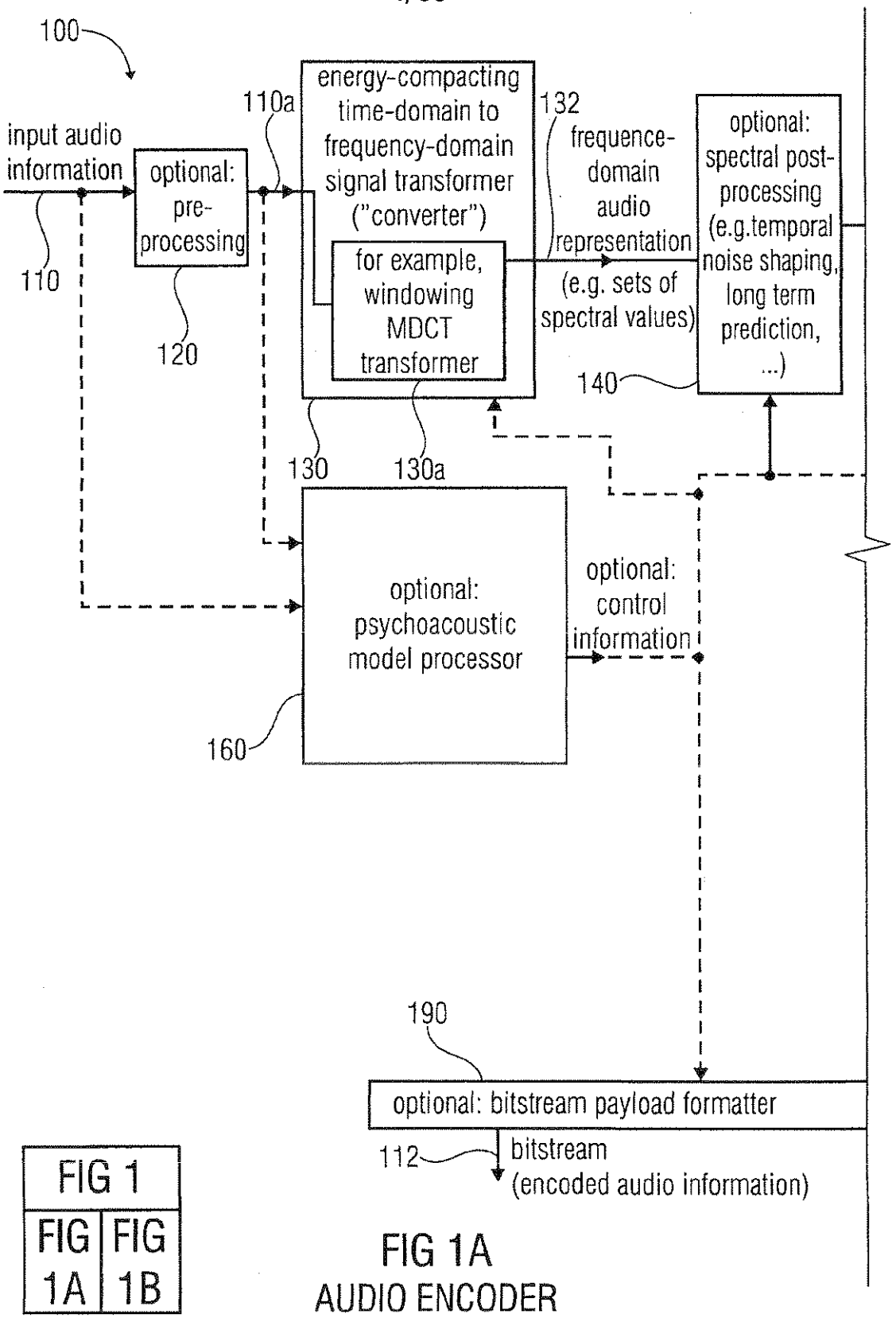


FIG 1A
AUDIO ENCODER

FIG 1	
FIG 1A	FIG 1B

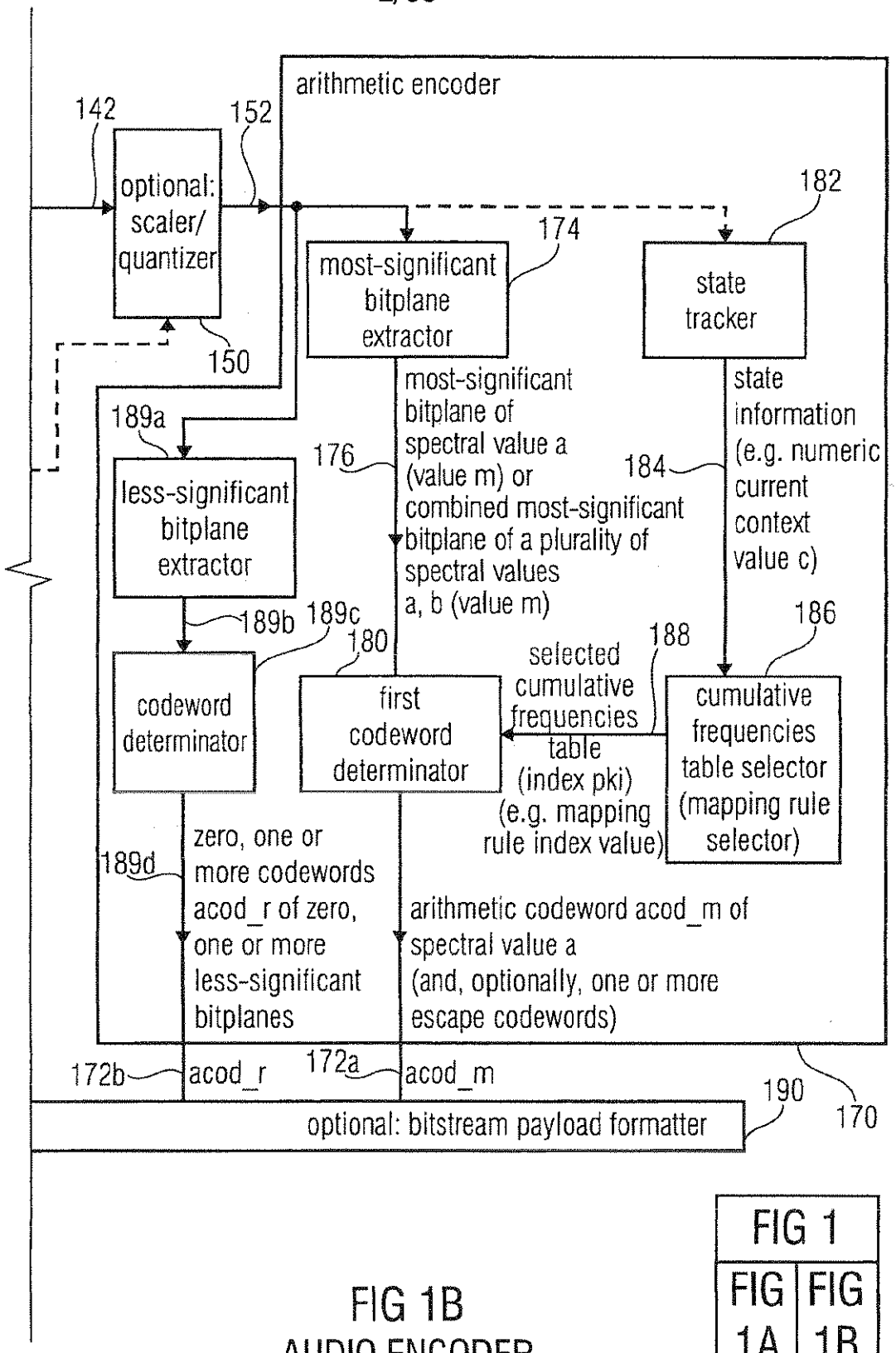


FIG 1B
AUDIO ENCODER

FIG 1	
FIG 1A	FIG 1B

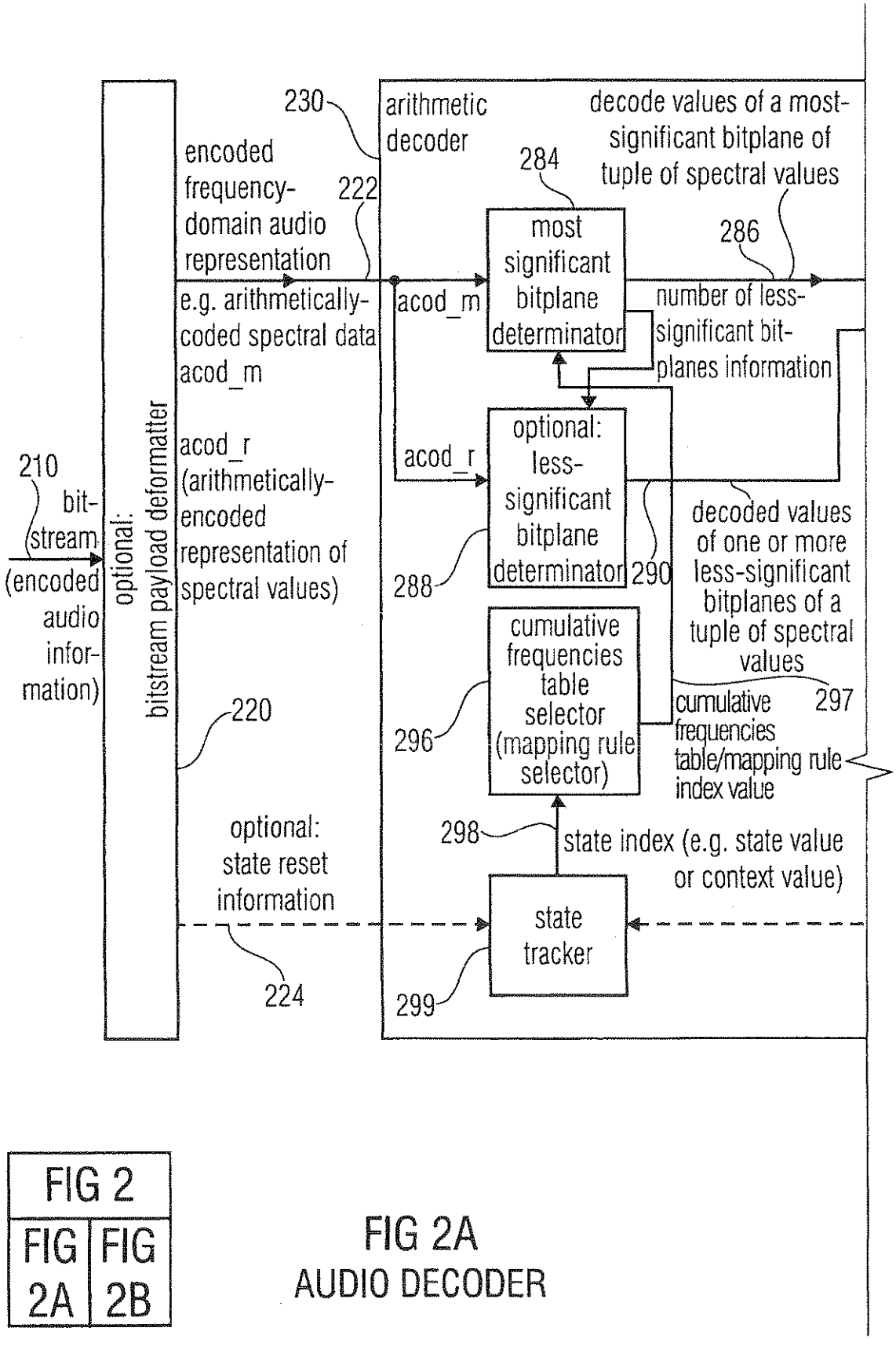


FIG 2	
FIG 2A	FIG 2B

FIG 2A
AUDIO DECODER

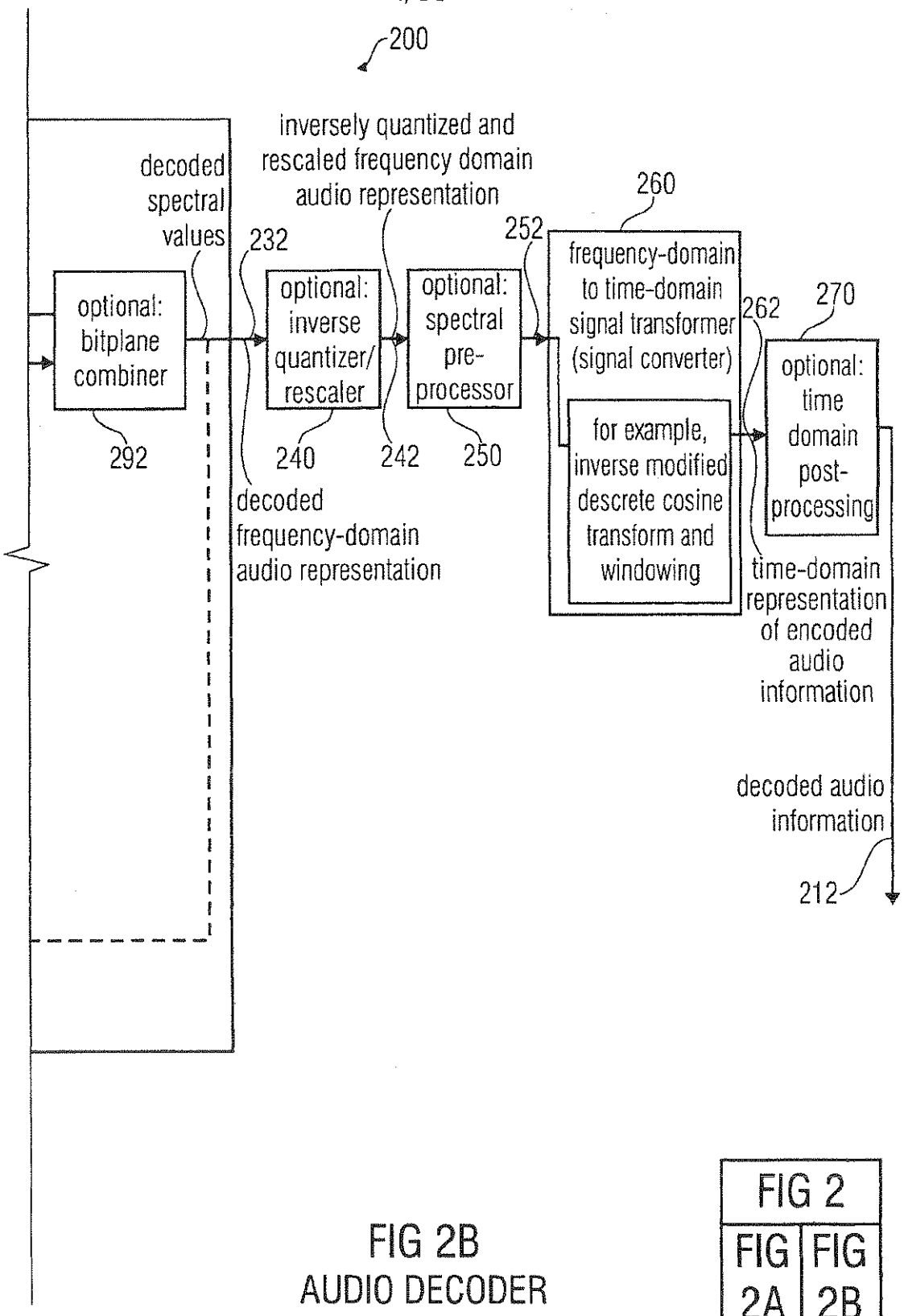


FIG 2B
AUDIO DECODER

FIG 2	
FIG 2A	FIG 2B

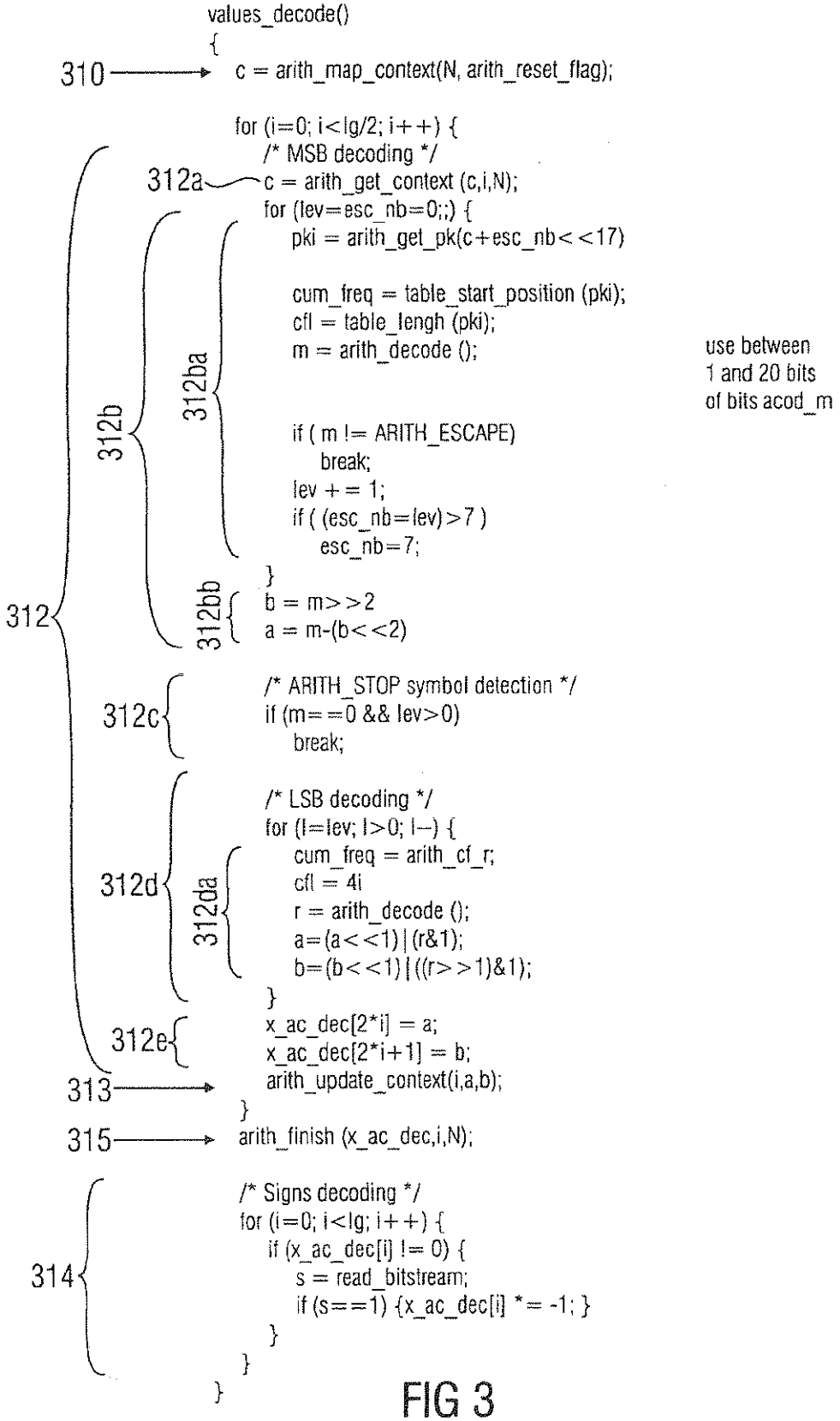

values_decode()
{
310
c = arith_map_context(N, arith_reset_flag);
for (i=0; i<lg/2; i++) {
/* MSB decoding */
312a
c = arith_get_context (c,i,N);
for (lev=esc_nb=0;;) {
pki = arith_get_pk(c+esc_nb<<17)
cum_freq = table_start_position (pki);
cfl = table_lengh (pki);
m = arith_decode ();
use between
1 and 20 bits
of bits acod_m
312b
312ba
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

