US 20020032712A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0032712 A1

Miyasaka et al. (43) Pub. Date: Mar. 14, 2002

(54) EXPONENTIAL CALCULATION DEVICE AND DECODING DEVICE

(76) Inventors: **Shuji Miyasaka**, Osaka (JP); **Takeshi Fujita**, Osaka (JP); **Masahiro Sueyoshi**, Osaka (JP); **Akihisa Kawamura**, Osaka (JP); **Masaharu Matsumoto**, Osaka (JP); **Takashi Katayama**, Osaka (JP); **Kazutaka Abe**, Osaka (JP); **Kosuke Nishio**, Osaka (JP)

Correspondence Address:
**Ratner & Prestia**
**P.O. Box 980**
**Valley Forge, PA 19482 (US)**

(21) Appl. No.: **09/929,156**

(22) Filed: **Aug. 14, 2001**

**Related U.S. Application Data**

(62) Division of application No. 09/244,155, filed on Feb. 3, 1999, now Pat. No. 6,304,890.

(30) **Foreign Application Priority Data**

Feb. 4, 1998 (JP) ..... 10-023022
May 14, 1998 (JP) ..... 10-131636
May 28, 1998 (JP) ..... 10-147348
Jan. 14, 1999 (JP) ..... 11-008562
Jan. 26, 1999 (JP) ..... 11-017702

**Publication Classification**

(51) **Int. Cl.**[7] ..... **G06F 7/38**
(52) **U.S. Cl.** ..... **708/606**

(57) **ABSTRACT**

A device of the present invention is an exponential calculation device for calculating $x^{-}(a/b)$ (where a and b are each an integer constant) for a given input value of x. The device includes: an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 b when x>A: a core section for outputting a value of z'=x' (a/b); and an output control section for outputting a value of z, wherein z=z' when x≧A and z=z'*2 a when x>A.

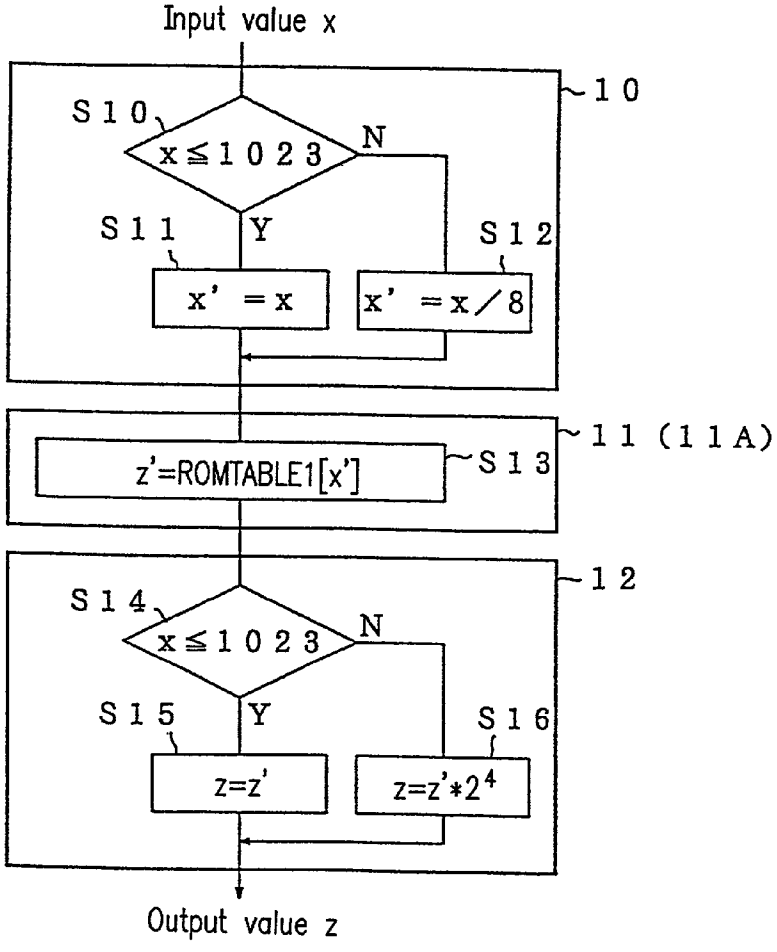

*FIG. 1*

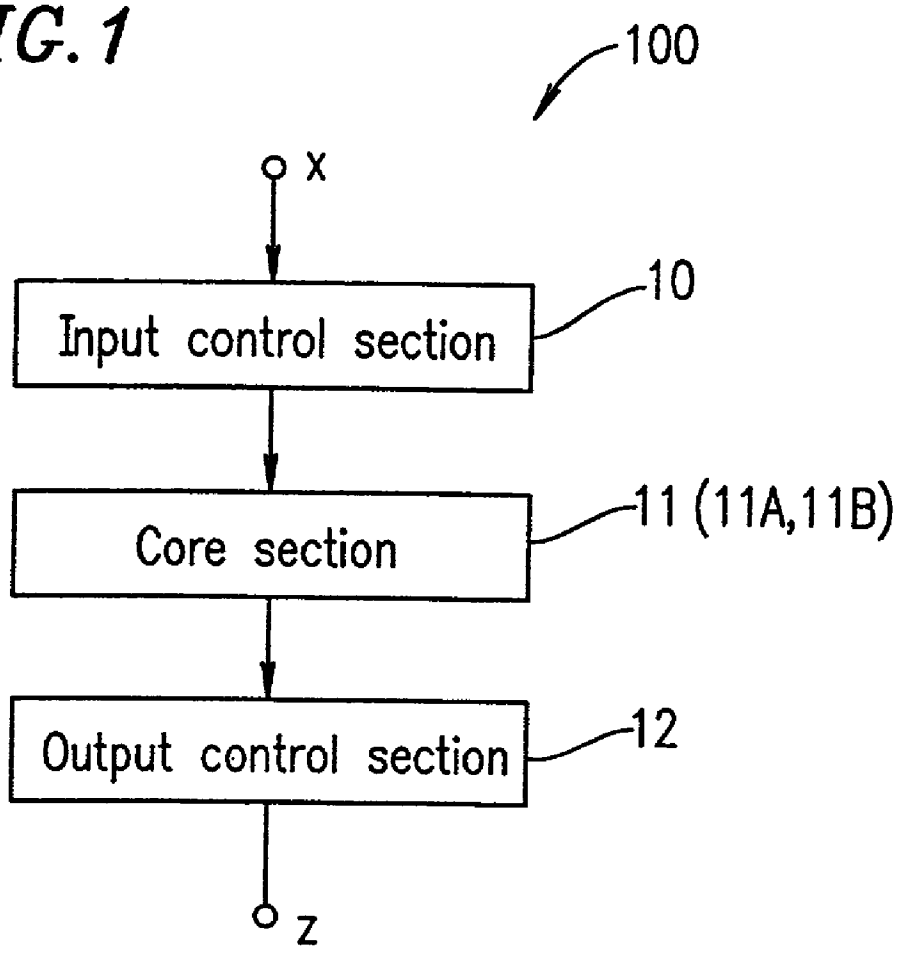
100
X
Input control section
10
Core section
11 (11A,11B)
Output control section
12
Z

*FIG. 2*

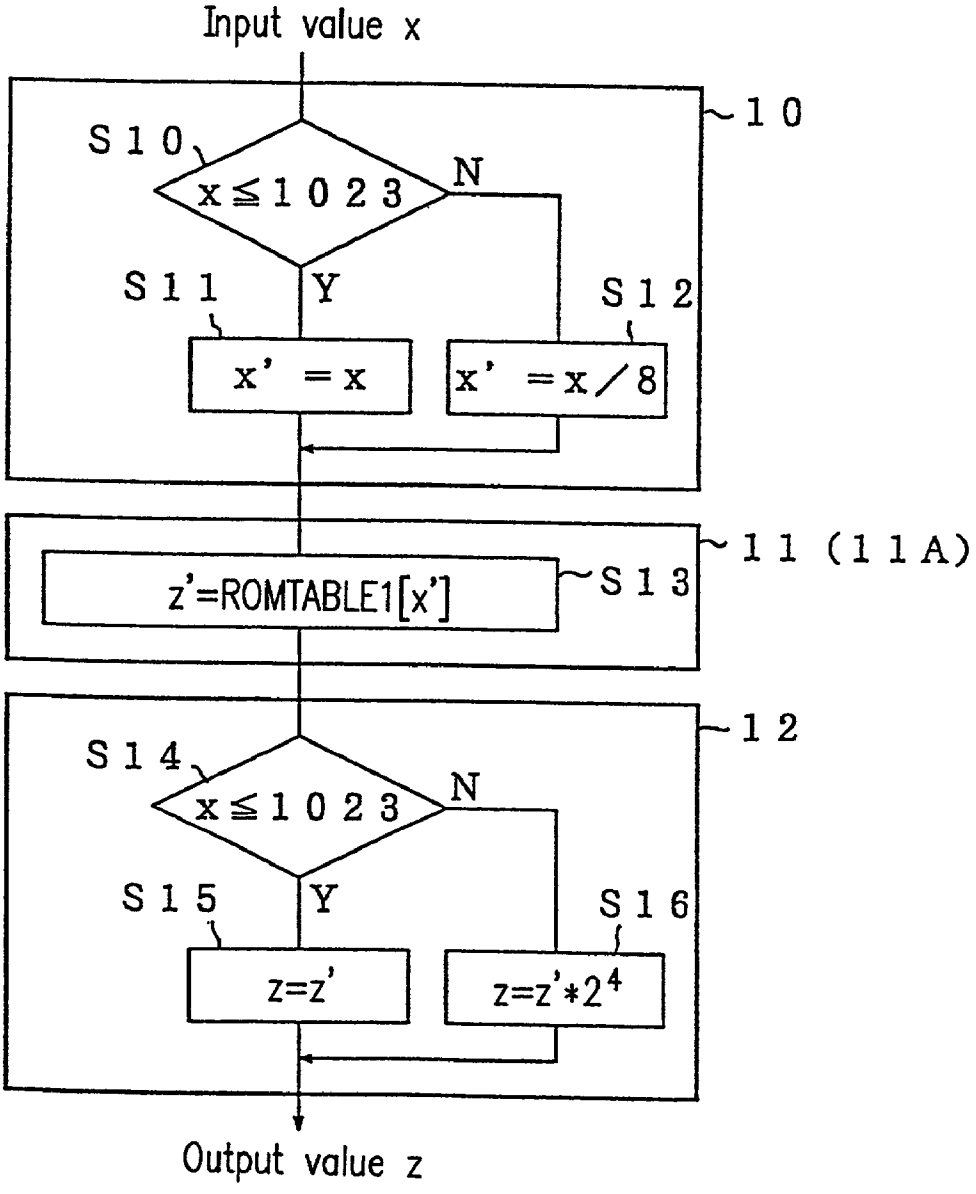
Input value x
10
S10
x≦1023
N
Y
S11
S12
x' =x
x' =x/8
11 (11A)
z'=ROMTABLE1[x']
S13
12
S14
x≦1023
N
Y
S15
S16
z=z'
z=z'*2^4
Output value z

FIG. 3

| | |
|---|---|
| 0 | 0. 000000 |
| 1 | 1. 000000 |
| 2 | 2. 519842 |
| 3 | 4. 326749 |
| 4 | 6. 349604 |
| 5 | 8. 549880 |
| 6 | 10. 902724 |
| 7 | 13. 390518 |
| 8 | 16. 000000 |
| 9 | 18. 720754 |
| | |
| 500 | 3968. 502530 |
| | |
| 1014 | 10187. 100875 |
| 1015 | 10200. 498344 |
| 1016 | 10213. 900213 |
| 1017 | 10227. 306480 |
| 1018 | 10240. 717141 |
| 1019 | 10254. 132195 |
| 1020 | 10267. 551638 |
| 1021 | 10280. 975446 |
| 1022 | 10294. 403678 |
| 1023 | 10307. 836271 |

ROM1

*FIG. 4*

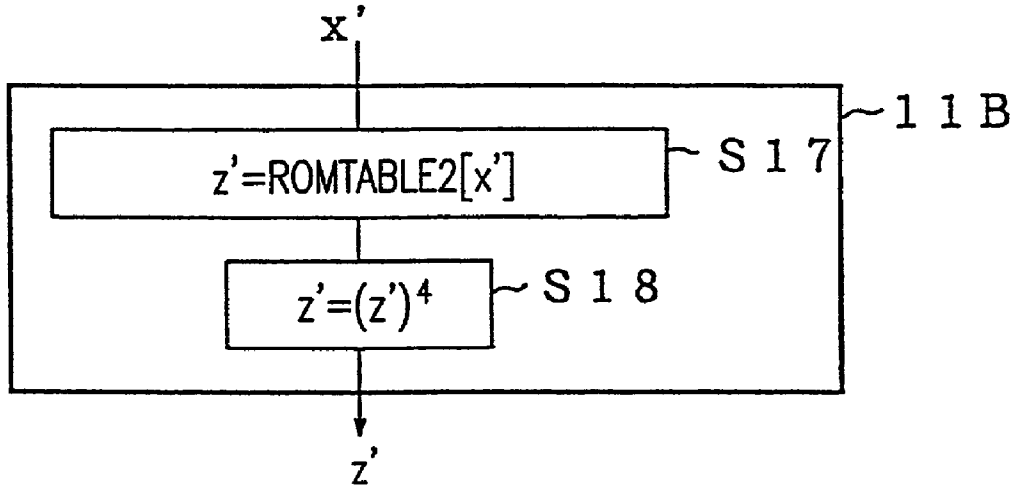

*FIG. 5*

| | |
|---|---|
| 0 | 0. 000000 |
| 1 | 1. 000000 |
| 2 | 1. 259921 |
| 3 | 1. 442250 |
| 4 | 1. 587401 |
| 5 | 1. 709976 |
| 6 | 1. 817121 |
| 7 | 1. 912931 |
| 8 | 2. 000000 |
| 9 | 2. 080084 |
| | |
| 500 | 7. 937005 |
| | |
| 1014 | 10. 046451 |
| 1015 | 10. 049752 |
| 1016 | 10. 053051 |
| 1017 | 10. 056349 |
| 1018 | 10. 059644 |
| 1019 | 10. 062936 |
| 1020 | 10. 066227 |
| 1021 | 10. 069516 |
| 1022 | 10. 072802 |
| 1023 | 10. 076086 |

ROM2

*FIG. 6*

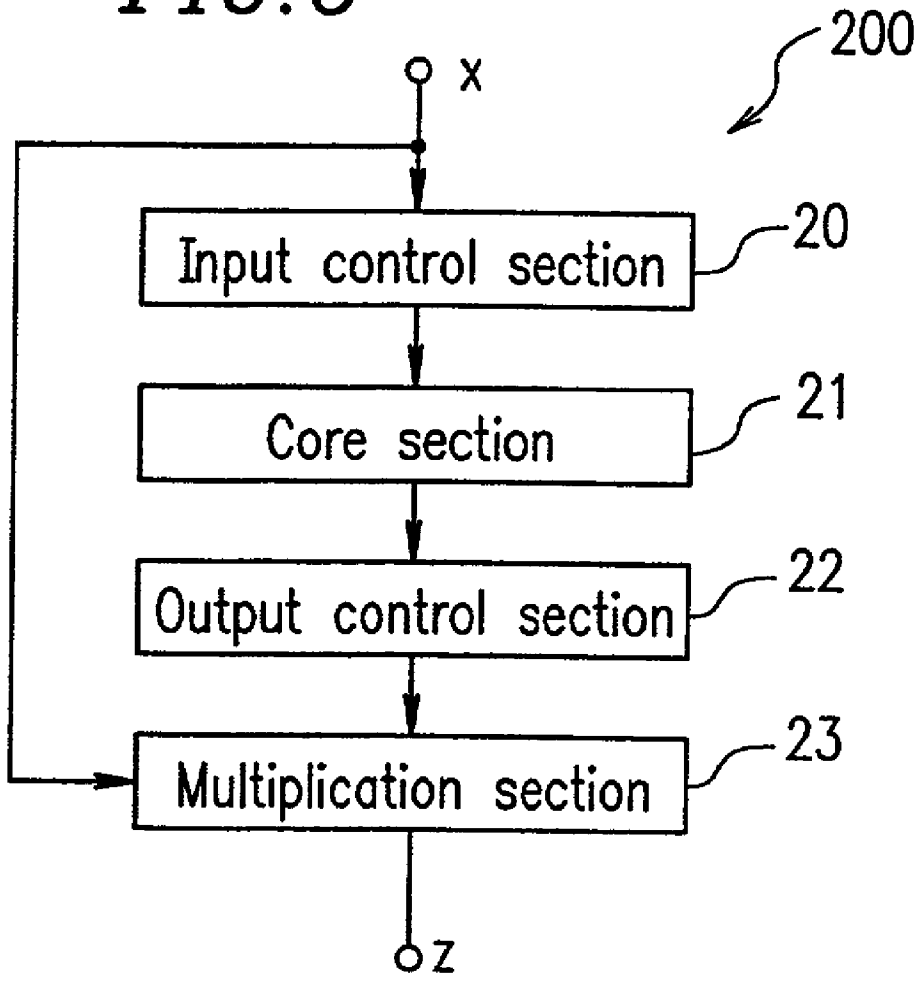
200
X
Input control section
20
Core section
21
Output control section
22
Multiplication section
23
Z

*FIG. 7*

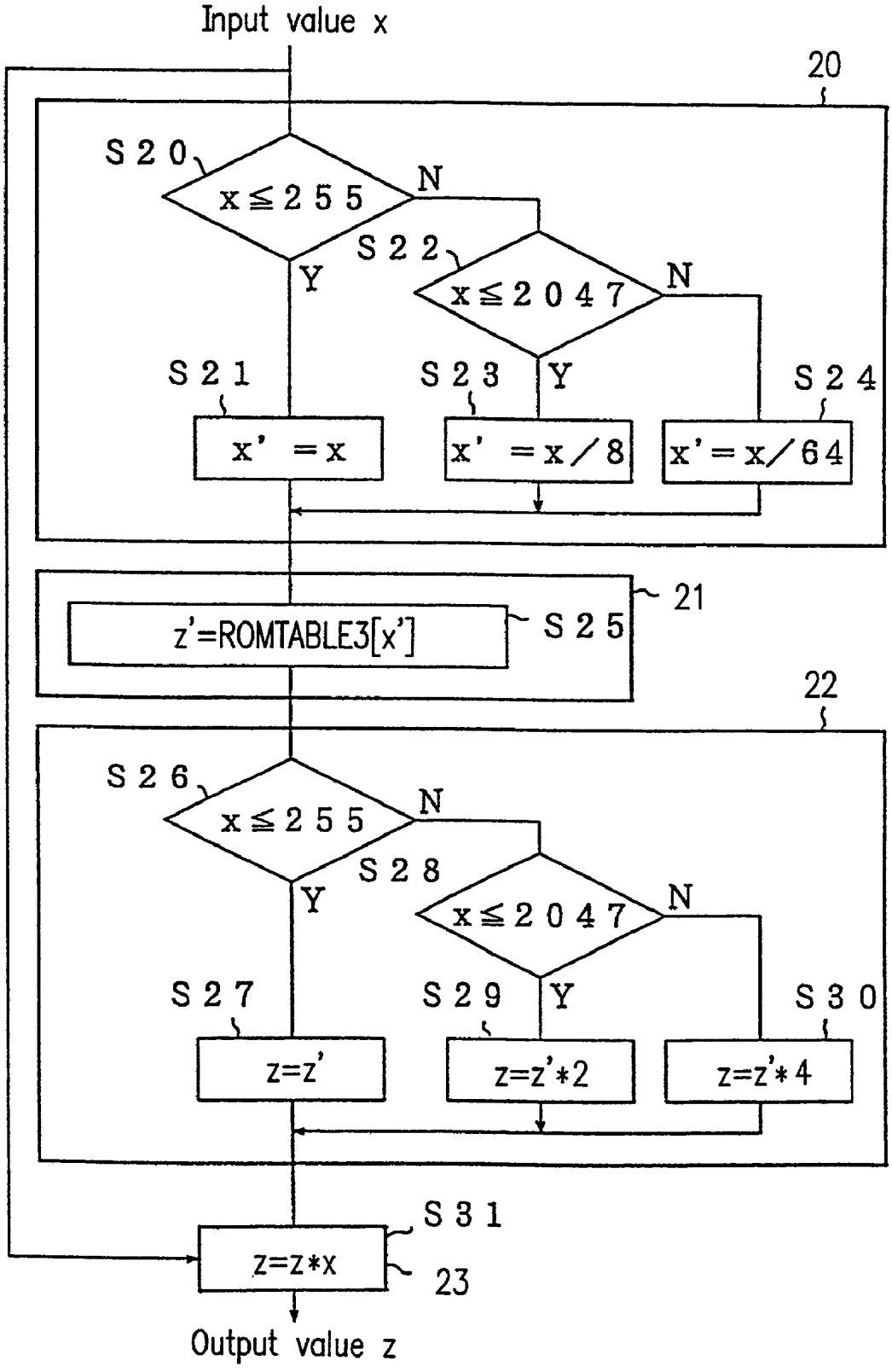
Input value x
20
S 2 0
x≦255
N
Y
S 2 2
x≦2047
N
Y
S 2 1
S 2 3
S 2 4
x' =x
x' =x/8
x'=x/64
21
z'=ROMTABLE3[x']
S 2 5
22
S 2 6
x≦255
N
Y
S 2 8
x≦2047
N
Y
S 2 7
S 2 9
S 3 0
z=z'
z=z'*2
z=z'*4
S 3 1
z=z*x
23
Output value z

FIG. 8

| | |
|---|---|
| 0 | 0. 000000 |
| 1 | 1. 000000 |
| 2 | 1. 259921 |
| 3 | 1. 442250 |
| 4 | 1. 587401 |
| 5 | 1. 709976 |
| 6 | 1. 817121 |
| 7 | 1. 912931 |
| 8 | 2. 000000 |
| 9 | 2. 080084 |
| | |
| 130 | 5. 065797 |
| | |
| 248 | 6. 282761 |
| 249 | 6. 291195 |
| 250 | 6. 299605 |
| 251 | 6. 307994 |
| 252 | 6. 316360 |
| 253 | 6. 324704 |
| 254 | 6. 333026 |
| 255 | 6. 341326 |

