de paridad.

El generador de bits de verificación 110 comprende una lógica para generar una palabra de paridad P en la salida paralela 110c. La palabra de paridad P se genera realizando una combinación de multiplicación de campo finito y codificación BCH en los dos operandos A,B recibidos en las dos entradas paralelas 110a, 110b. La lógica del generador de bits de verificación 110 se dispone para generar la palabra de paridad P directamente a partir de los dos operandos A,B, en lugar de generar un resultado de multiplicación primer y después generar la palabra de paridad P a partir de ese resultado de multiplicación, lo que los inventores descubrieron que era más eficaz (por ejemplo, un retraso más pequeño y/o menos puertas lógicas). El generador de bits de verificación 110 no transmite un resultado de multiplicación correspondiente al producto C.

En esta realización, la lógica del generador de bits de verificación 110 se derivó sustituyendo una expresión convencional que definía una multiplicación por los campos de Galois $GF(2^k)$ - el mismo tipo de multiplicación que se realiza por el subcircuito de multiplicación de campos finitos 105 - en una expresión convencional que definía los bits de paridad de un código BCH binario (*n, k, t*) (donde *k* es el número de bits en el "mensaje", por ejemplo, en cada uno de los dos operandos A,B y en el producto C, *t* es el número de errores que el circuito 100 siempre puede corregir, dondequiera que ocurran (un mayor número de errores puede corregirse con frecuencia, pero no en todos los casos), y *n-k* es el número de bits en la palabra de paridad P).

El circuito 100 incluye adicionalmente un generador de corrección 115 con dos entradas 115a, 115b y una salida 115c. Las dos entradas 115a, 115b están conectadas a, y tienen la misma anchura que, la salida 105c del subcircuito de multiplicación 105 y la salida 110c del generador de bits de verificación 110 respectivamente. La salida 115c es una salida paralela de *k* bits.

El generador de corrección 115 se dispone para recibir el producto C y la palabra de paridad P en sus dos entradas 115a, 115b, y para generar en su salida 115c un valor de corrección E. En esta realización, el generador de corrección 115 comprende una lógica para generar el valor de corrección E realizando la decodificación BCH en el producto C y la palabra de paridad P, como se describirá en más detalle a continuación con referencia a la figura 2.

El circuito 100 incluye adicionalmente un subcircuito de combinación 120 con dos entradas paralelas de k bits 120a, 120b, que se conectan a la salida 105c del subcircuito de multiplicación 105 y la salida 115c del generador de corrección 115 respectivamente, y una salida paralela de *k* bits 120c.

El subcircuito de combinación 120 se dispone para recibir el producto C y el valor de corrección E, y para generar en la salida paralela 120c un producto corregido C'. En esta realización, el subcircuito de combinación 120 consiste en una pluralidad de puertas XOR (no mostradas), una respectiva para cada bit de la salida paralela 120c. Cada bit del

producto C se combina a través de una operación or exclusiva (por una respectiva de las puertas XOR) con un bit correspondiente del valor de corrección E, lo que hace que los bit erróneos del producto C se inviertan (es decir, una lógica 0 se invierte a una lógica 1 y viceversa) en el producto corregido C', corrigiendo así los bits erróneos del producto C.

Haciendo referencia a la figura 2, el generador de corrección 115 comprende un generador de síndrome 125 y un localizador de errores 130, y genera en su salida 115c un valor de corrección de k bits E (mostrados individualmente como e_{k-1}... e_0 en la figura 2).

El generador de síndrome 125 tiene dos entradas 125a, 125b y una salida paralela de t bits de ancho 125c. Las dos entradas 125a, 125b se conectan a, y tienen la misma anchura que, las dos entradas paralelas 115a, 115b del generador de corrección 115 respectivamente. El generador de síndrome 125 se dispone para recibir el producto C y la palabra de paridad P y para generar en su salida 125c un polinomio de localización de error St...S1. El generador de síndrome 125 se dispone para generar el polinomio de localización de error St...S1 usando el algoritmo Peterson-Gorenstein-Zierler ya conocido, pero otros métodos adecuados de generación del polinomio de localización de errores St...S1 serán evidentes para los expertos en la técnica y pueden usarse en otras realizaciones.

El localizador de error 130 tiene una entrada paralela de t bits de ancho 130a, que se conecta a la salida del generador de síndrome 125, y tiene una salida paralela de k bits de ancho 130c conectada a la salida 115c del generador de corrección 115.

El localizador de errores 130 se dispone para recibir el polinomio de localización de errores St... S1 y para generar en su salida 130c un valor de corrección E, que consiste en localizadores de errores k $e_{k-1}...e_0$. En esta realización, los localizadores de errores se generan encontrando las raíces del polinomio de localización de errores St... S1 usando el algoritmo de búsqueda de Chien ya conocido. Ventajosamente, el localizador de errores 130 no busca las raíces correspondientes a la palabra de paridad P (ya que en esta realización no se desea la corrección de la palabra de paridad P), que tiende a reducir la cantidad de lógica en el localizador de errores 130 en comparación con la cantidad que se requerirá si se buscaron las raíces correspondientes a la palabra de paridad P. En otras realizaciones, el localizador de errores 130 puede buscar las raíces correspondientes a la palabra de paridad P. En dichas realizaciones, la salida del generador de corrección puede ser de $t+k$ bits de ancho, y el generador de corrección puede disponerse para generar en su salida un valor de corrección adecuado para corregir una cualquiera de una pluralidad de errores diferentes, incluyendo la pluralidad errores en la palabra de paridad y errores en el producto.

Haciendo referencia a la figura 3, en la segunda realización de la invención, un circuito 200 incluye un subcircuito de multiplicación de campos finitos 205, un generador de bits de verificación 210, un generador de corrección 215 y un subcircuito de combinación 220. Cada uno de estos componentes es sustancialmente el mismo que el componente correspondiente (del mismo nombre) de la primera realización. El subcircuito de multiplicación 205, el generador de bits de verificación 210 y el generador de corrección 215 se conectan juntos de la misma manera que en la primera realización; estos componentes no han de describirse adicionalmente.

El circuito 200 también incluye un subcircuito de detección de errores 250, un subcircuito de máscara de bits 255, un subcircuito de autorización de salida 260 y un registro de salida 265.

El subcircuito de detección de errores 250 tiene dos entradas 250a, 250b y una salida de un bit 250c. Las dos entradas 250a, 250b se conectan a, y tienen la misma anchura que, la salida 205c del subcircuito de multiplicación 205 y la salida 210c del generador de bits de verificación 210, respectivamente. El subcircuito de detección de errores 250 está dispuesto para detectar un error en el producto C y para detectar un error en la palabra de paridad P, ya se produzcan los errores individualmente o al mismo tiempo. Por consiguiente, el subcircuito de detección de errores 250 puede detectar una o más de una pluralidad de errores diferentes.

Puesto que el subcircuito de detección de errores 250 se conecta al generador de bits de verificación 210, no necesita generar bits de verificación directamente de los dos operandos A,B. Esto puede verse como una compartición de recursos ventajosa entre el subcircuito de detección de errores 250 y el generador de corrección 215, puesto que se usa un módulo común (el generador de bits de verificación 210) para generar la palabra de paridad P para ambos en lugar de incluir cada uno su propia lógica para derivar la palabra de paridad P directamente de los dos operandos A,B.

Haciendo referencia a la figura 4, el subcircuito de detección de errores 250 comprende un módulo de generación de bits de verificación 251 y un módulo de comparación 252.

El módulo de generación de bits de verificación 251 tiene una entrada paralela de k bits 251a, conectada a una primera de las entradas paralelas 250a del subcircuito de detección de errores 250, y una salida paralela 251c que tiene n k bits de ancho.

El módulo de generación de bits de verificación 251 se dispone para recibir el producto C y para generar en su salida 251c una palabra de paridad adicional P'. La palabra de paridad adicional P' se genera de una manera correspondiente a la codificación BCH usada por el generador de bits de verificación 210 para generar la palabra de paridad P. Por lo tanto, si no hay ningún error presente en la palabra de paridad P y no hay ningún error presente en el producto C, entonces la palabra de paridad adicional P' será igual a la palabra de paridad P.

El módulo de comparación 252 tiene dos entradas 252a, 252b y una salida de un bit 252c conectada a la salida 250c del subcircuito de detección de errores 250. Las dos entradas 252a, 252b se conectan a, y tienen la misma anchura que, la salida 251c del módulo de generación de bits de verificación 251 y la segunda de las entradas paralelas 250b del subcircuito de detección de errores 250, respectivamente.

El módulo de comparación 252 está dispuesto para recibir la palabra de paridad adicional P' y la palabra de paridad P, y para generar en su salida 252c un indicador de errores F. El indicador de errores F es indicativo de si se ha detectado una cualquiera de una pluralidad de errores diferentes, incluyendo la pluralidad un error en el producto C y un error en la palabra de paridad P.

Aunque son posibles otras disposiciones adecuadas, en esta realización el módulo de comparación 252 consiste en un módulo de or exclusiva 253 y un módulo de evaluación 254.

El módulo de o exclusivo 253 tiene dos entradas paralelas 253a, 253b, que tienen cada una *n k* bits de ancho, conectadas a la salida 251c del módulo de generación de bits de verificación 251 y la segunda de las entradas paralelas 250b del subcircuito de detección de errores 250, respectivamente, y una salida paralela 253c de la misma anchura.

El módulo de o exclusivo 253 se dispone para recibir la palabra de paridad P y la palabra de paridad adicional P', y para generar en su salida 253c una tercera palabra de paridad P", realizando una operación de or exclusiva a nivel de bit en la palabra de paridad P y la palabra de paridad adicional P'.

El módulo de evaluación 254 tiene una entrada paralela 254a, que tiene *n k* bits de ancho, conectada a la salida paralela 253c del módulo de o exclusivo253, y una salida de un bit 254c conectada a la salida 252c del módulo de comparación 252.

El módulo de evaluación 254 se dispone para recibir la tercera palabra de paridad P", para evaluarla con el fin de determinar si todos sus bits son de lógica 0, y para generar en su salida 254c el indicador de error F. Si todos los bits de la tercera palabra de paridad P" son cero, entonces el módulo de evaluación 254 ajusta el valor del indicador de errores F a una lógica 1 para indicar que no está presente ningún error en el producto C o en la palabra de paridad P, si no el módulo de evaluación 254 ajusta el valor del indicador de errores F a una lógica 0 para indicar que una de pluralidad de errores se ha detectado.

Haciendo referencia de nuevo a la figura 3, el subcircuito de máscara de bits 255 tiene dos entradas 255a, 255b y una salida paralela de k bits 255c. Las dos entradas 255a, 255b se conectan a, y tienen la misma anchura que, la salida 250c del subcircuito de detección de errores 250 y la salida 215c del generador de corrección 215, respectivamente.

