República Federativa do Brasil
Ministério da Economia
Instituto Nacional da Propriedade Industrial

(11) BR 112012017258-1 B1

(22) Data do Depósito: 11/01/2011

(45) Data de Concessão: 29/12/2020

(54) Título: CODIFICADOR DE ÁUDIO, DECODIFICADOR DE ÁUDIO, MÉTODO DE CODIFICAÇÃO E DECODIFICAÇÃO DE UMA INFORMAÇÃO DE ÁUDIO, PARA OBTENÇÃO DE UM VALOR DE SUB-REGIÃO DE CONTEXTO COM BASE EM UMA NORMA DE VALORES ESPECTRAIS PREVIAMENTE DECODIFICADOS

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

(30) Prioridade Unionista: 12/01/2010 US 61/294,357.

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

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

(86) Pedido PCT: PCT EP2011050275 de 11/01/2011

(87) Publicação PCT: WO 2011/086067 de 21/07/2011

(85) Data do Início da Fase Nacional: 12/07/2012

(57) Resumo: CODIFICADOR DE ÁUDIO, DECODIFICADOR DE ÁUDIO, MÉTODO DE CODIFICA-ÇÃO E DECODIFICAÇÃO DE UMA INFORMAÇÃO DE ÁUDIO, E PROGRAMA DE COMPUTADOR PARA OBTENÇÃO DE UM VALOR DE SUB-REGIÃO DE CONTEX-TO COM BASE EM UMA NORMA DE VALORES ESPECTRAIS PREVIAMENTE DE-CODIFICADOS Um decodificador de áudio para prover uma informação de áudio decodificada com base em uma informação de áudio codificada compreende um decodificador aritmético para prover diversos valores espectrais decodificados com base em uma represen-tação aritmeticamente codificada dos valores espectrais e um conversor de domínio de frequência em domínio de tempo para prover uma representação de áudio de domínio de tempo utilizando os valores espectrais decodificados para obter a informa-ção de áudio decodificada. 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 dependendo de um estado de contexto descrito por um valor de contexto corrente numérico.

CODIFICADOR DE ÁUDIO, DECODIFICADOR DE ÁUDIO, MÉTODO DE CODIFICAÇÃO E DECODIFICAÇÃO DE UMA INFORMAÇÃO DE ÁUDIO, PARA OBTENÇÃO DE UM VALOR DE SUB-REGIÃO DE CONTEXTO COM BASE EM UMA NORMA DE VALORES ESPECTRAIS PREVIAMENTE DECODIFICADOS

CAMPO TÉCNICO

1. As realizações de acordo com a invenção estão relacionadas a um decodificador de áudio para prover uma informação de áudio decodificada 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 decodificada 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.

2. As realizações de acordo com a invenção estão relacionadas a uma codificação espectral sem ruído aprimorada, que pode ser utilizada em um codificador ou decodificador de áudio, como, por exemplo, um assim chamado codificador de fala e áudio unificado (USAC).

HISTÓRICO DA INVENÇÃO

3.A seguir, o histórico da invenção será explicado resumidamente a fim de facilitar a compreensão da invenção e suas vantagens. Durante a última década, grandes esforços foram envidados para criar a possibilidade de armazenar e distribuir digitalmente o conteúdo de áudio com boa eficiência da taxa de bits. Uma realização importante desta maneira é a definição da Norma Internacional ISO/IEC 14496-3. A Parte 3 desta Norma se refere a uma codificação e decodificação de conteúdo de áudio, e a subparte

4 da parte 3 se refere à codificação de áudiogeral. A subparte 4 da ISO/IEC 14496 parte 3 define um conceito para codificação e decodificação de conteúdo de áudio geral. Além disso, outros aperfeiçoamentos foram propostos a fim de melhorar a qualidade e/ou reduzir a taxa de bitsexigida.

4. De acordo com o conceito descrito na dita Norma, um sinal de áudio de domínio de tempo é convertido em uma representação de tempo-frequência. A transformação do domínio de tempo em domínio de tempo-frequência é geralmente realizada utilizando blocos de transformação, que também são denominados “quadros”, de amostras de domínio de tempo. Descobriu-se que é vantajoso utilizar quadros de sobreposição, que são mudados, por exemplo, por metade de um quadro, uma vez que a sobreposição permite evitar eficientemente (ou pelo menos reduzir) os artefatos. Ainda, descobriu-se que um janelamento deve ser realizado para evitar os artefatos originários deste processamento de quadrostemporariamente limitados.

5. Pela transformação de uma parte em janela do sinal de áudio de entrada do domínio de tempo em domínio da frequência de tempo, uma compactação de energia é obtida em muitos casos, de modo que alguns dos valores espectrais compreendam uma magnitude significantemente maior que diversos outros valores espectrais. Assim, há, em muitos casos, um número comparavelmente pequeno de valores espectrais tendo uma magnitude, que está significantemente acima de uma magnitude média dos valores espectrais. Um exemplo típico de um domínio de tempo para transformação de domínio de frequência em domínio de tempo resultante em uma compactação de energia é chamada transformação

de cosseno discreto modificado (MDCT).

6. Os valores espectrais são geralmente escalados e quantizados de acordo com um modelo psicoacústico, de modo que os erros de quantização sejam comparavelmente menores para valores espectrais psicoacusticamente mais importantes, e são comparavelmente maiores para valores espectrais psicoacusticamente menos importantes. Os valores espectrais escalados e quantizados são codificados para prover uma representação eficiente de taxa de bits destes.

7. Por exemplo, o uso da assim chamada codificação de Huffman de coeficientes espectrais quantizados é descrito no Padrão Internacional ISO/IEC 14496-3:2005(E), parte 3, subparte 4.

8. Entretanto, foi observado que a qualidade da codificação dos valores espectrais tem um impacto significativo na taxa de bits necessária. Ainda, foi observado que a complexidade de um decodificador de áudio, que é geralmente implementado em um dispositivo do consumidor portátil, e que deve ser então criado de forma barata e com baixo consumo de energia, é dependente da codificação utilizada na codificação dos valores espectrais.

9. Em vista desta situação, há uma necessidade de um conceito para codificar e decodificar um conteúdo de áudio, que provê uma mudança melhorada entre eficiência da taxa de bits e eficiência do recurso.

SUMÁRIO DA INVENÇÃO

10. Uma realização, de acordo com a invenção, cria um decodificador de áudio para prover uma informação de áudio decodificada com base em uma informação de áudio codificada. O decodificador de áudio compreende um decodificador aritmético para

prover diversos 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 em domínio de tempo para prover uma representação de áudio de domínio de tempo que utiliza os valores espectrais decodificados, a fim de obter a informação de áudio decodificada. 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 (cujo código de símbolo tipicamente descreve um valor espectral ou diversos valores espectrais ou um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais) dependendo de um estado de contexto descrito por um valor de contexto corrente numérico. O decodificador aritmético é configurado para determinar o valor de contexto corrente numérico dependendo de diversos valores espectrais previamente decodificados. O decodificador aritmético é também configurado para obter diversos valores de sub-região de contexto com base em valores espectrais previamente decodificados e para armazenar os ditos valores de sub-região de contexto. O decodificador aritmético é configurado para derivar um valor de contexto corrente numérico associado a um ou mais valores espectrais a serem decodificados (ou, mais precisamente, definindo um contexto para a decodificação do um ou mais valores espectrais a serem decodificados) dependendo dos valores de sub-região de contexto armazenados. O decodificador aritmético é configurado para computar a norma de um vetor formado por diversos valores espectrais previamente decodificados para obter um valor de sub-região de contexto comum associado aos diversos valores espectrais

previamente decodificados.

11. Esta realização da invenção tem como base o achado de que uma informação de sub-região de contexto de memória eficiente pode ser obtida pela computação da norma de um vetor formado por diversos valores espectrais previamente decodificados, uma vez que a norma deste vetor formado por diversos valores espectrais previamente decodificados compreende a informação de contexto mais relevante. Ao formar uma norma, os sinais dos valores espectrais são geralmente descartados. No entanto, descobriu-se que os sinais dos valores espectrais compreendem somente um impacto subordinado sobre o estado do contexto, caso exista, e, portanto, pode ser omitido sem comprometer gravemente a significância do valor de sub-região de contexto. Além disso, foi descoberto que a formação de uma norma de um vetor formado por diversos valores espectrais previamente decodificados, que geralmente traz consigo um efeito de ponderação, permite uma redução de uma quantidade de informação, enquanto ainda resulta em um valor de contexto que reflete a situação de contexto corrente com precisão suficiente. Resumindo, uma exigência de memória para armazenar o contexto na forma de diversos valores de sub-região de contexto pode ser mantida pequena pelo armazenamento de valores de sub-região de contexto que são baseados em uma computação da norma de um vetor formado por diversos valores espectrais previamente decodificados (em vez dos próprios valores espectrais).

12. Em uma realização preferida, o decodificador aritmético é configurado para somar valores absolutos de diversos valores espectrais previamente decodificados, que estão, preferencialmente, porém não necessariamente, associados a bins de

frequência adjacentes do conversor de domínio de frequência em domínio de tempo e a uma parte temporal comum da informação de áudio, para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados. Descobriu-se que a soma dos valores absolutos de diversos valores espectrais previamente decodificados, correspondentes a uma computação de norma, é uma forma particularmente eficiente de computar valores de sub-região de contexto significativos. Deve ser aqui observado que a computação da soma de valores absolutos de um vetor é igual à computação de uma assim chamada norma L-1 do vetor. Em outras palavras, a computação da soma de valores absolutos de um vetor é um exemplo de uma computação de uma norma.

13. Em uma realização preferida, o decodificador aritmético é configurado para quantizar a norma de diversos valores espectrais previamente decodificados, que estão associados a bins de frequência adjacentes do conversor de domínio de frequência em domínio de tempo e a uma parte temporal comum da informação de áudio, para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados. A quantização da norma pode, por exemplo, compreender a computação da norma em uma escala discreta (por exemplo, uma soma de valores inteiros absolutos) e também limitar o resultado.

14. Em uma realização preferida, o decodificador aritmético é configurado para quantizar a norma de diversos valores espectrais previamente decodificados, que estão, preferencialmente, porém não necessariamente, associados a bins de

frequência adjacentes do conversor de domínio de frequência em domínio de tempo e a uma parte temporal comum da informação de áudio, para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados. Descobriu-se que a quantização da dita norma pode ajudar a manter a quantidade de informação razoavelmente pequena. Por exemplo, a quantização pode ajudar a reduzir o número de bits necessários para uma representação do valor de sub-região de contexto e, portanto, pode facilitar a provisão de um valor de contexto corrente numérico tendo um número pequeno de bits.

15. Em uma realização preferida, o decodificador aritmético é configurado para somar valores absolutos de valores espectrais previamente decodificados, que são codificados utilizando um valor de código comum, para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados. Descobriu-se que a precisão do contexto é particularmente alta se um valor de sub-região de contexto comum for formado para tais valores espectrais que são codificados utilizando um valor de código comum. Assim, cada valor de sub-região de contexto pode corresponder a um valor de código que, por sua vez, traz consigo uma boa eficiência de memória ao armazenar o valor de sub-região de contexto.

16. Em uma realização preferida, o decodificador aritmético é configurado para prover valores espectrais discretos decodificados sinalizados para o conversor de domínio de frequência em domínio de tempo, e para somar valores absolutos correspondentes aos valores espectrais decodificados sinalizados para obter o valor de sub-região de contexto comum associado aos

diversos valores espectrais previamente decodificados. Descobriu-se que é algumas vezes benéfico, em termos de qualidade do áudio, ter valores sinalizados como valores de entrada para o conversor de domínio de frequência em domínio de tempo, uma vez que isto permite considerar fases na reconstrução do conteúdo de áudio. No entanto, descobriu-se também que a omissão da informação de fase (ou seja, da informação de sinal sobre os valores espectrais) nos valores de sub-região de contexto não prejudica gravemente a precisão da informação de estado de contexto derivada utilizando os valores de sub-região de contexto, uma vez que a informação de fase, na maioria dos casos, não está fortemente correlacionada entre diferentes bins de frequência.

17. Em uma realização preferida, o decodificador aritmético é configurado para derivar um valor de soma limitado a partir de uma soma de valores absolutos de valores espectrais discretos previamente decodificados (ou para derivar um valor de norma limitado a partir de uma norma de um vetor formado por diversos valores espectrais discretos previamente decodificados), de modo que uma faixa de possíveis valores para o valor de soma limitado seja menor que uma faixa de possíveis valores de soma (ou de tal modo que uma faixa de possíveis valores para o valor de norma limitado seja menor que uma faixa de possíveis valores de norma). Descobriu-se que a limitação dos valores de sub-região de contexto permite reduzir diversos bits necessários para armazenar os valores de sub-região de contexto. Descobriu-se também que uma limitação razoável dos valores de sub-região de contexto não resulta em uma perda significativa de informação, uma vez que, para valores espectrais maiores que um determinado limiar, o

contexto não mais se altera significativamente.

18. Em uma realização preferida, o decodificador aritmético é configurado para obter um valor de contexto corrente numérico dependendo de diversos valores de sub-região de contexto associados a diferentes conjuntos de valores espectrais previamente decodificados. Este conceito permite considerar eficientemente diferentes contextos para a decodificação de diferentes valores espectrais (ou tuplos de valores espectrais). Mantendo-se uma granularidade suficientemente fina dos valores de sub-região de contexto, de modo que diversos valores de sub-região de contexto sejam utilizados para obter um valor único de contexto corrente numérico, é possível armazenar uma informação de sub-região de contexto significativa ainda que universalmente utilizável, a partir da qual o real valor de contexto numérico pode ser derivado pouco antes da decodificação de um valor espectral (ou um tuplo de valores espectrais) a ser decodificado.

19. Em uma realização preferida, o decodificador aritmético é configurado para obter uma representação numérica de um valor de contexto corrente numérico, de modo que uma primeira parte da representação numérica do valor de contexto corrente numérico seja determinada por um primeiro valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados (ou, de forma mais geral, um primeiro valor de norma ou valor de norma limitado), e de modo que uma segunda parte da representação numérica do valor de contexto corrente numérico seja determinada por um segundo valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados (ou, de forma mais geral, um

segundo valor de norma ou valor de norma limitado). Descobriu-se que é possível aplicar eficientemente os valores de sub-região de contexto na derivação de um valor de contexto corrente numérico. Em particular, descobriu-se que os valores de sub-região de contexto computados conforme discutido acima são bem adequados para compor um valor de contexto corrente numérico. Descobriu-se que os valores de sub-região de contexto computados conforme discutido acima são bem adequados para determinar diferentes partes de uma representação numérica do valor de contexto corrente numérico. Assim, tanto uma computação eficiente dos valores de sub-região de contexto como uma derivação eficiente ou atualização do valor de contexto corrente numérico pode ser alcançada.

20. Em uma realização preferida, o decodificador aritmético é configurado para obter o valor de contexto corrente numérico, de modo que um primeiro valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados (ou um primeiro valor de norma ou valor de norma limitado) e um segundo valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados (ou um segundo valor de norma ou valor de norma limitado) compreendam diferentes pesos no valor de contexto corrente numérico. Assim, as diferentes distâncias dos valores espectrais, nas quais os valores de sub-região de contexto são baseadas, a partir do um ou mais valores espectrais a serem atualmente decodificados, podem ser levadas em consideração. Alternativamente, uma posição relativa diferente entre os valores espectrais, na qual os valores de sub-região de contexto são baseados, e o um ou mais valores espectrais para serem atualmente

decodificados, pode ser levada em consideração aplicando-se diferentes pesos numéricos no valor de contexto corrente numérico. Também, uma atualização iterativa do valor de contexto corrente numérico pode ser facilitada por meio deste conceito, uma vez que os pesos numéricos de partes de uma representação numérica podem ser facilmente alterados aplicando-se uma operação de mudança.

21. Em uma realização preferida, o decodificador aritmético é configurado para modificar uma representação numérica de um valor de contexto prévio numérico, que descreve um estado de contexto associado a um ou mais valores espectrais previamente decodificados, dependendo de um valor de soma ou um valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados (ou um valor de norma ou valor de norma limitado), para obter uma representação numérica de um valor de contexto corrente numérico que descreve um estado de contexto associado a um ou mais valores espectrais a serem decodificados. Desta forma, uma atualização particularmente eficiente do valor de contexto corrente numérico pode ser obtida, em que uma recomputação completa do valor de contexto corrente numérico é evitada.

22. Em uma realização preferida, o decodificador aritmético é configurado para verificar se uma soma de diversos valores de sub-região de contexto é menor ou igual a um valor limiar de soma predeterminado, e para seletivamente modificar o valor de contexto corrente numérico dependendo de um resultado da verificação, em que cada um dos valores de sub-região de contexto é um valor de soma ou um valor de soma limitado de valores absolutos de uma pluralidade associada de valores espectrais

previamente decodificados (ou um valor de norma ou valor de norma limitado). Assim, a presença de uma região ampliada de valores espectrais comparativamente pequenos pode ser detectada e o resultado da detecção pode ser aplicado para uma adaptação do contexto. Por exemplo, pode-se concluir a partir da presença desta região ampliada de valores espectrais comparativamente pequenos que há alta probabilidade de que o valor espectral a ser decodificado utilizando o valor de contexto corrente numérico também seja comparativamente pequeno. Assim, o contexto pode ser adaptado de uma forma particularmente eficiente.

23. Em uma realização preferida, o decodificador aritmético é configurado para considerar diversos valores de sub-região de contexto definidos por valores espectrais previamente decodificados associados a uma parte temporal prévia do conteúdo de áudio, e também para considerar pelo menos um valor de sub-região de contexto definido por valores espectrais previamente decodificados associados a uma parte temporal corrente do conteúdo de áudio, para obter um valor de contexto corrente numérico associado a um ou mais valores espectrais a serem decodificados e associados à parte temporal corrente do conteúdo de áudio, de modo que um ambiente tanto de valores espectrais previamente decodificados temporariamente adjacentes da parte temporal prévia como os valores espectrais previamente decodificados adjacentes à frequência da parte temporal corrente sejam considerados para obter o valor de contexto corrente numérico. Assim, um contexto particularmente significativo pode ser obtido. Também, deve ser observado que a derivação acima descrita dos valores de sub-região de contexto mantém razoavelmente pequenas as exigências de memória

para armazenar os valores de sub-região de contexto da parte temporal prévia.

24. Em uma realização preferida, o decodificador aritmético é configurado para armazenar um conjunto de valores de sub-região de contexto, sendo cada um dos valores de sub-região de contexto baseado em um valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados (ou, de forma mais geral, um valor de norma de um vetor formado por diversos valores espectrais previamente decodificados), para uma determinada parte temporal da informação de áudio, e para utilizar os valores de sub-região de contexto para derivar um valor de contexto corrente numérico para decodificação de um ou mais valores espectrais de uma parte temporal da informação de áudio seguindo a determinada parte temporal da informação de áudio enquanto deixa os valores espectrais individuais previamente decodificados para a determinada parte temporal da informação de áudio não considerada ao derivar o valor de contexto corrente numérico. Assim, a eficiência na computação do valor de contexto corrente numérico pode ser aumentada. Também, não é mais necessário armazenar os valores espectrais individuais previamente decodificados por um período prolongado.

25. Em uma realização preferida, o decodificador aritmético é configurado para decodificar separadamente um valor de magnitude e um sinal de um valor espectral. Neste caso, o decodificador aritmético é configurado para deixar sinais de valores espectrais previamente decodificados não considerados ao determinar o valor de contexto corrente numérico para a

decodificação de um valor espectral a ser decodificado. Descobriu-se que esta manipulação separada do valor absoluto e do sinal de um valor espectral não resulta em uma grave degradação da eficiência da codificação, mas reduz significativamente a complexidade computacional. Além disso, descobriu-se que a computação dos valores de sub-região de contexto com base na computação de uma norma de um vetor formado por diversos valores espectrais previamente decodificados é bem adaptada para uso em combinação com este conceito.

26. Uma realização da 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 tempo em domínio de frequência com compactação de energia 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 compreende um codificador aritmético configurado para codificar um valor espectral ou uma versão pré-processada deste, ou, equivalentemente, diversos valores espectrais ou uma versão pré-processada deste, utilizando uma senha de comprimento 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, ou, equivalentemente, diversos valores espectrais ou um valor de um plano de bits mais significativo de diversos valores espectrais 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, dependendo de um estado de contexto descrito por um valor de contexto corrente numérico. O codificador aritmético é configurado para determinar o valor de contexto corrente numérico dependendo de diversos valores espectrais previamente codificados. O codificador aritmético é configurado para obter diversos valores de sub-região de contexto com base em valores espectrais previamente codificados, para armazenar os ditos valores de sub-região de contexto e para derivar um valor de contexto corrente numérico associado a um ou mais valores espectrais a serem codificados (ou, mais precisamente, que definem um contexto para codificação dos valores espectrais a serem codificados), dependendo dos valores de sub-região de contexto armazenados. O codificador aritmético é configurado para computar a norma de um vetor formado por diversos valores espectrais previamente codificados, para obter um valor de sub-região de contexto comum associado aos diversos valores espectrais previamente codificados.

27. O dito codificador de áudio é baseado no mesmo *timing* do decodificador de áudio acima descrito. Também, o dito codificador de áudio pode ser complementado por qualquer uma das características e funcionalidades descritas acima com relação ao decodificador de áudio.

28. Outra realização de acordo com a invenção cria um método para prover uma informação de áudio decodificada com base em uma informação de áudio codificada.

29. Outra realização de acordo com a invenção cria um

método para prover uma informação de áudio codificada com base em uma informação de áudio de entrada.

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

BREVE DESCRIÇÃO DAS FIGURAS

31. As realizações, de acordo com a presente invenção, serão subsequentemente descritas tendo referência às figuras anexas, nas quais:

32. A figura 1 mostra um diagrama em blocos esquemático de um codificador de áudio, de acordo com uma realização da invenção;

33. A figura 2 mostra um diagrama em blocos esquemático de um decodificador de áudio, de acordo com uma realização da invenção;

34. A figura 3 mostra uma representação em código do pseudo-programa de um algoritmo “values_decode()” de decodificação valores espectrais;

35. A figura 4 mostra uma representação esquemática de um contexto para um cálculo de estado;

36. A figura 5a mostra uma representação em código do pseudo-programa de um algoritmo “arith_map_context()” para mapeamento de um contexto;

37. A figura 5b mostra uma representação em código do pseudo-programa de outro algoritmo “arith_map_context()” para mapeamento de um contexto;

38. A figura 5c mostra uma representação em código do pseudo-programa de um algoritmo “arith_get_context()” para obter um valor de estado de contexto;

39. A figura 5d mostra uma representação em código do pseudo-programa de outro algoritmo “arith_get_context()” para obter um valor de estado de contexto;

40. A figura 5e mostra uma representação em código do pseudo-programa de um algoritmo “arith_get_pk()” para derivar um valor do índice da tabela de frequências cumulativas “pki” de um valor de estado (ou uma variável de estado);

41. A figura 5f mostra uma representação em código do pseudo-programa de outro algoritmo “arith_get_pk()” para derivar um valor do índice da tabela de frequências cumulativas “pki” de um valor de estado (ou uma variável de estado);

42. A figura 5g mostra uma representação em código do pseudo-programa de um algoritmo “arith_decode()” para aritmeticamente decodificar um símbolo de uma senha de comprimento variável;

43. A figura 5h mostra uma primeira parte de uma representação em código do pseudo-programa de outro algoritmo “arith_decode()” para aritmeticamente decodificar um símbolo de uma senha de comprimento variável;

44. A figura 5i mostra uma segunda parte de uma representação em código do pseudo-programa de outro algoritmo “arith_decode()” para aritmeticamente decodificar um símbolo de uma senha de comprimento variável;

45. A figura 5j mostra uma representação em código do pseudo-programa de um algoritmo para derivar valores absolutos a,b de valores espectrais de um valor comum m;

46. A figura 5k mostra uma representação em código do pseudo-programa de um algoritmo para inserir os valores

decodificados a,b em uma matriz de valores espectrais decodificados;

47. A figura 5l mostra uma representação em código do pseudo-programa de um algoritmo “arith_update_context()” para obter um valor de sub-região de contexto com base em valores absolutos a,b de valores espectrais decodificados;

48. A figura 5m mostra uma representação em código do pseudo-programa de um algoritmo “arith_finish()” para preencher as entradas de uma matriz de valores espectrais decodificados e uma matriz de valores de sub-região de contexto;

49. A figura 5n mostra uma representação em código do pseudo-programa de outro algoritmo para derivar valores absolutos a,b de valores espectrais decodificados de um valor comum m;

50. A figura 5o mostra uma representação em código do pseudo-programa de um algoritmo “arith_update_context()” para atualizar uma matriz de valores espectrais decodificados e uma matriz de valores de sub-região de contexto;

51. A figura 5p mostra uma representação em código do pseudo-programa de um algoritmo “arith_save_context()” para preencher entradas de uma matriz de valores espectrais decodificados e entradas de uma matriz de valores de sub-região de contexto;

52. A figura 5q mostra uma legenda das definições;

53. A figura 5r mostra outra legenda das definições;

54. A figura 6a mostra uma representação de sintaxe de um bloco de dados brutos da codificação de áudio e voz unificada (USAC);

55. A figura 6b mostra uma representação de sintaxe

de um único elemento de canal;

56. A figura 6c mostra uma representação de sintaxe de um elemento do par de canal;

57. A figura 6d mostra uma representação de sintaxe de uma informação de controle “ICS”;

58. A figura 6e mostra uma representação de sintaxe de um fluxo de canal de domínio de frequência;

59. A figura 6f mostra uma representação de sintaxe de dados espectrais aritmeticamente codificados;

60. A figura 6g mostra uma representação de sintaxe de decodificação de um conjunto de valores espectrais;

61. A figura 6h mostra outra representação de sintaxe de decodificação de um conjunto de valores espectrais;

62. A figura 6i mostra uma legenda dos elementos de dados e variáveis;

63. A figura 6j mostra outra legenda de elementos de dados e variáveis;

64. A figura 7 mostra um diagrama em blocos esquemático de um codificador de áudio, de acordo com o primeiro aspecto da invenção;

65. A figura 8 mostra um diagrama em blocos esquemático de um decodificador de áudio, de acordo com o primeiro aspecto da invenção;

66. A figura 9 mostra uma representação gráfica de um mapeamento de um valor de contexto corrente numérico em um valor de índice de regra de mapeamento, de acordo com o primeiro aspecto da invenção;

67. A figura 10 mostra um diagrama em blocos

esquemático de um codificador de áudio, de acordo com um segundo aspecto da invenção;

68. A figura 11 mostra um diagrama em blocos esquemático de um decodificador de áudio, de acordo com o segundo aspecto da invenção;

69. A figura 12 mostra um diagrama em blocos esquemático de um codificador de áudio, de acordo com um terceiro aspecto da invenção;

70. A figura 13 mostra um diagrama em blocos esquemático de um decodificador de áudio, de acordo com o terceiro aspecto da invenção;

71. A figura 14a mostra uma representação esquemática de um contexto para um cálculo de estado, como é utilizado de acordo com o projeto de trabalho 4 do Padrão de Projeto do Padrão de Projeto USAC;

72. A figura 14b mostra uma visão geral das tabelas conforme utilizadas no esquema de codificação aritmético de acordo com o projeto de trabalho 4 do Padrão de Projeto USAC;

73. A figura 15a mostra uma representação esquemática de um contexto para um cálculo de estado, como é utilizado nas realizações de acordo com a invenção;

74. A figura 15b mostra uma visão geral das tabelas conforme utilizadas no esquema de codificação aritmético de acordo com a presente invenção;

75. A figura 16a mostra uma representação gráfica de uma demanda de memória para somente leitura para o esquema de codificação silenciosa, de acordo com a presente invenção, e de acordo com o projeto de trabalho 5 do Padrão de Projeto USAC e de

acordo com a Codificação de Huffman AAC (codificação de áudio avançado);

76. A figura 16b mostra uma representação gráfica de uma demanda de memória para 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 5 do Padrão de Projeto USAC;

77. A figura 17 mostra uma representação esquemática de uma disposição para uma comparação de uma codificação silenciosa de acordo com o projeto de trabalho 3 ou projeto de trabalho 5 do Padrão de Projeto USAC com um esquema de codificação de acordo com a presente invenção;

78. A figura 18 mostra uma representação em tabela das taxas de bits médios produzidos por um codificador aritmético de USAC de acordo com o projeto de trabalho 3 do Padrão de Projeto USAC e de acordo com uma realização da presente invenção;

79. A figura 19 mostra uma representação em tabela de níveis mínimo e máximo do reservatório de bits para um decodificador aritmético de acordo com o projeto de trabalho 3 do Padrão de Projeto USAC e para um decodificador aritmético de acordo com uma realização da presente invenção;

80. A figura 20 mostra uma representação em tabela dos números da complexidade média de decodificação de um fluxo de bits 32-kbits de acordo com o projeto de trabalho 3 do Padrão de Projeto USAC para diferentes versões do codificador aritmético;

81. A figuras 21(1) e 21(2) mostram uma representação em tabela de um conteúdo da tabela “ari_lookup_m[600]”;

82. As figuras 22(1) a 22(4) mostram uma

representação em tabela de um conteúdo da tabela “ari_hash_m[600]”;

83. As figuras 23(1) a 23(7) mostram uma representação em tabela de um conteúdo da tabela “ari_cf_m[96][17]”; e

84. A figura 24 mostra uma representação em tabela de um conteúdo da tabela “ari_cf_r[]”.

DESCRIÇÃO DETALHADA DAS REALIZAÇÕES

Codificador de áudio de acordo com a figura 7

85. A figura 7 mostra um diagrama em 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 prover, com base neste, uma informação de áudio codificada 712. O codificador de áudio compreende um conversor de domínio de tempo em domínio de frequência com 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 deste, que utiliza uma senha de comprimento variável a fim de obter a informação de áudio codificada 712 (que pode compreender, por exemplo, diversas senhas de comprimento variável).

86. 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 (ou seja, em uma senha de comprimento variável) dependendo de um estado de contexto. 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, dependendo de um estado de contexto (corrente). O codificador aritmético é configurado para determinar o estado de contexto corrente, ou um valor de contexto corrente numérico que descreve o estado de contexto corrente, dependendo de diversos valores espectrais previamente codificados (preferivelmente, mas não necessariamente, adjacentes). Para esta finalidade, o codificador aritmético é configurado para avaliar a tabela *hash*, cujas entradas definem tanto valores numéricos de estado significativos entre os valores de contexto numéricos como limites de intervalos de valores de valores de contexto numéricos, em que um valor de índice de regra de mapeamento está individualmente associado a um valor de contexto (corrente) numérico sendo um valor de estado significativo, e em que um valor de índice de regra de mapeamento comum está associado a diferentes valores de contexto (corrente) numéricos encontrando-se dentro de um intervalo delimitado por limites de intervalo (em que os limites de intervalo são preferivelmente definidos pelas entradas da tabela *hash*).

87. Como pode ser visto, o mapeamento de um valor espectral (da representação de áudio de domínio de frequência 722), ou de um plano de bits mais significativo de um valor

espectral, em um valor de código (da informação de áudio codificada 712), pode ser realizado por uma codificação do valor espectral 740 que utiliza uma regra de mapeamento 742. Um rastreador de estado 750 pode ser configurado para rastrear o estado de contexto. O rastreador de estado 750 provê uma informação 754 que descreve o estado de contexto corrente. A informação 754 que descreve o estado de contexto corrente pode preferivelmente tomar a forma de um valor de contexto corrente numérico. Um selecionador da regra de mapeamento 760 é configurado para 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. Assim, um selecionador da regra de mapeamento 760 provê a informação da regra de mapeamento 742 à codificação do valor espectral 740. A informação da regra de mapeamento 742 pode tomar a forma de um valor de índice de regra de mapeamento ou da tabela de frequências cumulativas selecionadas dependendo de um valor de índice de regra de mapeamento. O selecionador da regra de mapeamento 760 compreende (ou pelo menos avalia) a tabela *hash* 752, cujas entradas definem ambos os valores de estado significativos entre os valores de contexto numéricos e limites e intervalos de valores de contexto numéricos, em que um valor de índice de regra de mapeamento está individualmente associado a um valor de contexto numérico sendo um valor de estado significativo, e em que um valor de índice de regra de mapeamento comum está associado a diferentes valores de contexto numéricos encontrando-se dentro de um intervalo delimitado por limites de intervalo. A tabela *hash* 762 é avaliada a fim de selecionar a

regra de mapeamento, ou seja, a fim de prover a informação da regra de mapeamento 742.

88. Para resumir o mencionado 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 em domínio de frequência. A codificação aritmética é dependente de contexto, de modo que uma regra de mapeamento (por exemplo, a tabela de frequências cumulativas) seja selecionada dependendo dos valores espectrais previamente codificados. Assim, os valores espectrais adjacentes em tempo e/ou frequência (ou, pelo menos, dentro de um ambiente predeterminado) um ao outro e/ou ao valor espectral atualmente codificado (ou seja, 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 apropriada, valores de contexto corrente numérico 754 providos por um rastreador de estado 750 são avaliados. Como tipicamente o número de diferentes regras de mapeamento é significantemente menor que o número de possíveis valores dos valores de contexto corrente numérico 754, o selecionador da regra de mapeamento 760 aloca as mesmas regras de mapeamento (descritas, por exemplo, por um valor de índice de regra de mapeamento) a um número comparavelmente grande de diferentes valores de contexto numéricos. Entretanto, há configurações espectrais tipicamente específicas (representadas por valores de contexto numéricos específicos) nos quais uma regra de mapeamento particular deve ser associada a fim de obter uma boa

eficiência de codificação.

89. Foi observado que a seleção de uma regra de mapeamento dependendo de um valor de contexto corrente numérico pode ser realizada com alta eficiência computacional, particularmente, se as entradas de uma única tabela *hash* definem tanto os valores de estado significativos como os limites de intervalos de valores de contexto (corrente) numéricos. Foi observado que este mecanismo é bem adaptado às exigências da seleção da regra de mapeamento, pois há muitos casos onde um único valor de estado significativo (ou valor de contexto numérico significativo) é embutido entre um intervalo esquerdo de diversos valores de estado não-significativos (nos quais uma regra de mapeamento comum está associada) e um intervalo direito de diversos valores de estado não-significativos (nos quais uma regra de mapeamento comum está associada). Ainda, o mecanismo da utilização de uma única tabela *hash,* cujas entradas definem tanto os valores de estado significativos como os limites de intervalos de valores de contexto (corrente) numéricos podem eficientemente lidar com diferentes casos, onde, por exemplo, há dois intervalos de valores de estado não-significativos adjacentes (também designados como valores de contexto não-significativos numéricos) sem um valor de estado significativo entre eles. Uma eficiência computacional particularmente alta é obtida devido a um número de acessos da tabela sendo mantidos pequenos. Por exemplo, uma única pesquisa de tabela iterativa é suficiente na maioria das realizações a fim de descobrir se o valor de contexto corrente numérico é igual a qualquer um dos valores de estado significativos, ou em quais dos intervalos de valores de estado

não-significativos o valor de contexto corrente numérico permanece. Consequentemente, o número de acessos da tabela que consomem tempo e energia pode ser mantido pequeno. Assim, o selecionador da regra de mapeamento 760,que utiliza a tabela *hash* 762, pode ser considerado como um selecionador da regra de mapeamento particularmente eficiente em termos de complexidade computacional, enquanto ainda permitem obter uma boa eficiência de codificação (em termos de taxa de bit).

90. Outros detalhes referentes à derivação da informação da regra de mapeamento 742 do valor de contexto corrente numérico 754 serão descritos abaixo.

Decodificador de áudio de acordo com a figura 8

91. A figura 8 mostra um diagrama em blocos esquemático de um decodificador de áudio 800. O decodificador de áudio 800 é configurado para receber uma informação de áudio codificada 810 e prover, com base neste, uma informação de áudio decodificada 812. O decodificador de áudio 800 compreende um decodificador aritmético 820 que é configurado para prover diversos valores espectrais 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 em domínio de tempo 830 que é configurado para receber os valores espectrais decodificados 822 e prover a representação de áudio de domínio de tempo 812, que pode constituir a informação de áudio decodificada, que utiliza os valores espectrais decodificados 822, a fim de obter uma informação de áudio decodificada 812.

92. O decodificador aritmético 820 compreende um

determinador do 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 do valor espectral 824 pode ser configurado para realizar um mapeamento dependendo de uma regra de mapeamento, que pode ser descrita por uma informação da regra de mapeamento 828a. A informação da regra de mapeamento 828a pode, por exemplo, tomar a forma de um valor de índice de regra de mapeamento, ou da tabela de frequências cumulativas selecionada (selecionada, por exemplo, dependendo de um valor de índice de regra de mapeamento).

93. O decodificador aritmético 820 é configurado para selecionar uma regra de mapeamento (por exemplo, a tabela de frequências cumulativas) que descreve um mapeamento de valores de código (descritos pela representação aritmeticamente codificada 821 de valores espectrais) em um código de símbolo (que descreve um ou mais valores espectrais, ou um plano de bits mais significativo deste) dependendo de um estado de contexto (que pode ser descrito pela informação do estado de contexto 826a). O decodificador aritmético 820 é configurado para determinar o estado de contexto corrente (descrito pelo valor de contexto corrente numérico) dependendo de diversos valores espectrais previamente decodificados. Para esta finalidade, um rastreador de estado 826 pode ser utilizado, que recebe uma informação que descreve os valores espectrais previamente decodificados e que provê, com base neste, um valor de contexto corrente numérico 826a

que descreve o estado de contexto corrente.

94. O decodificador aritmético é também configurado para avaliar a tabela *hash* 829, cujas entradas definem tanto valores numéricos de estado significativos entre os valores de contexto numéricos como limites de intervalos de valores de contexto numéricos, a fim de selecionar a regra de mapeamento, em que um valor de índice de regra de mapeamento está individualmente associado a um valor de contexto numérico sendo um valor de estado significativo, e em que um valor de índice de regra de mapeamento comum está associado a diferentes valores de contexto numéricos encontrando-se dentro de um intervalo delimitado por limites de intervalo. A avaliação da tabela *hash* 829 pode, por exemplo, ser realizada utilizando um avaliador da tabela *hash* que pode ser parte do selecionador da regra de mapeamento 828. Assim, uma informação da regra de mapeamento 828a, por exemplo, na forma de um valor de índice de regra de mapeamento, é obtido com base no valor de contexto corrente numérico 826a que descreve o estado de contexto corrente. O selecionador da regra de mapeamento 828 pode, por exemplo, determinar o valor de índice de regra de mapeamento 828a dependendo de um resultado da avaliação da tabela *hash* 829. De modo alternativo, uma avaliação da tabela *hash* 829 pode prover diretamente o valor de índice de regra de mapeamento.

95. Referente 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, a tabela de frequências cumulativas) que é, em média, bem adaptada aos valores espectrais a ser decodificados, conforme a regra de mapeamento é selecionada

dependendo do estado de contexto corrente (descrito, por exemplo, pelo valor de contexto corrente numérico), que por sua vez é determinado dependendo dos diversos valores espectrais previamente decodificados. Assim, as dependências estatísticas entre os valores espectrais adjacentes a ser codificados podem ser explorados. Além disso, o decodificador aritmético 820 pode ser implementado eficientemente, com uma boa troca entre a complexidade computacional, tamanho da tabela, e eficiência da codificação, que utiliza o selecionador da regra de mapeamento 828. Pela avaliação de uma (única) tabela *hash* 829, nas entradas que descrevem ambos os valores de estado significativos e limites de intervalo dos intervalos de valores de estado não-significativos, uma única pesquisa de tabela iterativa pode ser suficiente a fim de derivar a informação da regra de mapeamento 828a do valor de contexto corrente numérico 826a. Assim, é possível mapear um número comparavelmente grande de diferentes possíveis valores de contexto (corrente) numéricos em um número comparavelmente pequeno de diferentes valores de índice de regra de mapeamento. Utilizando a tabela *hash* 829, conforme descrito acima, é possível explicar a observação que, em muitos casos, um único valor de estado significativo isolado (valor de contexto significativo) é incorporado entre um intervalo esquerdo de valores de estado não-significativos (valores de contexto não-significativos) e um intervalo direito de valores de estado não-significativos (valores de contexto não-significativos), em que um diferente valor de índice de regra de mapeamento é associado ao valor de estado significativo (valor de contexto significativo), quando comparado aos valores de estado (valores de contexto) do

intervalo esquerdo e os valores de estado (valores de contexto) do intervalo direito. Entretanto, o uso da tabela *hash* 829 é também bem adequado para situações em que os dois intervalos de valores de estado numérico são imediatamente adjacentes, sem um valor de estado significativo entre eles.

96. Para concluir, o selecionador da regra de mapeamento 828, que avalia a tabela *hash* 829, traz consigo uma eficiência particularmente boa ao selecionar uma regra de mapeamento (ou ao prover um valor de índice de regra de mapeamento) dependendo do estado de contexto corrente (ou dependendo do valor de contexto corrente numérico que descreve o estado de contexto corrente), pois o mecanismo de *hashing* é bem adaptado aos típicos cenários do contexto em um decodificador de áudio.

97. Outros detalhes serão descritos abaixo.

Mecanismo de *Hashing* do Valor de Contexto de acordo com a figura 9

98. A seguir, um mecanismo de *hashing* de contexto será revelado, que pode ser implementado no selecionador da regra de mapeamento 760 e/ou no selecionador da regra de mapeamento 828. A tabela *hash* 762 e/ou a tabela *hash* 829 podem ser utilizadas para implementar o dito valor de mecanismo de *hashing* de contexto.

99. Agora com referência à figura 9, que mostra um cenário de *hashing* do valor de contexto corrente numérico, outros detalhes serão descritos. Na representação gráfica da figura 9, uma abscissa 910 descreve valores do valor de contexto corrente numérico (ou seja, valores de contexto numéricos). Uma ordenada 912 descreve valores de índice de regra de mapeamento. As

marcações 914 descrevem valores de índice de regra de mapeamento para valores de contexto não-significativos numéricos (que descrevem estados não-significativos). As marcações 916 descrevem os valores de índice de regra de mapeamento para valores de contexto significativos numéricos “individuais” (verdadeiros) que descrevem estados significativos individuais (verdadeiros). As marcações 916 descrevem valores de índice de regra de mapeamento para valores de contexto numéricos “impróprios” que descrevem estados significativos “impróprios”, em que um estado significativo “impróprio” é um estado significativo no qual o mesmo valor de índice de regra de mapeamento está associado a um dos intervalos de contexto adjacentes de contextos não-significativos numéricos.

100. Como pode ser visto, uma entrada da tabela *hash* “ari_hash_m[i1]” descreve um estado significativo individual (verdadeiro) tendo um valor de contexto numérico de c1. Como pode ser visto, o valor de índice de regra de mapeamento mriv1 está associado ao estado significativo individual (verdadeiro) tendo o valor de contexto numérico c1. Assim, o valor de contexto numérico c1 e o valor de índice de regra de mapeamento mriv1 podem ser descritos pela entrada da tabela *hash* “ari_hash_m[i1]”. Um intervalo 932 de valores de contexto numéricos é delimitado pelo valor de contexto numérico c1, em que o valor de contexto numérico c1 não pertence ao intervalo 932, de modo que o maior valor de contexto numérico de intervalo 932 seja igual a c1 – 1. Um valor de índice de regra de mapeamento de mriv4 (que é diferente de mriv1) está associado aos valores de contexto numéricos do intervalo 932. O valor de índice de regra de mapeamento mriv4

pode, por exemplo, ser descrito pela entrada de tabela “ari_lookup_m[i1-1]” da tabela adicional “ari_lookup_m”.

101. Além disso, um valor de índice de regra de mapeamento mriv2 pode ser associado a valores de contexto numéricos encontrando-se dentro de um intervalo 934. Um limite inferior de intervalo 934 é determinado pelo valor de contexto numérico c1, que é um valor de contexto numérico significativo, em que o valor de contexto numérico c1 não pertence ao intervalo 932. Assim, o menor valor do intervalo 934 é igual a c1 + 1 (supondo os valores de número inteiro de contexto numérico). Outro limite do intervalo 934 é determinado pelo valor de contexto numérico c2, em que o valor de contexto numérico c2 não pertence ao intervalo 934, de modo que o maior valor do intervalo 934 seja igual a c2 – 1. O valor de contexto numérico c2 é um valor chamado valor de contexto numérico “impróprio”, que é descrito por uma entrada da tabela *hash* “ari_hash_m[i2]”. Por exemplo, o valor de índice de regra de mapeamento mriv2 pode ser associado ao valor de contexto numérico c2, de modo que o valor de contexto numérico associado ao valor de contexto numérico “impróprio” significativo c2 seja igual ao valor de índice de regra de mapeamento associado ao intervalo 934 delimitado pelo valor de contexto numérico c2. Além disso, um intervalo 936 de valor de contexto numérico é também delimitado pelo valor de contexto numérico c2, em que o valor de contexto numérico c2 não pertence ao intervalo 936, de modo que o menor valor de contexto numérico do intervalo 936 seja igual a c2 + 1. Um valor de índice de regra de mapeamento mriv3, que é tipicamente diferente do valor de índice de regra de mapeamento mriv2, é associado aos valores de contexto numéricos do intervalo 936.

102. Como pode ser visto, o valor de índice de regra de mapeamento mriv4, que está associado ao intervalo 932 de valores de contexto numéricos, pode ser descrito por uma entrada “ari_lookup_m[i1-1]” da tabela “ari_lookup_m”, o índice da regra de mapeamento mriv2, que é associado aos valores de contexto numéricos do intervalo 934, pode ser descrito por uma entrada de tabela “ari_lookup_m[i1]” da tabela “ari_lookup_m”, e o valor de índice de regra de mapeamento mriv3 pode ser descrito por uma entrada de tabela “ari_lookup_m[i2]” da tabela “ari_lookup_m”. No exemplo dado aqui, o valor do índice da tabela *hash* i2, pode ser maior, em 1, que o valor do índice da tabela *hash* i1.

103. Como pode ser visto da figura 9, o selecionador da regra de mapeamento 760 ou o selecionador da regra de mapeamento 828 pode receber um valor de contexto corrente numérico 764, 826a, e decidir, avaliando as entradas da tabela “ari_hash_m”, se o valor de contexto corrente numérico é um valor de estado significativo (independente se for um valor de estado significativo “individual” ou um valor de estado significativo “impróprio”), ou se o valor de contexto corrente numérico dentro de um dos intervalos 932, 934, 936, que são delimitados pelos valores de estado significativos (“individuais” ou “impróprios”) c1, c2. Tanto a verificação se o valor de contexto corrente numérico é igual a um valor de estado significativo c1, c2 quanto uma avaliação dos intervalos 932, 934, 936 na qual o valor de contexto corrente numérico permanece (no caso em que o valor de contexto corrente numérico não é igual a um valor de estado significativo) podem ser realizadas utilizando uma única pesquisa comum da tabela *hash*.

104. Além disso, uma avaliação da tabela *hash* “ari_hash_m” pode ser utilizada para obter um valor do índice da tabela *hash* (por exemplo, i1-1, i1 ou i2). Assim, o selecionador da regra de mapeamento 760, 828 pode ser configurado para obter, avaliando uma única tabela *hash* 762, 829 (por exemplo, a tabela *hash* “ari_hash_m”), um valor do índice da tabela *hash* (por exemplo, i1-1, i1 ou i2) designando um valor de estado significativo (por exemplo, c1 ou c2) e/ou um intervalo (por exemplo, 932,934,936) e uma informação se o valor de contexto corrente numérico for um valor de contexto significativo (também designado como valor de estado significativo) ou não.

105. Além disso, se for observado em uma avaliação da tabela *hash* 762, 829, “ari_hash_m”, que o valor de contexto corrente numérico não é um valor de contexto “significativo” (ou valor de estado “significativo”), o valor do índice da tabela *hash* (por exemplo, i1-1, i1 ou i2) obtido de uma avaliação da tabela *hash* (“ari_hash_m”) pode ser utilizado para obter um valor de índice de regra de mapeamento associado a um intervalo 932, 934, 936 de valores de contexto numéricos. Por exemplo, o valor do índice da tabela *hash* (por exemplo, i1-1, i1 ou i2) pode ser utilizado para designar uma entrada da tabela de mapeamento adicional (por exemplo, “ari_lookup_m”), que descreve os valores de índice de regra de mapeamento associados aos intervalos 932, 934, 936 cujo valor de contexto corrente numérico permanece.

106. Para mais detalhes, a referência é feita à discussão detalhada abaixo do algoritmo “arith_get_pk” (em que há diferentes opções para este algoritmo “arith_get_pk()”, exemplos dos quais são mostrados nas figuras 5e e 5f).