~ROM3

*FIG. 9*

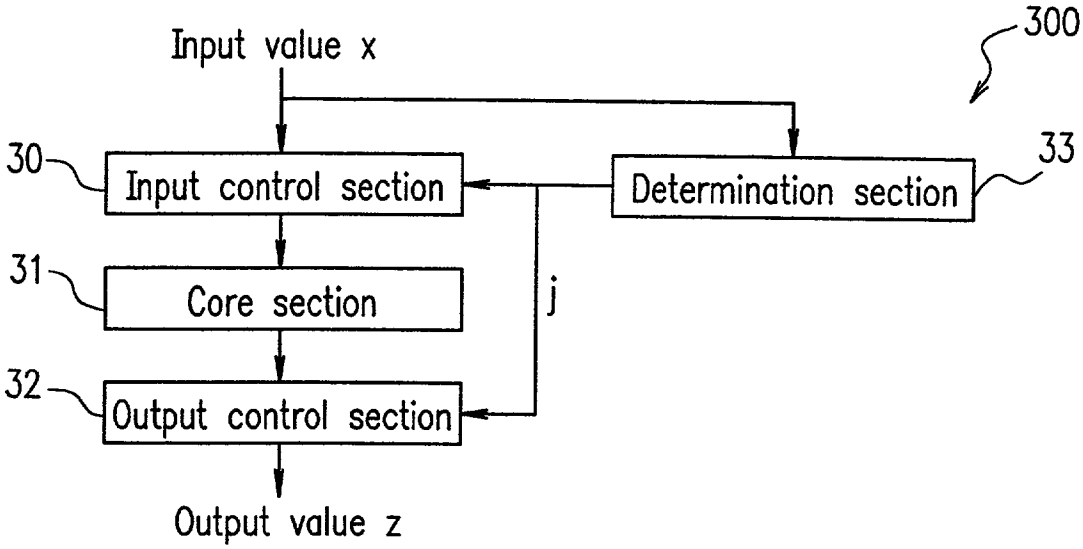
Input value x
300
30
Input control section
33
Determination section
31
Core section
j
32
Output control section
Output value z

*FIG. 10*

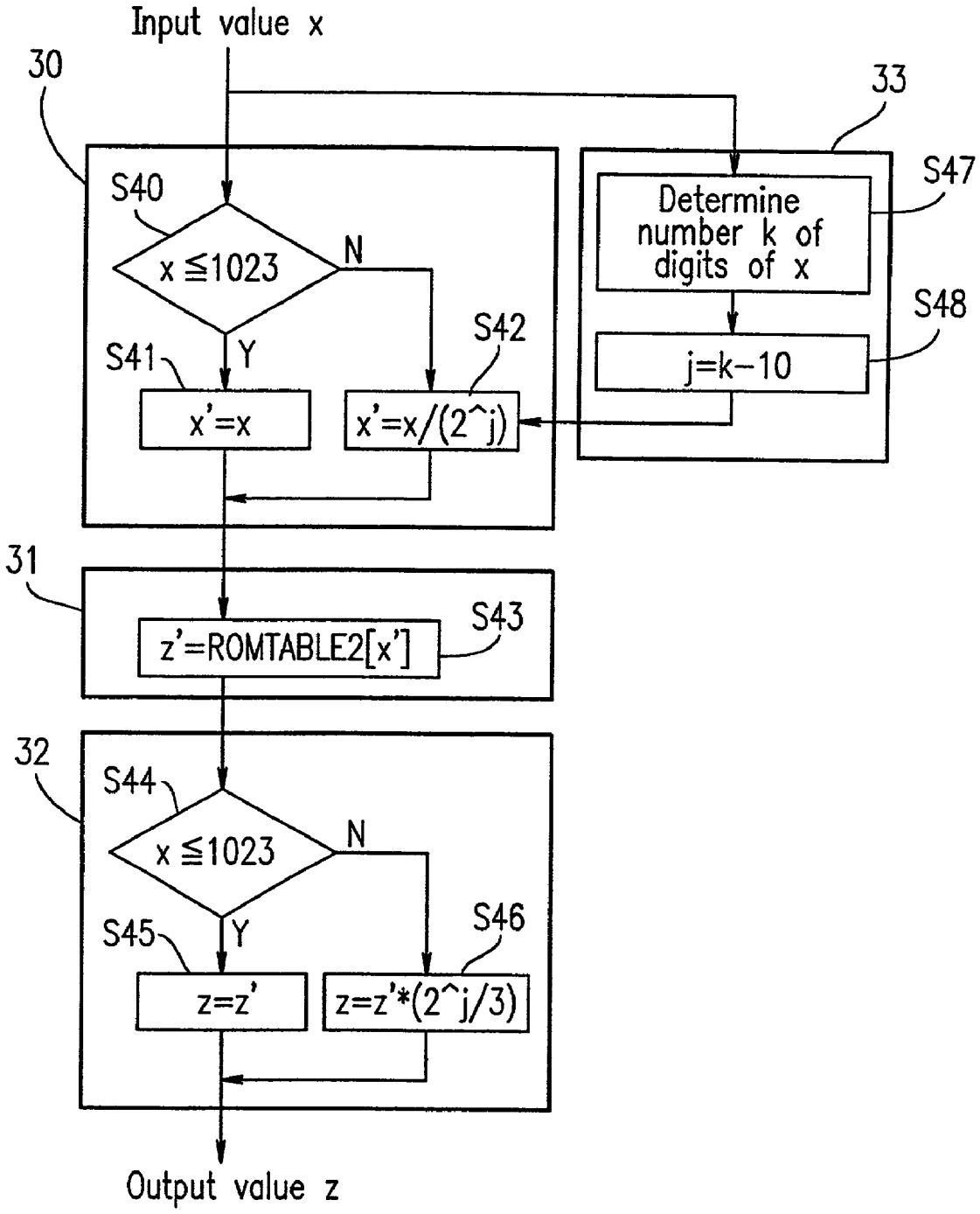
Input value x
30
33
S40
x ≦1023
N
Y
S41
x'=x
S42
x'=x/(2^j)
S47
Determine number k of digits of x
S48
j=k-10
31
S43
z'=ROMTABLE2[x']
32
S44
x ≦1023
N
Y
S45
z=z'
S46
z=z'*(2^j/3)
Output value z

*FIG. 11*

| adrs:j | data:2^(j/3) |
|---|---|
| 0 | 01000000000000000000000000000000 |
| 1 | 01010000101000101000101111100110 |
| 2 | 01100101100101111111101010010100 |
| 3 | 01111111111111111111111111111111 |

FIG. 12

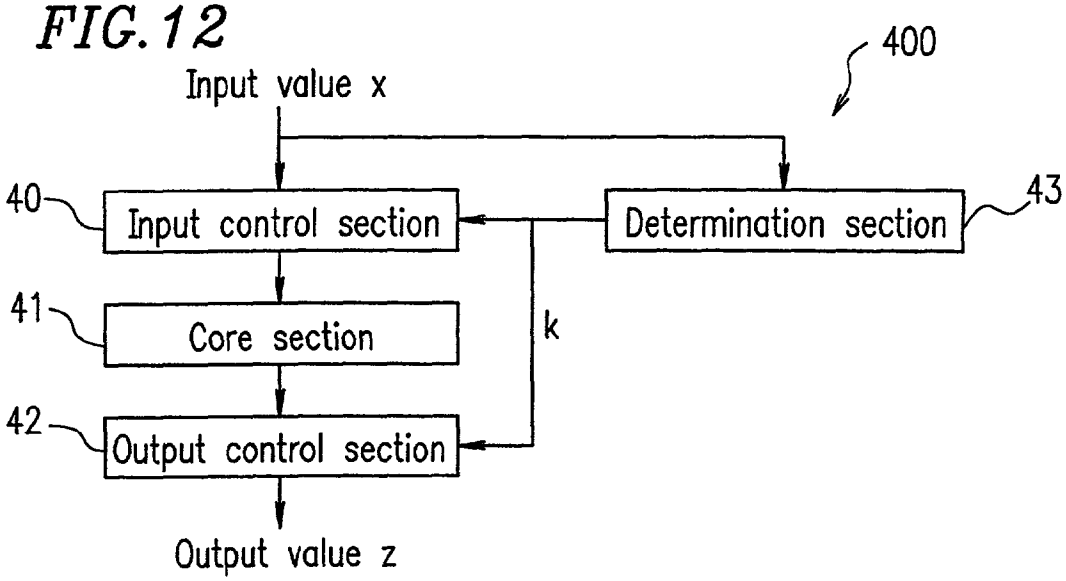
400
Input value x
40
Input control section
Determination section
43
41
Core section
k
42
Output control section
Output value z

*FIG. 13*

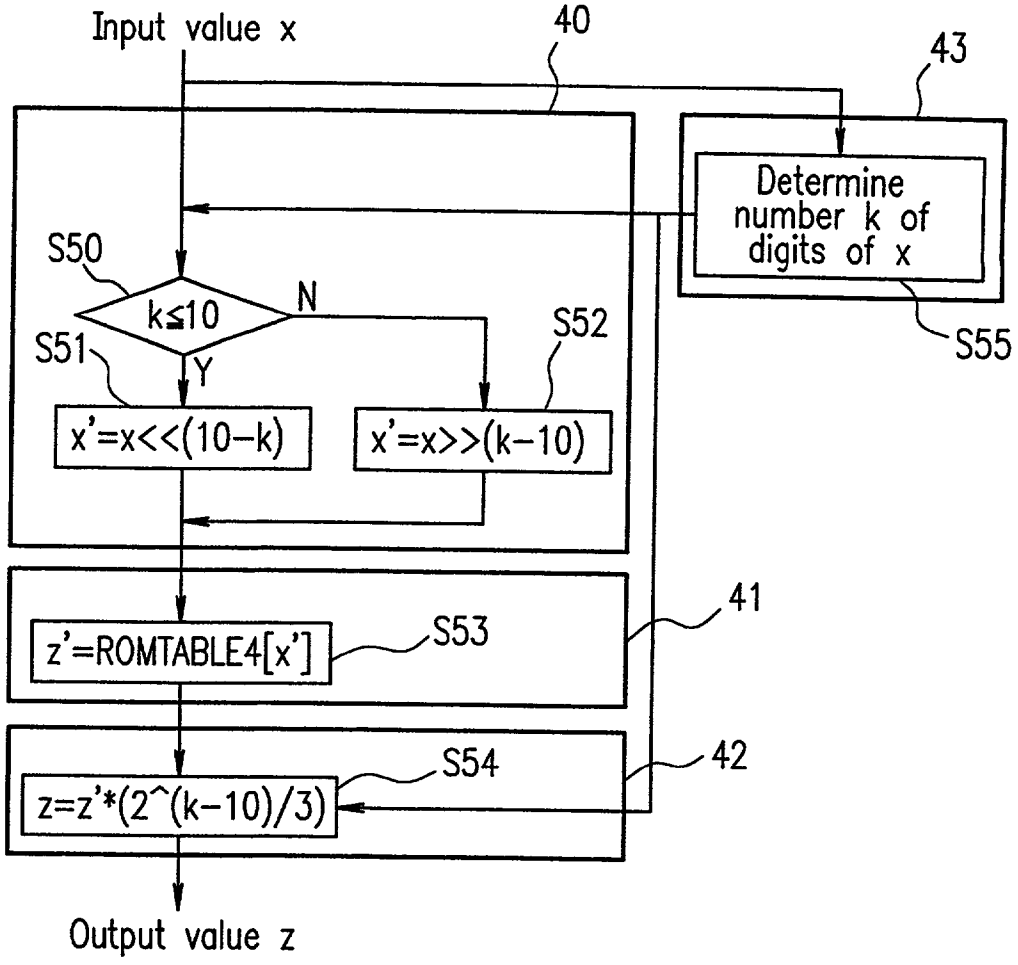
Input value x
40
43
Determine number k of digits of x
S55
S50
k≤10
N
Y
S51
S52
x'=x<<(10−k)
x'=x>>(k−10)
41
z'=ROMTABLE4[x']
S53
42
S54
z=z'*(2^(k−10)/3)
Output value z

FIG. 14

ROM4

| | |
|---|---|
| 0 | 8.000000 |
| 1 | 8.005205 |
| 2 | 8.010403 |
| 3 | 8.015595 |
| 4 | 8.020779 |
| 5 | 8.025957 |
| 6 | 8.031129 |
| 7 | 8.036293 |
| 8 | 8.041452 |
| 9 | 8.046603 |
| 10 | 8.051748 |
| | |
| 504 | 10.053051 |
| 505 | 10.056349 |
| 506 | 10.059644 |
| 507 | 10.062936 |
| 508 | 10.066227 |
| 509 | 10.069516 |
| 510 | 10.072802 |
| 511 | 10.076086 |

*FIG. 15*

| adrs:k | data:2^((k−B)/3) (B=10) |
|---|---|
| 0 | 00000110010110010111111110101001 |
| 1 | 00001000000000000000000000000000 |
| 2 | 00001010000101000101000101111100 |
| 3 | 00001100101100101111111101010010 |
| 4 | 00010000000000000000000000000000 |
| 5 | 00010100001010001010001011111001 |
| 6 | 00011001011001011111111010100101 |
| 7 | 00100000000000000000000000000000 |
| 8 | 00101000010100010100010111110011 |
| 9 | 00110010110010111111110101001010 |
| 10 | 01000000000000000000000000000000 |
| 11 | 01010000101000101000101111100110 |
| 12 | 01100101100101111111101010010100 |
| 13 | 01111111111111111111111111111111 |

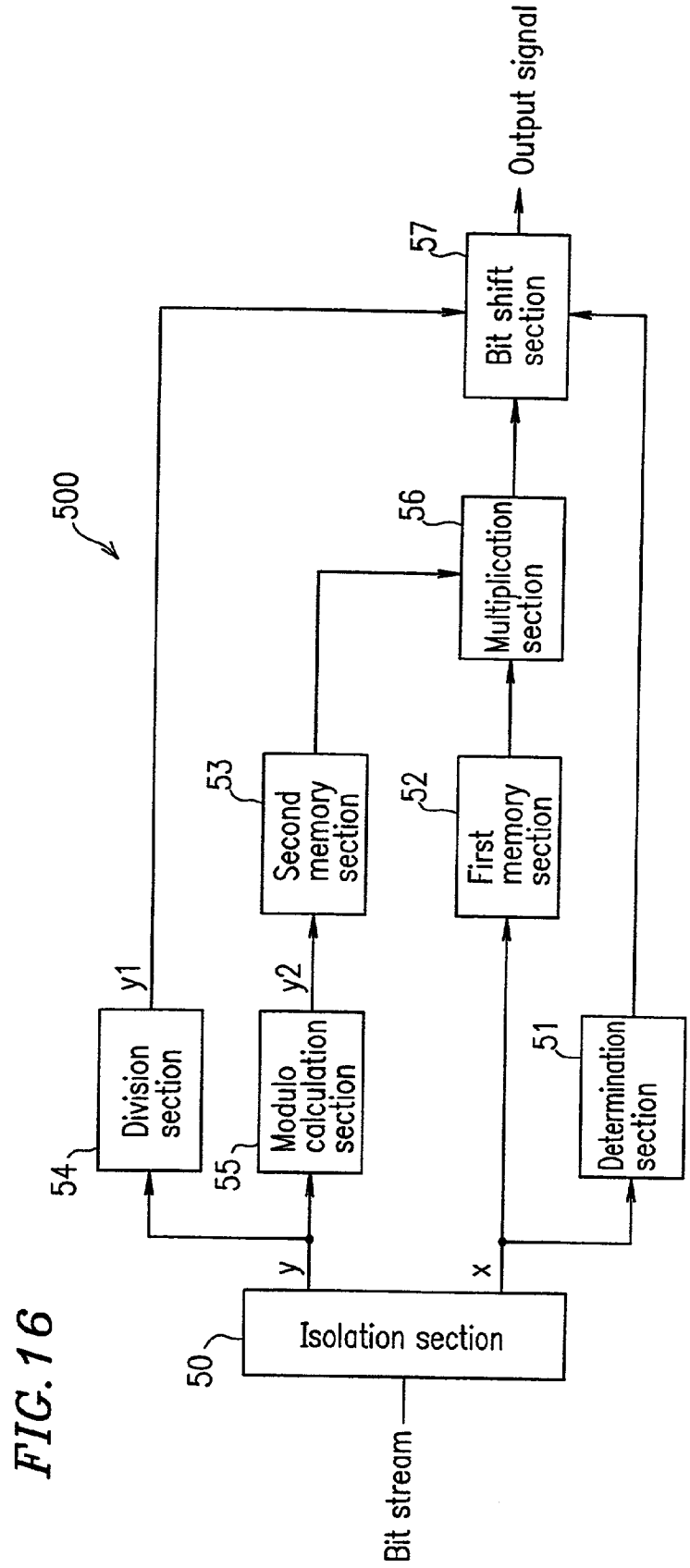
Output signal
57
Bit shift section
500
56
Multiplication section
53
Second memory section
52
First memory section
y1
y2
Division section
Modulo calculation section
Determination section
54
55
51
y
x
Isolation section
50
Bit stream

FIG. 16

*FIG. 17*

| adrs;q | data F(q) |
|---|---|
| 0 | 00000000000000000000000000000000 |
| 1 | 00000000100000000000000000000000 |
| 2 | 00000001010000010100001010000101111 |
| 3 | 00000010000101001110100101111001110 |
| 4 | 00000011001011001011111111110101100 |
| 5 | 00000100010001100110001001110101 |
| 6 | 00000101011100111000110001110010 |
| 7 | 00000110101100011111110010000000 |
| 8 | 00000111111111111111111111111111 |
| 9 | 00001001010111000100000110101110 |
| 10 | 00001010110001011010110100101000 |
| 11 | 00001100001110110101110100101100 |
| 12 | 00001101101111001000111111101001 |
| 13 | 00001111010010001001111011110100 |
| 14 | 00010000110111101111100110010101 |
| 15 | 00010010011111111001000001000111 |
| 16 | 00010100001010001010001011111001 |
| 17 | 00010101110110110000110111110110 |
| 18 | 00010111100101100011000000101100 |
| 19 | 00011001010110011000110110000100 |
| 20 | 00011011001001001110100010111010 |
| 21 | 00011100111110111111111001111001 |
| 22 | 00011110110100101000101011110001 |
| 23 | 00100000101101000101100000101001 |
| 24 | 00100010100111010010111001101101 |
| 25 | 00100100100011001101101101010110 |
| 26 | 00100110100000011001011111110011001 |
| 27 | 00101000001111111111111111111111 |
| 28 | 00101010100000011001000101000011 |
| 29 | 00101100100011000111000010100111 |
| 30 | 00101110100110111110001011101011 |
| 31 | 00110000101100001111111111001100 |
| 32 | 00000000000001100101100101111111 |
| 33 | 00000000000001101001110110010100 |
| 34 | 00000000000001101110001001011001 |
| 35 | 00000000000001110010011111100110 |
| ≈ | ≈ |
| 8186 | 00101000010001110011000111110010 |
| 8187 | 00101000010010001101111111100111 |
| 8188 | 00101000010010101000110111100000 |
| 8189 | 00101000010011000011101111011110 |
| 8190 | 00101000010011011110100111100000 |
| 8191 | 00101000010011111100101111111100111 |

*FIG. 18*

| adrs;p | data G(p) |
|---|---|
| 0 | 01000000000000000000000000000000 |
| 1 | 01001100000110111111100000101000 |
| 2 | 01011010100000100111100110011001 |
| 3 | 01101011101000100111111001100101 |

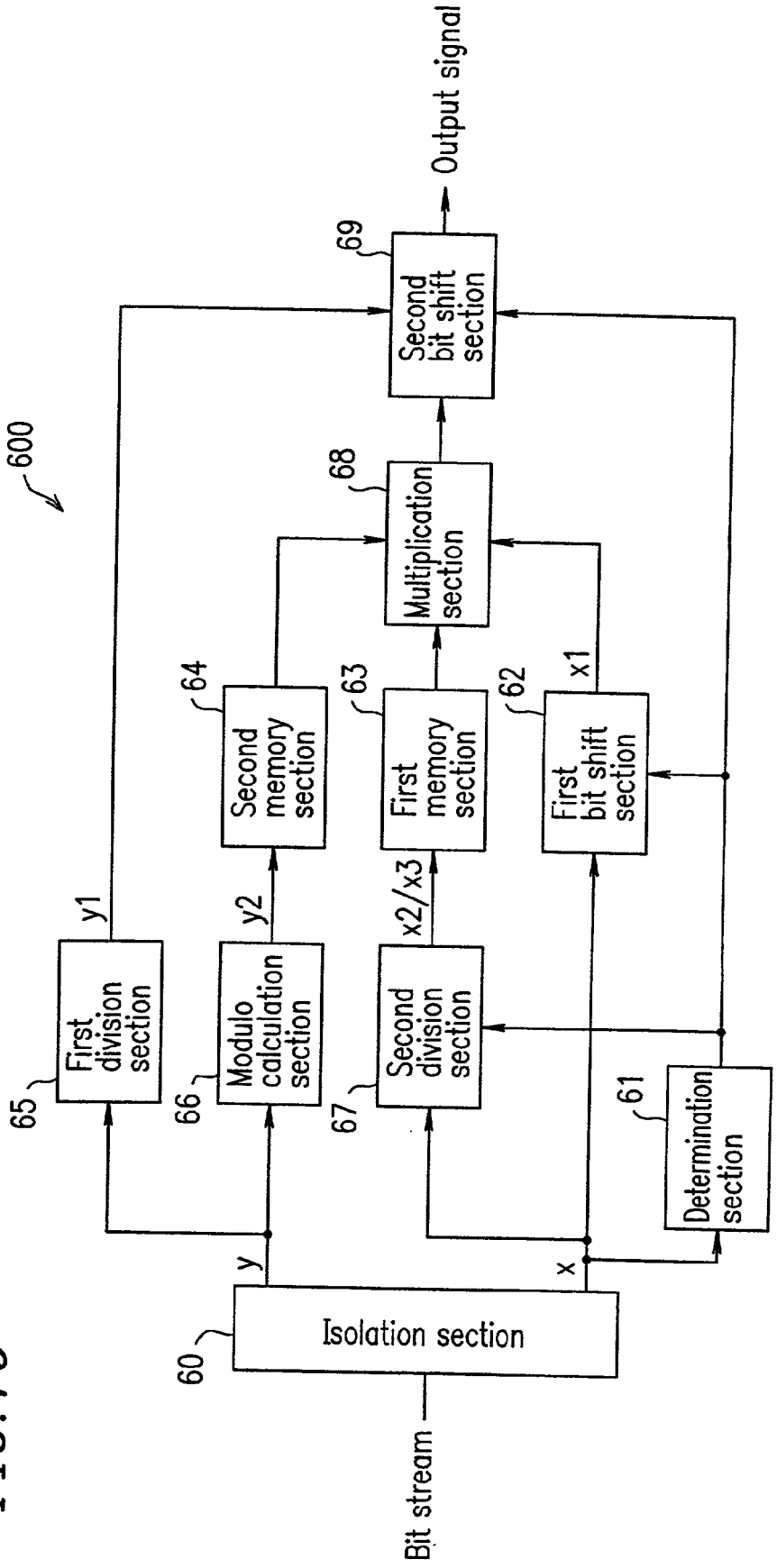
Output signal
69
Second bit shift section
600
68
Multiplication section
64
Second memory section
63
First memory section
62
First bit shift section
x1
y1
y2
x2/x3
First division section
Modulo calculation section
Second division section
65
66
67
61
Determination section
y
x
60
Isolation section
Bit stream

