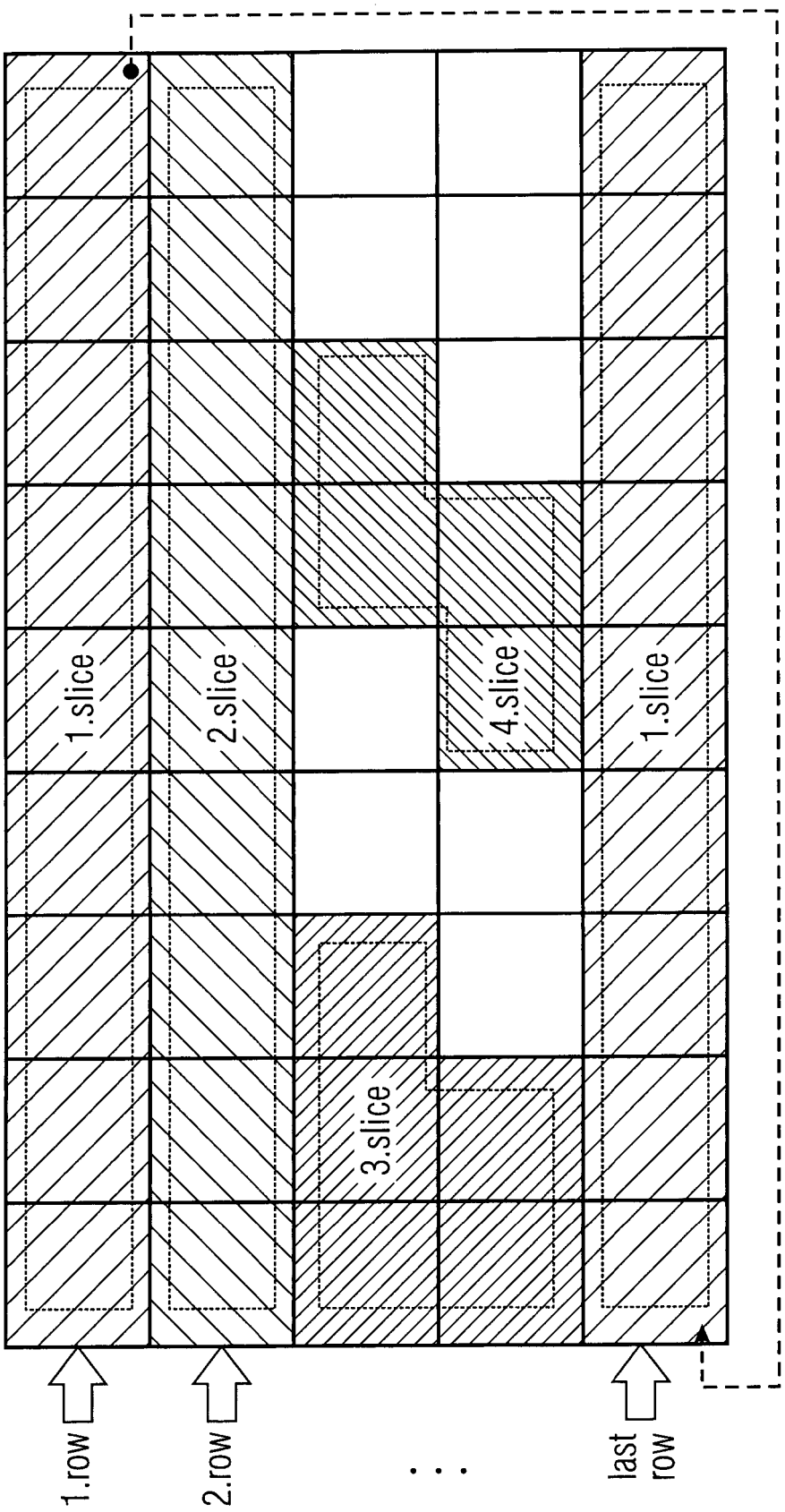

FIG 1

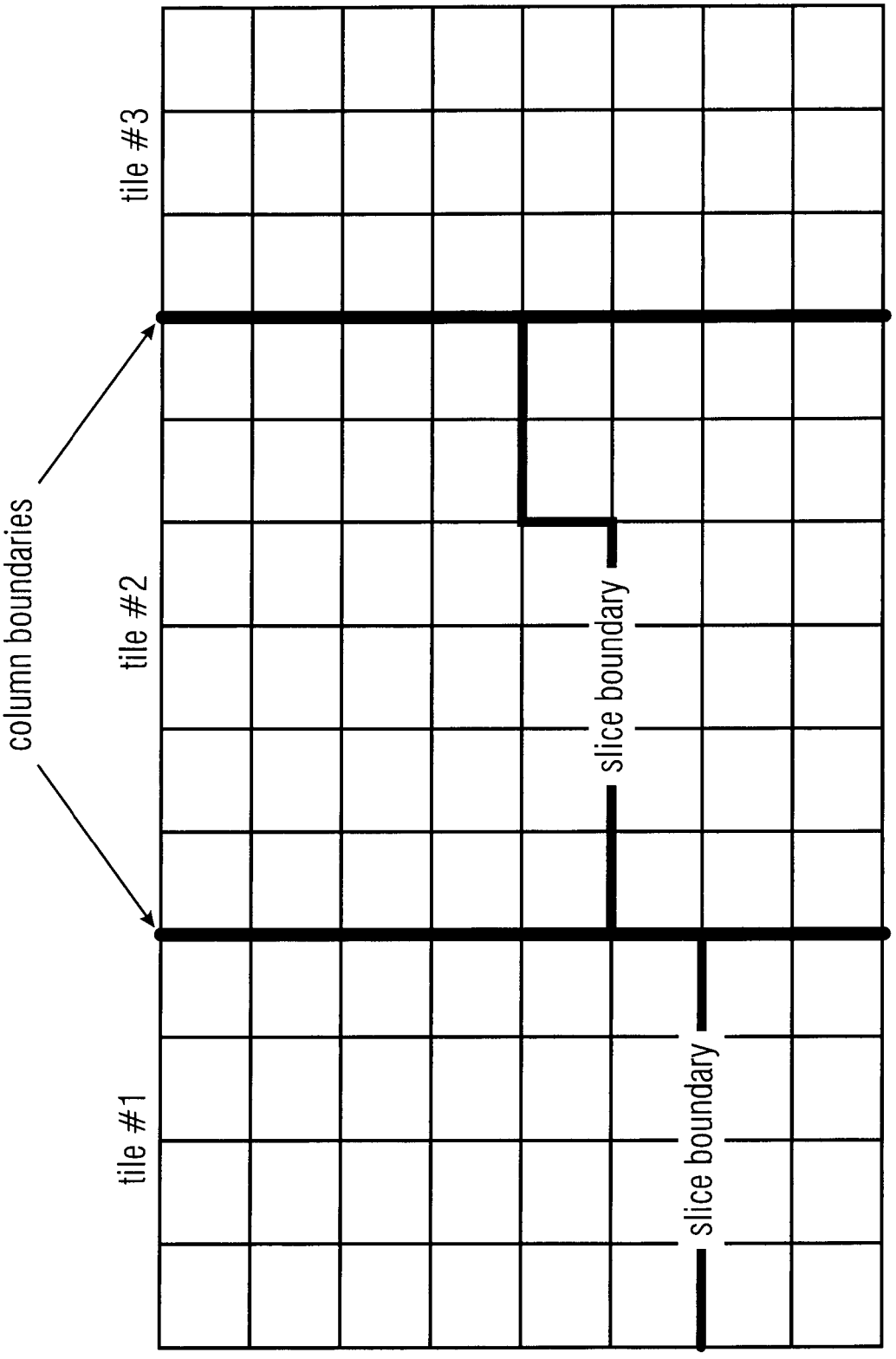
tile #3
tile #2
tile #1
column boundaries
slice boundary
slice boundary