107. Além disso, deve ser observado que o tamanho dos intervalos pode diferir de um caso para outro. Em alguns casos, um intervalo de valores de contexto numéricos compreende um único valor de contexto numérico. Entretanto, em muitos casos, um intervalo pode compreender diversos valores de contexto numéricos.

Codificador de áudio de acordo com a figura 10

108. A figura 10 mostra um diagrama em blocos esquemático de um codificador de áudio 1000 de acordo com uma realização da invenção. O codificador de áudio 1000 de acordo com a figura 10 é semelhante ao codificador de áudio 700 de acordo com a figura 7, de modo que os sinais idênticos e meios sejam designados com números idênticos de referência nas figuras 7 e 10.

109. O codificador de áudio 1000 é configurado para receber uma informação de áudio de entrada 710 e prover, com base neste, uma informação de áudio codificada 712. O codificador de áudio 1000 compreende um conversor de domínio de tempo em domínio de frequência com compactação de energia 720, que é configurado para prover uma representação 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 1000 também compreende um codificador aritmético 1030 configurado para codificar um valor espectral (fora do conjunto de valores espectrais formando a representação de áudio de domínio de frequência 722), ou uma versão pré-processada deste, que utiliza uma senha de comprimento variável para obter a informação de áudio codificada 712 (que pode compreender, por exemplo, diversas senhas de comprimento

variável).

110. O codificador aritmético 1030 é configurado para mapear um valor espectral, ou diversos valores espectrais, ou um valor de um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais, em um valor de código (ou seja, em uma senha de comprimento variável) dependendo de um estado de contexto. O codificador aritmético 1030 é configurado para selecionar uma regra de mapeamento que descreve um mapeamento de um valor espectral, ou de diversos valores espectrais, ou de um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais, em um valor de código dependendo de um estado de contexto. O codificador aritmético é configurado para determinar o estado de contexto corrente dependendo de diversos valores espectrais previamente codificados (preferivelmente, mas não necessariamente adjacentes). Para esta finalidade, o codificador aritmético é configurado para modificar uma representação numérica de um valor de contexto prévio numérico, que descreve um estado de contexto associado a um ou mais valores espectrais previamente codificados (por exemplo, para selecionar uma regra de mapeamento correspondente), dependendo de um valor da sub-região de contexto, para obter uma representação numérica de um valor de contexto corrente numérico que descreve um estado de contexto associado a um ou mais valores espectrais a ser codificados (por exemplo, para selecionar uma regra de mapeamento correspondente).

111. Como pode ser visto, o mapeamento de um valor espectral, ou de diversos valores espectrais, ou de um plano de bits mais significativo de um valor espectral ou de diversos

valores espectrais, em um valor de código pode ser realizado por uma codificação do valor espectral 740 que utiliza uma regra de mapeamento descrito por uma informação da regra de mapeamento 742. Um rastreador de estado 750 pode ser configurado para rastrear o estado de contexto. O rastreador de estado 750 pode ser configurado para modificar uma representação numérica de um valor de contexto prévio numérico, que descreve um estado de contexto associado a uma codificação de um ou mais valores espectrais previamente codificados, dependendo de um valor da sub-região de contexto, para obter uma representação numérica de um valor de contexto corrente numérico que descreve um estado de contexto associado a uma codificação de um ou mais valores espectrais a ser codificados. A modificação da representação numérica do valor de contexto prévio numérico pode, por exemplo, ser realizada por um modificador de representação numérica 1052, que recebe o valor de contexto prévio numérico e um ou mais valores da sub-região de contexto e provê o valor de contexto corrente numérico. Assim, o rastreador de estado 1050 provê uma informação 754 que descreve o estado de contexto corrente, por exemplo, na forma de um valor de contexto corrente numérico. Um selecionador da regra de mapeamento 1060 pode selecionar uma regra de mapeamento, por exemplo, a tabela de frequências cumulativas, que descreve um mapeamento de um valor espectral, ou de diversos valores espectrais, ou de um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais, em um valor de código. Assim, o selecionador da regra de mapeamento 1060 provê a informação da regra de mapeamento 742 à codificação espectral 740.

112. Deve ser observado que, em algumas realizações, o

rastreador de estado 1050 pode ser idêntico ao rastreador de estado 750 ou ao rastreador de estado 826. Também deve ser observado que o selecionador da regra de mapeamento 1060 pode, em algumas realizações, ser idêntico ao selecionador da regra de mapeamento 760, ou ao selecionador da regra de mapeamento 828.

113. Para resumir o mencionado acima, o codificador de áudio 1000 realiza uma codificação aritmética de uma representação de áudio de domínio de frequência provida pelo conversor de domínio de tempo em domínio de frequência. A codificação aritmética é dependente de contexto, de modo que uma regra de mapeamento (por exemplo, a tabela de frequências cumulativas) seja selecionada dependendo dos valores espectrais previamente codificados. Assim, os valores espectrais adjacentes em tempo e/ou em frequência (ou pelo menos dentro de um ambiente predeterminado) um ao outro e/ou ao valor espectral atualmente codificado (ou seja, 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.

114. Ao determinar o valor de contexto corrente numérico, uma representação numérica de um valor de contexto prévio numérico, que descreve um estado de contexto associado a um ou mais valores espectrais previamente codificados, é modificado dependendo de um valor da sub-região de contexto, para obter uma representação numérica de um valor de contexto corrente numérico que descreve um estado de contexto associado a um ou mais valores espectrais a ser codificados. Esta abordagem permite evitar um recálculo completo do valor de contexto corrente numérico, cujo

recálculo completo consome uma quantidade significante de recursos nas abordagens convencionais. Uma grande variedade de possibilidades existe para a modificação da representação numérica do valor de contexto prévio numérico, incluindo uma combinação de um redimensionamento de uma representação numérica do valor de contexto prévio numérico, uma adição de um valor da sub-região de contexto ou um valor derivado deste para a representação numérica do valor de contexto prévio numérico ou a uma representação numérica processada do valor de contexto prévio numérico, uma substituição de uma parte da representação numérica (em vez de toda a representação numérica) do valor de contexto prévio numérico dependendo do valor da sub-região de contexto, e assim por diante. Assim, tipicamente a representação numérica do valor de contexto corrente numérico é obtida com base na representação numérica do valor de contexto prévio numérico e também com base em pelo menos um valor da sub-região de contexto, em que tipicamente uma combinação de operações é realizada para combinar o valor de contexto prévio numérico com um valor da sub-região de contexto, como, por exemplo, duas ou mais operações fora de uma operação de adição, uma operação de subtração, uma operação de multiplicação, uma operação de divisão, uma operação Boolean-AND, uma operação Boolean-OR, uma operação Boolean-NAND, uma operação Boolean NOR, uma operação Boolean-negation, uma operação complementar ou uma operação de mudança. Assim, pelo menos uma parte da representação numérica do valor de contexto prévio numérico é tipicamente mantida inalterada (exceto para uma mudança opcional em uma posição diferente) ao derivar o valor de contexto corrente numérico do valor de contexto prévio numérico. Em contraste,

outras partes da representação numérica do valor de contexto prévio numérico são mudadas dependendo de um ou mais valores da sub-região de contexto. Assim, o valor de contexto corrente numérico pode ser obtido com um esforço computacional comparavelmente pequeno, enquanto evita um recálculo completo do valor de contexto corrente numérico.

115. Assim, um valor de contexto corrente numérico significativo pode ser obtido, que é bem adequado para o uso pelo selecionador da regra de mapeamento 1060.

116. Consequentemente, uma codificação eficiente pode ser obtida mantendo o cálculo de contexto suficientemente simples.

Decodificador de áudio de acordo com a figura 11

117. A figura 11 mostra um diagrama em blocos esquemático de um decodificador de áudio 1100. O decodificador de áudio 1100 é semelhante ao decodificador de áudio 800 de acordo com a figura 8, de modo que os sinais idênticos, meios e funcionalidades sejam designados com números idênticos de referência.

118. O decodificador de áudio 1100 é configurado para receber uma informação de áudio codificada 810 e prover, com base neste, uma informação de áudio decodificada 812. O decodificador de áudio 1100 compreende um decodificador aritmético 1120 que é configurado para prover diversos valores espectrais decodificados 822 com base em uma representação aritmeticamente codificada 821 dos valores espectrais. O decodificador de áudio 1100 também compreende um conversor de domínio de frequência em domínio de tempo 830 que é configurado para receber os valores espectrais decodificados 822 e prover a representação de áudio de domínio de

tempo 812, que pode constituir a informação de áudio decodificada, que utiliza os valores espectrais decodificados 822, a fim de obter uma informação de áudio decodificada 812.

119. O decodificador aritmético 1120 compreende um determinador do 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 do valor espectral 824 pode ser configurado para realizar o mapeamento dependendo de uma regra de mapeamento, que pode ser descrito por uma informação da regra de mapeamento 828a. A informação da regra de mapeamento 828a pode, por exemplo, compreender um valor de índice de regra de mapeamento, ou pode compreender um conjunto de entradas selecionadas da tabela de frequências cumulativas.

120. O decodificador aritmético 1120 é configurado para selecionar uma regra de mapeamento (por exemplo, a 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) dependendo de um estado de contexto, cujo estado de contexto pode ser descrito pela informação do estado de contexto 1126a. A informação do estado de contexto 1126a pode tomar uma forma de um valor de contexto corrente numérico. O decodificador aritmético 1120 é configurado para determinar o estado de contexto corrente dependendo de diversos valores espectrais previamente decodificados 822. Para esta finalidade, um

rastreador de estado 1126 pode ser utilizado, que recebe uma informação que descreve os valores espectrais previamente decodificados. O decodificador aritmético é configurado para modificar uma representação numérica do valor de contexto prévio numérico, que descreve um estado de contexto associado a um ou mais valores espectrais previamente decodificados, dependendo de um valor da sub-região de contexto, para obter uma representação numérica de um valor de contexto corrente numérico que descreve um estado de contexto associado a um ou mais valores espectrais a ser decodificados. Uma modificação da representação numérica do valor de contexto prévio numérico pode, por exemplo, ser realizada por um modificador de representação numérica 1127, que faz parte do rastreador de estado 1126. Assim, a informação do estado de contexto corrente 1126a é obtida, por exemplo, na forma de um valor de contexto corrente numérico. A seleção da regra de mapeamento pode ser realizada por um selecionador da regra de mapeamento 1128, que deriva uma informação da regra de mapeamento 828a da informação do estado de contexto corrente 1126a, e que provê a informação da regra de mapeamento 828a ao determinador do valor espectral 824.

121. Referente à funcionalidade do decodificador de sinal de áudio 1100, deve ser observado que o decodificador aritmético 1120 é configurado para selecionar uma regra de mapeamento (por exemplo, a tabela de frequências cumulativas) que é, em média, bem adaptada ao valor espectral a ser decodificado, conforme a regra de mapeamento é selecionada dependendo do estado de contexto corrente, que, por sua vez, é determinada dependendo de diversos valores espectrais previamente decodificados. Assim,

as dependências estatísticas entre os valores espectrais adjacentes a ser decodificados podem ser explorados.

122. Além disso, modificando uma representação numérica de um valor de contexto prévio numérico que descreve um estado de contexto associado a uma decodificação de um ou mais valores espectrais previamente decodificados, dependendo de um valor da sub-região de contexto, para obter uma representação numérica de um valor de contexto corrente numérico que descreve um estado de contexto associado a uma decodificação de um ou mais valores espectrais a ser decodificados, é possível obter uma informação significativa sobre o estado de contexto corrente, que é bem adequado para um mapeamento de um valor de índice de regra de mapeamento, com esforço computacional comparavelmente pequeno. Mantendo pelo menos uma parte de uma representação numérica do valor de contexto prévio numérico (possivelmente em um mudado por bit ou uma versão escalada) enquanto atualiza outra parte da representação numérica do valor de contexto prévio numérico dependendo dos valores da sub-região de contexto que não foram considerados no valor de contexto prévio numérico, mas que devem ser considerados no valor de contexto corrente numérico, um número de operações para derivar o valor de contexto corrente numérico pode ser mantido razoavelmente pequeno. Ainda, é possível explorar o fato de que os contextos utilizados para decodificar os valores espectrais adjacentes são tipicamente semelhantes ou correlacionados. Por exemplo, um contexto para uma decodificação de um primeiro valor espectral (ou de uma primeira pluralidade de valores espectrais) é dependente de um primeiro conjunto de valores espectrais previamente decodificados. Um contexto de

decodificação de um segundo valor espectral (ou um segundo conjunto de valores espectrais), que é adjacente ao primeiro valor espectral (ou o primeiro conjunto de valores espectrais) pode compreender um segundo conjunto de valores espectrais previamente decodificados. Como o primeiro valor espectral e o segundo valor espectral são assumidos como adjacentes (por exemplo, com relação às frequências associadas), o primeiro conjunto de valores espectrais, que determina o contexto para a codificação do primeiro valor espectral, pode compreender certa sobreposição com o segundo conjunto de valores espectrais, que determina o contexto para a decodificação do segundo valor espectral. Assim, pode ser facilmente entendido que o estado de contexto para a decodificação do segundo valor espectral compreenda certa correlação com o estado de contexto para a decodificação do primeiro valor espectral. Uma eficiência computacional da derivação de contexto, ou seja, da derivação do valor de contexto corrente numérico, pode ser obtido explorando tais correlações. Foi observado que a correlação entre os estados de contexto para uma decodificação de valores espectrais adjacentes (por exemplo, entre o estado de contexto descrito pelo valor de contexto prévio numérico e pelo estado de contexto descrito pelo valor de contexto corrente numérico) pode ser explorada eficientemente modificando somente estas partes do valor de contexto prévio numérico que são dependentes dos valores da sub-região de contexto não considerados para a derivação do estado de contexto numérico anterior, e derivando o valor de contexto corrente numérico do valor de contexto prévio numérico.

123. Para concluir, os conceitos descritos aqui

permitem uma eficiência computacional particularmente boa ao derivar o valor de contexto corrente numérico.

124. Outros detalhes serão descritos abaixo.

Codificador de áudio de acordo com a figura 12

125. A figura 12 mostra um diagrama em blocos esquemático de um codificador de áudio, de acordo com uma realização da invenção. O codificador de áudio 1200 de acordo com a figura 12 é semelhante ao codificador de áudio 700 de acordo com a figura 7, de modo que meios idênticos, sinais e funcionalidades sejam designados com números idênticos de referência.

126. O codificador de áudio 1200 é configurado para receber uma informação de áudio de entrada 710 e prover, com base neste, uma informação de áudio codificada 712. O codificador de áudio 1200 compreende um conversor de domínio de tempo em domínio de frequência com compactação de energia 720 que é configurado para prover uma representação de áudio de domínio de frequência 722 com base em uma representação de áudio 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 1200 também compreende um codificador aritmético 1230 configurado para codificar um valor espectral (fora do conjunto de valores espectrais que formam a representação de áudio de domínio de frequência 722), ou diversos valores espectrais, ou uma versão pré-processada deste, que utiliza uma senha de comprimento variável para obter a informação de áudio codificada 712 (que pode compreender, por exemplo, diversas senhas de comprimento variável).

127. O codificador aritmético 1230 é configurado para

mapear um valor espectral, ou diversos valores espectrais, ou um valor de um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais, em um valor de código (ou seja, em uma senha de comprimento variável), dependendo de um estado de contexto. O codificador aritmético 1230 é configurado para selecionar uma regra de mapeamento que descreve um mapeamento de um valor espectral, ou de diversos valores espectrais, ou de um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais, em um valor de código, dependendo do estado de contexto. O codificador aritmético é configurado para determinar o estado de contexto corrente dependendo de diversos valores espectrais previamente codificados (preferivelmente, mas não necessariamente, adjacentes). Para esta finalidade, o codificador aritmético é configurado para obter diversos valores da sub-região de contexto com base em valores espectrais previamente codificados, para armazenar os ditos valores da sub-região de contexto, e para derivar um valor de contexto corrente numérico associado a um ou mais valores espectrais a ser codificados dependendo dos valores da sub-região de contexto armazenados. Além disso, o codificador aritmético é configurado para calcular a norma de um vetor formado pelos diversos valores espectrais previamente codificados, a fim de obter um valor comum da sub-região de contexto associado a diversos valores espectrais previamente codificados.

128. Como pode ser visto, o mapeamento de um valor espectral, ou de diversos valores espectrais, ou de um plano de bits mais significativo de um valor espectral ou de diversos valores espectrais, em um valor de código pode ser realizado por

uma codificação do valor espectral 740 que utiliza uma regra de mapeamento descrito por uma informação da regra de mapeamento 742. Um rastreador de estado 1250 pode ser configurado para rastrear o estado de contexto e pode compreender um computador do valor da sub-região de contexto 1252, para calcular a norma de um vetor formado por diversos valores espectrais previamente codificados, a fim de obter um valor comum da sub-região de contexto associado a diversos valores espectrais previamente codificados. O rastreador de estado 1250 é também preferivelmente configurado para determinar o estado de contexto corrente dependendo de um resultado do dito cálculo de um valor da sub-região de contexto realizado pelo computador do valor da sub-região de contexto 1252. Assim, o rastreador de estado 1250 provê uma informação 1254, que descreve o estado de contexto corrente. Um selecionador da regra de mapeamento 1260 pode selecionar uma regra de mapeamento, por exemplo, a 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. Assim, o selecionador da regra de mapeamento 1260 provê a informação da regra de mapeamento 742 à codificação espectral 740.

129. Para resumir o mencionado acima, o codificador de áudio 1200 realiza uma codificação aritmética de uma representação de áudio de domínio de frequência provida pelo conversor de domínio de tempo em domínio de frequência 720. A codificação aritmética é dependente de contexto, de modo que uma regra de mapeamento (por exemplo, a tabela de frequências cumulativas) seja selecionada dependendo de valores espectrais previamente codificados. Assim, os valores espectrais adjacentes em tempo e/ou

frequência (ou, pelo menos, dentro de um ambiente predeterminado) um ao outro e/ou ao valor espectral atualmente codificado (ou seja, 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.

130. A fim de prover um valor de contexto corrente numérico, um valor da sub-região de contexto associado a diversos valores espectrais previamente codificados é obtido com base em um cálculo de uma norma de um vetor formado por diversos valores espectrais previamente codificados. O resultado da determinação do valor de contexto corrente numérico é aplicado na seleção do estado de contexto corrente, ou seja, na seleção de uma regra de mapeamento.

131. Calculando a norma de um vetor formado por diversos valores espectrais previamente codificados, uma informação significativa que descreve uma parte do contexto de um ou mais valores espectrais a serem codificados pode ser obtida, em que a norma de um vetor de valores espectrais previamente codificados pode tipicamente ser representada com um número comparavelmente pequeno de bits. Assim, a quantidade da informação de contexto, que precisa ser armazenada para uso futuro na derivação de um valor de contexto corrente numérico, pode ser mantida suficientemente pequena aplicando a abordagem discutida acima para o cálculo dos valores da sub-região de contexto. Foi observado que a norma de um vetor de valores espectrais previamente codificados tipicamente compreende a informação mais significativa referente ao estado do contexto. Em contraste, foi

observado que o sinal dos ditos valores espectrais previamente codificados tipicamente compreende um impacto subordinado no estado do contexto, de modo que faz sentido negar o sinal dos valores espectrais previamente decodificados a fim de reduzir a quantidade de informação a ser armazenada para uso futuro. Ainda, foi observado que o cálculo de uma norma de um vetor de valores espectrais previamente codificados é uma abordagem razoável para a derivação de um valor da sub-região de contexto, como o efeito médio, que é tipicamente obtido pelo cálculo da norma, deixa a informação mais importante sobre o estado de contexto substancialmente não afetado. Para resumir, o valor da sub-região de contexto cálculo realizado pelo computador do valor da sub-região de contexto 1252 permite prover uma informação da sub-região de contexto compacto para armazenamento e reuso futuro, em que a informação mais relevante sobre o estado de contexto é preservada em vez da redução da quantidade de informação.

132. Assim, uma codificação eficiente da informação de áudio de entrada 710 pode ser obtida, enquanto mantém o esforço computacional e uma quantidade de dados a ser armazenado pelo codificador aritmético 1230 suficientemente pequeno.

Decodificador de áudio de acordo com a figura 13

133. A figura 13 mostra um diagrama em blocos esquemático de um decodificador de áudio 1300. Como o decodificador de áudio 1300 é semelhante ao decodificador de áudio 800 de acordo com a figura 8, e ao decodificador de áudio 1100 de acordo com a figura 11, meios idênticos, sinais e funcionalidades são designados como números idênticos.

134. O decodificador de áudio 1300 é configurado para

receber uma informação de áudio codificada 810 e prover, com base neste, uma informação de áudio decodificada 812. O decodificador de áudio 1300 compreende um decodificador aritmético 1320 que é configurado para prover diversos valores espectrais decodificados 822 com base em uma representação aritmeticamente codificada 821 dos valores espectrais. O decodificador de áudio 1300 também compreende um conversor de domínio de frequência em domínio de tempo 830 que é configurado para receber os valores espectrais decodificados 822 e prover a representação de áudio de domínio de tempo 812, que pode constituir a informação de áudio decodificada, que utiliza os valores espectrais decodificados 822, a fim de obter uma informação de áudio decodificada 812.

135. O decodificador aritmético 1320 compreende um determinador do 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 do valor espectral 824 pode ser configurado para realizar um mapeamento dependendo de uma regra de mapeamento, que é descrita por uma informação da regra de mapeamento 828a. A informação da regra de mapeamento 828a pode, por exemplo, compreender um valor de índice de regra de mapeamento, ou um conjunto selecionado de entradas da tabela de frequências cumulativas.

136. O decodificador aritmético 1320 é configurado para selecionar uma regra de mapeamento (por exemplo, a 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) dependendo de um estado de contexto (que pode ser descrito pela informação do estado de contexto 1326a). O decodificador aritmético 1320 é configurado para determinar o estado de contexto corrente dependendo de diversos valores espectrais previamente decodificados 822. Para esta finalidade, um rastreador de estado 1326 pode ser utilizado, que recebe uma informação que descreve os valores espectrais previamente decodificados. O decodificador aritmético é também configurado para obter diversos valores da sub-região de contexto com base em valores espectrais previamente decodificados e para armazenar os ditos valores da sub-região de contexto. O decodificador aritmético é configurado para derivar um valor de contexto corrente numérico associado a um ou mais valores espectrais a serem decodificados dependendo dos valores da sub-região de contexto armazenados. O decodificador aritmético 1320 é configurado para calcular a norma de um vetor formado por diversos valores espectrais previamente decodificados, a fim de obter um valor comum da sub-região de contexto associado a diversos valores espectrais previamente decodificados.

137. O cálculo da norma de um vetor formado por diversos valores espectrais previamente codificados, a fim de obter um valor comum da sub-região de contexto associado a diversos valores espectrais previamente decodificados, pode, por exemplo, ser realizado pelo computador do valor da sub-região de contexto 1327, que faz parte do rastreador de estado 1326. Assim, uma informação do estado de contexto corrente 1326a é obtida com

base nos valores da sub-região de contexto, em que o rastreador de estado 1326 preferivelmente provê um valor de contexto corrente numérico associado a um ou mais valores espectrais a ser decodificados dependendo dos valores da sub-região de contexto armazenados. A seleção das regras de mapeamento pode ser realizada por um selecionador da regra de mapeamento 1328, que deriva uma informação da regra de mapeamento 828a da informação do estado de contexto corrente 1326a, e que provê a informação da regra de mapeamento 828a ao determinador do valor espectral 824.

138. Referente à funcionalidade do decodificador de sinal de áudio 1300, deve ser observado que o decodificador aritmético 1320 é configurado para selecionar uma regra de mapeamento (por exemplo, a tabela de frequências cumulativas) que é, em média, bem adaptada ao valor espectral a ser decodificado, conforme a regra de mapeamento é selecionada dependendo do estado de contexto corrente, que, por sua vez, é determinada dependendo de diversos valores espectrais previamente decodificados. Assim, as dependências estatísticas entre valores espectrais adjacentes a ser decodificados podem ser exploradas.

139. Entretanto, foi observado que é eficiente, em termos de uso da memória, armazenar valores da sub-região de contexto, que têm como base o cálculo de uma norma de um vetor formado em diversos valores espectrais previamente decodificados, para uso futuro na determinação do valor de contexto numérico. Também foi observado que tais valores da sub-região de contexto ainda compreendem a informação de contexto mais relevante. Assim, o conceito utilizado pelo rastreador de estado 1326 constitui um bom compromisso entre a eficiência da codificação, a eficiência

computacional e a eficiência de armazenamento.

140. Outros detalhes serão descritos abaixo.

Codificador de áudio de acordo com a figura 1

141. A seguir, um codificador de áudio, de acordo com uma realização da presente invenção, será descrito. A figura 1 mostra um diagrama em blocos esquemático de tal codificador de áudio 100.

142. O codificador de áudio 100 é configurado para receber uma informação de áudio de entrada 110 e prover, com base neste, 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 prover, com base neste, uma informação pré-processada de áudio de entrada 110a. O codificador de áudio 100 também compreende um transformador de sinal de domínio de tempo em domínio de frequência com compactação de energia 130, que é também designado como conversor de sinal. O conversor de sinal 130 é configurado para receber a informação de áudio de entrada 110, 110a e prover, com base neste, uma informação de áudio de domínio de frequência 132, que preferivelmente toma a 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 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 diversas estruturas de áudio subsequentes, de sobreposição ou não sobreposição da informação de

áudio de entrada 110, 110a e prover, com base neste, 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 associados a cada estrutura.

143. O transformador de sinal de domínio de tempo em domínio de frequência com compactação de energia 130 pode compreender um banco de filtros com compactação de energia, que provê valores espectrais associados a diferentes faixas de frequência, de sobreposição ou não sobreposição. Por exemplo, o transformador de sinal 130 pode compreender um transformador MDCT de janelamento 130a, que é configurado para colocar em janelas a informação de áudio de entrada 110, 110a (ou a estrutura deste) que utiliza uma janela de transformação e para realizar uma transformação de cosseno discreto modificado da informação de áudio de entrada em janela 110, 110a (ou a estrutura em janela deste). Assim, a representação de áudio de domínio de frequência 132 pode compreender um conjunto de, por exemplo, 1024 valores espectrais na forma de coeficientes MDCT associados a uma estrutura da informação de áudio de entrada.

144. 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 prover, com base neste, uma representação de áudio de domínio de frequência pós-processada 142. O pós-processador espectral 140 pode, por exemplo, ser configurado para realizar uma forma de ruído temporale/ou uma previsão a longo prazo e/ou qualquer outro pós-processamento espectral conhecido na

técnica. O codificador de áudio ainda compreende, opcionalmente, um multiplicador de frequências/quantificador 150, que é configurado para receber a representação de áudio de domínio de frequência 132 ou a versão pós-processada 142 deste e prover uma representação de áudio de domínio de frequência escalada e quantizada 152.

145. O codificador de áudio 100 ainda compreende, opcionalmente, um processador do modelo psicoacústico 160, que é configurado para receber a informação de áudio de entrada 110 (ou a versão pós-processada 110a deste) e prover, com base neste, uma informação de controle opcional, que pode ser utilizada para o controle do transformador de sinal de domínio de tempo em domínio de frequência com compactação de energia 130, para o controle do pós-processador espectral opcional 140 e/ou para o controle do multiplicador de frequências/quantificador opcional 150. Por exemplo, o processador do 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. Assim, o processador do modelo psicoacústico 160 pode prover informação de controle, que é utilizado pelo codificador de áudio 100 a fim de ajustar a escala da representação de áudio de domínio de frequência 132, 142 pelo multiplicador de frequências/quantificador 150 e/ou a resolução de quantização aplicada pelo multiplicador de frequências/quantificador 150. Consequentemente, faixas do fator

de escala perceptualmente importantes (ou seja, grupos de valores espectrais adjacentes que são particularmente importantes para a percepção humana do conteúdo de áudio) são escaladas com um grande fator de escala e quantizadas com resolução comparavelmente alta, enquanto faixas do fator de escala perceptualmente menos importantes (ou seja, grupos de valores espectrais adjacentes) são escaladas com um fator de escala comparavelmente menor e quantizadas com uma resolução de quantização comparavelmente menor. Assim, os valores espectrais escalados de frequências perceptualmente mais importantes são tipicamente significantemente maiores que os valores espectrais de frequências perceptualmente menos importantes.

146. O codificador de áudio também compreende um codificador aritmético 170, que é configurado para receber a versão escalada e quantizada 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 mesmo a própria representação de áudio de domínio de frequência 132) e prover informação da senha aritmética 172a com base neste, de modo que a informação da senha aritmética representa a representação de áudio de domínio de frequência 152.

147. O codificador de áudio 100 também compreende um formatador de payload do fluxo de bits 190, que é configurado para receber a informação da senha aritmética 172a. O formatador de payload do fluxo de bits 190 é também tipicamente configurado para receber informação adicional, como, por exemplo, a informação do fator de escala que descreve quais fatores de escala foram aplicados pelo multiplicador de frequências/quantificador 150.

Além disso, o formatador de payload do fluxo de bits 190 pode ser configurado para receber outra informação de controle. O formatador de payload do fluxo de bits 190 é configurado para prover o fluxo de bits 112 com base na informação recebida montando o fluxo de bits de acordo com uma sintaxe do fluxo de bits desejada, que será discutida abaixo.

148. A seguir, detalhes referentes ao codificador aritmético 170 serão descritos. O codificador aritmético 170 é configurado para receber diversos valores espectrais pós-processados e escalados e quantizados da representação de áudio de domínio de frequência 132. O codificador aritmético compreende um extrator do plano de bits mais significante 174, ou mesmo de dois valores espectrais, 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 mais bits (por exemplo, dois ou três bits), que são os bits mais significativos do valor espectral. Assim, o extrator do plano de bits mais significativo 174 provê um valor do plano de bits mais significativo 176 de um valor espectral.

149. De modo alternativo, entretanto, o extrator do plano de bits mais significativo 174 pode prover um valor combinado do plano de bits mais significativo m que combina o plano de bits mais significativo de diversos valores espectrais (por exemplo, de valores espectrais a e b). O plano de bits mais significativo do valor espectral a é designado com m. De modo alternativo, o valor combinado do plano de bits mais significativo de diversos valores espectrais a,b é designado com m.

150. 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 do plano de bits mais significativo m. Opcionalmente, o determinador de senha 180 pode também prover uma ou mais senhas de escape (também designada aqui como “ARITH_ESCAPE”) indicando, por exemplo, quantos planos de bits menos significativos estão disponíveis (e, consequentemente, indicando o peso numérico do plano de bits mais significativo). O primeiro determinador de senha 180 pode ser configurado para prover a senha associada a um valor do plano de bits mais significativo m que utiliza a tabela de frequências cumulativas selecionada tendo (ou sendo referenciada por) um índice da tabela de frequências cumulativas pki.

151. A fim de determinar em qual tabela de frequências cumulativas deve ser selecionado, o codificador aritmético preferivelmente compreende um rastreador de estado 182, que é configurado para rastrear o estado do codificador aritmético, por exemplo, observando quais valores espectrais foram codificados previamente. O rastreador de estado 182 consequentemente provê uma informação de estado 184, por exemplo, um valor de estado designado com “s” ou “t” ou “c”. O codificador aritmético 170 também compreende um selecionador da tabela de frequências cumulativas 186, que é configurado para receber a informação de estado 184 e prover uma informação 188 que descreve a tabela de frequências cumulativas selecionada ao determinador de senha 180. Por exemplo, o selecionador da tabela de frequências cumulativas 186 pode prover um índice da tabela de frequências cumulativas “pki” que descreve qual tabela de frequências cumulativas, fora de

um conjunto de 96 tabelas de frequências cumulativas, é selecionado para uso pelo determinador de senha. De modo alternativo, o selecionador da tabela de frequências cumulativas 186 pode prover toda a tabela de frequências cumulativas selecionada ou uma sub-tabela ao determinador de senha. Assim, o determinador de senha 180 pode usar a tabela de frequências cumulativas selecionada ou sub-tabela para a provisão da senha acod_m[pki][m] do valor do plano de bits mais significativo m, de modo que a senha atual acod_m[pki][m] que codifica o valor do plano de bits mais significativo m seja dependente do valor de m e o índice da tabela de frequências cumulativas pki, e consequentemente na informação de estado corrente 184. Outros detalhes referentes ao processo de codificação e o formato de senha obtido serão descritos abaixo.

152. Deve ser observado, entretanto, que em algumas realizações, o rastreador de estado 182 pode ser idêntico a, ou tomar a funcionalidade, do rastreador de estado 750, do rastreador de estado 1050 ou do rastreador de estado 1250. Também deve ser observado que o selecionador da tabela de frequências cumulativas 186 pode, em algumas realizações, ser idêntico a, ou tomar a funcionalidade, do selecionador da regra de mapeamento 760, do selecionador da regra de mapeamento 1060, ou do selecionador da regra de mapeamento 1260. Além disso, o primeiro determinador de senha 180 pode, em algumas realizações, ser idêntico a, ou tomar a funcionalidade, da codificação do valor espectral 740.

153. O codificador aritmético 170 ainda compreende um extrator do 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 escalada e quantizada 152, se um ou mais dos valores espectrais a ser codificado exceder a faixa de valores que podem ser codificados utilizando o plano de bits mais significativo somente. Os planos de bits menos significativos podem compreender um ou mais bits, conforme desejado. Assim, o extrator do plano de bits menos significativo 189a provê uma informação do plano de bits menos significativo 189b. O codificador aritmético 170 também compreende um segundo determinador de senha 189c, que é configurado para receber a informação do plano de bits menos significativo 189d e prover, com base neste, 0, 1 ou mais senhas “acod_r” que representa 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 da codificação aritmética ou qualquer outro algoritmo de codificação a fim de derivar as senhas do plano de bit menos significativo “acod_r” da informação do plano de bits menos significativo 189b.

154. Deve ser observado aqui que o número de planos de bits menos significativos pode variar dependendo do valor dos valores espectrais escalados e quantizados 152, de modo que possa ter nenhum plano de bits menos significativo, se o valor espectral escalado e quantizado a ser codificado é comparavelmente pequeno, de modo que possa ter um plano de bits menos significativo se o valor espectral corrente escalado e quantizado a ser codificado for de uma faixa média e de modo que possa ter mais que um plano de bits menos significativo se o valor espectral escalado e quantizado a ser codificado tiver um valor comparavelmente grande.

155. Para resumir o mencionado acima, o codificador

aritmético 170 é configurado para codificar valores espectrais escalados e quantizados, que são descritos pela informação 152, que utiliza um processo de codificação hierárquico. O plano de bits mais significativo (que compreende, por exemplo, um, dois ou três bits por valor espectral) de um ou mais valores espectrais, é codificado para obter uma senha aritmética “acod_m[pki][m]” de um valor do plano de bits mais significativo m. Um ou mais planos de bits menos significativos (cada um dos planos de bits menos significativos que compreende, por exemplo, um, dois ou três bits) de um ou mais valores espectrais 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 esta finalidade, 96 diferentes tabelas de frequências cumulativas estão disponíveis para a codificação do valor m dependendo de um estado do codificador aritmético 170, ou seja, dependendo de valores espectrais previamente codificados. Assim, 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 DE REDEFINIÇÃO

156. O codificador de áudio 100 pode opcionalmente ser configurado para decidir se uma melhoria na taxa de bit pode ser obtida redefinindo o contexto, por exemplo, definindo o índice de estado a um valor padrão. Assim, o codificador de áudio 100 pode ser configurado para prover uma informação de reset (por exemplo, chamada “arith_reset_flag”) indicando se o contexto para a codificação aritmética é redefinida, e também indicando se o

contexto para a decodificação aritmética em um decodificador correspondente deve ser redefinido.

157. Detalhes referentes ao formato do fluxo de bits e às tabelas de frequência cumulativa aplicadas serão discutidos abaixo.

Decodificador de áudio de acordo com a figura 2

158. A seguir, um decodificador de áudio, de acordo com uma realização da invenção, será descrito. A figura 2 mostra um diagrama em blocos esquemático de tal decodificador de áudio 200.

159. 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 decodificada 212 com base no fluxo de bits 210.

160. O decodificador de áudio 200 compreende um deformatador de payload do 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 codificada de áudio de domínio de frequência 222. Por exemplo, o deformatador de payload do fluxo de bits 220 pode ser configurado para extrair do fluxo de bits 210 dados espectrais aritmeticamente decodificados como, por exemplo, uma senha aritmética “acod_m [pki][m]” que representa o valor do plano de bits mais significativo m de um valor espectral a, ou de diversos valores espectrais a, b, e uma senha “acod_r” que representa um conteúdo de um plano de bits menos significativo do valor espectral a, ou de diversos valores espectrais a, b, da

representação de áudio de domínio de frequência. Assim, a representação codificada de áudio de domínio de frequência 222 constitui (ou compreende) uma representação aritmeticamente codificada de valores espectrais. O deformatador de payload do fluxo de bits 220 é ainda configurado para extrair da informação adicional de controle do fluxo de bits, que não é mostrado na figura 2. Além disso, o deformatador de payload do fluxo de bits é opcionalmente configurado para extrair do fluxo de bits 210, uma informação de reset do estado 224, que é também designada como sinalizador de redefinição aritmético ou “arith_reset_flag”.

161. 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 codificada de áudio de domínio de frequência 220 e, opcionalmente, a informação de reset de estado 224. O decodificador aritmético 230 é também configurado para prover uma representação decodificada de áudio de domínio de frequência 232, que pode compreender uma representação decodificada de valores espectrais. Por exemplo, a representação decodificada de áudio de domínio de frequência 232 pode compreender uma representação decodificada de valores espectrais, que são descritos pela representação codificada de áudio de domínio de frequência 220.

162. O decodificador de áudio 200 também compreende um quantificador/remultiplicador de frequências inversas opcionais 240, que é configurado para receber a representação decodificada de áudio de domínio de frequência 232 e prover, com base neste, uma representação de áudio de domínio de frequência inversamente

quantizada e redimensionada 242.

163. 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 inversamente quantizada e redimensionada 242 e prover, com base neste, uma versão pré-processada 252 da representação de áudio de domínio de frequência inversamente quantizada e redimensionada 242. O decodificador de áudio 200 também compreende um transformador de sinal de domínio de frequência em domínio de tempo 260, que é também designado como um “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 inversamente quantizada e redimensionada 242 (ou, de modo alternativo, a representação de áudio de domínio de frequência inversamente quantizada e redimensionada 242 ou a representação decodificada de áudio de domínio de frequência 232) e prover, com base neste, uma representação de domínio de tempo 262 da informação de áudio. O transformador de sinal de domínio de frequência em domínio de tempo 260 pode, por exemplo, compreender um transformador para realizar uma transformação de cosseno discreto modificado inverso (IMDCT) e um janelamento apropriado (bem como outras funcionalidades auxiliares, como, por exemplo, uma sobreposição e adição).

164. 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 da informação de áudio e para obter a informação de áudio decodificada 212 que utiliza um pós-processamento de domínio de

tempo. Entretanto, se o pós-processamento for omitido, a representação de domínio de tempo 262 pode ser idêntica à informação de áudio decodificada 212.

165. Deve ser observado aqui que o quantificador/remultiplicador de frequências inversas 240, o pré-processador espectral 250, o transformador de sinal de domínio de frequência em domínio de tempo 260 e o pós-processador de domínio de tempo 270 podem ser controlados dependendo da informação de controle, que é extraída do fluxo de bits 210 pelo deformatador de payload do fluxo de bits 220.

166. Para resumir toda a funcionalidade do decodificador de áudio 200, uma representação decodificada de áudio de domínio de frequência 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 codificada de domínio de frequência 222 que utiliza o decodificador aritmético 230. Subsequentemente, o conjunto de, por exemplo, 1024 valores espectrais, que pode ser de coeficientes MDCT, é inversamente quantizado, redimensionado e pré-processado. Assim, um conjunto de valores espectrais pré-processados espectralmente inversamente quantizados e redimensionados (por exemplo, 1024 coeficientes MDCT) é obtido. Posteriormente, uma representação de domínio de tempo de uma estrutura de áudio é derivada do conjunto pré-processado de forma espectral, inversamente quantizado e redimensionado de valores de domínio de frequência (por exemplo, coeficientes MDCT). Assim, uma representação de domínio de tempo de uma estrutura de áudio é obtida. A representação de domínio de tempo de uma dada estrutura

de áudio pode ser combinada com as representações de domínio de tempo de estruturas de áudio prévias e/ou subsequentes. Por exemplo, uma sobreposição e adição entre as representações de domínio de tempo de estruturas de áudio subsequentes pode ser realizada a fim de suavizar as transações entre as representações de domínio de tempo das estruturas de áudio adjacentes e a fim de obter uma anulação de *aliasing*. Para detalhes referentes à reconstrução da informação de áudio decodificada 212 com base na representação decodificada de áudio de domínio de frequência de tempo 232, a referência é feita, por exemplo, ao Padrão Internacional ISO/IEC 14496-3, parte 3, subparte 4 onde uma discussão detalhada é dada. Entretanto, outros esquemas de sobreposição mais elaborada e anulação de *aliasing* podem ser utilizados.

167. A seguir, alguns detalhes referentes ao decodificador aritmético 230 serão descritos. O decodificador aritmético 230 compreende um determinador do plano de bits mais significativo 284, que é configurado para receber a senha aritmética acod_m [pki][m] que descreve o valor do plano de bits mais significativo m. O determinador do plano de bits mais significativo 284 pode ser configurado para utilizar a tabela de frequências cumulativas fora de um conjunto que compreende diversas 96 tabelas de frequências cumulativas para derivar o valor do plano de bits mais significativo m da senha aritmética “acod_m [pki][m]”.

168. O determinador do plano de bits mais significativo 284 é configurado para derivar valores 286 de um plano de bits mais significativo de um ou mais valores espectrais

com base na senha acod_m. O decodificador aritmético 230 ainda compreende um determinador do plano de bits menos significativo 288, que é configurado para receber uma ou mais senhas “acod_r” que representa um ou mais planos de bits menos significativos de um valor espectral. Assim, o determinador do 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 do plano de bits 292, que é configurado para receber os valores decodificados 286 do plano de bits mais significativo de um ou mais valores espectrais e os valores decodificados 290 de um ou mais planos de bits menos significativos dos valores espectrais se tais planos de bits menos significativos estão disponíveis para os valores correntes espectrais. Assim, o combinador do plano de bits 292 provê valores espectrais decodificados, que fazem parte da representação decodificada de áudio de domínio de frequência 232. Naturalmente, o decodificador aritmético 230 é tipicamente configurado para prover diversos valores espectrais a fim de obter um conjunto completo de valores espectrais decodificados associados a uma estrutura corrente do conteúdo de áudio.

169. O decodificador aritmético 230 ainda compreende um selecionador da tabela de frequências cumulativas 296, que é configurado para selecionar uma das 96 tabelas de frequências cumulativas dependendo 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 dependendo dos valores espectrais previamente decodificados. A informação de

estado pode opcionalmente ser redefinida a uma informação de estado padrão em resposta à informação de reset de estado 224. Assim, o selecionador da tabela de frequências cumulativas 296 é configurado para prover um índice (por exemplo, pki) da tabela de frequências cumulativas selecionada, ou a tabela de frequências cumulativas selecionada ou a própria sub-tabela, para aplicação na decodificação do valor do plano de bits mais significativo m dependendo da senha “acod_m”.

170. Para resumir a funcionalidade do decodificador de áudio 200, o decodificador de áudio 200 é configurado para receber uma representação eficientemente codificada de áudio de domínio de frequência da taxa de bits 222 e para obter uma representação decodificada de áudio de domínio de frequência com base neste. No decodificador aritmético 230, que é utilizado para obter a representação decodificada de áudio de domínio de frequência 232 com base na representação codificada de áudio de domínio de frequência 222, uma probabilidade de diferentes combinações de valores do plano de bits mais significativo de valores espectrais adjacentes é explorada utilizando um decodificador aritmético 280, que é configurado para aplicar a tabela de frequências cumulativas. Em outras palavras, as dependências estatísticas entre os valores espectrais são explorados selecionando diferentes tabelas de frequências cumulativas fora de um conjunto que compreende 96 diferentes tabelas de frequências cumulativas dependendo de um índice de estado 298, que é obtido observando os valores espectrais decodificados previamente calculados.

171. Deve ser observado que o rastreador de estado 299 pode ser idêntico a, ou pode tomar a funcionalidade, do rastreador

de estado 826, do rastreador de estado 1126, ou do rastreador de estado 1326. O selecionador da tabela de frequências cumulativas 296 pode ser idêntico a ou pode tomar a funcionalidade, do selecionador da regra de mapeamento 828, do selecionador da regra de mapeamento 1128, ou do selecionador da regra de mapeamento 1328. O determinador do plano de bits mais significativo 284 pode ser idêntico a, ou pode tomar a funcionalidade, do determinador do valor espectral 824.

Visão geral da ferramenta de codificação espectral silenciosa

172. A seguir, detalhes referentes à codificação e decodificação do algoritmo, que é realizada, por exemplo, pelo codificador aritmético 170 e pelo decodificador aritmético 230, serão explicados.

173. O foco deve ser colocado na descrição da decodificação do algoritmo. Deve ser observado, entretanto, que uma codificação do algoritmo correspondente poder ser realizada de acordo com os ensinamentos da decodificação do algoritmo, em que mapeamentos entre valores espectrais codificados e decodificados são inversos, e em que o cálculo do valor de índice de regra de mapeamento é substancialmente idêntico. Em um codificador, os valores espectrais codificados assumem o lugar dos valores espectrais decodificados. Ainda, os valores espectrais a ser codificados assumem o lugar dos valores espectrais a ser decodificados.

174. Deve ser observado que a decodificação, que será discutida a seguir, é utilizada a fim de permitir a chamada “codificação espectral silenciosa” de valores espectrais escalados

e quantizados tipicamente pós-processados. A codificação espectral silenciosa é utilizada em um conceito de codificação/decodificação de áudio (ou em qualquer outro conceito de codificação/decodificação) para ainda reduzir a redundância do espectro quantizado, que é obtido, por exemplo, por um transformador de domínio de tempo em domínio de frequência com compactação de energia. O esquema espectral de codificação silenciosa, que é utilizado nas realizações da invenção, tem como base uma codificação aritmética com um contexto dinamicamente adaptado.

175. Em algumas realizações, de acordo com a invenção, o esquema espectral de codificação silenciosa tem como base 2 tuplos, ou seja, dois coeficientes espectrais próximos são combinados. Cada tuplo duplo é dividido no sinal, o plano de bit a bit mais significativo, e os planos de bits menos significativos restantes. A codificação silenciosa para o plano de bit a bit mais significativo m utiliza tabelas de frequências cumulativas dependentes de contexto derivadas de quatro tuplos duplos previamente decodificados. A codificação silenciosa é alimentada pelos valores espectrais quantizados e utiliza tabelas de frequências cumulativas dependentes de contexto derivadas de quatro tuplos duplos próximos previamente decodificados. Aqui, próximo em tempo e frequência é considerado, 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 comprimento variável (e pelo decodificador aritmético para derivar valores decodificados de um código binário de comprimento variável).

176. Por exemplo, o codificador aritmético 170 produz um código binário para um determinado conjunto de símbolos e suas respectivas probabilidades (ou seja, dependendo das respectivas probabilidades). O código binário é gerado pelo mapeamento de um intervalo de probabilidade, onde o conjunto de símbolos fica em uma senha.

177. A codificação silenciosa do plano de bits menos significativo r restante utiliza uma única tabela de frequências cumulativas. As frequências cumulativas correspondem, por exemplo, a uma distribuição uniforme dos símbolos que ocorrem nos planos de bits menos significativos, ou seja, é esperado que haja a mesma probabilidade que um 0 ou um 1 ocorre nos planos de bits menos significativos.

178. A seguir, outra breve visão geral da ferramenta de codificação espectral silenciosa será dada. A codificação espectral silenciosa é utilizada para ainda reduzir a redundância do espectro quantizado. O esquema espectral de codificação silenciosa tem como base uma codificação aritmética, com um contexto dinamicamente adaptado. A codificação silenciosa é alimentada pelos valores espectrais quantizados e utiliza as tabelas de frequências cumulativas dependentes de contexto derivadas de, por exemplo, quatro tuplos duplos próximos previamente decodificados de valores espectrais. Aqui, próximo, em tempo e frequência, é considerado conforme ilustrado na figura 4. As tabelas de frequências cumulativas são então utilizadas pelo codificador aritmético para gerar um código binário de comprimento variável.

