(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 486 940**

(51) Int. CI.:

H04L 1/00 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **14.08.2008 E 08793258 (8)**

(97) Fecha y número de publicación de la concesión europea: **07.05.2014 EP 2183867**

(54) Título: **Método de transmisión de datos**

(30) Prioridad:

14.08.2007 KR 20070082032
16.08.2007 KR 20070082236 21.08.2007 US 957063 P 22.08.2007 US 957369 P 22.08.2007 US 957334 P 23.08.2007 US 957454 P
23.11.2007 KR 20070120389
23.11.2007 KR 20070120390
23.11.2007 KR 20070120391 13.02.2008 US 28478

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
19.08.2014

(73) Titular/es:

LG ELECTRONICS INC. (100.0%)
20, YEOUIDO-DONG YEONGDEUNGPO-GU
SEOUL 150-721, KR

(72) Inventor/es:

KIM, KI HWAN;
LEE, YOUNG SEOB;
KANG, SEUNG HYUN y
CHUNG, JAE HOON

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCIÓN

Método de transmisión de datos

Campo técnico

La presente invención se refiere a un método de transmisión de datos en un sistema de acceso inalámbrico, y más concretamente, a diversos métodos de división de datos de entrada en bloques de código en consideración del tamaño de un código de detección de errores.

Antecedentes de la técnica

En transmisión de datos, son importantes la eficiencia de transmisión de datos y la transmisión de datos fiable. Para aumentar la eficiencia de transmisión de datos, se usan generalmente métodos de división de datos antes de la transmisión y métodos que usan un código de detección de errores para comprobar si los datos contienen un error.

El control de errores se refiere a un mecanismo para detectar y corregir errores generados durante la transmisión de datos. Los esquemas de control de errores incluyen un esquema de petición de repetición automática (ARQ), un esquema de corrección de errores sin canal de retorno (FEC), y un esquema de corrección de errores hacia atrás (BEC).

El esquema de ARQ permite, para transmisión de datos fiable en una línea de comunicación, a un lado de recepción comprobar si ocurre un error por medio de una señal de reconocimiento (ACK) y tiempo de espera y permite a un lado de transmisión retransmitir una trama en la que ha ocurrido un error. El esquema de ARQ, que se llama un esquema de petición de retransmisión automático, permite a un lado de recepción detectar un error y solicitar retransmisión de datos. En el esquema de FEC, un lado de transmisión añade redundancia a los caracteres o tramas antes de la transmisión y un lado de recepción detecta y corrige los errores usando la redundancia. El esquema de BEC añade redundancia para detectar errores y transmite una señal de ARQ para retransmisión de datos a un lado de transmisión.

La detección de errores se refiere a una técnica para permitir a un lado de recepción reconocer si ha ocurrido un error durante la transmisión. Un código de detección de errores se refiere a un código que soporta la técnica de detección de errores. Las técnicas de detección de errores incluyen comprobación de paridad, suma de comprobación, comprobación de redundancia cíclica (CRC), y técnicas de código ponderado.

La corrección de errores se refiere a una técnica de codificación que incluye suficiente redundancia en un bloque de datos transmitido de manera que un lado de recepción puede deducir los caracteres de transmisión a partir del bloque de datos recibido. En términos de un modelo de capas de interconexión de sistemas abiertos (OSI), una corrección de errores se implementa principalmente en una capa de enlace de datos. Mientras tanto, una detección de errores se refiere a una técnica de codificación en la que se añade redundancia de manera que el lado de recepción pueda detectar la aparición de un error y hacer una petición de retransmisión.

La corrección de errores incluye un esquema de código de bloque en el que se añade una longitud prescrita de redundancia a una longitud predeterminada de información (señal) de manera que un lado de recepción pueda corregir errores y un esquema de código de convolución en el que un codificador tiene una memoria para usar, durante la codificación, una parte de las señales de entrada previamente además de las señales de entrada actualmente.

Un código de bloque incluye códigos de Hamming, códigos Reed-Solomon como códigos cíclicos, códigos Bose-Chaudhuri-Hocquenghem (BCH), y códigos de comprobación de redundancia cíclica (CRC). Un código de convolución incluye códigos Viterbi y turbo códigos.

La comprobación de paridad se usa de manera más general cuando el número de bits de información es pequeño y la probabilidad de generación de errores es baja. Aunque la comprobación de paridad se usa ampliamente en comunicación asíncrona debido a la simplicidad de la misma, es difícil detectar errores cuando el número de errores es un número par. La comprobación de paridad incluye una comprobación de paridad impar en la que el número de unos en caracteres que codifican los bits de paridad se fija a un número impar y una comprobación de paridad par en la que el número de unos en los caracteres que codifican los bits de paridad se fija a un número par.

La CRC, que es uno de los métodos de detección de errores, se refiere a una técnica en la que un lado de transmisión añade un resultado extraído de un polinomio a partir de los datos transmitidos a una secuencia de comprobación de trama (FCS) y transmite el campo adjunto y una comprobación del lado de recepción para errores confirmando si el resultado extraído es idéntico a un resultado extraído realizado por el mismo método en el lado de recepción. La CRC es potente y una configuración hardware de la misma es simple. Un resto obtenido dividiendo una trama de datos original que va a ser transmitida por el lado de transmisión mediante un polinomio generador de CRC es la FCS. El polinomio generador de la CRC, que es un divisor para la división, es necesario para generar la FCS. La FCS está unida a un extremo de la trama de datos original de manera que una trama resultado (añadiendo la FCS a los datos originales) se puede dividir de manera precisa por un polinomio predefinido. Es decir, la FCS

calculada para la trama de datos original está unida a un extremo de la trama. Aquí, el polinomio predefinido se conoce como el divisor o polinomio de CRC.

Un lado de recepción realiza la CRC después de recibir la trama resultado. El lado de recepción comprueba un resto dividiendo una trama de datos recibida por el mismo polinomio de CRC usado durante la transmisión. El lado de recepción detecta errores comprobando si es 0 un resto obtenido dividiendo los datos transmitidos junto con la redundancia por los datos originales. Si el resto no es 0, se determina que ha ocurrido un error durante la transmisión.

El documento “Correction of reference measurement channel for 2.048 kbps”, ERICSSON, BORRADOR DEL 3GPP, RA-020376 describe un canal de medición de referencia de UL para 2.048 kbps.

El documento TS 36.212 del 3GPP; v1.3.0 describe multiplexación y codificación de canal, especialmente que concierne a segmentación de bloque de código.

El documento “Analysis of per code block CRC y per transport block CRC”, SAMSUNG, BORRADOR DEL 3GPP, R1-073108 describe una conexión de CRC para bloques de código.

El documento “on the consideration of CRC attachment”, LG, BORRADOR DEL 3GPP, R1-073504 describe información acerca de conexión de CRC.

El documento “Text change to code block segmentation”, MOTOROLA, BORRADOR DEL 3GPP, R1-073863 describe métodos de segmentación de bloque de código.

El documento “Code Block Segmentation for LTE Channel coding”, MOTOROLA, BORRADOR DEL 3GPP, R1-071059_CODE_BLOCK_SEGMENT describe métodos de segmentación de bloque de código para codificación de Canal LTE.

Descripción

Problema técnico

En un método de conexión de CRC usado de manera general y método de segmentación de bloque de datos, un código de CRC está unido a un bloque de datos y entonces se segmenta el bloque de datos, transmitiendo por ello el bloque de datos en forma de bloques de códigos. En este caso, un lado de recepción recibe todos los bloques segmentados y combina secuencialmente los bloques de código. El lado de recepción puede determinar si el bloque de datos restaurado contiene errores a través de CRC. Por lo tanto, dado que se hace una determinación en cuanto a si el bloque de datos restaurado contiene errores después de que todos los bloques de código se restauren secuencialmente, los errores no se pueden detectar rápidamente.

Además, si un error está presente en cualquier ubicación del bloque de datos recibido, llega a ser complicado un proceso de restauración de errores realizado por el lado de recepción ya que se aumenta el número de bloques de código.

Adicionalmente, se debería considerar un código de CRC unido a cada bloque de código para calcular el número y tamaño de los bloques de código. No obstante, un método de cálculo del número y tamaño de bloques de código por una unidad de segmentación de bloques de datos usada de manera general no considera que el código de CRC esté unido a cada bloque de código.

La presente invención que planea resolver estos problemas proporciona un método de transmisión de datos eficiente.

Un objeto de la presente invención es proporcionar diversos métodos de segmentación de un bloque de datos considerando el tamaño de un código de detección de errores unido a cada bloque de código.

Otro objeto de la presente invención es proporcionar diversos métodos de cálculo del número de bloques de código.

Aún otro objeto de la presente invención es proporcionar diversos métodos de asignación de datos en consideración al número y tamaño de los bloques de código, y/o el tamaño de los códigos de detección de errores.

Un objeto adicional de la presente invención es proporcionar un método de transmisión de datos eficiente en base a los objetos anteriores.

Solución técnica

La presente invención describe realizaciones ejemplares para segmentar un bloque de datos en un sistema de acceso inalámbrico. La presente invención también describe un método de segmentación de un bloque de datos en consideración al tamaño de un código de detección de errores y un método de conexión del código de detección de errores a los bloques de código.

Según un aspecto de la presente invención, un método de transmisión de datos en un sistema de acceso inalámbrico incluye: generar una secuencia de bits de entrada uniendo un primer código de detección de errores a los datos; si el tamaño B de la secuencia de bits de entrada es mayor que el tamaño máximo Z de los bloques de código, calcular el número C de los bloques de código, usar el tamaño B de la secuencia de bits de entrada, el tamaño máximo Z de los bloques de código, y el tamaño L de un segundo código de detección de errores que va a ser unido a cada uno de los bloques de código; calcular un tamaño B' de una secuencia de bits de entrada modificada, usar el número C de los bloques de código, el tamaño L del segundo código de detección de errores, y el tamaño B de la secuencia de bits de entrada; obtener el tamaño K de los bloques de código a partir de valores prefijados, en base a un valor obtenido dividiendo el tamaño B' de una secuencia de bits de entrada modificada por el número C de los bloques de código; segmentar la secuencia de bits de entrada para tener el número C de los bloques de código y el tamaño K de los bloques de código; generar los bloques de código uniendo el segundo código de detección de errores a cada una de las secuencias de bits de entrada segmentada; y codificar en canal los bloques de código.

El número C de los bloques de código se puede fijar a un entero redondeando por exceso un valor obtenido dividiendo el tamaño B de la secuencia de bits de entrada por el resultado de restar el tamaño L del segundo código de detección de errores a ser unido a cada uno de los bloques de código del tamaño máximo Z de los bloques de código.

El primer código de detección de errores y el segundo código de detección de errores se pueden generar por diferentes polinomios.

El tamaño B de la secuencia de bits de entrada se puede fijar a un valor obtenido añadiendo el tamaño A de los datos al tamaño del primer código de detección de errores, y el segundo código de errores a ser unido a cada uno de los bloques de código se puede unir adicionalmente para detectar si cada uno de los bloques de código contiene errores.

Si el tamaño B de la secuencia de bits de entrada es menor que el tamaño máximo Z de los bloques de código, el número C de los bloques de código se puede fijar a 1.

El tamaño B' de la secuencia de bits de entrada modificada se puede fijar a un valor obtenido añadiendo el tamaño B de la secuencia de bits de entrada al resultado de multiplicar el número C de los bloques de código por el tamaño L del segundo código de detección de errores sea incluido en cada uno de los bloques de código. El tamaño K de los bloques de código puede satisfacer la condición de que un valor obtenido multiplicando el número C de los bloques de código por el tamaño K de los bloques de código es mayor o igual que el tamaño B' de la secuencia de bits de entrada modificada.

El tamaño K_+ de los primeros bloques de código en el tamaño K de los bloques de código puede tener el tamaño más pequeño entre los valores prefijados, y el tamaño K_- de los segundos bloques de código en el tamaño K de los bloques de código pueden tener el tamaño más grande entre los valores prefijados.

El número C_- de los segundos bloques de código se puede fijar a un entero redondeando por defecto un valor obtenido dividiendo, por una diferencia entre el tamaño K_+ de los primeros bloques de código y el tamaño K_- de los segundos bloques de código, el resultado de restar el tamaño B' de la secuencia de bits de entrada modificada de un valor obtenido multiplicando el número C de los bloques de código por el tamaño K_+ de los primeros bloques de código.

El número C_+ de los primeros bloques de código se puede fijar a un valor obtenido restando el número C_- de los segundos bloques de código del número C de los bloques de código.

El método de transmisión de datos puede incluir además: calcular la longitud F de bits de relleno restando el tamaño de secuencia de bits de entrada modificado, del resultado de añadir un valor obtenido multiplicando el número C_+ de los primeros bloques de código por el tamaño K_+ de los primeros bloques de código a un valor obtenido multiplicando el número C_- de los segundos bloques de código por el tamaño K_- de los segundos bloques de código; y asignar los bits de relleno a un primer bloque de código entre los bloques de código.

El método de transmisión de datos puede incluir además: asignar datos a áreas excepto para el tamaño de los bits de relleno y el tamaño del segundo código de detección de errores en un primer bloque de código de los bloques de código; y asignar los datos a áreas excepto para el tamaño del segundo código de detección de errores en bloques de código que comienzan con un segundo bloque de código.

La asignación de datos al primer bloque de código además puede incluir unir el segundo código de detección de errores al primer bloque de código, y la asignación de datos a los bloques de código que comienzan con el segundo bloque de código además puede incluir unir el segundo código de detección de errores a los bloques de código que comienzan con el segundo bloque de código.

El tamaño máximo Z de los bloques de código es 6.144 bits.

Según otro aspecto de la presente invención, un método de transmisión de datos en un sistema de acceso inalámbrico incluye: generar una segunda secuencia de bits de entrada uniendo un primer código de detección de errores a una primera secuencia de bits de entrada; calcular el tamaño B de la segunda secuencia de bits de entrada usando el tamaño A de la primera secuencia de bits de entrada y el tamaño L del primer código de detección de errores; si el tamaño B de la segunda secuencia de bits de entrada es mayor que el tamaño máximo Z de los bloques de código, calcular el número C de los bloques de código, usando el tamaño B de la segunda secuencia de bits de entrada, el tamaño máximo Z de los bloques de código, y el tamaño L de un segundo código de detección de errores que va a ser unido a cada uno de los bloques de código; calcular un segundo tamaño de bits de entrada modificado B', usando el número C de los bloques de código, el tamaño L del segundo código de detección de errores, y el tamaño B de la segunda secuencia de bits de entrada; obtener los tamaños K, K_+, y K_- de los bloques de código a partir de valores prefijados, en base a en un valor obtenido dividiendo el segundo tamaño de bits de entrada modificado B' por el número C de los bloques de código; segmentar la segunda secuencia de bits de entrada para tener el número C de los bloques de código y los tamaños obtenidos K, K_+, y K_-, de los bloques de código; generar los bloques de código uniendo el segundo código de detección de errores a cada una de las segundas secuencias de bits de entrada segmentadas; y codificar en canal los bloques de código.

Según aún otro aspecto de la presente invención, un método de transmisión de datos en un sistema de acceso inalámbrico, que comprende: generar una segunda secuencia de bits de entrada uniendo un primer código de detección de errores a una primera secuencia de bits de entrada; calcular el tamaño B de la segunda secuencia de bits de entrada usando el tamaño A de la primera secuencia de bits de entrada y el tamaño L del primer código de detección de errores; si el tamaño B de la segunda secuencia de bits de entrada es mayor que el tamaño máximo Z de bloques de código, calcular un número C de los bloques de código, usando el tamaño B de la segunda secuencia de bits de entrada, el tamaño máximo Z de los bloques de código, y el tamaño L de un segundo código de detección de errores que va a ser unido a cada uno de los bloques de código; calcular el tamaño B' de una segunda secuencia de bits de entrada modificada, usando el número C de los bloques de código, el tamaño L del segundo código de detección de errores, y el tamaño B de la segunda secuencia de bits de entrada; obtener el tamaño Kr de los bloques de código dividiendo el tamaño B' de la segunda secuencia de bits de entrada modificada por el número C de los bloques de código; segmentar la segunda secuencia de bits de entrada para tener el número C de los bloques de código y el tamaño obtenido Kr de los bloques de código; generar los bloques de código uniendo el segundo código de detección de errores a cada una de las segundas secuencias de bits de entrada segmentadas; y codificar en canal los bloques de código.

El primer y segundo códigos de detección de errores se generan mediante polinomios diferentes.

Según aún otro aspecto de la presente invención, un método de transmisión de datos en un sistema de acceso inalámbrico, que comprende: si el tamaño B de una secuencia de bits de entrada es mayor que el tamaño máximo Z de los bloques de código, calcular un número C de los bloques de código, usando el tamaño B de la secuencia de bits de entrada, el tamaño máximo Z de los bloques de código, y el tamaño L de un código de detección de errores que va a ser unido a cada uno de los bloques de código; calcular el tamaño B' de una secuencia de bits de entrada modificada, usando el número C de los bloques de código, el tamaño L del código de detección de errores, y el tamaño B de la secuencia de bits de entrada; obtener el tamaño K de los bloques de código a partir de valores predeterminados, en base a un valor obtenido dividiendo el tamaño B' de la secuencia de bits de entrada modificada por el número C de los bloques de código; y segmentar la secuencia de bits de entrada para tener el número C de los bloques de código y el tamaño K de los bloques de código.

Según aún otro aspecto de la presente invención, un método de transmisión de datos en un sistema de acceso inalámbrico, que comprende: generar una secuencia de bits de entrada uniendo un primer código de detección de errores a un bloque de transporte; si el tamaño B de la secuencia de bits de entrada es mayor que el tamaño máximo Z de bloques de código, calcular un número C de los bloques de código, usando el tamaño B de la secuencia de bits de entrada, el tamaño máximo Z de los bloques de código, y el tamaño L de un código de detección de errores que va a ser unido a cada uno de los bloques de código; calcular el tamaño B' de una secuencia de bits de entrada modificada, usando el número C de los bloques de código, el tamaño L del código de detección de errores, y el tamaño B de la secuencia de bits de entrada; obtener el tamaño Kr de los bloques de código dividiendo el tamaño B' de la secuencia de bits de entrada modificada por el número C de los bloques de código; y segmentar la secuencia de bits de entrada para tener el número C de los bloques de código y el tamaño Kr de los bloques de código.

Efectos ventajosos

La presente invención tiene las siguientes ventajas.

En primer lugar, dado que un lado de recepción puede determinar si existen errores siempre que se reciben bloques de código, se puede realizar un proceso de restauración de errores eficiente.

En segundo lugar, se puede realizar de manera precisa una segmentación de un bloque de datos y se pueden unir de una manera precisa códigos de CRC calculando el número de bloques de código y segmentando el bloque de datos en consideración a la longitud de los códigos de CRC.

En tercer lugar, cuando se segmenta un bloque de datos en bloques de código, el número de bloques de código se calcula en consideración a la longitud de los códigos de CRC y los datos se asignan entonces a los bloques de código, aumentando por ello la eficiencia de procesamiento de datos.

En cuarto lugar, si el tamaño de una entrada de secuencia de bits a un sistema en menor que un tamaño máximo que se puede dividir en el sistema, no se realiza innecesariamente una segmentación de la secuencia de bits de entrada. En su lugar, se usan los bits de entrada correspondientes para un bloque de código. Es innecesario volver a unir un código de detección de errores debido a que se puede usar el código de detección de errores para detectar un error de los bits de entrada. Por lo tanto, es posible procesar rápidamente el bloque de código.

En quinto lugar, los datos se pueden transmitir de manera eficiente a través de diversas realizaciones de la presente invención.

Descripción de los dibujos

Los dibujos anexos, que se incluyen para proporcionar una comprensión adicional de la invención, ilustran realizaciones de la invención y junto con la descripción sirven para explicar el principio de la invención.