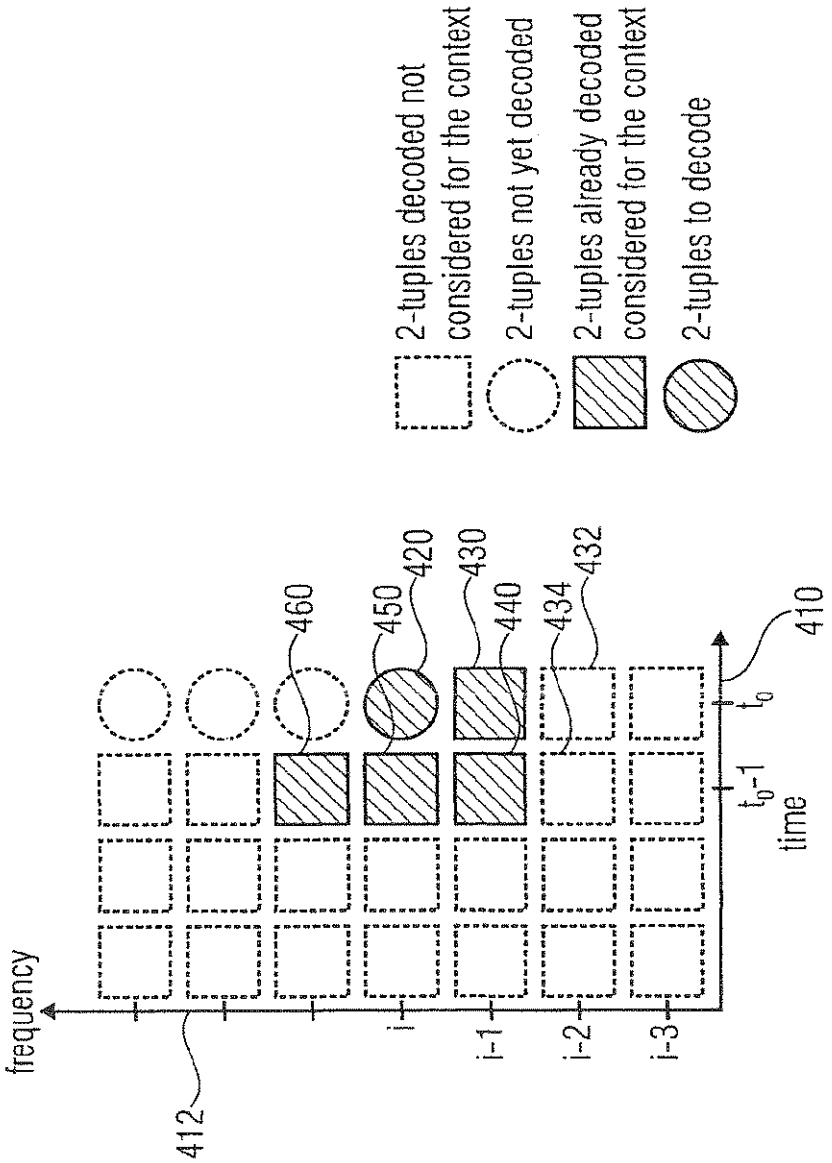


FIG 4
CONTEXT FOR THE STATE CALCULATION

```
/*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

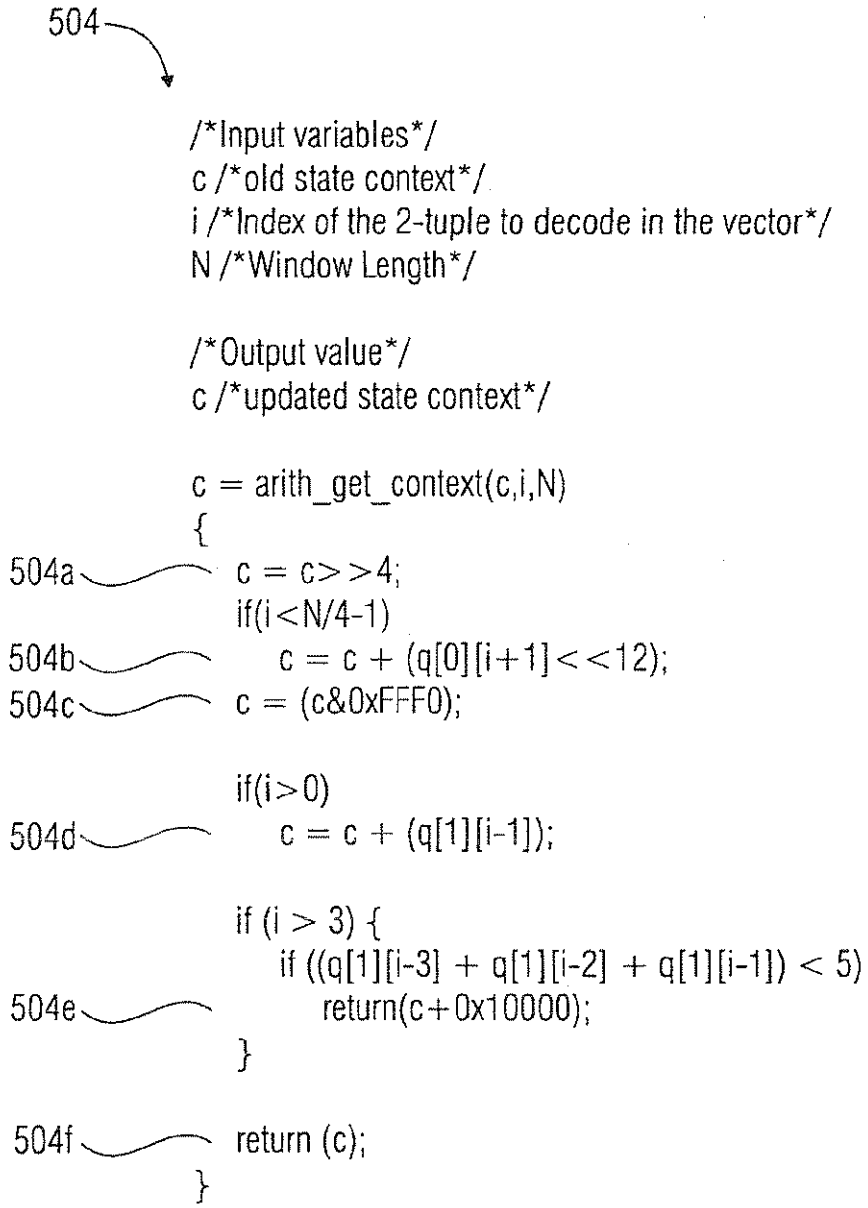

504
/*Input variables*/
c /*old state context*/
i /*Index of the 2-tuple to decode in the vector*/
N /*Window Length*/
/*Output value*/
c /*updated state context*/
c = arith_get_context(c,i,N)
{
504a
c = c>>4;
if(i<N/4-1)
504b
c = c + (q[0][i+1]<<12);
504c
c = (c&0xFFF0);
if(i>0)
504d
c = c + (q[1][i-1]);
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);
    }
506c ---> return ari_lookup_m[i_max];
}
```

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 = (value<<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 []	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_ac_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
  if (core_mode == 1) {
  lpd_channel_stream();
  }
  else {
  fd_channel_stream();
  }
}
	1	**uimsbf**

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==0) {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_pki(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

Table 15 - Syntax of UsacSingleChannelElement()

Syntax	No. of bits	Mnemonic

UsacSingleChannelElement(indepFlag)
{
  UsacCoreCoderData(1, indepFlag);
 	
  if (sbrRatioIndex > 0) {
  UsacSbrData(1, indepFlag);
  }
}
		optional

FIG 6K

Table 16 – Syntax of UsacChannelPairElement()

Syntax	No. of bits	Mnemonic

UsacChannelPairElement(indepFlag)
{
  if (stereoConfigIndex == 1) {
  nrCoreCoderChannels = 1;
  } else {
  nrCoreCoderChannels = 2;
  }
 	
  UsacCoreCoderData(nrCoreCoderChannels, indepFlag);
 	
  if (sbrRatioIndex > 0) {
  if (stereoConfigIndex == 0 || stereoConfigIndex == 3) {
  nrSbrChannels = 2;
  } else {
  nrSbrChannels = 1;
  }
  UsacSbrData(nrSbrChannels, indepFlag);
  }
 	
  if (stereoConfigIndex > 0) {
  Mps212Data(indepFlag);
  }
}
		optional
		
		optional

FIG 6L

Table 19 – Syntax of ics_info()

Syntax	No. of bits	Mnemonic
ics_info()		
{		
window_sequence;	2	uimsbf
window_shape;	1	uimsbf
if (window_sequence == EIGHT_SHORT_SEQUENCE) {		
max_sfb;	4	uimsbf
scale_factor_grouping;	7	uimsbf
}		
else {		
max_sfb;	6	uimsbf
}		
}		

FIG 6M

FIG 6N

Table 20 – Syntax of UsacCoreCoderData()

Syntax	No. of bits	Mnemonic
UsacCoreCoderData(nrChannels, indepFlag)		
{		
for (ch=0; ch < nrChannels; ch++) {		
core_mode[ch];	1	**uimsbf**
}		
if (nrChannels == 2) {	} optional	
StereoCoreToolInfo(core_mode);		
}		
for (ch=0; ch<nrChannels; ch++) {		
if (core_mode[ch] == 1) {		
lpd_channel_stream(indepFlag);		
}		
else {		
if ((nrChannels == 1) || (core_mode[0] != core_mode[1])) {	} optional	
tns_data_present[ch];	1	**uimsbf**
}		
fd_channel_stream(common_window, common_tw,		
tns_data_present[ch], noiseFilling, indepFlag);		
}		
}		
}		

FIG 60

Table 22 – Syntax of fd_channel_stream()

Syntax	No. of bits	Mnemonic
fd_channel_stream(common_window, common_tw, tns_data_present, noiseFilling, indepFlag)		
{		
global_gain;	8	uimsbf
if (noiseFilling) {		
noise_level;	3	uimsbf
noise_offset;	5	uimsbf
} optional		
else {		
noise_level = 0;		
}		
if (!common_window) {		
ics_info();		
}		
if (tw_mdct) { optional		
if (!common_tw) {		
tw_data();		
}		
}		
scale_factor_data ();		
if (tns_data_present) { optional		
tns_data ();		
}		
ac_spectral_data(indepFlag);		
fac_data_present;	1	uimsbf
if (fac_data_present) { optional		
fac_length = (window_sequence == EIGHT_SHORT_SEQUENCE) ? ccfl/16 : ccfl/8;		
fac_data(1, fac_length);		
}		
}		

Table 27 – Syntax of ac_spectral_data()

Syntax	No. of bits	Mnemonic
ac_spectral_data(indepFlag)		
{		
if (indepFlag) {		
arith_reset_flag = 1;		
} else {		
arith_reset_flag;	1	**uimsbf**
}		
for (win = 0; win < num_windows; win++) {		Note 1
arith_data(lg, arith_reset_flag && (win==0));		
}		
}		
Note 1: num_windows indicates the number of windows in the current window_sequence. In case window_sequence is EIGHT_SHORT_SEQUENCE num_windows equals 8. In all other cases num_windows equals 1		

optional

optional

FIG 6P

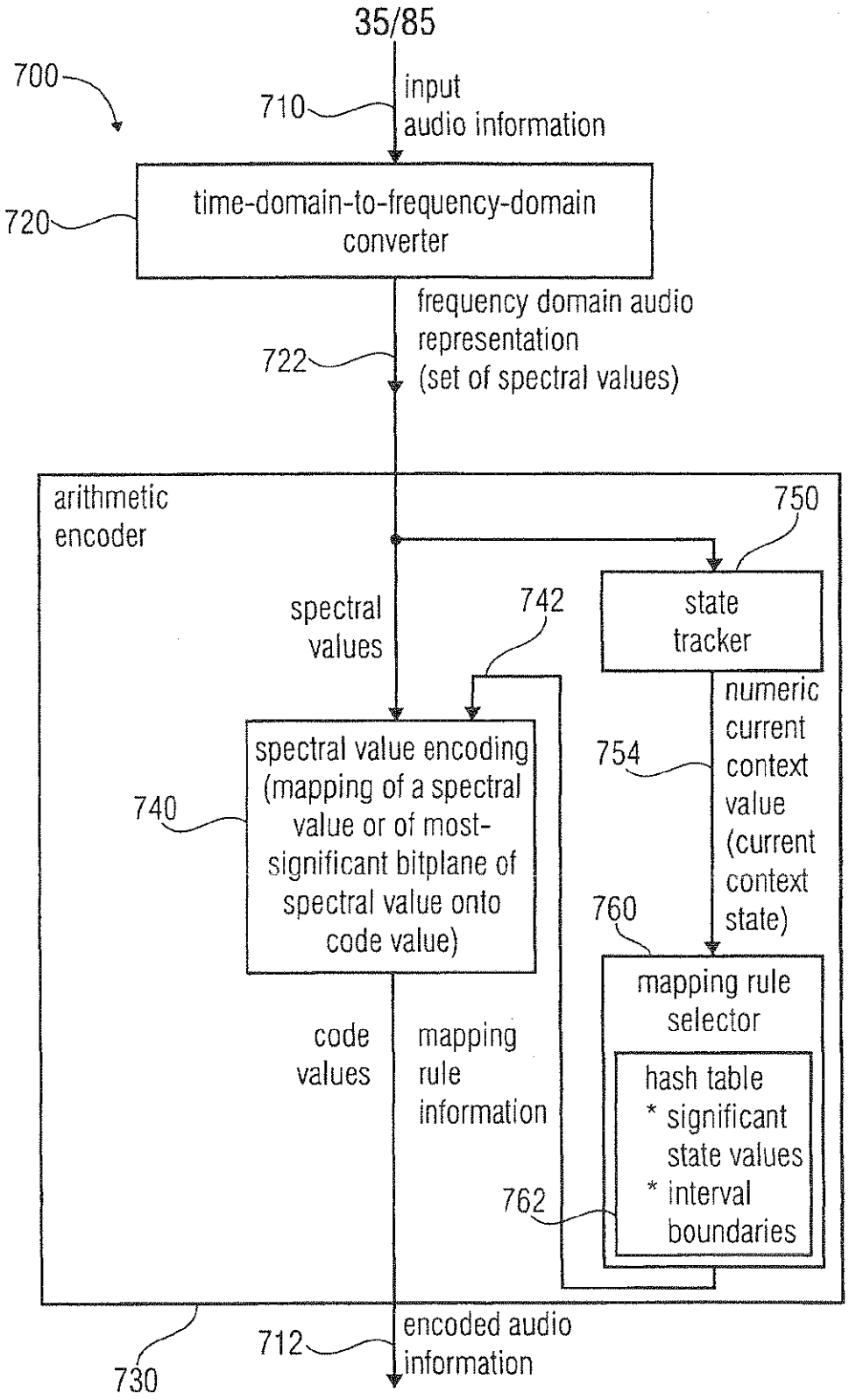

700
710
input
audio information
time-domain-to-frequency-domain
converter
720
frequency domain audio
representation
(set of spectral values)
722
arithmetic
encoder
750
state
tracker
spectral
values
742
numeric
current
context
value
(current
context
state)
754
spectral value encoding
(mapping of a spectral
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
730
712
encoded audio
information

FIG 7

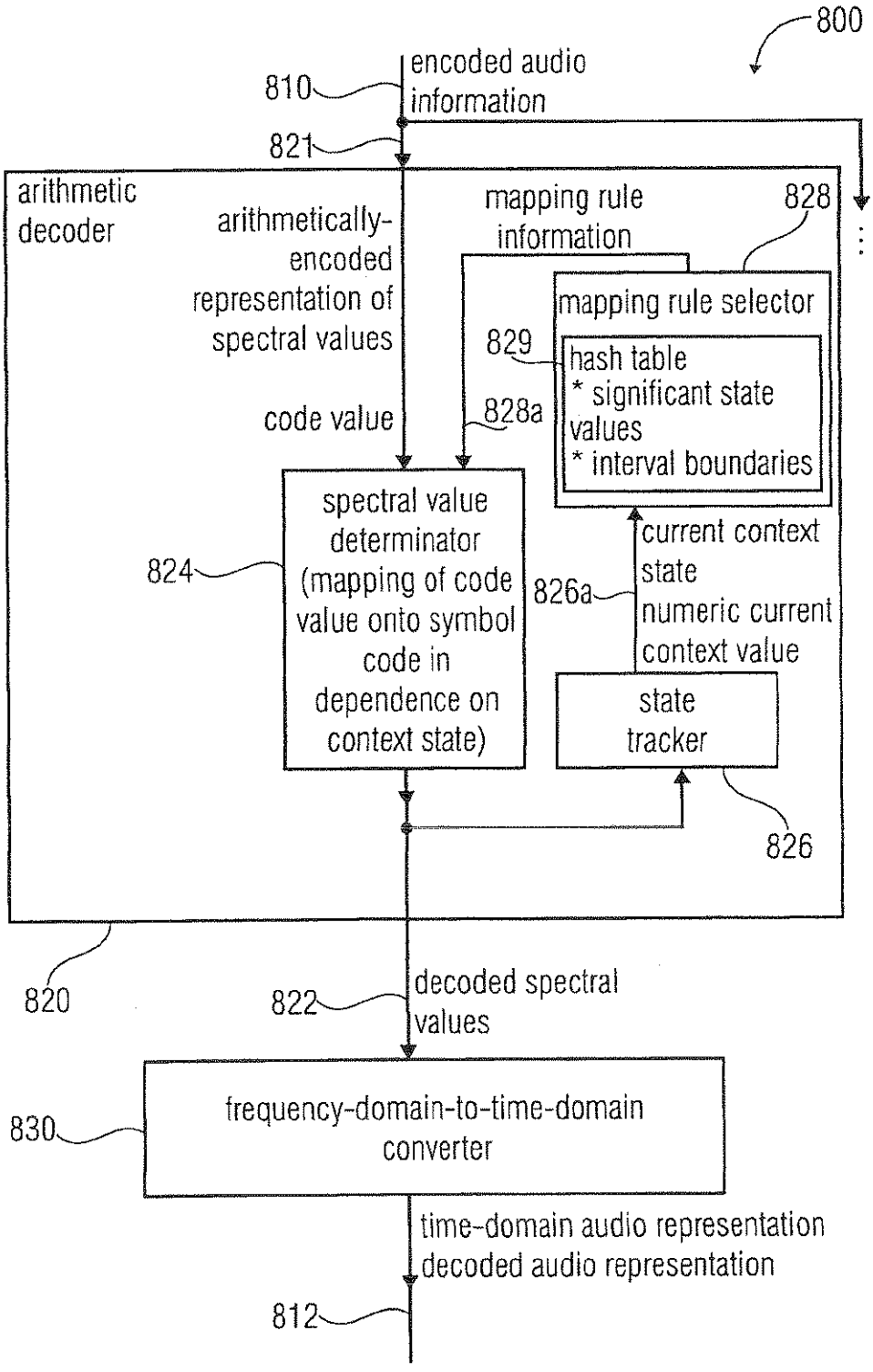
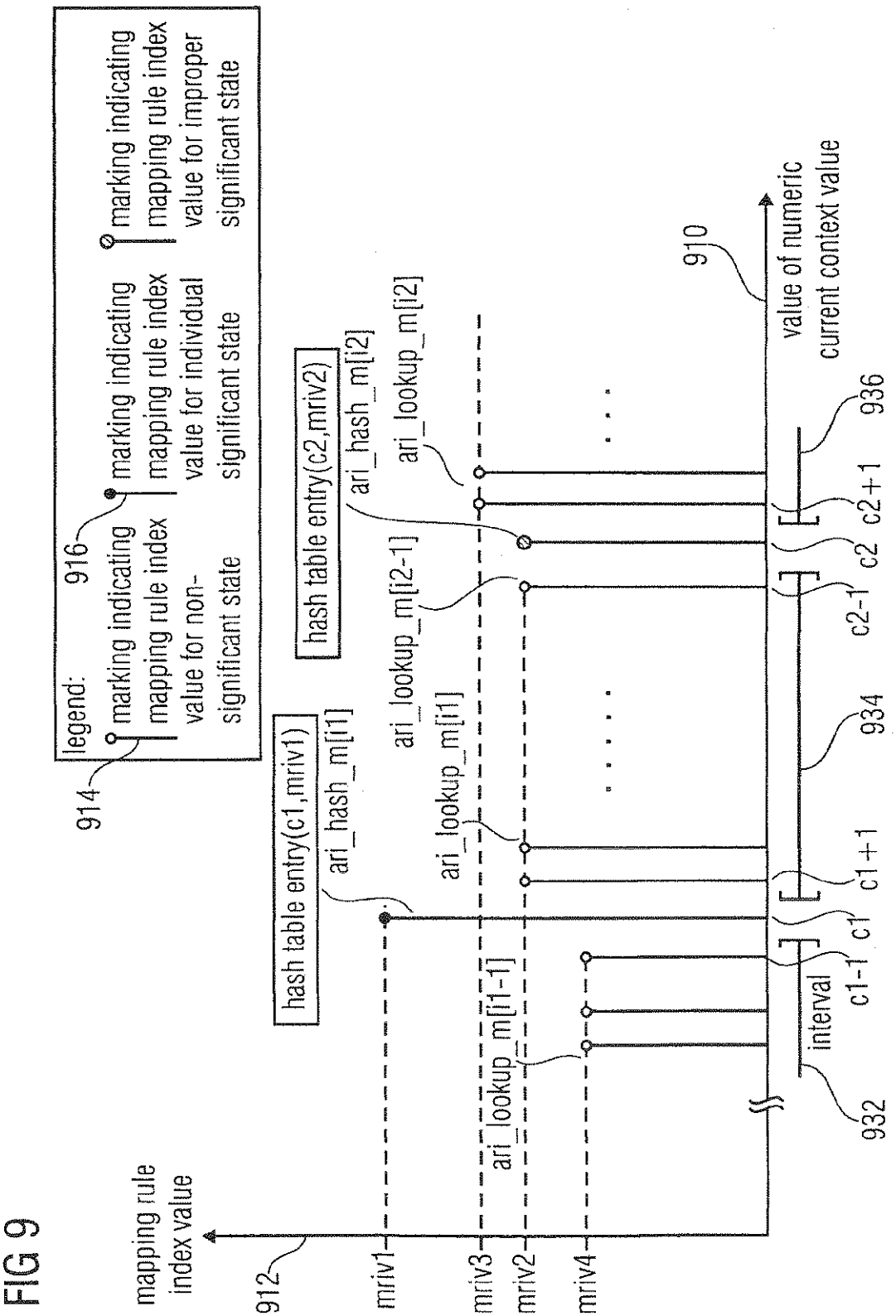