FIG. 19

*FIG. 20*

| adrs;q | data H(q) |
|---|---|
| 0 | 0000000000000000000000000000000000 |
| 1 | 0000100000000000000000000000000000 |
| 2 | 0000101000001010000101000010111111100 |
| 3 | 0000101111000100111011101000100100 |
| 4 | 0000110010110010111111111010100010 |
| 5 | 0000111011010111000000011111011111 0 |
| 6 | 0000111010000100101110110100000101 |
| 7 | 0000111110100110110101110110111101 |
| 8 | 0001000000000000000000000000000000 |
| 9 | 0001000010100100000000101111111100 |
| 10 | 0001000100111100010010000100000 1 |
| 11 | 0001000111001010101101100001000 10 |
| 12 | 0001001001010000101111111111100001 |
| 13 | 0001001011001111110001000100011111 |
| 14 | 0001001101000111111111000101010 10 |
| 15 | 0001001110111010110011010110010 1 |
| 16 | 0001010000010100001010000101111110 01 |
| 17 | 0001010010010001111111100000010101 |
| 18 | 0001010011111011101000111010000011 |
| 19 | 0001010101011000111000101111101 10 |
| 20 | 0001010110110111001000001001010 1 |
| 21 | 0001011000001001001000110110101 10 |
| 22 | 0001011001101010100100111001 1000 |
| 23 | 0001011011000000000111101010 10100 |
| 24 | 0001011100010011011101000100 1001 |
| 25 | 0001011110110010001100011011010 01 |
| 26 | 0001011110110011001100010010 00011 |
| 27 | 0001100000000000000000000000000000 |
| 28 | 0001100001001010111011110010 1000 |
| 29 | 0001100010010100000110101101 11000 |
| 30 | 0001100011011011100111001011 0111 |
| 31 | 0001100100100000110001100001011 01 |
| 32 | 0001100101100101111111110101001 01 |
| 33 | 0001100110101001000001111100000 10 |
| 34 | 0001100111101010101110011001 0111 |
| 35 | 0001101000101011001001001101 0000 |
| 499 | 0011111110111010000100100100 10011 |
| 500 | 0011111110111111101111111001001 11101 |
| 501 | 0011111111000100111010000111110011 |
| 502 | 0011111111001010010100001100 11001 |
| 503 | 0011111111001111110111011100 10011 |
| 504 | 0011111111010101010001101111111 00100 |
| 505 | 0011111111011010011111110110001 101 |
| 506 | 0011111111011111111011111111 00101 00 |
| 507 | 0011111111110010100111111011111 11010 |
| 508 | 0011111111110101010011100011000011 |
| 509 | 0011111111110111111111011111111 10010 |
| 510 | 0011111111111010101010001110001 010 |
| 511 | 0011111111111010101010011100011 01 |
| 512 | 0011111111111111111111111111111111 |

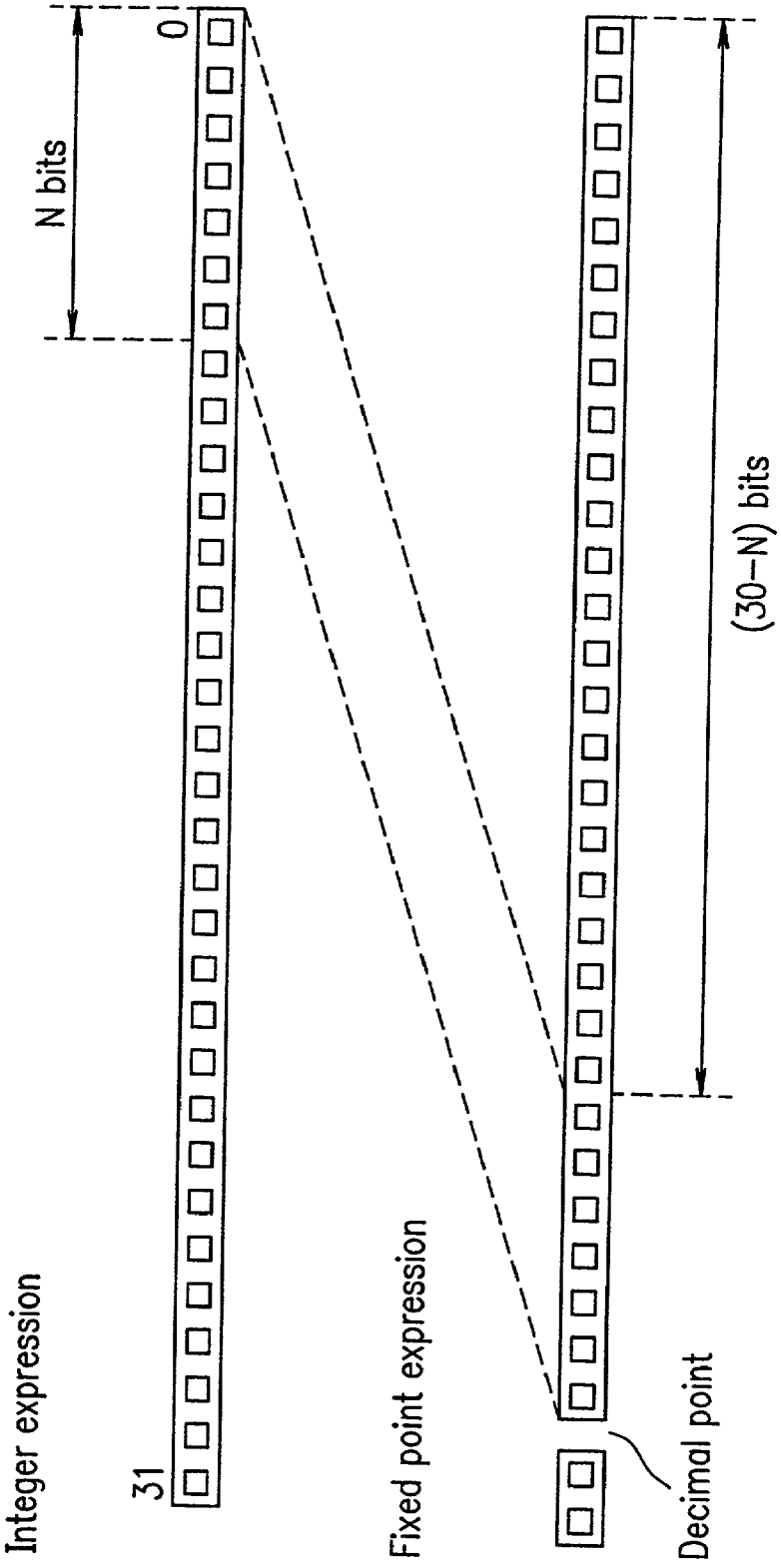
N bits
0
31
Integer expression
Fixed point expression
(30−N) bits
Decimal point

FIG. 21

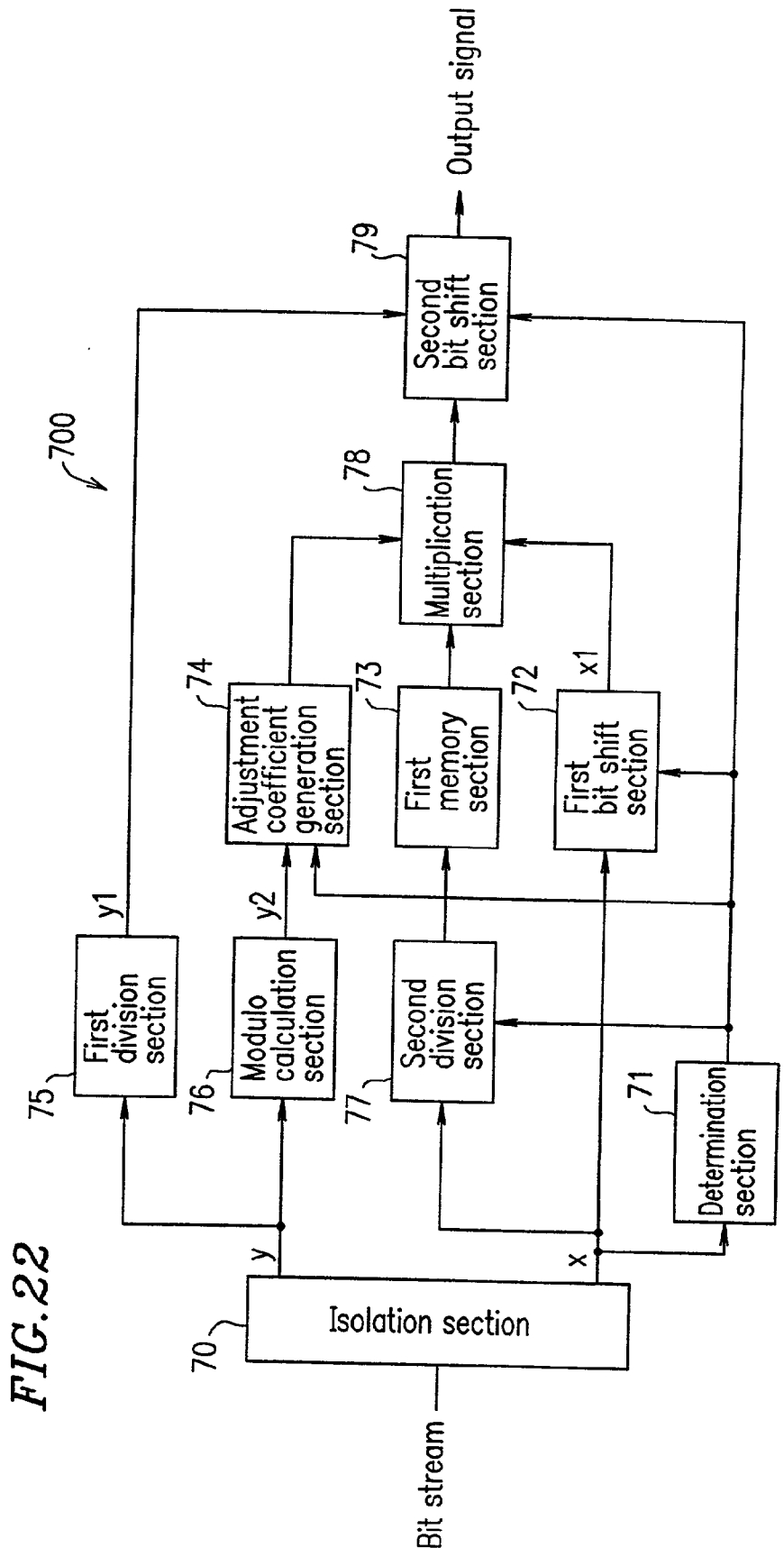
Output signal
79
Second bit shift section
700
78
Multiplication section
74
Adjustment coefficient generation section
73
First memory section
72
First bit shift section
x1
y1
y2
First division section
Modulo calculation section
Second division section
75
76
77
71
Determination section
y
x
70
Isolation section
Bit stream

FIG. 22

*FIG. 23*

| adrs;(p,j) | data I(p,k) |
|---|---|
| (0,0) | 00010000000000000000000000000000 |
| (0,1) | 00010100001010001010001011111001 |
| (0,2) | 00011001011001011111111010100101 |
| (0,3) | 00100000000000000000000000000000 |
| (0,4) | 00101000010100010100010111110011 |
| (1,0) | 00010011000001101111111000001010 |
| (1,1) | 00010111111111001000100001101011 1 |
| (1,2) | 00011111000110100001101111111100111 |
| (1,3) | 00100110000011011111111000001010 0 |
| (1,4) | 00101111111110010001000011010111 0 |
| (2,0) | 00010110101000001001111001100110 |
| (2,1) | 00011100100000100011111000000111 |
| (2,2) | 00100011111101011001101011000011 1 |
| (2,3) | 00101101010000010011111001100110 0 |
| (2,4) | 00111001000001000111110000001110 |
| (3,0) | 00011010111010001001111111001100 1 |
| (3,1) | 00100001111001110001111110010010 1 |
| (3,2) | 00101010101101110000001000010001 |
| (3,3) | 00110101110100010011111100110010 |
| (3,4) | 01000011111001110001111100100101 1 |

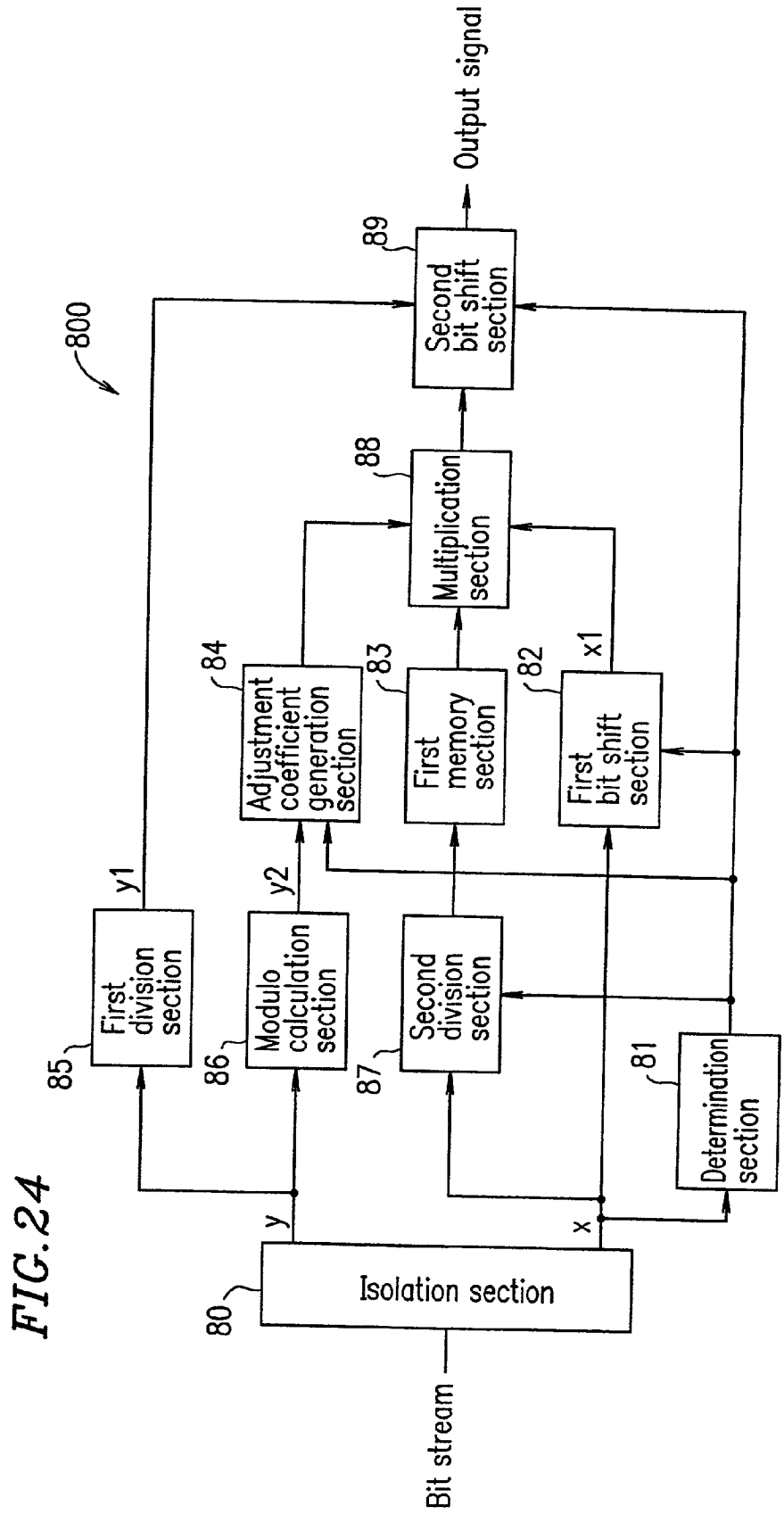
800
Output signal
89
Second bit shift section
88
Multiplication section
84
Adjustment coefficient generation section
83
First memory section
82
First bit shift section
x1
y1
y2
First division section
Modulo calculation section
Second division section
85
86
87
81
Determination section
y
x
80
Isolation section
Bit stream

FIG. 24

FIG.25

| adrs;(p,j) | data I (p,j) |
|---|---|
| (0,1) | 0000001010000101000101000101111 |
| (0,2) | 0000001110010110010111111110101100 |
| (0,3) | 0000010000000000000000000000000000 |
| (0,4) | 0000010101000010100010100010111110 |
| (0,5) | 0000011001011001011111111101010001 |
| (0,6) | 0000100000000000000000000000000000 |
| (0,7) | 0000101000001010001010001011111100 |
| (0,8) | 0000110010110010111111111010100010 |
| (0,9) | 0001000000000000000000000000000000 |
| (0,10) | 0001010000010100010100010111111001 |
| (0,11) | 0001100101100101111111110101001101 |
| (0,12) | 0010000000000000000000000000000000 |
| (0,13) | 0010100001010001010001011111100011 |
| (1,1) | 0000001011111111110010001000011010 |
| (1,2) | 0000001111000110100001101111111100 |
| (1,3) | 0000010011000001101111111100000010 |
| (1,4) | 0000010111111111001000100001101101 |
| (1,5) | 0000011110001101000011011111111001 |
| (1,6) | 0000100110000011011111111000000101 |
| (1,7) | 0000101111111110010001000001101011 |
| (1,8) | 0000111110001101000011011111110011 |
| (1,9) | 0001001100000110111111110000001010 |
| (1,10) | 0001011111111100100010000011010111 |
| (1,11) | 0001111000110100001101111111001111 |
| (1,12) | 0010011000001101111111100000010100 |
| (1,13) | 0010111111111001000100001101011110 |
| (2,1) | 0000001110010000010001111100000000 |
| (2,2) | 0000010001111110101100110101100000 |
| (2,3) | 0000010110101000000100111100110001 |
| (2,4) | 0000011100100000010001111100000001 |
| (2,5) | 0000100011111101011001101011000001 |
| (2,6) | 0000101101010000001001111001100011 |
| (2,7) | 0000111001000000100011111000000011 |
| (2,8) | 0001000111111010110011010110000011 |
| (2,9) | 0001011010100000100111100110011010 |
| (2,10) | 0001110010000010001111100000000111 |
| (2,11) | 0010001111110101100110101100000111 |
| (2,12) | 0010110101000000100111100110011100 |
| (2,13) | 0011100100000010001111100000001110 |
| (3,1) | 0000010000011110011100011111100100 |
| (3,2) | 0000010101010110111000000010000010 |
| (3,3) | 0000011010111010001001111111001110 |
| (3,4) | 0000100001111001110001111110010010 |
| (3,5) | 0000101010101101110000000100001000 |
| (3,6) | 0000110101110100010011111100110010 |
| (3,7) | 0001000011111001110001111100100010 |
| (3,8) | 0001010101011011100000010000010000 |
| (3,9) | 0001101011101000100111111100110010 |
| (3,10) | 0010000111110011100011111100100101 |
| (3,11) | 0010101010110111000000010000010001 |
| (3,12) | 0011010111010001001111111001100010 |
| (3,13) | 0100001111100111000111111001001011 |

FIG.26

| adrs;q | data H(q) |
|---|---|
| 0 | 00110010110010111111111010100101 0 |
| 1 | 00110010110111001110011001010010 |
| 2 | 00110010111011011100010000100011 |
| 3 | 00110010111111110100101101100111 1 |
| 4 | 00110011000011110101111001101001 |
| 5 | 00110011001000000000110110000001 00 |
| 6 | 00110011001100000110011001010111 1 |
| 7 | 00110011010000001011100110111111 11 |
| 8 | 00110011010100100000011111000001 00 |
| 9 | 00110011011000101010000011010000 |
| 10 | 00110011011100110010011101110011 |
| 11 | 00110011100000011101000111000000 00 |
| 12 | 00110011100101000000101010000001 10 |
| 13 | 00110011101001000111111000001011 1 |
| 14 | 00110011101101001101100011000011 |
| 15 | 00110011111000101001010110001101 1 |
| ≈ | ≈ ≈ |
| 236 | 00111111100100111110100111011010 1 |
| 237 | 00111111100110010110001011111111 00 |
| 238 | 00111111100111101101101000111101 0 |
| 239 | 00111111010010001001111100110010 |
| 240 | 00111111101010011100001100010011 1 |
| 241 | 00111111101011110011010010101110 0 |
| 242 | 00111111101101001010010001101010 0 |
| 243 | 00111111101110100001001001001001 1 |
| 244 | 00111111101111111011111110010011 101 |
| 245 | 00111111111000100111010000111110 011 |
| 246 | 00111111111001010010100000110011 001 |
| 247 | 00111111111001111110110111010010 011 |
| 248 | 00111111111010101000110111111100 100 |
| 249 | 00111111111011010011111110110001 101 |
| 250 | 00111111111011111111011111111100 10100 |
| 251 | 00111111111100101001111101111111 010 |
| 252 | 00111111111101010100111000110000 11 |
| 253 | 00111111111101111111110111111111 10010 |
| 254 | 00111111111111010101010001110001 010 |
| 255 | 00111111111111010101010011100011 01 |