El subcircuito de máscara de bits 255 se dispone para recibir el indicador de errores F y el valor de corrección E y produce, y para generar en su salida 255c un valor de corrección enmascarado E'. El subcircuito de máscara de bits 255 se dispone para ajustar cada bit del valor de corrección enmascarado E' a una lógica 0 si el indicador de errores F se ajusta a una lógica 1 (es decir, si no se ha detectado ningún error), o si no ajusta el valor de cada bit para que sea igual a un bit correspondiente respectivo del valor de corrección E.

Se apreciará que, en efecto, el subcircuito de máscara de bits 255 suprime el valor de corrección E si no se ha detectado ningún error, por ejemplo, en el producto C o en la palabra de paridad P. Aunque son posibles otras disposiciones adecuadas, en esta realización el valor de corrección enmascarado E' es el resultado de una operación AND a nivel de bit realizada sobre el valor de corrección E y la inversa lógica (es decir, un lógica 1 se convierte en una lógica 0 y viceversa) del indicador de errores F. El subcircuito de combinación 220 tiene dos entradas paralelas de k bits 220a, 220b, que se conectan a la salida 205c del subcircuito de multiplicación 205 y la salida 255c del subcircuito de máscara de bits 255 respectivamente, y una salida paralela de k bits 220c.

El subcircuito de combinación 220 se dispone para recibir el valor de corrección enmascarado E' y el producto C, y para generar en su salida 220c un producto corregido C'. Aunque son posibles otras disposiciones adecuadas, en esta realización el subcircuito de combinación 220 consiste en una pluralidad de puertas XOR (no mostradas), una respectiva para cada bit de la salida 220c. Cada bit del producto C se combina a través de una operación XOR (por una respectiva de las puertas XOR) con un bit correspondiente del valor de corrección enmascarado E'.

Si el indicador de errores F se ajusta a una lógica 0 (es decir, si se ha detectado un error), el valor del valor de corrección enmascarado E' será igual al del valor de corrección E. Por lo tanto, la operación XOR realizada en el producto C y el valor de corrección enmascarado E' puede hacer que se corrijan los bits erróneos del producto C en el producto corregido C'.

Si el indicador de errores F se ajusta a una lógica 1, el valor del producto corregido C' será el mismo que el del producto C, ya que cada bit del valor de corrección enmascarado E' se ajustará a una lógica 0.

El subcircuito de autorización de salida 260 tiene dos entradas de un bit 260a, 260b, conectadas al reloj del circuito y a la salida 250c del subcircuito de detección de errores 250, y una salida de un bit 260c.

El subcircuito de autorización de salida 260 se dispone para recibir el indicador de errores F y una señal de reloj CLK, y para producir en su salida 260c una señal de reloj controlada ECLK. Si el indicador de errores F se ajusta a una lógica 0, la señal de reloj controlada ECLK se ajustará a una lógica 0. En esta realización, el subcircuito de autorización de salida 260 consiste en una puerta AND, dispuesta para recibir el indicador de errores F y la señal de reloj CLK en sus entradas 260a, 260b y para generar la señal de reloj controlada ECLK en su salida 260c.

El registro de salida 265 tiene una entrada de k bits 265a conectada a la salida 220c del subcircuito de combinación 220, una entrada de reloj 265b conectada a la salida 260c del subcircuito de autorización de salida 260, y una salida de k bits 265c.

El registro de salida 265 está dispuesto para recibir el producto corregido C' y la señal de reloj controlada ECLK, y para generar en su salida 265c, en respuesta a un evento de temporización (por ejemplo, flanco ascendente o un flanco descendente) de la señal de reloj controlada ECLK, una salida del circuito Cout que corresponde al producto corregido C'.

Con referencia a la línea temporal 500 de la figura 5, el subcircuito de detección de errores 250 genera, en su salida 250c, un indicador de errores F después de un primer periodo de tiempo Tdetect partiendo de un primer flanco de reloj ascendente de la señal de reloj CLK (o cualquier otro evento de temporización en la señal de reloj CLK, tal como un flanco descendente). Antes del final del primer periodo de tiempo Tdetect, el subcircuito de multiplicación 205 habrá generado el producto C en su salida 205c.

La señal de reloj CLK tiene un periodo de reloj Tclock que es mayor que el primer periodo de tiempo Tdetect.

Si el indicador de errores F se ajusta a una lógica 1, que indica que no hay ningún error en el producto C, un flanco ascendente en la señal de reloj CLK (o cualquier otro evento de temporización en la señal de reloj CLK) recibido en la salida 260b del subcircuito de autorización de salida 260 hará que se genera un valor de lógica 1 en la salida 260c del subcircuito de autorización de salida 260, por ejemplo, causará un flanco ascendente (u otro evento de temporización correspondiente al evento de temporización en la señal de reloj CLK) en la señal de reloj controlada ECLK. Por lo tanto, si el indicador de errores F indica que no se ha detectado ningún error, el registro de salida 265 generará en su salida 265c, después de un periodo aproximadamente igual al periodo de reloj Tclock, una salida de circuito Cout que corresponde al producto C.

El generador de corrección 215 genera, en su salida 215c, una salida de corrección E después de un segundo periodo de tiempo Tcorrect partiendo de un flanco de reloj ascendente de la señal de reloj CLK (o cualquier otro evento de temporización en la señal de reloj CLK). El segundo periodo de tiempo Tcorrect es mayor que el primer periodo de tiempo Tdetect, y mayor que el periodo de reloj Tclock.

En una última parte del segundo periodo de tiempo Tcorrect, que comienza cuando el primer periodo de tiempo Tdetect finaliza y que termina cuando el segundo periodo de tiempo Tcorrect finaliza, el subcircuito de multiplicación 205 habrá generado el producto C en su salida 205c pero el generador de corrección 215 todavía no habrá generado en su salida 215c una salida de corrección E correspondiente. Por lo tanto, cualquier error en el producto C no se corregirá en el producto corregido C' hasta después del segundo periodo de tiempo Tcorrect, ya que el generador de corrección 215 no habrá generado la salida de corrección correspondiente E antes de entonces.

El subcircuito de autorización de salida 260 se dispone para impedir que el registro de salida 265 genere una salida de circuito Cout que corresponde al producto corregido C' hasta después de que se haya corregido un error detectado en el producto C en el producto corregido C'. El error detectado en el producto C hará que el indicador de errores F se ajuste a una lógica 0 antes del final del periodo de reloj Tclock, que, a su vez, ajustará la señal de reloj controlada ECLK a una lógica 0. Por consiguiente, el siguiente flanco ascendente en la señal de reloj CLK (u otro evento de temporización en la señal de reloj CLK) aunque no se propagará a la señal de reloj controlada ECLK en la salida del subcircuito de autorización de salida 260c, que permanecerá en lógica 0 al menos hasta el flanco ascendente posterior (u otro evento de temporización en la señal de reloj CLK) en la señal de reloj CLK. Esto permite al generador de corrección 215 el tiempo que necesita para generar en su salida 215c la salida de corrección E para combinar con el producto C con el fin de corregir el error detectado en la misma.

Puesto que el periodo de reloj Tclock es más corto que el segundo periodo de tiempo Tcorrect, el circuito 200 puede generar potencialmente más resultados de multiplicación en un periodo de tiempo determinado de lo que sería capaz si el periodo de reloj Tclock fuese el mismo o mayor que el segundo periodo de tiempo Tcorrect.

El experto en la técnica esperará que el periodo de reloj Tclock sea el mismo o mayor que el segundo periodo de tiempo Tcorrect, de manera que un error detectado en el producto C pueda corregirse si es necesario en cualquier periodo de reloj. Pero, típicamente, los errores son relativamente raros. Usando un periodo de reloj Tclock que sea más corto que el segundo periodo de tiempo Tcorrect, y evitando que el registro de salida 265 actualice su salida 265c cuando se ha detectado un error en el producto C (con el fin de permitir el tiempo adicional requerido para generar la salida de corrección E), es posible generar más resultados de multiplicación en un periodo de tiempo determinado.

A modo de ejemplo, la operación del circuito 200 se describirá con referencia a la figura 6. Inicialmente, durante dos periodos de la señal de reloj clk, el circuito 200 recibe los valores de los operandos A,B (A1,B1 y A2,B2) a partir de los cuales se generan los valores correctos (C1 y C2) del producto C. El subcircuito de detección de errores 250 ajusta el indicador F a una lógica 0 durante estos dos periodos, para indicar que no se ha detectado ningún error.

Cuando el valor (C3) del producto C está en error, en el tercer periodo de la señal de reloj clk, éste se detecta por el subcircuito de detección de error 250, que ajusta el indicador F a una lógica 0. Puesto que el indicador F se ajusta a una lógica 0, el subcircuito de autorización de salida 260 ajusta el reloj controlado eclk a una lógica 0.

En el siguiente flanco ascendente de la señal de reloj clk, que marca el comienzo del cuarto periodo, el indicador F permanece ajustado a una lógica 0 lo que impide que el reloj controlado eclk se ajuste a una lógica 1 en respuesta a la señal de reloj clk en transición a una lógica 1. Como resultado, el valor erróneo (C3) del producto C no se genera en la salida 265c del registro de salida 265. Esto proporciona al generador de corrección 215 el tiempo que requiere para generar el valor de corrección apropiado E. Posteriormente, y antes del final del cuarto periodo, el circuito de combinación 220 recibe el valor de corrección enmascarado correspondiente E', que combina con el valor erróneo (C3) del producto C para generar el producto corregido C'.

En el siguiente, es decir, quinto, periodo de reloj de la señal de reloj clk, se introducen dos nuevos valores (A4,B4) de los operandos A,B, lo que hace que se genere un producto C que tiene el valor corregido (C4). Puesto que en ese momento no se ha detectado ningún error en el producto C, el subcircuito de detección de error 250 ajusta el indicador F a una lógica 1. Esto hace que el reloj controlado eclk cambie a una lógica 1, ya que la señal de reloj clk se ajusta a una lógica 1. El cambio del reloj controlado eclk, de una lógica 0 a una lógica 1, hace que se genere el valor del producto corregido C' como la salida de circuito Cout en la salida 265c del registro de salida 265.

En el sexto periodo de reloj de la señal de reloj clk, el valor correcto (C4) del producto C se genera como la salida de circuito Cout en la salida 265c del registro de salida 265.

