República Federativa do Brasil

Ministério do Desenvolvimento, Indústria, Comércio e Serviços

Instituto Nacional da Propriedade Industrial

(11) BR 122022013496-3 B1

(22) Data do Depósito: 19/10/2010

(45) Data de Concessão: 16/05/2023

(54) Título: CODIFICADOR DE ÁUDIO, DECODIFICADOR DE ÁUDIO, MÉTODO PARA CODIFICAR UMA INFORMAÇÃO DE ÁUDIO, MÉTODO PARA DECODIFICAR UMA INFORMAÇÃO DE ÁUDIO QUE UTILIZA UMA DETECÇÃO DE UM GRUPO DE VALORES ESPECTRAIS PREVIAMENTE DECODIFICADOS

(51) Int.Cl.: G10L 19/00.

(30) Prioridade Unionista: 20/10/2009 US 61/253,459.

(73) Titular(es): FRAUNHOFER-GESELLSCHAFT ZUR FÖRDERUNG DER ANGEWANDTEN FORSCHUNG E.V..

(72) Inventor(es): GUILLAUME FUCHS; VIGNESH SUBBARAMAN; NIKOLAUS RETTELBACH; MARKUS MULTRUS; MARC GAYER; PATRICK WARMBOLD; CHRISTIAN GRIEBEL; OLIVER WEISS.

(86) Pedido PCT: PCT EP2010065725 de 19/10/2010

(87) Publicação PCT: WO 2011/048098 de 28/04/2011

(85) Data do Início da Fase Nacional: 06/07/2022

(62) Pedido Original do Dividido: BR112012009445-9 - 19/10/2010

(57) Resumo: Um decodificador de áudio (200) para prover uma informação de áudio decodificado (212) com base em uma informação de áudio codificada (210) compreende um decodificador aritmético (230) para prover uma pluralidade de valores espectrais decodificados (232) com base em uma representação aritmeticamente codificada (222) dos valores espectrais e um conversor de domínio de frequência para domínio de tempo (260) para prover uma representação de áudio de domínio de tempo (262) utilizando os valores espectrais decodificados, a fim de obter as informações de áudio decodificado. O decodificador aritmético (230) é configurado para selecionar uma regra de mapeamento que descreve um mapeamento de um valor de código em um código de símbolo em dependência de um estado de contexto. O decodificador aritmético é configurado para determinar ou modificar o estado de contexto atual em dependência de uma pluralidade de valores espectrais previamente decodificados.

CODIFICADOR DE ÁUDIO, DECODIFICADOR DE ÁUDIO, MÉTODO PARA CODIFICAR UMA INFORMAÇÃO DE ÁUDIO, MÉTODO PARA DECODIFICAR UMA INFORMAÇÃO DE ÁUDIO QUE UTILIZA UMA DETECÇÃO DE UM GRUPO DE VALORES ESPECTRAIS PREVIAMENTE DECODIFICADOS

Pedido dividido do BR 11 2012 009445-9 depositado em 19/10/2010.

CAMPO TÉCNICO

As realizações, de acordo com a invenção, são relacionadas a um decodificador de áudio para prover uma informação de áudio decodificado com base em uma informação de áudio codificada, um codificador de áudio para prover uma informação de áudio codificada com base em uma informação de áudio de entrada, um método para prover uma informação de áudio decodificado com base em uma informação de áudio codificada, um método para prover uma informação de áudio codificada com base em uma informação de áudio de entrada e um programa de computador.

As realizações, de acordo com a invenção, são relacionadas a uma codificação silenciosa espectral melhorada, que pode ser utilizada em um codificador ou decodificador de áudio, como, por exemplo, um denominado codificador de fala e áudio unificado (USAC).

HISTÓRICO DA INVENÇÃO

A seguir, o histórico da invenção será brevemente explicado a fim de facilitar o entendimento da invenção e as suas vantagens. Durante a última década, foram envidados grandes esforços para criar a possibilidade de armazenar digitalmente e distribuir conteúdos de áudio com boa eficiência de taxa de bits. Uma conquista importante sobre essa forma é a definição do Padrão

Internacional ISO/IEC 14496-3. A parte 3 desse padrão se refere a uma codificação e decodificação de conteúdos de áudio, e a subparte 4 da parte 3 é relacionada à codificação de áudio em geral. ISO/IEC 14496 parte 3, subparte 4 define um conceito para codificar e decodificar conteúdo de áudio em geral. Além disso, melhorias adicionais foram propostas a fim de melhorar a qualidade e/ou reduzir a taxa de bits necessária.

De acordo com o conceito descrito no dito Padrão, um sinal de áudio de domínio de tempo é convertido em uma representação de frequência de tempo. A transformação do domínio de tempo para o domínio de frequência de tempo é tipicamente realizada utilizando blocos de transformação, que também são designados como “estruturas”, das amostras de domínio de tempo. Descobriu-se que é vantajoso utilizar estruturas de sobreposição, que são alteradas, por exemplo, pela metade de uma estrutura, porque a sobreposição permite evitar de modo eficiente (ou pelo menos reduzir) artefatos. Além disso, descobriu-se que um janelamento deve ser realizado a fim de evitar os artefatos que se originam desse processamento de estruturas temporariamente limitadas.

Ao transformar uma parte janelada do sinal de áudio de entrada do domínio de tempo para o domínio de frequência de tempo, uma compactação de energia é obtida em muitos casos, de modo que alguns dos valores espectrais compreendam uma magnitude significativamente maior que uma pluralidade de outros valores espectrais. Da mesma forma, há, em muitos casos, um número comparativamente pequeno de valores espectrais tendo uma magnitude, que está significativamente acima de uma magnitude

média dos valores espectrais. Um exemplo típico de uma transformação de domínio de tempo para domínio de frequência de tempo que resulta em uma compactação de energia é a denominada transformada do cosseno discreta modificada (MDCT).

Os valores espectrais são geralmente escalonados e quantificados de acordo com um modelo psicoacústico, de modo que erros de quantificação sejam comparativamente menores para valores espectrais psicoacusticamente mais importantes e sejam comparativamente maiores para valores espectrais psicoacusticamente menos importantes. Os valores espectrais escalonados e quantificados são codificados a fim de prover sua representação de taxa de bits eficiente.

Por exemplo, o uso de uma denominada codificação de Huffman de coeficientes espectrais quantificados é descrito no Padrão Internacional ISO/IEC 14496-3:2005(E), parte 3, subparte 4.

Entretanto, descobriu-se que a qualidade da codificação dos valores espectrais tem um impacto significativo na taxa de bits necessária. Também, descobriu-se que a complexidade de um decodificador de áudio, que é geralmente implementado em um dispositivo de consumidor portátil e que deve, portanto, ser barato e de baixo consumo de energia, é dependente da codificação utilizada para codificar os valores espectrais.

Tendo em vista essa situação, há uma necessidade de um conceito para uma codificação e decodificação de um conteúdo de áudio, que provê uma troca melhorada entre a eficiência de taxa de bits e eficiência do recurso.

SUMÁRIO DA INVENÇÃO

Uma realização, de acordo com a invenção, cria um

decodificador de áudio para prover uma informação de áudio decodificado (ou representação de áudio decodificado) com base em uma informação de áudio codificada (ou representação de áudio codificado). O decodificador de áudio compreende um decodificador aritmético para prover uma pluralidade de valores espectrais decodificados com base em uma representação aritmeticamente codificada dos valores espectrais. O decodificador de áudio também compreende um conversor de domínio de frequência para domínio de tempo para prover uma representação de áudio de domínio de tempo utilizando os valores espectrais decodificados, a fim de obter as informações de áudio decodificado. O decodificador aritmético é configurado para selecionar uma regra de mapeamento que descreve um mapeamento de um valor de código em um código de símbolo em dependência de um estado de contexto. O decodificador aritmético é configurado para determinar o estado de contexto atual em dependência de uma pluralidade de valores espectrais previamente decodificados. O decodificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais previamente decodificados, que atenda, individualmente ou considerado juntamente, uma condição predeterminada em relação a suas magnitudes e para determinar ou modificar o estado de contexto atual em dependência de um resultado da detecção.

Essa realização, de acordo com a invenção, tem base na descoberta de que a presença de um grupo de uma pluralidade de valores espectrais previamente decodificados (preferencial, mas não necessariamente, adjacentes), que atendam à condição predeterminada em relação a suas magnitudes, permite uma determinação particularmente eficiente do estado de contexto

atual, uma vez que esse grupo de valores espectrais previamente decodificados (preferencialmente, adjacentes) é um aspecto característico dentro da representação espectral e pode, portanto, ser utilizado para facilitar a determinação do estado de contexto atual. Ao detectar um grupo de uma pluralidade de valores espectrais previamente decodificados (preferencialmente, adjacentes) que compreendem, por exemplo, uma magnitude particularmente pequena, é possível reconhecer partes de amplitude comparativamente baixa dentro do espectro e ajustar (determinar ou modificar) o estado de contexto atual da mesma forma, de modo que valores espectrais adicionais possam ser codificados e decodificados com boa eficiência de codificação (em termos de taxa de bits). De modo alternativo, grupos de uma pluralidade de valores espectrais adjacentes previamente decodificados que compreendem uma amplitude comparativamente grande podem ser detectados e o contexto pode ser adequadamente ajustado (determinado ou modificado) para aumentar a eficiência da codificação e decodificação. Além disso, a detecção de grupos de uma pluralidade de valores espectrais previamente decodificados (preferencialmente, adjacentes) que atendam, individualmente ou considerados juntamente, à condição predeterminada, é geralmente executável com esforço computacional menor que uma computação de contexto na qual muitos valores espectrais previamente decodificados são combinados. Para resumir, a realização discutida acima, de acordo com a invenção, permite uma computação de contexto simplificada e permite um ajuste do contexto para especificar constelações de sinal nas quais há grupos de valores espectrais comparativamente pequenos adjacentes ou grupos de

valores espectrais comparativamente grandes adjacentes.

Em uma realização preferida, o decodificador aritmético é configurado para determinar ou modificar o estado de contexto atual independente dos valores espectrais previamente decodificados em resposta à detecção de que a condição predeterminada é atendida. Da mesma forma, um mecanismo eficiente particularmente de modo computacional é obtido para a derivação de um valor que descreve o contexto. Descobriu-se que uma adaptação significativa do contexto pode ser alcançada se uma detecção de um grupo de uma pluralidade de valores espectrais previamente decodificados, que atende à condição predeterminada, resultar em um mecanismo simples que não precise de uma combinação numérica de demanda computacional de valores espectrais previamente decodificados. Assim, o esforço computacional é reduzido quando comparado a outras abordagens. Também, uma aceleração da derivação de contexto pode ser alcançada ao omitir etapas de cálculo complexas que são dependentes da detecção, porque esse conceito é tipicamente ineficiente em uma implementação em software executada em um processador.

Em uma realização preferida, o decodificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados, que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes.

Em uma realização preferida, o decodificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados que, individualmente ou considerado juntamente, compreende uma

magnitude que é menor que uma magnitude limite predeterminada, e para determinar o estado de contexto atual em dependência do resultado da detecção. Descobriu-se que um grupo de uma pluralidade de valores espectrais comparativamente baixos adjacentes pode ser utilizado para selecionar um contexto que é bem adaptado para essa situação. Se houver um grupo de valores espectrais comparativamente pequenos adjacentes, há uma probabilidade significativa de que o valor espectral a ser codificado depois também compreenda um valor comparativamente pequeno. Da mesma forma, um ajuste do contexto provê uma boa eficiência de codificação e pode auxiliar na evasão de computações de contexto que levam tempo.

Em uma realização preferida, o decodificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados, em que cada um dos valores espectrais previamente decodificados é um valor zero, e para determinar o estado de contexto em dependência do resultado da detecção. Descobriu-se que, devido a efeitos de encobrimento espectral ou temporal, há geralmente grupos de valores espectrais adjacentes que consideram um valor zero. A realização descrita provê uma administração eficiente para essa situação. Além disso, a presença de um grupo de valores espectrais adjacentes, que são quantificados a zero, torna muito provável que o valor espectral a ser decodificado depois seja um valor zero ou um valor espectral comparativamente grande, o que resulta no efeito de encobrimento.

Em uma realização preferida, o decodificador aritmético é configurado para detectar um grupo de uma pluralidade

de valores espectrais adjacentes previamente decodificados, que compreende um valor de soma que é menor que um valor limite predeterminado, e para determinar o estado de contexto em dependência de um resultado da detecção. Descobriu-se que, além dos grupos de valores espectrais adjacentes que são zero, também, grupos de valores espectrais adjacentes que são quase zero em uma média (isto é, um valor de soma que é menor que um valor limite predeterminado), constituem um aspecto característico de uma representação espectral (por exemplo, uma representação de frequência de tempo do conteúdo de áudio) que pode ser utilizado para a adaptação do contexto.

Em uma realização preferida, o decodificador aritmético é configurado para ajustar o estado de contexto atual a um valor predeterminado em resposta à detecção da condição predeterminada. Descobriu-se que essa reação é muito simples de implementar e ainda resulta em uma adaptação do contexto que provê uma boa eficiência de codificação.

Em uma realização preferida, o decodificador aritmético é configurado para omitir seletivamente um cálculo do estado de contexto atual em dependência dos valores numéricos de uma pluralidade de valores espectrais previamente decodificados em resposta da detecção da condição predeterminada. Da mesma forma, a computação de contexto é significativamente simplificada em resposta da detecção de um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados que atenda à condição predeterminada. Ao evitar o esforço computacional, um consumo de energia do decodificador de sinal de áudio também é reduzido, o que provê vantagens significativas em dispositivos

móveis.

Em uma realização preferida, o decodificador aritmético é configurado para ajustar o estado de contexto atual a um valor que sinalize a detecção da condição predeterminada. Ao ajustar o estado de contexto a esse valor, que pode estar dentro de uma variação predeterminada de valores, a avaliação posterior do estado de contexto pode ser controlada. Entretanto, deve ser observado que o valor ao qual o estado de contexto atual está ajustado pode ser dependente de outros critérios também, apesar de o valor poder estar em uma variação característica de valores que sinaliza a detecção da condição predeterminada.

Em uma realização preferida, o decodificador aritmético é configurado para mapear um código de símbolo em um valor espectral decodificado.

Em uma realização preferida, o decodificador aritmético é configurado para avaliar valores espectrais de uma primeira região de frequência de tempo, para detectar um grupo de uma pluralidade de valores espectrais que atenda, individualmente ou considerado juntamente, à condição predeterminada em relação a suas magnitudes. O decodificador aritmético é configurado para obter um valor numérico que representa o estado de contexto, em dependência de valores espectrais de uma segunda região de frequência de tempo, que é diferente da primeira região de frequência de tempo, se a condição predeterminada não for atendida. Descobriu-se que é recomendável detectar um grupo de uma pluralidade de valores espectrais que atenda à condição predeterminada em relação à magnitude dentro de uma região que difere da região normalmente utilizada para a computação de

contexto. Isso se deve ao fato de que uma extensão, por exemplo, uma extensão de frequência, de regiões compreendendo valores espectrais comparativamente pequenos ou valores espectrais comparativamente grandes, é tipicamente maior que uma dimensão de uma região de valores espectrais que devem ser considerados para um cálculo numérico de um valor numérico que representa o estado de contexto. Da mesma forma, é recomendável analisar as diferentes regiões para a detecção de um grupo de uma pluralidade de valores espectrais que atenda à condição predeterminada e para a computação numérica de um valor numérico que representa o estado de contexto (em que o cálculo numérico pode somente ser esperado em uma segunda etapa se a detecção não prover um bit).

Em uma realização preferida, o decodificador aritmético é configurado para avaliar uma ou mais tabelas de dispersão para selecionar uma regra de mapeamento em dependência do estado de contexto. Descobriu-se que a seleção da regra de mapeamento pode ser controlada pelo mecanismo de detecção de uma pluralidade de valores espectrais adjacentes que atenda à condição predeterminada.

Uma realização, de acordo com a invenção, cria um codificador de áudio para prover uma informação de áudio codificada, com base em uma informação de áudio de entrada. O codificador de áudio compreende um conversor de domínio de compactação de energia para domínio de tempo para frequência para prover uma representação de áudio de domínio de frequência, com base em uma representação de domínio de tempo da informação de áudio de entrada, de modo que a representação de áudio de domínio de frequência compreenda um conjunto de valores espectrais. O

codificador de áudio também compreende um codificador aritmético configurado para codificar um valor espectral ou uma versão pré-processada dele, utilizando uma senha de extensão variável. O codificador aritmético é configurado para mapear um valor espectral ou um valor de um plano de bits mais significativo de um valor espectral em um valor de código. O codificador aritmético é configurado para selecionar uma regra de mapeamento que descreve um mapeamento de um valor espectral ou de um plano de bits mais significativo de um valor espectral em um valor de código em dependência do estado de contexto. O codificador aritmético é configurado para determinar o estado de contexto atual em dependência de uma pluralidade de valores espectrais adjacentes previamente codificados. O codificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente codificados, que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes e para determinar o estado de contexto atual em dependência de um resultado da detecção.

Esse codificador de sinal de áudio tem base nas mesmas descobertas que o decodificador de sinal de áudio discutidas acima. Descobriu-se que o mecanismo para adaptação do contexto, que apresentou ser eficiente para a decodificação de um conteúdo de áudio, também deve ser aplicado no lado do codificador, a fim de permitir um sistema consistente.

Uma realização, de acordo com a invenção, cria um método para prover informações de áudio decodificado com base na informação de áudio codificada.

Ainda, outra realização, de acordo com a

invenção, cria um método para prover informação de áudio codificada com base em uma informação de áudio de entrada.

Outra realização, de acordo com a invenção, cria um programa de computador para realizar um dos ditos métodos.

Os métodos e o programa de computador têm base nas mesmas descobertas que a do decodificador de áudio descrito acima e do codificador de áudio descrito acima.

BREVE DESCRIÇÃO DAS FIGURAS

As realizações, de acordo com a presente invenção, serão subsequentemente descritas tendo como referência as figuras anexas, nas quais:

A Figura 1 apresenta um diagrama de blocos esquemático de um codificador de áudio, de acordo com uma realização da invenção;

A Figura 2 apresenta um diagrama de blocos esquemático de um decodificador de áudio, de acordo com uma realização da invenção;

A Figura 3 apresenta uma representação de código de pseudo-programa de um algoritmo “value_decode()” para decodificar um valor espectral;

A Figura 4 apresenta uma representação esquemática de um contexto para um cálculo de estado;

A Figura 5a apresenta a representação de código de pseudo-programa de um algoritmo “arith_map_context ()” para mapear um contexto;

As Figuras 5b e 5c apresentam uma representação de código de pseudo-programa de um algoritmo “arith_get_context ()” para obter um valor de estado de contexto;

A Figura 5d apresenta uma representação de código de pseudo-programa de um algoritmo “get_pk(s)” para derivar um valor índice da tabela de frequências cumulativas „pki” de uma variável de estado;

A Figura 5e apresenta uma representação de código de pseudo-programa de um algoritmo “arith_get_pk(s)” para derivar um valor índice da tabela de frequências cumulativas „pki” de um valor de estado;

A Figura 5f apresenta uma representação de código de pseudo-programa de um algoritmo “get_pk(unsigned long s)” para derivar um valor índice da tabela de frequências cumulativas „pki” de um valor de estado;

A Figura 5g apresenta uma representação de código de pseudo-programa de um algoritmo “arith_decode ()” para decodificar aritmeticamente um símbolo de uma senha de extensão variável;

A Figura 5h apresenta uma representação de código de pseudo-programa de um algoritmo “arith_update_context ()” para atualizar o contexto;

A Figura 5i apresenta uma legenda de definições e variáveis;

A Figura 6a apresenta uma representação de sintaxe de um bloco de dados brutos da codificação de fala e áudio unificada (USAC);

A Figura 6b apresenta uma representação de sintaxe de um elemento de canal único;

A Figura 6c apresenta a representação de sintaxe de um elemento de par de canais;

A Figura 6d apresenta uma representação de sintaxe de uma informação de controle “ics”;

A Figura 6e apresenta uma representação de sintaxe de uma corrente de canal de domínio de frequência;

A Figura 6f apresenta uma representação de sintaxe de dados espectrais aritmeticamente codificados;

A Figura 6g apresenta uma representação de sintaxe para decodificar um conjunto de valores espectrais;

A Figura 6h apresenta uma legenda de elementos e variáveis de dados;

A Figura 7 apresenta um diagrama de blocos esquemático de um codificador de áudio, de acordo com outra realização da invenção;

A Figura 8 apresenta um diagrama de blocos esquemático de um decodificador de áudio, de acordo com outra realização da invenção;

A Figura 9 apresenta uma disposição para uma comparação de uma codificação silenciosa, de acordo com o projeto de trabalho 3 do padrão de projeto da USAC com um esquema de codificação, de acordo com a presente invenção:

A Figura 10a apresenta uma representação esquemática de um contexto para um cálculo de estado, conforme é utilizado, de acordo com o projeto de trabalho 4 do padrão de projeto da USAC;

A Figura 10b apresenta uma representação esquemática de um contexto para um cálculo de estado, conforme é utilizado nas realizações, de acordo com a invenção;

A Figura 11a apresenta uma visão geral da tabela,

conforme utilizada no esquema de codificação aritmética, de acordo com o projeto de trabalho 4 do padrão de projeto da USAC;

A Figura 11b apresenta uma visão geral da tabela, conforme utilizada no esquema de codificação aritmética, de acordo com a presente invenção;

A Figura 12a apresenta uma representação gráfica de uma demanda de memória de somente leitura para os esquemas de codificação silenciosa, de acordo com a presente invenção e de acordo com o projeto de trabalho 4 do padrão de projeto da USAC;

A Figura 12b apresenta uma representação gráfica de uma demanda de memória de somente leitura de dados do decodificador USAC total, de acordo com a presente invenção e de acordo com o conceito de acordo com o projeto de trabalho 4 do padrão de projeto da USAC;

A Figura 13a apresenta uma representação de tabela de taxas de bits médias que são utilizadas por um codificador de codificação de fala e áudio unificada, utilizando um codificador aritmético, de acordo com o projeto de trabalho 3 do padrão de projeto da USAC e um decodificador aritmético, de acordo com uma realização da presente invenção;

A Figura 13b apresenta uma representação de tabela de um controle de reservatório de bits para um codificador de codificação de fala e áudio unificada, utilizando o codificador aritmético de acordo com o projeto de trabalho 3 do padrão de projeto da USAC e o codificador aritmético de acordo com uma realização da presente invenção;

A Figura 14 apresenta uma representação de tabela de taxas de bits médias para um codificador de USAC, de acordo com

o projeto de trabalho 3 do padrão de projeto da USAC, e de acordo com uma realização da presente invenção;

A Figura 15 apresenta uma representação de tabela de taxas de bits mínimas, máximas e médias de USAC na base de uma estrutura;

A Figura 16 apresenta uma representação de tabela dos melhores e piores casos na base de estrutura;

As Figuras 17(1) e 17(2) apresentam uma representação de tabela de um conteúdo de uma tabela "ari_s_hash[387]";

A Figura 18 apresenta uma representação de tabela de um conteúdo de uma tabela "ari_gs_hash[225]";

As Figuras 19(1) e 19(2) apresentam uma representação de tabela de um conteúdo de uma tabela "ari_cf_m[64][9]"; e

As Figuras 20(1) e 20(2) apresentam uma representação de tabela de um conteúdo de uma tabela "ari_s_hash[387].

DESCRIÇÃO DETALHADA DAS REALIZAÇÕES

1. CODIFICADOR DE ÁUDIO, DE ACORDO COM A FIGURA 7

A Figura 7 apresenta um diagrama de blocos esquemático de um codificador de áudio, de acordo com uma realização da invenção. O codificador de áudio 700 é configurado para receber uma informação de áudio de entrada 710 e para prover, em sua base, uma informação de áudio codificada 712. O codificador de áudio compreende um conversor de domínio de tempo para domínio de frequência de compactação de energia 720 que é configurado para prover uma representação de áudio de domínio de frequência 722 com

base em uma representação de domínio de tempo da informação de áudio de entrada 710, de modo que a representação de áudio de domínio de frequência 722 compreenda um conjunto de valores espectrais. O codificador de áudio 700 também compreende um codificador aritmético 730 configurado para codificar um valor espectral (fora do conjunto de valores espectrais que forma a representação de áudio de domínio de frequência 722), ou uma versão pré-processada dele, utilizando uma senha de extensão variável para obter a informação de áudio codificada 712 (que pode compreender, por exemplo, uma pluralidade de senhas de extensão variável).

O codificador aritmético 730 é configurado para mapear um valor espectral ou um valor de um plano de bits mais significativo de um valor espectral em um valor de código (isto é, em uma senha de extensão variável), em dependência de um estado de contexto. O codificador aritmético 730 é configurado para selecionar uma regra de mapeamento que descreve um mapeamento de um valor espectral ou de um plano de bits mais significativo de um valor espectral em um valor de código, em dependência de um estado de contexto. O codificador aritmético é configurado para determinar o estado de contexto atual em dependência de uma pluralidade de valores espectrais previamente codificados (preferencial, mas não necessariamente, adjacentes). Para este fim, o codificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente codificados, que atenda, individualmente ou considerada juntamente, a uma condição predeterminada em relação a suas magnitudes, e determinar o estado de contexto atual em dependência

de um resultado da detecção.

Como pode ser visto, o mapeamento de um valor espectral ou de um plano de bits mais significativo de um valor espectral em um valor de código pode ser realizado por uma codificação de valor espectral 740 utilizando uma regra de mapeamento 742. Um rastreador de estado 750 pode ser configurado para rastrear o estado de contexto e pode compreender um detector de grupo 752 para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente codificados que atenda, individualmente ou considerada juntamente, à condição predeterminada em relação a suas magnitudes. O rastreador de estado 750 também é preferencialmente configurado para determinar o estado de contexto atual em dependência do resultado da dita detecção realizada pelo detector de grupo 752. Da mesma forma, o rastreador de estado 750 provê uma informação 754 que descreve o estado de contexto atual. Um seletor de regra de mapeamento 760 pode selecionar uma regra de mapeamento, por exemplo, uma tabela de frequências cumulativas, que descreve um mapeamento de um valor espectral ou de um plano de bits mais significativo de um valor espectral em um valor de código. Da mesma forma, o seletor de regra de mapeamento 760 provê as informações de regra de mapeamento 742 à codificação espectral 740.

Para resumir o dito acima, o codificador de áudio 700 realiza uma codificação aritmética de uma representação de áudio de domínio de frequência provida pelo conversor de domínio de tempo para domínio de frequência. A codificação aritmética é dependente do contexto, de modo que uma regra de mapeamento (por exemplo, uma tabela de frequências cumulativas) seja selecionada

em dependência de valores espectrais previamente codificados. Da mesma forma, valores espectrais adjacentes no tempo e/ou frequência (ou pelo menos, dentro de um ambiente predeterminado) entre si e/ou ao valor espectral atualmente codificado (isto é, valores espectrais dentro de um ambiente predeterminado do valor espectral atualmente codificado) são considerados na codificação aritmética para ajustar a distribuição de probabilidade avaliada pela codificação aritmética. Ao selecionar uma regra de mapeamento adequada, uma detecção é realizada a fim de detectar se há um grupo de uma pluralidade de valores espectrais adjacentes previamente codificados que atenda, individualmente ou considerado juntamente, à condição predeterminada em relação a suas magnitudes. O resultado dessa detecção é aplicado na seleção do estado de contexto atual, isto é, na seleção de uma regra de mapeamento. Ao detectar se há um grupo de uma pluralidade de valores espectrais que são particularmente pequenos ou particularmente grandes, é possível reconhecer aspectos especiais dentro da representação de áudio de domínio de frequência, que podem ser uma representação de frequência de tempo. Os aspectos especiais como, por exemplo, um grupo de uma pluralidade de valores espectrais particularmente pequenos ou particularmente grandes, indicam que um estado de contexto específico deve ser utilizado, uma vez que esse estado de contexto específico pode prover uma eficiência de codificação particularmente boa. Assim, a detecção do grupo de valores espectrais adjacentes que atenda à condição predeterminada, que é tipicamente utilizada em combinação a uma avaliação de contexto alternativa com base em uma combinação de uma pluralidade de valores espectrais previamente codificados,

provê um mecanismo que permite uma seleção eficiente de um contexto adequado se a informação de áudio de entrada considerar alguns estados especiais (por exemplo, compreender uma variação de frequência encoberta ampla).

Da mesma forma, uma codificação eficiente pode ser alcançada enquanto mantém o cálculo do contexto suficientemente simples.

2. DECODIFICADOR DE ÁUDIO, DE ACORDO COM A FIGURA 8

A Figura 8 apresenta um diagrama de blocos esquemático de um decodificador de áudio 800. O decodificador de áudio 800 é configurado para receber uma informação de áudio codificada 810 e para prover, com base nisso, uma informação de áudio decodificado 812. O decodificador de áudio 800 compreende um decodificador aritmético 820 que é configurado para prover uma pluralidade de valores espectrais decodificados 822 com base em uma representação aritmeticamente codificada 821 dos valores espectrais. O decodificador de áudio 800 também compreende um conversor de domínio de frequência para domínio de tempo 830 que é configurado para receber os valores espectrais decodificados 822 e para prover a representação de áudio de domínio de tempo 812, que pode constituir as informações de áudio decodificado, utilizando os valores espectrais decodificados 822, a fim de obter uma informação de áudio decodificado 812.

O decodificador aritmético 820 compreende um determinador de valor espectral 824 que é configurado para mapear um valor de código da representação aritmeticamente codificada 821 de valores espectrais em um código de símbolo que representa um ou

mais dos valores espectrais decodificados ou pelo menos uma parte (por exemplo, um plano de bits mais significativo) de um ou mais dos valores espectrais decodificados. O determinador de valor espectral 824 pode ser configurado para realizar o mapeamento em dependência de uma regra de mapeamento, que pode ser descrita por uma informação de regra de mapeamento 828a.

O decodificador aritmético 820 é configurado para selecionar uma regra de mapeamento (por exemplo, uma tabela de frequências cumulativas) que descreve um mapeamento de um valor de código (descrito pela representação aritmeticamente codificada 821 de valores espectrais) em um código de símbolo (que descreve um ou mais valores espectrais) em dependência de um estado de contexto (que pode ser descrito pela informação de estado de contexto 826a). O decodificador aritmético 820 é configurado para determinar o estado de contexto atual em dependência de uma pluralidade de valores espectrais previamente decodificados 822. Para esse fim, um rastreador de estado 826 pode ser utilizado, que recebe uma informação que descreve os valores espectrais previamente decodificados. O decodificador aritmético também é configurado para detectar um grupo de uma pluralidade de valores espectrais previamente decodificados (preferencial, mas não necessariamente, adjacentes), que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes, e para determinar o estado de contexto atual (descrito, por exemplo, pela informação de estado de contexto 826a) em dependência de um resultado da detecção.

A detecção do grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados que atenda à

condição predeterminada em relação a suas magnitudes pode, por exemplo, ser realizada por um detector de grupo, que é parte do rastreador de estado 826. Da mesma forma, uma informação de estado de contexto atual 826a é obtida. A seleção da regra de mapeamento pode ser realizada por um seletor de regra de mapeamento 828, que deriva uma informação de regra de mapeamento 828a da informação de estado de contexto atual 826a, e que provê a informação de regra de mapeamento 828a ao determinador de valor espectral 824.

Em relação à funcionalidade do decodificador de sinal de áudio 800, deve ser observado que o decodificador aritmético 820 é configurado para selecionar uma regra de mapeamento (por exemplo, uma tabela de frequências cumulativas) que é, em uma média, bem adaptada ao valor espectral a ser decodificado, uma vez que a regra de mapeamento é selecionada em dependência do estado de contexto atual, que por sua vez é determinado em dependência de uma pluralidade de valores espectrais previamente decodificados. Da mesma forma, dependências estatísticas entre valores espectrais adjacentes a serem decodificados podem ser exploradas. Ademais, ao detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes, é possível adaptar a regra de mapeamento a condições especiais (ou padrões) de valores espectrais previamente decodificados. Por exemplo, uma regra de mapeamento específica pode ser selecionada se um grupo de uma pluralidade de valores espectrais comparativamente pequenos adjacentes previamente decodificados for identificado ou se um grupo de uma pluralidade

de valores espectrais comparativamente grandes adjacentes previamente decodificados for identificado. Descobriu-se que a presença de um grupo de valores espectrais comparativamente grandes ou de um grupo de valores espectrais comparativamente pequenos pode ser considerada uma indicação significativa de que uma regra de mapeamento dedicada, especialmente adaptada para essa condição, deve ser utilizada. Da mesma forma, uma computação de contexto pode ser facilitada (ou acelerada) ao explorar a detecção desse um grupo de uma pluralidade de valores espectrais. Também, podem ser consideradas características de um conteúdo de áudio que não poderiam ser consideradas facilmente sem aplicar o conceito mencionado acima. Por exemplo, a detecção de um grupo de uma pluralidade de valores espectrais que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes, pode ser realizada com base em um conjunto de valores espectrais diferente, quando comparado ao conjunto de valores espectrais utilizado para uma computação de contexto normal.

Detalhes adicionais serão descritos abaixo.

3. CODIFICADOR DE ÁUDIO, DE ACORDO COM A FIGURA 1

A seguir, será descrito um codificador de áudio, de acordo com uma realização da presente invenção. A Figura 1 apresenta um diagrama de blocos esquemático desse um codificador de áudio 100.

O codificador de áudio 100 é configurado para receber uma informação de áudio de entrada 110 e para prover, com base nisso, um fluxo de bits 112, que constitui uma informação de áudio codificada. O codificador de áudio 100 opcionalmente

compreende um pré-processador 120, que é configurado para receber a informação de áudio de entrada 110 e para prover, com base nisso, uma informação de áudio de entrada pré-processada 110a. O codificador de áudio 100 também compreende um transformador de sinal de domínio de tempo para domínio de frequência de compactação de energia 130, que também é designado como conversor de sinal. O conversor de sinal 130 é configurado para receber a informação de áudio de entrada 110, 110a e para prover, com base nisso, uma informação de áudio de domínio de frequência 132, que preferencialmente toma forma de um conjunto de valores espectrais. Por exemplo, o transformador de sinal 130 pode ser configurado para receber uma estrutura da informação de áudio de entrada 110, 110a (por exemplo, um bloco de amostras de domínio de tempo) e para prover um conjunto de valores espectrais que representa o conteúdo de áudio da respectiva estrutura de áudio. Além disso, o transformador de sinal 130 pode ser configurado para receber uma pluralidade de estruturas de áudio, de sobreposição ou sem sobreposição, subsequentes da informação de áudio de entrada 110, 110a e para prover, com base nisso, uma representação de áudio de domínio de frequência de tempo, que compreende uma sequência de conjuntos subsequentes de valores espectrais, um conjunto de valores espectrais associado a cada estrutura.

O transformador de sinal de domínio de tempo para domínio de frequência de compactação de energia 130 pode conter um banco de filtro de compactação de energia, que provê valores espectrais associados a diferentes variações de frequência, de sobreposição ou sem sobreposição, diferentes. Por exemplo, o transformador de sinal 130 pode compreender um transformador de

MDCT de janelamento 130a, que é configurado para janelar a informação de áudio de entrada 110, 110a (ou uma estrutura sua) utilizando uma janela de transformação e para realizar uma transformada do cosseno discreta modificada da informação de áudio de entrada janelada 110, 110a (ou de sua estrutura janelada). Da mesma forma, a representação de áudio de domínio de frequência 132 pode compreender um conjunto de, por exemplo, 1024 valores espectrais na forma de coeficientes de MDCT associados a uma estrutura da informação de áudio de entrada.