FIG 2

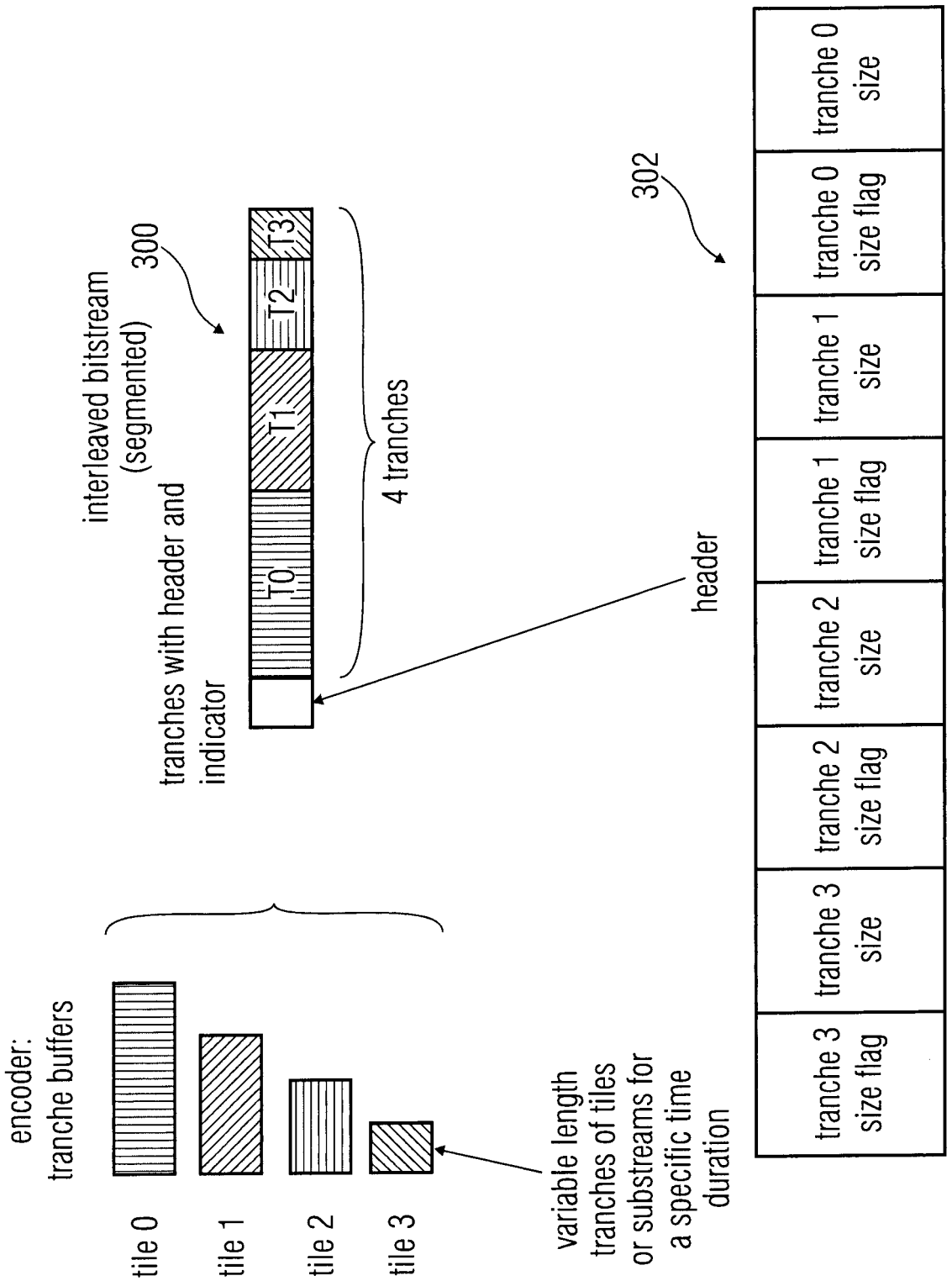
encoder:
tranche buffers
tile 0
tile 1
tile 2
tile 3
variable length
tranches of tiles
or substreams for
a specific time
duration
interleaved bitstream
(segmented)
300
tranches with header and
indicator
T0
T1
T2
T3
4 tranches
header
302
tranche 3
size flag
tranche 3
size
tranche 2
size flag
tranche 2
size
tranche 1
size flag
tranche 1
size
tranche 0
size flag
tranche 0
size

