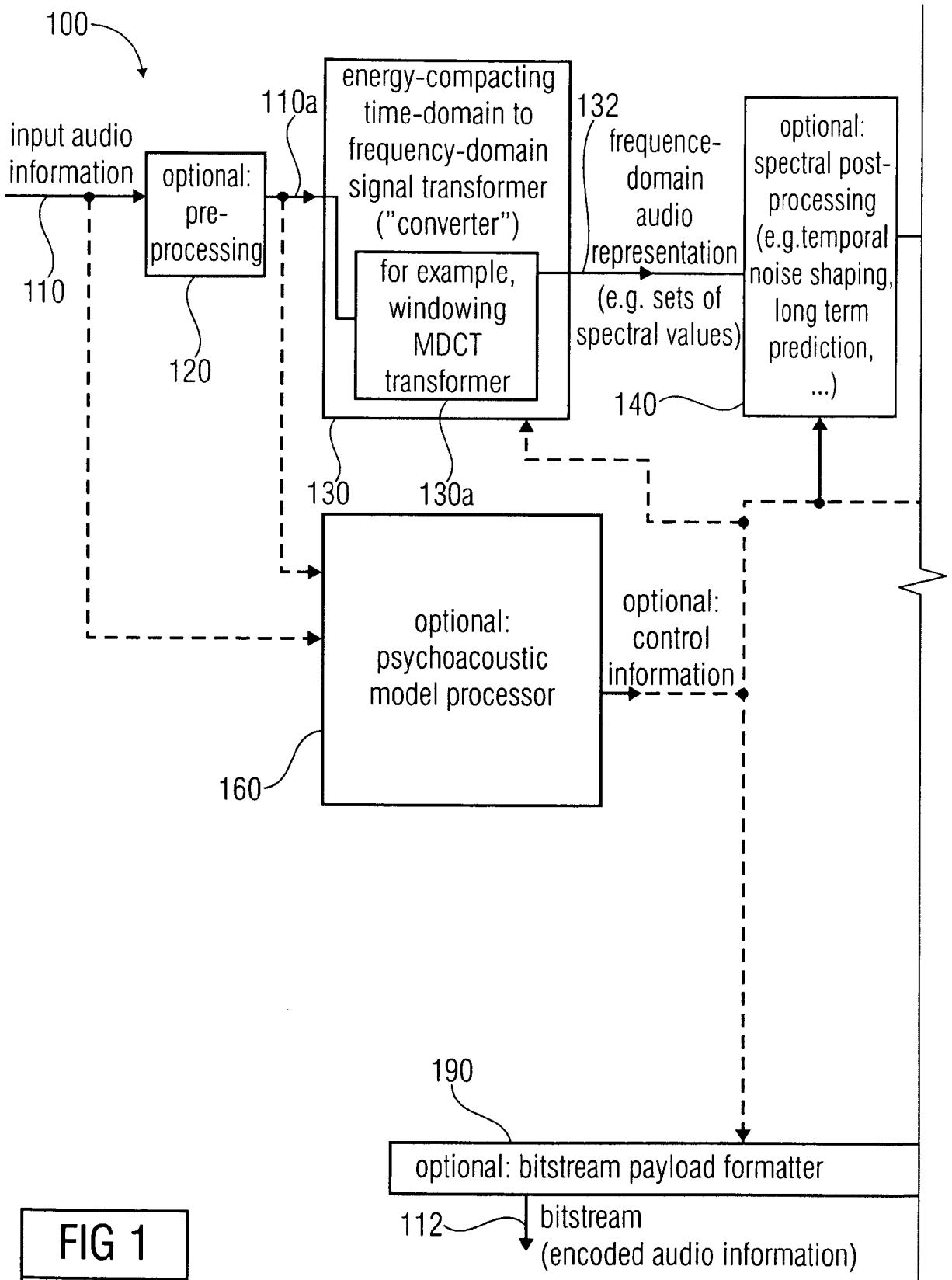


FIG 1A
AUDIO ENCODER

FIG 1	
FIG 1A	FIG 1B

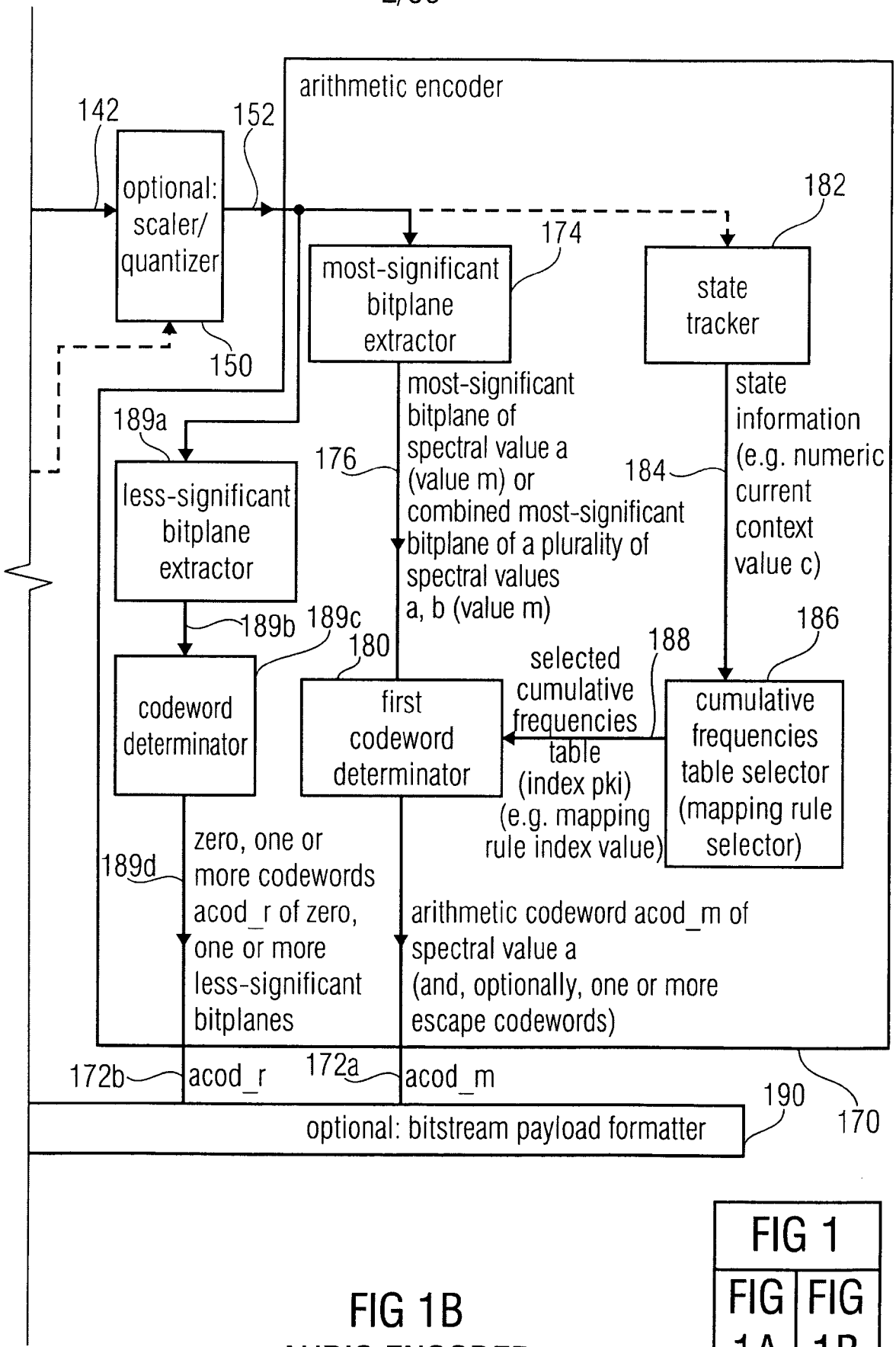


FIG 1B
AUDIO ENCODER

FIG 1

FIG 1A	FIG 1B

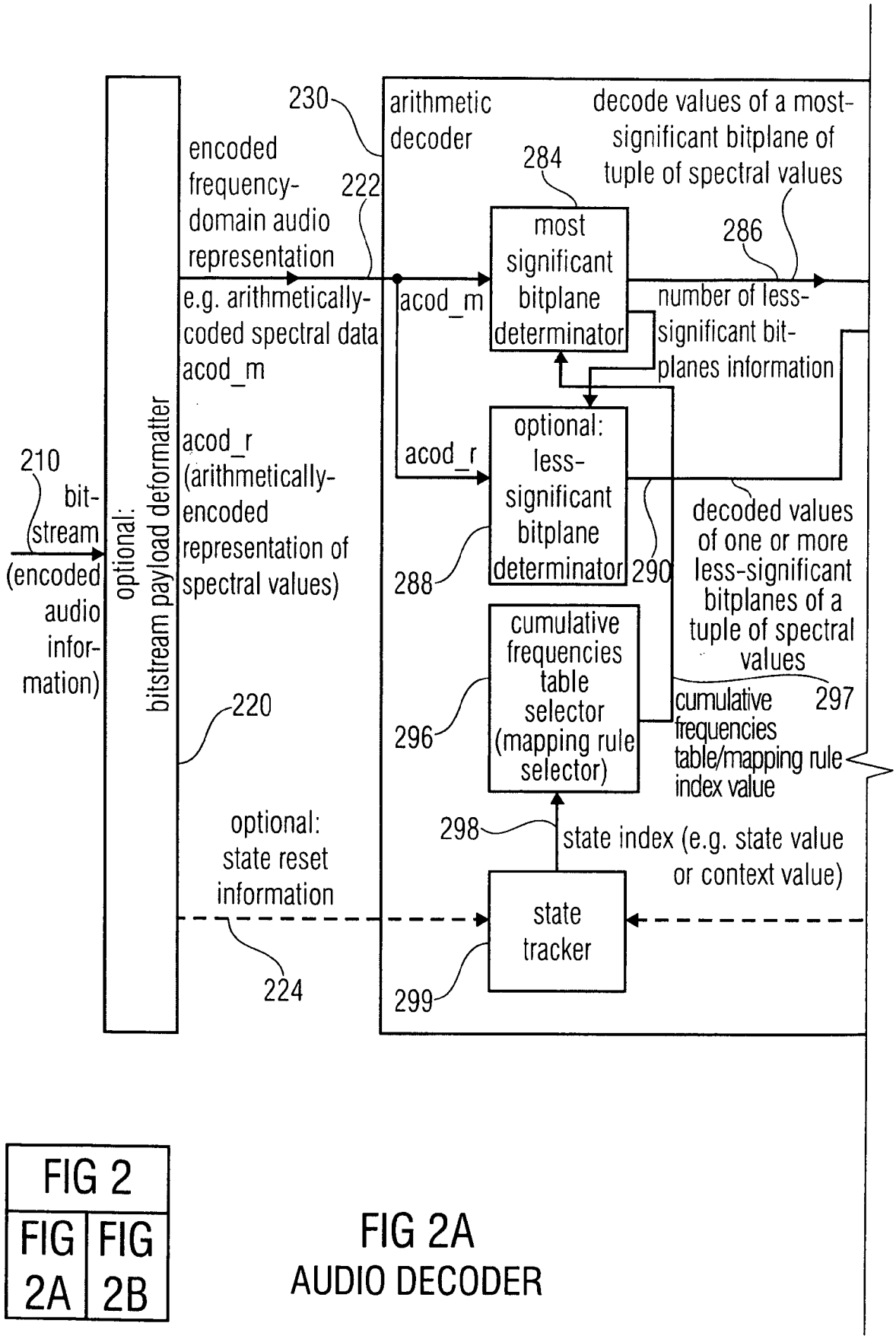


FIG 2

FIG 2A	FIG 2B

FIG 2A
AUDIO DECODER

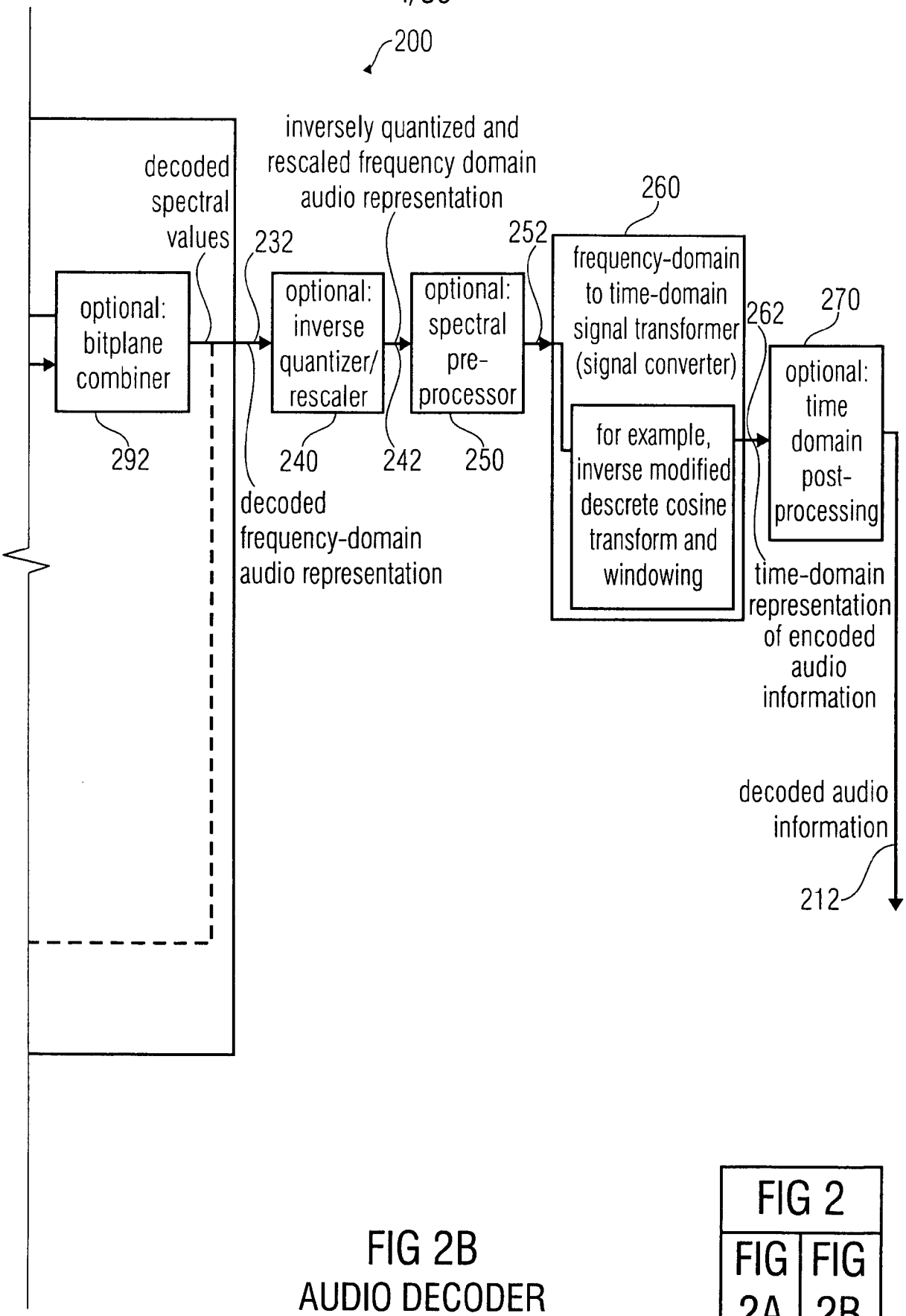


FIG 2B
AUDIO DECODER

FIG 2	
FIG 2A	FIG 2B

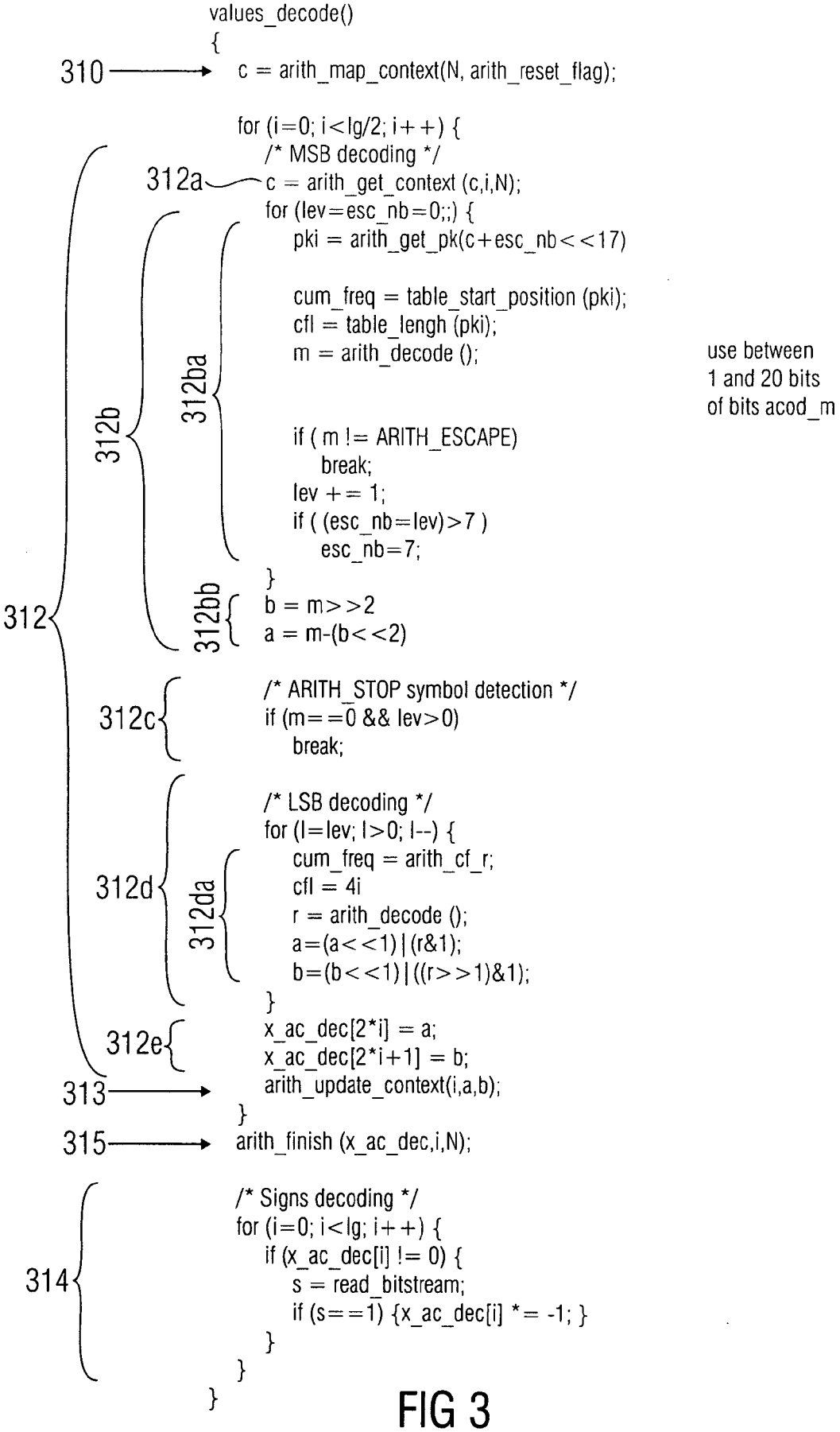

values_decode()
{
310
c = arith_map_context(N, arith_reset_flag);
for (i=0; i<lg/2; i++) {
/* MSB decoding */
312a
c = arith_get_context (c,i,N);
for (lev=esc_nb=0;;) {
312b
312ba
pki = arith_get_pk(c+esc_nb<<17)
cum_freq = table_start_position (pki);