FIG. 27

| | |
|---|---|
| 0 | 0. 000000 |
| 1 | 1. 000000 |
| 2 | 2. 519842 |
| 3 | 4. 326749 |
| 4 | 6. 349604 |
| 5 | 8. 549880 |
| 6 | 10. 902724 |
| 7 | 13. 390518 |
| 8 | 16. 000000 |
| 9 | 18. 720754 |
| ⋮ | ⋮ |
| 8180 | 164817. 910834 |
| 8181 | 164844. 776566 |
| 8182 | 164871. 643393 |
| 8183 | 164898. 511315 |
| 8184 | 164925. 380331 |
| 8185 | 164952. 250442 |
| 8186 | 164979. 121647 |
| 8187 | 165005. 993946 |
| 8188 | 165032. 867339 |
| 8189 | 165059. 741827 |
| 8190 | 165086. 617408 |
| 8191 | 165113. 494083 |

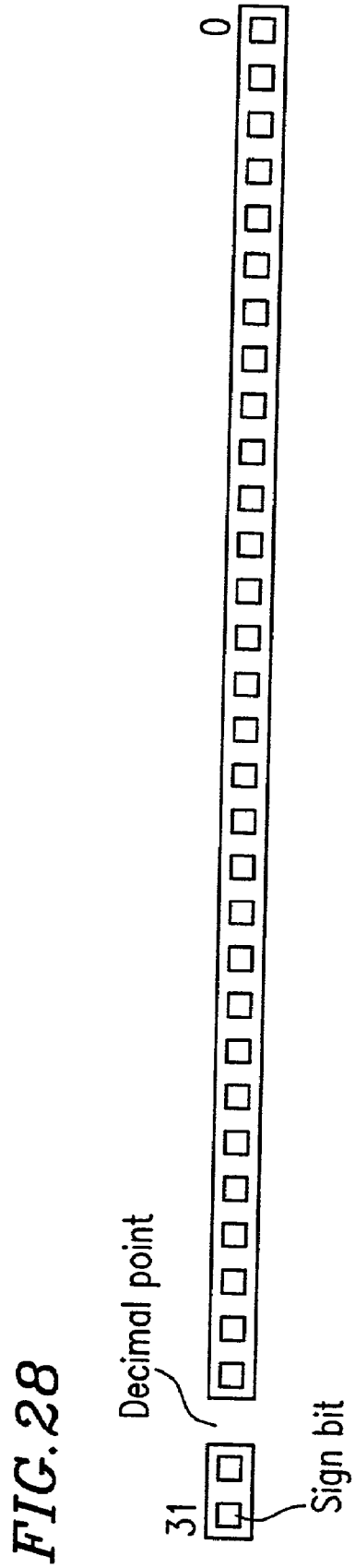


FIG. 28

*FIG. 29*

| adrs;x | data f(x) |
|---|---|
| 0 | 0000000000000000000000000000000000 |
| 1 | 0000000000000000000010000000000000 |
| 2 | 0000000000000000000010100001010001 |
| 3 | 0000000000000000000100010100111010 |
| 4 | 0000000000000000000110010110010111 |
| 5 | 0000000000000000000100010001100110 |
| 6 | 0000000000000000000101011100111000 |
| 7 | 0000000000000000000110101100011111 |
| 8 | 0000000000000000000111111111111111 |
| 9 | 0000000000000000000100101011100001000 |
| 10 | 0000000000000000000101011000101101 |
| 11 | 0000000000000000000110000111011010 |
| 12 | 0000000000000000000110110111100100 |
| 13 | 0000000000000000000111101001000100 |
| 14 | 0000000000000000000100001101111011 |
| 15 | 0000000000000000000100100111111100 |
| 16 | 0000000000000000000101000010100001 |
| 17 | 0000000000000000000101011101101100 |
| 18 | 0000000000000000000101111001011000 |
| 19 | 0000000000000000000110010101100110 |
| 20 | 0000000000000000000110110010010001 |
| 21 | 0000000000000000000111001111011111 |
| 22 | 0000000000000000000111101101001010 |
| 23 | 0000000000000000000100000101101000 |
| 24 | 0000000000000000000100010100111010 |
| 25 | 0000000000000000000100100100011001 |
| 26 | 0000000000000000000100110100000110 |
| 27 | 0000000000000000000101000011111111 |
| 28 | 0000000000000000000101010100000110 |
| 29 | 0000000000000000000101100100011000 |
| 30 | 0000000000000000000101110100110111 |
| 31 | 0000000000000000000110000101100000 |
| 32 | 0000000000000000000110010110010111 |
| 33 | 0000000000000000000110100111011001 |
| 34 | 0000000000000000000110111000100101 |
| 35 | 0000000000000000000111001001111100 |
| ≈ | ≈ |
| 8186 | 0010100001000111001100011111001 |
| 8187 | 0010100001001000110111111100111 |
| 8188 | 0010100001001010100011011111000 |
| 8189 | 0010100001001100001110111101111 |
| 8190 | 0010100001001101111010011111000 |
| 8191 | 0010100001001111100101111111001 |

FIG. 30

| adrs;y%4 | data g(y) |
|---|---|
| 0 | 0100000000000000000000000000000000 |
| 1 | 0100110000011011111110000010100 0 |
| 2 | 0101101010000010011110011001100 1 |
| 3 | 0110101110100010011111100110010 1 |

# EXPONENTIAL CALCULATION DEVICE AND DECODING DEVICE

## BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to an exponential calculation device and a decoding device for performing a predetermined calculation for a given encoded bit stream. More particularly, the present invention relates to an exponential calculation device for calculating x̂ (a/b) and a decoding device for calculating x̂ (a/b) *2 (y/c) for given integer values x and y contained in a bit stream which has been encoded based on the MPEG (moving picture experts group) 2 audio encoding standard "AAC".

[0003] 2. Description of the Related Art

[0004] Conventionally, in order to calculate z=x̂ (a/b) for a given value of x (where a and b are constant values), values of x̂ (a/b) for all the possible values of x are stored in a table whose addresses are associated with the respective values of x. The value of z for a given value of x is obtained by reading out the value stored in the table.

[0005] In this specification, "x to a power of (a/b)" is represented as "x̂ (a/b)", where the symbol "^" denotes the use of an exponential function, and the symbol "/" denotes a division function. The symbol "*" will also be used herein to denote a multiplication function.

[0006] **FIG. 27** illustrates a table containing values of x̂ (4/3) for values of x in a range of 0≦x≦8191. A decoding operation based on the MPEG 2 audio encoding standard AAC requires calculation of x̂ (4/3) for a value of x in the range from 0 to 8191. When the conventional method of obtaining the value of x̂ (a/b) is employed in the exponential calculation involved in decoding data encoded based on the MPEG 2 audio encoding standard AAC, an input value of x is used as an address for reading out the corresponding value from a table provided in advance, as illustrated in **FIG. 27**, to obtain the value of x̂ (4/3) an output value z.

[0007] In order to decode an encoded bit stream so as to obtain the intended data, it is necessary to perform a predetermined calculation process. A bit stream encoded based on the MPEG 2 audio encoding standard AAC contains a quantization code "x" and a scaling factor "y". The quantization code "x" and the scaling factor "y" are both integers. In the case of the MPEG 2 audio encoding standard AAC, the decoding device calculates x̂ (a/b)*2 (y/c) (where a,b and c are constants) for given values of x and y. Details of the MPEG 2 audio encoding standard AAC are disclosed in ISO/IEC 13818-7:1997, pp.57-59.

[0008] Conventionally, an operation of calculating x̂ (a/b)*2 (y/c) for given values of x and y has been performed as follows with a DSP (digital signal processor) of a fixed point format.

[0009] **FIG. 28** illustrates an exemplary 32-bit fixed point format. Referring to **FIG. 28**, where the bit **0** is the least significant bit and the bit **31** is the most significant bit, the decimal point is fixed between the bit **30** and the bit **29**. In particular, bit **30** represents the integral portion of a value, while the bits **29-0** represent the fractional portion of the value. Bit **31** is the sign bit indicating whether the value is positive or negative. Such a numeric representation with a fixed decimal point is called a "fixed point format".

[0010] For example, a value "f" represented in the 32-bit fixed point format, as illustrated in **FIG. 28**, can vary in the range of −2≦f<2. Then, in order to perform a calculation with a fixed-point-format processing unit, such as a DSP, values involved in the calculation are adjusted so that they are each in the range of +2.0 before performing the intended calculation.

[0011] In the MPEG **2** audio encoding standard AAC, x can vary in the range of 0≦x<8192, whereas y can vary in the range of −100≦y<156. Consider a calculation of Expression (1) below for x and y having such ranges.

$$\hat{x}\ (4/3)*2\ (y/4) \tag{1}$$

[0012] Expression (1) may be calculated as follows based on the prior art technique with a fixed-point-format processing unit. First, Expression (1) is converted to Expression (2) below.

$$\hat{x}\ (4/3)*2\ ((y\%4)/4)*2\ \mathrm{int}(y/4) \tag{2}$$

[0013] In Expression (2), "(y%4)" represents a remainder of a division y/4, and "int(y/4)" represents a quotient (integer) of the division y/4. In the calculation of Expression (2), x̂ (4/3) is first calculated. The calculation x̂ (4/3) is typically performed using a ROM table. In particular, values of x̂ (4/3) for various values of x are calculated and stored in a table in advance. Then, the value of x̂ (4/3) for a given value of x is obtained by referencing the ROM table using the value of x as the address of the table.

[0014] The calculation of Expression (1) or (2) assumes the use of a fixed-point-format processing unit. In particular, different values of x̂ (4/3) are normalized and stored in a table in advance, with the maximum value thereof being normalized to a value which does not exceed 1.0.

[0015] **FIG. 29** illustrates the contents of a first table for use with a conventional decoding device. The table illustrated in **FIG. 29** contains values of f(x) for the range of0≦x<8192 at addresses which correspond to the respective values of x. The values of f(x) are obtained by first calculating x̂ (4/3) for various values of x, and then dividing the calculated values by 2-18 to normalize the values so that none of them exceeds 1.0.

[0016] In the calculation of Expression (2), 2 ((y%4)/4) is then calculated. Typically, 2 ((y%4)/4) is also calculated using a ROM table. In particular, different values of 2 ((y%4)/4) for various integers (y%4) are calculated and stored in a table in advance. Then, an integer (y%4) is used as an address for referencing the table so as to obtain the value of 2 ((y%4)/4) for the integer (y%4).

[0017] **FIG. 30** illustrates the contents of a second table for use with a conventional decoding device. The second table illustrated in **FIG. 30** contains values of g(y) for the address values (y%4) in the range from 0 to 3. The values of g(y) are obtained by normalizing values of 2 ((y%4)/4) so that none of them exceeds 2.0.

[0018] After such a table is provided, the value of Expression (2) can be obtained. First, the first table illustrated in **FIG. 29** is referenced using an input integer x as an address so as to obtain the normalized value f(x) for x̂ (4/3). Then, (y%4) is calculated for a given integer of y. Subsequently,

the calculated value of (y%4) is used as an address to obtain a value of g(y)=2 ((y%4)/4) from the second table illustrated in **FIG. 30**. Then, a multiplier is used to calculate f(x)*g(y) so as to obtain a multiplied value f(x)*g(y)=(x,y). Since the obtained value f(x) is within the range of ±1 while the obtained value g(y) is within the range of ±2, the multiplication result h(x,y) should always has a value in the range of ±2.

[**0019**] Then, h(x,y) is multiplied by 2 int(y/4) so as to obtain a multiplied value i(x,y) as a calculation result of Expression (2). The multiplication can be performed by a bit shift operation. In the bit shift operation, it is assumed that a limit operation (so called a "clipping operation") is performed when the final calculation result exceeds the range of ±2. Thus, in the prior art, the calculation i(x,y)=x̂ (4/3)*2 (y/4) is performed using a large-capacity table.

[**0020**] Use of the above conventional method for obtaining a value of x̂ (a/b) requires a large memory area for the table containing different values of x̂ (a/b). For example, assuming that the stored values of x̂ (a/b) are 32-bit fixed point values, the table illustrated in **FIG. 27** would require a memory area of 32 K bytes. Thus, when x has a wide variable range, large-scale hardware has to be employed for the large memory area required.

[**0021**] Moreover, in the conventional method of calculating x̂ (a/b)*2 (y/c), the number of significant digits of the calculated value f(x) is undesirably small for relatively small input integers of x, as those in the upper portion of **FIG. 29**. For example, the value f(x) corresponding to address x=2 uses only the lower 14 bits of the 32 bits that are available. Thus, a calculation performed by using such a table results in a poor accuracy in the calculation result. Moreover, the conventional calculation method employs a wide range of x (i.e., 0-8191), and accordingly requires a memory area as large as 8191*32 bits. Consequently, a conventional decoding device requires a table having a very large capacity.

## SUMMARY OF THE INVENTION

[**0022**] According to one aspect of the invention, an exponential calculation device for calculating x̂ (a/b) (where a and b are each an integer constant) for a given input value of x is provided. The device includes: an input control section for outputting a value of x', wherein x̂ x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 b when x>A; a core section for outputting a value of z'=x' (a/b); and an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 a when x>A.

[**0023**] In one embodiment of the invention, the input control section calculates the value of x' by shifting x down by b bits when x>A.

[**0024**] In another embodiment of the invention, the core section stores in a table values of x' (a/b) for all possible values of x' (where 0≦x'≦A).

[**0025**] In still another embodiment of the invention, A is set to a value such that x is less than or equal to A at a frequency equal to or greater than a predetermined value.

[**0026**] According to another aspect of this invention, an exponential calculation device for calculating x̂ (a/b) (where a and b are each an integer constant) for a given input value of x is provided. The device includes: an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 (b*n) when x>A (where n is an integer which satisfies x/2 (b*n)≦A); a core section for outputting a value of z'=x' (a/b); and an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 (a*n) when x>A.

[**0027**] In one embodiment of the invention, the input control section calculates the value of x' by shifting x down by (b*n) bits when x>A.

[**0028**] In another embodiment of the invention, the core section stores in a table values of x' (a/b) for all possible values of x'(where 0≦x'≦A).

[**0029**] In still another embodiment of the invention, A is set to a value such that x is less than or equal to A at a frequency equal to or greater than a predetermined value.

[**0030**] According to still another aspect of this invention, an exponential calculation device for calculating x̂ (a/b) (where a and b are each an integer constant) for a given input value of x is provided. The device includes: an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/**2** b when x>A; a core section for outputting a value of z'=x' ((a%b)/b); an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 (a%b) when x>A; and a multiplication section for outputting z*(x̂ (int(a/b))).

[**0031**] In one embodiment of the invention, A is set to a value such that x is less than or equal to A at a frequency equal to or greater than a predetermined value.

[**0032**] According to still another aspect of this invention, an exponential calculation device for calculating x̂ (a/b) (where a and b are each an integer constant) for a given input value of x is provided. The device includes: a determination section for outputting a parameter j (j is an integer which is not negative) for specifying a value of x; an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 j when x>A; a core section for outputting a value of z'=x'(a/b); and an output control section for outputting a value of z, wherein z=z'when x≦A and z=z'*2 (j*a/b) when x>A.

[**0033**] In one embodiment of the invention, the output control section includes a memory section for storing values of 2 (j*a/b) using j as an address; and the value of z is calculated by reading out a corresponding value of 2 (j*a/b) from the memory section when x>A.

[**0034**] In another embodiment of the invention, A=2 B (B is an integer constant); and the determination section determines the value of j such that 2 (B+(j−1))≦x>2 (B+j) when x>2 b and such that j=0 when x≦2 b.

[**0035**] According to still another aspect of this invention, an exponential calculation device for calculating x̂ (a/b) (where a and b are each an integer constant) for a given input value of x is provided. The device includes: a determination section for outputting a value of k which denotes a number of digits of x represented in a binary form; an input control section for outputting a value of x', wherein the value of x' is obtained by shifting x up by (B−k) bits when the value of (B−k) is positive and by shifting x down by (k−B) bits when

the value of (B−k) is negative; a core section for outputting a value of z'=x' (a/b); and an output control section for outputting a value of z=z'*(2 ((k−B)*a/b)).

[0036] In one embodiment of the invention, the output control section includes a memory section for storing values of 2 ((k−B)*a/b) using k as an address; and the value of z is calculated by reading out a corresponding value of 2 ((k−B)*a/b) from the memory section.

[0037] In another embodiment of the invention, the core section includes a memory section for storing values of q (a/b) (where 2 (B−1)≦q>2 B, and q is an integer) using at least lower (B−1) bits of q as an address; and the value of z' is calculated by reading out a corresponding value of q (a/b) from the memory section using the at least lower (B−1) bits of x' as an address.

[0038] According to still another aspect of this invention, there is provided a decoding device for extracting a variable quantization code x and a variable scaling factor y from an input bit stream containing the quantization code x and the scaling factor y and calculating x̂ (a/b)*2 (y/c) (where a, b and c are each an integer constant). The device includes: an isolation section for isolating x and y from the input bit stream; a first bit shift section for, when x≦2 N (where N is a predetermined threshold value), shifting x by a number of bits according to N so as to convert x to x1 which is in a fixed point format; a first memory section for storing, for all possible values of q (where q is an integer such that 0≦q≦2 N), using q as an address, values of H(q) which have been converted to a fixed point format by shifting q ((a−b)/b) by a number of bits according to M (where M is an integer value of (N*(a−b))/b); a second memory section for storing, for all possible values of p (where 0≦p<c), values of G(p)=2 (p/c) in a fixed point format; a division section for calculating y1, where y1=int(y/c); a modulo calculation section for calculating y2=y%c; a multiplication section for outputting a value obtained by multiplying together a value of H(x) which has been read out from the first memory section using x as an address, a value of G(y2) which has been read out from the second memory section using y2 as an address and a value of x1 which has been converted by the first bit shift section; and a second bit shift section for shifting the output value from the multiplication section by a number of bits according to y1 obtained by the division section.

[0039] In one embodiment of the invention, the number of bits according to N is (D−N) (where D is a number of significant digits of a fractional portion of a number involved in a fixed point calculation operation); and the number of bits according to y1 is (y1+(N+M)).

[0040] In another embodiment of the invention, M is an integer obtained by rounding up (N*(a−b))/b.

[0041] In still another embodiment of the invention, a=4, b=3 and c=4.

[0042] In still another embodiment of the invention, the device further includes: a determination section for outputting a determination result being true when x≦2 N and being false when x>2 N; and a second division section for outputting the value of x without alteration when the determination result is true and for outputting a value obtained by dividing x by (2 (b*i)). The first bit shift section shifts x by a number of bits according to (N+b*i) (where i is an integer which is not negative) when the determination result is false. The first memory section stores values of H(q) for values of q which are within the variable range of x. The second bit shift section shifts the value output from the multiplication section by a number of bits according to y1 when the determination result is true and by a number of bits according to y1 and a*i when the determination result is false.

[0043] In still another embodiment of the invention, the number of bits according to N is (D−N); the number of bits according to (N+b*i) is D−(N+b*i); the number of bits according to according to M is (D−M); the number of bits according to according to y1 is (y1+N+M); and the number of bits according to y1 and a*i is (y1+N+M+a*i).

[0044] In still another embodiment of the invention, the value of i is determined so that x satisfies 2 (N+b(i−1))<x≦2 (N+bi).

[0045] In still another embodiment of the invention, the device further includes a determination section for outputting a determination result being true when x≦2 N and being false when x>2 N, and further outputting a parameter i for specifying a value of x. The first bit shift section and the second division section and the second shifting section operate according to the determination result from the determination section.

[0046] In still another embodiment of the invention, the second bit shift section shifts the value output from the multiplication section by a number of bits according to y1 when the determination result is true and by a number of bits according to y1+(a*i) when the determination result is false.

[0047] According to still another aspect of this invention, there is provided a decoding device for extracting a variable quantization code x and a variable scaling factor y from an input bit stream containing the quantization code x and the scaling factory and calculating x̂ (a/b)*2 (y/c) (where a, b and c are each an integer constant). The device includes: an isolation section for isolating x and y from the input bit stream; a determination section for outputting a determination result being true when x≦2 N (where N is a predetermined threshold value) and being false when x>2 N, and further outputting a parameter j for specifying a value of x; a first bit shift section for shifting x by a number of bits according to N when the determination result is true and by a number of bits according to (N+j) when the determination result is false so as to convert x to x1 which is in a fixed point format; a first division section for calculating y1, where y1=int(y/c); a second division section for outputting a value of x', wherein x'=x when the determination result is true and x' is an integer value of (x/(2 j)); a first memory section for storing, for all possible values of q (where q is an integer such that 0≦q≦2N) within a variable range of x', using q as an address, values of H(q) which have been converted to a fixed point format by shifting q ((a−b)/b) by a number of bits according to M (where M is an integer value of (N*(a−b))/b); an adjustment coefficient generation section for generating values of I (p, j) in a fixed point format according to 2 (p/c) *2 (j/b) for all possible values of p such that 0≦p<c and all possible values of j; a modulo calculation section for calculating y2=y%c: a multiplication section for outputting a value obtained by multiplying together a value of H(x') which has been read out from the first memory section using x' as an address, a value of I(y2,j) corresponding to y2 and j which has been generated by the adjustment coefficient generation section and a value of x1 which has been con-

verted by the first bit shift section; and a second bit shift section for shifting the output value from the multiplication section by a number of bits according to y1 when $x \leqq 2^N$ and by a number of bits according to y1 and j when $x > 2^N$.

[0048] In one embodiment of the invention, the adjustment coefficient generation section includes a second memory section for storing values of I(p,j) using p and j as an address.

[0049] In another embodiment of the invention, the value of j is determined so as to satisfy $2^{(N+(j-1)} < x \leqq 2^{(N+j)}$ when $x > 2^N$, whereas j=0 when $x \leqq 2^N$.