179. O codificador aritmético produz um código binário

para um determinado conjunto de símbolos e suas respectivas probabilidades. O código binário é gerado pelo mapeamento de um intervalo de probabilidade, onde o conjunto de símbolos permanece, em uma senha.

Processo de decodificação

11.1 Visão geral do processo de decodificação

180. A seguir, uma visão geral do processo de codificação de um valor espectral será apresentada, tendo como referência a figura 3, que mostra uma representação do código do pseudo-programa do processo de decodificação de diversos valores espectrais.

181. O processo de decodificação de diversos valores espectrais compreende uma inicialização 310 de um contexto. A inicialização 310 do contexto compreende uma derivação do contexto corrente de um contexto prévio, que utiliza a função “arith_map_context(N, arith_reset_flag)”. A derivação do contexto corrente de um contexto prévio pode seletivamente compreender uma redefinição do contexto. A redefinição do contexto e a derivação do contexto corrente de um contexto prévio serão discutidas abaixo.

182. A decodificação de diversos valores espectrais também compreende uma iteração de uma decodificação do valor espectral 312 e uma atualização de contexto 313, cuja atualização de contexto 313 é realizada pela função “arith_update_context(i, a,b)” que é descrita abaixo. A decodificação do valor espectral 312 e a atualização de contexto 312 são repetidas lg/2 vezes, em que lg/2 indica o número de 2 tuplos de valores espectrais a ser decodificados (por exemplo, para uma estrutura de áudio), a menos

que o símbolo chamado “ARITH_STOP” seja detectado. Além disso, a decodificação de um conjunto de valores espectrais lg também compreende uma decodificação de sinais 314 e uma etapa de acabamento 315.

183. A decodificação 312 de um tuplo de valores espectrais compreende um cálculo do valor de contexto 312a, uma decodificação do plano de bits mais significativo 312b, uma detecção do símbolo de parada aritmética 312c, uma adição do plano de bits menos significativo 312d, e uma atualização da matriz 312e.

184. O cálculo do valor de estado 312a compreende uma chamada da função “arith_get_context(c,i,N)” conforme mostrado, por exemplo, na figura 5c ou 5d. Assim, um valor (de estado) de contexto corrente numérico c é provido como um valor de retorno da função chamada da função “arith_get_context(c,i,N)”. Como pode ser visto, o valor de contexto prévio numérico (também designado como “c”), que serve como uma variável de entrada na função “arith_get_context(c,i,N)”, é atualizada para obter, como um valor de retorno, o valor de contexto corrente numérico c.

185. A decodificação do plano de bits mais significativo 312b compreende uma execução iterativa de uma decodificação de algoritmo 312ba, e uma derivação 312bb de valores a,b do valor de resultado m do algoritmo 312ba. Na preparação do algoritmo 312ba, a variável lev é inicializada a zero. O algoritmo 312ba é repetido, até que uma instrução de “interrupção” (ou condição) seja atingida. O algoritmo 312ba compreende um cálculo de um índice de estado “pki” (que também serve como um índice da tabela de frequências cumulativas) dependendo do valor de contexto

corrente numérico c, e também dependendo do valor de nível “esc_nb” que utiliza a função “arith_get_pk()”, que é discutido abaixo (e realizações destas são mostradas, por exemplo, nas figuras. 5e e 5f). O algoritmo 312ba também compreende a seleção da tabela de frequências cumulativas dependendo do índice de estado “pki”, que é retornado pela chamada da função “arith_get_pk”, em que uma variável “cum_freq” pode ser definida a um endereço inicial de uma das 96 tabelas de frequências cumulativas (ou sub-tabelas) dependendo do índice de estado “pki”. Uma variável “cfl” pode também ser inicializada em um comprimento da tabela de frequências cumulativas selecionada (ou uma sub-tabela), que é, por exemplo, igual a um número de símbolos no alfabeto, ou seja, o número de diferentes valores que pode ser decodificado. O comprimento de todas as tabelas de frequências cumulativas (ou sub-tabelas) de “ari_cf_m[pki=0][17]” a “ari_cf_m[pki=95][17]” disponíveis para a decodificação do valor do plano de bits mais significativo m é 17, como 16 diferentes valores do plano de bits mais significativo e um símbolo de escape (“ARITH_ESCAPE”) podem ser decodificados.

186. Subsequentemente, um valor do plano de bits mais significativo m pode ser obtido executando a função “arith_decode()”, levando em consideração a tabela de frequências cumulativas selecionada (descrita pela variável “cum_freq” e a variável “cfl”). Ao derivar o valor do plano de bits mais significativo m, os bits chamados “acod_m” do fluxo de bits 210 podem ser avaliados (veja, por exemplo, a figura 6g ou a figura 6h).

187. O algoritmo 312ba também compreende verificar se

o valor do plano de bits mais significativo m é igual a um símbolo de escape “ARITH_ESCAPE”, ou não. Se o valor do plano de bits mais significativo m não for igual ao símbolo de escape aritmético, o algoritmo 312ba é abortado (condição de “interrupção”) e as instruções restantes do algoritmo 312ba são então puladas. Assim, a execução do processo é continuada com a definição do valor b e do valor a na etapa 312bb. Em contraste, se o valor do plano de bits mais significativo decodificado m for idêntico ao símbolo de escape aritmético, ou “ARITH_ESCAPE”, o valor de nível “lev” é aumentado por um. O valor de nível “esc_nb” é definido para ser igual ao valor de nível “lev”, a menos que a variável “lev” seja maior que sete, cujo caso, a variável “esc_nb” é definida para ser igual a sete. Conforme mencionado, o algoritmo 312ba é então repetido até que o valor do plano de bits mais significativo decodificado m seja diferente do símbolo de escape aritmético, em que um contexto modificado seja utilizado (por causa do parâmetro de entrada da função “arith_get_pk()” ser adaptado dependendo do valor da variável “esc_nb”).

188. Logo que o plano de bits mais significativo é decodificado utilizando execução de um período ou a execução iterativa do algoritmo 312ba, ou seja, um valor do plano de bits mais significativo m diferente do símbolo de escape aritmético foi decodificado, o variável do valor espectral “b” é definido para ser igual a diversos (por exemplo, 2) bits mais significativos do valor do plano de bits mais significativo m, e a variável do valor espectral “a” é definida (por exemplo, 2) nos bits menos significativos do valor do plano de bits mais significativo m. Detalhes referentes a esta funcionalidade podem ser vistos, por

exemplo, no número de referência 312bb.

189. Subsequentemente, é verificado na etapa 312c, se um símbolo de parada aritmética está presente. Este é o caso se o valor do plano de bits mais significativo m for igual a zero e a variável “lev” for maior que zero. Assim, uma condição de parada aritmética é sinalizada por uma condição “incomum”, na qual o valor do plano de bits mais significativo m é igual a zero, enquanto a variável “lev” indica que um peso numérico aumentado está associado ao valor do plano de bits mais significativo m. Em outras palavras, uma condição de parada aritmética é detectada se o fluxo de bits indica que um peso numérico aumentado, maior que um peso numérico mínimo, deve ser dado a um valor do plano de bits mais significativo que é igual a zero, que é uma condição que não ocorre em uma situação de codificação normal. Em outras palavras, uma condição de parada aritmética é sinalizada se um símbolo de escape aritmético codificado for seguido por um valor do plano de bits mais significativo codificado de 0.

190. Depois de uma avaliação se houver uma condição de parada aritmética, que é realizada na etapa 212c, os planos de bits menos significativos são obtidos, por exemplo, conforme mostrado no número de referência 212d na figura 3. Para cada plano de bits menos significativo, dois valores binários são decodificados. Um dos valores binários está associado à variável a (ou o primeiro valor espectral de um tuplo de valores espectrais) e um dos valores binários está associado à variável b (ou um segundo valor espectral de um tuplo de valores espectrais). Um número de planos de bits menos significativos é designado pela variável lev.

191. Na decodificação de um ou mais planos de bits menos significativos (se houver) um algoritmo 212da é realizado de forma iterativa, em que um número de execuções do algoritmo 212da é determinado pela variável “lev”. Deve ser observado aqui que a primeira iteração do algoritmo 212da é realizada com base nos valores das variáveis a, b conforme definido na etapa 212bb. Outras iterações do algoritmo 212da devem ser realizadas com base nos valores da variável atualizada da variável a, b.

192. No início de uma iteração, uma tabela de frequências cumulativas é selecionada. Subsequentemente, uma decodificação aritmética é realizada para obter um valor de uma variável r, em que o valor da variável r descreve diversos bits menos significativos, por exemplo, um bit menos significativo associado à variável a e um bit menos significativo associado à variável b. A função “ARITH_DECODE” é utilizada para obter o valor r, em que a tabela de frequências cumulativas “arith_cf_r” é utilizada para a decodificação aritmética.

193. Subsequentemente, os valores das variáveis a e b são atualizados. Para esta finalidade, a variável a é mudada à esquerda por um bit, e o bit menos significativo da variável mudada a é definido o valor definido pelo bit menos significativo do valor r. A variável b é mudada à esquerda por um bit, e o bit menos significativo da variável mudada b é definido o valor definido por bit 1 da variável r, em que o bit 1 da variável r tem um peso numérico de 2 na representação binária da variável r. O algoritmo 412ba é então repetido até que todos os bits menos significativos sejam decodificados.

194. Depois da decodificação dos planos de bits menos

significativos, uma matriz “x_ac_dec” é atualizada em que os valores das variáveis a,b são armazenados nas entradas da dita matriz tendo índices da matriz 2*i e 2*i+1.

195. Subsequentemente, o estado de contexto é atualizado chamando a função “arith_update_context(i,a,b)”, detalhes que serão explicados abaixo tendo como referência a figura 5g.

196. Subsequente à atualização do estado de contexto, que é realizada na etapa 313, os algoritmos 312 e 313 são repetidos, até que a variável de execução i atinja o valor de lg/2 ou uma condição de parada aritmética seja detectada.

197. Subsequentemente, um algoritmo de acabamento “arith_finish()” é realizado, como pode ser visto no número de referência 315. Detalhes do algoritmo de acabamento “arith_finish()” serão descritos abaixo tendo como referência a figura 5m.

198. Subsequente ao algoritmo de acabamento 315, os sinais dos valores espectrais são decodificados utilizando o algoritmo 314. Como pode ser visto, os sinais dos valores espectrais que são diferentes de zero são individualmente codificados. No algoritmo 314, os sinais são lidos para todos os valores espectrais tendo índices i entre i=0 e i=lg-1 que não são zero. Para cada valor espectral não zero tendo um índice do valor espectral i entre i=0 e i=lg-1, um valor (tipicamente um único bit) s é lido do fluxo de bits. Se o valor de s, que é lido do fluxo de bits é igual a 1, o sinal do dito valor espectral é invertido. Para esta finalidade, o acesso é feito à matriz “x_ac_dec”, para determinar se o valor espectral tendo o índice i

é igual a zero e para atualizar o sinal dos valores espectrais decodificados. Entretanto, deve ser observado que os sinais das variáveis a, b não são mudados na decodificação de sinal 314.

199. Realizando o algoritmo de acabamento 315 antes da decodificação de sinais 314, é possível redefinir todos os bins necessários depois de um símbolo ARITH_STOP.

200. Deve ser observado aqui que o conceito para obter os valores dos planos de bits menos significativos não é de relevância particular em algumas realizações de acordo com a presente invenção. Em algumas realizações, a decodificação de quaisquer planos de bits menos significativos pode ser omitida. De modo alternativo, diferentes algoritmos de decodificação podem ser utilizados para esta finalidade.

11.2 Ordem de decodificação de acordo com a figura 4

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

202. Os coeficientes espectrais quantizados “x_ac_dec[]” são silenciosamente codificados e transmitidos (por exemplo, no fluxo de bits) começando do coeficiente com frequência mais baixa e continuando até o coeficiente com frequência mais alta.

203. Consequentemente, os coeficientes espectrais quantizados “x_ac_dec[]” são silenciosamente codificados começando do coeficiente com frequência mais baixa e continuando ao coeficiente com frequência mais alta. Os coeficientes espectrais quantizados são decodificados por grupos de dois coeficientes sucessivos (por exemplo, adjacentes na frequência) a e b reunindo

em um tuplo duplo (a,b) (também designado como {a,b}). Deve ser observado aqui que os coeficientes espectrais quantizados são às vezes designados como “qdec”.

204. Os coeficientes decodificados “x_ac_dec[]” para um modo de domínio de frequência (por exemplo, coeficientes decodificados para uma codificação de áudio avançado, por exemplo, obtido utilizando uma transformação de cosseno discreto modificado, conforme discutido em ISO/IEC 14496, parte 3, subparte 4) são então armazenados em uma matriz “x_ac_quant[g][win][sfb][bin]”. A ordem de transmissão das senhas de codificação silenciosa é de modo que quando elas são decodificadas na ordem recebida e armazenada na matriz, “bin” é o índice que aumenta mais rapidamente, e “g” é o índice que aumenta mais lentamente. Dentro de uma senha, a ordem da decodificação é a,b.

205. Os coeficientes decodificados “x_ac_dec[]” para a excitação codificada por transformação (TCX) são armazenados, por exemplo, diretamente em uma matriz “x_tcx_invquant[win][bin]”, e uma ordem da transmissão da senha de codificação silenciosa é de modo que quando elas são decodificadas na ordem recebida e armazenada na matriz “bin” é o índice que aumenta mais rapidamente, e “win” é o índice que aumenta mais lentamente. Dentro de uma senha, uma ordem da decodificação é a, b. Em outras palavras, se os valores espectrais descrevem uma excitação codificada por transformação do filtro de previsão linear de um codificador de discurso, os valores espectrais a, b são associados às frequências adjacentes e de aumento da excitação codificada por transformação. Os coeficientes espectrais associados a uma

frequência inferior são tipicamente codificados e decodificados antes de um coeficiente espectral associado a uma frequência mais alta.

206. Notavelmente, o decodificador de áudio 200 pode ser configurado para aplicar a representação decodificada de domínio de frequência 232, que é provida pelo decodificador aritmético 230, para uma geração “direta” de uma representação do sinal de áudio de domínio de tempo que utiliza uma transformação do sinal de domínio de frequência em domínio de tempo e para uma provisão “indireta” de uma representação do sinal de áudio de domínio de tempo que utiliza um decodificador de domínio de frequência em domínio de tempo e um filtro de previsão linear excitado pela saída do transformador de sinal de domínio de frequência em domínio de tempo.

207. Em outras palavras, o decodificador aritmético, a funcionalidade que é discutida aqui em detalhes, é bem adequada de valores espectrais de decodificação 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 (ou sintetizar) um sinal de discurso codificado no domínio de previsão linear. Assim, o decodificador aritmético é bem adequado para uso em um decodificador de áudio que pode lidar com o conteúdo de áudio codificado de domínio de frequência e conteúdo de áudio codificado de domínio de frequência previsivo linear (modo de excitação codificada por transformação-domínio de previsão linear).

11.3 Inicialização de contexto de acordo com as figuras 5a e 5b

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

209. A inicialização de contexto compreende um mapeamento entre um contexto anterior e um contexto corrente de acordo com o algoritmo “arith_map_context()”, um primeiro exemplo do que é mostrado na figura 5a e um segundo exemplo do que é mostrado na figura 5b.

210. Como pode ser visto, o contexto corrente é armazenado em uma variável global “q[2][n_context]” que toma a forma de uma matriz tendo uma primeira dimensão de 2 e uma segunda dimensão de “n_context”. Um contexto anterior pode opcionalmente (mas não necessariamente) ser armazenado em uma variável “qs[n_context]” que toma a forma da tabela tendo uma dimensão de “n_context” (se for utilizado).

211. Tendo como referência o algoritmo de exemplo “arith_map_context” na figura 5a, a variável de entrada N descreve um comprimento de uma janela corrente e a variável de entrada “arith_reset_flag” indica se o contexto deve ser redefinido. Além disso, a variável global “previous_N” descreve um comprimento de uma janela prévia. Deve ser observado aqui que tipicamente um número de valores espectrais associado a uma janela é, pelo menos aproximadamente, igual à metade de um comprimento da dita janela em termos de amostras de domínio de tempo. Além disso, deve ser observado que um número de 2 tuplos de valores espectrais é, consequentemente, pelo menos aproximadamente igual a um quarto de

um comprimento da dita janela em termos de amostras de domínio de tempo.

212. Tendo como referência o exemplo da figura 5a, o mapeamento do contexto pode ser realizado de acordo com o algoritmo “arith_map_context()”. Deve ser observado aqui que a função “arith_map_context()” define as entradas “q[0][j]” da matriz de contexto corrente q a zero para j=0 aj=N/4-1, se o indicador “arith_reset_flag” estiver ativo e consequentemente indica que o contexto deve ser redefinido. Caso contrário, ou seja, se o indicador “arith_reset_flag” estiver inativo, as entradas “q[0][j]” da matriz de contexto corrente q são derivadas das entradas “q[1][k]” da matriz de contexto corrente q. Deve ser observado que a função “arith_map_context()” de acordo com a figura 5a define as entradas “q[0][j]” da matriz de contexto corrente q aos valores “q[1][k]” da matriz de contexto corrente q, se o número de valores espectrais associado à corrente (por exemplo, codificado pelo domínio de frequência) a estrutura de áudio for idêntico ao número de valores espectrais associado à estrutura de áudio anterior para j=k=0 a j=k=N/4-1.

213. Um mapeamento mais complicado é realizado se o número de valores espectrais que está associado à estrutura corrente de áudio for diferente do número de valores espectrais que está associado à estrutura de áudio anterior. Entretanto, detalhes referentes ao mapeamento neste caso não são particularmente relevantes para esta ideia principal da presente invenção, de modo que a referência é feita ao código do pseudo-programa da figura 5a para detalhes.

214. Além disso, um valor de inicialização para o

valor de contexto corrente numérico c é retornado pela função “arith_map_context()”. Este valor de inicialização é, por exemplo, igual ao valor da entrada “q[0][0]” mudada à esquerda em 12 bits. Assim, o valor de contexto (corrente) numérico c é apropriadamente inicializado para uma atualização iterativa.

215. Além disso, a figura 5b mostra outro exemplo de um algoritmo “arith_map_context()” que pode de modo alternativo ser utilizado. Para detalhes, a referência é feita ao código do pseudo-programa na figura 5b.

216. Para resumir o mencionado acima, o indicador “arith_reset_flag” determina se o contexto deve ser redefinido. Se o indicador for verdadeiro, um sub-algoritmo de redefinição 500a do algoritmo “arith_map_context()” é chamado. De modo alternativo, entretanto, se o indicador “arith_reset_flag” estiver inativo (que indica que nenhuma redefinição do contexto deve ser realizada), o processo de decodificação começa com uma fase de inicialização onde o vetor do elemento de contexto (ou matriz) q é atualizado copiando e mapeando os elementos de contexto da estrutura prévia armazenados em q[1][] em q[0][]. Os elementos de contexto dentro de q são armazenados em 4 bits por 2 tuplos. A cópia e/ou mapeamento do elemento de contexto são realizados em um sub-algoritmo 500b.

217. No exemplo da figura 5b, o processo de decodificação começa com uma fase de inicialização onde um mapeamento é feito entre o contexto anterior salvo armazenado em qs e o contexto da estrutura corrente q. O contexto anterior qs é armazenado em 2 bits por linha de frequência.

11.4 Cálculo do valor de estado de acordo com

as figuras 5c e 5d

218. A seguir, o cálculo do valor de estado 312a será descrito em mais detalhes.

219. Um primeiro algoritmo de exemplo será descrito tendo como referência a figura 5c e um segundo algoritmo de exemplo será descrito tendo como referência a figura 5d.

220. Deve ser observado que o valor de contexto corrente numérico c (conforme mostrado na figura 3) pode ser obtido como um valor de retorno da função “arith_get_context(c,i,N)”, uma representação do código do pseudo-programa que é mostrada na figura 5c. De modo alternativo, entretanto, o valor de contexto corrente numérico c pode ser obtido como um valor de retorno da função “arith_get_context(c,i)”, uma representação do código do pseudo-programa que é mostrada na figura 5d.

221. Referente ao cálculo do valor de estado, a referência é também feita à figura 4, que mostra o contexto utilizado para uma avaliação de estado, ou seja, para o cálculo de um valor de contexto corrente numérico c. A figura 4 mostra uma representação bidimensional de valores espectrais, sobre o tempo e frequência. Uma abscissa 410 descreve o tempo, e uma ordenada 412 descreve a frequência. Como pode ser visto na figura 4, um tuplo 420 de valores espectrais para decodificar (preferivelmente que utiliza o valor de contexto corrente numérico), 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 tempo cujos valores espectrais do tuplo 120, tendo o índice de frequência i,

devem ser decodificados. 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 tuplo 420 de valores espectrais serem decodificados, e o tuplo 430 de valores espectrais é considerado para o contexto que é utilizado para a decodificação do tuplo 420 de valores espectrais. De forma semelhante, um tuplo 440 de valores espectrais tendo um índice de tempo t0-1 e um índice de frequência de i-1, um tuplo 450 de valores espectrais tendo um índice de tempo t0-1 e um índice de frequência de i, e um tuplo 460 de valores espectrais tendo um índice de tempo t0-1 e um índice de frequência de i+1, já é decodificado antes de o tuplo 420 de valores espectrais ser decodificado, e são considerados para a determinação do contexto, que é utilizado para decodificar o tuplo 420 de valores espectrais. Os valores espectrais (coeficientes) já decodificados no tempo quando os valores espectrais do tuplo 420 são decodificados e considerados para o contexto são mostrados por um quadrado sombreado. Em contraste, alguns outros valores espectrais já decodificados (no tempo quando os valores espectrais do tuplo 420 são decodificados), mas não considerados para o contexto (para a decodificação dos valores espectrais do tuplo 420) são representados por quadrados tendo linhas tracejadas, e outros valores espectrais (que ainda não são decodificados no tempo quando os valores espectrais do tuplo 420 são decodificados) são mostrados por círculos tendo linhas tracejadas. Os tuplos representados por quadrados tendo linhas tracejadas e os tuplos representados por círculos tendo linhas tracejadas não são utilizados para determinar o contexto de decodificação dos valores

espectrais do tuplo 420.

222. Entretanto, deve ser observado que alguns destes valores espectrais, que não são utilizados para o cálculo “regular” ou “normal” do contexto de decodificação dos valores espectrais do tuplo 420 podem, entretanto, ser avaliados para a detecção de diversos valores espectrais adjacentes previamente decodificados que cumprem, individualmente ou juntos, uma condição predeterminada referente às suas magnitudes. Detalhes referentes a esta questão serão discutidos abaixo.

223. Agora com referência à figura 5c, detalhes do algoritmo “arith_get_context(c,i,N)” serão descritos. A figura 5c mostra a funcionalidade da dita função “arith_get_context(c,i,N)” na forma de um código do pseudo-programa, que utiliza as convenções da linguagem C e/ou linguagem C++ bem conhecidas. Assim, outros detalhes referentes ao cálculo do valor de contexto corrente numérico “c” que é realizado pela função “arith_get_context(c,i,N)” serão descritos.

224. Deve ser observado que a função “arith_get_context(c,i,N)” recebe, como variáveis de entrada, um “contexto de estado antigo”, que pode ser descrito por um valor de contexto prévio numérico c. A função “arith_get_context(c,i,N)” também recebe, como uma variável de entrada, um índice i de um tuplo duplo de valores espectrais para decodificar. O índice i é tipicamente um índice de frequência. Uma variável de entrada N descreve um comprimento da janela de uma janela, na qual os valores espectrais são decodificados.

225. A função “arith_get_context(c,i,N)” provê, como um valor de saída, uma versão atualizada da variável de entrada c,

que descreve um contexto de estado atualizado, e que pode ser considerado como um valor de contexto corrente numérico. Para resumir, a função “arith_get_context(c,i,N)” recebe um valor de contexto prévio numérico c como uma variável de entrada e provê uma versão atualizada deste, que é considerada como um valor de contexto corrente numérico. Além disso, a função “arith_get_context” considera as variáveis i, N, e também avalia a matriz “global” q[][].

226. Referente aos detalhes da função “arith_get_context(c,i,N)”, deve ser observado que a variável c, que inicialmente representa o valor de contexto prévio numérico em uma forma binária, é mudada à direita em 4 bits em uma etapa 504a. Assim, os quatro bits menos significativos do valor de contexto prévio numérico (representado pela variável de entrada c) são descartados. Ainda, os pesos numéricos de outros bits dos valores de contexto prévios numéricos são reduzidos, por exemplo, um fator de 16.

227. Além disso, se o índice i do tuplo duplo for menor que N/4-1, ou seja, não assume um valor máximo, o valor de contexto corrente numérico é modificado em que o valor da entrada q[0][i+1] é adicionado aos bits 12 a 15 (ou seja, aos bits tendo um peso numérico de 2^{12}, 2^{13}, 2^{14}, e 2^{15}) do valor de contexto mudado que é obtido na etapa 504a. Para esta finalidade, a entrada q[0][i+1] da matriz q[][] (ou, mais precisamente, uma representação binária do valor representado pela dita entrada) é mudada à esquerda em 12 bits. A versão mudada do valor representado pela entrada q[0][i+1] é então adicionada ao valor de contexto c, que é derivado na etapa 504a, ou seja, a representação

numérica do valor de contexto prévio numérico um mudado por bit (mudado à direita em 4 bits). Deve ser observado aqui que a entrada q [0][i+1] da matriz q[][] representa um valor da sub-região associado a uma parte prévia do conteúdo de áudio (por exemplo, uma parte do conteúdo de áudio tendo índice de tempo t0-1, conforme definido com referência à figura 4), e com uma frequência mais alta (por exemplo, uma frequência tendo um índice de frequência i+1, conforme definido com referência à figura 4) que o tuplo de valores espectrais a ser atualmente decodificado (que utiliza o valor de contexto corrente numérico c emitido pela função “arith_get_context(c,i,N)”). Em outras palavras, se o tuplo 420 de valores espectrais deve ser decodificado utilizando o valor de contexto corrente numérico, a entrada q[0][i+1] pode ter como base o tuplo 460 de valores espectrais previamente decodificados.

228. Uma adição seletiva da entrada q[0][i+1] da matriz q[][] (mudada à esquerda em 12 bits) é mostrada no número de referência 504b. Como pode ser visto, a adição do valor representado pela entrada q[0][i+1] é naturalmente realizada somente se o índice de frequência i não designar um tuplo de valores espectrais tendo o índice de frequência mais alta i=N/4-1.

229. Subsequentemente, em uma etapa 504c, uma operação Boolean AND- é realizada, na qual o valor da variável c é AND-combinado com um valor hexadecimal de 0xFFF0 para obter um valor atualizado da variável c. Realizando tal operação AND, os quatro bits menos significativos da variável c são efetivamente definidos a zero.

230. Em uma etapa 504d, o valor da entrada q[1][i-1] é adicionado ao valor da variável c, que é obtido pela etapa 504c,

para então atualizar o valor da variável c. Entretanto, a dita atualização da variável c na etapa 504d é somente realizada se o índice de frequência i do tuplo duplo para decodificar for maior que zero. Deve ser observado que a entrada q[1][i-1] é um valor da sub-região de contexto com base em um tuplo de valores espectrais previamente decodificados da parte corrente do conteúdo de áudio para frequências menores que as frequências dos valores espectrais a ser decodificados que utilizam o valor de contexto corrente numérico. Por exemplo, a entrada q[1][i-1] da matriz q[][] pode ser associada ao tuplo 430 tendo o índice de tempo t0 e o índice de frequência i-1, se for assumido que o tuplo 420 de valores espectrais deve ser decodificado utilizando o valor de contexto corrente numérico retornado pela presente execução da função “arith_get_context(c,i,N)”.

231. Para resumir, os bits 0, 1, 2, e 3 (ou seja, uma parte de quatro bits menos significativos) do valor de contexto prévio numérico são descartados na etapa 504a alternando-os da representação numérica binária do valor de contexto prévio numérico. Além disso, os bits 12, 13, 14, e 15 da variável mudada c (ou seja, do valor de contexto mudado prévio numérico) são definidos para ter valores definidos pelo valor da sub-região de contexto q[0][i+1] na etapa 504b. Os bits 0, 1, 2, e 3 do valor de contexto mudado prévio numérico (ou seja, bits 4, 5, 6, e 7 do valor de contexto prévio numérico original) são sobrescritos pelo valor da sub-região de contexto q[1][i-1] nas etapas 504c e 504d.

232. Consequentemente, pode ser dito que os bits 0 a 3 do valor de contexto prévio numérico representam o valor da sub-região de contexto associado ao tuplo 432 de valores espectrais,

os bits 4 a 7 do valor de contexto prévio numérico representam o valor da sub-região de contexto associado a um tuplo 434 de valores espectrais previamente decodificados, os bits 8 a 11 do valor de contexto prévio numérico representam o valor da sub-região de contexto associado ao tuplo 440 dos valores espectrais previamente decodificados e os bits 12 a 15 do valor de contexto prévio numérico representam um valor da sub-região de contexto associado ao tuplo 450 de valores espectrais previamente decodificados. O valor de contexto prévio numérico, que é inserido na função “arith_get_context(c,i,N)”, é associado a uma decodificação do tuplo 430 de valores espectrais.

233. O valor de contexto corrente numérico, que é obtido como uma variável de saída da função “arith_get_context(c,i,N)”, é associado a uma decodificação do tuplo 420 de valores espectrais. Assim, os bits 0 a 3 dos valores de contexto corrente numérico descrevem o valor da sub-região de contexto associado ao tuplo 430 dos valores espectrais, os bits 4 a 7 do valor de contexto corrente numérico descrevem o valor da sub-região de contexto associado ao tuplo 440 de valores espectrais, os bits 8 a 11 do valor de contexto corrente numérico descrevem o valor numérico da sub-região associado ao tuplo 450 de valor espectral e os bits 12 a 15 do valor de contexto corrente numérico descreveu o valor da sub-região de contexto associado ao tuplo 460 de valores espectrais. Assim, pode ser visto que uma parte do valor de contexto prévio numérico, a saber, os bits 8 a 15 do valor de contexto prévio numérico, são também incluídos no valor de contexto corrente numérico, como os bits 4 a 11 do valor de contexto corrente numérico. Em contraste, os bits 0 a 7 do

valor de contexto corrente prévio numérico são descartados ao derivar a representação numérica do valor de contexto corrente numérico da representação numérica do valor de contexto prévio numérico.

234. Em uma etapa 504e, a variável c que representa o valor de contexto corrente numérico é seletivamente atualizada se o índice de frequência i do tuplo duplo para decodificar for maior que um número predeterminado de, por exemplo, 3. Neste caso, ou seja, se i for maior que 3, é determinado se a soma dos valores da sub-região de contexto q[1][i-3], q[1][i-2], e q[1][i-1] for menor que (ou igual a) um valor predeterminado de, por exemplo, 5. Caso seja descoberto que a soma dos ditos valores da sub-região de contexto for menor que o dito valor predeterminado, um valor hexadecimal de, por exemplo, 0x10000, é adicionado à variável c. Assim, a variável c é definida de modo que a variável c indica se há uma condição na qual os valores da sub-região de contexto q[1][i-3], q[1][i-2], e q[1][i-1] compreendem um valor de soma particularmente pequeno. Por exemplo, o bit 16 do valor de contexto corrente numérico pode agir como um indicador para indicar tal condição.

235. Para concluir, o valor de retorno da função “arith_get_context(c,i,N)” é determinado pelas etapas 504a, 504b, 504c, 504d, e 504e, onde o valor de contexto corrente numérico é derivado do valor de contexto prévio numérico nas etapas 504a, 504b, 504c, e 504d, e em que um indicador indicando um ambiente de valores espectrais previamente decodificados tendo, em média, valores absolutos particularmente pequenos, é derivado na etapa 504e e adicionado à variável c. Assim, o valor da variável c

obtido nas etapas 504a, 504b, 504c, 504d é retornado, em uma etapa 504f, como um valor de retorno da função “arith_get_context(c,i,N)”, se uma condição avaliada na etapa 504e não for cumprida. Em contraste, o valor da variável c, que é derivado nas etapas 504a, 504b, 504c, e 504d, é aumentado pelo valor hexadecimal de 0x10000 e o resultado desta operação de aumento é retornado, na etapa 504e, se a condição avaliada na etapa 540e for cumprida.

236. Para resumir o mencionado acima, deve ser observado que o decodificador silencioso emite 2 tuplos de coeficientes espectrais quantizados não assinados (conforme será descrito em mais detalhes abaixo). Primeiramente, o estado c do contexto é calculado com base nos coeficientes espectrais previamente decodificados “ao redor” do tuplo duplo para decodificar. Em uma realização preferida, o estado (que é, por exemplo, representado por um valor de contexto numérico) é adicionalmente atualizado utilizando o estado de contexto do último tuplo duplo decodificado (que é designado como um valor de contexto prévio numérico), considerando somente dois novos tuplos duplos (por exemplo, 2 tuplos 430 e 460). O estado é codificado nos 17 bits (por exemplo, utilizando uma representação numérica de um valor de contexto corrente numérico) e é retornado pela função “arith_get_context()”. Para detalhes, a referência é feita à representação do código do programa da figura 5c.

237. Além disso, deve ser observado que um código do pseudo-programa de uma realização alternativa da função “arith_get_context()” é mostrado na figura 5d. A função “arith_get_context(c,i)” de acordo com a figura 5d é semelhante à

função “arith_get_context(c,i,N)” de acordo com a figura 5c. Entretanto, a função “arith_get_context(c,i)” de acordo com a figura 5d não compreende um manuseio especial ou decodificação de tuplos de valores espectrais que compreendem um índice mínimo de frequência de i=0 ou um índice máximo de frequência de i=N/4-1.

11.5 Seleção da regra de mapeamento

238. A seguir, a seleção de uma regra de mapeamento, por exemplo, a tabela de frequências cumulativas que descreve um mapeamento de um valor de senha em um código de símbolo, será descrita. A seleção da regra de mapeamento é feita dependendo de um estado de contexto, que é descrito pelo valor de contexto corrente numérico c.

11.5.1 Seleção da regra de mapeamento utilizando o algoritmo de acordo com a figura 5e

239. A seguir, a seleção de uma regra de mapeamento utilizando a função “arith_get_pk(c)” será descrita. Deve ser observado que a função “arith_get_pk()” é chamada no início do sub-algoritmo 312ba ao decodificar um valor de código “acod_m” para prover um tuplo de valores espectrais. Deve ser observado que a função “arith_get_pk(c)” é chamada com diferentes argumentos em diferentes iterações do algoritmo 312b. Por exemplo, em uma primeira iteração do algoritmo 312b, a função “arith_get_pk(c)” é chamada com um argumento que é igual ao valor de contexto corrente numérico c, provido pela prévia execução da função “arith_get_context(c,i,N)” na etapa 312a. Em contraste, em outras iterações do sub-algoritmo 312ba, a função “arith_get_pk(c)” é chamada com um argumento que é uma soma do valor de contexto corrente numérico c provida pela função “arith_get_context(c,i,N)”

na etapa 312a, e uma versão de bit mudado do valor da variável “esc_nb”, em que o valor da variável “esc_nb” é mudado à esquerda em 17 bits. Assim, o valor de contexto corrente numérico c provido pela função “arith_get_context(c,i,N)” é utilizado como um valor de entrada da função “arith_get_pk()” na primeira iteração do algoritmo 312ba, ou seja, na decodificação de valores espectrais comparavelmente pequenos. Em contraste, ao decodificar valores espectrais comparavelmente maiores, a variável de entrada da função “arith_get_pk()” é modificada onde o valor da variável “esc_nb”, é levado em consideração, conforme é mostrado na figura 3.

240. Agora com referência à figura 5e, que mostra uma representação do código do pseudo-programa de uma primeira realização da função “arith_get_pk(c)”, deve ser observado que a função “arith_get_pk()” recebe a variável c como um valor de entrada, em que a variável c descreve o estado do contexto, e em que a variável de entrada c da função “arith_get_pk()” é igual ao valor de contexto corrente numérico provido como uma variável de retorno da função “arith_get_context()” pelo menos em algumas situações. Além disso, deve ser observado que a função “arith_get_pk()” provê, como uma variável de saída, a variável “pki”, que descreve um índice de um modelo de probabilidade e que pode ser considerado como um valor de índice de regra de mapeamento.

241. Tendo como referência a figura 5e, pode ser visto que a função “arith_get_pk()” compreende uma inicialização da variável 506a, em que a variável “i_min” é inicializada para ter o valor de -1. De forma semelhante, a variável i é definida para ser

igual à variável “i_min”, de modo que a variável i é também inicializada a um valor de -1. A variável “i_max” é inicializada para assumir um valor que é menor, em 1, que o número de entradas da tabela “ari_lookup_m[]” (detalhes que serão descritos tendo como referência as figuras 21(1) e 21(2)). Assim, as variáveis “i_min” e “i_max” definem um intervalo.

242. Subsequentemente, uma pesquisa 506b é realizada para identificar um valor de índice que designa uma entrada da tabela “ari_hash_m”, de modo que o valor da variável de entrada c da função “arith_get_pk()” dentro de um intervalo definido pela dita entrada e uma entrada adjacente.

243. Na pesquisa 506b, um sub-algoritmo 506ba é repetido, enquanto uma diferença entre as variáveis “i_max” e “i_min” for maior que 1. No sub-algoritmo 506ba, a variável i é definida para ser igual a um meio aritmético dos valores das variáveis “i_min” e “i_max”. Consequentemente, a variável i designa uma entrada da tabela “ari_hash_m[]” em um meio de um intervalo da tabela definido pelos valores das variáveis “i_min” e “i_max”. Subsequentemente, a variável j é definida para ser igual ao valor da entrada “ari_hash_m[i]” da tabela “ari_hash_m[]”. Assim, a variável j assume um valor definido por uma entrada da tabela “ari_hash_m[]”, cuja entrada permanece no meio de um intervalo da tabela definido pelas variáveis “i_min” e “i_max”. Subsequentemente, o intervalo definido pelas variáveis “i_min” e “i_max” é atualizado se o valor da variável de entrada c da função “arith_get_pk()” for diferente de um valor de estado definido pelos bits superiores da entrada de tabela “j=ari_hash_m[i]” da tabela “ari_hash_m[]”. Por exemplo, os “bits superiores” (bits 8 e

acima) das entradas da tabela “ari_hash_m[]”descrevem valores de estado significativos. Assim, o valor “j>>8” descreve um valor de estado significativo representado pela entrada “j=ari_hash_m[i]” da tabela “ari_hash_m[]” designada pela tabela *hash*-valor de índice i. Assim, se o valor da variável c for menor que o valor “j>>8”, isto significa que o valor de estado descrito pela variável c é menor que um valor de estado significativo descrito pela entrada “ari_hash_m[i]” da tabela “ari_hash_m[]”. Neste caso, o valor da variável “i_max” é definido para ser igual ao valor da variável i, que por sua vez tem o efeito que um tamanho do intervalo definido por “i_min” e “i_max” é reduzido, em que o novo intervalo é aproximadamente igual à metade inferior do intervalo prévio. Se for observado que a variável de entrada c da função “arith_get_pk()” é maior que o valor “j>>8”, que significa que o valor de contexto descrito pela variável c é maior que um valor de estado significativo descrito pela entrada “ari_hash_m[i]” da matriz “ari_hash_m[]”, o valor da variável “i_min” é definido para ser igual ao valor da variável i. Assim, o tamanho do intervalo definido pelos valores das variáveis “i_min” e “i_max” é reduzido aproximadamente metade do tamanho do intervalo prévio, definido pelos valores prévios das variáveis “i_min” e “i_max”. Para ser mais preciso, o intervalo definido pelo valor atualizado da variável “i_min” e pelo valor prévio da variável (não mudado) “i_max” é aproximadamente igual à metade superior do intervalo prévio no caso em que o valor da variável c é maior que o valor de estado significativo definido pela entrada “ari_hash_m[i]”.

244. Se, entretanto, for observado que o valor de contexto descrito pela variável de entrada c do algoritmo

“arith_get_pk()” é igual ao valor de estado significativo definido pela entrada “ari_hash_m[i]” (ou seja, c==(j>>8)), um valor de índice de regra de mapeamento definido pelos 8 bits mais inferiores da entrada “ari_hash_m[i]” é retornado como o valor de retorno da função “arith_get_pk()” (instrução “return (j&0xFF)”).

245. Para resumir o mencionado acima, uma entrada “ari_hash_m[i]”, os bits superiores (bits 8 e acima) dos quais descrevem um valor de estado significativo, é avaliada em cada iteração 506ba, e o valor de contexto (ou valor de contexto corrente numérico) descrito pela variável de entrada c da função “arith_get_pk()” é comparado com o valor de estado significativo descrito pela dita entrada de tabela “ari_hash_m[i]”. Se o valor de contexto representado pela variável de entrada c for menor que o valor de estado significativo representado pela entrada de tabela “ari_hash_m[i]”, o limite superior (descrito pelo valor “i_max”) do intervalo da tabela é reduzido, e se o valor de contexto descrito pela variável de entrada c é maior que o valor de estado significativo descrito pela entrada de tabela “ari_hash_m[i]”, o limite inferior (que é descrito pelo valor da variável “i_min”) do intervalo da tabela é aumentado. Em ambos os ditos casos, o sub-algoritmo 506ba é repetido, a menos que o tamanho do intervalo (definido pela diferença entre “i_max” e “i_min”) seja menor que, ou igual a, 1. Se, em contraste, o valor de contexto descrito pela variável c for igual ao valor de estado significativo descrito pela entrada de tabela “ari_hash_m[i]”, a função “arith_get_pk()” é abortada, em que o valor de retorno é definido pelos 8 bits mais inferiores da entrada de tabela “ari_hash_m[i]”.

246. Se, entretanto, a pesquisa 506b for finalizada por causa do tamanho do intervalo atingir seu valor mínimo (“i_max” - “i_min” é menor que, ou igual a, 1), o valor de retorno da função “arith_get_pk()” é determinado por uma entrada “ari_lookup_m[i_max]” da tabela “ari_lookup_m[]”, que pode ser vista no número de referência 506c. Assim, as entradas da tabela “ari_hash_m[]” definem tanto os valores de estado significativos como os limites de intervalos. No sub-algoritmo 506ba, os limites de intervalo da pesquisa “i_min” e “i_max” são adaptado de forma iterativa, de modo que a entrada “ari_hash_m[i]” da tabela “ari_hash_m[]”, um índice da tabela *hash* i do qual permanece, pelo menos aproximadamente, no centro do intervalo da pesquisa definido pelos valores de limite de intervalo “i_min” e “i_max”, pelo menos aproxima um valor de contexto descrito pela variável de entrada c. É então atingido que o valor de contexto descrito pela variável de entrada c dentro de um intervalo definido por “ari_hash_m[i_min]” e “ari_hash_m[i_max]” depois da conclusão das iterações do sub-algoritmo 506ba, a menos que o valor de contexto descrito pela variável de entrada c seja igual a um valor de estado significativo descrito por uma entrada da tabela “ari_hash_m[]”.

247. Se, entretanto, a repetição iterativa do sub-algoritmo 506ba for terminada por causa do tamanho do intervalo (definido por “i_max” - i_min”) ela atinge ou excede seu valor mínimo, é assumido que o valor de contexto descrito pela variável de entrada c não é um valor de estado significativo. Neste caso, o índice “i_max”, que designa um limite superior do intervalo, é, entretanto utilizado. O valor superior “i_max” do intervalo, que é atingido na última iteração do sub-algoritmo 506ba, é reutilizado

como um valor de índice da tabela para um acesso à tabela “ari_lookup_m”. A tabela “ari_lookup_m[]” descreve valores de índice de regra de mapeamento associado aos intervalos de diversos valores de contexto numéricos adjacentes. Os intervalos, nos quais os valores de índice de regra de mapeamento descritos pelas entradas da tabela “ari_lookup_m[]” estão associados, são definidos pelos valores de estado significativos descritos pelas entradas da tabela “ari_hash_m[]”. As entradas da tabela “ari_hash_m” definem ambos os valores de estado significativos e os limites de intervalos de valores de contexto numéricos adjacentes. Na execução do algoritmo 506b, é determinado se o valor de contexto numérico descrito pela variável de entrada c é igual a um valor de estado significativo, e se este não for o caso, cujo intervalo de valores de contexto numéricos (dos diversos intervalos, limites dos quais são definidos pelos valores de estado significativo) o valor de contexto descrito pela variável de entrada c permanece. Assim, o algoritmo 506b cumpre uma funcionalidade dupla para determinar se a variável de entrada c descreve um valor de estado significativo e, se não for o caso, para identificar um intervalo, delimitado por valores de estado significativos, no qual o valor de contexto representado pela variável de entrada c permanece. Assim, o algoritmo 506e é particularmente eficiente e requer somente um número comparavelmente pequeno de acessos da tabela.

248. Para resumir o mencionado acima, o estado de contexto c determina a tabela de frequências cumulativas utilizada para decodificar o plano de bit a bit mais significativo m. O mapeamento de c ao índice da tabela de frequências cumulativas

“pki” correspondente conforme realizado pela função “arith_get_pk()”. Uma representação do código do pseudo-programa da dita função “arith_get_pk()” foi explicado tendo como referência a figura 5e.

249. Para resumir o mencionado acima, o valor m é decodificado utilizando a função “arith_decode()” (que é descrita em mais detalhes abaixo) chamada com a tabela de frequências cumulativas “arith_cf_m[pki][]”, onde “pki” corresponde ao índice (também designado como valor de índice de regra de mapeamento) retornado pela função “arith_get_pk()”, que é descrita com referência à figura 5e.

11.5.2 Seleção da regra de mapeamento utilizando o algoritmo de acordo com a figura 5f

250. A seguir, outra realização de uma seleção do algoritmo da regra de mapeamento “arith_get_pk()” será descrito com referência à figura 5f que mostra uma representação do código do pseudo-programa de tal algoritmo, que pode ser utilizado na decodificação de um tuplo de valores espectrais. O algoritmo de acordo com a figura 5f pode ser considerado como uma versão otimizada (por exemplo, versão otimizada de velocidade) do algoritmo, “get_pk()” ou do algoritmo “arith_get_pk()”.

251. O algoritmo “arith_get_pk()” de acordo com a figura 5f recebe, como uma variável de entrada, uma variável c que descreve o estado do contexto. A variável de entrada c pode, por exemplo, representar um valor de contexto corrente numérico.

252. O algoritmo “arith_get_pk()” provê, como uma variável de saída, uma variável “pki”, que descreve o índice de uma distribuição de probabilidade (ou modelo de probabilidade)

associado a um estado do contexto descrito pela variável de entrada c. A variável “pki” pode, por exemplo, ser um valor de índice de regra de mapeamento.

253. O algoritmo de acordo com a figura 5f compreende uma definição dos conteúdos da matriz “i_diff[]”. Como pode ser visto, uma primeira entrada da matriz “i_diff[]” (tendo um índice da matriz 0) é igual a 299 e outras entradas da matriz (tendo índices da matriz 1 a 8) tomam os valores de 149, 74, 37, 18, 9, 4, 2, e 1. Assim, o tamanho da etapa para a seleção de um valor do índice da tabela *hash* “i_min” é reduzido com cada iteração, como as entradas das matrizes “i_diff[]” definem os ditos tamanhos das etapas. Para detalhes, a referência é feita à discussão abaixo.

254. Entretanto, diferentes tamanhos das etapas, por exemplo, diferentes conteúdos da matriz “i_diff[]” podem geralmente ser escolhidos, em que os conteúdos da matriz “i_diff[]” podem naturalmente ser adaptados ao tamanho da tabela *hash* “ari_hash_m[i]”.

255. Deve ser observado que a variável “i_min” é inicializada para assumir um valor de 0 correto no início do algoritmo “arith_get_pk()”.

256. Em uma etapa de inicialização 508a, uma variável s é inicializada dependendo da variável de entrada c, em que uma representação numérica da variável c é mudada à esquerda em 8 bits a fim de obter a representação numérica da variável s.

257. Subsequentemente, uma pesquisa de tabela 508b é realizada, a fim de identificar um valor de índice da tabela *hash* “i_min” de uma entrada da tabela *hash* “ari_hash_m[]”, de modo que o valor de contexto descrito pelo valor de contexto c dentro de um

intervalo que é delimitado pelo valor de contexto descrito pela entrada da tabela *hash* “ari_hash_m[i_min]” e um valor de contexto descrito por outra entrada da tabela *hash* “ari_hash_m” cuja outra entrada “ari_hash_m” seja adjacente (em termos de seu valor do índice da tabela *hash*) à entrada da tabela *hash* “ari_hash_m[i_min]”. Assim, o algoritmo 508b permite determinar um valor de índice da tabela *hash* “i_min” designando uma entrada “j=ari_hash_m[i_min]” da tabela *hash* “ari_hash_m[]”, de modo que a entrada da tabela *hash* “ari_hash_m[i_min]” pelo menos aproxime o valor de contexto descrito pela variável de entrada c.