O codificador de áudio 100 pode ainda, opcionalmente, compreender um pós-processador espectral 140, que é configurado para receber a representação de áudio de domínio de frequência 132 e para prover, com base nisso, uma representação de áudio de domínio de frequência pós-processada 142. O pós-processador espectral 140 pode, por exemplo, ser configurado para realizar uma formação de ruído temporal e/ou uma previsão de longo prazo e/ou qualquer pós-processamento espectral conhecido na técnica. O codificador de áudio ainda compreende, opcionalmente, um escalonador/quantificador 150, que é configurado para receber a representação de áudio de domínio de frequência 132 ou a sua versão pós-processada 142 e para prover uma representação de áudio de domínio de frequência escalonada e quantificada 152.

O codificador de áudio 100 ainda compreende, opcionalmente, um processador de modelo psicoacústico 160, que é configurado para receber a informação de áudio de entrada 110 (ou sua versão pós-processada 110a) e para prover, com base nisso, uma informação de controle opcional, que pode ser utilizada para o controle do transformador de sinal de domínio de tempo para

domínio de frequência de compactação de energia 130, para o controle do pós-processador espectral opcional 140 e/ou para o controle do escalonador/quantificador opcional 150. Por exemplo, o processador de modelo psicoacústico 160 pode ser configurado para analisar a informação de áudio de entrada, para determinar quais componentes da informação de áudio de entrada 110, 110a são particularmente importantes para a percepção humana do conteúdo de áudio e quais componentes da informação de áudio de entrada 110, 110a são menos importantes para a percepção do conteúdo de áudio. Da mesma forma, o processador de modelo psicoacústico 160 pode prover informação de controle, que é utilizada pelo codificador de áudio 100 a fim de ajustar o escalonamento da representação de áudio de domínio de frequência 132, 142 pelo escalonador/quantificador 150 e/ou a resolução de quantificação aplicada pelo escalonador/quantificador 150. Consequentemente, faixas de fator de escala perceptualmente importantes (isto é, grupos de valores espectrais adjacentes que são particularmente importantes para a percepção humana do conteúdo de áudio) são escalonadas com um fator de escalonamento grande e quantificado com resolução comparativamente alta, enquanto faixas de fator de escala perceptualmente menos importantes (isto é, grupos de valores espectrais adjacentes) são escalonadas com um fator de escalonamento comparativamente menor e quantificadas com uma resolução de quantificação comparativamente menor. Da mesma forma, os valores espectrais escalonados de frequências mais importantes são tipicamente maiores de modo significativo que valores espectrais de frequências perceptualmente menos importantes.

O codificador de áudio também compreende um

codificador aritmético 170, que é configurado para receber a versão escalonada e quantificada 152 da representação de áudio de domínio de frequência 132 (ou, de modo alternativo, a versão pós-processada 142 da representação de áudio de domínio de frequência 132 ou até a representação de áudio de domínio de frequência 132 em si) e para prover informações de senha aritméticas 172a com base nisso, de modo que as informações de senha aritméticas representem a representação de áudio de domínio de frequência 152.

O codificador de áudio 100 também compreende um formatador de carga útil de fluxo de bits 190, que é configurado para receber as informações de senha aritméticas 172a. O formatador de carga útil de fluxo de bits 190 também é tipicamente configurado para receber informações adicionais, como, por exemplo, informações de fator de escala que descrevem quais fatores de escala foram aplicados pelo escalonador/quantificador 150. Além disso, o formatador de carga útil de fluxo de bits 190 pode ser configurado para receber outras informações de controle. O formatador de carga útil de fluxo de bits 190 é configurado para prover o fluxo de bits 112 com base nas informações recebidas pela montagem do fluxo de bits de acordo com uma sintaxe de fluxo de bits desejada, que será discutida abaixo.

A seguir, serão descritos detalhes em relação ao codificador aritmético 170. O codificador aritmético 170 é configurado para receber uma pluralidade de valores espectrais escalonados, quantificados e pós-processados da representação de áudio de domínio de frequência 132. O codificador aritmético compreende um extrator de plano de bits mais significativo 174, que é configurado para extrair um plano de bits mais significativo

m de um valor espectral. Deve ser observado aqui que o plano de bits mais significativo pode compreender um ou até mais bits (por exemplo, dois ou três bits), que são os bits mais significativos do valor espectral. Assim, o extrator de plano de bits mais significativo 174 provê um valor de plano de bits mais significativo 176 de um valor espectral.

O codificador aritmético 170 também compreende um primeiro determinador de senha 180, que é configurado para determinar uma senha aritmética acod_m [pki][m] que representa o valor de plano de bits mais significativo m. Opcionalmente, o determinador de senha 180 também pode prover uma ou mais senhas de escape (também aqui designadas com “ARITH_ESCAPE”) indicando, por exemplo, quantos planos de bits menos significativos estão disponíveis (e, consequentemente, indicando a ponderação numérica do plano de bits mais significativo). O primeiro determinador de senha 180 pode ser configurado para prover a senha associada a um valor de plano de bits mais significativo m utilizando uma tabela de frequências cumulativas selecionada tendo (ou sendo mencionada por) um índice de tabela de frequências cumulativas pki.

A fim de determinar como qual tabela de frequências cumulativas deve ser selecionada, o codificador aritmético preferencialmente compreende um rastreador de estado 182, que é configurado para rastrear o estado do codificador aritmético, por exemplo, ao observar quais valores espectrais foram codificados anteriormente. O rastreador de estado 182 provê consequentemente uma informação de estado 184, por exemplo, um valor de estado designado com “s” ou “t”. O codificador aritmético 170 também compreende um seletor de tabela de frequências

cumulativas 186, que é configurada para receber a informação de estado 184 e para prover uma informação 188 que descreve a tabela de frequências cumulativas selecionada ao determinador de senha 180. Por exemplo, o seletor de tabela de frequências cumulativas 186 pode prover um índice de tabela de frequências cumulativas „pki” que descreve qual tabela de frequências cumulativas, fora de um conjunto de 64 tabelas de frequências cumulativas, é selecionada para uso pelo determinador de senha. De modo alternativo, o seletor de tabela de frequências cumulativas 186 pode prover toda a tabela de frequências cumulativas selecionada ao determinador de senha. Assim, o determinador de senha 180 pode utilizar a tabela de frequências cumulativas selecionada para a provisão da senha acod_m[pki][m] do valor de plano de bits mais significativo m, de modo que a senha real acod_m[pki][m] que codifica o valor de plano de bits mais significativo m é dependente do valor de m e do índice de tabela de frequências cumulativas pki e, consequentemente, da informação de estado atual 184. Detalhes adicionais em relação ao processo de codificação e o formato de senha obtido serão descritos abaixo.

O codificador aritmético 170 ainda compreende um extrator de plano de bits menos significativo 189a, que é configurado para extrair um ou mais planos de bits menos significativos da representação de áudio de domínio de frequência escalonada e quantificada 152, se um ou mais dos valores espectrais a serem codificados excederem a variação de valores codificáveis utilizando o plano de bits mais significativo somente. Os planos de bits menos significativos podem compreender um ou mais bits, conforme desejado. Da mesma forma, o extrator de

plano de bits menos significativo 189a provê uma informação de plano de bit menos significativo 189b. O codificador aritmético 170 também compreende um segundo determinador de senha 189c, que é configurado para receber a informação de plano de bit menos significativo 189d e para prover, com base nisso, 0, 1 ou mais senhas "acod_r" que representam o conteúdo de 0, 1 ou mais planos de bits menos significativos. O segundo determinador de senha 189c pode ser configurado para aplicar um algoritmo de codificação aritmética ou qualquer outro algoritmo de codificação a fim de derivar as senhas de plano de bits menos significativos "acod_r" da informação de plano de bit menos significativo 189b.

Deve ser observado aqui que o número de planos de bits menos significativos pode variar em dependência do valor dos valores espectrais escalonados e quantificados 152, de modo que possa não haver mais plano de bits menos significativo, se o valor espectral escalonado e quantificado a ser codificado for comparativamente pequeno, de modo que possa haver um plano de bits menos significativo se o valor espectral escalonado e quantificado atual a ser codificado for de uma variação média e de modo que haja mais de um plano de bits menos significativo se o valor espectral escalonado e quantificado a ser codificado considerar um valor comparativamente grande.

Para resumir o mencionado acima, o codificador aritmético 170 é configurado para codificar valores espectrais escalonados e quantificados, que são descritos pela informação 152, utilizando um processo de codificação hierárquico. O plano de bits mais significativo (compreendendo, por exemplo, um, dois ou três bits por valor espectral) é codificado para obter uma senha

aritmética “acod_m[pki][m]” de um valor de plano de bits mais significativo. Um ou mais planos de bits menos significativos (cada um dos planos de bits menos significativos compreendendo, por exemplo, um, dois ou três bits) são codificados para obter uma ou mais senhas “acod_r”. Ao codificar o plano de bits mais significativo, o valor m do plano de bits mais significativo é mapeado a uma senha acod_m[pki][m]. Para este fim, 64 diferentes tabelas de frequências cumulativas estão disponíveis para a codificação do valor m em dependência de um estado do codificador aritmético 170, isto é, em dependência de valores espectrais previamente codificados. Da mesma forma, a senha “acod_m[pki][m]” é obtida. Além disso, uma ou mais senhas “acod_r” são providas e incluídas no fluxo de bits se um ou mais planos de bits menos significativos estiverem presentes.

DESCRIÇÃO DO REAJUSTE

O codificador de áudio 100 pode opcionalmente ser configurado para decidir se uma melhoria na taxa de bits pode ser obtida ao reajustar um contexto, por exemplo, ao ajustar o índice de estado a um valor padrão. Da mesma forma, o codificador de áudio 100 pode ser configurado para prover uma informação de reajuste (por exemplo, chamada de “arith_reset_flag”) que indica se o contexto para a codificação aritmética é reajustado e também que indica se o contexto para a decodificação aritmética em um decodificador correspondente deve ser reajustada.

Os detalhes em relação ao formato de fluxo de bits e as tabelas de frequências cumulativas aplicadas serão discutidos abaixo.

4. DECODIFICADOR DE ÁUDIO

A seguir, um decodificador de áudio, de acordo com uma realização da invenção, será descrito. A Figura 2 apresenta um diagrama de blocos esquemático desse um decodificador de áudio 200.

O decodificador de áudio 200 é configurado para receber um fluxo de bits 210, que representa uma informação de áudio codificada e que pode ser idêntico ao fluxo de bits 112 provido pelo codificador de áudio 100. O decodificador de áudio 200 provê uma informação de áudio decodificado 212 com base no fluxo de bits 210.

O decodificador de áudio 200 compreende um formatador de carga útil de fluxo de bits opcional 220, que é configurado para receber o fluxo de bits 210 e para extrair do fluxo de bits 210 uma representação de áudio de domínio de frequência codificada 222. Por exemplo, o formatador de carga útil de fluxo de bits 220 pode ser configurado para extrair do fluxo de bits 210 dados espectrais aritmeticamente codificados como, por exemplo, uma senha aritmética “acod_m [pki][m]” que representa o valor de plano de bits mais significativo m de um valor espectral a e uma senha “acod_r” que representa o conteúdo de um plano de bits menos significativo do valor espectral a da representação de áudio de domínio de frequência. Assim, a representação de áudio de domínio de frequência codificada 222 constitui (ou compreende) uma representação aritmeticamente codificada de valores espectrais. O desformatador de carga útil de fluxo de bits 220 é ainda configurado para extrair do fluxo de bits informações adicionais de controle, que não são apresentadas na Figura 2. Além disso, o desformatador de carga útil de fluxo de bits é opcionalmente

configurado para extrair do fluxo de bits 210 uma informação de reajuste de estado 224, que também é designada como sinalização de reajuste aritmético ou “arith_reset_flag”.

O decodificador de áudio 200 compreende um decodificador aritmético 230, que também é designado como “decodificador silencioso espectral”. O decodificador aritmético 230 é configurado para receber a representação de áudio de domínio de frequência codificada 220 e, opcionalmente, a informação de reajuste de estado 224. O decodificador aritmético 230 também é configurado para prover uma representação de áudio de domínio de frequência decodificada 232, que pode compreender uma representação decodificada de valores espectrais. Por exemplo, a representação de áudio de domínio de frequência decodificada 232 pode compreender uma representação decodificada de valores espectrais, que são descritos pela representação de áudio de domínio de frequência codificada 220.

O decodificador de áudio 200 também compreende um quantificador inverso/re-escalonador opcional 240, que é configurado para receber a representação de áudio de domínio de frequência decodificada 232 e para prover, com base nisso, uma representação de áudio de domínio de frequência quantificada inversamente e re-escalonada 242.

O decodificador de áudio 200 ainda compreende um pré-processador espectral opcional 250, que é configurado para receber a representação de áudio de domínio de frequência quantificada inversamente e re-escalonada 242 e para prover, com base nisso, uma versão pré-processada 252 da representação de áudio de domínio de frequência quantificada inversamente e re-

escalonada 242. O decodificador de áudio 200 também compreende um transformador de sinal de domínio de frequência para domínio de tempo 260, que também é designado como “conversor de sinal”. O transformador de sinal 260 é configurado para receber a versão pré-processada 252 da representação de áudio de domínio de frequência quantificada inversamente e re-escalonada 242 (ou, de modo alternativo, a representação de áudio de domínio de frequência quantificada inversamente e re-escalonada 242 ou a representação de áudio de domínio de frequência decodificada 232) e para prover, com base nisso, uma representação de domínio de tempo 262 das informações de áudio. O transformador de sinal de domínio de frequência para domínio de tempo 260 pode, por exemplo, compreender um transformador para realizar uma transformada de cosseno discreta modificada inversa (IMDCT) e um janelamento adequado (assim como outras funcionalidades auxiliares, como, por exemplo, uma sobreposição e adição).

O decodificador de áudio 200 pode ainda compreender um pós-processador de domínio de tempo opcional 270, que é configurado para receber a representação de domínio de tempo 262 das informações de áudio e para obter as informações de áudio decodificado 212 utilizando um pós-processamento de domínio de tempo. Entretanto, se o pós-processamento for omitido, a representação de domínio de tempo 262 pode ser idêntica às informações de áudio decodificado 212.

Deve ser observado aqui que o quantificador inverso/re-escalonador 240, o pré-processador espectral 250, o transformador de sinal de domínio de frequência para domínio de tempo 260 e o pós-processador de domínio de tempo 270 podem ser

controlados em dependência da informação de controle, que é extraída do fluxo de bits 210 pelo desformatador de carga útil de fluxo de bits 220.

Para resumir a funcionalidade geral do decodificador de áudio 200, uma representação de áudio de domínio de frequência decodificada 232, por exemplo, um conjunto de valores espectrais associado a uma estrutura de áudio da informação de áudio codificada, pode ser obtido com base na representação de domínio de frequência codificada 222 utilizando o decodificador aritmético 230. Subsequentemente, o conjunto de, por exemplo, 1024 valores espectrais, que podem ser coeficientes de MDCT, são inversamente quantificados, re-escalonados e pré-processados. Da mesma forma, um conjunto de valores espectrais quantificados inversamente, re-escalonados e pré-processados espectralmente (por exemplo, 1024 coeficientes de MDCT) é obtido. Posteriormente, uma representação de domínio de tempo de uma estrutura de áudio é derivada do conjunto de valores de domínio de frequência quantificados inversamente, re-escalonados e pré-processados espectralmente (por exemplo, coeficientes de MDCT). Da mesma forma, uma representação de domínio de tempo de uma estrutura de áudio é obtida. A representação de domínio de tempo de uma determinada estrutura de áudio pode ser combinada às representações de domínio de tempo de estruturas de áudio anteriores e/ou subsequentes. Por exemplo, uma sobreposição e adição entre as representações de domínio de tempos de estruturas de áudio subsequentes pode ser realizada a fim de suavizar as transições entre as representações de domínio de tempo das estruturas de áudio adjacentes e a fim de obter um cancelamento de

aliasing. Para detalhes em relação à reconstrução das informações de áudio decodificado 212 com base na representação de áudio de domínio de frequência decodificada de tempo 232, é feita referência, por exemplo, ao Padrão Internacional ISO/IEC 14496-3, parte 3, subparte 4, no qual é dada uma discussão detalhada. Entretanto, outros esquemas de sobreposição e cancelamento de *aliasing* mais elaborados podem ser utilizados.

A seguir, alguns detalhes em relação ao decodificador aritmético 230 serão descritos. O decodificador aritmético 230 compreende um determinador de plano de bits mais significativo 284, que é configurado para receber a senha aritmética acod_m [pki][m] que descreve o valor de plano de bits mais significativo m. O determinador de plano de bits mais significativo 284 pode ser configurado para utilizar uma tabela de frequências cumulativas fora de um conjunto compreendendo uma pluralidade de 64 tabelas de frequências cumulativas para derivar o valor de plano de bits mais significativo m da senha aritmética “acod_m [pki][m]”.

O determinador plano de bits mais significativo 284 é configurado para derivar valores 286 de um plano de bits mais significativo de valores espectrais com base na senha acod_m. O decodificador aritmético 230 ainda compreende um determinador de plano de bits menos significativo 288, que é configurado para receber uma ou mais senhas “acod_r” que representam um ou mais planos de bits menos significativos de um valor espectral. Da mesma forma, o determinador de plano de bits menos significativo 288 é configurado para prover valores decodificados 290 de um ou mais planos de bits menos significativos. O decodificador de áudio

200 também compreende um combinador de plano de bits 292, que é configurado para receber os valores decodificados 286 do plano de bits mais significativo dos valores espectrais e os valores decodificados 290 de um ou mais planos de bits menos significativos dos valores espectrais se esses planos de bits menos significativos estiverem disponíveis para os valores espectrais atuais. Da mesma forma, o combinador de plano de bits 292 provê valores espectrais decodificados, que são parte da representação de áudio de domínio de frequência decodificada 232. Naturalmente, o decodificador aritmético 230 é tipicamente configurado para prover uma pluralidade de valores espectrais a fim de obter um conjunto completo de valores espectrais decodificados associado a uma estrutura atual do conteúdo de áudio.

O decodificador aritmético 230 ainda compreende um seletor de tabela de frequências cumulativas 296, que é configurado para selecionar uma das 64 tabelas de frequências cumulativas em dependência de um índice de estado 298 que descreve um estado do decodificador aritmético. O decodificador aritmético 230 ainda compreende um rastreador de estado 299, que é configurado para rastrear um estado do decodificador aritmético em dependência dos valores espectrais previamente decodificados. A informação de estado pode opcionalmente ser reajustada a uma informação de estado padrão em resposta à informação de reajuste de estado 224. Da mesma forma, o seletor de tabela de frequências cumulativas 296 é configurado para prover um índice (por exemplo, pki) de uma tabela de frequências cumulativas selecionada ou uma tabela de frequências cumulativas selecionada em si, para

aplicação na decodificação do valor de plano de bits mais significativo m em dependência da senha “acod_m”.

Para resumir a funcionalidade do decodificador de áudio 200, o decodificador de áudio 200 é configurado para receber uma representação de áudio de domínio de frequência codificada eficientemente da taxa de bits 222 e para obter uma representação de áudio de domínio de frequência decodificada com base nisso. No decodificador aritmético 230, que é utilizado para obter a representação de áudio de domínio de frequência decodificada 232 com base na representação de áudio de domínio de frequência codificada 222, uma probabilidade de diferentes combinações de valores do plano de bits mais significativo de valores espectrais adjacentes é explorada ao utilizar um decodificador aritmético 280, que é configurado para aplicar uma tabela de frequências cumulativas. Em outras palavras, as dependências estatísticas entre valores espectrais são exploradas ao selecionar diferentes tabelas de frequências cumulativas fora de um conjunto compreendendo 64 diferentes tabelas de frequências cumulativas em dependência de um índice de estado 298, que é obtido ao observar os valores espectrais decodificados previamente computados.

5. VISÃO GERAL DA FERRAMENTA DE CODIFICAÇÃO SILENCIOSA ESPECTRAL

A seguir, detalhes em relação ao algoritmo de codificação e decodificação, que é realizada, por exemplo, pelo codificador aritmético 170 e pelo decodificador aritmético 230 serão explicados.

É colocado foco na descrição do algoritmo de decodificação. Deve ser observado; entretanto, que um algoritmo de

codificação correspondente pode ser realizado de acordo com os ensinamentos do algoritmo de decodificação, em que os mapeamentos são inversos.

Deve ser observado que a decodificação, que será discutida a seguir, é utilizada a fim de permitir uma denominada "codificação silenciosa espectral" de valores espectrais, tipicamente pré-processados, escalonados e quantificados. A codificação silenciosa espectral é utilizada em um conceito de codificação/decodificação de áudio para reduzir adicionalmente a redundância do espectro quantificado, que é obtido, por exemplo, por um transformador de domínio de tempo para um domínio de frequência de compactação de energia.

O esquema da codificação silenciosa espectral, que é utilizado nas realizações da invenção, tem base em uma codificação aritmética em conjunto com um contexto dinamicamente adaptado. A codificação silenciosa é alimentada por (originais ou representações codificadas de) valores espectrais quantificados e utiliza, dependendo do contexto, as tabelas de frequências cumulativas derivadas, por exemplo, de uma pluralidade de valores espectrais próximos previamente decodificados. Aqui, a proximidade tanto em tempo como em frequência é considerada conforme ilustrado na Figura 4. As tabelas de frequências cumulativas (que serão explicadas abaixo) são então utilizadas pelo codificador aritmético para gerar um código binário de extensão variável e pelo decodificador aritmético para derivar valores decodificados de um código binário de extensão variável.

Por exemplo, o codificador aritmético 170 produz um código binário para um determinado conjunto de símbolos em

dependência das respectivas probabilidades. O código binário é gerado ao mapear um intervalo de probabilidade, no qual o conjunto de símbolos existe, para uma senha.

A seguir, outra breve visão geral da ferramenta de codificação silenciosa espectral será dada. A codificação silenciosa espectral é utilizada para reduzir adicionalmente a redundância do espectro quantificado. O esquema da codificação silenciosa espectral tem base na codificação aritmética em conjunto a um contexto dinamicamente adaptado. A codificação silenciosa é alimentada pelos valores espectrais quantificados e utiliza o contexto dependendo das tabelas de frequências cumulativas derivadas de, por exemplo, sete valores espectrais próximos previamente decodificados

Aqui, a proximidade em ambos, tempo e frequência, é considerada, conforme ilustrado na Figura 4. As tabelas de frequências cumulativas são então utilizadas pelo codificador aritmético para gerar um código binário de extensão variável.

O codificador aritmético produz um código binário para um determinado conjunto de símbolos e suas respectivas probabilidades. O código binário é gerado ao mapear um intervalo de probabilidade, onde o conjunto de símbolos existe para uma senha.

6. PROCESSO DE DECODIFICAÇÃO

6.1 VISÃO GERAL DO PROCESSO DE DECODIFICAÇÃO

A seguir, uma visão geral do processo de decodificação de um valor espectral será dado fazendo referência à Figura 3, que apresenta uma representação de código de pseudo-programa do processo de decodificação de uma pluralidade de

valores espectrais.

O processo de decodificação de uma pluralidade de valores espectrais compreende uma inicialização 310 de um contexto. A inicialização 310 do contexto compreende uma derivação do contexto atual de um contexto anterior utilizando a função “arith_map_context (lg)”. A derivação do contexto atual de um contexto anterior pode compreender um reajuste do contexto. Tanto o reajuste do contexto como a derivação do contexto atual de um contexto anterior serão discutidos abaixo.

A decodificação de uma pluralidade de valores espectrais também compreende uma iteração de uma decodificação do valor espectral 312 e uma atualização de contexto 314, essa atualização de contexto é realizada por uma função “Arith_update_context(a,i,lg) que é descrita abaixo. A decodificação do valor espectral 312 e a atualização de contexto 314 são repetidas lg vezes, em que lg indica o número de valores espectrais a serem decodificados (por exemplo, para uma estrutura de áudio). A decodificação do valor espectral 312 compreende um cálculo de valor de contexto 312a, uma decodificação de plano de bits mais significativo 312b e uma adição de plano de bits menos significativo 312c.

A computação do valor de estado 312a compreende a computação de um primeiro valor de estado s utilizando a função “arith_get_context(i, lg, arith_reset_flag, N/2)”, essa função retorna o primeiro valor de estado s. A computação do valor de estado 312a também compreende uma computação de um valor de nível “lev0” e de um valor de nível “lev”, esses valores de nível “lev0”, „lev” são obtidos ao trocar o primeiro valor de estado s

para a direita em 24 bits. A computação do valor de estado 312a também compreende uma computação de um segundo valor de estado t de acordo com a fórmula apresentada na Figura 3 no número de referência 312a.

A decodificação de plano de bits mais significativo 312b compreende uma execução iterativa de um algoritmo de decodificação 312ba, em que uma variável j é inicializada a 0 antes de uma primeira execução do algoritmo 312ba.

O algoritmo 312ba compreende uma computação de um índice de estado „pki” (que também serve como um índice de tabela de frequências cumulativas) em dependência do segundo valor de estado t, e também em dependência dos valores de nível „lev” e lev0, utilizando uma função “arith_get_pk()”, que é discutida abaixo. O algoritmo 312ba também compreende a seleção de uma tabela de frequências cumulativas em dependência do índice de estado pki, em que uma variável “cum_freq” pode ser ajustada a um endereço de início de uma das 64 tabelas de frequências cumulativas em dependência do índice de estado pki. Também, uma variável “cfl” pode ser inicializada a uma extensão da tabela de frequências cumulativas selecionada, que é, por exemplo, igual ao número de símbolos no alfabeto, isto é, o número de diferentes valores que podem ser decodificados. As extensões de todas as tabelas de frequências cumulativas de “arith_cf_m[pki=0][9]” a “arith_cf_m[pki=63][9]” disponíveis para a decodificação do valor de plano de bits mais significativo m é 9, uma vez que oito diferentes valores de plano de bits mais significativo e um símbolo de escape pode ser decodificado. Subsequentemente, um

valor de plano de bits mais significativo m pode ser obtido ao executar uma função “arith_decode()”, levando em consideração a tabela de frequências cumulativas selecionada (descrita pela variável “cum_freq” e pela variável “cfl”). Ao derivar o valor de plano de bits mais significativo m, os bits chamados “acod_m” do fluxo de bits 210 podem ser avaliados (vide, por exemplo, a Figura 6g).

O algoritmo 312ba também compreende a verificação se o valor de plano de bits mais significativo m é igual a um símbolo de escape “ARITH_ESCAPE” ou não. Se o valor de plano de bits mais significativo m não for igual ao símbolo de escape aritmético, o algoritmo 312ba é abortado (condição de “rompimento”) e as instruções remanescentes do algoritmo 312ba são, portanto, puladas. Da mesma forma, a execução do processo é continuada com o ajuste do valor espectral a para ser igual ao valor de plano de bits mais significativo m (instrução “a=m”). Ao contrário, se o valor de plano de bits mais significativo decodificado m for idêntico ao símbolo de escape aritmético “ARITH_ESCAPE”, o valor de nível „lev” é aumentado em um. Conforme mencionado, o algoritmo 312ba é, então, repetido até o valor de plano de bits mais significativo decodificado m ser diferente do símbolo de escape aritmético.

Assim que a decodificação de plano de bits mais significativo é concluída, isto é, um valor de plano de bits mais significativo m diferente do símbolo de escape aritmético for decodificado, o valor espectral variável „a” é ajustado para ser igual ao valor de plano de bits mais significativo m. Subsequentemente, os planos de bits menos significativos são

obtidos, por exemplo, conforme apresentado no número de referência 312c na Figura 3. Para cada plano de bits menos significativo do valor espectral, um dos dois valores binários é decodificado. Por exemplo, um valor de plano de bits menos significativo r é obtido. Subsequentemente, o valor espectral variável „a” é atualizado ao trocar o conteúdo do valor espectral variável „a” para a esquerda em 1 bit e ao adicionar o valor de plano de bits menos significativo decodificado atualmente r como um bit menos significativo. Entretanto, deve ser observado que o conceito para obter os valores dos planos de bits menos significativos não é de relevância particular para a presente invenção. Em algumas realizações, a decodificação de quaisquer planos de bits menos significativos pode até ser omitida. De modo alternativo, diferentes algoritmos de decodificação podem ser utilizados para este fim.

6.2 ORDEM DA DECODIFICAÇÃO, DE ACORDO COM A FIGURA 4

A seguir, a ordem de decodificação dos valores espectrais será descrita.

Coeficientes espectrais são codificados silenciosamente e transmitidos (por exemplo, no fluxo de bits) começando com o coeficiente de menor frequência e progredindo para o coeficiente de maior frequência.

Os coeficientes de uma codificação de áudio avançada (por exemplo, obtidos utilizando uma transformada do cosseno discreta modificada, conforme discutido no ISO/IEC 14496, parte 3, subparte 4) são armazenados em um arranjo chamado “x_ac_quant[g][win][sfb][bin]”, e a ordem de transmissão da senha

de codificação silenciosa (por exemplo acod_m, acod_r) se de modo que, quando eles forem decodificados na ordem recebida e armazenada no arranjo, "caixa" (o índice de frequência), seja o índice de incrementação mais rápida e "g" seja o índice de incrementação mais rápida.

Coeficientes espectrais associados a uma frequência menor são codificados antes dos coeficientes espectrais associados a uma frequência maior.

Coeficientes da excitação codificada de transformação (tcx) são armazenados diretamente em um arranjo x_tcx_invquant[win][bin], e a ordem da transmissão das senhas de codificação silenciosa é de modo que, quando eles são decodificados na ordem recebida e armazenados no arranjo, "caixa" é o índice de incrementação mais rápida e "win" é o índice de incrementação mais devagar. Em outras palavras, se os valores espectrais descrevem uma excitação codificada de transformação do filtro de previsão linear de um codificador de fala, os valores espectrais a são associados a frequências adjacentes e de aumento da excitação codificada de transformação.

Coeficientes espectrais associados a uma frequência menor são codificados antes dos coeficientes espectrais associados a uma frequência maior.

Notavelmente, o decodificador de áudio 200 pode ser configurado para aplicar a representação de áudio de domínio de frequência decodificada 232, que é provida pelo decodificador aritmético 230, tanto para uma geração "direta" de um sinal de áudio de representação de domínio de tempo utilizando uma transformação de sinal de domínio de frequência para domínio de

tempo como para uma provisão “indireta” de uma representação de sinal de áudio utilizando tanto um decodificador de domínio de frequência para domínio de tempo como um filtro de previsão linear excitado pela saída do transformador de sinal de domínio de frequência para domínio de tempo.

Em outras palavras, o decodificador aritmético 200, cuja funcionalidade é aqui discutida em detalhes, é bem adequado para decodificar valores espectrais de uma representação de domínio de frequência de tempo de um conteúdo de áudio codificado no domínio de frequência e para a provisão de uma representação de domínio de frequência de tempo de um sinal de estímulo para um filtro de previsão linear adaptado para decodificar um sinal de fala decodificado no domínio de previsão linear. Assim, o decodificador aritmético é bem adequado para uso em um decodificador de áudio que é capaz de administrar tanto o conteúdo de áudio codificado por domínio de frequência quanto o conteúdo de áudio codificado por domínio de frequência de previsão linear (modo de domínio de previsão linear de excitação codificada de transformação).

6.3. INICIALIZAÇÃO DE CONTEXTO, DE ACORDO COM AS FIGURAS 5A E 5B

A seguir, a inicialização de contexto (também designada como um “mapeamento de contexto”), que é realizada em uma etapa 310, será descrita.

A inicialização de contexto compreende um mapeamento entre um contexto anterior e um contexto atual, de acordo com o algoritmo “arith_map_ contexto()”, que é apresentado na Figura 5a. Como pode ser visto, o contexto atual é armazenado

em uma variável global q[2][n_context] que toma forma de um arranjo tendo uma primeira dimensão de dois e uma segunda dimensão de n_context. Um contexto anterior é um armazenado em uma variável qs[n_context], que toma a forma de uma tabela tendo uma dimensão de n_context. A variável "previous_lg" descreve um número de valores espectrais de um contexto anterior.

A variável "lg" descreve um número de coeficientes espectrais para decodificar na estrutura. A variável "previous_lg" descreve um número anterior de linhas espectrais de uma estrutura anterior.

Um mapeamento do contexto pode ser realizado de acordo com o algoritmo "arith_map_context()". Deve ser observado aqui que a função "arith_map_context()" ajusta as entradas q[0][i] do arranjo de contexto atual q aos valores qs[i] do arranjo de contexto anterior qs, se o número de valores espectrais associado à estrutura de áudio atual (por exemplo, codificada por domínio de frequência) for idêntico ao número de valores espectrais associado à estrutura anterior de áudio para i=0 a i=lg-1.

Entretanto, um mapeamento mais complicado é realizado se o número de valores espectrais associado à estrutura de áudio atual for diferente do número de valores espectrais associado à estrutura anterior de áudio. Entretanto, detalhes em relação ao mapeamento, nesse caso, não são particularmente relevantes para a ideia principal da presente invenção, de modo que é feita referência ao código de pseudo-programa da Figura 5a para detalhes.

6.4 COMPUTAÇÃO DO VALOR DE ESTADO, DE ACORDO COM AS FIGURAS 5B E 5C

A seguir, a computação do valor de estado 312a será descrita em mais detalhes.

Deve ser observado que o primeiro valor de estado s (conforme apresentado na Figura 3) pode ser obtido como um valor de retorno da função "arith_get_context(i, lg, arith_reset_flag, N/2)", uma representação de código de pseudo-programa que é apresentada nas Figuras 5b e 5c.

Em relação à computação do valor de estado, é feita referência à Figura 4, que apresenta o contexto utilizado para uma avaliação de estado. A Figura 4 apresenta uma representação bidimensional de valores espectrais, tanto ao longo do tempo como de frequência. Uma abscissa 410 descreve o tempo, e uma ordenada 412 descreve a frequência. Como pode ser visto na Figura 4, um valor espectral 420 para decodificar, é associado a um índice de tempo t0 e um índice de frequência i. Como pode ser visto, para o índice de tempo t0, os tuplos tendo índices de frequência i-1, i-2 e i-3 já são decodificados no momento em que o valor espectral 420 tendo o índice de frequência i deve ser decodificado. Como pode ser visto da Figura 4, um valor espectral 430 tendo um índice de tempo t0 e um índice de frequência i-1 já é decodificado antes de o valor espectral 420 ser decodificado, e o valor espectral 430 é considerado para o contexto que é utilizado para a decodificação do valor espectral 420. Semelhantemente, um valor espectral 434 tendo um índice de tempo t0 e um índice de frequência i-2, já é decodificado antes de o valor espectral 420 ser decodificado, e o valor espectral 434 é considerado para o contexto que é utilizado para decodificar o valor espectral 420. Semelhantemente, um valor espectral 440 tendo um índice de tempo

t-1 e um índice de frequência de i-2, um valor espectral 444 tendo um índice de tempo t-1 e um índice de frequência i-1, um valor espectral 448 tendo um índice de tempo t-1 e um índice de frequência i, um valor espectral 452 tendo um índice de tempo t-1 e um índice de frequência i+1 e um valor espectral 456 tendo um índice de tempo t-1 e um índice de frequência i+2, já são decodificados antes de o valor espectral 420 ser decodificado, e são considerados para a determinação do contexto, que é utilizado para decodificar o valor espectral 420. Os valores espectrais (coeficientes) já decodificados no momento em que o valor espectral 420 é decodificado e considerados para o contexto são apresentados por quadrados sombreados. Ao contrário, alguns outros valores espectrais já decodificados (no momento em que o valor espectral 420 é decodificado), que são representados por quadrados tendo linhas tracejadas, e outros valores espectrais, que ainda não são decodificados (no momento em que o valor espectral 420 é decodificado) e que são apresentados por círculos tendo linhas tracejadas, não são utilizados para determinar o contexto para decodificar o valor espectral 420.