800
encoded audio information
810
821
arithmetic decoder
arithmetically-encoded representation of spectral values
code value
mapping rule information
828
mapping rule selector
829
hash table
* significant state values
* interval boundaries
828a
spectral value determinator (mapping of code value onto symbol code in dependence on context state)
824
current context state
numeric current context value
826a
state tracker
826
820
822
decoded spectral values
830
frequency-domain-to-time-domain converter
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)
ari_hash_m[i1]
hash table entry(c2,mriv2)
ari_hash_m[i2]
ari_lookup_m[i1]
ari_lookup_m[i2-1]
ari_lookup_m[i1-1]
ari_lookup_m[i2]
mapping rule index value
912
mriv1
mriv3
mriv2
mriv4
910
value of numeric current context value
interval
c1-1
c1
c1+1
c2-1
c2
c2+1
932
934
936

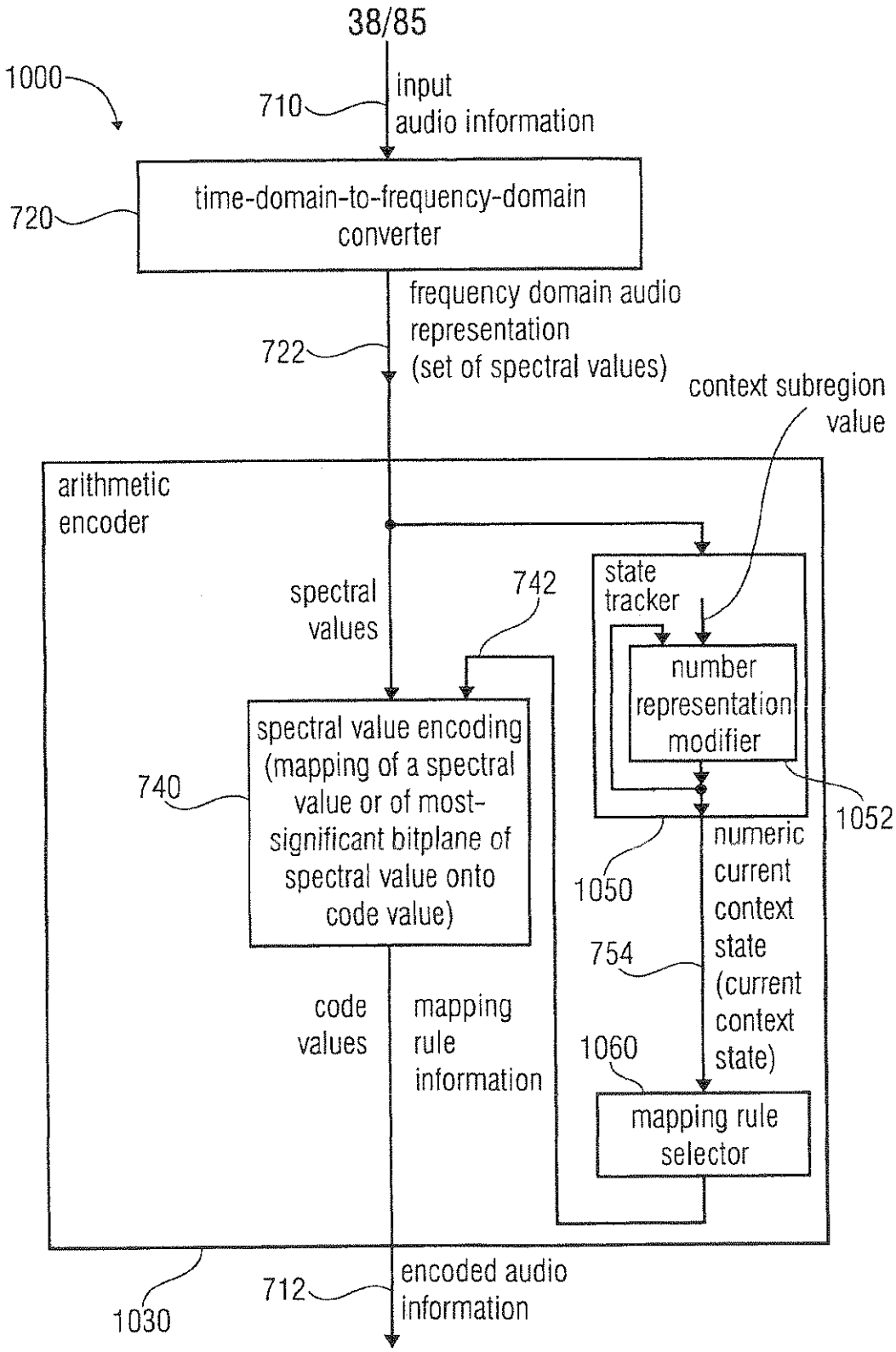

38/85
1000
710
input
audio information
720
time-domain-to-frequency-domain
converter
722
frequency domain audio
representation
(set of spectral values)
context subregion
value
arithmetic
encoder
spectral
values
742
state
tracker
number
representation
modifier
1052
740
spectral value encoding
(mapping of a spectral
value or of most-
significant bitplane of
spectral value onto
code value)
1050
numeric
current
context
state
(current
context
state)
754
code
values
mapping
rule
information
1060
mapping rule
selector
1030
712
encoded audio
information

FIG 10

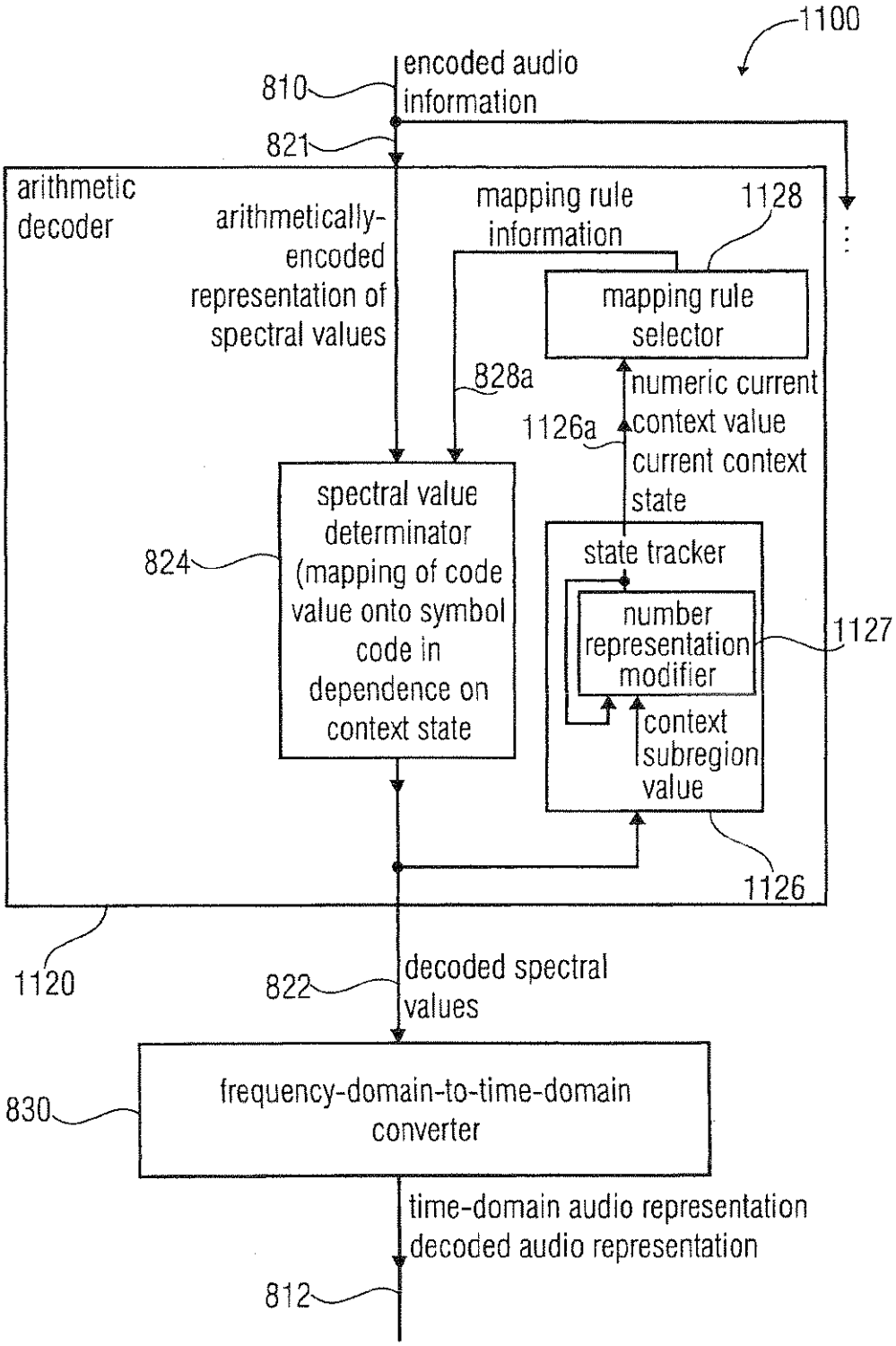

1100
encoded audio information
810
821
arithmetic decoder
arithmetically-encoded representation of spectral values
mapping rule information
1128
mapping rule selector
828a
numeric current context value
1126a
current context state
spectral value determinator (mapping of code value onto symbol code in 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

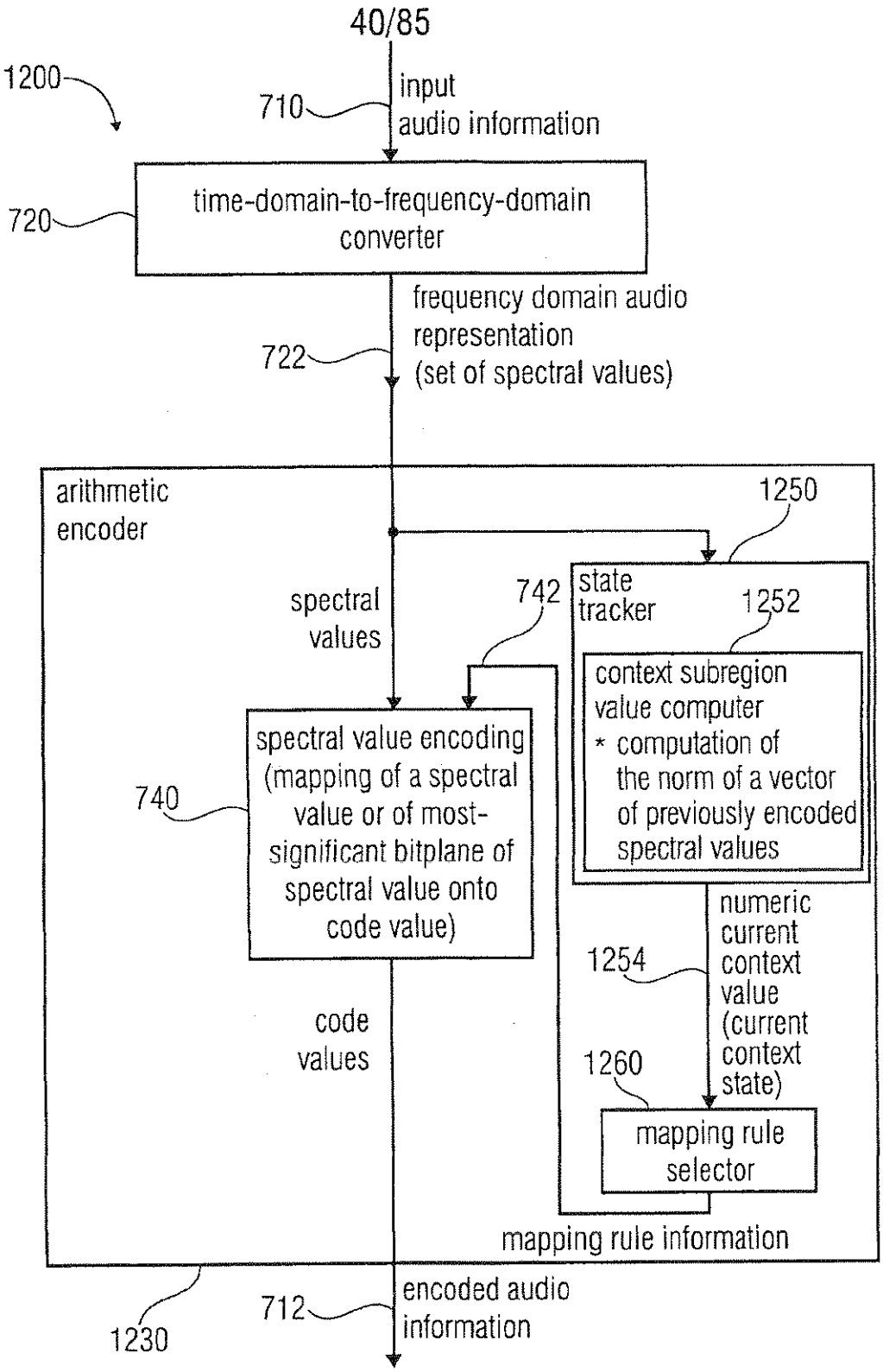

40/85
1200
710
input
audio information
720
time-domain-to-frequency-domain
converter
722
frequency domain audio
representation
(set of spectral values)
arithmetic
encoder
1250
spectral
values
742
state
tracker
1252
context subregion
value computer
* computation of
the norm of a vector
of previously encoded
spectral values
740
spectral value encoding
(mapping of a spectral
value or of most-
significant bitplane of
spectral value onto
code value)
1254
numeric
current
context
value
(current
context
state)
code
values
1260
mapping rule
selector
mapping rule information
1230
712
encoded audio
information

FIG 12

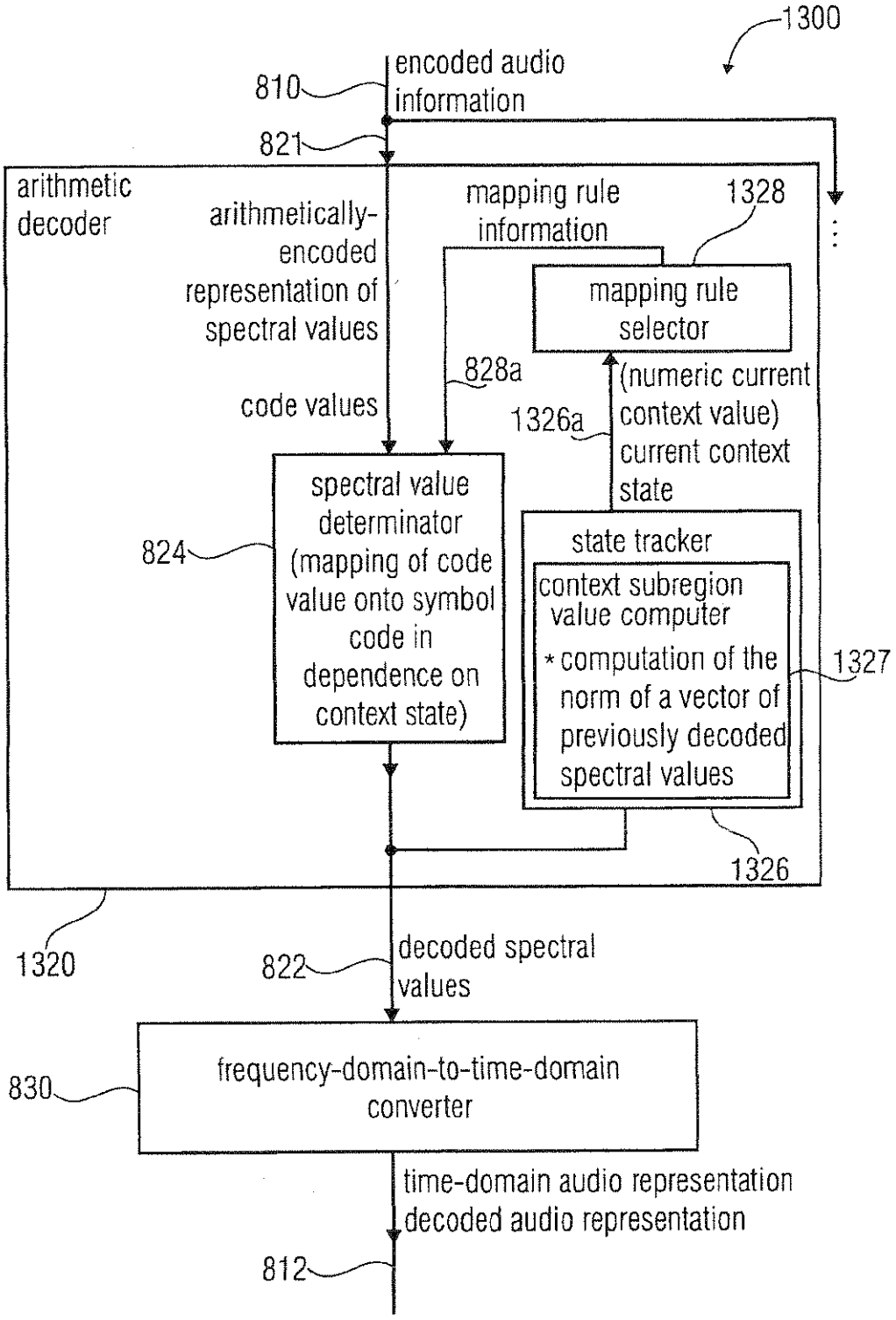

1300
encoded audio information
810
821
arithmetic decoder
arithmetically-encoded representation of spectral values
code values
mapping rule information
1328
mapping rule selector
828a
(numeric current context value)
current context state
1326a
spectral value determinator (mapping of code value onto symbol 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