[0050] In still another embodiment of the invention, a=4, b=3 and c=4.

[0051] According to still another aspect of this invention, there is provided a decoding device for extracting a variable quantization code x and a variable scaling factor y from an input bit stream containing the quantization code x and the scaling factory and calculating $\hat{x}\ (a/b)2\ (y/c)$ (where a, b and c are each an integer constant). The device includes: an isolation section for isolating x and y from the input bit stream; a determination section for determining a number k of digits of x represented in a binary form; a second division section for outputting a value of x', wherein the value of x' is obtained by shifting x up by (N−k) bits when the value of (N−k) is positive and by shifting x down by (k−N) bits when the value of (N−k) is negative; a first bit shift section for shifting x by a number of bits according to k so as to convert x to x1 which is in a fixed point format; a first division section for calculating y1, where y1=int(y/c); a first memory section for storing, using at least lower (N−1) bits of q (where q is an integer such that $2^{(N-1)} \leqq q < 2^N$) as an address, values of H(q) which have been converted to a fixed point format by shifting q ((a−b)/b) by a number of bits according to M (where M is an integer value of (N*(a−b))/b); an adjustment coefficient generation section for generating values of I(p,k) in a fixed point format according to $2^{(p/c)*2\ ((k-N)/b)}$ for all possible values of p such that $0 \leqq p < c$ and all possible values of k; a modulo calculation section for calculating y2=y%c; a multiplication section for outputting a value obtained by multiplying together a value of H(x') which has been read out from the first memory section using the at least lower (N−1) bits of x' as an address, a value of I(y2,k) corresponding to y2 and k which has been generated by the adjustment coefficient generation section and a value of x1 which has been converted by the first bit shift section; and a second bit shift section for shifting the output value from the multiplication section by a number of bits according to y1, M and k.

[0052] In one embodiment of the invention, the adjustment coefficient generation section includes a second memory section for storing values of I(p,k) using p and k as an address.

[0053] In another embodiment of the invention, a=4, b=3 and c=4.

[0054] Thus, the invention described herein makes possible the advantages of: (1) reducing the hardware scale of an exponential calculation device which is capable of calculating $\hat{x}\ (a/b)$ for a given value of x, by reducing the memory area to be used for a table; (2) realizing a decoding device capable of calculating $\hat{x}\ (a/b)*2\ (y/c)$ with a high accuracy using a fixed-point-format processing unit even when the integer x is considerably smaller than the maximum value of the variable range thereof; (3) realizing a decoding device which only has a table for relatively small integers x and still is capable of calculating $\hat{x}\ (a/b)*2\ (y/c)$ even for a large value of x by using the small-capacity table; and (4) realizing a calculation method capable of a calculation involving a large value of x using a small-capacity table by the use of a simple bit shift operation, thereby realizing a decoding device capable of rapidly calculating $\hat{x}\ (a/b)*2\ (y/c)$ for any given values of x and y.

[0055] These and other advantages of the present invention will become apparent to those skilled in the art upon reading and understanding the following detailed description with reference to the accompanying figures.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0056] **FIG. 1** is a block diagram illustrating a structure of an exponential calculation device **100** according to Example 1 of the present invention;

[0057] **FIG. 2** is a flow chart illustrating a signal processing method for the exponential calculation device **100**;

[0058] **FIG. 3** illustrates the contents of a ROM Table 1 of a core section **11**A of the exponential calculation device **100**;

[0059] **FIG. 4** is a flow chart illustrating a signal processing method for another core section **11**B of the exponential calculation device **100**;

[0060] **FIG. 5** illustrates the contents of a ROM Table 2 of the core section **11**B of the exponential calculation device **100**;

[0061] **FIG. 6** is a block diagram illustrating a structure of an exponential calculation device **200** according to Example 2 of the present invention;

[0062] **FIG. 7** is a flow chart illustrating a signal processing method for the exponential calculation device **200**;

[0063] **FIG. 8** illustrates the contents of a ROM Table 3 of a core section **21** of the exponential calculation device **200**;

[0064] **FIG. 9** is a block diagram illustrating a structure of an exponential calculation device **300** according to Example 3 of the present invention;

[0065] **FIG. 10** is a flow chart illustrating a signal processing method for the exponential calculation device **300**;

[0066] **FIG. 11** illustrates a table for providing a value of 2(j/3) in an output control section **32** of the exponential calculation device **300**;

[0067] **FIG. 12** is a block diagram illustrating a structure of an exponential calculation device **400** according to Example 4 of the present invention;

[0068] **FIG. 13** is a flow chart illustrating a signal processing method for the exponential calculation device **400**;

[0069] **FIG. 14** illustrates the contents of a ROM Table 4 of a core section **41** of the exponential calculation device **400**;

[0070] **FIG. 15** illustrates a table for providing a value of $2^{((k-B)/3)}$ in an output control section **42** of the exponential calculation device **400**;

[0071] **FIG. 16** is a block diagram illustrating a structure of a decoding device **500** according to Example 5 of the present invention;

[0072] **FIG. 17** illustrates the contents of a first memory section **52** of the decoding device **500**;

[0073] **FIG. 18** illustrates the contents of a second memory section **53** of the decoding device **500** and a second memory section **64** of a decoding device **600**;

[0074] **FIG. 19** is a block diagram illustrating a structure of the decoding device **600** according to Example 6 of the present invention;

[0075] **FIG. 20** illustrates the contents of a first memory section **63** of the decoding device **600** and a first memory section **73** of a decoding device **700**;

[0076] **FIG. 21** illustrates a method of converting an integer value to value of a fixed point format;

[0077] **FIG. 22** is a block diagram illustrating a structure of the decoding device **700** according to Example 7 of the present invention;

[0078] **FIG. 23** illustrates the contents of an adjustment coefficient generation section **74** of the decoding device **700**;

[0079] **FIG. 24** is a block diagram illustrating a structure of the decoding device **800** according to Example 8 of the present invention;

[0080] **FIG. 25** illustrates the contents of an adjustment coefficient generation section **84** of the decoding device **800**;

[0081] **FIG. 26** illustrates the contents of a first memory section **83** of the decoding device **800**;

[0082] **FIG. 27** illustrates table of values of $\hat{x}$ (4/3) for $0 \leqq x \leqq 8191$ which is provided in a conventional exponential calculation device;

[0083] **FIG. 28** illustrates an exemplary fixed point format;

[0084] **FIG. 29** illustrates the contents of a first table used in a conventional decoding device; and

[0085] **FIG. 30** illustrates the contents of a second table used in the conventional decoding device.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0086] The principle of the present invention will first be described. An exponential calculation device of the present invention only requires a table containing values of $\hat{x}$ (a/b) for input values of x less than or equal to a predetermined value A, and still is capable of obtaining a value of $\hat{x}$ (a/b) for any input value of x. The present invention does not require a large table for storing values of $\hat{x}$ (a/b) for input values of x greater than A, and accordingly requires a memory area much smaller than that of the conventional exponential calculation device which requires a large table for storing values of $\hat{x}$ (a/b) for all the possible input values of x.

[0087] The basic principle of operation of the exponential calculation device of the present invention will now be described based on arithmetic expressions. Assuming the input value to the exponential calculation device is x and the value to be obtained is $z=\hat{x}$ (a/b), where the input value x represents an amplitude of a signal used in decoding audio data, for example, and A denotes a threshold value of the variable range of x.

[0088] When $x \leqq A$, the value of $\hat{x}$ (a/b) for the input value of x is already provided in a table, whereby the value of $\hat{x}$ (a/b) can easily be obtained or retrieved from the table.

[0089] When x>A, x is first divided by a predetermined value to obtain x' such that x' A. A value x' (a/b) is read out from the table and is multiplied by the predetermined value so as to obtain the value of $\hat{x}$ (a/b) for the given value of x. In this way, a value of $\hat{x}$ (a/b) for any given value of x can be obtained. For example, when a=4 and b=3, and when x is in the range of $8*A \leqq x>A$, x' is first obtained as x'=x/8. Then, a corresponding value is read out from the table for the value of x', thereby obtaining z'. Since z'=(x/8) (4/3), the value of $z=\hat{x}$ (4/3) can be obtained as z'*(8 (4/3))=z'*16.

### EXAMPLE 1

[0090] An exponential calculation device according to

[0091] Example 1 of the present invention will now be described with reference to the figures. The exponential calculation device of this example performs an exponential calculation $\hat{x}$ (4/3). In this example, the range of the input value of x is 0-8191. The range of the exponentiated value $\hat{x}$ (4/3) and the range of the input value of x conform to the MPEG 2 audio AAC standard (see ISO/IEC 13818-7:1997, P57).

[0092] **FIG. 1** is a block diagram illustrating a structure of an exponential calculation device **100** according to Example 1 of the present invention. The exponential calculation device **100** includes an input control section **10**, a core section **11** and an output control section **12**.

[0093] **FIG. 2** is a flow chart illustrating signal processing performed by the exponential calculation device **100**, showing respective signal processing steps performed by the input control section **10**, the core section **11** and the output control section **12**. Referring to **FIG. 2**, the respective components of the exponential calculation device **100** will now be described in detail.

[0094] The input control section **10** determines if the input value of x is less than or equal to a threshold value A (e.g., 1023 in this example). If the input value of x is less than or equal to 1023, the input control section **10** provides the value of x, without alteration, as x' to the core section **11**. If the input value of x exceeds 1023, the input control section **10** provides a value obtained by dividing x by 2 3 as x' to the core section **11**.

[0095] The core section **11** provides the value of x' (4/3) for the value of x', which is in the range from 0 to 1023.

[0096] The core section **11** illustrated in **FIG. 2** is one example of the core section **11** and will hereinbelow be referred to as the "core section **11**A". The core section **11**A has a ROM Table 1 (z'=ROMTABLE1[x']). The core section **11**A outputs a value read out from the ROM Table 1 for the value of x' as z'.

[0097] **FIG. 3** illustrates the contents of the ROM Table 1 of the core section **11**A. The ROM Table 1 contains values of x' (4/3) which have been calculated for value of x' in the range from 0 to 1023.

[0098] The output control section **12** provides the output value z' from the core section **11**A, without alteration, as z when the input value of x is less than or equal to 1023, and provides a value obtained by multiplying the output value from core section **11**A by 2 4 as z when x exceeds 1023.

[0099] An operation of the exponential calculation device **100** having such a structure will now be described. First, step **S10** determines if a given value of x is less than or equal to 1023. When x does not exceed 1023, the process proceeds to step **S11**, thereby providing the value of x, without alteration, as x' to the core section **11**A. If it is determined that x exceeds 1023, the process proceeds to step **S12** so as to provide x' by dividing x by 2 3 to the core section **11**A. The division of x by 2 3 can be performed, for example, by shifting x down by 3 bits or by first adding 4 to x and then shifting down the obtained value by 3 bits for a "0-down-1-up" round-off process, where 0.0's are rounded down and 0.1's are rounded up.

[0100] Next, in step **S13**, the core section **11**A reads out from the ROM Table 1, as illustrated in **FIG. 3, a** value of x' (4/3) for the value of x' provided by the input control section **10**.

[0101] **FIG. 4** is a flow chart illustrating a signal processing procedure performed by another exemplary core section **11**B for use with the exponential calculation device **100**. The core sections **11**A and **11**B, as illustrated in **FIGS. 2 and 4**, respectively, may be used inter-changeably in the present invention. The core section **11**B stores in a ROM Table 2 values of $\hat{x}$ (L/b) for values of x' from 0 to the threshold value A, wherein L=a/m (where m is a positive constant). The core section **11**B raises z' (from the ROM Table 2) to the power of m, and outputs the resulting value as new z'.

[0102] **FIG. 5** illustrates the contents of the ROM Table 2 of the core section **11**B. The core section **11**B uses the ROM Table 2 (z'=ROMTABLE2[x']) which stores values of x' (L/b) for m=4 and L=1 (i.e., x' (⅓)) for values of x' in the range from 0 to 1023. Referring to **FIG. 4**, in step **S17**, z' is read out from the ROM Table 2 and, in step **S18**, z' is raised to the power of 4, thereby obtaining the value of x' (4/3). Such an additional exponential calculation as is required in this core section **11**B can easily be processed by a multiplier of a DSP, thereby maintaining a small capacity of the ROM table.

[0103] Referring back to **FIG. 2**, in step **S14**, the output control section **12** determines if the input value of x is less than or equal to 1023. When x is less than or equal to 1023, the process proceeds to step **S15**, thereby outputting the output value from the core section **11**, without alteration. When it is determined that the input value of x exceeds 1023 in step **S14**, the process proceeds to step **S16**, thereby outputting a value obtained by multiplying the output value from the core section **11** by **2** 4.

[0104] As described above, the present invention does not require providing values of $\hat{x}$ (a/b) for all the possible values of x contemplated (e.g., 0-8191 in this example), but only requires providing values of $\hat{x}$ (a/b) for input values of x less than or equal to the predetermined threshold value A=1023. In this way, values of $\hat{x}$ (a/b) for input values of x exceeding A can also be approximately calculated. Thus, it is possible to effect an exponential calculation with small-scale hardware.

[0105] Herein, **1023** is selected as the threshold value A for the following reasons. Under the MPEG 2 audio AAC standard, the input value of $\hat{x}$ (4/3) can range from 0 to 8191. In fact, however, values from 0 to 1023 are involved in most cases. Therefore, for x ranging from 0 to 1023, the calculation $\hat{x}$ (4/3) should accurately be performed so as not to deteriorate the sound quality. However, for x of 1024 or greater, since x rarely takes such a large value, the sound quality would not significantly deteriorate even if the calculation is not as accurate as that for x ranging from 0 to 1023. Accordingly, for x exceeding 1023, an approximate calculation can be performed according to the method of the present example.

[0106] Among the values from 0 to 1023, actual calculations particularly involve, in most cases, values from 0 to 255. Accordingly, the statistic threshold value A may be set to 255. The following example of the invention will discuss an example where the statistic threshold value is set to 255.

EXAMPLE 2

[0107] Example 2 of the present invention will now be described with reference to the figures. This example provides an exponential calculation device for obtaining a value of $\hat{x}$ (4/3), as an exemplary exponentiated value $\hat{x}$ (a/b). The exponential calculation device of the present example can be characterized in that it performs the exponential calculation $\hat{x}$ (a/b) by performing two exponential calculations for an integral portion of a/b and for a fractional portion thereof (where a and b are each an integer, and a>b). The exponential calculation for the integral portion of a/b can therefore be expressed as $\hat{x}$ (int(a/b)), whereas the exponential calculation for the fractional portion of a/b can be expressed as $\hat{x}$ ((a%b)/b). The symbol "int( )" as used herein denotes a round-down operation, where any fractional portion of the number is cut off, and the symbol "%" as used herein denotes a modulo operation for obtaining a remainder (integer) of a division.

[0108] As in Example 1, the range of input values of x is from 0 to 8191, conforming to the MPEG 2 audio AAC standard (see ISO/IEC 13818-7:1997, P57).

[0109] **FIG. 6** is a block diagram illustrating a structure of an exponential calculation device **200** according to Example **2** of the present invention. The exponential calculation device **200** includes an input control section **20**, a core section **21**, an output control section **22** and a multiplication section **23**.

[0110] **FIG. 7** is a flow chart illustrating a signal processing method of the exponential calculation device **200**, showing the respective signal processing procedures performed by the input control section **20**, the core section **21**, the output control section **22** and the multiplication section **23**.

[0111] When the input value of x is less than or equal to 255 (first threshold value A1), the input control section **20** provides the value of x, without alteration, to the core section **21** as x'. When the input value of x exceeds 255 but is less than or equal to 2047 (second threshold value A2), the input control section **20** provides a value obtained by dividing x by 2 3 to the core section **21** as x'. When the input value of x is equal to or grater than 2048, the input control section **20** provides a value obtained by dividing x by **26** to the core section **21** as x'.

[0112] **FIG. 8** illustrates the contents of a ROM Table 3 of the exponential calculation device **200**. The core section **21** has the ROM Table 3, as illustrated in **FIG. 8**, and provides values of x' ((a%b)/b) (i.e., x' (⅓)) for values of x' from 0 to 255. The values of x' (⅓) for values of x' from 0 to 255 are stored in the ROM Table 3 in advance.

[0113] When the input value of x is less than or equal to the first threshold value A1, the output control section **22** outputs the output value from the core section **21**, without alteration. When the input value of x exceeds the first threshold value A1 but less than or equal to the second threshold value A2, the output control section **22** outputs a value obtained by multiplying the output value z' from the core section **21** by 2 (a%b) as z. When the input value of x exceeds the second threshold value A2, the output control section **22** outputs a value obtained by multiplying the output value from the core section **21** by 2 ((a%b)*2). The multiplication section **23** multiplies x̂ (int (a/b)) with the output value z from the output control section **22** so as to output the multiplied value as a new value of z.

[0114] An operation of the exponential calculation device **200** of Example 2 having such a structure will now be described. Referring to **FIG. 7**, in step S**20**, the input control section **20** determines if the input value of x is less than or equal to the first threshold value, 255. When the input value of x is less than or equal to 255, the process proceeds to step S**21**, thereby providing the value x, without alteration, to the core section **21** as x'. If it is determined in step S**20** that the input value of x is equal to or greater than 256, the process proceeds to step S**22**, where it is determined if the input value of x is less than or equal to the second threshold value A2, 2047. When x is less than or equal to 2047, the process proceeds to step S**23**, where a value obtained by dividing x by 2 3 is provided to the core section **21** as x'. When it is determined in step S**22** that the input value of x is equal to or greater than 2048, the process proceeds to step S**24**, where a value obtained by dividing x by 2 6 is provided to the core section **21** as x.

[0115] The division of x by 2 3 can be performed, for example, by shifting x down by 3 bits or by first adding 4 to x and then shifting down the obtained value by 3 bits for the "0-down-1-up" round-off process. The division of x by 2 6 can be performed, for example, by shifting x down by 6 bits or by first adding 32 to x and then shifting down the obtained value by 6 bits for the "0-down-1-up" round-off process.

[0116] Then, in step S**25**, the core section **21** reads out a value of x' (⅓) from the ROM Table 3, as illustrated in **FIG. 8**, for a value of x' provided by the input control section **20**.

[0117] Then, in step S**26**, the output control section **22** determines if the input value of x is less than or equal to 255. If x is less than or equal to 255, the process proceeds to step S**27**, thereby outputting the output value from the core section **21**, without alteration. When it is determined in step S**26** that the input value of x is equal to or greater than 256, the process proceeds to step S**28**. Step S**28** determines if the input value of x is less than or equal to 2047. If x is less than or equal to 2047, the process proceeds to step S**29**. If x is equal to or greater than 2048, the process proceeds to step S**30**. Step S**29** outputs a value obtained by multiplying the output value from the core section **21** by 2 1, and Step S**30** outputs a value obtained by multiplying the output value from the core section **21** by 2 2.

[0118] Finally, instep S**31**, the multiplication section **23** multiplies the input value of x with the output value from the output control section **22**. In this instance, the value to be obtained is x̂ (4/3), and the output value from the output control section **22** is x̂ (⅓). Therefore, x̂ (4/3) is calculated by multiplying the original value of x with x̂ (⅓).

[0119] As described above, in the present example, an exponential calculation device for calculating x̂ (a/b) (where a>b) for an input value of x includes a core section for providing values of x' ((a%b)/b) for values of x' from 0 to A. Moreover, the exponential calculation device of the present invention is provided with a multiplication section for multiplying a value of x̂ (int (a/b)), for an input value of x, with an output value from the output control section.

[0120] Thus, it is not required to store values of x̂ (a/b) for all the possible input values of x contemplated (0 to 8191 in this example), but it is only required to store values of x̂ ((a%b)/b) for input values of x less than or equal to the predetermined value A (255 in this example). In this way, it is possible to exponentiate any value of x. Thus, it is possible to perform, using small-scale hardware, an exponential calculation with a fractional exponent.

## EXAMPLE 3

[0121] An exponential calculation device according to Example 3 of the present invention will now be described with reference to the figures. This example provides an exponential calculation device for obtaining a value of x̂ (⅓), as an exemplary exponentiated value x̂ (a/b). The input value of x is assumed to range from 0 to 8191.

[0122] **FIG. 9** is a block diagram illustrating a structure of an exponential calculation device **300** according to Example 3 of the present invention. The exponential calculation device **300** includes an input control section **30**, a core section **31**, an output control section **32** and a determination section **33**.

[0123] **FIG. 10** is a flow chart illustrating a signal processing method of the exponential calculation device **300**, showing the respective signal processing procedures performed by the input control section **30**, the core section **31**, the output control section **32** and the determination section **33**.

[0124] The input control section **30** determines if the input value of x is less than or equal to the threshold value A (e.g., 1023 in this example). If x is less than or equal to 1023, the input control section **30** provides the value of x, without alteration, to the core section **31** as x'. If x is greater than 1023, the input control section **30** provides a value obtained by dividing x by 2 j to the core section **31** as x'.

[0125] The value of j is determined by the determination section **33** as follows. The determination section **33** first determines the number k of digits of x represented in a binary form, and then determines the value of j such that j=k−10. For example, when the input value of x is equal to or greater than 1024 and less than 2048, the value of j is 1. The value "10" is the number of digits of a value 1024 represented in the binary form. Accordingly, the value of j represents by how many digits x is greater than the threshold value A in the binary form.

[0126] As the core section **11**B of Example 1, the core section **31** includes a ROM Table 2, as illustrated in **FIG. 5**.

Values of x' (⅓) for x1 from 0 to 1023 have been calculated and stored in the ROM Table 2. The core section **31** provides values of x' (⅓) for x' from 0 to 1023. When the input value of x is less than or equal to 1023, the output control section **32** outputs the value of z' from the core section **31**, without alteration, as z. When the input value of x is greater than **1023**, the output control section **32** outputs a value obtained by multiplying the value of z' output from the core section **31** by 2 (j/3), as z.