Entretanto, deve ser observado que alguns desses valores espectrais, que não são utilizados para a computação “regular” (ou “normal”) do contexto para decodificar o valor espectral 420 podem, todavia, ser avaliados para uma detecção de uma pluralidade de valores espectrais adjacentes previamente decodificados que atende, individualmente ou considerada juntamente, à condição predeterminada em relação a suas magnitudes.

Tendo como referência agora às Figuras 5b e 5c,

que apresentam a funcionalidade da função “arith_get_context()” na forma de um código de pseudo-programa, alguns detalhes a mais em relação ao cálculo do primeiro valor de contexto “s”, que é realizado pela função “arith_get_context()”, serão descritos.

Deve ser observado que a função “arith_get_context()” recebe, como variáveis de entrada, um índice i do valor espectral para decodificar. O índice i é tipicamente um índice de frequência. Uma variável de entrada lg descreve um número (total) de coeficientes quantificados esperados (para uma estrutura de áudio atual). Uma variável N descreve um número de linhas da transformação. Um sinalizador “arith_reset_flag” indica se o contexto deve ser reajustado. A função “arith_get_context” provê, como um valor de saída, uma variável „t”, que representa um índice de estado concatenado s e um nível de plano de bits previsto lev0.

A função “arith_get_context()” utiliza variáveis inteiras a0, c0, c1, c2, c3, c4, c5, c6, lev0, e “região”.

A função “arith_get_context()” compreende como blocos funcionais principais, um primeiro processamento de reajuste aritmético 510, uma detecção 512 de um grupo de uma pluralidade de valores espectrais zero adjacentes previamente decodificados, um primeira configuração de variável 514, uma segunda configuração de variável 516, uma adaptação de nível 518, uma configuração de valor de região 520, uma adaptação de nível 522, uma limitação de nível 524, um processamento de reajuste aritmético 526, uma terceira configuração de variável 528, uma quarta configuração de variável 530, uma quinta configuração de variável 532, uma adaptação de nível 534 e uma computação de valor

de retorno seletiva 536.

No primeiro processamento de reajuste aritmético 510, é verificado se a sinalização de reajuste aritmético "arith_reset_flag" está ajustada, enquanto o índice do valor espectral para decodificar é igual a zero. Nesse caso, um valor de contexto de zero é retornado e a função é abortada.

Na detecção 512 de um grupo de uma pluralidade de valores espectrais zero previamente decodificados, que é somente realizada se a sinalização de reajuste aritmético estiver inativa e o índice i do valor espectral para decodificar for diferente de zero, uma variável chamada "sinalização" é inicializada em 1, conforme apresentado no número de referência 512a, e uma região de valor espectral que deve ser avaliada é determinada, conforme apresentado no número de referência 512b. Subsequentemente, a região de valores espectrais, que é determinada conforme apresentado no número de referência 512b, é avaliada, conforme apresentado no número de referência 512c. Se for descoberto que há uma região suficiente de valores espectrais zero previamente decodificados, um valor de contexto de 1 é retornado, conforme apresentado no número de referência 512d. Por exemplo, um limite de índice de frequência superior "lim_max" é ajustado a i+6, a menos que o índice i do valor espectral a ser decodificado esteja próximo a um índice de frequência máximo lg-1, em cujo caso uma configuração especial do limite de índice de frequência superior é feita, conforme apresentado no número de referência 512b. Ademais, um limite de índice de frequência inferior "lim_min" é ajustado a -5, a menos que o índice i do valor espectral para decodificar esteja próximo a zero (i+lim_min<0), em cujo caso uma computação

especial do limite de índice de frequência inferior lim_min é realizada, conforme apresentado no número de referência 512b. Ao avaliar a região de valores espectrais determinada na etapa 512b, uma avaliação é primeiro realizada para índices de frequência negativos k entre o limite de índice de frequência inferior lim_min e zero. Para índices de frequência k entre lim_min e zero, é verificado se pelo menos um dos valores de contexto q[0][k].c e q[1][k].c é igual a zero. Se, entretanto, ambos os valores de contexto q[0][k].c e q[1][k].c forem diferentes de zero para quaisquer índices de frequência k entre lim_min e zero, conclui-se que não há grupo de valores espectrais zero suficientes e a avaliação 512c é abortada. Subsequentemente, valores de contexto q[0][k].c para índices de frequência entre zero e lim_max são avaliados. Se for descoberto que qualquer um dos valores de contexto q[0][k].c para qualquer um dos índices de frequência entre zero e lim_max é diferente de zero, conclui-se que não há grupo de valores espectrais zero previamente decodificados suficiente, e a avaliação 512c é abortada. Se, entretanto, descobriu-se que para todos os índices de frequência k entre lim_min e zero, há pelo menos um valor de contexto q[0][k].c ou q[1][k].c que é igual a zero e se houver um valor de contexto zero q[0][k].c para cada índice de frequência k entre zero e lim_max, conclui-se que há um grupo de valores espectrais zero previamente decodificados suficiente. Da mesma forma, um valor de contexto de 1 é retornado nesse caso para indicar essa condição, sem qualquer cálculo adicional. Em outras palavras, os cálculos 514, 516, 518, 520, 522, 524, 526, 528, 530, 532, 534, 536 são pulados, se um grupo de uma pluralidade de valores de contexto suficiente

q[0][k].c, q[1][k].c tendo um valor de zero for identificado. Em outras palavras, o valor de contexto retornado, que descreve o estado de contexto (s), é determinado independente dos valores espectrais previamente decodificados em resposta da detecção que a condição predeterminada é atendida.

De outra forma, isto é, se não houver grupo de valores de contexto suficiente [q][0][k].c, [q][1][k].c, que são zero, pelo menos algumas das computações 514, 516, 518, 520, 522, 524,526, 528, 530, 532, 534, 536 são executadas.

Se a primeira configuração de variável 514, que é executada seletivamente se (e somente se) o índice i do valor espectral a ser decodificado for menor que 1, a variável a_0 é inicializada para obter o valor de contexto q[1][i-1], e a variável c0 é inicializada para obter o valor absoluto da variável a0. A variável „lev0” é inicializada para obter o valor zero. Subsequentemente, as variáveis „lev0” e c0 são aumentadas se a variável a0 compreender um valor absoluto comparativamente grande, isto é, for menor que -4 ou maior ou igual a 4. O aumento das variáveis „lev0” e c0 é realizado iterativamente, até que o valor da variável a0 seja trazido a uma variação entre -4 e 3 por uma operação de troca para a direita (etapa 514b).

Subsequentemente, as variáveis c0 e „lev0” são limitadas a valores máximos de 7 e 3, respectivamente (etapa 514c).

Se o índice i do valor espectral a ser decodificado for igual a 1 e a sinalização de reajuste aritmético (“arith_reset_flag”) estiver ativa, um valor de contexto é retornado, que é computado meramente com base nas variáveis c0 e

lev0 (etapa 514d). Da mesma forma, somente um único valor espectral previamente decodificado tendo o mesmo índice de tempo que o valor espectral para decodificar e tendo um índice de frequência que é menor, em 1, que o índice de frequência i do valor espectral a ser decodificado é considerado para a computação de contexto (etapa 514d). De outra forma, isto é, se não houver funcionalidade de reajuste aritmético, a variável c4 é inicializada (etapa 514e).

Para concluir, na primeira configuração de variável 514, as variáveis c0 e „lev0” são inicializadas em dependência de um valor espectral previamente decodificado, decodificado para a mesma estrutura que o valor espectral a ser atualmente decodificado e para uma caixa espectral anterior i-1. A variável c4 é inicializada em dependência de um valor espectral previamente decodificado, decodificado para uma estrutura anterior de áudio (tendo índice de tempo t-1) e tendo uma frequência que é menor (por exemplo, por uma caixa de frequência) que a frequência associada ao valor espectral a ser atualmente decodificado.

A segunda configuração de variável 516 que é executada seletivamente se (e somente se) o índice de frequência do valor espectral a ser atualmente decodificado for maior que 1, compreende uma inicialização das variáveis c1 e c6 e uma atualização da variável lev0. A variável c1 é atualizada em dependência de um valor de contexto q[1][i-2].c associado a um valor espectral previamente decodificado da estrutura de áudio atual, uma frequência que é menor (por exemplo, por duas caixas de frequência) que uma frequência de um valor espectral a ser decodificado atualmente. Semelhantemente, a variável c6 é

inicializada em dependência de um valor de contexto q[0][i-2].c, que descreve um valor espectral previamente decodificado de uma estrutura anterior (tendo índice de tempo t-1), uma frequência associada que é menor que (por exemplo, por duas caixas de frequência) que uma frequência associada ao valor espectral a ser decodificado atualmente. Além disso, a variável de nível „lev0” é ajustada a um valor de nível q[1][i-2].l associado a um valor espectral previamente decodificado da estrutura atual, uma frequência associada que é menor (por exemplo, por duas caixas de frequência) que uma frequência associada ao valor espectral a ser decodificado atualmente, se q[1][i-2].l for maior que lev0.

A adaptação de nível 518 e a configuração de valor de região 520 são seletivamente executadas, se (e somente se) o índice i do valor espectral a ser decodificado for maior que 2. Na adaptação de nível 518, a variável de nível „lev0” é aumentada a um valor de q[1][i-3].l, se o valor de nível q[1][i-3].l que é associado a um valor espectral previamente decodificado da estrutura atual, uma frequência associada que é menor (por exemplo, por três caixas de frequência) que a frequência associada ao valor espectral a ser decodificado atualmente, for maior que o valor de nível lev0.

Na configuração de valor de região 520, uma variável de “região” é ajustada em dependência de uma avaliação, na qual a região espectral, fora de uma pluralidade de regiões espectrais, o valor espectral a ser decodificado atualmente é disposto. Por exemplo, se for descoberto que o valor espectral a ser atualmente decodificado é associado a uma caixa de frequência (tendo o índice de caixa de frequência i) que está no primeiro (o

menor) quarto das caixas de frequência ($0 \leq i < N/4$), a variação de "região" da região é ajustada a zero. De outra forma, se o valor espectral a ser decodificado atualmente for associado a uma caixa de frequência que está em um segundo quarto das caixas de frequência associadas à estrutura atual ($N/4 \leq i < N/2$), a região variável é ajustada a um valor 1. De outra forma, isto é, se o valor espectral a ser decodificado atualmente for associado a uma caixa de frequência que está na segunda (superior) metade das caixas de frequência ($N/2 \leq i < N$), a região variável é ajustada a 2. Assim, a região variável é ajustada em dependência de uma avaliação à qual a região de frequência do valor espectral a ser decodificado atualmente está associada. Duas ou mais regiões de frequência podem ser diferenciadas.

Uma adaptação de nível adicional 522 é executada se (e somente se) o valor espectral a ser decodificado atualmente compreender um índice espectral que é maior que 3. Nesse caso, a variável de nível „lev0" é aumentada (ajustada ao valor q[1][i-4].l), se o valor de nível q[i][i-4].l, que é associado a um valor espectral previamente decodificado da estrutura atual, que é associada a uma frequência que é menor, por exemplo, por quatro caixas de frequência, que uma frequência associada ao valor espectral a ser decodificado atualmente for maior que o nível atual „lev0" (etapa 522). A variável de nível „lev0" se limita ao valor máximo de 3 (etapa 524).

Se uma condição de reajuste aritmético for detectada e o índice i do valor espectral a ser decodificado atualmente for maior que 1, o valor de estado é retornado em dependência das variáveis c0, c1, lev0, assim como em dependência

da variável de “região” da região (etapa 526). Da mesma forma, os valores espectrais previamente decodificados de quaisquer estruturas anteriores são deixados fora de consideração se uma condição de reajuste aritmético for determinada.

Na terceira configuração de variável 528, a variável c2 é ajustada ao valor de contexto q[0][i].c, que é associado a um valor espectral previamente decodificado da estrutura anterior de áudio (tendo índice de tempo t-1), esse valor espectral previamente decodificado é associada à mesma frequência que o valor espectral a ser decodificado atualmente.

Na quarta configuração de variável 530, a variável c3 é ajustada ao valor de contexto q[0][i+1].c, que é associado a um valor espectral previamente decodificado da estrutura anterior de áudio tendo um índice de frequência i+1, a menos que o valor espectral a ser decodificado atualmente estiver associado ao maior índice de frequência possível lg-1.

Na quinta configuração de variável 532, a variável c5 é ajustada ao valor de contexto q[0][i+2].c, que é associado a um valor espectral previamente decodificado da estrutura anterior de áudio tendo índice de frequência i+2, a menos que o índice de frequência i do valor espectral a ser decodificado atualmente estiver bem próximo ao valor índice de frequência máximo (isto é, tiver o valor de índice de frequência lg-2 ou lg-1).

Uma adaptação adicional da variável de nível „lev0” é realizada se o índice de frequência i for igual a zero (isto é, se o valor espectral a ser decodificado atualmente for o mais baixo valor espectral). Nesse caso, a variável de nível

„lev0” é aumentada de zero para 1, se a variável c2 ou c3 tiver um valor de 3, o que indica que um valor espectral previamente decodificado de uma estrutura anterior de áudio, que é associada à mesma frequência ou mesmo a uma frequência maior, quando comparada à frequência associada ao valor espectral a ser codificado atualmente, tem um valor comparativamente maior.

Na computação de valor de retorno seletiva 536, o valor de retorno é computado em dependência de se o índice i dos valores espectrais a ser decodificado atualmente tem o valor zero, 1 ou um valor maior. O valor de retorno é computado em dependência da variáveis c2, c3, c5 e lev0, conforme indicado no número de referência 536a, se o índice i tiver o valor zero. O valor de retorno é computado em dependência das variáveis c0, c2, c3, c4, c5, e „lev0”, conforme apresentado no número de referência 536b, se o índice i tiver o valor 1. O valor de retorno é computado em dependência da variável c0, c2, c3, c4, c1, c5, c6, “região” e lev0, se o índice i tiver um valor que é diferente de zero ou 1 (número de referência 536c).

Para resumir o mencionado acima, o valor de computação de contexto “arith_get_context()” compreende uma detecção 512 de um grupo de uma pluralidade de valores espectrais zero previamente decodificados (ou pelo menos, valores espectrais suficientemente pequenos). Se um grupo de valores espectrais zero previamente decodificados suficiente for encontrado, a presença de um contexto especial é indicado ao ajustar o valor de retorno a 1. De outra forma, o valor de computação de contexto é realizado. Pode ser geralmente dito que no valor de computação de contexto, o valor de índice i é avaliado a fim de decidir quantos valores

espectrais previamente decodificados devem ser avaliados. Por exemplo, um número de valores espectrais previamente decodificados avaliados é reduzido se um índice de frequência i do valor espectral a ser decodificado atualmente estiver próximo a um limite inferior (por exemplo, zero) ou próximo a um limite superior (por exemplo, lg-1). Além disso, mesmo se o índice de frequência i do valor espectral a ser decodificado atualmente estiver suficientemente longe de um valor mínimo, diferentes regiões espectrais são diferenciadas pela configuração de valor de região 520. Da mesma forma, diferentes propriedades estatísticas de diferentes regiões espectrais (por exemplo, a primeira região de frequência espectral baixa, segunda região de frequência espectral média e terceira região de frequência espectral alta) são consideradas. O valor de contexto, que é calculado com um valor de retorno, é dependente da variável de "região", de modo que o valor de contexto retornado dependa se um valor espectral a ser decodificado atualmente está em uma primeira região de frequência predeterminada ou em uma segunda região de frequência predeterminada (ou em qualquer outra região de frequência predeterminada).

6.5 SELEÇÃO DA REGRA DE MAPEAMENTO

A seguir, a seleção de uma regra de mapeamento, por exemplo, uma tabela de frequências cumulativas, que descreve um mapeamento de um valor de código em um código de símbolo, será descrita. A seleção da regra de mapeamento é feita em dependência do estado de contexto, que é descrito pelo valor de estado s ou t.

6.5.1 SELEÇÃO DA REGRA DE MAPEAMENTO UTILIZANDO O ALGORITMO, DE ACORDO COM A FIGURA 5D

A seguir, a seleção de uma regra de mapeamento utilizando a função “get_pk”, de acordo com a Figura 5d, será descrita. Deve ser observado que a função “get_pk” pode ser realizada para obter o valor de “pki” no sub-algoritmo 312ba do algoritmo da Figura 3. Assim, a função “get_pk” pode tomar o lugar da função “arith_get_pk” no algoritmo da Figura 3.

Também deve ser observado que uma função “get_pk”, de acordo com a Figura 5d, pode avaliar a tabela “ari_s_hash[387]”, de acordo com as Figuras 17(1) e 17(2), e uma tabela “ari_gs_hash”[225], de acordo com a Figura 18.

A função „get_pk” recebe, como uma variável de entrada, um valor de estado s, que pode ser obtido por uma combinação da variável „t”, de acordo com a Figura 3 e as variáveis “lev”, „lev0”, de acordo com a Figura 3. A função „get_pk” também é configurada para retornar, como um valor de retorno, um valor de uma variável “pki”, que designa uma regra de mapeamento ou uma tabela de frequências cumulativas. A função „get_pk” é configurada para mapear o valor de estado s em um valor índice de regra de mapeamento “pki”.

A função „get_pk” compreende uma primeira avaliação de tabela 540 e uma segunda avaliação de tabela 544. A primeira avaliação de tabela 540 compreende uma inicialização de variável 541 na qual as variáveis i_min, i_max, e i são inicializadas, conforme apresentado no número de referência 541. A primeira avaliação de tabela 540 também compreende uma busca de tabela iterativa 542, no curso da qual uma determinação é feita em relação a saber se há uma entrada da tabela “ari_s_hash” que corresponda ao valor de estado s. Se essa correspondência for

identificada durante a busca de tabela iterativa 542, a função get_pk é abortada, em que um valor de retorno da função é determinado pela entrada da tabela “ari_s_hash” que corresponde ao valor de estado s, conforme será explicado em mais detalhes. Se, entretanto, não for encontrada a correspondência perfeita entre o valor de estado s e uma entrada da tabela “ari_s_hash” durante o curso da busca de tabela iterativa 542, uma verificação de entrada limite 543 é realizada.

Retornando agora aos detalhes da primeira avaliação de tabela 540, pode ser visto que um intervalo de busca é definido pelas variáveis i_min e i_max. A busca de tabela iterativa 542 é repetida, contando que o intervalo definido pelas variáveis i_min e i_max seja suficientemente amplo, que pode ser verdadeira se a condição i_max-i_min > 1 for atendida. Subsequentemente, a variável i é ajustada, pelo menos aproximadamente, para designar o meio do intervalo (i=i_min+(i_max-i_min)/2). Subsequentemente, uma variável j é ajustada a um valor que é determinado pelo arranjo “ari_s_hash” em uma posição de arranjo designada pela variável i (número de referência 542). Deve ser observado aqui que cada entrada da tabela “ari_s_hash” descreve tanto um valor de estado, que é associado à entrada da tabela, como um valor índice de regra de mapeamento, que é associado à entrada da tabela. O valor de estado, que é associado à entrada da tabela, é descrito pelos bits mais significativos (bits 8 a 31) da entrada da tabela, enquanto os valores índice da regra de mapeamento são descritos pelos bits menores (por exemplo, bits 0 a 7) da dita entrada da tabela. O limite inferior i_min ou o limite superior i_max são adaptados em

dependência de se o valor de estado s é menor que um valor de estado descrito pelos 24 bits mais significativos da entrada “ari_s_hash[i]” da tabela “ari_s_hash” mencionada pela variável i. Por exemplo, se o valor de estado s for menor que o valor de estado descrito pelos 24 bits mais significativos da entrada “ari_s_hash[i]”, o limite superior i_max do intervalo da tabela é ajustado ao valor i. Da mesma forma, o intervalo da tabela para a próxima iteração da busca de tabela iterativa 542 é restrita à metade inferior do intervalo da tabela (de i_min a i_max) utilizado para a atual iteração da busca de tabela iterativa 542. Se, ao contrário, o valor de estado s for maior que os valores de estado descritos pelos 24 bits mais significativos da entrada da tabela “ari_s_hash[i]”, então, o limite inferior i_min do intervalo da tabela para a próxima iteração da busca de tabela iterativa 542 é ajustado ao valor i, de modo que a metade superior do intervalo da tabela atual (entre i_min e i_max) seja utilizada como o intervalo da tabela para a próxima busca de tabela iterativa. Se; entretanto, for descoberto que o valor de estado s é idêntico ao valor de estado descrito pelos 24 bits mais significativos da entrada da tabela “ari_s_hash[i]”, o valor índice de regra de mapeamento descrito pelos 8 bits menos significativos da entrada da tabela “ari_s_hash[i]” é retornado pela função “get_pk” e a função é abortada.

A busca de tabela iterativa 542 é repetida até que o intervalo da tabela definido pelas variáveis i_min e i_max seja suficientemente pequeno.

A verificação de entrada limite 543 é (opcionalmente) executada para suplementar a busca de tabela

iterativa 542. Se o índice variável i for igual à variável de índice i_max após a conclusão da busca de tabela iterativa 542, é feita uma verificação final se o valor de estado s é igual a um valor de estado descrito pelos 24 bits mais significativos de uma entrada da tabela "ari_s_hash[i_min]", e um valor índice de regra de mapeamento descrito pelos 8 bits menos significativos da entrada "ari_s_hash[i_min]" é retornado, nesse caso, como um resultado da função "get_pk". Ao contrário, se o índice variável i for diferente do índice variável i_max, então uma verificação é realizada quanto a saber se um valor de estado s é igual a um valor de estado descrito pelos 24 bits mais significativos da entrada da tabela "ari_s_hash[i_max]", e um valor índice de regra de mapeamento descrito pelos 8 bits menos significativos da dita entrada da tabela "ari_s_hash[i_max]" é retornado como um valor de retorno da função "get_pk" nesse caso.

Entretanto, deve ser observado que a verificação de entrada limite 543 pode ser considerada opcional em sua integridade.

Subsequente à primeira avaliação de tabela 540, a segunda avaliação de tabela 544 é realizada, a menos que um "atingimento direto" tenha ocorrido durante a primeira avaliação de tabela 540, em que o valor de estado s é idêntico a um dos valores de estado descritos pelas entradas da tabela "ari_s_hash" (ou, mais precisamente, pelos seus 24 bits mais significativos).

A segunda avaliação de tabela 544 compreende uma inicialização de variável 545, na qual o variáveis de índice i_min, i e i_max são inicializados, conforme apresentado no número de referência 545. A segunda avaliação de tabela 544 também

compreende uma busca de tabela iterativa 546, no curso da qual a tabela “ari_gs_hash” é pesquisada em relação a uma entrada que represente um valor de estado idêntico ao valor de estado s. Por fim, a segunda busca de tabela 544 compreende uma determinação de valor de retorno 547.

A busca de tabela iterativa 546 é repetida, contanto que o intervalo da tabela definido pelas variáveis de índice i_min e i_max seja amplo o suficiente (por exemplo, contando que i_max − i_min > 1). Na iteração da busca de tabela iterativa 546, a variável i é ajustada ao centro do intervalo da tabela definido por i_min e i_max (etapa 546a). Subsequentemente, uma entrada j da tabela “ari_gs_hash” é obtida em uma localização da tabela determinada pelo índice variável i (546b). Em outras palavras, a entrada da tabela “ari_gs_hash[i]” é uma entrada da tabela no centro do intervalo da tabela atual definido pelos índices da tabela i_min e i_max. Subsequentemente, o intervalo da tabela para a próxima iteração da busca de tabela iterativa 546 é determinado. Para esse fim, o valor de índice i_max que descreve o limite superior do intervalo da tabela é ajustado ao valor i, se o valor de estado s for menor que um valor de estado descrito pelos 24 bits mais significativos da entrada da tabela “j=ari_gs_hash[i]” (546c). Em outras palavras, a metade inferior do intervalo atual da tabela é selecionada como um novo intervalo da tabela para a próxima iteração da busca de tabela iterativa 546 (etapa 546c). De outra forma, se o valor de estado s for maior que um valor de estado descrito pelos 24 bits mais significativos da entrada da tabela “j=ari_gs_hash[i]”, o valor de índice i_min é ajustado ao valor i. Da mesma forma, a metade superior do

intervalo atual da tabela é selecionada como um novo intervalo da tabela para a próxima iteração da busca de tabela iterativa 546 (etapa 546d). Se, entretanto, for descoberto que o valor de estado s é idêntico a um valor de estado descrito pelos 24 bits mais elevados da entrada da tabela “j=ari_gs_hash[i]” , o índice variável i_max é ajustado ao valor i+1 ou ao valor 224 (se i+1 for maior que 224), e a busca de tabela iterativa 546 é abortada. Entretanto, se o valor de estado s for diferente do valor de estado descrito pelos 24 bits mais significativos de “j=ari_gs_hash[i]”, a busca de tabela iterativa 546 é repetida com o intervalo da tabela recentemente ajustado definido pelos valores índice atualizados i_min e i_max, a menos que o intervalo da tabela seja bem pequeno (i_max – i_min ≤ 1). Assim, o tamanho do intervalo do intervalo da tabela (definido por i_min e i_max) é iterativamente reduzido até um “atingimento direto” ser detectado (s==(j>>8)) ou o intervalo alcançar um tamanho permitido mínimo (i_max – i_min ≤ 1). Por fim, após um aborto da busca de tabela iterativa 546, uma entrada da tabela “j=ari_gs_hash[i_max]” é determinada e um valor índice de regra de mapeamento, que é descrito pelos 8 bits menos significativos da dita entrada da tabela “j=ari_gs_hash[i_max]”, é retornado ao valor de retorno da função “get_pk”. Da mesma forma, o valor índice de regra de mapeamento é determinado em dependência do limite superior i_max do intervalo da tabela (definido por i_min e i_max) após a conclusão ou aborto da busca de tabela iterativa 546.

As avaliações de tabela descritas acima 540, 544, ambas usam a busca de tabela iterativa 542, 546, permitem o exame das tabelas “ari_s_hash” e “ari_gs_hash” em relação à presença de

um determinado estado significativo com eficiência computacional muito alta. Em particular, um número de operações de acesso de tabela pode ser mantido razoavelmente pequeno, mesmo no pior caso. Descobriu-se que um ordenamento numérico das tabelas “ari_s_hash” e “ari_gs_hash” permite a aceleração da busca para um valor de dispersão adequado. Além disso, um tamanho de tabela pode ser mantido pequeno, uma vez que a inclusão de símbolos de escape nas tabelas “ari_s_hash” e “ari_gs_hash” não é necessário. Assim, um mecanismo de dispersão de contexto eficiente é estabelecido mesmo se houver um amplo número de diferentes estados: Em um primeiro estágio (primeira avaliação de tabela 540), uma busca por um atingimento direto é conduzida (s==(j>>8)).

No segundo estágio (segunda avaliação de tabela 544), variações do valor de estado s podem ser mapeadas em valores índice da regra de mapeamento. Assim, uma administração bem balanceada dos estados particularmente significativos, para a qual há uma entrada associada na tabela “ari_s_hash”, e estados menos significativos, para os quais há uma administração com base na variação, pode ser realizada. Da mesma forma, a função “get_pk” constitui uma implementação eficiente de uma seleção de regra de mapeamento.

Para quaisquer detalhes adicionais, é feita referência ao código de pseudo-programa da Figura 5d, que representa a funcionalidade da função “get_pk” em uma representação, de acordo com a linguagem de programação bem conhecida.

6.5.2 SELEÇÃO DE REGRA DE MAPEAMENTO UTILIZANDO O ALGORITMO, DE ACORDO COM A FIGURA 5E

A seguir, outro algoritmo para uma seleção da regra de mapeamento será descrito tendo como referência a Figura 5e. Deve ser observado que o algoritmo “arith_get_pk”, de acordo com a Figura 5e, recebe, como uma variável de entrada, um valor de estado s que descreve um estado do contexto. A função “arith_get_pk” provê, como um valor de saída ou valor de retorno, um índice “pki” de um modelo de probabilidade, que pode ser um índice para a seleção de uma regra de mapeamento, (por exemplo, uma tabela de frequências cumulativas).

Deve ser observado que a função „arith_get_pk”, de acordo com a Figura 5e, pode tomar a funcionalidade da função “arith_get_pk” da função “value_decode” da Figura 3.

Também deve ser observado que a função “arith_get_pk” pode, por exemplo, avaliar a tabela ari_s_hash, de acordo com a Figura 20, e a tabela ari_gs_hash, de acordo com a Figura 18.

A função “arith_get_pk”, de acordo com a Figura 5e, compreende uma primeira avaliação de tabela 550 e uma segunda avaliação de tabela 560. Na primeira avaliação de tabela 550, um escaneamento linear é feito na tabela ari_s_hash, para obter uma entrada j=ari_s_hash[i] da dita tabela. Se um valor de estado descrito pelos 24 bits mais significativos de uma entrada da tabela j=ari_s_hash[i] da tabela ari_s_hash for igual ao valor de estado s, um valor índice de regra de mapeamento „pki” descrito pelos 8 bits menos significativos da dita entrada da tabela identificada j=ari_s_hash[i] é retornado e a função “arith_get_pk” é abortada. Da mesma forma, todas as 387 entradas da tabela ari_s_hash são avaliadas em uma sequência ascendente a menos que

um “atingimento direto” (valor de estado s igual ao valor de estado descrito pelos 24 bits mais significativos de uma entrada da tabela j) seja identificado.

Se um atingimento direto não for identificado dentro da primeira avaliação de tabela 550, uma segunda avaliação de tabela 560 é executada. No curso da segunda avaliação de tabela, um escaneamento linear com índices de entrada i que aumentam linearmente de zero a um valor máximo de 224 é realizado. Durante a segunda avaliação de tabela, uma entrada “ari_gs_hash[i]” da tabela “ari_gs_hash” para a tabela i é lida, e a entrada da tabela “j=ari_gs_hash[i]” é avaliada de modo que seja determinado se o valor de estado representado pelos 24 bits mais significativos da entrada da tabela j é maior que o valor de estado s. Se esse for o caso, um valor índice de regra de mapeamento descrito pelos 8 bits menos significativos da dita entrada da tabela j é retornado como o valor de retorno da função “arith_get_pk”, e a execução da função “arith_get_pk” é abortada. Se; entretanto, o valor de estado s não for menor que o valor de estado descrito pelos 24 bits mais significativos da entrada da tabela atual j=ari_gs_hash[i], o escaneamento em todas as entradas da tabela ari_gs_hash é continuado ao aumentar o índice de tabela i. Se; entretanto, o valor de estado s for maior ou igual a qualquer um dos valores de estado descritos pelas entradas da tabela ari_gs_hash, um valor índice de regra de mapeamento „pki” definido pelos 8 bits menos significativos da última entrada da tabela ari_gs_hash é retornado como o valor de retorno da função “arith_get_pk”.

Para resumir, a função “arith_get_pk”, de acordo

com a Figura 5e, realiza uma dispersão de duas etapas. Em uma primeira etapa, uma busca de um atingimento direto é realizada, em que é determinado se o valor de estado s é igual ao valor de estado definido por qualquer uma das de uma primeira tabela "ari_s_hash". Se um atingimento direto for identificado na primeira avaliação de tabela 550, um valor de retorno é obtido da primeira tabela "ari_s_hash" e a função "arith_get_pk" é abortada. Se; entretanto, nenhum atingimento direto for identificado na primeira avaliação de tabela 550, a segunda avaliação de tabela 560 é realizada. Na segunda avaliação de tabela, uma avaliação com base na variação é realizada. As entradas subsequentes da segunda tabela "ari_gs_hash" define as variações. Se for descoberto que o valor de estado s existe dentro dessa variação (que é indicado pelo fato de que o valor de estado descrito pelos 24 bits mais significativos da entrada da tabela atual "j=ari_gs_hash[i]" é maior que o valor de estado s), o valor índice de regra de mapeamento "pki" descrito pelos 8 bits menos significativos da entrada da tabela j=ari_gs_hash[i] é retornado.

6.5.3 SELEÇÃO DA REGRA DE MAPEAMENTO UTILIZANDO O ALGORITMO, DE ACORDO COM A FIGURA 5F

A função "get_pk", de acordo com a Figura 5f, é substancialmente equivalente à função "arith_get_pk", de acordo com a Figura 5e. Da mesma forma, é feita referência à discussão acima. Para detalhes adicionais, é feita referência à representação de pseudo-programa na Figura 5f.

Deve ser observado que a função „get_pk", de acordo com a Figura 5f, pode tomar o lugar da função "arith_get_pk" chamada na função "value_decode" da Figura 3.

6.6. FUNÇÃO "ARITH DECODE()", DE ACORDO COM A FIGURA 5G

A seguir, a funcionalidade da função "arith_decode()" será discutida em detalhes tendo como referência a Figura 5g. Deve ser observado que a função "arith_decode()" utiliza a função auxiliadora "arith_first_symbol (void)", que retorna VERDADEIRO, se for o primeiro símbolo da sequência e FALSO, de outra forma. A função "arith_decode()" também utiliza a função auxiliadora "arith_get_next_bit(void)", que obtém e provê o próximo bit do fluxo de bits.

Além disso, a função "arith_decode()" utiliza as variáveis globais "baixa", "alta" e "valor". Ainda, a função "arith_decode()" recebe, como uma variável de entrada, a variável "cum_freq[]", que aponta para uma primeira entrada ou elemento (tendo índice de elemento ou índice de entrada 0) da tabela de frequências cumulativas selecionada. Também, a função "arith_decode()" utiliza a variável de entrada "cfl", que indica a extensão da tabela de frequências cumulativas selecionada designada pela variável "cum_freq[]".

A função "arith_decode()" compreende, como uma primeira etapa, uma inicialização de variável 570a, que é realizada se a função auxiliadora "arith_first_symbol()" indicar que um primeiro símbolo da sequência de símbolos está sendo decodificado. A inicialização de valor 550a inicializa a variável "valor" em dependência de uma pluralidade de, por exemplo, 20 bits, que são obtidos do fluxo de bits utilizando a função auxiliadora "arith_get_next_bit", de modo que a variável "valor" toma o valor representado pelos ditos bits. Também, a variável

"baixa" é inicializada para tomar o valor de 0 e a variável "alta" é inicializada para tomar o valor de 1048575.

Em uma segunda etapa 570b, a variável "variação" é ajustada a um valor, que é maior, em 1, que a diferença entre os valores das variáveis "alta" e "baixa". A variável "cum" é ajustada a um valor que representa uma posição relativa do valor da variável "valor" entre o valor da variável "baixa" e o valor da variável "alta". Da mesma forma, a variável "cum" toma, por exemplo, um valor entre 0 e 2^{16} em dependência do valor da variável "valor".

O apontador p é inicializado a um valor que é menor, em 1, que o endereço de início da tabela de frequências cumulativas selecionada.

O algoritmo "arith_decode()" também compreende uma busca de tabela de frequências cumulativas iterativa 570c. A busca de tabela de frequências cumulativas iterativa é repetida até que a variável cfl é menor ou igual a 1. Na busca de tabela de frequências cumulativas iterativa 570c, a variável de apontador q é ajustada a um valor, que é igual à soma do valor atual da variável de apontador p e metade do valor da variável "cfl". Se o valor da entrada *q da tabela de frequências cumulativas selecionada, essa entrada é endereçada pela variável de apontador q, é maior que o valor da variável "cum", a variável de apontador p é ajustada ao valor da variável de apontador q, e a variável "cfl" é incrementada. Por fim, a variável "cfl" é trocada para direita em um bit, com isso, dividindo efetivamente o valor da variável "cfl" em 2 e negligenciando a parte do módulo.