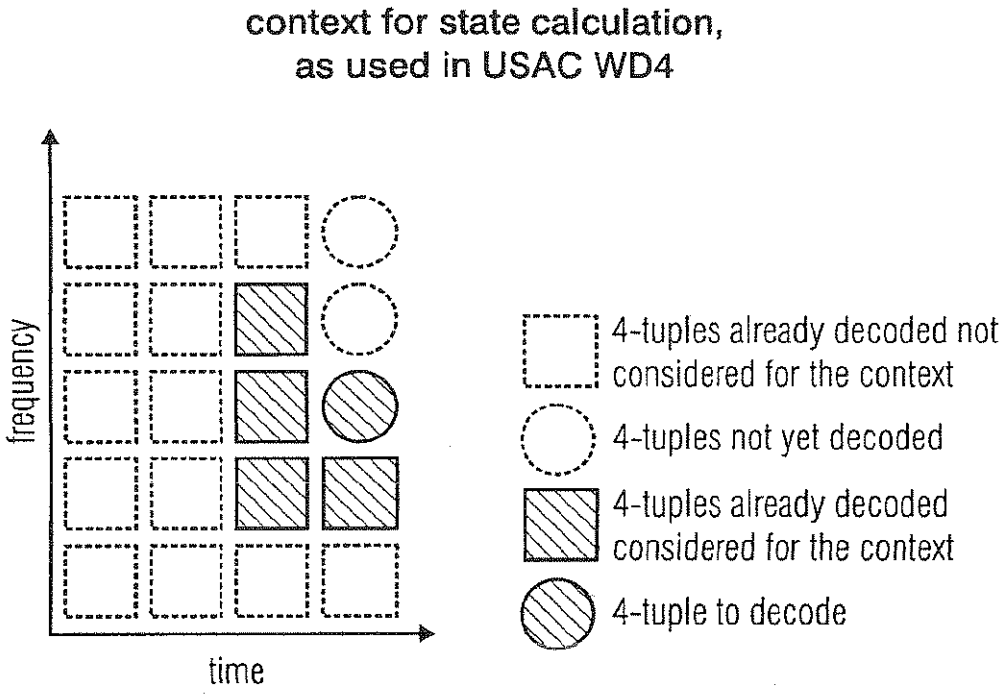

context for state calculation,
as used in USAC WD4
frequency
time
4-tuples already decoded not considered for the context
4-tuples not yet decoded
4-tuples already decoded considered for the context
4-tuple to decode

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 ,I	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

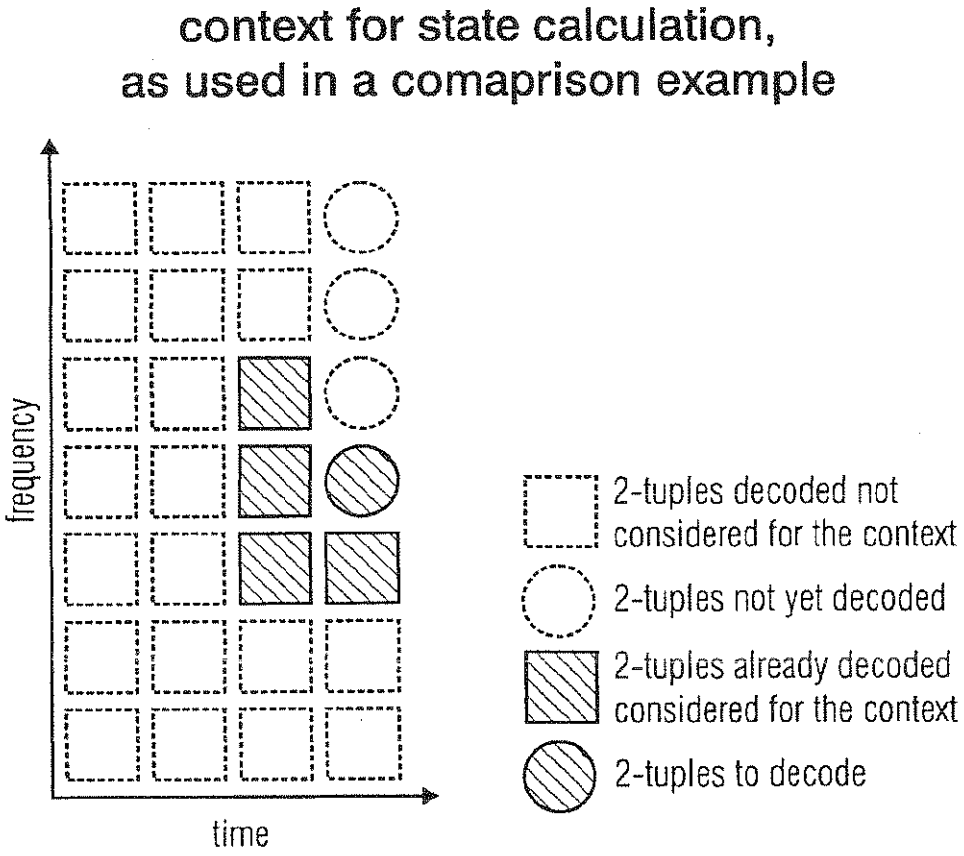

context for state calculation,
as used in a comaprison example
frequency
time
2-tuples decoded not considered for the context
2-tuples not yet decoded
2-tuples already decoded considered for the context
2-tuples to decode

FIG 15A

TABLES AS USED IN A COMAPRISON EXAMPLE

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

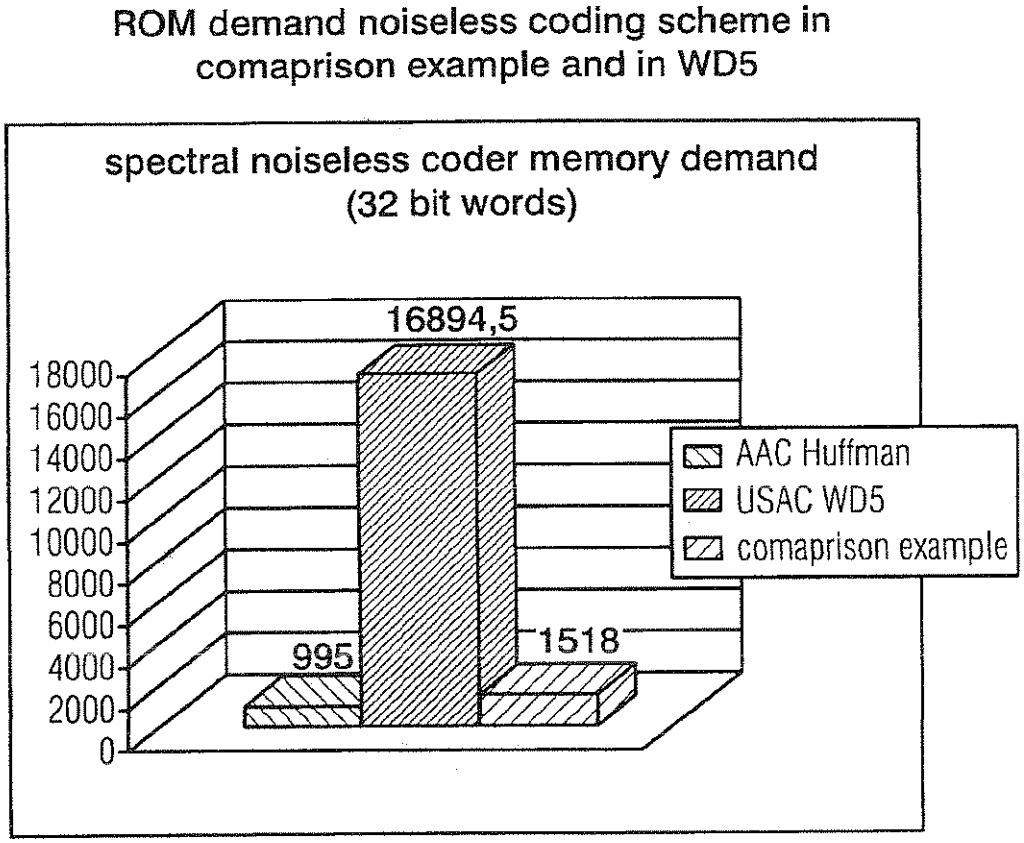

ROM demand noiseless coding scheme in comaprison example and in WD5
spectral noiseless coder memory demand (32 bit words)
16894,5
1518
995
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
AAC Huffman
USAC WD5
comaprison example

FIG 16A

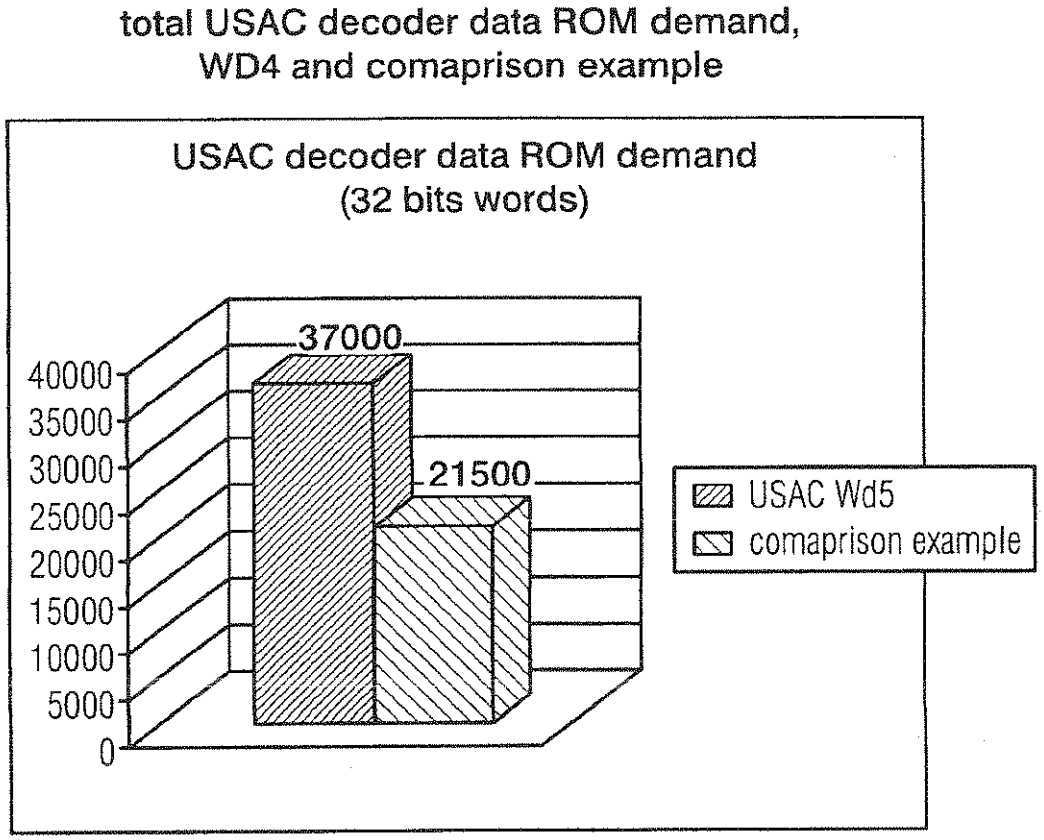

total USAC decoder data ROM demand,
WD4 and comaprison example
USAC decoder data ROM demand
(32 bits words)
37000
21500
40000
35000
30000
25000
20000
15000
10000
5000
0
USAC Wd5
comaprison example

FIG 16B

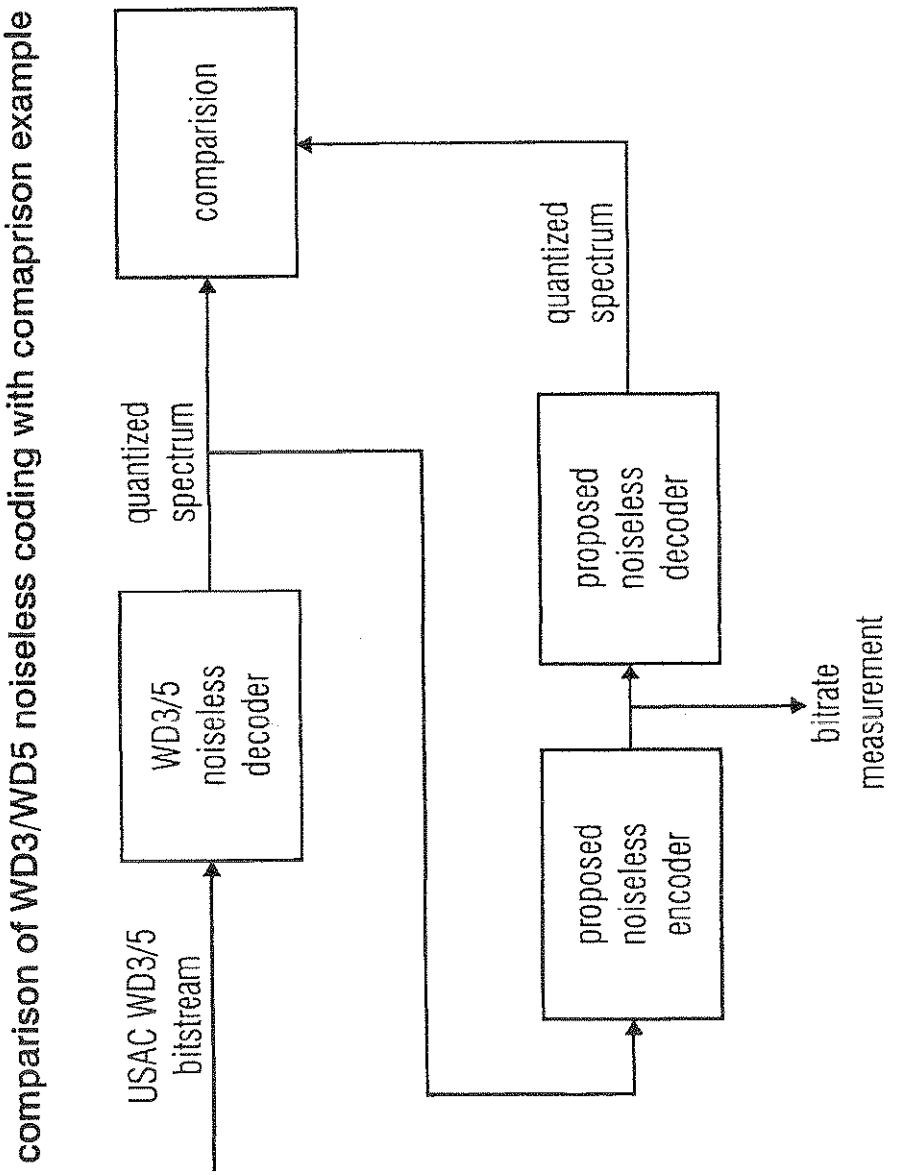

comparison of WD3/WD5 noiseless coding with comaprison example
USAC WD3/5 bitstream
WD3/5 noiseless decoder
quantized spectrum
comparision
quantized spectrum
proposed noiseless decoder
proposed noiseless encoder
bitrate measurement

FIG 17

table: average bitrates produced by the WD3 arithemtic coder and the comaprison example. Net bitrates do not include bits for byte alignment or fill bits

operating mode	WD3 (kbit/s)	comaprison example (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 comaprison example

operating mode	bitreservoir control					
	comaprison example			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