En los dibujos:

La FIG. 1 es un diagrama que ilustra un proceso realizado en cada una de una unidad de conexión de CRC y una unidad de segmentación de bloque de datos;

La FIG. 2 es un diagrama que ilustra un proceso de conversión de un bloque de datos en bloques de código;

La FIG. 3 es un diagrama que ilustra un proceso de segmentación de un bloque de datos en consideración a un tamaño de CRC según una realización ejemplar de la presente invención;

La FIG. 4 es un diagrama que ilustra un ejemplo de un proceso de conexión de códigos de CRC a bloques de código según una realización ejemplar de la presente invención;

La FIG. 5 es un diagrama que ilustra otro ejemplo de un proceso de conexión de códigos de CRC a bloques de código según una realización ejemplar de la presente invención;

La FIG. 6 es un diagrama que ilustra un proceso de segmentación de un bloque de datos y conexión de códigos de CRC en consideración a un tamaño de CRC según una realización ejemplar de la presente invención;

La FIG. 7 es un diagrama que ilustra un proceso de conversión de un bloque de datos en bloques de código en consideración a un tamaño de CRC según una realización ejemplar de la presente invención;

La FIG. 8 es un diagrama de flujo que ilustra un proceso de segmentación de un bloque de datos en consideración a un tamaño de CRC según una realización ejemplar de la presente invención;

La FIG. 9 es un diagrama de flujo que ilustra un proceso de cálculo del número de bloques de código según una realización ejemplar de la presente invención;

La FIG. 10 es un diagrama de flujo que ilustra un proceso de segmentación de un bloque de datos usando un tamaño de secuencia de bits de entrada modificado según una realización ejemplar de la presente invención; y

La FIG. 11 es un diagrama de flujo que ilustra un proceso de conversión de un bloque de datos en bloques de código cuando el número de bloques de código es 1 según una realización ejemplar de la presente invención.

Modo de la invención

La presente invención proporciona un método de transmisión de datos y segmentación de bloque de código en un sistema de acceso inalámbrico, especialmente un método de segmentación de un bloque de datos en consideración al tamaño de códigos de detección de errores y un método de obtención del número de bloques de código.

Las realizaciones ejemplares descritas en lo sucesivo son combinaciones de elementos y rasgos de la presente invención. Los elementos o rasgos se pueden considerar como selectivos a menos que se mencione de otro modo. Cada elemento o rasgo se puede poner en práctica sin que se combine con otros elementos o rasgos. Además, una realización de la presente invención se puede construir combinando partes de los elementos y/o rasgos. Los órdenes de operación descritos en las realizaciones de la presente invención se pueden reorganizar. Algunas construcciones de cualquier realización se pueden incluir en otra realización y se pueden sustituir con construcciones correspondientes de otra realización.

En la siguiente descripción de la presente invención, se omitirá una descripción detallada de procedimientos o pasos conocidos cuando puedan oscurecer la materia objeto de la presente invención.

En realizaciones ejemplares de la presente invención, se hace una descripción de una relación de transmisión y recepción de datos entre una estación base y una estación móvil. Aquí, el término 'estación base' se refiere a un nodo terminal de una red que comunica directamente con la estación móvil. En algunos casos, una operación específica descrita como realizada por la estación base se puede realizar por un nodo superior de la estación base.

Esto es, es evidente que, en una red compuesta de una pluralidad de nodos de red que incluyen una estación base, se pueden realizar diversas operaciones realizadas para comunicación con una estación móvil por la estación base u otros nodos de red excepto para la estación base. El término 'estación base' se puede sustituir con el término 'estación fija', 'Nodo B', 'eNodo B' (eNB), 'punto de acceso', etc. El término 'estación móvil' (MS) se puede sustituir con el término 'terminal', 'equipo de usuario' (UE), 'estación móvil de abonado' (MSS), etc.

Adicionalmente, el término 'lado de transmisión' significa un nodo que transmite un servicio de voz o datos, y el término 'lado de recepción' significa un nodo que recibe el servicio de voz o datos. Por lo tanto, en el enlace ascendente, la estación móvil puede ser el lado de transmisión y la estación base puede ser el lado de recepción. De manera similar, en el enlace descendente, la estación móvil puede ser el lado de recepción y la estación base puede ser el lado de transmisión.

Mientras tanto, la estación móvil puede incluir un asistente digital personal (PDA), un teléfono celular, un teléfono del servicio de comunicación personal (PCS), un teléfono del sistema global para móviles (GSM), un teléfono de acceso múltiple por división de código de banda ancha (WCDMA), un teléfono del sistema de banda ancha móvil (MBS), etc.

Las realizaciones de la presente invención se pueden lograr por diversos medios, por ejemplo, hardware, microprogramas, software, o una combinación de los mismos.

En una configuración hardware, se puede lograr un método según realizaciones ejemplares de la presente invención mediante uno o más circuitos integrados de aplicaciones específicas (ASIC), procesadores de señal digital (DSP), dispositivos de procesamiento de señal digital (DSPD), dispositivos de lógica programable (PLD), disposiciones de puertas programables en campo (FPGA), procesadores, controladores, micro controladores, microprocesadores, etc.

En una configuración de microprogramas o software, se puede lograr un método según realizaciones ejemplares de la presente invención mediante un módulo, un procedimiento, una función, etc. que realizan las funciones u operaciones descritas anteriormente. Un código software se puede almacenar en una unidad de memoria y accionar por un procesador. La unidad de memoria se sitúa en el interior o exterior del procesador y puede transmitir y recibir datos con el procesador a través de diversos medios conocidos.

Las realizaciones de la presente invención se pueden soportar por documentos descritos en al menos uno de los sistemas de acceso inalámbricos (por ejemplo, el sistema IEEE 802, el sistema 3GPP, el sistema LTE del 3GPP, y el sistema del 3GPP2). Especialmente, los documentos descritos en la TS 36.212 V8.0.0 (09-2007) del 3GPP a la TS 36.212 V8.3.0 (05-2008) del 3GPP pueden soportar realizaciones de la presente invención.

La siguiente descripción detallada incluye términos específicos a fin de proporcionar una comprensión minuciosa de la presente invención. No obstante, esos términos específicos se pueden modificar sin apartarse del alcance de la presente invención.

La FIG. 1 es un diagrama que ilustra un proceso realizado en cada una de una unidad de conexión de CRC y una unidad de segmentación de bloque de datos.

Una unidad de unidad de conexión de CRC y una unidad de segmentación de bloque de datos usadas de manera general se describirán ahora con referencia a la FIG. 1. Un bloque de datos 100 se puede segmentar en múltiples bloques de código cuando sea necesario. Un bloque de código 160 se genera mediante segmentación del bloque de datos 100.

Si un usuario introduce el bloque de datos 100, una unidad de conexión de CRC 120 une un código de CRC al bloque de datos 100. El bloque de datos que incluye el código de CRC se divide en una longitud necesaria de bloques de datos mediante una unidad de segmentación de bloque de datos 140 y constituye el bloque de código 160 que tiene una o más longitudes. Este proceso se muestra secuencialmente en el lado derecho de la FIG. 1.

La unidad de conexión de CRC 120 une un código de CRC al bloque de datos que tiene una longitud prescrita de manera que un lado de recepción puede usar el bloque de datos 100 para detectar un error. Para este fin, la unidad de conexión de CRC 120 genera bits de paridad de CRC de una longitud prescrita usando la ecuación de generación de CRC en base al bloque de datos de entrada (paso S101). A continuación, la unidad de conexión de CRC 120 une hace delante o hacia atrás los bits de paridad de CRC a un bloque de datos hacia delante para generar una forma conectada en serie del bloque de datos (paso S102).

El bloque de datos unido con CRC se segmenta en uno o más bloques de código múltiples mediante la unidad de segmentación de bloque de datos 140. La unidad de segmentación de bloque de datos 140 segmenta el bloque de datos de entrada en uno o más bloques de código a través de los cuatro procesos siguientes.

La unidad de segmentación de bloque de datos 140 fija el número de bloques de código a un entero redondeando por exceso el resultado de dividir el tamaño del bloque de datos unidos con CRC por un tamaño máximo permitido del bloque de código (paso S103).

La unidad de segmentación de bloque de datos 140 calcula el tamaño de cada bloque de código según el número de bloques de código determinados en el paso S103 (paso S104).

En el paso S104, la suma de añadir el tamaño de cada uno de los bloques de código puede ser mayor que el tamaño del bloque de datos unido con CRC. En este caso, un valor obtenido restando el tamaño del bloque de datos unido con CRC del tamaño de todos los bloques de código se fija a la longitud de los bits de relleno (paso S105).

Si se determinan el número y tamaño de los bloques de código y la longitud de los bits de relleno, la unidad de segmentación de bloque de datos 140 segmenta el bloque de datos unido con CRC y asigna datos a cada bloque de código (paso S106). En el paso S106, los bits de relleno y datos se asignan secuencialmente al principio del primer bloque entre los bloques de código y los siguientes datos se asignan secuencialmente comenzando con el segundo bloque.

La FIG. 2 es un diagrama que ilustra un proceso de conversión de un bloque de datos en bloques de código.

En la FIG. 2, se pueden emplear las unidades y el método usado en la FIG. 1.

Con referencia a la FIG. 2, se introduce un bloque de datos 200 a una unidad de conexión de CRC 220. Los bits de CRC se unen al bloque de datos 200 mediante la unidad de conexión de CRC 220 para generar un bloque de datos unido con CRC 230. El bloque de datos unido con CRC se introduce a una unidad de segmentación de bloque de datos 240 y luego se segmenta en bloques de código. Los bits de relleno se unen al principio del primer bloque de un bloque de código 260 y los datos se asignan a la otra parte del mismo. Los datos se asignan secuencialmente comenzando con un segundo bloque de código.

En realizaciones ejemplares de la presente invención, se asume que los códigos de CRC se usan como un tipo de códigos de detección de errores deseables que se pueden unir a los bloques de código mediante la unidad de segmentación de bloque de datos. Además, el término ‘bloque de datos’ es los bits de entrada que se introducen a la unidad de segmentación de bloque de datos y se puede conocer como un primer bloque de datos. Si se realiza segmentación de los bits de entrada, se genera un bloque de código y un segundo bloque de datos.

<Primera Realización>

La FIG. 3 es un diagrama que ilustra un proceso de segmentación de un bloque de datos en consideración a un tamaño de CRC según una realización ejemplar de la presente invención.

Con referencia a la FIG. 3, se introduce un bloque de datos (por ejemplo, un primer bloque de datos 300) a una unidad de segmentación de bloque de datos 320 y se segmenta en uno o más bloques de código (por ejemplo, un segundo bloque de datos). Los datos se asignan secuencialmente a los bloques de código. Los bloques de código se introducen a una unidad de conexión de CRC 340. En este caso, el bloque de datos puede incluir un código de detección de errores dentro del mismo y el tamaño del código de detección de errores es deseablemente de 24 bits.

La unidad de conexión de CRC 340 genera códigos de CRC y une los códigos de CRC a los bloque de código, excepto cuando el bloque de datos se compone de un bloque de código que incluye un código de detección de errores (por ejemplo, excepto cuando un tamaño de bloque de datos B es menor o igual que un tamaño de bloque de código máximo Z). De esta manera el bloque de datos 300 se segmenta en el bloque de código 360 a través de la unidad de segmentación de bloque de datos 320 y la unidad de conexión de CRC 340. En la FIG. 3, el bloque de código 360 significa uno o más bloques de datos segmentados.

Con referencia a la FIG. 3, si el bloque de datos 300 se introduce a la unidad de segmentación de bloque de datos 320, la unidad de segmentación de bloque de datos 320 calcula el número C de bloques de código con respecto a los datos de entrada. En este caso, la unidad de segmentación de bloque de datos 320 puede calcular el número C de bloques de código en consideración al tamaño L de un código de CRC que va a ser unido a cada uno de los bloques de código finales (paso S301).

En lo sucesivo, se describirán diversos métodos de cálculo del número C de bloques de código usados en el paso S301.

Se describirá ahora un primer método de cálculo del número de bloques de código en el paso S301 según una realización ejemplar de la presente invención.

La Ecuación 1 siguiente ilustra un ejemplo de cálculo del número C de bloques de código.

[Ecuación 1]

C' = $\lceil B/Z \rceil$; $\lceil x \rceil$ es un entero redondeado por exceso x

si C' * L + B > C' * Z

C = C' + 1

además

C = C'

fin

La unidad de segmentación de bloque de datos 320 fija, a un valor temporal C', un entero redondeando por exceso un valor obtenido dividiendo el tamaño de bloque de datos B por el tamaño de bloque de código máximo Z. Si un valor obtenido añadiendo el tamaño de bloque de datos B al resultado de multiplicar el valor temporal C' por el tamaño de CRC L es mayor que un valor obtenido multiplicando el valor temporal C' por el tamaño de bloque de código máximo Z, el número C de bloques de código se fija a un valor obtenido añadiendo 1 al valor temporal C', si no, el número C de bloques de código se fija al valor temporal C'.

Mientras tanto, la Ecuación 2 mostrada más adelante se puede usar cuando el tamaño de bloque de datos B introducido a la unidad de segmentación de bloque de datos es menor o igual que el tamaño de bloque de código máximo Z.

[Ecuación 2]

si B $\leq$ Z

C = 1

además

C' = $\lceil B/Z \rceil$

Si C' * L + B > C' * Z

C = C' + 1

además

C = C'

fin

fin

La Ecuación 1 y Ecuación 2 usan el valor temporal C' para calcular el número C de bloques de código. Es decir, el número de bloques de código se puede calcular de manera precisa adquiriendo el valor temporal obtenido redondeando por exceso el resultado de dividir el tamaño de la entrada de bloque de datos a la unidad de segmentación de bloque de datos 320 por el tamaño de bloque de código máximo.

Otras formas de la Ecuación 1 y Ecuación 2 se ilustran en la Ecuación 3 y Ecuación 4. Esto es, el número de bloques de código se calcula directamente sin usar el valor temporal C'.

[Ecuación 3]

Si $\lceil B/Z \rceil$ * L + B > $\lceil B/Z \rceil$ * Z

C = $\lceil B/Z \rceil$ + 1

además

C = $\lceil B/Z \rceil$

fin

Mientras tanto, se puede usar la Ecuación 4 siguiente cuando el tamaño B de la entrada de bloque de datos a la unidad de segmentación de bloque de datos es menor o igual que el tamaño de bloque de datos máximo Z.

[Ecuación 4]

Si B $\leq$ Z

C = 1

además

si $\lceil B/Z \rceil$ * L + B > $\lceil B/Z \rceil$ * Z

C = $\lceil B/Z \rceil$ + 1

además

C = $\lceil B/Z \rceil$

fin

fin

Se describirá ahora un segundo método de cálculo del número de bloques de código en el paso descrito anteriormente S301 según la realización ejemplar de la presente invención.

La Ecuación 5 siguiente ilustra un método de cálculo del número C de los bloques de código usando un tamaño de secuencia de bits de entrada modificado B'.

[Ecuación 5]

Si $\lceil B/Z \rceil$ * L + B > $\lceil B/Z \rceil$ * Z

B' = B + ($\lceil B/Z \rceil$ + 1) * L

además

B' = B + $\lceil B/Z \rceil$ * L

fin

C = $\lceil B'/Z \rceil$

En la Ecuación 5, el tamaño de secuencia de bits de entrada modificado B' se calcula para obtener el número de bloques de código. Se asume que un valor, obtenido redondeando por exceso el resultado de dividir el tamaño de bloque de datos B por el tamaño de bloque de código máximo Z, multiplicado por el tamaño de CRC L más el tamaño de bloque de datos B es 'M'. También se asume que un valor, obtenido redondeando por exceso el resultado de dividir el tamaño de bloque de datos B por el tamaño de bloque de código máximo Z, multiplicado por el tamaño de bloque de código máximo Z es 'N'.

Si M es mayor que N, el tamaño de secuencia de bits de entrada modificado B' para calcular el número C y tamaño K de bloques de código asume un valor obtenido redondeando por exceso el resultado de dividir el tamaño de bloque de datos B por el tamaño de bloque de código máximo Z, más 1 multiplicado por el tamaño de CRC L más el tamaño de bloque de datos B.

Si M es menor que N, el tamaño de la secuencia de bits de entrada modificado B' asume un valor que redondea por exceso el resultado de dividir el tamaño de bloque de datos B por el tamaño de bloque de código máximo Z, multiplicado por el tamaño de CRC L más el tamaño de bloque de datos B.

Por lo tanto, el número C de bloques de código se fija a un entero obtenido redondeando por exceso el resultado de dividir el tamaño de secuencia de bits de entrada modificado B' para calcular el número C de bloques de código y el tamaño de bloque de código K por el tamaño de bloque de código máximo Z.

Mientras tanto, cuando el tamaño de bloque de datos B introducido a la unidad de segmentación de bloque de datos es menor o igual que el tamaño de bloque de código máximo Z, se usa la Ecuación 6 siguiente.

[Ecuación 6]

Si B ≤ Z

B' = B

además

si $\lceil B/Z \rceil$ * L + B > $\lceil B/Z \rceil$ * Z

B' = B + ($\lceil B/Z \rceil$ + 1) * L

además

B' = B + $\lceil B/Z \rceil$ * L

fin

fin

C = $\lceil B'/Z \rceil$

La Ecuación 5 y Ecuación 6 muestran un método de cálculo del número C de bloques de código que usa el tamaño de secuencia de bits de entrada modificado B' sin usar el valor temporal C' a diferencia de la Ecuación 1 y Ecuación 2. Es decir, se puede obtener el número C de bloques de código usando el tamaño de secuencia de bits de entrada modificado B'.

Se describirá ahora un tercer método de cálculo del número de bloques de código en el paso S301 según la realización ejemplar de la presente invención.

La Ecuación 7 siguiente ilustra otro ejemplo de cálculo del número C de bloques de código.

[Ecuación 7]

$$C = \lceil B/(Z-L) \rceil$$

El número C de bloques de código se puede fijar a un entero redondeando por exceso el resultado de dividir el tamaño de bloque de datos B por un valor obtenido restando el tamaño de CRC L del tamaño de bloque de código máximo Z.

Mientras tanto, si el tamaño de bloque de datos B introducido a la unidad de segmentación de bloque de datos es menor o igual que el tamaño de bloque de código máximo Z, se puede usar la Ecuación 8 siguiente.

[Ecuación 8]

Si B ≤ Z

C = 1

además

$C = \lceil B/(Z-L) \rceil$

fin

La Ecuación 7 y Ecuación 8 ilustran un método de cálculo del número C de bloques de código en consideración al tamaño de CRC L para cada bloque de código. Es decir, el tamaño de CRC L se puede considerar cuando se segmenta el tamaño de bloque de datos B, dividiendo el tamaño de bloque de datos B por un valor obtenido restando el tamaño de CRC L del tamaño de bloque de código máximo Z. En consecuencia, la segmentación del bloque de datos se puede realizar de manera precisa según los requisitos del usuario.

Se describirá ahora un cuarto método de cálculo del número C de bloques de código en el paso S301 según la realización ejemplar de la presente invención.

La Ecuación 9 mostrada más adelante ilustra el caso en el que el tamaño de bloque de código máximo Z es variable.

[Ecuación 9]

Z' = Z – a

$$C = \lceil B / Z' \rceil$$

La unidad de segmentación de datos puede fijar el número C de bloques de código a un entero redondeando por exceso el resultado de dividir el tamaño de bloque de datos B por un valor Z' que es menor que el tamaño de bloque de código máximo Z por 'a' (donde 'a' es un número natural). Aquí, 'a' puede ser un tamaño necesario según el tamaño de CRC L o los entornos del sistema.

Mientras tanto, si el tamaño de bloque de datos (o bits de entrada) B es menor que el tamaño de bloque de código máximo Z, se puede usar la Ecuación 10 siguiente.

[Ecuación 10]

Si $B \leq Z$

$C = 1$

además

$Z' = Z - a$

$$C = \lceil B / Z' \rceil$$

fin

Se describirá ahora un quinto método de cálculo del número de bloques de código en el paso S301 según la realización ejemplar de la presente invención.

La unidad de segmentación de datos sustituye el tamaño de bloque de código máximo Z con un valor específico x y fija el número C de bloques de código a un entero redondeando por exceso el resultado de dividir el tamaño de bloque de datos B por el valor específico x.

La Ecuación 11 siguiente ilustra un ejemplo de cálculo del número C de bloques de código.

[Ecuación 11]

$Z = x$

$$C = \lceil B / Z \rceil$$

Mientras tanto, si el tamaño de bloque de datos (o bits de entrada) B introducido al bloque de datos es menor o igual que el tamaño de bloque de código máximo Z, se puede usar la Ecuación 12.

[Ecuación 12]

Si $B \leq Z$

$C = 1$

además

$Z = x$

$$C = \lceil B / Z \rceil$$

fin

La Ecuación 11 y Ecuación 12 se pueden usar cuando el tamaño de bloque de código máximo Z es variable. Es decir, dado que el tamaño de bloque de código máximo se puede cambiar según los entornos del sistema, la Ecuación 11 y Ecuación 12 se usan de manera flexible.

Con referencia de nuevo a la FIG. 3, si se determina el número C de bloques de código en el paso S301, el tamaño de cada bloque de código y el número de bloques de código que tienen una longitud específica se calculan usando al menos uno del número C de bloques de código y el tamaño de secuencia de bits de entrada modificado B' (paso S302).