258. A pesquisa de tabela 508b compreende uma execução iterativa de um sub-algoritmo 508ba, em que o sub-algoritmo 508ba é executado para um número predeterminado de, por exemplo, nove iterações. Na primeira etapa do sub-algoritmo 508ba, a variável i é definida a um valor que é igual a uma soma de um valor de uma variável “i_min” e um valor de uma entrada de tabela “i_diff[k]”. Deve ser observado aqui que k é uma variável de execução, que é aumentada, começando de um valor inicial de k=0, com cada iteração do sub-algoritmo 508ba. A matriz “i_diff[]” define predeterminar os valores de aumento, em que os valores de aumento diminuem com o aumento do índice da tabela k, ou seja, com aumento dos números de iterações.

259. Em uma segunda etapa do sub-algoritmo 508ba, um valor de uma entrada de tabela “ari_hash_m[]” é copiado em uma variável j. Preferivelmente, os bits superiores das entradas da tabela “ari_hash_m[]” descrevem valores de estado significativos de um valor de contexto numérico, e os bits mais inferiores (bits 0 a 7) das entradas da tabela “ari_hash_m[]”descrevem valores de

índice de regra de mapeamento associado aos respectivos valores de estado significativos.

260. Em uma terceira etapa do sub-algoritmo 508ba, o valor da variável S é comparado com o valor da variável j, e a variável “i_min” é seletivamente definida ao valor “i+1” se o valor da variável s for maior que o valor da variável j. Subsequentemente, a primeira etapa, a segunda etapa, e a terceira etapa do sub-algoritmo 508ba são repetidas para um número predeterminado de vezes, por exemplo, nove vezes. Assim, em cada execução do sub-algoritmo 508ba, o valor da variável “i_min” é aumentado por i_diff[]+1, se, e somente se, o valor de contexto descrito pelo índice da tabela *hash* atualmente válido i_min + i_diff[] for menor que o valor de contexto descrito pela variável de entrada c. Assim, o valor de índice da tabela *hash* “i_min” é (de forma iterativa) aumentado em cada execução do sub-algoritmo 508ba se (e somente se) o valor de contexto descrito pela variável de entrada c e, consequentemente, pela variável s, for maior que o valor de contexto descrito pela entrada “ari_hash_m[i=i_min + diff[k]]”.

261. Além disso, deve ser observado que somente uma única comparação, a saber, a comparação como se o valor da variável s fosse maior que o valor da variável j, é realizada em cada execução do sub-algoritmo 508ba. Assim, o algoritmo 508ba é computacionalmente particularmente eficiente. Além disso, deve ser observado que há diferentes possíveis resultados com relação ao valor final da variável “i_min”. Por exemplo, é possível que o valor da variável “i_min” depois da última execução do sub-algoritmo 512ba seja de modo que o valor de contexto descrito pela

entrada de tabela “ari_hash_m[i_min]” seja menor que o valor de contexto descrito pela variável de entrada c, e que o valor de contexto descrito pela entrada de tabela “ari_hash_m[i_min +1]” seja maior que o valor de contexto descrito pela variável de entrada c. De modo alternativo, pode acontecer que depois da última execução do sub-algoritmo 508ba, o valor de contexto descrito pela entrada da tabela *hash* “ari_hash_m[i_min −1]” seja menor que o valor de contexto descrito pela variável de entrada c, e que o valor de contexto descrito pela entrada “ari_hash_m[i_min]” seja maior que o valor de contexto descrito pela variável de entrada c. De modo alternativo, entretanto, pode acontecer que o valor de contexto descrito pela entrada da tabela *hash* “ari_hash_m[i_min]” seja idêntico ao valor de contexto descrito pela variável de entrada c.

262. Por esta razão, uma provisão do valor de retorno com base na decisão 508c é realizada. A variável j é definida para ter o valor da entrada da tabela *hash* “ari_hash_m[i_min]”. Subsequentemente, é determinado se o valor de contexto descrito pela variável de entrada c (e também pela variável s) é maior que o valor de contexto descrito pela entrada “ari_hash_m[i_min]” (primeiro caso definido por uma condição “s>j”), ou se o valor de contexto descrito pela variável de entrada c for menor que o valor de contexto descrito pela entrada da tabela *hash* “ari_hash_m[i_min]” (segundo caso definido por uma condição “c<j>>8”), ou se o valor de contexto descrito pela variável de entrada c é igual ao valor de contexto descrito pela entrada “ari_hash_m[i_min]” (terceiro caso).

263. No primeiro caso, (s>j), uma entrada

“ari_lookup_m[i_min +1]” da tabela “ari_lookup_m[]” designada pelo valor de índice da tabela “i_min+1” é retornada como o valor de saída da função “arith_get_pk()”. No segundo caso (c<(j>>8)), uma entrada “ari_lookup_m[i_min]” da tabela “ari_lookup_m[]” designada pelo valor de índice da tabela “i_min” é retornado como o valor de retorno da função “arith_get_pk()”. No terceiro caso (ou seja, se o valor de contexto descrito pela variável de entrada c for igual ao valor de estado significativo descrito pela entrada de tabela “ari_hash_m[i_min]”), um valor de índice de regra de mapeamento descrito pelos 8 bits mais inferiores da entrada da tabela *hash* “ari_hash_m[i_min]” é retornado como o valor de retorno da função “arith_get_pk()”.

264. Para resumir o mencionado acima, uma pesquisa de tabela particularmente simples é realizada na etapa 508b, em que a pesquisa de tabela provê um valor da variável de uma variável “i_min” sem distinguir se o valor de contexto descrito pela variável de entrada c é igual a um valor de estado significativo definido por uma das entradas da tabela de estado “ari_hash_m[]” ou não. Na etapa 508c, que é realizada subsequente à pesquisa de tabela 508b, uma relação de magnitude entre o valor de contexto descrito pela variável de entrada c e um valor de estado significativo descrito pela entrada da tabela *hash* “ari_hash_m[i_min]” é avaliada, e o valor de retorno da função “arith_get_pk()” é selecionado dependendo de um resultado da dita avaliação, em que o valor da variável “i_min”, que é determinado na avaliação da tabela 508b, é considerado para selecionar um valor de índice de regra de mapeamento mesmo se o valor de contexto descrito pela variável de entrada c for diferente do

valor de estado significativo descrito pela entrada da tabela *hash* “ari_hash_m[i_min]” .

265. Ainda deve ser observado que a comparação no algoritmo deve preferivelmente (ou de modo alternativo) ser feita entre o índice de contexto (valor de contexto numérico) c e j=ari_hash_m[i]>>8. Ainda, cada entrada da tabela “ari_hash_m[]” representa um índice de contexto, codificado além do 8° bit, e seu modelo de probabilidade correspondente codificado nos primeiros 8 bits (bits menos significativos). Na implementação corrente, estamos interessados principalmente no conhecimento se o presente contexto c é maior que ari_hash_m[i]>>8, que é equivalente à detecção se s=c<<8 for também maior que ari_hash_m[i].

266. Para resumir o mencionado acima, visto que o estado de contexto é calculado (que pode, por exemplo, ser obtido utilizando o algoritmo “arith_get_context(c,i,N)” de acordo com a figura 5c, ou o algoritmo “arith_get_context(c,i)” de acordo com a figura 5d, o plano de bit a bit mais significativo é decodificado utilizando o algoritmo “arith_decode” (que será descrito abaixo) chamado com a tabela de frequências cumulativas apropriada correspondente ao modelo de probabilidade correspondente ao estado de contexto. A correspondência é feita pela função “arith_get_pk()”, por exemplo, a função “arith_get_pk()” que foi discutida com referência à figura 5f.

11.6 Decodificação aritmética

11.6.1 Decodificação aritmética utilizando o algoritmo de acordo com a figura 5g

267. A seguir, a funcionalidade da função “arith_decode()” será discutida em detalhes com referência à

figura 5g.

268. Deve ser observado que a função “arith_decode()” utiliza a função de ajuda “arith_first_symbol (void)”, que retorna VERDADEIRA, se for o primeiro símbolo da sequência e FALSA caso contrário. A função “arith_decode()” também utiliza a função de ajuda “arith_get_next_bit(void)”, que obtém e provê o próximo bit do fluxo de bits.

269. Além disso, a função “arith_decode()” utiliza as variáveis globais “low” [baixa], “high” [alta] e “value” [valor]. Ainda, a função “arith_decode()” recebe, como uma variável de entrada, a variável “cum_freq[]”, que aponta em direção a uma primeira entrada ou elemento (tendo o índice de elemento ou índice de entrada 0) da tabela de frequências cumulativas selecionada ou sub-tabela de frequências cumulativas. Ainda, a função “arith_decode()” utiliza a variável de entrada “cfl”, que indica o comprimento da tabela de frequências cumulativas selecionada ou sub-tabela de frequências cumulativas designada pela variável “cum_freq[]”.

270. A função “arith_decode()” compreende, como uma primeira etapa, uma inicialização da variável 570a, que é realizada se a função de ajuda “arith_first_symbol()” indicar que o primeiro símbolo de uma sequência de símbolos está sendo decodificado. A inicialização do valor 550a inicializa a variável “value” dependendo de diversos, por exemplo, 16 bits, que são obtidos do fluxo de bits utilizando a função de ajuda “arith_get_next_bit”, de modo que a variável “value” tenha o valor representado pelos ditos bits. Ainda, a variável “low” é inicializada para ter o valor de 0, e a variável “high” é

inicializada para ter o valor de 65535.

271. Em uma segunda etapa 570b, a variável “range” [faixa] é definida a um valor, que é maior, em 1, que a diferença entre os valores das variáveis “high” e “low”. A variável “cum” é definida a um valor que representa uma posição relativa do valor da variável “value” entre o valor da variável “low” e o valor da variável “high”. Assim, a variável “cum” tem, por exemplo, um valor entre 0 e 2^{16} dependendo do valor da variável “value”.

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

273. O algoritmo “arith_decode()” também compreende uma pesquisa iterativa da tabela de frequências cumulativas 570c. A pesquisa iterativa da tabela de frequências cumulativas é repetida até que a variável cfl seja menor que ou igual a 1. Na pesquisa iterativa da tabela de frequências cumulativas 570c, a variável do apontador q é definida a um valor, que é igual a uma soma do valor corrente da variável do apontador p e metade do valor da variável “cfl”. Se o valor da entrada *q da tabela de frequências cumulativas selecionada, cuja entrada é direcionada pela variável do apontador q, for maior que o valor da variável “cum”, a variável do apontador p é definida ao valor da variável do apontador q, e a variável “cfl” é aumentada. Finalmente, a variável “cfl” é mudada à direita por um bit, assim divide efetivamente o valor da variável “cfl” por 2 e ignora a parte do módulo.

274. Assim, a pesquisa iterativa da tabela de frequências cumulativas 570c efetivamente compara o valor da

variável “cum” com diversas entradas da tabela de frequências cumulativas selecionada, a fim de identificar um intervalo dentro da tabela de frequências cumulativas selecionada, que é delimitado pelas entradas da tabela de frequências cumulativas, obtendo a soma do valor dentro do intervalo identificado. Assim, as entradas da tabela de frequências cumulativas selecionada definem os intervalos, em que um respectivo valor do símbolo está associado a cada um dos intervalos da tabela de frequências cumulativas selecionada. Ainda, as larguras dos intervalos entre dois valores adjacentes da tabela de frequências cumulativas definem as probabilidades dos símbolos associados aos ditos intervalos, de modo que a tabela de frequências cumulativas selecionada em sua totalidade defina uma distribuição de probabilidade de diferentes símbolos (ou valores de símbolos). Detalhes referentes às tabelas de frequências cumulativas disponíveis serão discutidos abaixo tendo como referência a figura 23.

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

276. O algoritmo “arith_decode” também compreende uma adaptação 570e das variáveis “high” e “low”. Se o valor do símbolo representado pela variável “symbol” for diferente de 0, a variável “high” é atualizada, conforme mostrado no número de referência 570e. Ainda, o valor da variável “low” é atualizado, conforme

mostrado no número de referência 570e. A variável “high” é definida a um valor que é determinado pelo valor da variável “low”, a variável “range” e a entrada tendo o índice “symbol −1” da tabela de frequências cumulativas selecionada. A variável “low” é aumentada, em que a magnitude do aumento é determinada pela variável “range” e a entrada da tabela de frequências cumulativas selecionada tendo o índice “symbol”. Assim, a diferença entre os valores das variáveis “low” e “high” é ajustada dependendo da diferença numérica entre as duas entradas adjacentes da tabela de frequências cumulativas selecionada.

277. Assim, se um valor do símbolo tendo uma baixa probabilidade for detectado, o intervalo entre os valores das variáveis “low” e “high” é reduzido à largura estreita. Em contraste, se o valor do símbolo detectado compreende uma probabilidade relativamente grande, a largura do intervalo entre os valores das variáveis “low” e “high” é ajustada ao valor comparavelmente grande. Novamente, a largura do intervalo entre os valores da variável “low” e “high” é dependente do símbolo detectado e as entradas da tabela de frequências cumulativas correspondentes.

278. O algoritmo “arith_decode()” também compreende uma renormalização do intervalo 570f, na qual o intervalo determinado na etapa 570e é de forma iterativa mudado e escalado até que a condição de “interrupção” seja atingida. Na renormalização do intervalo 570f, uma operação de mudança para baixo seletiva 570fa é realizada. Se a variável “high” for menor que 32768, nada é feito, e a renormalização do intervalo continua com uma operação de aumento do tamanho do intervalo 570fb. Se,

entretanto, a variável “high” não for menor que 32768 e a variável “low” for maior ou igual a 32768, as variáveis “values”, “low” e “high” são todas reduzidas a 32768, de modo que um intervalo definido pelas variáveis “low” e “high” seja mudado para baixo, e de modo que o valor da variável “value” também seja mudado para baixo. Se, entretanto, for observado que o valor da variável “high” não é menor que 32768, e que a variável “low” não é maior ou igual a 32768, e que a variável “low” é maior ou igual a 16384 e que a variável “high” é menor que 49152, as variáveis “value”, “low” e “high” são todas reduzidas a 16384, assim reduzindo o intervalo entre os valores das variáveis “high” e “low” e também o valor da variável “value”. Se, entretanto, nenhuma das condições acima for cumprida, a renormalização do intervalo é abortada.

279. Se, entretanto, qualquer uma das condições mencionadas acima, que são avaliadas na etapa 570fa, for cumprida, a operação de aumento do intervalo 570fb é executada. Na operação de aumento do intervalo 570fb, o valor da variável “low” é dobrado. Ainda, o valor da variável “high” é dobrado, e o resultado da operação de dobrar é aumentado em 1. Ainda, o valor da variável “value” é dobrado (mudada à esquerda por um bit), e um bit do fluxo de bits, que é obtido pela função de ajuda “arith_get_next_bit” é utilizado como o bit menos significativo. Assim, o tamanho do intervalo entre os valores das variáveis “low” e “high” é aproximadamente dobrado, e a precisão da variável “value” é aumentada utilizando um novo bit do fluxo de bits. Conforme mencionado acima, as etapas 570fa e 570fb são repetidas até que a condição de “interrupção” seja atingida, ou seja, até que o intervalo entre os valores das variáveis “low” e “high” seja

grande suficiente.

280. Com referência à funcionalidade do algoritmo “arith_decode()”, deve ser observado que o intervalo entre os valores das variáveis “low” e “high” é reduzido na etapa 570e dependendo de duas entradas adjacentes da tabela de frequências cumulativas referenciadas pela variável “cum_freq”. Se um intervalo entre dois valores adjacentes da tabela de frequências cumulativas selecionada for pequeno, ou seja, se os valores adjacentes são comparavelmente juntos, o intervalo entre os valores das variáveis “low” e “high”, que é obtido na etapa 570e, será comparavelmente pequeno. Em contraste, se duas entradas adjacentes da tabela de frequências cumulativas são mais espaçadas, o intervalo entre os valores das variáveis “low” e “high”, que é obtido na etapa 570e, será comparavelmente grande.

281. Consequentemente, se o intervalo entre os valores das variáveis “low” e “high”, que é obtido na etapa 570e, for comparavelmente pequeno, um grande número de renormalização das etapas de intervalo será executado para redimensionar o intervalo ao tamanho “suficiente” (de modo que nenhuma das condições de uma avaliação de condição 570fa seja cumprida). Assim, um número comparavelmente grande de bits do fluxo de bits será utilizado a fim de aumentar a precisão da variável “value”. Se, em contraste, o tamanho do intervalo obtido na etapa 570e for comparavelmente grande, somente um número menor de repetições das etapas de normalização do intervalo 570fa e 570fb será necessário a fim de renormalizar o intervalo entre os valores das variáveis “low” e “high” ao tamanho “suficiente”. Assim, somente um número comparavelmente pequeno de bits do fluxo de bits será utilizado

para aumentar a precisão da variável “value” e para preparar uma decodificação do próximo símbolo.

282. Para resumir o mencionado acima, se um símbolo é decodificado, que compreende uma probabilidade comparavelmente alta, e no qual um grande intervalo é associado pelas entradas da tabela de frequências cumulativas selecionada, somente um número comparavelmente pequeno de bits será lido do fluxo de bits a fim de permitir uma decodificação de um símbolo subsequente. Em contraste, se um símbolo é decodificado, que compreende uma probabilidade comparavelmente pequena e no qual um pequeno intervalo é associado pelas entradas da tabela de frequências cumulativas selecionada, um número comparavelmente grande de bits será considerado do fluxo de bits a fim de preparar uma decodificação do próximo símbolo.

283. Assim, as entradas das tabelas de frequências cumulativas refletem as probabilidades dos diferentes símbolos e também refletem um número de bits necessários para a decodificação de uma sequência de símbolos. Pela variação da tabela de frequências cumulativas dependendo de um contexto, ou seja, dependendo dos símbolos previamente decodificados (ou valores espectrais), por exemplo, selecionando diferentes tabelas de frequências cumulativas dependendo do contexto, as dependências estocásticas entre os diferentes símbolos podem ser exploradas, que permitem uma codificação eficiente da taxa de bits particular dos símbolos subsequentes (ou adjacentes).

284. 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 por uma função “arith_get_pk()” para determinar o valor do plano de bits mais significativo m (que pode ser definido ao valor do símbolo representado pela variável “symbol” de retorno).

285. Para resumir o mencionado acima, o decodificador aritmético é uma implementação de número inteiro utilizando o método da geração de identificação com escala. Para detalhes, a referência é feita ao livro “*Introduction to Data Compression*” de K. Sayood, Terceira Edição, 2006, Elsevier Inc.

286. O código do programa de computador de acordo com a figura 5g descreve o algoritmo utilizado de acordo com uma realização da invenção.

11.6.2 Decodificação aritmética utilizando o algoritmo de acordo com as figuras 5h e 5i

287. As figuras 5h e 5i mostram uma representação do código do pseudo-programa de outra realização do algoritmo “arith_decode()”, que pode ser utilizado como uma alternativa para o algoritmo “arith_decode” descrito com referência à figura 5g.

288. Deve ser observado que os algoritmos de acordo com a figura 5g e as figuras 5h e 5i podem ser utilizados no algoritmo “values_decode()” de acordo com a figura 3.

289. Para resumir, o valor m é decodificado utilizando a função “arith_decode()” chamada com a tabela de frequências cumulativas “arith_cf_m[pki][]” em que “pki” corresponde ao índice retornado pela função “arith_get_pk()”. O codificador (ou decodificador) aritmético é uma implementação de número inteiro utilizando o método de geração de identificação com escala. Para detalhes, a referência é feita ao livro “*Introduction to Data*

Compression” de K. Sayood, Terceira Edição, 2006, Elsevier Inc. O código do programa de computador de acordo com a figura 5h e 5i descreve o algoritmo utilizado.

11.7 Mecanismo de escape

290. A seguir, o mecanismo de escape, que é utilizado na decodificação do algoritmo “values_decode()” de acordo com a figura 3, será brevemente discutido.

291. Quando o valor decodificado m (que é provido como um valor de retorno da função “arith_decode()”) for o símbolo de escape “ARITH_ESCAPE”, as variáveis “lev” e “esc_nb” são aumentadas em 1, e o outro valor m é decodificado. Neste caso, a função “arith_get_pk()” é chamada mais uma vez com o valor “c+ esc_nb<<17” como argumento de entrada, onde a variável “esc_nb” descreve o número de símbolos de escape previamente decodificados para o mesmo tuplo duplo e delimitado a 7.

292. Para resumir, se um símbolo de escape for identificado, assume-se que o valor do plano de bits mais significativo m compreende um peso numérico aumentado. Além disso, a decodificação numérica corrente é repetida, em que um valor de contexto corrente numérico modificado “c+ esc_nb<<17” é utilizado como uma variável de entrada à função “arith_get_pk()”. Assim, um valor de índice de regra de mapeamento diferente “pki” é tipicamente obtido nas diferentes iterações do sub-algoritmo 312ba.

11.8 Mecanismo de parada aritmética

293. A seguir, o mecanismo de parada aritmética será descrito. O mecanismo de parada aritmética permite a redução do número de bits necessários no caso em que a parte da frequência

superior é completamente quantizada a 0 em um codificador de áudio.

294. Em uma realização, um mecanismo de parada aritmética pode ser implementado como segue: visto que o valor m não é o símbolo de escape, “ARITH_ESCAPE”, o decodificador verifica se o m sucessivo forma um símbolo “ARITH_ESCAPE”. Se uma condição “esc_nb >0&&m==0” for verdadeira, o símbolo “ARITH_STOP” é detectado e o processo de decodificação é finalizado. Neste caso, o decodificador vai diretamente para a função “arith_finish()” que será descrita abaixo. A condição significa que o restante da estrutura é composto por valores 0.

11.9 Decodificação do plano de bits menos significativo

295. A seguir, a decodificação de um ou mais planos de bits menos significativos será descrita. A decodificação do plano de bits menos significativo, é realizada, por exemplo, na etapa 312d mostrada na figura 3. De modo alternativo, entretanto, os algoritmos conforme mostrados nas figuras 5j e 5n podem ser utilizados.

11.9.1 Decodificação do plano de bits menos significativo de acordo com a figura 5j

296. Agora com referência à figura 5j, pode ser visto que os valores das variáveis a e b são derivados do valor m. Por exemplo, a representação numérica do valor m é deslocada à direita em 2 bits para obter a representação numérica da variável b. Além disso, o valor da variável a é obtido subtraindo uma versão deslocada do bit do valor da variável b, deslocado por bit à esquerda em 2 bits, do valor da variável m.

297. Subsequentemente, uma decodificação aritmética dos valores do plano de bits menos significativo r é repetida, em que o número de repetições é determinado pelo valor da variável “lev”. Um valor do plano de bits menos significativo r é obtido utilizando a função “arith_decode”, em que a tabela de frequências cumulativas adaptada à decodificação do plano de bit menos significativo é utilizada (tabela de frequências cumulativas “arith_cf_r”). Um bit menos significativo (tendo um peso numérico de 1) da variável r descreve um plano de bits menos significativo do valor espectral representado pela variável a, e um bit tendo um peso numérico de 2 da variável r descreve um bit menos significativo do valor espectral representado pela variável b. Assim, a variável a é atualizada mudando a variável a para a esquerda em 1 bit e adicionando o bit tendo o peso numérico de 1 da variável r como o bit menos significativo. De forma semelhante, a variável b é atualizada mudando a variável b à esquerda por um bit e adicionando o bit tendo o peso numérico de 2 da variável r.

298. Assim, os dois bits que carregam informação mais significativa das variáveis a,b são determinados pelo valor do plano de bits mais significativo m, e um ou mais bits menos significativos (se houver) dos valores a e b são determinados por um ou mais valores do plano de bits menos significativo r.

299. Para resumir o mencionado acima, se o símbolo “ARITH_STOP” não for encontrado, os planos de bits restantes são então decodificados, se houver algum, para o tuplo duplo presente. Os planos de bits restantes são decodificados do nível mais significativo ao menos significativo chamando a função “arith_decode()” lev várias vezes com a tabela de frequências

cumulativas “arith_cf_r[]”. Os planos de bits decodificados r permitem a refinação do valor previamente decodificado m de acordo com o algoritmo, um código do pseudo-programa que é mostrado na figura 5j.

11.9.2 Decodificação da faixa de bit menos significativo de acordo com a figura 5n

300. De modo alternativo, entretanto, o algoritmo de uma representação do código do pseudo-programa que é mostrada na figura 5n pode também ser utilizado para a decodificação do plano de bits menos significativo. Neste caso, se o símbolo “ARITH_STOP” não for encontrado, os planos de bits restantes são então decodificados, se houver algum, para o presente tuplo duplo. Os planos de bits restantes são decodificados do nível mais significativo ao menos significativo chamando “lev” vezes “arith_decode()” com a tabela de frequências cumulativas “arith_cf_r()”. Os planos de bits decodificados r permitem a refinação do valor previamente decodificado m de acordo com o algoritmo mostrado na figura 5n.

11.10 Atualização de contexto

11.10.1 Atualização de contexto de acordo com a figura 5k, 5l, e 5m

301. A seguir, as operações utilizadas para concluir a decodificação do tuplo de valores espectrais serão descritas, tendo como referência as figuras 5k e 5l. Além disso, uma operação será descrita que é utilizada para concluir uma decodificação de um conjunto de tuplos de valores espectrais associados a uma parte corrente (por exemplo, uma estrutura corrente) de um conteúdo de áudio.

302. Agora com referência à figura 5k, pode ser visto que a entrada tendo o índice de entrada 2*i da matriz “x_ac_dec[]” é definida para ser igual a a, e que a entrada tendo o índice de entrada “2*i+1” da matriz “x_ac_dec[]” é definida para ser igual a b depois da decodificação de bit menos significativo 312d. Em outras palavras, no ponto depois da decodificação de bit menos significativo 312d, o valor não sinalizado do tuplo duplo (a,b), é completamente decodificado. É salvo no elemento (por exemplo, a matriz “x_ac_dec[]”) que mantém os coeficientes espectrais de acordo com o algoritmo mostrado na figura 5k.

303. Subsequentemente, o contexto “q” também é atualizado para o próximo tuplo duplo. Deve ser observado que esta atualização de contexto também deve ser realizada para o último tuplo duplo. Esta atualização de contexto é realizada pela função “arith_update_context()”, uma representação do código do pseudo-programa do qual é mostrado na figura 5l.

304. Agora com referência à figura 5l, pode ser visto que a função “arith_update_context(i,a,b)” recebe, como variáveis de entrada, coeficientes espectrais quantizados decodificados não sinalizados (ou valores espectrais) a, b do tuplo duplo. Além disso, a função “arith_update_context” também recebe, como uma variável de entrada, um índice i (por exemplo, um índice de frequência) do coeficiente espectral quantizado para decodificar. Em outras palavras, a variável de entrada i pode, por exemplo, ser um índice do tuplo de valores espectrais, valores absolutos que são definidos pelas variáveis de entrada a, b. Como pode ser visto, a entrada “q[1][i]” da matriz “q[][]” pode ser definida a um valor que é igual a a+b+1. Além disso, o valor da entrada

“q[1][i]” da matriz “q[][]” pode ser limitado ao valor hexadecimal de “0xF”. Assim, a entrada “q[1][i]” da matriz “q[][]” é obtida calculando uma soma de valores absolutos do tuplo atualmente decodificados {a,b} de valores espectrais tendo o índice de frequência i, e adicionando 1 ao resultado da dita soma.

305. Deve ser observado aqui que a entrada “q[1][i]” da matriz “q[][]” pode ser considerada como um valor da sub-região de contexto, pois descreve uma sub-região do contexto que é utilizado para uma decodificação subsequente de valores espectrais adicionais (ou tuplos de valores espectrais).

306. Deve ser observado aqui que a soma dos valores absolutos a e b dos dois valores espectrais atualmente decodificados (versões assinadas que são armazenadas nas entradas “x_ac_dec[2*i]” e “x_ac_dec[2*i+1]” da matriz “x_ac_dec[]”), pode ser considerada como o cálculo de uma norma (por exemplo, uma norma L1) dos valores espectrais decodificados.

307. Foi observado que os valores da sub-região de contexto (ou seja, entradas da matriz “q[][]”), que descrevem uma norma de um vetor formado por diversos valores espectrais previamente decodificados são particularmente significativos e com memória eficiente. Foi observado que tal norma, que é calculada com base em diversos valores espectrais previamente decodificados, compreende informação significativa de contexto em uma forma compacta. Foi observado que o sinal dos valores espectrais não é tipicamente particularmente relevante para a escolha do contexto. Também foi observado que a formação de uma norma através dos diversos valores espectrais previamente decodificados tipicamente mantém a informação mais importante, embora alguns detalhes sejam

descartados. Além disso, foi observado que uma limitação do valor de contexto corrente numérico ao valor máximo tipicamente não resulta em uma perda grave de informação. Preferivelmente, foi observado que é mais eficiente utilizar o mesmo estado de contexto para valores espectrais significativos que são maiores que um valor limite predeterminado. Assim, a limitação dos valores da sub-região de contexto traz consigo mais melhoria da eficiência da memória. Além disso, foi observado que a limitação dos valores da sub-região de contexto a certo valor máximo permite uma atualização eficiente computacional e particularmente simples do valor de contexto corrente numérico, que foi descrito, por exemplo, com referência às figuras 5c e 5d. Limitando os valores da sub-região de contexto ao valor comparavelmente pequeno (por exemplo, a um valor de 15), um estado de contexto que tem como base diversos valores da sub-região de contexto pode ser representado na forma eficiente, que foi discutida tendo como referência as figuras 5c e 5d.

308. Além disso, foi observado que uma limitação dos valores da sub-região de contexto dos valores entre 1 e 15, traz consigo um compromisso particularmente bom entre precisão e eficiência da memória, pois 4 bits são suficientes a fim de armazenar tal valor da sub-região de contexto.

309. Entretanto, deve ser observado que em outras realizações, um valor da sub-região de contexto pode ter como base um único valor decodificado espectral somente. Neste caso, a formação de uma norma pode opcionalmente ser omitida.

310. O próximo tuplo duplo da estrutura é decodificado depois da conclusão da função “arith_update_context” aumentando i

em 1 e refazendo o mesmo processo conforme descrito acima, começando pela função “arith_get_context()”.

311. Quando os 2 tuplos lg/2 são decodificados dentro da estrutura, ou com o símbolo de parada de acordo com “ARITH_ESCAPE” ocorre, o processo de decodificação da amplitude espectral termina e a decodificação dos sinais começa.

312. Detalhes referentes à decodificação dos sinais foram discutidos com referência à figura 3, em que a decodificação dos sinais é mostrada em número de referência 314.

313. Visto que todos os coeficientes espectrais quantizados não sinalizados são decodificados, o sinal de acordo é adicionado. Para cada valor quantizado não nulo de “x_ac_dec”, um bit é lido. Se o valor de bit de leitura for igual a 0, o valor quantizado é positivo, nada é feito e o valor sinalizado é igual ao valor não sinalizado previamente decodificado. Caso contrário (ou seja, se o valor de bit de leitura é igual a 1), o coeficiente decodificado (ou valor espectral) é negativo e o complemento de dois é considerado do valor não sinalizado. Os bits do sinal são lidos das frequências baixa a mais alta. Para detalhes, a referência é feita às figuras 3 e as explicações referentes à decodificação de sinais 314.

314. A decodificação é finalizada chamando a função “arith_finish()”. Os coeficientes espectrais restantes são definidos a 0. Os respectivos estados de contexto são atualizados correspondentemente.

315. Para detalhes, a referência é feita à figura 5m, que mostra uma representação do código do pseudo-programa da função “arith_finish()”. Como pode ser visto, a função

“arith_finish()” recebe uma variável de entrada lg que descreve os coeficientes espectrais quantizados decodificados. Preferivelmente, a variável de entrada lg da função “arith_finish” descreve um número de coeficientes espectrais atualmente decodificados, deixando os coeficientes espectrais desconsiderados, onde um valor 0 foi alocado em resposta à detecção de um símbolo “ARITH_STOP”. Uma variável de entrada N da função “arith_finish” descreve um comprimento da janela de uma janela corrente (ou seja, uma janela associada à parte corrente do conteúdo de áudio). Tipicamente, um número de valores espectrais associado a uma janela de comprimento N é igual a N/2 e um número de 2 tuplos de valores espectrais associado a uma janela de comprimento da janela N é igual a N/4.

316. A função “arith_finish” também recebe, como um valor de entrada, um vetor “x_ac_dec” de valores espectrais decodificados, ou pelo menos uma referência a tal vetor de coeficientes decodificados espectrais.

317. A função “arith_finish” é configurada para definir as entradas da matriz (ou vetor) “x_ac_dec”, para as quais nenhum valor espectral foi decodificado devido à presença de uma condição de parada aritmética, a 0. Além disso, a função “arith_finish” define valores da sub-região de contexto “q[1][i]”, que são associados a um valor espectral para qual nenhum valor foi decodificado devido à presença de uma condição de parada aritmética, ao valor predeterminado de 1. O valor predeterminado de 1 corresponde ao tuplo dos valores espectrais em que ambos os valores espectrais são iguais a 0.

318. Assim, a função “arith_finish()” permite

atualizar toda a matriz (ou vetor) “x_ac_dec[]” de valores espectrais e também toda a matriz de valores da sub-região de contexto “q[1][i]”, mesmo em uma presença de uma condição de parada aritmética.

11.10.2 Atualização de contexto de acordo com as figuras 5o e 5p

319. A seguir, outra realização da atualização de contexto será descrita tendo como referência as figuras 5o e 5p. No ponto em que o valor não sinalizado do tuplo duplo (a,b) for completamente decodificado, o contexto q é então atualizado para o próximo tuplo duplo. A atualização também é realizada se o tuplo duplo presente for o último tuplo duplo. Ambas as atualizações são feitas pela função “arith_update_context()”, a representação do código do pseudo-programa que é mostrada na figura 5o.

320. O próximo tuplo duplo da estrutura é então decodificado aumentando i em 1 e chamando a função arith_decode(). Se os 2 tuplos lg/2 já foram decodificados com a estrutura, ou se o símbolo de parada “ARITH_STOP” ocorreu, a função “arith_finish()” é chamada. O contexto é salvo e armazenado na matriz (ou vetor) “qs” para a próxima estrutura. Um código do pseudo-programa da função “arith_save_context()” é mostrado na figura 5p.

321. Visto que todos os coeficientes espectrais quantizados não sinalizados são decodificados, o sinal é então adicional. Para cada valor não quantizado de “qdec”, um bit é lido. Se o valor de bit de leitura for igual a 0, o valor quantizado é positivo, nada é feito e o valor sinalizado é igual ao valor não sinalizado previamente decodificado. Caso contrário,

o coeficiente decodificado é negativo e o complemento de dois é considerado do valor não sinalizado. Os bits sinalizados são lidos das frequências baixas às altas.

11.11 Sumário do processo de decodificação

322. A seguir, o processo de decodificação será brevemente resumido. Para detalhes, a referência é feita à discussão acima e também às figuras 3, 4, 5a, 5c, 5e, 5g, 5j, 5k, 5l, e 5m. Os coeficientes espectrais quantizados “x_ac_dec[]” são silenciosamente decodificados começando do coeficiente com frequência mais baixa e continuando ao coeficiente com frequência mais alta. Eles são decodificados por grupos de dois coeficientes sucessivos a,b reunindo um tuplo duplo (a,b).

323. Os coeficientes decodificados “x_ac_dec[]” para o domínio de frequência (ou seja, para um modo de domínio de frequência) são então armazenados na matriz “x_ac_quant[g][win][sfb][bin]”. Uma ordem de transmissão das senhas de codificação silenciosa é de modo que quando elas são decodificadas na ordem recebida e armazenada na matriz, “bin” é o índice que aumenta mais rapidamente e “g” é o índice que aumenta mais lentamente. Dentro de uma senha, uma ordem de decodificação é a, depois b. Os coeficientes decodificados “x_ac_dec[]” para “TCX” (ou seja, para uma decodificação de áudio que utiliza uma excitação codificada por transformação) são armazenados (por exemplo, diretamente) na matriz “x_tcx_invquant[win][bin]” e uma ordem da transmissão das senhas de codificação silenciosa é de modo que quando eles são decodificados na ordem recebida e armazenada na matriz, “bin” é o índice que aumenta mais rapidamente e “win” é o índice que aumenta mais lentamente. Dentro

de uma senha, a ordem de decodificação é a, depois b.

324. Primeiro, o indicador “arith_reset_flag” determina se o contexto deve ser redefinido. Se o indicador for verdadeiro, isto é considerado na função “arith_map_context”.

325. O processo de decodificação começa com uma fase de inicialização onde o vetor do elemento de contexto “q” é atualizado copiando e mapeando os elementos de contexto da estrutura prévia armazenada em “q[1][]” em “q[0][]”. Os elementos de contexto dentro de “q” são armazenados em um 4 bits por 2 tuplos. Para detalhes, a referência é feita ao código do pseudo-programa da figura 5a.

326. O decodificador silencioso emite 2 tuplos de coeficientes espectrais quantizados não sinalizados. Primeiramente, o estado c do contexto é calculado com base nos coeficientes espectrais previamente decodificados ao redor do tuplo duplo para decodificar. Desta forma, o estado é adicionalmente atualizado utilizando o estado de contexto do último tuplo duplo decodificado considerando somente dois novos tuplos duplos. O estado é decodificado em 17 bits e é retornado pela função “arith_get_context”. Uma representação do código do pseudo-programa da função “arith_get_context” é mostrada na figura 5c.

327. O estado de contexto c determina a tabela de frequências cumulativas utilizada para decodificar o plano de bit a bit mais significativo m. O mapeamento de c ao índice da tabela de frequências cumulativas correspondente “pki” é realizado por uma função “arith_get_pk()”. Uma representação do código do pseudo-programa da função “arith_get_pk()” é mostrada na figura

5e.

328. 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 por “arith_get_pk()”. O codificador (e decodificador) aritmético é uma implementação do número inteiro utilizando um método de geração de identificação com escala. O código do pseudo-programa de acordo com a figura 5g descreve o algoritmo utilizado.

329. Quando o valor decodificado m for o símbolo de escape “ARITH_ESCAPE”, as variáveis “lev” e “esc_nb” são aumentadas em 1 e outro valor m é decodificado. Neste caso, a função “get_pk()” é chamada mais uma vez com o valor “c+ esc_nb<<17” como argumento de entrada, onde “esc_nb” é o número de símbolos de escape previamente decodificados para o mesmo tuplo duplo e delimitado a 7.

330. Visto que o valor m não é o símbolo de escape “ARITH_ESCAPE”, o decodificador verifica se o m sucessivo forma um símbolo “ARITH_STOP”. Se a condição “(esc_nb>0&&m==0)” for verdadeira, o símbolo “ARITH_STOP” é detectado e o processo de decodificação é finalizado. O decodificador vai diretamente para a decodificação do sinal descrita posteriormente. A condição significa que o restante da estrutura é composto por 0 valores.

331. Se o símbolo “ARITH_STOP” não for encontrado, os planos de bits restantes são então decodificados, se houver algum, para o presente tuplo duplo. Os planos de bits restantes são decodificados do nível mais significativo ao menos significativo, chamando “arith_decode()” lev várias vezes com a tabela de frequências cumulativas “arith_cf_r[]”. Os planos de bits

decodificados r permitem a refinação do valor previamente decodificado m, de acordo com o algoritmo de um código do pseudo-programa que é mostrado na figura 5j. Neste ponto, o valor não sinalizado do tuplo duplo (a,b) é completamente decodificado. É salvo no elemento que mantém os coeficientes espectrais de acordo com o algoritmo, uma representação do código do pseudo-programa que é mostrada na figura 5k.

332. O contexto “q” é também atualizado para o próximo tuplo duplo. Deve ser observado que esta atualização de contexto também deve ser realizada para o último tuplo duplo. Esta atualização de contexto é realizada pela função “arith_update_context()”, uma representação do código do pseudo-programa que é mostrada na figura 5l.

333. O próximo tuplo duplo da estrutura é então decodificado aumentando i em 1 e refazendo o mesmo processo conforme descrito acima, começando pela função “arith_get_context()”. Quando os 2 tuplos lg/2 são decodificados dentro da estrutura, ou quando o símbolo de parada “ARITH_STOP” ocorre, o processo de decodificação da amplitude espectral termina e a decodificação dos sinais começa.

334. A decodificação é finalizada chamando uma função “arith_finish()”. Os coeficientes espectrais restantes são definidos a 0. Os respectivos estados de contexto são atualizados correspondentemente. Uma representação do código do pseudo-programa da função “arith_finish” é mostrada na figura 5m.

335. Visto que todos os coeficientes espectrais quantizados não sinalizados são decodificados, o sinal de acordo é adicionado. Para cada valor quantizado não nulo de “x_ac_dec”, um

bit é lido. Se o valor de bit de leitura for igual a 0, o valor quantizado é positivo, e nada é feito, e o valor sinalizado é igual ao valor não sinalizado previamente decodificado. Caso contrário, o coeficiente decodificado é negativo e o complemento de dois é considerado do valor não sinalizado. Os bits sinalizados são lidos das frequências baixas às altas.

11.12 Legendas

336. A figura 5q mostra uma legenda das definições que está relacionada aos algoritmos de acordo com as figuras 5a, 5c, 5e, 5f, 5g, 5j, 5k, 5l, e 5m.

337. A figura 5r mostra uma legenda das definições que está relacionada aos algoritmos de acordo com as figuras 5b, 5d, 5f, 5h, 5i, 5n, 5o, e 5p.

Tabelas de mapeamento

338. Em uma realização, de acordo com a invenção, particularmente tabelas vantajosas “ari_lookup_m”, “ari_hash_m”, e “ari_cf_m” são utilizadas para a execução da função “arith_get_pk()” de acordo com a figura 5e ou a figura 5f, e para a execução da função “arith_decode()” conforme discutido com referência às figuras 5g, 5h e 5i. Entretanto, deve ser observado que diferentes tabelas podem ser utilizadas em algumas realizações de acordo com a invenção.

12.1 Tabela “ari_hash_m[600]” de acordo com a figura 22

339. Um conteúdo de uma implementação particularmente vantajosa da tabela “ari_hash_m”, que é utilizado pela função “arith_get_pk”, uma primeira realização que foi descrita com referência à figura 5e, e uma segunda realização que foi descrita

com referência à figura 5f, é mostrada na tabela da figura 22. Deve ser observado que a tabela da figura 22 lista as 600 entradas da tabela (ou matriz) “ari_hash_m[600]”. Também deve ser observado que a representação em tabela da figura 22 mostra os elementos em uma ordem do índice de elementos, de modo que o primeiro valor “0x000000100UL” corresponda à entrada de tabela “ari_hash_m[0]” tendo um índice de elemento (ou índice da tabela) 0, e de modo que o último valor “0x7ffffffff4fUL” corresponda à entrada de tabela “ari_hash_m[599]” tendo o índice de elemento ou índice da tabela 599. Ainda deve ser observado aqui que “0x” indica que as entradas de tabela da tabela “ari_hash_m[]” são representadas em um formato hexadecimal. Além disso, deve ser observado aqui que o sufixo “UL” indica que as entradas de tabela da tabela “ari_hash_m[]” são representadas como valores de número inteiro “longos” não sinalizados (tendo uma precisão de 32 bits).

340. Além disso, deve ser observado que as entradas de tabela da tabela “ari_hash_m[]” de acordo com a figura 22 são dispostas em uma ordem numérica, a fim de permitir a execução da pesquisa da tabela 506b, 508b, 510b da função “arith_get_pk()”.

341. Ainda deve ser observado que os 24 bits mais significativos das entradas de tabela da tabela “ari_hash_m” representam certos valores de estado significativos, enquanto os 8 bits menos significativos representam valores de índice de regra de mapeamento “pki”. Assim, as entradas da tabela “ari_hash_m[]” descrevem um mapeamento de “impacto direto” de um valor de contexto em um valor de índice de regra de mapeamento “pki”.

342. Entretanto, os 24 bits mais altos das entradas da tabela “ari_hash_m[]” representam, ao mesmo tempo, limites de

intervalos de valores de contexto numéricos, no quais o mesmo valor de índice de regra de mapeamento está associado. Detalhes referentes a este conceito já foram discutidos acima.

12.2 Tabela “ari_lookup_m” de acordo com a figura 21

343. Um conteúdo de uma realização particularmente vantajosa da tabela “ari_lookup_m” é mostrado na tabela da figura 21. Deve ser observado aqui que a tabela da figura 21 lista as entradas da tabela “ari_lookup_m”. As entradas são referenciadas por um índice de entrada do tipo número inteiro unidimensional (também designado como “índice de elemento” ou “índice da matriz” ou “índice da tabela”) que é, por exemplo, designado como “i_max” ou “i_min”. Deve ser observado que a tabela “ari_lookup_m”, que compreende um total de 600 entradas, é bem adequada para uso pela função “arith_get_pk” de acordo com a figura 5e ou a figura 5f. Também deve ser observado que a tabela “ari_lookup_m” de acordo com a figura 21 é adaptada para cooperar com a tabela “ari_hash_m” de acordo com a figura 22.

344. Deve ser observado que as entradas da tabela “ari_lookup_m[600]” são listadas em ordem crescente do índice da tabela “i” (por exemplo, “i_min” ou “i_max”) entre 0 e 599. O termo “0x” indica que as entradas da tabela são descritas em um formato hexadecimal. Assim, a primeira entrada de tabela “0x02” corresponde à entrada de tabela “ari_lookup_m[0]” tendo o índice da tabela 0 e a última entrada de tabela “0x5E” corresponde à entrada de tabela “ari_lookup_m[599]” tendo o índice da tabela 599.

345. Também deve ser observado que as entradas da

tabela “ari_lookup_m[]” são associadas aos intervalos definidos pelas entradas adjacentes da tabela “arith_hash_m[]”. Assim, as entradas da tabela “ari_lookup_m” descrevem valores de índice de regra de mapeamento associados aos intervalos de valores de contexto numéricos, em que os intervalos são definidos pelas entradas da tabela “arith_hash_m”.

12.3. Tabela “ari_cf_m[96][17]” de acordo com a figura 23

346. A figura 23 mostra um conjunto de 96 tabelas de frequências cumulativas (ou sub-tabelas) “ari_cf_m[pki][17]”, uma que é selecionada pelo codificador de áudio 100, 700 ou um decodificador de áudio 200, 800, por exemplo, para a execução da função “arith_decode()”, ou seja, para uma decodificação do valor do plano de bits mais significativo. A selecionada das 96 tabelas de frequências cumulativas (ou sub-tabelas) mostradas na figura 23 considera uma função da tabela “cum_freq[]” na execução da função “arith_decode()”.

347. Como pode ser visto da figura 23, cada sub-bloco representa a tabela de frequências cumulativas tendo 17 entradas. Por exemplo, um primeiro sub-bloco 2310 representa as 17 entradas da tabela de frequências cumulativas para “pki=0”. Um segundo sub-bloco 2312 representa as 17 entradas da tabela de frequências cumulativas para “pki=1”. Finalmente, um 96° sub-bloco 2396 representa as 17 entradas da tabela de frequências cumulativas para “pki=95”. Assim, a figura 23 efetivamente representa 96 diferentes tabelas de frequências cumulativas (ou sub-tabelas) de “pki=0” para “pki=95”, em que cada uma das 96 tabelas de frequências cumulativas é representada por um sub-bloco (envolvido

por suportes enrolados), e em que cada uma das ditas tabelas de frequências cumulativas compreende 17 entradas.

348. Dentro de um sub-bloco (por exemplo, um sub-bloco 2310 ou 2312, ou um sub-bloco 2396), um primeiro valor descreve uma primeira entrada da tabela de frequências cumulativas (tendo um índice da matriz ou índice da tabela de 0), e um último valor descreve uma última entrada da tabela de frequências cumulativas (tendo um índice da matriz ou índice da tabela de 16).

349. Assim, cada sub-bloco 2310, 2312, 2396 da representação da tabela da figura 23 representa as entradas da tabela de frequências cumulativas para uso pela função “arith_decode” de acordo com a figura 5g, ou de acordo com as figuras 5h e 5i. A variável de entrada “cum_freq[]” da função “arith_decode” descreve qual das 96 tabelas de frequências cumulativas (representadas pelos sub-blocos individuais de 17 entradas da tabela “arith_cf_m”) deve ser utilizada para a decodificação dos coeficientes espectrais correntes.