Da mesma forma, a busca de tabela de frequências

cumulativas iterativa 570c compara efetivamente o valor da variável “cum” a uma pluralidade de entradas da tabela de frequências cumulativas selecionada, a fim de identificar um intervalo dentro da tabela de frequências cumulativas selecionada, que é ligado pelas entradas da tabela de frequências cumulativas, de modo que o valor cum exista dentro do intervalo identificado. Da mesma forma, as entradas da tabela de frequências cumulativas selecionada definem intervalos, em que um valor de símbolo respectivo é associado a cada um dos intervalos da tabela de frequências cumulativas selecionada. Também, as amplitudes dos intervalos entre dois valores adjacentes da tabela de frequências cumulativas definem probabilidades dos símbolos associados aos ditos intervalos, de modo que a tabela de frequências cumulativas selecionada defina em sua integridade a distribuição de probabilidade dos diferentes símbolos (ou valores de símbolo). Detalhes em relação às tabelas de frequências cumulativas disponíveis serão discutidos abaixo tendo como referência a Figura 19.

Tendo como referência novamente a Figura 5g, o valor de símbolo é derivado do valor da variável de apontador p, em que o valor de símbolo é derivado conforme apresentado no número de referência 570d. Assim, a diferença entre o valor da variável de apontador p e o endereço de início “cum_freq” é avaliada a fim de obter o valor de símbolo, que é representado pela variável “símbolo”.

O algoritmo “arith_decode” também compreende uma adaptação 570e das variáveis “alta” e “baixa”. Se o valor de símbolo representado pela variável “símbolo” for diferente da 0, a

variável “alta” é atualizada, conforme apresentado no número de referência 570e. Também, o valor da variável “baixa” é atualizado, conforme apresentado no número de referência 570e. A variável “alta” é ajustada a um valor que é determinado pelo valor da variável “baixa”, a variável “variação” e a entrada tendo o índice “símbolo –1” da tabela de frequências cumulativas selecionada. A variável “baixa” é aumentada, em que a magnitude do aumento é determinada pela variável “variação” e a entrada da tabela de frequências cumulativas selecionada tendo o índice “símbolo”. Da mesma forma, a diferença entre os valores das variáveis “baixa” e “alta” é ajustada em dependência da diferença numérica entre duas entradas adjacentes da tabela de frequências cumulativas selecionada.

Da mesma forma, se um valor de símbolo tendo uma baixa probabilidade for detectado, o intervalo entre os valores das variáveis “baixa” e “alta” é reduzido a uma amplitude estreita. Ao contrário, se o valor de símbolo detectado compreender uma probabilidade relativamente ampla, a amplitude do intervalo entre os valores das variáveis “baixa” e “alta” é ajustada a um valor comparativamente grande. Novamente, a amplitude do intervalo entre os valores da variável “baixa” e “alta” é dependente do símbolo detectado e as entradas correspondentes da tabela de frequências cumulativas.

O algoritmo “arith_decode()” também compreende uma normalização de intervalo 570f, na qual o intervalo determinado na etapa 570e é iterativamente trocado e escalonado até que a condição de “quebra” seja alcançada. Na normalização de intervalo 570f, uma operação descendente da troca seletiva 570fa é

realizada. Se a variável “alta” for menor que 524286, nada é feito e a normalização de intervalo continua com uma operação de aumento do tamanho do intervalo 570fb. Se; entretanto, a variável “alta” não for menor que 524286 e a variável “baixa” for maior ou igual a 524286, as variáveis “valores”, “baixa” e “alta” são todas reduzidas em 524286, de modo que um intervalo definido pelas variáveis “baixa” e “alta” é trocado para baixo, e de modo que o valor da variável “valor” também é trocado para baixo. Se; entretanto, for descoberto que o valor da variável “alta” não é menor que 524286 e que a variável “baixa” não é maior ou igual a 524286 e que a variável “baixa” é maior ou igual a 262143 e que a variável “alta” é menor que 786429, as variáveis “valor”, “baixa” e “alta” são todas reduzidas em 262143, trocando assim para baixo o intervalo entre os valores das variáveis “alta” e “baixa” e também o valor da variável “valor”. Se; entretanto, nenhuma das condições acima for atendida, a normalização de intervalo é abortada.

Se; entretanto, qualquer uma das condições mencionadas acima, que são avaliadas na etapa 570fa, for atendida, a operação de aumento do intervalo 570fb é executada. Na operação de aumento do intervalo 570fb, o valor da variável “baixa” é dobrado. Também, o valor da variável “alta” é dobrado e o resultado da duplicação é aumentado em 1. Também, o valor da variável “valor” é dobrado (trocado para a esquerda em um bit) e um bit do fluxo de bits, que é obtido pela função auxiliadora “arith_get_next_bit” é utilizado como o bit menos significativo. Da mesma forma, o tamanho do intervalo entre os valores das variáveis “baixa” e “alta” é aproximadamente dobrado, e a precisão

da variável “valor” é aumentada ao utilizar um novo bit do fluxo de bits. Conforme mencionado acima, as etapas 570fa e 570fb são repetidas até que a condição de “quebra” seja atingida, isto é, até que o intervalo entre os valores das variáveis “baixa” e “alta” seja grande o suficiente.

Em relação à funcionalidade do algoritmo “arith_decode()”, deve ser observado que o intervalo entre os valores das variáveis “baixa” e “alta” é reduzido na etapa 570e em dependência de duas entradas adjacentes da tabela de frequências cumulativas mencionadas pela variável “cum_freq”. Se um intervalo entre dois valores adjacentes da tabela de frequências cumulativas selecionada for pequeno, isto é, se os valores adjacentes estiverem comparativamente próximos juntos, o intervalo entre os valores das variáveis “baixa” e “alta”, que é obtido na etapa 570e, será comparativamente pequeno. Ao contrário, se duas entradas adjacentes da tabela de frequências cumulativas forem espaçadas adicionalmente, o intervalo entre os valores das variáveis “baixa” e “alta”, que é obtido na etapa 570e, será comparativamente grande.

Consequentemente, se o intervalo entre os valores das variáveis “baixa” e “alta”, que é obtido na etapa 570e, for comparativamente pequeno, um amplo número de etapas de renormalização de intervalo será executado para re-escalonar o intervalo a um tamanho “suficiente” (de modo que nenhuma das condições da avaliação de condição 570fa seja atendida). Da mesma forma, um número comparativamente grande de bits do fluxo de bits será utilizado a fim de aumentar a precisão da variável “valor”. Se, ao contrário, o tamanho do intervalo obtido na etapa 570e for

comparativamente grande, somente um número menor de repetições das etapas de normalização do intervalo 570fa e 570fb será necessário a fim de normalizar novamente o intervalo entre os valores das variáveis “baixa” e “alta” a um tamanho “suficiente”. Da mesma forma, somente um número comparativamente pequeno de bits do fluxo de bits será utilizado para aumentar a precisão da variável “valor” e para preparar uma decodificação de um próximo símbolo.

Para resumir o mencionado acima, se um símbolo for decodificado, que compreende uma probabilidade comparativamente alta, e ao qual um intervalo grande é associado pelas entradas da tabela de frequências cumulativas selecionada, somente um número comparativamente pequeno de bits será lido do fluxo de bits a fim de permitir a decodificação de um símbolo subsequente. Ao contrário, se um símbolo for decodificado, que compreende uma probabilidade comparativamente pequena e ao qual um intervalo pequeno é associado pelas entradas da tabela de frequências cumulativas selecionada, um número comparativamente grande de bits será considerado do fluxo de bits a fim de preparar uma decodificação do próximo símbolo.

Da mesma forma, as entradas das tabelas de frequências cumulativas refletem as probabilidades dos diferentes símbolos e também refletem um número de bits necessário para decodificar uma sequência de símbolos. Ao variar a tabela de frequências cumulativas em dependência de um contexto, isto é, em dependência de símbolos previamente decodificados (ou valores espectrais), por exemplo, ao selecionar diferentes tabelas de frequências cumulativas em dependência do contexto, dependências estocásticas entre os diferentes símbolos podem ser exploradas, o

que permite uma codificação de taxa de bits particular eficiente dos símbolos subsequentes (ou adjacentes).

Para resumir o mencionado acima, a função "arith_decode()", que foi descrita com referência à Figura 5g, é chamada com a tabela de frequências cumulativas "arith_cf_m[pki][]", correspondente ao índice "pki" retornado pela função "„arith_get_pk()" para determinar o valor de plano de bits mais significativo m (que pode ser ajustado ao valor de símbolo representado pela variável "símbolo" de retorno).

6.7 MECANISMO DE ESCAPE

Enquanto o valor de plano de bits mais significativo decodificado m (que é retornado como um valor de símbolo pela função "arith_decode ()") é o símbolo de escape "ARITH_ESCAPE", um valor de plano de bits mais significativo adicional m é decodificado e a variável "lev" é incrementada em 1. Da mesma forma, uma informação é obtida sobre a significância numérica do valor de plano de bits mais significativo m, assim como o número de planos de bits menos significativos a serem decodificados.

Se um símbolo de escape "ARITH_ESCAPE" for decodificado, a variável de nível "lev" é aumentada em 1. Da mesma forma, o valor de estado que é inserido à função "arith_get_pk" também é modificado, de modo que um valor representado pelos bits mais altos (24 bits e mais) seja aumentado para as próximas iterações do algoritmo 312ba.

6.8 ATUALIZAÇÃO DE CONTEXTO, DE ACORDO COM A FIGURA 5H

Uma vez que o valor espectral é completamente

decodificado (isto é, todos os planos de bits menos significativos foram adicionados, as tabelas de contexto q e qs são atualizadas ao chamar a função "arith_update_context(a,i,lg))". A seguir, detalhes em relação à função "arith_update_context(a,i,lg)" serão descritos tendo como referência a Figura 5h, que apresenta a representação de código de pseudo-programa da dita função.

A função "arith_update_context()" recebe, como variáveis de entrada, o coeficiente espectral quantificado decodificado a, o índice i do valor espectral a ser decodificado (ou do valor espectral decodificado) e o número lg de valores espectrais (ou coeficientes) associados à estrutura de áudio atual.

Em uma etapa 580, o valor espectral (ou coeficiente) quantificado atualmente decodificado a é copiado na tabela de contexto ou arranjo de contexto q. Da mesma forma, a entrada q[1][i] da tabela de contexto q é ajustada a. Também, a variável "a0" é ajustada ao valor de "a".

Em uma etapa 582, o valor de nível q[1][i].l da tabela de contexto q é determinado. Pelo padrão, o valor de nível q[1][i].l da tabela de contexto q é ajustado a zero. Entretanto, se o valor absoluto do valor espectral atualmente codificado for maior que 4, o valor de nível q[1][i].l é incrementado. Com cada incremento, a variável "a" é trocada para a direita em um bit. O incremento do valor de nível q[1][i].l é repetido até que o valor absoluto da variável a0 seja menor ou igual a 4.

Em uma etapa 584, um valor de contexto de 2 bits q[1][i].c da tabela de contexto q é ajustado. O valor de contexto de 2 bits q[1][i].c é ajustado ao valor zero se o valor espectral

decodificado atualmente a for igual a zero. De outra forma, se o valor absoluto do valor espectral decodificado a for menor ou igual a 1, o valor de contexto de 2 bits q[1][i].c é ajustado a 1. De outra forma, se o valor absoluto do valor espectral atualmente decodificado a for menor ou igual a 3, o valor de contexto de 2 bits q[1][i].c é ajustado a 2. De outra forma, isto é, se o valor absoluto do valor espectral decodificado atualmente a for maior que 3, o valor de contexto de 2 bits q[1][i].c é ajustado a 3. Da mesma forma, o valor de contexto de 2 bits q[1][i].c é obtido por uma quantificação muito bruta do coeficiente espectral decodificado atualmente a.

Em uma etapa subsequente 586, que é somente realizada se o índice i do valor espectral decodificado atualmente for igual ao número lg de coeficientes (valores espectrais) na estrutura (isto é, se o último valor espectral da estrutura for decodificado) e o modo central for um modo central de domínio de previsão linear (que é indicado por “core_mode==1”), as entradas q[1][j].c são copiadas na tabela de contexto qs[k]. A cópia é realizada conforme apresentado no número de referência 586, de modo que o número lg de valores espectrais na estrutura atual seja considerado para a cópia das entradas q[1][j].c à tabela de contexto qs[k]. Além disso, a variável “previous_lg” toma o valor 1024.

De modo alternativo, entretanto, as entradas q[1][j].c da tabela de contexto q são copiadas na tabela de contexto qs[j] se o índice i do coeficiente espectral decodificado atualmente atingir o valor de lg e o modo central for um domínio de frequência modo central (indicado por “core_mode==0”).

Nesse caso, a variável "previous_lg" é ajustada ao mínimo entre o valor de 1024 e o número lg de valores espectrais na estrutura.

6. RESUMO DO PROCESSO DE DECODIFICAÇÃO

A seguir, o processo de decodificação será brevemente resumido. Para detalhes, é feita referência à discussão acima e também às Figuras 3, 4 e 5a a 5i.

Os coeficientes quantificados espectrais a são codificados silenciosamente e transmitidos, começando do menor coeficiente de frequência e progredindo ao maior coeficiente de frequência.

Os coeficientes da codificação de áudio avançada (AAC) são armazenados no arranjo "x_ac_quant[g][win][sfb][bin]", e a ordem de transmissão das senhas de codificação silenciosa é de modo que quando eles forem decodificados na ordem recebida e armazenada no arranjo, a caixa seja o índice de incrementação mais rápida e g seja o índice de incrementação mais lenta. A caixa de índice designa caixas de frequência. O índice "sfb" designa faixas de fator de escala. O índice "win" designa janelas. O índice "g" designa estruturas de áudio.

Os coeficientes da excitação codificada de transformação são armazenados diretamente em um arranjo "x_tcx_invquant[win][bin]", e a ordem da transmissão das senhas de codificação silenciosa é de modo que quando eles forem decodificados na ordem recebida e armazenados no arranjo, a "caixa" seja o índice de incrementação mais rápida e "win" é o índice de incrementação mais lenta.

Primeiro, é feito um mapeamento entre o contexto

anterior salvo armazenado na tabela ou arranjo de contexto “qs” e o contexto da estrutura atual q (armazenado na tabela ou arranjo de contexto q). O contexto anterior “qs” é armazenado em 2 bits por linha de frequência (ou por caixa de frequência).

O mapeamento entre o contexto anterior salvo armazenado na tabela de contexto “qs” e o contexto da estrutura atual armazenado na tabela de contexto “q” é realizado utilizando a função “arith_map_context()”, uma representação de código de pseudo-programa que é apresentada na Figura 5a.

O decodificador silencioso produz coeficientes quantificados espectrais assinalados “a”.

Primeiro, o estado do contexto é calculado com base nos coeficientes espectrais previamente decodificados que circundam os coeficientes quantificados espectrais a decodificar. O estado do contexto s corresponde aos 24 primeiros bits do valor retornado pela função “arith_get_context()”. Os bits acima do 24º bit do valor retornado correspondem ao nível de plano de bits previsto lev0. A variável „lev” é inicializada a lev0. Uma representação de código de pseudo-programa da função “arith_get_context” é apresentada nas Figuras 5b e 5c.

Uma vez que o estado s e o nível previsto „lev0” são conhecidos, o plano sensato de dois bits mais significativos m é decodificado utilizando a função “arith_decode()”, alimentado com a tabela de frequências cumulativas adequada correspondente ao modelo de probabilidade correspondente ao estado de contexto.

É feita correspondência pela função “arith_get_pk()”.

A representação de código de pseudo-programa da

função “arith_get_pk()” é apresentada na Figura 5e.

Um código de pseudo-programa de outra função “get_pk” que pode tomar o lugar da função “arith_get_pk()” é apresentado na Figura 5f. Um código de pseudo-programa de outra função “get_pk”, que pode tomar o lugar da função “arith_get_pk()” é apresentado na Figura 5d.

O valor m é decodificado utilizando a função “arith_decode()” chamada com a tabela de frequências cumulativas, “arith_cf_m[pki][], onde „pki” corresponde ao índice retornado pela função “arith_get_pk()” (ou, de modo alternativo, pela função “get_pk()”).

O codificador aritmético é uma implementação inteira utilizando o método de gerador de indicação com escalonamento (vide, por exemplo, K. Sayood “Introduction to Data Compression” third edition, 2006, Elsevier Inc.). O código pseudo-C apresentado na Figura 5g descreve o algoritmo utilizado.

Quando o valor decodificado m for o símbolo de escape, “ARITH_ESCAPE”, outro valor m é decodificado e a variável „lev” é incrementada em 1. Uma vez que o valor m não é o símbolo de escape, “ARITH_ESCAPE”, os planos de bits restantes são então decodificados a partir do nível mais significativo ao menos significativo, ao chamar „lev” vezes a função “arith_decode()” com a tabela de frequências cumulativas “arith_cf_r[]”. A dita tabela de frequências cumulativas “arith_cf_r[] pode, por exemplo, descrever uma distribuição de probabilidade constante.

Os planos de bits decodificados r permitem a refinação do valor previamente decodificado m da seguinte maneira:

```
a = m;
```

```
para (i=0; i<lev;i++) {
r = arith_decode (arith_cf_r,2);
a = (a<<1) | (r&1);
}
```

Uma vez que o coeficiente quantificado espectral a é completamente decodificado, as tabelas de contexto q ou o contexto armazenado qs são atualizados pela função "arith_update_context()", para os próximos coeficientes quantificados espectrais a decodificar.

A representação de código de pseudo-programa da função "arith_update_context()" é apresentada na Figura 5h.

Além disso, uma legenda de definições é apresentada na Figura 5i.

7. TABELAS DE MAPEAMENTO

Em uma realização, de acordo com a invenção, as tabelas particularmente vantajosas "ari_s_hash" e "ari_gs_hash" e "ari_cf_m" são utilizadas para a execução da função "get_pk", que foi discutida com referência à Figura 5d ou para a execução da função "arith_get_pk", que foi discutida com referência à Figura 5e ou para a execução da função "get_pk", que foi discutida com referência a 5f e para a execução da função "arith_decode" que foi discutida com referência à Figura 5g.

7.1. TABELA "ARI_S_HASH[387]", DE ACORDO COM A FIGURA 17

Um conteúdo de uma implementação particularmente vantajosa da tabela "ari_s_hash", que é utilizada pela função "get_pk" que foi descrita com referência à Figura 5d, é apresentado na tabela da Figura 17. Deve ser observado que a

tabela da Figura 17 lista as 387 entradas da tabela "ari_s_hash[387]". Também deve ser observado que a representação de tabela da Figura 17 apresenta os elementos na ordem dos índices de elemento, de modo que o primeiro valor "0x00000200" corresponda a uma entrada da tabela "ari_s_hash[0]" tendo índice de elemento (ou índice de tabela) 0, de modo que o último valor "0x03D0713D" corresponda a uma entrada da tabela "ari_s_hash[386]" tendo índice de elemento ou índice de tabela 386. Deve ser adicionalmente observado aqui que "0x" indica que as entradas de tabela da tabela "ari_s_hash" são representadas em um formato hexadecimal. Além disso, as entradas de tabela da tabela "ari_s_hash", de acordo com a Figura 17, são dispostas em ordem numérica a fim de permitir a execução da primeira avaliação de tabela 540 da função "get_pk".

Deve ser ainda observado que os 24 bits mais significativos das entradas de tabela da tabela "ari_s_hash" representam valores de estado, enquanto os 8 bits menos significativos representam valores índice da regra de mapeamento pki.

Assim, as entradas da tabela "ari_s_hash" descrevem um mapeamento de "atingimento direto" de um valor de estado em um valor índice de regra de mapeamento "pki".

7.2 TABELA "ARI GS HASH", DE ACORDO COM A FIGURA 18

Um conteúdo de uma realização particularmente vantajosa da tabela "ari_gs_hash" é apresentado na tabela da Figura 18. Deve ser observado aqui que a tabela 18 lista as entradas da tabela "ari_gs_hash". As ditas entradas são mencionadas por um índice de entrada de tipo inteiro

unidimensional (também designado como “índice de elemento” ou “índice de arranjo” ou “índice de tabela”), que é, por exemplo, designado com “i”. Deve ser observado que a tabela “ari_gs_hash” que compreende um total de 225 entradas, é bem adequada para uso pela segunda avaliação de tabela 544 da função “get_pk” descrita na Figura 5d.

Deve ser observado que as entradas da tabela “ari_gs_hash” são listadas em uma ordem crescente do índice de tabela i para o índice de tabela valores i entre zero e 224. O termo “0x” indica que as entradas de tabela são descritas em um formato hexadecimal. Da mesma forma, a primeira entrada da tabela “0X00000401” corresponde à entrada da tabela “ari_gs_hash[0]” tendo índice de tabela 0 e a última entrada da tabela “0Xffffff3f” corresponde à entrada da tabela “ari_gs_hash[224]” tendo índice de tabela 224.

Também deve ser observado que as entradas de tabela são ordenadas em uma maneira numericamente crescente, de modo que as entradas de tabela sejam bem adequadas para a segunda avaliação de tabela 544 da função “get_pk”. Os 24 bits mais significativos das entradas de tabela da tabela “ari_gs_hash” descrevem limites entre as variações de valores de estado, e os 8 bits menos significativos das entradas descrevem valores índice da regra de mapeamento “pki” associados às variações de valores de estado definidas pelos 24 bits mais significativos.

7.3 TABELA “ARI_CF_M”, DE ACORDO COM A FIGURA 19

A Figura 19 apresenta um conjunto de 64 tabelas de frequências cumulativas “ari_cf_m[pki][9]”, uma das quais é selecionada por um codificador de áudio 100, 700, ou um

decodificador de áudio 200, 800, por exemplo, para a execução da função “arith_decode”, isto é, para a decodificação do valor de plano de bits mais significativo. A selecionada das 64 tabelas de frequências cumulativas apresentada na Figura 19 toma a função da tabela “cum_freq[]” na execução da função “arith_decode()”.

Como pode ser visto na Figura 19, cada linha representa uma tabela de frequências cumulativas tendo 9 entradas. Por exemplo, uma primeira linha 1910 representa as 9 entradas de uma tabela de frequências cumulativas para “pki=0”. Uma segunda linha 1912 representa as 9 entradas de uma tabela de frequências cumulativas para “pki=1”. Por fim, uma 64ª linha 1964 representa as 9 entradas de uma tabela de frequências cumulativas para “pki=63”. Assim, a Figura 19 representa efetivamente 64 diferentes tabelas de frequências cumulativas para “pki=0” a um “pki=63”, em que cada uma das 64 tabelas de frequências cumulativas é representada por uma única linha e em que cada uma das ditas tabelas de frequências cumulativas compreende 9 entradas.

Dentro de uma linha (por exemplo, uma linha 1910 ou uma linha 1912 ou uma linha 1964), um valor mais à esquerda descreve uma primeira entrada de uma tabela de frequências cumulativas e um valor mais à direita descreve a última entrada de uma tabela de frequências cumulativas.

Da mesma forma, cada linha 1910, 1912, 1964 da representação de tabela da Figura 19 representa as entradas de uma tabela de frequências cumulativas para uso pela função “arith_decode”, de acordo com a Figura 5g. A variável de entrada “cum_freq[]” da função “arith_decode” descreve qual das 64 tabelas de frequências cumulativas (representadas por linhas individuais

de 9 entradas) da tabela “ari_cf_m” deve ser utilizada para a decodificação dos coeficientes espectrais atuais.

7.4 TABELA “ARI S HASH”, DE ACORDO COM A FIGURA 20

A Figura 20 apresenta uma alternativa para a tabela “ari_s_hash”, que pode ser utilizada em combinação à função alternativa “arith_get_pk()” ou “get_pk()”, de acordo com a Figura 5e ou 5f.

A tabela “ari_s_hash”, de acordo com a Figura 20, compreende 386 entradas, que são listadas na Figura 20 em uma ordem crescente do índice de tabela. Assim, o primeiro valor da tabela “0x0090D52E” corresponde à entrada da tabela “ari_s_hash[0]” tendo índice de tabela 0 e a última entrada da tabela “0x03D0513C” corresponde à entrada da tabela “ari_s_hash[386]” tendo índice de tabela 386.

O “0x” indica que as entradas de tabela são representadas em uma forma hexadecimal. Os 24 bits mais significativos das entradas da tabela “ari_s_hash” descrevem estados significativos e os 8 bits menos significativos das entradas da tabela “ari_s_hash” descrevem valores índice da regra de mapeamento.

Da mesma forma, as entradas da tabela “ari_s_hash” descrevem um mapeamento de estados significativos em valores índice da regra de mapeamento “pki”.

8. AVALIAÇÃO DE DESEMPENHO E VANTAGENS

As realizações, de acordo com a invenção, usam funções (ou algoritmos) atualizadas e um conjunto atualizado de tabelas, conforme discutido acima, a fim de obter uma troca

melhorada entre a complexidade da computação, exigências de memória e eficiência de codificação.

Falando de modo geral, as realizações, de acordo com a invenção, criam uma codificação silenciosa espectral melhorada.

A presente descrição descreve realizações para CE em codificação silenciosa espectral melhorada de coeficientes espectrais. O esquema proposto tem base no esquema de codificação com base no contexto “original”, conforme descrito no projeto de trabalho 4 do padrão de projeto da USAC, mas reduz significativamente as exigências de memória (RAM, ROM), enquanto mantém um desempenho de codificação silenciosa. Uma transcodificação sem perda de WD3 (isto é, da saída de um codificador de áudio que provê um fluxo de bits, de acordo com o projeto de trabalho 3 do padrão de projeto da USAC) provou ser possível. O esquema aqui descrito é, em geral, escalonável, permitindo compensações adicionais alternativas entre exigências de memória e desempenho de codificação. As realizações, de acordo com a invenção, visam a substituição do esquema de codificação silenciosa espectral conforme utilizado no projeto de trabalho 4 do padrão de projeto da USAC.

O esquema de codificação aritmética aqui descrito tem base no esquema como no modelo de referência 0 (RM0) ou no projeto de trabalho 4 (WD4) do padrão de projeto da USAC. Os coeficientes espectrais anteriores na frequência ou no modelo de tempo de um contexto. Esse contexto é utilizado para a seleção de tabelas de frequências cumulativas para o codificador aritmético (codificador ou decodificador). Comparado à realização, de acordo

com WD4, a modelagem do contexto é adicionalmente melhorada e as tabelas que possuem as probabilidades de símbolo foram reclassificadas. O número de diferentes modelos de probabilidades foi aumentado de 32 para 64.

As realizações, de acordo com a invenção, reduzem os tamanhos de tabela (demanda ROM de dados) para 900 palavras de extensão de 32 bits ou 3600 bytes. Ao contrário, as realizações, de acordo com WD4, do padrão de projeto da USAC requerem 16894,5 palavras ou 76578 bytes. A demanda RAM estática é reduzida, em algumas realizações, de acordo com a invenção, de 666 palavras (2664 bytes) para 72 (288 bytes) por canal de codificador central. Ao mesmo tempo, preserva completamente o desempenho de codificação e pode até atingir um ganho de aproximadamente 1,04% a 1,39%, comparado à taxa de dados gerais sobre todos os 9 pontos de operação. Todos os fluxos de bits do projeto de trabalho 3 (WD3) podem ser transcodificados em uma maneira sem perda sem afetar as restrições do reservatório de bits.

O esquema proposto, de acordo com as realizações da invenção, é escalonável: compensações flexíveis da demanda de memória e desempenho de codificação são possíveis. Ao aumentar, os tamanhos de tabela ao ganho de codificação podem ser adicionalmente aumentados.

A seguir, uma breve discussão do conceito da codificação, de acordo com WD4 do padrão de projeto da USAC, é provida para facilitar o entendimento das vantagens do conceito aqui descrito. Em USAC WD4, um esquema de codificação aritmética com base no contexto é utilizado para a codificação silenciosa de coeficientes quantificados espectrais. Como contexto, os

coeficientes espectrais decodificados são utilizados, que são anteriores em frequência e tempo. De acordo com WD4, um número máximo de 16 coeficientes espectrais é utilizado como contexto, 12 dos quais são anteriores no tempo. Ambos, os coeficientes espectrais utilizados para o contexto e para serem decodificados, são agrupados como 4 tuplos (isto é, quatro coeficientes espectrais próximos em frequência, vide Figura 10a). O contexto é reduzido e mapeado em uma tabela de frequências cumulativas, que é então utilizada para decodificar os próximos 4 tuplos de coeficientes espectrais.

Para o esquema de codificação silenciosa WD4 completo, uma demanda de memória (ROM) de 16894,5 palavras (67578 bytes) é necessária. Adicionalmente, 666 palavras (2664 bytes) de ROM estática por canal de codificador central são necessárias para armazenar os estados para a próxima estrutura.

A representação de tabela da Figura 11a descreve as tabelas, conforme utilizadas no esquema de codificação aritmética WD4 de USAC.

Uma demanda de memória total de um decodificador WD4 de USAC completo é estimada para ser 37000 palavras (148000 bytes) para ROM de dados sem um código de programa e 10000 a 17000 palavras para RAM estática. Pode ser claramente visto que tabelas de codificador silencioso consomem aproximadamente 45% da demanda de ROM de dados totais. A maior tabela individual já consome 4096 palavras (16384 bytes).

Descobriu-se que ambos, o tamanho da combinação de todas as tabelas e as amplas tabelas individuais, excedem tamanhos de cache típicos, conforme providos pelos chips de ponto

fixo para dispositivos portáteis de orçamento baixo, que está em uma variação típica de 8 a 32 kBytes (por exemplo, ARM9e, TIC64xx etc.). Isso significa que o conjunto de tabelas pode não ser provavelmente armazenado na RAM de dados rápidos, o que permite um acesso aleatório rápido aos dados. Isso faz com que todo o processo de decodificação seja desacelerado.

A seguir, o novo esquema proposto será brevemente descrito.

Para superar os problemas mencionados acima, um esquema de codificação silenciosa melhorado é proposto para substituir o esquema como em WD4 do padrão de projeto da USAC. Como um esquema de codificação aritmética com base no contexto, ele tem base no esquema de WD4 do padrão de projeto da USAC, mas os aspectos de um esquema modificado para derivação de tabelas de frequências cumulativas do contexto. Além disso, a derivação de contexto e codificação de símbolo é realizada na granularidade de um único coeficiente espectral (oposto a 4 tuplos, como em WD4 do padrão de projeto da USAC). No total, 7 coeficientes espectrais são utilizados para o contexto (pelo menos em alguns casos). Pela redução no mapeamento, um dos 64 modelos de probabilidades no total ou tabelas de frequências cumulativas (em WD4: 32) é selecionado.

A Figura 10b apresenta uma representação gráfica de um contexto para o cálculo de estado, conforme utilizado no esquema proposto (em que um contexto utilizado para a região detecção zero não é apresentado na Figura 10b).

A seguir, será provida uma breve discussão em relação à redução da demanda de memória, que pode ser alcançada ao

utilizar o esquema de codificação proposto. O novo esquema proposto apresenta uma demanda de ROM total de 900 palavras (3600 Bytes) (vide a tabela da Figura 11b que descreve as tabelas conforme utilizadas no esquema de codificação proposto).

Comparada à demanda de ROM do esquema de codificação silenciosa em WD4 do padrão de projeto da USAC, a demanda de ROM é reduzida em 15994,5 palavras (64978 Bytes)(vide também a Figura 12a, essa figura apresenta uma representação gráfica da demanda de ROM do esquema de codificação silenciosa, conforme proposto e do esquema de codificação silenciosa em WD4 do padrão de projeto da USAC). Isso reduz a demanda de ROM geral de um decodificador de USAC completo de aproximadamente 37000 palavras para aproximadamente 21000 palavras ou em mais de 43% (vide Figura 12b, que apresenta a representação gráfica de uma demanda de ROM de dados de decodificador de USAC total, de acordo com WD4 do padrão de projeto da USAC, assim como de acordo com a presente proposta).

Além disso, a quantidade de derivação de contexto de informações necessárias na próxima estrutura (RAM estática) também é reduzida. De acordo com WD4, o conjunto completo de coeficientes (maximamente 1152) uma resolução de tipicamente 16 bits adicionais a um índice de grupo por 4 tuplos de 10 bits de resolução que precisa ser armazenado, que soma até 666 palavras (2664 Bytes) por canal de codificador central (decodificador WD4 de USAC completo: aproximadamente 10000 a 17000 palavras).

O novo esquema, que é utilizado nas realizações, de acordo com a invenção, reduz as informações persistentes a somente 2 bits por coeficiente espectral, que soma até 72 palavras

(288 Bytes) no total por canal de codificador central. TA demanda na memória estática pode ser reduzida em 594 palavras (2376 Bytes).

A seguir, alguns detalhes em relação a um possível aumento de eficiência de codificação serão descritos. A eficiência de codificação das realizações, de acordo com a nova proposta, foi comprada em relação à aos fluxos de bits de qualidade de referência, de acordo com WD3 do padrão de projeto da USAC. A comparação foi realizada por meio de um transcodificador, com base em um decodificador de software de referência. Para detalhes em relação à comparação da codificação silenciosa, de acordo com WD3 do padrão de projeto da USAC e o esquema de codificação proposto, é feita referência à Figura 9, que apresenta a representação esquemática de uma disposição de teste.

Embora a demanda de memória seja drasticamente reduzida em realizações de acordo com a invenção quando comparado a realizações de acordo com WD3 ou WD4 do padrão de projeto da USAC, a eficiência de codificação não é somente mantida, mas discretamente aumentada. A eficiência de codificação está na média aumentada em 1,04% a 1,39%. Para detalhes, é feita referência à tabela da Figura 13a, que apresenta a representação de tabela de taxas de bits médias produzida pelo codificador de USAC utilizando o projeto de trabalho codificador aritmético e um codificador de áudio (por exemplo, codificador de áudio USAC), de acordo com uma realização da invenção.

Pela medida do nível de preenchimento de recipiente de bits, foi apresentado que a codificação silenciosa proposta é capaz de transcodificar sem perda o fluxo de bits de

WD3 para cada ponto de operação. Para detalhes, é feita referência à tabela da Figura 13b que apresenta a representação de tabela de um controle de reservatório de bits para um codificador de áudio de acordo com o WD3 de USAC e um codificador de áudio de acordo com uma realização da presente invenção.

Detalhes sobre taxas de bits médias por modo de operação, taxas de bits mínimas, máximas e médias em uma base de estrutura e um melhor/pior desempenho de caso em uma base de estrutura podem ser encontrados nas tabelas das Figuras 14, 15, e 16, em que a tabela da Figura 14 apresenta uma representação de tabela de taxas de bits médias para um codificador de áudio de acordo com o WD3 de USAC e para um codificador de áudio de acordo com uma realização da presente invenção, em que a tabela da Figura 15 apresenta uma representação de tabela de taxas de bits mínimas, máximas e médias de um codificador de áudio USAC em uma base de estrutura, e em que a tabela da Figura 16 apresenta uma representação de tabela do melhor e do pior caso em uma base de estrutura.

Além disso, deve ser observado que as realizações de acordo com a presente invenção provêem uma boa capacidade de escalonação. Ao adaptar o tamanho da tabela, uma compensação entre exigências de memória, complexidade computacional e eficiência de codificação pode ser ajustada de acordo com as exigências.

9. SINTAXE DE FLUXO DE BITS

9.1. CARGAS ÚTEIS DO CODIFICADOR SILENCIOSO ESPECTRAL

A seguir, alguns detalhes sem relação a cargas úteis do codificador silencioso espectral serão descritos. Em

algumas realizações, há uma pluralidade de diferentes modos de codificação, como por exemplo, um denominado domínio de previsão linear, "modo de codificação" e um modo de codificação de "domínio de frequência". No modo de codificação de domínio de previsão linear, uma formação de ruído é realizada com base em uma análise de previsão linear do sinal de áudio e um sinal formado por ruído é codificado no domínio de frequência. No modo de domínio de frequência, uma formação de ruído é realizada com base em uma análise psicoacústica e uma versão formada por ruído do conteúdo de áudio é codificada no domínio de frequência.