[**0127**] An operation of the exponential calculation device having such a structure will be described with reference to **FIG. 10**. The input control section **30** of **FIG. 10** first receives an input value of x and determines, in step **S40**, if the value of x is less than or equal to 1023. When x is less than or equal to 1023, the process proceeds to step **S41**, whereby the value of x is provided, without alteration, to the core section **31** as x'. When it is determined in step **S40**, that the input value of x is greater than 1023, the process proceeds to step **S42**, a value obtained by dividing x by **2** j is provided to the core section **31** as x'.

[**0128**] The value of j is determined by the determination section **33** as follows. In particular, the determination section **33** first determines the number k of digits of x represented in the binary form in step **S47**. Then, j=k−10 is obtained in step **S48**.

[**0129**] The division of x by 2 j can be performed, for example, by shifting x down by j bits or by first adding 2 (j−1) to x and then shifting down the obtained value by j bits for the "0-down-1-up" round-off process.

[**0130**] Next, in step **S43**, the core section **31** reads out the value of x' (⅓) from the ROM Table 2 in **FIG. 5** for the value of x' provided from the input control section **30**.

[**0131**] Finally, in step **S44**, the output control section **32** determines if the input value of x is less than or equal to 1023. When x is less than or equal to 1023, the process proceeds to step **S45**, whereby the output value from the core section **31** is output, without alteration. When it is determined in step **S44** that the input value of x is greater than 1023, the process proceeds to step **S46**, thereby outputting a value obtained by multiplying the output value from the core section **31** by 2 (j/3).

[**0132**] **FIG. 11** illustrates a table for providing a value of 2 (j/3) in an output control section **32** of the exponential calculation device **300**. Values of 2 (j/3) can be stored in advance in a table using j as an address, so that a value of 2 (j/3) for a given value of j can be obtained by reading out the value from the table. The multiplication of the output value from the core section **31** by 2 (j/3) can be performed using an ordinary multiplier.

[**0133**] As described above, the present example does not require providing values of x̂ (⅓) for all the possible values of x contemplated (e.g., 0-8191 in this example), but only requires providing values of x̂ (⅓) for input values of x less than or equal to the predetermined threshold value A=1023. In this way, values of x̂ (⅓) for input values of x exceeding A can also be approximately calculated. Thus, it is possible to perform an exponential calculation with small-scale hardware.

[**0134**] Even when x exceeds the threshold value A, the loss of bits can be minimized, so that the approximate value can be obtained with a high accuracy.

[**0135**] In the present example, the threshold value A is set to 1023, the highest possible value represented by 10 bits, and the determination section **33** is constructed so as to output a value which indicates by how many bits the number of bits of x is greater than 10 bits. However, the present invention is not limited to such a structure. Alternatively, when the use of an even smaller memory area is desired, for example, the threshold value A can be set to 255, the highest possible value represented by 8 bits, and the determination section **33** can be constructed accordingly so as to output a value which indicates by how many bits the number of bits of x is greater than 8 bits. Conversely, when a higher calculation accuracy is desired, the threshold value A can be set to 4095, the highest possible value represented by 12 bits, and the determination section **33** can be constructed accordingly so as to output a value which indicates by how many bits the number of bits of x is greater than 12 bits.

## EXAMPLE 4

[**0136**] An exponential calculation device according to Example 4 of the present invention will now be described with reference to the figures. This example provides an exponential calculation device for obtaining a value of x̂ (⅓), as an exemplary exponentiated value x̂ (a/b). The input value of x is assumed to range from 0 to 8191.

[**0137**] **FIG. 12** is a block diagram illustrating a structure of an exponential calculation device **400** according to Example 4 of the present invention. The exponential calculation device **400** includes an input control section **40**, a core section **41**, an output control section **42** and a determination section **43**.

[**0138**] **FIG. 13** is a flow chart illustrating a signal processing method of the exponential calculation device **400**, showing the respective signal processing procedures performed by the input control section **40**, the core section **41**, the output control section **42** and the determination section **43**.

[**0139**] The determination section **43** determines the number k of digits of x represented in the binary form. The input control section **40** determines if k is less than or equal to a threshold value B (e.g., **10** in this example). When x is less than or equal to 10, the input control section **40** provides to the core section **41** a value obtained by shifting x up by (B−k) bits as x'. When x is greater than 10, the input control section **40** provides to the core section **41** a value obtained by shifting x down by (k−B) bits as x'.

[**0140**] The core section **41** has a ROM Table 4 (z'= ROMTABLE4[x']) and provides values of x' (⅓) for values of x' ranging from 2 (B−1) to 2 B−1 (from 512 to 1023 in this example). The output control section **42** outputs a value obtained by multiplying z' output from the core section **41** by 2 ((k−B)/3).

[**0141**] **FIG. 14** illustrates the contents of a ROM Table 4 of a core section **41** of the exponential calculation device **400**. Values of x' (⅓) for values of x' from 512 to 1023 are stored in the ROM Table 4 in advance.

[**0142**] Note that, referring to **FIG. 14, a** value of x' (⅓) for x'=512 (=2 (B−1)) is stored at address **0**, and a value of x' (⅓) for x'=1023 (=2 B−1) is stored at address **511**. The table contains such values for the following reason. Where k denotes the number of bits of x represented in the binary

form, the input control section **40** calculates x' by shifting x up by (B−k) bits when B−k is positive and by shifting x down by (k−B) bits when B−k is negative, whereby the $B^{th}$ bit of x' is always 1. Thus, the value is always in the range from 2 (B−1) to 2 B−1. Therefore, the table of **FIG. 14** can be referenced using only the lower (B−1) bits of x' as an address. As a result, the size of the table can be reduced to half.

[**0143**] An operation of the exponential calculation device having such a structure will be described with reference to **FIG. 13**. The determination section **43** of **FIG. 12** first determines the number k of digits of x represented in the binary form in step **S55**. The input control section **40** receives the input value of x, and determines, in step **S50**, if k is less than or equal to B (B=10 in this example). When k is less than or equal to 10, the process proceeds to step **S51**, thereby providing to the core section **41** a value obtained by shifting x up by (B−k) bits as x'. When k is greater than **10**, the process proceeds to step **S52**, thereby providing to the core section **41** a value obtained by shifting x down by (k−B) bits.

[**0144**] Then, in step **S53**, the core section **41** reads out from the ROM Table 4 of **FIG. 14 a** value of x' (⅓) for the value of x' provided from the input control section **40**. As described above, a value of x' (⅓)for x'=512(=2 (B−1)) is stored at address **0**, and a value of x' (⅓) for x'=1023 (=2 B−1) is stored at address **512**. Therefore, only the lower (B−1) bits are used as an address for referencing the table of **FIG. 14**.

[**0145**] Finally, instep **S54**, the output control section **42** of **FIG. 12** outputs a value obtained by multiplying the output value from the core section **41** by 2 ((k−B)/3).

[**0146**] **FIG. 15** illustrates a table for providing a value of 2 ((k−B)/3) in an output control section **42** of the exponential calculation device **400**. Values of 2 ((k−B)/3) can be stored in advance in a table using k as an address, so that a value of 2 ((k−B)/3)for a given value of k can be obtained by reading out the value from the table. The multiplication of the output value from the core section **41** by 2 ((k−B)/3) can be performed using an ordinary multiplier.

[**0147**] As described above, the present example does not require providing values of x̂ (⅓) for all the possible values of x contemplated (e.g., 0-8192 in this example), but only requires providing values of x' (⅓) for input values of x less than or equal to the predetermined threshold value B (e.g., 10 in this example). In this way, values of x̂ (⅓) for input values of x which has more than B bits can also be approximately calculated. Thus, it is possible to perform an exponential calculation with small-scale hardware.

[**0148**] Even when x has more than B bits, the loss of bits can be minimized, so that the approximate value can be obtained with a high accuracy.

[**0149**] Particularly, the present example provides an advantage that the size of the table can be reduced to half while maintaining substantially the same level of accuracy as that in Example 3 in calculating x̂ (⅓) for x having more than B bits.

EXAMPLE 5

[**0150**] A decoding-device according to Example 5 of the present invention will now be described with reference to the figures. As described above in the prior art section, a bit stream encoded based on the MPEG 2 audio encoding standard AAC contains a quantization code "x" and a scaling factor "y". A decoding device of the present example is an encoding device for calculating x̂ (a/b)*2 (y/c) for values of x and y contained in the audio bit stream.

[**0151**] The quantization code x is any integer in the range from 0 to 8191, for example, and the scaling factor y is any integer from −100 to 156. These variable ranges of x and y conform to the MPEG 2 audio encoding standard AAC (see ISO/IEC 13818-7:1997). In the following description, it is assumed that x and y take these variable ranges, though the present invention is not limited thereto.

[**0152**] **FIG. 16** is a block diagram illustrating a structure of a decoding device **500** according to Example **5** of the present invention. The decoding device **500** includes an isolation section **50**, a determination section **51**, a first memory section **52**, a second memory section **53**, a division section **54**, a modulo calculation section **55**, a multiplication section **56** and a bit shift section **57**.

[**0153**] Referring to **FIG. 16**, the isolation section **50** isolates the quantization code x and the scaling factor y from an input bit stream. The determination section **51** determines if the quantization code x output from the isolation section **50** is less than or equal to 2 N (where N is a predetermined integer threshold value). The determination section **51** outputs a determination result being true when x≦2 N and being false when x>2 N. The first memory section **52** stores values according to q (a/b) in the fixed point format for integer values of q ranging from 0 to (2−Q−1) (wherein Q is an integer).

[**0154**] The first memory section **52** has the following structure. Where q is any integer within the variable range of x which is isolated from the bit stream by the isolation section **50** (0≦q<2 Q−1), the first memory section **52** stores values of q (a/b) for all the possible values of q in the fixed point format. Thus, the first memory section **52** stores values of F(q) using q as an address. When the integer value q exceeds 2 N (N is a predetermined threshold value), F(q) represents values obtained by normalizing values of q (a/b) with 2 M'. When the integer value q is in the range from 0 to 2 N, F(q) represents values obtained by normalizing values of q a/b) with 2 M' and then shifting the resulting value by (M'−M) bits. M is an integer value of (N*a)/b, and M' is an integer value of (Q*a) divided by b. The integer M is a value obtained by rounding up the value of (N*a)/b, and the integer M' is a value obtained by rounding up (Q*a) /b.

[**0155**] **FIG. 17** illustrates the contents of the first memory section **52** of the decoding device **500**. The memory section **52**, as illustrated in **FIG. 17**, stores values of F(q) where a=4, b=3, N=5, Q=13, M=7 and M'=18. For each of the values in **FIG. 17** in the fixed point format, the most significant bit is a sign bit and a decimal point is between the second bit and the third bit from the most significant bit. The second bit next to the most significant bit and D bits on the right of the decimal point are significant digits to be calculated.

[**0156**] The division section **54** divides the scaling factor y output from the isolation section **50** so as to obtain the quotient y1=int(y/c). The modulo calculation section **55** obtains the remainder y2=(y%c) of the division of the scaling factor y by c.

[0157] The second memory section **53** stores in the fixed point format values of G(p)=2 (p/c) for integral values of p in the range from 0 to 3 (where p is an integer within the variable range of y2) which is calculated by the modulo calculation section **55**.

[0158] **FIG. 18** illustrates the contents of the second memory section **53** of the decoding device **500**, which contains values of G(p) where c=4. The fixed point format of the values G(p) in **FIG. 18** may be the same as that in **FIG. 28**.

[0159] The multiplication section **56** multiplies a value of F(x) read out from the first memory section **52** by a value of G(y2) read out from the second memory section **53**.

[0160] The bit shift section **57** shifts the output value from the multiplication section **56** by (y1−(M'−M)) bits, if the determination result from the determination section **51** is true, and by y1 bits, if the determination result from the determination section **51** is false, thereby outputting the shifted value.

[0161] An operation of the decoding device **500** having such a structure will be described while assuming a=4, b=3, c=4, M=7, N=5, Q=13 and D=30. The isolation section **50** of **FIG. 16** isolates the quantization code x and the scaling factor y from an input bit stream. Then, the determination section **51** determines if the quantization code x is less than or equal to 25. The division section **54** obtains the quotient y1 of the division of the scaling factor y by 4, and the modulo calculation section **55** obtains the remainder y2 of the division of the scaling factor y by **4**. The multiplication section **56** multiplies the value of F(x) from address x of the first memory section **52** by the value of G(y2) from the address y2 of the second memory section **53**.

[0162] Through these operations, a value of ($\hat{x}$ (4/3)*2 ((y%4)/4)) has been output from the multiplication section **56** according to Expression (2) converted from Expression (1). Then, the bit shift section **57** shifts the output value from the multiplication section **56** by y1−(18−7) bits, if the determination result from the determination section **51** is true, and by y1 bits, if the determination result from the determination section **51** is false. The effect of such an operation is as follows. First, the operation of shifting by y1 bits corresponds to the calculation 2 int(y/4), the third term in Expression (2) (which is a multiplication of the first, second and third terms). The output value from the multiplication section **56** is multiplied by the quotient 2 y1 output from the division section **54** so as to obtain an output signal of the original magnitude.

[0163] When the determination result from the determination section **51** is true, the operation of shifting by −(18−7)bits is performed for the following reason. When x is greater than 25, the value of $\hat{x}$ (4/3) has been normalized by 2 18. When x is less than or equal to 2 5, another value is stored which has been obtained by shifting up the value of $\hat{x}$ (4/3) by (18−7) bits as compared with that when x is greater than 25. Thus, the value is shifted by additional 11 bits so as to adjust the final calculation result.

[0164] As described above, in the present example, even when the value of x is considerably smaller than the maximum value of the variable range thereof, the value of $\hat{x}$ (4/3) can be retained with more significant bits up to the final stage of outputting the calculation result, whereby it is possible to calculate x(4/3)*2 (y/4) with a high accuracy. In the present example, N=5 and M=7, though any other values can be selected for N and M. For example, N may alternatively be set to 9, where M=9*4/3=12.

## EXAMPLE 6

[0165] A decoding device according to Example 6 of the present invention will now be described with reference to the figures. The decoding device of this example receives a bit stream encoded based on the MPEG 2 audio decoding standard, isolates the quantization code x and the scaling factor y contained therein, and calculates Expression (3) below which is a generalized form of Expression (1).

$$\hat{x}\ (a/b)*2\ (y/c) \tag{3}$$

[0166] In the present example, the variable range of the quantization code x is from 0 to 4096, and the variable range of the scaling factor y is from −100 to 156, for example. In the present example, Expression (3) is converted to Expression (4) below.

$$x*\hat{x}\ ((a-b)/b)*2\ ((y\%c)/c)*2\ \mathrm{int}(y/c) \tag{4}$$

[0167] An operation of decoding an input bit stream by using Expression (4) and a memory section requiring even smaller memory capacity will be described.

[0168] **FIG. 19** is a block diagram illustrating a structure of a decoding device **600** according to Example **6** of the present invention. The decoding device **600** includes an isolation section **60**, a determination section **61**, a first bit shift section **62**, a first memory section **63**, a second memory section **64**, a first division section **65**, a modulo calculation section **66**, a second division section **67**, a multiplication section **68** and a second bit shift section **69**.

[0169] Referring to **FIG. 19**, the isolation section **60** isolates the quantization code x and the scaling factor y from the input bit stream. The determination section **61** determines if the quantization code x is less than or equal to 2 N (where N is a threshold value corresponding to the number of addresses of the first memory section **63**), thereby outputting a determination result being true, if $x \leqq 2$ N, and being false, if x>2 N. When x>2 N, the determination section **61** further outputs a parameter i for specifying a value of the variable x (where i is an integer which is not negative; and the variable range of x is from 2 N to 2 (N+b*i)). The value of i is determined so that x satisfies 2 (N+3(i−1)<$x \leqq 2$ (N+3i). The determination result from the determination section **61** and the value of the parameter i are provided to the second division section **67**, the first bit shift section **62** and the second bit shift section **69**.

[0170] Assuming that the multiplication section **68**, which will be described later in greater detail, functions to perform a calculation for an input value having a bit width of (D+2) in the fixed point format, where the most significant bit of the input value indicates whether the value is positive or negative, and the second most significant bit and the D bits to the right of the decimal point represent the significant digits of the input value. Then, the first bit shift section **62** converts the integer x to x1 which is a value having the fixed point format by shifting x by D−N bits, if the determination result from the determination section **61** is true, and by D−(N+b*i) bits (utilizing the value of i), if the determination result from the determination section **61** is false.

[0171] The first division section **65** performs a division of the value of y output from the isolation section **60** by c so as to output the quotient y1 thereof. The modulo calculation section **66** performs a division of the value of y output from the isolation section **60** by c so as to output the remainder y2thereof. The second division section **67** receives the value of x which has been isolated by the isolation section **60**, so as to output the value of x, without alteration, as x2 , if the determination result from the determination section **61** is true, and to output an integer value of x̂ (2 (b*i)) as x3 , if the determination result from the determination section **61** is false.

[0172] The first memory section **63** stores values of H(q) for values of q (q is an integer value in the range from 0 to 2 N). The values of H(q) are obtained by converting values of q ((a−b) /b) into the fixed point format by shifting the values of q ((a−b)/b) by (D−M) bits (where M is an integer value of N*(a−b) divided by b).

[0173] The second memory section **64** stores in the fixed point format values of G(p)=2 (p/c) for values of y2 (y2=p) output from the modulo calculation section **66**. The contents of the second memory section **64** may be the same as that shown in **FIG. 18**, that is, values of 2 (p/c) for integer values of p in the range from 0 to 3.

[0174] The multiplication section **68** multiplies together a value from address x2 or x3 of the first memory section **63**, a value from address y2 of the second memory section **64** and a value of x1 output from the first bit shift section **62**. Where M is an integer value of N*(a−b) divided by b, the second bit shift section **69** shifts the output value from the multiplication section **68** by (y1+N+M) bits, if the determination result from the determination section **61** is true, and by (y1+N+M+a*i) bits, if the determination result from the determination section **61** is false.

[0175] **FIG. 20** illustrates the contents of the first memory section **63** of the decoding device **600**. The table illustrated in **FIG. 20** contains values of H(q) in the fixed point format which are obtained by normalizing values of q ((a−b)/b) with 2 M (where q is an integer value in the range from 0 to 2 N). In this example, the values of H(q) are obtained where a=4, b=3, N=9, M=3 and D=30. M=3 is calculated from 9*(4−3)/3. As described above, in the format of this table, the most significant bit is a sign bit, and a decimal point is between the second bit and the third bit from the most significant bit. Thus, a value of H(q) is at least −2 and at most less than 2.

[0176] **FIG. 21** illustrates a method of converting an integer value to value of a fixed point format. In the present example, N=9, M=3 and D=30, though these values may be different, of course. For example, N may alternatively be set to 10, while M is set to an integer value, 4, which is obtained by rounding up 10*(4−3)/3.

[0177] An operation of the decoding device **600** having such a structure will be described while assuming a=4, b=3, c=4, M=3, N=9 and D=30, with reference to FIGS. **18** to **27**. First, the isolation section **60** of **FIG. 19** isolates the quantization code x and the scaling factor y from the input bit stream. Then, the determination section **61** determines if the quantization codex is less than or equal to 2 9 (=512), and outputs a determination result being true, if x is less than or equal to 29, and being false, if x exceeds 29. The determination section **61** further outputs a value of i according to the variable range of the quantization code X.

[0178] Next, the first bit shift section **62** converts the integer value of x to x1 having the fixed point format by shifting x by (30−9) bits, if the determination result from the determination section **61** is true, and by 30−(9+3) bits, if the determination result from the determination section **61** is false.

[0179] This operation will further be described below in detail. As illustrated in **FIG. 28**, in the fixed point format of the present example, the most significant bit is a sign bit, and a decimal point is between the second bit and the third bit from the most significant bit. Thus, when x is a value which has **9** bits or less (i.e., less than or equal to 29), x can be normalized by shifting x up by (30−9)=21 so that x is 1.0 or less as represented in the fixed point format, as illustrated in **FIG. 21**.

[0180] Similarly, when an input integer value of x has **10** to (9+3)=12 bits, x can be normalized by shifting x up by 30−(9+3)=18 so that x is 1.0 or less as represented in the fixed point format. Based on such a principle, the first bit shift section **62** converts the integer of x to x1 having the fixed point format by shifting x by (30−9)=21 bits, when the determination result from the determination section **61** is true, i.e., when x is less than or equal to 512. When the determination result from the determination section **61** is false, i.e., when x is greater than 512 and less than or equal to 212, the first bit shift section **62** converts the integer value of x to x1 having the fixed point format by shifting x by 30−(9+3)=18 bits.

[0181] The second division section **67** outputs the value of x, without alteration, as x2 , when the determination result from the determination section **61** is true, and outputs an integer value of x/8 as x3, when the determination result from the determination section **61** is false, i.e., when x is greater than 29 and less than or equal to 212. The value of x3 is used as an address of the first memory section **63** because the first memory section **63** only stores values according to q (⅓) for integer values of q from 0 to 29, i.e., 512. Therefore, when the determination result from the determination section **61** is true, i.e., when x is less than or equal to 512, the value of x is output, without alteration, as x2 , whereas when the determination result from the determination section **61** is false, i.e., when x is greater than 29and less than or equal to 212, x3 less than or equal to 512 is generated and output by calculating an integer value of x/8.

[0182] The first division section **65** obtains a quotient y1 of a division of y by 4, and the modulo calculation section **66** obtains a remainder y2 of the division of y by 4. Then, the multiplication section **68** multiplies together a value of H(x2 ) from address x2 of the first memory section **63**, a value of G(y2) from address y2 of the second memory section **64** and a value x1 output from the first bit shift section **62**.

[0183] Through these operations, the multiplication section **68** provides a value corresponding to x*x̂ (⅓)*2 ((y%4)/4) according to Expression (4) which is converted from Expression (3) (a value before the number of digits, i.e., gain, thereof is adjusted). This is because the output x1 from the first bit shift section **62** represents a value of x as converted into the fixed point format, the first memory section **63** stores values corresponding to x̂ (⅓) in the fixed point format, and the second memory section **64** stores

values according to 2 (q/4) for integer values of p in the range from 0 to 3 in the fixed point format.