Haciendo referencia a la figura 7, un circuito de corrección de errores de acuerdo con una tercera realización de la invención comprende un bloque o subcircuito funcional 305 que tiene dos entradas paralelas 305a, 305b que tienen una anchura igual m, y una salida 305c también de la misma anchura m. El bloque funcional se dispone para recibir en sus entradas 305a, 305b los operandos respectivos A, B, y realizar una función en los operandos para generar la salida C. En esta realización, la función es una multiplicación y el bloque funcional 305 es un multiplicador que es igual que el de la primera realización. Con el fin de detectar y corregir errores en la operación del bloque funcional 305, el circuito comprende adicionalmente un subcircuito predictor de paridad 310 y un bloque o subcircuito de corrección 315. El predictor de paridad 310 tiene dos entradas paralelas 310a, 310b, cada una de las cuales tiene la misma anchura m que las entradas al bloque funcional, y una salida 310c de anchura k. El predictor de paridad 310 está dispuesto para recibir los operandos A, B en sus entradas y para generar a partir de ellos un código de paridad, que es una predicción de un código de paridad que debería producirse como resultado de una etapa de codificación realizada en la salida del bloque funcional 305, tal como se describirá en más detalle a continuación. El bloque de corrección 315 tiene una primera entrada 315a de la misma anchura m que la salida 305c del bloque funcional 305, y una segunda entrada 315b de la misma anchura k que la salida del predictor de paridad 310, y una salida 315c de la misma anchura m que la salida 305c del bloque funcional 105. El bloque de corrección 315 está dispuesto para recibir en su primera entrada 315a el resultado C transmitido por el bloque funcional 305, y para generar a partir de éste un código de paridad. El predictor de paridad 310 está dispuesto de manera que el código de paridad predicho que genera, si no surgen errores, será el mismo, para cualquier operando dado A, B, que el generado en el bloque de corrección 315. El bloque de corrección se dispone entonces para comparar el código de paridad que recibe del predictor de paridad 310 con el código de paridad que genera del resultado C, y a partir de ellos, detectar y localizar errores en el resultado C, y corregirlos, para generar de este modo una salida corregida C'.

Haciendo referencia a la figura 8, ahora se describirá el código de paridad generado a partir de la salida del bloque funcional en el bloque de corrección 315, y predicho por el predictor de paridad 310. En esta realización, el resultado C transmitido por el bloque funcional es un código de salida de 20 bits que comprende 20 bits C0 a C19. En términos

generales, los códigos de paridad se generan dividiendo los bits del resultado C0 a C19 en un primer conjunto de grupos, en este caso cuatro filas de cinco bits cada una, y generando un código de paridad para cada grupo, o fila, y también dividiendo el resultado en un segundo conjunto de grupos, en este caso cinco columnas de cuatro bits cada una, y generando un código de paridad para cada grupo o columna. El código de paridad para cada uno de los grupos puede detectarse de cualquier manera adecuada, tal como códigos de Hamming o códigos BCH, que pueden usarse para determinar el número de errores en el grupo, pero no su ubicación en el grupo. El código de paridad transmitido por el predictor de paridad comprende un conjunto de códigos de paridad, correspondiendo cada uno a uno de los códigos de paridad generados para uno de los grupos de bits en el bloque de corrección 315. Comparando los códigos de paridad generados a partir de la salida C en el bloque de corrección 315, con los generados en el predictor de paridad 310, el bloque de corrección 315 puede determinar el número de errores en cada uno del primer conjunto de grupos (es decir, cada fila) y el número de errores en cada uno del segundo conjunto de grupos (es decir, en cada columna). A partir de estos números, siempre que el número de errores no sea demasiado alto, puede determinarse la ubicación exacta de los errores. Esto es porque cualquier combinación de una fila y una columna identifica únicamente un bit en la salida. Una vez que los errores se han localizado, el bloque de corrección se dispone para corregirlos, para generar una salida corregida C'.

Se apreciará que el bloque de corrección puede comprender subcircuitos separados dispuestos para realizar cada una de las etapas descritas. En esta realización, comprende un primer subcircuito de asignación 315d dispuesto para asignar los bits de salida al primer conjunto de grupos, un segundo circuito de asignación 315e dispuesto para asignar los bits de salida al segundo conjunto de grupos, un primer subcircuito de generación de códigos de paridad 315f dispuesto para generar el código de paridad para cada uno del primer conjunto de grupos, un segundo subcircuito de generación de códigos de paridad 315g dispuesto para generar los códigos de paridad para cada uno del segundo conjunto de grupos, un primer subcircuito de detección de errores 315h dispuesto para comparar el primer conjunto de códigos de paridad con los códigos correspondientes generados por el predictor de paridad, y para determinar el número de errores en cada uno del primer conjunto de grupos de bits, un segundo subcircuito de detección de errores 315i dispuesto para comparar el segundo conjunto de códigos de paridad con los códigos correspondientes generados por el predictor de paridad, y para determinar el número de errores en cada uno del segundo conjunto de grupos de bits, un subcircuito de localización de errores 315j dispuesto para identificar, a partir de la comparación de ambos conjuntos de códigos de paridad, la localización de errores en la salida C, y un subcircuito de corrección de errores 315k dispuesto para corregir la salida C para generar la salida corregida C'. Sin embargo, se apreciará que cada una de estas funciones puede no realizarse por una parte dedicada separada del circuito, y en otras realizaciones, los subcircuitos pueden disponerse para realizar dos o más de estas funciones en combinación.

Además, se apreciará que, aunque los dos conjuntos grupos pueden visualizarse fácilmente por medio de una tabla rectangular de filas y columnas, la etapa de asignación sólo necesita asignar cada bit a dos grupos: uno del primer conjunto y uno del segundo conjunto. Por ejemplo, para la asignación mostrada en la figura 8, asumiendo que los bits C0 a C19 se disponen en orden número en la salida, los primeros cinco bits pueden asignarse al primer grupo en el primer conjunto (correspondiente a la primera fila) y bloques posteriores de cinco bits pueden asignarse a grupos posteriores en el primer conjunto. Entonces, cada quinto bit que comienza con el primero puede asignarse al primer grupo en el segundo conjunto (correspondiente a la primera columna), cada quinto bit que comienza con el segundo bit puede asignarse al segundo grupo, etc. También se apreciará que los bits que se asignan a qué grupos no es importante. Considerando el formato de la tabla de la figura 8, los 20 bits pueden disponerse de manera dentro de las 20 celdas de la tabla, y los bits aún pueden agruparse según las filas y columnas, aunque esto puede no tener ninguna relación regular con su posición en la salida.

En esta realización, los códigos de paridad se generan usando un código de Hamming simple que puede detectar errores dobles en cada columna, y un código BCH que detectará como mucho 6 errores en cada fila. Para un mejor entendimiento de la codificación a nivel de fila y columna, el procedimiento de codificación de paridad, como se realiza por el bloque de corrección 315, ahora se describirá en más detalle con un circuito ejemplar, considerando un multiplicador de campo finito de bits paralelos de 20 bits como ejemplo del bloque funcional.

Detección de error usando paridad de código Hamming

Los 20 bits se disponen en una tabla de cuatro filas y cinco columnas como se muestra en la figura 8. Los 20 bits de la salida C se identifican en orden como C0 a C19 y cada uno se asigna a una posición en la tabla como se muestra en la figura 8. Las filas se codifican con códigos Hamming, y específicamente, cada fila se codifica con código Ham (9,5). En otras palabras, se requiere una paridad de 4 bits para detectar un error doble (es decir, hasta dos errores de bits) en una fila. La información de paridad de 4 para la primera fila se da por la siguiente expresión:

$$P1 = C0 \cdot C2 \cdot C4 \quad (1)$$

$$P2 = C1 \cdot C2 \cdot C3 \cdot C4 \quad (2)$$

$$P3 = C0 \cdot C3 \cdot C4 \quad (3)$$

$$P4 = C1 \cdot C2 \cdot C4 \quad (4)$$

De forma análoga, cada fila se codifica por separado y se trata como una palabra de código diferente, dando como resultado cuatro códigos de paridad de cuatro bits, uno para cada fila como se muestra en la figura 8. Cada una de las columnas se codifica usando paridad simple. Cada dos bits se protegen generando un bit de paridad de columna CP como se muestra en la figura 8. Las paridades de columna de la primera y segunda columnas se determinan como se muestra en las ecuaciones que se indican a continuación. Las otras tres paridades de columna se generan exactamente de la misma manera que la de C*P0* a C*P3*, como se representa a continuación:

$$CP0 = C0 \cdot C10 \quad (5)$$

$$CP1 = C5 \cdot C15 \quad (6)$$

$$CP2 = C1 \cdot C11 \quad (7)$$

$$CP3 = C6 \cdot C16 \quad (8)$$

El conjunto de ecuaciones de Ec. (1) a Ec. (4), es decir, los códigos de detección de error para las filas, se disponen para detectar los errores de múltiple aparición, hasta dos errores en cada fila. De forma análoga, la Ec. (5) a Ec. (8) que se computa para cada columna puede usarse para detectar la presencia de hasta dos errores en cada columna. A partir de los códigos de detección de errores para las columnas y las filas, siempre que no estén presentes demasiados errores, el bit o bits particulares que están en error pueden detectarse. Por ejemplo, si únicamente está presente un error en el bit C6, solamente se detectará un error en una columna, columna 2, y un error en una fila, fila 2. Por lo tanto, el bit que está tanto en la fila 2 como la columna 2 puede identificarse como el bit que está en error, y corregirse. Este uso de filas y columnas, o más generalmente dos conjuntos diferentes de grupos, se denomina en el presente documento como corrección de error de paridad cruzada. Algunos de los patrones de error que esta técnica puede corregir se muestran en la figura 9.

El predictor de paridad 310 puede duplicar simplemente la estructura del bloque funcional 305 y los subcircuitos en el bloque de corrección 315 que asignan los bits en la salida del bloque funcional a los dos conjuntos de grupos y derivar los códigos de paridad para los grupos. Sin embargo, en esta realización, con el fin de ahorrar espacio y simplificar el circuito, el predictor de paridad está diseñado para derivar los códigos de paridad de la manera más sencilla a partir de las entradas del sistema A, B, sin derivar el resultado C como una etapa intermedia. Puesto que el resultado C no necesita derivarse específicamente, esto puede reducir el tamaño del circuito significativamente.

Corrección de múltiples errores

La descripción anterior explica cómo se detectan los errores tanto en el primer conjunto de grupos (filas) como el segundo conjunto de grupos (columnas). Pero, por supuesto, solamente identificar los errores no es suficiente para corregirlos. Es necesario usar códigos de codificación de errores clásicos, un proceso separado (que necesita una subsección de circuito separada denominada a menudo un decodificador) con el fin de identificar las posiciones de los bits erróneos y corregirlos.