En lo sucesivo, se describirán diversos métodos para calcular un tamaño de bloque de código K en el paso S302 según una realización ejemplar de la presente invención.

El tamaño de bloque de código K puede tener diversos tamaños según los requisitos de sistema. En la realización ejemplar de la presente invención, se asumen los casos en los que cada tamaño de bloque de código K es constante, o tiene tamaños K_+ y K_-. No obstante, es evidente que se pueden usar diversos tamaños de bloques de código. En realizaciones ejemplares de la presente invención, el bloque de código se segmenta del bloque de datos (o bits de entrada) y se puede conocer como un segmento.

Se describirá ahora un primer método de cálculo del tamaño de bloque de código K cuando el tamaño de cada bloque de código es el mismo en el paso S302.

La Ecuación 13 ilustra un ejemplo de cálculo del tamaño de bloque de código K cuando el tamaño de cada bloque de código es constante.

[Ecuación 13]

$$K = \lceil B'/C \rceil$$

La Ecuación 13 muestra un método de cálculo para el tamaño de bloque de código K cuando el tamaño de cada bloque de código es constante. Es decir, el tamaño de bloque de código K se fija a un entero redondeando por exceso el resultado de dividir el tamaño de bits de entrada modificado B' por el número C de bloques de código. En este caso, el tamaño de bits de entrada modificado B' es un valor temporal para obtener el número y tamaño de bloques de código.

Si el tamaño de bits de entrada modificado B' para calcular el número y tamaño de bloques de código no se calcula en el paso S301, un valor, (C x L + B), obtenido multiplicando el número C de bloques de código por el tamaño de CRC L y luego añadiendo el tamaño de bloque de datos B al resultado multiplicado se puede usar como el tamaño de bits de entrada modificado B'.

Se describirá ahora un segundo método de cálculo de un primer tamaño de bloque de código K_+, cuando el tamaño de bloque de código k tiene un tamaño específico K_+ o K_- en el paso S302.

La Ecuación 14 ilustra un ejemplo de cálculo del primer tamaño de bloque de código K_+.

[Ecuación 14]

K_+ es un valor mínimo de K,

donde K satisface $C * K \geq B + C * L$ o $C * K \geq B'$

Un bloque de código que tiene el tamaño K_+ usa un valor K en la Tabla 1 mostrada más adelante. En este caso, una condición del valor K es que el resultado de multiplicar el número C de bloques de código por K es mayor o igual que un valor obtenido añadiendo el tamaño de secuencia de entrada B al resultado de multiplicar el número C de bloques de código por el tamaño de CRC L. Otra condición del valor K es que el resultado de multiplicar el número C de bloques de código por K es mayor o igual que el tamaño de bits de entrada modificado B'. Esto es, el valor K_+ puede tener un valor mínimo entre los valores K que satisfacen cualquiera de dos condiciones de la Ecuación 14.

Si el tamaño de bloque de código se calcula usando el método que se ilustra en la Ecuación 14, dado que el tamaño de bloque de código se obtiene considerando el tamaño de CRC L, el bloque de datos de entrada (o bits de entrada) se puede dividir de manera precisa en las longitudes deseadas.

La Ecuación 15 ilustra otro ejemplo de cálculo del primer tamaño de bloque de código K_+.

[Ecuación 15]

K_+ es un valor mínimo de K,

donde K satisface $C * (K - L) \geq B'$

En la Ecuación 15, el bloque de código que tiene el tamaño K_+ usa un valor mínimo de K que satisface la siguiente condición en la Tabla 1. Es decir, el valor más pequeño se usa como K_+ entre los valores K que satisfacen una condición de que un valor obtenido multiplicando el número C de bloques de código por el resultado de restar el tamaño de CRC L de K es mayor o igual al tamaño de bits de entrada modificado B'.

La Ecuación 16 siguiente se puede usar cuando el tamaño de bits de entrada modificado B' en la Ecuación 15 se fija a un valor obtenido añadiendo el tamaño de bits de entrada B al resultado de multiplicar el número C de bloques de código por el tamaño de CRC L.

[Ecuación 16]

K_+ es un valor mínimo de K,

donde K satisface $C * (K - L) \geq B + C * L$

En la Ecuación 16, un bloque de código que tiene el tamaño K_+ puede usar el valor K mostrado en la Tabla 1.

La Tabla 1 siguiente ilustra parámetros para el valor K que se pueden usar en la Ecuación 13 a la Ecuación 16.

[Tabla 1]

i	K_i	f_1	f_2	i	K_i	f_1	f_2	i	K_i	f_1	f_2	i	K_i	f_1	f_2
1	40	3	10	48	416	25	52	95	1120	67	140	142	3200	111	240
2	48	7	12	49	424	51	106	96	1152	35	72	143	3264	443	204
3	56	19	42	50	432	47	72	97	1184	19	74	144	3328	51	104
4	64	7	16	51	440	91	110	98	1216	39	76	145	3392	51	212
5	72	7	18	52	448	29	168	99	1248	19	78	146	3456	451	192
6	80	11	20	53	456	29	114	100	1280	199	240	147	3520	257	220
7	88	5	22	54	464	247	58	101	1312	21	82	148	3584	57	336
8	96	11	24	55	472	29	118	102	1344	211	252	149	3648	313	228
9	104	7	26	56	480	89	180	103	1376	21	86	150	3712	271	232
10	112	41	84	57	488	91	122	104	1408	43	88	151	3776	179	236
11	120	103	90	58	496	157	62	105	1440	149	60	152	3840	331	120
12	128	15	32	59	504	55	84	106	1472	45	92	153	3904	363	244
13	136	9	34	60	512	31	64	107	1504	49	846	154	3968	375	248
14	144	17	108	61	528	17	66	108	1536	71	48	155	4032	127	168
15	152	9	38	62	544	35	68	109	1568	13	28	156	4096	31	64
16	160	21	120	63	560	227	420	110	1600	17	80	157	4160	33	130
17	168	101	84	64	576	65	96	111	1632	25	102	158	4224	43	264
18	176	21	44	65	592	19	74	112	1664	183	104	159	4288	33	134
19	184	57	46	66	608	37	76	113	1696	55	954	160	4352	477	408
20	192	23	48	67	624	41	234	114	1728	127	96	161	4416	35	138
21	200	13	50	68	640	39	80	115	1760	27	110	162	4480	233	280
22	208	27	52	69	656	185	82	116	1792	29	112	163	4544	357	142
23	216	11	36	70	672	43	252	117	1824	29	114	164	4608	337	480
24	224	27	56	71	688	21	86	118	1856	57	116	165	4672	37	146
25	232	85	58	72	704	155	44	119	1888	45	354	166	4736	71	444
26	240	29	60	73	720	79	120	120	1920	31	120	167	4800	71	120
27	248	33	62	74	736	139	92	121	1952	59	610	168	4864	37	152
28	256	15	32	75	752	23	94	122	1984	185	124	169	4928	39	462
29	264	17	198	76	768	217	48	123	2016	113	420	170	4992	127	234
30	272	33	68	77	784	25	98	124	2048	31	64	171	5056	39	158
31	280	103	210	78	800	17	80	125	2112	17	66	172	5120	39	80
32	288	19	36	79	816	127	102	126	2176	171	136	173	5184	31	96
33	296	19	74	80	832	25	52	127	2240	209	420	174	5248	113	902
34	304	37	76	81	848	239	106	128	2304	253	216	175	5312	41	166
35	312	19	78	82	864	17	48	129	2368	367	444	176	5376	251	336
36	320	21	120	83	880	137	110	130	2432	265	456	177	5440	43	170
37	328	21	82	84	896	215	112	131	2496	181	468	178	5504	21	86
38	336	115	84	85	912	29	114	132	2560	39	80	179	5568	43	174
39	344	193	86	86	928	15	58	133	2624	27	164	180	5632	45	176
40	352	21	44	87	944	147	118	134	2688	127	504	181	5696	45	178
41	360	133	90	88	960	29	60	135	2752	143	172	182	5760	161	120
42	368	81	46	89	976	59	122	136	2816	43	88	183	5824	89	182
43	376	45	94	90	992	65	124	137	2880	29	300	184	5888	323	184
44	384	23	48	91	1008	55	84	138	2944	45	92	185	5952	47	186
45	392	243	98	92	1024	31	64	139	3008	157	188	186	6016	23	94
46	400	151	40	93	1056	17	66	140	3072	47	96	187	6080	47	190
47	408	155	102	94	1088	171	204	141	3136	13	28	188	6144	263	480

En la Tabla 1, los parámetros f_1 y f_2 se pueden variar según el valor K que es un tamaño de datos de entrada.

Se describirá ahora un tercer método de cálculo de un tamaño de bloque de código K_- cuando el tamaño de bloque de código K tiene un tamaño específico K_+ o K_- en paso S302.

El valor K_- se puede fijar a un valor máximo entre los valores K que son menores que K_+ calculados en cualquiera de la Ecuación 14 a la Ecuación 16. El valor K puede usar valores mostrados en la Tabla 1. La Ecuación 17 siguiente ilustra un método de cálculo del valor K_-.

[Ecuación 17]

K_- es un valor máximo de K,

donde K satisface $K < K_+$

Cuando se calcula K_- usando valores mostrados en la Tabla 1 y la Ecuación 17, la segmentación del bloque de datos de entrada (o bits de entrada) se puede realizar de manera precisa en consideración al tamaño de un bloque de código al que se une un código de CRC.

Se ha hecho una descripción de un método de cálculo de los tamaños de bloque de código K_+ y K_- cuando el tamaño del bloque de código tiene un tamaño específico a través de la Ecuación 14 a la Ecuación 17. En este caso, es necesario obtener los números C_+ y C_- de bloques de código que tienen los tamaños de bloque de código K_+ y K_-, respectivamente, para dividir de manera precisa el bloque de datos de entrada.

En lo sucesivo, se describirán métodos de cálculo del número C_- de bloques de código que tienen el tamaño específico K_- en el paso S302.

La Ecuación 18 ilustra un primer método de cálculo del número C_- de los segundos bloques de código que tienen el tamaño K_-.

[Ecuación 18]

$$C_- = \left\lfloor \frac{C \cdot K_+ - B'}{\Delta_K} \right\rfloor$$

El número C_- de los segundos bloques de código se puede calcular por un entero redondeando por defecto el resultado de dividir un valor del número total C de bloques de código multiplicado por el primer tamaño de bloque de código K_+ menos el tamaño de secuencia de bits de entrada modificado B' por una diferencia de valor Δ_K entre K_+ y K_-. El tamaño de secuencia de bits de entrada modificado B' es un valor temporal para calcular el número y tamaño de bloques de código.

En la Ecuación 18, el número C_- de bloques de código que tiene el tamaño K_- se calcula usando el tamaño de secuencia de bits de entrada modificado B'. Por lo tanto, la segmentación de la secuencia de bloque de datos de entrada (o bits de entrada) se puede realizar de manera precisa en consideración al tamaño de CRC L incluido en cada bloque de datos.

La Ecuación 18 se puede expresar por la Ecuación 19 siguiente.

[Ecuación 19]

$$C_- = \left\lfloor \frac{C \cdot K_+ - (B + C \cdot L)}{\Delta_K} \right\rfloor$$

La Ecuación 19 muestra que el tamaño de secuencia de bits de entrada modificado B' se fija a un valor obtenido multiplicando el número C de bloques de código por el tamaño de CRC L y luego añadiendo el tamaño de bloque de datos B al resultado multiplicado.

Se describirá ahora un segundo método de cálculo del número C_- de bloques de código que tiene el segundo tamaño de bloque de código K_- en el paso S302.

La Ecuación 20 ilustra un ejemplo de cálculo del número C_- de los segundos bloques de código que tienen el tamaño K_-.

[Ecuación 20]

$$C_- = \left\lfloor \frac{C \cdot (K_+ - L) - B'}{D} \right\rfloor$$

El número C_- de los segundos bloques de código se puede calcular por un entero redondeando por defecto el resultado de dividir un valor de C* (K_+ – L) – B' por una diferencia D entre K_+ y K_-.

Es decir, la Ecuación 20 muestra un método de cálculo del número C_- de los segundos bloques de código considerando el tamaño de CRC L a ser incluido en el primer bloque de código.

La Ecuación 21 ilustra un ejemplo de expresión de la Ecuación 20 usando el tamaño de secuencia de bits de entrada modificado B'. Es decir, el tamaño de secuencia de bits de entrada modificado B' se fija a un valor obtenido añadiendo el tamaño de bloque de datos B al resultado de multiplicar el número C de bloques de código por el tamaño de CRC L.

[Ecuación 21]

$$C_- = \left\lfloor \frac{C \cdot (K_+ - L) - (B + C \cdot L)}{D} \right\rfloor$$

Un método de cálculo del número C_+ de los primeros bloques de código que tiene el tamaño específico K_+ en el paso S302 es como sigue.

[Ecuación 22]

$$C_+ = C - C_-$$

En la Ecuación 22, el número C_+ de los primeros bloques de código que tiene el tamaño específico K_+ se calcula restando el número C_- de los segundos bloques de código calculados en la Ecuación 19 a la Ecuación 21 del número total C de bloques de código.

Con referencia de nuevo a la FIG. 3, el tamaño de cada bloque de código calculado en el paso S302 puede se fijo o cada bloque de código puede tener un tamaño específico K_+ o K_-. El tamaño de bloque de datos K se puede determinar según los requisitos del sistema.

Al dividir el bloque de datos, un valor obtenido añadiendo los tamaños de todos los bloques de código puede ser mayor que el tamaño de secuencia de bits de entrada modificado B' para calcular el número C de bloques de código y el tamaño de bloque de código K. En este caso, se calcula una longitud F de bits de relleno que corresponden a una diferencia entre el valor obtenido añadiendo los tamaños de todos los bloques de código y el tamaño de secuencia de bits de entrada modificado B' (paso S303).

Los bits de relleno sirven para igualar un bloque de datos de entrada inicialmente con los tamaños de bloques de código cuando se une un código de detección de errores a los bloques de código segmentados del bloque de datos. Si el número de bloques de código es 0, la longitud de bits de relleno F también es 0.

Ahora se describirán métodos de cálculo de la longitud de bits de relleno F.

La longitud de bits de relleno F se puede calcular restando el tamaño de secuencia de bits de entrada modificado B' del resultado de multiplicar el número C de bloques de código por el tamaño de bloque de código K.

La Ecuación 23 siguiente ilustra un primer método de cálculo de la longitud de bits de relleno F.

[Ecuación 23]

$$F = C * K - B'$$

La Ecuación 23 muestra el método de cálculo de la longitud de bits de relleno cuando el bloque de datos de entrada se segmenta en bloques de código de la misma longitud.

La Ecuación 24 siguiente expresa la Ecuación 23 usando el tamaño de secuencia de bits de entrada modificado B'.

[Ecuación 24]

$$F = C * K – (B + C * L)$$

Un segundo método de cálculo de la longitud de bits de relleno F en el paso S303 es como sigue.

La Ecuación 25 ilustra un ejemplo de cálculo de la longitud de bits de relleno F. Esto es, la Ecuación 25 muestra un método de cálculo de la longitud de bits de relleno cuando el bloque de datos de entrada (o bits de entrada) tiene un tamaño específico (por ejemplo, K_+ o K_-).

[Ecuación 25]

$$F = C_+ * K_+ + C_- * K_- – B'$$

La longitud de bits de relleno F se puede calcular restando el tamaño de secuencia de bits de entrada modificado B' de la suma de los tamaños de todos los bloques de código. Es decir, la longitud de bit de relleno F se puede calcular restando el tamaño de secuencia de bits de entrada modificado B' de un valor de $(C_+ \times K_+) + (C_- \times K_-)$.

Usando la Ecuación 25, las realizaciones ejemplares de la presente invención se pueden aplicar incluso cuando los tamaños de bloques de código segmentados de un bloque de datos son diferentes. Dado que la longitud de bits de relleno se puede calcular cuando se incluye un código de detección de errores en los bloques de código, se pueden generar de manera precisa los bloques de código.

En la Ecuación 26 siguiente, el tamaño de secuencia de bits de entrada modificado B' se fija a un valor obtenido multiplicando el tamaño de código de detección de errores L por el número C de bloques de código y luego añadiendo el tamaño de bloque de datos (o bits de entrada) B al resultado multiplicado.

[Ecuación 26]

$$F = C_+ * K_+ + C_- * K_- – (B + C * L)$$

Con referencia de nuevo a la FIG. 3, si el número de bloques de código, el tamaño de bloque de código, y la longitud de bits de relleno son determinados, la unidad de segmentación de bloque de datos 320 puede asignar secuencialmente datos a los bloques de código (paso S304).

En el paso S304, si el bloque de datos (300) se compone de un bloque de código incluyendo un código de detección de errores, los datos se asignan al bloque de datos 300 y no se introduce a la unidad de conexión de CRC 340. Este es el caso en el que el tamaño del bloque de datos 300 es menor o igual que el tamaño máximo del bloque de código 360. Por lo tanto, el bloque de datos se segmenta mediante la unidad de segmentación de bloque de datos 320 y se omite un paso para unir un código de CRC a cada bloque de código segmentado. Es decir, el bloque de datos de entrada 300 se usa directamente como el bloque de código 360. Por consiguiente, solamente está presente en el bloque de código 360 un código de CRC incluido inicialmente y no se necesita unir el código de CRC generado desde la unidad de conexión de CRC 340 al bloque de código 360.

Si los datos se asignan a los bloques de código, los bloques de código se introducen a la unidad de conexión de CRC 340. La unidad de conexión de CRC 340 genera bits de paridad de CRC de una longitud prescrita usando una ecuación de generación de CRC en base a los bloques de código (paso S305).

La unidad de conexión de CRC une los bits de paridad de CRC generados en el paso S305 a una parte trasera de los bloques de código en una dirección hacia delante o hacia atrás. Finalmente, se generan los bloques de código unidos con CRC (paso S306).

<Segunda Realización>

La FIG. 4 es un diagrama que ilustra un ejemplo de un proceso de unión de códigos CRC a bloques de código según una realización ejemplar de la presente invención.

Con referencia a la FIG. 4, un bloque de datos (un primer bloque de datos 400) se introduce a una unidad de segmentación de bloque de datos 420 y se segmenta en bloques de código (un segundo bloque de datos). Los bloques de código 430 en consideración a un tamaño de CRC se introducen a la unidad de conexión de CRC 440 para generar un bloque de código unido con CRC 460. Este proceso es similar al método de la FIG. 3, excepto para los pasos S304 y S306 en la FIG. 3.

Otra realización ejemplar de la presente invención asume que el tamaño de bloque de código 430 introducido a la unidad de conexión de CRC es igual al tamaño del bloque de código unido con CRC 460. Es decir, se incluye un tamaño de CRC en el tamaño del bloque de código introducido a la unidad de conexión de CRC. Por consiguiente, cuando se calcula el número y tamaño de bloques de código con respecto a los datos de entrada, es deseable para la unidad de segmentación de bloque de datos considerar el tamaño de un código de CRC a ser unido dentro de cada bloque de código.

Si el número C de bloques de código, tamaño de bloque de código K, y longitud de bits de relleno F son determinados, la unidad de segmentación de bloque de datos 420 asignó secuencialmente datos a los bloques de código. En este momento, se asignan bits de relleno y datos al primer bloque de los bloques de código.

En otra realización ejemplar de la presente invención, se asignan datos a los bloques de código en consideración al tamaño de CRC L a ser unido en la unidad de conexión de CRC. Por lo tanto, se asignan los bits de relleno y los datos al primer bloque de código y se asigna un valor que consta de ceros o unos correspondiente al tamaño del código de CRC para designar un área de CRC. No obstante, el valor solamente indica el área de CRC y no significa que un código de CRC esté unido al área de CRC.

Los datos se asignan secuencialmente comenzando con el segundo bloque de código al último bloque de código. Un valor que consta de ceros o unos indica que un área de CRC está asignada en cada bloque de código para asegurar el área de CRC.

La unidad de conexión de CRC une códigos de CRC a los bloques de código. En este caso, se generan bits de paridad de CRC que corresponden al área de CRC rellenada con ceros o unos y los bits de paridad de CRC se unen en una dirección hacia delante o hacia atrás. Este proceso se realiza para cada bloque de código. El tamaño L del código de CRC puede ser 24 bits.

A través del método descrito anteriormente, el bloque de datos que tiene un tamaño prescrito se segmenta en los bloques de código unidos con CRC a través de la unidad de segmentación de bloque de datos 420 y la unidad de conexión de CRC 440.