12.4 Tabela “ari_cf_r[]” de acordo com a figura 24

350. A figura 24 mostra um conteúdo da tabela “ari_cf_r[]”.

351. As quatro entradas da dita tabela são mostradas na figura 24. Entretanto, deve ser observado que a tabela “ari_cf_r” pode eventualmente ser diferente em outras realizações.

Avaliação de desempenho e vantagens

352. As realizações, de acordo com a invenção, utilizam as funções atualizadas (ou algoritmos) e um conjunto de tabelas atualizadas, conforme discutido acima, a fim de obter uma

desvantagem melhorada entre a complexidade computacional, exigência de memória, e eficiência da codificação.

353. Falando de forma geral, as realizações, de acordo com a invenção, criam uma codificação espectral silenciosa melhorada. As realizações, de acordo com a presente invenção, descrevem uma melhoria da codificação espectral silenciosa em USAC (codificação de áudio e voz unificada).

354. As realizações, de acordo com a invenção, criam uma proposta atualizada para CE na codificação espectral silenciosa melhorada de coeficientes espectrais, com base nos esquemas conforme apresentado nos papéis de entrada MPEG m16912 e m17002. Ambas as propostas foram avaliadas, resultados potenciais eliminados e as forças combinadas.

355. Como em m16912 e m17002, a proposta resultante tem como base o contexto original com base no esquema de codificação aritmético como o projeto de trabalho 5 USAC (o padrão de projeto na codificação de áudio e voz unificada), mas pode significantemente reduzir as exigências de memória (memória de acesso aleatório (RAM) e memória para somente leitura (ROM)) sem aumentar a complexidade computacional, enquanto mantém a eficiência da codificação. Além disso, foi provado ser possível uma transcodificação sem perdas de fluxos de bits de acordo com o projeto de trabalho 3 do Padrão de Projeto USAC e de acordo com o projeto de trabalho 5 do Padrão de Projeto USAC. As realizações, de acordo com a invenção, têm o objetivo de substituir o esquema espectral de codificação silenciosa conforme utilizado no projeto de trabalho 5 do Padrão de Projeto USAC.

356. O esquema de codificação aritmético descrito aqui

tem como base o esquema como no modelo de referência 0 (RM0) ou o projeto de trabalho 5 (WD) do Padrão de Projeto USAC. Os coeficientes espectrais na frequência ou em tempo modelam um contexto. Este contexto é utilizado para a seleção de tabelas de frequências cumulativas para o codificador aritmético. Comparado ao projeto de trabalho 5 (WD), o modelamento de contexto é ainda melhorado e as tabelas que mantém as probabilidades do símbolo foram treinadas novamente. O número de diferentes modelos de probabilidade foi elevado de 32 a 96.

357. As realizações, de acordo com a invenção, reduzem o tamanho das tabelas (demanda da ROM de dados) a 1518 palavras de 32 bits de comprimento ou 6072-bytes (WD 5: 16.894,5 palavras ou 67.578-bytes). A demanda de RAM estática é reduzida de 666 palavras (2.664 bytes) para 72 palavras (288 bytes) por canal do codificador de núcleo. Ao mesmo tempo, preserva completamente o desempenho da codificação e pode ainda atingir um ganho de aproximadamente 1,29 a 1,95% comparado a taxa de dados total sobre todos os 9 pontos operacionais. Todos os fluxos de bits do projeto de trabalho 3 e do projeto de trabalho 5 podem ser transcodificados sem perdas, sem afetar as restrições do reservatório de bit.

358. A seguir, uma breve descrição dos conceitos de codificação de acordo com o projeto de trabalho 5 do Padrão de Projeto USAC será provida para facilitar a compreensão das vantagens do conceito descrito aqui. Subsequentemente, algumas realizações preferidas, de acordo com a invenção, serão descritas.

359. No projeto de trabalho 5 USAC, um contexto com base no esquema de codificação aritmético é utilizado para a

codificação silenciosa de coeficientes espectrais quantizados. Como o contexto, os coeficientes decodificados espectrais são utilizados, que são prévios na frequência e tempo. No projeto de trabalho 5, um número máximo de 16 coeficientes espectrais é utilizado como contexto, 12 deles sendo prévios em tempo. Ainda, os coeficientes espectrais utilizados para o contexto e ser decodificados, são agrupados como conjuntos de 4 tuplos (ou seja, 4 coeficientes espectrais próximos na frequência, ver a figura 14a). O contexto é reduzido e mapeado em uma tabela de frequências cumulativas, que é então utilizado para decodificar o próximo conjunto de 4 tuplos de coeficientes espectrais.

360. Para o esquema de codificação silenciosa completa do projeto de trabalho 5, uma demanda de memória (memória para somente leitura (ROM)) de 16894,5 palavras (67578 byte) é necessária. Adicionalmente, 666 palavras (2664 byte) da RAM estática por canal do codificador de núcleo são necessárias para armazenar os estados para a próxima estrutura. A representação em tabela da figura 14b descreve as tabelas conforme utilizadas no esquema de codificação aritmética USAC WD4.

361. Deve ser observado aqui que com relação à codificação silenciosa, os projetos de trabalho 4 e 5 do padrão de projeto USAC são os mesmos. Ambos usam o mesmo codificador silencioso.

362. Uma demanda de memória total de um decodificador completo USAC WD5 é estimada a 37000 palavras (148000-byte) para ROM de dados sem código do programa e 10000 a 17000 palavras para a RAM estática. Pode ser claramente visto que as tabelas do codificador silencioso consomem aproximadamente 45% da demanda

total da ROM de dados. A maior tabela individual já consome 4096 palavras (16384-byte).

363. Foi observado que, o tamanho da combinação de todas as tabelas e as grandes tabelas individuais excedem os tamanhos de cache típicos conforme providos por processadores de ponto fixo utilizados nos dispositivos portáteis do consumidor, que está em uma faixa típica de 8 a 32 Kbytes (por exemplo, ARM9e, TI C64XX etc.). Isto significa que o conjunto de tabelas pode provavelmente não ser armazenado na RAM de dados rápidos, que permite um acesso aleatório rápido aos dados. Isto faz com que todo o processo de decodificação fique lento.

364. Além disso, foi observado que a tecnologia atual da codificação de áudio com sucesso como HE-AAC foi aprovada para ser implementada na maioria dos dispositivos móveis. HE-AAC utiliza um esquema de codificação de Huffman de entropia com um tamanho da tabela de 995 palavras. Para detalhes, a referência é feita a ISO/IEC JTC1/SC29/WG11 N2005, MPEG98, Fevereiro 1998, San Jose, "*Revised Report on Complexity of MPEG-2 AAC2*".

365. No 90° Encontro MPEG, em papéis de entrada de MPEG m16912 e m17002, duas propostas foram apresentados cujo objetivo é reduzir as exigências de memória e melhorar a eficiência da codificação do esquema de codificação silenciosa. Analisando ambas as propostas, as conclusões a seguir poderiam ser consideradas.

366. Uma redução significativa da demanda de memória é possível reduzindo a dimensão da senha. Conforme mostrado no documento de entrada de MPEG m17002, reduzindo a dimensão dos conjuntos de 4 tuplos para conjuntos de 1 tuplo, a demanda de

memória poderia ser reduzida de 16984,5 para 900 palavras sem infringir na eficiência da codificação; e

367. A redundância adicional poderia ser removida aplicando um livro de código da distribuição de probabilidade não uniforme para a codificação LSB, em vez de utilizar a distribuição de probabilidade uniforme.

368. No curso destas avaliações, foi identificado que o movimento a partir de um esquema de codificação de um conjunto de 4 tuplos para um conjunto de 1 tuplo, teve um impacto significativo na complexidade computacional: uma redução da dimensão da codificação aumenta pelo mesmo fator que o número de símbolos a codificar. Isto significa que para a redução dos conjuntos de 4 tuplos para conjuntos de 1 tuplo, as operações necessárias para determinar o contexto, acessar as tabelas *hash* e decodificar o símbolo devem ser realizadas com quatro vezes mais frequência que antes. Com um algoritmo mais sofisticado para determinado contexto, isto leva a um aumento na complexidade computacional por um fator de 2,5 ou x.xxPCU.

369. A seguir, o novo esquema proposto, de acordo com as realizações da presente invenção, será brevemente descrito.

370. Para superar a questão de consumo de memória e a complexidade computacional, um esquema de codificação silenciosa melhorado é proposto para substituir o esquema como no projeto de trabalho 5 (WD5). O foco principal no desenvolvimento foi colocado na redução da demanda de memória, enquanto mantém a eficiência de compressão e não aumenta a complexidade computacional. Mais especificamente, a meta era atingir uma boa troca (ou a melhor) no espaço de complexidade multi-dimensão do desempenho de compressão,

complexidade e exigências de memória.

371. A proposta do novo esquema de codificação empresta a característica principal do codificador silencioso WD5, a saber, a adaptação do contexto. O contexto é derivado utilizando coeficientes espectrais previamente decodificados, tão próximo quanto em WD5 da estrutura passada e presente (em que uma estrutura pode ser considerada como uma parte do conteúdo de áudio). Entretanto, os coeficientes espectrais são agora codificados combinando dois coeficientes juntos para formar um tuplo duplo. Outra diferença permanece no fato de que os coeficientes espectrais são agora divididos em três partes, o sinal, os bits mais significativos ou o bit mais significativo (MSBs) e os bits menos significativos ou o bit menos significativo (LSBs). O sinal é codificado independentemente da magnitude que é ainda dividida em duas partes, os bits mais significativos (ou bits mais significativos) e o restante dos bits (ou bits menos significativos), se existirem. Os 2 tuplos nos quais a magnitude dos dois elementos é inferior ou igual a 3 são codificados diretamente pela codificação de MSBs. Caso contrário, uma senha de escape é transmitida primeiramente para sinalizar qualquer plano de bits adicional. Na versão base, a informação ausente, os LSBs e o sinal, são ambos codificados utilizando a distribuição uniforme de probabilidade. De modo alternativo, uma diferente distribuição de probabilidade pode ser utilizada.

372. A redução do tamanho da tabela é ainda possível, visto que:

somente probabilidades para 17 símbolos precisam ser armazenadas: símbolo {[0;+3], [0;+3]}+ESC;

não há necessidade em armazenar uma tabela de agrupamento (egrupos, dgrupos, dgvetores);

o tamanho da tabela *hash* pode ser reduzido com um treinamento realizado.

373. A seguir, alguns detalhes referentes à codificação de MSBs serão descritos. Conforme já mencionado, uma das principais diferenças entre WD5 do Padrão de Projeto USAC, uma proposta submetida no 90° Encontro de MPEG e a proposta atual é a dimensão dos símbolos. Em WD5 do Padrão de Projeto USAC, conjuntos de 4 tuplos foram considerados para a geração de contexto e a codificação silenciosa. Em uma proposta submetida no 90° Encontro de MPEG, os conjuntos de 1 tuplo foram utilizados em vez de reduzir as exigências da ROM. Durante o desenvolvimento, os 2 tuplos foram os mais comprometidos para reduzir as exigências da ROM, sem aumentar a complexidade computacional. Em vez de considerar quatro conjuntos de 4 tuplos para a inovação do contexto, agora quatro tuplos duplos são considerados. Conforme mostrado na figura 15a, três tuplos duplos surgem da estrutura passada (também designada como uma parte prévia do conteúdo de áudio) e outra surge da estrutura presente (também designada como a parte corrente do conteúdo de áudio).

374. A redução no tamanho da tabela é feita devido a três fatores principais. Primeiro, somente probabilidades para 17 símbolos precisam ser armazenadas (ou seja, símbolo {[0;+3], [0;+3]} + ESC). As tabelas de agrupamento (ou seja, egrupos, dgrupos, e dgvetores) não são mais necessárias. Finalmente, o tamanho da tabela *hash* foi reduzido realizando um treinamento apropriado.

375. Embora a dimensão fosse reduzida de quatro para dois, a complexidade foi mantida na faixa em WD5 do Padrão de Projeto USAC. Foi obtido simplificando tanto a geração de contexto como o acesso da tabela *hash*.

376. As diferentes simplificações e otimizações foram feitas de forma que o desempenho da codificação não foi afetado, e mais levemente melhorado. Foi obtido principalmente aumentando o número de modelos de probabilidade de 32 para 96.

377. A seguir, alguns detalhes referentes à codificação LSBs serão descritos. As LSBs são codificadas com uma distribuição uniforme de probabilidade em algumas realizações. Comparado ao WD5 do Padrão de Projeto USAC, as LSBs são agora consideradas dentro dos 2 tuplos em vez dos conjuntos de 4 tuplos.

378. A seguir alguns detalhes referentes à codificação do sinal serão explicados. O sinal é codificado sem utilizar o codificador de núcleo aritmético para redução de complexidade. O sinal é transmitido em 1 bit somente quando a magnitude correspondente é não nula. 0 significa um valor positivo e 1 significa um valor negativo.

379. A seguir, alguns detalhes referentes à demanda de memória serão explicados. O novo esquema proposto exibe uma demanda de ROM total de no máximo 1522.5 novas palavras (6090-bytes). Para detalhes, a referência é feita à tabela da figura 15b, que descreve as tabelas conforme utilizadas no esquema de codificação proposto. Comparado à demanda de ROM do esquema de codificação silenciosa em WD5 do Padrão de Projeto USAC, a demanda de ROM é reduzida a pelo menos 15462 palavras (61848 bytes). Agora finaliza na mesma ordem de magnitude que a exigência de memória

necessária para o decodificador de Huffman AAC em HE-AAC (995 palavras ou 3980-bytes). Para detalhes, a referência é feita à ISO/IEC JTC1/SC29/WG11 N2005, MPEG98, Fevereiro 1998, San Jose, *"Revised Report on Complexity of MPEG-2 AAC2"*, e também à figura 16a. Isto reduz toda a demanda de ROM do codificador silencioso para mais que 92% e um decodificador completo de USAC de aproximadamente 37000 palavras a aproximadamente 21500 palavras, ou para mais que 41%. Para detalhes, a referência é novamente feita às figuras 16a e 16b, em que a figura 16a mostra uma demanda de ROM de um esquema de codificação silenciosa conforme proposto, e de um esquema de codificação silenciosa de acordo com WD4 do Padrão de Projeto USAC, e em que a figura 16b mostra uma demanda de dados total da ROM do decodificador USAC de acordo com o esquema proposto e de acordo com WD4 do Padrão de Projeto USAC.

380. Ainda, uma quantidade de informação necessária para a derivação de contexto na próxima estrutura (ROM estática) é também reduzida. Em WD5 do Padrão de Projeto USAC, o conjunto completo de coeficientes (um máximo de 1152 coeficientes) com uma resolução de tipicamente 16 bits adicionais a um índice de grupo por conjunto de 4 tuplos de uma resolução de 10 bits necessária para ser armazenada, que soma até 666 palavras (2664-bytes) por canal do codificador de núcleo (decodificador WD4 completo USAC: aproximadamente 10000 a 17000 palavras). O novo esquema reduz a informação persistente para somente 2 bits por coeficiente espectral, que soma até 72 palavras (288-byte) no total por canal do codificador de núcleo. A demanda da memória estática pode ser reduzida a 594 palavras (2376-byte).

381. A seguir, alguns detalhes referentes ao possível

aumento de eficiência da codificação serão descritos. A eficiência da decodificação das realizações, de acordo com a nova proposta, foi comparada com os fluxos de bits da qualidade de referência de acordo com o projeto de trabalho 3 (WD3) e WD5 do Padrão de Projeto USAC. A comparação foi realizada por meio de um transcodificador, com base em um decodificador do software de referência. Para detalhes referentes à dita comparação da codificação silenciosa de acordo com WD3 ou WD5 do Padrão de Projeto USAC e o esquema de codificação proposto, a referência é feita à figura 17, que mostra uma representação esquemática de uma disposição do teste para uma comparação da codificação silenciosa WD3/5 com o esquema de codificação proposto.

382. Ainda, a demanda de memória nas realizações, de acordo com a invenção, foi comparada às realizações de acordo com WD3 (ou WD5) do Padrão de Projeto USAC.

383. A eficiência da codificação não é somente mantida, mas levemente aumentada. Para detalhes, a referência é feita à tabela da figura 18, que mostra uma representação em tabela de taxas de bits médios produzidos pelo codificador aritmético WD3 (ou um codificador de áudio USAC utilizando um codificador aritmético WD3), e um codificador de áudio (por exemplo, USAC codificador de áudio) de acordo com uma realização da invenção.

384. Detalhes sobre as taxas de bits médios por modo operacional podem ser encontrados na tabela da figura 18.

385. Além disso, a figura 19 mostra uma representação em tabela de níveis mínimo e máximo do reservatório de bits para o codificador aritmético WD3 (ou um codificador de áudio utilizando

o codificador aritmético WD3) e um codificador de áudio de acordo com uma realização da presente invenção.

386. A seguir, alguns detalhes referentes à complexidade computacional serão descritos. A redução da dimensionalidade da codificação aritmética geralmente leva a um aumento da complexidade computacional. De fato, a redução da dimensão por um fator de dois realizará as rotinas do codificador aritmético duas vezes.

387. Entretanto, foi observado que este aumento de complexidade pode ser limitado por várias otimizações introduzidas no novo esquema de codificação proposto de acordo com as realizações da presente invenção. A geração de contexto foi muito simplificada em algumas realizações de acordo com a invenção. Para cada tuplo duplo, o contexto pode ser muito atualizado a partir do último contexto gerado. As probabilidades são armazenadas agora em 14 bits em vez de 16 bits, o que evita as operações de 64 bits durante o processo de decodificação. Além disso, o mapeamento do modelo de probabilidade foi muito otimizado em algumas realizações de acordo com a invenção. O pior caso foi drasticamente reduzido e limitado a 10 iterações em vez de 95.

388. Como um resultado, a complexidade computacional do esquema de codificação silenciosa proposto foi mantida na mesma faixa que em WD5. Uma estimativa de “caneta e papel” foi realizada por diferentes versões da codificação silenciosa e é registrada na tabela da figura 20. Isto mostra que o novo esquema de codificação é apenas aproximadamente 13% menos complexo que um codificador aritmético WD5.

389. Para resumir o mencionado acima, pode ser visto

que realizações, de acordo com a presente invenção, provêm uma boa troca entre complexidade computacional, exigências de memória e eficiência da codificação.

14. Sintaxe do fluxo de bits

14.1 Payloads do Codificador espectral silencioso

390. A seguir, alguns detalhes referentes aos payloads do codificador espectral silencioso serão descritos. Em algumas realizações, há diversos modos diferentes de codificação como, por exemplo, o modo de codificação de “domínio de previsão linear” e o modo de codificação de “domínio de frequência”. No modo de codificação de domínio de previsão linear, uma forma de ruído é realizada com base em uma análise de previsão linear do sinal de áudio, e um sinal em forma de ruído é codificado no domínio de frequência. No modo de codificação de domínio de frequência, uma forma de ruído é realizada com base em uma análise psicoacústica e uma versão em forma de ruído do conteúdo de áudio é codificada no domínio de frequência.

391. Coeficientes espectrais do sinal codificado do “domínio de previsão linear” e do sinal codificado do “domínio de frequência” são quantizados por escala e então silenciosamente codificados por uma codificação aritmética dependente de contexto de forma adaptativa. Os coeficientes quantizados são unidos em 2 tuplos antes de serem transmitidos da frequência mais baixa para a frequência mais alta. Cada tuplo duplo é dividido em um sinal s, o plano de bit a bit mais significativo m, e um ou mais planos de bits menos significativos r restantes (se houver). O valor m é codificado de acordo com um contexto definido pelos coeficientes espectrais próximos. Em outras palavras, m é codificado de acordo

com os coeficientes próximos. Os planos de bits menos significativos restantes r são codificados por entropia sem considerar o contexto. Por meios de m e r, a amplitude destes coeficientes espectrais pode ser reconstruída no lado do decodificador. Para todos os símbolos não nulos, os sinais s são codificados fora do codificador aritmético utilizando 1 bit. Em outras palavras, os valores m e r formam os símbolos do codificador aritmético. Finalmente, os sinais s, são codificados fora do codificador aritmético utilizando 1 bit por coeficiente quantizado não nulo.

392. Um procedimento detalhado da codificação aritmética será descrito aqui.

14.2 Elementos de sintaxe

393. A seguir, a sintaxe do fluxo de bits de um fluxo de bits que carrega a informação espectral aritmeticamente codificada será descrita tendo como referência as figuras 6a a 6j.

394. A figura 6a mostra uma representação de sintaxe do bloco de dados brutos USAC (“usac_raw_data_block()”).

395. O bloco de dados brutos USAC compreende um ou mais elementos de canal único (“single_channel_element()”) e/ou um ou mais elementos do par de canal (“channel_pair_element()”).

396. Agora com referência à figura 6b, a sintaxe de um elemento de canal único é descrita. Os elementos de canal único compreendem um fluxo de canal do domínio de previsão linear (“lpd_channel_stream ()”) ou um fluxo de canal de domínio de frequência (“fd_channel_stream ()”) dependendo do modo de núcleo.

397. A figura 6c mostra uma representação de sintaxe de um elemento do par de canal. Um elemento do par de canal

compreende informação do modo de código (“core_mode0”, “core_mode1”). Além disso, o elemento do par de canal pode compreender uma informação de configuração “ics_info()”. Adicionalmente, dependendo da informação do modo de código, o elemento do par de canal compreende um fluxo de canal do domínio de previsão linear ou um fluxo de canal de domínio de frequência associado a um primeiro dos canais, e o elemento do par de canal também compreende um fluxo de canal do domínio de previsão linear ou um fluxo de canal de domínio de frequência associado a um segundo dos canais.

398. A informação de configuração “ics_info()”, uma representação de sintaxe que é mostrada na figura 6d, compreende diversos itens diferentes de informação de configuração, que não são de relevância particular para a presente invenção.

399. Um fluxo de canal de domínio de frequência (“fd_channel_stream ()”), uma representação de sintaxe que é mostrada na figura 6e, compreende uma informação de ganho (“global_gain”) e uma informação de configuração (“ics_info ()”). Além disso, o fluxo de canal de domínio de frequência compreende dados do fator de escala (“scale_factor_data ()”), que descrevem fatores de escala utilizados para a escala de valores espectrais de diferentes faixas de fator de escala, e que é aplicado, por exemplo, pelo multiplicador de frequências 150 e pelo remultiplicador de frequências 240. O fluxo de canal de domínio de frequência também compreende dados espectrais aritmeticamente codificados (“ac_spectral_data ()”), que representam valores espectrais aritmeticamente codificados.

400. Os dados espectrais aritmeticamente codificados

(“ac_spectral_data()”), uma representação de sintaxe que é mostrada na figura 6f, compreendem um sinalizador de redefinição aritmético opcional (“arith_reset_flag”), que é utilizado para seletivamente redefinir o contexto, conforme descrito acima. Além disso, os dados espectrais aritmeticamente decodificados compreendem diversos blocos de dados aritméticos (“arith_data”), que carregam os valores espectrais aritmeticamente codificados. A estrutura dos blocos de dados aritmeticamente codificados depende do número das faixas de frequência (representado pela variável “num_bands”) e também do estado do sinalizador de redefinição aritmético, conforme serão discutidos a seguir.

401. A seguir, a estrutura do bloco de dados aritmeticamente codificado será descrita tendo como referência a figura 6g, que mostra uma representação de sintaxe dos ditos blocos de dados aritmeticamente codificados. A representação de dados dentro do bloco de dados aritmeticamente codificados depende do número lg de valores espectrais a ser codificado, do status do sinalizador de redefinição aritmético e também do contexto, ou seja, os valores espectrais previamente codificados.

402. O contexto para a codificação do conjunto corrente (por exemplo, 2 tuplos) de valores espectrais é determinado de acordo com o algoritmo de determinação de contexto mostrado no número de referência 660. Detalhes com relação ao algoritmo de determinação de contexto foram explicados acima, tendo como referência as figuras 5a e 5b. O bloco de dados aritmeticamente codificados compreende lg/2 para definição das senhas, cada conjunto de senhas que representa diversos (por exemplo, um tuplo duplo) valores espectrais. Um conjunto de senhas

compreende uma senha aritmética “acod_m[pki][m]” que representa um valor do plano de bits mais significativo m do tuplo de valores espectrais utilizando entre 1 e 20 bits. Além disso, o conjunto de senhas compreende uma ou mais senhas “acod_r[r]” se o tuplo de valores espectrais requerer 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 14 bits.

403. Se, entretanto, um ou mais planos de bits menos significativos são necessários (além do plano de bits mais significativo) para uma representação correta dos valores espectrais, isto é sinalizado utilizando uma ou mais senhas aritméticas de escape (“ARITH_ESCAPE”). Assim, pode ser geralmente dito que para um valor espectral, é determinado 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 são necessários, isto é sinalizado por uma ou mais senhas aritméticas de escape “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 que 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 aritméticas de escape estiverem incluídas no fluxo de bits. Seguindo uma ou mais senhas aritméticas de escape, uma senha aritmética “acod_m[pki][m]” está incluída no fluxo de bits, conforme mostrado no número de referência 663, em que “pki” designa o índice atualmente válido do modelo de probabilidade

(considerando a adaptação do contexto causada pela inclusão das senhas aritméticas de escape em consideração) e em que m designa o valor do plano de bits mais significativo do valor espectral a ser codificado ou decodificado (em que m é diferente da senha “ARITH_ESCAPE”).

404. Conforme discutido acima, a presença de qualquer plano de bits menos significativo resulta em uma presença de uma ou mais senhas “acod_r[r]”, cada uma representa 1 bit de um plano de bits menos significativo de um primeiro valor espectral e cada um que também representa 1 bit de um plano de bits menos significativo de um segundo valor espectral. Uma ou mais senhas “acod_r[r]” são codificadas de acordo com uma tabela de frequências cumulativas correspondente, que pode, por exemplo, ser constante e independente de contexto. Entretanto, diferentes mecanismos para a seleção da tabela de frequências cumulativas para uma decodificação de uma ou mais senhas “acod_r[r]” são possíveis.

405. Além disso, deve ser observado que o contexto é atualizado depois da codificação de cada tuplo de valores espectrais, conforme mostrado no número de referência 668, de modo que o contexto seja tipicamente diferente da codificação e decodificação de dois tuplos de valores espectrais subsequente.

406. A figura 6i mostra uma legenda de definições e elementos de ajuda que definem a sintaxe do bloco de dados aritmeticamente codificado.

407. Além disso, uma sintaxe alternativa dos dados aritméticos “arith_data()” é mostrada na figura 6h, com uma legenda correspondente de definições e elementos de ajuda mostrada

na figura 6j.

408. Para resumir o mencionado acima, um formato do 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 aritmeticamente codificados é codificado de modo que se encaixe no algoritmo de decodificação discutido acima.

409. Além disso, geralmente deve ser observado que a codificação é a operação inversa da decodificação, de modo que possa geralmente ser assumido que o codificador realiza uma consulta de tabela utilizando as tabelas mencionadas acima, que é aproximadamente inversa à consulta de tabela realizada pelo decodificador. Geralmente, pode ser dito que um técnico no assunto que conhece o algoritmo de decodificação e/ou a sintaxe do fluxo de bits desejada poderá facilmente desenhar um codificador aritmético, que provê os dados definidos na sintaxe do fluxo de bits e necessários por um decodificador aritmético.

410. Além disso, deve ser observado que os mecanismos para determinar o valor de contexto corrente numérico e para derivar um valor de índice de regra de mapeamento podem ser idênticos em um codificador de áudio e um decodificador de áudio, pois é tipicamente desejado que o decodificador de áudio utilize o mesmo contexto ao codificador de áudio, de modo que a decodificação seja adaptada à codificação.

15. Alternativas de implementação

411. Embora alguns aspectos foram descritos no contexto de um aparelho, é claro que estes aspectos também representam uma descrição do método correspondente, onde um bloco

ou dispositivo corresponde a uma etapa do método ou uma característica de uma etapa do método. De forma equivalente, os aspectos descritos no contexto de uma etapa do método também representam uma descrição de um bloco ou item ou característica correspondente de um aparelho correspondente. Algumas ou todas as etapas do método podem ser executadas por (ou utilizando) um aparelho de hardware como, por exemplo, um microprocessador, um computador programável ou um circuito eletrônico. Em algumas realizações, uma ou mais das etapas mais importantes do método podem ser executadas por tal aparelho.

412. 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 com fio como a Internet.

413. Dependendo de certas 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 Blue-Ray, um CD, uma ROM, uma PROM, uma EPROM, uma EEPROM ou uma memória FLASH, tendo sinais de controle eletricamente legíveis armazenados nele, que coopera (ou são capazes de cooperar) com um sistema de computador programável de modo que o respectivo método seja realizado. Desta forma, o meio de armazenamento digital pode ser legível por computador.

414. Algumas realizações, de acordo com a invenção, compreendem um carregador de dados tendo sinais de controle eletricamente legíveis, que podem cooperar com um sistema de computador programável, de modo que um dos métodos descritos aqui

seja realizado.

415. Geralmente, as realizações da presente invenção podem ser implementadas como um produto de programa de computador com um código do programa, o código do programa sendo operativo para realizar um dos métodos quando o produto de programa de computador opera em um computador. O código do programa pode, por exemplo, ser armazenado em um carregador legível por máquina.

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

417. Em outras palavras, uma realização do método inventivo é, desta forma, um programa de computador tendo um código do programa para realizar um dos métodos descritos aqui, quando o programa de computador opera em um computador.

418. Outra realização do método inventivo é, desta forma, um carregador de dados (ou um meio de armazenamento digital, ou um meio legível por computador) que compreende, registrado nele, o programa de computador para realizar um dos métodos descritos aqui. O carregador de dados, o meio de armazenamento digital ou o meio registrado são tipicamente tangíveis e/ou não transitórios.

419. Outra realização do método inventivo é, desta forma, um fluxo de dados ou uma sequência de sinais que representa o programa de computador para realizar um dos métodos descritos aqui. O fluxo de dados ou a sequência de sinais pode, por exemplo, ser configurado para ser transferido através de uma conexão de comunicação de dados, por exemplo, através da Internet.

420. Outra realização compreende um meio de

processamento, por exemplo, um computador, ou um dispositivo lógico programável, configurado ou adaptado para realizar um dos métodos descritos aqui.

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

422. Outra realização, de acordo com a invenção, compreende um aparelho ou um sistema configurado para transferir (por exemplo, eletronicamente ou opticamente) um programa de computador para realizar um dos métodos descritos aqui a um receptor. O receptor pode, por exemplo, ser um computador, um dispositivo móvel, um dispositivo de memória ou semelhantes. O aparelho ou sistema pode, por exemplo, compreender um servidor de arquivo para transferir o programa de computador ao receptor.

423. Em algumas realizações, um dispositivo lógico programável (por exemplo, uma matriz de portas programáveis em campo) pode ser utilizado para realizar algumas ou todas as funcionalidades dos métodos descritos aqui. Em algumas realizações, uma matriz de portas programáveis em campo pode cooperar com um microprocessador a fim de realizar um dos métodos descritos aqui. Geralmente, os métodos são preferivelmente realizados por qualquer aparelho de hardware.

424. As realizações descritas acima são meramente ilustrativas para os princípios da presente invenção. É entendido que as modificações e variações das disposições e os detalhes descritos aqui serão evidentes a outros técnicos no assunto. É o objetivo, desta forma, ser limitado somente pelo escopo das reivindicações da patente iminente e não pelos detalhes

específicos apresentados em forma de descrição e explicação das realizações aqui.

16. Conclusões

425. Para concluir, as realizações, de acordo com a invenção, compreendem um ou mais dos aspectos a seguir, em que os aspectos podem ser utilizados individualmente ou em combinação.

Estado de mecanismo de contexto de *hashing*

426. De acordo com um aspecto da invenção, os estados na tabela *hash* são considerados como estados significativos e limites de grupo. Isto permite reduzir significantemente o tamanho das tabelas necessárias.

Atualização adicional de contexto

427. De acordo com um aspecto, algumas realizações, de acordo com a invenção, compreendem uma forma computacionalmente eficiente para atualizar o contexto. Algumas realizações utilizam uma atualização adicional de contexto na qual um valor de contexto corrente numérico é derivado de um valor de contexto prévio numérico.

Derivação de contexto

428. De acordo com um aspecto da invenção, utilizando uma soma de dois valores absolutos espectrais está a associação de um truncamento. É um tipo de quantização do vetor de ganho dos coeficientes espectrais (como a oposição à quantização do vetor de ganho convencional). Objetiva limitar a ordem do contexto, enquanto transporta a informação mais significativa do próximo.

429. Outras tecnologias, que são aplicadas nas realizações, de acordo com a invenção, são descritas nos pedidos de patente não pré-publicados PCT EP2101/065725, PCT

EP2010/065726, e PCT EP 2010/065727. Além disso, em algumas realizações, de acordo com a invenção, um símbolo de parada é utilizado. Além disso, em algumas realizações, somente os valores não sinalizados são considerados para o contexto.

430. Entretanto, os pedidos de patente Internacional não pré-publicados revelam os aspectos que estão ainda em algumas realizações de acordo com a invenção.

431. Por exemplo, uma identificação de região zero é utilizada em algumas realizações da invenção. Assim, o chamado "indicador de pequeno valor" é definido (por exemplo, bit 16 do valor de contexto corrente numérico c).

432. Em algumas realizações, o cálculo de contexto dependente da região pode ser utilizado. Entretanto, em outras realizações, um cálculo de contexto dependente da região pode ser omitido a fim de manter a complexidade e o tamanho das tabelas razoavelmente pequenas.

433. Além disso, o contexto de *hashing* utilizando uma função *hash* é um aspecto importante da invenção. O contexto de *hashing* pode ter como base o conceito de duas tabelas que é descrito nos pedidos de patente Internacional não pré-publicados referenciados acima. Entretanto, adaptações específicas do contexto de *hashing* podem ser utilizadas em algumas realizações a fim de aumentar a eficiência computacional. Entretanto, em outras realizações, de acordo com a invenção, o contexto de *hashing* que é descrito nos pedidos de patente Internacional não pré-publicados referenciados acima pode ser utilizado.

434. Além disso, deve ser observado que o contexto *hashing* adicional é bastante simples e computacionalmente

eficiente. Ainda, a independência de contexto dos sinais de valores, que é utilizada em algumas realizações da invenção, ajuda a simplificar o contexto, assim mantendo as exigências de memória razoavelmente baixas.

435. Em algumas realizações da invenção, uma derivação de contexto utilizando uma soma de dois valores espectrais e uma limitação de contexto é utilizada. Estes dois aspectos podem ser combinados. Ambos objetivam limitar a ordem de contexto transportando a informação mais significativa do próximo.

436. Em algumas realizações, um indicador de pequeno valor é utilizado que pode ser semelhante a uma identificação de um grupo de diversos valores zero.

437. Em algumas realizações, de acordo com a invenção, um mecanismo de parada aritmética é utilizado. O conceito é semelhante ao uso de um símbolo “fim do bloco” em JPEG, que tem uma função comparável. Entretanto, em algumas realizações da invenção, o símbolo (“ARITH_STOP”) não é incluído explicitamente no codificador de entropia. Ainda, uma combinação de símbolos já existentes, que não poderia ocorrer previamente, é utilizada, ou seja, “ESC+0”. Em outras palavras, o decodificador de áudio é configurado para detectar uma combinação de símbolos existentes, que não são normalmente utilizados para representar um valor numérico, e para interpretar a ocorrência de tal combinação de símbolos já existentes como uma condição de parada aritmética.

438. Uma realização, de acordo com a invenção, utiliza um mecanismo de *hashing* de contexto de duas tabelas.

439. Para resumir, algumas realizações, de acordo com a invenção, podem compreender um ou mais dos quarto aspectos

principais:

contexto estendido para detector tanto regiões zero quanto regiões de pequena amplitude no próximo;

contexto de *hashing*;

geração do estado de contexto: atualização adicional do estado de contexto; e

derivação de contexto: quantização específica dos valores de contexto incluindo a soma das amplitudes e limitação.

440. Para concluir, um aspecto das realizações, de acordo com a presente invenção, permanece em uma atualização adicional de contexto. As realizações, de acordo com a invenção, compreendem um conceito eficiente para a atualização do contexto, que evita os cálculos extensivos do projeto de trabalho (por exemplo, do projeto de trabalho 5). Ainda, as operações simples de mudança e operações lógicas são utilizadas em algumas realizações. A atualização simples de contexto facilita o cálculo do contexto significantemente.

441. Em algumas realizações, o contexto é independente do sinal dos valores (por exemplo, os valores espectrais decodificados). Esta independência do contexto a partir do sinal dos valores traz consigo uma complexidade reduzida do contexto variável. Este conceito tem como base a observação que uma negação do sinal no contexto não traz consigo uma degradação severa da eficiência da codificação.

442. De acordo com um aspecto da invenção, o contexto é derivado utilizando a soma de dois valores espectrais. Assim, as exigências de memória para armazenamento do contexto são significantemente reduzidas. Assim, o uso de um valor de contexto,

que representa a soma de dois valores espectrais, pode ser considerado como vantajoso em alguns casos.

443. Ainda, a limitação do contexto traz consigo uma melhoria significativa em alguns casos. Além da derivação do contexto utilizando a soma de dois valores espectrais, a matriz das entradas do contexto “q” estão limitadas ao valor máximo de “0xF” em algumas realizações, que por sua vez resulta em uma limitação das exigências de memória. Esta limitação da matriz dos valores do contexto matriz “q” traz consigo algumas vantagens.

444. Em algumas realizações, o chamado “indicador de pequeno valor” é utilizado. Ao obter o contexto variável c (que é também designado como um valor de contexto corrente numérico), um indicador é definido se os valores de algumas entradas “q[1][i-3]” a “q[1][i-1]” forem muito pequenos. Assim, o cálculo do contexto pode ser realizado com alta eficiência. Valores de contexto particularmente significativos (por exemplo, valor de contexto corrente numérico) podem ser obtidos.

445. Em algumas realizações, um mecanismo de parada aritmética é utilizado. O mecanismo “ARITH_STOP” permite uma parada eficiente da codificação aritmética ou decodificação se há somente zeros valores deixados. Assim, a eficiência da codificação pode ser melhorada a custos moderados em termos de complexidade.

446. De acordo com um aspecto da invenção, um mecanismo de contexto de *hashing* de duas tabelas é utilizado. O mapeamento do contexto é realizado utilizando um algoritmo de divisão de intervalo que avalia a tabela “ari_hash_m” em combinação com uma avaliação da tabela de consulta subsequente da tabela “ari_lookup_m”. Este algoritmo é mais eficiente que o

algoritmo WD3.

447. A seguir, alguns detalhes adicionais serão discutidos.

448. Deve ser observado aqui que as tabelas “arith_hash_m[600]” e “arith_lookup_m[600]” são duas tabelas distintas. A primeira é utilizada para mapear um único índice de contexto (por exemplo, valor de contexto numérico) no índice do modelo de probabilidade (por exemplo, valor de índice de regra de mapeamento) e a segunda é utilizada para mapear um grupo de contextos consecutivos, delimitados pelos índices de contexto em “arith_hash_m[]”, no único modelo de probabilidade.

449. Ainda deve ser observado que a tabela “arith_cf_msb[96][16]” pode ser utilizada como uma alternativa à tabela “ari_cf_m[96][17]”, embora as dimensões sejam levemente diferentes. “ari_cf_m[][]” e “ari_cf_msb[][]” podem se referir à mesma tabela, pois os 17° coeficientes do modelo de probabilidades são sempre zero. Às vezes não é levado em consideração ao contar o espaço necessário para armazenar as tabelas.

450. Para resumir o mencionado acima, algumas realizações, de acordo com a invenção, provêm uma nova codificação silenciosa proposta (codificação ou decodificação), que geram modificações no projeto de trabalho USAC MPEG (por exemplo, no projeto de trabalho USAC MPEG 5). As ditas modificações podem ser vistas nas figures anexas e também na descrição relacionada.

451. Como um marco de conclusão, deve ser observado que o prefixo “ari” e o prefixo “arith” nos nomes das variáveis, matrizes, funções, e assim por diante, são utilizados de forma permutável.

REIVINDICAÇÕES

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

um decodificador aritmético (230; 820) para prover diversos valores espectrais decodificados (232; 822) com base em uma representação aritmeticamente codificada (222; 821) dos valores espectrais constantes na informação de áudio codificada (210; 810); e

um conversor de domínio de frequência em domínio de tempo (260; 830) para prover uma representação de áudio de domínio de tempo (262; 812) utilizando os valores espectrais decodificados (232; 822), para obter a informação de áudio decodificada (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 (acod_m, value) da representação aritmeticamente codificada (821) de valores espectrais em um código de símbolo (symbol) representando um ou mais dos valores espectrais decodificados ou pelo menos uma parte de um ou mais dos valores espectrais decodificados dependendo de um estado de contexto descrito por um valor de contexto corrente numérico (c); e

em que o decodificador aritmético (230; 820) é configurado para determinar o valor de contexto corrente numérico (c) dependendo de diversos valores espectrais previamente decodificados;

em que o decodificador aritmético é configurado para obter diversos valores de sub-região de contexto (q[0][i-1], q[0][i],q[0][i+1],q[1][i-1]) que descrevem sub-regiões do contexto com base em valores espectrais previamente decodificados e para armazenar os ditos valores de sub-região de contexto;

em que o decodificador aritmético é configurado para derivar um valor de contexto corrente numérico (c) associado a um ou mais valores espectrais a serem decodificados dependendo dos valores de sub-região de contexto armazenados (q[0][i-1], q[0][i],q[0][i+1],q[1][i-1]);

em que o decodificador aritmético é configurado para computar a norma de um vetor formado por diversos valores espectrais previamente decodificados (a,b), para obter um valor de sub-região de contexto comum (q[1][i]) associado aos diversos valores espectrais previamente decodificados.

2. DECODIFICADOR DE ÁUDIO, de acordo com a reivindicação 1, caracterizado por o decodificador aritmético é configurado para somar valores absolutos de diversos valores espectrais previamente decodificados, que estão associados a bins de frequência adjacentes do conversor de domínio de frequência em domínio de tempo e a uma parte temporal comum da informação de áudio, para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados.

3. DECODIFICADOR DE ÁUDIO, de acordo com a reivindicação 1, caracterizado por o decodificador aritmético é configurado para quantizar a norma de diversos valores espectrais previamente decodificados, que estão associados a bins de

frequência adjacentes do conversor de domínio de frequência em domínio de tempo e a uma parte temporal comum da informação de áudio, para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados.

4. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 3, caracterizado por o decodificador aritmético é configurado para somar valores absolutos de diversos valores espectrais previamente decodificados (a, b), que são codificados utilizando um valor de código comum (acod_m, value), para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados.

5. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 4, caracterizado por o decodificador aritmético é configurado para prover valores espectrais decodificados sinalizados para o conversor de domínio de frequência em domínio de tempo, e para somar valores absolutos correspondentes aos valores espectrais decodificados sinalizados para obter o valor de sub-região de contexto comum associado aos diversos valores espectrais previamente decodificados.

6. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 5, caracterizado por o decodificador aritmético é configurado para derivar um valor de soma limitado a partir de uma soma de valores absolutos de valores espectrais previamente decodificados, de modo que uma faixa de possíveis valores representados pelo valor de soma limitado seja menor que uma faixa de possíveis valores de soma.

7. DECODIFICADOR DE ÁUDIO, de acordo com uma das

reivindicações 1 a 6, caracterizado por o decodificador aritmético é configurado para obter um valor de contexto corrente numérico (c) dependendo de diversos valores de sub-região de contexto (q[0][i-1], q[0],[i], q[0][i+1], q[1][i-1]) associados a diferentes conjuntos de valores espectrais previamente decodificados.

8. DECODIFICADOR DE ÁUDIO, de acordo com a reivindicação 7, caracterizado por o decodificador aritmético ser configurado para obter uma representação numérica de um valor de contexto corrente numérico (c), de modo que uma primeira parte da representação numérica do valor de contexto corrente numérico seja determinada por um primeiro valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados, e de modo que uma segunda parte da representação numérica do valor de contexto corrente numérico seja determinada por um segundo valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados.

9. DECODIFICADOR DE ÁUDIO, de acordo com a reivindicação 7 ou com a reivindicação 8, em que o decodificador aritmético é configurado para obter o valor de contexto corrente numérico (c) de modo que um primeiro valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados e um segundo valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados sejam caracterizados por compreender diferentes pesos no valor de contexto corrente numérico (c).

10. DECODIFICADOR DE ÁUDIO, de acordo com uma das

reivindicações 7 a 9, caracterizado por o decodificador aritmético ser configurado para modificar a representação numérica de um valor de contexto prévio numérico (c), que descreve um estado de contexto associado a um ou mais valores espectrais previamente decodificados, dependendo de um valor de soma ou de um valor de soma limitado (q[1][i-1]) de valores absolutos de diversos valores espectrais previamente decodificados, para obter uma representação numérica de um valor de contexto corrente numérico (c) que descreve um estado de contexto associado a um ou mais valores espectrais a serem decodificados.

11. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 10, caracterizado por o decodificador aritmético é configurado para verificar se uma soma de diversos valores de sub-região de contexto (q[1][i-3], q[1][i-2], q[1][i-1]) é menor ou igual a um valor limiar de soma predeterminado, e para seletivamente modificar o valor de contexto corrente numérico (c) dependendo de um resultado da verificação,

em que cada um dos valores de sub-região de contexto (q[1][i-3], q[1][i-2], q[1][i-1]) é um valor de soma ou um valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados associados.

12. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 11, caracterizado por o decodificador aritmético é configurado para considerar diversos valores de sub-região de contexto (q[0][i-3], q[0][i], q[0][i+1]) definidos por valores espectrais previamente decodificados associados a uma parte temporal prévia do conteúdo de áudio, e também para considerar pelo menos um valor de sub-região de contexto (q[1][i-

1]) definido por valores espectrais previamente decodificados associados a uma parte temporal corrente do conteúdo de áudio, para obter um valor de contexto corrente numérico (c) associado a um ou mais valores espectrais a serem decodificados e associado à parte temporal corrente do conteúdo de áudio,

de modo que um ambiente tanto de valores espectrais previamente decodificados temporariamente adjacentes da parte temporal prévia como os valores espectrais previamente decodificados adjacentes à frequência da parte temporal corrente sejam considerados para obter o valor de contexto corrente numérico (c).

13. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 12, caracterizado por o decodificador aritmético é configurado para armazenar um conjunto de valores de sub-região de contexto, sendo cada um dos valores de sub-região de contexto um valor de soma ou valor de soma limitado de valores absolutos de diversos valores espectrais previamente decodificados, para uma determinada parte temporal da informação de áudio, e para utilizar os valores de sub-região de contexto para derivar um valor de contexto corrente numérico (c) para decodificação de um ou mais valores espectrais de uma parte temporal da informação de áudio seguindo a determinada parte temporal da informação de áudio enquanto deixa os valores espectrais individuais previamente decodificados para a determinada parte temporal da informação de áudio não considerada ao derivar o valor de contexto corrente numérico (c).

14. DECODIFICADOR DE ÁUDIO, de acordo com uma das reivindicações 1 a 13, caracterizado por o decodificador

aritmético é configurado para decodificar separadamente um valor de magnitude e um sinal de um valor espectral, e

em que o decodificador aritmético é configurado para deixar sinais de valores espectrais previamente decodificados não considerados ao determinar o estado de contexto corrente numérico (c) para a decodificação de um valor espectral a ser decodificado.

15. 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 sendo caracterizado por compreender:

um conversor de domínio de tempo em domínio de frequência com 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 pré-processada deste, utilizando uma senha de comprimento variável (acod_m, acod_r), em que o codificador aritmético 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 a informação de áudio codificada compreende diversas senhas de comprimento variável,

em que o codificador aritmético é configurado

para selecionar uma regra de mapeamento que descreve um mapeamento de um ou mais valores espectrais, ou de um plano de bits mais significativo de um ou mais valores espectrais, em um valor de código, dependendo de um estado de contexto (s) descrito por um valor de contexto corrente numérico (c); e

em que o codificador aritmético é configurado para determinar o valor de contexto corrente numérico (c) dependendo de diversos valores espectrais previamente codificados,

em que o codificador aritmético é configurado para obter diversos valores de sub-região de contexto (q[][]) que descrevem sub-regiões do contexto com base em valores espectrais previamente codificados, para armazenar os ditos valores de sub-região de contexto, e para derivar um valor de contexto corrente numérico (c), associado a um ou mais valores espectrais a serem codificados, dependendo dos valores de sub-região de contexto armazenados,

em que o codificador aritmético é configurado para computar a norma de um vetor formado por diversos valores espectrais previamente codificados, para obter um valor de sub-região de contexto comum associado aos diversos valores espectrais previamente codificados.