[0184] Finally, the second bit shift section **69** shifts the value output from the multiplication section **68** by a number of bits according to the value of (y1+9+3), when the determination result from the determination section **61** is true, and by (y1+9+3+4*1) bits, when the determination result from the determination section **61** is false, and i=1 i.e., when x is greater than **29** and less than or equal to **212**. The effect of this operation is as follows. First, the operation of shifting the output value by y1 bits is a calculation corresponding to the fourth term of Expression (4), i.e., 2 int(y/4). The value y1 is a quotient of the division of y by **4** obtained by the first division section **65**.

[0185] When the determination result from the determination section **61** is true, the operation of shifting the output value by (y1+9+3) bits is performed for the following reason. When x is less than or equal to **29**, the value of x, which is originally an integer, is normalized with **29** by the first bit shift section **62** so as to be converted to a value which does not exceed 1.0. The respective values stored in the first memory section **63**, which are originally less than or equal to 2 (9/3), are also converted to values which do not exceed 1.0. In other words, the output value from the multiplication section **68** is multiplied by 2(y1+9+3), i.e., shifted up by (y1+9+3) bits, so as to convert the output value to a value as represented in the original integer format.

[0186] When x is greater than **29** and less than or equal to **212**, the value of x, which is originally an integer, is normalized with 2 (9/3) by the first bit shift section **62** so as to be converted to a value which does not exceed 1.0. As described above, the respective values stored in the first memory section **63**, which are originally less than or equal to 2 (9/3), are also normalized with 2 3 so as to be converted to values which do not exceed 1.0. Moreover, the second division section **67** calculates the input value of x divided by 8 so as to reference the first memory section **63** (a table for exponential calculation with an exponent of ⅓) using the calculated value as an address. Therefore, the value output from the first memory section **63** is one half of x̂ (⅓), because Expression (5) below holds true.

$$(x/8)\text{^}(1/3) = x\text{^}(1/3)*(1/8)\text{^}(1/3) \qquad (5)$$
$$= x\text{^}(1/3)*(1/2)$$

[0187] The multiplication of the output value from the multiplication section **68** by 2 (9+3+3+1) is equivalent to shifting up the output value by (9+3+3+1)=16 bits. This bit shift operation converts the output value from the multiplication section **68** to a value in the original integer representation.

[0188] For these reasons, the second bit shift section **69** shifts the output value from the multiplication section **68** by (y1+N+M) bits, when the determination result from the determination section **61** is true, and by (y1+N+M+a*1) bits, when the determination result from the determination section **61** is false, and i=1. It should be understood that since the magnitude correlation among these values in such a numerical representation is relative, the operation to be performed by the second bit shift section **69** is for, when the determination result from the determination section **61** is false, adjusting the output of the multiplication section **68** so as to be greater than that when the determination result is true by a*i=4*1=4 bits.

[0189] As described above, in the present example, it is possible to perform a calculation using a small-capacity table as that illustrated in **FIG. 21** only by a simple bit shift operation even when x has a large value. Therefore, it is possible to rapidly calculate x̂ (4/3)*2 (y/4) for any value of the quantization code x.

[0190] In the present example, the quantization code x is assumed to range from 0 to 4095, whereas it ranges from 0 to 8191 according to the MPEG 2 AAC standard. A method of performing a calculation in the case where the quantization code x is 4096 or greater, while assuming i=2. It should be noted herein that 4096 is 2 (9+3).

[0191] When the input value of x is equal to or greater than 2 (9+3), the first bit shift section **62** first converts x to x1 having the fixed point format by shifting x by D−(N+b*i)= 30−(9+3*2) bits. The second division section **67** obtains x2 which is an integer value of x̂ (2 (3*2)) for the given value of x. Then, the multiplication section **68** multiplies together a value from address x2 of the first memory section **63**, a value from address y2 of the second memory section **64** and x1. Finally, the second bit shift section **69** shifts the output value from the multiplication section **68** by (y1 +N+M+ a*i)=(y1+9+3+4*2) bits.

[0192] The effect of the operation by the second bit shift section **69** of shifting the output value of the multiplication section **68** by (y1+N+M+8) bits when x is equal to or greater than 2 (N+3) is as follows. First, the operation of shifting by y1 bits corresponds to the calculation of 2 int(y/4), i.e., the fourth term of Expression (4). This is because y1 is an output from the division section **65**, and the value thereof is a quotient of the division of y by 4.

[0193] When x is greater than 2 (N+3), x which is originally an integer is normalized with 2 (N+6) by the first bit shift section **62** so as to be converted to a value which does not exceed 1.0. As described above, the respective values stored in the first memory section **63**, which are originally less than or equal to 2 (N/3), are also normalized with 2 M so as to be converted to values which do not exceed 1.0. Moreover, the second division section **67** calculates x divided by 64 so as to reference the first memory section **63** using the quotient as an address. Therefore, the value output from the first memory section **63** is one quarter of x̂ (⅓), because Expression (6) below holds true.

$$(x/64)\text{^}(1/3) = x\text{^}(1/3)*(1/64)\text{^}(1/3) \qquad (6)$$
$$= x\text{^}(1/3)*(1/4)$$

[0194] The multiplication of the output value from the multiplication section **68** by 2 (N+M+6+2) is equivalent to shifting up the output value from multiplication section **68** by (N+M+6+2) bits. This bit shift operation converts the output value from the multiplication section **68** to a value in the original integer representation.

[0195] For these reasons, the second bit shift section **69** shifts the output value from the multiplication section **68** by

(y1+N+M+8)bits, when x is greater than 2 (N+3). It should be understood that since the magnitude correlation among these values in such a numerical representation is relative, the operation to be performed by the second bit shift section **69** is for, when x is greater than 2 (N+3), adjusting the output of the multiplication section **68** so as to be greater than that when x is less than 2 N by 8 bits.

EXAMPLE 7

[**0196**] A decoding device according to Example 7 of the present invention will now be described with reference to the figures. The decoding device of this example also receives a bit stream encoded based on the MPEG 2 audio decoding standard, isolates the quantization code x and the scaling factor y contained therein, and calculates Expression (3) below which is a generalized form of Expression (1).

$$\hat{x}\ (a/b)*2\ (y/c) \tag{3}$$

[**0197**] In the present example, the variable range of the quantization code x is from 0 to 8191 , and the variable range of the scaling factor y is from −100 to 156, for example. In the present example, Expression (3) is converted to Expression (4) below.

$$x*\hat{x}\ ((a-b)/b)*2\ ((y\%c)/c)*2\ \mathrm{int}(y/c) \tag{4}$$

[**0198**] An operation of decoding an input bit stream by using Expression (4) and a memory section requiring even smaller memory capacity will be described.

[**0199**] **FIG. 22** is a block diagram illustrating a structure of a decoding device **700** according to Example 6 of the present invention. The decoding device **700** includes an isolation section **70**, a determination section **71**, a first bit shift section **72**, a first memory section **73**, an adjustment coefficient generation section **74**, a first division section **75**, a modulo calculation section **76**, a second division section **77**, a multiplication section **78** and a second bit shift section **79**.

[**0200**] Referring to **FIG. 22**, the isolation section **70** isolates the quantization code x and the scaling factor y from the input bit stream. The determination section **71** determines if the quantization code x is less than or equal to 2 N (where N is a threshold value corresponding to the number of addresses of the first memory section **73**), thereby outputting a determination result being true, if $x \leqq 2$ N, and being false, if x>2 N. When x>2 N, the determination section **71** further outputs a parameter j for specifying a value of the variable x (where j is an integer which is not negative; and the variable range of x is from 2 N to 2 (N+j)). The value of j is determined so that x satisfies $2^{(N+(j-1)}<x\leqq 2^{(N+j)}$ when x is greater than 2 N, while j is set to 0 when x is less than or equal to 2 N. The determination result from the determination section **71** and the value of the parameter j are provided to the second division section **77**, the first bit shift section **72**, the second bit shift section **79** and the adjustment coefficient generation section **74**.

[**0201**] Assuming that the multiplication section **78**, which will be described later in greater detail, functions to perform a calculation for an input value having a bit width of (D+2) in the fixed point format, where the most significant bit of the input value indicates whether the value is positive or negative, and the second most significant bit and the D bits to the right of the decimal point represent the significant digits of the input value. Then, the first bit shift section **72** converts the integer x to x1 which is a value having the fixed point format by shifting x by D−N bits, if the determination result from the determination section **71** is true, and by D−(N+j) bits (utilizing the value of j), if the determination result from the determination section **71** is false.

[**0202**] The first division section **75** performs a division of the value of y output from the isolation section **70** by c so as to output the quotient y1 thereof. The modulo calculation section **76** performs a division of the value of y output from the isolation section **70** by c so as to output the remainder y2 thereof. The second division section **77** receives the value of x which has been isolated by the isolation section **70**, so as to output the value of x, without alteration, as x2, if the determination result from the determination section **71** is true, and to output an integer value of x/(2 j) as x3, if the determination result from the determination section **71** is false.

[**0203**] The first memory section **73** stores values of H(q) for values of q (q is an integer value in the range from 0 to 2 N). The values of H(q) are obtained by converting values of q ((a−b) /b) into the fixed point format by shifting the values of q ((a−b)/b) by (D−M) bits (where M is an integer value of N*(a−b) divided by b).

[**0204**] The adjustment coefficient generation section **74** generates in the fixed point format values of $2^{(p/c)*2}$ (j/b) as I(p,j) for the value of y2 (y2=p) output from the modulo calculation section **76** and the value of j. The adjustment coefficient generation section **74** has a memory section where values of I(p,j) have been stored in advance.

[**0205**] **FIG. 23** illustrates the contents of the adjustment coefficient generation section **74** of the decoding device **700**. The adjustment coefficient generation section **74** illustrated in **FIG. 23** contains values of I(p,j) where b=3 and c=4. The following should be noted herein. The maximum value of $2^{(p/4)*2}$ (j/3) is j=13−N in the present example, which is, when N=9, $2^{(¾)*2}$ (4/3), thus exceeding 2.0. Since the present example also employs the fixed point format, as illustrated in **FIG. 28**, the table for I(p,j) stores values obtained by shifting down values of $2^{(p/4)*2}$ (j/3) by 2 bits. Of course, the adjustment for the bit shift operation by 2 bits should be accounted for in an output signal. This will be described later.

[**0206**] The multiplication section **78** multiplies together a value from address x2 or x3 of the first memory section **73**, a value from address y2, j of the adjustment coefficient generation section **74** and a value of x1 output from the first bit shift section **72**. Where M is an integer value of N*(a−b) divided by b, the second bit shift section **79** shifts the output value from the multiplication section **78** by (y1+N+M) bits, if the determination result from the determination section **71** is true, and by (y1+N+M+j) bits, if the determination result from the determination section **71** is false. During this operation, the above-described adjustment for the 2 bits due to the structure illustrated in **FIG. 23** is performed, i.e., the already shifted value is further shifted up by 2 bits.

[**0207**] As in the first memory section **63** of the decoding device **600** according to Example 6 of the present invention, the first memory section **73** has a table containing values, as illustrated in **FIG. 20**. The table illustrated in **FIG. 20** contains values of H(q) in the fixed point format which are obtained by normalizing values of q ((a−b) /b) with 2 M

(where q is an integer value in the range from 0 to 2 N). In this example, the values of H(q) are obtained where a=4, b=3, N=9, M=3 and D=30. M=3 is calculated from 9*(4−3)/3. As described above, in the format of this table, the most significant bit is a sign bit, and a decimal point is between the second bit and the third bit from the most significant bit. Thus, a value of H(q) is at least −2 and at most less than 2.

[0208] As described above, **FIG. 21** illustrates a method of converting an integer value to value of a fixed point format. In the present example, N=9, M=3 and D=30, though these values may be different, of course. For example, N may alternatively be set to 10, while M is set to an integer value, 4, which is obtained by rounding up 10*(4−3)/3.

[0209] An operation of the decoding device **700** having such a structure will be described while assuming a=4, b=3, c=4, M=3, N=9 and D=30, with reference to FIGS. **20** to **23**. First, the isolation section **70** of **FIG. 22** isolates the quantization code x and the scaling factor y from the input bit stream. Then, the determination section **71** determines if the quantization codex is less than or equal to 2 9 (=512), and outputs a determination result being true, if x is less than or equal to 2 9, and being false, if x exceeds 29. The determination section **71** further outputs a value of j according to the variable range of the quantization code x.

[0210] Next, the first bit shift section **72** converts the integer value of x to x1 having the fixed point format by shifting x by (30−9) bits, if the determination result from the determination section **71** is true, and by 30−(9+j) bits, if the determination result from the determination section **71** is false.

[0211] This operation will further be described below in detail. As illustrated in **FIG. 28**, in the fixed point format of the present example, the most significant bit is a sign bit, and a decimal point is between the second bit and the third bit from the most significant bit. Thus, when x is a value which has 9 bits or less (i.e., less than or equal to 29), x can be normalized by shifting x up by (30−9)=21 so that x is 1.0 or less as represented in the fixed point format, as illustrated in **FIG. 21**.

[0212] Similarly, when an input integer value of x has 10 bits or more (e.g., 11 bits), x can be normalized by shifting x up by 30−11=19 so that x is 1.0 or less as represented in the fixed point format. Based on such a principle, the first bit shift section **72** converts the integer of x to x1 having the fixed point format by shifting x by (30−9)=21 bits, when the determination result from the determination section **71** is true, i.e., when x is less than or equal to 512. When the determination result from the determination section **71** is false, the first bit shift section **72** converts the integer value of x to x1 having the fixed point format by shifting x by a number of bits according to the number of bits of x.

[0213] The second division section **77** outputs the value of x, without alteration, as x2 , when the determination result from the determination section **71** is true, and outputs an integer value of x/(2−j) as x3, when the determination result from the determination section **71** is false, i.e., when x is greater than 2-9. The value of x3 is used as an address of the first memory section **73** because the first memory section **73** only stores values according to q (⅓) for integer values of q from 0 to 29, i.e., 512. Therefore, when the determination result from the determination section **71** is true, i.e., when x is less than or equal to 512, the value of x is output, without alteration, as x2, whereas when the determination result from the determination section **71** is false, x3 less than or equal to 512 is generated and output by calculating an integer value of x/(2 j).

[0214] The first division section **75** obtains a quotient y1 of a division of y by 4, and the modulo calculation section **76** obtains a remainder y2 of the division of y by 4. Then, the multiplication section **78** multiplies together a value of H (x2) from address x2 of the first memory section **73**, a value of I(y2,j) from address y2,j of the adjustment coefficient generation section **74** and a value x1 output from the first bit shift section **72**.

[0215] Through these operations, the multiplication section **78** provides a value corresponding to x*x̂ (⅓)*2 ((y%4)/4) according to Expression (4) which is converted from Expression (3) (a value before the number of digits, i.e., gain, thereof is adjusted). This is because the output x1 from the first bit shift section **72** represents a value of x as converted into the fixed point format, the first memory section **73** stores values corresponding to x̂ (⅓) in the fixed point format, and the adjustment coefficient generation section **74** stores values of 2 (p/4)*2 (j/3) for integer values p from 0 to 3 and values of j in the fixed point format. The value 2 (p/4) is multiplied by 2 (j/3) for the following reason. When x is greater than 2 N, the second division section **77** calculates the input value of x divided by 2 j, so as to reference the first memory section **73** (a table for exponential calculation with an exponent of ⅓) using the calculated value as an address. Therefore, the value output from the first memory section **73** is an value of x̂ (⅓) divided by 2 (j/3), because Expression (6) below holds true.

$$(x/2\hat{}j))\hat{}(1/3) = x\hat{}(1/3)*(1/(2\hat{}j))\hat{}(1/3) \qquad (6)$$
$$= x\hat{}(1/3)/2\hat{}(j/3)$$

[0216] Thus, in order to calculate x(⅓f.), the output value from the first memory section **73** has to be multiplied by 2 (j/3), and therefore values thereof are provided in the adjustment coefficient generation section **74** in advance.

[0217] Finally, the second bit shift section **79** shifts the value output from the multiplication section **78** by a number of bits according to the value of (y1+9+3), when the determination result from the determination section **71** is true, and by (y1+9+3+2) bits, when the determination result from the determination section **71** is false, and j=2, for example, i.e., when x is greater than 2 10 and less than or equal to 2-11. The effect of this operation is as follows. First, the operation of shifting the output value by y1 bits is a calculation corresponding to the fourth term of Expression (4), i.e., 2 int(y/4). The value y1 is a quotient of the division of y by 4 obtained by the first division section **75**.

[0218] When the determination result from the determination section **71** is true, the operation of shifting the output value by (y1+9+3) bits is performed for the following reason. When x is less than or equal to 2-9, the value of x, which is originally an integer, is normalized with 2-9 by the first bit shift section **72** so as to be converted to a value which does not exceed 1.0. The respective values stored in

the first memory section **73**, which are originally less than or equal to 2 (9/3), are also normalized with 2 3 so as to be converted to values which do not exceed 1.0. In other words, the output value from the multiplication section **78** is multiplied by 2 (y1+9+3), i.e., shifted up by (y1+9+3) bits, so as to convert the output value to a value as represented in the original integer format.

[**0219**] When x is greater than 29, e.g., in the range from 2 10 to 2 11, the value of x, which is originally an integer, is normalized with 2 (9+2) by the first bit shift section **72** so as to be converted to a value which does not exceed 1.0. The respective values stored in the first memory section **73**, which are originally less than or equal to 2 (9/3), are also normalized with 2 3 so as to be converted to values which do not exceed 1.0.

[**0220**] The multiplication of the output value from the multiplication section **78** by 2 (9+3+2) is equivalent to shifting up the output value from the multiplication section **78** by (9+3+2)=14 bits. This bit shift operation converts the output value from the multiplication section **78** to a value in the original integer representation.

[**0221**] For these reasons, the second bit shift section **79** shifts the output value from the multiplication section **78** by (y1+N+M) bits, when the determination result from the determination section **71** is true, and by (y1+N+M+j) bits, when the determination result from the determination section **71** is false. It should be noted herein that in the present example, the adjustment for the 2 bits due to the structure illustrated in **FIG. 23** is performed, as described above, i.e., the already shifted value is further shifted up by 2 bits.

[**0222**] It should be understood that since the magnitude correlation among these values in such a numerical representation is relative, the operation to be performed by the second bit shift section **79** is for, when the determination result from the determination section **71** is false, adjusting the output of the multiplication section **78** so as to be greater than that when the determination result is true by j bits.

[**0223**] As described above, in the present example, it is possible to perform a calculation using a small-capacity table as that illustrated in **FIG. 20** only by a simple bit shift operation even when x has a large value. Therefore, it is possible to rapidly calculate x̂ (4/3)*2 (y/4) for any value of the quantization code x. In Example 7, as compared to Example 6, the adjustment coefficient generation section **74** is required to have a larger table, but calculation can be performed with an even higher accuracy.

## EXAMPLE 8

[**0224**] A decoding device according to Example 8 of the present invention will now be described with reference to the figures. The decoding device of this example also receives a bit stream encoded based on the MPEG 2 audio decoding standard, isolates the quantization code x and the scaling factory contained therein, and calculates Expression (3) below which is a generalized form of Expression (1).

$$\hat{x}\ (a/b)*2\ (y/c) \tag{3}$$

[**0225**] In the present example, the variable range of the quantization code x is from 0 to 8191, and the variable range of the scaling factor y is from −100 to 156, for example. In the present example, Expression (3) is converted to Expression (4) below.

$$x*\hat{x}\ ((a-b)/b)*2\ ((y\%c)/c)*2\ \mathrm{int}(y/c) \tag{4}$$

[**0226**] An operation of decoding an input bit stream by using Expression (4) and a memory section requiring even smaller memory capacity will be described.

[**0227**] **FIG. 24** is a block diagram illustrating a structure of a decoding device **800** according to Example 8 of the present invention. The decoding device **800** includes an isolation section **80**, a determination section **81**, a first bit shift section **82**, a first memory section **83**, an adjustment coefficient generation section **84**, a first division section **85**, a modulo calculation section **86**, a second division section **87**, a multiplication section **88** and a second bit shift section **89**.

[**0228**] Referring to **FIG. 24**, the isolation section **80** isolates the quantization code x and the scaling factory from the input bit stream. The determination section **81** determines the number k of digits of the quantization code x represented in the binary form, and outputs the value of k. The determination section **81** provides the value of k to the second division section **87**, the first bit shift section **82**, the second bit shift section **89** and the adjustment coefficient generation section **84**.

[**0229**] Assuming that the multiplication section **88**, which will be described later in greater detail, functions to perform a calculation for an input value having a bit width of (D+2) in the fixed point format, where the most significant bit of the input value indicates whether the value is positive or negative, and the second most significant bit and the D bits to the right of the decimal point represent the significant digits of the input value. Then, the first bit shift section **82** converts the integer x to x1 which is a value having the fixed point format by shifting x by D−k bits.

[**0230**] The first division section **85** performs a division of the value of y output from the isolation section **80** by 4 so as to output the quotient y1 thereof. The modulo calculation section **86** performs a division of the value of y output from the isolation section **80** by 4 so as to output the remainder y2 thereof. The second division section **87** obtains an integer value of x/(2 (k−N)) (where N is a predetermined threshold value) for a value of x output from the isolation section **80** so as to output the obtained value as x'. In particular, the second division section **87** calculates x' by shifting x up by (N−k) bits, when N−k is positive, and by shifting x down by (k−N) bits, when N−k is negative.

[**0231**] The first memory section **83** stores values of H(q) for values of q (q is an integer value equal to or greater than 2 (N−1) and less than 2 N) using the lower (N−1) bits of q as an address. The values of H(q) are obtained by converting values of q((a−b) /b) into the fixed point format by shifting the values of q ((a−b)/b) by (D−M) bits (where M is an integer value of N*(a−b) divided by b).

[**0232**] The adjustment coefficient generation section **84** generates in the fixed point format values of 2 (p/c)*2 ((k−N)/b) as I(p,k) for the value of y2 (y2=p) output from the modulo calculation section **86** and the value of k. The adjustment coefficient generation section **84** has a memory section where values of I(p,k) have been stored in advance.