FIG 3

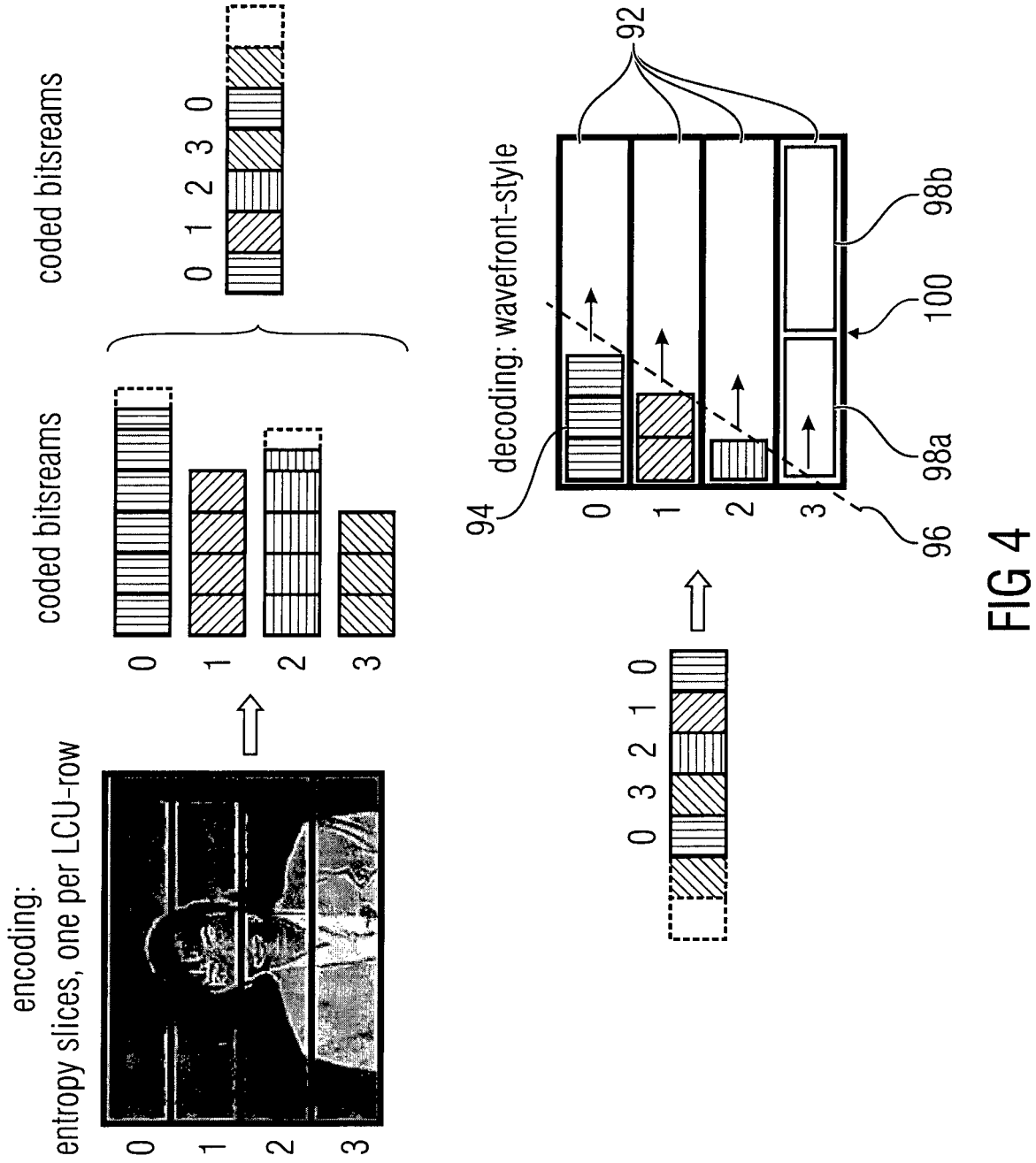
encoding:
entropy slices, one per LCU-row
0
1
2
3
coded bitsreams
0
1
2
3
coded bitsreams
0 1 2 3 0
0 3 2 1 0
decoding: wavefront-style
92
94
96
98a
98b
100
0
1
2
3

FIG 4

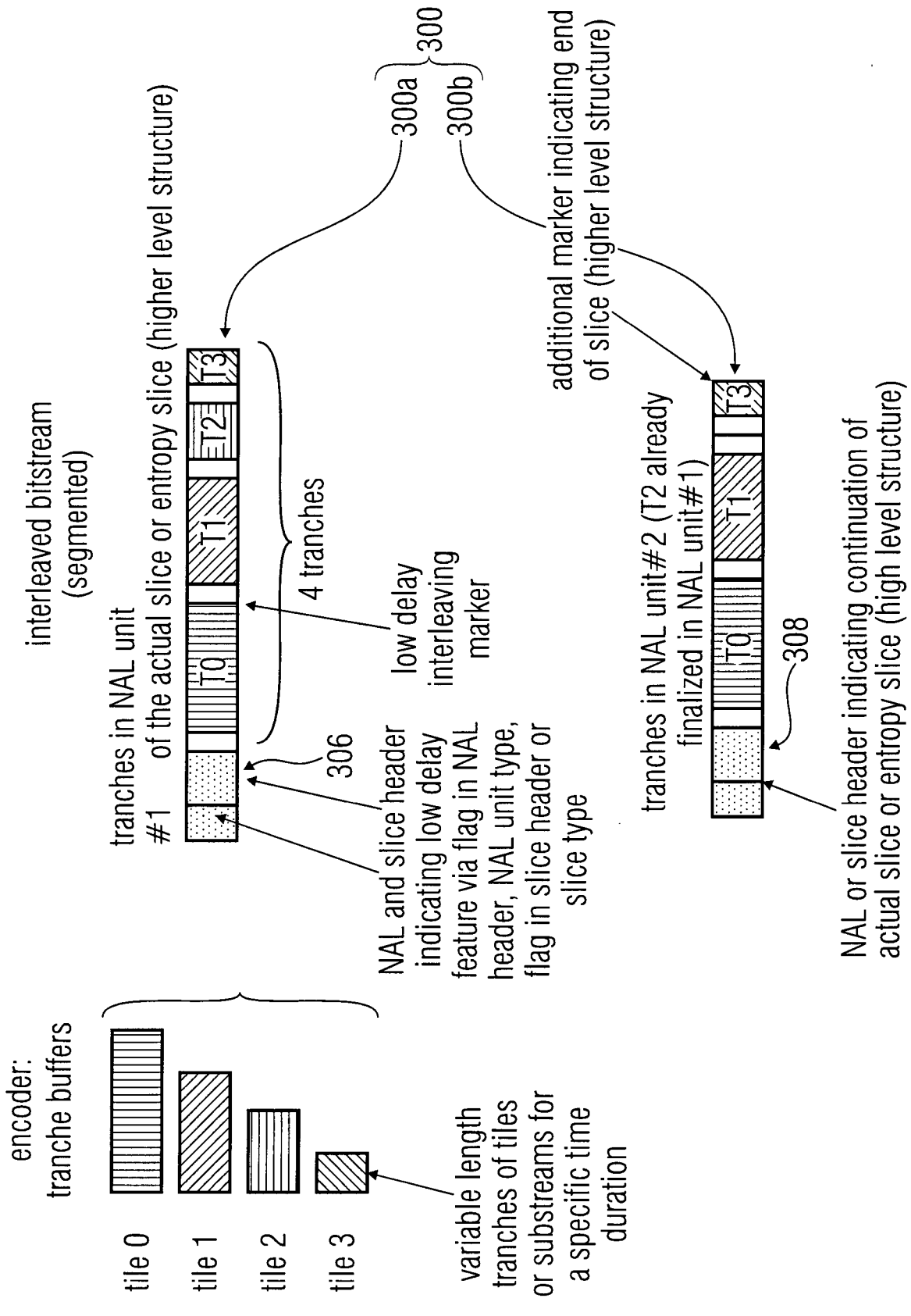
encoder:
tranche buffers
tile 0
tile 1
tile 2
tile 3
variable length tranches of tiles or substreams for a specific time duration
interleaved bitstream
(segmented)
tranches in NAL unit
#1 of the actual slice or entropy slice (higher level structure)
T0
T1
T2
T3
4 tranches
300a
300b
300
low delay interleaving marker
306
NAL and slice header indicating low delay feature via flag in NAL header, NAL unit type, flag in slice header or slice type
additional marker indicating end of slice (higher level structure)
tranches in NAL unit#2 (T2 already finalized in NAL unit#1)
T0
T1
T3
308
NAL or slice header indicating continuation of actual slice or entropy slice (high level structure)