16. MÉTODO PARA PROVER UMA INFORMAÇÃO DE ÁUDIO DECODIFICADA COM BASE EM UMA INFORMAÇÃO DE ÁUDIO CODIFICADA, o método sendo caracterizado por compreender:

provisão de diversos valores espectrais decodificados com base em uma representação aritmeticamente codificada dos valores espectrais constantes na informação de áudio codificada; e

provisão de uma representação de áudio de domínio de tempo utilizando os valores espectrais decodificados, para obter a informação de áudio decodificada;

em que a provisão dos diversos valores espectrais decodificados compreende a seleção de uma regra de mapeamento que descreve um mapeamento de um valor de código (acod_m; value) da representação aritmeticamente codificada (821) de valores espectrais em um código de símbolo (symbol) representando um ou mais dos valores espectrais decodificados, ou um plano de bits mais significativo de um ou mais dos valores espectrais decodificados dependendo de um estado de contexto descrito por um valor de contexto corrente numérico (c); e

em que o valor de contexto corrente numérico (c) é determinado dependendo de diversos valores espectrais previamente decodificados;

em que diversos valores de sub-região de contexto que descrevem sub-regiões do contexto são obtidos com base em valores espectrais previamente decodificados e armazenados;

em que um valor de contexto corrente numérico (c) associado a um ou mais valores espectrais a serem decodificados é derivado dependendo dos valores de sub-região de contexto armazenados; e

em que uma norma (a+b) de um vetor formado por diversos valores espectrais previamente decodificados é computada, para obter um valor de sub-região de contexto comum (q[1][i]) associado aos diversos valores espectrais previamente decodificados (a,b).

17. MÉTODO PARA PROVER UMA INFORMAÇÃO DE ÁUDIO

CODIFICADA COM BASE EM UMA INFORMAÇÃO DE ÁUDIO DE ENTRADA, o método sendo caracterizado por compreender:

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 em domínio de frequência com compactação de energia, de modo que a representação de áudio de domínio de frequência compreenda um conjunto de valores espectrais; e

codificação aritmética de um valor espectral, ou uma versão pré-processada desta, utilizando uma senha de comprimento variável, em que um valor espectral ou um valor de um plano de bits mais significativo de um valor espectral seja mapeado em um valor de código;

em que a regra de mapeamento que descreve um mapeamento de um ou mais valores espectrais, ou de um plano de bits mais significativo de um ou mais valores espectrais, em um valor de código é selecionada dependendo de um estado de contexto descrito por um valor de contexto corrente numérico (c);

em que um valor de contexto corrente numérico (c) é determinado dependendo de diversos valores espectrais adjacentes previamente codificados;

em que diversos valores de sub-região de contexto que descrevem sub-regiões do contexto são obtidos com base em valores espectrais previamente codificados, em que um valor de contexto corrente numérico (c) associado a um ou mais valores espectrais a serem codificados é derivado dependendo dos valores de sub-região de contexto armazenados (q[0][i-1], q[0][i], q[0][i+1], q[1][i-1]); e

em que uma norma de um vetor formado por diversos valores espectrais previamente codificados é computada para obter um valor de sub-região de contexto comum (q[1][i]) associado aos diversos valores espectrais previamente codificados;

em que a informação de áudio codificada compreende diversas senhas de comprimento variável.

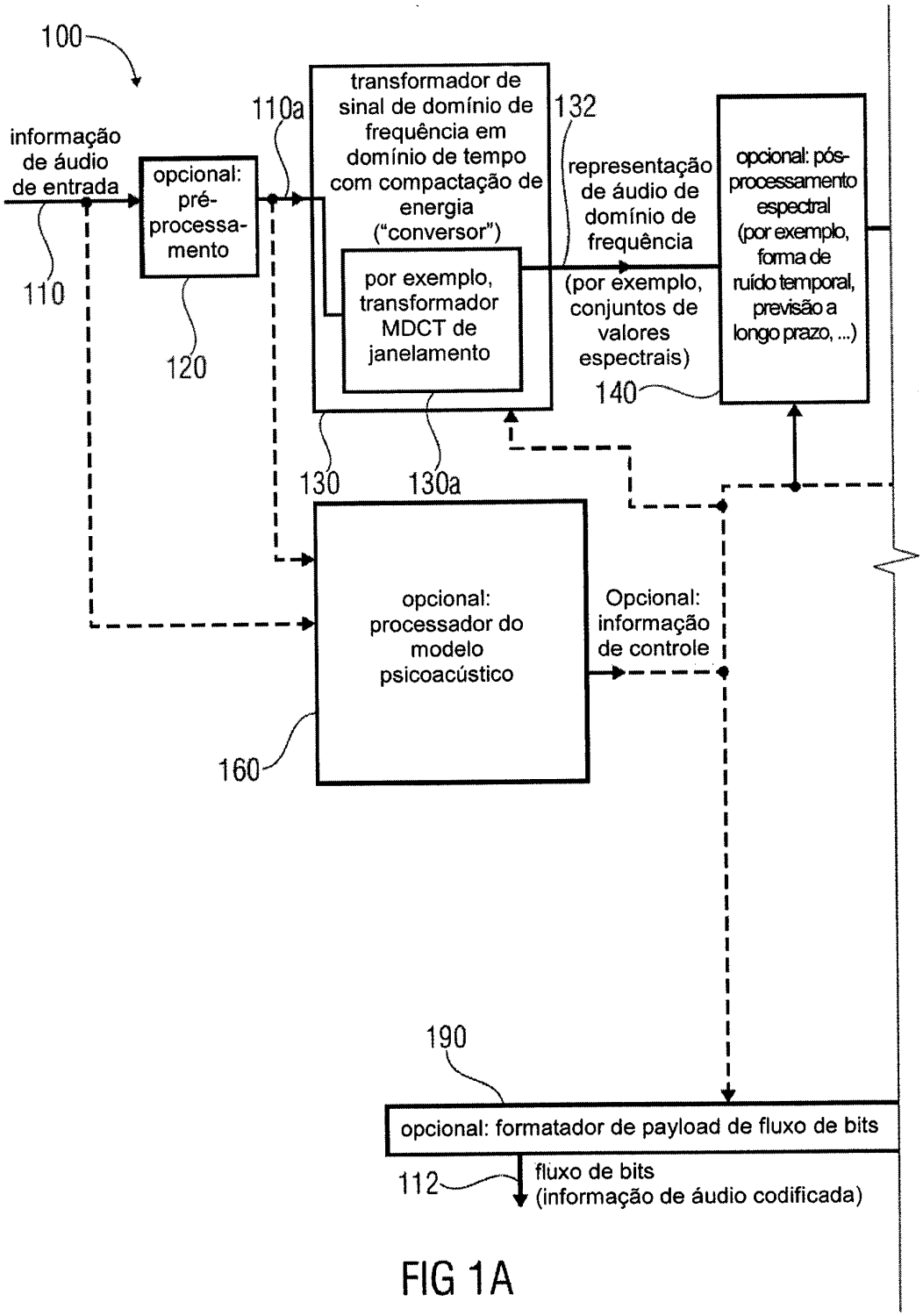
100
informação de áudio de entrada
110
opcional: pré-processamento
120
110a
transformador de sinal de domínio de frequência em domínio de tempo com compactação de energia ("conversor")
por exemplo, transformador MDCT de janelamento
130
130a
132
representação de áudio de domínio de frequência
(por exemplo, conjuntos de valores espectrais)
140
opcional: pós-processamento espectral (por exemplo, forma de ruído temporal, previsão a longo prazo, ...)
opcional: processador do modelo psicoacústico
160
Opcional: informação de controle
190
opcional: formatador de payload de fluxo de bits
112
fluxo de bits (informação de áudio codificada)

FIG 1A

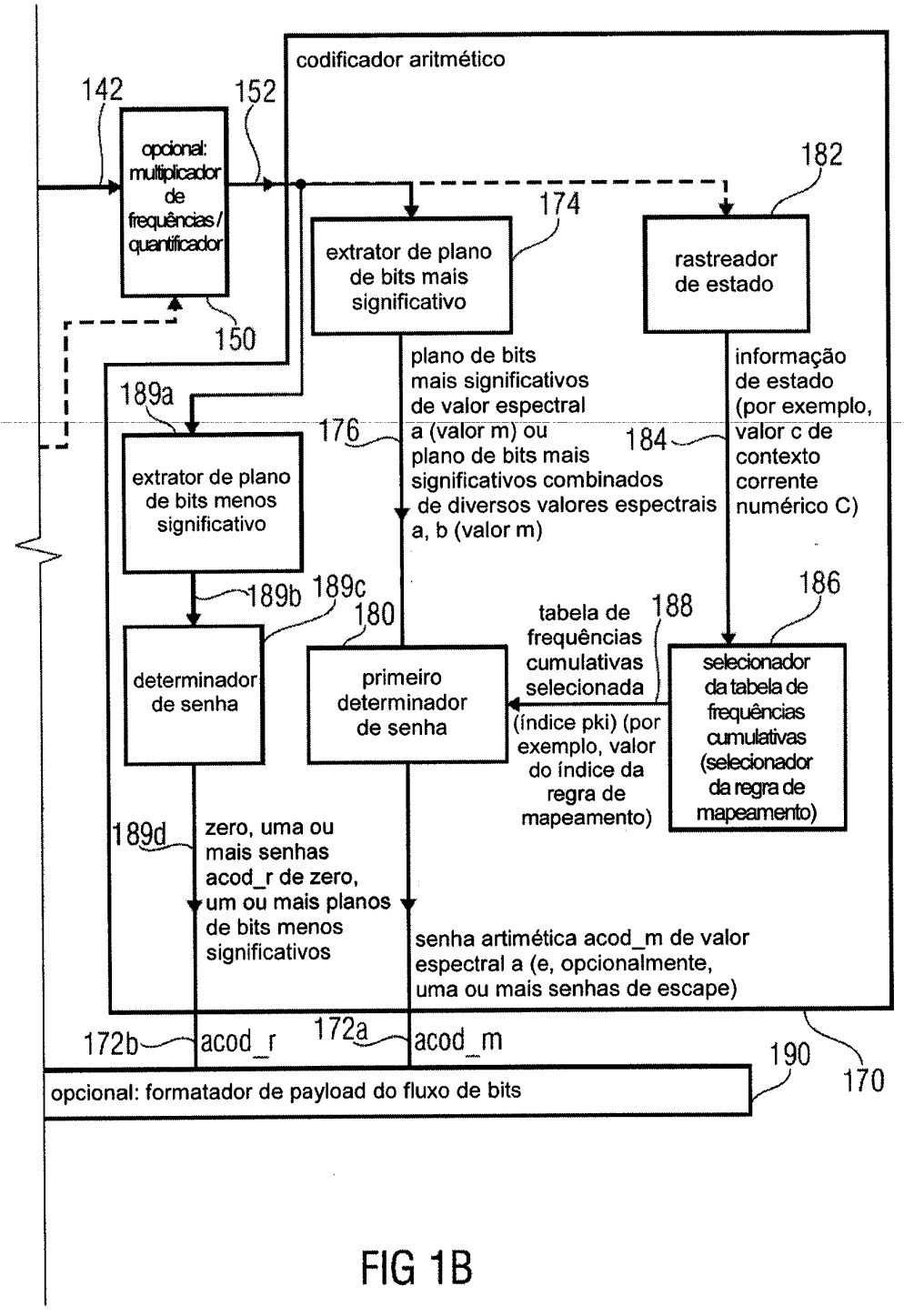
codificador aritmético
142
152
opcional: multiplicador de frequências / quantificador
150
182
174
extrator de plano de bits mais significativo
rastreador de estado
plano de bits mais significativos de valor espectral a (valor m) ou plano de bits mais significativos combinados de diversos valores espectrais a, b (valor m)
176
184
informação de estado (por exemplo, valor c de contexto corrente numérico C)
189a
extrator de plano de bits menos significativo
189b
189c
180
186
188
determinador de senha
primeiro determinador de senha
tabela de frequências cumulativas selecionada (índice pki) (por exemplo, valor do índice da regra de mapeamento)
selecionador da tabela de frequências cumulativas (selecionador da regra de mapeamento)
189d
zero, uma ou mais senhas acod_r de zero, um ou mais planos de bits menos significativos
senha artimética acod_m de valor espectral a (e, opcionalmente, uma ou mais senhas de escape)
172b
acod_r
172a
acod_m
190
170
opcional: formatador de payload do fluxo de bits

FIG 1B

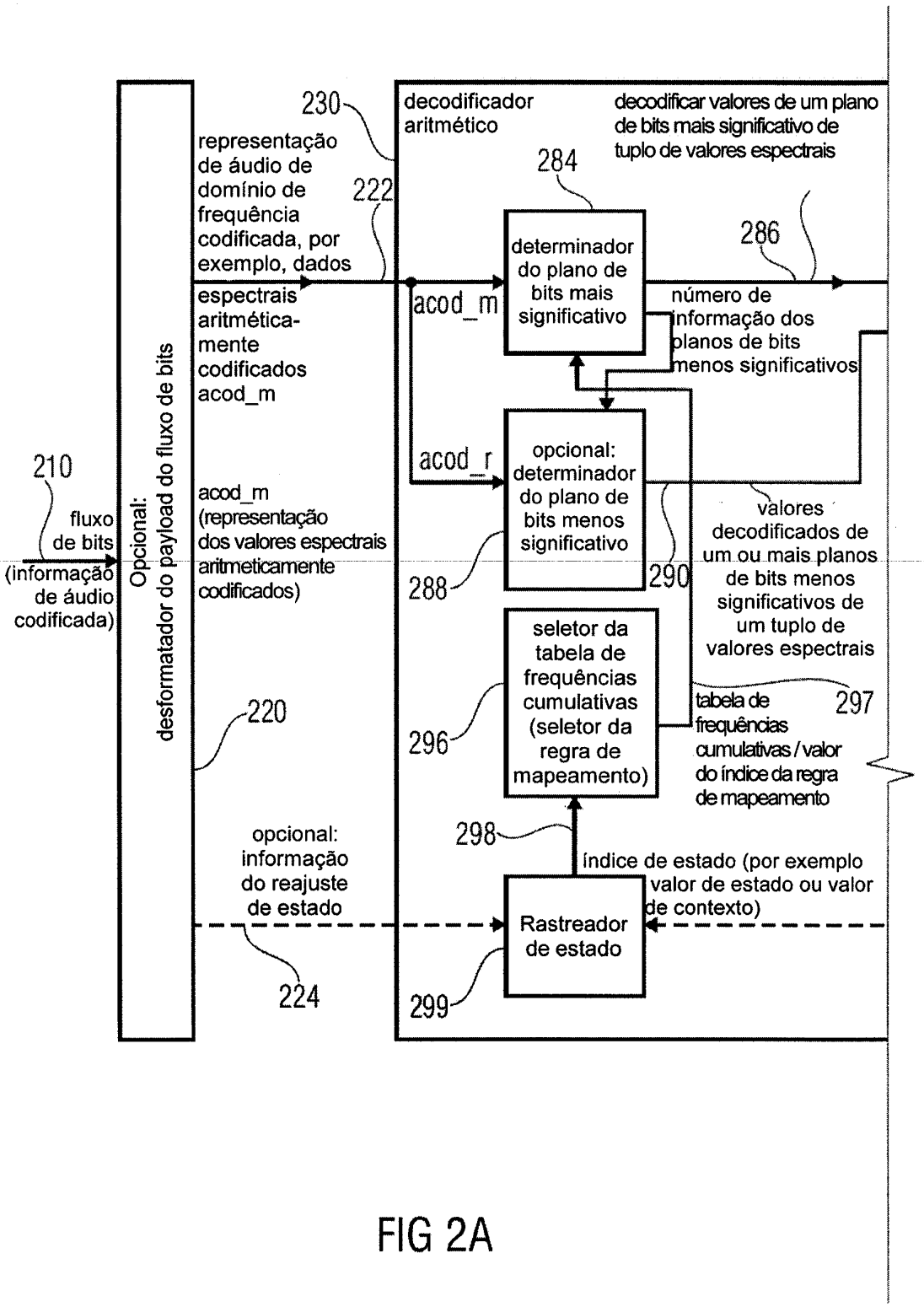
230
decodificador aritmético
decodificar valores de um plano de bits mais significativo de tuplo de valores espectrais
284
222
representação de áudio de domínio de frequência codificada, por exemplo, dados espectrais aritmética-mente codificados acod_m
determinador do plano de bits mais significativo
286
acod_m
número de informação dos planos de bits menos significativos
acod_r
opcional: determinador do plano de bits menos significativo
210
fluxo de bits (informação de áudio codificada)
Opcional: desformatador do payload do fluxo de bits
acod_m (representação dos valores espectrais aritmeticamente codificados)
288
290
valores decodificados de um ou mais planos de bits menos significativos de um tuplo de valores espectrais
seletor da tabela de frequências cumulativas (seletor da regra de mapeamento)
296
297
tabela de frequências cumulativas / valor do índice da regra de mapeamento
220
298
índice de estado (por exemplo valor de estado ou valor de contexto)
opcional: informação do reajuste de estado
Rastreador de estado
224
299

FIG 2A

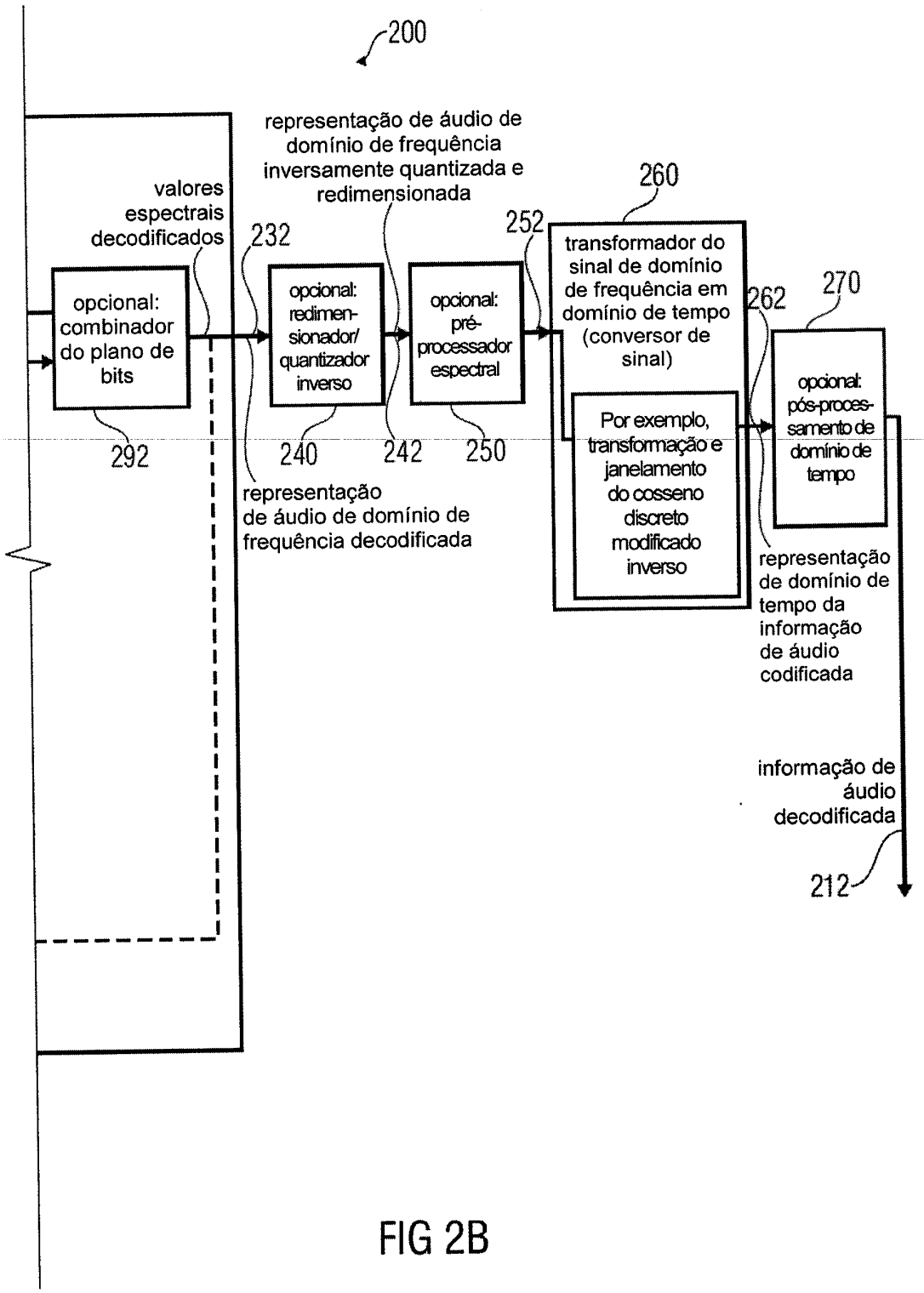
200
valores espectrais decodificados
opcional: combinador do plano de bits
292
representação de áudio de domínio de frequência inversamente quantizada e redimensionada
232
opcional: redimen-sionador/ quantizador inverso
240
representação de áudio de domínio de frequência decodificada
242
opcional: pré-processador espectral
250
252
260
transformador do sinal de domínio de frequência em domínio de tempo (conversor de sinal)
Por exemplo, transformação e janelamento do cosseno discreto modificado inverso
262
270
opcional: pós-proces-samento de domínio de tempo
representação de domínio de tempo da informação de áudio codificada
informação de áudio decodificada
212

FIG 2B

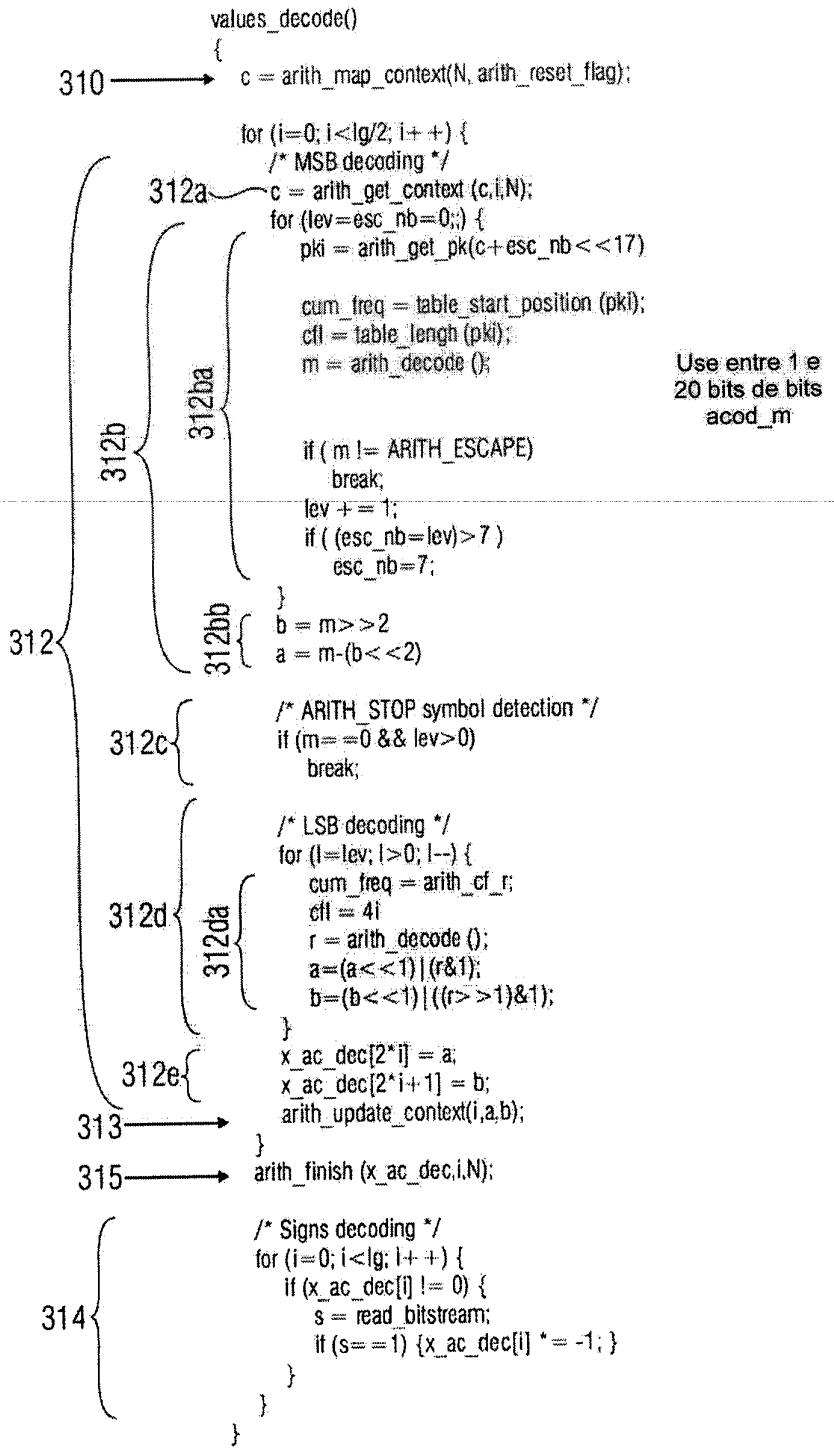
values_decode()
{
310
c = arith_map_context(N, arith_reset_flag);
for (i=0; i<lg/2; i++) {
/* MSB decoding */
312a
c = arith_get_context (c,i,N);
for (lev=esc_nb=0;;) {
312b
312ba
pki = arith_get_pk(c+esc_nb<<17)
cum_freq = table_start_position (pki);
cfl = table_lengh (pki);
m = arith_decode ();
Use entre 1 e 20 bits de bits acod_m
if (m != ARITH_ESCAPE)
break;
lev += 1;
if ((esc_nb=lev)>7)
esc_nb=7;
}
312bb
b = m>>2
a = m-(b<<2)
312
312c
/* ARITH_STOP symbol detection */
if (m==0 && lev>0)
break;
312d
/* LSB decoding */
for (l=lev; l>0; l--) {
312da
cum_freq = arith_cf_r;
cfl = 4i
r = arith_decode ();
a=(a<<1)|(r&1);
b=(b<<1)|((r>>1)&1);
}
312e
x_ac_dec[2*i] = a;
x_ac_dec[2*i+1] = b;
313
arith_update_context(i,a,b);
}
315
arith_finish (x_ac_dec,i,N);
/* Signs decoding */
314
for (i=0; i<lg; i++) {
if (x_ac_dec[i] != 0) {
s = read_bitstream;
if (s==1) {x_ac_dec[i] *= -1; }
}
}
}

FIG 3

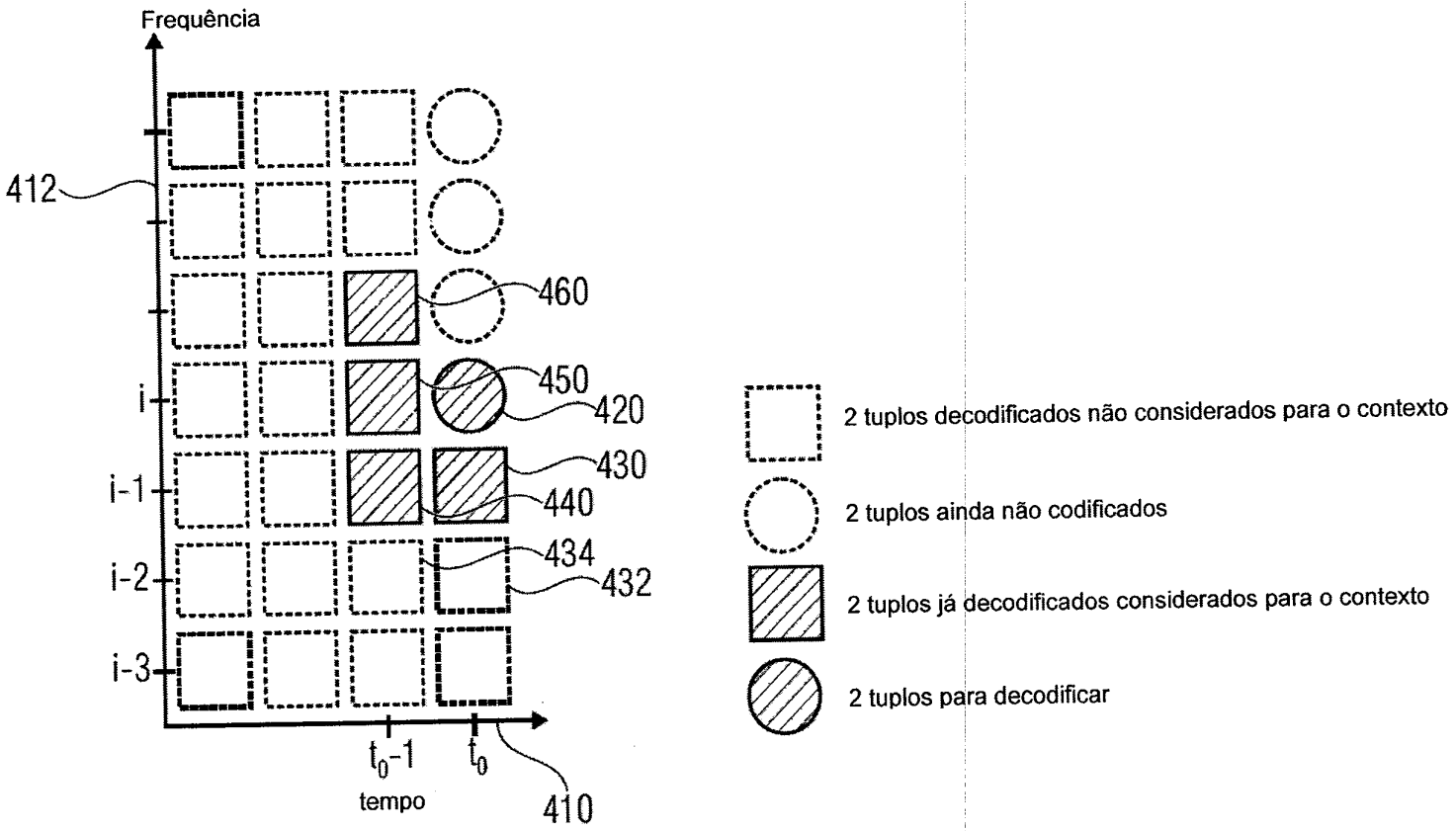
Frequência
412
460
450
420
i
430
i-1
440
434
i-2
432
i-3
t_0-1
t_0
tempo
410
2 tuplos decodificados não considerados para o contexto
2 tuplos ainda não codificados
2 tuplos já decodificados considerados para o contexto
2 tuplos para decodificar

FIG 4

```
/*Input variables*/
N /*Length of the current window*/
arith_reset_flag /*Arithmetic coder reset flag*/

/*Global variables*/
previous_N /*Length of the previous window */

c = arith_map_context(N,arith_reset_flag)
{
    if (arith_reset_flag) {
        for (j=0; j<N/4; j++) {
            q[0][j]=0;
        }
    } else {
        ratio = ((float)previous_N) / ((float)N);
        for (j=0; j<N/4; j++) {
            k = (int) ((float) j * ratio);
            q[0][j] = q[1][k];
        }
    }

    previous_N=N;

    return(q[0][0]<<12);
}
```

500a

500b

FIG 5A

```
/*Input variables*/
lg /*Number of sepctral coefficients to decode in the frame*/
arith_reset_flag /*Arithmetic coder reset flag*/
/*Global variables*/
previous_lg /*Previous number of spectral lines of the previous frame*/

c=arith_map_context (lg,arith_reset_flag)
{
   v=w=0

   if(arith_reset_flag){
      for(j=0; j<lg/2; j++){
         q[0][v++]=0;
      }
   else{
      ratio= ((float)previous_lg)/((float)lg);
      for(j=0; j<lg/2; j++){
         k = (int) ((float)) ((j)*ratio);
         q[0][v++] = qs[w+k];
      }
   }

   previous_lg=lg;

   return(q[0][0]<<12);
}
```

FIG 5B

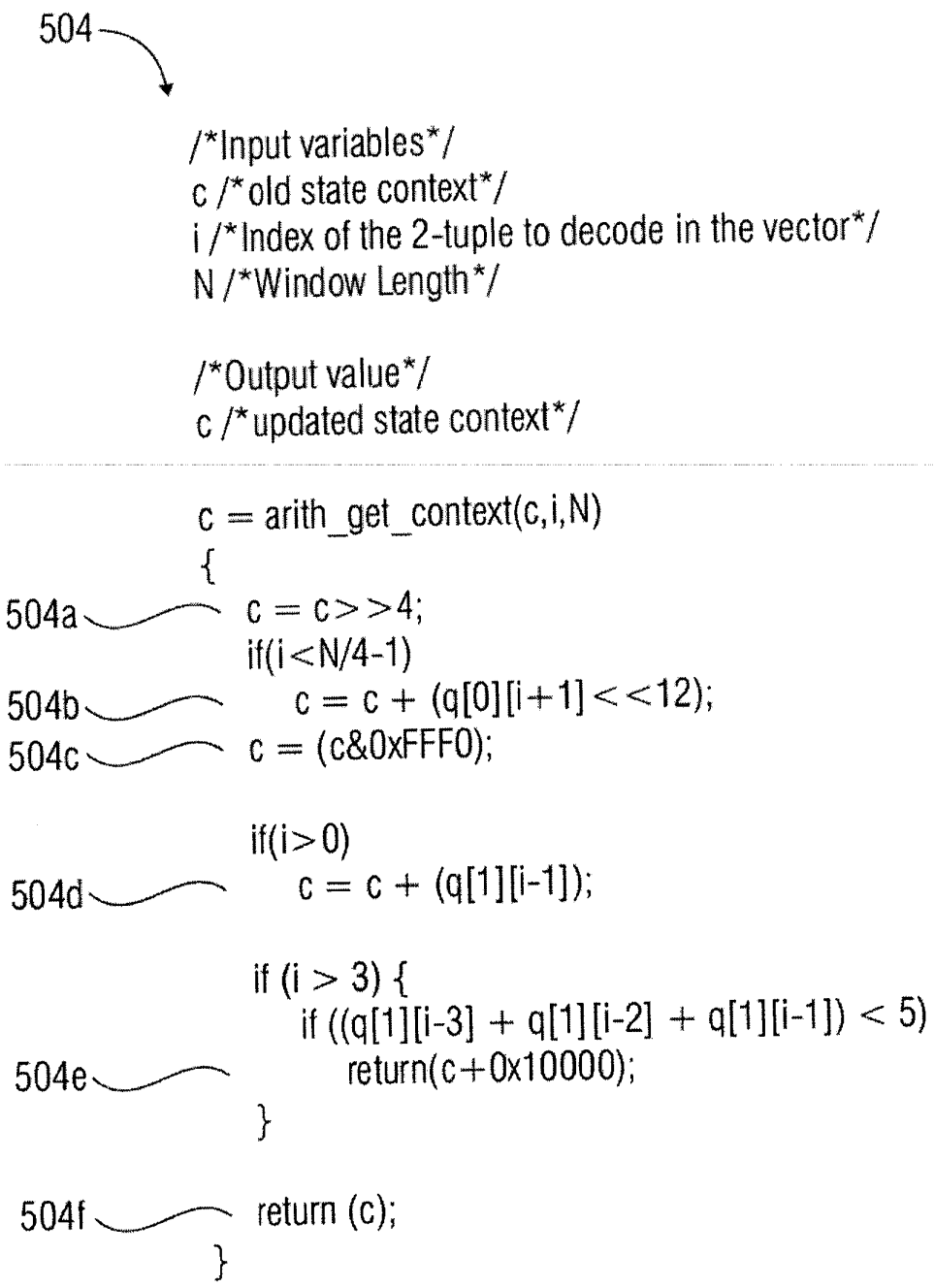
504
/*Input variables*/
c /*old state context*/
i /*Index of the 2-tuple to decode in the vector*/
N /*Window Length*/
/*Output value*/
c /*updated state context*/
c = arith_get_context(c,i,N)
{
504a
c = c>>4;
if(i<N/4-1)
504b
c = c + (q[0][i+1]<<12);
504c
c = (c&0xFFF0);
if(i>0)
504d
c = c + (q[1][i-1]);
if (i > 3) {
if ((q[1][i-3] + q[1][i-2] + q[1][i-1]) < 5)
504e
return(c+0x10000);
}
504f
return (c);
}

FIG 5C

```
/*Input variables*/
c /*old state context*/
i /*Index of the 2-tuple to decode in the vector*/
/*Output value*/
c /*updated state context*/


c=arith_get_context(c,i)
{
    c=c>>4;
    c=(c)+(q[0][i+1]<<12);
    c=(c&0xFFF0)+(q[1][i-1]);

    if(i > 3) {
        if((q[1][i-3] + q[1][i-2] + q[1][i-1]) < 5)
            return(c+0x10000);
    }

    return(c);
}
```

FIG 5D

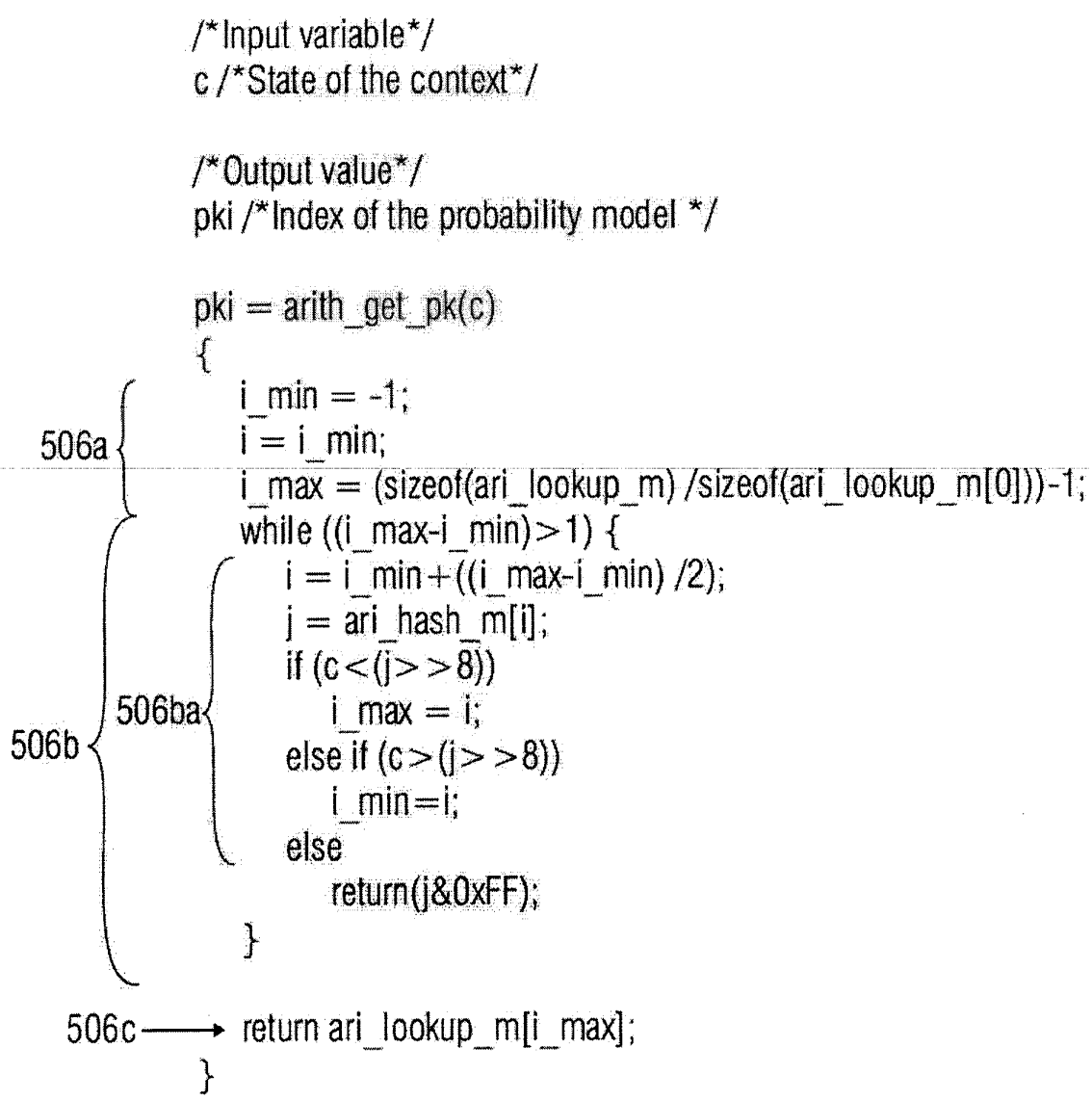
/*Input variable*/
c /*State of the context*/
/*Output value*/
pki /*Index of the probability model */
pki = arith_get_pk(c)
{
i_min = -1;
i = i_min;
i_max = (sizeof(ari_lookup_m) /sizeof(ari_lookup_m[0]))-1;
while ((i_max-i_min)>1) {
i = i_min+((i_max-i_min) /2);
j = ari_hash_m[i];
if (c<(j>>8))
i_max = i;
else if (c>(j>>8))
i_min=i;
else
return(j&0xFF);
}
return ari_lookup_m[i_max];
}
506a
506b
506ba
506c

FIG 5E

```
/*Input variable*/
c /*State of the context*/
/*Output value*/
pki /*Index of the probability model */
/*constants*/
i_diff[]={ 299, 149, 74, 37, 18, 9, 4, 2, 1};

pki=arith_get_pk(c) {
    i_min=0;
    s=c<<8;
    for(k=0;k<9;k++) {
        i=i_min+i_diff[k];
        j=ari_hash_m[i];
        if(s>j) {
            i_min=i+1;
        }
    }

    j=ari_hash_m[i_min];
    if(s>j)
        return(ari_lookup_m[i_min+1]);
    else if(c<(j>>8))
        return(ari_lookup_m[i_min]);
    else
        return(j&0xFF);
}
```

508a, 508b, 508ba, 508c

FIG 5F

```
/*helper functions*/
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[] */

symbol = arith_decode(cum_freq, cfl)
{
   if (arith_first_symbol()) {
      value = 0;
      for (i=1; i<=16; i++) {
         value = (val<<1) | arith_get_next_bit();
      }
      low  = 0;
      high  = 65535;
   }

   range = high-low+1;
   cum =((((int) (value-low+1))<<14)-((int) 1))/range;
   p = cum_freq-1;
```

570a

570b

FIG 5G(1)

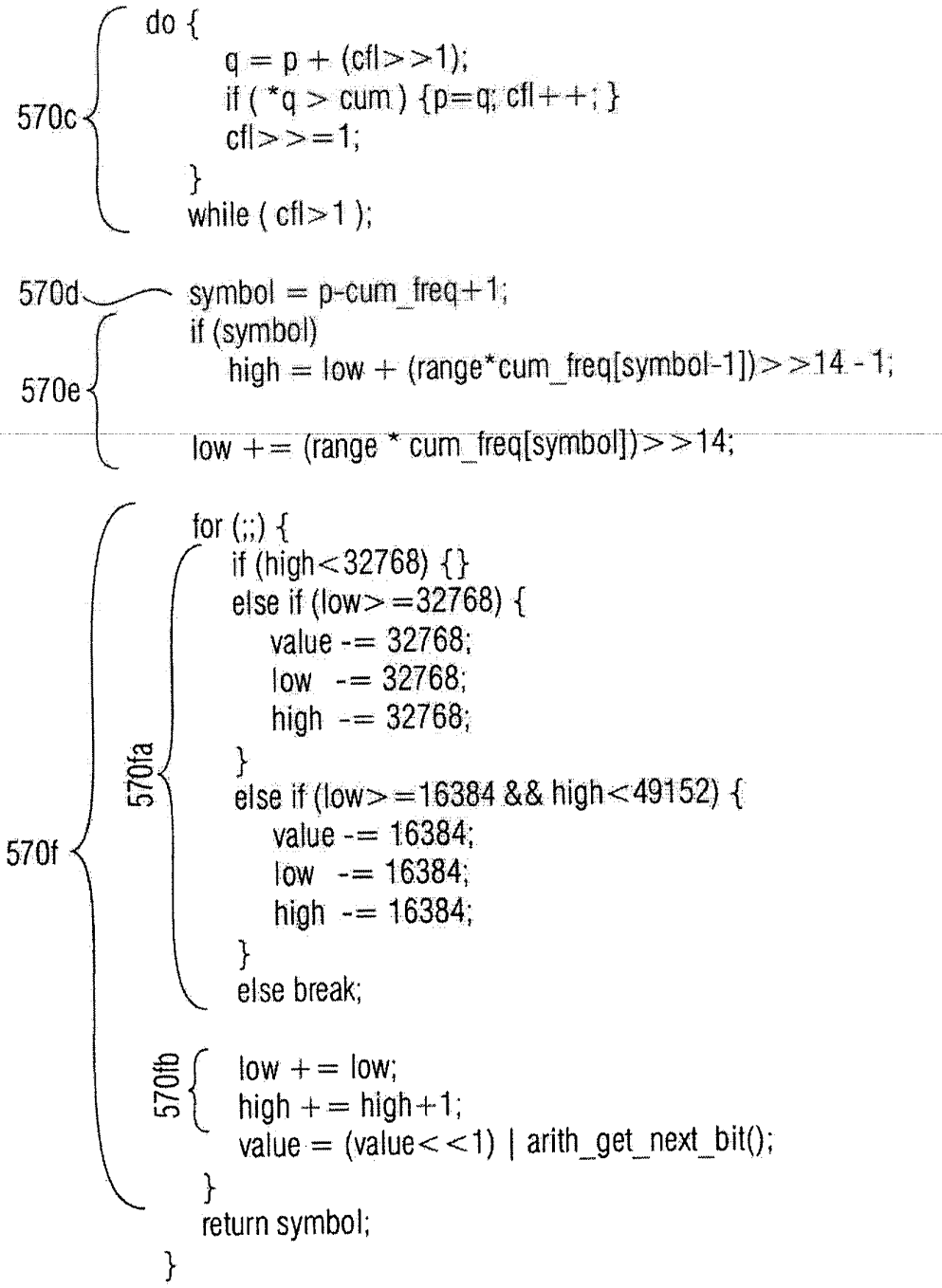
570c
do {
q = p + (cfl>>1);
if (*q > cum) {p=q; cfl++; }
cfl>>=1;
}
while (cfl>1);
570d
symbol = p-cum_freq+1;
570e
if (symbol)
high = low + (range*cum_freq[symbol-1])>>14 - 1;
low += (range * cum_freq[symbol])>>14;
570f
for (;;) {
570fa
if (high<32768) {}
else if (low>=32768) {
value -= 32768;
low -= 32768;
high -= 32768;
}
else if (low>=16384 && high<49152) {
value -= 16384;
low -= 16384;
high -= 16384;
}
else break;
570fb
low += low;
high += high+1;
value = (value<<1) | arith_get_next_bit();
}
return symbol;
}

FIG 5G(2)

```
/*helper functions*/
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[] */

symbol = arith_decode(cum_freq, cfl)
{
   if (arith_first_symbol()) {
      value = 0;
      for (i=1; i<=16; i++) {
         value = (val<<1) | arith_get_next_bit();
      }
      low  = 0;
      high = 65535;
   }

   range = high-low+1;
   cum =((((int) (value-low+1))<<14)-((int) 1));
   p = cum_freq-1;

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

<CONTINUA NA FIGURA 5I>

FIG 5H

<CONTINUAÇÃO DA FIGURA 5H>

```
    while ( cfl>1 );

    symbol = p-cum_freq+1;
    if (symbol)
        high = low + (range*cum_freq[symbol-1])>>14 - 1;

    low += (range * cum_freq[symbol])>>14;

    for (;;) {
        if (high<32768) {}
        else if (low>=32768) {
            value -= 32768;
            low  -= 32768;
            high  -= 32768;
        }
        else if (low>=16384 && high<49152) {
            value -= 16384;
            low  -= 16384;
            high  -= 16384;
        }
        else break;

        low += low;
        high += high+1;
        value = (value<<1) | arith_get_next_bit();
    }
    return symbol;
}
```

FIG 5I

```
b = m>>2;
a = m-(b<<2);
for (j=0;j<lev;j++) {
    r = arith_decode(arith_cf_r,4);
    a = (a<<1) | (r&1);
    b = (b<<1) | ((r>>1)&1);
}
```

FIG 5J

```
x_ac_dec[2*i]   = a
x_ac_dec[2*i+1] = b;
```

FIG 5K

```
/*input variables*/
a,b /* Decoded unsigned quantized spectralcoefficients of the 2-tuple */
i /* Index of the quantized spectral coefficient to decode */

arith_update_context(i, a, b)
{
    q[1][i] = a+b+1;
    if (q[1][i]>0xF)
        q[1][i] = 0xF;
}
```

FIG 5L

```
/*input variables*/
offset /*number of decoded 2-tuple */
N /*Window length */
x_ac_dec /*vector of decoded spectal coefficients*/

arith_finish(x_ac_dec,offset,N)
{
   for(i=offset ;i<N/4;i++) {
      x_ac_dec[2*i] = 0;
      x_ac_dec[2*i+1] = 0;
      q[1][i] = 1;
   }
}
```

FIG 5M

```
b= m>>2
a = m&0x03;
for(j=0;j<lev;j++){
   r = arith_decode(arith_cf_r,4);
   a = (a<<1) | (r&1);
   b = (b<<1) |((r>>1)&1);

}
```

FIG 5N

```
/*input variables*/
a,b /*Decoded unsigned quantized spectralcoefficients of the 2-tuple*/
i /*Index of the quantized spectral coefficient to decode*/

arith_update_context (){
   qdec[2*i]=a
   qdec[2*i+1]=b;
   q[1][i]=a+b+1;

   if(q[1][i]>0xF)
      q[1][i]=0xF;
}
```

FIG 5O

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

arith_save_context(i,lg){

    for(;i<N/4;i++){
        qdec[2*i]=0;
        qdec[2*i+1]=0;
        q[1][i]=1;
    }

    if(core_mode==1){
        ratio= ((float) lg)/((float)1024);
        for(j=0; j<512; j++){
            k = (int) ((float) j*ratio);
            qs[j] = q[1][k];
        }
        previous_lg = 512;
    }
    else{
        for(j=0; j<512; j++){
            qs[j] = q[1][j];
        }
        previous_lg = MIN(1024,lg);
    }
}
```

FIG 5P

Definições

a,b	2 tuplos para decodificar (coeficiente quantizado de 2 tuplos para decodificar)
m	o plano de 2 bits mais significativo do coeficiente espectral quantizado para decodificar.
r	os planos de bits menos significativos do coeficiente espectral quantizado para decodificar.
lev	nível dos planos de bits remanescentes. Este corresponde ao número de planos de bits menos significativos.
arith_hash_m[]	estados de contexto do mapeamento da tabela *hash* em um índice da tabela de frequências cumulativas pki.
arith_lookup_m[]	grupo de mapeamento da tabela de visualização dos estados de contexto em um índice da tabela de frequências cumulativas pki.
arith_cf_m[pki][17]	modelos das frequências cumulativas para o plano de 2 bits mais significativos m e o símbolo ARITH_ESCAPE.
arith_cf_r [lsbidx][]	frequências cumulativas para o símbolo dos planos de bits menos significativos r.
arith_cf_r []	frequências cumulativas para o símbolo dos planos de bits menos significativos r.
q[2][]	elementos de contexto de 2 tuplos da estrutura prévia e atual.
x_ace_dec[]	os coeficientes espectrais quantizados decodificados.
arith_reset_flag	flag que indica se o contexto espectral silencioso deve ser redefinido.
ARITH_STOP	símbolo de parada consistindo na sucessão do símbolo ARITH_ESCAPE e m = 0. Quando isto ocorrer, o restante da estrutura é decodificado com valores zero.
N	comprimento da janela. Para o modo FD é deduzido de window_sequence e para TCX N=2*LG.
previous_N	comprimento da janela anterior.