No obstante, si el bloque de datos 400 es menor o igual que el tamaño de bloque de código máximo, el bloque de datos no pasa a través de la unidad de segmentación de bloque de datos 420 y la unidad de conexión de CRC 440. Esto es, si el bloque de datos 400 está compuesto de un bloque de código que incluye un código de CRC, dado que el bloque de datos 400 ya ha incluido el código de CRC, solamente se asignan los datos al bloque de datos 400 y el código de CRC no se une en la unidad de conexión de CRC.

Cuando se configura el bloque de código introducido a la unidad de conexión de CRC, un ejemplo detallado de inclusión del tamaño de CRC en el bloque de código es como sigue.

La Tabla 2 ilustra una ecuación de configuración del bloque de código considerando el tamaño de CRC.

[Tabla 2]

Sintaxis	Valor	Notas
para k = 0 a F-1		
O_{0k} = 0 o 1		
fin para		
K = F		
S = 0		
para r = 0 a C- 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
fin si		
mientras (k < K_r)		
si (k < K_r – L)		
$O_{rk} = b_s$		
s = s + 1		
Además		

Sintaxis	Valor	Notas
O_{rk} = 0 o 1		
fin si		
k = k + 1		
fin mientras		
k = 0		
fin para		

La Tabla 3 ilustra una ecuación de configuración del bloque de código considerando el tamaño de CRC cuando el bloque de datos se compone de un bloque de código incluyendo un código de detección de errores.

[Tabla 3]

Sintaxis	Valor	Notas
si C = 1		
L = 0		
fin si		
para k = 0 a F-1		
O_{0k} = 0 o 1		
fin para		
K = F		
S = 0		
para r = 0 a C-1		
si r < C_-		
K_r = K_-		
además		
K_r = K_+		
fin si		
mientras (k < K_r)		
si (k < K_r – L)		
O_{rk} = b_s		
s = s + 1		
Además		
O_{rk} = 0 o 1		
fin si		
k = k + 1		
fin mientras		
k = 0		

Sintaxis	Valor	Notas
fin para		

Los parámetros principales usados en la Tabla 2 y la Tabla 3 son como sigue. Un parámetro 'F' indica la longitud de bits de relleno, 'O_{rk}' indica una salida de una unidad de segmentación de bloque de datos, 'r' indica un número de bloque de código, y 'k' indica un número de bits del bloque de orden r.

Se asume que el tamaño de bloque de código es K_+ y K_- (donde K_- es menor que K_+). Los parámetros C_+ y C_- indican los números de bloques de código específicos, es decir, los números de bloques de código que tienen los tamaños K_+ y K_-, respectivamente. Un parámetro 'L' mostrado en la Tabla 2 y Tabla 3 indica el tamaño de un código de CRC a ser unido a cada bloque de código y 'K_r' indica un tamaño a ser aplicado a un bloque de código.

Un ejemplo de unión de un código de CRC a cada bloque de código generado en consideración a un tamaño de CRC se ilustra en la Ecuación 27.

[Ecuación 27]

$$O_{rk} = O_{rk}; \; k = 0, 1, 2, \ldots, K_r - L - 1$$

$$O_{rk} = P_{r(Kr-k-1)}; \; k = K_r - L, K_r - L + 1, K_r - L + 2, \ldots, K_r - 1 \; (= K_r - L + L - 1)$$

Cuando el tamaño de bloque de datos B es menor que el tamaño de bloque de código máximo Z, un ejemplo de unión de un código de CRC a cada bloque de código generado considerando el tamaño de CRC L se ilustra en la Ecuación 28.

[Ecuación 28]

si C = 1

derivación

además

$O_{rk} = O_{rk}$; k = 0, 1, 2, …, $K_r - L - 1$

$O_{rk} = P_{r(Kr-k-1)}$; k = $K_r - L$, $K_r - L + 1$, $K_r - L + 2$, …, $K_r - 1$ (= $K_r - L + L - 1$)

fin si

Al calcular los parámetros para unión de CRC, se asume que el tamaño aplicado al bloque de código es K_r y un tamaño de CRC unido a cada bloque de código es L. Los bits de entrada se pueden indicar mediante O_{r0}, O_{r1}, O_{r2}, …, O_{rKr-1}. Los bits de paridad se pueden indicar mediante p_{r0}, p_{r1}, p_{r2}, …, p_{rL-1}. Los bits de paridad de CRC se pueden generar usando una ecuación de generación de CRC en base a los bits de entrada. Después de que se unen los bits de paridad de CRC, los bits se pueden indicar mediante O_{r0}, O_{r1}, O_{r2}, …, O_{rKr-1} o C_{r0}, C_{r1}, C_{r2}, …, C_{rKr-1} en el mismo espacio o diferentes espacios. Los bits de paridad de CRC se pueden unir en una dirección hacia delante o hacia atrás de un bloque de código según los requisitos del sistema.

<Tercera Realización>

La FIG. 5 es un diagrama que ilustra otro ejemplo de un proceso de unión de códigos de CRC a bloques de código según una realización ejemplar de la presente invención.

En la FIG. 5, el tamaño de un bloque de código de entrada 530 introducido a una unidad de conexión de CRC 540 tiene un valor obtenido restando un tamaño de CRC L del tamaño de un bloque de código de salida 560. Aunque la operación de una unidad de segmentación de bloque de datos 520 para dividir un bloque de datos (un primer bloque de datos 500) es similar a la operación de las unidades de segmentación de bloque de datos mostradas en la FIG. 3 y FIG. 4, un método de asignación de datos a un bloque de código (un segundo bloque de datos) y un método de generación de un bloque unido con CRC son diferentes de los métodos mostrados en la FIG. 3 y FIG. 4.

Esto es, aunque se considera un tamaño de CRC para dividir el bloque de datos, un área de CRC no se asegura en el tamaño de los bloques de código introducidos a la unidad de conexión de CRC, lo cual es diferente del método mostrado en la FIG. 4.

Con referencia a la FIG. 5, el primer bloque de datos 500 se introduce a la unidad de segmentación de bloque de datos 520 y segmenta en el bloque de código 530 y el bloque de código se introduce a la unidad de conexión de CRC 540, generando por ello el bloque de código unido con CRC.

En la FIG. 5 si el bloque de datos 500 se introduce a la unidad de segmentación de bloque de datos 520, la unidad de segmentación de bloque de datos 520 calcula el número C y el tamaño Z de bloques de código en consideración a un tamaño de CRC. La unidad de segmentación de bloque de datos 520 también calcula una longitud de bits de relleno F usando el número C y el tamaño K de bloques de código. La unidad de segmentación de bloque de datos 520 asigna los bits de relleno y datos al primer bloque entre los bloques de código. En este caso, se asignan secuencialmente datos que corresponden a una longitud que resta la longitud de bits de código y el tamaño de CRC L del tamaño de bloque de código. El tamaño de CRC L puede ser 24 bits.

La unidad de segmentación de bloque de datos asigna secuencialmente datos que corresponden a una longitud que resta el tamaño de CRC del tamaño de bloque de código al segundo código. Un proceso de asignación de datos se repite por el número de los otros bloques de código. A través del proceso de asignación de datos, se configuran los bloques de código introducidos a la unidad de conexión de CRC.

Es decir, el número C de bloques de código calculado por la unidad de segmentación de bloque de datos se entrega a la unidad de conexión de CRC de manera que la unidad de conexión de CRC puede usar el número C de bloques de código para unir códigos de CRC a los bloques de código. Suponiendo que el tamaño de cada bloque de código introducido a la unidad de conexión de CRC es constante, el tamaño del bloque de código de entrada 530 se obtiene multiplicando el número C de bloques de código por el resultado de restar el tamaño de CRC K del tamaño de bloque de código K. Alternativamente, el tamaño del bloque de código de entrada 530 se puede fijar a la suma de tamaños de bloque de código cada uno que tiene un valor calculado restando el tamaño de CRC L del tamaño de bloque de código K.

Cuando se calcula el tamaño entero del bloque de código 530 introducido a la unidad de conexión de CRC según la realización ejemplar de la presente invención descrita en la FIG. 5, el bloque de código 530 tiene un tamaño menor que el tamaño calculado por el método en la FIG. 4 por el tamaño de códigos de CRC que van a ser unidos a los bloques de código.

Con referencia a la FIG. 5, la unidad de conexión de CRC genera un tamaño prescrito de bits de paridad de CRC usando una ecuación de generación de CRC en base a los bloques de código de entrada y el número de bloques de código. La unidad de conexión de CRC une los bits de paridad al final de cada bloque de código en una dirección hacia delante o hacia atrás.

Usando el método descrito con referencia a la FIG. 5, el bloque de datos que tiene un tamaño prescrito se compone de bloques de código unidos con CRC que pasan a través de la unidad de segmentación de bloque de datos y la unidad de conexión de CRC.

No obstante, si el bloque de datos se compone de un bloque de código que incluye un código de CRC, se omite un paso de pasar a través de la unidad de conexión de CRC 540 después de asignar datos. Esto es, si el tamaño de bloque de datos es menor o igual que el tamaño de bloque de código máximo, es innecesario dividir el bloque de datos, y el código de CRC incluido originalmente en el bloque de datos se usa sin unir repetidamente los códigos de CRC a través de la unidad de conexión de CRC 540.

Un ejemplo detallado de configuración de un bloque de código introducido a la unidad de conexión de CRC cuando un código de CRC no se incluye en el bloque de código aunque el tamaño de CRC se considera para configurar el bloque de código se describirá ahora según otra realización ejemplar de la presente invención.

La Tabla 4 siguiente ilustra una ecuación de configuración de un bloque de código que no incluye un código de CRC aunque el tamaño de CRC se considera para generar el bloque de código.

[Tabla 4]

Sintaxis	Valor	Notas
para k = 0 a F-1		
O_{0k} = 0 o 1		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		
si r < C-		
K_r = K-		

Sintaxis	Valor	Notas
Además		
$K_r = K_+$		
fin si		
mientras ($k < K_r - L$)		
$O_{rk} = b_s$		
$k = k + 1$		
$s = s + 1$		
fin mientras		
$k = 0$		
fin para		

La Tabla 5 siguiente ilustra una ecuación de configuración de un bloque de código que tiene en consideración un tamaño de CRC pero no incluye un código de CRC cuando el bloque de datos se compone de un bloque de código que incluye el código de CRC.

[Tabla 5]

Sintaxis	Valor	Notas
si $C = 1$		
$L = 0$		
fin si		
para $k = 0$ a $F-1$		
$O_{0k} = 0$ o 1		
fin para		
$k = F$		
$s = 0$		
para $r = 0$ a $C-1$		
si $r < C_-$		
$K_r = K_-$		
Además		
$K_r = K_+$		
fin si		
mientras ($k < K_r - L$)		
$O_{rk} = b_s$		
$k = k + 1$		
$s = s + 1$		
fin mientras		

Sintaxis	Valor	Notas
k = 0		
fin para		

Los parámetros principales usados en la Tabla 4 y Tabla 5 son como sigue. 'F' indica la longitud de bits de relleno, 'O_{rk}' indica una salida de la unidad de segmentación de bloque de datos, 'r' indica un número de bloque de código, y 'k' indica un número de bits del bloque de orden r.

Se asume que los bloques de código tienen tamaños K_+ y K_- (donde K_- es menor que K_+). Los parámetros 'C_+' y 'C_-' indican los números de bloques de código específicos, es decir, los números de bloques de código que tienen los tamaños K_+ y K_-, respectivamente. Un parámetro 'L' mostrado en la Tabla 4 y Tabla 5 indica un tamaño de CRC a ser unido a cada bloque de código y 'K_r' indica un tamaño a ser aplicado a un bloque de código.

Con referencia a la FIG. 5, se configura el bloque de código que considera el tamaño de CRC pero no incluye el código de CRC y la Ecuación 27 se puede usar para unir el código de CRC a los bloques de código.

Si el bloque de datos no está segmentado y se compone de un bloque de código que incluye el código de CRC, se configura el bloque de código que tiene en consideración el tamaño de CRC pero no incluye el código de CRC y la Ecuación 28 se puede usar para unir un código de CRC al bloque de código.

Los bits después del conexión de CRC se pueden indicar mediante O_{r0}, O_{r1}, O_{r2},…, O_{rKr-1} o C_{r0}, C_{r1}, C_{r2}, …, C_{rKr-1}. Esto es, los bits se pueden comprender en el mismo espacio o diferentes espacios.

<Cuarta Realización>

La FIG. 6 es un diagrama que ilustra un proceso de segmentación de un bloque de datos y unión de códigos de CRC en consideración de un tamaño de CRC según una realización ejemplar de la presente invención.

Un bloque de datos (un primer bloque de datos 600) que tiene un tamaño prescrito se introduce a un módulo de función (una unidad de segmentación de bloque de datos y de conexión de CRC) 620 para generar un bloque de código (un segundo bloque de datos 640). El módulo de función 620 segmenta el bloque de datos y une simultáneamente códigos de CRC. El bloque de datos 600 puede incluir un código de CRC dentro del mismo antes de ser introducido al módulo de función 620.

Si el bloque de datos 600 se introduce al módulo de función 620, el módulo de función 620 calcula el número C de bloques de código considerando un tamaño de CRC L (paso S601). El módulo de función 620 calcula un tamaño de bloque de código K usando el número C de bloques de código (paso S602). El módulo de función 620 calcula una longitud de bits de relleno F usando el número C de bloques de código y el tamaño de bloque de código K (paso S603).

Los métodos de cálculo del número C de bloques de código, el tamaño de bloque de código K, y la longitud de bits de relleno F usada en los pasos S601 a S603 pueden usar uno o más métodos usados en la FIG. 3 a FIG. 5. No obstante, un método de asignación de datos y códigos de CRC a bloques de código en el paso S604 es diferente de los métodos descritos con referencia a la FIG. 3 a la FIG. 5. El tamaño de CRC L puede ser de 24 bits.

Con referencia a la FIG. 6, el paso S604 para asignación de datos y códigos de CRC a bloques de código es como sigue.

Mientras que los bits de relleno se unen al principio del primer bloque de código y datos que corresponden a una longitud excepto para el tamaño de CRC L se asignan en el primer bloque de código, un código de CRC se une al final del primer bloque de código en una dirección hacia delante o hacia atrás. En el segundo bloque de código, se asignan los datos que corresponden a una longitud excepto para el tamaño de CRC y el código de CRC se une al final del segundo bloque de código en una dirección hacia delante o hacia atrás. El proceso anterior para asignar datos en los bloques de código que sigue el segundo bloque de código se repite para cada uno del número C de los bloques de código restantes. El tamaño de bloque de código K puede tener el mismo tamaño o valores específicos (por ejemplo, K_+ o K_-). Los valores K_+ y K_- indican una cantidad de variación diminuta de K. Usando el método mostrado en la FIG. 6, el bloque de datos que tiene un tamaño prescrito se puede componer de un bloque de código unido con CRC 640 a través del módulo de función 620. No obstante, si el bloque de datos se compone de un bloque de código que incluye un código de CRC, el código de CRC no se une después de que se asignan los datos, debido a que el bloque de datos incluye el código de CRC.

Después de que se genera el bloque de código, se realiza la codificación de canal. Codificación de canal se refiere a un proceso de conversión de un código original generado por un lado de transmisión de manera que un lado de recepción puede detectar y/o corregir errores durante la transmisión de datos a través de un canal. Es decir,

codificación de canal se refiere a un proceso para superar un error en entornos de canal que tienen potencia limitada o anchos de banda limitados.

Se pueden aplicar diversos métodos para codificación de canal. La codificación de canal incluye codificación lineal y codificación cíclica de un tipo sin memoria, y codificación de convolución de mordedura de cola y turbo codificación de un tipo de memoria.

La FIG. 7 es un diagrama que ilustra un proceso de conversión de un bloque de datos en bloques de código en consideración a un tamaño de CRC según una realización ejemplar de la presente invención.

Con referencia a la FIG. 7, si un bloque de datos (un primer bloque de datos 700) se introduce a un módulo de función (conexión de CRC y segmentación de bloque de datos) 720, el bloque de datos se segmenta en un bloque de código (un segundo bloque de datos 740) en consideración a un tamaño de CRC L. Un proceso de conversión del bloque de datos en el bloque de código a través de la unidad de conexión de CRC y segmentación de bloque de datos puede usar el proceso mostrado en la FIG. 6. En la FIG. 7, el bloque de datos 700 puede incluir un código de CRC para el bloque de datos antes de que el bloque de datos se introduzca al módulo de función.

Se describirá ahora un ejemplo de realización de segmentación de bloque de datos y conexión de CRC en un módulo de función.

La Tabla 6 siguiente muestra una ecuación de configuración del bloque de datos para segmentar el bloque de datos y unir el código de CRC cuando la segmentación de bloque de datos y conexión de código de CRC se realizan en un módulo de función 720.

[Tabla 6]

Sintaxis	Valor	Notas
para k = 0 a F-1		
O_{0k} = 0 o 1		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
fin si		
mientras (k < K_r)		
si (k < K_r – L)		
$O_{rk} = b_s$		Asignar b_s a O_{rk} y generar simultáneamente bits de paridad
s = s + 1		
además		
$O_{rk} = p_{(L-1-(k-(Kr-L)))}$		
fin si		
k = k + 1		
fin mientras		

Sintaxis	Valor	Notas
k = 0		
fin para		

La Tabla 7 siguiente muestra una ecuación de configuración del bloque de código para segmentar el bloque de datos y unir los códigos de CRC, cuando el bloque de datos no se segmenta y se compone de un bloque de código que incluye un código de CRC (es decir, C=1) y cuando la segmentación del bloque de datos y conexión de código de CRC se realizan en un módulo de función 720.

[Tabla 7]

Sintaxis	Valor	Notas
si C = 1		
L = 0		
fin si		
para k = 0 a F-1		
O_{0k} = 0 o 1		
fin para		
k = F		
s = 0		
para r = 0 a C-1		
si r < C-		
$K_r = K_-$		
además		
$K_r = K_+$		
fin si		
mientras (k < K_r)		
si (k < K_r – L)		
si C = 1		
$O_{rk} = b_s$		Asignar b_s a O_{rk} y generar simultáneamente bits de paridad
además		
$O_{rk} = b_s$		
fin si		
s = s + 1		
además		
O_{rk} = pr(Kr-k-1)		
fin si		

Sintaxis	Valor	Notas
k = k + 1		
fin mientras		
k = 0		
fin para		

Los parámetros principales usados en la Tabla 6 y Tabla 7 son como sigue. Un parámetro 'F' indica la longitud de bits de relleno, 'O_{rk}' indica una salida de una unidad de segmentación de bloque de datos, 'r' indica un número de bloque de código, y 'k' indica un número de bits del bloque de orden r.

Se asume que los bloques de código tienen tamaños K_+ y K_- (donde K_- es menor que K_+). Los parámetros 'C_+' y 'C_-' indican los números de bloques de código específicos, es decir, los números de bloques de código que tienen los tamaños K_+ y K_-, respectivamente. Un parámetro 'L' mostrado en la Tabla 6 y Tabla 7 indica un tamaño de CRC a ser unido por bloque de código y 'K_r' indica un tamaño a ser aplicado a un bloque de código final.

Los bits de paridad de CRC para conexión de código de CRC se pueden indicar mediante p_{r0}, p_{r1}, p_{r2}, …, p_{rL-1}. Los bits de paridad de CRC se pueden generar usando una ecuación de generación de CRC en base a los bloques de entrada. Los bits después de la segmentación de datos y conexión de CRC se pueden indicar mediante O_{r0}, O_{r1}, O_{r2}, …, O_{rKr-1} o C_{r0}, C_{r1}, C_{r2}, …, C_{rKr-1}. Esto es, los bits se pueden incluir en el mismo espacio o diferentes espacios.

<Quinta Realización>

Una realización ejemplar adicional de la presente invención se puede construir mediante una combinación de los métodos descritos en la primera a cuarta realizaciones.

Los datos de un tamaño grande se pueden segmentar, antes de la transmisión, en datos de tamaños adecuados según los requisitos del sistema para transferir datos de manera efectiva. Por consiguiente, es necesario segmentar un bloque de datos de entrada mediante un método adecuado. Puede ser importante determinar qué método va a ser usado para segmentación de bloque de datos.

La unidad de segmentación de bloque de datos y conexión de CRC según la realización ejemplar adicional de la presente invención es un módulo de función para segmentar un bloque de datos lógico (o bits de entrada) y unir códigos de CRC. El módulo de función puede segmentar el bloque de datos de entrada en bloques de código de un número adecuado en consideración al tamaño de los códigos de detección de errores (por ejemplo, códigos de CRC) incluidos en los bloques de código a ser segmentados.

Si los bits de entrada se segmentan en bloques (o segmentos) de código, se puede determinar un tamaño de bloque de código máximo Z según los requisitos del sistema. En las realizaciones ejemplares de la presente invención, el tamaño de bloque de código máximo es deseablemente de 6.144 bits.