FIG 5

nal unit(NumBytesInNALunit, num low delay tranches) {	Descriptor
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
fnal_unit_type	u(6)
NumBytesInRBSP=0	
nalUnitHeaderBytes=1	
if(nal_unit_type==1|| nal_unit_type== 4|| nal_unit_type== 5) {	
temporal_id	u(3)
output_flag	u(1)
low_delay_encapsulation_flag	u(1)
low_delay_cyclic_flag	u(1)
reserved_one_2bits	u(2)
nalUnitHeaderBytes+=1	
}	
if(low_delay_encapsulation_flag !=1)	
{	
for(i=nalUnitHeaderBytes;i<NumBytesInNALunit; i++) {	
if(i+2<NumBytesInNALunit && next bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
i+=2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++ **]**	b(8)
}	
}	
else{	
for (i=0, i++, i<num_low_delay_tranches){	
NumLDBytesInRBSP[i]=0	
}	

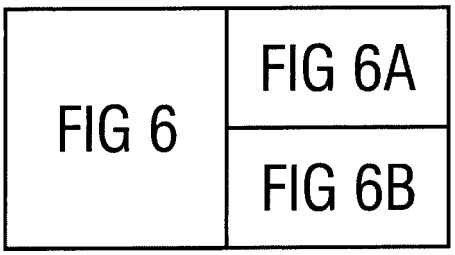

FIG 6A

```
    tranch_id=-1
    for ( i=0, i++, i<num_low_delay_tranches){
      NumBytesInRBSP[i]=0
    }
    for( i=nalUnitHeaderBytes;i<NumBytesInNALunit;i++ ) {
      if( i+2<NumBytesInNALunit && next_bits( 24 ) == 0x000002 ) {
        if( low_delay_cyclic_flag) {
          tranche_id=(tranche_id++) % num_low_delay_tranches
          i+=2
        }
        else{
          tranche_id                                              u(8)
          i+=3
        }
      }
      else
      if( i+2<NumBytesInNALunit && next_bits( 24 ) == 0x000003 ) {
        LD_rbsp_byte[ NumLDBytesInRBSP[tranche_id]++ ] [tranche_id]  b(8)
        LD_rbsp_byte[ NumLDBytesInRBSP[tranche_id]++ ] [tranche_id]  b(8)
        i+=2
        emultation_prevention_three_byte /* equal to 0x03 */          f(8)
      } else
        LD_rbsp_byte[ NumLDBytesInRBSP[tranche_id]++ ] [tranche_id]  b(8)
    }
    for ( i=0, i++, i<num_low_delay_tranches){
      for ( i=0, i++, i< NumLDBytesInRBSP[i] ){
        rbsp_byte[ NumBytesInRBSP++]=LD_rbsp_byte[i][i]
      }
    }
  }
}
```

FIG 6B

FIG 6

FIG 6A

FIG 6B

...	
low_delay_encapsulaiton_present_flag	u(1)
if(low_delay_encapsulaiton_present_flag ==1){	
low_delay_cyclic_flag	u(1)
num_low_delay_tranches_flag	u(1)
if (num_low_delay_tranches_flag==1){	
num_low_delay_tranches	ue(v)
for (i=0; i <num_low_delay_tranches; i++){	
low_delay_tranche_ lenght_minus1 [i]	ue(v)
}	
}	
}	
...	