cfl = table_lengh (pki);
m = arith_decode ();
use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

FIG 3

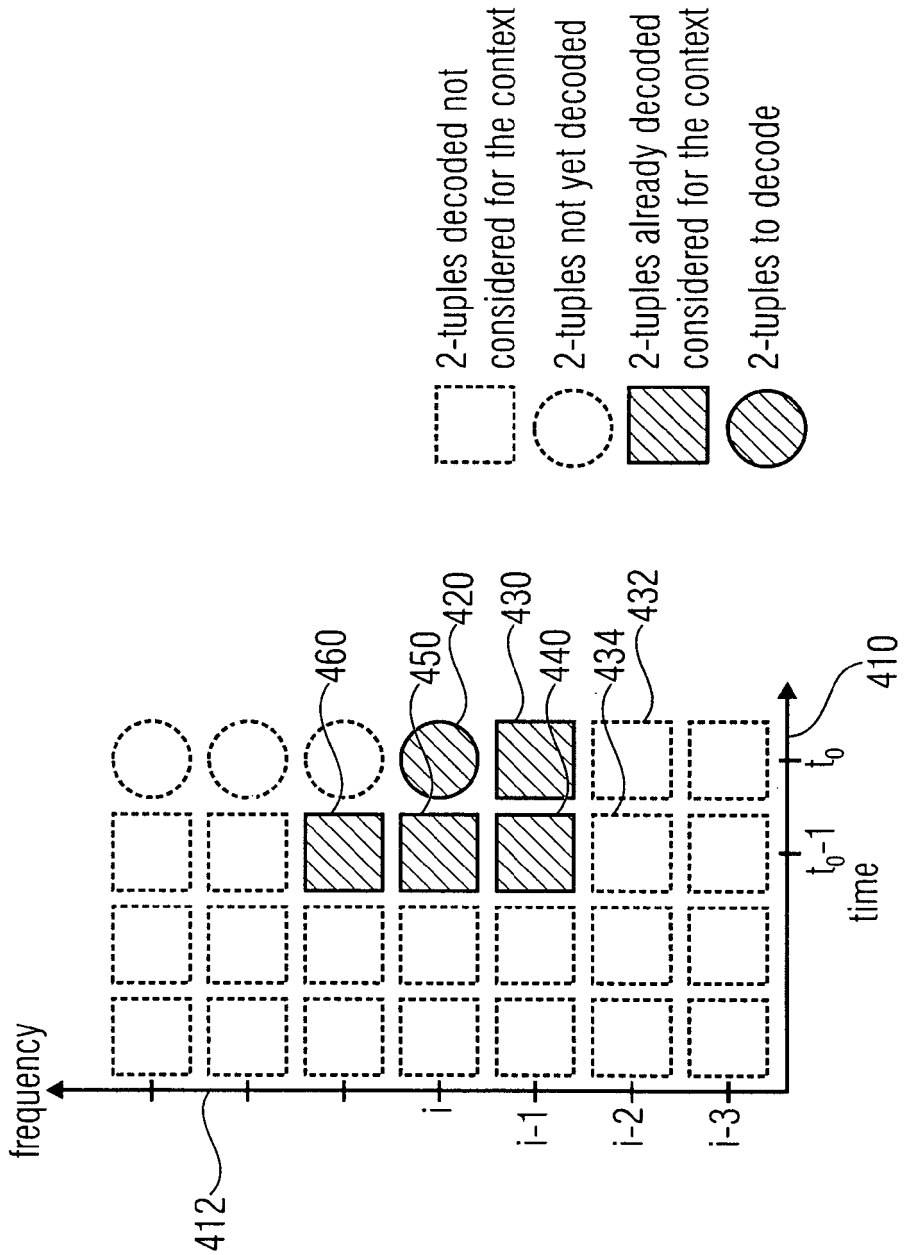


FIG 4
CONTEXT FOR THE STATE CALCULATION

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

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

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

    previous_N=N;

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

500a

500b

FIG 5A

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

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

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

   previous_lg=lg;