FIG 21 ari_lookup_m[742]

```
static unsigned short ari_lookup_m[742] = {
0x01,0x34,0x0D,0x13,0x12,0x25,0x00,0x3A,0x05,0x00,0x21,0x13,0x1F,0x1A,0x1D,0x36,
0x24,0x2B,0x1B,0x33,0x37,0x29,0x1D,0x33,0x37,0x33,0x37,0x33,0x37,0x33,0x2C,0x00,
0x21,0x13,0x25,0x2A,0x00,0x21,0x24,0x12,0x2C,0x1E,0x37,0x24,0x1F,0x35,0x37,0x24,
0x35,0x37,0x35,0x37,0x38,0x2D,0x21,0x29,0x1E,0x21,0x13,0x2D,0x36,0x38,0x29,0x36,
0x37,0x24,0x36,0x38,0x37,0x38,0x00,0x20,0x23,0x20,0x23,0x36,0x38,0x24,0x3B,0x24,
0x26,0x29,0x1F,0x30,0x2D,0x0D,0x12,0x3F,0x2D,0x21,0x1C,0x2A,0x00,0x21,0x12,0x1E,
0x36,0x38,0x36,0x37,0x3F,0x1E,0x0D,0x1F,0x2A,0x1E,0x21,0x24,0x12,0x2A,0x3C,0x21,
0x24,0x1F,0x3C,0x21,0x29,0x36,0x38,0x36,0x37,0x38,0x21,0x1E,0x00,0x3B,0x25,0x1E,
0x20,0x10,0x1F,0x3C,0x20,0x23,0x29,0x08,0x23,0x12,0x08,0x23,0x21,0x38,0x00,0x20,
0x13,0x20,0x3B,0x1C,0x20,0x3B,0x29,0x20,0x23,0x24,0x21,0x24,0x21,0x24,0x3B,0x13,
0x23,0x26,0x23,0x13,0x21,0x24,0x26,0x29,0x12,0x22,0x2B,0x02,0x1E,0x0D,0x1F,0x2D,
0x00,0x0D,0x12,0x00,0x3C,0x21,0x29,0x3C,0x21,0x2A,0x3C,0x3B,0x22,0x1E,0x20,0x10,
0x1F,0x3C,0x0D,0x29,0x3C,0x21,0x24,0x08,0x23,0x20,0x38,0x39,0x3C,0x20,0x13,0x3C,
0x00,0x0D,0x13,0x1F,0x3C,0x09,0x26,0x1F,0x08,0x09,0x26,0x12,0x08,0x23,0x29,0x20,
0x23,0x21,0x24,0x20,0x13,0x20,0x3B,0x16,0x20,0x3B,0x29,0x20,0x3B,0x29,0x20,0x3B,
0x13,0x21,0x24,0x29,0x0B,0x13,0x09,0x3B,0x13,0x09,0x3B,0x13,0x21,0x3B,0x13,0x0D,
0x26,0x29,0x26,0x29,0x3D,0x12,0x22,0x28,0x2E,0x04,0x08,0x13,0x3C,0x3B,0x3C,0x20,
0x10,0x3C,0x21,0x07,0x08,0x10,0x00,0x08,0x0D,0x29,0x08,0x0D,0x29,0x08,0x09,0x13,
0x20,0x23,0x39,0x08,0x09,0x13,0x08,0x09,0x16,0x08,0x09,0x10,0x12,0x20,0x3B,0x3D,
0x09,0x26,0x20,0x3B,0x24,0x39,0x09,0x26,0x20,0x0D,0x13,0x00,0x09,0x13,0x20,0x0D,
0x26,0x12,0x20,0x3B,0x13,0x21,0x26,0x0B,0x12,0x09,0x3B,0x16,0x09,0x3B,0x3D,0x09,
0x26,0x0D,0x13,0x26,0x3D,0x1C,0x12,0x1F,0x28,0x2E,0x07,0x0B,0x08,0x09,0x00,0x39,
0x0B,0x08,0x26,0x08,0x09,0x13,0x20,0x0B,0x39,0x10,0x39,0x0D,0x13,0x20,0x10,0x12,
0x09,0x13,0x20,0x3B,0x13,0x09,0x26,0x0B,0x09,0x3B,0x1C,0x09,0x3B,0x13,0x20,0x3B,
0x13,0x09,0x26,0x0B,0x16,0x0D,0x13,0x09,0x13,0x09,0x13,0x26,0x3D,0x1C,0x1F,0x28,
0x2E,0x07,0x10,0x39,0x0B,0x39,0x39,0x13,0x39,0x0B,0x39,0x0B,0x39,0x26,0x39,0x10,
0x20,0x3B,0x16,0x20,0x10,0x09,0x26,0x0B,0x13,0x09,0x13,0x26,0x1C,0x0B,0x3D,0x1C,
0x1F,0x28,0x2B,0x07,0x0C,0x39,0x0B,0x39,0x0B,0x0C,0x0B,0x26,0x0B,0x26,0x3D,0x0D,
0x1C,0x14,0x28,0x2B,0x39,0x0B,0x0C,0x0E,0x3D,0x1C,0x0D,0x12,0x22,0x2B,0x07,0x0C,
0x0E,0x3D,0x1C,0x10,0x1F,0x2B,0x0C,0x0E,0x19,0x14,0x10,0x1F,0x28,0x0C,0x0E,0x19,
0x14,0x26,0x22,0x2B,0x0C,0x0E,0x19,0x14,0x26,0x28,0x0E,0x19,0x14,0x26,0x28,0x0E,
0x19,0x14,0x28,0x0E,0x19,0x14,0x22,0x28,0x2B,0x0E,0x14,0x2B,0x31,0x00,0x3A,0x3A,
0x05,0x05,0x1B,0x1D,0x33,0x06,0x35,0x35,0x20,0x21,0x37,0x21,0x24,0x05,0x1B,0x2C,
0x2C,0x2C,0x06,0x34,0x1E,0x34,0x00,0x08,0x36,0x09,0x21,0x26,0x1C,0x2C,0x00,0x02,
0x02,0x02,0x3F,0x04,0x04,0x04,0x34,0x39,0x20,0x0A,0x0C,0x39,0x0B,0x0F,0x07,0x07,
0x07,0x07,0x34,0x39,0x39,0x0A,0x0C,0x39,0x0C,0x0F,0x07,0x07,0x07,0x00,0x39,0x39,
0x0C,0x0F,0x07,0x07,0x39,0x0C,0x0F,0x07,0x39,0x0C,0x0F,0x39,0x39,0x0C,0x0F,0x39,
0x0C,0x39,0x0C,0x0F,0x00,0x11,0x27,0x17,0x2F,0x27,0x00,0x27,0x17,0x00,0x11,0x17,
0x00,0x11,0x17,0x11,0x00,0x27,0x15,0x11,0x17,0x01,0x15,0x11,0x15,0x11,0x15,0x15,
0x17,0x00,0x27,0x01,0x27,0x27,0x15,0x00,0x27,0x11,0x27,0x15,0x15,0x15,0x27,0x15,
0x15,0x15,0x15,0x17,0x2F,0x11,0x17,0x27,0x27,0x27,0x11,0x27,0x15,0x27,0x27,0x15,
0x15,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,
0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,
0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x17,0x2F,0x2B,0x00,0x27,0x00,0x00,0x11,
0x15,0x00,0x11,0x11,0x27,0x27,0x15,0x17,0x15,0x17,0x15,0x17,0x27,0x17,0x27,0x17,
0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,
0x27,0x15,0x27,0x27,0x15,0x27};
```

FIG 22(1)

ari_hash_m[742]

```
static unsigned long ari_hash_m[742] = {
0x00000104UL, 0x0000030AUL, 0x00000510UL, 0x00000716UL,
0x00000A1FUL, 0x00000F2EUL, 0x00011100UL, 0x00111103UL,
0x00111306UL, 0x00111436UL, 0x00111623UL, 0x00111929UL,
0x00111F2EUL, 0x0011221BUL, 0x00112435UL, 0x00112621UL,
0x00112D12UL, 0x00113130UL, 0x0011331DUL, 0x00113535UL,
0x00113938UL, 0x0011411BUL, 0x00114433UL, 0x00114635UL,
0x00114F29UL, 0x00116635UL, 0x00116F24UL, 0x00117433UL,
0x0011FF0FUL, 0x00121102UL, 0x0012132DUL, 0x00121436UL,
0x00121623UL, 0x00121912UL, 0x0012213FUL, 0x0012232DUL,
0x00122436UL, 0x00122638UL, 0x00122A29UL, 0x00122F2BUL,
0x0012322DUL, 0x00123436UL, 0x00123738UL, 0x00123B29UL,
0x0012411DUL, 0x00124536UL, 0x00124938UL, 0x00124F12UL,
0x00125535UL, 0x00125F29UL, 0x00126535UL, 0x0012B837UL,
0x0013112AUL, 0x0013131EUL, 0x0013163BUL, 0x0013212DUL,
0x0013233CUL, 0x00132623UL, 0x00132F2EUL, 0x0013321EUL,
0x00133521UL, 0x00133824UL, 0x0013411EUL, 0x00134336UL,
0x00134838UL, 0x00135135UL, 0x00135537UL, 0x00135F12UL,
0x00137637UL, 0x0013FF29UL, 0x00140024UL, 0x00142321UL,
0x00143136UL, 0x00143321UL, 0x00143F25UL, 0x00144321UL,
0x00148638UL, 0x0014FF29UL, 0x00154323UL, 0x0015FF12UL,
0x0016F20CUL, 0x0018A529UL, 0x00210031UL, 0x0021122CUL,
0x00211408UL, 0x00211713UL, 0x00211F2EUL, 0x0021222AUL,
0x00212408UL, 0x00212710UL, 0x00212F2EUL, 0x0021331EUL,
0x00213436UL, 0x00213824UL, 0x0021412DUL, 0x0021431EUL,
0x00214536UL, 0x00214F1FUL, 0x00216637UL, 0x00220004UL,
0x0022122AUL, 0x00221420UL, 0x00221829UL, 0x00221F2EUL,
0x0022222DUL, 0x00222408UL, 0x00222623UL, 0x00222929UL,
0x00222F2BUL, 0x0022321EUL, 0x00223408UL, 0x00223724UL,
0x00223A29UL, 0x0022411EUL, 0x00224436UL, 0x00224823UL,
0x00225134UL, 0x00225621UL, 0x00225F12UL, 0x00226336UL,
0x00227637UL, 0x0022FF29UL, 0x0023112DUL, 0x0023133CUL,
0x00231420UL, 0x00231916UL, 0x0023212DUL, 0x0023233CUL,
0x00232509UL, 0x00232929UL, 0x0023312DUL, 0x00233308UL,
0x00233509UL, 0x00233724UL, 0x0023413CUL, 0x00234421UL,
0x00234A13UL, 0x0023513CUL, 0x00235421UL, 0x00235F1FUL,
0x00236421UL, 0x0023FF29UL, 0x00240024UL, 0x0024153BUL,
0x00242108UL, 0x00242409UL, 0x00242726UL, 0x00243108UL,
0x00243409UL, 0x00243610UL, 0x00244136UL, 0x00244321UL,
0x00244523UL, 0x00244F1FUL, 0x00245423UL, 0x0024610AUL,
0x00246423UL, 0x0024FF29UL, 0x00252510UL, 0x00253121UL,
0x0025343BUL, 0x00254121UL, 0x00254510UL, 0x00254F25UL,
0x00255221UL, 0x0025FF12UL, 0x00266513UL, 0x0027F529UL,
0x0029F101UL, 0x002CF224UL, 0x00310030UL, 0x0031122AUL,
0x00311420UL, 0x00311816UL, 0x0031212CUL, 0x0031231EUL,
0x00312408UL, 0x00312710UL, 0x0031312AUL, 0x0031321EUL,
```

FIG 22(2)

0x00313408UL, 0x00313623UL, 0x0031411EUL, 0x0031433CUL,
0x00320007UL, 0x0032122DUL, 0x00321420UL, 0x00321816UL,
0x0032212DUL, 0x0032233CUL, 0x00322509UL, 0x00322916UL,
0x0032312DUL, 0x00323420UL, 0x00323710UL, 0x00323F2BUL,
0x00324308UL, 0x00324623UL, 0x00324F25UL, 0x00325421UL,
0x00325F1FUL, 0x00326421UL, 0x0032FF29UL, 0x00331107UL,
0x00331308UL, 0x0033150DUL, 0x0033211EUL, 0x00332308UL,
0x00332420UL, 0x00332610UL, 0x00332929UL, 0x0033311EUL,
0x00333308UL, 0x0033363BUL, 0x00333A29UL, 0x0033413CUL,
0x00334320UL, 0x0033463BUL, 0x00334A29UL, 0x0033510AUL,
0x00335320UL, 0x00335824UL, 0x0033610AUL, 0x00336321UL,
0x00336F12UL, 0x00337623UL, 0x00341139UL, 0x0034153BUL,
0x00342108UL, 0x00342409UL, 0x00342610UL, 0x00343108UL,
0x00343409UL, 0x00343610UL, 0x00344108UL, 0x0034440DUL,
0x00344610UL, 0x0034510AUL, 0x00345309UL, 0x0034553BUL,
0x0034610AUL, 0x00346309UL, 0x0034F824UL, 0x00350029UL,
0x00352510UL, 0x00353120UL, 0x0035330DUL, 0x00353510UL,
0x00354120UL, 0x0035430DUL, 0x00354510UL, 0x00354F28UL,
0x0035530DUL, 0x00355510UL, 0x00355F1FUL, 0x00356410UL,
0x00359626UL, 0x0035FF12UL, 0x00366426UL, 0x0036FF12UL,
0x0037F426UL, 0x0039D712UL, 0x003BF612UL, 0x003DF81FUL,
0x00410004UL, 0x00411207UL, 0x0041150DUL, 0x0041212AUL,
0x00412420UL, 0x0041311EUL, 0x00413308UL, 0x00413509UL,
0x00413F2BUL, 0x00414208UL, 0x00420007UL, 0x0042123CUL,
0x00421409UL, 0x00422107UL, 0x0042223CUL, 0x00422409UL,
0x00422610UL, 0x0042313CUL, 0x00423409UL, 0x0042363BUL,
0x0042413CUL, 0x00424320UL, 0x0042463BUL, 0x00425108UL,
0x00425409UL, 0x0042FF29UL, 0x00431107UL, 0x00431320UL,
0x0043153BUL, 0x0043213CUL, 0x00432320UL, 0x00432610UL,
0x0043313CUL, 0x00433320UL, 0x0043353BUL, 0x00433813UL,
0x00434108UL, 0x00434409UL, 0x00434610UL, 0x00435108UL,
0x0043553BUL, 0x00435F25UL, 0x00436309UL, 0x0043753BUL,
0x0043FF29UL, 0x00441239UL, 0x0044143BUL, 0x00442139UL,
0x00442309UL, 0x0044253BUL, 0x00443108UL, 0x00443220UL,
0x0044353BUL, 0x0044410AUL, 0x00444309UL, 0x0044453BUL,
0x00444813UL, 0x0044510AUL, 0x00445309UL, 0x00445510UL,
0x00445F25UL, 0x0044630DUL, 0x00450026UL, 0x00452713UL,
0x00453120UL, 0x0045330DUL, 0x00453510UL, 0x00454120UL,
0x0045430DUL, 0x00454510UL, 0x00455120UL, 0x0045530DUL,
0x00456209UL, 0x00456410UL, 0x0045FF12UL, 0x00466513UL,
0x0047FF22UL, 0x0048FF25UL, 0x0049F43DUL, 0x004BFB25UL,
0x004EF825UL, 0x004FFF18UL, 0x00511339UL, 0x00512107UL,
0x00513409UL, 0x00520007UL, 0x00521107UL, 0x00521320UL,
0x00522107UL, 0x00522409UL, 0x0052313CUL, 0x00523320UL,
0x0052353BUL, 0x00524108UL, 0x00524320UL, 0x00531139UL,
0x00531309UL, 0x00532139UL, 0x00532309UL, 0x0053253BUL,
0x00533108UL, 0x0053340DUL, 0x00533713UL, 0x00534108UL,
0x0053453BUL, 0x00534F2BUL, 0x00535309UL, 0x00535610UL,
0x00535F25UL, 0x0053643BUL, 0x00541139UL, 0x00542139UL,

FIG 22(3)

0x00542309UL, 0x00542613UL, 0x00543139UL, 0x00543309UL,
0x00543510UL, 0x00543F2BUL, 0x00544309UL, 0x00544510UL,
0x00544F28UL, 0x0054530DUL, 0x0054FF12UL, 0x00553613UL,
0x00553F2BUL, 0x00554410UL, 0x0055510AUL, 0x0055543BUL,
0x00555F25UL, 0x0055633BUL, 0x0055FF12UL, 0x00566513UL,
0x00577413UL, 0x0059FF28UL, 0x005CC33DUL, 0x005EFB28UL,
0x005FFF18UL, 0x00611339UL, 0x00612107UL, 0x00613320UL,
0x0061A724UL, 0x00621107UL, 0x0062140BUL, 0x00622107UL,
0x00622320UL, 0x00623139UL, 0x00623320UL, 0x00631139UL,
0x0063130CUL, 0x00632139UL, 0x00632309UL, 0x00633139UL,
0x00633309UL, 0x00633626UL, 0x00633F2BUL, 0x00634309UL,
0x00634F2BUL, 0x0063543BUL, 0x0063FF12UL, 0x0064343BUL,
0x00643F2BUL, 0x0064443BUL, 0x00645209UL, 0x00665513UL,
0x0066610AUL, 0x00666526UL, 0x0067A616UL, 0x0069843DUL,
0x006CF612UL, 0x006EF326UL, 0x006FFF18UL, 0x0071130CUL,
0x00721107UL, 0x00722239UL, 0x0072291CUL, 0x0072340BUL,
0x00731139UL, 0x00732239UL, 0x0073630BUL, 0x0073FF12UL,
0x0074430BUL, 0x00755426UL, 0x00776F28UL, 0x0077410UL,
0x0078843DUL, 0x007CF416UL, 0x007EF326UL, 0x007FFF18UL,
0x00822239UL, 0x00831139UL, 0x0083430BUL, 0x0084530BUL,
0x0087561CUL, 0x00887F25UL, 0x00888426UL, 0x008AF61CUL,
0x008F0018UL, 0x008FFF18UL, 0x00911107UL, 0x0093230BUL,
0x0094530BUL, 0x0097743DUL, 0x00998C25UL, 0x00999616UL,
0x009EF825UL, 0x009FFF18UL, 0x00A3430BUL, 0x00A4530BUL,
0x00A7743DUL, 0x00AA9F2BUL, 0x00AAA616UL, 0x00ABD61FUL,
0x00AFFF18UL, 0x00B3330BUL, 0x00B44426UL, 0x00B7643DUL,
0x00BB971FUL, 0x00BBB53DUL, 0x00BEF512UL, 0x00BFFF18UL,
0x00C22139UL, 0x00C5330EUL, 0x00C7633DUL, 0x00CCAF2EUL,
0x00CCC616UL, 0x00CFFF18UL, 0x00D4440EUL, 0x00D6420EUL,
0x00DDCF2EUL, 0x00DDD516UL, 0x00DFFF18UL, 0x00E4330EUL,
0x00E6841CUL, 0x00EEE61CUL, 0x00EFFF18UL, 0x00F3320EUL,
0x00F55319UL, 0x00F8F41CUL, 0x00FAFF2EUL, 0x00FF002EUL,
0x00FFF10CUL, 0x00FFF33DUL, 0x00FFF722UL, 0x00FFFF18UL,
0x01000232UL, 0x0111113EUL, 0x01112103UL, 0x0111311AUL,
0x0112111AUL, 0x01122130UL, 0x01123130UL, 0x0112411DUL,
0x01131102UL, 0x01132102UL, 0x01133102UL, 0x01141108UL,
0x01142136UL, 0x01143136UL, 0x01144135UL, 0x0115223BUL,
0x01211103UL, 0x0121211AUL, 0x01213130UL, 0x01221130UL,
0x01222130UL, 0x01223102UL, 0x01231104UL, 0x01232104UL,
0x01233104UL, 0x01241139UL, 0x01241220UL, 0x01242220UL,
0x01251109UL, 0x0125223BUL, 0x0125810AUL, 0x01283212UL,
0x0131111AUL, 0x01312130UL, 0x0131222CUL, 0x0131322AUL,
0x0132122AUL, 0x0132222DUL, 0x0132322DUL, 0x01331207UL,
0x01332234UL, 0x01333234UL, 0x01341139UL, 0x01343134UL,
0x01344134UL, 0x01348134UL, 0x0135220BUL, 0x0136110BUL,
0x01365224UL, 0x01411102UL, 0x01412104UL, 0x01431239UL,
0x01432239UL, 0x0143320AUL, 0x01435134UL, 0x01443107UL,
0x01444134UL, 0x01446134UL, 0x0145220EUL, 0x01455134UL,
0x0147110EUL, 0x01511102UL, 0x01521239UL, 0x01531239UL,