Como se ha indicado anteriormente, en el sistema descrito es posible eliminar los decodificadores complejos usando los "códigos cruzados" bastante sencillos y el bloque de corrección comprende, y se dispone para usar, una lógica *AND-XOR* simple para realizar la corrección. Por ejemplo, haciendo referencia a la figura 9a, en la que los bits se

disponen en una tabla equivalente a la de la figura 8, se supone que los bits C0, C1, C5 y C6 están en error. El sistema puede detectar que hay dos bits erróneos en la fila 1 (C0 y C2) usando el código de Hamming de la fila 1 y, de forma análoga, que hay dos errores en la fila 2 (C5 y C6) se detecta mediante el código de Hamming de la fila 2. Pero estos códigos en solitario pueden determinar únicamente que 2 bits en la fila 1 y la fila 2 están en error, pero no su ubicación. Para descubrir qué bits en cada fila están en error, el sistema se dispone para usar las paridades de columna ya que el bit C0 se protege por CP0, el bit C5 se protege por CP1. De forma análoga, los bits C2 y C7 están protegidos por CP2 y CP3. Usando la combinación de paridad tanto de fila como de columna, el bloque de corrección 315 se dispone para determinar qué bits están en error. De forma similar, si los bits C3 y C18 están en error, estos pueden ambos detectarse mediante este método, como en los errores en los bits C12 y C16. Haciendo referencia a las figuras 9b, 9c y 9d, otros grupos de errores de bit que pueden detectarse son: C1, C2, C5 y C6; C11, C13 y C17; C3, C7 y C9; C2 y C3; C6 y C11; C12 y C14; C0, C1, C2, C5, C6 y C7; C12, C13, C14, C17, C18 y C19.

Haciendo referencia a la figura 10, en una realización adicional, el bloque funcional es un multiplicador de campo finito de 64 bits. En este caso, el bloque de corrección se dispone para definir la tabla como que tiene cuatro filas de 16 bits. En esta realización, el bloque de corrección está dispuesto para usar códigos BCH para detectar el número de errores en cada fila (detección de error de fila) ya que puede detectar más número de errores en cada fila que el código de Hamming. La figura 10 muestra patrones ejemplares de errores en un multiplicador de campo finito de 64 bits que pueden localizarse con decodificación BCH en cada fila y codificación de paridad simple en cada columna. Por ejemplo, con un código BCH(3,1,16), se pueden detectar fácilmente hasta 6 errores por fila que, por lo tanto, aumentan claramente el número de bits que se corrigen en comparación con el simple código de Hamming.

Detección de errores usando paridad de código BCH

El principio básico y diseño de la detección de múltiples errores basada en código BCH de bits paralelos se explicará ahora para el mismo multiplicador de 20 bits que se muestra en la figura 8. Se apreciará por el experto en la técnica que el principio puede extenderse para el multiplicador de 64 bits. Se considerará un simple caso de BCH(15,5,7), donde n = 15 y k = 5. En este ejemplo, se considera un multiplicador PB de bits paralelos sobre GF(25). Se considerará la primera fila de cinco bits como un código BCH. Entonces, como n = 15 y k = 5, se obtiene la siguiente expresión:

$$M(x) = C4x^4+C3x^3+C2x^2+C1x+C0 \quad (9)$$

$$x^{n-k}M(x) = x^{n-k}(C4x^4 + C3x^3 + C2x^2 + C1x + C0)$$

$$= C4x^{14} +C3x^{13} +C2x^{12} +C1x^{11} +C0x10.(10)$$

Los bits de verificación de paridad se generan mediante lo siguiente:

$$P(x) = x^{n-k}M(x) \bmod g(x). \quad (11)$$

Se considerará que el polinomio del generador será g(x) = x10 + x8+x5+x4+x2+x+1. Entonces, la expresión de paridad para la primera fila para la detección de 6 bits será,

$$P(x) = p9x^9 +p8x^8 +p7x^7 +p6x^6 + p5x^5 +p4x^4 +p3x^3 +p2x^2+p1x^1+p0 \ (12)$$

Si se considera un código BCH de corrección de 3 bits, éste puede detectar 6 errores de bit en una única palabra de código. Para detectar múltiples errores en un código de 5 bits, se necesitan diez bits de paridad. Los diez bits de paridad se dan por:

$p0 = c0+c2+c4,$	$p0 = d0+d2+d4+e0+e1+ e2+e3,$
$p1=c0+c1+c2+c3+c4,$	$p1=d0+d1+d2+d3+d4,$
$p2=c0+c1+c3,$	$p2=d0+d1+d3+e1+e2+e3,$
$p3=c1+c2+c4,$	$p3=d1+d2+d4+e0+e2+e3,$
$p4=c0+c3+c4,$	$p4= d0+d3+d4+e0+e2,$
$p5=c0+c1+c2,$	$p5=d0+d1+d2+e2,$
$p6=c1+c2+c3,$	$p6=d1+d2+d3+e0+e3,$
$p7=c2+c3+c4,$	$p7=d2+d3+d4+e1,$
$p8=c0+c2+c3,$	$p8=d0+d2+d3+ e0+e1+e3,$
$p9=c1+c3+c4,$	$p9=d0+d3+d4+e0+e2,$

donde *dx* y ex son términos de producto internos del multiplicador como se define por Reyhani-Masoleh y M. A. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2m)," IEEE Trans. Computers, vol. 53, Nº 8, págs. 945-959, 2004.

El patrón de ejemplos para el código de paridad cruzada basado en código BCH es como se muestra en la figura 11. El sistema se dispone para usar el código BCH de detección de 6 errores de bit en cada fila de 16 bits. En cada columna de 4 bits, se dispone usar códigos de paridad simple, tal como en el caso del esquema basado en hamming. Por lo tanto, puede detectar 2 errores en cada columna y 6 errores en cada fila. Esto significa que la técnica puede corregir hasta 12 errores de bit determinados. Algunos de los ejemplos del patrón se destacan con colores en la figura 11. Los patrones similares indican el error múltiple en el mismo grupo.

Códigos cruzados en multiplicadores de dígitos en serie

El esquema de paridad cruzada propuesto ahora se considerará para un multiplicador más práctico, tal como un multiplicador a nivel de palabra o un multiplicador de dígitos en serie. Con fines experimentales, se ha considerado un multiplicador de dígitos en serie de 163 bits que es el multiplicador de tamaño estándar para operaciones ECC seguras configuradas por NIST y FIPS. Se cree que es el primer intento de sintetizar un multiplicador de dígitos en serie de corrección de múltiples errores de 163 bits. Esto se debe a que las técnicas de detección y corrección de errores conocidas se adaptan mejor a multiplicadores de bits en paralelo ya que proporcionan un gasto de área enorme debido a la parte de detección, decodificación y corrección de error completa en paralelo que transcurre en paralelo a la lógica del multiplicador real.

Haciendo referencia a la figura 12, la complejidad del esquema propuesto se evalúa para tal arquitectura de multiplicador de dígitos en serie para entender mejor los requisitos de espacio para un tamaño de multiplicador de 10 bits, 15 bits, 20 bits, 32 bits, 48 bits, 64 bits y 90 bits.

El circuito de multiplicación de dígitos en serie para este experimento se diseñó usando una arquitectura de multiplicador de acumulador individual. El algoritmo de multiplicación fue como se muestra a continuación:

Entrada: A(x) = ∑m-1

i=0 ai.x^{i}, B(x) = ∑m-1

i=0 bi.x^{i}, P(x).

Salida: C(x) = A (x).B(x)modP(x). Etapa1: C = 0.

Etapa2: para i = 0 a Óm/DÒ – 1 sí Etapa3: C=Bi.A+C.

Etapa4: A = A.á^{D}.

Etapa5: *fin para*

Etapa6: regresar (C mod P(x))

Resultados experimentales

El modelo de conducta del código basado tanto en Hamming como BCH se implementó usando VHDL y se comprobó su exactitud funcional usando el simulador Modelsim. Los esquemas se comprobaron y se verificaron para un multiplicador de bits en paralelo de diversos tamaños, incluyendo estructuras de multiplicador de 10, 15, 20, 32,

48, 64 y 90 bits. Después, los diseños se sintetizar usando el compilador de diseño Synopsys. Se evaluó la variación en área, la potencia de estos diseños usando tecnologías TSMC tanto de 180 nm como de 90 nm.

Análisis de área y energía de la implementación propuesta

La figura 12 muestra los consumos de espacio de los multiplicadores de bits en paralelo de diversos tamaños. Las figuras 13 y 14 muestran el área de bloques de corrección de errores (incluyendo el generador de paridad) en tecnología tanto de 180 como de 90 nm. Resulta obvio a partir de la figura 13 que el consumo de espacio de la técnica basada en BCH es únicamente ligeramente mayor que el del código cruzado basado en Hamming. Esto se debe al hecho de que las secciones de decodificador intensivo de área de ambos códigos se reemplazan por un detector y corrector de error basados en paridad cruzada simple.

El gasto de área del método basado en la paridad cruzada propuesto se representa en la Tabla I. Se observa para el análisis experimental que el gasto de área para los esquemas basados tanto en BCH como en Hamming está notablemente cercano. El gasto de área para un multiplicador de 10 bits muy sencillo es únicamente del 142 %. Según el tamaño del multiplicador cree, el porcentaje de gasto de área debido al circuito de generación de paridad y la lógica de corrección se hace más pequeño y finalmente para un multiplicador de 90 bits que puede corregir múltiples errores es únicamente del 101 %. Es bastante pequeño en comparación con los esquemas de corrección de múltiples errores clásicos basados únicamente en código de corrección de error individual. Aunque el diseño no trata en su totalidad con todos los patrones de error, es poco probable que se produzca un patrón que esté fuera del alcance del esquema propuesto y, por lo tanto, no pueda corregirse. Esto se debe al hecho de que la probabilidad de una interferencia de partícula de radiación que pueda causar un cambio de múltiples bits es, por ejemplo, únicamente de 1 en 1 millón de ciclos de reloj. Por lo tanto, el esquema propuesto puede proporcionar una capacidad de enmascaramiento de error excelente con un gasto de área tan bajo como del 101 %.

TABLA I

COMPARACIÓN DE GASTO DE ÁREA DE DIVERSOS TAMAÑOS DE MULTIPLICADOR		
Nº de bits	Hamming	BCH
10	142 %	160 %
15	123 %	152 %
20	121 %	140 %
32	108 %	120 %
48	105 %	116 %
64	104 %	114 %
90	101 %	106 %

La Tabla II compara nuestro enfoque de código de paridad con otros esquemas de corrección de error disponibles en la bibliografía publicada, en este caso A. Reyhani-Masoleh y M. A. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2m)," IEEE Trans. Computers, vol. 53, Nº 8, págs. 945-959, 2004; J. Mathew, J. Singh, A. M. Jabir, M. Hosseinabady y D. K. Pradhan, "Fault Tolerant Bit Parallel Finite Field Multipliers using LDPC Codes", en Proceedings of the IEEE International Symposium on Circuits and Systems, 2008, págs. 1684-1687; y M. Poolakkaparambil, J. Mathew, A. M. Jabir, D. K. Pradhan y S. P. Mohanty, "BCH Code Based Multiple Bit Error Correction in Finite Field Multiplier Circuits", en Proceedings of the 12th IEEE International Symposium on Quality Electronic Design, 2011, págs. 615-620.