Coeficientes espectrais de ambos, um sinal codificado de "domínio de previsão linear" e um sinal codificado de "domínio de frequência" são quantificados escalonares e, então, codificados silenciosamente por uma codificação aritmética de contexto adaptativamente dependente. Os coeficientes quantificados são transmitidos da menor frequência à maior frequência. Cada coeficiente quantificado individual é dividido em plano sensato de 2 bits mais significativos m e os planos de bits menos significativos restantes r. O valor m é codificado de acordo com a proximidade dos coeficientes. Os planos de bits menos significativos restantes r são codificados por entropia, sem considerar o contexto. Os valores m e r formam os símbolos do codificador aritmético.

Um procedimento de decodificação aritmética detalhado é aqui descrito.

9.2. ELEMENTOS DE SINTAXE

A seguir, a sintaxe do fluxo de bits de um fluxo de bits que carrega as informações espectrais codificadas

aritmeticamente serão descritas tendo como referência as Figuras 6a a 6h.

A Figura 6a apresenta uma representação de sintaxe do denominado Bloco de dados brutos USAC ("usac_raw_data_block()").

O bloco de dados brutos USAC compreende um ou mais elementos de canal único ("single_channel_element()") e/ou um ou mais elementos de par de canais ("channel_pair_element()").

Tendo como referência agora a Figura 6b, a sintaxe de um elemento de canal único é descrita. O elemento de canal único compreende uma corrente de canais de domínio de previsão linear ("lpd_channel_stream ()") ou uma corrente de canal de domínio de frequência ("fd_channel_stream ()") em dependência do modo central.

A Figura 6c apresenta uma representação de sintaxe de um elemento de par de canais. Um elemento de par de canais compreende informações de modo central ("core_mode0", "core_mode1"). Além disso, o elemento de par de canais pode compreender uma informação de configuração "ics_info()". Adicionalmente, dependendo da informação de modo central, o elemento de par de canais compreende uma corrente de canais de domínio de previsão linear ou uma corrente de canais de domínio de frequência associada com um primeiro dos canais, e o elemento de par de canais também compreende uma corrente de canais de domínio de previsão linear ou uma corrente de canal de domínio de frequência associada ao segundo dos canais.

As informações de configuração "ics_info()", uma representação de sintaxe que é apresentada na Figura 6d,

compreende uma pluralidade de diferentes itens de informações de configuração, que não são de relevância particular para a presente invenção.

Uma corrente de canais de domínio de frequência ("fd_channel_stream ()"), uma representação de sintaxe da qual é apresentada na Figura 6e, compreende uma informação de ganho ("global_gain") e uma informação de configuração ("ics_info ()"). Além disso, a corrente de canal de domínio de frequência compreende dados de fator de escala ("scale_factor_data ()"), que descrevem fatores de escala utilizados para o escalonamento de valores espectrais de diferentes faixas de fator de escala e que são aplicados, por exemplo, pelo escalonador 150 e pelo re-escalonador 240. A corrente de canais de domínio de frequência também compreende dados espectrais aritmeticamente codificados ("ac_spectral_data ()"), que representam os valores espectrais codificados aritmeticamente.

Os dados espectrais aritmeticamente codificados ("ac_spectral_data()"), uma representação de sintaxe dos quais é apresentada na Figura 6f, compreende uma sinalização de reajuste aritmético ideal ("arith_reset_flag"), que é utilizada para reconfigurar seletivamente o contexto, conforme descrito acima. Além disso, os dados espectrais aritmeticamente codificados compreendem uma pluralidade de blocos de dados aritméticos ("arith_data"), que carregam valores espectrais codificados aritmeticamente. A estrutura dos blocos de dados codificados aritmeticamente depende do número de faixas de frequência (representadas pela variável "num_bands") e também do estado da sinalização de reajuste aritmético, conforme será discutido a

seguir.

A estrutura do bloco de dados codificados aritmeticamente será descrita tendo como referência a Figura 6g, que apresenta uma representação de sintaxe dos ditos blocos de dados codificados aritmeticamente. A representação de dados dentro do bloco de dados codificados aritmeticamente depende do número lg de valores espectrais a ser codificado, o status da sinalização de reajuste aritmético e também do contexto, isto é, os valores espectrais previamente codificados.

O contexto para a codificação do conjunto de valores espectrais atual é determinado de acordo com o algoritmo de determinação de contexto apresentado no número de referência 660. Detalhes em relação ao algoritmo de determinação de contexto foram discutidos acima tendo como referência a Figura 5a. O bloco de dados codificados aritmeticamente compreende lg conjuntos de senhas, cada conjunto de senhas representando um valor espectral. Um conjunto de senhas compreende uma senha aritmética “acod_m [pki][m]” representando um valor de plano de bits mais significativo m do valor espectral utilizando entre 1 e 20 bits. Além disso, o conjunto de senhas compreende uma ou mais senhas “acod_r[r]” se o valor espectral precisar de mais planos de bits que o plano de bits mais significativo para uma representação correta. A senha “acod_r [r]” representa um plano de bits menos significativo utilizando entre 1 e 20 bits.

Se; entretanto, um ou mais planos de bits menos significativos forem necessários (além do plano de bits mais significativos) para uma representação adequada do valor espectral, isso é sinalizado ao utilizar uma ou mais senhas de

escape aritméticas (“ARITH_ESCAPE”). Assim, pode ser dito de modo geral que para um valor espectral, determina-se quantos planos de bits (o plano de bits mais significativo e, possivelmente, um ou mais planos de bits menos significativos adicionais) são necessários. Se um ou mais planos de bits menos significativos forem necessários, isso é sinalizado por uma ou mais senhas de escape aritméticas “acod_m [pki][ARITH_ESCAPE]”, que são codificadas de acordo com uma tabela de frequências cumulativas atualmente selecionada, um índice da tabela de frequências cumulativas da qual é dado pela variável pki. Além disso, o contexto é adaptado, como pode ser visto nos números de referência 664, 662, se uma ou mais senhas de escape aritméticas são incluídas no fluxo de bits. Depois da uma ou mais senhas de escape aritméticas, uma senha aritmética “acod_m [pki][m]” é incluída no fluxo de bits, conforme apresentado no número de referência 663, em que pki designa o índice de modelo de probabilidade atualmente válido (levando em consideração a adaptação de contexto causada pela inclusão das senhas de escape aritméticas), e em que m designa o valor de plano de bits mais significativo do valor espectral a ser codificado ou decodificado.

Conforme discutido acima, a presença de quaisquer planos de bits menos significativos resulta na presença de uma ou mais senhas “acod_r [r]”, cada uma das quais representa um bit do plano de bits menos significativo. A uma ou mais senhas “acod_r[r]” são codificadas de acordo com uma tabela de frequências cumulativas correspondente, que é constante e independente do contexto.

Além disso, deve ser observado que o contexto é

atualizado após a codificação de cada valor espectral, conforme apresentado no número de referência 668, de modo que o contexto seja tipicamente diferente para a codificação dos dois valores espectrais subsequentes.

A Figura 6h apresenta uma legenda de definições e elementos de ajuda que definem a sintaxe do bloco de dados codificados aritmeticamente.

Para resumir o mencionado acima, um formato de fluxo de bits foi descrito, que pode ser provido pelo codificador de áudio 100 e que pode ser avaliado pelo decodificador de áudio 200. O fluxo de bits dos valores espectrais codificados aritmeticamente é codificado de modo a ajustar o algoritmo de decodificação discutido acima.

Além disso, deve ser observado, de modo geral, que a codificação é a operação inversa da decodificação, de modo que se possa presumir geralmente que o codificador realiza uma pesquisa da tabela utilizando as tabelas discutidas acima, que é aproximadamente inversa à pesquisa da tabela realizada pelo decodificador. De modo geral, pode ser dito que um técnico no assunto que conhece o algoritmo de decodificação e/ou a sintaxe de fluxo de bits desejada será facilmente capaz de projetar um codificador aritmético, que provê os dados definidos na sintaxe de fluxo de bits e necessários pelo decodificador aritmético.

10. ALTERNATIVAS DE IMPLEMENTAÇÃO

Apesar de alguns aspectos terem sido descritos no contexto de um equipamento, é claro que esses aspectos também representam uma descrição do método correspondente, no qual um bloco ou dispositivo corresponde a uma etapa do método ou uma

característica de uma etapa do método. De maneira análoga, os aspectos descritos no contexto de uma etapa do método também representam uma descrição de um bloco ou item ou característica correspondente de um equipamento correspondente. Algumas ou todas as etapas do método podem ser executadas por (ou utilizando) um equipamento de hardware, como, por exemplo, um microprocessador, um computador programável ou um circuito elétrico. Em algumas realizações, alguma ou mais das etapas mais importantes do método podem ser executadas por esse equipamento.

O sinal de áudio codificado inventivo pode ser armazenado em um meio de armazenamento digital ou pode ser transmitido em um meio de transmissão, como um meio de transmissão sem fio ou um meio de transmissão por fio, como a Internet.

Dependendo de determinadas exigências de implementação, as realizações da invenção podem ser implementadas em hardware ou em software. A implementação pode ser realizada utilizando um meio de armazenamento digital, por exemplo, um disquete, um DVD, um Blu-Ray, um CD, uma ROM, uma PROM, uma EPROM, uma EEPROM ou uma memória FLASH, tendo sinais de controle legíveis eletronicamente armazenados nele, que cooperam (ou são capazes de cooperar) com um sistema de computador programável, de modo que o respectivo método seja realizado. Portanto, o meio de armazenamento digital pode ser legível por computador.

Algumas realizações, de acordo com a invenção, compreendem um carregador de dados tendo sinais de controle legíveis eletronicamente, que são capazes de cooperar com um sistema de computador programável, de modo que um dos métodos aqui descritos seja realizado.

De modo geral, as realizações da presente invenção podem ser implementadas como um produto de programa de computador com um código de programa, o código de programa sendo operado para realizar um dos métodos quando o produto de programa de computador executar em um computador. O código de programa pode, por exemplo, ser armazenado em um carregador legível pela máquina.

Outras realizações compreendem o programa de computador para realizar um dos métodos aqui descritos, armazenado em um carregador legível pela máquina.

Em outras palavras, uma realização do método inventivo é; portanto, um programa de computador tendo um código de programa para realizar um dos métodos aqui descritos, quando o programa de computador executar em um computador.

Uma realização adicional dos métodos inventivos é, portanto, um carregador de dados (ou um meio de armazenamento digital ou meio legível por computador) compreendendo, gravado em si, o programa de computador para realizar um dos métodos aqui descritos.

Uma realização adicional do método inventivo é, portanto, um fluxo de dados ou uma sequência de sinais que representam o programa de computador para realizar um dos métodos aqui descritos. O fluxo de dados ou a sequência de sinais pode, por exemplo, ser configurado para ser transferido por meio de uma conexão de comunicação de dados, por exemplo, por meio da Internet.

Uma realização adicional compreende um meio de processamento, por exemplo, um computador, ou um dispositivo de

lógica programável, configurado ou adaptado para realizar um dos métodos aqui descritos.

Uma realização adicional compreende um computador tendo instalado nele o programa de computador para realizar um dos métodos aqui descritos.

Em algumas realizações, um dispositivo de lógica programável (por exemplo, uma matriz de porta de campo programável) pode ser utilizado para realizar alguma ou todas as funcionalidades dos métodos aqui descritos. Em algumas realizações, uma matriz de porta de campo programável pode cooperar com um microprocessador a fim de realizar um dos métodos aqui descritos. De modo geral, os métodos são preferencialmente realizados por qualquer equipamento de hardware.

As realizações descritas acima são meramente ilustrativas para os princípios da presente invenção. É entendido que modificações e variações das disposições e detalhes aqui descritos serão aparentes aos outros técnicos no assunto. Pretende-se, portanto, limitar-se somente pelo escopo das reivindicações da patente iminentes e não pelos detalhes específicos apresentados a título de descrição e explicação das realizações aqui.

Embora o acima mencionado tenha sido particularmente apresentado e descrito com referência às realizações particulares acima, deve ser entendido pelos técnicos no assunto que diversas outras alterações nas formas e detalhes podem ser feitas sem se desviar de seu espírito e escopo. Deve ser entendido que diversas alterações podem ser feitas na adaptação de diferentes realizações sem se desviar do conceito mais amplo aqui

revelado e compreendido pelas reivindicações que seguem.

11. CONCLUSÃO

Para concluir, pode ser observado que as realizações, de acordo com a invenção, criam um esquema de codificação silenciosa espectral melhorada. As realizações, de acordo com a nova proposta, permitem a redução significativa da demanda de memória de 16894,5 palavras para 900 palavras (ROM) e de 666 palavras para 72 (RAM estática por canal de codificador central). Isso permite a redução da demanda de ROM de dados do sistema completo em aproximadamente 43% em uma realização. Simultaneamente, o desempenho de codificação não é só completamente mantido, mas uma média até melhorada. Provou-se que uma transcodificação sem perda de WD3 (ou de um fluxo de bits provido de acordo com WD3 do padrão de projeto da USAC) é possível. Da mesma forma, uma realização, de acordo com a invenção, é obtida ao adotar a decodificação silenciosa aqui descrita no próximo projeto de trabalho do padrão de projeto da USAC.

Para resumir, em uma realização, a nova codificação silenciosa proposta pode gerar as modificações no projeto de trabalho de USAC MPEG em relação à sintaxe do elemento de fluxo de bits “arith_data()”, conforme apresentado na Figura 6g, em relação às cargas úteis do codificador silencioso espectral, conforme escrito acima e conforme apresentado na Figura 5h, em relação à codificação silenciosa espectral, conforme descrito acima, em relação ao contexto para o cálculo de estado, conforme apresentado na Figura 4, em relação às definições, conforme apresentado na Figura 5i, em relação ao processo de

decodificação, conforme descrito com referência às Figuras 5a, 5b, 5c, 5e, 5g, 5h e em relação às tabelas, conforme apresentado nas Figuras 17, 18, 20 e com relação à função “get_pk”, conforme apresentado na Figura 5d. De modo alternativo; entretanto, a tabela “ari_s_hash”, de acordo com a Figura 20, pode ser utilizada em vez da tabela “ari_s_hash” da Figura 17, e a função “get_pk” da Figura 5f pode ser utilizada em vez da função “get_pk”, de acordo com a Figura 5d.

REIVINDICAÇÕES

1. DECODIFICADOR DE ÁUDIO (200;800) PARA PROVER UMA INFORMAÇÃO DE ÁUDIO DECODIFICADO (212;812) COM BASE EM UMA INFORMAÇÃO DE ÁUDIO CODIFICADA (210;810), o decodificador de áudio é caracterizado por compreender:

um decodificador aritmético (230;820) para prover uma pluralidade de valores espectrais decodificados (232;822) com base em uma representação aritmeticamente codificada (222;821) dos valores espectrais; e

um conversor de domínio de frequência para domínio de tempo (260;830) para prover uma representação de áudio de domínio de tempo (262;812) utilizando os valores espectrais decodificados (232;822), a fim de obter as informações de áudio decodificado (212;812);

em que o decodificador aritmético (230;820) é configurado para selecionar uma regra de mapeamento (297; cum_freq[]) que descreve um mapeamento de um valor de código (valor) em um código de símbolo (símbolo) em dependência de um estado de contexto (s); e

em que o decodificador aritmético (230;820) é configurado para determinar o estado de contexto atual (s) em dependência de uma pluralidade de valores espectrais previamente decodificados,

em que o decodificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais previamente decodificados, que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes, e para determinar ou modificar o estado de

contexto atual (s) em dependência de um resultado da detecção;

em que o conversor de domínio de frequência para domínio de tempo (260;830) é configurado transformador para realizar uma transformada de cosseno discreta modificada inversa (IMDCT) e um janelamento.

2. DECODIFICADOR DE ÁUDIO (200; 800), de acordo com a reivindicação 1, caracterizado por o decodificador aritmético ser configurado para determinar ou modificar o estado de contexto atual (s) independente dos valores espectrais previamente decodificados em resposta da detecção que a condição predeterminada é atendida.

3. DECODIFICADOR DE ÁUDIO (200; 800), de acordo com a reivindicação 1 ou 2, caracterizado por o decodificador aritmético ser configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados, que atenda, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes.

4. DECODIFICADOR DE ÁUDIO, de acordo com qualquer uma das reivindicações de 1 a 3, caracterizado por o decodificador aritmético (230) é configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados que, individualmente ou considerado juntamente, compreende uma magnitude que é menor que uma magnitude limite predeterminada e para determinar ou modificar o estado de contexto atual (s) em dependência de um resultado da detecção.

5. DECODIFICADOR DE ÁUDIO, de acordo com qualquer uma das reivindicações de 1 a 4, caracterizado por o decodificador

aritmético ser configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados, em que cada um dos valores espectrais previamente decodificados é um valor zero e para determinar ou modificar o estado de contexto (s) em dependência de um resultado da detecção.

6. DECODIFICADOR DE ÁUDIO, de acordo com qualquer uma das reivindicações de 1 a 4, caracterizado por o decodificador aritmético ser configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados, que compreende um valor de soma que é menor que um valor limite predeterminado e para determinar ou modificar o estado atual (s) em dependência de um resultado da detecção;

7. DECODIFICADOR DE ÁUDIO, de acordo com qualquer uma das reivindicações de 1 a 6, caracterizado por o decodificador aritmético ser configurado para detectar o estado de contexto atual (s) a um valor predeterminado em resposta à detecção que um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados atende, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes.

8. DECODIFICADOR DE ÁUDIO, de acordo com a reivindicação 7, caracterizado por o decodificador aritmético (230) ser configurado para omitir seletivamente um cálculo do estado de contexto (s) em dependência de valores numéricos de uma pluralidade de valores espectrais previamente decodificados em resposta à detecção que um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados atende, individualmente ou considerado juntamente, a uma condição

predeterminada em relação a suas magnitudes.

9. DECODIFICADOR DE ÁUDIO, de acordo com qualquer uma das reivindicações de 1 a 6, caracterizado por o decodificador aritmético ser configurado para ajustar o estado de contexto atual (s) para estar dentro de uma variação de valores que sinaliza a detecção de um grupo de uma pluralidade de valores espectrais adjacentes previamente decodificados que atende, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes, em resposta à detecção.

10. DECODIFICADOR DE ÁUDIO, de acordo com de acordo com qualquer uma das reivindicações de 1 a 9, caracterizado por o decodificador aritmético ser configurado para mapear um código de símbolo (símbolo; m) em um valor espectral decodificado (a).

11. DECODIFICADOR DE ÁUDIO, de acordo com de acordo com qualquer uma das reivindicações de 1 a 10, caracterizado por o decodificador aritmético ser configurado para avaliar valores espectrais previamente decodificados de uma primeira região de frequência de tempo e para detectar um grupo de uma pluralidade de valores espectrais que atende, individualmente ou considerado juntamente, à condição predeterminada em relação a suas magnitudes, e

em que o decodificador aritmético é configurado para obter um valor numérico que representa o estado de contexto (s) se a condição predeterminada não for atendida, em dependência de valores espectrais previamente decodificados de uma segunda região de frequência de tempo que é diferente da primeira região de frequência de tempo.

12. DECODIFICADOR DE ÁUDIO, de acordo com de acordo com qualquer uma das reivindicações de 1 a 11, caracterizado por o decodificador aritmético ser configurado para avaliar uma ou mais tabelas de dispersão (ari_s_hash, ari_gs_hash) para selecionar uma regra de mapeamento (ari_cf_m[pki][9]) em dependência do estado de contexto (s).

13. CODIFICADOR DE ÁUDIO (100;700) para prover uma informação de áudio codificada (112;712) com base em uma informação de áudio de entrada (110;710), o codificador de áudio é caracterizado por compreender:

um conversor de domínio de tempo para domínio de frequência de compactação de energia (130;720) para prover uma representação de áudio de domínio de frequência (132;722) com base em uma representação de domínio de tempo (110;710) da informação de áudio de entrada, de modo que a representação de áudio de domínio de frequência (132;722) compreenda um conjunto de valores espectrais; e

um codificador aritmético (170;730) configurado para codificar um valor espectral (a) ou uma versão sua pré-processada, utilizando uma senha de extensão variável (acod_m, acod_r), em que o codificador aritmético (170) é configurado para mapear um valor espectral (a) ou um valor (m) de um plano de bits mais significativo de um valor espectral (a) em um valor de código (acod_m),

em que o codificador aritmético é configurado para selecionar uma regra de mapeamento que escreve um mapeamento de um valor espectral ou de um plano de bits mais significativo de um valor espectral, em um valor de código, em dependência de um

estado de contexto (s); e

em que o codificador aritmético é configurado para determinar o estado de contexto atual (s) em dependência de uma pluralidade de valores espectrais previamente codificados,

em que o codificador aritmético é configurado para detectar um grupo de uma pluralidade de valores espectrais previamente codificados, que atende, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes, e para determinar ou modificar o estado de contexto atual (s) em dependência de um resultado da detecção;

em que o conversor de compactação de energia de domínio de tempo para domínio de frequência é configurado para janelar a informação de áudio de entrada (110, 110a) (ou uma estrutura sua) utilizando uma janela de transformação e para realizar uma transformada do cosseno discreta modificada da informação de áudio de entrada janelada.

14. CODIFICADOR DE ÁUDIO (100; 700), de acordo com a reivindicação 13, caracterizado por o codificador aritmético ser configurado para determinar ou modificar o estado de contexto atual (s) independente dos valores espectrais previamente codificados em resposta à detecção que a condição predeterminada é atendida.

15. CODIFICADOR DE ÁUDIO (100; 700), de acordo com qualquer uma das reivindicações 13 ou 14, caracterizado por o codificador aritmético ser configurado para detectar um grupo de uma pluralidade de valores espectrais adjacentes previamente codificados, que atende, individualmente ou considerado juntamente, a uma condição predeterminada em relação a suas

magnitudes.

16. MÉTODO PARA PROVER UMA INFORMAÇÃO DE ÁUDIO DECODIFICADO COM BASE EM UMA INFORMAÇÃO DE ÁUDIO CODIFICADO, o método é caracterizado por compreender:

a provisão de uma pluralidade de valores espectrais decodificados com base em uma representação aritmeticamente codificada dos valores espectrais; e

a provisão de uma representação de áudio de domínio de tempo utilizando os valores espectrais decodificados, a fim de obter a informações de áudio decodificado;

em que o conversor de domínio de frequência para domínio de tempo (260;830) é configurado transformador para realizar uma transformada de cosseno discreta modificada inversa (IMDCT) e um janelamento;

em que a provisão da pluralidade de valores espectrais decodificados compreende a seleção de uma regra de mapeamento que descreve um mapeamento de um valor de código (acod_m; valor) que representa um valor espectral ou um plano de bits mais significativo de um valor espectral, em uma forma codificada em um código de símbolo (símbolo) que representa um valor espectral ou um plano de bits mais significativo de um valor espectral, em uma forma decodificada, em dependência de um estado de contexto; e

em que o estado de contexto atual é determinado em dependência de uma pluralidade de valores espectrais previamente decodificados,

em que um grupo de uma pluralidade de valores espectrais previamente decodificados, que atende, individualmente

ou considerado juntamente, a uma condição predeterminada em relação a suas magnitudes é detectado, e em que o estado de contexto atual é determinado ou modificado em dependência de um resultado da detecção;

17. MÉTODO PARA PROVER UMA INFORMAÇÃO DE ÁUDIO CODIFICADO COM BASE EM UMA INFORMAÇÃO DE ÁUDIO DE ENTRADA, o método é caracterizado por compreender:

a provisão de uma representação de áudio de domínio de frequência com base em uma representação de domínio de tempo da informação de áudio de entrada utilizando uma conversão de domínio de tempo para domínio de frequência de compactação de energia, de modo que a representação de áudio de domínio de frequência compreenda um conjunto de valores espectrais;

em que a provisão de uma representação de áudio de domínio de frequência compreende o janelamento da informação de áudio de entrada (ou uma estrutura sua) utilizando uma janela de transformação e para realizar uma transformada do cosseno discreta modificada da informação de áudio de entrada janelada; e

a codificação aritmética de um valor espectral, ou uma versão sua pré-processada, utilizando uma senha de extensão variável, em que um valor espectral ou um valor de um plano de bits mais significativo de um valor espectral é mapeado em um valor de código;

em que uma regra de mapeamento que descreve um mapeamento de um valor espectral ou de um plano de bits mais significativo de um valor espectral, em um valor de código é selecionada em dependência de um estado de contexto; e

em que um estado de contexto atual é determinado

em dependência de uma pluralidade de valores espectrais adjacentes previamente codificados; e

em que um grupo de uma pluralidade de valores espectrais previamente decodificados, que atende, individual ou juntamente, a uma condição predeterminada em relação a suas magnitudes, é detectado e o estado de contexto atual é determinado ou modificado em dependência de um resultado da detecção;

18. Meio não transitório legível por máquina caracterizado por compreender instruções que, quando executadas, realizam r o método das reivindicações 16 ou 17.