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

FIG 5B

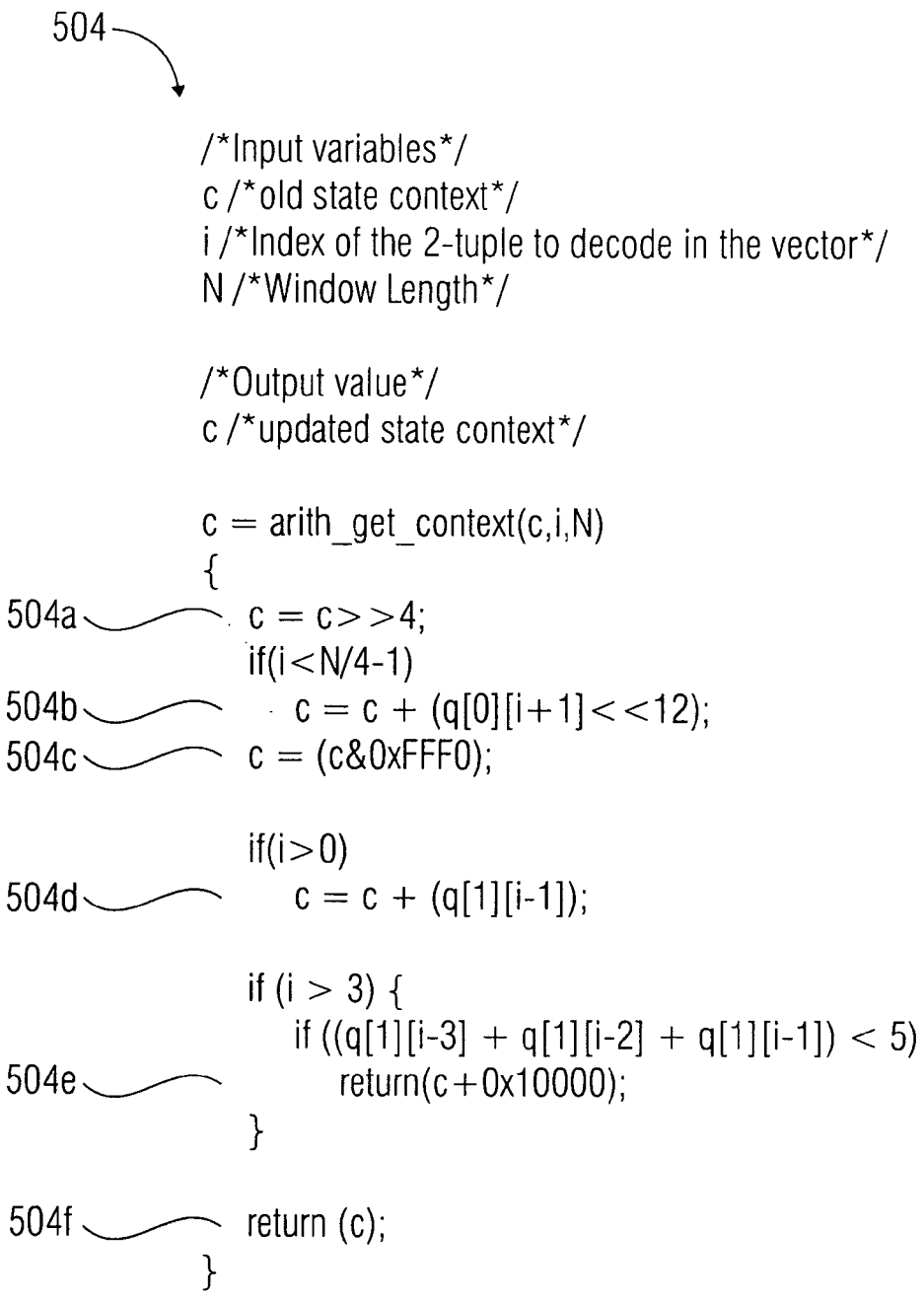

504
/*Input variables*/
c /*old state context*/
i /*Index of the 2-tuple to decode in the vector*/
N /*Window Length*/
/*Output value*/
c /*updated state context*/
c = arith_get_context(c,i,N)
{
504a
c = c>>4;
if(i<N/4-1)
504b
c = c + (q[0][i+1]<<12);
504c
c = (c&0xFFF0);
if(i>0)
504d
c = c + (q[1][i-1]);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

FIG 5C

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


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

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

    return(c);
}
```

FIG 5D

```
/*Input variable*/
c /*State of the context*/

/*Output value*/
pki /*Index of the probability model */

pki = arith_get_pk(c)
{
    i_min = -1;                                                    506a
    i = i_min;
    i_max = (sizeof(ari_lookup_m) /sizeof(ari_lookup_m[0]))-1;
    while ((i_max-i_min)>1) {                                      506b
        i = i_min+((i_max-i_min) /2);
        j = ari_hash_m[i];
        if (c<(j>>8))                                              506ba
            i_max = i;
        else if (c>(j>>8))
            i_min=i;
        else
            return(j&0xFF);
    }
return ari_lookup_m[i_max];                                        506c
}
```

FIG 5E

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

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

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

508a, 508b, 508ba, 508c

FIG 5F

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

/* global variables */
low
high
value

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

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

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

570a

570b

FIG 5G(1)

FIG 5G(1)	FIG 5G
FIG 5G(2)	

```
570c  do {
          q = p + (cfl>>1);
          if ( *q > cum ) {p=q; cfl++; }
          cfl>>=1;
      }
      while ( cfl>1 );

570d  symbol = p-cum_freq+1;
      if (symbol)
570e      high = low + (range*cum_freq[symbol-1])>>14 - 1;

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

570f  for (;;) {
570fa     if (high<32768) {}
          else if (low>=32768) {
              value -= 32768;
              low  -= 32768;
              high -= 32768;
          }
          else if (low>=16384 && high<49152) {
              value -= 16384;
              low  -= 16384;
              high -= 16384;
          }
          else break;

570fb     low += low;
          high += high+1;
          value = (value<<1) | arith_get_next_bit();
      }
      return symbol;
  }
```

FIG 5G(2)

FIG 5G(1)	FIG 5G
FIG 5G(2)	

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

/* global variables */
low
high
value

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

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

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

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

<CONTINUED IN FIG 5I>

FIG 5H

<CONTINUATION FROM FIG 5H>

```
    while ( cfl>1 );