Para una comparación equitativa, se ha usado el multiplicador de 32 bits. Éste muestra que este método puede corregir más número de errores con menor gasto de área en comparación con los otros diseños ya conocidos.

Tabla II

COMPARACIÓN CON OTROS ENFOQUES PARA MULTIPLICADOR DE 32 BITS					
Propiedad	Masoleh y col. 2004	Mathew y col. 2008 [12]	BCH	Paridad Cruzada (Ham)	Paridad Cruzada (BCH)
Nº corrección de errores	individual	individual	3 errores	hasta 6 errores	hasta 12 errores
Técnica de codificación	Hamming	LDPC	BCH Clásico	Hamming + Paridad Simple	BCH + Paridad Simple
Gasto	>100 %	>100 %	150,4 %	108 %	120,4 %

Se ha analizado la disipación de energía del esquema propuesto. La figura 15 y la figura 16 comparan el consumo de energía de los diseños basados tanto en hamming como BCH. Puesto que tienen un gasto de área comparable, la disipación de energía también es bastante similar en los dos esquemas.

Aunque las realizaciones de las figuras 7 a 16 que se han descrito anteriormente se refieren a un diseño de circuito, en otras realizaciones, el sistema es un canal sobre el que se transmite una comunicación, y la salida es la comunicación según se recibe del canal. El predictor de paridad se dispone en el extremo de transmisión del canal y se dispone para generar los códigos de paridad de la comunicación antes de la transmisión, y añadir los códigos de paridad generados para la transmisión con la comunicación. El bloque de corrección se dispone en el extremo receptor del sistema, y se dispone para recibir el mensaje por el canal, junto con los códigos de paridad del predictor de paridad. Después, el bloque de corrección puede realizar las etapas de corrección que se han descrito anteriormente en el mensaje recibido. En otros casos, el sistema puede ser un circuito de memoria, en cuyo caso la salida pueden ser datos recuperados del circuito de memoria, y los códigos de paridad generados por el predictor de paridad pueden almacenarse en la memoria y recuperarse con los datos, de manera que el circuito de corrección pueda realizar las etapas de corrección sobre los datos recuperados.

Las anteriores realizaciones se han descrito únicamente a modo de ejemplo; el alcance de la invención se define por las siguientes reivindicaciones.

REIVINDICACIONES

1. Un circuito corrector de errores, dispuesto para recibir una señal de reloj, que comprende:

un componente (205) dispuesto para generar una primera salida a partir de una primera entrada y una segunda entrada;
un detector de errores (250) dispuesto para generar un indicador de errores (F) indicativo de si ha detectado o no un error en la primera salida, basándose en la primera salida, la primera entrada y la segunda entrada;
un generador de corrección (215) para generar una salida de corrección después de un primer periodo de tiempo que comienza con un evento de temporización en la señal de reloj, basándose en la primera salida, la primera entrada y la segunda entrada; y
un generador de salida (260, 265) dispuesto para generar una salida del circuito corrector de errores después de un segundo periodo de tiempo que comienza con el evento de temporización en la señal de reloj,
en el que si el indicador de errores (F) indica que se ha detectado un error en la primera salida entonces el segundo periodo de tiempo es mayor que el primer periodo de tiempo, o si el indicador de errores (F) indica que se ha detectado un error en la primera salida entonces el segundo periodo de tiempo no es mayor que el primer periodo de tiempo,
y en el que si el indicador de errores (F) indica que se ha detectado un error en la primera salida entonces la salida del circuito corrector de errores comprende una combinación de la primera salida y la salida de corrección por lo que el error detectado en la primera salida se corrige, o si el indicador de errores (F) indica que se ha detectado un error en la primera salida entonces la salida del circuito corrector de error corresponde directamente a la primera salida.

2. Un circuito de acuerdo con la reivindicación 1, en el que el generador de salida comprende un registro de salida (265) que tiene una salida (265c), siendo la salida del registro de salida la salida del circuito corrector de errores, y en el que el generador de salida (260, 265) se dispone para retrasar el registro de salida en la actualización de su salida cuando el indicador de errores indica que se ha detectado un error en la primera salida, causando así que el segundo periodo de tiempo sea mayor que el primer periodo de tiempo.

3. Un circuito de acuerdo con la reivindicación 1 o la reivindicación 2, en el que el generador de salida (260, 265) comprende un componente de autorización de salida (260) dispuesto para generar un reloj controlado basándose en la señal de reloj y el indicador de errores, y en el que el registro de salida (265) recibe el reloj controlado en una entrada de reloj del mismo, retrasando de este modo el registro de salida en la actualización de su salida cuando el indicador de error indica que se ha detectado un error en la primera salida.

4. Un circuito de acuerdo con cualquier reivindicación anterior, que comprende adicionalmente un generador de bits de verificación (210),
en el que el generador de bits de verificación se dispone para generar, basándose en la primera entrada y la segunda entrada, al menos un bit de verificación, y
en el que el detector de errores (250) y el generador de corrección (215) se disponen para generar el indicador de errores y la salida de corrección, respectivamente, basándose en la primera salida y dicho al menos un bit de verificación.

5. Un circuito de acuerdo con la reivindicación 4, en el que el detector de errores (250) está dispuesto para generar el indicador de error basándose en el primer resultado y el al menos un bit de verificación, siendo el indicador de errores indicativo de si el detector de errores (250) ha detectado uno cualquiera de una pluralidad de errores diferentes que el detector de errores se dispone para detectar, comprendiendo la pluralidad de errores diferentes un error en la primera salida y un error en el al menos un bit de verificación.

6. Un circuito de acuerdo con la reivindicación 4 o la reivindicación 5, en el que el generador de corrección (215) se usa para generar la salida de corrección basándose en la primera salida y el al menos un bit de verificación, usándose la salida de corrección para corregir uno cualquiera de una pluralidad de errores diferentes, comprendiendo la pluralidad de errores diferentes un error en la primera salida y un error en el al menos un bit de verificación.

7. Un circuito de acuerdo con una cualquiera de las reivindicaciones 4 a 6, en el que el generador de bits de verificación (210) se dispone para generar dicho al menos un bit de verificación directamente de la primera entrada y la segunda entrada, sin generar por separado la primera salida.

8. Un circuito de acuerdo con cualquier reivindicación anterior, en el que el generador de corrección (215) se dispone para generar la salida de corrección generando un polinomio de localización de errores y después buscando raíces del polinomio de localización de errores, en el que el generador de corrección (215) busca únicamente raíces correspondientes a la primera salida.

9. Un circuito de acuerdo con cualquier reivindicación anterior, en el que el componente (205) dispuesto para generar la primera salida se dispone para generar la primera salida realizando una operación aritmética en la primera y segunda entradas.

10. Un circuito de acuerdo con la reivindicación 9, en el que la operación aritmética es una operación aritmética de campo finito, tal como una multiplicación sobre un campo de Galois $GF(2^k)$.

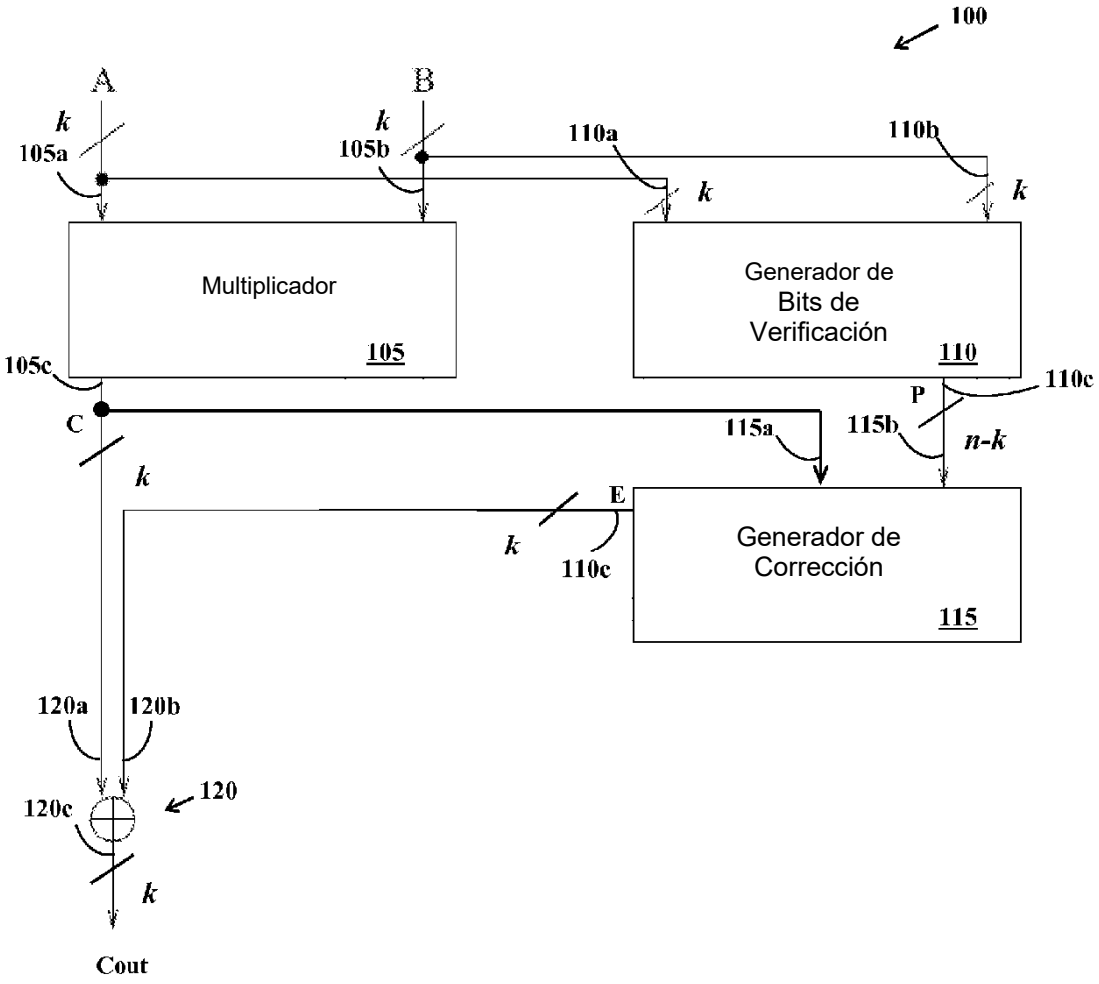

100
A
B
k
105a
k
105b
110a
110b
k
k
Multiplicador
105
Generador de
Bits de
Verificación
110
105c
110c
P
C
115a
115b
n-k
k
E
k
110c
Generador de
Corrección
115
120a
120b
120c
120
k
Cout