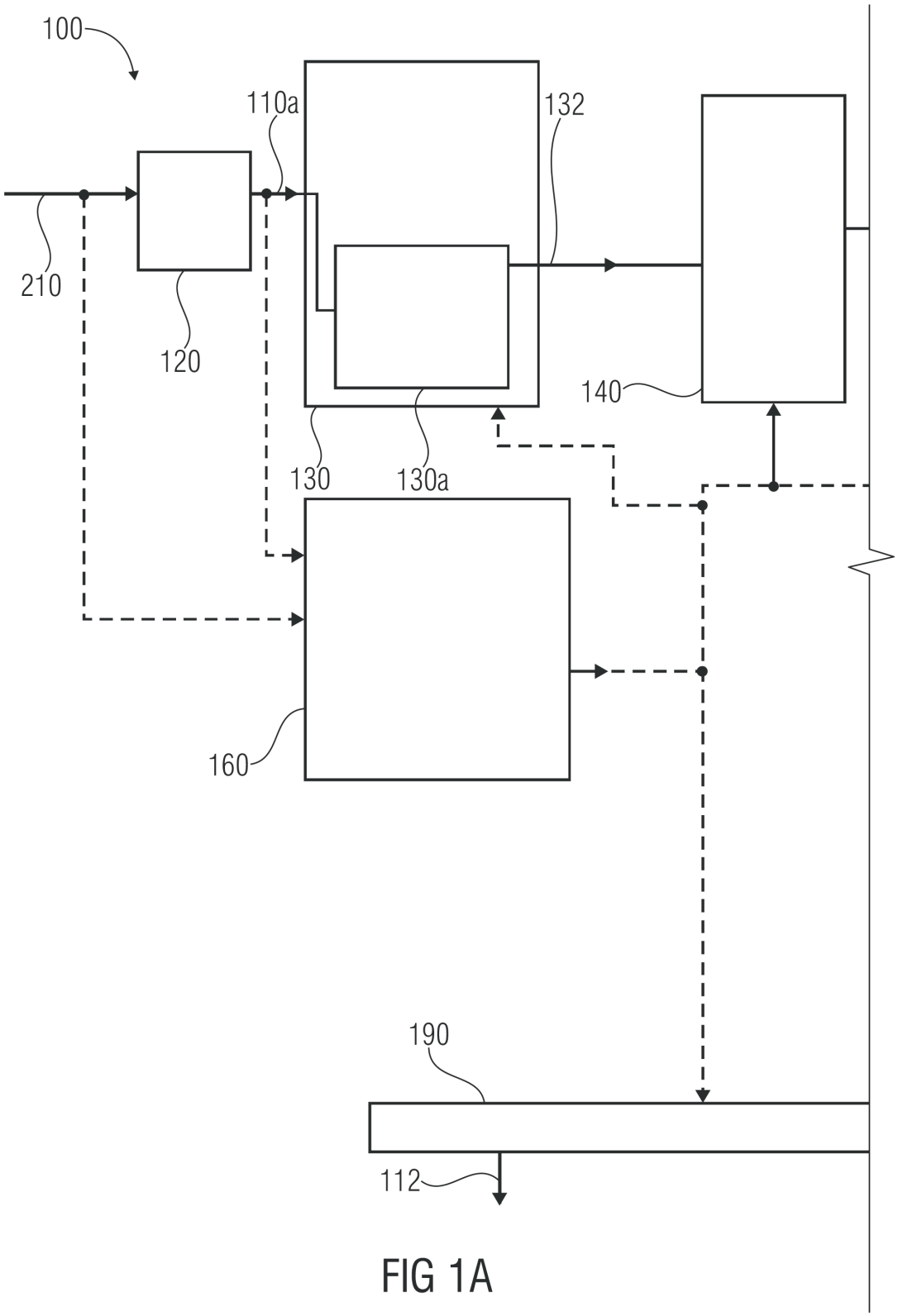


FIG 1A

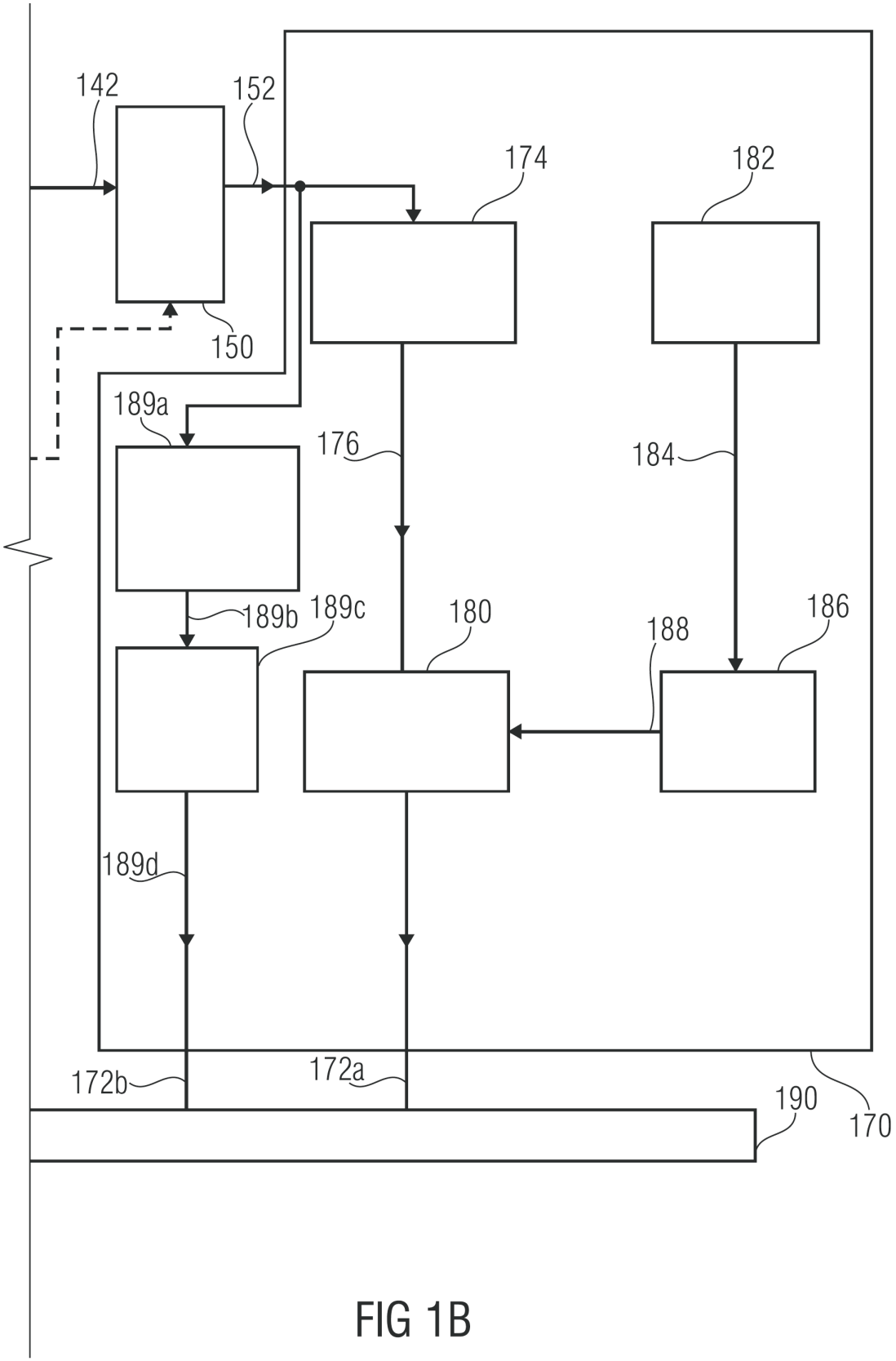

142
152
174
182
150
189a
176
184
189b
189c
180
188
186
189d
172b
172a
190
170

FIG 1B

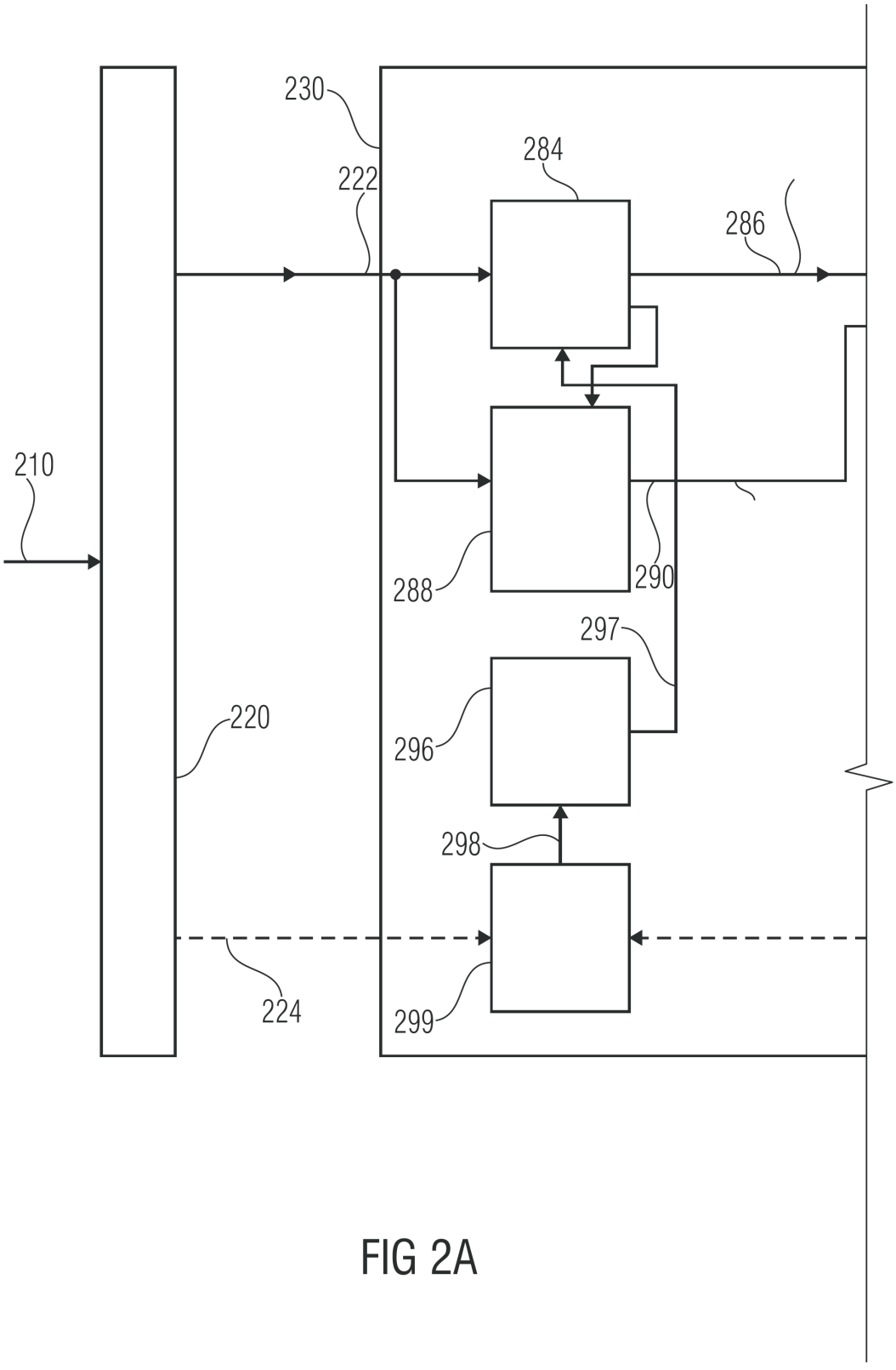


FIG 2A

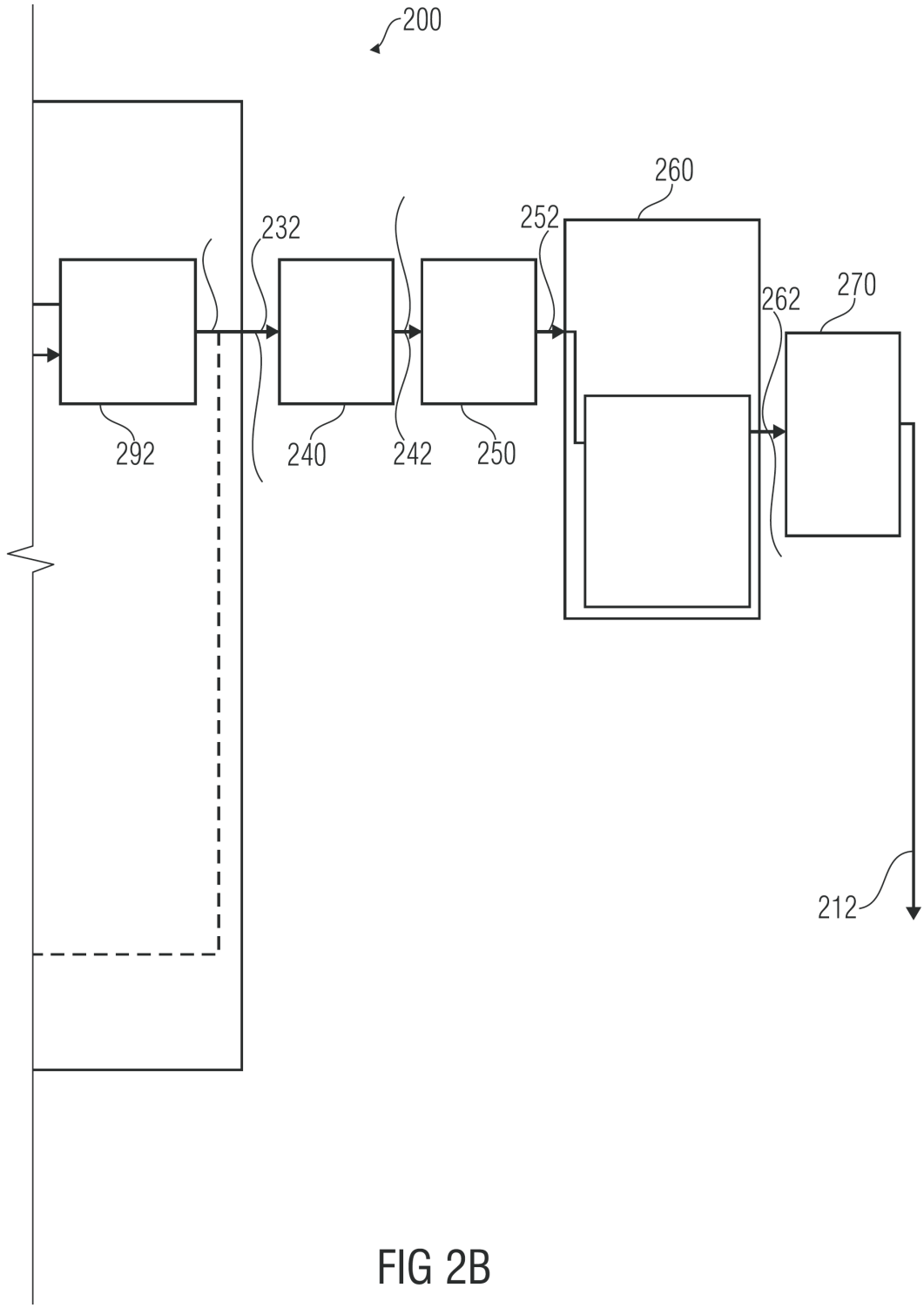


FIG 2B

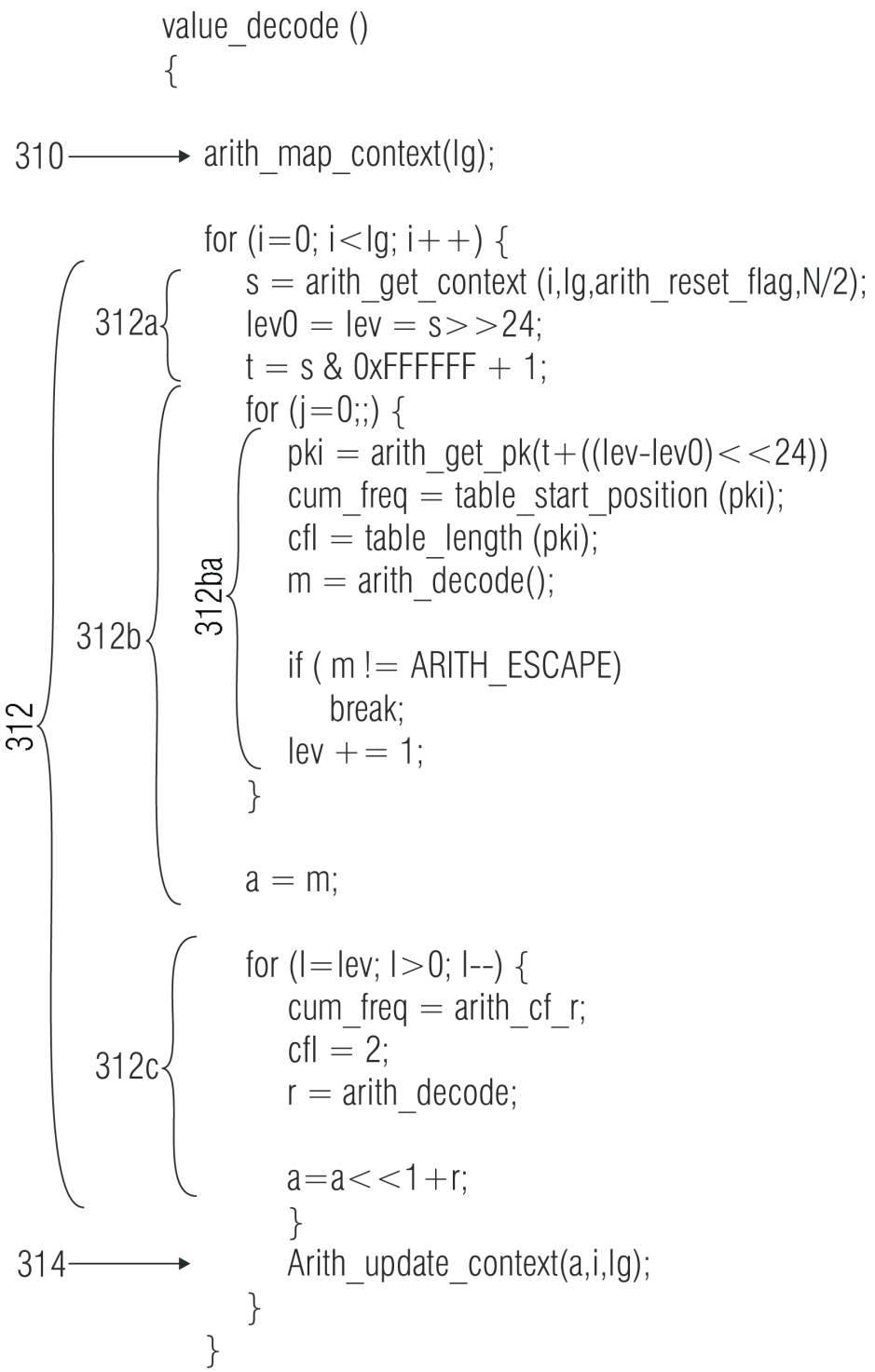

value_decode ()
{
310
arith_map_context(lg);
for (i=0; i<lg; i++) {
312
312a
s = arith_get_context (i,lg,arith_reset_flag,N/2);
lev0 = lev = s>>24;
t = s & 0xFFFFFF + 1;
312b
for (j=0;;) {
312ba
pki = arith_get_pk(t+((lev-lev0)<<24))
cum_freq = table_start_position (pki);
cfl = table_length (pki);
m = arith_decode();
if (m != ARITH_ESCAPE)
break;
lev += 1;
}
a = m;
312c
for (l=lev; l>0; l--) {
cum_freq = arith_cf_r;
cfl = 2;
r = arith_decode;
a=a<<1+r;
}
314
Arith_update_context(a,i,lg);
}
}

FIG 3

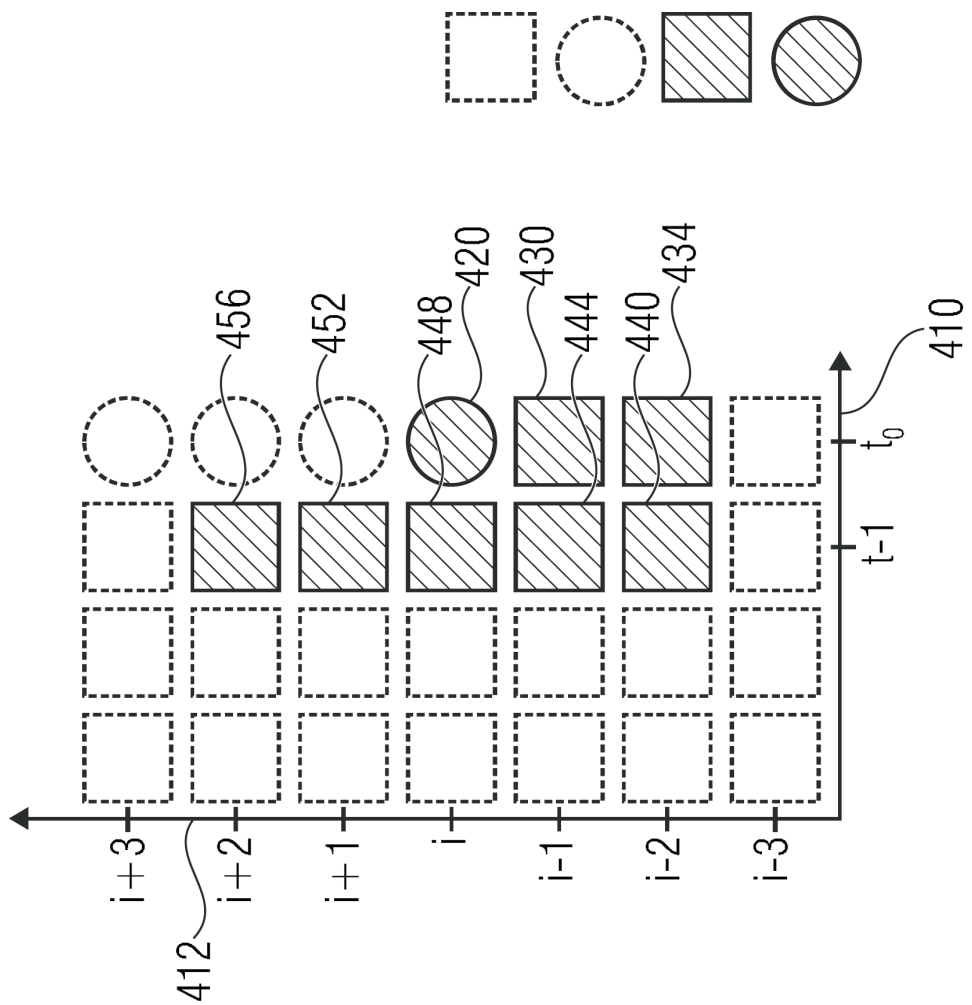


FIG 4

```
/*Input variables*/
lg /*number of sepctral coefficients to decode in the frame*/
previous_lg /Previous number of spectral lines of the previous frame*/

arith_map_context()
{
        v=w=0

        ratio= ((float)previous_lg)/((float)lg);
        for(j=0; j<lg; j++){
                k = (int) ((float)) ((j)*ratio);
                q[0][v++].c = qs[w+k];
        }

        previous_lg=lg;
}
```

FIG 5A

```
/*Input variables*/
i  /*Index of the spectral value to decode in the vector*/
lg /*Number of expected quantized coefficients*/
N  /*Number of lines of the transformation*/
ari_reset_flag  /*flag indicating whether the context should be reset*/
/*Output value*/
t  /*Concatenated state index s and predicted bit-plane level lev0*/

arith_get_context()
{
    int a0,c0,c1,c2,c3,c4,c5,c6,lev0,region;

    if(arith_reset_flag && i==0)
        return(0);
    if((!arith_reset_flag) && (i!=0)){
        int k;
        int lim_min,lim_max;
        int flag=1;
        lim_max = i+6;
        if((i+lim_max)>lg-1)
            lim_max=lg-1-i;
        lim_min = -5;
        if((i+lim_min)<0)
            lim_min=-i;
        for(k=lim_min;k<0;k++)
            if(q[0][k].c!=0 && q[1][k].c!=0)
                flag=0; break;
        for(;k<=lim_max;k++)
            if(q[0][k].c!=0)
                flag=0; break;
        if(flag)
            return(1);
    }
    if(i>0){
        a0=q[1][i-1];
        c0=ABS(a0);
        lev0=0;
        while((a0<-4)||(a0>=4)){
            a0>>=1;
            lev0++;
            c0=4+lev0;
        }
        if(c0>7)
            c0=7;
        if(lev0>3)
            lev0=3;

        if(arith_reset_flag && i==1)
            return((2+c0)|(lev0<<24));
        c4=q[0][i-1].c;
    }
```

510, 512, 512a, 512b, 512c, 512d, 514, 514a, 514b, 514c, 514d, 514e

FIG 5B

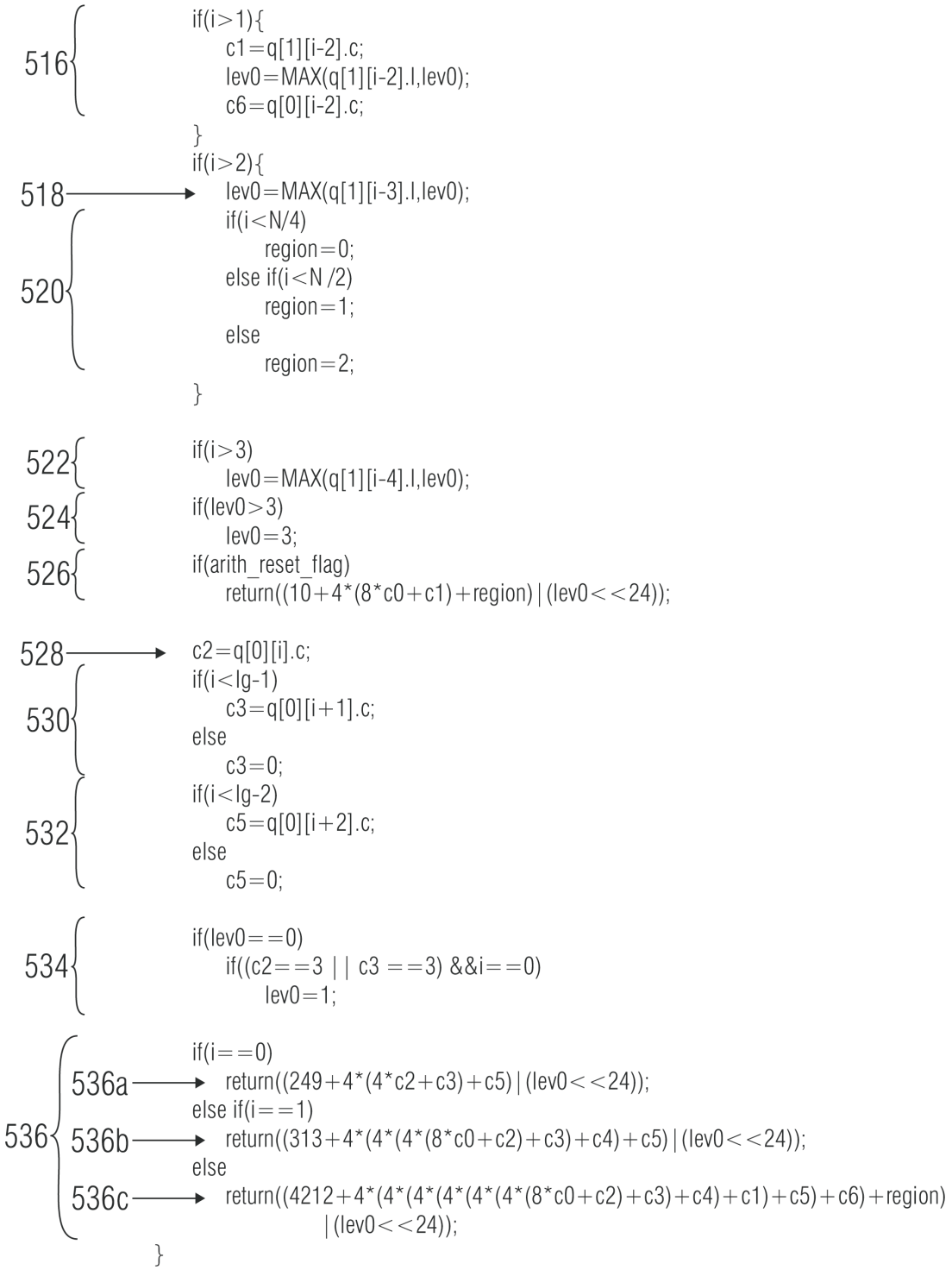

516
if(i>1){
c1=q[1][i-2].c;
lev0=MAX(q[1][i-2].l,lev0);
c6=q[0][i-2].c;
}
if(i>2){
518
lev0=MAX(q[1][i-3].l,lev0);
520
if(i<N/4)
region=0;
else if(i<N/2)
region=1;
else
region=2;
}
522
if(i>3)
lev0=MAX(q[1][i-4].l,lev0);
524
if(lev0>3)
lev0=3;
526
if(arith_reset_flag)
return((10+4*(8*c0+c1)+region)|(lev0<<24));
528
c2=q[0][i].c;
530
if(i<lg-1)
c3=q[0][i+1].c;
else
c3=0;
532
if(i<lg-2)
c5=q[0][i+2].c;
else
c5=0;
534
if(lev0==0)
if((c2==3 || c3 ==3) &&i==0)
lev0=1;
536
if(i==0)
536a
return((249+4*(4*c2+c3)+c5)|(lev0<<24));
else if(i==1)
536b
return((313+4*(4*(4*(8*c0+c2)+c3)+c4)+c5)|(lev0<<24));
else
536c
return((4212+4*(4*(4*(4*(4*(8*c0+c2)+c3)+c4)+c1)+c5)+c6)+region)
|(lev0<<24));
}

FIG 5C

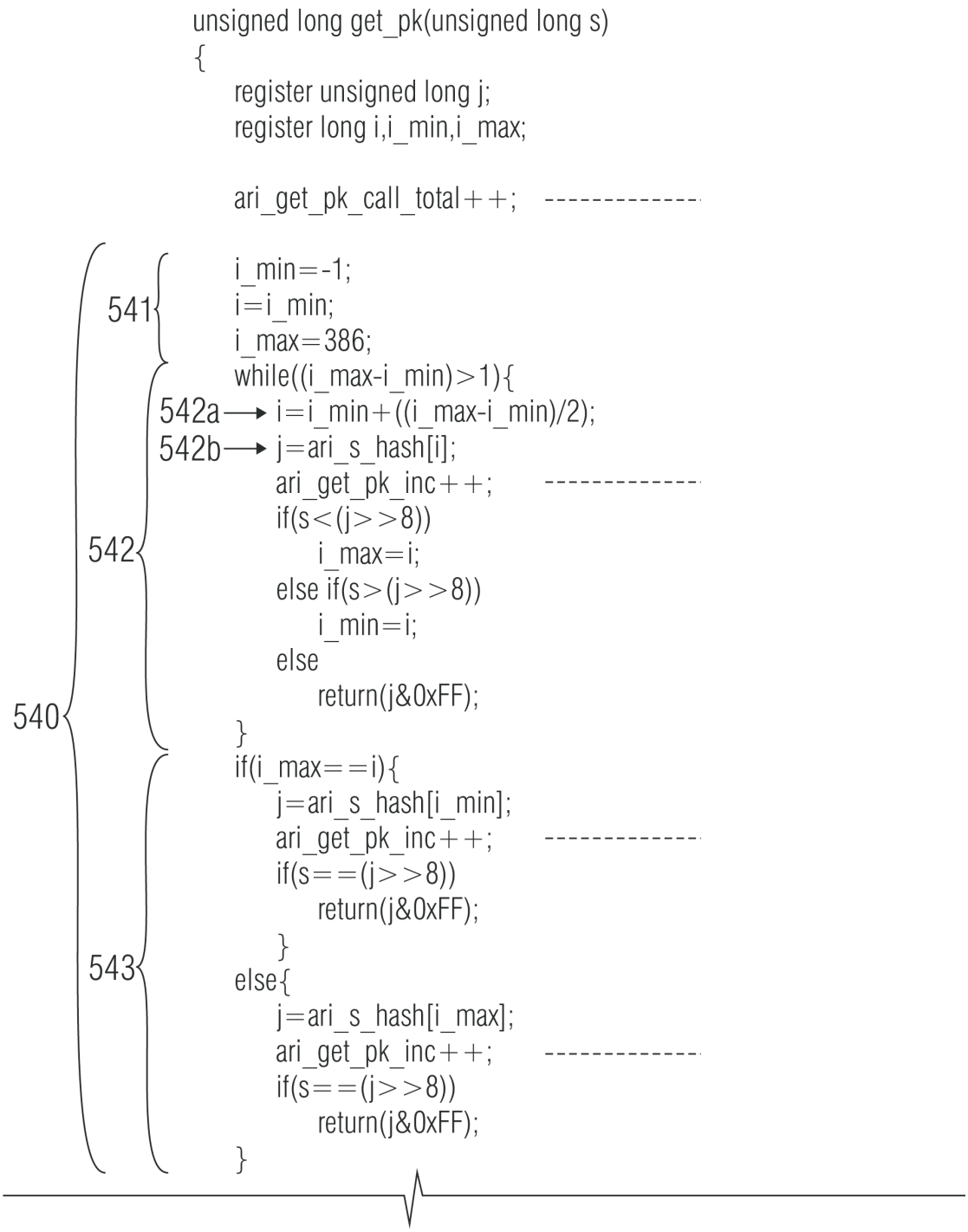

unsigned long get_pk(unsigned long s)
{
register unsigned long j;
register long i,i_min,i_max;
ari_get_pk_call_total++;
i_min=-1;
i=i_min;
i_max=386;
while((i_max-i_min)>1){
i=i_min+((i_max-i_min)/2);
j=ari_s_hash[i];
ari_get_pk_inc++;
if(s<(j>>8))
i_max=i;
else if(s>(j>>8))
i_min=i;
else
return(j&0xFF);
}
if(i_max==i){
j=ari_s_hash[i_min];
ari_get_pk_inc++;
if(s==(j>>8))
return(j&0xFF);
}
else{
j=ari_s_hash[i_max];
ari_get_pk_inc++;
if(s==(j>>8))
return(j&0xFF);
}
541
542a
542b
542
540
543

FIG 5D1

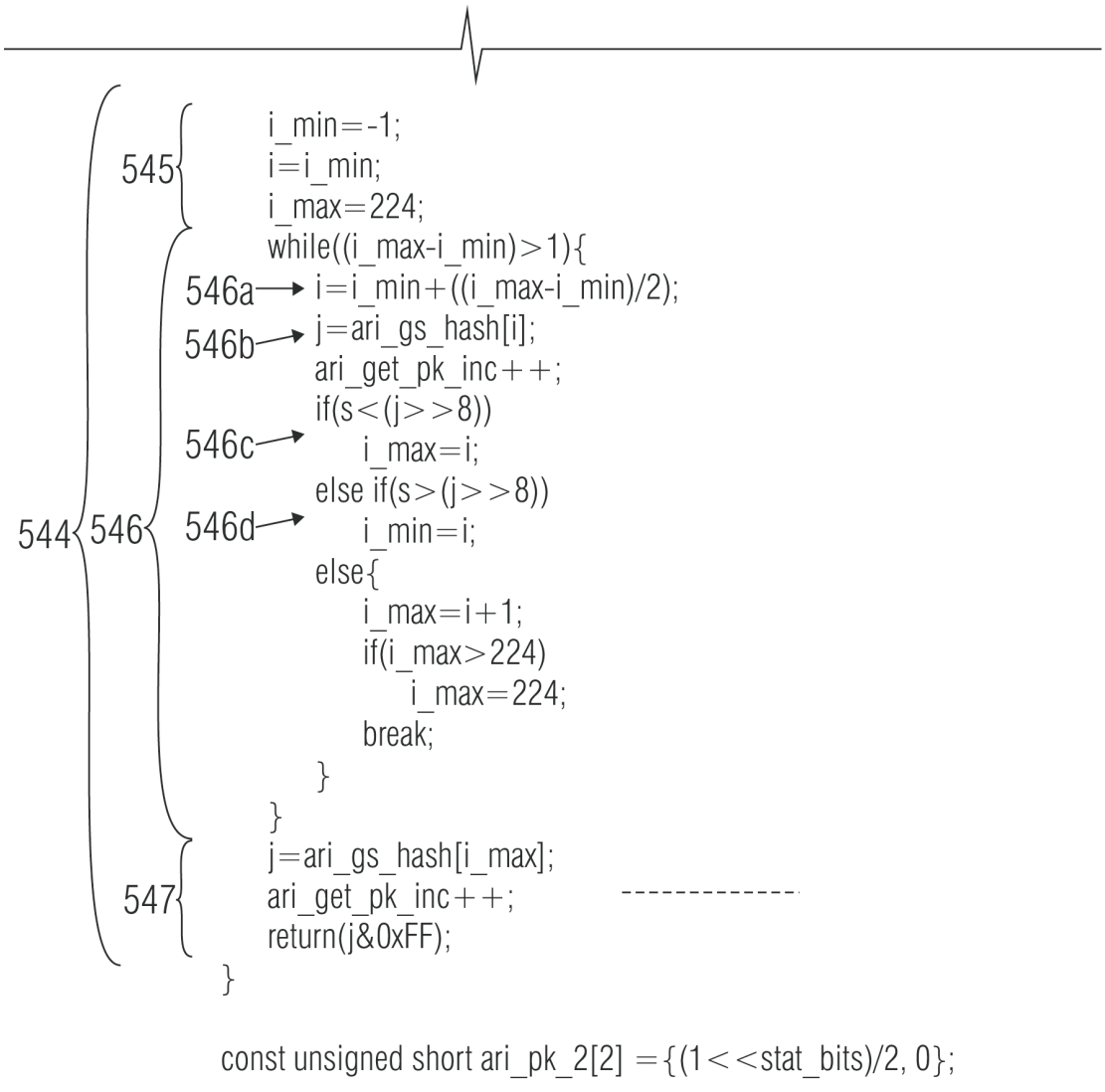

544
545
546
546a
546b
546c
546d
547
i_min=-1;
i=i_min;
i_max=224;
while((i_max-i_min)>1){
i=i_min+((i_max-i_min)/2);
j=ari_gs_hash[i];
ari_get_pk_inc++;
if(s<(j>>8))
i_max=i;
else if(s>(j>>8))
i_min=i;
else{
i_max=i+1;
if(i_max>224)
i_max=224;
break;
}
}
j=ari_gs_hash[i_max];
ari_get_pk_inc++;
return(j&0xFF);
}
const unsigned short ari_pk_2[2] ={(1<<stat_bits)/2, 0};

FIG 5D2

```
/*Input variable*/
s   /*State of the context*/
/*Output value*/
pki /*Index of the probability model */

arith_get_pk(s)
{
    register unsigned long i,j;

    for (i=0;i<387;i++)
    {
        j=ari_s_hash[i];
        if ( (j>>8)==s ) return j&255;
    }
    for (i=0;i<225;i++)
    {
        j=ari_gs_hash[i];
        if ( s<(j>>8) ) return j&255;
    }
    return j&255;
}
```

550

560

FIG 5E

```
unsigned long get_pk(unsigned long s)
{
    register unsigned longlong j;
    register unsigned long  i;

    for (i=0;i<387;i++)
    {
        j=ari_s_hash[i];
        if ( (j>>8)==s )
            return j&0xFF;
    }
    for(i=0;i<225;i++){
        j=ari_gs_hash[i];
        if ( s<(j>>8) ) return j&0xFF;
    }
    return(j&0xFF);
}
```

FIG 5F

```
/*helper funtions*/
bool arith_first_symbol(void);
      /* Return TRUE if it is the first symbol of the sequence, FALSE otherwise*/
Ushort arith_get_next_bit(void);
      /* Get the next bit of the bitstream*/

/* global variables */
low
high
value

/* Input variables */
cum_freq[];   /* cumulative frequencies table*/
cfl;              /* length of cum_freq[] */

arith_decode()
{
    if(arith_first_symbol())
    {
        value = 0;
        for (i=1; i<=20; i++)
        {
        value = (val<<1) | arith_get_next_bit();
        }
        low=0;
        high=1048575;
    }

    range = high-low+1;
    cum =((((int64) (value-low+1))<<16)-((int64) 1))/((int64) range);
    p = cum_freq-1;

    do
    {
        q=p+(cfl>>1);
        if ( *q > cum ) {p=q; cfl++; }
        cfl>>=1;
    }
        while ( cfl>1 );
```

570a

570b

570c

FIG 5G1

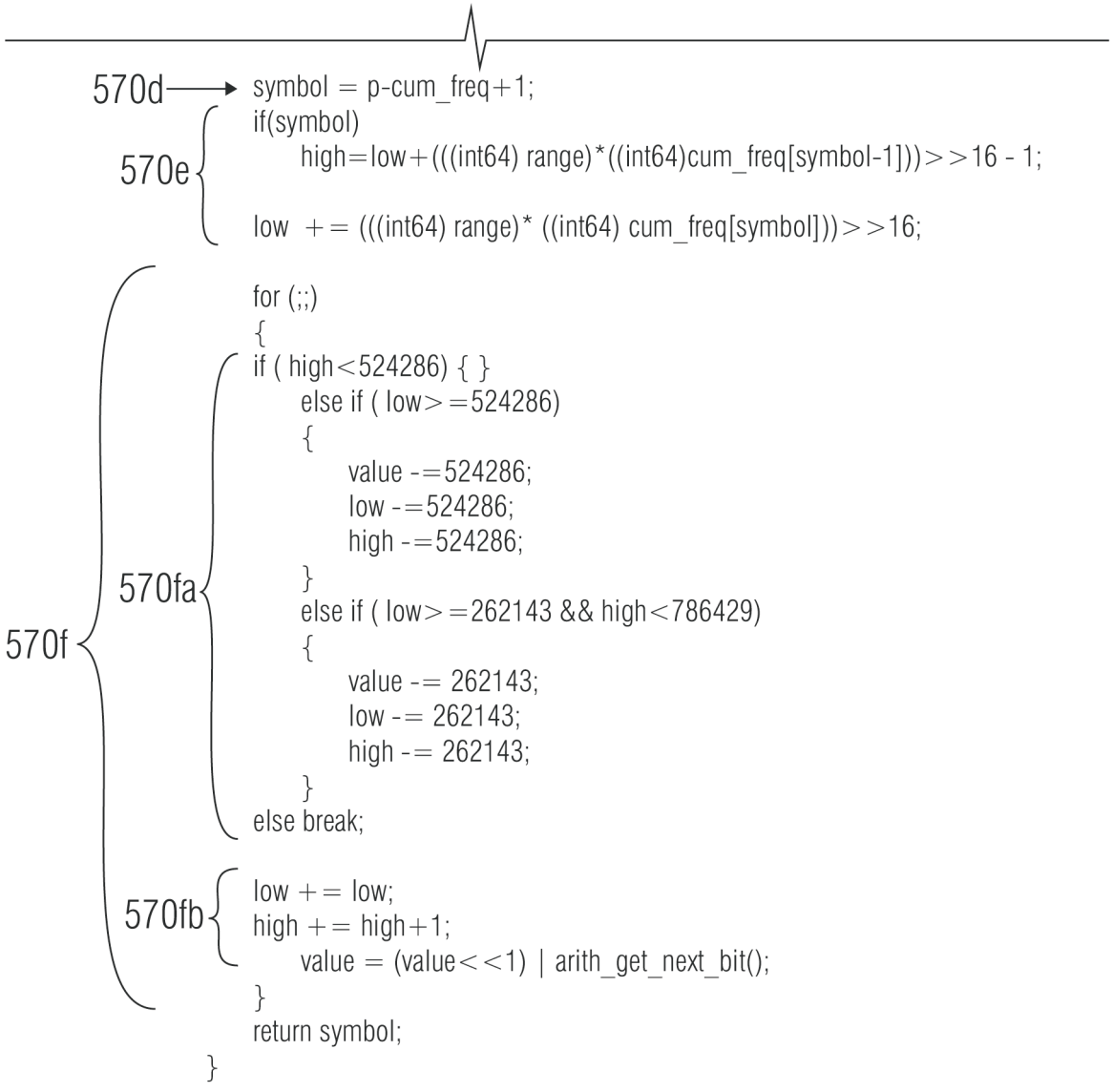

570d
symbol = p-cum_freq+1;
if(symbol)
570e
high=low+(((int64) range)*((int64)cum_freq[symbol-1]))>>16 - 1;
low += (((int64) range)* ((int64) cum_freq[symbol]))>>16;
for (;;)
{
570f
if (high<524286) { }
570fa
else if (low>=524286)
{
value -=524286;
low -=524286;
high -=524286;
}
else if (low>=262143 && high<786429)
{
value -= 262143;
low -= 262143;
high -= 262143;
}
else break;
570fb
low += low;
high += high+1;
value = (value<<1) | arith_get_next_bit();
}
return symbol;
}

FIG 5G2

```
/*input variables*/
a /*Decoded quantized spectral coefficient */
i /*Index of the quantized spectral coefficient to decode*/
lg /*number of coefficients in the frame*/

arith_update_context()
{    int a0;

580 ──► q[1][i]=a0=a;
        q[1][i].l=0;
        while(ABS(a0)>4){                    ┐
582         a0=a0>>1;                        │
            q[1][i].l++;                     │
        }                                    ┘
        if(a==0)                             ┐
            q[1][i].c=0;                     │
        else if(ABS(a)<=1)                   │
            q[1][i].c=1;                     │
584     else if(ABS(a)<=3)                   │
            q[1][i].c=2;                     │
        else                                 │
            q[1][i].c=3;                     ┘

        if(i==lg && core_mode==1){           ┐
            ratio= ((float) lg)/((float)1024);│
            for(j=0; j<1024; j++){           │
586             k = (int) ((float) j*ratio);│
                qs[j] = q[1][k].c ;          │
            }                                │
            previous_lg = 1024;              │
        }                                    ┘
        if(i==lg/4 && core_mode==0){         ┐
            for(j=0; j<MIN(lg,1024; j++){    │
588             qs[j] = q[1][j].c;           │
            }                                │
            previous_lg = MIN(1024,lg);      │
        }                                    ┘
}
```