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

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

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

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

FIG 5I

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

FIG 5J

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

FIG 5K

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

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

FIG 5L

FIG 5M

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

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

FIG 5N

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

   }
```

FIG 5O

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

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

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

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

arith_save_context(i,lg){

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

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

FIG 5P

Definitions

a,b	2-tuple to decode (2-tuple quantized coefficient to decode)
m	The most significant 2-bits wise plane of the quantized spectral coefficient to decode.
r	The least significant bit planes of the quantized spectral coefficient to decode.
lev	Level of the remaining bit-planes. It corresponds to the number of less significant bit planes.
arith_hash_m[]	Hash table mapping context states to a cumulative frequencies table index pki.
arith_lookup_m[]	Look-up table mapping group of context states to a cumulative frequencies table index pki.
arith_cf_m[pki][17]	Models of the cumulative frequencies for the most significant 2-bits wise plane m and the ARITH_ESCAPE symbol.
arith_cf_r [lsbidx][]	Cumulative frequencies for the least significant bit-planes symbol r.
arith_cf_r []	Cumulative frequencies for the least significant bit-planes symbol r
q[2][]	2-tuple context elements of the previous and current frame.
x_ace_dec[]	The decoded quantized spectral coefficients.
arith_reset_flag	Flag which indicates if the spectral noiseless context must be reset.
ARITH_STOP	Stop symbol consisting of the succession of ARITH_ESCAPE symbol and m=0. When it occurs, the rest of the frame is decoded with zero values.
N	Window length. For FD mode it is deduced from the window_sequence and for TCX N=2*lg.
previous_N	Length of the previous window.

FIG 5Q

Definitions

a,b	The 2-tuple quantized coefficient to decode
m	The most significant 2-bits wise plane of the quantized spectral coefficient to decode.
r	The most significant 2-bits wise plane of the quantized spectral coefficient to decode.
lev	Level of the remaining bit-planes. It corresponds to number the bit planes less significant than the most significant 2 bits-wise plane.
arith_hash_m[]	Hash table mapping context states to a cumulative frequencies table index pki.
arith_lookup_m[]	Look-up table mapping group of context states to a cumulative frequencies table index pki.
arith_cf_m[pki][17]	Models of the cumulative frequencies for the most significant 2-bits wise plane m and the ARITH_ESCAPE symbol.
arith_cf_r []	Cumulative frequencies for the least significant bit-planes symbol r
previous_lg	number of transmitted spectral coefficients previously decoded by the arithmetic decoder
q[2][]	The current context of 2-tuples uses for decoding the current frame.
qs[]	The past context stored for the next frame.
qdec[]	The decoded quantized spectral coefficients.
arith_reset_flag	Flag which indicates if the spectral noiseless context must be reset.
ARITH_STOP	Stop symbol consisting of the succession of ARITH_ESCAPE symbol and m=0. When it occurs, the rest of the frame is decoded with zero values.
N	Window length. For AAC it is deduced from the window_sequence (see section 6.8.3.1) and for TCX N=2.lg.

FIG 5R

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

FIG 6A

Syntax of single_channel_element()

Syntax	No. of bits	Mnemonic
single_channel_element() {		
core_mode	1	**uimsbf**

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

}		

FIG 6B

Syntax of channel_pair_element()

Syntax	No. of bits	Mnemonic
channel_pair_element()		
{		
core_mode0	**1**	**uimsbf**
core_mode1	**1**	**uimsbf**
ics_info();	optional: common ics_info for two channels	
if (core_mode0 == 1) {		
lpd_channel_stream();		
}		
else {		