FIG 22(4)

```
0x01532239UL, 0x01533107UL, 0x0155220EUL, 0x01555134UL,
0x0157110EUL, 0x01611107UL, 0x01621239UL, 0x01631239UL,
0x01661139UL, 0x01666134UL, 0x01711107UL, 0x01721239UL,
0x01745107UL, 0x0177110CUL, 0x01811107UL, 0x01821107UL,
0x0185110CUL, 0x0188210CUL, 0x01911107UL, 0x01933139UL,
0x01A11107UL, 0x01A31139UL, 0x01F5220EUL, 0x02000001UL,
0x02000127UL, 0x02000427UL, 0x02000727UL, 0x02000E2FUL,
0x02110000UL, 0x02111200UL, 0x02111411UL, 0x02111827UL,
0x02111F2FUL, 0x02112411UL, 0x02112715UL, 0x02113200UL,
0x02113411UL, 0x02113715UL, 0x02114200UL, 0x02121200UL,
0x02121301UL, 0x02121F2FUL, 0x02122200UL, 0x02122615UL,
0x02122F2FUL, 0x02123311UL, 0x02123F2FUL, 0x02124411UL,
0x02131211UL, 0x02132311UL, 0x02133211UL, 0x02184415UL,
0x02211200UL, 0x02211311UL, 0x02211F2FUL, 0x02212311UL,
0x02212F2FUL, 0x02213211UL, 0x02221201UL, 0x02221311UL,
0x02221F2FUL, 0x02222311UL, 0x02222F2FUL, 0x02223211UL,
0x02223F2FUL, 0x02231211UL, 0x02232211UL, 0x02232F2FUL,
0x02233211UL, 0x02233F2FUL, 0x02287515UL, 0x022DAB17UL,
0x02311211UL, 0x02311527UL, 0x02312211UL, 0x02321211UL,
0x02322211UL, 0x02322F2FUL, 0x02323311UL, 0x02323F2FUL,
0x02331211UL, 0x02332211UL, 0x02332F2FUL, 0x02333F2FUL,
0x0237FF17UL, 0x02385615UL, 0x023D9517UL, 0x02410027UL,
0x02487827UL, 0x024E3117UL, 0x024FFF2FUL, 0x02598627UL,
0x025DFF2FUL, 0x025FFF2FUL, 0x02687827UL, 0x026DFA17UL,
0x026FFF2FUL, 0x02796427UL, 0x027E4217UL, 0x027FFF2FUL,
0x02888727UL, 0x028EFF2FUL, 0x028FFF2FUL, 0x02984327UL,
0x029F112FUL, 0x029FFF2FUL, 0x02A76527UL, 0x02AEF717UL,
0x02AFFF2FUL, 0x02B7C827UL, 0x02BEF917UL, 0x02BFFF2FUL,
0x02C66527UL, 0x02CD5517UL, 0x02CFFF2FUL, 0x02D63227UL,
0x02DDD527UL, 0x02DFFF2BUL, 0x02E84717UL, 0x02EEE327UL,
0x02EFFF2FUL, 0x02F54527UL, 0x02FCF817UL, 0x02FFEF2BUL,
0x02FFFA2FUL, 0x02FFFE2FUL, 0x03000127UL, 0x03000201UL,
0x03111200UL, 0x03122115UL, 0x03123200UL, 0x03133211UL,
0x03211200UL, 0x03213127UL, 0x03221200UL, 0x03345215UL,
0x04000F17UL, 0x04122F17UL, 0x043F6515UL, 0x043FFF17UL,
0x044F5527UL, 0x044FFF17UL, 0x045F0017UL, 0x045FFF17UL,
0x046F6517UL, 0x04710027UL, 0x047F4427UL, 0x04810027UL,
0x048EFA15UL, 0x048FFF2FUL, 0x049F4427UL, 0x049FFF2FUL,
0x04AEA727UL, 0x04AFFF2FUL, 0x04BE9C15UL, 0x04BFFF2FUL,
0x04CE5427UL, 0x04CFFF2FUL, 0x04DE3527UL, 0x04DFFF17UL,
0x04EE4627UL, 0x04EFFF17UL, 0x04FEF327UL, 0x04FFFF2FUL,
0x06000F27UL, 0x069FFF17UL, 0x06FFFF17UL, 0x08110017UL,
0x08EFFF15UL, 0xFFFFFF00UL };
```

ari_cf_m[64][17]

2310

2312

```
unsigned short ari_cf_m[64][17] = {
{708,706,579,569,568,567,479,469,297,138,97,91,72,52,38,34,0},
{7619,6917,6519,6412,5514,5003,4683,4563,3907,3297,3125,3060,2904,2718,2631,2590,0},
{7263,4888,4810,4803,1889,415,335,327,195,72,52,49,36,20,15,14,0},
{3626,2197,2188,2187,582,57,47,46,30,12,9,8,6,4,3,2,0},
{7806,5541,5451,5441,2720,834,691,674,487,243,179,167,139,98,77,70,0},
{6684,4101,4058,4055,1748,426,368,364,322,257,235,232,228,222,217,215,0},
{9162,5964,5831,5819,3269,866,658,638,535,348,258,244,234,214,195,186,0},
{10638,8491,8365,8351,4418,2067,1859,1834,1190,601,495,478,356,217,174,164,0},
{13389,10514,10032,9961,7166,3488,2655,2524,2015,1140,760,672,585,426,325,283,0},
{14861,12788,12115,11952,9987,6657,5323,4984,4324,3001,2205,1943,1764,1394,1115,978,0},
{12876,10004,9661,9610,7107,3435,2711,2595,2257,1508,1059,952,893,753,609,538,0},
{15125,13591,13049,12874,11192,8543,7406,7023,6291,4922,4104,3769,3465,2890,2486,2275,0},
{14574,13106,12731,12638,10453,7947,7233,7037,6031,4618,4081,3906,3465,2802,2476,2349,0},
{15070,13179,12517,12351,10742,7657,6200,5825,5264,3998,3014,2662,2510,2153,1799,1564,0},
{15542,14466,14007,13844,12489,10409,9481,9132,8305,6940,6193,5867,5458,4743,4291,4047,0},
{15165,14384,14084,13934,12911,11485,10844,10513,10002,8993,8380,8051,7711,7036,6514,6233,0},
{15642,14279,13625,13393,12348,9971,8405,7858,7335,6119,4918,4376,4185,3719,3231,2860,0},
{13408,13407,11471,11218,11217,11216,9473,9216,6480,3689,2857,2690,2256,1732,1405,1302,0},
{16098,15584,15191,14931,14514,13578,12703,12103,11830,11172,10475,9867,9695,9281,8825,8389,0},
{15844,14873,14277,13996,13230,11535,10205,9543,9107,8086,7085,6419,6214,5713,5195,4731,0},
{16131,15720,15443,15276,14848,13971,13314,12910,12591,11874,11225,10788,10573,10077,9585,9209,0},
{16331,16330,12283,11435,11434,11433,8725,8049,6065,4138,3187,2842,2529,2171,1907,1745,0},
{16011,15292,14782,14528,14008,12767,11556,10921,10591,9759,8813,8043,7855,7383,6863,6282,0},
{16380,16379,15159,14610,14609,14608,12859,12111,11046,9536,8348,7713,7216,6533,5964,5546,0},
{16367,16333,16294,16253,16222,16143,16048,15947,15915,15832,15731,15619,15589,15512,15416,15310,0},
```

FIG 23(1)

{15967,15319,14937,14753,14010,12638,11787,11360,10805,9706,8934,8515,8166,7456,6911,6575,0},
{4906,3005,2985,2984,875,102,83,81,47,17,12,11,8,5,4,3,0},
{7217,4346,4269,4264,1924,428,340,332,280,203,179,175,171,164,159,157,0},
{16010,15415,15032,14805,14228,13043,12168,11634,11265,10419,9645,9110,8892,8378,7850,7437,0},
{8573,5218,5046,5032,2787,771,555,533,443,286,218,205,197,181,168,162,0},
{11474,8095,7822,7796,4632,1443,1046,1004,748,351,218,194,167,121,93,83,0},
{16152,15764,15463,15264,14925,14189,13536,13070,12846,12314,11763,11277,1131,10777,10383,10011,0},
{14187,11654,11043,10919,8498,4885,3778,3552,2947,1835,1283,1134,998,749,585,514,0},
{14162,11527,10759,10557,8601,5417,4105,3753,3286,2353,1708,1473,1370,1148,959,840,0},
{16205,15902,15669,15498,15213,14601,14068,13674,13463,12970,12471,12061,11916,11564,11183,10841,0},
{15043,12972,12092,11792,10265,7446,5934,5379,4883,3825,3036,2647,2507,2185,1901,1699,0},
{15320,13694,12782,12352,11191,8936,7433,6671,6255,5366,4622,4158,4020,3712,3420,3198,0},
{16255,16020,15768,15600,15416,14963,14440,14006,13875,13534,13137,12697,12602,12364,12084,11781,0},
{15627,14503,13906,13622,12557,10527,9269,8661,8117,6933,5994,5474,5222,4664,4166,3841,0},
{16366,16365,14547,14160,14159,14158,11969,11473,8735,6147,4911,4530,3865,3180,2710,2473,0},
{16257,16038,15871,15754,15536,15071,14673,14390,14230,13842,13452,13136,13021,12745,12434,12154,0},
{15855,14971,14338,13939,13239,11782,10585,9805,9444,8623,7846,7254,7079,6673,6262,5923,0},
{9492,6318,6197,6189,3004,652,489,477,333,143,96,90,78,60,50,47,0},
{16313,16191,16063,15968,15851,15590,15303,15082,14968,14704,14427,14177,14095,13899,13674,13457,0},
{8485,5473,5389,5383,2411,494,386,377,278,150,117,112,103,89,81,78,0},
{10497,7154,6959,6943,3788,1004,734,709,517,238,152,138,120,90,72,66,0},
{16317,16226,16127,16040,15955,15762,15547,15345,15277,15111,14922,14723,14671,14546,14396,14239,0},
{16382,16381,15858,15540,15539,15538,14704,14168,13768,13092,12452,11925,11683,11268,10841,10460,0},
{5974,3798,3758,3755,1275,205,166,162,95,35,26,24,18,11,8,7,0},
{3532,2258,2246,2244,731,135,118,115,87,45,36,34,29,21,17,16,0},
{7466,4882,4821,4811,2476,886,788,771,688,531,469,457,437,400,369,361,0},
{9580,5772,5291,5216,3444,1496,1025,928,806,578,433,384,366,331,296,273,0},

FIG 23(2)

```
{10692,7730,7543,7521,4679,1746,1391,1346,1128,692,495,458,424,353,291,268,0},
{11040,7132,6549,6452,4377,1875,1253,1130,958,631,431,370,346,296,253,227,0},
{12687,9332,8701,8585,6266,3093,2182,2004,1683,1072,712,608,559,458,373,323,0},
{13429,9853,8860,8584,6806,4039,2862,2478,2239,1764,1409,1224,1178,1077,979,903,0},
{14685,12163,11061,10668,9101,6345,4871,4263,3908,3200,2668,2368,2285,2106,1942,1819,0},
{13295,11302,10999,10945,7947,5036,4490,4385,3391,2185,1836,1757,1424,998,833,785,0},
{4992,2993,2972,2970,1269,575,552,549,530,505,497,495,493,489,486,485,0},
{15419,13862,13104,12819,11429,8753,7220,6651,6020,4667,3663,3220,2995,2511,2107,1871,0},
{12468,9263,8912,8873,5758,2193,1625,1556,1187,589,371,330,283,200,149,131,0},
{15870,15076,14615,14369,13586,12034,10990,10423,9953,8908,8031,7488,7233,6648,6101,5712,0},
{1693,978,976,975,194,18,16,15,11,7,6,5,4,3,2,1,0},
{7992,5218,5147,5143,2152,366,282,276,173,59,38,35,27,16,11,10,0} };
```

2364

FIG 23(3)

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

FIG 24

context for state calculation,
as used in a proposed scheme

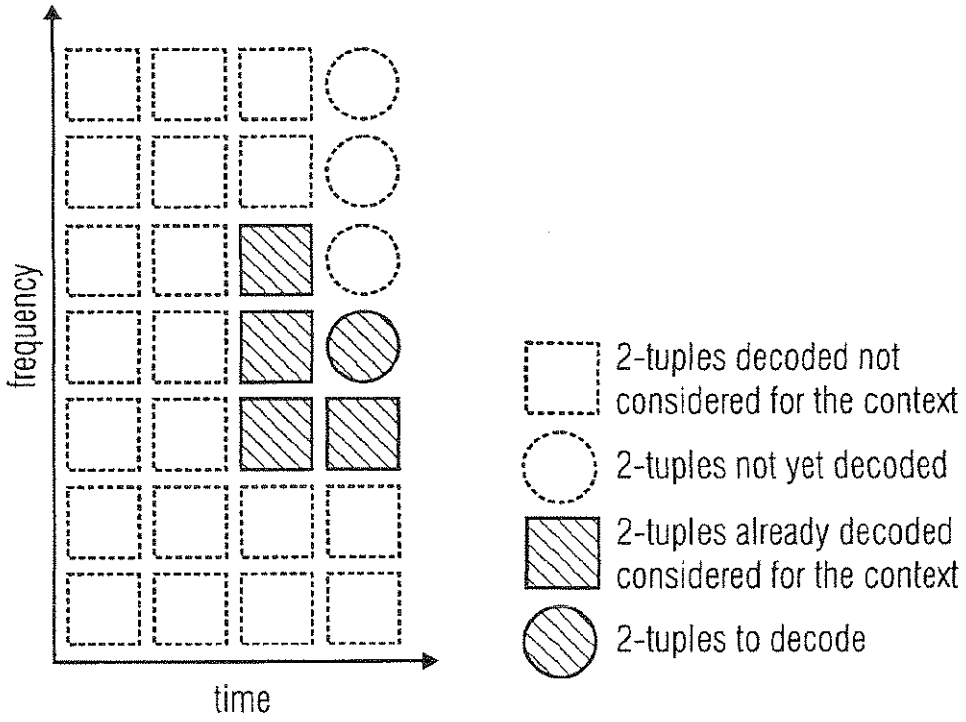


FIG 25

table: averaged coding performance for transcoding of WD6 reference quality bitstreams (average over all 9 operating points)

M17558 difference after transcoding (% of total bitrate)	new proposal: difference after transcoding (% of total bitrate)	gain new proposal over M17558 (% of total bitrate)
-1.74	-2.45	0.71

FIG 26

coding performance for transcoding of WD6 reference quality bitstreams per operating point

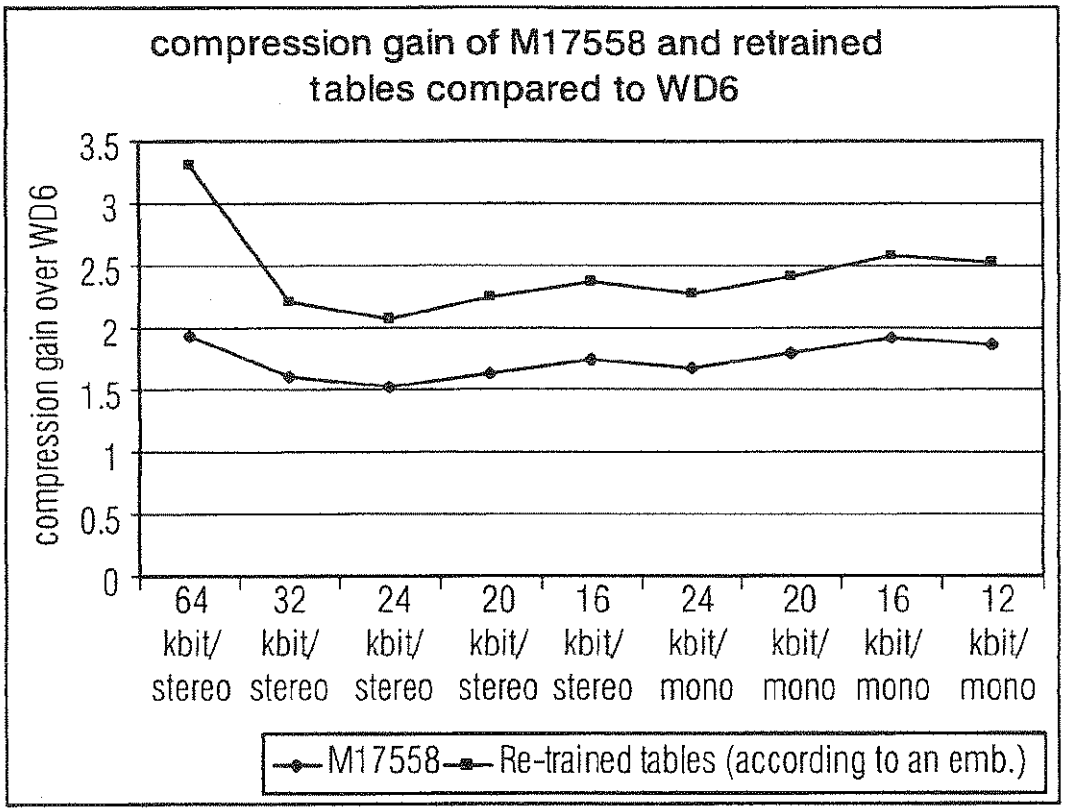


FIG 27

comparison of noiseless coder memory demand for WD6, M17588 and new proposal

	WD6 (32 bit words)	M17558 (32 bit words)	new proposal (32 bit words)
ROM	16894.5	1519.5	1441.0
RAM (per channel)	592	64	64

FIG 28

tables as used in the Re-trained coding scheme

table name	description	unit of data	memory (words of 32 Bit)
arith_hash_m[742]	Hash table mapping states of the context to a group of states	1 word	742
arith_lookup_m[742]	Look-up table mapping groups of states to a cumulative frequencies table	1/4 word	185.5
arith_cf_m[64][16]	Models of the cumulative frequencies for the most significant 2-bits wise plane m and the ARITH_ESCAPE symbol	1/2 word	512
arith_cf_r[3]	Models of the cumulative frequencies for the least significant bits	1/2 word	1.5
total			**1441.0**

FIG 29

average complexity numbers for decoding the 32 kbit/s WD6 reference quality bitstreams for the different arithmetic coder versions

	WD6 (not optimized)	WD6 (optimized)	new proposal (base)	new proposal (enhanced)
PCU (MHz)	0.826	0.594	0.600	0.601

FIG 30

average complexity numbers for decoding the 12 kbit/s WD6 reference quality bitstreams for the different arithmetic coder versions

	WD6 (not optimized)	WD6 (optimized)	new proposal (base)	new proposal (enhanced)
PCU (MHz)	0.355	0.237	0.261	0.262

FIG 31

average bitrates produced by the arithmetic coder in proposal and WD6

operating mode	average bitrate in kbit/s					
	new proposal			WD6		
	FD mode	wLPT mode	total	FD mode	wLPT mode	total
Test 1, 64kbps stereo	48.59	---	48.59	50.71	---	50.71
Test 2, 32kbps stereo	24.50	24.68	24.56	25.52	25.22	25.43
Test 3, 24kbps stereo	17.82	16.93	17.59	18.54	17.34	18.23
Test 4, 20kbps stereo	15.13	14.08	14.87	15.78	14.47	15.46
Test 5, 16kbps stereo	12.06	11.33	11.71	12.64	11.80	12.24
Test 6, 24kbps mono	19.56	17.31	18.97	20.38	17.73	19.69
Test 7, 20kbps mono	15.79	14.29	15.41	16.52	14.68	16.05
Test 8, 16kbps mono	12.64	11.69	12.18	13.30	12.17	12.75
Test 9, 12kbps mono	9.06	8.48	8.78	9.53	8.86	9.20

FIG 32

minimum, maximum and average bitrates of USAC on a frame basis using the proposed scheme

operating mode	minimum bitrate (kbit/s)	maximum bitrate (kbit/s)	average bitrate (kbit/s)
Test 1, 64kbps stereo	41.81	100.10	61.88
Test 2, 32kbps stereo	16.95	46.64	31.29
Test 3, 24kbps stereo	6.49	36.10	23.50
Test 4, 20kbps stereo	2.97	30.88	19.55
Test 5, 16kbps stereo	5.39	25.47	15.62
Test 6, 24kbps mono	3.44	35.85	23.45
Test 7, 20kbps mono	1.50	30.11	19.52
Test 8, 16kbps mono	5.92	25.52	15.59
Test 9, 12kbps mono	4.11	19.32	11.70