Una secuencia de bits de entrada (o bloque de datos) introducida al módulo de función se puede indicar mediante b_0, b_1, b_2, …, b_{B-1}. El tamaño de los bits de entrada se puede indicar por 'B' (donde B está por encima de 1). Si el tamaño de bits de entrada B es mayor que el tamaño de bloque de código máximo Z, se puede realizar una segmentación de bits de entrada. El tamaño de CRC L puede ser de 24 bits. Esto es, un código de CRC de 24 bits, que es un tipo de un código de corrección de errores, se puede unir a cada bloque de código generado mediante segmentación de los bits de entrada.

Si la longitud de bits de relleno no es 0, se pueden añadir los bits de relleno al principio del primer bloque de código. Si el tamaño de bits de entrada B es menor que 40 bits, los bits de relleno se añaden al principio del bloque de código. Los bits de relleno se fijan a nulo en la entrada al módulo de función.

El número total C de bloques de código generados mediante segmentación de los bits de entrada se puede calcular mediante la Ecuación 29 siguiente.

[Ecuación 29]

si $B \leq Z$ ↵

$L = 0$ ↵

$C = 1$ ↵

B' = B ↵

además ↵

L = 24 ↵

$C = \lceil B/(Z-L) \rceil$ ↵

B' = B + C * L ↵

fin si ↵

En la Ecuación 29, B indica el tamaño de secuencia de bits de entrada (o bloque de datos) y B' indica un valor obtenido añadiendo el tamaño de secuencia de bits de entrada B al resultado de multiplicar el número C de bloques de código por el tamaño de CRC L. Es decir, B' indica un tamaño de secuencia de bits de entrada modificado para calcular el número y tamaño de bloques de código.

En la Ecuación 29, si el tamaño de bits de entrada es menor que el tamaño de bloque de código máximo Z, el tamaño de los códigos de detección de errores a ser unidos a los bloques de código se puede fijar a 0 y el número total C de bloques de código se puede fijar a 1. El tamaño de secuencia de bits de entrada modificado B' se puede fijar para ser igual al tamaño de secuencia de bits de entrada B.

Si el tamaño de bits de entrada B es mayor que el tamaño de bloque de código máximo Z, el tamaño de CRC L se puede fijar a 24 bits y el número total C de bloques de código se puede fijar a un valor obtenido redondeando por exceso un valor obtenido dividiendo el tamaño de secuencia de bits de entrada B por el resultado de restar el tamaño de CRC del tamaño de bloque de código máximo Z. El tamaño de secuencia de bits de entrada modificado B' se puede fijar a un valor obtenido añadiendo el tamaño de bits de entrada B al resultado de multiplicar el número total C de bloques de código por el tamaño de CRC L.

Si el número de bloques de código no es 0, los bloques de código generados a partir del bloque de función se pueden indicar mediante c_{r0}, c_{r1}, c_{r2}, c_{r3},..., $c_{r(Kr-1)}$ (donde r indica un número de bloque de código y K_r indica el tamaño del bloque de código de orden r).

El módulo de función debería calcular el tamaño de cada bloque de código después de calcular el número de bloques de código a través de la Ecuación 29. Los bits de entrada se pueden segmentar en bloques de código que tiene el mismo tamaño o que tienen un tamaño específico (por ejemplo, K_+ o K_-). Es evidente que los bloques de código tienen diversos tamaños según los requisitos del sistema o usuario.

En otra realización ejemplar de la presente invención, los tamaños de bloques de código pueden ser K_+ o K_-. En este caso, el número C de bloques de código no debería ser 0.

En una realización ejemplar de la presente invención, el tamaño de los primeros bloques de código (o primeros segmentos) se puede indicar por K_+. El tamaño K_+ se puede determinar a partir de los valores K en la Tabla 1. El tamaño de K_+ se puede determinar por un valor mínimo entre los valores K que satisfacen la condición de que un valor obtenido multiplicando el número C de bloques de código por el tamaño de bloque de código K está por encima del tamaño de secuencia de bits de entrada modificado B'.

Si el número C de bloques de código es 1, el número C_+ de bloques de código que tienen el tamaño K_+ es 1 y el número C_- de bloques de código que tienen el tamaño K_- es 0.

Si el número de bloques de código es 2 o más (C>1), el tamaño K_- de los segundos bloques de código (o segundos segmentos) se puede determinar mediante los valores K mostrados en la Tabla 1. Deseablemente, el tamaño K_- tiene un valor máximo entre valores K menores que el tamaño K_+. Una variación de cantidad Δ_K de K indica una diferencia entre K_+ y K_-.

El número C_- de los segundos bloques de código que tienen el tamaño K_- se puede calcular redondeando por defecto un valor obtenido dividiendo un valor, que se obtiene restando el tamaño de secuencia de bits de entrada modificado B' del resultado de multiplicar el número C de bloques de código por el tamaño del primer bloque de código K_+, por la variación de cantidad Δ_K de K.

El número C_+ de los primeros bloques de código que tienen el tamaño K_+ se puede calcular restando el número C_- de los segundos bloques de código del número total C de bloques de código.

Dado que el tamaño de CRC L incluido en el bloque de código se considera para calcular el número C de bloques de código y el tamaño de bloque de código K, la suma de los bloques de código puede ser mayor que el tamaño de

secuencia de bits de entrada modificado B'. Entonces los bits de relleno que corresponden a la diferencia se pueden añadir al primer bloque de código.

En otra realización ejemplar de la presente invención, se puede calcular una longitud de bits de relleno F mediante una diferencia entre el tamaño de secuencia de bits de entrada modificado B' y un valor de $(C_+ \times K_+) + (C_- \times K_-)$.

La Tabla 8 ilustra una ecuación de configuración generada considerando el tamaño de CRC.

[Tabla 8]

Sintaxis	Valor	Notas
para k = 0 a F-1		
C_{0k} = <NULO>		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
Fin si		
mientras (k < K_r - L)		
$c_{rk} = b_s$		
k = k + 1		
s = s + 1		
Fin mientras		
si C > 1		
mientras k < Kr		
$c_{rk} = p_{r(Kr-k-1)}$		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$ según la sub cláusula 5.1.1
k = k + 1		
fin mientras		
Fin si		
k = 0		
fin para		

Con referencia a la Tabla 8, el módulo de función configura un bloque de código incluyendo el bloque de código de tamaño de CRC. Una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$.

En otra realización ejemplar de la presente invención, un código de detección de errores incluido previamente en el bloque de datos de entrada se puede conocer como una CRC de bloque de transporte (TB) (o primer código de detección de errores) y un código de detección de errores incluido en bloques de código a ser segmentado se puede conocer como una CRC de bloque de código (CB) (o segundo código de detección de errores). Un bloque de datos inicial antes de que es unida una CRC de TB a los bits de entrada se puede conocer como bits de entrada inicial o un bloque de transporte. El tamaño de una CRC de TB puede ser de 24 bits.

Si no se realiza segmentación del bloque de datos en la quinta realización (es decir, C=1), se puede configurar el bloque de datos de entrada inicialmente mediante un bloque de código final que incluye una CRC de TB. No obstante, el bloque de código final se puede configurar uniendo una CRC de CB en lugar de la CRC de TB según los requisitos del usuario y las realizaciones de la presente invención.

<Sexta Realización>

Otra realización ejemplar de la presente invención se puede construir mediante una combinación de los métodos descritos en la primera a cuarta realizaciones.

Una unidad de segmentación de bloque de datos y conexión de CRC según otra realización ejemplar de la presente invención es un módulo de función para segmentar un bloque de datos de entrada en bloques de código de tamaños adecuados en consideración a los códigos de detección de errores (por ejemplo, códigos de CRC) incluidos en los bloques de código a ser segmentados. Si se segmenta una secuencia de bits de entrada en bloques de código, se determina un tamaño que se puede segmentar de manera máxima según los requisitos del sistema. En las realizaciones ejemplares de la presente invención, un tamaño de bloque de código máximo Z puede ser de 6.144 bits.

Una secuencia de bits de entrada (o bloque de datos) introducida al módulo de función se puede indicar mediante b_0, b_1, b_2, …, b_{B-1}. Se asume que un tamaño de bits de entrada está por encima de 1. Si el tamaño de bits de entrada B es mayor que el tamaño de bloque de código máximo Z, se realiza segmentación de los bits de entrada. Un tamaño de CRC a ser unido a bloques de código se considera para segmentar los bits de entrada.

Si la longitud de los bits de relleno no es 0, los bits de relleno se pueden añadir al principio del primer bloque de código. Los bits de relleno se fijan a nulo en la entrada del módulo de función.

El número total C de bloques de código generados mediante segmentación de los bits de entrada se puede calcular mediante la Ecuación 30 siguiente.

[Ecuación 30]

Si $B \leq Z$

$C = 1$

$B' = B$

Además

$$C = \lceil B/(Z-L) \rceil$$

$B' = B + C \times L$

Fin

En la Ecuación 30, B indica el tamaño los bits de entrada (o bloque de datos) y B' indica un valor obtenido añadiendo el tamaño de bits de entrada B al resultado de multiplicar el número C de bloques de código por el tamaño de CRC L. Es decir, B' indica un tamaño de secuencia de bits de entrada modificado para adquirir el tamaño de bloque de código K.

Si el número de bloques de código no es 0, los bloques de código generados a partir del bloque de función se pueden indicar mediante c_{r0}, c_{r1}, c_{r2}, c_{r3},…, $c_{r(Kr-1)}$ (donde r indica un número de bloque de código y K_r indica el tamaño del bloque de código de orden r).

El módulo de función debería calcular el tamaño de cada bloque de código después de calcular el número de bloques de código a través de la Ecuación 30. Cada uno de los bloques de código puede tener un tamaño K_+ o K_-. En este caso, el número de bloques de código no debería ser 0.

En la realización ejemplar de la presente invención, el tamaño de un primer bloque de código se puede indicar por K_+. El tamaño K_+ se puede determinar a partir de los valores K en la Tabla 1. El tamaño de K_+ puede tener un valor mínimo entre los valores K que satisfacen la condición de que un valor obtenido dividiendo el tamaño de secuencia de bits de entrada modificado B' por el número C de bloques de código sea mayor o igual que los valores K.

Si el número C de bloques de código es 1, el número C_+ de primeros bloques de código que tienen el tamaño K_+ es 1 y el número C_- de segundos bloques de código que tienen el tamaño K_- es 0.

Si el número de bloques de código es 2 o más (C>1), el tamaño K_- de los segundos bloques de código se puede fijar a un valor máximo entre los valores K, mostrados en la Tabla 1, que es menor que K_+. Una variación de cantidad Δ_K de K indica una diferencia entre K_+ y K_-.

El número C_- de los segundos bloques de código que tienen el tamaño K_- se puede calcular redondeando por defecto un valor obtenido dividiendo un valor, que se obtiene restando el tamaño de secuencia de bits de entrada modificado B' del resultado de dividir el número C de bloques de código por el primer tamaño de bloque de código K_+, por la variación de cantidad Δ_K de K. El número C_+ de los primeros bloques de código que tienen el tamaño K_+ se puede calcular restando el número C_- de segundos bloques de código del número total C de bloques de código. El tamaño L de bits de paridad de CRC se puede fijar a 24 bits.

Dado que el tamaño de CRC L se considera para calcular el número y tamaño de bloques de código, el tamaño de bloque de código puede ser mayor que el tamaño de secuencia de bits de entrada modificado B'. En este caso, los bits de relleno que corresponden a la diferencia se pueden añadir al primer bloque de código.

En otra realización ejemplar de la presente invención, se puede calcular una longitud de bits de relleno F por una diferencia entre el tamaño de secuencia de bits de entrada modificado B' y un valor de $(C_+ \times K_+) + (C_- \times K_-)$.

La Tabla 9 ilustra una ecuación de configuración de un bloque de código generada considerando el tamaño de CRC.

[Tabla 9]

Sintaxis	Valor	Notas
para k = 0 a F-1		
c_{0k} = <NULO>		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
fin si		
mientras (k < K_r - L)		
$c_{rk} = b_s$		
k = k + 1		
s = s + 1		
fin mientras		
si C > 1		
mientras k < Kr		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2}, …, $p_{r(L-1)}$ según la tabla 1
$c_{rk} = p_{r(Kr-k-1)}$		

Sintaxis	Valor	Notas
k = k + 1		
fin mientras		
fin si		
k = 0		
fin para		

En la Tabla 9, el módulo de función configura un bloque de código incluyendo el tamaño de CRC. Una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$.

<Séptima Realización>

Otra realización ejemplar de la presente invención se puede construir mediante una combinación de los métodos descritos en la primera a cuarta realizaciones.

Otra realización ejemplar de la presente invención define un módulo de función para conexión de CRC y segmentación de bits de entrada. El módulo de función puede segmentar el bloque de datos de entrada en bloques de código de un número adecuado en consideración al tamaño de códigos de detección de errores (por ejemplo, códigos de CRC) incluidos en los bloques de código a ser segmentados. Si una secuencia de bits de entrada se segmenta en bloques de código, un tamaño Z que se puede segmentar de manera máxima se determina según los requisitos del sistema. En las realizaciones ejemplares de la presente invención, el tamaño de bloque de código máximo puede ser de 6.144 bits.

Los bloques de código generados mediante segmentación de bits de entrada introducidos al módulo de función se pueden indicar mediante b_0, b_1, b_2, …, b_{B-1}. La segmentación de los bits de entrada se puede realizar cuando el tamaño de bits de entrada B es mayor que el tamaño de bloque de código máximo.

Si el tamaño de bits de entrada B no es 0 y la longitud de bits de relleno no es 0, los bits de relleno se pueden añadir al principio del primer bloque de código. Si el tamaño de bits de entrada B es menor que 40 bits, los bits de relleno se añaden al principio del bloque de código. Los bits de relleno se fijan a nulo.

El número de bloques de código generados mediante segmentación de los bits de entrada se puede calcular mediante la Ecuación 31 siguiente.

[Ecuación 31]

Si $B \leq Z$

$B' = B$

Además

Si $(\lceil B/Z \rceil \times L + B > \lceil B/Z \rceil \times Z)$

$B' = B + (\lceil B/Z \rceil + 1) \times L$

Además

$B' = B + (\lceil B/Z \rceil \times L$

Fin

Fin

$C = \lceil B'/Z \rceil$

En la Ecuación 31, si el tamaño de bits de entrada B es menor o igual que el tamaño de bloque de código máximo Z, el tamaño de bits de entrada B es igual al tamaño de secuencia de bits de entrada modificado B'. Cuando el tamaño de bits de entrada B es mayor que el tamaño de bloque de código máximo Z, se usa el siguiente método.

Si un valor, obtenido añadiendo B al resultado de multiplicar L por un valor obtenido redondeando por exceso el resultado de dividir B por Z, es mayor que un valor, obtenido multiplicando Z por un valor obtenido redondeando por exceso el resultado de dividir B por Z, el tamaño de secuencia de bits de entrada modificado B' asume un valor obtenido redondeando por exceso el resultado de dividir B por Z, más 1 multiplicado por el tamaño de CRC L más el tamaño de bloque de datos B.

A través del proceso anterior, el número C de bloques de código se determina mediante un valor obtenido redondeando por exceso el resultado de dividir el tamaño de secuencia de bits de entrada modificado B' por el tamaño de bloque de código máximo Z. Los bits de entrada segmentados del bloque de función se pueden indicar mediante c_{r0}, c_{r1}, c_{r2}, c_{r3},..., $c_{r(Kr-1)}$ (donde r indica un número de bloque de código y K_r indica el tamaño del bloque de código de orden r).

El módulo de función debería calcular el tamaño de cada bloque de código después de calcular el número de bloques de código a través de la Ecuación 31. Cada uno de los bloques de código puede tener un tamaño K_+ o K_-. En este caso, el número de bloques de código no debería ser 0.

En la realización ejemplar de la presente invención, el tamaño de un primer bloque de código se puede indicar por K_+. El tamaño K_+ se puede determinar a partir de los valores K en la Tabla 1. El tamaño de K_+ puede tener un valor mínimo entre los valores K que satisfacen la condición de que un valor obtenido dividiendo el tamaño de secuencia de bits de entrada modificado B' por el número C de bloques de código sea mayor o igual que los valores K.

Si el número C de bloques de código es 1, el número C_+ de primeros bloques de código que tienen el tamaño K_+ es 1 y el número C_- de segundos bloques de código que tienen el tamaño K_- es 0.

Si el número de bloques de código es 2 o más (C>1), el tamaño K_- de los segundos bloques de código se puede determinar por un valor máximo entre los valores K, mostrados en la Tabla 1, que es menor que K_+. Una variación de cantidad Δ_K de K indica una diferencia entre K_+ y K_-.

El número C_- de segundos bloques de código que tienen el tamaño K_- se puede calcular redondeando por defecto un valor obtenido dividiendo un valor, que se obtiene restando el tamaño de secuencia de bits de entrada modificado B' del resultado de dividir el número C de bloques de código por el primer tamaño de bloque de código K_+, por la variación de cantidad Δ_K de K. El número C_+ de los primeros bloques de código que tienen el tamaño K_+ se puede calcular restando el número C_- de segundos bloques de código del número total C de bloques de código. El tamaño L de bits de paridad de CRC se puede fijar a 24 bits.

Dado que el tamaño de CRC L se considera para calcular el número y tamaño de bloques de código, el tamaño de bloque de código puede ser mayor que el tamaño de secuencia de bits de entrada modificado B'. Entonces los bits de relleno que corresponden a la diferencia se pueden añadir al primer bloque de código.

En otra realización ejemplar de la presente invención, se puede calcular una longitud de bits de relleno F por una diferencia entre el tamaño de secuencia de bits de entrada modificado B' y un valor de $(C_+ \times K_+) + (C_- \times K_-)$.

La Tabla 10 ilustra una ecuación de configuración de un bloque de código generada considerando el tamaño de CRC.

[Tabla 10]

Sintaxis	Valor	Notas
para k = 0 a F-1		
c_{0k} = <NULO>		
fin para		
k = F		
s = 0		
para r = 0 a C – 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		

Sintaxis	Valor	Notas
fin si		
si (C > 1)		
$K_r = K_r - L$		
Fin		
mientras k < K_r		
$c_{rk} = b_s$		
k = k + 1		
s = s + 1		
fin mientras		
k = 0		
fin para		

Se puede entender a través de la Tabla 10 que el módulo de función configura bloques de código mediante segmentación de bits de entrada en consideración a un tamaño de CRC.

<Octava Realización>

Otra realización ejemplar de la presente invención se puede construir mediante una combinación de los métodos descritos en la primera a cuarta realizaciones.

Los datos de un tamaño grande se pueden segmentar, antes de la transmisión, en datos de tamaños adecuados según los requisitos del sistema para transferir datos de manera efectiva. Por consiguiente, es necesario segmentar un bloque de datos de entrada mediante un método adecuado. Puede ser importante determinar qué método va a ser usado para segmentación de bloque de datos.

Una unidad de segmentación de bloque de datos y de conexión de CRC según otra realización ejemplar de la presente invención construye un módulo de función. El módulo de función segmenta el bloque de datos de entrada en bloques de código de un número adecuado en consideración al tamaño de códigos de detección de errores (por ejemplo, códigos de CRC) incluidos en los bloques de código a ser segmentados. Si una secuencia de bits de entrada se segmenta en bloques de código, un tamaño que se puede segmentar de manera máxima se determina según los requisitos del sistema. En las realizaciones ejemplares de la presente invención, el tamaño de bloque de código máximo puede ser de 6.144 bits.

Una secuencia de bits de entrada (o bloques de datos) introducida al módulo de función se puede indicar mediante b_0, b_1, b_2, …, b_{B-1}. El tamaño de los bits de entrada se puede indicar por 'B' (donde B está por encima de 1). Si el tamaño de bits de entrada B es mayor que el tamaño de bloque de código máximo Z, los bits de entrada se pueden segmentar en consideración de un tamaño de CRC. El tamaño de CRC puede ser de 24 bits. Esto es, un código de CRC de 24 bits, que es un tipo de un código de detección de errores, se puede unir a cada bloque de código generado mediante segmentación de la secuencia de bits de entrada.

Si la longitud de bits de relleno no es 0, los bits de relleno se pueden añadir al principio del primer bloque de código. Si el tamaño de bits de entrada B es menor que 40 bits, los bits de relleno se añaden al principio del bloque de código. Los bits de relleno se fijan a nulo.

El número C de bloques de código generados mediante segmentación de los bits de entrada se puede calcular mediante la Ecuación 32 siguiente.