FIG 5Q

Definições

a,b	O coeficiente quantizado de 2 tuplos para decodificar
m	o plano de 2 bits mais significativo do coeficiente espectral quantizado para decodificar.
r	o plano de 2 bits mais significativo do coeficiente espectral quantizado para decodificar.
lev	nível dos planos de bits remanescentes. Este corresponde ao número que os planos de bits menos significativos do que o plano de 2 bit a bit mais significativos.
arith_hash_m[]	estados de contexto do mapeamento da tabela *hash* em um índice da tabela de frequências cumulativas pki.
arith_lookup_m[]	grupo de mapeamento da tabela de visualização dos estados de contexto em um índice da tabela de frequências cumulativas pki.
arith_cf_m[pki][17]	modelos das frequências cumulativas para o plano de 2 bits mais significativos m e o símbolo ARITH_ESCAPE.
arith_cf_r []	frequências cumulativas para o símbolo dos planos de bits menos significativos r.
previous_lg	número de coeficientes espectrais transmitidos previamente decodificados pelo decodificador aritmético.
q[2][]	os usos de contexto corrente de 2 tuplos para decodificar a estrutura atual.
qs[]	o contexto passado armazenado para a próxima estrutura.
qdec[]	os coeficientes espectrais quantizados decodificados.
arith_reset_flag	flag que indica se o contexto espectral silencioso deve ser redefinido.
ARITH_STOP	símbolo de parada consistindo na sucessão do símbolo ARITH_ESCAPE e m = 0. Quando isto ocorrer, o restante da estrutura é decodificado com valores zero.
N	comprimento da janela. Para AAC é deduzido de window_sequence (ver seção 6.8.3.1) e para TCX N = 2.l.g..

FIG 5R

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

FIG 6A

Sintaxe de single_channel_element()

Sintaxe	Nº de bits	Mnemônica
single_channel_element()		
{		
core_mode	**1**	**uimsbf**
if (core_mode == 1) {		
lpd_channel_stream();		
}		
else {		
fd_channel_stream();		
}		
}		

FIG 6B

Sintaxe	Nº de bits	Mnemônica
channel_pair_element()		
{		
core_mode0	1	**uimsbf**
core_mode1	1	**uimsbf**
ics_info();	Opcional: ics_info comum para dois canais	
if (core_mode0 == 1) {		
lpd_channel_stream();		
}		
else {		
fd_channel_stream();		
}		
if (core_mode1 == 1) {		
lpd_channel_stream();		
}		
else {		
fd_channel_stream();		
}		
}		

FIG 6C

Sintaxe	Nº de bits	Mnemônica	
ics_info()			
{			
window_length;	1	uimsbf	
if(window_length !=0) {			
transform_length;	1	uimsbf	
}			
else {			
transform_length=0;			
}			
window_shape;	1	uimsbf	
if (window_length !=0 && transform_length !=0){			Opcional
max_sfb;	4	uimsbf	
scale_factor_grouping;	7	uimsbf	
}			
else {			
max_sfb;	6	uimsbf	
}			
}			

FIG 6D

Sintaxe	Nº de bits	Mnemônica
fd_channel_stream() {		
global_gain;	**8**	**uimsbf**
ics_info(); (a menos que incluído no elemento de par de canal)		
scale_factor_data ();		
ac_spectral_data ();		
}		

FIG 6E

Sintaxe	Nº de bits	Mnemônica
ac_spectral_data() {		
arith_reset_flag	**1**	**uimsbf**

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

}		

FIG 6F

Sintaxe	Nº de bits	Mnemônica
arith_data(lg, arith_reset_flag)		
{		
c = arith_map_context(N, arith_reset_flag);		
for (i=0; i<lg/2; i++) {		
/* MSB decoding */		
c = arith_get_context (c,i,N);		
for (lev=esc_nb=0;;) {		
662 pki = arith_get_pk(c+esc_nb<<17)		
663 **acod_m**[pki][m]	1..20	vlclbf
if (m != ARITH_ESCAPE)		
break;		
664 lev += 1;		
if ((esc_nb=lev)>7)		
esc_nb=7;		
}		
b = m>>2;		
a = m – (b<<2);		
/* ARITH_STOP symbol detection */		
if (m==0 && lev>0)		
break;		
/* LSB decoding */		
for (l=lev; l>0; l--) {		
acod_r[r]	1..20	vlclbf
a=(a<<1)|(r&1);		
b=(b<<1)|((r>>1)&1);		
}		
x_ac_dec[2*i] = a;		
x_ac_dec[2*i+1] = b;		
668 arith_update_context(i,a,b);		
}		
arith_finish (x_ac_dec,lg,N);		
/* Signs decoding */		
for (i=0; i<lg; i++) {		
if (x_ac_dec[i] != 0) {		
s;	1	uimsbf
if (s==1) {x_ac_dec[i] *= -1; }		
}		
}		
}		

FIG 6G

Sintaxe	Nº de bits	Mnemônica
Arith_data(lg, arith_reset_flag){		
c=arith_map_context(lg, arith_reset_flag);		
for (i=0; i<lg/2; i++) {		
/*MSBs decoding*/		
c = arith_get_context (c,i);		
for (lev=esc_nb=0;;) {		
pki = arith_get_pk(c+esc_nb<<17)		
acod_m[pki][m]	1..20	vlclbf
if (m != ARITH_ESCAPE)		
break;		
lev += 1;		
if((esc_nb=lev)>7)		
esc_nb=7;		
}		
b=m>>2;		
a=m-(b<<2);		
/*ARITH_STOP symbol detection*/		
if(m==0 && lev>0)		
break;		
/*LSBs decoding*/		
for (l=lev; l>0; l--) {		
acod_r[r]	1..20	vlclbf
a=(a<<1)|(r&1);		
b=(b<<1)|((r>>1)&1);		
}		
arith_update_context(a,b,i);		
}		
arith_save_arith (l,lg);		
/*Signs decoding*/		
for (i=0; i<lg/2; i++) {		
if(a!=0){		
s;	1	uimsbf
If(s) a=-a;		
}		
if(b!=0){		
s;	1	uimsbf
if(s) b=-b;		
}		
}		
}		

FIG 6H

Definições

arith_data()	elemento de dados para decodificar os dados do codificador silencioso espectral.
arith_reset_flag	*flag* que indica se o contexto silencioso espectral deve ser redefinido.
acod _m[pki][m]	senha aritmética necessária para decodificação do plano de 2 bits mais significativos m dos coeficientes espectrais quantizados de um tuplo duplo.
acod_r[lsbidx][]	senha aritmética necessária para decodificação dos planos de bits residuais r do coeficiente espectral quantizado de um tuplo duplo.
s	o sinal codificado do coeficiente quantizado espectral não-nulo.
Elementos de	ajuda
a,b	2 tuplos correspondentes aos coeficientes espectrais quantizados.
m	o plano de 2 bits mais significativo de 2 tuplos para decodificar.
r	o plano de bit a bit menos significativo de 2 tuplos para decodificar.
lg	número de coeficientes quantizados para decodificar.
N	comprimento da janela. Para o modo FD é deduzido da window_sequence e para TCX N=2*lg.
i	índice de 2 tuplos para decodificar dentro da estrutura.
pki	índice da tabela de frequências cumulativas utilizadas pelo decodificador aritmético para decodificar m.
arith_get_pk ()	função que retorna o índice pki da tabela de frequências cumulativas necessária para decodificar a senha **acod_m[pki][m]**.
c	estado de contexto.
lsbidx	índice das tabelas de frequências cumulativas utilizadas pelo codificador aritmético para decodificar r.
lev	nível de planos de bits para decodificar além do plano de 2 bits mais significativo.
ARITH_ESCAPE	símbolo de escape que indica planos de bits adicionais além dos dois planos de bits mais significativos.
esc_nb	número do símbolo ARITH_ESCAPE já decodificado para os 2 tuplos presentes. O valor é vinculado a 7.
x_ac_dec[]	elemento que mantém os coeficientes espectrais decodificados.
arith_map_context()	inicializa os contextos necessários para decodificar a presente estrutura.
arith_get_context()	calcula o estado de contexto para decodificar os símbolos presentes de 2 tuplos m.
arith_update_context()	Atualiza o contexto para os próximos 2 tuplos.
arith_finish ()	finaliza a decodificação silenciosa.

FIG 6I

Definições	
arith_data()	elemento de dados para decodificar os dados do codificador silencioso espectral.
arith_reset_flag	*flag* que indica se o contexto silencioso espectral deve ser redefinido.
acod _m[pki][m]	senha aritmética necessária para decodificação do plano de 2 bits mais significativos m dos coeficientes espectrais quantizados de um tuplo duplo.
arith_r[]	senha aritmética necessária para decodificação dos planos de bits residuais r do coeficiente espectral quantizado de um tuplo duplo.
s	o sinal codificado do coeficiente quantizado espectral não-nulo.
Elementos de	ajuda
a,b	Os coeficientes espectrais quantizados de 2 tuplos para decodificar.
m	o plano de 2 bits mais significativo de 2 tuplos para decodificar.
r	o plano de bit a bit menos significativo de 2 tuplos para decodificar.
lg	número de coeficientes quantizados para decodificar.
i	índice de 2 tuplos para decodificar dentro da estrutura.
pki	índice da tabela de frequências cumulativas utilizadas pelo decodificador aritmético para decodificar m.
arith_get_pk ()	função que retorna o índice pki da tabela de frequências cumulativas necessária para decodificar a senha acod_m[pki][m].
c	estado de contexto
lev	nível de planos de bits para decodificar além do plano de 2 bits mais significativo.
ARITH_ESCAPE	símbolo de escape que indica planos de bits adicionais para decodificar além dos dois planos de bits mais significativos.
esc_nb	número do símbolo ARITH_ESCAPE já decodificado para os 2 tuplos presentes. O valor é vinculado a 7.
arith_map_context()	inicializa os contextos necessários para decodificar a presente estrutura.
arith_get_context()	calcula o estado de contexto para decodificar os símbolos presentes de 2 tuplos m.
arith_update_context()	atualiza o contexto para os próximos 2 tuplos.
arith_save_context()	salva o contexto para a próxima estrutura para decodificar.

FIG 6J

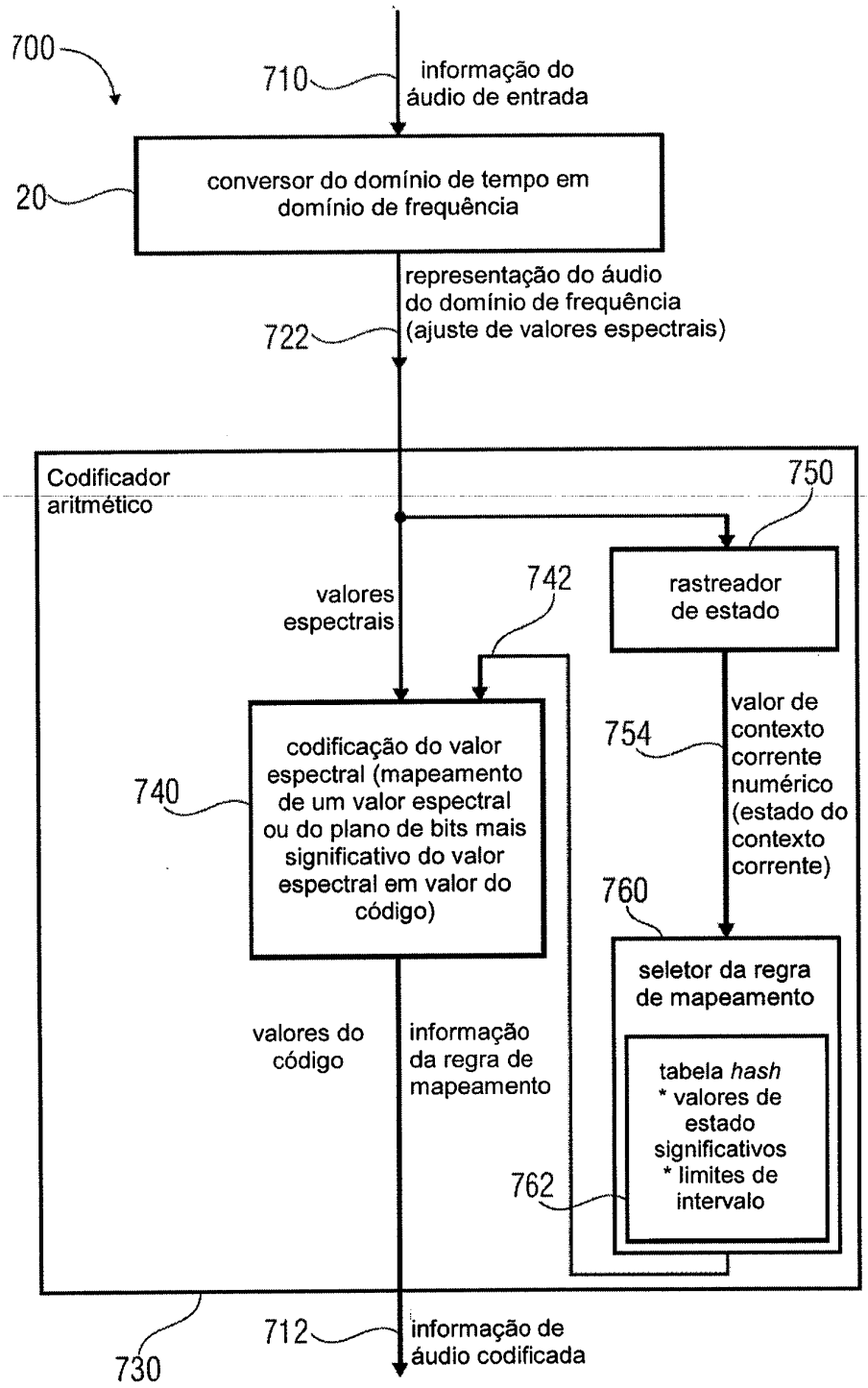
700
710
informação do áudio de entrada
20
conversor do domínio de tempo em domínio de frequência
representação do áudio do domínio de frequência (ajuste de valores espectrais)
722
Codificador aritmético
750
rastreador de estado
valores espectrais
742
valor de contexto corrente numérico (estado do contexto corrente)
754
codificação do valor espectral (mapeamento de um valor espectral ou do plano de bits mais significativo do valor espectral em valor do código)
740
760
seletor da regra de mapeamento
tabela hash
* valores de estado significativos
* limites de intervalo
762
valores do código
informação da regra de mapeamento
712
informação de áudio codificada
730

FIG 7

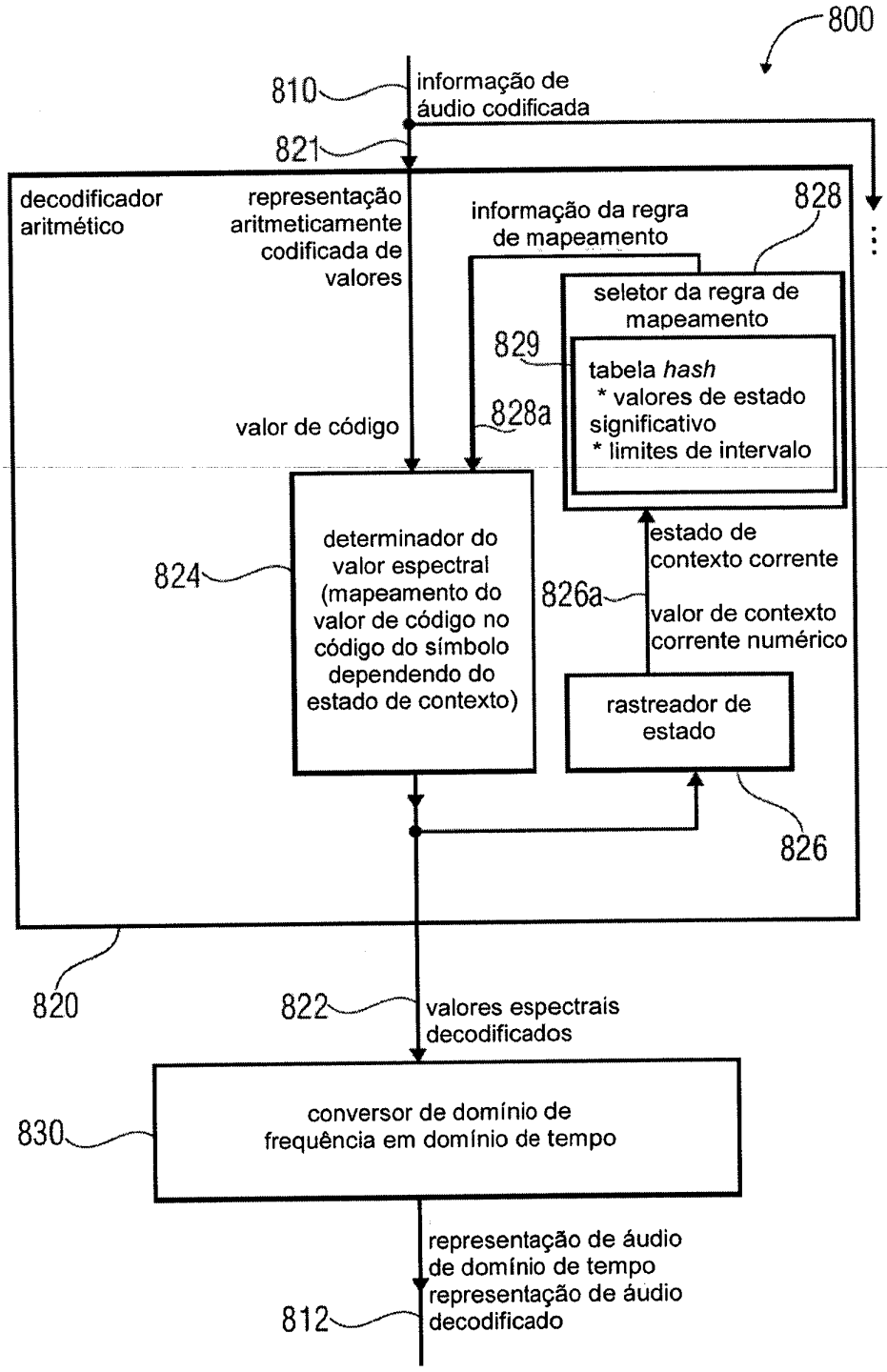
800
810
informação de áudio codificada
821
decodificador aritmético
representação aritmeticamente codificada de valores
informação da regra de mapeamento
828
seletor da regra de mapeamento
829
tabela hash
* valores de estado significativo
* limites de intervalo
828a
valor de código
824
determinador do valor espectral (mapeamento do valor de código no código do símbolo dependendo do estado de contexto)
estado de contexto corrente
826a
valor de contexto corrente numérico
rastreador de estado
826
820
822
valores espectrais decodificados
830
conversor de domínio de frequência em domínio de tempo
representação de áudio de domínio de tempo
representação de áudio decodificado
812

FIG 8

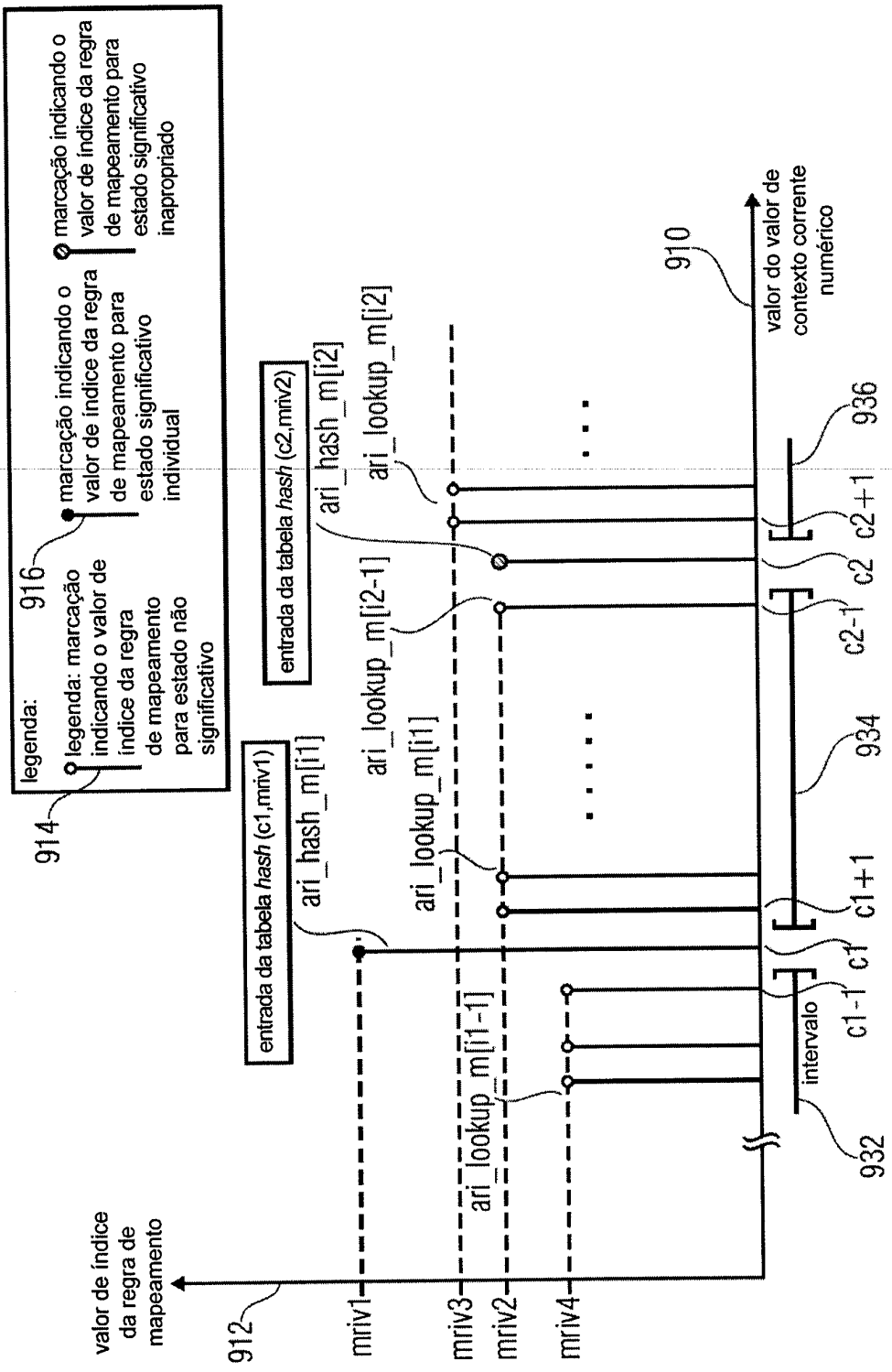
legenda:
916
legenda: marcação indicando o valor de índice da regra de mapeamento para estado não significativo
marcação indicando o valor de índice da regra de mapeamento para estado significativo individual
marcação indicando o valor de índice da regra de mapeamento para estado significativo inapropriado
914
entrada da tabela hash (c1,mriv1)
entrada da tabela hash (c2,mriv2)
ari_hash_m[i1]
ari_hash_m[i2]
ari_lookup_m[i1]
ari_lookup_m[i2]
ari_lookup_m[i1-1]
ari_lookup_m[i2-1]
valor de índice da regra de mapeamento
912
mriv1
mriv3
mriv2
mriv4
intervalo
c1-1
c1
c1+1
c2-1
c2
c2+1
932
934
936
910
valor do valor de contexto corrente numérico

FIG 9

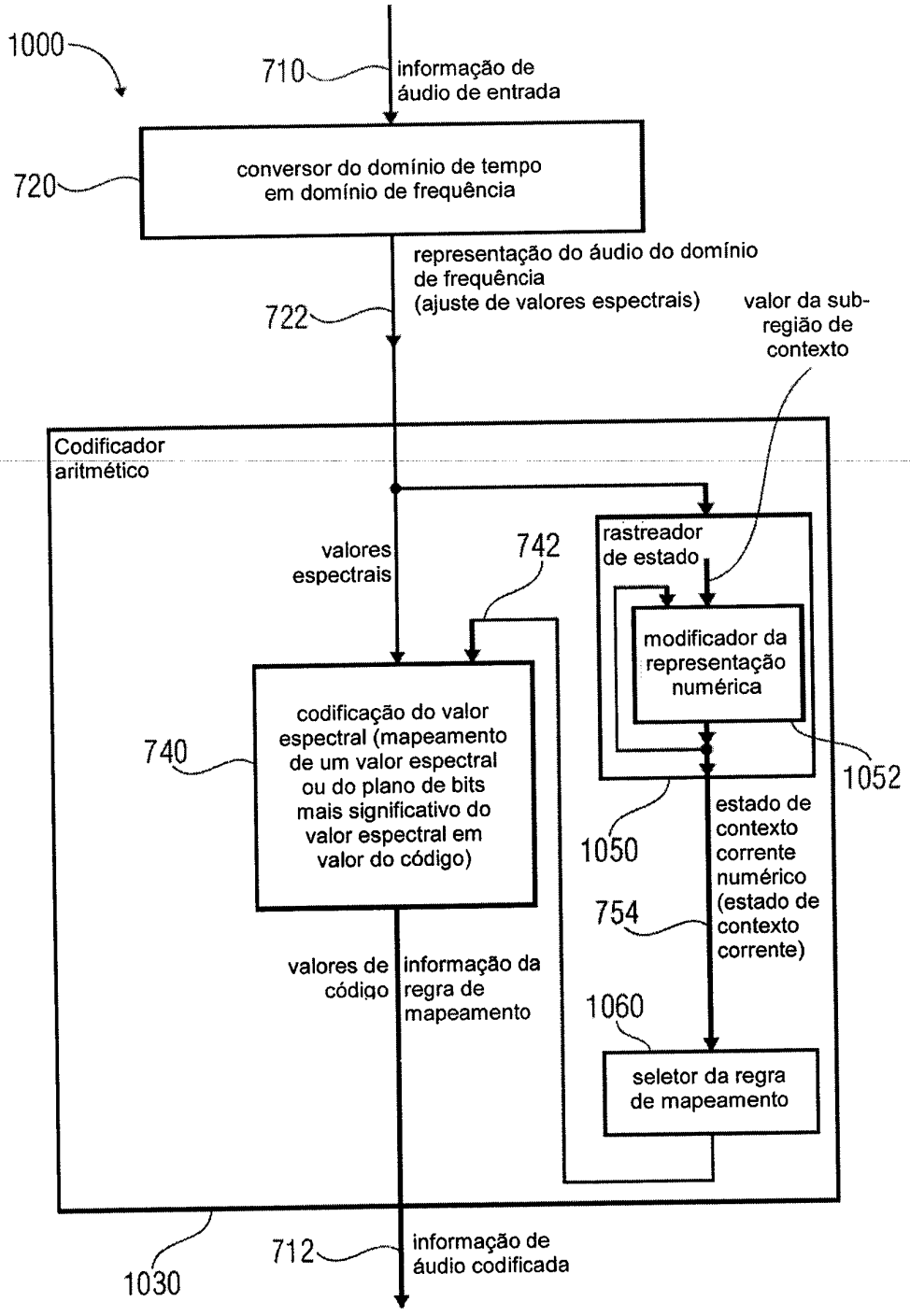
1000
710
informação de áudio de entrada
720
conversor do domínio de tempo em domínio de frequência
representação do áudio do domínio de frequência (ajuste de valores espectrais)
722
valor da sub-região de contexto
Codificador aritmético
valores espectrais
742
rastreador de estado
modificador da representação numérica
codificação do valor espectral (mapeamento de um valor espectral ou do plano de bits mais significativo do valor espectral em valor do código)
740
1052
1050
estado de contexto corrente numérico (estado de contexto corrente)
754
valores de código
informação da regra de mapeamento
1060
seletor da regra de mapeamento
1030
712
informação de áudio codificada

FIG 10

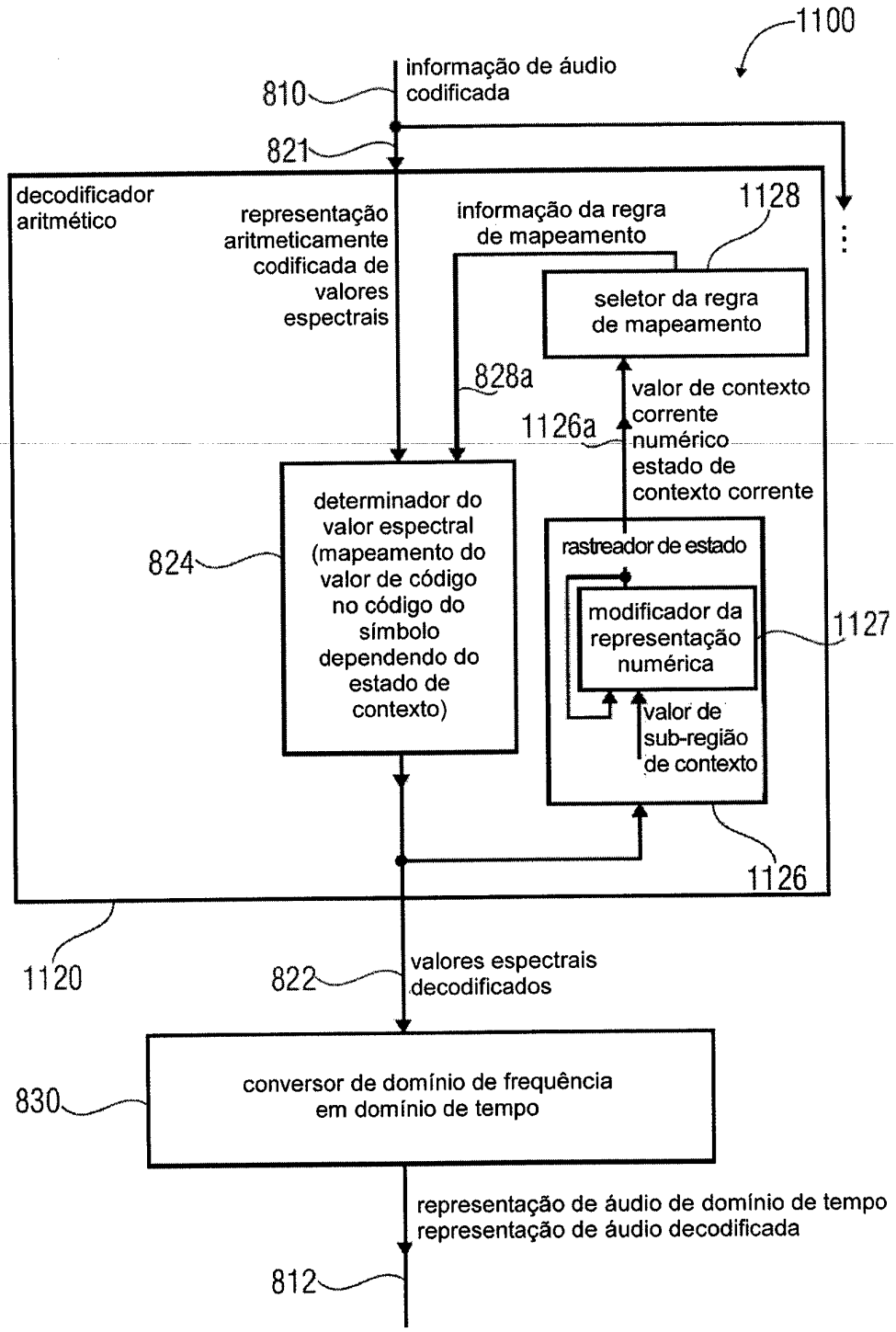
1100
informação de áudio codificada
810
821
decodificador aritmético
representação aritmeticamente codificada de valores espectrais
informação da regra de mapeamento
1128
seletor da regra de mapeamento
828a
1126a
valor de contexto corrente numérico
estado de contexto corrente
determinador do valor espectral (mapeamento do valor de código no código do símbolo dependendo do estado de contexto)
824
rastreador de estado
modificador da representação numérica
1127
valor de sub-região de contexto
1126
1120
822
valores espectrais decodificados
830
conversor de domínio de frequência em domínio de tempo
representação de áudio de domínio de tempo
representação de áudio decodificada
812

FIG 11

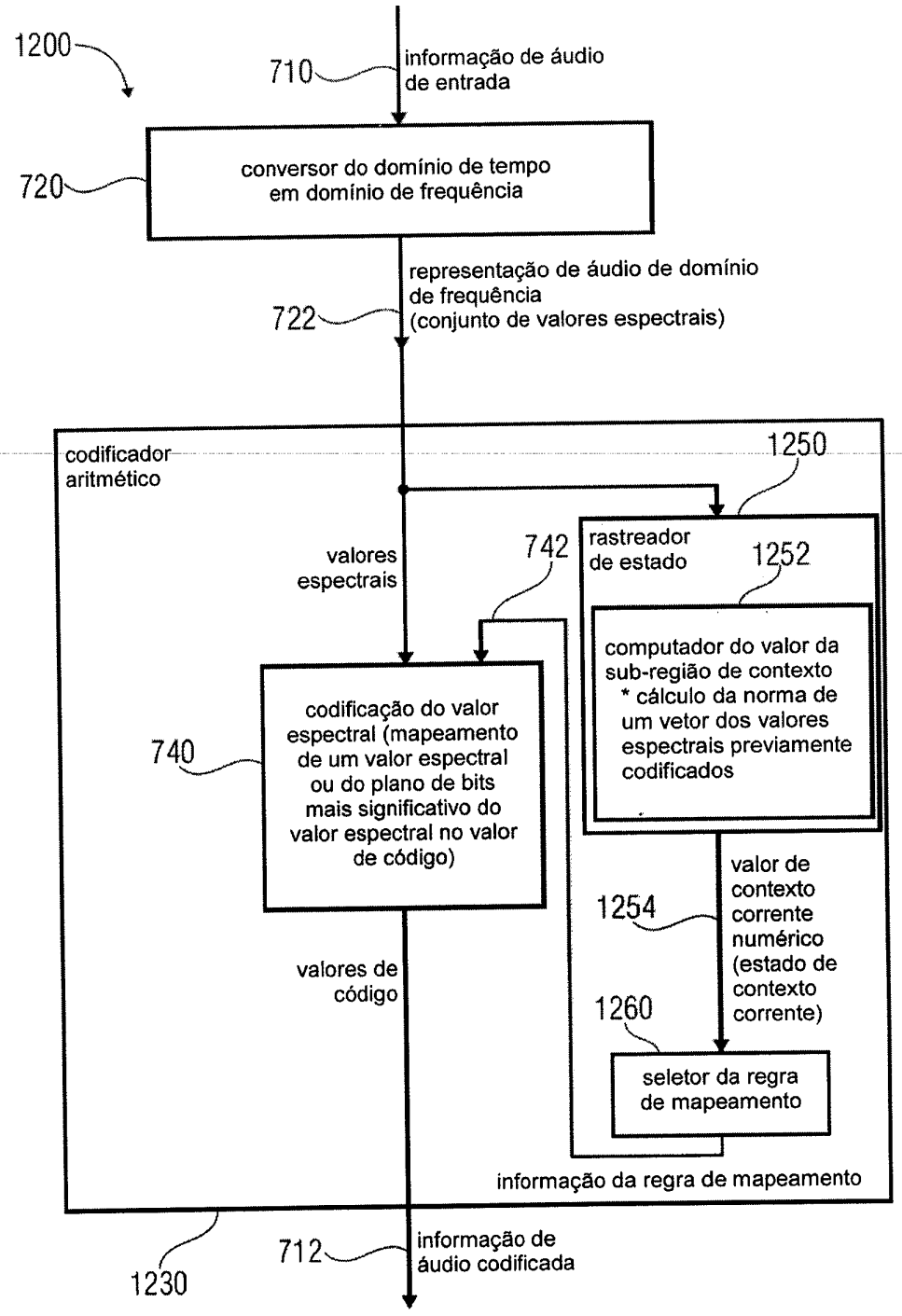
1200
710
informação de áudio de entrada
720
conversor do domínio de tempo em domínio de frequência
representação de áudio de domínio de frequência (conjunto de valores espectrais)
722
codificador aritmético
1250
rastreador de estado
1252
valores espectrais
742
computador do valor da sub-região de contexto * cálculo da norma de um vetor dos valores espectrais previamente codificados
codificação do valor espectral (mapeamento de um valor espectral ou do plano de bits mais significativo do valor espectral no valor de código)
740
valor de contexto corrente numérico (estado de contexto corrente)
1254
valores de código
1260
seletor da regra de mapeamento
informação da regra de mapeamento
1230
712
informação de áudio codificada

FIG 12

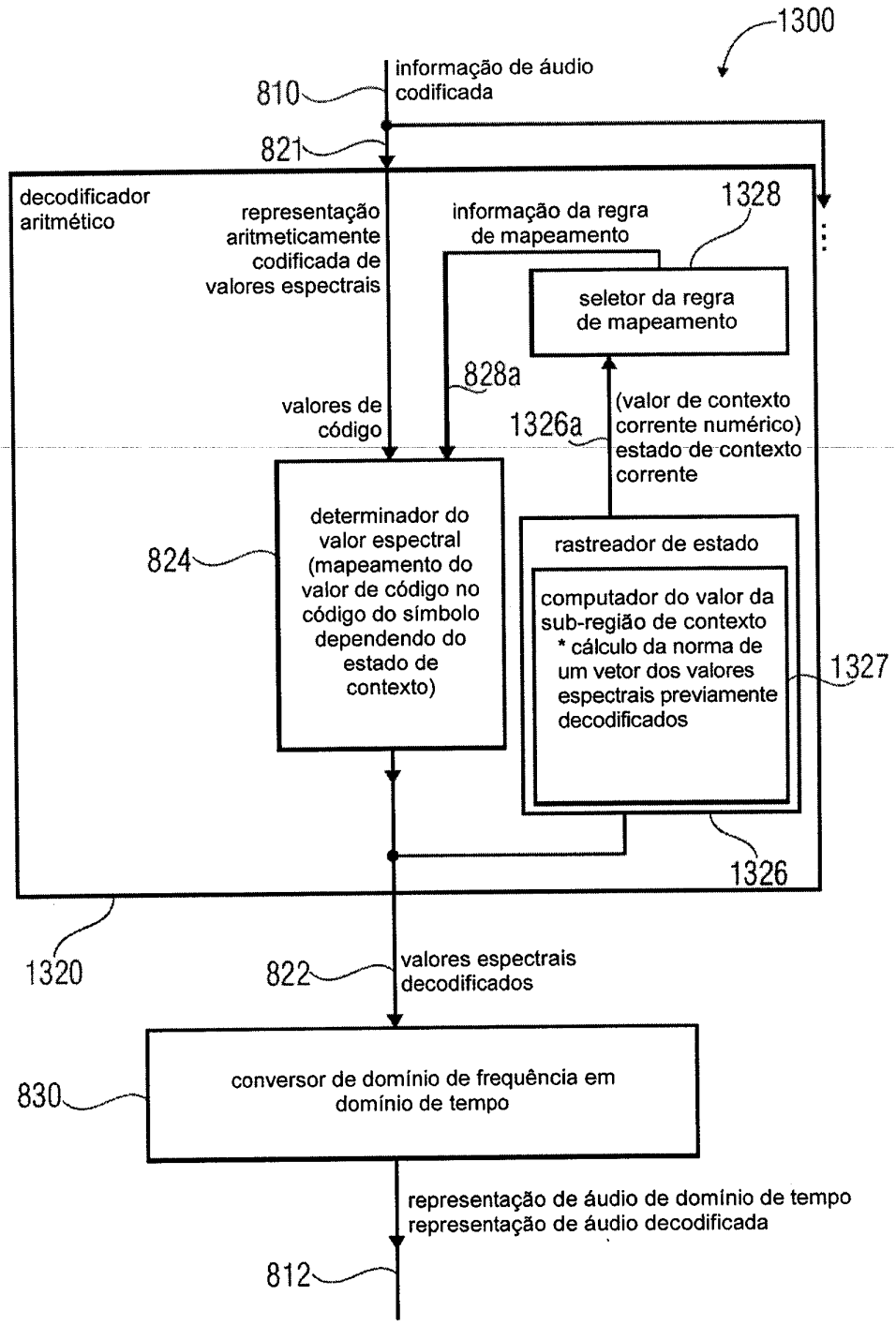
1300
informação de áudio codificada
810
821
decodificador aritmético
representação aritmeticamente codificada de valores espectrais
informação da regra de mapeamento
1328
seletor da regra de mapeamento
828a
valores de código
1326a
(valor de contexto corrente numérico) estado de contexto corrente
determinador do valor espectral (mapeamento do valor de código no código do símbolo dependendo do estado de contexto)
824
rastreador de estado
computador do valor da sub-região de contexto * cálculo da norma de um vetor dos valores espectrais previamente decodificados
1327
1326
1320
822
valores espectrais decodificados
conversor de domínio de frequência em domínio de tempo
830
representação de áudio de domínio de tempo
representação de áudio decodificada
812

FIG 13

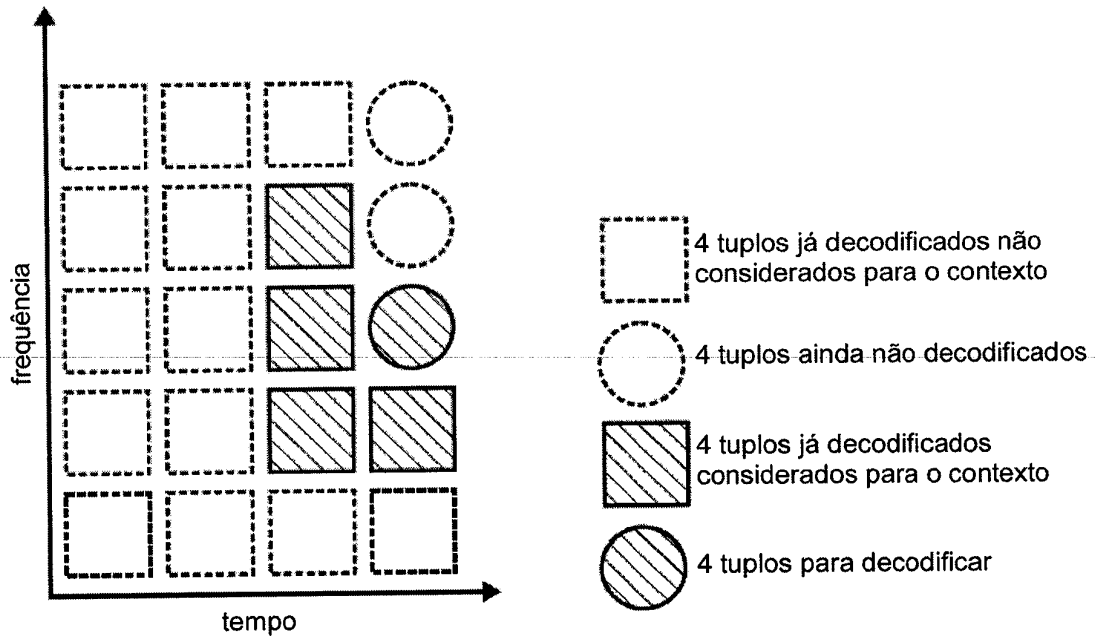
frequência
tempo
4 tuplos já decodificados não considerados para o contexto
4 tuplos ainda não decodificados
4 tuplos já decodificados considerados para o contexto
4 tuplos para decodificar

FIG 14A

TABELAS CONFORME UTILIZADAS NO ESQUEMA DE CODIFICAÇÃO ARITMÉTICA USAC WD4			
Nome da tabela	Descrição	Unidade de dados	Memória (palavras de 32 bits)
arith_cf_ng_hash[128]	Contexto de mapeamento da tabela *hash* em um índice de modelo da probabilidade.	Palavra	128
arith_cf_ng[32][545]	Frequências cumulativas de grupos para cada modo de distribuição de probabilidade, I	½ palavra	8720
egroups[8][8][8][8]	Índice de grupo de 4 tuplos	½ palavra	2048
dgvectors[4*4096]	Índice de grupo do mapa e índice do elemento para 4 tuplos.	¼ palavra	4096
dgroups[544]	Índice de grupo do mapa ao número cardinal do grupo e deslocamento em dgvectors.	Palavra	544
arith_cf_ne[2701]	Frequências cumulativas do símbolo do índice do elemento	½ Palavra	1350,5
arith_cf_r[16]	Frequências cumulativas de planos de bits menos significativos	½ palavra	8
total			**16894,5**