fd_channel_stream();		
}		
if (core_mode1 == 1) {		
lpd_channel_stream();		
}		
else {		
fd_channel_stream();		
}		
}		

FIG 6C

Syntax of ics_info()

Syntax	No. of bits	Mnemonic
ics_info()		
{		
window_length;	1	**uimsbf**
if(window_length !=0) {		
transform_length;	1	**uimsbf**
}		
else {		
transform_length=0;		
}		
window_shape;	1	**uimsbf**
if (window_length !=0 && transform_length !=0){		optional
max_sfb;	4	**uimsbf**
scale_factor_grouping;	7	**uimsbf**
}		
else {		
max_sfb;	6	**uimsbf**
}		
}		

FIG 6D

Syntax of fd_channel_stream()

Syntax	No. of bits	Mnemonic
fd_channel_stream() {		
global_gain;	**8**	**uimsbf**
ics_info(); (unless included in channel pair element)		
scale_factor_data ();		
ac_spectral_data ();		
}		

FIG 6E

Syntax of ac_spectral_data()

Syntax	No. of bits	Mnemonic
ac_spectral_data() {		
arith_reset_flag	**1**	**uimsbf**

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

}		

FIG 6F

FIG 6G

Syntax of arith_data()

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

Table Syntax of arith_data()

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

FIG 6H

Definitions	
arith_data()	Data element to decode the spectral noiseless coder data
arith_reset_flag	Flag which indicates if the spectral noiseless context must be reset.
acod _m[pki][m]	Arithmetic codeword necessary for decoding of the most significant 2-bits wise plane m of the quantized spectral coefficients of a 2-tuple.
acod_r[lsbidx][]	Arithmetic codeword necessary for decoding of the residual bit-planes r of the quantized spectral coefficient of a 2-tuple.
s	The coded sign of the non-null spectral quantized coefficient.
Help elements	
a,b	2-tuple corresponding to quantized spectral coefficients
m	The most significant 2-bits wise plane of the 2-tuple to decode.
r	The least significant bit planes of the 2-tuple to decode.
lg	Number of quantized coefficients to decode.
N	Window length. For FD mode it is deduced from the window_sequence and for TCX N=2*lg.
i	Index of 2-tuples to decode within the frame.
pki	Index of the cumulative frequencies table used by the arithmetic decoder for decoding m.
arith_get_pk ()	Function that returns the index pki of cumulative frequencies table necessary to decode the codeword **acod_m[pki][m]**.
c	State of context
lsbidx	Index to the cumulative frequencies tables used by the arithmetic coder for decoding r.
lev	Level of bit-planes to decode beyond the most significant 2-bits wise plane.
ARITH_ESCAPE	Escape symbol that indicates additional bit-planes to decode beyond the two most significant bit planes.
esc_nb	Number of ARITH_ESCAPE symbol already decoded for the present 2-tuple. The value is bounded to 7.
x_ac_dec[]	Element holding the decoded spectral coefficients
arith_map_context()	Initializes the contexts needed for decoding the present frame.
arith_get_context()	Computes the context state for decoding the present 2-tuple m symbols.
arith_update_context()	Updates the context for the next 2-tuple.
arith_finish ()	Finish the noiseless decoding.

FIG 6I

Definitions

arith_data()	Data element to decode the spectral noiseless coder data
arith_reset_flag	Flag which indicates if the spectral noiseless context must be reset.
acod _m[pki][m]	Arithmetic codeword necessary for decoding of the most significant 2-bits wise plane m of the quantized spectral coefficients of a 2-tuple.
arith_r[]	Arithmetic codeword necessary for decoding of the residual bit-planes r of the quantized spectral coefficient of a 2-tuple.
s	The coded sign of the non-null spectral quantized coefficient.

Help elements

a,b	The 2-tuple quantized coefficients to decode
m	The most significant 2-bits wise plane of the 2-tuple to decode.
r	The least significant bit wise plane of the 2-tuple to decode.
lg	Number of quantized coefficients to decode.
i	Index of 2-tuple to decode within the frame.
pki	Index of the cumulative frequencies table used by the arithmetic decoder for decoding m.
arith_get_pk ()	Function that returns the index pki of cumulative frequencies table necessary to decode the codeword acod_m[pki][m].
c	State of context
lev	Level of bit-planes to decode beyond the most significant 2-bits wise plane.
ARITH_ESCAPE	Escape symbol that indicates additional bit-planes to decode beyond the two most significant bit planes.