[0233] **FIG. 25** illustrates the contents of the adjustment coefficient generation section **84** of the decoding device **800**. The adjustment coefficient generation section **84** of **FIG. 25** stores values of I(p,k) where b=3 and c=4. The following should be noted herein. The maximum value of 2 (p/4)*2 ((k−N)/3)is **2** (¾)*2 (4/3) in the present example, where k=13 and N=9, thus exceeding 2.0. Since the present example also employs the fixed point format, as illustrated in **FIG. 28**, the table for I(p,k) stores values obtained by shifting down values of 2 (p/4)*2 ((k−N)/3) by 2 bits. Of course, the adjustment for the bit shift operation by 2 bits should be accounted for in an output signal. This will be described later.

[0234] The multiplication section **88** multiplies together a value from address x' of the first memory section **83**, a value from address y2, k of the adjustment coefficient generation section **84** and a value of x1 output from the first bit shift section **82**. Where M is an integer value of N*(a−b) divided by b, the second bit shift section **89** shifts the output value from the multiplication section **88** by (y1+M+k) bits based on the determination result from the determination section **81**. During this operation, the above-described adjustment for the 2 bits due to the structure illustrated in **FIG. 25** is performed, i.e., the already shifted value is further shifted up by 2 bits.

[0235] **FIG. 26** illustrates the contents of the first memory section **83** of the decoding device **800**. The table illustrated in **FIG. 26** contains values of H(q) for values of q (q is an integer value equal to or greater than 2 (N−1) and less than 2 N) using the lower (N−1) bits of q as an address. The values of H(q) are obtained by converting values of q ((a−b)/b) into the fixed point format by shifting the values of q ((a−b)/b) by (D−M) bits (where M is an integer value of N*(a−b) divided by b). In this example, the values of H(q) are obtained where a=4, b=3, N=9, M=3 and D=30. M=3 is calculated from 9*(4−3)/3. As described above, in the format of this table, the most significant bit is a sign bit, and a decimal point is between the second bit and the third bit from the most significant bit. Thus, a value of H(q) is at least −2 and at most less than 2.

[0236] It should be noted herein, in **FIG. 26, a** value of H(q) for q=256 (=2 (N−1)) is stored at address **0**, and a value of H(q) for q=511 (=2 N−1) is stored at address 255. The table contains such values for the following reason. Where k denotes the number of bits of x represented in the binary form, the second division section **87** calculates x' by shifting x up by (N−k) bits, when N−k is positive, and by shifting x down by (k−N) bits, when N−k is negative, whereby the $N^{th}$ bit of x' is always 1. Therefore, the table of **FIG. 26** can be referenced using only the lower (N−1) bits of x' as an address. As a result, the size of the table can be reduced to half.

[0237] As described above, **FIG. 21** illustrates a method of converting an integer value to value of a fixed point format. In the present example, N=9, M=3 and D=30, though these values may be different, of course. For example, N may alternatively be set to 10, while M is set to an integer value, 4, which is obtained by rounding up 10*(4−3)/3.

[0238] An operation of the decoding device **800** having such a structure will be described while assuming a=4, b=3, c=4, M=3, N=9 and D=30, with reference to **FIGS. 21 and 24** to **26**. First, the isolation section **80** of **FIG. 24** isolates the quantization code x and the scaling factor y from the input bit stream. Then, the determination section **81** determines the number k of digits of the quantization code x represented in the binary form, and outputs the value of k.

[0239] Next, the first bit shift section **82** converts the integer value of x to x1 having the fixed point format by shifting x by (30−k) bits.

[0240] This operation will further be described below in detail. As illustrated in **FIG. 28**, in the fixed point format of the present example, the most significant bit is a sign bit, and a decimal point is between the second bit and the third bit from the most significant bit. Thus, x can be normalized by shifting x up by (30−k) so that x is 1.0 or less as represented in the fixed point format, as illustrated in **FIG. 21**.

[0241] The second division section **87** obtains a value of x' by shifting x up by (N−k) bits, when N−k is positive, and shifting x down by (k−N) bits, when N−k is negative, so as to output the obtained value of x'. The lower (N−1) bits of x' are used as an address of the first memory section **83** because the $N^{th}$ bit of x' is always 1, as described above.

[0242] The first division section **85** obtains a quotient y1 of a division of y by 4, and the modulo calculation section **86** obtains a remainder y2 of the division of y by 4. Then, the multiplication section **88** multiplies together a value of H(x2 ) from address x2 (where x2 is a value represented by the lower (N−1) bits of x') of the first memory section **83**, a value of I(y2,k) from address x2, k of the adjustment coefficient generation section **84** and a value x1 output from the first bit shift section **82**.

[0243] Through these operations, the multiplication section **88** provides a value corresponding to x*x̂ (⅓)*2 ((y%4)/4) according to Expression (4) which is converted from Expression (3) (a value before the number of digits, i.e., gain, thereof is adjusted). This is because the output x1 from the first bit shift section **82** represents a value of x as converted into the fixed point format, the first memory section **83** stores values corresponding to x̂ (⅓) in the fixed point format, and the adjustment coefficient generation section **84** stores values of 2 (p/4)*2 ((k−N)/3) for integer values p from 0 to 3 and values of k in the fixed point format. The value 2 (q/4) is multiplied by 2 ((k−N)/3) for the following reason. The input value of x has been converted to x'=x/(2 (k−N)) using the number k of digits of x by the second division section **87**, and values obtained by raising x' to the power of ⅓ are stored in the first memory section **83**. Therefore, the value output from the first memory section **83** is a value of x̂ (⅓) divided by 2 ((k−N)/3), because Expression (7) below holds true.

$$(x/(2\text{^}(k-N)))\text{^}(1/3) = x\text{^}(1/3)*(1/(2\text{^}(k-N)))\text{^}(1/3) \qquad (7)$$
$$= x\text{^}(1/3)/2\text{^}((k-N)/3)$$

[0244] Thus, in order to calculate x̂ (⅓), the output value from the first memory section **83** has to be multiplied by 2 ((k−N)/3), and therefore values thereof are provided in the adjustment coefficient generation section **84** in advance.

[0245] Finally, the second bit shift section **89** shifts the value output from the multiplication section **88** by a number

of bits according to the value of (y1+M+k) based on the value of k obtained by the determination section **81**. The effect of this operation is as follows. First, the operation of shifting the output value by y1 bits is a calculation corresponding to the fourth term of Expression (4), i.e., 2 int(y4). The value y1 is a quotient of the division of y by 4 obtained by the first division section **85**.

[**0246**] Moreover, the operation of shifting the output value by (y1+M+k) bits is performed for the following reason. The value of x, which is originally an integer, is normalized with 2 k by the first bit shift section **82** so as to be converted to a value which does not exceed 1.0. The respective values stored in the first memory section **83**, which are originally less than or equal to 2 M (M=9/3 in this example), are also normalized with 2 M so as to be converted to values which do not exceed 1.0. In other words, the output value from the multiplication section **88** is multiplied by 2 (y1+M+k), i.e., shifted up by (y1+M+k) bits, so as to convert the output value to a value as represented in the original integer format.

[**0247**] It should be noted herein that in the present example the adjustment for the 2 bits due to the structure illustrated in **FIG. 25** is performed, as described above, i.e., the already shifted value is further shifted up by 2 bits.

[**0248**] It should be understood that since the magnitude correlation among these values in such a numerical representation is relative, and M is a predetermined constant, the operation to be performed by the second bit shift section **89** is to adjust the output of the multiplication section **88** according to the value of k obtained by the determination section **81** and y1 .

[**0249**] As described above, in the present example, it is possible to perform a calculation using a small-capacity table as that illustrated in **FIG. 26** only by a simple bit shift operation even when x has a large value. Therefore, it is possible to rapidly calculate x̂ (4/3)*2 (y/4) for any value of the quantization code x. In Example 8, as compared to Example 7, the adjustment coefficient generation section **84** is required to have a larger table, but the size of the table to be provided to the first memory section can be reduced.

[**0250**] The present invention provides an exponential calculation device for calculating x̂ (a/b) for an input value of x, where a small-capacity table is provided for input values of x within a predetermined range, in order to approximately calculate x̂ (a/b). Thus, in order to calculate x̂ (a/b), a table with a large capacity is not required for storing values of x̂ (a/b) for all the possible values of x contemplated. Thus, it is possible to approximately calculate x̂ (a/b) with small-scale hardware. Particularly, the accuracy in the approximate calculation can be well maintained by providing a table for input values of x' less than or equal to the threshold value A while providing a parameter j for specifying a value of the variable x such that x'=x/2 j, or by providing a table for values of x' equal to or greater than 2 (B−1) and less than 2 B−1.

[**0251**] According to the present invention, when the input value of x exceeds the threshold value A, x can be divided and converted by an easy method by shifting x down by a number of bits. This is effective particularly when x is likely to be less than or equal to the threshold value A.

[**0252**] According to the present invention, the core section holds in a table values of x' (a/b) or x' (L/b) for all the possible values of x', whereby it is possible to convert x' to z' by referencing a small-capacity table, thus reducing the cost of an exponential calculation. Particularly, when the core section has in a table values of x' (L/b), values of z' to be contained in the table are smaller (and thus a smaller number of digits), thereby further reducing the capacity of the ROM table.

[**0253**] According to the present invention, when the threshold value A is determined based on the occurrence rate of x so that x is less than or equal to A with a frequency of a predetermined value or greater, the load on the core section can be reduced without substantially lowering the quality of the decoding process for a signal.

[**0254**] According to the present invention, the output control section has a memory section storing values of 2 (j*a/b) or 2 ((k−B)*a/b), and the input value of the output control section is multiplied by a value read out from the memory section using j or k as an address so as to output the multiplied value. Thus, practice of the present invention does not substantially add to the amount of calculation required.

[**0255**] According to the present invention, where B is an integer and A=2 B, the determination section determines the value of j so that x satisfies 2 (B+(j−1))≦x<2 (B+j), when x is greater than A. Therefore, the value of j is the number of bits of x represented in the binary form minus the value of B, thus realizing the operation by the input control section by means of a simple bit shift operation.

[**0256**] According to the present invention, the core section has a memory section storing values of q (a/b) for integer values of q (where q is equal to or greater than 2 (B−1) and less than 2 B) using at least the lower (B−1) bits of q as an address. The value of z' is multiplied by a value read out from the memory section using at least the lower (B−1) bits of x' as an address, and the multiplied value is output, thereby realizing addressing by a very simple bit operation.

[**0257**] In the decoding device of the present invention, even when the value of the quantization code x is considerably smaller than the maximum value of the variable range thereof, the value of x̂ (a/b) can be retained with more significant bits up to the final stage of outputting the calculation result, whereby it is possible to calculate x(a/b)*2 (y/c) with a high accuracy.

[**0258**] In the decoding device of the present invention, even when the quantization code x has a small value, it is possible to accurately calculate x(a/b)*2 (y/c) by a processing procedure of the fixed point format (DSP). Moreover, even when the value of the quantization code x is greater than the value stored in the first memory section of a small scale, calculation can be performed using the small-scale first memory section. Therefore, it is possible to rapidly calculate x̂ (a/b)*2 (y/c) for any value of the quantization code x.

[**0259**] Various other modifications will be apparent to and can be readily made by those skilled in the art without departing from the scope and spirit of this invention. Accordingly, it is not intended that the scope of the claims appended hereto be limited to the description as set forth herein, but rather that the claims be broadly construed.

What is claimed is:

**1**. An exponential calculation device for calculating $\hat{x}$ (a/b) (where a and b are each an integer constant) for a given input value of x, comprising:

an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2−b when x>A;

a core section for outputting a value of Z'=x' (a/b); and

an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 a when x>A.

**2**. An exponential calculation device according to claim 1, wherein the input control section calculates the value of x' by shifting x down by b bits when x>A.

**3**. An exponential calculation device according to claim 1, wherein the core section stores in a table values of x' (a/b) for all possible values of x' (where 0≦x'≦A).

**4**. An exponential calculation device according to claim 1, wherein A is set to a value such that x is less than or equal to A at a frequency equal to or greater than a predetermined value.

**5**. An exponential calculation device for calculating $\hat{x}$ (a/b) (where a and b are each an integer constant) for a given input value of x, comprising:

an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 (b*n) when x>A (where n is an integer which satisfies x/2 (b*n)≦A);

a core section for outputting a value of z'=x' (a/b); and

an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 (a*n) when x>A.

**6**. An exponential calculation device according to claim 5, wherein the input control section calculates the value of x' by shifting x down by (b*n) bits when x>A.

**7**. An exponential calculation device according to claim 5, wherein the core section stores in a table values of x' (a/b) for all possible values of x' (where 0≦x'≦A).

**8**. An exponential calculation device according to claim 5, wherein A is set to a value such that x is less than or equal to A at a frequency equal to or greater than a predetermined value.

**9**. An exponential calculation device for calculating $\hat{x}$ (a/b) (where a and b are each an integer constant) for a given input value of x, comprising:

an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 b when x>A;

a core section for outputting a value of z'=x' ((a%b)/b);

an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 (a%b) when x>A; and

a multiplication section for outputting z*($\hat{x}$ (int(a/b))).

**10**. An exponential calculation device according to claim 9, wherein A is set to a value such that x is less than or equal to A at a frequency equal to or greater than a predetermined value.

**11**. An exponential calculation device for calculating $\hat{x}$ (a/b) (where a and b are each an integer constant) for a given input value of x, comprising:

a determination section for outputting a parameter j (j is an integer which is not negative) for specifying a value of x;

an input control section for outputting a value of x', wherein x'=x when x≦A (where A is a threshold value within a variable range of x) and x'=x/2 j when x>A;

a core section for outputting a value of z'=x' (a/b); and

an output control section for outputting a value of z, wherein z=z' when x≦A and z=z'*2 (j*a/b) when x>A.

**12**. An exponential calculation device according to claim 11, wherein:

the output control section includes a memory section for storing values of 2 (j*a/b) using j as an address; and

the value of z is calculated by reading out a corresponding value of 2 (j*a/b) from the memory section when x>A.

**13**. An exponential calculation device according to claim 11, wherein:

A=2 B (B is an integer constant); and

the determination section determines the value of j such that 2 (B+(j−1))≦x<2 (B+j) when x>2 b and such that j=0 when x≦2 b.

**14**. An exponential calculation device for calculating $\hat{x}$ (a/b) (where a and b are each an integer constant) for a given input value of x, comprising:

a determination section for outputting a value of k which denotes a number of digits of x represented in a binary form;

an input control section for outputting a value of x', wherein the value of x' is obtained by shifting x up by (B−k) bits when the value of (B−k) is positive and by shifting x down by (k−B) bits when the value of (B−k) is negative;

a core section for outputting a value of z'=x' (a/b); and

an output control section for outputting a value of z=z' (2 ((k−B)*a/b)).

**15**. An exponential calculation device according to claim 14, wherein:

the output control section includes a memory section for storing values of 2 ((k−B)*a/b) using k as an address; and

the value of z is calculated by reading out a corresponding value of 2 ((k−B)*a/b) from the memory section.

**16**. An exponential calculation device according to claim 14, wherein:

the core section includes a memory section for storing values of q (a/b) (where 2 (B−1)≦q<2 B, and q is an integer) using at least lower (B−1) bits of q as an address; and

the value of z' is calculated by reading out a corresponding value of q (a/b) from the memory section using the at least lower (B−1) bits of x' as an address.

**17**. A decoding device for extracting a variable quantization code x and a variable scaling factor y from an input bit stream containing the quantization code x and the scaling factor y and calculating $\hat{x}$ (a/b)*2 (y/c) (where a, b and c are each an integer constant), the device comprising:

an isolation section for isolating x and y from the input bit stream;

a first bit shift section for, when x≦2 N (where N is a predetermined threshold value), shifting x by a number of bits according to N so as to convert x to x1 which is in a fixed point format;

a first memory section for storing, for all possible values of q (where q is an integer such that 0≦q≦2 N), using q as an address, values of H(q) which have been converted to a fixed point format by shifting q ((a−b)/b) by a number of bits according to M (where M is an integer value of (N*(a−b))/b);

a second memory section for storing, for all possible values of p (where 0p<c), values of G(p)=2 (p/c) in a fixed point format;

a division section for calculating y1, where y1=int(y/c);

a modulo calculation section for calculating y2=y%c;

a multiplication section for outputting a value obtained by multiplying together a value of H(x) which has been read out from the first memory section using x as an address, a value of G(y2) which has been read out from the second memory section using y2 as an address and a value of x1 which has been converted by the first bit shift section; and

a second bit shift section for shifting the output value from the multiplication section by a number of bits according to y1 obtained by the division section.

**18**. A decoding device according to claim 17, wherein:

the number of bits according to N is (D−N) (where D is a number of significant digits of a fractional portion of a number involved in a fixed point calculation operation); and

the number of bits according to y1 is (y1+(N+M)).

**19**. A decoding device according to claim 17, wherein:

M is an integer obtained by rounding up (N*(a−b))/b.

**20**. A decoding device according to claim 17, wherein a=4, b=3 and c=4.

**21**. A decoding device according to claim 17, further comprising:

a determination section for outputting a determination result being true when x≦2 N and being false when x>2 N; and

a second division section for outputting the value of x without alteration when the determination result is true and for outputting a value obtained by dividing x by (2 (b*i)), wherein:

the first bit shift section shifts x by a number of bits according to (N+b*i) (where i is an integer which is not negative) when the determination result is false;

the first memory section stores values of H(q) for values of q which are within the variable range of x; and

the second bit shift section shifts the value output from the multiplication section by a number of bits according to y1 when the determination result is true and by a number of bits according to y1 and a*i when the determination result is false.

**22**. A decoding device according to claim 21, wherein:

the number of bits according to N is (D−N);

the number of bits according to (N+b*i) is D−(N+b*i);

the number of bits according to M is (D−M);

the number of bits according to y1 is (y1+N+M); and

the number of bits according to y1 and a*i is (y1+N+M+a*i).

**23**. A decoding device according to claim 21, wherein the value of i is determined so that x satisfies 2 (N+b(i−1))<x≦2 (N+bi).

**24**. A decoding device according to claim 21, further comprising a determination section for outputting a determination result being true when x≦2 N and being false when x>2 N, and further outputting a parameter i for specifying a value of x, wherein:

the first bit shift section and the second division section and the second shifting section operate according to the determination result from the determination section.

**25**. A decoding device according to claim 24, wherein the second bit shift section shifts the value output from the multiplication section by a number of bits according to y1 when the determination result is true and by a number of bits according to y1+(a*i) when the determination result is false.

**26**. A decoding device for extracting a variable quantization code x and a variable scaling factor y from an input bit stream containing the quantization code x and the scaling factor y and calculating x̂ (a/b)*2 (y/c) (where a, b and c are each an integer constant), the device comprising:

an isolation section for isolating x and y from the input bit stream;

a determination section for outputting a determination result being true when x≦2 N (where N is a predetermined threshold value) and being false when x>2 N, and further outputting a parameter j for specifying a value of x;

a first bit shift section for shifting x by a number of bits according to N when the determination result is true and by a number of bits according to (N+j) when the determination result is false so as to convert x to x1 which is in a fixed point format;

a first division section for calculating y1, where y1=int(y/c);

a second division section for outputting a value of x', wherein x'=x when the determination result is true and x' is an integer value of (x/(2 j));

a first memory section for storing, for all possible values of q (where q is an integer such that 0≦q≦2 N) within a variable range of x', using q as an address, values of H(q) which have been converted to a fixed point format by shifting q ((a−b)/b) by a number of bits according to M (where M is an integer value of (N*(a−b))/b);

an adjustment coefficient generation section for generating values of I(p, j) in a fixed point format according to 2 (p/c)*2 (j/b) for all possible values of p such that 0≦p<c and all possible values of j;

a modulo calculation section for calculating y2=y%c;

a multiplication section for outputting a value obtained by multiplying together a value of H(x') which has been

read out from the first memory section using x' as an address, a value of I(y2, j) corresponding to y2 and j which has been generated by the adjustment coefficient generation section and a value of x1 which has been converted by the first bit shift section; and

a second bit shift section for shifting the output value from the multiplication section by a number of bits according to y1 when x≦2 N and by a number of bits according to y1 and j when x>2 N.

**27**. A decoding device according to claim 26, wherein the adjustment coefficient generation section includes a second memory section for storing values of I(p,j) using p and j as an address.

**28**. A decoding device according to claim 26, wherein the value of j is determined so as to satisfy 2 (N+(j−1)<x≦2 (N+j) when x>2 N, whereas j=0 when x≦2 N.

**29**. A decoding device according to claim 26, wherein a=4, b=3 and c=4.

**30**. A decoding device for extracting a variable quantization code x and a variable scaling factor y from an input bit stream containing the quantization code x and the scaling factor y and calculating x̂ (a/b)*2 (y/c) (where a, b and c are each an integer constant), the device comprising:

an isolation section for isolating x and y from the input bit stream;

a determination section for determining a number k of digits of x represented in a binary form;

a second division section for outputting a value of x', wherein the value of x' is obtained by shifting x up by (N−k) bits when the value of (N−k) is positive and by shifting x down by (k−N) bits when the value of (N−k) is negative;

a first bit shift section for shifting x by a number of bits according to k so as to convert x to x1 which is in a fixed point format;

a first division section for calculating y1, where y1=int(y/c);

a first memory section for storing, using at least lower (N−1) bits of q (where q is an integer such that 2 (N−1)≦q<2 N) as an address, values of H(q) which have been converted to a fixed point format by shifting q ((a−b)/b) by a number of bits according to M (where M is an integer value of (N*(a−b))/b);

an adjustment coefficient generation section for generating values of I (p,k) in a fixed point format according to 2 (p/c)*2 ((k−N)/b) for all possible values of p such that 0≦p<c and all possible values of k;

a modulo calculation section for calculating y2=y%c;

a multiplication section for outputting a value obtained by multiplying together a value of H(x') which has been read out from the first memory section using the at least lower (N−1) bits of x' as an address, a value of I(y2,k) corresponding to y2 and k which has been generated by the adjustment coefficient generation section and a value of x1 which has been converted by the first bit shift section; and

a second bit shift section for shifting the output value from the multiplication section by a number of bits according to y1, M and k.

**31**. A decoding device according to claim **30**, wherein the adjustment coefficient generation section includes a second memory section for storing values of I(p,k) using p and k as an address.

**32**. A decoding device according to claim **30**, wherein a=4, b=3 and c=4.

* * * * *