FIG 7

low_delay_slice_layer_rbsp( ) {	**Descriptor**
slice_header( )	
tranche_id=0	
for (i=0, i++, i<num_low_delay_tranches){	
tranche_slice_data( )	
}	
rbsp_slice_trailing_bits( )	

FIG 8

	Descriptor
slice_header( ){	
entropy_slice_flag	u(1)
low_delay_slice_flag	u(1)
....	
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag==0){	
slice_adress	u(v)
if(low_delay_slice_flag==1)	
no_cabac_reset_flag	u(1)
}	
....	
if(slice_type==P || slice_type==B)	
5_minus_max_num_merge_cand	ue(v)
for(i=0; i<num_substreams_minus1+1; i++){	
substream_length_mode	u(2)
substream_length[i]	u(v)
}	
}	

FIG 9

	Descriptor
substream_marker(gradual_decoder_refresh_steps){	
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
nal_unit_type	u(6)
substream_ID	u(8)
if(gradual_decoder_refresh_steps< 1)	
is_intra	u(1)
reserved_one_7bits	u(7)
}	
}	

FIG 10

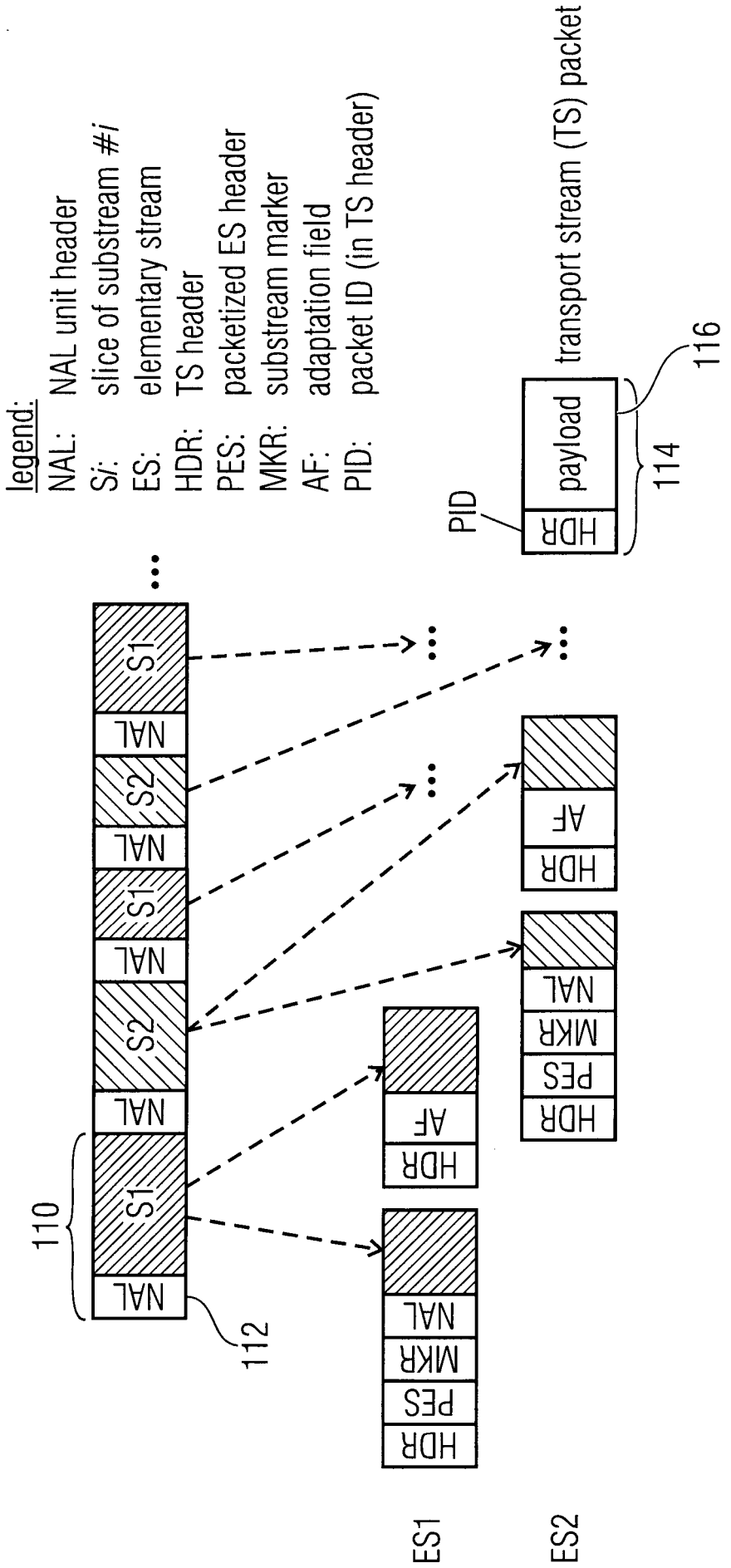
legend:
NAL: NAL unit header
Si: slice of substream #i
ES: elementary stream
HDR: TS header
PES: packetized ES header
MKR: substream marker
AF: adaptation field
PID: packet ID (in TS header)
transport stream (TS) packet
HDR
payload
PID
114
116
110
112
S1
S2
NAL
HDR
AF
PES
MKR
ES1
ES2