Figura 1

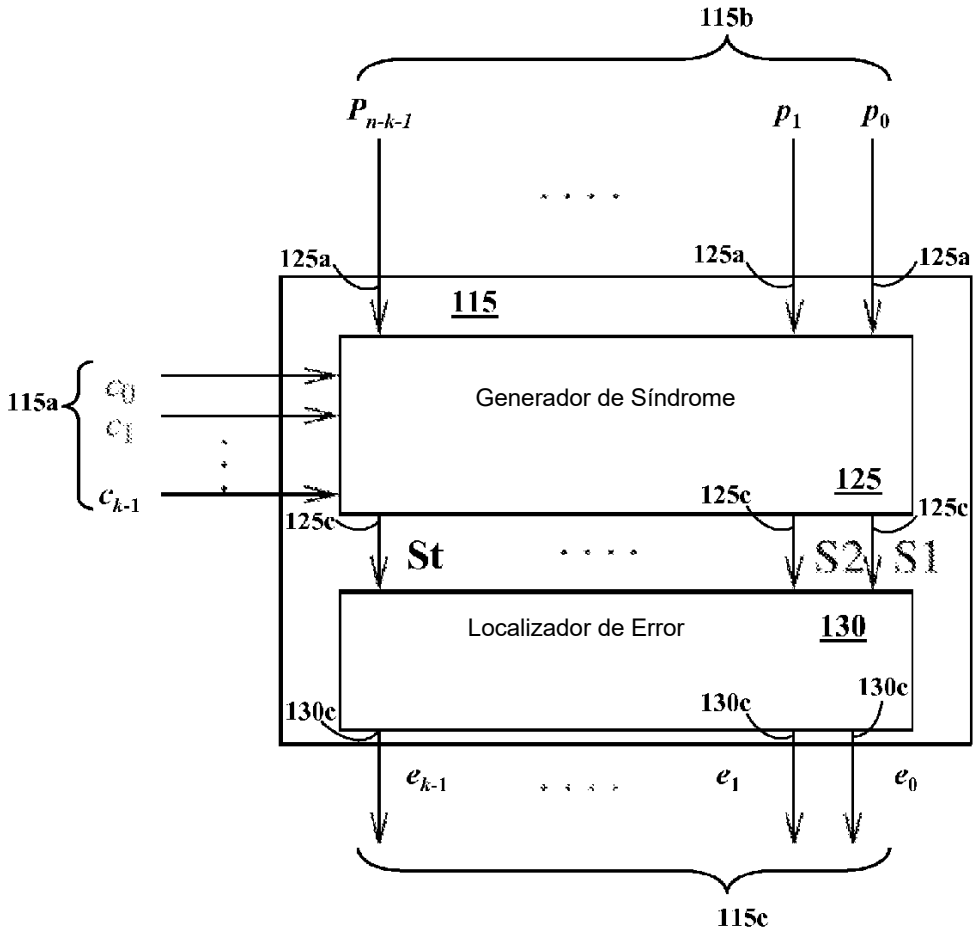

115b
P_{n-k-1}
p_1
p_0
125a
125a
125a
115
115a
c_0
c_1
c_{k-1}
Generador de Síndrome
125
125c
125c
125c
St
S2
S1
Localizador de Error
130
130c
130c
130c
e_{k-1}
e_1
e_0
115c

Figura 2

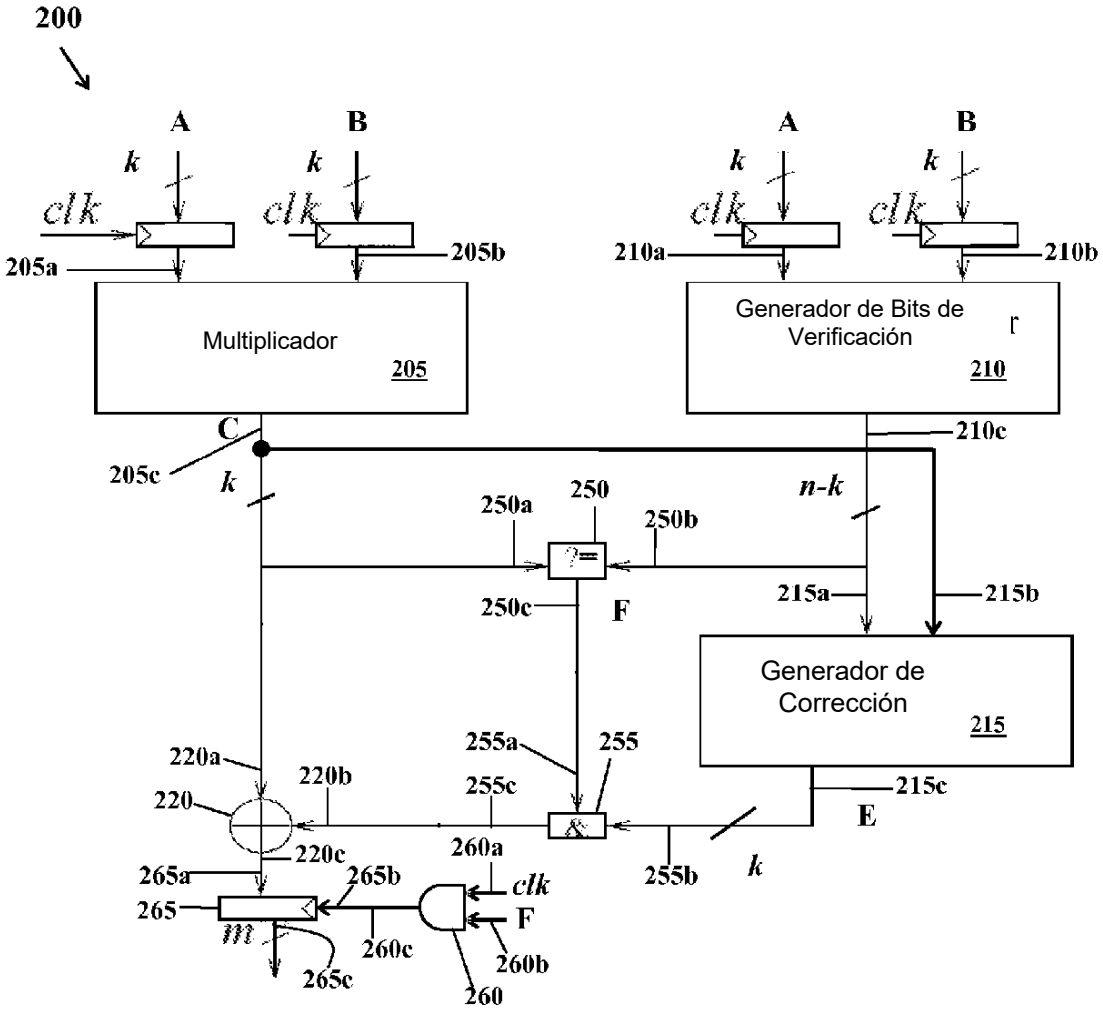

200
A
B
k
clk
205a
205b
Multiplicador
205
C
205c
k
A
B
210a
210b
Generador de Bits de Verificación
210
r
210c
n-k
250a
250
250b
250c
F
215a
215b
Generador de Corrección
215
215c
E
255a
255
255c
255b
220a
220b
220
220c
260a
265a
265
265b
260c
265c
m
clk
F
260b
260