esc_nb	Number of ARITH_ESCAPE symbol already decoded for the present 2-tuple. The value is bounded to 7.
arith_map_context()	Initializes the contexts needed for decoding the present frame.
arith_get_context()	Computes the context state for decoding the present 2-tuple m symbols.
arith_update_context()	Updates the context for the next 2-tuple.
arith_save_context()	Save the context for the next frame to decode.

FIG 6J

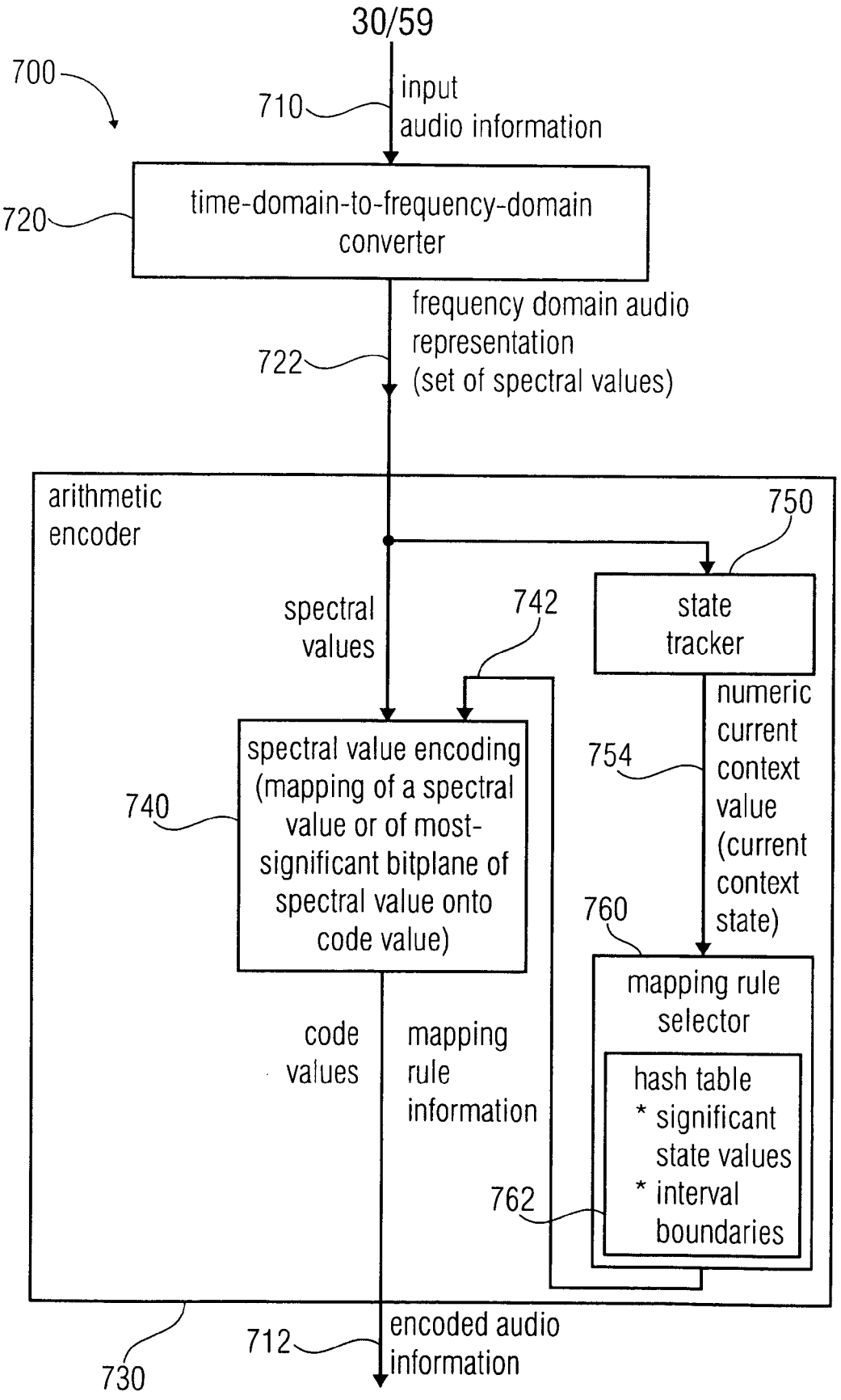

700
710
input
audio information
720
time-domain-to-frequency-domain
converter
722
frequency domain audio
representation
(set of spectral values)
arithmetic
encoder
750
spectral
values
742
state
tracker
numeric
current
context
value
(current
context
state)
754
spectral value encoding
(mapping of a spectral
value or of most-
significant bitplane of
spectral value onto
code value)
740
760
mapping rule
selector
hash table
* significant
state values
* interval
boundaries
762
code
values
mapping
rule
information
712
encoded audio
information
730

FIG 7

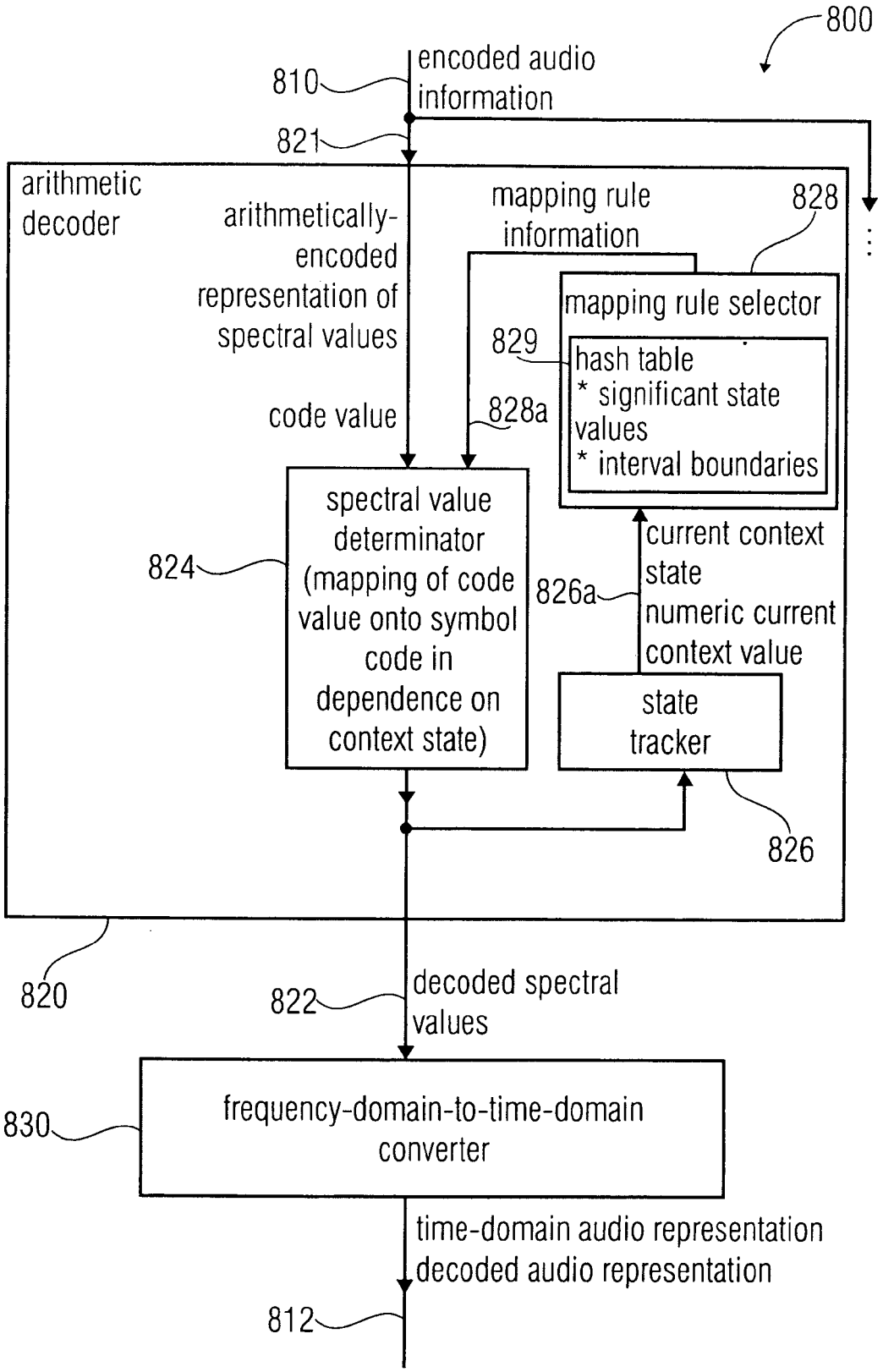
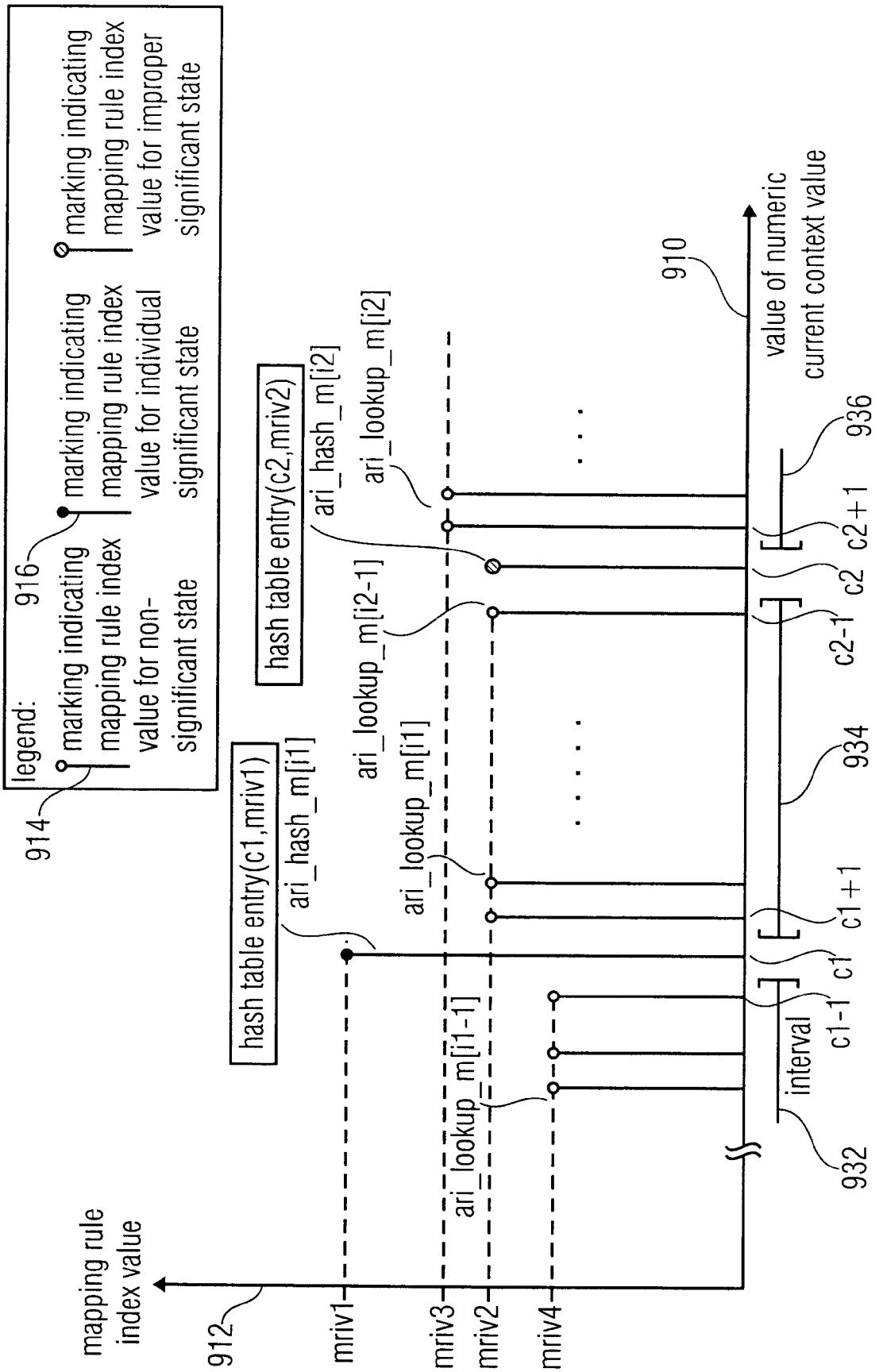

800
encoded audio information
810
821
arithmetic decoder
arithmetically-encoded representation of spectral values
code value
mapping rule information
828
mapping rule selector
829
hash table
* significant state values
* interval boundaries
828a
spectral value determinator (mapping of code value onto symbol code in dependence on context state)
824
current context state numeric current context value
826a
state tracker
826
820
822
decoded spectral values
frequency-domain-to-time-domain converter
830
time-domain audio representation decoded audio representation
812

FIG 8

FIG 9


legend:
marking indicating mapping rule index value for non-significant state
916
marking indicating mapping rule index value for individual significant state
marking indicating mapping rule index value for improper significant state
914
hash table entry(c1,mriv1)
hash table entry(c2,mriv2)
ari_hash_m[i1]
ari_hash_m[i2]
ari_lookup_m[i1]
ari_lookup_m[i2-1]
ari_lookup_m[i1-1]
910
value of numeric current context value
912
mapping rule index value
mriv1
mriv3
mriv2
mriv4
c1-1
c1
c1+1
c2-1
c2
c2+1
interval
932
934
936

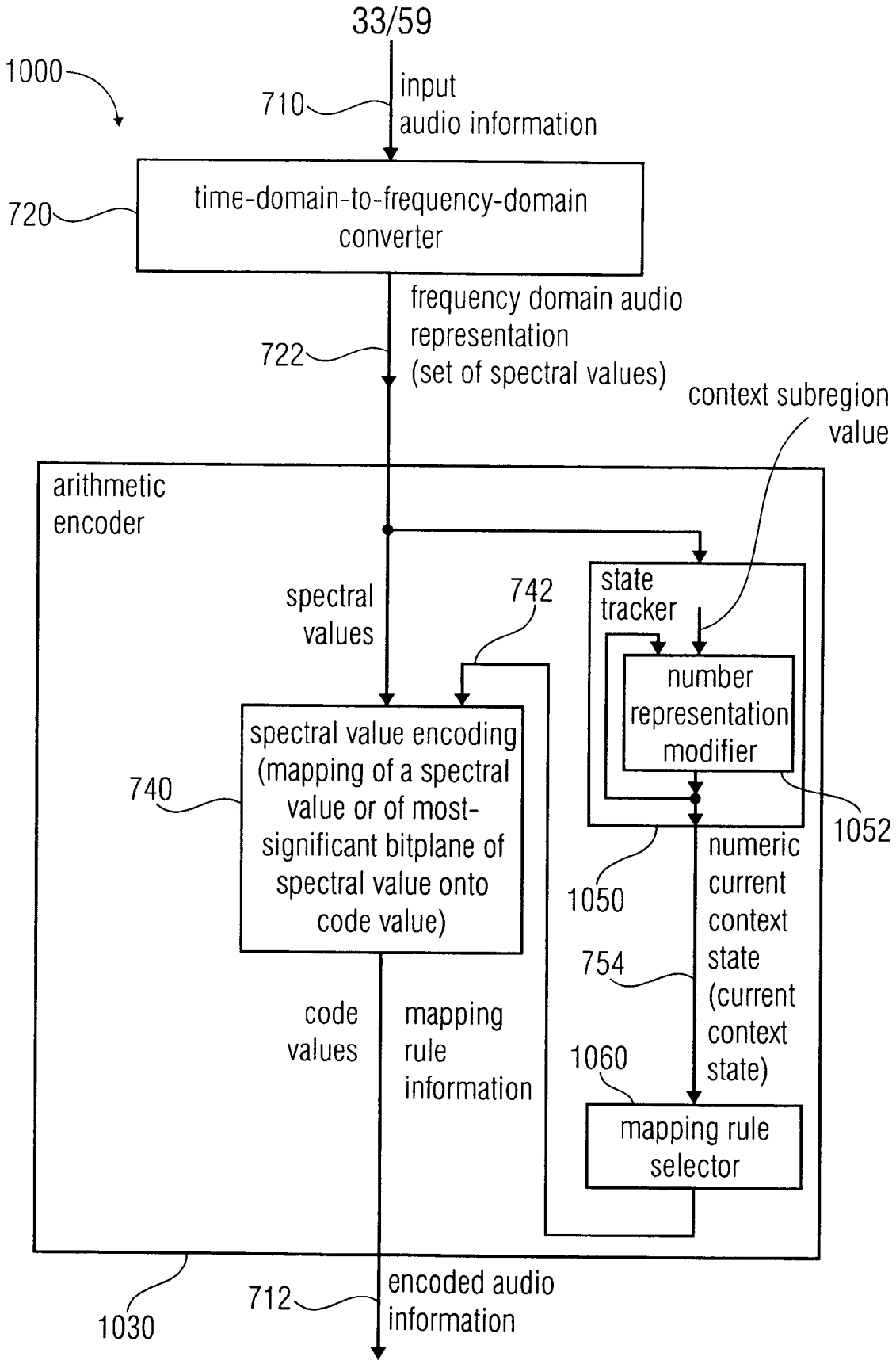

1000
710
input
audio information
720
time-domain-to-frequency-domain
converter
frequency domain audio
representation
(set of spectral values)
722
context subregion
value
arithmetic
encoder
spectral
values
742
state
tracker
number
representation
modifier
1052
spectral value encoding
(mapping of a spectral
value or of most-
significant bitplane of
spectral value onto
code value)
740
1050
numeric
current
context
state
(current
context
state)
754
1060
code
values
mapping
rule
information
mapping rule
selector
1030
712
encoded audio
information

FIG 10

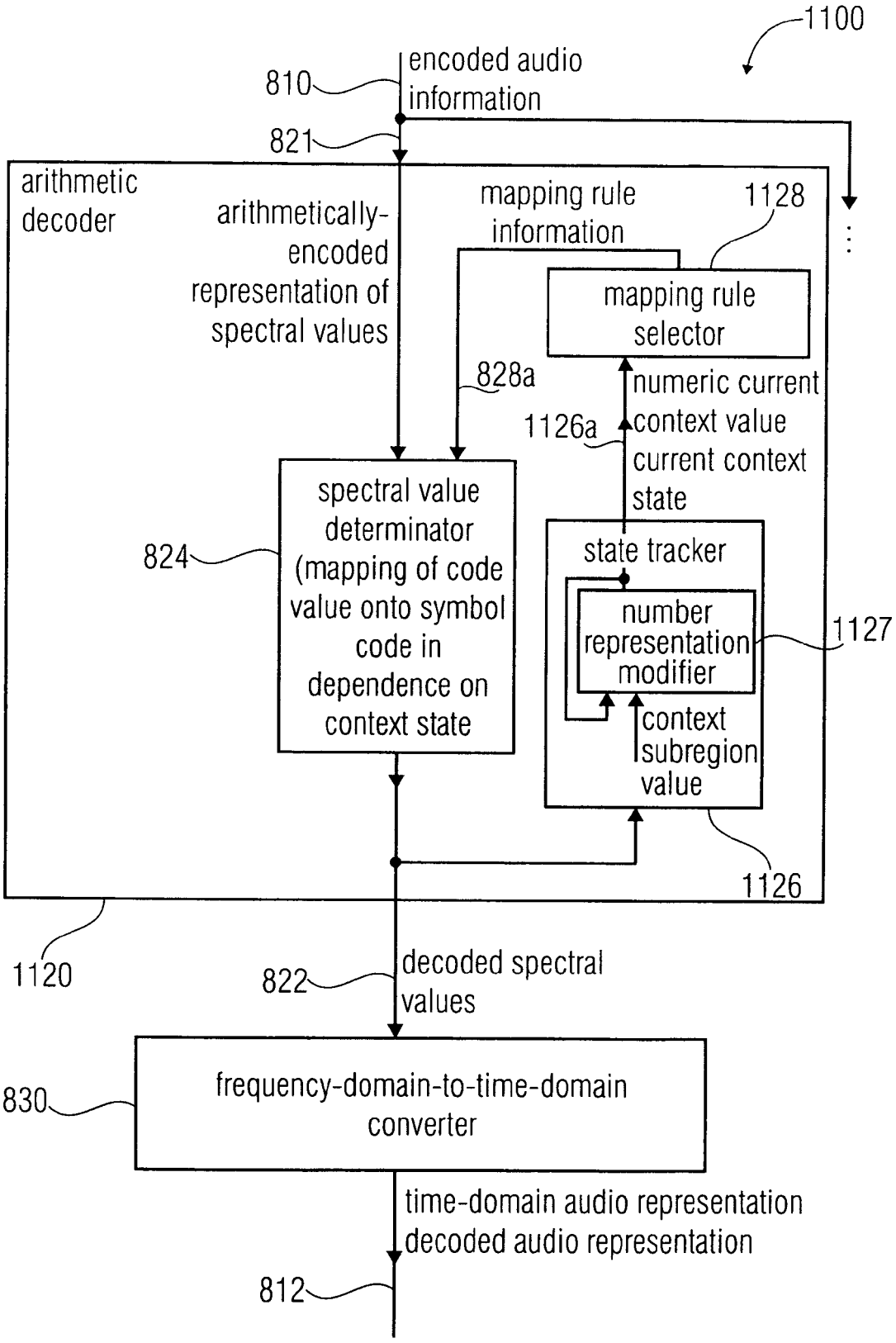

1100
encoded audio information
810
821
arithmetic decoder
arithmetically-encoded representation of spectral values
mapping rule information
1128
mapping rule selector
828a
numeric current context value
1126a
current context state
spectral value determinator (mapping of code value onto symbol code in dependence on context state
824
state tracker
number representation modifier
1127
context subregion value
1126
1120
822
decoded spectral values
830
frequency-domain-to-time-domain converter
time-domain audio representation
decoded audio representation
812

FIG 11

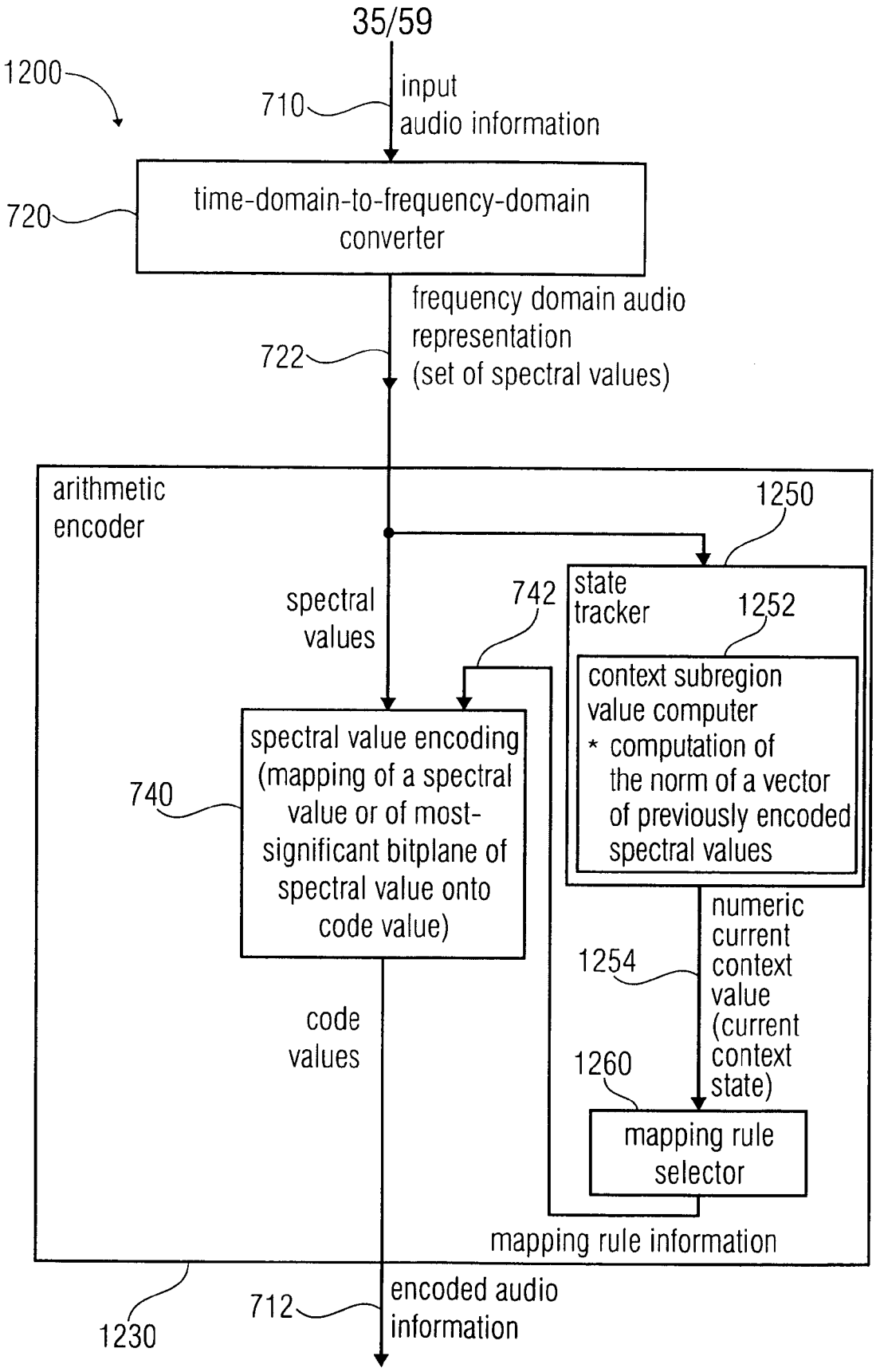

35/59
1200
710
input
audio information
720
time-domain-to-frequency-domain
converter
722
frequency domain audio
representation
(set of spectral values)
arithmetic
encoder
1250
spectral
values
742
state
tracker
1252
context subregion
value computer
* computation of
the norm of a vector
of previously encoded
spectral values
740
spectral value encoding
(mapping of a spectral
value or of most-
significant bitplane of
spectral value onto
code value)
numeric
current
context
value
(current
context
state)
1254
code
values
1260
mapping rule
selector
mapping rule information
1230
712
encoded audio
information

FIG 12

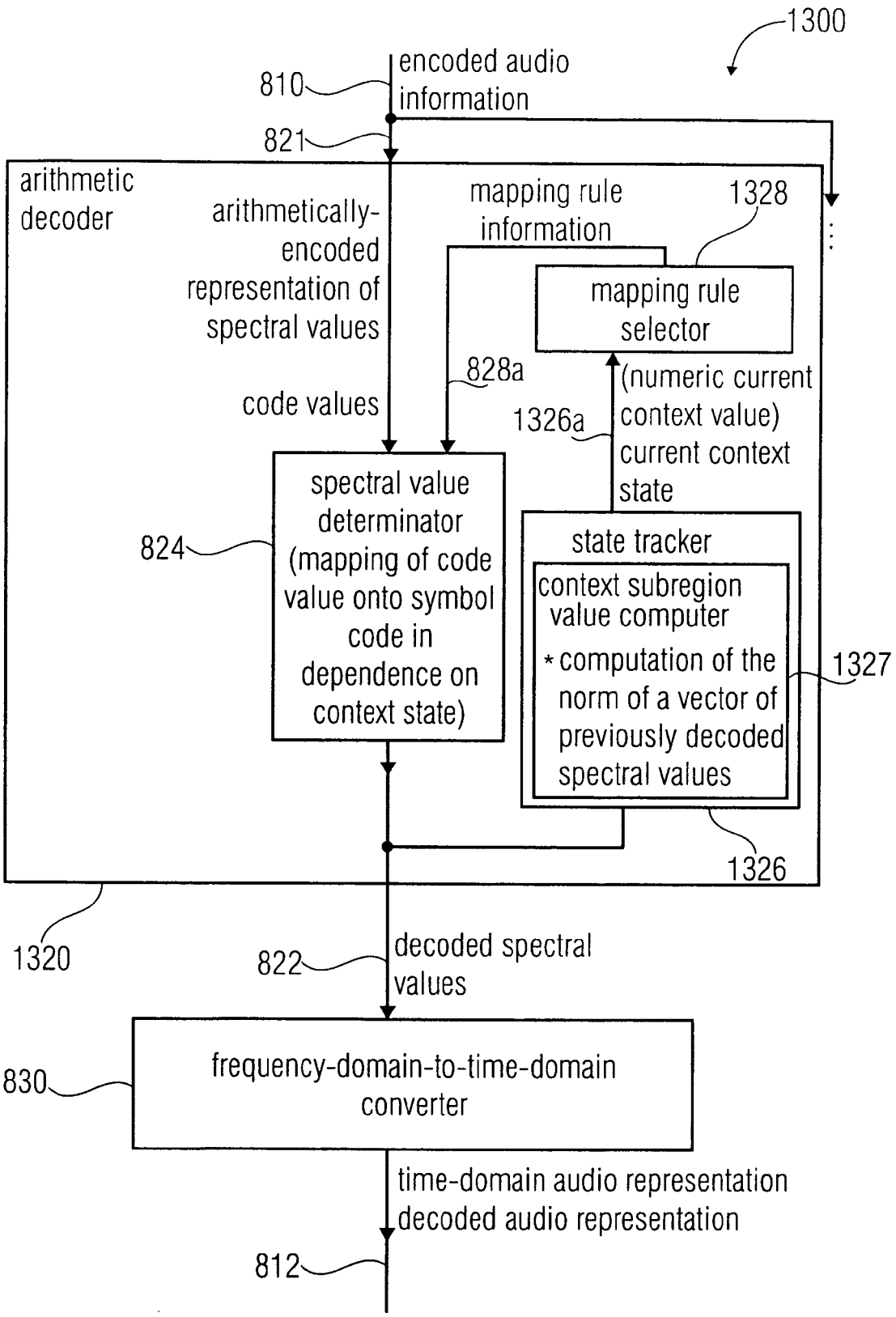

1300
encoded audio information
810
821
arithmetic decoder
arithmetically-encoded representation of spectral values
code values
mapping rule information
1328
mapping rule selector
828a
(numeric current context value) current context state
1326a