FIG 11

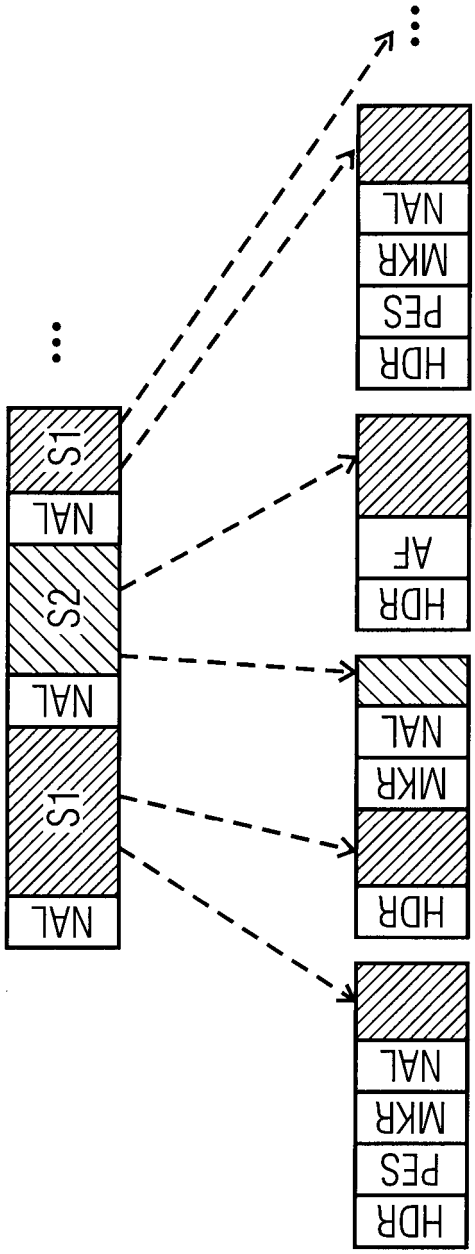
S1
NAL
S2
NAL
S1
NAL
HDR
PES
MKR
NAL
HDR
AF
NAL
MKR
HDR
NAL
MKR
PES
HDR

FIG 12

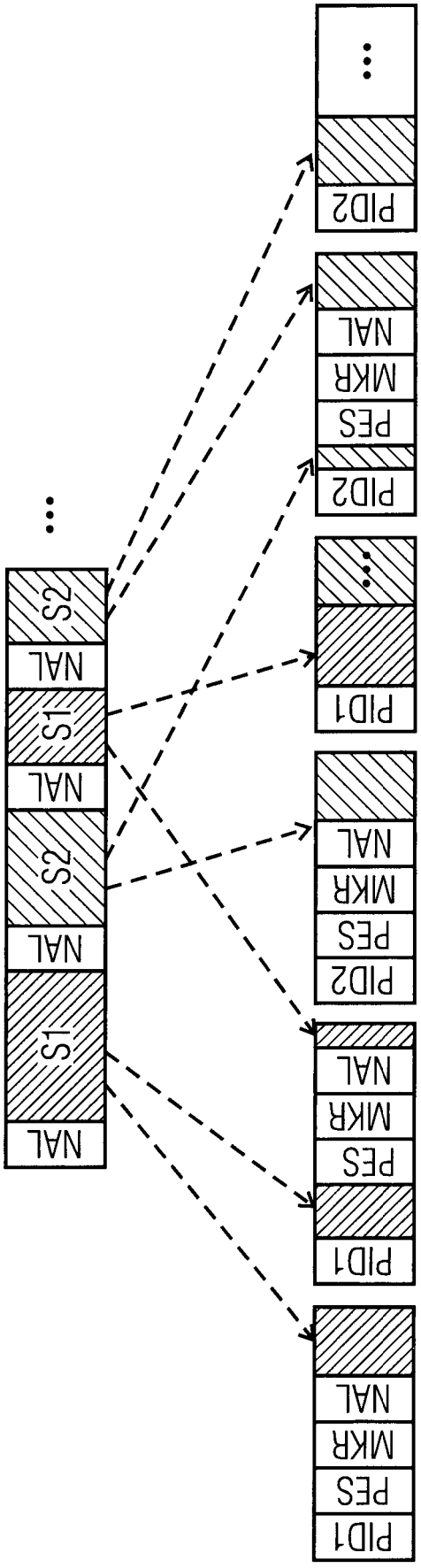
S2
NAL
S1
NAL
S2
NAL
S1
NAL
PID2
NAL
MKR
PES
PID2
PID1
NAL
MKR
PES
PID2
NAL
MKR
PES
PID1
NAL
MKR
PES
PID1

FIG 13

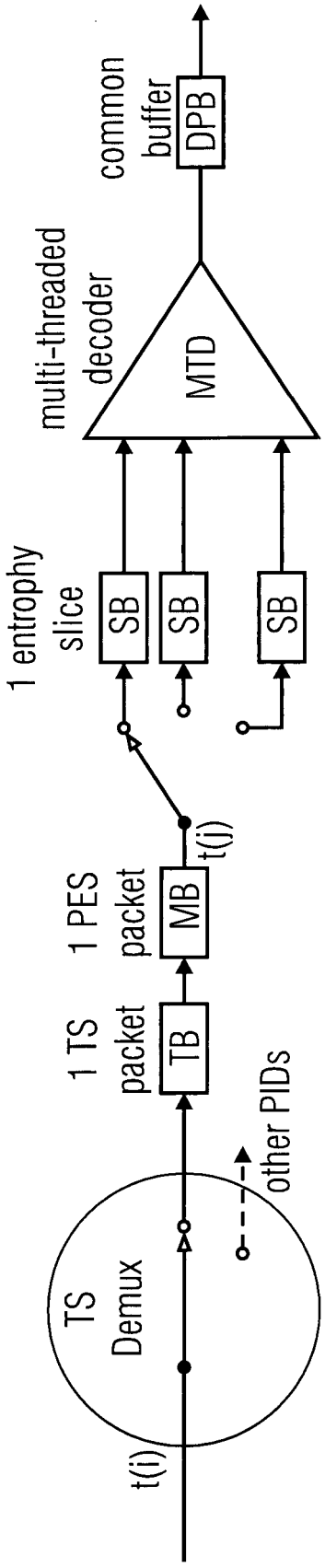
t(i)
TS
Demux
other PIDs
1 TS
packet
TB
1 PES
packet
MB
t(j)
1 entrophy
slice
SB
SB
SB
multi-threaded
decoder
MTD
common
buffer
DPB

FIG 14

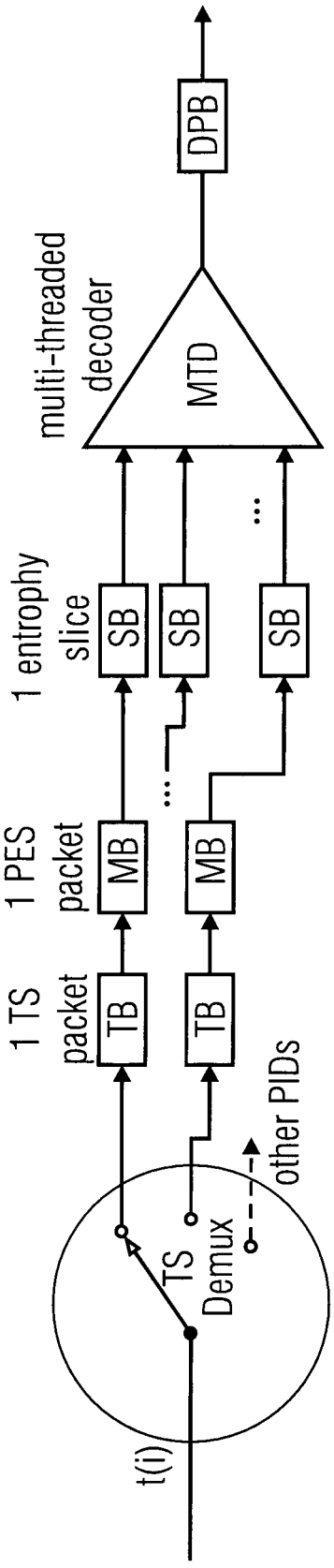
DPB
multi-threaded decoder
MTD
1 entrophy slice
SB
SB
SB
1 PES packet
MB
MB
1 TS packet
TB
TB
other PIDs
TS Demux
t(i)