FIG 5H

a

m

r

lev

lev0

arith_s_hash[]

arith_gs_hash[]

arith_cf_m[pki][9]

arith_cf_r []

previous_lg

N

q[2][]

qs[]

arith_reset_flag

FIG 5I

```
usac_raw_data_block ()
{
    single_channel_element (); and/or
    channel_pair_element ();
}
```

FIG 6A

single_channel_element() {		
core_mode	**1**	**uimsbf**

if (core_mode == 1) {
lpd_channel_stream();
}
else {
fd_channel_stream();
}		

}		

FIG 6B

```
channel_pair_element()
{
    core_mode0                                  1          uimsbf
    core_mode1                                  1          uimsbf

            ics_info();


    if ( core_mode0 == 1 ) {
        lpd_channel_stream();
    }
    else {
        fd_channel_stream();
    }

    if ( core_mode1 == 1 ) {
        lpd_channel_stream();
    }
    else {
        fd_channel_stream();

    }
}
```

FIG 6C

Petição 870220059685, de 06/07/2022, pág. 141/170

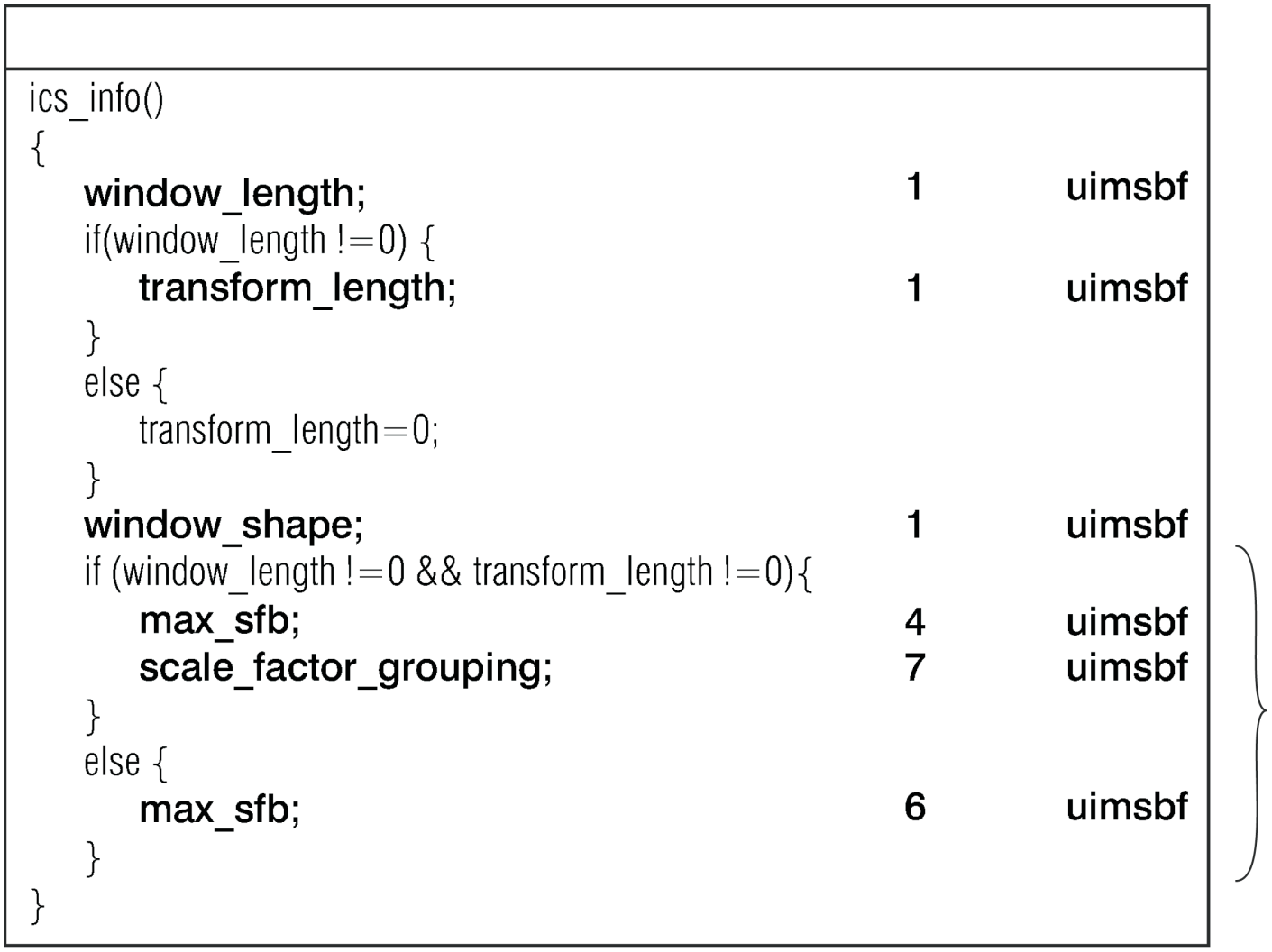


FIG 6D

fd_channel_stream() {		
global_gain;	**8**	**uimsbf**
ics_info();		
scale_factor_data ();		
ac_spectral_data (); }		

FIG 6E

ac_spectral_data() {		
arith_reset_flag	**1**	**uimsbf**

for (win=0; win<num_windows; win++){
arith_data(num_bands, arith_reset_flag)
}
}		

FIG 6F

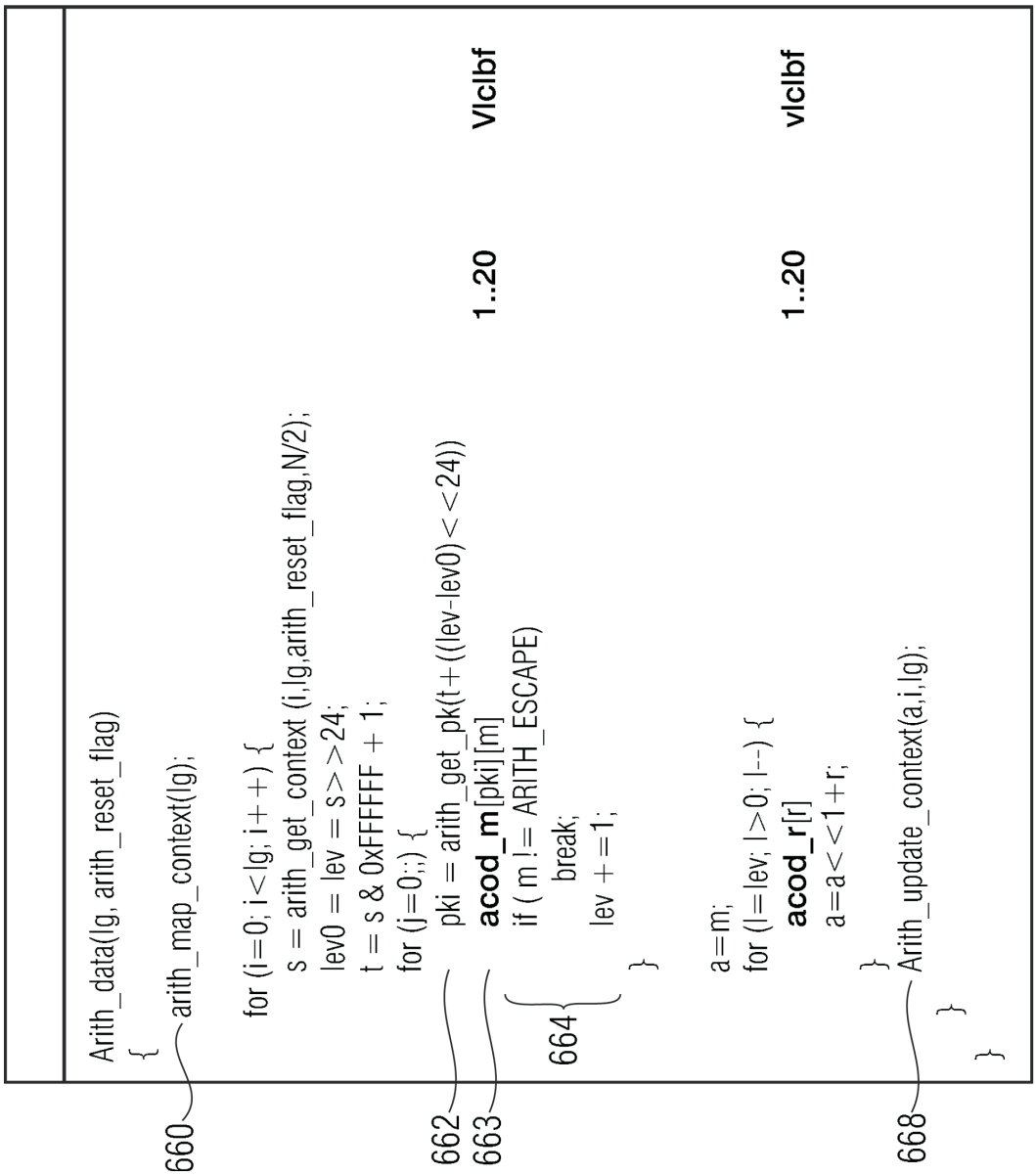

Arith_data(lg, arith_reset_flag)
{
arith_map_context(lg);
for (i=0; i<lg; i++) {
s = arith_get_context (i,lg,arith_reset_flag,N/2);
lev0 = lev = s>>24;
t = s & 0xFFFFFF + 1;
for (j=0;;) {
pki = arith_get_pk(t+((lev-lev0)<<24))
1..20
VlcIbf
acod_m[pki][m]
if (m != ARITH_ESCAPE)
break;
lev +=1;
}
a=m;
for (l=lev; l>0; l--) {
acod_r[r]
1..20
vlcIbf
a=a<<1+r;
}
Arith_update_context(a,i,lg);
}
}
660
662
663
664
668

FIG 6G

arith_data()

arith_reset_flag

acod _cf_m[pki][a]

arith_cf_r[]

Help elements

a

m

r

N

lg

i

pki

arith_get_pk ()

t

arith_get_context ()

lev0

s

lev

ARITH_ESCAPE

FIG 6H

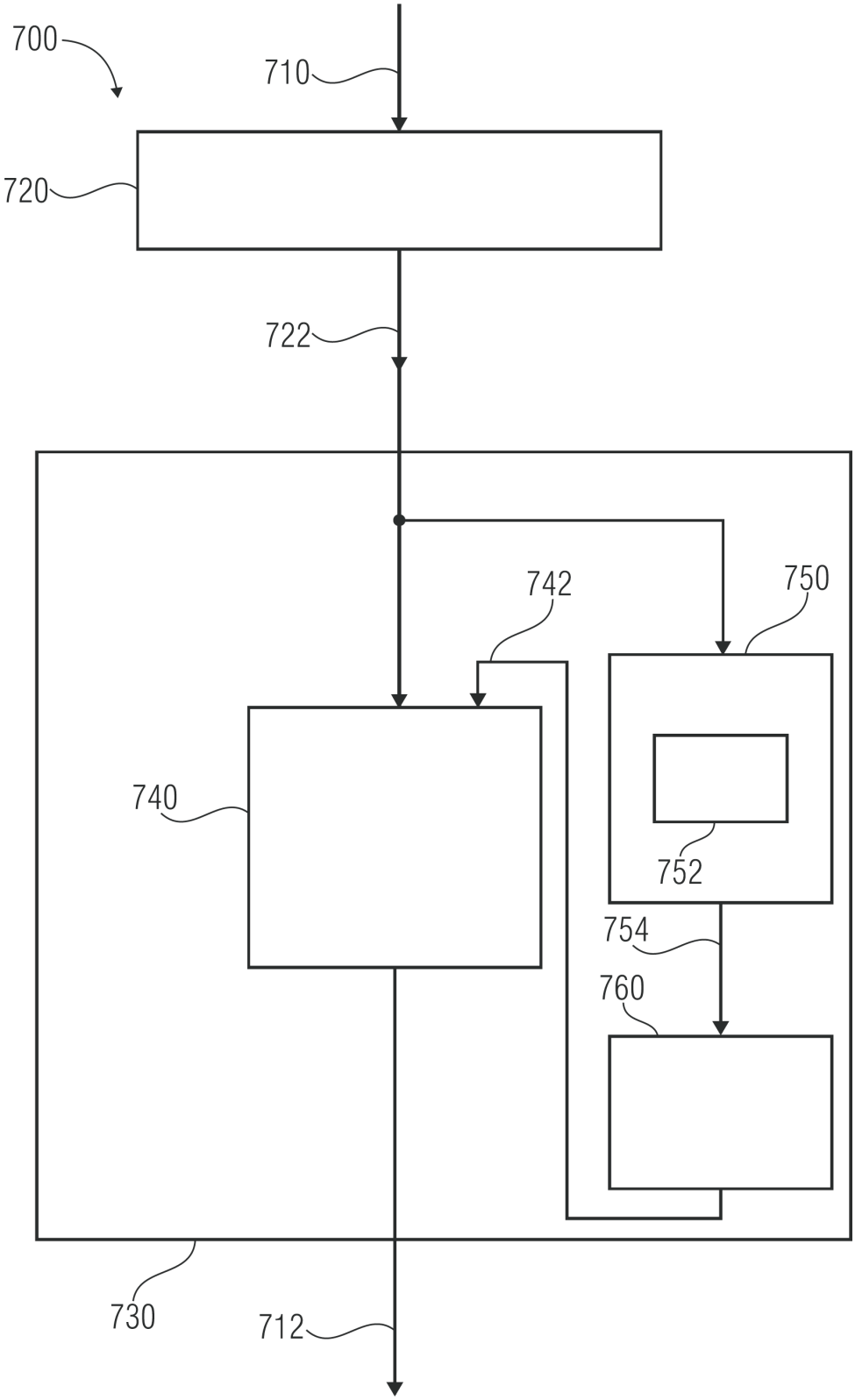


FIG 7

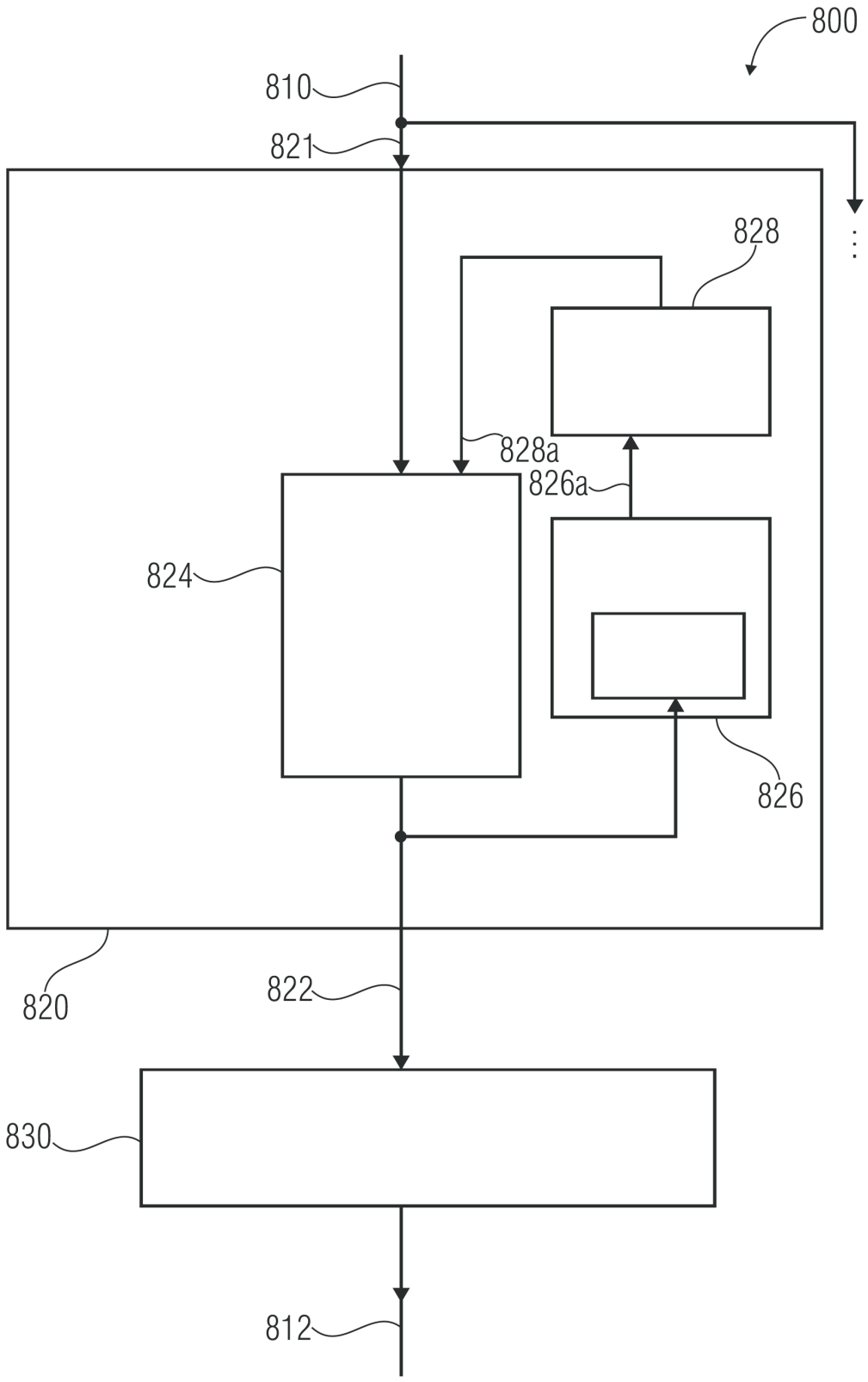


FIG 8

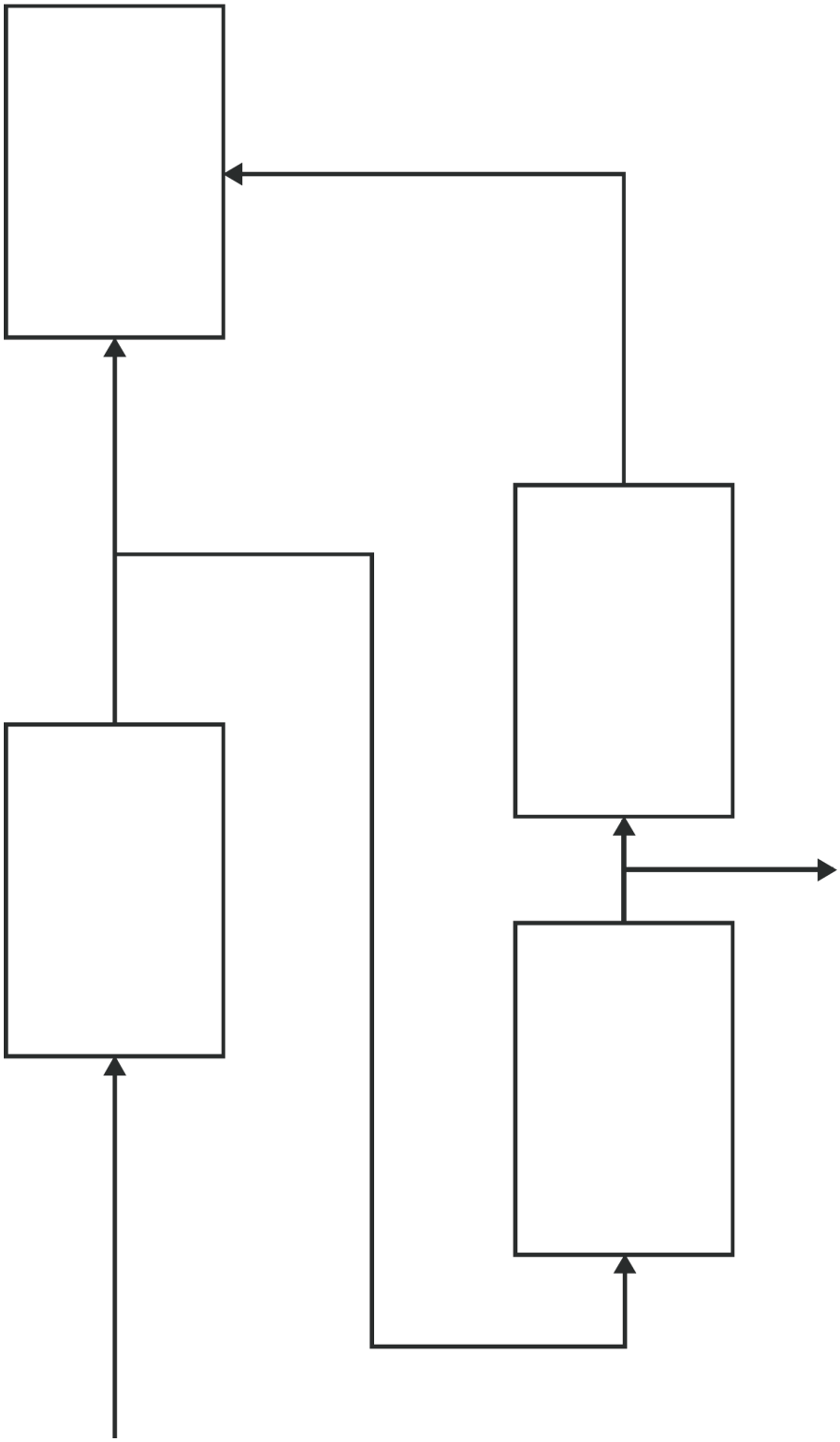


FIG 9

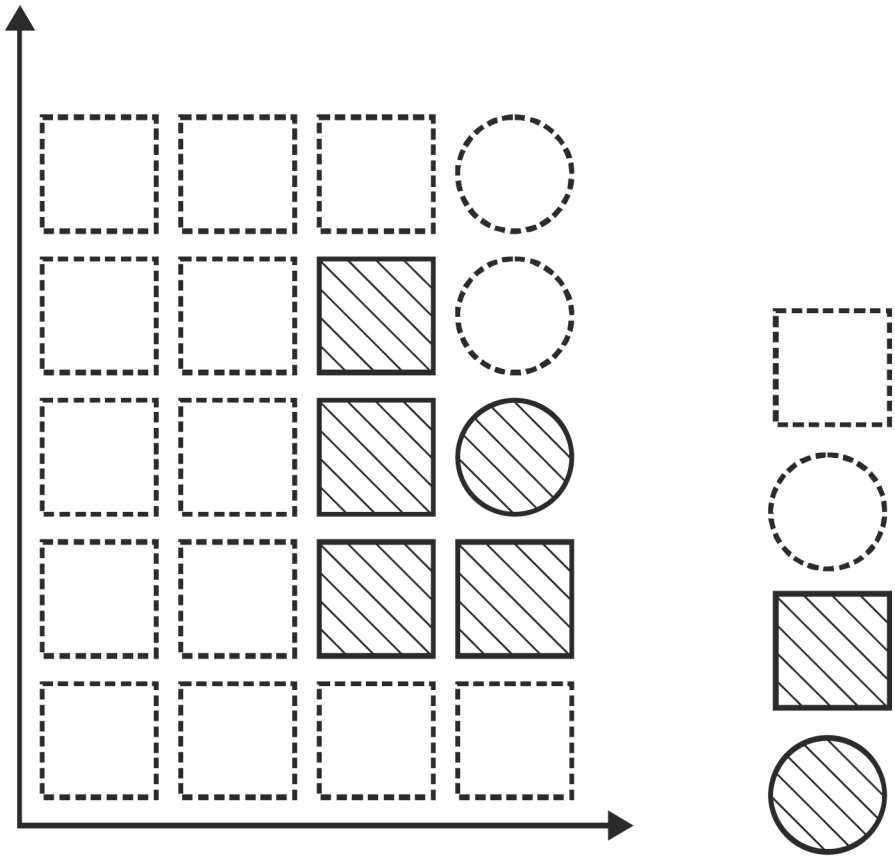


FIG 10A

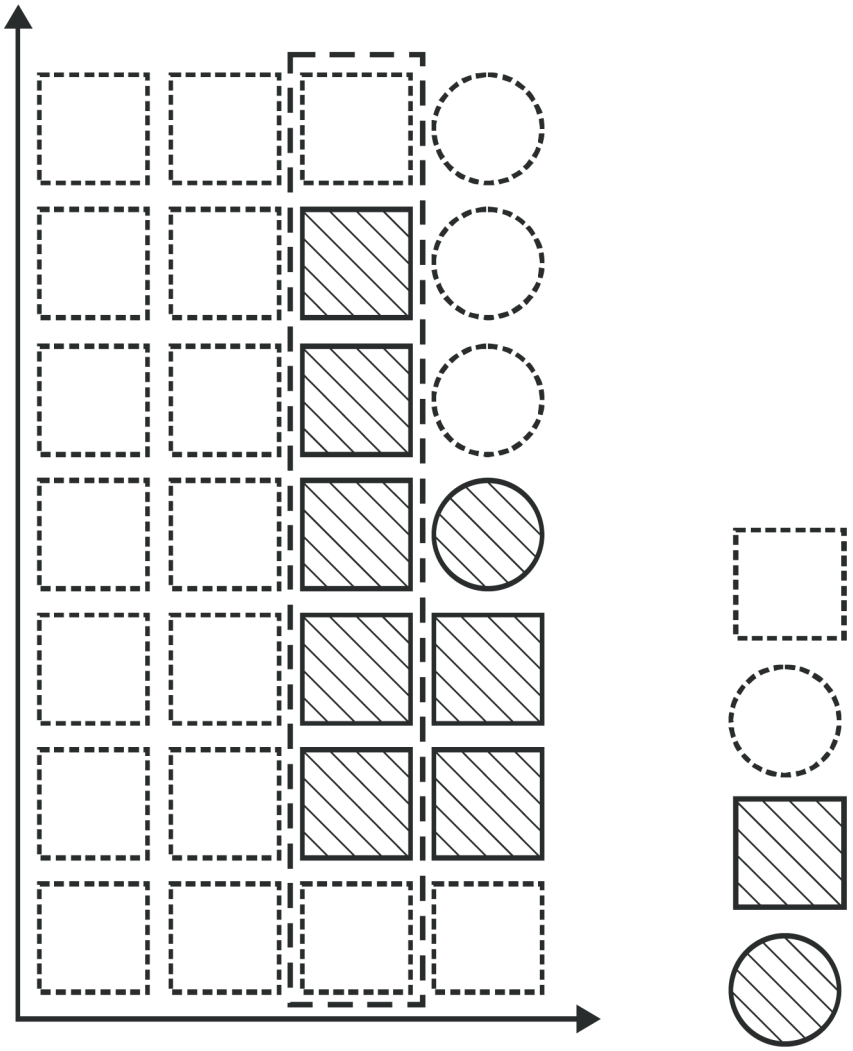


FIG 10B

Petição 870220059685, de 06/07/2022, pág. 150/170

arith_cf_ng_hash[128]			128
arith_cf_ng[32][545]			8720
egroups[8][8][8][8]			2048
dgvectors[4*4096]			4096
dgroups[544]			544
arith_cf_ne[2701]			1350.5
arith_cf_r[16]			8
			16894.5

FIG 11A

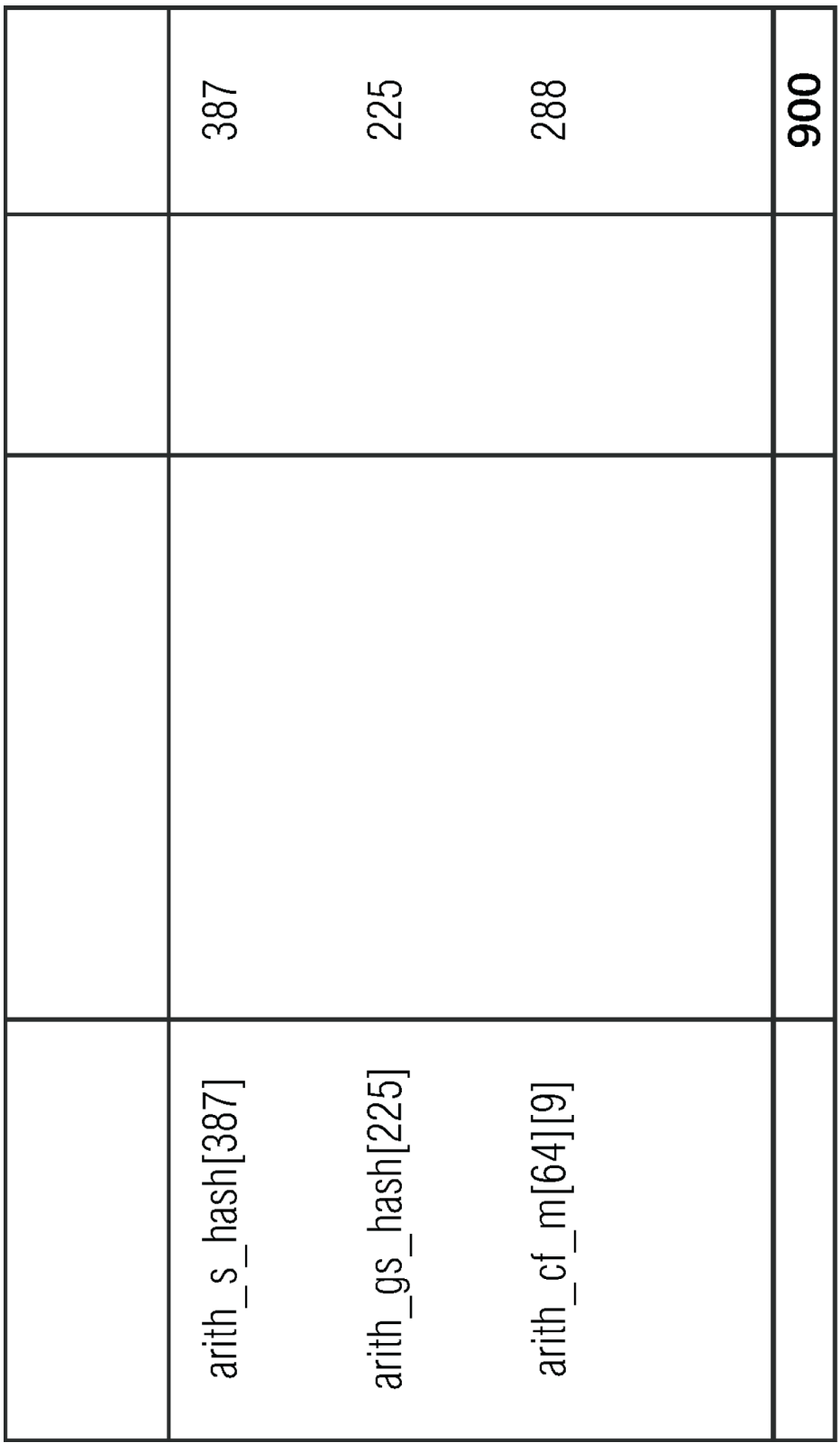

arith_s_hash[387]
arith_gs_hash[225]
arith_cf_m[64][9]
387
225
288
900

FIG 11B

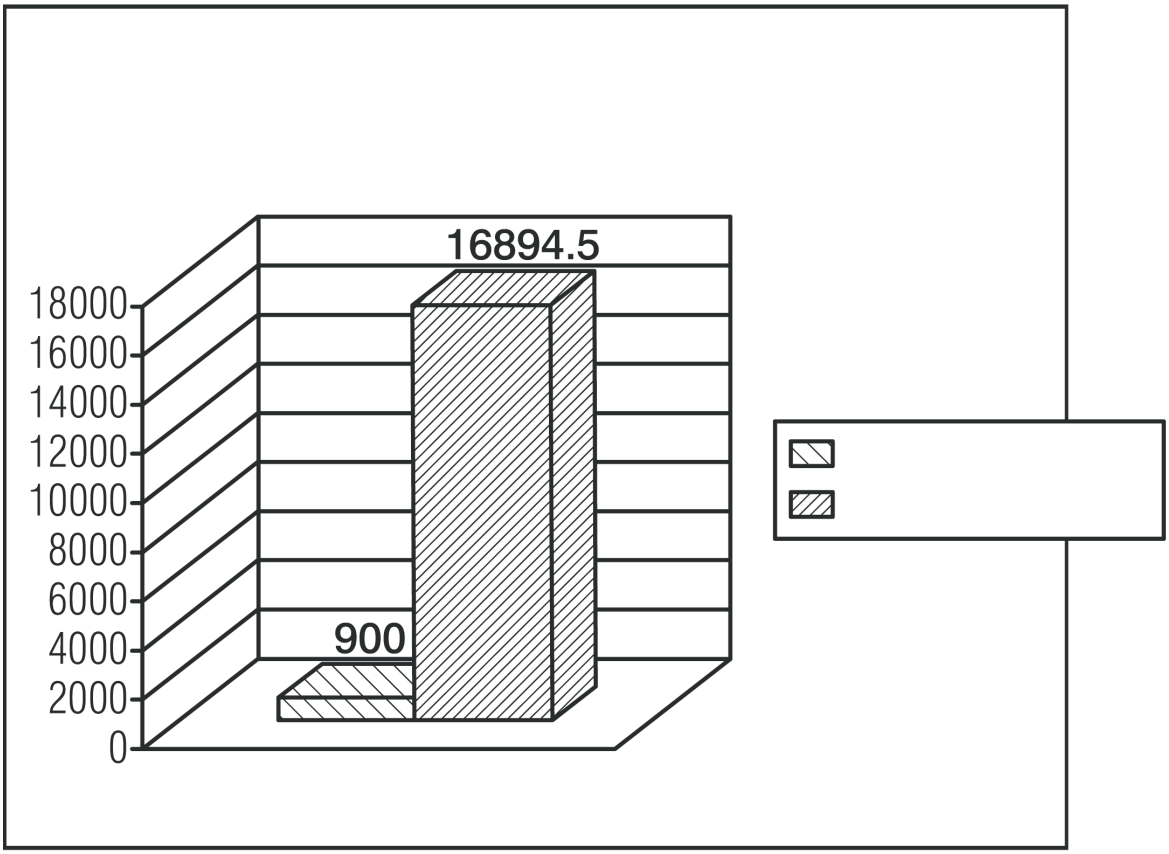

16894.5
900
18000
16000
14000
12000
10000
8000
6000
4000
2000
0

FIG 12A

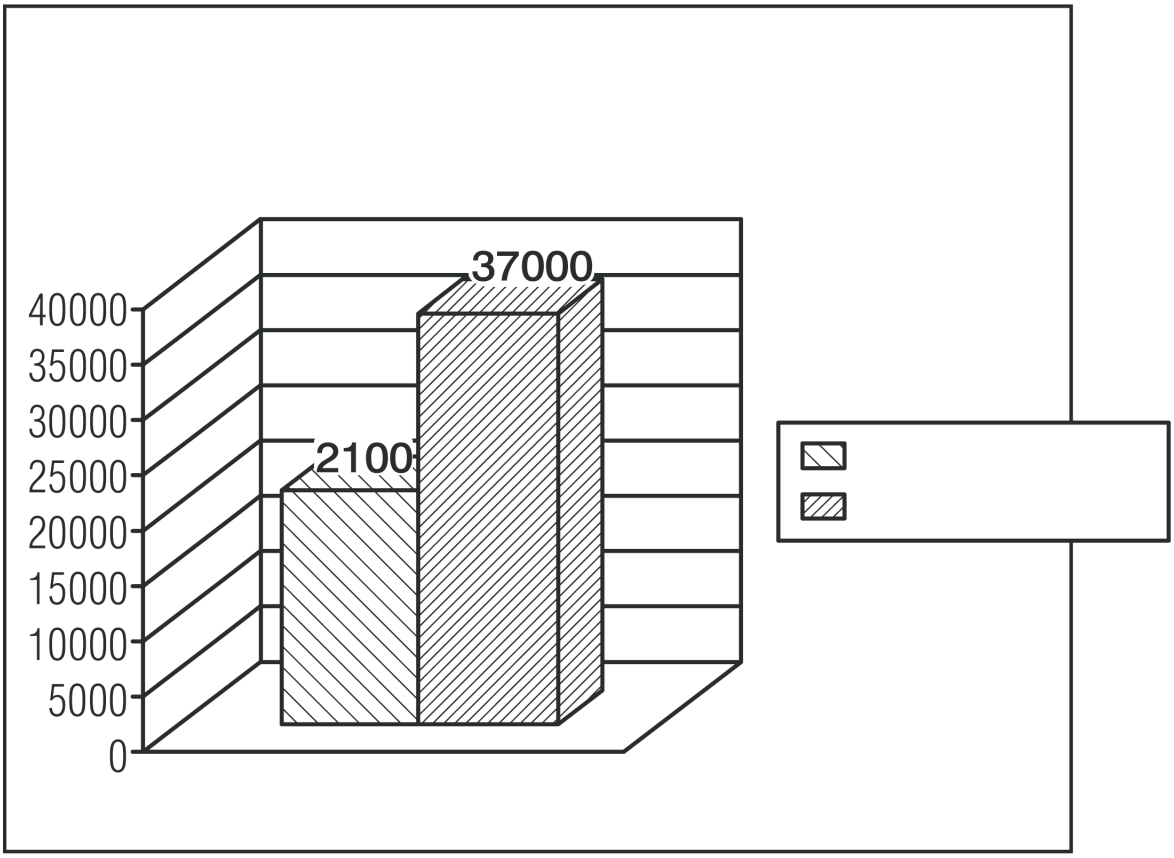


FIG 12B

	64.00	63.34	-0.66	-1.04
	32.00	31.66	-0.34	-1.05
	24.00	23.73	-0.27	-1.11
	20.00	19.78	-0.22	-1.11
	16.00	15.82	-0.18	-1.10
	24.00	23.68	-0.32	-1.32
	20.00	19.72	-0.28	-1.39
	16.00	15.79	-0.21	-1.31
	12.00	11.86	-0.14	-1.19