FIG 14B

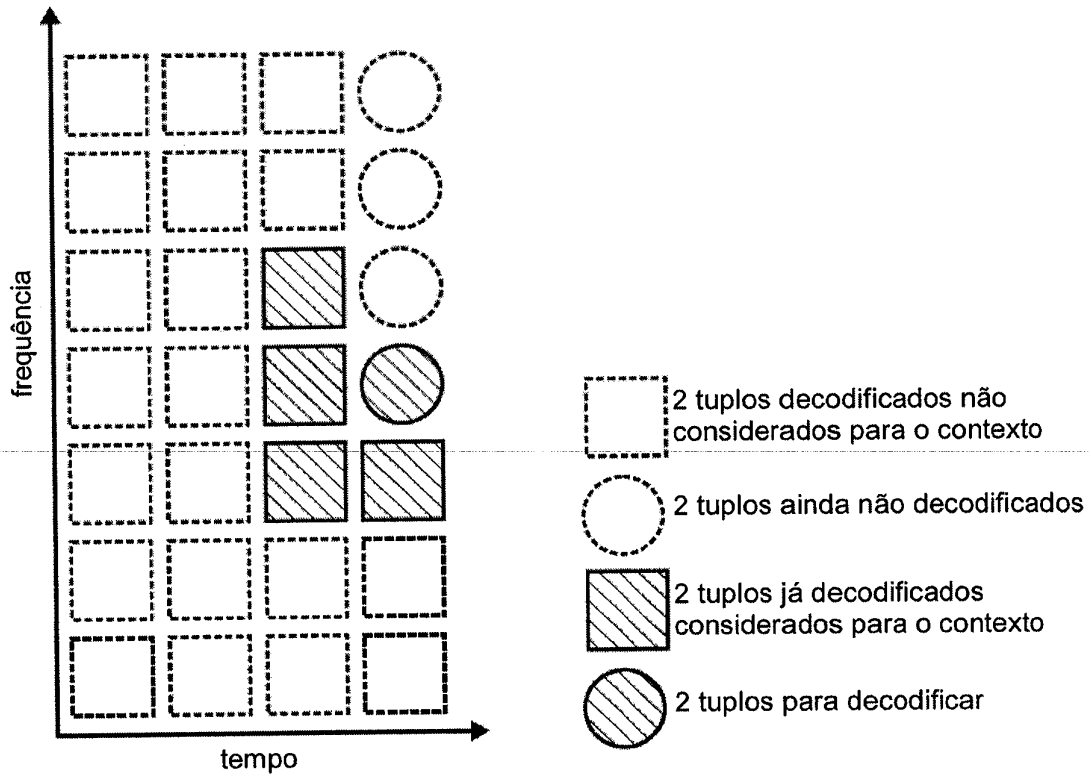
frequência
tempo
2 tuplos decodificados não considerados para o contexto
2 tuplos ainda não decodificados
2 tuplos já decodificados considerados para o contexto
2 tuplos para decodificar

FIG 15A

Nome da tabela	Descrição	Unidade de dados	Memória (palavras de 32 bits)
arith_hash[600]	Estados de mapeamento da tabela *hash* em um grupo de estados	1 palavra	600
arith_lookup[600]	Mapeamento da tabela de visualização	¼ palavra	150
arith_cf_msb[96][16]	Grupos de estados para uma tabela de frequências cumulativas Modelos das frequências cumulativas para o plano de 2 bits mais significativo m e o símbolo ARITH_ESCAPE	½ palavra	768
total			**1518**

FIG 15B

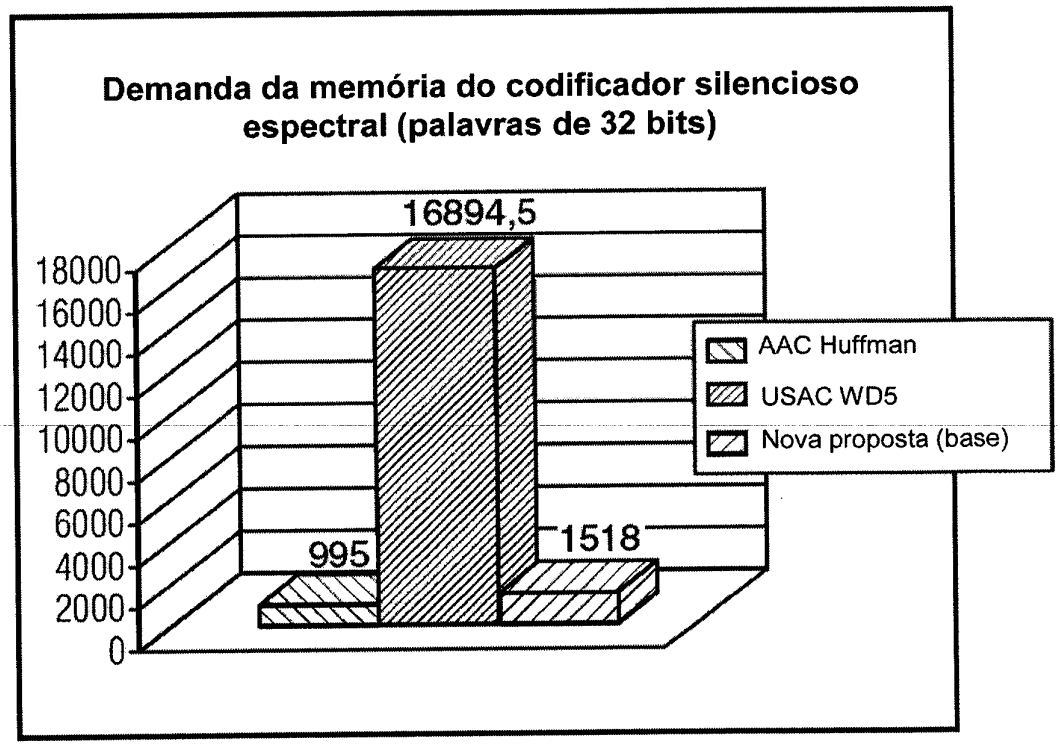
Demanda da memória do codificador silencioso espectral (palavras de 32 bits)
16894,5
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
995
1518
AAC Huffman
USAC WD5
Nova proposta (base)

FIG 16A

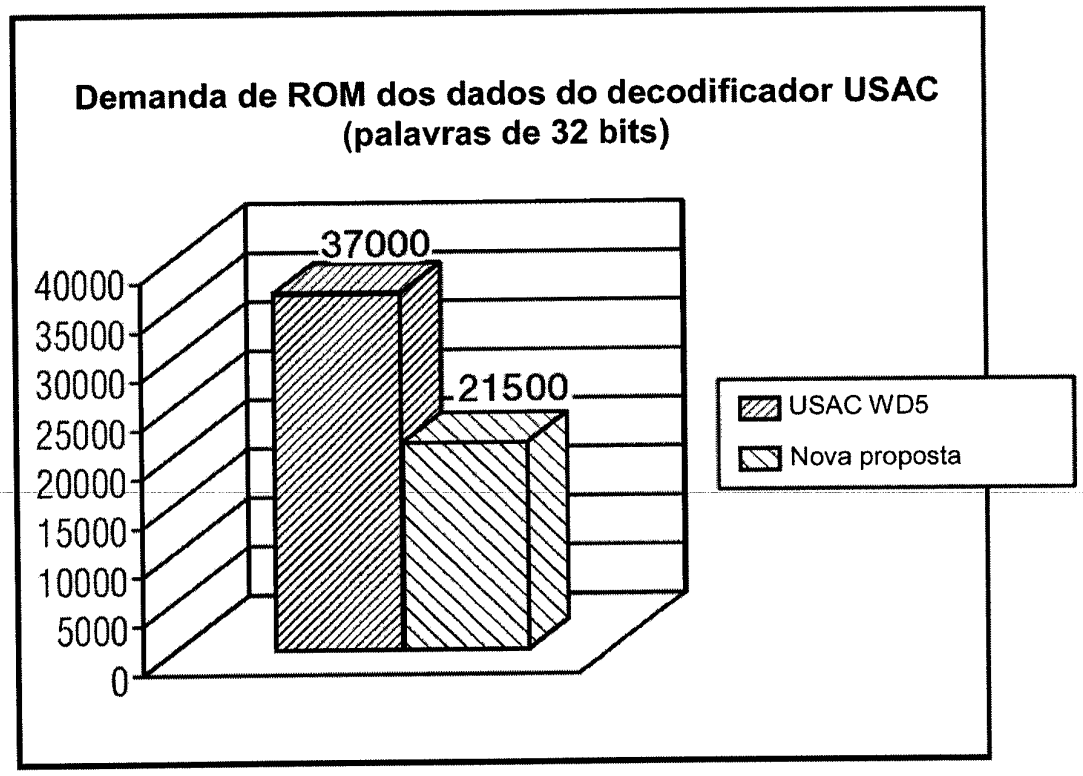
Demanda de ROM dos dados do decodificador USAC
(palavras de 32 bits)
37000
21500
40000
35000
30000
25000
20000
15000
10000
5000
0
USAC WD5
Nova proposta

FIG 16B

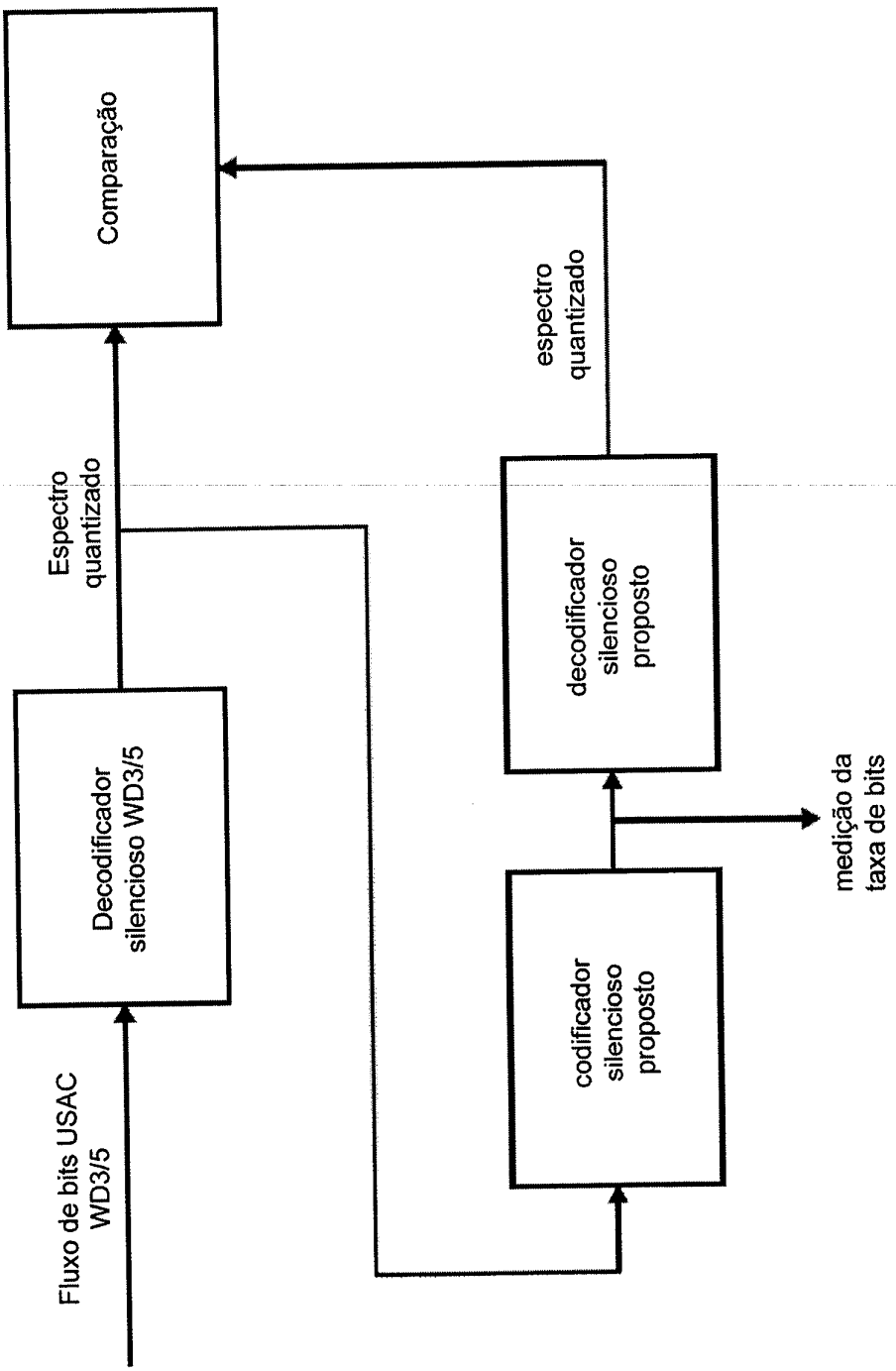
Comparação
espectro quantizado
Espectro quantizado
decodificador silencioso proposto
Decodificador silencioso WD3/5
medição da taxa de bits
codificador silencioso proposto
Fluxo de bits USAC WD3/5

FIG 17

Tabela: taxas de bits médias produzidas pelo codificador aritmético WD3 e a nova proposta. Taxas de bits líquidas não incluem bits para alinhamento de byte ou bits de preenchimento.				
Modo operacional	WD3 (kbit/s)	Nova proposta (kbit/s	Diferença após a transcodificação (kbit/s)	Diferença após a transcodificação (% de taxa de bit total)
Teste 1, 64s	64,00	62,78	-1,22	-1,90
Teste 2, 32s	32,00	31,47	-0,53	-166
Teste 3, 24s	24,00	23,61	-0,39	-1,64
Teste 4, 20s	20,00	19,65	-0,35	-1,74
Teste 5, 16s	16,00	15,72	-0,28	-1,75
Teste 6, 24m	24,00	23,56	-0,44	-1,83
Teste 7, 20m	20,00	19,61	-039	-1,95
Teste 8, 16m	16,00	15,70	-0,30	-1,87
Teste 9, 12m	12,00	11,77	-0,23	-1,92

FIG 18

Modo operacional	Controle do reservatório de bits					
	Nova proposta			WD3		
	mín.	máx.	média	mín.	máx.	média
Teste 1, 64kbps estéreo	3739	9557	8874	2314	9557	7018
Teste 2, 32kbps estéreo	2335	4505	4293	582	4505	3529
Teste 3, 24kbps estéreo	2184	4704	4472	957	4704	3871
Teste 4, 20kbps estéreo	2688	4864	4660	712	4864	3854
Teste 5, 16kbps estéreo	2965	5006	4859	724	5006	4234
Teste 6, 24kbps mono	2185	4704	4457	1002	4704	3927
Teste 7, 20kbps mono	2782	4864	4690	1192	4864	3935
Teste 8, 16kbps mono	2916	5006	4905	1434	5006	4450
Teste 9, 12kbps mono	3645	5184	5107	2256	5184	4787

FIG 19

	WD3	versão base
PCU (MHz)	0,953	0,823

FIG 20

```
static unsigned short ari_lookup_m[600] = {
0x02,0x01,0x03,0x38,0x3C,0x44,0x45,0x05,
0x07,0x37,0x41,0x08,0x0A,0x07,0x38,0x44,
0x0B,0x14,0x3E,0x3B,0x0C,0x14,0x3E,0x0C,
0x2B,0x5F,0x0F,0x42,0x3B,0x44,0x11,0x36,
0x42,0x41,0x13,0x2B,0x3E,0x3B,0x0C,0x14,
0x3E,0x14,0x2B,0x3E,0x15,0x07,0x3B,0x11,
0x25,0x42,0x41,0x16,0x36,0x3A,0x41,0x25,
0x36,0x3A,0x14,0x3E,0x36,0x42,0x36,0x3A,
0x2B,0x3E,0x42,0x38,0x41,0x17,0x19,0x42,
0x3B,0x44,0x19,0x1C,0x42,0x3B,0x44,0x1D,
0x2B,0x38,0x12,0x2B,0x1E,0x20,0x42,0x41,
0x22,0x36,0x37,0x41,0x24,0x36,0x42,0x41,
0x26,0x07,0x3A,0x2B,0x3E,0x01,0x3E,0x27,
0x07,0x37,0x44,0x3F,0x29,0x42,0x3B,0x44,
0x2A,0x07,0x37,0x41,0x29,0x07,0x38,0x2B,
0x3E,0x36,0x3E,0x39,0x42,0x34,0x07,0x3B,
0x26,0x36,0x42,0x41,0x36,0x42,0x3B,0x36,
0x3A,0x02,0x3A,0x03,0x38,0x07,0x37,0x07,
0x37,0x39,0x3A,0x3F,0x3B,0x41,0x44,0x57,
0x2C,0x07,0x3C,0x03,0x31,0x42,0x3C,0x22,
0x36,0x38,0x2A,0x07,0x27,0x2E,0x07,0x38,
0x44,0x2F,0x07,0x37,0x41,0x03,0x32,0x42,
0x3B,0x44,0x32,0x07,0x38,0x26,0x07,0x3A,
0x26,0x3E,0x02,0x2A,0x07,0x3B,0x33,0x03,
0x42,0x3B,0x44,0x1B,0x36,0x42,0x41,0x32,
0x07,0x3B,0x26,0x36,0x42,0x3B,0x36,0x3E,
0x39,0x42,0x39,0x42,0x3B,0x26,0x07,0x42,
0x41,0x36,0x42,0x37,0x41,0x36,0x42,0x3B,
0x07,0x3A,0x03,0x3B,0x07,0x42,0x3C,0x39,
0x42,0x3C,0x07,0x38,0x07,0x38,0x37,0x38,
0x3C,0x41,0x44,0x57,0x3B,0x3A,0x33,0x37,
0x33,0x03,0x02,0x33,0x07,0x3C,0x33,0x07,
0x3B,0x2A,0x34,0x42,0x41,0x34,0x07,0x38,
0x36,0x3E,0x26,0x07,0x3C,0x39,0x42,0x41,
0x33,0x36,0x42,0x3C,0x26,0x07,0x42,0x41,
0x07,0x38,0x07,0x3A,0x39,0x37,0x39,0x42,
0x3B,0x26,0x07,0x37,0x41,0x01,0x07,0x42,
0x3C,0x39,0x42,0x3B,0x03,0x3F,0x03,0x3B,
```

FIG 21(1)

```
0x39,0x42,0x3B,0x39,0x37,0x3C,0x39,0x37,
0x3C,0x03,0x38,0x3A,0x3B,0x3C,0x41,0x44,
0x57,0x03,0x33,0x03,0x26,0x37,0x34,0x42,
0x41,0x39,0x3F,0x34,0x42,0x39,0x37,0x39,
0x42,0x3B,0x26,0x07,0x37,0x41,0x07,0x3B,
0x07,0x3A,0x03,0x42,0x41,0x39,0x42,0x3C,
0x39,0x42,0x3C,0x07,0x3F,0x03,0x3B,0x03,
0x3B,0x39,0x37,0x3C,0x02,0x42,0x3C,0x03,
0x3D,0x40,0x3C,0x41,0x44,0x57,0x03,0x39,
0x3D,0x03,0x3B,0x39,0x42,0x41,0x39,0x3A,
0x03,0x3F,0x39,0x3F,0x02,0x3E,0x3F,0x02,
0x40,0x3C,0x41,0x44,0x27,0x03,0x03,0x03,
0x3D,0x3F,0x40,0x39,0x41,0x44,0x03,0x3D,
0x42,0x43,0x03,0x3C,0x44,0x03,0x3D,0x40,
0x3C,0x02,0x3C,0x44,0x3D,0x43,0x3C,0x02,
0x41,0x44,0x3D,0x43,0x3C,0x03,0x44,0x3D,
0x43,0x41,0x02,0x44,0x3D,0x41,0x44,0x43,
0x41,0x44,0x43,0x41,0x44,0x03,0x3C,0x44,
0x46,0x47,0x49,0x4B,0x50,0x4D,0x4F,0x0B,
0x56,0x0C,0x0C,0x2B,0x03,0x52,0x0D,0x10,
0x1A,0x56,0x56,0x57,0x58,0x58,0x0C,0x26,
0x01,0x01,0x02,0x3D,0x1A,0x1E,0x56,0x5A,
0x5B,0x5B,0x00,0x27,0x15,0x0C,0x39,0x39,
0x26,0x01,0x02,0x02,0x3D,0x57,0x57,0x27,
0x57,0x00,0x39,0x39,0x02,0x02,0x03,0x27,
0x27,0x02,0x27,0x02,0x02,0x3D,0x5E,0x5D,
0x5F,0x5E,0x5D,0x5D,0x5D,0x5D,0x5C,0x5C,
0x5F,0x5D,0x5D,0x5D,0x5E,0x5D,0x5D,0x5C,
0x5C,0x5C,0x5C,0x5F,0x5D,0x5D,0x5D,0x5E,
0x5D,0x5C,0x5C,0x5E,0x5C,0x5E,0x5C,0x5C,
0x5F,0x5D,0x5C,0x5D,0x5F,0x5D,0x5D,0x5F,
0x5E,0x5F,0x5D,0x5F,0x5D,0x5F,0x5F,0x5D,
0x5F,0x5D,0x5F,0x5D,0x5F,0x5F,0x5F,0x5F,
0x5F,0x5F,0x5F,0x5F,0x44,0x5F,0x5E,0x5D,
0x5D,0x5C,0x5C,0x5D,0x5C,0x5C,0x5C,0x5F,
0x5E,0x5D,0x5E,0x5D,0x5E,0x5D,0x5E,0x5F,
0x5E,0x5F,0x5E,0x5C,0x5E,0x5C,0x5E,0x5E,

};
```

FIG 21(2)

```
static unsigned long ari_hash_m[600] = {
0x00000100UL,0x00000302UL,0x0000053AUL,0x0000073BUL,0x00000A41UL,0x00000F44UL,
0x00111104UL,0x00111306UL,
0x00111542UL,0x0011173BUL,0x00111F44UL,0x00112209UL,0x0011242BUL,0x0011263EUL,
0x0011293CUL,0x00113105UL,
0x0011330CUL,0x0011352BUL,0x0011383AUL,0x00113F44UL,0x0011440CUL,0x0011462BUL,
0x00114F41UL,0x0011530CUL,
0x0012110DUL,0x0012120EUL,0x00121436UL,0x00121637UL,0x00121941UL,0x00122110UL,
0x00122312UL,0x00122507UL,
0x00122738UL,0x00123156UL,0x00123314UL,0x00123507UL,0x0012373AUL,0x00123F44UL,
0x00124212UL,0x0012452BUL,
0x00124F44UL,0x00125314UL,0x0012762BUL,0x00131121UL,0x00131323UL,0x00131542UL,
0x0013211DUL,0x00132216UL,
0x00132436UL,0x00132738UL,0x0013311DUL,0x00133329UL,0x00133507UL,0x00133838UL,
0x00134115UL,0x0013432BUL,
0x0013463EUL,0x00134F44UL,0x0013532BUL,0x0013FF3BUL,0x00142307UL,0x00143126UL,
0x00143407UL,0x00143E44UL,
0x00144307UL,0x0014FF3BUL,0x0015633EUL,0x0017863BUL,0x0021005AUL,0x00211218UL,
0x00211407UL,0x00211637UL,
0x00211941UL,0x0021211AUL,0x0021221BUL,0x00212436UL,0x00212637UL,0x00212941UL,
0x00213118UL,0x00213320UL,
0x00213507UL,0x00213F44UL,0x00214314UL,0x00220001UL,0x0022121FUL,0x00221407UL,
0x0022173BUL,0x00222121UL,
0x00222323UL,0x00222542UL,0x0022283BUL,0x0022311DUL,0x00223325UL,0x00223507UL,
0x00223737UL,0x00224115UL,
0x00224436UL,0x0022463EUL,0x00224F44UL,0x00225436UL,0x00225F44UL,0x0022622BUL,
0x0023112DUL,0x00231330UL,
0x00231542UL,0x0023183CUL,0x0023212DUL,0x00232228UL,0x00232407UL,0x00232637UL,
0x00232941UL,0x00233115UL,
0x0023332BUL,0x00233542UL,0x0023383BUL,0x0023412AUL,0x00234336UL,0x00234642UL,
0x00234F44UL,0x00235407UL,
0x00235F44UL,0x0023653EUL,0x0023FF3BUL,0x00241307UL,0x00241F44UL,0x00242336UL,
0x00242542UL,0x00242F44UL,
0x00243234UL,0x00243407UL,0x00243738UL,0x00244126UL,0x00244307UL,0x0024463AUL,
0x00244F44UL,0x00245442UL,
0x00245F44UL,0x00246236UL,0x0024FF3CUL,0x00252442UL,0x00252F44UL,0x00253303UL,
0x00253F44UL,0x00254303UL,
0x00254F44UL,0x00255303UL,0x0025FF41UL,0x0026733EUL,0x0027FF44UL,0x002AF203UL,
0x002FFF44UL,0x0031115BUL,
0x00311331UL,0x00311542UL,0x00312121UL,0x0031222DUL,0x00312436UL,0x00312637UL,
0x00312F44UL,0x00313335UL,
```

FIG 22(1)

```
0x00313507UL,0x00313F44UL,0x00314332UL,0x0031FF3CUL,0x0032112CUL,0x00321335UL,
0x00321542UL,0x0032183CUL,
0x0032212CUL,0x00322330UL,0x00322542UL,0x0032273BUL,0x0032312DUL,0x00323231UL
0x00323407UL,0x00323637UL,
0x00323A41UL,0x0032412AUL,0x00324407UL,0x00324642UL,0x00324F44UL,0x00325336UL,
0x00325507UL,0x00325F44UL,
0x00326234UL,0x0032FF3CUL,0x00331127UL,0x00331332UL,0x00331542UL,0x0033212EUL,
0x00332334UL,0x00332407UL,
0x00332637UL,0x00332941UL,0x00333115UL,0x00333235UL,0x00333407UL,0x00333738UL,
0x0033412AUL,0x00334336UL,
0x00334637UL,0x00334F44UL,0x00335234UL,0x00335407UL,0x0033573AUL,0x00335F44UL,
0x00336407UL,0x0033FF3CUL,
0x00341307UL,0x00341F44UL,0x00342307UL,0x00342542UL,0x00342F44UL,0x00343234UL,
0x00343442UL,0x0034373BUL,
0x00344126UL,0x00344307UL,0x00344542UL,0x0034483BUL,0x00345126UL,0x00345307UL,
0x00345637UL,0x00345F44UL,
0x00346442UL,0x0034FF3CUL,0x00352442UL,0x00353139UL,0x00353303UL,0x00353637UL,
0x00353F44UL,0x00354303UL,
0x00354637UL,0x00354F44UL,0x00355442UL,0x00356102UL,0x00356303UL,0x0035FF41UL,
0x00366203UL,0x0037733DUL,
0x0039E43AUL,0x003BF641UL,0x003FFF44UL,0x00411227UL,0x00411334UL,0x0041212CUL,
0x00412407UL,0x0041312EUL,
0x00413334UL,0x0041FF3CUL,0x00421127UL,0x00421334UL,0x00421542UL,0x00422127UL,
0x00422334UL,0x00422542UL,
0x00422F44UL,0x00423232UL,0x00423407UL,0x0042373BUL,0x00424126UL,0x00424336UL,
0x00424542UL,0x00424F44UL,
0x00425407UL,0x0042FF3CUL,0x00431339UL,0x00431542UL,0x00432133UL,0x00432407UL,
0x0043273BUL,0x00432F44UL,
0x00433234UL,0x00433407UL,0x00433637UL,0x00433F44UL,0x00434234UL,0x00434442UL,
0x00434738UL,0x00435126UL,
0x00435542UL,0x00435F44UL,0x00436442UL,0x0043FF41UL,0x00441303UL,0x00442126UL,
0x00442307UL,0x00442542UL,
0x00442F44UL,0x00443239UL,0x00443442UL,0x0044373BUL,0x00443F44UL,0x00444234UL,
0x00444442UL,0x00444637UL,
0x00444F44UL,0x00445307UL,0x00445637UL,0x00445F44UL,0x00446537UL,0x0044FF41UL,
0x00452442UL,0x00452F44UL,
0x00453303UL,0x00453537UL,0x00453F44UL,0x00454303UL,0x00454638UL,0x00454F44UL,
0x00455303UL,0x00455638UL,
0x00455F44UL,0x00456437UL,0x0045FF41UL,0x00466203UL,0x00478438UL,0x0049FF44UL,
0x004CF741UL,0x004FFF44UL,
```

FIG 22(2)

0x00511226UL,0x00512127UL,0x00512339UL,0x00521127UL,0x00521339UL,0x00522127UL,
0x00522339UL,0x00522637UL,
0x00523126UL,0x00523403UL,0x00524126UL,0x00524307UL,0x00531126UL,0x00531307UL,
0x00532126UL,0x00532307UL,
0x00532537UL,0x00532F44UL,0x00533239UL,0x00533442UL,0x0053373BUL,0x00534126UL,
0x00534542UL,0x00534F44UL,
0x00535442UL,0x0053FF3CUL,0x00542303UL,0x00542638UL,0x00543102UL,0x00543307UL,
0x00543638UL,0x00543F44UL,
0x00544307UL,0x00544537UL,0x00545102UL,0x00545303UL,0x0054FF41UL,0x00552437UL,
0x00552F44UL,0x00553437UL,
0x00553F44UL,0x00554303UL,0x0055463BUL,0x00554F44UL,0x00555307UL,0x00555537UL,
0x00556102UL,0x00556342UL,
0x00566203UL,0x00577342UL,0x0059733AUL,0x005CF53CUL,0x005FFF44UL,0x00611239UL,
0x00621127UL,0x00623307UL,
0x00631126UL,0x00632442UL,0x00632F44UL,0x00633303UL,0x00633638UL,0x00634102UL,
0x00634303UL,0x0063FF41UL,
0x00642303UL,0x00642F44UL,0x00643303UL,0x00643F44UL,0x00644207UL,0x00655203UL,
0x00665F44UL,0x00666303UL,
0x00677203UL,0x0069A53BUL,0x006DF841UL,0x006FFF44UL,0x00711239UL,0x00721127UL,
0x00722239UL,0x00733239UL,
0x00744437UL,0x00755203UL,0x00776F44UL,0x00777437UL,0x007BF33FUL,0x007FFF44UL,
0x00822239UL,0x00833203UL,
0x00854540UL,0x00888102UL,0x00888538UL,0x008BF23DUL,0x008FFF44UL,0x00922239UL,
0x0094333DUL,0x0096533DUL,
0x00998F44UL,0x00999303UL,0x009BF53BUL,0x009FFF44UL,0x00A44203UL,0x00A6633FUL,
0x00AA9F44UL,0x00AAA303UL,
0x00ADF33DUL,0x00AFFF44UL,0x00B33203UL,0x00B55440UL,0x00BBAC44UL,0x00BBB53BUL
0x00BFFF44UL,0x00C33203UL,
0x00C6423DUL,0x00CCBF44UL,0x00CCC303UL,0x00CFFF44UL,0x00D4423DUL,0x00DDD303U
0x00DFFF44UL,0x00E6453CUL,
0x00EEE342UL,0x00EFFF44UL,0x00F55343UL,0x00F7FF44UL,0x00FFEF44UL,0x00FFF33DUL,
0x00FFF744UL,0x01000145UL,
0x01011147UL,0x01111248UL,0x0111224AUL,0x0111324DUL,0x0112114CUL,0x0112214EUL,
0x01123108UL,0x01131150UL,
0x01132150UL,0x01133150UL,0x01141126UL,0x01144100UL,0x01211151UL,0x01212153UL,
0x01213108UL,0x01221154UL,
0x01222155UL,0x01223117UL,0x01224212UL,0x01231215UL,0x01232215UL,0x01233215UL,
0x01241126UL,0x01242239UL,
0x0124322BUL,0x01251103UL,0x01255100UL,0x01311159UL,0x01312155UL,0x01313117UL,
0x01315201UL,0x0132122CUL,

FIG 22(3)

```
0x0132222CUL,0x0132321DUL,0x01331157UL,0x01332158UL,0x0133315OUL,0x01341126UL,
0x01342127UL,0x01343100UL,
0x01344100UL,0x01351103UL,0x01355100UL,0x01369100UL,0x0141115AUL,0x01421227UL,
0x01422227UL,0x01431226UL,
0x01432226UL,0x01441127UL,0x01442127UL,0x01443100UL,0x01444100UL,0x01458100UL,
0x01511157UL,0x01531239UL,
0x01555100UL,0x01611157UL,0x01631239UL,0x01777100UL,0x01D33102UL,0x01FFF203UL,
0x0200055DUL,0x0200085DUL,
0x02000F5FUL,0x0211155DUL,0x02111F44UL,0x02112F5FUL,0x02121F44UL,0x02122F44UL,
0x02133F5FUL,0x0217FF5FUL,
0x021FFF44UL,0x02211F44UL,0x02212F44UL,0x02221F44UL,0x0222245EUL,0x02222F5FUL,
0x02223F5FUL,0x02232F5FUL,
0x02233F5FUL,0x02243F5FUL,0x022BFF5FUL,0x022FFF44UL,0x02322F44UL,0x02323F5FUL,
0x02332F5FUL,0x0233355CUL,
0x02333F5FUL,0x02334F5FUL,0x0233FF5CUL,0x02343F5FUL,0x0234FF5CUL,0x02353F5FUL,
0x02364F5FUL,0x0239655CUL,
0x023FFF44UL,0x02433F5FUL,0x0246445CUL,0x024A955DUL,0x024FFF44UL,0x0257435EUL,
0x025AC55CUL,0x025FFF44UL,
0x0267565DUL,0x026FFF44UL,0x0278655DUL,0x027FFF44UL,0x0288875DUL,0x028CC95DUL,
0x028FFF44UL,0x029A965DUL,
0x029FFF44UL,0x02ABA65DUL,0x02AFFF44UL,0x02BAFF5FUL,0x02BFFF44UL,0x02CCC45DUL,
0x02CFFF44UL,0x02DFFF44UL,
0x02EEE65DUL,0x02EFFF44UL,0x02F7335EUL,0x02FF0044UL,0x02FFEF44UL,0x02FFFF44UL,
0x0400045EUL,0x04000F5FUL,
0x04111F5DUL,0x04234F5DUL,0x042DFF5CUL,0x04343F5DUL,0x043FFF5FUL,0x044FFF5FUL,
0x045ED75CUL,0x045FFF5FUL,
0x046DFF5FUL,0x046FFF5FUL,0x047B5C5CUL,0x047FFF5FUL,0x048B435EUL,0x048FFF5FUL,
0x0499FF5CUL,0x04EFFF5FUL,
0x04FDFE5DUL,0x04FFFF5FUL,0x0600075EUL,0x06DFFF5FUL,0x06FFFF5FUL,0x08BFFF5CUL,
0x08FFFF5CUL,0x7FFFFFFF4FUL,

};
```

FIG 22(4)

```
unsigned short ari_cf_m [96][17] = {
{ 7529, 5263, 5173, 5162, 2636, 825, 674, 653, 511, 281, 210, 195, 173, 130, 105,
96,
  0
},
{ 10351, 7392, 7203, 7178, 4327, 1620, 1279, 1230, 1008, 606, 436, 399, 362, 288, 233,
212,
  0
},
{ 12505, 9566, 9216, 9157, 6631, 3220, 2541, 2418, 2086, 1404, 1024, 922, 858, 721,
597, 535,
  0
},
{ 14710,12600,11956,11801,10114, 7056, 5723, 5381, 4870, 3724, 2870, 2566, 2437, 2122,
1809, 1609,
  0
},
{ 4186, 2623, 2608, 2607, 939, 109, 86, 84, 63, 23, 14, 13, 11, 8, 5, 4,
  0
},
{ 7310, 4598, 4547, 4544, 2079, 354, 264, 259, 204, 90, 42, 38, 34, 25, 14, 11,
  0
},
{ 9998, 6785, 6641, 6630, 4087, 1058, 770, 746, 621, 339, 166, 143, 131, 104, 67,
49,
  0
},
{ 13801,11125,10550,10472, 8498, 5030, 3694, 3509, 3101, 2141, 1247, 1038, 975, 821,
623, 440,
  0
},
{ 5784, 3557, 3523, 3521, 1359, 164, 122, 119, 85, 29, 16, 14, 12, 8, 5, 4,
  0
},
{ 7617, 4716, 4649, 4645, 2129, 361, 258, 252, 191, 80, 39, 34, 30, 22, 13, 10,
  0
},
{ 9553, 6223, 6064, 6052, 3524, 912, 643, 620, 501, 262, 132, 112, 102, 79, 52, 39,
  0
},
{ 8423, 5300, 5186, 5179, 2629, 539, 375, 362, 276, 119, 61, 53, 46, 33, 21, 17,
  0
},
{ 9570, 6162, 5952, 5936, 3574, 1016, 702, 670, 538, 282, 152, 129, 117, 90, 63,
50,
  0
},
```

2310

2312

FIG 23(1)

```
{ 7355, 4678, 4630, 4628, 1834, 252, 190, 187, 124, 38, 21, 19, 16, 10, 6, 5,
  0
},
{ 8916, 5717, 5627, 5622, 2647, 471, 340, 332, 242, 97, 46, 41, 36, 25, 15, 12,
  0
},
{ 10839, 7394, 7198, 7184, 4273, 1169, 832, 803, 643, 327, 167, 143, 130, 100, 66,
49,
  0
},
{ 8036, 5132, 5045, 5040, 2207, 350, 250, 243, 167, 56, 31, 28, 23, 15, 10, 8,
  0
},
{ 9663, 6256, 6110, 6101, 3064, 633, 441, 426, 311, 127, 65, 57, 49, 34, 22, 18,
  0
},
{ 11237, 7629, 7361. 7339, 4460, 1302, 898, 859, 675, 333, 178, 152, 136, 104, 73,
57,
  0
},
{ 10436, 6930, 6718, 6703, 3735, 902, 619, 595, 446, 189, 98, 85, 74, 51, 34, 28,
  0
},
{ 11678, 7933, 7574, 7541, 4826, 1560, 1057, 1002, 796, 406, 218, 184, 165, 124, 88,
70,
  0
},
{ 10597, 7265, 7057, 7039, 3904, 1071, 768, 738, 541, 227, 127, 111, 93, 60, 40,
34,
  0
},
{ 11298, 7892, 7596, 7566, 4416, 1276, 873, 832, 611, 261, 141, 122, 103, 70, 47,
40,
  0
},
{ 6435, 4238, 4210, 4208, 1558, 216, 174, 172, 116, 39, 24, 22, 18, 11, 7, 6,
  0
},
{ 8744, 5744, 5671, 5667, 2636, 496, 375, 368, 273, 113, 58, 52, 45, 32, 19, 15,
  0
},
{ 10899, 7632, 7457, 7444, 4488, 1270, 942, 914, 737, 383, 204, 177, 161, 122, 80,
62,
  0
},
```

FIG 23(2)

```
{ 7929, 5176, 5119, 5116, 2169, 307, 230, 226, 155, 51, 28, 25, 21, 13, 8, 7,
  0
},
{ 9520, 6304, 6204, 6199, 3044, 585, 424, 415, 304, 118, 59, 52, 45, 31, 19, 15,
  0
},
{ 11187, 7805, 7600, 7586, 4554, 1292, 928, 896, 703, 342, 178, 153, 138, 103, 68,
52,
  0
},
{ 10371, 7037, 6881, 6871, 3721, 860, 613, 596, 440, 176, 88, 77, 67, 46, 28, 23,
  0
},
{ 8562, 5804, 5738, 5734, 2472, 467, 374, 369, 233, 76, 48, 44, 34, 20, 13, 11,
  0
},
{ 10078, 6838, 6713, 6705, 3407, 782, 591, 578, 407, 164, 92, 83, 69, 46, 29, 24,
  0
},
{ 11694, 8342, 8108, 8089, 4999, 1559, 1143, 1105, 858, 424, 240, 210, 185, 135, 92,
73,
  0
},
{ 9115, 6191, 6084, 6077, 2924, 604, 455, 445, 304, 109, 64, 58, 47, 29, 20, 17,
  0
},
{ 10786, 7434, 7256, 7244, 3968, 1006, 734, 712, 517, 213, 118, 104, 88, 60, 40,
34,
  0
},
{ 12216, 8840, 8535, 8507, 5492, 1860, 1342, 1289, 1007, 505, 289, 249, 221, 163, 115,
94,
  0
},
{ 11473, 8085, 7843, 7825, 4602, 1304, 931, 898, 669, 283, 153, 135, 116, 80, 53,
44,
  0
},
{ 12650, 9279, 8908, 8872, 5924, 2160, 1539, 1470, 1162, 585, 329, 282, 251, 184, 130,
107,
  0
},
{ 12597, 9310, 8859, 8806, 6019, 2457, 1741, 1643, 1310, 684, 401, 343, 304, 225, 164,
137,
  0
},
```

FIG 23(3)

```
{ 11509, 8255, 8030, 8012, 4706, 1489, 1135, 1103,  793,  347,  212,  192,  154,   98,   68,
59,
  0
},
{ 11946, 8648, 8370, 8344, 5106, 1628, 1185, 1142,  842,  366,  206,  183,  153,  100,   68,
57,
  0
},
{ 13088, 9856, 9444, 9397, 6468, 2535, 1831, 1745, 1384,  700,  407,  352,  310,  224,  160,
132,
  0
},
{ 12262, 9021, 8683, 8647, 5519, 1929, 1405, 1342, 1024,  481,  280,  245,  210,  146,  104,
89,
  0
},
{ 13353,10225, 9742, 9678, 6908, 2945, 2133, 2017, 1619,  875,  524,  450,  401,  298,  218,
183,
  0
},
{ 10055, 6990, 6879, 6872, 3544,  882,  687,  675,  490,  209,  121,  112,   95,   63,   41,   36,
  0
},
{ 10647, 7427, 7278, 7270, 3911,  968,  723,  705,  513,  203,  109,   98,   83,   55,   34,   28,
  0
},
{ 11221, 8027, 7861, 7851, 4397, 1264,  981,  961,  689,  293,  174,  159,  132,   83,   54,
46,
  0
},
{ 11700, 8429, 8209, 8192, 4825, 1451, 1079, 1049,  767,  327,  186,  167,  140,   92,   61,
52,
  0
},
{ 12949, 9762, 9414, 9379, 6351, 2418, 1786, 1717, 1351,  686,  398,  345,  301,  216,  152,
125,
  0
},
{ 12208, 8979, 8703, 8682, 5437, 1780, 1303, 1261,  955,  422,  231,  204,  175,  120,   77,
63,
  0
},
{ 13277,10148, 9741, 9698, 6807, 2800, 2047, 1960, 1563,  808,  463,  398,  352,  255,  174,
140,
  0
},
```

FIG 23(4)

```
{ 12426, 9284, 8976, 8947, 5753, 2124, 1609, 1555, 1166,  548,  333,  298,  247,  162,  113,
99,
  0
},
{ 13492,10473,10042, 9988, 7177, 3133, 2347, 2244, 1797,  970,  594,  520,  457,  331,
241,  204,
  0
},
{ 12407, 9334, 9013, 8977, 5920, 2276, 1708, 1642, 1284,  642,  392,  348,  299,  211,  153,
132,
  0
},
{ 13579,10631,10182,10121, 7466, 3396, 2542, 2420, 1975, 1112,  691,  604,  542,  408,
300,  256,
  0
},
{ 15415,14163,13591,13335,12111, 9885, 8640, 8070, 7541, 6363, 5413, 4919, 4688, 4157,
3662, 3364,
  0
},
{ 15645,14640,14124,13838,12910,11112, 9969, 9338, 8909, 7923, 7060, 6507, 6315, 5841,
5365, 5003,
  0
},
{ 13803,11130,10691,10636, 7807, 3900, 3070, 2956, 2343, 1300,  866,  781,  667,  462,
337,  292,
  0
},
{ 15381,14112,13568,13348,12083, 9783, 8595, 8105, 7565, 6347, 5408, 4945, 4704, 4158,
3671, 3377,
  0
},
{ 15595,14555,14062,13813,12830,10944, 9820, 9259, 8813, 7769, 6872, 6351, 6149, 5655,
5160, 4798,
  0
},
{ 15855,15124,14719,14487,13812,12440,11517,10975,10628, 9802, 9036, 8493, 8312,
7869,7383, 6992,
  0
},
{ 15484,14324,13800,13578,12460,10325, 9149, 8632, 8118, 6954, 6030, 5548, 5320, 4797,
4296, 3961,
  0
},
{ 15407,14130,13600,13402,12107, 9730, 8527, 8059, 7475, 6205, 5245, 4803, 4563, 4022,
3557, 3260,
  0
},
```

FIG 23(5)

```
{ 15475,14322,13830,13621,12482,10384, 9270, 8779, 8266, 7125, 6200, 5730, 5493, 4962,
4469, 4140,
  0
},
{ 15596,14601,14153,13948,12948,11040, 9997, 9517, 9048, 7949, 7054, 6560, 6325, 5785,
5280, 4913,
  0
},
{ 15871,15158,14789,14602,13921,12540,11660,11198,10847,10009, 9224, 8719, 8532,
8078, 7594, 7190,
  0
},
{ 15474,14312,13843,13669,12397,10171, 9109, 8697, 8063, 6764, 5886, 5470, 5181, 4574,
4090, 3790,
  0
},
{ 15720,14858,14443,14249,13373,11680,10700,10226, 9779. 8747, 7881, 7386, 7156,
6611, 6097, 5712,
  0
},
{ 16133,15786,15587,15472,15104,14369,13854,13555,13314,12738,12214,11861,11701,
11314,10905,10579,
  0
},
{ 2594, 1645, 1633, 1631, 535, 108, 92, 89, 64, 34, 27, 25, 21, 15, 12, 11,
  0
},
{ 7130, 4521, 4444, 4431, 2088, 598, 489, 469, 378, 235, 185, 173, 153, 120, 100,
94,
  0
},
{ 1326, 780, 778, 777, 173, 17, 15, 14, 11, 7, 6, 5, 4, 3, 2, 1,
  0
},
{ 4974, 2832, 2814, 2812, 945, 102, 78, 76, 56, 26, 16, 15, 13, 9, 6, 5,
  0
},
{ 3593, 2051, 2043, 2042, 530, 36, 28, 27, 18, 8, 6, 5, 4, 3, 2, 1,
  0
},
{ 5521, 3055, 3028, 3026, 1052, 102, 72, 69, 49, 20, 12, 11, 9, 6, 4, 3,
  0
},
{ 4294, 2539, 2520, 2518, 812, 83, 61, 59, 41, 17, 12, 11, 9, 6, 4, 3,
  0
},
```

FIG 23(6)

```
{ 4456, 2667, 2653, 2652, 724, 67, 54, 53, 31, 11, 8, 7, 5, 3, 2, 1,
  0
},
{ 6684, 3922, 3872, 3869, 1425, 186, 133, 129, 86, 35, 21, 19, 15, 9, 6, 5,
  0
},
{ 5087, 3024, 2988, 2985, 965, 99, 71, 69, 43, 15, 10, 9, 7, 4, 3, 2,
  0
},
{ 7587, 4403, 4297, 4289, 1806, 279, 181, 171, 118, 49, 30, 27, 22, 15, 11, 10,
  0
},
{ 6242, 4036, 3962, 3957, 1604, 337, 256, 251, 154, 49, 31, 28, 20, 10, 6, 5,
  0
},
{ 3727, 2352, 2346, 2345, 568, 53, 46, 45, 27, 9, 7, 6, 5, 3, 2, 1,
  0
},
{ 6369, 3904, 3876, 3874, 1372, 186, 150, 147, 101, 42, 27, 25, 20, 13, 9, 8,
  0
},
{ 5012, 3060, 3046, 3045, 921, 76, 60, 59, 37, 13, 9, 8, 6, 4, 3, 2,
  0
},
{ 5471, 3590, 3569, 3568, 1118, 176, 153, 151, 83, 26, 19, 18, 13, 7, 5, 4,
  0
},
{ 5866, 3774, 3736, 3733, 1366, 204, 163, 159, 101, 37, 26, 24, 19, 11, 8, 7,
  0
},
{ 8648, 5538, 5437, 5427, 2527, 521, 397, 382, 269, 121, 81, 75, 60, 40, 30, 27,
  0
},
{ 7300, 5124, 5049, 5043, 2284, 682, 582, 574, 341, 124, 90, 86, 59, 29, 20, 18,
  0
},
{ 7183, 4913, 4817, 4808, 2167, 517, 404, 393, 252, 92, 63, 59, 43, 24, 17, 15,
  0
},
{ 4749, 3285, 3273, 3272, 938, 157, 141, 140, 81, 27, 21, 20, 15, 8, 6, 5,
  0
},
{ 6602, 4705, 4676, 4674, 1689, 388, 347, 344, 192, 63, 49, 47, 33, 16, 11, 10,
  0
},
```

FIG 23(7)

```
{ 6849, 4648, 4599, 4596, 1868, 367, 304, 299, 187, 68, 49, 46, 34, 19, 14, 13,
  0
},
{ 16383,16382,14098,13672,13671,13670,11058,10499, 8080, 5456, 4230, 3829, 3412,
2876, 2494, 2264,
  0
},
{ 16383,16382,14436,14062,14061,14060,11574,11038, 8597, 5905, 4621, 4204, 3720,
3106, 2674, 2430,
  0
},
{ 16383,16382,14635,14264,14263,14262,11898,11372, 8910, 6216, 4935, 4506, 3962,
3287, 2835, 2578,
  0
},
{ 16383,16382,14851,14380,14379,14378,12610,12032, 9670, 7609, 6543, 6085, 5335,
4590, 4112, 3825,
  0
}
};
```

2396

FIG 23(8)

```
unsigned short ari_cf_r [4] ={(3<<14)/4,(2<<14)/4,(1<<14)/4, 0};
```

FIG 24