spectral value determinator (mapping of code value onto symbol code in dependence on context state)
824
state tracker
context subregion value computer
* computation of the norm of a vector of previously decoded spectral values
1327
1326
1320
822
decoded spectral values
830
frequency-domain-to-time-domain converter
time-domain audio representation
decoded audio representation
812

FIG 13

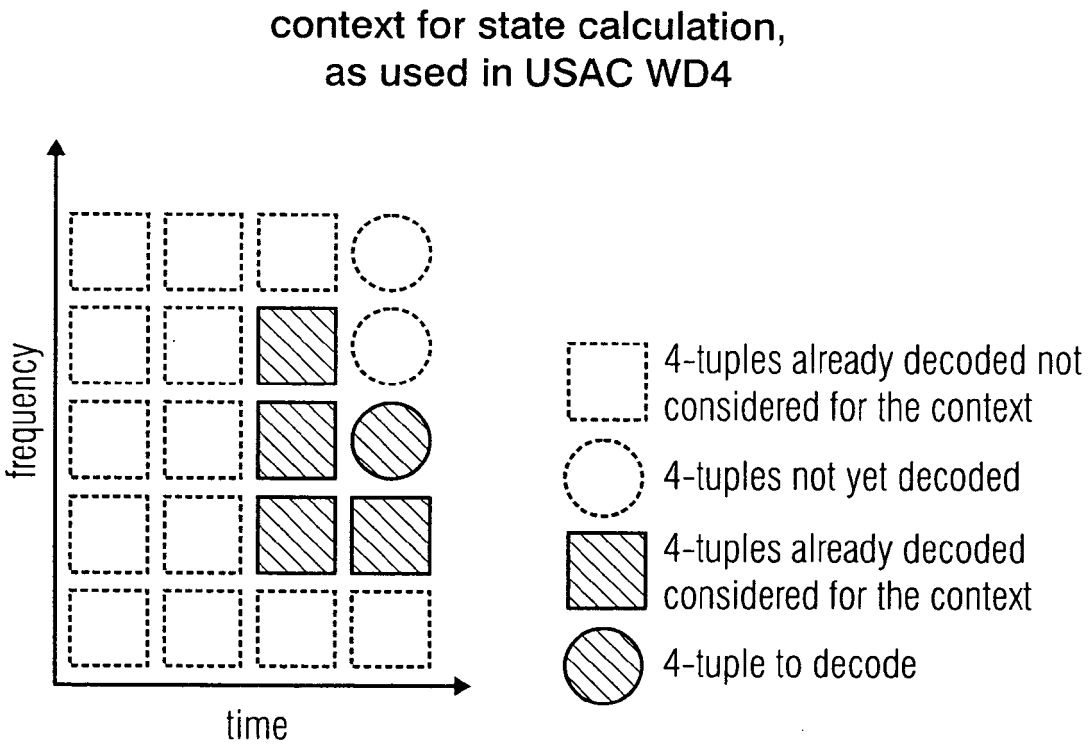

context for state calculation,
as used in USAC WD4
frequency
time
4-tuples already decoded not considered for the context
4-tuples not yet decoded
4-tuples already decoded considered for the context
4-tuple to decode

FIG 14A

TABLES AS USED IN USAC WD4 ARITHMETIC CODING SCHEME

table name	description	unit of data	memory (words of 32 Bit)
arith_cf_ng_hash[128]	Hash table mapping context to a probability model index.	word	128
arith_cf_ng[32][545]	Cumulative frequencies of groups for each probability distribution mode ,l	1/2 word	8720
egroups[8][8][8][8]	Group index of 4 tuple.	1/2 word	2048
dgvectors[4*4096]	Map group index and element index to 4-tuple.	1/4 word	4096
dgroups[544]	Map group index to cardinal of the group and offset in dgvectors	word	544
arith_cf_ne[2701]	Cumulative frequencies of the element index symbol	1/2 word	1350.5
arith_cf_r[16]	Cumulative frequencies of least significant bit planes	1/2 word	8
total			**16894.5**

FIG 14B

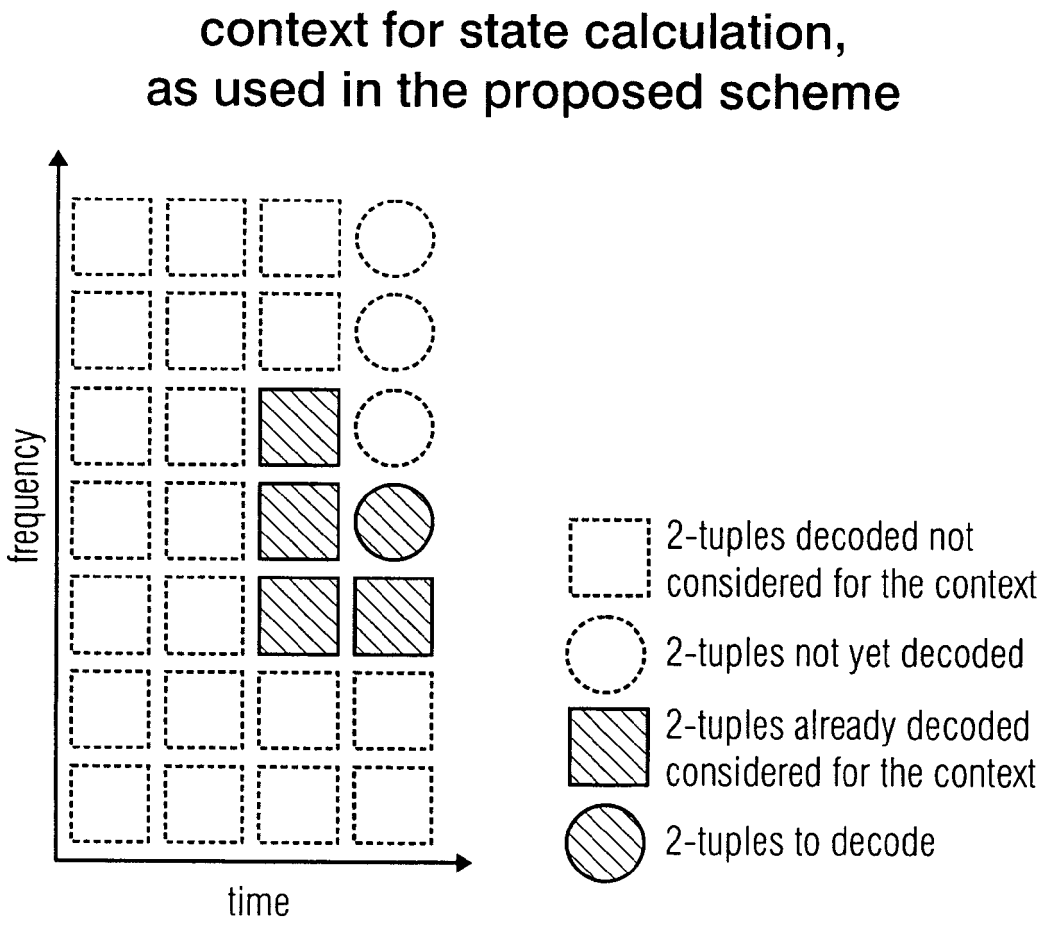

context for state calculation,
as used in the proposed scheme
frequency
time
2-tuples decoded not considered for the context
2-tuples not yet decoded
2-tuples already decoded considered for the context
2-tuples to decode

FIG 15A

table name	description	unit of data	memory (words of 32 Bit)
arith_hash[600]	Hash table mapping states of the context to a group of states	1 word	600
arith_lookup[600]	Look-up table mapping	1/4 word	150
arith_cf_msb[96][16]	groups of states to a cumulative frequencies table Models of the cumulative frequencies for the most significant 2-bits wise plane m and the ARITH_ESCAPE symbol	1/2 word	768
total			**1518**

FIG 15B
TABLES AS USED IN THE PROPOSED CODING SCHEME

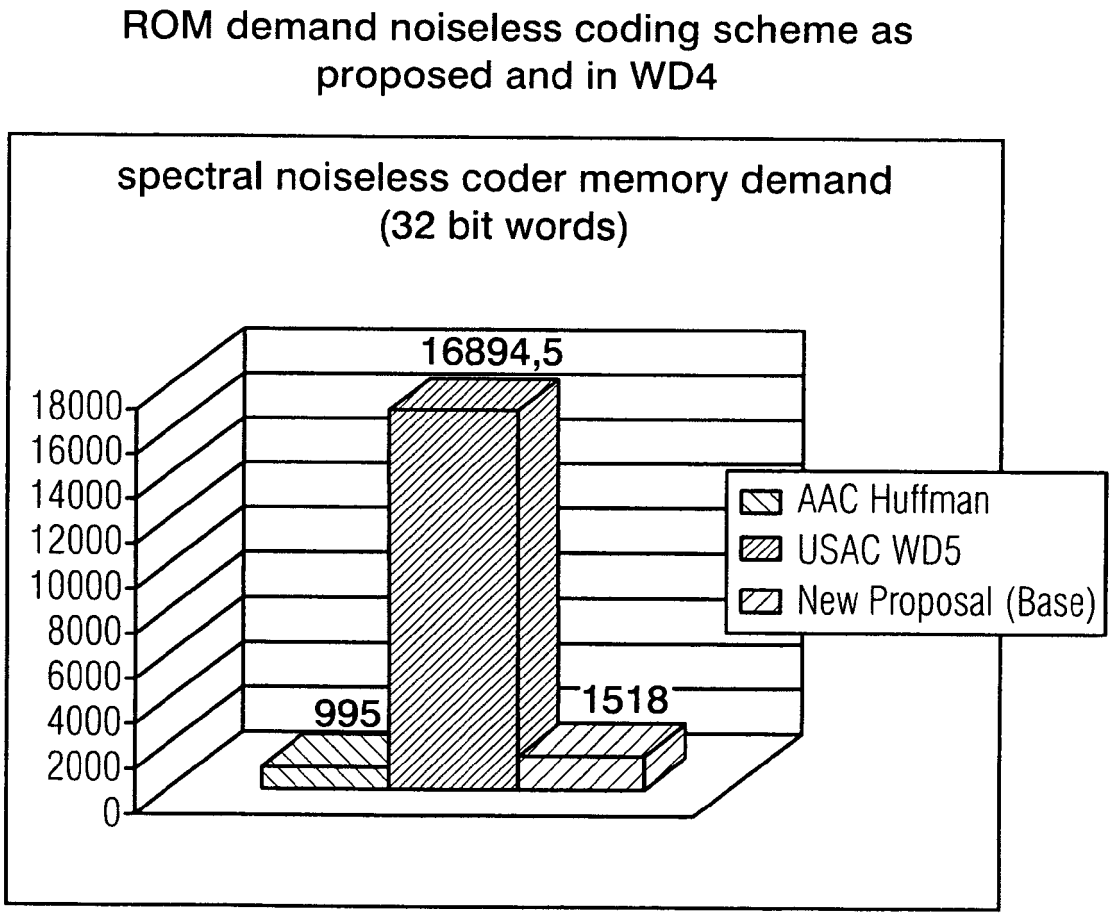

ROM demand noiseless coding scheme as proposed and in WD4
spectral noiseless coder memory demand (32 bit words)
16894,5
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
995
1518
AAC Huffman
USAC WD5
New Proposal (Base)

FIG 16A

total USAC decoder data ROM demand,
WD4 and scheme as proposed

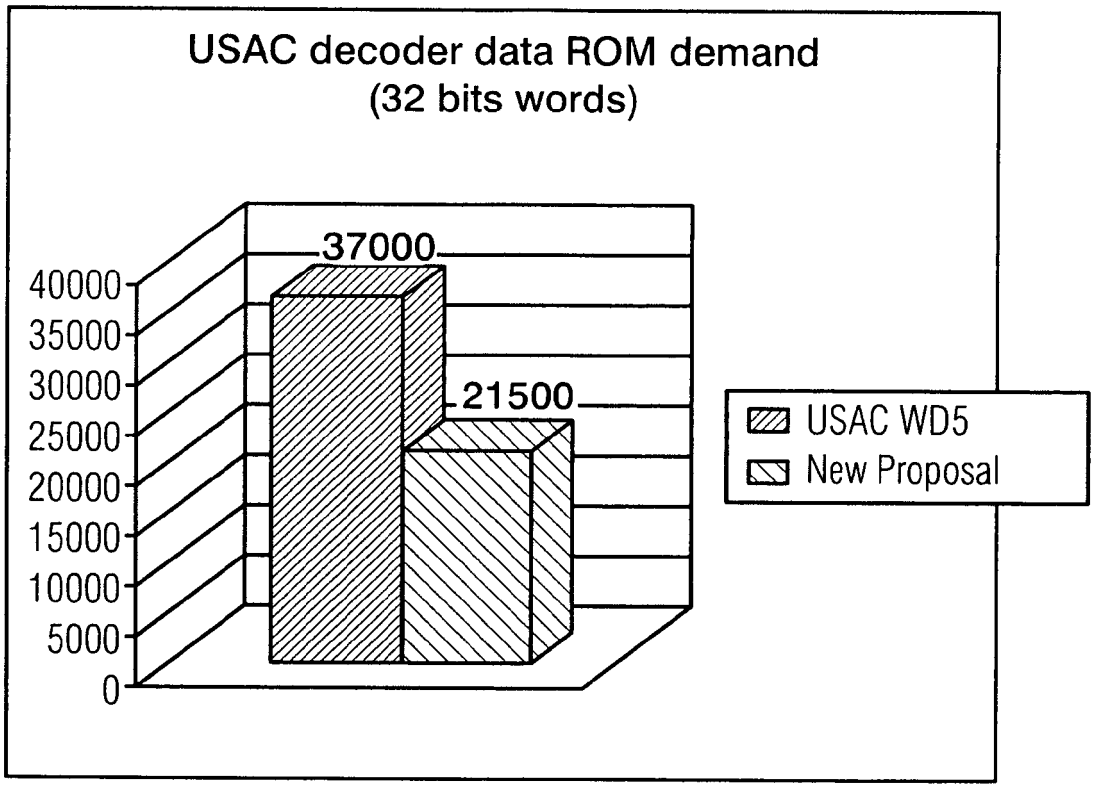


FIG 16B

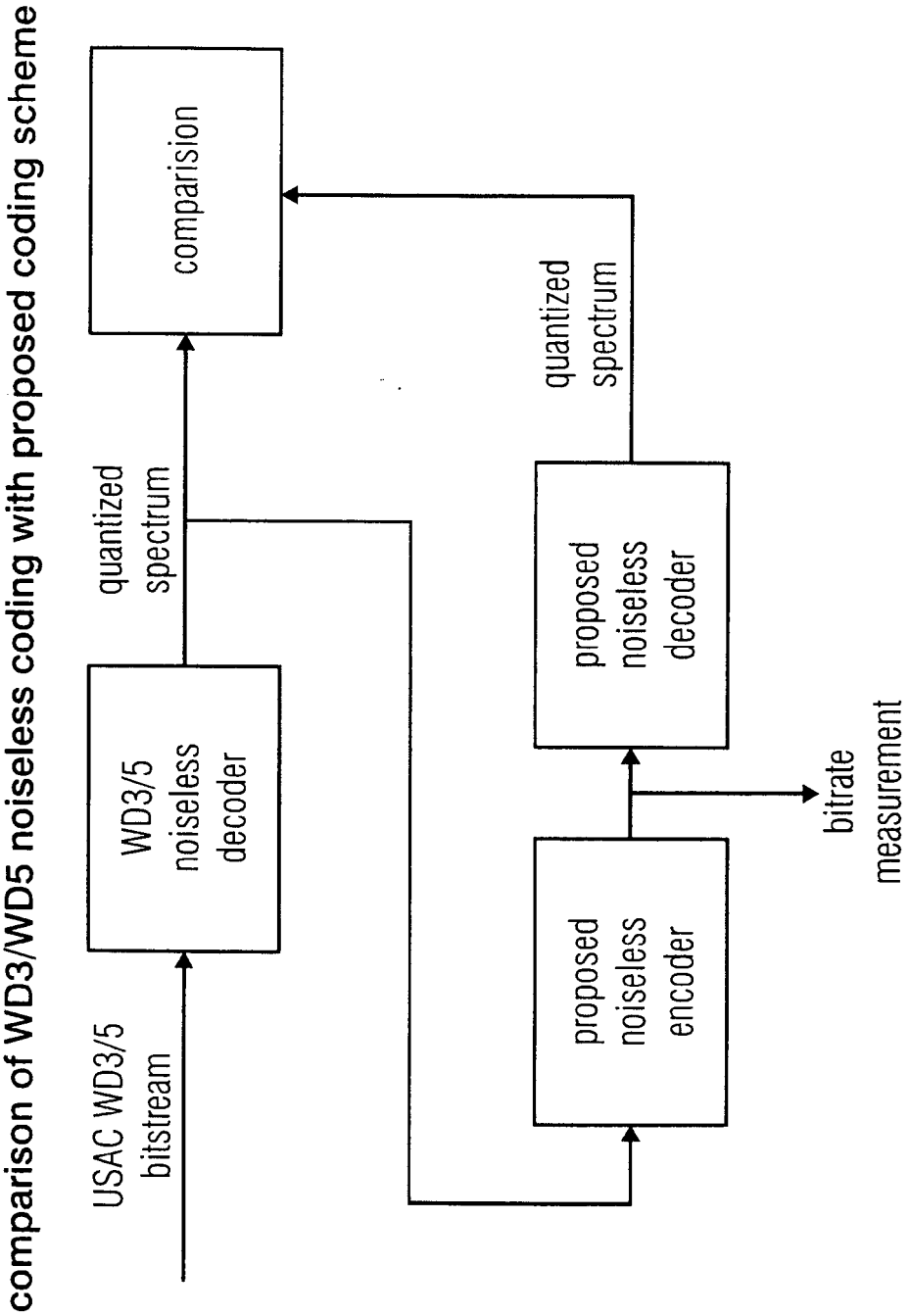

comparison of WD3/WD5 noiseless coding with proposed coding scheme
USAC WD3/5 bitstream
WD3/5 noiseless decoder
quantized spectrum
comparision
quantized spectrum
proposed noiseless decoder
proposed noiseless encoder
bitrate measurement

FIG 17

Table: Average bitrates produced by the WD3 arithemtic coder and the new proposal. Net bitrates do not include bits for byte alignment or fill bits

operating mode	WD3 (kbit/s)	new proposal (kbit/s)	difference after transcoding (kbit/s)	difference after transcoding (% of total bitrate)
Test 1, 64s	64.00	62.78	-1.22	-1.90
Test 2, 32s	32.00	31.47	-0.53	-1.66
Test 3, 24s	24.00	23.61	-0.39	-1.64
Test 4, 20s	20.00	19.65	-0.35	-1.74
Test 5, 16s	16.00	15.72	-0.28	-1.75
Test 6, 24m	24.00	23.56	-0.44	-1.83
Test 7, 20m	20.00	19.61	-0.39	-1.95
Test 8, 16m	16.00	15.70	-0.30	-1.87
Test 9, 12m	12.00	11.77	-0.23	-1.92

FIG 18

Table: minimum and maximum bitreservoir levels for WD3 arithemtic coder and proposal

operating mode	bitreservoir control					
	new proposal			WD3		
	min	max	avg	min	max	avg
Test 1, 64kbps stereo	3739	9557	8874	2314	9557	7018
Test 2, 32kbps stereo	2335	4505	4293	582	4505	3529
Test 3, 24kbps stereo	2184	4704	4472	957	4704	3871
Test 4, 20kbps stereo	2688	4864	4660	712	4864	3854
Test 5, 16kbps stereo	2965	5006	4859	724	5006	4234
Test 6, 24kbps mono	2185	4704	4457	1002	4704	3927
Test 7, 20kbps mono	2782	4864	4690	1192	4864	3935
Test 8, 16kbps mono	2916	5006	4905	1434	5006	4450
Test 9, 12kbps mono	3645	5184	5107	2256	5184	4787

FIG 19

Table: average complexity numbers for decoding the 32 kbit/s WD3 bitstream for the different version of the arithmetic coder.

	WD3	base version
PCU (MHz)	0.953	0.823

FIG 20

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

FIG 21(1)

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

};
```