[Ecuación 32]

$$C = \lceil B/(Z-L) \rceil$$

$$B' = B + C * L$$

En la Ecuación 32, 'B' indica el tamaño de bits de entrada (o bloque de datos) y un tamaño de secuencia de bits de entrada modificado B' se obtiene añadiendo el tamaño de bits de entrada B al resultado de multiplicar el número C de bloques de código por el tamaño de CRC L.

Esto es, el tamaño de secuencia de bits de entrada modificado B' es un valor temporal para calcular el número y tamaño de bloques de código.

Si el número de bloques de código no es 0, los bloques de código generados a partir del bloque de función se pueden indicar mediante c_{r0}, c_{r1}, c_{r2}, c_{r3},..., $c_{r(Kr-1)}$ (donde r indica un número de bloque de código y K_r indica el tamaño del bloque de código de orden r).

El módulo de función debería calcular el tamaño de cada bloque de código después de calcular el número de bloques de código a través de la Ecuación 32. Cada uno de los bloques de código puede tener un tamaño K_+ o K_-. En este caso, el número de bloques de código debería estar por encima de 1.

En la realización ejemplar de la presente invención, el tamaño de un primer bloque de código se puede indicar por K_+. El tamaño K_+ se puede determinar a partir de los valores K en la Tabla 1. El tamaño K_+ puede tener un valor mínimo entre los valores K que satisfacen la condición de que un valor obtenido dividiendo el tamaño de secuencia de bits de entrada modificado B' por el número C de bloques de código es mayor o igual que los valores K. Si el número C de bloques de código es 1, el número C_+ de primeros bloques de código que tienen el tamaño K_+ es 1 y el número C_- de segundos bloques de código que tienen el tamaño K_- es 0.

Si el número de bloques de código es 2 o más (C>1), el tamaño K_- de segundos bloques de código se puede determinar por un valor máximo entre los valores K, mostrados en la Tabla 1, que es menor que K_+. Una variación de cantidad Δ_K de K indica una diferencia entre K_+ y K_-.

El número C_- de los segundos bloques de código que tienen el tamaño K_- se puede calcular redondeando por defecto un valor obtenido dividiendo un valor, que se obtiene restando el tamaño de secuencia de bits de entrada modificado B' del resultado de dividir el número C de bloques de código por el primer tamaño de bloque de código K_+, por la variación de cantidad Δ_K de K. El número C_+ de los primeros bloques de código que tienen el tamaño K_+ se puede calcular restando el número C_- de segundos bloques de código del número total C de bloques de código.

Dado que el tamaño de CRC L se considera para calcular el número y tamaño de bloques de código, el tamaño de bloque de código puede ser mayor que el tamaño de secuencia de bits de entrada modificado B'. Entonces los bits de relleno que corresponden a la diferencia se pueden añadir al primer bloque de código.

En otra realización ejemplar de la presente invención, se puede calcular una longitud de bits de relleno F mediante una diferencia entre el tamaño de secuencia de bits de entrada modificado B' y un valor de $(C_+ \times K_+) + (C_- \times K_-)$.

La Tabla 11 ilustra una ecuación de configuración de un bloque de código generada considerando el tamaño de CRC.

[Tabla 11]

Sintaxis	Valor	Notas
para k = 0 a F-1		
c_{0k} = <NULO>		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
fin si		

Sintaxis	Valor	Notas
mientras (k < K_r - L)		
C_{rk} = b_s		
k = k + 1		
s = s + 1		
fin mientras		
si C > 1		
mientras k < Kr		
C_{rk} = $p_{r(Kr-k-1)}$		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2}, …, $p_{r(L-1)}$ según la tabla 1
k = k + 1		
fin mientras		
fin si		
k = 0		
fin para		

Se puede entender a través de la Tabla 11 que el módulo de función configura bloques de código en consideración al tamaño de CRC. En la Tabla 11 una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$.

<Novena Realización>

Otra realización ejemplar de la presente invención se puede construir mediante una combinación de los métodos descritos en la primera a cuarta realizaciones.

Un método según la novena realización es similar al método descrito en la octava realización. Es decir, el método de cálculo del número de bloques de código generados mediante segmentación de bits de entrada puede usar el método descrito en la octava realización. Además, los métodos para calcular bits de relleno, el tamaño de bloques de código, el número de bloques de código son similares a los métodos descritos en la octava realización. En lo sucesivo, solamente se describirán diferentes partes en el método de configuración de los bloques de código segmentados.

La Tabla 12 siguiente ilustra una ecuación de configuración de un bloque de código generada considerando un tamaño de CRC.

[Tabla 12]

Sintaxis	Valor	Notas
para k = 0 a F-1		
c_{0k} = <NULO>		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		

Sintaxis	Valor	Notas
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
fin si		
mientras k < K_r		
si (k < K_r – L)		
$C_{rk} = b_s$		
s = s + 1		
además		
$c_{rk} = p_{r(Kr-k-1)}$		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2}, …, $p_{r(L-1)}$ según la tabla 1
fin si		
k = k + 1		
fin mientras		
k = 0		
fin para		

Se puede entender a través de la Tabla 12 que el módulo de función configura bloques de código en consideración al tamaño de CRC. En la Tabla 12 una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$.

<Décima Realización>

Otra realización ejemplar de la presente invención se puede configurar mediante una combinación de los métodos descritos en la primera a cuarta realizaciones ejemplares.

La décima realización es similar a la quinta realización. Los métodos para calcular el número de bloques de código, el tamaño de bloques de código, y la longitud de bits de relleno son los mismos que los métodos descritos en la quinta realización. No obstante, un método para asignar datos a cada bloque de código y unir simultáneamente códigos de detección de errores es diferente del método descrito en la quinta realización.

La Tabla 13 siguiente ilustra una ecuación de configuración de un bloque de código generado considerando un tamaño de CRC.

[Tabla 13]

Sintaxis	Valor	Notas
para k = 0 a F-1		
c_{0k} = <NULO>		
fin para		
k = F		

Sintaxis	Valor	Notas
s = 0		
para r = 0 a C- 1		
si r < C_-		
$K_r = K_-$		
además		
$K_r = K_+$		
Fin si		
mientras (k < K_r – L)		
$c_{rk} = b_s$		
k = k + 1		
s = s + 1		
Fin mientras		
si C > 1		
mientras k < Kr		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2}, …, $p_{r(L-1)}$ según la sub cláusula 5.1.1 con el polinomio generador gCRCB (D). Para el cálculo de CRC se asume que los bits de relleno, si están presentes, tienen el valor 0.
$c_{rk} = p_{r(Kr+L-Kr)}$		
k = k + 1		
fin mientras		
Fin si		
k = 0		
fin para		

Una descripción fundamental en conjunto con la Tabla 13 es similar a la descripción con la Tabla 8. Por lo tanto, las partes repetitivas pueden referirse a la descripción en la Tabla 8.

En la Tabla 13, si el número C de bloques de código está por encima de 1, los bits de paridad de CRC se unen a cada bloque de código. Una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular un bit de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$. Los bits de paridad de CRC se pueden unir a lugares de los bits de paridad de CRC incluidos en bloques de código en una dirección hacia delante y hacia atrás. Se asume en la realización ejemplar de la presente invención que los bits de paridad de CRC se unen en la dirección hacia delante.

La Ecuación 33 ilustra la conexión de los bits de paridad de CRC en una dirección hacia delante.

[Ecuación 33]

Si C > 1

mientras k < Kr

$c_{rk} = p_{r(k+L-Kr)}$

k = k + 1

fin mientras

fin si

En la Ecuación 33, los bits de paridad de CRC ($p_{r(k+L-Kr)}$ introducidos a los bloques de código se unen en una dirección hacia delante. Esto es, un código de CRC se introduce después de que se introducen los datos a un bloque de código y un código de CRC se introduce después de que se introducen datos a otro bloque de código. A través de esos procesos, se asignan simultáneamente los datos y código de CRC a los bloques de código.

La FIG. 8 es un diagrama de flujo que ilustra un proceso de segmentación de un bloque de datos en consideración a un tamaño de CRC según una realización ejemplar de la presente invención.

Los datos de un tamaño grande se pueden segmentar, antes de la transmisión, en datos de tamaños adecuados según los requisitos del sistema para transferir datos de manera efectiva. Por consiguiente, es necesario segmentar un bloque de datos de entrada mediante un método adecuado. Puede ser importante determinar qué método va a ser usado para segmentación de bloque de datos.

Una unidad de segmentación de bloque de datos lógico y de conexión de CRC según la realización ejemplar de la presente invención se compone de un módulo de función para segmentar un bloque de datos de entrada en consideración al tamaño de los códigos de detección de errores (por ejemplo, códigos de CRC) incluidos en bloques de código a ser segmentados. No obstante, cuando se segmenta una secuencia de bits de entrada en bloques de código, se puede determinar el tamaño que se puede segmentar de manera máxima según los requisitos del sistema.

El bloque de datos puede incluir previamente un código de CRC dentro del mismo. Un bloque de datos inicial antes de que el código de CRC se una al bloque de datos se puede conocer como bits de entrada iniciales o bloque de transporte. El bloque de transporte puede implicar datos transmitidos desde una capa superior.

Con referencia a la FIG. 8, un bloque de datos que tiene un tamaño B se introduce a un módulo de función (paso S801). El bloque de datos puede incluir un primer código de CRC que indica un código de detección de errores para bits de entrada.

El módulo de función compara el tamaño B del bloque de datos (o bits de entrada) con un tamaño de bloque de código máximo Z (paso S802).

Si el tamaño de bloque de datos B es mayor que el tamaño de bloque de código máximo Z en el paso S802, el módulo de función calcula el número C de bloques de código segmentados mediante el bloque de datos de entrada en consideración a un tamaño de CRC a ser unido a los bloques de código (paso S803).

Después de calcular el número C de los bloques de código en el paso S803, el módulo de función calcula el tamaño K_+ o K_- de cada bloque de código (paso S804).

El tamaño de cada bloque de código se puede calcular mediante diversos métodos en el paso S804. Esto es, se pueden usar métodos de cálculo del tamaño de cada bloque de código descrito en la primera a novena realizaciones de la presente invención. Como ejemplo, se puede calcular el tamaño de cada bloque de código usando el tamaño de secuencia de bits de entrada modificado B'.

Después de calcular el número y tamaño de bloques de código considerando un tamaño de CRC, el módulo de función calcula una longitud de bits de relleno restando el tamaño de secuencia de bits de entrada modificado B' del tamaño de bloque de código (paso S805).

Después de calcular el número y tamaño de bloques de código a través de los pasos S803 a S805, el módulo de función asigna datos y un segundo código de CRC (paso S806). El segundo de código de CRC es diferente en función del primer código de CRC incluido en el bloque de datos. El primer código de CRC es un código de detección de errores para el bloque de datos y el segundo código de CRC es un código de detección de errores para bloques de código segmentados por el bloque de datos.

Si el tamaño de bloque de datos B no es mayor que el tamaño de bloque de código máximo Z en el paso S802, es innecesario segmentar el bloque de datos. Es decir, el número C de los bloques de código se fija a 1 y se omiten los pasos para calcular el tamaño de bloque de código y la longitud de bits de relleno. Por consiguiente, los datos se pueden asignar directamente al bloque de datos de entrada (paso S807). En este momento, se puede usar el primer código de CRC incluido en el bloque de datos y el segundo código de CRC no se une nuevamente.

La FIG. 9 es un diagrama de flujo que ilustra un proceso de cálculo del número C de bloques de código según una realización ejemplar de la presente invención.

Los pasos S901 y S902 en la FIG. 9 son similares a los pasos S801 y S802 en la FIG. 8 y por lo tanto se omitirá una descripción de los mismos.

Con referencia a la FIG. 9, si el tamaño de bloque de datos B es mayor que el tamaño de bloque de código máximo Z en el paso S902, el número C de bloques de código se calcula redondeando por exceso un valor que se obtiene dividiendo el tamaño de bloque de datos B por el resultado de restar el segundo tamaño de código de CRC L del tamaño de bloque de código máximo Z (paso S903).

Si el tamaño de bloque de datos B no es mayor que el tamaño de bloque de código máximo Z en el paso S902, es innecesario segmentar el bloque de datos de entrada y el número C de bloques de código se fija a 1 (paso S904).

La FIG. 10 es un diagrama de flujo que ilustra un proceso de segmentación de un bloque de datos que usa un tamaño de secuencia de bits de entrada modificado según una realización ejemplar de la presente invención.

Los datos de un tamaño grande se pueden segmentar, antes de la transmisión, en datos de tamaños adecuados según los requisitos del sistema para transferir datos de manera efectiva. Por consiguiente, es necesario segmentar un bloque de datos de entrada mediante un método adecuado. Puede ser importante determinar qué método va a ser usado para segmentación de bloque de datos.

Una unidad de segmentación de bloque de datos lógicos y de conexión de CRC según la realización ejemplar de la presente invención se compone de un módulo de función. El módulo de función segmenta un bloque de datos de entrada en bloques de código de un número adecuado y añade datos calculando el tamaño de cada bloque de código.

Con referencia a la FIG. 10, si un bloque de datos (es decir, bits de entrada) se introduce al módulo de función, el módulo de función calcula el número C de bloques de código segmentados mediante el bloque de datos (paso S1001). El bloque de datos puede incluir un primer código de CRC que indica un código de detección de errores para los bits de entrada.

Después de calcular del número C de bloques de código, el módulo de función calcula un tamaño de secuencia de bits de entrada modificado B' para adquirir el tamaño K de cada bloque de código (paso S1002).

En el paso S1002, el tamaño de secuencia de bits de entrada modificado B' se puede calcular añadiendo el tamaño de bloque de datos B al resultado de multiplicar el número C de bloques de código por un segundo tamaño de CRC L a ser unido al bloque de código. El tamaño de secuencia de bits de entrada modificado se usa para calcular el tamaño K de cada bloque de código.

El bloque de código se puede componer de primeros bloques de código y segundos bloques de código. El tamaño K_+ de los primeros bloques de código y el tamaño K_- de los segundos bloques de código no son valores fijos y son variables según los requisitos del sistema. El tamaño K indica el tamaño de cada bloque de código cuando los bloques de código tienen un tamaño fijo.

Después de que se calcula el tamaño de secuencia de bits de entrada modificado B' en el paso S1002, se puede calcular el tamaño K_+ de los primeros bloques de código usando un valor mínimo entre los valores k mostrados en la Tabla 1. No obstante, los valores K deberían estar por encima de un valor obtenido dividiendo el tamaño de secuencia de bits de entrada modificado B' por el número C de bloques de código (paso S1003).

Si se determina el tamaño K_+ de los primeros bloques de código, se calcula el tamaño K_- de los segundos bloques de código usando un número máximo de valores K mostrados en la Tabla 1. No obstante, los valores K para calcular el tamaño K_- de los segundos bloques de código debería ser un valor máximo entre valores menores que el tamaño K_+ de los primeros bloques de código (paso S1004).

El módulo de función puede calcular el número C_- de los segundos bloques de código usando el primer tamaño de bloque de código K_+ y el tamaño de secuencia de bits de entrada modificado B'. El módulo de función calcula una diferencia de valor Δ_K entre el primer tamaño de bloque de código K_+ y el segundo tamaño de bloque de código K_-. El módulo de función calcula el número C_- de los segundos bloques de código redondeando por defecto un valor obtenido dividiendo un valor del número C de bloques de código multiplicado por el primer tamaño de bloque de código menos el tamaño de secuencia de bits de entrada modificado por la diferencia de valor Δ_K (paso S1005).

El número C_+ de primeros bloques de código se calcula restando el número C_- de segundos bloques de código del número total C de bloques de código (paso S1006).

El tamaño adecuado de bloques de código segmentados por los bits de entrada se puede calcular usando el método descrito con referencia a la FIG. 10. En más detalle, se puede calcular el tamaño de cada bloque de código usando el tamaño de secuencia de bits de entrada modificado incluso cuando el tamaño de cada bloque de código es diferente según los requisitos del sistema así como cuando los bloques de código tienen un tamaño fijo.

La FIG. 11 es un diagrama que ilustra un proceso de conversión de un bloque de datos en bloques de código cuando el número C de bloques de código es 1 según una realización ejemplar de la presente invención.

Los datos de un tamaño grande se pueden segmentar, antes de la transmisión, en datos de tamaños adecuados según los requisitos del sistema. Por consiguiente, es necesario segmentar un bloque de datos de entrada mediante un método adecuado. Puede ser importante determinar qué método va a ser usado para segmentación de bloque de datos.

Una unidad de segmentación de bloque de datos lógicos y de conexión de CRC según la realización ejemplar de la presente invención se compone de un módulo de función para segmentar un bloque de datos de entrada considerando el tamaño de un código de detección de errores (por ejemplo, código de CRC) incluido en los bloques de código segmentados por una secuencia de bits de entrada.

No obstante, se puede determinar un tamaño de segmentación máximo según los requisitos del sistema cuando se segmenta una secuencia de bits de entrada en bloques de código. Se dará una descripción de un método de procesamiento de datos cuando un tamaño de bits de entrada B es menor o igual que el tamaño de bloque de código máximo Z.

La FIG. 11 muestra un proceso de conversión de bits de entrada 1100 introducidos a un módulo de función (una unidad de conexión de CRC y unidad de segmentación de bits de entrada 1120) en un bloque de código 1140. En más detalle, los bits de entrada 1100 se introducen al módulo de función 1120. Los bits de entrada 1100 pueden incluir un primer código de CRC que indica un código de detección de errores para los bits de entrada 1100.

El módulo de función 1120 compara el tamaño de los bits de entrada 1100 con el tamaño de bloque de código máximo Z. Si el tamaño de bits de entrada es menor o igual que el tamaño de bloque de código máximo Z, los bits de entrada no se segmentan en el módulo de función 1120 y se puede componer del bloque de código 1140.

Por lo tanto, los bits de entrada 1100 se convierten en el bloque de código 1140 y el bloque de código 1140 puede usar el primer código de CRC. Dado que el primer código de CRC se usa como un código de detección de errores para el bloque de código 1140, el módulo de función 1120 no une un segundo código de CRC al bloque de código 1140.

Con referencia a la FIG. 11, si el tamaño de bits de entrada es menor o igual que el tamaño de bloque de código máximo Z, los bits de entrada 1100 no se segmentan y se usan como el bloque de código 1140. Esto es, el número C de bloques de código es 1 y un tamaño de bloque de código K puede ser igual al tamaño de bits de entrada B. Además, el primer código de CRC incluido en los bits de entrada 1100 se puede usar sin unir el segundo código de CRC al bloque de código 1140. Por consiguiente, es posible procesar rápidamente los datos.

<Undécima Realización>

Otra realización ejemplar de la presente invención se puede configurar mediante una combinación de los métodos descritos en la primera a décima realizaciones.

La undécima realización se puede usar cuando el tamaño de bloque de código K es siempre constante y pueden darse bits de relleno. El proceso de cálculo del número C de bloques de código es el mismo que el proceso descrito en la cuarta o quinta realización. No obstante, en la undécima realización, el tamaño Kr de cada bloque de código es el mismo.

La Ecuación 34 siguiente ilustra un método de cálculo del tamaño de bloque de código Kr y una longitud de bits de relleno F (es decir, el número de bits de relleno).

[Ecuación 34]

$$Kr = B'/C$$

Número de bits de relleno: $F = C * Kr - B'$

En la Ecuación 34, el tamaño de bloque de código Kr se calcula dividiendo el tamaño de secuencia bits de entrada modificado B' por el número C de bloques de código. En este momento, 'r' indica un índice de un bloque de código. La longitud de bits de relleno F se calcula restando el tamaño de secuencia de bits de entrada modificado B' del resultado de multiplicar el número C de bloques de código por el tamaño de bloque de código Kr. Es decir, la undécima realización muestra un método de cálculo del tamaño de bloque de código Kr cuando el tamaño de cada uno de todos los bloques de código es el mismo.

La Tabla 14 ilustra una ecuación de configuración de un bloque de código generada en consideración a un tamaño de CRC y la longitud de bits de relleno F.

[Tabla 14]

Sintaxis	Valor	Notas
para k = 0 a F-1		
c_{0k} = <NULO>		
fin para		
k = F		
s = 0		
para r = 0 a C- 1		
mientras (k < K_r – L)		
c_{rk} = b_s		
k = k + 1		
s = s + 1		
fin mientras		
si C > 1		
mientras k < Kr		
c_{rk} = $p_{r(k+L-Kr)}$		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2}, …, $p_{r(L-1)}$ según la sub cláusula 5.1.1 con el polinomio generador gCRC24B(D). Para el cálculo de CRC se asume que los bits de relleno, si están presentes, tienen el valor 0.
k = k + 1		
fin mientras		
fin si		
k = 0		
fin para		

Se puede entender a través de la Tabla 14 que el módulo de función configura bloques de código en consideración a un tamaño de CRC. En la Tabla 14 una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$.