FIG 15

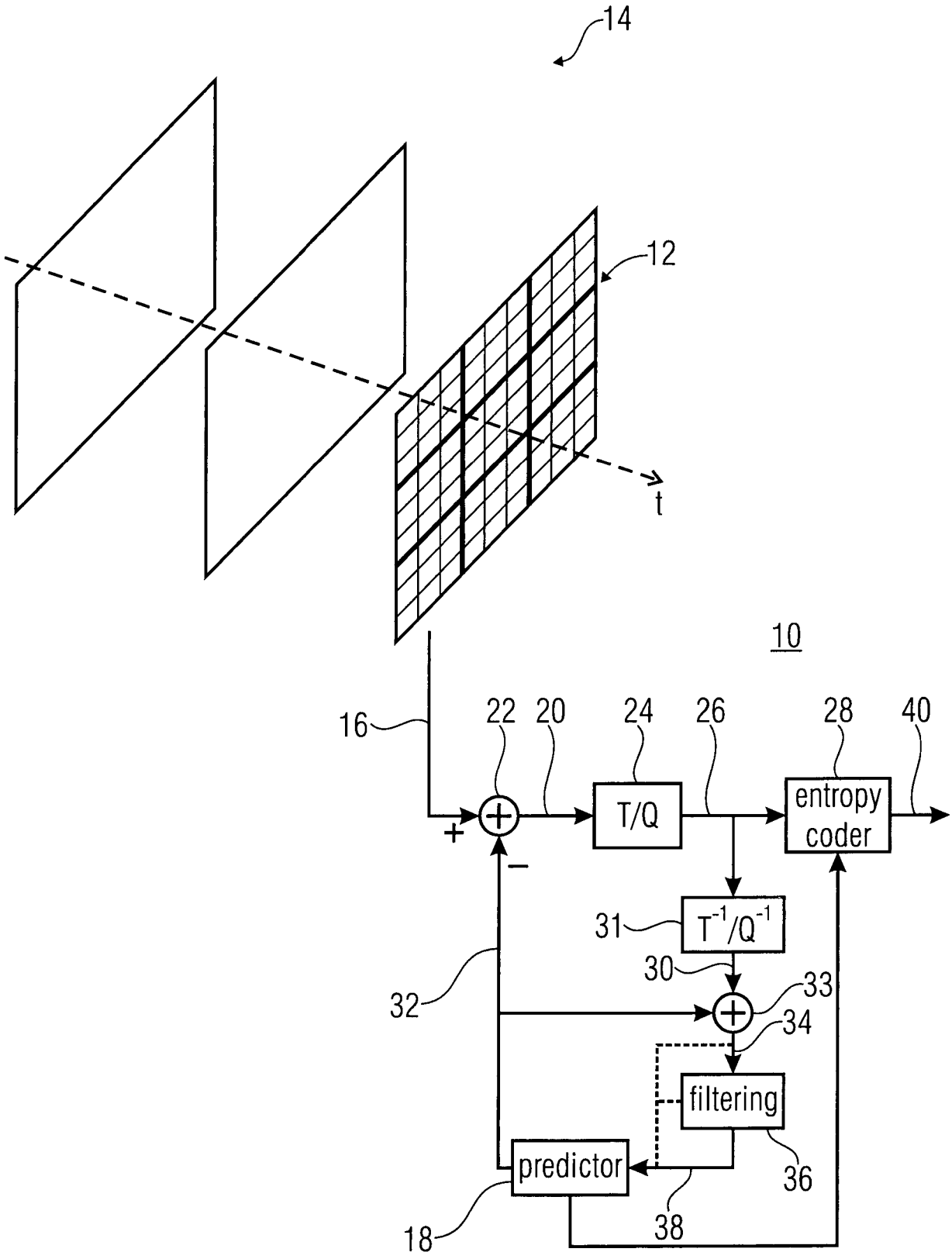
14
12
t
10
16
22
20
24
26
28
40
T/Q
entropy coder
+
−
31
T^{-1}/Q^{-1}
30
33
32
34
filtering
predictor
36
38
18

FIG 16

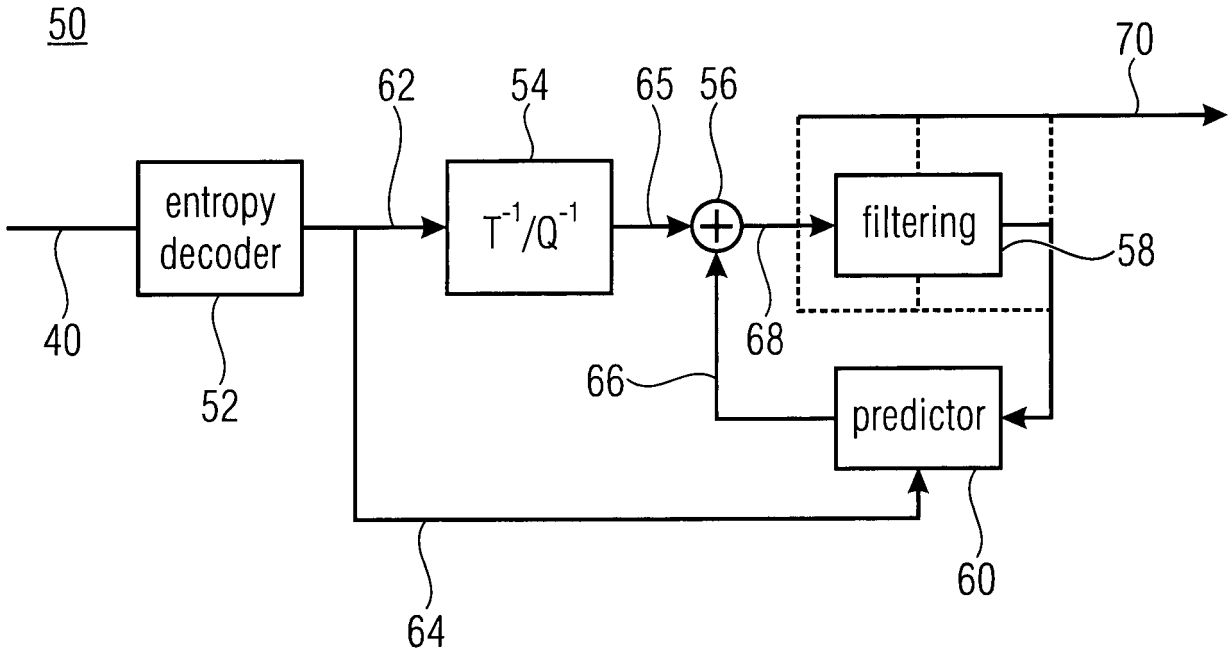
50
70
62
54
65
56
entropy decoder
T^{-1}/Q^{-1}
filtering
58
40
52
68
66
predictor
60
64