FIG 21(2)

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

FIG 22(1)

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

FIG 22(2)

0x00511226UL,0x00512127UL,0x00512339UL,0x00521127UL,0x00521339UL,0x00522127UL,
0x00522339UL,0x00522637UL,
0x00523126UL,0x00523403UL,0x00524126UL,0x00524307UL,0x00531126UL,0x00531307UL,
0x00532126UL,0x00532307UL,
0x00532537UL,0x00532F44UL,0x00533239UL,0x00533442UL,0x0053373BUL,0x00534126UL,
0x00534542UL,0x00534F44UL,
0x00535442UL,0x0053FF3CUL,0x00542303UL,0x00542638UL,0x00543102UL,0x00543307UL,
0x00543638UL,0x00543F44UL,
0x00544307UL,0x00544537UL,0x00545102UL,0x00545303UL,0x0054FF41UL,0x00552437UL,
0x00552F44UL,0x00553437UL,
0x00553F44UL,0x00554303UL,0x0055463BUL,0x00554F44UL,0x00555307UL,0x00555537UL,
0x00556102UL,0x00556342UL,
0x00566203UL,0x00577342UL,0x0059733AUL,0x005CF53CUL,0x005FFF44UL,0x00611239UL,
0x00621127UL,0x00623307UL,
0x00631126UL,0x00632442UL,0x00632F44UL,0x00633303UL,0x00633638UL,0x00634102UL,
0x00634303UL,0x0063FF41UL,
0x00642303UL,0x00642F44UL,0x00643303UL,0x00643F44UL,0x00644207UL,0x00655203UL,
0x00665F44UL,0x00666303UL,
0x00677203UL,0x0069A53BUL,0x006DF841UL,0x006FFF44UL,0x00711239UL,0x00721127UL,
0x00722239UL,0x00733239UL,
0x00744437UL,0x00755203UL,0x00776F44UL,0x00777437UL,0x007BF33FUL,0x007FFF44UL,
0x00822239UL,0x00833203UL,
0x00854540UL,0x00888102UL,0x00888538UL,0x008BF23DUL,0x008FFF44UL,0x00922239UL,
0x0094333DUL,0x0096533DUL,
0x00998F44UL,0x00999303UL,0x009BF53BUL,0x009FFF44UL,0x00A44203UL,0x00A6633FUL,
0x00AA9F44UL,0x00AAA303UL,
0x00ADF33DUL,0x00AFFF44UL,0x00B33203UL,0x00B55440UL,0x00BBAC44UL,0x00BBB53BUL,
0x00BFFF44UL,0x00C33203UL,
0x00C6423DUL,0x00CCBF44UL,0x00CCC303UL,0x00CFFF44UL,0x00D4423DUL,0x00DDD303UL,
0x00DFFF44UL,0x00E6453CUL,
0x00EEE342UL,0x00EFFF44UL,0x00F55343UL,0x00F7FF44UL,0x00FFEF44UL,0x00FFF33DUL,
0x00FFF744UL,0x01000145UL,
0x01011147UL,0x01111248UL,0x0111224AUL,0x0111324DUL,0x0112114CUL,0x0112214EUL,
0x01123108UL,0x01131150UL,
0x01132150UL,0x01133150UL,0x01141126UL,0x01144100UL,0x01211151UL,0x01212153UL,
0x01213108UL,0x01221154UL,
0x01222155UL,0x01223117UL,0x01224212UL,0x01231215UL,0x01232215UL,0x01233215UL,
0x01241126UL,0x01242239UL,
0x0124322BUL,0x01251103UL,0x01255100UL,0x01311159UL,0x01312155UL,0x01313117UL,
0x01315201UL,0x0132122CUL,

FIG 22(3)

```
0x0132222CUL,0x0132321DUL,0x01331157UL,0x01332158UL,0x01333150UL,0x01341126UL,
0x01342127UL,0x01343100UL,
0x01344100UL,0x01351103UL,0x01355100UL,0x01369100UL,0x0141115AUL,0x01421227UL,
0x01422227UL,0x01431226UL,
0x01432226UL,0x01441127UL,0x01442127UL,0x01443100UL,0x01444100UL,0x01458100UL,
0x01511157UL,0x01531239UL,
0x01555100UL,0x01611157UL,0x01631239UL,0x01777100UL,0x01D33102UL,0x01FFF203UL,
0x0200055DUL,0x0200085DUL,
0x02000F5FUL,0x0211155DUL,0x02111F44UL,0x02112F5FUL,0x02121F44UL,0x02122F44UL,
0x02133F5FUL,0x0217FF5FUL,
0x021FFF44UL,0x02211F44UL,0x02212F44UL,0x02221F44UL,0x0222245EUL,0x02222F5FUL,
0x02223F5FUL,0x02232F5FUL,
0x02233F5FUL,0x02243F5FUL,0x022BFF5FUL,0x022FFF44UL,0x02322F44UL,0x02323F5FUL,
0x02332F5FUL,0x0233355CUL,
0x02333F5FUL,0x02334F5FUL,0x0233FF5CUL,0x02343F5FUL,0x0234FF5CUL,0x02353F5FUL,
0x02364F5FUL,0x0239655CUL,
0x023FFF44UL,0x02433F5FUL,0x0246445CUL,0x024A955DUL,0x024FFF44UL,0x0257435EUL,
0x025AC55CUL,0x025FFF44UL,
0x0267565DUL,0x026FFF44UL,0x0278655DUL,0x027FFF44UL,0x0288875DUL,0x028CC95DUL,
0x028FFF44UL,0x029A965DUL,
0x029FFF44UL,0x02ABA65DUL,0x02AFFF44UL,0x02BAFF5FUL,0x02BFFF44UL,0x02CCC45DUL,
0x02CFFF44UL,0x02DFFF44UL,
0x02EEE65DUL,0x02EFFF44UL,0x02F7335EUL,0x02FF0044UL,0x02FFEF44UL,0x02FFFF44UL,
0x0400045EUL,0x04000F5FUL,
0x04111F5DUL,0x04234F5DUL,0x042DFF5CUL,0x04343F5DUL,0x043FFF5FUL,0x044FFF5FUL,
0x045ED75CUL,0x045FFF5FUL,
0x046DFF5FUL,0x046FFF5FUL,0x047B5C5CUL,0x047FFF5FUL,0x048B435EUL,0x048FFF5FUL,
0x0499FF5CUL,0x04EFFF5FUL,
0x04FDFE5DUL,0x04FFFF5FUL,0x0600075EUL,0x06DFFF5FUL,0x06FFFF5FUL,0x08BFFF5CUL,
0x08FFFF5CUL,0x7FFFFFFF4FUL,

};
```

FIG 22(4)

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

2310

2312

FIG 23(1)

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

FIG 23(2)

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

FIG 23(3)

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

FIG 23(4)

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

FIG 23(5)

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

FIG 23(6)

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

FIG 23(7)

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

2396

FIG 23(8)

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

FIG 24