FIG 33

average bitrates produced by USAC coder using WD6 arithmetic coder and new proposal

operating mode	nominal bitrate (kbit/s)	new proposal (kbit/s)	difference after transcoding (kbit/s)	difference after transcoding (% of nominal bitrate)
Test 1, 64kbps stereo	64.00	61.88	-2.12	-3.31
Test 2, 32kbps stereo	32.00	31.29	-0.71	-2.21
Test 3, 24kbps stereo	24.00	23.50	-0.50	-2.07
Test 4, 20kbps stereo	20.00	19.55	-0.45	-2.25
Test 5, 16kbps stereo	16.00	15.62	-0.38	-2.37
Test 6, 24kbps mono	24.00	23.45	-0.55	-2.28
Test 7, 20kbps mono	20.00	19.52	-0.48	-2.42
Test 8, 16kbps mono	16.00	15.59	-0.41	-2.58
Test 9, 12kbps mono	12.00	11.70	-0.30	-2.52

FIG 34

best and worst cases on a frame basis

operating mode	best case		worst case	
	(kbit/s)	(%)	(kbit/s)	(%)
Test 1, 64kbps stereo	-17.88	-26.42	5.06	7.92
Test 2, 32kbps stereo	-5.90	-18.33	1.80	6.32
Test 3, 24kbps stereo	-4.29	-18.36	1.72	7.76
Test 4, 20kbps stereo	-4.28	-18.32	0.55	3.22
Test 5, 16kbps stereo	-3.25	-20.63	1.25	8.90
Test 6, 24kbps mono	-4.97	-19.61	1.72	8.53
Test 7, 20kbps mono	-4.17	-20.60	1.98	11.67
Test 8, 16kbps mono	-3.18	-19.89	1.15	7.22
Test 9, 12kbps mono	-2.32	-17.52	1.30	11.50

FIG 35

bitreservoir limits

operating mode	bitreservoir control					
	new proposal			WD6		
	min	max	avg	min	max	avg
Test 1, 64kbps stereo	4009	9557	9103	1676	9557	6159
Test 2, 32kbps stereo	2161	4505	4203	770	4505	3315
Test 3, 24kbps stereo	1565	4704	4121	1045	4704	3253
Test 4, 20kbps stereo	2013	4864	4372	1328	4864	3212
Test 5, 16kbps stereo	2197	5006	4472	1494	5006	3335
Test 6, 24kbps mono	1847	4704	4221	738	4704	3272
Test 7, 20kbps mono	2253	4864	4456	1272	4864	3358
Test 8, 16kbps mono	2596	5006	4549	1505	5006	3666
Test 9, 12kbps mono	1974	5184	4494	1256	5184	3556

FIG 36

table 7.1 - 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 37

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.	changes
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	changes
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	changes
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. changes	
esc_nb	Number of ARITH_ESCAPE symbol already decoded for the present 2-tuple. The value is bounded to 7.	changes
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 38

Definitions

a,b	The 2-tuple quantized coefficient to decode	changes
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.	changes
arith_lookup_m[]	Look-up table mapping group of context states to a cumulative frequencies table index pki.	changes
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 [] (change)	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.	changes
N	Window length. For AAC it is deduced from the window_sequence (see section 6.8.3.1) and for TCX N=2.lg.	changes

FIG 39

```
/*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 40A

FIG 40B

```
/*Input variables*/
c /*old state context*/
i /*index of the 4-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][1]<<12);
    c=(c&0xFFF0)+(q[1][-1]);

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

    return(c);
}
```

FIG 40C

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

pki=arith_get_pk(c){
    i_min=-1;
    i=i_min;
    i_max=741;

    while((i_max-i_min)>1){
        i=i_min+((i_max-i_min)/2);
        j=ari_hash_m[i];
        if(c<(j>>8))
            i_max=i;
        else if(c>(j>>8))
            i_min=i;
        else
            return(j&0xFF);
    }

    return ari_lookup_m[i_max];
}
```

FIG 40D

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

/* global variables */
low
high
value

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

symbol=arith_decode(cum_freq,cfl)
{
    if(arith_first_symbol())
    {
        value = 0;
        for (i=1; i<=20; i++)
        {
        value = (val<<1) | arith_get_next_bit();
        }
        s->low=0;
        s->high= 65535;
        s->vobf= value;

    }

    low = s->low;
    high = s->high;
    value = s->vobf;

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

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

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

<continued in FIG 40E>

FIG 40E <continuation of FIG 40D>

```
low  += ((range*(cum_freq[symbol-1]))>>stat_bitsnew);

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();
    bp++;
}

s->low = low;
s->high = high;
s->vobf = value;

return(symbol);
}
```

FIG 40F

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

}
```

FIG 40G

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

arith_update_context(){
    qdec[2*i]=a
    qdec[2*i+1]=b;
    q[1][i]=a+b+1;
    if(q[1][i]>0xF)
        q[1][i]=0xF;
}
```

FIG 40H

```
/*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/2;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[k] = q[1][j];
        }
        previous_lg = 512;
    }
    else{
        for(j=0; j<512; j++){
            qs[j] = q[1][j];
        }
        previous_lg = MIN(1024,lg);
    }
}
```

FIG 41(1)

set of Re-trained Tables

```
static unsigned short ari_lookup_m[742] = {
0x01,0x34,0x0D,0x13,0x12,0x25,0x00,0x3A,
0x05,0x00,0x21,0x13,0x1F,0x1A,0x1D,0x36,
0x24,0x2B,0x1B,0x33,0x37,0x29,0x1D,0x33,
0x37,0x33,0x37,0x33,0x37,0x33,0x2C,0x00,
0x21,0x13,0x25,0x2A,0x00,0x21,0x24,0x12,
0x2C,0x1E,0x37,0x24,0x1F,0x35,0x37,0x24,
0x35,0x37,0x35,0x37,0x38,0x2D,0x21,0x29,
0x1E,0x21,0x13,0x2D,0x36,0x38,0x29,0x36,
0x37,0x24,0x36,0x38,0x37,0x38,0x00,0x20,
0x23,0x20,0x23,0x36,0x38,0x24,0x3B,0x24,
0x26,0x29,0x1F,0x30,0x2D,0x0D,0x12,0x3F,
0x2D,0x21,0x1C,0x2A,0x00,0x21,0x12,0x1E,
0x36,0x38,0x36,0x37,0x3F,0x1E,0x0D,0x1F,
0x2A,0x1E,0x21,0x24,0x12,0x2A,0x3C,0x21,
0x24,0x1F,0x3C,0x21,0x29,0x36,0x38,0x36,
0x37,0x38,0x21,0x1E,0x00,0x3B,0x25,0x1E,
0x20,0x10,0x1F,0x3C,0x20,0x23,0x29,0x08,
0x23,0x12,0x08,0x23,0x21,0x38,0x00,0x20,
0x13,0x20,0x3B,0x1C,0x20,0x3B,0x29,0x20,
0x23,0x24,0x21,0x24,0x21,0x24,0x3B,0x13,
0x23,0x26,0x23,0x13,0x21,0x24,0x26,0x29,
0x12,0x22,0x2B,0x02,0x1E,0x0D,0x1F,0x2D,
0x00,0x0D,0x12,0x00,0x3C,0x21,0x29,0x3C,
0x21,0x2A,0x3C,0x3B,0x22,0x1E,0x20,0x10,
0x1F,0x3C,0x0D,0x29,0x3C,0x21,0x24,0x08,
0x23,0x20,0x38,0x39,0x3C,0x20,0x13,0x3C,
0x00,0x0D,0x13,0x1F,0x3C,0x09,0x26,0x1F,
0x08,0x09,0x26,0x12,0x08,0x23,0x29,0x20,
0x23,0x21,0x24,0x20,0x13,0x20,0x3B,0x16,
0x20,0x3B,0x29,0x20,0x3B,0x29,0x20,0x3B,
0x13,0x21,0x24,0x29,0x0B,0x13,0x09,0x3B,
0x13,0x09,0x3B,0x13,0x21,0x3B,0x13,0x0D,
0x26,0x29,0x26,0x29,0x3D,0x12,0x22,0x28,
0x2E,0x04,0x08,0x13,0x3C,0x3B,0x3C,0x20,
0x10,0x3C,0x21,0x07,0x08,0x10,0x00,0x08,
0x0D,0x29,0x08,0x0D,0x29,0x08,0x09,0x13,
0x20,0x23,0x39,0x08,0x09,0x13,0x08,0x09,
0x16,0x08,0x09,0x10,0x12,0x20,0x3B,0x3D,
0x09,0x26,0x20,0x3B,0x24,0x39,0x09,0x26,
0x20,0x0D,0x13,0x00,0x09,0x13,0x20,0x0D,
0x26,0x12,0x20,0x3B,0x13,0x21,0x26,0x0B,
0x12,0x09,0x3B,0x16,0x09,0x3B,0x3D,0x09,
0x26,0x0D,0x13,0x26,0x3D,0x1C,0x12,0x1F,
0x28,0x2E,0x07,0x0B,0x08,0x09,0x00,0x39,
```

FIG 41(2)

```
0x0B,0x08,0x26,0x08,0x09,0x13,0x20,0x0B,
0x39,0x10,0x39,0x0D,0x13,0x20,0x10,0x12,
0x09,0x13,0x20,0x3B,0x13,0x09,0x26,0x0B,
0x09,0x3B,0x1C,0x09,0x3B,0x13,0x20,0x3B,
0x13,0x09,0x26,0x0B,0x16,0x0D,0x13,0x09,
0x13,0x09,0x13,0x26,0x3D,0x1C,0x1F,0x28,
0x2E,0x07,0x10,0x39,0x0B,0x39,0x39,0x13,
0x39,0x0B,0x39,0x0B,0x39,0x26,0x39,0x10,
0x20,0x3B,0x16,0x20,0x10,0x09,0x26,0x0B,
0x13,0x09,0x13,0x26,0x1C,0x0B,0x3D,0x1C,
0x1F,0x28,0x2B,0x07,0x0C,0x39,0x0B,0x39,
0x0B,0x0C,0x0B,0x26,0x0B,0x26,0x3D,0x0D,
0x1C,0x14,0x28,0x2B,0x39,0x0B,0x0C,0x0E,
0x3D,0x1C,0x0D,0x12,0x22,0x2B,0x07,0x0C,
0x0E,0x3D,0x1C,0x10,0x1F,0x2B,0x0C,0x0E,
0x19,0x14,0x10,0x1F,0x28,0x0C,0x0E,0x19,
0x14,0x26,0x22,0x2B,0x0C,0x0E,0x19,0x14,
0x26,0x28,0x0E,0x19,0x14,0x26,0x28,0x0E,
0x19,0x14,0x28,0x0E,0x19,0x14,0x22,0x28,
0x2B,0x0E,0x14,0x2B,0x31,0x00,0x3A,0x3A,
0x05,0x05,0x1B,0x1D,0x33,0x06,0x35,0x35,
0x20,0x21,0x37,0x21,0x24,0x05,0x1B,0x2C,
0x2C,0x2C,0x06,0x34,0x1E,0x34,0x00,0x08,
0x36,0x09,0x21,0x26,0x1C,0x2C,0x00,0x02,
0x02,0x02,0x3F,0x04,0x04,0x04,0x34,0x39,
0x20,0x0A,0x0C,0x39,0x0B,0x0F,0x07,0x07,
0x07,0x07,0x34,0x39,0x39,0x0A,0x0C,0x39,
0x0C,0x0F,0x07,0x07,0x07,0x00,0x39,0x39,
0x0C,0x0F,0x07,0x07,0x39,0x0C,0x0F,0x07,
0x39,0x0C,0x0F,0x39,0x39,0x0C,0x0F,0x39,
0x0C,0x39,0x0C,0x0F,0x00,0x11,0x27,0x17,
0x2F,0x27,0x00,0x27,0x17,0x00,0x11,0x17,
0x00,0x11,0x17,0x11,0x00,0x27,0x15,0x11,
0x17,0x01,0x15,0x11,0x15,0x11,0x15,0x15,
0x17,0x00,0x27,0x01,0x27,0x27,0x15,0x00,
0x27,0x11,0x27,0x15,0x15,0x15,0x27,0x15,
0x15,0x15,0x15,0x17,0x2F,0x11,0x17,0x27,
0x27,0x27,0x11,0x27,0x15,0x27,0x27,0x15,
0x15,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,
0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,
0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,
0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,0x27,
0x17,0x2F,0x27,0x17,0x2F,0x27,0x17,0x2F,
0x17,0x2F,0x2B,0x00,0x27,0x00,0x00,0x11,
0x15,0x00,0x11,0x11,0x27,0x27,0x15,0x17,
0x15,0x17,0x15,0x17,0x27,0x17,0x27,0x17,
0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,
0x27,0x17,0x27,0x17,0x27,0x17,0x27,0x17,
0x27,0x15,0x27,0x27,0x15,0x27,
};
```

```
static unsigned long ari_hash_m[742] = {
0x00000104UL,0x0000030AUL,0x00000510UL,0x00000716UL,0x00000A1FUL,0x00000F2E
UL,0x00011100UL,0x00111103UL,
0x00111306UL,0x00111436UL,0x00111623UL,0x00111929UL,0x00111F2EUL,0x0011221BU
L,0x00112435UL,0x00112621UL,
0x00112D12UL,0x00113130UL,0x0011331DUL,0x00113535UL,0x00113938UL,0x0011411B
UL,0x00114433UL,0x00114635UL,
0x00114F29UL,0x00116635UL,0x00116F24UL,0x00117433UL,0x0011FF0FUL,0x00121102U
L,0x0012132DUL,0x00121436UL,
0x00121623UL,0x00121912UL,0x0012213FUL,0x0012232DUL,0x00122436UL,0x00122638U
L,0x00122A29UL,0x00122F2BUL,
0x0012322DUL,0x00123436UL,0x00123738UL,0x00123B29UL,0x0012411DUL,0x00124536
UL,0x00124938UL,0x00124F12UL,
0x00125535UL,0x00125F29UL,0x00126535UL,0x0012B837UL,0x0013112AUL,0x0013131E
UL,0x0013163BUL,0x0013212DUL,
0x0013233CUL,0x00132623UL,0x00132F2EUL,0x0013321EUL,0x00133521UL,0x00133824
UL,0x0013411EUL,0x00134336UL,
0x00134838UL,0x00135135UL,0x00135537UL,0x00135F12UL,0x00137637UL,0x0013FF29U
L,0x00140024UL,0x00142321UL,
0x00143136UL,0x00143321UL,0x00143F25UL,0x00144321UL,0x00148638UL,0x0014FF29U
L,0x00154323UL,0x0015FF12UL,
0x0016F20CUL,0x0018A529UL,0x00210031UL,0x0021122CUL,0x00211408UL,0x00211713
UL,0x00211F2EUL,0x0021222AUL,
0x00212408UL,0x00212710UL,0x00212F2EUL,0x0021331EUL,0x00213436UL,0x00213824U
L,0x0021412DUL,0x0021431EUL,
0x00214536UL,0x00214F1FUL,0x00216637UL,0x00220004UL,0x0022122AUL,0x00221420U
L,0x00221829UL,0x00221F2EUL,
0x0022222DUL,0x00222408UL,0x00222623UL,0x00222929UL,0x00222F2BUL,0x0022321E
UL,0x00223408UL,0x00223724UL,
0x00223A29UL,0x0022411EUL,0x00224436UL,0x00224823UL,0x00225134UL,0x00225621U
L,0x00225F12UL,0x00226336UL,
0x00227637UL,0x0022FF29UL,0x0023112DUL,0x0023133CUL,0x00231420UL,0x00231916
UL,0x0023212DUL,0x0023233CUL,
0x00232509UL,0x00232929UL,0x0023312DUL,0x00233308UL,0x00233509UL,0x00233724U
L,0x0023413CUL,0x00234421UL,
0x00234A13UL,0x0023513CUL,0x00235421UL,0x00235F1FUL,0x00236421UL,0x0023FF29
UL,0x00240024UL,0x0024153BUL,
0x00242108UL,0x00242409UL,0x00242726UL,0x00243108UL,0x00243409UL,0x00243610U
L,0x00244136UL,0x00244321UL,
0x00244523UL,0x00244F1FUL,0x00245423UL,0x0024610AUL,0x00246423UL,0x0024FF29
UL,0x00252510UL,0x00253121UL,
0x0025343BUL,0x00254121UL,0x00254510UL,0x00254F25UL,0x00255221UL,0x0025FF12U
L,0x00266513UL,0x0027F529UL,
0x0029F101UL,0x002CF224UL,0x00310030UL,0x0031122AUL,0x00311420UL,0x00311816
UL,0x0031212CUL,0x0031231EUL,
0x00312408UL,0x00312710UL,0x0031312AUL,0x0031321EUL,0x00313408UL,0x00313623U
L,0x0031411EUL,0x0031433CUL,
```

FIG 42(1)

0x00320007UL,0x0032122DUL,0x00321420UL,0x00321816UL,0x0032212DUL,0x0032233C
UL,0x00322509UL,0x00322916UL,
0x0032312DUL,0x00323420UL,0x00323710UL,0x00323F2BUL,0x00324308UL,0x00324623
UL,0x00324F25UL,0x00325421UL,
0x00325F1FUL,0x00326421UL,0x0032FF29UL,0x00331107UL,0x00331308UL,0x0033150D
UL,0x0033211EUL,0x00332308UL,
0x00332420UL,0x00332610UL,0x00332929UL,0x0033311EUL,0x00333308UL,0x0033363BU
L,0x00333A29UL,0x0033413CUL,
0x00334320UL,0x0033463BUL,0x00334A29UL,0x0033510AUL,0x00335320UL,0x00335824
UL,0x0033610AUL,0x00336321UL,
0x00336F12UL,0x00337623UL,0x00341139UL,0x0034153BUL,0x00342108UL,0x00342409U
L,0x00342610UL,0x00343108UL,
0x00343409UL,0x00343610UL,0x00344108UL,0x0034440DUL,0x00344610UL,0x0034510A
UL,0x00345309UL,0x0034553BUL,
0x0034610AUL,0x00346309UL,0x0034F824UL,0x00350029UL,0x00352510UL,0x00353120U
L,0x0035330DUL,0x00353510UL,
0x00354120UL,0x0035430DUL,0x00354510UL,0x00354F28UL,0x0035530DUL,0x00355510
UL,0x00355F1FUL,0x00356410UL,
0x00359626UL,0x0035FF12UL,0x00366426UL,0x0036FF12UL,0x0037F426UL,0x0039D712
UL,0x003BF612UL,0x003DF81FUL,
0x00410004UL,0x00411207UL,0x0041150DUL,0x0041212AUL,0x00412420UL,0x0041311E
UL,0x00413308UL,0x00413509UL,
0x00413F2BUL,0x00414208UL,0x00420007UL,0x0042123CUL,0x00421409UL,0x00422107
UL,0x0042223CUL,0x00422409UL,
0x00422610UL,0x0042313CUL,0x00423409UL,0x0042363BUL,0x0042413CUL,0x00424320
UL,0x0042463BUL,0x00425108UL,
0x00425409UL,0x0042FF29UL,0x00431107UL,0x00431320UL,0x0043153BUL,0x0043213C
UL,0x00432320UL,0x00432610UL,
0x0043313CUL,0x00433320UL,0x0043353BUL,0x00433813UL,0x00434108UL,0x00434409U
L,0x00434610UL,0x00435108UL,
0x0043553BUL,0x00435F25UL,0x00436309UL,0x0043753BUL,0x0043FF29UL,0x00441239
UL,0x0044143BUL,0x00442139UL,
0x00442309UL,0x0044253BUL,0x00443108UL,0x00443220UL,0x0044353BUL,0x0044410A
UL,0x00444309UL,0x0044453BUL,
0x00444813UL,0x0044510AUL,0x00445309UL,0x00445510UL,0x00445F25UL,0x0044630D
UL,0x00450026UL,0x00452713UL,
0x00453120UL,0x0045330DUL,0x00453510UL,0x00454120UL,0x0045430DUL,0x00454510
UL,0x00455120UL,0x0045530DUL,
0x00456209UL,0x00456410UL,0x0045FF12UL,0x00466513UL,0x0047FF22UL,0x0048FF25U
L,0x0049F43DUL,0x004BFB25UL,
0x004EF825UL,0x004FFF18UL,0x00511339UL,0x00512107UL,0x00513409UL,0x00520007U
L,0x00521107UL,0x00521320UL,
0x00522107UL,0x00522409UL,0x0052313CUL,0x00523320UL,0x0052353BUL,0x00524108U
L,0x00524320UL,0x00531139UL,
0x00531309UL,0x00532139UL,0x00532309UL,0x0053253BUL,0x00533108UL,0x0053340D
UL,0x00533713UL,0x00534108UL,
0x0053453BUL,0x00534F2BUL,0x00535309UL,0x00535610UL,0x00535F25UL,0x0053643B
UL,0x00541139UL,0x00542139UL,