FIG 13A

	3653	9557	8137	2314	9557	7018
	1808	4505	4196	581	4505	3530
	1538	4704	4408	957	4704	3871
	2367	4864	4600	712	4864	3854
	2712	5006	4804	724	5006	4234
	2185	4704	4457	1002	4704	3927
	2599	4864	4630	1192	4864	3935
	2820	5006	4876	1434	5006	4450
	3529	5184	5081	2256	5184	4787

FIG 13B

	53.73	---	53.73	54.40	---	54.40
	25.31	26.34	25.60	25.80	26.61	26.02
	18.27	19.17	18.50	18.66	19.40	18.85
	15.50	15.93	15.61	15.83	16.12	15.90
	12.45	12.60	12.52	12.80	12.73	12.77
	19.94	19.51	19.73	20.41	19.42	20.15
	16.15	15.91	16.08	16.56	16.12	16.45
	13.02	12.59	12.81	13.45	12.73	13.09
	9.35	9.66	9.51	9.68	9.71	9.70

FIG 14

	15.26	101.79	63.34
	13.13	48.61	31.66
	11.69	36.58	23.73
	3.09	30.94	19.78
	4.02	26.47	15.82
	1.47	37.35	23.68
	1.38	31.13	19.72
	11.40	24.64	15.79
	8.72	18.91	11.86

FIG 15

	-30.87	-33.06	6.14	9.07
	-10.33	-28.63	2.17	6.77
	-11.86	-30.75	1.85	7.71
	-7.45	-30.27	1.67	8.36
	-5.43	-27.89	1.50	9.42
	-17.06	-45.83	1.25	4.36
	-15.86	-41.46	0.88	3.38
	-4.75	-24.85	1.11	7.31
	-3.95	-26.33	0.82	6.99

FIG 16

```
/*
Entropy:
fu mem.: 1.2792 bit (100.00 %)
no mem.  : 1.6289 bit (127.34 %)
split:   : 1.2971 bit (101.40 %)
*/

/*  1224 States, Entropy increase: 0.000384 */

/*Final Entropy : 1.297556 */

/*Total states = 612;*/
/*Signicant states = 387;*/
/*Pseudo states = 225;*/
/*Proba models = 64;*/
unsigned long long ari_get_pk_inc=0;
unsigned long long ari_get_pk_call_total=0;

static unsigned long ari_s_hash[387] = {
0x00000200,0x00000B01,0x00000C02,0x00000D03,0x00000F25,0x0000101C,0x0000110B,
0x00001327,
0x0000142F,0x00002B25,0x00002C22,0x00002D14,0x00002F2D,0x0000302B,0x0000312B,
0x00003330,
0x00003432,0x00003532,0x00004C32,0x00005031,0x00005131,0x0000FA02,0x0000FB01,
0x0000FC1C,
0x0000FE1C,0x0000FF1C,0x0001001E,0x00010A2E,0x00010B25,0x00010E25,0x00010F25,
0x00013938,
0x00013A04,0x00013B02,0x00013C01,0x0010393D,0x00107504,0x00107605,0x00107706,
0x0010790D,
0x00107A07,0x00107B08,0x0010850D,0x00108609,0x0010870A,0x00108B0E,0x0010B50B,
0x0010B60C,
0x0010B70D,0x0010B90B,0x0010BA1D,0x0010BB16,0x0010C615,0x0010C70C,0x0010F521,
0x0010F628,
0x0010F728,0x00110528,0x00117516,0x0011760E,0x0011770F,0x00117A12,0x00117B07,
0x0011870E,
0x0011B514,0x0011B615,0x0011B70C,0x0011B914,0x0011BA15,0x0011BB1D,0x0011C619,
0x0011C715,
0x00147516,0x00147610,0x00147711,0x0014791D,0x00147A0C,0x00147B0E,0x0014851B,
0x00148616,
0x00148707,0x0014890B,0x00148A1D,0x00148B16,0x0014950B,0x0014961D,0x0014B615,
0x0014B71D,
0x0014BA14,0x0014BB15,0x0014C614,0x0014C715,0x0015052D,0x0015750B,0x0015760C,
0x00157710,
0x0015790B,0x00157A1D,0x00157B16,0x0015850B,0x0015861D,0x0015870C,0x00158914,
0x00158A15,
0x00158B1D,0x0015B619,0x0015B714,0x0020751D,0x00207612,0x00207713,0x0020791D,
0x00207A0C,
0x00207B0E,0x0020850B,0x0020860C,0x00208710,0x00208A1D,0x00208B0C,0x0020B615,
0x0020B71D,
0x0021750B,0x0021760C,0x0021770E,0x0021790B,0x00217A1D,0x00217B12,0x00218514,
0x00218615,
0x0021870C,0x0021891C,0x00218A15,0x00218B1D,0x0021B619,0x0021B715,0x0021BA22,
0x0021BB19,
0x00247514,0x0024761D,0x0024770E,0x00247914,0x00247A15,0x00247B0C,0x00248514,
0x0024861D,
0x00248716,0x0024891C,0x00248A15,0x00248B1D,0x0024B619,0x0024B715,0x0025751C,
0x0025760B,
```

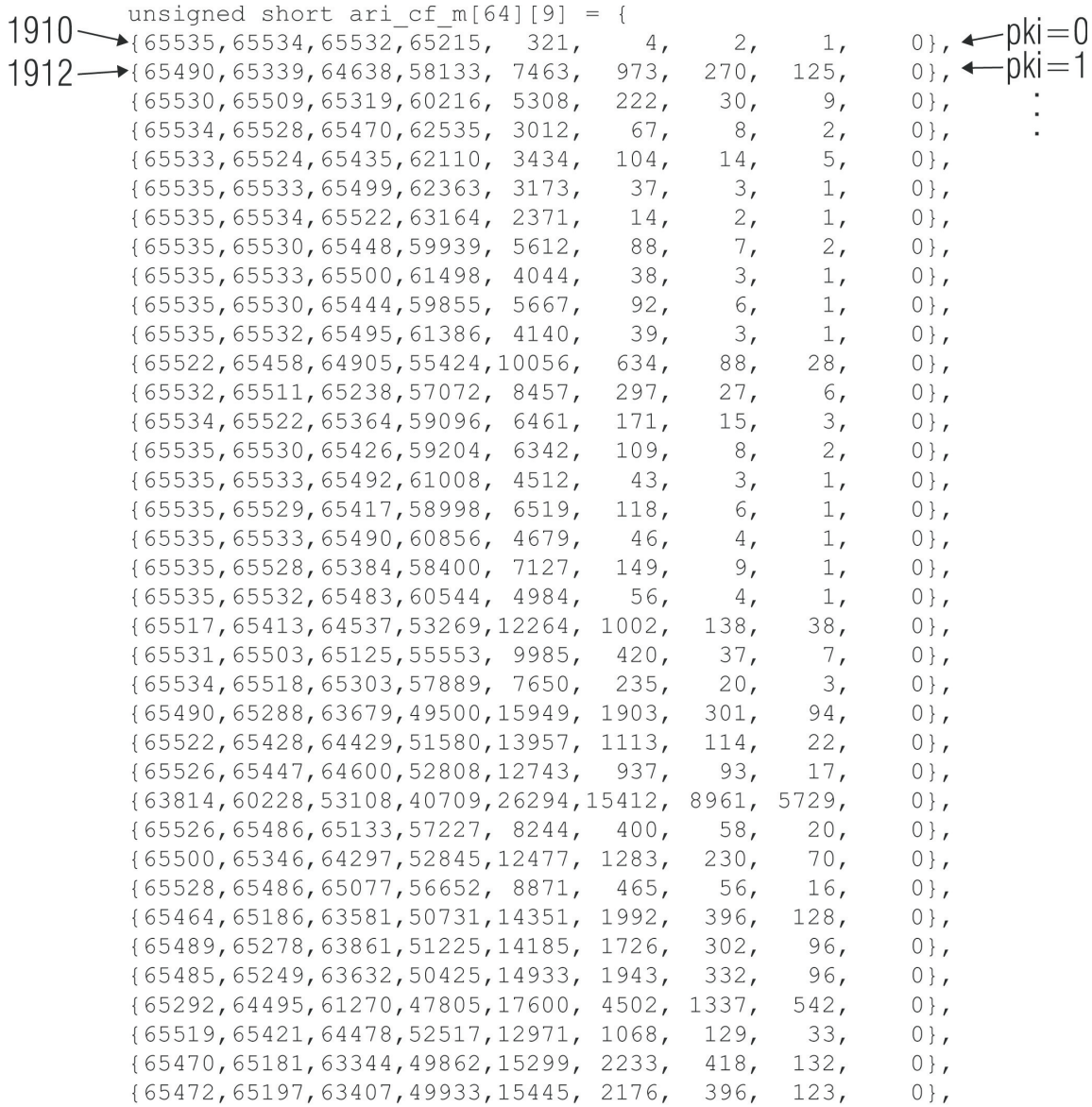
FIG 17(1)

```
0x0025771B,0x0025791C,0x00257A0B,0x00257B1B,0x0025851C,0x0025860B,0x0025871B,
0x0025890B,
0x00258A1D,0x00258B1B,0x0025B618,0x0025B722,0x0025BA1F,0x0025BB18,0x0025C61F,
0x0025C718,
0x0025CA1F,0x0025CB1F,0x003A9D1C,0x003A9E0B,0x003A9F0B,0x004FB125,0x004FB21C,
0x004FB31C,
0x00907514,0x00907615,0x00907716,0x00907919,0x00907A15,0x00907B1D,0x00907D1C,
0x00907E1C,
0x00907F14,0x00908522,0x00908614,0x0090871D,0x00908918,0x00908A19,0x00908B14,
0x00908D24,
0x00909523,0x0090961F,0x0090992B,0x0090B517,0x0090B618,0x0090B719,0x0090B91F,
0x0090BA22,
0x0090BB19,0x0090BD1C,0x0090BE1C,0x0090BF1C,0x0090C52B,0x0090C61F,0x0090C718,
0x0090C917,
0x0090CA1F,0x0090CB1F,0x0090CD23,0x0090D52E,0x0090D62C,0x0090D92C,0x0090F52D,
0x0090F62D,
0x0090F72F,0x0090F925,0x0090FA2E,0x0090FB2D,0x0090FD1E,0x0090FE1E,0x0091052F,
0x0091062F,
0x00910928,0x00910D25,0x00917519,0x00917615,0x0091771D,0x00917922,0x00917A14,
0x00917B15,
0x00918619,0x00918714,0x00918A18,0x00918B18,0x0091B618,0x0091B722,0x0091BA18,
0x0091BB18,
0x00947518,0x00947614,0x0094771D,0x0094791F,0x00947A19,0x00947B14,0x00948520,
0x00948619,
0x00948714,0x00948A18,0x00948B18,0x0094B61F,0x0094B718,0x0094BA17,0x0094BB1F,
0x0094C617,
0x0094C717,0x00957520,0x00957622,0x00957714,0x00957924,0x00957A18,0x00957B18,
0x00958524,
0x00958618,0x00958718,0x0095892B,0x00958A17,0x00958B1F,0x0095B52B,0x0095B617,
0x0095B71F,
0x0095B92B,0x0095BA17,0x0095BB17,0x0095C52C,0x0095C617,0x0095C717,0x0095C92C,
0x0095CA2B,
0x0095CB2C,0x00A0751F,0x00A07614,0x00A07715,0x00A0791F,0x00A07A19,0x00A07B14,
0x00A0851F,
0x00A08622,0x00A08714,0x00A08917,0x00A08A18,0x00A08B18,0x00A0B52B,0x00A0B61F,
0x00A0B718,
0x00A0BA17,0x00A0BB1F,0x00A0C617,0x00A0C717,0x00A17524,0x00A17622,0x00A17714,
0x00A17924,
0x00A17A18,0x00A17B18,0x00A1861F,0x00A18718,0x00A18A17,0x00A18B17,0x00A1B617,
0x00A1B71F,
0x00A1BA17,0x00A1BB17,0x00A47524,0x00A47618,0x00A47714,0x00A4792B,0x00A47A18,
0x00A47B18,
0x00A4852B,0x00A4861F,0x00A48718,0x00A4892B,0x00A48A17,0x00A48B17,0x00A4B52C,
0x00A4B617,
0x00A4B717,0x00A4B92C,0x00A4BA17,0x00A4BB17,0x00A4C52E,0x00A4C62B,0x00A4C72B,
0x00A4C92E,
0x00A4CA2C,0x00A4CB2C,0x00A5752C,0x00A57617,0x00A57718,0x00A5792B,0x00A57A17,
0x00A57B1F,
0x00A5852C,0x00A58617,0x00A5871F,0x00A5892C,0x00A58A17,0x00A58B17,0x00A5B52E,
0x00A5B62B,
0x00A5B717,0x00A5B92E,0x00A5BA2C,0x00A5BB2C,0x00A5C52E,0x00A5C62C,0x00A5C72C,
0x00A5C92E,
0x00A5CA2C,0x00A5CB2C,0x00BA9D2D,0x00BA9E2E,0x00BA9F2E,0x00CDD938,0x00CE2D38,
0x00CEE938,
0x00CF2D38,0x00D02D38,0x00D0313A,0x00D0713A,0x0110762F,0x0110B632,0x0110B731,
0x03D0113B,
0x03D0213C,0x03D02D3D,0x03D0313D,0x03D0413C,0x03D04D3D,0x03D0513C,0x03D05D3D,
0x03D06139,
0x03D0693D,0x03D06D3D,0x03D0713D
};
```

FIG 17(2)

```
static unsigned long ari_gs_hash[225] = {
0x00000401,0x0001491A,0x0001590B,0x00017621,0x0001891C,0x0009492A,0x000DFA38,
0x000F6D3D,
0x0010003B,0x0010B21B,0x0011321C,0x00116B29,0x0011B31D,0x00126B1E,0x00136623,
0x00146729,
0x00146F3B,0x0015321F,0x00156E27,0x00163320,0x00182725,0x00186727,0x00196323,
0x001C4721,
0x001E3F30,0x001E433B,0x00203F2A,0x0020463B,0x0020F322,0x00216A2E,0x00226723,
0x00245625,
0x00256724,0x00286625,0x002D3726,0x002D573A,0x00316627,0x00326628,0x00344729,
0x00366628,
0x003D4329,0x00416A2A,0x0042533A,0x00916A2A,0x00926B2B,0x0093E72E,0x00956B2C,
0x009D362D,
0x009D3B39,0x009E4330,0x00A2672E,0x00AD372F,0x01145630,0x01146B27,0x011C8231,
0x01226732,
0x012CC333,0x01413B34,0x019CA335,0x019CB338,0x01ACB736,0x01AD823D,0x01C37F37,
0x02156738,
0x0218AB3B,0x021C9B35,0x021E0738,0x021FB73D,0x0220E335,0x02216B3C,0x02217234,
0x0222B33C,
0x02239B3B,0x0223B23A,0x0224673B,0x0238A739,0x0240B23D,0x024CBF38,0x024CC23D,
0x024D8738,
0x0297AF3A,0x02986727,0x0298A33B,0x0298A738,0x029CAF3B,0x029CC33A,0x02A0AB35,
0x02A3E736,
0x02AC773B,0x02B0B335,0x02B3A73B,0x02C0D73C,0x02C1E735,0x03108E3D,0x03109737,
0x0311D639,
0x03147F3C,0x0314B236,0x0317A639,0x0317D629,0x0317DB33,0x03187627,0x0318AF3B,
0x0318F61A,
0x0319D739,0x031C953B,0x031D633C,0x031FCF39,0x0320873B,0x0320963A,0x03222639,
0x0323833C,
0x03239A27,0x0323EA2F,0x03242631,0x03242B3B,0x03249727,0x0325AB39,0x0327A73C,
0x0327C728,
0x03287727,0x03287E3A,0x03288737,0x032BAA39,0x032C7527,0x032D2337,0x032E9B39,
0x032EA23B,
0x032EBF3C,0x032F7E39,0x0330C63C,0x0332B23B,0x0332F230,0x03339F3B,0x0333EE27,
0x03348F30,
0x0336AB3C,0x0338A73B,0x033A7639,0x033A7F1A,0x033C793B,0x033C9A34,0x033CA33B,
0x033CA738,
0x033D0A3C,0x033DB339,0x033DFF3C,0x033E9739,0x0340CB3C,0x0344573B,0x0344AA3C,
0x0348263B,
0x034C7B3C,0x034CBB3A,0x034CD33C,0x0390B73D,0x0390E937,0x0393653D,0x0394B73B,
0x0394E33D,
0x0394FA38,0x03950A3C,0x0396CF3D,0x03971A36,0x0398673C,0x0398E13B,0x03994E39,
0x039C733B,
0x039D191A,0x039D4536,0x039E053C,0x039E6E3D,0x039E9D34,0x039F8D39,0x03A0C93B,
0x03A67939,
0x03A69D29,0x03A6D637,0x03A85A3C,0x03AE5B3B,0x03AEDB3D,0x03AF2E3C,0x03B0A13B,
0x03B2B139,
0x03B3123B,0x03B36339,0x03B3AD3C,0x03B42E33,0x03B4733B,0x03B4F53C,0x03B51F36,
0x03B59139,
0x03B5CB3C,0x03B61737,0x03B93A3C,0x03B98F39,0x03B9F53C,0x03BA063B,0x03BA2A3C,
0x03BB2739,
0x03BD3B3B,0x03BDC939,0x03BDF534,0x03BF9A39,0x03C1653B,0x03C19E2A,0x03C20527,
0x03C3633B,
0x03C3823C,0x03C3A527,0x03C45A3B,0x03C4993C,0x03C5B23B,0x03C5D527,0x03C9563B,
0x03C9A93C,
0x03CA063B,0x03CB0E3C,0x03CCB53B,0x03CD1E3C,0x03CED23D,0x03CEDF3C,0x03CFFA39,
0x40BC673E,
0xFFFFFF3F
};
```

FIG 18




```
unsigned short ari_cf_m[64][9] = {
{65535,65534,65532,65215,  321,    4,    2,    1,    0},
{65490,65339,64638,58133, 7463,  973,  270,  125,    0},
{65530,65509,65319,60216, 5308,  222,   30,    9,    0},
{65534,65528,65470,62535, 3012,   67,    8,    2,    0},
{65533,65524,65435,62110, 3434,  104,   14,    5,    0},
{65535,65533,65499,62363, 3173,   37,    3,    1,    0},
{65535,65534,65522,63164, 2371,   14,    2,    1,    0},
{65535,65530,65448,59939, 5612,   88,    7,    2,    0},
{65535,65533,65500,61498, 4044,   38,    3,    1,    0},
{65535,65530,65444,59855, 5667,   92,    6,    1,    0},
{65535,65532,65495,61386, 4140,   39,    3,    1,    0},
{65522,65458,64905,55424,10056,  634,   88,   28,    0},
{65532,65511,65238,57072, 8457,  297,   27,    6,    0},
{65534,65522,65364,59096, 6461,  171,   15,    3,    0},
{65535,65530,65426,59204, 6342,  109,    8,    2,    0},
{65535,65533,65492,61008, 4512,   43,    3,    1,    0},
{65535,65529,65417,58998, 6519,  118,    6,    1,    0},
{65535,65533,65490,60856, 4679,   46,    4,    1,    0},
{65535,65528,65384,58400, 7127,  149,    9,    1,    0},
{65535,65532,65483,60544, 4984,   56,    4,    1,    0},
{65517,65413,64537,53269,12264, 1002,  138,   38,    0},
{65531,65503,65125,55553, 9985,  420,   37,    7,    0},
{65534,65518,65303,57889, 7650,  235,   20,    3,    0},
{65490,65288,63679,49500,15949, 1903,  301,   94,    0},
{65522,65428,64429,51580,13957, 1113,  114,   22,    0},
{65526,65447,64600,52808,12743,  937,   93,   17,    0},
{63814,60228,53108,40709,26294,15412, 8961, 5729,    0},
{65526,65486,65133,57227, 8244,  400,   58,   20,    0},
{65500,65346,64297,52845,12477, 1283,  230,   70,    0},
{65528,65486,65077,56652, 8871,  465,   56,   16,    0},
{65464,65186,63581,50731,14351, 1992,  396,  128,    0},
{65489,65278,63861,51225,14185, 1726,  302,   96,    0},
{65485,65249,63632,50425,14933, 1943,  332,   96,    0},
{65292,64495,61270,47805,17600, 4502, 1337,  542,    0},
{65519,65421,64478,52517,12971, 1068,  129,   33,    0},
{65470,65181,63344,49862,15299, 2233,  418,  132,    0},
{65472,65197,63407,49933,15445, 2176,  396,  123,    0},
```

FIG 19(1)

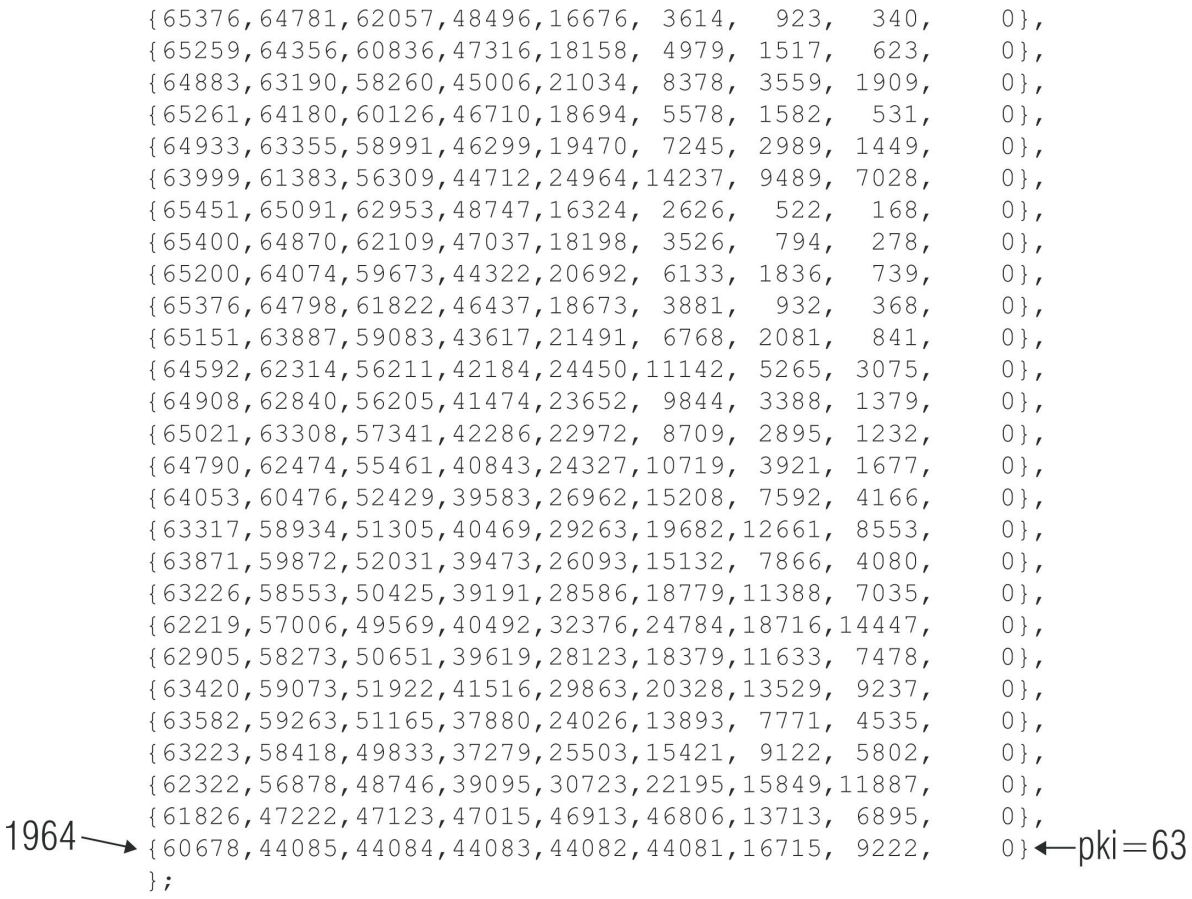

{65376,64781,62057,48496,16676, 3614, 923, 340, 0},
{65259,64356,60836,47316,18158, 4979, 1517, 623, 0},
{64883,63190,58260,45006,21034, 8378, 3559, 1909, 0},
{65261,64180,60126,46710,18694, 5578, 1582, 531, 0},
{64933,63355,58991,46299,19470, 7245, 2989, 1449, 0},
{63999,61383,56309,44712,24964,14237, 9489, 7028, 0},
{65451,65091,62953,48747,16324, 2626, 522, 168, 0},
{65400,64870,62109,47037,18198, 3526, 794, 278, 0},
{65200,64074,59673,44322,20692, 6133, 1836, 739, 0},
{65376,64798,61822,46437,18673, 3881, 932, 368, 0},
{65151,63887,59083,43617,21491, 6768, 2081, 841, 0},
{64592,62314,56211,42184,24450,11142, 5265, 3075, 0},
{64908,62840,56205,41474,23652, 9844, 3388, 1379, 0},
{65021,63308,57341,42286,22972, 8709, 2895, 1232, 0},
{64790,62474,55461,40843,24327,10719, 3921, 1677, 0},
{64053,60476,52429,39583,26962,15208, 7592, 4166, 0},
{63317,58934,51305,40469,29263,19682,12661, 8553, 0},
{63871,59872,52031,39473,26093,15132, 7866, 4080, 0},
{63226,58553,50425,39191,28586,18779,11388, 7035, 0},
{62219,57006,49569,40492,32376,24784,18716,14447, 0},
{62905,58273,50651,39619,28123,18379,11633, 7478, 0},
{63420,59073,51922,41516,29863,20328,13529, 9237, 0},
{63582,59263,51165,37880,24026,13893, 7771, 4535, 0},
{63223,58418,49833,37279,25503,15421, 9122, 5802, 0},
{62322,56878,48746,39095,30723,22195,15849,11887, 0},
{61826,47222,47123,47015,46913,46806,13713, 6895, 0},
1964
{60678,44085,44084,44083,44082,44081,16715, 9222, 0}
pki=63
};

FIG 19(2)

```
static unsigned long ari_s_hash[387] = {
0x0090D52E,0x0090CB1F,0x00A4CB2C,0x00003330,0x00107A07,0x00907A15,0x00207A0C,
0x00147A0C,
0x00247A15,0x00A07A19,0x00947A19,0x00A47A18,0x0010B70D,0x0090B719,0x0020B71D,
0x0014B71D,
0x00A0B718,0x0094B718,0x0024B715,0x00A4B717,0x0110B731,0x0000FE1C,0x0090FE1E,
0x00013B02,
0x00A5C92E,0x0095C92C,0x003A9E0B,0x00000B01,0x00BA9E2E,0x0090992B,0x0011B514,
0x00A5B52E,
0x0095B52B,0x0090D62C,0x0010850D,0x0014851B,0x00248514,0x0020850B,0x00908522,
0x00948520,
0x00A4852B,0x00A0851F,0x00003432,0x00107B08,0x00207B0E,0x00907B1D,0x00147B0E,
0x00247B0C,
0x00A07B14,0x00947B14,0x00A47B18,0x00910928,0x03D0713D,0x00D0713A,0x0000FF1C,
0x0090F52D,
0x0010F521,0x00013C01,0x03D05D3D,0x00A5CA2C,0x0025CA1F,0x0095CA2B,0x003A9F0B,
0x00000C02,
0x0021790B,0x0025791C,0x00917922,0x0015790B,0x00A17924,0x00A5792B,0x00957924,
0x00BA9F2E,
0x00CF2D38,0x00000200,0x0011B615,0x0091B618,0x0021B619,0x0015B619,0x00A1B617,
0x0095B617,
0x0025B618,0x00A5B62B,0x004FB125,0x00108609,0x00148616,0x0020860C,0x00908614,
0x0024861D,
0x00948619,0x00A08622,0x00A4861F,0x0090CD23,0x00003532,0x00010A2E,0x00002B25,
0x0010B90B,
0x0090B91F,0x00A4B92C,0x0001001E,0x03D0213C,0x0090F62D,0x0010F628,0x00A5CB2C,
0x0025CB1F,
0x0095CB2C,0x00000D03,0x00117A12,0x00217A1D,0x00917A14,0x00257A0B,0x00157A1D,
0x00A17A18,
0x00A57A17,0x00957A18,0x0011B70C,0x0091B722,0x0021B715,0x0015B714,0x00A1B71F,
0x0025B722,
0x0095B71F,0x00A5B717,0x004FB21C,0x0010870A,0x00148707,0x00208710,0x0090871D,
0x00248716,
0x00948714,0x00A08714,0x00A48718,0x00907D1C,0x00010B25,0x00002C22,0x0010BA1D,
0x0090BA22,
0x0014BA14,0x00A0BA17,0x0094BA17,0x00A4BA17,0x03D0693D,0x0090F72F,0x0010F728,
0x0025851C,
0x0015850B,0x00218514,0x00A5852C,0x00958524,0x00117B07,0x00217B12,0x00257B1B,
0x00917B15,
0x00157B16,0x00A17B18,0x00A57B1F,0x00957B18,0x0090D92C,0x004FB31C,0x03D0413C,
0x00907E1C,
0x0090C52B,0x00A4C52E,0x00002D14,0x00D02D38,0x03D02D3D,0x0010BB16,0x0090BB19,
0x0014BB15,
0x00A0BB1F,0x0094BB1F,0x00A4BB17,0x0025860B,0x0015861D,0x00218615,0x00918619,
0x00A58617,
0x00958618,0x00A1861F,0x00000F25,0x00CEE938,0x00004C32,0x0011B914,0x00A5B92E,
0x0095B92B,
0x0014890B,0x0024891C,0x00908918,0x00A4892B,0x00A08917,0x00907F14,0x0010C615,
0x0090C61F,
0x0014C614,0x0094C617,0x00A4C62B,0x00A0C617,0x00910D25,0x00107504,0x00907514,
0x00147516,
0x0020751D,0x00247514,0x00A0751F,0x00947518,0x00A47524,0x0090F925,0x0011870E,
0x0025871B,
0x0015870C,0x0021870C,0x00918714,0x00A5871F,0x00A18718,0x00958718,0x03D06139,
0x0000101C,
0x03D04D3D,0x0011BA15,0x0091BA18,0x0021BA22,0x00A1BA17,0x0025BA1F,0x00A5BA2C,
0x0095BA17,
0x00148A1D,0x00208A1D,0x00248A15,0x00908A19,0x00948A18,0x00A08A18,0x00A48A17,
0x0010393D,
```

FIG 20(1)

```
0x0010C70C,0x0090C718,0x0014C715,0x0094C717,0x00A0C717,0x00A4C72B,0x00010E25,
0x00002F2D,
0x00107605,0x00907615,0x00147610,0x00207612,0x0024761D,0x00A07614,0x00947614,
0x00A47618,
0x0110762F,0x0090BD1C,0x00CDD938,0x0000FA02,0x0090FA2E,0x0000110B,0x03D0113B,
0x00A5C52E,
0x0095C52C,0x0011BB1D,0x0091BB18,0x0021BB19,0x0014950B,0x00A1BB17,0x0025BB18,
0x00A5BB2C,
0x0095BB17,0x00909523,0x00108B0E,0x00148B16,0x00208B0C,0x00248B1D,0x00908B14,
0x00948B18,
0x00A08B18,0x00A48B17,0x00010F25,0x0000302B,0x00107706,0x00907716,0x00147711,
0x00207713,
0x0024770E,0x00A07715,0x0094771D,0x00A47714,0x0091052F,0x00110528,0x0090BE1C,
0x0015052D,
0x03D06D3D,0x0000FB01,0x0090FB2D,0x0025890B,0x00A5892C,0x00158914,0x0021891C,
0x0095892B,
0x0011C619,0x0025C61F,0x00A5C62C,0x0095C617,0x00117516,0x0021750B,0x0015750B,
0x00917519,
0x0025751C,0x00A17524,0x00957520,0x00A5752C,0x0014961D,0x0090961F,0x0090C917,
0x00A4C92E,
0x0000312B,0x00D0313A,0x03D0313D,0x0090BF1C,0x0091062F,0x0010B50B,0x0090B517,
0x00A0B52B,
0x00A4B52C,0x0000FC1C,0x00258A1D,0x00A58A17,0x00158A15,0x00218A15,0x00918A18,
0x00A18A17,
0x00958A17,0x00013938,0x00001327,0x0011C715,0x0025C718,0x00A5C72C,0x0095C717,
0x0011760E,
0x0021760C,0x00917615,0x0015760C,0x0025760B,0x00A17622,0x00957622,0x00A57617,
0x00005031,
0x00908D24,0x0090CA1F,0x00A4CA2C,0x0010790D,0x00907919,0x0020791D,0x0014791D,
0x00247914,
0x00A0791F,0x0094791F,0x00A4792B,0x00CE2D38,0x0010B60C,0x0090B618,0x0014B615,
0x0020B615,
0x00A0B61F,0x0094B61F,0x0024B619,0x00A4B617,0x0110B632,0x00258B1B,0x00158B1D,
0x00218B1D,
0x00A58B17,0x00918B18,0x00A18B17,0x00958B1F,0x0090FD1E,0x00013A04,0x0000142F,
0x003A9D1C,
0x00BA9D2D,0x0011770F,0x0021770E,0x00157710,0x0091771D,0x0025771B,0x00A17714,
0x00957714,
0x00A57718,0x00005131,0x03D0513C
};
```

FIG 20(2)