FIG 17

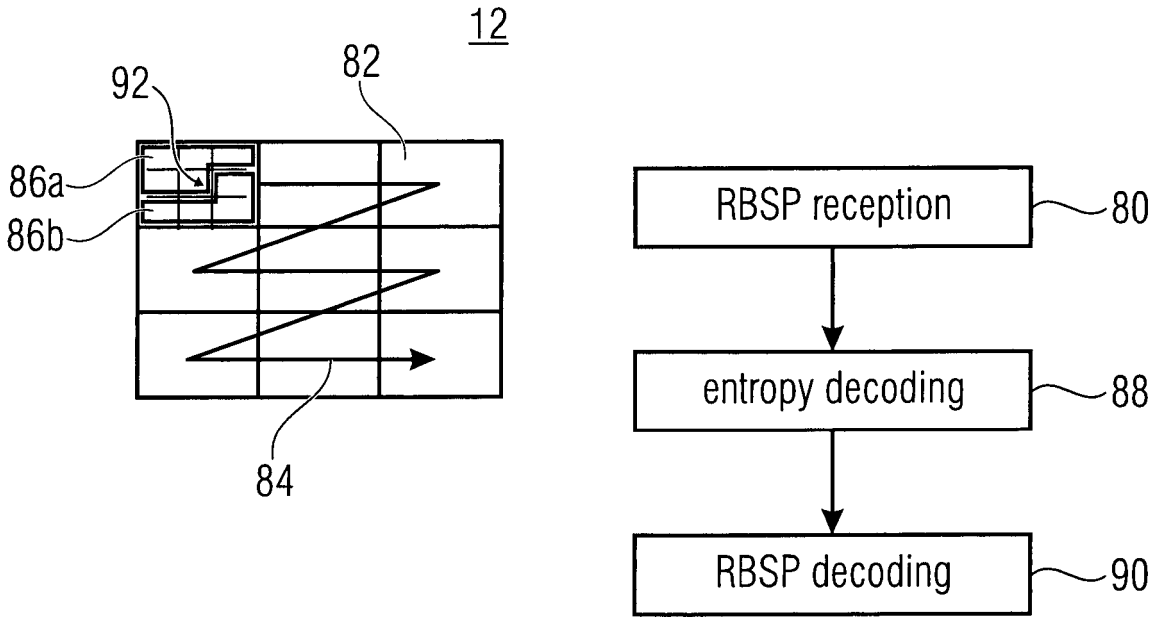
12
92
82
86a
86b
84
RBSP reception
80
entropy decoding
88
RBSP decoding
90

FIG 18

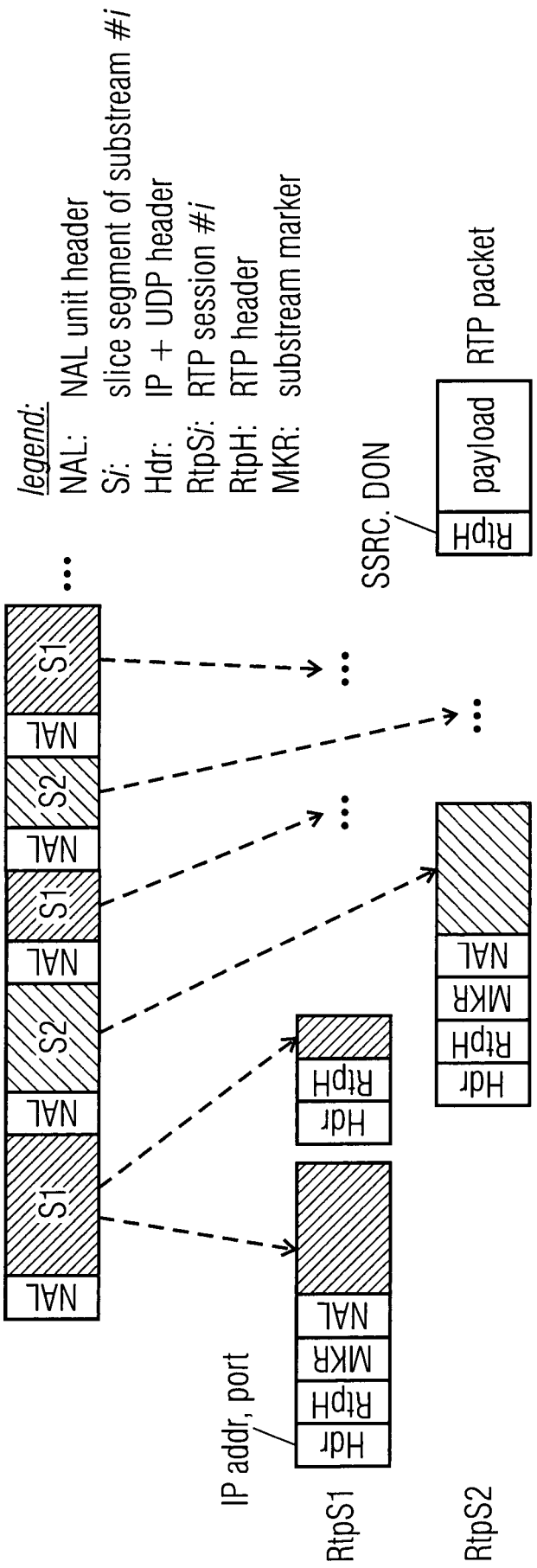
legend:
NAL: NAL unit header
Si: slice segment of substream #i
Hdr: IP + UDP header
RtpSi: RTP session #i
RtpH: RTP header
MKR: substream marker
RTP packet
payload
RtpH
SSRC. DON
S1
S2
NAL
Hdr
RtpH
MKR
IP addr, port
RtpS1
RtpS2

FIG 19