FIG 42(2)

0x00542309UL,0x00542613UL,0x00543139UL,0x00543309UL,0x00543510UL,0x00543F2BU
L,0x00544309UL,0x00544510UL,
0x00544F28UL,0x0054530DUL,0x0054FF12UL,0x00553613UL,0x00553F2BUL,0x00554410
UL,0x0055510AUL,0x0055543BUL,
0x00555F25UL,0x0055633BUL,0x0055FF12UL,0x00566513UL,0x00577413UL,0x0059FF28
UL,0x005CC33DUL,0x005EFB28UL,
0x005FFF18UL,0x00611339UL,0x00612107UL,0x00613320UL,0x0061A724UL,0x00621107
UL,0x0062140BUL,0x00622107UL,
0x00622320UL,0x00623139UL,0x00623320UL,0x00631139UL,0x0063130CUL,0x00632139U
L,0x00632309UL,0x00633139UL,
0x00633309UL,0x00633626UL,0x00633F2BUL,0x00634309UL,0x00634F2BUL,0x0063543B
UL,0x0063FF12UL,0x0064343BUL,
0x00643F2BUL,0x0064443BUL,0x00645209UL,0x00665513UL,0x006661 0AUL,0x00666526
UL,0x0067A616UL,0x0069843DUL,
0x006CF612UL,0x006EF326UL,0x006FFF18UL,0x0071130CUL,0x00721107UL,0x00722239
UL,0x0072291CUL,0x0072340BUL,
0x00731139UL,0x00732239UL,0x0073630BUL,0x0073FF12UL,0x0074430BUL,0x00755426
UL,0x00776F28UL,0x00777410UL,
0x0078843DUL,0x007CF416UL,0x007EF326UL,0x007FFF18UL,0x00822239UL,0x00831139
UL,0x0083430BUL,0x0084530BUL,
0x0087561CUL,0x00887F25UL,0x00888426UL,0x008AF61CUL,0x008F0018UL,0x008FFF18
UL,0x00911107UL,0x0093230BUL,
0x0094530BUL,0x0097743DUL,0x00998C25UL,0x00999616UL,0x009EF825UL,0x009FFF18
UL,0x00A3430BUL,0x00A4530BUL,
0x00A7743DUL,0x00AA9F2BUL,0x00AAA616UL,0x00ABD61FUL,0x00AFFF18UL,0x00B3
330BUL,0x00B44426UL,0x00B7643DUL,
0x00BB971FUL,0x00BBB53DUL,0x00BEF512UL,0x00BFFF18UL,0x00C22139UL,0x00C53
30EUL,0x00C7633DUL,0x00CCAF2EUL,
0x00CCC616UL,0x00CFFF18UL,0x00D4440EUL,0x00D6420EUL,0x00DDCF2EUL,0x00DD
D516UL,0x00DFFF18UL,0x00E4330EUL,
0x00E6841CUL,0x00EEE61CUL,0x00EFFF18UL,0x00F3320EUL,0x00F55319UL,0x00F8F41
CUL,0x00FAFF2EUL,0x00FF002EUL,
0x00FFF10CUL,0x00FFF33DUL,0x00FFF722UL,0x00FFFF18UL,0x01000232UL,0x0111113
EUL,0x01112103UL,0x0111311AUL,
0x0112111AUL,0x01122130UL,0x01123130UL,0x0112411DUL,0x01131102UL,0x01132102
UL,0x01133102UL,0x01141108UL,
0x01142136UL,0x01143136UL,0x01144135UL,0x0115223BUL,0x01211103UL,0x0121211A
UL,0x01213130UL,0x01221130UL,
0x01222130UL,0x01223102UL,0x01231104UL,0x01232104UL,0x01233104UL,0x01241139U
L,0x01241220UL,0x01242220UL,
0x01251109UL,0x0125223BUL,0x0125810AUL,0x01283212UL,0x0131111AUL,0x01312130
UL,0x0131222CUL,0x0131322AUL,
0x0132122AUL,0x0132222DUL,0x0132322DUL,0x01331207UL,0x01332234UL,0x01333234
UL,0x01341139UL,0x01343134UL,
0x01344134UL,0x01348134UL,0x0135220BUL,0x0136110BUL,0x01365224UL,0x01411102U
L,0x01412104UL,0x01431239UL,
0x01432239UL,0x0143320AUL,0x01435134UL,0x01443107UL,0x01444134UL,0x01446134U
L,0x0145220EUL,0x01455134UL,

FIG 42(3)

```
0x0147110EUL,0x01511102UL,0x01521239UL,0x01531239UL,0x01532239UL,0x01533107U
L,0x0155220EUL,0x01555134UL,
0x0157110EUL,0x01611107UL,0x01621239UL,0x01631239UL,0x01661139UL,0x01666134U
L,0x01711107UL,0x01721239UL,
0x01745107UL,0x0177110CUL,0x01811107UL,0x01821107UL,0x0185110CUL,0x0188210C
UL,0x01911107UL,0x01933139UL,
0x01A11107UL,0x01A31139UL,0x01F5220EUL,0x02000001UL,0x02000127UL,0x02000427
UL,0x02000727UL,0x02000E2FUL,
0x02110000UL,0x02111200UL,0x02111411UL,0x02111827UL,0x02111F2FUL,0x02112411U
L,0x02112715UL,0x02113200UL,
0x02113411UL,0x02113715UL,0x02114200UL,0x02121200UL,0x02121301UL,0x02121F2FU
L,0x02122200UL,0x02122615UL,
0x02122F2FUL,0x02123311UL,0x02123F2FUL,0x02124411UL,0x02131211UL,0x02132311U
L,0x02133211UL,0x02184415UL,
0x02211200UL,0x02211311UL,0x02211F2FUL,0x02212311UL,0x02212F2FUL,0x02213211U
L,0x02221201UL,0x02221311UL,
0x02221F2FUL,0x02222311UL,0x02222F2FUL,0x02223211UL,0x02223F2FUL,0x02231211U
L,0x02232211UL,0x02232F2FUL,
0x02233211UL,0x02233F2FUL,0x02287515UL,0x022DAB17UL,0x02311211UL,0x02311527
UL,0x02312211UL,0x02321211UL,
0x02322211UL,0x02322F2FUL,0x02323311UL,0x02323F2FUL,0x02331211UL,0x02332211U
L,0x02332F2FUL,0x02333F2FUL,
0x0237FF17UL,0x02385615UL,0x023D9517UL,0x02410027UL,0x02487827UL,0x024E3117
UL,0x024FFF2FUL,0x02598627UL,
0x025DFF2FUL,0x025FFF2FUL,0x02687827UL,0x026DFA17UL,0x026FFF2FUL,0x0279642
7UL,0x027E4217UL,0x027FFF2FUL,
0x02888727UL,0x028EFF2FUL,0x028FFF2FUL,0x02984327UL,0x029F112FUL,0x029FFF2F
UL,0x02A76527UL,0x02AEF717UL,
0x02AFFF2FUL,0x02B7C827UL,0x02BEF917UL,0x02BFFF2FUL,0x02C66527UL,0x02CD55
17UL,0x02CFFF2FUL,0x02D63227UL,
0x02DDD527UL,0x02DFFF2BUL,0x02E84717UL,0x02EEE327UL,0x02EFFF2FUL,0x02F545
27UL,0x02FCF817UL,0x02FFEF2BUL,
0x02FFFA2FUL,0x02FFFE2FUL,0x03000127UL,0x03000201UL,0x03111200UL,0x03122115
UL,0x03123200UL,0x03133211UL,
0x03211200UL,0x03213127UL,0x03221200UL,0x03345215UL,0x04000F17UL,0x04122F17U
L,0x043F6515UL,0x043FFF17UL,
0x044F5527UL,0x044FFF17UL,0x045F0017UL,0x045FFF17UL,0x046F6517UL,0x04710027
UL,0x047F4427UL,0x04810027UL,
0x048EFA15UL,0x048FFF2FUL,0x049F4427UL,0x049FFF2FUL,0x04AEA727UL,0x04AFFF
2FUL,0x04BE9C15UL,0x04BFFF2FUL,
0x04CE5427UL,0x04CFFF2FUL,0x04DE3527UL,0x04DFFF17UL,0x04EE4627UL,0x04EFFF
17UL,0x04FEF327UL,0x04FFFF2FUL,
0x06000F27UL,0x069FFF17UL,0x06FFFF17UL,0x08110017UL,0x08EFFF15UL,0x7FFFFFF
F00UL,
};
```

FIG 42(4)

```
unsigned short ari_cf_m[96][17] = {
{ 708, 706, 579, 569, 568, 567, 479, 469, 297, 138, 97, 91, 72, 52, 38, 34,
0
},
{ 7619, 6917, 6519, 6412, 5514, 5003, 4683, 4563, 3907, 3297, 3125, 3060, 2904, 2718, 2631,
2590,
0
},
{ 7263, 4888, 4810, 4803, 1889, 415, 335, 327, 195, 72, 52, 49, 36, 20, 15, 14,
0
},
{ 3626, 2197, 2188, 2187, 582, 57, 47, 46, 30, 12, 9, 8, 6, 4, 3, 2,
0
},
{ 7806, 5541, 5451, 5441, 2720, 834, 691, 674, 487, 243, 179, 167, 139, 98, 77, 70,
0
},
{ 6684, 4101, 4058, 4055, 1748, 426, 368, 364, 322, 257, 235, 232, 228, 222, 217, 215,
0
},
{ 9162, 5964, 5831, 5819, 3269, 866, 658, 638, 535, 348, 258, 244, 234, 214, 195, 186,
0
},
{ 10638, 8491, 8365, 8351, 4418, 2067, 1859, 1834, 1190, 601, 495, 478, 356, 217, 174,
164,
0
},
{ 13389,10514,10032, 9961, 7166, 3488, 2655, 2524, 2015, 1140, 760, 672, 585, 426, 325,
283,
0
},
{ 14861,12788,12115,11952, 9987, 6657, 5323, 4984, 4324, 3001, 2205, 1943, 1764, 1394,
1115, 978,
0
},
{ 12876,10004, 9661, 9610, 7107, 3435, 2711, 2595, 2257, 1508, 1059, 952, 893, 753, 609,
538,
0
},
{ 15125,13591,13049,12874,11192, 8543, 7406, 7023, 6291, 4922, 4104, 3769, 3465, 2890,
2486, 2275,
0
},
{ 14574,13106,12731,12638,10453, 7947, 7233, 7037, 6031, 4618, 4081, 3906, 3465, 2802,
2476, 2349,
0
```

FIG 43(1)

},
{ 15070,13179,12517,12351,10742, 7657, 6200, 5825, 5264, 3998, 3014, 2662, 2510, 2153,
1799, 1564,
0
},
{ 15542,14466,14007,13844,12489,10409, 9481, 9132, 8305, 6940, 6193, 5867, 5458, 4743,
4291, 4047,
0
},
{ 15165,14384,14084,13934,12911,11485,10844,10513,10002, 8993, 8380, 8051, 7711, 7036,
6514, 6233,
0
},
{ 15642,14279,13625,13393,12348, 9971, 8405, 7858, 7335, 6119, 4918, 4376, 4185, 3719,
3231, 2860,
0
},
{ 13408,13407,11471,11218,11217,11216, 9473, 9216, 6480, 3689, 2857, 2690, 2256, 1732,
1405, 1302,
0
},
{ 16098,15584,15191,14931,14514,13578,12703,12103,11830,11172,10475, 9867, 9695, 9281,
8825, 8389,
0
},
{ 15844,14873,14277,13996,13230,11535,10205, 9543, 9107, 8086, 7085, 6419, 6214, 5713,
5195, 4731,
0
},
{
16131,15720,15443,15276,14848,13971,13314,12910,12591,11874,11225,10788,10573,10077,
9585, 9209,
0
},
{ 16331,16330,12283,11435,11434,11433, 8725, 8049, 6065, 4138, 3187, 2842, 2529, 2171,
1907, 1745,
0
},
{ 16011,15292,14782,14528,14008,12767,11556,10921,10591, 9759, 8813, 8043, 7855, 7383,
6863, 6282,
0
},
{ 16380,16379,15159,14610,14609,14608,12859,12111,11046, 9536, 8348, 7713, 7216, 6533,
5964, 5546,
0
},

FIG 43(2)

```
{
16367,16333,16294,16253,16222,16143,16048,15947,15915,15832,15731,15619,15589,15512,
15416,15310,
0
},
{ 15967,15319,14937,14753,14010,12638,11787,11360,10805, 9706, 8934, 8515, 8166, 7456,
6911, 6575,
0
},
{ 4906, 3005, 2985, 2984, 875, 102, 83, 81, 47, 17, 12, 11, 8, 5, 4, 3,
0
},
{ 7217, 4346, 4269, 4264, 1924, 428, 340, 332, 280, 203, 179, 175, 171, 164, 159, 157,
0
},
{ 16010,15415,15032,14805,14228,13043,12168,11634,11265,10419, 9645, 9110, 8892, 8378,
7850, 7437,
0
},
{ 8573, 5218, 5046, 5032, 2787, 771, 555, 533, 443, 286, 218, 205, 197, 181, 168, 162,
0
},
{ 11474, 8095, 7822, 7796, 4632, 1443, 1046, 1004, 748, 351, 218, 194, 167, 121, 93, 83,
0
},
{
16152,15764,15463,15264,14925,14189,13536,13070,12846,12314,11763,11277,11131,10777,
10383,10011,
0
},
{ 14187,11654,11043,10919, 8498, 4885, 3778, 3552, 2947, 1835, 1283, 1134, 998, 749, 585,
514,
0
},
{ 14162,11527,10759,10557, 8601, 5417, 4105, 3753, 3286, 2353, 1708, 1473, 1370, 1148,
959, 840,
0
},
{
16205,15902,15669,15498,15213,14601,14068,13674,13463,12970,12471,12061,11916,11564,
11183,10841,
0
},
{ 15043,12972,12092,11792,10265, 7446, 5934, 5379, 4883, 3825, 3036, 2647, 2507, 2185,
1901, 1699,
0
```

FIG 43(3)

```
},
{ 15320,13694,12782,12352,11191, 8936, 7433, 6671, 6255, 5366, 4622, 4158, 4020, 3712,
3420, 3198,
0
},
{
16255,16020,15768,15600,15416,14963,14440,14006,13875,13534,13137,12697,12602,12364,
12084,11781,
0
},
{ 15627,14503,13906,13622,12557,10527, 9269, 8661, 8117, 6933, 5994, 5474, 5222, 4664,
4166, 3841,
0
},
{ 16366,16365,14547,14160,14159,14158,11969,11473, 8735, 6147, 4911, 4530, 3865, 3180,
2710, 2473,
0
},
{
16257,16038,15871,15754,15536,15071,14673,14390,14230,13842,13452,13136,13021,12745,
12434,12154,
0
},
{ 15855,14971,14338,13939,13239,11782,10585, 9805, 9444, 8623, 7846, 7254, 7079, 6673,
6262, 5923,
0
},
{ 9492, 6318, 6197, 6189, 3004, 652, 489, 477, 333, 143, 96, 90, 78, 60, 50, 47,
0
},
{
16313,16191,16063,15968,15851,15590,15303,15082,14968,14704,14427,14177,14095,13899,
13674,13457,
0
},
{ 8485, 5473, 5389, 5383, 2411, 494, 386, 377, 278, 150, 117, 112, 103, 89, 81, 78,
0
},
{ 10497, 7154, 6959, 6943, 3788, 1004, 734, 709, 517, 238, 152, 138, 120, 90, 72, 66,
0
},
{
16317,16226,16127,16040,15955,15762,15547,15345,15277,15111,14922,14723,14671,14546,
14396,14239,
0
},
```

FIG 43(4)

```
{
16382,16381,15858,15540,15539,15538,14704,14168,13768,13092,12452,11925,11683,11268,
10841,10460,
0
},
{ 5974, 3798, 3758, 3755, 1275, 205, 166, 162, 95, 35, 26, 24, 18, 11, 8, 7,
0
},
{ 3532, 2258, 2246, 2244, 731, 135, 118, 115, 87, 45, 36, 34, 29, 21, 17, 16,
0
},
{ 7466, 4882, 4821, 4811, 2476, 886, 788, 771, 688, 531, 469, 457, 437, 400, 369, 361,
0
},
{ 9580, 5772, 5291, 5216, 3444, 1496, 1025, 928, 806, 578, 433, 384, 366, 331, 296, 273,
0
},
{ 10692, 7730, 7543, 7521, 4679, 1746, 1391, 1346, 1128, 692, 495, 458, 424, 353, 291,
268,
0
},
{ 11040, 7132, 6549, 6452, 4377, 1875, 1253, 1130, 958, 631, 431, 370, 346, 296, 253,
227,
0
},
{ 12687, 9332, 8701, 8585, 6266, 3093, 2182, 2004, 1683, 1072, 712, 608, 559, 458, 373,
323,
0
},
{ 13429, 9853, 8860, 8584, 6806, 4039, 2862, 2478, 2239, 1764, 1409, 1224, 1178, 1077, 979,
903,
0
},
{ 14685,12163,11061,10668, 9101, 6345, 4871, 4263, 3908, 3200, 2668, 2368, 2285, 2106,
1942, 1819,
0
},
{ 13295,11302,10999,10945, 7947, 5036, 4490, 4385, 3391, 2185, 1836, 1757, 1424, 998, 833,
785,
0
},
{ 4992, 2993, 2972, 2970, 1269, 575, 552, 549, 530, 505, 497, 495, 493, 489, 486, 485,
0
},
{ 15419,13862,13104,12819,11429, 8753, 7220, 6651, 6020, 4667, 3663, 3220, 2995, 2511,
2107, 1871,
```

FIG 43(5)

```
0
},
{ 12468, 9263, 8912, 8873, 5758, 2193, 1625, 1556, 1187, 589, 371, 330, 283, 200, 149,
131,
0
},
{ 15870,15076,14615,14369,13586,12034,10990,10423, 9953, 8908, 8031, 7488, 7233, 6648,
6101, 5712,
0
},
{ 1693, 978, 976, 975, 194, 18, 16, 15, 11, 7, 6, 5, 4, 3, 2, 1,
0
},
{ 7992, 5218, 5147, 5143, 2152, 366, 282, 276, 173, 59, 38, 35, 27, 16, 11, 10,
0
}
};
```

FIG 43(6)

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

FIG 44