En otra realización ejemplar de la presente invención, un código de detección de errores incluido previamente en el bloque de datos se puede conocer como una CRC de bloque de transporte (TB) (o primer código de detección de errores) y un código de detección de errores incluido en bloques de código a ser segmentados se puede conocer como una CRC de bloque de código (CB) (o segundo código de detección de errores).

Cuando no se realiza una segmentación del bloque de datos de entrada (es decir, C=1), el bloque de datos de entrada se puede construir mediante un bloque de código final que incluye una CRC de TB. No obstante, el bloque de código final se puede construir uniendo una CRC de CB en lugar de una CRC de TB según los requisitos del usuario y las realizaciones de la presente invención.

<Duodécima Realización>

Otra realización ejemplar de la presente invención se puede configurar mediante una combinación de los métodos descritos en la primera a décima realizaciones ejemplares. La duodécima realización se puede usar cuando el

tamaño de bloque de código es constante y no pueden darse bits de relleno. Esto es, el tamaño de bits de entrada es igual al número total de bloques de código.

La duodécima realización se puede usar cuando el tamaño Kr de cada bloque de código es el mismo. El proceso de cálculo del número C de los bloques de código es el mismo que el proceso descrito en la cuarta o quinta realización.

La Ecuación 35 siguiente ilustra un método de cálculo del tamaño de bloque de código Kr.

[Ecuación 35]

$$Kr = B'/C$$

En la Ecuación 35, el tamaño de bloque de código Kr se calcula dividiendo el tamaño de secuencia bits de entrada modificado B' por el número C de bloques de código. En este momento, Kr indica el tamaño de un bloque de código y 'r' indica un índice de un bloque de código.

La Tabla 15 ilustra una ecuación de configuración generada en consideración a un tamaño de CRC L.

[Tabla 15]

Sintaxis	Valor	Notas
k = 0		
s = 0		
para r = 0 a C- 1		
mientras ($k < K_r - L$)		
$c_{rk} = b_s$		
k = k + 1		
s = s + 1		
fin mientras		
si C > 1		
mientras k < Kr		
$c_{rk} = p_{r(k+L-Kr)}$		La secuencia c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-L-1)}$ se usa para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2}, …, $p_{r(L-1)}$ según la sub cláusula 5.1.1 con el polinomio generador gCRC24B(D). Para el cálculo de CRC se asume que los bits de relleno, si están presentes, tienen el valor 0.
k = k + 1		
fin mientras		
fin si		
k = 0		
fin para		

Se puede entender a través de la Tabla 15 que el módulo de función configura bloques de código en consideración a un tamaño de CRC. En la Tabla 15, una secuencia de bloque de código c_{r0}, c_{r1}, c_{r2},…, $c_{r(Kr-1)}$ se puede usar para calcular los bits de paridad de CRC p_{r0}, p_{r1}, p_{r2},…, $p_{r(L-1)}$.

En otra realización ejemplar de la presente invención, un código de detección de errores incluido previamente en el bloque de datos se puede conocer como una CRC de bloque de transporte (TB) (o primer código de detección de

errores) y un código de detección de errores incluido en bloques de código a ser segmentados se puede conocer como una CRC de bloque de código (CB) (o segundo código de detección de errores).

Cuando no se segmenta el bloque de datos de entrada (es decir, C=1), el bloque de datos de entrada se puede construir mediante un bloque de código final que incluye una CRC de TB. No obstante, el bloque de código final se puede construir uniendo una CRC de CB en lugar de una CRC de TB según los requisitos del usuario y las realizaciones de la presente invención.

Será evidente para los expertos en la técnica que se pueden hacer diversas modificaciones y variaciones en la presente invención sin apartarse del alcance de la invención. De esta manera, se pretende que la presente invención cubra las modificaciones y variaciones de esta invención a condición de que queden dentro del alcance de las reivindicaciones adjuntas y sus equivalentes.

APLICABILIDAD INDUSTRIAL

Las realizaciones ejemplares de la presente invención se pueden aplicar a sistemas de acceso inalámbricos. Además, las realizaciones ejemplares de la presente invención se pueden aplicar a todos los métodos de generación de bloques de códigos segmentados por una secuencia de bits de entrada. Los bloques de código generados según las realizaciones ejemplares de la presente invención son aplicables a codificación, multiplexación, e intercalado de un canal físico. La presente invención no está limitada a lo que se ha mostrado y descrito de manera particular anteriormente en este documento, en su lugar de ser aplicada a todas las tecnologías dentro del alcance de la presente invención.

REIVINDICACIONES

1. Un método de transmisión de datos de enlace ascendente en un sistema de acceso inalámbrico, que comprende:

generar una secuencia de bits de entrada uniendo un primer código de detección de errores a los datos;

calcular un número C de los bloques de código usando un tamaño B de la secuencia de bits de entrada, un tamaño máximo Z de los bloques de código, y un tamaño L de un segundo código de detección de errores que va a ser unido a cada uno de los bloques de código;

calcular un tamaño B' de una secuencia de bits de entrada modificada usando el número C de los bloques de código, el tamaño L del segundo código de detección de errores, y el tamaño B de la secuencia de bits de entrada;

obtener un tamaño de cada uno de los bloques de código a partir de valores predeterminados, en base a un valor obtenido dividiendo el tamaño B' de la secuencia de bits de entrada modificada por el número C de los bloques de código;

segmentar la secuencia de bits de entrada mediante iteraciones cada una que produce una secuencia de bits de entrada segmentada, en donde el número de iteraciones de segmentación corresponde al número C de los bloques de código;

para cada iteración de segmentación, generar un bloque de código generando el segundo código de detección de errores en base a la secuencia de bits de entrada segmentada, y unir el segundo código de detección de errores generado a la secuencia de bits de entrada segmentada; y

codificar en canal los bloques de código,

en donde el número C de los bloques de código se calcula mediante un entero que redondea por exceso un valor obtenido dividiendo el tamaño B de la secuencia de bits de entrada por el resultado de restar el tamaño L del segundo código de detección de errores a ser unido a cada uno de los bloques de código del tamaño máximo Z de los bloques de código, tal como C = $\lceil B/(Z-L)) \rceil$

2. El método según la reivindicación 1,

en donde el primer código de detección de errores y el segundo código de detección de errores se generan por diferentes polinomios.

3. El método según cualquiera de las reivindicaciones 1 a 2,

en donde el tamaño B de la secuencia de bits de entrada se fija a un valor obtenido añadiendo el tamaño A de los datos al tamaño del primer código de detección de errores, y en donde el segundo código de errores a ser unido a cada uno de los bloques de código se une adicionalmente para detectar si cada uno de los bloques de código contiene errores.

4. El método según cualquiera de las reivindicaciones 1 a 3,

en donde el tamaño B' de la secuencia de bits de entrada modificada se calcula por $B' = B + C \cdot L$, donde B es el tamaño de la secuencia de bits de entrada, C un número de bloques de código, y L es el tamaño del segundo código de detección de errores.

5. El método según cualquiera de las reivindicaciones 1 a 4,

en donde el tamaño de cada uno de los bloques de código satisface la condición de que un valor obtenido multiplicando el número C de los bloques de código por el tamaño de cada uno de los bloques de código es mayor o igual que el tamaño B' de la secuencia de bits de entrada modificado.

6. El método según la reivindicación 5,

en donde el tamaño K_+ de los primeros bloques de código de los bloques de código es el tamaño más pequeño entre los valores predeterminados, y

el tamaño K_- de los segundos bloques de código de los bloques de código es el tamaño más grande entre los valores predeterminados que son menores que el tamaño K_+.

7. El método según la reivindicación 6,

en donde un número C_- de los segundos bloques de código se calcula mediante un entero que redondea por defecto un valor obtenido dividiendo, por una diferencia entre el tamaño K_+ de los primeros bloques de código y

el tamaño K_- de los segundos bloques de código, el resultado de restar el tamaño B' de la secuencia de bits de entrada modificada a partir de un valor obtenido multiplicando el número C de los bloques de código por el tamaño K_+ de los primeros bloques de código, tal como $C_- = \left\lfloor \frac{C \cdot K_+ - B'}{\Delta_K} \right\rfloor$, donde Δ_K es la diferencia de valor entre el tamaño K_+ de los primeros bloques de código, y el tamaño K_- de los segundos bloques de código.

8. El método según la reivindicación 7,

en donde un número C_+ de los primeros bloques de código se fija a un valor obtenido restando el número C_- de los segundos bloques de código del número C de los bloques de código.

9. El método según la reivindicación 8, que además comprende:

calcular el número de bits de relleno F usando $F = C_+ \cdot K_+ + C_- \cdot K_- - B'$, donde C_+ es el número de los primeros bloques de código, K_+ es el tamaño de los primeros bloques de código, el C_- es el número de los segundos bloques de código, K_- es el tamaño de los segundos bloques de código, y B' es el tamaño de la secuencia de bits de entrada modificado; y

asignar los bits de relleno a un bloque de código inicial entre los bloques de código.

10. El método según la reivindicación 9, que además comprende:

asignar la secuencia de bits de entrada a un bloque de código inicial de los bloques de código dejando áreas para los bits de relleno y el segundo código de detección de errores el cual se unirá al bloque de código inicial; y

asignar la secuencia de bits de entrada restante a bloques de código posteriores después de que el bloque de código inicial que deja áreas para el segundo código de detección de errores que se unirá a los bloques de código posteriores.

11. El método según la reivindicación 10,

en donde la asignación de la secuencia de bits de entrada al primer bloque de código adicional que comprende unir el segundo código de detección de errores y la asignación de los bits de relleno al bloque de código inicial; y

en donde la asignación de la secuencia de bits de entrada restante a los bloques de código posteriores que además comprende unir el segundo código de detección de errores a los bloques de código posteriores.

12. El método según cualquiera de las reivindicaciones 1 a 11,

en donde el tamaño máximo Z de los bloques de código es 6.144 bits.

13. Un aparato para transmitir datos de enlace ascendente en un sistema de acceso inalámbrico, el aparato que comprende:

una unidad de transmisión configurada para transmitir los datos de enlace ascendente;

una unidad de segmentación de bloque código y de conexión de Comprobación de Redundancia Cíclica, CRC, para soportar transmisión de los datos de enlace ascendente; y

una unidad de codificación de canal,

en donde la unidad de segmentación de bloque de código y conexión de CRC se configura para:

recibir una secuencia de bits de entrada que se genera uniendo un primer código de detección de errores a los datos;

la unidad segmentación de bloque de código y de conexión de CRC se configura además para:

calcular un número C de los bloques de código usando un tamaño B de la secuencia de bits de entrada, un tamaño máximo Z de los bloques de código, y un tamaño L de un segundo código de detección de errores que va a ser unido a cada uno de los bloques de código;

calcular un tamaño B' de una secuencia de bits de entrada modificada usando el número C de los bloques de código, el tamaño L del segundo código de detección de errores, y el tamaño B de la secuencia de bits de entrada;

obtener un tamaño de cada uno de los bloques de código a partir de valores predeterminados, en base a un valor obtenido dividiendo el tamaño B' de la secuencia de bits de entrada modificada por el número C de los bloques de código;

segmentar la secuencia de bits de entrada mediante iteraciones cada una que produce una secuencia de bits de entrada segmentada, en donde el número de iteraciones de segmentación corresponde al número C de los bloques de código; y

para cada iteración de segmentación, generar un bloque de código generando el segundo código de detección de errores en base a la secuencia de bits de entrada segmentada, y unir el segundo código de detección de errores generado a la secuencia de bits de entrada segmentada;

la unidad de codificación de canal se configura para realizar una codificación de canal de los bloques de código, y

el número C de los bloques de código se calcula mediante un entero que redondea por exceso un valor obtenido dividiendo el tamaño B de la secuencia de bits de entrada por el resultado de restar el tamaño L del segundo código de detección de errores a ser unido a cada uno de los bloques de código del tamaño máximo Z de los bloques de código, tal como C = $\lceil B/(Z-L)) \rceil$

14. El aparato según la reivindicación 13,

en donde el tamaño B' de la secuencia de bits de entrada modificada se calcula por $B' = B + C \cdot L$, donde B es el tamaño de la secuencia de bits de entrada, C es el número de bloques de código, y L es el tamaño del segundo código de detección de errores.

FIG. 1

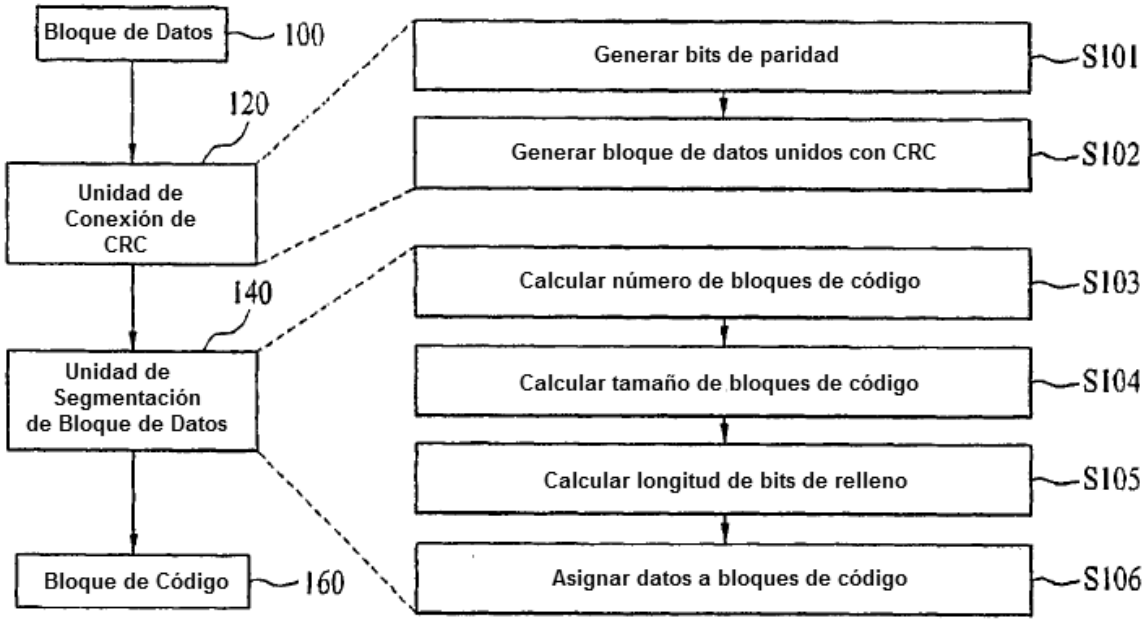

Bloque de Datos
100
120
Unidad de Conexión de CRC
140
Unidad de Segmentación de Bloque de Datos
Bloque de Código
160
Generar bits de paridad
S101
Generar bloque de datos unidos con CRC
S102
Calcular número de bloques de código
S103
Calcular tamaño de bloques de código
S104
Calcular longitud de bits de relleno
S105
Asignar datos a bloques de código
S106

FIG. 2

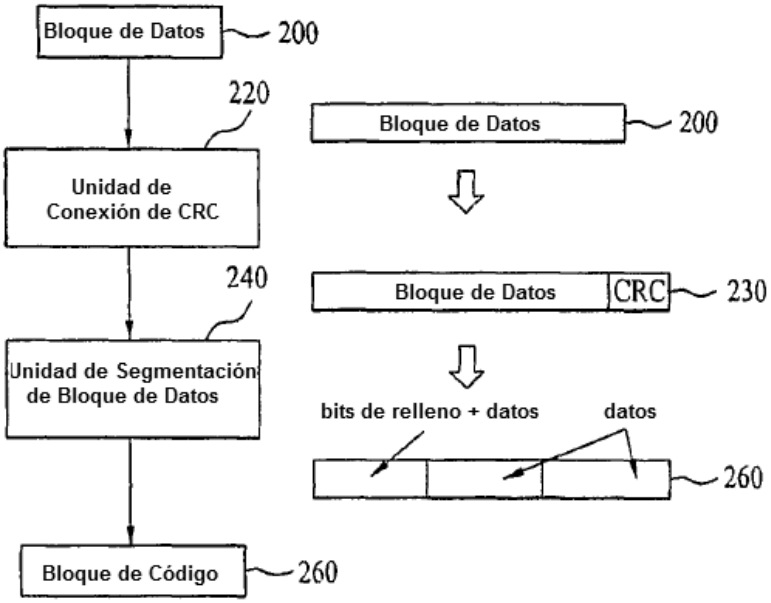

Bloque de Datos
200
220
Unidad de Conexión de CRC
240
Unidad de Segmentación de Bloque de Datos
Bloque de Código
260
Bloque de Datos
200
Bloque de Datos
CRC
230
bits de relleno + datos
datos
260

FIG. 3

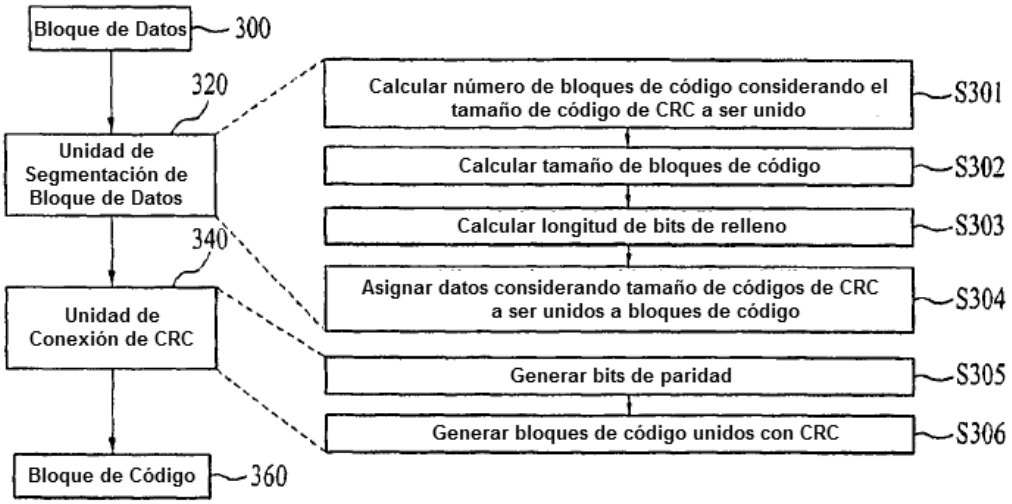

Bloque de Datos
300
320
Unidad de Segmentación de Bloque de Datos
340
Unidad de Conexión de CRC
Bloque de Código
360
Calcular número de bloques de código considerando el tamaño de código de CRC a ser unido
S301
Calcular tamaño de bloques de código
S302
Calcular longitud de bits de relleno
S303
Asignar datos considerando tamaño de códigos de CRC a ser unidos a bloques de código
S304
Generar bits de paridad
S305
Generar bloques de código unidos con CRC
S306

FIG. 4

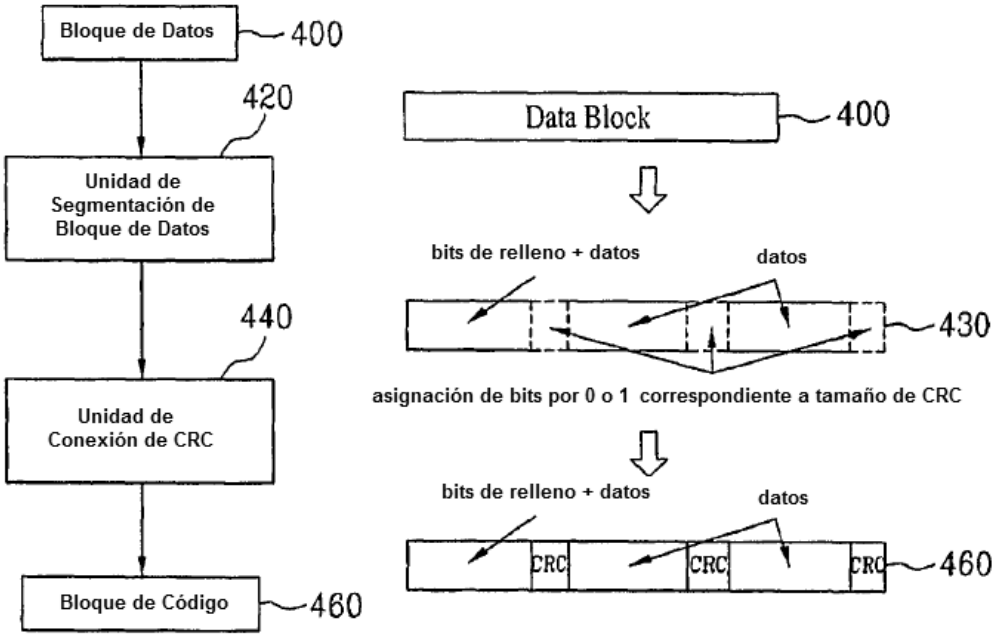

Bloque de Datos
400
420
Unidad de Segmentación de Bloque de Datos
440
Unidad de Conexión de CRC
Bloque de Código
460
Data Block
400
bits de relleno + datos
datos
430
asignación de bits por 0 o 1 correspondiente a tamaño de CRC
bits de relleno + datos
datos
CRC
CRC
CRC
460