Figura 3

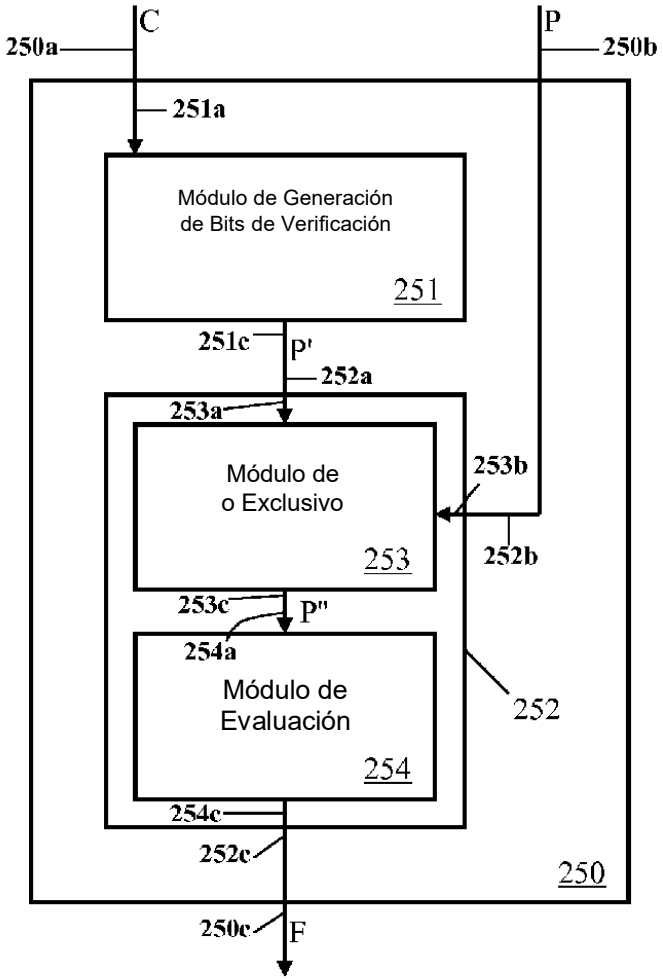


Figura 4

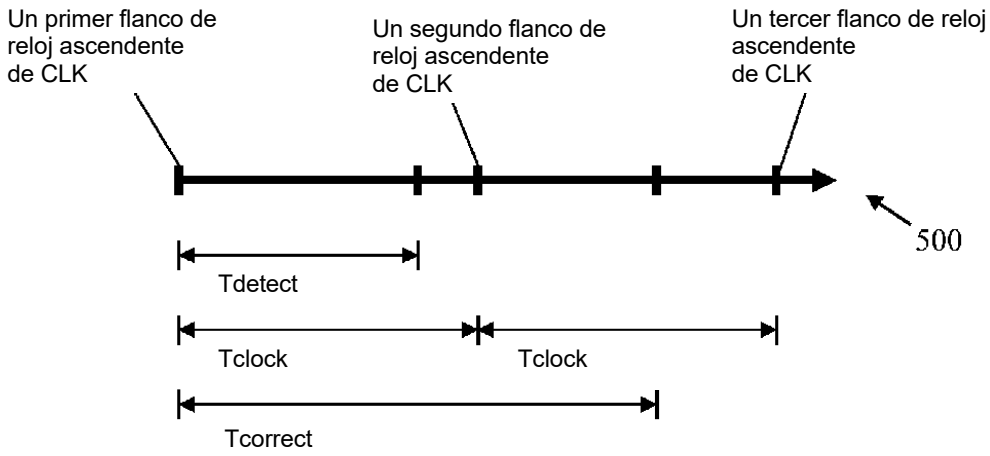


Figura 5

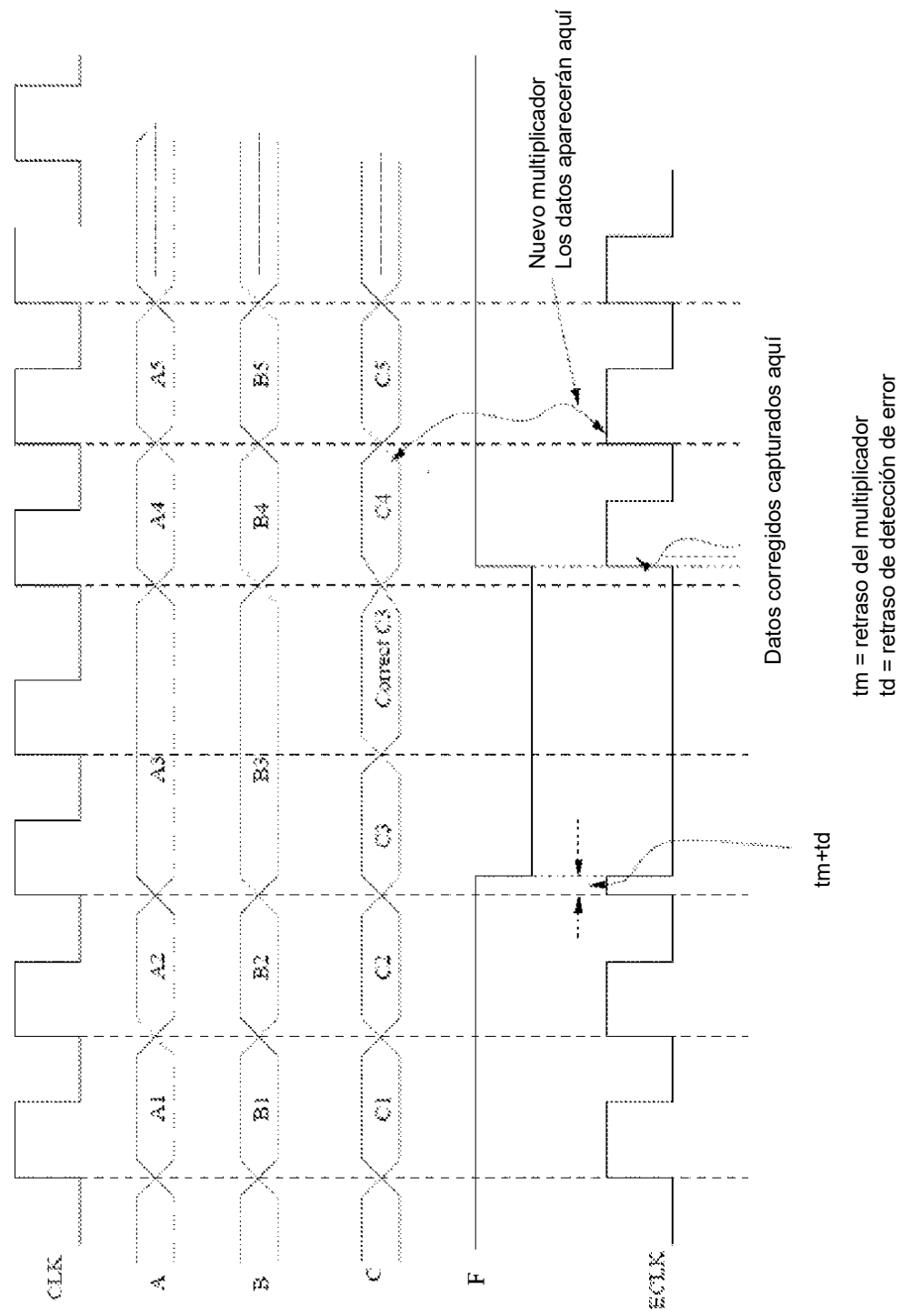

CLK
A
B
C
F
ECLK
A1
A2
A3
A4
A5
B1
B2
B3
B4
B5
C1
C2
C3
C4
C5
Correct C3
Nuevo multiplicador
Los datos aparecerán aquí
Datos corregidos capturados aquí
tm+td
tm = retraso del multiplicador
td = retraso de detección de error

Figura 6

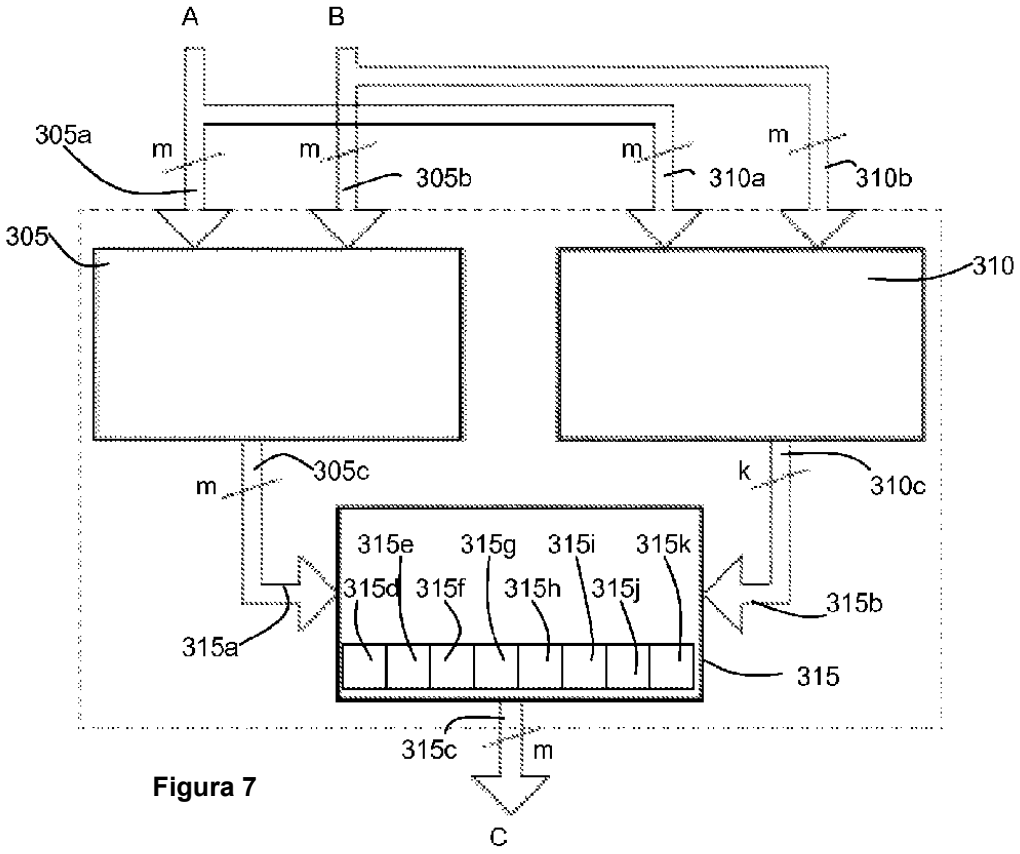


Figura 7

Figura 8

C0	C1	C2	C3	C4	Ham/BCH Paridad-1
C5	C6	C7	C8	C9	Ham/BCH Paridad-2
C10	C11	C12	C13	C14	Ham/BCH Paridad-3
C15	C16	C17	C18	C19	Ham/BCH Paridad-4
CP0	CP2	CP4	CP6	CP8	
CP1	CP3	CP5	CP7	CP9	

Figura 9a

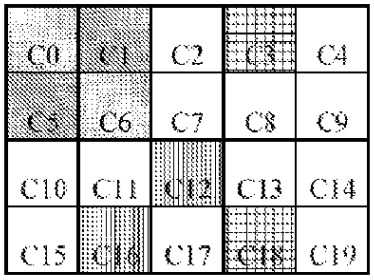
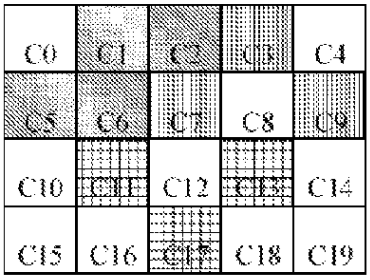


Figura 9b



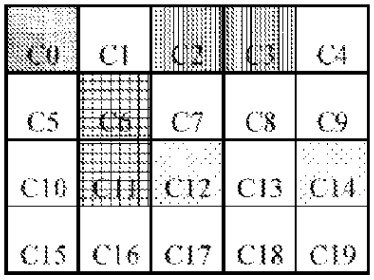


Figura 9c

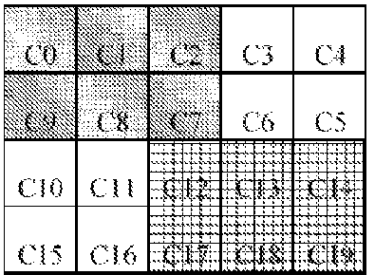


Figura 9d

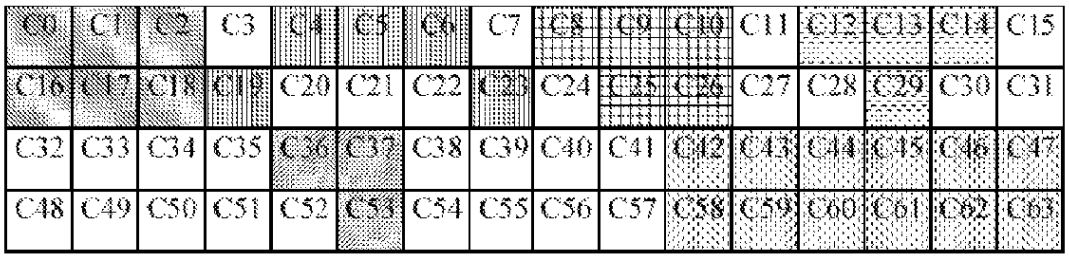
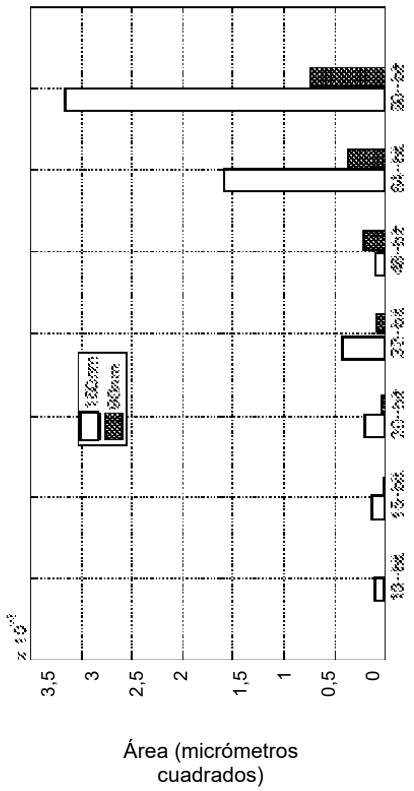


Figura 10

Figura 12



Tamaño del multiplicador

Figura 11

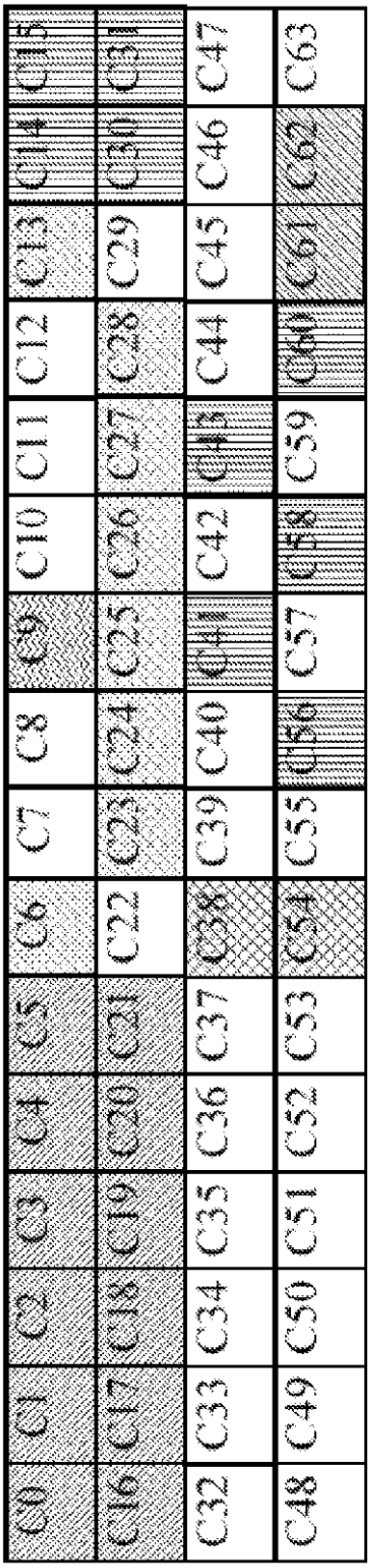


Figura 13

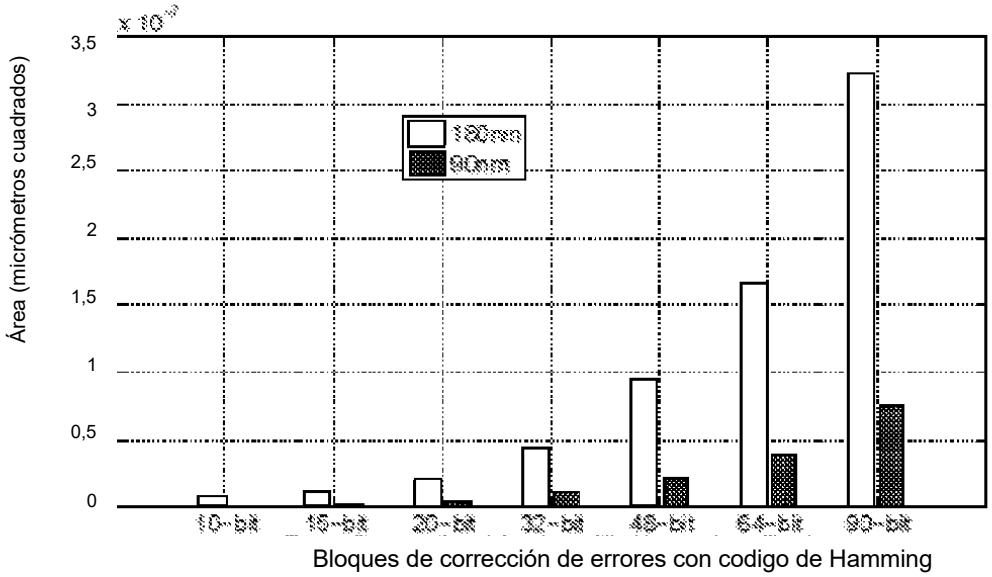


Figura 14

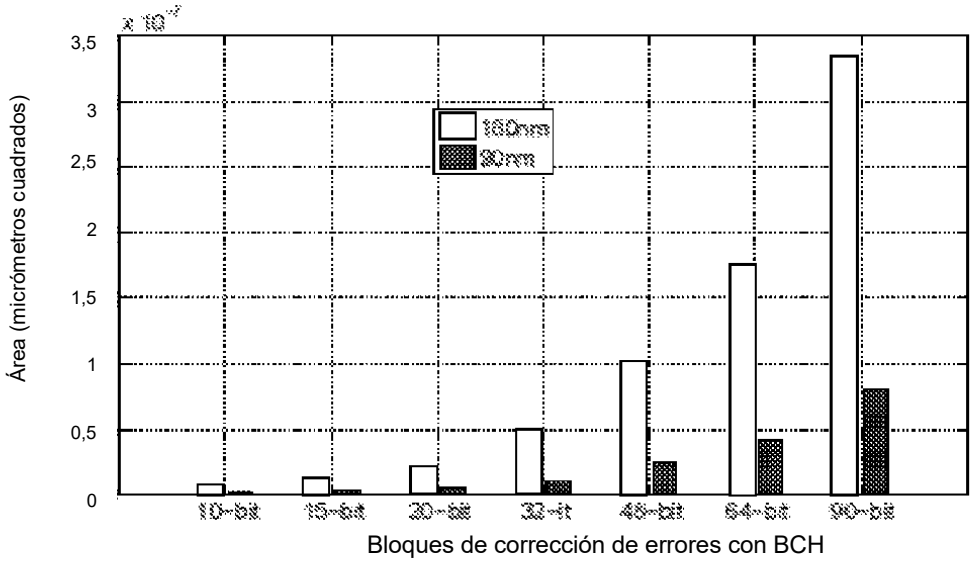
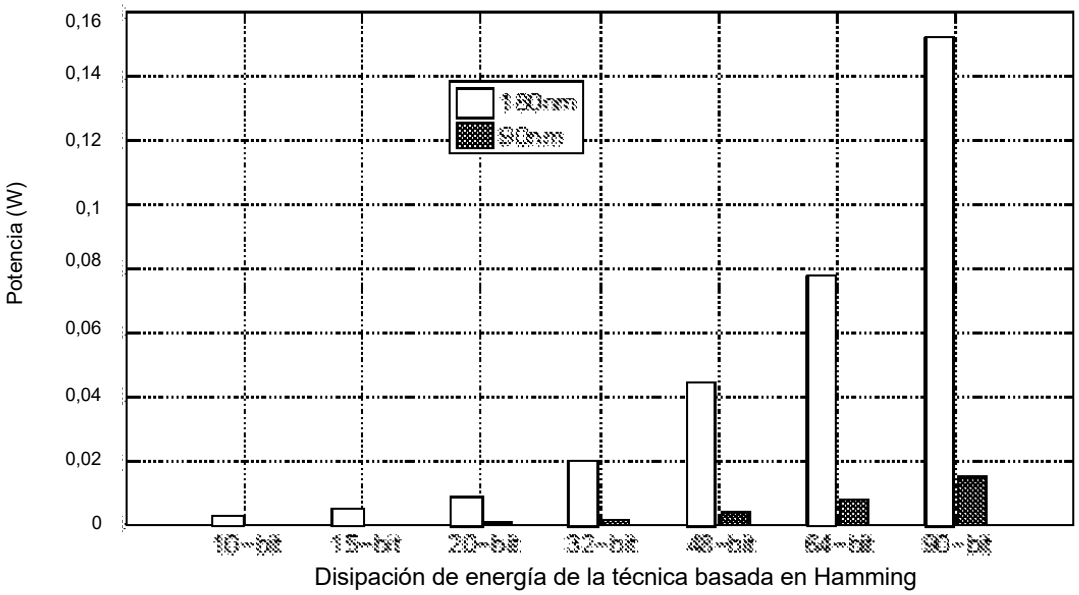
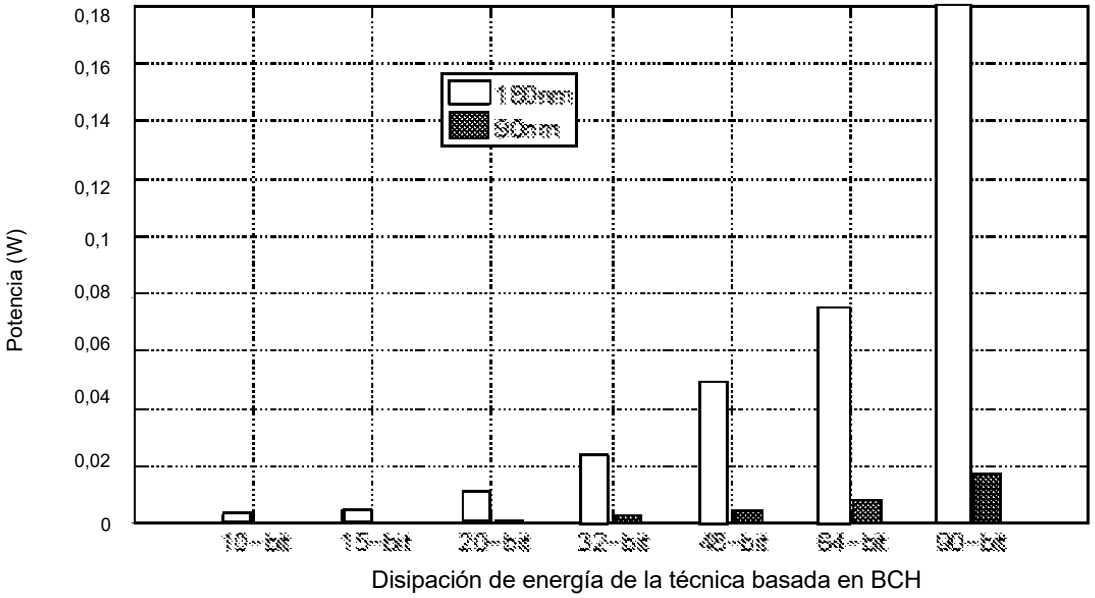


Figura 15



Disipación de energía de la técnica basada en Hamming

Figura 16



Disipación de energía de la técnica basada en BCH