FIG. 5

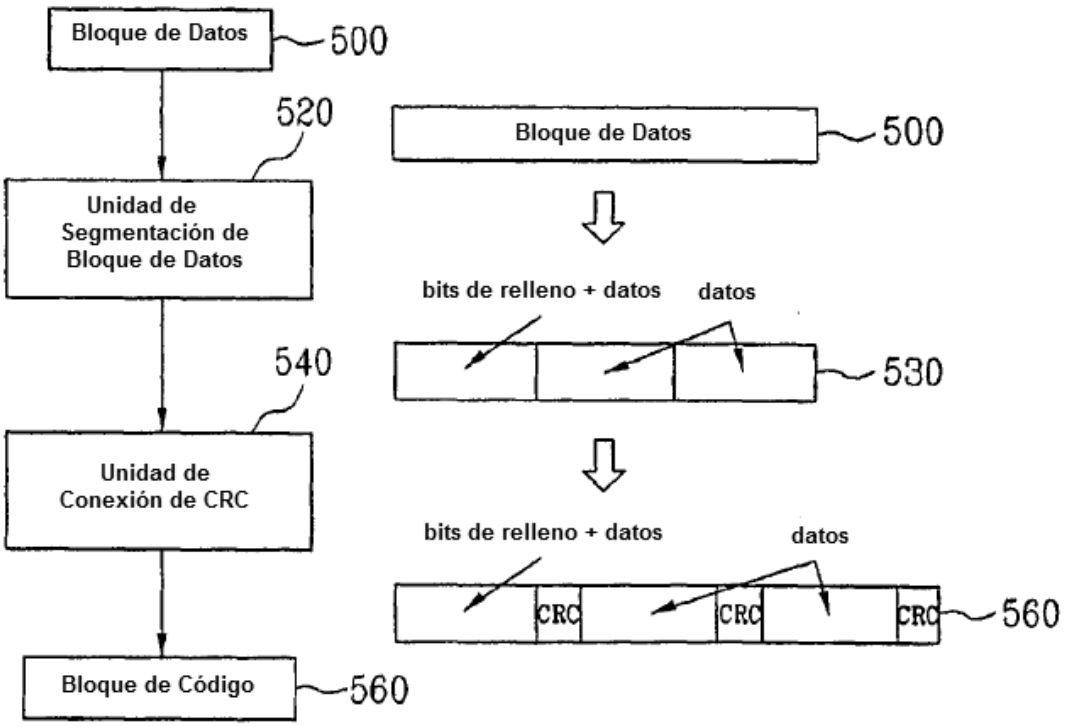

Bloque de Datos
500
520
Unidad de Segmentación de Bloque de Datos
540
Unidad de Conexión de CRC
Bloque de Código
560
Bloque de Datos
500
bits de relleno + datos
datos
530
bits de relleno + datos
datos
CRC
CRC
CRC
560

FIG. 6

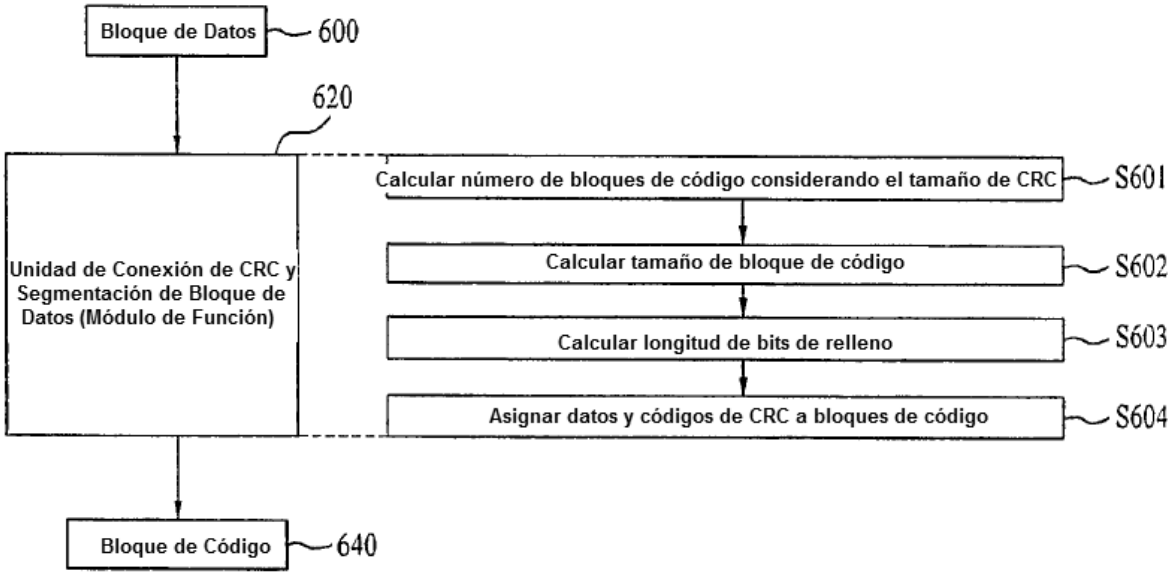

Bloque de Datos
600
620
Unidad de Conexión de CRC y Segmentación de Bloque de Datos (Módulo de Función)
Calcular número de bloques de código considerando el tamaño de CRC
S601
Calcular tamaño de bloque de código
S602
Calcular longitud de bits de relleno
S603
Asignar datos y códigos de CRC a bloques de código
S604
Bloque de Código
640

FIG. 7

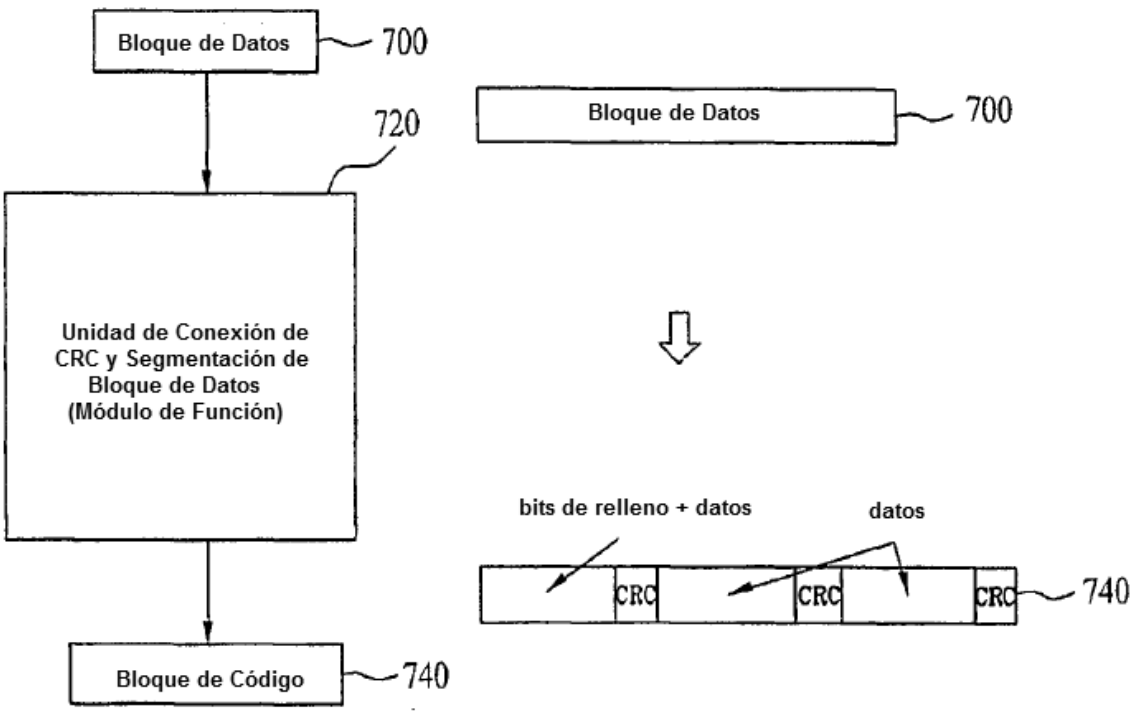

Bloque de Datos
700
720
Unidad de Conexión de CRC y Segmentación de Bloque de Datos (Módulo de Función)
Bloque de Código
740
Bloque de Datos
700
bits de relleno + datos
datos
CRC
CRC
CRC
740

FIG. 8

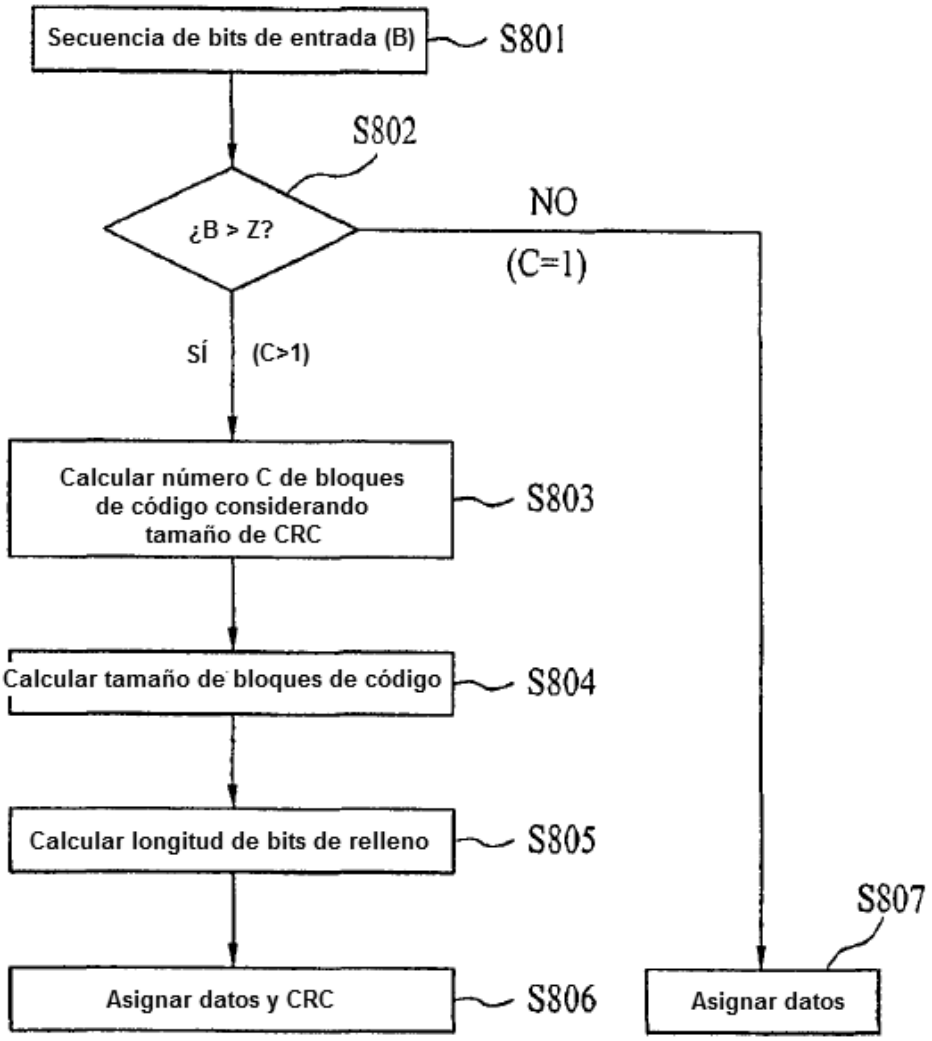
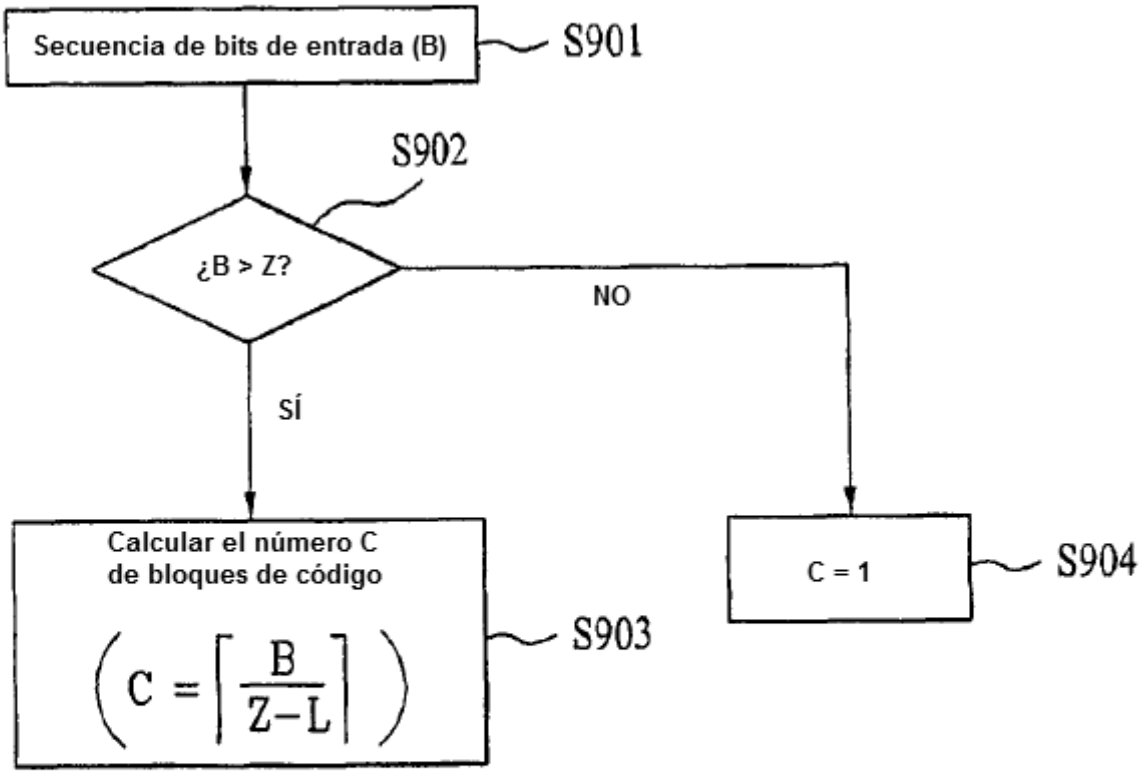

Secuencia de bits de entrada (B)
S801
S802
¿B > Z?
NO
(C=1)
SÍ
(C>1)
Calcular número C de bloques de código considerando tamaño de CRC
S803
Calcular tamaño de bloques de código
S804
Calcular longitud de bits de relleno
S805
S807
Asignar datos y CRC
S806
Asignar datos
B: Tamaño de secuencia de bits de entrada
Z: tamaño de bloque de código máximo
C: número de bloques de código

FIG. 9

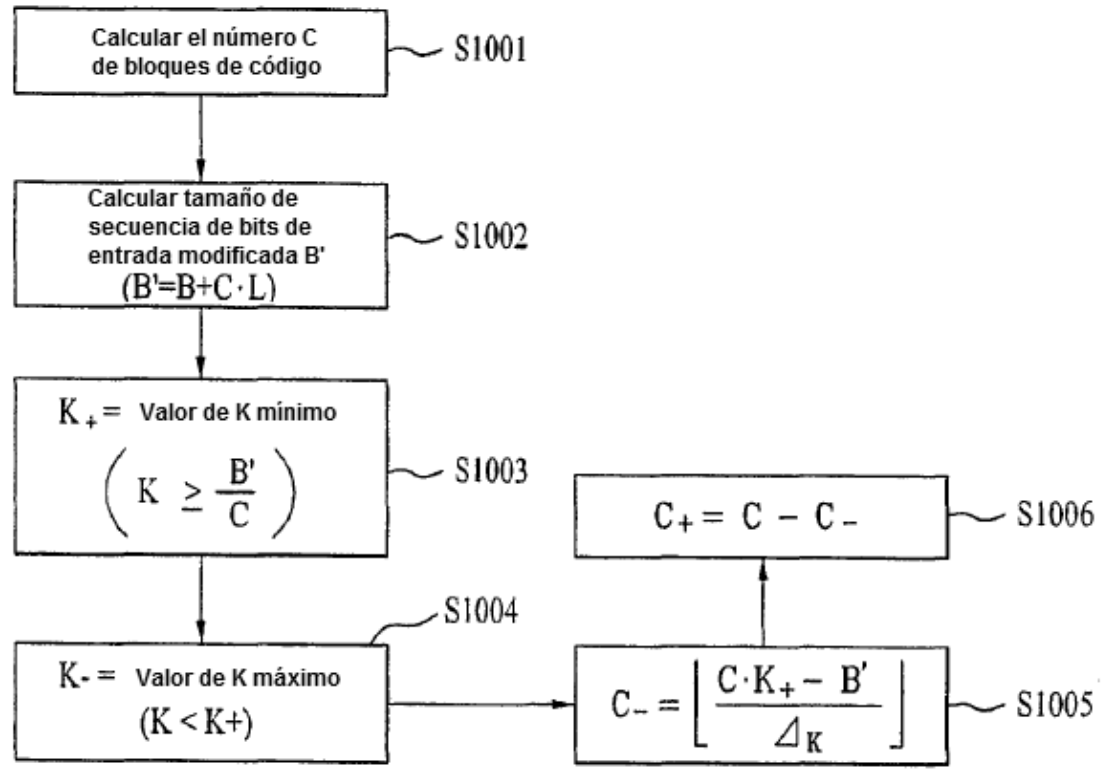


B: Tamaño de secuencia de bits de entrada

Z: tamaño de bloque de código máximo

C: número de bloques de código

FIG. 10



B: tamaño de secuencia de bits de entrada

L: segundo tamaño de CRC

C: número de bloques de código

Z: tamaño de bloques de código máximo

Δ_K: $K_+ - K_-$

K_+: primer tamaño de bloque de código

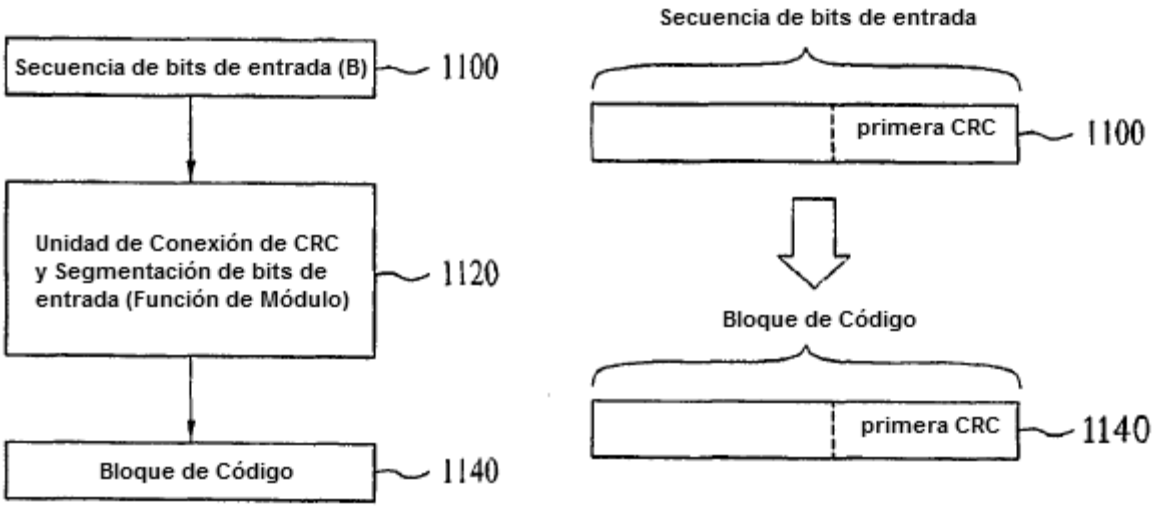
K_-: segundo tamaño de bloque de código

C_+: número de primeros bloques de código

C_-: número de segundos bloques de código

B': tamaño de secuencia de bits de entrada modificado

FIG. 11


Secuencia de bits de entrada (B)
1100
Unidad de Conexión de CRC y Segmentación de bits de entrada (Función de Módulo)
1120
Bloque de Código
1140
Secuencia de bits de entrada
primera CRC
1100
Bloque de Código
primera CRC
1140